ATSCALE

WHITEPAPER

Enabling Natural Language Prompting with AtScale's Semantic Layer and Generative AI

Jeff Curran, Data Science Team Lead at AtScale

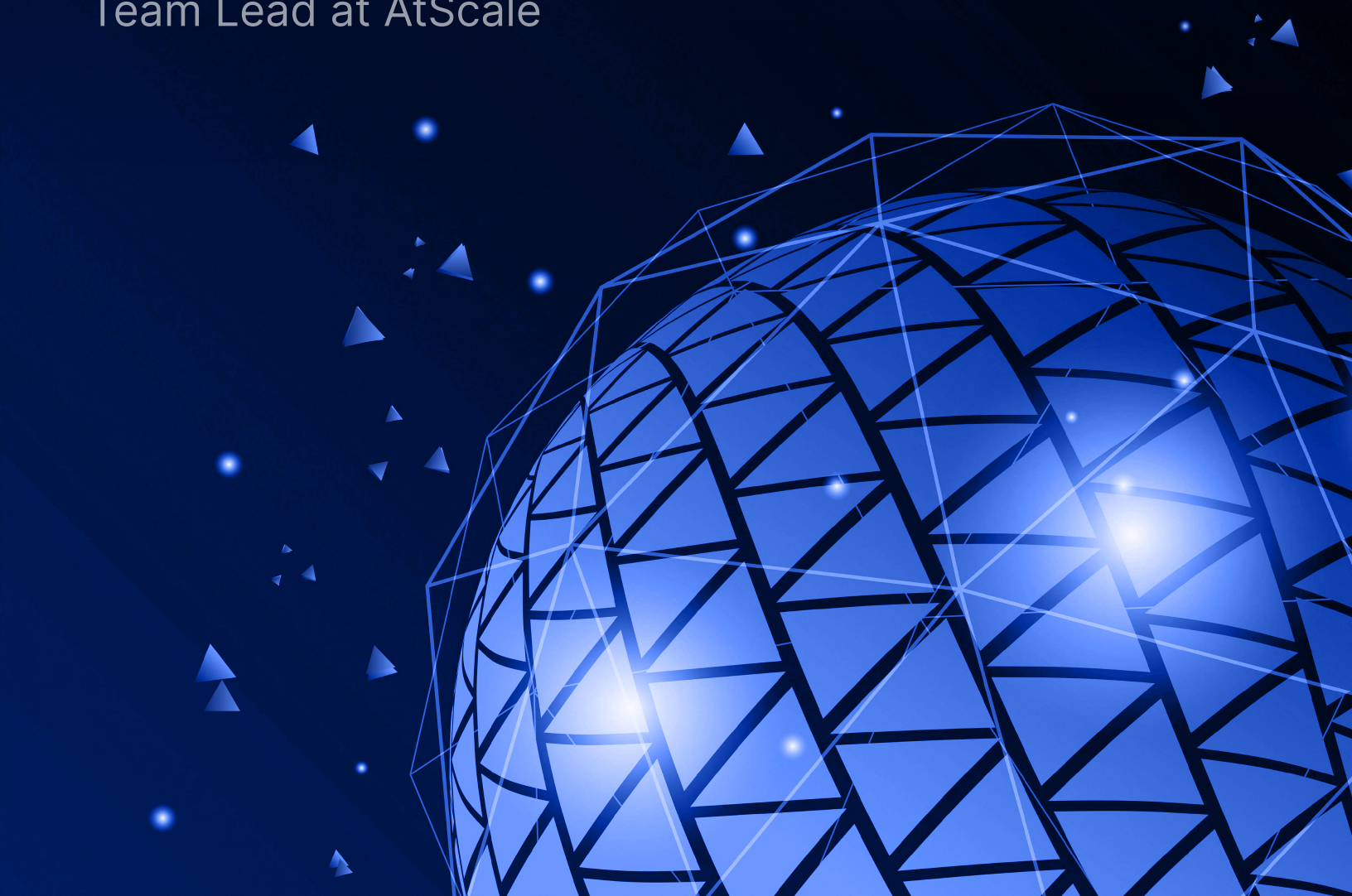

Table of Contents

Abstract

As businesses continue to grow their internal data warehouses, the bottleneck of human analysts in processing this data has become more pronounced. In light of this, there has been renewed interest in Text-to-SQL solutions, particularly those that could leverage the power of LLMs. However, such solutions typically struggle without a source of business logic and schema interactions to build off of. To alleviate these issues, **we look into how integrating the AtScale Semantic Layer and AtScale Query Engine with an LLM can impact performance for Text-to-SQL tasks**. Such an integration provides the LLM with a wealth of business-side metadata, while removing the requirement for the LLM to create metrics from scratch or generate joins.

In this paper, we explore how such an integration vastly improves result consistency and accuracy. A control system was established to evaluate this system that leverages few-shot prompting but only has access to the source schema. This control system achieved an accuracy of 20% across our question set.

> Impressively, our evaluation system integrating the AtScale Semantic Layer and the Query Engine with the LLM saw accuracy rise to 92.5%.

Thus, we show that the integration of AtScale's Semantic Layer and Query engine can bring Text-to-SQL solutions to a level of accuracy that is viable for day-to-day use cases.

1. Introduction

With the ever-increasing importance given to data-driven approaches to problem-solving, the number of questions asked of the world's data continues to grow [6]. Moving from data to answers has historically been solved using human-built analysis, queries, and dashboards. While these human-built solutions offer a viable solution for moving from raw data to digestible answers, they have a bottleneck in the form of human man-hours and data talent availability [1].

For example, while asking a question like "What was our total sales last month by region?" may be considered simple and only take a moment to ask, answering that question may require an analyst to undergo considerable work. So, while the time to ask may be on the scale of seconds, the time to answer may take minutes to hours, depending on the question's complexity and the structure of the source data. While this could in theory be alleviated by bringing on more personnel, this solution is often non-viable given the global shortage of tech talent [1]. Even worse, if the analysis is done in an ad-hoc fashion, the results of this question may be different if asked to multiple analysts.

Both of these factors can be a significant detriment to the decision-making process, and a substantial deterrent for those considering data-driven approaches. As such, there is a desire to use modern AI technologies to accelerate these kinds of Text-To-SQL tasks.

However, the problem of converting a natural language question into a database-readable query has been discussed and explored for years to little effect [2], so why is it now any different? In short, there is a belief that the explosive gains in performance shown in Generative AI models like Chat-GPT have finally made such solutions possible [4]. Couple that with the drastic improvements made to the accessibility of these Large Language Models (LLMs), and the ever-growing body of data that businesses generate, and the time is right for the problem of Natural Language Query (NLQ) tasks to be solved.

As previously mentioned, the task of NLQ solutions involves creating syntactically correct SQL queries against databases using a human-readable question as a starting point. Modern NLQ solutions hope that if an LLM can be adequately trained and provided the relevant data, then it can generate the SQL required to answer pertinent business questions on the fly. This would alleviate the existing bottleneck around human resources, and empower data-driven decision-making. However, even with the improvements seen in modern LLMs, there have been a number of known limitations to existing NLQ solutions:

1. LLMs struggle when database schemas get complex, and cannot make required table joins correctly.
2. LLMs are generative by design. This is great in some contexts, but it can provide inconsistent or incorrect results when generating calculations and KPIs for underlying data.
3. Database table and column names are often defined in ways that make sense in terms of a company's data engineering best practices. However, these may differ from the terminology used by the business. As a result, the LLM will struggle to generate correct queries without context of how to map from business to database terminology.

To resolve these limitations, there is ongoing research around adding a contextual layer on top of the database where joins, KPIs, and business terminology is defined. This layer, known as a Semantic Layer, can provide the LLM with the necessary metadata it needs to consistently complete the translation between natural language questions and viable SQL queries. While several Semantic Layers have recently begun development to try and resolve this issue, the AtScale Semantic Layer is a fully realized Semantic Layer that has been at the core of AtScale's technology since the company was founded over a decade ago.

> The AtScale Semantic Layer was built to enable centralized, business-friendly, and consistent definitions of KPIs and table relationships across a company's BI users regardless of the BI tool.

Thus, different teams and users can leverage KPIs and be confident that they will have the correct definition. This same fully fleshed out semantic model definition can be applied to today's NLQ tasks to provide invaluable metadata to the LLM. On top of this, additional existing technology, called the AtScale Query Engine, can be used to further empower NLQ tasks by completely removing the need for the LLM to generate joins.

This work aims to show how the AtScale Semantic Layer and underlying Query Engine can provide the necessary technology for NLQ solutions to be viable in today's business world. Specifically, we will look at how an AtScale-backed NLQ solution performs when asked real-world questions. In doing so, we hope to show the promise of this technology and identify areas where the solution could be further innovated.

