Record Management System

Project Report

Group 3

Gudipalli Sri Sai Prabhath Reddy

Yashwant Dontam

857-398-7947 (Prabhath)

857-230-5413 (Yashwant)

gudipalli.s@northeastern.edu

dontam.y@northeastern.edu

Percentage of Effort Contributed by Student1: 50%

Percentage of Effort Contributed by Student2: 50%

Signature of Student 1: Gudipalli Sri Sai Prabhath Reddy

Signature of Student 2: Yashwant Dontam

Submission Date: 12/10/2023

1. Introduction

Data-driven policing could transform the way law enforcement agencies manage their data. A good Records Management System (RMS) would go a long way toward achieving notable efficiency and precision. A record management system can offer data entry, retrieval, and analysis. Correct and up-to-date information on the suspects, their criminal records, the case details, and the evidence items must be stored in the record management system. This could help identify suspects, track criminal records, and build case files. An RMS can also be used to store information within the organization, like details and schedules of the personnel working. This approach signifies a significant step ahead in adapting law enforcement to the demands of the 21st century. Down the road, this can help law enforcement identify trends, patterns, and potential leads, enhancing their ability to prevent and solve crimes efficiently and effectively. Our project takes a law enforcement division standpoint to manage and analyze data related to criminal activities, personnel, resources, and community interactions in their effort to enforce the law.

Theory for Record Management:

In our context of a law enforcement division's record management system, data can be recorded and managed from two fundamental perspectives: Case and Incident record management and Administrative record management. In case and incident management, the LED records core information about incident reports, cases, victims, suspects, and evidence. For every incident, the LED must record the incident ID, date and time, location, and status (open, closed, or under investigation). The case ID, assigned officers, incident, case description, and case status (open, closed, or pending) will be recorded for the cases. For the victim data, the victim ID, name, contact, and statement will be registered. The suspect's ID, name, date of birth, address, and criminal record must be stored. The data about the evidence, like evidence ID, type (physical, digital), description, chain of custody, and storage location will be recorded.

When it comes to personnel and administrative records management, the LED looks to record the officers' details, shift schedules, access control, and vehicle records. For the officers' data, officer ID, name, rank, badge number, and contact will be recorded. For every day of the week, the LED must record and store the schedule ID, employee ID, shift dates, and times. The user IDs, passwords, and role-based access should be stored for authentication and access control. For the vehicles owned by the department, vehicle ID, model, license plate number, assigned officer, and maintenance history will be recorded.

Other requirements:

1) One incident report can be connected to 0 to an infinite number of cases; one case should have oneincident report.
2) Law enforcement officers can be associated with multiple cases, but each case must have at leastone appointed officer.
3) Cases can be linked with 0 to multiple number of evidence items.
4) Each suspect must be linked with at least one case, but cases should have at least one suspect tomultiple suspects.
5) Every case should include information about at least one victim, while the victim records can beconnected to zero to infinite cases.
6) Every case should have a record of at least one administrative document, while the administrativedocuments can be related to infinite cases.
7) User accounts control access to the system, while each user has specific access rights

2. Methodology

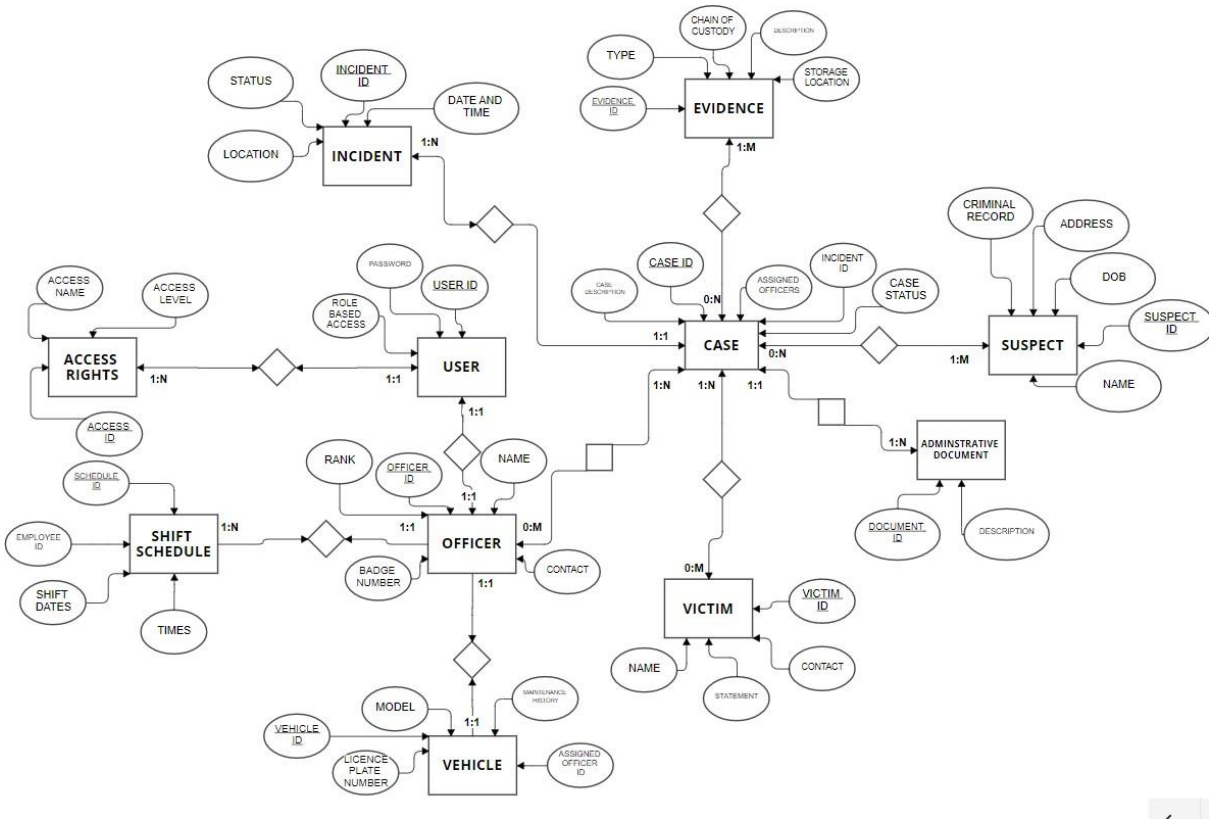


Figure 2.1: EER Model

The EER diagram models a law enforcement or security system, detailing how incidents, cases, evidence, suspects, officers, victims, and vehicles are interrelated. Key points include:

- Incidents are recorded with time and location details, and can be linked to multiple pieces of evidence and cases.
- Evidence is cataloged with unique identifiers and has a tracking mechanism for custody and storage.
- Cases are built around incidents and can involve multiple suspects and officers, with each case having a unique status.
- Suspects and victims have personal details recorded, and suspects additionally have criminal records linked.
- Officers have ranks, badge numbers, and are assigned to shifts and vehicles, with their schedules and vehicle maintenance histories tracked.
- Administrative documents are maintained for supplementary case documentation. Users of the system have role-based access, indicating a secure, permission-based system.
- Access rights are determined by levels and names, controlling what parts of the system users can interact with

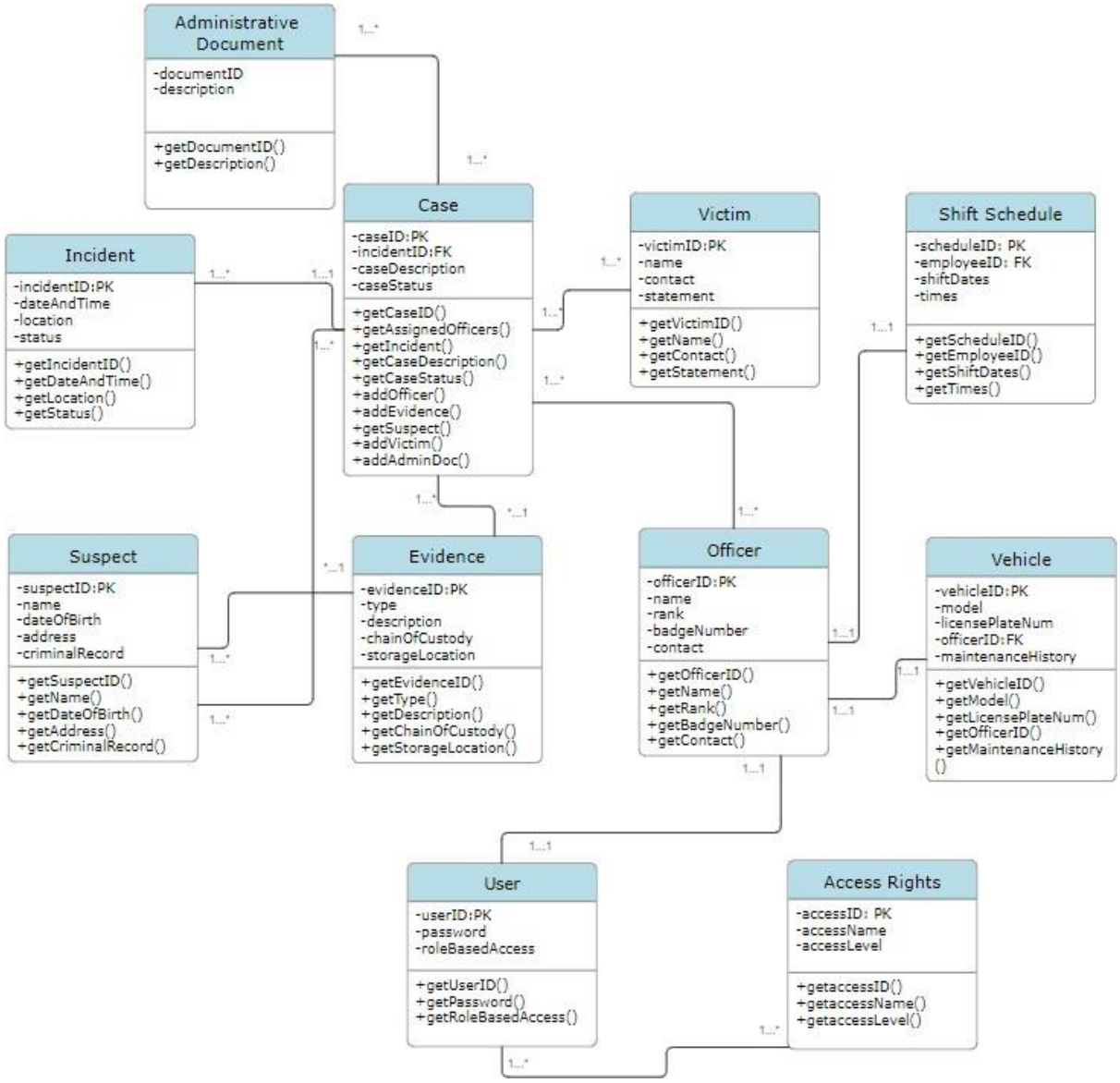


Figure 2.2: UML Model

The UML class diagram models the structure of a system likely used by law enforcement to manage incidents, cases, and related information. It outlines the attributes and methods of entities such as incidents, cases, suspects, evidence, officers, vehicles, victims, users, access rights, and administrative documents, specifying how these entities are related and interact with each other. Each class has attributes unique to the entity it represents, as well as methods to retrieve or manipulate these attributes. Relationships between classes indicate that an incident can be linked to multiple cases, suspects, and pieces of evidence, while officers are associated with vehicles and shift schedules. Users have role-based access, suggesting a system with controlled permissions. The structure implies a complex, interconnected system with defined operations to manage and access law enforcement data.

3. Relational Model

Case (CaseID, Case Description, Assigned Officers, Case Status)
Primary key: CaseID

Involved (CaseID, SuspectID)

Foreign keys: CaseID, SuspectID ; NOT NULL

CaseID referring to the primary key in the Entity Case;

SuspectID referring to the primary key in the Entity Suspect.

Suspect (SuspectID, SuspectName, SuspectDOB, SuspectAddress, CriminalRecord)
Primary key: SuspectID

Administrative Document (DocumentID, Description, CaseID)
Primary key: DocumentID;

Foreign key: CaseID; NOT NULL

CaseID referring to the primary key in the Entity Case

Victim (VictimID, VictimName, VictimContact, VictimStatement, CaseID)
Primary key: VictimID;
Foreign key: CaseID; NOT NULL
CaseID referring to the primary key in the Entity Case

Evidence (EvidenceID, EvidenceType, ChainOfCustody, EvidenceDescription, StorageLocation, CaseID)
Primary key: EvidenceID;
Foreign key: CaseID; NOT NULL
CaseID referring to the primary key in the Entity Case

Incident (IncidentID, IncidentLocation, IncidentStatus, IncidentDate, IncidentTime, CaseID)
Primary key: IncidentID;
Foreign key: CaseID; NOT NULL
CaseID referring to the primary key in the Entity Case

AssignedTo (CaseID, OfficerID)

Foreign keys: CaseID, OfficerID; NOT NULL

CaseID referring to the primary key in the Entity Case;

OfficerID referring to the primary key in the Entity Officer.

Officer (OfficerID, OfficerName, OfficerContact, OfficerRank, BadgeNumber, VehicleID, ScheduleID, UserID)

Primary key: OfficerID;

Foreign keys: VehicleID, ScheduleID, UserID; NOT NULL

VehicleID referring to the primary key in the Entity Vehicle;

ScheduleID referring to the primary key in the Entity Schedule;

UserID referring to the primary key in the Entity User.

