Police Department Database Management System

Ipsita Singh (111601033), Shubham Gupta(111601024), Ishan Gadgil (111601007)

Contents

1. Contribution

Schema: Ipsita Singh, Ishan Gadgil and Shubham Gupta
Drew ERD: Ipsita Singh
Insert data in all tables: Shubham Gupta
Code Testing: Ishan Gadgil
Views and Triggers: Ipsita Singh, Ishan Gadgil and Shubham Gupta
Roles: Shubham Gupta
GUI: Ipsita Singh and Shubham Gupta
Use Cases: Ishan Gadgil
BCNF Form - Ipsita Singh
Functions and Procedures: Ishan Gadgil
Report: Ishan Gadgil
Code Optimisation : Shubham Gupta

2 Introduction

In this project, our aim was to come up with a reasonably scalable database with GUI maintained for a police department.

2.1 Requirements

1) The database stores the data of police employees as well as that related to the criminals and crimes occured.

2) Database has police employees data of each district of Kerala state which store the information regarding their rank, salary etc.

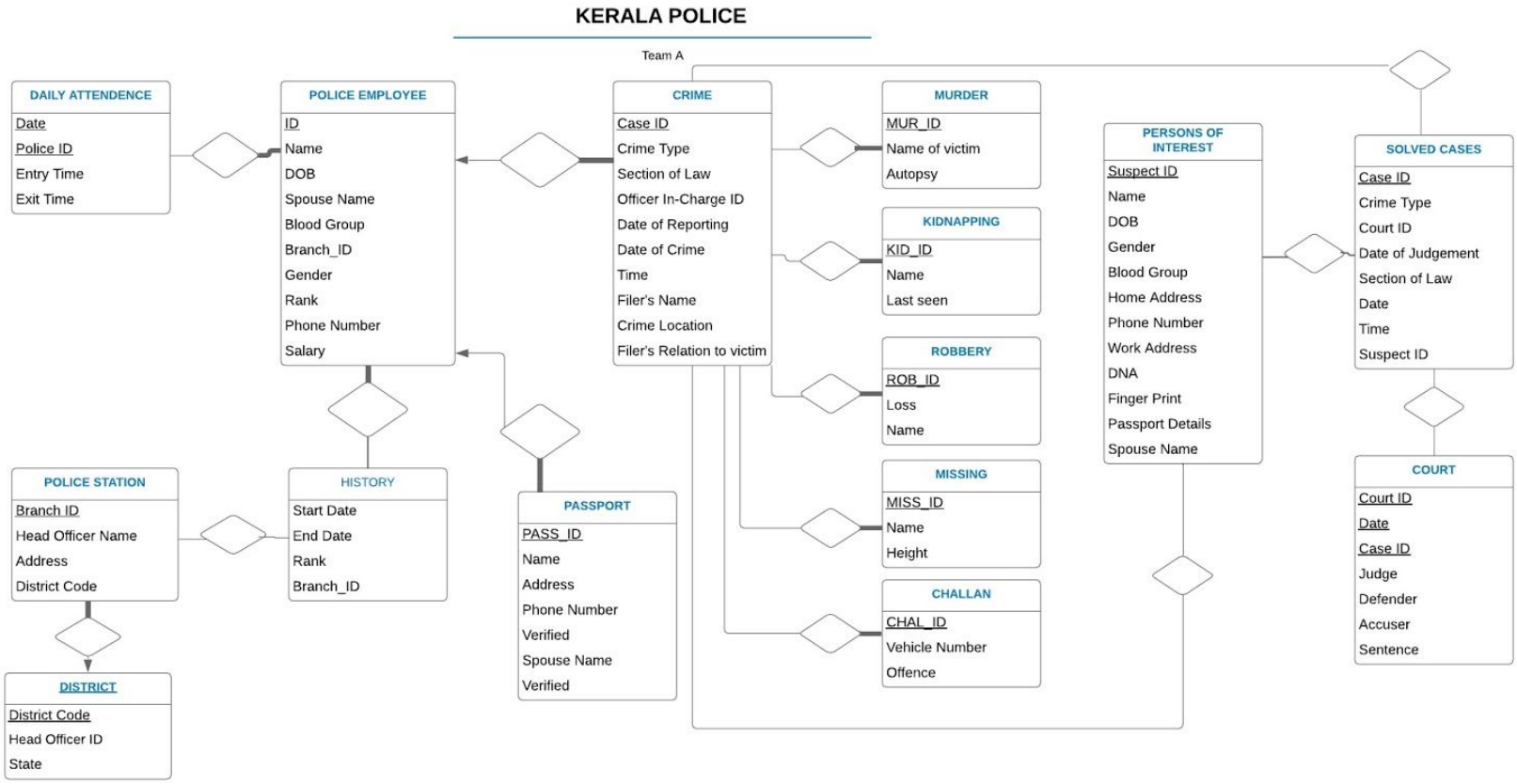
3) The database should also accounts for the past history of the employees.

