

TPC Benchmark™ H Full Disclosure Report

Sun Microsystems Sun Fire™ X4240 Server Using Sybase IQ 12.6 Single Application Server

**Submitted for Review
Report Date: May 15, 2008**

TPC Benchmark H Full Disclosure Report

First Printing

© 2004, 2005, 2006, 2007, 2008 Sun Microsystems, Inc.

901 San Antonio Road, Palo Alto, California 94303 U.S.A.

All rights reserved. This product and related documentation are protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or related documentation may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the United States Government is subject to the restrictions set forth in DFARS 252.227-7013 (c)(1)(ii) and FAR 52.227-19, Rights in Technical Data and Computer Software (October 1988).

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

TRADEMARKS

Sun, Sun Microsystems, the Sun logo, Sun Fire V440 Server, SMCC, the SMCC logo, SunSoft, the SunSoft logo, Solaris, SunOS, OpenWindows, DeskSet, ONC, and NFS are trademarks or registered trademarks of Sun Microsystems, Inc. All other product names mentioned herein are the trademarks of their respective owners.

All SPARC trademarks, including the SCD Compliant Logo, are trademarks or registered trademarks of SPARC International, Inc. SPARCstation, SPARCserver, SPARCengine, SPARCworks, and SPARCcompiler are licensed exclusively to Sun Microsystems, Inc. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK™ and Sun™ Graphical User Interfaces were developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

TPC-H Benchmark™ is a trademark of the Transaction Processing Performance Council.

Sybase IQ are registered trademarks of Sybase Inc.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.

Sun Microsystems, Inc., believes that the information in this document is accurate as of its publication date. The information in this document is subject to change without notice. Sun Microsystems, Inc., assumes no responsibility for any errors that may appear in this document.

The pricing information in this document is believed to accurately reflect prices in effect on Report Date: May 15, 2008. However, Sun Microsystems and Sybase Corporation provide no warranty on the pricing information in this document.

The performance information in this document is for guidance only. System performance is highly dependent on many factors including system hardware, system and user software, and user application characteristics. Customer applications must be carefully evaluated before estimating performance. Sun Microsystems, Inc., does not warrant or represent that a user can or will achieve a similar performance. No warranty on system performance or price/performance is expressed or implied in this document.



**Sun Fire™ X4240 Server
with Sybase IQ 12.6 Single
Application Server**

TPC-H Rev. 2.6.2

Report Date: May 15, 2008

Total System Cost

Composite Query per Hour Metric

Price/Performance

\$28132.60

8,294.1
QphH@300GB

\$ 3.39
per QphH@300GB

Database Size

Database Manager

Operating System

Other Software

Availability Date

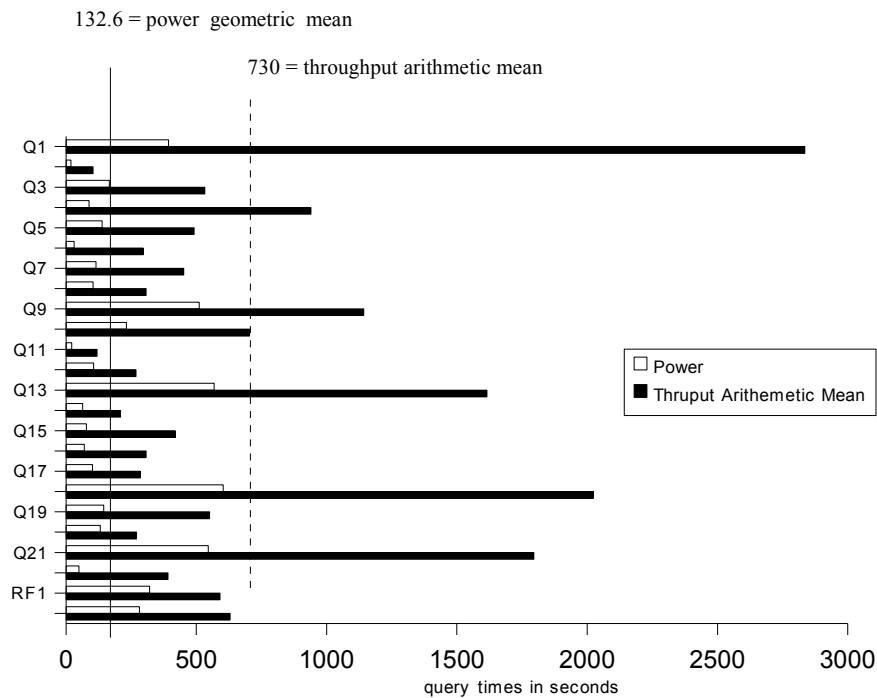
300GB

**Sybase IQ 12.6
Single Application
Server**

Solaris 10

**Solaris Volume
Manager**

May 15, 2008



Database Load Time = 5 Hrs 5 min

Load Includes Backup: N

Total Storage/Database Size= 3.6

RAID (Base tables): RAID 1

RAID (Base tables and auxiliary data structures): RAID 1

RAID (All): N

System Configuration:

SunFire X4240 Server each with
2 Opteron 3.2 GHz Dual Core processors
16 GB memory
16 X 73GB internal disks (10K RPM)

Total Storage: 1087.8 GB

(in this calculation 1 GB is defined as 1024 * 1024 * 1024)



**Sun Fire™ X4240 Server
with Sybase IQ 12.6 Single
Application Server**

TPC-H Rev. 2.6.2

Report Date: May 15, 2008

Description	Part Number	Source	Unit Price	Qty	Ext. Price	3 Yr. maint.
Server Hardware						
SunFire X4240, 2x3.2 GHz CPUs, 16GB memory, 16 x 73GB disks			10,648.00	1	10,648.00	
3 Year Gold Warranty Upgrade – X4140 Server				1		2,157.11
Sun Server Discount					1,064.80	215.71
Server Hardware Subtotal					9,583.20	1,941.40
External Storage						
None						
Server Software						
Sybase IQ Single App Svr – per cpu core	11467		2,595.00	4	10,380.00	
Sybase IQ 3 Years Extended Support 24 x 7	98477		1,557	4		6,228.00
Sybase discount					0.00	
Server Software Subtotal					10,380.00	6,228.00
Total					19,963.20	8,169.40
3 Yr. Cost					28,132.60	
QpH@ 100GB					8,294.1	
\$/QpH@ 100GB					3.39	

Service for Sun products is from Sun Microsystems, Inc.
Service for Sybase products is from Sybase Inc.

Notes (Source):

1. Sun Microsystems Inc.
 2. Sybase Inc.
- PriceQuotes provided in Appendix G

Audited by: Brad Askins, InfoSizing, Inc. (www.sizing.com)

Prices used in TPC benchmarks reflect the actual prices a customer would pay for a one-time purchase of the standard components. Individually negotiated discounts are not permitted. Special prices based on assumptions about past or future purchase are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details, see the pricing sections of the TPC benchmark specifications. If you find that the stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org. Thank you.



**Sun Fire™ X4240 Server
with Sybase IQ 12.6 Single
Application Server**

TPC-H Rev. 2.6.2

Report Date: May 15, 2008

Numerical Quantities

Measurement Results:

Database Scale Factor	= 300GB
Total Data Storage / Database Size	= 3.6
Start of database load time	= 01-15-2008 14:18:32
End of database load time	= 01-15-2008 19:24:26
Database Load Time	= 5 Hrs 5 min
Query Streams for Throughput Test	= 6
TPC-H Power	= 8,144.9
TPC-H Throughput	= 8,446.0
TPC-H Composite Query-per-Hour Rating (QphH@300GB)	= 8,294.1
Total System Price Over 3 Years (\$US)	= 28132.60
TPC-H Price/Performance Metric (\$/QphH@300GB)	= 3.39

Measurement Intervals:

Measurement Interval in Throughput Test (Ts)	= 16,879 seconds
--	------------------

Duration of Stream Execution:

Stream ID	Seed	Start Date	Start Time	End Date	End Time	Duration
Stream 00	115192426	16-Jan-08	1:24:52	16-Jan-08	2:41:40	1:16:48
Stream 01	115192427	16-Jan-08	2:46:24	16-Jan-08	7:16:14	4:29:50
Stream 02	115192428	16-Jan-08	2:46:24	16-Jan-08	7:27:43	4:41:19
Stream 03	115192429	16-Jan-08	2:46:24	16-Jan-08	7:17:53	4:31:29
Stream 04	115192430	16-Jan-08	2:46:24	16-Jan-08	6:58:11	4:11:47
Stream 05	115192431	16-Jan-08	2:46:24	16-Jan-08	7:13:07	4:26:43
Stream 06	115192432	16-Jan-08	2:46:24	16-Jan-08	7:11:41	4:25:17
Refresh		16-Jan-08	2:46:24	16-Jan-08	5:18:38	2:32:14



**Sun Fire™ X4240 Server
with Sybase IQ 12.6 Single
Application Server**

TPC-H Rev. 2.6.2

Report Date: May 15, 2008

TPC-H Timing Intervals (in seconds)

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Stream 00	393.6	18.6	166.1	88.2	139.2	31	115	104.2	511.3	232.2	21.8	105.6
Stream 01	2606.6	276.7	937.9	161.7	414	258.4	347.7	140	1153.4	472.9	120.9	169.3
Stream 02	3406.9	115	290.5	729.4	582.1	158	531	453.8	1719.8	756	21.9	403.1
Stream 03	3375	96.3	272.3	1207.6	640.4	323.8	535.1	507	1176.6	576.5	54.5	173.9
Stream 04	2230.7	71.2	464.9	972.2	691.5	90.1	487.4	311.4	1245.3	659.6	222.5	470.7
Stream 05	2498.4	26.5	621.8	1349	372.3	85	341.3	191	683.6	586.1	197.8	212.9
Stream 06	2888.8	34.9	606.4	1219.5	246.8	862	467	240.5	876.2	1176.5	96	180
Minimum	2230.70	26.50	272.30	161.70	246.80	85.00	341.30	140.00	683.60	472.90	21.90	169.30
Average	2834.40	103.43	532.30	939.90	491.18	296.22	451.58	307.28	1142.48	704.60	118.93	268.32
Maximum	3406.90	276.70	937.90	1349.00	691.50	862.00	535.10	507.00	1719.80	1176.50	222.50	470.70



**Sun Fire™ X4240 Server
with Sybase IQ 12.6 Single
Application Server**

TPC-H Rev. 2.6.2

Report Date: May 15, 2008

	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream 00	568.7	63.8	78.2	70	101.6	603.4	145.2	131.8	546	50	321.2	281.8
Stream 01	1529	168.6	472.1	181.2	510.5	2207.1	535.8	451.1	2609.1	464.7	601.8	561.3
Stream 02	1891.8	530.1	407.3	462.7	422.8	2099.3	586.8	235.5	544.1	529.1	620.1	517.8
Stream 03	1850.7	192.3	273.6	185.7	223.6	2019.4	468.6	220.8	1505.5	408.8	536.8	633.7
Stream 04	1385.8	110.2	478.1	200.9	207.7	1664.7	663.5	226.3	2079.9	166	482.7	865.9
Stream 05	1376.2	117.4	633.2	585.9	215	2164.7	533	278.6	2710.4	206.1	700.4	601.8
Stream 06	1659.6	135.7	251.6	230.1	130.3	1988	515.5	214.8	1325	569.6	604.4	598.1
Minimum	1376.20	110.20	251.60	181.20	130.30	1664.70	468.60	214.80	544.10	166.00	482.70	517.80
Average	1615.52	209.05	419.32	307.75	284.98	2023.87	550.53	271.18	1795.67	390.72	591.03	629.77
Maximum	1891.80	530.10	633.20	585.90	510.50	2207.10	663.50	451.10	2710.40	569.60	700.40	865.90

Table of Contents

1. General Items.....	13
1.1 Benchmark Sponsor.....	13
1.2 Parameter Settings.....	13
1.3 Configuration Diagram.....	14
2. Clause 1 Logical Database Design.....	15
2.1 Database Definition Statements.....	15
2.2 Physical Organization.....	15
2.3 Horizontal Partitioning.....	15
2.4 Replication.....	15
3. Clause 2 Queries and Refresh Functions.....	16
3.1 Query Language.....	16
3.2 Verifying Method for Random Number Generation.....	16
3.3 Generating Values for Substitution Parameters.....	16
3.4 Query Text and Output Data from Qualification Database.....	16
3.5 Query Substitution Parameters and Seeds Used.....	16
3.6 Query Isolation Level.....	17
3.7 Source Code of Refresh Functions.....	17
4. Clause 3 Database System Properties.....	18
4.1 ACID Properties.....	18
4.2 Atomicity.....	18
4.2.1 Completed Transaction.....	18
4.2.2 Aborted Transaction.....	18
4.3 Consistency.....	18
4.3.1 Consistency Test.....	19
4.4 Isolation.....	19
4.4.1 Read-Write Conflict with Commit.....	19
4.4.2 Read-Write Conflict with Rollback.....	19
4.4.3 Write-Write Conflict with Commit.....	19
4.4.4 Write-Write Conflict with Rollback.....	20
4.4.5 Concurrent Progress of Read and Write Transactions.....	20
4.4.6 Read-Only Query Conflict with Update Transaction.....	20
4.5 Durability.....	20
4.5.1 Failure of a Durable Medium.....	21
4.5.2 System Crash.....	21

4.5.3 Memory Failure.....	21
5. Clause 4 Scaling and Database Population.....	22
5.1 Ending Cardinality of Tables.....	22
5.2 Distribution of Tables and Logs Across Media.....	22
5.3 Database partition/replication mapping.....	23
5.4 RAID Feature.....	24
5.5 Modifications to the DBGEN.....	24
5.6 Database Load Time.....	24
5.7 Data Storage Ratio.....	24
5.8 Database Load Mechanism Details and Illustration.....	25
5.9 Qualification Database Configuration.....	25
6. Clause 5 Performance Metrics and Execution Rules.....	26
6.1 System Activity Between Load and Performance Tests.....	26
6.2 Steps in the Power Test.....	26
6.3 Timing Intervals for Each Query and Refresh Functions.....	26
6.4 Number of Streams for the Throughput Test.....	26
6.5 Start and End Date/Times for Each Query Stream.....	26
6.6 Total Elapsed Time of the Measurement Interval.....	26
6.7 Refresh Function Start Date/Time and Finish Date/Time.....	27
6.8 Timing Intervals for Each Query and Each Refresh Function for Each Stream.....	28
6.9 Performance Metrics.....	28
6.10 The Performance Metric and Numerical Quantities from Both Runs.....	28
6.11 System Activity Between Performance Tests.....	28
7. Clause 6 SUT and Driver Implementation.....	29
7.1 Driver.....	29
7.2 Implementation-Specific Layer.....	29
7.3 Profile-Directed Optimization.....	29
8. Clause 7 Pricing.....	30
8.1 Hardware and Software Used.....	30
8.2 Total Three Year Price.....	30
8.3 Availability Date.....	30
9. Auditor's Information and Attestation Letter.....	31
Appendix A. Solaris 10 and Sybase IQ 12.6 Parameters.....	32
Appendix C. Query Text and Query Output.....	98
Appendix E. Implementation-Specific Layer/Driver Code.....	108
Appendix F. Misc database scripts.....	110
Appendix G. Pricing information.....	112

Benchmark Sponsors:	Brad Carlile Director, Enterprise Benchmarking Sun Microsystems, Inc. 8305 S. W. Creekside Place Beaverton, OR 97008	Sumit Kundu Marketing Director Sybase, Inc. 1 Sybase Drive Dublin, CA 94568
---------------------	--	---

April 1, 2008

I verified the TPC Benchmark™ H performance of the following configuration:

Platform: **Sun Fire V4240 Server**
 Database Manager: **Sybase IQ 12.6 Single Application Server**
 Operating System: **Solaris 10**

The results were:

CPU (Speed)	Memory	Disks	QphH@300GB
Sun Fire V4240 with:			
2 x AMD Opteron Dual Core (3.2 GHz)	16 GB Main	16 x 73 GB (internal)	8294.1

In my opinion, this performance result was produced in compliance with the TPC's requirements for the benchmark. The following verification items were given special attention:

- The database records were defined with the proper layout and size
- The database population was generated using DBGEN
- The database was properly scaled to 300GB and populated accordingly
- The compliance of the database auxiliary data structures was verified
- The database load time was correctly measured and reported

- The required ACID properties were verified and met
- The query input variables were generated by QGEN
- The query text was produced using minor modifications
- The execution of the queries against the SF1 database produced compliant answers
- A compliant implementation specific layer was used to drive the tests
- The throughput tests involved 6 query streams
- The ratio between the longest and the shortest query was such that no query timing was adjusted
- The execution times for queries and refresh functions were correctly measured and reported
- The repeatability of the measured results was verified
- The required amount of database log was configured
- The system pricing was verified for major components and maintenance
- The major pages from the FDR were verified for accuracy

Additional Audit Notes:

None.

Respectfully Yours,

The image shows two handwritten signatures in black ink. The signature on the left is 'Francois Rabb' and the signature on the right is 'Bradley J. Askins'. Both signatures are written in a cursive, flowing style.

Francois Rabb, President

Bradley J. Askins, Auditor

TPC Benchmark H Overview

The TPC Benchmark™ H (TPC-H) is a Decision Support benchmark. It is a suite of business-oriented ad-hoc queries and concurrent modifications. The queries and the data populating the database have been chosen to have broad industry-wide relevance while maintaining a sufficient degree of ease of implementation. This benchmark illustrates Decision Support systems that:

- Examine large volumes of data
- Execute queries with a high degree of complexity
- Give answers to critical business questions

TPC-H evaluates the performance of various Decision Support systems by the execution of sets of queries against a standard database under controlled conditions. The TPC-H queries:

- Give answers to real-world business questions
- Simulate generated ad-hoc queries
- Are far more complex than most OLTP transactions
- Include a rich breadth of operators and selectivity constraints
- Generate intensive activity on the part of the database server component of the system under test
- Are executed against a database complying to specific population and scaling requirements
- Are implemented with constraints derived from staying closely synchronized with an on-line production database

1. General Items

1.1 Benchmark Sponsor

A statement identifying the benchmark sponsor(s) and other participating companies must be provided.

Sun Microsystems, Inc. and Sybase Inc. are the sponsors of this TPC-H benchmark.

1.2 Parameter Settings

Settings must be provided for all customer-tunable parameters and options that have been changed from the defaults found in actual products, including but not limited to:

- *Database Tuning Options*
- *Optimizer/Query execution options*
- *Query processing tool/language configuration parameters*
- *Recovery/commit options*
- *Consistency/locking options*
- *Operating system and configuration parameters*
- *Configuration parameters and options for any other software component incorporated into the pricing structure*
- *Compiler optimization options*

Appendix A contains the Solaris and Sybase IQ parameters used in this benchmark.

1.3 Configuration Diagram

Provide diagrams of both the measured and priced configurations, accompanied by a description of the differences.

Configuration (Priced and Measured are identical):



SUN Fire X4240 Server

2 X 3.2 GHz Opteron Dual Core processors

16 GB Memory

16 X 73 GB 10K RPM internal disks

No External Storage

2. Clause 1 Logical Database Design

2.1 Database Definition Statements

Listings must be provided for all table definition statements and all other statements used to set up the test and qualification databases.

Appendix B contains the programs and scripts that create and analyze the tables and indexes for the TPC-H database.

2.2 Physical Organization

The physical organization of tables and indices within the test and qualification databases must be disclosed. If the column ordering of any table is different from that specified in Clause 1.4, it must be noted.

No record clustering or index clustering was used. Column ordering was changed for some tables. Refer to the table create statements in Appendix B for further details.

2.3 Horizontal Partitioning

Horizontal partitioning of tables and rows in the test and qualification databases (see Clause 1.5.4) must be disclosed.

Horizontal partitioning was not used for any of the tables.

2.4 Replication

Any replication of physical objects must be disclosed and must conform to the requirements of Clause 1.5.6.

No replication was used.

3. Clause 2 Queries and Refresh Functions

3.1 Query Language

The query language used to implement the queries must be identified.

SQL was the query language used to implement all queries.

3.2 Verifying Method for Random Number Generation

The method of verification for the random number generation must be described unless the supplied DBGEN and QGEN were used.

TPC supplied versions 2.6.2 of DBGEN and QGEN were used for this TPC-H benchmark.

3.3 Generating Values for Substitution Parameters

The method used to generate values for substitution parameters must be disclosed. If QGEN is not used for this purpose, then the source code of any non-commercial tool used must be disclosed. If QGEN is used, the version number, release number, modification number, and patch level of QGEN must be disclosed.

The supplied QGEN version 2.6.2 was used to generate the substitution parameters.

3.4 Query Text and Output Data from Qualification Database

The executable query text used for query validation must be disclosed along with the corresponding output data generated during the execution of the query text against the qualification database. If minor modifications (see Clause 2.2.3) have been applied to any functional query definitions or approved variants in order to obtain executable query text, these modifications must be disclosed and justified. The justification for a particular minor query modification can apply collectively to all queries for which it has been used. The output data for the power and throughput tests must be made available electronically upon request.

Appendix C contains the query text and query output. The standard queries were used throughout with the following modifications:

- In Q1, Q4, Q5, Q6, Q10, Q12, Q14, Q15 and Q20, the "dateadd" function is used to perform date arithmetic.
- In Q7, Q8 and Q9, the "datepart" function is used to extract part of a date (e.g., "year").
- In Q2, Q3, Q10, Q18 and Q21, the "top" function is used to restrict the number of output rows.
- The semicolon (;) is used as a command delimiter.

3.5 Query Substitution Parameters and Seeds Used

The query substitution parameters used for all performance tests must be disclosed in tabular format, along with the seeds used to generate these parameters.

Appendix D contains the seed and query substitution parameters.

3.6 Query Isolation Level

The isolation level used to run the queries must be disclosed. If the isolation level does not map closely to the levels defined in Clause 3.4, additional descriptive detail must be provided.

The queries and transactions were run with isolation level 3 (repeatable read).

3.7 Source Code of Refresh Functions

The details of how the refresh functions were implemented must be disclosed (including source code of any non-commercial program used).

Appendix B contains the source code for the refresh functions.

4. Clause 3 Database System Properties

4.1 ACID Properties

The ACID (Atomicity, Consistency, Isolation and Durability) properties of transaction processing systems must be supported by the system under test during the timed portion of this benchmark. Since TPC-H is not a transaction processing benchmark, the ACID properties must be evaluated outside the timed portion of the test.

Source code for the ACID test is included in Appendix B.

4.2 Atomicity

The system under test must guarantee that transactions are atomic; the system will either perform all individual operations on the data, or will assure that no partially-completed operations leave any effects on the data.

4.2.1 Completed Transaction

Perform the ACID Transaction for a randomly selected set of input data and verify that the appropriate rows have been changed in the ORDERS, LINEITEM, and HISTORY tables

1. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for a randomly selected order key.
2. The ACID Transaction was performed using the order key from step 1.
3. The ACID Transaction committed.
4. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for the same order key. It was verified that the appropriate rows had been changed.

4.2.2 Aborted Transaction

Perform the ACID Transaction for a randomly selected set of input data, substituting a ROLLBACK of the transaction for the COMMIT of the transaction. Verify that the appropriate rows have not been changed in the ORDERS, LINEITEM, and HISTORY tables.

1. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for a randomly selected order key.
2. The ACID Transaction was performed using the order key from step 1. The transaction was stopped prior to the commit.
3. The ACID Transaction was ROLLED BACK.
4. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for the same order key. It was verified that the appropriate rows had not been changed.

4.3 Consistency

Consistency is the property of the application that requires any execution of transactions to take the database from one consistent state to another.

4.3.1 Consistency Test

Verify that ORDERS and LINEITEM tables are initially consistent, submit the prescribed number of ACID Transactions with randomly selected input parameters, and re-verify the consistency of the ORDERS and LINEITEM.

1. The consistency of the ORDERS and LINEITEM tables was verified based on a sample of order keys.
2. 100 ACID Transactions were submitted by each of twenty one execution streams.
3. The consistency of the ORDERS and LINEITEM tables was re-verified.

4.4 Isolation

Operations of concurrent transactions must yield results which are indistinguishable from the results which would be obtained by forcing each transaction to be serially executed to completion in the proper order.

4.4.1 Read-Write Conflict with Commit

Demonstrate isolation for the read-write conflict of a read-write transaction and a read-only transaction when the read-write transaction is committed.

1. An ACID Transaction was started for a randomly selected O_KEY, L_KEY, and DELTA. The ACID Transaction was suspended prior to COMMIT.
2. An ACID Query was started for the same O_KEY used in step 1.
3. The ACID Transaction was resumed and COMMITTED.
4. The ACID Query completed. It returned the data as committed by the ACID Transaction.

4.4.2 Read-Write Conflict with Rollback

Demonstrate isolation for the read-write conflict of a read-write transaction and a read-only transaction when the read-write transaction is rolled back.

1. An ACID Transaction was started for a randomly selected O_KEY, L_KEY, and DELTA. The ACID Transaction was suspended prior to ROLLBACK.
2. An ACID Query was started for the same O_KEY used in step 1. The ACID Query did not see the uncommitted changes made by the ACID Transaction.
3. The ACID Transaction was ROLLED BACK.
4. The ACID Query completed.

4.4.3 Write-Write Conflict with Commit

Demonstrate isolation for the write-write conflict of two update transactions when the first transaction is committed.

1. An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. T1 was suspended prior to COMMIT.
2. Another ACID Transaction, T2, was started using the same O_KEY and L_KEY and a randomly selected DELTA.
3. T2 waited.
4. T1 was allowed to COMMIT and T2 completed.

-
5. It was verified that $T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE + (\Delta T1*(T1.L_EXTENDEDPRICE/T1.L_QUANTITY))$

4.4.4 Write-Write Conflict with Rollback

Demonstrate isolation for the write-write conflict of two update transactions when the first transaction is rolled back.

1. An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. T1 was suspended prior to ROLLBACK.
2. Another ACID Transaction, T2, was started using the same O_KEY and L_KEY and a randomly selected DELTA.
3. T2 waited.
4. T1 was allowed to ROLLBACK and T2 completed.
5. It was verified that $T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE$.

4.4.5 Concurrent Progress of Read and Write Transactions

Demonstrate the ability of read and write transactions affecting different database tables to make progress concurrently.

1. An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. T1 was suspended prior to ROLLBACK.
2. Another Transaction, T2, was started which did the following:

For random values of PS_PARTKEY and PS_SUPPKEY, all columns of the PARTSUPP table for which PS_PARTKEY and PS_SUPPKEY are equal, are returned.
3. T2 completed.
4. T1 was allowed to COMMIT.
5. It was verified that appropriate rows in ORDERS, LINEITEM and HISTORY tables were changed.

4.4.6 Read-Only Query Conflict with Update Transaction

Demonstrate that the continuous submission of arbitrary (read-only) queries against one or more tables of the database does not indefinitely delay update transactions affecting those tables from making progress.

1. A Transaction, T1, executing Q1 against the qualification database, was started using a randomly selected DELTA.
2. An ACID Transaction T2, was started for a randomly selected O_KEY, L_KEY and DELTA.
3. T2 completed and appropriate rows in the ORDERS, LINEITEM and HISTORY tables had been changed.
4. Transaction T1 completed executing Q1.

4.5 Durability

The SUT must guarantee durability: the ability to preserve the effects of committed transactions and insure database consistency after recovery from any one of the failures listed in Clause 3.5.3.

4.5.1 Failure of a Durable Medium

Guarantee the database and committed updates are preserved across a permanent irrecoverable failure of any single durable medium containing TPC-H database tables or recovery log tables.

All disks containing TPC-H tables, indexes and the catalog file are on RAID1 volumes. When one of these disks was removed from the server during the durability test, Sybase IQ continued to run as if nothing happened until the (non-mirrored) IQ temp segment on that disk was referenced. At that time IQ crashed, as expected. Power to the X4240 server server was then cut off. The outcome is described in section 4.5.2.

4.5.2 System Crash

Guarantee the database and committed updates are preserved across an instantaneous interruption (system crash/system hang) in processing which requires the system to reboot to recover.

The system crash and memory failure tests were combined by cutting off power to the X4240 server during the durability test. When power was restored, the system rebooted and the database was restarted. The durability success file and the HISTORY table were compared successfully.

4.5.3 Memory Failure

Guarantee the database and committed updates are preserved across failure of all or part of memory (loss of contents).

See section 4.5.2.

5. Clause 4 Scaling and Database Population

5.1 Ending Cardinality of Tables

The cardinality (i.e., the number of rows) of each table of the test database, as it existed at the completion of the database load (see clause 4.2.5) must be disclosed.

<i>Table</i>	<i>Rows</i>
<i>Lineitem</i>	<i>1,799,989,091</i>
<i>orders</i>	<i>450,000,000</i>
<i>partsupp</i>	<i>240,000,000</i>
<i>Part</i>	<i>60,000,000</i>
<i>Customer</i>	<i>45,000,000</i>
<i>Supplier</i>	<i>3,000,000</i>
<i>Nation</i>	<i>25</i>
<i>Region</i>	<i>5</i>

5.2 Distribution of Tables and Logs Across Media

The distribution of tables and logs across all media must be explicitly described.

- All tables and indexes were stored on 8 RAID 1 volumes. Each volume was constructed from two raw partitions using the Solaris Volume Manager.
- The Temp database for Sybase IQ was configured using 16 raw partitions, one on each of the internal disks. The Temp database was not mirrored.

All the details for configuring the storage used for the tables, logs, temp database, swap space, etc. are provided in appendix B.

5.3 Database partition/replication mapping

The mapping of database partitions/replications must be explicitly described.

Database partitioning/replication was not used.

5.4 RAID Feature

Implementations may use some form of RAID to ensure high availability. If used for data, auxiliary storage (e.g. indexes) or temporary space, the level of RAID must be disclosed for each device.

RAID 1 was used for all base tables and auxiliary data structures. In addition, the Sybase IQ utility db file and tpch log file also resided on a RAID 1 device on each node.

5.5 Modifications to the DBGEN

Any modifications to the DBGEN (see Clause 4.2.1) source code must be disclosed. In the event that a program other than DBGEN was used to populate the database, it must be disclosed in its entirety.

The supplied DBGEN version 2.6.2 was used to generate the database population for this benchmark.

5.6 Database Load Time

The database load time for the test database (see clause 4.3) must be disclosed.

The database load time was =5 Hrs 5 min

5.7 Data Storage Ratio

The data storage ratio must be disclosed. It is computed as the ratio between the total amount of priced disk space, and the chosen test database size as defined in Clause 4.1.3.

The data storage ratio is computed from the following information:

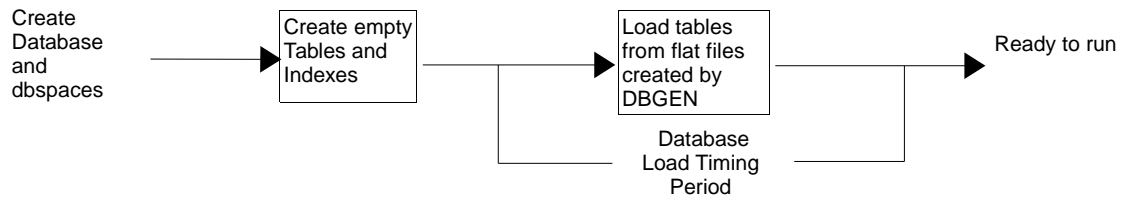
Disk Type	# Of Disks	Space Per Disk*	Sub-Total Disk Space**
internal	16	73 GB	1087.8 GB
external	0	0 GB	0 GB
		Total Space	1087.8 GB
		Data Storage Ratio	3.6

* Disk manufacturer definition of one GB is 10^9 bytes

**In this calculation one GB is defined as 2^{30} bytes

5.8 Database Load Mechanism Details and Illustration

The details of the database load must be described, including a block diagram illustrating the overall process.



The test database was loaded using flat files. All load scripts are included in Appendix B.

5.9 Qualification Database Configuration

Any differences between the configuration of the qualification database and the test database must be disclosed.

The qualification database used identical scripts to create and load the data with adjustments for size differences.

6. Clause 5 Performance Metrics and Execution Rules

6.1 System Activity Between Load and Performance Tests

Any system activity on the SUT that takes place between the conclusion of the load test and the beginning of the performance test must be fully disclosed.

1. Auditor requested queries were run against the database to verify the correctness of the load

All scripts and queries used are included in Appendix F

6.2 Steps in the Power Test

The details of the steps followed to implement the power test (e.g., system boot, database restart, etc.) must be disclosed.

The following steps were used to implement the power test:

1. RF1 Refresh Transaction
2. Stream 00 Execution
3. RF2 Refresh Transaction

6.3 Timing Intervals for Each Query and Refresh Functions

The timing intervals for each query and for both refresh functions must be reported for the power test.

The timing intervals for each query and both update functions are reported on the page titled “Numerical Quantities”, contained in the beginning of this document and replicated in the Executive Summary document.

6.4 Number of Streams for the Throughput Test

The number of execution streams used for the throughput test must be disclosed.

6 streams were used for the throughput test.

6.5 Start and End Date/Times for Each Query Stream

The start time and finish time for each query stream must be reported for the throughput test.

The start times and finish times for each query stream in the throughput test are reported on the page titled “Numerical Quantities”, contained in the beginning of this document and replicated in the Executive Summary document.

6.6 Total Elapsed Time of the Measurement Interval

The total elapsed time of the measurement interval must be reported for the throughput test.

The total elapsed time of the throughput test is reported on the page titled “Numerical Quantities”, contained in the beginning of this document and replicated in the Executive Summary document.

6.7 Refresh Function Start Date/Time and Finish Date/Time

Start and finish time for each refresh function in the refresh stream must be reported for the throughput test.

The start and finish times for each refresh function:

Refresh Function	Start Date	Start Time	End Date	End Time
RF1	16-Jan-08	2:46:24	16-Jan-08	2:56:26
RF2	16-Jan-08	2:56:27	16-Jan-08	3:05:48
RF1	16-Jan-08	3:05:48	16-Jan-08	3:16:08
RF2	16-Jan-08	3:26:10	16-Jan-08	3:34:47
RF1	16-Jan-08	3:34:48	16-Jan-08	3:43:44
RF2	16-Jan-08	3:43:45	16-Jan-08	3:54:19
RF1	16-Jan-08	3:54:19	16-Jan-08	4:02:21
RF2	16-Jan-08	4:07:23	16-Jan-08	4:21:48
RF1	16-Jan-08	4:26:49	16-Jan-08	4:38:29
RF2	16-Jan-08	4:38:29	16-Jan-08	4:48:31
RF1	16-Jan-08	4:48:31	16-Jan-08	4:58:36
RF2	16-Jan-08	5:08:40	16-Jan-08	5:18:38

6.8 Timing Intervals for Each Query and Each Refresh Function for Each Stream

The timing intervals for each query of each stream and each refresh function must be reported for the throughput test.

The timing intervals for each query and each refresh function for the throughput test are reported on the page titled “Numerical Quantities”, contained in the beginning of this document and replicated in the Executive Summary document.

6.9 Performance Metrics

The computed performance metric, related numerical quantities and price performance metric must be reported.

The performance metrics, and the numbers on which they are based, are reported on the page titled “Numerical Quantities”, contained in the beginning of this document and replicated in the Executive Summary document.

6.10 The Performance Metric and Numerical Quantities from Both Runs

The performance metric and numerical quantities from both runs must be disclosed.

Performance results from the first two executions of the TPC-H benchmark indicated the following percent difference for the three metrics:

QppH@ 300GB	Qpth@ 300GB	QphH@ 300GB
8,146.7	8,545.7	8,343.8
8,144.9	8,446.0	8,294.1
-0.02%	-1.18%	-0.60%

6.11 System Activity Between Performance Tests

Any activity on the SUT that takes place between the conclusion of Run1 and the beginning of Run2 must be disclosed.

The database was not restarted after it was loaded or between the two runs.

7. Clause 6 SUT and Driver Implementation

7.1 Driver

A detailed description of how the driver performs its functions must be supplied, including any related source code or scripts. This description should allow an independent reconstruction of the driver.

The entire test is run by executing the `do_test` shellscript.

The text of `do_test` is provided in Appendix E and the texts of all the scripts invoked by `do_test` are provided in Appendix B.

The query streams within the power and throughput tests are generated by a script called `gen_streams_new.ksh` which uses QGEN to generate the query stream files.

The Power Test portion of `do_test` is performed by executing the `update_power.sql` script which runs the refresh functions and the power stream queries.

The Throughput Test portion of `ntest` is performed by executing the 6 query stream scripts, `stream1.sql` – `stream6.sql`, along with simultaneously executing the `update_throughput6.sql` script.

7.2 Implementation-Specific Layer

If an implementation-specific layer is used, then a detailed description of how it performs its functions must be supplied, including any related source code or scripts. This description should allow an independent reconstruction of the implementation-specific layer.

All database configuration was done through scripts disclosed in Appendix B.

The performance tests are performed using `dbisqlc`. `dbisqlc` is a Sybase-provided utility that allows SQL statements to be executed against a Sybase IQ database. The `dbisqlc` utility is invoked from the command-line on the SUT. It reads input from files containing SQL statements and sends results to `stdout`. `dbisqlc` uses information in the `.odbc.ini` file to connect to the database. The performance test scripts utilizing `dbisqlc` can be found in Appendix E.

The ACID tests are performed using `dbtest`. `dbtest` is a Sybase-provided utility, similar to `dbisqlc`, but providing additional scripting capabilities. It is invoked from the command-line on the SUT and uses information in the `.odbc.ini` file to connect to the database. ACID test scripts utilizing `dbtest` can be found in Appendix B.

7.3 Profile-Directed Optimization

If profile-directed optimization as described in Clause 5.2.9 is used, such use must be disclosed.

Profile-directed optimization was not used.

8. Clause 7 Pricing

8.1 Hardware and Software Used

A detailed list of hardware and software used in the priced system must be reported. Each item must have vendor part number, description, and release/revision level, and either general availability status or committed delivery date. If package-pricing is used, contents of the package must be disclosed. Pricing source(s) and effective date(s) of price(s) must also be reported.

