**Table of Contents**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **Date** | **Name of the Experiment** | **Pg.No** | **Sign** |
| 1 |  | CREATION OF TABLES (DDL ,DML COMMANDS) |  |  |
| 2 |  | CREATION OF TABLES WITH CONSTRAINTS |  |  |
| 3 |  | WHERE CLAUSE CONDITIONS AND AGGREGATE FUNCTIONS |  |  |
| 4 |  | SIMPLE JOIN OPERATIONS |  |  |
| 5 |  | NATURAL, EQUI AND OUTER JOIN OPERATIONS |  |  |
| 6 |  | USER DEFINED FUNCTIONS AND STORED PROCEDURES |  |  |
| 7 |  | DCL AND TCL COMMANDS |  |  |
| 8 |  | TRIGGERS |  |  |
| 9 |  | VIEWS AND INDEX |  |  |
| 10 |  | XML DATABASE AND XML SCHEMA VALIDATION |  |  |
| 11 |  | DOCUMENT,COLUMN AND GRAPH BASED DATA USING NOSQL DATABASE |  |  |
| 12 |  | DEVELOP A SIMPLE GUI BASED DATABASE APPLICATION |  |  |

**EX: NO: 1 CREATION OF TABLES (DDL COMMANDS)**

**AIM:**

To execute and verify the Data Definition Language commands.

**SQL Command Categories**

SQL commands are grouped into four major categories depending on their functionality. They are as follows:

**Data Definition Language (DDL)**

These SQL commands are used for creating, modifying, and dropping the structure of database objects. The commands are CREATE, ALTER, DROP, RENAME, and TRUNCATE.

**Data Manipulation Language (DML)**

These SQL commands are used for storing, retrieving, modifying, and deleting data. These commands are SELECT, INSERT, UPDATE, and DELETE.

**Transaction Control Language (TCL)**

These SQL commands are used for managing changes affecting the data. These commands are COMMIT, ROLLBACK, and SAVEPOINT.

**Data Control Language (DCL)**

These SQL commands are used for providing security to database objects. These commands are GRANT and REVOKE.

**DDL** (DATA DEFINITION LANGUAGE)

* CREATE
* ALTER
* DROP
* TRUNCATE
* RENAME

**PROCEDURE**

STEP 1: Start

STEP 2: Create the table with its essential attributes.

STEP 3: Execute different Commands and extract information from the table.

STEP 4: Stop

**SQL COMMANDS**

1. COMMAND NAME: **CREATE**

COMMAND DESCRIPTION: **CREATE** command is used to create objects in the database.

**CREATE <OBJ.TYPE> <OBJ.NAME> (COLUMN NAME.1<DATATYPE> (SIZE), COLUMN NAME.1 <DATATYPE> (SIZE)……………………………);**

**Syntax For Create A from an Existing Table With All Fields**

**CREATE TABLE <TARGET TABLE NAME> AS SELECT \* FROM <SOURCE TABLE NAME>;**

2. COMMAND NAME: **DROP**

COMMAND DESCRIPTION: **DROP** command is used to delete the object from the database.

**Syntax for drop a new column:**

**ALTER TABLE <TABLE NAME> DROP COLUMN <COLUMN NAME>;**

**Syntax for drop a table:**

Drop table <tablename>;

3. COMMAND NAME: **TRUNCATE**

COMMAND DESCRIPTION: **TRUNCATE** command is used to remove all the records from the table

**Syntax truncating the tables.**

**Truncate table <tablename>;**

4. COMMAND NAME: **ALTER**

COMMAND DESCRIPTION: **ALTER** command is used to alter the structure of database.

