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

K. MEGHASYAM

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

```
CREATE TABLE Incidents (
    IncidentID INT PRIMARY KEY,
    IncidentType VARCHAR(255),
    IncidentDate DATE,
    Location VARCHAR(255),
    Description TEXT,
    Status VARCHAR(50),
    VictimID INT,
    SuspectID INT,
    FOREIGN KEY (VictimID) REFERENCES Victims(VictimID),
    FOREIGN KEY (SuspectID) REFERENCES Suspects(SuspectID)
);
-- Insert data into Incidents table
INSERT INTO Incidents (IncidentID, IncidentType, IncidentDate, Location, Description, Status, VictimID, SuspectID)
VALUES
```

(1, 'Robbery', '2023-01-10', '123 Main St, City, State, Zip, USA', 'Armed robbery at a convenience store.', 'Open', 1, 1),

(2, 'Homicide', '2023-02-15', '456 Elm St, City, State, Zip, USA', 'Fatal shooting incident.', 'Closed', 2, 2),

(3, 'Theft', '2023-03-20', '789 Oak St, City, State, Zip, USA', 'Burglary at a residence.', 'Under Investigation', 3, 3);

```
mysql> select * from Incidents;
+------------+--------------+--------------+-----------------+-----------------------------------+--------------------+----------+-----------+
| IncidentID | IncidentType | IncidentDate | Location        | Description                       | Status             | VictimID | SuspectID |
+------------+--------------+--------------+-----------------+-----------------------------------+--------------------+----------+-----------+
|          1 | Robbery      | 2023-01-10   | 123 Main St, USA | Armed robbery at a convenience store. | closed             |        1 |         1 |
|          2 | Homicide     | 2023-02-15   | 456 Elm St, USA  | Fatal shooting incident.          | open               |        2 |         2 |
|          3 | Theft        | 2023-03-20   | 789 Oak St, USA  | Burglary at a residence.          | Under Investigation |        3 |         3 |
+------------+--------------+--------------+-----------------+-----------------------------------+--------------------+----------+-----------+
3 rows in set (0.02 sec)
```

2. Victims:

• VictimID (Primary Key)

• FirstName

• LastName

• DateOfBirth

• Gender

• Contact Information (e.g., Address, Phone Number)

```
-- Create Victims table
CREATE TABLE Victims (
    VictimID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    DateOfBirth DATE,
    Gender VARCHAR(10),
    ContactInformation TEXT
);
-- Insert data into Victims table
INSERT INTO Victims (VictimID, FirstName, LastName, DateOfBirth, Gender, ContactInformation) VALUES
    (1, 'Sai', 'Karri', '1990-05-15', 'Male', '123 Main St, No:8976678776, USA'),
    (2, 'Arun', 'Dantu', '1985-09-20', 'Female', '456 Elm St, No:7683455445,USA'),
    (3, 'Lohith', 'Namburi', '1978-03-10', 'Male', '789 Oak St, No:7834653775, USA');
```

```
mysql> select * from victims;
+----------+-----------+----------+-------------+--------+---------------------------------+
| VictimID | FirstName | LastName | DateOfBirth | Gender | ContactInformation              |
+----------+-----------+----------+-------------+--------+---------------------------------+
|        1 | Sai       | Karri    | 1990-05-15  | Male   | 123 Main St, No:8976678776, USA |
|        2 | Arun      | Dantu    | 1985-09-20  | Female | 456 Elm St, No:7683455445, USA  |
|        3 | Lohith    | Namburi  | 1978-03-10  | Male   | 789 Oak St, No:7834653775, USA  |
|        4 | Naveen    | Utala    | 1980-04-05  | Male   |  324 Food St, No:6789653775, USA |
+----------+-----------+----------+-------------+--------+---------------------------------+
```

3. Suspects:

• SuspectID (Primary Key)

• FirstName

• LastName

• DateOfBirth

• Gender

• Contact Information

```
-- Create Suspects table
CREATE TABLE Suspects (
    SuspectID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    DateOfBirth DATE,
    Gender VARCHAR(10),
    ContactInformation TEXT
);
INSERT INTO Suspects (SuspectID, FirstName, LastName, DateOfBirth, Gender, ContactInformation) VALUES
    (1, 'Robert', 'Williams', '1982-07-25', 'Male', '321 Pine St, USA'),
    (2, 'Emily', 'Davis', '1995-12-18', 'Female', '654 Birch St, USA'),
    (3, 'David', 'Martinez', '1989-06-30', 'Male', '987 Cedar St, USA');
```

```
mysql> select * from suspects;
+-----------+-----------+----------+-------------+--------+---------------------------------+
| SuspectID | FirstName | LastName | DateOfBirth | Gender | ContactInformation              |
+-----------+-----------+----------+-------------+--------+---------------------------------+
|         1 | Robert    | Williams | 1982-07-25  | Male   | 321 Pine St, No:678778875, USA  |
|         2 | Emily     | Davis    | 1995-12-18  | Female | 654 Birch St, No:8978866767, USA |
|         3 | David     | Martinez | 1989-06-30  | Male   | 987 Cedar St, No:9703766576, USA |
|         4 | Joe       | Dan      | 1990-03-09  | Male   | 217 jar St, No:8703766576, USA  |
+-----------+-----------+----------+-------------+--------+---------------------------------+
```

4. Law Enforcement Agencies:

• AgencyID (Primary Key)

• AgencyName

• Jurisdiction

• Contact Information

• Officer(s) (Link to Officers within the agency)

```
-- Create Law Enforcement Agencies table
CREATE TABLE LawEnforcementAgencies (
    AgencyID INT PRIMARY KEY,
    AgencyName VARCHAR(255),
    Jurisdiction VARCHAR(255),
    ContactInformation TEXT
);
-- Insert data into LawEnforcementAgencies table
INSERT INTO LawEnforcementAgencies (AgencyID, AgencyName, Jurisdiction, ContactInformation)
VALUES
    (1, 'City Police Department', 'City', '123-456-7890, citypd@example.com'),
    (2, 'County Sheriff Office', 'County', '987-654-3210, sheriff@example.com'),
    (3, 'State Bureau of Investigation', 'State', '555-555-5555, sbi@example.com');
```

```
mysql> select * from lawenforcementagencies;
+----------+-------------------------------+-------------+-----------------------------------+
| AgencyID | AgencyName                    | Jurisdiction | ContactInformation                |
+----------+-------------------------------+-------------+-----------------------------------+
|        1 | City Police Department        | City        | 123-456-7890, citypd@example.com  |
|        2 | County Sheriff Office         | County      | 987-654-3210, sheriff@example.com |
|        3 | State Bureau of Investigation | State       | 555-555-5555, sbi@example.com     |
+----------+-------------------------------+-------------+-----------------------------------+
```

5. Officers:

• OfficerID (Primary Key)

