**Title: "Secret Services Database Management System: Safeguarding the Shadows"**

**Introduction:**

In the clandestine world of intelligence and covert operations, information is the key to success. The "Secret Services Database Management System" is a Python-based project designed to efficiently store, manage, and retrieve crucial details related to agents, criminals, safe houses, gadgets, and missions within the realm of secretive operations. Leveraging the power of MySQL connectivity, this database system aims to provide a secure and organized platform for the guardians of national security.

The Secret Services Database Management System is not just a repository; it is a digital fortress, a guardian of secrets, ensuring that the intelligence community has the tools needed to operate effectively in the world of shadows. As we embark on this project, we acknowledge the critical nature of the information it manages and commit to developing a robust, secure, and user-friendly system that empowers those who safeguard our nations from the threats that lurk in the dark.

**Project Overview:**

The Secret Services Database Management System serves as a comprehensive repository for a myriad of data points crucial to the functioning of covert operations. As we delve into the shadows, this project is meticulously crafted to address the intricate needs of intelligence agencies, ensuring that every piece of information is not only stored securely but also easily accessible when needed.

MODULES USED

1. **mysql-connector-python**

MySQL Connector/Python enables Python programs to access MySQL databases

**Installation-**

Run the following command in windows command prompt-

**python -m pip install mysql-connector-python**

1. **Prettytable**

A simple Python library for easily displaying tabular data in a visually appealing ASCII table format.

**Installation-**

Run the following command in windows command prompt-

**python -m pip install prettytable**

CODE TO CREATE REQUIRED DATABASE AND TABLES-

#python -m pip install mysql-connector-python

import mysql.connector as mysql

password=input("Enter your MySQL password:")

def DataBaseCreation():

    # Objective: This Function will create a Database "Secret\_Services" if not exists.

    try:

        mydb = mysql.connect(host="localhost", user="root", passwd=f"{password}")

        cursor = mydb.cursor()

        cursor.execute("create database if not exists Secret\_Services")

        print("Created Database Successfully")

    except:

        print("Database can't be created")

def TablesCreation():

    # Objective: This Function will create the required tables if not exists.

    try:

        mydb = mysql.connect(host="localhost", user="root", passwd=f"{password}", database="Secret\_Services")

        cursor = mydb.cursor()

        cursor.execute("create table if not exists Agents(Agent\_ID int(5) primary key,Code\_Name varchar(15) Unique,Agent\_Name varchar(20),Division varchar(20),Date\_of\_Joining date,Cases\_Solved int(3))")

        cursor.execute("create table if not exists Criminals(Criminal\_ID int(5) primary key,Code\_Name varchar(15) Unique,Criminal\_Name varchar(20),City varchar(25),Country varchar(25))")

        cursor.execute("""create table if not exists Safe\_Houses(Place\_ID int(5) primary key,Code\_Word varchar(15) Unique,Address varchar(40),City varchar(25),Country varchar(25),Operated\_by varchar(15),

foreign key (Operated\_by) references Agents (Code\_Name) on update cascade on delete cascade)""")

        cursor.execute("""create table if not exists Gadgets(Gadget\_ID int(5) primary key,Gadget\_type varchar(15),Quantity int(4),Stored\_at int(5),

foreign key (Stored\_at) references Safe\_Houses(Place\_ID) on update cascade on delete cascade)""")

        cursor.execute("""create table if not exists Missions(Mission\_id int(5) primary key,Mission\_Name varchar(15) Unique,Assigned\_to varchar(15),Against varchar(15),Starting\_Date date,Completion\_Date date,

foreign key (Assigned\_to) references Agents (Code\_Name) on update cascade on delete cascade, foreign key (Against) references Criminals (Code\_Name) on update cascade on delete cascade)""")

        print("Required Tables Created")

    except:

        print("Required Tables can't be created")

DataBaseCreation()

TablesCreation()

OUTPUT-



CODE TO WORK WITH TABLES-

#To use right modules. Run the following command in windows terminal/command prompt.

#python -m pip install mysql-connector-python prettytable

import mysql.connector as mysql

from prettytable import PrettyTable,from\_db\_cursor

password=input("Enter your MySQL password:")

try:

    mydb = mysql.connect(host="localhost", user="root", passwd=f"{password}", database="Secret\_Services")

    cursor = mydb.cursor()

    print("Connection Established Successfully.")

except:

    print("Failed to establish connection.")

    exit()

def Table\_Agents():

    def Add():

        Agent\_ID=int(input("Enter Agent ID:"))

        Code\_Name=input("Enter Agent's Code Name:")

        Agent\_Name=input("Enter Agent's Name:")

        Division=input("Enter Their Respected Division:")

        Date\_of\_Joining=input("Enter Date of Joining[yyyy-mm-dd]:")

        Cases\_Solved=int(input("Enter Total Cases Solved:"))

        try:

            cursor.execute(f"Insert into Agents values({Agent\_ID},'{Code\_Name}','{Agent\_Name}','{Division}','{Date\_of\_Joining}',{Cases\_Solved})")

            mydb.commit()

            print("Data Added Successfully.")

        except:

            print("Failed to Add Data.")

    def Display():

        try:

            cursor.execute("Select \* from Agents")

            results=from\_db\_cursor(cursor)

            print(results)

        except:

            print("Unable to fetch Data.")

    def Update():

        Old\_Agent\_ID=int(input("Enter Current Agent ID:"))

        New\_Agent\_ID=int(input("Enter New Agent ID:"))

        Code\_Name=input("Enter Agent's Code Name:")

        Agent\_Name=input("Enter Agent's Name:")

        Division=input("Enter Their Respected Division:")

        Date\_of\_Joining=input("Enter Date of Joining[yyyy-mm-dd]:")

        Cases\_Solved=int(input("Enter Total Cases Solved:"))

        try:

            cursor.execute(f"Update Agents set Agent\_ID={New\_Agent\_ID}, Code\_Name='{Code\_Name}', Agent\_Name='{Agent\_Name}', Division='{Division}', Date\_of\_Joining='{Date\_of\_Joining}', Cases\_Solved={Cases\_Solved} where Agent\_ID={Old\_Agent\_ID}")

            mydb.commit()

            print("Data Updated Successfully.")

        except:

            print("Failed to Update Data.")

    def Search():

        Agent\_ID=int(input("Enter Agent ID to Search:"))

        try:

            cursor.execute(f"Select \* from Agents where Agent\_ID={Agent\_ID}")

            results=from\_db\_cursor(cursor)

            print(results)

        except:

            print("Failed to Search Data.")

    def Delete():

        print("Proceed with caution.")

