NoSQL DBs (Not Only Structured Query Language)

NoSQL databases (aka "not only SQL") are non-tabular databases and store data differently than relational tables. NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. They provide flexible schemas and scale easily with large amounts of data and high user loads.

Difference between two most popular SQL and NoSQL Dbs-

MySQL	MongoDB
Matured or stable	Its new and updated frequently
It follows tabular structure	It follows document structure like JSON format
It needs a proper schema	Its flexible in nature
Managing complex relations among different tables is easy	Its not that great in complex managing relationship
Its scales vertically	Horizontaly scalable

Working with MongoDB

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.

Features of MongoDB

i. Rich JSON Documents-

- The most natural and productive way to work with data.
- Supports arrays and nested objects as values.
- Allows for flexible and dynamic schemas.
- The document model maps to the objects in your application code, making data easy to work with.

```
{
 "name": "notebook",
 "qty": 50,
 "rating": [ { "score": 8 }, { "score": 9 } ],
 "size": { "height": 11, "width": 8.5, "unit": "in" },
 "status": "A",
 "tags": [ "college-ruled", "perforated"]
}
```

ii. Powerful query language-

- Rich and expressive query language that allows you to filter and sort by any field, no matter how nested it may be within a document.
- Support for aggregations and other modern use-cases such as geo-based search, graph search, and text search.
- Queries are themselves JSON, and thus easily composable. No more concatenating strings to dynamically generate SQL queries.

```
> db.collection.find( { qty: { $gt: 4 } } )
```

OUTPUT:

```
{ "_id": "apples", "qty": 5 }
{ "_id": "bananas", "qty": 7 }
```

iii. power of a relational database, and more...

- Full ACID(Atomicity, Consistency, Isolation, Durability) transactions.
- Support for joins in queries.
- Two types of relationships instead of one: reference and embedded.

iv. Charts

- The fastest way to create visualizations of MongoDB data.
- Built for the document model.
- Visualize live data from any of your MongoDB instances. Available on MongoDB Atlas.

v. BI Connector

- Allow any BI tool that can speak the MySQL protocol to work with your MongoDB data.

- Leverage the BI tools your organization already uses.
- Perform federated analytics, combining data from MongoDB and other databases.

MongoDBAtlas(Online, Cloud) vs MongoDB Compass(Offline)

- MongoDB Atlas belongs to "MongoDB Hosting" category of the tech stack, while MongoDB Compass can be primarily classified under "Database Tools".
- MongoDB Atlas is a global cloud database service built and run by the team behind MongoDB. Enjoy the flexibility and scalability of a document database, with the ease and automation of a fully managed service on your preferred cloud.
    - Atlas is a mongoDB service on cloud. It uses AWS, Azure and GCP cloud services to cater developers all around the globe for managing mongoDB databases.
    - Its a globally available cloud database service for all kinds of modern applications.
    - Visit [ATLAS HOMEPAGE]() for more details.

Installing MongoDB (Community Edition)

```
- Link to Download MongoDB - [[Download Link](https://www.mongodb.com/try/download/community2)]
- Offical Documentation to Install - [[Official Documents](https://docs.mongodb.com/manual/administration/install-commun
```

Installing MongoDB Python package

```
!python -m pip install pymongo
```

Working in MongoDB - CRUD(CREATE, READ, UPDATE and DELETE.)

Establish connection

```
import pymongo
client = pymongo.MongoClient("mongodb://localhost:27017/")
client
```