2. AtScale's Semantic Layer and Query Engine

The core of AtScale's technology are the Universal Semantic Layer and the Query Engine. The Universal Semantic Layer exists as a place to define the business logic and KPIs relevant to making business decisions. The digestion of the Semantic Layer is then done downstream using Business Intelligence (BI) tools such as Power BI, Excel, or Tableau, or via the AtScale Python library. These tools can then communicate directly with AtScale using the language of their choice, whether it be DAX, PostgreSQL, MDX, or REST API queries. To ensure its Semantic Layer is truly universal, AtScale developed the Query Engine to resolve the relationships between the BI tool, the Semantic Layer, and the underlying data warehouse. The query engine exposes the semantic model as a single logical table for the BI tool to query, making it BI tool agnostic. It then translates queries against that table into the SQL dialect required to execute the query against the underlying warehouse, making it warehouse-agnostic. By combining the Query Engine's simplification benefits with the Semantic Layer's additional contextual information, AtScale enables NLQ products to perform at market-leading levels. These benefits are expanded on below.

2.1 Semantic Metadata

The AtScale Semantic Layer provides a location to define additional metadata that can be fed to the LLM. This includes business-relevant names, warehouse names, and freeform descriptions, giving the LLM the necessary context to move from NLQ to SQL.

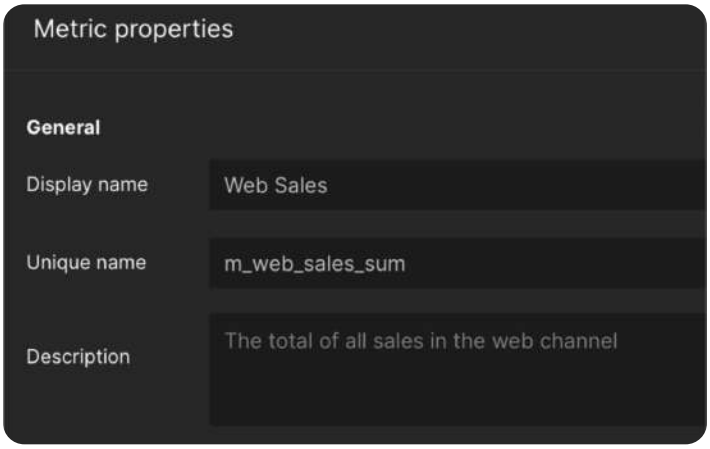

Fig 1: An example of a Measure (aggregation) in AtScale

While Fig 1 provides an example of this metadata via a measure, which is a numeric KPI, this metadata can similarly be provided for hierarchy levels and secondary attributes, covering all categorical KPIs as well.

2.2 The AtScale Query Engine

Traditionally, the Query Engine enabled AtScale users to connect to their Semantic Model from any combination of BI tools and data warehouses and gave them the flexibility to switch anytime. To accomplish this, queries submitted against a Semantic Layer are done as what is called an Inbound Query. Inbound Queries are submitted as either PostgreSQL, DAX, or MDX, where the queries all reference a single logical table that contains all objects in the Semantic Layer represented as columns. However, for our experiment's sake, we will ask the LLM only to generate queries in PostgreSQL. Once an Inbound Query is submitted, AtScale will use the definitions of the KPIs and joins defined in the Semantic Layer to translate the query into

Outbound SQL. In addition, the Outbound SQL is also automatically generated to handle orphaned fact table records, custom formatting (representing Nulls as 0's), and provide divide by zero protection. This translated query is then executed against the underlying data warehouse in that warehouse's SQL Dialect. An example of this is shown in Fig 2.

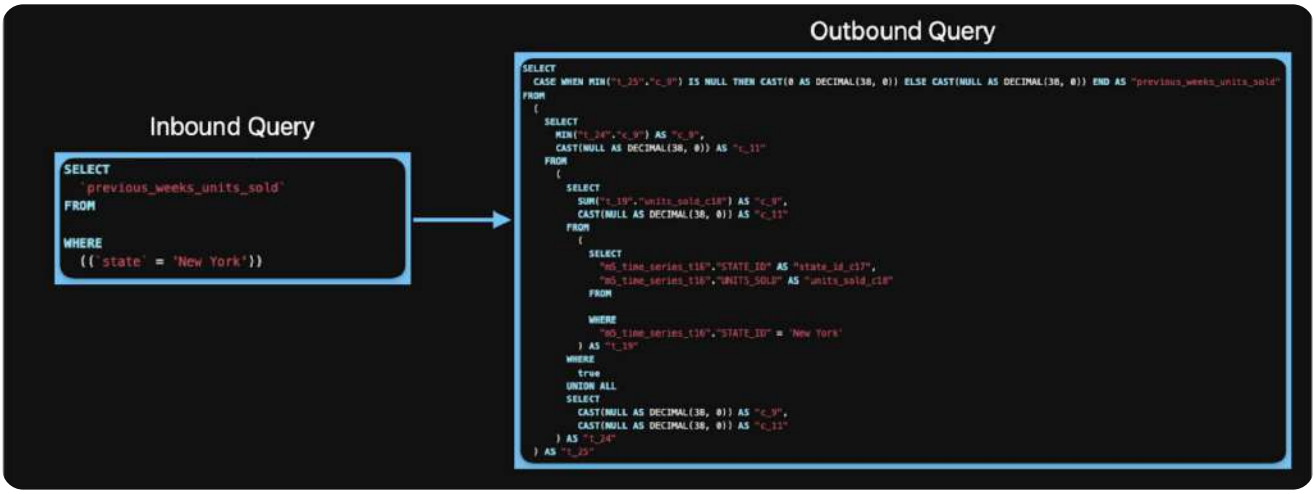

Fig 2. Translation of an inbound query into an outbound query using the AtScale Query Engine

In addition, this technology can also be applied to an NLQ use case. By tasking the LLM to create Inbound Queries instead of queries against the warehouse directly, we can leverage the same workflow as a BI tool and vastly simplify the problem the LLM is trying to solve. Since Inbound Queries are made against a single logic table, we can remove the need for the LLM to do joins or multi-table operations altogether. In addition, since the AtScale Query Engine will handle the translation between logical column names and KPIs, the LLM will never have to generate complex business logic. This simplifies the question, transforming even the most complex NLQ question into something almost solvable for the LLM.

> With the combination of the AtScale Semantic Layer and the AtScale Query Engine, an LLM can access a wealth of business-relevant metadata while simplifying the question being asked of it.

3. Experimental Setup

To facilitate this experiment, the following setup was leveraged:

1. Publicly Available Dataset: this experiment uses the TPC-DS Benchmark [5] dataset. The Transaction Processing Performance Council (TPC) provides this dataset for free and is downloadable using scripts provided on the TPC website. It is also frequently pre-loaded as a sample database by warehouse providers.

2. Evaluation Question Set: the set of natural language questions used for this experiment will be included in the paper's appendices for public use. This 40-question set was made leveraging the complexity spectrum described in the data.world paper authored by Juan Sequeda on LLMs empowered by Knowledge Graphs [4].

3. Publicly Available Semantic Layer: the success of this experiment is heavily dependent on the quality of the Semantic Layer the LLM will leverage. As such, AtScale stores this model definition in a publicly accessible location.

4. The AtScale Query Engine: will be leverage-able in the form of the AtScale Developer Edition, which will enable users to use a trial version of the AtScale Semantic Layer.

5. An out of the box LLM solution: In our case, we will be using Google's Gemini Pro 1.5 model due to credit availability. This model will be used to determine the performance of the control as well as the AtScale-empowered solution. No additional fine-tuning will be done on this model outside of prompt engineering.

3.1 Experimental Dataset

Several datasets were considered for this paper. The original hope was to leverage the benchmark dataset used in Juan Sequeda's data.world paper, but given the small number of rows in each table, there was concern that an incorrect query could return results that appear correct.

Instead, the dataset used for this experiment comes from the Transaction Processing Performance Council. This dataset contains two dozen tables and millions of rows of data, and has been used to benchmark data warehouse performance for years.

3.2 Evaluation Questions

To evaluate the performance of the Semantic Layer-backed LLM, a set of questions was created that follows an adapted version of the framework described in the data.world paper mentioned above. This framework splits questions along two spectrums: question complexity and schema complexity. In the original paper, these two spectrums are defined as:

- **Low question complexity:** Pertains to business reporting use cases, aimed at facilitating daily business operations. From a technical standpoint, these questions are translated into SELECT-FROM SQL queries [4].
- **High question complexity:** Arises in the context of Metrics and Key Performance Indicators (KPIs) within an organization. These questions are posed to make informed strategic decisions crucial for organizational success. From a technical standpoint, these questions are translated to SQL queries involving aggregations and mathematical functions [4].
- **Low schema complexity:** Small number of tables (i.e. 0 - 4), denormalized schema [4].
- **High schema complexity:** Larger number of tables (> 4), normalized schema, many-to-many join tables, etc [4].

Since the AtScale Semantic Layer abstracts away a degree of the underlying table complexity, this was adapted to mean the following:

- **Low question complexity:** Leverage only Dimensional Attributes, i.e. columns from the base tables accessible with SELECT-FROM SQL queries.
- **High question complexity:** Utilize Measures and Calculated Measures from the Semantic Model. Measures are used to define aggregations and Key Performance Indicators (KPIs) within the model.
- **Low schema complexity:** Small number of fact tables or dimensions (1-2)
- **High schema complexity:** Larger number of fact tables or dimensions (>= 3)

3.3 Experiment Semantic Layer

Both the semantic layer associated with this white paper, and the method of leveraging the semantic layer will be made available for public use. The semantic layer itself is defined in the Semantic Markup Language, AtScale's model definition language, and is housed in a publicly accessible git repository mentioned in Section 3.