        Agent\_ID=int(input("Enter Agent ID to Delete:"))

        try:

            cursor.execute(f"delete from Agents where Agent\_ID={Agent\_ID}")

            mydb.commit()

            print("Data Deleted Successfully.")

        except:

            print("Unable to Delete Data.")

    while True:

        print(' ----------------------------- ')

        print('|        A:Add                |')

        print('|        D:Display            |')

        print('|        U:Update             |')

        print('|        S:Search             |')

        print('|        X:Delete             |')

        print('|        E:Exit               |')

        print(' ----------------------------- ')

        print('                               ')

        choice=input('Enter the function to perform: ')

        print('                               ')

        if choice in "Aa":

            Add()

        elif choice in "Dd":

            Display()

        elif choice in "Uu":

            Update()

        elif choice in "Ss":

            Search()

        elif choice in "Xx":

            Delete()

        else:

            break

def Table\_Criminals():

    def Add():

        Criminal\_ID=int(input("Enter Criminal ID:"))

        Code\_Name=input("Enter Criminal's Code Name:")

        Criminal\_Name=input("Enter Criminal's Name:")

        City=input("Enter City:")

        Country=input("Enter Country:")

        try:

            cursor.execute(f"Insert into Criminals values({Criminal\_ID},'{Code\_Name}','{Criminal\_Name}','{City}','{Country}')")

            mydb.commit()

            print("Data Added Successfully.")

        except:

            print("Failed to Add Data.")

    def Display():

        try:

            cursor.execute("Select \* from Criminals")

            results=from\_db\_cursor(cursor)

            print(results)

        except:

            print("Unable to fetch Data.")

    def Update():

        Old\_Criminal\_ID=int(input("Enter Current Criminal ID:"))

        New\_Criminal\_ID=int(input("Enter New Criminal ID:"))

        Code\_Name=input("Enter Criminal's Code Name:")

        Criminal\_Name=input("Enter Criminal's Name:")

        City=input("Enter City:")

        Country=input("Enter Country:")

        try:

            cursor.execute(f"Update Criminals set Criminal\_ID={New\_Criminal\_ID}, Code\_Name='{Code\_Name}', Criminal\_Name='{Criminal\_Name}', City='{City}', Country='{Country}' where Criminal\_ID={Old\_Criminal\_ID}")

            mydb.commit()

            print("Data Updated Successfully.")

        except:

            print("Failed to Update Data.")

    def Search():

        Criminal\_ID=int(input("Enter Criminal ID to Search:"))

        try:

            cursor.execute(f"Select \* from Criminals where Criminal\_ID={Criminal\_ID}")

            results=from\_db\_cursor(cursor)

            print(results)

        except:

            print("Failed to Search Data.")

    def Delete():

        print("Proceed with caution.")

        Criminal\_ID=int(input("Enter Criminal ID to Delete:"))

        try:

            cursor.execute(f"delete from Criminals where Criminal\_ID={Criminal\_ID}")

            mydb.commit()

            print("Data Deleted Successfully.")

        except:

            print("Unable to Delete Data.")

    while True:

        print(' ----------------------------- ')

        print('|        A:Add                |')

        print('|        D:Display            |')

        print('|        U:Update             |')

        print('|        S:Search             |')

        print('|        X:Delete             |')

        print('|        E:Exit               |')

        print(' ----------------------------- ')

        print('                               ')

        choice=input('Enter the function to perform: ')

        print('                               ')

        if choice in "Aa":

            Add()

        elif choice in "Dd":

            Display()

        elif choice in "Uu":

            Update()

        elif choice in "Ss":

            Search()

        elif choice in "Xx":

            Delete()

        else:

            break

def Table\_Safe\_Houses():

    def Add():

        Place\_ID=int(input("Enter Place ID:"))

        Code\_Word=input("Enter Place's Code Word:")

        Address=input("Enter Address:")

        City=input("Enter City:")

        Country=input("Enter Country:")

        Operated\_by=input("Enter Code Word of the Agent operating the place:")

        try:

            cursor.execute(f"Insert into Safe\_Houses values({Place\_ID},'{Code\_Word}','{Address}','{City}','{Country}','{Operated\_by}')")

            mydb.commit()

            print("Data Added Successfully.")

        except:

            print("Failed to Add Data.")

    def Display():

        try:

            cursor.execute("Select \* from Safe\_Houses")

            results=from\_db\_cursor(cursor)

            print(results)

        except:

            print("Unable to fetch Data.")

    def Update():

        Old\_Place\_ID=int(input("Enter Current Place ID:"))

        New\_Place\_ID=int(input("Enter New Place ID:"))

        Code\_Word=input("Enter Place's Code Word:")

        Address=input("Enter Address:")

        City=input("Enter City:")

        Country=input("Enter Country:")

        Operated\_by=input("Enter Code Word of the Agent operating the place:")

        try:

            cursor.execute(f"Update Safe\_Houses set Place\_ID={New\_Place\_ID}, Code\_Word='{Code\_Word}', Address='{Address}', City='{City}', Country='{Country}', Operated\_by='{Operated\_by}' where Place\_ID={Old\_Place\_ID}")

            mydb.commit()

            print("Data Updated Successfully.")

        except:

            print("Failed to Update Data.")

    def Search():

        Place\_ID=int(input("Enter Place ID to Search:"))

        try:

            cursor.execute(f"Select \* from Safe\_Houses where Place\_ID={Place\_ID}")

            results=from\_db\_cursor(cursor)

            print(results)

        except:

            print("Failed to Search Data.")

    def Delete():

        print("Proceed with caution.")

