Scenerio based question:

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

Ans. Reducer will work but using Limit 1 in HiveQL query can potentially affect the performance of the reducer phase.

**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. If you have installed Apache Hive on top of my Hadoop cluster using metastore configuration, multiple clients trying to access Hive at same time will result in concurrent access to the metastore and potentially lead to concurrency-related issues

Metastore concurrency: The metastore in Hive, by default uses an embedded Derby database as its metastore database. Derby, being a single-user database, does not handle concurrent connections well. When multiple clients attempt to access the Hive metastore simultaneously, then may experience conflicts and race conditions, leading to inconsistent or incorrect results.

**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. To solve this problem we can use portioning.

Step-1 Partition table: We can partition the transaction\_details table by the month column

hive> CREATE TABLE transaction\_details (

> cust\_id INT,

> amount FLOAT,

> country STRING

> )

> PARTITIONED BY (month STRING)

> ROW FORMAT DELIMITED

> FIELDS TERMINATED BY ',';

Step-2 Insert record with appropriate partition: When inserting the records into the transaction\_details table, you need to ensure that each record is assigned to the correct partition based on the month value. Ex. for records in January, you would insert them into the partition month= ‘January’ and so on.

hive> INSERT INTO TABLE transaction\_details PARTITION (month='January') VALUES (cust\_id, amount, country);

Step -3: Query total revenue of the month:

SELECT month, SUM(amount) AS total\_revenue

FROM transaction\_details

GROUP BY month;

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

**Ans.**

Step-1: Alter the table and adds a new partition

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

Step -2:After adding the partition, you can proceed to insert records specific to the month of December into the newly added partition. Ensure that the data is inserted into correct partition by specifying the partition name in the INSERT statement.

INSERT INTO TABLE transaction\_details PARTITION (month='December')

VALUES (cust\_id, amount, country);

**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. Before inserting data into dynamic partition we need to set a property set hive.exec.dynamic.partition.mode=nonstrict;

After setting this property if we insert the data then it will work fine.

**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. Create a external table corresponds to the structure of the csv file.

CREATE EXTERNAL TABLE IF NOT EXISTS sample\_table (

id INT,

first\_name STRING,

last\_name STRING,

email STRING,

gender STRING,

ip\_address STRING

)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

WITH SERDEPROPERTIES (

'separatorChar' = ',',

'quoteChar' = '\"',

'escapeChar' = '\\'

)

STORED AS TEXTFILE

LOCATION '/temp';

After creating external table we can load data from /temp location

LOAD DATA INPATH '/temp/sample.csv' INTO TABLE sample\_table;

**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. To solve this problem we can follow these steps:

1. Combine small csv files: Start combining these small files into larger files. This process is known as file consolidation or file merging. We can use Hadoop Map Reduce or Spark to perform this task efficiently.

2. Choose an optimal file size: Determine the optimal file size for you consolidated files. The ideal file size depends on various factors such as cluster configuration, hardware resources, and workload characterstics. A common approach is to choose a file size between 128 MB to 1 GB, as it provides a good balance between read performance and resource utilization.

3. Convert CSV files into Parquet file format: Convert the consolidated CSV files to a more optimized file format like Parquet. Parquet provides efficient compression and column pruning capabilities, which can significantly improve query performance.

4. Define external table: Create an external table in Hive that corresponds to the consolidated and converted parquet file.

**Q. LOAD DATA LOCAL INPATH ‘Home/country/state/’**

**OVERWRITE INTO TABLE address;**

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

Ans. It seems like an issue in the statement itself. Correct statement will be:

LOAD DATA LOCAL INPATH ‘file:///Home/country/state/--.csv’ OVERWRITE INTO TABLE address;

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

Ans. Yes it is possible to add more nodes to a Hive cluster even if you already have 100 nodes. Hive is designed to scale horizontally, allowing you to add more nodes to increase the cluster’s processing power and storage capacity.

**Practical question:**

**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. Create table CUSTOMERS

(

ID int,

NAME string,

AGE int,

ADDRESS string,

SALARY float

)

row format delimited

fields terminated by ‘,’;

Create table ORDER

(

ID int,

DATE date,

CUSTOMER\_ID int,

AMOUNT float

)

row format delimited

Fields terminated by ‘,’;

Inner join:

SELECT \*

FROM CUSTOMERS

INNER JOIN ORDER ON CUSTOMERS.ID = ORDER.CUSTOMER\_ID;

Left Outer join:

SELECT \*

FROM CUSTOMERS

LEFT OUTER JOIN ORDER ON CUSTOMERS.ID = ORDER.CUSTOMER\_ID;

Right Outer join:

SELECT \*

FROM CUSTOMERS

RIGHT OUTER JOIN ORDER ON CUSTOMERS.ID = ORDER.CUSTOMER\_ID;

Full outer join: To achieve a full outer join you can use a combination of a left outer join and a right outer join and then combine the results using the UNION ALL operator.

SELECT \*

FROM CUSTOMERS

LEFT OUTER JOIN ORDER ON CUSTOMERS.ID = ORDER.CUSTOMER\_ID

UNION ALL

SELECT \*

FROM CUSTOMERS

RIGHT OUTER JOIN ORDER ON CUSTOMERS.ID = ORDER.CUSTOMER\_ID

WHERE CUSTOMERS.ID IS NULL;

The above query first performs a left outer join and then right outer join. The WHERE clause filters out the row where “CUSTOMERS.ID” is not present, simulating the behavior of a full outer join.