4) There will be information related to each police station of the each district, and there are several police stations in a district.

5) There are two types of cases i.e general complaints and criminal cases. General complaints do not involve any severe crime complaints like missing complaint etc. While criminal complaints involve complaints like murder, kidnapping, robbery etc

6) Each complain has an id where first 3-characters indicate the type of crime. All types of criminal cases have an associated tables for suspects/person of interest.

3 Entity Relation Diagram



4 Database Schema and Normalization

All the tables are in BCNF form.

5 Roles, Triggers and Views

5.1 Roles

There are several police employees in the police department, therefore they are assigned roles and permissions according to their designation.

The four roles (arranged in increasing order of hierarchy) in our database are
i) Clerk
ii) Constable
iii) Superintendent
iv) Director

The clerk is to be given permissions relevant to his job. He/she can only access(i.e. select) the view employee and all permissions to the table attendance.

As we the rank increases, the number of permissions granted to the roles also increases. For example a superintendent has all privileges granted to the constable. Also, he/she will have some additional privileges. But there would be some privileges which only the director would have. (Ex. Deleting an entry from the crime database)

	Attend.	Employee	History	Police Station	Crime	Crime Table	POI	Solved Cases	Court	Passport
Clerk	S, I, D	S[3]		S						
Constable		S[4]		S	S	S,(I)[1]	S	S	S	S,I,D
Super	S	S[3]	S[2]	S	S,I,U	S,I U	S,I,U,D	S,I,U	S, I,U	S, I, D
Dir.	S	S[3]	S	S	S,I,U,D	S,I U,D	S,I,U,D	S,I,U	S,I,U	S,I,D

The Crime Table column implies the following tables
1) Challan
2) Kidnapping
3) Missing
4) Murder

The Employee column implies the following views
1) employee_level1
2) employee

Also, director role has all permission on the Police_Employee table

Notations:
1 : (I) is only on Challan and Missing relations
2 : Not on ranks above (i.e a superintendent can't view the history of a director)
3 : Select on view employee_level1
4 : Select on view employee

5.2 Views

The following views have been implemented.

1) View employee_level1
The view employee_level1 abstracts out the some of the attributes from the Police_Employee table. The attributes Spouse_name, Phone_number and salary are not included in this view. Select privilege are granted to clerk, superintendent and director.

2) View employee
This view is similar to the view employee_level1 but also includes the attribute salary. Only the role superintendent is granted select permission on this view.

3) View poi_level1
Abstraction of the tabe Person_of_Interest without the Phone_no, DNA and finger_print attributes. The select privilege on this view is granted to the role constable.

4) View superintendent_history
This view is created to ensure that the superintendent can access the history of all the employees except those with role director (i.e. hierarchical access to the employee history) The select privilege on this view is granted only to superintendent role.

5) View attendance_level2
This view is created to ensure that the superintendent can access the attendance of all the employees except those with role director (i.e. hierarchical access to the attendance history) The select privilege on this view is granted only to superintendent role.

6) View solved_cases_data

This view displays all the details about the cases which have been solved. Select privilege granted to all roles.

7) View ongoing_cases_data
This view displays all the details about the cases which are open. Select privilege granted to all roles.

5.3 Triggers

A trigger is a stored program executed automatically to respond to a specific event e.g., insert, update or delete occurred in a table. The following triggers have been implemented in this database:

1) Trigger insert_crime
When an entry is inserted into the crime table a corresponding entry is inserted into the corresponding table. Example, if a murder is inserted into the crime table, corresponding entry is inserted into the Murder table.

