11/3/25

Lab-2

Mongodb Lab Exercise

(1) i) Create a collection by name Customers

```
db.CreateCollection("Customers")
```

ii) Insert at least 5 values into the table.

```
db.Customers.insertMany([
  { Cust_id: 1, Acc_Bal: 1500, Acc_Type: 'x' },
  { Cust_id: 2, Acc_Bal: 2500, Acc_Type: 'z' },
  { Cust_id: 3, Acc_Bal: 900, Acc_Type: 'y' },
  { Cust_id: 4, Acc_Bal: 1700, Acc_Type: 'x' },
  { Cust_id: 5, Acc_Bal: 500, Acc_Type: 'z' }
]);
```

iii) 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' })
```

```
[
  {
    cust_id: 2,
    Acc_Bal: 2500,
    Acc_Type: 'z'
  }
]
```

PAGE NO :
DATE :

(iv) Determine Minimum and Maximum account balance for each customer_id

```
db.Customers.aggregate([
  {
    $group: {
      _id: "$cust_id",
      min_balance: { $min: "$Acc_Bal" },
      max_balance: { $max: "$Acc_Bal" }
    }
  }
])
```

```
[ { _id: 3, min_bal: 900, max_bal: 900 },
  { _id: 1, min_bal: 1500, max_bal: 1500 },
  { _id: 4, min_bal: 1700, max_bal: 1700 },
  { _id: 5, min_bal: 500, max_bal: 500 },
  { _id: 2, min_bal: 2500, max_bal: 2500 }
]
```

3 Schemas: Products, Users, Orders.

(2)

(i)
```
db.products.insertMany([
  {
    product_id: "001",
    name: "Smartphone",
    category: "Electronics",
    price: "29,99",
    available_quantity: 50
  },
  :
```

(ii)
```
db.users.insertMany([
  {
    user_id: "123abc",
```

```
name : "Alice",
cart : [
    { product_id : "001", quantity : 1 },
    { product_id : "004", quantity : 2 }
],
orders : [ ]
},
]
```

(iii)

```
db.orders.insertMany([
{
  order_id : "101",
  user_id : "123abc",
  products : [
    { product_id : "001", quantity : 1,
      price : 29999 },
    { product_id : "004", quantity : 2,
      price : 1999 } ],
  total_price : 33997,
  order_date : new Date ("2025-03-11")
},
```

a) Retrieve All Products.

```
db.products.find();
```

```
[
  {
    product_id : "001",
    name : 'Smartphone',
    category : 'Electronics',
    price : 29.99,
    available_quantity : 50
  },
]
```

PAGE NO :
DATE :

```
{
product_id: '005',
name: 'Jeans',
category: 'Apparel',
price: 49.99,
available_quantity: 150
}
]
```

b) Retrieve products with specific category

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

[ {
    product_id: '004',
    name: 'T-shirt'
    category: 'Apparel'
    price: 19.99
    available_quantity: 200
},
{ product_id: '005'
    name: 'Jeans'
    category: 'Apparel'
    price: 49.99
    available_quantity: 150
} ]
```

c) Retrieve products with Quantity greater than 0

```
db.products.find({ available_quantity:
{$gt: 0}});
```

PAGE NO :
DATE :

d) Retrieve Products sorted by Price in descending order.

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

004
005
003
001
002

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 (User with ID "789ghi")

```
db.users.find({user_id:"789ghi"},{cart:1});
```

cart: [
{ product_id: "003", quantity: 2 },
{ product_id: "005", quantity: 1 },
]

g) Retrieve orders Placed by a User (User ID with "123abc")

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

PAGE NO :
DATE :

```
[{ order_id: '101',
  user_id: '123abc',
  products: [
  { product_id: '001', quantity: 1, price: 299.99},
  { product_id: '004', quantity: 2, price: 19.99},
  ],
  total_price: 339.97,
  order_date: '2025-03-11'
  }
]
```

b) Retrieve Total Price of Orders Placed by a user (user with ID "123abc")

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

```
[{ _id: '123abc'
  total_order_price: 339.97 }]
```

Additional queries:

1. Calculate Total No of Products in each category.

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

Electronics: 3, Apparel: 2

PAGE NO :
DATE :

2. Calculate Total Price of Products in each category.

```
db.products.aggregate([
 { $group: {_id:"$category", total_price:
   { $sum: {$multiply: ["$price",
   "$available_quantity"]} }}}
]);
```

- electronics : 48998.2
- Apparel : 1149.65

3. Find Average Price of Products.

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

[{avg_price : 253.994}]

4. Find Products with quantity less than 10

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

5. Sort Products by Price in Descending Order.

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

PAGE NO :
DATE :

6. Calculate Total Price of Order Placed by Each User.

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

```
[ { _id: '123abc', total_price: 339.97
  _id: '789ghi', total_price: 249.97
  _id: '456def', total_price: 799.99 } ]
```

7. Find User with Highest Total Price of Order

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

```
[ { _id: "456def", total: 799.99 } ]
```

8. Find Average Total Price of Orders.

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

```
[ { _id: null, avg_order_price: 463.31 } ]
```

11/3/25