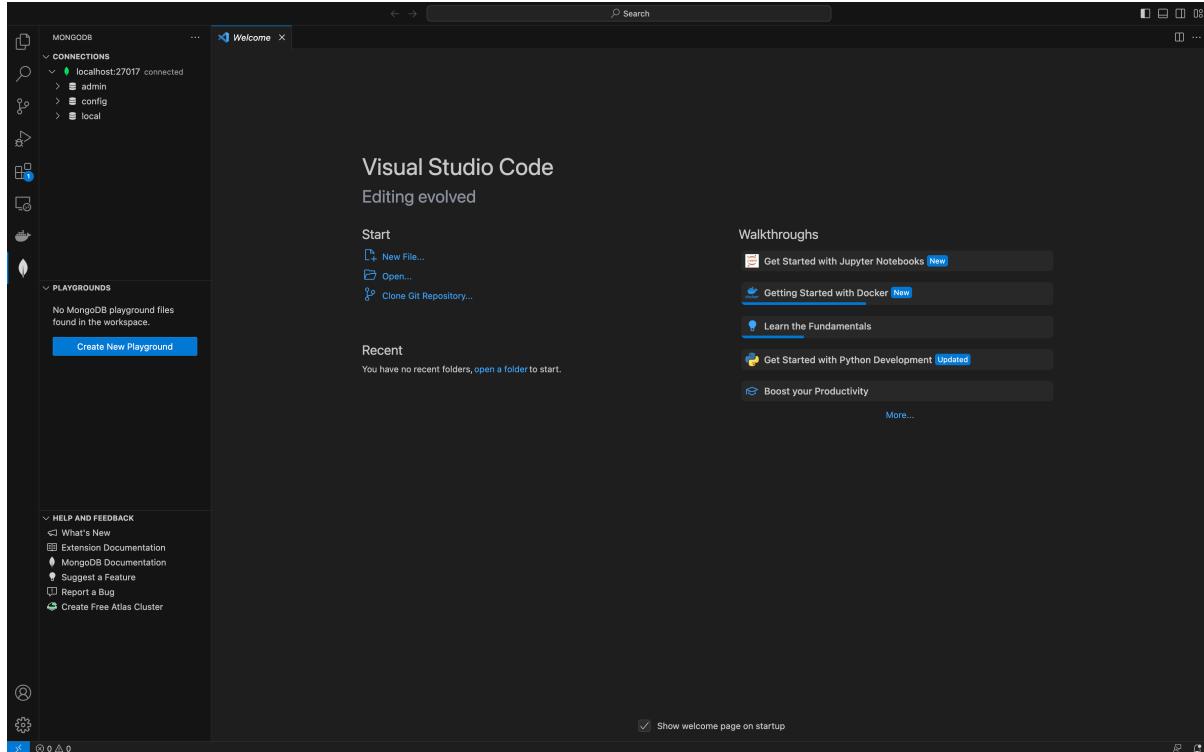


Part 1 : DB Systems

1. Open VS Code and connect to MONGODB

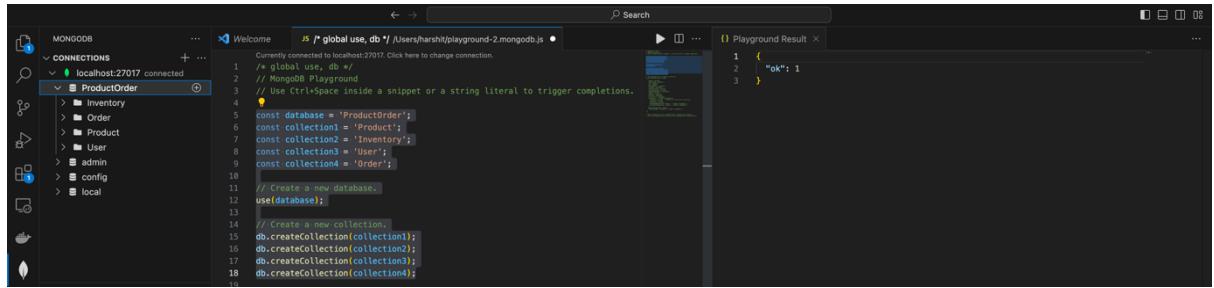
Following screenshot shows that MONGODB connection was successfully setup :



Sample playground screenshot:

A screenshot of the same Visual Studio Code interface, but now with a playground session open. The title bar shows the path '/global use, db /Users/harshit/playground-1.mongodb.js'. The main code editor contains the following JavaScript code for interacting with a MongoDB database:1 // global use, db /
2 // MongoDB Playground
3 // Make sure you are connected to enable completions and to be able to run a playground.
4 // Use Ctrl+Space inside a snippet or a string literal to trigger completions.
5 // The result of the last command run in a playground is shown on the results panel.
6 // By default the first 20 documents will be returned with a cursor.
7 // Use `console.log()` to print to the debug output.
8 // For more documentation on playgrounds please refer to
9 // <https://www.mongodb.com/docs/mongodb-vscode/playgrounds/>
10 // Select the database to use.
11 use('mongodbVsCodePlaygroundDB');
12 // Insert a few documents into the sales collection.
13 db.getCollection('sales').insertMany(
14 [{ item: 'abc', price: 10, quantity: 2, date: new Date('2014-01-01T00:00:00Z') },
15 { item: 'abc', price: 10, quantity: 20, date: new Date('2014-02-01T00:00:00Z') },
16 { item: 'xyz', price: 5, quantity: 1, date: new Date('2014-03-15T00:00:00Z') },
17 { item: 'xyz', price: 5, quantity: 20, date: new Date('2014-04-01T00:00:00Z') },
18 { item: 'xyz', price: 5, quantity: 10, date: new Date('2014-03-15T00:00:00Z') },
19 { item: 'xyz', price: 5, quantity: 20, date: new Date('2014-04-04T11:21:39.736Z') },
20 { item: 'abc', price: 10, quantity: 10, date: new Date('2014-04-04T21:23:13.331Z') },
21 { item: 'abc', price: 10, quantity: 10, date: new Date('2014-04-04T21:23:13.331Z') },
22 { item: 'def', price: 7.5, quantity: 5, date: new Date('2015-06-04T05:08:13Z') },
23 { item: 'def', price: 7.5, quantity: 10, date: new Date('2015-06-04T05:08:13Z') },
24 { item: 'abc', price: 10, quantity: 5, date: new Date('2016-02-06T20:20:13Z') },
25]);
26
27 // Run a find command to view items sold on April 4th, 2014.
28 const salesOnApril4th = db.getCollection('sales').find({
29 | date: { \$gte: new Date('2014-04-04'), \$lt: new Date('2014-04-05') }
30 }).count();
31
32 // Print a message to the output window.
33 console.log(`\$salesOnApril4th sales occurred in 2014.`);
34
35 // Here we run an aggregation and open a cursor to the results.
36 // Use `\$.toJSON()` to exhaust the cursor to return the whole result set.
37 // You can use `\$.hasNext()/.next()` to iterate through the cursor page by page.
38 db.getCollection('sales').aggregate([
39 { \$match: { date: { \$gt: new Date('2014-01-01'), \$lt: new Date('2015-01-01') } } },
40 { \$match: { total: { \$gt: 0 } } },
41 { \$group: { _id: '\$item', totalSaleAmount: { \$sum: { \$multiply: ['\$price', '\$quantity'] } } } }
42]);
43
44The status bar at the bottom indicates 'Ln 1, Col 1' and 'JavaScript'.