Refer to the Executive Summary.

8.2 Total Three Year Price

The total 3-year price of the entire configuration must be reported, including hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.

The total 3-year price of the configuration is \$28132.60. For details of pricing, see the second page of the Executive Summary.

Discounts were taken from actual price quotes, available to any buyer with like conditions, provided by Sun Microsystems Inc. and Sybase Inc. The respective price quotes are included in Appendix G of this document.

8.3 Availability Date

The committed delivery date for general availability of products used in the price calculations must be reported. When the priced system includes products with different availability dates, the reported availability date for the priced system must be the date at which all components are committed to be available.

All hardware and software components used in the measured configuration are available as of May 15, 2008.

9. Auditor's Information and Attestation Letter

The auditor's agency name, address, phone number, and Attestation letter with a brief audit summary report indicating compliance must be included in the full disclosure report. A statement should be included specifying who to contact in order to obtain further information regarding the audit process.

The auditor's attestation letter follows the table of contents.

Appendix A. Solaris 10 and Sybase IQ 12.6 Parameters

This Appendix contains Solaris kernel parameters and environment variables and Sybase IQ system parameters.

Sybase IQ Server Configuration Parameters

utility.cfg

```
-n util_x4240
-c 32m
-gd all
-gl all
-gm 15
-gp 4096
-ti 4400
-tl 300
-iqtc 128
-igmc 128
```

tpch.cfg

```
# note the port number here must matche the one in
# .odbc.ini
```

```
-n tpch_x4240
-x tcpip{port=3002}
-c 12m
-gd all
-gl all
-gm 30
-gp 4096
-ti 4400
-tl 300
-igmc 9000
-iqtc 5200
-igmt 1600
-igqgovern 12
-igqpartition 2
```

Sybase IQ Database Options

(altered from default)

options.sql

```
SET OPTION PUBLIC.Allow_Nulls_By_Default='Off';
SET OPTION PUBLIC.Append_Load='On';
SET OPTION PUBLIC.Flatten_Subqueries = 'On';
SET OPTION PUBLIC.Force_No_Scroll_Cursors='On';
SET OPTION PUBLIC.Load_Memory_Mb=0;
SET OPTION PUBLIC.Max_IQ_Threads_Per_Connection=100;
SET OPTION PUBLIC.Minimize_Storage='On';
SET OPTION PUBLIC.Notify_Modulus=10000000;
SET OPTION PUBLIC.Query_Temp_Space_Limit=0;
SET OPTION PUBLIC.Row_Counts='On';
SET OPTION PUBLIC.Default_Like_Range_Selectivity = 1;
SET OPTION PUBLIC.Query_Plan='off';
SET OPTION PUBLIC.Hash_Thrashing_Percent=100;

SET OPTION
PUBLIC.SignificantDigitsForDoubleEquality=15;

SET OPTION PUBLIC.Default_Having_Selectivity = 1;

SET OPTION PUBLIC.Garray_Fill_Factor_Percent=2;
SET OPTION PUBLIC.Main_Reserved_DBSpace_MB=100;
```

```
SET OPTION PUBLIC.Max_Hash_Rows = 10000000;
SET OPTION PUBLIC.Prefetch_Threads_Percent=15;
```

```
SET OPTION PUBLIC.Sweeper_Threads_Percent=10 ;
```

```
SET OPTION PUBLIC.Sort_Phase1_Helpers=2;
SET OPTION PUBLIC.Wash_Area_Buffers_Percent = '18';
```

Sybase IQ Environment Variables

```
SYBASE="/export/home/sybase"
export SYBASE
SYBASE_OCS="OCS-12_5"
export SYBASE_OCS
ASDIR="${SYBASE}/ASIQ-12_6"
export ASDIR
PATH="${ASDIR}/bin:${SYBASE}/${SYBASE_OCS}/bin:${PATH}:/etc:."
export PATH
LD_LIBRARY_PATH_64="${ASDIR}/lib:${LD_LIBRARY_PATH_64}"
export LD_LIBRARY_PATH_64
LD_LIBRARY_PATH="${ASDIR}/lib:${LD_LIBRARY_PATH}"
export LD_LIBRARY_PATH
```

.odbc.ini

```
[ODBC Data Sources]
tpch=ASIQ Driver
utility_db=ASIQ Driver

[tpch]
Driver=/export/home/sybase/asiq12/lib/dbodbc7_r.so.1
EngineName=tpch_x4240
CommLinks=tcpip{host=10.8.33.153;Port=3002}
DatabaseName=tpch
UserID=DBA
Password=SQL
DBG=yes
LOG=/tmp/tpch_odbc.log

[utility_db]
Driver=/export/home/sybase/asiq12/lib/dbodbc7_r.so.1
EngineName=util_x4240
CommLinks=tcpip{host=10.8.33.153;Port=2638}
DatabaseName=utility_db
UserID=DBA
Password=SQL
DBG=yes
LOG=/tmp/utility_db_odbc.log
```

Solaris Parameters

(altered from default)

/etc/system

```
set tune_t_fsflushr=600
set autoup=36000000
set lotsfree = 4096
set bufhwm = 10000
```

Appendix B. Programs and Scripts

check_query1.bash

```
#!/bin/bash
#
# First remove the rf1.lock so that the Query Stream
will start
#
rm -f /export/home/sybase/run/scripts/rf1.lock
#
# Sleep while the rf2.lock file exists
# when the query stream completes it will remove the
rf2.lock
#
while [ -f /export/home/sybase/run/scripts/rf2.lock ]
do
# Wait for the Query Steam to complete
# check every 10 seconds
# echo "Lock File Exists"
sleep 10
done
# Return Control to the RF stream
```

create_database.sql

```
CREATE DATABASE '/sybase2/tpch.db'
TRANSACTION LOG ON
COLLATION 'ISO_BINENG'
CASE RESPECT
PAGE SIZE 4096
BLANK PADDING ON
JAVA ON
JCONNECT ON
IQ PATH '/sybase2/M01'
IQ PAGE SIZE 524288
TEMPORARY PATH '/sybase2/T01'
```

create_dbspaces.sql

```
set temporary option on_error='continue';
create dbspace main2 as '/sybase2/M02' iq store;
create dbspace main3 as '/sybase2/M03' iq store;
create dbspace main4 as '/sybase2/M04' iq store;

create dbspace iqtemp2 as '/sybase2/T02' iq
temporary store;
create dbspace iqtemp3 as '/sybase2/T03' iq
temporary store;
create dbspace iqtemp4 as '/sybase2/T04' iq
temporary store;
create dbspace iqtemp5 as '/sybase2/T05' iq
temporary store;
create dbspace iqtemp6 as '/sybase2/T06' iq
temporary store;
create dbspace iqtemp7 as '/sybase2/T07' iq
temporary store;
create dbspace iqtemp8 as '/sybase2/T08' iq
temporary store;
```

create_tables_int.sql

```
CREATE TABLE region
(
r_regionkey unsigned int,
r_name char(25),
r_comment varchar(152),
PRIMARY KEY (r_regionkey)
);
```

```
CREATE TABLE nation
(
n_nationkey unsigned int,
n_name char(25),
n_regionkey unsigned int,
n_comment varchar(152),
PRIMARY KEY (n_nationkey)
);
CREATE HG INDEX n_regionkey_hg ON
nation(n_regionkey) ;
```

```
CREATE TABLE supplier
(
s_suppkey unsigned int,
s_name char(25),
s_address varchar(40),
s_nationkey unsigned int,
s_phone char(15),
s_acctbal double precision,
s_comment varchar(101),
PRIMARY KEY (s_suppkey)
);
CREATE HG INDEX s_nationkey_hg ON
supplier(s_nationkey) ;
```

```
CREATE TABLE part
(
p_partkey unsigned int,
p_name varchar(55),
p_mfgr char(25),
p_brand char(10),
p_type varchar(25),
p_size int,
p_container char(10),
p_retailprice double precision,
p_comment varchar(23),
PRIMARY KEY(p_partkey)
);
```

```
CREATE TABLE partsupp
(
ps_partkey unsigned int,
ps_suppkey unsigned int,
ps_availqty integer,
ps_supplycost double precision,
ps_comment varchar(199),
PRIMARY KEY (ps_partkey, ps_suppkey)
);
CREATE HG INDEX ps_partkey_hg ON
partsupp(ps_partkey) ;
CREATE HG INDEX ps_suppkey_hg ON
partsupp(ps_suppkey) ;
```

```
CREATE TABLE customer
(
c_custkey unsigned int,
c_name varchar(25),
c_address varchar(40),
c_nationkey unsigned int,
c_phone char(15),
c_acctbal double precision,
c_mktsegment char(10),
c_comment varchar(117),
PRIMARY KEY(c_custkey)
);
CREATE HG INDEX c_nationkey_hg ON
customer(c_nationkey) ;
```

```
CREATE TABLE orders
(
o_orderkey unsigned int,
o_custkey unsigned int,
o_orderstatus char(1),
```



```

o_totalprice          double precision,
o_orderdate           date,
o_orderpriority       char(15),
o_clerk               char(15),
o_shippriority        int,
o_comment             varchar(79),
PRIMARY KEY (o_orderkey)
);
CREATE HG INDEX o_custkey_hg ON orders(o_custkey) ;
CREATE DATE INDEX o_orderdate_date ON
orders(o_orderdate) ;

CREATE TABLE lineitem
(
  l_orderkey          unsigned int,
  l_partkey           unsigned int,
  l_suppkey           unsigned int,
  l_linenumber        int,
  l_quantity          double precision,
  l_extendedprice     double precision,
  l_discount          double precision,
  l_tax               double precision,
  l_returnflag        char(1),
  l_linestatus        char(1),
  l_shipdate          date,
  l_commitdate        date,
  l_receiptdate       date,
  l_shipinstruct      char(25),
  l_shipmode          char(10),
  l_comment           varchar(44)
);

CREATE HG INDEX l_orderkey_hg ON
lineitem(l_orderkey) ;
CREATE HG INDEX l_partkey_hg ON lineitem(l_partkey) ;
CREATE HG INDEX l_suppkey_hg ON lineitem(l_suppkey) ;
CREATE DATE INDEX l_shipdate_date ON
lineitem(l_shipdate) ;
CREATE DATE INDEX l_receiptdate_date ON
lineitem(l_receiptdate);

```

tpch_rf_int.sql

```

=====
create table refresh_control ( rf1_data_set int not
null, rf2_data_set int not null);
insert into refresh_control values (0,0);
commit;
CREATE PROCEDURE DBA.tpch_rf1 (IN c_directory
varchar(128),
                                IN c_stream varchar(3))
ON EXCEPTION RESUME
BEGIN
  DECLARE delim_ascii integer;
  DECLARE c_data_set varchar(3);
  DECLARE i_data_set integer;
  DECLARE c_cmd long varchar;
  DECLARE outfile_name varchar(128); -- Debug
  DECLARE outfile_name2 varchar(128); -- Debug
  DECLARE c_lf varchar(2);
  DECLARE t_qstart timestamp;
  DECLARE t_qstop timestamp;
  DECLARE n_seconds numeric(16,5);
  DECLARE c_sqlstate CHAR(5);
  SET t_qstart = now(*);
  SET c_lf=char(10);
  SELECT rf1_data_set INTO i_data_set FROM
refresh_control;
  SET c_data_set=CAST(i_data_set+1 AS varchar(3));
  SET c_cmd='load table orders ('+c_lf;
  SET c_cmd=c_cmd+' o_orderkey
'+char(39)+'|'+char(39)+'', '+c_lf;
  SET c_cmd=c_cmd+' o_custkey

```

```

'+char(39)+'|'+char(39)+'', '+c_lf;
  SET c_cmd=c_cmd+' o_orderstatus
'+char(39)+'|'+char(39)+'', '+c_lf;
  SET c_cmd=c_cmd+' o_totalprice
'+char(39)+'|'+char(39)+'', '+c_lf;
  SET c_cmd=c_cmd+' o_orderdate date('+char(39)+'YYYY-
MM-DD'+char(39)+'', filler(1), '+c_lf;
  SET c_cmd=c_cmd+' o_orderpriority
'+char(39)+'|'+char(39)+'', '+c_lf;
  SET c_cmd=c_cmd+' o_clerk '+char(39)+'|'+char(39)+'',
'+c_lf;
  SET c_cmd=c_cmd+' o_shippriority
'+char(39)+'|'+char(39)+'', '+c_lf;
  SET c_cmd=c_cmd+' o_comment
'+char(39)+'|'+char(39)+' )'+c_lf;
  SET c_cmd=c_cmd+'from
'+char(39)+c_directory+'orders.tbl.u'+c_data_set+char(
39)+c_lf;
  SET c_cmd=c_cmd+'row delimited by
'+char(39)+'\\x0a'+char(39)+' quotes off escapes off
preview on;';
EXECUTE IMMEDIATE c_cmd;
SELECT SQLSTATE INTO c_sqlstate;
IF c_sqlstate != '00000' THEN
  ROLLBACK;
  RAISERROR 23002 'RF1 failed at Step 1 with
SQLSTATE: ', c_sqlstate;
  RETURN(1);
END IF;
SET c_cmd='load table lineitem ('+c_lf;
SET c_cmd=c_cmd+' l_orderkey
'+char(39)+'|'+char(39)+'', '+c_lf;
SET c_cmd=c_cmd+' l_partkey
'+char(39)+'|'+char(39)+'', '+c_lf;
SET c_cmd=c_cmd+' l_suppkey
'+char(39)+'|'+char(39)+'', '+c_lf;
SET c_cmd=c_cmd+' l_linenumber
'+char(39)+'|'+char(39)+'', '+c_lf;
SET c_cmd=c_cmd+' l_quantity
'+char(39)+'|'+char(39)+'', '+c_lf;
SET c_cmd=c_cmd+' l_extendedprice
'+char(39)+'|'+char(39)+'', '+c_lf;
SET c_cmd=c_cmd+' l_discount
'+char(39)+'|'+char(39)+'', '+c_lf;
SET c_cmd=c_cmd+' l_tax '+char(39)+'|'+char(39)+'',
'+c_lf;
SET c_cmd=c_cmd+' l_returnflag
'+char(39)+'|'+char(39)+'', '+c_lf;
SET c_cmd=c_cmd+' l_linestatus
'+char(39)+'|'+char(39)+'', '+c_lf;
SET c_cmd=c_cmd+' l_shipdate date('+char(39)+'YYYY-
MM-DD'+char(39)+'', filler(1), '+c_lf;
SET c_cmd=c_cmd+' l_commitdate
date('+char(39)+'YYYY-MM-DD'+char(39)+'', filler(1),
'+c_lf;
SET c_cmd=c_cmd+' l_receiptdate
date('+char(39)+'YYYY-MM-DD'+char(39)+'', filler(1),
'+c_lf;
SET c_cmd=c_cmd+' l_shipinstruct
'+char(39)+'|'+char(39)+'', '+c_lf;
SET c_cmd=c_cmd+' l_shipmode
'+char(39)+'|'+char(39)+'', '+c_lf;
SET c_cmd=c_cmd+' l_comment
'+char(39)+'|'+char(39)+' )'+c_lf;
SET c_cmd=c_cmd+'from
'+char(39)+c_directory+'lineitem.tbl.u'+c_data_set+cha
r(39)+c_lf;
SET c_cmd=c_cmd+'row delimited by
'+char(39)+'\\x0a'+char(39)+c_lf+'quotes off escapes
off preview on;';
EXECUTE IMMEDIATE c_cmd;
SELECT SQLSTATE INTO c_sqlstate;
IF c_sqlstate != '00000' THEN
  rollback;
  RAISERROR 23002 'RF1 failed at Step 2 with
SQLSTATE: ', c_sqlstate;
  RETURN(1);
END IF;

```

```

UPDATE refresh_control SET
rf1_data_set=cast(c_data_set AS integer);
COMMIT;
SET t_qstop = now(*);
SET
n_seconds=cast(datediff(millisecond,t_qstart,t_qstop)
AS numeric(16,5))/1000;
SET c_cmd='Stream updates Update
update_'+c_stream+'_RF1 LENGTH -- '+cast(n_seconds AS
varchar(20))+ ' seconds' ;
SELECT c_cmd;
RETURN(0);
END;
CREATE PROCEDURE DBA.tpch_rf2 (in c_directory
varchar(128),
in c_stream varchar(3))
ON exception resume
BEGIN
DECLARE delim_ascii integer;
DECLARE c_data_set varchar(3);
DECLARE i_data_set integer;
DECLARE c_cmd long varchar;
DECLARE outfile_name varchar(128); -- Debug
DECLARE c_lf varchar(2);
DECLARE t_qstart timestamp;
DECLARE t_qstop timestamp;
DECLARE n_seconds numeric(16,5);
DECLARE c_sqlstate CHAR(5);
SET t_qstart = now(*);
SET c_lf=char(10);
SELECT rf2_data_set INTO i_data_set FROM
refresh_control;
SET c_data_set=CAST(i_data_set+1 AS varchar(3));
CREATE TABLE #delete_table ( d_orderkey UNSIGNED
INT, PRIMARY KEY (d_orderkey) );
SET c_cmd='load table #delete_table (d_orderkey
'+char(39)+'\\x0a'+char(39)+' ' +c_lf;
SET c_cmd=c_cmd+'from
'+char(39)+c_directory+'delete.'+c_data_set+char(39)+c
_lf;
SET c_cmd=c_cmd+'quotes off '+c_lf;
SET c_cmd=c_cmd+'escapes off; '+c_lf;
EXECUTE IMMEDIATE c_cmd;
SELECT SQLSTATE INTO c_sqlstate;
IF c_sqlstate != '00000' THEN
ROLLBACK;
SET c_cmd='RF2 failed at Step 1 with SQLSTATE:
'+c_sqlstate;
RAISERROR 23002 c_cmd;
RETURN(1);
END IF;
DELETE lineitem FROM lineitem
WHERE l_orderkey in (select d_orderkey from
#delete_table);
SELECT SQLSTATE INTO c_sqlstate;
IF c_sqlstate != '00000' THEN
ROLLBACK;
SET c_cmd='RF2 failed at Step 2 with SQLSTATE:
'+c_sqlstate;
RAISERROR 23002 c_cmd;
RETURN(1);
END IF;
DELETE orders FROM orders
WHERE o_orderkey in (select d_orderkey from
#delete_table);
SELECT SQLSTATE INTO c_sqlstate;
IF c_sqlstate != '00000' THEN
ROLLBACK;
SET c_cmd='RF2 failed at Step 3 with SQLSTATE:
'+c_sqlstate;
RAISERROR 23002 c_cmd;
RETURN(1);
END IF;
UPDATE refresh_control SET
rf2_data_set=CAST(c_data_set AS integer);
COMMIT;
DROP TABLE #delete_table;
SET t_qstop = now(*);

```

```

SET
n_seconds=cast(datediff(millisecond,t_qstart,t_qstop)
as numeric(16,5))/1000;
SET c_cmd='Stream updates Update
update_'+c_stream+'_RF2 LENGTH -- '+cast(n_seconds as
varchar(20))+ ' seconds' ;
SELECT c_cmd;
RETURN(0);
END;
=====

```

load_region.sql

```

=====
LOAD TABLE REGION (
R_REGIONKEY
R_NAME
R_COMMENT
)
FROM '/sybase_stage/region.tbl'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
=====

```

load_nation.sql

```

=====
LOAD TABLE NATION (
N_NATIONKEY
N_NAME
N_REGIONKEY
N_COMMENT
)
FROM '/sybase_stage/nation.tbl'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
=====

```

load_customer.sql

```

=====
LOAD TABLE CUSTOMER (
C_CUSTKEY
C_NAME
C_ADDRESS
C_NATIONKEY
C_PHONE
C_ACCTBAL
C_MKTSEGMENT
C_COMMENT
)
FROM '/sybase_stage/customer.tbl'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
=====

```

load_part.sql

```

=====
LOAD TABLE PART (
P_PARTKEY
P_NAME
P_MFGR
P_BRAND
P_TYPE
P_SIZE
P_CONTAINER
)

```

```

P_RETAILPRICE      | | ,
P_COMMENT          | | ,
)
FROM '/sybase_stage/part.tbl'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;

```

load_supplier.sql

```

LOAD TABLE SUPPLIER (
S_SUPPKEY          | | ,
S_NAME            | | ,
S_ADDRESS         | | ,
S_NATIONKEY       | | ,
S_PHONE           | | ,
S_ACCTBAL         | | ,
S_COMMENT         | | ,
)
FROM '/sybase_stage/supplier.tbl'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;

```

load_partsupp.sql

```

LOAD TABLE PARTSUPP (
PS_PARTKEY        | | ,
PS_SUPPKEY        | | ,
PS_AVAILQTY       | | ,
PS_SUPPLYCOST     | | ,
PS_COMMENT        | | ,
)
FROM '/sybase_stage/partsupp.tbl'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;

```

load_orders.sql

```

LOAD TABLE ORDERS (
O_ORDERKEY        | | ,
O_CUSTKEY         | | ,
O_ORDERSTATUS     | | ,
O_TOTALPRICE      | | ,
O_ORDERDATE       | | ,
O_ORDERPRIORITY   | | ,
O_CLERK           | | ,
O_SHIPPRIORITY    | | ,
O_COMMENT         | | ,
)
FROM
'/sybase_stage/orders.tbl'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;

```

load_lineitem.sql

```

=====

```

```

LOAD TABLE LINEITEM (
L_ORDERKEY        | | ,
L_PARTKEY         | | ,
L_SUPPKEY         | | ,
L_LINENUMBER      | | ,
L_QUANTITY        | | ,
L_EXTENDEDPRICE   | | ,
L_DISCOUNT       | | ,
L_TAX             | | ,
L_RETURNFLAG      | | ,
L_LINestatus      | | ,
L_SHIPDATE        | | ,
L_COMMITDATE      | | ,
L_RECEIPTDATE     | | ,
L_SHIPINSTRUCT    | | ,
L_SHIPMODE        | | ,
L_COMMENT         | | ,
)
FROM
'/sybase_stage/lineitem.tbl'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
checkpoint;
commit;

```

update_power.sql

```

=====
create variable qstart timestamp;
create variable qstop timestamp;
create variable c_sqlstate CHAR(5);
create variable c_path varchar(128);
set c_path='/sybase_stage/';
set qstart=now(*);
select 'Stream 0 RF1 START -- ', qstart ;
call tpch_rf1 (c_path,'0');
set qstop=now(*);
select 'Stream 0 Update RF1 LENGTH --
',cast(datediff(millisecond,qstart,qstop) as numeric)/
1000, ' seconds';
select 'Stream 0 RF1 FINISH -- ', qstop ;
-- Sleep Until the query stream completes
set qstart = now(*);
select 'Stream 0 RF WAITING -- ', qstart;
xp_cmdshell('/export/home/sybase/run/scripts/check_que
ry1.bash');
set qstart = now(*);
select 'Stream 0 RF CONTINUING -- ', qstart;
set qstart = now(*);
select 'Stream 0 RF2 START -- ', qstart ;
call tpch_rf2 (c_path,'0');
set qstop=now(*);
select 'Stream 0 Update RF2 LENGTH --
',cast(datediff(millisecond,qstart,qstop) as numeric)/
1000, ' seconds';
select 'Stream 0 RF2 FINISH -- ', qstop ;

```

update_throughput6.sql

```

=====
create variable qstart timestamp;
create variable qstop timestamp;
create variable c_sqlstate CHAR(5);
create variable c_path varchar(128);
set qstart = now(*);
set c_path='/sybase_stage/';
select 'Stream updates START -- ', qstart ;
select @@servername, db_name();
call tpch_rf1 (c_path,'1');
commit;
tpch_wait;

```

```

call tpch_rf2 (c_path,'1');
commit;
tpch_wait;
call tpch_rf1 (c_path,'2');
commit;
tpch_wait;
call tpch_rf2 (c_path,'2');
commit;
tpch_wait;
call tpch_rf1 (c_path,'3');
commit;
tpch_wait;
call tpch_rf2 (c_path,'3');
commit;
tpch_wait;
call tpch_rf1 (c_path,'4');
commit;
tpch_wait;
call tpch_rf2 (c_path,'4');
commit;
tpch_wait;
call tpch_rf1 (c_path,'5');
commit;
tpch_wait;
call tpch_rf2 (c_path,'5');
commit;
-- to have fixed wait time, comment next line and uncomment the line after
tpch_wait;
--xp_cmdshell('sleep 4740'); -- sleep 4680 seconds 78 minutes to better prepare
refresh1 stream1 run2
call tpch_rf1 (c_path,'6');
commit;
tpch_wait;
call tpch_rf2 (c_path,'6');
commit;
set qstop = now(*);
select 'Stream updates STOP -- ', qstop ;

```

gen_streams_new.ksh

```

=====
#!/bin/ksh

if (( $# < 2 ))
then
    echo "usage: $0  seed scale_factor num_streams"
    exit
fi

PATH=/export/home/sybase/ASIQ-12_5/bin:/export/home/sybase/OCS-12_5/bin:/usr/openwin/bin:/bin:/usr/dist/pkgsg/forte_dev/SUNWspro/bin:/usr/ccs/bin:/usr/sr/dt/bin:/usr/dist/pkgsg/devpro,v4.0/5.x-sparc/bin:/usr/dist/local/exe:/usr/dist/exe:/usr/ucb:/usr/sbin:/net/josie/export/home/e18/rgostan/bin:/export/home/sybase/run/scripts:/etc:/export/home/sybase/run/tpch/appendix/dbgen
export PATH
export DSS_PATH=/export/home/sybase/run/scripts;
export DSS_CONFIG=/export/home/sybase/run/tpch/appendix/dbgen;
export DSS_DIST=dists.dss;
export
DSS_QUERY=/export/home/sybase/run/tpch/appendix/templates/queries;
#export
DSS_QUERY=/export/home/sybase/run/tpch/appendix/templates/queries.debug;

seed=$1;
sf=$2;
ns=$3

i=0

```

```

while ((i<=ns))
do
    qgen -c -p $i -l qparm${i}.txt -i $DSS_QUERY/init.sql -t
    $DSS_QUERY/complete.sql -r $seed -s $sf \
        1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 > stream${i}.sql
    ((seed=seed+1))
    ((i=i+1))
done

```

```
((last_seed=seed-1))
```

```
echo $last_seed
```

stream0.sql

```

=====
-- using 115192426 as a seed to the RNG
create variable qstart timestamp;
create variable qstop timestamp;
select 'Stream begin time -- ',now(*) as "time";
-- @(#)14.sql 2.1.8.1
-- TPC-H/TPC-R Promotion Effect Query (Q14)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 0 Query 14 START -- ', qstart ;
select
    100.00 * sum(case
        when p_type like 'PROMO%'
        then l_extendedprice * (1 -
l_discount)
        else 0
    end) / sum(l_extendedprice * (1 - l_discount))
as promo_revenue
from
    lineitem,
    part
where
    l_partkey = p_partkey
    and l_shipdate >= '1995-01-01'
    and l_shipdate <
dateadd(month,1,'1995-01-01');
commit ;
set qstop = now(*);
select 'Stream 0 Query 14 STOP -- ', qstop ;
select 'Stream 0 Query 14 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)2.sql 2.1.8.2
-- TPC-H/TPC-R Minimum Cost Supplier Query (Q2)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 0 Query 2 START -- ', qstart ;
select top 100
    s_acctbal,
    s_name,
    n_name,
    p_partkey,
    p_mfgr,
    s_address,
    s_phone,
    s_comment
from
    part,
    supplier,
    partsupp,
    nation,
    region
where
    p_partkey = ps_partkey
    and s_suppkey = ps_suppkey
    and p_size = 9

```

```

and p_type like '%BRASS'
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = 'ASIA'
and ps_supplycost = (
    select
        min(ps_supplycost)
    from
        partsupp,
        supplier,
        nation,
        region
    where
        p_partkey = ps_partkey
        and s_suppkey = ps_suppkey
        and s_nationkey = n_nationkey
        and n_regionkey = r_regionkey
        and r_name = 'ASIA'
)
order by
    s_acctbal desc,
    n_name,
    s_name,
    p_partkey;
commit ;
set qstop = now(*);
select 'Stream 0 Query 2 STOP -- ', qstop ;
select 'Stream 0 Query 2 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)9.sql 2.1.8.1
-- TPC-H/TPC-R Product Type Profit Measure Query (Q9)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 0 Query 9 START -- ', qstart ;
select
    nation,
    o_year,
    sum(amount) as sum_profit
from
    (
        select
            n_name as nation,
            datepart(year, o_orderdate) as
o_year,
            l_extendedprice * (1 -
l_discount) - ps_supplycost * l_quantity as amount
        from
            part,
            supplier,
            lineitem,
            partsupp,
            orders,
            nation
        where
            s_suppkey = l_suppkey
            and ps_suppkey = l_suppkey
            and ps_partkey = l_partkey
            and p_partkey = l_partkey
            and o_orderkey = l_orderkey
            and s_nationkey = n_nationkey
            and p_name like '%deep%'
    ) as profit
group by
    nation,
    o_year
order by
    nation,
    o_year desc;
commit ;
set qstop = now(*);
select 'Stream 0 Query 9 STOP -- ', qstop ;
select 'Stream 0 Query 9 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;

```

```

-- @(#)20.sql 2.1.8.1
-- TPC-H/TPC-R Potential Part Promotion Query (Q20)
-- Function Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 0 Query 20 START -- ', qstart ;
select
    s_name,
    s_address
from
    supplier,
    nation
where
    s_suppkey in (
        select
            ps_suppkey
        from
            partsupp
        where
            ps_partkey in (
                select
                    p_partkey
                from
                    part
                where
                    p_name like
'cyan%'
            )
        and ps_availqty > (
            select
                0.5 *
sum(l_quantity)
            from
                lineitem
            where
                l_partkey =
ps_partkey
                and l_suppkey =
ps_suppkey
                and l_shipdate
>= '1997-01-01'
                and l_shipdate
< dateadd(year,1,'1997-01-01')
        )
        and s_nationkey = n_nationkey
        and n_name = 'INDIA'
    )
order by
    s_name;
commit ;
set qstop = now(*);
select 'Stream 0 Query 20 STOP -- ', qstop ;
select 'Stream 0 Query 20 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)6.sql 2.1.8.1
-- TPC-H/TPC-R Forecasting Revenue Change Query (Q6)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 0 Query 6 START -- ', qstart ;
select
    sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
    l_shipdate >= '1995-01-01'
    and l_shipdate < dateadd(year,1,'1995-01-01')
    and l_discount between 0.07 - 0.01 and 0.07 +
0.01
    and l_quantity < 25;
commit ;
set qstop = now(*);
select 'Stream 0 Query 6 STOP -- ', qstop ;

```

```

select 'Stream 0 Query 6 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)17.sql 2.1.8.1
-- TPC-H/TPC-R Small-Quantity-Order Revenue Query
(Q17)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 0 Query 17 START -- ', qstart ;
select
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#45'
    and p_container = 'MED JAR'
    and l_quantity < (
        select
            0.2 * avg(l_quantity)
        from
            lineitem
        where
            l_partkey = p_partkey
    ) ;
commit ;
set qstop = now(*) ;
select 'Stream 0 Query 17 STOP -- ', qstop ;
select 'Stream 0 Query 17 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)18.sql 2.1.8.1
-- TPC-H/TPC-R Large Volume Customer Query (Q18)
-- Function Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 0 Query 18 START -- ', qstart ;
select top 100
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice,
    sum(l_quantity)
from
    customer,
    orders,
    lineitem
where
    o_orderkey in (
        select
            l_orderkey
        from
            lineitem
        group by
            l_orderkey having
                sum(l_quantity) > 315
    )
    and c_custkey = o_custkey
    and o_orderkey = l_orderkey
group by
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice
order by
    o_totalprice desc,
    o_orderdate ;
commit ;
set qstop = now(*) ;
select 'Stream 0 Query 18 STOP -- ', qstop ;

```

```

select 'Stream 0 Query 18 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)8.sql 2.1.8.1
-- TPC-H/TPC-R National Market Share Query (Q8)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 0 Query 8 START -- ', qstart ;
select
    o_year,
    sum(case
        when nation = 'UNITED STATES' then
            volume
        else 0
    end) / sum(volume) as mkt_share
from
    (
        select
            datepart(year, o_orderdate) as
o_year,
            l_extendedprice * (1 -
l_discount) as volume,
            n2.n_name as nation
        from
            part,
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2,
            region
        where
            p_partkey = l_partkey
            and s_suppkey = l_suppkey
            and l_orderkey = o_orderkey
            and o_custkey = c_custkey
            and c_nationkey =
n1.n_nationkey
            and n1.n_regionkey =
r_regionkey
            and r_name = 'AMERICA'
            and s_nationkey =
n2.n_nationkey
            and o_orderdate between
'1995-01-01' and '1996-12-31'
            and p_type = 'SMALL PLATED
BRASS'
    ) as all_nations
group by
    o_year
order by
    o_year ;
commit ;
set qstop = now(*) ;
select 'Stream 0 Query 8 STOP -- ', qstop ;
select 'Stream 0 Query 8 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)21.sql 2.1.8.1
-- TPC-H/TPC-R Suppliers Who Kept Orders Waiting Query
(Q21)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 0 Query 21 START -- ', qstart ;
select top 100
    s_name,
    count(*) as numwait
from
    supplier,
    lineitem l1,
    orders,

```

```

nation
where
  s_suppkey = l1.l_suppkey
  and o_orderkey = l1.l_orderkey
  and o_orderstatus = 'F'
  and l1.l_receiptdate > l1.l_commitdate
  and exists (
    select
      *
    from
      lineitem l2
    where
      l2.l_orderkey = l1.l_orderkey
      and l2.l_suppkey <>
l1.l_suppkey
  )
  and not exists (
    select
      *
    from
      lineitem l3
    where
      l3.l_orderkey = l1.l_orderkey
      and l3.l_suppkey <>
l1.l_suppkey
      and l3.l_receiptdate >
l3.l_commitdate
  )
  and s_nationkey = n_nationkey
  and n_name = 'VIETNAM'
group by
  s_name
order by
  numwait desc,
  s_name;
commit ;
set qstop = now(*);
select 'Stream 0 Query 21 STOP -- ', qstop ;
select 'Stream 0 Query 21 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)13.sql 2.1.8.1
-- TPC-H/TPC-R Customer Distribution Query (Q13)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 0 Query 13 START -- ', qstart ;
select
  c_count,
  count(*) as custdist
from
  (
    select
      c_custkey,
      count(o_orderkey)
    from
      customer left outer join orders
on
      c_custkey = o_custkey
      and o_comment not like
'%pending%requests%'
    group by
      c_custkey
  ) as c_orders (c_custkey, c_count)
group by
  c_count
order by
  custdist desc,
  c_count desc;
commit ;
set qstop = now(*);
select 'Stream 0 Query 13 STOP -- ', qstop ;
select 'Stream 0 Query 13 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)3.sql 2.1.8.1

-- TPC-H/TPC-R Shipping Priority Query (Q3)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 0 Query 3 START -- ', qstart ;
select top 10
  l_orderkey,
  sum(l_extendedprice * (1 - l_discount)) as
revenue,
  o_orderdate,
  o_shippriority
from
  customer,
  orders,
  lineitem
where
  c_mktsegment = 'BUILDING'
  and c_custkey = o_custkey
  and l_orderkey = o_orderkey
  and o_orderdate < '1995-03-18'
  and l_shipdate > '1995-03-18'
group by
  l_orderkey,
  o_orderdate,
  o_shippriority
order by
  revenue desc,
  o_orderdate;
commit ;
set qstop = now(*);
select 'Stream 0 Query 3 STOP -- ', qstop ;
select 'Stream 0 Query 3 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)22.sql 2.1.8.1
-- TPC-H/TPC-R Global Sales Opportunity Query (Q22)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 0 Query 22 START -- ', qstart ;
select
  cntrycode,
  count(*) as numcust,
  sum(c_acctbal) as totacctbal
from
  (
    select
      substring(c_phone,1,2) as
cntrycode,
      c_acctbal
    from
      customer
    where
      substring(c_phone,1,2) in
('21', '20', '32', '17',
'25', '10', '14')
      and c_acctbal > (
        select
          avg(c_acctbal)
        from
          customer
        where
          c_acctbal > 0.00
          and
            substring(c_phone,1,2) in
('21',
'20', '32', '17', '25', '10', '14')
        )
      )
    and not exists (
      select
        *
      from
        orders
    where

```

```

c_custkey = o_custkey = *
) as custsale ) from lineitem
group by cntrycode where l_orderkey = o_orderkey
order by cntrycode; l_receiptdate
commit ; )
set qstop = now(*) ; group by
select 'Stream 0 Query 22 STOP -- ', qstop ; o_orderpriority
select 'Stream 0 Query 22 LENGTH -- ', order by
cast(datediff(millisecond,qstart,qstop) as o_orderpriority;
numeric)/1000, ' seconds' ; commit ;
-- @(#)16.sql 2.1.8.1 set qstop = now(*) ;
-- TPC-H/TPC-R Parts/Supplier Relationship Query (Q16) select 'Stream 0 Query 4 STOP -- ', qstop ;
-- Functional Query Definition select 'Stream 0 Query 4 LENGTH -- ',
-- Approved February 1998 cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)11.sql 2.1.8.1
-- TPC-H/TPC-R Important Stock Identification Query
(Q11)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 0 Query 16 START -- ', qstart ;
select
    p_brand,
    p_type,
    p_size,
    count(distinct ps_suppkey) as supplier_cnt
from
    partsupp,
    part
where
    p_partkey = ps_partkey
    and p_brand <> 'Brand#44'
    and p_type not like 'PROMO BRUSHED%'
    and p_size in (25, 16, 14, 5, 18, 46, 23, 44)
    and ps_suppkey not in (
        select
            s_suppkey
        from
            supplier
        where
            s_comment like '%Customer
%Complaints%'
    )
group by
    p_brand,
    p_type,
    p_size
order by
    supplier_cnt desc,
    p_brand,
    p_type,
    p_size;
commit ;
set qstop = now(*) ;
select 'Stream 0 Query 16 STOP -- ', qstop ;
select 'Stream 0 Query 16 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)4.sql 2.1.8.1
-- TPC-H/TPC-R Order Priority Checking Query (Q4)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 0 Query 4 START -- ', qstart ;
select
    o_orderpriority,
    count(*) as order_count
from
    orders
where
    o_orderdate >= '1995-11-01'
    and o_orderdate <
dateadd(month,3,'1995-11-01')
    and exists (
        select
            l_suppkey,
            sum(l_extendedprice * (1 - l_discount))
        from
            lineitem
        where
            l_shipdate >= '1994-01-01'
            and l_shipdate <

