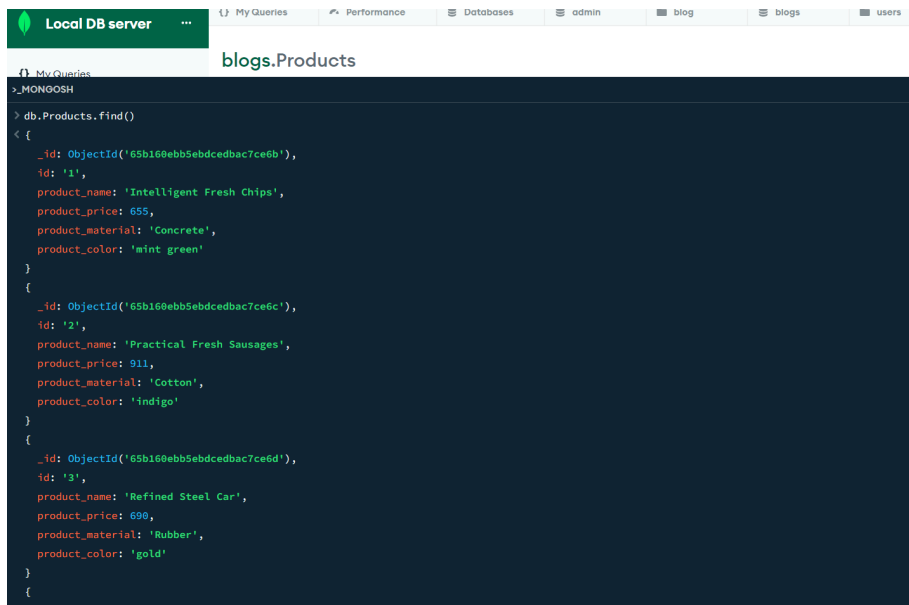


MongoDB Day-1 Task

For the following question write the corresponding MongoDB queries for the given data

1. Find all the information about each products

Ans :



The screenshot shows the MongoDB Local DB server interface. The top bar includes tabs for 'My Queries', 'Performance', 'Databases', 'admin', 'blog', 'blogs', and 'users'. The 'blogs' tab is selected, and the database 'blogs' is open. The collection 'Products' is selected. The command prompt shows the following query:

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

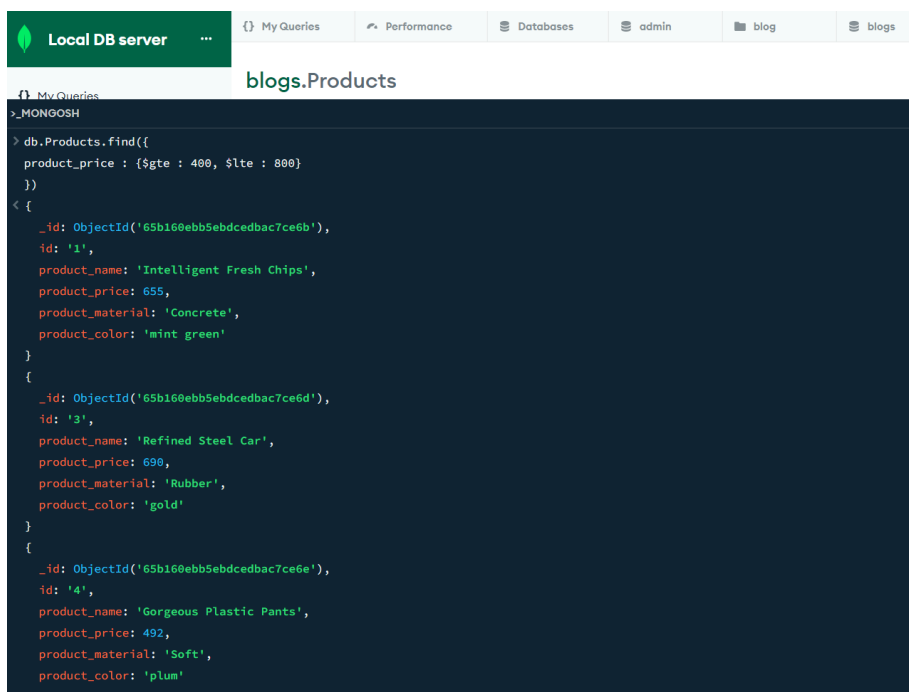
The result is a JSON array of three product documents:

```
{
  "_id": ObjectId("65b160ebb5ebdcedbac7ce6b"),
  "id": "1",
  "product_name": "Intelligent Fresh Chips",
  "product_price": 655,
  "product_material": "Concrete",
  "product_color": "mint green"
},
{
  "_id": ObjectId("65b160ebb5ebdcedbac7ce6c"),
  "id": "2",
  "product_name": "Practical Fresh Sausages",
  "product_price": 911,
  "product_material": "Cotton",
  "product_color": "indigo"
},
{
  "_id": ObjectId("65b160ebb5ebdcedbac7ce6d"),
  "id": "3",
  "product_name": "Refined Steel Car",
  "product_price": 690,
  "product_material": "Rubber",
  "product_color": "gold"
}
```

