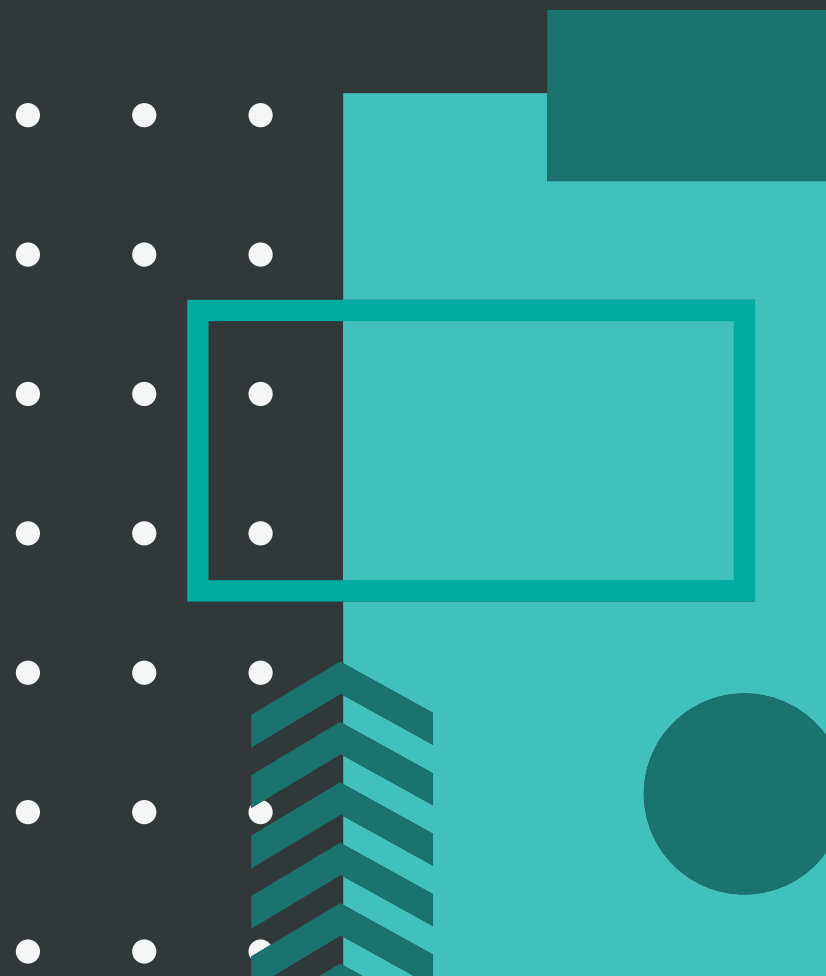
Performance Benchmark Report

AtScale Cloud Data Warehouse Benchmark: Amazon Redshift

Quantified Results for Query Performance, Compute Cost, Complexity and Concurrency



ATSCALE

TABLE OF CONTENTS

Background 1

Benchmarking Methodology 2

- Benchmark Dataset 2
- Benchmark Queries 3
- Test Harness 3
- Configuration Tested 4
- Query Performance Test Methodology 5
- Concurrency Test Methodology 5
- Compute Cost Calculations 5

Summary Results 6

- Query Performance Test Results 7

Concurrency Test Results 8

- Compute Cost Test Results 9
- Complexity Test Results 10

Conclusion 13

Background

The enterprise has entered into a new era of data warehousing. Driven by the increasing popularity of the public cloud, new data warehouse technologies are making inroads into the traditional on-premise data warehouse market. By offering customers the power of a relational, scale-out data platform without the overhead of managing it, cloud data warehouses promise to make more data available at a lower cost with fewer data management headaches.

In this inaugural cloud data warehouse benchmark analysis, we set out to quantify vendor benefits while comparing and contrasting the most popular offerings.

Using the standard TPC-DS (10TB) benchmarking framework, we set out to test the performance and scalability boundaries of the various options. In addition, we also examined the operational cost dimension and we challenged the traditional data modeling techniques by testing an alternative to raw TPC-DS SQL.

