Case Study: Crime Analysis and Reporting System (C.A.R.S.)

SHIVAM SINGH

Instructions

- Project submissions should be done through the partcipants' Github repository and the link should be shared with trainers and Hexavarsity.
- Each section builds upon the previous one, and by the end, you will have a comprehensive Crime Analysis and reporting system implemented with a strong focus on SQL, control flow statements, loops, arrays, collections, exception handling, database interaction and Unit Testing.
- Follow object-oriented principles throughout the project. Use classes and objects to model real-world entities, encapsulate data and behavior, and ensure code reusability.
- Throw user defined exceptions from corresponding methods and handled.
- The following Directory structure is to be followed in the application.
    - **entity**
        - Create entity classes in this package. All entity class should not have any business logic.
    - **dao**
        - Create Service Provider interface to showcase functionalities.
        - Create the implementation class for the above interface with db interaction.
    - **exception**
        - Create user defined exceptions in this package and handle exceptions whenever needed.
    - **util**
        - Create a DBPropertyUtil class with a static function which takes property file name as parameter and returns connection string.
        - Create a DBConnUtil class which holds static method which takes connection string as parameter file and returns connection object(Use method defined in DBPropertyUtil class to get the connection String).
    - **main**
        - Create a class MainModule and demonstrate the functionalities in a menu driven application.

Key Functionalities:

The primary objective of this project is to develop a comprehensive **Crime Analysis and Reporting System (CARS)** that addresses the above-mentioned challenges and provides law enforcement agencies with a robust, user-friendly, and secure platform for crime data management and reporting.

1. **Schema design:**

Entities:

1. **Incidents:**

    - IncidentID (Primary Key)

- IncidentType (e.g., Robbery, Homicide, Theft)
- IncidentDate
- Location (Geospatial Data: Latitude and Longitude)
- Description
- Status (e.g., Open, Closed, Under Investigation)
- VictimID (Foreign Key, linking to Victims)
- SuspectId(Foreign Key, Linking to Suspect)

2. **Victims:**
    - VictimID (Primary Key)
    - FirstName
    - LastName
    - DateOfBirth
    - Gender
    - Contact Information (e.g., Address, Phone Number)
3. **Suspects:**
    - SuspectID (Primary Key)
    - FirstName
    - LastName
    - DateOfBirth
    - Gender
    - Contact Information

4. **Law Enforcement Agencies:**
    - AgencyID (Primary Key)
    - AgencyName
    - Jurisdiction
    - Contact Information
    - Officer(s) (Link to Officers within the agency)
5. **Officers:**

- OfficerID (Primary Key)
- FirstName
- LastName
- BadgeNumber
- Rank
- Contact Information
- AgencyID (Foreign Key, linking to Law Enforcement Agencies)

6. **Evidence:**
    - EvidenceID (Primary Key)
    - Description
    - Location Found
    - IncidentID (Foreign Key, linking to Incidents)
7. **Reports:**
    - ReportID (Primary Key)
    - IncidentID (Foreign Key, linking to Incidents)
    - ReportingOfficer (Foreign Key, linking to Officers)
    - ReportDate
    - ReportDetails
    - Status (e.g., Draft, Finalized)

Relationships:

- An Incident can have multiple Victims and Suspects.
- An Incident is associated with one Law Enforcement Agency.
- An Officer works for a single Law Enforcement Agency.
- Evidence can be linked to an Incident.
- Reports are generated for Incidents by ReportingOfficers.

Coding

Create the model/entity classes corresponding to the schema within package entity with variables declared private, constructors(default and parametrized) and getters,setters)

Service Provider Interface/Abstract class

- Keep the interfaces and implementation classes in package dao

Create ICrimeAnalysisService Interface/abstract classs with the following methods

// **Create a new incident**
createIncident();
parameters- **Incident** object
return type Boolean

// **Update the status of an incident**
updateIncidentStatus();
parameters- **Status** object,incidentid
return type Boolean
// **Get a list of incidents within a date range**
getIncidentsInDateRange();
parameters- startDate, endDate
return type Collection of Incident objects

// **Search for incidents based on various criteria**
searchIncidents(IncidentType criteria);
parameters- **IncidentType object**
return type Collection of Incident objects
// **Generate incident reports**
generateIncidentReport();
parameters- **Incident object**
return type Report object
// **Create a new case and associate it with incidents**
createCase();
parameters- caseDescription string, collection of Incident Objects
return type Case object
// **Get details of a specific case**
Case getCaseDetails(int caseId);
parameters- caseDescription string, collection of Incident Objects
return type Case object
// Update case details
updateCaseDetails();
parameters- Case object
return type boolean
// **Get a list of all cases**
List<Case> getAllCases();
parameters- None
return type Collection of cases

7: Connect your application to the SQL database:

1. Write code to establish a connection to your SQL database.

Create a utility class **DBConnection** in a package **util** with a static variable **connection** of Type **Connection** and a static method **getConnection()** which returns connection.

Connection properties supplied in the connection string should be read from a property file.

Create a utility class **PropertyUtil** which contains a static method named **getPropertyString()** which reads a property fie containing connection details like hostname, dbname, username, password, port number and returns a connection string.

