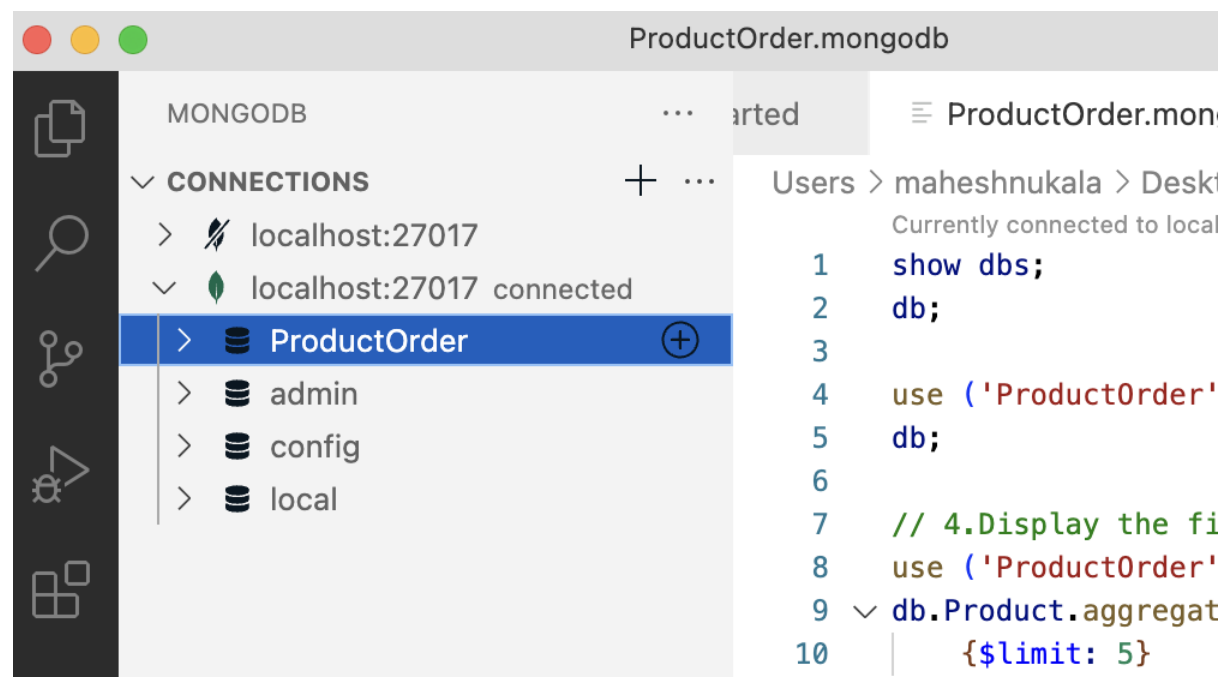
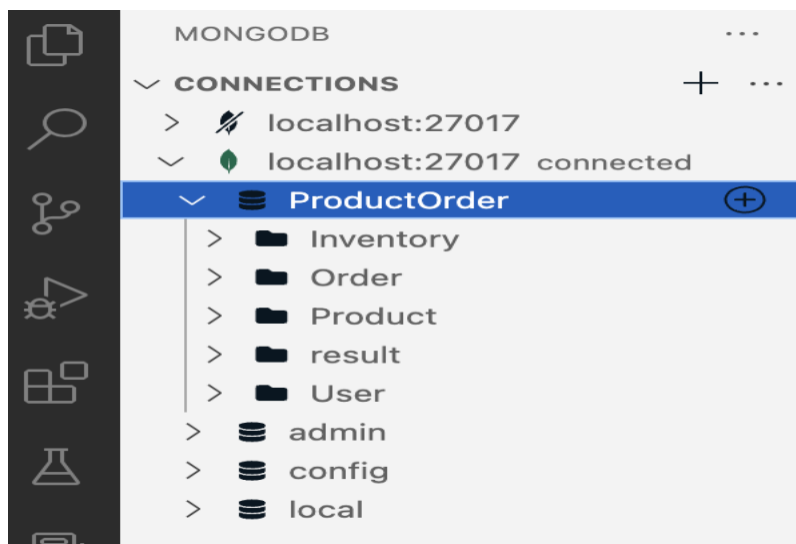
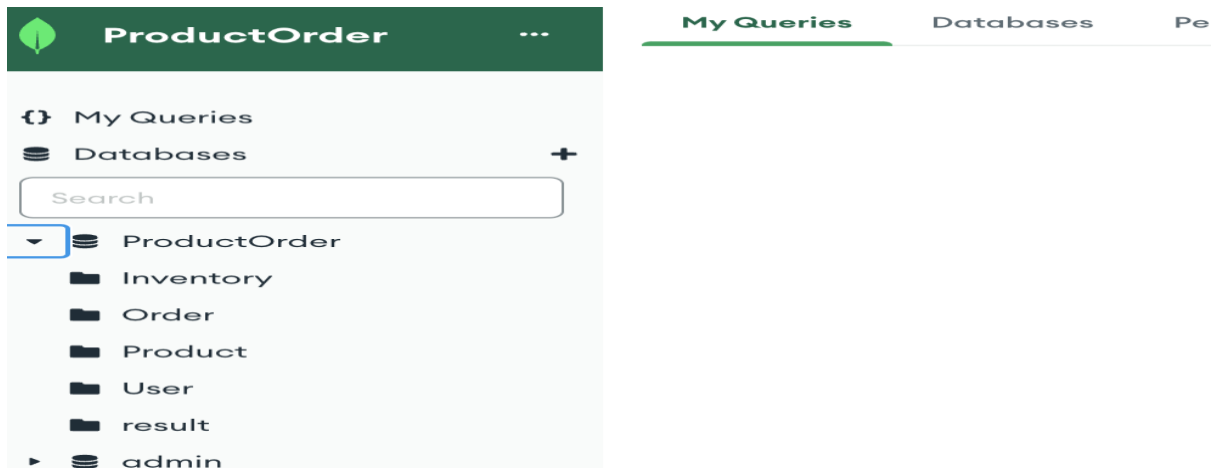


