MONGODB ASSESSMENT 2

NAME: KOUSIK C

REG.NO:21BDS0039

QS:

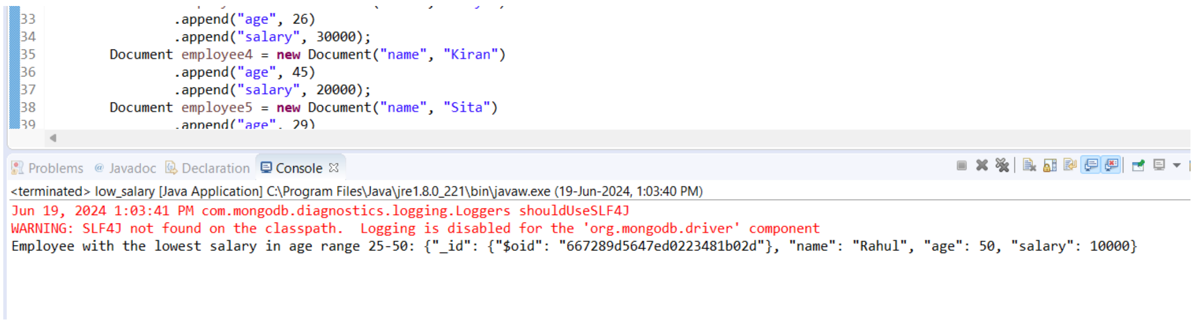
Create one database with any name and collection called employee having records name, salary and age. The salaries of each individual should be between 10000-40000 and you have to find the employee with lowest salary in age range of 25-50.

Java Code:

```
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Filters;
import com.mongodb.client.model.Sorts;
import org.bson.Document;


import java.util.Arrays;


public class low_salary {
    public static void main(String[] args) {
        // Creating a Mongo client
        MongoClient mongoClient = MongoClients.create("mongodb://localhost:27017");


        // Accessing the database
        MongoDatabase database = mongoClient.getDatabase("CompanyDB");

        // Retrieving a collection
        MongoCollection<Document> collection = database.getCollection("Employee");

        // Dropping the collection if it exists to start fresh
        collection.drop();

        // Inserting documents
        Document employee1 = new Document("name", "Amit")
                .append("age", 30)
                .append("salary", 25000);
        Document employee2 = new Document("name", "Raj")
                .append("age", 28)
                .append("salary", 15000);
        Document employee3 = new Document("name", "Priya")
                .append("age", 26)
                .append("salary", 30000);
        Document employee4 = new Document("name", "Kiran")
                .append("age", 45)
                .append("salary", 20000);
        Document employee5 = new Document("name", "Sita")
                .append("age", 29)
                .append("salary", 35000);
        Document employee6 = new Document("name", "Rahul")
                .append("age", 50)
                .append("salary", 10000);

        collection.insertMany(Arrays.asList(employee1, employee2, employee3, employee4,
employee5, employee6));
```

```
        // Finding the employee with the lowest salary in the age range 25-50
        Document lowestSalaryEmployee = collection.find(Filters.and(
                Filters.gte("age", 25),
                Filters.lte("age", 50)))
                .sort(Sorts.ascending("salary"))
                .first();

        if (lowestSalaryEmployee != null) {
            System.out.println("Employee with the lowest salary in age range 25-50: " + 
lowestSalaryEmployee.toJson());
        } else {
            System.out.println("No employee found in the specified age range");
        }

        // Closing the client
        mongoClient.close();
    }
}
```

OUTPUT:



```
                .append("age", 26)
                .append("salary", 30000);
        Document employee4 = new Document("name", "Kiran")
                .append("age", 45)
                .append("salary", 20000);
        Document employee5 = new Document("name", "Sita")
                .append("age", 29)
```

```
<terminated> low_salary [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw.exe (19-Jun-2024, 1:03:40 PM)
Jun 19, 2024 1:03:41 PM com.mongodb.diagnostics.logging.Loggers shouldUseSLF4J
WARNING: SLF4J not found on the classpath.  Logging is disabled for the 'org.mongodb.driver' component
Employee with the lowest salary in age range 25-50: {"_id": {"$oid": "667289d5647ed0223481b02d"}, "name": "Rahul", "age": 50, "salary": 10000}
```