7: Service implementation

1. Create a Service class **CrimeAnalysisServiceImpl** in **package dao** with a static variable named connection of type **Connection** which can be assigned in the constructor by invoking the **getConnection()** method in **DBConnection** class

2. Provide implementation for all the methods in the interface/abstract clsass

8: Exception Handling

Create the exceptions in package **c.myexceptions**

Define the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method,

1. **IncidentNumberNotFoundException** :throw this exception when user enters an invalid patient number which doesn't exist in db

9. Main Method

Create class named MainModule with main method in main package.

Trigger all the methods in service implementation class

10. Unit Testing

Creating JUnit test cases for a **Crime Analysis and Reporting System** is essential to ensure the correctness and reliability of your system. Below are some example questions to guide the creation of JUnit test cases for various components of the system:

1. **Incident Creation:**
    - Does the createIncident method correctly create an incident with the provided attributes?
    - Are the attributes of the created incident accurate?
2. **Incident Status Update:**
    - Does the updateIncidentStatus method effectively update the status of an incident?
    - Does it handle invalid status updates appropriately?

Case.py

```
from Case_Study.UTIL.DB_CONNECTION import DBConnection


class Cases(DBConnection):
    def __init__(self, case_id=None, description=None, case_date=None,
status=None):
        self.case_id = case_id
        self.description = description
        self.case_date = case_date
        self.status = status

    def create_table(self):
        create_query = '''
            create table if not exists Cases(
            case_id int primary key,
            description varchar(150),
            case_date date,
            status varchar(30)
            )'''
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(create_query)
        print("Case table created successfully")
```

Criminal_Analysis.py

```
from Case_Study.UTIL.DB_CONNECTION import DBConnection
from Case_Study.ENTITY.ICRIMEANALYSIS_SERVICE import I_crime_analysis_service
from Case_Study.DAO.INCIDENTS import Incidents
from Case_Study.DAO.REPORTS import Reports
from Case_Study.DAO.CASE import Cases


class crime_analysis_service_impl(Incidents, Reports, Cases, DBConnection,
I_crime_analysis_service):
    def __init__(self):
        super(Incidents, self).__init__()

    def createIncident(self):
        incident = Incidents()
        incident.insert_into()

    def updateIncidentStatus(self):
        incident = Incidents()
        incident.update_table()

    def getIncidentsInDateRange(self):
        start_date = input("Enter the start date(yyyy-mm-dd): ")
        end_date = input("Enter the input date(yyyy-mm-dd): ")
        res = [incident for incident in Incidents.incidents if start_date <=
incident.incident_date <= end_date]
        for i in res:
            print(i)

    def searchIncidents(self):
        self.incident_id = int(input('Enter the incident id to search the
incident details: '))
        search_query = f'select * from Incidents where incident_id =
```

```
{self.incident_id}'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(search_query)
        data = stmt.fetchall()
        for i in data:
            print(i)
        print("Search successfully")

    def generateIncidentReport(self):
        self.incident_id = int(input("Enter the incident id to generate a
report: "))
        report_query = f'select * from Reports where incident_id =
{self.incident_id}'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(report_query)
        data = stmt.fetchall()
        for i in data:
            print(i)
        print("Reports generated successfully")

    def createCase(self):
        self.case_id = int(input("Enter the case id: "))
        self.description = input("Enter the description: ")
        self.case_date = input("Enter the case date: ")
        self.status = input("Enter the status: ")

        query = 'insert into Cases(case_id, description, case_date, status)
values(%s,%s,%s,%s)'
        data = [(self.case_id, self.description, self.case_date, self.status)]
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.executemany(query, data)
        DBConnection.connection.commit()
        print("Created case successfully")

    def getCaseDetails(self):
        self.case_id = int(input("Enter the case Id to get details: "))
        get_query = f'select * from Cases where case_id={self.case_id}'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(get_query)
        data = stmt.fetchall()
        for i in data:
            print(i)
        print("Case details displayed successfully")

    def updateCaseDetails(self):
        self.case_id = int(input("Enter the case Id to update details: "))
        self.description = input("Enter the description: ")
        self.case_date = input("Enter the case date: ")
        self.status = input("Enter the status: ")
        update_query = f'update Cases set description=%s, case_date=%s,
status=%s where case_id=%s'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(update_query)
        print("Case updated successfully")

    def getAllCases(self):
```

```
        get_query = f'select * from Cases'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(get_query)
        data = stmt.fetchall()
        for i in data:
            print(i)



# obj = crime_analysis_service_impl()
# obj.generateIncidentReport()
```