• FirstName

• LastName

• BadgeNumber

• Rank

• Contact Information

• AgencyID (Foreign Key, linking to Law Enforcement Agencies)

```
-- Create Officers table
CREATE TABLE Officers (
    OfficerID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    BadgeNumber VARCHAR(50),
    Rank VARCHAR(50),
    ContactInformation TEXT,
    AgencyID INT,
    FOREIGN KEY (AgencyID) REFERENCES LawEnforcementAgencies(AgencyID)
);
-- Insert data into Officers table
INSERT INTO Officers (OfficerID, FirstName, LastName, BadgeNumber, `Rank`, ContactInformation, AgencyID)
VALUES
    (1, 'Mark', 'Johnson', '1234', 'Sergeant', 'mark.johnson@citypd.com', 1),
    (2, 'Sarah', 'Brown', '5678', 'Detective', 'sarah.brown@sheriff.com', 2),
    (3, 'Chris', 'Wilson', '91011', 'Captain', 'chris.wilson@sbi.com', 3);
```

```
mysql> select * from officers;
+-----------+-----------+----------+-------------+-----------+-------------------------+----------+
| OfficerID | FirstName | LastName | BadgeNumber | Rank      | ContactInformation      | AgencyID |
+-----------+-----------+----------+-------------+-----------+-------------------------+----------+
|         1 | Mark      | Johnson  | 1234        | Sergeant  | mark.johnson@citypd.com |        1 |
|         2 | Sarah     | Brown    | 5678        | Detective | sarah.brown@sheriff.com |        2 |
|         3 | Chris     | Wilson   | 91011       | Captain   | chris.wilson@sbi.com    |        3 |
+-----------+-----------+----------+-------------+-----------+-------------------------+----------+
```

6. Evidence:

• EvidenceID (Primary Key)

• Description

• Location Found

• IncidentID (Foreign Key, linking to Incidents)

```
-- Create Evidence table
CREATE TABLE Evidence (
    EvidenceID INT PRIMARY KEY,
    Description TEXT,
    LocationFound VARCHAR(255),
    IncidentID INT,
    FOREIGN KEY (IncidentID) REFERENCES Incidents(IncidentID)
);
-- Insert data into Evidence table
INSERT INTO Evidence (EvidenceID, Description, LocationFound, IncidentID)
VALUES
    (1, 'Weapon used in the robbery', 'Behind the convenience store', 1),
    (2, 'Shell casings found at the crime scene', 'In the victim\'s home', 2),
    (3, 'Fingerprint evidence', 'On the stolen items', 3);
```

```
mysql> select * from evidence;
+------------+---------------------------------------+-------------------------------+------------+
| EvidenceID | Description                           | LocationFound                 | IncidentID |
+------------+---------------------------------------+-------------------------------+------------+
|          1 | Weapon used in the robbery            | Behind the convenience store |          1 |
|          2 | Shell casings found at the crime scene | In the victim's home         |          2 |
|          3 | Fingerprint evidence                  | On the stolen items          |          3 |
+------------+---------------------------------------+-------------------------------+------------+
```

7. Reports:

• ReportID (Primary Key)

• IncidentID (Foreign Key, linking to Incidents)

• ReportingOfficer (Foreign Key, linking to Officers)

• ReportDate

• ReportDetails

• Status (e.g., Draft, Finalized)

```
-- Create Reports table
```

```
CREATE TABLE Reports (
   ReportID INT PRIMARY KEY,
   IncidentID INT,
   ReportingOfficer INT,
   ReportDate DATE,
   ReportDetails TEXT,
   Status VARCHAR(50),
   FOREIGN KEY (IncidentID) REFERENCES Incidents(IncidentID),
   FOREIGN KEY (ReportingOfficer) REFERENCES Officers(OfficerID)
);
-- Insert data into Reports table
INSERT INTO Reports (ReportID, IncidentID, ReportingOfficer, ReportDate, ReportDetails, Status)
VALUES
   (1, 1, 1, '2023-01-12', 'Initial incident report filed.', 'Draft'),
   (2, 2, 2, '2023-02-17', 'Finalized investigation report submitted.', 'Finalized'),
   (3, 3, 3, '2023-03-25', 'Investigation ongoing.', 'Draft');
```

```
mysql> select * from reports;
+----------+------------+-----------------+------------+----------------------------------------+-----------+
| ReportID | IncidentID | ReportingOfficer | ReportDate | ReportDetails                          | Status    |
+----------+------------+-----------------+------------+----------------------------------------+-----------+
|        1 |          1 |               1 | 2023-01-12 | Initial incident report filed.         | Draft     |
|        2 |          2 |               2 | 2023-02-17 | Finalized investigation report submitted. | Finalized |
|        3 |          3 |               3 | 2023-03-25 | Investigation ongoing.                 | Draft     |
+----------+------------+-----------------+------------+----------------------------------------+-----------+
```

Coding

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

Incidents Entity Class:

```
class Incidents:
    def __init__(self, incident_id, incident_type, incident_date, location, description,
status, victim_id, suspect_id):
        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
```

```
    # Getters
    def get_incident_id(self):
        return self.__incident_id

    def get_incident_type(self):
        return self.__incident_type

    def get_incident_date(self):
        return self.__incident_date

    def get_location(self):
        return self.__location

    def get_description(self):
        return self.__description

    def get_status(self):
        return self.__status

    def get_victim_id(self):
        return self.__victim_id

    def get_suspect_id(self):
        return self.__suspect_id

    # Setters
    def set_incident_id(self, incident_id):
        self.__incident_id = incident_id

    def set_incident_type(self, incident_type):
        self.__incident_type = incident_type

    def set_incident_date(self, incident_date):
        self.__incident_date = incident_date

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

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

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

    def set_victim_id(self, victim_id):
        self.__victim_id = victim_id

    def set_suspect_id(self, suspect_id):
        self.__suspect_id = suspect_id
```

Victims Class:

```
class Victims:
    def __init__(self, victim_id, first_name, last_name, date_of_birth, gender,
contact_information):
        self.__victim_id = victim_id
        self.__first_name = first_name
        self.__last_name = last_name
        self.__date_of_birth = date_of_birth
        self.__gender = gender
        self.__contact_information = contact_information
```

```
    # Getters
    def get_victim_id(self):
        return self.__victim_id

    def get_first_name(self):
        return self.__first_name

    def get_last_name(self):
        return self.__last_name

    def get_date_of_birth(self):
        return self.__date_of_birth

    def get_gender(self):
        return self.__gender

    def get_contact_information(self):
        return self.__contact_information

    # Setters
    def set_victim_id(self, victim_id):
        self.__victim_id = victim_id

    def set_first_name(self, first_name):
        self.__first_name = first_name

    def set_last_name(self, last_name):
        self.__last_name = last_name

    def set_date_of_birth(self, date_of_birth):
        self.__date_of_birth = date_of_birth

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

    def set_contact_information(self, contact_information):
        self.__contact_information = contact_information
```