db.products.find()

2. Find the product price which are between 400 to 800

Ans :



The screenshot shows the MongoDB Local DB server interface. The top bar includes tabs for 'My Queries', 'Performance', 'Databases', 'admin', 'blog', 'blogs', and 'users'. The 'blogs' tab is selected, and the database 'blogs' is open. The collection 'Products' is selected. The command prompt shows the following query:

```
> db.Products.find({
  product_price : {$gte : 400, $lte : 800}
})
```

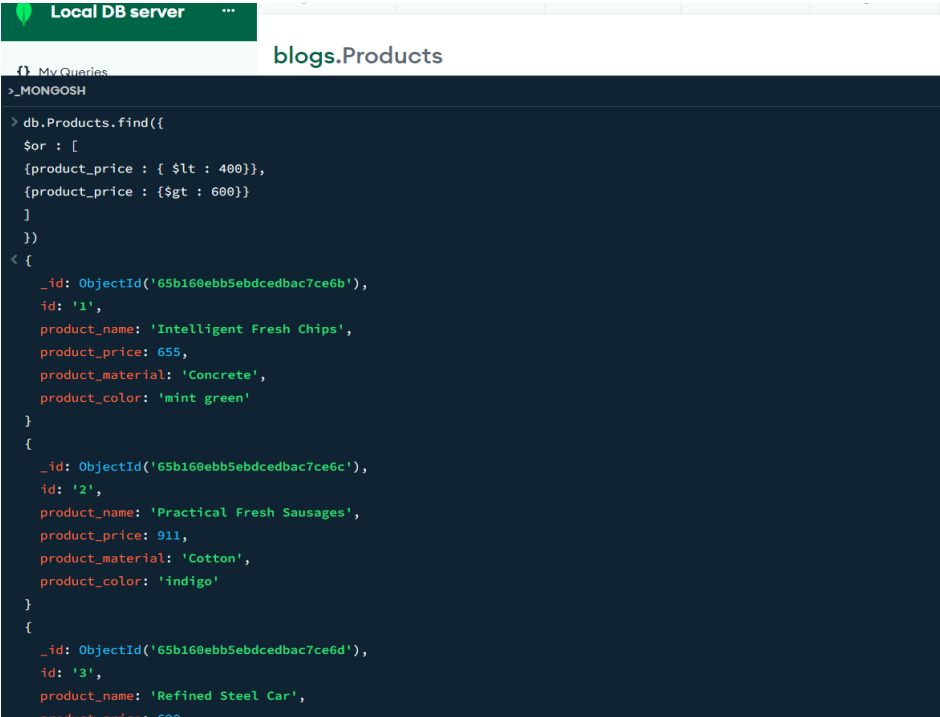
The result is a JSON array of two product documents:

```
{
  "_id": ObjectId("65b160ebb5ebdcedbac7ce6b"),
  "id": "1",
  "product_name": "Intelligent Fresh Chips",
  "product_price": 655,
  "product_material": "Concrete",
  "product_color": "mint green"
},
{
  "_id": ObjectId("65b160ebb5ebdcedbac7ce6d"),
  "id": "3",
  "product_name": "Refined Steel Car",
  "product_price": 690,
  "product_material": "Rubber",
  "product_color": "gold"
}
```

db.Products.find({ product_price : {\$gte : 400, \$lte : 800} })

