1. State the main differences between MongoDB and MySQL?

The high-level differences between SQL and NoSQL databases are as follows:

- **Schema:** SQL databases store information in the form of tables with a fixed schema, whereas NoSQL databases can follow a dynamic schema.

- **Scaling:** SQL databases are generally scaled **vertically** by increasing the CPU and RAM to increase the power of the existing database. NoSQL databases are more suitable for **horizontal scaling**, where you keep adding more servers to improve performance.

- **Structure:**SQL databases are designed to handle advanced querying requirements,whereas NoSQL databases are utilised to handle complex data and can be scaled as per requirement.

2. How is data stored in MongoDB?

The databases in the MongoDB store all the information in the form of **collections**. These collections act as tables in SQL and store all the information in them in the form of **documents**. The documents are stored in MongoDB in JavaScript Object Notation (JSON) format. The information is recorded in the form of a key-value pair.

3. Define aggregation and aggregation pipeline in MongoDB?

Aggregation, in simple terms, means summarising the information present in multiple documents with a single computed value like **'sum'**, **'average'**, **'max**' and **'min'**. These values are pretty important from an analytics perspective since it helps in making comparisons between different entities.
One of the most fundamental aggregation functions is **$group,** which will help us compute values like total, mean and maximum.

The aggregation pipeline is a framework. The aggregation framework in MongoDB exists as **a pipeline of multiple stages.** Each stage takes in a set of documents, performs certain operations and outputs a list of documents which then pass further to the next stage. This way, only a subset of the entire collection is in the pipeline at any time down the line, thereby reducing the load on the server.

4. Explain what indexes are in MongoDB and mention the basic syntax to use indexes in MongoDB?

Indexes are defined as 'special data structures that store a small portion of the collection's data set in an easy-to-traverse form'. Indexes improve the efficiency of find() operations by reducing the number of documents that you need to scan.

Indexes store the value of a specific field or set of fields in a specific order. Indexes are defined at the collection level and using them we can return sorted results.

For generating a new index on any given field, you need to use the following syntax:

db.collection.createIndex({<field_name>:<order>})

5. What is sharding in MongoDB?

Sharding is a method used by MongoDB to support huge data sets that have high read-write operations.

Sharding belongs to the horizontal scaling methodology. Here, the entire collection of a database is divided into several chunks, and then each of these chunks is stored in separate replica sets or **shards**. This way, the querying performance improves as it runs only on a specific shard and not the entire collection.

6. What is replication in MongoDB?

So, replication in MongoDB essentially refers to the creation of new secondary nodes of data from a single primary node. All the nodes or servers would contain the same number of documents and would receive different kinds of read/write requests based on their availability. Replication gives a significant boost to your data delivery needs. Two of its major advantages are:

● **Data availability:** If a server goes down, another server can take the workload. It improves fault tolerance and if a server goes down, you have backups of the same data on other servers as well.

● **Improved efficiency:** In case of requests for collections containing a huge number of records (say in millions), the client can send requests to multiple replica sets instead of a single server to improve efficiency.

7. Write syntax for the following operations in MongoDB:

- Create collection
- Drop collection
- Insert document
- Print x number of documents in a collection
- Sort all the documents in a given collection on the basis of a particular field

The syntax for above operations are as follows:

- **To create a collection:** db.createCollection(name,options)

Here, name is the name of the collection to be created and options is an optional field specifying the options about size and indexing.
Example: To create a collection named purchases in a database named upgrad, the query would be as follows:

```
use upgrad

db.createCollection('purchases'')
```

- **To drop a collection:** db.Collection_Name.drop()
- **To insert a document:** db.COLLECTION_NAME.insert(document) Example:

db.Orders.insert({'Order_ID':'1','name':'book','sales':400,'profit':60})

- **To print x number of documents in a collection:**

  db.COLLECTION_NAME.find().limit(x)

- **To sort all the documents in a given collection on the basis of a particular field:**

  db.COLLECTION_NAME.find().sort({KEY:1})

Example: db.purchases.find({'Sales':{$lt:**15**}}).sort({'Sales':1})

Here, we find all the documents in **purchases** collection whose sales value is less than 15 and then sort the results in ascending order.

To sort in descending order, we could have used -1 in place of 1.

8. Why is MongoDB not preferred over a 32-bit system?

The server's total storage size for a 32-bit build of MongoDB would be limited to 2 GB only. This may not be a preferable choice for a production deployment of MongoDB on a 32-bit machine.

Generally, 64-bit systems are preferred for working with MongoDB. When you run a 64-bit build of MongoDB, it provides virtually unlimited storage size.

9. Explain briefly about _id field in MongoDB?

All the documents in a collection must have a unique _id field acting as a primary key. The _id field is always the first field in the documents. In case a document is inserted without an _id field, the MongoDB driver automatically adds the _id field and generates an ObjectId for it.

10. Is it possible to achieve primary key-foreign key relationships in MongoDB?

By default, MongoDB does not support such primary key-foreign key relationships. However, we can achieve this concept by embedding one document inside another. For example, a contact document can be embedded inside the learner document.