Evidence.py

```
from Case_Study.UTIL.DB_CONNECTION import DBConnection


class Evidence(DBConnection):
    def __init__(self, evidence_id=None, description=None, location=None,
incident_id=None):
        self.evidence_id = evidence_id
        self.description = description
        self.location = location
        self.incident_id = incident_id

    def create_table(self):
        create_query = '''
        create table if not exists Evidence(
        evidence_id int primary key,
        description varchar(150),
        location varchar(50),
        incident_id int
        )'''
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(create_query)
        print("Evidence table created successfully")

    def insert_into(self):
        self.evidence_id = int(input("Enter the evidence id: "))
        self.description = input("Enter the description: ")
        self.location = input("Enter the location: ")
        self.incident_id = input("Enter the incident id: ")

        insert_query = 'insert into Evidence(evidence_id, description,
location, incident_id) values(%s,%s,%s,%s)'
        data = [(self.evidence_id, self.description, self.location,
self.incident_id)]
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.executemany(insert_query, data)
        DBConnection.connection.commit()
        print("Values inserted successfully")

    def update_table(self):
        self.evidence_id = int(input("Enter the evidence id: "))
        self.description = input("Enter the description: ")
        self.location = input("Enter the location: ")
        self.incident_id = input("Enter the incident id: ")
```

```
        update_query = 'update Evidence set description=%s, location=%s,
incident_id=%s where evidence_id=%s'
        data = [(self.description, self.location, self.incident_id,
self.evidence_id)]
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(update_query, data)
        DBConnection.connection.commit()
        print("Values updated successfully")

    def delete_table(self):
        self.evidence_id = int(input("Enter the evidence id to delete values:
"))
        delete_query = f'delete from Evidence where
evidence_id={self.evidence_id}'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(delete_query)
        DBConnection.connection.commit()
        print("Values deleted successfully")

    def select_table(self):
        select_query = 'select * from Evidence'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(select_query)
        data = stmt.fetchall()
        for i in data:
            print(i)
        print("Values displayed successfully")
```

Incidents.py

```
from Case_Study.UTIL.DB_CONNECTION import DBConnection
from Case_Study.EXCEPTION.INCIDENTNUMBERNOTFOUND import
IncidentNumberNotFoundException

class Incidents(DBConnection):
    incidents = []

    def __init__(self, incident_id=None, incident_type=None,
incident_date=None, location=None, description=None, status=None,
victim_id=None, suspect_id=None):
        self.incident_id = incident_id
        self.incident_type = incident_type
        self.incident_date = incident_date
        self.location = location
        self.description = description
        self.status = status
        self.victim_id = victim_id
        self.suspect_id = suspect_id

    def create_table(self):
        create_query = '''
            create table if not exists Incidents(
            incident_id int primary key,
            incident_type varchar(30),
            incident_date date,
            location varchar(30),
```

```
            description varchar(100),
            status varchar(30),
            victim_id int,
            suspect_id int
            )
        '''
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(create_query)
        print("Incidents table created successfully")

    def insert_into(self):
        self.incident_id = int(input("Enter the incident id: "))
        self.incident_type = input("Enter the incident type: ")
        self.incident_date = input("Enter the incident date: ")
        self.location = input("Enter the location: ")
        self.description = input("Enter the description: ")
        self.status = input("Enter the status: ")
        self.victim_id = int(input("Enter the victim id: "))
        self.suspect_id = int(input("Enter the suspect id: "))

        insert_query = 'insert into Incidents(incident_id, incident_type,
incident_date, location, description, status, victim_id, suspect_id)
values(%s,%s,%s,%s,%s,%s,%s,%s)'
        data = [(self.incident_id, self.incident_type, self.incident_date,
self.location, self.description, self.status, self.victim_id, self.suspect_id)]
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.executemany(insert_query, data)
        DBConnection.connection.commit()
        print("Data inserted successfully")
        return 'Incident created successfully'

    def update_table(self):
        try:
            self.incident_id = int(input("Enter the incident id to update the
values: "))
            self.incident_type = input("Enter the incident type: ")
            self.incident_date = input("Enter the incident date: ")
            self.location = input("Enter the location: ")
            self.description = input("Enter the description: ")
            self.status = input("Enter the status: ")
            self.victim_id = int(input("Enter the victim id: "))
            self.suspect_id = int(input("Enter the suspect id: "))
            if not self.incident_exists(self.incident_id):
                raise IncidentNumberNotFoundException("Incident id not found")

            update_query = 'update Incidents set incident_type=%s,
incident_date=%s, location=%s, description=%s, status=%s, victim_id=%s,
suspect_id=%s where incident_id=%s'
            data = [(self.incident_type, self.incident_date, self.location,
self.description, self.status, self.victim_id, self.suspect_id,
self.incident_id)]
            DBConnection.getConnection()
            stmt = DBConnection.connection.cursor()
            stmt.executemany(update_query, data)
            DBConnection.connection.commit()
            print("Updated successfully")
            return 'Values updated successfully'

        except IncidentNumberNotFoundException as e:
```

```
            print(e)
        except Exception as e:
            print(e)

    def delete_table(self):
        try:
            self.incident_id = int(input("Enter the incident id to delete data:
"))
            if not self.incident_exists(self.incident_id):
                raise IncidentNumberNotFoundException("Incident id not found")
            delete_query = f'delete from Incidents where incident_id =
{self.incident_id}'
            DBConnection.getConnection()
            stmt = DBConnection.connection.cursor()
            stmt.execute(delete_query)
            DBConnection.connection.commit()
            print("Deleted successfully")

        except IncidentNumberNotFoundException as e:
            print(e)
        except Exception as e:
            print(e)

    def select_table(self):
        select_query = 'select * from Incidents'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(select_query)
        data = stmt.fetchall()
        for i in data:
            Incidents.incidents.append(i)
            print(i)
        Incidents.incidents = [list(i) for i in Incidents.incidents]
        print("Values displayed successfully")

    def incident_exists(self, incident_id):
        try:
            DBConnection.getConnection()
            stmt = DBConnection.connection.cursor()
            select_query = f'SELECT COUNT(*) FROM Incidents WHERE incident_id =
{incident_id}'
            stmt.execute(select_query)
            result = stmt.fetchone()
            if result and result[0] > 0:
                return True
            else:
                return False

        except Exception as e:
            print(f"Error checking incident existence: {e}")
            return False

    def __str__(self):
        return """f'Incident ID: {self.incident_id}', f'Incident Type:
{self.incident_type}',f'Incident date: {self.incident_date}',
         f'Location: {self.location}', f'Description: {self.description}',
f'Status: {self.status}',
          f'Victim ID: {self.victim_id}', f'Suspect ID: {self.suspect_id}' """
```

