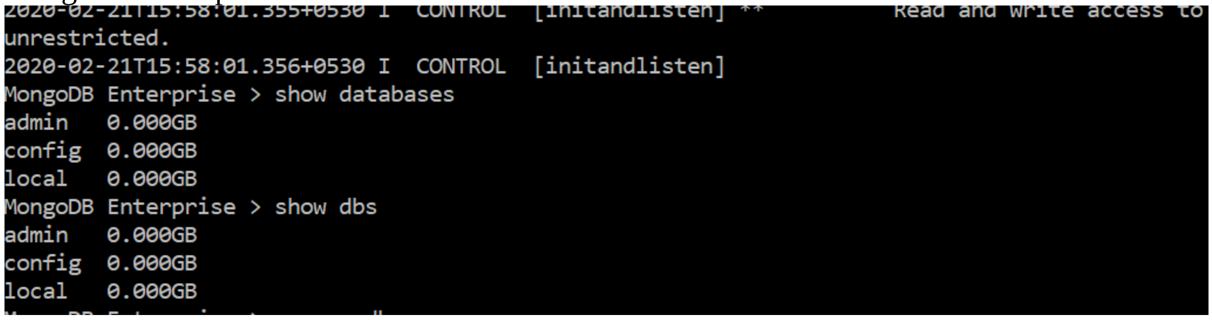
ASSIGNMENT 3

19MCB0022

MONGODB LAB EXERCISE

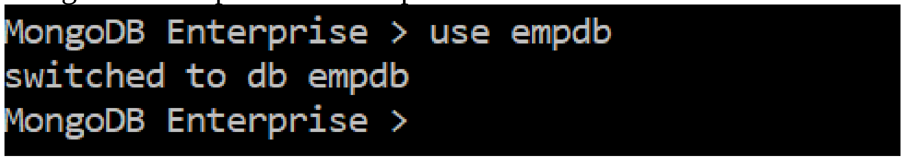
1. DISPLAY ALL DATABASES

   MongoDB Enterprise > show databases



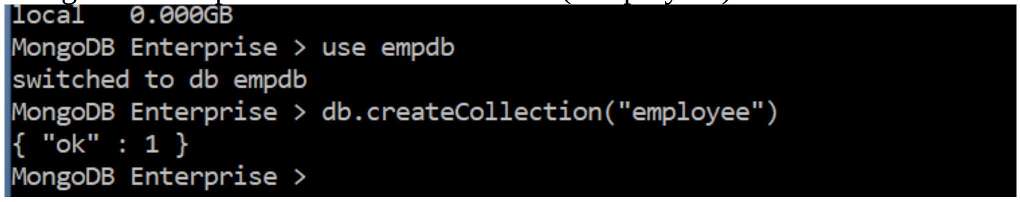
```
2020-02-21T15:58:01.355+0530 I  CONTROL  [initandlisten] **          Read and write access to
unrestricted.
2020-02-21T15:58:01.356+0530 I  CONTROL  [initandlisten]
MongoDB Enterprise > show databases
admin   0.000GB
config  0.000GB
local   0.000GB
MongoDB Enterprise > show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
```

2. ADD AN empdb DATABASE
   MongoDB Enterprise > use empdb



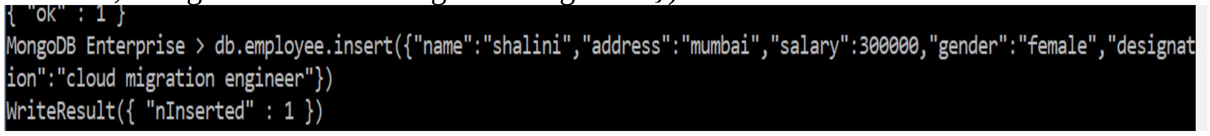
```
MongoDB Enterprise > use empdb
switched to db empdb
MongoDB Enterprise >
```

3. CREATE A COLLECTION BY NAME EMPLOYEE
   MongoDB Enterprise > db.createCollection("employee")



```
local   0.000GB
MongoDB Enterprise > use empdb
switched to db empdb
MongoDB Enterprise > db.createCollection("employee")
{ "ok" : 1 }
MongoDB Enterprise >
```

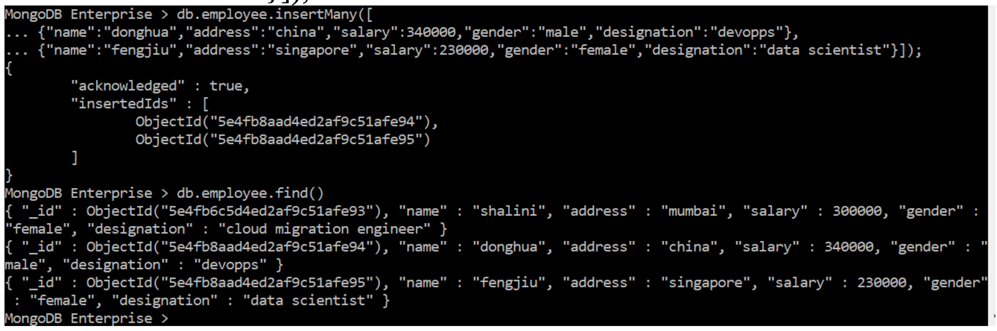
4. Insert records for employee
   MongoDB Enterprise >
   db.employee.insert({"name":"shalini","address":"mumbai","salary":300000,"gender":"female","designation":"cloud migration engineer"})



```
{ "ok" : 1 }
MongoDB Enterprise > db.employee.insert({"name":"shalini","address":"mumbai","salary":300000,"gender":"female","designat
ion":"cloud migration engineer"})
WriteResult({ "nInserted" : 1 })
```

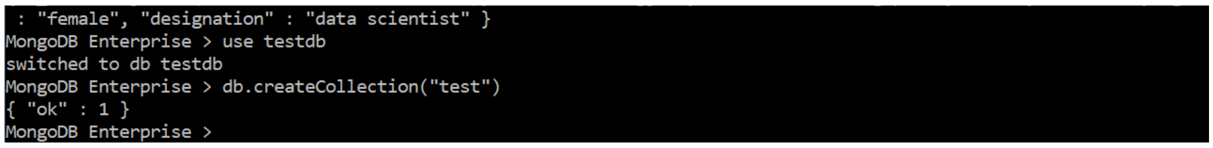
5. Insert multiple commands
   MongoDB Enterprise > db.employee.insertMany([
   ...
   {"name":"donghua","address":"china","salary":340000,"gender":"male","designation":"devopps"},

...
{"name":"fengjiu","address":"singapore","salary":230000,"gender":"female","designation":"data scientist"}]);



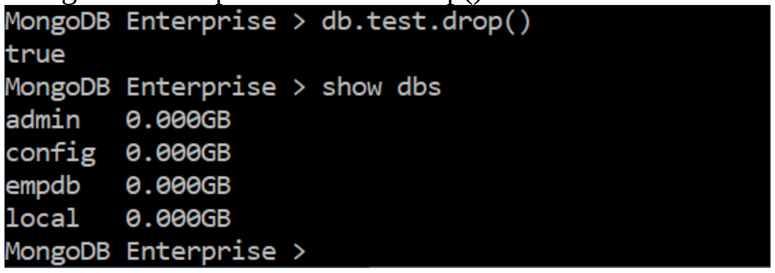
```
MongoDB Enterprise > db.employee.insertMany([
... {"name":"donghua","address":"china","salary":340000,"gender":"male","designation":"devopps"},
... {"name":"fengjiu","address":"singapore","salary":230000,"gender":"female","designation":"data scientist"}]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5e4fb8aad4ed2af9c51afe94"),
                ObjectId("5e4fb8aad4ed2af9c51afe95")
        ]
}
MongoDB Enterprise > db.employee.find()
{ "_id" : ObjectId("5e4fb6c5d4ed2af9c51afe93"), "name" : "shalini", "address" : "mumbai", "salary" : 300000, "gender" :
"female", "designation" : "cloud migration engineer" }
{ "_id" : ObjectId("5e4fb8aad4ed2af9c51afe94"), "name" : "donghua", "address" : "china", "salary" : 340000, "gender" : "
male", "designation" : "devopps" }
{ "_id" : ObjectId("5e4fb8aad4ed2af9c51afe95"), "name" : "fengjiu", "address" : "singapore", "salary" : 230000, "gender"
 : "female", "designation" : "data scientist" }
MongoDB Enterprise >
```

6. Create a db named testdb and insert a collection by name test
   MongoDB Enterprise > use testdb
   switched to db testdb



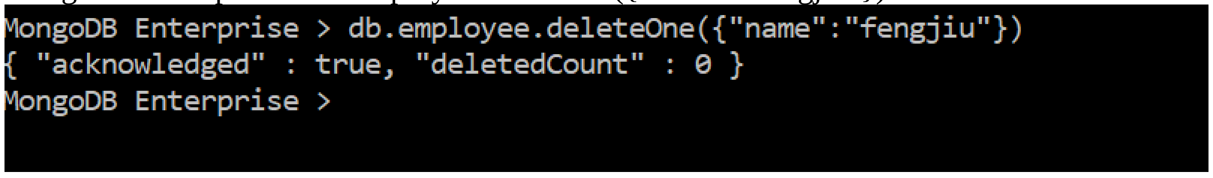
```
 : "female", "designation" : "data scientist" }
MongoDB Enterprise > use testdb
switched to db testdb
MongoDB Enterprise > db.createCollection("test")
{ "ok" : 1 }
MongoDB Enterprise >
```

7. Write commands to drop test collection and testdb
   MongoDB Enterprise > db.test.drop()



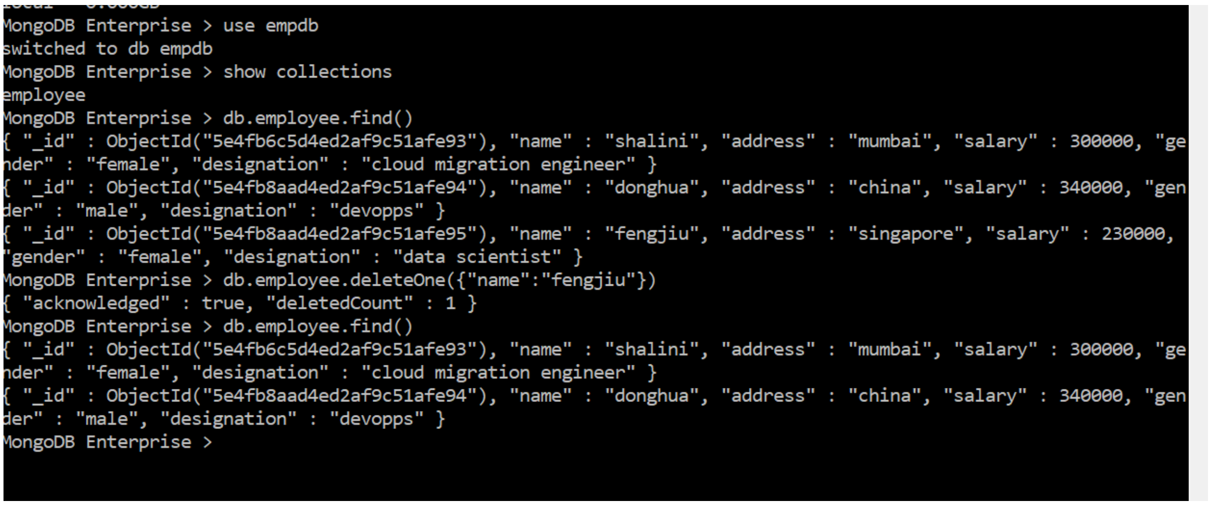
```
MongoDB Enterprise > db.test.drop()
true
MongoDB Enterprise > show dbs
admin   0.000GB
config  0.000GB
empdb   0.000GB
local   0.000GB
MongoDB Enterprise >
```

