CAMPUS: VIT AP

NAME: JYOTHI REDDY.PULA

REGISTRATION NUMBER: 23BCE7882

MERN ASSIGNMENT(21ST JUNE 2025)

1. Q: Insert a single document into a collection named `students` with fields: name, age, and course.

Theory: `insertOne()` adds a single document to a collection.

Query:

```
js
CopyEdit
db.students.insertOne({ name: "John", age: 21, course: "CS" });
```

2. Q: Insert multiple documents into the **employees** collection with fields: name, salary, and department.

Theory: `insertMany()` adds multiple documents at once.

Query:

```
js
CopyEdit
db.employees.insertMany([
  { name: "Alice", salary: 35000, department: "HR" },
  { name: "Bob", salary: 40000, department: "IT" }
]);
```

3. Q: Find all documents from the **products** collection.

Theory: Use `{}` to match all documents.

Query:

```
js
CopyEdit
db.products.find({});
```

4. Q: Find all documents in the **users** collection where age is greater than 25.

Theory: Use `$gt` for "greater than".

Query:

```
js
CopyEdit
db.users.find({ age: { $gt: 25 } });
```

5. **Q: Find documents from the** `orders` **collection where status is either "pending" or "shipped".**

Theory: `$in` matches any value in a list.

Query:

```
js
CopyEdit
db.orders.find({ status: { $in: ["pending", "shipped"] } });
```

6. Q: Update the email field of a user where username is "john_doe" in the **users** collection.

Theory: Use `$set` to modify a field.

Query:

```
js
CopyEdit
db.users.updateOne(
  { username: "john_doe" },
  { $set: { email: "john@example.com" } }
);
```

7. Q: Delete a document from the **students** collection where roll is 101.

Theory: `deleteOne()` removes a single matching document.

Query:

```
js
CopyEdit
db.students.deleteOne({ roll: 101 });
```

8. Q: Find all employees with salary greater than or equal to 30000.

Theory: `$gte` means "greater than or equal to".

Query:

```
js
CopyEdit
db.employees.find({ salary: { $gte: 30000 } });
```

9. Q: Retrieve all books where author is "Chetan Bhagat" and `publishedYear` is after 2010.

Theory: Combine conditions using `{}`.

Query:

```
js
CopyEdit
db.books.find({ author: "Chetan Bhagat", publishedYear: { $gt: 2010 }
});
```

10. Q: Count the number of documents in the `customers` collection where city is "Delhi".

Theory: `countDocuments()` counts matches.

Query:

```
js
CopyEdit
db.customers.countDocuments({ city: "Delhi" });
```

11. Q: Find the first 5 users from the `users` collection using `limit()`.

Theory: `limit()` restricts result count.

Query:

```
js
CopyEdit
db.users.find().limit(5);
```

12. Q: Skip the first 10 documents and retrieve the next 5 from the `logs` collection.

Theory: Use `skip()` and `limit()` together for pagination.

Query:

```
js
CopyEdit
db.logs.find().skip(10).limit(5);
```

13. Q: Sort all products in ascending order of price.

Theory: Use `.sort({ field: 1 })` for ascending.

Query:

```
js
CopyEdit
db.products.find().sort({ price: 1 });
```

14. Q: Sort users in descending order of `createdAt` date.

Theory: Use `.sort({ field: -1 })` for descending.

Query:

```
js
CopyEdit
db.users.find().sort({ createdAt: -1 });
```

15. Q: Retrieve only the name and email fields of users (hide `_id`).

Theory: Projection defines which fields to show.

Query:

```
js
CopyEdit
db.users.find({}, { _id: 0, name: 1, email: 1 });
```

16. Q: Find all students whose marks are between 60 and 90.

Theory: Use `$gte` and `$lte` for range.

Query:

```
js
CopyEdit
db.students.find({ marks: { $gte: 60, $lte: 90 } });
```

17. Q: Retrieve documents from `sales` where amount < 500 or > 5000.

Theory: `$or` allows multiple conditions.

Query:

```
js
CopyEdit
db.sales.find({
  $or: [{ amount: { $lt: 500 } }, { amount: { $gt: 5000 } }]
});
```

18. Q: Update the status to "completed" for all orders where `deliveryDate` is not null.

Theory: `$ne` means "not equal".

Query:

```
js
CopyEdit
db.orders.updateMany(
  { deliveryDate: { $ne: null } },
  { $set: { status: "completed" } }
);
```

19. Q: Delete all inactive users from the `users` collection where active is false.

Theory: `deleteMany()` removes multiple matches.

Query:

```
js
CopyEdit
db.users.deleteMany({ active: false });
```

20. Q: Find users who are either from "Bangalore" or have age greater than 30.

Theory: Use logical OR `$or` operator.

Query:

```
js
CopyEdit
db.users.find({
  $or: [{ city: "Bangalore" }, { age: { $gt: 30 } }]
});
```