Database Performance Comparison

(Microsoft SQL Server vs. Snowflake)

By Rehan Khan

Summary

Efficiency to analytical insights is always at the forefront of any organization. The infrastructure plays an important role in ensuring time to insights are streamlined. In my capstone project, I meticulously analyzed and compared the performance and functionality of Microsoft SQL Server On-Premises and a Snowflake cloud environment containerized within Azure, focusing on the ETL process of the popular Adventure Works dataset. Leveraging my skills, I diligently executed the ETL procedures for both systems and seamlessly integrated them into my Python notebook. By connecting both environments to the same notebook, I conducted an in-depth comparative analysis, meticulously scrutinizing factors such as efficiency, scalability, and ease of use. Through this comprehensive evaluation, I unearthed valuable insights into the strengths and weaknesses of each platform, contributing to a deeper understanding of their respective capabilities and applicability in real-world scenarios.

Data used for Capstone: https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms

Section 1: Data Wrangling

We obtained the adventure works file from the Microsoft website, restoring it as a .bak file to both environments. To facilitate analysis, we first wrote a query that we would use to join three tables to begin our comparison using a Jupyter Notebook to processes our data using the Python programming language. Find 2 sample schemas below of the Adventure Works database below:

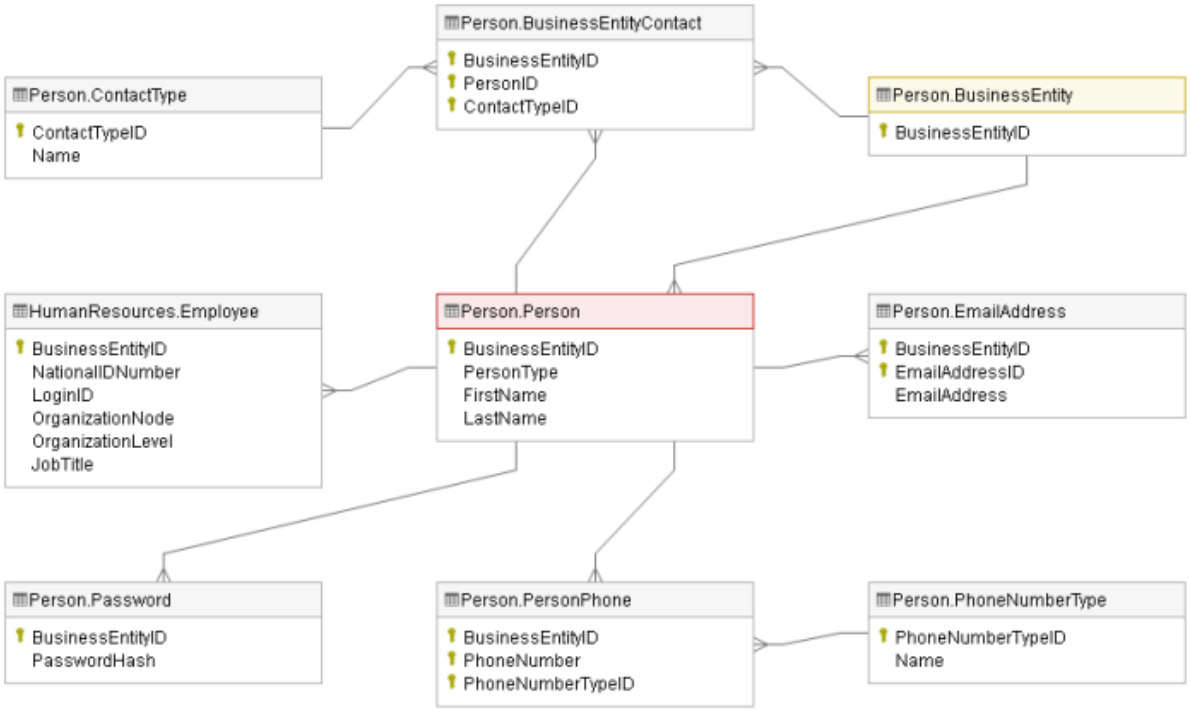

Figure 1: Names and addresses of individual customers, vendors and employees.