8. Delete the first document that matches a given name by user
   MongoDB Enterprise > db.employee.deleteOne({"name":"fengjiu"})



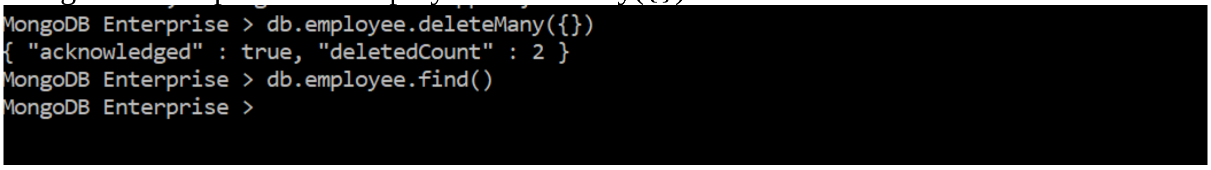
```
MongoDB Enterprise > db.employee.deleteOne({"name":"fengjiu"})
{ "acknowledged" : true, "deletedCount" : 0 }
MongoDB Enterprise >
```

9. Display ALL Documents



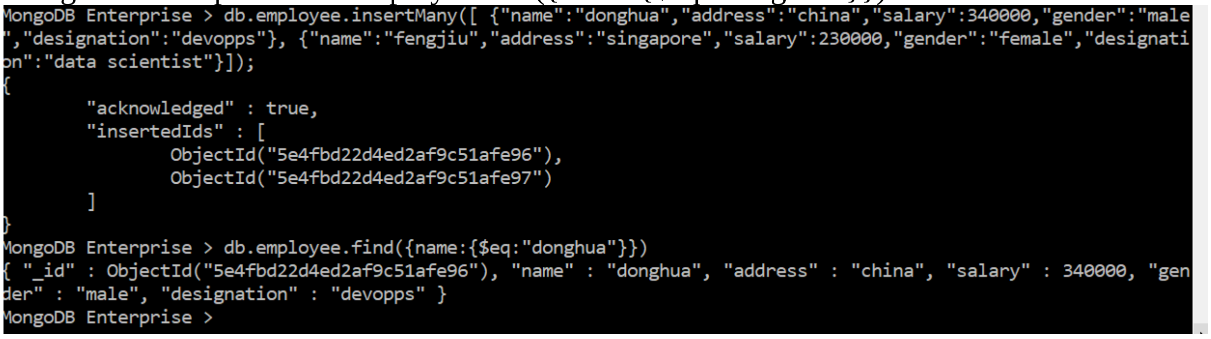
```
MongoDB Enterprise > use empdb
switched to db empdb
MongoDB Enterprise > show collections
employee
MongoDB Enterprise > db.employee.find()
{ "_id" : ObjectId("5e4fb6c5d4ed2af9c51afe93"), "name" : "shalini", "address" : "mumbai", "salary" : 300000, "ge
nder" : "female", "designation" : "cloud migration engineer" }
{ "_id" : ObjectId("5e4fb8aad4ed2af9c51afe94"), "name" : "donghua", "address" : "china", "salary" : 340000, "gen
der" : "male", "designation" : "devopps" }
{ "_id" : ObjectId("5e4fb8aad4ed2af9c51afe95"), "name" : "fengjiu", "address" : "singapore", "salary" : 230000,
"gender" : "female", "designation" : "data scientist" }
MongoDB Enterprise > db.employee.deleteOne({"name":"fengjiu"})
{ "acknowledged" : true, "deletedCount" : 1 }
MongoDB Enterprise > db.employee.find()
{ "_id" : ObjectId("5e4fb6c5d4ed2af9c51afe93"), "name" : "shalini", "address" : "mumbai", "salary" : 300000, "ge
nder" : "female", "designation" : "cloud migration engineer" }
{ "_id" : ObjectId("5e4fb8aad4ed2af9c51afe94"), "name" : "donghua", "address" : "china", "salary" : 340000, "gen
der" : "male", "designation" : "devopps" }
MongoDB Enterprise >
```

10. Delete all documents
    MongoDB Enterprise > db.employee.deleteMany({})



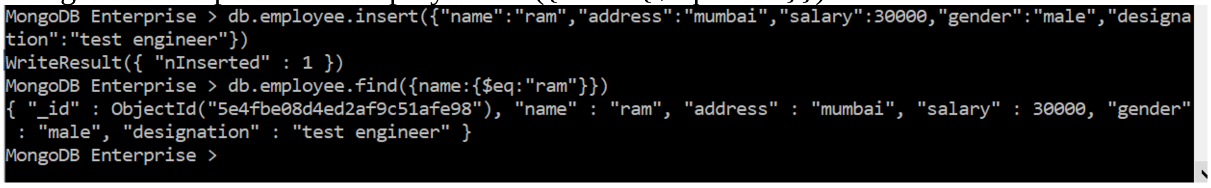
```
MongoDB Enterprise > db.employee.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 2 }
MongoDB Enterprise > db.employee.find()
MongoDB Enterprise >
```

11. Display document which matches the name provided
    MongoDB Enterprise > db.employee.find({name:{$eq:"donghua"}})



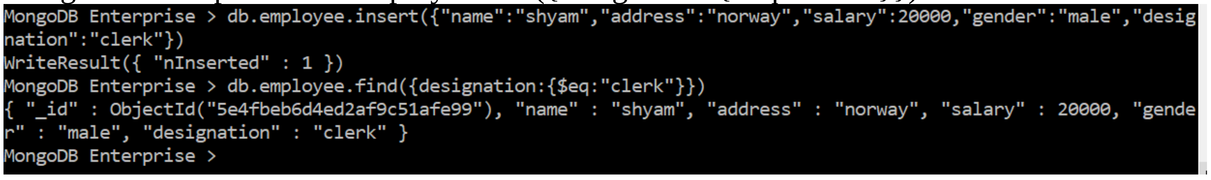
```
MongoDB Enterprise > db.employee.insertMany([ {"name":"donghua","address":"china","salary":340000,"gender":"male
","designation":"devopps"}, {"name":"fengjiu","address":"singapore","salary":230000,"gender":"female","designati
on":"data scientist"}]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5e4fbd22d4ed2af9c51afe96"),
                ObjectId("5e4fbd22d4ed2af9c51afe97")
        ]
}
MongoDB Enterprise > db.employee.find({name:{$eq:"donghua"}})
{ "_id" : ObjectId("5e4fbd22d4ed2af9c51afe96"), "name" : "donghua", "address" : "china", "salary" : 340000, "gen
der" : "male", "designation" : "devopps" }
MongoDB Enterprise >
```

12. Display an employee whose name is "ram"
    MongoDB Enterprise > db.employee.find({name:{$eq:"ram"}})



```
MongoDB Enterprise > db.employee.insert({"name":"ram","address":"mumbai","salary":30000,"gender":"male","designa
tion":"test engineer"})
WriteResult({ "nInserted" : 1 })
MongoDB Enterprise > db.employee.find({name:{$eq:"ram"}})
{ "_id" : ObjectId("5e4fbe08d4ed2af9c51afe98"), "name" : "ram", "address" : "mumbai", "salary" : 30000, "gender"
 : "male", "designation" : "test engineer" }
MongoDB Enterprise >
```

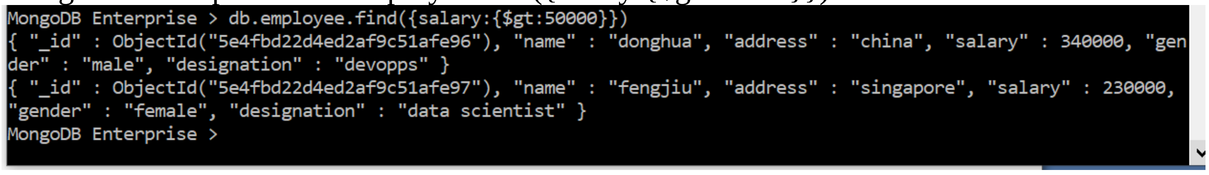
13. Display details of employee whose designation is clerk
    MongoDB Enterprise > db.employee.find({designation:{$eq:"clerk"}})



```
MongoDB Enterprise > db.employee.insert({"name":"shyam","address":"norway","salary":20000,"gender":"male","desig
nation":"clerk"})
WriteResult({ "nInserted" : 1 })
MongoDB Enterprise > db.employee.find({designation:{$eq:"clerk"}})
{ "_id" : ObjectId("5e4fbeb6d4ed2af9c51afe99"), "name" : "shyam", "address" : "norway", "salary" : 20000, "gende
r" : "male", "designation" : "clerk" }
MongoDB Enterprise >
```

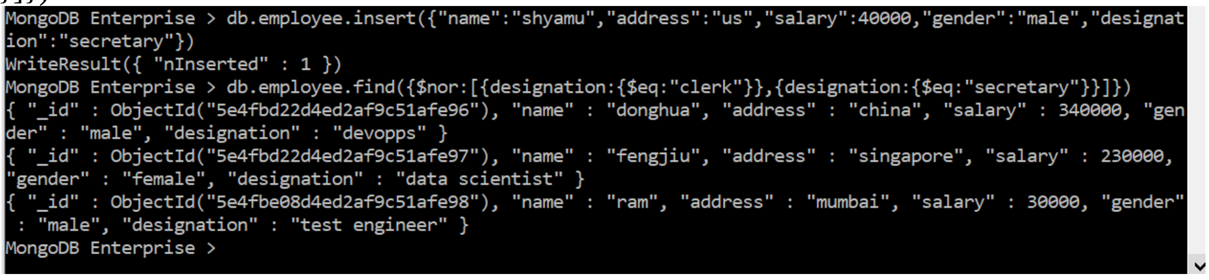
14. Display details who get salary greater than 50000

MongoDB Enterprise > db.employee.find({salary:{$gt:50000}})



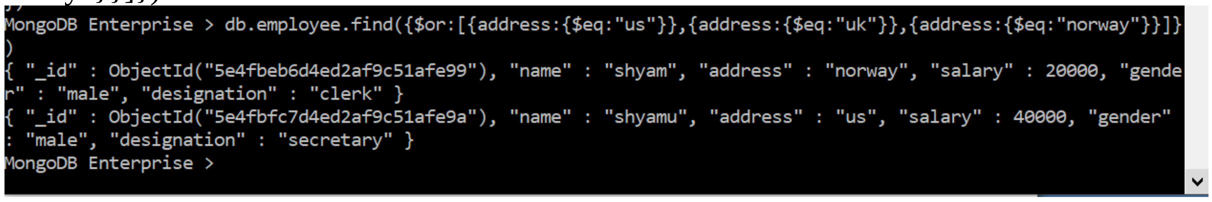
```
MongoDB Enterprise > db.employee.find({salary:{$gt:50000}})
{ "_id" : ObjectId("5e4fbd22d4ed2af9c51afe96"), "name" : "donghua", "address" : "china", "salary" : 340000, "gen
der" : "male", "designation" : "devopps" }
{ "_id" : ObjectId("5e4fbd22d4ed2af9c51afe97"), "name" : "fengjiu", "address" : "singapore", "salary" : 230000,
"gender" : "female", "designation" : "data scientist" }
MongoDB Enterprise >
```

