ASSIGNMENT NO.03

TITLE: SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator

PROBLEM STATEMENT: To Design SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator.

OBJECTIVES:

- To list and understand DML commands and SQL Set Operations and functions
- Use database techniques such as SQL DML statements

OUTCOMES:

- Students will be able to write queries for given requirements, using SQL DML Commands
- Students will be able to write queries for given requirements, using SQL Set Operations and functions

SOFTWARE & HARDWARE REQUIREMENTS:

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

THEORY:

Data Manipulation Language (DML) statements are used for managing data in database. DML commands are not auto-committed. It means changes made by DML command are not permanent to database, it can be rolled back.

INSERT COMMAND

Insert command is used to insert data into a table. Following is its general syntax,

```
INSERT into table-name values(data1,data2,..)
```

Consider a table Student with following fields :S_id, S_Name, age
mysql>INSERT into Student values(101,'Adam',15);
The above command will insert a record into Student table.

Example to Insert NULL value to a column

Both the statements below will insert NULL value into age column of the Student table.
INSERT into Student(id,name) values(102,'Alex');
Or,
INSERT into Student values(102,'Alex',null);

The above command will insert only two column value other column is set to null.

UPDATE COMMAND

Update command is used to update a row of a table. Following is its general syntax,

```
UPDATE table-name set column-name = value where condition;
```

Lets see an example,

mysql> update Student set age=18 where s_id=102;

Example to Update multiple columns

mysql> UPDATE Student set s_name='Abhi',age=17 where s_id=103;

DELETE COMMAND

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,

```
DELETE from table-name;
```

Example to Delete all Records from a Table

mysql>DELETE from Student;

The above command will delete all the records from Student table.

Example to Delete a particular Record from a Table

mysql> DELETE from Student where s_id=103;

The above command will delete the record where s_id is 103 from Student table.

SELECT COMMAND

Select query is used to retrieve data from a tables. It is the most used SQL query. We can retrieve complete tables, or partial by mentioning conditions using WHERE clause.

Syntax of SELECT Query

```
SELECT column-name1, column-name2, column-name3, column-nameN from table-name;
```

Example for SELECT Query

mysql> SELECT s_id, s_name, age from Student.

The above query will fetch information of s_id, s_name and age column from Student table

A special character asterisk * is used to address all the data(belonging to all columns) in a query. SELECT statement uses * character to retrieve all records from a table.

mysql> SELECT * from student;

The above query will show all the records of Student table, that means it will show complete Student table as result.

The SELECT statement has many optional clauses:

- **WHERE** specifies which rows to retrieve.

- **GROUP BY** groups rows sharing a property so that an aggregate function can be applied to each group.
- **HAVING** selects among the groups defined by the GROUP BY clause**.**
- **ORDER BY** specifies an order in which to return the rows.
- **AS** provides an alias which can be used to temporarily rename tables or columns.

The WHERE clause can be combined with AND, OR, and NOT operators.
The AND and OR operators are used to filter records based on more than one condition:
The AND operator displays a record if all the conditions separated by AND is TRUE. The OR operator displays a record if any of the conditions separated by OR is TRUE.
The NOT operator displays a record if the condition(s) is NOT TRUE.

AND Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

OR Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

NOT Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

Operator	**Description**	**Example**
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A = B) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.

<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Examples of AND, OR, NOT Operators

Consider following Customers table

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

AND Example

The following SQL statement selects all fields from "Customers" where country is "Germany" AND city is "Berlin":

```
mysql> SELECT * FROM Customers
WHERE Country='Germany' AND City='Berlin';
```

OR Example

The following SQL statement selects all fields from "Customers" where city is "Berlin" OR "München":

```
mysql>SELECT * FROM Customers WHERE City='Berlin' OR City='München';
```

NOT Example

The following SQL statement selects all fields from "Customers" where country is NOT "Germany":

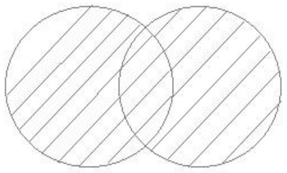
```
mysql> SELECT * FROM Customers WHERE NOT Country='Germany';
```

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.