2. Create a database "ProductOrder" and create collections "Product", "Inventory", "User" and "Order" in it.



```

MONGOOB
  CONNECTIONS
    localhost:27017 connected
      ProductOrder
        Inventory
        Order
        Product
        User
      admin
      config
      local

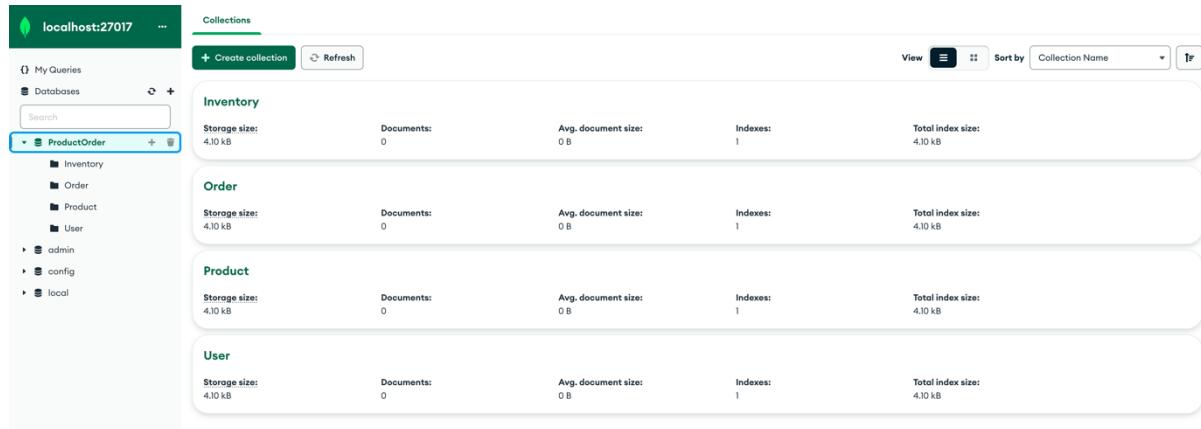
Welcome
JS /* global use, db */ /Users/harshit/playground-2.mongodbs.js
Currently connected to localhost:27017. Click here to change connection.

1 // global use, db /
2 // MongoDB Playground
3 // Use Ctrl+Space inside a snippet or a string literal to trigger completions.
4
5 const database = 'ProductOrder';
6 const collection1 = 'Product';
7 const collection2 = 'Inventory';
8 const collection3 = 'User';
9 const collection4 = 'Order';
10
11 // Create a new database.
12 use(database);
13
14 // Create a new collection.
15 db.createCollection(collection1);
16 db.createCollection(collection2);
17 db.createCollection(collection3);
18 db.createCollection(collection4);
19

```

3. Open MongoDBCompass and navigate to the "ProductOrder" database.

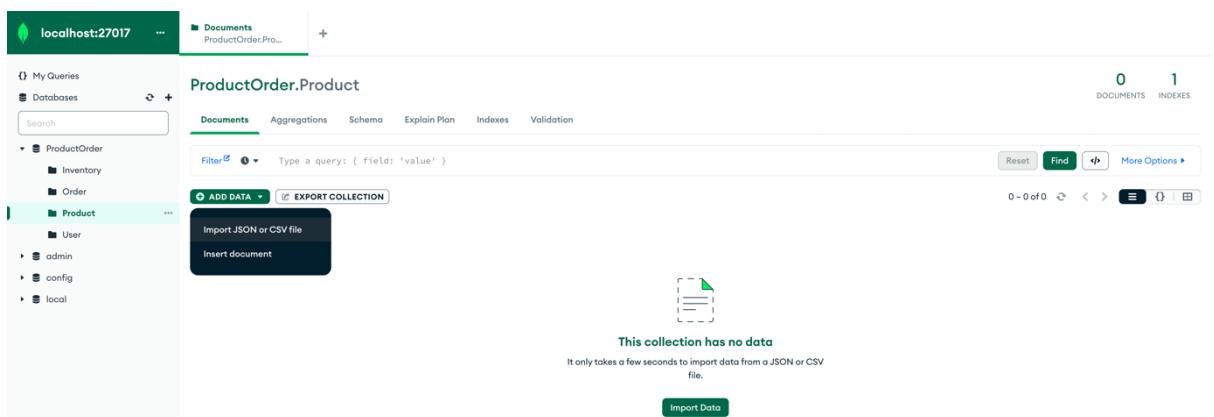
Before adding files:



Collection	Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
Inventory	4.10 kB	0	0 B	1	4.10 kB
Order	4.10 kB	0	0 B	1	4.10 kB
Product	4.10 kB	0	0 B	1	4.10 kB
User	4.10 kB	0	0 B	1	4.10 kB

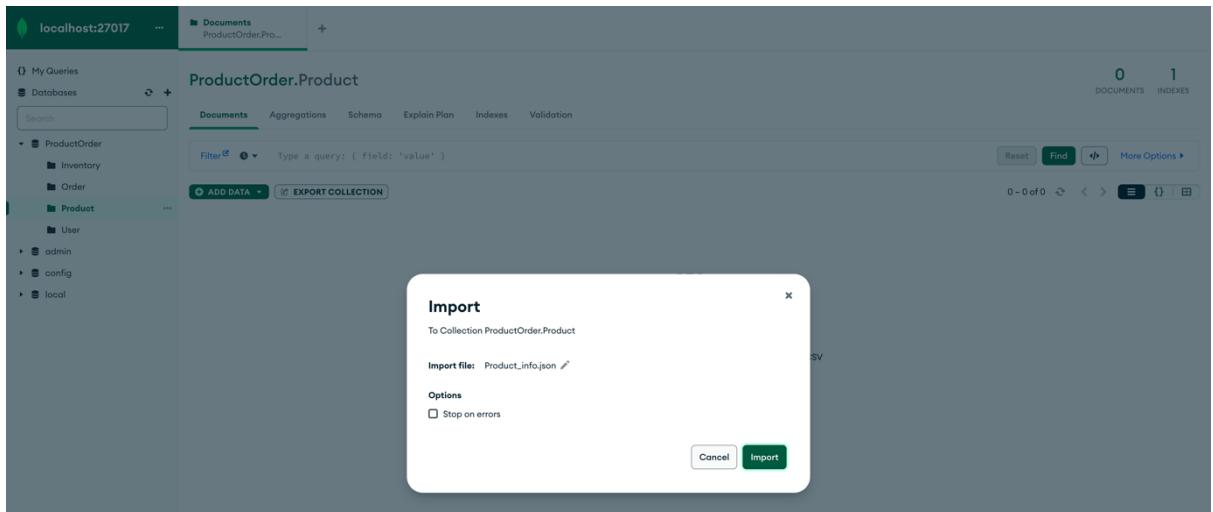
After adding the files:

- 3.1 Add "Product_info.json" file into the "Product" collection.



This collection has no data
It only takes a few seconds to import data from a JSON or CSV file.

