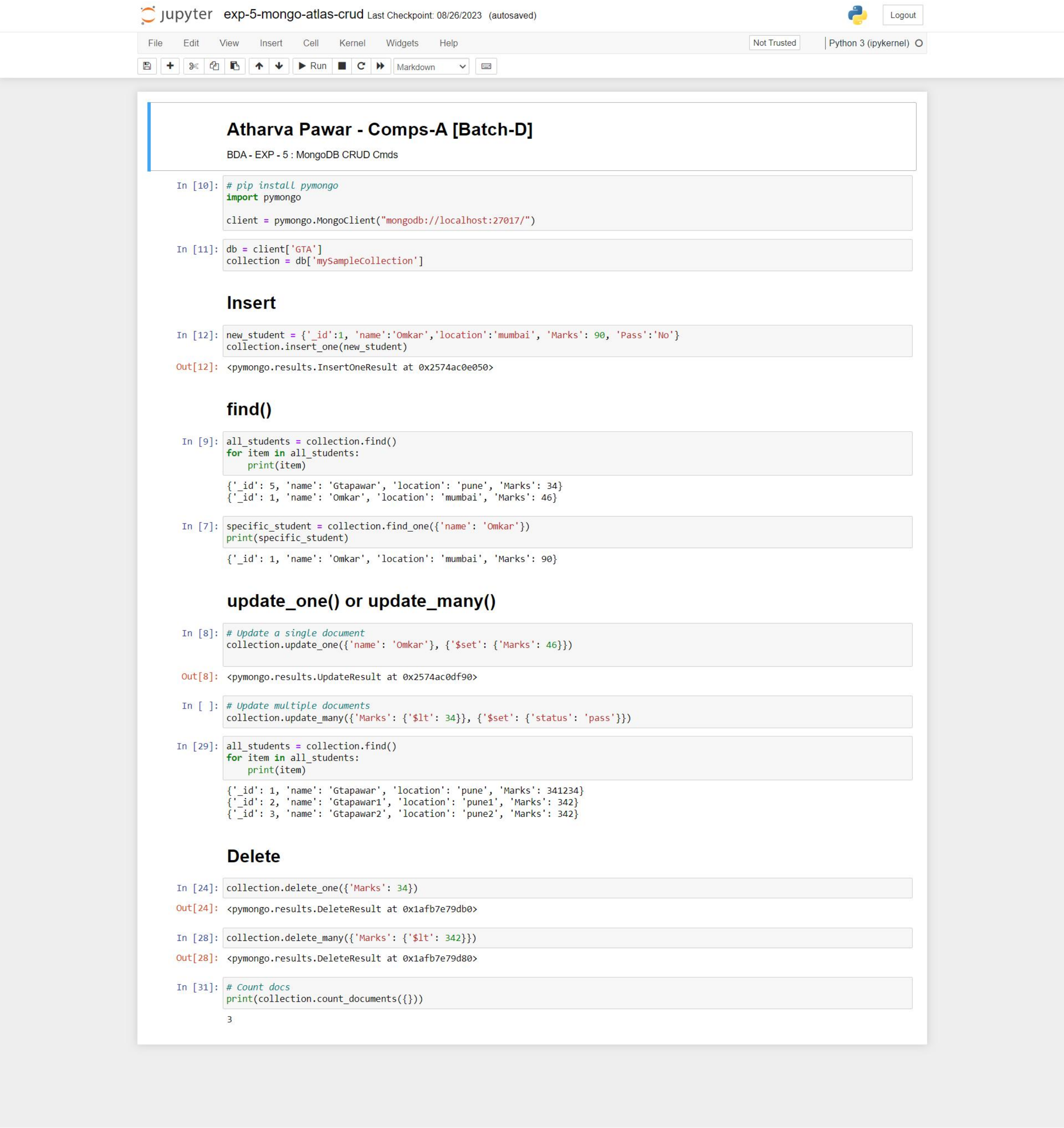

jupyter
exp-5-mongo-atlas-crud Last Checkpoint: 08/26/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