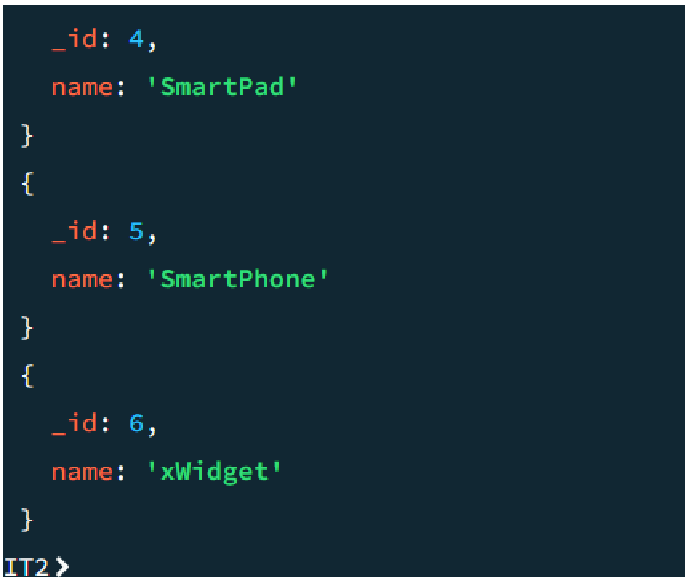
QS1:

Find the names for which the price is less than 799 or has storage of 1024

Code:

```
db.products1.find({
   $or: [
      { price: { $lt: 799 } },
      { storage: 1024 }
   ]
}, {
   name: 1
})
```

Output:



```
   _id: 4,
   name: 'SmartPad'
 }
 {
   _id: 5,
   name: 'SmartPhone'
 }
 {
   _id: 6,
   name: 'xWidget'
 }
IT2>
```

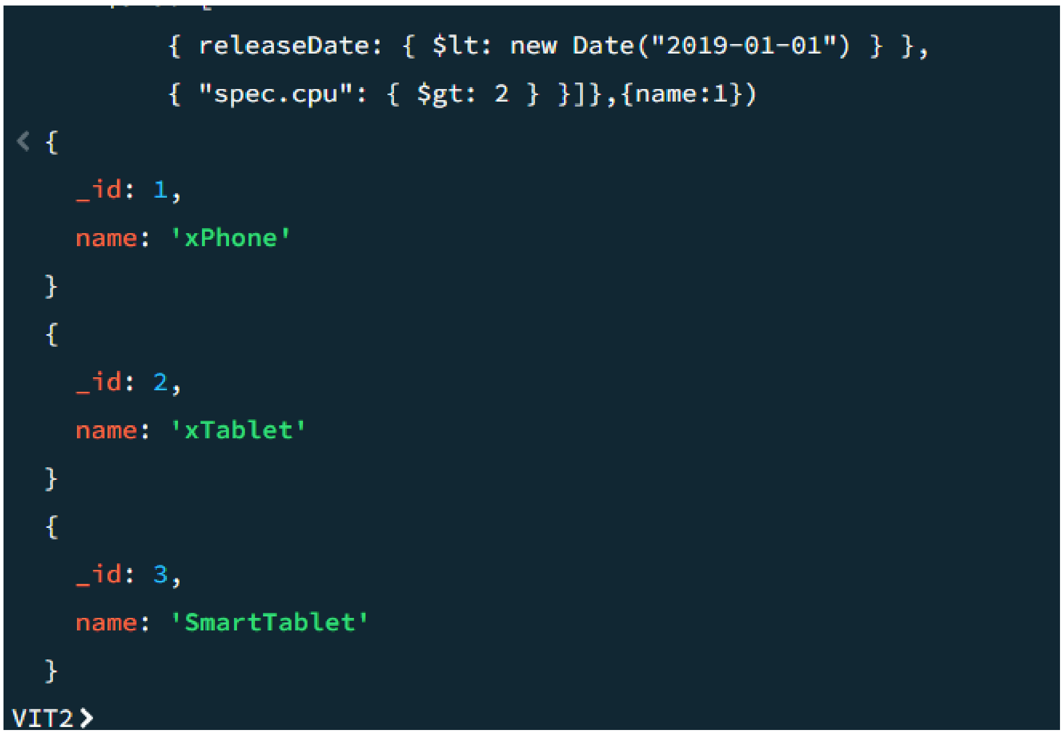
QS2:

Find the products that were released before 2019 and have a CPU greater than 2

Code:

```
db.Products1.find({
  $and: [
    { releaseDate: { $lt: new Date("2019-01-01") } },
    { "spec.cpu": { $gt: 2 } }
  ]
})
```

Output:

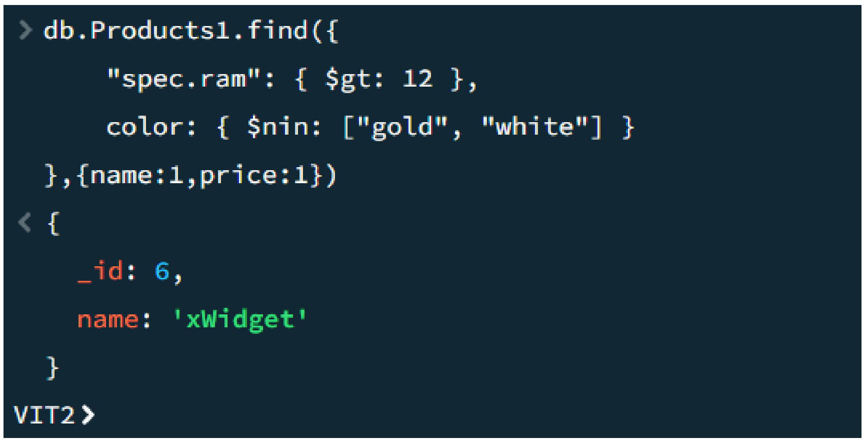


```
          { releaseDate: { $lt: new Date("2019-01-01") } },
          { "spec.cpu": { $gt: 2 } }]},{name:1})
< {
    _id: 1,
    name: 'xPhone'
  }
  {
    _id: 2,
    name: 'xTablet'
  }
  {
    _id: 3,
    name: 'SmartTablet'
  }
VIT2>
```

QS3:

Find the products which are not gold or white and has ram of more than 12

Code and Output:

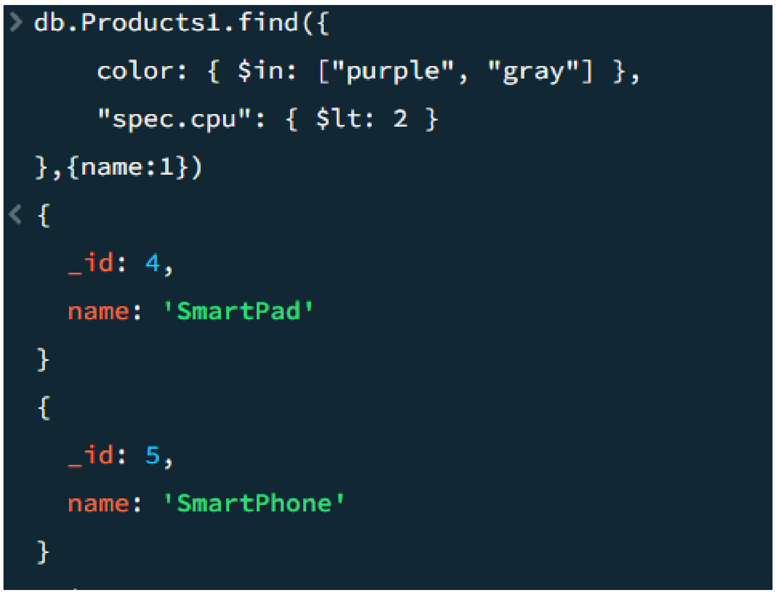


```
> db.Products1.find({
      "spec.ram": { $gt: 12 },
      color: { $nin: ["gold", "white"] }
  },{name:1,price:1})
< {
    _id: 6,
    name: 'xWidget'
  }
VIT2>
```

QS4:

Find products that are available in purple or gray and have a CPU less than 2

Code and Output:

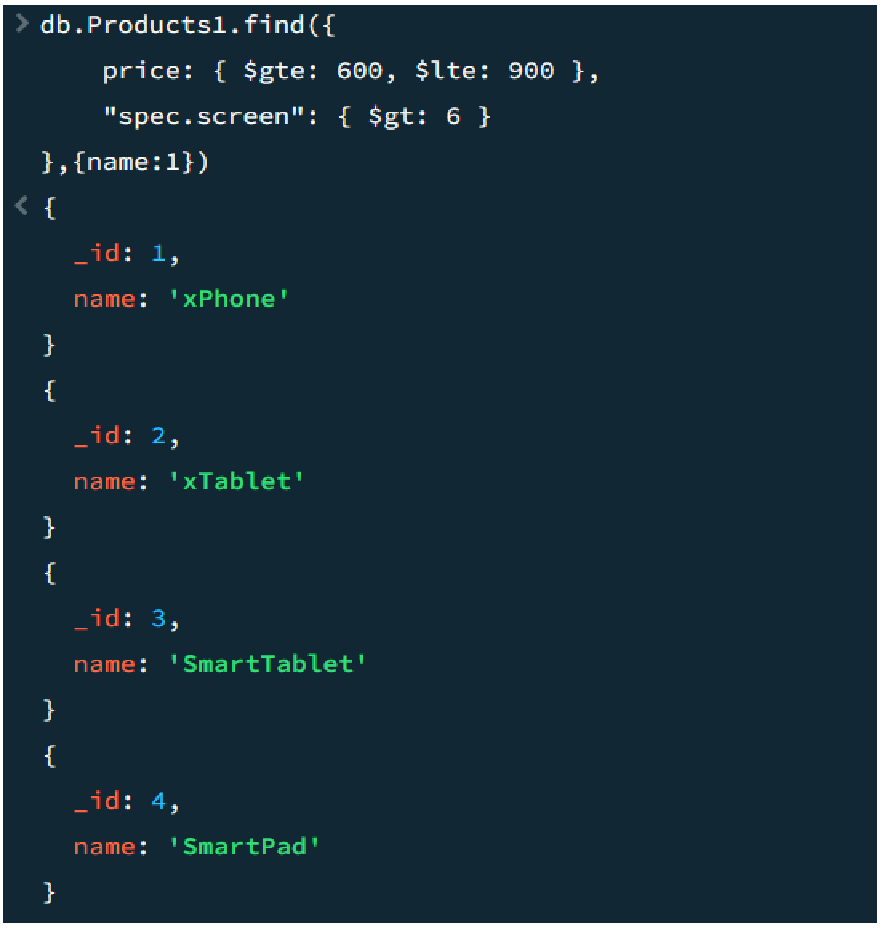


```
> db.Products1.find({
      color: { $in: ["purple", "gray"] },
      "spec.cpu": { $lt: 2 }
  },{name:1})
< {
    _id: 4,
    name: 'SmartPad'
  }
  {
    _id: 5,
    name: 'SmartPhone'
  }
```

QS5:

Find products with a price between 600 and 900 and a screen size greater than 6 inches

Code and Output:

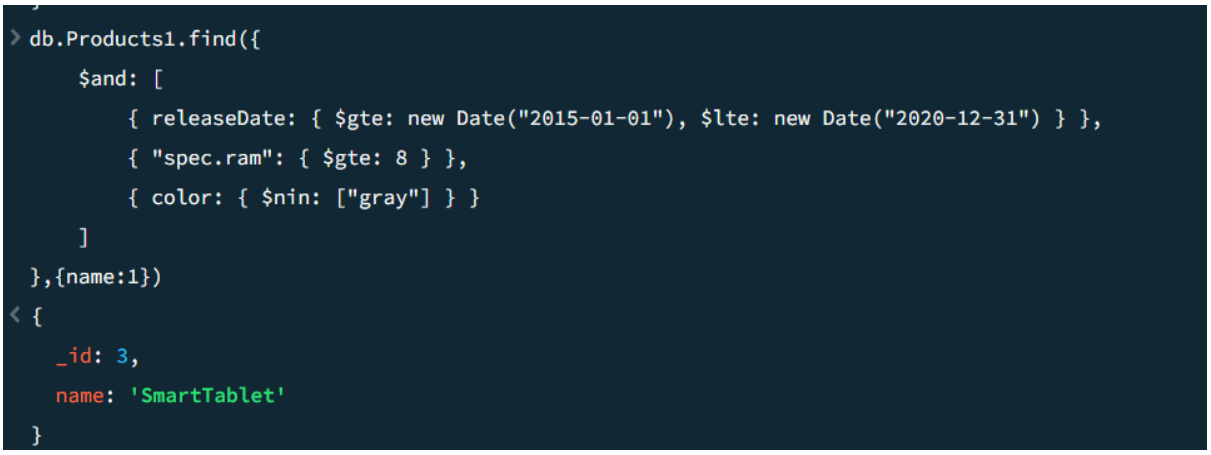


```
> db.Products1.find({
      price: { $gte: 600, $lte: 900 },
      "spec.screen": { $gt: 6 }
  },{name:1})
< {
    _id: 1,
    name: 'xPhone'
  }
  {
    _id: 2,
    name: 'xTablet'
  }
  {
    _id: 3,
    name: 'SmartTablet'
  }
  {
    _id: 4,
    name: 'SmartPad'
  }
```