Law_enforcement_agency.py

```
from Case_Study.UTIL.DB_CONNECTION import DBConnection


class Law_Enforcement_Agencies(DBConnection):
    def __init__(self, agency_id=None, agency_name=None, jurisdiction=None,
phone_num=None, officer=None):
        self.agency_id = agency_id
        self.agency_name = agency_name
        self.jurisdiction = jurisdiction
        self.phone_num = phone_num
        self.officer = officer

    def create_table(self):
        create_query = '''
            create table Law_Enforcement_Agencies(
            agency_id int primary key,
            agency_name varchar(30),
            jurisdiction varchar(50),
            phone_num varchar(20),
            officer varchar(30)
        )'''
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(create_query)
        print("Law Enforcement Agencies table created successfully")

    def insert_into(self):
        self.agency_id = int(input("Enter the agency id: "))
        self.agency_name = input("Enter the agency name: ")
        self.jurisdiction = input("Enter the jurisdiction: ")
        self.phone_num = input("Enter the phone number: ")
        self.officer = input("Enter the officer: ")

        insert_query = 'insert into Law_Enforcement_Agencies(agency_id,
agency_name, jurisdiction, phone_num, officer) values(%s,%s,%s,%s,%s)'
        data = [(self.agency_id, self.agency_name, self.jurisdiction,
self.phone_num, self.officer)]
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.executemany(insert_query, data)
        DBConnection.connection.commit()
        print("Values inserted successfully")

    def update_table(self):
        self.agency_id = int(input("Enter the agency id to update the values:
"))
        self.agency_name = input("Enter the agency name: ")
        self.jurisdiction = input("Enter the jurisdiction: ")
        self.phone_num = input("Enter the phone number: ")
        self.officer = input("Enter the officer: ")

        update_query = 'update Law_Enforcements_Agencies set agency_name=%s,
jurisdiction=%s, phone_num=%s, officer=%s where agency_id=%s'
        data = [(self.agency_name, self.jurisdiction, self.phone_num,
self.officer, self.agency_id)]
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(update_query, data)
        DBConnection.connection.commit()
```

```
        print("Values updated successfully")

    def delete_table(self):
        self.agency_id = int(input("Enter the agency id to delete values: "))
        delete_query = f'delete from Law_Enforcement_Agencies where
agency_id={self.agency_id}'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(delete_query)
        DBConnection.connection.commit()
        print("Values deleted successfully")

    def select_table(self):
        select_query = 'select * from Law_Enforcement_Agencies'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(select_query)
        data = stmt.fetchall()
        for i in data:
            print(i)
        print("Values displayed successfully")
```

Officers.py

```
from Case_Study.UTIL.DB_CONNECTION import DBConnection


class Officers(DBConnection):
    def __init__(self, officer_id=None, first_name=None, last_name=None,
badge_no=None, officer_rank=None, phone_num=None, agency_id=None):
        self.officer_id = officer_id
        self.first_name = first_name
        self.last_name = last_name
        self.badge_no = badge_no
        self.officer_rank = officer_rank
        self.phone_num = phone_num
        self.agency_id = agency_id

    def create_table(self):
        create_query = ''' create table if not exists Officers(
            officer_id int primary key,
            first_name varchar(30),
            last_name varchar(30),
            badge_no varchar(10),
            officer_rank varchar(30),
            phone_num varchar(20),
            agency_id int
        )'''
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(create_query)
        print("Officers table created successfully")

    def insert_into(self):
        self.officer_id = int(input("Enter the officer id: "))
        self.first_name = input("Enter the first name: ")
        self.last_name = input("Enter the last name: ")
        self.badge_no = input("Enter the badge number: ")
```

```
        self.officer_rank = input("Enter the rank: ")
        self.phone_num = input("Enter the phone number: ")
        self.agency_id = input("Enter the agency id: ")

        insert_query = 'insert into Officers(officer_id, first_name, last_name,
badge_no, officer_rank, phone_num, agency_id) values(%s,%s,%s,%s,%s,%s,%s)'
        data = [(self.officer_id, self.first_name, self.last_name,
self.badge_no, self.officer_rank, self.phone_num, self.agency_id)]
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.executemany(insert_query, data)
        DBConnection.connection.commit()
        print("Values inserted successfully")

    def update_table(self):
        self.officer_id = int(input("Enter the officer id to update values: "))
        self.first_name = input("Enter the first name: ")
        self.last_name = input("Enter the last name: ")
        self.badge_no = input("Enter the badge number: ")
        self.officer_rank = input("Enter the rank: ")
        self.phone_num = input("Enter the phone number: ")
        self.agency_id = input("Enter the agency id: ")

        update_query = 'update Officers set first_name=%s, last_name=%s,
badge_no=%s, officer_rank=%s, phone_num=%s, agency_id=%s where officer_id=%s'
        data = [(self.first_name, self.last_name, self.badge_no,
self.officer_rank, self.phone_num, self.agency_id, self.officer_id)]
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(update_query, data)
        DBConnection.connection.commit()
        print("Values updated successfully")

    def delete_table(self):
        self.officer_id = int(input("Enter the officer id to delete values: "))
        delete_query = f'delete from Officers where
officer_id={self.officer_id}'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(delete_query)
        DBConnection.connection.commit()
        print("Values deleted successfully")

    def select_table(self):
        select_query = 'select * from Officers'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(select_query)
        data = stmt.fetchall()
        for i in data:
            print(i)
        print("Values displayed successfully")
```