Suspects Class:

```
class Suspects:
    def __init__(self, suspect_id, first_name, last_name, date_of_birth, gender,
contact_information):
        self.__suspect_id = suspect_id
        self.__first_name = first_name
        self.__last_name = last_name
        self.__date_of_birth = date_of_birth
        self.__gender = gender
        self.__contact_information = contact_information

    # Getters
    def get_suspect_id(self):
        return self.__suspect_id

    def get_first_name(self):
        return self.__first_name

    def get_last_name(self):
        return self.__last_name

    def get_date_of_birth(self):
```

```
        return self.__date_of_birth

    def get_gender(self):
        return self.__gender

    def get_contact_information(self):
        return self.__contact_information

    # Setters
    def set_suspect_id(self, suspect_id):
        self.__suspect_id = suspect_id

    def set_first_name(self, first_name):
        self.__first_name = first_name

    def set_last_name(self, last_name):
        self.__last_name = last_name

    def set_date_of_birth(self, date_of_birth):
        self.__date_of_birth = date_of_birth

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

    def set_contact_information(self, contact_information):
        self.__contact_information = contact_information
```

LawEnforcementAgencies Class:

```
class LawEnforcementAgencies:
    def __init__(self, agency_id, agency_name, jurisdiction, contact_information):
        self.__agency_id = agency_id
        self.__agency_name = agency_name
        self.__jurisdiction = jurisdiction
        self.__contact_information = contact_information

    # Getters
    def get_agency_id(self):
        return self.__agency_id

    def get_agency_name(self):
        return self.__agency_name

    def get_jurisdiction(self):
        return self.__jurisdiction

    def get_contact_information(self):
        return self.__contact_information

    # Setters
    def set_agency_id(self, agency_id):
        self.__agency_id = agency_id

    def set_agency_name(self, agency_name):
        self.__agency_name = agency_name

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

    def set_contact_information(self, contact_information):
        self.__contact_information = contact_information
```

Officers Class:

```
class Officers:
    def __init__(self, officer_id, first_name, last_name, badge_number, rank,
contact_information, agency_id):
        self.__officer_id = officer_id
        self.__first_name = first_name
        self.__last_name = last_name
        self.__badge_number = badge_number
        self.__rank = rank
        self.__contact_information = contact_information
        self.__agency_id = agency_id

    # Getters
    def get_officer_id(self):
        return self.__officer_id

    def get_first_name(self):
        return self.__first_name

    def get_last_name(self):
        return self.__last_name

    def get_badge_number(self):
        return self.__badge_number

    def get_rank(self):
        return self.__rank

    def get_contact_information(self):
        return self.__contact_information

    def get_agency_id(self):
        return self.__agency_id

    # Setters
    def set_officer_id(self, officer_id):
        self.__officer_id = officer_id

    def set_first_name(self, first_name):
        self.__first_name = first_name

    def set_last_name(self, last_name):
        self.__last_name = last_name

    def set_badge_number(self, badge_number):
        self.__badge_number = badge_number

    def set_rank(self, rank):
        self.__rank = rank

    def set_contact_information(self, contact_information):
        self.__contact_information = contact_information

    def set_agency_id(self, agency_id):
        self.__agency_id = agency_id
```

Reports Class:

```
class Reports:
    def __init__(self, report_id, incident_id, reporting_officer, report_date,
report_details, status):
```

```
        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

    # Getters
    def get_report_id(self):
        return self.__report_id

    def get_incident_id(self):
        return self.__incident_id

    def get_reporting_officer(self):
        return self.__reporting_officer

    def get_report_date(self):
        return self.__report_date

    def get_report_details(self):
        return self.__report_details

    def get_status(self):
        return self.__status

    # Setters
    def set_report_id(self, report_id):
        self.__report_id = report_id

    def set_incident_id(self, incident_id):
        self.__incident_id = incident_id

    def set_reporting_officer(self, reporting_officer):
        self.__reporting_officer = reporting_officer

    def set_report_date(self, report_date):
        self.__report_date = report_date

    def set_report_details(self, report_details):
        self.__report_details = report_details

    def set_status(self, status):
        self.__status = status
```

Evidence Class:

```
class Evidence:
    def __init__(self, evidence_id, description, location_found, incident_id):
        self.__evidence_id = evidence_id
        self.__description = description
        self.__location_found = location_found
        self.__incident_id = incident_id

    # Getters
    def get_evidence_id(self):
        return self.__evidence_id

    def get_description(self):
        return self.__description

    def get_location_found(self):
```

```
        return self.__location_found

    def get_incident_id(self):
        return self.__incident_id

    # Setters
    def set_evidence_id(self, evidence_id):
        self.__evidence_id = evidence_id

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

    def set_location_found(self, location_found):
        self.__location_found = location_found

    def set_incident_id(self, incident_id):
        self.__incident_id = incident_id
```

Service Provider Interface/Abstract class

• Keep the interfaces and implementation classes in package dao .Create ICrimeAnalysisService Interface/abstract classs with the following method

// Create a new incident

createIncident();

parameters- Incident object

return type Boolean

```
from util.DBConnUtil import dbConnection
from entity.Incidents import  Incidents
from entity.ICase import  ICase
from Dao.Icase import ICase
from MyExceptions.InvalidNameError import InvalidNameError , StringCheck
from MyExceptions.IncidentNumberNotFoundException import IncidentNumberNotFoundException
class  ICrimeAnalysisService(dbConnection):
    def __init__(self):
        self.incident_id = " "
        self.incident_type = " "
        self.incident_date = " "
        self.location = " "
        self.description = " "
        self.status = " "
        self.victim_id = " "
        self.suspect_id = " "
        print(self.incident_id
,self.incident_type,self.incident_date,self.location,self.description
,self.status,self.victim_id,self.suspect_id )

    def create_incident(self):
        try:

            incident_id=int(input("enter incident id: "))
            self.incident_id=incident_id

            incident_type=input("enter incident type: ")
            self.incident_type=incident_type
```

```
            incident_date=input("enter incident date in (yyyy-mm-dd) format: ")
            self.incident_date=incident_date

            location=input("enter location: ")
            self.location=location

            description=input("enter description about incident: ")
            self.description=description

            status=input("enter status: ")
            self.status=status

            victim_id=int(input("enter victim id: "))
            self.victim_id=victim_id

            suspect_id=int(input("enter suspect id:"))
            self.suspect_id=suspect_id


            # SQL query to insert a new incident
            sql_query = """
                        INSERT INTO Incidents (IncidentID, IncidentType, IncidentDate,
Location, Description, Status, victimid, suspectid)
                        VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
                        """
            # Parameters for the SQL query
            values = [(incident_id, incident_type, incident_date,
                       location, description, status, victim_id, suspect_id)]
            self.open()
            self.stmt.executemany(sql_query, values)
            self.conn.commit()
            print('Records Inserted Successfully..')
            self.close()
            return True  # Return True if the operation is successful
        except Exception as e:
            print(f"Error creating incident: {e}")
            return False  # Return False if the operation fails
```

