Q.1 develop a python program to create read update and delete documents in a mongodb collection.

from pymongo import MongoClient

# Connect to MongoDB

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

db = client['mydatabase']

collection = db['mycollection']

# Create a document

def create\_document(document):

result = collection.insert\_one(document)

print(f"Document inserted with \_id: {result.inserted\_id}")

# Read documents

def read\_documents(query={}):

documents = collection.find(query)

for doc in documents:

print(doc)

# Update a document

def update\_document(query, new\_values):

result = collection.update\_one(query, {'$set': new\_values})

print(f"Documents matched: {result.matched\_count}, Documents modified: {result.modified\_count}")

# Delete a document

def delete\_document(query):

result = collection.delete\_one(query)

print(f"Documents deleted: {result.deleted\_count}")

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

# Create a document

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

create\_document(document)

# Read documents

print("Reading documents:")

read\_documents()

# Update a document

query = {"name": "John"}

new\_values = {"age": 31}

update\_document(query, new\_values)

# Read documents again to see the update

print("Reading documents after update:")

read\_documents()

# Delete a document

delete\_document(query)

# Read documents again to see the deletion

print("Reading documents after deletion:")

read\_documents()

OUTPUT:



Q.2 program to insert valkid data into mongoDB using exception handling for errors.

from pymongo import MongoClient, errors

# Connect to MongoDB

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

db = client['mydatabase']

collection = db['mycollection']

# Function to insert data with exception handling

def insert\_data(document):

try:

result = collection.insert\_one(document)

print(f"Document inserted with \_id: {result.inserted\_id}")

except errors.DuplicateKeyError:

print("Error: Duplicate key error. The document already exists.")

except errors.WriteError as e:

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

except errors.PyMongoError as e:

print(f"An error occurred: {e}")

# Example usage

document = {"\_id": 1, "name": "John", "age": 30, "city": "New York"}

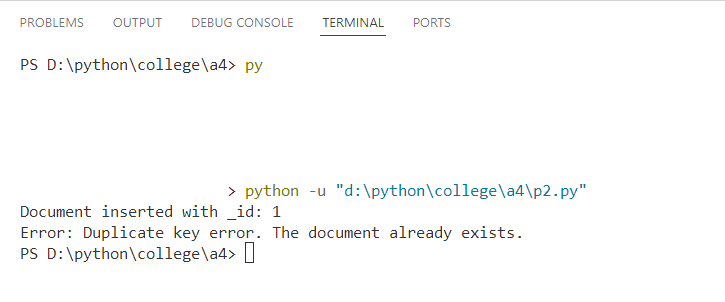
insert\_data(document)

# Trying to insert a duplicate document to trigger an error

duplicate\_document = {"\_id": 1, "name": "Jane", "age": 25, "city": "Los Angeles"}

insert\_data(duplicate\_document)

OUTPUT:



Q.3 use MongoDB’s aggregation pipeline to find the average salary of employees in a collection.

from pymongo import MongoClient

# Connect to MongoDB

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

db = client['employee']

collection = db['empinfo']

# Function to add documents

def add\_documents():

    documents = [

        {"name": "John", "age": 30, "position": "Manager", "salary": 50000},

        {"name": "Jane", "age": 25, "position": "Developer", "salary": 60000},

        {"name": "Mike", "age": 35, "position": "Analyst", "salary": 55000},

        {"name": "Sara", "age": 28, "position": "Designer", "salary": 52000},

        {"name": "Tom", "age": 40, "position": "Director", "salary": 70000}

    ]

    result = collection.insert\_many(documents)

    print(f"Documents inserted with \_ids: {result.inserted\_ids}")

# Aggregation pipeline to calculate the average salary

pipeline = [

    {

        '$group': {

            '\_id': None,

            'average\_salary': {'$avg': '$salary'}

        }

    }

]

# Execute the aggregation pipeline

def calculate\_average\_salary():

    result = list(collection.aggregate(pipeline))

    if result:

        average\_salary = result[0]['average\_salary']

        print(f"The average salary of employees is: {average\_salary}")

    else:

        print("No data found.")

# Example usage

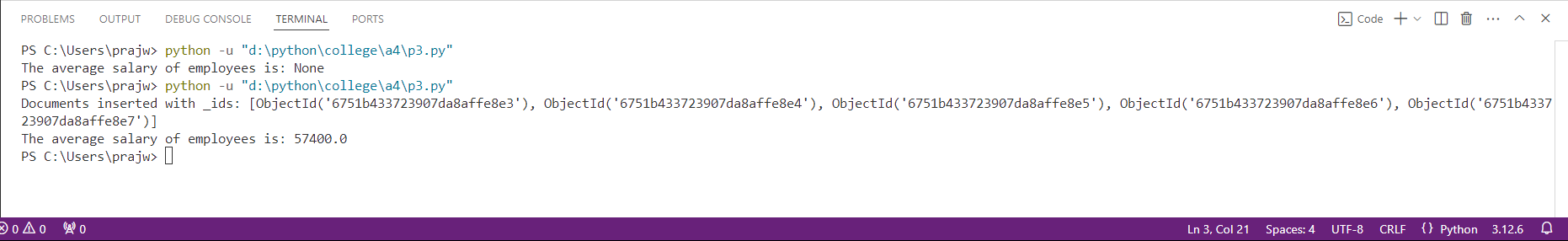
if \_\_name\_\_ == "\_\_main\_\_":

    # Add documents to the collection

    add\_documents()

    # Calculate the average salary

    calculate\_average\_salary()

OUTPUT:  
  


Q.4 develop a python program to create a database name employee with the collection name empinfo. add 5 documents in the collection perform read update and delete operations on documents in a MongoDB

from pymongo import MongoClient

# Connect to MongoDB

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

db = client['employee']

collection = db['empinfo']

# Function to add documents

def add\_documents():

documents = [

{"name": "John", "age": 30, "position": "Manager"},

{"name": "Jane", "age": 25, "position": "Developer"},

{"name": "Mike", "age": 35, "position": "Analyst"},

{"name": "Sara", "age": 28, "position": "Designer"},

{"name": "Tom", "age": 40, "position": "Director"}

]

result = collection.insert\_many(documents)

print(f"Documents inserted with \_ids: {result.inserted\_ids}")

# Function to read documents

def read\_documents():

documents = collection.find()

for doc in documents:

print(doc)

# Function to update a document

def update\_document(query, new\_values):

result = collection.update\_one(query, {'$set': new\_values})

print(f"Documents matched: {result.matched\_count}, Documents modified: {result.modified\_count}")

# Function to delete a document

def delete\_document(query):

result = collection.delete\_one(query)

print(f"Documents deleted: {result.deleted\_count}")

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

# Add documents to the collection

add\_documents()

# Read documents from the collection

print("Reading documents:")

read\_documents()

# Update a document in the collection

query = {"name": "John"}

new\_values = {"age": 31}

update\_document(query, new\_values)

# Read documents again to see the update

print("Reading documents after update:")

read\_documents()

# Delete a document from the collection

delete\_document(query)

# Read documents again to see the deletion

print("Reading documents after deletion:")

read\_documents()

OUTPUT:

A group of people on a white background

Description automatically generated