**Name :-** Prashant Suresh Shirgave

**Roll No:-**3 **Batch:**T1

**Class:** TY(CSE-AIML)

## Experiment No. 6

**Title :-** Study of Open Source NOSQL Database: MongoDB (Installation, Basic CRUD operations, Execution).

**Aim :** Demonstrate Installation of MongoDB on Ubunt and Basic CRUD operations.

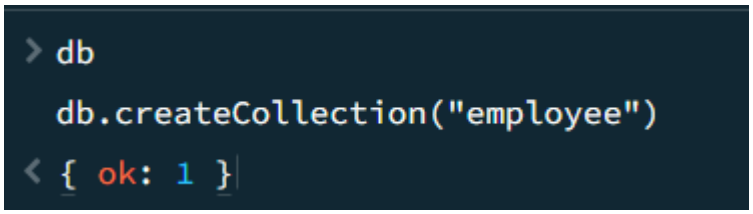
## Implementation:

**Consider the following**
employee(empid, name, salary, designation)

**Execute the following queries:**
**1) Create employee collection.**

```
db.createCollection("employee")
```

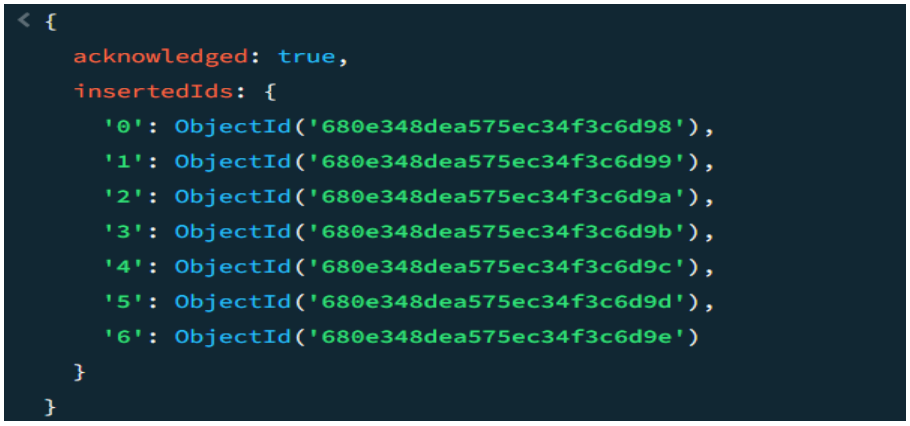
Output:




**2) Insert at least 5 records with meaningful data.**

```
db.employee.insertMany([
  { empid: 101, name: "Sanika", salary: 69000, designation: "Developer" },
  { empid: 102, name: "Prashant", salary: 68000, designation: "Supervisor" },
  { empid: 103, name: "Pranali", salary: 62000, designation: "Analyst" },
  { empid: 104, name: "Radha", salary: 59000, designation: "Supervisor" },
  { empid: 105, name: "Rushi", salary: 65000, designation: "Manager" },
  { empid: 106, name: "Saniya", salary: 61000, designation: "Intern" },
  { empid: 107, name: "Akshata", salary: 57000, designation: "Consultant" }
])
```