3. Find the product price which are not between 400 to 600

Ans :



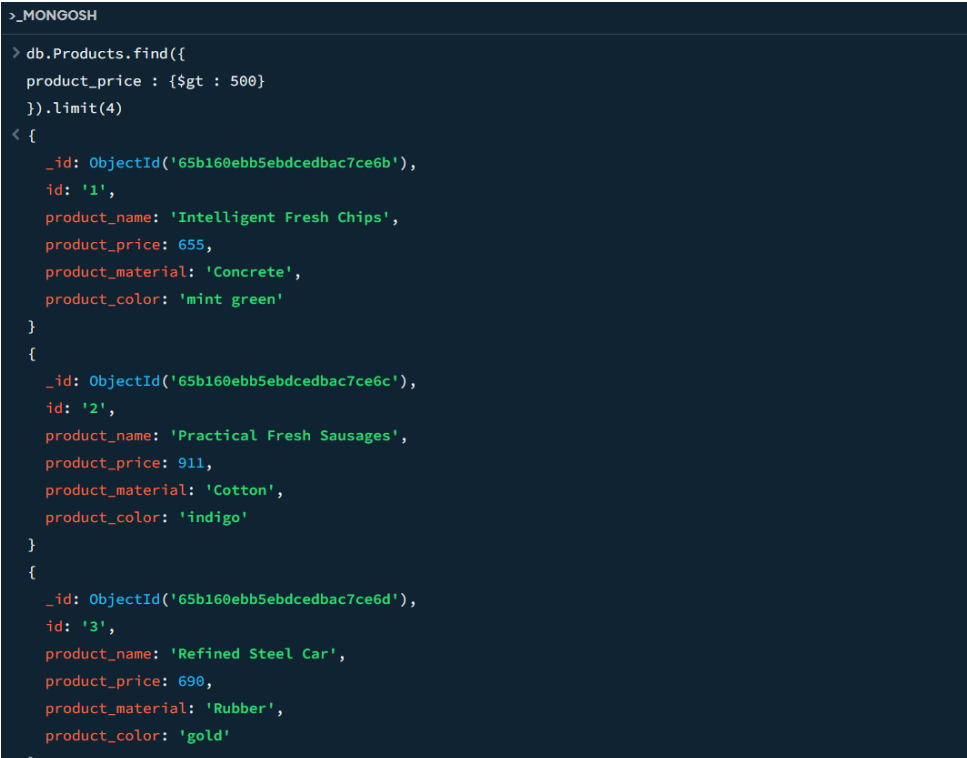
```
Local DB server
blogs.Products

>_MONGOSH
> db.Products.find({
  $or : [
    {product_price : { $lt : 400}},
    {product_price : { $gt : 600}}
  ]
})
< {
  _id: ObjectId('65b160ebb5ebdcedbac7ce6b'),
  id: '1',
  product_name: 'Intelligent Fresh Chips',
  product_price: 655,
  product_material: 'Concrete',
  product_color: 'mint green'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce6c'),
  id: '2',
  product_name: 'Practical Fresh Sausages',
  product_price: 911,
  product_material: 'Cotton',
  product_color: 'indigo'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce6d'),
  id: '3',
  product_name: 'Refined Steel Car',
  product_price: 690,
  product_material: 'Rubber',
  product_color: 'gold'
}
```

```
db.Products.find({$or : [{product_price : { $lt : 400}}, {product_price : { $gt : 600}}]})
```

4. List the four product which are grater than 500 in price

Ans :



```
>_MONGOSH
> db.Products.find({
  product_price : { $gt : 500}
}).limit(4)
< {
  _id: ObjectId('65b160ebb5ebdcedbac7ce6b'),
  id: '1',
  product_name: 'Intelligent Fresh Chips',
  product_price: 655,
  product_material: 'Concrete',
  product_color: 'mint green'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce6c'),
  id: '2',
  product_name: 'Practical Fresh Sausages',
  product_price: 911,
  product_material: 'Cotton',
  product_color: 'indigo'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce6d'),
  id: '3',
  product_name: 'Refined Steel Car',
  product_price: 690,
  product_material: 'Rubber',
  product_color: 'gold'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce6e'),
  id: '4',
  product_name: 'Rubber',
  product_price: 690,
  product_material: 'Rubber',
  product_color: 'gold'
}
```

```
db.Products.find({product_price : { $gt : 500}}).limit(4)
```

