Q 1. What is the definition of Hive? What is the present version of Hive?

ANS - a hive can refer to a data warehousing infrastructure and query execution system. Apache Hive, for example, is an open-source data warehouse infrastructure built on top of Apache Hadoop. It provides a way to query and analyze large datasets stored in Hadoop's distributed file system using a SQL-like language called HiveQL.

At that time, the latest stable version of Apache Hive was 3.1.2

Q 2. No, Hive is not typically suitable for OLTP (Online Transaction Processing) systems. Hive is primarily designed for OLAP (Online Analytical Processing) workloads, which involve complex analytical queries and processing large volumes of data. Here are a few reasons why Hive may not be suitable for OLTP systems: Performance, Data Consistency Data Consistency, Data Updates

Q 3. How is HIVE different from RDBMS? Does hive support ACID transactions. If not then give the proper reason.

1. Data Model: Hive uses a schema-on-read approach, where the structure of data is applied during query execution rather than enforcing a predefined schema on the data itself. In contrast, RDBMS follows a schema-on-write approach, where a fixed schema is enforced when inserting data into the database.

2. Query Language: Hive uses a SQL-like language called HiveQL, which is similar to SQL but has some differences and extensions to support the distributed computing model of Hadoop. RDBMS typically use SQL as the standard query language.

3. Data Processing Paradigm: Hive is designed for batch-oriented processing of large datasets, optimized for OLAP workloads. RDBMS, on the other hand, are designed for OLTP workloads that involve real-time transactional operations on smaller, frequently updated datasets.

Regarding ACID (Atomicity, Consistency, Isolation, Durability) transactions, historically Hive did not support them The introduction of Hive Transactions and ACID support allows for atomicity, consistency, and isolation of transactions on Hive tables. This feature enables more complex data operations and concurrent updates while maintaining data consistency. It is important to note that the level of ACID support may vary depending on the version of Hive being used.

Q 4. Explain the hive architecture and the different components of a Hive architecture?

The Hive architecture consists of various components that work together to enable data processing and querying on large datasets. Here are the key components of a typical Hive architecture:

1. User Interface (UI): The User Interface component allows users to interact with Hive and submit queries. It provides different interfaces, such as a command-line interface (CLI), a web-based graphical interface (Hive Web UI), or programming interfaces (JDBC or ODBC) for integrating Hive with other applications.

2. Driver: The Driver component receives HiveQL queries from the user interface and coordinates the execution of those queries.

3. Metastore: The Metastore is a central component that stores metadata information about tables, databases, columns, partitions, and other schema-related details. It maintains the schema information and the mapping between Hive tables and the underlying data files or directories in the file system.

4. Compiler/Planner: The Compiler or Planner component takes the parsed query from the Driver and generates an execution plan. It performs optimizations

5. Execution Engine: The Execution Engine executes the query plan generated by the Compiler/Planner. Hive supports multiple execution engines, such as MapReduce, Tez, and Spark.

6. Metastore Warehouse: The Metastore Warehouse is the physical storage where the metadata maintained by the Metastore is stored. It can be implemented using a relational database like MySQL or PostgreSQL.

7. External Systems: Hive supports different file formats like CSV, JSON, Parquet, etc. Additionally, Hive can access data stored in external databases or data warehouses through connectors like JDBC or ODBC.

These components work together to enable users to define, manage, and query large datasets using HiveQL in a distributed computing environment. The architecture allows Hive to leverage the scalability and fault tolerance provided by Hadoop's ecosystem while providing a SQL-like interface for data processing and analysis.

5. Mention what Hive query processor does? And Mention what are the

components of a Hive query processor?

ANS - The Hive query processor plays a crucial role in executing queries in Hive. It takes the user's HiveQL query, performs query parsing, optimization, and plan generation, and coordinates the execution of the query.

The components of a Hive query processor include:

1. Parser: The parser component is responsible for parsing the user's HiveQL query and converting it into an abstract syntax tree (AST). It checks the query's syntax and ensures it adheres to the HiveQL grammar rules.

2. Semantic Analyzer: The semantic analyzer performs semantic analysis on the AST generated by the parser. It verifies the query's semantic correctness, including checking the existence of tables, columns, and proper usage of functions and expressions. The semantic analyzer also resolves references to table aliases and performs other semantic validations.

3. Query Optimizer: The query optimizer takes the parsed and validated query and performs various optimization techniques to improve query performance. It includes logical optimization, such as predicate pushdown, join reordering, and query rewriting to simplify the query.

4. Query Planner: The query planner takes the optimized query and generates an execution plan. It determines the steps and stages required to execute the query efficiently.

5. Task Generator: The task generator component converts the execution plan into a set of tasks that can be executed in a distributed computing environment.

6. Metastore Interaction: The query processor interacts with the Metastore to fetch metadata information about tables, columns, partitions, and other schema-related details. This interaction is necessary for validating the query, accessing table statistics, and optimizing query execution based on metadata.

Hive query processor to handle the parsing, semantic analysis, optimization, and planning of queries.

Q 6. What are the three different modes in which we can operate Hive?

ANS- The three modes of operating Hive are

1. Local Mode: In Local Mode, Hive runs in a single JVM (Java Virtual Machine) process, typically on a local machine. This mode is suitable for development, testing, and small-scale data processing tasks

2. MapReduce Mode: MapReduce Mode is the default mode of operation for Hive. In this mode, Hive leverages the MapReduce framework of Apache Hadoop for distributed data processing

3. Spark Mode: Hive also supports operating in Spark Mode, leveraging the Apache Spark framework for distributed data processing

It's important to note that the specific mode of operation is determined by the Hive configuration and the environment in which Hive is deployed

Q 7. Features and Limitations of Hive.

ANS - it also has certain limitations that are important to consider. Let's explore the features and limitations of Hive:

- Features of Hive:

1. SQL-like Query Language: Hive provides a SQL-like query language called HiveQL, which allows users to write queries using familiar SQL syntax.

2. Scalability: Hive is designed to handle large-scale data processing

3. Extensibility: Hive supports user-defined functions (UDFs) and user-defined aggregates (UDAs), enabling users to extend its functionality by writing custom functions in various programming languages

4. Schema Evolution: Hive provides schema-on-read flexibility, allowing users to query and analyze data without enforcing a predefined schema.

5. Data Format Support: Hive supports a wide range of data formats, including CSV, JSON, Parquet, Avro, and more

- Limitations of Hive:

1. Batch Processing: Hive is primarily optimized for batch processing and is not well-suited for real-time or interactive querying

2. Limited Update and Delete Support: Historically, Hive had limited support for updates and deletes in tables

3. High Latency: Due to its reliance on Hadoop's MapReduce or other execution engines, Hive queries can have higher latency compared to traditional databases.

4. Lack of Indexing: Hive does not provide built-in indexing mechanisms for data

Q 8. How to create a Database in HIVE?

ANS -

A. Start by opening the Hive command-line interface or any other interface interact with Hive.

B. Once in the Hive shell create a database:

CREATE DATABASE database_name;

Replace database_name with the desired name for database.

SHOW DATABASES;

This command will display a list of all the databases available in Hive To switch to the newly created database, use the USE command:

USE database_name;

This command sets the specified database as the default database for subsequent Hive commands. Any subsequent queries or operations will be performed within this database.

using the format database_name.table_name.