**ALTER <TABLE NAME> MODIFY <COLUMN NAME> <DATATYPE>(SIZE);**

**Syntax for alter table with multiple column:**

**SQL > ALTER <TABLE NAME> MODIFY <COLUMN NAME1> <DATATYPE>**

**(SIZE), MODIFY <COLUMN NAME2> <DATATYPE> (SIZE)…….;**

**Syntax for add a new column:**

**SQL> ALTER TABLE <TABLE NAME> ADD (<COLUMN NAME1> <DATA**

**TYPE> <SIZE>,<COLUMN NAME2> <DATA TYPE> <SIZE>,…………………………);**

5. COMMAND NAME: **RENAME**

COMMAND DESCRIPTION: **RENAME** command is used to rename the objects.

**Syntax For Renaming A table**

**Rename table <oldname> To <newname>;**

**Syntax For Renaming A Column**

**ALTER** **TABLE** tablename **RENAME** **COLUMN** old column name **TO** new column name;

**Data base commands:**

**Create database:** create database <databasename>;

**Show database:** show databases;

**Use:** use <databasename>;

**Show table:** show tables;

**Description of a table:** desc <tablename>;

**QUERY: 01**

Q1. Write a query to create a table employee with empno, ename, designation, and

salary.

**QUERY: 01**

**SQL>CREATE TABLE EMP (EMPNO INT(4),ENAME VARCHAR(10),DESIGNATION VARCHAR(10),SALARY FLOAT(8,2));**

**Table created.**

**QUERY: 02**

Q2. Write a query to display the column name and datatype of the table employee.

**SQL> DESC EMP;**

**Name Null? Type**

**----------------------------------------- -------- -------------**

**EMPNO NUMBER(4)**

**ENAME VARCHAR2(10)**

**DESIGNATIN VARCHAR2(10)**

**SALARY NUMBER(8,2)**

**QUERY: 03**

Q3. Write a query for create a new table from an existing table with all the fields.

**QUERY: 03**

**SQL> CREATE TABLE EMP1 AS SELECT \* FROM EMP;**

**Table created.**

**SQL> DESC EMP1**

**Name Null? Type**

**----------------------------------------- -------- ------------------**

**EMPNO NUMBER(4)**

**ENAME VARCHAR(10)**

**DESIGNATIN VARCHAR(10)**

**SALARY NUMBER(8,2)**

**QUERY: 04**

Q4. Write a query to create a table from an existing table with selected fields.

**Syntax**

**SQL> CREATE TABLE <TARGET TABLE NAME> SELECT EMPNO, ENAME**

**FROM <SOURCE TABLE NAME>;**

**QUERY: 04**

**SQL> CREATE TABLE EMP2 AS SELECT EMPNO, ENAME FROM EMP;**

**Table created.**

**SQL> DESC EMP2;**

**Name Null? Type**

**----------------------------------------- -------- ----------------------**

**EMPNO NUMBER (4)**

**ENAME VARCHAR(10)**

**ALTER & MODIFICATION ON TABLE**

**QUERY: 06**

Q6. Write a Query to Alter the column EMPNO NUMBER (4) TO EMPNO NUMBER(6).

**QUERY: 06**

**SQL>ALTER TABLE EMP MODIFY EMPNO NUMBER (6);**

**Table altered.**

**SQL> DESC EMP;**

**Name Null? Type**

**----------------------------------------- -------- ----------------------------**

**EMPNO NUMBER(6)**

**ENAME VARCHAR(10)**

**DESIGNATIN VARCHAR(10)**

**SALARY NUMBER(8,2)**

**QUERY: 07**

Q7. Write a Query to Alter the table employee with multiple columns (EMPNO,

ENAME.)

