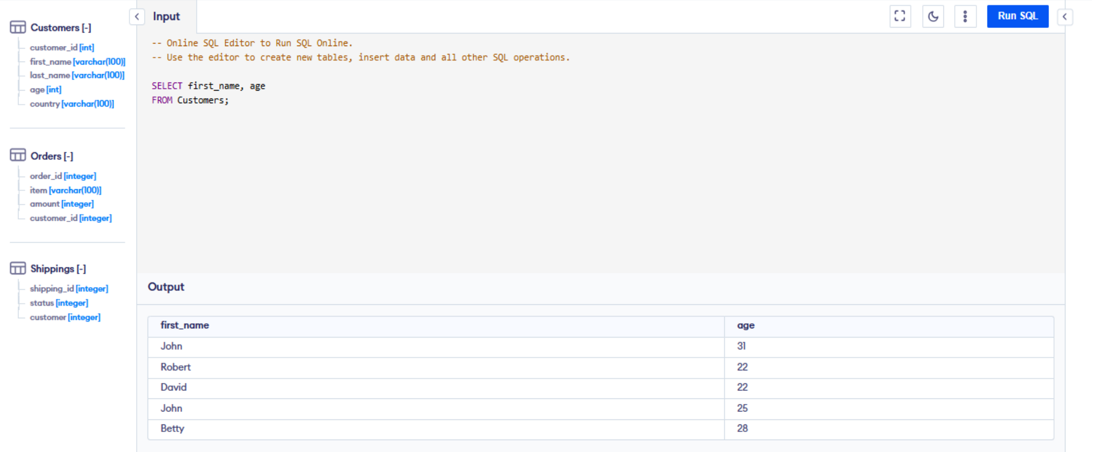
YoungDevIntern

SQL-DATABASE-TASKS

3/9/2025
MARYAM TARIQ
linkedin

USING PROGRAMIZ SQLONLINE COMPILER



Customers [-]
- customer_id [int]
- first_name [varchar(100)]
- last_name [varchar(100)]
- age [int]
- country [varchar(100)]

Orders [-]
- order_id [integer]
- item [varchar(100)]
- amount [integer]
- customer_id [integer]

Shippings [-]
- shipping_id [integer]
- status [integer]
- customer [integer]

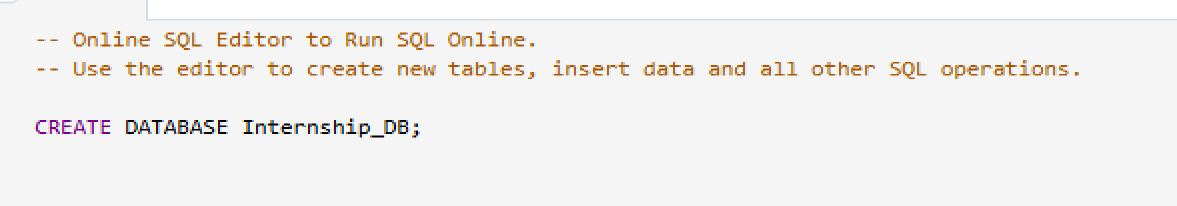
```sql
-- Online SQL Editor to Run SQL Online.
-- Use the editor to create new tables, insert data and all other SQL operations.

SELECT first_name, age
FROM Customers;
```

first_name	age
John	31
Robert	22
David	22
John	25
Betty	28

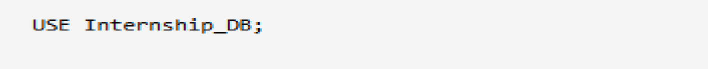
Basic Tasks (Week 1)

1. Database & Table Setup
   - Create Internship_DB, tables Employees and Departments.
   - Insert 5 records into each table.
2. Basic SELECT Queries
   - Retrieve employees' names and positions.
   - List departments with locations.
3. Filtering & Sorting
   - Query employees earning > $50,000.
   - Sort employees by name, list departments in specific cities.

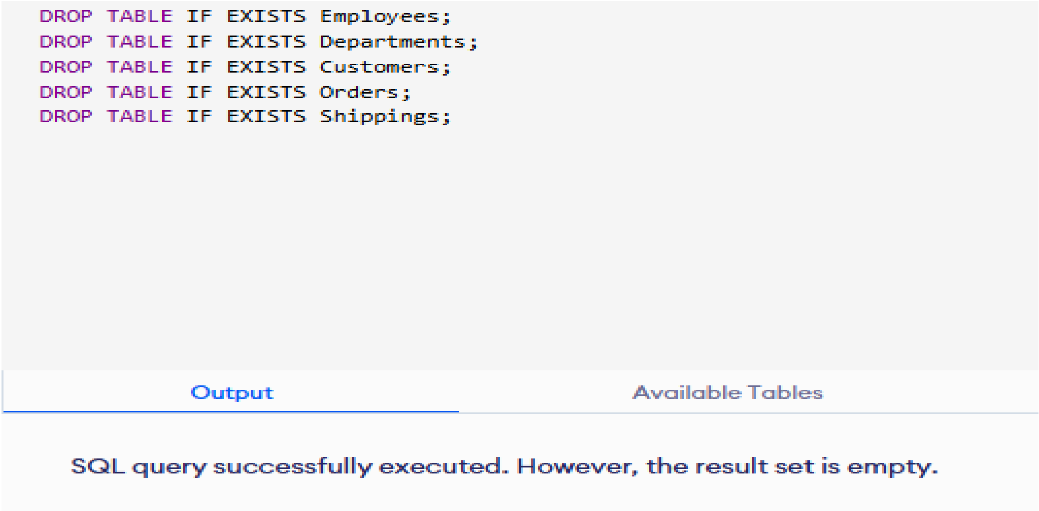
1. Database & Table Setup



```sql
-- Online SQL Editor to Run SQL Online.
-- Use the editor to create new tables, insert data and all other SQL operations.

CREATE DATABASE Internship_DB;
```

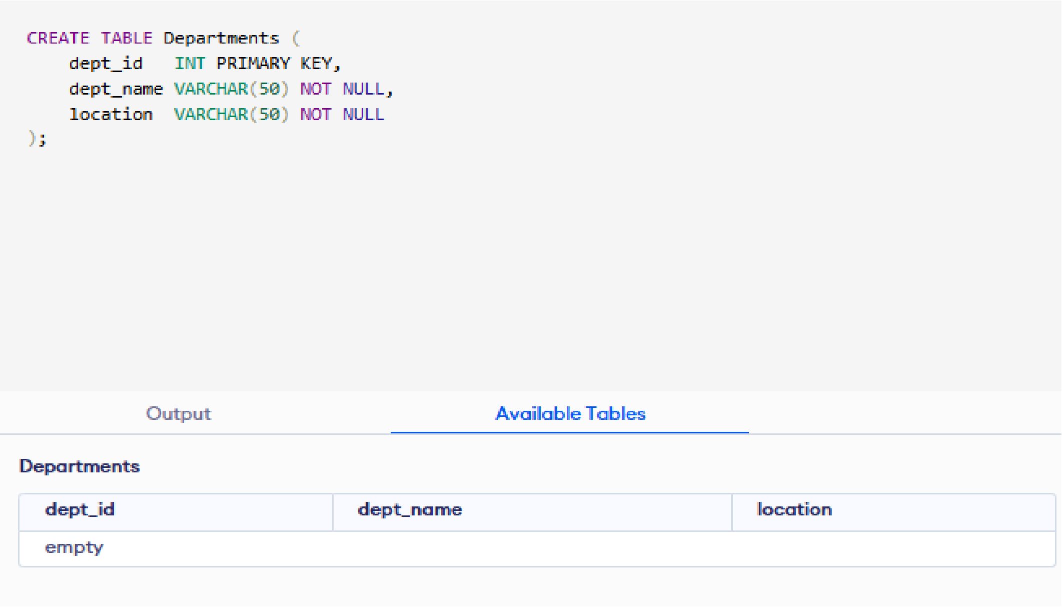


```sql
USE Internship_DB;
```



```sql
DROP TABLE IF EXISTS Employees;
DROP TABLE IF EXISTS Departments;
DROP TABLE IF EXISTS Customers;
DROP TABLE IF EXISTS Orders;
DROP TABLE IF EXISTS Shippings;
```

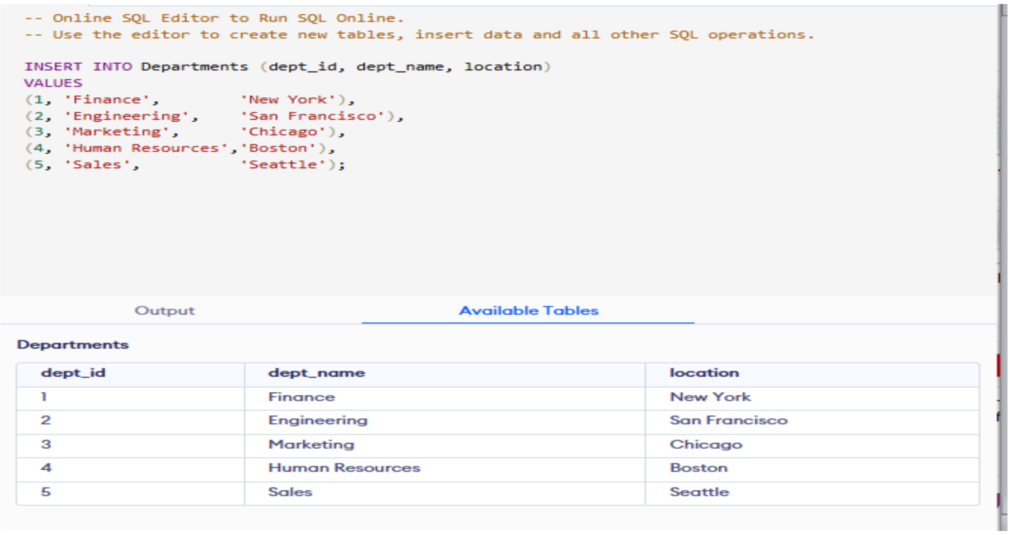
SQL query successfully executed. However, the result set is empty.



```sql
CREATE TABLE Departments (
    dept_id   INT PRIMARY KEY,
    dept_name VARCHAR(50) NOT NULL,
    location  VARCHAR(50) NOT NULL
);
```

