LAB-2 : Mongodb Excercises

(11/9/25)

① Customers :

i) Create a collection with attributes : cust_id , acc_bal, acc_type and insert atleast 5 values into the table.

```
> db.createCollection ("Customer");
{ok : 1}

> db. Customers. insertMany ([
  {cust_id : 1  , acc_bal : 1500, acc_type : 'Z'},
  {cust_id : 2  , acc_bal : 900, acc_type : 'X'},
  {cust_id : 3  , acc_bal : 2000, acc_type : "Z"},
  {cust_id : 4  , acc_bal : 1100, acc_type : "Y"},
  {cust_id : 5  , acc_bal : 1800, acc_type : 'Z'}
  ])
{acknowledged : true}
```

ii) write a query to display those records whose total balance is greater than 1200 of acc type 'Z' for each cust_id.

```
> db. Customers.find ({acc_bal : {$gt : 1200}, acc_type : "Z"})
```

iii) Determine min and max account balance for each customer

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

② E-commerce platform :

```
> db. createCollection ("Products")
> db. createCollection ("Users")
> db. createCollection ("orders")

> db. Products . insertMany ([
  {_id : 1, name : "Laptop", category : "Electronics", price : 800, quantity : 10},
  {_id : 2, name : "Phone", category : "Electronics", price : 500, quantity : 15}
  {_id : 3, name : "Headphone", cat : "Accessories", price : 50 , quantity : 25}
  {_id : 4, name : "Shoes", cat : "Fashion", price : 90 , quantity : 30}
  {_id : 5, name : "Washing machine", cat : "Appliance", price : 300 quantity : 5}
])
```

```
> db.Users.insertMany([
{_id: "123abc", name: "Alice", cart: [{prod-id: 1, quant: 1},
                                      {prod-id: 3, quant: 1}]},
{_id: "789ghi", name: "Bob", cart: [{prod-id: 2, quant: 1},
                                    {prod-id: 4, quant: 1}]}
])
```

• Retrieve all products

```
> db.Products.find()
```

• Retrieve Products in specific category (ex: Electronics).

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

• Retrieve Products with quantity greater than 0

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

• Retrieve Products sorted by price in ascending order

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

• Retrieve Products with price less than or equal to $100

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

• Retrieve orders placed by a user

```
> db.Orders.aggregate([{$match: {user-id: "123abc"}},
                {$group: {_id: "$user-id", total-spent:
                  {$sum: "$total-price"}}}])
```

• Retrieve Total price of orders placed by a user

```
> db.Products.aggregate([{$group: {_id: "$category",
            total-products: {$sum: 1}}}])
```

Additional queries:

1) Calculate total no. of prods in each category

```
> db.Products.aggregate([{$group: {_id: "$category",
      total-products: {$sum: 1}}}])
```

2) Calculate Total price of products in each category

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

3) Find average price of products.

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

4) Find prods with quantity less than 10

```
> db. Product. find ( {quantity : {$lt : 10}})
```

5) Sort prod. by price in descending order

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

6) Calculate ~~total~~ price of orders placed by each user

```
> db. Orders . agg ([{$group: {_id : "$user_id", total_sent: {$sum }}]
```

7) Find users with highest total price of orders

```
> db . orders . aggregate ([ {$group : {_id: "$user_id",
  total_spent : { $sum : "$total_price"}}}, {$sort : {
~~Find average total price of orders~~
  total_spent : -1}}, {$limit : 1}])
```

8) Find average total price of orders

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

[illegible] 1/3/25