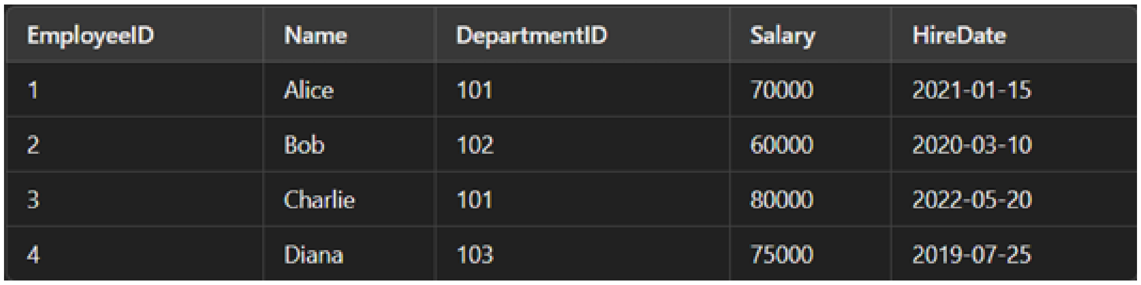
SQLite3 Exercises

Scenario: Employee Management System

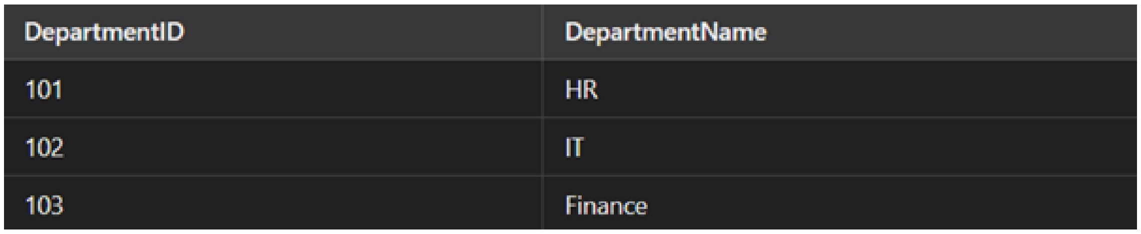
A company maintains a database with two tables:

- **Employees**: Stores information about employees.



EmployeeID	Name	DepartmentID	Salary	HireDate
1	Alice	101	70000	2021-01-15
2	Bob	102	60000	2020-03-10
3	Charlie	101	80000	2022-05-20
4	Diana	103	75000	2019-07-25

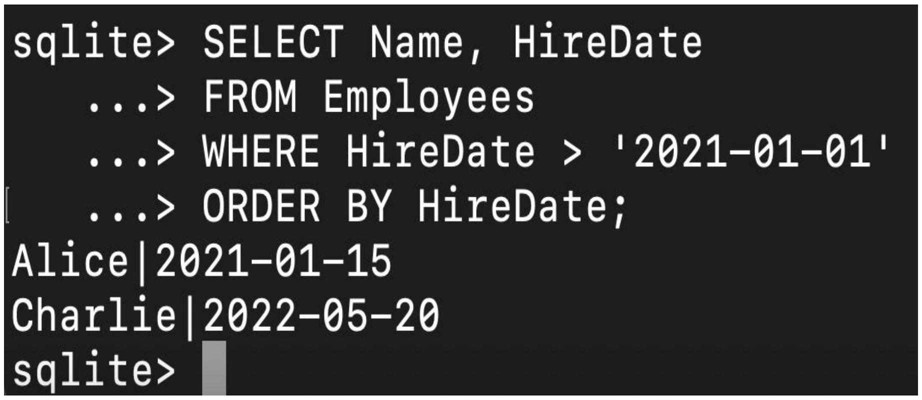
- **Departments: Stores information about departments.**



DepartmentID	DepartmentName
101	HR
102	IT
103	Finance

Q1. Write a query to list the names of employees hired after January 1, 2021.

```
SELECT Name, HireDate
FROM Employees
WHERE HireDate > '2021-01-01'
ORDER BY HireDate;
```



```
sqlite> SELECT Name, HireDate
   ...> FROM Employees
   ...> WHERE HireDate > '2021-01-01'
   ...> ORDER BY HireDate;
Alice|2021-01-15
Charlie|2022-05-20
sqlite>
```

Q2. Write a query to calculate the average salary of employees in each department.