User (UserID, RoleBasedAccess, Password)

Primary key: UserID

Access Rights (AccessID, AccessName, AccessLevel, UserID)

Primary key: AccessID;

Foreign keys: UserID; NOT NULL

UserID referring to the primary key in the Entity User

Vehicle (VehicleID, VehicleModel, LicencePlateNumber, MaintenanceHistory)

Primary key: VehicleID

Shift Schedule (ScheduleID, ShiftDates, ShiftTime)

Primary key: ScheduleID

4. SQL Demonstration

Queries showing the Tables and the Columns

1.

```
CREATE TABLE
   AccessRights ( AccessID
   INT PRIMARY KEY,
   AccessName TEXT,
   AccessLevel TEXT,
   UserID INT NOT
   NULL,
   FOREIGN KEY (UserID) REFERENCES User(UserID)
);
```

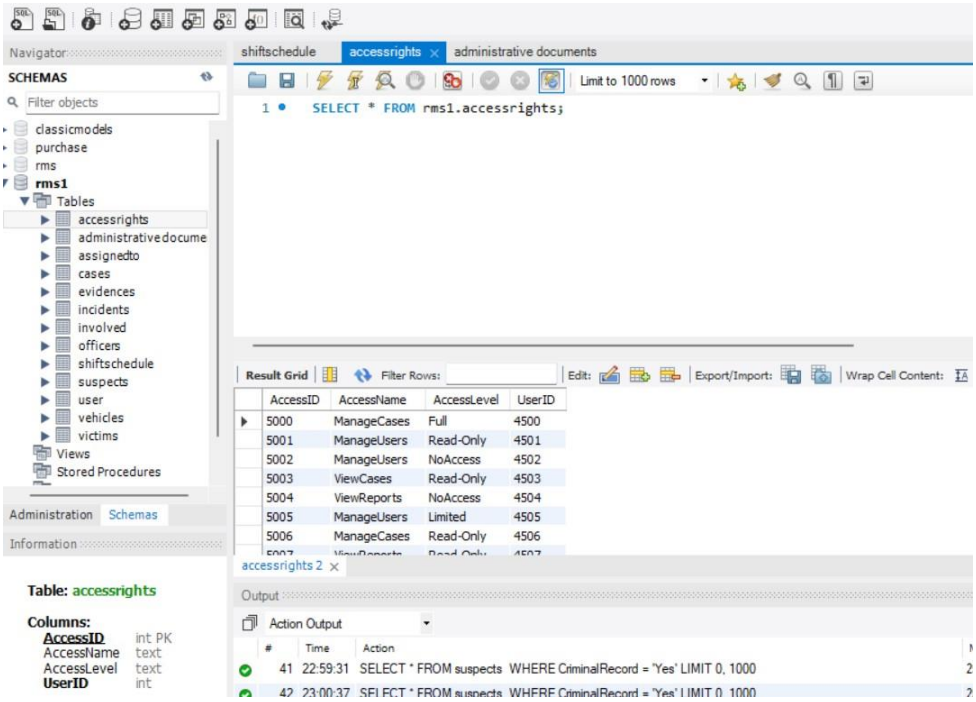


Figure 4.1.1

2.

```
CREATE TABLE
   AdministrativeDocuments (
   DocumentID INT PRIMARY KEY,
   Description TEXT,
   CaseID INT NOT NULL,
   FOREIGN KEY (CaseID) REFERENCES Cases(CaseID)
);
```

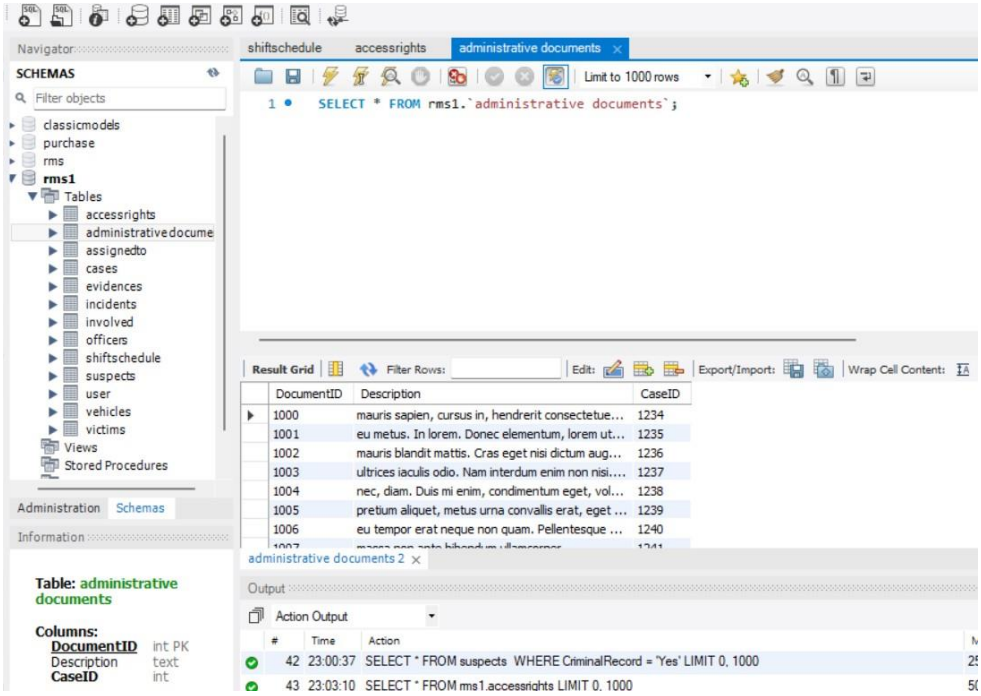


Figure 4.1.2

3.

```
CREATE TABLE
   AssignedTo (CaseID INT
   NOT NULL, OfficerID
   INT NOT NULL,
   PRIMARY KEY (CaseID, OfficerID),
   FOREIGN KEY (CaseID) REFERENCES Cases(CaseID),
   FOREIGN KEY (OfficerID) REFERENCES Officers(OfficerID)
);
```

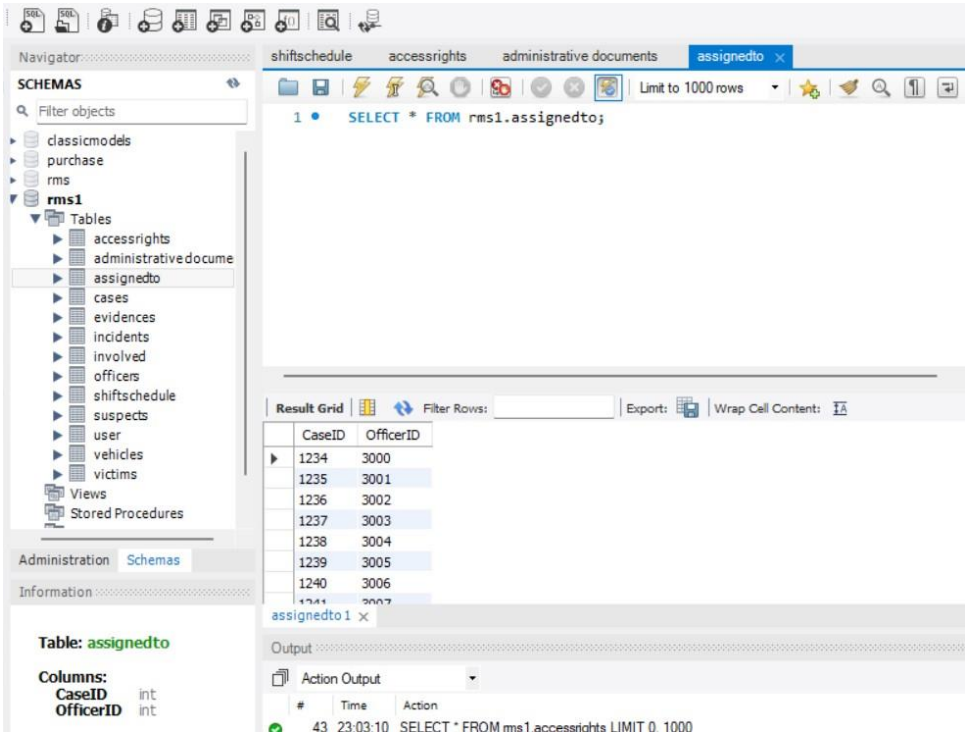


Figure 4.1.3

4.

```
CREATE TABLE cases (
   CaseID INT PRIMARY
   KEY,
   CaseDescription
   TEXT,
   AssignedOfficers
   TEXT,       CaseStatus
   TEXT
);
```

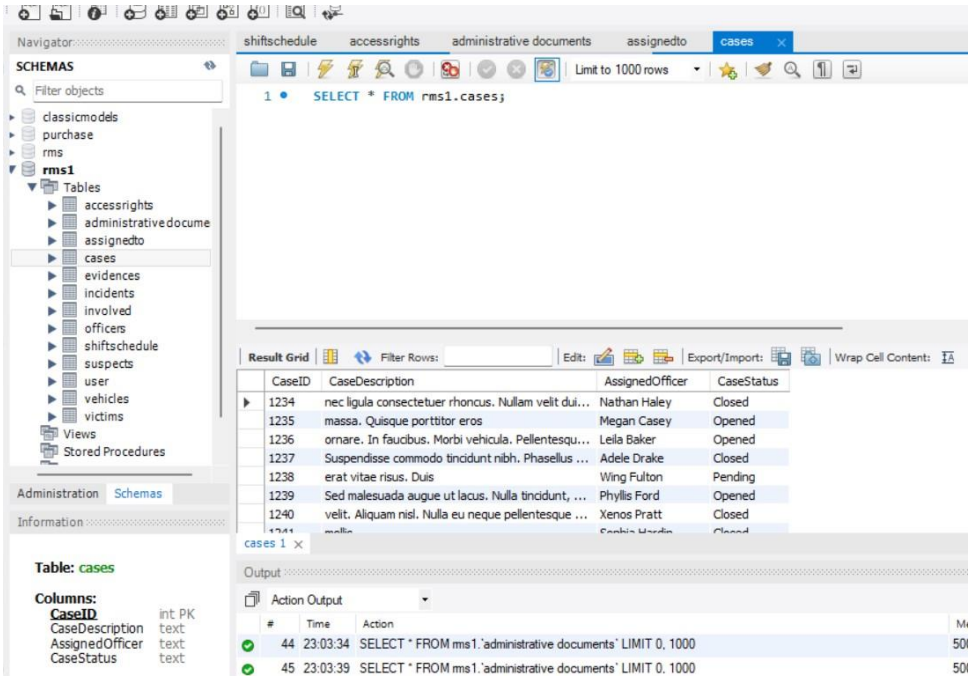


Figure 4.1.4

5.

```
CREATE TABLE evidences (
   EvidenceID INT PRIMARY
   KEY,
   EvidenceType TEXT,
   ChainOfCustody TEXT,
   EvidenceDescription
   TEXT,StorageLocation
   TEXT, CaseID INT NOT
   NULL,
   FOREIGN KEY (CaseID) REFERENCES Case(CaseID)
);
```

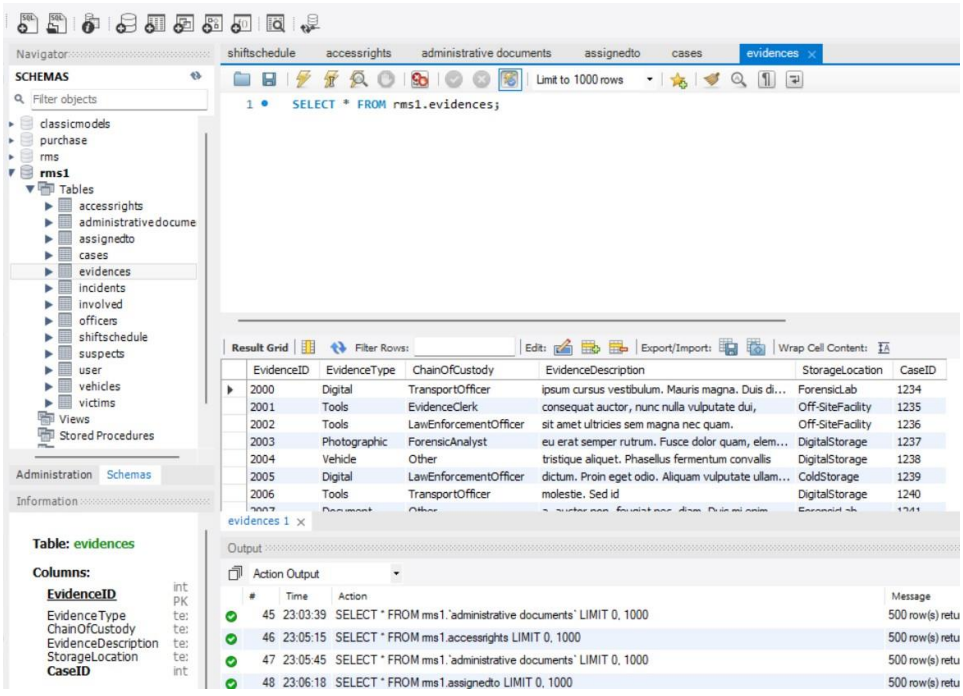


Figure 4.1.5

6.

```
CREATE TABLE incidents (
   IncidentID INT PRIMARY
   KEY,
   IncidentLocation
   TEXT,IncidentStatus
   TEXT, IncidentDate
   DATE, IncidentTime
   TIME, CaseID INT
   NOT NULL,
   FOREIGN KEY (CaseID) REFERENCES Cases(CaseID)
);
```

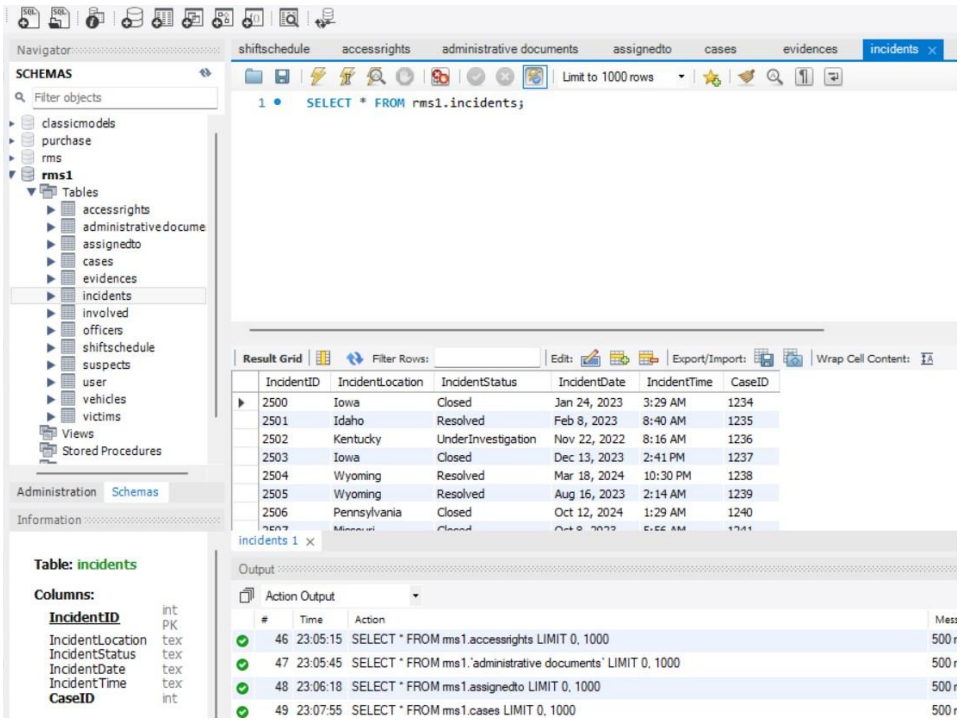


Figure 4.1.6

7.

```
CREATE TABLE Involved
   ( CaseID INT NOT
   NULL, SuspectID INT
   NOT NULL,
   PRIMARY KEY (CaseID, SuspectID),
   FOREIGN KEY (CaseID) REFERENCES Cases(CaseID),
   FOREIGN KEY (SuspectID) REFERENCES Suspect(SuspectID)
);
```

