Implementing Slow-Changing Dimensions in a Data Warehouse using Hive and Spark

Business Overview:

In today's rapidly evolving market landscape, businesses face the challenge of maintaining an accurate and comprehensive historical record of their data. The primary goal of this project is not just to implement Slowly Changing Dimensions (SCD) within a data warehouse architecture but to revolutionize the way businesses capture and store changes over time in key business data, such as customer information, product details, and sales transactions. This will be achieved by utilizing advanced tools like Apache Hive and Apache Spark, marking a significant step forward in our data management capabilities.

SCD techniques are critical and indispensable for businesses that require a historical perspective for accurate reporting, trend analysis, and decision-making. These dimensions are not just tools but strategic assets that allow a data warehouse to keep up-to-date records and maintain previous versions of data entities. This capability is particularly vital for sectors such as retail, finance, and healthcare, where understanding changes over time is beneficial and can provide strategic insights into customer behavior, product performance, and operational efficiency, potentially shaping the future of these industries.

Moreover, implementing SCD helps in compliance with regulatory requirements that mandate the retention and auditing of historical data. It enhances the organization's ability to track and analyze the lifecycle of data elements, from creation to modification, providing a transparent audit trail.

The project will focus on deploying a robust data processing and analysis platform supporting different SCD types to handle various data update scenarios efficiently. By leveraging Hive for scalable data storage and Spark for real-time data processing, the solution will offer a high-performance, scalable, and flexible environment capable of handling complex data transformations and large volumes of data.

The result will empower the business to make more informed decisions based on comprehensive data insights, ensure higher data quality and accuracy, and improve overall data governance and compliance. This strategic approach to data management will provide the business with a competitive edge by enabling a deeper understanding of critical business trends and patterns over time.

The primary objective is to enhance the data warehouse's ability to handle changes in data over time, which includes capturing changes in customer information, product details, and other critical business entities. This will allow the organization to view historical and current data within the same framework.

Aim:
To build a robust data warehousing solution incorporating SCD to handle data changes efficiently and effectively. The project will leverage Hive and Spark for data storage and processing, ensuring the data warehouse can operate with high performance and scalability.

Dataset Description:
The project will use a subset of the AdventureWorks dataset, focusing on entities like customers, products, and sales orders. These entities are prone to changes over time, making them ideal candidates for demonstrating the effectiveness of SCD.

- Customer: Tracks changes in customer details like address, contact info, and status.
- Product: Monitors modifications in product specifications, categories, and pricing.
- Sales Order: Captures updates in order statuses, billing, and shipping details.

Tech Stack:
Languages: SQL for data manipulation and Scala for Spark data processing scripts.
Services: AWS RDS, AWS EMR
Frameworks: Hadoop, Hive, Spark, Sqoop, Hue

AWS RDS:
Amazon Relational Database Service (AWS RDS) simplifies the setup, operation, and scaling of a relational database in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks, freeing you to focus on your applications and business. AWS RDS offers several database instance types and can run popular engines like Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database, and SQL Server. Users benefit from the capability of scaling the computational and storage resources with only a few mouse clicks or an API call, often with no downtime. RDS makes it easy to use replication to enhance availability and reliability for production workloads.

AWS EMR:
Amazon Elastic MapReduce (EMR) is a cloud-native big data platform that allows the processing vast amounts of data quickly and cost-effectively across resizable clusters of Amazon EC2 instances. It supports various big data frameworks, including Apache Hadoop, Spark, HBase, and Presto, making running ETL, interactive analytics, data processing, and machine learning applications easier. EMR is optimized for cloud efficiency, automatically scaling to meet workload demands and allowing users to take advantage of lower-cost spot instances. It integrates seamlessly with AWS services like S3, RDS, and DynamoDB for enhanced data storage and database capabilities.

