DATE: PAGE:

Lab-02

Mongodb Lab Exercise.

Perform the following DB operations using MongoDB.

a) Create a collection by name customers with the following attributes.

Cust_id, Acc_Bal, Acc_Type.

```
db.createcollection("customers");
```

b) Insert at least 5 values into the table.

```
db.Customers.insertMany([
{cust_id:1, Acc_Bal:1500, Acc_Type:"Z"},
{cust_id:2, Acc_Bal:1100, Acc_Type:"A"},
{cust_id:3, Acc_Bal:2000, Acc_Type:"Z"},
{cust_id:4, Acc_Bal:900, Acc_Type:"B"},
{cust_id:5, Acc_Bal:1300, Acc_Type:"Z"}])
```

c) Write a query to display those records whose total account balance is greater than 1200 of account type 'Z' for each customer_id.

```
db.Customers.find({Acc_Bal:{$gt:1200}, Acc_Type:"Z"})
```

d) Determine Minimum and maximum account balance for each customer i

DATE: PAGE:

```
db.customers.aggregate([{
  $group:{
    _id: "$cust_id",
    Min_Balance:{ $min: "$ACC_Bal"},
    Max_Balance: {$max: "$ACC_Bal"}}}]).
```

2. You are developing an e-commerce platform where users can browse and purchase products. Each product has a unique identifier, a name, a category, a price, and available quantity. Additionally, users can add products to their cart and place orders. Design a query to

```
db.createCollection("Products")
db.Products.insertMany([
{_id:1, name: "Laptop", category: "Electronics", price:1000,
        quantity:5},
{_id: 2, name: "Phone", category: "Electronics", price:500,
      quantity: 10},
{_id: 3, name: "T-shirt", category: "clothing", price: 30,
      quantity:20},
{_id:4, name: "Headphones", category: "Electronics", price:150,
quantity: 15},
{_id:5, name: "shoes", category: "Footwear", price: 80,
        quantity:12}]).

db.createCollection("users").
db.users.insertmany([
{_id:"789ghi", name: "John Doe", cart:[{product_id:1,
quantity:1}, {product_id:3, quantity:2}]},
{_id:"456xyz", name: "Alice Smith", cart:[{product_id:2,
quantity: 1}, {product_id:4, quantity:1}]}])
```

DATE: PAGE:

```
db.createCollection("orders")
db.Orders.insertMany([
{_id:"123abc", user_id:"789ghi", products:[{product_id:
1, quantity:1}, {product_id:3, quantity:2}], total_price:1060},
{_id:"456def", user_id:"456xyz", products:[{product_id:
2, quantity:1}, {product_id:4, quantity:1}], total_price:650}])
```

a) Retrieve: All products.:

```
db.Products.find({})
```

b) Retrieve products in a specific category (e.g., electronics)

```
db.Products.find({category: "Electronics"})
```

c) Retrieve products with quantity greater than 0.

```
db.Products.find({quantity:{$gt:0}})
```

d) Retrieve products sorted by price in Ascending order.

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

e) Retrieve products with price less than or equal to $100.

```
db.Products.find({price:{$lte:100}}).
```

f) Retrieve products added to a user's cart (id with "789ghi").

```
db.Users.find({_id:"789ghi"},{cart:1}).
```

g) Retrieve orders placed by a user (with "123abc")

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

DATE: PAGE:

h) Retrieve total price of orders placed by a user (with "123abc")

db.orders.aggregate([{ $match: {_id: "123abc"}}, {$group: {_id: "$_id", TotalPrice: { $sum: "$total_price"}}}]).

Additional Aggregation.

1. Calculate total Number of Products in each category.
   db.Products.aggregate([{ $group: {_id: "$category", TotalProducts: {$sum:1}}}]).

2. Calculate total price of products in each category.
   db.Products.aggregate([{$group: {_id: "$category", TotalPrice: { $sum: "$price"}}}]).

3. Find Average Price of products.
   db.Products.aggregate([{ $group: {_id: null, AvgPrice: { $avg: "$price"}}}]).

4. Find products with quantity less than 10.
   db.Products.find({quantity: { $lt:10}})

5. Sort products by price in descending order.
   db.Products.find().sort({price:-1}).

6. Calculate total price of orders placed by each user,
   db.orders.aggregate([{$group: {_id: "$user_id", Total_price: { $sum: "$total_price"}}}])

7. Find users with the highest total price of orders.

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

Find average total price of orders.

```
      db.orders.aggregate([
  {$group:{_id:null, AvgTotalPrice:{$avg:"total_price"}}}
      ])
```

[illegible]
11/3/25