Q 10. What do you mean by describe and describe extended and describe formatted with respect to database and table

I the describe command is used to retrieve metadata information about databases, tables, and columns. It provides insights into the structure and properties of these objects. The describe command has several variations, including describe database, describe TABLE, and additional options like EXTENDED and FORMATTED. Let's explore the meaning of each variant:

1. DESCRIBE DATABASE database_name: This command displays the metadata information about a specific database. It provides details such as the database name, its location in HDFS, and any other database-specific properties.

2. DESCRIBE TABLE table_name; : This command retrieves the metadata information of a specific table. It includes details like the table name, column names, data types, partitioning information, and table-specific properties.

3. DESCRIBE EXTENDED TABLE table_name : The DESCRIBE EXTENDED command provides additional information about a table compared to the regular DESCRIBE TABLE command. It includes details such as the table's physical properties, storage format, serialization information, and statistics like the number of rows.

4. DESCRIBE FORMATTED table_name;: The DESCRIBE FORMATTED command displays the metadata information of a table in a formatted and more readable manner

The DESCRIBE command and its variations are useful for understanding the structure and properties of databases and tables in Hive. They provide insights that can help with data exploration, query optimization

11.How to skip header rows from a table in Hive?

To skip header rows from a table in Hive, use the combination of Hive's built-in functions and table properties. Here's one approach to achieve this:

Assuming have a table named my_table with header rows, follow these steps to skip the header rows during data retrieval:

Create a new table with the same structure as my_table, but without the header rows. use the TBLPROPERTIES clause to specify the number of header rows to skip

```
CREATE TABLE my_table_no_header

LIKE my_table

TBLPROPERTIES ('skip.header.line.count'='1');
```

1. In the above example, 'skip.header.line.count'='1' indicates that one header row should be skipped. Adjust the value based on the number of header rows present in table.

2. Insert the data from the original table (my_table) into the new table (my_table_no_header), excluding the header rows.

```
INSERT INTO TABLE my_table_no_header

SELECT * FROM my_table;
```

1. This query retrieves data from my_table and inserts it into my_table_no_header, excluding the header rows based on the table property set in the previous step.

2. Once the data is inserted into my_table_no_header, use it for further processing, querying, or analysis, knowing that the header rows have been skipped.

12.What is a hive operator? What are the different types of hive operators?

ANS - In Hive, operators are symbols or keywords used in HiveQL queries to perform specific operations on data. They are used to manipulate, filter, transform, aggregate, or join data within Hive tables. Hive operators play a vital role in data processing and analysis tasks. There are several types of operators in Hive:

1. Arithmetic Operators: Arithmetic operators perform mathematical operations on numeric data. Hive supports common arithmetic operators such as addition (+), subtraction (-), multiplication (*), division (/), modulo (%), and unary minus (-) for negation.

2. Comparison Operators: Comparison operators are used to compare values and determine their relationships. Hive supports comparison operators like equal to (=), not equal to (!=), greater than (>), less than (<), greater than or equal to (>=), less than or equal to (<=), and the NULL-safe equality operator (<=>).

3. Logical Operators: Logical operators are used to combine or negate conditions in Hive queries. The supported logical operators include AND, OR, and NOT. These operators are typically used in conjunction with comparison operators to construct complex conditional expressions.

4. Assignment Operator: The assignment operator (=) is used to assign values to variables in Hive. It is commonly used with the SET command to define Hive configuration variables.

5. Bitwise Operators: Hive provides bitwise operators for performing bitwise operations on integer values. These operators include bitwise AND (&), bitwise OR (|), bitwise XOR (^), bitwise complement (~), left shift (<<), and right shift (>>).

6. String Operators: Hive supports various string operators for manipulating and comparing string data. Some of the common string operators include concatenation (||), string equality (=), string inequality (!=), string length (LENGTH), substring extraction (SUBSTRING), and pattern matching with regular expressions (RLIKE).

7. Aggregate Functions: While not technically operators, aggregate functions are essential for data analysis in Hive. Aggregate functions, such as SUM, AVG, COUNT, MIN, MAX, and GROUP BY, are used to perform calculations on groups of data and produce summary results.

Q 13. Explain about the Hive Built-In Functions

ANS - Hive provides a wide range of built-in functions that can be used in HiveQL queries to perform various data manipulation, transformation, and analysis tasks. These functions can be categorized into different types based on their functionality. Here are some common categories of Hive built-in functions:

1. Mathematical Functions: Hive offers a variety of mathematical functions, such as ABS, ROUND, CEIL, FLOOR, EXP, LOG, POWER, SQRT, and RAND. These functions allow to perform arithmetic calculations, rounding, exponentiation, logarithmic operations, and more on numeric data.

2. String Functions: Hive provides a comprehensive set of string functions for manipulating and analyzing string data. These functions include CONCAT, SUBSTRING, LENGTH, LOWER, UPPER, TRIM, REPLACE, REGEXP_EXTRACT, SPLIT, INSTR, LOCATE, and more. They enable to perform operations like concatenation, substring extraction, case conversions, trimming, pattern matching, and searching within strings.

3. Date and Time Functions: Hive supports various functions to handle date and time data. Functions like CURRENT_DATE, CURRENT_TIMESTAMP, YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, DATEDIFF, DATE_ADD, DATE_SUB, FROM_UNIXTIME, and UNIX_TIMESTAMP allow to extract, manipulate, and format date and time values.

4. Aggregate Functions: Hive includes a set of aggregate functions used for summarizing data. These functions, such as SUM, AVG, MIN, MAX, COUNT, GROUP BY, HAVING, and COLLECT_SET, enable to perform calculations on groups of data and generate summary results.

5. Conditional Functions: Hive provides conditional functions to handle conditional logic in queries. Functions like CASE, WHEN, THEN, ELSE, IF, and COALESCE allow to define conditional expressions, make decisions based on conditions, and handle null values.

6. Type Conversion Functions: Hive offers functions for converting data types. For example, CAST allows to convert data from one type to another, and functions like TO_DATE, TO_UNIX_TIMESTAMP, FROM_UNIXTIME, and UNIX_TIMESTAMP help convert date, time, and timestamp values between different formats.

7. Collection Functions: Hive supports functions for working with collections like arrays and maps. Functions such as ARRAY, MAP, EXPLODE, POSEXPLODE, GET_JSON_OBJECT, and TRANSFORM enable to manipulate and extract values from complex data structures.

Q 14. Write hive DDL and DML commands.

ANS - Hive provides both Data Definition Language (DDL) and Data Manipulation Language (DML) commands to create and manage database schemas, tables, and perform data operations

- Create Database: Creates a new database in Hive.

CREATE DATABASE database_name;

Create Table: Creates a new table in Hive.

```
CREATE TABLE table_name (
  column1 data_type,
  column2 data_type,
  ...
);
```

Alter Table: Modifies an existing table by adding, renaming, or modifying columns, changing table properties, or updating the table location.

```
ALTER TABLE table_name ADD COLUMN new_column data_type;
ALTER TABLE table_name RENAME TO new_table_name;
ALTER TABLE table_name SET TBLPROPERTIES ('key' = 'value');
ALTER TABLE table_name SET LOCATION 'hdfs://path/to/new/location';
```

- Data Manipulation Language (DML) Commands:

Insert Into: Inserts data into a table.

```
INSERT INTO TABLE table_name VALUES (value1, value2, ...);
```

Select: Retrieves data from one or more tables based on specified conditions.

```
SELECT column1, column2, ... FROM table_name WHERE condition;
```

Update: Updates data in a table based on specified conditions.

```
UPDATE table_name
 SET column1 = new_value1, column2 = new_value2 WHERE condition;
```

Q 15.Explain about SORT BY, ORDER BY, DISTRIBUTE BY and

CLUSTER BY in Hive.