O/P:

```
enter your choice1
enter incident id: 4
enter incident type: Murder
enter incident date in (yyyy-mm-dd) format: 2023-04-09
enter location: 387 Ma St,Usa
enter description about incident: Pre-Planned Murder
enter status: open
enter victim id: 4
enter suspect id:4
--Database Is Connected--
Records Inserted Successfully..
```

```
_________________Records In Incidents Table_________________________
(1, 'Robbery', datetime.date(2023, 1, 10), '123 Main St, USA', 'Armed robbery at a convenience store.', 'closed', 1, 1)
(2, 'Homicide', datetime.date(2023, 2, 15), '456 Elm St, USA', 'Fatal shooting incident.', 'open', 2, 2)
(3, 'Theft', datetime.date(2023, 3, 20), '789 Oak St, USA', 'Burglary at a residence.', 'Under Investigation', 3, 3)
(4, 'Murder', datetime.date(2023, 4, 9), '387 Ma St,Usa', 'Pre-Planned Murder', 'open', 4, 4)
```

// Update the status of an incident

updateIncidentStatus();

parameters- Status object,incidentid

return type Boolean

```
def update_incident_status(self):
    try:
        self.select()
        incidentid=int(input("enter incidentid: "))
        check_query = """ SELECT Incidentid from Incidents"""
        self.incidentid = incidentid
        self.open()
        ids = self.stmt.execute(check_query)
        recods = self.stmt.fetchall()
        lists = [i[0] for i in recods]
        if incidentid in lists:
            status=input("enter status to get updated: ")
            self.status=status

            # SQL query to update the status of an incident
            sql_query = """ UPDATE Incidents
                            SET Status = %s
                            WHERE IncidentID = %s
                        """
            data=[(status,incidentid)]
            self.open()
            self.stmt.executemany(sql_query,data )
            self.conn.commit()
            print('Records Updated Successfully..')
            self.select()
            self.close()
            return True  # Return True if the operation is successful
        else:
            raise IncidentNumberNotFoundException()
    except Exception as e:
        print(f"Error updating incident status: {e}")
        return False  # Return False if the operation fails
```

O/P:

```
enter incidentid: 1
--Database Is Connected--
enter status to get updated: open
--Database Is Connected--
Records Updated Successfully..
```

```
_________________Records In Incidents Table_________________________
(1, 'Robbery', datetime.date(2023, 1, 10), '123 Main St, USA', 'Armed robbery at a convenience store.', 'open', 1, 1)
```

// Get a list of incidents within a date range

getIncidentsInDateRange();

parameters- startDate, endDate

return type Collection of Incident objects

```
def get_incidents_in_date_range(self, start_date, end_date):

        # SQL query to select incidents within the specified date range
        sql_query = """ SELECT * FROM Incidents
                        WHERE IncidentDate BETWEEN %s AND %s
                    """
        # Parameters for the SQL query
        values = (start_date, end_date)
        self.open()
        self.stmt.execute(sql_query,values)
        recods = self.stmt.fetchall()
        print('')
        print('_________________Records In Date Range _____________________')
        for i in recods:
            print(i)
        self.close()

        if not recods:
            print("No Incidents Found in the Given Dates")
        return f'Incidents details fetched successfully'
```

O/P:

```
enter your choice3
enter startdate in yyyy-mm-dd: 2023-01-10
enter enddate in yyyy-mm-dd: 2023-03-21
--Database Is Connected--


_________________Records In Date Range _________________________
(1, 'Robbery', datetime.date(2023, 1, 10), '123 Main St, USA', 'Armed robbery at a convenience store.', 'open', 1, 1)
(2, 'Homicide', datetime.date(2023, 2, 15), '456 Elm St, USA', 'Fatal shooting incident.', 'open', 2, 2)
(3, 'Theft', datetime.date(2023, 3, 20), '789 Oak St, USA', 'Burglary at a residence.', 'Under Investigation', 3, 3)
Connection Closed.
```

// Search for incidents based on various criteria

searchIncidents(IncidentType criteria);

parameters- IncidentType object

return type Collection of Incident objects

```
def search_incidents(self, incident_type):
    try:
        check_query=""" SELECT IncidentType FROM Incidents"""
        self.open()
        self.stmt.execute(check_query)
        records=self.stmt.fetchall()
        checking=[i[0] for i in records]
```

```
        print(checking)
        self.close()
        if not isinstance(incident_type,str):
            print("invalid data type")
        incident_type=incident_type.lower()
        for incidenttype in checking:
            incidenttype=incidenttype.lower()
            if incident_type==incidenttype:
                # SQL query to search for incidents based on the given criteria
                sql_query = """
                                SELECT *
                                FROM Incidents
                                WHERE IncidentType = %s
                            """
                # Parameters for the SQL query
                values = [(incident_type)]
                self.open()
                self.stmt.execute(sql_query, values)
                recods = self.stmt.fetchall()
                for i in recods:
                    print(i)
                self.close()
            else:
                print(f"No Incident found with Incident type: {incident_type}")

            return f'Incidents details fetched successfully'
    except Exception as e:
        print(f"An unexpected error occurred: {str(e)}")
```

O/P:

```
enter your choice4
enter Incident type: robbery
--Database Is Connected--
['Robbery', 'Homicide', 'Theft', 'Murder']
Connection Closed.
--Database Is Connected--
(1, 'Robbery', datetime.date(2023, 1, 10), '123 Main St, USA', 'Armed robbery at a convenience store.', 'open', 1, 1)
Connection Closed.
```

// Generate incident reports

generateIncidentReport();

parameters- Incident object

return type Report object

```
def generate_incident_report(self,incidentid):
    try:
        self.open()
        select_Incidents_str = '''
                                SELECT * FROM Reports
                                WHERE IncidentID = %s
                            '''
        self.stmt.execute(select_Incidents_str, (incidentid,))
        customer_data = self.stmt.fetchone()

        if not customer_data:
                raise IncidentNumberNotFoundException()
        else:
```

```
                print('\nIncident Report:')
                print(f"ReportID: {customer_data[0]}")
                print(f"IncidentID: {customer_data[1]}")
                print(f"ReportingOfficer: {customer_data[2]}")
                print(f"ReportDate: {customer_data[3]}")
                print(f"ReportDetails: {customer_data[4]}")
                print(f"Status: {customer_data[5]}")
                # print(f"VictimID: {customer_data[6]}")
                # print(f"SuspectID: {customer_data[7]}")

        self.close()
    except Exception as e:
        print(f"An unexpected error occurred: {str(e)}")
```

O/P:

```
enter your choice5
enter incident id: 3
--Database Is Connected--

Incident Report:
ReportID: 3
IncidentID: 3
ReportingOfficer: 3
ReportDate: 2023-03-25
ReportDetails: Investigation ongoing.
Status: Draft
Connection Closed.
```

