Python Case Study

Crime report and analysis system

-Muskan Saxena(PGET)

Entity Package

Evidence.py

```
from entity.incidents import Incidents

class Evidence(Incidents):
    def __init__(self):
        super().__init__()
        self.evidenceid=0
        self.description=' '
        self.locationfound=' '
        self.incidentid=0

    #Setters

    def set_evidenceid(self,value):
        self.evidenceid=value

    def set_description(self, value):
        self.description= value

    def set_locationfound(self, value):
        self.locationfound = value

    def set_incidentid(self, value):
        self.incidentid =value


    #Getters

    def get_evidenceid(self):
        return self.evidenceid

    def get_description(self):
        return self.description

    def get_locationfound(self):
        return self.locationfound

    def get_incidentid(self):
        return self.incidentid
    def str(self):
```

```
        return f'Evidence ID:{self.evidenceid}
Description:{self.description} Location Found:{self.locationfound}\n ' \
               f'Incident ID:{self.incidentid}'
```

incidents.py

```
from entity.suspects import Suspects
from entity.victims import Victims

class Incidents(Suspects, Victims):
    def __init__(self):
        super().__init__()
        self.incidentid=0
        self.incidenttype=' '
        self.incidentdate=' '
        self.location=' '
        self.description=' '
        self.status=' '
        self.victimid=0
        self.suspectid = 0
    #Setters

    def set_incidentid(self,value):
        self.incidentid=value

    def set_incidenttype(self, value):
        self.incidenttype = value

    def set_incidentdate(self, value):
        self.incidentdate = value

    def set_location(self, value):
        self.location= value

    def set_description(self, value):
        self.description = value

    def set_status(self, value):
        self.status = value

    def set_victimid(self, value):
        self.victimid = value

    def set_suspectid(self, value):
        self.suspectid = value

    #Getters

    def get_incidentid(self):
        return self.incidentid

    def get_incidenttype(self):
        return self.incidenttype

    def get_incidentdate(self):
        return self.incidentdate

    def get_location(self):
        return self.location
```

```
    def get_description(self):
        return self.description

    def get_status(self):
        return self.status

    def get_victimid(self):
        return self.victimid

    def get_suspectid(self):
        return self.suspectid

    def str(self):
        return f'Incident ID:{self.incidentid} Incident
Type:{self.incidenttype} Incident Date:{self.incidentdate}\n ' \
               f'Location:{self.location} Description:{self.description}
Status:{self.status} Victim ID:{self.victimid} Suspect ID:{self.suspectid}'
```

lawinforcementagencies.py

```
from entity.officers import Officers

class LawInforcementAgencies(Officers):
    def __init__(self):
        super().__init__()
        self.agencyid=0
        self.agencyname=' '
        self.jurisdiction=' '
        self.contactnumber=0
        self.officerid=0

    #Setters

    def set_agencyid(self,value):
        self.agencyid=value

    def set_agencyname(self, value):
        self.agencyname = value

    def set_jurisdiction(self, value):
        self.jurisdiction= value

    def set_contactnumber(self, value):
        self.contactnumber= value

    def set_officerid(self, value):
        self.officerid= value


    #Getters

    def get_agencyid(self):
        return self.agencyid
```

```
    def get_agencyname(self):
        return self.agencyname

    def get_jurisdiction(self):
        return self.jurisdiction

    def get_contactnumber(self):
        return self.contactnumber

    def get_officerid(self):
        return self.officerid


    def str(self):
        return f'Agency ID:{self.agencyid} Agency Name:{self.agencyname}
Jurisdiction:{self.jurisdiction}\n ' \
               f'Contact Number:{self.contactnumber} Officer
ID:{self.officerid}'
```

officers.py

```
from util.DBConnUtil import DBConnection
class Officers(DBConnection):
    def __init__(self):
        super().__init__()
        self.officerid=0
        self.firstname=' '
        self.lastname=' '
        self.badgenumber=0
        self.rank_=0
        self.contactnumber=0

    #Setters

    def set_officerid(self,value):
        self.officerid=value

    def set_firstname(self, value):
        self.firstname = value

    def set_lastname(self, value):
        self.lastname = value

    def set_badgenumber(self, value):
        self.badgenumber= value

    def set_rank_(self, value):
        self.rank_ = value

    def set_contactnumber(self, value):
        self.contactnumber = value



    #Getters

    def get_officerid(self):
```

```
        return self.officerid

    def get_firstname(self):
        return self.firstname

    def get_lastname(self):
        return self.lastname

    def get_badgenumber(self):
        return self.badgenumber

    def get_rank_(self):
        return self.rank_

    def get_contactnumber(self):
        return self.contactnumber

    def str(self):
        return f'Officer ID:{self.officerid} First Name:{self.firstname}
Last Name:{self.lastname}\n ' \
               f'Badge Number:{self.badgenumber} Rank:{self.rank_} Contact
Number:{self.contactnumber}'
```

reports.py

```
from entity.incidents import Incidents
from entity.officers import Officers
class Reports(Incidents, Officers):
    def __init__(self):
        super().__init__()
        self.reportid=0
        self.incidentid=0
        self.reportingofficer=0
        self.reportdate=' '
        self.reportdetails=' '
        self.status=' '

    #Setters

    def set_reportid(self,value):
        self.reportid=value

    def set_incidentid(self, value):
        self.incidentid = value

    def set_reportingofficer(self, value):
        self.reportingofficer= value

    def set_reportdate(self, value):
        self.reportdate= value

    def set_reportdetails(self, value):
        self.reportdetails= value

    def set_status(self, value):
        self.status= value


    #Getters
```

```
    def get_reportid(self):
        return self.reportid

    def get_incidentid(self):
        return self.incidentid

    def get_reportingofficer(self):
        return self.reportingofficer

    def get_reportdate(self):
        return self.reportdate

    def get_reportdetails(self):
        return self.reportdetails

    def get_status(self):
        return self.status



    def str(self):
        return f'Report ID:{self.reportid} Incident ID:{self.incidentid}
Reporting Officer:{self.reportingofficer}\n ' \
               f'Report Date:{self.reportdate} Report
Details:{self.reportdetails} Status:{self.status}'
```

suspects.py

```
from util.DBConnUtil import DBConnection
class Suspects(DBConnection):
    def __init__(self):
        super().__init__()
        self.suspectid=0
        self.firstname=' '
        self.lastname=' '
        self.dateofbirth=' '
        self.gender=' '
        self.contactnumber=0

    #Setters

    def set_suspectid(self,value):
        self.suspectid=value

    def set_firstname(self, value):
        self.firstname = value

    def set_lastname(self, value):
        self.lastname = value

    def set_dateofbirth(self, value):
        self.dateofbirth= value

    def set_gender(self, value):
        self.gender = value

    def set_contactnumber(self, value):
        self.contactnumber = value
```

```
    #Getters

    def get_suspectid(self):
        return self.suspectid

    def get_firstname(self):
        return self.firstname

    def get_lastname(self):
        return self.lastname

    def get_dateofbirth(self):
        return self.dateofbirth

    def get_gender(self):
        return self.gender

    def get_contactnumber(self):
        return self.contactnumber

    def str(self):
        return f'Suspect ID:{self.suspectid} First Name:{self.firstname}
Last Name:{self.lastname}\n ' \
               f'Date Of Birth:{self.dateofbirth} Gender:{self.gender}
Contact Number:{self.contactnumber}'
```

victims.py

```
from util.DBConnUtil import DBConnection
class Victims(DBConnection):
    def __init__(self):
        super().__init__()
        self.victimid=0
        self.firstname=' '
        self.lastname=' '
        self.dateofbirth=' '
        self.gender=' '
        self.contactnumber=0

    #Setters

    def set_victimid(self,value):
        self.victimid=value

    def set_firstname(self, value):
        self.firstname = value

    def set_lastname(self, value):
        self.lastname = value

    def set_dateofbirth(self, value):
        self.dateofbirth= value

    def set_gender(self, value):
        self.gender = value

    def set_contactnumber(self, value):
```

```
        self.contactnumber = value


    #Getters

    def get_victimid(self):
        return self.victimid

    def get_firstname(self):
        return self.firstname

    def get_lastname(self):
        return self.lastname

    def get_dateofbirth(self):
        return self.dateofbirth

    def get_gender(self):
        return self.gender

    def get_contactnumber(self):
        return self.contactnumber

    def str(self):
        return f'Victim ID:{self.victimid} First Name:{self.firstname} Last
Name:{self.lastname}\n ' \
               f'Date Of Birth:{self.dateofbirth} Gender:{self.gender}
Contact Number:{self.contactnumber}'
```

