Group A

Assignment No: 1B

Aim:

Design at least 10 SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator.

Objective: To learn and understand DML statements in MySQL.

Hardware requirements:

Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software requirements:

Ubuntu 14 Operating System, MySQL

Theory:

DML command

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.

1) INSERT command

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

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

Lets see an example,

Consider a table **Student** with following fields.

S_id	S_Name	age

INSERT into Student values(101,'Adam',15);

The above command will insert a record into **Student** table.

S_id	S_Name	age
101	Adam	15

2) 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,

update Student set age=18 where s_id=102;

Example to Update multiple columns

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

3) 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

DELETE from Student;

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

Example to Delete a particular Record from a Table

Consider **Student** table

DELETE from Student where s_id=103;

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,

- Aggregrate Functions
- Scalar Functions

Aggregrate Functions

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

1) AVG()

Average returns average value after calculating from values in a numeric column.

Its general Syntax is,

*SELECT **AVG**(column_name) from table_name*

e.g.

*SELECT **avg(salary)** from Emp;*

2) 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;*

Example using COUNT()

Consider following **Emp** table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000

SQL query to count employees, satisfying specified condition is,

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

3) FIRST()

First function returns first value of a selected column

Syntax for FIRST function is,

*SELECT **FIRST**(column_name) from table-name*

SQL query

SELECT FIRST(salary) from Emp;

4) LAST()

LAST return the return last value from selected column

Syntax of LAST function is,

SELECT **LAST**(column_name) from *table-name*

SQL query will be,

SELECT LAST(salary) from emp;

5) 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,

SELECT MAX(salary) from emp;

6) 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,

SELECT MIN(salary) from emp;

7) 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,

SELECT SUM(salary) from emp;

Scalar Functions

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

1) UCASE()

UCASE function is used to convert value of string column to Uppercase character.

Syntax of UCASE,

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

Example of UCASE()

SQL query for using UCASE is,

SELECT UCASE(name) from emp;

2) 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*

3) 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*

4) 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*

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.

1) AND operator

AND operator is used to set multiple conditions with *Where* clause.

Example of AND

SELECT * from Emp WHERE salary < 10000 **AND** age > 25

2) OR operator

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.

Example of OR

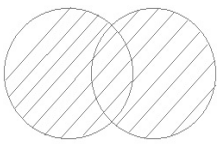
SELECT * from Emp WHERE salary > 10000 **OR** age > 25

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.

3) 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

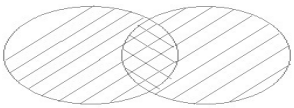
select * from First

UNION

select * from second

4) Union All

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



Union All query will be like,

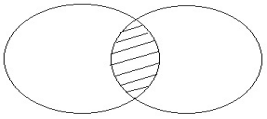
select * from First

UNION ALL

select * from second

5) 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,

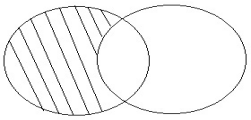
select * from First

INTERSECT

select * from second

6) 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.



Minus query will be, select * from First MINUS
select * from second

TITLE: Design at least 10 SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator.

---

mysql> show databases;

```
+--------------------+
| Database
|
+--------------------+
| information_schema |
| A |
| Abhi |
| PVG |
| RENUKA |
| mysql |
| nishant |
| performance_schema |
| renuka |
| sys |
| time |
+--------------------+
11 rows in set (0.11 sec)
```

mysql> use Abhi;

Database changed

mysql> create table Employee(emp_no int,emp_name varchar(20),date date,position varchar(20));

Query OK, 0 rows affected (0.75 sec)

mysql> alter table Employee add salary int;

Query OK, 0 rows affected (0.68 sec)
Records: 0
Duplicates: 0
Warnings: 0

mysql> insert into Employee values('01','abc','2018-07-11','clerk','50000');

Query OK, 1 row affected (0.08 sec)

mysql> insert into Employee values('02','abhi','2018-05-11','ceo','150000');

Query OK, 1 row affected (0.08 sec)

mysql> insert into Employee values('03','xyz','2018-05-21','hr','100000');

Query OK, 1 row affected (0.04 sec)

mysql> insert into Employee values('04','aqwgy','2018-06-21','te','10000');

Query OK, 1 row affected (0.03 sec)

mysql> insert into Employee values('05','sfhjfh','2018-07-21','gt','12000');

Query OK, 1 row affected (0.03 sec)

mysql> create table TE(emp_no int,emp_namevarchar(20),join_date date,position varchar(20),salary int);

Query OK, 0 rows affected (0.36 sec)

mysql> insert into TE values('01','abc','2018-07-11','clerk','50000');