```



```

dateadd(month,3,'1994-01-01')
group by
    l_suppkey;

select
    s_suppkey,
    s_name,
    s_address,
    s_phone,
    total_revenue
from
    supplier,
    revenue0
where
    s_suppkey = supplier_no
and total_revenue = (
    select
        max(total_revenue)
    from
        revenue0
    )
order by
    s_suppkey;

drop view revenue0;
commit ;
set qstop = now(*);
select 'Stream 0 Query 15 STOP -- ', qstop ;
select 'Stream 0 Query 15 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)l.sql 2.1.8.1
-- TPC-H/TPC-R Pricing Summary Report Query (Q1)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 0 Query 1 START -- ', qstart ;
select
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as
sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 +
l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineitem
where
    l_shipdate <= dateadd(day,-116,'1998-12-01')
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus;
commit ;
set qstop = now(*);
select 'Stream 0 Query 1 STOP -- ', qstop ;
select 'Stream 0 Query 1 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)10.sql 2.1.8.1
-- TPC-H/TPC-R Returned Item Reporting Query (Q10)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 0 Query 10 START -- ', qstart ;
select top 20

```

```

c_custkey,
c_name,
sum(l_extendedprice * (1 - l_discount)) as
revenue,
c_acctbal,
n_name,
c_address,
c_phone,
c_comment
from
    customer,
    orders,
    lineitem,
    nation
where
    c_custkey = o_custkey
and l_orderkey = o_orderkey
and o_orderdate >= '1994-06-01'
and o_orderdate <
dateadd(month,3,'1994-06-01')
and l_returnflag = 'R'
and c_nationkey = n_nationkey
group by
    c_custkey,
    c_name,
    c_acctbal,
    c_phone,
    n_name,
    c_address,
    c_comment
order by
    revenue desc;
commit ;
set qstop = now(*);
select 'Stream 0 Query 10 STOP -- ', qstop ;
select 'Stream 0 Query 10 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)19.sql 2.1.8.1
-- TPC-H/TPC-R Discounted Revenue Query (Q19)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 0 Query 19 START -- ', qstart ;
select
    sum(l_extendedprice * (1 - l_discount)) as
revenue
from
    lineitem,
    part
where
    (
        p_partkey = l_partkey
and p_brand = 'Brand#32'
and p_container in ('SM CASE', 'SM
BOX', 'SM PACK', 'SM PKG')
and l_quantity >= 3 and l_quantity <= 3
+ 10
and p_size between 1 and 5
and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN
PERSON'
)
or
    (
        p_partkey = l_partkey
and p_brand = 'Brand#52'
and p_container in ('MED BAG', 'MED
BOX', 'MED PKG', 'MED PACK')
and l_quantity >= 19 and l_quantity <=
19 + 10
and p_size between 1 and 10
and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN
PERSON'
)
)

```

```

or
(
    p_partkey = l_partkey
    and p_brand = 'Brand#15'
    and p_container in ('LG CASE', 'LG
BOX', 'LG PACK', 'LG PKG')
    and l_quantity >= 27 and l_quantity <=
27 + 10
    and p_size between 1 and 15
    and l_shipmode in ('AIR', 'AIR REG')
    and l_shipinstruct = 'DELIVER IN
PERSON'
);
commit ;
set qstop = now(*) ;
select 'Stream 0 Query 19 STOP -- ', qstop ;
select 'Stream 0 Query 19 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)5.sql 2.1.8.1
-- TPC-H/TPC-R Local Supplier Volume Query (Q5)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 0 Query 5 START -- ', qstart ;
select
    n_name,
    sum(l_extendedprice * (1 - l_discount)) as
revenue
from
    customer,
    orders,
    lineitem,
    supplier,
    nation,
    region
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and l_suppkey = s_suppkey
    and c_nationkey = s_nationkey
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'MIDDLE EAST'
    and o_orderdate >= '1995-01-01'
    and o_orderdate < dateadd(year,1,'1995-01-01')
group by
    n_name
order by
    revenue desc ;
commit ;
set qstop = now(*) ;
select 'Stream 0 Query 5 STOP -- ', qstop ;
select 'Stream 0 Query 5 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)7.sql 2.1.8.1
-- TPC-H/TPC-R Volume Shipping Query (Q7)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 0 Query 7 START -- ', qstart ;
select
    supp_nation,
    cust_nation,
    l_year,
    sum(volume) as revenue
from
    (
        select
            n1.n_name as supp_nation,
            n2.n_name as cust_nation,
            datepart(year, l_shipdate) as
l_year,
            l_extendedprice * (1 -
l_discount) as volume
        from
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2
        where
            s_suppkey = l_suppkey
            and o_orderkey = l_orderkey
            and c_custkey = o_custkey
            and s_nationkey =
n1.n_nationkey
            and c_nationkey =
n2.n_nationkey
            and (
                (n1.n_name =
'MOZAMBIQUE' and n2.n_name = 'UNITED STATES')
                or (n1.n_name = 'UNITED
STATES' and n2.n_name = 'MOZAMBIQUE')
            )
            and l_shipdate between
'1995-01-01' and '1996-12-31'
        ) as shipping
group by
    supp_nation,
    cust_nation,
    l_year
order by
    supp_nation,
    cust_nation,
    l_year ;
commit ;
set qstop = now(*) ;
select 'Stream 0 Query 7 STOP -- ', qstop ;
select 'Stream 0 Query 7 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)12.sql 2.1.8.1
-- TPC-H/TPC-R Shipping Modes and Order Priority Query
(Q12)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 0 Query 12 START -- ', qstart ;
select
    l_shipmode,
    sum(case
        when o_orderpriority = '1-URGENT'
        or o_orderpriority = '2-HIGH'
        then 1
        else 0
    end) as high_line_count,
    sum(case
        when o_orderpriority <> '1-URGENT'
        and o_orderpriority <> '2-HIGH'
        then 1
        else 0
    end) as low_line_count
from
    orders,
    lineitem
where
    o_orderkey = l_orderkey
    and l_shipmode in ('REG AIR', 'AIR')
    and l_commitdate < l_receiptdate
    and l_shipdate < l_commitdate
    and l_receiptdate >= '1995-01-01'
    and l_receiptdate <
dateadd(year,1,'1995-01-01')
group by
    l_shipmode
order by
    l_shipmode ;

```

```

commit ;
set qstop = now(*) ;
select 'Stream 0 Query 12 STOP -- ', qstop ;
select 'Stream 0 Query 12 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;

select 'Stream completion time -- ',now(*) as "time";

disconnect;

exit;

```

stream1.sql

```

-- using 115192427 as a seed to the RNG
create variable qstart timestamp;
create variable qstop timestamp;
select 'Stream begin time -- ',now(*) as "time";
-- @(#)21.sql      2.1.8.1
-- TPC-H/TPC-R Suppliers Who Kept Orders Waiting Query (Q21)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 1 Query 21 START -- ', qstart ;
select top 100
    s_name,
    count(*) as numwait
from
    supplier,
    lineitem l1,
    orders,
    nation
where
    s_suppkey = l1.l_suppkey
    and o_orderkey = l1.l_orderkey
    and o_orderstatus = 'F'
    and l1.l_receiptdate > l1.l_commitdate
    and exists (
        select
            *
        from
            lineitem l2
        where
            l2.l_orderkey = l1.l_orderkey
            and l2.l_suppkey <> l1.l_suppkey
    )
    and not exists (
        select
            *
        from
            lineitem l3
        where
            l3.l_orderkey = l1.l_orderkey
            and l3.l_suppkey <> l1.l_suppkey
            and l3.l_receiptdate > l3.l_commitdate
    )
    and s_nationkey = n_nationkey
    and n_name = 'JORDAN'
group by
    s_name
order by
    numwait desc,
    s_name;

commit ;
set qstop = now(*) ;
select 'Stream 1 Query 21 STOP -- ', qstop ;
select 'Stream 1 Query 21 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)3.sql      2.1.8.1

```

```

-- TPC-H/TPC-R Shipping Priority Query (Q3)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 1 Query 3 START -- ', qstart ;
select top 10
    l_orderkey,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    o_orderdate,
    o_shippriority
from
    customer,
    orders,
    lineitem
where
    c_mktsegment = 'MACHINERY'
    and c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate < '1995-03-04'
    and l_shipdate > '1995-03-04'
group by
    l_orderkey,
    o_orderdate,
    o_shippriority
order by
    revenue desc,
    o_orderdate;

commit ;
set qstop = now(*) ;
select 'Stream 1 Query 3 STOP -- ', qstop ;
select 'Stream 1 Query 3 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)18.sql      2.1.8.1
-- TPC-H/TPC-R Large Volume Customer Query (Q18)
-- Function Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 1 Query 18 START -- ', qstart ;
select top 100
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice,
    sum(l_quantity)
from
    customer,
    orders,
    lineitem
where
    o_orderkey in (
        select
            l_orderkey
        from
            lineitem
        group by
            l_orderkey having
                sum(l_quantity) > 313
    )
    and c_custkey = o_custkey
    and o_orderkey = l_orderkey
group by
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice
order by
    o_totalprice desc,

```

```

o_orderdate;
commit ;
set qstop = now(*);
select 'Stream 1 Query 18 STOP -- ', qstop ;
select 'Stream 1 Query 18 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)5.sql      2.1.8.1
-- TPC-H/TPC-R Local Supplier Volume Query (Q5)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 1 Query 5 START -- ', qstart ;
select

```

```

    n_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue
from

```

```

    customer,
    orders,
    lineitem,
    supplier,
    nation,
    region
where

```

```

    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and l_suppkey = s_suppkey
    and c_nationkey = s_nationkey
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'AFRICA'
    and o_orderdate >= '1995-01-01'
    and o_orderdate < dateadd(year,1,'1995-01-01')
group by

```

```

    n_name
order by
    revenue desc;
commit ;

```

```

set qstop = now(*);
select 'Stream 1 Query 5 STOP -- ', qstop ;
select 'Stream 1 Query 5 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)11.sql     2.1.8.1
-- TPC-H/TPC-R Important Stock Identification Query (Q11)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 1 Query 11 START -- ', qstart ;
select

```

```

    ps_partkey,
    sum(ps_supplycost * ps_availqty) as value
from

```

```

    partsupp,
    supplier,
    nation
where

```

```

    ps_suppkey = s_suppkey
    and s_nationkey = n_nationkey
    and n_name = 'JORDAN'
group by

```

```

    ps_partkey having
        sum(ps_supplycost * ps_availqty) > (
            select
                sum(ps_supplycost * ps_availqty) *
0.0000003333

```

```

        from
            partsupp,
            supplier,
            nation

```

```

        where

```

```

ps_suppkey = s_suppkey
and s_nationkey = n_nationkey
and n_name = 'JORDAN'

```

```

        )
order by
    value desc;
commit ;

```

```

set qstop = now(*);
select 'Stream 1 Query 11 STOP -- ', qstop ;
select 'Stream 1 Query 11 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)7.sql      2.1.8.1
-- TPC-H/TPC-R Volume Shipping Query (Q7)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 1 Query 7 START -- ', qstart ;
select

```

```

    supp_nation,
    cust_nation,
    l_year,
    sum(volume) as revenue
from

```

```

    (
        select
            n1.n_name as supp_nation,
            n2.n_name as cust_nation,
            datepart(year, l_shipdate) as l_year,
            l_extendedprice * (1 - l_discount) as volume
        from

```

```

            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2
        where

```

```

            s_suppkey = l_suppkey
            and o_orderkey = l_orderkey
            and c_custkey = o_custkey
            and s_nationkey = n1.n_nationkey
            and c_nationkey = n2.n_nationkey
            and (
                (n1.n_name = 'INDIA' and
                    n2.n_name = 'MOZAMBIQUE')
                or (n1.n_name = 'MOZAMBIQUE'
                    and n2.n_name = 'INDIA')
            )
        and l_shipdate between '1995-01-01' and

```

```

'1996-12-31'
    ) as shipping
group by

```

```

    supp_nation,
    cust_nation,
    l_year
order by

```

```

    supp_nation,
    cust_nation,
    l_year;
commit ;

```

```

set qstop = now(*);
select 'Stream 1 Query 7 STOP -- ', qstop ;
select 'Stream 1 Query 7 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)6.sql      2.1.8.1
-- TPC-H/TPC-R Forecasting Revenue Change Query (Q6)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);

```

```

select 'Stream 1 Query 6 START --', qstart ;
select
    sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
    l_shipdate >= '1995-01-01'
    and l_shipdate < dateadd(year,1,'1995-01-01')
    and l_discount between 0.04 - 0.01 and 0.04 + 0.01
    and l_quantity < 25;
commit ;
set qstop = now(*);
select 'Stream 1 Query 6 STOP --', qstop ;
select 'Stream 1 Query 6 LENGTH --', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, 'seconds' ;
-- @(#)20.sql      2.1.8.1
-- TPC-H/TPC-R Potential Part Promotion Query (Q20)
-- Function Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 1 Query 20 START --', qstart ;
select
    s_name,
    s_address
from
    supplier,
    nation
where
    s_suppkey in (
        select
            ps_suppkey
        from
            partsupp
        where
            ps_partkey in (
                select
                    p_partkey
                from
                    part
                where
                    p_name like 'orange%'
            )
        and ps_availqty > (
            select
                0.5 * sum(l_quantity)
            from
                lineitem
            where
                l_partkey = ps_partkey
                and l_suppkey =
                    ps_suppkey
                and l_shipdate >=
                    '1995-01-01'
                and l_shipdate < dateadd(year,1,'1995-01-01')
            )
        and s_nationkey = n_nationkey
        and n_name = 'UNITED KINGDOM'
    )
order by
    s_name;
commit ;
set qstop = now(*);
select 'Stream 1 Query 20 STOP --', qstop ;
select 'Stream 1 Query 20 LENGTH --', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, 'seconds' ;
-- @(#)17.sql      2.1.8.1
-- TPC-H/TPC-R Small-Quantity-Order Revenue Query (Q17)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 1 Query 17 START --', qstart ;
select
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#42'
    and p_container = 'MED CAN'
    and l_quantity < (
        select
            0.2 * avg(l_quantity)
        from
            lineitem
        where
            l_partkey = p_partkey
    );
commit ;
set qstop = now(*);
select 'Stream 1 Query 17 STOP --', qstop ;
select 'Stream 1 Query 17 LENGTH --', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, 'seconds' ;
-- @(#)12.sql      2.1.8.1
-- TPC-H/TPC-R Shipping Modes and Order Priority Query (Q12)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 1 Query 12 START --', qstart ;
select
    l_shipmode,
    sum(case
        when o_orderpriority = '1-URGENT'
            or o_orderpriority = '2-HIGH'
        then 1
        else 0
    end) as high_line_count,
    sum(case
        when o_orderpriority <> '1-URGENT'
            and o_orderpriority <> '2-HIGH'
        then 1
        else 0
    end) as low_line_count
from
    orders,
    lineitem
where
    o_orderkey = l_orderkey
    and l_shipmode in ('SHIP', 'RAIL')
    and l_commitdate < l_receiptdate
    and l_shipdate < l_commitdate
    and l_receiptdate >= '1995-01-01'
    and l_receiptdate < dateadd(year,1,'1995-01-01')
group by
    l_shipmode
order by
    l_shipmode;
commit ;
set qstop = now(*);
select 'Stream 1 Query 12 STOP --', qstop ;
select 'Stream 1 Query 12 LENGTH --', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, 'seconds' ;
-- @(#)16.sql      2.1.8.1
-- TPC-H/TPC-R Parts/Supplier Relationship Query (Q16)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 1 Query 16 START --', qstart ;

```

```

select
    p_brand,
    p_type,
    p_size,
    count(distinct ps_suppkey) as supplier_cnt
from
    partsupp,
    part
where
    p_partkey = ps_partkey
    and p_brand <> 'Brand#34'
    and p_type not like 'MEDIUM ANODIZED%'
    and p_size in (29, 25, 18, 47, 34, 23, 2, 10)
    and ps_suppkey not in (
        select
            s_suppkey
        from
            supplier
        where
            s_comment like '%Customer%Complaints%'
    )
group by
    p_brand,
    p_type,
    p_size
order by
    supplier_cnt desc,
    p_brand,
    p_type,
    p_size;
commit ;
set qstop = now(*);
select 'Stream 1 Query 16 STOP -- ', qstop ;
select 'Stream 1 Query 16 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)15.sql      2.1.8.1
-- TPC-H/TPC-R Top Supplier Query (Q15)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 1 Query 15 START -- ', qstart ;
create view revenue1 (supplier_no, total_revenue) as
select
    l_suppkey,
    sum(l_extendedprice * (1 - l_discount))
from
    lineitem
where
    l_shipdate >= '1997-01-01'
and l_shipdate < dateadd(month,3,'1997-01-01')
group by
    l_suppkey;

select
    s_suppkey,
    s_name,
    s_address,
    s_phone,
    total_revenue
from
    supplier,
    revenue1
where
    s_suppkey = supplier_no
and total_revenue = (
    select
        max(total_revenue)
    from
        revenue1
    )
order by

```

```

s_suppkey;

drop view revenue1;
commit ;
set qstop = now(*);
select 'Stream 1 Query 15 STOP -- ', qstop ;
select 'Stream 1 Query 15 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)13.sql      2.1.8.1
-- TPC-H/TPC-R Customer Distribution Query (Q13)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 1 Query 13 START -- ', qstart ;
select
    c_count,
    count(*) as custdist
from
    (
        select
            c_custkey,
            count(o_orderkey)
        from
            customer left outer join orders on
                c_custkey = o_custkey
                and o_comment not like '%pending
                %requests%'
        group by
            c_custkey
    ) as c_orders (c_custkey, c_count)
group by
    c_count
order by
    custdist desc,
    c_count desc;
commit ;
set qstop = now(*);
select 'Stream 1 Query 13 STOP -- ', qstop ;
select 'Stream 1 Query 13 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)10.sql      2.1.8.1
-- TPC-H/TPC-R Returned Item Reporting Query (Q10)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 1 Query 10 START -- ', qstart ;
select top 20
    c_custkey,
    c_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    c_acctbal,
    n_name,
    c_address,
    c_phone,
    c_comment
from
    customer,
    orders,
    lineitem,
    nation
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate >= '1993-03-01'
    and o_orderdate < dateadd(month,3,'1993-03-01')
    and l_returnflag = 'R'
    and c_nationkey = n_nationkey
group by
    c_custkey,

```

```

c_name,
c_acctbal,
c_phone,
n_name,
c_address,
c_comment
order by
    revenue desc;
commit ;
set qstop = now(*);
select 'Stream 1 Query 10 STOP -- ', qstop ;
select 'Stream 1 Query 10 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, 'seconds' ;
-- @(#)2.sql      2.1.8.2
-- TPC-H/TPC-R Minimum Cost Supplier Query (Q2)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 1 Query 2 START -- ', qstart ;
select top 100
    s_acctbal,
    s_name,
    n_name,
    p_partkey,
    p_mfgr,
    s_address,
    s_phone,
    s_comment
from
    part,
    supplier,
    partsupp,
    nation,
    region
where
    p_partkey = ps_partkey
and s_suppkey = ps_suppkey
and p_size = 46
and p_type like '%NICKEL'
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = 'AFRICA'
and ps_supplycost = (
    select
        min(ps_supplycost)
    from
        partsupp,
        supplier,
        nation,
        region
    where
        p_partkey = ps_partkey
and s_suppkey = ps_suppkey
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = 'AFRICA'
)
order by
    s_acctbal desc,
    n_name,
    s_name,
    p_partkey;
commit ;
set qstop = now(*);
select 'Stream 1 Query 2 STOP -- ', qstop ;
select 'Stream 1 Query 2 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, 'seconds' ;
-- @(#)8.sql      2.1.8.1
-- TPC-H/TPC-R National Market Share Query (Q8)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 1 Query 8 START -- ', qstart ;
select
    o_year,
    sum(case
        when nation = 'MOZAMBIQUE' then volume
        else 0
    end) / sum(volume) as mkt_share
from
    (
        select
            datepart(year, o_orderdate) as o_year,
            l_extendedprice * (1 - l_discount) as volume,
            n2.n_name as nation
        from
            part,
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2,
            region
        where
            p_partkey = l_partkey
and s_suppkey = l_suppkey
and l_orderkey = o_orderkey
and o_custkey = c_custkey
and c_nationkey = n1.n_nationkey
and n1.n_regionkey = r_regionkey
and r_name = 'AFRICA'
and s_nationkey = n2.n_nationkey
and o_orderdate between '1995-01-01' and
'1996-12-31'
    ) as all_nations
group by
    o_year
order by
    o_year;
commit ;
set qstop = now(*);
select 'Stream 1 Query 8 STOP -- ', qstop ;
select 'Stream 1 Query 8 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, 'seconds' ;
-- @(#)14.sql     2.1.8.1
-- TPC-H/TPC-R Promotion Effect Query (Q14)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 1 Query 14 START -- ', qstart ;
select
    100.00 * sum(case
        when p_type like 'PROMO%'
        then l_extendedprice * (1 - l_discount)
        else 0
    end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
    lineitem,
    part
where
    l_partkey = p_partkey
and l_shipdate >= '1995-01-01'
and l_shipdate < dateadd(month,1,'1995-01-01');
commit ;
set qstop = now(*);
select 'Stream 1 Query 14 STOP -- ', qstop ;
select 'Stream 1 Query 14 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, 'seconds' ;

```

```

-- @(#)19.sql      2.1.8.1
-- TPC-H/TPC-R Discounted Revenue Query (Q19)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 1 Query 19 START -- ', qstart ;
select
    sum(l_extendedprice* (1 - l_discount)) as revenue
from
    lineitem,
    part
where
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#34'
        and p_container in ('SM CASE', 'SM BOX', 'SM PACK',
'SM PKG')
        and l_quantity >= 8 and l_quantity <= 8 + 10
        and p_size between 1 and 5
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#35'
        and p_container in ('MED BAG', 'MED BOX', 'MED
PKG', 'MED PACK')
        and l_quantity >= 20 and l_quantity <= 20 + 10
        and p_size between 1 and 10
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#54'
        and p_container in ('LG CASE', 'LG BOX', 'LG PACK',
'LG PKG')
        and l_quantity >= 23 and l_quantity <= 23 + 10
        and p_size between 1 and 15
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    );
commit ;
set qstop = now(*);
select 'Stream 1 Query 19 STOP -- ', qstop ;
select 'Stream 1 Query 19 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)9.sql      2.1.8.1
-- TPC-H/TPC-R Product Type Profit Measure Query (Q9)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 1 Query 9 START -- ', qstart ;
select
    nation,
    o_year,
    sum(amount) as sum_profit
from
    (
        select
            n_name as nation,
            datepart(year, o_orderdate) as o_year,
            l_extendedprice * (1 - l_discount) -
ps_supplycost * l_quantity as amount
            from
                part,
                supplier,

```

```

lineitem,
partsupp,
orders,
nation
where
    s_suppkey = l_suppkey
    and ps_suppkey = l_suppkey
    and ps_partkey = l_partkey
    and p_partkey = l_partkey
    and o_orderkey = l_orderkey
    and s_nationkey = n_nationkey
    and p_name like '%coral%'
) as profit
group by
    nation,
    o_year
order by
    nation,
    o_year desc;
commit ;
set qstop = now(*);
select 'Stream 1 Query 9 STOP -- ', qstop ;
select 'Stream 1 Query 9 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)22.sql      2.1.8.1
-- TPC-H/TPC-R Global Sales Opportunity Query (Q22)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 1 Query 22 START -- ', qstart ;
select
    entrycode,
    count(*) as numcust,
    sum(c_acctbal) as totacctbal
from
    (
        select
            substring(c_phone,1,2) as entrycode,
            c_acctbal
        from
            customer
        where
            substring(c_phone,1,2) in
                ('31', '24', '22', '14', '11', '18', '10')
            and c_acctbal > (
                select
                    avg(c_acctbal)
                from
                    customer
                where
                    c_acctbal > 0.00
            and substring(c_phone,1,2) in
                ('31', '24',
'22', '14', '11', '18', '10')
            )
        )
    and not exists (
        select
            *
        from
            orders
        where
            o_custkey = c_custkey
    )
) as custsale
group by
    entrycode
order by
    entrycode;
commit ;
set qstop = now(*);
select 'Stream 1 Query 22 STOP -- ', qstop ;

```



```

select 'Stream 1 Query 22 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)1.sql      2.1.8.1
-- TPC-H/TPC-R Pricing Summary Report Query (Q1)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 1 Query 1 START -- ', qstart ;
select
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineitem
where
    l_shipdate <= dateadd(day,-63,'1998-12-01')
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus;
commit ;
set qstop = now(*) ;
select 'Stream 1 Query 1 STOP -- ', qstop ;
select 'Stream 1 Query 1 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)4.sql      2.1.8.1
-- TPC-H/TPC-R Order Priority Checking Query (Q4)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 1 Query 4 START -- ', qstart ;
select
    o_orderpriority,
    count(*) as order_count
from
    orders
where
    o_orderdate >= '1993-08-01'
    and o_orderdate < dateadd(month,3,'1993-08-01')
    and exists (
        select
            *
        from
            lineitem
        where
            l_orderkey = o_orderkey
            and l_commitdate < l_receiptdate
    )
group by
    o_orderpriority
order by
    o_orderpriority;
commit ;
set qstop = now(*) ;
select 'Stream 1 Query 4 STOP -- ', qstop ;
select 'Stream 1 Query 4 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;

select 'Stream completion time -- ',now(*) as "time";

```

disconnect;

exit;

=====

stream2.sql

```

-- using 115192428 as a seed to the RNG
create variable qstart timestamp;
create variable qstop timestamp;
select 'Stream begin time -- ',now(*) as "time";
-- @(#)6.sql      2.1.8.1
-- TPC-H/TPC-R Forecasting Revenue Change Query (Q6)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 2 Query 6 START -- ', qstart ;
select
    sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
    l_shipdate >= '1995-01-01'
    and l_shipdate < dateadd(year,1,'1995-01-01')
    and l_discount between 0.02 - 0.01 and 0.02 + 0.01
    and l_quantity < 24;
commit ;
set qstop = now(*) ;
select 'Stream 2 Query 6 STOP -- ', qstop ;
select 'Stream 2 Query 6 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)17.sql     2.1.8.1
-- TPC-H/TPC-R Small-Quantity-Order Revenue Query (Q17)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 2 Query 17 START -- ', qstart ;
select
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#44'
    and p_container = 'JUMBO BOX'
    and l_quantity < (
        select
            0.2 * avg(l_quantity)
        from
            lineitem
        where
            l_partkey = p_partkey
    );
commit ;
set qstop = now(*) ;
select 'Stream 2 Query 17 STOP -- ', qstop ;
select 'Stream 2 Query 17 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)14.sql     2.1.8.1
-- TPC-H/TPC-R Promotion Effect Query (Q14)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 2 Query 14 START -- ', qstart ;
select

```

```

100.00 * sum(case
    when p_type like 'PROMO%'
        then l_extendedprice * (1 - l_discount)
    else 0
end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
    lineitem,
    part
where
    l_partkey = p_partkey
    and l_shipdate >= '1995-01-01'
    and l_shipdate < dateadd(month,1,'1995-01-01');
commit ;
set qstop = now(*);
select 'Stream 2 Query 14 STOP -- ', qstop ;
select 'Stream 2 Query 14 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)16.sql      2.1.8.1
-- TPC-H/TPC-R Parts/Supplier Relationship Query (Q16)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 2 Query 16 START -- ', qstart ;
select
    p_brand,
    p_type,
    p_size,
    count(distinct ps_suppkey) as supplier_cnt
from
    partsupp,
    part
where
    p_partkey = ps_partkey
    and p_brand <> 'Brand#14'
    and p_type not like 'ECONOMY PLATED%'
    and p_size in (32, 3, 22, 11, 47, 45, 13, 16)
    and ps_suppkey not in (
        select
            s_suppkey
        from
            supplier
        where
            s_comment like '%Customer%Complaints%'
    )
group by
    p_brand,
    p_type,
    p_size
order by
    supplier_cnt desc,
    p_brand,
    p_type,
    p_size;
commit ;
set qstop = now(*);
select 'Stream 2 Query 16 STOP -- ', qstop ;
select 'Stream 2 Query 16 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)19.sql      2.1.8.1
-- TPC-H/TPC-R Discounted Revenue Query (Q19)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 2 Query 19 START -- ', qstart ;
select
    sum(l_extendedprice*(1 - l_discount)) as revenue
from
    lineitem,
    part
where
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#41'
        and p_container in ('SM CASE', 'SM BOX', 'SM PACK',
'SM PKG')
        and l_quantity >= 3 and l_quantity <= 3 + 10
        and p_size between 1 and 5
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#23'
        and p_container in ('MED BAG', 'MED BOX', 'MED
PKG', 'MED PACK')
        and l_quantity >= 10 and l_quantity <= 10 + 10
        and p_size between 1 and 10
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#54'
        and p_container in ('LG CASE', 'LG BOX', 'LG PACK',
'LG PKG')
        and l_quantity >= 30 and l_quantity <= 30 + 10
        and p_size between 1 and 15
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    );
commit ;
set qstop = now(*);
select 'Stream 2 Query 19 STOP -- ', qstop ;
select 'Stream 2 Query 19 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)10.sql      2.1.8.1
-- TPC-H/TPC-R Returned Item Reporting Query (Q10)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 2 Query 10 START -- ', qstart ;
select top 20
    c_custkey,
    c_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    c_acctbal,
    n_name,
    c_address,
    c_phone,
    c_comment
from
    customer,
    orders,
    lineitem,
    nation
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate >= '1993-12-01'
    and o_orderdate < dateadd(month,3,'1993-12-01')
    and l_returnflag = 'R'
    and c_nationkey = n_nationkey
group by
    c_custkey,
    c_name,
    c_acctbal,
    c_phone,
    n_name,

```

```

        c_address,
        c_comment
order by
    revenue desc;
commit ;
set qstop = now(*);
select 'Stream 2 Query 10 STOP -- ', qstop ;
select 'Stream 2 Query 10 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)9.sql      2.1.8.1
-- TPC-H/TPC-R Product Type Profit Measure Query (Q9)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 2 Query 9 START -- ', qstart ;
select
    nation,
    o_year,
    sum(amount) as sum_profit
from
    (
        select
            n_name as nation,
            datepart(year, o_orderdate) as o_year,
            l_extendedprice * (1 - l_discount) -
ps_supplycost * l_quantity as amount
        from
            part,
            supplier,
            lineitem,
            partsupp,
            orders,
            nation
        where
            s_suppkey = l_suppkey
            and ps_suppkey = l_suppkey
            and ps_partkey = l_partkey
            and p_partkey = l_partkey
            and o_orderkey = l_orderkey
            and s_nationkey = n_nationkey
            and p_name like '%brown%'
        ) as profit
group by
    nation,
    o_year
order by
    nation,
    o_year desc;
commit ;
set qstop = now(*);
select 'Stream 2 Query 9 STOP -- ', qstop ;
select 'Stream 2 Query 9 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)2.sql      2.1.8.2
-- TPC-H/TPC-R Minimum Cost Supplier Query (Q2)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 2 Query 2 START -- ', qstart ;
select top 100
    s_acctbal,
    s_name,
    n_name,
    p_partkey,
    p_mfgr,
    s_address,
    s_phone,
    s_comment
from

```

```

part,
supplier,
partsupp,
nation,
region
where
    p_partkey = ps_partkey
    and s_suppkey = ps_suppkey
    and p_size = 34
    and p_type like '%COPPER'
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'EUROPE'
    and ps_supplycost = (
        select
            min(ps_supplycost)
        from
            partsupp,
            supplier,
            nation,
            region
        where
            p_partkey = ps_partkey
            and s_suppkey = ps_suppkey
            and s_nationkey = n_nationkey
            and n_regionkey = r_regionkey
            and r_name = 'EUROPE'
    )
order by
    s_acctbal desc,
    n_name,
    s_name,
    p_partkey;
commit ;
set qstop = now(*);
select 'Stream 2 Query 2 STOP -- ', qstop ;
select 'Stream 2 Query 2 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)15.sql     2.1.8.1
-- TPC-H/TPC-R Top Supplier Query (Q15)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 2 Query 15 START -- ', qstart ;
create view revenue2 (supplier_no, total_revenue) as
    select
        l_suppkey,
        sum(l_extendedprice * (1 - l_discount))
    from
        lineitem
    where
        l_shipdate >= '1994-01-01'
        and l_shipdate < dateadd(month,3,'1994-01-01')
    group by
        l_suppkey;

select
    s_suppkey,
    s_name,
    s_address,
    s_phone,
    total_revenue
from
    supplier,
    revenue2
where
    s_suppkey = supplier_no
    and total_revenue = (
        select
            max(total_revenue)
        from

```

```

revenue2
)
order by
  s_suppkey;

drop view revenue2;
commit ;
set qstop = now(*);
select 'Stream 2 Query 15 STOP -- ', qstop ;
select 'Stream 2 Query 15 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, 'seconds' ;
-- @(#)8.sql      2.1.8.1
-- TPC-H/TPC-R National Market Share Query (Q8)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 2 Query 8 START -- ', qstart ;
select
  o_year,
  sum(case
    when nation = 'GERMANY' then volume
    else 0
  end) / sum(volume) as mkt_share
from
  (
    select
      datepart(year, o_orderdate) as o_year,
      l_extendedprice * (1 - l_discount) as volume,
      n2.n_name as nation
    from
      part,
      supplier,
      lineitem,
      orders,
      customer,
      nation n1,
      nation n2,
      region
    where
      p_partkey = l_partkey
      and s_suppkey = l_suppkey
      and l_orderkey = o_orderkey
      and o_custkey = c_custkey
      and c_nationkey = n1.n_nationkey
      and n1.n_regionkey = r_regionkey
      and r_name = 'EUROPE'
      and s_nationkey = n2.n_nationkey
      and o_orderdate between '1995-01-01' and
'1996-12-31'
      and p_type = 'STANDARD POLISHED'
      and n2.n_name = 'GERMANY'
  ) as all_nations
group by
  o_year
order by
  o_year;
commit ;
set qstop = now(*);
select 'Stream 2 Query 8 STOP -- ', qstop ;
select 'Stream 2 Query 8 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, 'seconds' ;
-- @(#)5.sql      2.1.8.1
-- TPC-H/TPC-R Local Supplier Volume Query (Q5)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 2 Query 5 START -- ', qstart ;
select
  n_name,

```

```

sum(l_extendedprice * (1 - l_discount)) as revenue
from
  customer,
  orders,
  lineitem,
  supplier,
  nation,
  region
where
  c_custkey = o_custkey
  and l_orderkey = o_orderkey
  and l_suppkey = s_suppkey
  and c_nationkey = s_nationkey
  and s_nationkey = n_nationkey
  and n_regionkey = r_regionkey
  and r_name = 'AMERICA'
  and o_orderdate >= '1995-01-01'
  and o_orderdate < dateadd(year,1,'1995-01-01')
group by
  n_name
order by
  revenue desc;
commit ;
set qstop = now(*);
select 'Stream 2 Query 5 STOP -- ', qstop ;
select 'Stream 2 Query 5 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, 'seconds' ;
-- @(#)22.sql     2.1.8.1
-- TPC-H/TPC-R Global Sales Opportunity Query (Q22)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 2 Query 22 START -- ', qstart ;
select
  cntrycode,
  count(*) as numcust,
  sum(c_acctbal) as totacctbal
from
  (
    select
      substring(c_phone,1,2) as cntrycode,
      c_acctbal
    from
      customer
    where
      substring(c_phone,1,2) in
('23', '18', '29', '16', '19', '34', '15')
      and c_acctbal > (
select
  avg(c_acctbal)
from
  customer
where
  c_acctbal > 0.00
and substring(c_phone,1,2) in
('23', '18',
'29', '16', '19', '34', '15')
)
    and not exists (
select
  *
from
  orders
where
  o_custkey = c_custkey
)
  ) as custsale
group by
  cntrycode
order by
  cntrycode;

```

```

commit ;
set qstop = now(*);
select 'Stream 2 Query 22 STOP -- ', qstop ;
select 'Stream 2 Query 22 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)12.sql      2.1.8.1
-- TPC-H/TPC-R Shipping Modes and Order Priority Query (Q12)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 2 Query 12 START -- ', qstart ;
select
    l_shipmode,
    sum(case
        when o_orderpriority = '1-URGENT'
            or o_orderpriority = '2-HIGH'
            then 1
        else 0
    end) as high_line_count,
    sum(case
        when o_orderpriority <> '1-URGENT'
            and o_orderpriority <> '2-HIGH'
            then 1
        else 0
    end) as low_line_count

```

```

from
    orders,
    lineitem
where
    o_orderkey = l_orderkey
    and l_shipmode in ('FOB', 'RAIL')
    and l_commitdate < l_receiptdate
    and l_shipdate < l_commitdate
    and l_receiptdate >= '1995-01-01'
    and l_receiptdate < dateadd(year,1,'1995-01-01')

```

```

group by
    l_shipmode
order by
    l_shipmode;
commit ;
set qstop = now(*);
select 'Stream 2 Query 12 STOP -- ', qstop ;
select 'Stream 2 Query 12 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)7.sql      2.1.8.1
-- TPC-H/TPC-R Volume Shipping Query (Q7)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 2 Query 7 START -- ', qstart ;
select
    supp_nation,
    cust_nation,
    l_year,
    sum(volume) as revenue
from
    (
        select
            n1.n_name as supp_nation,
            n2.n_name as cust_nation,
            datepart(year, l_shipdate) as l_year,
            l_extendedprice * (1 - l_discount) as volume
        from
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2