15. Display employees not designation clerk or secretary
    MongoDB Enterprise >
    db.employee.find({$nor:[{designation:{$eq:"clerk"}},{designation:{$eq:"secretary"}
    }]})



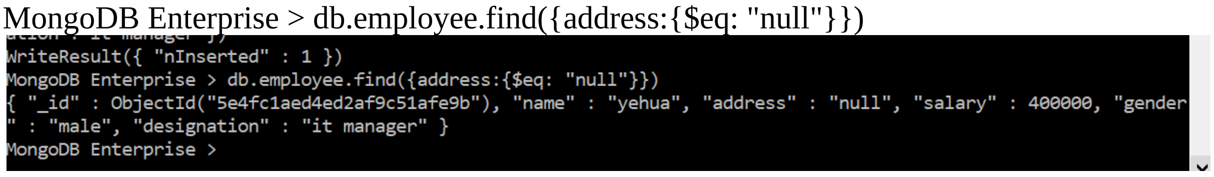
```
MongoDB Enterprise > db.employee.insert({"name":"shyamu","address":"us","salary":40000,"gender":"male","designat
ion":"secretary"})
WriteResult({ "nInserted" : 1 })
MongoDB Enterprise > db.employee.find({$nor:[{designation:{$eq:"clerk"}},{designation:{$eq:"secretary"}}]})
{ "_id" : ObjectId("5e4fbd22d4ed2af9c51afe96"), "name" : "donghua", "address" : "china", "salary" : 340000, "gen
der" : "male", "designation" : "devopps" }
{ "_id" : ObjectId("5e4fbd22d4ed2af9c51afe97"), "name" : "fengjiu", "address" : "singapore", "salary" : 230000,
"gender" : "female", "designation" : "data scientist" }
{ "_id" : ObjectId("5e4fbe08d4ed2af9c51afe98"), "name" : "ram", "address" : "mumbai", "salary" : 30000, "gender"
 : "male", "designation" : "test engineer" }
MongoDB Enterprise >
```

16. Display only employees from us,uk,Norway
    MongoDB Enterprise >
    db.employee.find({$or:[{address:{$eq:"us"}},{address:{$eq:"uk"}},{address:{$eq:"n
    orway"}}]})



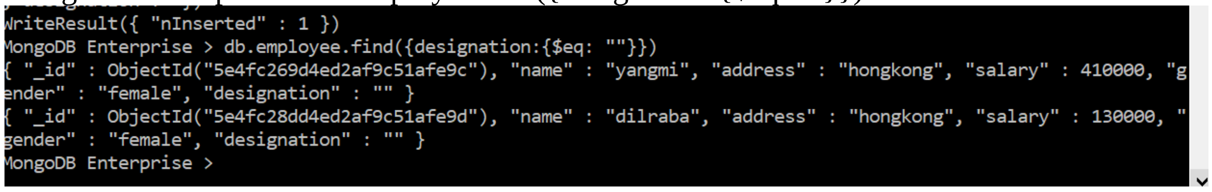
```
MongoDB Enterprise > db.employee.find({$or:[{address:{$eq:"us"}},{address:{$eq:"uk"}},{address:{$eq:"norway"}}]}
)
{ "_id" : ObjectId("5e4fbeb6d4ed2af9c51afe99"), "name" : "shyam", "address" : "norway", "salary" : 20000, "gende
r" : "male", "designation" : "clerk" }
{ "_id" : ObjectId("5e4fbfc7d4ed2af9c51afe9a"), "name" : "shyamu", "address" : "us", "salary" : 40000, "gender"
: "male", "designation" : "secretary" }
MongoDB Enterprise >
```

17. Display all whose address is null
    MongoDB Enterprise > db.employee.find({address:{$eq: "null"}})



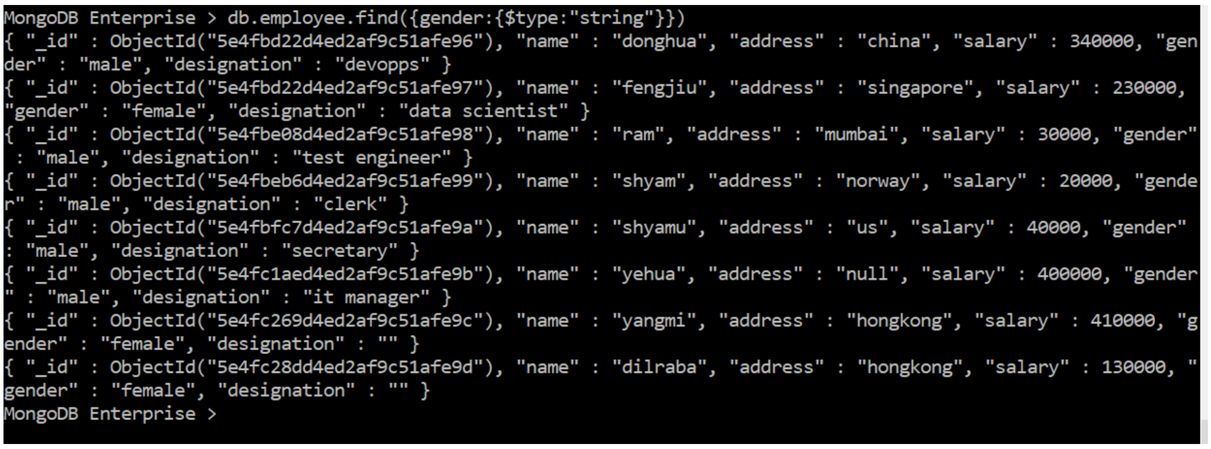
```
WriteResult({ "nInserted" : 1 })
MongoDB Enterprise > db.employee.find({address:{$eq: "null"}})
{ "_id" : ObjectId("5e4fc1aed4ed2af9c51afe9b"), "name" : "yehua", "address" : "null", "salary" : 400000, "gender
" : "male", "designation" : "it manager" }
MongoDB Enterprise >
```

18. Display for designation null
    MongoDB Enterprise > db.employee.find({designation:{$eq: ""}})



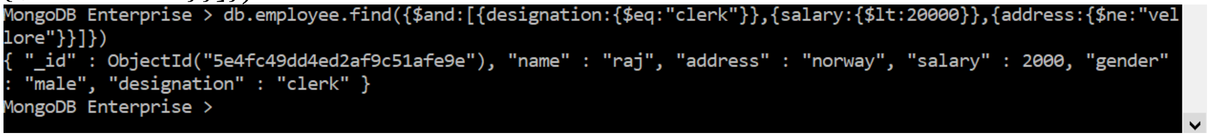
```
WriteResult({ "nInserted" : 1 })
MongoDB Enterprise > db.employee.find({designation:{$eq: ""}})
{ "_id" : ObjectId("5e4fc269d4ed2af9c51afe9c"), "name" : "yangmi", "address" : "hongkong", "salary" : 410000, "g
ender" : "female", "designation" : "" }
{ "_id" : ObjectId("5e4fc28dd4ed2af9c51afe9d"), "name" : "dilraba", "address" : "hongkong", "salary" : 130000, "
gender" : "female", "designation" : "" }
MongoDB Enterprise >
```

19. Gender of type string
    MongoDB Enterprise > db.employee.find({gender:{$type:"string"}})



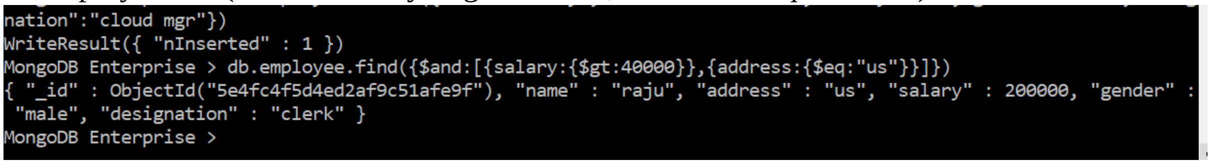
```
MongoDB Enterprise > db.employee.find({gender:{$type:"string"}})
{ "_id" : ObjectId("5e4fbd22d4ed2af9c51afe96"), "name" : "donghua", "address" : "china", "salary" : 340000, "gen
der" : "male", "designation" : "devopps" }
{ "_id" : ObjectId("5e4fbd22d4ed2af9c51afe97"), "name" : "fengjiu", "address" : "singapore", "salary" : 230000,
"gender" : "female", "designation" : "data scientist" }
{ "_id" : ObjectId("5e4fbe08d4ed2af9c51afe98"), "name" : "ram", "address" : "mumbai", "salary" : 30000, "gender"
 : "male", "designation" : "test engineer" }
{ "_id" : ObjectId("5e4fbeb6d4ed2af9c51afe99"), "name" : "shyam", "address" : "norway", "salary" : 20000, "gende
r" : "male", "designation" : "clerk" }
{ "_id" : ObjectId("5e4fbfc7d4ed2af9c51afe9a"), "name" : "shyamu", "address" : "us", "salary" : 40000, "gender"
: "male", "designation" : "secretary" }
{ "_id" : ObjectId("5e4fc1aed4ed2af9c51afe9b"), "name" : "yehua", "address" : "null", "salary" : 400000, "gender
" : "male", "designation" : "it manager" }
{ "_id" : ObjectId("5e4fc269d4ed2af9c51afe9c"), "name" : "yangmi", "address" : "hongkong", "salary" : 410000, "g
ender" : "female", "designation" : "" }
{ "_id" : ObjectId("5e4fc28dd4ed2af9c51afe9d"), "name" : "dilraba", "address" : "hongkong", "salary" : 130000, "
gender" : "female", "designation" : "" }
MongoDB Enterprise >
```

20. Display employees salary less than 20000 and not from vellore
    MongoDB Enterprise >
    db.employee.find({$and:[{designation:{$eq:"clerk"}},{salary:{$lt:20000}},{address:{$ne:"vellore"}}]})



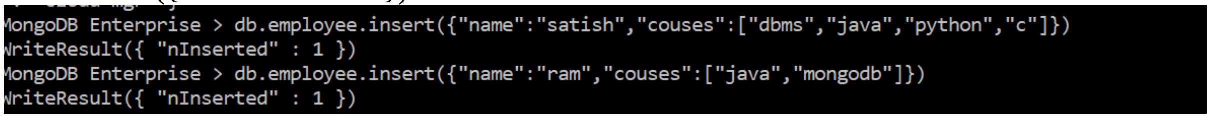
```
MongoDB Enterprise > db.employee.find({$and:[{designation:{$eq:"clerk"}},{salary:{$lt:20000}},{address:{$ne:"vel
lore"}}]})
{ "_id" : ObjectId("5e4fc49dd4ed2af9c51afe9e"), "name" : "raj", "address" : "norway", "salary" : 2000, "gender"
: "male", "designation" : "clerk" }
MongoDB Enterprise >
```