5. Find the product name and product material of each products

Ans :

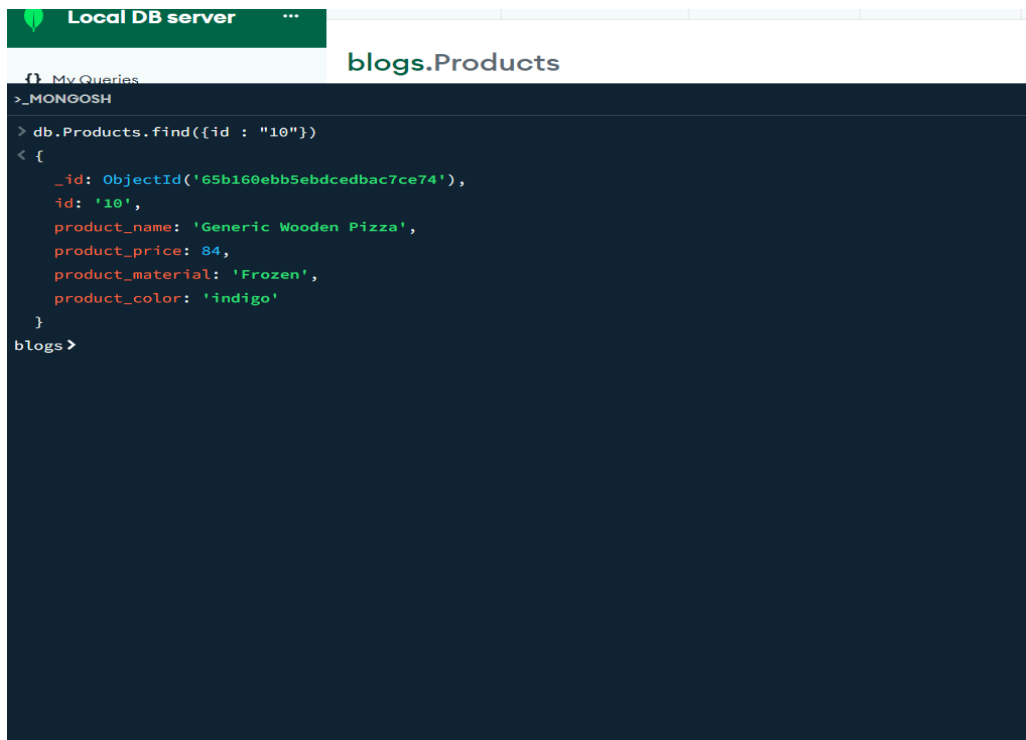
```
>_MONGOSH

> db.Products.find().projection({
  product_name : 1,
  product_material : 1
})
< {
  _id: ObjectId('65b160ebb5ebdcedbac7ce6b'),
  product_name: 'Intelligent Fresh Chips',
  product_material: 'Concrete'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce6c'),
  product_name: 'Practical Fresh Sausages',
  product_material: 'Cotton'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce6d'),
  product_name: 'Refined Steel Car',
  product_material: 'Rubber'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce6e'),
  product_name: 'Gorgeous Plastic Pants',
  product_material: 'Soft'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce6f'),
  product_name: 'Generic Fresh Fruit',
  product_material: 'Soft'
}
```

```
db.Products.find().projection({product_name : 1,product_material : 1})
```

6. Find the product with a row id of 10

Ans :



The screenshot shows the MongoDB Local DB server interface. The top bar indicates 'Local DB server' and 'blogs.Products'. Below this, the 'My Queries' tab is active, showing a query in the MongoDB shell. The query is `db.Products.find({id : "10"})`. The result is a single document with the following fields: `_id` (ObjectId), `id` (10), `product_name` (Generic Wooden Pizza), `product_price` (84), `product_material` (Frozen), and `product_color` (indigo).

```
>_MONGOSH

> db.Products.find({id : "10"})
< {
  _id: ObjectId('65b160ebb5ebdcedbac7ce74'),
  id: '10',
  product_name: 'Generic Wooden Pizza',
  product_price: 84,
  product_material: 'Frozen',
  product_color: 'indigo'
}
blogs>
```

```
db.Products.find({id : "10"})
```

