# Experiment 1.3

**Student Name:** Subhradip Majumder          **UID:** 23BAI70059

**Branch:** BE-AIT-CSE          **Section/Group:** 23AML_KRG-1

**Semester:** 5th          **Date of Performance:** 21 July 2025

**Subject Name:** ADBMS          **Subject Code:** 23CSP-333

| MEDIUM - LEVEL |
|---|

1. **Problem Title:** Department Salary Champions
2. **Problem Description:** 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.

1. **SQL Commands:**
   a. Create the tables and insert values.

```sql
CREATE TABLE department (
    id INT PRIMARY KEY,
    dept_name VARCHAR(50)
);
CREATE TABLE employee (
    id INT,
    name VARCHAR(50),
    salary INT,
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES department(id)
);

INSERT INTO department (id, dept_name) VALUES
(1, 'IT'),
(2, 'SALES');
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);
```

b.  Use a subquery get the department wise max salary.

```
select s.name, s.salary, s.department_id, d.dept_name
from employee s
inner join department d on d.id = s.department_id
where s.salary in
(select max(e.salary) Max_sal
from employee e
group by department_id)
order by department_id;
```

## 2. Output:

| | Name | Owner | Type | Created_datetime |
|---|---|---|---|---|
| 1 | employee | dbo | user table | 2025-08-20 09:43:18.250 |

| | Column_name | Type | Computed | Length | Prec | Scale | Nullable | TrimTrailingBlanks | FixedLenNullInSource | Collation |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int | no | 4 | 10 | 0 | yes | (n/a) | (n/a) | NULL |
| 2 | name | varchar | no | 50 | | | yes | no | yes | SQL_Latin1_General_CP1_CI_AS |
| 3 | salary | int | no | 4 | 10 | 0 | yes | (n/a) | (n/a) | NULL |
| 4 | department_id | int | no | 4 | 10 | 0 | yes | (n/a) | (n/a) | NULL |

| | Identity | Seed | Increment | Not For Replication |
|---|---|---|---|---|
| 1 | No identity column defined. | NULL | NULL | NULL |

| | RowGuidCol |
|---|---|
| 1 | No rowguidcol column defined. |

| | Data_located_on_filegroup |
|---|---|
| 1 | PRIMARY |

| | constraint_type | constraint_name | delete_action | update_action | status_enabled | status_for_replication | constraint_keys |
|---|---|---|---|---|---|---|---|
| 1 | FOREIGN KEY | FK__employee__depart__73BA3083 | No Action | No Action | Enabled | Is_For_Replication | department_id |
| 2 | | | | | | | REFERENCES sql_query.dbo.department (id) |

*Figure 1 Employee Table*

| | Name | Owner | Type | Created_datetime |
|---|---|---|---|---|
| 1 | department | dbo | user table | 2025-08-20 09:43:18.240 |

| | Column_name | Type | Computed | Length | Prec | Scale | Nullable | TrimTrailingBlanks | FixedLenNullInSource | Collation |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int | no | 4 | 10 | 0 | no | (n/a) | (n/a) | NULL |
| 2 | dept_name | varchar | no | 50 | | | yes | no | yes | SQL_Latin1_General_CP1_CI_AS |

| | Identity | Seed | Increment | Not For Replication |
|---|---|---|---|---|
| 1 | No identity column defined. | NULL | NULL | NULL |

| | RowGuidCol |
|---|---|
| 1 | No rowguidcol column defined. |

| | Data_located_on_filegroup |
|---|---|
| 1 | PRIMARY |

| | index_name | index_description | index_keys |
|---|---|---|---|
| 1 | PK__departme__3213E83F686ED361 | clustered, unique, primary key located on PRIMARY | id |

| | constraint_type | constraint_name | delete_action | update_action | status_enabled | status_for_replication | constraint_keys |
|---|---|---|---|---|---|---|---|
| 1 | PRIMARY KEY (clustered) | PK__departme__3213E83F686ED361 | (n/a) | (n/a) | (n/a) | (n/a) | id |

*Figure 2 Department Table*

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

*Figure 3 Output*

3. **Learning Outcomes:**

   a. I learned how to perform join with the subquery.
   b. I understood how the subqueries actually work.
   c. I learnt how different joins works with subquery.

4. **Problem Title:** Merging Employee Histories: Who Earned Least?
5. **Problem Description:** 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:
    a.   Combine two tables A and B.
    b.  Return each EmpID with their lowest salary, and the corresponding Ename.
6. **SQL Commands:**
    a.  Create the tables.

```sql
create table A(
    id int,
    ename varchar(5),
    salary int);
create table B(
    id int,
    ename varchar(5),
    salary int);
insert into A values
(1,'AA',1000),
(2,'BB',300);
insert into B values
(2,'BB',400),
(3,'CC',100);
```

    b.  Use a subquery get the enames with min salary.

```sql
select id, ename, Min(salary) as salary
from (
    select id, ename, salary from A
    union
    select id, ename, salary from B
) as combined
group by id, ename;
```

7. **Output:**

| Name | Owner | Type | Created_datetime |
|---|---|---|---|
| 1 | A | dbo | user table | 2025-08-20 10:10:57.657 |

| | Column_name | Type | Computed | Length | Prec | Scale | Nullable | TrimTrailingBlanks | FixedLenNullInSource | Collation |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int | no | 4 | 10 | 0 | yes | (n/a) | (n/a) | NULL |
| 2 | ename | varchar | no | 5 | | | yes | no | yes | SQL_Latin1_General_CP1_CI_AS |
| 3 | salary | int | no | 4 | 10 | 0 | yes | (n/a) | (n/a) | NULL |

| | Identity | Seed | Increment | Not For Replication |
|---|---|---|---|---|
| 1 | No identity column defined. | NULL | NULL | NULL |

| | RowGuidCol |
|---|---|
| 1 | No rowguidcol column defined. |

| | Data_located_on_filegroup |
|---|---|

*Figure 1 A table*

| | Name | Owner | Type | Created_datetime |
|---|---|---|---|---|
| 1 | B | dbo | user table | 2025-08-20 10:11:08.533 |

| | Column_name | Type | Computed | Length | Prec | Scale | Nullable | TrimTrailingBlanks | FixedLenNullInSource | Collation |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int | no | 4 | 10 | 0 | yes | (n/a) | (n/a) | NULL |
| 2 | ename | varchar | no | 5 | | | yes | no | yes | SQL_Latin1_General_CP1_CI_AS |
| 3 | salary | int | no | 4 | 10 | 0 | yes | (n/a) | (n/a) | NULL |

| | Identity | Seed | Increment | Not For Replication |
|---|---|---|---|---|
| 1 | No identity column defined. | NULL | NULL | NULL |

| | RowGuidCol |
|---|---|
| 1 | No rowguidcol column defined. |

| | Data_located_on_filegroup |
|---|---|
| 1 | PRIMARY |

*Figure 2 B table*

| | id | ename | salary |
|---|---|---|---|
| 1 | 1 | AA | 1000 |
| 2 | 2 | BB | 300 |
| 3 | 3 | CC | 100 |

*Figure 3 Output*

8. **Learning Outcomes:**

   a. I learned how to perform union with the subquery.
   b. I learned some of the build functions of the Microsoft SQL server.
   c. I learned about aliases in the SQL queries.