University Institute of Engineering

Department of Computer Science & Engineering

EXPERIMENT: 3

NAME	: Johnson Kumar	**UID**	: 23BCS12654
SECTION	: KRG_2A	**SEMESTER**: 5TH	
SUBJECT CODE: 23CSP-339		**SUBJECT**	: ADBMS

I. Aim Of The Practical :

[MEDIUM] **Department Salary Champions**

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.

[HARD] **Merging Employee Histories: Who Earned Least?**

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.

II. Tools Used: SQL Server Management Studio

III. Code:

```
----------MEDIUM LEVEL: Department Salary Champions
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);

--solution
select dept_name, name, salary from
employee as e
inner join
department as d
on e.department_id=d.id
where e.salary in(
    select max(e2.salary)
    from employee as e2
    where e2.department_id=e.department_id)
order by d.dept_name


----------HARD LEVEL: Merging Employee Histories: Who Earned Least?
create table tablea(empid int, ename varchar(30),salary int)
insert into tablea values
(1,'aa',1000),
(2,'bb',300);

create table tableb(empid int, ename varchar(30), salary int)
insert into tableb values
(2,'bb',400),
(3,'cc',200);

select empid, ename, min(salary) from
(select * from tablea union all select * from tableb) as t
group by empid, ename
```

University Institute of Engineering

Department of Computer Science & Engineering

Output :

[MEDIUM]

Tables:

	id	dept_name
1	1	IT
2	2	SALES

.....Department table

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

.....Employee table

Final:

Results | Messages

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

.....Department-wise max salary

[HARD]

Tables:

	empid	ename	salary
1	1	aa	1000
2	2	bb	300

.....First Employee table

	empid	ename	salary
1	2	bb	400
2	3	cc	200

.....Second Employee table

Final:

Results | Messages

	empid	ename	(No column name)
1	1	aa	1000
2	2	bb	300
3	3	cc	200

.....Combined table with min salary

University Institute of Engineering

Department of Computer Science & Engineering

IV. Learning Outcomes :

- I learned how to design normalized relational schemas using primary and foreign keys to maintain data integrity.
- I practiced creating and linking tables to represent departments and employees in a relational database.
- I used INNER JOIN operations to combine data across related tables and extract meaningful insights.
- I applied correlated subqueries to filter data based on grouped conditions, such as identifying the highest salary within each department.
- I retrieved and sorted data using WHERE clauses and ORDER BY statements to enhance query readability.
- I simulated real-world HR scenarios by identifying top earners in each department using SQL logic.
- I merged datasets from multiple sources using UNION ALL while preserving duplicate records.
- I used GROUP BY in combination with MIN() to determine the lowest salary earned by each employee across different tables.
- I handled overlapping employee records and resolved salary discrepancies through aggregation techniques.
- I mimicked payroll audits and historical salary tracking by consolidating employee data from separate tables.