SLIDE

MONGODB OPERATORS

What are MongoDB operators?

- MongoDB offers different types of operators that can be used to interact with the database.
- Operators are special symbols or keywords that inform a compiler or an interpreter to carry out mathematical or logical operations.
- The query operators enhance the functionality of MongoDB by allowing developers to create complex queries to interact with data sets that match their applications.
- MongoDB offers the following query operator types:
    i. Comparison
    ii. Logical
    iii. Element
    iv. Evaluation
    v. Geospatial
    vi. Array
    vii. Bitwise
    viii. Comments

We are going to discuss **comparison** and **logical** operators in this slide

Comparison Operators

Comparison Operators

MongoDB comparison operators can be used to compare values in a document. The following table contains the common comparison operators.

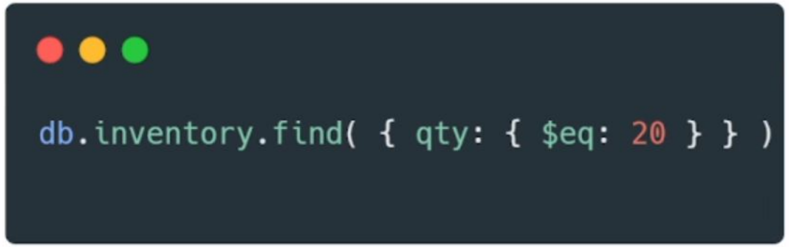
Operator	Description
$eq	Matches values that are equal to the given value.
$gt	Matches if values are greater than the given value.
$lt	Matches if values are less than the given value.
$gte	Matches if values are greater or equal to the given value.
$lte	Matches if values are less or equal to the given value.
$in	Matches any of the values in an array.
$ne	Matches values that are not equal to the given value.
$nin	Matches none of the values specified in an array.

Comparison Operators ($eq)

In this example, we retrieve the document with the exact **qty** value **20**.

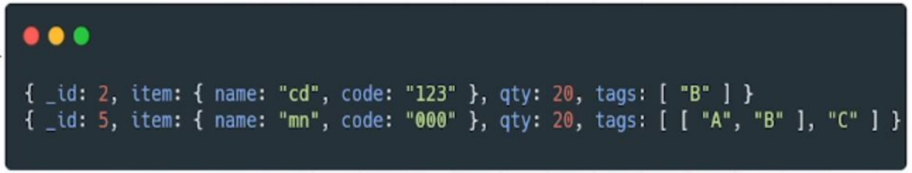
Syntax: `{field: {$eq: value}}`

Input:



```
db.inventory.find( { qty: { $eq: 20 } } )
```

Output:



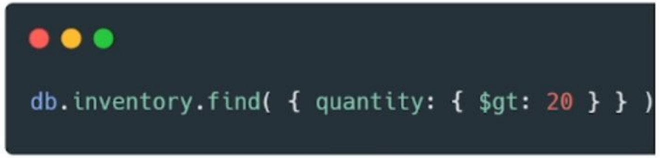
```
{ _id: 2, item: { name: "cd", code: "123" }, qty: 20, tags: [ "B" ] }
{ _id: 5, item: { name: "mn", code: "000" }, qty: 20, tags: [ [ "A", "B" ], "C" ] }
```

Comparison Operators ($gt)

In this example, we retrieve the documents where the **quantity** is greater than **20**.

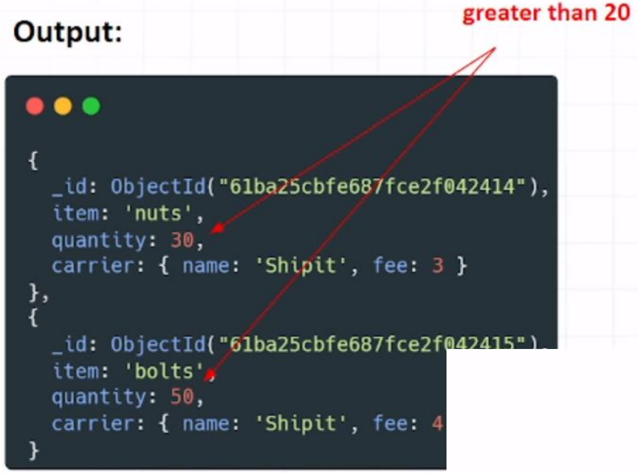
Syntax: `{field: {$gt: value}}`

Input:



```
db.inventory.find( { quantity: { $gt: 20 } } )
```

Output:



