FRIDAY 11 346-020

```
>>> c.execute("DROP table IF EXISTS foodl")
<sqlite3.Cursor at ~~oxoo~~ 0X25...30>
```

MongoDB → NoSQL

→ Steps are in mongodb github
→ Download & install mongodb community server
or
for mac ~~/usr/local/bin/~~

```
% brew install mongodb-community
```

```
% ~~pip install pymongo~~
```

→ Create account in mongodb atlas
→ They give you free 500 MB cloud storage
Create a free cluster
eg ClusterName - cluster0

SATURDAY 12 347-019

→ Starting the server -

```
$ % brew services start mongodb-community
```

→ To stop -

```
%   "   "   stop   "
```

→ In your atlas,
create your 1st database user
whitelist your IP address ⇒ make it local

$$\begin{tikzcd} & \text{local} \\ \text{Mongodb} \arrow[ur] \arrow[dr] & \\ & \text{Remote} \arrow[r, Rightarrow] & \text{Online} \arrow[r, Rightarrow] & \text{Atlas} \arrow[d] \\ & & & \text{Useful for application} \end{tikzcd}$$

SUNDAY 13

F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	JAN
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	2021

14 MONDAY 349-017

→ MySQL is not there in Google collab
Sqlite is there.

NoSQL → Not Only SQL

MongoDB
- → NoSQL → Doesn't require schema → more flexible
- → JSON format (document based)
- → horizontally scalable
- → Unstructured

→ In SQL, if you want to add a column, You need to change the entire & add the data from the start.

15 TUESDAY 350-016

→ Queries in SQL are very slow compared to NoSQL because it parses through each column but in NoSQL, it is key value pairs So, it is much faster

SQL → MySQL, Oracle, dB2, Sqlite
NoSQL → MongoDB, Cassandra, Neo4j, bigquery, influxdB,

→ Database plays an important role in designing an architecture.

→ Charts – fastest way to create visualizations in MongoDB

DEC 2020 T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

WEDNESDAY 16 351-015

→ Mongodb can be connected to any BI tool (BI connector) (PowerBI, Tableau, ClickBU, ...) & we can retrieve, visualize, prepare report of your data.

Pymongo install

~~! python -m pip install pymongo~~
~~% python " " " "~~ (or)
% pip3 install pymongo

MongodB new connection string –

THURSDAY 17 352-014

"mongodb://localhost:27017/"

→ In a team, diff url may be given by your company.

→ Checking whether your pymongo is installed properly & you're able to connect to mongodB using python code –

```
>>> url = "mongodb://localhost:27017/"
db_name = "abc"
client = pymongo.MongoClient(url) # establishing connection with mongodb
dataBase = client[db_name] # creating a dB
```

JAN 2021 F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

18 FRIDAY 353-013

→ You cannot see 'abc' db in mongodb compass even though you've created. It'll only be visible when you store something in it.

```
#list all the databases
>>> client.list_database_names()
['admin', 'config', 'local']
```

```
# function that returns ~~True~~ when a
# db is present in the list
>>> def checkExistence_DB(DB_NAME, client):
        DBlist = client.list_database_names()
        if DB_NAME in DBlist:
            print(f"DB: '{DB_NAME}' exists")
            return True
        print(f"DB: '{DB_NAME}' not present yet")
        return False
```

19 SATURDAY 354-012

```
>>> _ = checkExisting_DB(DB_NAME = ~~DB_NAME~~ 'abc',
                         client)
o/p - DB: 'abc' not present yet
```

→ Collection ⇒ like table in SQL

20 SUNDAY

```
>>> collection = dataBase["iNeuron_Products"]
    collection
Collection(Database(MongoClient(- - -), 'abc'), 'iNeuron_Products')
```

DEC 2020 T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

NOSQL: DB → collection → document (record)

SQL: DB → table → records (rows)

MONDAY 21 356-010

```
>>> def checkExistence_Collect(Collect_NAME, DB_NAME, db):
        collection_list = db.list_collection_names()
        if Collect_NAME in collection_list:
            print(f"Collection: '{Collect_NAME}' in Database: '{DB_NAME}' exists")
            return True
        print(f"                    " does not
              exist or documents are not
              present in the collection")
        return False
>>> _ = checkExistence_Collect("iNeuron_Products",
                               'abc', dataBase)
o/p - Collection: 'iNeuron_Products' in Database: 'abc'
      does not exist or documents are
      not present in the collection
```

TUESDAY 22 357-009

① Inserting a record

```
>>> record = { 'companyName': 'iNeuron',        → JSON
    ↓          'product': 'Affordable AI',
  document     'courseOffered': 'Deep Learning'}
    collection.insert_one(record)
```

→ Now, abc & iNeuron_Products are available.

→ By default some id is allocated to the above record.

```
>>> record = {'kstt': 'abd efg',
              'product': 'affordable abc'}
    collection.insert_one(record)
```

diff schema record can also be added

JAN 2021 F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

23 WEDNESDAY 358-008

② Inserting multiple records

```
>>> list_of_records = [ {'companyName' : 'iNeuron',
                         'product' : 'Affordable AI',
                         'courseOffered' : 'ML with
                                  Deployment'},
        JSON            {   " : ",
                            " : ",
                            " : 'DL for NLP &
                                 computer vision'},
                        {   " : "
                            " : 'Master program',
                            " : 'DS Masters Program'}
                     ]
```