Query OK, 1 row affected (0.03 sec)

mysql> insert into TE values('02','abhi','2018-05-11','ceo','150000');

Query OK, 1 row affected (0.04 sec)

mysql> insert into TE values('03','xyz','2018-05-21','hr','100000');

Query OK, 1 row affected (0.04 sec)

mysql> insert into TE values('04','aqwgy','2018-06-21','te','10000');

Query OK, 1 row affected (0.05 sec)

mysql> insert into TE values('05','sfhjfh','2018-07-21','gt','12000');

Query OK, 1 row affected (0.04 sec)

mysql> select * from TE;

```
+--------+----------+------------+----------+---------+
| emp_no | emp_name | join_date
| position | salary |
+--------+----------+------------+----------+---------+
| 1 | abc | 2018-07-11 | clerk |
50000 |
| 2 | abhi | 2018-05-11 | ceo | 150000 |
| 3 | xyz | 2018-05-21 | hr | 100000 |
| 4 | aqwgy | 2018-06-21 | te |
```

10000 |
| 5 | sfhjfh | 2018-07-21 | gt |
12000 |
+--------+----------+------------+----------+---------+
5 rows in set (0.04 sec)

mysql> select * from Employee;

+--------+----------+------------+----------+---------+
| emp_no | emp_name | date
| position | salary |
+--------+----------+------------+----------+---------+
| 1 | abc | 2018-07-11 | clerk |
50000 |
2	abhi	2018-05-11	ceo	150000
3	xyz	2018-05-21	hr	100000
4	aqwgy	2018-06-21	te
10000
5	sfhjfh	2018-07-21	gt
12000 |
+--------+----------+------------+----------+---------+
5 rows in set (0.00 sec)

mysql> update TE set emp_name='gjgj' where emp_no='5';

Query OK, 1 row affected (0.13 sec)
Rows matched: 1
Changed: 1
Warnings: 0

mysql> select * from TE;
+--------+----------+------------+----------+---------+
| emp_no | emp_name | join_date
| position | salary |
+--------+----------+------------+----------+---------+
| 1 | abc | 2018-07-11 | clerk |
50000 |
2	abhi	2018-05-11	ceo	150000
3	xyz	2018-05-21	hr	100000
4	aqwgy	2018-06-21	te
10000
5	gjgj	2018-07-21	gt
12000 |
+--------+----------+------------+----------+---------+
5 rows in set (0.00 sec)

mysql> select * from Employee union select * from TE;
+--------+----------+------------+----------+ --------+
| emp_no | emp_name | date
| position | salary |
+--------+----------+------------+----------+---------+
| 1 | abc | 2018-07-11 | clerk |

50000 |
2	abhi	2018-05-11	ceo	150000
3	xyz	2018-05-21	hr	100000
4	aqwgy	2018-06-21	te
10000
5	sfhjfh	2018-07-21	gt
12000
5	gjgj	2018-07-21	gt
12000 |
+--------+----------+------------+----------+---------+
6 rows in set (0.01 sec)

mysql> select * from Employee union all select * from TE;

+--------+----------+------------+----------+---------+| emp_no | emp_name | date
| position | salary |
+--------+----------+------------+----------+---------+
| 1 | abc | 2018-07-11 | clerk |
50000 |
2	abhi	2018-05-11	ceo	150000
3	xyz	2018-05-21	hr	100000
4	aqwgy	2018-06-21	te
10000
5	sfhjfh	2018-07-21	gt
12000
1	abc	2018-07-11	clerk
50000
2	abhi	2018-05-11	ceo	150000
3	xyz	2018-05-21	hr	100000
4	aqwgy	2018-06-21	te
10000
5	gjgj	2018-07-21	gt
12000 |
+--------+----------+------------+----------+---------+
10 rows in set (0.00 sec)

mysql> select distinct emp_no from Employee where emp_no in(select emp_no from TE);
+--------+
| emp_no |
+--------+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
+--------+
5 rows in set (0.03 sec)

mysql> select * from Employee;
+--------+----------+------------+----------+---------+
| emp_no | emp_name | date
| position | salary |

+--------+----------+------------+----------+---------+| 1 | abc | 2018-07-11 | clerk |
50000 |
2	abhi	2018-05-11	ceo	150000
3	xyz	2018-05-21	hr	100000
4	aqwgy	2018-06-21	te
10000
5	sfhjfh	2018-07-21	gt
12000 |
+--------+----------+------------+----------+---------+
5 rows in set (0.00 sec)

mysql> select * from TE;

