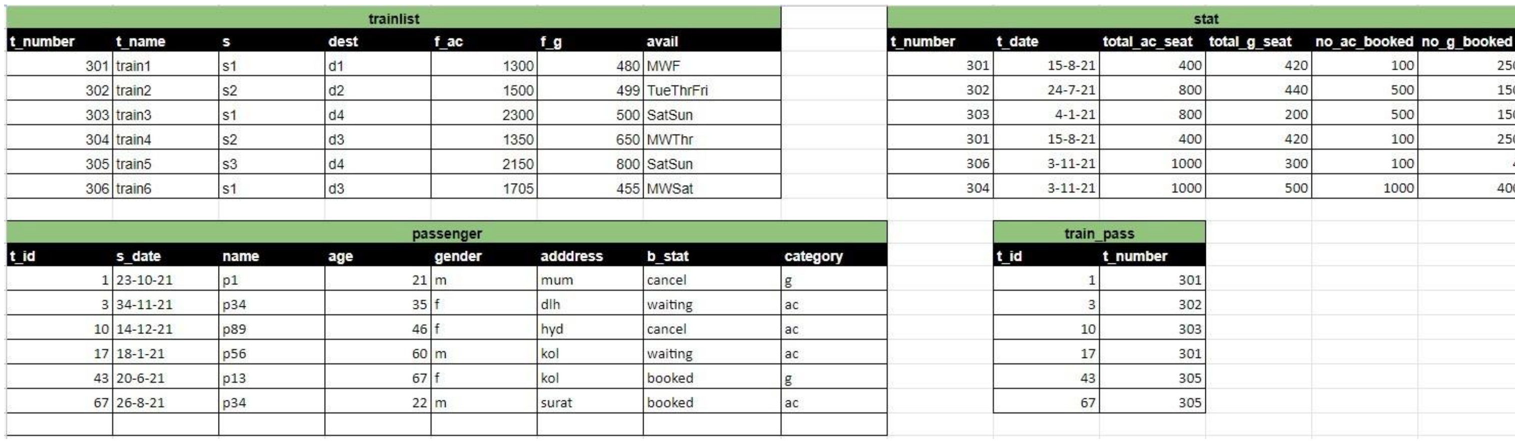
SESSION 14 CASE STUDY

Make a database for Railway Reservation System.



trainlist						
t_number	t_name	s	dest	f_ac	f_g	avail
301	train1	s1	d1	1300	480	MWF
302	train2	s2	d2	1500	499	TueThrFri
303	train3	s1	d4	2300	500	SatSun
304	train4	s2	d3	1350	650	MWThr
305	train5	s3	d4	2150	800	SatSun
306	train6	s1	d3	1705	455	MWSat

stat					
t_number	t_date	total_ac_seat	total_g_seat	no_ac_booked	no_g_booked
301	15-8-21	400	420	100	25
302	24-7-21	800	440	500	15
303	4-1-21	800	200	500	15
301	15-8-21	400	420	100	25
306	3-11-21	1000	300	100	4
304	3-11-21	1000	500	1000	40

passenger							
t_id	s_date	name	age	gender	adddress	b_stat	category
1	23-10-21	p1	21	m	mum	cancel	g
3	34-11-21	p34	35	f	dlh	waiting	ac
10	14-12-21	p89	46	f	hyd	cancel	ac
17	18-1-21	p56	60	m	kol	waiting	ac
43	20-6-21	p13	67	f	kol	booked	g
67	26-8-21	p34	22	m	surat	booked	ac

train_pass	
t_id	t_number
1	301
3	302
10	303
17	301
43	305
67	305

Q. Display a passengers details whose booked train travels on Saturday.
--> SELECT * FROM (trainlist JOIN train_pass USING(t_number)) JOIN passenger USING(t_id)) WHERE avail LIKE '%Sat%';

--------------------MondoDB------------------------------
>>>>Install the dependencies<<<<
sudo apt update

sudo apt install dirmngr gnupg apt-transport-https ca-certificates software-properties-common
>y

>>>>Import the repository<<<<
wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -

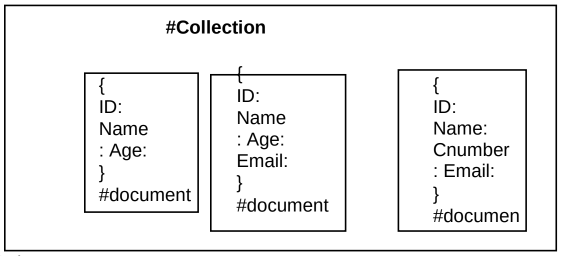
sudo add-apt-repository 'deb [arch=amd64] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/4.4 multiverse'

>>>>install the mongodb-org meta-package<<<<
sudo apt install mongodb-org
>y

>>>>Start the MongoDB daemon and enable it to start on boot by typing<<<<
sudo systemctl enable --now mongod

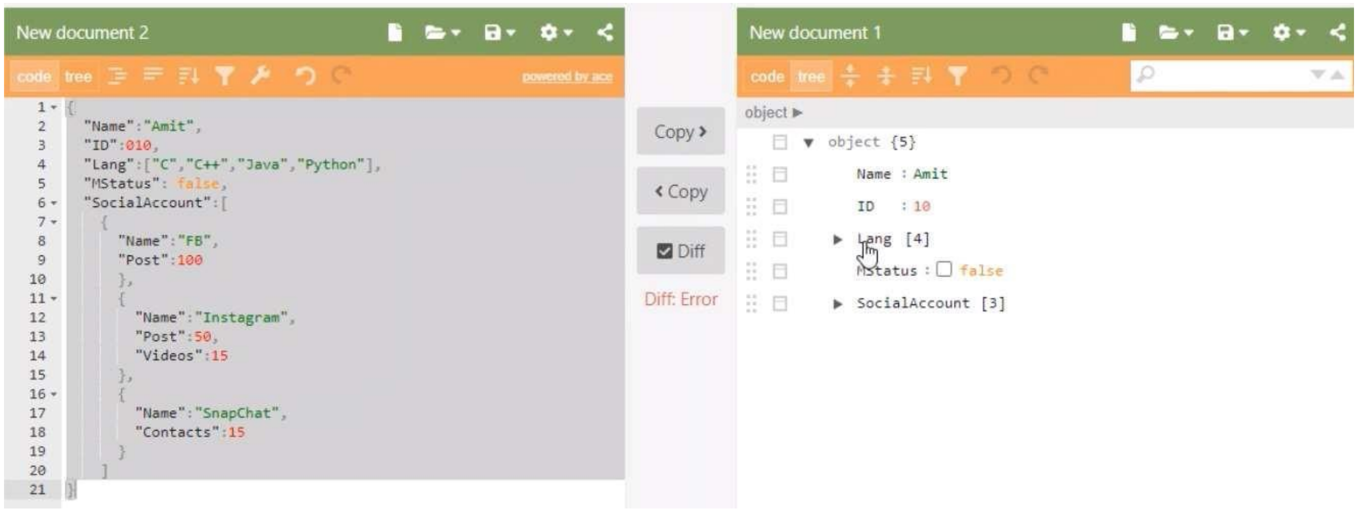
>>>>To verify whether the installation has completed successfully, connect to the MongoDB database server using the mongo tool, and print the connection status<<<<
mongo --eval 'db.runCommand({ connectionStatus: 1})'

NOSQL Jargon:
Collection --> Tables
Documents ---> Rows
Fields/objects -->



Column

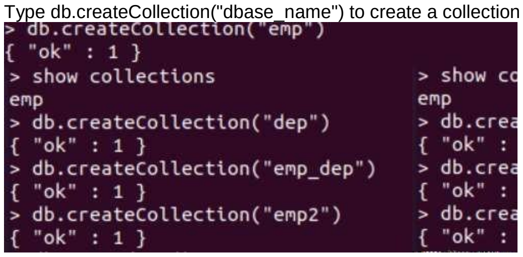
In below example on the **right side** we have a **Tree structure** of the **JSON file** which is on the **Left side**. **Q.Wht if we want to access an object in collections ?**
--> Every object have a UNIQUE ID you can see below Screen shot.
Q What if we want to access an object which is within the object?
--> No worries. For e.g.: If we want **to access** let's say **SocialAccount Object** then just fetch main Object with ID.
Now there are 5 objects within main object so 1st Object is Name similarly 5th Object is SocialAccount, So **Id number is 0,5.**
For e.g.: we want **to access SnapChat** which is inside SocialAccount object so It's **Id is 0,5,3.** Bcoz it is 3rd object in SocialAccount.



```
{
  "Name":"Amit",
  "ID":010,
  "Lang":["C","C++","Java","Python"],
  "MStatus": false,
  "SocialAccount":[
    {
      "Name":"FB",
      "Post":100
    },
    {
      "Name":"Instagram",
      "Post":50,
      "Videos":15
    },
    {
      "Name":"SnapChat",
      "Contacts":15
    }
  ]
}
```