        Place\_ID=int(input("Enter Place ID to Delete:"))

        try:

            cursor.execute(f"delete from Safe\_Houses where Place\_ID={Place\_ID}")

            mydb.commit()

            print("Data Deleted Successfully.")

        except:

            print("Unable to Delete Data.")

    while True:

        print(' ----------------------------- ')

        print('|        A:Add                |')

        print('|        D:Display            |')

        print('|        U:Update             |')

        print('|        S:Search             |')

        print('|        X:Delete             |')

        print('|        E:Exit               |')

        print(' ----------------------------- ')

        print('                               ')

        choice=input('Enter the function to perform: ')

        print('                               ')

        if choice in "Aa":

            Add()

        elif choice in "Dd":

            Display()

        elif choice in "Uu":

            Update()

        elif choice in "Ss":

            Search()

        elif choice in "Xx":

            Delete()

        else:

            break

def Table\_Gadgets():

    def Add():

        Gadget\_ID=int(input("Enter Gadget ID:"))

        Gadget\_type=input("Enter Gadget Type:")

        Quantity=input("Enter Quantity:")

        Stored\_at=input("Enter Safe House's ID:")

        try:

            cursor.execute(f"Insert into Gadgets values({Gadget\_ID},'{Gadget\_type}','{Quantity}','{Stored\_at}')")

            mydb.commit()

            print("Data Added Successfully.")

        except:

            print("Failed to Add Data.")

    def Display():

        try:

            cursor.execute("Select \* from Gadgets")

            results=from\_db\_cursor(cursor)

            print(results)

        except:

            print("Unable to fetch Data.")

    def Update():

        Old\_Gadget\_ID=int(input("Enter Current Gadget ID:"))

        New\_Gadget\_ID=int(input("Enter New Gadget ID:"))

        Gadget\_type=input("Enter Gadget Type:")

        Quantity=input("Enter Quantity:")

        Stored\_at=input("Enter Safe House's ID:")

        try:

            cursor.execute(f"Update Gadgets set Gadget\_ID={New\_Gadget\_ID}, Gadget\_type='{Gadget\_type}', Quantity='{Quantity}', Stored\_at='{Stored\_at}' where Gadget\_ID={Old\_Gadget\_ID}")

            mydb.commit()

            print("Data Updated Successfully.")

        except:

            print("Failed to Update Data.")

    def Search():

        Gadget\_ID=int(input("Enter Gadget ID to Search:"))

        try:

            cursor.execute(f"Select \* from Gadgets where Gadget\_ID={Gadget\_ID}")

            results=from\_db\_cursor(cursor)

            print(results)

        except:

            print("Failed to Search Data.")

    def Delete():

        print("Proceed with caution.")

        Gadget\_ID=int(input("Enter Gadget ID to Delete:"))

        try:

            cursor.execute(f"delete from Gadgets where Gadget\_ID={Gadget\_ID}")

            mydb.commit()

            print("Data Deleted Successfully.")

        except:

            print("Unable to Delete Data.")

    while True:

        print(' ----------------------------- ')

        print('|        A:Add                |')

        print('|        D:Display            |')

        print('|        U:Update             |')

        print('|        S:Search             |')

        print('|        X:Delete             |')

        print('|        E:Exit               |')

        print(' ----------------------------- ')

        print('                               ')

        choice=input('Enter the function to perform: ')

        print('                               ')

        if choice in "Aa":

            Add()

        elif choice in "Dd":

            Display()

        elif choice in "Uu":

            Update()

        elif choice in "Ss":

            Search()

        elif choice in "Xx":

            Delete()

        else:

            break

def Table\_Missions():

    def Add():

        Mission\_ID=int(input("Enter Mission ID:"))

        Mission\_Name=input("Enter Mission's Name:")

        Assigned\_to=input("Enter Code Name of the Agent the Mission is assigned to:")

        Against=input("Enter Code Name of the Criminal the Mission is against:")

        Starting\_Date=input("Enter Starting Date[yyyy-mm-dd]:")

        Completion\_Date="NULL"

        print("Completion Date is always NULL when a Mission is started.")

        try:

            cursor.execute(f"Insert into Missions values({Mission\_ID},'{Mission\_Name}','{Assigned\_to}','{Against}','{Starting\_Date}',{Completion\_Date})")

            mydb.commit()

            print("Data Added Successfully.")

        except:

            print("Failed to Add Data.")

    def Display():

        try:

            cursor.execute("Select \* from Missions")

            results=from\_db\_cursor(cursor)

            print(results)

        except:

            print("Unable to fetch Data.")

    def Update():

        Old\_Mission\_ID=int(input("Enter Current Mission ID:"))

        New\_Mission\_ID=int(input("Enter New Mission ID:"))

        Mission\_Name=input("Enter Mission's Name:")

        Assigned\_to=input("Enter Code Name of the Agent the Mission is assigned to:")

        Against=input("Enter Code Name of the Criminal the Mission is against:")

        Starting\_Date=input("Enter Starting Date[yyyy-mm-dd]:")

        Completion\_Date=input("Enter Completion Date[yyyy-mm-dd]:")

        try:

            cursor.execute(f"Update Missions set Mission\_ID={New\_Mission\_ID}, Mission\_Name='{Mission\_Name}', Assigned\_to='{Assigned\_to}', Against='{Against}', Starting\_Date='{Starting\_Date}', Completion\_Date='{Completion\_Date}' where Mission\_ID={Old\_Mission\_ID}")

            mydb.commit()

            print("Data Updated Successfully.")

        except:

            print("Failed to Update Data.")

    def Search():

        Mission\_ID=int(input("Enter Mission ID to Search:"))

        try:

            cursor.execute(f"Select \* from Missions where Mission\_ID={Mission\_ID}")

            results=from\_db\_cursor(cursor)

            print(results)

        except:

            print("Failed to Search Data.")

    def Delete():

        print("Proceed with caution.")