// Create a new case and associate it with incidents

createCase();

parameters- caseDescription string, collection of Incident Objects

return type Case object

```
def create_case(self):
    try:
        # Description = input("enter description about case: ")
        # self.Description=Description

        victim_id = int(input("enter victim id: "))
        self.victim_id = victim_id

        suspect_id = int(input("enter suspect id: "))
        self.suspect_id = suspect_id

        officer_id = int(input("enter officer id: "))
        self.officer_id = officer_id

        query=  """ INSERT INTO ICase(IncidentID, IncidentType, VictimName, SuspectName,
OfficerName, CaseDescription)
            SELECT
                i.IncidentID,
                i.IncidentType,
                CONCAT(v.FirstName, ' ', v.LastName) AS VictimName,
                CONCAT(s.FirstName, ' ', s.LastName) AS SuspectName,
                CONCAT(o.FirstName, ' ', o.LastName) AS OfficerName,
                i.description  as CaseDescription
            FROM
```

```
                Incidents i
                LEFT JOIN Victims v ON i.VictimID = v.VictimID
                LEFT JOIN Suspects s ON i.SuspectID = s.SuspectID
                LEFT JOIN Officers o ON i.IncidentID = o.OfficerID
            WHERE

                i.VictimID = %s
                AND i.SuspectID = %s
                AND i.IncidentID = %s
            """
            # Parameters for the SQL query
    values = [(victim_id, suspect_id, officer_id)]
    self.open()
    self.stmt.executemany(query, values)
    self.conn.commit()
    print('Records Inserted Successfully..')
    self.close()
    return True  # Return True if the operation is successful
except Exception as e:
    print(f"Error creating case: {e}")
    return False  # Return False if the operation
```

O/P:

```
enter your choice6
enter victim id: 3
enter suspect id: 3
enter officer id: 3
--Database Is Connected--
Records Inserted Successfully..
```

```
_________________Records In Case Table_____________________
(1, 1, 'Robbery', 'Sai Karri', 'Robert Williams', 'Mark Johnson', 'Armed robbery at a convenience store.')
(2, 2, 'Homicide', 'Arun Dantu', 'Emily Davis', 'Sarah Brown', 'Fatal shooting incident.')
(6, 3, 'Theft', 'Lohith Namburi', 'David Martinez', 'Chris Wilson', 'Burglary at a residence.')
```

// Get details of a specific case

Case getCaseDetails(int caseId);

parameters- caseDescription string, collection of Incident Objects

return type Case object

```
def CaseDetails(self, Caseid):
    try:
        # if not isinstance(Caseid, int) or Caseid < 0:
        #     raise InvalidIDError()
        self.open()
        select_customer_str = '''
            SELECT * FROM ICase
            WHERE CaseID = %s
        '''
        self.stmt.execute(select_customer_str, (Caseid,))
        customer_data = self.stmt.fetchone()

        if not customer_data:
            print(f"No Case found with CaseID: {Caseid}")
        else:
```

```
            print('\nCase Details:')
            print(f"CaseID: {customer_data[0]}")
            print(f"IncidentID: {customer_data[1]}")
            print(f"IncidentType: {customer_data[2]}")
            print(f"VictimName: {customer_data[3]}")
            print(f"SuspectName: {customer_data[4]}")
            print(f"OfficerName: {customer_data[5]}")
            print(f"CaseDescription: {customer_data[6]}")
        self.close()
    except Exception as e:
        print(f"An unexpected error occurred: {str(e)}")
```

O/P:

```
enter Caseid: 6
--Database Is Connected--

Case Details:
CaseID: 6
IncidentID: 3
IncidentType: Theft
VictimName: Lohith Namburi
SuspectName: David Martinez
OfficerName: Chris Wilson
CaseDescription: Burglary at a residence.
```

// Update case details

updateCaseDetails();

parameters- Case object

return type boolean

```
def updateCaseDetails(self,caseid):
    try:
        self.view_allcases()
        check_query=""" SELECT CaseId from ICase"""
        self.open()
        ids=self.stmt.execute(check_query)
        recods = self.stmt.fetchall()
        lists=[i[0] for i in recods]
        print(lists)
        self.close()
        if caseid in lists:
            Id = caseid
            update_str = 'UPDATE ICase SET '
            data = []

            incidentid= input('Enter incidentID :')
            self.incidentid = incidentid
            update_str += 'incidentid=%s, '
            data.append(self.incidentid)

            IncidentType = input('Enter IncidentType :')
            if IncidentType:
```

```
                self.IncidentType = IncidentType
                update_str += 'IncidentType=%s, '
                data.append(self.IncidentType)

            VictimName = input('Enter VictimName: ')
            if VictimName:
                self.VictimName = VictimName
                update_str += 'VictimName=%s, '
                data.append(self.VictimName)

            SuspectName = input('Enter SuspectName : ')
            if SuspectName:
                self.SuspectName = SuspectName
                update_str += 'SuspectName=%s, '
                data.append(self.SuspectName)

            OfficerName = input('Enter OfficerName : ')
            if OfficerName:
                self.OfficerName = OfficerName
                update_str += 'OfficerName=%s, '
                data.append(self.OfficerName)

            CaseDescription = input('Enter CaseDescription : ')
            if CaseDescription:
                self.CaseDescription =CaseDescription
                update_str += 'CaseDescription=%s, '
                data.append(self.CaseDescription)

            update_str = update_str.rstrip(', ')
            update_str += ' WHERE CaseID=%s'
            data.append(Id)

            self.open()
            self.stmt.execute(update_str, data)
            self.conn.commit()
            print('Record updated successfully.')
            self.view_allcases()
            return True
        else:
            print("Caseid Not Found IN Cases Table")
    except Exception as e:
        print(f"Error updating case details: {e}")
        return False  # Return False if the operation fails
```

O/P:

```
Enter incidentID :3
Enter IncidentType :
Enter VictimName: Srikar Pudi
Enter SuspectName : Chris Evans
Enter OfficerName : Rohit Shaw
Enter CaseDescription :
--Database Is Connected--
Record updated successfully.
--Database Is Connected--


_________________Records In Case Table_________________________
(1, 1, 'Robbery', 'Sai Karri', 'Robert Williams', 'Mark Johnson', 'Armed robbery at a convenience store.')
(2, 2, 'Homicide', 'Arun Dantu', 'Emily Davis', 'Sarah Brown', 'Fatal shooting incident.')
(6, 3, 'Theft', 'Srikar Pudi', 'Chris Evans', 'Rohit Shaw', 'Burglary at a residence.')
```

// Get a list of all cases

List<Case> getAllCases();

parameters- None

return type Collection of cases

```
def view_allcases(self):
    self.open()
    select_str = '''select * from Icase '''
    self.stmt.execute(select_str)
    recods = self.stmt.fetchall()
    print('')
    print('_________________Records In Case Table_____________________')
    for i in recods:
        print(i)
    self.close()
    return f'Case details fetched successfully'
```

O/P:

```
enter your choice9
--Database Is Connected--


_________________Records In Case Table_________________________
(1, 1, 'Robbery', 'Sai Karri', 'Robert Williams', 'Mark Johnson', 'Armed robbery at a convenience store.')
(2, 2, 'Homicide', 'Arun Dantu', 'Emily Davis', 'Sarah Brown', 'Fatal shooting incident.')
(6, 3, 'Theft', 'Srikar Pudi', 'Chris Evans', 'Rohit Shaw', 'Burglary at a residence.')
Connection Closed.
```

