Bay City SQL: Mini-Challenges

Here are a series of challenges designed to test your SQL and JDBC skills. Add a new menu option in main for each challenge you implement. Good luck, agent!

Beginner Challenges

Challenge 1: The Informant

Objective: Create a feature to look up a single citizen by their Alias. This is useful for quickly getting intel on a known associate.

SQL Hint: You'll need a SELECT statement with a WHERE clause that filters by the Alias column. Use a PreparedStatement to avoid SQL injection.

```
SELECT Name, Alias, WantedLevel FROM GTA.Citizens WHERE Alias = ?
```

Java Implementation Stub:

```
public static void runInformantCheck() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter the alias of the citizen to look up: ");
    String alias = scanner.nextLine();

    String query = "SELECT Name, Alias, WantedLevel FROM GTA.Citizens WHERE Alias
= ?";

    try (Connection conn = DriverManager.getConnection(DB_URL);
         PreparedStatement stmt = conn.prepareStatement(query)) {

        stmt.setString(1, alias);
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {
            System.out.println("\n--- Citizen Profile ---");
            System.out.printf("Name: %s\nAlias: %s\nWanted Level: %d\n",
                    rs.getString("Name"),
                    rs.getString("Alias"),
                    rs.getInt("WantedLevel"));
        } else {
            System.out.println("\nNo citizen found with that alias.");
        }

    } catch (SQLException e) {
        System.err.println("Error: " + e.getMessage());
    }
}
```

Expected Output (if "Shadow" is entered):

```
--- Citizen Profile ---
Name: Jane Doe
Alias: Shadow
Wanted Level: 3
```

Challenge 2: Vehicle Registry

Objective: List all vehicles of a specific Brand. The user should be able to input a brand name (e.g., "Maibatsu") and see all registered vehicles from that manufacturer.

SQL Hint: This is another SELECT with a WHERE clause, this time on the Vehicles table.

```
SELECT Type, Brand, IsStolen FROM GTA.Vehicles WHERE Brand = ?
```

Java Implementation Stub:

```
public static void runVehicleRegistry() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter vehicle brand to search for: ");
    String brand = scanner.nextLine();

    String query = "SELECT c.Name, v.Type, v.Brand FROM GTA.Vehicles v " +
                   "JOIN GTA.Citizens c ON v.CitizenID = c.ID WHERE v.Brand = ?";

    try (Connection conn = DriverManager.getConnection(DB_URL);
         PreparedStatement stmt = conn.prepareStatement(query)) {
```

```
        stmt.setString(1, brand);
        ResultSet rs = stmt.executeQuery();

        System.out.printf("\n--- Vehicles of brand: %s ---\n", brand);
        while (rs.next()) {
            System.out.printf("Owner: %-20s Type: %-15s Brand: %-15s\n",
                    rs.getString("Name"),
                    rs.getString("Type"),
                    rs.getString("Brand"));
        }
    } catch (SQLException e) {
        System.err.println("Error: " + e.getMessage());
    }
}
```

Intermediate Challenges

Challenge 3: Citizen's Vehicle Fleet

Objective: Allow the user to enter a citizen's Name and see a complete list of all vehicles they own, including whether each vehicle is stolen.

SQL Hint: You'll need to JOIN the Citizens and Vehicles tables and then WHERE filter by the citizen's Name.

```
SELECT v.Type, v.Brand, v.IsStolen
FROM GTA.Vehicles v
JOIN GTA.Citizens c ON v.CitizenID = c.ID
WHERE c.Name = ?
```

Java Implementation Stub:

```
public static void runCitizenFleetReport() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter citizen's name to see their vehicle fleet: ");
    String name = scanner.nextLine();

    String query = "SELECT v.Type, v.Brand, v.IsStolen " +
                "FROM GTA.Vehicles v JOIN GTA.Citizens c ON v.CitizenID = c.ID " +
               "WHERE c.Name = ?";

   try (Connection conn = DriverManager.getConnection(DB_URL);
        PreparedStatement stmt = conn.prepareStatement(query)) {

       stmt.setString(1, name);
       ResultSet rs = stmt.executeQuery();

       System.out.printf("\n--- Vehicle Fleet for %s ---\n", name);
       while (rs.next()) {
           System.out.printf("Type: %-15s Brand: %-15s Stolen: %b\n",
                   rs.getString("Type"),
                   rs.getString("Brand"),
                   rs.getBoolean("IsStolen"));
       }
   } catch (SQLException e) {
       System.err.println("Error: " + e.getMessage());
   }
}
```

Challenge 4: Average Mission Payout

Objective: Calculate and display the average reward for all missions in the database.