Dao Package

crimeanalysisserviceimpl.py

```
from dao.evidencedao import EvidenceDao
from dao.incidentsdao import IncidentsDao
from dao.lawinforcementagenciesdao import LawInforcementAgenciesDao
from dao.officersdao import OfficersDao
from dao.reportsdao import ReportsDao
from dao.suspectsdao import SuspectsDao
from dao.victimsdao import VictimsDao
from exception.incidentnumbernotfoundexception import
IncidentNumberNotFound

class
CrimeAnalysisServiceImpl(EvidenceDao,IncidentsDao,LawInforcementAgenciesDao
,OfficersDao,ReportsDao,SuspectsDao,VictimsDao,IncidentNumberNotFound):

#Update Incident Status

    def updateincidentstatus(self):
       try:
          self.open()
          incidentid = int(input('Input Incident ID to be Updated: '))
          self.status = input('Enter Status: ')
          data = [( self.status, incidentid)]
          update_str = '''UPDATE Incidents SET  status=%s
```

```
                            WHERE incidentid = %s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error updating Incident: {e}"

# Get Incidents within a range
    def getIncidentsInDateRange(self):
        try:
            print('Enter Start Date (YYYY-MM-DD): ')
            startdate = input()
            print('Enter End Date (YYYY-MM-DD): ')
            enddate = input()
            self.open()
            select_str = f'''SELECT * FROM Incidents WHERE incidentdate
BETWEEN %s AND %s'''
            self.stmt.execute(select_str, (startdate, enddate))
            records = self.stmt.fetchall()
            self.close()
            print("Records in table")
            for i in records:
                print(i)
        except Exception as e:
             print(e)



#Search Incidents
    def searchIncidents(self):
        try:
            self.open()
            incidenttype = input('Input Incident Type to see the details: ')
            data = f"'{incidenttype}'"
            select_str = f'''SELECT * FROM Incidents WHERE
incidenttype={data} '''
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            return records

        except Exception as e:
            print(e)

# Generate Incident Reports
    def generatIncidentReport(self, incidentid):

        print('Enter Incident ID to get Reports: ')
        try:
            self.open()
            select_str = f'''SELECT * FROM Reports WHERE
incidentid={incidentid}'''
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            if not records:
                raise IncidentNumberNotFound
            return records
        except IncidentNumberNotFound as e:
            return e
```

```
        except Exception as e:
            print(e)
            return None
```

evidencedao.py

```
from entity.evidence import Evidence

class EvidenceDao(Evidence):
    def __init__(self):
        super().__init__()

    def perform_evidence_actions(self):
        while True:
            print("(Evidence) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT
0.EXIT")
            ch = int(input("Enter choice: "))
            if ch == 1:
                self.create_evidence_table()
            elif ch == 2:
                print(self.add_evidence())
            elif ch == 3:
                print(self.update_evidence())
            elif ch == 4:
                print(self.delete_evidence())
            elif ch == 5:
                self.select_evidence()
            elif ch == 0:
                break
            else:
                print("Invalid choice")

    def create_evidence_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Evidence (
                evidenceid INT  PRIMARY KEY,
                description VARCHAR(255),
                locationfound VARCHAR(255),
                incidentid INT,
                FOREIGN KEY(incidentid) REFERENCES Incidents(incidentid) ON
DELETE CASCADE ON UPDATE CASCADE)'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print('Evidence Table Created successfully.')
        except Exception as e:
            print(f"Error creating Evidence table: {e}")

    def add_evidence(self):
        try:
            self.open()
            self.evidenceid = int(input('Enter Evidence ID: '))
            self.description = input('Enter Description: ')
            self.locationfound = input('Enter Location Found: ')
            self.incidentid = int(input('Enter Incident ID: '))

            data = [(self.evidenceid, self.description, self.locationfound,
self.incidentid)]
```

```
            insert_str = '''INSERT INTO
Evidence(evidenceid,description,locationfound,incidentid)
                            VALUES(%s, %s, %s, %s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error adding Evidence: {e}"

    def update_evidence(self):
        try:
            self.open()
            evidenceid = int(input('Input Evidence ID to be Updated: '))
            self.description = int(input('Enter Description: '))
            self.locationfound = int(input('Enter Location Found: '))
            self.incidentid = input('Enter Incident ID: ')

            data = [(self.description, self.locationfound,
self.incidentid,evidenceid)]
            update_str = '''UPDATE Evidence SET description=%s,
locationfound=%s, incidentid=%s
                            WHERE evidenceid = %s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error updating Evidence: {e}"

    def delete_evidence(self):
        try:
            self.open()
            evidenceid = int(input('Input Evidence ID to be Deleted: '))
            delete_str = f'''DELETE FROM Evidence WHERE evidenceid =
{evidenceid}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error deleting Evidence: {e}"

    def select_evidence(self):
        try:
            select_str = '''SELECT * FROM Evidence'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print('Records In Evidence Table:')
            for i in records:
                print(i)
        except Exception as e:
            print(f"Error selecting Evidence: {e}")
```

incidentsdao.py

```
from entity.incidents import Incidents

class IncidentsDao(Incidents):
    def __init__(self):
        super().__init__()

    def perform_incidents_actions(self):
        while True:
            print("(Incidents) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT
0.EXIT")
            ch = int(input("Enter choice: "))
            if ch == 1:
                self.create_incidents_table()
            elif ch == 2:
                print(self.add_incidents())
            elif ch == 3:
                print(self.update_incidents())
            elif ch == 4:
                print(self.delete_incidents())
            elif ch == 5:
                self.select_incidents()
            elif ch == 0:
                break
            else:
                print("Invalid choice")

    def create_incidents_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Incidents (
                incidentid INT  PRIMARY KEY,
                incidenttype VARCHAR(50),
                incidentdate DATE,
                location VARCHAR(100),
                description VARCHAR(255),
                status VARCHAR(50),
                victimid INT,
                suspectid INT,
                FOREIGN KEY(victimid) REFERENCES Victims(victimid) ON
DELETE CASCADE ON UPDATE CASCADE,
                FOREIGN KEY(suspectid) REFERENCES Suspects(suspectid) ON
DELETE CASCADE ON UPDATE CASCADE)'''

            self.open()
            self.stmt.execute(create_str)
            self.close()
            print('Incidents Table Created successfully.')
        except Exception as e:
            print(f"Error creating Incidents table: {e}")

    def add_incidents(self):
        try:
            self.open()
            self.incidentid = int(input('Enter Incident ID: '))
            self.incidenttype = input('Enter Incident Type: ')
            self.incidentdate = input('Enter Incident Date: ')
```

```
            self.location= input('Enter Location: ')
            self.description = input('Enter Description: ')
            self.status = input('Enter Status: ')
            self.victimid = int(input('Enter Victim ID: '))
            self.suspectid = int(input('Enter Suspect ID: '))
            data = [(self.incidentid, self.incidenttype, self.incidentdate,
self.location,self.description,self.status,self.victimid,self.suspectid)]
            insert_str = '''INSERT INTO
Incidents(incidentid,incidenttype,incidentdate,location,description,status,
victimid,suspectid)
                            VALUES(%s, %s, %s, %s,%s, %s, %s, %s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error adding Incident{e}"

    def update_incidents(self):
        try:
            self.open()
            incidentid = int(input('Input Incident ID to be Updated: '))
            self.incidenttype = input('Enter Description: ')
            self.incidentdate = input('Enter Incident Date: ')
            self.location = input('Enter Location: ')
            self.description = input('Enter Description: ')
            self.status = input('Enter Status: ')
            self.victimid = int(input('Enter Victim ID: '))
            self.suspectid= int(input('Enter Suspect ID: '))
            data = [(self.incidenttype, self.incidentdate, self.location
,self.description,self.status,self.victimid,self.suspectid,incidentid)]
            update_str = '''UPDATE Incidents SET
incidenttype=s%,incidentdate=s%,location=s%,description=%s, status=%s,
victimid=%s,suspectid=s%
                            WHERE incidentid = %s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error updating Incident: {e}"

    def delete_incidents(self):
        try:
            self.open()
            incidentid = int(input('Input Incident ID to be Deleted: '))
            delete_str = f'''DELETE FROM Incident WHERE incidentid =
{incidentid}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error deleting Incident: {e}"

    def select_incidents(self):
        try:
            select_str = '''SELECT * FROM Incidents'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
```

```
            self.close()
            print('Records In Incidents Table:')
            for i in records:
                print(i)
        except Exception as e:
            print(f"Error selecting Incidents: {e}")
```

