Q1. MongoDB is a popular open-source, document-oriented NoSQL database that provides high performance, high availability, and easy scalability. In non-relational databases like MongoDB, data is stored in flexible, JSON-like documents instead of rigid, pre-defined schemas found in SQL databases. Non-relational databases are preferred in scenarios where:

There is a need for flexible schema design, allowing for dynamic and evolving data structures.
Large volumes of rapidly changing structured, semi-structured, and unstructured data need to be managed.
Horizontal scalability is required, allowing for easy distribution of data across multiple servers or clusters.
Rapid development and iteration are necessary, as non-relational databases offer greater agility in development and deployment.

Q2. Features of MongoDB include:

Document-Oriented: MongoDB stores data in flexible, JSON-like documents.
Scalability: It supports horizontal scaling through sharding.
High Performance: MongoDB offers high-performance read and write operations due to its memory-mapped files.
High Availability: It provides replication and automatic failover.
Rich Query Language: MongoDB supports rich queries, including field, range queries, and regular expressions.
Indexing: It supports secondary indexes for efficient query execution.
Aggregation Framework: MongoDB provides an aggregation framework for data processing and analytics.
Flexible Schema: MongoDB does not enforce a schema, allowing for dynamic and evolving data structures.
GridFS: It allows for storage and retrieval of large files like images, videos, and audio files.

Q3. Here's a Python code to connect to MongoDB, create a database, and a collection:

```
from pymongo import MongoClient

# Connect to MongoDB server
client = MongoClient('mongodb://localhost:27017/')

# Create a database
mydb = client['mydatabase']

# Create a collection
mycol = mydb['mycollection']
```

Q4. Here's a code to insert one record and insert many records into the collection, and print them using find() and find_one():

```
# Insert one record
record = {"name": "John", "age": 30}
mycol.insert_one(record)

# Insert many records
records = [
   {"name": "Alice", "age": 25},
   {"name": "Bob", "age": 35}
]
mycol.insert_many(records)

# Print inserted records
print("All records:")
for doc in mycol.find():
   print(doc)

print("One record:")
print(mycol.find_one())
```

Q5. The find() method in MongoDB is used to query documents from a collection based on specified criteria. You can use various query operators and modifiers to filter the documents. Here's a simple code to demonstrate the find() method:

```
# Find documents where age is greater than 25
for doc in mycol.find({"age": {"$gt": 25}}):
    print(doc)
```
This code will print all documents from the collection where the age field is greater than 25.

Q6. The sort() method in MongoDB is used to sort the documents in a collection based on one or more keys. It takes one parameter, a dictionary specifying the field(s) to sort by and the order of sorting (ascending or descending). Here's an example:

```
# Sort documents by age in descending order
for doc in mycol.find().sort("age", -1):
    print(doc)
```
This code will print all documents from the collection sorted by the age field in descending order.

Q7.

delete_one(): This method is used to delete a single document from a collection that matches a specified filter.
delete_many(): This method is used to delete multiple documents from a collection that match a specified filter.
drop(): This method is used to drop an entire collection from the database.
These methods are used for data manipulation and management in MongoDB. delete_one() and delete_many() are used to remove documents from a collection based on specific criteria, while drop() is used to remove an entire collection from the database.