University Institute of Engineering

Department of Computer Science & Engineering

EXPERIMENT : 3

NAME : GURVEER SINGH MANGAT UID: 23BCS11074

BRANCH : BE-CSE SECTION/GROUP : KRG_1A

SEMESTER : 5TH SUBJECT CODE : 23CSP-333

SUBJECT NAME : ADBMS

1. Aim Of The Practical :

Max Value without Duplicates [EASY]

- Create a table of Employee IDs.
- Insert sample IDs (with duplicates).
- Write a query to return the maximum EmpID excluding duplicate values using subqueries.

Department Salary Champions [MEDIUM]

- Create `dept` and `employee` tables with a relationship.
- Insert sample department and employee data.
- Use subqueries to find the employee(s) with the highest salary in each department.
- If multiple employees share the max salary in a department, include all.

Merging Employee Histories: Who Earned Least? [HARD]

- Create two legacy tables (TableA and TableB).
- Insert sample records (some overlapping).
- Merge both tables and find the minimum salary per employee using subqueries.

2. Tools Used: SQL Server Management Studio

3. Code:

```sql
--easy question
 --easy question
/*
GENERATE AN EMPLOYEE RELATIN WITH ONLY A ONE ATTRIBUTE I.E, EMP_ID
TASK: DIND THE MAX EMP_ID, BUT EXCLUDING THE DUPLICATES
*/
CREATE TABLE EMPLOYEE(
EMPID INT
);
INSERT INTO EMPLOYEE(EMPID) VALUES
(1),
(1),
(2),
(2),
(5),
(5),
(6),
(7),
(8),
(8);
SELECT MAX(EMPID) AS [MAX_UNIQUE]
FROM Employee WHERE EmpID IN
(SELECT EmpID FROM Employee GROUP BY EmpID HAVING count(EmpID)=1);



--medium question
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);

--main
select d.dept_name,e.name,e.salary from employee as e
inner join department as d on d.id=e.department_id
```

```sql
where e.salary in(
select MAX(e2.salary)
from employee as e2
where e2.department_id=e.department_id
)
order by  dept_name;

-- group by approach
select d.dept_name,e.name,e.salary from employee as e
inner join department as d on d.id=e.department_id
where e.salary in(
select MAX(e2.salary)
from employee as e2
group by e2.department_id
);

--hard question
CREATE TABLE TableA (
    Empid INT,
    Ename VARCHAR(50),
    Salary INT
);

CREATE TABLE TableB (
    Empid INT,
    Ename VARCHAR(50),
    Salary INT
);

INSERT INTO TableA VALUES (1, 'AA', 1000), (2, 'BB', 300);
INSERT INTO TableB VALUES (2, 'BB', 400), (3, 'CC', 100);

--TIP; AFF OVER NUMBER DATA ONLY IS WRONG
--TAKE FIRST LETTER OF EMPNAME WILL CONVERT IN ASCII
select empid, ename ,MIN(salary) AS salary from(
select * from tableA as a
union all
select * from tableB as b
) as INTERMIDIATE_RESULT
group by empid,ename;
```

4. Output :

[EASY]

	MAX_UNIQUE
1	7

[MEDIUM]

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

[HARD]

	empid	ename	salary
1	1	AA	1000
2	2	BB	300
3	3	CC	100

5. Learning Outcomes:

- Learn to create and define relational database tables using the CREATE TABLE command, along with understanding common data types such as `INT` and VARCHAR.
- Build practical skills in setting up primary keys to ensure each record can be uniquely identified.
- Understand how to define and enforce foreign key constraints to preserve data consistency between linked tables (e.g., Books linked to Authors).
- Gain the ability to perform INNER `JOIN` operations to merge records from multiple tables using a shared key (such as author_id).
- Learn how to structure normalized relational schemas with foreign key relationships for real-world examples like departments and courses. • Become comfortable inserting several rows into related tables using the INSERT INTO statement.
- Master the use of subqueries alongside GROUP BY and HAVING to summarize and filter aggregated results.
- Apply query logic to select data from a parent table based on conditions derived from aggregated results in a related child table.