This dataset was ingested in both Snowflake and SQL Server for the sole purpose of comparing the two data warehouse systems. We started our data wrangling by writing a simple query from the Adventure Works schema and connecting to our SQL and Snowflake instance respectively.

```
# SQL query
query = """ SELECT
    SOH.SalesOrderID,
    SOH.OrderDate,
    P.ProductID,
    P.Name AS ProductName,
    SOD.OrderQty,
    SOD.UnitPrice,
    SOD.LineTotal
FROM
Sales.SalesOrderHeader AS SOH
JOIN Sales.SalesOrderDetail AS SOD ON SOH.SalesOrderID = SOD.SalesOrderID
JOIN Sales.Customer AS C ON SOH.CustomerID = C.CustomerID
JOIN Production.Product AS P ON SOD.ProductID = P.ProductID
ORDER BY SOH.OrderDate DESC """
```

Figure 2 Query used to compare the two systems.

We initialized two different dataframes for each datawarehouse. The dataframes were named “snowflake_df” for snowflake and “sql_df” for Microsoft SQL Server respectively. It was noted that in the creation of each dataframe the Snowflake dataframe took significantly less time to load than SQL Server dataframe (7.8 seconds for SQL Server versus 0.6 seconds for Snowflake).

Section 2: Exploratory Data Analysis

In conducting exploratory data analysis, a comparative examination of histograms extracted from both Microsoft SQL Server and Snowflake was performed to discern potential variations in data distribution and granularity across the platforms. This analysis provided insights into the distinct characteristics of data representation and storage mechanisms, shedding light on any disparities or similarities in the datasets. Additionally, the investigation unveiled the top 5 most ordered products, furnishing crucial insights into consumer preferences and market demand within the dataset. Such findings serve as pivotal points of reference for further analytical endeavors and decision-making processes in data-driven contexts. See below an example of histograms that were created to examine the distribution of order dates. The bars in blue represent a histogram from Snowflake and grey represents Microsoft SQL Server.

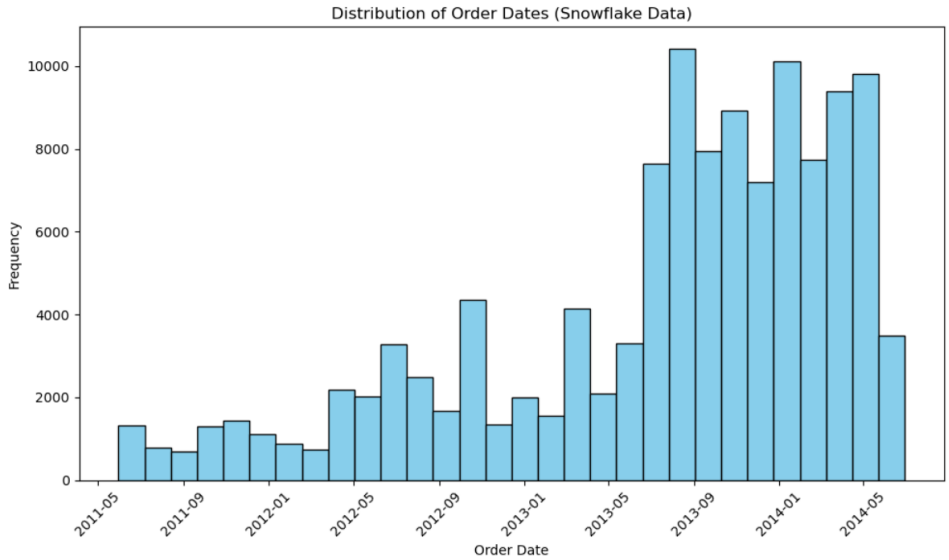

Figure 3 Distribution of Order Dates (Snowflake)