In summary, we focused our testing on the following 4 areas:

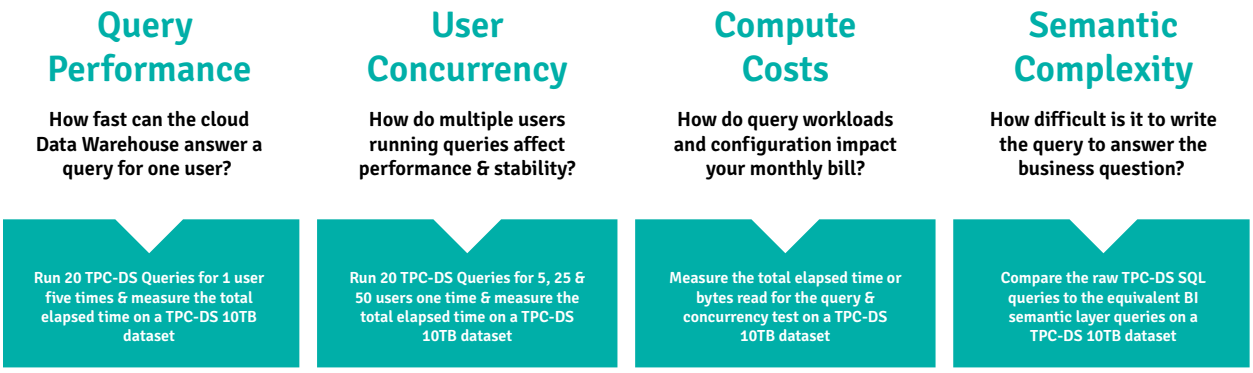


Figure 1: Benchmark Testing Topics

The results of this study use the above metrics to quantify Redshift's results in these four areas.

Benchmarking Methodology

Benchmark Dataset

We used the TPC-DS benchmark v2.11.0 from the Transaction Processing Council (TPC) for our tests. We chose the 10TB (scale factor 10,000) version for the test given that this version's largest fact table (store_sales) at 28+ billion rows and the largest dimension (customer) at 65 million rows is a significant scale challenge for most data platforms. In addition, the TPC-DS benchmark is ubiquitous amongst the database warehouse vendors and we felt it represented a reasonable real-life analytics schema and set of queries.

Table Name	Row Size	Row Count
call_center	305	54
catalog_page	139	40,000
catalog_returns	166	1,440,033,112
catalog_sales	226	14,399,964,710
customer	132	65,000,000
customer_address	110	32,500,000
customer_demographics	42	1,920,800
date_dim	141	73,049
household_demographics	21	7,200
income_band	16	20
inventory	16	1,311,525,000
item	281	402,000
promotions	124	2,000
reason	38	70
ship_mode	56	20
store	263	1,500
store_returns	134	2,879,970,104
store_sales	164	28,799,983,563
time_dim	59	86,400
warehouse	117	25
web_page	96	4,002
web_returns	162	720,020,485
web_sales	226	7,199,963,324
web_site	292	78

THE TPC-DS 10TB DATASET HAS:

1. Multiple fact tables
2. Large fact tables
3. Large dimensions

Figure 2: TPC-DS 10TB Table SIzes

.

Benchmark Queries

We selected a representative set of 20 queries from the 99 TPC-DS queries set to keep the run time and costs of running the benchmarks within reason without having to downsize data size. The queries were chosen in no particular order and were selected to eliminate redundancy and to ensure the usage of most tables. It was an imperative to benchmark the cloud data warehouse vendors with the largest data we could afford and test to reveal real-life differences in the respective offerings.

