PART – A

1. What is HBase?

HBase is a distributed, column-oriented database system that runs on top of the Hadoop Distributed File System (HDFS).

2. What is the default mode of replication in HBase?

HBase uses synchronous replication by default, which means that data is written to multiple nodes in the cluster before a write operation is considered successful.

3. What is the difference between HBase and RDBMS?

HBase is a NoSQL database system that is designed for scalability and high availability, while RDBMS is a relational database system that is designed for data consistency and transactional support.

4. Explain why to use HBase?

High capacity storage system
Distributed design to cater large tables
Column-Oriented Stores
Horizontally Scalable
High performance & Availability
Base goal of HBase is millions of columns, thousands of versions and billions of rows
Unlike HDFS (Hadoop Distribute File System), it supports random real time CRUD operations

5. What is Cassandra?

Cassandra is a distributed, NoSQL database system designed to handle large amounts of data across many commodity servers while providing high availability and fault tolerance.

6. What is the role of the Cassandra Coordinator?

The Cassandra Coordinator is responsible for routing requests to the appropriate nodes in the cluster based on the partition key of the data being accessed.

7. What is the use of Cassandra Query Language (CQL)?

Cassandra Query Language (CQL) is a query language used to interact with Cassandra databases. It is similar to SQL in syntax and structure, but is optimized for use with NoSQL data models.

8. What is Pig Latin?

Pig Latin is a high-level language used to process large datasets in Apache Hadoop. It is similar to SQL and allows developers to write data processing jobs in a simple and concise way using a set of predefined operators.

9. What is the difference between Pig and Hive?

Pig and Hive are both high-level languages used for data processing in Apache Hadoop, but there are some key differences between them. Pig is a procedural language that focuses on the data flow, while Hive is a declarative language that focuses on the data structure. Pig is suitable for ad-hoc data processing and exploration, while Hive is more suited for querying and analyzing large datasets.

10. What is Hive in Hadoop?

Hive is a data warehousing tool built on top of Hadoop that allows developers to query and analyze large datasets stored in Hadoop Distributed File System (HDFS) using SQL-like syntax. It provides a SQL interface to Hadoop and allows users to perform data analysis and exploration in a familiar way.

11. What is the difference between Hive and Pig?

Hive and Pig are both high-level languages used for data processing in Apache Hadoop, but there are some key differences between them. Hive is a declarative language that focuses on the data structure, while Pig is a procedural language that focuses on the data flow. Hive is more suited for querying and analyzing large datasets, while Pig is suitable for ad-hoc data processing and exploration. Additionally, Hive is optimized for structured data, while Pig can handle both structured and unstructured data.

12. What are Buckets in Hive?

Buckets in Hive are a way of partitioning data based on the hash value of a column or set of columns. A bucketed table is divided into a fixed number of buckets, and each bucket contains a subset of the data based on the hash value of the specified columns. Bucketing is a technique used in Hive to optimize queries that involve joins, as it ensures that rows with the same hash value are stored in the same bucket, making join operations more efficient.

13. What is HiveQL?

HiveQL is the SQL-like query language used in Apache Hive for querying and analyzing data stored in Hadoop Distributed File System (HDFS). It is a declarative language that allows users to write SQL-like queries to manipulate data in Hadoop, including filtering, aggregating, and joining large datasets.

14. What are the uses of Pig in Hadoop?

Pig is a high-level data processing language used in Apache Hadoop for processing and analyzing large datasets. Some of the common uses of Pig include:

Data processing: Pig can be used to process large volumes of data stored in Hadoop, including filtering, cleaning, transforming, and aggregating data.

Ad-hoc analysis: Pig allows developers to perform ad-hoc data analysis and exploration using a simple and concise language.

Data integration: Pig can be used to integrate data from multiple sources, including structured and unstructured data.

ETL (Extract, Transform, Load): Pig can be used for ETL processing, which involves extracting data from various sources, transforming it into a common format, and loading it into a target system.

Machine learning: Pig can be used for machine learning tasks, such as building models and making predictions based on large datasets.

15. What is a subquery in HiveQL?

A subquery in HiveQL is a query that is nested inside another query. It is used to retrieve data that will be used as input for another query. Subqueries can be used in the SELECT, FROM, and WHERE clauses of a HiveQL query and can be correlated or non-correlated. Correlated subqueries are evaluated for each row of the outer query, while non-correlated subqueries are evaluated only once.

PART – B

1. Analyze in detail about Hive data manipulation, queries, and data types.
2. Discuss about :
   (i) Pig
   (ii) Pig Latin
3. Demonstrate about HBase and Hbase clients in detail.
4. Discuss the about Cassandra clients.
5. Show the method of invoking the Grunt shell.
6. Explain about Grunt in detail.

