



School of Computer Science, UPES, Dehradun.

A

Project File

on

DBMS

B.TECH. - III Semester

**AUG. - 2025.**

**Submitted to:**

Kalluri Shareef Babu

**Submitted by:**

Name: Ansh Goyal & Kartik Jindal

Batch: 29

# INTRODUCTION

Crime Data Analysis is an essential part of modern policing and investigation.

Law enforcement agencies collect large amounts of data related to crime incidents, locations, suspects, officers, and case progress.

Managing this data manually leads to problems such as:

- Delayed access to case records
- Difficulty in updating or retrieving information
- Poor accuracy in tracking crime patterns
- Risk of data loss

To solve these issues, a structured **Database Management System (DBMS)** is required.

This project, **Crime Data Analysis System**, provides a centralized platform where crime-related information can be stored, updated, and analyzed efficiently using MySQL and a Flask-based graphical interface.

# OBJECTIVES

The main objectives of the Crime Data Analysis system are:

1. To design a relational database structure for storing crime-related data.
2. To implement CRUD (Create, Read, Update, Delete) operations using SQL.
3. To use database concepts like Primary Keys, Foreign Keys, Joins, Views, and Triggers.
4. To build a simple, user-friendly web-based GUI using Flask, HTML, CSS.

5. To help users view, manage, and analyze crime cases through an organized interface.

## **SYSTEM REQUIREMENTS**

### **Software Requirements**

- Python 3.x
- Flask Framework
- MySQL Server / MySQL Workbench
- VS Code Editor
- mysql-connector-python

### **Hardware Requirements**

- 4GB RAM
- Dual-core processor
- 500MB free storage

## **SYSTEM ARCHITECTURE**

The system follows a **3-tier architecture**:

### **1. Frontend (Presentation Layer)**

- Built using HTML, CSS, Bootstrap
- Includes interactive forms and tables
- User can add, view, and manage cases

### **2. Backend (Application Layer)**

- Implemented using Python Flask

- Handles routing, request processing, form data
- Communicates with MySQL database

### **3. Database Layer**

- MySQL relational database
- Stores cases, crime types, officers, and locations
- Enforces constraints and manages data integrity

## **DATABASE DESIGN**

The Crime Data Analysis System uses a normalized relational database.

Main tables include:

---

### **(a) crime\_case**

Stores primary details of each case.

<b>Column Name</b>	<b>Description</b>
--------------------	--------------------

case_id	Auto-increment unique case number
case_code	Case reference code
case_date	Incident date
type_id	Type of crime (FK)
location_id	Location details (FK)
officer_id	Officer in charge (FK)
description	Brief summary
status	Open / Closed

---

**(b) location**

| area | city | state |

Stores the place where crime occurred.

---

**(c) officer**

| officer\_name |

Stores officer information entered manually.

---

**(d) crime\_type**

Stores types of crimes (e.g., Theft, Robbery, Assault).

## NORMALIZATION APPLIED

**First Normal Form (1NF)**

- All values are atomic.
- No repeating groups.
- Each field contains only one piece of information.

**Second Normal Form (2NF)**

- Table has a single-column primary key.
- No partial dependencies.

**Third Normal Form (3NF)**

- No transitive dependencies.
- Non-key attributes depend only on primary key.

→ This ensures **faster queries, no redundancy**, and **clean relational structure**.

## SQL QUERIES USED

Below are SQL statements actually used in the project:

---

### 1. JOIN Query (Case display in GUI)

```
SELECT c.case_id, c.case_code, c.case_date, c.status,  
       t.type_name,  
       CONCAT(l.area, ', ', l.city, ', ', l.state) AS location,  
       o.officer_name  
FROM crime_case c  
JOIN crime_type t ON c.type_id = t.type_id  
JOIN location l ON c.location_id = l.location_id  
JOIN officer o ON c.officer_id = o.officer_id;
```

#### Explanation:

This query merges multiple tables to show the full case details in a single table.

---

### 2. View for Open Cases

```
CREATE VIEW open_cases AS  
SELECT case_id, case_code, case_date  
FROM crime_case  
WHERE status = 'Open';
```

### **Explanation:**

Creates a virtual table showing only open cases.  
Useful for filtering active investigations.

---

### **3. Trigger Example**

```
CREATE TRIGGER case_insert_log  
AFTER INSERT ON crime_case  
FOR EACH ROW  
INSERT INTO case_log(action, case_code)  
VALUES ('Case Inserted', NEW.case_code);
```

### **Explanation:**

Automatically logs whenever a new case is added.

---

### **4. Group By Query (Statistics counters)**

```
SELECT status, COUNT(*)  
FROM crime_case  
GROUP BY status;
```

### **Explanation:**

Used for showing total open and closed cases on the dashboard.

## **GUI WORKING (FRONTEND)**

The GUI is built using Flask + HTML + CSS.

### **✓ Add New Case**

User enters case details through a modal form.

## **✓ Manual Location Entry**

Area, City, State fields are provided.

## **✓ Manual Officer Entry**

User enters officer name (no dropdown).

## **✓ View Case Details**

Shows full information with location and officer.

## **✓ Delete Case**

Removes entry from database.

## **✓ Statistics**

Shows total cases, open cases, and closed cases.

The design is clean, responsive, and easy to operate.

## **CONCLUSION**

The Crime Data Analysis System successfully demonstrates:

- Database creation
- Table relationships
- Joins, views, triggers
- CRUD operations
- GUI development using Flask

The project meets all objectives of DBMS and provides a functional system for crime data record-keeping and analysis.

## **FUTURE SCOPE**

The system can be further enhanced by adding:

- Role-based login (admin, officer, viewer)
- Heatmap-based crime analysis (GIS integration)
- Predictive analytics using machine learning
- Automatic report generation
- Secure authentication system