        Mission\_ID=int(input("Enter Mission ID to Delete:"))

        try:

            cursor.execute(f"delete from Missions where Mission\_ID={Mission\_ID}")

            mydb.commit()

            print("Data Deleted Successfully.")

        except:

            print("Unable to Delete Data.")

    while True:

        print(' ----------------------------- ')

        print('|        A:Add                |')

        print('|        D:Display            |')

        print('|        U:Update             |')

        print('|        S:Search             |')

        print('|        X:Delete             |')

        print('|        E:Exit               |')

        print(' ----------------------------- ')

        print('                               ')

        choice=input('Enter the function to perform: ')

        print('                               ')

        if choice in "Aa":

            Add()

        elif choice in "Dd":

            Display()

        elif choice in "Uu":

            Update()

        elif choice in "Ss":

            Search()

        elif choice in "Xx":

            Delete()

        else:

            break

while True:

    print(' ----------------------------- ')

    print('|        A:Agents             |')

    print('|        C:Criminals          |')

    print('|        S:Safe\_Houses        |')

    print('|        G:Gadgets            |')

    print('|        M:Missions           |')

    print('|        E:Exit               |')

    print(' ----------------------------- ')

    print('                               ')

    choice=input('Enter the table to work with: ')

    print('                               ')

    if choice in "Aa":

        Table\_Agents()

    elif choice in "Cc":

        Table\_Criminals()

    elif choice in "Ss":

        Table\_Safe\_Houses()

    elif choice in "Gg":

        Table\_Gadgets()

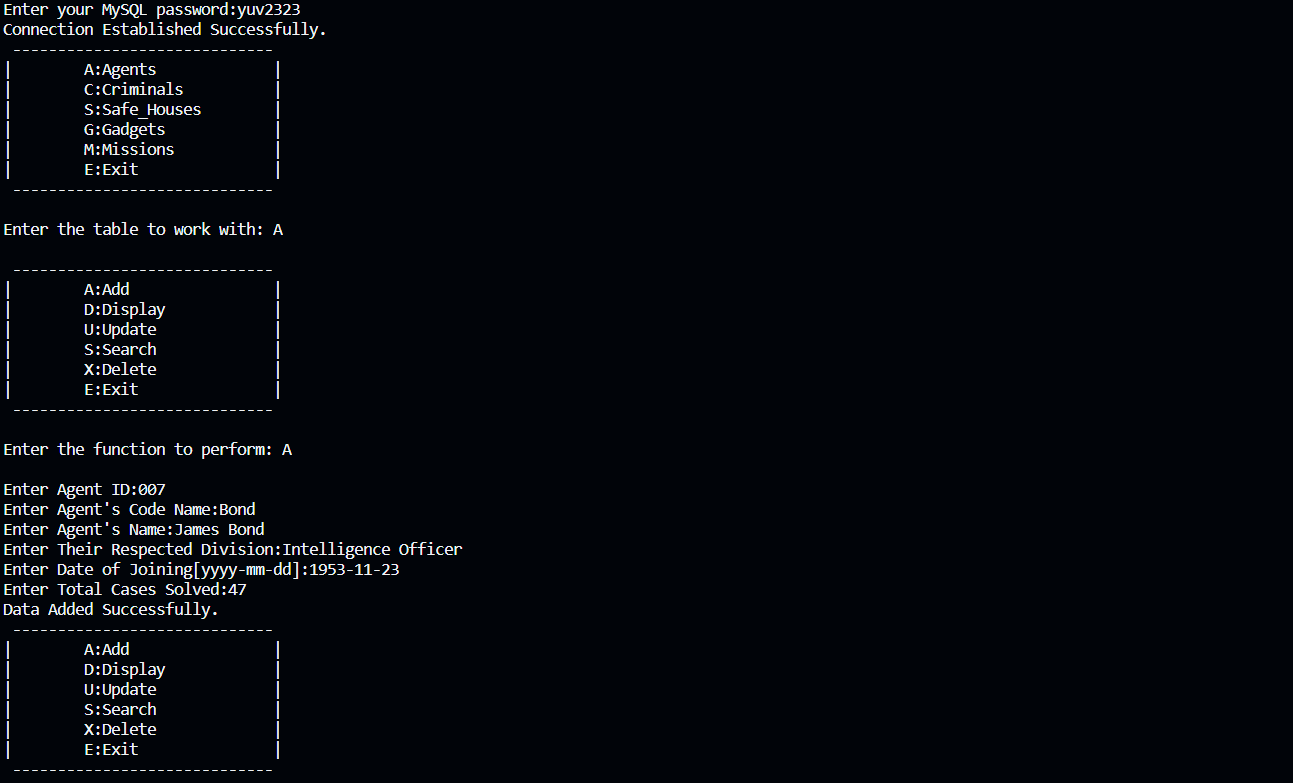
    elif choice in "Mm":

        Table\_Missions()

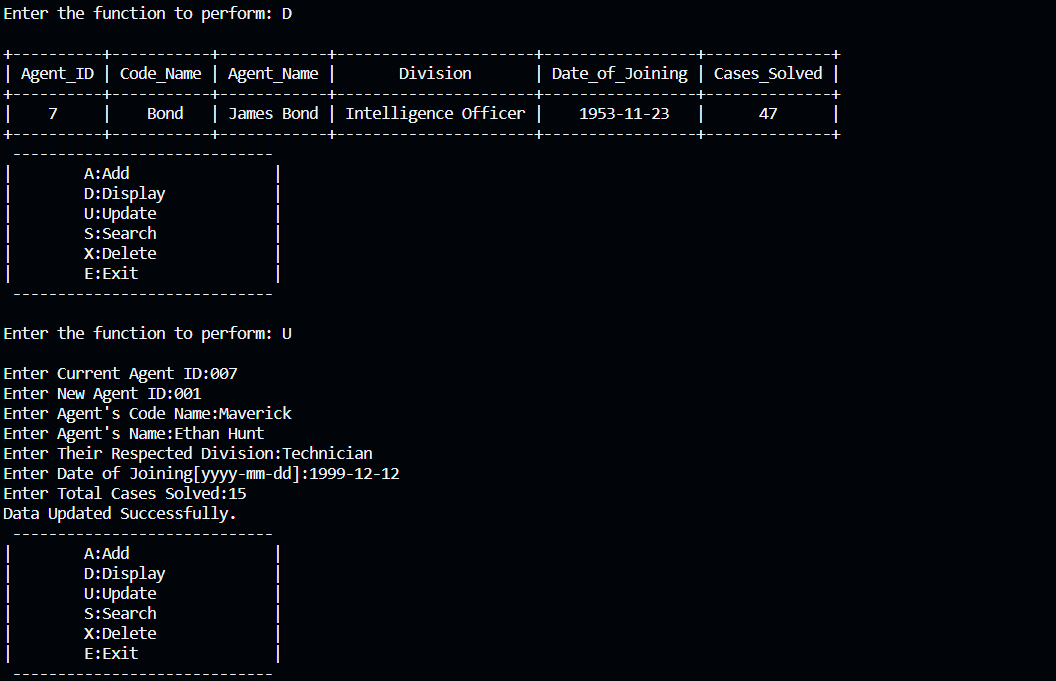
    else:

