MongoDB Day 1

For the following question write the corresponding MongoDB queries

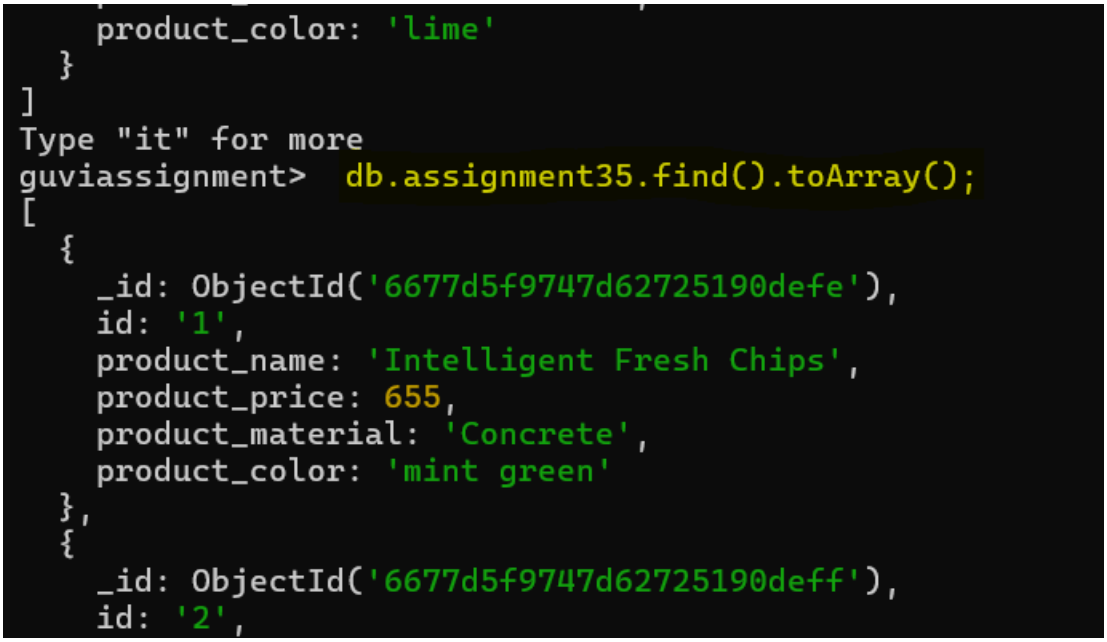
DB: guviassignment
Collection: assignment35

1. Find all the information about each products

Answer: Using just db.collection.find() will stop the iteration at 20 documents by default and to further display other documents the recommendation would be using “it”. So to avoid using “it”, I have used db.collection.find().toArray() which will return all available documents(in this case, we have a total of 25 documents and all 25 docs will be displayed without any interruption).

FYI: I tried using db.collection.find().pretty() but it still defaulted to return only 20 documents. So I have used toArray().

Query: **db.assignment35.find().toArray();**



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

Answer: With the help of ‘gte’(greater than equal) and ‘lte’(less than equal) I was able to pull the results.

Query: **db.assignment35.find({product_price: {$gte: 400, $lte: 800}})**

```
  }
]
guviassignment> db.assignment35.find({product_price: {$gte: 400, $lte: 800}} )
[
  {
    _id: ObjectId('6677d5f9747d62725190defe'),
    id: '1',
    product_name: 'Intelligent Fresh Chips',
    product_price: 655,
    product_material: 'Concrete',
    product_color: 'mint green'
  },
  {
    _id: ObjectId('6677d5f9747d62725190df00'),
    id: '3',
    product_name: 'Refined Steel Car',
    product_price: 690,
    product_material: 'Rubber',
    product_color: 'gold'
  },
```

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

   Answer: $not will help one to eliminate a set of documents with a set of condition.