ANS - In Hive, SORT BY, ORDER BY, DISTRIBUTE BY, and CLUSTER BY are keywords used to control the sorting and distribution of data during query execution

1. SORT BY: The SORT BY clause is used to sort the output of a query based on one or more columns. It guarantees the order of the final result set but does not affect the distribution of data across reducers

```
  sql
```

```
SELECT column1, column2
FROM table_name
SORT BY column1 ASC, column2 DESC;
```

2. ORDER BY: The ORDER BY clause is similar to SORT BY, as it also sorts the output of a query based on one or more columns. SORT BY, ORDER BY affects the data distribution across reducers and ensures that the data is globally sorted.

sql

```
SELECT column1, column2
FROM table_name
ORDER BY column1 ASC, column2 DESC;
```

3. DISTRIBUTE BY: The DISTRIBUTE BY clause is used to control the data distribution across reducers based on one or more columns. It divides the data into multiple groups based on the specified columns and assigns each group to a reducer. The data within each group is not guaranteed to be sorted.

Example:

sql

```
SELECT column1, column2
FROM table_name
DISTRIBUTE BY column1;
```

4. CLUSTER BY: The CLUSTER BY clause is similar to DISTRIBUTE BY, as it also controls the data distribution across reducers based on one or more columns

sql

```
SELECT column1, column2
FROM table_name
CLUSTER BY column1;
```

It's important to note that ORDER BY and CLUSTER BY can have a significant impact on query performance, especially for large datasets, as they involve shuffling and sorting data across reducers. DISTRIBUTE BY, on the other hand, primarily focuses on data distribution and does not enforce sorting. SORT BY is used for local sorting within each reducer but does not affect data distribution

Q 16.Difference between "Internal Table" and "External Table" and Mention

when to choose “Internal Table” and “External Table” in Hive?

ANS - there are two types of tables: Internal Table and External Table. The main difference between them lies in how they manage data storage and their behavior

1. Internal Table:

- Internal tables, also known as managed tables, manage both the table's metadata (schema) and the data itself.

- The data associated with an internal table is stored in a directory managed by Hive. Hive assumes full control over the data, including data deletion and metadata management.

- When drop an internal table, Hive removes both the metadata and the data associated with the table.

2. External Table:

- External tables allow Hive to reference data stored outside of Hive's control, such as in Hadoop Distributed File System (HDFS), Amazon S3, or a remote file system.

- The external table's metadata, including schema and table properties, is managed by Hive, but the data itself remains external and is not managed by Hive. Hive only tracks the location and schema of the external data.

- When drop an external table, Hive only removes the metadata associated with the table, leaving the underlying data intact.

- Choose Internal Table when:

Hive to fully manage the table's data.

- The data is generated by Hive jobs and doesn't need to be accessed or modified outside of Hive.

Hive to handle the data's lifecycle, including data deletion.

- Choose External Table when:

- The data is shared across multiple systems or tools.

- access or modify the data using other tools or processes outside of Hive.

preserve the data even if the table is dropped in Hive.

- The data is already existing and managed outside of Hive.

Q 17. Where does the data of a Hive table get stored?

ANS –

In Hive, the location where the data of a table is stored depends on whether the table is an Internal Table or an External Table.

1. Internal Table:

- For Internal Tables, Hive manages both the metadata (schema) and the data itself.
- The data associated with an Internal Table is stored in a directory within the Hive warehouse directory on the Hadoop Distributed File System (HDFS) or any other storage system configured as the default warehouse directory.
- By default, the location for Internal Tables follows the pattern: <warehouse_directory>/<database_name>.db/<table_name>.

2. External Table:

- For External Tables, Hive manages only the metadata (schema) of the table, while the actual data remains external to Hive.
- The data of an External Table is stored in a location external to Hive, such as in HDFS, Amazon S3, or a remote file system.
- The location of the external data is specified during the table creation using the LOCATION keyword.

Q 18. Is it possible to change the default location of a managed table?

ANS - No, it is not possible to change the default location of a managed table in Hive. The default location for managed tables is determined by the warehouse directory specified in the Hive configuration.

Q 19. What is a metastore in Hive? What is the default database provided by

Apache Hive for metastore?

ANS - In Hive, the metastore refers to the metadata storage component that manages the schema and other metadata information for Hive tables, partitions, columns, and other database objects. It acts as a central repository for storing and retrieving metadata related to Hive tables and provides the necessary information for query execution and data management

Q 20. Why does Hive not store metadata information in HDFS?

ANS - Hive does not store metadata information in HDFS (Hadoop Distributed File System) because HDFS is designed to handle large-scale distributed storage of data files, rather than efficiently managing metadata.

few reasons why Hive does not store metadata in HDFS:

Scalability Performance Flexibility AND Data independence.

Q 21. What is a partition in Hive? And why do we perform partitioning in

Hive?

ANS - In Hive, a partition is a way to organize data within a table based on one or more columns. It involves dividing the data into smaller, more manageable parts or subsets based on specific criteria. Each partition represents a subset of the data that shares common characteristics defined by the partitioning columns.

Q 22. What is the difference between dynamic partitioning and static

partitioning?

1. ANS - Static Partitioning:

- Static partitioning requires users to explicitly define the partition values during data insertion or loading.
- Partition values are predetermined and known in advance before inserting the data into the table.

Static partitioning is suitable when the partition values are already known and can be easily determined based on the data being loaded.

2. Dynamic Partitioning:

- Dynamic partitioning allows Hive to automatically determine and create partitions based on the values in the data being inserted or loaded.
- The partition column values are derived from the data itself, eliminating the need for explicit partition value specification.
- Hive scans the data being inserted, extracts the partition column values, and creates partitions accordingly.

Both static and dynamic partitioning have their advantages and use cases. Static partitioning provides more control over the partition values, while dynamic partitioning automates the partition creation process based on the data content.

Q 23. How do you check if a particular partition exists?

ANS –

Start by opening a Hive session or accessing Hive through a command-line interface.

Use the SHOW PARTITIONS statement followed by the table name to retrieve the list of

Alternatively, use the DESCRIBE FORMATTED statement to obtain detailed information about the table and its partitions, including the partition names. Scan the output of the SHOW PARTITIONS or DESCRIBE FORMATTED command to check if the desired partition is present.

Q 24. How can you stop a partition form being queried?

ANS - Open a Hive session or connect to Hive through a command-line interface.

Execute the following command to alter the partition and set it as non-readable:

ALTER TABLE table_name PARTITION (partition_column='partition_value') SET TBLPROPERTIES ('EXTERNAL'='TRUE');

Replace table_name with the name of the table, partition_column with the actual column name used for partitioning, and partition_value with the value of the partition to stop from being queried

Q 25. Why do we need buckets? How Hive distributes the rows into buckets?

ANS - Data Organization: Buckets provide a way to organize data within a partition into smaller, more evenly distributed units. It helps improve data locality and can optimize query performance by reducing data skew.