Import Data



```

_id: ObjectId('646defa0556b3b0f42ac648e')
sku: "SMY-12801"
code: "Sony-01"
price: 100000
created: "2021-08-09 12:32:56"
last_updated: "2021-08-09 12:32:56"
brand: "Sony"
model: "Bravia-X"
warranty: 5

_id: ObjectId('646defa0556b3b0f42ac648f')
sku: "SMY-12802"
code: "Sony-02"
price: 100000
created: "2021-09-19 08:23:45"
last_updated: "2021-09-19 08:23:45"
brand: "Sony"
model: "Bravia-X"
warranty: 5

_id: ObjectId('646defa0556b3b0f42ac6490')
sku: "SMY-12803"
code: "Samsung-01"
price: 60000
created: "2021-07-13 16:46:32"
last_updated: "2021-07-13 16:46:32"
brand: "Samsung"
model: "Q2"
warranty: 3

_id: ObjectId('646defa0556b3b0f42ac6491')
sku: "LG-22801"
code: "LG-01"
price: 60000
created: "2021-10-28 18:34:21"
last_updated: "2021-10-28 18:34:21"

```

Import completed.
10 documents written.

The above screenshot confirms that 10 documents were imported into the Product collection from the json provided.

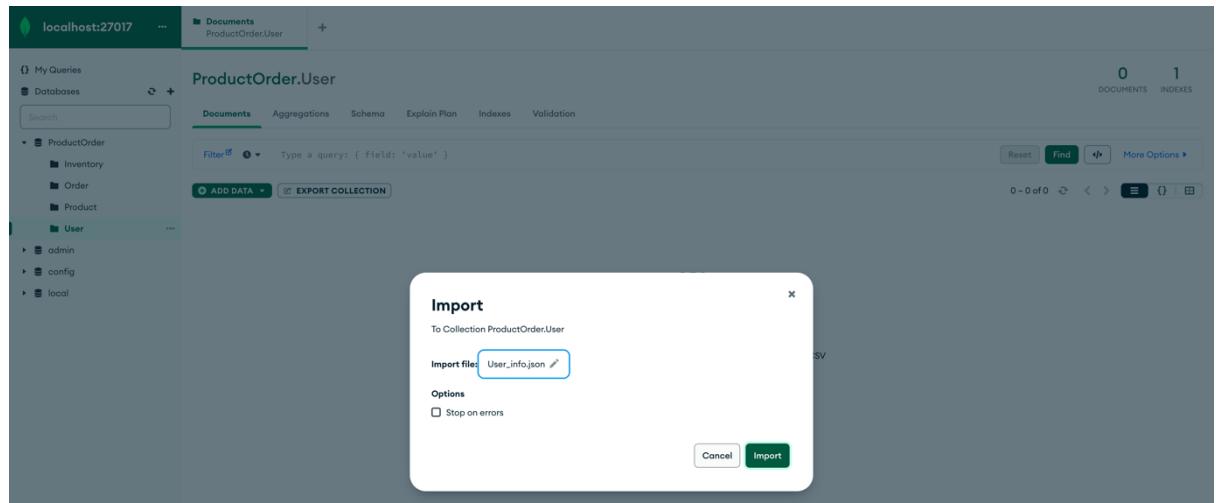
3.2 Add "Inventory_info.json" file into "Inventory" collection.

The screenshot shows the MongoDB Compass interface. On the left, the sidebar navigation includes 'My Queries', 'Databases', and a tree view for the 'ProductOrder' database with 'Inventory' selected. The main area is titled 'ProductOrder.Inventory' and shows a 'Documents' tab. A search bar and filter dropdown are at the top. Below is a table with columns 'Document' and 'More Options'. A modal window titled 'Import' is overlaid, showing the path 'To Collection ProductOrder.Inventory' and 'Import file: Inventory_info.json'. It has an 'Options' section with a checkbox 'Stop on errors' and two buttons 'Cancel' and 'Import'.

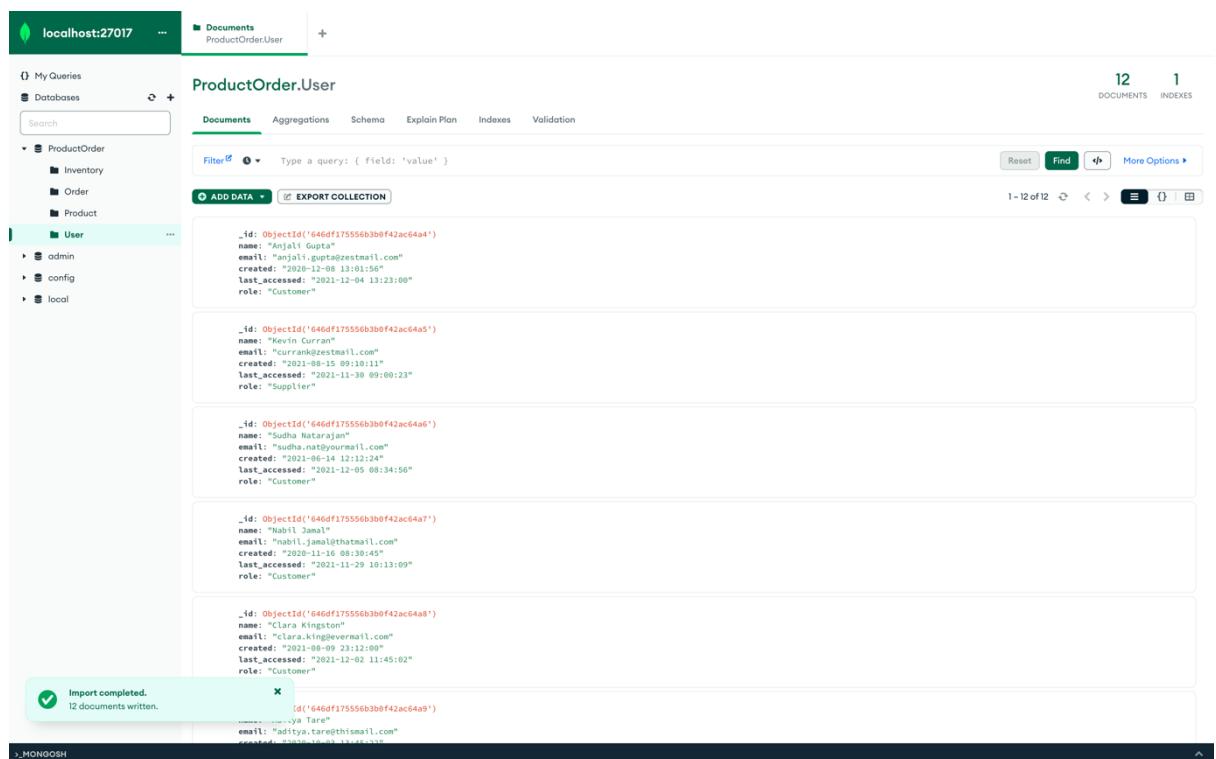
The screenshot shows the same MongoDB Compass interface after the import process. The 'Documents' table now lists 10 documents. Each document is represented by a row with an '_id' field containing a unique ObjectId, an 'sku' field, a 'quantity' field, and a 'last_updated' field showing a timestamp. At the bottom of the table, a green message box says 'Import completed. 10 documents written.' The overall interface remains consistent with the first screenshot, showing the same sidebar and main collection view.