        break

OUTPUTS-

1. Table Agent-

Adding Data

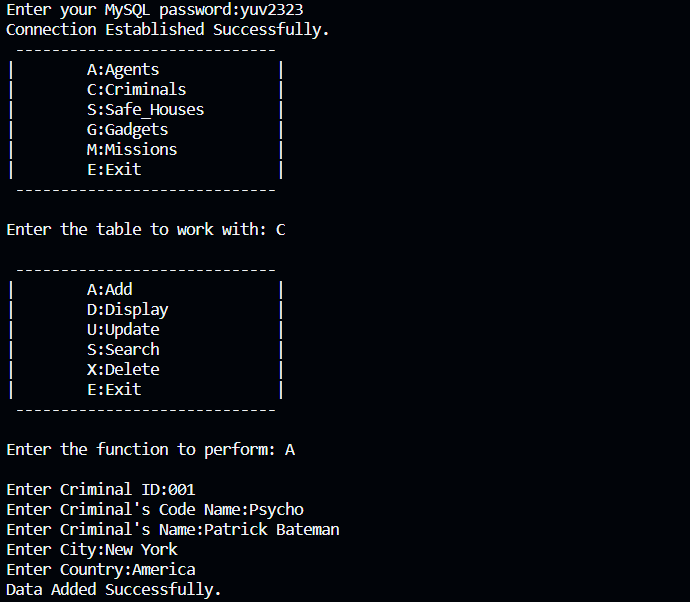


DISPLAYING AND UPDATING DATA

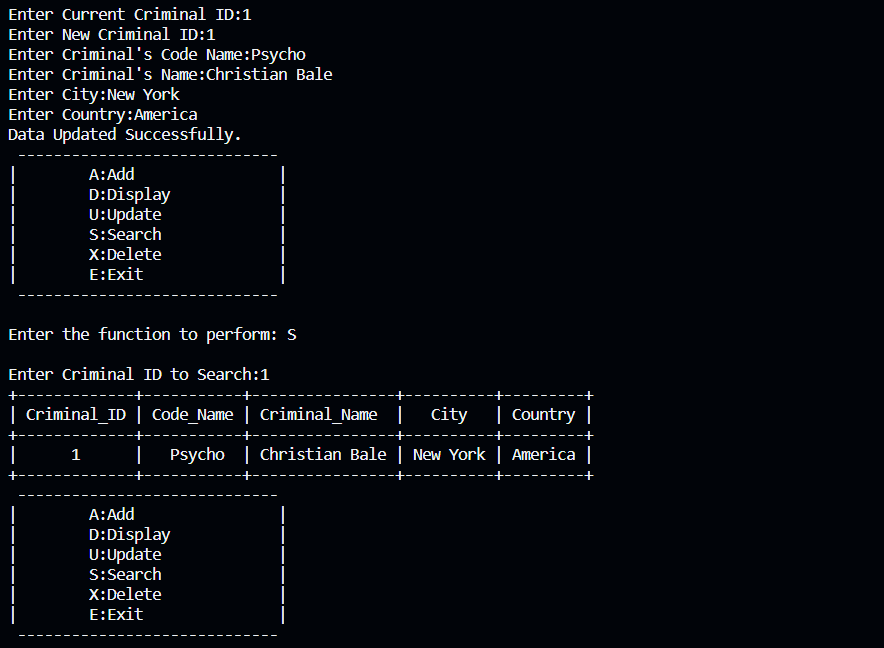
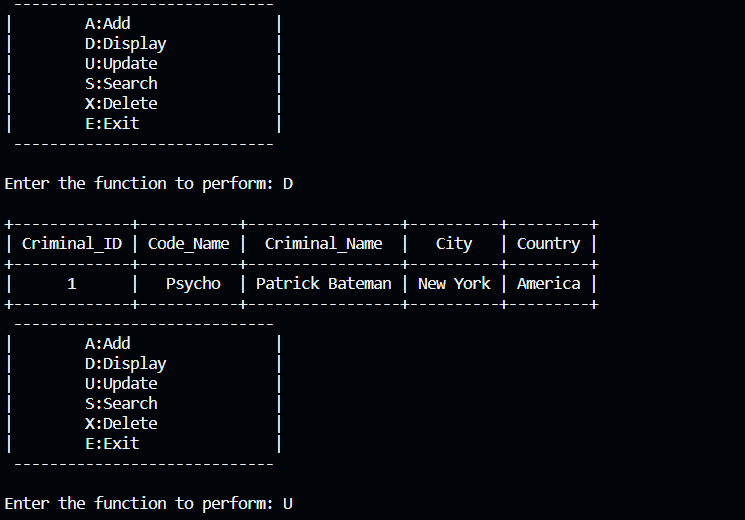