```

```

where
    s_suppkey = l_suppkey
    and o_orderkey = l_orderkey
    and c_custkey = o_custkey
    and s_nationkey = n1.n_nationkey
    and c_nationkey = n2.n_nationkey
    and (
        (n1.n_name = 'ALGERIA' and
         (n2.n_name = 'GERMANY')
         or (n1.n_name = 'GERMANY' and
            (n2.n_name = 'ALGERIA')
            )
        )
    and l_shipdate between '1995-01-01' and
    '1996-12-31'
    ) as shipping
group by
    supp_nation,
    cust_nation,
    l_year
order by
    supp_nation,
    cust_nation,
    l_year;
commit ;
set qstop = now(*);
select 'Stream 2 Query 7 STOP -- ', qstop ;
select 'Stream 2 Query 7 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)13.sql     2.1.8.1
-- TPC-H/TPC-R Customer Distribution Query (Q13)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 2 Query 13 START -- ', qstart ;
select
    c_count,
    count(*) as custdist
from
    (
        select
            c_custkey,
            count(o_orderkey)
        from
            customer left outer join orders on
                c_custkey = o_custkey
                and o_comment not like '%pending
            %accounts%'
        group by
            c_custkey
    ) as c_orders (c_custkey, c_count)
group by
    c_count
order by
    custdist desc,
    c_count desc;

```

```

commit ;
set qstop = now(*);
select 'Stream 2 Query 13 STOP -- ', qstop ;
select 'Stream 2 Query 13 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)18.sql     2.1.8.1
-- TPC-H/TPC-R Large Volume Customer Query (Q18)
-- Function Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 2 Query 18 START -- ', qstart ;
select top 100
    c_name,
    c_custkey,

```

```

o_orderkey,
o_orderdate,
o_totalprice,
sum(l_quantity)
from
customer,
orders,
lineitem
where
o_orderkey in (
select
l_orderkey
from
lineitem
group by
l_orderkey having
sum(l_quantity) > 314
)
and c_custkey = o_custkey
and o_orderkey = l_orderkey
group by
c_name,
c_custkey,
o_orderkey,
o_orderdate,
o_totalprice
order by
o_totalprice desc,
o_orderdate;
commit ;
set qstop = now(*);
select 'Stream 2 Query 18 STOP -- ', qstop ;
select 'Stream 2 Query 18 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)1.sql 2.1.8.1
-- TPC-H/TPC-R Pricing Summary Report Query (Q1)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 2 Query 1 START -- ', qstart ;
select
l_returnflag,
l_linestatus,
sum(l_quantity) as sum_qty,
sum(l_extendedprice) as sum_base_price,
sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
avg(l_quantity) as avg_qty,
avg(l_extendedprice) as avg_price,
avg(l_discount) as avg_disc,
count(*) as count_order
from
lineitem
where
l_shipdate <= dateadd(day,-71,'1998-12-01')
group by
l_returnflag,
l_linestatus
order by
l_returnflag,
l_linestatus;
commit ;
set qstop = now(*);
select 'Stream 2 Query 1 STOP -- ', qstop ;
select 'Stream 2 Query 1 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)4.sql 2.1.8.1
-- TPC-H/TPC-R Order Priority Checking Query (Q4)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 2 Query 4 START -- ', qstart ;
select
o_orderpriority,
count(*) as order_count
from
orders
where
o_orderdate >= '1996-02-01'
and o_orderdate < dateadd(month,3,'1996-02-01')
and exists (
select
*
from
lineitem
where
l_orderkey = o_orderkey
and l_commitdate < l_receiptdate
)
group by
o_orderpriority
order by
o_orderpriority;
commit ;
set qstop = now(*);
select 'Stream 2 Query 4 STOP -- ', qstop ;
select 'Stream 2 Query 4 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)20.sql 2.1.8.1
-- TPC-H/TPC-R Potential Part Promotion Query (Q20)
-- Function Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 2 Query 20 START -- ', qstart ;
select
s_name,
s_address
from
supplier,
nation
where
s_suppkey in (
select
ps_suppkey
from
partsupp
where
ps_partkey in (
select
p_partkey
from
part
where
p_name like 'beige%'
)
and ps_availqty > (
select
0.5 * sum(l_quantity)
from
lineitem
where
l_partkey = ps_partkey
and l_suppkey =
ps_suppkey
and l_shipdate >=
'1994-01-01'
and l_shipdate < dateadd(year,1,'1994-01-01')
)
)
and s_nationkey = n_nationkey

```

```

        and n_name = 'JORDAN'
order by
    s_name;
commit ;
set qstop = now(*);
select 'Stream 2 Query 20 STOP -- ', qstop ;
select 'Stream 2 Query 20 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)3.sql      2.1.8.1
-- TPC-H/TPC-R Shipping Priority Query (Q3)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 2 Query 3 START -- ', qstart ;
select top 10
    l_orderkey,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    o_orderdate,
    o_shippriority
from
    customer,
    orders,
    lineitem
where
    c_mktsegment = 'BUILDING'
    and c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate < '1995-03-20'
    and l_shipdate > '1995-03-20'
group by
    l_orderkey,
    o_orderdate,
    o_shippriority
order by
    revenue desc,
    o_orderdate;
commit ;
set qstop = now(*);
select 'Stream 2 Query 3 STOP -- ', qstop ;
select 'Stream 2 Query 3 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)11.sql     2.1.8.1
-- TPC-H/TPC-R Important Stock Identification Query (Q11)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 2 Query 11 START -- ', qstart ;
select
    ps_partkey,
    sum(ps_supplycost * ps_availqty) as value
from
    partsupp,
    supplier,
    nation
where
    ps_suppkey = s_suppkey
    and s_nationkey = n_nationkey
    and n_name = 'ARGENTINA'
group by
    ps_partkey having
        sum(ps_supplycost * ps_availqty) > (
            select
                sum(ps_supplycost * ps_availqty) *
0.0000003333
            from
                partsupp,
                supplier,
                nation
        where

```

```

        ps_suppkey = s_suppkey
        and s_nationkey = n_nationkey
        and n_name = 'ARGENTINA'
    )
order by
    value desc;
commit ;
set qstop = now(*);
select 'Stream 2 Query 11 STOP -- ', qstop ;
select 'Stream 2 Query 11 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)21.sql     2.1.8.1
-- TPC-H/TPC-R Suppliers Who Kept Orders Waiting Query (Q21)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 2 Query 21 START -- ', qstart ;
select top 100
    s_name,
    count(*) as numwait
from
    supplier,
    lineitem l1,
    orders,
    nation
where
    s_suppkey = l1_suppkey
    and o_orderkey = l1_orderkey
    and o_orderstatus = 'F'
    and l1_receiptdate > l1_commitdate
    and exists (
        select
            *
        from
            lineitem l2
        where
            l2_orderkey = l1_orderkey
            and l2_suppkey <> l1_suppkey
    )
    and not exists (
        select
            *
        from
            lineitem l3
        where
            l3_orderkey = l1_orderkey
            and l3_suppkey <> l1_suppkey
            and l3_receiptdate > l3_commitdate
    )
    and s_nationkey = n_nationkey
    and n_name = 'ETHIOPIA'
group by
    s_name
order by
    numwait desc,
    s_name;
commit ;
set qstop = now(*);
select 'Stream 2 Query 21 STOP -- ', qstop ;
select 'Stream 2 Query 21 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;

select 'Stream completion time -- ',now(*) as "time";
disconnect;
exit;
=====

```

stream3.sql

```
-- using 115192429 as a seed to the RNG
create variable qstart timestamp;
create variable qstop timestamp;
select 'Stream begin time -- ',now(*) as "time";
-- @(#)8.sql 2.1.8.1
-- TPC-H/TPC-R National Market Share Query (Q8)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 3 Query 8 START -- ', qstart ;
select
    o_year,
    sum(case
        when nation = 'UNITED STATES' then
            volume
        else 0
        end) / sum(volume) as mkt_share
from
    (
        select
            datepart(year, o_orderdate) as
o_year,
            l_extendedprice * (1 -
l_discount) as volume,
            n2.n_name as nation
        from
            part,
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2,
            region
        where
            p_partkey = l_partkey
            and s_suppkey = l_suppkey
            and l_orderkey = o_orderkey
            and o_custkey = c_custkey
            and c_nationkey =
n1.n_nationkey
            and n1.n_regionkey =
r_regionkey
            and r_name = 'AMERICA'
            and s_nationkey =
n2.n_nationkey
            and o_orderdate between
'1995-01-01' and '1996-12-31'
            and p_type = 'STANDARD
BURNISHED BRASS'
        ) as all_nations
group by
    o_year
order by
    o_year;
commit ;
set qstop = now(*) ;
select 'Stream 3 Query 8 STOP -- ', qstop ;
select 'Stream 3 Query 8 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)5.sql 2.1.8.1
-- TPC-H/TPC-R Local Supplier Volume Query (Q5)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 3 Query 5 START -- ', qstart ;
select
    n_name,
    sum(l_extendedprice * (1 - l_discount)) as
```

```
revenue
from
    customer,
    orders,
    lineitem,
    supplier,
    nation,
    region
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and l_suppkey = s_suppkey
    and c_nationkey = s_nationkey
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'ASIA'
    and o_orderdate >= '1996-01-01'
    and o_orderdate < dateadd(year,1,'1996-01-01')
group by
    n_name
order by
    revenue desc;
commit ;
set qstop = now(*) ;
select 'Stream 3 Query 5 STOP -- ', qstop ;
select 'Stream 3 Query 5 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)4.sql 2.1.8.1
-- TPC-H/TPC-R Order Priority Checking Query (Q4)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 3 Query 4 START -- ', qstart ;
select
    o_orderpriority,
    count(*) as order_count
from
    orders
where
    o_orderdate >= '1993-11-01'
    and o_orderdate <
dateadd(month,3,'1993-11-01')
    and exists (
        select
            *
        from
            lineitem
        where
            l_orderkey = o_orderkey
            and l_commitdate <
l_receiptdate
    )
group by
    o_orderpriority
order by
    o_orderpriority;
commit ;
set qstop = now(*) ;
select 'Stream 3 Query 4 STOP -- ', qstop ;
select 'Stream 3 Query 4 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)6.sql 2.1.8.1
-- TPC-H/TPC-R Forecasting Revenue Change Query (Q6)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 3 Query 6 START -- ', qstart ;
select
    sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
```



```

l_shipdate >= '1996-01-01'
and l_shipdate < dateadd(year,1,'1996-01-01')
and l_discount between 0.07 - 0.01 and 0.07 +
0.01
and l_quantity < 25;
commit ;
set qstop = now(*) ;
select 'Stream 3 Query 6 STOP -- ', qstop ;
select 'Stream 3 Query 6 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)17.sql 2.1.8.1
-- TPC-H/TPC-R Small-Quantity-Order Revenue Query
(Q17)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 3 Query 17 START -- ', qstart ;
select
sum(l_extendedprice) / 7.0 as avg_yearly
from
lineitem,
part
where
p_partkey = l_partkey
and p_brand = 'Brand#41'
and p_container = 'JUMBO JAR'
and l_quantity < (
select
0.2 * avg(l_quantity)
from
lineitem
where
l_partkey = p_partkey
);
commit ;
set qstop = now(*) ;
select 'Stream 3 Query 17 STOP -- ', qstop ;
select 'Stream 3 Query 17 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)7.sql 2.1.8.1
-- TPC-H/TPC-R Volume Shipping Query (Q7)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 3 Query 7 START -- ', qstart ;
select
supp_nation,
cust_nation,
l_year,
sum(volume) as revenue
from
(
select
n1.n_name as supp_nation,
n2.n_name as cust_nation,
datepart(year, l_shipdate) as
l_year,
l_extendedprice * (1 -
l_discount) as volume
from
supplier,
lineitem,
orders,
customer,
nation n1,
nation n2
where
s_suppkey = l_suppkey
and o_orderkey = l_orderkey
and c_custkey = o_custkey
and s_nationkey =
n1.n_nationkey

```

```

and c_nationkey =
n2.n_nationkey
and (
(n1.n_name = 'PERU' and
n2.n_name = 'UNITED STATES')
or (n1.n_name = 'UNITED
STATES' and n2.n_name = 'PERU')
)
and l_shipdate between
'1995-01-01' and '1996-12-31'
) as shipping
group by
supp_nation,
cust_nation,
l_year
order by
supp_nation,
cust_nation,
l_year;
commit ;
set qstop = now(*) ;
select 'Stream 3 Query 7 STOP -- ', qstop ;
select 'Stream 3 Query 7 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)1.sql 2.1.8.1
-- TPC-H/TPC-R Pricing Summary Report Query (Q1)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 3 Query 1 START -- ', qstart ;
select
l_returnflag,
l_linestatus,
sum(l_quantity) as sum_qty,
sum(l_extendedprice) as sum_base_price,
sum(l_extendedprice * (1 - l_discount)) as
sum_disc_price,
sum(l_extendedprice * (1 - l_discount) * (1 +
l_tax)) as sum_charge,
avg(l_quantity) as avg_qty,
avg(l_extendedprice) as avg_price,
avg(l_discount) as avg_disc,
count(*) as count_order
from
lineitem
where
l_shipdate <= dateadd(day,-79,'1998-12-01')
group by
l_returnflag,
l_linestatus
order by
l_returnflag,
l_linestatus;
commit ;
set qstop = now(*) ;
select 'Stream 3 Query 1 STOP -- ', qstop ;
select 'Stream 3 Query 1 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)18.sql 2.1.8.1
-- TPC-H/TPC-R Large Volume Customer Query (Q18)
-- Function Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 3 Query 18 START -- ', qstart ;
select top 100
c_name,
c_custkey,
o_orderkey,
o_orderdate,
o_totalprice,
sum(l_quantity)
from

```

```

customer,
orders,
lineitem
where
  o_orderkey in (
    select
      l_orderkey
    from
      lineitem
    group by
      l_orderkey having
        sum(l_quantity) > 312
  )
  and c_custkey = o_custkey
  and o_orderkey = l_orderkey
group by
  c_name,
  c_custkey,
  o_orderkey,
  o_orderdate,
  o_totalprice
order by
  o_totalprice desc,
  o_orderdate;
commit ;
set qstop = now(*);
select 'Stream 3 Query 18 STOP -- ', qstop ;
select 'Stream 3 Query 18 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)22.sql 2.1.8.1
-- TPC-H/TPC-R Global Sales Opportunity Query (Q22)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 3 Query 22 START -- ', qstart ;
select
  cntrycode,
  count(*) as numcust,
  sum(c_acctbal) as totacctbal
from
  (
    select
      substring(c_phone,1,2) as
cntrycode,
      c_acctbal
    from
      customer
    where
      substring(c_phone,1,2) in
        ('12', '31', '23', '14',
'20', '22', '19')
      and c_acctbal > (
        select
          avg(c_acctbal)
        from
          customer
        where
          c_acctbal > 0.00
          and
substring(c_phone,1,2) in
          ('12',
'31', '23', '14', '20', '22', '19')
        )
      and not exists (
        select
          *
        from
          orders
        where
          o_custkey =
c_custkey
      )
    ) as custsale
group by
  cntrycode

```

```

order by
  cntrycode;
commit ;
set qstop = now(*);
select 'Stream 3 Query 22 STOP -- ', qstop ;
select 'Stream 3 Query 22 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)14.sql 2.1.8.1
-- TPC-H/TPC-R Promotion Effect Query (Q14)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 3 Query 14 START -- ', qstart ;
select
  100.00 * sum(case
    when p_type like 'PROMO%'
      then l_extendedprice * (1 -
l_discount)
    else 0
  end) / sum(l_extendedprice * (1 - l_discount))
as promo_revenue
from
  lineitem,
  part
where
  l_partkey = p_partkey
  and l_shipdate >= '1996-01-01'
  and l_shipdate <
dateadd(month,1,'1996-01-01');
commit ;
set qstop = now(*);
select 'Stream 3 Query 14 STOP -- ', qstop ;
select 'Stream 3 Query 14 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)9.sql 2.1.8.1
-- TPC-H/TPC-R Product Type Profit Measure Query (Q9)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 3 Query 9 START -- ', qstart ;
select
  nation,
  o_year,
  sum(amount) as sum_profit
from
  (
    select
      n_name as nation,
      datepart(year, o_orderdate) as
o_year,
      l_extendedprice * (1 -
l_discount) - ps_supplycost * l_quantity as amount
    from
      part,
      supplier,
      lineitem,
      partsupp,
      orders,
      nation
    where
      s_suppkey = l_suppkey
      and ps_suppkey = l_suppkey
      and ps_partkey = l_partkey
      and p_partkey = l_partkey
      and o_orderkey = l_orderkey
      and s_nationkey = n_nationkey
      and p_name like '%beige%'
    ) as profit
group by
  nation,
  o_year
order by

```

```

        nation,
        o_year desc;
commit ;
set qstop = now(*) ;
select 'Stream 3 Query 9 STOP -- ', qstop ;
select 'Stream 3 Query 9 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)10.sql 2.1.8.1
-- TPC-H/TPC-R Returned Item Reporting Query (Q10)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 3 Query 10 START -- ', qstart ;
select top 20
        c_custkey,
        c_name,
        sum(l_extendedprice * (1 - l_discount)) as
revenue,
        c_acctbal,
        n_name,
        c_address,
        c_phone,
        c_comment
from
        customer,
        orders,
        lineitem,
        nation
where
        c_custkey = o_custkey
        and l_orderkey = o_orderkey
        and o_orderdate >= '1994-09-01'
        and o_orderdate <
dateadd(month,3,'1994-09-01')
        and l_returnflag = 'R'
        and c_nationkey = n_nationkey
group by
        c_custkey,
        c_name,
        c_acctbal,
        c_phone,
        n_name,
        c_address,
        c_comment
order by
        revenue desc;
commit ;
set qstop = now(*) ;
select 'Stream 3 Query 10 STOP -- ', qstop ;
select 'Stream 3 Query 10 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)15.sql 2.1.8.1
-- TPC-H/TPC-R Top Supplier Query (Q15)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 3 Query 15 START -- ', qstart ;
create view revenue3 (supplier_no, total_revenue) as
select
        l_suppkey,
        sum(l_extendedprice * (1 - l_discount))
from
        lineitem
where
        l_shipdate >= '1997-01-01'
        and l_shipdate <
dateadd(month,3,'1997-01-01')
group by
        l_suppkey;

select
        s_suppkey,

```

```

        s_name,
        s_address,
        s_phone,
        total_revenue
from
        supplier,
        revenue3
where
        s_suppkey = supplier_no
        and total_revenue = (
                select
                        max(total_revenue)
                from
                        revenue3
                )
order by
        s_suppkey;

drop view revenue3;
commit ;
set qstop = now(*) ;
select 'Stream 3 Query 15 STOP -- ', qstop ;
select 'Stream 3 Query 15 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)11.sql 2.1.8.1
-- TPC-H/TPC-R Important Stock Identification Query (Q11)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 3 Query 11 START -- ', qstart ;
select
        ps_partkey,
        sum(ps_supplycost * ps_availqty) as value
from
        partsupp,
        supplier,
        nation
where
        ps_suppkey = s_suppkey
        and s_nationkey = n_nationkey
        and n_name = 'KENYA'
group by
        ps_partkey having
                sum(ps_supplycost * ps_availqty) > (
                        select
                                sum(ps_supplycost *
ps_availqty) * 0.0000003333
                        from
                                partsupp,
                                supplier,
                                nation
                        where
                                ps_suppkey = s_suppkey
                                and s_nationkey =
n_nationkey
                                and n_name = 'KENYA'
                        )
order by
        value desc;
commit ;
set qstop = now(*) ;
select 'Stream 3 Query 11 STOP -- ', qstop ;
select 'Stream 3 Query 11 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)20.sql 2.1.8.1
-- TPC-H/TPC-R Potential Part Promotion Query (Q20)
-- Function Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 3 Query 20 START -- ', qstart ;
select

```

```

s_name,
s_address
from
supplier,
nation
where
s_suppkey in (
select
ps_suppkey
from
partsupp
where
ps_partkey in (
select
p_partkey
from
part
where
p_name like
'lemon%'
)
and ps_availqty > (
select
0.5 *
sum(l_quantity)
from
lineitem
where
l_partkey =
ps_partkey
and l_suppkey =
ps_suppkey
and l_shipdate
>= '1997-01-01'
and l_shipdate
< dateadd(year,1,'1997-01-01')
)
and s_nationkey = n_nationkey
and n_name = 'BRAZIL'
order by
s_name;
commit ;
set qstop = now(*);
select 'Stream 3 Query 20 STOP -- ', qstop ;
select 'Stream 3 Query 20 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)2.sql 2.1.8.2
-- TPC-H/TPC-R Minimum Cost Supplier Query (Q2)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 3 Query 2 START -- ', qstart ;
select top 100
s_acctbal,
s_name,
n_name,
p_partkey,
p_mfgr,
s_address,
s_phone,
s_comment
from
part,
supplier,
partsupp,
nation,
region
where
p_partkey = ps_partkey
and s_suppkey = ps_suppkey
and p_size = 22
and p_type like '%STEEL'
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = 'AFRICA'
and ps_supplycost = (
select
min(ps_supplycost)
from
partsupp,
supplier,
nation,
region
where
p_partkey = ps_partkey
and s_suppkey = ps_suppkey
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = 'AFRICA'
)
order by
s_acctbal desc,
n_name,
s_name,
p_partkey;
commit ;
set qstop = now(*);
select 'Stream 3 Query 2 STOP -- ', qstop ;
select 'Stream 3 Query 2 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)21.sql 2.1.8.1
-- TPC-H/TPC-R Suppliers Who Kept Orders Waiting Query
(Q21)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 3 Query 21 START -- ', qstart ;
select top 100
s_name,
count(*) as numwait
from
supplier,
lineitem l1,
orders,
nation
where
s_suppkey = l1.l_suppkey
and o_orderkey = l1.l_orderkey
and o_orderstatus = 'F'
and l1.l_receiptdate > l1.l_commitdate
and exists (
select
*
from
lineitem l2
where
l2.l_orderkey = l1.l_orderkey
and l2.l_suppkey <>
l1.l_suppkey
)
and not exists (
select
*
from
lineitem l3
where
l3.l_orderkey = l1.l_orderkey
and l3.l_suppkey <>
l1.l_suppkey
and l3.l_receiptdate >
l3.l_commitdate
)
and s_nationkey = n_nationkey
and n_name = 'RUSSIA'
group by
s_name
order by
numwait desc,
s_name;

```

```

commit ;
set qstop = now(*) ;
select 'Stream 3 Query 21 STOP -- ', qstop ;
select 'Stream 3 Query 21 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)19.sql 2.1.8.1
-- TPC-H/TPC-R Discounted Revenue Query (Q19)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 3 Query 19 START -- ', qstart ;
select
sum(l_extendedprice* (1 - l_discount)) as
revenue
from
lineitem,
part
where
(
p_partkey = l_partkey
and p_brand = 'Brand#43'
and p_container in ('SM CASE', 'SM
BOX', 'SM PACK', 'SM PKG')
and l_quantity >= 9 and l_quantity <= 9
+ 10
and p_size between 1 and 5
and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN
PERSON'
)
or
(
p_partkey = l_partkey
and p_brand = 'Brand#51'
and p_container in ('MED BAG', 'MED
BOX', 'MED PKG', 'MED PACK')
and l_quantity >= 11 and l_quantity <=
11 + 10
and p_size between 1 and 10
and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN
PERSON'
)
or
(
p_partkey = l_partkey
and p_brand = 'Brand#53'
and p_container in ('LG CASE', 'LG
BOX', 'LG PACK', 'LG PKG')
and l_quantity >= 27 and l_quantity <=
27 + 10
and p_size between 1 and 15
and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN
PERSON'
);
commit ;
set qstop = now(*) ;
select 'Stream 3 Query 19 STOP -- ', qstop ;
select 'Stream 3 Query 19 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)13.sql 2.1.8.1
-- TPC-H/TPC-R Customer Distribution Query (Q13)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 3 Query 13 START -- ', qstart ;
select
c_count,
count(*) as custdist
from
(

```

```

select
c_custkey,
count(o_orderkey)
from
customer left outer join orders
on
c_custkey = o_custkey
and o_comment not like
'%pending%accounts%'
group by
c_custkey
) as c_orders (c_custkey, c_count)
group by
c_count
order by
custdist desc,
c_count desc ;
commit ;

```

```

set qstop = now(*) ;
select 'Stream 3 Query 13 STOP -- ', qstop ;
select 'Stream 3 Query 13 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)16.sql 2.1.8.1
-- TPC-H/TPC-R Parts/Supplier Relationship Query (Q16)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 3 Query 16 START -- ', qstart ;
select
p_brand,
p_type,
p_size,
count(distinct ps_suppkey) as supplier_cnt
from
partsupp,
part
where
p_partkey = ps_partkey
and p_brand <> 'Brand#44'
and p_type not like 'STANDARD POLISHED%'
and p_size in (7, 24, 11, 50, 38, 14, 41, 8)
and ps_suppkey not in (
select
s_suppkey
from
supplier
where
s_comment like '%Customer
%Complaints%'
)
group by
p_brand,
p_type,
p_size
order by
supplier_cnt desc,
p_brand,
p_type,
p_size ;
commit ;

```

```

set qstop = now(*) ;
select 'Stream 3 Query 16 STOP -- ', qstop ;
select 'Stream 3 Query 16 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)12.sql 2.1.8.1
-- TPC-H/TPC-R Shipping Modes and Order Priority Query
(Q12)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 3 Query 12 START -- ', qstart ;
select

```

```

l_shipmode,
sum(case
  when o_orderpriority = '1-URGENT'
    or o_orderpriority = '2-HIGH'
    then 1
  else 0
end) as high_line_count,
sum(case
  when o_orderpriority <> '1-URGENT'
    and o_orderpriority <> '2-HIGH'
    then 1
  else 0
end) as low_line_count
from
  orders,
  lineitem
where
  o_orderkey = l_orderkey
  and l_shipmode in ('MAIL', 'RAIL')
  and l_commitdate < l_receiptdate
  and l_shipdate < l_commitdate
  and l_receiptdate >= '1995-01-01'
  and l_receiptdate <
dateadd(year,1,'1995-01-01')
group by
  l_shipmode
order by
  l_shipmode;
commit ;
set qstop = now(*)
select 'Stream 3 Query 12 STOP -- ', qstop ;
select 'Stream 3 Query 12 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)3.sql 2.1.8.1
-- TPC-H/TPC-R Shipping Priority Query (Q3)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*)
select 'Stream 3 Query 3 START -- ', qstart ;
select top 10
  l_orderkey,
  sum(l_extendedprice * (1 - l_discount)) as
revenue,
  o_orderdate,
  o_shippriority
from
  customer,
  orders,
  lineitem
where
  c_mktsegment = 'HOUSEHOLD'
  and c_custkey = o_custkey
  and l_orderkey = o_orderkey
  and o_orderdate < '1995-03-06'
  and l_shipdate > '1995-03-06'
group by
  l_orderkey,
  o_orderdate,
  o_shippriority
order by
  revenue desc,
  o_orderdate;
commit ;
set qstop = now(*)
select 'Stream 3 Query 3 STOP -- ', qstop ;
select 'Stream 3 Query 3 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;

select 'Stream completion time -- ',now(*) as "time";
disconnect;
exit;

```

stream4.sql

```

=====
-- using 115192430 as a seed to the RNG
create variable qstart timestamp;
create variable qstop timestamp;
select 'Stream begin time -- ',now(*) as "time";
-- @(#)5.sql 2.1.8.1
-- TPC-H/TPC-R Local Supplier Volume Query (Q5)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*)
select 'Stream 4 Query 5 START -- ', qstart ;
select
  n_name,
  sum(l_extendedprice * (1 - l_discount)) as
revenue
from
  customer,
  orders,
  lineitem,
  supplier,
  nation,
  region
where
  c_custkey = o_custkey
  and l_orderkey = o_orderkey
  and l_suppkey = s_suppkey
  and c_nationkey = s_nationkey
  and s_nationkey = n_nationkey
  and n_regionkey = r_regionkey
  and r_name = 'EUROPE'
  and o_orderdate >= '1996-01-01'
  and o_orderdate < dateadd(year,1,'1996-01-01')
group by
  n_name
order by
  revenue desc;
commit ;
set qstop = now(*)
select 'Stream 4 Query 5 STOP -- ', qstop ;
select 'Stream 4 Query 5 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)21.sql 2.1.8.1
-- TPC-H/TPC-R Suppliers Who Kept Orders Waiting Query
(Q21)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*)
select 'Stream 4 Query 21 START -- ', qstart ;
select top 100
  s_name,
  count(*) as numwait
from
  supplier,
  lineitem l1,
  orders,
  nation
where
  s_suppkey = l1.l_suppkey
  and o_orderkey = l1.l_orderkey
  and o_orderstatus = 'F'
  and l1.l_receiptdate > l1.l_commitdate
  and exists (
    select
      *
    from
      lineitem l2
    where
      l2.l_orderkey = l1.l_orderkey
      and l2.l_suppkey <>

```

```

11.1_suppkey
    )
    and not exists (
        select
            *
        from
            lineitem l3
        where
            l3.l_orderkey = l1.1_orderkey
            and l3.l_suppkey <>
11.1_suppkey
            and l3.l_receiptdate >
13.1_commitdate
    )
    and s_nationkey = n_nationkey
    and n_name = 'KENYA'
group by
    s_name
order by
    numwait desc,
    s_name;
commit ;
set qstop = now(*);
select 'Stream 4 Query 21 STOP -- ', qstop ;
select 'Stream 4 Query 21 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)14.sql 2.1.8.1
-- TPC-H/TPC-R Promotion Effect Query (Q14)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 4 Query 14 START -- ', qstart ;
select
    100.00 * sum(case
        when p_type like 'PROMO%'
            then l_extendedprice * (1 -
l_discount)
        else 0
        end) / sum(l_extendedprice * (1 - l_discount))
as promo_revenue
from
    lineitem,
    part
where
    l_partkey = p_partkey
    and l_shipdate >= '1996-01-01'
    and l_shipdate <
dateadd(month,1,'1996-01-01');
commit ;
set qstop = now(*);
select 'Stream 4 Query 14 STOP -- ', qstop ;
select 'Stream 4 Query 14 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)19.sql 2.1.8.1
-- TPC-H/TPC-R Discounted Revenue Query (Q19)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 4 Query 19 START -- ', qstart ;
select
    sum(l_extendedprice* (1 - l_discount)) as
revenue
from
    lineitem,
    part
where
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#45'
        and p_container in ('SM CASE', 'SM
BOX', 'SM PACK', 'SM PKG')
        and l_quantity >= 4 and l_quantity <= 4
+ 10
        and p_size between 1 and 5
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN
PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#34'
        and p_container in ('MED BAG', 'MED
BOX', 'MED PKG', 'MED PACK')
        and l_quantity >= 12 and l_quantity <=
12 + 10
        and p_size between 1 and 10
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN
PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#42'
        and p_container in ('LG CASE', 'LG
BOX', 'LG PACK', 'LG PKG')
        and l_quantity >= 23 and l_quantity <=
23 + 10
        and p_size between 1 and 15
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN
PERSON'
    );
commit ;
set qstop = now(*);
select 'Stream 4 Query 19 STOP -- ', qstop ;
select 'Stream 4 Query 19 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)15.sql 2.1.8.1
-- TPC-H/TPC-R Top Supplier Query (Q15)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 4 Query 15 START -- ', qstart ;
create view revenue4 (supplier_no, total_revenue) as
select
    l_suppkey,
    sum(l_extendedprice * (1 - l_discount))
from
    lineitem
where
    l_shipdate >= '1995-01-01'
    and l_shipdate <
dateadd(month,3,'1995-01-01')
group by
    l_suppkey;

select
    s_suppkey,
    s_name,
    s_address,
    s_phone,
    total_revenue
from
    supplier,
    revenue4
where
    s_suppkey = supplier_no
    and total_revenue = (
        select
            max(total_revenue)
        from
            revenue4
    )
order by
    s_suppkey;

```

```

drop view revenue4;
commit ;
set qstop = now(*) ;
select 'Stream 4 Query 15 STOP -- ', qstop ;
select 'Stream 4 Query 15 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)17.sql 2.1.8.1
-- TPC-H/TPC-R Small-Quantity-Order Revenue Query
(Q17)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 4 Query 17 START -- ', qstart ;
select
sum(l_extendedprice) / 7.0 as avg_yearly
from
lineitem,
part
where
p_partkey = l_partkey
and p_brand = 'Brand#43'
and p_container = 'JUMBO CAN'
and l_quantity < (
select
0.2 * avg(l_quantity)
from
lineitem
where
l_partkey = p_partkey
);
commit ;
set qstop = now(*) ;
select 'Stream 4 Query 17 STOP -- ', qstop ;
select 'Stream 4 Query 17 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)12.sql 2.1.8.1
-- TPC-H/TPC-R Shipping Modes and Order Priority Query
(Q12)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 4 Query 12 START -- ', qstart ;
select
l_shipmode,
sum(case
when o_orderpriority = '1-URGENT'
or o_orderpriority = '2-HIGH'
then 1
else 0
end) as high_line_count,
sum(case
when o_orderpriority <> '1-URGENT'
and o_orderpriority <> '2-HIGH'
then 1
else 0
end) as low_line_count
from
orders,
lineitem
where
o_orderkey = l_orderkey
and l_shipmode in ('TRUCK', 'RAIL')
and l_commitdate < l_receiptdate
and l_shipdate < l_commitdate
and l_receiptdate >= '1996-01-01'
and l_receiptdate <
dateadd(year,1,'1996-01-01')
group by
l_shipmode
order by
l_shipmode;

```

```

commit ;
set qstop = now(*) ;
select 'Stream 4 Query 12 STOP -- ', qstop ;
select 'Stream 4 Query 12 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)6.sql 2.1.8.1
-- TPC-H/TPC-R Forecasting Revenue Change Query (Q6)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 4 Query 6 START -- ', qstart ;
select
sum(l_extendedprice * l_discount) as revenue
from
lineitem
where
l_shipdate >= '1996-01-01'
and l_shipdate < dateadd(year,1,'1996-01-01')
and l_discount between 0.05 - 0.01 and 0.05 +
0.01
and l_quantity < 25;
commit ;
set qstop = now(*) ;
select 'Stream 4 Query 6 STOP -- ', qstop ;
select 'Stream 4 Query 6 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)4.sql 2.1.8.1
-- TPC-H/TPC-R Order Priority Checking Query (Q4)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 4 Query 4 START -- ', qstart ;
select
o_orderpriority,
count(*) as order_count
from
orders
where
o_orderdate >= '1996-06-01'
and o_orderdate <
dateadd(month,3,'1996-06-01')
and exists (
select
*
from
lineitem
where
l_orderkey = o_orderkey
and l_commitdate <
l_receiptdate
)
group by
o_orderpriority
order by
o_orderpriority;
commit ;
set qstop = now(*) ;
select 'Stream 4 Query 4 STOP -- ', qstop ;
select 'Stream 4 Query 4 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)9.sql 2.1.8.1
-- TPC-H/TPC-R Product Type Profit Measure Query (Q9)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 4 Query 9 START -- ', qstart ;
select
nation,
o_year,

```



```

sum(amount) as sum_profit
from
(
select
n_name as nation,
datepart(year, o_orderdate) as
o_year,
l_extendedprice * (1 -
l_discount) - ps_supplycost * l_quantity as amount
from
part,
supplier,
lineitem,
partsupp,
orders,
nation
where
s_suppkey = l_suppkey
and ps_suppkey = l_suppkey
and ps_partkey = l_partkey
and p_partkey = l_partkey
and o_orderkey = l_orderkey
and s_nationkey = n_nationkey
and p_name like '%white%'
) as profit
group by
nation,
o_year
order by
nation,
o_year desc;
commit ;
set qstop = now(*);
select 'Stream 4 Query 9 STOP -- ', qstop ;
select 'Stream 4 Query 9 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)8.sql 2.1.8.1
-- TPC-H/TPC-R National Market Share Query (Q8)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 4 Query 8 START -- ', qstart ;
select
o_year,
sum(case
when nation = 'MOZAMBIQUE' then volume
else 0
end) / sum(volume) as mkt_share
from
(
select
datepart(year, o_orderdate) as
o_year,
l_extendedprice * (1 -
l_discount) as volume,
n2.n_name as nation
from
part,
supplier,
lineitem,
orders,
customer,
nation n1,
nation n2,
region
where
p_partkey = l_partkey
and s_suppkey = l_suppkey
and l_orderkey = o_orderkey
and o_custkey = c_custkey
and c_nationkey =
n1.n_nationkey
and n1.n_regionkey =
r_regionkey
and r_name = 'AFRICA'
and s_nationkey =
n2.n_nationkey
and o_orderdate between
'1995-01-01' and '1996-12-31'
and p_type = 'PROMO BRUSHED
STEEL'
) as all_nations
group by
o_year
order by
o_year;
commit ;
set qstop = now(*);
select 'Stream 4 Query 8 STOP -- ', qstop ;
select 'Stream 4 Query 8 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)16.sql 2.1.8.1
-- TPC-H/TPC-R Parts/Supplier Relationship Query (Q16)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 4 Query 16 START -- ', qstart ;
select
p_brand,
p_type,
p_size,
count(distinct ps_suppkey) as supplier_cnt
from
partsupp,
part
where
p_partkey = ps_partkey
and p_brand <> 'Brand#34'
and p_type not like 'LARGE ANODIZED%'
and p_size in (42, 4, 11, 20, 40, 8, 17, 13)
and ps_suppkey not in (
select
s_suppkey
from
supplier
where
s_comment like '%Customer
%Complaints%'
)
group by
p_brand,
p_type,
p_size
order by
supplier_cnt desc,
p_brand,
p_type,
p_size;
commit ;
set qstop = now(*);
select 'Stream 4 Query 16 STOP -- ', qstop ;
select 'Stream 4 Query 16 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)11.sql 2.1.8.1
-- TPC-H/TPC-R Important Stock Identification Query
(Q11)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 4 Query 11 START -- ', qstart ;
select
ps_partkey,
sum(ps_supplycost * ps_availqty) as value
from
partsupp,
supplier,
nation