Data Skew Handling: In large datasets, certain values or ranges of values may occur more frequently than others, leading to data skew.

Join Optimization: Buckets can improve join performance in Hive. When joining two tables on a common column, if both tables are bucketed on that column and have the same number of buckets, Hive can perform a more efficient join operation called "bucket map join.

Bucket Distribution: Hive uses a hash function to determine the bucket number for each row based on the value of the bucketing column.

Bucketing Limitations: It's important to note that bucketing is not suitable for all scenarios

Q 26. In Hive, how can you enable buckets?

ANS - Start by creating a new table or altering an existing table to enable buckets. use the CREATE TABLE or ALTER TABLE statement accordingly.

Specify the number of buckets create using the CLUSTERED BY clause followed by the number of buckets and the column(s) used for bucketing

```
CREATE TABLE table_name (
  column1 data_type,
  column2 data_type,
  ...
)
CLUSTERED BY (bucketing_column) INTO num_buckets;
```

When altering an existing table, use the CLUSTERED BY clause within the ALTER TABLE statement.

Ensure that the bucketing column has a good distribution of values to effectively distribute the data across the buckets. enabling buckets for an existing table may require reorganizing the data, which can be a time-consuming process, especially for large tables.

Q 27.How does bucketing help in the faster execution of queries?

ANS - Hive helps in the faster execution of queries

Reduced Data Skew: In large datasets, certain values or ranges of values may occur more frequently than others, leading to data skew. Bucketing evenly distributes the data across multiple buckets based on a hash function applied to the bucketing column.

Efficient Join Operations: When joining two bucketed tables on a common column, if both tables have the same number of buckets and are bucketed on that column, Hive can perform a more efficient join operation called "bucket map join.

Improved Data Locality: Bucketing can help improve data locality in distributed environments.

Efficient Data Sampling: Bucketing can be beneficial for efficient data sampling. When sampling data from a bucketed table, Hive can sample a specific number of buckets instead of scanning the entire table.

Enhanced Query Optimization: Hive's query optimizer takes advantage of bucketing information during query planning.

Q 28. How to optimise Hive Performance? Explain in very detail.

ANS - Optimizing Hive performance involves several techniques and best practices. Here are several key aspects to consider for improving Hive performance:

1. Data Organization:

- Partitioning: Partitioning divides data into smaller, more manageable units. It helps reduce the amount of data processed per query, improves data pruning, and enhances query performance.

- Bucketing: Bucketing evenly distributes data within partitions, reduces data skew, and facilitates efficient join operations

2. Data Formats:

- Columnar Formats: Storing data in columnar formats like ORC (Optimized Row Columnar) or Parquet provides better compression, efficient column-wise scans, and predicate pushdown

- Compression: Applying compression to data reduces storage requirements and I/O costs. However, it's important to find the right balance between compression ratio and query performance.

3. Query Optimization:

- Predicate Pushdown: Pushing down filters and predicates closer to the data source reduces the amount of data processed during queries. This reduces I/O and improves query performance.

- Join Optimization: Enabling bucketing and using bucketed tables for join operations can leverage bucket map join, reducing data movement and improving join performance.

- Cost-Based Optimizer: Hive's Cost-Based Optimizer (CBO) analyzes statistics and query patterns to generate more efficient execution plans.

4. File System Considerations:

- HDFS Block Size: Set the HDFS block size to match the optimal file size for the data and workload. This reduces input/output operations and improves data locality.

- Input/Output Parallelism: Configuring the number of mappers and reducers based on the cluster resources and workload requirements ensures parallel execution and balanced resource utilization.

5. Hardware Considerations:

- Memory Configuration: Adjust memory settings like heap size, shuffle memory, and cache size appropriately to avoid out-of-memory errors and optimize query performance.

Data Skew Handling:

- Use skew join optimization techniques like skew join and map-side join to handle data skew and improve query performance for skewed data distributions.

7. Query Tuning:

- Limit Data Size: Reduce the data size processed per query by applying filters, limiting partitions

- Use Vectorization: Enable vectorized query execution (`hive.vectorized.execution.enabled=true`) to leverage CPU SIMD instructions and improve query processing speed.

8. Caching:

- Enable Caching: Utilize Hive's query result caching (`hive.cache.query.enabled=true`) to cache query results and reduce query execution time for repeated queries.

9. Hardware Scaling:

- Scale Out: Consider adding more nodes to the cluster to distribute the workload and improve query performance.

- Scale Up: Upgrade hardware resources like CPU, memory, and storage on existing nodes to handle larger workloads efficiently.

10. Monitoring and Tuning:

- Regularly monitor Hive performance metrics, collect query statistics, and analyze query execution plans to identify bottlenecks and areas for optimization.

Q 29. What is the use of Hcatalog?

ANS - HCatalog is a table and storage management layer in Apache Hive that provides a unified metadata and schema abstraction for accessing and managing data across different data processing frameworks

1. Metadata Management: HCatalog stores metadata information about tables, partitions, columns, and their associated schemas in a centralized metastore

2. Schema and Data Discovery: HCatalog enables users to discover and explore the available data schemas and tables in the Hadoop ecosystem.

3. Data Integration and Interoperability: HCatalog facilitates data integration by allowing different data processing frameworks like Hive, Pig, and MapReduce to share and access data through a common interface

5. Partition Management: HCatalog simplifies the management of partitioned tables by providing a unified interface for creating, altering, and querying

7. Data Security and Access Control: HCatalog integrates with Apache Ranger or other security frameworks to provide access control and data governance capabilities.

Q 30. Explain about the different types of join in Hive.

ANS - In Hive, there are several types of joins available for combining data from multiple tables

1. Inner Join: An inner join returns only the matching rows from both tables based on the join condition.

2. Left (Outer) Join: A left join returns all the rows from the left table and the matching rows from the right table based on the join condition.

3. Right (Outer) Join: A right join returns all the rows from the right table and the matching rows from the left table based on the join condition.

4. Full (Outer) Join: A full join returns all the rows from both tables, regardless of whether there is a match or not.

5. Left Semi Join: A left semi join returns the rows from the left table where there is a match in the right table based on the join condition.

6. Left Anti Join: A left anti join returns the rows from the left table where there is no match in the right table based on the join condition.

7. Cartesian Join (Cross Join): A cartesian join returns the combination of all rows from both tables

8. Map-side Join: A map-side join is an optimization technique used when one of the tables is small enough to fit in memory.

Q 31. Is it possible to create a Cartesian join between 2 tables, using Hive?

ANS - it is possible to create a Cartesian join between two tables using Hive. A Cartesian join, also known as a cross join, returns the combination of all rows from both tables, resulting in a potentially large result set.

To perform a Cartesian join in Hive, use the CROSS JOIN keyword or simply omit the join condition in the JOIN clause. Here's an example of how to create a Cartesian join between two tables:

In this example, table1 and table2 are the names of the two tables want to join. The CROSS JOIN keyword is used to specify that a Cartesian join should be performed.

32.Explain the SMB Join in Hive?

ANS – SMB (Sort-Merge-Bucket) join, also known as bucketed map join, is a join optimization technique in Hive that combines the benefits of bucketing and map-side joins to improve query performance

1. Bucketing: First, both tables involved in the join operation must be bucketed on the join key. Bucketing evenly distributes the data across multiple buckets based on a hash function applied to the join key.