Hadoop:
Apache Hadoop is an open-source software framework for the storage and large-scale processing of datasets on clusters of commodity hardware. It is designed to scale from a single

server to thousands of machines, each offering local computation and storage. Hadoop splits files into large blocks and distributes them across nodes in a cluster. It then uses the MapReduce programming model to process the data in parallel. This approach provides high fault tolerance, efficiency, and scalability. As the foundation of many other big data platforms, Hadoop is integral for managing the global data explosion businesses face.

Hive:
Apache Hive is a data warehouse software project built on top of Apache Hadoop for providing data query and analysis. Hive allows writing SQL-like queries converted into MapReduce, Tez, or Spark jobs, making it a crucial tool for enabling data summarization, query, and analysis. Designed to handle large datasets stored in Hadoop's HDFS and compatible file systems such as Amazon S3 filesystem, Hive is particularly useful for performing complex data analysis and transformations over large data volumes. It also supports database transactions, enabling users to perform insert, update, and delete operations.

Spark:
Apache Spark is a unified analytics engine for big data processing with built-in modules for streaming, SQL, machine learning, and graph processing. Spark provides high-level APIs in Java, Scala, Python, and R and an optimized engine that supports general computation graphs for data analysis. It can run in Hadoop clusters through YARN or Spark's standalone mode, and it can process data in HDFS, HBase, Cassandra, Hive, and any Hadoop format. It is designed to perform batch processing (similar to MapReduce) and new workloads like streaming, interactive queries, and machine learning.

Sqoop:
Apache Sqoop is a tool that transfers data between Hadoop and relational databases or mainframes. You can use Sqoop to import data from a relational database management system (RDBMS) such as MySQL or Oracle into the Hadoop Distributed File System (HDFS), transform the data in Hadoop MapReduce, and then export the data back to an RDBMS. Sqoop automates most of this process, relying on the database to describe the schema for importing the data. Sqoop uses a connector-based architecture that provides plugins to connect to various RDBMS databases.

Hue:
Hue, or Hadoop User Experience, is an open-source web interface for Hadoop that supports a suite of data access applications such as a browser for HDFS, a job tracker interface, and a full-featured query editor. Developed by Cloudera, Hue makes it easier for users to interact with Hadoop ecosystem components like Hive, Pig, and Spark via a user-friendly interface. It allows users to query data, browse the file system, manage jobs, and develop SQL-like scripts to simplify and accelerate Hadoop usage across a wider audience within a company, thereby democratizing access to the power of distributed computing.

Implementation Details:

1. **Designing SCD Models**: Define the SCD type for each data entity based on the business requirements—Type 1 for overwriting records, Type 2 for keeping full history, and Type 3 for storing previous and current values.
2. **Data Processing Workflows**: Develop Spark scripts to transform and process data as per the SCD logic. This includes extracting data from source systems, applying transformations, and loading the processed data into Hive tables.
3. **Hive Table Configuration**: Configure Hive tables to store historical data and handle SCD. This involves schema design, partitioning strategies, and indexing to optimize data retrieval.
4. **Data Validation and Auditing**: Implement data validation checks to ensure the data's integrity and accuracy. Set up auditing mechanisms to track data changes and process executions.
5. **Performance Optimization**: Utilize Spark's in-memory processing to enhance performance. Optimize Hive queries by tuning configurations, using appropriate file formats, and indexing.

Key Learning Takeaways:

- Getting the overview of the project
- Understanding DataWarehousing using Hive
- Understanding Slowly Changing Dimensions
- Types of slow-changing dimension
- Understanding Parquet and ORC file formats
- Setting up the AdventureWorks Dataset in AWS RDS
- Configuring AWS EMR cluster for Big Data Services
- Transferring the data to Hive using Scoop
- Denormalizing the Data for data analysis
- Saving as Parquet Data commands and running scoop jobs
- Viewing the tables created in Hive using Hue
- Understanding the Changing Dimensions in customer demographics
- What is ELT and ETL, similarities, differences, and their use
- Introduction to Data Lake is a Storage Repository
- Transformation for SCD Type-1 on Credit Card Table
- Tuning and Configuring Hive for SCD
- Implementing SCD 2 & 4 in Hive and Spark