The above screenshot confirms that 10 documents were imported into the Inventory collection from the json provided.

3.3 Add "User_info.json" file into the "User" collection.



The screenshot shows the MongoDB Compass interface with the database 'ProductOrder' selected. The 'User' collection is currently active. A modal window titled 'Import' is displayed, prompting the user to import data from a file named 'User_info.json'. The modal includes options for stopping imports if errors occur. After clicking 'Import', a confirmation message at the bottom indicates 'Import completed. 12 documents written.'



The screenshot shows the 'ProductOrder.User' collection in MongoDB Compass after the import process. The document list now displays 12 entries, each representing a user with fields such as _id, name, email, created, last_accessed, and role. A confirmation message at the bottom of the interface states 'Import completed. 12 documents written.'

The above screenshot confirms that 12 documents were imported into the User collection from the json provided.

3.4 Add "Order_info.json" file into "Order" collection.

The screenshot shows the MongoDB Compass interface with the database 'ProductOrder' and collection 'Order'. A modal window titled 'Import' is open, prompting to import data into the 'ProductOrder.Order' collection. The 'Import file' field contains 'Order_info.json'. There are options to 'Stop on errors' and buttons for 'Cancel' and 'Import'. The main interface shows 0 documents and 1 index.

The screenshot shows the 'ProductOrder.Order' collection after the import process. The document count is now 10 and the index count is 1. The list of 10 imported documents is displayed, each containing fields such as _id, created, last_updated, items, total_price, discount, net_price, status, and user_email. A message at the bottom indicates 'Import completed. 10 documents written.'

The above screenshot confirms that 10 documents were imported into the Order collection from the json provided.

After having imported all the json files, we have:

The screenshot shows the MongoDB Compass interface with the following details:

- localhost:27017** is the selected database.
- Collections** section:
 - Inventory**: Storage size: 4.10 kB, Documents: 10, Avg. document size: 93.00 B, Indexes: 1, Total index size: 4.10 kB.
 - Order**: Storage size: 4.10 kB, Documents: 10, Avg. document size: 300.00 B, Indexes: 1, Total index size: 4.10 kB.
 - Product**: Storage size: 4.10 kB, Documents: 10, Avg. document size: 189.00 B, Indexes: 1, Total index size: 4.10 kB.
 - User**: Storage size: 4.10 kB, Documents: 12, Avg. document size: 169.00 B, Indexes: 1, Total index size: 4.10 kB.

4. Display the first 5 rows of product, inventory, user, and order collection.

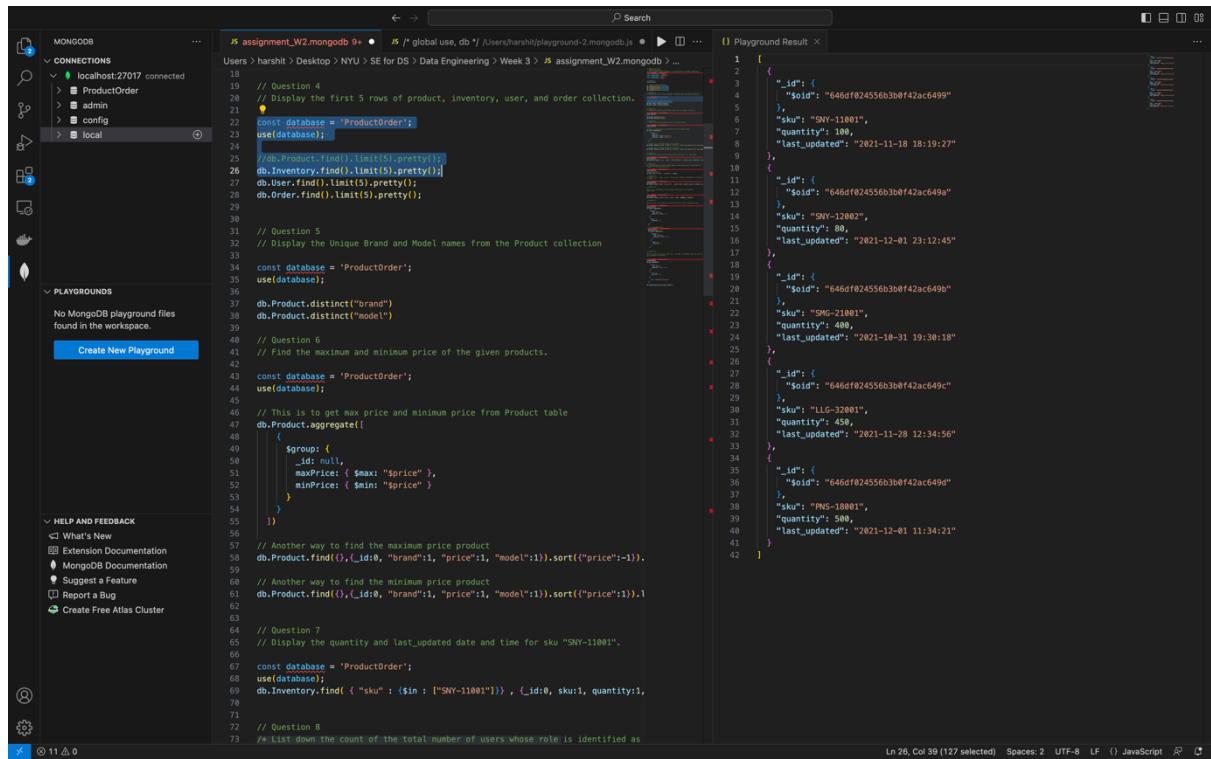
First 5 rows of Product:

The screenshot shows the MongoDB Compass playground with the following details:

- CONNECTIONS**: **localhost:27017 connected**.
- PLAYGROUNDS**: A script named **assignment_W2.mongodb.js** is running, containing the following code:19 // Question 4
20 // Display the first 5 rows of product, inventory, user, and order collection.
21
22 const database = 'ProductOrder';
23 useDatabase();
24
25 db.Product.find().limit(5).pretty();
26 db.Inventory.find().limit(5).pretty();
27 db.User.find().limit(5).pretty();
28 db.Order.find().limit(5).pretty();
- Playground Result**: The result shows the first 5 documents of the **Product** collection.