lawinforcementagenciesdao.py

```
from entity.lawinforcementagencies import LawInforcementAgencies

class LawInforcementAgenciesDao(LawInforcementAgencies):
    def __init__(self):
        super().__init__()

    def perform_lia_actions(self):
        while True:
            print("(LawInforcementAgencies) 1.CREATE 2.INSERT 3.UPDATE
4.DELETE 5.SELECT 0.EXIT")
            ch = int(input("Enter choice: "))
            if ch == 1:
                self.create_lia_table()
            elif ch == 2:
                print(self.add_lia())
            elif ch == 3:
                print(self.update_lia())
            elif ch == 4:
                print(self.delete_lia())
            elif ch == 5:
                self.select_lia()
            elif ch == 0:
                break
            else:
                print("Invalid choice")

    def create_lia_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS
LawInforcementAgencies (
                agencyid INT  PRIMARY KEY,
                agencyname VARCHAR(100),
                jurisdiction VARCHAR(255),
                contactnumber VARCHAR(15),
                officerid INT,
                FOREIGN KEY(officerid) REFERENCES Officers(officerid) ON
DELETE CASCADE ON UPDATE CASCADE)'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print('LawInforcementAgencies Table Created successfully.')
        except Exception as e:
            print(f"Error creating LawInforcementAgencies table: {e}")

    def add_lia(self):
        try:
            self.open()
            self.agencyid = int(input('Enter Agency ID: '))
            self.agencyname = input('Enter Agency Name: ')
            self.jurisdiction = input('Enter Jurisdiction: ')
            self.contactnumber = input('Enter Contact Number: ')
            self.officerid=int(input('Enter Officer ID'))
```

```
            data = [(self.agencyid, self.agencyname, self.jurisdiction,
self.contactnumber,self.officerid)]
            insert_str = '''INSERT INTO
LawInforcementAgencies(agencyid,agencyname,jurisdiction,contactnumber,offic
erid)
                            VALUES(%s, %s, %s, %s,%s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error adding LawInforcementAgencies: {e}"

    def update_lia(self):
        try:
            self.open()
            agencyid = int(input('Input Agency ID to be Updated: '))
            self.agencyname = input('Enter Agency Name: ')
            self.jurisdiction = input('Enter Jurisdiction: ')
            self.contactnumber = input('Enter Contact Number: ')
            self.officerid = int(input('Enter Officer ID: '))

            data = [(self.agencyname, self.jurisdiction,
self.contactnumber,self.officerid,agencyid)]
            update_str = '''UPDATE LawInforcementAgencies SET
agencyname=%s, jurisdiction=%s, contactnumber=%s, officerid=s%
                            WHERE agencyid = %s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error updating LawInforcementAgencies: {e}"

    def delete_lia(self):
        try:
            self.open()
            agencyid = int(input('Input Agency ID to be Deleted: '))
            delete_str = f'''DELETE FROM LawInforcementAgencies WHERE
agencyid = {agencyid}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error deleting LawInforcementAgencies: {e}"

    def select_lia(self):
        try:
            select_str = '''SELECT * FROM LawInforcementAgencies'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print('Records In LawInforcementAgencies Table:')
            for i in records:
                print(i)
        except Exception as e:
            print(f"Error selecting LawInforcementAgencies: {e}")
```

officersdao.py

```
from entity.officers import Officers

class OfficersDao(Officers):
    def __init__(self):
        super().__init__()

    def perform_officers_actions(self):
        while True:
            print("(Officers) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT
0.EXIT")
            ch = int(input("Enter choice: "))
            if ch == 1:
                print(self.create_officers_table())
            elif ch == 2:
                print(self.add_officers())
            elif ch == 3:
                print(self.update_officers())
            elif ch == 4:
                print(self.delete_officers())
            elif ch == 5:
                self.select_officers()
            elif ch == 0:
                break
            else:
                print("Invalid choice")

    def create_officers_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Officers (
                officerid INT  PRIMARY KEY,
                firstname VARCHAR(50),
                lastname VARCHAR(50),
                badgenumber INT,
                rank_ INT,
                contactnumber VARCHAR(15))'''

            self.open()
            self.stmt.execute(create_str)
            self.close()
            print('Officers Table Created successfully.')
        except Exception as e:
            print(f"Error creating Officers table: {e}")

    def add_officers(self):
        try:
            self.open()
            self.officerid = int(input('Enter Officer ID: '))
            self.firstname = input('Enter First Name: ')
            self.lastname = input('Enter Last Name: ')
            self.bagdenumber= int(input('Enter Badge Number: '))
            self.rank_ = int(input('Enter Rank: '))
            self.contactnumber = input('Enter Contact Number: ')
            data = [(self.officerid, self.firstname, self.lastname,
self.badgenumber,self.rank_,self.contactnumber)]
            insert_str = '''INSERT INTO
```

```
Officers(officerid,firstname,lastname,badgenumber,rank_,contactnumber)
                            VALUES(%s, %s, %s, %s,%s, %s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error adding Officer{e}"

    def update_officers(self):
        try:
            self.open()
            officerid = int(input('Input Officer ID to be Updated: '))
            self.firstname = input('Enter First Name: ')
            self.lastname = input('Enter Last Name: ')
            self.badgenumber = int(input('Enter Location: '))
            self.rank_ = int(input('Enter Rank: '))
            self.contactnumber = input('Enter Contact Number: ')
            data = [(self.firstname, self.lastname, self.badgenumber
,self.rank_,self.contactnumber,officerid)]
            update_str = '''UPDATE Officers SET
firstname=%s,lastname=%s,badgenumber=%s,rank_=%s, contactnumber=%s
                            WHERE officerid = %s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error updating Officer: {e}"

    def delete_officers(self):
        try:
            self.open()
            officerid = int(input('Input Officer ID to be Deleted: '))
            delete_str = f'''DELETE FROM Officers WHERE officerid =
{officerid}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error deleting Officer: {e}"

    def select_officers(self):
        try:
            select_str = '''SELECT * FROM Officers'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print('Records In Officers Table:')
            for i in records:
                print(i)
        except Exception as e:
            print(f"Error selecting Officers: {e}")
```

reportersdao.py

```
from entity.reports import Reports

class ReportsDao(Reports):
    def __init__(self):
        super().__init__()

    def perform_reports_actions(self):
        while True:
            print("(Reports) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT
0.EXIT")
            ch = int(input("Enter choice: "))
            if ch == 1:
                self.create_reports_table()
            elif ch == 2:
                print(self.add_reports())
            elif ch == 3:
                print(self.update_reports())
            elif ch == 4:
                print(self.delete_reports())
            elif ch == 5:
                self.select_reports()
            elif ch == 0:
                break
            else:
                print("Invalid choice")

    def create_reports_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Reports (
                reportid INT  PRIMARY KEY,
                incidentid INT,
                reportingofficer INT,
                reportdate date,
                reportdetails VARCHAR(200),
                status VARCHAR(50),
                FOREIGN KEY(reportingofficer) REFERENCES
Officers(officerid) ON DELETE CASCADE ON UPDATE CASCADE)'''

            self.open()
            self.stmt.execute(create_str)
            self.close()
            print('Reports Table Created successfully.')
        except Exception as e:
            print(f"Error creating Reports table: {e}")

    def add_reports(self):
        try:
            self.open()
            self.reportid = int(input('Enter Report ID: '))
            self.incidentid= int(input('Enter Incident ID: '))
            self.reportingofficer= int(input('Enter Reporting Officer ID:
'))
            self.reportdate= input('Enter Report Date: ')
            self.reportdetails = input('Enter Report Details: ')
            self.status = input('Enter Status: ')
            data = [(self.reportid, self.incidentid, self.reportingofficer,
self.reportdate,self.reportdetails,self.status)]
            insert_str = '''INSERT INTO
Reports(reportid,incidentid,reportingofficer,reportdate,reportdetails,statu
s)
                            VALUES(%s, %s, %s, %s,%s, %s)'''
```

```
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error adding Report{e}"

    def update_reports(self):
        try:
            self.open()
            reportid = int(input('Input Report ID to be Updated: '))
            self.incidentid = input('Enter Incident ID: ')
            self.reportingofficer = input('Enter Reporting Officer: ')
            self.reportdate = int(input('Enter Report Date: '))
            self.reportdetails = int(input('Enter Report Details: '))
            self.status = input('Enter Status: ')
            data = [(self.incidentid, self.reportingofficer,
self.reportdate ,self.reportdetails,self.status,reportid)]
            update_str = '''UPDATE Reports SET
incidentid=s%,reportingofficer=s%,reportdate=s%,reportdetails=%s, status=%s
                            WHERE reportid = %s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error updating Report: {e}"

    def delete_reports(self):
        try:
            self.open()
            reportid = int(input('Input Report ID to be Deleted: '))
            delete_str = f'''DELETE FROM Reports WHERE reportid =
{reportid}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error deleting Report: {e}"

    def select_reports(self):
        try:
            select_str = '''SELECT * FROM Reports'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print('Records In Reports Table:')
            for i in records:
                print(i)
        except Exception as e:
            print(f"Error selecting Reports: {e}")
```

