**Scenario Based questions:**

**Q: Will the reducer work or not if you use “Limit 1” in any HiveQL query?**

**Ans:** In Hive, the use of "LIMIT 1" in a query would cause the reducer phase to be skipped. This is because the "LIMIT 1" clause effectively limits the results of the query to a single row, which can be returned directly from the mapper without the need for a reducer. As a result, the data is not passed to the reducer phase and no aggregation or processing takes place in this stage.

**Q: Suppose I have installed Apache Hive on top of my Hadoop cluster using default metastore configuration. Then, what will happen if we have multiple clients trying to access Hive at the same time?**

**Ans:** The metastore database will be shared by all clients accessing apache hive using the default metastore setup. In this setup, the metadata regarding the schema and structure of the data kept in the Hadoop cluster is saved and retrieved by the metastore, which serves as a common service accessed by all Hive clients. Conflicts between the various clients may occur if they attempt to edit the same information simultaneously while many clients are accessing the Hive metastore. Hive has locking methods that permit only one client to edit the metadata at a time in order to prevent these conflicts.

**Q: Suppose, I create a table that contains details of all the transactions done by the customers: CREATE TABLE transaction\_details (cust\_id INT, amount FLOAT, month STRING, country STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’ ;**

Now, after inserting 50,000 records 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?

**Ans:** The "month" field may be used to split the "transaction details" table. Based on the value of the partition column, partitioning divides the data in the table into distinct subdirectories. Due to Hive's ability to process just the partitions that are pertinent to the query, less data must be processed, which improves query performance. The information kept in the "transaction details" database can be compressed. Data size reduction by compression makes data more readily readable from disc, which can enhance query performance. The "month" field of the "transaction details" database can be used to group data. As Hive can only scan the pertinent buckets, bucketing separates comparable records into distinct files, which can increase query speed.

**Q: How can you add a new partition for the month December in the above partitioned table?**

**Ans:** ALTER TABLE transaction\_details ADD PARTITION (month='December');

LOAD DATA INPATH '/path/to/data/for/December' INTO TABLE transaction\_details PARTITION (month='December');

to remove the error "Dynamic partition strict mode requires at least one static partition column", you can either turn off dynamic partition strict mode in Hive, or you can specify the values for at least one of the partition columns in the insert statement.

**Q: 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?**

**Ans:** set hive.exec.dynamic.partition.mode=nonstrict

**Q: Suppose, I have a CSV file – ‘sample.csv’ present in ‘/temp’ directory with the following entries:**

**id first\_name last\_name email gender ip\_address**

**How will you consume this CSV file into the Hive warehouse using built-in SerDe?**

**Ans:** To consume a CSV file into the Hive warehouse using built-in SerDe, you can create a table in Hive that maps to the structure of the CSV file, and then load the data from the file into the table using the LOAD DATA INPATH command.

**Q: Suppose, I have a lot of small CSV files present in the 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?**

**Ans:** Solution to this problem is to use the Hive's "external table" feature to combine the small CSV files into a single Hive table without copying the data into the Hive warehouse. This way, you can avoid degrading the performance of the system while still being able to access the data as if it were stored in a single Hive table.

**Q:**

LOAD DATA LOCAL INPATH ‘Home/country/state/’

OVERWRITE INTO TABLE address;

The above statement failed to execute. What can be the cause?

**Ans:** Incorrect path.

**Q: Is it possible to add 100 nodes when we already have 100 nodes in Hive? If yes, how?**

**Ans:** Install Hadoop and Hive on the new nodes: Make sure that the version of Hadoop and Hive you are using is compatible with the existing cluster. Update the cluster configuration: Add the new nodes's hostnames or IP addresses to the workers or slaves file in your Hadoop configuration.

***Hive Practical questions:***

**Q: Hive Join operations**

**Create a table named CUSTOMERS(ID | NAME | AGE | ADDRESS | SALARY)**

**Create a Second table ORDER(OID | DATE | CUSTOMER\_ID | AMOUNT)**

**Now perform different joins operations on top of these tables**

**(Inner JOIN, LEFT OUTER JOIN ,RIGHT OUTER JOIN ,FULL OUTER JOIN)**

**Ans:** # Creating tables:

CREATE TABLE CUSTOMERS

(

ID INT,

NAME STRING,

AGE INT,

ADDRESS STRING,

SALARY FLOAT

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

CREATE TABLE ORDERS

(

OID INT,

DATE STRING,

CUSTOMER\_ID INT,

AMOUNT FLOAT

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

*# Executing join queries*

SELECT \* FROM CUSTOMERS cI NNER JOIN ORDERS o ON c.ID = o.CUSTOMER\_ID;

