other-way

December 6, 2023

1 Importing libaries

[ ]:
```
from pymongo import MongoClient
import csv
```

2 Establish a connection to the MongoDB server running on the local machine

[ ]:
```
client = MongoClient('mongodb://localhost:27017/')
db = client['crime_database']
```

3 Access collections within the database

[ ]:
```
crime_types_col = db['crime_types']
incidents_col = db['incidents']
```

4 Function that imports data from a CSV file into the MongoDB collections

[ ]:
```
def import_data():
    file_path = 'C:/Users/User/OneDrive/Desktop/DATA_PROJECT/Crimes-in-Boston/
 ↪crime.csv'
    batch_size = 1000  # Number of documents to insert at a time
    incidents_batch = []
    processed_rows = 0

    with open(file_path, mode='r') as file:
        reader = csv.DictReader(file)
        print("Starting data import...")
        for row in reader:
            # Process each row and create a document for the incidents␣
 ↪collection
            incident = {
                "ID": row['INCIDENT_NUMBER'],
```

```
                "OffenceCode": row['OFFENSE_CODE'],
                "Shootings": row['SHOOTING'].strip().upper() == 'Y',
                "DateTime": {
                    "Year": int(row['YEAR']),
                    "Month": int(row['MONTH']),
                    "DayOfWeek": row['DAY_OF_WEEK'],
                    "Hour": int(row['HOUR'])
                },
                "Place": {
                    "Street": row['STREET'],
                    "Latitude": float(row['Lat']) if row['Lat'] else None,
                    "Longitude": float(row['Long']) if row['Long'] else None,
                    "Location": row['Location']
                }
            }
            incidents_batch.append(incident)
            processed_rows += 1

            # Inserting batch when it reaches the batch size
            if len(incidents_batch) == batch_size:
                incidents_col.insert_many(incidents_batch)
                incidents_batch = []
                print(f"Processed and inserted {processed_rows} rows...")

        # Insert any remaining documents in the last batch
        if incidents_batch:
            incidents_col.insert_many(incidents_batch)
            print(f"Processed and inserted {processed_rows} rows...")

    print("Data import completed.")
```

5 Additional CRUD functions with print statements for feedback

[ ]:
```
def create_incident(incident_data):
    incidents_col.insert_one(incident_data)
    print(f"Inserted incident with ID: {incident_data['ID']}")

def read_incident(incident_id):
    incident = incidents_col.find_one({"ID": incident_id})
    print(f"Read incident: {incident}")
    return incident

def update_incident(incident_id, update_data):
    incidents_col.update_one({"ID": incident_id}, {"$set": update_data})
    print(f"Updated incident with ID: {incident_id}")
```

```
def delete_incident(incident_id):
    incidents_col.delete_one({"ID": incident_id})
    print(f"Deleted incident with ID: {incident_id}")
```

6 Import data

[ ]:
```
import_data()
```

7 Query 1: Count of Incidents by Year

[ ]:
```
def query_incidents_by_year():
    result = incidents_col.aggregate([
        {"$group": {"_id": "$DateTime.Year", "total_incidents": {"$sum": 1}}},
        {"$sort": {"_id": 1}}
    ])
    for item in result:
        print(f"Year: {item['_id']}, Total Incidents:␣
 ↪{item['total_incidents']}")
```

8 Query 2: Top 5 Most Common Crime Types

[ ]:
```
def query_top_crime_types():
    result = incidents_col.aggregate([
        {"$group": {"_id": "$OffenceCode", "count": {"$sum": 1}}},
        {"$sort": {"count": -1}},
        {"$limit": 5}
    ])
    for item in result:
        print(f"Offence Code: {item['_id']}, Count: {item['count']}")
```

9 Query 3: Total number of crimes per hour

[ ]:
```
def query_crimes_by_hour():
    result = incidents_col.aggregate([
        {"$group": {"_id": "$DateTime.Hour", "count": {"$sum": 1}}},
        {"$sort": {"_id": 1}}
    ])
    for item in result:
        print(f"Hour: {item['_id']}, Number of Crimes: {item['count']}")
```

10 Query 4: Crimes with/without shooting per year

[ ]:
```
def query_shooting_vs_nonshooting_by_year():
    result = incidents_col.aggregate([
        {"$group": {
            "_id": {
                "Year": "$DateTime.Year",
                "ShootingInvolved": "$Shootings"
            },
            "count": {"$sum": 1}
        }},
        {"$sort": {"_id.Year": 1, "_id.ShootingInvolved": -1}}
    ])
    for item in result:
        shooting_status = "With Shooting" if item["_id"]["ShootingInvolved"]␣
↪else "Without Shooting"
        print(f"Year: {item['_id']['Year']}, {shooting_status}:␣
↪{item['count']}")
```

11 Executing queries

[ ]:
```
print("\nExecuting queries...")
query_incidents_by_year()
query_top_crime_types()
query_crimes_by_hour()
query_shooting_vs_nonshooting_by_year()
```