The semantic layer provided includes all the KPI definitions, joins, and logical hierarchies required to answer the question set used in this experiment. This model is incomplete in terms of the number of KPIs defined or tables used, but it can always be added to as needed. That said, the semantic layer used should cover a wide range of questions focused on sales, shipping, and customer demographics, as shown in Fig 3.

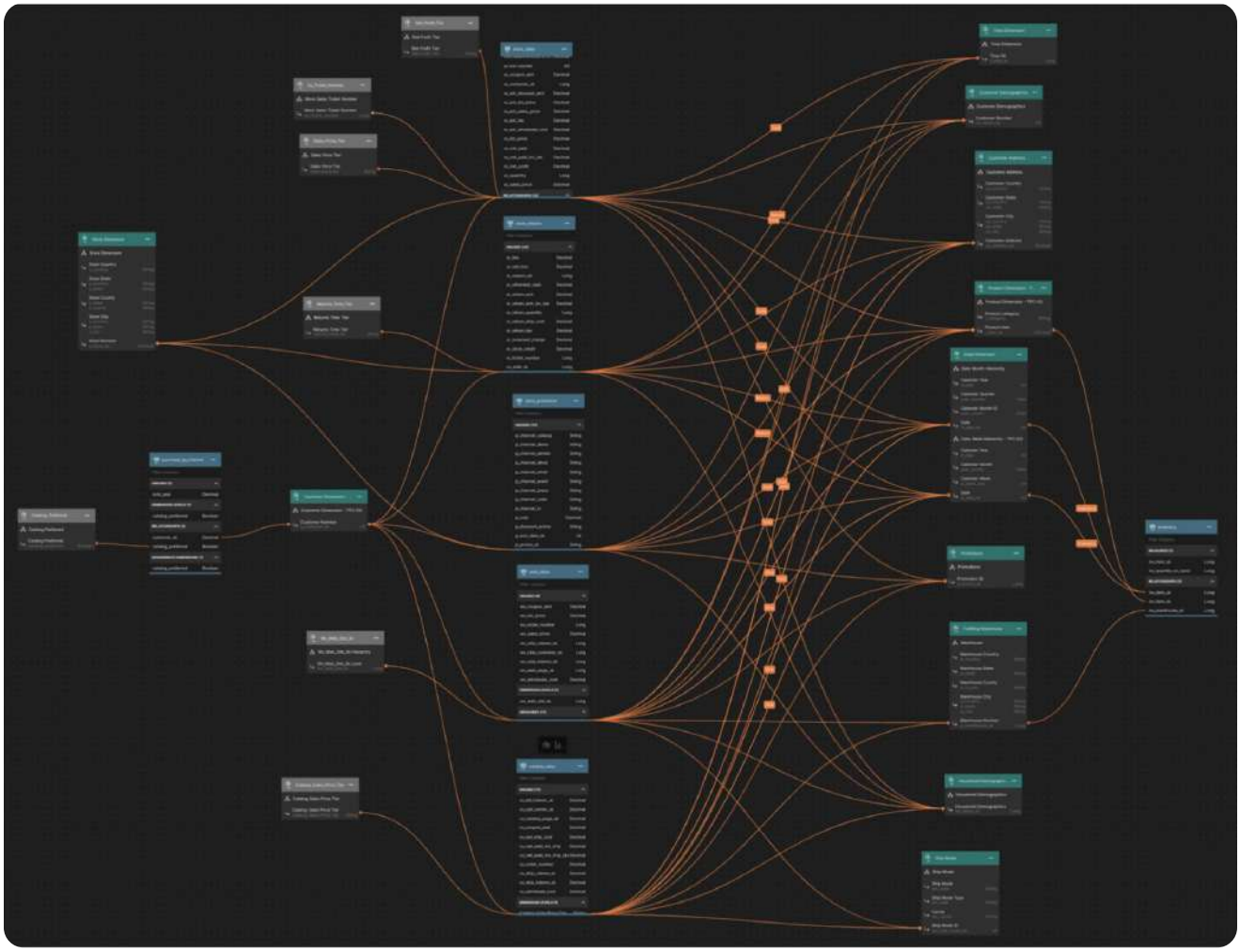

Fig 3: TPC-DS Modeled in AtScale

3.4 Experiment Evaluation

For this experiment, we calculated the accuracy in term evaluation accuracy.

EVALUATION ACCURACY

This captures the percentage of the time the LLM provides runnable SQL code that returns the intended results when run.

To account for the generative nature of the LLMs, we will run three iterations of our tests. All performance values will express the average accuracy across these three iterations.

In addition, the average performance will also be evaluated over the four quadrants defined in section 3.2 of this document.

Note that the LLM was also instructed to respond with “I’m not sure.” if it receives a question that is either unsolvable or irrelevant to the provided metadata. Since all the questions in our test set were deemed solvable with the underlying data, such a response will be counted as incorrect.

3.5 LLM Leveraged

For this paper, the Google Gemini Pro 1.5 model was leveraged. This model did not receive additional fine-tuning, but it was provided with a few-shot prompt to help it better generate queries for the AtScale Query Engine. The examples in this few-shot prompt were from separate dummy datasets, and did not include any examples relevant to the TPC-DS dataset.

This same model was used for the control, though a different prompt was provided. In place of the Semantic Model metadata, the prompt for the control contained the warehouse schema of the underlying tables in the form of a DDL, including primary and foreign key definitions.

For this experiment, the model was set to a temperature of 0.0. The Google documentation [3] states that the temperature only comes into effect when sampling parameters are set on the model. Since we are aiming to test out-of-the-box LLM performance without fine-tuning, these parameters were not set.

3.6 Prompting System for Control Set

To establish our control, the system described in Fig 4 was used. This system feeds a question and the schema DDL into a few-shot prompt. This prompt is then sent to the Gemini Model to generate the SQL. If the LLM deems the question unsolvable or irrelevant, it will return with 'Not Sure,' though the prompt specifies that a SQL response is always preferred. If SQL is returned, the query's result is evaluated.

If there is a SQL error while executing the query, the question is again posed to the LLM, asking it to generate a new response. The LLM is given three chances to generate runnable SQL before it returns the error. If the query runs without error, that result is then returned. This result is then manually checked for correctness.

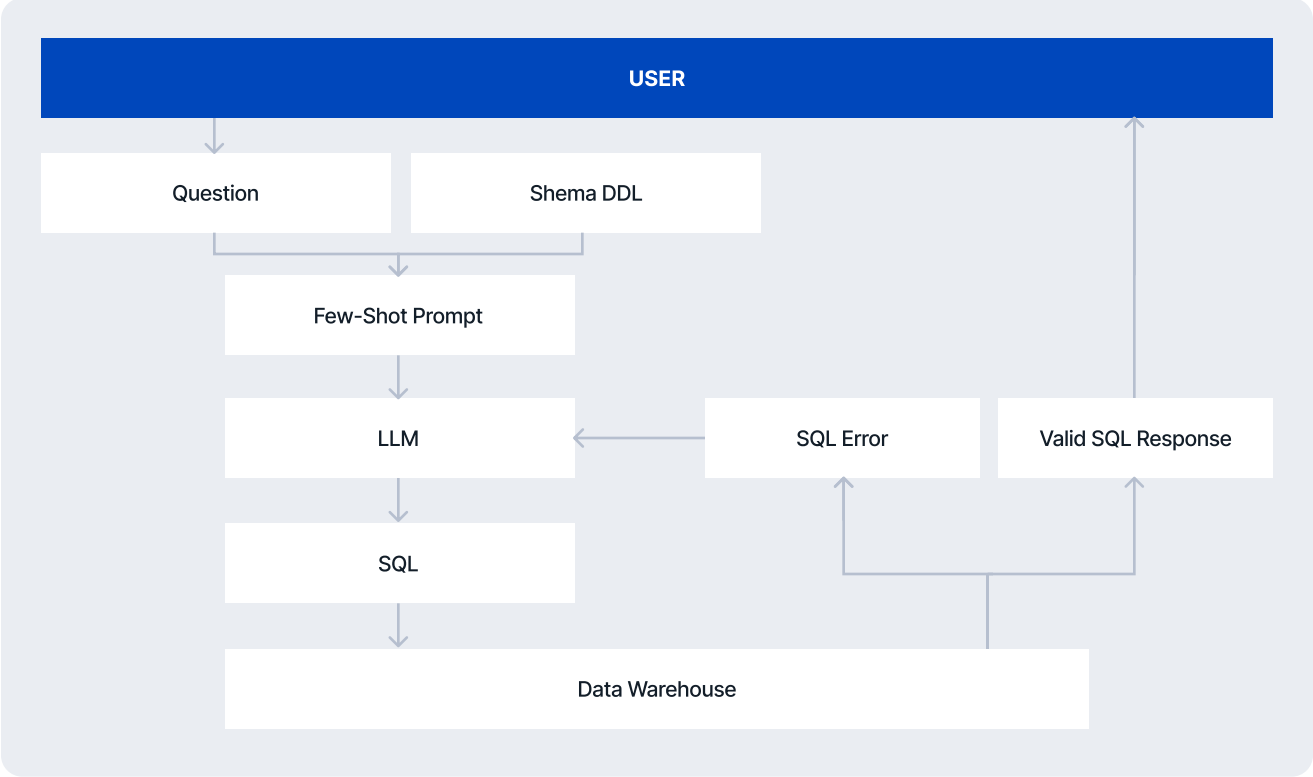

Fig 4. Flow diagram for the prompting system used with control Set

3.7 Prompting System for Evaluation Set

For the evaluation set, the system described in Fig 5 was used. This system differs from the control in a few key ways. First, instead of the DDL being provided in the prompt, metadata about the semantic model's logical query table is provided. It is important to specify here that only metadata from the Semantic Layer is provided to the LLM, no data from the underlying warehouse tables is sent to the LLM. Second, in this system, the generated queries are submitted against the AtScale Query Engine instead of the data warehouse. The final change is that AtScale determines the validity of the SQL syntax for the query made against it, as opposed to the data warehouse.