21. MongoDB Enterprise >
    db.employee.find({$and:[{salary:{$gt:40000}},{address:{$eq:"us"}}]})



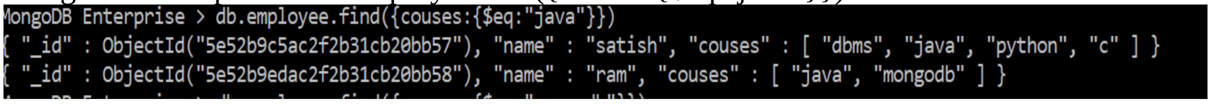
```
nation":"cloud mgr"})
WriteResult({ "nInserted" : 1 })
MongoDB Enterprise > db.employee.find({$and:[{salary:{$gt:40000}},{address:{$eq:"us"}}]})
{ "_id" : ObjectId("5e4fc4f5d4ed2af9c51afe9f"), "name" : "raju", "address" : "us", "salary" : 200000, "gender" :
 "male", "designation" : "clerk" }
MongoDB Enterprise >
```

22. Insert the following documents in the employee collection
    MongoDB Enterprise >
    db.employee.insert({"name":"satish","couses":["dbms","java","python","c"]})
    WriteResult({ "nInserted" : 1 })
    MongoDB Enterprise >
    db.employee.insert({"name":"ram","couses":["java","mongodb"]})
    WriteResult({ "nInserted" : 1 })



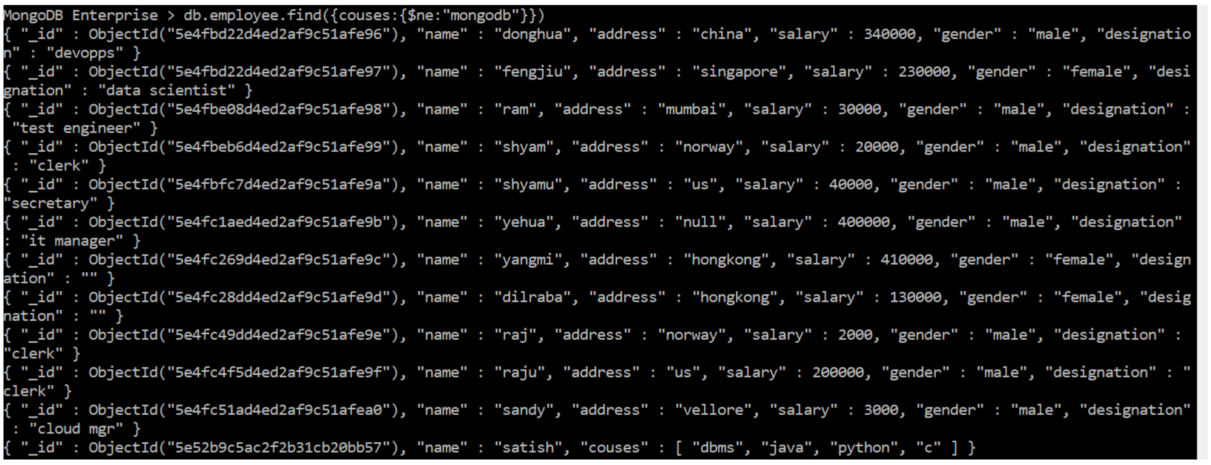
```
MongoDB Enterprise > db.employee.insert({"name":"satish","couses":["dbms","java","python","c"]})
WriteResult({ "nInserted" : 1 })
MongoDB Enterprise > db.employee.insert({"name":"ram","couses":["java","mongodb"]})
WriteResult({ "nInserted" : 1 })
```

23. Display all employees who teach Java.
    MongoDB Enterprise > db.employee.find({couses:{$eq:"java"}})



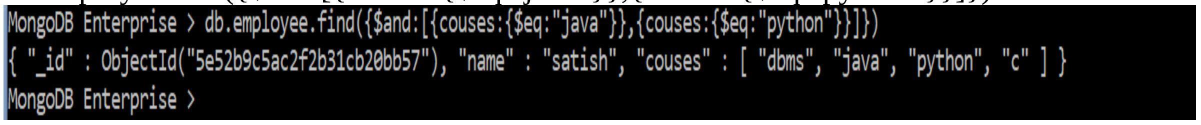
```
MongoDB Enterprise > db.employee.find({couses:{$eq:"java"}})
{ "_id" : ObjectId("5e52b9c5ac2f2b31cb20bb57"), "name" : "satish", "couses" : [ "dbms", "java", "python", "c" ] }
{ "_id" : ObjectId("5e52b9edac2f2b31cb20bb58"), "name" : "ram", "couses" : [ "java", "mongodb" ] }
```

24. Display employees who are not teaching MongoDB.
    MongoDB Enterprise > db.employee.find({couses:{$ne:"mongodb"}})



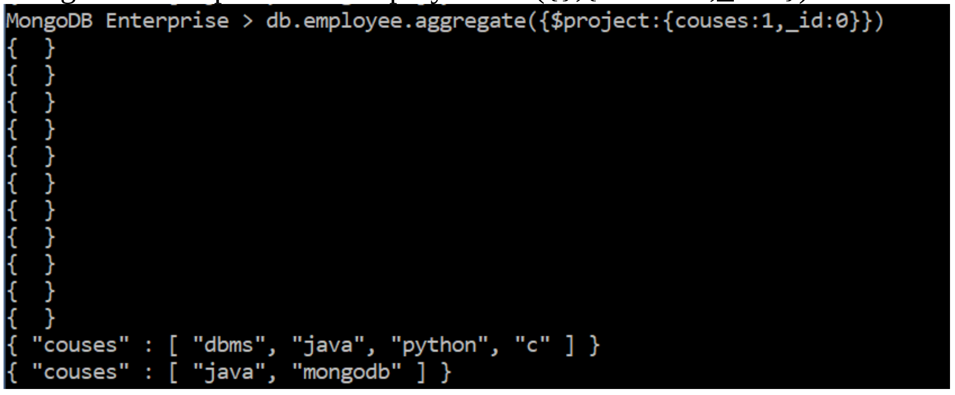
```
MongoDB Enterprise > db.employee.find({couses:{$ne:"mongodb"}})
{ "_id" : ObjectId("5e4fbd22d4ed2af9c51afe96"), "name" : "donghua", "address" : "china", "salary" : 340000, "gender" : "male", "designatio
n" : "devopps" }
{ "_id" : ObjectId("5e4fbd22d4ed2af9c51afe97"), "name" : "fengjiu", "address" : "singapore", "salary" : 230000, "gender" : "female", "desi
gnation" : "data scientist" }
{ "_id" : ObjectId("5e4fbe08d4ed2af9c51afe98"), "name" : "ram", "address" : "mumbai", "salary" : 30000, "gender" : "male", "designation" :
 "test engineer" }
{ "_id" : ObjectId("5e4fbeb6d4ed2af9c51afe99"), "name" : "shyam", "address" : "norway", "salary" : 20000, "gender" : "male", "designation"
 : "clerk" }
{ "_id" : ObjectId("5e4fbfc7d4ed2af9c51afe9a"), "name" : "shyamu", "address" : "us", "salary" : 40000, "gender" : "male", "designation" :
"secretary" }
{ "_id" : ObjectId("5e4fc1aed4ed2af9c51afe9b"), "name" : "yehua", "address" : "null", "salary" : 400000, "gender" : "male", "designation"
: "it manager" }
{ "_id" : ObjectId("5e4fc269d4ed2af9c51afe9c"), "name" : "yangmi", "address" : "hongkong", "salary" : 410000, "gender" : "female", "design
ation" : "" }
{ "_id" : ObjectId("5e4fc28dd4ed2af9c51afe9d"), "name" : "dilraba", "address" : "hongkong", "salary" : 130000, "gender" : "female", "desig
nation" : "" }
{ "_id" : ObjectId("5e4fc49dd4ed2af9c51afe9e"), "name" : "raj", "address" : "norway", "salary" : 2000, "gender" : "male", "designation" :
"clerk" }
{ "_id" : ObjectId("5e4fc4f5d4ed2af9c51afe9f"), "name" : "raju", "address" : "us", "salary" : 200000, "gender" : "male", "designation" : "
clerk" }
{ "_id" : ObjectId("5e4fc51ad4ed2af9c51afea0"), "name" : "sandy", "address" : "vellore", "salary" : 3000, "gender" : "male", "designation"
 : "cloud mgr" }
{ "_id" : ObjectId("5e52b9c5ac2f2b31cb20bb57"), "name" : "satish", "couses" : [ "dbms", "java", "python", "c" ] }
```

25. Display employees who teach Java and Python.
    MongoDB Enterprise >
    db.employee.find({$and:[{couses:{$eq:"java"}},{couses:{$eq:"python"}}]})



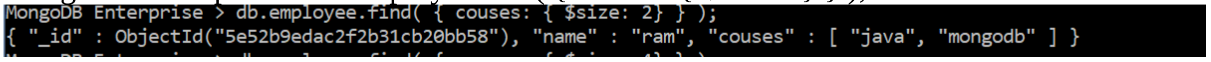
```
MongoDB Enterprise > db.employee.find({$and:[{couses:{$eq:"java"}},{couses:{$eq:"python"}}]})
{ "_id" : ObjectId("5e52b9c5ac2f2b31cb20bb57"), "name" : "satish", "couses" : [ "dbms", "java", "python", "c" ] }
MongoDB Enterprise >
```

26. Display the employees who teach 2 subjects.
    MongoDB Enterprise > db.employee.find({},{couses:2,_id:0})



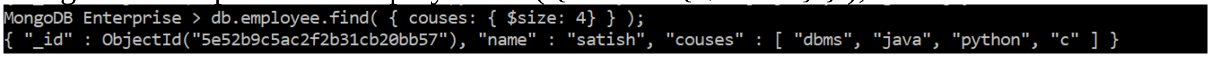
```
MongoDB Enterprise > db.employee.aggregate({$project:{couses:1,_id:0}})
{  }
{  }
{  }
{  }
{  }
{  }
{  }
{  }
{  }
{  }
{  }
{ "couses" : [ "dbms", "java", "python", "c" ] }
{ "couses" : [ "java", "mongodb" ] }
```

27. Match an array exactly $size
    MongoDB Enterprise > db.employee.find({ couses: { $size: 2} });



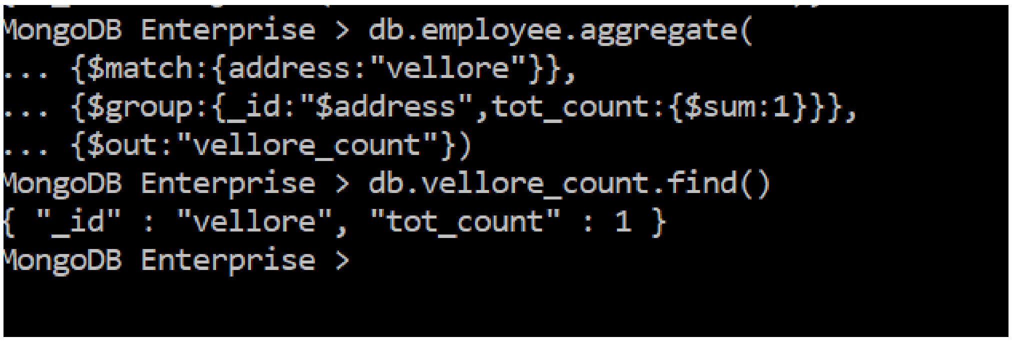
```
MongoDB Enterprise > db.employee.find( { couses: { $size: 2} } );
{ "_id" : ObjectId("5e52b9edac2f2b31cb20bb58"), "name" : "ram", "couses" : [ "java", "mongodb" ] }
```