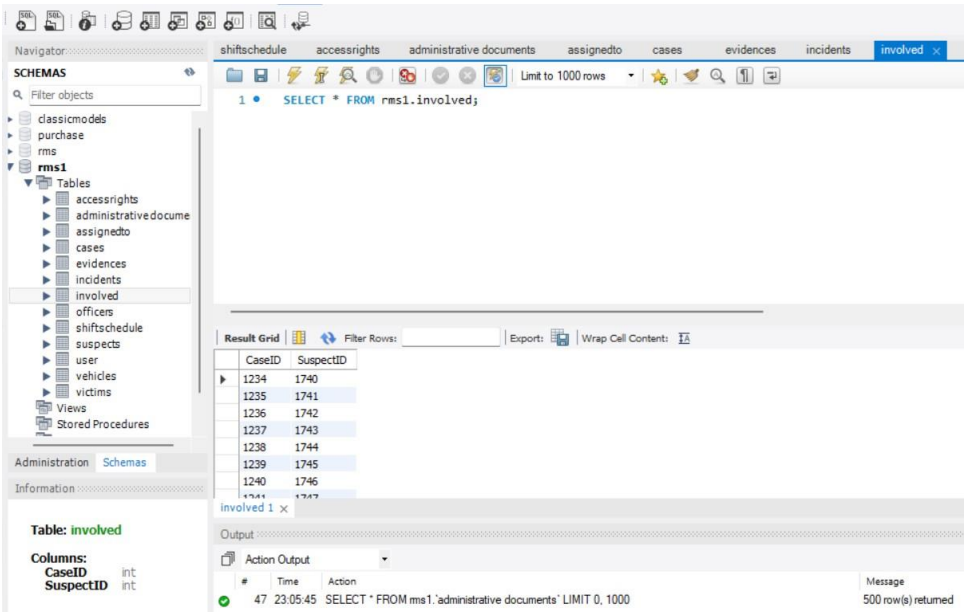


Figure 4.1.7

8.

```
CREATE TABLE Officers (
    OfficerID INT PRIMARY
    KEY,
    OfficerName TEXT,
    OfficerContact TEXT,
    OfficerRank TEXT,
    BadgeNumber TEXT,
    VehicleID INT NOT
    NULL, ScheduleID INT
    NOT NULL,UserID INT
    NOT NULL,
    FOREIGN KEY (VehicleID) REFERENCES Vehicle(VehicleID),
    FOREIGN KEY (ScheduleID) REFERENCES
    Schedule(ScheduleID),FOREIGN KEY (UserID) REFERENCES
    User(UserID)
);
```

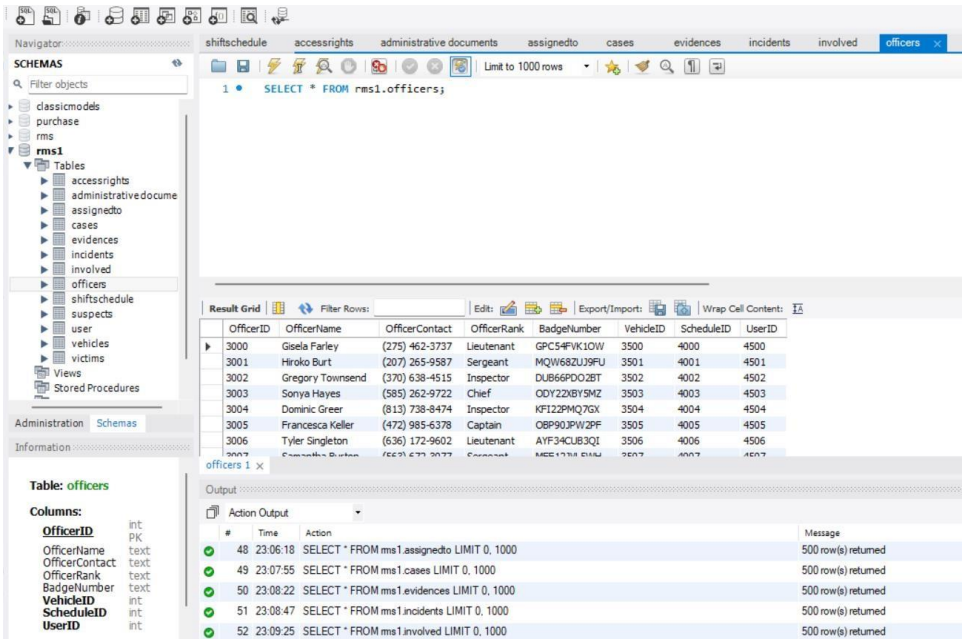


Figure 4.1.8

9.

```
CREATE TABLE
   ShiftSchedule ( ScheduleID
   INT PRIMARY KEY,
   ShiftDates
   DATE,
   ShiftTime TIME
);
```

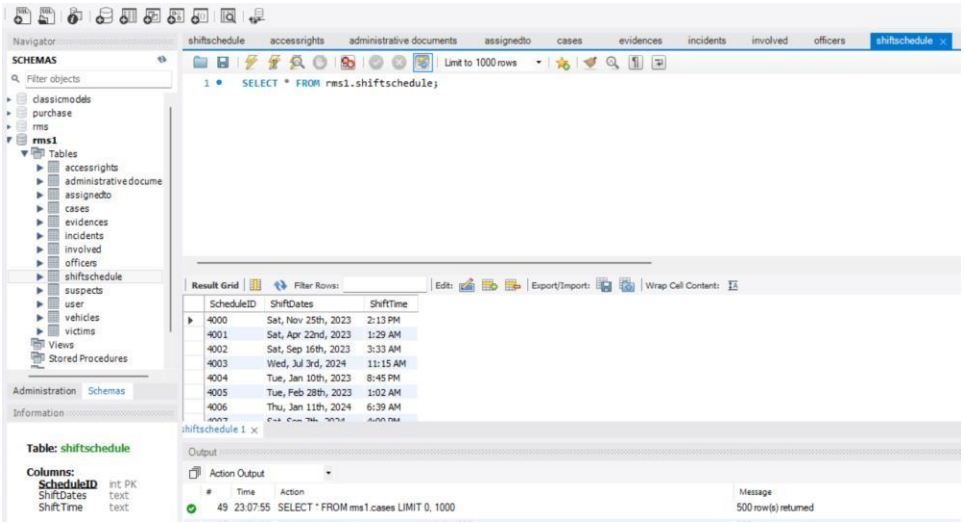


Figure 4.1.9

10.

```
CREATE TABLE suspects (
   SuspectID INT PRIMARY
   KEY,
   SuspectName TEXT,
   SuspectDOB DATE,
   SuspectAddress
   TEXT,
   CriminalRecord
   TEXT
);
```

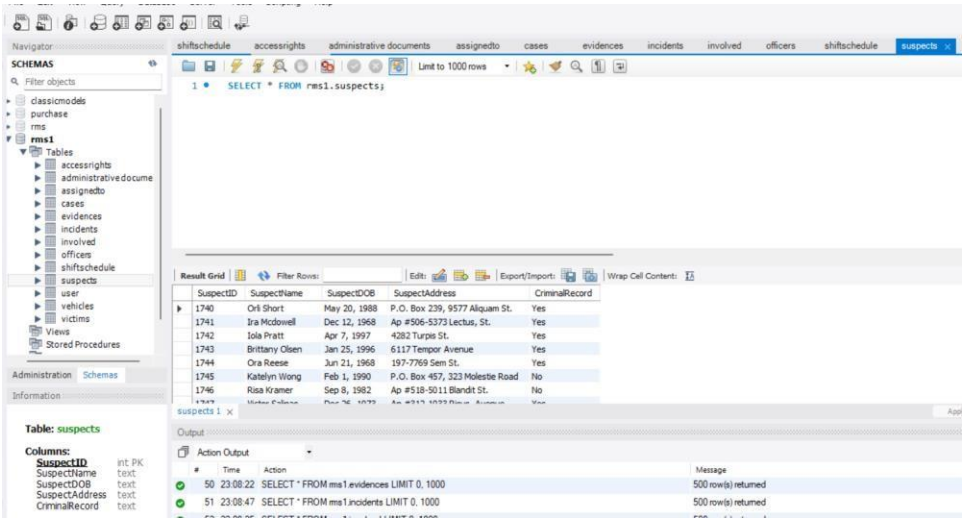


Figure 4.1.10

11.

```
CREATE TABLE Users (
   UserID INT PRIMARY
   KEY,
```

```
    RoleBasedAccess
    TEXT,Password
    TEXT
);
```

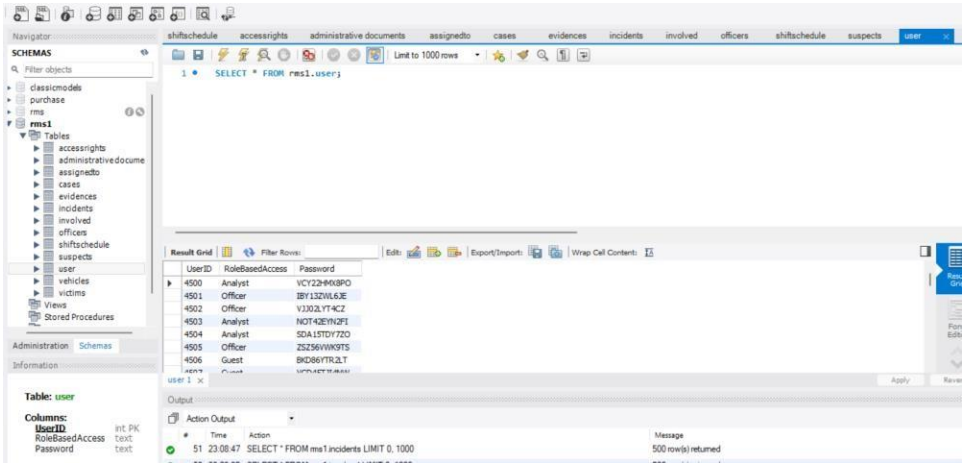


Figure 4.1.11

12.

```
CREATE TABLE Vehicles (
   VehicleID INT PRIMARY
   KEY,
   VehicleModel TEXT,
   LicencePlateNumber
   TEXT,
   MaintenanceHistory
   TEXT
);
```

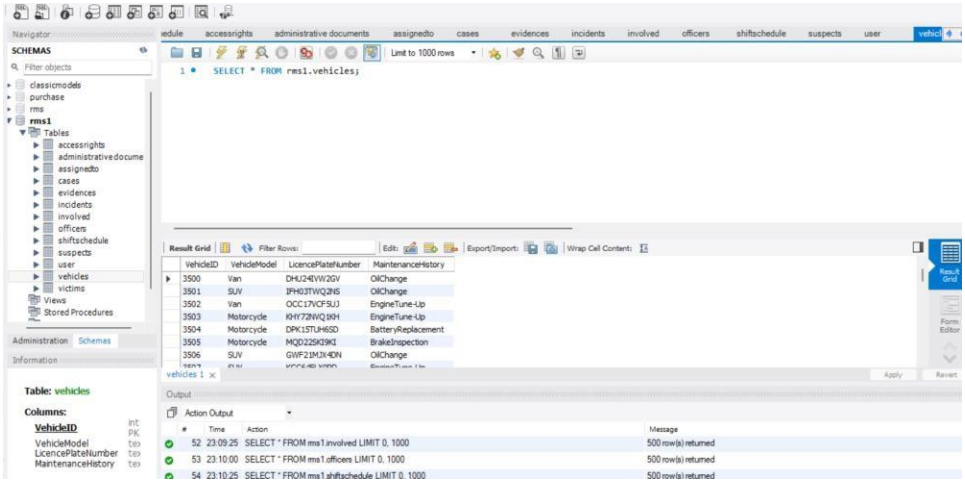


Figure 4.1.12

13.

```
CREATE TABLE Victims (
   VictimID INT PRIMARY
   KEY, VictimName TEXT,
   VictimContact TEXT,
   VictimStatement TEXT,
   CaseID INT NOT NULL,
   FOREIGN KEY (CaseID) REFERENCES Cases(CaseID)
);
```