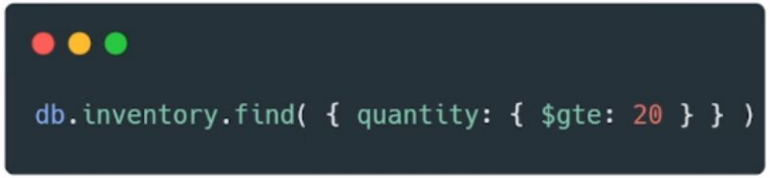
```
{
  _id: ObjectId("61ba25cbfe687fce2f042414"),
  item: 'nuts',
  quantity: 30,
  carrier: { name: 'Shipit', fee: 3 }
},
{
  _id: ObjectId("61ba25cbfe687fce2f042415"),
  item: 'bolts',
  quantity: 50,
  carrier: { name: 'Shipit', fee: 4
}
```

Comparison Operators ($gte)

In this example, we retrieve the documents where the **quantity** is greater than or equal to **20**.

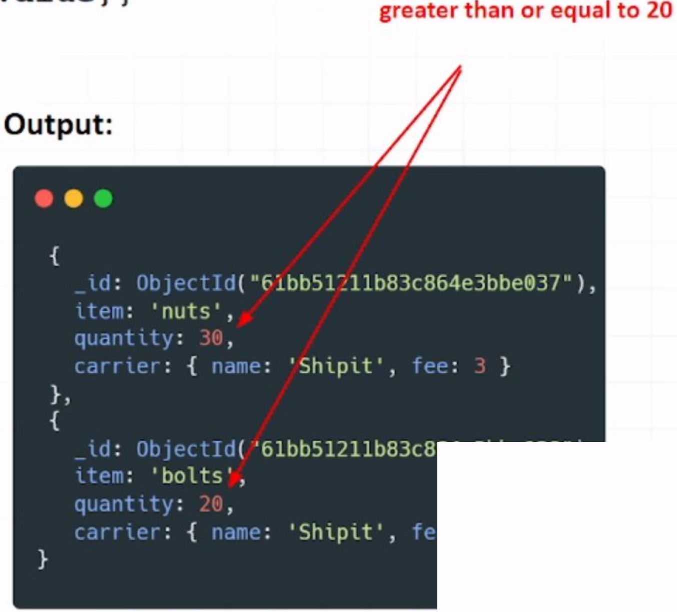
Syntax: `{field: {$gte: value}}`

Input:



```
db.inventory.find( { quantity: { $gte: 20 } } )
```

Output:



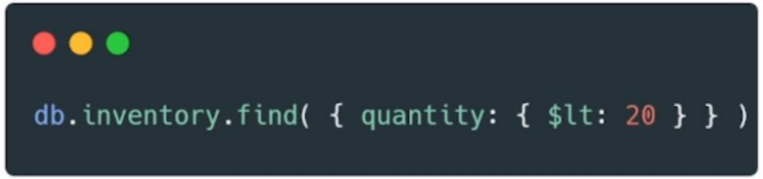
```
{
   _id: ObjectId("61bb51211b83c864e3bbe037"),
   item: 'nuts',
   quantity: 30,
   carrier: { name: 'Shipit', fee: 3 }
 },
 {
   _id: ObjectId("61bb51211b83c8
   item: 'bolts',
   quantity: 20,
   carrier: { name: 'Shipit', fe
}
```

Comparison Operators ($lt)

In this example, we retrieve the documents where the **quantity** is less than **20**.

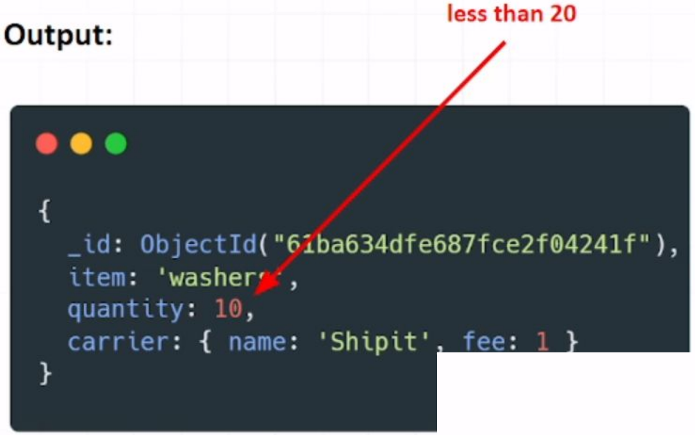
Syntax: `{field: {$lt: value}}`

Input:



```
db.inventory.find( { quantity: { $lt: 20 } } )
```

Output:



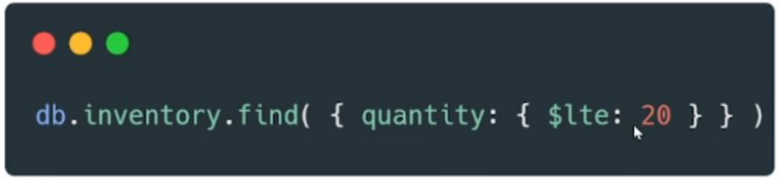
```
{
  _id: ObjectId("61ba634dfe687fce2f04241f"),
  item: 'washers',
  quantity: 10,
  carrier: { name: 'Shipit', fee: 1 }
}
```

Comparison Operators ($lte)

In this example, we retrieve the documents where the **quantity** is less than or equal to **20**.

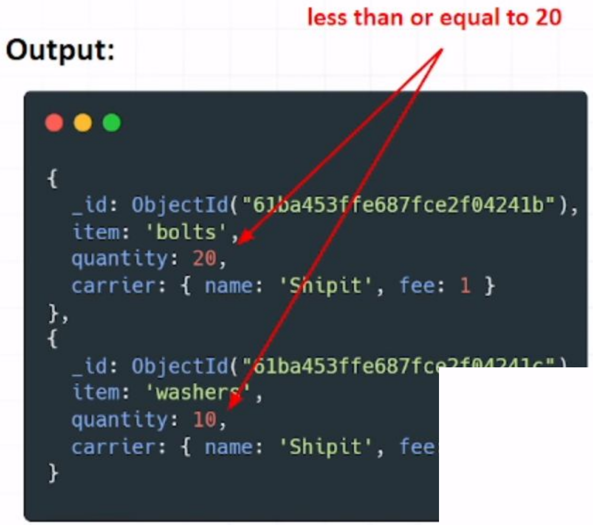
Syntax: `{field: {$lte: value}}`

Input:



```
db.inventory.find( { quantity: { $lte: 20 } } )
```

Output:



```
{
  _id: ObjectId("61ba453ffe687fce2f04241b"),
  item: 'bolts',
  quantity: 20,
  carrier: { name: 'Shipit', fee: 1 }
},
{
  _id: ObjectId("61ba453ffe687fce2f04241c")
  item: 'washers',
  quantity: 10,
  carrier: { name: 'Shipit', fee
}
```

Comparison Operators ($in)

In this example, we retrieve the documents where the **quantity** contains the **given values**.

Syntax: `{ field: { $in: [<value1>, <value2>, ... <valueN> ] } }`

Input:

```
db.inventory.find( { quantity: { $in: [ 5, 15 ] } }, { _id: 0 } )
```

Output:

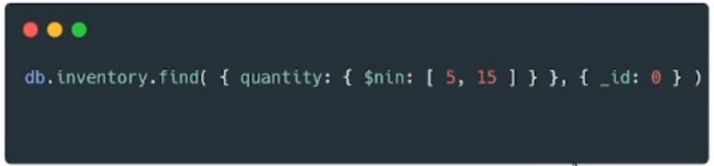
```
{ item: 'Erasers', quantity: 15, tags: [ 'school', 'home' ] },
{ item: 'Books', quantity: 5, tags: [ 'school', 'storage', 'home' ] }
```

Comparison Operators ($nin)

In this example, we retrieve the documents where the **quantity** do not contain the **given values.**

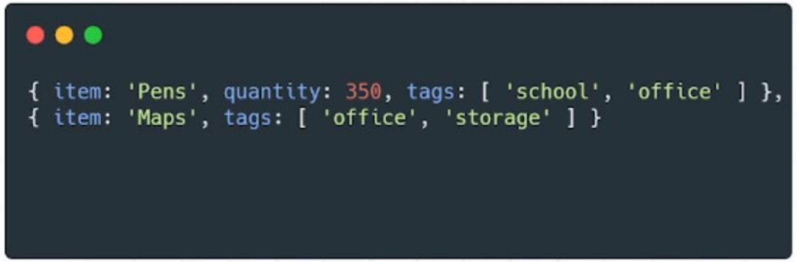
Syntax: `{ field: { $nin: [ <value1>, <value2> ... <valueN> ] } }`