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

Creating Database

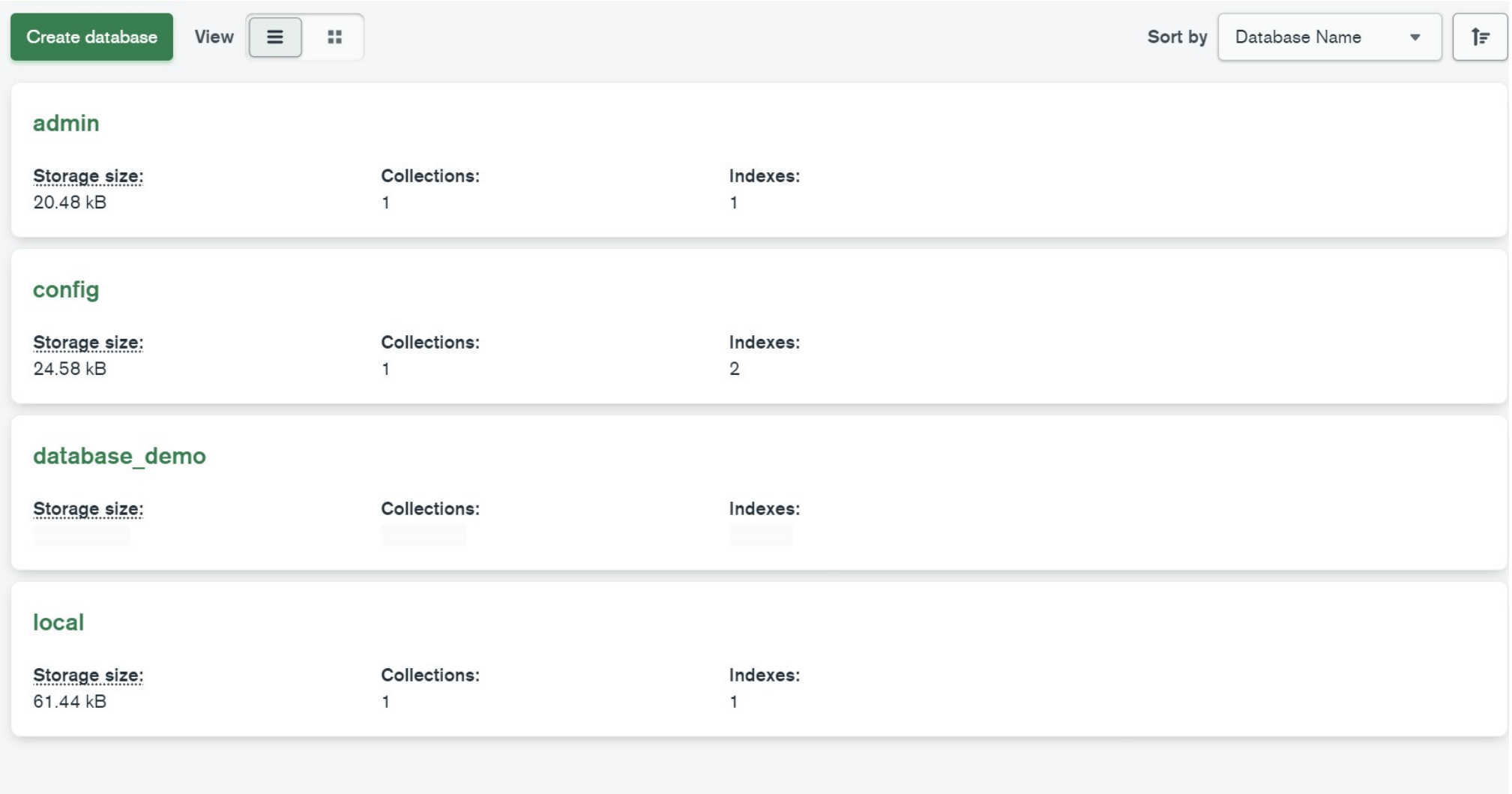
```
DEFAULT_CONNECTION_URL = "mongodb://localhost:27017/"
DB_NAME = "database_demo"

# Establish a connection with mongoDB
client = pymongo.MongoClient(DEFAULT_CONNECTION_URL)
```

```
# Create a DB
dataBase = client[DB_NAME]
```

```
client.list_database_names()
```