SELECT

d.DepartmentName AS department_name,

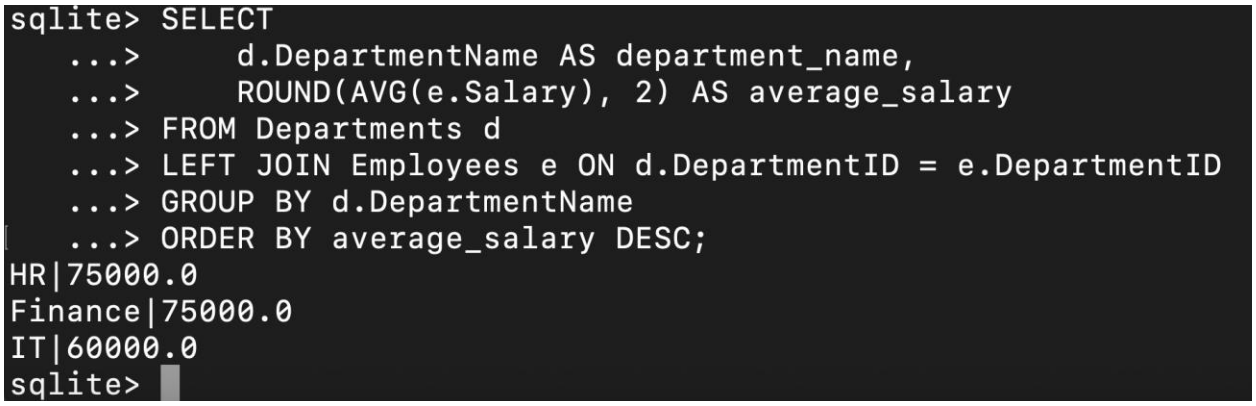
ROUND(AVG(e.Salary), 2) AS average_salary

FROM Departments d

LEFT JOIN Employees e ON d.DepartmentID = e.DepartmentID

GROUP BY d.DepartmentName

ORDER BY average_salary DESC;



```
sqlite> SELECT
   ...>     d.DepartmentName AS department_name,
   ...>     ROUND(AVG(e.Salary), 2) AS average_salary
   ...> FROM Departments d
   ...> LEFT JOIN Employees e ON d.DepartmentID = e.DepartmentID
   ...> GROUP BY d.DepartmentName
   ...> ORDER BY average_salary DESC;
HR|75000.0
Finance|75000.0
IT|60000.0
sqlite>
```

Q3. Write a query to find the department name where the total salary is the highest.