Reports.py

```
from Case_Study.UTIL.DB_CONNECTION import DBConnection


class Reports(DBConnection):
    def __init__(self, report_id=None, incident_id=None,
reporting_officer=None, report_date=None, report_details=None, status=None):
        self.report_id = report_id
        self.incident_id = incident_id
        self.reporting_officer = reporting_officer
        self.report_date = report_date
        self.report_details = report_details
        self.status = status

    def create_table(self):
        create_query = '''
        create table if not exists Reports(
        report_id int primary key,
        incident_id int,
        reporting_officer varchar(30),
        report_date date,
        report_details varchar(150),
        status varchar(20)
        )'''
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(create_query)
        print("Reports table successfully created")

    def insert_into(self):
        self.report_id = int(input("Enter the report id: "))
        self.incident_id = input("Enter the incident id: ")
        self.reporting_officer = input("Enter the reporting officer: ")
        self.report_date = input("Enter the report date: ")
        self.report_details = input("Enter the report details: ")
        self.status = input("Enter the status: ")

        insert_query = 'insert into Reports(report_id, incident_id,
reporting_officer, report_date, report_details, status)
values(%s,%s,%s,%s,%s,%s)'
        data = [(self.report_id, self.incident_id, self.reporting_officer,
self.report_date, self.report_details, self.status)]
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.executemany(insert_query, data)
        DBConnection.connection.commit()
        print("Values inserted successfully")

    def update_table(self):
        self.report_id = int(input("Enter the report id: "))
        self.incident_id = input("Enter the incident id: ")
        self.reporting_officer = input("Enter the reporting officer: ")
        self.report_date = input("Enter the report date: ")
        self.report_details = input("Enter the report details: ")
        self.status = input("Enter the status: ")

        update_query = 'update Reports set incident_id=%s,
reporting_officer=%s, report_date=%s, report_details=%s, status=%s where
report_id=%s'
        data = [(self.incident_id, self.reporting_officer, self.report_date,
```

```
self.report_details, self.status, self.report_id)]
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(update_query, data)
        DBConnection.connection.commit()
        print("Values updated successfully")

    def delete_table(self):
        self.report_id = int(input("Enter the report id to delete values: "))
        delete_query = f'delete from Reports where report_id={self.report_id}'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(delete_query)
        DBConnection.connection.commit()
        print("Values deleted successfully")

    def select_table(self):
        select_query = 'select * from Reports'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(select_query)
        data = stmt.fetchall()
        for i in data:
            print(i)
        print("Values displayed successfully")
```

Suspects.py

```
from Case_Study.UTIL.DB_CONNECTION import DBConnection


class Suspects(DBConnection):
    def __init__(self, suspect_id=None, first_name=None, last_name=None,
dob=None, gender=None, address=None, phone_num=None):
        self.suspect_id = suspect_id
        self.first_name = first_name
        self.last_name = last_name
        self.dob = dob
        self.gender = gender
        self.address = address
        self.phone_num = phone_num

    def create_table(self):
        create_query = '''
            create table if not exists Suspects(
            suspect_id int primary key,
            first_name varchar(30),
            last_name varchar(30),
            dob date,
            gender char,
            address varchar(30),
            phone_num varchar(20))
            '''
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(create_query)
        print("Suspects table created successfully")

    def insert_into(self):
        self.suspect_id = int(input("Enter the suspect id: "))
```

```
        self.first_name = input("Enter the first name: ")
        self.last_name = input("Enter the last name: ")
        self.dob = input("Enter the date of birth: ")
        self.gender = input("Enter the gender: ")
        self.address = input("Enter the address: ")
        self.phone_num = input("Enter the phone number: ")

        insert_query = 'insert into Suspects(suspect_id, first_name, last_name,
dob, gender, address, phone_num) values(%s,%s,%s,%s,%s,%s,%s)'
        data = [(self.suspect_id, self.first_name, self.last_name, self.dob,
self.gender, self.address, self.phone_num)]
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.executemany(insert_query, data)
        DBConnection.connection.commit()
        print("Values inserted successfully")

    def update_table(self):
        self.suspect_id = int(input("Enter the suspect id to update the values:
"))
        self.first_name = input("Enter the first name: ")
        self.last_name = input("Enter the last name: ")
        self.dob = input("Enter the date of birth: ")
        self.gender = input("Enter the gender: ")
        self.address = input("Enter the address: ")
        self.phone_num = input("Enter the phone number: ")

        update_query = 'update Suspects set first_name=%s, last_name=%s,
dob=%s, gender=%s, address=%s, phone_num=%s where suspect_id=%s'
        data = [(self.first_name, self.last_name, self.dob, self.gender,
self.address, self.phone_num, self.suspect_id)]
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(update_query, data)
        DBConnection.connection.commit()
        print("Values updated successfully")

    def delete_table(self):
        self.suspect_id = int(input("Enter the suspect id to delete values: "))
        delete_query = f'delete from Suspects where
suspect_id={self.suspect_id}'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(delete_query)
        DBConnection.connection.commit()
        print("Values deleted successfully")

    def select_table(self):
        select_query = 'select * from Suspects'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(select_query)
        data = stmt.fetchall()
        for i in data:
            print(i)
        print("Values displayed successfully")
```

