



**University Institute of Engineering**  
**Department of Computer Science & Engineering**

### **EXPERIMENT:3**

**NAME :Aishwary Singh**

**BRANCH : BE-CSE**

**SEMESTER : 5<sup>TH</sup>**

**SUBJECT NAME : ADBMS**

**UID : 23BCS11568**

**SECTION : KRG\_3B**

**SUBJECT : 23CSP-339**

#### **1. AIM:-**

You are given an EMP table that contains a list of employee IDs (EMP\_ID). Some employee IDs may appear multiple times, representing duplicate entries.

Write an SQL query (using **subqueries**) to:

- Identify and exclude all employee IDs that appear more than once in the table.
- From the remaining unique employee IDs, find the **highest employee ID**.

Return the result as a single column named single\_highest.

**Software Used** -SQL Management Studio

#### **Source Code**

```
create database subquery;
```

```
use subquery;
```

```
create table Emp(Emp_id int);
```

```
insert into Emp(Emp_id)
```

```
values
```

```
(2),
```

```
(4),
```

```
(4),
```

```
(6),
```

```
(6),
```

```
(7),
```

```
(8),
```

```
(8);
```

```
SELECT MAX(Emp_Id) AS [single_highest]FROM Emp
```

```
WHERE Emp_id NOT IN
```

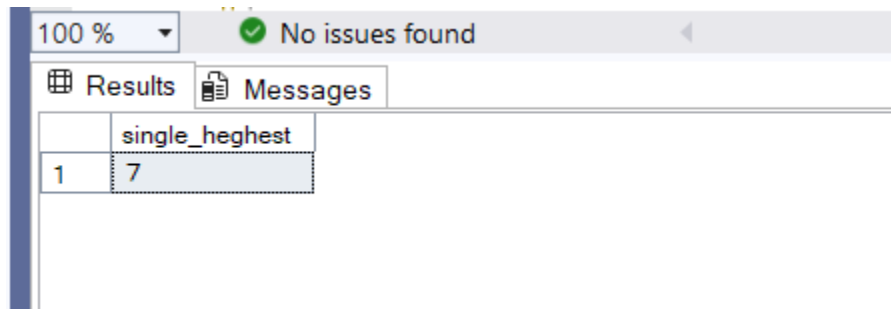
```
(
```

```
SELECT Emp_id FROM Emp
```

```
GROUP BY Emp_id
```

```
HAVING COUNT(EMP_ID)>1)
```

## Output



The screenshot shows the SQL Management Studio interface. At the top, there is a status bar indicating '100 %' zoom and 'No issues found'. Below this, there are two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with two columns: 'single\_highest' and an unnamed column. The first row of the table contains the values '1' and '70000'.

	single_highest
1	70000

## Medium Level

**Aim-** Given tables:

- department(id, dept\_name)
- employee(id, name, salary, department\_id)

Write a SQL query to retrieve employees with the highest salary in each department, displaying their name, salary, and department name, sorted by department name.

**Software Used-**SQL Management Studio

## Source Code

```
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);  
  
SELECT E.name, E.salary, D.dept_name, D.id  
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

```

## Output

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

## Hard Level

### Aim

Given tables:

- TABLE1(EMPID, Ename, Salary)
- TABLE2(EMPID, Ename, Salary)

Write a SQL query to combine the records from both tables, and for each EMPID, select the employeename and salary with the minimum values. The result should display one row per EMPID.

**Software Used**-SQL Management Studio

### Source Code

```

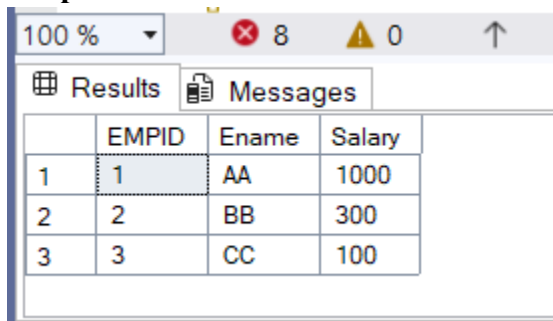
CREATE TABLE TABLE1(
    EMPID INT,
    Ename VARCHAR(20),
    Salary INT
)
CREATE TABLE TABLE2(
    EMPID INT,
    Ename VARCHAR(20),
    Salary INT
)
INSERT INTO TABLE1(EMPID,Ename,Salary) VALUES
(1,'AA',1000),
(2,'BB',300);

```

```
INSERT INTO TABLE2(EMPID,Ename,Salary) VALUES  
(2,'BB',400),  
(3,'CC',100);
```

```
SELECT EMPID,min(Ename) as Ename,MIN(Salary) as Salary  
FROM  
(  
SELECT *FROM TABLE1  
UNION  
SELECT *FROM TABLE2)  
AS RES  
GROUP BY EMPID
```

## Output



	EMPID	Ename	Salary
1	1	AA	1000
2	2	BB	300
3	3	CC	100

## Learning Outcomes

- Acquired hands-on experience in **creating databases, tables, and inserting data**.
- Practiced writing **subqueries** for advanced filtering and data aggregation.
- Gained proficiency in using **JOINS** to combine and analyze data from multiple tables.
- Learned techniques to manage **duplicates** and consolidate results using **UNION** and **aggregate functions**.
- Strengthened **problem-solving skills** in retrieving, interpreting, and presenting specific information from datasets.