Departments

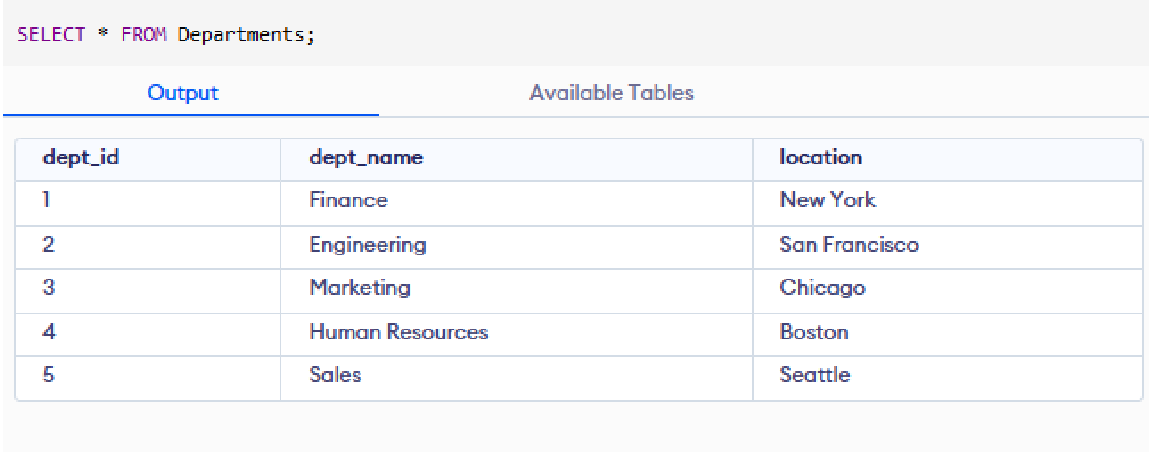
dept_id	dept_name	location
empty		



```sql
-- Online SQL Editor to Run SQL Online.
-- Use the editor to create new tables, insert data and all other SQL operations.

INSERT INTO Departments (dept_id, dept_name, location)
VALUES
(1, 'Finance',         'New York'),
(2, 'Engineering',     'San Francisco'),
(3, 'Marketing',       'Chicago'),
(4, 'Human Resources','Boston'),
(5, 'Sales',           'Seattle');
```

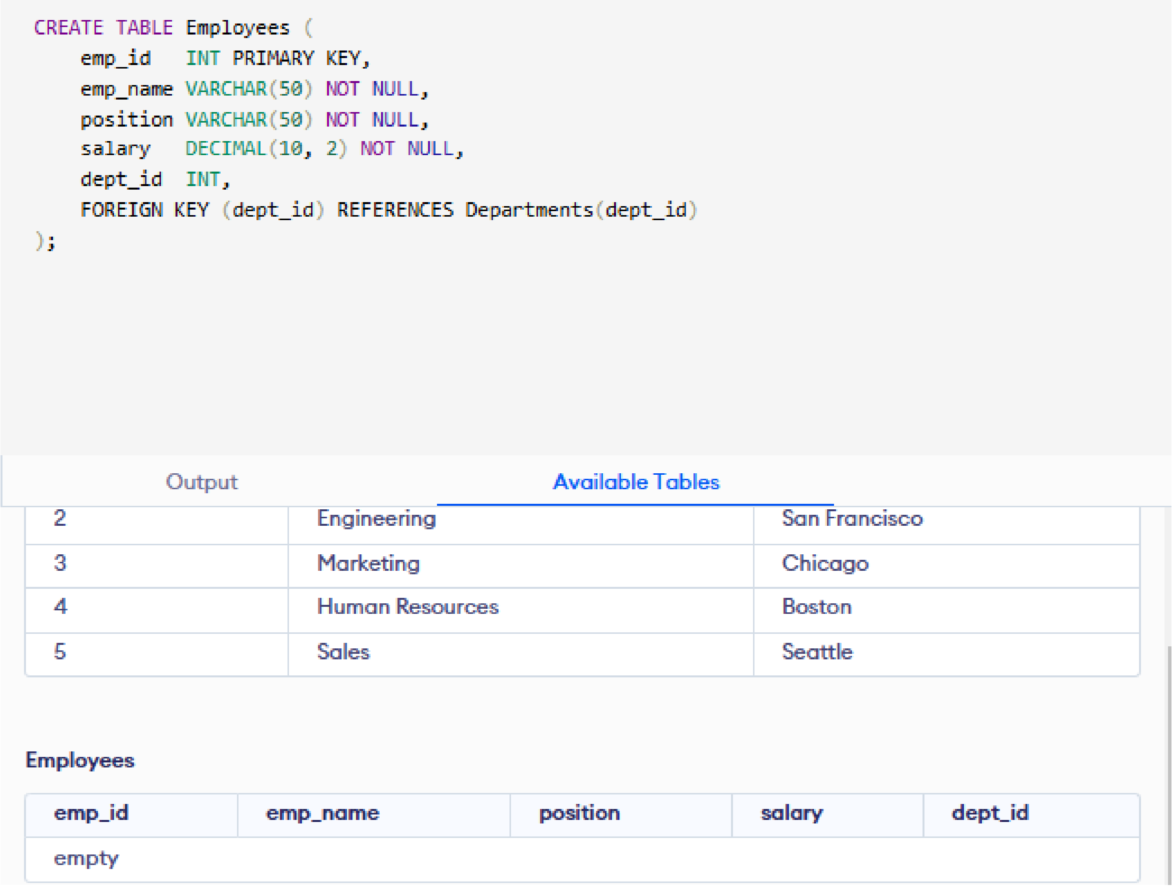
Departments

dept_id	dept_name	location
1	Finance	New York
2	Engineering	San Francisco
3	Marketing	Chicago
4	Human Resources	Boston
5	Sales	Seattle



```
SELECT * FROM Departments;
```

dept_id	dept_name	location
1	Finance	New York
2	Engineering	San Francisco
3	Marketing	Chicago
4	Human Resources	Boston
5	Sales	Seattle

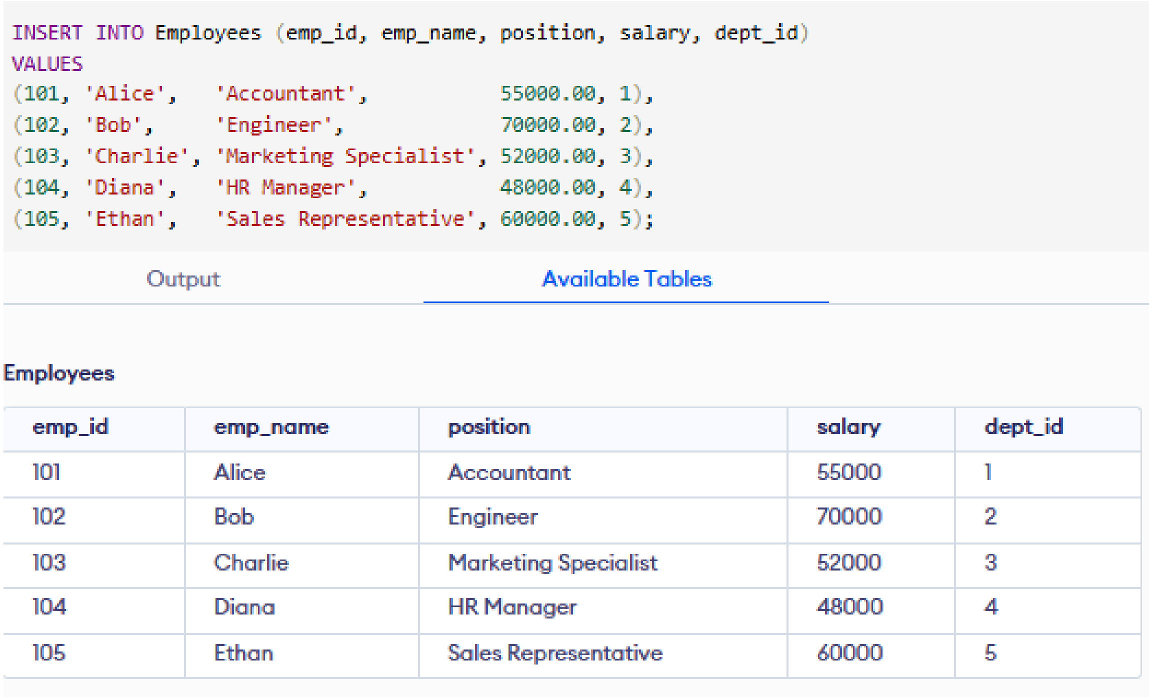


```
CREATE TABLE Employees (
    emp_id   INT PRIMARY KEY,
    emp_name VARCHAR(50) NOT NULL,
    position VARCHAR(50) NOT NULL,
    salary   DECIMAL(10, 2) NOT NULL,
    dept_id  INT,
    FOREIGN KEY (dept_id) REFERENCES Departments(dept_id)
);
```