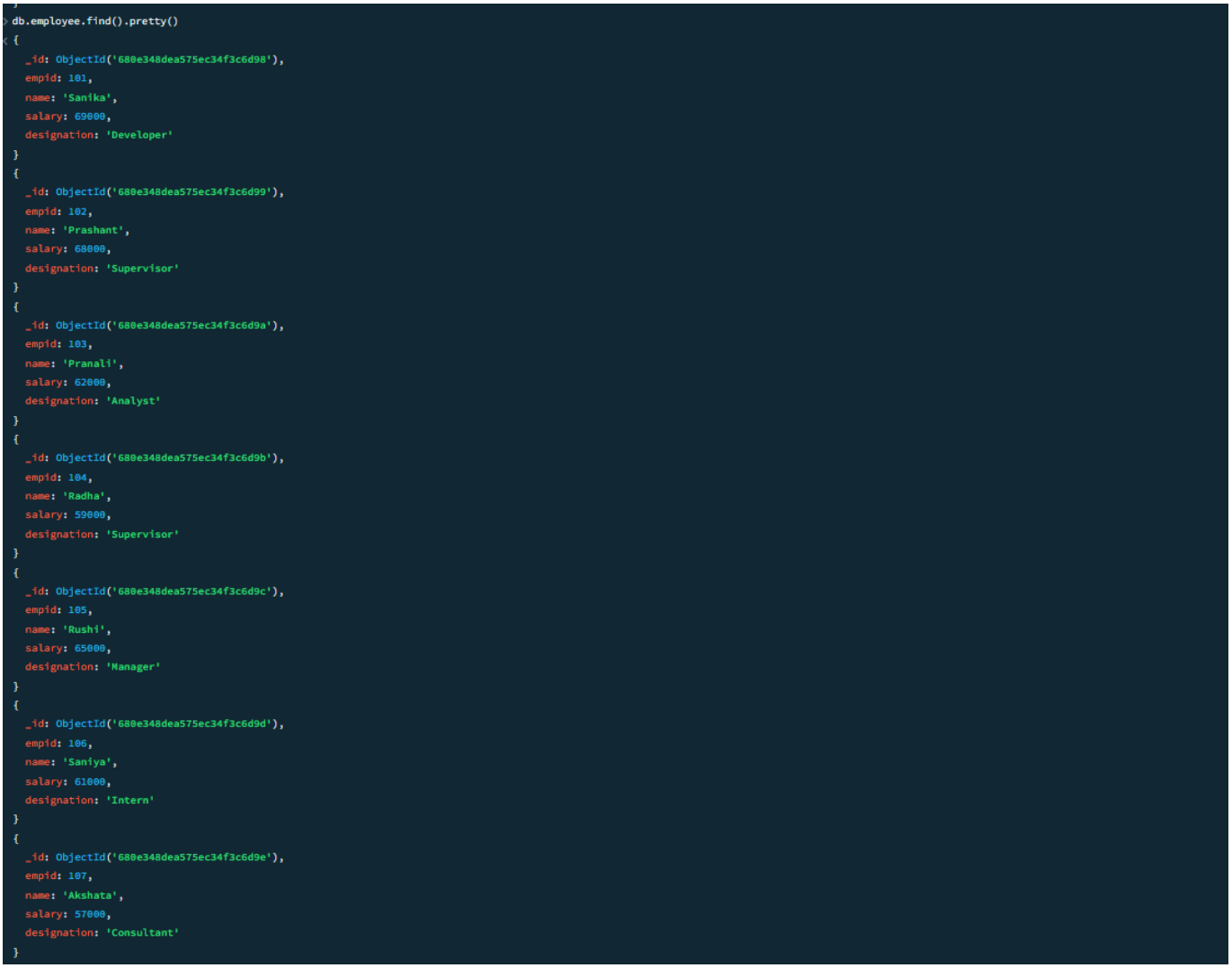
Output:



**3) Display all employees.**

db.employee.find().pretty()

Output:




**4) Update designation of employee 101 to 'Asst. Prof.'**

```
db.employee.updateOne(
{ empid: 101 },
{ $set: { designation: "Asst. Prof." } }
)
```

Output:

**5) Delete the employees having designation “Supervisor”.**

```
db.employee.deleteMany(
 { designation: "Supervisor" }
)
```

Output:

```
{
  acknowledged: true,
  deletedCount: 2
}
```

```
{
  _id: ObjectId('680e348dea575ec34f3c6d98'),
  empid: 101,
  name: 'Sanika',
  salary: 69000,
  designation: 'Asst. Prof.'
}
{
  _id: ObjectId('680e348dea575ec34f3c6d9a'),
  empid: 103,
  name: 'Pranali',
  salary: 62000,
  designation: 'Analyst'
}
{
  _id: ObjectId('680e348dea575ec34f3c6d9c'),
  empid: 105,
  name: 'Rushi',
  salary: 65000,
  designation: 'Manager'
}
{
  _id: ObjectId('680e348dea575ec34f3c6d9d'),
  empid: 106,
  name: 'Saniya',
  salary: 61000,
  designation: 'Intern'
}
{
  _id: ObjectId('680e348dea575ec34f3c6d9e'),
  empid: 107,
  name: 'Akshata',
  salary: 57000,
  designation: 'Consultant'
}
```

