Experiment No 3:

Student Name: Bhavya Gupta **UID: 23BIS70147**
Branch: BE CSE (IS) **Section/Group: 23AIT-KRG(2A)**
Semester: 5th **Date of Performance:25 July 2025**
Subject Name: ADBMS **Subject Code: 23-CSP-333**

1. Aim/Overview of the practical:

[EASY] You are given with employee table with only one attribute that is emp_id which contains values as:

Employee (emp_id)
2
4
4
6
6
7
8
8
8

Task: find the maximum value for emp_id, but excluding the duplicate employee id's. (Only with sub-queries)
Output: 7
Explaination: if we exclude duplicates such as, 4, 6, and 8, & from the rest i.E., 2,7 the maximum is 7.

[MEDIUM] Department Salary Champions:

In a bustling corporate organization, each department strives to retain the most talented (and well-compensated) employed. 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 result should present the department name, employee name, and salary of these top-tier professionals arranged by department.

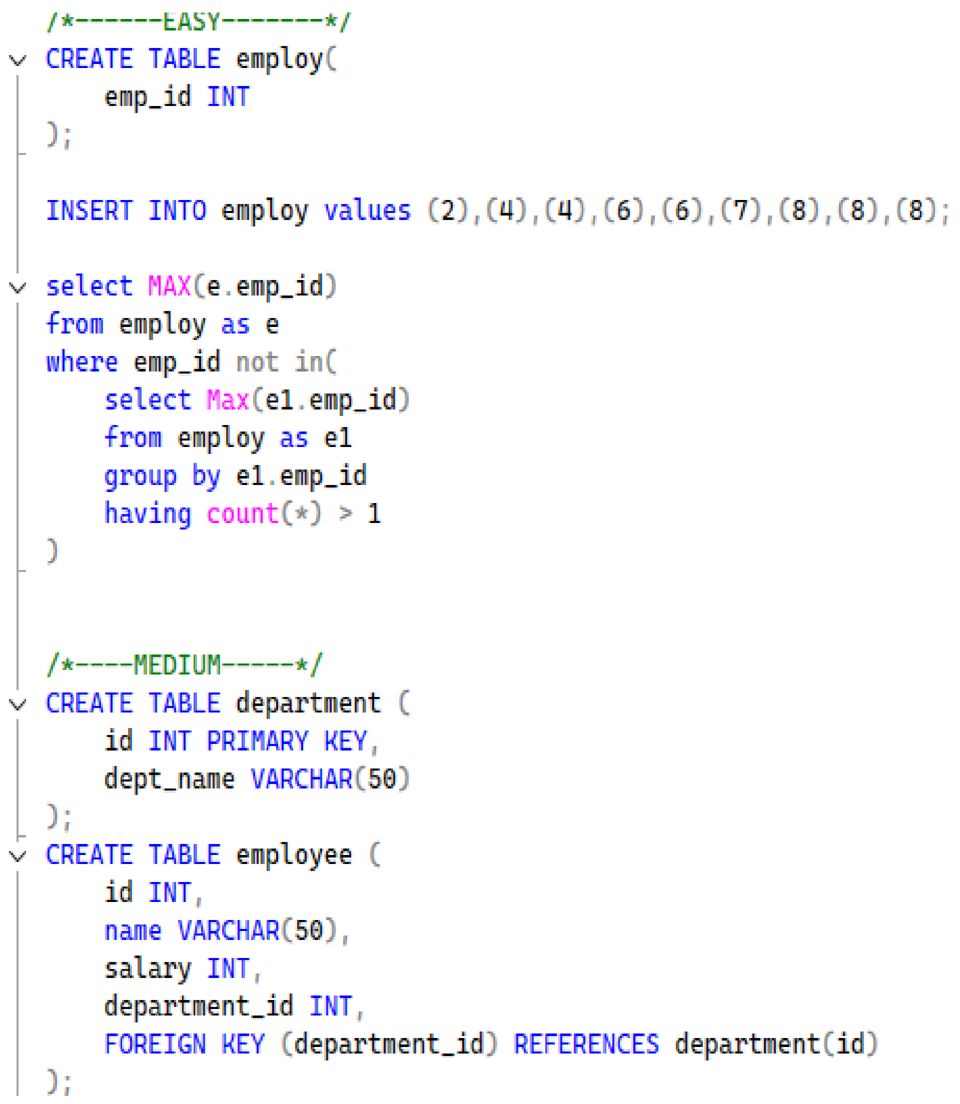
[HARD] Merging Employee Histories: Who Earned Least?

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.