Input:



```
db.inventory.find( { quantity: { $nin: [ 5, 15 ] } }, { _id: 0 } )
```

Output:



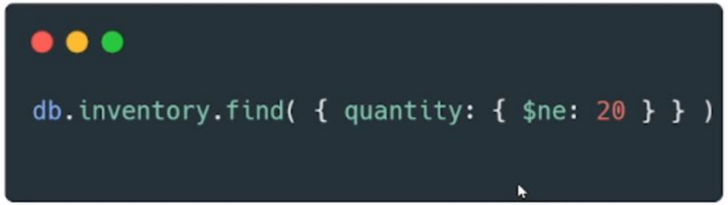
```
{ item: 'Pens', quantity: 350, tags: [ 'school', 'office' ] },
{ item: 'Maps', tags: [ 'office', 'storage' ] }
```

Comparison Operators ($ne)

In this example, we retrieve the documents where the **quantity** is not equal to the **given values.**

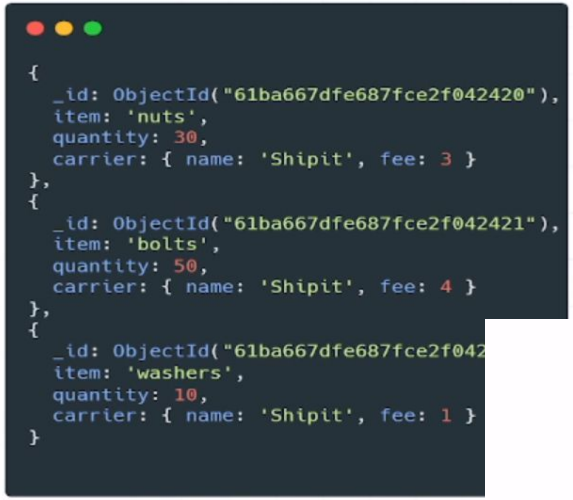
Syntax: `{ field: { $ne: value } }`

Input:



```
db.inventory.find( { quantity: { $ne: 20 } } )
```

Output:



```
{
  _id: ObjectId("61ba667dfe687fce2f042420"),
  item: 'nuts',
  quantity: 30,
  carrier: { name: 'Shipit', fee: 3 }
},
{
  _id: ObjectId("61ba667dfe687fce2f042421"),
  item: 'bolts',
  quantity: 50,
  carrier: { name: 'Shipit', fee: 4 }
},
{
  _id: ObjectId("61ba667dfe687fce2f042
  item: 'washers',
  quantity: 10,
  carrier: { name: 'Shipit', fee: 1 }
}
```

Logical Operators ($ne)

MongoDB logical operators can be used to filter data based on given conditions. These operators provide a way to combine multiple conditions. Each operator equates the given condition to a true or false value.

Operator	Description
$and	Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
$not	Inverts the effect of a query expression and returns documents that do *not* match the query expression.
$nor	The opposite of the OR operator. The logical NOR operator will join two or more queries and return documents that do not match the given query condition
$or	Joins query clauses with a logical OR returns all documents tl the conditions of either clause.

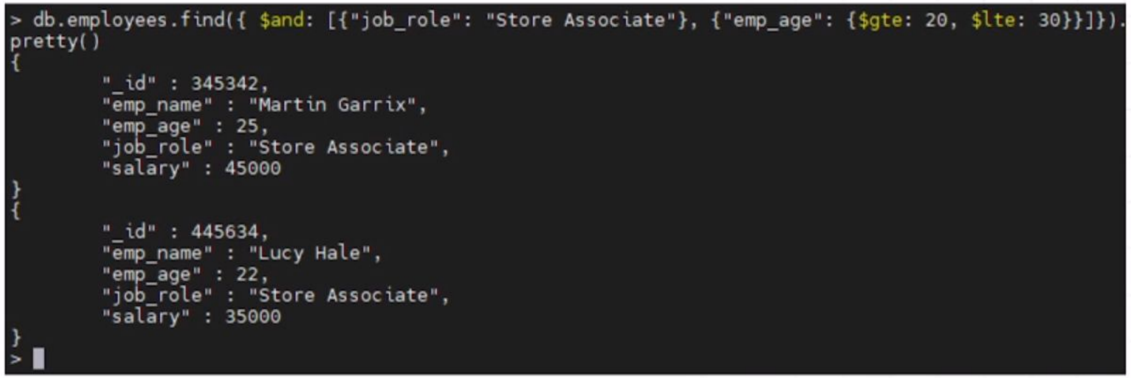
Logical Operators ($and)