Similar to the control system, if AtScale detects a SQL syntax error, the LLM will generate a new query. As in the control, the LLM is given up to three chances to generate SQL. If a query runs without error, the loop breaks and the result of running that query is returned to the user.

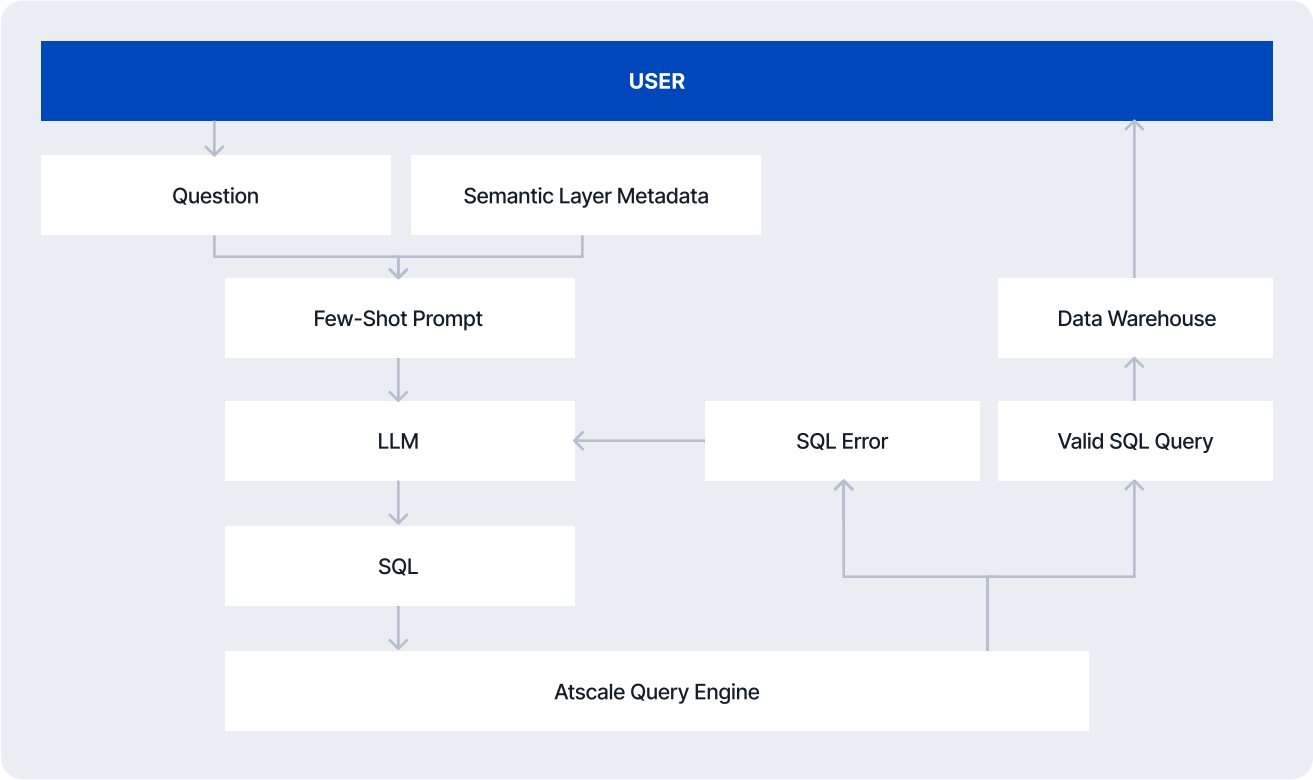

Fig 5: Flow diagram for the prompting system used with evaluation set

4. Results

In this experiment, 40 questions across four quadrants of complexity were considered. These questions were posed to our control and evaluation systems, providing the results in Table 1.

QUESTION COMPLEXITY	SCHEMA COMPLEXITY	ACCURACY		NET IMPROVEMENT
		W/O ATSCALE SEMANTIC LAYER	W/ ATSCALE SEMANTIC LAYER	
Low	Low	60%	100%	40%
Low	High	10%	100%	90%
High	Low	10%	100%	90%
High	High	0%	70%	70%
Overall		20%	92.5%	72.5%

Table 1: Performance Table Comparing Control to Evaluation Sets

As seen above, leveraging AtScale's Semantic Layer and Query Engine has had a profound effect on the LLM's performance. Across the lower 3 difficulty quadrants, the Semantic layer-backed LLM performed at 100% accuracy, compared to 26.6% accuracy without the Semantic Layer. Even in the most difficult quadrant, the Semantic Layer-backed LLM's 70% accuracy rate far exceeded the control system's rate of 0%.

4.1 Discussion

With the AtScale enhanced LLM, even queries involving multi-table operations and KPIs are made solvable. This is a massive improvement over the control system which was completely ineffective at answering questions in either of our High Complexity categories. For example, a question like "What was the sum of web sales for each product brand in the year 2002?", requires sql that defines the "sum of web sales" KPI, as well as joins between the underlying web_sales, date_dim, and item_dim tables. This level of high schema and question complexity questions was unsolvable for the control NLQ system, but returned the correct result when using the AtScale backed NLQ system.

In spite of our success, there is additional work that could be done to further improve the model. This work's primary focus will be adjusting how the semantic model's metadata is delivered to the LLM. Specifically, we will explore segmenting the metadata into smaller sections in the hope that column names will be processed better.

Additionally, the feedback loop dealing with the case where bad queries are submitted against the Query Engine can be improved. If specific issues with the generated query can be identified, the LLM can be given more specific feedback for the later iterations of the question loop. This could result in the LLM reaching a runnable query more quickly, saving the user time and money.

The LLM was also allowed to respond with 'Not Sure' when it did not believe the query was solvable with the provided metadata. Since all of our questions were solvable with the provided semantic layer, no such responses should have occurred. The fact the LLM did respond with this message contributed to the inaccurate results and suggests that there are limits to the complexity of questions the LLM can process.

Similarly, during manual analysis of the generated evaluation queries, there were a few cases where the LLM was seen to struggle with generating inaccurate answers.

1. Column Name Hallucinations: Even though the column names were provided, there were cases where the LLM referenced column names that did not exist in the table. The name generated by the LLM was always a simplified version of a column from the provided metadata.
2. Multiple Numeric Omissions: In one case where a question involved multiple numeric filters, only a single filter was included in the response query.

In addition, a more advanced system to accommodate filter synonyms will be explored. In our current system, all filters are passed in a way that explicitly matches the value in the underlying table in terms of case sensitivity and spelling. For example, if a product was in the category "Sports", a question asking for that category would have to match that spelling and case exactly. For example, if you wanted the sales for every product in the sports category, the question would have to look something like: "what were the total sales for each product in product category Sports?". For the future, we are looking to make the filtering system more flexible by incorporating advanced filtering behavior into AtScale.

In the future, I believe the most profound impact would be seen by providing a dataset to fine-tune the LLM itself. By restricting the LLM to work off of only a single table, not allowing any joins or aggregations, we are likely fighting any other SQL-related training data that was provided to the model. If a fine-tuning dataset was provided to the model, I believe we would achieve similar or better results as those seen in this paper with zero-shot prompting. This would improve processing time while reducing the number of tokens included in the prompt, reducing the LLM query cost.

5. Key Takeaways

This paper has shown that pairing the power of AtScale's Semantic Layer and Query Engine with an out-of-the-box LLM can provide a viable NLQ solution for simple business questions.

Using our test set of 40 questions, we achieved an overall accuracy of 92.5%, performing better than our control set (20%) by more than 70 percentage points. In addition, for all but the most complex queries in the test set, the Semantic Layer-backed LLM performed at 100% accuracy.

Going forward, further research will be conducted to improve the prompt sent to the LLM and mitigate the few issues seen today. Additionally, we believe that a set of additional training data designed to prepare an LLM for work with the AtScale Query Engine could vastly improve performance on even higher complexity question sets. In conclusion, the AtScale Semantic Layer provides a viable solution to accomplishing basic NLQ tasks.

6. References

[1] Adastra, and Forum Research Inc. “Adastra 2024 Data Professionals Market Survey Forecast (North America).” Adastracorp.Com, 2024, https://staging2.adastracorp.com/downloads/report/north-america-adastra-2024-data-professionals-market-survey-forecast.pdf.

