



WORKSHEET 3

Student Name: Supriya Dutta

UID: 23BCS13291

Branch: CSE(3rd Year)

Section/Group: Krg-1-A

Semester: 5th

Date of Performance: 14/08/25

Subject Name: ADBMS

Subject Code: 23CSP-333

1. AIM:

[MEDIUM]

To write and execute a SQL query that returns the maximum EMP_ID value from the Employee table **excluding any EMP_IDs that appear more than once** (i.e., only consider IDs that occur exactly once), and to verify the result with sample data.

Objectives

- Understand grouping and filtering duplicate rows using GROUP BY and HAVING.
- Use a subquery in the WHERE clause to select only non-duplicated EMP_IDs.
- Return the maximum of those unique EMP_IDs using an aggregate function.

Expected outcome

- The query should return 7 for the provided sample data (since 7 is the largest EMP_ID that occurs exactly once).

[HARD]

To use subqueries to (a) identify products that have never been sold, and (b) compute total quantity sold per product by embedding an aggregate subquery in the SELECT clause; demonstrate both queries on the provided TBL_PRODUCTS and TBL_PRODUCTSALES sample data.

Objectives

- Write a NOT IN (or LEFT JOIN-based) subquery to list product ID, NAME, and DESCRIPTION for products with no sales records.
- Use a scalar subquery in the SELECT list to compute QTY_SOLD = SUM(QUALITYSOLD) per product.
- Handle NULL (no-sales) results appropriately if needed (e.g., understanding NULL vs 0 in results).

Expected outcome

- A result set listing the product(s) never sold (their id, name, description).
- A result set that shows each product name and the corresponding total quantity sold (NULL or 0 for products with no sales).

2. Tools Used : SQL Server Management Studio

DBMS SCRIPT:

--Q1: Medium Level

```
CREATE TABLE Employee(
  EMP_ID INT
)
```

```
INSERT into Employee(EMP_ID) values (2)
INSERT into Employee(EMP_ID) values (4)
INSERT into Employee(EMP_ID) values (4)
INSERT into Employee(EMP_ID) values (6)
INSERT into Employee(EMP_ID) values (6)
INSERT into Employee(EMP_ID) values (7)
INSERT into Employee(EMP_ID) values (8)
INSERT into Employee(EMP_ID) values (8)
```

-- return the max empid excluding the duplicates using subqueries e.g in this case 7

```
SELECT Max(EMP_ID) from Employee
where EMP_ID IN
(
  select EMP_ID from Employee
  group by EMP_ID
  having count(*)=1
)
```

--Q2: Hard Level

```
CREATE TABLE TBL_PRODUCTS
(
  ID INT PRIMARY KEY IDENTITY,
  [NAME] NVARCHAR(50),
  [DESCRIPTION] NVARCHAR(250)
)
```

```
CREATE TABLE TBL_PRODUCTSALES
(
  ID INT PRIMARY KEY IDENTITY,
```

```
PRODUCTID INT FOREIGN KEY REFERENCES TBL_PRODUCTS(ID),
UNITPRICE INT,
QUALITYSOLD INT
)
```

```
INSERT INTO TBL_PRODUCTS VALUES ('TV','52 INCH BLACK COLOR LCD TV')
INSERT INTO TBL_PRODUCTS VALUES ('LAPTOP','VERY THIN BLACK COLOR ACER
LAPTOP')
INSERT INTO TBL_PRODUCTS VALUES ('DESKTOP','HP HIGH PERFORMANCE
DESKTOP')
```

```
INSERT INTO TBL_PRODUCTSALES VALUES (3,450,5)
INSERT INTO TBL_PRODUCTSALES VALUES (2,250,7)
INSERT INTO TBL_PRODUCTSALES VALUES (3,450,4)
INSERT INTO TBL_PRODUCTSALES VALUES (3,450,9)
```

```
SELECT *FROM TBL_PRODUCTS
SELECT *FROM TBL_PRODUCTSALES
```

--Task : Find the id , name , description of product which has not been sold for once
--output: id, name, description

```
SELECT p.ID, p.NAME, p.DESRIPTION
FROM TBL_PRODUCTS p
WHERE p.ID NOT IN (
    SELECT PRODUCTID
    FROM TBL_PRODUCTSALES
);
```

--Task 2: Find the total quantity sold for w=each respective products
--output: name Qty_Sold(SUM)
--You will use subquery in select clause

```
SELECT
    p.NAME,
    (
        SELECT SUM(s.QUALITYSOLD)
        FROM
            TBL_PRODUCTSALES s
        WHERE
            s.PRODUCTID = p.ID
    ) AS QTY_SOLD
FROM TBL_PRODUCTS p;
```

3. OUTPUT:

--Medium Level:

Results		Messages
	(No column name)	
1	7	

--Medium Level:

Results				Messages
	ID	NAME	DESCRIPTION	
1	1	TV	52 INCH BLACK COLOR LCD TV	

	NAME	QTY_SOLD
1	TV	NULL
2	LAPTOP	7
3	DESKTOP	18

4. Learning Outcomes:

- Students will be able to **apply subqueries in SQL** to filter and retrieve specific data from relational tables.
- Students will learn to **use GROUP BY and HAVING clauses** to handle duplicate values and enforce conditions on aggregated data.
- Students will understand how to **identify unmatched records** (e.g., products not sold) using NOT IN subqueries.
- Students will gain experience in **calculating aggregated results** such as total quantity sold per product using subqueries in the SELECT clause.

- Students will develop the ability to **design and test SQL queries on sample datasets**, strengthening their skills in database problem-solving and query optimization.