SEARCHING AND DELETING DATA

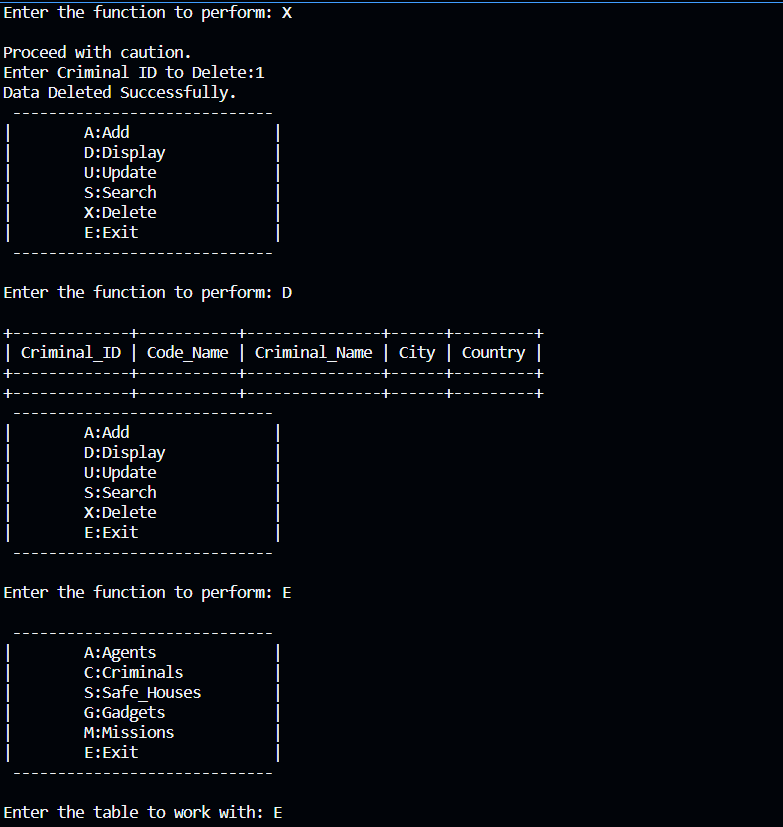
1. Table Criminals



ADDING DATA

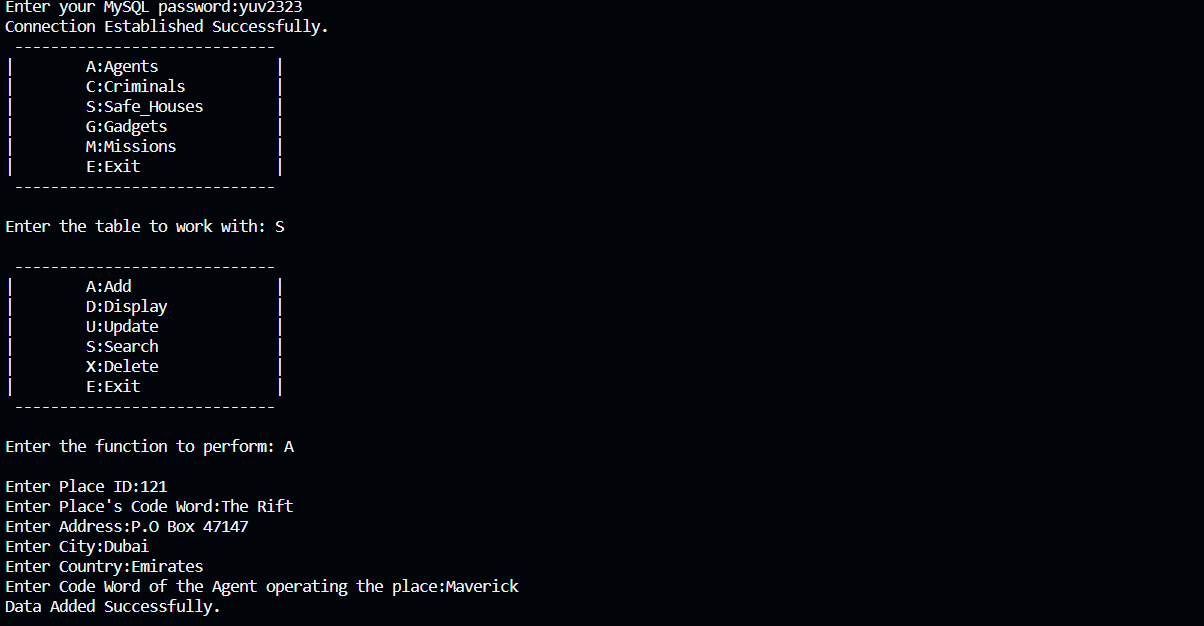


Displaying, updating and searching data

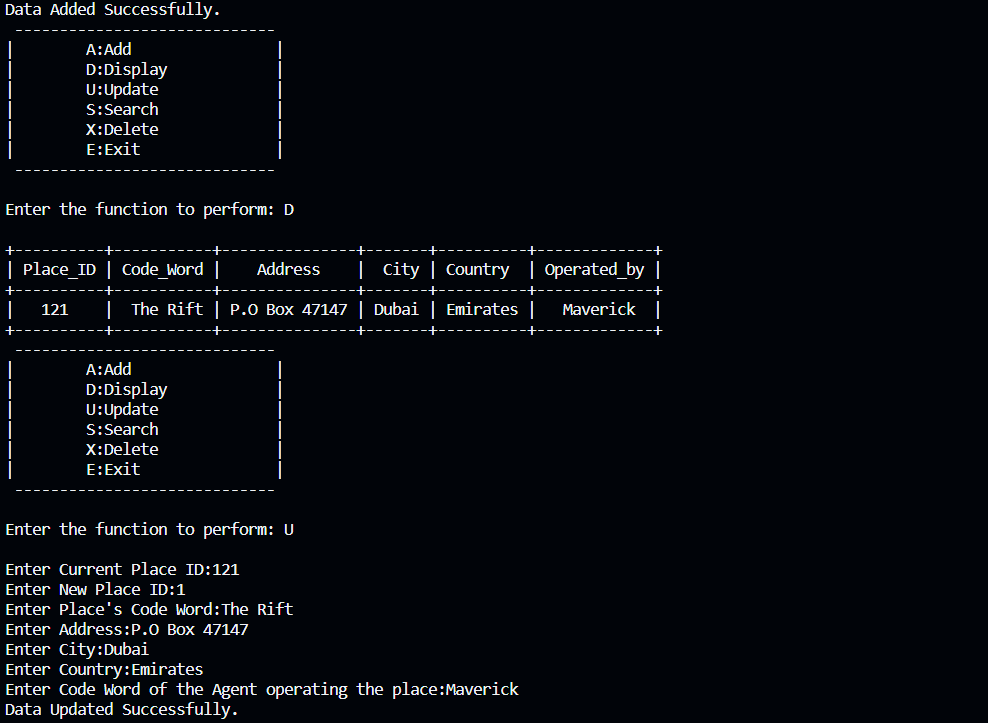


DELETING DATA

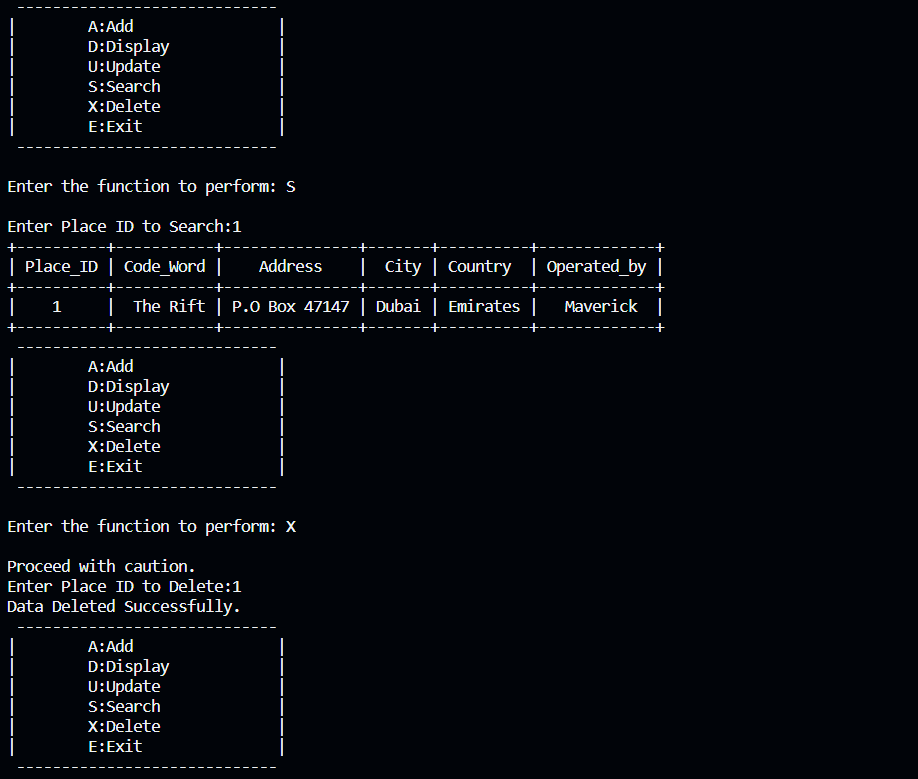