1. Open VS Code and connect to MONGODB



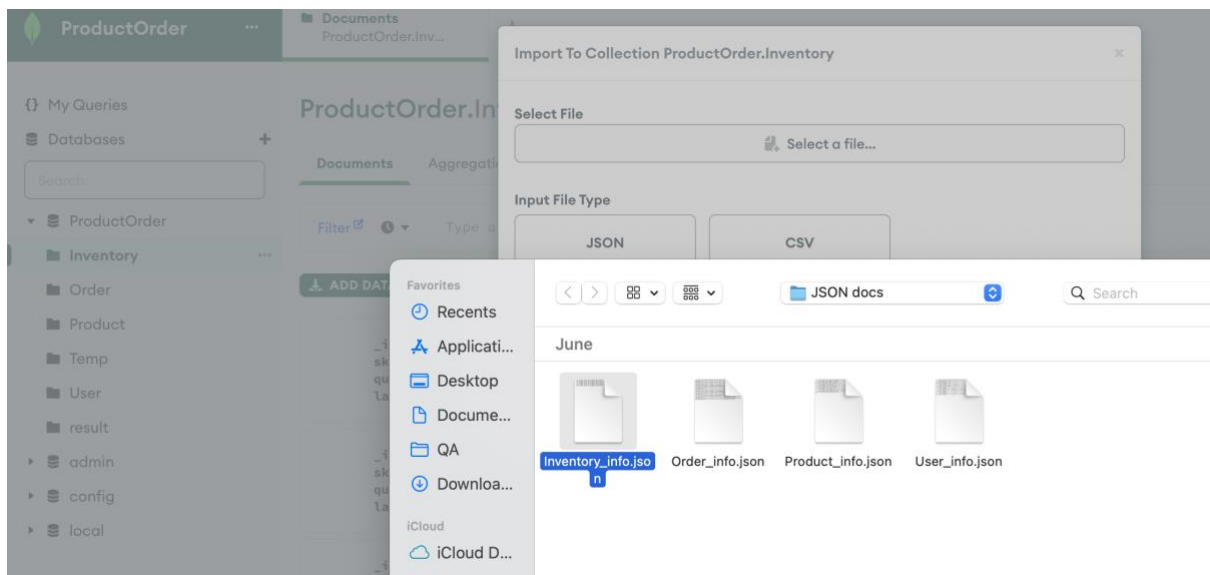
2. Create a database "ProductOrder" and create collections "Product","Inventory","User", and "Order" in it.





