Experiment 3

Student Name: Yogesh Kohli
UID: 23BCS10291
Branch: BE CSE
Section/Group: KRG-3-B
Semester: 5
Date of Performance: 13-9-25
Subject Name: ADBMS
Subject Code: 23CSP-333

1. Aim:

Max Value without Duplicates [EASY]

- Create a table of Employee IDs.
- Insert sample IDs (with duplicates).
- Write a query to return the maximum EmpID excluding duplicate values using subqueries.

Department Salary Champions [MEDIUM]

- Create dept and employee tables with a relationship.
- Insert sample department and employee data.
- Use subqueries to find the employee(s) with the highest salary in each department.
- If multiple employees share the max salary in a department, include all.

Merging Employee Histories: Who Earned Least? [HARD]

- Create two legacy tables (TableA and TableB).
- Insert sample records (some overlapping).
- Merge both tables and find the minimum salary per employee using subqueries.

2. Tools Used:

Microsoft SQL server

3. Procedure:

EASY PROBLEM SOLUTION:

```
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);
```

MEDIUM PROBLEM SOLUTION:

```
CREATE TABLE Departments (
   DeptID INT PRIMARY KEY,
   DeptName VARCHAR(50)
);

CREATE TABLE Employees (
   EmpID INT PRIMARY KEY,
   EmpName VARCHAR(100),
   DeptID INT,
   Salary DECIMAL(10, 2),
   FOREIGN KEY (DeptID) REFERENCES Departments(DeptID)
);
INSERT INTO Departments (DeptID, DeptName) VALUES
(1, 'Sales'),
(2, 'Engineering'),
(3, 'HR');

INSERT INTO Employees (EmpID, EmpName, DeptID, Salary) VALUES
(101, 'Alice', 1, 70000),
(102, 'Bob', 1, 85000),
(103, 'Charlie', 2, 90000),
(104, 'David', 2, 90000),
(105, 'Eve', 3, 60000),
(106, 'Frank', 3, 58000);
```

```
SELECT
    E.EmpID,
    E.EmpName,
    E.DeptID,
    D.DeptName,
    E.Salary
FROM
    Employees E
JOIN
    Departments D ON E.DeptID = D.DeptID
WHERE
    E.Salary = (
        SELECT MAX(Salary)
        FROM Employees
        WHERE DeptID = E.DeptID
    )
ORDER BY
    E.DeptID, E.EmpName;
```

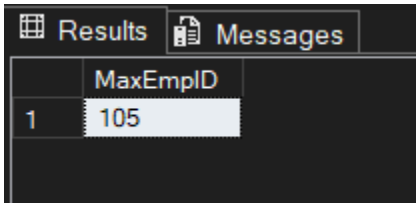
HARD PROBLEM SOLUTION:

```
CREATE TABLE TableA (
 EmpID INT,
EmpName VARCHAR(100),
Salary DECIMAL(10, 2)
);

CREATE TABLE TableB (
    EmpID INT,
    EmpName VARCHAR(100),
    Salary DECIMAL(10, 2)
);

INSERT INTO TableA (EmpID, EmpName, Salary) VALUES
(101, 'Alice', 70000),
(102, 'Bob', 85000),
(103, 'Charlie', 90000),
(104, 'David', 78000);

INSERT INTO TableB (EmpID, EmpName, Salary) VALUES
(102, 'Bob', 82000),
(103, 'Charlie', 91000),
```

```
(105, 'Eve', 60000),
(106, 'Frank', 58000);

SELECT
    M.EmpID,
    M.EmpName,
    M.Salary
FROM
(

    SELECT EmpID, EmpName, Salary FROM TableA
    UNION ALL
    SELECT EmpID, EmpName, Salary FROM TableB
) AS M
WHERE
    M.Salary = (

        SELECT MIN(Salary)
        FROM
        (
            SELECT Salary FROM TableA WHERE EmpID = M.EmpID
            UNION ALL
            SELECT Salary FROM TableB WHERE EmpID = M.EmpID
        ) AS Salaries
    )
ORDER BY M.EmpID;
```

4. Output:

Easy:



Medium:

Results | Messages

	EmpID	EmpName	DeptID	DeptName	Salary
1	102	Bob	1	Sales	85000.00
2	103	Charlie	2	Engineering	90000.00
3	104	David	2	Engineering	90000.00
4	105	Eve	3	HR	60000.00

Hard:

Results | Messages

	EmpID	EmpName	Salary
1	101	Alice	70000.00
2	102	Bob	82000.00
3	103	Charlie	90000.00
4	104	David	78000.00
5	105	Eve	60000.00
6	106	Frank	58000.00

5. Learning Outcome:

- Learn to create and define relational database tables using the CREATE TABLE command, along with understanding common data types such as INT and VARCHAR.
- Build practical skills in setting up primary keys to ensure each record can be uniquely identified.
- Understand how to define and enforce foreign key constraints to preserve data consistency between linked tables (e.g., Books linked to Authors).
- Gain the ability to perform INNER JOIN operations to merge records from multiple tables using a shared key (such as author_id).
- Learn how to structure normalized relational schemas with foreign key relationships for real-world examples like departments and courses.
- Become comfortable inserting several rows into related tables using the INSERT INTO statement.
- Master the use of subqueries alongside GROUP BY and HAVING to summarize and filter aggregated results.
- Apply query logic to select data from a parent table based on conditions derived from aggregated results in a related child table.