

DBMS PROJECT – CRIME DATABASE –

FINAL REPORT

SUBMITTED BY:

GROUP 3:

DEVANANDA A : ROLL NO 33

LIZONA LOY PARAYIL : ROLL NO 47

KRISHNAPRIYA R : ROLL NO 46

M S GAURI SANKAR : ROLL NO 48

NIDHIN K BIJU : ROLL NO 55

SUBMITTED TO:

Mrs.JOSNA V R

# INDEX

|  |  |  |
| --- | --- | --- |
| Sl no. | Content | Page no. |
| 1. | Functional  Requirements | 3 - 4 |
| 2. | E R Diagram | 5 |
| 3. | SQL Query | 6 - 7 |
| 4. | Procedure | 8 - 10 |
| 5. | Trigger | 11 - 12 |
| 6. | Tables | 13 - 15 |
| 7. | UI | 16 - 17 |
| 8. | Conclusion | 18 |

**GITHUB LINK:** [**https://github.com/Devananda-A/DB MS.git**](https://github.com/Devananda-A/DBMS.git)

**FUNCTIONAL REQUIREMENTS:**

## Scope

The system supports registering criminal profiles in the database. It allows updating criminal profiles whenever new information is available. Crimes can be recorded and linked directly to the criminals involved. Officers can be assigned to cases for proper investigation. Victim information can be recorded and managed. Witness statements can also be maintained in the system. The progress of each case is tracked from opening to closure. The system monitors criminal status, including whether the criminal is Arrested, Wanted, or Released.

## 1. Criminal Management

The system allows users to add criminal records into the database. Each criminal record includes a unique Criminal\_ID for identification. The full name of the criminal is stored in the record. Age and gender of the criminal are maintained. The address of the criminal is recorded to help locate and contact them. The status of the criminal, which can be Arrested, Wanted, or Released, is tracked. A photograph of the criminal can be uploaded and associated with the record.

## 2. Crime Management

The system allows registering a new crime entry in the database. Each crime record has a unique Crime\_ID to distinguish it from others. The type of crime, such as Theft, Assault, or Homicide, is stored. The date and time when the crime occurred are recorded. The location of the crime is maintained for investigation purposes. The severity level of the crime is indicated. A detailed description of the crime is stored to provide context for investigators.

## 3. Case Management

The system allows opening new cases and closing existing cases. Each case is assigned a unique Case\_ID. Related Crime\_IDs are linked to the case to indicate which crimes are under investigation. Criminal\_IDs involved in the case are also associated with the case. An officer is assigned to the case using their Officer\_ID. The status of the case can be Open, Closed, or On Hold. The start date of the case is recorded when it is opened. The end date of the case is recorded when it is closed.

## 4. Officer Management

The system maintains records for each officer. Each officer has a unique Officer\_ID. The name of the officer is stored. The rank of the officer is recorded. Contact information for the officer is maintained. The police station assigned to the officer is recorded using Station\_ID.

## 5. Station Management

The system stores information for each police station. Each station has a unique Station\_ID. The name of the station is recorded. The location of the station is stored. The contact number for the station is maintained.

## 6. Victim Management

The system allows adding records of victims involved in crimes. Each victim has a unique Victim\_ID. The full name of the victim is stored. The age and gender of the victim are recorded. The address of the victim is maintained. Contact information for the victim is stored. Victims are linked to the relevant cases in the system.

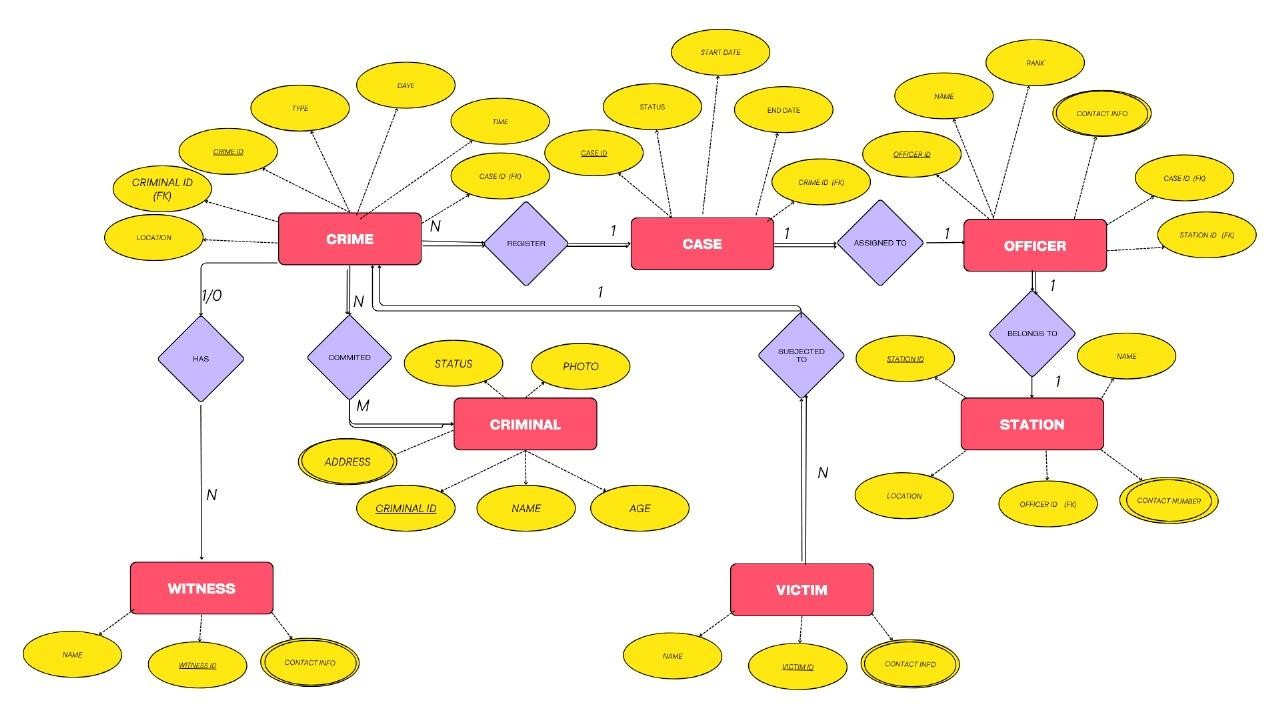
## 7. Witness Management

The system allows adding records for witnesses. Each witness has a unique Witness\_ID. The full name of the witness is stored. The witness’s statement is recorded. Contact information for the witness is maintained. Witnesses are linked to the relevant cases in which they provided testimony.

