**Lab 1: Connecting to MongoDB**

**Task: Connect to a MongoDB instance and list all databases.**

**Solution**

from pymongo import MongoClient

# Connect to MongoDB

client = MongoClient("mongodb://localhost:27017/")

# List all databases

print(client.list\_database\_names())

**Lab 2: Creating a Database and Collection**

**Task: Create a database named school and a collection named students.**

**Solution**

db = client["school"] # Create or get the database

students = db["students"] # Create or get the collection

print("Database and collection created successfully")

**Lab 3: Inserting One Document**

**Task: Insert a student record into the students collection.**

**Solution**

student\_data = {"name": "Alice", "age": 21, "major": "Computer Science"}

result = students.insert\_one(student\_data)

print("Inserted document ID:", result.inserted\_id)

**Lab 4: Inserting Multiple Documents**

**Task: Insert multiple student records into the collection.**

**Solution**

students\_data = [

{"name": "Bob", "age": 22, "major": "Physics"},

{"name": "Charlie", "age": 23, "major": "Mathematics"},

{"name": "David", "age": 20, "major": "Biology"}

]

result = students.insert\_many(students\_data)

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

**Lab 5: Finding One Document**

**Task: Retrieve a single student record.**

**Solution**

student = students.find\_one({"name": "Alice"})

print(student)

**Lab 6: Finding Multiple Documents**

**Task: Retrieve all student records.**

**Solution**

for student in students.find():

print(student)

**Lab 7: Using Query Operators**

**Task: Find all students older than 21.**

**Solution**

for student in students.find({"age": {"$gt": 21}}):

print(student)

**Lab 8: Updating a Document**

**Task: Update Alice's age to 22.**

**Solution**

students.update\_one({"name": "Alice"}, {"$set": {"age": 22}})

print("Updated successfully")

**Lab 9: Updating Multiple Documents**

**Task: Increase the age of all students by 1 year.**

**Solution**

students.update\_many({}, {"$inc": {"age": 1}})

print("Updated successfully")

**Lab 10: Deleting a Document**

**Task: Delete a student named Bob.**

**Solution**

students.delete\_one({"name": "Bob"})

print("Deleted successfully")

**Lab 11: Deleting Multiple Documents**

**Task: Delete all students majoring in Biology.**

**Solution**

students.delete\_many({"major": "Biology"})

print("Deleted successfully")

**Lab 12: Counting Documents**

**Task: Count the number of students in the collection.**

**Solution**

count = students.count\_documents({})

print("Total students:", count)

**Lab 13: Sorting Query Results**

**Task: Retrieve all students sorted by age in descending order.**

**Solution**

for student in students.find().sort("age", -1):

print(student)

**Lab 14: Using Projection**

**Task: Retrieve only the names of all students.**

**Solution**

for student in students.find({}, {"name": 1, "\_id": 0}):

print(student)

**Lab 15: Creating an Index**

**Task: Create an index on the name field.**

**Solution**

students.create\_index("name")

print("Index created")

**Lab 16: Using Aggregation**

**Task: Find the average age of students.**

**Solution**

pipeline = [{"$group": {"\_id": None, "average\_age": {"$avg": "$age"}}}]

result = list(students.aggregate(pipeline))

print("Average age:", result[0]["average\_age"] if result else "No data")

**Lab 17: Using $lookup for Joins**

**Task: Create a grades collection and join it with students.**

**Solution**

grades = db["grades"]

grades.insert\_many([

{"student\_name": "Alice", "grade": "A"},

{"student\_name": "Charlie", "grade": "B"}

])

pipeline = [

{

"$lookup": {

"from": "grades",

"localField": "name",

"foreignField": "student\_name",

"as": "student\_grades"

}

}

]

for student in students.aggregate(pipeline):

print(student)

**Lab 18: Transactions (MongoDB 4.0+)**

**Task: Use transactions to insert two documents atomically.**

**Solution**

session = client.start\_session()

try:

with session.start\_transaction():

students.insert\_one({"name": "Eve", "age": 24, "major": "Engineering"}, session=session)

students.insert\_one({"name": "Frank", "age": 25, "major": "Business"}, session=session)

print("Transaction committed")

except Exception as e:

print("Transaction aborted:", e)

**Lab 19: Exporting Data**

**Task: Export student data to a JSON file.**

**Solution**

import json

students\_list = list(students.find({}, {"\_id": 0}))

with open("students.json", "w") as file:

json.dump(students\_list, file, indent=4)

print("Data exported successfully")

**Lab 20: Importing Data**

**Task: Import student data from a JSON file.**

**Solution**

with open("students.json", "r") as file:

data = json.load(file)

students.insert\_many(data)

print("Data imported successfully")