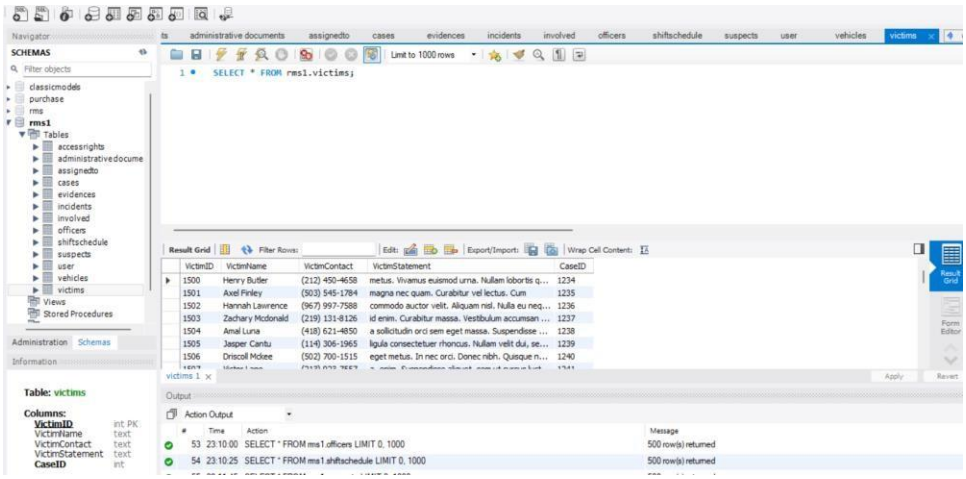


Figure 4.1.13

4.2 Sample Queries

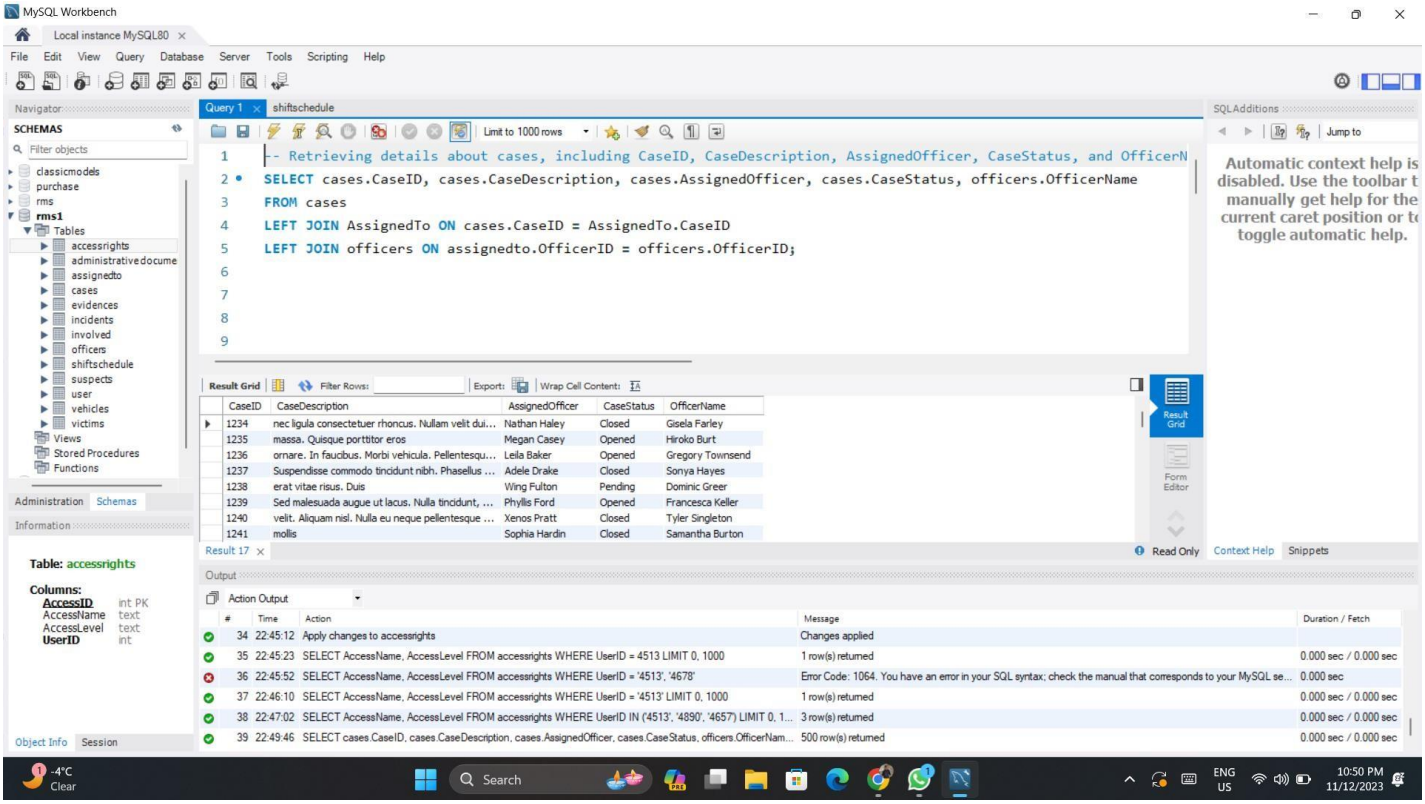


Figure 4.2.1

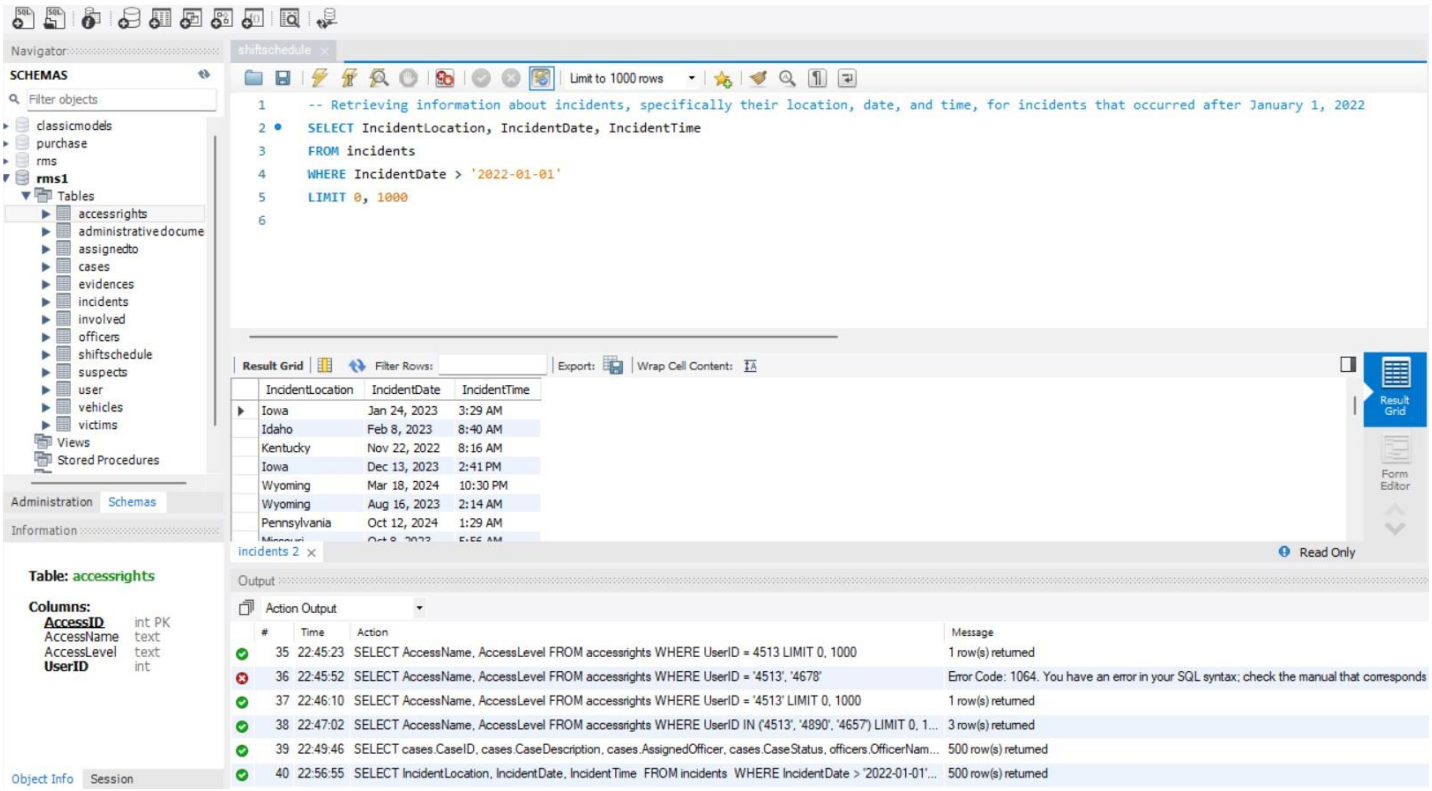

Navigator
SCHEMAS
Filter objects
classicmodels
purchase
rms
rms1
Tables
accessrights
administrativedocume
assignedto
cases
evidences
incidents
involved
officers
shiftschedule
suspects
user
vehicles
victims
Views
Stored Procedures
Administration
Schemas
Information
Table: accessrights
Columns:
AccessID int PK
AccessName text
AccessLevel text
UserID int
Object Info
Session
shiftschedule
Limit to 1000 rows
1 -- Retrieving information about incidents, specifically their location, date, and time, for incidents that occurred after January 1, 2022
2 SELECT IncidentLocation, IncidentDate, IncidentTime
3 FROM incidents
4 WHERE IncidentDate > '2022-01-01'
5 LIMIT 0, 1000
6
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
IncidentLocation IncidentDate IncidentTime
Iowa Jan 24, 2023 3:29 AM
Idaho Feb 8, 2023 8:40 AM
Kentucky Nov 22, 2022 8:16 AM
Iowa Dec 13, 2023 2:41 PM
Wyoming Mar 18, 2024 10:30 PM
Wyoming Aug 16, 2023 2:14 AM
Pennsylvania Oct 12, 2024 1:29 AM
incidents 2
Read Only
Result Grid
Form Editor
Output
Action Output
Time Action Message
35 22:45:23 SELECT AccessName, AccessLevel FROM accessrights WHERE UserID = 4513 LIMIT 0, 1000 1 row(s) returned
36 22:45:52 SELECT AccessName, AccessLevel FROM accessrights WHERE UserID = '4513', '4678' Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds
37 22:46:10 SELECT AccessName, AccessLevel FROM accessrights WHERE UserID = '4513' LIMIT 0, 1000 1 row(s) returned
38 22:47:02 SELECT AccessName, AccessLevel FROM accessrights WHERE UserID IN ('4513', '4890', '4657') LIMIT 0, 1... 3 row(s) returned
39 22:49:46 SELECT cases.CaseID, cases.CaseDescription, cases.AssignedOfficer, cases.CaseStatus, officers.OfficerNam... 500 row(s) returned
40 22:56:55 SELECT IncidentLocation, IncidentDate, IncidentTime FROM incidents WHERE IncidentDate > '2022-01-01'... 500 row(s) returned

Figure 4.2.2

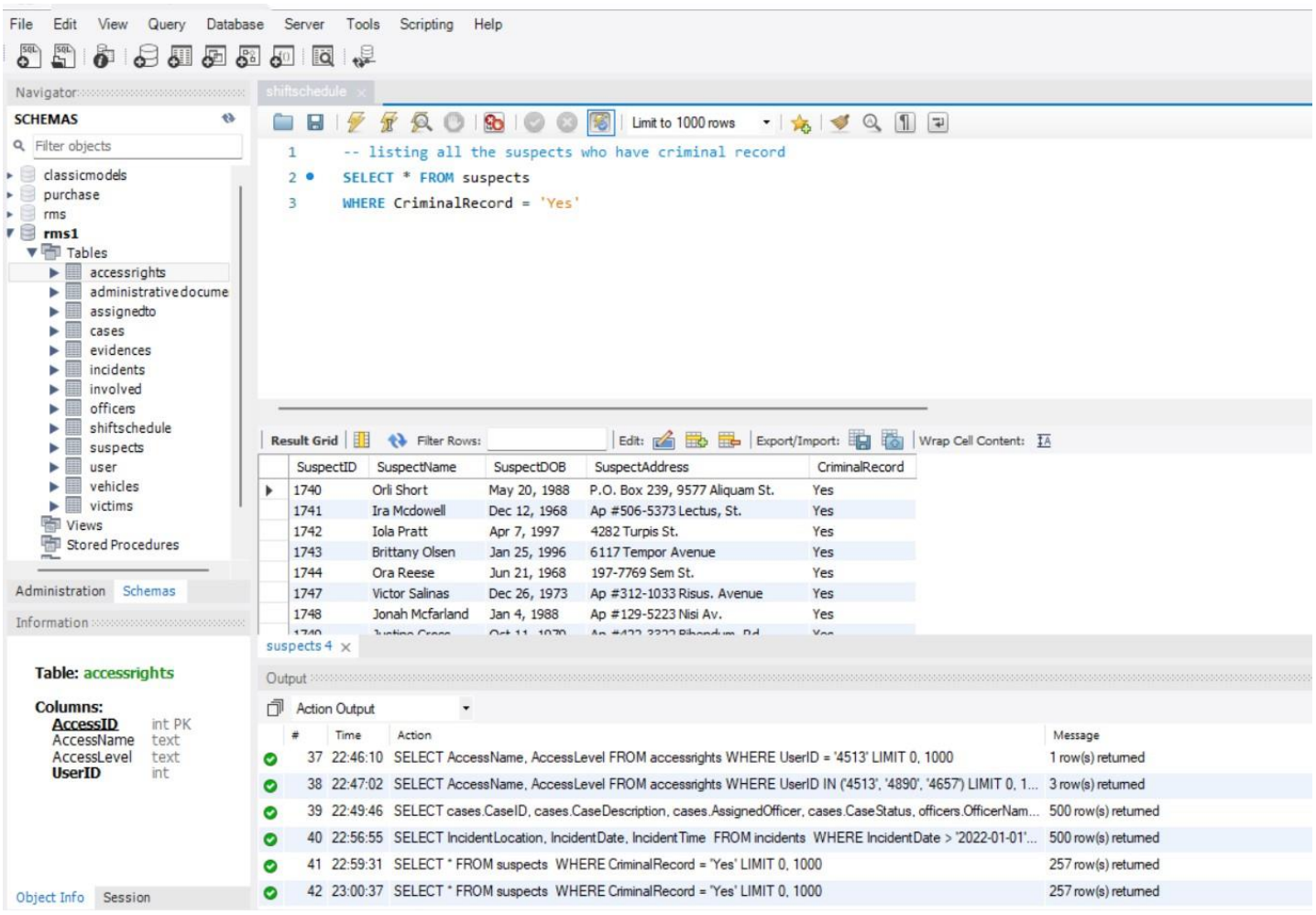

File Edit View Query Database Server Tools Scripting Help
Navigator:
SCHEMAS
Filter objects
classicmodels
purchase
rms
rms1
Tables
accessrights
administrativedocume
assignedto
cases
evidences
incidents
involved
officers
shiftschedule
suspects
user
vehicles
victims
Views
Stored Procedures
Administration Schemas
Information
Table: accessrights
Columns:
AccessID int PK
AccessName text
AccessLevel text
UserID int
Object Info Session
shiftschedule
Limit to 1000 rows
1 -- listing all the suspects who have criminal record
2 SELECT * FROM suspects
3 WHERE CriminalRecord = 'Yes'
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
SuspectID SuspectName SuspectDOB SuspectAddress CriminalRecord
1740 Orli Short May 20, 1988 P.O. Box 239, 9577 Aliquam St. Yes
1741 Ira Mcdowell Dec 12, 1968 Ap #506-5373 Lectus, St. Yes
1742 Iola Pratt Apr 7, 1997 4282 Turpis St. Yes
1743 Brittany Olsen Jan 25, 1996 6117 Tempor Avenue Yes
1744 Ora Reese Jun 21, 1968 197-7769 Sem St. Yes
1747 Victor Salinas Dec 26, 1973 Ap #312-1033 Risus. Avenue Yes
1748 Jonah Mcfarland Jan 4, 1988 Ap #129-5223 Nisi Av. Yes
suspects 4
Output
Action Output
Time Action Message
37 22:46:10 SELECT AccessName, AccessLevel FROM accessrights WHERE UserID = '4513' LIMIT 0, 1000 1 row(s) returned
38 22:47:02 SELECT AccessName, AccessLevel FROM accessrights WHERE UserID IN ('4513', '4890', '4657') LIMIT 0, 1... 3 row(s) returned
39 22:49:46 SELECT cases.CaseID, cases.CaseDescription, cases.AssignedOfficer, cases.CaseStatus, officers.OfficerNam... 500 row(s) returned
40 22:56:55 SELECT IncidentLocation, IncidentDate, IncidentTime FROM incidents WHERE IncidentDate > '2022-01-01'... 500 row(s) returned
41 22:59:31 SELECT * FROM suspects WHERE CriminalRecord = 'Yes' LIMIT 0, 1000 257 row(s) returned
42 23:00:37 SELECT * FROM suspects WHERE CriminalRecord = 'Yes' LIMIT 0, 1000 257 row(s) returned

