ASSIGNMENT NO.04

TITLE: SQL queries using SQL DML statements: all types of Join, Sub-Query

PROBLEM STATEMENT: To Design SQL queries for suitable database application using SQL DML statements: all types of Join, Sub-Query.

OBJECTIVES:

- To use database techniques such as SQL Joins, Sub-queries.
- To evaluate the appropriate use of joins and sub-queries.

OUTCOMES:

- Students will be able to write queries forgiven requirements using SQL Joins, Sub-queries
- Students will be able to evaluate when to use joins and sub-queries appropriately

SOFTWARE & HARDWARE REQUIREMENTS:

1. 64-bit Open source Linux or its derivative
2. MySQL Server

THEORY:

SQL Join is used to fetch data from two or more tables, which is joined to appear as single set of data. SQL Join is used for combining column from two or more tables by using values common to both tables. Join Keyword is used in SQL queries for joining two or more tables. Minimum required condition for joining table, is (n-1) where n, is number of tables. A table can also join to itself known as, Self Join.

Types of Join

The following are the types of JOIN that we can use in SQL.

- Inner
- Outer
- Left
- Right

Cross JOIN or Cartesian Product

This type of JOIN returns the cartesian product of rows from the tables in Join. It will return a table which consists of records which combines each row from the first table with each row of the second table.

Cross JOIN Syntax is,

```
SELECT column-name-list FROM table-name1 CROSS JOIN table-name2;
```

Example of Cross JOIN: Consider following tables

The **class** table

ID	NAME
1	ABHI
2	ADAM
4	ALEX

The **class_info** table

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

Cross JOIN query will be,

mysql> SELECT * from class cross JOIN class_info;

The result table will look like,

ID	NAME	ID	Address
1	ABHI	1	DELHI
2	ADAM	1	DELHI
4	ALEX	1	DELHI
1	ABHI	2	MUMBAI
2	ADAM	2	MUMBAI
4	ALEX	2	MUMBAI
1	ABHI	3	CHENNAI
2	ADAM	3	CHENNAI
4	ALEX	3	CHENNAI

INNER JOIN OR EQUI JOIN

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the query.

Inner Join Syntax is,

```
SELECT column-name-list from table-name1 INNER JOIN table-name2
WHERE table-name1.column-name = table-name2.column-name;
```

Example of Inner JOIN

mysql> SELECT * from class, class_info where class.id = class_info.id;

The result table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI

Natural JOIN: Natural Join is a type of Inner join which is based on column having same name and same datatype present in both the tables to be joined.
Natural Join Syntax is,

```
SELECT * from table-name1 NATURAL JOIN table-name2;
```

Natural join query will be,
mysql> SELECT * from class NATURAL JOIN class_info;
The result table will look like,

ID	NAME	Address
1	ABHI	DELHI
2	ADAM	MUMBAI
3	ALEX	CHENNAI

Outer Join is based on both matched and unmatched data. Outer Joins subdivide further into,

- Left Outer Join
- Right Outer Join
- Full Outer Join

Left Outer Join: The left outer join returns a result table with the matched data of two tables then remaining rows of the left table and null for the right table's column.
Left Outer Join syntax is,

```
SELECT column-name-list from table-name1 LEFT OUTER JOIN table-name2
on table-name1.column-name = table-name2.column-name;
```

Left Outer Join query will be,
mysql>SELECT * FROM class LEFT OUTER JOIN class_info ON (class.id=class_info.id);
The result table will look like,

ID	NAME	ID	Address
1	ABHI	1	DELHI
2	ADAM	2	MUMBAI
3	ALEX	3	CHENNAI
4	ANU	null	null
5	ASHISH	null	null

Right Outer Join: The right outer join returns a result table with the matched data of two tables then remaining rows of the right table and null for the left table's columns.
Right Outer Join Syntax is,

```
select column-name-list from table-name1 RIGHT OUTER JOIN table-name2 on
table-name1.column-name = table-name2.column-name;
```

Right Outer Join query will be,

SELECT * FROM class RIGHT OUTER JOIN class_info on (class.id=class_info.id);

The result table will look like,

ID	NAME	ID	Address
1	ABHI	1	DELHI
2	ADAM	2	MUMBAI
3	ALEX	3	CHENNAI
null	null	7	NOIDA
null	null	8	PANIPAT

Full Outer Join: The full outer join returns a result table with the matched data of two table then remaining rows of both left table and then the right table.

Full Outer Join Syntax is,

select ***column-name-list*** from ***table-name1*** FULL OUTER JOIN ***table-name2*** on ***table-name1.column-name = table-name2.column-name***;

Full Outer Join query will be like,

mysql> SELECT * FROM class FULL OUTER JOIN class_info on (class.id=class_info.id);

The result table will look like,

ID	NAME	ID	Address
1	ABHI	1	DELHI
2	ADAM	2	MUMBAI
3	ALEX	3	CHENNAI
4	ANU	null	null
5	ASHISH	null	null
null	null	7	NOIDA
null	null	8	PANIPAT

SQL SUBQUERY

Subquery or Inner query or Nested query is a query in a query. SQL subquery is usually added in the WHERE Clause of the SQL statement. Most of the time, a subquery is used when you know how to search for a value using a SELECT statement, but do not know the exact value in the database.