2) Trigger suspect_to_victim
Trigger to see if any suspect has become victim of a crime(Murder or kidnapping). While inserting into the murder/kidnapping table if the victim is already a person of interest, a message is displayed "This person is in Suspect Table".

3) Trigger tr_insert_crime
Whenever an update/insert is made to the Crime table, an entry is inserted into the Insert_Crime_Audit table. This table has logs of all the updates/insert made to the Crime table.

4) Trigger tr_insert_Solved_cases
Whenever an update is made to the Solved_Cases table, an entry is inserted into the Insert_Solved_Cases_Audit table. This table has logs of all the updates made to the Solved_Cases table.

6. Function and Procedures

The following functions and procedures have been implemented on this database:

1) Procedure proc_city_station to get the details of all the police stations in a given city.

2) Function fun_attendance_per to check the percentage of attendance of an employee.

3) Procedure rankEmp to display all the employees for a given rank.

4) Procedure mark_entry to mark entry into the Attendance table with current date and time.

5) Procedure mark_exit to mark exit into the Attendance table with current time.

6) Function num_cases to find the number of open cases ongoing under a police officer.

7) Procedure Crime_stat to give the crime statistics.

8) Procedure Criminal_Finder to give the details of criminal based on weapon used in crime.

9) Procedure to know the summary of solved _cases

10) Procedure todayBirthday() to find the employees who are having birthday today.

11) Procedure presentToday() to find the number of employees present today.

12) Procedure myHistory to find the posting history of a given employee

13) Procedures missingQuery1 and missingQuery2 to identify a missing person. The procedure missingQuery1 takes input age, color and height. While missingQuery2 takes input color and height. Both the procedures input the probable list of people who match the given inputs.

14) Procedure crimeCourtHistory to find the legal status and history of a case.

15) Function openCaseCount to display the number of open cases.

7 Use Cases

Query 1

Show all the police employee who have the Rank ‘SP’. This is obtained by selecting Name from the police employee table where rank = ‘SP’.

```
MariaDB [project]> select Name from Police_Employee where Rank = 'SP';
+---------------+
| Name          |
+---------------+
| Sanjeev Tyagi |
| Nissar Khan   |
+---------------+
2 rows in set (0.00 sec)
```

Query 2

Show all the employees who were absent on 1st February, 2019. This is obtained by selecting all the employees who were NOT in the output returned by query (all employees present on 1st February, 2019)

```
MariaDB [project]> select * from Police_Employee where Police_ID not in (select Police_ID from Attendance where Date='2019-02-01');
```

Police_ID	Name	DOB	Spouse_name	Blood_Group	Gender	Rank	Phone_number	salary	Department
0001	Rakesh	1965-01-01	Anita	A+	M	Constable	2147483647	35000	Police

1 row in set (0.00 sec)

Query 3

Displaying all murders reported between 1 January, 2018 and 1 January, 2019. This can be obtained by doing inner join on the tables crime and murder where murder ID = crime ID and selecting all murders where date of reporting is between 1 January, 2018 and 1 January, 2019.

```
MariaDB [project]> select * from Crime join Murder on Crime.Case_ID = Murder.Murder_ID where Crime_Type='Murder' and Date_Of_Reporting between '2018-01-01' and '2019-01-01';
```

Case_ID	Crime_Type	Police_ID	Date_Of_Reporting	Date_Of_Crime	Time_Of_Reporting	Time_Of_Crime	Filer_Name	Crime_Location	Filer_Relation_To_Victim	Murder_ID	Name_Of_Victim	Weapon	Autopsy
0001	Murder	0001	2019-01-01	NULL	10:50:00	NULL	A	NULL	Brother	0001	X	NULL	Y
102	Murder	IPS1110	2018-09-17	2018-09-16	12:15:00	23:03:00	Piyush thakur	Railways colony,Thrissur	Father	102	Pradyumn Thakur	Poison,Hanged with Rope, IRon Rod	Y

2 rows in set (0.00 sec)

Query 4

Displaying all the crime types and the frequency of each crime type in our database.

```
MariaDB [project]> select Crime_Type, count(Crime_Type)
    -> from Crime group by (Crime_Type);
```