## 8. Suspect Management

The system allows adding records for suspects. Each suspect has a unique AADHAAR number for identification. The full name of the suspect is recorded. The age and gender of the suspect are maintained. The address of the suspect is stored. The status of the suspect, which can be Guilty or Not Guilty, is tracked. A photograph of the suspect can be added to the record. Suspects are linked to the relevant crimes and cases.

**ER DIAGRAM:**



**SQL QUERY:**

(PROCEDURES,TRIGGERS,)

CREATE DATABASE IF NOT EXISTS crimedatabase;

USE crimedatabase;

CREATE TABLE IF NOT EXISTS Station ( station\_id INT PRIMARY KEY AUTO\_INCREMENT, station\_name VARCHAR(100), location VARCHAR(150), phone VARCHAR(15)

);

CREATE TABLE IF NOT EXISTS Officer ( officer\_id INT PRIMARY KEY AUTO\_INCREMENT, officer\_name VARCHAR(100), officer\_rank VARCHAR(50), station\_id INT, contact\_no VARCHAR(15),

FOREIGN KEY (station\_id) REFERENCES Station(station\_id)

);

CREATE TABLE IF NOT EXISTS CaseFile ( case\_id INT PRIMARY KEY AUTO\_INCREMENT, case\_title VARCHAR(150), case\_type VARCHAR(100), date\_reported DATE, status VARCHAR(30), officer\_id INT,

FOREIGN KEY (officer\_id) REFERENCES Officer(officer\_id)

);

CREATE TABLE IF NOT EXISTS Crime ( crime\_id INT PRIMARY KEY AUTO\_INCREMENT,

case\_id INT, crime\_description TEXT, crime\_date DATE, crime\_location VARCHAR(150),

FOREIGN KEY (case\_id) REFERENCES CaseFile(case\_id)

);

CREATE TABLE IF NOT EXISTS Criminal ( criminal\_id INT PRIMARY KEY AUTO\_INCREMENT, name VARCHAR(100), gender VARCHAR(10), age INT, address VARCHAR(150),

crime\_id INT,

FOREIGN KEY (crime\_id) REFERENCES Crime(crime\_id)

);

CREATE TABLE IF NOT EXISTS Victim ( victim\_id INT PRIMARY KEY AUTO\_INCREMENT, name VARCHAR(100), age INT, contact\_no VARCHAR(15), address VARCHAR(150), case\_id INT,

FOREIGN KEY (case\_id) REFERENCES CaseFile(case\_id)

);

CREATE TABLE IF NOT EXISTS Witness ( witness\_id INT PRIMARY KEY AUTO\_INCREMENT, name VARCHAR(100), contact\_no VARCHAR(15), statement TEXT, case\_id INT,

FOREIGN KEY (case\_id) REFERENCES CaseFile(case\_id)

);

CREATE TABLE IF NOT EXISTS Station\_Log ( log\_id INT PRIMARY KEY AUTO\_INCREMENT,

officer\_id INT, action VARCHAR(50), log\_time DATETIME

);

CREATE TABLE IF NOT EXISTS Case\_Log ( log\_id INT PRIMARY KEY AUTO\_INCREMENT,

case\_id INT, old\_status VARCHAR(30), new\_status VARCHAR(30), updated\_on DATETIME

);

CREATE TABLE IF NOT EXISTS Criminal\_Log ( log\_id INT PRIMARY KEY AUTO\_INCREMENT,

criminal\_id INT,

log\_message VARCHAR(255), log\_time DATETIME

);

**PROCEDURE:**

DELIMITER //

CREATE PROCEDURE AddCaseWithOfficer(

IN p\_title VARCHAR(150),

IN p\_type VARCHAR(100),

IN p\_officer INT

)

BEGIN

INSERT INTO CaseFile(case\_title, case\_type, date\_reported, status, officer\_id) VALUES(p\_title, p\_type, CURDATE(), 'Open', p\_officer);

SELECT \* FROM CaseFile WHERE officer\_id = p\_officer ORDER BY case\_id DESC LIMIT 3;

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE RegisterCriminal(

IN p\_name VARCHAR(100),

IN p\_gender VARCHAR(10),

IN p\_age INT,

IN p\_address VARCHAR(150),

IN p\_crime\_id INT

)

BEGIN

INSERT INTO Criminal(name, gender, age, address, crime\_id)

VALUES(p\_name, p\_gender, p\_age, p\_address, p\_crime\_id);

SELECT c.crime\_id, c.crime\_description, cf.case\_title, cf.status

FROM Crime c

JOIN CaseFile cf ON cf.case\_id = c.case\_id

WHERE c.crime\_id = p\_crime\_id;

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE AddVictim(

IN p\_name VARCHAR(100),

IN p\_age INT,

IN p\_contact VARCHAR(15),

IN p\_address VARCHAR(150),

IN p\_case INT

)

BEGIN

INSERT INTO Victim(name, age, contact\_no, address, case\_id)

VALUES(p\_name, p\_age, p\_contact, p\_address, p\_case);

SELECT name, contact\_no FROM Victim WHERE case\_id = p\_case;

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE CloseCase(IN p\_case INT)

BEGIN

UPDATE CaseFile SET status = 'Closed' WHERE case\_id = p\_case;

SELECT cf.case\_id, cf.case\_title, cf.status, o.officer\_name

FROM CaseFile cf

JOIN Officer o ON cf.officer\_id = o.officer\_id

WHERE cf.case\_id = p\_case;