2	Engineering	San Francisco
3	Marketing	Chicago
4	Human Resources	Boston
5	Sales	Seattle

Employees

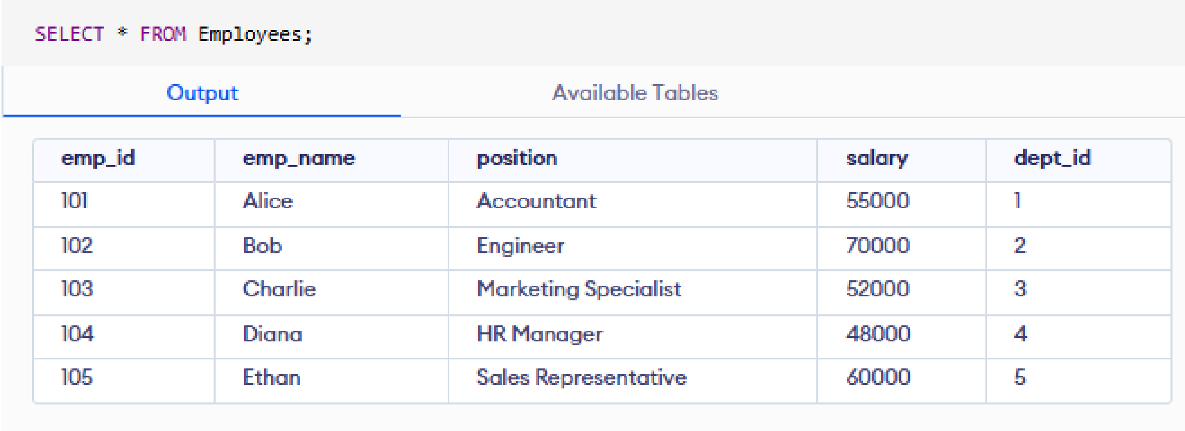
emp_id	emp_name	position	salary	dept_id
empty				



```
INSERT INTO Employees (emp_id, emp_name, position, salary, dept_id)
VALUES
(101, 'Alice',   'Accountant',           55000.00, 1),
(102, 'Bob',     'Engineer',             70000.00, 2),
(103, 'Charlie', 'Marketing Specialist', 52000.00, 3),
(104, 'Diana',   'HR Manager',           48000.00, 4),
(105, 'Ethan',   'Sales Representative', 60000.00, 5);
```

Employees

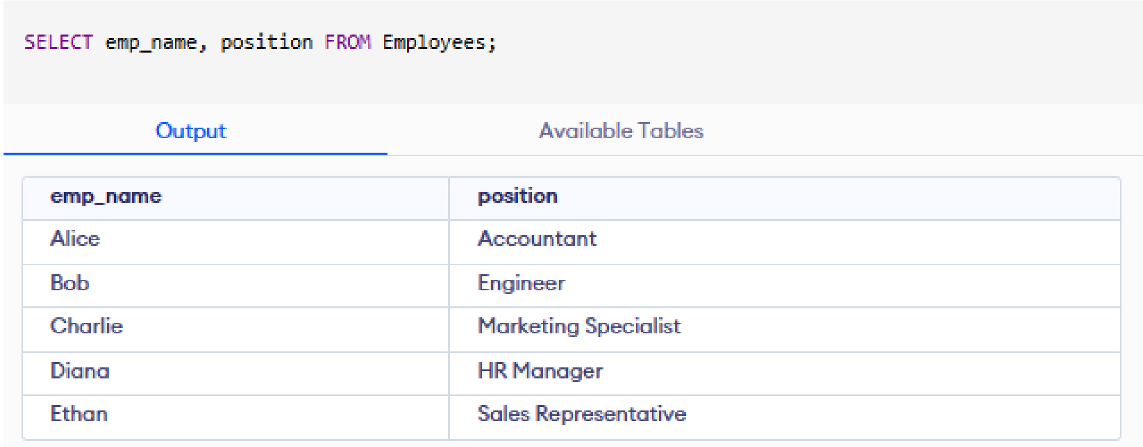
emp_id	emp_name	position	salary	dept_id
101	Alice	Accountant	55000	1
102	Bob	Engineer	70000	2
103	Charlie	Marketing Specialist	52000	3
104	Diana	HR Manager	48000	4
105	Ethan	Sales Representative	60000	5



```
SELECT * FROM Employees;
```

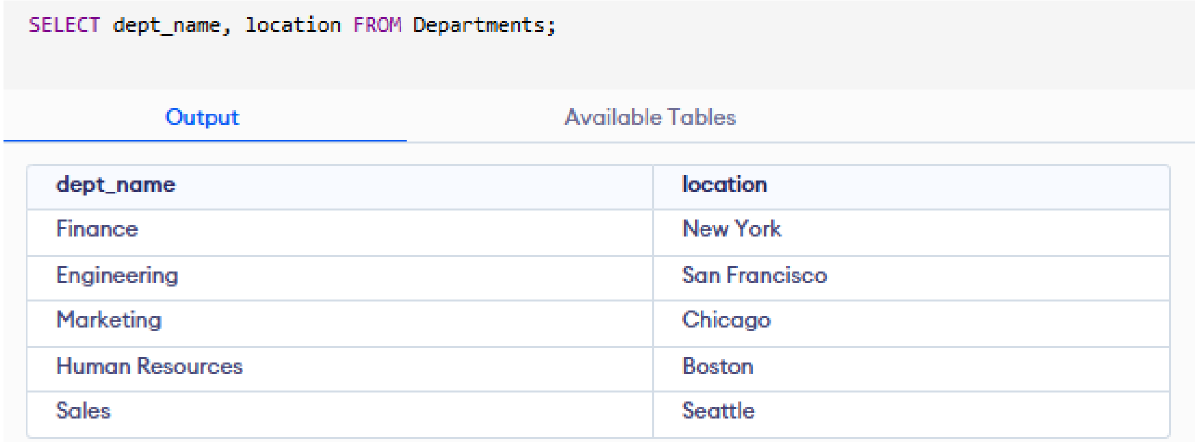
emp_id	emp_name	position	salary	dept_id
101	Alice	Accountant	55000	1
102	Bob	Engineer	70000	2
103	Charlie	Marketing Specialist	52000	3
104	Diana	HR Manager	48000	4
105	Ethan	Sales Representative	60000	5

2. Basic SELECT Queries



```
SELECT emp_name, position FROM Employees;
```

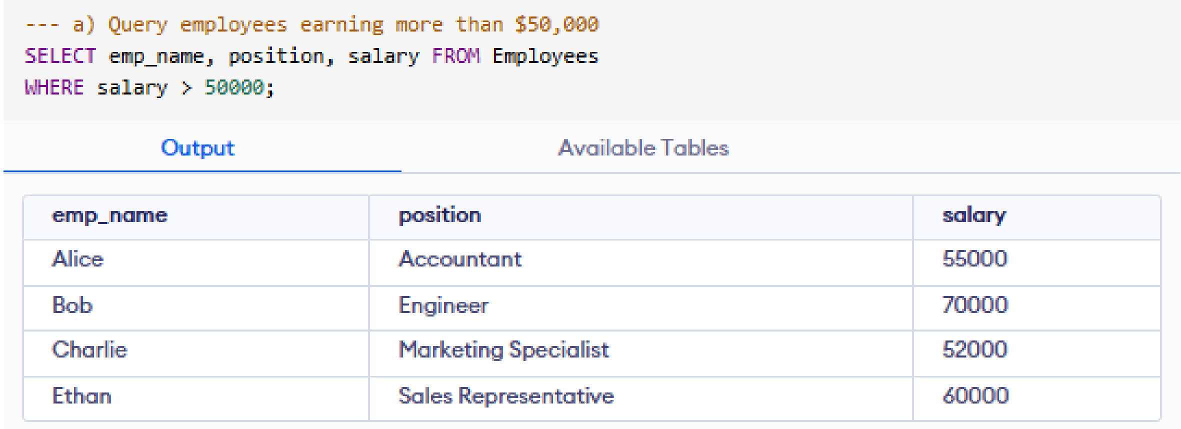
emp_name	position
Alice	Accountant
Bob	Engineer
Charlie	Marketing Specialist
Diana	HR Manager
Ethan	Sales Representative



```sql
SELECT dept_name, location FROM Departments;
```

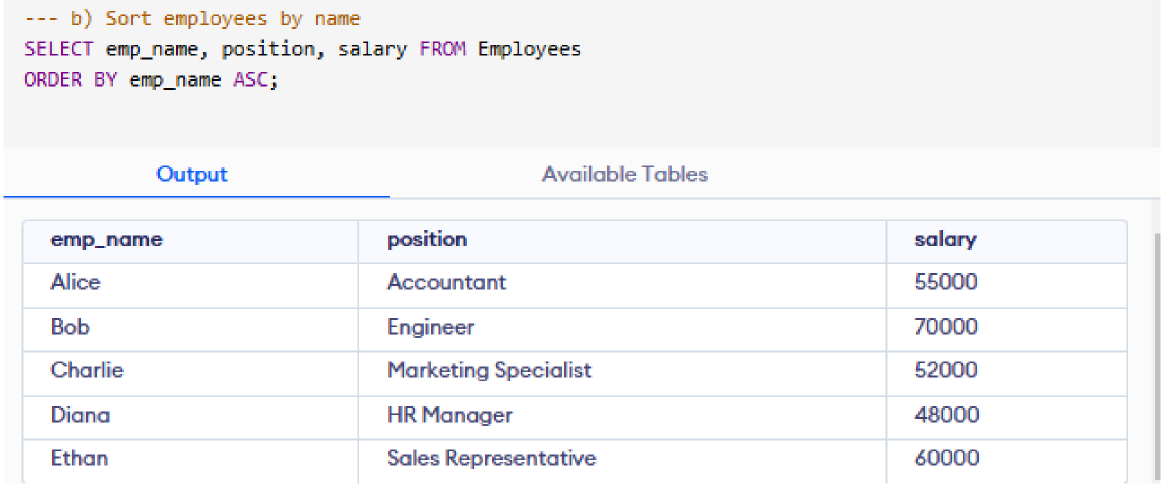
dept_name	location
Finance	New York
Engineering	San Francisco
Marketing	Chicago
Human Resources	Boston
Sales	Seattle