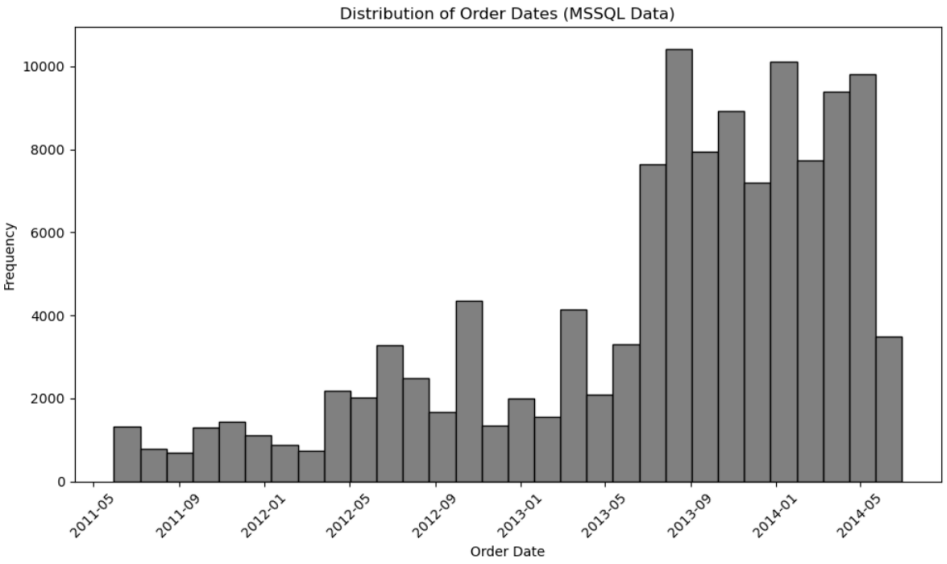

Figure 4 Distribution of Order Dates MSSQL

Section 3: Pre-processing and Modeling

Effective data preprocessing stands out as a vital stage in readying raw data for utilization in any machine learning algorithm. Given that the existing data was deliberately generated, it remains relatively clean without any instances of missing values, obviating the need for removal or imputation procedures. However, a crucial step involves transforming categorical data with string values, necessitating the encoding of non-numeric values into numeric equivalents. This transformation becomes imperative, particularly for models like logistic regression and neural networks that exclusively process numeric inputs.

For this step, I selected only relevant values for our dataframe. The parameters selected were:

- SalesOrderID
- OrderDate
- ProductID
- UnitPrice
- LineTotal
- OrderYear

The Line Total was the target variable that we wanted to predict. We split the data into training and test sets using the Sci-kit learn “train_test_split” function, using a test size of 20 percent and random state of 42. The code snippet below shows how the training and test sets were created.

```
# Selecting relevant columns
selected_columns = ['SalesOrderID', 'OrderDate', 'ProductID', 'UnitPrice', 'LineTotal', 'OrderYear']
data = snowflake_df[selected_columns]

# Split data into features (X) and target variable (y)
X = data[['SalesOrderID', 'ProductID', 'UnitPrice', 'LineTotal', 'OrderYear']]
y = data['LineTotal']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 5 Creation of Training and Test sets for model building.

We evaluated two different models to test for the prediction of our target variable. Those two models were Random Forest and Linear Regression. Random Forest is an ensemble learning algorithm used for both classification and regression tasks. It operates by constructing a multitude of decision trees during training and outputs the mode of the classes (classification) or the mean prediction (regression) of the individual trees.

Logistic regression, a classification model, predicts the probabilities of a sample observation being labeled as class value 1. If the probability is equal to or greater than 0.5, the outcome is '1'; otherwise, it is '0'. In this context, we instantiated the Logistic Regression object from the scikit-learn library and fitted it with our training data. Following the fitting process, we generated predictions using features from the test data.

Both models were trained and tested using both the data from our SQL Server Instance and Snowflake Instance to ensure that all values were relatively similar.

Section 4: Conclusion

We found that the models were accurately able to predict values from both AdventureWorks datasets for Snowflake and Microsoft SQL Server respectively. To gain a holistic understanding of

how the two data warehouses differed we created a dataframe that compared and contrasted different features of two. These features for comparison were:

- Scalability
- Cost
- Maintenance
- Performance
- Security
- Speed
- Integrations

A snippet of a dataframe that was used to make the comparison can be found below:

	Aspect	Snowflake	SQL_Server_On_Premises
0	Scalability	Highly scalable, elastic, pay-as-you-go model	Requires hardware procurement and setup, limited scalability
1	Cost	Pay for what you use, may be more cost-effective for small to medium-sized workloads	Upfront hardware and software costs, may be expensive for large workloads
2	Maintenance	Managed service, minimal maintenance required	Requires in-house expertise for maintenance, updates, and backups
3	Performance	Optimized for cloud, may have faster performance for certain workloads	Depends on hardware configuration, may be slower for large workloads
4	Security	Built-in security features, granular access controls	Requires additional security measures, regular updates
5	Time to Load Dataframe (Speed)	0.6 Seconds	16.8 Seconds
6	Connection to Python	Built-In Snowflake Connector (very easy)	Build Custom Connection through SQL Alchemy (easy)

Figure 6 Comparison table for SQL Server vs. Snowflake

To conclude, based on the findings from Figure 6, we felt that the Snowflake data warehouse was a better overall solution when comparing it to Microsoft SQL Server. The reason we felt this way was mainly due to the overall ease of use, scalability, speed and cost difference when outsourcing the management of the cloud environment. Cloud data warehouses are popular because you don't have to worry about managing the server in-house and it can seamlessly allow you to add more data to your cloud environment instantaneously.