```

```

where
    ps_suppkey = s_suppkey
    and s_nationkey = n_nationkey
    and n_name = 'BRAZIL'
group by
    ps_partkey having
        sum(ps_supplycost * ps_availqty) > (
            select
                sum(ps_supplycost *
                    ps_availqty) * 0.0000003333
            from
                partsupp,
                supplier,
                nation
            where
                ps_suppkey = s_suppkey
                and s_nationkey =
                    n_nationkey
        )
order by
    value desc;
commit ;
set qstop = now(*);
select 'Stream 4 Query 11 STOP -- ', qstop ;
select 'Stream 4 Query 11 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)2.sql 2.1.8.2
-- TPC-H/TPC-R Minimum Cost Supplier Query (Q2)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 4 Query 2 START -- ', qstart ;
select top 100
    s_acctbal,
    s_name,
    n_name,
    p_partkey,
    p_mfgr,
    s_address,
    s_phone,
    s_comment
from
    part,
    supplier,
    partsupp,
    nation,
    region
where
    p_partkey = ps_partkey
    and s_suppkey = ps_suppkey
    and p_size = 10
    and p_type like '%BRASS'
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'EUROPE'
    and ps_supplycost = (
        select
            min(ps_supplycost)
        from
            partsupp,
            supplier,
            nation,
            region
        where
            p_partkey = ps_partkey
            and s_suppkey = ps_suppkey
            and s_nationkey = n_nationkey
            and n_regionkey = r_regionkey
            and r_name = 'EUROPE'
    )
order by
    s_acctbal desc,
    n_name,
    s_name,

```

```

        p_partkey;
commit ;
set qstop = now(*);
select 'Stream 4 Query 2 STOP -- ', qstop ;
select 'Stream 4 Query 2 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)10.sql 2.1.8.1
-- TPC-H/TPC-R Returned Item Reporting Query (Q10)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 4 Query 10 START -- ', qstart ;
select top 20
    c_custkey,
    c_name,
    sum(l_extendedprice * (1 - l_discount)) as
revenue,
    c_acctbal,
    n_name,
    c_address,
    c_phone,
    c_comment
from
    customer,
    orders,
    lineitem,
    nation
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate >= '1993-07-01'
    and o_orderdate <
dateadd(month,3,'1993-07-01')
    and l_returnflag = 'R'
    and c_nationkey = n_nationkey
group by
    c_custkey,
    c_name,
    c_acctbal,
    c_phone,
    n_name,
    c_address,
    c_comment
order by
    revenue desc;
commit ;
set qstop = now(*);
select 'Stream 4 Query 10 STOP -- ', qstop ;
select 'Stream 4 Query 10 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)18.sql 2.1.8.1
-- TPC-H/TPC-R Large Volume Customer Query (Q18)
-- Function Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 4 Query 18 START -- ', qstart ;
select top 100
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice,
    sum(l_quantity)
from
    customer,
    orders,
    lineitem
where
    o_orderkey in (
        select
            l_orderkey
        from

```

```

                lineitem
            group by
                l_orderkey having
                    sum(l_quantity) > 314
        )
        and c_custkey = o_custkey
        and o_orderkey = l_orderkey
group by
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice
order by
    o_totalprice desc,
    o_orderdate;
commit ;
set qstop = now(*);
select 'Stream 4 Query 18 STOP -- ', qstop ;
select 'Stream 4 Query 18 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)1.sql 2.1.8.1
-- TPC-H/TPC-R Pricing Summary Report Query (Q1)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 4 Query 1 START -- ', qstart ;
select
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as
sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 +
l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineitem
where
    l_shipdate <= dateadd(day,-87,'1998-12-01')
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus;
commit ;
set qstop = now(*);
select 'Stream 4 Query 1 STOP -- ', qstop ;
select 'Stream 4 Query 1 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)13.sql 2.1.8.1
-- TPC-H/TPC-R Customer Distribution Query (Q13)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 4 Query 13 START -- ', qstart ;
select
    c_count,
    count(*) as custdist
from
    (
        select
            c_custkey,
            count(o_orderkey)
        from
            customer left outer join orders
on

```

```

        c_custkey = o_custkey
        and o_comment not like
        '%pending%accounts%'
    group by
        c_custkey
    ) as c_orders (c_custkey, c_count)
group by
    c_count
order by
    custdist desc,
    c_count desc;
commit ;
set qstop = now(*);
select 'Stream 4 Query 13 STOP -- ', qstop ;
select 'Stream 4 Query 13 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)7.sql 2.1.8.1
-- TPC-H/TPC-R Volume Shipping Query (Q7)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 4 Query 7 START -- ', qstart ;
select
    supp_nation,
    cust_nation,
    l_year,
    sum(volume) as revenue
from
    (
        select
            n1.n_name as supp_nation,
            n2.n_name as cust_nation,
            datepart(year, l_shipdate) as
l_year,
            l_extendedprice * (1 -
l_discount) as volume
        from
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2
        where
            s_suppkey = l_suppkey
            and o_orderkey = l_orderkey
            and c_custkey = o_custkey
            and s_nationkey =
n1.n_nationkey
            and c_nationkey =
n2.n_nationkey
            and (
                (n1.n_name = 'INDONESIA'
and n2.n_name = 'MOZAMBIQUE')
                or (n1.n_name =
'MOZAMBIQUE' and n2.n_name = 'INDONESIA')
            )
            and l_shipdate between
'1995-01-01' and '1996-12-31'
    ) as shipping
group by
    supp_nation,
    cust_nation,
    l_year
order by
    supp_nation,
    cust_nation,
    l_year;
commit ;
set qstop = now(*);
select 'Stream 4 Query 7 STOP -- ', qstop ;
select 'Stream 4 Query 7 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)22.sql 2.1.8.1

```

```
-- TPC-H/TPC-R Global Sales Opportunity Query (Q22)
-- Functional Query Definition
-- Approved February 1998
```

```
set qstart = now(*) ;
select 'Stream 4 Query 22 START -- ', qstart ;
select
  c_ntrycode,
  count(*) as numcust,
  sum(c_acctbal) as totacctbal
from
  (
    select
      substring(c_phone,1,2) as
c_ntrycode,
      c_acctbal
    from
      customer
    where
      substring(c_phone,1,2) in
('27', '22', '13', '18',
'16', '12', '10')
      and c_acctbal > (
        select
          avg(c_acctbal)
        from
          customer
        where
          c_acctbal > 0.00
          and
            substring(c_phone,1,2) in
('22', '13', '18', '16', '12', '10')
            )
      and not exists (
        select
          *
        from
          orders
        where
          o_custkey =
c_custkey
      )
    ) as custsale
group by
  c_ntrycode
order by
  c_ntrycode ;
commit ;
set qstop = now(*) ;
select 'Stream 4 Query 22 STOP -- ', qstop ;
select 'Stream 4 Query 22 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)3.sql 2.1.8.1
-- TPC-H/TPC-R Shipping Priority Query (Q3)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 4 Query 3 START -- ', qstart ;
select top 10
  l_orderkey,
  sum(l_extendedprice * (1 - l_discount)) as
revenue,
  o_orderdate,
  o_shippriority
from
  customer,
  orders,
  lineitem
where
  c_mktsegment = 'BUILDING'
  and c_custkey = o_custkey
  and l_orderkey = o_orderkey
  and o_orderdate < '1995-03-22'
```

```
and l_shipdate > '1995-03-22'
group by
  l_orderkey,
  o_orderdate,
  o_shippriority
order by
  revenue desc,
  o_orderdate ;
commit ;
set qstop = now(*) ;
select 'Stream 4 Query 3 STOP -- ', qstop ;
select 'Stream 4 Query 3 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)20.sql 2.1.8.1
-- TPC-H/TPC-R Potential Part Promotion Query (Q20)
-- Function Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 4 Query 20 START -- ', qstart ;
select
  s_name,
  s_address
from
  supplier,
  nation
where
  s_suppkey in (
    select
      ps_suppkey
    from
      partsupp
    where
      ps_partkey in (
        select
          p_partkey
        from
          part
        where
          p_name like
'snow%'
      )
      and ps_availqty > (
        select
          0.5 *
sum(l_quantity)
        from
          lineitem
        where
          l_partkey =
ps_partkey
          and l_suppkey =
ps_suppkey
          and l_shipdate
>= '1995-01-01'
          and l_shipdate
< dateadd(year,1,'1995-01-01')
      )
    )
  and s_nationkey = n_nationkey
  and n_name = 'PERU'
order by
  s_name ;
commit ;
set qstop = now(*) ;
select 'Stream 4 Query 20 STOP -- ', qstop ;
select 'Stream 4 Query 20 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;

select 'Stream completion time -- ',now(*) as "time";
disconnect ;
exit ;
```

stream5.sql

```
=====
-- using 115192431 as a seed to the RNG
create variable qstart timestamp;
create variable qstop timestamp;
select 'Stream begin time -- ',now(*) as "time";
-- @(#)21.sql      2.1.8.1
-- TPC-H/TPC-R Suppliers Who Kept Orders Waiting Query (Q21)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 5 Query 21 START -- ', qstart ;
select top 100
    s_name,
    count(*) as numwait
from
    supplier,
    lineitem l1,
    orders,
    nation
where
    s_suppkey = l1.l_suppkey
    and o_orderkey = l1.l_orderkey
    and o_orderstatus = 'F'
    and l1.l_receiptdate > l1.l_commitdate
    and exists (
        select
            *
        from
            lineitem l2
        where
            l2.l_orderkey = l1.l_orderkey
            and l2.l_suppkey <> l1.l_suppkey
    )
    and not exists (
        select
            *
        from
            lineitem l3
        where
            l3.l_orderkey = l1.l_orderkey
            and l3.l_suppkey <> l1.l_suppkey
            and l3.l_receiptdate > l3.l_commitdate
    )
    and s_nationkey = n_nationkey
    and n_name = 'GERMANY'
group by
    s_name
order by
    numwait desc,
    s_name;
commit ;
set qstop = now(*);
select 'Stream 5 Query 21 STOP -- ', qstop ;
select 'Stream 5 Query 21 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)15.sql      2.1.8.1
-- TPC-H/TPC-R Top Supplier Query (Q15)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 5 Query 15 START -- ', qstart ;
create view revenue5 (supplier_no, total_revenue) as
    select
        l_suppkey,
        sum(l_extendedprice * (1 - l_discount))
    from
```

```
        lineitem
    where
        l_shipdate >= '1997-01-01'
    and l_shipdate < dateadd(month,3,'1997-01-01')
    group by
        l_suppkey;

select
    s_suppkey,
    s_name,
    s_address,
    s_phone,
    total_revenue
from
    supplier,
    revenue5
where
    s_suppkey = supplier_no
    and total_revenue = (
        select
            max(total_revenue)
        from
            revenue5
    )
order by
    s_suppkey;

drop view revenue5;
commit ;
set qstop = now(*);
select 'Stream 5 Query 15 STOP -- ', qstop ;
select 'Stream 5 Query 15 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)4.sql      2.1.8.1
-- TPC-H/TPC-R Order Priority Checking Query (Q4)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 5 Query 4 START -- ', qstart ;
select
    o_orderpriority,
    count(*) as order_count
from
    orders
where
    o_orderdate >= '1994-03-01'
    and o_orderdate < dateadd(month,3,'1994-03-01')
    and exists (
        select
            *
        from
            lineitem
        where
            l_orderkey = o_orderkey
            and l_commitdate < l_receiptdate
    )
group by
    o_orderpriority
order by
    o_orderpriority;
commit ;
set qstop = now(*);
select 'Stream 5 Query 4 STOP -- ', qstop ;
select 'Stream 5 Query 4 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)6.sql      2.1.8.1
-- TPC-H/TPC-R Forecasting Revenue Change Query (Q6)
-- Functional Query Definition
-- Approved February 1998
```

```

set qstart = now(*);
select 'Stream 5 Query 6 START -- ', qstart ;
select
    sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
    l_shipdate >= '1996-01-01'
    and l_shipdate < dateadd(year,1,'1996-01-01')
    and l_discount between 0.02 - 0.01 and 0.02 + 0.01
    and l_quantity < 24;
commit ;
set qstop = now(*);
select 'Stream 5 Query 6 STOP -- ', qstop ;
select 'Stream 5 Query 6 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds';
-- @(#)7.sql      2.1.8.1
-- TPC-H/TPC-R Volume Shipping Query (Q7)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 5 Query 7 START -- ', qstart ;
select
    supp_nation,
    cust_nation,
    l_year,
    sum(volume) as revenue
from
    (
        select
            n1.n_name as supp_nation,
            n2.n_name as cust_nation,
            datepart(year, l_shipdate) as l_year,
            l_extendedprice * (1 - l_discount) as volume
        from
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2
        where
            s_suppkey = l_suppkey
            and o_orderkey = l_orderkey
            and c_custkey = o_custkey
            and s_nationkey = n1.n_nationkey
            and c_nationkey = n2.n_nationkey
            and (
                (n1.n_name = 'ARGENTINA' and
                    n2.n_name = 'INDIA')
                or (n1.n_name = 'INDIA' and
                    n2.n_name = 'ARGENTINA')
            )
            and l_shipdate between '1995-01-01' and
                '1996-12-31'
    ) as shipping
group by
    supp_nation,
    cust_nation,
    l_year
order by
    supp_nation,
    cust_nation,
    l_year;
commit ;
set qstop = now(*);
select 'Stream 5 Query 7 STOP -- ', qstop ;
select 'Stream 5 Query 7 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds';
-- @(#)16.sql    2.1.8.1

```

```

-- TPC-H/TPC-R Parts/Supplier Relationship Query (Q16)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 5 Query 16 START -- ', qstart ;
select
    p_brand,
    p_type,
    p_size,
    count(distinct ps_suppkey) as supplier_cnt
from
    partsupp,
    part
where
    p_partkey = ps_partkey
    and p_brand <> 'Brand#14'
    and p_type not like 'PROMO BURNISHED%'
    and p_size in (42, 30, 34, 45, 13, 8, 32, 17)
    and ps_suppkey not in (
        select
            s_suppkey
        from
            supplier
        where
            s_comment like '%Customer%Complaints%'
    )
group by
    p_brand,
    p_type,
    p_size
order by
    supplier_cnt desc,
    p_brand,
    p_type,
    p_size;
commit ;
set qstop = now(*);
select 'Stream 5 Query 16 STOP -- ', qstop ;
select 'Stream 5 Query 16 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds';
-- @(#)19.sql    2.1.8.1
-- TPC-H/TPC-R Discounted Revenue Query (Q19)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 5 Query 19 START -- ', qstart ;
select
    sum(l_extendedprice* (1 - l_discount)) as revenue
from
    lineitem,
    part
where
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#52'
        and p_container in ('SM CASE', 'SM BOX', 'SM PACK',
            'SM PKG')
        and l_quantity >= 9 and l_quantity <= 9 + 10
        and p_size between 1 and 5
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#22'
        and p_container in ('MED BAG', 'MED BOX', 'MED
            PKG', 'MED PACK')
        and l_quantity >= 13 and l_quantity <= 13 + 10
    )

```

```

and p_size between 1 and 10
and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN PERSON'
)
or
(
p_partkey = l_partkey
and p_brand = 'Brand#41'
and p_container in ('LG CASE', 'LG BOX', 'LG PACK',
'LG PKG')
and l_quantity >= 30 and l_quantity <= 30 + 10
and p_size between 1 and 15
and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN PERSON'
);
commit ;
set qstop = now(*);
select 'Stream 5 Query 19 STOP -- ', qstop ;
select 'Stream 5 Query 19 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, 'seconds' ;
-- @(#)18.sql 2.1.8.1
-- TPC-H/TPC-R Large Volume Customer Query (Q18)
-- Function Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 5 Query 18 START -- ', qstart ;
select top 100
c_name,
c_custkey,
o_orderkey,
o_orderdate,
o_totalprice,
sum(l_quantity)
from
customer,
orders,
lineitem
where
o_orderkey in (
select
l_orderkey
from
lineitem
group by
l_orderkey having
sum(l_quantity) > 315
)
and c_custkey = o_custkey
and o_orderkey = l_orderkey
group by
c_name,
c_custkey,
o_orderkey,
o_orderdate,
o_totalprice
order by
o_totalprice desc,
o_orderdate;
commit ;
set qstop = now(*);
select 'Stream 5 Query 18 STOP -- ', qstop ;
select 'Stream 5 Query 18 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, 'seconds' ;
-- @(#)14.sql 2.1.8.1
-- TPC-H/TPC-R Promotion Effect Query (Q14)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 5 Query 14 START -- ', qstart ;

```

```

select
100.00 * sum(case
when p_type like 'PROMO%'
then l_extendedprice * (1 - l_discount)
else 0
end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
lineitem,
part
where
l_partkey = p_partkey
and l_shipdate >= '1996-01-01'
and l_shipdate < dateadd(month,1,'1996-01-01');
commit ;
set qstop = now(*);
select 'Stream 5 Query 14 STOP -- ', qstop ;
select 'Stream 5 Query 14 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, 'seconds' ;
-- @(#)22.sql 2.1.8.1
-- TPC-H/TPC-R Global Sales Opportunity Query (Q22)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 5 Query 22 START -- ', qstart ;
select
cntrycode,
count(*) as numcust,
sum(c_acctbal) as totacctbal
from
(
select
substring(c_phone,1,2) as cntrycode,
c_acctbal
from
customer
where
substring(c_phone,1,2) in ('12', '14', '19', '31', '10', '16', '33')
and c_acctbal > (
select
avg(c_acctbal)
from
customer
where
c_acctbal > 0.00
and substring(c_phone,1,2) in ('12', '14',
'19', '31', '10', '16', '33')
)
and not exists (
select
*
from
orders
where
o_custkey = c_custkey
)
) as custsale
group by
cntrycode
order by
cntrycode;
commit ;
set qstop = now(*);
select 'Stream 5 Query 22 STOP -- ', qstop ;
select 'Stream 5 Query 22 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, 'seconds' ;
-- @(#)11.sql 2.1.8.1
-- TPC-H/TPC-R Important Stock Identification Query (Q11)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 5 Query 11 START -- ', qstart ;
select
    ps_partkey,
    sum(ps_supplycost * ps_availqty) as value
from
    partsupp,
    supplier,
    nation
where
    ps_suppkey = s_suppkey
    and s_nationkey = n_nationkey
    and n_name = 'KENYA'
group by
    ps_partkey having
        sum(ps_supplycost * ps_availqty) > (
            select
                sum(ps_supplycost * ps_availqty) *
0.0000003333
            from
                partsupp,
                supplier,
                nation
            where
                ps_suppkey = s_suppkey
                and s_nationkey = n_nationkey
                and n_name = 'KENYA'
        )
order by
    value desc;
commit ;
set qstop = now(*);
select 'Stream 5 Query 11 STOP -- ', qstop ;
select 'Stream 5 Query 11 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)13.sql 2.1.8.1
-- TPC-H/TPC-R Customer Distribution Query (Q13)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 5 Query 13 START -- ', qstart ;
select
    c_count,
    count(*) as custdist
from
    (
        select
            c_custkey,
            count(o_orderkey)
        from
            customer left outer join orders on
                c_custkey = o_custkey
                and o_comment not like '%pending
%accounts%'
        group by
            c_custkey
    ) as c_orders (c_custkey, c_count)
group by
    c_count
order by
    custdist desc,
    c_count desc;
commit ;
set qstop = now(*);
select 'Stream 5 Query 13 STOP -- ', qstop ;
select 'Stream 5 Query 13 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)3.sql 2.1.8.1
-- TPC-H/TPC-R Shipping Priority Query (Q3)

```

```

-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 5 Query 3 START -- ', qstart ;
select top 10
    l_orderkey,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    o_orderdate,
    o_shippriority
from
    customer,
    orders,
    lineitem
where
    c_mktsegment = 'HOUSEHOLD'
    and c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate < '1995-03-08'
    and l_shipdate > '1995-03-08'
group by
    l_orderkey,
    o_orderdate,
    o_shippriority
order by
    revenue desc,
    o_orderdate;
commit ;
set qstop = now(*);
select 'Stream 5 Query 3 STOP -- ', qstop ;
select 'Stream 5 Query 3 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)1.sql 2.1.8.1
-- TPC-H/TPC-R Pricing Summary Report Query (Q1)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 5 Query 1 START -- ', qstart ;
select
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineitem
where
    l_shipdate <= dateadd(day,-95,'1998-12-01')
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus;
commit ;
set qstop = now(*);
select 'Stream 5 Query 1 STOP -- ', qstop ;
select 'Stream 5 Query 1 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)2.sql 2.1.8.2
-- TPC-H/TPC-R Minimum Cost Supplier Query (Q2)
-- Functional Query Definition
-- Approved February 1998

```



```

set qstart = now(*);
select 'Stream 5 Query 2 START -- ', qstart ;
select top 100
    s_acctbal,
    s_name,
    n_name,
    p_partkey,
    p_mfgr,
    s_address,
    s_phone,
    s_comment
from
    part,
    supplier,
    partsupp,
    nation,
    region
where
    p_partkey = ps_partkey
    and s_suppkey = ps_suppkey
    and p_size = 48
    and p_type like '%NICKEL'
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'AMERICA'
    and ps_supplycost = (
        select
            min(ps_supplycost)
        from
            partsupp,
            supplier,
            nation,
            region
        where
            p_partkey = ps_partkey
            and s_suppkey = ps_suppkey
            and s_nationkey = n_nationkey
            and n_regionkey = r_regionkey
            and r_name = 'AMERICA'
    )
order by
    s_acctbal desc,
    n_name,
    s_name,
    p_partkey;
commit ;
set qstop = now(*);
select 'Stream 5 Query 2 STOP -- ', qstop ;
select 'Stream 5 Query 2 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)5.sql      2.1.8.1
-- TPC-H/TPC-R Local Supplier Volume Query (Q5)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 5 Query 5 START -- ', qstart ;
select
    n_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue
from
    customer,
    orders,
    lineitem,
    supplier,
    nation,
    region
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and l_suppkey = s_suppkey
    and c_nationkey = s_nationkey
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'MIDDLE EAST'
    and o_orderdate >= '1996-01-01'
    and o_orderdate < dateadd(year,1,'1996-01-01')
group by
    n_name
order by
    revenue desc;
commit ;
set qstop = now(*);
select 'Stream 5 Query 5 STOP -- ', qstop ;
select 'Stream 5 Query 5 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)8.sql      2.1.8.1
-- TPC-H/TPC-R National Market Share Query (Q8)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 5 Query 8 START -- ', qstart ;
select
    o_year,
    sum(case
        when nation = 'INDIA' then volume
        else 0
    end) / sum(volume) as mkt_share
from
    (
        select
            datepart(year, o_orderdate) as o_year,
            l_extendedprice * (1 - l_discount) as volume,
            n2.n_name as nation
        from
            part,
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2,
            region
        where
            p_partkey = l_partkey
            and s_suppkey = l_suppkey
            and l_orderkey = o_orderkey
            and o_custkey = c_custkey
            and c_nationkey = n1.n_nationkey
            and n1.n_regionkey = r_regionkey
            and r_name = 'ASIA'
            and s_nationkey = n2.n_nationkey
            and o_orderdate between '1995-01-01' and
'1996-12-31'
            and p_type = 'PROMO PLATED STEEL'
    ) as all_nations
group by
    o_year
order by
    o_year;
commit ;
set qstop = now(*);
select 'Stream 5 Query 8 STOP -- ', qstop ;
select 'Stream 5 Query 8 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)20.sql     2.1.8.1
-- TPC-H/TPC-R Potential Part Promotion Query (Q20)
-- Function Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 5 Query 20 START -- ', qstart ;

```

```

select
    s_name,
    s_address
from
    supplier,
    nation
where
    s_suppkey in (
        select
            ps_suppkey
        from
            partsupp
        where
            ps_partkey in (
                select
                    p_partkey
                from
                    part
                where
                    p_name like 'forest%'
            )
            and ps_availqty > (
                select
                    0.5 * sum(l_quantity)
                from
                    lineitem
                where
                    l_partkey = ps_partkey
                    and l_suppkey =
                        ps_suppkey
                    and l_shipdate >=
                        '1994-01-01'
                    and l_shipdate < dateadd(year,1,'1994-01-01')
            )
            and s_nationkey = n_nationkey
            and n_name = 'GERMANY'
    )
order by
    s_name;
commit ;
set qstop = now(*);
select 'Stream 5 Query 20 STOP -- ', qstop ;
select 'Stream 5 Query 20 LENGTH -- ', cast((datediff(millisecond,qstart,qstop)
as numeric)/1000, 'seconds' ;
-- @(#)12.sql 2.1.8.1
-- TPC-H/TPC-R Shipping Modes and Order Priority Query (Q12)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 5 Query 12 START -- ', qstart ;
select
    l_shipmode,
    sum(case
        when o_orderpriority = '1-URGENT'
            or o_orderpriority = '2-HIGH'
        then 1
        else 0
    end) as high_line_count,
    sum(case
        when o_orderpriority <> '1-URGENT'
            and o_orderpriority <> '2-HIGH'
        then 1
        else 0
    end) as low_line_count
from
    orders,
    lineitem
where
    o_orderkey = l_orderkey
    and l_shipmode in ('RAIL', 'TRUCK')
    and l_commitdate < l_receiptdate
        and l_shipdate < l_commitdate
        and l_receiptdate >= '1996-01-01'
        and l_receiptdate < dateadd(year,1,'1996-01-01')
group by
    l_shipmode
order by
    l_shipmode;
commit ;
set qstop = now(*);
select 'Stream 5 Query 12 STOP -- ', qstop ;
select 'Stream 5 Query 12 LENGTH -- ', cast((datediff(millisecond,qstart,qstop)
as numeric)/1000, 'seconds' ;
-- @(#)17.sql 2.1.8.1
-- TPC-H/TPC-R Small-Quantity-Order Revenue Query (Q17)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 5 Query 17 START -- ', qstart ;
select
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#45'
    and p_container = 'WRAP BOX'
    and l_quantity < (
        select
            0.2 * avg(l_quantity)
        from
            lineitem
        where
            l_partkey = p_partkey
    );
commit ;
set qstop = now(*);
select 'Stream 5 Query 17 STOP -- ', qstop ;
select 'Stream 5 Query 17 LENGTH -- ', cast((datediff(millisecond,qstart,qstop)
as numeric)/1000, 'seconds' ;
-- @(#)10.sql 2.1.8.1
-- TPC-H/TPC-R Returned Item Reporting Query (Q10)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 5 Query 10 START -- ', qstart ;
select top 20
    c_custkey,
    c_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    c_acctbal,
    n_name,
    c_address,
    c_phone,
    c_comment
from
    customer,
    orders,
    lineitem,
    nation
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate >= '1994-04-01'
    and o_orderdate < dateadd(month,3,'1994-04-01')
    and l_returnflag = 'R'
    and c_nationkey = n_nationkey
group by
    c_custkey,

```

```

c_name,
c_acctbal,
c_phone,
n_name,
c_address,
c_comment
order by
    revenue desc;
commit ;
set qstop = now(*);
select 'Stream 5 Query 10 STOP -- ', qstop ;
select 'Stream 5 Query 10 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)9.sql      2.1.8.1
-- TPC-H/TPC-R Product Type Profit Measure Query (Q9)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 5 Query 9 START -- ', qstart ;
select
    nation,
    o_year,
    sum(amount) as sum_profit
from
    (
        select
            n_name as nation,
            datepart(year, o_orderdate) as o_year,
            l_extendedprice * (1 - l_discount) -
ps_supplycost * l_quantity as amount
            from
                part,
                supplier,
                lineitem,
                partsupp,
                orders,
                nation
            where
                s_suppkey = l_suppkey
                and ps_suppkey = l_suppkey
                and ps_partkey = l_partkey
                and p_partkey = l_partkey
                and o_orderkey = l_orderkey
                and s_nationkey = n_nationkey
                and p_name like '%tan%'
        ) as profit
group by
    nation,
    o_year
order by
    nation,
    o_year desc;

```

```

commit ;
set qstop = now(*);
select 'Stream 5 Query 9 STOP -- ', qstop ;
select 'Stream 5 Query 9 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;

```

```
select 'Stream completion time -- ',now(*) as "time";
```

```
disconnect;
```

```
exit;
```

```

=====
stream6.sql
=====

```

```
-- using 115192432 as a seed to the RNG
```

```

create variable qstart timestamp;
create variable qstop timestamp;
select 'Stream begin time -- ',now(*) as "time";
-- @(#)10.sql      2.1.8.1
-- TPC-H/TPC-R Returned Item Reporting Query (Q10)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 6 Query 10 START -- ', qstart ;
select top 20
    c_custkey,
    c_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    c_acctbal,
    n_name,
    c_address,
    c_phone,
    c_comment
from
    customer,
    orders,
    lineitem,
    nation
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate >= '1995-01-01'
    and o_orderdate < dateadd(month,3,'1995-01-01')
    and l_returnflag = 'R'
    and c_nationkey = n_nationkey
group by
    c_custkey,
    c_name,
    c_acctbal,
    c_phone,
    n_name,
    c_address,
    c_comment
order by
    revenue desc;
commit ;
set qstop = now(*);
select 'Stream 6 Query 10 STOP -- ', qstop ;
select 'Stream 6 Query 10 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)3.sql      2.1.8.1
-- TPC-H/TPC-R Shipping Priority Query (Q3)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 6 Query 3 START -- ', qstart ;
select top 10
    l_orderkey,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    o_orderdate,
    o_shippriority
from
    customer,
    orders,
    lineitem
where
    c_mktsegment = 'AUTOMOBILE'
    and c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate < '1995-03-24'
    and l_shipdate > '1995-03-24'
group by
    l_orderkey,
    o_orderdate,

```

```

order by      o_shippriority
              revenue desc,
              o_orderdate;
commit ;
set qstop = now(*);
select 'Stream 6 Query 3 STOP -- ', qstop ;
select 'Stream 6 Query 3 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)15.sql      2.1.8.1
-- TPC-H/TPC-R Top Supplier Query (Q15)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 6 Query 15 START -- ', qstart ;
create view revenue6 (supplier_no, total_revenue) as
select
    l_suppkey,
    sum(l_extendedprice * (1 - l_discount))
from
    lineitem
where
    l_shipdate >= '1995-01-01'
and l_shipdate < dateadd(month,3,'1995-01-01')
group by
    l_suppkey;

```

```

select
    s_suppkey,
    s_name,
    s_address,
    s_phone,
    total_revenue
from
    supplier,
    revenue6
where
    s_suppkey = supplier_no
and total_revenue = (
    select
        max(total_revenue)
    from
        revenue6
)
order by
    s_suppkey;

```

```

drop view revenue6;
commit ;
set qstop = now(*);
select 'Stream 6 Query 15 STOP -- ', qstop ;
select 'Stream 6 Query 15 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)13.sql      2.1.8.1
-- TPC-H/TPC-R Customer Distribution Query (Q13)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 6 Query 13 START -- ', qstart ;
select
    c_count,
    count(*) as custdist
from
    (
        select
            c_custkey,
            count(o_orderkey)
        from
            customer left outer join orders on

```

```

c_custkey = o_custkey
and o_comment not like '%pending

```

```

%deposits%'
              group by
                  c_custkey
            ) as c_orders (c_custkey, c_count)
group by
    c_count
order by
    custdist desc,
    c_count desc;
commit ;
set qstop = now(*);
select 'Stream 6 Query 13 STOP -- ', qstop ;
select 'Stream 6 Query 13 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)6.sql      2.1.8.1
-- TPC-H/TPC-R Forecasting Revenue Change Query (Q6)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 6 Query 6 START -- ', qstart ;
select
    sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
    l_shipdate >= '1996-01-01'
and l_shipdate < dateadd(year,1,'1996-01-01')
and l_discount between 0.07 - 0.01 and 0.07 + 0.01
and l_quantity < 24;

```

```

commit ;
set qstop = now(*);
select 'Stream 6 Query 6 STOP -- ', qstop ;
select 'Stream 6 Query 6 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)8.sql      2.1.8.1
-- TPC-H/TPC-R National Market Share Query (Q8)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 6 Query 8 START -- ', qstart ;
select
    o_year,
    sum(case
        when nation = 'ALGERIA' then volume
        else 0
    end) / sum(volume) as mkt_share
from
    (
        select
            datepart(year, o_orderdate) as o_year,
            l_extendedprice * (1 - l_discount) as volume,
            n2.n_name as nation
        from
            part,
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2,
            region
        where
            p_partkey = l_partkey
            and s_suppkey = l_suppkey
            and l_orderkey = o_orderkey
            and o_custkey = c_custkey
            and c_nationkey = n1.n_nationkey

```

```

and n1.n_regionkey = r_regionkey
and r_name = 'AFRICA'
and s_nationkey = n2.n_nationkey
and o_orderdate between '1995-01-01' and
'1996-12-31'
and p_type = 'PROMO ANODIZED STEEL'
) as all_nations
group by
o_year
order by
o_year;
commit ;
set qstop = now(*) ;
select 'Stream 6 Query 8 STOP --', qstop ;
select 'Stream 6 Query 8 LENGTH --', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, 'seconds' ;
-- @(#)9.sql 2.1.8.1
-- TPC-H/TPC-R Product Type Profit Measure Query (Q9)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 6 Query 9 START --', qstart ;
select
nation,
o_year,
sum(amount) as sum_profit
from
(
select
n_name as nation,
datepart(year, o_orderdate) as o_year,
l_extendedprice * (1 - l_discount) -
ps_supplycost * l_quantity as amount
from
part,
supplier,
lineitem,
partsupp,
orders,
nation
where
s_suppkey = l_suppkey
and ps_suppkey = l_suppkey
and ps_partkey = l_partkey
and p_partkey = l_partkey
and o_orderkey = l_orderkey
and s_nationkey = n_nationkey
and p_name like '%sky%'
) as profit
group by
nation,
o_year
order by
nation,
o_year desc;
commit ;
set qstop = now(*) ;
select 'Stream 6 Query 9 STOP --', qstop ;
select 'Stream 6 Query 9 LENGTH --', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, 'seconds' ;
-- @(#)7.sql 2.1.8.1
-- TPC-H/TPC-R Volume Shipping Query (Q7)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 6 Query 7 START --', qstart ;
select
supp_nation,
cust_nation,

```

```

l_year,
sum(volume) as revenue
from
(
select
n1.n_name as supp_nation,
n2.n_name as cust_nation,
datepart(year, l_shipdate) as l_year,
l_extendedprice * (1 - l_discount) as volume
from
supplier,
lineitem,
orders,
customer,
nation n1,
nation n2
where
s_suppkey = l_suppkey
and o_orderkey = l_orderkey
and c_custkey = o_custkey
and s_nationkey = n1.n_nationkey
and c_nationkey = n2.n_nationkey
and (
(n1.n_name = 'CHINA' and
n2.n_name = 'ALGERIA')
or (n1.n_name = 'ALGERIA' and
n2.n_name = 'CHINA')
)
and l_shipdate between '1995-01-01' and
'1996-12-31'
) as shipping
group by
supp_nation,
cust_nation,
l_year
order by
supp_nation,
cust_nation,
l_year;
commit ;
set qstop = now(*) ;
select 'Stream 6 Query 7 STOP --', qstop ;
select 'Stream 6 Query 7 LENGTH --', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, 'seconds' ;
-- @(#)4.sql 2.1.8.1
-- TPC-H/TPC-R Order Priority Checking Query (Q4)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*) ;
select 'Stream 6 Query 4 START --', qstart ;
select
o_orderpriority,
count(*) as order_count
from
orders
where
o_orderdate >= '1996-10-01'
and o_orderdate < dateadd(month,3,'1996-10-01')
and exists (
select
*
from
lineitem
where
l_orderkey = o_orderkey
and l_commitdate < l_receiptdate
)
group by
o_orderpriority
order by
o_orderpriority;

```

```

commit ;
set qstop = now(*);
select 'Stream 6 Query 4 STOP -- ', qstop ;
select 'Stream 6 Query 4 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)11.sql      2.1.8.1
-- TPC-H/TPC-R Important Stock Identification Query (Q11)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 6 Query 11 START -- ', qstart ;
select

```

```

    ps_partkey,
    sum(ps_supplycost * ps_availqty) as value
from
    partsupp,
    supplier,
    nation
where
    ps_suppkey = s_suppkey
    and s_nationkey = n_nationkey
    and n_name = 'BRAZIL'
group by
    ps_partkey having
        sum(ps_supplycost * ps_availqty) > (
            select
                sum(ps_supplycost * ps_availqty) *
0.0000003333
            from
                partsupp,
                supplier,
                nation
            where
                ps_suppkey = s_suppkey
                and s_nationkey = n_nationkey
                and n_name = 'BRAZIL'
        )
order by
    value desc;

```

```

commit ;
set qstop = now(*);
select 'Stream 6 Query 11 STOP -- ', qstop ;
select 'Stream 6 Query 11 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)22.sql      2.1.8.1
-- TPC-H/TPC-R Global Sales Opportunity Query (Q22)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 6 Query 22 START -- ', qstart ;
select

```

```

    centrycode,
    count(*) as numcust,
    sum(c_acctbal) as totacctbal
from
    (
        select
            substring(c_phone,1,2) as centrycode,
            c_acctbal
        from
            customer
        where
            substring(c_phone,1,2) in ('23', '30', '10', '19', '20', '15', '25')
            and c_acctbal > (
                select
                    avg(c_acctbal)
                from
                    customer