**Conclusion:** Students will able to install Mongodb on Unbutu operating system and solve queries using MongoDB.

**Name :-** Prashant Suresh Shiragave

**Roll No:-**3 **Batch:**T1

**Class:** TY(CSE-AIML)

## Experiment No. 7

**Title :-** Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators).

**Aim :** To learn MongoDB basics.

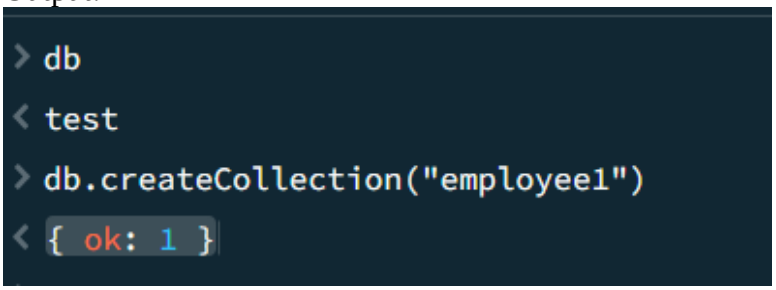
## Implementation:

**Consider the following**
employee(empid, name, salary, designation)

**Execute the following queries:**
**1) Create employee collection.**

```
db.createCollection("employee1")
```

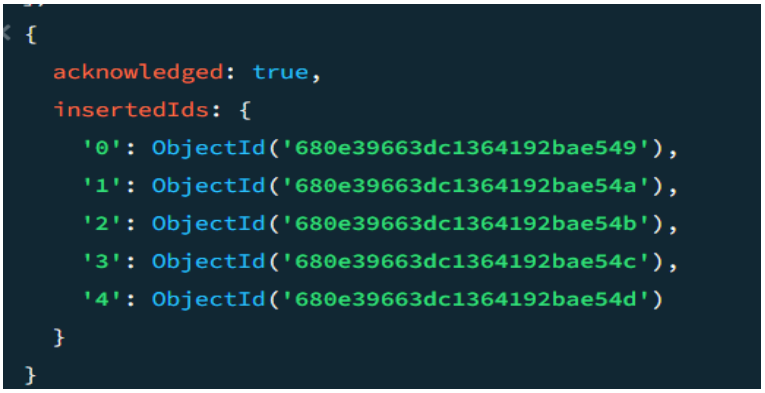
Output:




**2) Insert at least 5 records with meaningful data.**

```
db.employee1.insertMany([
{ empid: 1, name: "Sanika", salary: 12000, designation: "Developer" },
{ empid: 2, name: "Prashant", salary: 9500, designation: "Designer" },
{ empid: 3, name: "Pranali", salary: 18000, designation: "Analyst" },
{ empid: 4, name: "Radha", salary: 5000, designation: "Tester" },
{ empid: 5, name: "Rushi", salary: 20000, designation: "Manager" }
])
```

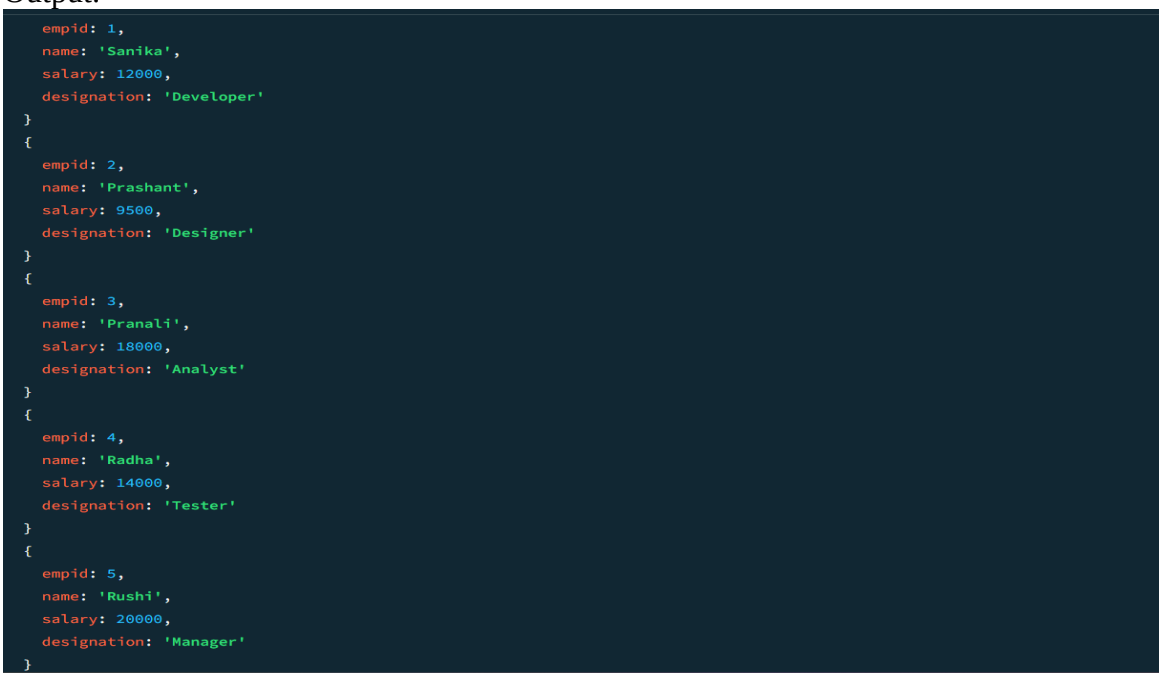
Output:



**3) Display all employees.**

db.employee1.find().pretty()

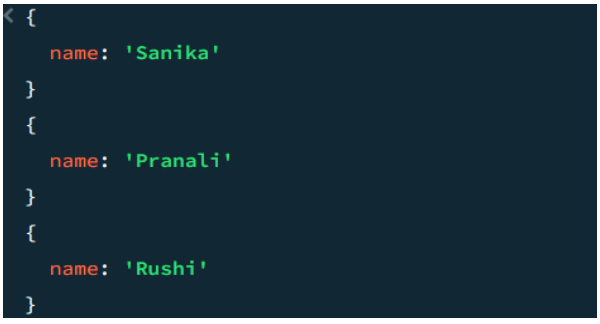
Output:




**4) Find name of the employees whose salary is >10000.**

```
db.employee1.find(
{ salary: { $gt: 10000 } },
{ name: 1, _id: 0 }
)
```

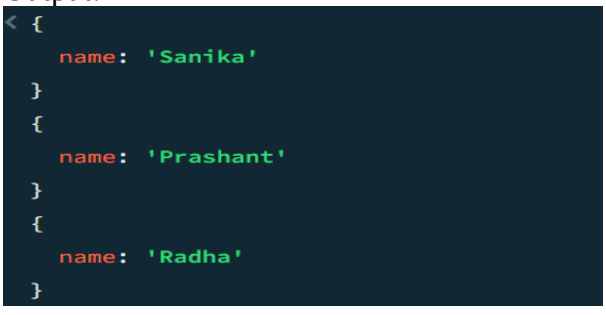
Output:




**5) Find name of the employees whose salary is <15000.**

```
db.employee1.find(
{ salary: { $lt: 15000 } },
{ name: 1, _id: 0 }
)
```

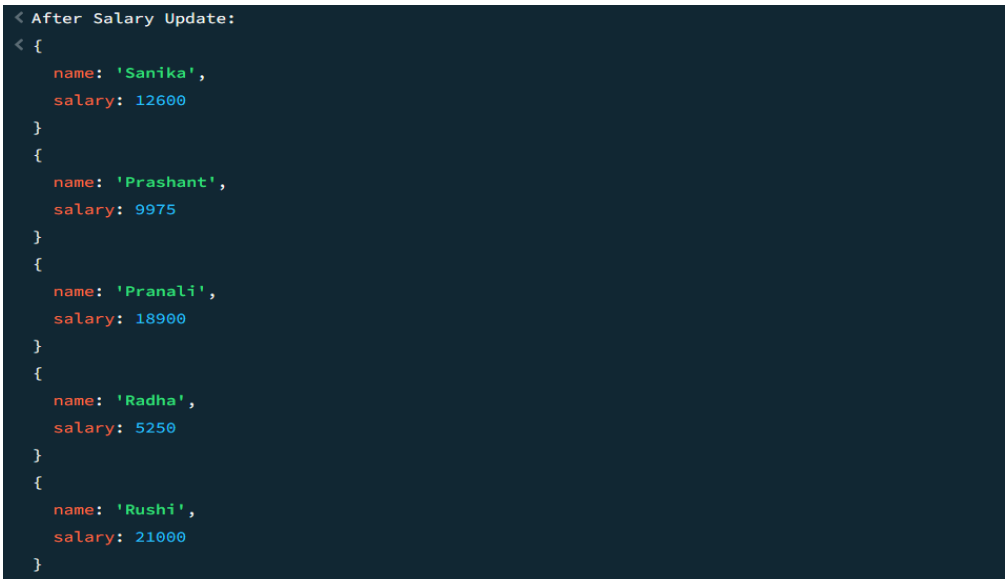
Output:



