**MONGO DB**

17. Create a database with suitable example using MongoDB and implement Inserting and saving document (batch insert, insert validation) Removing document Updating document (document replacement, using modifiers, upserts, updating documents, returning updated documents)

pip install pymongo

Step 2: Connect to MongoDB

from pymongo import MongoClient

# Connect to the MongoDB server running on localhost at default port 27017

client = MongoClient('localhost', 27017)

# Access or create a database named 'example\_db'

db = client['example\_db']

# Access or create a collection named 'example\_collection'

collection = db['example\_collection']

Step 3: Inserting and Saving Documents

Batch Insert

# Define multiple documents to insert

documents = [

{"name": "John Doe", "age": 30, "city": "New York"},

{"name": "Jane Smith", "age": 25, "city": "Los Angeles"},

# Add more documents as needed

]

# Insert the documents into the collection in a batch

result = collection.insert\_many(documents)

# Print the inserted document IDs

print("Inserted IDs:", result.inserted\_ids)

Insert Validation

# Define a document with validation

document = {"name": "Bob", "age": "not a number", "city": "Chicago"}

# Attempt to insert the document, which will trigger a validation error

try:

result = collection.insert\_one(document)

except Exception as e:

print(f"Insertion error: {e}")

Step 4: Removing Document

# Remove a document based on a query (delete one matching document)

result = collection.delete\_one({"name": "John Doe"})

# Print the number of documents deleted (should be 1 if a match was found)

print("Deleted count:", result.deleted\_count)

Step 5: Updating Document

Document Replacement

# Find a document and replace its content

result = collection.replace\_one({"name": "Jane Smith"}, {"name": "Jane Doe", "age": 28, "city": "San Francisco"})

# Print the number of documents matched and modified (should be 1 if a match was found)

print("Matched and modified:", result.matched\_count)

Using Modifiers

# Update a document using modifiers (e.g., incrementing the 'age' field)

result = collection.update\_one({"name": "Jane Doe"}, {"$inc": {"age": 1}})

# Print the number of documents matched and modified (should be 1 if a match was found)

print("Matched and modified:", result.matched\_count)

Upserts

# Update a document or insert it if it doesn't exist (upsert)

result = collection.update\_one({"name": "Alex"}, {"$set": {"age": 35}}, upsert=True)

# Print the number of documents matched and modified (should be 1 if a match was found or 0 if it's an insert)

print("Matched and modified:", result.matched\_count)

Returning Updated Documents

# Update a document and return the original and updated documents

result = collection.find\_one\_and\_update(

{"name": "Jane Doe"},

{"$set": {"age": 30}},

return\_document=True

)

# Print the original and updated documents

print("Original document:", result["value"])

print("Updated document:", result)

18. Execute at least 10 queries on any suitable MongoDB database

1.INSERT A DOC

db.example\_collection.insertOne({name: "Alice", age: 28, city: "Seattle"})

2.BATCH INSERT DOCS

db.example\_collection.insertMany([

{name: "Bob", age: 35, city: "New York"},

{name: "Charlie", age: 42, city: "San Francisco"}

])

3. QUERY FOR DOC

db.example\_collection.find({city: "New York"})

4. UPDATE DOC

db.example\_collection.updateOne({name: "Alice"}, {$set: {age: 29}})

5. REMOVE A DOC

db.example\_collection.deleteOne({name: "Charlie"})

6. COUNT DOC

db.example\_collection.countDocuments({city: "Seattle"})

7. SORT ELEMENTS’

db.example\_collection.find().sort({age: 1})

8. SELECT SPECIFIC FIELDS – PROJECTION

db.example\_collection.find({city: "New York"}, {name: 1, \_id: 0})

9. AGGREGATION

db.example\_collection.aggregate([

{$group: {\_id: "$city", count: {$sum: 1}}}

])

10. INDEXING

db.example\_collection.createIndex({name: 1})

19. Execute at least 10 queries on any suitable MongoDB database that demonstrates following: $ where queries CRUD Database commands

1. Insert Document:

db.example\_collection.insertOne({name: "John", age: 25, city: "New York"})

2. Batch Insert Documents:

db.example\_collection.insertMany([

{name: "Alice", age: 30, city: "San Francisco"},

{name: "Bob", age: 35, city: "Los Angeles"}

])

3. Query Using $where:

db.example\_collection.find({

$where: function() {

return this.age > 30 && this.city === "San Francisco";

}

})

22. Create a database with suitable example using MongoDB and implement

Inserting and saving document (batch insert, insert validation)

Removing document

Updating document

from pymongo import MongoClient

# Connect to the MongoDB server running on localhost at default port 27017

client = MongoClient('localhost', 27017)

# Access or create a database named 'example\_db'

db = client['example\_db']

# Access or create a collection named 'example\_collection'

collection = db['example\_collection']

# Inserting and saving documents

# Batch Insert

documents = [

{"name": "John Doe", "age": 30, "city": "New York"},

{"name": "Jane Smith", "age": 25, "city": "Los Angeles"},

{"name": "Bob Johnson", "age": 35, "city": "Chicago"}

]

# Batch insert documents into the collection

result = collection.insert\_many(documents)

# Print the inserted document IDs

print("Inserted IDs:", result.inserted\_ids)

# Insert Validation

try:

# Attempt to insert a document with validation (age must be a number)

result = collection.insert\_one({"name": "Alice", "age": "not\_a\_number", "city": "Seattle"})

except Exception as e:

print(f"Insertion error: {e}")

# Removing a document

# Remove a document with a specific condition (e.g., where name is "John Doe")

result = collection.delete\_one({"name": "John Doe"})

print("Deleted count:", result.deleted\_count)

# Updating a document

# Update a document by changing the age of "Jane Smith" to 26

result = collection.update\_one({"name": "Jane Smith"}, {"$set": {"age": 26}})

print("Matched and modified:", result.matched\_count)