   Query: **db.assignment35.find({product_price: {$not: {$gte: 400, $lte: 600}}}).toArray()**

```
    product_color: 'azure'
  }
]
Type "it" for more
guviassignment> db.assignment35.find({product_price: {$not: {$gte: 400, $lte: 600}}}).toArray()
[
  {
    _id: ObjectId('6677d5f9747d62725190defe'),
    id: '1',
    product_name: 'Intelligent Fresh Chips',
    product_price: 655,
    product_material: 'Concrete',
    product_color: 'mint green'
  },
  {
    _id: ObjectId('6677d5f9747d62725190deff'),
    id: '2',
    product_name: 'Practical Fresh Sausages',
    product_price: 911,
    product_material: 'Cotton',
    product_color: 'indigo'
  },
```

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

   Answer: With the help of limit() we can list a limited number of documents.

   Query: **db.assignment35.find({product_price: {$gte: 500}}).limit(4);**

```
]
guviassignment> db.assignment35.find({product_price: {$gte: 500}}).limit(4);
[
  {
    _id: ObjectId('6677d5f9747d62725190defe'),
    id: '1',
    product_name: 'Intelligent Fresh Chips',
    product_price: 655,
    product_material: 'Concrete',
    product_color: 'mint green'
  },
  {
```

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

   Answer, Query:

   db.assignment35.find({},{product_name:1, product_material: 1}).toArray();

```
]
guviassignment> db.assignment35.find({},{product_name:1, product_material: 1}).toArray();
[
  {
    _id: ObjectId('6677d5f9747d62725190defe'),
    product_name: 'Intelligent Fresh Chips',
    product_material: 'Concrete'
  },
  {
    _id: ObjectId('6677d5f9747d62725190deff'),
    product_name: 'Practical Fresh Sausages',
    product_material: 'Cotton'
```

6. Find the product with a row id of 10

Answer: In this given example, we got a field with appropriate id(string). To find a particular string make use of the following query and pull the document.

Query: **db.assignment35.find({id: "10"});**

```
Type "it" for more
guviassignment> db.assignment35.find({id: "10"});
[
  {
    _id: ObjectId('6677d5f9747d62725190df07'),
    id: '10',
    product_name: 'Generic Wooden Pizza',
    product_price: 84,
    product_material: 'Frozen',
    product_color: 'indigo'
  }
]
```

7. Find only the product name and product material

Answer: to eliminate the default inclusion “_id”, set it to 0, so only those expected fields can be displayed.

Query: **db.assignment35.find({}, {product_name:1, product_material: 1, _id:0}).toArray();**

```
Type "it" for more
guviassignment> db.assignment35.find({}, {product_name:1, product_material: 1, _id:0}).toArray();
[
  {
    product_name: 'Intelligent Fresh Chips',
    product_material: 'Concrete'
  },
  {
    product_name: 'Practical Fresh Sausages',
    product_material: 'Cotton'
  },
  { product_name: 'Refined Steel Car', product_material: 'Rubber' },
  { product_name: 'Gorgeous Plastic Pants', product_material: 'Soft' },
  { product_name: 'Sleek Cotton Chair', product_material: 'Fresh' },
```

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

Answer, Query: **db.assignment35.find({product_material: "Soft"}).toArray();**

```
guviassignment> db.assignment35.find({product_material: "Soft"}).toArray();
[
  {
    _id: ObjectId('6677d5f9747d62725190df01'),
    id: '4',
    product_name: 'Gorgeous Plastic Pants',
    product_price: 492,
    product_material: 'Soft',
    product_color: 'plum'
  },
  {
```

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

Answer: Make use of or to help add different conditions.

Query: **db.assignment35.find({$or:[{product_color: "indigo"}, {product_price: 492}]}).toArray();**

```
[]
guviassignment> db.assignment35.find({$or:[{product_color: "indigo"}, {product_price: 492}]}).toArray();
[
  {
    _id: ObjectId('6677d5f9747d62725190deff'),
    id: '2',
    product_name: 'Practical Fresh Sausages',
    product_price: 911,
    product_material: 'Cotton',
    product_color: 'indigo'
  },
  {
    _id: ObjectId('6677d5f9747d62725190df01'),
```

10. Delete the products which product price value are 28

Answer: Since there is no duplicates, we are good with deleteOne, but there are situations in which we need to handle duplicates, and if we have duplicates then make use of aggregate to fing the duplicates and then get rid of the duplicates.

Query: **db.assignment35.deleteOne({product_price: 28});**