Victims.py

```
from Case_Study.UTIL.DB_CONNECTION import DBConnection


class Victims(DBConnection):
    def __init__(self, victim_id=None, first_name=None, last_name=None,
dob=None, gender=None, address=None, phone_num=None):
        self.victim_id = victim_id
        self.first_name = first_name
        self.last_name = last_name
        self.dob = dob
        self.gender = gender
        self.address = address
        self.phone_num = phone_num

    def create_table(self):
        create_query = '''
            create table if not exists Victims(
            victim_id int primary key,
            first_name varchar(30),
            last_name varchar(30),
            dob date,
            gender char,
            address varchar(30),
            phone_num varchar(20))
            '''
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(create_query)
        print("Victims table created successfully")

    def insert_into(self):
        self.victim_id = int(input("Enter the victim id: "))
        self.first_name = input("Enter the first name: ")
        self.last_name = input("Enter the last name: ")
        self.dob = input("Enter the date of birth: ")
        self.gender = input("Enter the gender: ")
        self.address = input("Enter the address: ")
        self.phone_num = input("Enter the phone number: ")

        insert_query = 'insert into Victims(victim_id, first_name, last_name,
dob, gender, address, phone_num) values(%s,%s,%s,%s,%s,%s,%s)'
        data = [(self.victim_id, self.first_name, self.last_name, self.dob,
self.gender, self.address, self.phone_num)]
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.executemany(insert_query, data)
        DBConnection.connection.commit()
        print("Values inserted successfully")

    def update_table(self):
        self.victim_id = int(input("Enter the victim id to update the values:
"))
        self.first_name = input("Enter the first name: ")
        self.last_name = input("Enter the last name: ")
        self.dob = input("Enter the date of birth: ")
        self.gender = input("Enter the gender: ")
        self.address = input("Enter the address: ")
        self.phone_num = input("Enter the phone number: ")
```

```
        update_query = 'update Victims set first_name=%s, last_name=%s, dob=%s,
gender=%s, address=%s, phone_num=%s where victim_id=%s'
        data = [(self.first_name, self.last_name, self.dob, self.gender,
self.address, self.phone_num, self.victim_id)]
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(update_query, data)
        DBConnection.connection.commit()
        print("Values updated successfully")

    def delete_table(self):
        self.victim_id = int(input("Enter the victim id to delete values: "))
        delete_query = f'delete from Victims where victim_id={self.victim_id}'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(delete_query)
        DBConnection.connection.commit()
        print("Values deleted successfully")

    def select_table(self):
        select_query = 'select * from Victims'
        DBConnection.getConnection()
        stmt = DBConnection.connection.cursor()
        stmt.execute(select_query)
        data = stmt.fetchall()
        for i in data:
            print(i)
        print("Values displayed successfully")
```

DB_Connection.py

```
import mysql.connector as sql
from mysql.connector import Error
from Case_Study.UTIL.PROPERTY_UTIL import propertyUtil

class DBConnection:
    connection = None

    @staticmethod
    def getConnection():
        if DBConnection.connection is None:
            try:
                connection_string = propertyUtil.getPropertyString()
                DBConnection.connection = sql.connect(**connection_string)

                if DBConnection.connection.is_connected():
                    print("Database connected successfully")

            except Error as e:
                print(f'Error : {e}')

        return DBConnection.connection
```

