QUERYING DATA USING HIVE

HOMEWORK 5

Q1. What is a metastore in Hive?

Metastore is a service that stores metadata related to Apache Hive and other services, in a backend RDBMS. It provides client access to this information by using metastore service API.

Q2. Where does the data of a Hive table gets stored?

Hive data are stored in RDBMS like mySQL, see supported RDBMS. Hive is a data warehouse database for Hadoop, where all data files are stored at HDFS location /user/hive/warehouse by default and also Hive stores its table data in a metastore.

Q3. Why Hive does not store metadata information in HDFS?

Hive stores metadata information in the metastore using RDBMS instead of HDFS because storing in metastore is to achieve low latency as HDFS read and write operations are time consuming processes i.e) high latency.

Q4. What is the difference between local and remote metastore?

>Local metastore service still runs in the same JVM (Java Virtual Machine) as Hive but it connects to a database running in a separate process either on same machine.

>Remote metastore runs in its own separate JVM not on hive service JVM.

Q5. What is the default database provided by Apache Hive for metastore?

Derby database is the default database by Apache for Apache Hive which supports only one user, so only one shell you can open.

Q6. What is the difference between external table and managed table?

>External tables are tables where Hive has loose coupling with the data whereas, Managed tables are Hive owned tables where the entire lifecycle of the table data are managed and controlled by Hive.

>We can identify the Managed or External table using the ' DESCRIBE FORMATTED table_name' statement in the Hive, which displays either MANAGED_TABLE or EXTERNAL_TABLE depending on the given table type.

Q7.Is it possible to change the default location of a managed table?

By using the LOCATION keyword, we can change the default location of Managed tables while creating the managed table in Hive i.e) by using LOCATION '<hdfs_path>'.

Q8. What is a partition in Hive?

Hive Partition is a way to split the dividing the tables into smaller tables based on the values of a column.

Here the data is stored in slices, so the query response time becomes faster.

Q9. Why do we perform partitioning in Hive?

Hive organizes tables into partitions i.e) a way of dividing a table into smaller parts based on the values of partitioned column. Using partition, it is easy to query a portion of the data since the query response time is faster.

Q10. What is dynamic partitioning and when is it used?

Dynamic partitioning is the strategic approach used to load the data from the non-partitioned table to the partition table. Here the values of partitioned columns exist within the table. So, it is not required to pass the values of partitioned columns manually.

Q11. Suppose, you create a table that contains details of all the transactions done by the customers of year 2022: CREATE TABLE transaction_details (cust_id INT, amount FLOAT, month STRING, country STRING) ROW FORMATDELIMITED FIELDS TERMINATED BY ‘,’ ; Now, after inserting 50,000 tuples in this table, you want to know the total revenue generated for each month. But, Hive is taking too much time to process this query. How will you solve this problem and list the steps that you will be taking in order to do so?

Let's solve this problem by partitioning the table according to each month. So, we scan only the partitioned data for each month instead of whole data sets. We know that partitioning an existing non-partitioned table directly is impossible. So, following are the steps to solve this problem,

1) Let us Create a partitioned table named transaction_details,

CREATE TABLE transaction_details (cust_id INT, amount FLOAT, country STRING) PARTITIONED BY (month STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’ ;

2) Enable the dynamic partitioning :

SET hive.exec.dynamic.partition = true;

SET hive.exec.dynamic.partition.mode = nonstrict;

3) Transfer the non–partitioned table data into the recently created partitioned table:

INSERT OVERWRITE TABLE transaction_details PARTITION (month) SELECT cust_id, amount, country, month FROM transaction_details;

Now, we can perform the query using each partition and therefore, decrease the query time.