EXPERIMENT- 03

Student Name: Chandan Singla **UID: 23BCS12759**

Branch: BE-CSE **Section/Group: KRG 1(A)**

Semester: 05 **Date of Performance: 14/08/25**

Subject Name: ADBMS **Subject Code: 23CSP-333**

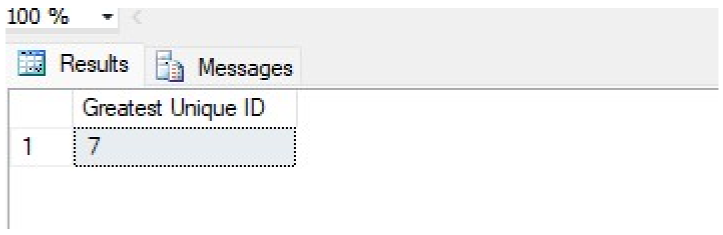
1. Aim:
Generate an employee relation with only one attribute i.e., EMP_ID. Then, find the max EMP_ID, but excluding the duplicates.

2. Objective:
To create an employee relation consisting of a single attribute **EMP_ID**, and then determine the maximum **EMP_ID** value by considering only unique employee IDs (i.e., excluding duplicates).

3. DBMS Script:

```
CREATE TABLE TBL_EMPLOYEE(
EMP_ID INT );
INSERT INTO TBL_EMPLOYEE VALUES (2),(4),(4),(6),(6),(7),(8),(8);
SELECT MAX(EMP_ID) as [Greatest Unique ID] FROM TBL_EMPLOYEE WHERE
EMP_ID IN
(SELECT EMP_ID FROM TBL_EMPLOYEE GROUP BY EMP_ID HAVING
COUNT(EMP_ID)=1);
```

4. Output:



```
   Greatest Unique ID
1  7
```

Department Salary Champions(Medium)

1. Aim:

In a bustling corporate organization, each department strives to retain the most talented (and well-compensated) employees. You have access to two key records: one lists every employee along with their salary and department, while the other details the names of each department. Your task is to identify the top earners in every department.

If multiple employees share the same highest salary within a department, all of them should be celebrated equally. The final result should present the department name, employee name, and salary of these top-tier professionals arranged by department.

2. Objective:

- **Understanding Subqueries:** Learned how to use subqueries to perform intermediate calculations (like finding the maximum salary per department) and use that result in the main query.
- **Handling Ties and Multiple Results:** Gained the skill to fetch all employees sharing the top salary within a department, not just one, ensuring accurate and fair results.
- **Data Integration & Presentation:** Practiced joining multiple tables (employees and departments) and arranging results logically, which reinforces skills in combining and presenting relational data efficiently.

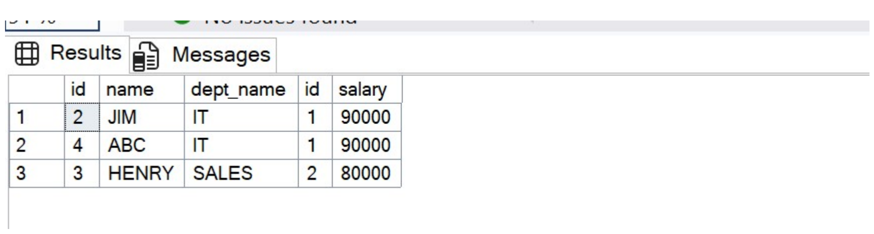
3. DBMS script:

```sql
CREATE TABLE department (
    id INT PRIMARY KEY,
    dept_name VARCHAR(50)
);

-- Create Employee Table
```

```
CREATE TABLE employees (
    id INT,
    name VARCHAR(50),
    salary INT,
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES department(id)
);


-- Insert into Department Table
INSERT INTO department (id, dept_name) VALUES
(1, 'IT'),
(2, 'SALES');

-- Insert into Employee Table
INSERT INTO employees (id, name, salary, department_id) VALUES
(1, 'JOE', 70000, 1),
(2, 'JIM', 90000, 1),
(3, 'HENRY', 80000, 2),
(4, 'SAM', 60000, 2),
(5, 'MAX', 90000, 1);

-- JOINS + SUBQUERIES
SELECT D.dept_name, E.name,E.salary, D.id
FROM department as D
join employees as E
ON E.department_id = D.id
WHERE E.salary IN
```

```
(
SELECT MAX(E2.SALARY)
FROM EMPLOYEES AS E2
WHERE E2.department_id = E.department_id
)
ORDER BY D.id;

-- JOINS + GROUP BY
SELECT D.dept_name, E.name,E.salary, D.id
FROM department as D
join employees as E
ON E.department_id = D.id
WHERE E.salary IN
(
SELECT MAX(E2.SALARY)
FROM EMPLOYEES AS E2
group by e2.department_id
)
ORDER BY D.id;
```

