HIVE

What is Hive ?

Hive is a data warehouse system which is used to analyze structured data. It is built on the top of Hadoop. It was developed by Facebook.

Hive provides the functionality of reading, writing, and managing large datasets residing in distributed storage. It runs SQL like queries called HQL (Hive query language) which gets internally converted to MapReduce jobs.

Using Hive, we can skip the requirement of the traditional approach of writing complex MapReduce programs. Hive supports Data Definition Language (DDL), Data Manipulation Language (DML), and User Defined Functions (UDF).

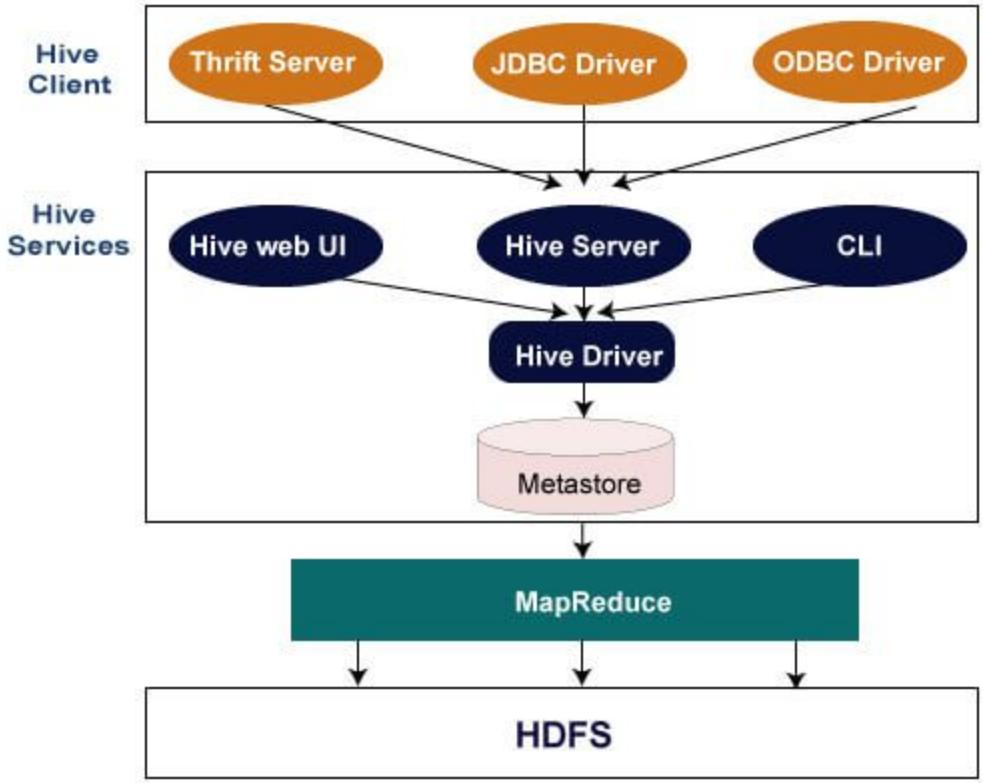
Features of Hive :

- It provides SQL-like queries (i.e., HQL) that are implicitly transformed to MapReduce or Spark jobs.
- It is capable of analyzing large datasets stored in HDFS.
- It allows different storage types such as plain text, RCFile, and HBase.
- It uses indexing to accelerate queries.
- It can operate on compressed data stored in the Hadoop ecosystem.
- It supports user-defined functions (UDFs) where user can provide its functionality.
- Hive is fast and scalable.

Limitations of Hive :

- Hive is not capable of handling real-time data.
- It is not designed for online transaction processing.
- Hive queries contain high latency.

Hive Architecture :



Hive Data Types :

- Integer Types : TINYINT, SMALLINT, INT, BIGINT
- Decimal Types : FLOAT, DOUBLE
- Date/Time Stamp : TIMESTAMP, DATE
- String Types : STRING, VARCHAR, CHAR
- Complex Types : STRUCT, MAP, ARRAY

Hive - CREATE DATABASE :

- hive> show databases;
- hive> create database demo(databasename); // Each database must contain a unique name. Otherwise, it will show an Execution error.

- hive> create a database if not exists demo(database name); // Suppress the warning generated by Hive on creating the database with the same name.
- hive> create database demo WITH DBPROPERTIES ('creator' = 'Gaurav Chawla', 'date' = '2019-06-03'); // Hive also allows assigning properties with the database in the form of key-value pairs.
- hive> describe database extended demo; // retrieve the information associated with the database

Hive - DROP DATABASE :

- hive> drop database database_name;

Hive - CREATE TABLE :

- Two types of tables: two types of table: - Internal table and External table
- The **internal tables** are also called managed tables as the lifecycle of their data is controlled by the Hive. By default, these tables are stored in a subdirectory under the directory defined by hive.metastore.warehouse.dir (i.e. /user/hive/warehouse). The internal tables are not flexible enough to share with other tools like Pig. If we try to drop the internal table, Hive deletes both table schema and data.
- **Syntax:** hive> create table if not exists demo.employee (Id int, Name string , Salary float) row format delimited fields terminated by ',' ;
- The external table allows us to create and access a table and data externally. The **external** keyword is used to specify the external table, whereas the **location** keyword is used to determine the location of loaded data. As the table is external, the data is not present in the Hive directory. Therefore, if we try to drop the table, the metadata of the table will be deleted, but the data still exists.
- **Syntax:** hive> create external table emplist (Id int, Name string , Salary float) row format delimited fields terminated by ',' location '/HiveDirectory';

Hive - LOAD DATA :

- Once the internal table has been created, the next step is to load the data into it. So, in Hive, we can easily load data from any file to the database.
- **Syntax:** LOAD DATA INPATH '<path to the source file>' INTO TABLE <database name>.<table name>;

PROBLEM STATEMENT:

1. Use the data present on this site: https://api.covid19india.org/raw_data.json

2. Save the data to Azure BLOB Storage using Data Bricks.

3. Preprocess or select the required data from the given data (Use Python + Hive).

4. Count of males and females affected by Covid.

5. Find out which city is most affected.

6. Calculate Recovery rate of each city or state.

7. Calculate the above and Store the cleaning data on Azure BLOB Storage.

SOLUTION:

Step 1: Upload the data on Databricks

- Create a cluster in Databricks and upload data(csv file) via the create table option.

  https://www.youtube.com/watch?v=Y_GPag19tZY&feature=emb_title

Step 2: Preprocess the data on Databricks

- First convert the spark dataframe into pandas dataframe.

  https://docs.databricks.com/spark/latest/spark-sql/spark-pandas.html

- Clean and visualize the dataset.

  https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d

Step 3: Mount Azure Blob storage containers to DBFS

- Mount Azure blob Storage to Databricks.

  https://docs.databricks.com/data/data-sources/azure/azure-storage.html

Step 4: Create an HD Insight Cluster

- Create an HD Insight Cluster

  https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-hadoop-provision-linux-clusters

Step 5: Hive Programming in HD Insight

- Hive in HDInsight.

  https://www.youtube.com/watch?v=K8ALa8ttJE4&feature=youtu.be

Step 6: Hive QL Script

- Write a HQL script in Beeline CLI.

  https://github.com/Azure-Samples/Azure-MachineLearning-DataScience/blob/master/Misc/DataScienceProcess/DataScienceScripts/sample_hive_create_db_tbls_load_data_generic.hql

Step 7: Execute Queries and Store the Query result on the Blob Container

- Execute all queries.
- Store the query result in Blob Container.