Figure 4.2.3

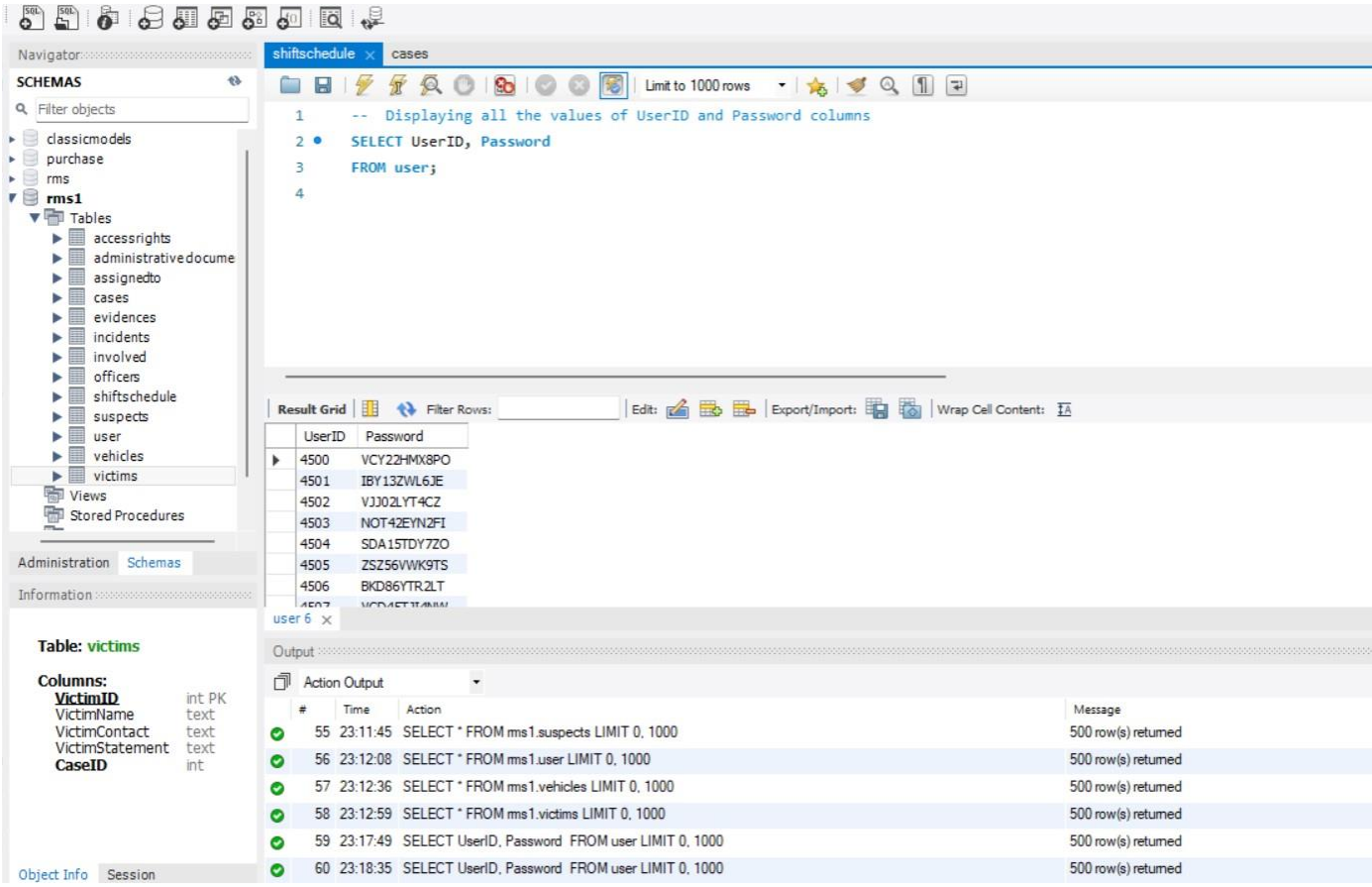

Navigator
SCHEMAS
Filter objects
classicmodels
purchase
rms
rms1
Tables
accessrights
administrativedocume
assignedto
cases
evidences
incidents
involved
officers
shiftschedule
suspects
user
vehicles
victims
Views
Stored Procedures
Administration
Schemas
Information
Table: victims
Columns:
VictimID int PK
VictimName text
VictimContact text
VictimStatement text
CaseID int
Object Info
Session
shiftschedule
cases
Limit to 1000 rows
1 -- Displaying all the values of UserID and Password columns
2 SELECT UserID, Password
3 FROM user;
4
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
UserID Password
4500 VCY22HMX8PO
4501 IBY13ZWL6JE
4502 VJJ02LYT4CZ
4503 NOT42EYN2FI
4504 SDA15TDY7ZO
4505 ZSZ56VWK9TS
4506 BKD86YTR2LT
user 6
Output
Action Output
Time Action Message
55 23:11:45 SELECT * FROM rms1.suspects LIMIT 0, 1000 500 row(s) returned
56 23:12:08 SELECT * FROM rms1.user LIMIT 0, 1000 500 row(s) returned
57 23:12:36 SELECT * FROM rms1.vehicles LIMIT 0, 1000 500 row(s) returned
58 23:12:59 SELECT * FROM rms1.victims LIMIT 0, 1000 500 row(s) returned
59 23:17:49 SELECT UserID, Password FROM user LIMIT 0, 1000 500 row(s) returned
60 23:18:35 SELECT UserID, Password FROM user LIMIT 0, 1000 500 row(s) returned

Figure 4.2.4

Nested query:

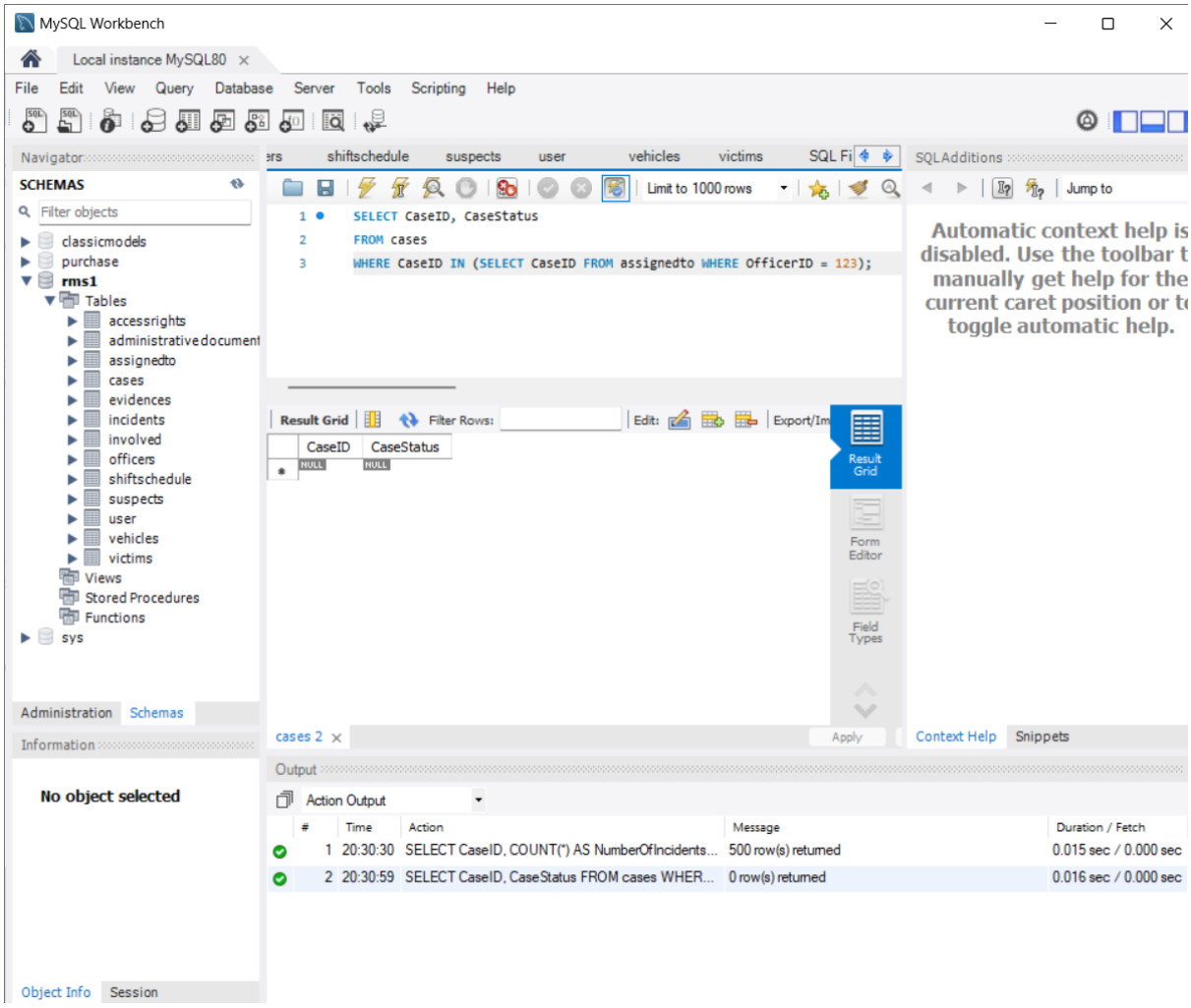


Figure 4.2.5

Correlated Query:

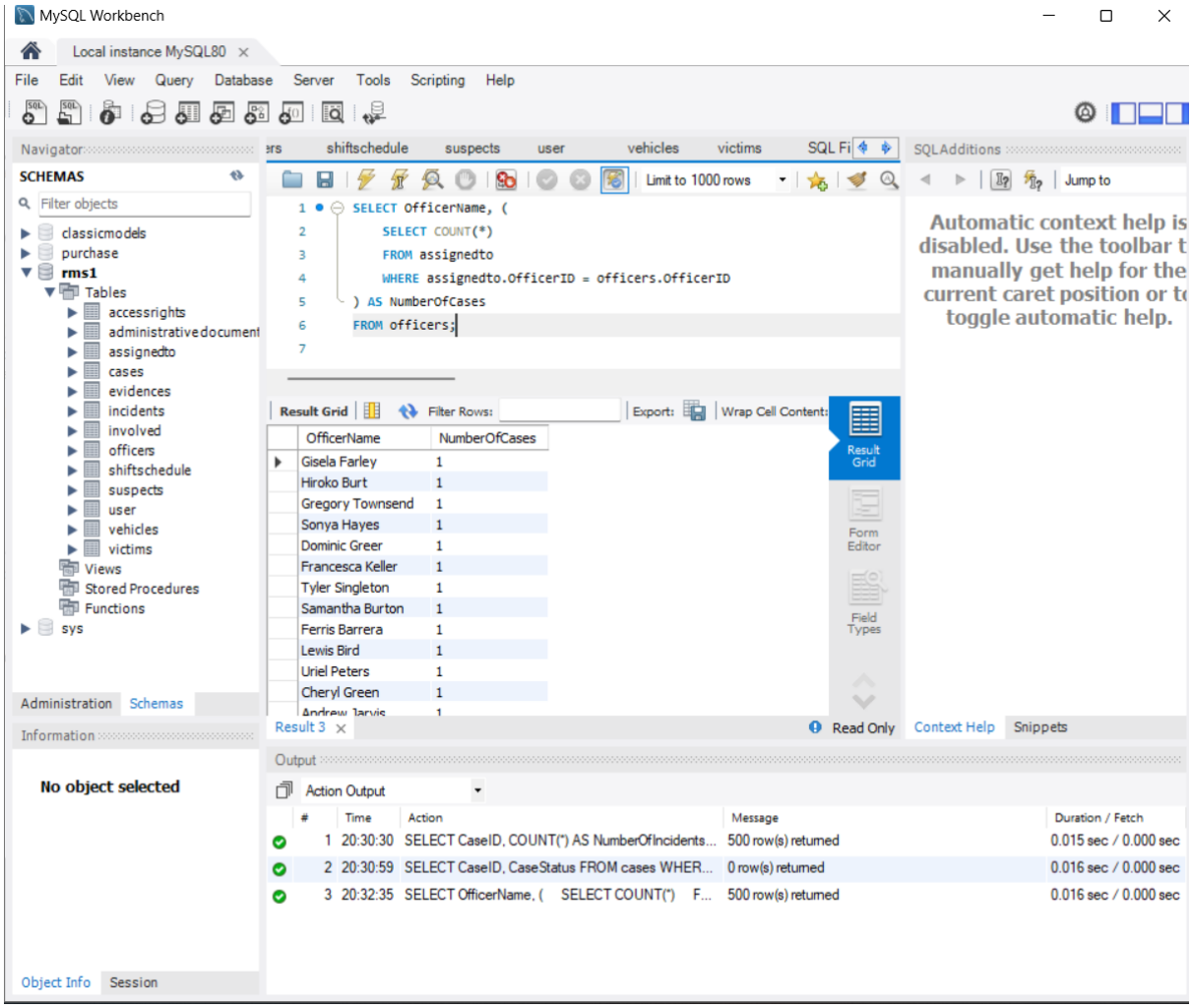


Figure 4.2.6

UNION (Set Operation):

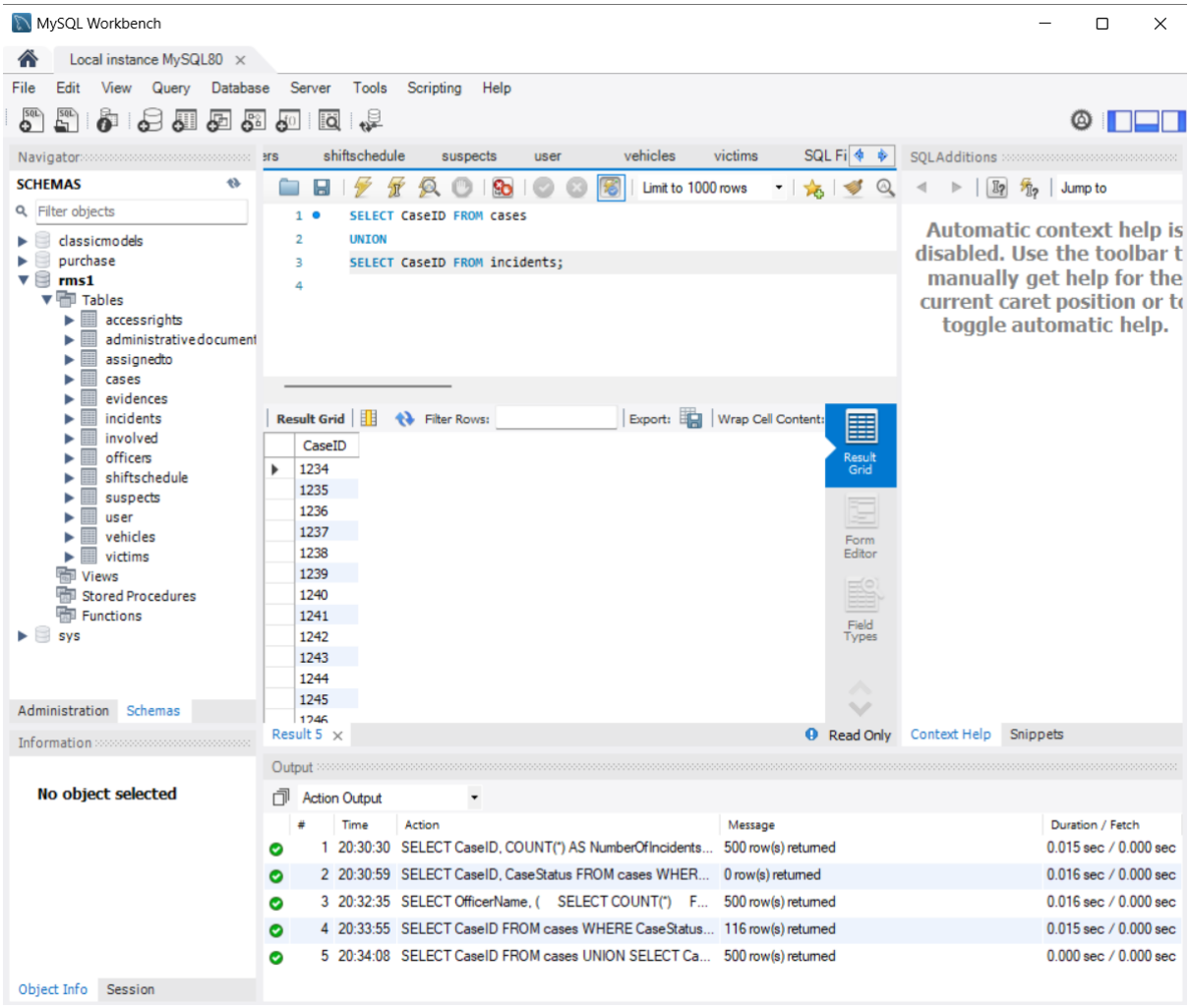


Figure 4.2.7

Subquery in FROM:

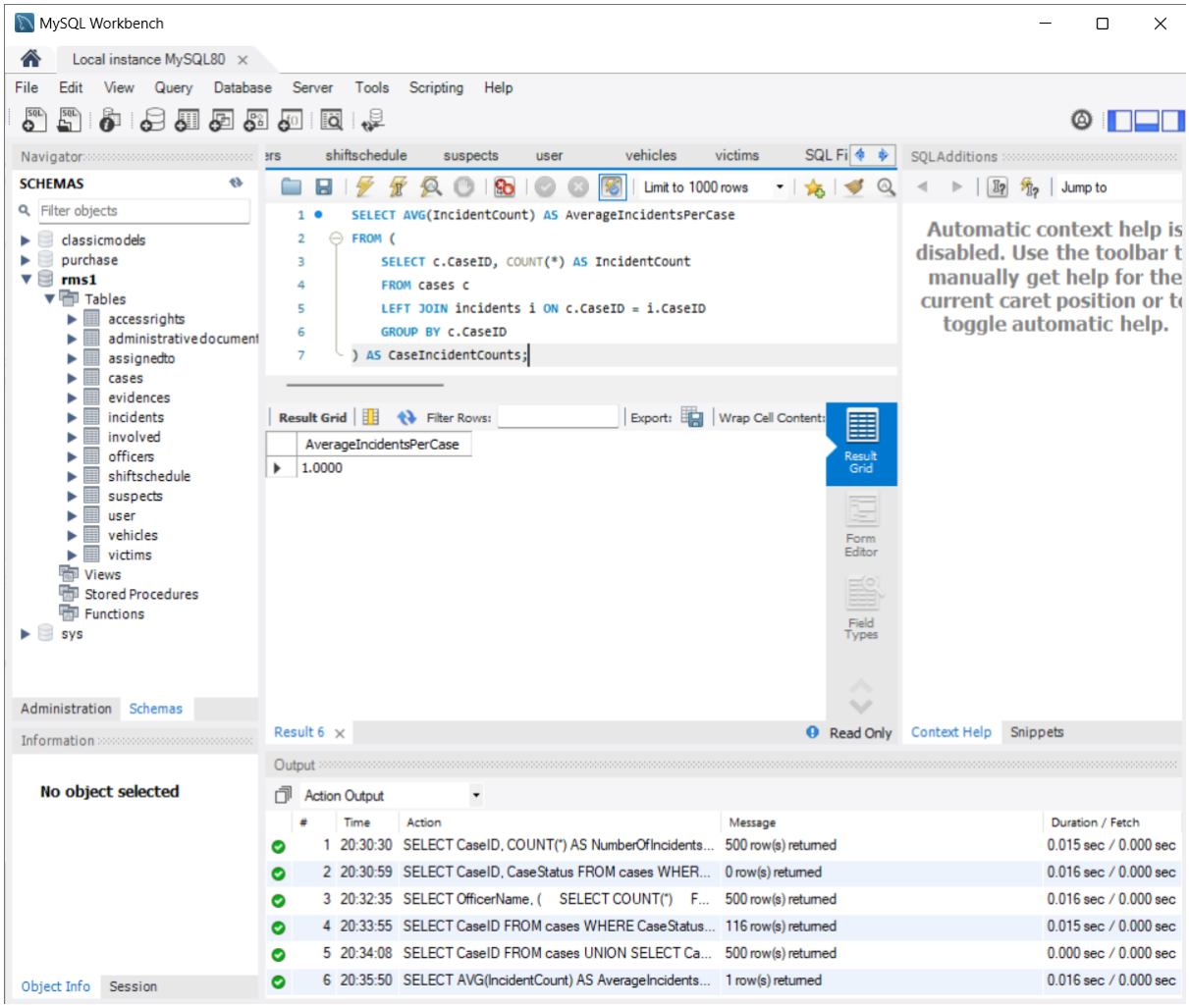


Figure 4.2.8

NOT EXISTS:

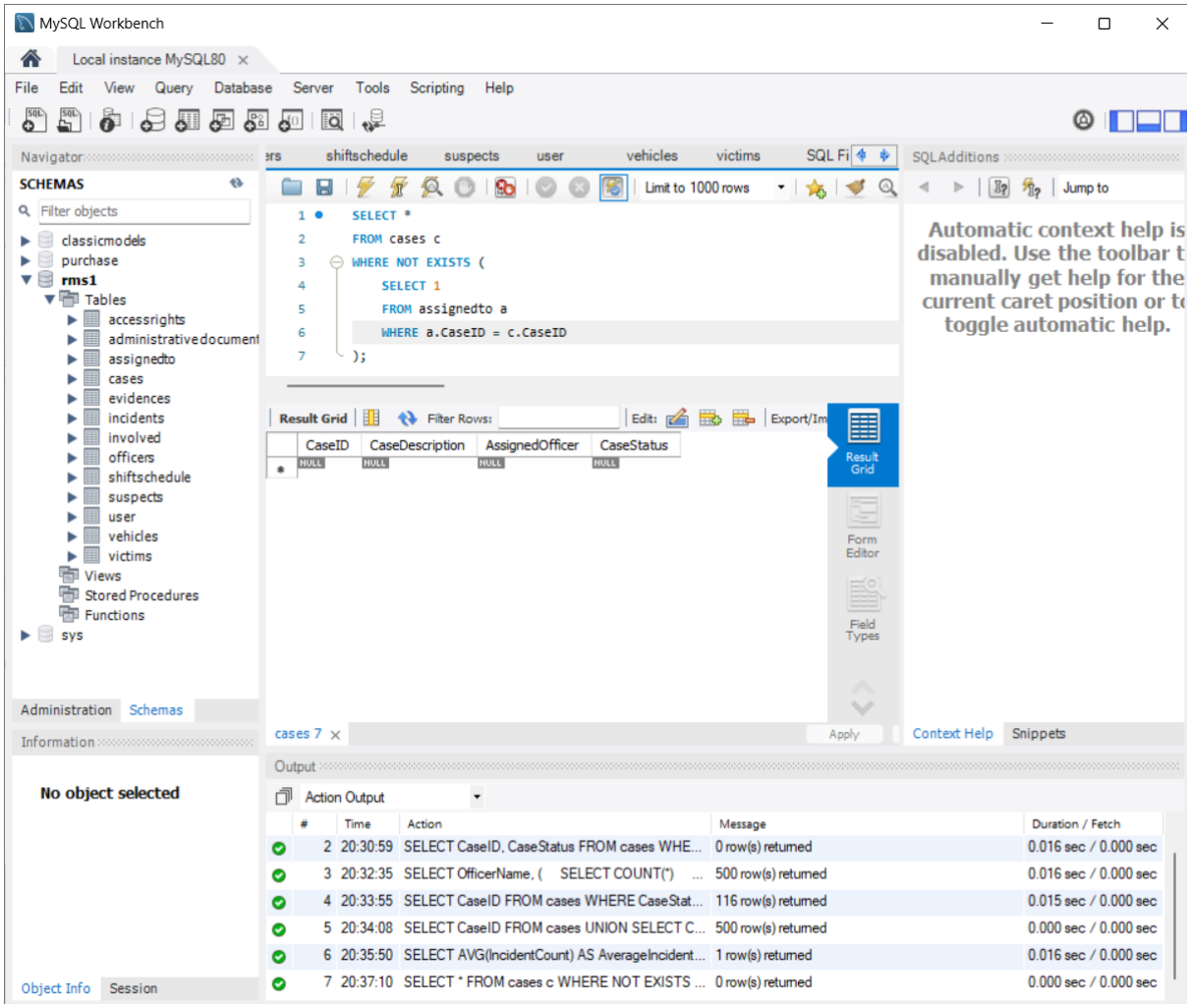
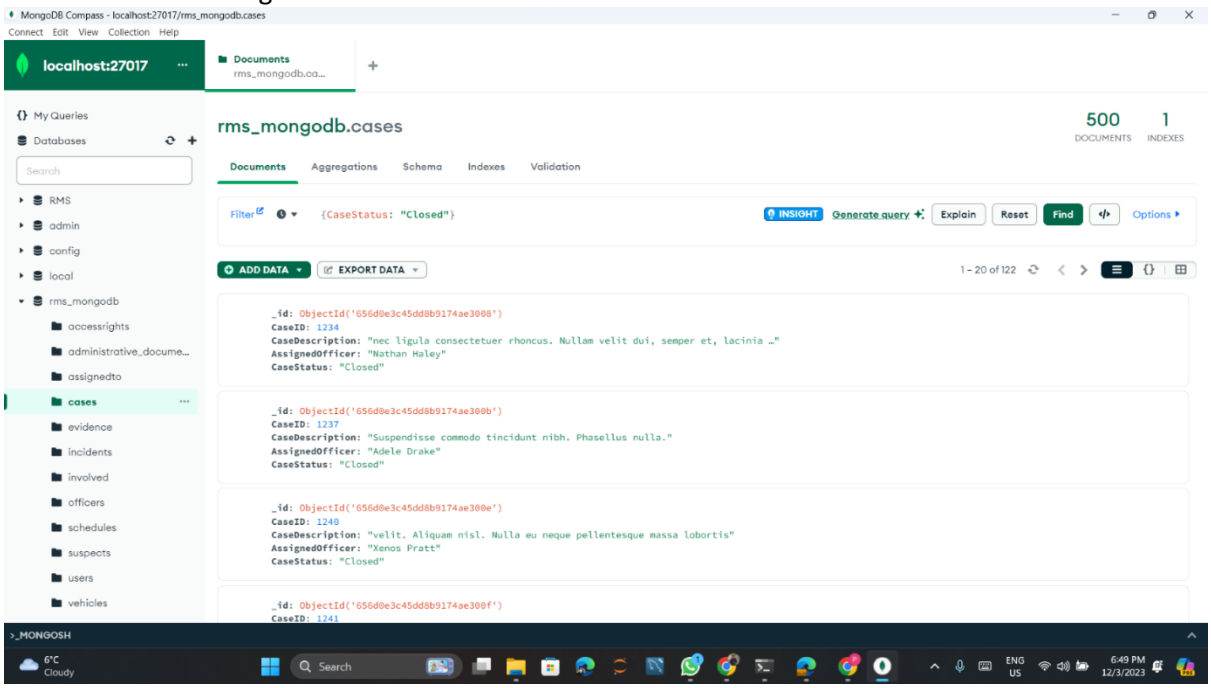


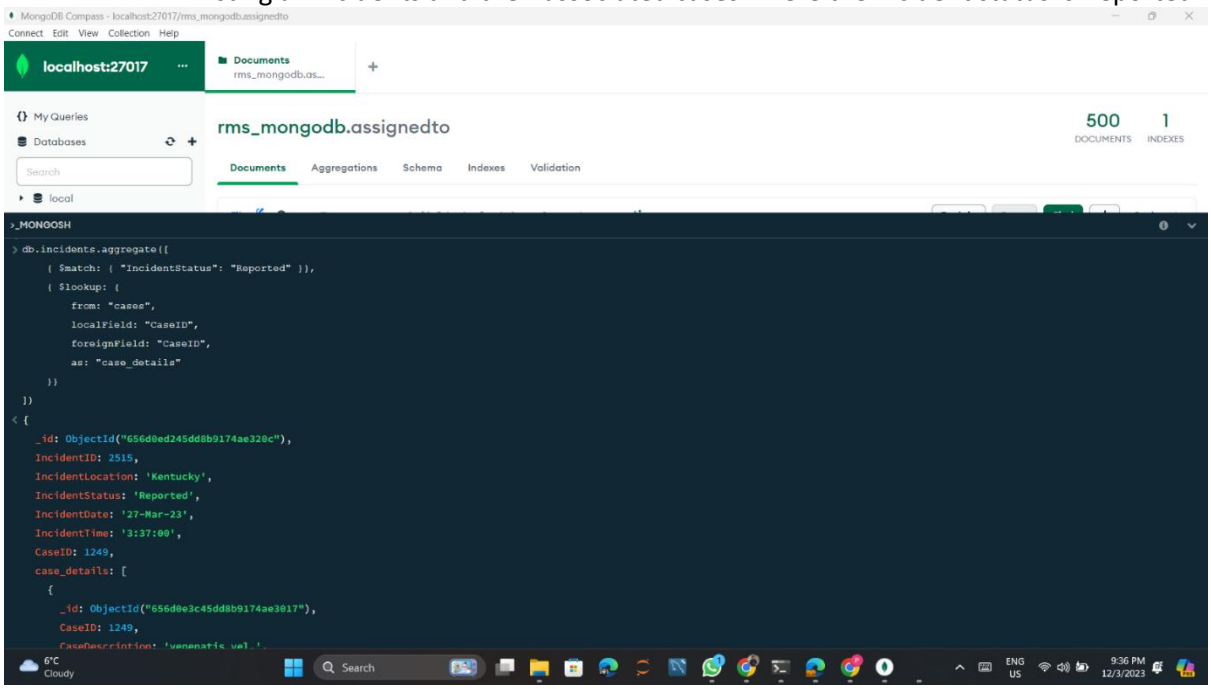
Figure 4.2.9

5. Nosql Implementation

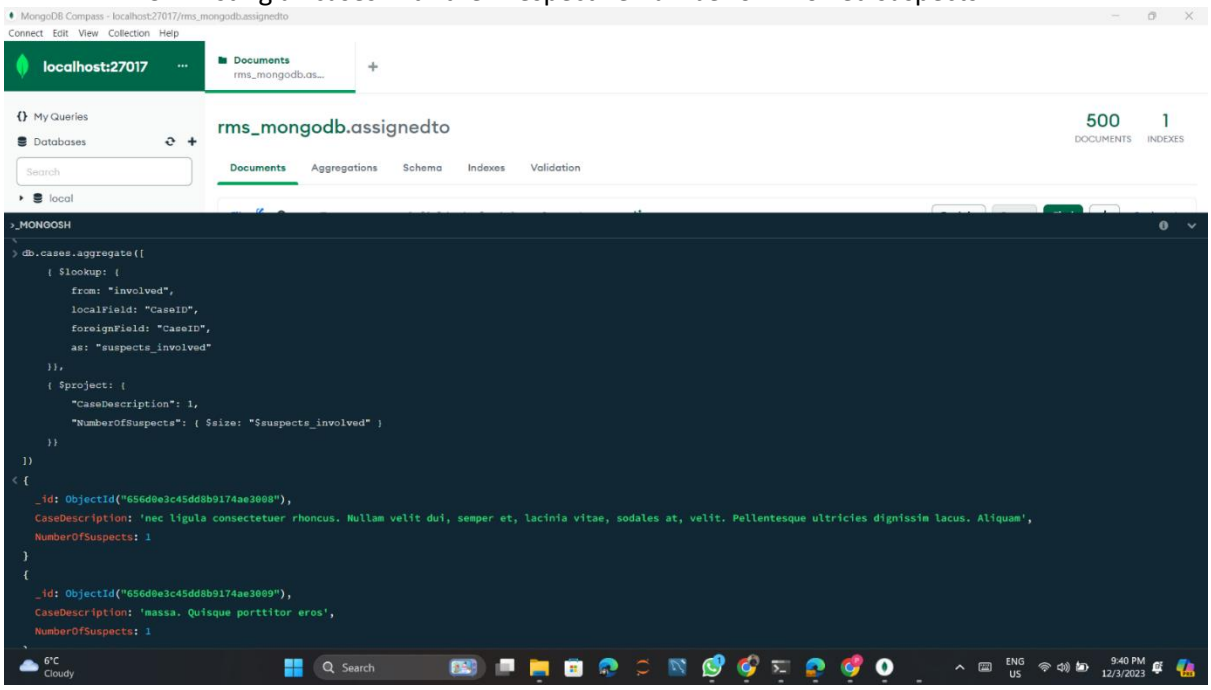
1. Listing all the cases where the case status is “Closed”