The following queries were used for the test:

TPC-DS Query #	Description
2	Report the ratios of weekly web and catalog sales increases from one year to the next year for each week. That is, compute the increase of Monday, Tuesday, ... Sunday sales from one year to the following.
7	Compute the average quantity, list price, discount, and sales price for promotional items sold in stores where the promotion is not offered by mail or a special event. Restrict the results to a specific gender, marital and educational status.
13	Calculate the average sales quantity, average sales price, average wholesale cost, total wholesale cost for store sales of different customer types (e.g., based on marital status, education status) including their household demographics, sales price and different combinations of state and sales profit for a given year.
15	Report the total catalog sales for customers in selected geographical regions or who made large purchases for a given year and quarter.
26	Computes the average quantity, list price, discount, sales price for promotional items sold through the catalog channel where the promotion was not offered by mail or in an event for given gender, marital status and educational status.
31	List counties where the percentage growth in web sales is consistently higher compared to the percentage growth in store sales in the first three consecutive quarters for a given year.
33	What is the monthly sales figure based on extended price for a specific month in a specific year, for manufacturers in a specific category in a given time zone. Group sales by manufacturer identifier and sort output by sales amount, by channel, and give Total sales.
42	For each item and a specific year and month calculate the sum of the extended sales price of store transactions.
48	Calculate the total sales by different types of customers (e.g., based on marital status, education status), sales price and different combinations of state and sales profit.
50	For each store count the number of items in a specified month that were returned after 30, 60, 90, 120 and more than 120 days from the day of purchase.

TPC-DS Query #	Description
52	Report the total of extended sales price for all items of a specific brand in a specific year and month.
53	Find the ID, quarterly sales and yearly sales of those manufacturers who produce items with specific characteristics and whose average monthly sales are larger than 10% of their monthly sales.
55	For a given year, month and store manager calculate the total store sales of any combination all brands.
56	Compute the monthly sales amount for a specific month in a specific year, for items with three specific colors across all sales channels. Only consider sales of customers residing in a specific time zone. Group sales by item and sort output by sales amount.
60	What is the monthly sales amount for a specific month in a specific year, for items in a specific category, purchased by customers residing in a specific time zone. Group sales by item and sort output by sales amount.
61	Find the ratio of items sold with and without promotions in a given month and year. Only items in certain categories sold to customers living in a specific time zone are considered.
71	Select the top revenue generating products, sold during breakfast or dinner time for one month managed by a given manager across all three sales channels.
88	How many items do we sell between pecific times of a day in certain stores to customers with one dependent count and 2 or less vehicles registered or 2 dependents with 4 or fewer vehicles registered or 3 dependents and five or less vehicles registered. In one row break the counts into sells from 8:30 to 9, 9 to 9:30, 9:30 to 10 ... 12 to 12:30
96	Compute a count of sales from a named store to customers with a given number of dependents made in a specified half hour period of the day.
98	Report on items sold in a given 30 day period, belonging to the specified category.

Figure 3: TPC-DS Test Queries

Test Harness

To ensure consistency for concurrency tests, we ran queries using v5.1.1 of Apache JMeter. The instructions, documentation, utility scripts, results and JMeter JMX files can be found in our GitHub repository and are available upon request.

We designed the JMeter test suites to run the following configurations:

- 1 concurrent user, 5 loops
- 5 concurrent users, 1 loop
- 25 concurrent users, 1 loop
- 50 concurrent users, 1 loop
- 100 concurrent users, 1 loop (tested with AtScale only)

We originally planned to run the 100 thread user concurrency test for Redshift but found challenges at the 100 concurrent user level. Running 100 concurrent users without the help of a semantic layer like AtScale proved to be a scaling challenge that resulted in extended run times as a result of query queuing. Redshift has an option for managing user concurrency automatically by spinning up additional virtual data warehouses. We did not test this feature because we wanted to keep our resource level fixed for apples to apples comparisons. As a result, we only ran the 100 thread tests with AtScale.

Configuration Tested

The following Redshift configuration was used for the test:

Vendor	Configuration	Compute Cost per Hour[1]
Redshift	dc2.8xlarge (6 nodes)	$28.80[2]

Figure 4: Data Warehouse Configuration

For the test, we used Redshift's "out of the box" configuration and standard edition. We did not manually tune any of the TPC-DS queries.

