

Experiment 1.3

Student Name: Harsh UID: 23BAI70474

Branch: BE-AIT-CSE Section/Group: 23AML-1 (B)

Semester: 5th Date of Performance: 19 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')

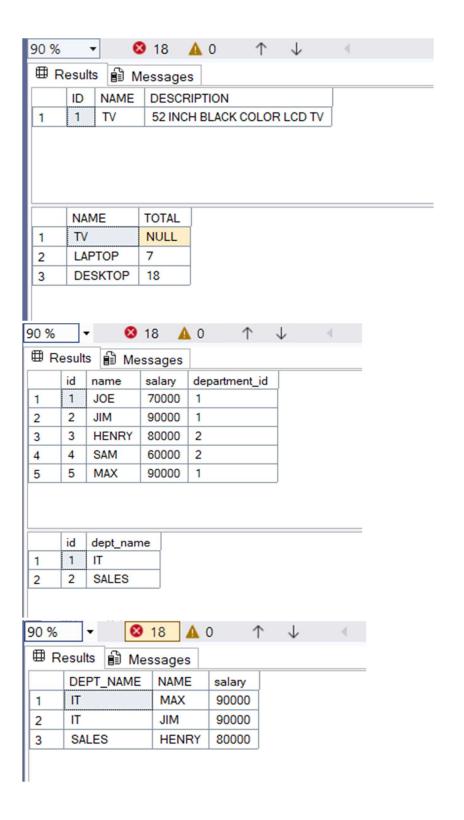
```
INSERT INTO TBL PRODUCTS VALUES ('DESKTOP', 'HP HIGH PERFORMANCE
DESKTOP')
INSERT INTO TBL PRODUCTSALES VALUES (3,450,5)
INSERT INTO TBL PRODUCTSALES VALUES (2,250,7)
INSERT INTO TBL PRODUCTSALES VALUES (3,450,4)
INSERT INTO TBL PRODUCTSALES VALUES (3,450,9)
SELECT *FROM TBL PRODUCTS
SELECT *FROM 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');
```

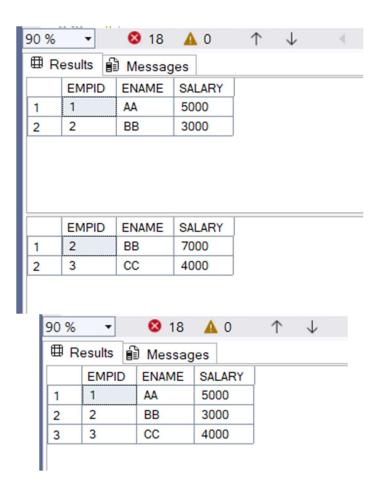
```
-- Insert into Employee Table
INSERT INTO employee (id, name, salary, department id) VALUES
(1, 'JOE', 70000, 1),
(2, 'JIM', 90000, 1),
(3, 'HENRY', 80000, 2),
(4, 'SAM', 60000, 2),
(5, 'MAX', 90000, 1);
SELECT * FROM employee
SELECT * FROM department
SELECT D.DEPT_NAME, E.NAME, E.salary
FROM
EMPLOYEE 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
90 %
             ② 18 ▲ 0 ↑ ↓

    ⊞ Results  
    ■ Messages

     ID NAME
                  DESCRIPTION
                  52 INCH BLACK COLOR LCD TV
     1
        TV
 2
         LAPTOP
                  VERY THIIN BLACK COLOR ACER LAPTOP
         DESKTOP | HP HIGH PERFORMANCE DESKTOP
         PRODUCTID
                    UNITPRICE
                              QUALTITYSOLD
                    450
                              5
     1
         3
                              7
                    250
                    450
 3
         3
 4
         3
                    450
```





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).