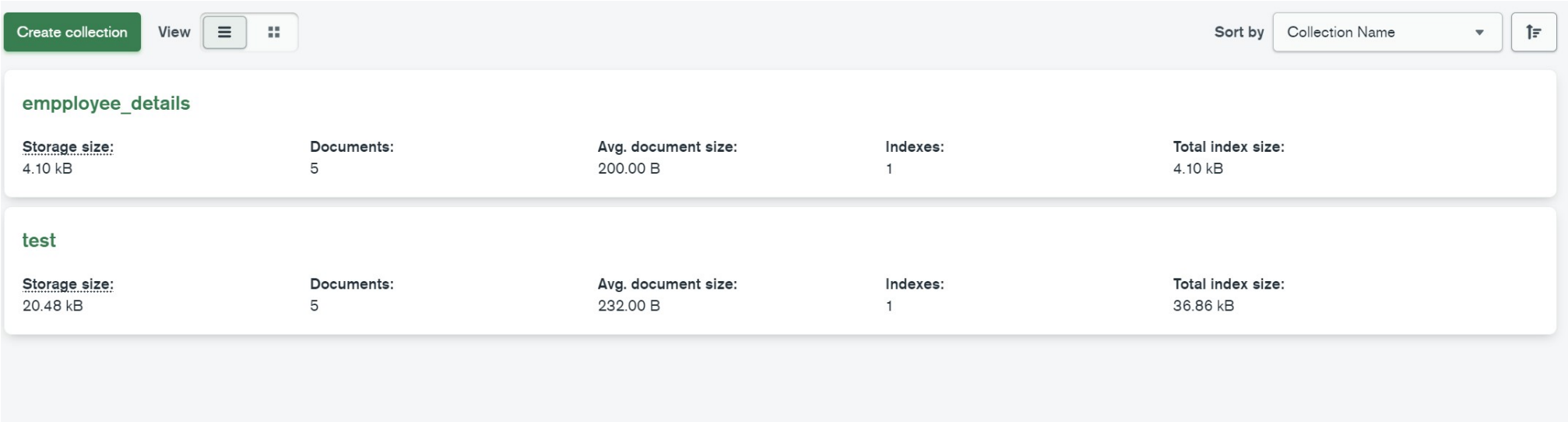
```
['admin', 'config', 'database_demo', 'local']
```



Adding Collection to Database

```
collection = dataBase["empployee_details"]
collection
```

```
Collection(Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False,
connect=True), 'database_demo'), 'empployee_details')
```



Adding Record to Collection - Create

Database doesnt appear until first collection is added

```
record = {'companyName': 'valuelabs',
         'name' : ["Syed","saqlain",123],
         "record_dict" :{"name" :"saqlain" , "mail_id" : "saqlain@fadfsaf.ai","ph_number" :543535},
         "designation"  : "datascientist"
         }

collection.insert_one(record)
```

```
<pymongo.results.InsertOneResult at 0x2b4af454040>
```

```
record = {'companyName': 'amazon',
         'name' : ["zaid","ahmed",234],
         "record_dict" :{"name" :"sabir" , "mail_id" : "saqlain@fadfsaf.ai","ph_number" :543545},
         "designation"  : "software engineer"
         }

collection.insert_one(record)
```

```
<pymongo.results.InsertOneResult at 0x2b4af457d30>
```