Footnotes:

1. Storage cost wasn't factored in (only compute cost)
2. On demand price: $4.80 per node

Query Performance Test Methodology

To test raw query performance, we ran the 20 TPC-DS queries with one concurrent user five times and calculated the total elapsed time to finish the queries. The elapsed time is simply the difference between the start and end time of the test as reported by JMeter. We disabled Redshift's query caching for this test.

Concurrency Test Methodology

To test how each data warehouse performs with different levels of user concurrency, we ran each of the 20 TPC-DS queries with 1, 5, 25 and 50 concurrent users using JMeter. We added a 750ms sleep between each query start and using a single connection pool that was sized according to the number of threads for the test. We used 1 loop (iteration) for the 5, 25, and 50 thread test and 5 loops for the 1 thread test. The elapsed time is simply the difference between the start and end time of each thread test as reported by JMeter. We disabled Redshift's query caching for this test.

Compute Cost Calculations

Redshift charges per hour for your choice of hardware configuration (disk and compute) and number of cluster nodes.

We calculated the compute costs by multiplying the total end-to-end run time as reported by JMeter for the concurrency test by the cluster compute cost per hour like so:

ConcurrencyRunTimeMinutes / 60 * ComputeCostPerHour

We explicitly excluded storage costs from our calculations. We found that storage cost was nominal across all platforms and given that it's a fixed cost, it was not subject to variation in our testing scenarios.

Summary Results

We also ran the same 20 TPC-DS queries through the AtScale platform for Redshift. AtScale's Acceleration Structures showed major benefits in accelerating query performance, improving user concurrency and reducing compute costs. The illustration below shows the extent of the benefits AtScale provides on top of the Redshift data warehouse:

Test	Improvement Factor with AtScale
	Redshift
Query Performance[1]	12.5x Faster
User Concurrency[2]	61x Faster
Compute Cost[3]	2.6x Cheaper
Complexity[4]	76% less complex SQL queries

Figure 5: Improvements with AtScale

Footnotes:

1. Elapsed time for executing 1 query five times
2. Elapsed time executing 1 (x5), 5, 25, 50 queries
3. Compute costs for cluster time for user concurrency test
4. Complexity score for SQL queries for number of:
   functions, operations, tables, objects & subqueries (AtScale = 258, TPC-DS = 1,057)

Query Performance Test Results

For the query performance test, we ran our 20 TPC-DS queries 5 times each using JMeter with a single thread. Even at a single concurrent user, we saw orders of magnitude improvement using AtScale on the Redshift data warehouse in this test.