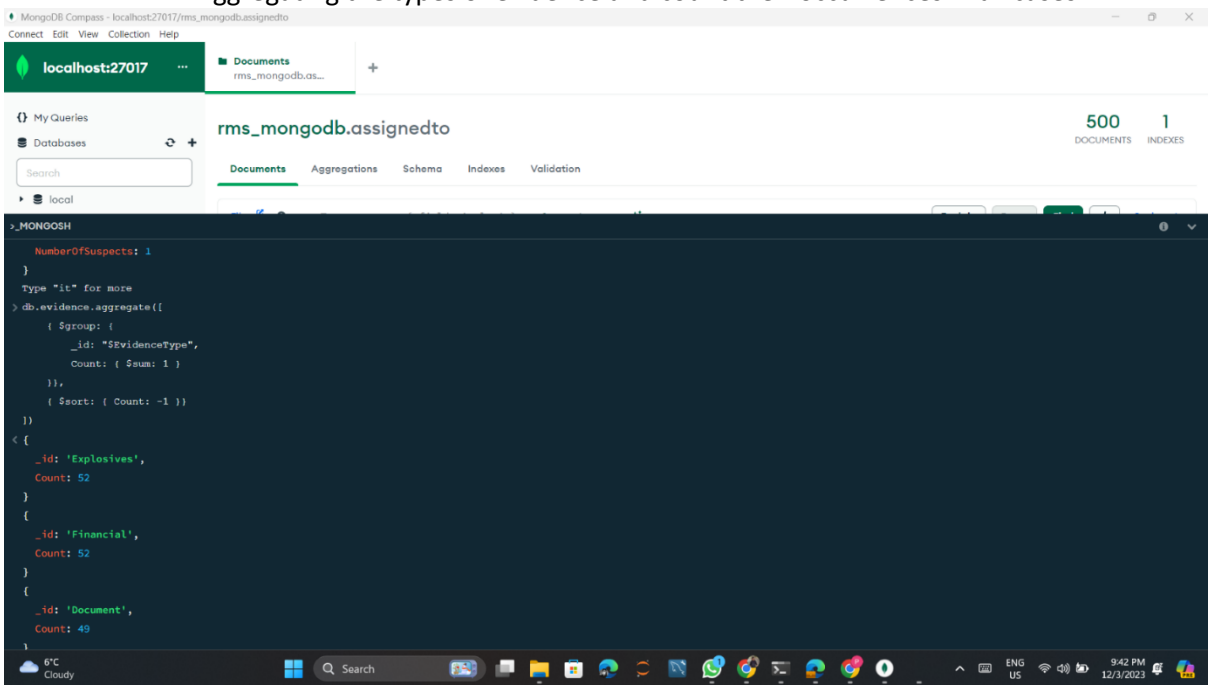
2. Listing all incidents and their associated cases where the incident status is Reported



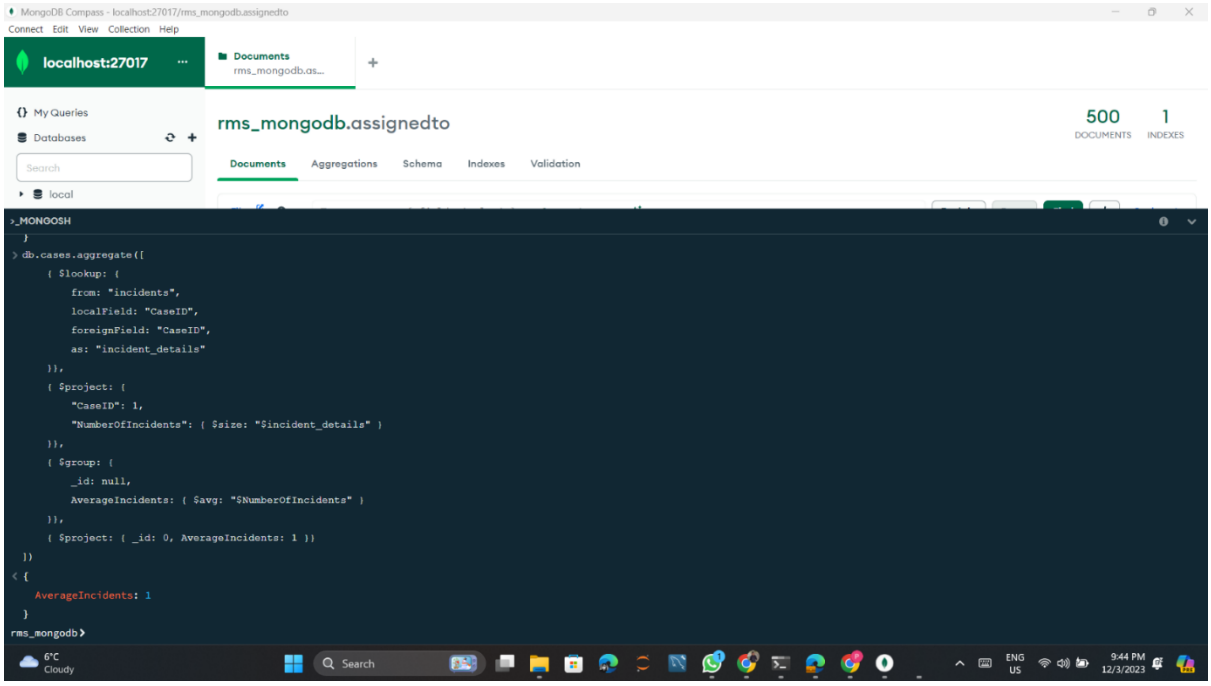
3. Listing all cases with their respective number of involved suspects



4. Aggregating the types of evidence and count their occurrences in all cases



5. Finding the average number of incidents per case



The answer is 1 because we gave unique incidentIDs

6. Application Demonstration

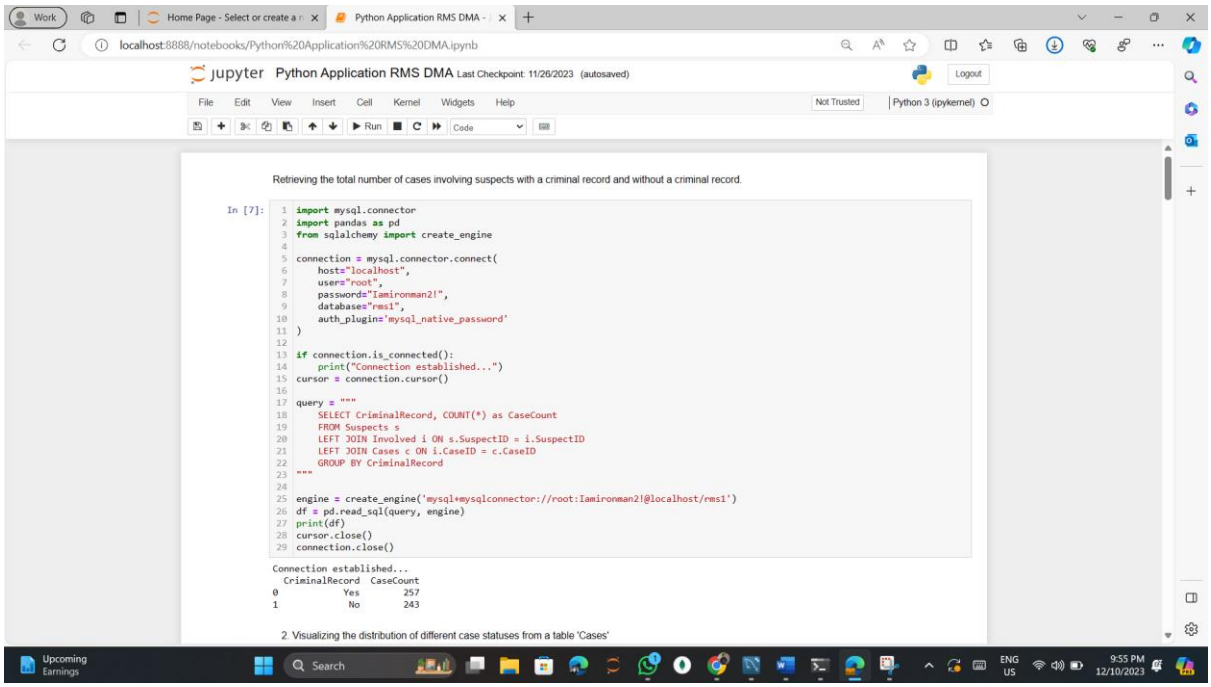


Fig 6.1

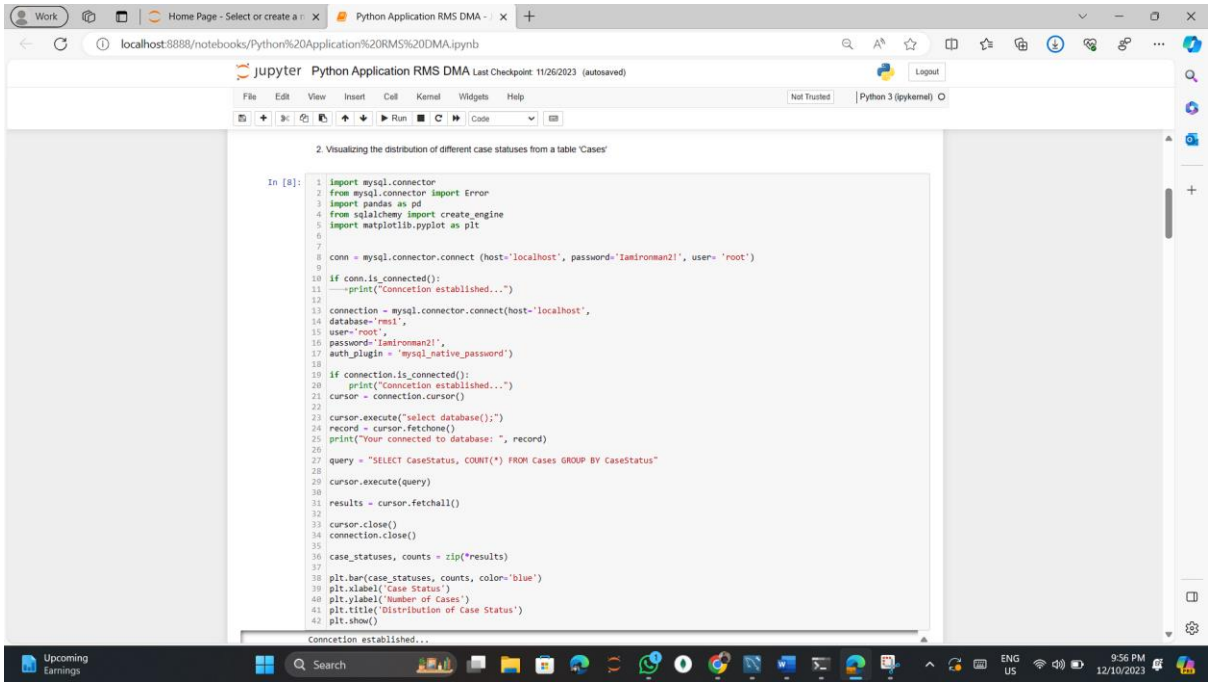


Fig 6.2

```
Conncetion established...
Connected to MySQL Server version  8.0.35
Your connected to database:  ('rms1',)
```

