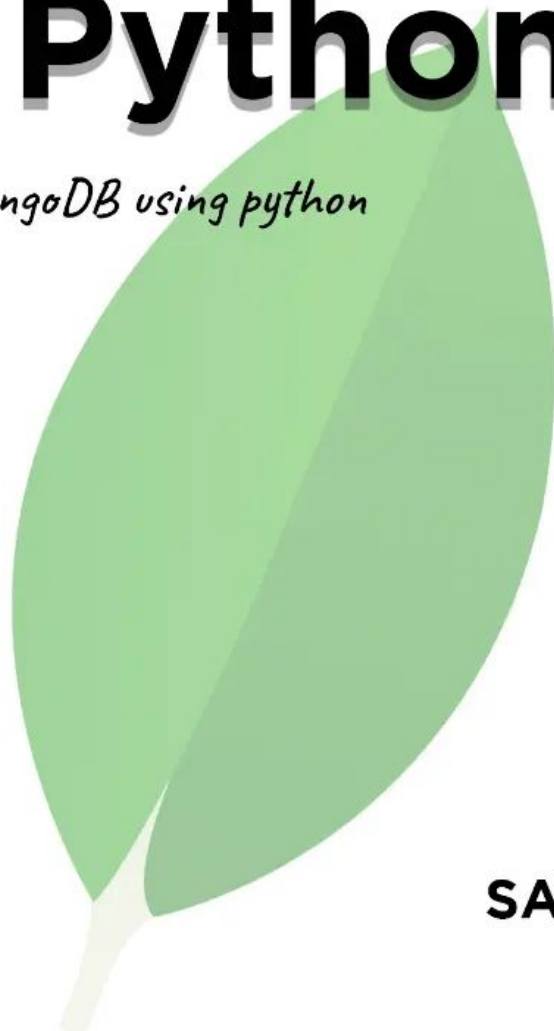


# MongoDB using Python

*CRUD operation with MongoDB using python*



# What Is MongoDB?

MongoDB is a document database with the scalability and flexibility that you want with the querying and indexing that you need

- MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time
- The document model maps to the objects in your application code, making data easy to work with
- Ad hoc queries, indexing, and real time aggregation provide powerful ways to access and analyze your data
- MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use

## Need to run MongoDB?

- High availability through built-in replication and failover
- Horizontal scalability with native sharding
- End-to-end security
- Native document validation and schema exploration with Compass
- Management tooling for automation, monitoring, and backup
- Fully elastic database as a service with built-in best practices



```
1  {
2    _id: "5cf0029cafff5056591b0ce7d",
3    firstname: 'Jane',
4    lastname: 'Wu',
5    address: {
6      street: '1 Circle Rd',
7      city: 'Los Angeles',
8      state: 'CA',
9      zip: '90404'
10   }
11 }
```

03

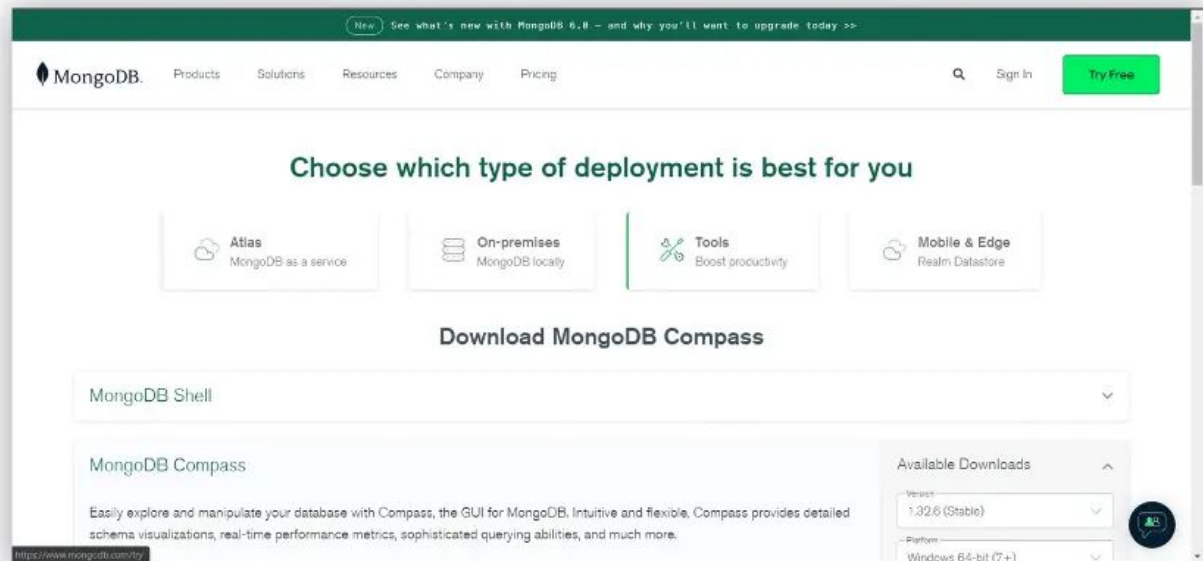
# How to perform CRUD operations in MongoDB using Python.