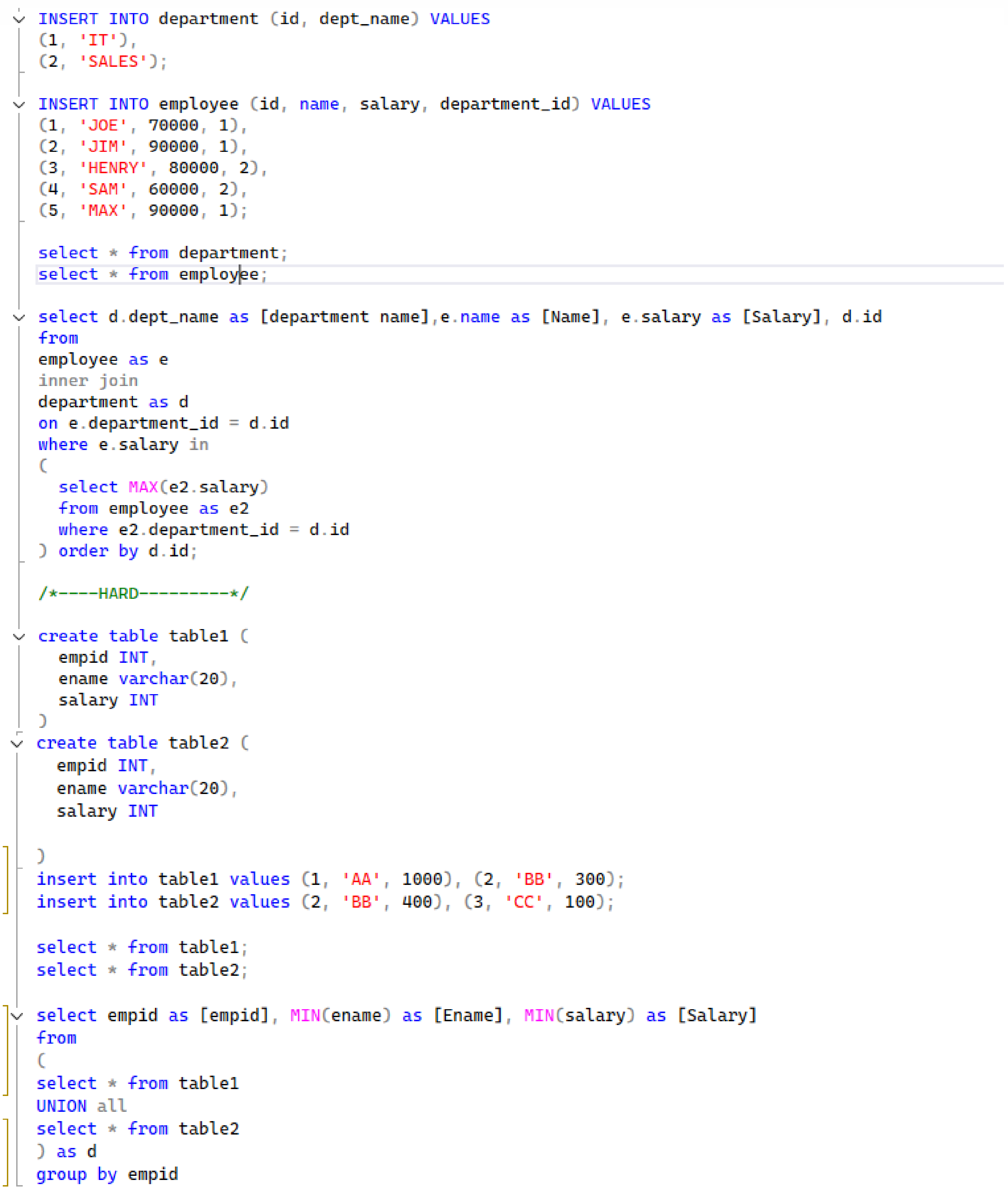
Objective:

1. Combine two tables A and B.
2. Return each EmpID with their lowest salary, and the corresponding Ename.

2. **Tools Used:** SQL Server Management Studio
3. **Code:**

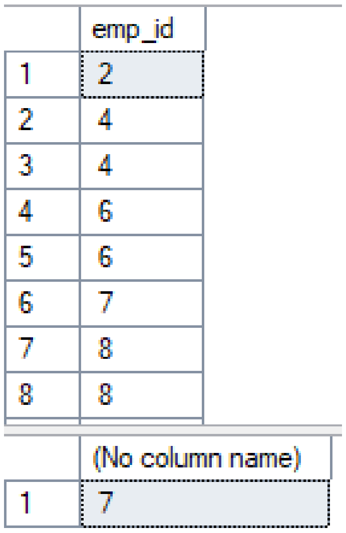


```
/*------EASY-------*/
CREATE TABLE employ(
    emp_id INT
);

INSERT INTO employ values (2),(4),(4),(6),(6),(7),(8),(8),(8);

select MAX(e.emp_id)
from employ as e
where emp_id not in(
    select Max(e1.emp_id)
    from employ as e1
    group by e1.emp_id
    having count(*) > 1
)


/*----MEDIUM-----*/
CREATE TABLE department (
    id INT PRIMARY KEY,
    dept_name VARCHAR(50)
);
CREATE TABLE employee (
    id INT,
    name VARCHAR(50),
    salary INT,
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES department(id)
);
```



