Q1. What is MongoDB? Explain non-relational databases in short. In which scenarios it is preferred to use MongoDB over SQL databases?

Answer :-

MongoDB is a cross-platform document-oriented database application that is open source. MongoDB, a NoSQL database application, employs documents that resemble JSON and may or may not include schemas. MongoDB was created by MongoDB Inc. and is distributed under the Server Side Public License, which some distributions consider to be non-free.

Non-relational document database MongoDB supports storage that is similar to JSON. The MongoDB database has full indexing support, replication, and a flexible data schema that makes it possible to store unstructured data. It also features comprehensive and user-friendly Interfaces.

MongoDB is a general-purpose database that supports applications across numerous sectors in a variety of ways (e.g., telecommunications, gaming, finances, healthcare, and retail). Since it addresses persistent issues in data management and software development, MongoDB has found a home in a variety of industries and job roles. MongoDB's typical use cases include:

Q2. State and Explain the features of MongoDB.

Schema-less Database: MongoDB offers this wonderful feature. It implies that many document types can be included in a single collection. In the MongoDB database, numerous documents may be stored in a single collection, each of which may have a distinct amount of fields, kind of content, and size. Unlike to relational databases, there is no requirement that one document be comparable to another. MongoDB offers databases a lot of flexibility as a result.

Dedicated to Documents: Instead of tables like in RDBMS, all the data is kept in it as documents. In contrast to RDBMS, it stores data in fields rather than rows and columns, which gives the data considerably more flexibility. Moreover, every document has a distinct object id.

Indexing : Every field in the documents in the MongoDB database is indexed with primary and secondary indices, making it quicker and easier to acquire or search for data from the data pool. If the data is not indexed, the database must search each document individually using the given query, which is time-consuming and ineffective.

Scalability: With the use of sharding, MongoDB offers horizontal scalability. With the shard key, a significant amount of data is divided into chunks and then distributed evenly among shards that are spread across numerous physical servers. Sharding is the process of distributing data across multiple servers. Moreover, it will add new computers to an active database.

Replication: MongoDB offers high availability and redundancy by making multiple copies of the data and sending each of these copies to a different server. This way, in the event that one server fails, the data can still be retrieved from the other server.

Aggregation: This function enables actions to be carried out on the grouped data to provide a single result or computed result. It is comparable to the GROUPBY clause in SQL. It offers three different aggregations, including a map-reduce function, an aggregation pipeline, and a single-purpose aggregation approach.

High Performance: exemplary performance Due to its characteristics like scalability, indexing, replication, and others, MongoDB has a very high performance and data persistence compared to other databases.

Q3. Write a code to connect MongoDB to Python. Also, create a database and a collection in MongoDB.

- pip install pymongo
- from pymongo import MongoClient
- client = MongoClient("mongodb://localhost:27017/")

```
#import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]
```

Q4. Using the database and the collection created in question number 3, write a code to insert one record,and insert many records. Use the find() and find_one() methods to print the inserted record.

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

mydict = { "name": "John", "address": "Highway 37" }

x = mycol.insert_one(mydict)
```

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

for x in mycol.find():

  print(x)
```

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

x = mycol.find_one()

print(x)
```

Q5. Explain how you can use the find() method to query the MongoDB database. Write a simple code to demonstrate this.

The find() method in MongoDB can also be used to select data from a table.

All instances in the selection are returned by the find() method.

A query object is the find() method's first argument. In this illustration, we utilise an empty query object, which chooses all of the collection's documents.

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

for x in mycol.find({},{ "address": 0 }):

  print(x)
```

Q6. Explain the sort() method. Give an example to demonstrate sorting in MongoDB.

To arrange the outcomes in ascending or descending order, use the sort() method.

One parameter for "fieldname" and one for "direction" are required by the sort() method (ascending is the default direction).

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

mydoc = mycol.find().sort("name")

for x in mydoc:

  print(x)
```

Q7. Explain why delete_one(), delete_many(), and drop() is used.

Delete the document

We employ the delete one() technique to remove a single document.

A query object specifying the document to be deleted is sent as the first parameter to the delete one() method.

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

myquery = { "address": "Mountain 21" }

mycol.delete_one(myquery)
```

Delete Many Documents

Use the delete_many() function to remove several documents.

A query object specifying the documents to be deleted is the first parameter of the delete_many() function.

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

myquery = { "address": {"$regex": "^S"} }

x = mycol.delete_many(myquery)

print(x.deleted_count, " documents deleted.")
```

Delete All Documents in a Collection

The delete_many() method accepts an empty query object to delete every document in the collection:

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]
```

```
x = mycol.delete_many({})

print(x.deleted_count, " documents deleted.")
```