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE GetOfficerCases(IN p\_officer\_id INT)

BEGIN

SELECT cf.case\_id, cf.case\_title, cf.case\_type, cf.status, cf.date\_reported

FROM CaseFile cf

WHERE cf.officer\_id = p\_officer\_id

ORDER BY cf.date\_reported DESC;

END //

CREATE PROCEDURE UpdateCaseStatus(IN p\_case\_id INT, IN p\_new\_status VARCHAR(30)) BEGIN

UPDATE CaseFile

SET status = p\_new\_status

WHERE case\_id = p\_case\_id;

SELECT case\_id, case\_title, status

FROM CaseFile

WHERE case\_id = p\_case\_id;

END //

CREATE PROCEDURE DeleteCriminalRecord(IN p\_criminal\_id INT)

BEGIN

DECLARE v\_case\_status VARCHAR(30);

SELECT cf.status INTO v\_case\_status

FROM Criminal cr

JOIN Crime c ON cr.crime\_id = c.crime\_id

JOIN CaseFile cf ON c.case\_id = cf.case\_id

WHERE cr.criminal\_id = p\_criminal\_id;

IF v\_case\_status = 'Closed' THEN

DELETE FROM Criminal WHERE criminal\_id = p\_criminal\_id;

SELECT CONCAT('Criminal ID ', p\_criminal\_id, ' deleted successfully') AS Message; ELSE

SELECT 'Cannot delete criminal record until case is closed' AS Message;

END IF;

END //

CREATE PROCEDURE SearchCrimeByDate(IN p\_start DATE, IN p\_end DATE)

BEGIN

SELECT c.crime\_id, c.crime\_description, c.crime\_date, cf.case\_title, o.officer\_name

FROM Crime c

JOIN CaseFile cf ON c.case\_id = cf.case\_id JOIN Officer o ON cf.officer\_id = o.officer\_id

WHERE c.crime\_date BETWEEN p\_start AND p\_end

ORDER BY c.crime\_date;

END //

CREATE PROCEDURE GetCriminalHistory(IN p\_criminal\_id INT)

BEGIN

SELECT cr.name AS Criminal\_Name, cr.gender, cr.age, cr.address, c.crime\_description, cf.case\_title, cf.status, o.officer\_name

FROM Criminal cr

JOIN Crime c ON cr.crime\_id = c.crime\_id

JOIN CaseFile cf ON c.case\_id = cf.case\_id JOIN Officer o ON cf.officer\_id = o.officer\_id

WHERE cr.criminal\_id = p\_criminal\_id;

END //

CREATE PROCEDURE VictimReport()

BEGIN

SELECT v.victim\_id, v.name AS Victim\_Name, v.age, v.contact\_no, cf.case\_title, cf.status FROM Victim v

JOIN CaseFile cf ON v.case\_id = cf.case\_id

ORDER BY cf.case\_id;

END //

**TRIGGER:**

DELIMITER //

CREATE TRIGGER AfterOfficerInsert

AFTER INSERT ON Officer

FOR EACH ROW

BEGIN

INSERT INTO Station\_Log(officer\_id, action, log\_time)

VALUES(NEW.officer\_id, 'New Officer Added', NOW());

END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER AfterCaseUpdate

AFTER UPDATE ON CaseFile

FOR EACH ROW

BEGIN

IF OLD.status <> NEW.status THEN

INSERT INTO Case\_Log(case\_id, old\_status, new\_status, updated\_on)

VALUES(NEW.case\_id, OLD.status, NEW.status, NOW());

END IF;

END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER AfterCriminalInsert

AFTER INSERT ON Criminal

FOR EACH ROW

BEGIN

INSERT INTO Criminal\_Log(criminal\_id, log\_message, log\_time)

VALUES(NEW.criminal\_id, CONCAT('Criminal Added: ', NEW.name), NOW());

END //

DELIMITER ;

CREATE TRIGGER BeforeCriminalDelete

BEFORE DELETE ON Criminal

FOR EACH ROW

BEGIN

DECLARE v\_status VARCHAR(30);

SELECT cf.status INTO v\_status

FROM Crime c

JOIN CaseFile cf ON c.case\_id = cf.case\_id

WHERE c.crime\_id = OLD.crime\_id;

IF v\_status <> 'Closed' THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Cannot delete criminal record until case is closed.';

END IF;

END //

CREATE TRIGGER AfterCaseInsert

AFTER INSERT ON CaseFile

FOR EACH ROW

BEGIN

INSERT INTO Case\_Log(case\_id, old\_status, new\_status, updated\_on)

VALUES(NEW.case\_id, NULL, NEW.status, NOW());

END //

CREATE TRIGGER AfterVictimInsert

AFTER INSERT ON Victim

FOR EACH ROW

BEGIN

INSERT INTO Case\_Log(case\_id, old\_status, new\_status, updated\_on)

VALUES(NEW.case\_id, 'Victim Added', 'Victim Added', NOW());

END //

CREATE TRIGGER BeforeCaseUpdate

BEFORE UPDATE ON CaseFile

FOR EACH ROW

BEGIN

IF OLD.status = 'Closed' AND NEW.status = 'Open' THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Cannot reopen a closed case.';

END IF;

END //

DELIMITER ;

10 SQL QUERIES:

**1. CREATE DATABASE IF NOT EXISTS CrimeDatabase;**

**USE CrimeDatabase;**

**CREATE TABLE Officer (**

**officer\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**officer\_name VARCHAR(100),**

**officer\_rank VARCHAR(50),**

**station\_id INT,**

**contact\_no VARCHAR(15)**

**);**

2.INSERT INTO Officer (officer\_name, officer\_rank, station\_id, contact\_no)

VALUES ('Arun Kumar', 'Inspector', 101, '9876543210');

3.ALTER TABLE Officer

ADD email VARCHAR(100);

4.SELECT o.officer\_name, c.case\_title, c.status

FROM Officer o

JOIN CaseFile c ON o.officer\_id = c.officer\_id;