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.

```
import mysql.connector as connection
from util.PropertyUtil import PropertyUtil
class dbConnection():
    def _init_(self):
        pass
    def open(self):
        try:
            l = PropertyUtil.getPropertyString(self)
            self.conn = connection.connect(host=l[0], database=l[3], username=l[1],
password=l[2])
            if self.conn:
                print("--Database Is Connected--")
            self.stmt = self.conn.cursor()
        except Exception as e:
            print(e)

    def close(self):
```

```
        try:
            self.conn.close()
            print('Connection Closed.')
        except Exception as e:
            print(e)

d = dbConnection()
d.open()
d.close()
```

```
class PropertyUtil:

    def getPropertyString(self):
        host = 'localhost'
        username = 'root'
        password = 'root'
        database = 'crimereportingsystem'
        return host, username, password, database
```

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

```
from util.DBConnUtil import dbConnection
from entity.Incidents import  Incidents
from entity.ICase import  ICase
from Dao.Icase import ICase
from MyExceptions.InvalidNameError import InvalidNameError , StringCheck
from MyExceptions.IncidentNumberNotFoundException import IncidentNumberNotFoundException
class  ICrimeAnalysisService(dbConnection):
    def __init__(self):
        self.incident_id = " "
        self.incident_type = " "
        self.incident_date = " "
        self.location = " "
        self.description = " "
        self.status = " "
        self.victim_id = " "
        self.suspect_id = " "
        print(self.incident_id
,self.incident_type,self.incident_date,self.location,self.description
,self.status,self.victim_id,self.suspect_id )

    def create_incident(self):
        try:

            incident_id=int(input("enter incident id: "))
            self.incident id=incident id

            incident_type=input("enter incident type: ")
            self.incident_type=incident_type

            incident date=input("enter incident date in (yyyy-mm-dd) format: ")
            self.incident_date=incident_date
```

```
            location=input("enter location: ")
            self.location=location

            description=input("enter description about incident: ")
            self.description=description

            status=input("enter status: ")
            self.status=status

            victim_id=int(input("enter victim id: "))
            self.victim_id=victim_id

            suspect_id=int(input("enter suspect id:"))
            self.suspect_id=suspect_id


            # SQL query to insert a new incident
            sql_query = """
                        INSERT INTO Incidents (IncidentID, IncidentType, IncidentDate,
Location, Description, Status, victimid, suspectid)
                        VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
                        """
            # Parameters for the SQL query
            values = [(incident_id, incident_type, incident_date,
                       location, description, status, victim_id, suspect_id)]
            self.open()
            self.stmt.executemany(sql_query, values)
            self.conn.commit()
            print('Records Inserted Successfully..')
            self.close()
            return True  # Return True if the operation is successful
        except Exception as e:
            print(f"Error creating incident: {e}")
            return False  # Return False if the operation fails

    # Package: dao


    def update_incident_status(self):
        try:
            self.select()
            incidentid=int(input("enter incidentid: "))
            check_query = """ SELECT Incidentid from Incidents"""
            self.incidentid = incidentid
            self.open()
            ids = self.stmt.execute(check_query)
            recods = self.stmt.fetchall()
            lists = [i[0] for i in recods]
            if incidentid in lists:
                status=input("enter status to get updated: ")
                self.status=status

                # SQL query to update the status of an incident
                sql_query = """ UPDATE Incidents
                                SET Status = %s
                                WHERE IncidentID = %s
                            """
                data=[(status,incidentid)]
                self.open()
                self.stmt.executemany(sql_query,data )
                self.conn.commit()
                print('Records Updated Successfully..')
```

```
                self.select()
                self.close()
                return True  # Return True if the operation is successful
            else:
                raise IncidentNumberNotFoundException()
        except Exception as e:
            print(f"Error updating incident status: {e}")
            return False  # Return False if the operation fails

    def get_incidents_in_date_range(self, start_date, end_date):

            # SQL query to select incidents within the specified date range
            sql_query = """ SELECT * FROM Incidents
                            WHERE IncidentDate BETWEEN %s AND %s
                        """
            # Parameters for the SQL query
            values = (start_date, end_date)
            self.open()
            self.stmt.execute(sql_query,values)
            recods = self.stmt.fetchall()
            print('')
            print('_________________Records In Date Range ____________________')
            for i in recods:
                print(i)
            self.close()

            if not recods:
                print("No Incidents Found in the Given Dates")
            return f'Incidents details fetched successfully'

    def search_incidents(self, incident_type):
        try:
            check_query=""" SELECT IncidentType FROM Incidents"""
            self.open()
            self.stmt.execute(check_query)
            records=self.stmt.fetchall()
            checking=[i[0] for i in records]
            print(checking)
            self.close()
            if not isinstance(incident_type,str):
                print("invalid data type")
            incident_type=incident_type.lower()
            for incidenttype in checking:
                incidenttype=incidenttype.lower()
                if incident_type==incidenttype:
                    # SQL query to search for incidents based on the given criteria
                    sql_query = """
                                    SELECT *
                                    FROM Incidents
                                    WHERE IncidentType = %s
                                """
                    # Parameters for the SQL query
                    values = [(incident_type)]
                    self.open()
                    self.stmt.execute(sql_query, values)
                    recods = self.stmt.fetchall()
                    for i in recods:
                        print(i)
                    self.close()
                else:
                    print(f"No Incident found with Incident type: {incident_type}")

                return f'Incidents details fetched successfully'
```

```
        except Exception as e:
            print(f"An unexpected error occurred: {str(e)}")

    def generate_incident_report(self,incidentid):
        try:
            self.open()
            select_Incidents_str = '''
                                SELECT * FROM Reports
                                WHERE IncidentID = %s
                            '''
            self.stmt.execute(select_Incidents_str, (incidentid,))
            customer_data = self.stmt.fetchone()

            if not customer_data:
                    raise IncidentNumberNotFoundException()
            else:
                    print('\nIncident Report:')
                    print(f"ReportID: {customer_data[0]}")