```

```

        where
            c_acctbal > 0.00
        and substring(c_phone,1,2) in ('10', '19', '20', '15', '25')
    )
    and not exists (
        select
            *
        from
            orders
        where
            o_custkey = c_custkey
    )
) as custsale

```

```

group by
    centrycode
order by
    centrycode;
commit ;
set qstop = now(*);
select 'Stream 6 Query 22 STOP -- ', qstop ;
select 'Stream 6 Query 22 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)18.sql      2.1.8.1
-- TPC-H/TPC-R Large Volume Customer Query (Q18)
-- Function Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 6 Query 18 START -- ', qstart ;
select top 100

```

```

    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice,
    sum(l_quantity)
from
    customer,
    orders,
    lineitem
where
    o_orderkey in (
        select
            l_orderkey
        from
            lineitem
        group by
            l_orderkey having
                sum(l_quantity) > 313
    )

```

```

    and c_custkey = o_custkey
    and o_orderkey = l_orderkey

```

```

group by
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice
order by
    o_totalprice desc,
    o_orderdate;

```

```

commit ;
set qstop = now(*);
select 'Stream 6 Query 18 STOP -- ', qstop ;
select 'Stream 6 Query 18 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)12.sql      2.1.8.1
-- TPC-H/TPC-R Shipping Modes and Order Priority Query (Q12)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 6 Query 12 START -- ', qstart ;
select
    l_shipmode,
    sum(case
        when o_orderpriority = '1-URGENT'
            or o_orderpriority = '2-HIGH'
            then 1
        else 0
    end) as high_line_count,
    sum(case
        when o_orderpriority <> '1-URGENT'
            and o_orderpriority <> '2-HIGH'
            then 1
        else 0
    end) as low_line_count
from
    orders,
    lineitem
where
    o_orderkey = l_orderkey
    and l_shipmode in ('REG AIR', 'TRUCK')
    and l_commitdate < l_receiptdate
    and l_shipdate < l_commitdate
    and l_receiptdate >= '1996-01-01'
    and l_receiptdate < dateadd(year,1,'1996-01-01')
group by
    l_shipmode
order by
    l_shipmode;
commit ;
set qstop = now(*);
select 'Stream 6 Query 12 STOP -- ', qstop ;
select 'Stream 6 Query 12 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)1.sql      2.1.8.1
-- TPC-H/TPC-R Pricing Summary Report Query (Q1)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 6 Query 1 START -- ', qstart ;
select
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineitem
where
    l_shipdate <= dateadd(day,-103,'1998-12-01')
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus;
commit ;
set qstop = now(*);
select 'Stream 6 Query 1 STOP -- ', qstop ;
select 'Stream 6 Query 1 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)5.sql      2.1.8.1
-- TPC-H/TPC-R Local Supplier Volume Query (Q5)

```

```

-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 6 Query 5 START -- ', qstart ;
select
    n_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue
from
    customer,
    orders,
    lineitem,
    supplier,
    nation,
    region
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and l_suppkey = s_suppkey
    and c_nationkey = s_nationkey
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'AFRICA'
    and o_orderdate >= '1996-01-01'
    and o_orderdate < dateadd(year,1,'1996-01-01')
group by
    n_name
order by
    revenue desc;
commit ;
set qstop = now(*);
select 'Stream 6 Query 5 STOP -- ', qstop ;
select 'Stream 6 Query 5 LENGTH -- ', cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds' ;
-- @(#)16.sql      2.1.8.1
-- TPC-H/TPC-R Parts/Supplier Relationship Query (Q16)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 6 Query 16 START -- ', qstart ;
select
    p_brand,
    p_type,
    p_size,
    count(distinct ps_suppkey) as supplier_cnt
from
    partsupp,
    part
where
    p_partkey = ps_partkey
    and p_brand <> 'Brand#44'
    and p_type not like 'SMALL POLISHED%'
    and p_size in (46, 23, 48, 12, 33, 26, 20, 22)
    and ps_suppkey not in (
        select
            s_suppkey
        from
            supplier
        where
            s_comment like '%Customer%Complaints%'
    )
group by
    p_brand,
    p_type,
    p_size
order by
    supplier_cnt desc,
    p_brand,
    p_type,
    p_size;

```

```

commit ;
set qstop = now(*) ;
select 'Stream 6 Query 16 STOP -- ', qstop ;
select 'Stream 6 Query 16 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)2.sql      2.1.8.2
-- TPC-H/TPC-R Minimum Cost Supplier Query (Q2)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 6 Query 2 START -- ', qstart ;
select top 100
    s_acctbal,
    s_name,
    n_name,
    p_partkey,
    p_mfgr,
    s_address,
    s_phone,
    s_comment

```

```

from
    part,
    supplier,
    partsupp,
    nation,
    region

```

```

where
    p_partkey = ps_partkey
    and s_suppkey = ps_suppkey
    and p_size = 35
    and p_type like '%TIN'
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'EUROPE'
    and ps_supplycost = (
        select
            min(ps_supplycost)
        from
            partsupp,
            supplier,
            nation,
            region
        where
            p_partkey = ps_partkey
            and s_suppkey = ps_suppkey
            and s_nationkey = n_nationkey
            and n_regionkey = r_regionkey
            and r_name = 'EUROPE'
    )

```

```

order by
    s_acctbal desc,
    n_name,
    s_name,
    p_partkey ;

```

```

commit ;
set qstop = now(*) ;
select 'Stream 6 Query 2 STOP -- ', qstop ;
select 'Stream 6 Query 2 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)14.sql     2.1.8.1
-- TPC-H/TPC-R Promotion Effect Query (Q14)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 6 Query 14 START -- ', qstart ;
select
    100.00 * sum(case
        when p_type like 'PROMO%'
        then l_extendedprice * (1 - l_discount)

```

```

        else 0
    end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
    lineitem,
    part
where
    l_partkey = p_partkey
    and l_shipdate >= '1996-01-01'
    and l_shipdate < dateadd(month,1,'1996-01-01') ;

```

```

commit ;
set qstop = now(*) ;
select 'Stream 6 Query 14 STOP -- ', qstop ;
select 'Stream 6 Query 14 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)19.sql     2.1.8.1
-- TPC-H/TPC-R Discounted Revenue Query (Q19)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 6 Query 19 START -- ', qstart ;
select
    sum(l_extendedprice * (1 - l_discount)) as revenue

```

```

from
    lineitem,
    part

```

```

where
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#54'
        and p_container in ('SM CASE', 'SM BOX', 'SM PACK',
'SM PKG')
        and l_quantity >= 4 and l_quantity <= 4 + 10
        and p_size between 1 and 5
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )

```

```

    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#55'
        and p_container in ('MED BAG', 'MED BOX', 'MED
PKG', 'MED PACK')
        and l_quantity >= 14 and l_quantity <= 14 + 10
        and p_size between 1 and 10
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )

```

```

    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#45'
        and p_container in ('LG CASE', 'LG BOX', 'LG PACK',
'LG PKG')
        and l_quantity >= 26 and l_quantity <= 26 + 10
        and p_size between 1 and 15
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    ) ;

```

```

commit ;
set qstop = now(*) ;
select 'Stream 6 Query 19 STOP -- ', qstop ;
select 'Stream 6 Query 19 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, ' seconds' ;
-- @(#)20.sql     2.1.8.1
-- TPC-H/TPC-R Potential Part Promotion Query (Q20)
-- Function Query Definition
-- Approved February 1998

```

```

set qstart = now(*) ;
select 'Stream 6 Query 20 START -- ', qstart ;

```



```

select
  s_name,
  s_address
from
  supplier,
  nation
where
  s_suppkey in (
    select
      ps_suppkey
    from
      partsupp
    where
      ps_partkey in (
        select
          p_partkey
        from
          part
        where
          p_name like 'powder%'
      )
    and ps_availqty > (
      select
        0.5 * sum(l_quantity)
      from
        lineitem
      where
        l_partkey = ps_partkey
        and l_suppkey =
ps_suppkey
        and l_shipdate >=
'1997-01-01'
        and l_shipdate < dateadd(year,1,'1997-01-01')
    )
    and s_nationkey = n_nationkey
    and n_name = 'RUSSIA'
)
order by
  s_name;
commit ;
set qstop = now(*);
select 'Stream 6 Query 20 STOP -- ', qstop ;
select 'Stream 6 Query 20 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, 'seconds' ;
-- @(#)17.sql      2.1.8.1
-- TPC-H/TPC-R Small-Quantity-Order Revenue Query (Q17)
-- Functional Query Definition
-- Approved February 1998

set qstart = now(*);
select 'Stream 6 Query 17 START -- ', qstart ;
select
  sum(l_extendedprice) / 7.0 as avg_yearly
from
  lineitem,
  part
where
  p_partkey = l_partkey
  and p_brand = 'Brand#42'
  and p_container = 'WRAP PACK'
  and l_quantity < (
    select
      0.2 * avg(l_quantity)
    from
      lineitem
    where
      l_partkey = p_partkey
  );
commit ;
set qstop = now(*);
select 'Stream 6 Query 17 STOP -- ', qstop ;
select 'Stream 6 Query 17 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)

```

```

as numeric)/1000, 'seconds' ;
-- @(#)21.sql      2.1.8.1
-- TPC-H/TPC-R Suppliers Who Kept Orders Waiting Query (Q21)
-- Functional Query Definition
-- Approved February 1998

```

```

set qstart = now(*);
select 'Stream 6 Query 21 START -- ', qstart ;
select top 100
  s_name,
  count(*) as numwait
from
  supplier,
  lineitem l1,
  orders,
  nation
where
  s_suppkey = l1.l_suppkey
  and o_orderkey = l1.l_orderkey
  and o_orderstatus = 'F'
  and l1.l_receiptdate > l1.l_commitdate
  and exists (
    select
      *
    from
      lineitem l2
    where
      l2.l_orderkey = l1.l_orderkey
      and l2.l_suppkey <> l1.l_suppkey
  )
  and not exists (
    select
      *
    from
      lineitem l3
    where
      l3.l_orderkey = l1.l_orderkey
      and l3.l_suppkey <> l1.l_suppkey
      and l3.l_receiptdate > l3.l_commitdate
  )
  and s_nationkey = n_nationkey
  and n_name = 'UNITED STATES'
)
group by
  s_name
order by
  numwait desc,
  s_name;
commit ;
set qstop = now(*);
select 'Stream 6 Query 21 STOP -- ', qstop ;
select 'Stream 6 Query 21 LENGTH -- ', cast(datediff(millisecond,qstart,qstop)
as numeric)/1000, 'seconds' ;

```

```

select 'Stream completion time -- ',now(*) as "time";

```

```

disconnect;

```

```

exit;

```

ACID Test Execution Code

atomicity test

```

dbtest acid_atomic_main.tst > acid_atomic_main.out

```

consistency test

```
dbtest acid_consistency_main.tst >
acid_consistency_main.out
```

isolation tests

```
dbtest acid_isolation_main1.tst >
acid_isolation_main1.out
dbtest acid_isolation_main2.tst >
acid_isolation_main2.out
dbtest acid_isolation_main3.tst >
acid_isolation_main3.out
dbtest acid_isolation_main4.tst >
acid_isolation_main4.out
dbtest acid_isolation_main5.tst >
acid_isolation_main5.out
dbtest acid_isolation_main6.tst >
acid_isolation_main6.out
```

durability test

```
dbtest acid_durability_main.tst >
acid_durability_main.out
```

ACID Test Source Code

acid_atomic_main.tst

```
stringconnect "dsn=tpch;"

execute {select now(*)} into times
print 'Atomicity test start = ', times
print ' '

include 'acid_functions.tst'
commit

%
% Atomicity test with rollback
%
print ' '
print 'Starting atomicity test with rollback'
print ' '

run test 'acid_atomic_setup.tst'

stringconnect "dsn=tpch;"
let counter=0

LOOP {
open cur2 {select ordr, line, delta from aa_whattodo
where seqnum=^}
substitute counter
print 'counter = ',counter
fetch cur2 into ordr, line, delta
if ROWSTATUS != FOUND then { BREAK LOOP } endif
print 'Acid transaction for: o_key-',ordr,' l_key-',
line,' delta-',delta

execute {select o_totalprice, l_quantity,
l_extendedprice
from orders, lineitem
where o_orderkey = l_orderkey and o_orderkey
=^ and l_linenumber = ^}
substitute ordr, line

into o_total, l_quan, l_price
print 'o_totalprice = ',o_total,' l_quantity =
',l_quan,
' l_extendedprice = ',l_price

execute {call acid_transaction(^, ^, ^, rprice,
quantity,
tax, disc, extprice,
ototal)
} substitute ordr, line, delta
commit
execute {select now(*)} into times
print 'commit : ', times

execute {select o_totalprice, l_quantity,
l_extendedprice
from orders, lineitem
where o_orderkey = l_orderkey and o_orderkey
=^ and l_linenumber = ^}
substitute ordr, line
into o_total, l_quan, l_price
print 'o_totalprice = ',o_total,' l_quantity =

into o_total, l_quan, l_price
print 'o_totalprice = ',o_total,' l_quantity =
',l_quan,
' l_extendedprice = ',l_price

execute {call acid_transaction(^, ^, ^, rprice,
quantity,
tax, disc, extprice,
ototal)
} substitute ordr, line, delta
close cur2
let counter = counter+1

rollback
execute {select now(*)} into times
print 'rollback : ', times

execute {select o_totalprice, l_quantity,
l_extendedprice
from orders, lineitem
where o_orderkey = l_orderkey and o_orderkey
=^ and l_linenumber = ^}
substitute ordr, line
into o_total, l_quan, l_price
print 'o_totalprice = ',o_total,' l_quantity =
',l_quan,
' l_extendedprice = ',l_price
print ' '

} ENDFLOOP

commit

%
% Atomicity test with commit
%
stringconnect "dsn=tpch;"
print ' '
print 'Starting atomicity test with commit '
print ' '
run test 'acid_atomic_setup.tst'

stringconnect "dsn=tpch;"

open curl {select ordr, line, delta from aa_whattodo}
LOOP {
fetch curl into ordr, line, delta
if ROWSTATUS != FOUND then { BREAK LOOP } endif
print 'Acid transaction for: o_key-',ordr,' l_key-',
line,' delta-',delta
execute {select o_totalprice, l_quantity,
l_extendedprice
from orders, lineitem
where o_orderkey = l_orderkey and o_orderkey
=^ and l_linenumber = ^}
substitute ordr, line
into o_total, l_quan, l_price
print 'o_totalprice = ',o_total,' l_quantity =
',l_quan,
' l_extendedprice = ',l_price

execute {call acid_transaction(^, ^, ^, rprice,
quantity,
tax, disc, extprice,
ototal)
} substitute ordr, line, delta
commit
execute {select now(*)} into times
print 'commit : ', times

execute {select o_totalprice, l_quantity,
l_extendedprice
from orders, lineitem
where o_orderkey = l_orderkey and o_orderkey
=^ and l_linenumber = ^}
substitute ordr, line
into o_total, l_quan, l_price
print 'o_totalprice = ',o_total,' l_quantity =
```

```

',l_quan,
      l_extendedprice = ',l_price
print ' '
} ENDLLOOP

close curl
commit

execute {select now(*)} into times
print 'Atomicity test end = ', times

End Test

=====
acid_atomic_setup.tst
=====

Description      "Creates aa_whattodo table"

stringconnect "dsn=tpch;"

% Drop Table if found

allow error -141
execute { commit }
execute { drop table aa_whattodo }
allow no error

execute {
create table aa_whattodo (
      seqnum      int      not null,
      ord         int      not null,
      line        int      null,
      delta       int      null)
}

print 'aa_whattodo CREATED!!'
execute {select now(*)} into times
print 'time = ', times

fetch {select count(*) from aa_whattodo } into ROWS
assert ROWS = 0

print 'Number of rows before load: ',ROWS

LOOP ({let counter = 0}; {counter < 5}; {let counter =
counter + 1})
{
      execute {call generate_acid_values()}
      into orderkey, linenumbr,delta
      execute {insert into aa_whattodo values ( ^ ,
^ , ^ , ^ ) }
      substitute counter, orderkey,
linenumbr, delta
      print counter, ' ',orderkey, ' ',linenumbr,' ',
delta
}
} ENDLLOOP

commit

fetch {select count(*) from aa_whattodo } into ROWS
assert ROWS = 5

print 'Number of rows after load: ',ROWS

disconnect

End Test

=====
acid_consistency_main.tst
=====

stringconnect "dsn=tpch;"

```

```

execute {select now(*)} into times
print 'Consistency test start = ', times
print ' '

include 'acid_functions.tst'

run test 'acid_consistency_setup.tst'

execute {select now(*)} into times
print 'Consistency test time = ', times
print ' '

run test '-o' 'acid_consistency_ql.ot'
'acid_consistency_query.tst'
disconnect

start test '-o' 'acid_consist_user1.ot' 'stream=1'
'acid_consistency_txn.tst'
sleep 1000
start test '-o' 'acid_consist_user2.ot' 'stream=2'
'acid_consistency_txn.tst'
sleep 1000
start test '-o' 'acid_consist_user3.ot' 'stream=3'
'acid_consistency_txn.tst'
sleep 1000
start test '-o' 'acid_consist_user4.ot' 'stream=4'
'acid_consistency_txn.tst'
sleep 1000
start test '-o' 'acid_consist_user5.ot' 'stream=5'
'acid_consistency_txn.tst'
sleep 1000
start test '-o' 'acid_consist_user6.ot' 'stream=6'
'acid_consistency_txn.tst'
sleep 1000
start test '-o' 'acid_consist_user7.ot' 'stream=7'
'acid_consistency_txn.tst'
sleep 1000
start test '-o' 'acid_consist_user8.ot' 'stream=8'
'acid_consistency_txn.tst'
sleep 1000
start test '-o' 'acid_consist_user9.ot' 'stream=9'
'acid_consistency_txn.tst'

synchronize 10
% let the log flush...
sleep 10000

stringconnect "dsn=tpch;"
run test '-o' 'acid_consistency_q2.ot'
'acid_consistency_query.tst'

execute {select now(*)} into times
print 'Consistency test end = ', times
print ' '

End Test

=====
acid_consistency_query.tst
=====

stringconnect "dsn=tpch;"

open curl {select stream, seqnum, ord, line, delta
from acid_table
      where seqnum > 10 order by seqnum}

print ' '

let n=1
LOOP {
      fetch curl into str, seq, ord, lin, delta

      fetch {select round(cast(o_totalprice as
numeric(26,16)),2)
      from orders where o_orderkey=^ }
      substitute ord into o_price

```

```

if ROWSTATUS != FOUND then { BREAK LOOP } endif
if n > 25 then { BREAK LOOP } endif

fetch { call acid_single_query (^) } substitute ord
into l_total

fetch {select cast(^ as numeric(12,2)) } substitute
o_price into o_price
fetch {select cast(^ as numeric(12,2)) } substitute
l_total into l_total

print 'orderkey = ', ord, '          o_totalprice =
', o_price,
      '          acid query = ', l_total

ASSERT (o_price = l_total)
then { print 'Did not compare correctly' }
ENDASSERT
let n=n+1
} ENDLOOP

disconnect

END Test

```

acid_consistency_setup.tst

```

=====
stringconnect "dsn=tpch;"

% Drop Table if found
allow error -141
execute { drop table acid_table }
allow no error

execute {
create table acid_table (
      stream int      not null,
      seqnum  int      not null,
      ordr    int      not null,
      line    int      null,
      delta   int      null)
}

execute {checkpoint}

print 'acid_table CREATED!!'

fetch {select count(*) from acid_table } into ROWS
assert ROWS = 0
print 'Number of rows before load: ',ROWS
commit

LOOP ({let i = 1}; {i <= 9}; { let i = i + 1})
{
  LOOP ({let j = 1}; {j <= 100}; {let j = j + 1})
  {
    execute { call generate_acid_values() } into
    ordr, line, delta
    execute { insert into acid_table values
(^,^,^,^,^) }
      substitute i,j,ordr,line,delta
    } endloop
    print (j-1)*i
  } endloop
}

commit

fetch {select count(*) from acid_table } into ROWS
assert ROWS = 900
print 'Number of rows after load: ',ROWS

End Test
=====

```

acid_consistency_txn.tst

```

=====
stringconnect "dsn=tpch;"

execute {select now(*)} into times
print 'Consistency test start = ', times
print ' '

LOOP ({let i = 1}; {i <= 100}; { let i = i + 1})
{
  fetch {select ordr, line, delta from acid_table
        where stream=^ and seqnum=^ }
        substitute stream, i
  if ROWSTATUS != FOUND then { print 'not enough rows'
    BREAK LOOP }
  endif

  print 'Acid Transaction ',i,
        ': o_key-', ordr , ' l_key-', line , '
delta-', delta

  execute {call acid_transaction( ^, ^, ^, rprice,
                                quantity,
                                tax, disc, extprice,
                                ototal)
        } substitute ordr, line, delta
  commit
  print 'committed'
  sleep 1000
}
ENDLOOP

synchronize 10

End Test

```

acid_durability_main.tst

```

=====
stringconnect "dsn=tpch;"

execute {select now(*)} into times
print 'Durability test start = ', times
print ' '

include 'acid_functions.tst'
run test 'acid_durability_setup.tst'

execute {select now(*)} into times
print 'Durability test time = ', times
print ' '

run test '-o' 'acid_durability_q1.ot'
'acid_durability_query.tst'

start test '-o' 'acid_dura_user1.ot' 'stream=1'
'acid_durability_txn.tst'
sleep 1000
start test '-o' 'acid_dura_user2.ot' 'stream=2'
'acid_durability_txn.tst'
sleep 1000
start test '-o' 'acid_dura_user3.ot' 'stream=3'
'acid_durability_txn.tst'
sleep 1000
start test '-o' 'acid_dura_user4.ot' 'stream=4'
'acid_durability_txn.tst'
sleep 1000
start test '-o' 'acid_dura_user5.ot' 'stream=5'
'acid_durability_txn.tst'
sleep 1000
start test '-o' 'acid_dura_user6.ot' 'stream=6'
'acid_durability_txn.tst'
sleep 1000
start test '-o' 'acid_dura_user7.ot' 'stream=7'

```

```

'acid_durability_txn.tst'
sleep 1000
start test '-o' 'acid_dura_user8.ot' 'stream=8'
'acid_durability_txn.tst'
sleep 1000
start test '-o' 'acid_dura_user9.ot' 'stream=9'
'acid_durability_txn.tst'

synchronize 10

execute {select now(*)} into times
print 'Durability test time = ', times
print ' '

run test '-o' 'acid_durability_q2.ot'
'acid_durability_query.tst'

execute {select now(*)} into times
print 'Durability test end = ', times
print ' '

End Test

=====
acid_durability_query.tst
=====

stringconnect "dsn=tpch;"

open curl {select stream, seqnum, ordr, line, delta
from acid_table
where seqnum > 5 order by seqnum}
print ' '

let n=1
LOOP {
  fetch curl into str, seq, ord, lin, delta

  fetch {select round(cast(o_totalprice as
numeric(26,16)),2)
from orders where o_orderkey=^ }
substitute ord into o_price

  if ROWSTATUS != FOUND then { BREAK LOOP } endif
  if n > 50 then { BREAK LOOP } endif

  fetch { call acid_single_query (^) } substitute ord
into l_total

  fetch {select cast(^ as numeric(12,2)) } substitute
o_price into o_price
  fetch {select cast(^ as numeric(12,2)) } substitute
l_total into l_total

  print 'orderkey = ', ord, ' o_totalprice =
', o_price,
' acid query = ', l_total

  ASSERT (o_price = l_total)
then { print 'Did not compare correctly' }
ENDASSERT
  let n=n+1
} ENDLLOOP

disconnect

END Test

=====
acid_durability_setup.tst
=====

stringconnect "dsn=tpch;"

% Drop Table if found

```

```

allow error -141
execute { drop table acid_table }
allow no error

execute {
create table acid_table (
stream int not null,
seqnum int not null,
ordr int not null,
line int null,
delta int null)
}

execute {checkpoint}

print 'acid_table CREATED!!!'

fetch {select count(*) from acid_table } into ROWS
assert ROWS = 0
print 'Number of rows before load: ',ROWS
commit

LOOP ({let i = 1}; {i <= 9}; { let i = i + 1})
{
  LOOP ({let j = 1}; {j <= 200}; { let j = j + 1})
  {
    execute { call generate_acid_values()} into
ordr, line, delta
    execute { insert into acid_table values
(^,^,^,^,^) }
substitute i,j,ordr,line,delta
  } endloop
  print (j-1)*i
} endloop

commit
execute {checkpoint}

fetch {select count(*) from acid_table } into ROWS
print 'Number of rows after load: ',ROWS

End Test

=====
acid_durability_txn.tst
=====

stringconnect "dsn=tpch;"

execute {select now(*)} into times
print 'Durability test start = ', times
print ' '
print 'stream trans. o_key l_key p_key s_key
delta date_t '

LOOP ({let i = 1}; {i <= 200}; { let i = i + 1})
{
  fetch {select ordr, line, delta from acid_table
where stream=^ and seqnum=^ }
substitute stream, i

  if ROWSTATUS != FOUND then { print 'not enough rows'
BREAK LOOP }
endif

  execute {select l_partkey, l_suppkey from lineitem
where l_orderkey=^ and l_linenum=^}
substitute ordr, line
into p_key, s_key

  execute {call acid_transaction( ^, ^, ^)
} substitute ordr, line, delta
into rprice, quantity, tax, disc, extprice,
ototal

  assert SQLCODE=0 then { DIE } endassert
}

```

```

commit
execute {select now(*)} into times
print stream, '
      txn ', i, '
      ordr, '
      line, '
      p_key, '
      s_key, '
      delta, '
      times, '
sleep 1000
}
ENDLOOP
synchronize 10
End Test
=====
acid_functions.tst
=====
print 'creating the sleep procedure'
allow error -265
execute { DROP PROCEDURE dbo.sleep}
allow no error
execute{ create procedure dbo.sleep(in sleep_time
integer default null)
begin
  declare command varchar(255);
  select 'xp_cmdshell ' 'sleep '+str(sleep_time)+'''
into command;
  execute immediate command
end;
}
print 'creating the Acid Transaction'
allow error -265
execute { DROP PROCEDURE acid_transaction }
allow no error
execute{ CREATE PROCEDURE acid_transaction(
      IN o_key      INT,
      IN l_key      INT,
      IN delta      INT,
      OUT rprice    Numeric(18,8),
      OUT quantity  INT,
      OUT tax        Numeric(18,8),
      OUT disc      Numeric(18,8),
      OUT extprice  Numeric(18,8),
      OUT ototal    Numeric(18,8)
)
ON EXCEPTION RESUME
BEGIN
  DECLARE pkey      INT ;
  DECLARE skey      INT ;
  DECLARE cost      NUMERIC(18,8) ;
  DECLARE new_extprice  NUMERIC(18,8) ;
  DECLARE new_ototal  NUMERIC(18,8) ;
  DECLARE new_quantity  INT ;
  DECLARE c_sqlstate  char(5);
  LOOP1: LOOP
    COMMIT;
    SELECT o_totalprice
      INTO ototal
      FROM orders
      WHERE o_orderkey = o_key ;
    SELECT l_quantity,
      l_extendedprice,
      l_partkey,
      l_suppkey,
      l_tax,
      l_discount
      INTO quantity,
      extprice,
      pkey,
      skey,
      tax,
      disc
      FROM lineitem
      WHERE l_orderkey = o_key
      AND l_linenumber = l_key;
    -- CLEAN UP IMPRECICE NUMBERS
    SET ototal = ototal -
      "TRUNCATE"("truncate"(extprice*(1-disc),2)*(1+tax),2);
    SET rprice = "TRUNCATE"((extprice / quantity),2);
    SET cost = "TRUNCATE"((rprice * delta),2);
    SET new_extprice = extprice + cost;
    SET new_ototal = "TRUNCATE"(new_extprice * (1.0 -
      disc),2);
    SET new_ototal = "TRUNCATE"(new_ototal * (1.0 +
      tax),2);
    SET new_ototal = ototal + new_ototal ;
    SET new_quantity = quantity + delta ;
    --
    -- Update LineItem
    --
    UPDATE lineitem
      SET l_quantity = new_quantity,
      l_extendedprice = new_extprice
      WHERE l_orderkey=o_key
      AND l_linenumber=l_key;
    SELECT SQLSTATE INTO c_sqlstate;
    IF c_sqlstate = '00000' THEN
      --
      -- Update Orders
      --
      UPDATE orders
        SET o_totalprice = new_ototal
        WHERE o_orderkey = o_key;
      SELECT SQLSTATE INTO c_sqlstate;
      IF c_sqlstate = '00000' THEN
        INSERT INTO history VALUES ( pkey, skey,
o_key, l_key, delta, now(*) ) ;
        SELECT SQLSTATE INTO c_sqlstate;
        IF c_sqlstate = '00000' THEN
          LEAVE LOOP1;
        END IF;
      END IF;
    END IF;
  END LOOP LOOP1;
  RETURN(0);
END;
}
print 'Acid transaction created'
print ' '
print 'Creating Acid query'
allow error -265
execute { DROP PROCEDURE acid_single_query }
allow no error
execute{
CREATE PROCEDURE acid_single_query( IN o_key INT, OUT
o_total NUMERIC(26,16) )
BEGIN
  SELECT o_total =
    sum ("truncate" ("truncate"(
      numeric(26,16)),2) *
      round(cast(l_extendedprice as
      numeric(26,16)),2) *
      (1 - round(cast(l_discount as
      numeric(26,16)),2)),2)
      * (1 + round(cast(l_tax as
      numeric(26,16)),2)) , 2))
    FROM lineitem WHERE l_orderkey = o_key;
  END
}
print 'Acid query created'

```

```

print ' '
print 'Creating Generate_acid_values function'
allow error -265
execute { DROP PROCEDURE generate_acid_values }
allow no error

execute{
create procedure generate_acid_values(
out orderkey int,
out linenumber int,
out delta int)
BEGIN

declare seed bigint;
declare rand_dbl double precision;
declare rand_int int;
declare out_key int;

declare times cursor for select
datediff(millisecond,convert(char(10),getdate(),
116),now(*));
declare random1 cursor for select rand(seed);
declare random cursor for select rand();
declare get_supp cursor for
select ps_suppkey from partsupp
where ps_suppkey = rand_int;
declare get_part cursor for
select ps_partkey from partsupp
where ps_suppkey = suppkey;

open times;
fetch next times into seed;
open random1;
fetch next random1 into rand_dbl;

set out_key = 0;
loop1:
while out_key = 0 LOOP
open random;
open get_order;

fetch next random into rand_dbl;
set rand_int = rand_dbl * 6001215 +1;
fetch next get_order into out_key;

close random;
close get_order;
end loop loop1;

set orderkey = out_key;

open get_linenumber;
fetch next get_linenumber into linenumber;
close get_linenumber;

open random;
fetch next random into rand_dbl;
set delta = rand_dbl * 100 + 1;
close random;

END
}
commit
execute {checkpoint}
print 'Generate_acid_values function created'
print ' '

print 'Creating Generate_Ps_Values function'

allow error -265
execute { DROP PROCEDURE generate_ps_values }
allow no error

execute{
create procedure generate_ps_values(
out partkey int,
out suppkey int)
BEGIN

declare seed bigint;
declare rand_dbl double precision;
declare rand_int int;
declare out_key int;
declare counter int;

declare times cursor for select
datediff(millisecond,convert(char(10),getdate(),
116),now(*));
declare random1 cursor for select rand(seed);
declare random cursor for select rand();
declare get_supp cursor for
select ps_suppkey from partsupp
where ps_suppkey = rand_int;
declare get_part cursor for
select ps_partkey from partsupp
where ps_suppkey = suppkey;

open times;
fetch next times into seed;
open random1;
fetch next random1 into rand_dbl;
close random1;

set out_key = 0;
while out_key = 0 LOOP
open random;
open get_supp ;

fetch next random into rand_dbl;
set rand_int = rand_dbl * 10000 +1;
fetch next get_supp into out_key;

close random;
close get_supp ;
end loop;
set suppkey = out_key;

set out_key = 0;
set counter = 0;
open random;
open get_part;
fetch next random into rand_dbl;
set rand_int = rand_dbl * 10 +1;

loop1:
while counter < rand_int LOOP
set counter = counter+1;
fetch next get_part into out_key;
end loop loop1;

set partkey = out_key;
close random;
close get_part;

END
}
commit
execute {checkpoint}
print 'Generate_Ps_Values function created'
print ' '

print 'Creating history table'

allow error -141
execute { drop table history }
allow no error

execute {
create table history (
h_p_key unsigned INT NOT NULL ,
h_s_key unsigned INT NOT NULL ,
h_o_key unsigned INT NOT NULL ,

```

```

h_l_key      INT NOT NULL,
h_delta      INT NOT NULL,
h_date_t     TIMESTAMP NOT NULL)
}

```

```

commit
execute {checkpoint}
print 'history table created'
print ' '

```

acid_isolation_main1.tst

```

stringconnect "dsn=tpch;"

execute {select now(*)} into times
print ' '
print ' '
print 'Isolation test 1'
print 'start = ', times
print ' '

```

```

include 'acid_functions.tst'
include 'acid_isolation_setup.tst'

```

```

start test 'acid_isolation_test1.tst'
start test 'acid_isolation_test1_query.tst'

```

End Test

acid_isolation_main2.tst

```

stringconnect "dsn=tpch;"

execute {select now(*)} into times
print ' '
print ' '
print 'Isolation test 2'
print 'start = ', times
print ' '

```

```

include 'acid_functions.tst'
include 'acid_isolation_setup.tst'

```

```

start test 'acid_isolation_test2.tst'
start test 'acid_isolation_test2_query.tst'

```

End Test

acid_isolation_main3.tst

```

stringconnect "dsn=tpch;"

execute {select now(*)} into times
print ' '
print ' '
print 'Isolation test 3'
print 'start = ', times
print ' '
print 'Isolation test start = ', times

```

```

include "acid_functions.tst"
include 'acid_isolation_setup.tst'

```

```

start test 'acid_isolation_test3_transaction1.tst'
start test 'acid_isolation_test3_transaction2.tst'

```

End Test

acid_isolation_main4.tst

```

stringconnect "dsn=tpch;"

execute {select now(*)} into times
print ' '
print ' '
print 'Isolation test 4'
print 'start = ', times
print ' '
print 'Isolation test start = ', times

```

```

include 'acid_functions.tst'
include 'acid_isolation_setup.tst'

```

```

start test 'acid_isolation_test4_transaction1.tst'
start test 'acid_isolation_test4_transaction2.tst'

```

End Test

acid_isolation_main5.tst

```

stringconnect "dsn=tpch;"

execute {select now(*)} into times
print ' '
print ' '
print 'Isolation test 5'
print 'start = ', times
print ' '

```

```

include 'acid_functions.tst'
include 'acid_isolation_setup.tst'

```

```

start test 'acid_isolation_test5_transaction1.tst'
start test 'acid_isolation_test5_query.tst'

```

End Test

acid_isolation_main6.tst

```

stringconnect "dsn=tpch;"

execute {select now(*)} into times
print ' '
print ' '
print 'Isolation test 6'
print 'start = ', times
print ' '

```

```

include 'acid_functions.tst'
include 'acid_isolation_setup.tst'

```

```

start test '-u' 'acid_isolation_test6_query.tst'
start test 'acid_isolation_test6_transaction1.tst'

```

End Test

acid_isolation_setup.tst

```

stringconnect "dsn=tpch;"

% Drop Table if found

allow error -141
execute { commit }
execute { drop table acid_isolation_table }
allow no error

```



```

execute {
create table acid_isolation_table (
            ordr      int      not null,
            line      int      null,
            delta     int      null)
}

execute {checkpoint}

print 'acid_isolation_table CREATED!!'
execute {select now(*)} into times
print 'time = ', times

fetch {select count(*) from acid_isolation_table }
into ROWS
assert ROWS = 0

print 'Number of rows before load: ',ROWS

execute {call generate_acid_values()} into orderkey,
linenumber,delta
execute {insert into acid_isolation_table values ( ^ ,
^ , ^ ) }
            substitute orderkey, linenumber, delta
print orderkey, ' ',linenumber,' ', delta

commit

fetch {select count(*) from acid_isolation_table }
into ROWS
assert ROWS = 1

print 'Number of rows after load: ',ROWS

disconnect

End Test

```

acid_isolation_test1.tst

```

=====
stringconnect "dsn=tpch;"

execute {select ordr, line, delta from
acid_isolation_table}
            into ordr, line, delta

    print 'The following are the data input values for
the ACID Transaction.'
    print '(user 1) o_key-',ordr, ' l_key-', line, '
delta-',delta

    execute {call acid_transaction( ^, ^, ^,
rprice, quantity, tax, disc, extprice, ototal)
} substitute ordr, line, delta

execute {select now(*)} into times
print 'User 1 waiting to commit = ', times
print ' '
synchronize 2
sleep 10000
execute {select now(*)} into times
print 'User 1 about to commit = ', times
commit

execute { select round(cast(o_totalprice as
numeric(18,2)),2)
            from orders where o_orderkey = ^}
            substitute ordr into o_total
print 'User 1 new values: '
print 'user 1 ordr= ', ordr
print 'user 1 o_total= ', o_total
print ' '

End Test

```

acid_isolation_test1_query.tst

```

=====
stringconnect "dsn=tpch;"

synchronize 2
print ' '
execute {select now(*)} into times
print 'User 2 start query = ', times

execute {select ordr from acid_isolation_table}
            into ordr

print 'user 2 ordr = ', ordr
fetch { call acid_single_query (^) } substitute ordr
into o_total
print 'user 2 o_total=' , o_total
print ' '

execute {select now(*)} into times
print 'User 2 completed query = ', times

disconnect

END Test