2. Sorting: The data within each bucket of both tables is sorted based on the join key.

3. Map-Side Join: During the join operation, Hive checks if both tables are bucketed on the same join key

4. Merge: The sorted and bucketed data from both tables is merged in a single pass based on the join key.

Improved Performance: SMB join reduces the data movement and network overhead by performing the join operation directly within the mapper

Bucketing and Sorting Requirements: SMB join requires both tables to be bucketed and sorted on the join key

Q 33. What is the difference between order by and sort by which one we should use?

ANS - In Hive, both ORDER BY and SORT BY clauses are used for sorting the result set of a query.

ORDER BY:

- The ORDER BY clause is used to sort the entire result set by one or more columns.

- It guarantees the global order of the result set, meaning that all rows are sorted together.

- It involves a total ordering of the data, which requires collecting and processing all the data before returning the sorted result.

- ORDER BY can be used with any query, including queries that involve joins, aggregations, or complex filtering conditions.

SORT BY:

- The SORT BY clause is used to sort the data within each reducer before the final output.

- It performs a local sort within each reducer, meaning that each reducer independently sorts its portion of the data.

- It does not guarantee the global order of the result set across all reducers. The final result will be a concatenation of sorted partitions from each reducer.

Q 34. What is the usefulness of the DISTRIBUTED BY clause in Hive?

ANS - The DISTRIBUTED BY clause in Hive is used to specify the distribution of data across multiple reducers during the query execution. Data Skew Handling: When working with large datasets, it's common for the data to be unevenly distributed, leading to data skew. By using the DISTRIBUTED BY clause, redistribute the data evenly across reducers and mitigate the impact of data skew.

Load Balancing: The DISTRIBUTED BY clause helps to balance the workload across reducers. By specifying an appropriate distribution key ensure that the data is evenly distributed among the reducers, preventing hotspots and improving the overall performance of the query.

Join Performance: In a join operation, the DISTRIBUTED BY clause can be used to distribute the data of both tables on the join key.

Grouping and Aggregation: When performing grouping or aggregation operations, the DISTRIBUTED BY clause can be used to distribute the data based on the grouping key. Customized Data Distribution: The DISTRIBUTED BY clause provides flexibility in defining the data distribution strategy based on specific requirements.

35.How does data transfer happen from HDFS to Hive?

ANS - When data is transferred from HDFS (Hadoop Distributed File System) to Hive, the process involves several steps.

Data Ingestion: The data that needs to be transferred to Hive is typically first ingested into HDFS, Data Definition: Before transferring the data to Hive, a table needs to be created in Hive that defines the schema and structure of the data, Data Loading: Once the table is defined, the data can be loaded into the Hive table from HDFS. Hive provides various methods for data loading LOAD DATA INPATH: This command is used to load data from a specified HDFS path into a Hive table. It copies the data from the HDFS path to the table's location in HDFS. INSERT INTO: This command is used to insert data into an existing Hive table External Tables: Hive also supports external tables, where the data remains in its original location in HDFS, and Hive only creates a metadata entry to access the data. Data Serialization: During the data transfer, Hive serializes the data from HDFS into a binary format that can be stored in the underlying storage format of the Hive table.

36.Wherever (Different Directory) I run the hive query, it creates a new

metastore_db, please explain the reason for it?

ANS - The creation of a new metastore_db directory in different directories when running Hive queries is likely due to the default configuration of the Hive metastore. The metastore_db directory is the default location where the Hive metastore database is stored, which contains the metadata information of the Hive tables, partitions, schemas, and other related details. The reason for creating a new metastore_db directory in different directories is typically related to the initialization and configuration of the Hive metastore. When run Hive queries in different directories, Hive creates a new metastore_db directory in that specific directory to store the metadata for the Hive tables in that location. Hive maintains metadata separately for each Hive instance to ensure data integrity and isolation. By creating a new metastore_db directory in different directories, each instance of Hive can have its own dedicated metastore database, preventing any interference or conflicts between different instances. It's worth noting that the metastore_db directory is just the default location, and configure Hive to use a different location for the metastore database by modifying the Hive configuration settings.

Q 37.What will happen in case have not issued the command: 'SET

hive.enforce.bucketing=true;' before bucketing a table in Hive?

ANS - If not issued the command "SET hive.enforce.bucketing = true;" before bucketing a table in Hive, the table will still be bucketed, but Hive will not enforce bucketing during query execution. Bucketing the Table: When bucket a table in Hive without setting "hive.enforce.bucketing" to true, Hive will still create the buckets as specified in the DDL statement Query Execution: When perform queries on the bucketed table without enforcing bucketing, Hive will not optimize the query execution based on the bucketing property Limited Query Optimization: Without enforcing bucketing, Hive may still benefit from other query optimization techniques

Q 38.Can a table be renamed in Hive?

ANS - Yes, a table can be renamed in Hive using the RENAME TABLE command. The RENAME TABLE command allows to change the name of an existing table in Hive without altering the table's schema or data.

Q 39. Write a query to insert a new column(new_col INT) into a hive table at a

position before an existing column (x_col)

ANS – In Hive, cannot directly insert a column at a specific position within an existing table. Hive tables are based on the concept of a schema-on-read, where the column positions are not fixed or enforced. Instead, the column order is determined when querying the table.

If need to achieve a similar effect of inserting a column before an existing column, need to create a new table with the desired column order and copy the data from the original table into the new table. Here's an example of how accomplish this:

Insert data into the new table by selecting the existing columns from the original table:

```
INSERT INTO new_table
SELECT
  NULL AS new_col,
  x_col,
  col2,
  col3
FROM
  original_table;
```

Q 40. What is serde operation in HIVE?

ANS - In Hive, SerDe (Serializer/Deserializer) refers to the mechanism used to serialize data into a format that can be stored in HDFS (Hadoop Distributed File System) and deserialize it back into a structured format for query processing. It allows Hive to read and write data in different file formats and interpret the data's structure.

Q 41. Explain how Hive Deserializes and serialises the data?

ANS - In Hive, the process of deserialization and serialization is handled by the SerDe (Serializer/Deserializer) mechanism. The SerDe is responsible for converting data between its structured format (rows and columns) and a serialized format suitable for storage and processing.

Q 42. Write the name of the built-in serde in hive.

ANS – Hive provides several built-in SerDe (Serializer/Deserializer) implementations for different file formatsLazySimpleSerDe

OrcSerDe

AvroSerDe

ParquetHiveSerDe

JsonSerDe

RegexSerDe

ThriftSerDe

RCFileSerDe

Q 43.What is the need of custom Serde?

Ans - The need for a custom SerDe (Serializer/Deserializer) in Hive arises when are working with a file format or data structure that is not supported by the built-in SerDes.

1. Non-standard data formats: If dealing with a data format that is not supported by the built-in SerDes

2. Complex data structures: If data has a complex structure, such as nested objects, arrays, or maps, a custom SerDe can help properly serialize and deserialize the data.

3. Performance optimization: In some cases, a custom SerDe can be developed to improve the performance of data processing in Hive.

4. Custom data transformations: A custom SerDe allows to define custom data transformations during serialization and deserialization

5. Compatibility with legacy systems: If need to work with data from legacy systems or applications that have their own data format, a custom SerDe can be created to bridge the gap between Hive and those systems.

Q 44. Can you write the name of a complex data type(collection data types) in

Hive?