**Syntax for alter table with multiple column:**

**SQL > ALTER <TABLE NAME> MODIFY <COLUMN NAME1> <DATATYPE>**

**(SIZE), MODIFY <COLUMN NAME2> <DATATYPE> (SIZE)…….;**

**QUERY: 07**

**SQL>ALTER TABLE EMP MODIFY EMPNO INT (7),MODIFY ENAME VARCHAR(12));**

**Table altered.**

**SQL> DESC EMP;**

**Name Null? Type**

**----------------------------------------- -------- ----------------------------**

**EMPNO NUMBER(7)**

**ENAME VARCHAR(12)**

**DESIGNATIN VARCHAR(10)**

**SALARY NUMBER(8,2);**

**QUERY: 08**

Q8. Write a query to add a new column in to employee

**QUERY: 08**

**SQL> ALTER TABLE EMP ADD QUALIFICATION VARCHAR2(6);**

**Table altered.**

**SQL> DESC EMP;**

**Name Null? Type**

**----------------------------------------- -------- ----------------------------**

**EMPNO NUMBER(7)**

**ENAME VARCHAR(12)**

**DESIGNATIN VARCHAR(10)**

**SALARY NUMBER(8,2)**

**QUALIFICATION VARCHAR2(6)**

**QUERY: 09**

Q9. Write a query to add multiple columns in to employee

**SQL>ALTER TABLE EMP ADD (DOB DATE, DOJ DATE);**

**Table altered.**

**SQL> DESC EMP;**

**Name Null? Type**

**----------------------------------------- -------- ----------------------------**

**EMPNO NUMBER(7)**

**ENAME VARCHAR(12)**

**DESIGNATIN VARCHAR2(10)**

**SALARY NUMBER(8,2)**

**QUALIFICATION VARCHAR(6)**

**DOB DATE**

**DOJ DATE**

**QUERY: 10**

Q10. Write the query to change the table name emp as employee

**SQL>** Rename table emp to employee;

**QUERY: 11**

Q11. Write the query to change the column name empno to eno of the table employee

**SQL>** **ALTER** **TABLE** employee **RENAME** **COLUMN** EMPNO **TO** ENO;

**SQL> DESC EMPLOYEE;**

**Name Null? Type**

**----------------------------------------- -------- ----------------------------**

**ENO NUMBER(7)**

**ENAME VARCHAR(12)**

**DESIGNATION VARCHAR(10)**

**SALARY NUMBER(8,2)**

**QUALIFICATION VARCHAR2(6)**

**DOB DATE**

**DOJ DATE**

**REMOVE / DROP**

**QUERY: 12**

Q12. Write a query to drop a column from an existing table employee

**SQL> ALTER TABLE EMPLOYEE DROP COLUMN DOJ;**

**SQL> DESC EMP;**

**Name Null? Type**

**----------------------------------------- -------- -------------**

**ENO NUMBER(7)**

**ENAME VARCHAR2(12)**

**DESIGNATIN VARCHAR2(10)**

**SALARY NUMBER(8,2)**

**QUALIFICATION VARCHAR2(6)**

**DOB DATE**

**QUERY: 13**

Q13. Write a query to truncate table employee

**SQL>** truncate table employee;

**QUERY: 14**

Q14. Write a query to drop table employee

**SQL> drop** table employee;

**DML COMMANDS**

**Data Manipulation Language (DML)**

These SQL commands are used for storing, retrieving, modifying, and deleting data. These commands are SELECT, INSERT, UPDATE, and DELETE.

**DML (DATA MANIPULATION LANGUAGE)**

* + - **SELECT**- It is used to retrieve information from the table. It is generally referred to as querying the table.
* **INSERT-** This is used to add one or more rows to a table. The values are separated by commas and the data types char and date are enclosed in apostrophes. The values must be entered in the same order as they are defined.
* **DELETE-** After inserting row in a table we can also delete them if required. The delete command consists of a from clause followed by an optional where clause.
* **UPDATE**- It is used to alter the column values in a table. A single column may be updated or more than one column could be updated.

**PROCEDURE:**

STEP 1: Start.

STEP 2: Create the table with its essential attributes.

STEP 3: Insert the record into table.

STEP 4: Update the existing records into the table.

STEP 5: Delete the records in to the table.

**SQL COMMANDS**

1. COMMAND NAME: **INSERT**

COMMAND DESCRIPTION: INSERT command is used to Insert objects in the database.

2. COMMAND NAME: **SELECT**

COMMAND DESCRIPTION: SELECT command is used to SELECT the object from the database.