PropertyUtil.py

```
class propertyUtil:

    @staticmethod
    def getPropertyString():
        connection_string = {
            'host': 'localhost',
            'database': 'crime_reporting_system',
            'user': 'root',
            'password': '765795'
        }

        return connection_string
```

main.py

```
from Case_Study.UTIL.DB_CONNECTION import DBConnection
from Case_Study.DAO.INCIDENTS import Incidents
from Case_Study.DAO.VICTIMS import Victims
from Case_Study.DAO.SUSPECTS import Suspects
from Case_Study.DAO.LAW_ENFORCEMENT_AGENCIES import Law_Enforcement_Agencies
from Case_Study.DAO.OFFICERS import Officers
from Case_Study.DAO.EVIDENCE import Evidence
from Case_Study.DAO.REPORTS import Reports
from Case_Study.DAO.CASE import Cases
from Case_Study.DAO.CRIMEANALYSIS_SERVICEIMPL import
crime_analysis_service_impl

try:
    connObj = DBConnection()
    con = connObj.getConnection()

    while True:
        incidentObj = Incidents()
        victimObj = Victims()
        suspectObj = Suspects()
        lawObj = Law_Enforcement_Agencies()
        officerObj = Officers()
        evidenceObj = Evidence()
        reportObj = Reports()
        serviceImplementObj = crime_analysis_service_impl()

        print("Select table to use functionalities")
        print("1.Incidents\n2.Victims\n3.Suspects\n4.Law Enforcement
Agencies\n5.Officers\n6.Evidence\n7.Reports\n8.crime analysis service
impl\n9.exit")
        ch = int(input("enter your choice:"))

        if ch == 1:
            while True:
                print("1.create Incidents\t2.insert Incidents\t3.update
incidents\n4.delete incidents\t5.select incidents\n6.Exit")
                choice = int(input("enter your choice:"))
                if choice == 1:
                    incidentObj.create_table()
                elif choice == 2:
                    incidentObj.insert_into()
                elif choice == 3:
                    incidentObj.update_table()
```

```
                elif choice == 4:
                    incidentObj.delete_table()
                elif choice == 5:
                    incidentObj.select_table()
                elif choice == 6:
                    print("exited successfully")
                    break
                else:
                    print("Wrong choice")

        elif ch == 2:
            while True:
                print("1.create victims\t2.insert victims\t3.update
victims\n4.delete victims\t5.select victims\n6.Exit")
                choice = int(input("enter your choice:"))
                if choice == 1:
                    victimObj.create_table()
                elif choice == 2:
                    victimObj.insert_into()
                elif choice == 3:
                    victimObj.update_table()
                elif choice == 4:
                    victimObj.delete_table()
                elif choice == 5:
                    victimObj.select_table()
                elif choice == 6:
                    print("Exited successfully")
                    break
                else:
                    print("Wrong choice")

        elif ch == 3:
            while True:
                print("1.create suspects\t2.insert suspects\t3.update
suspects\n4.delete suspects\t5.select suspects\n6.Exit")
                choice = int(input("enter your choice:"))
                if choice == 1:
                    suspectObj.create_table()
                elif choice == 2:
                    suspectObj.insert_into()
                elif choice == 3:
                    suspectObj.update_table()
                elif choice == 4:
                    suspectObj.delete_table()
                elif choice == 5:
                    (suspectObj.select_table())
                elif choice == 6:
                    print("Exited successfully")
                    break
                else:
                    print("Wrong choice")

        elif ch == 4:
            while True:
                print("1.create law agencies\t2.insert law agencies\t3.update
law agencies\n4.delete law agencies\t5.select law agencies\n6.Exit")
                choice = int(input("enter your choice:"))
                if choice == 1:
                    lawObj.create_table()
                elif choice == 2:
                    lawObj.insert_into()
```

```
                elif choice == 3:
                    lawObj.update_table()
                elif choice == 4:
                    lawObj.delete_table()
                elif choice == 5:
                    lawObj.select_table()
                elif choice == 6:
                    print("Exited successfully")
                    break
                else:
                    print("Wrong choice")

        elif ch == 5:
            while True:
                print("1.create officers\t2.insert officers\t3.update
officers\n4.delete officers\t5.select officers\n6.Exit")
                choice = int(input("enter your choice"))
                if choice == 1:
                    officerObj.create_table()
                elif choice == 2:
                    officerObj.insert_into()
                elif choice == 3:
                    officerObj.update_table()
                elif choice == 4:
                    officerObj.delete_table()
                elif choice == 5:
                    officerObj.select_table()
                elif choice == 6:
                    print("Exited successfully")
                    break
                else:
                    print("Wrong choice")

        elif ch == 6:
            while True:
                print("1.create evidence\t2.insert evidence\t3.update
evidence\n4.delete evidence\t5.select evidence\n6.Exit")
                choice = int(input("enter your choice"))
                if choice == 1:
                    evidenceObj.create_table()
                elif choice == 2:
                    evidenceObj.insert_into()
                elif choice == 3:
                    evidenceObj.update_table()
                elif choice == 4:
                    evidenceObj.delete_table()
                elif choice == 5:
                    evidenceObj.select_table()
                elif choice == 6:
                    print("Exited successfully")
                    break
                else:
                    print("Wrong choice")
        elif ch == 7:
            while True:
                print("1.create reports\t2.insert reports\t3.update
reports\n4.delete reports\t5.select reports\n6.Exit")
                choice = int(input("enter your choice"))
                if choice == 1:
                    reportObj.create_table()
                elif choice == 2:
```

```
                    reportObj.insert_into()
                elif choice == 3:
                    reportObj.update_table()
                elif choice == 4:
                    reportObj.delete_table()
                elif choice == 5:
                    reportObj.select_table()
                elif choice == 6:
                    print("Exited successfully")
                    break
                else:
                    print("Wrong choice")

        elif ch == 8:
            while True:
                print("1.create Incident\t2.update incident status\t3.get
incidents in date range officers\n4.search incidents\t5.generate incident
report\n6.create case\t7.get case details\t8.update case details\t9.get all
cases\t10.exit")
                choice = int(input("enter your choice"))
                if choice == 1:
                    serviceImplementObj.createIncident()
                elif choice == 2:
                    serviceImplementObj.updateIncidentStatus()
                elif choice == 3:
                    serviceImplementObj.getIncidentsInDateRange()
                elif choice == 4:
                    serviceImplementObj.searchIncidents()
                elif choice == 5:
                    serviceImplementObj.generateIncidentReport()
                elif choice == 6:
                    serviceImplementObj.createCase()
                elif choice == 7:
                    serviceImplementObj.getCaseDetails()
                elif choice == 8:
                    serviceImplementObj.updateCaseDetails()
                elif choice == 9:
                    serviceImplementObj.getAllCases()
                elif choice == 10:
                    print("Exited successfully")
                    break
                else:
                    print("Wrong choice")

        elif ch == 9:
            print("Exited successfully")
            break

        else:
            print("Wrong choice")

except Exception as e:
    print(f"Unhandled error: {e}")

finally:
    DBConnection.connection.close()
    print("Database connection closed")
```

