DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CHANDIGARH UNIVERSITY — Discover. Learn. Empower.

Experiment 3

EASY

- **Aim:**
  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
- **Theory:**
  Subqueries are queries written inside another SQL query to provide intermediate results that can be used by the outer query. They are useful for filtering, transforming, or aggregating data when a single query alone cannot achieve the desired result. Subqueries can be used in different clauses such as SELECT, FROM, and WHERE, and may return a single value, a single column, or even an entire table.
  Based on their return values, subqueries can be classified into different types: single-value subqueries (returning one scalar value), single-column subqueries (returning one column with multiple values), and table subqueries (returning multiple rows and columns).
  Subqueries are commonly used with operators such as =, IN, EXISTS, ANY, and ALL for filtering and comparison. They are also powerful when combined with aggregate functions like COUNT, SUM, or MAX, enabling advanced data analysis and reporting.
- **Queries:**

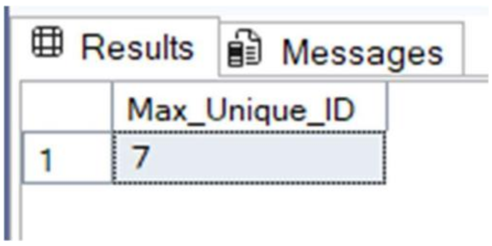
```
USE Experiments;

--EASY
CREATE TABLE Employee (
  Emp_ID INT
);

INSERT INTO Employee (Emp_ID) VALUES
(2), (4), (4), (6), (6), (7), (8), (8), (8);

--Finding max Emp_ID excluding duplicates
SELECT MAX(Emp_ID) AS Max_Unique_ID
FROM Employee
WHERE Emp_ID IN (
    SELECT Emp_ID
    FROM Employee
    GROUP BY Emp_ID
    HAVING COUNT(*) = 1
);
```

- **Output:**



	Max_Unique_ID
1	7

- **Results:**
The query successfully identified and excluded duplicate employee IDs using subqueries and then retrieved the maximum value among the unique IDs. The output confirmed that the highest unique Emp_ID in the table is **7**.
- **Learning Outcomes:**
    1. I have learnt the concept of subqueries and how they can be used inside other SQL queries.
    2. I have learnt how to use GROUP BY and HAVING to identify unique or duplicate values in a dataset.
    3. I have learnt how to combine subqueries with aggregate functions such as MAX() to extract meaningful results.
    4. I have learnt the difference between single-value, single-column, and table subqueries.
    5. I have learnt how subqueries make complex queries easier by breaking them into smaller logical parts.

Experiment 3

MEDIUM

- **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.
- **Theory:**
  Subqueries and joins play an important role when analyzing relational data across multiple tables. A **subquery** can be used to find the maximum salary within each department, and then the outer query can retrieve the employee details that match those maximum values. Alternatively, **joins** allow data from multiple tables, such as employee and department records, to be combined for meaningful results.
  The concept of **correlated subqueries** is particularly useful in such cases, where the subquery depends on the outer query for filtering results on a per-department basis. Additionally, aggregate functions such as `MAX()` help identify the highest salary in each department; while grouping and ordering allow for structured, department-wise outputs. These techniques are commonly applied in real-world business analytics to recognize top performers and ensure fair representation when multiple employees share the same highest salary.
- **Queries:**

```
USE Experiments;

CREATE TABLE TBL_EMP_SALARY (
  Emp_ID INT Primary Key,
  Emp_Name VARCHAR(MAX),
  Salary INT,
  Dept_ID VARCHAR(MAX)
);

CREATE TABLE TBL_DEPT_SALARY (
Dept_ID INT Primary Key,
Dept_Name VARCHAR(MAX)
);
```

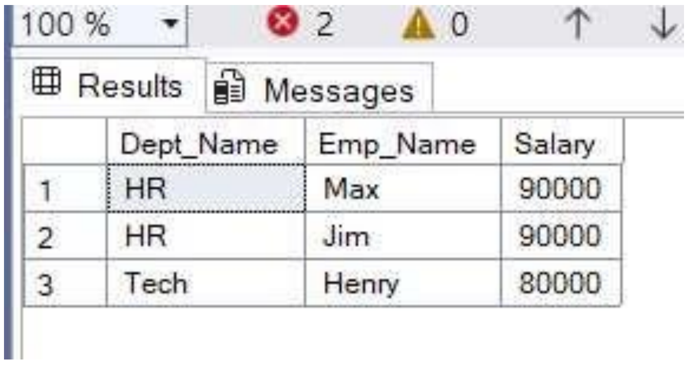
```
INSERT INTO TBL_EMP_SALARY(Emp_ID, Emp_Name, Salary, Dept_ID)
VALUES
(1, 'Joe', 70000, 1),
(2, 'Jim', 90000, 1),
(3, 'Henry', 80000,2),
(4, 'Sam', 60000,2),
(5, 'Max', 90000,1);

INSERT INTO TBL_DEPT_SALARY(Dept_ID, Dept_Name)
Values
(1,'HR'),
(2, 'Tech');

Select * from TBL_EMP_SALARY;
Select * from TBL_DEPT_SALARY;

Select D.Dept_Name, E.Emp_Name, E.Salary
From TBL_EMP_SALARY AS E
Inner Join
TBL_DEPT_SALARY AS D
ON
E.Dept_ID = D.Dept_ID
Where E.Salary IN
(
Select Max(Salary)
From TBL_EMP_SALARY AS E2
Where E2.Dept_ID = E.Dept_ID
)
Order by D.Dept_Name;
```