Additional Questions:

1. Implement the technique that can be used to minimize the amount of intermediate data and improve the performance of your Pig script.

   One technique that can be used to minimize the amount of intermediate data and improve the performance of a Pig script is to use the COMBINER function. The COMBINER function allows for the aggregation of intermediate data on the Map side of a MapReduce job, reducing the amount of data that needs to be transferred to the Reduce side for further processing.

   Here's an example of how to use the COMBINER function in a Pig script:

   ```
   -- Load data from a file
   data = LOAD '/path/to/file' USING PigStorage(',');

   -- Group data by key
   grouped_data = GROUP data BY $0;

   -- Calculate the sum for each group
   summed_data = FOREACH grouped_data GENERATE group, SUM(data.$1);

   -- Use the Combiner function to aggregate data on the Map side
   summed_data = GROUP summed_data ALL;

   -- Store the results to a file
   STORE summed_data INTO '/path/to/output' USING PigStorage(',');
   ```

   In this example, we first load data from a file and group it by a key. We then calculate the sum for each group using the SUM function. To use the COMBINER function, we group the data again using the ALL keyword, which tells Pig to aggregate the data on the Map side. This reduces the amount of data that needs to be transferred to the Reduce side for further processing.

   By using the COMBINER function in this way, we can significantly reduce the amount of intermediate data generated by our Pig script and improve its performance.

2. Explain about developing and testing Pig Latin Scripts.

   Developing and testing Pig Latin scripts involves the following steps:

   Install Pig: Install Pig on your system by downloading the binary distribution from the Apache Pig website and following the installation instructions.

   Write the script: Write your Pig Latin script in a text editor, such as Notepad or Sublime Text. Save the script with a .pig file extension.

   Load data: Use the LOAD statement to load data into Pig from a file or Hadoop Distributed File System (HDFS).

   Transform data: Use Pig Latin statements to transform the data as required, including filtering, cleaning, and aggregating data.

   Store data: Use the STORE statement to store the results of your data processing back to HDFS or a file.

Test the script: Run the script using the Pig interpreter or command-line interface to test the results. You can also use Pig's built-in debugging tools to identify and fix any errors in the script.

Here is an example Pig Latin script that loads data from a file, filters the data, and stores the results back to a file:

```
-- Load data from a file
data = LOAD '/path/to/file' USING PigStorage(',');

-- Filter the data
filtered_data = FILTER data BY $0 >= 18 AND $2 == 'Male';

-- Store the results to a file
STORE filtered_data INTO '/path/to/output' USING PigStorage(',');
```

To run the script, save it to a file with a .pig extension, and then run the following command in the terminal:

```
pig -x local /path/to/script.pig
```

This will run the script in local mode and store the results in a file called "output" in the specified directory. You can then inspect the output file to verify the results.

3. Identify the factors need to consider before making the switch from Relational Database to Cassandra.

Here are some factors to consider before making the switch:

Data Model: Cassandra is a NoSQL database that uses a different data model than traditional relational databases. Before migrating, it's important to analyze the data model of the existing database and design a new data model that is suitable for Cassandra's data model. This will involve a careful analysis of the data access patterns and queries that will be used with Cassandra.

Scalability: Cassandra is designed to scale horizontally, meaning that it can add nodes to the cluster to increase performance and capacity. Before migrating, it's important to analyze the existing database's scalability requirements and ensure that Cassandra can meet those requirements.

Data Consistency: Cassandra is a eventually consistent database, meaning that data changes are not immediately propagated to all nodes in the cluster. This can lead to some inconsistencies in the data. Before migrating, it's important to consider the consistency requirements of the data and determine if Cassandra can meet those requirements.

Data Volume: Cassandra is designed to handle large volumes of data, but it's important to ensure that the new cluster can handle the volume of data that will be migrated from the existing database. It's also important to consider the ongoing growth of the data and ensure that the new cluster can handle that growth.

Data Access Patterns: Before migrating, it's important to analyze the data access patterns of the existing database and determine if they will be suitable for Cassandra. Cassandra is optimized for specific access patterns, and it's important to design the new data model to take advantage of those patterns.

Security: Cassandra has a different security model than traditional relational databases, and it's important to ensure that the new security model can meet the existing security requirements.

Expertise: Migrating to Cassandra requires expertise in both Cassandra and the existing database. It's important to ensure that the migration team has the necessary expertise to make the migration successful.

Overall, migrating from a relational database to Cassandra requires careful planning and consideration of many factors. It's important to thoroughly analyze the existing database and ensure that Cassandra can meet the performance, scalability, consistency, and security requirements of the data.