Exception_Handling.py

```
class IncidentNumberNotFoundException(Exception):
    def __init__(self, msg="Incident id not found"):
        self.msg = msg
        super().__init__(msg)
```

Testing.py

```
import unittest
from Case_Study.DAO.INCIDENTS import Incidents


class MyTestCase(unittest.TestCase):
    def setUp(self):
        self.incident = Incidents()

    # testing whether an incident is created or not
    def test_incident(self):
        print("Create a new incident with incident id = 5")
        result = self.incident.insert_into()
        self.assertEqual('Incident created successfully', result)

    # testing whether incident status updated or not
    def test_update(self):
        print("Updating the status of incident. Set status = Investigation ")
        result = self.incident.update_table()
        self.assertEqual('Values updated successfully', result)


if __name__ == '__main__':
    unittest.main()
```

```
Database connected successfully
Select table to use functionalities
1.Incidents
2.Victims
3.Suspects
4.Law Enforcement Agencies
5.Officers
6.Evidence
7.Reports
8.crime analysis service impl
9.exit
enter your choice:
```

```
enter your choice:1
1.create Incidents  2.insert Incidents  3.update incidents
4.delete incidents  5.select incidents
6.Exit
enter your choice:
```

```
enter your choice:2
Enter the incident id: 2
Enter the incident type: shootout
Enter the incident date: 2024-02-07
Enter the location: lokhandwala
Enter the description: murders
Enter the status: open
Enter the victim id: 2
Enter the suspect id: 2
Data inserted successfully
```

```
mysql> select*from incidents;
+-------------+---------------+---------------+---------------+-----------------------------+--------+-----------+------------+
| incident_id | incident_type | incident_date | location      | description                 | status | victim_id | suspect_id |
+-------------+---------------+---------------+---------------+-----------------------------+--------+-----------+------------+
|           1 | robbery       | 2024-02-07    | police station | van stole from police station | open   |         1 |          1 |
|           2 | shootout      | 2024-02-07    | lokhandwala   | murders                     | open   |         2 |          2 |
+-------------+---------------+---------------+---------------+-----------------------------+--------+-----------+------------+
2 rows in set (0.01 sec)
```

```
enter your choice:2
1.create victims    2.insert victims    3.update victims
4.delete victims    5.select victims
6.Exit
enter your choice:1
Victims table created successfully
```

```
mysql> show tables;
+---------------------------------------+
| Tables_in_crime_reporting_system      |
+---------------------------------------+
| incidents                             |
| victims                               |
+---------------------------------------+
2 rows in set (0.00 sec)
```

```
enter your choice:3
1.create suspects   2.insert suspects   3.update suspects
4.delete suspects   5.select suspects
6.Exit
enter your choice:1
Suspects table created successfully
```

```
mysql> show tables;
+-----------------------------------+
| Tables_in_crime_reporting_system  |
+-----------------------------------+
| incidents                         |
| suspects                          |
| victims                           |
+-----------------------------------+
3 rows in set (0.00 sec)
```

```
enter your choice:4
1.create law agencies   2.insert law agencies   3.update law agencies
4.delete law agencies   5.select law agencies
6.Exit
enter your choice:1
Law Enforcement Agencies table created successfully
```

```
enter your choice:5
1.create officers   2.insert officers   3.update officers
4.delete officers   5.select officers
6.Exit
enter your choice1
Officers table created successfully
```

```
enter your choice:6
1.create evidence   2.insert evidence   3.update evidence
4.delete evidence   5.select evidence
6.Exit
enter your choice1
Evidence table created successfully
```

```
enter your choice:7
1.create reports    2.insert reports    3.update reports
4.delete reports    5.select reports
6.Exit
enter your choice1
Reports table successfully created
```

```
enter your choice:1
1.create Incidents  2.insert Incidents  3.update incidents
4.delete incidents  5.select incidents
6.Exit
enter your choice:7
Wrong choice
```