[2] Androutsopoulos, I., et al. “Natural language interfaces to databases – an introduction.” Natural Language Engineering, vol. 1, no. 1, Mar. 1995, pp. 29–81, https://doi.org/10.1017/s135132490000005x.

[3] “Experiment with Parameter Values | Generative AI on Vertex AI | Google Cloud.” Google, cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/adjust-parameter-values.

[4] Sequeda, Juan, et al. “A benchmark to understand the role of knowledge graphs on large language model’s accuracy for question answering on enterprise SQL databases.” Proceedings of the 7th Joint Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), 9 June 2024, https://doi.org/10.1145/3661304.3661901.

[5] Transaction Processing Performance Council (TPC). TPC-DS Specification V 3.2.0, June 2021, www.tpc.org/tpc_documents_current_versions/pdf/tpc-ds_v3.2.0.pdf.

[6] Wavestone. 2024 DATA AND AI LEADERSHIP EXECUTIVE SURVEY, 2024, wwa.wavestone.com/app/uploads/2023/12/DataAI-ExecutiveLeadershipSurveyFinalAsset.pdf.

7. Appendix

7.1 Question Set

Low Question/Low Schema Complexity

1. What are all of the unique store numbers in the state of Tennessee?
2. What are the first and last names of all customers that have more than 5 dependents?
3. What is the name, manager and floor space of each store in the city of Midway?
4. What is the name of each product of brand name 'corpcorp #1'?
5. What brand names are made by the manufacturer named able?
6. What shipping carriers are available for shipping type 'EXPRESS'?
7. What is the name and square footage for each warehouse in the city of Fairview?
8. What is the name of each product with a current price greater than 70?
9. For each store name, how many employees work there?
10. What is the name of each warehouse with more than 140000 square feet of space?

High Question/Low Schema Complexity

1. What is the total count of customers for each customer home state?
2. What is the number of store returns for each product?
3. What was the overall web sales for the year 2002?
4. What is the average sales quantity per store number?
5. What is the total net profit per product name?
6. What is the count of catalog customers?
7. For each brand name, what was the quantity sold in store?
8. What is the total quantity sold each year?
9. What is the count of products in each product category?
10. What is the total web sales for each customer home state?

Low Question/High Schema Complexity

1. What is the customer id and vehicle count for every customer in income band 9?
2. What is the net profit tier for each store name and gender in the 2002 sales year?
3. What was the first name and gender of each customer that shopped in the store named 'ese' in 2001?
4. For every store in the city of Midway, what is the store's name and the customer income bands seen during the 2002 sales year?
5. What is the first name, income band, vehicle count, and buy potential for each customer that lives in the state of New Jersey?
6. What is the first name, gender, income band, and home state name for each customer that was born in the country of CYPRUS?
7. What shipping types were used by customers with the last name 'Abel' in the year 2002?
8. What is the name and size of each product sold to customers of gender 'F' living in the city of Wright?
9. What cities have customers with more than 5 dependents and are in income band 7?
10. What product names were bought by customers with 4 dependents for the sales year of 2002?

High Question/High Schema Complexity

1. What is the count of customers for each state that had more than 500000000 in total catalog and web sales in 2002?
2. For each store state in the year 2002, what was the count of store customers and the average sales quantity?
3. What were the store sales in 2002 for each store manager in the state of Tennessee?
4. For each website, what is the quantity sold and total shipping cost for customers with a shipping address in the state of New Jersey?
5. What was the average store net profit for each product brand sold in stores in the state of Tennessee in the 2002 sales year?
6. What are the store quantity sold and returned, for each store in the state of Tennessee, also include store hours and store floor space?
7. What was the sum of web sales for each product brand in the year 2002?
8. What is the total catalog sales by zip code for customers living in the state of Maryland?
9. What is the product brand name and color for all products with non-zero inventory on hand in the year 2002?
10. For every store name and product category, what is the average store sales quantity?