Syntax: { $and: [{ <expression1> }, { <expression2> } , ... , { <expressionN> }] }

Input:

```
db.employees.find({ $and: [{"job_role": "Store Associate"}, {"emp_age": {$gte: 20, $lte: 30}}]}).pretty()
```

Output:

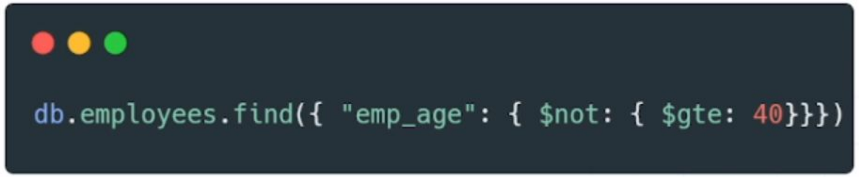


```
> db.employees.find({ $and: [{"job_role": "Store Associate"}, {"emp_age": {$gte: 20, $lte: 30}}]}).
pretty()
{
        "_id" : 345342,
        "emp_name" : "Martin Garrix",
        "emp_age" : 25,
        "job_role" : "Store Associate",
        "salary" : 45000
}
{
        "_id" : 445634,
        "emp_name" : "Lucy Hale",
        "emp_age" : 22,
        "job_role" : "Store Associate",
        "salary" : 35000
}
>
```

Logical Operators ($not)

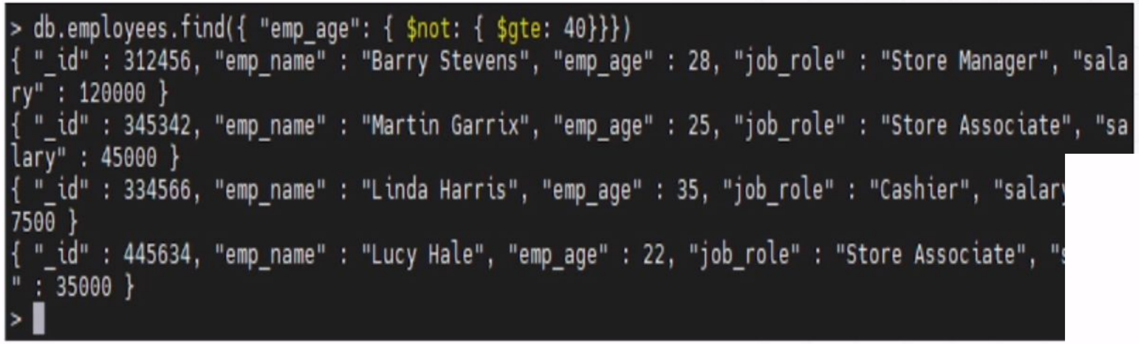
Syntax: { field: { $not: { <operator-expression> } } }

Input:



```
db.employees.find({ "emp_age": { $not: { $gte: 40}}})
```

Output:



```
> db.employees.find({ "emp_age": { $not: { $gte: 40}}})
{ "_id" : 312456, "emp_name" : "Barry Stevens", "emp_age" : 28, "job_role" : "Store Manager", "sala
ry" : 120000 }
{ "_id" : 345342, "emp_name" : "Martin Garrix", "emp_age" : 25, "job_role" : "Store Associate", "sa
lary" : 45000 }
{ "_id" : 334566, "emp_name" : "Linda Harris", "emp_age" : 35, "job_role" : "Cashier", "salary
7500 }
{ "_id" : 445634, "emp_name" : "Lucy Hale", "emp_age" : 22, "job_role" : "Store Associate", "
" : 35000 }
>
```