28. Match an array irrespective of the order
    MongoDB Enterprise > db.employee.find({ couses: { $size: 4} });



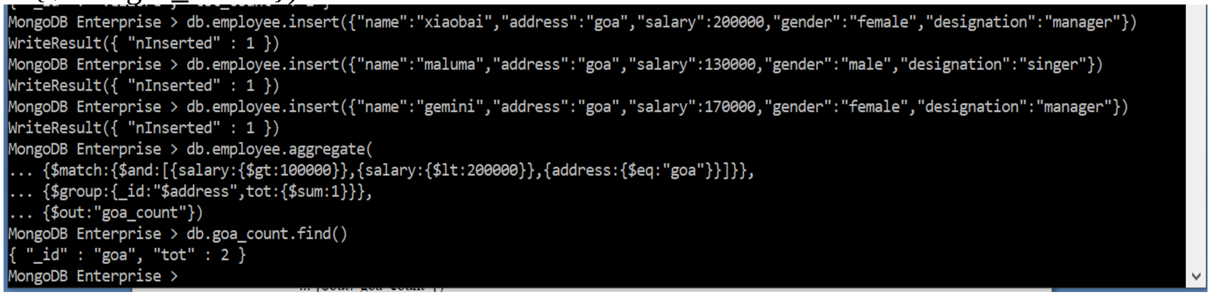
```
MongoDB Enterprise > db.employee.find( { couses: { $size: 4} } );
{ "_id" : ObjectId("5e52b9c5ac2f2b31cb20bb57"), "name" : "satish", "couses" : [ "dbms", "java", "python", "c" ] }
```

29. Write the address and total count of employees from vellore to a collection by name vellore_count
    MongoDB Enterprise > db.employee.aggregate(
    ... {$match:{address:"vellore"}},
    ... {$group:{_id:"$address",tot_count:{$sum:1}}},
    ... {$out:"vellore_count"})



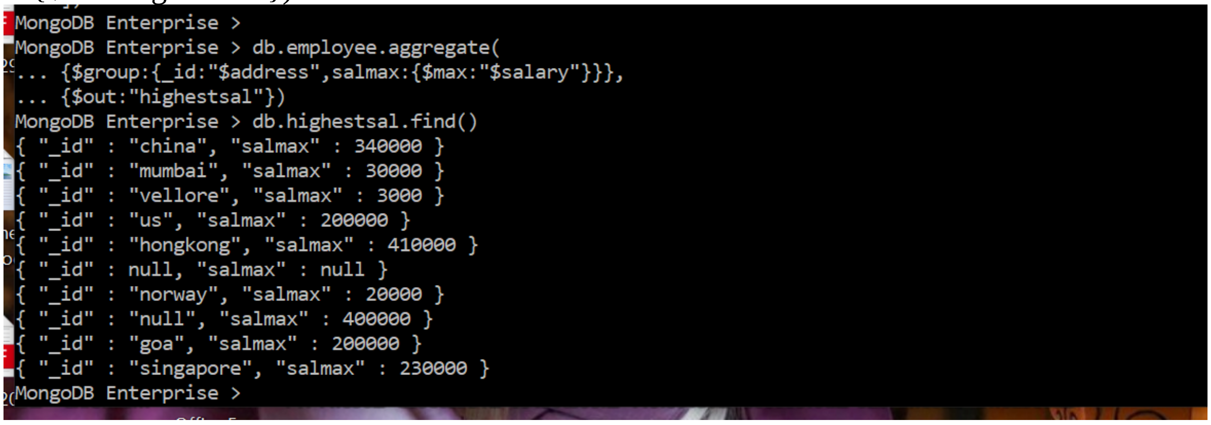
```
MongoDB Enterprise > db.employee.aggregate(
... {$match:{address:"vellore"}},
... {$group:{_id:"$address",tot_count:{$sum:1}}},
... {$out:"vellore_count"})
MongoDB Enterprise > db.vellore_count.find()
{ "_id" : "vellore", "tot_count" : 1 }
MongoDB Enterprise >
```

30. Write the total count of employees who are from goa and who earn a salary greater than 100000 and less than 200000 to a collection by name goa_details
    MongoDB Enterprise > db.employee.aggregate(
    ...
    {$match:{$and:[{salary:{$gt:100000}},{salary:{$lt:200000}},{address:{$eq:"goa"}}]}},
    ... {$group:{_id:"$address",tot:{$sum:1}}},
    ... {$out:"goa_count"})



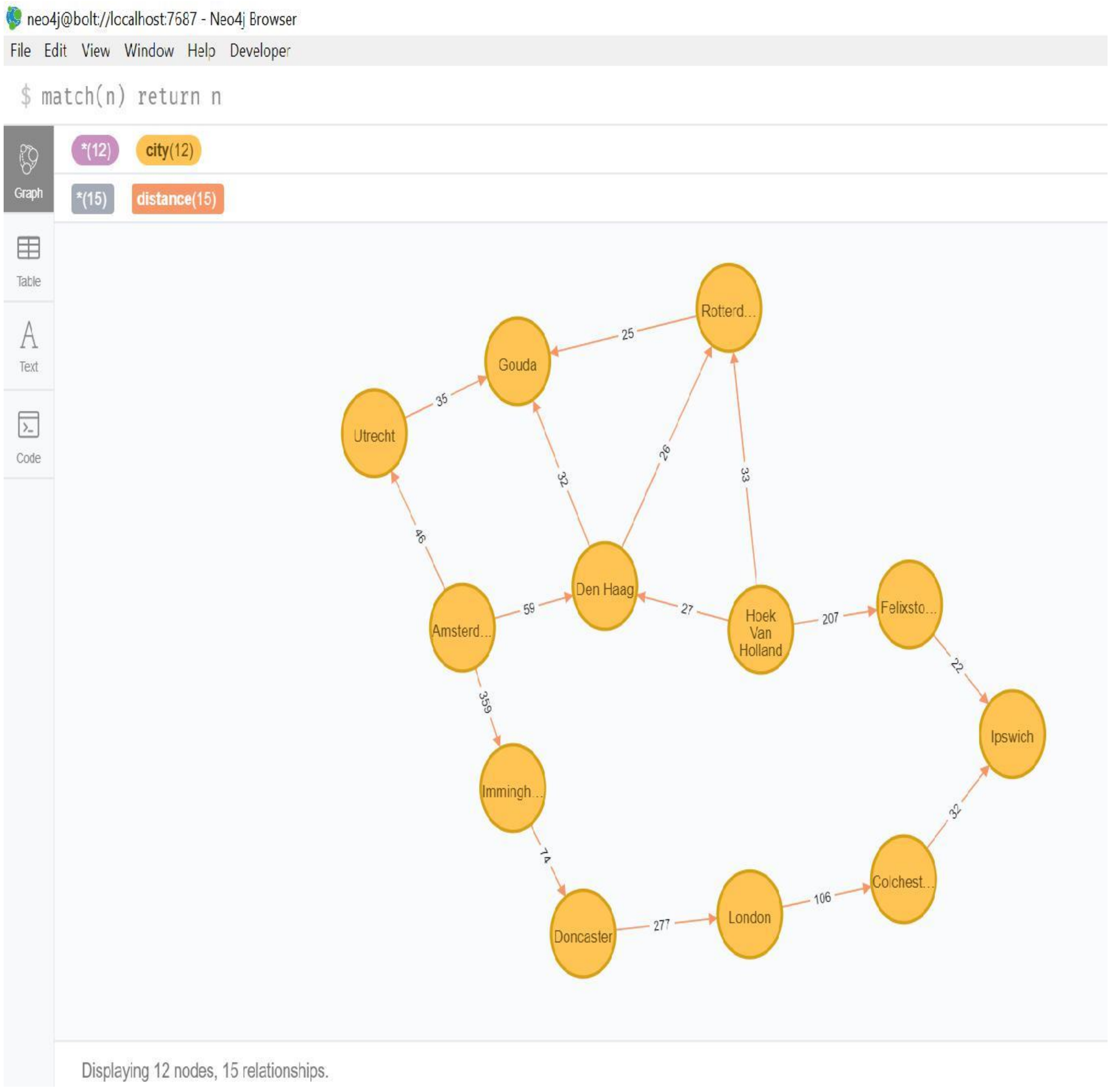
```
MongoDB Enterprise > db.employee.insert({"name":"xiaobai","address":"goa","salary":200000,"gender":"female","designation":"manager"})
WriteResult({ "nInserted" : 1 })
MongoDB Enterprise > db.employee.insert({"name":"maluma","address":"goa","salary":130000,"gender":"male","designation":"singer"})
WriteResult({ "nInserted" : 1 })
MongoDB Enterprise > db.employee.insert({"name":"gemini","address":"goa","salary":170000,"gender":"female","designation":"manager"})
WriteResult({ "nInserted" : 1 })
MongoDB Enterprise > db.employee.aggregate(
... {$match:{$and:[{salary:{$gt:100000}},{salary:{$lt:200000}},{address:{$eq:"goa"}}]}},
... {$group:{_id:"$address",tot:{$sum:1}}},
... {$out:"goa_count"})
MongoDB Enterprise > db.goa_count.find()
{ "_id" : "goa", "tot" : 2 }
MongoDB Enterprise >
```

31. write the address and maximum salary of all employees from each city to a collection by name highestsalary
    MongoDB Enterprise > db.employee.aggregate(
    ... {$group:{_id:"$address",salmax:{$max:"$salary"}}},
    ... {$out:"highestsal"})



```
MongoDB Enterprise >
MongoDB Enterprise > db.employee.aggregate(
... {$group:{_id:"$address",salmax:{$max:"$salary"}}},
... {$out:"highestsal"})
MongoDB Enterprise > db.highestsal.find()
{ "_id" : "china", "salmax" : 340000 }
{ "_id" : "mumbai", "salmax" : 30000 }
{ "_id" : "vellore", "salmax" : 3000 }
{ "_id" : "us", "salmax" : 200000 }
{ "_id" : "hongkong", "salmax" : 410000 }
{ "_id" : null, "salmax" : null }
{ "_id" : "norway", "salmax" : 20000 }
{ "_id" : "null", "salmax" : 400000 }
{ "_id" : "goa", "salmax" : 200000 }
{ "_id" : "singapore", "salmax" : 230000 }
MongoDB Enterprise >
```

Neo4j

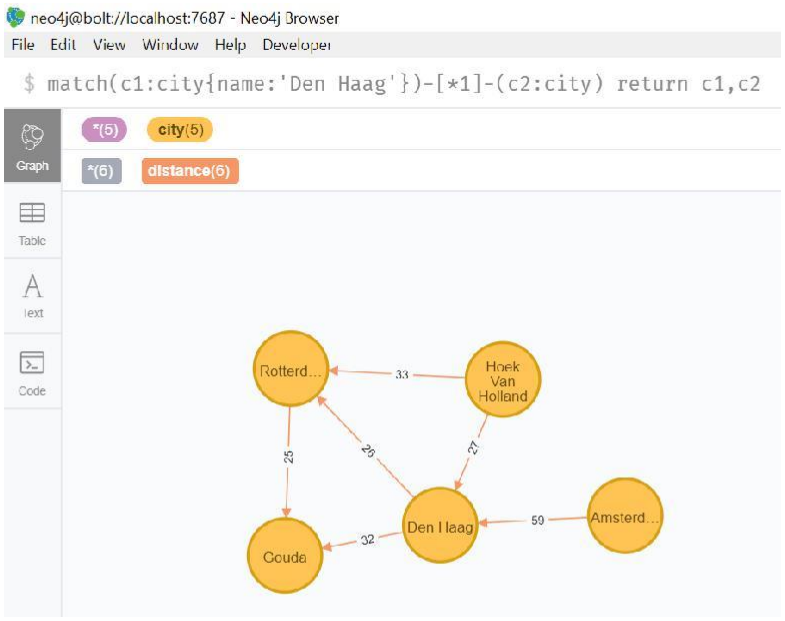


```
$ match(n) return n
```

Displaying 12 nodes, 15 relationships.