                    print(f"IncidentID: {customer_data[1]}")
                    print(f"ReportingOfficer: {customer_data[2]}")
                    print(f"ReportDate: {customer_data[3]}")
                    print(f"ReportDetails: {customer_data[4]}")
                    print(f"Status: {customer_data[5]}")
                    # print(f"VictimID: {customer_data[6]}")
                    # print(f"SuspectID: {customer_data[7]}")

            self.close()
        except Exception as e:
            print(f"An unexpected error occurred: {str(e)}")

    def create_case(self):
        try:
            # Description = input("enter description about case: ")
            # self.Description=Description

            victim_id = int(input("enter victim id: "))
            self.victim_id = victim_id

            suspect_id = int(input("enter suspect id: "))
            self.suspect_id = suspect_id

            officer_id = int(input("enter officer id: "))
            self.officer_id = officer_id

            # incident_id=int(input("enter incident id: "))
            # self.incident_id=incident_id
            #
            # incident_type=input("enter incident type: ")
            # self.incident_type=incident_type
            #
            # VictimName=input("enter victim name: ")
            # self.VictimName=VictimName
            #
            # SuspectName=input("enter suspect name: ")
            # self.SuspectName=SuspectName
            #
            # OfficerName=input("enter officer name: ")
            # self.OfficerName=OfficerName
            #
            # CaseDescription=input("enter description about case: ")
            # self.CaseDescription=CaseDescription

            # SQL query to insert a new incident
```

```
            # sql_query = """
            #             INSERT INTO ICases (IncidentID, IncidentType, VictimName,
SuspectName, OfficerName, CaseDescription)
            #             VALUES (%s, %s, %s, %s, %s, %s)
            #             """
            query=  """ INSERT INTO ICase(IncidentID, IncidentType, VictimName,
SuspectName, OfficerName, CaseDescription)
                    SELECT
                        i.IncidentID,
                        i.IncidentType,
                        CONCAT(v.FirstName, ' ', v.LastName) AS VictimName,
                        CONCAT(s.FirstName, ' ', s.LastName) AS SuspectName,
                        CONCAT(o.FirstName, ' ', o.LastName) AS OfficerName,
                        i.description  as CaseDescription
                    FROM
                        Incidents i
                        LEFT JOIN Victims v ON i.VictimID = v.VictimID
                        LEFT JOIN Suspects s ON i.SuspectID = s.SuspectID
                        LEFT JOIN Officers o ON i.IncidentID = o.OfficerID
                    WHERE

                        i.VictimID = %s
                        AND i.SuspectID = %s
                        AND i.IncidentID = %s
                    """
                    # Parameters for the SQL query
            values = [(victim_id, suspect_id, officer_id)]
            self.open()
            self.stmt.executemany(query, values)
            self.conn.commit()
            print('Records Inserted Successfully..')
            self.close()
            return True  # Return True if the operation is successful
        except Exception as e:
            print(f"Error creating case: {e}")
            return False  # Return False if the operation

    def CaseDetails(self, Caseid):
        try:
            # if not isinstance(Caseid, int) or Caseid < 0:
            #     raise InvalidIDError()
            self.open()
            select_customer_str = '''
                SELECT * FROM ICase
                WHERE CaseID = %s
            '''
            self.stmt.execute(select_customer_str, (Caseid,))
            customer_data = self.stmt.fetchone()

            if not customer_data:
                print(f"No Case found with CaseID: {Caseid}")
            else:
                print('\nCase Details:')
                print(f"CaseID: {customer_data[0]}")
                print(f"IncidentID: {customer_data[1]}")
                print(f"IncidentType: {customer_data[2]}")
                print(f"VictimName: {customer_data[3]}")
                print(f"SuspectName: {customer_data[4]}")
                print(f"OfficerName: {customer_data[5]}")
                print(f"CaseDescription: {customer_data[6]}")
            self.close()
        except Exception as e:
            print(f"An unexpected error occurred: {str(e)}")
```

```
    def updateCaseDetails(self,caseid):
        try:
            self.view_allcases()
            check_query=""" SELECT CaseId from ICase"""
            self.open()
            ids=self.stmt.execute(check_query)
            recods = self.stmt.fetchall()
            lists=[i[0] for i in recods]
            print(lists)
            self.close()
            if caseid in lists:
                Id = caseid
                update_str = 'UPDATE ICase SET '
                data = []

                incidentid= input('Enter incidentID :')
                self.incidentid = incidentid
                update_str += 'incidentid=%s, '
                data.append(self.incidentid)

                IncidentType = input('Enter IncidentType :')
                if IncidentType:
                    self.IncidentType = IncidentType
                    update_str += 'IncidentType=%s, '
                    data.append(self.IncidentType)

                VictimName = input('Enter VictimName: ')
                if VictimName:
                    self.VictimName = VictimName
                    update_str += 'VictimName=%s, '
                    data.append(self.VictimName)

                SuspectName = input('Enter SuspectName : ')
                if SuspectName:
                    self.SuspectName = SuspectName
                    update_str += 'SuspectName=%s, '
                    data.append(self.SuspectName)

                OfficerName = input('Enter OfficerName : ')
                if OfficerName:
                    self.OfficerName = OfficerName
                    update_str += 'OfficerName=%s, '
                    data.append(self.OfficerName)

                CaseDescription = input('Enter CaseDescription : ')
                if CaseDescription:
                    self.CaseDescription =CaseDescription
                    update_str += 'CaseDescription=%s, '
                    data.append(self.CaseDescription)

                update_str = update_str.rstrip(', ')
                update_str += ' WHERE CaseID=%s'
                data.append(Id)

                self.open()
                self.stmt.execute(update_str, data)
                self.conn.commit()
                print('Record updated successfully.')
                self.view_allcases()
                return True
            else:
                print("Caseid Not Found IN Cases Table")
```

```
        except Exception as e:
            print(f"Error updating case details: {e}")
            return False  # Return False if the operation fails


    def view_allcases(self):
        self.open()
        select_str = '''select * from Icase '''
        self.stmt.execute(select_str)
        recods = self.stmt.fetchall()
        print('')
        print('__________________Records In Case Table_____________________')
        for i in recods:
            print(i)
        self.close()
        return f'Case details fetched successfully'
    def select(self):
        self.open()
        select_str = '''select * from Incidents '''
        self.stmt.execute(select_str)
        recods = self.stmt.fetchall()
        print('')
        print('__________________Records In Incidents Table_____________________')
        for i in recods:
            print(i)
        self.close()
        return f'Incidents details fetched successfully'
```

