ADS 2024, Project Assignment 1

**Mastering Snowflake: Sentiment Analysis and Performance Experiments**

Github : <https://github.com/MaCoHa/ADS2024-Project_1.git>

Contents

[Introduction 1](#_Toc181710611)

[Experimental Methodology and Results 2](#_Toc181710612)

[Sentiment Analysis using SQL vs UDTF 2](#_Toc181710613)

[Datasets 2](#_Toc181710614)

[Setup 2](#_Toc181710615)

[Results 2](#_Toc181710616)

[Performance Experiments Using TPC-H 3](#_Toc181710617)

[Datasets 3](#_Toc181710618)

[Setup 3](#_Toc181710619)

[Results 3](#_Toc181710620)

[Discussion of Results 4](#_Toc181710621)

[Conclusion 5](#_Toc181710622)

[Appendix 5](#_Toc181710623)

[Sentiment Analysis training time 5](#_Toc181710624)

[Sentiment Analysis testing time 6](#_Toc181710625)

[TPC-H Query 1 Performance Experiments 6](#_Toc181710626)

[TPC-H Query 5 Performance Experiments 7](#_Toc181710627)

[TPC-H Query 18 Performance Experiments 7](#_Toc181710628)

# Introduction

# Experimental Methodology and Results

## Sentiment Analysis using SQL vs UDTF

### Datasets

for the sentiment analysis we were given two datasets one big for training the model and one for testing the model. Both datasets were large collections of yelp reviews which contained the review in text and the star rating between 0 to 4 stars. Before using each dataset in the models I first removed all datasets that did not have 0 or 4 stars, then I clean the text in each set so the text did not contain any special characters or numbers. that only reviews with 0 or 4 stars remained. After that I could use them to train or test my model. Both the clean up and my SQL and UDTF can be found on my [GitHub](https://github.com/MaCoHa/ADS2024-Project_1/tree/main/Sql_code).

### Setup

To run the benchmark of my two models I created a python script, the script used an extension created by Snowflake to run queries through python. Through this extension I could set up connection parameters such as not using cached results. With that I could send the queries I used for training and testing my SQL and UDTF models.

For benchmarking I ran

* 4 query groups [ SQL training, SQL testing, UDTF training, UDTF testing]
* each group was run 3 times

To get the time I record the query id of each query and then queried Snowflake query history for the query, warehouse and elapsed time in seconds and milliseconds. Then I would sum the results of all queries inside a group before dividing the results by 3. For SQL training and SQL testing which consisted of multiple queries in each group I also summed their results. an important note that all queries were run on a X-small warehouse in snowflake.

### Results

When running my models the SQL had 63.9% accuracy while UDTF had 71% accuracy. Despite using the same formulars and structure for creating the model. I suspect that the difference might be a difference in pythons’ way of handling floating point numbers compared to SQL.

For sentiment analysis I ran both the training and testing of both the SQL and UDTF 3 times each. I then extracted the data time of each query from snowflakes query history. This allowed me to calculate the average time for each query. I have then plotted the training and testing of the SQL and UDTF model against each other. Figure 1 is a bar chart comparing the SQL and UDTF training time where Figure 2 is a bar chart comparing SQL and UDTF testing / prediction time.

Figure 1 training time for the UDTF and SQL Sentiment Analysis. Large figure in appendix

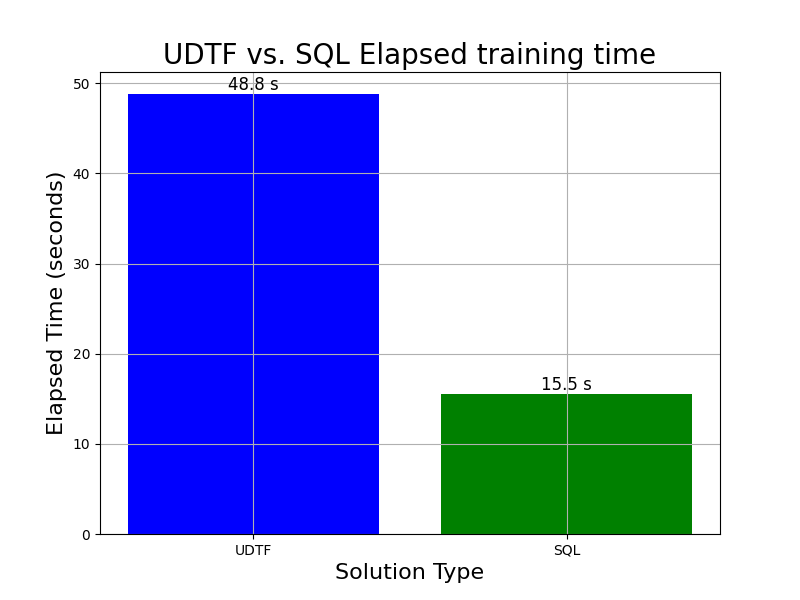
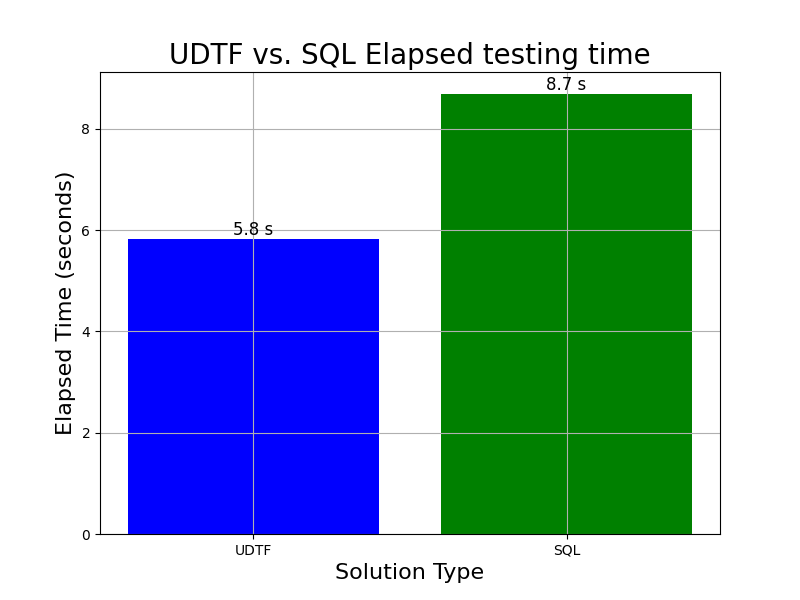


