MongoDB Commands & Queries

Basic MongoDB Shell Commands

Show all databases:
show dbs

Use a specific database:
use myDatabase

Show all collections in the current database:
show collections

CRUD Operations in MongoDB

Creating a Database and Collection

Create a new database and collection:
use mflix

db.createCollection("users")

Inserting Documents

Insert a single document:
db.users.insertOne({ "name": "John Doe", "age": 30, "email": "john@example.com" })

Insert multiple documents:
db.users.insertMany([

{ "name": "Alice", "age": 25 },

{ "name": "Bob", "age": 28 }

])

Finding Documents

Find all documents (Equivalent to SELECT * FROM users; in SQL):
db.users.find()

Find documents with a specific condition:
db.users.find({ "name": "Davos Seaworth" })

Find movies released in Mexico with an IMDB rating of at least 7:
db.movies.find({ "countries": "Mexico", "imdb.rating": { $gte: 7 } })

Find movies from 2010 that won at least 5 awards or belong to the Drama genre:
db.movies.find({

```
  "year": 2010,
  $or: [
    { "awards.wins": { $gte: 5 } },
    { "genres": "Drama" }
  ]
})
```

Counting Documents

Count how many movies match a query:
db.movies.countDocuments({

```
  "year": 2010,
  $or: [
    { "awards.wins": { $gte: 5 } },
    { "genres": "Drama" }
  ]
})
```

Projecting Specific Fields

Return only the name field of movies that match a query:
db.movies.find(

```
  { "year": 2010, $or: [ { "awards.wins": { $gte: 5 } }, { "genres": "Drama" } ] },
  { "name": 1, "_id": 0 }
)
```

MongoDB with Python (PyMongo)

Connecting to MongoDB in Python

Install PyMongo:
pip install pymongo

Connect to MongoDB:
from pymongo import MongoClient

client = MongoClient('mongodb://user_name:password@localhost:27017')

db = client['ds4300']

collection = db['myCollection']

Performing CRUD Operations in PyMongo

Insert a document:
post = {

"author": "Mark",

"text": "MongoDB is Cool!",

"tags": ["mongodb", "python"]

}

post_id = collection.insert_one(post).inserted_id

print(post_id)

Count documents in a collection:
collection.count_documents({})

MongoDB provides a powerful and flexible approach to handling JSON-like data, making it an essential tool for modern applications that require scalable and efficient data storage.

More on MongoDB & PyMongo

Using PyMongo to Connect to MongoDB

PyMongo is a Python library used to interface with MongoDB databases. It allows applications to perform CRUD (Create, Read, Update, Delete) operations on MongoDB collections.

To install PyMongo, use:

pip install pymongo

To connect to a MongoDB instance running locally:

```
from pymongo import MongoClient

client = MongoClient('mongodb://user_name:password@localhost:27017')
```

Getting a Database and Collection

Once connected, you can access a database and its collections:

```
db = client['ds4300']  # or client.ds4300

collection = db['myCollection']  # or db.myCollection
```

Inserting a Single Document

To insert a document into a collection:

```
post = {
    "author": "Mark",
    "text": "MongoDB is Cool!",
    "tags": ["mongodb", "python"]
}
```

```
post_id = collection.insert_one(post).inserted_id

print(post_id)  # Prints the inserted document's unique ID
```

Querying Data

To find all movies released in the year 2000:

```
from bson.json_util import dumps

movies_2000 = db.movies.find({"year": 2000})

print(dumps(movies_2000, indent=2))  # Pretty print the results
```

Using PyMongo in Jupyter Notebooks

To use PyMongo in a Jupyter Notebook: Activate your Python virtual environment (conda or venv).

Install dependencies:
pip install pymongo jupyterlab

Download and unzip the sample Jupyter notebooks from the provided link.

Navigate to the folder where you unzipped the files and run:
jupyter lab

PyMongo provides a flexible way to interact with MongoDB, allowing efficient storage and retrieval of JSON-like documents in Python applications.