Crime_Type	count(Crime_Type)
Murder	3
Kidnapping	1
Robbery	1
Missing	1
Challan	1

5 rows in set (0.00 sec)

Query 5

If given a weapon, show which person of interest has history of using it and the case in which he had used the weapon. This is obtained by joining the Murder table on Murder ID and then joining the Crime table on Case ID and then selecting where weapon = “givenWeapon”.

```
MariaDB [project]> select Name, DOB, Gender, Case_ID, Suspect_ID from
    -> Person_of_interest join Murder on Murder.Murder_ID = Person_of_interest.Case_Involved join Crime on Crime.Case_ID = Person_of_interest.Case_Involved
    -> where Weapon='Tibbetian Knife';
+----------------+------------+--------+---------+------------+
| Name           | DOB        | Gender | Case_ID | Suspect_ID |
+----------------+------------+--------+---------+------------+
| Ganesh Gaitonde | 1997-10-25 | M      | 101     | S101       |
+----------------+------------+--------+---------+------------+
1 row in set (0.00 sec)
```

Query 6

Showing all the stations in the city “Palakkad”.

```
MariaDB [project]> select * from Police_Station where City='Palakkad';
+-----------+-----------+-------------+-----------------------------------------+----------+-------------+
| Branch_ID | Police_ID | Building_No | Street                                  | City     | District_ID |
+-----------+-----------+-------------+-----------------------------------------+----------+-------------+
| KL101     | IPS1001   |           1 | KSHB Collony, Kalleppully               | Palakkad | 678005      |
| KL102     | IPS1105   |          13 | Hemambika Nagar, Railway Colony, Olavakkode | Palakkad | 678731      |
+-----------+-----------+-------------+-----------------------------------------+----------+-------------+
2 rows in set (0.00 sec)
```

Query 7

Given employee name show his posting history. This is done by selecting all the tuples in the table History where Police_ID is the ID corresponding to the employee name.

```
MariaDB [project]> select * from History
    -> where Police_ID in (select Police_ID from Police_Employee where Name='Abhishek Khandekar');
+-----------+-----------+------------+------------+------+
| Police_ID | Branch_ID | Start_Date | End_Date   | Rank |
+-----------+-----------+------------+------------+------+
| IPS1205   | KL110     | 2014-12-01 | 2015-12-01 | SP   |
| IPS1205   | KL120     | 2001-02-05 | 2014-12-01 | SP   |
+-----------+-----------+------------+------------+------+
2 rows in set (0.00 sec)
```

Query 8

Show the history of every case being heard in different courts.

```
MariaDB [project]> select * from Court;
+----------+------------+----------+------------------+-----------------+----------------------+-----------------------------------------------+
| Court_ID | Date       | Crime_ID | Name_Of_Judge    | Name_Of_Defender | Name_Of_Accuser      | Sentence                                      |
+----------+------------+----------+------------------+-----------------+----------------------+-----------------------------------------------+
| C001     | 2008-06-12 | 101      | Aadish Parmar    | Rustum Pavri    | Amit Makhija         | No conclusion, court is adjourned till 2008:8:25 |
| C001     | 2018-12-12 | 102      | Aadish Parmar    | Anita Modi      | Parvez Shaikh        | Suraj also involved,next date 2019:2:01       |
| C002     | 2018-02-02 | 103      | Diljit Dosanjh   | Preetam Das     | Arijit Singh         | Imprison for 10 years under IPC Sec. 359,closed |
| C122     | 2019-05-22 | 104      | Keerthi Kulsekara | Siraj Alam      | Jagdish chandra patnik | Imprison for 23 years under IPC Sec. 395,closed |
+----------+------------+----------+------------------+-----------------+----------------------+-----------------------------------------------+
4 rows in set (0.00 sec)
```