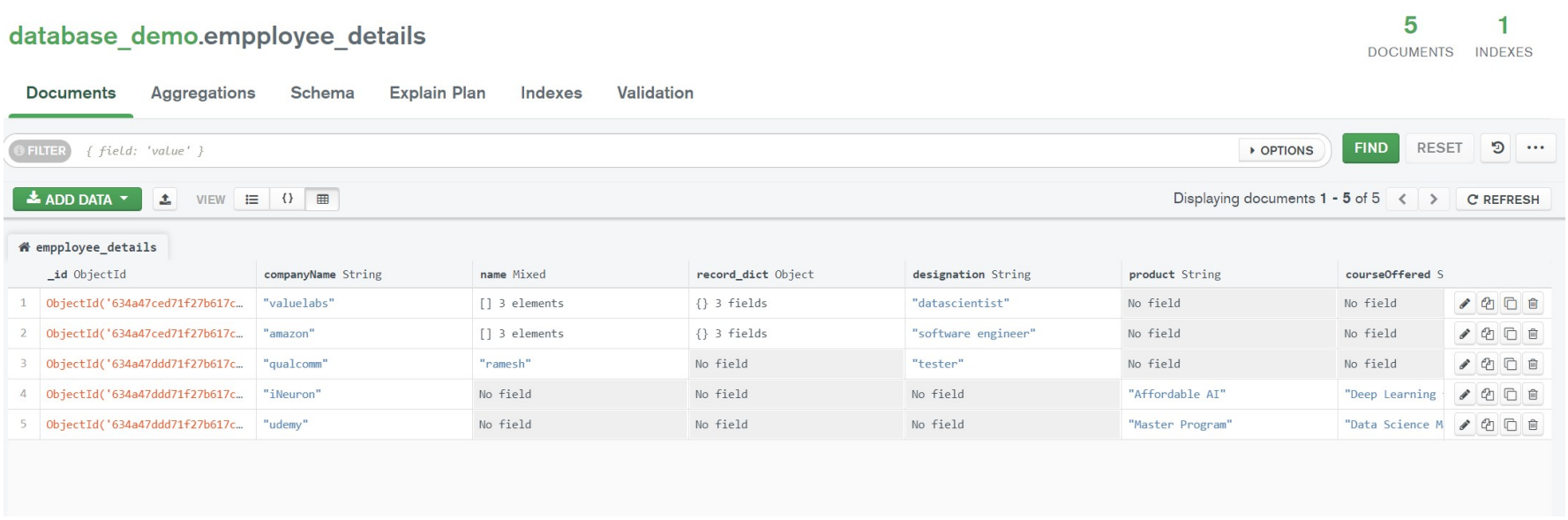
```
list_of_records = [
    {'companyName': 'qualcomm',
      "name" : "ramesh",
    "designation"  : "tester"
         },

    {'companyName': 'iNeuron',
     'product': 'Affordable AI',
     'courseOffered': 'Deep Learning for NLP and Computer vision'},
```

```
    {'companyName': 'udemy',
     'product': 'Master Program',
     'courseOffered': 'Data Science Masters Program',
    "test" : "ffsdfsffsf",
     "complex" : [{"name" : "sudhanshu","list" : [554,545,454,54,5,4]},{"email_id" :"sudhanshu@dffsf"},{"phone_

    }]

rec = collection.insert_many(list_of_records)
```



	_id ObjectId	companyName String	name Mixed	record_dict Object	designation String	product String	courseOffered S
1	ObjectId('634a47ced71f27b617c…	"valuelabs"	[] 3 elements	{} 3 fields	"datascientist"	No field	No field
2	ObjectId('634a47ced71f27b617c…	"amazon"	[] 3 elements	{} 3 fields	"software engineer"	No field	No field
3	ObjectId('634a47ddd71f27b617c…	"qualcomm"	"ramesh"	No field	"tester"	No field	No field
4	ObjectId('634a47ddd71f27b617c…	"iNeuron"	No field	No field	No field	"Affordable AI"	"Deep Learning
5	ObjectId('634a47ddd71f27b617c…	"udemy"	No field	No field	No field	"Master Program"	"Data Science M

```
# each record is assigned a unique_ids
inserted_IDs = rec.inserted_ids
```

```
for idx, unique_ids in enumerate(inserted_IDs):
    print(f"{idx}. {unique_ids}")
```

```
0. 634a48ddd71f27b617c5f30d
1. 634a48ddd71f27b617c5f30e
2. 634a48ddd71f27b617c5f30f
```