To install pymongo we need to type the below command in the command prompt.

```
pip install pymongo
```

We will be using MongoDB compass, a GUI used to interact with data stored in MongoDB. You can find the MongoDB compass installation here:

<https://www.mongodb.com/docs/compass/master/install/>



SAVE IT 

## 04

Firstly, we need to import the pymongo package and To connect to the MongoDB server, we will use the MongoClient method.

```
import pymongo

"""connect with MongoDB"""
connection_url = "mongodb://localhost:27017/"

# To connect to the MongoDB server, we will use the MongoClient method.
client = pymongo.MongoClient(connection_url)
print("client: ", client)

#output
# client: MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True)
```

## Create Database

We can create a database using the below command.

```
# To create a database using the below command.
database_name = "employee_db"
employee_db = client[database_name]
```

# 05 Inserting documents in the collection

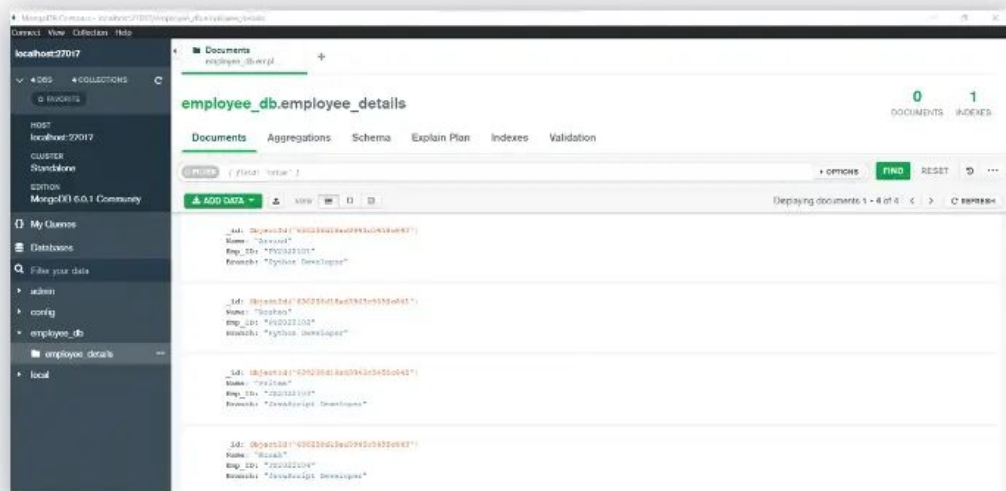
```
# To create a database using the below command.
database_name = "employee_db"
employee_db = client[database_name]

# collection name
collection_name = "employee_details"
collection = employee_db[collection_name]

# To list the available collections in a database we can use the below command.
print("collection: ", employee_db.list_collection_names())

"""Inserting documents in the collection"""
# To insert a single document, we use insert_one() method.
document = {"Name": "Arvind", "Emp_ID": "PY2022101", "Branch": "Python Developer"}
collection.insert_one(document)

# To insert multiple documents, we will use insert_many() method.
documents = [
    {"Name": "Roshan", "Emp_ID": "PY2022102", "Branch": "Python Developer"},
    {"Name": "Pritam", "Emp_ID": "JA2022103", "Branch": "JavaScript Developer"},
    {"Name": "Ronak", "Emp_ID": "JS2022104", "Branch": "JavaScript Developer"},
]
collection.insert_many(documents)
```