3. Filtering & Sorting:



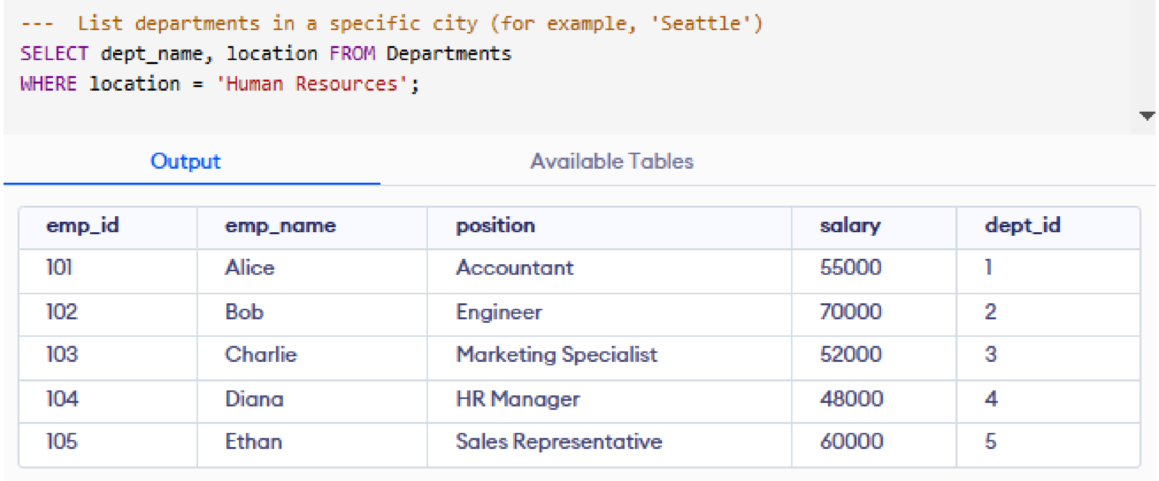
```sql
--- a) Query employees earning more than $50,000
SELECT emp_name, position, salary FROM Employees
WHERE salary > 50000;
```

emp_name	position	salary
Alice	Accountant	55000
Bob	Engineer	70000
Charlie	Marketing Specialist	52000
Ethan	Sales Representative	60000



```sql
--- b) Sort employees by name
SELECT emp_name, position, salary FROM Employees
ORDER BY emp_name ASC;
```

emp_name	position	salary
Alice	Accountant	55000
Bob	Engineer	70000
Charlie	Marketing Specialist	52000
Diana	HR Manager	48000
Ethan	Sales Representative	60000



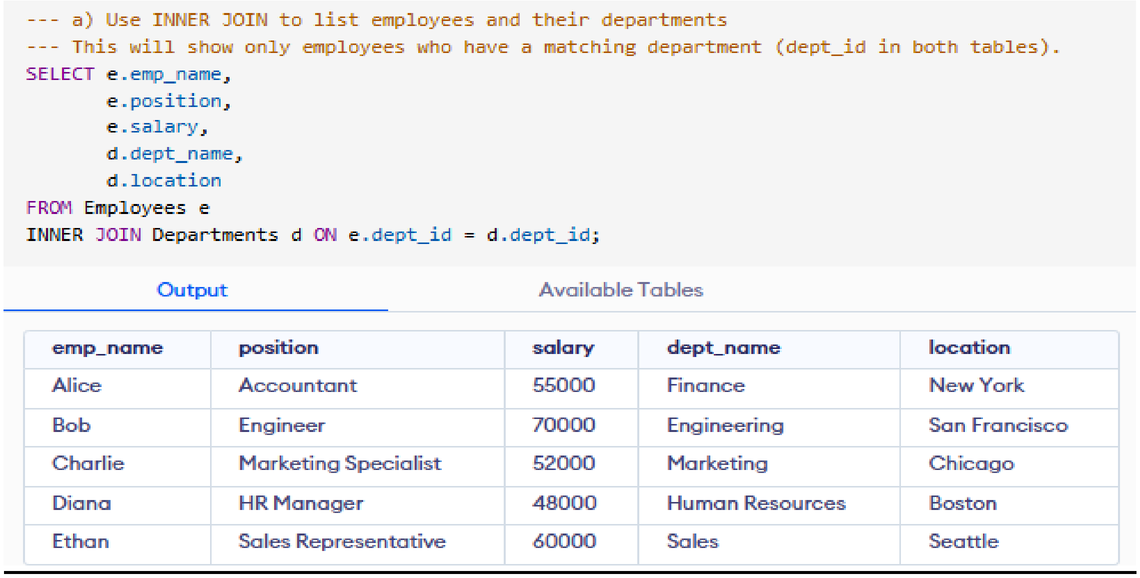
```sql
---  List departments in a specific city (for example, 'Seattle')
SELECT dept_name, location FROM Departments
WHERE location = 'Human Resources';
```

emp_id	emp_name	position	salary	dept_id
101	Alice	Accountant	55000	1
102	Bob	Engineer	70000	2
103	Charlie	Marketing Specialist	52000	3
104	Diana	HR Manager	48000	4
105	Ethan	Sales Representative	60000	5

Intermediate Tasks (Week 2-3)

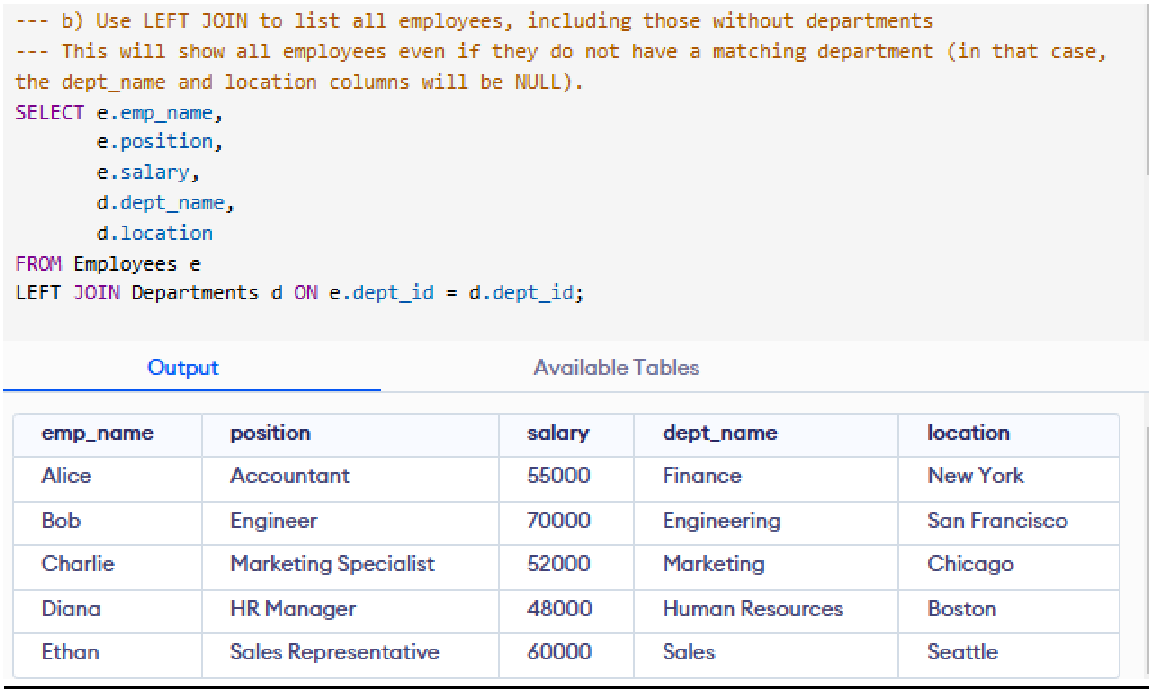
1. JOIN Operations
   - Use INNER JOIN to list employees and their departments.
   - Use LEFT JOIN to list all employees, including those without departments.
2. Aggregation Functions
   - Calculate average salary, total employees per department, and highest salary in each department.
3. Subqueries
   - Find employees earning more than the department's average salary.
   - List departments with more than 3 employees.