1. TABLE SAFE HOUSES



ADDING DATA

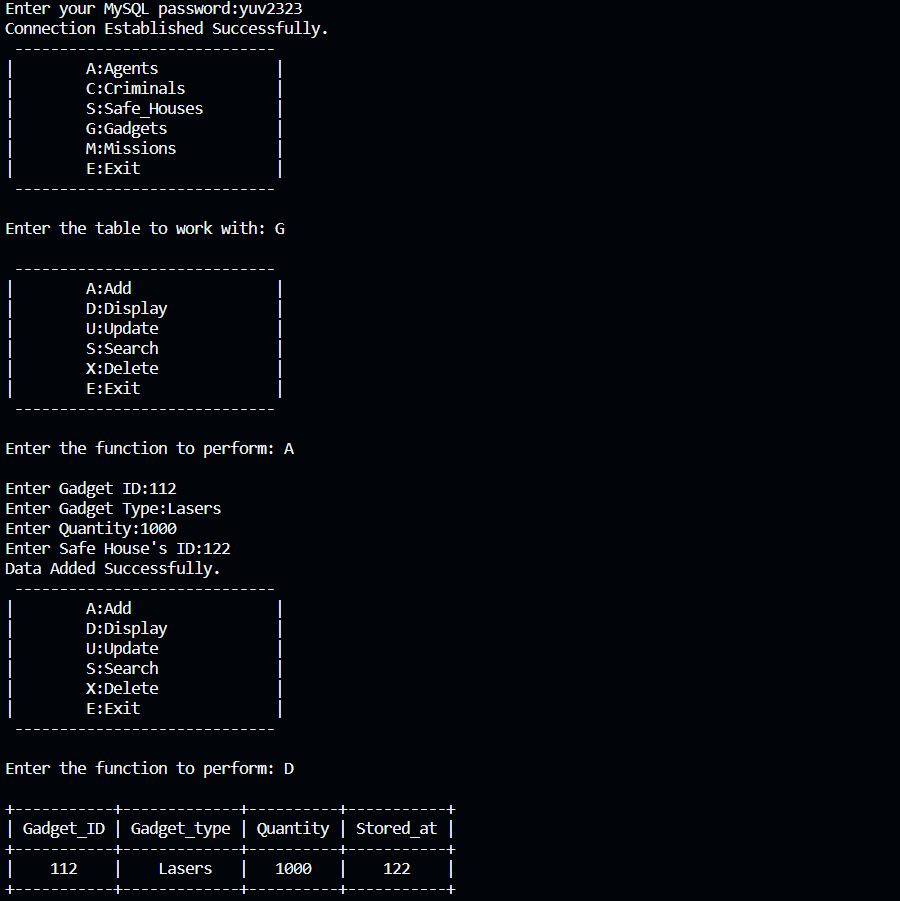


DISPLAYING AND UPDATING DATA

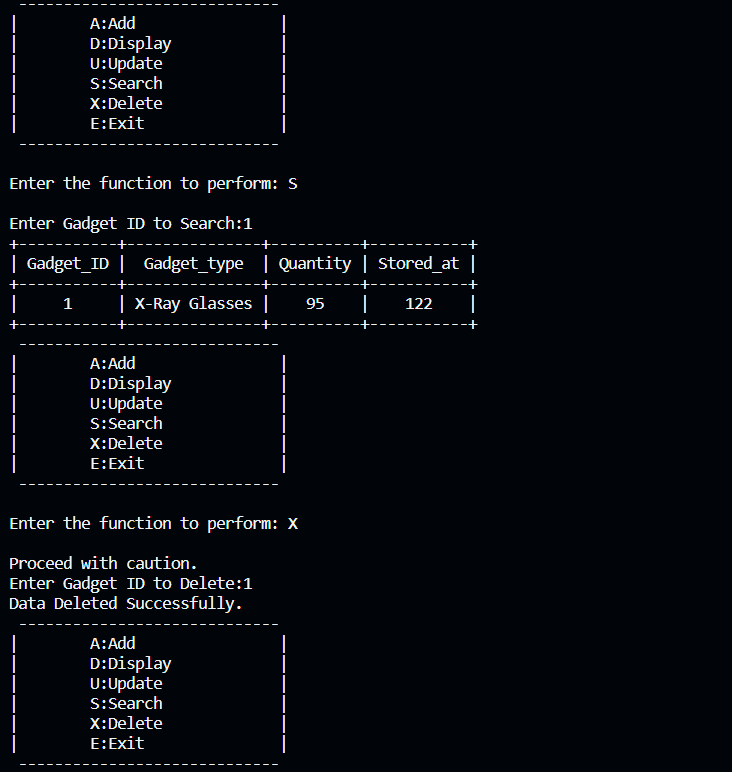


Searching and deleting data

1. Table Gadgets



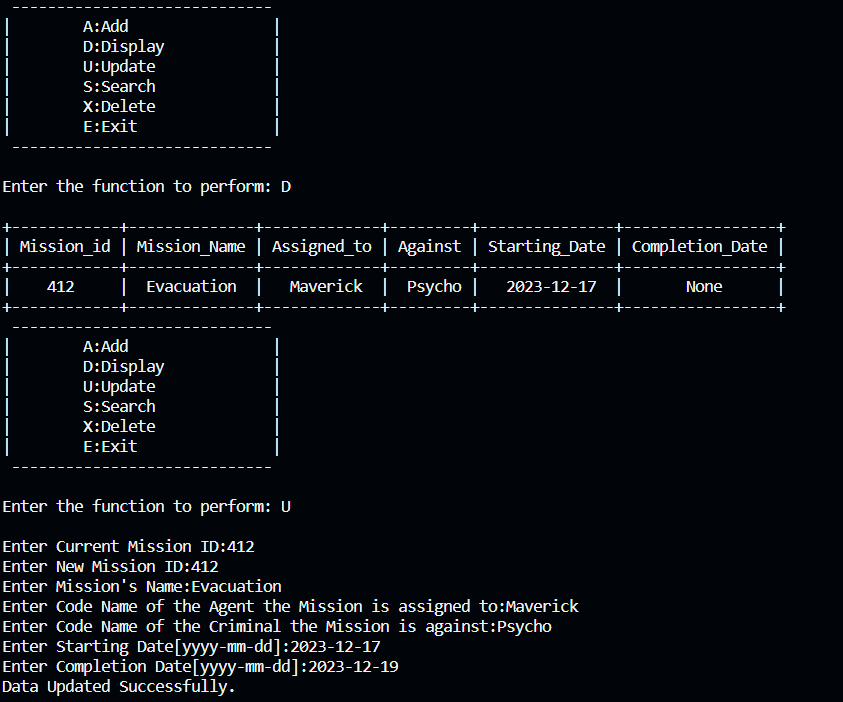
Adding and Displaying Data