7.2 Control SQL DDL

The following DDL is written in GoogleSQL. This DDL was provided in the prompt of our control system.

```
CREATE TABLE `tpcds.household_demographics`
(
  hd_demo_sk INT64 NOT NULL,
  hd_income_band_sk INT64,
  hd_buy_potential STRING,
  hd_dep_count INT64,
  hd_vehicle_count INT64,
  PRIMARY KEY (hd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (hd_income_band_sk) REFERENCES tpcds.income_band
(ib_income_band_sk) NOT ENFORCED
);
CREATE TABLE `tpcds.web_sales`
(
  ws_sold_date_sk INT64,
  ws_sold_time_sk INT64,
  ws_ship_date_sk INT64,
  ws_item_sk INT64 NOT NULL,
  ws_bill_customer_sk INT64,
  ws_bill_cdemo_sk INT64,
  ws_bill_hdemo_sk INT64,
  ws_bill_addr_sk INT64,
  ws_ship_customer_sk INT64,
  ws_ship_cdemo_sk INT64,
  ws_ship_hdemo_sk INT64,
  ws_ship_addr_sk INT64,
  ws_web_page_sk INT64,
  ws_web_site_sk INT64,
  ws_ship_mode_sk INT64,
  ws_warehouse_sk INT64,
  ws_promo_sk INT64,
  ws_order_number INT64 NOT NULL,
  ws_quantity INT64,
```

```
  ws_wholesale_cost NUMERIC(7,2),
  ws_list_price NUMERIC(7,2),
  ws_sales_price NUMERIC(7,2),
  ws_ext_discount_amt NUMERIC(7,2),
  ws_ext_sales_price NUMERIC(7,2),
  ws_ext_wholesale_cost NUMERIC(7,2),
  ws_ext_list_price NUMERIC(7,2),
  ws_ext_tax NUMERIC(7,2),
  ws_coupon_amt NUMERIC(7,2),
  ws_ext_ship_cost NUMERIC(7,2),
  ws_net_paid NUMERIC(7,2),
  ws_net_paid_inc_tax NUMERIC(7,2),
  ws_net_paid_inc_ship NUMERIC(7,2),
  ws_net_paid_inc_ship_tax NUMERIC(7,2),
  ws_net_profit NUMERIC(7,2),
  PRIMARY KEY (ws_item_sk, ws_order_number) NOT ENFORCED,
  FOREIGN KEY (ws_sold_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED,
  FOREIGN KEY (ws_sold_time_sk) REFERENCES tpcds.time_dim
(t_time_sk) NOT ENFORCED,
  FOREIGN KEY (ws_ship_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED,
  FOREIGN KEY (ws_item_sk) REFERENCES tpcds.item (i_item_sk) NOT
ENFORCED,
  FOREIGN KEY (ws_bill_customer_sk) REFERENCES tpcds.customer
(c_customer_sk) NOT ENFORCED,
  FOREIGN KEY (ws_bill_cdemo_sk) REFERENCES
tpcds.customer_demographics (cd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (ws_bill_hdemo_sk) REFERENCES
tpcds.household_demographics (hd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (ws_bill_addr_sk) REFERENCES tpcds.customer_address
(ca_address_sk) NOT ENFORCED,
  FOREIGN KEY (ws_ship_customer_sk) REFERENCES tpcds.customer
(c_customer_sk) NOT ENFORCED,
  FOREIGN KEY (ws_ship_cdemo_sk) REFERENCES
tpcds.customer_demographics (cd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (ws_ship_hdemo_sk) REFERENCES
```

```
 tpcds.household_demographics (hd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (ws_ship_addr_sk) REFERENCES tpcds.customer_address
(ca_address_sk) NOT ENFORCED,
  FOREIGN KEY (ws_web_page_sk) REFERENCES tpcds.call_center
(wp_web_page_sk) NOT ENFORCED,
  FOREIGN KEY (ws_web_site_sk) REFERENCES tpcds.catalog_page
(web_site_sk) NOT ENFORCED,
  FOREIGN KEY (ws_ship_mode_sk) REFERENCES tpcds.ship_mode
(sm_ship_mode_sk) NOT ENFORCED,
  FOREIGN KEY (ws_warehouse_sk) REFERENCES tpcds.warehouse
(w_warehouse_sk) NOT ENFORCED,
  FOREIGN KEY (ws_promo_sk) REFERENCES tpcds.promotion (p_promo_sk)
NOT ENFORCED
);

CREATE TABLE `tpcds.web_site`
(
  web_site_sk INT64 NOT NULL,
  web_site_id STRING NOT NULL,
  web_rec_start_date DATE,
  web_rec_end_date DATE,
  web_name STRING,
  web_open_date_sk INT64,
  web_close_date_sk INT64,
  web_class STRING,
  web_manager STRING,
  web_mkt_id INT64,
  web_mkt_class STRING,
  web_mkt_desc STRING,
  web_market_manager STRING,
  web_company_id INT64,
  web_company_name STRING,
  web_street_number STRING,
  web_street_name STRING,
  web_street_type STRING,
  web_suite_number STRING,
```

```
  web_city STRING,
  web_county STRING,
  web_state STRING,
  web_zip STRING,
  web_country STRING,
  web_gmt_offset NUMERIC(5,2),
  web_tax_percentage NUMERIC(5,2),
  PRIMARY KEY (web_site_sk) NOT ENFORCED,
  FOREIGN KEY (web_open_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED,
  FOREIGN KEY (web_close_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED
);

CREATE TABLE `tpcds.income_band`
(
  ib_income_band_sk INT64 NOT NULL,
  ib_lower_bound INT64,
  ib_upper_bound INT64,
  PRIMARY KEY (ib_income_band_sk) NOT ENFORCED
);

CREATE TABLE `tpcds.call_center`
(
  cc_call_center_sk INT64 NOT NULL,
  cc_call_center_id STRING NOT NULL,
  cc_rec_start_date DATE,
  cc_rec_end_date DATE,
  cc_closed_date_sk INT64,
  cc_open_date_sk INT64,
  cc_name STRING,
  cc_class STRING,
  cc_employees INT64,
  cc_sq_ft INT64,
  cc_hours STRING,
  cc_manager STRING,
```

```
  cc_mkt_id INT64,
  cc_mkt_class STRING,
  cc_mkt_desc STRING,
  cc_market_manager STRING,
  cc_division INT64,
  cc_division_name STRING,
  cc_company INT64,
  cc_company_name STRING,
  cc_street_number STRING,
  cc_street_name STRING,
  cc_street_type STRING,
  cc_suite_number STRING,
  cc_city STRING,
  cc_county STRING,
  cc_state STRING,
  cc_zip STRING,
  cc_country STRING,
  cc_gmt_offset NUMERIC(5,2),
  cc_tax_percentage NUMERIC(5,2),
  PRIMARY KEY (cc_call_center_sk) NOT ENFORCED,
  FOREIGN KEY (cc_closed_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED,
  FOREIGN KEY (cc_open_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED
);

CREATE TABLE `tpcds.catalog_page`
(
  cp_catalog_page_sk INT64 NOT NULL,
  cp_catalog_page_id STRING NOT NULL,
  cp_start_date_sk INT64,
  cp_end_date_sk INT64,
cp_department STRING,
  cp_catalog_number INT64,
  cp_catalog_page_number INT64,
  cp_description STRING,
```

```
  cp_type STRING,
  PRIMARY KEY (cp_catalog_page_sk) NOT ENFORCED,
  FOREIGN KEY (cp_end_date_sk) REFERENCES tpcds.date_dim (d_date_sk)
NOT ENFORCED,
  FOREIGN KEY (cp_start_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED
);

CREATE TABLE `tpcds.promotion`
(
  p_promo_sk INT64 NOT NULL,
  p_promo_id STRING NOT NULL,
  p_start_date_sk INT64,
  p_end_date_sk INT64,
  p_item_sk INT64,
  p_cost NUMERIC(15,2),
  p_response_target INT64,
  p_promo_name STRING,
  p_channel_dmail STRING,
  p_channel_email STRING,
  p_channel_catalog STRING,
  p_channel_tv STRING,
  p_channel_radio STRING,
  p_channel_press STRING,
  p_channel_event STRING,
  p_channel_demo STRING,
  p_channel_details STRING,
  p_purpose STRING,
  p_discount_active STRING,
  PRIMARY KEY (p_promo_sk) NOT ENFORCED,
  FOREIGN KEY (p_start_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED,
  FOREIGN KEY (p_end_date_sk) REFERENCES tpcds.date_dim (d_date_sk)
NOT ENFORCED,
  FOREIGN KEY (p_item_sk) REFERENCES tpcds.item (i_item_sk) NOT
ENFORCED
);
```

```
CREATE TABLE `tpcds.store_sales`
(
  ss_sold_date_sk INT64,
  ss_sold_time_sk INT64,
  ss_item_sk INT64 NOT NULL,
  ss_customer_sk INT64,
  ss_cdemo_sk INT64,
  ss_hdemo_sk INT64,
  ss_addr_sk INT64,
  ss_store_sk INT64,
  ss_promo_sk INT64,
  ss_ticket_number INT64 NOT NULL,
  ss_quantity INT64,
  ss_wholesale_cost NUMERIC(7,2),
  ss_list_price NUMERIC(7,2),
  ss_sales_price NUMERIC(7,2),
  ss_ext_discount_amt NUMERIC(7,2),
  ss_ext_sales_price NUMERIC(7,2),
  ss_ext_wholesale_cost NUMERIC(7,2),
  ss_ext_list_price NUMERIC(7,2),
  ss_ext_tax NUMERIC(7,2),
  ss_coupon_amt NUMERIC(7,2),
  ss_net_paid NUMERIC(7,2),
  ss_net_paid_inc_tax NUMERIC(7,2),
  ss_net_profit NUMERIC(7,2),
  PRIMARY KEY (ss_item_sk, ss_ticket_number) NOT ENFORCED,
  FOREIGN KEY (ss_sold_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED,
  FOREIGN KEY (ss_sold_time_sk) REFERENCES tpcds.time_dim
(t_time_sk) NOT ENFORCED,
  FOREIGN KEY (ss_item_sk) REFERENCES tpcds.item (i_item_sk) NOT
ENFORCED,
  FOREIGN KEY (ss_customer_sk) REFERENCES tpcds.customer
(c_customer_sk) NOT ENFORCED,
  FOREIGN KEY (ss_cdemo_sk) REFERENCES tpcds.customer_demographics
```

```
(cd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (ss_hdemo_sk) REFERENCES tpcds.household_demographics
(hd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (ss_addr_sk) REFERENCES tpcds.customer_address
(ca_address_sk) NOT ENFORCED,
  FOREIGN KEY (ss_store_sk) REFERENCES tpcds.store (s_store_sk) NOT
ENFORCED,