SQL Hint: Use the AVG() aggregate function on the Reward column in the Missions table.

```
SELECT AVG(Reward) AS AveragePayout FROM GTA.Missions
```

Java Implementation Stub:

```
public static void runAveragePayout() {
   String query = "SELECT AVG(Reward) AS AveragePayout FROM GTA.Missions";
   try (Connection conn = DriverManager.getConnection(DB_URL);
        PreparedStatement stmt = conn.prepareStatement(query);
        ResultSet rs = stmt.executeQuery()) {

       System.out.println("\n--- Mission Payout Analysis ---");
       if (rs.next()) {
           System.out.printf("The average mission payout is: $%.2f\n",
                   rs.getDouble("AveragePayout"));
       }
   } catch (SQLException e) {
       System.err.println("Error: " + e.getMessage());
   }
}
```

Advanced Challenges

Challenge 5: Agents Without Missions

Objective: Find all citizens who have not completed any missions. These could be new agents or civilians to watch.

SQL Hint: A LEFT JOIN from Citizens to Missions is perfect for this. You can then find rows where the mission information is NULL.

```
SELECT c.Name
FROM GTA.Citizens c
LEFT JOIN GTA.Missions m ON c.ID = m.CitizenID
WHERE m.CitizenID IS NULL
```

Java Implementation Stub:

```
public static void runInactiveAgentReport() {
    String query = "SELECT c.Name FROM GTA.Citizens c " +
                   "LEFT JOIN GTA.Missions m ON c.ID = m.CitizenID " +
                   "WHERE m.CitizenID IS NULL";
    try (Connection conn = DriverManager.getConnection(DB_URL);
         PreparedStatement stmt = conn.prepareStatement(query);
         ResultSet rs = stmt.executeQuery()) {

        System.out.println("\n--- Agents With No Recorded Missions ---");
        while (rs.next()) {
            System.out.printf("Name: %s\n", rs.getString("Name"));
        }
    } catch (SQLException e) {
        System.err.println("Error: " + e.getMessage());
    }
}
```

Challenge 6: Highest Earning Criminals

Objective: Identify the top 3 citizens who have earned the most from missions involving stolen vehicles. This requires joining all three tables.

SQL Hint: This is a complex one. You need to JOIN all three tables, FILTER for missions done by people with stolen cars, GROUP BY the citizen, and then ORDER by their total earnings, taking only the top 3. Many databases support a TOP or LIMIT clause.

```
-- For SQL Server
SELECT TOP 3 c.Name, SUM(m.Reward) AS TotalEarnings
FROM GTA.Citizens c
JOIN GTA.Missions m ON c.ID = m.CitizenID
JOIN GTA.Vehicles v ON c.ID = v.CitizenID
WHERE v.IsStolen = 1
GROUP BY c.Name
ORDER BY TotalEarnings DESC
```

Java Implementation Stub:

```
public static void runHighestEarningCriminals() {
    String query = "SELECT TOP 3 c.Name, SUM(m.Reward) AS TotalEarnings " +
                   "FROM GTA.Citizens c " +
                   "JOIN GTA.Missions m ON c.ID = m.CitizenID " +
                   "JOIN GTA.Vehicles v ON c.ID = v.CitizenID " +
                   "WHERE v.IsStolen = 1 " +
                   "GROUP BY c.Name " +
                   "ORDER BY TotalEarnings DESC";
    try (Connection conn = DriverManager.getConnection(DB_URL);
         PreparedStatement stmt = conn.prepareStatement(query);
         ResultSet rs = stmt.executeQuery()) {

        System.out.println("\n--- Top 3 Highest Earning Criminals (with stolen cars) ---");
        while (rs.next()) {
            System.out.printf("%-20s Total Earnings: $%.2f\n",
                rs.getString("Name"),
                rs.getDouble("TotalEarnings"));
        }
    } catch (SQLException e) {
        System.err.println("Error: " + e.getMessage());
    }
}
```