24 THURSDAY 359-007

```
~~>>> inserted_IDs = rec.~~
>>> rec = collection.insert_many(list_of_records)

>>> inserted_IDs = rec.inserted_ids
                                Key, value pairs
    for idx,unique_ids in enumerate(inserted_IDs):
            print(f"{idx}. {unique_ids}")

    0. 5f38 - - - - - - - 72
    1. 5f38 - - - - - - - 73
    2. 5f38 - - - - - - - 74
```

③ Custom id

```
>>> COLLECTION_NAME = "iNeuron_Faculties"
    faculties = dataBase[COLLECTION_NAME]
```

FRIDAY 360-006 25

```
list_of_records_user_defined_id = [
      {"_id" : "6",
       "companyName" : "iNeuron",
       "Faculty" : "Sudhanshu"},
      {"_id" : "7",
         "  :   "
         "  :  "virat Sagar"}]

faculties_record = faculties.insert_many
                   (list_of_records_user_
                    defined_id)
```

④ find() method

```
>>> find_first_record = faculties.find_one()
    print(f"first record of collection: \
    \n{COLLECTION_NAME} is=\
    \n {find_first_record}")

    first record of collection:
    iNeuron_faculties is=
    {id : '1' , 'companyName' : 'iNeuron',
     'Faculty' : 'Sudhanshu'}
```

SATURDAY 361-005 26

→ all the records-

```
>>> all_record = faculties.find()
    for idx,record in enumerate(all_record):
         print(f"{idx} : {record}")
    0 : {'_id' : '1' , 'companyName' : 'iNeuron', 'Faculty' : 'Sudhanshu
    1 :         2   ,           ,          ,          ,
    2 :         6   ,           ,          ,          Sudhanshu
    3 :         7   ,           ,          ,          ,
```

SUNDAY 27

28 MONDAY 363-003

```
(or)
for x in all_record:
        print(x)

{'_id': '1', 'companyName': 'iNeuron', 'Faculty': 'Sudhanshu'}
{        2                       |                  |
         6                       (                  ).
         7
>>> all_record
<pymongo.cursor.Cursor at 0x20 - - - >
                  ↓
            cursor object
```

→ finding all the records having some field

29 TUESDAY 364-002

```
>>> all_record = faculties.find({},{'Faculty'})
for idx, record in enumerate(all_record):
        print(f"{idx}: {record}")
0 : {'_id': '1', 'Faculty': 'Sudhanshu'}
1 :          2
2 :          6
3 :          7
```

→ Query to filter out data

```
>>> query1 = {"_id": "1"}
    results = faculties.find(query1)
    for data in results:
            print(data)

{'_id': '1', 'companyName': 'iNeuron',
                    'Faculty': 'Sudhanshu'}
```

30 WEDNESDAY 365-001

```
>>> query2 = {"_id": {"$gt": "1"}}
    "
    "
    "
{'_id': '2', 'companyName': 'iNeuron', 'Faclty': }
         6                     |            |
         7                     |            |
```

⑤ Query to delete one or many documents

```
>>> some_data = [
        {'_id': '3', 'companyName': 'iNeuron', 'Faculty': 'XYZ'},
        {        4                        "              'ABC'},
        {        5                        "              'PQR'}]
    faculties.insert_many(some_data)
```

→ Deleting one document

```
>>> query_to_delete = {'Faculty': 'XYZ'}
    faculties.delete_one(query_to_delete)
```

31 THURSDAY 366-000

→ Deleting multiple records

```
>>> multi_query_to_delete = {'_id': {'$gte': '4'}}
    faculties.delete_many(multi_query_to_delete)
```

→ Deleting all the documents in the record

```
faculties.delete_many({})
```

⑥ Drop the entire collection

```
>>> faculties.drop()
```

⑦ Update >>> products = dataBase["iNeuron_Products"]

```
>>> present_data = {'courseOffered': 'ML with deployment'}

new_data = {"$set": {'courseOffered':
                        'ML and DL with deployment'}
           }
products.update_one(present_data, new_data)
```

If there are multiple records with ML with deployment only one of them is updated.

Telephone Index

Name & Address | Phones | Mobile | E-mail / Fax

```
>>> present_data = {'product' :'Affordable AI'}
    new_data = {'$set': {'product' : 'affordable
                                       everything'}}
    products. update_many(present_data, new_data)
```
↓
If there are multiple records having Affordable AI, all are updated

⑧ Retrieving N records

```
>>> N_records = 3
    N_record = products. find(). limit(N_records)
    for idx, record in enumerate(N_record):
        print(f"{record} \n")
```

{'_id': ObjectId('5f.... 71'), 'companyName': 'iNeuron.ai', 'product': ~~Affordable AI~~ 'fast sf ---', 'courseOffered': 'DL for computerVision'}
✓(W)
{'_id': ('72'), — 'Affordable everything' " }

{'id' ('73') — }

MongodB Atlas

After creating a cluster
click on Connect

① Whitelist a connection IP address
⇒ Allow access from everywhere

Telephone Index

Name & Address | Phones | Mobile | E-mail / Fax

⇒ Add IP address

② Create a database User
Username → gautham
Password → 12345
✓ Create db user

③ choose a connection method –
connect your application → python application

Driver: Python
Version: 3.6 or later

☑ include full driver code example.
& copy the code

& change <password> in the copied code to 1234.
<dbname> to test → one you've already created
↓ or,
MyFirstdataBase

1. Connecting through ~~Jupyter Notebook~~
→ Open jupyter note book &
>>> import pymongo
>>> → paste the copied & edited code & ~~execute~~ execute

2. Connecting through Mongodb compass
→ paste the url & connect
>>> We can perform all the same operations on atlas by changing url
url = f" " → paste the url you've copied
~~→ All the changes done!~~