5. UPDATE CaseFile

SET status = 'Closed'

WHERE case\_id = 1;

6. DELETE FROM Criminal

WHERE criminal\_id = 1;

7. DROP TABLE IF EXISTS CaseFile;

8. SELECT officer\_id, COUNT(case\_id) AS total\_cases

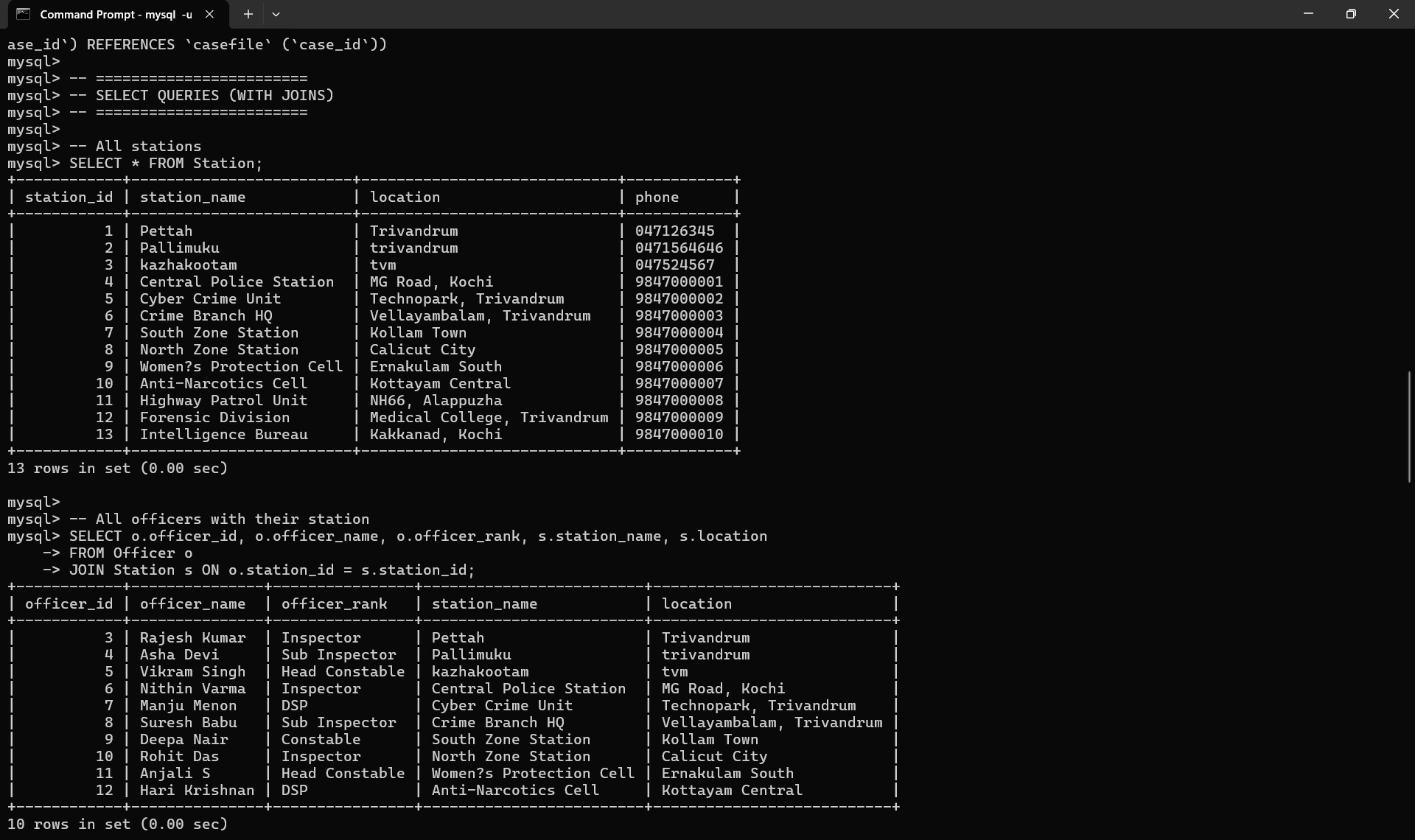
FROM CaseFile

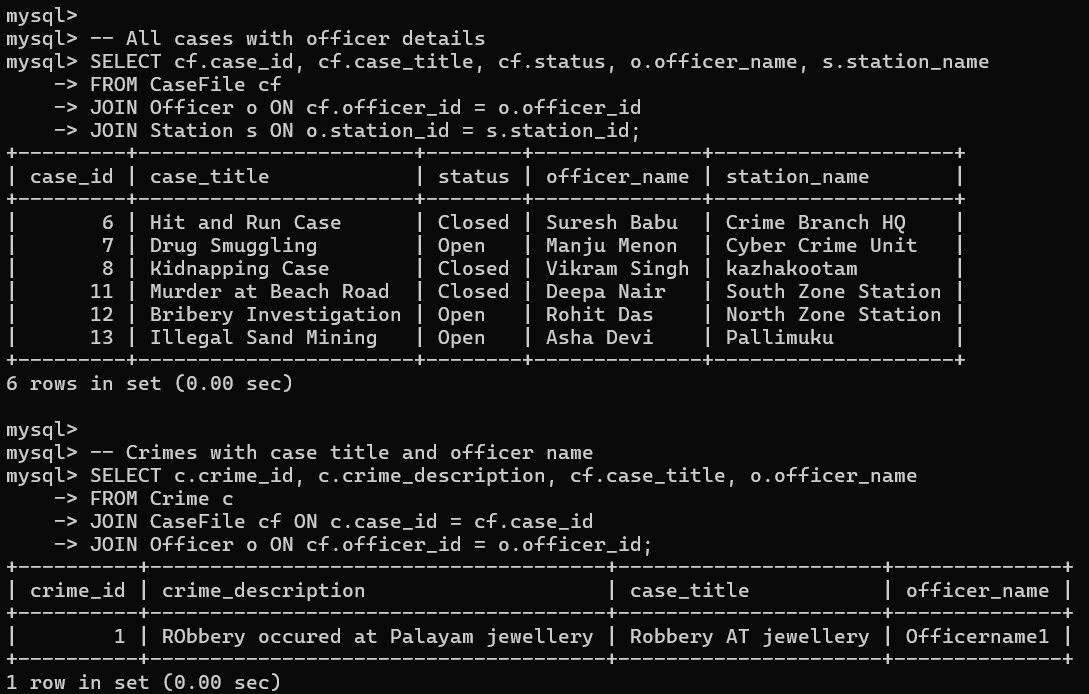
GROUP BY officer\_id;