ANS - Certainly! Hive supports several complex data types, also known as collection data types

1. Array: An array is an ordered collection of elements of the same data type. It is represented as `array<data_type>`.

2. Map: A map is an unordered collection of key-value pairs, where the keys and values can have different data types.

3. Struct: A struct is a collection of fields with different data types. It allows to group related data together.

4. Union: A union is a data type that can store values of different data types. It is represented as `uniontype<data_type1, data_type2, ...>`

These complex data types provide flexibility in handling structured data within Hive tables

Q 45. Can hive queries be executed from script files? How?

ANS - Yes, Hive queries can be executed from script files. Hive provides a command-line interface (CLI) that allows to run Hive queries interactively or from script files.

To execute Hive queries from a script file

1. Create a text file and save it with a `.hql` extension

2. Open the script file in a text editor and write Hive queries in the file. Each query should be written on a separate line.

3. Save the script file after writing queries.

4. Open a terminal or command prompt and navigate to the directory where the script file is located.

Q 46. What are the default record and field delimiter used for hive text files?

ANS - In Hive, the default record delimiter used for text files is the newline character (`\n`). Each newline character indicates the end of a record in a text file.

The default field delimiter used for text files in Hive is a tab character (`\t`).

However, it's important to note that the default delimiters can be changed based on the configuration settings in Hive. The delimiters can be customized using the `hive.default.fileformat` property and the `hive.serde.field.delim` property in the Hive configuration.

Q 47. How do you list all databases in Hive whose name starts with s?

ANS - To list all databases in Hive whose names start with 's',

CODE-

SHOW DATABASES LIKE 's*';

This command uses the `LIKE` clause with a wildcard (`*`) to match databases whose names start with 's'. The wildcard (`*`) represents any number of characters

Q 48. What is the difference between LIKE and RLIKE operators in Hive?

ANS - In Hive, the `LIKE` and `RLIKE` operators are used for pattern matching in queries

1. LIKE Operator:
   - The `LIKE` operator performs simple pattern matching using wildcard characters.
   - It supports two wildcard characters: `%` and `_`.
   - The `%` wildcard matches any sequence of characters (including zero characters).
   - The `_` wildcard matches any single character.
2. RLIKE Operator:
   - The `RLIKE` operator performs pattern matching using regular expressions (regex).
   - It supports more advanced pattern matching using regular expression syntax.

- It allows for complex pattern matching and manipulation of text data.

The key difference between `LIKE` and `RLIKE` is that `LIKE` uses simple wildcard characters (`%` and `_`), while `RLIKE` uses regular expressions for more advanced pattern matching.

Q 49. How to change the column data type in Hive?

ANS - In Hive change the column data type of a table using the `ALTER TABLE` statement with the `CHANGE` clause.

```sql
ALTER TABLE table_name CHANGE column_name new_column_name new_data_type;
```

Explanation of the syntax:

- `table_name`: The name of the table containing the column
- `column_name`: The name of the column
- `new_column_name`: (Optional) The new name for the column,
- `new_data_type

```sql
ALTER TABLE employees CHANGE salary salary DOUBLE;
```

This command will modify the data type of the `salary` column in the `employees` table from `INT` to `DOUBLE`. If want to rename the column

```sql
ALTER TABLE employees CHANGE salary new_salary DOUBLE;
```

Q 50 .How will you convert the string ’51.2’ to a float value in the particular

column?

ANS - To convert the string `'51.2'` to a float value in a particular column in Hive, you can use the `CAST` function. The `CAST` function is used to explicitly convert one data type to another

sql

SELECT CAST('51.2' AS FLOAT) AS float_value;

In this example, the `CAST` function is used to convert the string `'51.2'` to a float data type. The result will be displayed as `float_value` in the output.

If want to update a column in a table with the converted float value, the `UPDATE` statement along with the `CAST` function:

sql

UPDATE table_name SET column_name = CAST('51.2' AS FLOAT) WHERE condition;

Replace `table_name` with the name of the table, `column_name` with the name of the column update, and `condition` with any specific conditions if needed.

51.What will be the result when you cast ‘abc’ (string) as INT?

ANS - attempt to cast the string `'abc'` as an `INT` in Hive, it will result in a `NULL` value. Hive will not be able to convert the non-numeric string `'abc'` into an integer, and as a result, the cast operation will produce a `NULL` value.

casting `'abc'` as an `INT`:

sql

SELECT CAST('abc' AS INT) AS int_value;

The output of this query will be:

int_value

NULL

Q 52. What does the following query do?

a. INSERT OVERWRITE TABLE employees

b. PARTITION (country, state)

c. SELECT ..., se.cnty, se.st

d. FROM staged_employees se;

ANS - The given query performs an `INSERT OVERWRITE` operation in Hive to populate the `employees` table using data from the `staged_employees` table. Let's break down the query into its components:

```sql
INSERT OVERWRITE TABLE employees
PARTITION (country, state)
SELECT ..., se.cnty, se.st
FROM staged_employees se;
```

a. `INSERT OVERWRITE TABLE employees`: This line specifies that the data will be inserted into the `employees` table. The `INSERT OVERWRITE` operation overwrites any existing data in the table.

b. `PARTITION (country, state)`: This line indicates that the `employees` table is partitioned by the `country` and `state` columns. Partitioning is a way of organizing data in Hive for efficient querying.

c. `SELECT ..., se.cnty, se.st`: This line specifies the columns to be selected from the `staged_employees` table. The `...` represents the other columns selected from `staged_employees` that are not explicitly mentioned.

d. `FROM staged_employees se`: This line specifies the source table from which the data will be retrieved. The table `staged_employees` is referenced with the alias `se`.

53.Write a query where you can overwrite data in a new table from the

existing table.

ANS - To overwrite data in a new table from an existing table in Hive, use the `INSERT OVERWRITE TABLE` statement along with a `SELECT` query. Here's an example:

```sql
INSERT OVERWRITE TABLE new_table
SELECT column1, column2, ...
FROM existing_table;
```

```

In this example, `new_table` is the name of the new table want to overwrite the data, and `existing_table` is the name of the existing table retrieve the data.

Make sure that the schema of the new table (`new_table`) matches the schema of the `SELECT` query's result. The `SELECT` query specifies the columns want to select from the existing table (`existing_table`) and insert into the new table.

By using `INSERT OVERWRITE TABLE`, the existing data in the new table will be completely replaced with the data from the existing table.

Q 54. What is the maximum size of a string data type supported by Hive?

Explain how Hive supports binary formats.

ANS - In Hive, the maximum size of a `STRING` data type is 2^31-1 (2,147,483,647) characters.

Q 55. What File Formats and Applications Does Hive Support?

ANS - Hive supports various file formats and integrates with several external applications.

File Formats:

1. Text File: Plain text files are one of the simplest file formats supported by Hive. Text files are typically stored in Hadoop Distributed File System (HDFS) or local file systems.

2. SequenceFile: SequenceFiles are binary files that store key-value pairs. They are a compact file format optimized for high-speed reading and writing.

3. RCFile (Record Columnar File): RCFile is a columnar file format that stores data in a column-wise manner.

Applications:

1. Apache Spark: Hive can integrate with Apache Spark, a fast and general-purpose distributed computing framework.

2. Apache Tez: Tez is an execution engine built on top of Apache Hadoop YARN.

3. Apache Flink: Hive can integrate with Apache Flink, a stream processing framework

4. Apache Kafka: Hive can read data directly from Apache Kafka, a distributed streaming platform.
```

Q 56. How do ORC format tables help Hive to enhance its performance?

