CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

Experiment 1

Student Name: Gaurav Monga

UID: 23BAI70156

Branch: BE-AIT-CSE

Section/Group: 23AIT_KRG-1

Semester: 5th

Date of Performance: 23 July, 2025

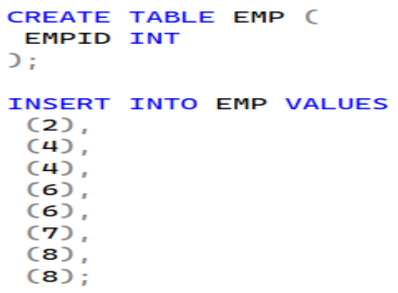
Subject Name: ADBMS

Subject Code: 23CSP-333

EASY - LEVEL

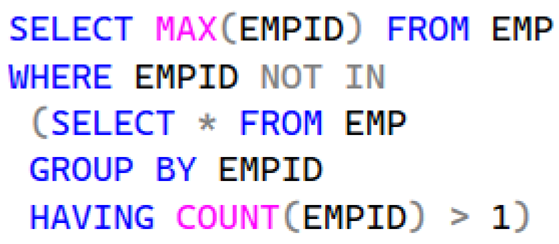
1. **Problem Statement:** You are given a table named EMPLOYEE with a single column, EMP_ID. The table contains multiple entries, some of which are duplicates. Your task is to write a single query that finds the maximum value among only those EMP_IDs that appear exactly once. You must solve this problem using a subquery.
2. **Procedure (Step-by-Step):**
   a) The inner query (SELECT EMPID FROM EMP GROUP BY EMPID HAVING COUNT(EMPID) = 1) groups the EMP table by EMPID and uses the HAVING clause to filter for groups where the count is exactly one. This effectively isolates the unique EMPIDs.
   b) The outer query (SELECT MAX(EMPID) FROM EMP WHERE EMPID IN (...)) then takes the result of the subquery and finds the maximum value from that set.
3. **SQL Commands:**

   a. Create Table EMP and insert values into it:



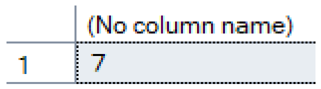
```
CREATE TABLE EMP (
 EMPID INT
);

INSERT INTO EMP VALUES
 (2),
 (4),
 (4),
 (6),
 (6),
 (7),
 (8),
 (8);
```

b. Using aggregate functions and subqueries to get the desired result:



```sql
SELECT MAX(EMPID) FROM EMP
WHERE EMPID NOT IN
 (SELECT * FROM EMP
 GROUP BY EMPID
 HAVING COUNT(EMPID) > 1)
```

4. **Output:**



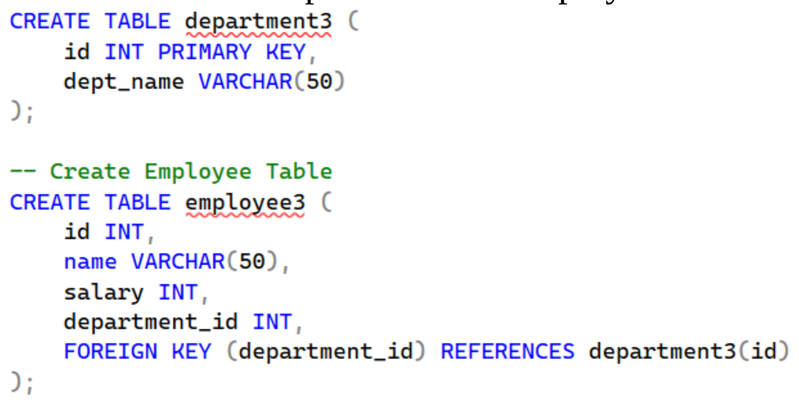
	(No column name)
1	7

Output of our query to get the maximum emp_id having count 1.

5. **Learning Outcome:**
    a. I learnt about subqueries in SQL.
    b. I learnt about aggregate functions in SQL.
    c. I learnt about the uses of the two in SQL.

MEDIUM - LEVEL

1. **Problem Statement:** 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.
2. **Procedure (Step-by-Step):**
   a. An INNER JOIN connects the employee and department tables on their common department_id, allowing the final output to include department names.
   b. The WHERE clause contains a subquery (SELECT MAX(e2.salary) FROM employee AS e2 WHERE e2.department_id = e.department_id). This subquery is "correlated" because it references a column (e.department_id) from the outer query. For each employee in the outer query, the subquery calculates the maximum salary for that employee's specific department.
   c. The outer query then filters the joined result set, keeping only those employees whose salary matches the MAX(salary) for their department, thus identifying the top earners.Design normalized tables for departments and the courses they offer, maintaining a foreign key relationship.
3. **SQL Commands:**
   a. Create the tables Department and Employee and insert values into them:



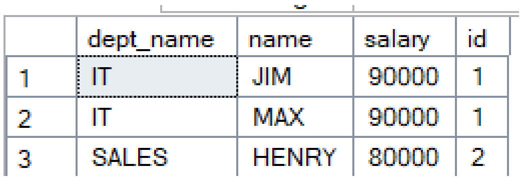
```
CREATE TABLE department3 (
    id INT PRIMARY KEY,
    dept_name VARCHAR(50)
);

-- Create Employee Table
CREATE TABLE employee3 (
    id INT,
    name VARCHAR(50),
    salary INT,
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES department3(id)
);
```

```sql
-- Insert into Department Table
INSERT INTO department3 (id, dept_name) VALUES
(1, 'IT'),
(2, 'SALES');

-- Insert into Employee Table
INSERT INTO employee3 (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);
```

b. Use inner join in association with where clause and subqueries to get our desired output:

```sql
SELECT D.dept_name, E.name, E.salary, D.id
FROM
department3 AS D
INNER JOIN
employee3 AS E
ON
D.id = E.department_id
WHERE E.salary IN
(
    SELECT MAX(E2.salary)
    FROM employee3 AS E2
    WHERE E2.department_id = E.department_id
    GROUP BY E2.department_id
)
ORDER BY D.dept_name
```

4. **Output:**



	dept_name	name	salary	id
1	IT	JIM	90000	1
2	IT	MAX	90000	1
3	SALES	HENRY	80000	2

Output of Employees having highest salary in their departments

5. **Learning Outcomes:**

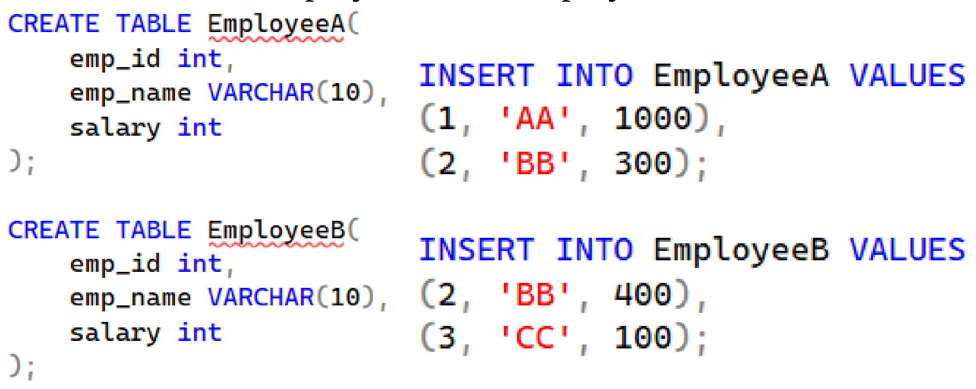
a. Learnt about the use of inner join in association with subqueries.

b. Learnt about the use of Group by and order by clauses

c. Learnt about the use of aliases.

HARD - LEVEL

1. **Problem Statement:** 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.
2. **Procedure (Step-by-Step):**
   a. The inner query (SELECT * FROM TABLEA UNION ALL SELECT * FROM TABLEB) combines all rows from both tables into a single, temporary result set. Using UNION ALL is crucial here because it includes all records, even if an employee appears in both tables, which is necessary to check for the lowest salary across both records.
   b. The outer query (SELECT EMPID, MIN(ENAME) AS ENAME, MIN(SALARY) AS SALARY FROM ... GROUP BY EMPID) then performs the main analysis. It groups the combined data by EMPID.
   c. Within each group, it applies the MIN() function to the SALARY column, effectively finding the lowest salary recorded for that employee across both tables. MIN(ENAME) is used to select a name for each EMPID since it's required for the output, and in this case, the MIN function can arbitrarily select one of the potentially identical names.
3. **SQL Commands:**
   a. Create the tables Employee A and Employee B and insert values into them:

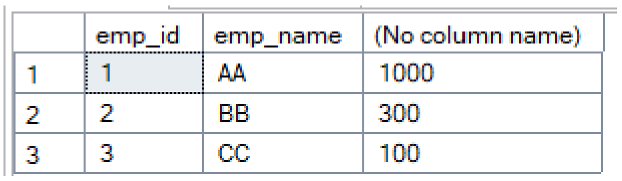


```
CREATE TABLE EmployeeA(
    emp_id int,
    emp_name VARCHAR(10),
    salary int
);

INSERT INTO EmployeeA VALUES
(1, 'AA', 1000),
(2, 'BB', 300);

CREATE TABLE EmployeeB(
    emp_id int,
    emp_name VARCHAR(10),
    salary int
);

INSERT INTO EmployeeB VALUES
(2, 'BB', 400),
(3, 'CC', 100);
```

   b. Use Union and subqueries to get the desired output:

```
SELECT emp_id, emp_name, MIN(SALARY)
FROM(
    SELECT * FROM EmployeeA
    UNION
    SELECT * FROM EmployeeB
)AS Intermediate_result
GROUP BY emp_id, emp_name;
```

4. **Output:**



	emp_id	emp_name	(No column name)
1	1	AA	1000
2	2	BB	300
3	3	CC	100

Output of Non-Duplicate employees with max salary

5. **Learning Outcomes:**

   a. Learnt about the use of set functions in association with subqueries.

   b. Learnt about the use of union and union all

   c. Learnt about the use of group by clause.