main

December 6, 2023

1 Establishing a Connection with MySQL Database

This code imports the mysql.connector module and sets the database credentials (username and password) for connecting to a MySQL database. This is a preliminary step for any database operations.

[30]:
```
import mysql.connector as mysql
from mysql.connector import Error

# MySQL DB credentials
user = "username"
password = "password"
```

2 Creating the 'BostonCrimes' Database

This function, create_database, uses a database cursor to create a new database named 'BostonCrimes'. It handles potential errors during creation and prints relevant messages for success or failure.

[38]:
```
def create_database(cursor):
    try:
        cursor.execute("CREATE DATABASE BostonCrimes")
        print("Database created successfully")
    except Error as e:
        print(f"Error creating database: {e}")
```

3 Defining and Creating Tables for Crime Data

This function create_tables is designed to set up the database structure for handling Boston crime data. It creates three distinct tables:

Crime_Types: Stores crime codes, their groupings, descriptions, and UCR (Uniform Crime Reporting) codes. It ensures that crime types are uniquely identified by a combination of Code and Code_Group.

Incidents: Captures individual crime incidents, including a unique ID, the offence code, a boolean indicating if shootings were involved, and the date of the incident.

Place: Details location information related to each incident, such as street, district, area, and specific location. It also links back to the Incidents table through a foreign key relationship with Incident_ID.

This structured approach facilitates comprehensive data storage and retrieval, necessary for effective crime data analysis. The function concludes by confirming the successful creation of these tables.

[39]:
```
def create_tables(cursor):
    # Crime Types table
    cursor.execute("""
    CREATE TABLE IF NOT EXISTS Crime_Types (
        Code VARCHAR(255),
        Code_Group VARCHAR(255),
        Description VARCHAR(255) NOT NULL,
        UCR VARCHAR(255),
        PRIMARY KEY (Code, Code_Group)
    )
    """)

    # Incidents table
    cursor.execute("""
    CREATE TABLE IF NOT EXISTS Incidents (
        ID VARCHAR(20) PRIMARY KEY,
        Offence_Code VARCHAR(255) NOT NULL,
        Shootings BOOLEAN,
        Date DATETIME NOT NULL
    )
    """)

    # Place table
    cursor.execute("""
    CREATE TABLE IF NOT EXISTS Place (
        Street VARCHAR(255),
        District VARCHAR(255),
        Area VARCHAR(255),
        Location VARCHAR(255) NOT NULL,
        Incident_ID VARCHAR(20),
        FOREIGN KEY (Incident_ID) REFERENCES Incidents(ID)
    )
    """)

    print("Tables created successfully")
```

4 Main Function to Initialize Database and Tables

The main function sets up the Boston crimes database. It connects to MySQL, creates the 'Boston-Crimes' database, and establishes necessary tables. Error handling is included for robustness, and it ensures proper closure of the database connection.

[40]:
```
def main():
    try:
        # Connect to MySQL Server
        db = mysql.connect(host="localhost", user=user, passwd=password)
        cursor = db.cursor()

        # Create database
        create_database(cursor)

        # Connect to the newly created database
        db.database = 'BostonCrimes'

        # Create tables
        create_tables(cursor)

    except Error as e:
        print(f"Error: {e}")
    finally:
        if db.is_connected():
            cursor.close()
            db.close()
            print("MySQL connection is closed")

if __name__ == "__main__":
    main()
```

```
Database created successfully
Tables created successfully
MySQL connection is closed
```

5 Establishing Connection to the BostonCrimes Database

This code snippet establishes a connection to the 'bostoncrimes' database using MySQL. It also imports the csv module for potential data handling. The connection parameters include host, user, password, and the specific database name. A cursor for executing database commands is also initialized.

[41]:
```
import mysql.connector as mysql
import csv

# Connect to the MySQL Database
db = mysql.connect(
    host="localhost",
    user="username",
    passwd="password",
    database="bostoncrimes"
)
```

```
cursor = db.cursor()
```

6 Inserting Crime Data into the Database from CSV

This code comprises a function insert_data to insert data into the Crime_Types, Incidents, and Place tables. It processes each row from a CSV file, preparing and executing SQL insertions. The script also commits transactions at intervals for efficiency. Exception handling is in place to manage any errors. After processing all rows, it commits any remaining data and closes the database connection.

[42]:
```
# Function to insert data into Crime_Types and Incidents
def insert_data(row):
    try:
        # Prepare data for Crime_Types and Incidents
        crime_type_data = (row['OFFENSE_CODE'], row['OFFENSE_CODE_GROUP'],␣
 ↪row['OFFENSE_DESCRIPTION'], row['UCR_PART'])
        incident_data = (row['INCIDENT_NUMBER'], row['OFFENSE_CODE'], 'Y' if␣
 ↪row['SHOOTING'] == 'Y' else None, row['OCCURRED_ON_DATE'])
        place_data = (row['STREET'], row['DISTRICT'], row['REPORTING_AREA'],␣
 ↪row['Location'], row['INCIDENT_NUMBER'])

        # Insert into Crime_Types
        cursor.execute("INSERT IGNORE INTO Crime_Types(Code, Code_Group,␣
 ↪Description, UCR) VALUES (%s, %s, %s, %s)", crime_type_data)

        # Insert into Incidents
        cursor.execute("INSERT IGNORE INTO Incidents(ID, Offence_Code,␣
 ↪Shootings, Date) VALUES (%s, %s, %s, %s)", incident_data)

        #Insert  into Place
        cursor.execute("INSERT INTO Place(Street, District, Area, Location,␣
 ↪Incident_ID) VALUES (%s, %s, %s, %s, %s)", place_data)

        if csv_reader.line_num % 100 == 0:
            db.commit()
    except mysql.connector.Error as err:
        print(f"Error: {err}")
        print(f"Error row: {row}")



# Read CSV and insert data
with open('crime.csv', 'r', encoding='cp1252') as file:
    csv_reader = csv.DictReader(file)

    for row in csv_reader:
```

```
        insert_data(row)

    # Final commit for any remaining rows
    db.commit()

# Close the connection
cursor.close()
db.close()
```