LAB-02

11/03/2025

PAGE NO :
DATE :

(1) Create collection named customers.

```
use myDatabase;
db.createCollection("customers");
```

output:

switched to db myDatabase;

(2) inserting at least 5 values

```
db.Customers.insertMany([
  {cust_id:1, Acc_Bal:1500, AccType:"z"},
  {cust_id:2, Acc_Bal:1100, AccType:"x"},
  {cust_id:3, Acc-Bal:1300, AccType:"z"},
  {cust_id:4, Acc_Bal:400, AccType:"y"},
  {cust_id:5, Acc_Bal:2000, AccType:"z"}
]);
```

(3) Query to display records where account balance >1200 and Acc_Type is 'z'

```
db.Customers.find({Acc-Bal:{$gt:1200}, Acc_Type:"z"});
```

output:

```
{"cust_id:1, "Acc_Bal:1500, AccType:"z"},
{"cust_id:3, Acc_Bal:1300, Acc_TYPE:"z"},
{"cust_id:5, "ACC_Bal:2000, Acc_TYPE:"z"}.
```

(4) Determine minimum & maximum account balance for each cust_id.

PAGE NO :
DATE :

```
db.Customers.aggregate([
  {
    $group : {
      _id = "$cust_id",
      Min_Balance : { $min : "$Acc_Bal" },
      Max_Balance : { $max : "$Acc_Bal" }
    }
  }
]);
```

output :-

```
{ _id : 1, Min_Balance : 1500, Max_Balance : 1500 }
{ _id :
```

(5) Create E-Commerce Schema.

```
db.CreateCollection("Products");
db.CreateCollection("users");
db.createCollection("Orders");
```

(6) Insert Sample Products.

```
db.Products.InsertMany([
  { _id : 1, name : "Laptop", category : "Electronics",
    Price : 800, quantity : 10 },
  { _id : 2, name : "Phone", category : "Electronics",
    price : 500, quantity : 5 },
  { _id : 3, name : "shoes", category : "Fashion", Price
    : 50, quantity : 20 }
]);
```

(7) Retrieve All products.

PAGE NO :
DATE :

db.Products.find();

(8) Retrieve Products in specific category.

db.products.find({category: "Electronics"});

Output:

{"_id":1, name: "Laptop", category: "Electronics", price: 800, quantity: 10}
{"_id":2, name: "Phone", category: "Electronics", price: 500, quantity: 5}

(9) Retrieve products sorted by price in Ascending Order.

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

Output:

{"_id":3, name: "Shoes", price: 80, ...}
{"_id":2, name: "Phone", price: 500, ...}
{"_id":1, name: "Laptop", price: 800, ...}

(10) Price less than or Equal to $100.

db.Products.find({Price: {$lte:100}});

Output:

{_id:3, name: "Shoes", Price: 50, ...}

(11) Retrieve products Added to a user's cart

db.users.findOne({_id: "789ghi"}).cart.

Output: [1, 3].

PAGE NO :
DATE :

12) Retrieve orders placed by user.

```
db.Orders.find({user_id: "123abc"});
```

Output:

```
{ _id: "Ord1", user_id: "123abc", Products: [1,2]
    total: 1300 }.
```

13) Retrive total price of orders placed by user.

```
db.Orders.aggregate([
  { $match: { user_id: "123abc" } },
  { $group: { _id: "$user_id", totalPrice: {
      $sum: "$total" } } }.
]);
```

Output :-

```
{ _id: "123abc", totalPrice: 1300 }.
```

14) Calculate total number of products in each category

```
db.Products.aggregate([
  { $group: { _id: "$category", totalProducts: { $sum: 1 } } }
]);
```

15) calculate total price of Products in each category

```
db.Products.aggregate([
  { $group: { _id: "$category", total price: { $sum:
      "$price" } } }
]);
```

PAGE NO :
DATE :

16) Find Average price of products.

```
db.Products.aggregate([
  {$group:{_id:null, avgPrice:{$avg:"$price"}}}
]);
```

17) Find Products with quantity < 10

```
db.products.find({quantity:{$lt:10}});
```

18) Sort products by price in descending ordn

```
db.Products.find().sort({price:-1});
```

19) calculate total price of orders by users.

```
db.orders.aggregate([
  {$group:{_id:"$user_id", totalPrice:{$sum:"$total"}}}
]);
```

S.P. 10/3/25

20) Find users with highest total price of orders

```
db.orders.aggregate([
  {$group:{_id:"$user_id", totalPrice:{$sum:"$total"}}},
  {$sort:{totalPrice:-1}}, {$limit:1}
]);
```

21) Find Average total price of orders

```
db.orders.aggregate([
  {$group:{_id:null, avgTotalPrice:{$avg:"$total"}}}
]);
```