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.



Example of UNION

Consider following tables

The **First** table,

ID	Name
1	abhi
2	adam

The **Second** table,

ID	Name
2	adam
3	Chester

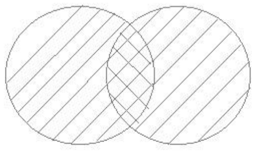
Union SQL query will be,

SELECT * from First **UNION** select * from second

The result table will look like,

ID	NAME
1	abhi
2	adam
3	Chester

Union All: This operation is similar to Union. But it also shows the duplicate rows.



Union All query will be like,

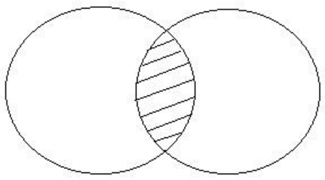
select * from First **UNION ALL** select * from second

The result table will look like,

ID	NAME
1	abhi
2	adam
2	adam
3	Chester

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.



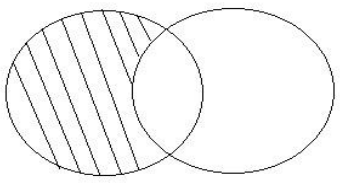
Intersect query will be,

select * from First **INTERSECT** select * from second;

The result table will look like

ID	NAME
2	adam

Minus
Minus operation combines result of two Select statements and return only those result which belongs to first set of result.



Minus query will be,

SELECT * from First **MINUS** select * from second

The result table will look like,

ID	NAME
1	abhi

LIKE CLAUSE

Like clause is used as condition in SQL query. Like clause compares data with an expression using wildcard operators. It is used to find similar data from the table.

Wildcard operators

There are two wildcard operators that are used in like clause.

- Percent sign % : represents zero, one or more than one character.
- Underscore sign _ : represents only one character.

Example of LIKE clause

Consider the following Student table.

s_id	s_Name	age
101	Adam	15
102	Alex	18
103	Abhi	17

mysql>SELECT * from Student where s_name like 'A%';

The above query will return all records where s_name starts with character 'A'.

s_id	s_Name	age
101	Adam	15
102	Alex	18
103	Abhi	17

mysql> SELECT * from Student where s_name like '_d%';

The above query will return all records from Student table where s_name contain 'd' as second character.

s_id	s_Name	age
101	Adam	15

ORDER BY CLAUSE

Order by clause is used with Select statement for arranging retrieved data in sorted order. The Order by clause by default sort data in ascending order. To sort data in descending order DESC keyword is used with Order by clause.

Syntax of Order By

SELECT ***column-list***|* from ***table-name*** order by asc|desc;

Example using Order by

Consider the following Emp table,

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

mysql> SELECT * from Emp order by salary;

The above query will return result in ascending order of the salary.

eid	name	age	salary
403	Rohan	34	6000
402	Shane	29	8000
405	Tiger	35	8000
401	Anu	22	9000
404	Scott	44	10000

Example of Order by DESC

Consider the Emp table described above,

mysql> SELECT * from Emp order by salary DESC;

The above query will return result in descending order of the salary.

eid	name	age	salary
404	Scott	44	10000
401	Anu	22	9000
405	Tiger	35	8000
402	Shane	29	8000
403	Rohan	34	6000

GROUP BY CLAUSE

Group by clause is used to group the results of a SELECT query based on one or more columns. It is also used with SQL functions to group the result from one or more tables.

Syntax for using Group by in a statement.

```
SELECT column_name, function(column_name)
FROM table_name
WHERE condition
GROUP BY column_name
```

Example of Group by in a Statement

Consider the Emp table mentioned in previous example.

Here we want to find name and age of employees grouped by their salaries

SQL query for the above requirement will be, mysql>SELECT name, age from Emp group by salary

Result will be,

name	age
Rohan	34
shane	29
anu	22

Example of Group by in a Statement with WHERE clause

SQL query will be,

mysql> select name, salary from Emp where age > 25 group by salary;

Result will be.

name	salary
Rohan	6000
Shane	8000
Scott	9000

HAVING CLAUSE

Having clause is used with SQL Queries to give more precise condition for a statement. It is used to mention condition in Group based SQL functions, just like WHERE clause.

