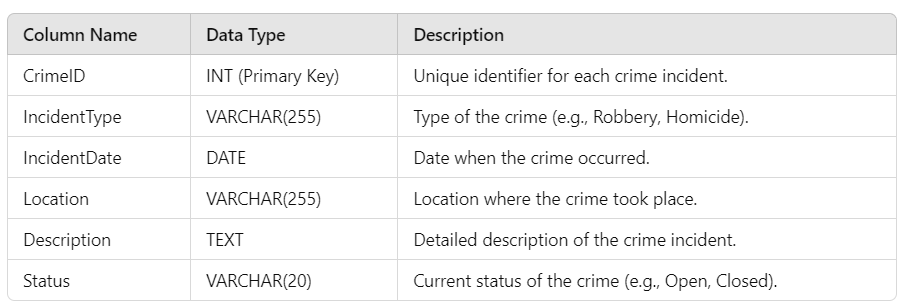
**Coding Challenge-Crime Management System**

**Entities and Relationships**

###  **Crime Table**

* **Description**: This table stores information about each crime incident.
* **Primary Key**: CrimeID – A unique identifier for each crime.



**Victim Table**

* **Description**: This table stores information about the victims involved in the crimes.
* **Primary Key**: VictimID – A unique identifier for each victim.
* **Foreign Key**: CrimeID – Links each victim to a specific crime.

A screenshot of a computer

Description automatically generated

**Suspect Table**

* **Description**: This table stores information about the suspects involved in the crimes.
* **Primary Key**: SuspectID – A unique identifier for each suspect.
* **Foreign Key**: CrimeID – Links each suspect to a specific crime.

A screenshot of a computer

Description automatically generated

**Relationships Between Tables**

**1. Crime and Victim**

* **Relationship**: One-to-Many
  + A single crime can involve multiple victims, but each victim is associated with only one crime.
  + The CrimeID in the **Victim** table is a foreign key that links back to the CrimeID in the **Crime** table.

**2. Crime and Suspect**

* **Relationship**: One-to-Many
  + A single crime can involve multiple suspects, but each suspect is associated with only one crime.
  + The CrimeID in the **Suspect** table is a foreign key that links back to the CrimeID in the **Crime** table.

**3. Victim and Suspect**

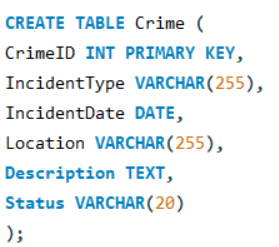
* **Relationship**: No direct relationship in the schema.
  + Victims and suspects are linked to the same crime through the CrimeID, but there is no direct relationship between the two entities.



Created the database



Used the crime\_management database



Created the Crime table to store crime incidents

A screenshot of a computer code

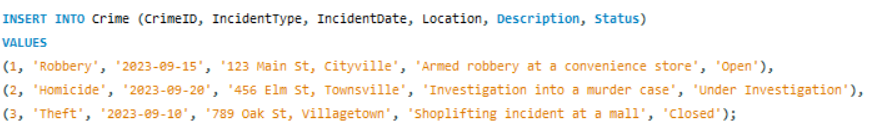
Description automatically generated

Created the Victim table to store victim details for each crime

A computer code with text

Description automatically generated

Created the Suspect table to store suspect details for each crime



Inserted data into the Crime table

A close-up of a computer screen

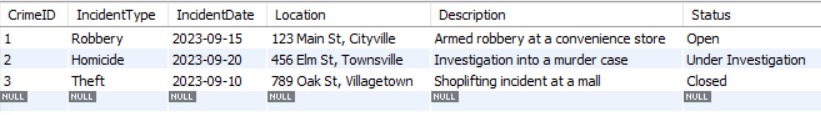
Description automatically generated

Inserted data into the Victim table

A close-up of a computer code

Description automatically generated

Inserted data into the Suspect table



Retrieved all records from the Crime table

A screenshot of a computer

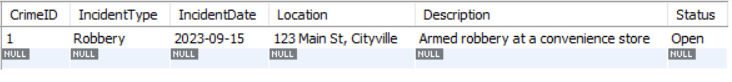
Description automatically generated

Retrieved all records from the Victim table

A screenshot of a computer

Description automatically generated

Retrieved all records from the Suspect table



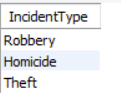
1) Select all open incidents

SELECT \* FROM Crime WHERE Status = 'Open';



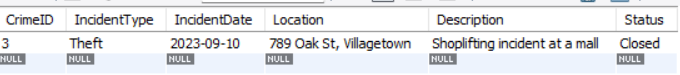
2) Find the total number of incidents

SELECT COUNT(\*) AS TotalIncidents FROM Crime;



3) List all unique incident types:

SELECT DISTINCT IncidentType FROM Crime;



4) Retrieve incidents that occurred between '2023-09-01' and '2023-09-10'

SELECT \* FROM Crime WHERE IncidentDate BETWEEN '2023-09-01' AND '2023-09-10';