4. Hadoop-specific file system types and HDFS commands.
Hadoop is a distributed computing framework that allows for the processing of large datasets across clusters of commodity hardware. Hadoop has its own file system, known as the Hadoop Distributed File System (HDFS), which is designed to handle large datasets by storing them across multiple machines.

There are several Hadoop-specific file system types that can be used with Hadoop, including:

HDFS: This is the primary file system used with Hadoop. It is designed to handle large files and large datasets by splitting them into smaller chunks, which are then distributed across multiple machines in the cluster. HDFS also includes features like replication, fault tolerance, and data locality, which help to ensure high availability and performance.

Hadoop Archive (HAR): This is a file format used for archiving HDFS data. HAR files are created by packing smaller files into larger ones, which can help to improve performance when reading and writing data.

Hadoop Compatible File System (HCFS): This is a generic interface that allows Hadoop to work with other file systems, such as Amazon S3 or Google Cloud Storage.

Here are some common HDFS commands:

hadoop fs -ls: This command lists the files and directories in the current directory on HDFS.
hadoop fs -mkdir: This command creates a new directory on HDFS.
hadoop fs -put: This command copies a file from the local file system to HDFS.
hadoop fs -get: This command copies a file from HDFS to the local file system.
hadoop fs -rm: This command deletes a file or directory from HDFS.
hadoop fs -cat: This command displays the contents of a file on HDFS.
hadoop fs -du: This command shows the amount of disk space used by a file or directory on HDFS.
hadoop fs -chmod: This command changes the permissions of a file or directory on HDFS.
hadoop fs -chown: This command changes the owner of a file or directory on HDFS.
hadoop fs -mv: This command moves a file or directory on HDFS to a new location.

5. Discuss about the technologies used to handle big data.
Handling big data requires technologies that can efficiently store, process, and analyze large and complex datasets. Here is a summary of the key technologies used to handle big data:

Hadoop: Hadoop is a distributed computing framework that allows for the storage and processing of large datasets across clusters of commodity hardware. It includes a distributed file system (HDFS) and a MapReduce processing engine that can handle large-scale data processing.

NoSQL databases: NoSQL databases are designed to handle large and unstructured data, making them well-suited for big data applications. Examples of NoSQL databases include MongoDB, Cassandra, and HBase.

Apache Spark: Apache Spark is an open-source data processing engine that can handle batch processing, stream processing, and machine learning tasks. It can run on top of Hadoop or other cluster managers.

Apache Kafka: Apache Kafka is a distributed streaming platform that can handle high-volume, real-time data streams. It can integrate with other big data technologies like Hadoop and Spark.

Cloud-based storage and computing: Cloud platforms like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) offer scalable storage and computing resources that can be used to handle big data. They also offer managed services for technologies like Hadoop and Spark.

Data warehouses: Data warehouses are specialized databases designed for storing and analyzing large amounts of structured data. Examples of data warehouse technologies include Amazon Redshift, Snowflake, and Google BigQuery.

Data lakes: Data lakes are repositories that can store large amounts of raw data in its native format. They can integrate with various big data technologies and enable data discovery, analysis, and processing at scale. Examples of data lake technologies include Amazon S3, Microsoft Azure Data Lake Storage, and Google Cloud Storage.

6. Discuss how to implement hive in Big Data.

Hive is a data warehouse infrastructure built on top of Hadoop that provides an SQL-like interface for querying and analyzing large datasets. Here's how to implement Hive in a Big Data environment:

Set up a Hadoop cluster: Hive runs on top of Hadoop, so the first step is to set up a Hadoop cluster. This involves installing and configuring the Hadoop Distributed File System (HDFS) and the MapReduce processing engine.

Install Hive: Once the Hadoop cluster is set up, you can install Hive. Hive can be installed on a master node or on a separate node, depending on your cluster configuration.

Create a Hive database: In Hive, data is organized into databases. To create a database, you can use the CREATE DATABASE command in the Hive shell or in a Hive script.

Define tables: In Hive, tables define the structure of the data and how it is stored in HDFS. You can create tables in Hive using SQL-like commands or by importing existing data.

Load data: After you define a table, you can load data into it. Hive provides several ways to load data, including LOAD DATA and INSERT INTO commands.

Query data: Once data is loaded into Hive, you can query it using SQL-like commands. Hive translates SQL-like commands into MapReduce jobs that run on the Hadoop cluster.

Optimize queries: Hive allows for query optimization to improve performance. This includes partitioning data, using indexing, and using the appropriate file formats for data storage.

Integrate with other tools: Hive can be integrated with other tools and technologies, such as Pig for data processing, Oozie for workflow scheduling, and Spark for in-memory data processing.

Overall, implementing Hive in a Big Data environment involves setting up a Hadoop cluster, installing Hive, defining databases and tables, loading data, querying data, optimizing queries, and integrating with other tools as needed.