Program No:	**3**
Roll No :	**1545**
Title of Program :	**MongoDB advance operations**
Objective :	**Sorting, Projection, View, Join, Index**

1. Sorting documents

sort according to name

```
> db.products.aggregate(
... [
... { $sort: {"name": 1}}
... ]);
{ "_id" : 101, "name" : "Paper", "price" : 2000, "qty" : 20, "rating" : 0.5 }
{ "_id" : 102, "name" : "Paper", "brand" : "Dunder Mifflin", "price" : 1000, "rating" : 5, "qty" : 100 }
{ "_id" : 108, "name" : "Phone", "brand" : "One Plus", "price" : 20000, "rating" : 5, "qty" : 500 }
{ "_id" : 103, "name" : "Printer", "brand" : "Dunder Mifflin", "price" : 5000, "rating" : 4, "qty" : 10 }
{ "_id" : 104, "name" : "Printer", "brand" : "Dunder Mifflin", "price" : 2000, "rating" : 3, "qty" : 20 }
{ "_id" : 111, "brand" : "Lee Cooper", "name" : "Shoes", "rating" : 0.5 }
{ "_id" : 105, "name" : "Staple", "brand" : "Dunder Mifflin", "price" : 500, "rating" : 3.5, "qty" : 50 }
{ "_id" : 106, "name" : "T-shirt", "brand" : "USPA", "price" : 1000, "rating" : 4.5, "qty" : 10 }
{ "_id" : 107, "name" : "T-shirt", "brand" : "Peter England", "price" : 1200, "rating" : 5, "qty" : 100 }
```

sort according to brand

```
> db.products.aggregate( [ { $sort: {"brand": 1}} ]);
{ "_id" : 101, "name" : "Paper", "price" : 2000, "qty" : 20, "rating" : 0.5 }
{ "_id" : 102, "name" : "Paper", "brand" : "Dunder Mifflin", "price" : 1000, "rating" : 5, "qty" : 100 }
{ "_id" : 103, "name" : "Printer", "brand" : "Dunder Mifflin", "price" : 5000, "rating" : 4, "qty" : 10 }
{ "_id" : 104, "name" : "Printer", "brand" : "Dunder Mifflin", "price" : 2000, "rating" : 3, "qty" : 20 }
{ "_id" : 105, "name" : "Staple", "brand" : "Dunder Mifflin", "price" : 500, "rating" : 3.5, "qty" : 50 }
{ "_id" : 111, "brand" : "Lee Cooper", "name" : "Shoes", "rating" : 0.5 }
{ "_id" : 108, "name" : "Phone", "brand" : "One Plus", "price" : 20000, "rating" : 5, "qty" : 500 }
{ "_id" : 107, "name" : "T-shirt", "brand" : "Peter England", "price" : 1200, "rating" : 5, "qty" : 100 }
{ "_id" : 106, "name" : "T-shirt", "brand" : "USPA", "price" : 1000, "rating" : 4.5, "qty" : 10 }
>
```

sort according to brand in descending order

```
> db.products.aggregate( [ { $sort: {"brand": -1}} ]);
{ "_id" : 106, "name" : "T-shirt", "brand" : "USPA", "price" : 1000, "rating" : 4.5, "qty" : 10 }
{ "_id" : 107, "name" : "T-shirt", "brand" : "Peter England", "price" : 1200, "rating" : 5, "qty" : 100 }
{ "_id" : 108, "name" : "Phone", "brand" : "One Plus", "price" : 20000, "rating" : 5, "qty" : 500 }
{ "_id" : 111, "brand" : "Lee Cooper", "name" : "Shoes", "rating" : 0.5 }
{ "_id" : 102, "name" : "Paper", "brand" : "Dunder Mifflin", "price" : 1000, "rating" : 5, "qty" : 100 }
{ "_id" : 103, "name" : "Printer", "brand" : "Dunder Mifflin", "price" : 5000, "rating" : 4, "qty" : 10 }
{ "_id" : 104, "name" : "Printer", "brand" : "Dunder Mifflin", "price" : 2000, "rating" : 3, "qty" : 20 }
{ "_id" : 105, "name" : "Staple", "brand" : "Dunder Mifflin", "price" : 500, "rating" : 3.5, "qty" : 50 }
{ "_id" : 101, "name" : "Paper", "price" : 2000, "qty" : 20, "rating" : 0.5 }
```