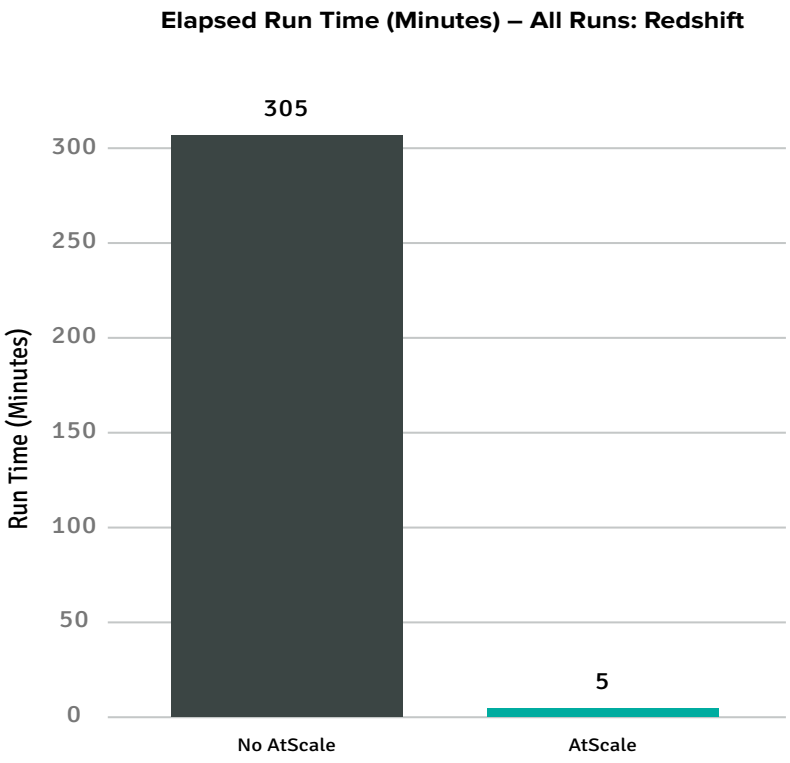


Figure 6: Elapsed Run Time for 1 Thread

Concurrency Test Results

For the user concurrency test, we ran consecutive JMeter suites configured to execute 1, 5, 25, and 50 queries at the same time to simulate user concurrency. Each test ran 1 iteration with the exception of the 1 thread test which ran 5 iterations sequentially.

In this test, we saw some real impact in query performance under additional user concurrency load. To be fair, Redshift offers the option for their enterprise edition for automated concurrency scaling. This option dynamically adds more cluster resources to handle concurrency bottlenecks. We chose not to enable this option in order to quantify performance for a fixed level of resource.