4. Output:



	id	name	dept_name	id	salary
1	2	JIM	IT	1	90000
2	4	ABC	IT	1	90000
3	3	HENRY	SALES	2	80000

Merging Employee Histories: Who Earned Least? (Hard)

1. Aim:

Two legacy HR systems (A and B) have separate records of employee salaries. These records may overlap. Management wants to **merge these datasets** and identify **each unique employee** (by EmpID) along with their **lowest recorded salary** across both systems.

i. Combine two tables A and B.

ii. Return each EmpID with their **lowest salary,** and the corresponding **Ename.**

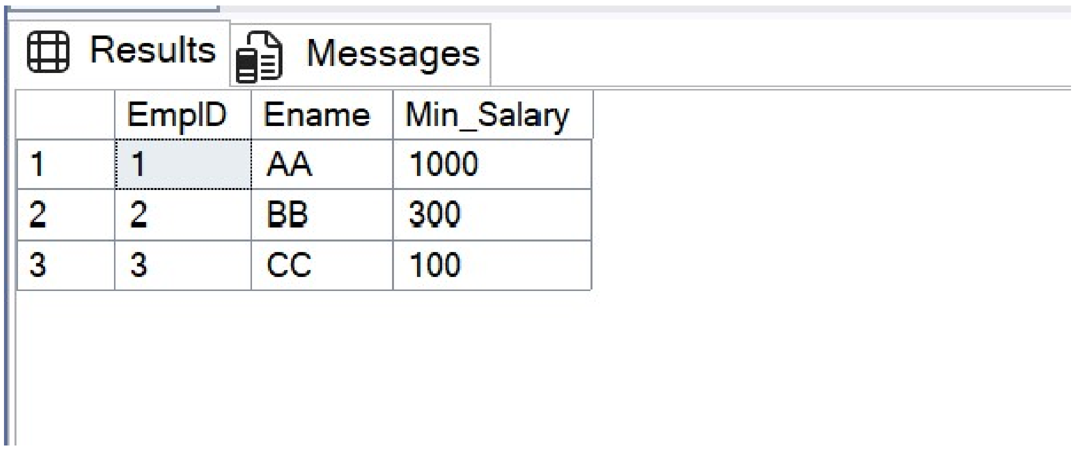
2. Objective:

- **Merging Data:** Learned how to combine multiple tables using UNION to handle overlapping employee records.
- **Finding Minimum Salary:** Practiced using aggregation to determine the lowest salary for each employee.
- **Handling Duplicates & Retrieval:** Reinforced skills in managing duplicate entries and retrieving associated information like employee name accurately.

3. DBMS script:

```sql
CREATE TABLE TABLE_A(
EMPID INT,
ENAME VARCHAR(20),
SALARY INT
);

CREATE TABLE TABLE_B(
EMPID INT,
ENAME VARCHAR(20),
SALARY INT
);

INSERT INTO TABLE_A VALUES
(1, 'AA', 1000),
(2,'BB', 300);

INSERT INTO TABLE_B VALUES
(2, 'BB', 400),
(3,'CC', 100);

SELECT EMPID, ENAME, MIN(SALARY) as MIN_SALARY
FROM
(SELECT * FROM TABLE_A AS A
UNION ALL
SELECT * FROM TABLE_B AS B)
AS INTER_RESULT
GROUP BY EMPID, ENAME;

-- APPROACH 02:
SELECT EMPID, MIN(ENAME) AS ENAME, MIN(SALARY) as MIN_SALARY
FROM
(SELECT * FROM TABLE_A AS A
UNION ALL
SELECT * FROM TABLE_B AS B)
AS INTER_RESULT
GROUP BY EMPID;
```

4. **Output:**



	EmpID	Ename	Min_Salary
1	1	AA	1000
2	2	BB	300
3	3	CC	100