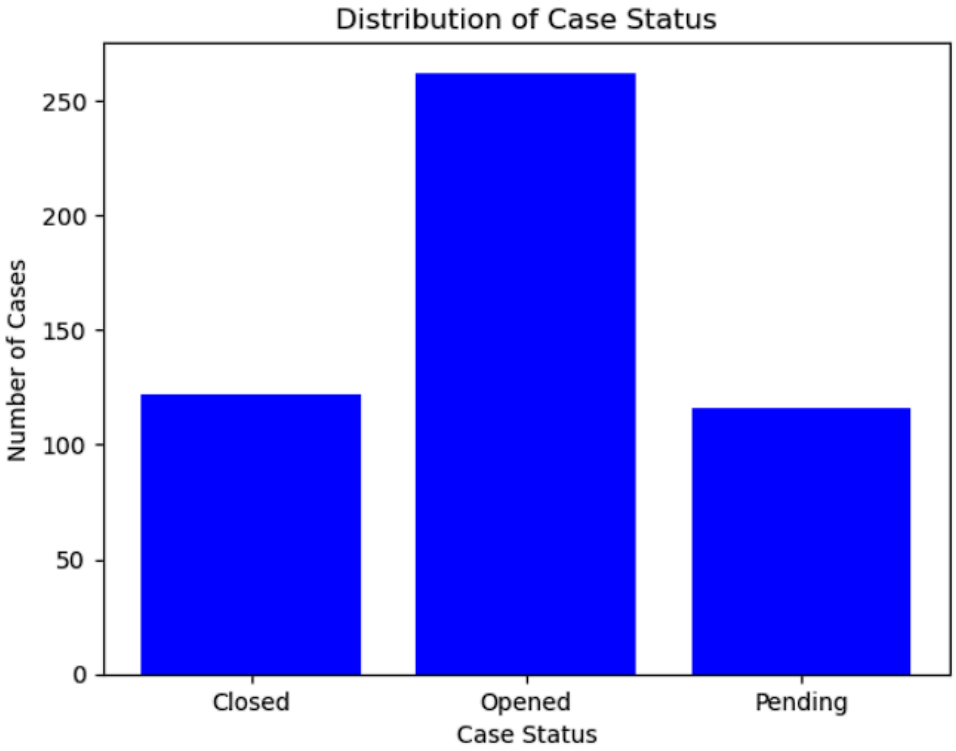


Fig 6.3

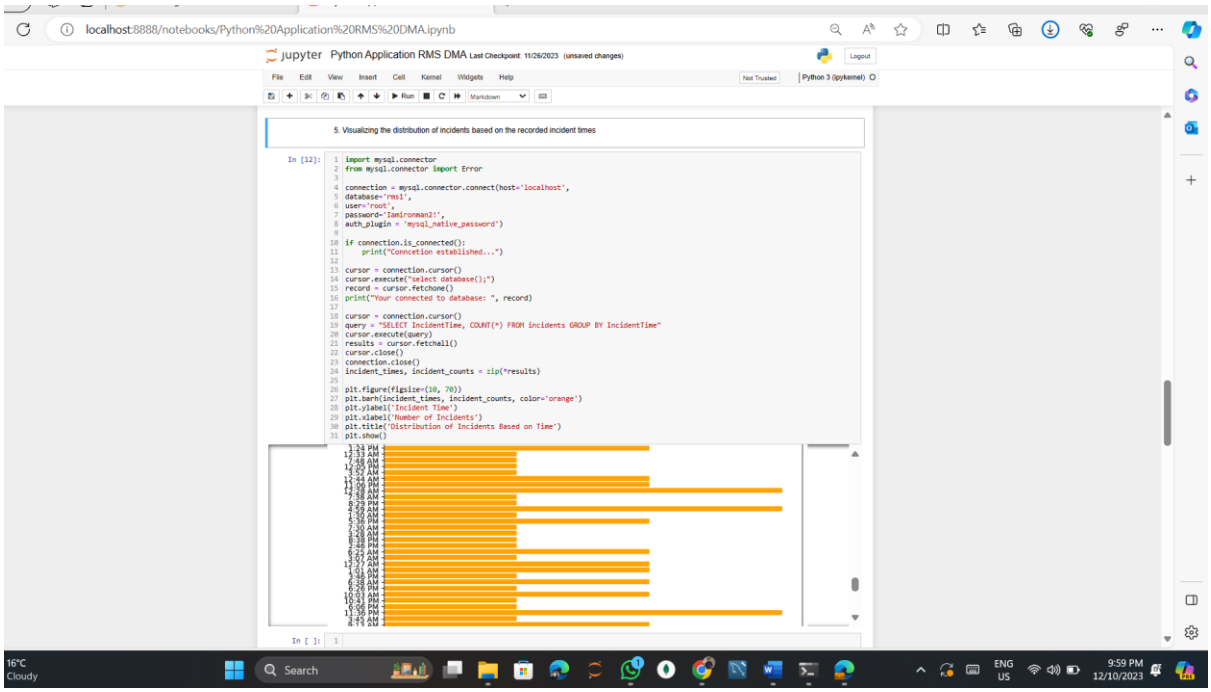


Fig 6.4

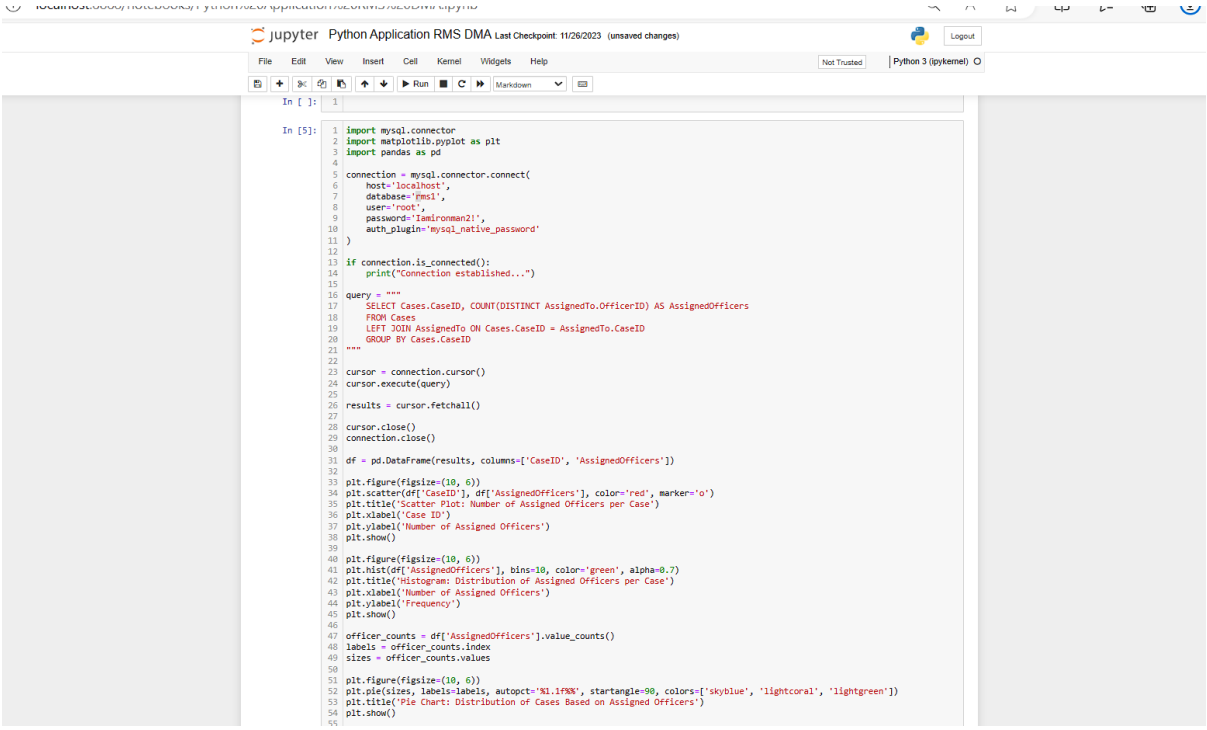

jupyter Python Application RMS DMA Last Checkpoint: 11/26/2023 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
In [ ]:
In [5]:
import mysql.connector
import matplotlib.pyplot as plt
import pandas as pd
connection = mysql.connector.connect(
host='localhost',
database='rms1',
user='root',
password='Iamironman2!',
auth_plugin='mysql_native_password'
)
if connection.is_connected():
print("Connection established...")
query = """
SELECT Cases.CaseID, COUNT(DISTINCT AssignedTo.OfficerID) AS AssignedOfficers
FROM Cases
LEFT JOIN AssignedTo ON Cases.CaseID = AssignedTo.CaseID
GROUP BY Cases.CaseID
"""
cursor = connection.cursor()
cursor.execute(query)
results = cursor.fetchall()
cursor.close()
connection.close()
df = pd.DataFrame(results, columns=['CaseID', 'AssignedOfficers'])
plt.figure(figsize=(10, 6))
plt.scatter(df['CaseID'], df['AssignedOfficers'], color='red', marker='o')
plt.title('Scatter Plot: Number of Assigned Officers per Case')
plt.xlabel('Case ID')
plt.ylabel('Number of Assigned Officers')
plt.show()
plt.figure(figsize=(10, 6))
plt.hist(df['AssignedOfficers'], bins=10, color='green', alpha=0.7)
plt.title('Histogram: Distribution of Assigned Officers per Case')
plt.xlabel('Number of Assigned Officers')
plt.ylabel('Frequency')
plt.show()
officer_counts = df['AssignedOfficers'].value_counts()
labels = officer_counts.index
sizes = officer_counts.values
plt.figure(figsize=(10, 6))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, colors=['skyblue', 'lightcoral', 'lightgreen'])
plt.title('Pie Chart: Distribution of Cases Based on Assigned Officers')
plt.show()

Fig 6.5

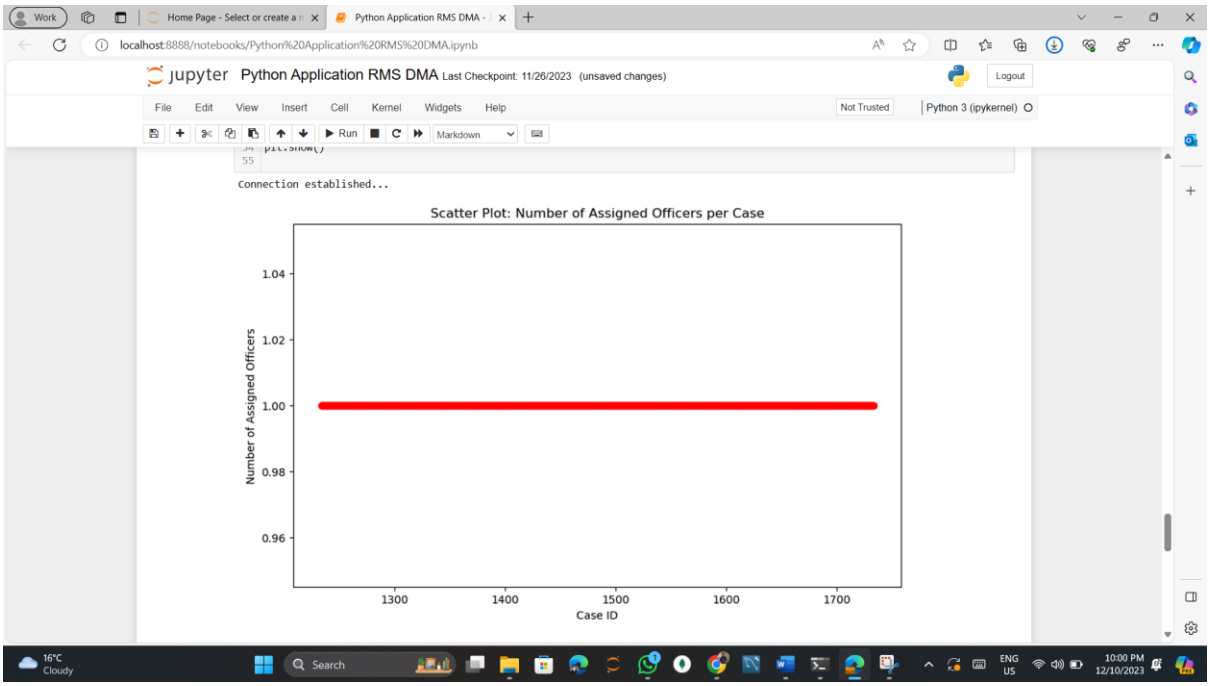

Work
Home Page - Select or create a n
Python Application RMS DMA -
localhost:8888/notebooks/Python%20Application%20RMS%20DMA.ipynb
jupyter Python Application RMS DMA Last Checkpoint: 11/26/2023 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
Connection established...
Scatter Plot: Number of Assigned Officers per Case
Number of Assigned Officers
1.04
1.02
1.00
0.98
0.96
1300
1400
1500
1600
1700
Case ID
16°C
Cloudy
Search
ENG
US
10:00 PM
12/10/2023

Fig 6.6

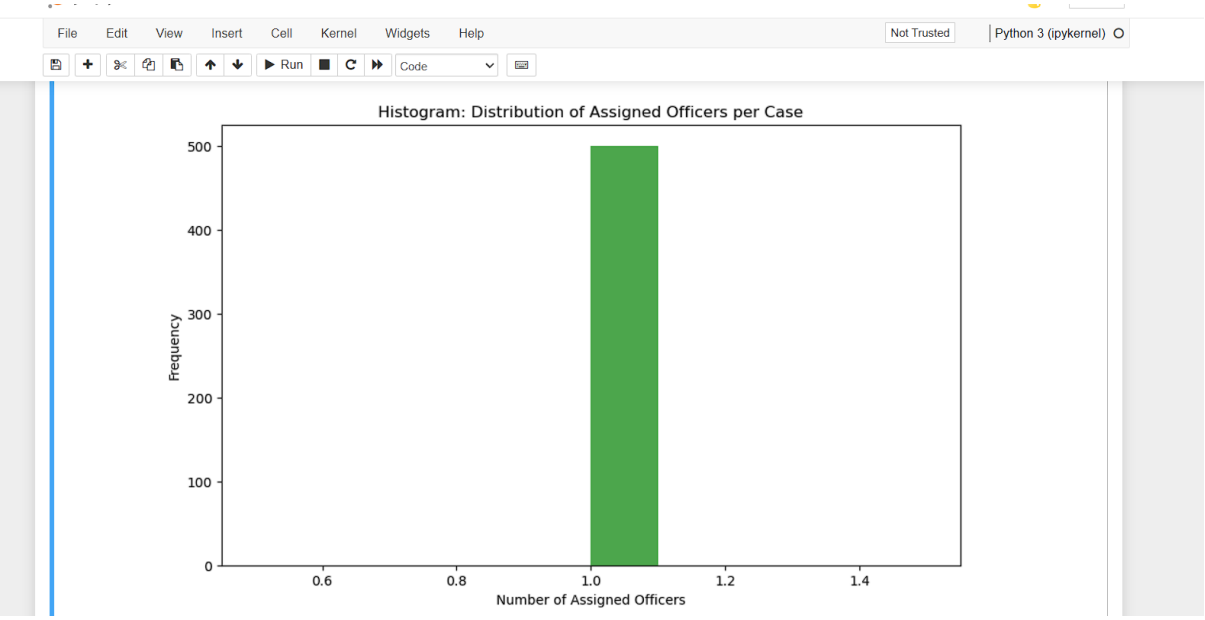

File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Histogram: Distribution of Assigned Officers per Case
Frequency
0
100
200
300
400
500
0.6
0.8
1.0
1.2
1.4
Number of Assigned Officers

Fig 6.7

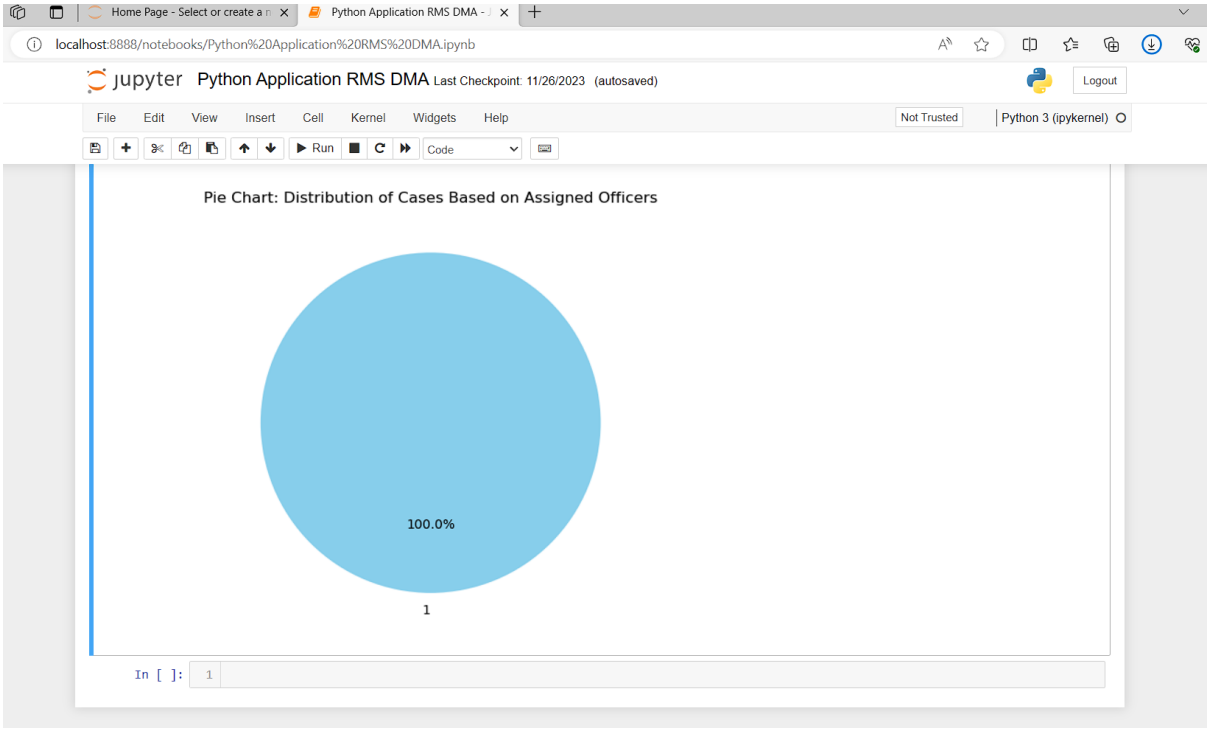

Home Page - Select or create a n
Python Application RMS DMA -
localhost:8888/notebooks/Python%20Application%20RMS%20DMA.ipynb
jupyter Python Application RMS DMA Last Checkpoint: 11/26/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Pie Chart: Distribution of Cases Based on Assigned Officers
100.0%
1
In [ ]:
1

Fig 6.8