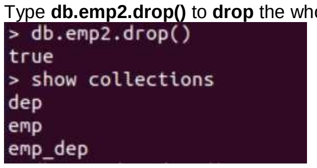
```
▼ object {5}
    Name : Amit
    ID   : 10
  ▶ Lang [4]
    MStatus : ☐ false
  ▶ SocialAccount [3]
```

Mongo commands:
Use mongo to start mongodb
Use show dbs to see all the database present. If you created to new but that dbase is empty then you won't see your new dbase. Type use databasename to create new database or switch to new dbase if already created.
Type db to see which dbase I am currently in.
Type db.createCollection("dbase_name") to create a collection in dbase.



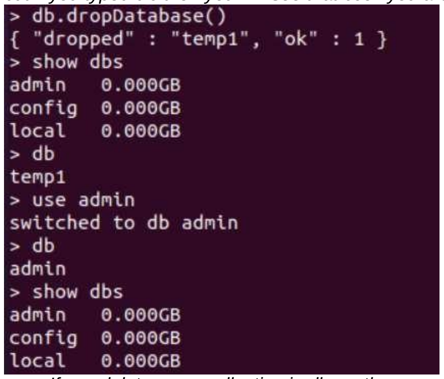
```
> db.createCollection("emp")
{ "ok" : 1 }
> show collections                    > show co
emp                                   emp
> db.createCollection("dep")          > db.crea
{ "ok" : 1 }                          { "ok" :
> db.createCollection("emp_dep")      > db.crea
{ "ok" : 1 }                          { "ok" :
> db.createCollection("emp2")         > db.crea
{ "ok" : 1 }                          { "ok" :
```

Type **db.emp2.drop()** to **drop** the whole **collection**.



```
> db.emp2.drop()
true
> show collections
dep
emp
emp_dep
```

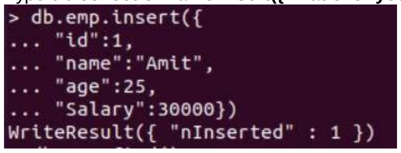
Type **db.dropDatabase()** to drop existing database. *Initially when you drop dbase within that dbase then you can't see dbase if typed **show dbs**, but if you typed **db** then you will see that bcoz you are still in that deleted database which is still in virtual memory.*



```
> db.dropDatabase()
{ "dropped" : "temp1", "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> db
temp1
> use admin
switched to db admin
> db
admin
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
```

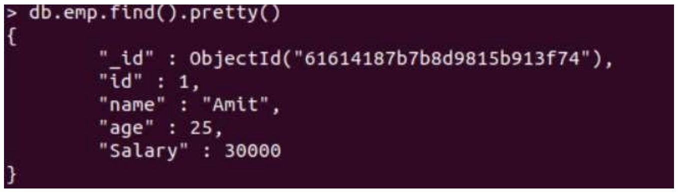
- *If you delete every collection in dbase then your dbase will not be seen if you type show dbs.*

Type **db.collectionname.insert({what ever you want to enter})**



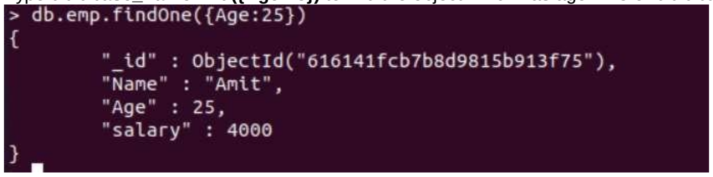
```
> db.emp.insert({
... "id":1,
... "name":"Amit",
... "age":25,
... "Salary":30000})
WriteResult({ "nInserted" : 1 })
```