suspectsdao.py

```
from entity.suspects import Suspects
```

```
class SuspectsDao(Suspects):
    def __init__(self):
        super().__init__()

    def perform_suspects_actions(self):
        while True:
            print("(Suspects) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT
0.EXIT")
            ch = int(input("Enter choice: "))
            if ch == 1:
                self.create_suspects_table()
            elif ch == 2:
                print(self.add_suspects())
            elif ch == 3:
                print(self.update_suspects())
            elif ch == 4:
                print(self.delete_suspects())
            elif ch == 5:
                self.select_suspects()
            elif ch == 0:
                break
            else:
                print("Invalid choice")

    def create_suspects_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Suspects (
                suspectid INT  PRIMARY KEY,
                firstname VARCHAR(50),
                lastname VARCHAR(50),
                dateofbirth DATE,
                gender VARCHAR(10),
                contactnumber VARCHAR(15))'''

            self.open()
            self.stmt.execute(create_str)
            self.close()
            print('Suspects Table Created successfully.')
        except Exception as e:
            print(f"Error creating Suspects table: {e}")

    def add_suspects(self):
        try:
            self.open()
            self.suspectid = int(input('Enter Suspect ID: '))
            self.firstname = input('Enter First Name: ')
            self.lastname = input('Enter Last Name: ')
            self.dateofbirth= input('Enter Date Of Birth: ')
            self.gender = input('Enter Gender: ')
            self.contactnumber = input('Enter Contact Number: ')
            data = [(self.suspectid, self.firstname, self.lastname,
self.dateofbirth,self.gender,self.contactnumber)]
            insert_str = '''INSERT INTO
Suspects(suspectid,firstname,lastname,dateofbirth,gender,contactnumber)
                            VALUES(%s, %s, %s, %s,%s, %s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error adding Suspect{e}"
```

```
    def update_suspects(self):
        try:
            self.open()
            suspectid = int(input('Input Suspect ID to be Updated: '))
            self.firstname = input('Enter First Name: ')
            self.lastname = input('Enter Last Name: ')
            self.dateofbirth = input('Enter Date Of Birth: ')
            self.gender = input('Enter Gender: ')
            self.contactnumber = input('Enter Contact Number: ')
            data = [(self.firstname, self.lastname, self.dateofbirth
,self.gender,self.contactnumber,suspectid)]
            update_str = '''UPDATE Suspects SET
firstname=%s,lastname=%s,dateofbirth=%s,gender=%s, contactnumber=%s
                            WHERE suspectid = %s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error updating Suspect: {e}"

    def delete_suspects(self):
        try:
            self.open()
            suspectid = int(input('Input Suspect ID to be Deleted: '))
            delete_str = f'''DELETE FROM Suspects WHERE suspectid =
{suspectid}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error deleting Suspect: {e}"

    def select_suspects(self):
        try:
            select_str = '''SELECT * FROM Suspects'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print('Records In Suspects Table:')
            for i in records:
                print(i)
        except Exception as e:
            print(f"Error selecting Suspects: {e}")
```

victimsdao.py

```
from entity.victims import Victims

class VictimsDao(Victims):
    def __init__(self):
        super().__init__()
```

```
    def perform_victims_actions(self):
        while True:
            print("(Victims) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT
0.EXIT")
            ch = int(input("Enter choice: "))
            if ch == 1:
                print(self.create_victims_table())
            elif ch == 2:
                print(self.add_victims())
            elif ch == 3:
                print(self.update_victims())
            elif ch == 4:
                print(self.delete_victims())
            elif ch == 5:
                self.select_victims()
            elif ch == 0:
                break
            else:
                print("Invalid choice")

    def create_victims_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Victims (
                victimid INT  PRIMARY KEY,
                firstname VARCHAR(50),
                lastname VARCHAR(50),
                dateofbirth DATE,
                gender VARCHAR(10),
                contactnumber VARCHAR(15))'''

            self.open()
            self.stmt.execute(create_str)
            self.close()
            print('Victims Table Created successfully.')
        except Exception as e:
            print(f"Error creating Victims table: {e}")

    def add_victims(self):
        try:
            self.open()
            self.victimid = int(input('Enter Victim ID: '))
            self.firstname = input('Enter First Name: ')
            self.lastname = input('Enter Last Name: ')
            self.dateofbirth= input('Enter Date Of Birth: ')
            self.gender = input('Enter Gender: ')
            self.contactnumber = input('Enter Contact Number: ')
            data = [(self.victimid, self.firstname, self.lastname,
self.dateofbirth,self.gender,self.contactnumber)]
            insert_str = '''INSERT INTO
Victims(victimid,firstname,lastname,dateofbirth,gender,contactnumber)
                            VALUES(%s, %s, %s, %s,%s, %s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error adding Victims{e}"

    def update_victims(self):
        try:
```

```
            self.open()
            victimid = int(input('Input Victim ID to be Updated: '))
            self.firstname = input('Enter First Name: ')
            self.lastname = input('Enter Last Name: ')
            self.dateofbirth = input('Enter Date Of Birth: ')
            self.gender = input('Enter Gender: ')
            self.contactnumber = input('Enter Contact Number: ')
            data = (self.firstname, self.lastname, self.dateofbirth
,self.gender,self.contactnumber,victimid)
            update_str = '''UPDATE Victims SET
firstname=%s,lastname=%s,dateofbirth=%s,gender=%s, contactnumber=%s
                            WHERE victimid = %s'''
            self.stmt.execute(update_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error updating Victim: {e}"

    def delete_victims(self):
        try:
            self.open()
            victimid = int(input('Input Suspect ID to be Deleted: '))
            delete_str = f'''DELETE FROM Suspects WHERE suspectid =
{victimid}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return f"Error deleting Victim: {e}"

    def select_victims(self):
        try:
            select_str = '''SELECT * FROM Victims'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print('Records In Victims Table:')
            for i in records:
                print(i)
        except Exception as e:
            print(f"Error selecting Victims: {e}")
```

Exception Package

incidentnumbernotfound.py

```
class IncidentNumberNotFound(Exception):
    def __init__(self):
        super().__init__(f'Incident ID not found')
```

