# 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*

## Background

In the realm of MLOps (Machine Learning Operations), the significance of a robust data warehousing solution cannot be overstated. A data warehouse serves as the backbone for managing, storing, and processing vast amounts of data essential for training, deploying, and maintaining machine learning models efficiently. Microsoft SQL Server and Snowflake are two prominent data warehouse tools, each offering distinct advantages. SQL Server, a row-level database, excels in transactional processing and traditional relational database operations. Its structured storage format is well-suited for transactional consistency and granular data access, making it ideal for applications where individual records are frequently updated or inserted. On the other hand, Snowflake adopts a columnar database architecture, optimizing data storage and retrieval for analytical workloads. By organizing data into columns rather than rows, Snowflake enhances query performance, particularly for analytical queries involving aggregations, filtering, and data analytics. This differentiation underscores the importance of selecting the appropriate data warehousing solution tailored to the specific needs and objectives of MLOps initiatives. Whether prioritizing real-time transactional processing or analytical performance, the choice between SQL Server and Snowflake profoundly impacts the efficiency, scalability, and agility of machine learning workflows within the MLOps ecosystem.

**Analytical Benchmarking**

To get a better understanding of both data warehouse products, we went through a full analytical cycle from data wrangling to model-building using a Python notebook. The reason for doing this was ensuring ease of use, speed and integrations for both products. To begin the process, we had to choose a dataset we would use to compare the ETL process for both.

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:

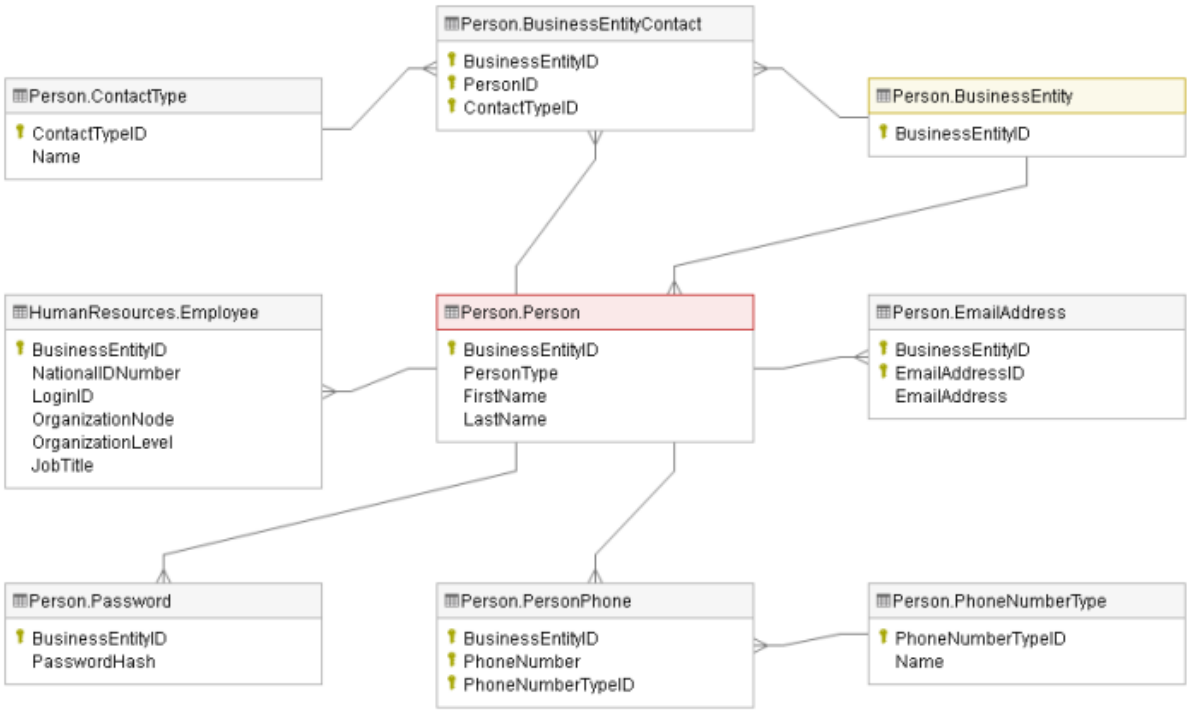


*Figure 1: Names and addresses of individual customers, vendors and employees.*

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).

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.

