Correlated Query

The inner query depends on value from outer query

Ex. Find employees who earn more than the average salary of their department.

```
Select E1.EMPNO, E1.empname, E1.sal, E1.deptno
From emp E1
Where E1.sal > (
        SELECT AVG(E2.sal)
        FROM EMP E2
        Where E1.DEPTNO = E2.DEPTNO
);
```

Outer Query: It retrieves employees whose salaries are higher than the department’s average salary.

Inner Query: It Calculates the average salary for the department of current employee(e1.deptno).

Correlation(E1.DEPTNO = E2.DEPTNO): Links the inner query to the current row of the outer query.

1. Outer Query

```
Select E1.EMPNO, E1.empname, E1.sal, E1.deptno
From emp E1
Where E1.sal >(………)
```

E1.EMPNO - Retrieves the employee number
E1.EMPNAME - Retrieves the employee name
E1.sal - Retrieves the salary
E1.deptno - Retrieves the department no
From emp E1 - Refers to the emp table, assigning it an alias E1 for easier reference
Where E1.sal >(…) - The condition checks if the employees salary (E1.sal) is greater than the average salary of their department(calculated by subquery)

2. Subquery

```
SELECT AVG(E2.sal)
         FROM EMP E2
         Where E1.DEPTNO = E2.DEPTNO
```

AVG(E2.SAL) - Calculates the average salary(sal) of employees
FROM EMP E2 - refers to the same Emp table but assigns it an alias E2 to distinguish it from E1 in the outer query.
Where E1.DEPTNO = E2.DEPTNO - Filters the records to consider only those employees(E2) who are in the same department as current employee(E1)

The subquery returns the average salary of the department to which the current employee(E1) belongs to.

FLOW:

1. Outer query initialization
   The outer query starts selecting the data from emp table with alias as E1.
2. Subquery execution
   For each employee(E1), the subquery calculates average salary of all employees(E2) within the same department(E1.DEPTNO = E2.DEPTNO)

- The where clause compares the salary of the current employee(E1.Sal) with the average salary returned by the subquery.

- If E1.sal is greater than the average salary of that department, then the record will be included in our resultset.

DATE Functions

DATEDIFF(): calculates the number of days between two dates.

Ex. To find the experience in years from employee table.

```
SELECT EmpNAME, EmpNo,
floor(datediff(curdate(),hiredate)/365) As Experience
From emp;
```

```
mysql> SELECT EmpNAME, EmpNo,
floor(datediff(curdate(),hiredate)/365) As Experience
    -> From emp;
+---------+-------+------------+
| EmpNAME | EmpNo | Experience |
+---------+-------+------------+
| SMITH   |  7369 |         44 |
| ALLEN   |  7499 |         44 |
| WARD    |  7521 |         44 |
| JONES   |  7566 |         43 |
| MARTIN  |  7654 |         43 |
| BLAKE   |  7698 |         43 |
| CLARK   |  7782 |         43 |
| SCOTT   |  7788 |         42 |
| KING    |  7839 |         43 |
| TURNER   |  7844 |          43 |
| ADAMS    |  7876 |          42 |
| JAMES    |  7900 |          43 |
| FORD     |  7902 |          43 |
| MILLER   |  7934 |          43 |
+----------+-------+-------------+
```

5. Date_Add()

Adds an interval (like days, weeks, months or years) to a given date

SYNTAX:

DATE_ADD(date, INTERVAL)

Ex. To add 1 week to current date

```
SELECT date_add(curdate(), INTERVAL 1 WEEK) as DATE;
mysql> SELECT date_add(curdate(), INTERVAL 1 WEEK) as
DATE;
+------------+
| DATE       |
+------------+
| 2025-03-25 |
+------------+
SELECT date_add(curdate(), INTERVAL 1 year) as DATE;

mysql> SELECT date_add(curdate(), INTERVAL 1 year) as
DATE;

+------------+

| DATE       |

+------------+

| 2026-03-18 |

+------------+

SELECT date_add(curdate(), INTERVAL 4 day ) as DATE;


mysql> SELECT date_add(curdate(), INTERVAL 4 day ) as
DATE;

+------------+

| DATE       |

+------------+

| 2025-03-22 |

+------------+
```

Ex. To find employees who joined in February

Month() -> this is a function to extract a month of a date

```
Select empname, hiredate

From emp
```

Where month(hiredate) = 2;

```
mysql> Select empname, hiredate
    -> From emp
    -> Where month(hiredate) = 2;
+----------+------------+
| empname | hiredate   |
+----------+------------+
| ALLEN    | 1981-02-20 |
| WARD     | 1981-02-22 |
+----------+------------+
```

TASK:

Table: Medicine

Columns: MID, Mname, price, exp_date

Requirement: find medicines expiring in 3 months.