Type **db.collection_name.find.pretty()** to display **all objects in readable format.**



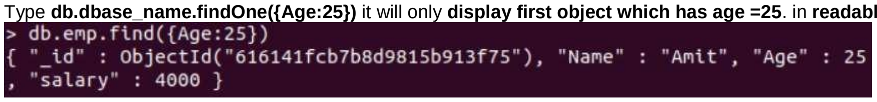
```
> db.emp.find().pretty()
{
        "_id" : ObjectId("61614187b7b8d9815b913f74"),
        "id" : 1,
        "name" : "Amit",
        "age" : 25,
        "Salary" : 30000
}
```

Type **db.dbase_name.find({Age:25})** to find the object which has age = 25 or **db.dbase_name.find({Age:25}).pretty()** as as above but will display in **readable format**.



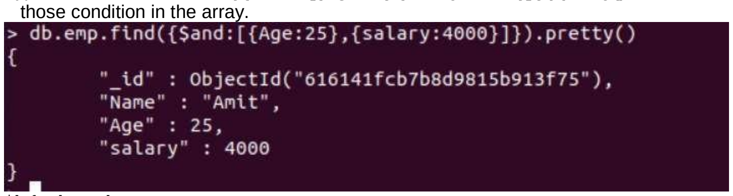
```
> db.emp.findOne({Age:25})
{
        "_id" : ObjectId("616141fcb7b8d9815b913f75"),
        "Name" : "Amit",
        "Age" : 25,
        "salary" : 4000
}
```

Type **db.dbase_name.findOne({Age:25})** it will only **display first object which has age =25**. in **readable format**



```
> db.emp.find({Age:25})
{ "_id" : ObjectId("616141fcb7b8d9815b913f75"), "Name" : "Amit", "Age" : 25
, "salary" : 4000 }
```

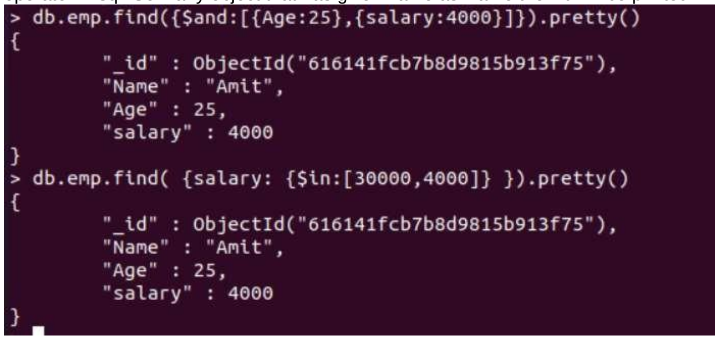
Type **db.dbase_name.find({ $and: [{Age:25},{Salary:30000}] }).pretty()**, to use "**and**" operator we use **$** first to tell that "and" is an operator + we have to create array to put those condition in the array.



```
> db.emp.find({$and:[{Age:25},{salary:4000}]}).pretty()
{
        "_id" : ObjectId("616141fcb7b8d9815b913f75"),
        "Name" : "Amit",
        "Age" : 25,
        "salary" : 4000
}
```

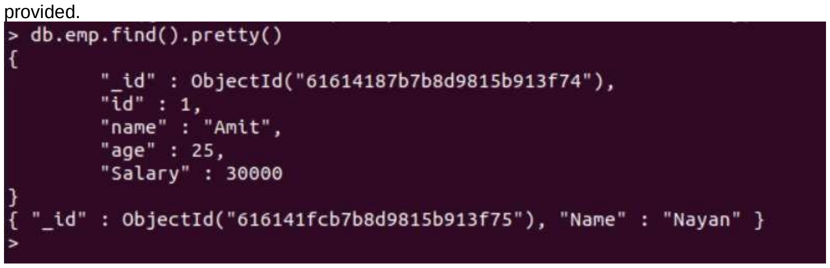
$lt for **less than** to
$lte less than equal to
$gt greater than to
$ne not equal to
$gte greater than equal to
$inc is to **increment data by 1**

Type **db.dbase_name.find({ Name: {$in:["ram","nitin","Rohit"]} }).pretty()**, here $in is a in operator in sql. So if any object that has given name as Name then it will be printed.