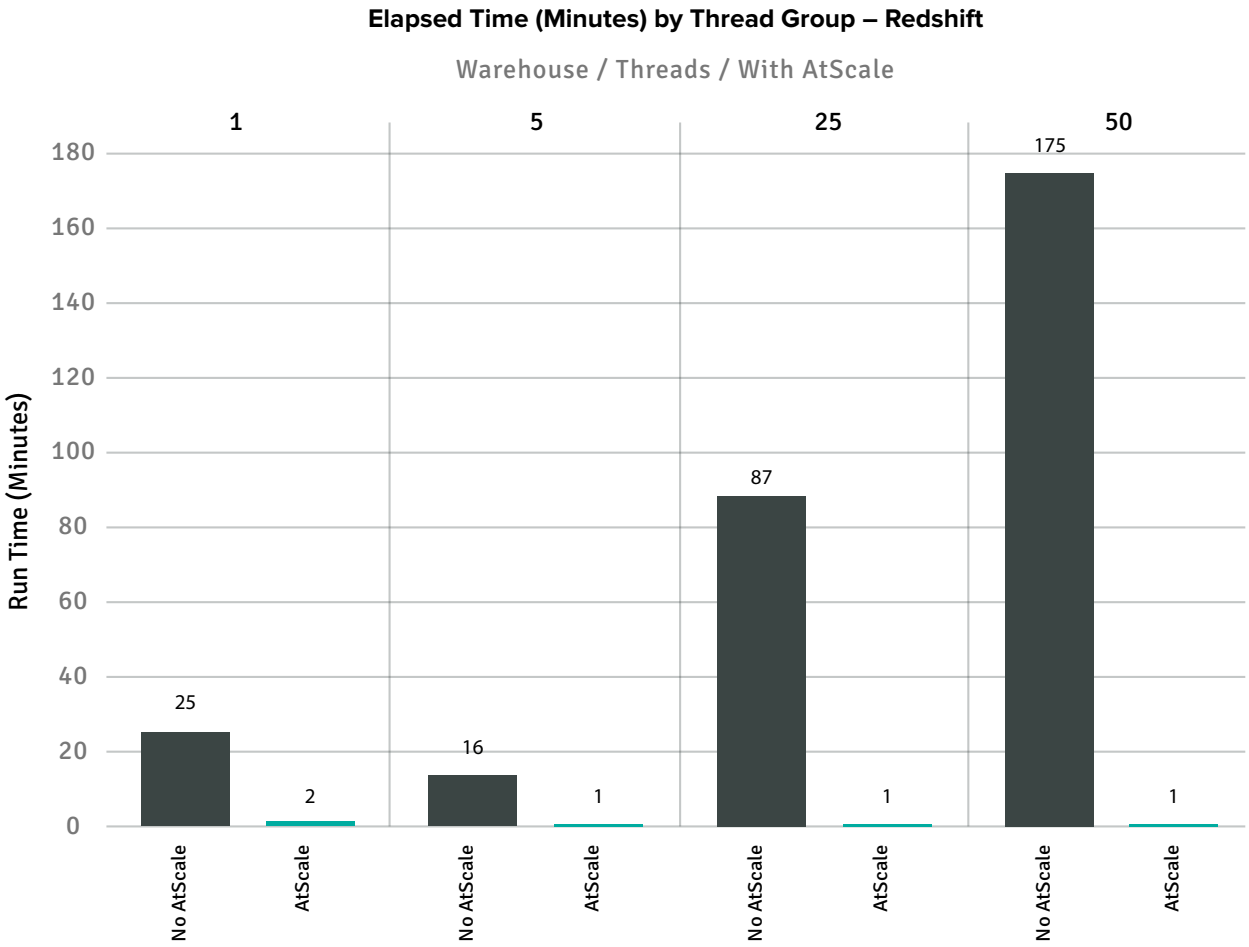


Figure 7: Elapsed Run Time for 1, 5, 25 & 50 Threads

Compute Cost Test Results

You will also see the value that AtScale can bring to cost predictability. By minimizing the amount of data scanned, AtScale takes less time to run queries, with fewer resources used, which means more users can run queries at the same time (higher concurrency) without additional hardware or resources.