7. Find only the product name and product material

Ans :

```
>_MONGOOSH

> db.Products.find().projection({
  product_name : 1,
  product_material : 1
})
< {
  _id: ObjectId('65b160ebb5ebdcedbac7ce6b'),
  product_name: 'Intelligent Fresh Chips',
  product_material: 'Concrete'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce6c'),
  product_name: 'Practical Fresh Sausages',
  product_material: 'Cotton'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce6d'),
  product_name: 'Refined Steel Car',
  product_material: 'Rubber'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce6e'),
  product_name: 'Gorgeous Plastic Pants',
  product_material: 'Soft'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce6f'),
  product_name: 'Awesome Wooden Ball',
  product_material: 'Soft'
}
```

```
db.Products.find().projection({product_name : 1,product_material : 1})
```

8. Find all products which contain the value of soft in product material

Ans :

```
blogs.Products

My Queries
>_MONGOOSH

> db.Products.find({
  product_material : "Soft"
})
< {
  _id: ObjectId('65b160ebb5ebdcedbac7ce6e'),
  id: '4',
  product_name: 'Gorgeous Plastic Pants',
  product_price: 492,
  product_material: 'Soft',
  product_color: 'plum'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce73'),
  id: '9',
  product_name: 'Awesome Wooden Ball',
  product_price: 28,
  product_material: 'Soft',
  product_color: 'azure'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce75'),
  id: '11',
  product_name: 'Unbranded Wooden Cheese',
  product_price: 26,
  product_material: 'Soft',
  product_color: 'black'
}
```

```
db.Products.find({product_material : "Soft"})
```

9. Find products which contain product color indigo and product price 492.00

Ans :

```
blogs.Products
My Queries
>_MONGOSH
> db.Products.find( { $or: [ { product_color: 'indigo' }, { product_price: 492 } ] } )
< {
  _id: ObjectId('65b160ebb5ebdcedbac7ce6c'),
  id: '2',
  product_name: 'Practical Fresh Sausages',
  product_price: 911,
  product_material: 'Cotton',
  product_color: 'indigo'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce6e'),
  id: '4',
  product_name: 'Gorgeous Plastic Pants',
  product_price: 492,
  product_material: 'Soft',
  product_color: 'plum'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce74'),
  id: '10',
  product_name: 'Generic Wooden Pizza',
  product_price: 84,
  product_material: 'Frozen',
  product_color: 'indigo'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce75'),
  id: '11',
  product_name: 'Intelligent Fresh Chips',
  product_price: 655,
  product_material: 'Concrete',
  product_color: 'mint green'
}
```

```
db.Products.find( { $or: [ { product_color: 'indigo' }, { product_price: 492 } ] } )
```

10. Delete the products which product price value are same

Ans :

```
>_MONGOSH
> var distinct = []
  var duplicate = []

  db.Products.find().forEach((data) => {
    if(distinct.indexOf(data.product_price) < 0){
      distinct.push(data.product_price)
    }else{
      duplicate.push(data._id)
    }
  })
  duplicate.forEach((id) => {
    db.Products.findOneAndDelete({
      "_id" : id
    })
  })
> db.Products.find()
< {
  _id: ObjectId('65b160ebb5ebdcedbac7ce6b'),
  id: '1',
  product_name: 'Intelligent Fresh Chips',
  product_price: 655,
  product_material: 'Concrete',
  product_color: 'mint green'
}
{
  _id: ObjectId('65b160ebb5ebdcedbac7ce6c'),
  id: '2',
  product_name: 'Practical Fresh Sausages',
  product_price: 911,
  product_material: 'Cotton',
  product_color: 'indigo'
}
```

```
var distinct = []
```

```
var duplicate = []
```

```
db.Products.find().forEach((data) => {
```

```
  if(distinct.indexOf(data.product_price) < 0){
```

```
        distinct.push(data.product_price)
    }else{
        duplicate.push(data._id)
    }
})
duplicate.forEach((id) => {
    db.Products.findOneAndDelete({
        "_id" : id
    })
})
})
```

XXXXXXXXXXXXXXXXXXXX