Getting Records - Read

```
find_first_record = collection.find_one()

print(f"The first record of collection: {collection.name} is=\
\n {find_first_record}")
```

```
The first record of collection: empployee_details is=
 {'_id': ObjectId('634a47ced71f27b617c5f301'), 'companyName': 'valuelabs', 'name': ['Syed', 'saqlain', 12
```

```
collection.find_one()
```

```
{'_id': ObjectId('634a47ced71f27b617c5f301'),
 'companyName': 'valuelabs',
 'name': ['Syed', 'saqlain', 123],
 'record_dict': {'name': 'saqlain',
  'mail_id': 'saqlain@fadfsaf.ai',
  'ph_number': 543535},
 'designation': 'datascientist'}
```

```
for i in collection.find():
    print(i)
```

```
{'_id': ObjectId('634a47ced71f27b617c5f301'), 'companyName': 'valuelabs', 'name': ['Syed', 'saqlain', 123
{'_id': ObjectId('634a47ced71f27b617c5f302'), 'companyName': 'amazon', 'name': ['zaid', 'ahmed', 234], 'r
{'_id': ObjectId('634a47ddd71f27b617c5f303'), 'companyName': 'qualcomm', 'name': 'ramesh', 'designation':
{'_id': ObjectId('634a47ddd71f27b617c5f304'), 'companyName': 'iNeuron', 'product': 'Affordable AI', 'cour
{'_id': ObjectId('634a47ddd71f27b617c5f305'), 'companyName': 'udemy', 'product': 'Master Program', 'cours
```

Query or filter out data in MongoDB

```
for i in collection.find({'companyName':"valuelabs"}):
    print(i)
```

```
{'_id': ObjectId('634a47ced71f27b617c5f301'), 'companyName': 'valuelabs', 'name': ['Syed', 'saqlain', 123
```

```
query1 = {"name": "Syed"}

results = collection.find(query1)
for data in results:
    print(data)
```

```
{'_id': ObjectId('634a47ced71f27b617c5f301'), 'companyName': 'valuelabs', 'name': ['Syed', 'saqlain', 123
```

Update Records - Update

Update Documentation - [Documentation]

Field Update Records - [field update operators]

```
all_record = collection.find()
for idx, record in enumerate(all_record):
    print(f"{record}\n")
```

```
{'_id': ObjectId('634a47ced71f27b617c5f301'), 'companyName': 'valuelabs', 'name': ['Syed', 'saqlain', 123

{'_id': ObjectId('634a47ced71f27b617c5f302'), 'companyName': 'amazon', 'name': ['zaid', 'ahmed', 234], 'r

{'_id': ObjectId('634a47ddd71f27b617c5f303'), 'companyName': 'qualcomm', 'name': 'ramesh', 'designation':

{'_id': ObjectId('634a47ddd71f27b617c5f304'), 'companyName': 'iNeuron', 'product': 'Affordable AI', 'cour

{'_id': ObjectId('634a47ddd71f27b617c5f305'), 'companyName': 'udemy', 'product': 'Master Program', 'cours
```

```
present_data = {'courseOffered': 'Deep Learning for NLP and Computer vision'}
new_data = {"$set":{'courseOffered': 'Full Stack Data Scientist'}}

collection.update_many(present_data,new_data)
all_record = collection.find()

for idx, record in enumerate(all_record):
    print(f"{record}\n")
```

```
{'_id': ObjectId('634a47ced71f27b617c5f301'), 'companyName': 'valuelabs', 'name': ['Syed', 'saqlain', 123

{'_id': ObjectId('634a47ced71f27b617c5f302'), 'companyName': 'amazon', 'name': ['zaid', 'ahmed', 234], 'r

{'_id': ObjectId('634a47ddd71f27b617c5f303'), 'companyName': 'qualcomm', 'name': 'ramesh', 'designation':

{'_id': ObjectId('634a47ddd71f27b617c5f304'), 'companyName': 'iNeuron', 'product': 'Affordable AI', 'cour

{'_id': ObjectId('634a47ddd71f27b617c5f305'), 'companyName': 'udemy', 'product': 'Master Program', 'cours
```