_id	name	brand	model	warranty
646defa0556b3b0f42ac648e	SONY-X1	Sony	X1	5
646defa0556b3b0f42ac648f	SONY-X2	Sony	X2	5
646defa0556b3b0f42ac6490	SONY-X3	Sony	X3	3
646defa0556b3b0f42ac6491	LG-LG01	LG	LG01	3

First 5 rows of Inventory:



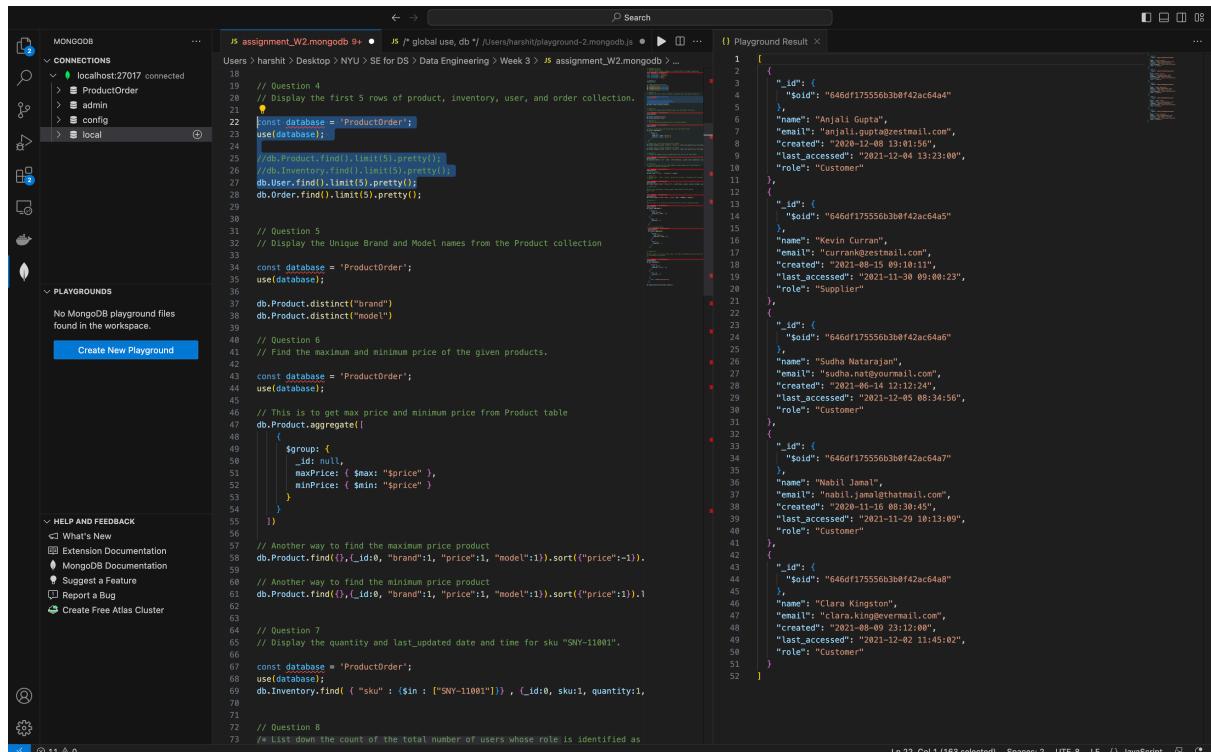
The screenshot shows the MongoDB Compass interface. On the left, the 'CONNECTIONS' panel lists 'localhost:27017 connected' with databases 'ProductOrder', 'admin', 'config', and 'local'. The 'PLAYGROUNDS' panel indicates 'No MongoDB playground files found in the workspace' and has a 'Create New Playground' button. The central area shows a code editor with a JavaScript file named 'assignment_W2.mongodb.js'. The code retrieves the first 5 rows from the 'Inventory' collection. The right panel displays the 'Playground Result' with the retrieved data, which includes documents for SKUs SNY-11001, SNY-12002, MG-21001, LIG-32001, and NS-18001.

```

1 [
2   {
3     "_id": {
4       "$oid": "646df024556b3bf42ac6499"
5     },
6     "sku": "SNY-11001",
7     "quantity": 100,
8     "last_updated": "2021-11-18 18:19:27"
9   },
10  {
11    "_id": {
12      "$oid": "646df024556b3bf42ac649a"
13    },
14    "sku": "SNY-12002",
15    "quantity": 80,
16    "last_updated": "2021-12-01 23:12:45"
17  },
18  {
19    "_id": {
20      "$oid": "646df024556b3bf42ac649b"
21    },
22    "sku": "MG-21001",
23    "quantity": 400,
24    "last_updated": "2021-10-31 19:30:18"
25  },
26  {
27    "_id": {
28      "$oid": "646df024556b3bf42ac649c"
29    },
30    "sku": "LIG-32001",
31    "quantity": 450,
32    "last_updated": "2021-11-28 12:34:56"
33  },
34  {
35    "_id": {
36      "$oid": "646df024556b3bf42ac649d"
37    },
38    "sku": "NS-18001",
39    "quantity": 500,
40    "last_updated": "2021-12-01 11:34:21"
41  },
42]

```

First 5 rows of User:



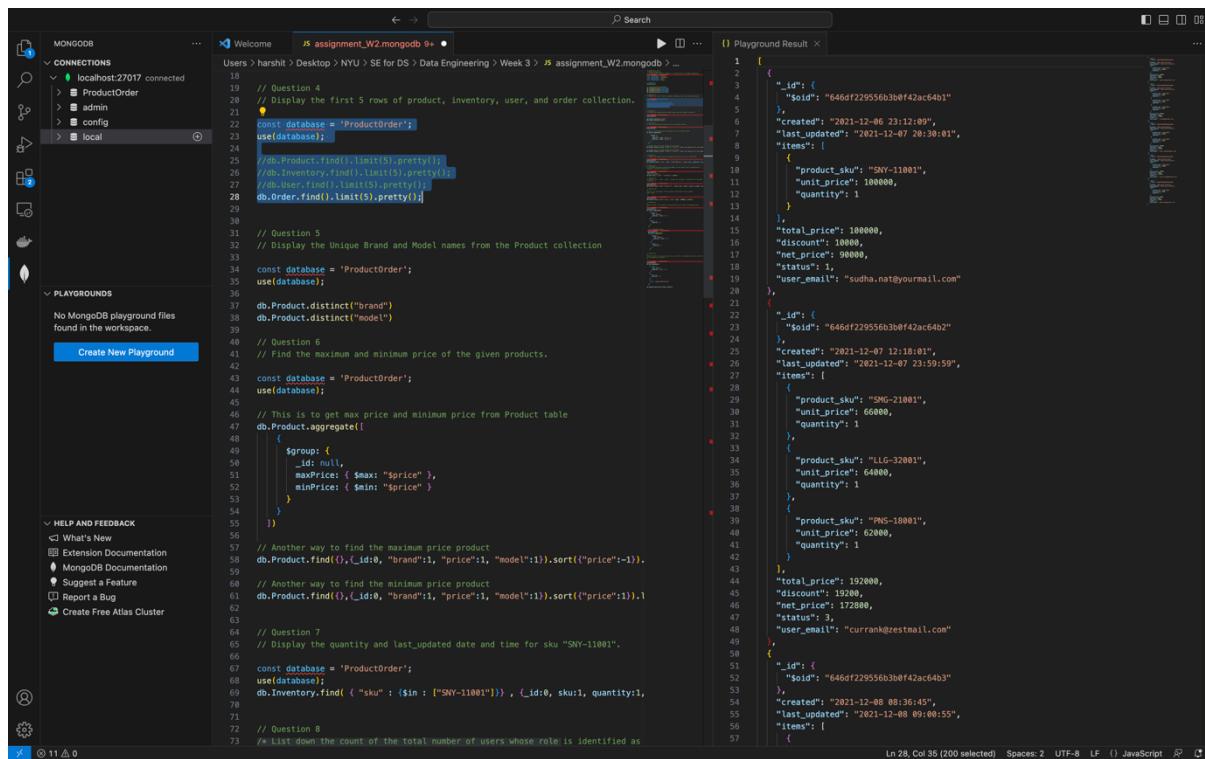
The screenshot shows the MongoDB Compass interface, identical to the previous one but with a different collection. The 'CONNECTIONS' panel shows 'localhost:27017 connected'. The 'PLAYGROUNDS' panel shows 'No MongoDB playground files found in the workspace'. The central code editor shows the same 'assignment_W2.mongodb.js' file. The right panel displays the 'Playground Result' with user documents for Anjali Gupta, Kevin Curran, Sudha Natarajan, Nabil Jamal, and Clara Kingston.

```

1 [
2   {
3     "_id": {
4       "$oid": "646df175556b3bf42ac64a4"
5     },
6     "name": "Anjali Gupta",
7     "email": "anjali.gupta@zestmail.com",
8     "created": "2020-12-08 13:01:56",
9     "last_accessed": "2021-12-04 13:23:00",
10    "role": "Customer"
11  },
12  {
13    "_id": {
14      "$oid": "646df175556b3bf42ac64a5"
15    },
16    "name": "Kevin Curran",
17    "email": "currankzestmail.com",
18    "created": "2021-08-15 09:10:11",
19    "last_accessed": "2021-11-30 09:00:23",
20    "role": "Supplier"
21  },
22  {
23    "_id": {
24      "$oid": "646df175556b3bf42ac64a6"
25    },
26    "name": "Sudha Natarajan",
27    "email": "sudha.nat@gmail.com",
28    "created": "2021-06-14 12:12:24",
29    "last_accessed": "2021-12-05 08:34:56",
30    "role": "Customer"
31  },
32  {
33    "_id": {
34      "$oid": "646df175556b3bf42ac64a7"
35    },
36    "name": "Nabil Jamal",
37    "email": "nabil.jamal@hotmail.com",
38    "created": "2020-11-16 08:30:45",
39    "last_accessed": "2021-11-29 10:13:09",
40    "role": "Customer"
41  },
42  {
43    "_id": {
44      "$oid": "646df175556b3bf42ac64a8"
45    },
46    "name": "Clara Kingston",
47    "email": "clara.king@vermail.com",
48    "created": "2021-08-09 23:12:00",
49    "last_accessed": "2021-12-02 11:45:02",
50    "role": "Customer"
51  }
52]