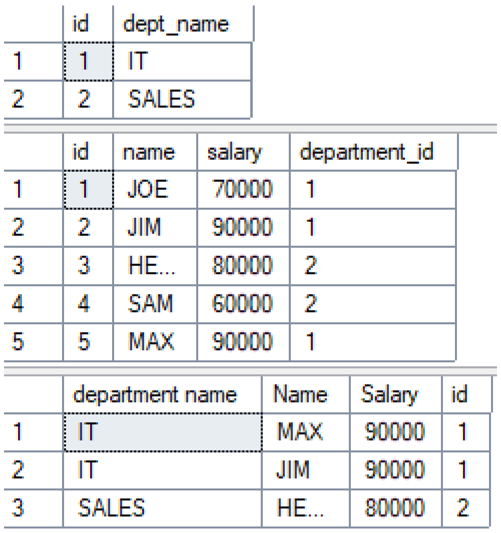
```
INSERT INTO department (id, dept_name) VALUES
(1, 'IT'),
(2, 'SALES');

INSERT INTO employee (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);

select * from department;
select * from employee;

select d.dept_name as [department name],e.name as [Name], e.salary as [Salary], d.id
from
employee as e
inner join
department as d
on e.department_id = d.id
where e.salary in
(
  select MAX(e2.salary)
  from employee as e2
  where e2.department_id = d.id
) order by d.id;

/*----HARD---------*/

create table table1 (
  empid INT,
  ename varchar(20),
  salary INT
)
create table table2 (
  empid INT,
  ename varchar(20),
  salary INT

)
insert into table1 values (1, 'AA', 1000), (2, 'BB', 300);
insert into table2 values (2, 'BB', 400), (3, 'CC', 100);

select * from table1;
select * from table2;

select empid as [empid], MIN(ename) as [Ename], MIN(salary) as [Salary]
from
(
select * from table1
UNION all
select * from table2
) as d
group by empid
```

4. Result/Output/Writing Summary:



	emp_id
1	2
2	4
3	4
4	6
5	6
6	7
7	8
8	8

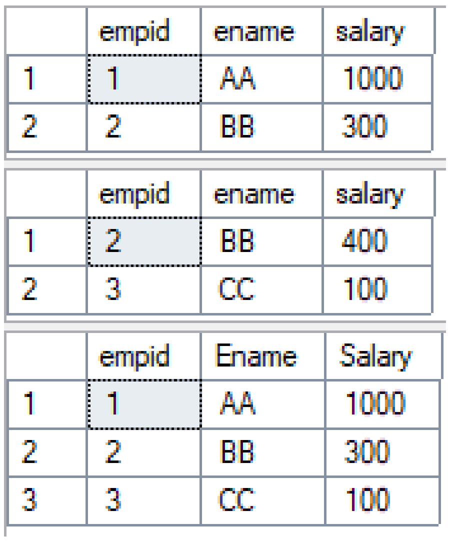
	(No column name)
1	7



	id	dept_name
1	1	IT
2	2	SALES

	id	name	salary	department_id
1	1	JOE	70000	1
2	2	JIM	90000	1
3	3	HE...	80000	2
4	4	SAM	60000	2
5	5	MAX	90000	1

	department name	Name	Salary	id
1	IT	MAX	90000	1
2	IT	JIM	90000	1
3	SALES	HE...	80000	2



	empid	ename	salary
1	1	AA	1000
2	2	BB	300

	empid	ename	salary
1	2	BB	400
2	3	CC	100

	empid	Ename	Salary
1	1	AA	1000
2	2	BB	300
3	3	CC	100

Learning outcomes (What I have learnt):

1. Learned to use subqueries for filtering duplicates and extracting results.
2. Practiced GROUP BY with aggregate functions for department-wise analysis.
3. Understood merging datasets using UNION ALL for combined insights.
4. Applied correlated subqueries to match names with min/max salaries.
5. Strengthened SQL problem-solving skills through real-world scenarios.