ASSIGNMENT NO. 7

Aim

Perform following SQL queries on the database created in assignment 1.

• Implementation of relational operators in SQL

• Boolean operators and pattern matching

• Arithmetic operations and built in functions

• Group functions

• Processing Date and Time functions

• Complex queries and set operators

Objective

Understand and implement DML statements.

Theory

Data Manipulation Language (DML) allows you to modify the database instance by inserting, modifying, and deleting its data. It is responsible for performing all types of data modification in a database.

There are three basic constructs which allow database program and user to enter data and information are:

Here are some important DML commands in SQL:

INSERT

UPDATE

DELETE

General syntax of **select** query:

```
select A1, A2, ..., An
from r1, r2, ..., rm
where P
```

Insert

INSERT INTO table_name

(col1, col2,…..)

values

(col1_value, col2_value, ……)

Update

update table_name

set

column_name::expression

….

where condition

Delete

delete from table_name

where condition;

MySQL Inbuilt Functions

The MID() function extracts a substring from a string (starting at any position).

MID(*string*, *start*, *length*)

The LENGTH() function returns the length of a string (in bytes).

LENGTH(*string*)

The STRCMP() function compares two strings.

STRCMP(*string1*, *string2*)

The SUBSTR() function extracts a substring from a string (starting at any position).

SUBSTR(*string*, *start*, *length*)

The LCASE() function converts a string to lower-case.

LCASE(*text*)

The ABS() function returns the absolute (positive) value of a number.

ABS(*number*)

The DATE() function extracts the date part from a datetime expression.

DATE(*expression*)

The NOW() function returns the current date and time.

NOW()

The TIME() function extracts the time part from a given time/datetime.

TIME(*expression*)

The BIN() function returns a binary representation of a number, as a string value.

BIN(*number*)

Aggregate Functions

The data that you need is not always stored in the tables. However, you can get it by performing the calculations of the stored data when you select it.

For example, you cannot get the total amount of each order by simply querying from the order details table because the order details table stores only quantity and price of each item. You have to select the quantity and price of an item for each order and calculate the order's total.

To perform such calculations in a query, you use aggregate functions.

By definition, an aggregate function performs a calculation on a set of values and returns a single value.

MySQL provides many aggregate functions that include AVG, COUNT, SUM, MIN, MAX, etc. An aggregate function ignores NULL values when it performs calculation except for the COUNT function.

AVG function

The AVG function calculates the average value of a set of values. It ignores NULL values in the calculation.

select avg(Salary) from Employee;

SUM function

The SUM function returns the sum of a set of values. The SUM function ignores NULL values. If no matching row found, the SUM function returns a NULL value.

MAX function

The MAX function returns the maximum value in a set of values.

MIN function

The MIN function returns the minimum value in a set of values.

Nested Queries

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, 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 command cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY command 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.

Multinational Company Manages information of Employees.Each Employee has unique ID.

Each Employee works on project in some department.

Design ER Diagram for Employee Management System

Create tables for above.

Employee(EmpID, EName, Address, Contactno, DOB, DOJ, Salary, DID, PId)

Department(DID, DName)

Project(PId, PName, LocationId)

Location(LocationId, LocationCity)

Example Queries:

1 select * from employee;

2 select * from employee where pid is null;

3 select * from employee where pid is not null;

4 select * from employee where salary>=20000;

5 select * from employee where salary>20000 and pid=102;

6 select * from employee where did<>11;

7 select * from employee where doj>='2012-01-01' and doj<='2012-12-31';

8 select * from employee where doj>='2012-01-01' and doj<='2012-12-31' and did=12;

9 select * from employee where doj between '2012-01-01' and '2012-12-31';

Boolean Operators and Pattern Matching

1. select * from Employee where Address like 'PUNE';

2. select * from Employee where EmpName like 'A%';

3. select * from Employee where Address like '%bad';

4. select * from Employee where EmpName like '_a%';

5. select * from Employee where EmpName like '____'; //four underscores

6. select * from Employee where Address like '%/_%';

7. select * from Employee where Address like '%bad' and EmpName like '____';

8. select * from Employee where Address not like 'PUNE';

Inbuilt Functions

select length(EName) from Employee;

select ucase(EName) from Employee;

select lcase(EName) from Employee;

select abs(salary) from Employee;

Aggregate function : sum, min, max, avg, count

1. select sum(Salary) from Employee;

2. select count(EmpID) from Employee;

3. select count(EmpID) from Employee where DId='D101';

4. select count(EmpID) from Employee where DOJ between '2016-01-01' and '2016-12-31';

5. select sum(Salary) from Employee where DId='D101';

6. select sum(Salary) from Employee where DId in ('D101', 'D102', 'D103');

7. select DId, sum(salary) from Employee group by DId;

8. select DId, sum(salary) as totalsal from Employee group by DId having totalsal>150000;

9. select DId, sum(salary) as totalsal from Employee where DOJ <= now() group by DId;

10. select DId, sum(salary) as totalsal from Employee where DOJ between '1992-01-01' and date() group by DId;

11. select DName, sum(Salary) as totalsal from Employee, Department where Emplyoee.DID=Department.DID group by DName;

Nested Queries

1. select * from employee where Salary **in** (select max(Salary) from employee);

2. select * from employee where Salary **not in** (select max(Salary) from employee);

3. select Salary from employee where Salary **<> any** (select e.Salary from employee as e where e.did=11);

4. select Salary from employee where Salary **> any** (select e.Salary from employee as e where e.did=11);

5. select Salary from employee where Salary **> some** (select e.Salary from employee as e where e.did=11);

6. select Salary from employee where Salary = **some** (select e.Salary from employee as e where e.did=11);

7. select * from employee where Salary >= **all (**select Salary from employee);

8. select Salary from employee where Salary > **all** (select e.Salary from employee as e where e.did=11);

9. select Salary from employee where **exists** (select e.Salary from employee as e where e.did=employee.did and did=11);

10. select Salary from employee where **not exists** (select e.Salary from employee as e where e.did=11 and e.did=employee.did);

Join

Natural Join

select * from Employee natural join Department;

Inner Join

select * from Employee inner join Department on Employee.DID=Department.DID;

Outer Join

```
select * from Employee left join Department on Employee.DID=Department.DID;

select * from Employee right join Department on Employee.DID=Department.DID;
```