SAVE IT 



## Retrieving the data from the collection

```
"""Retrieving the data from the collection"""
# 1. Retrieving a single document
query = {"Name": "Arvind"}
print(collection.find_one(query))

# 2. Retrieving multiple documents from the collection.
query = {"Branch": "Python Developer"}
result = collection.find(query)
print("result: ", result)
for i in result:
    print(i)

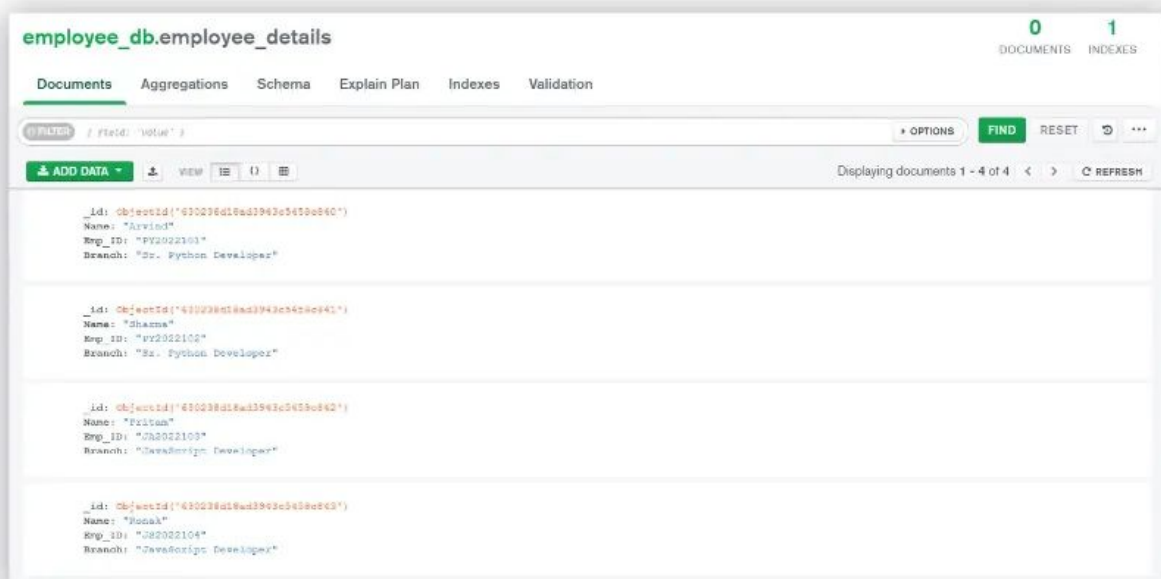
# # OUTPUT
# {'_id': ObjectId('630238d18ad3943c5458c840'), 'Name': 'Arvind', 'Emp_ID': 'PY2022101', 'Branch': 'Python Developer'}
# result: <pymongo.cursor.Cursor object at 0x0000019CC5BCB970>
# {'_id': ObjectId('630238d18ad3943c5458c840'), 'Name': 'Arvind', 'Emp_ID': 'PY2022101', 'Branch': 'Python Developer'}
# {'_id': ObjectId('630238d18ad3943c5458c841'), 'Name': 'Roshan', 'Emp_ID': 'PY2022102', 'Branch': 'Python Developer'}
```

# Updating the documents in the collection

```
"""Updating the documents in the collection"""
# 1. Updating a single document
query = {"Emp_ID": {"$eq": "PY2022102"}}
present_data = collection.find_one(query)

new_data = {"$set": {"Name": "Sharma"}}
collection.update_one(present_data, new_data)

# 2. To update multiple documents we use update_many() method.
present_data = {"Branch": "Python Developer"}
new_data = {"$set": {"Branch": "Sr. Python Developer"}}
collection.update_many(present_data, new_data)
```



employee_db.employee_details		0	1
		DOCUMENTS	INDEXES
Documents Aggregations Schema Explain Plan Indexes Validation			
FINDER / Field: 'value'		+ OPTIONS FIND RESET REFRESH	
ADD DATA VIEW REFRESH			
Displaying documents 1 - 4 of 4			
<pre>{ "_id": ObjectId("630238d18ed3943c5458e642"), "Name": "Arvind", "Emp_ID": "PY2022101", "Branch": "Sr. Python Developer" }</pre>			
<pre>{ "_id": ObjectId("630238d18ed3943c5458e643"), "Name": "Sharma", "Emp_ID": "PY2022102", "Branch": "Sr. Python Developer" }</pre>			
<pre>{ "_id": ObjectId("630238d18ed3943c5458e642"), "Name": "Rishabh", "Emp_ID": "PY2022103", "Branch": "DataScience Developer" }</pre>			
<pre>{ "_id": ObjectId("630238d18ed3943c5458e643"), "Name": "Rishabh", "Emp_ID": "PY2022104", "Branch": "DataScience Developer" }</pre>			

# Deleting documents from the collection

```
"""Deleting documents from the collection"""  
# 1. Deleting a single document  
  
query = {"Emp_ID": "JA2022103"}  
collection.delete_one(query)  
  
# 2. Deleting multiple documents  
  
query = {"Branch": "Sr. Python Developer"}  
collection.delete_many(query)
```

