Problem Statement

1. What is NoSQL data base?

Ans: A NOSQL database is a non-relational database which deals with data that is not organized in form of relation or tables and doesn't implement normalization for data storage unlike relational databases.
NOSQL databases don't have definite schema, thus they are highly scalable horizontally as user can alter the schema of tables dynamically any time when it is required. Thus, NOSQL databases provide flexibility in terms of change management. NOSQL databases don't need high end expertise in terms of management as compared to RDBMS Databases.
They deal with the soaring data volumes in storage management and they are cheaper as compared to RDBMS databases.
Since it stores data in form of key value pairs, it provides random access while performing read and write operations. Thus, it shows faster performance.

2. How does data get stored in NoSQL database?

Ans: There are different types of NOSQL databases and they exhibit different storage mechanisms.

In KEY-VALUE Databases, data is structured in tables and stored in a cell in a combination of key and corresponding value. In a key-value store, there is no schema and the value of the data is opaque. Values are identified and accessed via a key, and stored values can be numbers, strings, counters, JSON, XML, HTML, binaries, images, short videos, and more. It is the most flexible NoSQL model because the application has complete control over what is stored in the value.

In Document Database/Document Store/Document-oriented database, some document stores may also be key-value databases. A document database is used for storing, retrieving, and managing semi-structured data. Unlike traditional relational databases, the data model in a document database is not structured in a table format of rows and columns. The schema can vary, providing far more flexibility for data modelling than relational databases. A document database uses documents as the structure for storage and queries. In this case, the term "document" may refer to a Microsoft Word or PDF document but is commonly a block of XML or JSON. Instead of columns with names and data types that are used in a relational database, a document contains a description of the data type and the value for that description. Each document can have the same or different structure. To add additional types of data to a document database, there is no need to modify the entire database schema as there is with a relational database. Data can simply be added by adding objects to the database. Documents are grouped into "collections," which serve a similar purpose to a relational table. A document database provides a query mechanism to search collections for documents with particular attributes.

In GRAPH Databases, data is stored in form of entities and relationships in between those entities. Entities are known as nodes, which have properties. Nodes are like object instances and Relations are known as edges that can have properties. Edges have directional significance; nodes are organized by relationships which allow you to find interesting patterns between the nodes. The organization of the graph lets the data to be stored once and then interpreted in different ways based on relationships. Relationships are the most prominent part of Graph databases and most of the value of graph databases is derived from the relationships. the flexibility of a graph model allows you to add new nodes and relationships without compromising your existing network or expensively migrating your data. All of your original data (and its original relationships) remain intact. With data relationships at their centre, graph databases are incredibly efficient when it comes to query speeds, even for deep and complex queries.

In Column Store Databases, data is stored in cells grouped in columns of data rather than as rows of data. Columns are logically grouped into column families. Column families can contain a virtually unlimited number of columns that can be created at runtime or the definition of the schema. Read and write is done using columns rather than rows. In comparison, most relational DBMS store data in rows, the benefit of storing data in columns, is fast search/ access and data aggregation. Relational databases store a single row as a continuous disk entry. Different rows are stored in different places on disk while Columnar databases store all the cells corresponding to a column as a continuous disk entry thus makes the search/access faster.

3. **What is a column family in HBase?**

Ans: HBASE table schema defines only column families, which are the key value pairs. A table may have multiple column families and each column family can have any number of columns. Subsequent column values are stored contiguously on the disk. Column is a collection of key value pairs. An HBASE family can have max 3 Column families.

4. **Why columns are not defined at the time of table creation in HBase?**

Ans: HBase is schema-less, it doesn't have the concept of fixed columns schema; defines only column families. It is highly scalable since columns are not defined at the time of table creation in HBASE. HBASE is horizontally scalable in terms of columns, as columns can be added later in column family.

5. **How does data get managed in HBase?**

Ans: Data in HBase is stored in Tables and these Tables are stored in Regions. When a Table becomes too big, the Table is partitioned into multiple Regions. These Regions are assigned to Region Servers across the cluster. Each Region Server hosts roughly the same number of Regions.

Each Region Server contains a Write-Ahead Log (called HLog) and multiple Regions. Each Region in turn is made up of a MemStore and multiple StoreFiles (HFile). The data lives in these StoreFiles in the form of Column Families. The MemStore holds in-memory modifications to the Store (data). The mapping of Regions to Region Server is kept in a system table called. METATABLE.
When trying to read or write data from HBase, the clients read the required Region information from the. META table and directly communicate with the appropriate Region Server. Each Region is identified by the start key (inclusive) and the end key (exclusive).

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

Ans: HBASE data is sorted into partitioned spaces called as SHARDS or REGIONS by row keys. Each Region is identified by the start key(inclusive) and the end key(exclusive). Region Servers manages multiple regions hosted on data nodes. Therefore, based on put operation's key, HBASE client locates the region server and eventually the region where insert operation will take place VIA METATABLE. This data cannot be written directly to HFILES directly as data in HFILES must be sorted in row key sequence. The intended data is written to cache memory called as MEMSTORE, which efficiently supports random write operations. Data in MEMSTORE is sorted in the same order as in HFILEs. When the MEMSTORE gathers ample data, the entire sorted data is written to HFILE entirely and after successful WRITE to HFILE, MEMSTORE flushes out. Although writing data to the MEMSTORE is efficient, it also introduces an element of risk: Information stored in MEMSTORE is stored in volatile memory, so if the system fails, all MEMSTORE information is lost. To help mitigate this risk, HBase saves updates in a WRITE-AHEAD-LOG (WAL) before writing the information to MEMSTORE. In this way, if a region server fails, information that was stored in that server's MEMSTORE can be recovered from its WAL.

7. **How many maximum number of columns can be added to HBASE Table?**

Ans: No such limitation. Any number of columns can be added to HBASE table. Although numerous columns can result into performance issues.