```

acid_isolation_test2.tst

```

=====
stringconnect "dsn=tpch;"

execute {select ordr, line, delta from
acid_isolation_table}
            into ordr, line, delta

    print 'The following are the data input values for
the ACID Transaction.'
    print '(user 1) o_key-',ordr, ' l_key-', line, '
delta-',delta

    execute {call acid_transaction( ^, ^, ^,
rprice, quantity, tax, disc, extprice, ototal)
} substitute ordr, line, delta

execute {select now(*)} into times
print 'User 1 waiting to roll back = ', times
print ' '
synchronize 2
sleep 10000
execute {select now(*)} into times
print 'User 1 about to roll back = ', times
rollback

execute { select round(cast(o_totalprice as
numeric(18,2)),2)
            from orders where o_orderkey = ^}
            substitute ordr into o_total
print 'User 1 new values: '
print 'user 1 ordr= ', ordr
print 'user 1 o_total= ', o_total
print ' '

End Test

```

acid_isolation_test2_query.tst

```

=====
stringconnect "dsn=tpch;"

synchronize 2
print ' '
execute {select now(*)} into times

```

```

print 'User 2 start query = ', times

execute {select ordr from acid_isolation_table}
into ordr

print 'user 2 ordr = ', ordr
fetch { call acid_single_query (^) } substitute ordr
into o_total
print 'user 2 o_total=' , o_total
print ' '

execute {select now(*)} into times
print 'User 2 completed query = ', times

disconnect

END Test

```

acid_isolation_test3_transaction1.tst

```

=====
stringconnect "dsn=tpch;"

execute {select now(*)} into times
print 'Isolation test 3 test start = ', times
print ' '

execute {select ordr, line, delta from
acid_isolation_table}
into ordr, line, delta

print 'User 1 -- The input data values for User 1
Acid Transaction.'
print 'User 1 -- o_key = ',ordr
print 'User 1 -- l_key = ',line
print 'User 1 -- delta1 = ',delta

print ' '
execute {select now(*)} into times
print 'User 1 -- Starting the Acid Transaction: ',
times

execute {call acid_transaction( ^, ^, ^ )}
substitute ordr, line, delta
into rprice, quantity, tax, disc, extprice,
ototal

print ' '
execute {select now(*)} into times
print 'User 1 -- Acid Transaction complete: ', times
print '30 second timer started'
SYNCHRONIZE 2
sleep 30000

print ' '
execute {select now(*)} into times
print 'User 1 -- starting commit: ', times

commit
print ' '
execute {select now(*)} into times
print 'User 1 -- transaction commit complete: ',
times

print ' '
print 'USER 1 -- original extendedprice = ', extprice
print 'USER 1 -- original quantity = ', quantity

fetch { select cast(^ as numeric(18,6))
+ (cast(^ as numeric(18,6))*(cast (^ as
numeric(18,6))
/cast (^ as numeric(18,6)))) }
substitute extprice, delta, extprice, quantity
into result1
% make it format nicely...
execute { select cast(^ as numeric(18,2)) }

```

substitute result1 into result2

```

print ' '
print 'User 1 -- result1 = '
print ' txnl_extendedprice + (delta1 *
(txnl_extendedprice/txnl_quantity))'
print 'User 1 -- result1= ', result2
print ' '

disconnect
End Test

```

acid_isolation_test3_transaction2.tst

```

=====
stringconnect "dsn=tpch;"

execute {select ordr, line, delta from
acid_isolation_table}
into ordr, line, delta
% generate a new set of values; we only use delta2
execute { call generate_acid_values()} into ordr2,
line2, delta2

print ' '
print 'User 2 - The input data values for the Acid
Transaction.'
print 'User 2 -- o_key = ',ordr
print 'User 2 -- l_key= ',line
print 'User 2 -- delta2 = ',delta2

SYNCHRONIZE 2
sleep 5000

print ' '
execute {select now(*)} into times
print 'User 2 -- Starting the Acid Transaction: ',
times

execute {call acid_transaction( ^, ^, ^ ) }
substitute ordr, line, delta2
into rprice, quantity, tax, disc, extprice,
ototal
execute {select round(cast(^ as numeric(20,6)),2) }
substitute extprice into extprice2

sleep 5000
print ' '
execute {select now(*)} into times
print 'User 2 -- About to commit: ', times
commit

execute {select now(*)} into times
print 'User 2 -- transaction commit complete: ', times

print ' '

print 'USER 2 -- original extendedprice = ', extprice2
print 'USER 2 -- original quantity = ', quantity
print ' '

End Test

```

acid_isolation_test4_transaction1.tst

```

=====
stringconnect "dsn=tpch;"

execute {select now(*)} into times
print 'Isolation test 3 test start = ', times
print ' '

execute {select ordr, line, delta from
acid_isolation_table}
into ordr, line, delta

```

```

print 'User 1 -- The input data values for User 1
Acid Transaction.'
print 'User 1 -- o_key = ',ordr
print 'User 1 -- l_key = ',line
print 'User 1 -- delta = ',delta

print ' '
execute {select now(*)} into times
print 'User 1 -- Starting the Acid Transaction: ',
times

execute {call acid_transaction( ^, ^, ^ )}
substitute ordr, line, delta
into rprice, quantity, tax, disc, extprice,
ototal

print ' '
execute {select now(*)} into times
print 'User 1 -- Acid Transaction complete: ', times
print '30 second timer started'
SYNCHRONIZE 2
sleep 30000

print ' '
execute {select now(*)} into times
print 'User 1 -- starting rollback: ', times

rollback
print ' '
execute {select now(*)} into times
print 'User 1 -- transaction rollback complete: ',
times

execute {select round(cast(^ as numeric(20,6)),2) }
substitute extprice into extprice2
print ' '
print 'USER 1 -- original extendedprice = ',
extprice2
print 'USER 1 -- original quantity = ', quantity
print ' '

disconnect
End Test

```

acid_isolation_test4_transaction2.tst

```

stringconnect "dsn=tpch;"

execute {select ordr, line, delta from
acid_isolation_table}
into ordr, line, delta
% generate a new set of values; we only use delta2
execute { call generate_acid_values()} into ordr2,
line2, delta2

print ' '
print 'User 2 - The input data values for the Acid
Transaction.'
print 'User 2 -- o_key = ',ordr
print 'User 2 -- l_key= ',line
print 'User 2 -- delta2 = ',delta2

SYNCHRONIZE 2
sleep 5000

print ' '
execute {select now(*)} into times
print 'User 2 -- Starting the Acid Transaction: ',
times

execute {call acid_transaction( ^, ^, ^ ) }
substitute ordr, line, delta2
into rprice, quantity, tax, disc, extprice,
ototal

```

```

execute {select round(cast(^ as numeric(20,6)),2) }
substitute extprice into extprice2

sleep 5000
print ' '
execute {select now(*)} into times
print 'User 2 -- About to commit: ', times
commit

execute {select now(*)} into times
print 'User 2 -- transaction commit complete: ', times
print ' '
print 'USER 2 -- original extendedprice = ', extprice2
print 'USER 2 -- original quantity = ', quantity
print ' '

End Test

```

acid_isolation_test5_query.tst

```

stringconnect "dsn=tpch;"

synchronize 2

execute { call generate_ps_values() } into ps_ptky,
ps_spky
print ' '
print 'user 2 ps_partkey = ', ps_ptky
print 'user 2 ps_suppkey = ', ps_spky
print ' '

execute {select now(*)} into times
print 'User 2 beginning query = ', times
execute {select * from partsupp where ps_partkey=^ and
ps_suppkey=^}
substitute ps_ptky, ps_spky
into ps_ptky, ps_spky, ps_aly, ps_spct, ps_ct

print ' '
print 'User2 gets all columns of the PARTSUPP table '
print ' for selected ps_partkey and ps_suppkey doing a
query.'
print ' '
print 'ps_partkey = ', ps_ptky, ' ps_suppkey = ',
ps_spky
print 'ps_availqty = ', ps_aly, ' ps_supplycost =
',ps_spct
print 'ps_comment = ', ps_ct
execute {select now(*)} into times
print 'User 2 query complete = ', times
print ' '

execute {select now(*)} into times
print 'User 2 about to commit = ', times
commit
execute {select now(*)} into times
print 'User 2 transaction commit complete = ', times

print ' '

End Test

```

acid_isolation_test5_transaction1.tst

```

stringconnect "dsn=tpch;"

execute {select ordr, line, delta from
acid_isolation_table}
into ordr, line, delta

print ' '
print 'The following are the input values for the
users1 ACID Transaction.'

```

```

print 'o_key = ',ordr,'      l_key = ',line,'
delta = ',delta
print ''
execute {select now(*)} into times
print 'User 1 isolation test time = ', times
print ''
print ''
execute {select o_totalprice from orders where
o_orderkey=^ }
      substitute ordr into o_tprice
execute {select l_extendedprice, l_quantity,l_partkey,
l_suppkey
      from lineitem
      where l_orderkey=^ and l_linenumber=^}
      substitute ordr, line
      into l_price, l_quant, l_ptky, l_spky
print 'User 1 o_totalprice = ', o_tprice
print 'User 1 l_extendedprice = ', l_price,'
l_quantity = ', l_quant
print 'User 1 l_partkey      = ', l_ptky,' l_suppkey
= ', l_spky
print ''

execute {select now(*)} into times
print 'User 1 starting acid transaction = ', times

execute {call acid_transaction( ^, ^, ^, rprice,
quantity, tax, disc,
      extprice, ototal) } substitute ordr, line,
delta

execute {select now(*)} into times
print 'User 1 waiting to commit = ', times
print ''
synchronize 2
sleep 10000
execute {select now(*)} into times
print 'User 1 about to commit = ', times
commit
execute {select now(*)} into times
print 'User 1 transaction commit complete = ', times

execute {select o_totalprice from orders where
o_orderkey=^ }
      substitute ordr into o_tprice
execute {select l_extendedprice, l_quantity
      from lineitem where l_orderkey=^ and
l_linenumber=^}
      substitute ordr, line
      into l_price, l_quant
print 'User 1 o_totalprice = ', o_tprice
print 'User 1 l_extendedprice = ', l_price,'
l_quantity = ', l_quant
print 'User 1 l_partkey      = ', l_ptky,' l_suppkey
= ', l_spky

print ''
execute {select * from history where h_o_key=^
and h_date_t=(select max(h_date_t) from
history where h_o_key=^)}
      substitute ordr, ordr
      into hpk, hsk, hok, hlk, hda, hdt

print 'User 1 history entry:'
print ' h_p_key = ', hpk
print ' h_s_key = ', hsk
print ' h_o_key = ', hok
print ' h_l_key = ', hlk
print ' h_delta = ', hda
print ' h_date_t = ', hdt

execute {select now(*)} into times
print 'User 1 isolation test time = ', times
print ''

End Test

```

acid_isolation_test6_query.tst

```

=====
stringconnect "dsn=tpch;"

print 'User1 Query: '
print ''
print 'User1 starts its query (Q1) here.'
execute {select now(*)} into qstart
print 'Start time for User1 Q1 =', qstart
print ''
compare fetchall {select
      l_returnflag,
      l_linestatus,
      sum(l_quantity) as sum_qty,
      sum(l_extendedprice) as sum_base_price,
      sum(l_extendedprice * (1 - l_discount)) as
sum_disc_price,
      sum(l_extendedprice * (1 - l_discount) * (1 +
l_tax)) as sum_charge,
      avg(l_quantity) as avg_qty,
      avg(l_extendedprice) as avg_price,
      avg(l_discount) as avg_disc,
      count(*) as count_order
from lineitem
where l_shipdate <= dateadd(day, -1, '1998-12-01')}
group by l_returnflag,l_linestatus
order by l_returnflag,l_linestatus
} in 'queryresult'

execute {select now(*)} into qstop
print 'Stop time for User1 Q1 =', qstop
print ''

End Test

```

acid_isolation_test6_transaction1.tst

```

=====
stringconnect "dsn=tpch;"

execute {select ordr, line, delta from
acid_isolation_table}
      into ordr, line, delta

execute {select now(*)} into qstart2
print 'User2 acid Transaction = ', qstart2
print 'o_key = ',ordr,'      l_key = ',line,'
delta = ',delta
print ''
execute {select o_totalprice from orders where
o_orderkey=^ }
      substitute ordr into o_tprice
execute {select l_extendedprice, l_quantity,l_partkey,
l_suppkey
      from lineitem where l_orderkey=^ and
l_linenumber=^}
      substitute ordr, line
      into l_price, l_quant, l_ptky, l_spky
print 'User 2 o_totalprice = ', o_tprice
print 'User 2 l_extendedprice = ', l_price,'
l_quantity = ', l_quant
print 'User 2 l_partkey      = ', l_ptky,'
l_suppkey = ', l_spky
print ''

execute {select now(*)} into qstart2
print 'Start Time for User2 Transaction = ', qstart2
print ''
execute {call acid_transaction( ^, ^, ^, rprice,
quantity,
      tax, disc, extprice,
ototal) }
      substitute ordr, line, delta

execute {select now(*)} into qstop2

```

```

print 'User 2 about to commit = ', qstop2
commit
execute {select now(*)} into qstop2
print 'User 2 transaction commit complete = ', qstop2
print ' '

```

```

2 backup wu 0 - 8920 68.34GB
3 unassigned wm 1176 - 7442 48.01GB
4 unassigned wm 5001 - 6306 10.00GB
7 unassigned wm 1 - 13 101.98MB
8 boot wu 0 - 0 7.84MB

```

```

execute {select o_totalprice from orders where
o_orderkey=^ }
substitute ordr
into o_tprice
execute {select l_extendedprice, l_quantity
from lineitem where l_orderkey=^ and
l_linenumber=^}
substitute ordr, line
into l_price, l_quant
print 'User 2 o_totalprice = ', o_tprice
print 'User 2 l_extendedprice = ', l_price, '
l_quantity = ', l_quant
print 'User 2 l_partkey = ', l_ptky, '
l_suppkey = ', l_spky
print ' '

```

```

c1t2d0
-----
1 unassigned wm 14 - 1175 8.90GB
2 backup wu 0 - 8920 68.34GB
3 unassigned wm 1176 - 7442 48.01GB
4 unassigned wm 5001 - 6306 10.00GB
7 unassigned wm 1 - 13 101.98MB
8 boot wu 0 - 0 7.84MB

```

```

print ' '
execute {select * from history
where h_o_key=^
and h_date_t=(select max(h_date_t) from
history where h_o_key=^)}
substitute ordr, ordr
into hpk, hsk, hok, hlk, hda, hdt
print 'User 2 history entry:'
print ' h_p_key = ', hpk
print ' h_s_key = ', hsk
print ' h_o_key = ', hok
print ' h_l_key = ', hlk
print ' h_delta = ', hda
print ' h_date_t = ', hdt

```

```

c1t3d0
-----
1 unassigned wm 1 - 1175 9.00GB
2 backup wu 0 - 8920 68.34GB
3 unassigned wm 1176 - 7442 48.01GB
4 unassigned wm 5001 - 6306 10.00GB
8 boot wu 0 - 0 7.84MB

```

```

print ' '
execute {select now(*)} into times
print 'User 2 completed = ', times

```

```

c1t4d0
-----
1 unassigned wm 1 - 1175 9.00GB
2 backup wu 0 - 8920 68.34GB
3 unassigned wm 1176 - 7442 48.01GB
4 unassigned wm 5001 - 6306 10.00GB
8 boot wu 0 - 0 7.84MB

```

End Test

```

c1t5d0
-----
1 unassigned wm 1 - 1175 9.00GB
2 backup wu 0 - 8920 68.34GB
3 unassigned wm 1176 - 7442 48.01GB
4 unassigned wm 5001 - 6306 10.00GB
8 boot wu 0 - 0 7.84MB

```

Disk Configuration Details

```

c1t6d0
-----
1 unassigned wm 1 - 1175 9.00GB
2 backup wu 0 - 8920 68.34GB
3 unassigned wm 1176 - 7442 48.01GB
4 unassigned wm 5001 - 6306 10.00GB
8 boot wu 0 - 0 7.84MB

```

Solaris Volume Manager Setup

Note: The instructions below pertain to the controller number and targets generated by the configuration used in the benchmark. Solaris chooses the controller numbers and target numbers at boot time depending upon the cabling configuration and slot location of the HBAs. Thus another equivalently configured system may not have the same the controller numbers and SCSI targets as shown below.

```

c1t7d0
-----
1 unassigned wm 1 - 1175 9.00GB
2 backup wu 0 - 8920 68.34GB
3 unassigned wm 1176 - 7442 48.01GB
4 unassigned wm 5001 - 6306 10.00GB
8 boot wu 0 - 0 7.84MB

```

Using the **format** command, partition the disks as follows

```

c1t8d0
-----
1 unassigned wm 1 - 1175 9.00GB
2 backup wu 0 - 8920 68.34GB
3 unassigned wm 1176 - 7442 48.01GB
4 unassigned wm 5001 - 6306 10.00GB
8 boot wu 0 - 0 7.84MB

```

```

c1t0d0
-----
0 root wm 8001 - 8920 7.05GB
1 swap wu 14 - 1175 8.90GB
2 backup wm 0 - 8920 68.34GB
3 unassigned wm 1176 - 7442 48.01GB
4 unassigned wm 7443 - 8920 11.32GB
7 unassigned wm 1 - 13 101.98MB
8 boot wu 0 - 0 7.84MB

```

```

c1t9d0
-----
1 unassigned wm 1 - 1175 9.00GB
2 backup wu 0 - 8920 68.34GB
3 unassigned wm 1176 - 7442 48.01GB
4 unassigned wm 5001 - 6306 10.00GB
8 boot wu 0 - 0 7.84MB

```

```

c1t1d0
-----
1 unassigned wm 14 - 1175 8.90GB

```

```

clt10d0
-----
 1 unassigned   wm      1 - 1175      9.00GB
 2   backup     wu      0 - 8920      68.34GB
 3 unassigned   wm    1176 - 7442    48.01GB
 4 unassigned   wm    5001 - 6306    10.00GB
 8   boot       wu      0 - 0           7.84MB

```

```

clt11d0
-----
 1 unassigned   wm      1 - 1175      9.00GB
 2   backup     wu      0 - 8920      68.34GB
 3 unassigned   wm    1176 - 7442    48.01GB
 4 unassigned   wm    5001 - 6306    10.00GB
 8   boot       wu      0 - 0           7.84MB

```

```

clt12d0
-----
 1 unassigned   wm      1 - 1175      9.00GB
 2   backup     wu      0 - 8920      68.34GB
 3 unassigned   wm    1176 - 7442    48.01GB
 4 unassigned   wm    5001 - 6306    10.00GB
 8   boot       wu      0 - 0           7.84MB

```

```

clt13d0
-----
 1 unassigned   wm      1 - 1175      9.00GB
 2   backup     wu      0 - 8920      68.34GB
 3 unassigned   wm    1176 - 7442    48.01GB
 4 unassigned   wm    5001 - 6306    10.00GB
 8   boot       wu      0 - 0           7.84MB

```

```

clt14d0
-----
 1 unassigned   wm      1 - 1175      9.00GB
 2   backup     wu      0 - 8920      68.34GB
 3 unassigned   wm    1176 - 7442    48.01GB
 4 unassigned   wm    5001 - 6306    10.00GB
 8   boot       wu      0 - 0           7.84MB

```

```

clt15d0
-----
 1 unassigned   wm      1 - 1175      9.00GB
 2   backup     wu      0 - 8920      68.34GB
 3 unassigned   wm    1176 - 7442    48.01GB
 4 unassigned   wm    5001 - 6306    10.00GB
 8   boot       wu      0 - 0           7.84MB

```

Next define

```

/dev/dsk/clt0d0s4
as a swap device.

```

Then use **svm** as follows to create the mirrors used for the IQ log and the database devices

```

metainit d2 1 1 clt6d0s1
metainit d3 1 1 clt7d0s1
metainit d1 -m d2 d3

```

```

metainit d10 1 1 clt0d0s4
metainit d11 1 1 clt1d0s4
metainit d15 -m d10 d11

```

```

metainit d20 1 1 clt2d0s4
metainit d21 1 1 clt3d0s4

```

```

metainit d25 -m d20 d21
metainit d30 1 1 clt4d0s4
metainit d31 1 1 clt5d0s4
metainit d35 -m d30 d31
metainit d40 1 1 clt6d0s4
metainit d41 1 1 clt7d0s4
metainit d45 -m d40 d41
metainit d50 1 1 clt8d0s4
metainit d51 1 1 clt9d0s4
metainit d55 -m d50 d51
metainit d60 1 1 clt10d0s4
metainit d61 1 1 clt11d0s4
metainit d65 -m d60 d61
metainit d70 1 1 clt12d0s4
metainit d71 1 1 clt13d0s4
metainit d75 -m d70 d71
metainit d80 1 1 clt14d0s4
metainit d81 1 1 clt15d0s4
metainit d85 -m d80 d81

```

File System Setup

run the following:

```

create a filesystem on /dev/md/rdisk/d1 and mount
/dev/md/dsk/d1 on /sybase2

```

```

newfs /dev/md/rdisk/d1

```

```

mount /dev/md/dsk/d1 /sybase2

```

```

echo "/dev/md/dsk/d1 /dev/md/rdisk/d1 /sybase2 ufs
1 yes -" >>/etc/vfstab

```

Database Device Links

Finally create the following links in /sybase2 to be used as temp devices:

```

/sybase2/T01 -> /dev/rdisk/clt0d0s1
/sybase2/T02 -> /dev/rdisk/clt1d0s1
/sybase2/T03 -> /dev/rdisk/clt2d0s1
/sybase2/T04 -> /dev/rdisk/clt3d0s1
/sybase2/T05 -> /dev/rdisk/clt4d0s1
/sybase2/T06 -> /dev/rdisk/clt5d0s1
/sybase2/T07 -> /dev/rdisk/clt6d0s1
/sybase2/T08 -> /dev/rdisk/clt7d0s1
/sybase2/T09 -> /dev/rdisk/clt8d0s1
/sybase2/T10 -> /dev/rdisk/clt9d0s1
/sybase2/T11 -> /dev/rdisk/clt10d0s1
/sybase2/T12 -> /dev/rdisk/clt11d0s1
/sybase2/T13 -> /dev/rdisk/clt12d0s1
/sybase2/T14 -> /dev/rdisk/clt13d0s1

```

and create the following links in /sybase2 to be used as database devices:

```

/sybase2/M01 -> /dev/md/rdisk/d15

```

```
/sybase2/M02 -> /dev/md/rdisk/d25  
/sybase2/M03 -> /dev/md/rdisk/d35  
/sybase2/M04 -> /dev/md/rdisk/d45  
/sybase2/M05 -> /dev/md/rdisk/d55  
/sybase2/M06 -> /dev/md/rdisk/d65  
/sybase2/M07 -> /dev/md/rdisk/d75  
/sybase2/M08 -> /dev/md/rdisk/d85
```

Appendix C. Query Text and Query Output

qualification query 1

```
=====
% select
% l_returnflag,
% l_linestatus,
% sum(l_quantity) as sum_qty,
% sum(l_extendedprice) as sum_base_price,
% sum(l_extendedprice * (1 - l_discount)) as
sum_disc_price,
% sum(l_extendedprice * (1 - l_discount) * (1 +
l_tax)) as sum_charge,
% avg(l_quantity) as avg_qty,
% avg(l_extendedprice) as avg_price,
% avg(l_discount) as avg_disc,
% count(*) as count_order
% from
% lineitem
% where
% l_shipdate <= dateadd(day,-90,'1998-12-01')
% group by
% l_returnflag,
% l_linestatus
% order by
% l_returnflag,
% l_linestatus;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.32000 seconds - current
time 16:40:13
'A', 'F', 37734107, 56586554400.7292032, 53758257134.86945
63, 55909065222.8284717, 25.5220058532573342, 38273.12973
46211374, .0499852958383577168, 1478493
'N', 'F', 991417, 1487504710.38000107, 1413082168.05409968
, 1469649223.19436967, 25.5164719205229819, 38284.4677608
483374, .0500934266742134809, 38854
'N', 'O', 74476040, 111701729697.737336, 106118230307.6073
83, 110367043872.495174, 25.5022267695849895, 38249.11798
8907361, .049996586053555131, 2920374
'R', 'F', 37719753, 56568041380.8983326, 53741292684.60453
75, 55889619119.8339581, 25.5057936126907617, 38250.85462
60985255, .0500094058300870121, 1478870
% total of 4 rows written
=====
```

qualification query 2

```
=====
% select top 100
% s_acctbal,
% s_name,
% n_name,
% p_partkey,
% p_mfgr,
% s_address,
% s_phone,
% s_comment
% from
% part,
% supplier,
% partsupp,
% nation,
% region
% where
% p_partkey = ps_partkey
% and s_suppkey = ps_suppkey
% and p_size = 15
```

```
% and p_type like 'BRASS'
% and s_nationkey = n_nationkey
% and n_regionkey = r_regionkey
% and r_name = 'EUROPE'
% and ps_supplycost = (
% select
% min(ps_supplycost)
% from
% partsupp,
% supplier,
% nation,
% region
% where
% p_partkey = ps_partkey
% and s_suppkey = ps_suppkey
% and s_nationkey = n_nationkey
% and n_regionkey = r_regionkey
% and r_name = 'EUROPE'
% )
% order by
% s_acctbal desc,
% n_name,
% s_name,
% p_partkey;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.76000 seconds - current
time 16:40:25
9938.53, 'Supplier#000005359', 'UNITED KINGDOM
', 185358, 'Manufacturer#4
', 'QKuHYh, vZGiwu2FWEJoLDx04', '33-429-790-6131', 'blithe
ly silent pinto beans are furiously. slyly final
deposits across'
9937.84, 'Supplier#000005969', 'ROMANIA
', 108438, 'Manufacturer#1
', 'ANDENSOSmk, miq23Xfb5Rwt6dvUcvt6Qa', '29-520-692-3537
', 'carefully slow deposits use furiously. slyly ironic
platelets above the ironic'
9936.22, 'Supplier#000005250', 'UNITED KINGDOM
', 249, 'Manufacturer#4
', 'B3rqp0xbSEim4Mpy2RH
J', '33-320-228-2957', 'blithely special packages are.
stealthily express deposits across the closely final
instructi'
9923.77000000000119, 'Supplier#000002324
', 'GERMANY
', 29821, 'Manufacturer#4
', 'y30D9UywSTOk', '17-779-299-1839', 'quickly express
packages breach quiet pinto beans. requ'
9871.22, 'Supplier#000006373', 'GERMANY
', 43868, 'Manufacturer#5
', 'J8fcXWstqM', '17-813-485-8637', 'never silent
deposits integrate furiously blit'
9870.78, 'Supplier#000001286', 'GERMANY
', 81285, 'Manufacturer#2
', 'YKA, E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH', '17-516-924-
4574', 'final theodolites cajole slyly special,'
9870.78, 'Supplier#000001286', 'GERMANY
', 181285, 'Manufacturer#4
', 'YKA, E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH', '17-516-924-
4574', 'final theodolites cajole slyly special,'
9852.52000000000119, 'Supplier#000008973
', 'RUSSIA
', 18972, 'Manufacturer#2
', 't5L67YdBYH6o, Vz24jpDyQ9', '32-188-594-7038', 'quickl
y regular instructions wake-- carefully unusual braids
into the expres'
9847.83, 'Supplier#000008097', 'RUSSIA
', 130557, 'Manufacturer#2
', 'xMe97bpE69NzdwLoX', '32-375-640-3593', 'slyly regular
dependencies sleep slyly furiously express dep'
9847.57, 'Supplier#000006345', 'FRANCE
', 86344, 'Manufacturer#1
', 'Vst3rzK3qG698u6ld8HhObYvvrTcSTsvQldDQDag', '16-886-766
-7945', 'silent pinto beans should have to snooze
carefully along the final reques'
% total of 100 rows written
```


=====
qualification query 3
=====

```
% select top 10
% l_orderkey,
% sum(l_extendedprice * (1 - l_discount)) as revenue,
% o_orderdate,
% o_shippriority
% from
% customer,
% orders,
% lineitem
% where
% c_mktsegment = 'BUILDING'
% and c_custkey = o_custkey
% and l_orderkey = o_orderkey
% and o_orderdate < '1995-03-15'
% and l_shipdate > '1995-03-15'
% group by
% l_orderkey,
% o_orderdate,
% o_shippriority
% order by
% revenue desc,
% o_orderdate;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.36000 seconds - current
time 16:40:27
2456423,406181.011100000024,'1995-03-05',0
3459808,405838.698899999917,'1995-03-04',0
492164,390324.061,'1995-02-19',0
1188320,384537.935899999976,'1995-03-09',0
2435712,378673.055799999952,'1995-02-26',0
4878020,378376.795200000048,'1995-03-12',0
5521732,375153.9215,'1995-03-13',0
2628192,373133.309399999976,'1995-02-22',0
993600,371407.459499999994,'1995-03-05',0
2300070,367371.145200000107,'1995-03-13',0
% total of 10 rows written
```

=====
qualification query 4
=====

```
% select
% o_orderpriority,
% count(*) as order_count
% from
% orders
% where
% o_orderdate >= '1993-07-01'
% and o_orderdate < dateadd(month,3,'1993-07-01')
% and exists (
% select
% *
% from
% lineitem
% where
% l_orderkey = o_orderkey
% and l_commitdate < l_receiptdate
% )
% group by
% o_orderpriority
% order by
% o_orderpriority;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.25000 seconds - current
time 16:40:31
```

```
'1-URGENT',10594
'2-HIGH',10476
'3-MEDIUM',10410
'4-NOT SPECIFIED',10556
'5-LOW',10487
% total of 5 rows written
```

=====
qualification query 5
=====

```
% select
% n_name,
% sum(l_extendedprice * (1 - l_discount)) as revenue
% from
% customer,
% orders,
% lineitem,
% supplier,
% nation,
% region
% where
% c_custkey = o_custkey
% and l_orderkey = o_orderkey
% and l_suppkey = s_suppkey
% and c_nationkey = s_nationkey
% and s_nationkey = n_nationkey
% and n_regionkey = r_regionkey
% and r_name = 'ASIA'
% and o_orderdate >= '1994-01-01'
% and o_orderdate < dateadd(year,1,'1994-01-01')
% group by
% n_name
% order by
% revenue desc;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.65000 seconds - current
time 16:40:36
'INDONESIA',55502041.1696999431
'VIETNAM',55295086.9966999531
'CHINA',53724494.2565999746
'INDIA',52035512.000200057
'JAPAN',45410175.6954000235
% total of 5 rows written
```

=====
qualification query 6
=====

```
% select
% sum(l_extendedprice * l_discount) as revenue
% from
% lineitem
% where
% l_shipdate >= '1994-01-01'
% and l_shipdate < dateadd(year,1,'1994-01-01')
% and l_discount between .06 - 0.01 and .06 + 0.01
% and l_quantity < 24;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.15000 seconds - current
time 16:40:41
123141078.228299007
% total of 1 rows written
```

=====
qualification query 7
=====

```

% select
% supp_nation,
% cust_nation,
% l_year,
% sum(volume) as revenue
% from
% (
% select
% n1.n_name as supp_nation,
% n2.n_name as cust_nation,
% datepart(year, l_shipdate) as l_year,
% l_extendedprice * (1 - l_discount) as volume
% from
% supplier,
% lineitem,
% orders,
% customer,
% nation n1,
% nation n2
% where
% s_suppkey = l_suppkey
% and o_orderkey = l_orderkey
% and c_custkey = o_custkey
% and s_nationkey = n1.n_nationkey
% and c_nationkey = n2.n_nationkey
% and (
% (n1.n_name = 'FRANCE' and n2.n_name = 'GERMANY')
% or (n1.n_name = 'GERMANY' and n2.n_name = 'FRANCE')
% )
% and l_shipdate between '1995-01-01' and '1996-12-31'
% ) as shipping
% group by
% supp_nation,
% cust_nation,
% l_year
% order by
% supp_nation,
% cust_nation,
% l_year;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.67000 seconds - current
time 16:40:43
'FRANCE', 'GERMANY'
',1995,54639732.7335999489
'FRANCE', 'GERMANY'
',1996,54633083.3075999737
'GERMANY', 'FRANCE'
',1995,52531746.6696999669
'GERMANY', 'FRANCE'
',1996,52520549.0223998487
% total of 4 rows written

```

qualification query 8

```

% select
% o_year,
% sum(case
% when nation = 'BRAZIL' then volume
% else 0
% end) / sum(volume) as mkt_share
% from
% (
% select
% datepart(year, o_orderdate) as o_year,
% l_extendedprice * (1 - l_discount) as volume,
% n2.n_name as nation
% from
% part,
% supplier,
% lineitem,
% orders,

```

```

% customer,
% nation n1,
% nation n2,
% region
% where
% p_partkey = l_partkey
% and s_suppkey = l_suppkey
% and l_orderkey = o_orderkey
% and o_custkey = c_custkey
% and c_nationkey = n1.n_nationkey
% and n1.n_regionkey = r_regionkey
% and r_name = 'AMERICA'
% and s_nationkey = n2.n_nationkey
% and o_orderdate between '1995-01-01' and
'1996-12-31'
% and p_type = 'ECONOMY ANODIZED STEEL'
% ) as all_nations
% group by
% o_year
% order by
% o_year;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.86000 seconds - current
time 16:40:47
1995,.0344358904066548347
1996,.041485521293530345
% total of 2 rows written

```

qualification query 9

```

% select
% nation,
% o_year,
% sum(amount) as sum_profit
% from
% (
% select
% n_name as nation,
% datepart(year, o_orderdate) as o_year,
% l_extendedprice * (1 - l_discount) - ps_supplycost *
l_quantity as
% amount
% from
% part,
% supplier,
% lineitem,
% partsupp,
% orders,
% nation
% where
% s_suppkey = l_suppkey
% and ps_suppkey = l_suppkey
% and ps_partkey = l_partkey
% and p_partkey = l_partkey
% and o_orderkey = l_orderkey
% and s_nationkey = n_nationkey
% and p_name like 'green'
% ) as profit
% group by
% nation,
% o_year
% order by
% nation,
% o_year desc;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.69000 seconds - current
time 16:40:48
'ALGERIA',1998,31342867.2345000029

```

```

'ALGERIA                ',1997,57138193.0233001232
'ALGERIA                ',1996,56140140.1330001235
'ALGERIA                ',1995,53051469.6533999741
'ALGERIA                ',1994,53867582.128600049
'ALGERIA                ',1993,54942718.132400012
'ALGERIA                ',1992,54628034.7126999021
'ARGENTINA              ',1998,30211185.708099997
'ARGENTINA              ',1997,50805741.752300003
'ARGENTINA              ',1996,51923746.5754999459
% total of 175 rows written

```

=====
qualification query 10
=====

```

% select top 20
% c_custkey,% c_name,
% sum(l_extendedprice * (1 - l_discount)) as revenue,
% c_acctbal,
% n_name,
% c_address,
% c_phone,
% c_comment
% from
% customer,
% orders,
% lineitem,
% nation
% where
% c_custkey = o_custkey
% and l_orderkey = o_orderkey
% and o_orderdate >= '1993-10-01'
% and o_orderdate < dateadd(month,3,'1993-10-01')
% and l_returnflag = 'R'
% and c_nationkey = n_nationkey
% group by
% c_custkey,
% c_name,
% c_acctbal,
% c_phone,
% n_name,
% c_address,
% c_comment
% order by
% revenue desc;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.50000 seconds - current
time 16:40:55
57040,'Customer#000057040',734235.2455,632.87,'JAPAN
','Eioyjf4pp','22-895-641-3466','requests sleep
blithely about the furiously i'
143347,'Customer#000143347',721002.694799999952,2557.4
700000000003,'EGYPT
','laReFYv,Kw4','14-742-935-3718','fluffily bold
excuses haggle finally after the u'
60838,'Customer#000060838',679127.307700000048,2454.77
,'BRAZIL
','64EaJ5vMAHWJlBOxJklpNc2RJiWE','12-913-494-9813','fu
riously even pinto beans integrate under the ruthless
foxes; ironic, even dolphins across the slyl'
101998,'Customer#000101998',637029.566699999809,3790.8
9,'UNITED KINGDOM
','01c9CILnNtfoQYmZj','33-593-865-6378','accounts doze
blithely! enticing, final deposits sleep blithely
special accounts. slyly express accounts pla'
125341,'Customer#000125341',633508.086,4983.5100000000
006,'GERMANY
','S290DD6bceU8QSuuEJznkNaK','17-582-695-5962','quickl
y express requests wake quickly blithely'
25501,'Customer#000025501',620269.784899999976,7725.04
,'ETHIOPIA
','W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ','15-874-808-6793',
'quickly special requests sleep evenly among the