sorting on name and brand

```
> db.products.aggregate(
... [
... { $sort: {"name": 1, "brand": 1}}
... ]);
{ "_id" : 101, "name" : "Paper", "price" : 2000, "qty" : 20, "rating" : 0.5 }
{ "_id" : 102, "name" : "Paper", "brand" : "Dunder Mifflin", "price" : 1000, "rating" : 5, "qty" : 100 }
{ "_id" : 108, "name" : "Phone", "brand" : "One Plus", "price" : 20000, "rating" : 5, "qty" : 500 }
{ "_id" : 103, "name" : "Printer", "brand" : "Dunder Mifflin", "price" : 5000, "rating" : 4, "qty" : 10 }
{ "_id" : 104, "name" : "Printer", "brand" : "Dunder Mifflin", "price" : 2000, "rating" : 3, "qty" : 20 }
{ "_id" : 111, "brand" : "Lee Cooper", "name" : "Shoes", "rating" : 0.5 }
{ "_id" : 105, "name" : "Staple", "brand" : "Dunder Mifflin", "price" : 500, "rating" : 3.5, "qty" : 50 }
{ "_id" : 107, "name" : "T-shirt", "brand" : "Peter England", "price" : 1200, "rating" : 5, "qty" : 100 }
{ "_id" : 106, "name" : "T-shirt", "brand" : "USPA", "price" : 1000, "rating" : 4.5, "qty" : 10 }
>
```

find product whose quantity is greater than 15 and sort them in descending order of quantity

```
> db.products.aggregate(
... [
... { $match: {"qty": {$gt: 15}}},
... { $sort: {"qty": -1}}
... ]);
{ "_id" : 108, "name" : "Phone", "brand" : "One Plus", "price" : 20000, "rating" : 5, "qty" : 500 }
{ "_id" : 102, "name" : "Paper", "brand" : "Dunder Mifflin", "price" : 1000, "rating" : 5, "qty" : 100 }
{ "_id" : 107, "name" : "T-shirt", "brand" : "Peter England", "price" : 1200, "rating" : 5, "qty" : 100 }
{ "_id" : 105, "name" : "Staple", "brand" : "Dunder Mifflin", "price" : 500, "rating" : 3.5, "qty" : 50 }
{ "_id" : 101, "name" : "Paper", "price" : 2000, "qty" : 20, "rating" : 0.5 }
{ "_id" : 104, "name" : "Printer", "brand" : "Dunder Mifflin", "price" : 2000, "rating" : 3, "qty" : 20 }
```

2. Projection

The projection operation allows you to select only specific fields we want to retrieve from a document rather than fetching the entire document

Write a query to display the name of product along with its quantity

```
> db.products.find(
... {},
... { name: 1, qty: 1 }
... );
{ "_id" : 101, "name" : "Paper", "qty" : 20 }
{ "_id" : 102, "name" : "Paper", "qty" : 100 }
{ "_id" : 103, "name" : "Printer", "qty" : 10 }
{ "_id" : 104, "name" : "Printer", "qty" : 20 }
{ "_id" : 105, "name" : "Staple", "qty" : 50 }
{ "_id" : 106, "name" : "T-shirt", "qty" : 10 }
{ "_id" : 107, "name" : "T-shirt", "qty" : 100 }
{ "_id" : 108, "name" : "Phone", "qty" : 500 }
{ "_id" : 111, "name" : "Shoes" }
```

```
> db.products.find( {}, { name: 1, qty: 1, _id: 0 } );
{ "name" : "Paper", "qty" : 20 }
{ "name" : "Paper", "qty" : 100 }
{ "name" : "Printer", "qty" : 10 }
{ "name" : "Printer", "qty" : 20 }
{ "name" : "Staple", "qty" : 50 }
{ "name" : "T-shirt", "qty" : 10 }
{ "name" : "T-shirt", "qty" : 100 }
{ "name" : "Phone", "qty" : 500 }
{ "name" : "Shoes" }
```

Display name and brand of product whose quantity is greater than 25

```
> db.products.find( { qty: {$gt: 25}}, { name: 1, brand: 1, _id: 0} );
{ "name" : "Paper", "brand" : "Dunder Mifflin" }
{ "name" : "Staple", "brand" : "Dunder Mifflin" }
{ "name" : "T-shirt", "brand" : "Peter England" }
{ "name" : "Phone", "brand" : "One Plus" }
>
```

3. Views

A view is a read-only variable object whose contents are defined by an aggregation pipeline. The mongodb view does not persist on the disk. A view content is computed on demand when a client queries the view

i. Create a Teacher collection with name, age, salary and years of experience

Create a view of teacher whose age is less than 30 years

```
> db.createView(
... "youngTeachers",
... "Teacher",
... [
... { $match: { age: {$lt: 30} } }
... ]);
{ "ok" : 1 }
> db.youngTeachers.find();
{ "_id" : ObjectId("687de0cec83c783b9464e86a"), "name" : "Amit", "age" : 29, "salary" : 47000, "exp" : 6 }
>
```