**Q . BUILD A DATA PIPELINE WITH HIVE**

**Download a data from the given location -**

**https://archive.ics.uci.edu/ml/machine-learning-databases/00360/**

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

Ans. create table air\_quality\_data

(

Record\_Date string,

Record\_time string,

CO\_GT FLOAT,

PT08\_S1\_CO INT,

NMHC\_GT INT,

C6H6\_GT FLOAT,

PT08\_S2\_NMHC INT,

NOx\_GT INT,

PT08\_S3\_NOx INT,

NO2\_GT INT,

PT08\_S4\_NO2 INT,

PT08\_S5\_O3 INT,

T FLOAT,

RH FLOAT,

AH FLOAT

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ';'

STORED AS TEXTFILE

Tblproperties(“skip.header.line.count”=”1”);

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

Ans. load data local inpath ‘file///config/workspace/AirQualityUCI.csv’ into table air\_quality\_data;

**3. Perform a select operation .**

Ans. select \* from air\_quality\_data;

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

Ans. INSERT OVERWRITE LOCAL DIRECTORY '/output'

> ROW FORMAT DELIMITED

> FIELDS TERMINATED BY ';'

> SELECT \* FROM air\_quality\_data;

**5. Perform group by operation .**

Ans. select record\_date from air\_quality\_data group by record\_date;

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

Ans.1.Filtering based on specific condition:

SELECT \*

FROM air\_quality\_data

WHERE CO\_GT > 2.0;

2. Filtering based on multiple condition:

SELECT \*

FROM air\_quality\_data

WHERE PT08\_S1\_CO < 1500 AND RH > 50.0;

3. Filtering based on pattern matching LIKE:

SELECT \*

FROM air\_quality\_data

WHERE record\_time LIKE '18%';

4. Filtering using ‘IN’ operator:

SELECT \*

FROM air\_quality\_data

WHERE record\_date IN ('10/03/2004', '10/04/2004');

5. Filtering with NULL check:

SELECT \*

FROM air\_quality\_data

WHERE NMHC\_GT IS NULL;

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

Ans. SELECT \*

FROM air\_quality\_data

WHERE record\_date RLIKE '^\\d{2}/\\d{2}/\\d{4}$';

In this example, the RLIKE operator is used to match the record\_date column against the specified regex pattern. The pattern ‘^\\d{2}/\\d{2}/\\d{4}$’ matches the date format in the dd/mm/yyyy pattern. The ^ and $ anchors ensure that the entire value matches the pattern from the beginning to the end. The \\d{2} matches two digits and \\d{4} matches four digits.

**9. alter table operation**

Ans. In Hive ALTER TABLE used to modify the structure or properties of the existing table.

1. ADD NEW COLUMN TO THE TABLE

ALTER TABLE table\_name ADD COLUMN new\_column\_name data\_type;

2. RENAME COLUMN:

ALTER TABLE table\_name CHANGE COLUMN old\_column\_name new\_column\_name data\_type;

3.Modify datatype of a column

ALTER TABLE table\_name CHANGE COLUMN column\_name new\_column\_name new\_data\_type;

4. DROP column:

ALTER TABLE table\_name DROP COLUMN column\_name;

**10 . drop table operation**

ANS. DROP TABLE alter\_quality\_data;

**12 . order by operation .**

Ans.

SELECT \* FROM alter\_table\_data order by C6H6\_GT;

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

Ans. SELECT \*

FROM air\_quality\_data

WHERE CO\_GT > 2.0;

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

Ans. SELECT \*

FROM table\_name

ORDER BY column\_name ASC;

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

Ans. SELECT DISTINCT column\_name

FROM table\_name;

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

Ans. SELECT \*

FROM air\_quality\_data

WHERE record\_time LIKE '18%';

**17 . union operation you have to perform .**

Ans. SELECT column1, column2

FROM table1

UNION

SELECT column1, column2

FROM table2;

The UNION operator eliminated duplicates rows and returns only distinct rows in the combined result set. If you want to include duplicate rows in the combined result set you can use the UNION\_ALL operator instead.

Note that the tables involved in the union operation must have the same or compatible column structures.

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

Ans. CREATE VIEW air\_quality\_data\_view AS SELECT \* FROM air\_quality\_data;

SELECT \* FROM air\_quality\_data\_view;

**Q . 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

# Establish a connection to Hive

conn = hive.Connection(host='localhost', port=10000, username='your\_username')

# Create a cursor to execute Hive queries

cursor = conn.cursor()

# Extract data from a table

cursor.execute("SELECT \* FROM table\_name")

data = cursor.fetchall()

# Create a subtable for data processing

cursor.execute("CREATE TABLE subtable AS SELECT column1, column2 FROM table\_name WHERE condition")

# Drop temporary tables

cursor.execute("DROP TABLE IF EXISTS subtable")

# Fetch rows to Python as a list of tuples

cursor.execute("SELECT column1, column2 FROM table\_name")

rows = cursor.fetchall()

# Perform join operation

cursor.execute("SELECT t1.column1, t2.column2 FROM table1 t1 JOIN table2 t2 ON t1.id = t2.id")

# Perform filter operation

cursor.execute("SELECT \* FROM table\_name WHERE condition")

# Close the cursor and connection

cursor.close()

conn.close()