1) Determine all the closest cities (just one hop) away from the Den Haag.

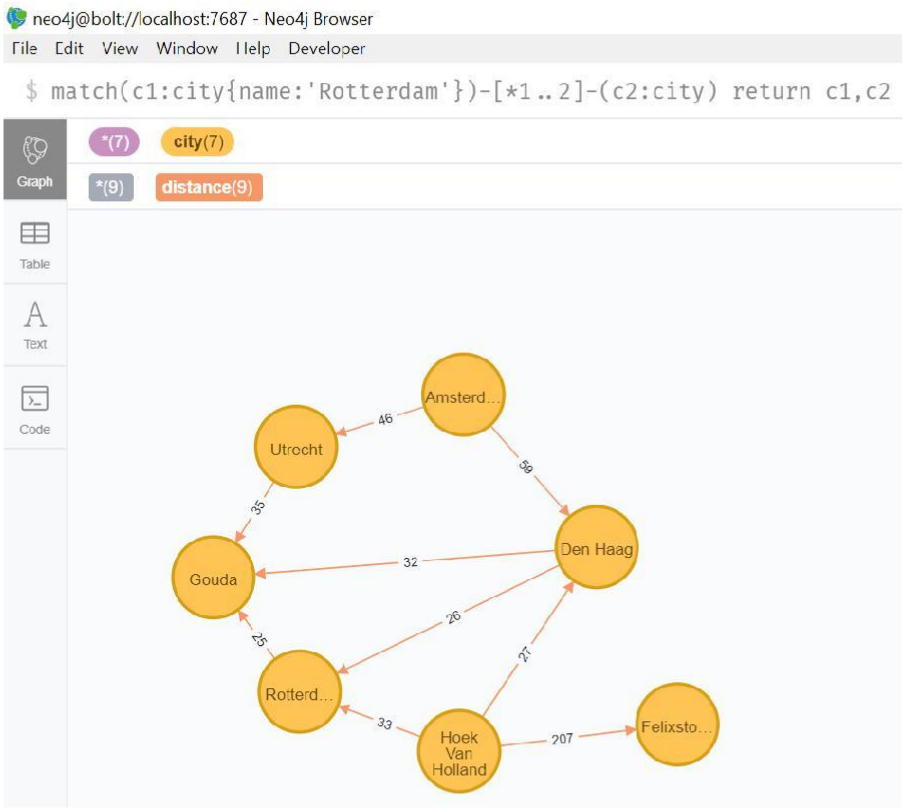
match(c1:city{name:'Den Haag'})-[*1]-(c2:city) return c1,c2



```
$ match(c1:city{name:'Den Haag'})-[*1]-(c2:city) return c1,c2
```

2) Determine all the cities that are one or two hops away from the City Rotterdam

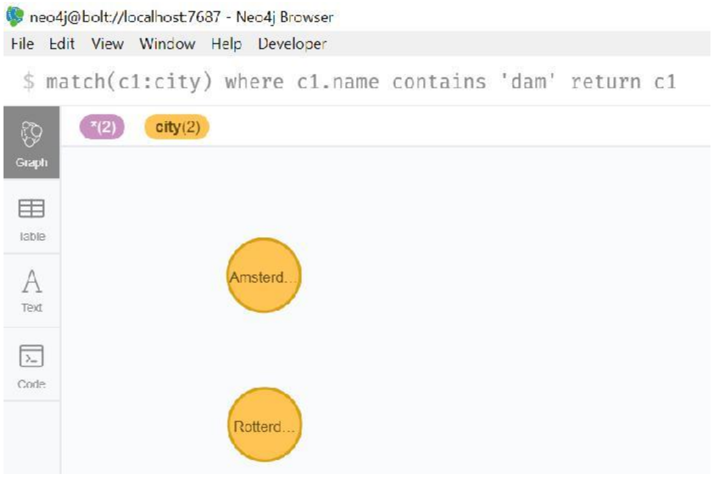
match(c1:city{name:'Rotterdam'})-[*1..2]-(c2:city) return c1,c2



```
$ match(c1:city{name:'Rotterdam'})-[*1..2]-(c2:city) return c1,c2
```

3) Display all the cities that contain dam in their name

match(c1:city) where c1.name contains 'dam' return c1



```
$ match(c1:city) where c1.name contains 'dam' return c1
```

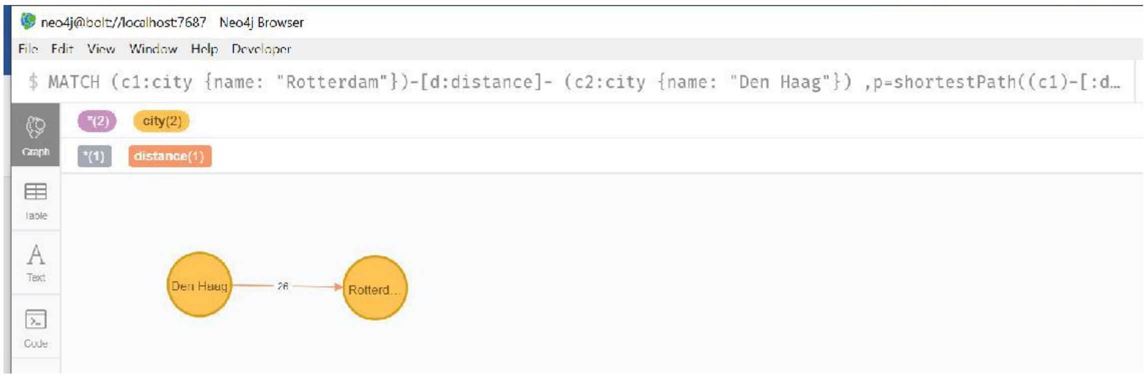
neo4j@bolt://localhost:7687 - Neo4j Browser
File Edit View Window Help Developer
*(2)
city(2)
Graph
Table
Text
Code
Amsterd...
Rotterd...

4) Determine the shortest path using number of hops between Rotterdam and Den Haag

MATCH (c1:city {name: "Rotterdam"})-[d:distance]- (c2:city {name: "Den Haag"}),

p=shortestPath((c1)-[:distance]-(c2))

RETURN p



```
$ MATCH (c1:city {name: "Rotterdam"})-[d:distance]- (c2:city {name: "Den Haag"}) ,p=shortestPath((c1)-[:d…
```

neo4j@bolt://localhost:7687 Neo4j Browser
File Edit View Window Help Developer
*(2)
city(2)
*(1)
distance(1)
Graph
Table
Text
Code
Den Haag
26
Rotterd...

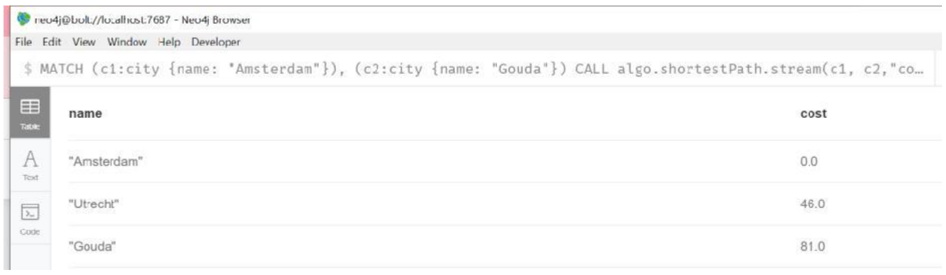
5) Determine the shortest path (using the weights) between Amsterdam and Gouda

MATCH (c1:city {name: "Amsterdam"}), (c2:city {name: "Gouda"})

CALL algo.shortestPath.stream(c1, c2,"cost")

YIELD nodeId, cost

RETURN algo.getNodeById(nodeId).name AS name, cost



```
$ MATCH (c1:city {name: "Amsterdam"}), (c2:city {name: "Gouda"}) CALL algo.shortestPath.stream(c1, c2,"co…
```

name	cost
"Amsterdam"	0.0
"Utrecht"	46.0
"Gouda"	81.0

neo4j@bolt://localhost:7687 - Neo4j Browser
File Edit View Window Help Developer
Table
Text
Code

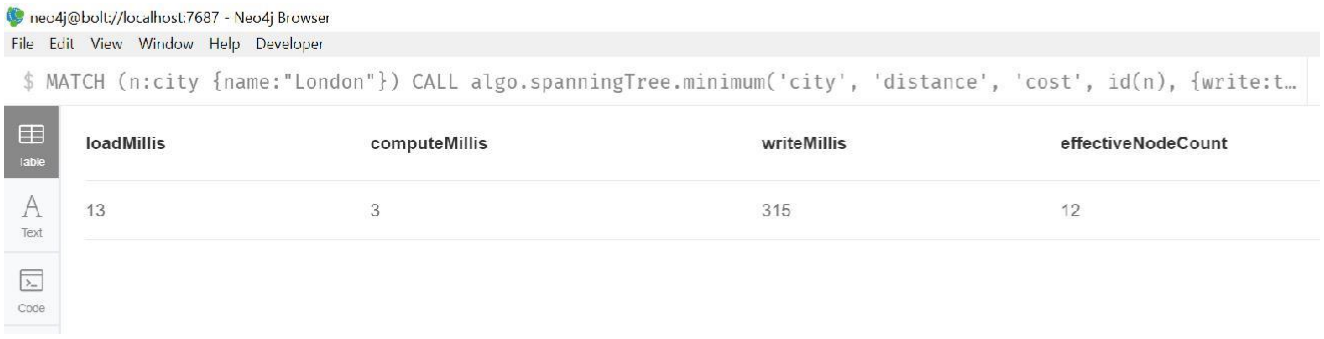
6) Determine how a visitor from London visit all the cities by covering a minimum distance

MATCH (n:city {name:"London"})