1. JOIN Operations



```sql
--- a) Use INNER JOIN to list employees and their departments
--- This will show only employees who have a matching department (dept_id in both tables).
SELECT e.emp_name,
       e.position,
       e.salary,
       d.dept_name,
       d.location
FROM Employees e
INNER JOIN Departments d ON e.dept_id = d.dept_id;
```

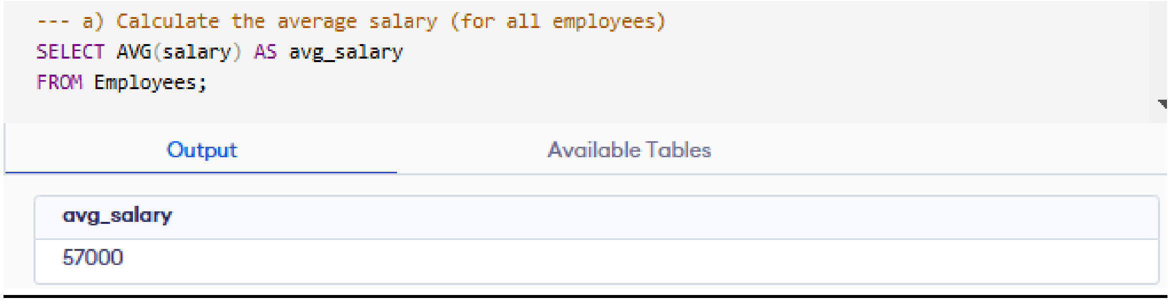
emp_name	position	salary	dept_name	location
Alice	Accountant	55000	Finance	New York
Bob	Engineer	70000	Engineering	San Francisco
Charlie	Marketing Specialist	52000	Marketing	Chicago
Diana	HR Manager	48000	Human Resources	Boston
Ethan	Sales Representative	60000	Sales	Seattle



```sql
--- b) Use LEFT JOIN to list all employees, including those without departments
--- This will show all employees even if they do not have a matching department (in that case,
the dept_name and location columns will be NULL).
SELECT e.emp_name,
       e.position,
       e.salary,
       d.dept_name,
       d.location
FROM Employees e
LEFT JOIN Departments d ON e.dept_id = d.dept_id;
```

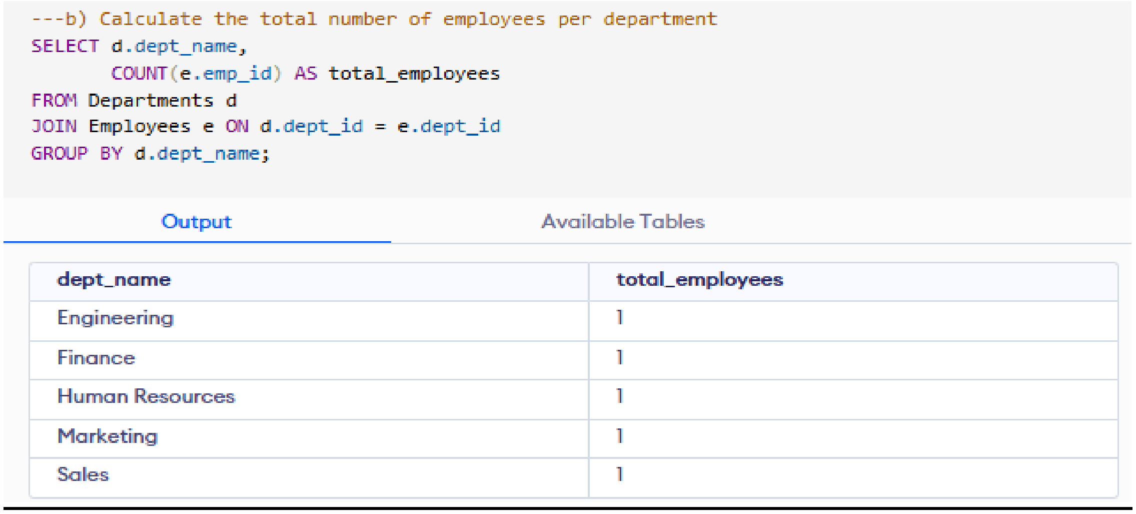
emp_name	position	salary	dept_name	location
Alice	Accountant	55000	Finance	New York
Bob	Engineer	70000	Engineering	San Francisco
Charlie	Marketing Specialist	52000	Marketing	Chicago
Diana	HR Manager	48000	Human Resources	Boston
Ethan	Sales Representative	60000	Sales	Seattle

2. Aggregation Functions:



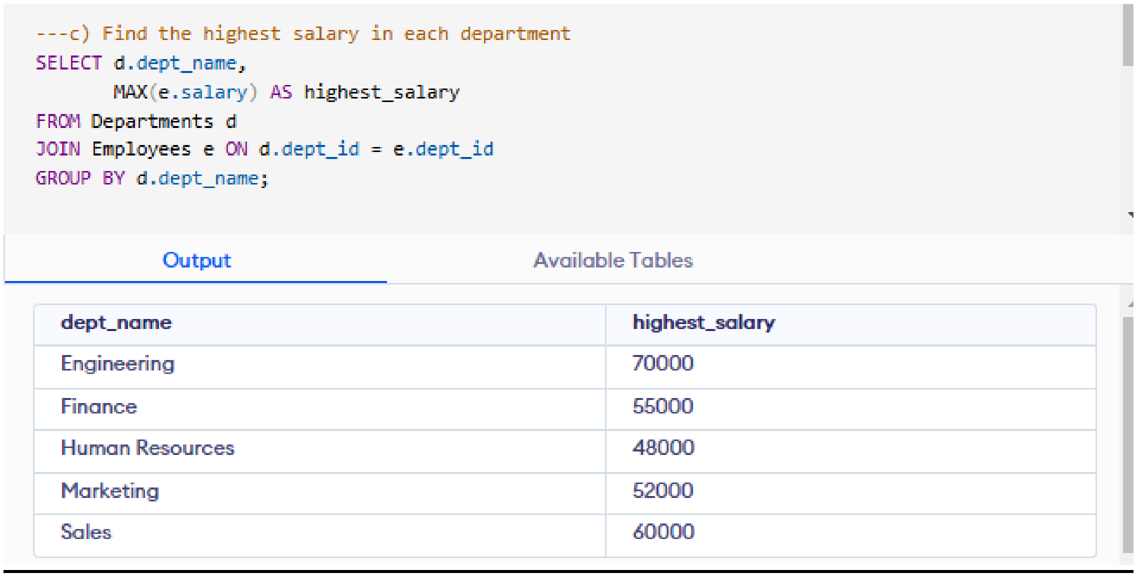
```sql
--- a) Calculate the average salary (for all employees)
SELECT AVG(salary) AS avg_salary
FROM Employees;
```

avg_salary
57000



```sql
---b) Calculate the total number of employees per department
SELECT d.dept_name,
       COUNT(e.emp_id) AS total_employees
FROM Departments d
JOIN Employees e ON d.dept_id = e.dept_id
GROUP BY d.dept_name;
```

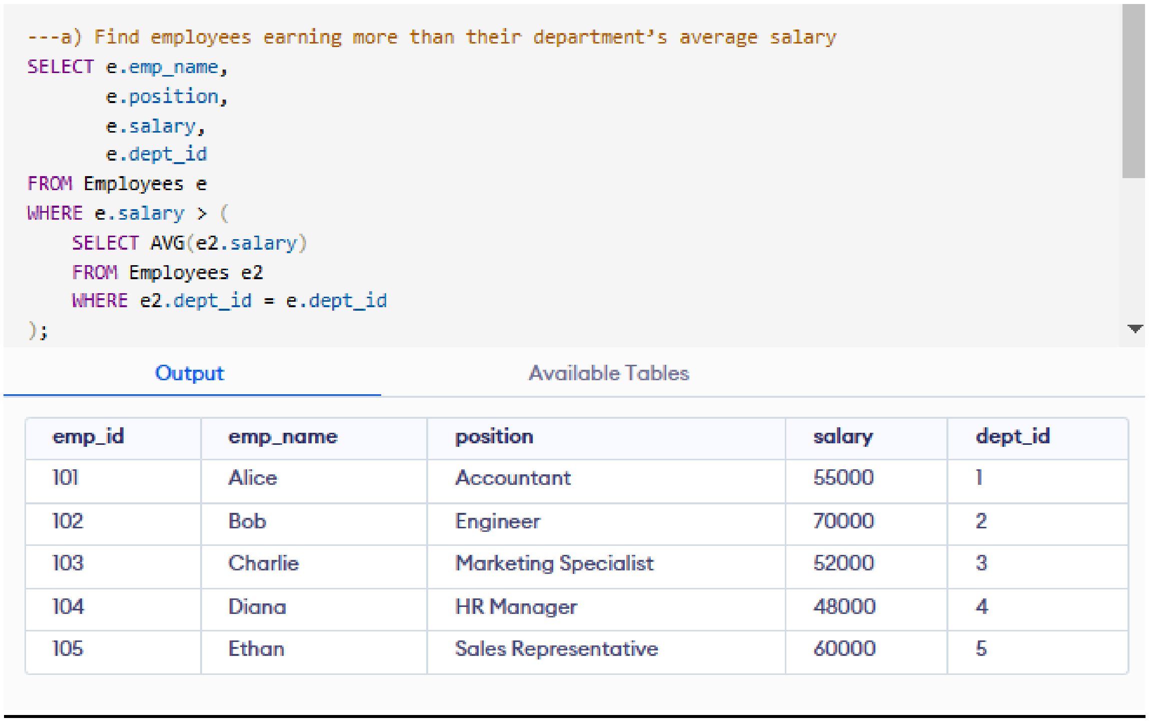
dept_name	total_employees
Engineering	1
Finance	1
Human Resources	1
Marketing	1
Sales	1



```sql
---c) Find the highest salary in each department
SELECT d.dept_name,
       MAX(e.salary) AS highest_salary
FROM Departments d
JOIN Employees e ON d.dept_id = e.dept_id
GROUP BY d.dept_name;
```