3. Open MongoDB Compass and navigate to the "ProductOrder" database.

- i) Add "Product_info.json" file into the "Product" collection.
- ii) Add "Inventory_info.json" file into "Inventory" collection.
- iii) Add "User_info.json" file into the "User" collection.
- iv) Add "Order_info.json" file into "Order" collection.



Same as above pic added all the files into the collections

Inventory data added:

DocumentsProductOrder.Inv...

+

ProductOrder.Inventory

Documents

Aggregations

Schema

Explain Plan

Indexes

Validation

Filter

⌚

Type a query: { field: 'value' }

ADD DATA

EXPORT COLLECTION

▶

```
    sku: "SNY-11001"
  quantity: 100
  last_updated: "2021-11-18 18:19:27"

  _id: ObjectId('6386f6d87c17a8c4b99d430e')
  sku: "SNY-12002"
  quantity: 80
  last_updated: "2021-12-01 23:12:45"

  _id: ObjectId('6386f6d87c17a8c4b99d430f')
  sku: "SMG-21001"
  quantity: 400
  last_updated: "2021-10-31 19:30:18"

  _id: ObjectId('6386f6d87c17a8c4b99d4310')
  sku: "LLG-32001"
  quantity: 450
  last_updated: "2021-11-28 12:34:56"

  _id: ObjectId('6386f6d87c17a8c4b99d4311')
  sku: "PNS-18001"
  quantity: 500
  last_updated: "2021-12-01 11:34:21"

  _id: ObjectId('6386f6d87c17a8c4b99d4312')
```

Product data added

ProductOrder.Product

Documents

Aggregations

Schema

Explain Plan

Indexes

Filter

⌚

Type a query: { field: 'value' }

ADD DATA

EXPORT COLLECTION

▶

```
  _id: ObjectId('6386f6bd7c17a8c4b99d4302')
  sku: "SNY-11001"
  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('6386f6bd7c17a8c4b99d4303')
  sku: "SNY-12002"
  code: "Sony-02"
  price: 120000
  created: "2021-09-19 08:23:45"
  last_updated: "2021-09-19 08:23:45"
```

6386f6bd7c17a8c4b99d4303

Order data added

ProductOrder.Order

Documents

Aggregations

Schema

Explain Plan

Filter



Type a query: { field: 'value' }

ADD DATA

EXPORT COLLECTION

```
_id: ObjectId('6386f6e27c17a8c4b99d4318')
created: "2021-12-06 23:12:09"
last_updated: "2021-12-07 20:30:01"
> items: Array
total_price: 100000
discount: 10000
net_price: 90000
status: 1
user_email: "sudha.nat@yourmail.com"
```

```
_id: ObjectId('6386f6e27c17a8c4b99d4319')
created: "2021-12-07 12:18:01"
last_updated: "2021-12-07 23:59:59"
> items: Array
total_price: 192000
```