Atharva Pawar - Comps-A [Batch-D]

BDA - EXP - 5 : MongoDB CRUD Cmds

In [10]:
```
# pip install pymongo
import pymongo

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

In [11]:
```
db = client['GTA']
collection = db['mySampleCollection']
```

Insert

In [12]:
```
new_student = {'_id':1, 'name':'Omkar','location':'mumbai', 'Marks': 90, 'Pass':'No'}
collection.insert_one(new_student)
```

Out[12]:
```
<pymongo.results.InsertOneResult at 0x2574ac0e050>
```

find()

In [9]:
```
all_students = collection.find()
for item in all_students:
    print(item)
```

```
{'_id': 5, 'name': 'Gtapawar', 'location': 'pune', 'Marks': 34}
{'_id': 1, 'name': 'Omkar', 'location': 'mumbai', 'Marks': 46}
```

In [7]:
```
specific_student = collection.find_one({'name': 'Omkar'})
print(specific_student)
```

```
{'_id': 1, 'name': 'Omkar', 'location': 'mumbai', 'Marks': 90}
```

update_one() or update_many()

In [8]:
```
# Update a single document
collection.update_one({'name': 'Omkar'}, {'$set': {'Marks': 46}})
```

Out[8]:
```
<pymongo.results.UpdateResult at 0x2574ac0df90>
```

In [ ]:
```
# Update multiple documents
collection.update_many({'Marks': {'$lt': 34}}, {'$set': {'status': 'pass'}})
```

In [29]:
```
all_students = collection.find()
for item in all_students:
    print(item)
```