Main Package

main.py

```
from dao.crimeanalysisserviceimpl import CrimeAnalysisServiceImpl
from dao.evidencedao import EvidenceDao
from dao.incidentsdao import IncidentsDao
from dao.lawinforcementagenciesdao import LawInforcementAgenciesDao
from dao.officersdao import OfficersDao
from dao.reportsdao import ReportsDao
from dao.suspectsdao import SuspectsDao
from dao.victimsdao import VictimsDao
from exception.incidentnumbernotfoundexception import
IncidentNumberNotFound
from util.DBConnUtil import DBConnection

def main():

    dbconnection = DBConnection()

    try:
        dbconnection.open()
        print("--Database Is Connected:--")
    except Exception as e:
        print(e)

    try:
        print("=" * 30)
        print("Crime Analysis and Reporting System")
        print("=" * 30)
        print("Welcome to Crime Analysis and Reporting System!")

        crime_report_system = CrimeAnalysisServiceImpl()

        while True:

print("1.Incidents\n2.Victims\n3.Suspects\n4.LawInforcementAgencies\n5.Offi
cers\n6.Evidence\n7.Reports\n0.I want to Manipulate the Data")
            ch = int(input("Enter choice: "))
            if ch == 1:
                i = IncidentsDao()
                i.perform_incidents_actions()
            elif ch == 2:
                v = VictimsDao()
                v.perform_victims_actions()
            elif ch == 3:
                s = SuspectsDao()
                s.perform_suspects_actions()
            elif ch == 4:
```

```
                l = LawInforcementAgenciesDao()
                l.perform_lia_actions()
            elif ch == 5:
                o = OfficersDao()
                o.perform_officers_actions()
            elif ch == 6:
                e = EvidenceDao()
                e.perform_evidence_actions()
            elif ch == 7:
                r = ReportsDao()
                r.perform_reports_actions()
            elif ch == 0:
                break
            else:
                print("Invalid choice")

        crime_report_system=CrimeAnalysisServiceImpl()

        while True:
            print("=" * 10)
            print("---MENU---")
            print("=" * 10)
            print("1.Update Incident Status\n2.Get Incidents in
Range\n3.Search Incidents\n4.Generate Incident Reports\n0.Exit")
            ch = int(input("Enter choice: "))
            if ch == 1:
                print(crime_report_system.updateincidentstatus())
            elif ch == 2:
                print(crime_report_system.getIncidentsInDateRange())
            elif ch == 3:
                print(crime_report_system.searchIncidents())
            elif ch == 4:

print(crime_report_system.generatIncidentReport(int(input('Enter Incident
ID to see Reports: '))))
            elif ch == 0:
                break
            else:
                print("Invalid choice")

    except IncidentNumberNotFound as e:
        print(e)

    finally:
        dbconnection.close()
        print("Thankyou for visiting Crime Report and Analysis System!")
        print("--Connection Is Closed:--")


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

Util Package

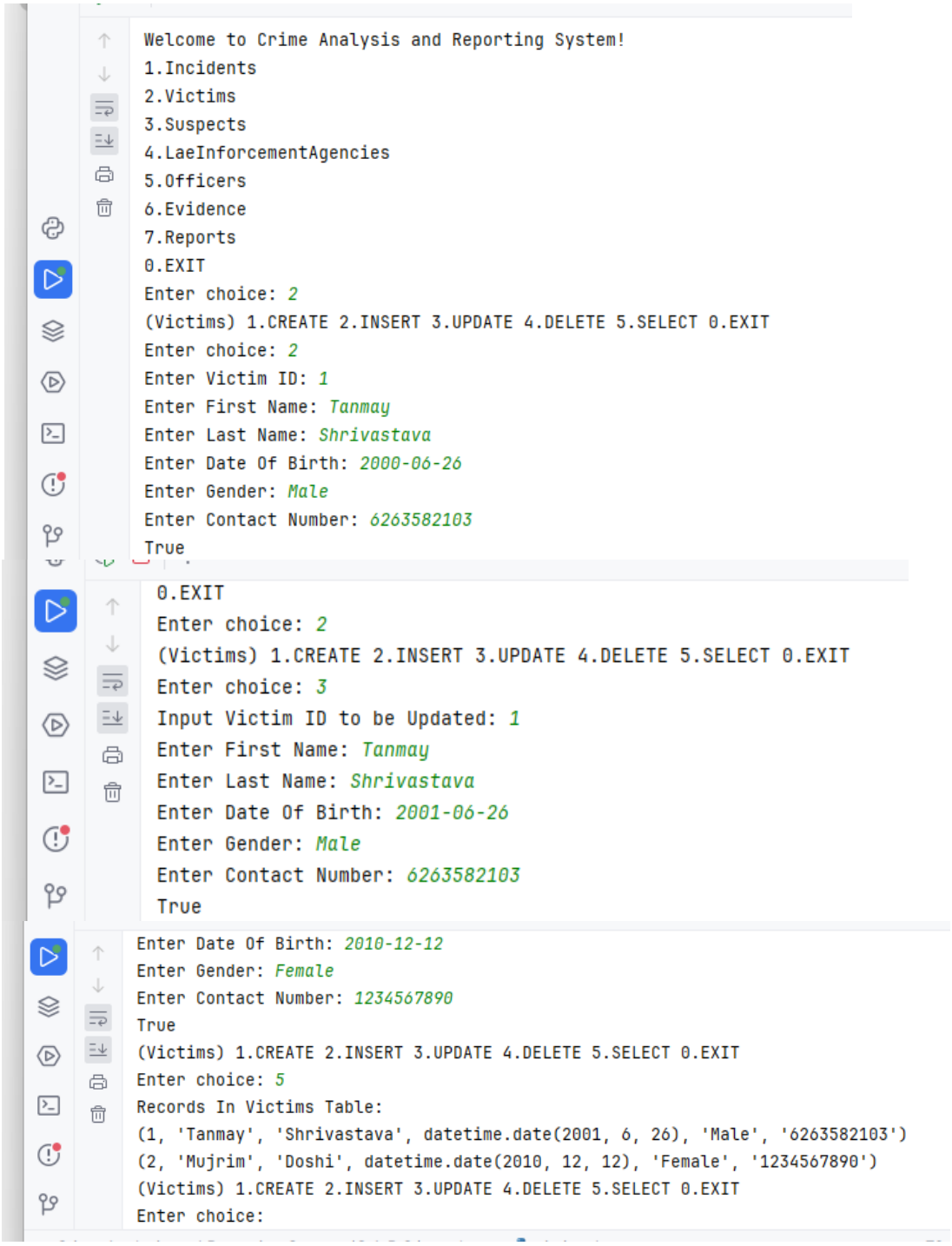
DBConnUtil.py

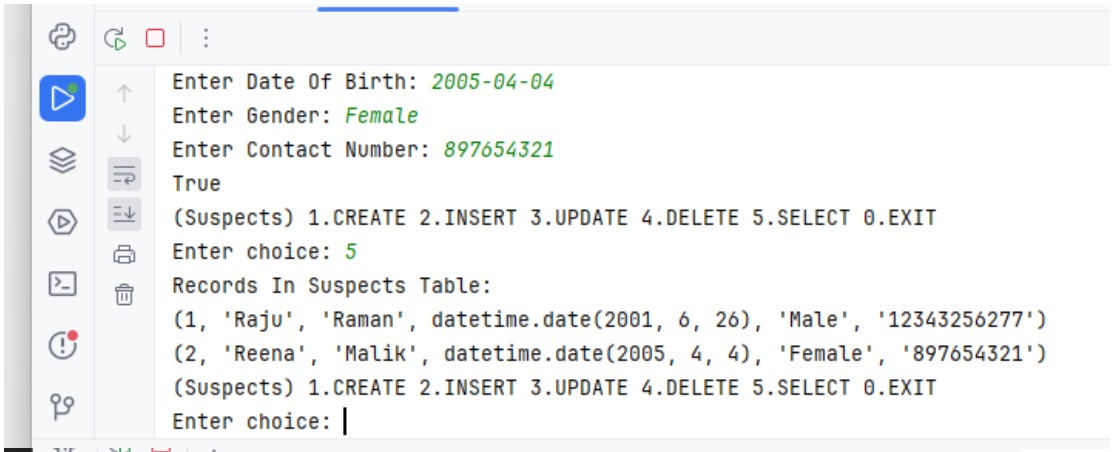
```
import sys
import mysql.connector as sql
from util.DBPropertyUtil import PropertyUtil


class DBConnection:
    def open(self):
        try:
            connection_properties=PropertyUtil.getConnectionString()
            self.conn=sql.connect(**connection_properties)
            self.stmt=self.conn.cursor()
        except Exception as e:
            print(str(e) + '--Database Is Not Connected:--')
            sys.exit(1)