- **Output:**



	Dept_Name	Emp_Name	Salary
1	HR	Max	90000
2	HR	Jim	90000
3	Tech	Henry	80000

- **Results:**
  The query successfully identified the top earners from each department. In cases of ties, multiple employees with the highest salary were included in the output, ensuring fairness. The final report displayed department names alongside their top-performing employees and their salaries.
- **Learning Outcomes:**
  1. I have learnt how to use subqueries with aggregate functions like `MAX()` to filter top results.
  2. I have learnt the concept of correlated subqueries that depend on outer query values.
  3. I have learnt how to combine multiple tables using `JOIN` to get meaningful results.
  4. I have learnt how to handle ties in queries to ensure all top values are included.
  5. I have learnt how to structure query results with `ORDER BY` for better readability.

Experiment 3

HARD

- **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.

  Objective
  1. Combine two tables A and B.
  2. Return each EmpID with their lowest salary, and the corresponding Ename.
- **Theory:**
  This experiment demonstrates the use of **UNION** and **aggregate functions** in SQL. The `UNION` operator is used to combine rows from multiple tables into a single dataset, eliminating duplicates unless `UNION ALL` is specified. This is especially useful when integrating records from different legacy systems.
  Once the data is combined, aggregate functions such as `MIN()` can be applied to identify the lowest value in a group. The `GROUP BY` clause groups records by employee ID, and `MIN(Salary)` ensures the lowest salary for each employee is selected. To retrieve the employee name alongside the salary, either a subquery or a join is applied with the aggregated result.
  This combination of `UNION`, `GROUP BY`, and `MIN()` is frequently used in data migration and consolidation scenarios where the goal is to merge historical data while preserving key insights.
- **Queries:**

```
Use Experiments;

Create Table TBL_A(
        Emp_ID INT Primary Key,
        Ename Varchar(MAX),
        Salary INT
);

Create Table TBL_B(
        Emp_ID INT Primary Key,
        Ename Varchar(MAX),
        Salary INT
);
```

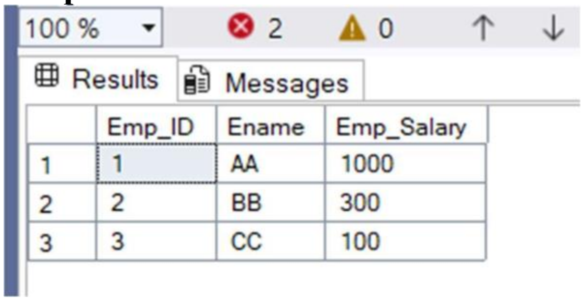
```
Insert Into TBL_A (Emp_ID, Ename, Salary)
Values
(1, 'AA', 1000),
(2, 'BB', 300);

Insert Into TBL_B(Emp_ID, Ename, Salary)
Values
(2, 'BB', 400),
(3, 'CC', 100);

Select * from TBL_B;

SELECT Emp_ID, Ename, Min(Salary) as Emp_Salary
From
        (       Select * from TBL_A
                UNION
                Select * from TBL_B
        ) AS Inter_Mediate
Group BY Emp_ID, Ename;
```

- **Output:**



	Emp_ID	Ename	Emp_Salary
1	1	AA	1000
2	2	BB	300
3	3	CC	100

- **Results:**
The query successfully merged data from both HR systems and identified the lowest salary for each employee. In cases where employees appeared in both tables, the lower salary was chosen. Employees unique to one system were also preserved.
- **Learning Outcomes:**
    1. I have learnt how to use the UNION operator to merge records from multiple tables.
    2. I have learnt the difference between UNION and UNION ALL.
    3. I have learnt how to use aggregate functions like MIN() with GROUP BY.
    4. I have learnt how to preserve unique employee records when consolidating multiple data sources.