CALL algo.spanningTree.minimum('city', 'distance', 'cost', id(n), {write:true, writeProperty:"MINST"})

YIELD loadMillis, computeMillis, writeMillis, effectiveNodeCount

RETURN loadMillis, computeMillis, writeMillis, effectiveNodeCount;


neo4j@bolt://localhost:7687 - Neo4j Browser
File Edit View Window Help Developer
Table
Text
Code

```
$ MATCH (n:city {name:"London"}) CALL algo.spanningTree.minimum('city', 'distance', 'cost', id(n), {write:t…
```

loadMillis	computeMillis	writeMillis	effectiveNodeCount
13	3	315	12

<u>query minimum spanning tree</u>

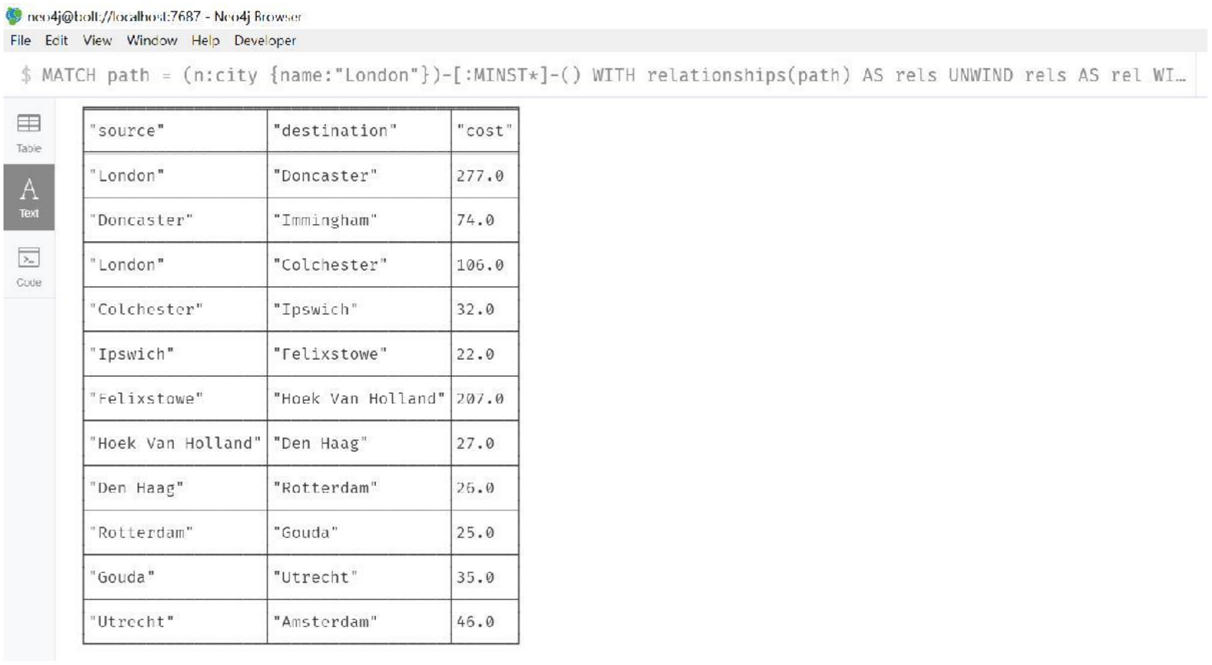
MATCH path = (n:city {name:"London"})-[:MINST*]-()

WITH relationships(path) AS rels

UNWIND rels AS rel

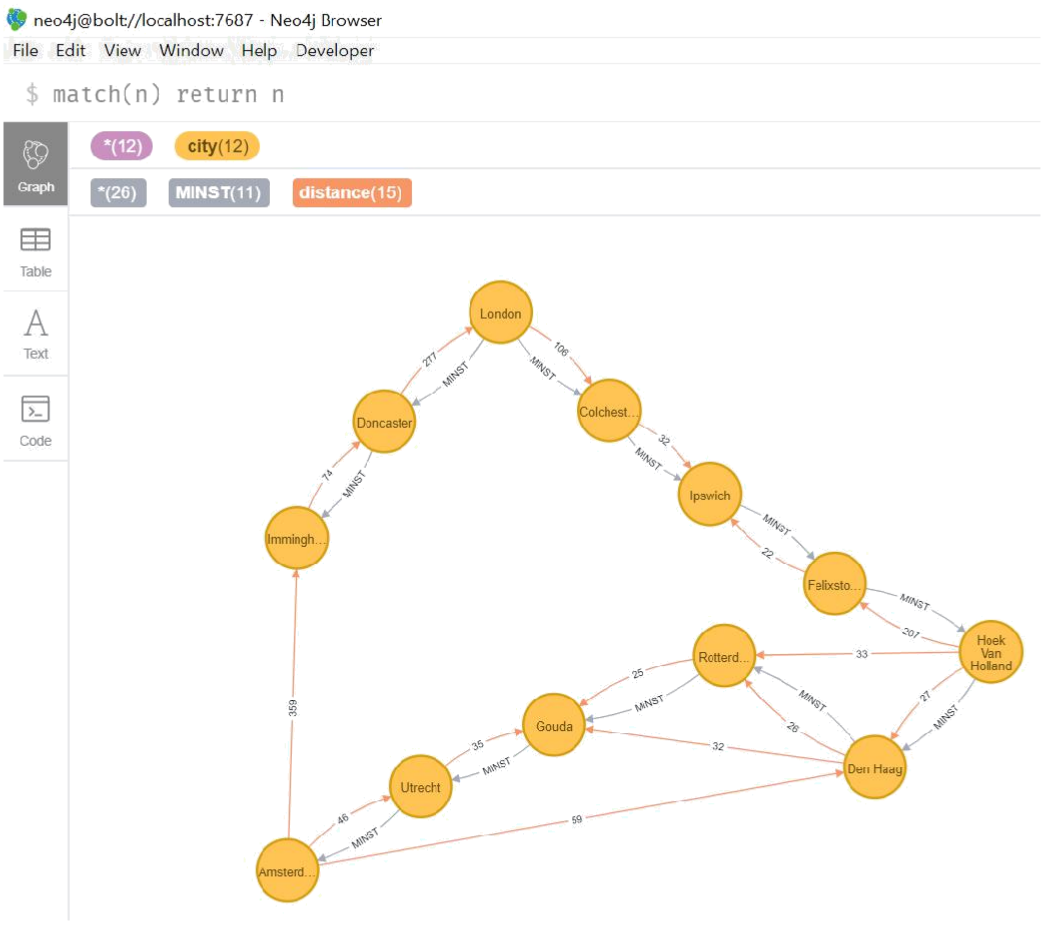
WITH DISTINCT rel AS rel

RETURN startNode(rel).name AS source, endNode(rel).name AS destination, rel.cost AS cost


neo4j@bolt://localhost:7687 - Neo4j Browser
File Edit View Window Help Developer
Table
Text
Code

```
$ MATCH path = (n:city {name:"London"})-[:MINST*]-() WITH relationships(path) AS rels UNWIND rels AS rel WI…
```

"source"	"destination"	"cost"
"London"	"Doncaster"	277.0
"Doncaster"	"Immingham"	74.0
"London"	"Colchester"	106.0
"Colchester"	"Ipswich"	32.0
"Ipswich"	"Felixstowe"	22.0
"Felixstowe"	"Hoek Van Holland"	207.0
"Hoek Van Holland"	"Den Haag"	27.0
"Den Haag"	"Rotterdam"	26.0
"Rotterdam"	"Gouda"	25.0
"Gouda"	"Utrecht"	35.0
"Utrecht"	"Amsterdam"	46.0


neo4j@bolt://localhost:7687 - Neo4j Browser

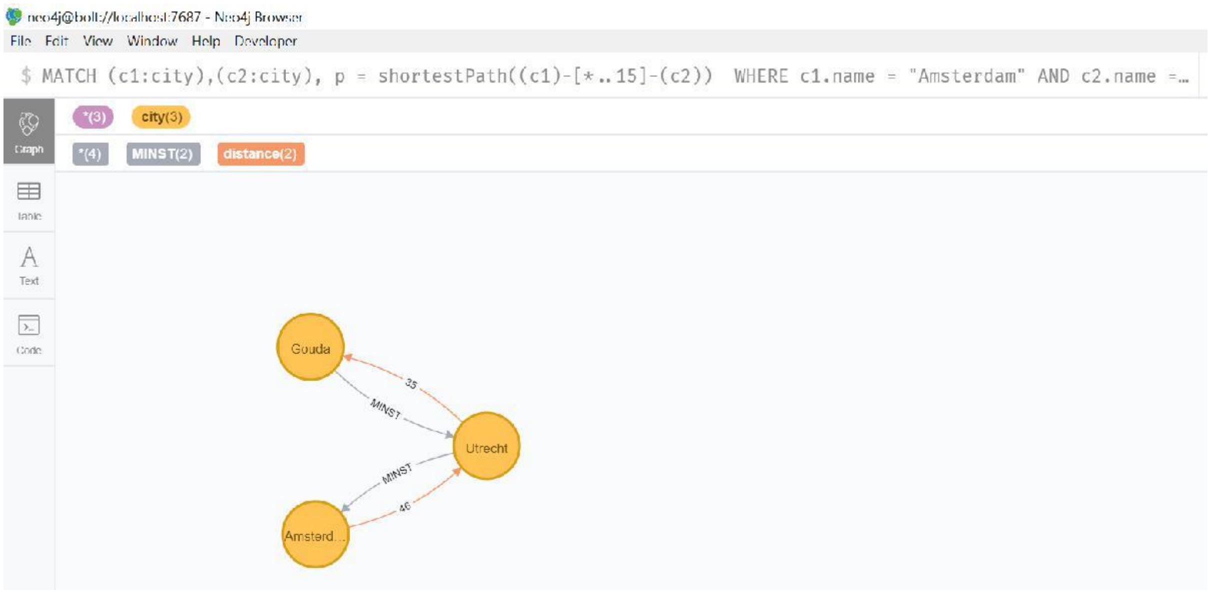
```
$ match(n) return n
```

7) Determine all the shortest paths between Amsterdam and Gouda

MATCH (c1:city),(c2:city),p = shortestPath((c1)-[*..15]-(c2))

WHERE c1.name = "Amsterdam" AND c2.name = "Gouda"