+--------+----------+------------+----------+---------+
| emp_no | emp_name | join_date
| position | salary |
+--------+----------+------------+----------+---------+
| 1 | abc | 2018-07-11 | clerk |
50000 |
2	abhi	2018-05-11	ceo	150000
3	xyz	2018-05-21	hr	100000
4	aqwgy	2018-06-21	te
10000
5	gjgj	2018-07-21	gt
12000 |
+--------+----------+------------+----------+---------+
5 rows in set (0.00 sec)

mysql> select distinct emp_name from Employee where emp_name in(select emp_name from TE);
+----------+
| emp_name |
+----------+
| abc |
| abhi |
| xyz |
| aqwgy |
+----------+
4 rows in set (0.00 sec)

mysql> select * from Employee;

+--------+----------+------------+----------+ --------+
| emp_no | emp_name | date
| position | salary |
+--------+----------+------------+----------+---------+
| 1 | abc | 2018-07-11 | clerk |
50000 |
2	abhi	2018-05-11	ceo	150000
3	xyz	2018-05-21	hr	100000
4	aqwgy	2018-06-21	te
10000 |

| 5 | sfhjfh | 2018-07-21 | gt |
12000 |
+--------+----------+-----------+----------+---------+
5 rows in set (0.00 sec)

mysql> select * from TE;

+--------+----------+-----------+----------+---------+
| emp_no | emp_name | join_date
| position | salary |
+--------+----------+-----------+----------+---------+
| 1 | abc | 2018-07-11 | clerk |
50000 |
2	abhi	2018-05-11	ceo	150000
3	xyz	2018-05-21	hr	100000
4	aqwgy	2018-06-21	te
10000
5	gjgj	2018-07-21	gt
12000 |
+--------+----------+-----------+----------+---------+
5 rows in set (0.00 sec)

mysql> select distinct emp_name from Employee where emp_name in(select emp_name from TE);

+----------+
| emp_name |
+----------+
| abc |
| abhi |
| xyz || aqwgy
|
+----------+
4 rows in set (0.00 sec)

mysql> select min(salary) from Employee;
+-------------+
| min(salary) |
+-------------+
|
10000 |
+-------------+
1 row in set (0.04 sec)

mysql> select max(salary) from Employee;

+-------------+
| max(salary) |
+-------------+
|
150000 |
+-------------+

1 row in set (0.00 sec)

mysql> select sum(salary) from Employee;

```
+-------------+
| sum(salary) |
+-------------+
|
322000 |
+-------------+
1 row in set (0.00 sec)
```

mysql> select avg(salary) from Employee;

```
+-------------+
| avg(salary) |
+-------------+|
64400.0000 |
+-------------+
1 row in set (0.00 sec)
mysql> select count(salary) from Employee;
+---------------+
| count(salary) |
+---------------+
|
5 |
+---------------+
1 row in set (0.00 sec)
```

mysql> select lcase(emp_no) from Employee;

```
+---------------+
| lcase(emp_no) |
+---------------+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
+---------------+
5 rows in set (0.00 sec)
```

mysql> select ucase(emp_no) from Employee;

```
+---------------+
| ucase(emp_no) |
+---------------+
| 1 |
| 2 |
| 3 |
| 4 || 5
|
```

```
+---------------+
5 rows in set (0.00 sec)
```

mysql> select lcase(salary) from Employee;

```
+---------------+
| lcase(salary) |
+---------------+
| 50000 |
| 150000 |
| 100000 |
| 10000 |
| 12000 |
+---------------+
5 rows in set (0.00 sec)
```

mysql> select mid(emp_no,1,3) from Employee;

```
+-----------------+
| mid(emp_no,1,3) |
+-----------------+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
+-----------------+
5 rows in set (0.01 sec)
```

mysql> select mid(emp_no,1,3) from Employee;

```
+-----------------+
| mid(emp_no,1,3) |
+-----------------+| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
+-----------------+
5 rows in set (0.00 sec)
```

mysql> select mid(emp_no,1,5) from Employee;

```
+-----------------+
| mid(emp_no,1,5) |
+-----------------+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
+-----------------+
5 rows in set (0.00 sec)
```

mysql> select mid(salary,1,3) from Employee;

```
+-----------------+
| mid(salary,1,3) |
+-----------------+
| 500 |
| 150 |
| 100 |
| 100 |
| 120 |
+-----------------+
5 rows in set (0.00 sec)
```

mysql> select mid(salary,1,5) from Employee;

```
+-----------------+
| mid(salary,1,5) |
+-----------------+
| 50000 |
| 15000 |
| 10000 |
| 10000 |
| 12000 |
+-----------------+
5 rows in set (0.00 sec)
```

mysql> select mid(emp_no,1,2) from Employee;

```
+-----------------+
| mid(emp_no,1,2) |
+-----------------+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
+-----------------+
5 rows in set (0.00 sec)
```