- **Design the MongoDB database schema to store user profiles, posts, and relationships between users.**

-Here I have created 2 schemas,

1st – User

2nd – Posts

1st User Schema -

```
db.users.insertOne({
name: "Sarthak Surve",
username: "Sarthak_ss01",
email: "SarthakSurve@gmail.com",
dateOfBirth: ISODate("2001-08-08"),
followers: ["64b220c22d87609bc0e7e2f8"],
following: ["64b220c22d87609bc0e7e2f8"],
interests: ["cricket", "music"] })
```

```
{
  acknowledged: true,
  insertedId: ObjectId("64b222a72d87609bc0e7e2f9")
}
```

2nd Post Schema

```
db.posts.insertOne({
 content: "Mission ISRO chandrayan 3",
 timestamp: new Date(),
 user: ObjectId("64b222a72d87609bc0e7e2f9"),
 likes: [ObjectId("64b222a72d87609bc0e7e2f9"),
ObjectId("64b223012d87609bc0e7e2fa")],
```

```
  comments: [
    {
      user: ObjectId("64b222a72d87609bc0e7e2f9"),
      content: "Congrats",
      timestamp: new Date()
    },
    {
      user: ObjectId("64b223012d87609bc0e7e2fa"),
      content: "Jai Hind",
      timestamp: new Date()
    }
  ]
})
```

```
{
  acknowledged: true,
  insertedId: ObjectId("64b223d22d87609bc0e7e2fb")
}
```

- **Insert a new user profile into the database.**

```
db.users.insertOne({
  name: "Sarthak Surve",
  username: "Sarthak_ss01",
  email: "SarthakSurve@gmail.com",
  dateOfBirth: ISODate("2001-08-08"),
  followers: [],
  following: [],
  interests: ["cricket", "music"]})
```

- **Retrieve a user's profile information, including their followers and followed users.**

```
db.users.aggregate([
 { $match: { _id: ObjectId("64b222a72d87609bc0e7e2f9") } },
 {
  $lookup: {
   from: "users",
   let: { followersIds: { $map: { input: "$followers", in: { $toObjectId: "$$this" } } } },
   pipeline: [
    { $match: { $expr: { $in: ["$_id", "$$followersIds"] } } },
    { $project: { name: 1, username: 1} }
   ],
   as: "followers"
  }
 },
 {
  $lookup: {
   from: "users",
   let: { followingIds: { $map: { input: "$following", in: { $toObjectId: "$$this" } } } },
   pipeline: [
    { $match: { $expr: { $in: ["$_id", "$$followingIds"] } } },
    { $project: { name: 1, username: 1} }
   ],
   as: "following"
  }
 }
])
```

- **Insert a new post associated with a user.**

```
db.posts.insertOne({
  content: "Hello this is just a sample post, hope u guys are great",
  timestamp: new Date(),
  user: ObjectId("64b222a72d87609bc0e7e2f9"),
  likes: [],
  comments: []
})
```

- **Retrieve a user's posts in reverse chronological order.**

  db.posts.find({ user: ObjectId("64b2290d2d87609bc0e7e2fe") }).sort({ timestamp: -1 })

- **Discuss the advantages of using MongoDB's document model for storing social media data.**

- Using a MongoDb Document model for Sociel media is that is can be highly scalable ,because mongodb provides flexible schema designs, new fields can be added to schema without making the existing schema change
- We can store complex data just by adding arrays in single page in mongoDB
- MongoDB is fast, so read and write operations can be implemented faster.
- It is easy to use, in other words we can say it is developer friendly.

- **Explain how MongoDB's replication and sharding features can help ensure high availability and scalability in a social media application.**

- Replication is duplicating data into various servers, it copies one data across various servers. This improvers data retrieval speed and data loss can also be prevented
- Sharding is partitioning of data across multiple machines.
- In context with social media apps, replication can help ensure high availability of data by mainlining multiple copies of data on different database servers.
- This means that if one server goes down, there are still others. Sharding can ensure scalability by distributing the data set across Mutiple shards.

Hotel_info
Refer to the schema. Write a query to display the hotel name along with the type. Display the details in the below format.
Give an alias name as `hotel_info`. Sort the result in descending order.

```
SELECT CONCAT('hotel_info ', Hotel_name, ' is a ', Hotel_type, ' hotel') AS output
FROM hotel_info ORDER BY hotel_name DESC;
```

Car & owner details based on car type
Write a query to display car id, car name and owner id of all the cars whose car type is 'Hatchback' or 'SUV'. Sort the result based on car id.

SELECT car_id, car_name, owner_id FROM cars WHERE car_type IN [“Hatchback”, “SUV”] ORDER BY car_id;