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

MongoDB : MongoDB is a non-relational document database that provides support for JSON-like storage. The MongoDB database has a flexible data model that enables you to store unstructured data, and it provides full indexing support, and replication with rich and intuitive APIs.

Non-relational Database : Non-relational databases (often called NoSQL databases) are different from traditional relational databases in that they store their data in a non-tabular form. Instead, non-relational databases might be based on data structures like documents. A document can be highly detailed while containing a range of different types of information in different formats. This ability to digest and organize various types of information side by side makes non-relational databases much more flexible than relational databases.

In which scenarios it is preferred to use MongoDB over SQL databases? : 1) High Availability 2) Unstable Schema 3) In-built Sharding 4) No DBA 5) Cloud Computing

Q2. State and Explain the features of MongoDB.

Schema less : Schema-less Database: It is the great feature provided by the MongoDB. A Schema-less database means one collection can hold different types of documents in it. Or in other words, in the MongoDB database, a single collection can hold multiple documents and these documents may consist of the different numbers of fields, content, and size. It is not necessary that the one document is similar to another document like in the relational databases. Due to this cool feature, MongoDB provides great flexibility to databases.

Document Oriented: In MongoDB, all the data stored in the documents instead of tables like in RDBMS. In these documents, the data is stored in fields(key-value pair) instead of rows and columns which make the data much more flexible in comparison to RDBMS. And each document contains its unique object id.

Indexing: In MongoDB database, every field in the documents is indexed with primary and secondary indices this makes easier and takes less time to get or search data from the pool of the data. If the data is not indexed, then database search each document with the specified query which takes lots of time and not so efficient.

Scalability: MongoDB provides horizontal scalability with the help of sharding. Sharding means to distribute data on multiple servers, here a large amount of data is partitioned into data chunks using the shard key, and these data chunks are evenly distributed across shards that reside across many physical servers. It will also add new machines to a running database.

Replication: MongoDB provides high availability and redundancy with the help of replication, it creates multiple copies of the data and sends these copies to a different server so that if one server fails, then the data is retrieved from another server.

Aggregation: It allows to perform operations on the grouped data and get a single result or computed result. It is similar to the SQL GROUPBY clause. It provides three different aggregations i.e, aggregation pipeline, map-reduce function, and single-purpose aggregation methods

High Performance: The performance of MongoDB is very high and data persistence as compared to another database due to its features like scalability, indexing, replication, etc.

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

```
import pymongo
client =
pymongo.MongoClient("mongodb+srv://addywang:addywang@cluster0.sa8shcv.mongodb.net/?
retryWrites=true&w=majority")
db = client.tes
mydb = client['sports']
coll = mydb['films']
```

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.

```
data =   {
    "Title": "Avatar",
    "Year": "2009",
    "Rated": "PG-13",
    "Released": "18 Dec 2009",
    "Runtime": "162 min",
    "Genre": "Action, Adventure, Fantasy",
  }
coll.insert_one(data)
```

```
data = [ {
    "Title": "I Am Legend",
    "Year": "2007",
    "Rated": "PG-13",
    "Released": "14 Dec 2007",
    "Runtime": "101 min",

  },
  {
    "Title": "300",
    "Year": "2006",
    "Rated": "R",
    "Released": "09 Mar 2007",
    "Runtime": "117 min",
  }]
coll.insert_many(data)
```

```
for i in coll.find():
    print(i)
```

```
coll.find_one()
```

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

syntax of find is following: collection_name.find()

when you write find statement then it will give iterator of all document, so using for loop you have to iterate

```
record = coll.find()
for i in record:
    print(i)
```

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

sort() : The sort() method specifies the order in which the query returns the matching documents from the given collection. You must apply this method to the cursor before retrieving any documents from the database. It takes a document as a parameter that contains a field: value pair that defines the sort order of the result set. The value is 1 or -1 specify an ascending or descending sort respectively.

```
for i in coll.find().sort('Year',-1):
    print(i)
```

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

delete_one() : To delete one document, we use the delete_one() method.

If the query finds more than one document, only the first occurrence is deleted. delete_many() : To delete more than one document, use the delete_many() method.

Drop() : You can delete a table, or collection as it is called in MongoDB, by using the drop() method.

In [ ]:

```
```