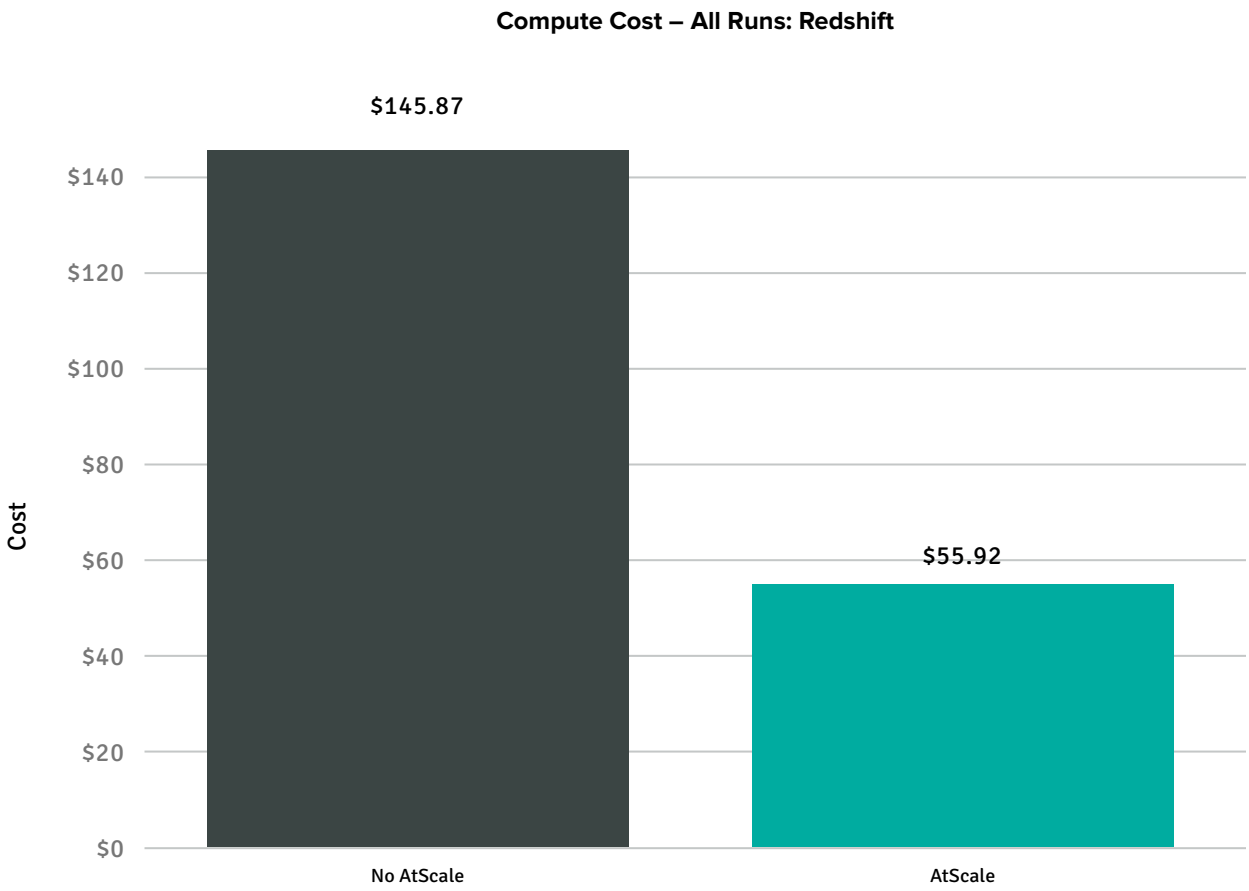


Figure 8: Compute Costs for All Thread Groups

Complexity Test Results

The TPC-DS benchmark provides a good illustration of just how hard it can be to write SQL to answer a simple business question. Translating tables and star schemas into business logic is not an easy task. With today's BI tools, our business users are spending more and more time dealing with data engineering tasks rather than getting answers to their business questions.

For example, with query #60 of the TPC-DS benchmark, the business question is fairly straightforward but the SQL to express it is not.

BUSINESS QUESTION

What is the monthly sales amount for a specific month in a specific year, for items in a specific category, purchased by customers residing in a specific time zone?

SQL TO ANSWER BUSINESS QUESTION:

TPC-DS Raw

```
with ss as (
 select
   i_item_id,sum(ss_ext_sales_price) total_sales
 from
   store_sales,
   date_dim,
      customer_address,
      item
 where
   i_item_id in (select
   i_item_id
from
   item
where i_category in ('Jewelry'))
   and   ss_item_sk              = i_item_sk
   and   ss_sold_date_sk         = d_date_sk
   and   d_year                  = 1999
   and   d_moy                   = 9
   and   ss_addr_sk              = ca_address_sk
   and   ca_gmt_offset           = -6
   group by i_item_id),
   cs as (
 select
   i_item_id,sum(cs_ext_sales_price) total_sales
 from
   catalog_sales,
   date_dim,
   customer_address,
   item
 where
   i_item_id in (select
   i_item_id
from
   item
where i_category in ('Jewelry'))
   and   cs_item_sk              = i_item_sk
   and   cs_sold_date_sk         = d_date_sk
   and   d_year                  = 1999
   and   d_moy                   = 9
   and   cs_bill_addr_sk         = ca_address_sk
   and   ca_gmt_offset           = -6
   group by i_item_id),
   ws as (
 select
   i_item_id,sum(ws_ext_sales_price) total_sales
 from
   web_sales,
   date_dim,
   customer_address,
   item
 where
   i_item_id in (select
   i_item_id
from
   item
   ...
```

26,640 bytes

Figure 9: TPC-DS Raw SQL to answer question

As you can see, it's not at all obvious what the query is doing and obviously there's a lot of repetition which makes it very prone to error.

