Assignment

Q1

MongoDB is a document-oriented NoSQL database that stores data as JSON-like BSON documents, allowing flexible schemas and horizontal scaling. Non-relational databases store data without fixed tables/rows; they use documents, key-value, wide-column, or graph models, favoring flexibility, schema-less design, and scalability over strict relations. Prefer MongoDB when data structure evolves often, needs flexible schema, high write throughput, horizontal scaling, or when working with nested/JSON data and agile iterations, rather than rigid joins and strong ACID transactions typical of many SQL systems.

Q2

- Document model (BSON) with schema flexibility for evolving fields and nested structures.
- Collections and databases created lazily on first insert.
- Powerful querying with filters, projections, sorting, limits, and indexes.
- Horizontal scalability via sharding and high availability via replication (core NoSQL traits; MongoDB supports them, implied in official docs context).
- Drivers for many languages; PyMongo for Python.
- Aggregation framework and cursor methods like sort() for ordered results.

Q3

python

```
import pymongo

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

db = client["mydatabase"]  # created on first insert

col = db["students"]       # created on first insert
```

Q4

python

```
import pymongo

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

db = client["mydatabase"]

col = db["students"]
```

```
one_doc = {"name":"Aisha","age":21,"dept":"CS"}

col.insert_one(one_doc)

many_docs = [
    {"name":"Ravi","age":22,"dept":"IT"},
    {"name":"Meera","age":20,"dept":"CS"}
]

col.insert_many(many_docs)

print(col.find_one({"name":"Aisha"}))

for d in col.find({"dept":"CS"}):
    print(d)
```

Q5

find() returns a cursor for matching documents; you pass a filter document to specify conditions, and you can chain projection, sort, limit, and skip.prisma+2

python

```
import pymongo


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

db = client["mydatabase"]

col = db["students"]


for d in col.find({"age":{"$gte":21}}, {"_id":0,"name":1,"age":1}):
    print(d)
```

Q6 sort() orders query results by fields; 1 for ascending, -1 for descending, and multiple fields are evaluated left to right.bmc+2

python

```
import pymongo

from pymongo import ASCENDING, DESCENDING
```

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

db = client["mydatabase"]

col = db["students"]

for d in col.find({}, {"_id":0,"name":1,"age":1}).sort([("age", ASCENDING), ("name", DESCENDING)]):
  print(d)
```

Q7

- delete_one(filter): deletes the first document matching the filter, use unique fields like _id for precise removal.
- delete_many(filter): deletes all documents matching the filter for bulk removals.
- drop(): drops (removes) an entire collection, including all documents and indexes, used to permanently remove the collection when no longer needed.