Query 9

Show the judgement of a solved case with a given case id. This is obtained by doing natural join on the Solved case table and the Court table on Court.Crime_ID = Solved_Cases.Case_ID where case ID = “givenCaseID”.

```
MariaDB [project]> select Court.Court_ID, Date, Name_Of_Judge , Section_Of_Law, Sentence from Court join Solved_Cases on Court.Crime_ID = Solved_Cases.Case_ID where Solved_Cases.Case_ID = '103';
+----------+------------+---------------+---------------+-----------------------------------------+
| Court_ID | Date       | Name_Of_Judge | Section_Of_Law | Sentence                                |
+----------+------------+---------------+---------------+-----------------------------------------+
| C002     | 2018-02-02 | Diljit Dosanjh | IPC 359       | Imprison for 10 years under IPC Sec. 359,closed |
+----------+------------+---------------+---------------+-----------------------------------------+
1 row in set (0.00 sec)
```

10) Implementation of procedure Criminal_Finder

```
MariaDB [Project]> CREATE PROCEDURE Criminal_Finder (Crime_weapon varchar(30))
    -> BEGIN
    -> SELECT Name, DOB,Gender,Case_ID, Suspect_ID
    -> FROM Person_of_interest JOIN Murder ON Murder.Murder_ID = Person_of_interest.Case_Involved JOIN C
rime ON Crime.Case_ID = Person_of_interest.Case_Involved
    -> WHERE Weapon = Crime_weapon;
    -> END;
    -> //
Query OK, 0 rows affected (0.055 sec)

MariaDB [Project]> call Criminal_Finder (Tibbetian Knife);
    -> //
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your Mari
aDB server version for the right syntax to use near 'Knife)' at line 1
MariaDB [Project]> call Criminal_Finder ('Tibbetian Knife');
    -> //
+-----------------+------------+--------+---------+------------+
| Name            | DOB        | Gender | Case_ID | Suspect_ID |
+-----------------+------------+--------+---------+------------+
| Ganesh Gaitonde | 1997-10-25 | M      | 101     | S101       |
+-----------------+------------+--------+---------+------------+
1 row in set (0.001 sec)

Query OK, 0 rows affected (0.002 sec)
```

11) Implementation of procedure Crime_stat

```
MariaDB [Project]> CREATE PROCEDURE Crime_stat ()
    -> BEGIN
    -> SELECT Crime_Type,COUNT(Crime_Type) FROM Crime GROUP BY (Crime_Type);
    -> END;
    -> //
Query OK, 0 rows affected (0.021 sec)

MariaDB [Project]> call Crime_stat();
    -> //
+------------+--------------------+
| Crime_Type | COUNT(Crime_Type) |
+------------+--------------------+
| Murder     |                 11 |
| Kidnapping |                  6 |
| Robbery    |                  7 |
| Missing    |                  6 |
| Challan    |                  5 |
+------------+--------------------+
5 rows in set (0.001 sec)

Query OK, 0 rows affected (0.002 sec)
```