Updating, Searching and Deleting data

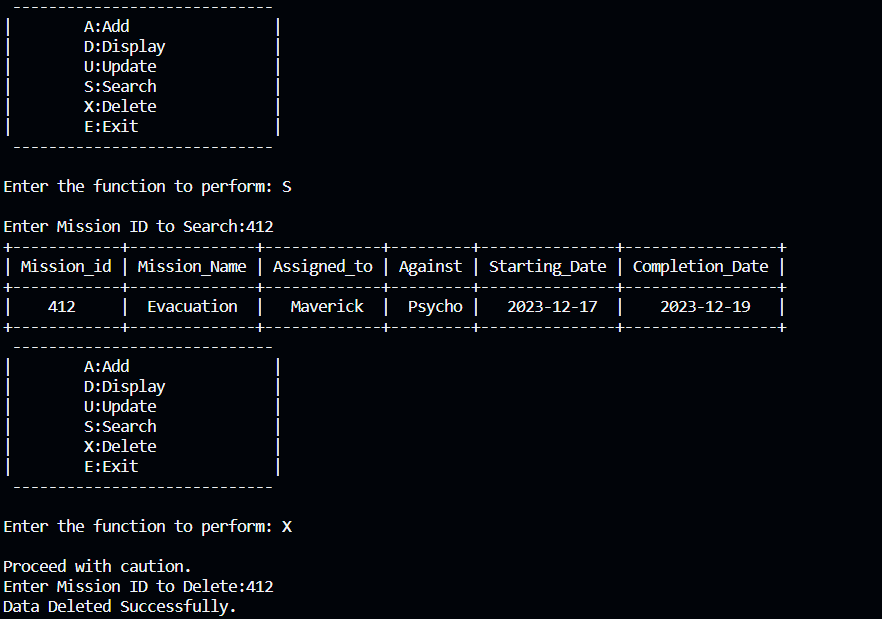
1. Table Missions



Adding Data



Displaying and Updating Data



Searching and Deleting Data

INDEX

1. INTRODUCTION
2. mODULES USED
3. CODE TO CREATE REQUIRED DATABASE AND TABLES
4. CODE TO WORK WITH TABLES