SELECT

d.DepartmentName AS department_name,

SUM(e.Salary) AS total_salary

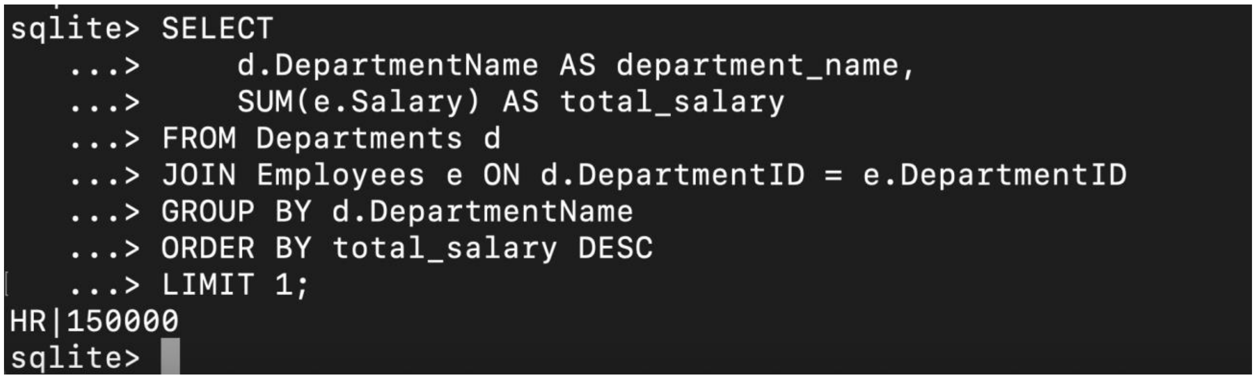
FROM Departments d

JOIN Employees e ON d.DepartmentID = e.DepartmentID

GROUP BY d.DepartmentName

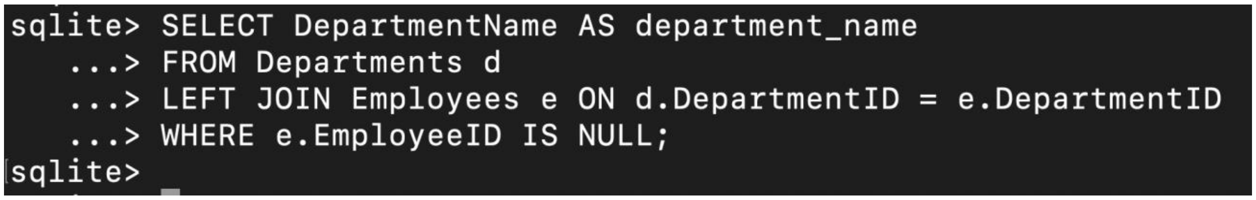
ORDER BY total_salary DESC

LIMIT 1;


```
sqlite> SELECT
   ...>     d.DepartmentName AS department_name,
   ...>     SUM(e.Salary) AS total_salary
   ...> FROM Departments d
   ...> JOIN Employees e ON d.DepartmentID = e.DepartmentID
   ...> GROUP BY d.DepartmentName
   ...> ORDER BY total_salary DESC
   ...> LIMIT 1;
HR|150000
sqlite>
```

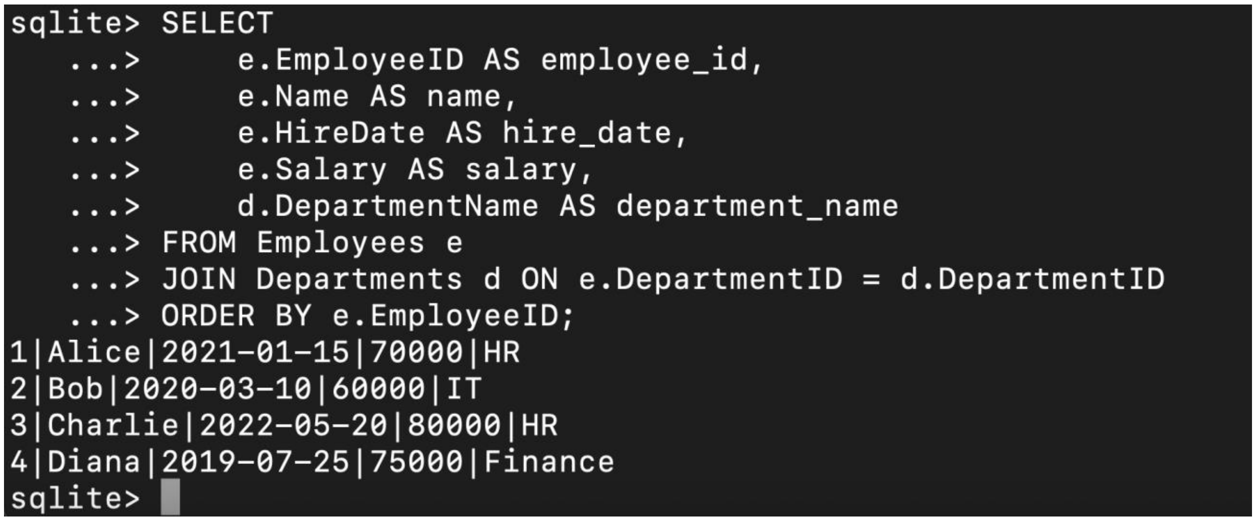
Q4. Write a query to list all departments that currently have no employees assigned.

```
SELECT DepartmentName AS department_name

FROM Departments d

LEFT JOIN Employees e ON d.DepartmentID = e.DepartmentID

WHERE e.EmployeeID IS NULL;
```


```
sqlite> SELECT DepartmentName AS department_name
   ...> FROM Departments d
   ...> LEFT JOIN Employees e ON d.DepartmentID = e.DepartmentID
   ...> WHERE e.EmployeeID IS NULL;
sqlite>
```

Q5. Write a query to fetch all employee details along with their department names.

```
SELECT

   e.EmployeeID AS employee_id,

   e.Name AS name,

   e.HireDate AS hire_date,

   e.Salary AS salary,

   d.DepartmentName AS department_name

FROM Employees e

JOIN Departments d ON e.DepartmentID = d.DepartmentID

ORDER BY e.EmployeeID;
```



```
sqlite> SELECT
   ...>     e.EmployeeID AS employee_id,
   ...>     e.Name AS name,
   ...>     e.HireDate AS hire_date,
   ...>     e.Salary AS salary,
   ...>     d.DepartmentName AS department_name
   ...> FROM Employees e
   ...> JOIN Departments d ON e.DepartmentID = d.DepartmentID
   ...> ORDER BY e.EmployeeID;
1|Alice|2021-01-15|70000|HR
2|Bob|2020-03-10|60000|IT
3|Charlie|2022-05-20|80000|HR
4|Diana|2019-07-25|75000|Finance
sqlite> 
```