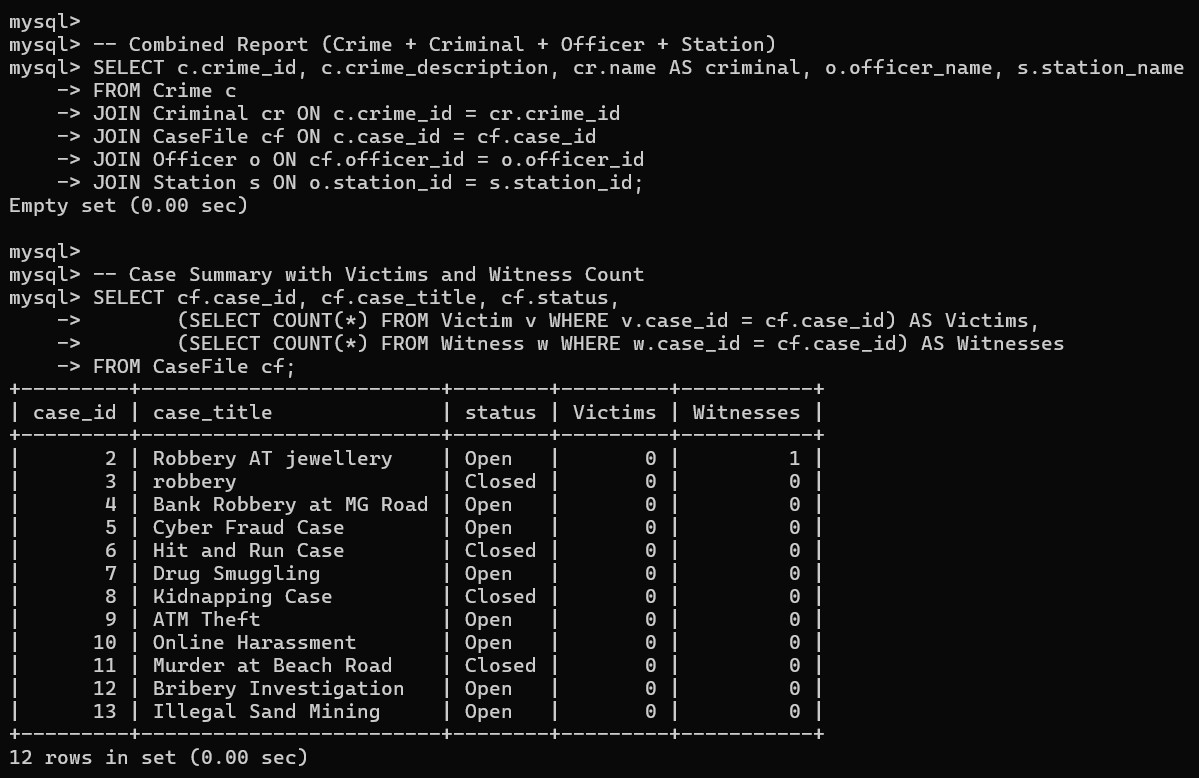
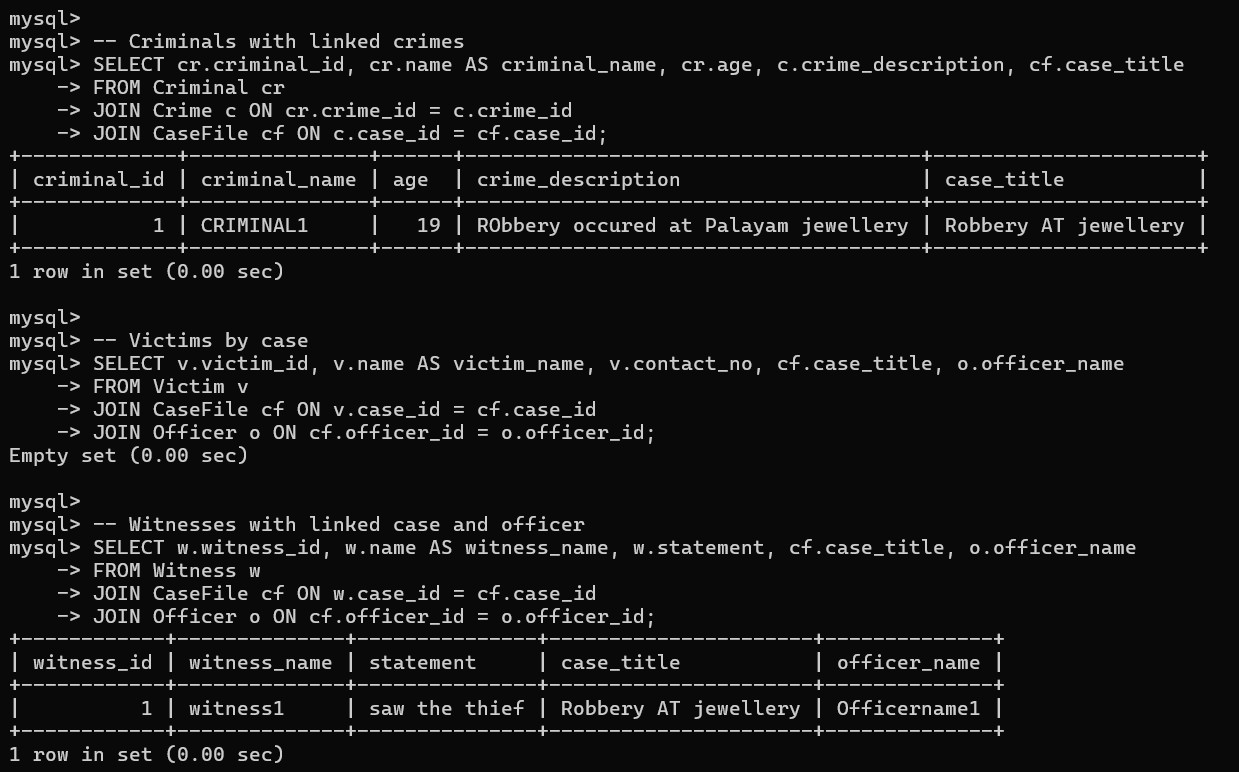
9. TRUNCATE TABLE CaseFile;