```

First 5 rows of Order:



The screenshot shows the MongoDB Compass interface. On the left, the 'CONNECTIONS' sidebar lists 'localhost:27017 connected' and the 'ProductOrder' database. The 'PLAYGROUNDS' sidebar indicates 'No MongoDB playground files found in the workspace'. The main area displays a code editor with the following MongoDB query:

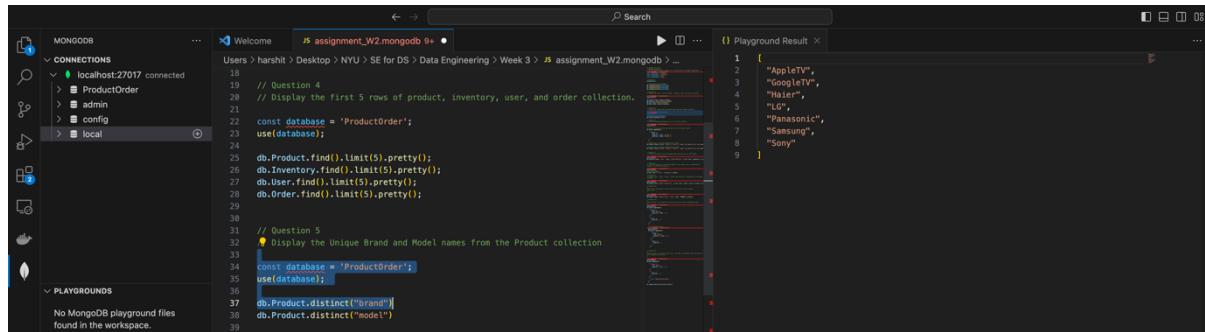
```

1 // Question 4
2 // Display the first 5 rows of product, inventory, user, and order collection.
3
4 const database = 'ProductOrder';
5 useDatabase();
6
7 db.Product.find().limit(5).pretty();
8 db.Inventory.find().limit(5).pretty();
9 db.User.find().limit(5).pretty();
10 db.Order.find().limit(5).pretty();
11
12
13 // Question 5
14 // Display the Unique Brand and Model names from the Product collection
15
16 const database = 'ProductOrder';
17 useDatabase();
18
19 db.Product.distinct("brand")
20 db.Product.distinct("model")
21
22 // Question 6
23 // Find the maximum and minimum price of the given products.
24
25 const database = 'ProductOrder';
26 useDatabase();
27
28 // This is to get max price and minimum price from Product table
29 db.Product.aggregate([
30   {
31     $group: {
32       _id: null,
33       maxPrice: { $max: "$price" },
34       minPrice: { $min: "$price" }
35     }
36   }
37 ]
38 )
39
40 // Another way to find the maximum price product
41 db.Product.find({ '_id': 0, "brand": 1, "model": 1 }).sort({ "price": -1 });
42
43 // Another way to find the minimum price product
44 db.Product.find({ '_id': 0, "brand": 1, "model": 1 }).sort({ "price": 1 });
45
46 // Question 7
47 // Display the quantity and last_updated date and time for sku "SNY-11001".
48
49 const database = 'ProductOrder';
50 useDatabase();
51 db.Inventory.find({ "sku": { $in: ["SNY-11001"] } }, { '_id': 0, sku: 1, quantity: 1 },
52
53 // Question 8
54 // List down the count of the total number of users whose role is identified as
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
    
```

The 'Playground Result' panel on the right shows the output of the query, which includes five documents from the Order collection, each containing an '_id', 'soldId', 'created', 'last_updated', and an array of 'items'. Each item has a 'product_sku', 'unit_price', and 'quantity' field.

5. Display the Unique Brand and Model names from the Product collection.

Unique brand names:



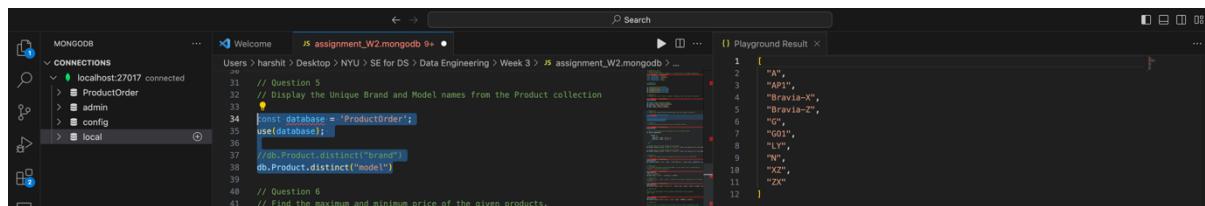
The screenshot shows the MongoDB Compass interface. The 'CONNECTIONS' sidebar lists 'localhost:27017 connected' and the 'ProductOrder' database. The 'PLAYGROUNDS' sidebar indicates 'No MongoDB playground files found in the workspace'. The main area displays a code editor with the following MongoDB query:

```

1 // Question 4
2 // Display the first 5 rows of product, inventory, user, and order collection.
3
4 const database = 'ProductOrder';
5 useDatabase();
6
7 db.Product.find().limit(5).pretty();
8 db.Inventory.find().limit(5).pretty();
9 db.User.find().limit(5).pretty();
10 db.Order.find().limit(5).pretty();
11
12
13 // Question 5
14 // Display the Unique Brand and Model names from the Product collection
15
16 const database = 'ProductOrder';
17 useDatabase();
18
19 db.Product.distinct("brand")
20 db.Product.distinct("model")
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
    
```

The 'Playground Result' panel on the right shows the output of the query, which is an array of unique brand names: AppleTV+, GoogleTV, Haier, LG, Panasonic, Samsung, and Sony.