User data added

ProductOrder.User



ProductOrder.User

Documents

Aggregations

Schema

Explain Plan

In

Filter



Type a query: { field: 'value' }

ADD DATA

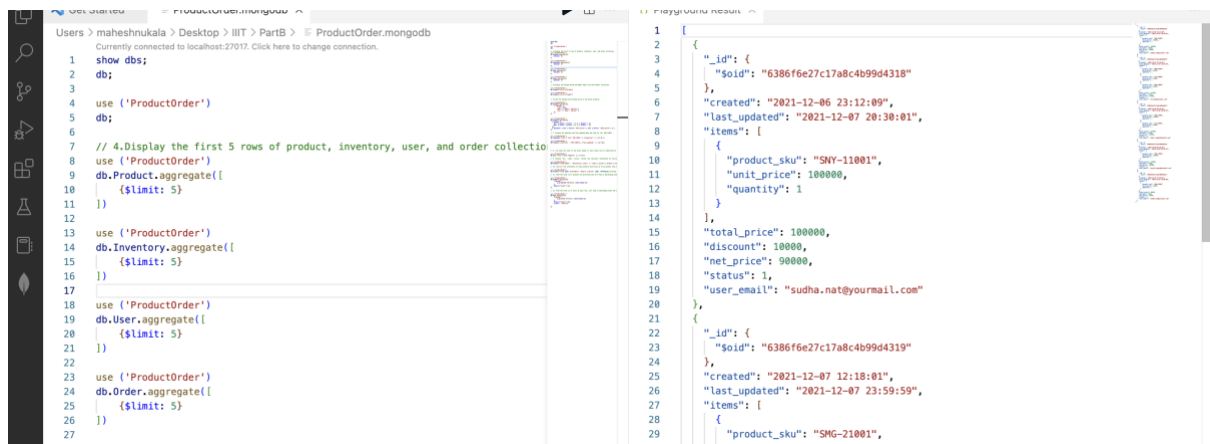
EXPORT COLLECTION

```
_id: ObjectId('6386f6ee7c17a8c4b99d4324')
name: "Anjali Gupta"
email: "anjali.gupta@zestmail.com"
created: "2020-12-08 13:01:56"
last_accessed: "2021-12-04 13:23:00"
role: "Customer"
```



```
_id: ObjectId('6386f6ee7c17a8c4b99d4325')
```

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



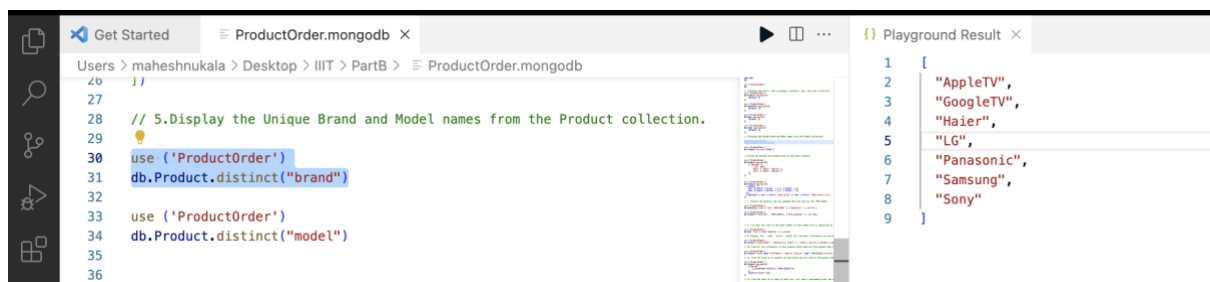
The screenshot shows the MongoDB Playground interface with four aggregation queries in the ProductOrder.mongodb database. The queries are as follows:

```
1 show dbs;
2 db;
3
4 use ('ProductOrder')
5 db;
6
7 // 4.Display the first 5 rows of product, inventory, user, and order collectio
8 use ('ProductOrder')
9 db.Product.aggregate([
10   {$limit: 5}
11 ])
12
13 use ('ProductOrder')
14 db.Inventory.aggregate([
15   {$limit: 5}
16 ])
17
18 use ('ProductOrder')
19 db.User.aggregate([
20   {$limit: 5}
21 ])
22
23 use ('ProductOrder')
24 db.Order.aggregate([
25   {$limit: 5}
26 ])
27
```

The results on the right show the first five documents from each collection:

- Product:** Documents with fields like `_id`, `$oid`, `created`, `last_updated`, and `items`.
- Inventory:** Documents with fields like `product_sku`, `unit_price`, `quantity`, `total_price`, `discount`, `net_price`, `status`, and `user_email`.
- User:** Documents with fields like `_id`, `$oid`, `created`, `last_updated`, and `items`.
- Order:** Documents with fields like `_id`, `$oid`, `created`, `last_updated`, and `items`.