5) List persons involved in incidents in descending order of age: (Assuming you have an age column in the victims table)

SELECT Name, Age FROM Victim

JOIN Crime ON Victim.CrimeID = Crime.CrimeID

UNION ALL

SELECT Name, Age FROM Suspect

JOIN Crime ON Suspect.CrimeID = Crime.CrimeID

ORDER BY Age DESC;

6) Find the average age of persons involved in incidents

SELECT AVG(Age) AS AverageAge FROM (

SELECT Age FROM Victim

UNION ALL

SELECT Age FROM Suspect);



7) List incident types and their counts, only for open cases

SELECT IncidentType, COUNT(\*) AS IncidentCount

FROM Crime

WHERE Status = 'Open'

GROUP BY IncidentType;

A screenshot of a computer

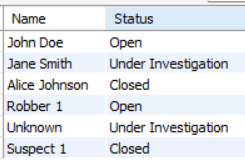
Description automatically generated

8) Find persons with names containing 'Doe':

SELECT Name FROM Victim WHERE Name LIKE '%Doe%'

UNION ALL

SELECT Name FROM Suspect WHERE Name LIKE '%Doe%';



9) Retrieve the names of persons involved in open cases and closed cases

SELECT v.Name, c.Status

FROM Victim v

JOIN Crime c ON v.CrimeID = c.CrimeID

UNION ALL

SELECT s.Name, c.Status

FROM Suspect s

JOIN Crime c ON s.CrimeID = c.CrimeID;

10) List incident types where there are persons aged 30 or 35 involved

SELECT DISTINCT c.IncidentType

FROM Crime c

LEFT JOIN Victim v ON c.CrimeID = v.CrimeID

LEFT JOIN Suspect s ON c.CrimeID = s.CrimeID

WHERE v.Age IN (30, 35) OR s.Age IN (30, 35);

A screenshot of a computer

Description automatically generated

11) Find persons involved in incidents of the same type as 'Robbery'

SELECT v.Name

FROM Victim v

JOIN Crime c ON v.CrimeID = c.CrimeID

WHERE c.IncidentType = 'Robbery'

UNION ALL

SELECT s.Name

FROM Suspect s

JOIN Crime c ON s.CrimeID = c.CrimeID

WHERE c.IncidentType = 'Robbery';

12) List incident types with more than one open case

SELECT IncidentType, COUNT(\*) AS CaseCount

FROM Crime

WHERE Status = 'Open'

GROUP BY IncidentType

HAVING COUNT(\*) > 1;

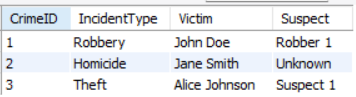
13) List all incidents with suspects whose names also appear as victims in other incidents

SELECT DISTINCT c.CrimeID, c.IncidentType, s.Name AS Suspect

FROM Crime c

JOIN Suspect s ON c.CrimeID = s.CrimeID

WHERE s.Name IN (SELECT Name FROM Victim);



14) Find the average age of persons involved in incidents

SELECT c.CrimeID, c.IncidentType, v.Name AS Victim, s.Name AS Suspect

FROM Crime c

LEFT JOIN Victim v ON c.CrimeID = v.CrimeID

LEFT JOIN Suspect s ON c.CrimeID = s.CrimeID;

15) Find incidents where the suspect is older than any victim

SELECT DISTINCT c.CrimeID, c.IncidentType

FROM Crime c

JOIN Suspect s ON c.CrimeID = s.CrimeID

WHERE s.Age > (SELECT MAX(Age) FROM Victim WHERE CrimeID = c.CrimeID);

16) Find suspects involved in multiple incidents

SELECT Name, COUNT(CrimeID) AS IncidentCount

FROM Suspect

GROUP BY Name

HAVING COUNT(CrimeID) > 1;

17) List incidents with no suspects involved

SELECT CrimeID, IncidentType FROM Crime

WHERE CrimeID NOT IN (SELECT CrimeID FROM Suspect);

A screenshot of a computer

Description automatically generated

18) List all cases where at least one incident is of type 'Homicide' and all other incidents are of type 'Robbery'

SELECT CrimeID, IncidentType

FROM Crime

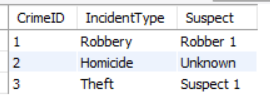
WHERE IncidentType = 'Homicide'

UNION ALL

SELECT CrimeID, IncidentType

FROM Crime

WHERE IncidentType = 'Robbery';



19) Retrieve a list of all incidents and the associated suspects, showing suspects for each incident, or 'No Suspect' if there are none

SELECT c.CrimeID, c.IncidentType, IFNULL(s.Name, 'No Suspect') AS Suspect

FROM Crime c

LEFT JOIN Suspect s ON c.CrimeID = s.CrimeID;



20) List all suspects who have been involved in incidents with incident types 'Robbery' or 'Assault'

SELECT DISTINCT s.Name

FROM Suspect s

JOIN Crime c ON s.CrimeID = c.CrimeID

WHERE c.IncidentType IN ('Robbery', 'Assault');