    def close(self):
        self.conn.close()
```

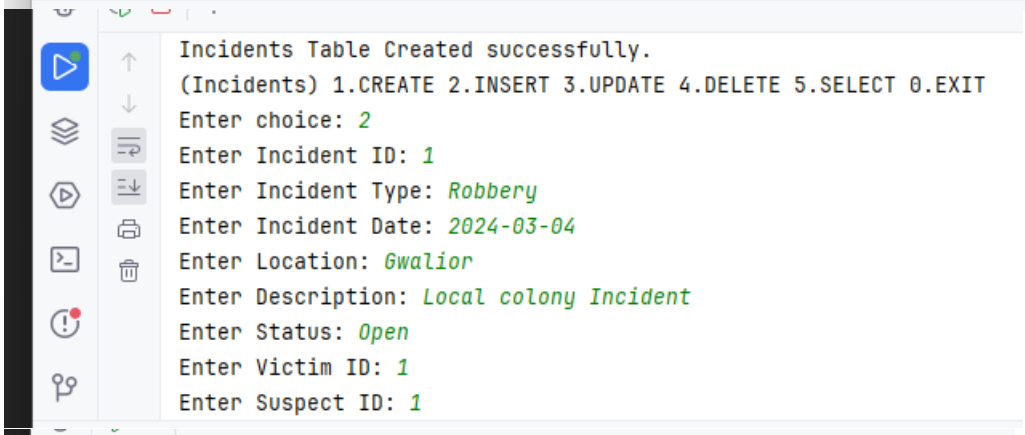
DBPropertyUtil.py

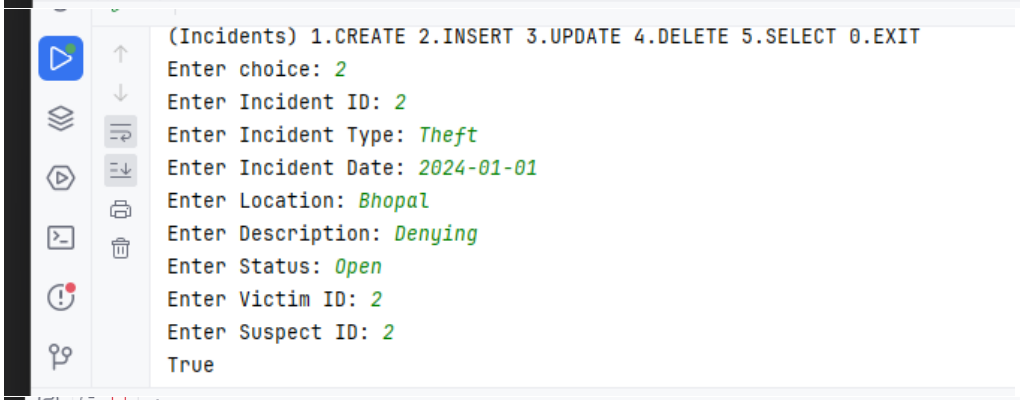
```
class PropertyUtil:
    connection_properties= None

    @staticmethod
    def getConnectionString():
        if PropertyUtil.connection_properties is None:
            host='localhost'
            database='cars'
            user='root'
            password='Muskan20'