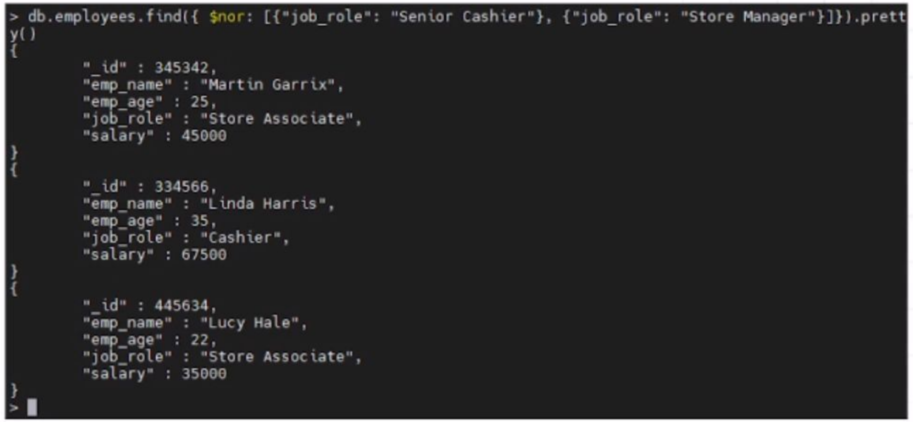
Logical Operators ($nor)

Syntax: { $nor: [{ <expression1> }, { <expression2> }, ... { <expressionN> }] }

Input:

```
db.employees.find({ $nor: [{"job_role": "Senior Cashier"}, {"job_role": "Store Manager"}]}).pretty()
```

Output:



```
> db.employees.find({ $nor: [{"job_role": "Senior Cashier"}, {"job_role": "Store Manager"}]}).prett
y()
{
        "_id" : 345342,
        "emp_name" : "Martin Garrix",
        "emp_age" : 25,
        "job_role" : "Store Associate",
        "salary" : 45000
}
{
        "_id" : 334566,
        "emp_name" : "Linda Harris",
        "emp_age" : 35,
        "job_role" : "Cashier",
        "salary" : 67500
}
{
        "_id" : 445634,
        "emp_name" : "Lucy Hale",
        "emp_age" : 22,
        "job_role" : "Store Associate",
        "salary" : 35000
}
>
```

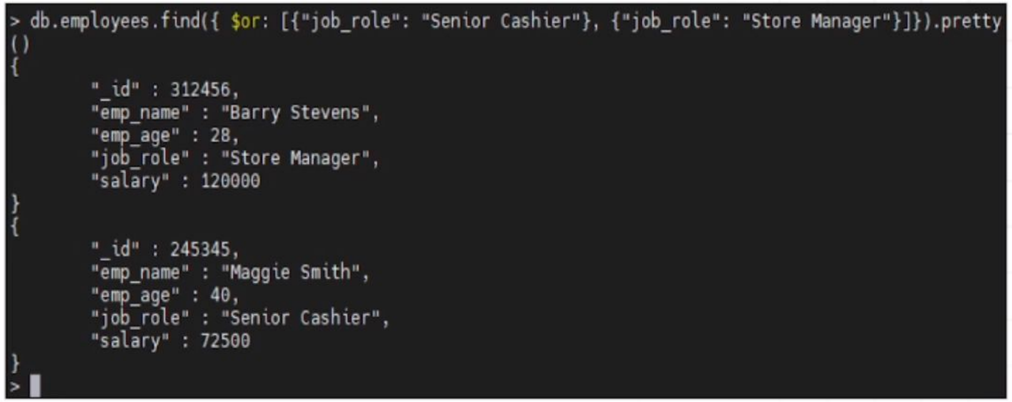
Logical Operators ($or)

Syntax: { $or: [{ <expression1> }, { <expression2> }, ... , { <expressionN> }] }

Input:

```
db.employees.find({ $or: [{"job_role": "Senior Cashier"}, {"job_role": "Store Manager"}]}).pretty()
```

Output:



```
> db.employees.find({ $or: [{"job_role": "Senior Cashier"}, {"job_role": "Store Manager"}]}).pretty
()
{
        "_id" : 312456,
        "emp_name" : "Barry Stevens",
        "emp_age" : 28,
        "job_role" : "Store Manager",
        "salary" : 120000
}
{
        "_id" : 245345,
        "emp_name" : "Maggie Smith",
        "emp_age" : 40,
        "job_role" : "Senior Cashier",
        "salary" : 72500
}
>
```

Thank you!