Data Manipulation language

Insert: adds a new row to table

Update: modify the existing data in table

Delete: remove the rows from table

Update:

Syntax:

```
Update table_name
Set Column1=value1, column2=value2,……
Where condition
```

Ex. To increase the salary of employees in department no 30 by 15%.

```
Update emp
Set sal = sal*1.15
Where deptno = 30;
```

```
mysql> Update emp
    -> Set sal = sal*1.15
    -> Where deptno = 30;
Query OK, 6 rows affected (0.02 sec)
Rows matched: 6  Changed: 6  Warnings: 0
```

```
mysql> select * from emp;
+-------+---------+-----------+------+------------+---------+---------+--------+
| EMPNO | EMPNAME | JOB       | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+---------+-----------+------+------------+---------+---------+--------+
|  7369 | SMITH   | CLERK     | 7902 | 1980-12-17 |  800.00 |    NULL |     20 |
|  7499 | ALLEN   | SALESMAN  | 7698 | 1981-02-20 | 1840.00 |  300.00 |     30 |
|  7521 | WARD    | SALESMAN  | 7698 | 1981-02-22 | 1437.50 |  500.00 |     30 |
|  7566 | JONES   | MANAGER   | 7839 | 1981-04-02 | 2975.00 |    NULL |     20 |
|  7654 | MARTIN  | SALESMAN  | 7698 | 1981-09-28 | 1437.50 | 1400.00 |     30 |
|  7698 | BLAKE   | MANAGER   | 7839 | 1981-05-01 | 3277.50 |    NULL |     30 |
|  7782 | CLARK   | MANAGER   | 7839 | 1981-06-09 | 2450.00 |    NULL |     10 |
|  7788 | SCOTT   | ANALYST   | 7566 | 1982-12-09 | 3000.00 |    NULL |     20 |
|  7839 | KING    | PRESIDENT | NULL | 1981-11-17 | 5000.00 |    NULL |     10 |
|  7844 | TURNER  | SALESMAN  | 7698 | 1981-09-08 | 1725.00 |    0.00 |     30 |
|  7876 | ADAMS   | CLERK     | 7788 | 1983-01-12 | 1100.00 |    NULL |     20 |
|  7900 | JAMES   | CLERK     | 7698 | 1981-12-03 | 1092.50 |    NULL |     30 |
|  7902 | FORD    | ANALYST   | 7566 | 1981-12-03 | 3000.00 |    NULL |     20 |
|  7934 | MILLER  | CLERK     | 7782 | 1982-01-23 | 1300.00 |    NULL |     10 |
+-------+---------+-----------+------+------------+---------+---------+--------+
```

Ex.To change the job of SMITH(7369) from clerk to senior clerk.

```
Update emp
Set job =  ‘SENIOR CLERK’
Where empno = 7369;
```

DELETE Statement:

Syntax:

```
Delete from table_name
Where condition;
```

Ex. To remove employee whose empno is 7900

```
Delete from emp
Where empno = 7900;
```

Ex. Delete all employees whose salary is less than 1000.

```
Delete from emp
Where sal < 1000;
```

Display employee names with leading spaces removed and trailing periods added

(e.g., 'ALLEN' should become 'ALLEN.').

```
SELECT CONCAT(TRIM(empname),  '.' ) as empNames

From emp;
```

```
mysql> SELECT CONCAT(TRIM(empname), '.') as empNames
    -> From emp;
+----------+
| empNames |
+----------+
| SMITH.   |
| ALLEN.   |
| WARD.    |
| JONES.   |
| MARTIN.  |
| BLAKE.   |
| CLARK.   |
| SCOTT.   |
| KING.    |
| TURNER.  |
| ADAMS.   |
| JAMES.   |
| FORD.    |
| MILLER.  |
+----------+
```

JOINS:

- Creating a Department Table:

```
Create table dept (
        DEPTNO INT(2) primary key,
        DNAME VARCHAR(20),
        LOC VARCHAR(15)
);
```

- Inserting values into dept table

```
INSERT INTO DEPT VALUES(10,'ACCOUNTING','NEW YORK');
INSERT INTO DEPT VALUES(20,'RESEARCH','DALLAS');
INSERT INTO DEPT VALUES(30,'SALES','CHICAGO');
INSERT INTO DEPT VALUES(40,'OPERATIONS','BOSTON');
```

```
mysql> SELECT * FROM DEPT;
+--------+------------+----------+
| DEPTNO | DNAME      | LOC      |
+--------+------------+----------+
|     10 | ACCOUNTING | NEW YORK |
|     20 | RESEARCH   | DALLAS   |
|     30 | SALES      | CHICAGO  |
|     40 | OPERATIONS | BOSTON   |
+--------+------------+----------+
```

JOINS

It allows us to retrieve data from multiple tables based on related columns.

TYPES OF JOINS:

1. Inner Join
2. Left Join (Left Outer Join)
3. Right Join (Right Outer Join)
4. Full Join (Full Outer Join)