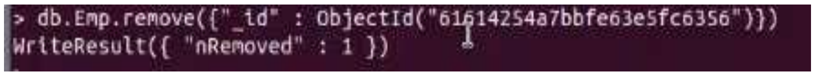


```
> db.emp.find({$and:[{Age:25},{salary:4000}]}).pretty()
{
        "_id" : ObjectId("616141fcb7b8d9815b913f75"),
        "Name" : "Amit",
        "Age" : 25,
        "salary" : 4000
}
> db.emp.find( {salary: {$in:[30000,4000]} }).pretty()
{
        "_id" : ObjectId("616141fcb7b8d9815b913f75"),
        "Name" : "Amit",
        "Age" : 25,
        "salary" : 4000
}
```

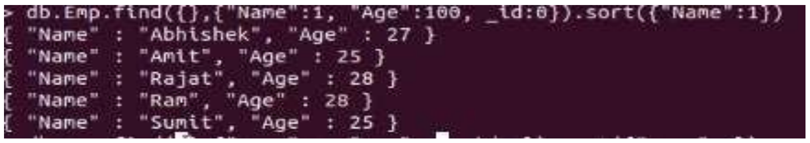
Type **db.dbase_name.update({Name:"Abhishek" }, {the value which I want to update})** with this command object which has name abhishek will be overwrite by the data which we provided.



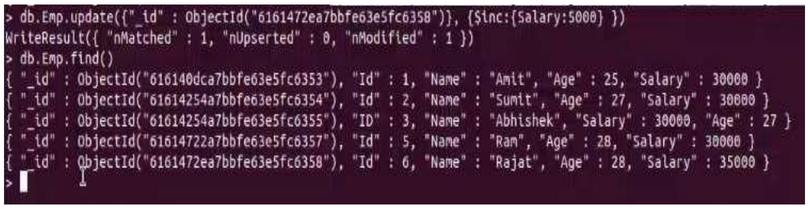
```
> db.emp.find().pretty()
{
        "_id" : ObjectId("61614187b7b8d9815b913f74"),
        "id" : 1,
        "name" : "Amit",
        "age" : 25,
        "Salary" : 30000
}
{ "_id" : ObjectId("616141fcb7b8d9815b913f75"), "Name" : "Nayan" }
>
```



```
> db.Emp.remove({"_id" : ObjectId("61614254a7bbfe63e5fc6356")})
WriteResult({ "nRemoved" : 1 })
```



```
> db.Emp.find({},{"Name":1, "Age":100, _id:0}).sort({"Name":1})
{ "Name" : "Abhishek", "Age" : 27 }
{ "Name" : "Amit", "Age" : 25 }
{ "Name" : "Rajat", "Age" : 28 }
{ "Name" : "Ram", "Age" : 28 }
{ "Name" : "Sumit", "Age" : 25 }
```



```
> db.Emp.update({"_id" : ObjectId("6161472ea7bbfe63e5fc6358")}, {$inc:{Salary:5000} })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Emp.find()
{ "_id" : ObjectId("616140dca7bbfe63e5fc6353"), "Id" : 1, "Name" : "Amit", "Age" : 25, "Salary" : 30000 }
{ "_id" : ObjectId("61614254a7bbfe63e5fc6354"), "Id" : 2, "Name" : "Sumit", "Age" : 27, "Salary" : 30000 }
{ "_id" : ObjectId("61614254a7bbfe63e5fc6355"), "ID" : 3, "Name" : "Abhishek", "Salary" : 30000, "Age" : 27 }
{ "_id" : ObjectId("61614722a7bbfe63e5fc6357"), "Id" : 5, "Name" : "Ram", "Age" : 28, "Salary" : 30000 }
{ "_id" : ObjectId("6161472ea7bbfe63e5fc6358"), "Id" : 6, "Name" : "Rajat", "Age" : 28, "Salary" : 35000 }
>
```