ANS - ORC (Optimized Row Columnar) format tables in Hive provide several performance enhancements, contributing to improved query execution and efficiency.

1. Columnar Storage: ORC stores data in a columnar format rather than row-based, which allows for better compression and efficient column-level operations. Predicate Pushdown: ORC format supports predicate pushdown, which means that filtering operations can be pushed down to the storage layer. Compression: ORC provides advanced compression techniques, such as dictionary encoding, run-length encoding, and lightweight compression algorithms like Snappy and Zlib. Indexing: ORC format supports different types of indexes, such as Bloom filters and min/max indexes.

Q 57. How can Hive avoid mapreduce while processing the query?

ANS - Hive can avoid MapReduce while processing a query by utilizing other execution engines or frameworks that provide faster and more efficient processing.

Tez Execution Engine: Hive can leverage Apache Tez as an alternative execution engine. Tez provides a directed acyclic graph (DAG) execution model, which allows for better performance and reduced overhead compared to traditional MapReduce. By configuring Hive to use Tez as the execution engine (`SET hive.execution.engine=tez;`), Hive can execute queries directly on Tez, bypassing MapReduce.Spark Execution Engine: Hive can also integrate with Apache Spark, a fast and general-purpose distributed computing framework. Vectorized Query Execution: Hive provides a vectorized query execution mode (`SET hive.vectorized.execution.enabled=true;`) that processes data in batches using columnar operations, resulting in improved performance. LLAP is an in-memory execution engine for Hive that allows for interactive and real-time query processing. LLAP caches data in memory and executes queries without going through MapReduce.

Q 58. What is view and indexing in hive?

ANS - In Hive, a view is a virtual table that does not store any data of its own but is defined based on a queryViews are created using the `CREATE VIEW` statement and can be queried like regular tables

Data Abstraction: Views abstract the underlying complexity of the data by presenting a simplified and customized representation of the data Security and Access Control: Views can be used to enforce security and access control by limiting the visibility of certain columns or rows to specific users or roles Simplified Querying: Views can encapsulate complex queries or frequently used joins, making it easier for users to write queries. Instead of writing lengthy and complex queries repeatedly

Q 59. Can the name of a view be the same as the name of a hive table?

ANS - No, the name of a view cannot be the same as the name of a Hive table. In Hive, table names and view names are stored in the same namespace, and they need to be unique within that namespace. To resolve this issue either choose a different name for the view or rename the existing table before creating the view with the desired name. Renaming a table can be done using the `ALTER TABLE` statement in Hive.

Q 60. What types of costs are associated in creating indexes on hive tables?

ANS - Creating indexes on Hive tables comes with certain costs, both in terms of performance and storage overhead. The costs associated with creating indexes on Hive tables include:

Index Creation Overhead: When creating an index on a Hive table, there is an initial overhead in terms of time and resources required to build the index. Index Maintenance Overhead: Once an index is created, it needs to be maintained and updated along with the underlying table. Whenever data is inserted, updated, or deleted in the table, corresponding changes need to be reflected in the index. Storage Overhead: Indexes require additional storage space to store the index data structures. The size of the index depends on various factors such as the number of indexed columns, the size of the indexed data, and the chosen index type. Query Performance Impact: Although indexes can improve query performance by allowing for efficient data lookup

Q 61. Give the command to see the indexes on a table.

ANS -

SHOW INDEXES ON table_name;

Replace `table_name` with the actual name of the table for which want to view the indexes. This command will display the information about the indexes created on the specified table, including the index name, indexed columns, index type, and other relevant details.

DESCRIBE EXTENDED table_name;

Q 62. Explain the process to access subdirectories recursively in Hive queries.

ANS –

In Hive, accessing subdirectories recursively can be achieved by using a combination of Hive built-in functions and recursive query constructs

1. Set the `hive.mapred.supports.subdirectories` property to `true` before executing the query. This property enables Hive to support subdirectory scanning.

Use the `INPUT__FILE__NAME` virtual column in query to retrieve the full path of the input file. This column contains the complete file path

3. Apply the necessary transformations or filtering on the `INPUT__FILE__NAME` column to process the subdirectories as needed use Hive built-in functions like `regexp_extract`, `substr`, or `split` to extract specific subdirectory information from the file path.

4. To traverse subdirectories recursively utilize a recursive query using Hive's `WITH RECURSIVE` clause. This allows to iterate over subdirectories and perform the desired operations recursively.

THE CODE

```
SET hive.mapred.supports.subdirectories=true;

WITH RECURSIVE subdirectories AS (
  SELECT INPUT__FILE__NAME AS file_path
  FROM your_table
  WHERE your_condition
  UNION ALL
  SELECT subfile.file_path
  FROM subdirectories
  JOIN (
    SELECT INPUT__FILE__NAME AS file_path
    FROM your_table
    WHERE your_condition
  ) subfile
  ON subfile.file_path LIKE CONCAT(subdirectories.file_path, '/%')
)
SELECT *
FROM subdirectories;
```

Q 63. If you run a select * query in Hive, why doesn't it run MapReduce?

ANS - When run a simple SELECT * query in Hive without any additional operations or transformations, it doesn't require the execution of a MapReduce job. This is because Hive leverages a feature called "Hive Optimization" or "Hive Query Optimization" to optimize and expedite certain types of queries, including simple projection queries like SELECT *

Q 64.What are the uses of Hive Explode?.

ANS - The Hive function explode is used to unnest or explode arrays or map data types in Hive. It takes a complex data type column as input and generates a new row for each element in the array or

map. The explode function is particularly useful for querying and analyzing nested data structures in Hive

Q 65. What is the available mechanism for connecting applications when we

run Hive as a server?

ANS –

When running Hive as a server, there are multiple mechanisms available for connecting applications to interact with Hive and execute queries. JDBC (Java Database Connectivity): Hive provides a JDBC driver that allows applications written in Java or other languages that support JDBC to connect to Hive. JDBC provides a standardized API for connecting to databasesODBC (Open Database Connectivity): ODBC is another widely used standard for connecting applications to databases Thrift API: Hive exposes a Thrift service, which is a remote procedure call (RPC) framework, allowing applications to communicate with Hive using various programming languages

Q 66. Can the default location of a managed table be changed in Hive?

ANS - No, the default location of a managed table in Hive cannot be changed.

In Hive, a managed table is a table where the data and metadata are managed by Hive itself. create a managed table, Hive determines the default location where the table's data will be stored. By default, the data is stored in the Hive warehouse directory, typically located in HDFS. The location is determined based on the Hive configuration property hive.metastore.warehouse.dir.

Q 67.What is the Hive ObjectInspector function?

ANS – In Apache Hive, the ObjectInspector is a fundamental component of the Hive infrastructure that helps in the processing of data. It is a class that enables Hive to inspect the internal structure of data and provides a way to access and manipulate the data stored in different Hive data types.

1. Data Inspection:
2. Data Manipulation:
3. SerDe Integration
4. Customization

Q 68. What is UDF in Hive?

ANS - In Hive, UDF stands for User-Defined Function. It refers to a custom function created by users to perform specific operations on data in Hive queries. UDFs allow users to extend the functionality of Hive by implementing their own custom logic.

