

# CODING CHALLENGES 3

1. Select all open incidents.

The screenshot shows a database query tool interface. The top pane displays a SQL query: `SELECT * FROM Crime WHERE Status = 'Open';`. The bottom pane shows the results in a table format. The table has columns: CrimeID, IncidentType, IncidentDate, Location, Description, and Status. The results show one row: CrimeID 1, IncidentType Robbery, IncidentDate 2023-09-15, Location 123 Main St, Cityville, Description Armed robbery at a convenience store, Status Open.

CrimeID	IncidentType	IncidentDate	Location	Description	Status
1	Robbery	2023-09-15	123 Main St, Cityville	Armed robbery at a convenience store	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'.

The screenshot shows a database query tool interface. At the top, a query editor displays the following SQL query:

```
1 SELECT *
2 FROM Crime
3 WHERE Status = 'Open';
4
```

Below the query editor, a toolbar includes options like 'Limit to 1000 rows', 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Content'. The 'Result Grid' tab is active, showing a table with the following data:

CrimeID	IncidentType	IncidentDate	Location	Description	Status
1	Robbery	2023-09-15	123 Main St, Cityville	Armed robbery at a convenience store	Open

At the bottom, the 'Output' pane shows a log of actions and messages:

#	Time	Action	Message
9	22:37:37	INSERT INTO Victim (VictimID, CrimeID, Name, ContactInfo, Injuries) VALUES (1, 1, 'John Doe', johndoe@ex...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0
10	22:37:43	INSERT INTO Suspect (SuspectID, CrimeID, Name, Description, CriminalHistory) VALUES (1, 1, 'Robber 1', 'A...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0
11	22:37:52	INSERT INTO Crime (CrimeID, IncidentType, IncidentDate, Location, Description, Status) VALUES (1, 'Robbe...	Error Code: 1062. Duplicate entry '1' for key 'crime.PRIMARY
12	22:38:01	CREATE TABLE Suspect ( SuspectID INT PRIMARY KEY, CrimeID INT, Name VARCHAR(255), Descriptio...	Error Code: 1050. Table 'suspect' already exists
13	22:38:16	SELECT * FROM Crime WHERE Status = 'Open' LIMIT 0, 1000	1 row(s) returned

5. List persons involved in incidents in descending order of age.

SELECT Name, Age

FROM Victim

ORDER BY Age DESC;

5. Find the average age of persons involved in incidents.

SELECT AVG(Age) AS AverageAge

FROM (

SELECT Age FROM Victim

UNION

SELECT Age FROM Suspect

) AS CombinedAges;

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

```
SELECT IncidentType, COUNT(*) AS IncidentCount  
FROM Crime  
WHERE Status = 'Open'  
GROUP BY IncidentType;
```

8. Find persons with names containing 'Doe'.

The screenshot shows a database management tool interface. The top menu bar includes 'File', 'Server', 'Tools', 'Scripting', and 'Help'. Below the menu is a toolbar with various icons. The main area displays a SQL query in a text editor, labeled 'Query 1'. The query is as follows:

```
1 SELECT *  
2 FROM Victim  
3 WHERE Name LIKE '%Doe%'  
4 UNION  
5 SELECT *  
6 FROM Suspect  
7 WHERE Name LIKE '%Doe%';  
8
```

Below the query editor, there is a 'Result Grid' section. It shows a table with the following data:

VictimID	CrimeID	Name	ContactInfo	Injuries
1	1	John Doe	johndoe@example.com	Minor injuries

At the bottom of the screenshot, there is an 'Action Output' section. It shows a log of database actions and their results:

#	Time	Action	Message
12	22:38:01	CREATE TABLE Suspect ( SuspectID INT PRIMARY KEY, CrimeID INT, Name VARCHAR(255), Descriptio...	Error Code: 1050. Table '...
13	22:38:16	SELECT * FROM Crime WHERE Status = 'Open' LIMIT 0, 1000	1 row(s) returned
14	22:41:59	SELECT COUNT(*) AS TotalIncidents FROM Crime LIMIT 0, 1000	1 row(s) returned
15	22:49:46	SELECT * FROM Crime WHERE IncidentDate BETWEEN '2023-09-01' AND '2023-09-10' LIMIT 0, 1000	1 row(s) returned
16	22:54:22	SELECT * FROM Victim WHERE Name LIKE '%Doe%' UNION SELECT * FROM Suspect WHERE Name LIK...	1 row(s) returned

