DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment 1.3

Student Name: Avin Mehla

UID: 23BAI70080

Branch: BE-AIT-CSE

Section/Group: 23AML-1 (A)

Semester: 5th

Date of Performance: 20 August 2025

Subject Name: ADBMS

Subject Code: 23CSP-333

1. **Experiment Name:**
   To understand and apply SQL concepts such as keys, joins, subqueries, and set operations for effective data retrieval and analysis.

2. **Objective:**

Medium-Level Problem

Problem Title: Top Earners in Each Department Using Joins and Aggregates

Procedure (Step-by-Step):

1. Create two tables:
   - Departments(DeptID, DeptName)
   - Employees(EmpID, EmpName, Salary, DeptID [foreign key referencing Departments]).
2. Insert at least 10–12 records into the Employees table, ensuring:
   - Multiple employees belong to the same department.
   - Some employees share the same highest salary in a department.
3. Write a query using JOIN to connect employees with their department names.
4. Use a subquery or window function to determine the maximum salary within each department.
5. Select the department name, employee name, and salary of only those employees whose salary matches the maximum salary of their department.
6. Order the result set by department name for clarity.

Hard-Level Problem

Problem Title: Merging Legacy HR Systems and Finding Lowest Salary per Employee

Procedure (Step-by-Step):

1. Create two tables to represent the legacy systems:
    - System A (EmpID, Ename, Salary)
    - System B (EmpID, Ename, Salary)
2. Insert at least 6–8 employee records into both tables, ensuring:
    - Some employees appear in both systems (overlap).
    - Some employees appear only in one system.
    - Salaries may differ for the same employee across systems.
3. Use UNION (or UNION ALL) to merge records from both tables into a single combined dataset.
4. For each EmpID, find the minimum salary across the merged dataset.
5. Select and display the EmpID, Employee Name, and Lowest Salary.
6. Order the results by EmpID for clarity.

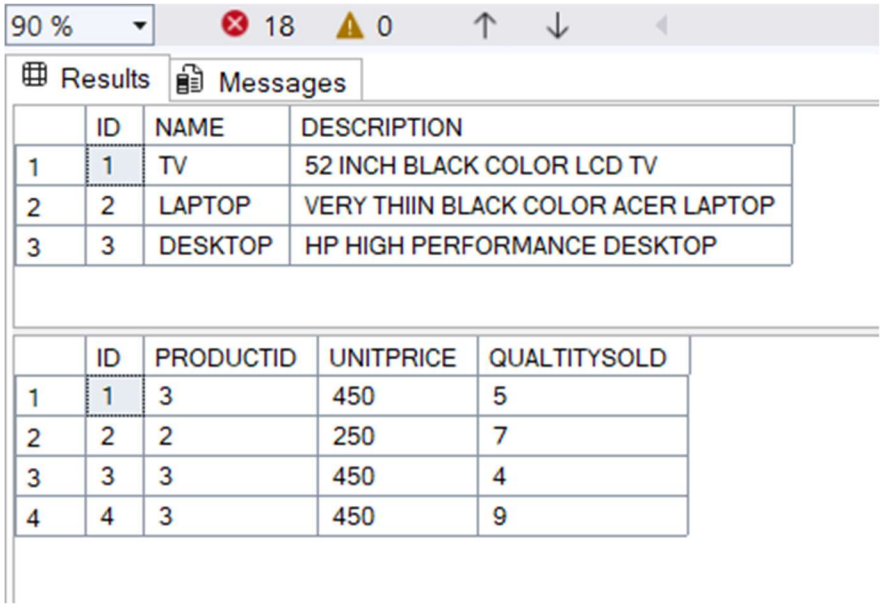
3. **Code:**

```
--EASY LEVEL--

CREATE TABLE E (EMPID INT)
INSERT INTO E VALUES (2), (4), (4), (6), (6), (7), (8), (8)
SELECT MAX(EMPID) AS [EMPID] FROM E WHERE EMPID NOT IN
(SELECT EMPID FROM E GROUP BY EMPID HAVING COUNT (*) >1)

CREATE TABLE TBL_PRODUCTS
(
    ID INT PRIMARY KEY IDENTITY,
    [NAME] NVARCHAR(50),
    [DESCRIPTION] NVARCHAR(250)
)

CREATE TABLE TBL_PRODUCTSALES
(
    ID INT PRIMARY KEY IDENTITY,
    PRODUCTID INT FOREIGN KEY REFERENCES TBL_PRODUCTS(ID),
    UNITPRICE INT,
    QUALTITYSOLD INT
)

INSERT INTO TBL_PRODUCTS VALUES ('TV','52 INCH BLACK COLOR LCD TV')
INSERT INTO TBL_PRODUCTS VALUES ('LAPTOP','VERY THIIN BLACK COLOR
ACER LAPTOP')
```

```
INSERT INTO TBL_PRODUCTS VALUES ('DESKTOP','HP HIGH PERFORMANCE DESKTOP')
INSERT INTO TBL_PRODUCTSALES VALUES (3,450,5)
INSERT INTO TBL_PRODUCTSALES VALUES (2,250,7)
INSERT INTO TBL_PRODUCTSALES VALUES (3,450,4)
INSERT INTO TBL_PRODUCTSALES VALUES (3,450,9)

SELECT *FROM TBL_PRODUCTS
SELECT *FROM TBL_PRODUCTSALES

SELECT P.ID, P.NAME, P.DESCRIPTION FROM TBL_PRODUCTS AS P WHERE P.ID NOT IN
(SELECT DISTINCT(S.PRODUCTID) FROM TBL_PRODUCTSALES AS S )

SELECT [NAME] ,
(SELECT SUM(QUALTITYSOLD) FROM TBL_PRODUCTSALES WHERE PRODUCTID = TBL_PRODUCTS.ID ) AS [TOTAL]
FROM TBL_PRODUCTS

 --MEDIUM LEVEL--
CREATE TABLE department (
   id INT PRIMARY KEY,
   dept_name VARCHAR(50)
);

-- Create Employee Table
CREATE TABLE employee (
   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 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 employee
SELECT * FROM department

SELECT D.DEPT_NAME,  E.NAME ,  E.salary
FROM
EMPLOYEE AS E
INNER JOIN
department AS D
ON
D.ID = E.department_id
WHERE salary IN
(
    SELECT MAX(SALARY)
    FROM
    employee AS E2
    WHERE E2.department_id = E.department_id
)
ORDER BY D.DEPT_NAME

--HARD--

CREATE TABLE A
(   EMPID INT PRIMARY KEY,
    ENAME VARCHAR(MAX),
    SALARY INT
)

CREATE TABLE B
(   EMPID INT PRIMARY KEY,
    ENAME VARCHAR(MAX),
    SALARY INT
```

```
)
INSERT INTO A VALUES (1,'AA', 5000), (2,'BB', 3000)
INSERT INTO B VALUES (2, 'BB', 7000), (3, 'CC', 4000)

SELECT * FROM A
SELECT * FROM B

SELECT EMPID , MIN(ENAME) AS ENAME , MIN(SALARY) AS SALARY
FROM
(
    SELECT * FROM A
    UNION ALL
    SELECT * FROM B
) AS INTERMEDIATE_RESULT
GROUP BY EMPID
```