12) Implementation of procedure proc_city_station

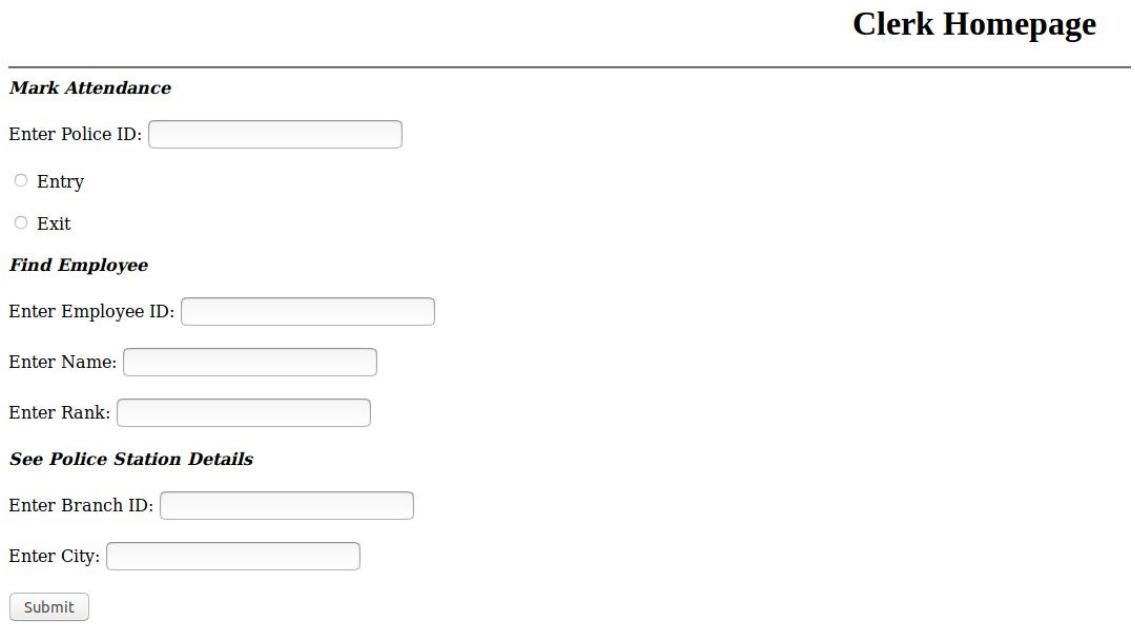
```
MariaDB [Project]> CREATE PROCEDURE proc_city_station(city varchar(50))
    -> BEGIN
    ->     SELECT * FROM Police_Station WHERE Police_Station.City = city;
    -> END;
    -> //
ERROR 1304 (42000): PROCEDURE proc_city_station already exists
MariaDB [Project]>
MariaDB [Project]> call proc_city_station('Palakkad');//
+-----------+-----------+-------------+-----------------------------------------+----------+-------------+
| Branch_ID | Police_ID | Building_No | Street                                  | City     | District_ID |
+-----------+-----------+-------------+-----------------------------------------+----------+-------------+
| KL101     | IPS1001   |           1 | KSHB Collony, Kalleppully               | Palakkad | 673004      |
| KL102     | IPS1002   |          13 | Hemambika Nagar, Railway Colony, Olavakkode | Palakkad | 678005      |
| KL107     | IPS1201   |           2 | 24 Harvard Street Rocklin               | Palakkad | 211179      |
+-----------+-----------+-------------+-----------------------------------------+----------+-------------+
3 rows in set (0.001 sec)

Query OK, 0 rows affected (0.001 sec)
```