Custom Functions: UDFs are functions that users define and implement themselves using programming languages supported by Hive, such as Java, Python, or Scala. Hive Function Types: UDFs in Hive can be classified into different types based on the number of input arguments and the return value .Registration: Before using a UDF in Hive, it needs to be registered with the Hive metastore. Registration involves specifying the function name, input parameter types, return type, and the class or script that implements the UDF logic. Once registered, the UDF becomes available for use in Hive queries.Usage in Queries: UDFs can be used in Hive queries like any other built-in function. Users can invoke the UDF by its registered name and pass input arguments as required. UDFs can be used in SELECT statements, WHERE clauses, GROUP BY clauses, and other parts of the query where functions are allowed.

Q 69.Write a query to extract data from hdfs to hive.

ANS - To extract data from HDFS and load it into Hive, use the LOAD DATA INPATH command in Hive

IN SQL-

```
LOAD DATA INPATH '/path/to/input/files'
OVERWRITE INTO TABLE your_table
PARTITION (date='2023-04-24')
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

Q 70.What is TextInputFormat and SequenceFileInputFormat in hive.

ANS - TextInputFormat:

TextInputFormat is the default input format in Hive for reading plain text files.

It treats each line in the input file as a separate record.

The input file is divided into input splits, and each split is processed by a separate mapper in a MapReduce job.

By default, each line of the input file is considered a separate row in Hive, and the fields within the line are delimited by a specific character (usually a tab or a comma).

SequenceFileInputFormat:

SequenceFileInputFormat is an input format in Hive used for reading data stored in the SequenceFile format.

SequenceFiles are binary files that store data in key-value pairs, where both the key and value can have complex data types.

SequenceFiles are commonly used in Hadoop and Hive for storing large amounts of structured or semi-structured data efficiently.

SequenceFileInputFormat reads the data from SequenceFiles and presents it to Hive as key-value pairs, where Hive can interpret the key and value based on the specified data types.

Q 71. How can you prevent a large job from running for a long time in a hive?

ANS -

Enable Query Optimization: Hive provides various optimization techniques to improve query performance. enable query optimization by setting the hive.optimize. properties to true. Partitioning and Bucketing: Partitioning and bucketing are techniques used to organize data in a Hive table. Partitioning involves dividing the data into logical partitions based on a specific column, which can improve query performance when filtering on that column. Use Appropriate File Formats: Choose the right file format for data based on use case. File formats like ORC (Optimized Row Columnar) and Parquet are highly efficient for Hive queries due to their compression and columnar storage capabilitiesTune Hive Configuration: Adjusting the Hive configuration parameters can significantly impact query performance. Spark Execution Engines: Hive supports multiple execution engines like MapReduce, Tez, and Spark. Tez and Spark provide significant performance improvements over MapReduce for most workloads.

Data Skew Handling: Data skew occurs when certain keys or values in a dataset have much larger proportions than others, leading to an imbalance in processing and performance degradation.

Q 72.When do we use explode in Hive?

ANS - In Hive, the explode function is used to transform a complex data type column into multiple rows, with each row containing an element from the original column. It is particularly useful when working with arrays or nested data structures

Q 73.Can Hive process any type of data formats? Why? Explain in very detail

ANS - Hive is capable of processing a wide range of data formats. The reason for this versatility lies in Hive's ability to support pluggable SerDes (Serialization/Deserialization) and file formats.

Q 74.Whenever we run a Hive query, a new metastore_db is created. Why?

ANS - The creation of a new metastore_db directory when running a Hive query can occur due to a misconfiguration or improper initialization of the Hive Metastore. The metastore_db directory is the default location where Hive stores its metadata, including database and table information, in an embedded Apache Derby database.

Q 75.Can we change the data type of a column in a hive table? Write a

complete query.

ANS - Yes, it is possible to change the data type of a column in a Hive table. However, there are some limitations and considerations when altering column data types in Hive.

To change the data type of a column in a Hive table, use the ALTER TABLE statement with the CHANGE COLUMN clause.

Q 76.While loading data into a hive table using the LOAD DATA clause, how

specify it is a hdfs file and not a local file ?

ANS -- When using the LOAD DATA clause to load data into a Hive table, specify that the file is located in HDFS (Hadoop Distributed File System) and not on the local file system by providing the HDFS path of the file.

To specify that the file is located in HDFS, need to specify the full HDFS path starting with hdfs:// followed by the HDFS directory where the file is located

Q 77.What is the precedence order in Hive configuration?

ANS -- Session-level Configuration: Configuration set explicitly for the current session using the SET command takes the highest precedence.

Database-level Configuration: Configuration set at the database level using the ALTER DATABASE command.

Table-level Configuration: Configuration set at the table level using the ALTER TABLE command.

Hive-site.xml Configuration: Configuration set in the hive-site.xml file, which is the main Hive configuration file.

Hive Configuration Variables: Configuration set using Hive configuration variables. These variables can be set in various ways, such as through environment variables, command-line arguments, or custom configuration files.

Q 78.Which interface is used for accessing the Hive metastore?

ANS -- The interface used for accessing the Hive metastore is called the Hive Metastore Thrift Service. It provides a Thrift-based API for interacting with the Hive metastore, which is a central repository that stores metadata information about Hive tables, databases, partitions, and other related entities.

Q 79.Is it possible to compress json in the Hive external table ?

ANS -- Yes, it is possible to compress JSON data in a Hive external table. Hive supports various compression codecs that can be used to compress the data stored in files, including JSON files.

Q 80.What is the difference between local and remote metastores?

ANS

The difference between local and remote metastores in Hive lies in the location and accessibility of the metastore database.

Local Metastore:

In a local metastore setup, the metastore database is co-located with the Hive server or the Hive client.

The metastore database is stored on the local file system of the machine where Hive is installed.

The local metastore is typically used in standalone or single-node Hive setups where Hive server and metastore are running on the same machine.

Remote Metastore:

In a remote metastore setup, the metastore database is hosted on a separate machine or a dedicated metastore server.

The metastore database can be hosted on a different machine or a cluster, accessible over the network.

The remote metastore is commonly used in distributed setups where multiple Hive instances or clients need to share the same metastore and access metadata concurrently.

Q 81.What is the purpose of archiving tables in Hive?

ANS --The purpose of archiving tables in Hive is to preserve and retain data in a compressed and immutable format for long-term storage or historical analysis.

Data Retention:

Storage Optimization:

Cost Reduction:

Data Analysis:

Q 82.What is DBPROPERTY in Hive?

ANS -- In Hive, DBPROPERTY is a function used to retrieve the value of a property associated with a database. It allows to access and query the properties or metadata of a database in Hive.

Q 83.Differentiate between local mode and MapReduce mode in Hive.

ANS - Local Mode:

In local mode, Hive runs the query on the local machine without involving any distributed processing framework like MapReduce.

The data is read from the local file system or Hadoop Distributed File System (HDFS) and processed locally.

Local mode is primarily used for small datasets or for testing/debugging purposes when want to quickly run queries without the overhead of setting up a full MapReduce job.

MapReduce mode, Hive leverages the power of the Hadoop ecosystem and executes queries using the MapReduce framework.

The data is distributed across multiple nodes in a cluster, and the processing is performed in a parallel and distributed manner.

MapReduce mode is designed for handling large-scale datasets that require distributed processing for improved performance and scalability.