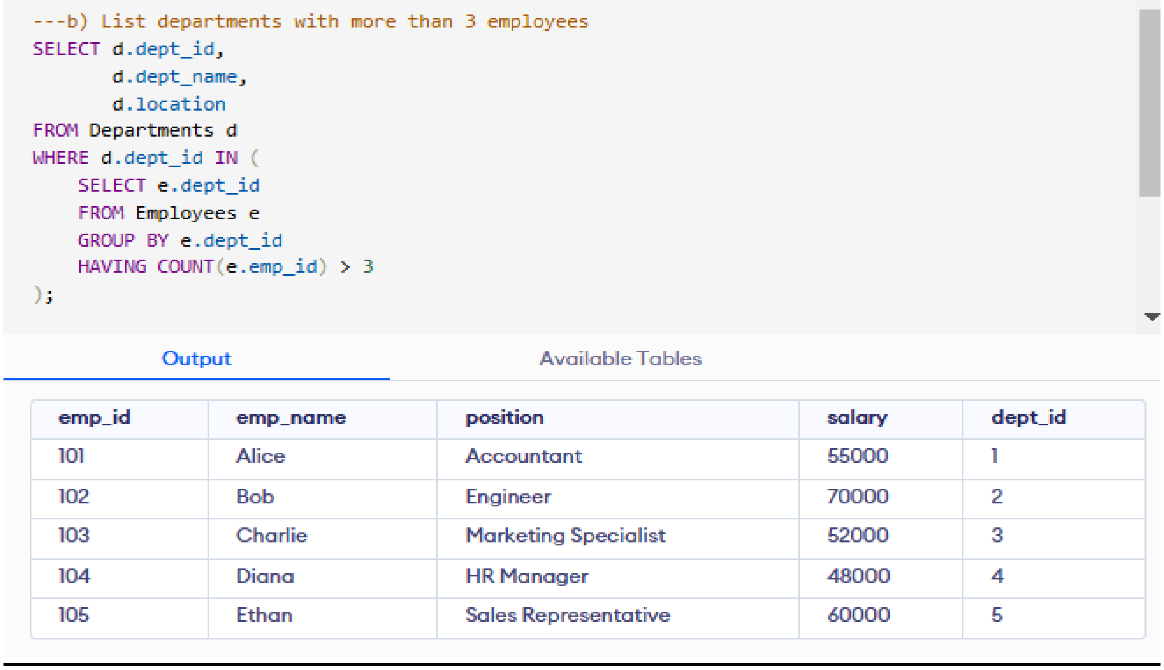
dept_name	highest_salary
Engineering	70000
Finance	55000
Human Resources	48000
Marketing	52000
Sales	60000

3. Subqueries



```sql
---a) Find employees earning more than their department's average salary
SELECT e.emp_name,
       e.position,
       e.salary,
       e.dept_id
FROM Employees e
WHERE e.salary > (
    SELECT AVG(e2.salary)
    FROM Employees e2
    WHERE e2.dept_id = e.dept_id
);
```

emp_id	emp_name	position	salary	dept_id
101	Alice	Accountant	55000	1
102	Bob	Engineer	70000	2
103	Charlie	Marketing Specialist	52000	3
104	Diana	HR Manager	48000	4
105	Ethan	Sales Representative	60000	5

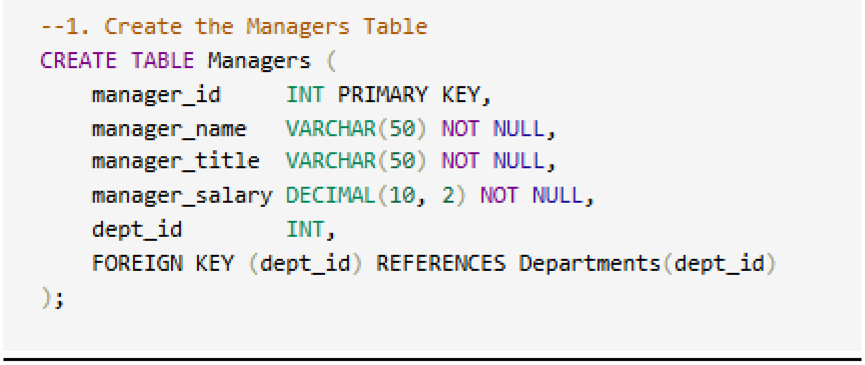


```
---b) List departments with more than 3 employees
SELECT d.dept_id,
       d.dept_name,
       d.location
FROM Departments d
WHERE d.dept_id IN (
    SELECT e.dept_id
    FROM Employees e
    GROUP BY e.dept_id
    HAVING COUNT(e.emp_id) > 3
);
```

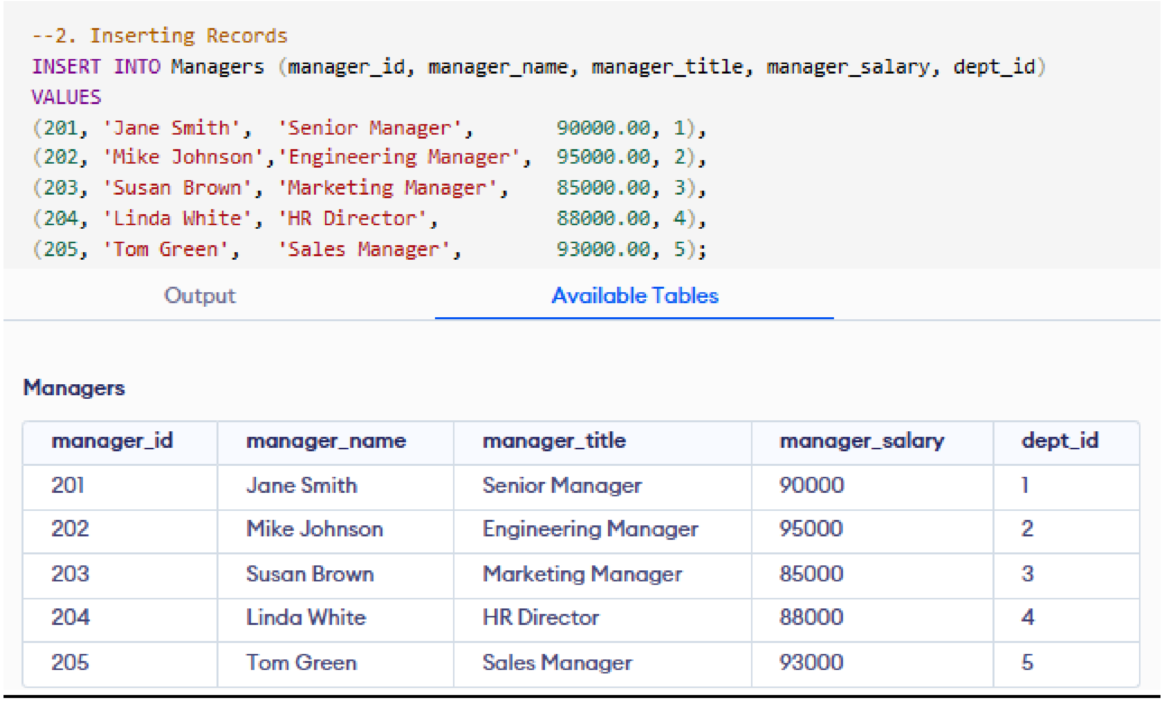
emp_id	emp_name	position	salary	dept_id
101	Alice	Accountant	55000	1
102	Bob	Engineer	70000	2
103	Charlie	Marketing Specialist	52000	3
104	Diana	HR Manager	48000	4
105	Ethan	Sales Representative	60000	5

Expert Tasks (Week 4)

1. Complex JOINs
   - Join Employees, Departments, and Managers to list employees with department and manager details.
2. Window Functions
   - Rank employees by salary within their department using ROW_NUMBER().
   - Rank employees across the company using RANK().
3. Data Modification & Transactions
   - Update employee salaries by 10%.
   - Use transactions to commit or roll back updates.



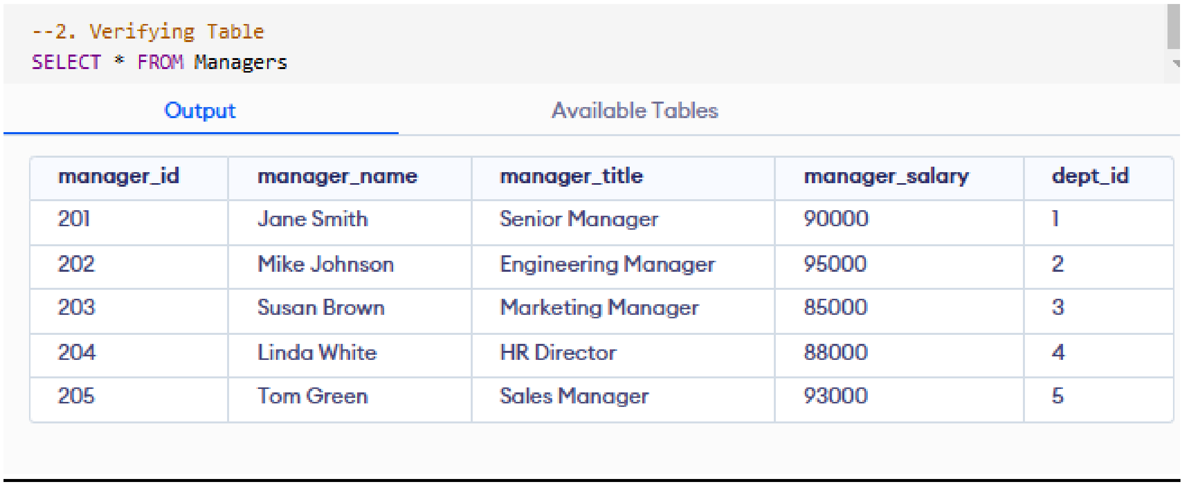
```
--1. Create the Managers Table
CREATE TABLE Managers (
    manager_id     INT PRIMARY KEY,
    manager_name   VARCHAR(50) NOT NULL,
    manager_title  VARCHAR(50) NOT NULL,
    manager_salary DECIMAL(10, 2) NOT NULL,
    dept_id        INT,
    FOREIGN KEY (dept_id) REFERENCES Departments(dept_id)
);
```



```
--2. Inserting Records
INSERT INTO Managers (manager_id, manager_name, manager_title, manager_salary, dept_id)
VALUES
(201, 'Jane Smith',  'Senior Manager',       90000.00, 1),
(202, 'Mike Johnson','Engineering Manager',  95000.00, 2),
(203, 'Susan Brown', 'Marketing Manager',    85000.00, 3),
(204, 'Linda White', 'HR Director',          88000.00, 4),
(205, 'Tom Green',   'Sales Manager',        93000.00, 5);
```