RETURN p

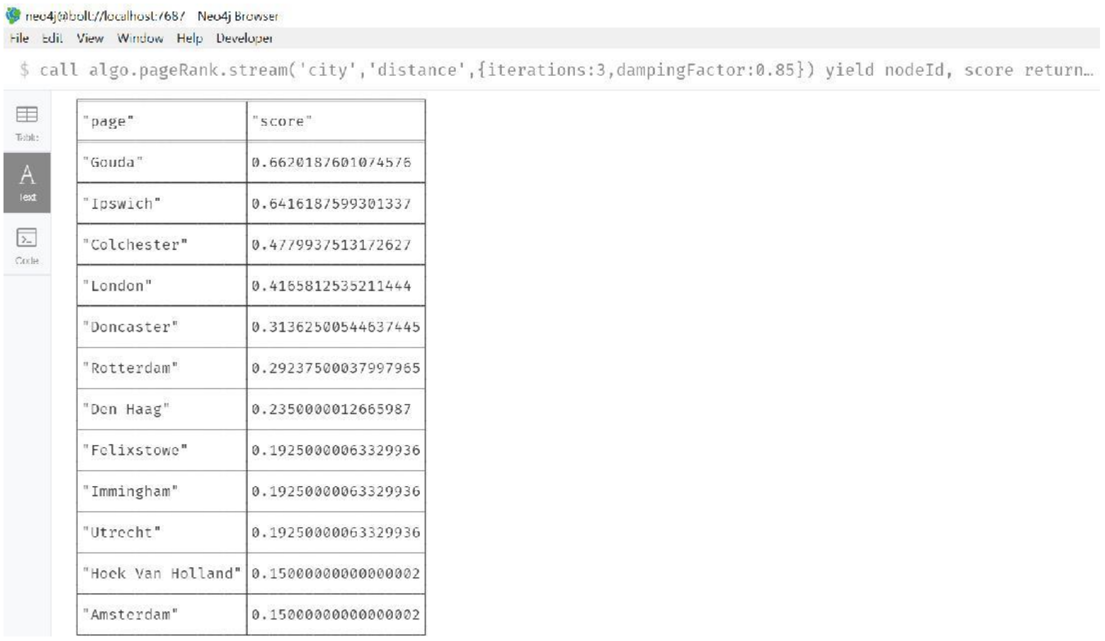

neo4j@bolt://localhost:7687 - Neo4j Browser

```
$ MATCH (c1:city),(c2:city), p = shortestPath((c1)-[*..15]-(c2)) WHERE c1.name = "Amsterdam" AND c2.name =…
```

8) Determine the shortest paths between all pairs of nodes in the graph

call algo.pageRank.stream('city','distance',{iterations:3,dampingFactor:0.85})

yield nodeId, score return algo.getNodeById(nodeId).name as page,score order by score desc



neo4j@bolt://localhost:7687 Neo4j Browser

```
$ call algo.pageRank.stream('city','distance',{iterations:3,dampingFactor:0.85}) yield nodeId, score return…
```

"page"	"score"
"Gouda"	0.6620187601074576
"Ipswich"	0.6416187599301337
"Colchester"	0.4779937513172627
"London"	0.4165812535211444
"Doncaster"	0.31362500544637445
"Rotterdam"	0.29237500037997965
"Den Haag"	0.2350000012665987
"Felixstowe"	0.19250000063329936
"Immingham"	0.19250000063329936
"Utrecht"	0.19250000063329936
"Hoek Van Holland"	0.15000000000000002
"Amsterdam"	0.15000000000000002

9) Compute the PageRank for the nodes in the graph (3 iterations).

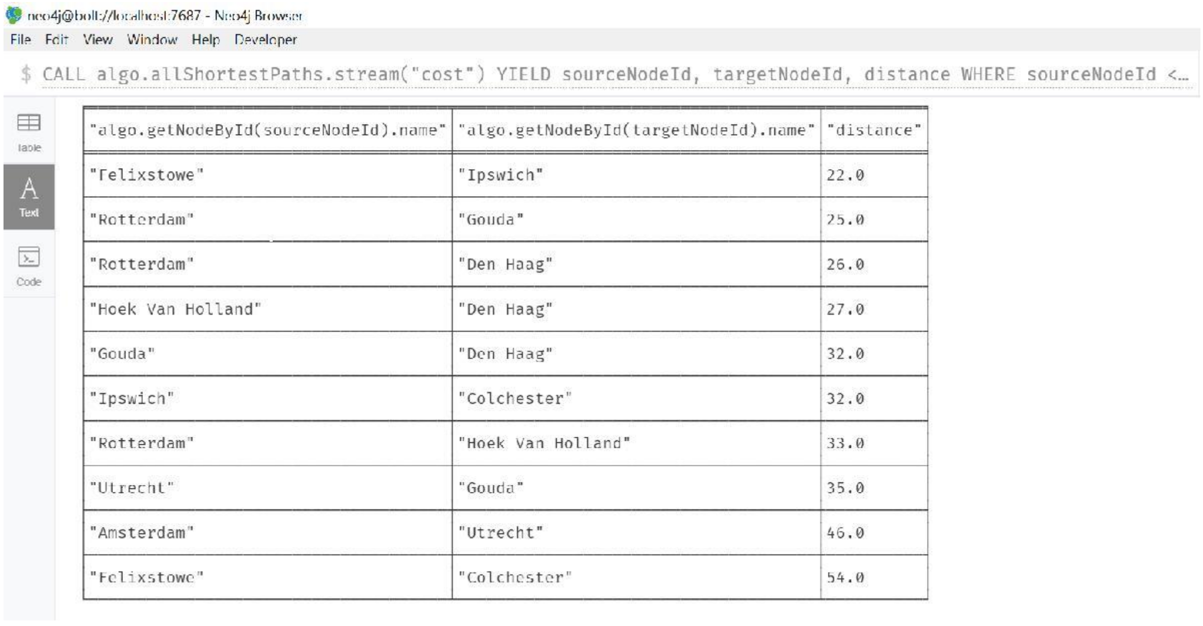
CALL algo.allShortestPaths.stream("cost")

YIELD sourceNodeId, targetNodeId, distance

WHERE sourceNodeId < targetNodeId

RETURN algo.getNodeById(sourceNodeId).name, algo.getNodeById(targetNodeId).name, distance

ORDER BY distance Limit 10



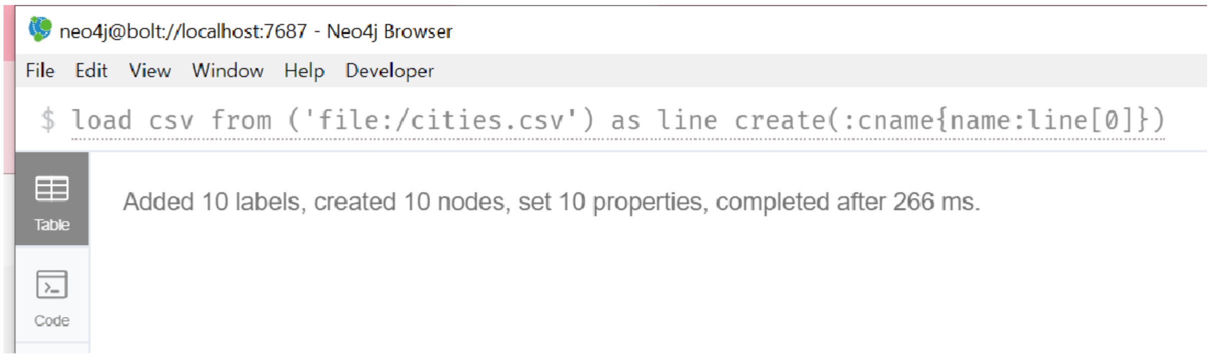
neo4j@bolt://localhost:7687 - Neo4j Browser

```
$ CALL algo.allShortestPaths.stream("cost") YIELD sourceNodeId, targetNodeId, distance WHERE sourceNodeId <…
```

"algo.getNodeById(sourceNodeId).name"	"algo.getNodeById(targetNodeId).name"	"distance"
"Felixstowe"	"Ipswich"	22.0
"Rotterdam"	"Gouda"	25.0
"Rotterdam"	"Den Haag"	26.0
"Hoek Van Holland"	"Den Haag"	27.0
"Gouda"	"Den Haag"	32.0
"Ipswich"	"Colchester"	32.0
"Rotterdam"	"Hoek Van Holland"	33.0
"Utrecht"	"Gouda"	35.0
"Amsterdam"	"Utrecht"	46.0
"Felixstowe"	"Colchester"	54.0

10) Load a CSV file containing names of few more cities in the database.

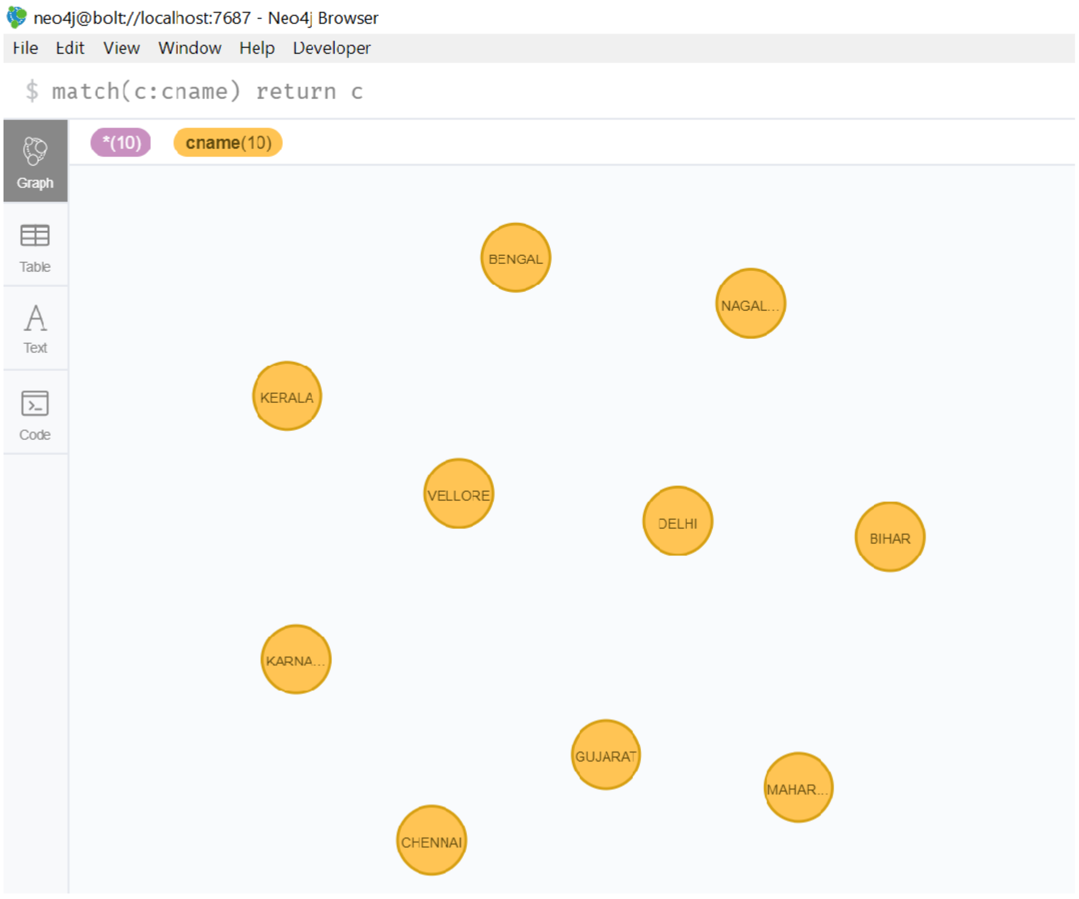
load csv from ('file:/cities.csv') as line create(:cname{name:line[0]})



neo4j@bolt://localhost:7687 - Neo4j Browser
File Edit View Window Help Developer
Table
Code

```
$ load csv from ('file:/cities.csv') as line create(:cname{name:line[0]})
```

```
Added 10 labels, created 10 nodes, set 10 properties, completed after 266 ms.
```



```
$ match(c:cname) return c
```