Create a view to display all the products from brand dunder mifflin

```
> db.createView( "DunderMifflinProducts", "products", [ { $match: { brand: "Dunder Mifflin" } } ]);
{ "ok" : 1 }
> db.DunderMifflinProducts.find();
{ "_id" : 102, "name" : "Paper", "brand" : "Dunder Mifflin", "price" : 1000, "rating" : 5, "qty" : 100 }
{ "_id" : 103, "name" : "Printer", "brand" : "Dunder Mifflin", "price" : 5000, "rating" : 4, "qty" : 10 }
{ "_id" : 104, "name" : "Printer", "brand" : "Dunder Mifflin", "price" : 2000, "rating" : 3, "qty" : 20 }
{ "_id" : 105, "name" : "Staple", "brand" : "Dunder Mifflin", "price" : 500, "rating" : 3.5, "qty" : 50 }
>
```

Drop a view

```
> db.DunderMifflinProducts.drop();
true
```

4. Joins

Joins in Mongodb are performed using lookup operation, it allows to join two or more operations. Can be done using aggregation pipeline

```
> db.inventory.find();
{ "_id" : ObjectId("687de74dc83c783b9464e86d"), "prodId" : 101, "pname" : "Pencil", "price" : 10, "qty" : 200 }
{ "_id" : ObjectId("687de77ac83c783b9464e86e"), "prodId" : 102, "pname" : "Notebook", "price" : 50, "qty" : 150 }
{ "_id" : ObjectId("687de77ac83c783b9464e86f"), "prodId" : 103, "pname" : "Eraser", "price" : 5, "qty" : 300 }
{ "_id" : ObjectId("687de77ac83c783b9464e870"), "prodId" : 104, "pname" : "Pen", "price" : 20, "qty" : 250 }
{ "_id" : ObjectId("687de77ac83c783b9464e871"), "prodId" : 105, "pname" : "Sharpener", "price" : 8, "qty" : 180 }
```

```
> db.orders.find();
{ "_id" : ObjectId("687de808c83c783b9464e872"), "orderId" : 1, "prodId" : 101, "custId" : 1, "num" : 20 }
{ "_id" : ObjectId("687de843c83c783b9464e873"), "orderId" : 2, "prodId" : 101, "custId" : 2, "num" : 30 }
{ "_id" : ObjectId("687de852c83c783b9464e874"), "orderId" : 3, "prodId" : 102, "custId" : 2, "num" : 20 }
{ "_id" : ObjectId("687de85fc83c783b9464e875"), "orderId" : 4, "prodId" : 102, "custId" : 3, "num" : 10 }
{ "_id" : ObjectId("687de870c83c783b9464e876"), "orderId" : 5, "prodId" : 103, "custId" : 4, "num" : 40 }
{ "_id" : ObjectId("687de884c83c783b9464e877"), "orderId" : 6, "prodId" : 103, "custId" : 3, "num" : 30 }
>
```

```
> db.createView(
... "sales",
... "orders",
... [
... { $lookup:
... { from: "inventory", localField: "prodId", foreignField: "prodId", as: "inventoryDocs" }
... },
... {
... $project:
... { _id: 0, prodId: 1, orderId: 1, num: 1, price: "$inventoryDocs.price" }
... },
... { $unwind: "$price" }
... ]);
{ "ok" : 1 }
> db.sales.find();
{ "orderId" : 1, "prodId" : 101, "num" : 20, "price" : 10 }
{ "orderId" : 2, "prodId" : 101, "num" : 30, "price" : 10 }
{ "orderId" : 3, "prodId" : 102, "num" : 20, "price" : 50 }
{ "orderId" : 4, "prodId" : 102, "num" : 10, "price" : 50 }
{ "orderId" : 5, "prodId" : 103, "num" : 40, "price" : 5 }
{ "orderId" : 6, "prodId" : 103, "num" : 30, "price" : 5 }
```

5. Index

Indexes support efficient execution of query. An index is a special data structure that stores the subset of data in a way that allows mongoDB to quickly locate documents in the collection. With index mongoDB doesn't have to scan the entire document but instead uses index to directly find the relevant data.

```
> db.products.createIndex(
... { brand: 1 });
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
>
```

```
> db.products.getIndexes();
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "products.products"
        },
        {
                "v" : 2,
                "key" : {
                        "brand" : 1
                },
                "name" : "brand_1",
                "ns" : "products.products"
        }
]
```