Figure 2 testing time for the UDTF and SQL Sentiment Analysis. Large figure in appendix

## Performance Experiments Using TPC-H

### Datasets

The datasets used for my TPC-H benchmarks were available in Snowflake which have shared databases available. Snowflake provides 4 different sizes of TPC-H sample databases each with an scale factor of SF1, SF10, SF100 and SF1000[[1]](#footnote-1).

### Setup

We were told to run experiments with queries 1,5 and 18 from the TPC-H sample queries which I found [here](https://docs.deistercloud.com/content/Databases.30/TPCH%20Benchmark.90/Sample%20querys.20.xml?embedded=true) but had to rewrite them to work with Snowflake syntax. For benchmarking I set up a python script that used a Snowflake connector where I could send a query and then query the query history table for the results of the first query. When running the benchmarks I had the following parameters.

Databases: [SF1, SF10, SF100, SF1000]

Warehouse: [XSMALL, SMALL, MEDIUM, LARGE]

Queries: [TPC-H-Q1, TPC-H-Q5, TPC-H-Q18]

I ran benchmarks where I had combinations of all parameters and each of these benchmarks were repeated 3 times. Warehouse in snowflake are the different sizes of computing clusters meaning **LARGE** have more computing than a **XSMALL.**

### Results

For each query I plotted graphs showing the time each query took on each database with different warehouses. Figure 3, Figure 4 and Figure 5 each shows the elapsed time for the queries on each scale factor database and warehouse size. By looking at each figure we see that the **Large** and **Medium** warehouse have a small improvement or close to similar time between scale factor 1 and scale factor 10. This I likely because that on scale factor 1 large and medium never uses all of their processing power and spend more time splitting the work between threads, because of the small data size. but on scale factor 10 they use all their processing power and possibly delegate the work more efficiently.

Figure 4 showing the execution time for tpc-h Query 5 with different Snowflake warehouses and data with different scale factors. Large figure in appendix

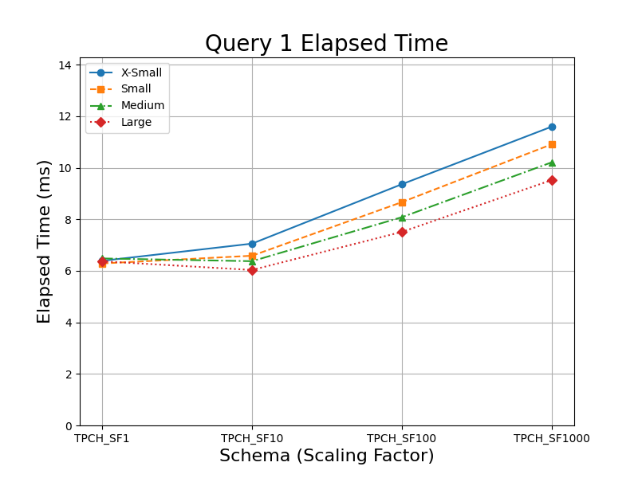


Figure 3 showing the execution time for tpc-h Query 1 with different Snowflake warehouses and data with different scale factors. Large figure in appendix



Figure 5 showing the execution time for tpc-h Query 18 with different Snowflake warehouses and data with different scale factors. Large figure in appendix

# Discussion of Results

Looking at Figure 1 or training time it clear that SQL is faster than UDTF by approximately 33,3 s. the reasons for SQL being faster is very likely due to the query optimizer and other optimizations. Snowflake is built around using SQL hence they have likely bult their service to optimize and parallelized SQL queries, when possible, to make I more efficient. Where with my UDTF which is code in python I likely not being optimized or parallelized by Snowflake. Since the training of the Sentiment model requires reading and processing a lot of data this favors the optimizations and parallelism.