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

```
SELECT Name  
FROM Victim  
WHERE CrimeID IN (SELECT CrimeID FROM Crime WHERE Status = 'Open')  
UNION  
SELECT Name  
FROM Suspect  
WHERE CrimeID IN (SELECT CrimeID FROM Crime WHERE Status = 'Open');
```

```
SELECT Name
FROM Victim
WHERE CrimeID IN (SELECT CrimeID FROM Crime WHERE Status = 'Closed')
UNION
SELECT Name
FROM Suspect
WHERE CrimeID IN (SELECT CrimeID FROM Crime WHERE Status = 'Closed');
```

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

```
SELECT DISTINCT IncidentType
FROM (
    SELECT IncidentType FROM Crime
    WHERE CrimeID IN (SELECT CrimeID FROM Victim WHERE Age IN (30, 35))
    UNION
    SELECT IncidentType FROM Crime
    WHERE CrimeID IN (SELECT CrimeID FROM Suspect WHERE Age IN (30, 35))
) AS IncidentsWithAge;
```

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

```
SELECT Name
FROM Victim
WHERE CrimeID IN (SELECT CrimeID FROM Crime WHERE IncidentType = 'Robbery')
UNION
SELECT Name
FROM Suspect
WHERE CrimeID IN (SELECT CrimeID FROM Crime WHERE IncidentType = 'Robbery');
```

12. List incident types with more than one open case.

```
SELECT IncidentType, COUNT(*) AS OpenCaseCount  
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.*  
FROM Crime c  
JOIN Suspect s ON c.CrimeID = s.CrimeID  
JOIN Victim v ON c.CrimeID = v.CrimeID  
WHERE s.Name = v.Name;
```

14. Retrieve all incidents along with victim and suspect details.

The screenshot displays a database management interface with a SQL query editor and a results grid. The query editor shows a SQL query that joins the Crime, Victim, and Suspect tables. The results grid shows the output of the query, which includes incident details and victim/suspect information.

**Query 1**

```
1 SELECT c.*, v.Name AS VictimName, v.ContactInfo AS VictimContact, v.Injuries,  
2 s.Name AS SuspectName, s.Description AS SuspectDescription, s.CriminalHistory  
3 FROM Crime c  
4 LEFT JOIN Victim v ON c.CrimeID = v.CrimeID  
5 LEFT JOIN Suspect s ON c.CrimeID = s.CrimeID;  
6
```

**Result Grid**

CrimeID	IncidentType	IncidentDate	Location	Description	Status	VictimName	VictimContact
1	Robbery	2023-09-15	123 Main St, Cityville	Armed robbery at a convenience store	Open	John Doe	john.doe@example.com
2	Homicide	2023-09-20	456 Elm St, Townsville	Investigation into a murder case	Under Inve...	Jane Smith	janesmith@example.com
3	Theft	2023-09-10	789 Oak St, Villagetown	Shoplifting incident at a mall	Closed	Alice Johnson	alicejohnson@example.com

**Result 7**

Output

#	Time	Action	Message
15	22:49:46	SELECT * FROM Crime WHERE IncidentDate BETWEEN '2023-09-01' AND '2023-09-10' LIMIT 0, 1000	1 row(s) returned
16	22:54:22	SELECT * FROM Victim WHERE Name LIKE '%Doe%' UNION SELECT * FROM Suspect WHERE Name LIK...	1 row(s) returned
17	23:15:26	SELECT IncidentType, COUNT(*) AS OpenCaseCount FROM Crime WHERE Status = 'Open' GROUP BY Inci...	0 row(s) returned
18	23:15:44	SELECT DISTINCT c.* FROM Crime c JOIN Suspect s ON c.CrimeID = s.CrimeID JOIN Victim v ON c.CrimeID ...	0 row(s) returned
19	23:15:58	SELECT c.*, v.Name AS VictimName, v.ContactInfo AS VictimContact, v.Injuries, s.Name AS SuspectNa...	3 row(s) returned

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

```
SELECT c.*  
FROM Crime c  
JOIN Suspect s ON c.CrimeID = s.CrimeID  
WHERE s.Age > ALL (SELECT Age FROM Victim WHERE CrimeID = c.CrimeID);
```

16. Find suspects involved in multiple incidents:

```
SELECT SuspectID, Name, COUNT(*) AS IncidentCount  
FROM Suspect  
GROUP BY SuspectID, Name  
HAVING COUNT(*) > 1;
```

17. List incidents with no suspects involved.

```
SELECT *  
FROM Crime  
WHERE CrimeID NOT IN (SELECT CrimeID FROM Suspect);
```

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

```
SELECT *  
FROM Crime  
WHERE IncidentType = 'Homicide'  
OR (IncidentType = 'Robbery' AND CrimeID NOT IN (SELECT CrimeID 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.*, COALESCE(s.Name, 'No Suspect') AS SuspectName
```

```
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.*
```

```
FROM Suspect s
```

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

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