```
{'_id': 1, 'name': 'Gtapawar', 'location': 'pune', 'Marks': 341234}
{'_id': 2, 'name': 'Gtapawar1', 'location': 'pune1', 'Marks': 342}
{'_id': 3, 'name': 'Gtapawar2', 'location': 'pune2', 'Marks': 342}
```

Delete

In [24]:
```
collection.delete_one({'Marks': 34})
```

Out[24]:
```
<pymongo.results.DeleteResult at 0x1afb7e79db0>
```

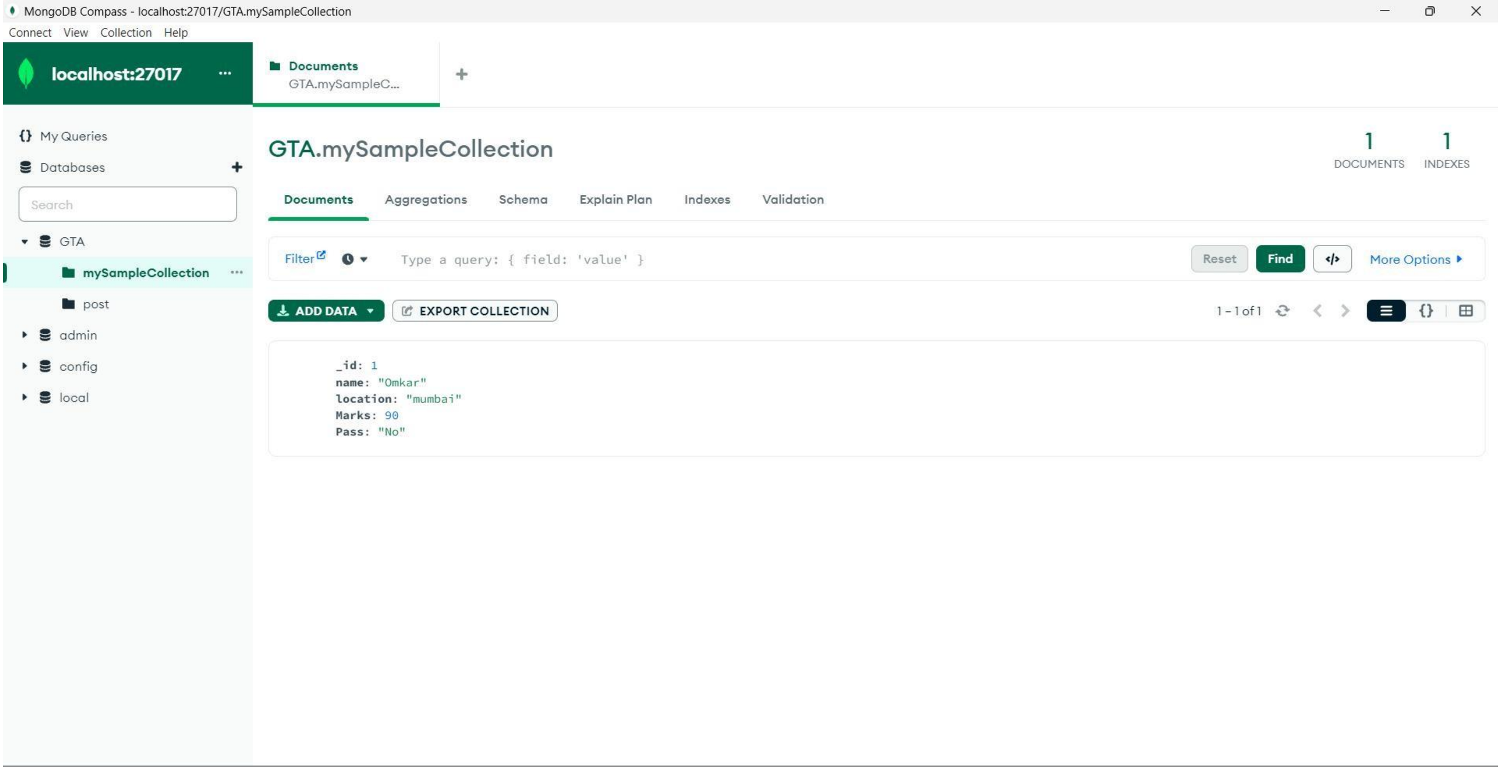
In [28]:
```
collection.delete_many({'Marks': {'$lt': 342}})
```

Out[28]:
```
<pymongo.results.DeleteResult at 0x1afb7e79d80>
```

In [31]:
```
# Count docs
print(collection.count_documents({}))
```

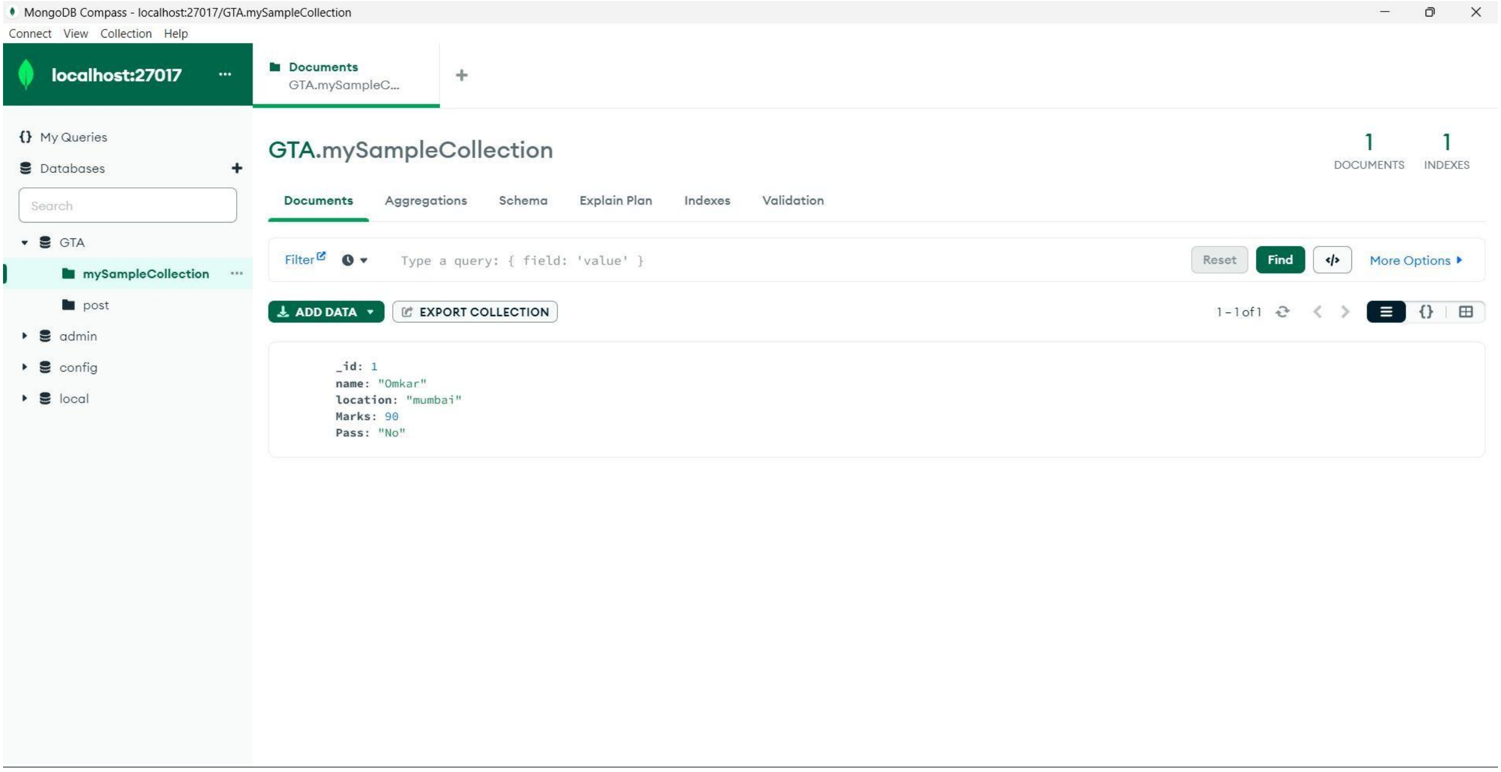
```
3
```

MongoDB Compass (Local)