Another reason for the difference in performance is that snowflake needs to create an environment to run the UDTF, then parse the SQL data table and when the UDTF is done read the output and parse the result back into an SQL table. These processes could also take some time depending on the implementation in snowflake.

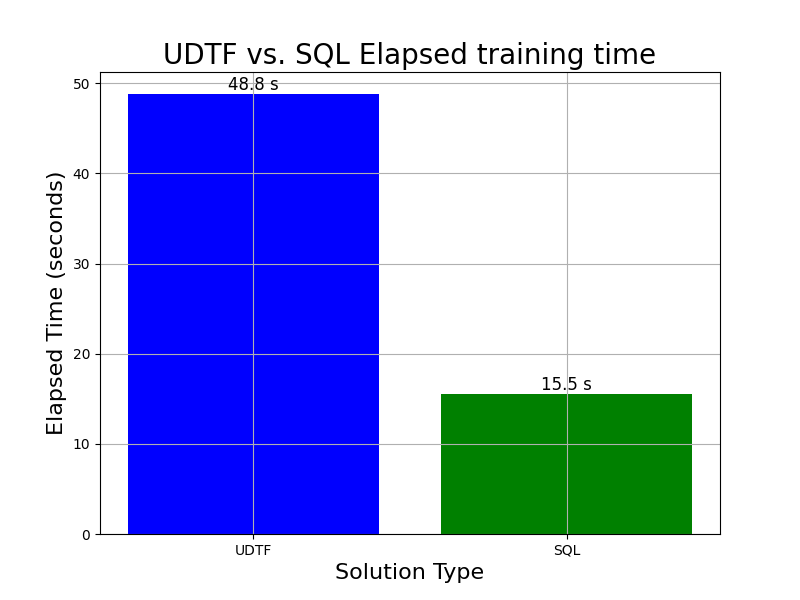
Looking at Figure 2 my UDTF is a little faster than the SQL with approximately 2,9 s. I still believe that Snowflake optimizes the query like before but this time there are a lot of look ups in the data to find the right values used in the calculation of an prediction. Here python is likely superior since I store the data needed in a dictionary making look up during calculation much faster since it a lot of single lookups. Since the SQL have to filter through the tables several times to find single values it likely slows it down a bit. It possible that if the SQL was allowed to use cache results it could perform better since some words are reused a lot in English sentences.

# Conclusion

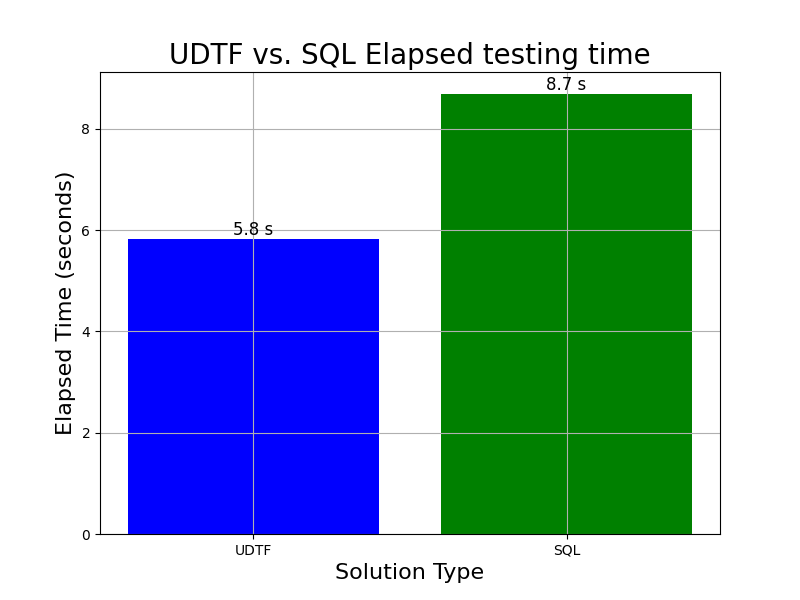
SQL works overall better than UDTF when working with large amount of data since snowflake optimizes and paralyzed queries to, I’m prove performance whereas if you use the correct data structures UDTF can in some situations work better than SQL, if there is many single lookups and less large amount of data work and handling.

# Appendix

## Sentiment Analysis training time



## Sentiment Analysis testing time



## TPC-H Query 1 Performance Experiments



## TPC-H Query 5 Performance Experiments



## TPC-H Query 18 Performance Experiments



1. https://docs.snowflake.com/en/user-guide/sample-data-tpch [↑](#footnote-ref-1)