Unique model names:



The screenshot shows the MongoDB Compass interface. The 'CONNECTIONS' sidebar lists 'localhost:27017 connected' and the 'ProductOrder' database. The 'PLAYGROUNDS' sidebar indicates 'No MongoDB playground files found in the workspace'. The main area displays a code editor with the following MongoDB query:

```

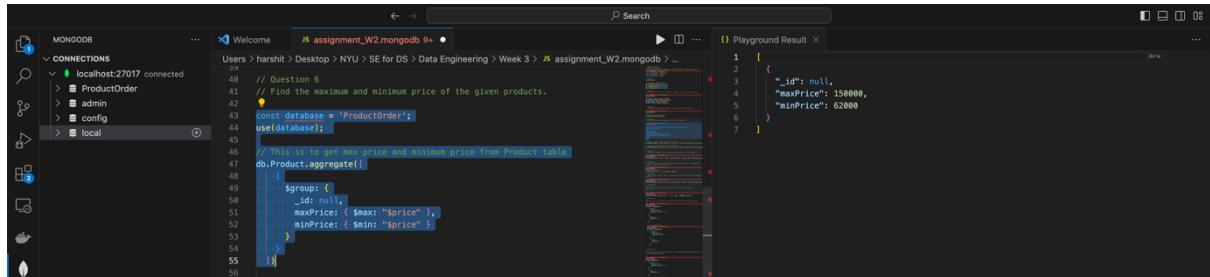
1 // Question 4
2 // Display the first 5 rows of product, inventory, user, and order collection.
3
4 const database = 'ProductOrder';
5 useDatabase();
6
7 db.Product.find().limit(5).pretty();
8 db.Inventory.find().limit(5).pretty();
9 db.User.find().limit(5).pretty();
10 db.Order.find().limit(5).pretty();
11
12
13 // Question 5
14 // Display the Unique Brand and Model names from the Product collection
15
16 const database = 'ProductOrder';
17 useDatabase();
18
19 db.Product.distinct("brand")
20 db.Product.distinct("model")
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
    
```

The 'Playground Result' panel on the right shows the output of the query, which is an array of unique model names: X, APT, Bravia-X, Bravia-Z, G, G01, LY, N, ZX, and ZX.

6. Find the maximum and minimum price of the given products.

Approach 1:

In this approach, I have used the aggregate function to find out the maximum and minimum prices in the Product collection:



The screenshot shows the MongoDB Compass interface. On the left, the 'CONNECTIONS' sidebar lists 'localhost:27017 connected' and 'ProductOrder'. The main area displays a code editor with the following aggregation pipeline:

```
// Question 6
// Find the maximum and minimum price of the given products.
const database = 'ProductOrder';
use(database);

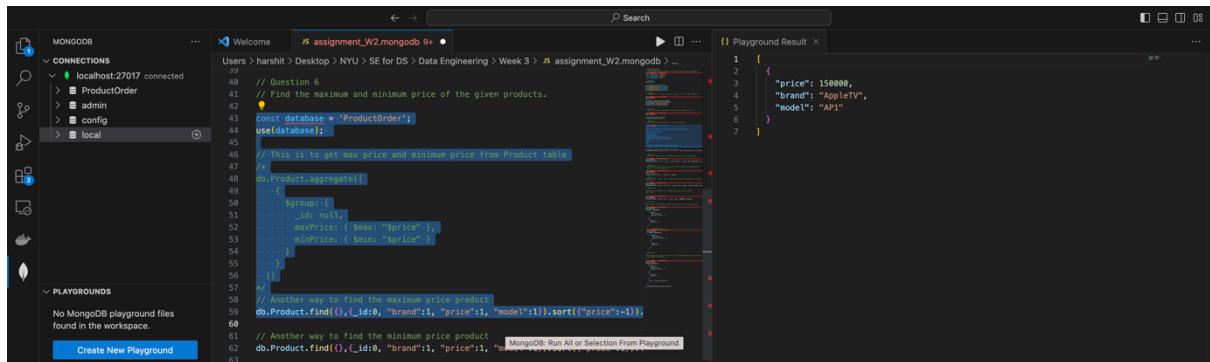
// This is to get max price and minimum price from Product table
db.Product.aggregate([
  {
    $group: {
      _id: null,
      maxPrice: { $max: "$price" },
      minPrice: { $min: "$price" }
    }
  }
])
```

To the right, the 'Playground Result' panel shows the output:

```
[{"_id": null, "maxPrice": 150000, "minPrice": 62000}]
```

Approach 2:

In this approach, I have sorted the documents in Product collection as per price and then displayed the topmost document. If the order of sorting is descending, we get the maximum price product and if the order of sorting is ascending, we get the minimum price product. Sorting in descending order to get maximum price product:



The screenshot shows the MongoDB Compass interface. On the left, the 'CONNECTIONS' sidebar lists 'localhost:27017 connected' and 'ProductOrder'. The main area displays a code editor with the following code:

```
// Question 6
// Find the maximum and minimum price of the given products.
const database = 'ProductOrder';
use(database);

// This is to get max price and minimum price from Product table
db.Product.aggregate([
  {
    $group: {
      _id: null,
      maxPrice: { $max: "$price" },
      minPrice: { $min: "$price" }
    }
  }
])

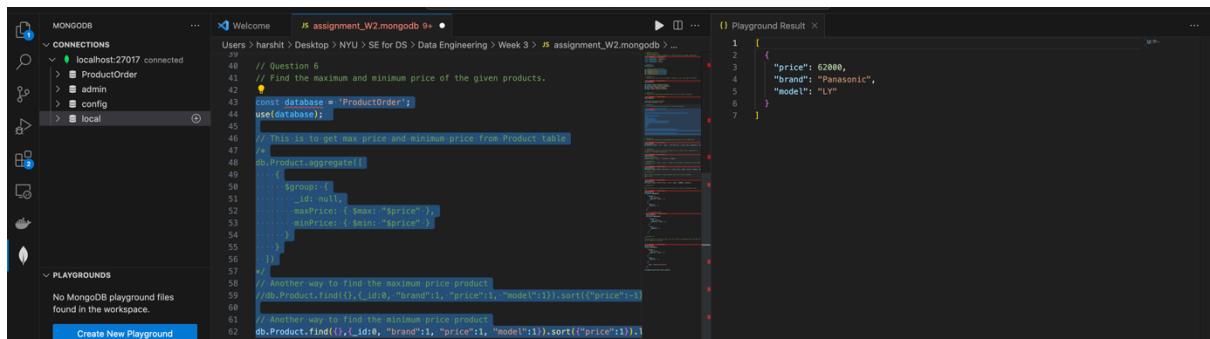
// Another way to find the maximum price product
db.Product.find({}).sort({price: -1}).limit(1)

// Another way to find the minimum price product
db.Product.find({}).sort({price: 1}).limit(1)
```