```
_id: 1
name: "Omkar"
location: "mumbai"
Marks: 90
Pass: "No"
```

MongoDB Compass (Local)



```
_id: 1
name: "Omkar"
location: "mumbai"
Marks: 90
Pass: "No"
```

Atharva Prashant Pawar (9427) [Batch-D]

DATE:

BDA Exp-5

Q1. Explain the Built-in Functions in NoSQL?

A. Built-in Functions in NoSQL:
NoSQL databases are designed to handle vast amounts of unstructured or semi-structured data. They offer built-in functions to perform various operations on this data.

B. Querying & Aggregation:
Builtin functions support querying & aggregation tasks similar to traditional databases, even though data may be stored differently.
Functions help extract, filter & aggregate data for analysis & reporting.

C. Data Transformation:
Builtin functions enable data transformation directly within the database. functions can convert data types, manipulate strings & perform mathematical operations.

D. Geospatial Operations:
NoSQL database often store location-based data. Builtin functions support geospatial queries & calculations.
Functions can find nearby locations, calculate distances & perform polygon operations.

E. Text Search & Analysis:
many NoSQL database include full-text search capabilities.
Builtin functions allow searching for keywords, phrases, or patterns within text fields.

② DATE:

Q2. Describe the various NoSQL Data types?

=> A. Document:

Represents data as documents, similar to JSON or XML. Each document holds key-value pairs where values can be strings, no., arrays, subdocuments, etc.

Popular Databases:
MongoDB, CouchBase, CouchDB

B. Key-value:

Simplest model with keys & associated values. Values can be of any data type, including strings, no., blobs. Suitable for caching & basic storage needs.
DBs: Redis, Amazon DynamoDB.

C. Column-Family:

Stores data in columns grouped into column families or column families into column families.
Each row can have different columns.
Suitable for sparse data or when data is structured as columns.
DBs: Apache Cassandra, HBase.

D. Graph:

Focuses on relationships b/w data entities.
Stores data as nodes (entities) & edges (relationships).
Efficient for traversing complex relationships.
DBs: Neo4j, Amazon Neptune.

FOR EDUCATIONAL USE