Q. 1. What is a metastore in Hive?

Metastore is the central repository of Apache Hive metadata. It stores metadata for Hive tables (like their schema and location) and partitions in a relational database. It provides client access to this information by using metastore service API

Q. 2. Where does the data of a Hive table gets stored?
data of a Hive table gets stored in HDFS.

For Managed table - location will be as per hive.warehouse.dir property in hive-site.xml config file . Default value is /usr/local/hive/warehouse/.
For External Tables- need to specify the location explicitly with LOCATION '<path>' in CREATE TABLE command

Q. 3. Why Hive does not store metadata information in HDFS?

Hive stores metadata information in the metastore using RDBMS instead of HDFS. The reason for choosing RDBMS is to achieve low latency as HDFS read/write operations are time consuming processes

Q. 4. What is the difference between local and remote metastore?

– Local Mode:
–
Basically, as the main HiveServer process, the Hive metastore service runs in the same process, but make sure Hive metastore database runs in a separate process, as well as it can be on a separate host, in Local mode.

– Remote Mode:
–
Whereas, the Hive metastore service runs in its own JVM process. Some processes(HiveServer2, HCatalog, Cloudera Impala™,) communicate with it with the help of the Thrift network API, in Remote mode

Q. 5. What is the default database provided by Apache Hive for metastore?

Ans : Derby is the default database for the embedded metastore.

Q. 6. What is the difference between external table and managed table

| Internal Table | External Table |
| --- | --- |
| Hive moves data to the warehouse directory. | Hive does not move data to the warehouse directory. |
| Dropping table deletes the data and metadata | Dropping table only deletes the metadata |
| Support ACID | Does not support ACID |

Q. 7. Is it possible to change the default location of a managed table

Yes, you can do it by using the clause – LOCATION '<hdfs_path>' we can change the default location of a managed table

Q. 8. What is a partition in Hive?

The partitioning in Hive means dividing the table into some parts based on the values of a particular column like date, course, city or country.

Q. 9. Why do we perform partitioning in Hive?

The advantage of partitioning is that since the data is stored in slices, the query response time becomes faster.

Q. 10. What is dynamic partitioning and when is it used?

Dynamic partitioning is the strategic approach to load the data from the nonpartitioned table where the single insert to the partition table is called a dynamic partition.

Use dynamic partitioning when data is not already physically categorized/grouped/partitioned. Dynamic partitioning will result in MapReduce job execution to group the data first and then partitions will be added to the table.

Q. 11. Suppose, you create a table that contains details of all the transactions done

Suppose, I 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 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?
ans)

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:

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 ',' ;
```

2. Enable dynamic partitioning in Hive:

```
SET hive.exec.dynamic.partition = true;

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

3. 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.