PropertyUtil.connection_properties={'host':host,'database':database,'user':
user,'password':password}
        return  PropertyUtil.connection_properties
```

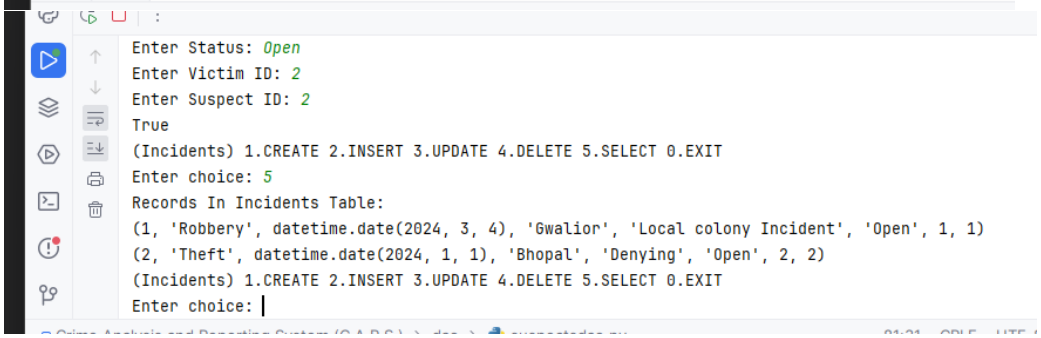
Output Snippets

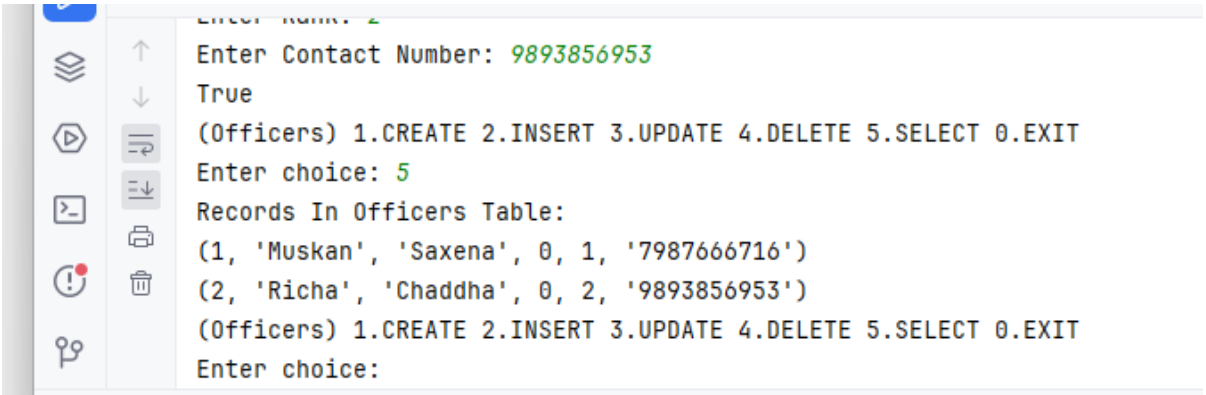

Welcome to Crime Analysis and Reporting System!
1.Incidents
2.Victims
3.Suspects
4.LaeInforcementAgencies
5.Officers
6.Evidence
7.Reports
0.EXIT
Enter choice: 2
(Victims) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 2
Enter Victim ID: 1
Enter First Name: Tanmay
Enter Last Name: Shrivastava
Enter Date Of Birth: 2000-06-26
Enter Gender: Male
Enter Contact Number: 6263582103
True
0.EXIT
Enter choice: 2
(Victims) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 3
Input Victim ID to be Updated: 1
Enter First Name: Tanmay
Enter Last Name: Shrivastava
Enter Date Of Birth: 2001-06-26
Enter Gender: Male
Enter Contact Number: 6263582103
True
Enter Date Of Birth: 2010-12-12
Enter Gender: Female
Enter Contact Number: 1234567890
True
(Victims) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 5
Records In Victims Table:
(1, 'Tanmay', 'Shrivastava', datetime.date(2001, 6, 26), 'Male', '6263582103')
(2, 'Mujrim', 'Doshi', datetime.date(2010, 12, 12), 'Female', '1234567890')
(Victims) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice:

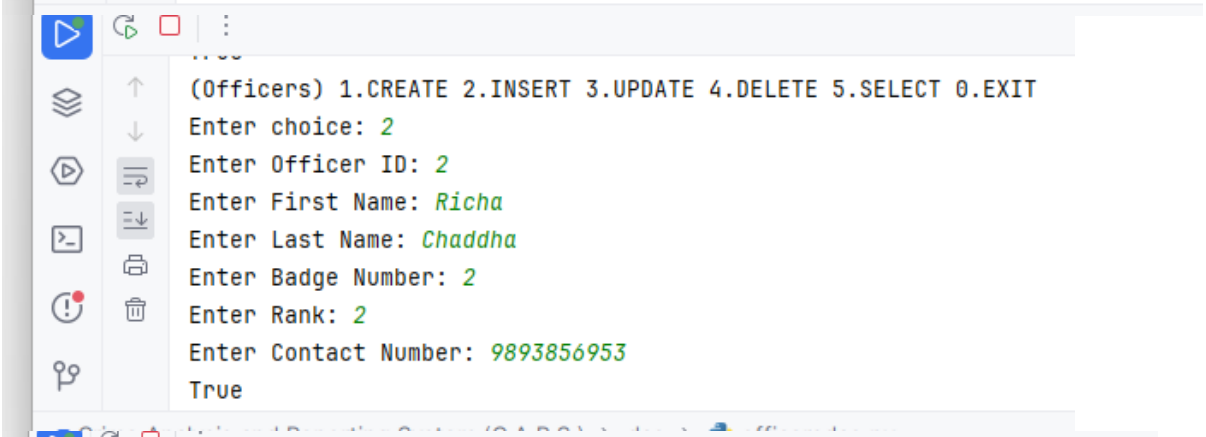

Enter Date Of Birth: 2005-04-04
Enter Gender: Female
Enter Contact Number: 897654321
True
(Suspects) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 5
Records In Suspects Table:
(1, 'Raju', 'Raman', datetime.date(2001, 6, 26), 'Male', '12343256277')
(2, 'Reena', 'Malik', datetime.date(2005, 4, 4), 'Female', '897654321')
(Suspects) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice:

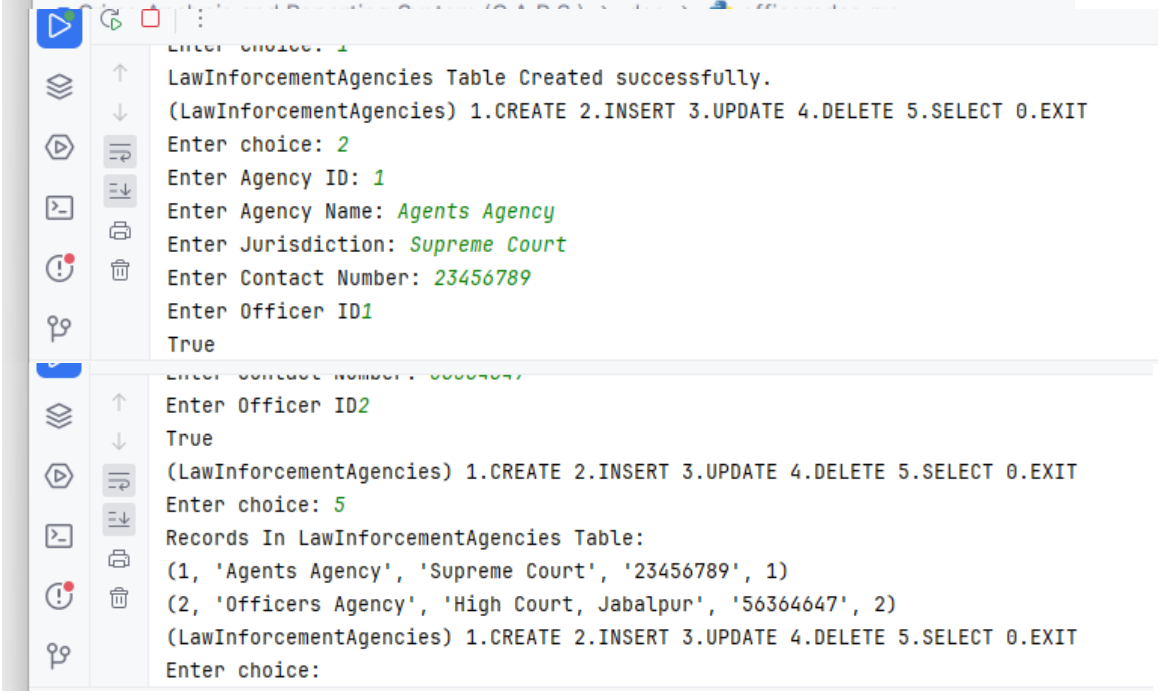

Incidents Table Created successfully.
(Incidents) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 2
Enter Incident ID: 1
Enter Incident Type: Robbery
Enter Incident Date: 2024-03-04
Enter Location: Gwalior
Enter Description: Local colony Incident
Enter Status: Open
Enter Victim ID: 1
Enter Suspect ID: 1

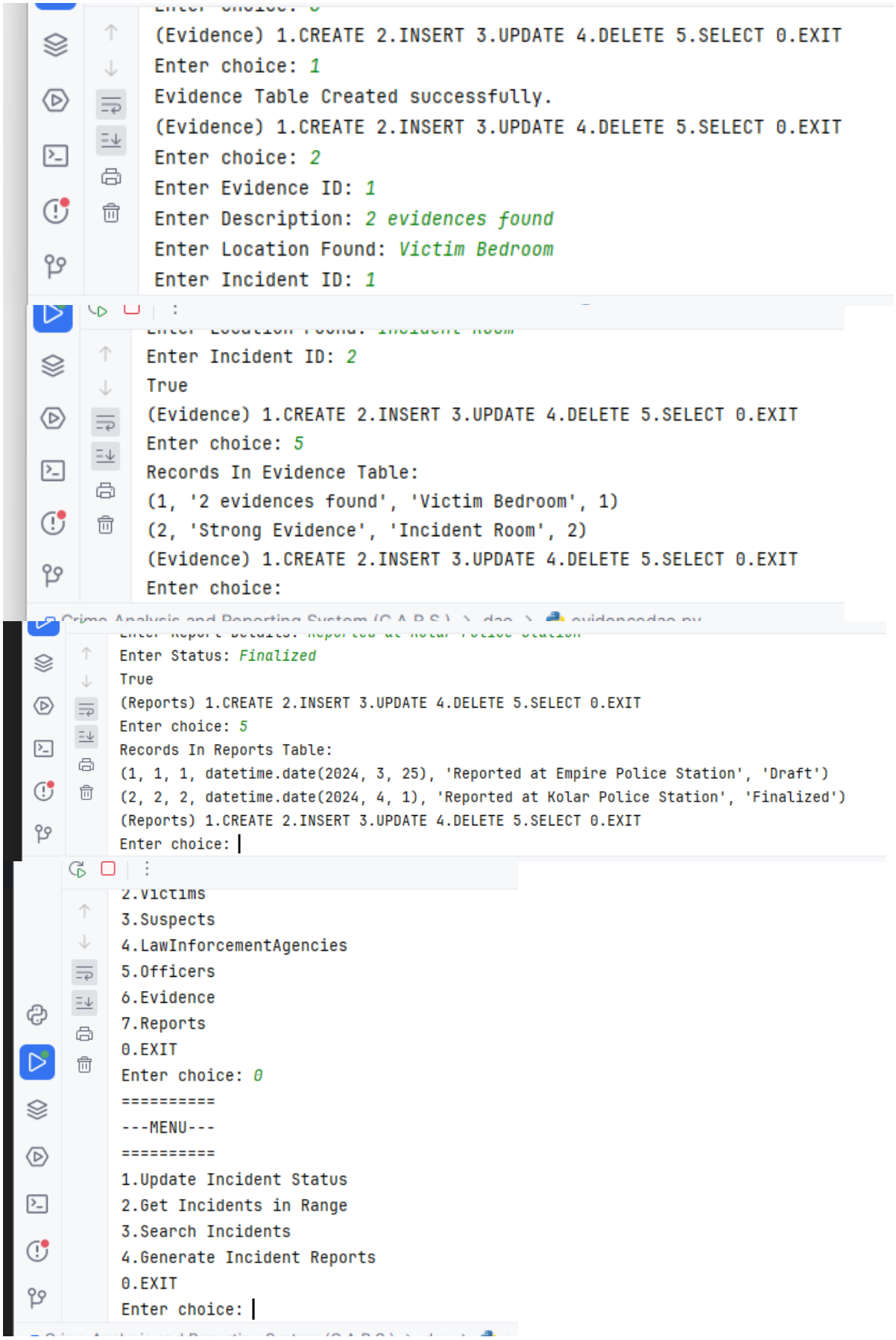

(Incidents) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 2
Enter Incident ID: 2
Enter Incident Type: Theft
Enter Incident Date: 2024-01-01
Enter Location: Bhopal
Enter Description: Denying
Enter Status: Open
Enter Victim ID: 2
Enter Suspect ID: 2
True

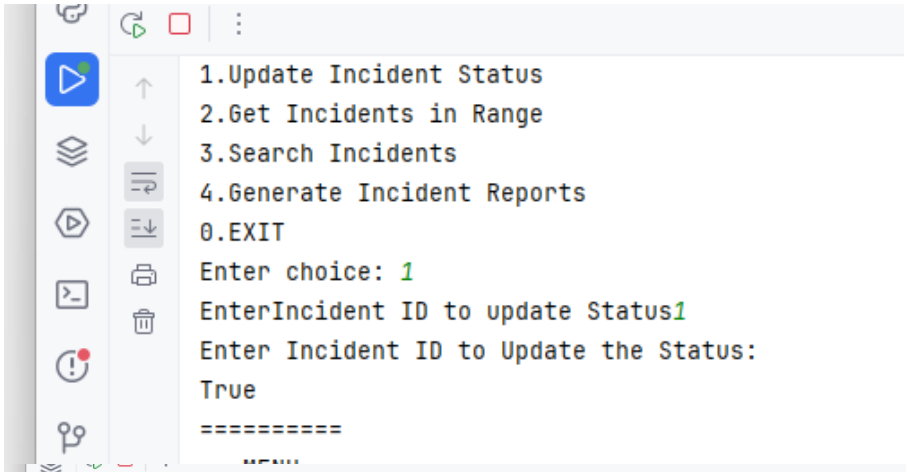

Enter Status: Open
Enter Victim ID: 2
Enter Suspect ID: 2
True
(Incidents) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 5
Records In Incidents Table:
(1, 'Robbery', datetime.date(2024, 3, 4), 'Gwalior', 'Local colony Incident', 'Open', 1, 1)
(2, 'Theft', datetime.date(2024, 1, 1), 'Bhopal', 'Denying', 'Open', 2, 2)
(Incidents) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice:

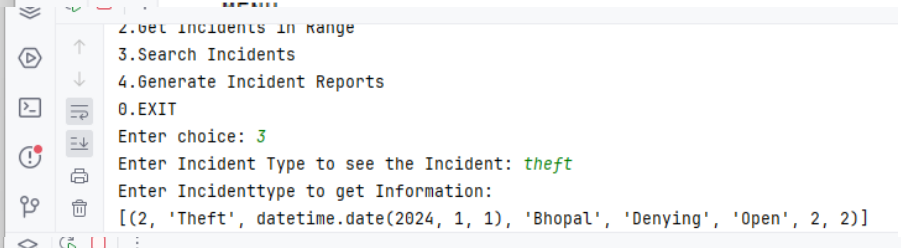

Enter Contact Number: 9893856953
True
(Officers) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 5
Records In Officers Table:
(1, 'Muskan', 'Saxena', 0, 1, '7987666716')
(2, 'Richa', 'Chaddha', 0, 2, '9893856953')
(Officers) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice:

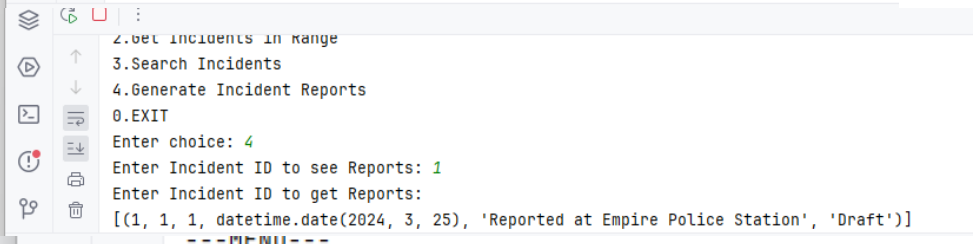

