**Assignment No:2**

**Title:** Write at least 10 SQL queries for a suitable database application using SQL DML statements that demonstrate all types of joins, sub-queries, and views.

**Problem Statement:** Create a set of at least 10 SQL queries for a specific database application, utilizing SQL DML statements such as Insert, Select, Update, and Delete, along with operators, functions, set operators, joins, sub-queries, and views to efficiently manage and query data.

**Objective:** To acquire knowledge of database query languages.

**Outcome:** Implement different SQL and PLSQL operations using suitable databases.

**Tools Required:** Ubuntu OS, Mysql.

**Theory:**

**1. Introduction:**

**1.1 DML: Data Manipulation Language** (DML) statements are used for managing data within schema objects. Some examples:

* SELECT - retrieve data from the a database
* INSERT - insert data into a table
* UPDATE - updates existing data within a table
* DELETE - deletes all records from a table, the space for the records remain

**1.1.1 Insert:** Insert command is used to insert data into a table. Following is its general syntax:

|  |  |
| --- | --- |
| **Syntax:**  INSERT INTO table\_name VALUES (value1,value2,value3,...);  OR  INSERT INTO table\_name (column1,column2,column3,...) VALUES (value1,value2,value3,...); | **Example**:  INSERT INTO Customers (CustomerName, City, Country) VALUES ('Cardinal', 'Stavanger', 'Norway'); |

**1.1.2 Delete:** Delete command is used to delete data from a table. Delete command can also be used with condition to delete a particular row. Following is its general syntax:

|  |  |
| --- | --- |
| **Syntax:**  DELETE from table-name where condition; | **Example**:  DELETE from Student; where address=’pune’; |

**1.1.3 Update:** Update command is used to update a row of a table. Following is its general syntax;

|  |  |
| --- | --- |
| **Syntax:**  UPDATE table-name set column-name = value where condition; | **Example**:  update Student set age=18 where s\_id=102; |

**2. SQL Functions:** SQL provides many built-in functions to perform operations on data. These functions are useful while performing mathematical calculations, string concatenations, sub-strings etc.

**2.1 AVG():** Average returns average value after calculating from values in a numeric column.

Its general Syntax is as below:

|  |  |
| --- | --- |
| **Syntax:**  SELECT AVG(column\_name) from table\_name; | **Example**:  SELECT avg(salary) from Emp; |

**2.2 COUNT():** Count returns the number of rows present in the table either based on some condition or without condition. Its general Syntax is as below:

|  |  |
| --- | --- |
| **Syntax:**  SELECT COUNT(column\_name) from table-name; | **Example**:  SELECT COUNT(name) from Emp where salary = 8000; |

**2.3 MAX():** MAX function returns maximum value from selected column of the table.

Syntax of MAX function is as below.

|  |  |
| --- | --- |
| **Syntax:**  SELECT MAX(column\_name) from table-name | **Example**:  SELECT MAX(salary) from emp; |

**2.4 MIN():** MIN function returns minimum value from selected column of the table.

Syntax of MIN function is as below.

|  |  |
| --- | --- |
| **Syntax:**  SELECT MIN(column\_name) from table-name | **Example**:  SELECT MIN(salary) from emp; |

**2.5 SUM():** SUM function returns total sum of a selected columns numeric values. Syntax for SUM is as below.

|  |  |
| --- | --- |
| **Syntax:**  SELECT SUM(column\_name) from table-name | **Example**:  SELECT SUM(salary) from emp; |

**3. Operators:**

AND and OR operators are used with Where clause to make more precise conditions for fetching

data from database by combining more than one condition together. AND operator is used to set multiple conditions with Where clause. OR operator is also used to combine multiple conditions with Where clause. The only difference between AND and OR is their behaviour. When we use AND to combine two or more than two conditions, records satisfying all the condition will be in the result. But in case of OR, atleast one condition from the conditions specified must be satisfied by any record to be in the result.

|  |
| --- |
| SELECT \* from Emp WHERE salary < 10000 AND age > 25  SELECT \* from Emp WHERE salary > 10000 OR age > 25 |

**4. Set Operation in SQL**

SQL supports few Set operations to be performed on table data. These are used to get meaningful

results from data, under different special conditions.

**4.1 Union**

UNION is used to combine the results of two or more Select statements. However it will eliminate duplicate rows from its result set. In case of union, number of columns and datatype must be same in both the tables.

|  |
| --- |
| select \* from FirstTable  UNION  select \* from SecondTable; |

**4.2 Intersect**

Intersect operation is used to combine two SELECT statements, but it only retuns the records

which are common from both SELECT statements. In case of Intersect the number of columns

and datatype must be same. MySQL does not support INTERSECT operator.

|  |
| --- |
| select \* from FirstTable  INTERSECT  select \* from SecondTable; |

**4.3 Minus**

Minus operation combines result of two Select statements and return only those result which

belongs to first set of result. MySQL does not support INTERSECT operator.

|  |
| --- |
| select \* from FirstTable  MINUS  select \* from SecondTable; |

**5. SQL Joins:**

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

different types of the JOINs in SQL are as below:

**(INNER) JOIN:** Returns records that have matching values in both tables

**LEFT (OUTER) JOIN:** Returns all records from the left table, and the matched records from the right table

**RIGHT (OUTER) JOIN:** Returns all records from the right table, and the matched records from the left table

**FULL (OUTER) JOIN:** Returns all records when there is a match in either left or right table

|  |  |  |
| --- | --- | --- |
| SQL INNER JOIN | SQL LEFT JOIN | SQL RIGHT JOIN |
|  | SQL FULL OUTER JOIN |  |

**Fig 1.** Types of Joins

**5.1 INNER JOIN:**

The INNER JOIN keyword selects records that have matching values in both tables.