```

```

special deposits. special deposi'
115831,'Customer#000115831',596423.867200000167,5098.1
,'FRANCE
','rFeBbEEyk dl
ne7zV5fDrmiql0K09wV7pxqCgIc','16-715-386-3788','carefu
lly bold excuses sleep alongside of the thinly idle'
84223,'Customer#000084223',594998.0238999999976,528.65,
'UNITED KINGDOM
','nAVZCs6BaWap rRM27N
2qBnzc5WBauxbA','33-442-824-8191','pending, final
ideas haggle final requests. unusual, regular
asymptotes affix according to the even foxes.'
54289,'Customer#000054289',585603.391799999952,5583.02
,'IRAN
','vXCxoCsU0Bad5JQI
,ookbZ','20-834-292-4707','express requests sublate
blithely regular requests. regular, even ideas solve.'
39922,'Customer#000039922',584878.113399999976,7321.10
99999999981,'GERMANY
','Zgy4s5012GKN4pLDPBU8m342gIw6R','17-147-757-8036','e
ven pinto beans haggle. slyly bold accounts inte'
6226,'Customer#000006226',576783.760599999905,2230.09,
'UNITED KINGDOM
','8gPu8,NPGkfyQQ0hcIYUGPIBwc,ybP5g','33-657-701-3391
','quickly final requests against the regular
instructions wake blithely final instructions. pa'
922,'Customer#00000922',576767.533299999833,3869.25,'
GERMANY
','Az9RFaut7NkPnc5zSD2PwHgVvr4jRzq','17-945-916-9648',
'boldly final requests cajole blith'
147946,'Customer#000147946',576455.132,2030.1300000000
003,'ALGERIA
','iANyZHjghyy7AjahOpTrYyhJ','10-886-956-3143','furiou
sly even accounts are blithely above the furiousl'
115640,'Customer#000115640',569341.193299999952,6436.1
,'ARGENTINA
','Vtgfia9qI
7EpHgecU1X','11-411-543-4901','final instructions are
slyly according to the'
73606,'Customer#000073606',568656.857799999952,1785.67
,'JAPAN
','xuR0Tro5yChDfOCrjkd2ol','22-437-653-6966','furiousl
y bold orbits about the furiously busy requests wake
across the furiously quiet theodolites. d'
110246,'Customer#000110246',566842.981499999881,7763.3
5,'VIETNAM
','7KzflgX
MDOq7sOkI','31-943-426-9837','dolphins sleep blithely
among the slyly final'
142549,'Customer#000142549',563537.236799999952,5085.9
89999999994,'INDONESIA
','ChqEoK43OysjdHbtKCp6dKqjNyvvv9','19-955-562-2398',
'regular, unusual dependencies boost slyly; ironic
attainments nag fluffily into the unusual packages?'
146149,'Customer#000146149',557254.9865,1791.55,'ROMAN
IA
','s87fvzFQpU','29-744-164-6487','silent, unusual
requests detect quickly slyly regul'
52528,'Customer#000052528',556397.350899999976,551.79,
'ARGENTINA
','NFztyTOR10UOJ','11-208-192-3205','unusual requests
detect. slyly dogged theodolites use slyly. deposit'
23431,'Customer#000023431',554269.536000000119,3381.86
,'ROMANIA
','HgiV0phghaIa9aydNoIlb','29-915-458-2654','instructi
ons nag quickly. furiously bold accounts cajol'
% total of 20 rows written

```

=====
qualification query 11
=====

```

% select
% ps_partkey,
% sum(ps_supplycost * ps_availqty) as value
% from
% partsupp,
% supplier,
% nation
% where
% ps_suppkey = s_suppkey
% and s_nationkey = n_nationkey

```

```

% and n_name = 'GERMANY'
% group by
% ps_partkey having
% sum(ps_supplycost * ps_availqty) > (
% select
% sum(ps_supplycost * ps_availqty) * 0.0001000000
% from
% partsupp,
% supplier,
% nation
% where
% ps_suppkey = s_suppkey
% and s_nationkey = n_nationkey
% and n_name = 'GERMANY'
% )
% order by
% value desc;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.49000 seconds - current
time 16:41:01
129760,17538456.85999999994
166726,16503353.91999999988
191287,16474801.96999999988
161758,16101755.53999999976
34452,15983844.72000000018
139035,15907078.34000000006
9403,15451755.61999999988
154358,15212937.87999999982
38823,15064802.85999999994
85606,15053957.1500000003
% total of 1048 rows written

```

qualification query 12

```

% select
% l_shipmode,
% sum(case
% when o_orderpriority = '1-URGENT'
% or o_orderpriority = '2-HIGH'
% then 1
% else 0
% end) as high_line_count,
% sum(case
% when o_orderpriority <> '1-URGENT'
% and o_orderpriority <> '2-HIGH'
% then 1
% else 0
% end) as low_line_count
% from
% orders,
% lineitem
% where
% o_orderkey = l_orderkey
% and l_shipmode in ('MAIL', 'SHIP')
% and l_commitdate < l_receiptdate
% and l_shipdate < l_commitdate
% and l_receiptdate >= '1994-01-01'
% and l_receiptdate < dateadd(year,1,'1994-01-01')
% group by
% l_shipmode
% order by
% l_shipmode;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.26000 seconds - current
time 16:41:03
'MAIL      ',6202,9324
'SHIP      ',6200,9262

```

```

% total of 2 rows written

```

qualification query 13

```

% select
% c_count,
% count(*) as custdist
% from
% (
% select
% c_custkey,
% count(o_orderkey)
% from
% customer left outer join orders on
% c_custkey = o_custkey
% and o_comment not like 'specialrequests'
% group by
% c_custkey
% ) as c_orders (c_custkey, c_count)
% group by
% c_count
% order by
% custdist desc,
% c_count desc;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.17000 seconds - current
time 16:41:06
0,50004
9,6641
10,6566
11,6058
8,5949
12,5553
13,4989
19,4748
7,4707
18,4625
% total of 42 rows written

```

qualification query 14

```

% select
% 100.00 * sum(case
% when p_type like 'PROMO'
% then l_extendedprice * (1 - l_discount)
% else 0
% end) / sum(l_extendedprice * (1 - l_discount)) as
promo_revenue
% from
% lineitem,
% part
% where
% l_partkey = p_partkey
% and l_shipdate >= '1995-09-01'
% and l_shipdate < dateadd(month,1,'1995-09-01');
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.24000 seconds - current
time 16:41:19
16.3807786263955563
% total of 1 rows written

```

qualification query 15

```

Executing command:
% create view revenue0 (supplier_no, total_revenue) as
% select
% l_suppkey,
% sum(l_extendedprice * (1 - l_discount))
% from
% lineitem
% where
% l_shipdate >= '1996-01-01'
% and l_shipdate < dateadd(month,3,'1996-01-01')
% group by
% l_suppkey;
% execution time 0.81000 seconds - current time
16:41:21

```

```

Executing command:
%
% select
% s_suppkey,
% s_name,
% s_address,
% s_phone,
% total_revenue
% from
% supplier,
% revenue0
% where
% s_suppkey = supplier_no
% and total_revenue = (
% select
% max(total_revenue)
% from
% revenue0
% )
% order by
% s_suppkey;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.27000 seconds - current
time 16:41:21
8449,'Supplier#000008449
','Wp34zim9qYFbVctdW','20-469-856-8873',1772627.208700
00005
% total of 1 rows written

```

qualification query 16

```

% select
% p_brand,
% p_type,
% p_size,
% count(distinct ps_suppkey) as supplier_cnt
% from
% partsupp,
% part
% where
% p_partkey = ps_partkey
% and p_brand <> 'Brand#45'
% and p_type not like 'MEDIUM POLISHED'
% and p_size in (49, 14, 23, 45, 19, 3, 36, 9)
% and ps_suppkey not in (
% select
% s_suppkey
% from
% supplier
% where
% s_comment like 'CustomerComplaints'
% )
% group by
% p_brand,
% p_type,

```

```

% p_size
% order by
% supplier_cnt desc,
% p_brand,
% p_type,
% p_size;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.27000 seconds - current
time 16:41:22
'Brand#41 ','MEDIUM BRUSHED TIN',3,28
'Brand#54 ','STANDARD BRUSHED COPPER',14,27
'Brand#11 ','STANDARD BRUSHED TIN',23,24
'Brand#11 ','STANDARD BURNISHED BRASS',36,24
'Brand#15 ','MEDIUM ANODIZED NICKEL',3,24
'Brand#15 ','SMALL ANODIZED BRASS',45,24
'Brand#15 ','SMALL BURNISHED NICKEL',19,24
'Brand#21 ','MEDIUM ANODIZED COPPER',3,24
'Brand#22 ','SMALL BRUSHED NICKEL',3,24
'Brand#22 ','SMALL BURNISHED BRASS',19,24
% total of 18314 rows written

```

qualification query 17

```

% select
% sum(l_extendedprice) / 7.0 as avg_yearly
% from
% lineitem,
% part
% where
% p_partkey = l_partkey
% and p_brand = 'Brand#23'
% and p_container = 'MED BOX'
% and l_quantity < (
% select
% 0.2 * avg(l_quantity)
% from
% lineitem
% where
% l_partkey = p_partkey
% );
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.22000 seconds - current
time 16:41:28
348406.054285713732
% total of 1 rows written

```

qualification query 18

```

% select top 100
% c_name,
% c_custkey,
% o_orderkey,
% o_orderdate,
% o_totalprice,
% sum(l_quantity)
% from
% customer,
% orders,
% lineitem
% where
% o_orderkey in (
% select
% l_orderkey
% from

```

```

% lineitem
% group by
% l_orderkey having
% sum(l_quantity) > 300
% )
% and c_custkey = o_custkey
% and o_orderkey = l_orderkey
% group by
% c_name,
% c_custkey,
% o_orderkey,
% o_orderdate,
% o_totalprice
% order by
% o_totalprice desc,
% o_orderdate;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.34000 seconds - current
time 16:41:29
'Customer#000128120',128120,4722021,'1994-04-07',54408
9.089999999881,323
'Customer#000144617',144617,3043270,'1997-02-12',53060
4.439999999994,317
'Customer#000013940',13940,2232932,'1997-04-13',522720
.61,304
'Customer#000066790',66790,2199712,'1996-09-30',515531
.82,327
'Customer#000046435',46435,4745607,'1997-07-03',508047
.99,309
'Customer#000015272',15272,3883783,'1993-07-28',500241
.33,302
'Customer#000146608',146608,3342468,'1994-06-12',49979
4.58,303
'Customer#000096103',96103,5984582,'1992-03-16',494398
.789999999994,312
'Customer#000024341',24341,1474818,'1992-11-15',491348
.26,302
'Customer#000137446',137446,5489475,'1997-05-23',48776
3.25,311
% total of 57 rows written

```

qualification query 19

```

% select
% sum(l_extendedprice* (1 - l_discount)) as revenue
% from
% lineitem,
% part
% where
% (
% p_partkey = l_partkey
% and p_brand = 'Brand#12'
% and p_container in ('SM CASE', 'SM BOX', 'SM PACK',
'SM PKG')
% and l_quantity >= 1 and l_quantity <= 1 + 10
% and p_size between 1 and 5
% and l_shipmode in ('AIR', 'AIR REG')
% and l_shipinstruct = 'DELIVER IN PERSON'
% )
% or
% (
% p_partkey = l_partkey
% and p_brand = 'Brand#23'
% and p_container in ('MED BAG', 'MED BOX', 'MED PKG',
'MED PACK')
% and l_quantity >= 10 and l_quantity <= 10 + 10
% and p_size between 1 and 10
% and l_shipmode in ('AIR', 'AIR REG')
% and l_shipinstruct = 'DELIVER IN PERSON'
% )
% or

```

```

% (
% p_partkey = l_partkey
% and p_brand = 'Brand#34'
% and p_container in ('LG CASE', 'LG BOX', 'LG PACK',
'LG PKG')
% and l_quantity >= 20 and l_quantity <= 20 + 10
% and p_size between 1 and 15
% and l_shipmode in ('AIR', 'AIR REG')
% and l_shipinstruct = 'DELIVER IN PERSON'
% );
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.35000 seconds - current
time 16:41:46
3083843.05780000031
% total of 1 rows written

```

qualification query 20

```

% select
% s_name,
% s_address
% from
% supplier,
% nation
% where
% s_suppkey in (
% select
% ps_suppkey
% from
% partsupp
% where
% ps_partkey in (
% select
% p_partkey
% from
% part
% where
% p_name like 'forest'
% )
% and ps_availqty > (
% select
% 0.5 * sum(l_quantity)
% from
% lineitem
% where
% l_partkey = ps_partkey
% and l_suppkey = ps_suppkey
% and l_shipdate >= '1994-01-01'
% and l_shipdate < dateadd(year,1,'1994-01-01')
% )
% )
% and s_nationkey = n_nationkey
% and n_name = 'CANADA'
% order by
% s_name;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.37000 seconds - current
time 16:41:51
'Supplier#000000020
', 'iybAE,RmTymrZVYaFZva2SH,j'
'Supplier#000000091
', 'YV45D7TkfdQanOOZ7q9QxkyGUapU1oOWU6q3'
'Supplier#000000197
', 'YC2Acon6kjY3zj3Fbxs2k4Vdf7X0cd2F'
'Supplier#000000226
', '83qOdu2EYRdPQAQhEtn
GRZED'
'Supplier#000000285

```

```
' , 'Br7elnnntlyxrw6ImgpJ7YdhFDjuBf '
'Supplier#000000378      ', 'FfbhyCxWvcPro08ltp9'
'Supplier#000000402
', 'i9Sw4DoyMhzhKXCH9By, AYSgmD'
'Supplier#000000530      ', '0qwCMwobKY
OcmLyfRXlagA8ukENJv, '
'Supplier#000000688      ', 'D
fw5ocppmZpYBBIPI718hCihLDZ5KhKX'
'Supplier#000000710      ', 'f19YpvOyb
QoYwjKC, oPycpGfieBAcwKJo'
% total of 204 rows written
```

qualification query 21

```
=====
% select top 100
% s_name,
% count(*) as numwait
% from
% supplier,
% lineitem l1,
% orders,
% nation
% where
% s_suppkey = l1.l_suppkey
% and o_orderkey = l1.l_orderkey
% and o_orderstatus = 'F'
% and l1.l_receiptdate > l1.l_commitdate
% and exists (
% select
% *
% from
% lineitem l2
% where
% l2.l_orderkey = l1.l_orderkey
% and l2.l_suppkey <> l1.l_suppkey
% )
% and not exists (
% select
% *
% from
% lineitem l3
% where
% l3.l_orderkey = l1.l_orderkey
% and l3.l_suppkey <> l1.l_suppkey
% and l3.l_receiptdate > l3.l_commitdate
% )
% and s_nationkey = n_nationkey
% and n_name = 'SAUDI ARABIA'
% group by
% s_name
% order by
% numwait desc,
% s_name;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.54000 seconds - current
time 16:41:53
'Supplier#000002829      ', 20
'Supplier#000005808      ', 18
'Supplier#000000262      ', 17
'Supplier#000000496      ', 17
'Supplier#000002160      ', 17
'Supplier#000002301      ', 17
'Supplier#000002540      ', 17
'Supplier#000003063      ', 17
'Supplier#000005178      ', 17
'Supplier#000008331      ', 17
% total of 100 rows written
=====
```

qualification query 22

```
% select
% c_ntrycode,
% count(*) as numcust,
% sum(c_acctbal) as totacctbal
% from
% (
% select
% substring(c_phone,1,2) as c_ntrycode,
% c_acctbal
% from
% customer
% where
% substring(c_phone,1,2) in
% ('13', '31', '23', '29', '30', '18', '17')
% and c_acctbal > (
% select
% avg(c_acctbal)
% from
% customer
% where
% c_acctbal > 0.00
% and substring(c_phone,1,2) in
% ('13', '31', '23', '29', '30', '18', '17')
% )
% and not exists (
% select
% *
% from
% orders
% where
% o_custkey = c_custkey
% )
% ) as custsale
% group by
% c_ntrycode
% order by
% c_ntrycode;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.18000 seconds - current
time 16:42:07
'13',888,6737713.98999999881
'17',861,6460573.72
'18',964,7236687.40000001431
'23',892,6701457.95000000954
'29',948,7158866.62999999642
'30',909,6808436.13000000119
'31',922,6806670.17999998569
% total of 7 rows written
Appendix D. Seed and Query Substitution Parameters
```

This Appendix contains Seed values and substitution parameters for each stream

Seed Values

```
stream0 115192426
stream1 115192427
stream2 115192428
stream3 115192429
stream4 115192430
stream5 115192431
stream6 115192432
```

Query Parameters

=====
stream0: 115192426
=====

14	1995-01-01								
2	9	BRASS	ASIA						
9	deep								
20	cyan	1997-01-01	INDIA						
6	1995-01-01	0.07	25						
17	Brand#45	MED JAR							
18	315								
8	UNITED STATES	AMERICA	SMALL PLATED						
BRASS									
21	VIETNAM								
13	pending	requests							
3	BUILDING	1995-03-18							
22	21	20	32	17	25	10			
14									
16	Brand#44	PROMO BRUSHED	25	16					
14	5	18	46	23	44				
4	1995-11-01								
11	ALGERIA	0.0000003333							
15	1994-01-01								
1	116								
10	1994-06-01								
19	Brand#32	Brand#52	Brand#15	3					
	19	27							
5	MIDDLE EAST	1995-01-01							
7	MOZAMBIQUE	UNITED STATES							
12	REG AIR	AIR	1995-01-01						

=====
stream1: 115192427
=====

21	JORDAN								
3	MACHINERY	1995-03-04							
18	313								
5	AFRICA	1995-01-01							
11	JORDAN	0.0000003333							
7	INDIA	MOZAMBIQUE							
6	1995-01-01	0.04	25						
20	orange	1995-01-01	UNITED KINGDOM						
17	Brand#42	MED CAN							
12	SHIP	RAIL	1995-01-01						
16	Brand#34	MEDIUM ANODIZED	29						
25	18	47	34	23	2	10			
15	1997-01-01								
13	pending	requests							
10	1993-03-01								
2	46	NICKEL	AFRICA						
8	MOZAMBIQUE	AFRICA	SMALL ANODIZED BRASS						
14	1995-01-01								
19	Brand#34	Brand#35	Brand#54	8					
	20	23							
9	coral								
22	31	24	22	14	11	18			
10									
1	63								
4	1993-08-01								

=====
stream2: 115192428
=====

6	1995-01-01	0.02	24						
17	Brand#44	JUMBO BOX							
14	1995-01-01								
16	Brand#14	ECONOMY PLATED	32	3					
22	11	47	45	13	16				
19	Brand#41	Brand#23	Brand#54	3					
	10	30							
10	1993-12-01								
9	brown								
2	34	COPPER	EUROPE						

15	1994-01-01								
8	GERMANY	EUROPE	STANDARD	POLISHED	BRASS				
5	AMERICA	1995-01-01							
22	23	18	29	16	19	34			
15									
12	FOB	RAIL	1995-01-01						
7	ALGERIA	GERMANY							
13	pending	accounts							
18	314								
1	71								
4	1996-02-01								
20	beige	1994-01-01	JORDAN						
3	BUILDING	1995-03-20							
11	ARGENTINA	0.0000003333							
21	ETHIOPIA								

=====
stream3: 115192429
=====

8	UNITED STATES	AMERICA	STANDARD						
BURNISHED	BRASS								
5	ASIA	1996-01-01							
4	1993-11-01								
6	1996-01-01	0.07	25						
17	Brand#41	JUMBO JAR							
7	PERU	UNITED STATES							
1	79								
18	312								
22	12	31	23	14	20	22			
19									
14	1996-01-01								
9	beige								
10	1994-09-01								
15	1997-01-01								
11	KENYA	0.0000003333							
20	lemon	1997-01-01	BRAZIL						
2	22	STEEL	AFRICA						
21	RUSSIA								
19	Brand#43	Brand#51	Brand#53	9					
	11	27							
13	pending	accounts							
16	Brand#44	STANDARD POLISHED	7						
24	11	50	38	14	41	8			
12	MAIL	RAIL	1995-01-01						
3	HOUSEHOLD	1995-03-06							

=====
stream4: 115192430
=====

5	EUROPE	1996-01-01							
21	KENYA								
14	1996-01-01								
19	Brand#45	Brand#34	Brand#42	4					
	12	23							
15	1995-01-01								
17	Brand#43	JUMBO CAN							
12	TRUCK	RAIL	1996-01-01						
6	1996-01-01	0.05	25						
4	1996-06-01								
9	white								
8	MOZAMBIQUE	AFRICA	PROMO BRUSHED	STEEL					
16	Brand#34	LARGE ANODIZED	42	4					
11	20	40	8	17	13				
11	BRAZIL	0.0000003333							
2	10	BRASS	EUROPE						
10	1993-07-01								
18	314								
1	87								
13	pending	accounts							
7	INDONESIA	MOZAMBIQUE							
22	27	22	13	18	16	12			
10									
3	BUILDING	1995-03-22							
20	snow	1995-01-01	PERU						

=====
stream5: 115192431
 =====

21	germany						
15	1997-01-01						
4	1994-03-01						
6	1996-01-01	0.02	24				
7	argentina india						
16	brand#14 promo burnished	42		30	34	45	
13	8	32	17				
19	brand#52 brand#22 brand#41	9		13	30		
18	315						
14	1996-01-01						
22	12	14	19	31	10	16	33
11	kenya 0.0000003333						
13	pending accounts						
3	household 1995-03-08						
1	95						
2	48 nickel america						
5	middle east 1996-01-01						
8	india asia promo plated steel						
20	forest 1994-01-01 germany						
12	rail truck 1996-01-01						
17	brand#45 wrap box						
10	1994-04-01						
9	tan						

=====
stream6: 115192432
 =====

10	1995-01-01						
3	AUTOMOBILE 1995-03-24						
15	1995-01-01						
13	pending deposits						
6	1996-01-01	0.07	24				
8	ALGERIA AFRICA PROMO ANODIZED STEEL						
9	sky						
7	CHINA ALGERIA						
4	1996-10-01						
11	BRAZIL 0.0000003333						
22	23	30	10	19	20	15	25
18	313						
12	REG AIR TRUCK 1996-01-01						
1	103						
5	AFRICA 1996-01-01						
16	Brand#44 SMALL POLISHED	46		23	48	12	
33	26	20	22				
2	35 TIN EUROPE						
14	1996-01-01						
19	Brand#54 Brand#55 Brand#45	4		14	26		
20	powder 1997-01-01 RUSSIA						
17	Brand#42 WRAP PACK						
21	UNITED STATES						

Appendix E. Implementation-Specific Layer/Driver Code

=====

do_test

```
dbisqlc -c "DSN=tpch" -q load_lineitem.sql >
load_lineitem.out &
loadlpid=$!
dbisqlc -c "DSN=tpch" -q load_region.sql
dbisqlc -c "DSN=tpch" -q load_nation.sql
dbisqlc -c "DSN=tpch" -q load_customer.sql
dbisqlc -c "DSN=tpch" -q load_part.sql
dbisqlc -c "DSN=tpch" -q load_supplier.sql
dbisqlc -c "DSN=tpch" -q load_partsupp.sql
dbisqlc -c "DSN=tpch" -q load_orders.sql
loadopid=$!
wait $loadopid
wait $loadlpid
end_load.out
echo " "
seed=`date '+%m%d%H%M%S'`;
echo $seed;
./gen_streams_new.ksh $seed 300
dbisqlc -c "DSN=tpch" -q dbtables-syb.sql >
rdtablest.out
dbisqlc -c "DSN=tpch" -q dew_cat1.sql >
dew_cat1_start.out
dbisqlc -c "DSN=tpch" -q dew_cat2.sql >
dew_cat2_start.out
dbisqlc -c "DSN=tpch" -q dew_cat3.sql >
dew_cat3_start.out
dbisqlc -c "DSN=tpch" -q check_options.sql >
check_options_start.out
dbisqlc -c "DSN=tpch" -q rfl1.sql > rfl1.out
dbisqlc -c "DSN=tpch" -q stream0.sql > stream0.out
dbisqlc -c "DSN=tpch" -q rf2.sql > rf2.out
dbisqlc -c "DSN=tpch" -q stream1.sql > stream1.out &
dbisqlc -c "DSN=tpch" -q stream2.sql > stream2.out &
dbisqlc -c "DSN=tpch" -q stream3.sql > stream3.out &
dbisqlc -c "DSN=tpch" -q stream4.sql > stream4.out &
dbisqlc -c "DSN=tpch" -q stream5.sql > stream5.out &
dbisqlc -c "DSN=tpch" -q stream6.sql > stream6.out &
dbisqlc -c "DSN=tpch" -q update_throughput6.sql >
update_throughput.out &
wait

#move *.out files to audit naming standard in
/sybase_stage/results"
((i=0))
while ((i<=nqs)) # move the streamX.out files to
audit naming standard
do
  mv stream${i}.out /sybase_stage/results/mls0${
i}q.out
  ((i=i+1))
done
mv update_power.out /sybase_stage/results/mls00rf.out
mv update_throughput.out
/sybase_stage/results/mls01rf.out

dbisqlc -c "DSN=tpch" -q rfl1.sql > rfl1.out
dbisqlc -c "DSN=tpch" -q stream0.sql > stream0.out
dbisqlc -c "DSN=tpch" -q rf2.sql > rf2.out
dbisqlc -c "DSN=tpch" -q stream1.sql > stream1.out &
dbisqlc -c "DSN=tpch" -q stream2.sql > stream2.out &
dbisqlc -c "DSN=tpch" -q stream3.sql > stream3.out &
dbisqlc -c "DSN=tpch" -q stream4.sql > stream4.out &
dbisqlc -c "DSN=tpch" -q stream5.sql > stream5.out &
dbisqlc -c "DSN=tpch" -q stream6.sql > stream6.out &
```

```
dbisqlc -c "DSN=tpch" -q update_throughput6.sql >
update_throughput.out &
wait
dbisqlc -q -c "DSN=tpch" -q dbtables-syb.sql >
rdtablest_end.out
dbisqlc -q -c "DSN=tpch" -q dew_cat1.sql >
dew_cat1_end.out
dbisqlc -q -c "DSN=tpch" -q dew_cat2.sql >
dew_cat2_end.out
dbisqlc -q -c "DSN=tpch" -q dew_cat3.sql >
dew_cat3_end.out
dbisqlc -q -c "DSN=tpch" check_options.sql >
check_options_end.out
```

Appendix F. Misc database scripts

The dbtables-syb.sql script was run to validate the correctness of the database after the database load. Three other scripts were used to extract basic information about tables and indexes from the database dew_cat1.sql, dew_cat2.sql, dew_cat3.sql.

Auditor Scripts

dbtables-syb.sql

```
=====
-- FILENAME
-- DBTABLES.SQL
-- DESCRIPTION
-- CHECK ROW COUNT AND ROW STRUCTURE/CONTENT FOR
EACH TABLE
-- IN THE TPC-H DATABASE.
--
-- =====
--
-- GET TIMESTAMP
SELECT 'START TIME', CONVERT(CHAR(30), GETDATE(),
120);
go
-- =====
-- TABLE: LINEITEM
-- =====
SELECT COUNT(*) FROM LINEITEM;
go
SELECT * FROM LINEITEM
WHERE L_ORDERKEY IN
( 4, 26598, 148577, 387431, 56704, 517442,
600000)
AND L_LINENUMBER = 1
ORDER BY L_ORDERKEY;
go
-- =====
-- TABLE: ORDERS
-- =====
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM ORDERS;
go
SELECT * FROM ORDERS
WHERE O_ORDERKEY IN ( 7, 44065, 287590, 411111,
483876, 599942 )
ORDER BY O_ORDERKEY;
go
-- =====
-- TABLE: PART
-- =====
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM PART;
go
SELECT * FROM PART
WHERE P_PARTKEY IN (1,984,8743,9028,13876,17899,20000)
ORDER BY P_PARTKEY;
go
-- =====
-- TABLE: PARTSUPP
-- =====
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM PARTSUPP;
go
SELECT* FROM PARTSUPP
```

```
WHERE PS_PARTKEY = 3398
AND PS_SUPPKY = (SELECT MIN(PS_SUPPKY)
FROM PARTSUPP WHERE PS_PARTKEY = 3398);
go
SELECT* FROM PARTSUPP
WHERE PS_PARTKEY =15873
AND PS_SUPPKY = (SELECT MIN(PS_SUPPKY)
FROM PARTSUPP WHERE PS_PARTKEY = 15873);
go
SELECT* FROM PARTSUPP
WHERE PS_PARTKEY = 11394
AND PS_SUPPKY = (SELECT MIN(PS_SUPPKY)
FROM PARTSUPP WHERE PS_PARTKEY = 11394);
go
SELECT* FROM PARTSUPP
WHERE PS_PARTKEY = 6743
AND PS_SUPPKY = (SELECT MIN(PS_SUPPKY)
FROM PARTSUPP WHERE PS_PARTKEY = 6743);
go
SELECT* FROM PARTSUPP
WHERE PS_PARTKEY = 19763
AND PS_SUPPKY = (SELECT MIN(PS_SUPPKY) FROM
PARTSUPP WHERE PS_PARTKEY =19763);
go
-- =====
-- TABLE: SUPPLIER
-- =====
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM SUPPLIER;
go
SELECT * FROM SUPPLIER
WHERE S_SUPPKY IN (83,265,492,784,901,1000)
ORDER BY S_SUPPKY;
go
-- =====
-- TABLE: CUSTOMER
-- =====
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM CUSTOMER;
go
SELECT * FROM CUSTOMER
WHERE C_CUSTKEY IN (832,2653,4924,7845,92016,108070)
ORDER BY C_CUSTKEY;
go
-- =====
-- TABLE: NATION & REGION
-- =====
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT * FROM REGION;
go
SELECT COUNT(*) FROM NATION;
go
SELECT * FROM NATION
WHERE N_NATIONKEY IN (3,10,14,20)
ORDER BY N_NATIONKEY;
go
-- =====
-- CHECK KEY VALUES
-- =====
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
if exists (select name from sysobjects where
name='MINMAX')
drop table MINMAX
go
CREATE TABLE MINMAX
(TNAME CHAR(15),
KEYMIN INTEGER,
KEYMAX INTEGER);
go
INSERT INTO MINMAX
```

```

SELECT 'LINEITEM_ORD',MIN(L_ORDERKEY),MAX(L_ORDERKEY)
FROM LINEITEM;
go
INSERT INTO MINMAX
SELECT
'LINEITEM_NBR',MIN(L_LINENUMBER),MAX(L_LINENUMBER)
FROM LINEITEM;
go
INSERT INTO MINMAX
SELECT 'ORDERS',MIN(O_ORDERKEY),MAX(O_ORDERKEY)
FROM ORDERS;
go
INSERT INTO MINMAX
SELECT 'CUSTOMER',MIN(C_CUSTKEY),MAX(C_CUSTKEY)
FROM CUSTOMER;
go
INSERT INTO MINMAX
SELECT 'PART',MIN(P_PARTKEY),MAX(P_PARTKEY)
FROM PART;
go
INSERT INTO MINMAX
SELECT 'SUPPLIER',MIN(S_SUPPKEY),MAX(S_SUPPKEY)
FROM SUPPLIER;
go
INSERT INTO MINMAX
SELECT 'PARTSUPP_PART',MIN(PS_PARTKEY),MAX(PS_PARTKEY)
FROM PARTSUPP;
go
INSERT INTO MINMAX
SELECT 'PARTSUPP_SUPP',MIN(PS_SUPPKEY),MAX(PS_SUPPKEY)
FROM PARTSUPP;
go
INSERT INTO MINMAX
SELECT 'NATION',MIN(N_NATIONKEY),MAX(N_NATIONKEY)
FROM NATION;
go
INSERT INTO MINMAX
SELECT 'REGION',MIN(R_REGIONKEY),MAX(R_REGIONKEY)
FROM REGION;
go
SELECT * FROM MINMAX;
go
if exists (select name from sysobjects where
name='MINMAX')
drop table MINMAX
go
SELECT 'END TIME', CONVERT(CHAR(30), GETDATE(), 120);
go

```

dew_cat1.sql

```

SELECT st.table_name,
       st.table_type,
       su.user_name,
       st.server_type
  from SYS.SYSTABLE st, SYS.SYSUSERPERMS su
 where creator = user_id
 order by 4,1,3;

```

dew_cat2.sql

```

select T.table_name      ,
       T.table_type      ,
       C.column_name     ,
       C.column_id
From   SYS.SYSTABLE T,
       SYS.SYSCOLUMN C,
       SYS.SYSDOMAIN D,
       SYS.SYSUSERPERMS SU
where  T.creator = SU.user_id
       and T.table_id = C.table_id
       and C.domain_id = D.domain_id
 order by 1,2;

```

dew_cat3.sql

```

SELECT index_name,T.table_name ,
       column_name ,
       index_type
  from  SYS.SYSTABLE T,
       SYS.SYSCOLUMN C,
       SYS.SYSINDEX I,
       SYS.SYSUSERPERMS UP,
       SYS.SYSFILE F,
       SYS.SYSIXCOL IC
 where T.table_id = C.table_id
       and C.table_id = I.table_id
       and T.file_id = F.file_id
       and I.table_id = IC.table_id
       AND I.index_id = IC.index_id
       AND IC.column_id = C.column_id
       and T.creator = UP.user_id;

```

Appendix G. Pricing information

Sybase pricing:

Company Sun Microsystems
 Contact Richard Gostanian
 Phone 781-442-3063
 Fax
 Address 1 Network Drive, Burlington MA 01803

Quotation for Software and Support

SYBASE Sales Rep: Hollie Nash
 Phone: 972-687-6412
 Fax: 972-687-6409

CBSS#

	Catalogue Number	Product Description	License Type	Machine	P/S	List Price Per Unit	Quantity	Discount	Extended Price	Extended Support Fees
1	11467	Sybase IQ Single App Svr, per cpu core	CP	Sun	P	2,595	4		10,380.00	3 YEARS
3	98477	3 yr support Single App Svr, per cpu core				1,557	4			6,228.00
4		Discount 5% (if total order > \$25,000)						.00		
5										
6										
7										
8										
9										
10										
11										
12										

Quote Date:

3/20/08

Valid thru:

Total

16,608.00

Licence + 3 year support

Payment terms : Net 30 Days

5400 LBJ Freeway, Suite 1500, Dallas, TX 75240



Sales Quotation

Quote Number: T-US-1241805-A

Quote Date: 5/13/08

Customer : RICHARD GOSTANIAN
 SUN MICROSYSTEMS INC
 1 NETWORK DRIVE
 BURLINGTON MA 01803
 Tel / Fax : 7814423063 /

Sun : Holli Babb
 Sun Microsystems, Inc.
 1617 Southwood Drive
 Nashua NH 03063
 Tel / Fax : 603-589-0548/781-394-0478

We are pleased to quote as follows:

Validity Period
60 Days

Credit Terms
Net 30 Days

Shipping Terms
Origin

Item	Product Number	Description	Qty	Unit List Price	Disc	Unit Net Price	Extended Net Price
Sun Proprietary & Confidential, Internal Use Only							
SunFire X4240 Server							
1	Config ID 6328737	Configuration: B14-FE2-EB-8G-JL8	1	\$12,805.11	N/A	\$11,524.60	\$11,524.60
1.1	B14-FE2-EB-8G-JL8	Sun Fire X4240 x64 Server: 2x AMD Opteron Model 2224 SE (3.2GHz/1MB) dual core processor, 4x 2GB DDR2-667 memory, No HDD, No DVD-RW, 2x PSU, Service processor, 4x 10/100/1000 Ethernet ports, 4x USB 2.0 ports, 1x 16-lane PCIe slot, 4x 8-lane PCIe slots, 1x 4-lane PCIe slot, no Power Cord, order Geo-specific X-option. Standard Configuration. RoHS-5.	1	\$5,745.00	10.00%	\$5,170.50	\$5,170.50
1.2	X311L	Localized Power Cord Kit North American/Asian This Product is Hazard Class Y, RoHS compliant.	2	N/C	N/A	N/C	N/C

YOU MUST READ THE FOLLOWING: THIS SUN QUOTATION AND ANY ORDER YOU SUBMIT FOR PRODUCTS OR SERVICES IS SUBJECT TO: (1) THE TERMS OF ANY EXISTING SALES AGREEMENT YOU HAVE WITH SUN GOVERNING THAT PRODUCT OR SERVICE, OR, IF NONE, BY SUN'S SALES TERMS FOUND AT <http://www.sun.com/sales/salesterms>, THE GENERAL TERMS OF WHICH ARE EITHER ATTACHED OR ON THE REVERSE SIDE HEREOF, AND (2) APPLICABLE SUN SERVICE LISTINGS AND STATEMENTS OF WORK FOUND AT <http://www.sun.com/service/servicelist> [(1) AND (2) COLLECTIVELY BEING CALLED "SUN SALES TERMS."]

ALL ORDERS MUST REFERENCE EITHER YOUR SALES AGREEMENT NUMBER OR THIS SALES QUOTATION AND BE IN CONFORMANCE WITH SUN SALES TERMS. ORDERS ARE SUBJECT TO ACCEPTANCE BY SUN EITHER THROUGH ISSUANCE OF AN ORDER ACKNOWLEDGEMENT OR DELIVERY OF THE PRODUCTS OR SERVICES. THIS QUOTATION REMAINS FIRM FOR THE PERIOD LISTED ABOVE, EXCEPT THAT SUN MAY MODIFY THIS SALES QUOTATION IF THERE IS A TYPOGRAPHICAL ERROR OR THE AVAILABILITY OF PRODUCTS, SERVICES, OR CREDIT CHANGE. SUN EQUIPMENT, OR PARTS OR COMPONENTS OF SUN EQUIPMENT, MAY BE NEW OR USED, REGARDLESS, SUN WARRANTY TERMS APPLY.





Sales Quotation

Quote Number: T-US-1241805-A

Quote Date: 5/13/08

Item	Product Number	Description	Qty	Unit List Price	Disc	Unit Net Price	Extended Net Price
1.3	IWU-B14-3G	This part number corresponds to the following 11i Service item: GOLD-SYS-SVC Sun Fire X4240. Warranty Upgrade to Gold Support for 3 Year.	1	\$2,157.11	10.00%	\$1,941.40	\$1,941.40
1.4	XRA-SS2CF-73G10K	73GB 10K RPM 2.5" SAS disk drive with Marlin bracket. RoHS-6. X-option.	16	\$259.00	10.00%	\$233.10	\$3,729.60
1.5	SG-XPCIE8SAS-I-Z	Sun StorageTek (TM) 8-Port internal SAS PCI-Express LSI 3081E Host Bus Adapter with RAID 0, 1, 1E support. RoHS-6. X-option	1	\$249.00	10.00%	\$224.10	\$224.10
1.6	X6321A	4 GB Memory kit DDR2-667 Registered ECC DIMMs (2x 2GB) for Sun Fire X4140/X4240/X4440 x64 servers. RoHS-5. X-Option.	2	\$255.00	10.00%	\$229.50	\$459.00

List Price Total:	\$12,805.11
-------------------	-------------

Total:	\$11,524.60
--------	-------------

WE CAN HELP YOU SAVE WHEN YOU FINANCE YOUR TECHNOLOGY SOLUTION THROUGH SUN MICROSYSTEMS GLOBAL FINANCIAL SERVICES (SMGFS). CHOOSE FROM A WIDE RANGE OF FLEXIBLE ,LOW RATE FINANCING AND LEASING PLANS. CONTACT US TODAY AT SMGFS_Quote_ AMER @ sun. com