+ Code + Text

Delete The records - Delete

```
query_to_delete = {"companyName": "udemy"}

collection.delete_one(query_to_delete)
```

```
<pymongo.results.DeleteResult at 0x2b4af41f5b0>
```

```
all_record = collection.find()

for idx, record in enumerate(all_record):
    print(f"{record}\n")
```

```
{'_id': ObjectId('634a47ced71f27b617c5f301'), 'companyName': 'valuelabs', 'name': ['Syed', 'saqlain', 123

{'_id': ObjectId('634a47ced71f27b617c5f302'), 'companyName': 'amazon', 'name': ['zaid', 'ahmed', 234], 'r
```

```
{'_id': ObjectId('634a47ddd71f27b617c5f303'), 'companyName': 'qualcomm', 'name': 'ramesh', 'designation':

{'_id': ObjectId('634a47ddd71f27b617c5f304'), 'companyName': 'iNeuron', 'product': 'Something', 'courseOf

{'_id': ObjectId('634a4e56b5deceb06d2424f0'), 'companyName': 'qualcomm', 'name': 'ramesh', 'designation':

{'_id': ObjectId('634a4e74b5deceb06d2424f2'), 'companyName': 'qualcomm', 'name': 'ramesh', 'designation':
```

```
multi_query_to_delete = {"name": "ramesh"}

collection.delete_many(multi_query_to_delete)
```

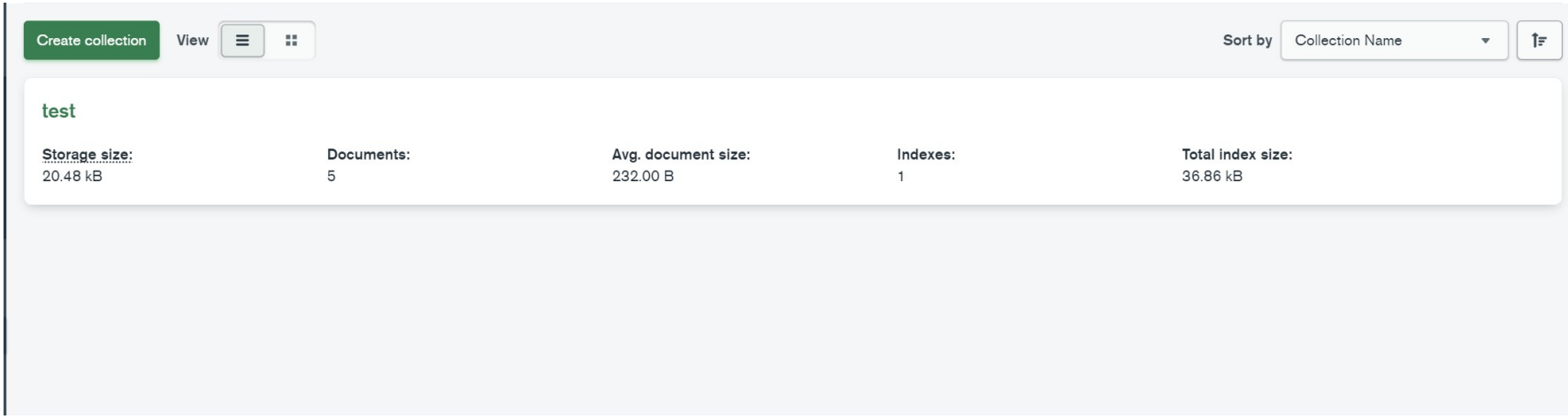
```
for idx, record in enumerate(collection.find()):
    print(f"{record}\n")
```

```
{'_id': ObjectId('634a47ced71f27b617c5f301'), 'companyName': 'valuelabs', 'name': ['Syed', 'saqlain', 123

{'_id': ObjectId('634a47ced71f27b617c5f302'), 'companyName': 'amazon', 'name': ['zaid', 'ahmed', 234], 'r
```

Delete Collection

```
collection.drop()
```



Deleting Database

```
print("List of databases before deletion\n-------------------------")
for x in client.list_database_names():
  print(x)

#delete database named 'organisation'
client.drop_database('database_demo')

print("\nList of databases after deletion\n-------------------------")
for x in client.list_database_names():
  print(x)
```

```
List of databases before deletion
-------------------------
```

```
admin
config
database_demo
local

List of databases after deletion
---------------------------
admin
config
local
```