Managers

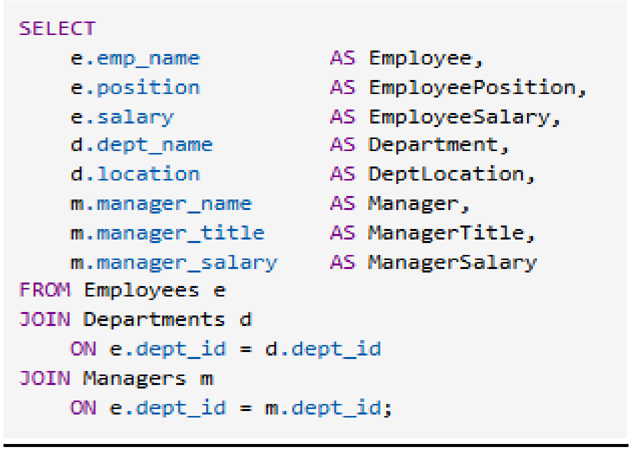
manager_id	manager_name	manager_title	manager_salary	dept_id
201	Jane Smith	Senior Manager	90000	1
202	Mike Johnson	Engineering Manager	95000	2
203	Susan Brown	Marketing Manager	85000	3
204	Linda White	HR Director	88000	4
205	Tom Green	Sales Manager	93000	5



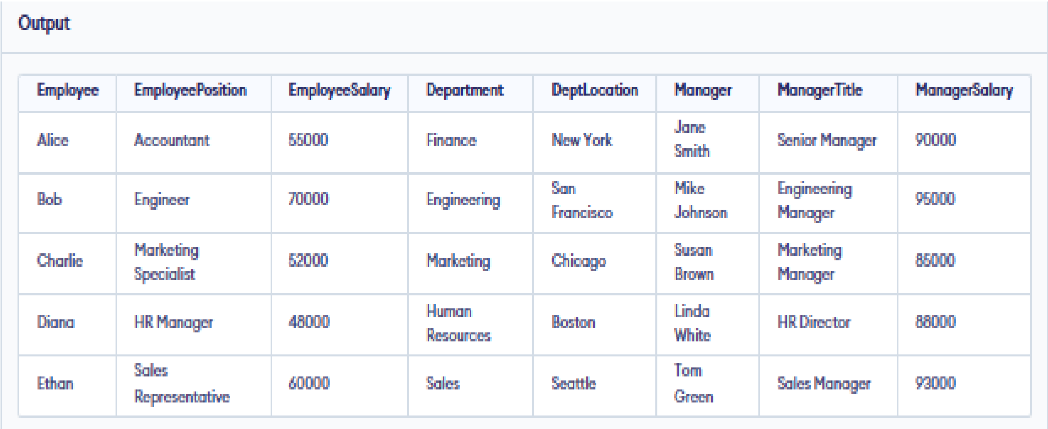
```
--2. Verifying Table
SELECT * FROM Managers
```

manager_id	manager_name	manager_title	manager_salary	dept_id
201	Jane Smith	Senior Manager	90000	1
202	Mike Johnson	Engineering Manager	95000	2
203	Susan Brown	Marketing Manager	85000	3
204	Linda White	HR Director	88000	4
205	Tom Green	Sales Manager	93000	5

1. Complex JOINs



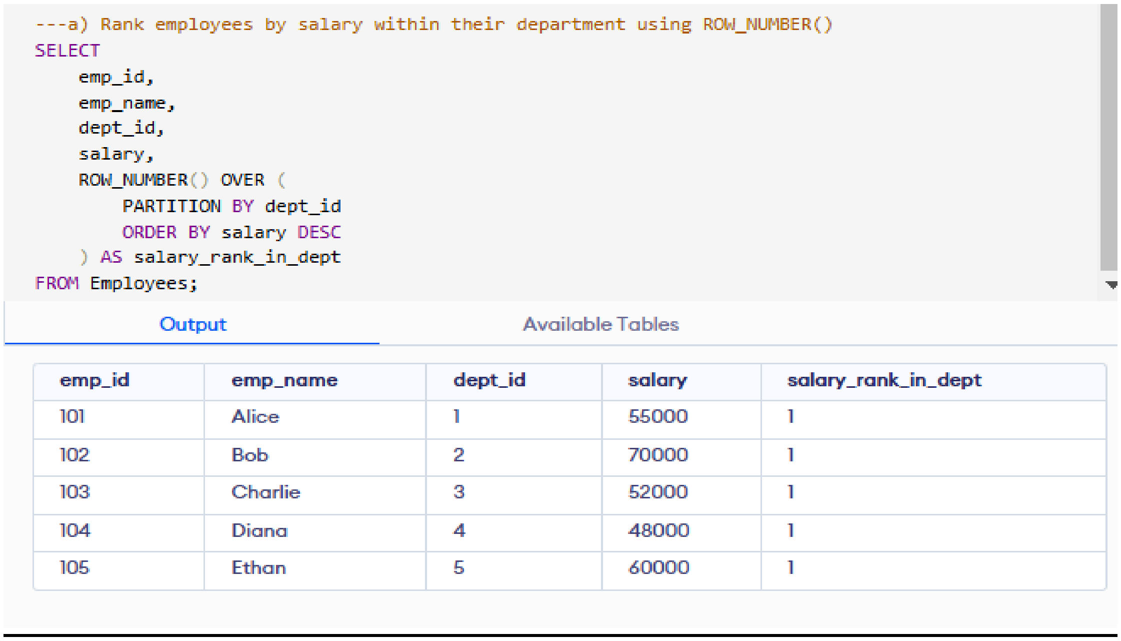
```
SELECT
    e.emp_name           AS Employee,
    e.position           AS EmployeePosition,
    e.salary             AS EmployeeSalary,
    d.dept_name          AS Department,
    d.location           AS DeptLocation,
    m.manager_name       AS Manager,
    m.manager_title      AS ManagerTitle,
    m.manager_salary     AS ManagerSalary
FROM Employees e
JOIN Departments d
    ON e.dept_id = d.dept_id
JOIN Managers m
    ON e.dept_id = m.dept_id;
```



Output

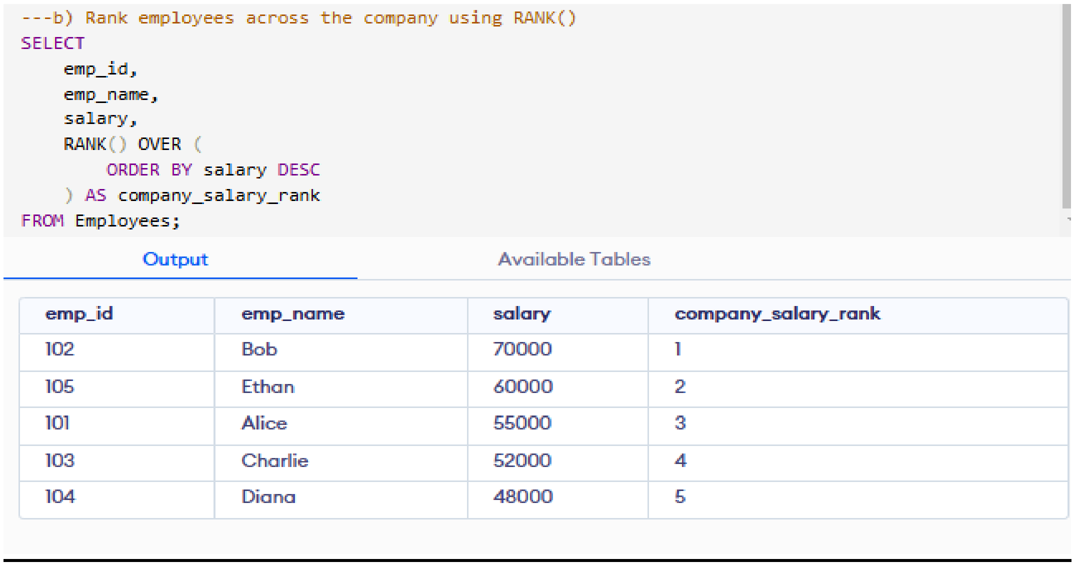
Employee	EmployeePosition	EmployeeSalary	Department	DeptLocation	Manager	ManagerTitle	ManagerSalary
Alice	Accountant	55000	Finance	New York	Jane Smith	Senior Manager	90000
Bob	Engineer	70000	Engineering	San Francisco	Mike Johnson	Engineering Manager	95000
Charlie	Marketing Specialist	52000	Marketing	Chicago	Susan Brown	Marketing Manager	85000
Diana	HR Manager	48000	Human Resources	Boston	Linda White	HR Director	88000
Ethan	Sales Representative	60000	Sales	Seattle	Tom Green	Sales Manager	93000

