**MONGODB basic Guide**

# **1.Download Mongodb**

<https://www.mongodb.com/try/download/community> - install latest version

this will download MongoDB compass

## Set mongodb as environment variable to use in command prompt

File location of mongodb in local server :C:\Program Files\MongoDB\Server\5.0\bin

Edit the system environment variable 🡪Environmental variable🡪 system variable🡪path🡪edit🡪New🡪copy the file location🡪ok🡪ok

Open command prompt

mongo –-version

## Connection refused error when login to localhost in compass

Services🡪 MongoDB Database Server (MongoDB)🡪start the service

## Set username and password

Host : 127.0.0.1:27017

In command prompt 🡪

mongo –-version

show dbs

use admin

* Admin authentication

db.createUser(

... {

... user:"root",

... pwd:"bhagya",

... roles: [ {role : "readWrite", db:"bankdb" }]

... }

... )

* database bankdb authentication

use bankdb

db.createUser(

... ... {

... ... user:"root",

... ... pwd:"bhagya",

... ... roles: [ {role : "readWrite", db:"bankdb" }]

... ... }

... ... )

# **2.Import Mongodb**

## **In Command Prompt**

mongo –u “root” -p “bhagya” –authenticationDatabase “bankdb”

mongo --port 27017 --authenticationDatabase "Biocalculus" -u "waferadmin" -p "confident123"

## **In Python**

Python need a MongoDb driver – PyMongo

In command prompt install pymongo

python -m pip install pymongo

from pymongo import MongoClient

Client=MongoClient(‘localhost’,27017)

or

client=MongoClient("mongodb://localhost:27017")

or

client=MongoClient("mongodb://root:bhagya@localhost:27017/bankdb")

client=MongoClient("mongodb://username:password@localhost:27017/db name")

client=MongoClient("mongodb://root:bhagya@localhost:27017/bankdb")

print(client.list\_database\_names())

db=client.bankdb

or

db=client[“bankdb”]

print(db.list\_collection\_names())

In mongodb declaring like this db=client.bankdb will create the database bankdb if it doesn’t exist.

# **3.Insert database**

Note : Database and collection are only created when it get contents. We don’t need any query to create database or collection .it will create when it doesn’t exist already.

# create database and collection and insert data into it

def create():

db=client.bankdemo

col=db.users

mydict={'name':'john','address': "newcity"}

x=col.insert\_one(mydict)

print(db.list\_collection\_names())

create()

records={}

records['account\_no']=users\_col.count\_documents({})+1

records['name']=name

records['mobile\_no']=mobile

records['address']=address

records['country']=country

records['email']=email

records['balance']=0

print(records)

rec=users\_col.insert\_one(records)

# **4. Select from record**

Col.find() or col.find\_one() – query used to select from record

Inside find({},{})

In first {} we can mention the conditions to select (like WHERE)

In second {} -object describing which field to include in query

{'\_id':0,'balance':0} – This means that select all except that is given 0

for x in users\_col.find({'account\_no':search\_by},{'\_id':0,'balance':0}):

data.append(x)

print(data)

## **Sort**

#ascending

data=[]

for x in users\_col.find({},{'\_id':0,'balance':0}).sort(sort\_by):

data.append(x)

print(data)

#descending

users\_col.find({},{'\_id':0,'balance':0}).sort(sort\_by,-1):

## **‘$gt’ or ‘$lt’**

Query used to filter the find option based on criteria greater than something

users\_col.find({'balance':{‘$gt’:40000}})

users\_col.find({'balance':{‘$lt’:40000}})

users\_col.find({'address':{‘$gt’: ‘s’ }})

#address starting with s greater than s alphabetically.

## **Regular expression**

Query used to filter the find option based on criteria

users\_col.find({'address':{‘$regex’: ‘^city’ }})

#address starting with s

users\_col.find({'address':{‘$regex’: ‘city$’ }})

#address end with s

users\_col.find({'address':{‘$regex’: ‘city’ }})

#address with city anywhere

users\_col.find({'address':{‘$regex’: ‘^city’, ‘$option’ : ‘$i’ }})

#address starting with s with case insensitive

# **5.Update record**

Col.update\_one({},{}) or col.update\_many({},{})

In first {} – Where condition

In second {} – set condition ={ ‘$set’ : {‘update\_field’: ‘update\_value’ } }

data=users\_col.update\_one({'account\_no':acc\_no},{'$set':{update\_by:set\_inp}})

print(data.modified\_count) # show how many records updated

# **6.Delete record**

data=users\_col.delete\_one({'account\_no':acc\_no})

del\_count=data.deleted\_count