**Table 1.** Product Table

|  |  |  |  |
| --- | --- | --- | --- |
| **ProductID** | **ProductName** | **CategoryID** | **Price** |
| 1 | Chais | 1 | 18 |
| 2 | Chang | 1 | 19 |
| 3 | Aniseed Syrup | 2 | 10 |

**Table 2.** Category Table

|  |  |  |
| --- | --- | --- |
| **CategoryID** | **CategoryName** | **Description** |
| 1 | Beverages | Soft drinks, coffees, teas, beers, and ales |
| 2 | Condiments | Sweet and savory sauces, relishes, spreads, and seasonings |
| 3 | Confections | Desserts, candies, and sweet breads |

|  |
| --- |
| **Syntax:**  SELECT ProductID, ProductName, CategoryName  FROM Products  INNER JOIN Categories ON Products.CategoryID = Categories.CategoryID; |

**5.2 LEFT JOIN**

The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

|  |  |
| --- | --- |
| **Syntax:**  SELECT column\_name(s)  FROM table1  LEFT JOIN table2  ON table1.column\_name = table2.column\_name; | SELECT Customers.CustomerName, Orders.OrderID  FROM Customers  LEFT JOIN Orders ON  Customers.CustomerID = Orders.CustomerID  ORDER BY Customers.CustomerName; |

**5.3 RIGHT JOIN**

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

|  |  |
| --- | --- |
| **Syntax:**  SELECT column\_name(s)  FROM table1  RIGHT JOIN table2  ON table1.column\_name = table2.column\_name; | SELECT Customers.CustomerName, Orders.OrderID  FROM Customers  RIGHT JOIN Orders ON  Customers.CustomerID = Orders.CustomerID  ORDER BY Customers.CustomerName; |

**5.4 FULL OUTER JOIN**

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

|  |  |
| --- | --- |
| **Syntax:**  SELECT column\_name(s)  FROM table1  FULL OUTER JOIN table2  ON table1.column\_name = table2.column\_name  WHERE condition; | SELECT Customers.CustomerName, Orders.OrderID  FROM Customers FULL OUTER JOIN Orders  ON Customers.CustomerID=Orders.CustomerID  ORDER BY Customers.CustomerName; |

**5.5 SQL Self Join**

A self join is a regular join, but the table is joined with itself.

|  |  |
| --- | --- |
| **Syntax:**  SELECT column\_name(s)  FROM table1 T1, table1 T2  WHERE condition; | SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City  FROM Customers A, Customers B  WHERE A.CustomerID <> B.CustomerID  AND A.City = B.City  ORDER BY A.City; |

**6. Subquery:**

In SQL a Subquery can be simply defined as a query within another query. In other words we can say that a Subquery is a query that is embedded in WHERE clause of another SQL query. Important rules for Subqueries:

* You can place the Subquery in a number of SQL clauses: WHERE clause, HAVING clause, FROM clause. Subqueries can be used with SELECT, UPDATE, INSERT, DELETE statements along with expression operator.
* It could be equality operator or comparison operator such as =, >, =, <= and Like operator.
* A subquery is a query within another query. The outer query is called as main query and inner query is called as subquery.
* The subquery generally executes first when the subquery doesn’t have any co-relation with the main query, when there is a co-relation the parser takes the decision on the fly on which query to execute on precedence and uses the output of the subquery accordingly.
* Subquery must be enclosed in parentheses. Subqueries are on the right side of the comparison operator. ORDER BY command cannot be used in a Subquery. GROUPBY command can be used to perform same function as ORDER BY command.
* Use single-row operators with singlerow Subqueries. Use multiple-row operators with multiple-row Subqueries.

|  |
| --- |
| **Syntax:**  SELECT column\_name  FROM table\_name  WHERE column\_name expression operator (SELECT COLUMN\_NAME from TABLE\_NAME WHERE ... ); |

|  |
| --- |
| **Example:**  Select NAME, LOCATION, PHONE\_NUMBER from DATABASE  WHERE ROLL\_NO IN (SELECT ROLL\_NO from STUDENT where SECTION=’A’); |

**7. View**

In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. You can add SQL statements and functions to a view and present the data as if the data were coming from one single table. A view is created with the CREATE VIEW statement.

|  |  |
| --- | --- |
| CREATE VIEW view\_name AS  SELECT column1, column2, ...  FROM table\_name  WHERE condition; | CREATE VIEW [Brazil Customers] AS  SELECT CustomerName, ContactName  FROM Customers  WHERE Country = 'Brazil'; |

**Conclusion:**

We have successfully implemented SQL DML statements such as Insert, Select, Update, Delete with operators, functions, set operator, joins, sub-queries, and views.