Interim Project

Project Topic -

Enhancing Hive by performing partitioning and Bucketing

Project dataset -

OnlineRetail.csv

Columns -

* invoice\_no: This column stores the unique identifier for each invoice or transaction.
* stock\_code: This column stores the stock code or identifier for each item sold in the transaction.
* item: This column stores the name or description of the item sold in the transaction.
* quantity: This column stores the quantity of the item sold in the transaction.
* invoice\_date: This column stores the date of the invoice or transaction.
* unit\_price: This column stores the unit price of the item sold in the transaction.
* customer\_id: This column stores the unique identifier for each customer.
* country: This column stores the country associated with the transaction.

Technology used -

* Hadoop Ecosystem
* Hive - Data warehouse infrastructure built on top of Hadoop for querying and analyzing large datasets.
* HDFS - Distributed file system for storing large datasets.

Hive and why we need it in Hadoop -

* SQL Interface to Hadoop: HiveQL provides a familiar SQL-like interface to interact with Hadoop data, making it accessible to users who are already familiar with SQL. This lowers the barrier to entry for data analysts and SQL developers who want to leverage Hadoop's capabilities without learning new programming paradigms.
* Scalability: Hive is designed to handle large-scale datasets stored in Hadoop's distributed file system. It leverages the distributed processing power of Hadoop to execute queries in parallel across multiple nodes, enabling scalable data processing for big data analytics.
* Optimized for Batch Processing: Hive is optimized for batch processing of large volumes of data. It is well-suited for running complex analytical queries, data transformations, and ETL (Extract, Transform, Load) operations on structured and semi-structured data stored in Hadoop.
* Integration with Hadoop Ecosystem: Hive seamlessly integrates with other components of the Hadoop ecosystem, such as HDFS, Apache HBase, Apache Spark, Apache Tez, and Apache YARN. This allows users to leverage the capabilities of these tools for various data processing tasks while using Hive as the query interface.

How do we optimize Hive performance by using Partitioning and Bucketing -

Partitioning:

Partitioning involves dividing data into multiple partitions based on the values of one or more columns. This helps in reducing the amount of data that needs to be scanned during query execution.

Benefits of Partitioning:

* Data Pruning: Queries only need to process data within specific partitions, reducing the amount of data scanned.
* Parallel Processing: Partitioned data can be processed in parallel across multiple nodes, improving query performance.

Bucketing:

Bucketing involves grouping data into a fixed number of buckets based on the hash value of a column. This helps in evenly distributing data and improving query performance for certain types of queries, especially joins and aggregations.

Benefits of Bucketing:

* Data Locality: Data within each bucket is stored together, improving locality for joins and aggregations.
* Sampling: Bucketing can be used for data sampling and approximate query processing.

Optimizing Hive Performance Using Partitioning and Bucketing:

* Partitioning and Bucketing Together: Use both partitioning and bucketing for optimal performance. Partition data by frequently used filtering columns and bucket data within each partition based on join or aggregation keys.
* Choose Appropriate Columns: Select partitioning and bucketing columns based on query patterns and access patterns. Consider columns used in filtering, joining, and aggregating data.
* Monitor and Tune: Regularly monitor query performance and adjust partitioning and bucketing strategies based on changing data and query patterns.

Solution Workflow -

1. Data Ingestion:Import the Online Retail data from the local system into HDFS.
2. Data Storage:Store ingested data in HDFS for further processing and analysis.
3. Optimization using Partitioning and Bucketing :Implement partitioning and bucketing techniques using Apache Hive to optimize data storage and query performance.
4. Data Analysis & Querying:Develop SQL queries to extract insights from the partitioned , bucketed dataset and original dataset.