  FOREIGN KEY (ss_promo_sk) REFERENCES tpcds.promotion (p_promo_sk)
NOT ENFORCED
);

CREATE TABLE `tpcds.catalog_returns`
(
  cr_returned_date_sk INT64,
  cr_returned_time_sk INT64,
  cr_item_sk INT64 NOT NULL,
  cr_refunded_customer_sk INT64,
  cr_refunded_cdemo_sk INT64,
  cr_refunded_hdemo_sk INT64,
  cr_refunded_addr_sk INT64,
  cr_returning_customer_sk INT64,
  cr_returning_cdemo_sk INT64,
  cr_returning_hdemo_sk INT64,
  cr_returning_addr_sk INT64,
  cr_call_center_sk INT64,
  cr_catalog_page_sk INT64,
  cr_ship_mode_sk INT64,
  cr_warehouse_sk INT64,
  cr_reason_sk INT64,
  cr_order_number INT64 NOT NULL,
  cr_return_quantity INT64,
  cr_return_amount NUMERIC(7,2),
  cr_return_tax NUMERIC(7,2),
  cr_return_amt_inc_tax NUMERIC(7,2),
  cr_fee NUMERIC(7,2),
  cr_return_ship_cost NUMERIC(7,2),
```

```
  cr_refunded_cash NUMERIC(7,2),
  cr_reversed_charge NUMERIC(7,2),
  cr_store_credit NUMERIC(7,2),
  cr_net_loss NUMERIC(7,2),
  PRIMARY KEY (cr_item_sk, cr_order_number) NOT ENFORCED,
  FOREIGN KEY (cr_returned_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED,
  FOREIGN KEY (cr_returned_time_sk) REFERENCES tpcds.time_dim
(t_time_sk) NOT ENFORCED,
  FOREIGN KEY (cr_item_sk, cr_order_number) REFERENCES
tpcds.catalog_sales (cs_item_sk, cs_order_number) NOT ENFORCED,
  FOREIGN KEY (cr_item_sk) REFERENCES tpcds.item (i_item_sk) NOT
ENFORCED,
  FOREIGN KEY (cr_refunded_customer_sk) REFERENCES tpcds.customer
(c_customer_sk) NOT ENFORCED,
  FOREIGN KEY (cr_refunded_cdemo_sk) REFERENCES
tpcds.customer_demographics (cd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (cr_refunded_hdemo_sk) REFERENCES
tpcds.household_demographics (hd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (cr_refunded_addr_sk) REFERENCES
tpcds.customer_address (ca_address_sk) NOT ENFORCED,
  FOREIGN KEY (cr_returning_customer_sk) REFERENCES tpcds.customer
(c_customer_sk) NOT ENFORCED,
  FOREIGN KEY (cr_returning_cdemo_sk) REFERENCES
tpcds.customer_demographics (cd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (cr_returning_hdemo_sk) REFERENCES
tpcds.household_demographics (hd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (cr_returning_addr_sk) REFERENCES
tpcds.customer_address (ca_address_sk) NOT ENFORCED,
  FOREIGN KEY (cr_call_center_sk) REFERENCES tpcds.call_center
(cs_call_center_sk) NOT ENFORCED,
  FOREIGN KEY (cr_catalog_page_sk) REFERENCES tpcds.catalog_page
(cp_catalog_page_sk) NOT ENFORCED,
  FOREIGN KEY (cr_ship_mode_sk) REFERENCES tpcds.ship_mode
(sm_ship_mode_sk) NOT ENFORCED,
  FOREIGN KEY (cr_warehouse_sk) REFERENCES tpcds.warehouse
```

```
(w_warehouse_sk) NOT ENFORCED,
  FOREIGN KEY (cr_reason_sk) REFERENCES tpcds.reason (r_reason_sk)
NOT ENFORCED
);

CREATE TABLE `tpcds.item`
(
  i_item_sk INT64 NOT NULL,
  i_item_id STRING NOT NULL,
  i_rec_start_date DATE,
  i_rec_end_date DATE,
  i_item_desc STRING,
  i_current_price NUMERIC(7,2),
  i_wholesale_cost NUMERIC(7,2),
  i_brand_id INT64,
  i_brand STRING,
  i_class_id INT64,
  i_class STRING,
  i_category_id INT64,
  i_category STRING,
  i_manufact_id INT64,
  i_manufact STRING,
  i_size STRING,
  i_formulation STRING,
  i_color STRING,
  i_units STRING,
  i_container STRING,
  i_manager_id INT64,
  i_product_name STRING,
  PRIMARY KEY (i_item_sk) NOT ENFORCED
);

CREATE TABLE `tpcds.web_page`
(
  wp_web_page_sk INT64 NOT NULL,
  wp_web_page_id STRING NOT NULL,
```

```
wp_rec_start_date DATE,
  wp_rec_end_date DATE,
  wp_creation_date_sk INT64,
  wp_access_date_sk INT64,
  wp_autogen_flag STRING,
  wp_customer_sk INT64,
  wp_url STRING,
  wp_type STRING,
  wp_char_count INT64,
  wp_link_count INT64,
  wp_image_count INT64,
  wp_max_ad_count INT64,
  PRIMARY KEY (wp_web_page_sk) NOT ENFORCED,
  FOREIGN KEY (wp_creation_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED,
  FOREIGN KEY (wp_access_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED,
  FOREIGN KEY (wp_customer_sk) REFERENCES tpcds.customer
(c_customer_sk) NOT ENFORCED
);

CREATE TABLE `tpcds.customer_address`
(
  ca_address_sk INT64 NOT NULL,
  ca_address_id STRING NOT NULL,
  ca_street_number STRING,
  ca_street_name STRING,
  ca_street_type STRING,
  ca_suite_number STRING,
  ca_city STRING,
  ca_county STRING,
  ca_state STRING,
  ca_zip STRING,
  ca_country STRING,
  ca_gmt_offset NUMERIC(5,2),
  ca_location_type STRING,
```

```
  PRIMARY KEY (ca_address_sk) NOT ENFORCED
);

CREATE TABLE `tpcds.date_dim`
(
  d_date_sk INT64 NOT NULL,
  d_date_id STRING NOT NULL,
  d_date DATE,
  d_month_seq INT64,
  d_week_seq INT64,
  d_quarter_seq INT64,
  d_year INT64,
  d_dow INT64,
  d_moy INT64,
  d_dom INT64,
  d_qoy INT64,
  d_fy_year INT64,
  d_fy_quarter_seq INT64,
  d_fy_week_seq INT64,
  d_day_name STRING,
  d_quarter_name STRING,
  d_holiday STRING,
  d_weekend STRING,
  d_following_holiday STRING,
  d_first_dom INT64,
  d_last_dom INT64,
  d_same_day_ly INT64,
  d_same_day_lq INT64,
  d_current_day STRING,
  d_current_week STRING,
  d_current_month STRING,
  d_current_quarter STRING,
  d_current_year STRING,
  PRIMARY KEY (d_date_sk) NOT ENFORCED
);
```

```
CREATE TABLE `tpcds.store`
(
  s_store_sk INT64 NOT NULL,
  s_store_id STRING NOT NULL,
  s_rec_start_date DATE,
  s_rec_end_date DATE,
  s_closed_date_sk INT64,
  s_store_name STRING,
  s_number_employees INT64,
  s_floor_space INT64,
  s_hours STRING,
  s_manager STRING,
  s_market_id INT64,
  s_geography_class STRING,
  s_market_desc STRING,
  s_market_manager STRING,
  s_division_id INT64,
  s_division_name STRING,
  s_company_id INT64,
  s_company_name STRING,
  s_street_number STRING,
  s_street_name STRING,
  s_street_type STRING,
  s_suite_number STRING,
  s_city STRING,
  s_county STRING,
  s_state STRING,
  s_zip STRING,
  s_country STRING,
  s_gmt_offset NUMERIC(5,2),
  s_tax_precentage NUMERIC(5,2),
  PRIMARY KEY (s_store_sk) NOT ENFORCED,
  FOREIGN KEY (s_closed_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED
);
```

```
CREATE TABLE `tpcds.ship_mode`
(
  sm_ship_mode_sk INT64 NOT NULL,
  sm_ship_mode_id STRING NOT NULL,
sm_type STRING,
  sm_code STRING,
  sm_carrier STRING,
  sm_contract STRING,
  PRIMARY KEY (sm_ship_mode_sk) NOT ENFORCED
);

CREATE TABLE `tpcds.inventory`
(
  inv_date_sk INT64 NOT NULL,
  inv_item_sk INT64 NOT NULL,
  inv_warehouse_sk INT64 NOT NULL,
  inv_quantity_on_hand INT64,
  PRIMARY KEY (inv_date_sk, inv_item_sk, inv_warehouse_sk) NOT
ENFORCED,
  FOREIGN KEY (inv_date_sk) REFERENCES tpcds.date_dim (d_date_sk)
NOT ENFORCED,
  FOREIGN KEY (inv_item_sk) REFERENCES tpcds.item (i_item_sk) NOT
ENFORCED,
  FOREIGN KEY (inv_warehouse_sk) REFERENCES tpcds.warehouse
(w_warehouse_sk) NOT ENFORCED
);

CREATE TABLE `tpcds.customer_demographics`
(
  cd_demo_sk INT64 NOT NULL,
  cd_gender STRING,
  cd_marital_status STRING,
  cd_education_status STRING,
  cd_purchase_estimate INT64,
  cd_credit_rating STRING,
  cd_dep_count INT64,
```

```
  cd_dep_employed_count INT64,
  cd_dep_college_count INT64,
  PRIMARY KEY (cd_demo_sk) NOT ENFORCED
);

CREATE TABLE `tpcds.web_returns`
(
  wr_returned_date_sk INT64,
  wr_returned_time_sk INT64,
  wr_item_sk INT64 NOT NULL,
  wr_refunded_customer_sk INT64,
  wr_refunded_cdemo_sk INT64,
  wr_refunded_hdemo_sk INT64,
  wr_refunded_addr_sk INT64,
  wr_returning_customer_sk INT64,
  wr_returning_cdemo_sk INT64,
  wr_returning_hdemo_sk INT64,
  wr_returning_addr_sk INT64,
  wr_web_page_sk INT64,
  wr_reason_sk INT64,
  wr_order_number INT64 NOT NULL,
  wr_return_quantity INT64,
  wr_return_amt NUMERIC(7,2),
  wr_return_tax NUMERIC(7,2),
  wr_return_amt_inc_tax NUMERIC(7,2),
  wr_fee NUMERIC(7,2),
  wr_return_ship_cost NUMERIC(7,2),
  wr_refunded_cash NUMERIC(7,2),
  wr_reversed_charge NUMERIC(7,2),
  wr_account_credit NUMERIC(7,2),
  wr_net_loss NUMERIC(7,2),
  PRIMARY KEY (wr_item_sk, wr_order_number) NOT ENFORCED,
  FOREIGN KEY (wr_returned_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED,
  FOREIGN KEY (wr_returned_time_sk) REFERENCES tpcds.time_dim
(t_time_sk) NOT ENFORCED,
```

```