SELECT \* FROM CUSTOMERS c LEFT JOIN ORDERS o ON c.ID = o.CUSTOMER\_ID;

SELECT \* FROM CUSTOMERS c RIGHT OUTER JOIN ORDERS o ON c.ID = o.CUSTOMER\_ID;

SELECT \* FROM CUSTOMERS c LEFT OUTER JOIN ORDERS o ON c.ID = o.CUSTOMER\_ID;

*# Download a data from the given location -* [*https://archive.ics.uci.edu/ml/machine-learning-databases/00360/*](https://archive.ics.uci.edu/ml/machine-learning-databases/00360/) *🡺* ***Done***

**1. Create a hive table as per given schema in your dataset.**

🡺Create external table air\_quality

(

`Date` date,

`time` string,

co\_gt float,

s1\_co int,

nmhc\_gt int,

C6h6\_gt float,

S2\_nmhc int,

Nox\_gt int,

S3\_nox int,

No2\_gt int,

S4\_no2 int,

s5\_o3 int,

T float,

RH float,

AH float

)

row format delimited

fields terminated by ';'

tblproperties ("skip.header.line.count" = "1");

location ' /user/air\_quality\_data'

**2. Try to place a data into table location.**

🡺 load data local inpath '/config/workspace' into table air\_quality;

**3. Perform a select operation.**

🡺select `Date`, s5\_o3 from air\_quality;

**4. Fetch the result of the select operation in your local as a csv file.**

🡺insert overwrite local directory ' /config/workspace' row format delimited fields terminated by ',' select \* from air\_quality;

**5. Perform group by operation.**

🡺select `Date`, avg(s1\_co) from air\_quality group by `Date`;

**7. Perform filter operation at least 5 kinds of filter examples.**

🡺select \* from air\_quality where s1\_co > 1500;

**8. Show and example of regex operation.**

🡺select regexp\_extract(`Date`, '(^|[^0-9])(20[0-9]{2}|19[0-9]{2})([^0-9]|$)', 2) as extracted\_year, avg(s5\_o3) from air\_quality group by regexp\_extract(`Date`, '(^|[^0-9])(20[0-9]{2}|19[0-9]{2})([^0-9]|$)', 2);

**9. Alter table operation.**

🡺alter table air\_quality change s1\_co S1\_CO float;

**10. Drop table operation.**

🡺drop table air\_quality;

**12. Order by operation.**

🡺select s1\_co, from air\_quality order by s5\_o3 asc;

**13. where clause operations you have to perform.**

🡺select \* from air\_quality where `Date` rlike '10/03/2004|10/03/2005';

**14. Sorting operation you have to perform.**

🡺select s5\_o3, `Date` from air\_quality order by s5\_o3 desc;

**15. Distinct operation you have to perform.**

🡺select count(distinct `Date`) from air\_quality;

**16. Like an operation you have to perform.**

🡺 select \* from air\_quality where `Date` rlike '10/03/2004|10/03/2005';

**18. Table view operation you have to perform.**

🡺create view air\_quality\_data as select no2\_gt, s5\_o3, `Date` from air\_quality;

***hive operation with python***

Create a python application that connects to the Hive database for extracting data, creating sub tables for data processing, drops temporary tables.fetch rows to python itself into a list of tuples and mimic the join or filter operations.

from pyhive import hive

import pandas as pd

# Establish a connection to Hive server

conn = hive.Connection(host='localhost', port=10000)

# Create a cursor to execute SQL commands

cursor = conn.cursor()

# Execute a SQL query to create a sub table

query\_create\_sub\_table = "create table air\_quality\_subtable as select `Date`, `time`, s1\_co, s5\_o3 from air\_quality"

cursor.execute(query\_create\_sub\_table)

# Execute a SQL query to fetch rows from the sub table

query\_fetch\_rows = "select \* from air\_quality"

cursor.execute(query\_fetch\_rows)

# Fetch the rows and convert them into a list of tuples

rows = cursor.fetchall()

list\_of\_tuples = [tuple(row) for row in rows]

# Mimic join or filter operations in Python

df = pd.DataFrame(list\_of\_tuples, columns=['s1\_co', 'S2\_nmhc', 'S3\_nox', 'S4\_no2'])

df\_filtered = df.loc[df['s5\_o3'] > 1500]

df\_joined = pd.merge(df, df\_filtered, on='`Date', how='inner')

# Execute a SQL query to drop the sub table

query\_drop\_sub\_table = "drop table air\_quality\_subtable"

cursor.execute(query\_drop\_sub\_table)

# Close the cursor and the connection

cursor.close()

conn.close()