In response to this challenge, for this benchmark study, we defined an AtScale virtual cube that drastically simplifies user queries by translating the raw tables and schema into a business semantic layer. The following screenshot is the TPC-DS model expressed in AtScale Design Center:

Figure 10: AtScale TPC-DS Data Model

Instead of writing complex SQL or engineering data models in the BI tool, this business question was easily answered with Tableau on AtScale as you can see below:

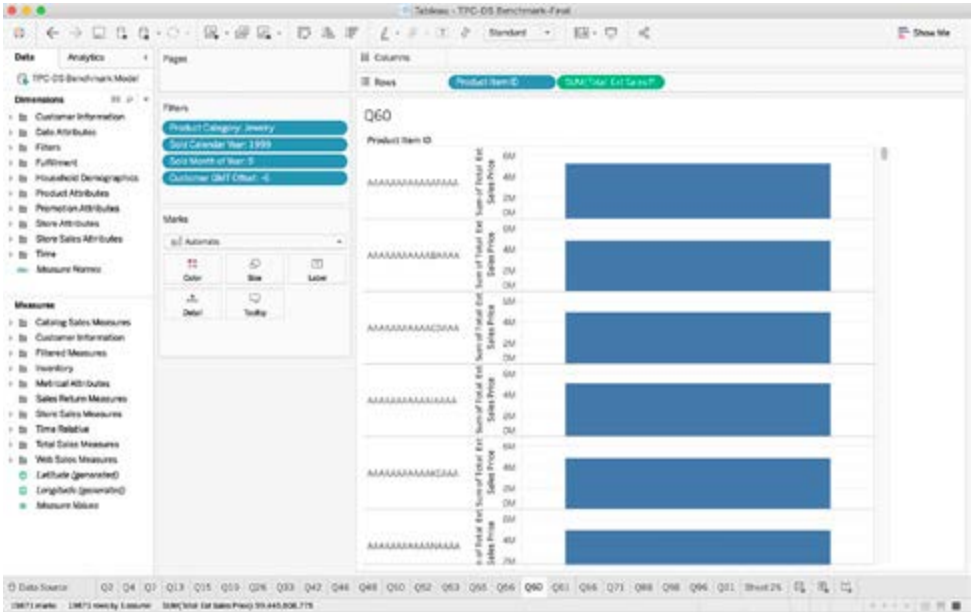


Figure 11: Tableau on AtScale TPC-DS Model for Query #60

The visualization above for TPC-DS query #60 generated the following SQL against AtScale:

AtScale SQL

```
SELECT
   `d_product_item_id`    AS `d_product_item_id`,
   SUM( `Total Ext Sales Price` ) AS `sum_total__ext_sales_price_ok`
FROM
   `tpc-ds benchmark model - redshift`.`tpc-ds benchmark model` `tpc_ds_benchmark_model`
WHERE
   `I Category` = 'Jewelry'
AND `Sold Calendar Year` = 1999
AND `Sold d_month_of_year` = 9
AND `d_customer_gmt_offset` = -6
GROUP BY 1
```

18,593 bytes

Figure 12: AtScale SQL to answer question

As you can see, the SQL written against a semantic model like AtScale's is human readable and understandable. In addition, this semantic model provided important context for query optimization which delivered the query acceleration, user concurrency improvements and cost reduction in our benchmark tests.

As a measure of complexity, we used an open source parser to break down each SQL statement into the following groups: number of functions used, number of arithmetic operations, number of tables accessed, number of objects usee and number of subqueries needed.

Here are the results:

Configuration	Complexity Factor					
	# of Functions	# of Operations	# of Tables	# of Objects	# of Subqueries	Total Score
Without AtScale	87	66	177	700	27	1,057
With AtScale	36	2	21	198	1	258

Figure 13: Complexity score for TPC-DS benchmark with and without AtScale semantic layer

Conclusion

As you can see from the benchmark results, the future for data warehousing is definitely in the cloud. The cloud data warehouses we tested prove that the cloud is a viable alternative with many performance advantages for data warehousing compared to the traditional on-premise options and Redshift is a terrific option.

We also proved that the inclusion of a semantic layer like AtScale's can make cloud data warehouses even better by:

1. Drastically simplifying queries for users
2. Insuring all users access the same, secure data
3. Increasing query performance by up to 12.5x
4. Improving user concurrency by up to 61x
5. Reducing cost by up to 2.6x

ATSCALE