  FOREIGN KEY (wr_item_sk, wr_order_number) REFERENCES
tpcds.web_sales (ws_item_sk, ws_order_number) NOT ENFORCED,
  FOREIGN KEY (wr_item_sk) REFERENCES tpcds.item (i_item_sk) NOT
ENFORCED,
  FOREIGN KEY (wr_refunded_customer_sk) REFERENCES tpcds.customer
(c_customer_sk) NOT ENFORCED,
  FOREIGN KEY (wr_refunded_cdemo_sk) REFERENCES
tpcds.customer_demographics (cd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (wr_refunded_hdemo_sk) REFERENCES
tpcds.household_demographics (hd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (wr_refunded_addr_sk) REFERENCES
tpcds.customer_address (ca_address_sk) NOT ENFORCED,
  FOREIGN KEY (wr_returning_customer_sk) REFERENCES tpcds.customer
(c_customer_sk) NOT ENFORCED,
  FOREIGN KEY (wr_returning_cdemo_sk) REFERENCES
tpcds.customer_demographics (cd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (wr_returning_hdemo_sk) REFERENCES
tpcds.household_demographics (hd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (wr_returning_addr_sk) REFERENCES
tpcds.customer_address (ca_address_sk) NOT ENFORCED,
  FOREIGN KEY (wr_web_page_sk) REFERENCES tpcds.web_page
(wp_web_page_sk) NOT ENFORCED,
  FOREIGN KEY (wr_reason_sk) REFERENCES tpcds.reason (r_reason_sk)
NOT ENFORCED
);

CREATE TABLE `tpcds.warehouse`
(
  w_warehouse_sk INT64 NOT NULL,
  w_warehouse_id STRING NOT NULL,
  w_warehouse_name STRING,
  w_warehouse_sq_ft INT64,
  w_street_number STRING,
  w_street_name STRING,
  w_street_type STRING,
  w_suite_number STRING,
```

```
  w_city STRING,
  w_county STRING,
  w_state STRING,
  w_zip STRING,
  w_country STRING,
  w_gmt_offset NUMERIC(5,2),
  PRIMARY KEY (w_warehouse_sk) NOT ENFORCED
);

CREATE TABLE `tpcds.reason`
(
  r_reason_sk INT64 NOT NULL,
  r_reason_id STRING NOT NULL,
  r_reason_desc STRING,
  PRIMARY KEY (r_reason_sk) NOT ENFORCED
);

CREATE TABLE `tpcds.time_dim`
(
  t_time_sk INT64 NOT NULL,
  t_time_id STRING NOT NULL,
  t_time INT64,
  t_hour INT64,
  t_minute INT64,
  t_second INT64,
  t_am_pm STRING,
  t_shift STRING,
  t_sub_shift STRING,
  t_meal_time STRING,
  PRIMARY KEY (t_time_sk) NOT ENFORCED
);

CREATE TABLE `tpcds.catalog_sales`
(
  cs_sold_date_sk INT64,
  cs_sold_time_sk INT64,
```

```
  cs_ship_date_sk INT64,
  cs_bill_customer_sk INT64,
  cs_bill_cdemo_sk INT64,
  cs_bill_hdemo_sk INT64,
  cs_bill_addr_sk INT64,
  cs_ship_customer_sk INT64,
  cs_ship_cdemo_sk INT64,
  cs_ship_hdemo_sk INT64,
  cs_ship_addr_sk INT64,
  cs_call_center_sk INT64,
  cs_catalog_page_sk INT64,
  cs_ship_mode_sk INT64,
  cs_warehouse_sk INT64,
  cs_item_sk INT64 NOT NULL,
  cs_promo_sk INT64,
  cs_order_number INT64 NOT NULL,
  cs_quantity INT64,
  cs_wholesale_cost NUMERIC(7,2),
  cs_list_price NUMERIC(7,2),
  cs_sales_price NUMERIC(7,2),
  cs_ext_discount_amt NUMERIC(7,2),
  cs_ext_sales_price NUMERIC(7,2),
  cs_ext_wholesale_cost NUMERIC(7,2),
  cs_ext_list_price NUMERIC(7,2),
  cs_ext_tax NUMERIC(7,2),
  cs_coupon_amt NUMERIC(7,2),
  cs_ext_ship_cost NUMERIC(7,2),
  cs_net_paid NUMERIC(7,2),
  cs_net_paid_inc_tax NUMERIC(7,2),
  cs_net_paid_inc_ship NUMERIC(7,2),
  cs_net_paid_inc_ship_tax NUMERIC(7,2),
  cs_net_profit NUMERIC(7,2),
  PRIMARY KEY (cs_item_sk, cs_order_number) NOT ENFORCED,
 FOREIGN KEY (cs_sold_date_sk) REFERENCES tpcds.date_dim (d_date_sk)
NOT ENFORCED,
```

```
  FOREIGN KEY (cs_sold_time_sk) REFERENCES tpcds.time_dim
(t_time_sk) NOT ENFORCED,
  FOREIGN KEY (cs_ship_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED,
  FOREIGN KEY (cs_bill_customer_sk) REFERENCES tpcds.customer
(c_customer_sk) NOT ENFORCED,
  FOREIGN KEY (cs_bill_cdemo_sk) REFERENCES
tpcds.customer_demographics (cd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (cs_bill_hdemo_sk) REFERENCES
tpcds.household_demographics (hd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (cs_bill_addr_sk) REFERENCES tpcds.customer_address
(ca_address_sk) NOT ENFORCED,
  FOREIGN KEY (cs_ship_customer_sk) REFERENCES tpcds.customer
(c_customer_sk) NOT ENFORCED,
  FOREIGN KEY (cs_ship_cdemo_sk) REFERENCES
tpcds.customer_demographics (cd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (cs_ship_hdemo_sk) REFERENCES
tpcds.household_demographics (hd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (cs_ship_addr_sk) REFERENCES tpcds.customer_address
(ca_address_sk) NOT ENFORCED,
  FOREIGN KEY (cs_call_center_sk) REFERENCES tpcds.call_center
(cs_call_center_sk) NOT ENFORCED,
  FOREIGN KEY (cs_catalog_page_sk) REFERENCES tpcds.catalog_page
(cp_catalog_page_sk) NOT ENFORCED,
  FOREIGN KEY (cs_ship_mode_sk) REFERENCES tpcds.ship_mode
(sm_ship_mode_sk) NOT ENFORCED,
  FOREIGN KEY (cs_warehouse_sk) REFERENCES tpcds.warehouse
(w_warehouse_sk) NOT ENFORCED,
  FOREIGN KEY (cs_item_sk) REFERENCES tpcds.item (i_item_sk) NOT
ENFORCED,
  FOREIGN KEY (cs_promo_sk) REFERENCES tpcds.promotion (p_promo_sk)
NOT ENFORCED
);
```

```
CREATE TABLE `tpcds.store_returns`
(
  sr_returned_date_sk INT64,
  sr_return_time_sk INT64,
  sr_item_sk INT64 NOT NULL,
  sr_customer_sk INT64,
  sr_cdemo_sk INT64,
  sr_hdemo_sk INT64,
  sr_addr_sk INT64,
  sr_store_sk INT64,
  sr_reason_sk INT64,
  sr_ticket_number INT64 NOT NULL,
  sr_return_quantity INT64,
  sr_return_amt NUMERIC(7,2),
  sr_return_tax NUMERIC(7,2),
  sr_return_amt_inc_tax NUMERIC(7,2),
  sr_fee NUMERIC(7,2),
  sr_return_ship_cost NUMERIC(7,2),
  sr_refunded_cash NUMERIC(7,2),
  sr_reversed_charge NUMERIC(7,2),
  sr_store_credit NUMERIC(7,2),
  sr_net_loss NUMERIC(7,2),
  PRIMARY KEY (sr_item_sk, sr_ticket_number) NOT ENFORCED,
  FOREIGN KEY (sr_item_sk, sr_ticket_number) REFERENCES
tpcds.store_sales (ss_item_sk, ss_ticket_number) NOT ENFORCED,
  FOREIGN KEY (sr_returned_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED,
  FOREIGN KEY (sr_return_time_sk) REFERENCES tpcds.time_dim
(t_time_sk) NOT ENFORCED,
  FOREIGN KEY (sr_item_sk) REFERENCES tpcds.item (i_item_sk) NOT
ENFORCED,
  FOREIGN KEY (sr_customer_sk) REFERENCES tpcds.customer
(c_customer_sk) NOT ENFORCED,
  FOREIGN KEY (sr_cdemo_sk) REFERENCES tpcds.customer_demographics
(cd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (sr_hdemo_sk) REFERENCES tpcds.household_demographics
```

```
(hd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (sr_addr_sk) REFERENCES tpcds.customer_address
(ca_address_sk) NOT ENFORCED,
  FOREIGN KEY (sr_store_sk) REFERENCES tpcds.store (s_store_sk) NOT
ENFORCED,
  FOREIGN KEY (sr_reason_sk) REFERENCES tpcds.promotion
(r_reason_sk) NOT ENFORCED
  );CREATE TABLE `tpcds.customer`
(
  c_customer_sk INT64 NOT NULL,
  c_customer_id STRING NOT NULL,
  c_current_cdemo_sk INT64,
  c_current_hdemo_sk INT64,
  c_current_addr_sk INT64,
  c_first_shipto_date_sk INT64,
  c_first_sales_date_sk INT64,
  c_salutation STRING,
  c_first_name STRING,
  c_last_name STRING,
  c_preferred_cust_flag STRING,
  c_birth_day INT64,
  c_birth_month INT64,
  c_birth_year INT64,
  c_birth_country STRING,
  c_login STRING,
  c_email_address STRING,
  c_last_review_date STRING,
  PRIMARY KEY (c_customer_sk) NOT ENFORCED,
  FOREIGN KEY (c_current_cdemo_sk) REFERENCES
tpcds.customer_demographics (cd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (c_current_hdemo_sk) REFERENCES
tpcds.household_demographics (hd_demo_sk) NOT ENFORCED,
  FOREIGN KEY (c_current_addr_sk) REFERENCES tpcds.customer_address
(ca_address_sk) NOT ENFORCED,
  FOREIGN KEY (c_first_shipto_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED,
```

```
  FOREIGN KEY (c_first_sales_date_sk) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED,
  FOREIGN KEY (c_last_review_date) REFERENCES tpcds.date_dim
(d_date_sk) NOT ENFORCED
);
```

Jeff Curran, Data Science Team Lead, AtScale

Jeff Curran is the Data Science Team Lead at AtScale and has been with the AtScale team for over two years. Jeff has a degree in Physics from Northeastern University and a Master of Business Intelligence and Data Analytics from Carnegie Mellon. Between his academic and professional experience, Jeff has been involved in the Data and Analytics space for over a decade.

ATSCALE