**6) Increment the salary of employees by 5%.**

```
db.employee1.updateMany(
 {},
 [
  { $set: { salary: { $multiply: ["$salary", 1.05] } } }
 ]
)
```

Output:



```
After Salary Update:
[
  {
    name: 'Sanika',
    salary: 12600
  }
  {
    name: 'Prashant',
    salary: 9975
  }
  {
    name: 'Pranali',
    salary: 18900
  }
  {
    name: 'Radha',
    salary: 5250
  }
  {
    name: 'Rushi',
    salary: 21000
  }
```

**7) Display top 2 employees having highest salary.**

```
db.employee1.find()
.sort({ salary: -1 })
.limit(2)
```

Output:

```
[
  {
    _id: ObjectId('680e3fad3dc1364192bae54f'),
    empid: 7,
    name: 'Prashant',
    salary: 26000,
    designation: 'Designer'
  }
  {
    _id: ObjectId('680e3fad3dc1364192bae54e'),
    empid: 6,
    name: 'Sanika',
    salary: 25000,
    designation: 'Developer'
  }
```

**Conclusion:** Students will able to solve queries using MongoDB.

**Name :-** Prashant Suresh Shirgave

**Roll No:-**3 **Batch:**T1

**Class:** TY(CSE-AIML)

## Experiment No. 8

**Title :-** Implement aggregation with suitable example using MongoDB.

**Aim :** To implement aggregation with suitable example using MongoDB.

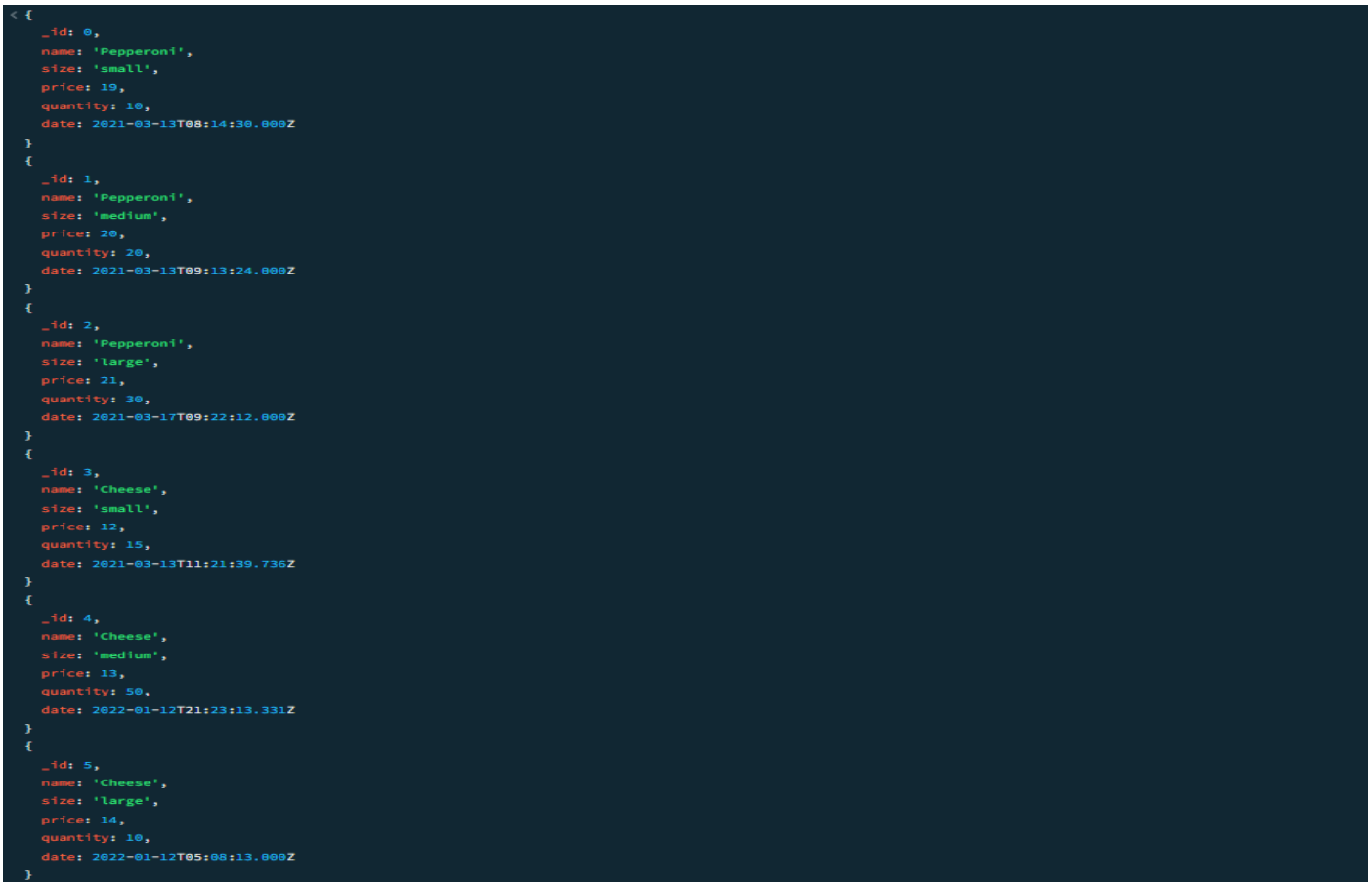
## Implementation:

**Execute the following queries:**

**1) Insert Query for Pizza Orders Collection**

```
db.orders.insertMany([
 { _id: 0, name: "Pepperoni", size: "small", price: 19, quantity: 10, date: ISODate("2021-03-13T08:14:30Z") },
 { _id: 1, name: "Pepperoni", size: "medium", price: 20, quantity: 20, date: ISODate("2021-03-13T09:13:24Z") },
 { _id: 2, name: "Pepperoni", size: "large", price: 21, quantity: 30, date: ISODate("2021-03-17T09:22:12Z") },
 { _id: 3, name: "Cheese", size: "small", price: 12, quantity: 15, date: ISODate("2021-03-13T11:21:39.736Z") },
 { _id: 4, name: "Cheese", size: "medium", price: 13, quantity: 50, date: ISODate("2022-01-12T21:23:13.331Z") },
 { _id: 5, name: "Cheese", size: "large", price: 14, quantity: 10, date: ISODate("2022-01-12T05:08:13Z") },
 { _id: 6, name: "Vegan", size: "small", price: 17, quantity: 10, date: ISODate("2021-01-13T05:08:13Z") },
 { _id: 7, name: "Vegan", size: "medium", price: 18, quantity: 10, date: ISODate("2021-01-13T05:10:13Z") }
]);
```

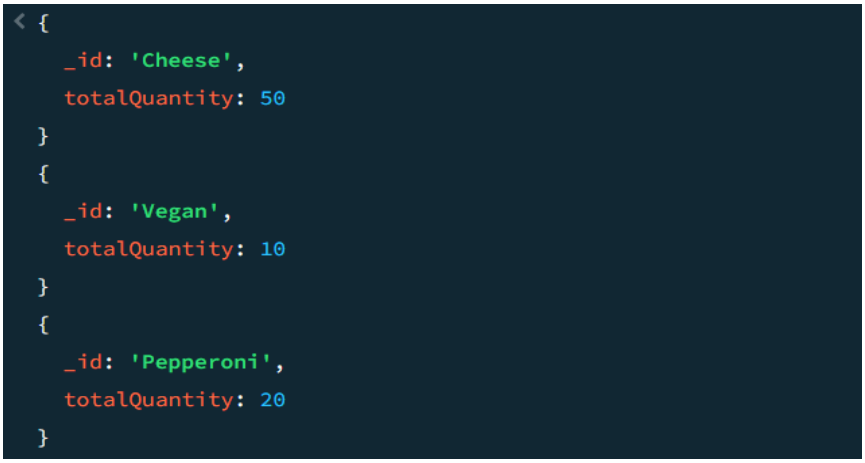
Output:



**2) Calculate Total Order Quantity of Medium Size Pizzas.**

```
db.orders.aggregate([
  // Stage 1: Filter pizza order documents by pizza size
  {
    $match: { size: "medium" }
  },
  // Stage 2: Group remaining documents by pizza name and calculate total quantity
  {
    $group: {
      _id: "$name",
      totalQuantity: { $sum: "$quantity" }
    }
  }
])
```

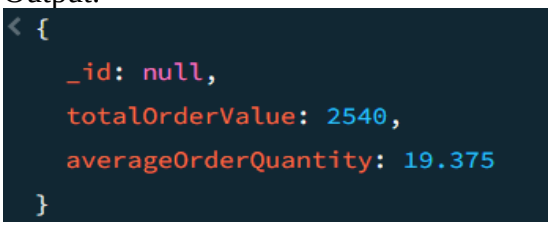
Output:




**3) Calculate Total Order Value and Average Order Quantity.**

```
 db.orders.aggregate([
  // Stage 1: Calculate the order value for each pizza order (quantity * price)
  {
    $addFields: {
      orderValue: { $multiply: ["$quantity", "$price"] }  // Add a new field 'orderValue'
    }
  },
  // Stage 2: Group all documents to calculate the total order value and average order quantity
  {
    $group: {
      _id: null,  // Aggregate over the entire collection
      totalOrderValue: { $sum: "$orderValue" },  // Calculate the total order value
      averageOrderQuantity: { $avg: "$quantity" }  // Calculate the average order quantity
    }
  }
])
```

Output:




**Conclusion:** Students are able to solve queries on aggregation from Mongodb using JAVA.

**Name :-** Prashant Suresh Shirgave

**Roll No:-**3 **Batch:**T1

**Class:** TY(CSE-AIML)

## Experiment No. 9

**Title :-** Implement Map Reduce operation with suitable example using MongoDB.

**Aim :** To understand Map Reduce operation.

### Implementation:

**1) Insert Data into Orders Collection**

```
db.orders.insertMany([
 {_id:1, ord_date:new Date("2020-03-01"), items:[{sku:"A100", qty:5}, {sku:"B200", qty:10}]},
 {_id:2, ord_date:new Date("2020-03-02"), items:[{sku:"A100", qty:15}, {sku:"C300", qty:20}]},
 {_id:3, ord_date:new Date("2020-02-28"), items:[{sku:"B200", qty:7}]},
 {_id:4, ord_date:new Date("2020-03-05"), items:[{sku:"A100", qty:10}, {sku:"C300", qty:5}]}
]);
```

**Output:**

```
--- Orders Inserted ---
{
  _id: 1,
  ord_date: 2020-03-01T00:00:00.000Z,
  items: [ { sku: 'A100', qty: 5 }, { sku: 'B200', qty: 10 } ]
}
{
  _id: 2,
  ord_date: 2020-03-02T00:00:00.000Z,
  items: [ { sku: 'A100', qty: 15 }, { sku: 'C300', qty: 20 } ]
}
{
  _id: 3,
  ord_date: 2020-02-28T00:00:00.000Z,
  items: [ { sku: 'B200', qty: 7 } ]
}
{
  _id: 4,
  ord_date: 2020-03-05T00:00:00.000Z,
  items: [ { sku: 'A100', qty: 10 }, { sku: 'C300', qty: 5 } ]
}
```

**1) Define the Map Function**

```
var mapFunction2 = function() {
  this.items.forEach(function(item) {
    emit(item.sku, { count: 1, qty: item.qty });
  });
};
```

**2) Define the Reduce Function**

```
var reduceFunction2 = function(keySKU, countObjVals) {
  var reducedVal = { count: 0, qty: 0 };

  countObjVals.forEach(function(value) {
    reducedVal.count += value.count;
    reducedVal.qty += value.qty;
  });

  return reducedVal;
};
```

**3) Define the Finalize Function**

```
var finalizeFunction2 = function(key, reducedVal) {
  if (reducedVal.count > 0) {
    reducedVal.avg = reducedVal.qty / reducedVal.count;
  } else {
    reducedVal.avg = 0;
  }
  return reducedVal;
};
```

**4) Perform Map-Reduce Operation**

```
db.orders.mapReduce(
  mapFunction2,
  reduceFunction2,
  {
    query: { ord_date: { $gte: new Date("2020-03-01") } },
    out: { merge: "map_reduce_example2" },
    finalize: finalizeFunction2
  })
```

```
< { result: 'map_reduce_example2', ok: 1 }
```

**5) Verify Results**

```
db.map_reduce_example2.find().sort({ _id: 1 });
```

```
   --- Map-Reduce Results (SKU-wise Order Count, Total Qty, Avg Qty) ---
< { _id: 'A100', value: { count: 3, qty: 30, avg: 10 } }
< { _id: 'B200', value: { count: 1, qty: 10, avg: 10 } }
< { _id: 'C300', value: { count: 2, qty: 25, avg: 12.5 } }
```

**Conclusion:**

Students are able to Map Reduce operation.