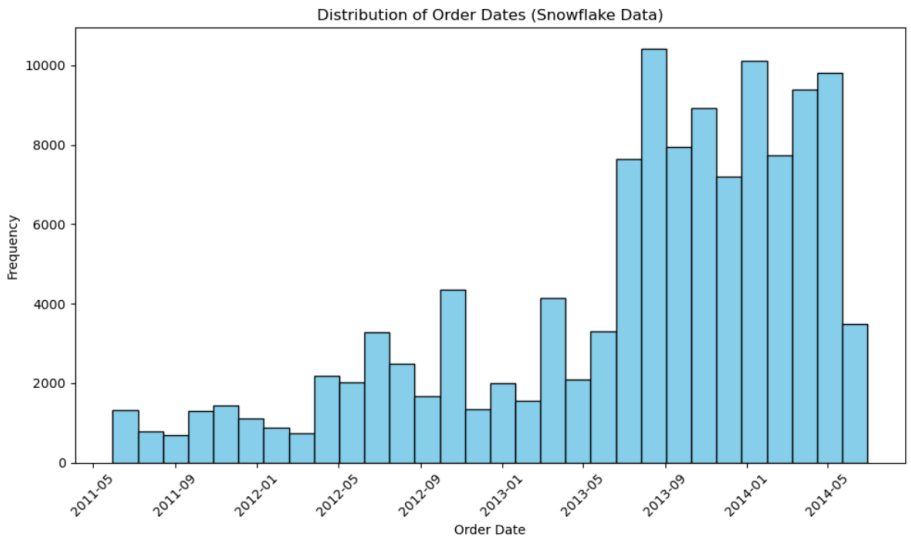


*Figure 3 Distribution of Order Dates (Snowflake)*

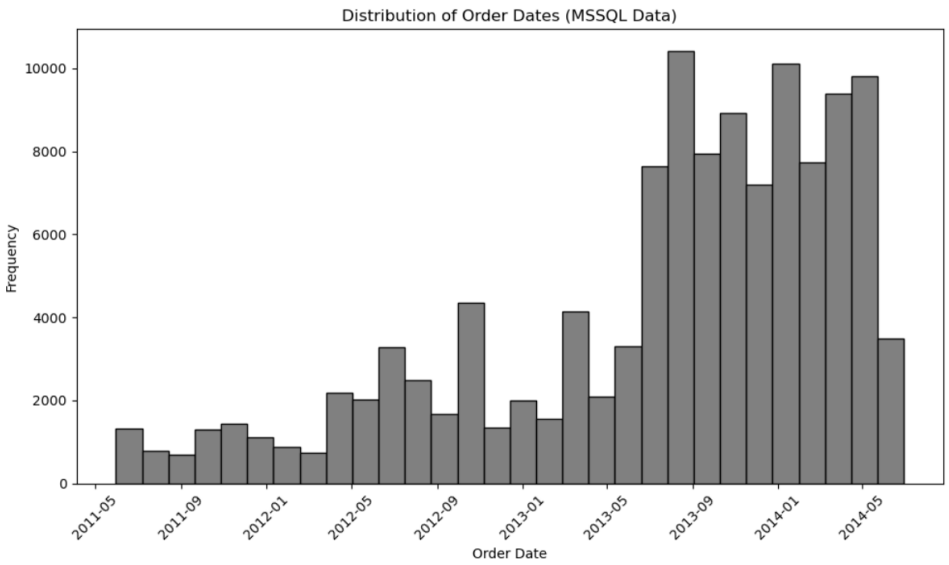


*Figure 4 Distribution of Order Dates MSSQL*

**Speed and Cost Comparisons**

When comparing Snowflake and SQL Server, several key factors come into play, including speed, cost, and ease of use. In terms of speed, Snowflake, with its distributed architecture and columnar storage format, often outperforms SQL Server, especially for analytical workloads involving large datasets. SQL Server, while capable, may face scalability challenges and require optimization efforts for comparable performance. Concerning cost, Snowflake operates on a subscription-based model, offering scalability and flexibility without the need for upfront hardware investments. In contrast, SQL Server typically involves higher initial setup costs and ongoing maintenance expenses, particularly when considering hardware infrastructure and licensing fees. Regarding ease of use, Snowflake's cloud-native platform and managed services simplify deployment, scaling, and administration tasks, making it more accessible for users without extensive database management expertise. SQL Server, while widely used and familiar to many, may require more manual configuration and maintenance efforts, especially in on-premises deployments. Ultimately, the choice between Snowflake and SQL Server depends on factors such as workload requirements, budget constraints, and organizational preferences for speed, cost-effectiveness, and ease of management.

**Conclusion**

In conclusion, Snowflake emerges as a superior choice for ML Ops compared to SQL Server across several critical dimensions. Firstly, Snowflake's unparalleled query performance, facilitated by its distributed architecture and columnar storage format, ensures efficient processing of large-scale analytical workloads essential for machine learning operations. Moreover, Snowflake's inherent scalability enables seamless expansion of computational resources to accommodate growing data volumes and evolving business needs, a crucial aspect in the dynamic landscape of ML Ops. From a cost-efficiency perspective, Snowflake's pay-as-you-go pricing model eliminates the need for upfront hardware investments and offers predictable, transparent billing based on actual usage,

mitigating financial risks associated with underutilized resources. Additionally, Snowflake's cloud-native design and managed services alleviate the administrative burden on ML Ops teams, allowing them to focus on strategic initiatives rather than infrastructure management. Taken together, Snowflake's superior query performance, scalability, cost efficiency, and ease of use make it the preferred data warehousing solution for powering ML Ops workflows, driving innovation, and delivering actionable insights at scale.

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 5 Comparison table for SQL Server vs. Snowflake*

To conclude, based on the findings from Figure 7, 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.