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



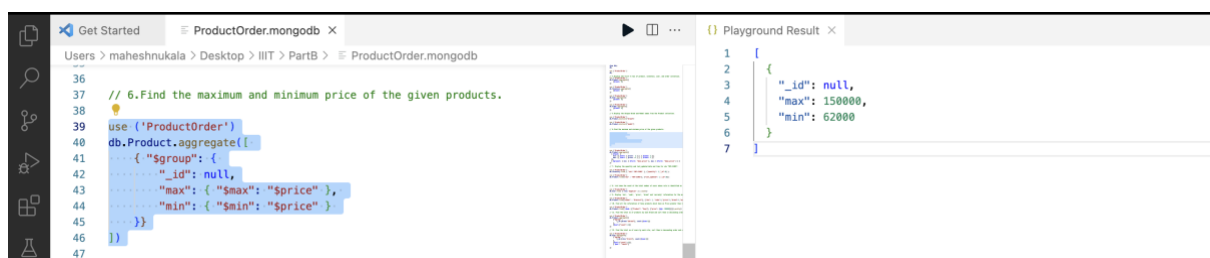
The screenshot shows the MongoDB Playground interface with two distinct queries in the ProductOrder.mongodb database. The queries are as follows:

```
27
28 // 5.Display the Unique Brand and Model names from the Product collection.
29
30 use ('ProductOrder')
31 db.Product.distinct("brand")
32
33 use ('ProductOrder')
34 db.Product.distinct("model")
35
36
```

The results on the right show the unique brand and model names:

- Brand:** AppleTV, GoogleTV, Haier, LG, Panasonic, Samsung, Sony.
- Model:** A, API, Bravia-X, Bravia-Z, G, G01, LY, N, XZ, ZX.

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



The screenshot shows the MongoDB Playground interface with an aggregation query in the ProductOrder.mongodb database. The query is as follows:

```
36
37 // 6.Find the maximum and minimum price of the given products.
38
39 use ('ProductOrder')
40 db.Product.aggregate([
41   {$group: {
42     "_id": null,
43     "max": { "$max": "$price" },
44     "min": { "$min": "$price" }
45   }}
46 ])
47
```

The results on the right show the maximum and minimum price of the given products:

```
{
  "_id": null,
  "max": 150000,
  "min": 62000
}
```

```

47
48
49 use ('ProductOrder')
50 db.Product.aggregate([
51   { $facet: {
52     min: [{ $sort: { price: 1 } }, { $limit: 1 }],
53     max: [{ $sort: { price: -1 } }, { $limit: 1 }],
54   }},
55   { $project: { min: { $first: "$min.price" }, max: { $first: "$max.price" } } }
56 ]])
57

```

```

1 [
2   {
3     "min": 62000,
4     "max": 150000
5   }
6 ]

```

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

```

57
58 // 7. Display the quantity and last_updated date and time for sku "SNY-11001".
59
60 use ('ProductOrder')
61 db.Inventory.find( { "sku": 'SNY-11001' }, { "quantity": 1, "_id": 0 } );
62
63 use ('ProductOrder')
64 db.Product.find( { "sku": "SNY-11001" }, { "last_updated": 1, "_id": 0 } );
65
66
67
68

```

```

1 [
2   {
3     "quantity": 100
4   }
5 ]

```

```

1 [
2   {
3     "last_updated": "2021-08-09 12:32:56"
4   }
5 ]