QS6:

Find products released between 2015 and 2020 that have a RAM of greater than or equal to 8GB and are available in white color

Code and Output:

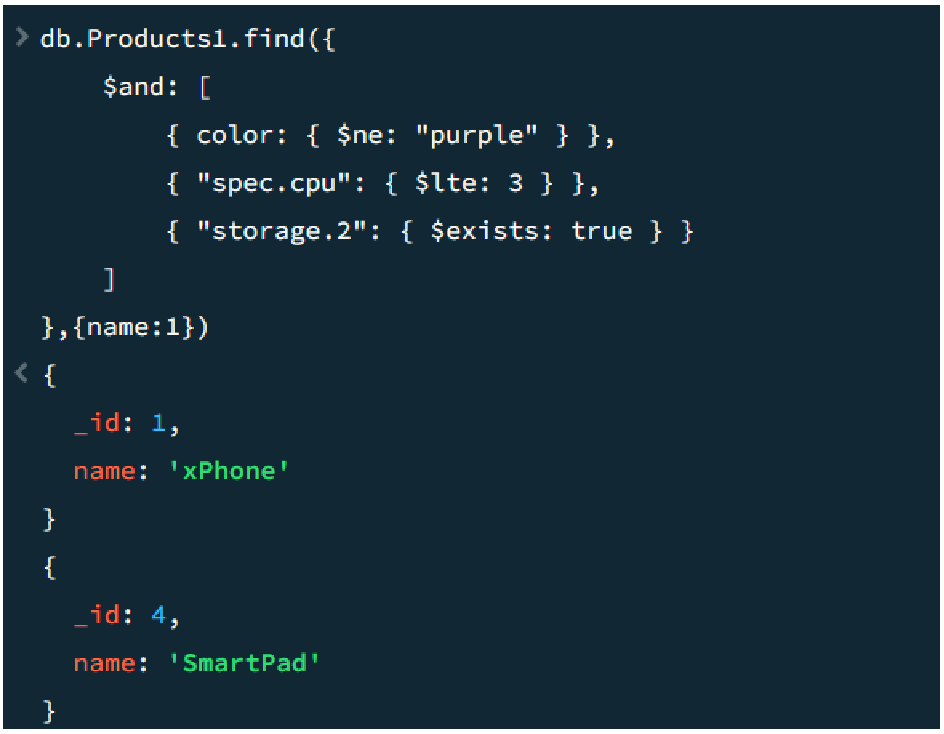


```
> db.Products1.find({
    $and: [
        { releaseDate: { $gte: new Date("2015-01-01"), $lte: new Date("2020-12-31") } },
        { "spec.ram": { $gte: 8 } },
        { color: { $nin: ["gray"] } }
    ]
  },{name:1})
< {
    _id: 3,
    name: 'SmartTablet'
  }
```

QS7:

Find products that are not available in purple, have a CPU less than or equal to 3, and more than two storage options

Code and Output:

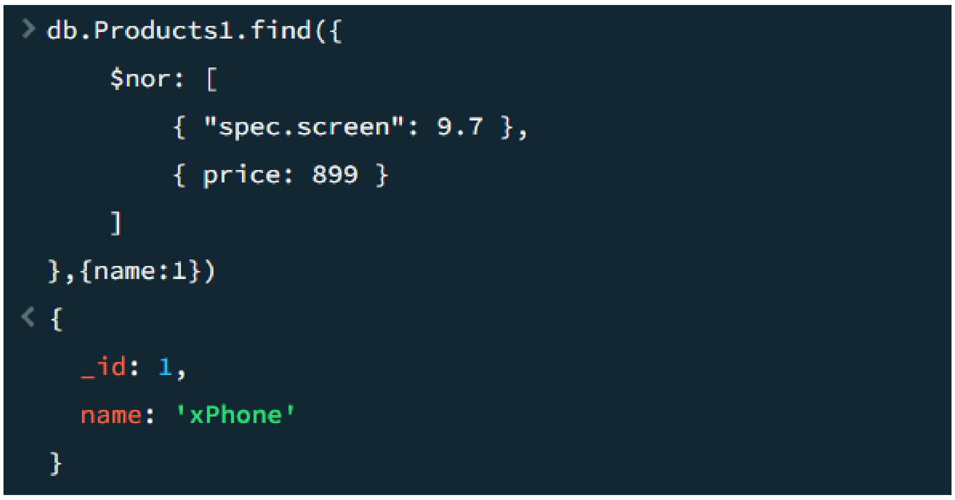


```
> db.Products1.find({
    $and: [
        { color: { $ne: "purple" } },
        { "spec.cpu": { $lte: 3 } },
        { "storage.2": { $exists: true } }
    ]
  },{name:1})
< {
    _id: 1,
    name: 'xPhone'
  }
  {
    _id: 4,
    name: 'SmartPad'
  }
```

QS8:

Find products that do not have a screen size of 9.7 inches or do not have a price of 899

Code and Output:

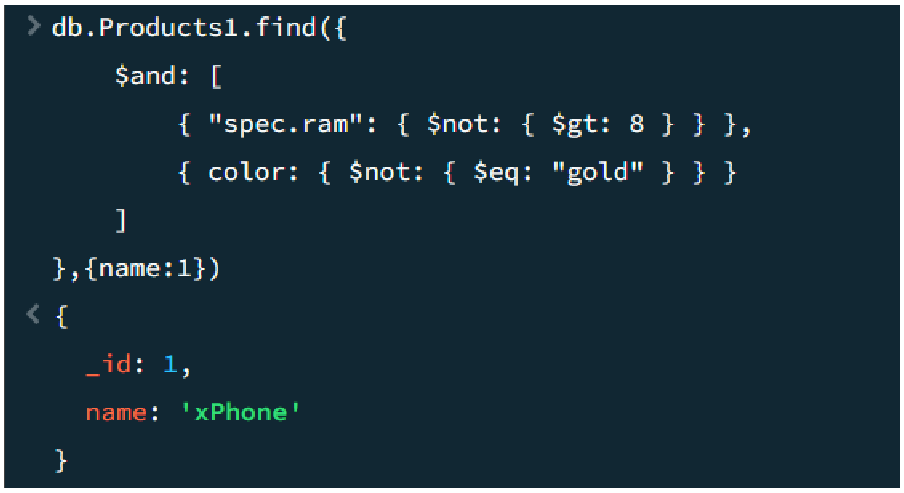


```
> db.Products1.find({
      $nor: [
          { "spec.screen": 9.7 },
          { price: 899 }
      ]
  },{name:1})
< {
    _id: 1,
    name: 'xPhone'
  }
```

QS9:

Find products that do not have a RAM greater than 8GB and are not available in gold

Code and Output:

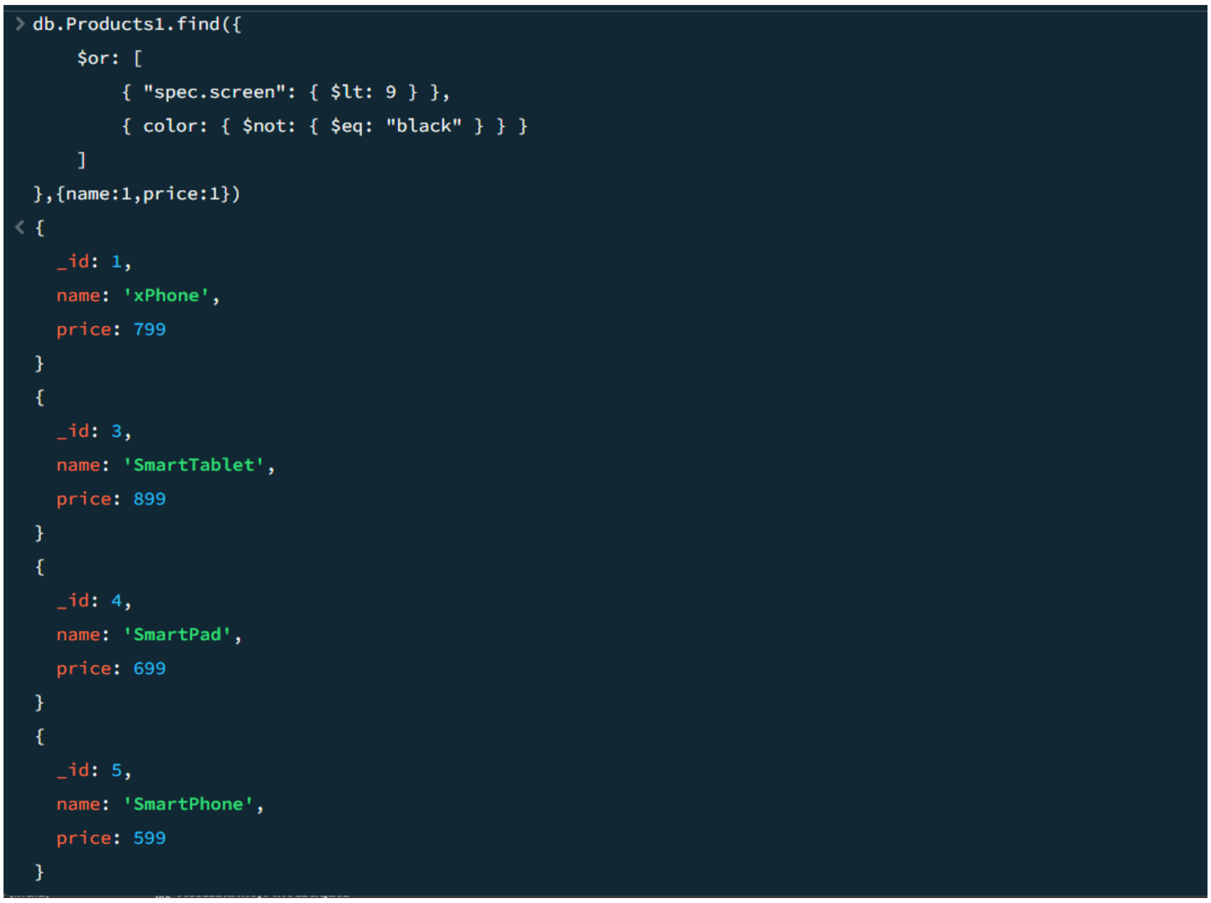


```
> db.Products1.find({
      $and: [
          { "spec.ram": { $not: { $gt: 8 } } },
          { color: { $not: { $eq: "gold" } } }
      ]
  },{name:1})
< {
    _id: 1,
    name: 'xPhone'
  }
```

QS10:

Find products that either have a screen size less than 9 inches or are not available in black.

Code and Output:

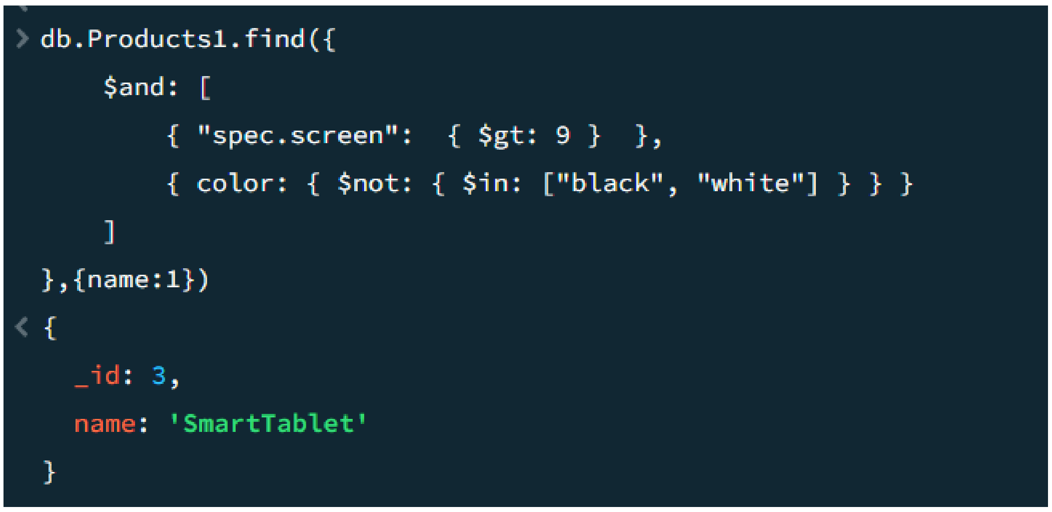


```
> db.Products1.find({
      $or: [
          { "spec.screen": { $lt: 9 } },
          { color: { $not: { $eq: "black" } } }
      ]
  },{name:1,price:1})
< {
    _id: 1,
    name: 'xPhone',
    price: 799
  }
  {
    _id: 3,
    name: 'SmartTablet',
    price: 899
  }
  {
    _id: 4,
    name: 'SmartPad',
    price: 699
  }
  {
    _id: 5,
    name: 'SmartPhone',
    price: 599
  }
```