(Officers) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 2
Enter Officer ID: 2
Enter First Name: Richa
Enter Last Name: Chaddha
Enter Badge Number: 2
Enter Rank: 2
Enter Contact Number: 9893856953
True

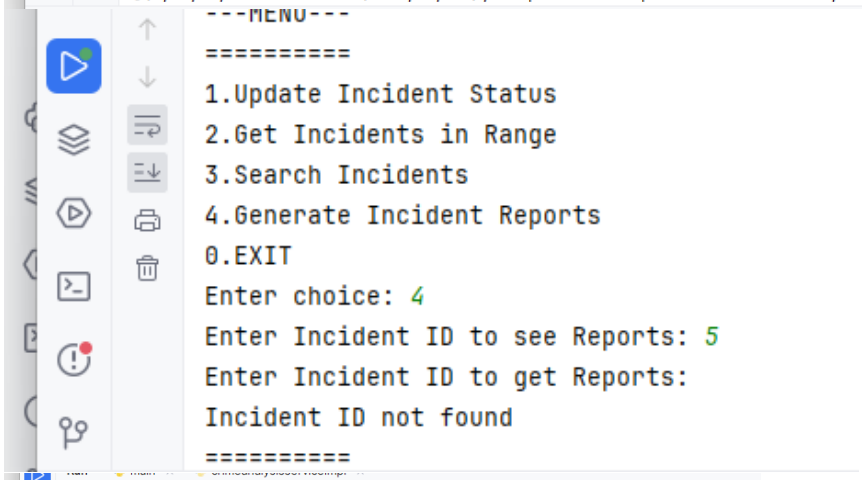

LawInforcementAgencies Table Created successfully.
(LawInforcementAgencies) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 2
Enter Agency ID: 1
Enter Agency Name: Agents Agency
Enter Jurisdiction: Supreme Court
Enter Contact Number: 23456789
Enter Officer ID1
True
Enter Officer ID2
True
(LawInforcementAgencies) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 5
Records In LawInforcementAgencies Table:
(1, 'Agents Agency', 'Supreme Court', '23456789', 1)
(2, 'Officers Agency', 'High Court, Jabalpur', '56364647', 2)
(LawInforcementAgencies) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice:

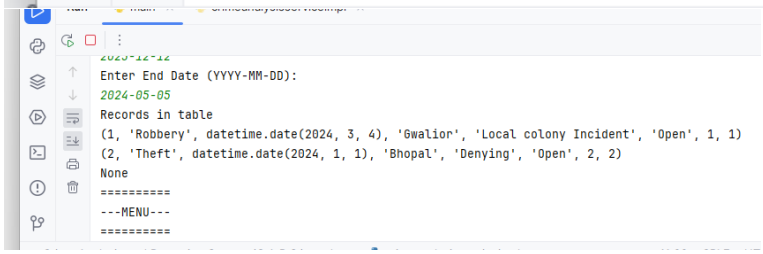

(Evidence) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 1
Evidence Table Created successfully.
(Evidence) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 2
Enter Evidence ID: 1
Enter Description: 2 evidences found
Enter Location Found: Victim Bedroom
Enter Incident ID: 1
Enter Incident ID: 2
True
(Evidence) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 5
Records In Evidence Table:
(1, '2 evidences found', 'Victim Bedroom', 1)
(2, 'Strong Evidence', 'Incident Room', 2)
(Evidence) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice:
Enter Status: Finalized
True
(Reports) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 5
Records In Reports Table:
(1, 1, 1, datetime.date(2024, 3, 25), 'Reported at Empire Police Station', 'Draft')
(2, 2, 2, datetime.date(2024, 4, 1), 'Reported at Kolar Police Station', 'Finalized')
(Reports) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice:
2.Victims
3.Suspects
4.LawInforcementAgencies
5.Officers
6.Evidence
7.Reports
0.EXIT
Enter choice: 0
==========
---MENU---
==========
1.Update Incident Status
2.Get Incidents in Range
3.Search Incidents
4.Generate Incident Reports
0.EXIT
Enter choice:


1.Update Incident Status
2.Get Incidents in Range
3.Search Incidents
4.Generate Incident Reports
0.EXIT
Enter choice: 1
EnterIncident ID to update Status1
Enter Incident ID to Update the Status:
True
==========


3.Search Incidents
4.Generate Incident Reports
0.EXIT
Enter choice: 3
Enter Incident Type to see the Incident: theft
Enter Incidenttype to get Information:
[(2, 'Theft', datetime.date(2024, 1, 1), 'Bhopal', 'Denying', 'Open', 2, 2)]


3.Search Incidents
4.Generate Incident Reports
0.EXIT
Enter choice: 4
Enter Incident ID to see Reports: 1
Enter Incident ID to get Reports:
[(1, 1, 1, datetime.date(2024, 3, 25), 'Reported at Empire Police Station', 'Draft')]


---MENU---
==========
1.Update Incident Status
2.Get Incidents in Range
3.Search Incidents
4.Generate Incident Reports
0.EXIT
Enter choice: 4
Enter Incident ID to see Reports: 5
Enter Incident ID to get Reports:
Incident ID not found
==========


Enter End Date (YYYY-MM-DD):
2024-05-05
Records in table
(1, 'Robbery', datetime.date(2024, 3, 4), 'Gwalior', 'Local colony Incident', 'Open', 1, 1)
(2, 'Theft', datetime.date(2024, 1, 1), 'Bhopal', 'Denying', 'Open', 2, 2)
None
==========
---MENU---
==========