To the right, the 'Playground Result' panel shows the output for the maximum price product:

```
[{"price": 150000, "brand": "AppleTV", "model": "A1"}]
```

Sorting in ascending order to get minimum price product:



The screenshot shows the MongoDB Compass interface. On the left, the 'CONNECTIONS' sidebar lists 'localhost:27017 connected' and 'ProductOrder'. The main area displays a code editor with the same code as the previous screenshot:

```
// Question 6
// Find the maximum and minimum price of the given products.
const database = 'ProductOrder';
use(database);

// This is to get max price and minimum price from Product table
db.Product.aggregate([
  {
    $group: {
      _id: null,
      maxPrice: { $max: "$price" },
      minPrice: { $min: "$price" }
    }
  }
])

// Another way to find the maximum price product
db.Product.find({}).sort({price: -1}).limit(1)

// Another way to find the minimum price product
db.Product.find({}).sort({price: 1}).limit(1)
```

To the right, the 'Playground Result' panel shows the output for the minimum price product:

```
[{"price": 62000, "brand": "Panasonic", "model": "L7"}]
```

In aggregate function approach we are losing out on the other details of the corresponding document.

7. Display the quantity and last updated date and time for sku "SNY-11001".

```

    const database = 'ProductOrder';
    useDatabase();
    db.Inventory.find({ $in: [ { sku: "SNY-11001" } ] }, { _id: 0, sku1: 1, quantity: 1, last_updated: 1 });
  
```

The screenshot shows the MongoDB Compass interface with a connection to 'localhost:27017'. A query is being run against the 'Inventory' collection in the 'ProductOrder' database. The query filters for the SKU 'SNY-11001' and selects the '_id' (omitted), 'sku1' (set to 1), 'quantity', and 'last_updated' fields.

8. List down the count of the total number of users whose role is identified as 'Supplier' from User collection

```

    const database = 'ProductOrder';
    useDatabase();
    db.User.find({ "role": "Customer" }).count()
  
```

The screenshot shows the MongoDB Compass interface with a connection to 'localhost:27017'. A query is being run against the 'User' collection in the 'ProductOrder' database. The query filters for users with the role 'Customer' and counts them.

9. Display 'sku', 'code', 'price', 'brand' and 'warranty' information for the model 'Bravia-X' .

```

    const database = 'ProductOrder';
    useDatabase();
    db.Product.find({ "model": "Bravia-X" }, { _id: 0, sku1: 1, code1: 1, price1: 1, brand1: 1, warranty1: 1 })
  
```

The screenshot shows the MongoDB Compass interface with a connection to 'localhost:27017'. A query is being run against the 'Product' collection in the 'ProductOrder' database. The query filters for the model 'Bravia-X' and selects the '_id' (omitted), 'sku1', 'code1', 'price1', 'brand1', and 'warranty1' fields.

10. Find all the information of Sony products which have an Price greater than 1 lakh.

```

    const database = 'ProductOrder';
    useDatabase();
    db.Product.find({ "brand": "Sony", "price": { $gt: 100000 } }).pretty()
  
```

The screenshot shows the MongoDB Compass interface with a connection to 'localhost:27017'. A query is being run against the 'Product' collection in the 'ProductOrder' database. The query filters for products from the brand 'Sony' where the price is greater than 100,000 and uses the 'pretty()' method to format the output.

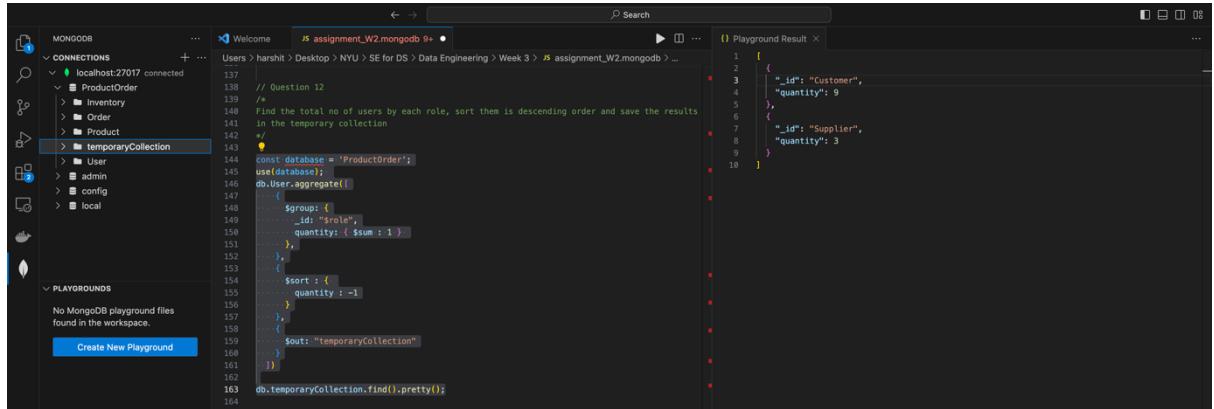
11. Find the total no of products by each Brand and sort them in descending order.

```

    db.Product.aggregate([
      {
        $group: {
          _id: "$brand",
          quantity: { $sum: 1 }
        }
      },
      {
        $sort: {
          quantity: -1
        }
      }
    ])
  
```

The screenshot shows the MongoDB Compass interface with a connection to 'localhost:27017'. A query is being run against the 'Product' collection in the 'ProductOrder' database. The query uses the 'aggregate()' method with two stages: grouping by brand and summing the quantity, and then sorting the results by quantity in descending order.

12. Find the total no of users by each role, sort them in descending order and save the results in the temporary collection.



The screenshot shows the MongoDB Compass interface. On the left, the 'CONNECTIONS' panel lists 'localhost:27017 connected' and the 'ProductOrder' database, which contains collections for 'Inventory', 'Order', and 'Product'. A new collection named 'temporaryCollection' is highlighted in blue. The 'PLAYGROUNDS' panel shows a message: 'No MongoDB playground files found in the workspace.' Below it is a 'Create New Playground' button. The main area displays a MongoDB query in the 'Welcome' tab of a playground file named 'assignment_W2.mongodb'. The code uses the aggregation framework to group users by role, sum their quantities, and then sort the results by quantity in descending order. The resulting document is shown in the 'Playground Result' panel:

```
1 [ 2 { 3   "_id": "Customer", 4   "quantity": 9 5 }, 6 { 7   "_id": "Supplier", 8   "quantity": 3 9 } 10 ] 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
```

```
1 [ 2 { 3   "_id": "Customer", 4   "quantity": 9 5 }, 6 { 7   "_id": "Supplier", 8   "quantity": 3 9 } 10 ] 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
```

Note: A new collection gets created by the name of “temporaryCollection” in the ProductOrder database and this can be seen on the left panel (temporaryCollection is highlighted).