2. Window Functions



```
---a) Rank employees by salary within their department using ROW_NUMBER()
SELECT
    emp_id,
    emp_name,
    dept_id,
    salary,
    ROW_NUMBER() OVER (
        PARTITION BY dept_id
        ORDER BY salary DESC
    ) AS salary_rank_in_dept
FROM Employees;
```

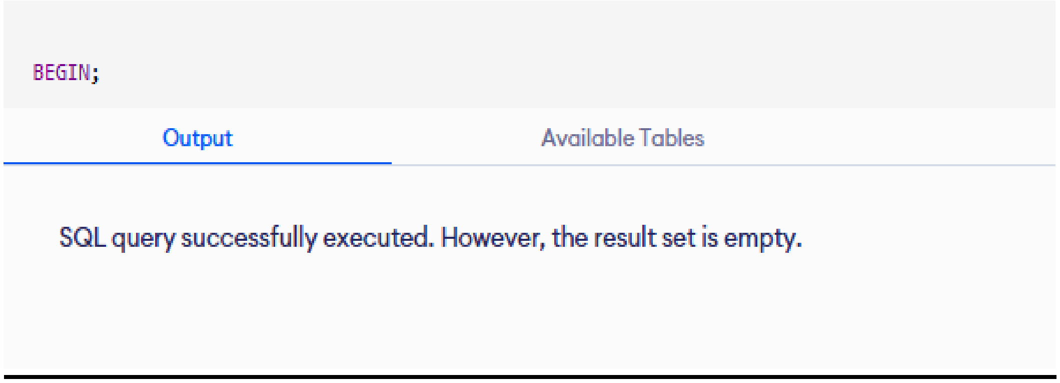
emp_id	emp_name	dept_id	salary	salary_rank_in_dept
101	Alice	1	55000	1
102	Bob	2	70000	1
103	Charlie	3	52000	1
104	Diana	4	48000	1
105	Ethan	5	60000	1



```
---b) Rank employees across the company using RANK()
SELECT
    emp_id,
    emp_name,
    salary,
    RANK() OVER (
        ORDER BY salary DESC
    ) AS company_salary_rank
FROM Employees;
```

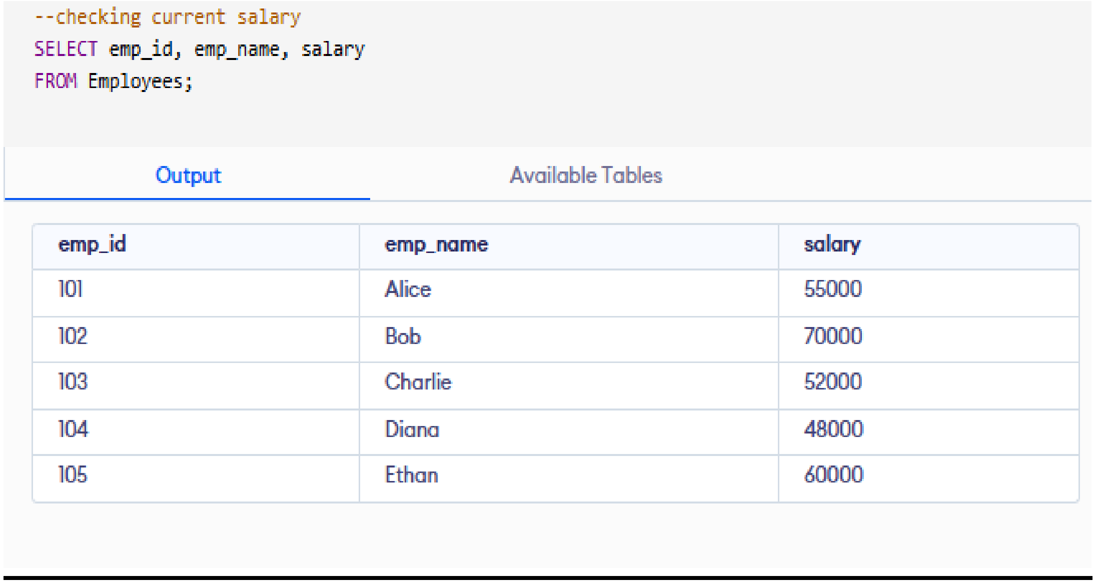
emp_id	emp_name	salary	company_salary_rank
102	Bob	70000	1
105	Ethan	60000	2
101	Alice	55000	3
103	Charlie	52000	4
104	Diana	48000	5

3. Data Modification & Transactions



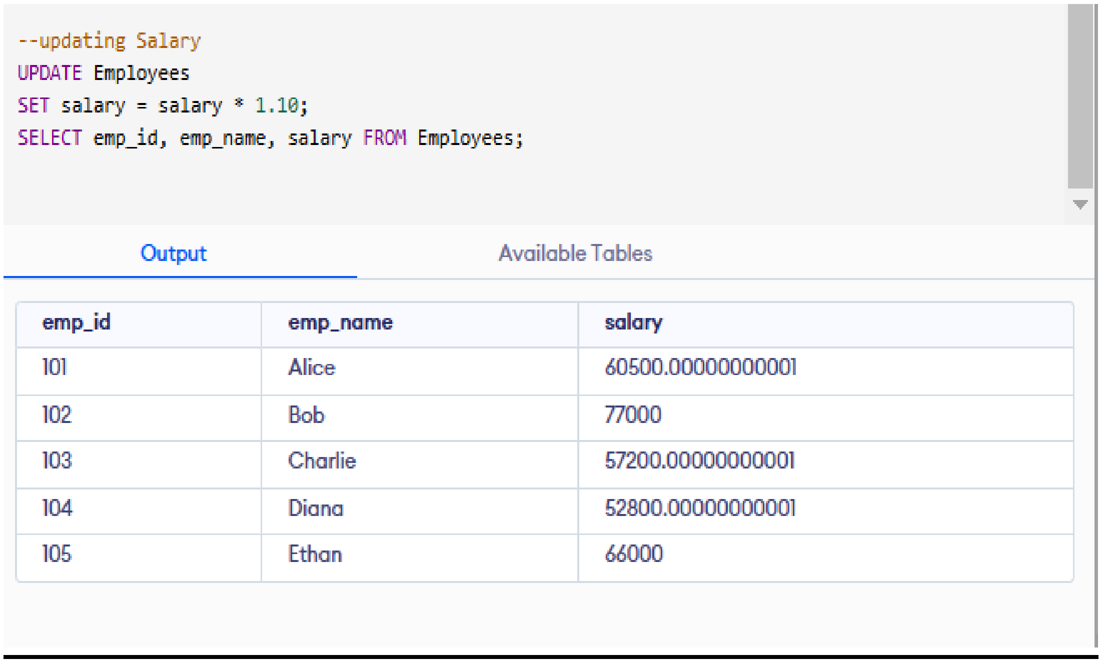
```
BEGIN;
```

SQL query successfully executed. However, the result set is empty.



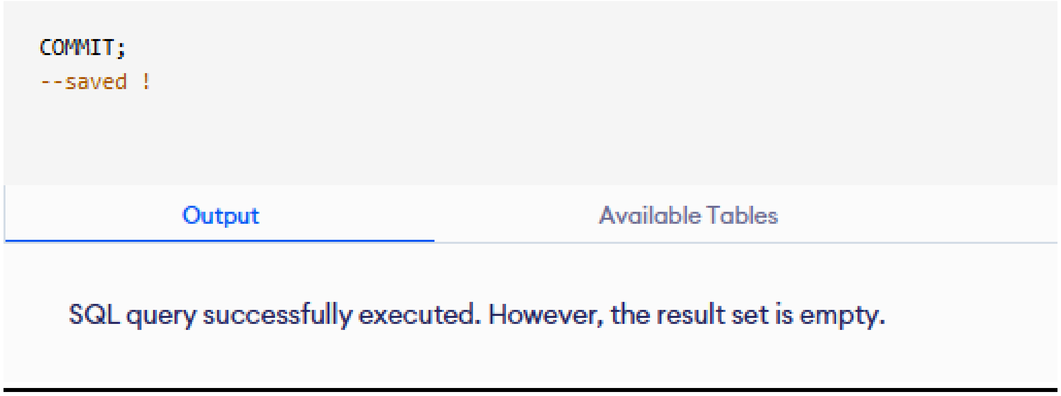
```
--checking current salary
SELECT emp_id, emp_name, salary
FROM Employees;
```

emp_id	emp_name	salary
101	Alice	55000
102	Bob	70000
103	Charlie	52000
104	Diana	48000
105	Ethan	60000



```
--updating Salary
UPDATE Employees
SET salary = salary * 1.10;
SELECT emp_id, emp_name, salary FROM Employees;
```

emp_id	emp_name	salary
101	Alice	60500.00000000001
102	Bob	77000
103	Charlie	57200.00000000001
104	Diana	52800.00000000001
105	Ethan	66000



```
COMMIT;
--saved !
```

```
SQL query successfully executed. However, the result set is empty.
```

THE END – THANK YOU