QS11:

Find products that do not have a screen size greater than 9 inches and are not available in black or white.

Code and Output:

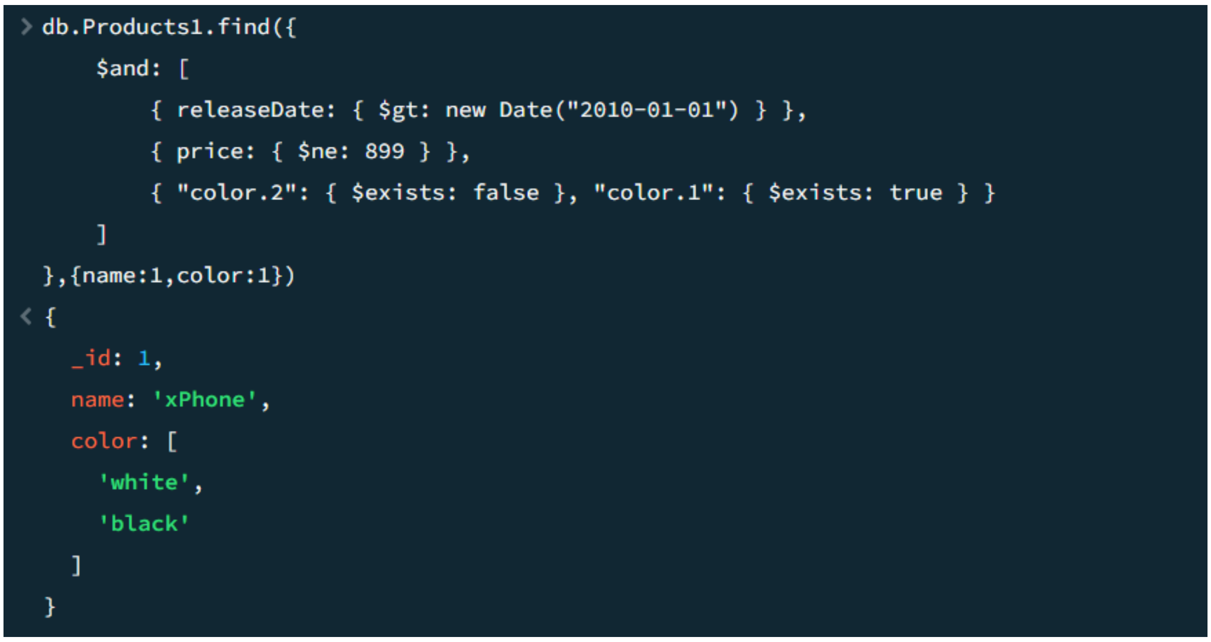


```
> db.Products1.find({
      $and: [
          { "spec.screen":  { $gt: 9 }  },
          { color: { $not: { $in: ["black", "white"] } } }
      ]
  },{name:1})
< {
    _id: 3,
    name: 'SmartTablet'
  }
```

QS12:

Find products that were released after January 1, 2010, do not have a price of 899, and have exactly two color options.

Code and Output:



```
> db.Products1.find({
      $and: [
          { releaseDate: { $gt: new Date("2010-01-01") } },
          { price: { $ne: 899 } },
          { "color.2": { $exists: false }, "color.1": { $exists: true } }
      ]
  },{name:1,color:1})
< {
    _id: 1,
    name: 'xPhone',
    color: [
      'white',
      'black'
    ]
  }
```