Subqueries are an alternate way of returning data from multiple tables.Subqueries can be used with the following SQL statements along with the comparison operators like =, <, >, >=, <= etc.

There are a few rules that subqueries must follow:

•Subqueries must be enclosed within parentheses.
•A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.
•An ORDER BY cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY can be used to perform the same function as the ORDER BY in a subquery.
•Subqueries that return more than one row can only be used with multiple value operators, such as the IN operator.
•The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.
•A subquery cannot be immediately enclosed in a set function.
•The BETWEEN operator cannot be used with a subquery; however, the BETWEEN operator can be used within the subquery.

SUBQUERIES WITH THE SELECT STATEMENT:

Subqueries are most frequently used with the SELECT statement. The basic syntax is as follows:

```
SELECT column_name [, column_name ]  FROM   table1 [, table2 ] WHERE  column_name
OPERATOR
  (SELECT column_name [, column_name ]
  FROM table1 [, table2 ]
  [WHERE]) CONDITION
```

SUBQUERIES WITH THE INSERT STATEMENT:

Subqueries also can be used with INSERT statements. The INSERT statement uses the data returned from the subquery to insert into another table. The selected data in the subquery can be modified with any of the character, date or number functions.

The basic syntax is as follows:

```
INSERT INTO table_name [ (column1 [, column2 ]) ]
        SELECT [ *|column1 [, column2 ]
        FROM table1 [, table2 ]
[ WHERE VALUE OPERATOR ]
```

Example:

Consider a table CUSTOMERS_BKP with similar structure as CUSTOMERS table. Now to copy complete CUSTOMERS table into CUSTOMERS_BKP, following is the syntax:

```
SQL> INSERT INTO CUSTOMERS_BKP SELECT * FROM CUSTOMERS
   WHERE ID IN (SELECT ID
            FROM CUSTOMERS) ;
```

SUBQUERIES WITH THE UPDATE STATEMENT:

The subquery can be used in conjunction with the UPDATE statement. Either single or multiple columns in a table can be updated when using a subquery with the UPDATE statement.

The basic syntax is as follows:

```
UPDATE table SET column_name = new_value [ WHERE OPERATOR [ VALUE ]
  (SELECT COLUMN_NAME
  FROM TABLE_NAME)
[ WHERE) ]
```

Example:

Assuming, we have CUSTOMERS_BKP table available which is backup of CUSTOMERS table. Following example updates SALARY by 0.25 times in CUSTOMERS table for all the customers whose AGE is greater than or equal to 27:

```
SQL> UPDATE CUSTOMERS SET SALARY = SALARY * 0.25
   WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP WHERE AGE >= 27);
```

SUBQUERIES WITH THE DELETE STATEMENT:

The subquery can be used in conjunction with the DELETE statement like with any other statements mentioned above. The basic syntax is as follows:

```
DELETE FROM TABLE_NAME [ WHERE OPERATOR [ VALUE ]
  (SELECT COLUMN_NAME
  FROM TABLE_NAME) [ WHERE) ]
```

Example:

Assuming, we have CUSTOMERS_BKP table available which is backup of CUSTOMERS table. Following example deletes records from CUSTOMERS table for all the customers whose AGE is greater than or equal to 27:

```
SQL> DELETE FROM CUSTOMERS WHERE AGE IN (SELECT AGE FROM
CUSTOMERS_BKP WHERE AGE >27 );
```

CONCLUSION:

QUESTIONS:

Consider Following Schema

Employee (Employee_id, First_name, last_name , hire_date, salary, Job_title, manager_id, department_id)

Departments(Department_id, Department_name, Manager_id, Location_id)

Locations(location_id ,street_address ,postal_code, city, state, country_id)

1. Write a query to find the names (first_name, last_name) and the salaries of the employees who have a higher salary than the employee whose last_name="Singh".

2. Write a query to find the names (first_name, last_name) of the employees who have a manager and work for a department based in the United States.

3. Find the names of all employees who works in the IT department.

4. Write a query to find the names (first_name, last_name), the salary of the employees whose salary is greater than the average salary.

5. Write a query to find the names (first_name, last_name), the salary of the employees who earn more than the average salary and who works in any of the IT departments.

6. Write a query to find the names (first_name, last_name), the salary of the employees who earn the same salary as the minimum salary for all departments.

7. Write a query to display the employee ID, first name, last names, salary of all employees whose salary is above average for their departments.

8. Write a query to find the employee id, name (last_name) along with their manager_id, manager name (last_name).

9. Find the names and hire date of the employees who were hired after 'Jones'.

10. Write a query to get the department name and number of employees in the department.