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 id.

```
number which doesn't exist in db
class  IncidentNumberNotFoundException(Exception):
    def __init__(self, message="Invalid Incident number "):
        self.message = message
        super().__init__(self.message)
```

```
_________________Records In Incidents Table_____________________
(1, 'Robbery', datetime.date(2023, 1, 10), '123 Main St, USA', 'Armed robbery at a convenience store.', 'open', 1, 1)
(2, 'Homicide', datetime.date(2023, 2, 15), '456 Elm St, USA', 'Fatal shooting incident.', 'open', 2, 2)
(3, 'Theft', datetime.date(2023, 3, 20), '789 Oak St, USA', 'Burglary at a residence.', 'Under Investigation', 3, 3)
(4, 'Murder', datetime.date(2023, 4, 9), '387 Ma St,Usa', 'Pre-Planned Murder', 'open', 4, 4)
Connection Closed.
enter incidentid: 5
--Database Is Connected--
Error updating incident status: Invalid Incident number
```

9. Main Method

Create class named MainModule with main method in main package.

Trigger all the methods in service implementation class.

```
from Dao.ICrimeAnalysisService import  ICrimeAnalysisService
from Dao.Victims import  Victims
from Dao.Suspect import Suspect
from Dao.incidents import incidents
from Dao.Icase import ICase
from MyExceptions.IncidentNumberNotFoundException import IncidentNumberNotFoundException
condition = True
create = True
try:

    while condition:
        incident2= ICrimeAnalysisService()
        incident1=incidents()
        victim1=Victims()
        suspect1=Suspect()
        icase1 = ICase()
        print("select table")

print("1.CrimeAnalysisService\n2.Incidents\n3.Victims\n4.Suspects\n5.Cases\n6.Exit")
        choice = int(input("enter your choice"))
        if choice == 1:
            while True:
                print("1.Create New Incident          \t2.Update status of
Incident\n3.Get Incidents in date range\t"
                      "4.Search For Incident\n5.Incident Reports      \t\t6.Create New
Case\n7.Get Case Details by ID\t"
                      "  \t8.Update Case Details\n9.List All Cases      \t\t\t10.View
Incident Details\n11.Exit")
                choice = int(input("enter your choice"))
                if choice == 1:
                    incident2.create_incident()
                elif choice == 2:
                    incident2.update_incident_status()
                elif choice == 3:
                    start_date=input("enter startdate in yyyy-mm-dd: ")
                    end_date = input("enter enddate in yyyy-mm-dd: ")
                    incident2.get_incidents_in_date_range(start_date,end_date)
                elif choice == 4:
                    Incident_type=input("enter Incident type: ")
                    incident2.search_incidents(Incident_type)
                elif choice == 5:
                    incidentid=input("enter incident id: ")
                    incident2.generate_incident_report(incidentid)
                elif choice == 6:
                    incident2.create_case()
                elif choice == 7:
                    Caseid=int(input("enter Caseid: "))
                    incident2.CaseDetails(Caseid)
                elif choice == 8:
                    caseid = int(input("enter caseid: "))
                    incident2.updateCaseDetails(caseid)
                elif choice == 9:
                    incident2.view_allcases()
                elif choice == 10:
                    incident2.select()
                else:
                    break
        elif choice == 2:
            while True:
                print("1.View Incident Details\n2.Remove Incident\n3.Exit")
                choice = int(input("enter your choice"))
                if choice == 1:
                    incident1.select()
```

```
                elif choice == 2:
                    incident1.delete()
                else:
                    break
        elif choice == 3:
            while True:
                print("1.Add Victim\n2.View Victims Details\n3.Remove Victim\n4.Exit")
                choice = int(input("enter your choice"))
                if choice == 1:
                    victim1.add_victim()
                elif choice == 2:
                    victim1.select()
                elif choice == 3:
                    victim1.delete()
                else:
                    break
        elif choice == 4:
            while True:
                print("1.Add Suspect\n2.View Suspects Details\n3.Remove Suspect\n4.Exit")
                choice = int(input("enter your choice"))
                if choice == 1:
                    suspect1.add_suspects()
                elif choice == 2:
                    suspect1.select1()
                elif choice == 3:
                    suspect1.delete()
                else:
                    break
        elif choice == 5:
            while True:
                print("1.View Case Details\n2.Remove Case\n3.Exit")
                choice = int(input("enter your choice"))
                if choice == 1:
                    icase1.select_case()
                elif choice == 2:
                    icase1.delete_case()
                else:
                    break
        else:
            break
except IncidentNumberNotFoundException as e:
    print(e)
except Exception as e:
    print(e)
```

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?

```
import unittest
from Dao import ICrimeAnalysisService
from entity import Incidents
```

```python
import pytest

class TestCrimeAnalysisService(unittest.TestCase):
    def setUp(self):
        # Initialize the service
        self.service = ICrimeAnalysisService.ICrimeAnalysisService()
        # Sample incident data
        self.incident_data = {
            'incident_id':4,
            'incident_type': 'Robbery',
            'incident_date': '2023-01-10',
            'location': '123 Main St, City, Usa',
            'description': 'Armed robbery at a convenience store.',
            'status': 'Open',
            'victim_id': 4,
            'suspect_id': 4
        }

    def test_create_incident(self):
        # Test createIncident method
        created = self.service.create_incident(**self.incident_data)
        self.assertTrue(created)


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

TestCase Validation:

```
C:\Users\Megha Syam\PycharmProjects\pythonProject5>pytest -v
============================================= test session starts =============================================
platform win32 -- Python 3.10.7, pytest-8.2.0, pluggy-1.5.0 -- C:\Users\Megha Syam\AppData\Local\Programs\Python\Python3
10\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\Megha Syam\PycharmProjects\pythonProject5
collected 1 item

Test/my_test.py::TestCrimeAnalysisService::test_create_incident PASSED                                [100%]

============================================== 1 passed in 0.47s ==============================================
```

2. Incident Status Update:

- Does the updateIncidentStatus method effectively update the status of an incident?
- Does it handle invalid status updates appropriately?

```python
import unittest
from Dao import ICrimeAnalysisService
from entity import Incidents
import pytest

class TestCrimeAnalysisService(unittest.TestCase):
    def setUp(self):
        # Initialize the service
        self.service = ICrimeAnalysisService.ICrimeAnalysisService()


    def test_update_incident_status(self):
        self.incident_data1 = {
            'incident_id': 3,
            'incident_type': 'Robbery',
            'incident_date': '2023-01-10',
```

```
            'location': '123 Main St, City, Usa',
            'description': 'Armed robbery at a convenience store.',
            'status': 'Open',
            'victim_id': 3,
            'suspect_id': 3
        }
        # Create a test incident
        incident = Incidents.Incidents(**self.incident_data1)
        # self.service.create_incident(incident)

        # Get the incident ID from the created incident
        incident_id = incident.incident_id

        # Test updateIncidentStatus method
        new_status = 'Closed'
        updated = self.service.update_incident_status(incident_id, new_status)

        # Retrieve the updated incident
        updated_incident = self.service.get_incident_details(incident_id)
        print(updated_incident)

        # Verify that the status has been updated correctly
        assert updated_incident[5]== new_status
        # self.assertTrue(updated)


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

```
C:\Users\Megha Syam\PycharmProjects\pythonProject5>pytest -v
============================================= test session starts =============================================
platform win32 -- Python 3.10.7, pytest-8.2.0, pluggy-1.5.0 -- C:\Users\Megha Syam\AppData\Local\Programs\Python\Python3
10\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\Megha Syam\PycharmProjects\pythonProject5
collected 1 item

Test/my_test.py::TestCrimeAnalysisService::test_update_incident_status PASSED                        [100%]

============================================== 1 passed in 0.57s ==============================================
```

```
C:\Users\Megha Syam\PycharmProjects\pythonProject5>pytest -v
============================================= test session starts =============================================
platform win32 -- Python 3.10.7, pytest-8.2.0, pluggy-1.5.0 -- C:\Users\Megha Syam\AppData\Local\Programs\Python\Python3
10\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\Megha Syam\PycharmProjects\pythonProject5
collected 2 items

Test/my_test.py::TestCrimeAnalysisService::test_create_incident PASSED                               [ 50%]
Test/my_test.py::TestCrimeAnalysisService::test_update_incident_status PASSED                        [100%]

============================================== 2 passed in 0.61s ==============================================
```