Syntax for having will be,

```
select column_name, function(column_name)
FROM table_name
WHERE column_name condition
GROUP BY column_name
HAVING function(column_name) condition
```

Example of HAVING Statement

Consider the following Sale table.

oid	order_name	previous_balance	customer
11	ord1	2000	Alex
12	ord2	1000	Adam
13	ord3	2000	Abhi
14	ord4	1000	Adam
15	ord5	2000	Alex

Suppose we want to find the customer whose previous_balance sum is more than 3000.

We will use the below SQL query,

SELECT * from sale group customer having sum(previous_balance) > 3000

Result will be,

oid	order_name	previous_balance	customer
11	ord1	2000	Alex

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. SQL functions are divided into two catagories,

- **Aggregate Functions**
- **Scalar Functions**

AGGREGATE FUNCTIONS

These functions return a single value after calculating from a group of values.Following are some frequently used Aggregrate functions.

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

Its general Syntax is,

```
SELECT AVG(column_name) from table_name;
```

SQL query to find average of salary from Emp table will be,

mysql> SELECT avg(salary) from Emp;

COUNT(): Count returns the number of rows present in the table either based on some condition or without condition.

Its general Syntax is,

```
SELECT COUNT(column_name) from table-name;
```

SQL query to count employees, satisfying specified condition is,

mysql> SELECT COUNT(name) from Emp where salary = 8000;

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

Syntax of MAX function is,

```
SELECT MAX(column_name) from table-name ;
```

SQL query to find Maximum salary is,

mysql> SELECT MAX(salary) from emp;

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

Syntax for MIN function is,

```
SELECT MIN(column_name) from table-name
```

SQL query to find minimum salary is,

mysql>SELECT MIN(salary) from emp;

SUM(): SUM function returns total sum of a selected columns numeric values.

Syntax for SUM is,

```
SELECT SUM(column_name) from table-name
```

SQL query to find sum of salaries will be,

mysql> SELECT SUM(salary) from emp;

SCALAR FUNCTIONS: Scalar functions return a single value from an input value. Following are soe frequently used Scalar Functions.

UCASE(): UCASE function is used to convert value of string column to Uppercase character. Syntax of UCASE,

```
SELECT UCASE(column_name) from table-name
```

LCASE(): LCASE function is used to convert value of string column to Lowecase character. Syntax for LCASE is,

```
SELECT LCASE(column_name) from table-name
```

MID(): MID function is used to extract substrings from column values of string type in a table. Syntax for MID function is,

```
SELECT MID(column_name, start, length) from table-name
```

ROUND(): ROUND function is used to round a numeric field to number of nearest integer. It is used on Decimal point values. Syntax of Round function is,

```
SELECT ROUND(column_name, decimals) from table-name
```

CONCLUSION:

QUESTIONS:

Create the Employee table using following schema

Employee (Employee_id, First_name, Last_name, Salary, Joining_date, Department,)

1. Insert 10 to 15 appropriate records in the Employee table.
2. Get First_Name,Last_Name from employee table
3. Get unique DEPARTMENT from employee table
4. Get FIRST_NAME ,Joining year,Joining Month and Joining Date from employee table

Select FIRST_NAME, year(joining_date),month(joining_date), DAY(joining_date) from EMPLOYEE

5. Get all employee details from the employee table order by Salary Ascending
6. Get all employee details from the employee table whose First_Name starts with A.
7. Update the Salary column by incrementing salary of all employees having salary less than 20000 by 5000.
8. Delete the department of employee no 004.
9. Find department wise minimum salary.
10. Find department wise Average salary in ascending order.

Consider Following Schema

Employee(employee_id, employee_name, City, Company_Name, Salary)

11. Find details of all employees who work for company “IBM” and live in city “Pune”.
12. Find names, and cities of all employees who work for “Infosys” or earn more than 30000.
13. Find all employees who are employees of “IBM” and not living in city “Mumbai”.
14. Find company wise maximum salary.
15. Find those companies whose employees earn higher salary, than average salary at “IBM”.