HIVE SCENARIO-BASED INTERVIEW QUESTIONS

1. **A hive table running in production with 8 columns data still date, from tomorrow Two more column's Data is coming. How to load 10 columns data from tomorrow?**

   Ans: drop hive table, recreate hive table with 10 columns. For pervious dates two new columns values will be null.

2. **If we have hive table with 8 columns, but data has only 10 columns, what data will load to hive table? Any error?**

   Ans: Yes, data will load without any errors, hive load only 8 columns data, it will ignore remaining to column's data.

3. **If we have a hive table with 10 columns, but data has only 9 columns, what data will load to 10th column in hive table? Any error?**

   Yes, data will load without any errors, but the 10th column in table is set to default null value.

4. **Let's say a Hive table is created as an external table. If we drop the table, will the data be accessible?**

   The data will be accessible even if the table gets dropped. We can get the data from the table's HDFS location.

5. **A Hive table is created as an external table at location say hdfs://usr/data/table_name. If we dump a data set which has the data as per the table structure, will we able to fetch the records from the table using a select query?**

   Yes, we will be able to fetch the records from the table after dumping the data set at the hive table external location.

6. **A Hive partition table is created which is partition by a column say yearofexperience. If we create a directory say yearofexperience=3 at the HDFS path of the table and dump the data**

set which is as per the table structure. Will the data be available if we execute the select query on the table?

No, the data will not be accessible by executing the select query on the table. After dumping the data files at table HDFS location for the partition, you will have to update the metadata using below command:

Ex: msck repair table <tablename>

7. **Let's take the same previous Hive partition table. If we drop the partition, will we able to access the data?**

   If a hive partition is created as a managed table, then after dropping the partition, data will also get removed from the path. But in the case of an external table, data will be accessible from the same external path of the hive partition table.

8. **Let's take the same previous Hive partition table partitioned by a column named yearofexperience. It has multiple partitions at the HDFS location. If we drop a partition directory say yearofexperience=3 from the HDFS location, will this partition be listed while querying show partitions on the table?**

   If we drop the partition directory say hive/warehouse/parti_test_ext/yearofexperience=3 from the HDFS location, it will be listed if you query show partitions on the table.

9. **Suppose we have created a Hive partition table which is partitioned by a column named city. We are getting data which has Empty/Null value for the partition column(city) and must load these data into the hive table with dynamic partition as it has multiple city records in the data set. In which partition the records, with an empty value for city column, will be available?**

   While loading the data into a table using dynamic partition if any null or empty value comes for a defined partition column, then it uses to create a default partition named __HIVE_DEFAULT_PARTITION__ at HDFS location and dump those records in that partition.

10. **Let's say we have created a Hive partition table. This table gets updated every day with a huge volume of data. As we already know that the table has a high volume of data, we want to restrict the query not to do a full scan on the table. How will you achieve this?**

You can achieve this by setting below properties:

SET hive. mapred. mode=strict;

Hive Strict Mode (hive. mapred. mode=strict) enables hive to restrict certain performance intensive operations. Such as – It restricts queries of partitioned tables without a WHERE clause.

11. **If we create a table with an EXTERNAL keyword, but not mentioning any location in the create table statement, which kind of table it will be – managed or external?**
    The created table will behave like an external table that means if you drop the table, data will be available at the table HDFS location.

12. **We have created a view on top of a Hive table. If we drop the Hive table, will the View be accessible?**
    The view will not be accessible. It will throw an error like Table not found. Because view internally excute select query on that table.

13. **Let's say you want to create a table which is having columns name like Hive keyword (say, timestamp, date, etc.) or column name having space (say "col 50"). How will you create the table in Hive?**
    You can mention the column name enclosed by backticks (`).
    For example- `timestamp` string, `col 50` string

14. **I have a lot of small CSV files present in /input directory in HDFS and I want to create a single Hive table corresponding to these files. The data in these files are in the format: {id, name, e-mail, country}. Now, as we know, Hadoop performance degrades when we use lots of small files.So, how will you solve this problem where we want to create a single Hive table for lots of small files without degrading the performance of the system?**

    One can use the sequencefile format which will group these small files together to form a single sequence file. The steps that will be followed in doing so are as follows:
    Step1: Create a temporary table:
    CREATE TABLE temp_table (id INT, name STRING, e-mail STRING, country STRING)
    ROW FORMAT FIELDS DELIMITED TERMINATED BY ',' STORED AS TEXTFILE;

    Step2: Load the data into temp_table:
    LOAD DATA INPATH '/input' INTO TABLE temp_table;

Step3: Create a table that will store data in sequencefile format:
CREATE TABLE sample_seqfile (id INT, name STRING, e-mail STRING, country STRING)
ROW FORMAT FIELDS DELIMITED TERMINATED BY ‘,’ STORED AS SEQUENCEFILE;

Step4: Transfer the data from the temporary table into the sample_seqfile table:
INSERT OVERWRITE TABLE sample SELECT * FROM temp_table;

Hence, a single sequencefile is generated which contains the data present in all of the input files and therefore, the problem of having lots of small files is finally eliminated.

15. **I am inserting data into a table based on partitions dynamically. But, I received an error – FAILED ERROR IN SEMANTIC ANALYSIS: Dynamic partition strict mode requires at least one static partition column. How will you remove this error?**

    To remove this error, one has to execute following commands:
    SET hive.exec.dynamic.partition = true;
    SET hive.exec.dynamic.partition.mode = nonstrict;

    Things to Remember:
    - By default, hive.exec.dynamic.partition configuration property is set to False in case you are using Hive whose version is prior to 0.9.0.
    - hive.exec.dynamic.partition.mode is set to strict by default. Only in non – strict mode Hive allows all partitions to be dynamic.

16. **I create a table that contains details of all the transactions done by the customers of year 2016: CREATE TABLE transaction_details (cust_id INT, amount FLOAT, month STRING, country STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’; Now, after inserting 50,000 tuples in this table, I want to know the total revenue generated for each month. But Hive is taking too much time in processing this query. How will you solve this problem and list the steps that I will be taking in order to do so?**

    We can solve this problem of query latency by partitioning the table according to each month. So, for each month we will be scanning only the partitioned data instead of whole data sets.

    As we know, we can’t partition an existing non-partitioned table directly. So, we will be taking following steps to solve the very problem:

Step1: Create a partitioned table, say partitioned_transaction:

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

Step 2: Enable dynamic partitioning in Hive:

SET hive.exec.dynamic.partition = true;

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

Step3: Transfer the data from the non – partitioned table into the newly created partitioned table:

INSERT OVERWRITE TABLE partitioned_transaction 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.