```

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

```

67
68
69 // 8. List down the count of the total number of users whose role is identified as 'Supplier'
70
71 use ('ProductOrder')
72 db.User.find( { role: 'Supplier' }, { count: 1 } );
73

```

```

1 3

```

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

```

74 // 9. Display 'sku', 'code', 'price', 'brand' and 'warranty' information for the model 'Bravia-X'
75
76 use ('ProductOrder')
77 db.Product.find( { "model": "Bravia-X", "sku": 1, "code": 1, "price": 1, "brand": 1, "warranty": 1, "_id": 0 } );
78
79 // 10. Find all the information of Sony products which have an Price greater than 1 lakh
80

```

```

1 [
2   {
3     "sku": "SNY-11001",
4     "code": "Sony-01",
5     "price": 100000,
6     "brand": "Sony",
7     "warranty": 5
8   }
9 ]

```

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

```

78 // 10. Find all the information of Sony products which have an Price greater than 1 lakh
79
80 use ('ProductOrder')
81 db.Product.find({ $and: [{ "brand": "Sony", ("price": { $gt: 100000 }) } ] })
82
83 // 11. Find the total no of products by each Brand and sort them in descending order.
84
85 use ('ProductOrder')
86 db.Product.aggregate([
87   { "$group" :
88     { "_id": {brand:"$brand"}, count:{$sum:1} }
89   },
90   { $sort: {"count":-1} }
91 ])

```

```

1 {
2   "_id": {
3     "brand": "Sony-82",
4     "sku": "SNY-12002",
5     "code": "Sony-82",
6     "price": 120000,
7     "created": "2021-09-19 08:23:45",
8     "last_updated": "2021-09-19 08:23:45",
9     "brand": "Sony",
10    "model": "Bravia-Z",
11    "warranty": 5
12  },
13 }
14
15 ]

```

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

```

83 // 11. Find the total no of products by each Brand and sort them in descending order.
84
85 use ('ProductOrder')
86 db.Product.aggregate([
87   { "$group" :
88     { "_id": {brand:"$brand"}, count:{$sum:1} }
89   },
90   { $sort: {"count":-1} }
91 ])
92
93 // 12. Find the total no of users by each role, sort them is descending order and save the results in t
94
95 use ('ProductOrder')
96 db.User.aggregate([
97   { "$group" :
98     { "_id": {role:"$role"}, count:{$sum:1} }
99   },
100   { $sort: {"count":-1} },
101   { $out : "result" }
102 ])
103
104 ]

```

```

1 {
2   "_id": {
3     "brand": "Samsung",
4     "count": 2
5   },
6 }
7
8 {
9   "_id": {
10    "brand": "LG",
11    "count": 2
12  },
13 }
14
15 {
16   "_id": {
17     "brand": "Sony",
18     "count": 2
19   },
20 }
21
22 {
23   "_id": {
24     "brand": "AppleTV",
25     "count": 1
26   },
27 }
28
29 {
30   "_id": {
31     "brand": "GoogleTV",
32     "count": 1
33   },
34 }
35
36 {
37   "_id": {
38     "brand": "Haier",
39     "count": 1
40   },
41 }
42
43 {
44   "_id": {
45     "brand": "Panasonic",
46     "count": 1
47   },
48 }
49
50 ]

```

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

```

// 12. Find the total no of users by each role, sor

use ('ProductOrder')
db.User.aggregate([
  { "$group" :
    { "_id": {role:"$role"}, count:{$sum:1} }
  },
  { $sort: {"count":-1} },
  { $out : "result" }
])

```

ProductOrder

Documents

ProductOrder.res...

My Queries

Databases

Search

ProductOrder

- Inventory
- Order
- Product
- User
- result

admin

config

ProductOrder.result

Documents

Aggregations

Schema

Explain Plan

Indexes

Validation

FilterType a query: { field: 'value' }

ADD DATAEXPORT COLLECTION

> _id: Object
count: 9

> _id: Object
count: 3