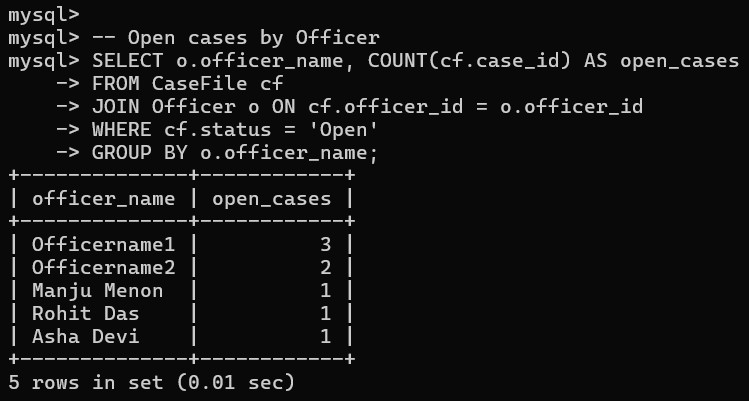
10. RENAME TABLE CaseFile TO Cases;

**TABLES:**

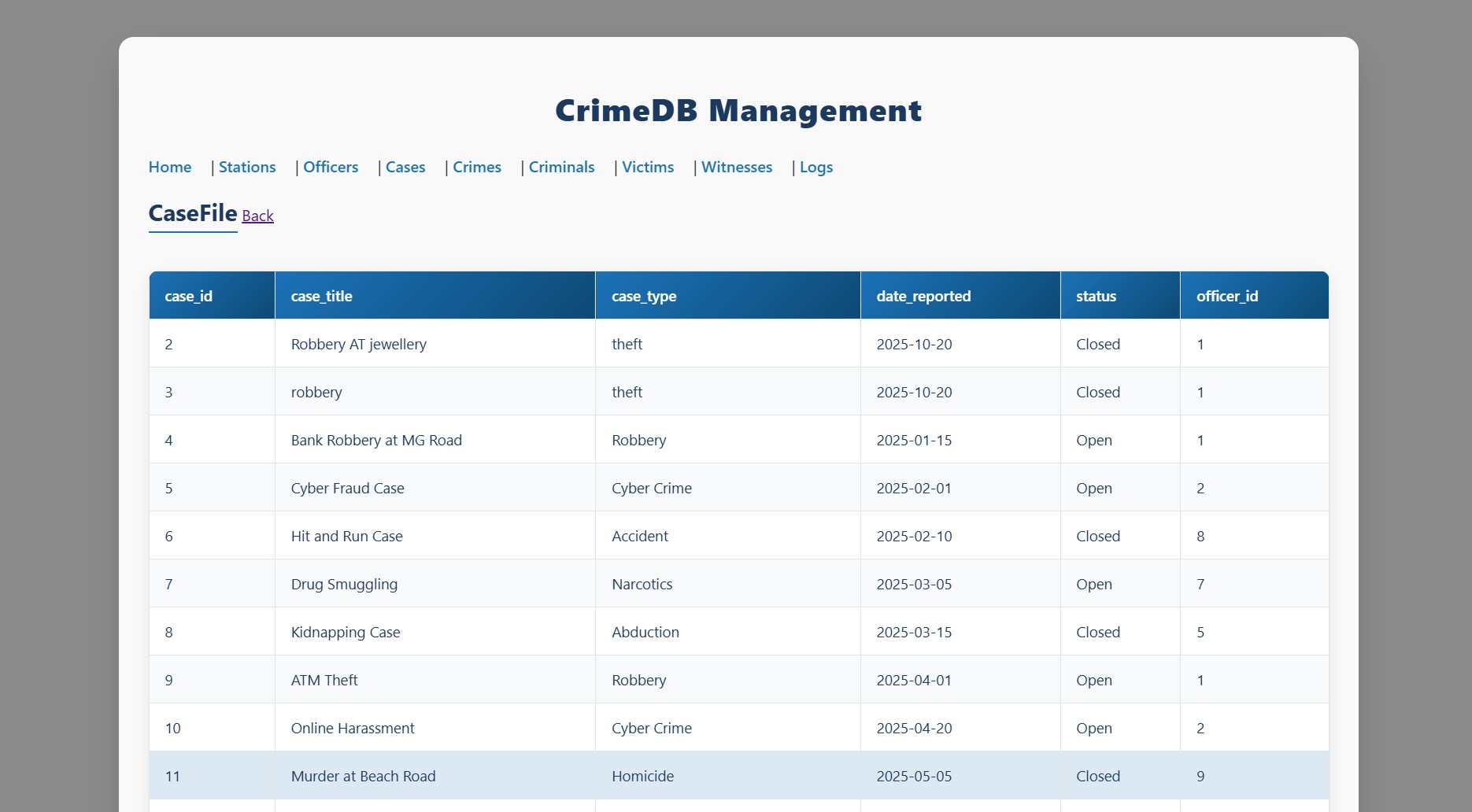
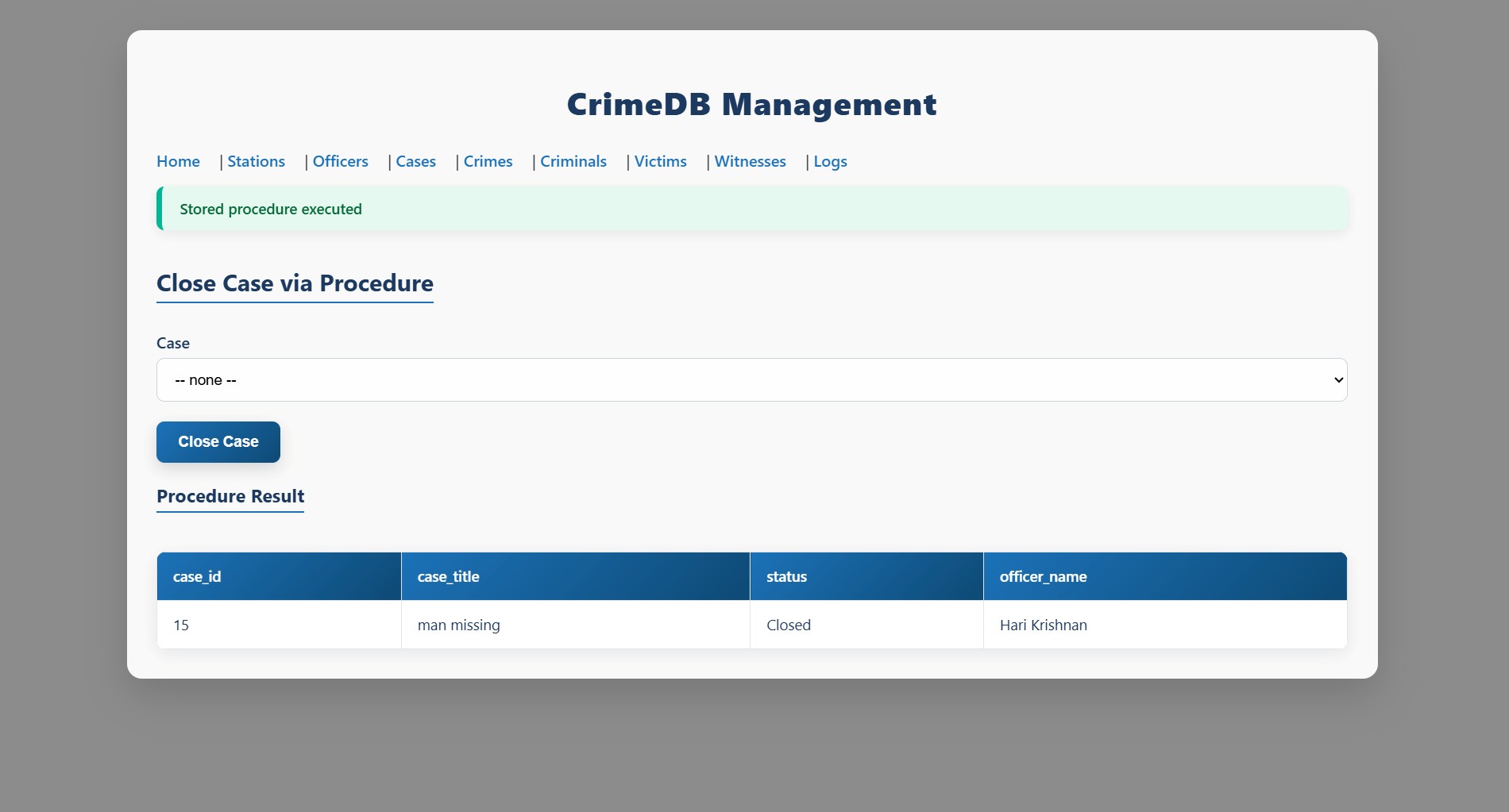
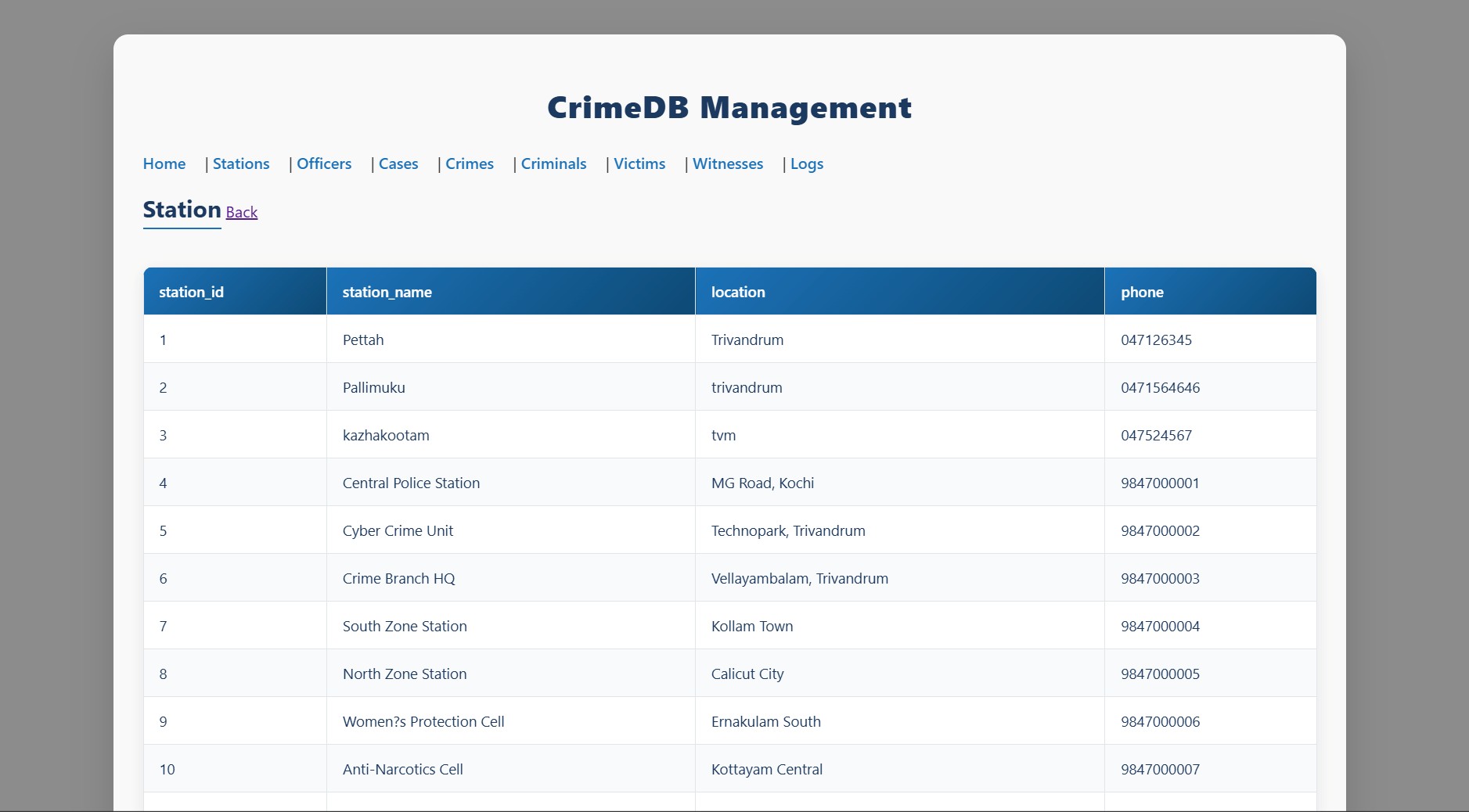
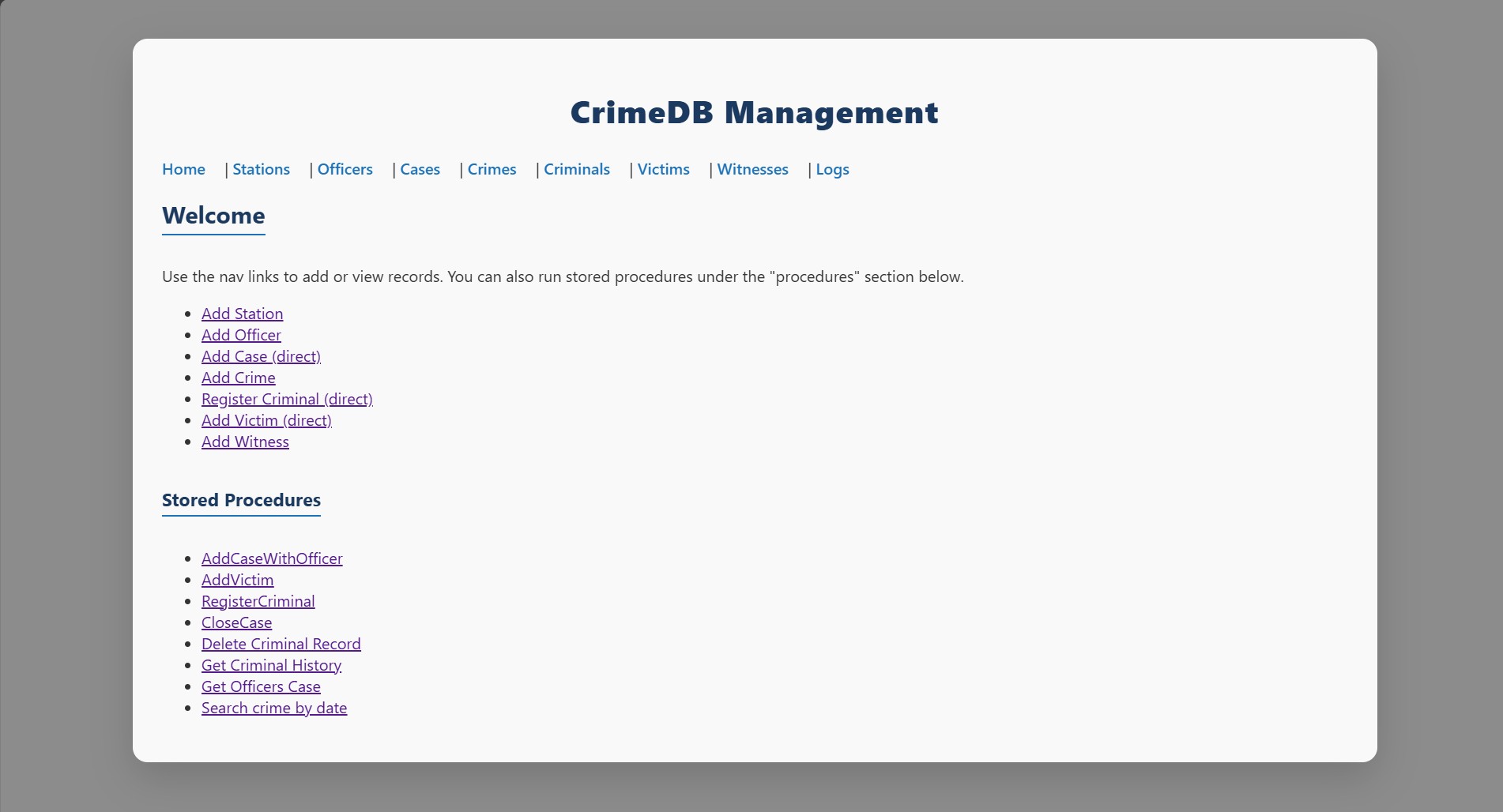








UI:



**CONCLUSION:**

The Crime Database Management System is an efficient and structured solution for storing and managing criminal, case, and officer information. It ensures data consistency, accuracy, and security through the use of SQL queries, stored procedures, and triggers. The system automates key operations like case updates and report generation, reducing manual effort and errors. By applying proper database design and relational modeling, it supports quick data retrieval and reliable performance. Overall, it serves as a practical and scalable tool that enhances crime tracking, transparency, and decision-making for law enforcement agencies.