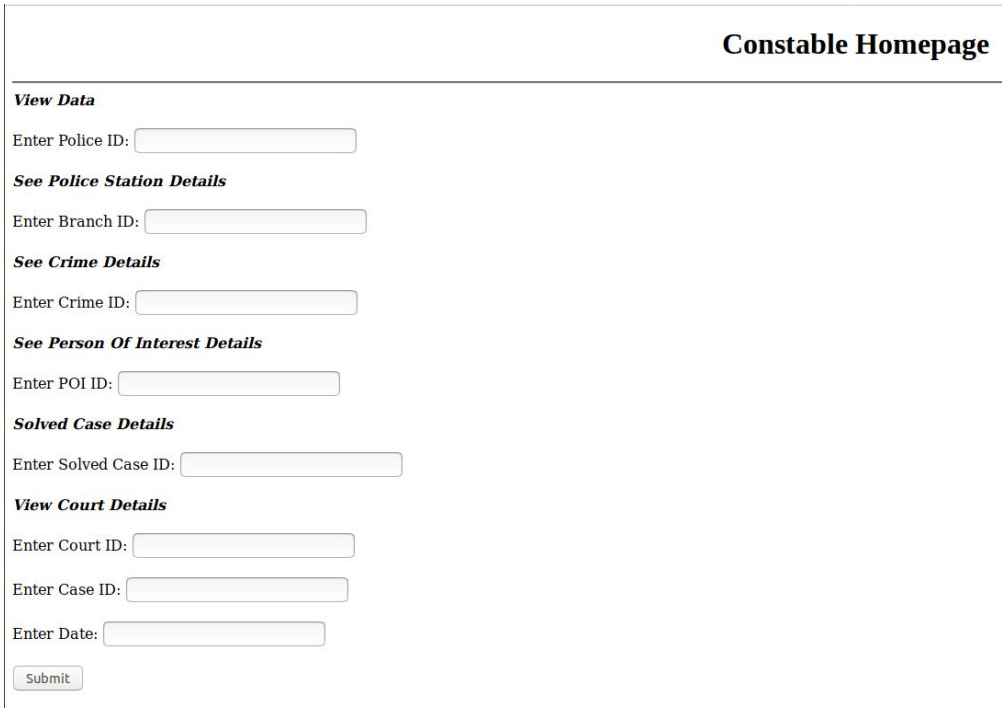
8 GUI

Some screenshots:
1) Main login page

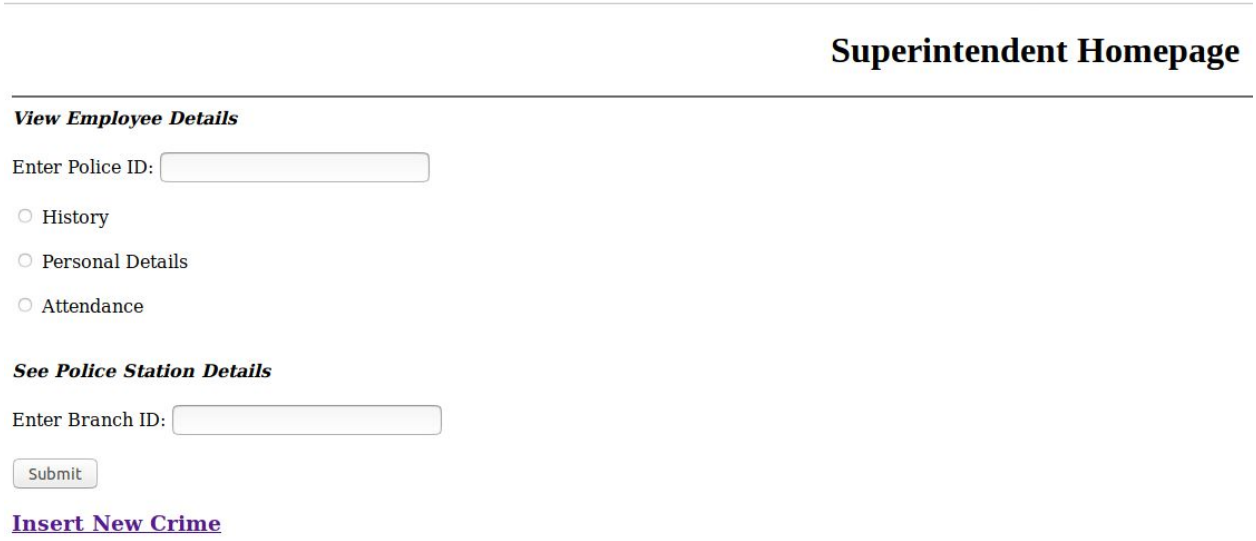
2) After a successful login clerk is directed to his homepage:



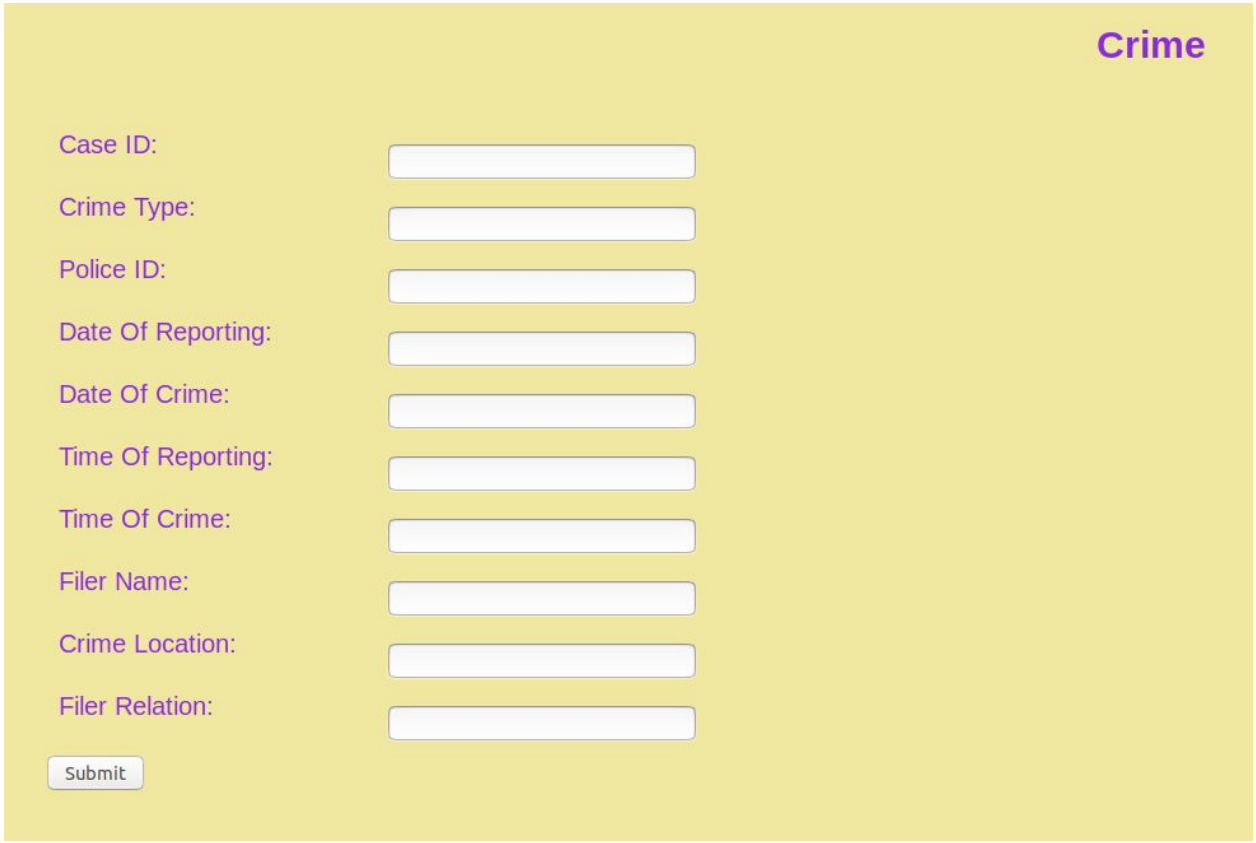
3) After a successful login constable is directed to his homepage -



4) After a successful login superintendent is directed to his homepage -



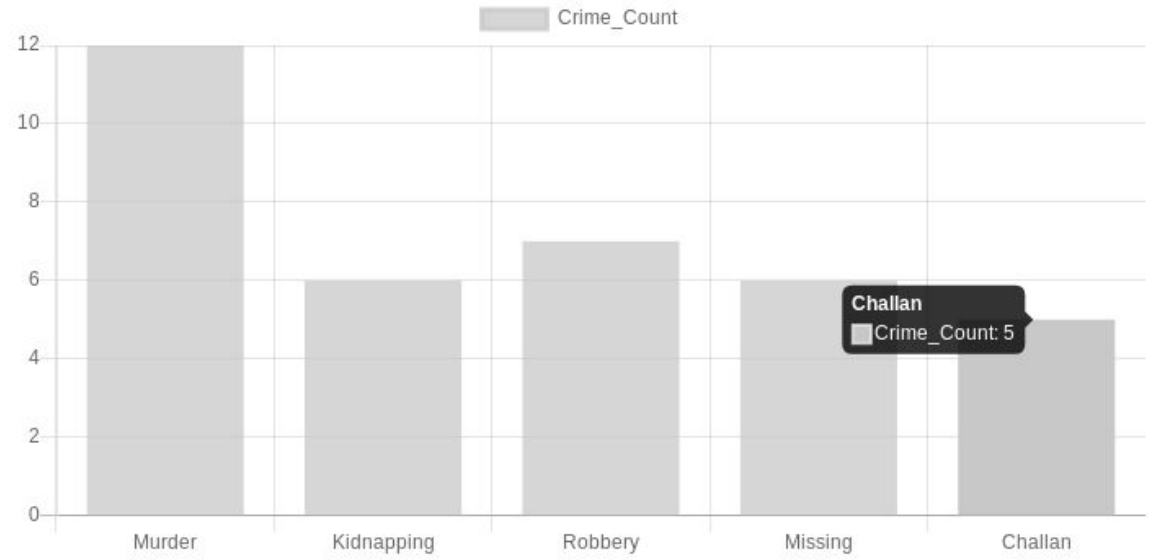
5) Interface to insert a new crime -



Graphs

Only the employees with the role director can view the graphs.

1) Total number of different types of crime.



2) Total number of cases under police officers currently