TASK 1:

1) What is NoSQL database..?

NoSQL is an approach to databases that represents a shift away from traditional relational database management systems (RDBMS). To define NoSQL, it is helpful to start by describing SQL, which is a query language used by RDBMS. Relational databases rely on tables, columns, rows, or schemas to organize and retrieve data. In contrast, NoSQL databases do not rely on these structures and use more flexible data models. NoSQL can mean "not SQL" or "not only SQL." As RDBMS have increasingly failed to meet the performance, scalability, and flexibility needs that next-generation, data-intensive applications require, NoSQL databases have been adopted by mainstream enterprises. NoSQL is particularly useful for storing unstructured data, which is growing far more rapidly than structured data and does not fit the relational schemas of RDBMS. Common types of unstructured data include: user and session data; chat, messaging, and log data; time series data such as IOT and device data; and large objects such as video and images.

Benefits of NoSQL :

NoSQL databases offer enterprises important advantages over traditional RDBMS, including:

- ***Scalability:** NoSQL databases use a horizontal scale-out methodology that makes it easy to add or reduce capacity quickly and non-disruptively with commodity hardware. This eliminates the tremendous cost and complexity of manual sharding that is necessary when attempting to scale RDBMS.*
- ***Performance:** By simply adding commodity resources, enterprises can increase performance with NoSQL databases. This enables organizations to continue to deliver reliably fast user experiences with a predictable return on investment for adding resources—again, without the overhead associated with manual sharding.*
- ***High Availability:** NoSQL databases are generally designed to ensure high availability and avoid the complexity that comes with a typical RDBMS architecture that relies on primary and secondary nodes. Some "distributed" NoSQL databases use a masterless architecture that automatically distributes data equally among multiple resources so that the application remains available for both read and write operations even when one node fails.*
- ***Global Availability:** By automatically replicating data across multiple servers, data centers, or cloud resources, distributed NoSQL databases can minimize latency and ensure a consistent application experience wherever users are located. An added benefit is a significantly reduced database management burden from manual RDBMS configuration, freeing operations teams to focus on other business priorities.*
- ***Flexible Data Modeling:** NoSQL offers the ability to implement flexible and fluid data models. Application developers can leverage the data types and query options that are the most natural fit to the specific application use case rather than those that fit the database schema. The result is a simpler interaction between the application and the database and faster, more agile development.*
-

2. How does data get stored in NoSQl database?

There are various NoSQL Databases. Each one uses a different method to store data. Some might use column store, some document, some graph, etc., Each database has its own unique characteristics.

- ***Key-value data stores:*** *Key-value NoSQL databases emphasize simplicity and are very useful in accelerating an application to support high-speed read and write processing of non-transactional data. Stored values can be any type of binary object (text, video, JSON document, etc.) and are accessed via a key. The application has complete control over what is stored in the value, making this the most flexible NoSQL model. Data is partitioned and replicated across a cluster to get scalability and availability. For this reason, key value stores often do not support transactions. However, they are highly effective at scaling applications that deal with high-velocity, non-transactional data.*
- ***Document stores:*** *Document databases typically store self-describing JSON, XML, and BSON documents. They are similar to key-value stores, but in this case, a value is a single document that stores all data related to a specific key. Popular fields in the document can be indexed to provide fast retrieval without knowing the key. Each document can have the same or a different structure.*
- ***Wide-column stores:*** *Wide-column NoSQL databases store data in tables with rows and columns similar to RDBMS, but names and formats of columns can vary from row to row across the table. Wide-column databases group columns of related data together. A query can retrieve related data in a single operation because only the columns associated with the query are retrieved. In an RDBMS, the data would be in different rows stored in different places on disk, requiring multiple disk operations for retrieval.*
- ***Graph stores:*** *A graph database uses graph structures to store, map, and query relationships. They provide index-free adjacency, so that adjacent elements are linked together without using an index.*

Example: MongoDB is an document store, where data is stored as ***Key: Value*** pairs in JSON format.

3. What is a column family in HBase?

Column families are the base storage mechanism in HBase.   AHBase table is comprised of one or more column families, each of which is stored in a separate set of regionfiles sharing a common key.

To express it in terms of an RDBMS, a column family is roughly analogous to a RDBMS table with the rowkey as a clustered primary key index.   AHBase table would then be a view which does a full outer join on a set of RDBMS tables which all share the same primary key (thus having a 1:1 relationship).   In this analogy, HBase region files map to pages in an RDBMS.

4. How many maximum number of columns can be added to HBase table?

There is no hard limit to number of columns in HBase , we can have more than 1 million columns but usually three column families are recommended (not more than three) because there is one MemStore(Its a write cache which stores new data before writing it into Hfiles) per Column Family, when one is full, they all flush.The more we add column families there will be more MemStore created and Memstore flush will be more frequent. It will degrade the performance.Depending on our data access patterns, we should consider wide table vs tall table layout.Please do not try to have join like operations between different tables .HBase will be not giving good performance for such type of queries.

5.How does data get managed in HBase?

NoSQL databases are designed for scalability where unstructured data is spread across multiple nodes. When data volumes increase you just need to add another node to accommodate the growth. The lack of structure in NoSQL databases relaxes stringent requirements of consistency enforced in relational databases to improve speed and agility. HBase, MongoDB and Cassandra are the three major options that provide NoSQL capabilities. The options differ in the features they provide, so the decision on which to use is informed by the workload that will be handled.

6.What happens internally when new data gets inserted into HBase table?

To write data to HBase, you use methods of the HTableInterface class. You can use the Java API directly, or use HBase Shell, Thrift API, REST API, or another client which uses the Java API indirectly. When you issue a Put, the coordinates of the data are the row, the column, and the timestamp. The timestamp is unique per version of the cell, and can be generated automatically or specified programmatically by your application, and must be a long integer