	ID	NAME	DESCRIPTION
1	1	TV	52 INCH BLACK COLOR LCD TV
2	2	LAPTOP	VERY THIIN BLACK COLOR ACER LAPTOP
3	3	DESKTOP	HP HIGH PERFORMANCE DESKTOP

	ID	PRODUCTID	UNITPRICE	QUALTITYSOLD
1	1	3	450	5
2	2	2	250	7
3	3	3	450	4
4	4	3	450	9

90 % 18 0

Results Messages

	ID	NAME	DESCRIPTION
1	1	TV	52 INCH BLACK COLOR LCD TV

	NAME	TOTAL
1	TV	NULL
2	LAPTOP	7
3	DESKTOP	18

90 % 18 0

Results Messages

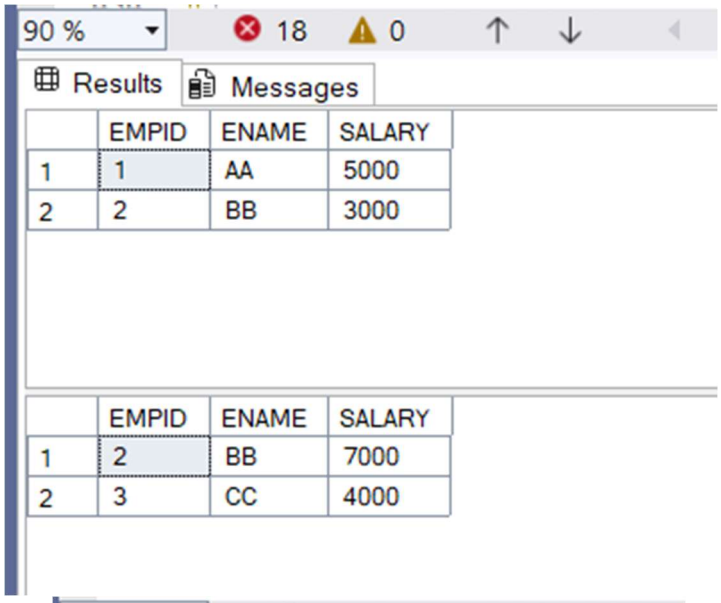
	id	name	salary	department_id
1	1	JOE	70000	1
2	2	JIM	90000	1
3	3	HENRY	80000	2
4	4	SAM	60000	2
5	5	MAX	90000	1

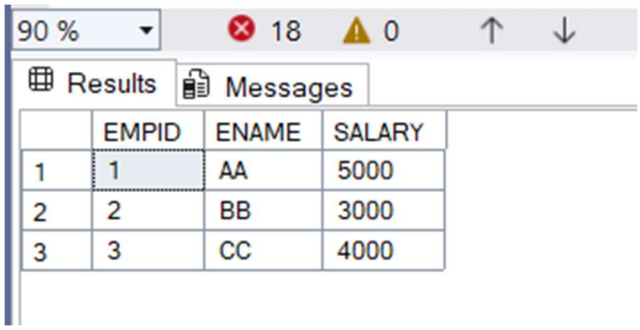
	id	dept_name
1	1	IT
2	2	SALES

90 % 18 0

Results Messages

	DEPT_NAME	NAME	salary
1	IT	MAX	90000
2	IT	JIM	90000
3	SALES	HENRY	80000





4. Learning Outcomes:

- Understand and implement **self-joins** and **foreign key relationships** for hierarchical data within the same table.
- Practiced **aggregate functions & subqueries** (MAX, SUM, COUNT).
- Applied **joins** to combine data across tables.
- Used UNION ALL and GROUP BY for data merging and summarisation.
- Improved **problem-solving** from easy (subqueries) → medium (joins) → hard (set operations).