3. COMMAND NAME: **UPDATE**

COMMAND DESCRIPTION: **UPDATE** command is used to UPDATE the records from the table

4. COMMAND NAME: **DELETE**

COMMAND DESCRIPTION: DELETE command is used to DELETE the Records form the table

**INSERT**

**QUERY: 01**

Q1. Write a query to insert the records in to employee.

**Syntax for Insert Records in to a table:**

**SQL > INSERT INTO <TABLE NAME> VALUES< VAL1, ‘VAL2’,…..);**

**A(**

**QUERY: 01**

INSERT A RECORD INTO AN EXISTING TABLE:

MYSQL>INSERT INTO EMP VALUES(101,'NAGARAJAN','LECTURER',15000);

MYSQL >INSERT INTO EMP VALUES(102 ,‘SARAVANAN’,’ LECTURER’,15000);

MYSQL >INSERT INTO EMP VALUES(103,’PANNERSELVAM’,’ ASST. PROF,20000);

MYSQL >INSERT INTO EMP VALUES(104,’CHINNI HOD’, ‘PROF’,45000);

1 row created.

**SELECT**

**QUERY: 02**

Q3. Write a query to display the records from employee.

**Syntax for select Records from the table:**

**SQL> SELECT \* FROM <TABLE NAME>**;

**QUERY: 02**

**DISPLAY THE EMP TABLE:**

SQL> SELECT \* FROM EMP;

EMPNO ENAME DESIGNATIN SALARY

---------- ------------ ---------- ----------

101 NAGARAJAN LECTURER 15000

102 SARAVANAN LECTURER 15000

103 PANNERSELVAM ASST. PROF 20000

104 CHINNI HOD, PROF 45000

**UPDATE**

**QUERY: 04**

Q1. Write a query to update the records from employee.

**Syntax for update Records from the table:**

SQL> UPDATE <<TABLE NAME> SET <COLUMNANE>=<VALUE> WHERE

<COLUMN NAME=<VALUE>;

**QUERY: 04**

SQL> UPDATE EMP SET SALARY=16000 WHERE EMPNO=101;

1 row updated.

SQL> SELECT \* FROM EMP;

EMPNO ENAME DESIGNATIN SALARY

---------- ------------ ---------- ----------

101 NAGARAJAN LECTURER 16000

102 SARAVANAN LECTURER 15000

103 PANNERSELVAM ASST. PROF 20000

104 CHINNI HOD, PROF 45000

**UPDATE MULTIPLE COLUMNS**

**QUERY: 05**

Q5. Write a query to update multiple records from employee.

**Syntax for update multiple Records from the table:**

SQL> UPDATE <<TABLE NAME> SET <COLUMNAME>=<VALUE> WHERE

<COLUMN NAME=<VALUE>;

**QUERY: 05**

SQL>UPDATE EMP SET SALARY = 16000, DESIGNATIN='ASST. PROF' WHERE

EMPNO=102;

1 row updated.

SQL> SELECT \* FROM EMP;

EMPNO ENAME DESIGNATIN SALARY

---------- ------------ ---------- ----------

101 NAGARAJAN LECTURER 16000

102 SARAVANAN ASST. PROF 15000

103 PANNERSELVAM ASST. PROF 20000

104 CHINNI HOD, PROF 45000

**DELETE**

**QUERY: 06**

Q5. Write a query to delete records from employee.

**Syntax for delete Records from the table:**

SQL> DELETE <TABLE NAME> WHERE <COLUMN NAME>=<VALUE>;

**QUERY: 06**

SQL> DELETE EMP WHERE EMPNO=103;

SQL> SELECT \* FROM EMP;

EMPNO ENAME DESIGNATIN SALARY

---------- ------------ ---------- ----------

101 NAGARAJAN LECTURER 16000

102 SARAVANAN ASST. PROF 15000

104 CHINNI HOD, PROF 45000

**Result:**

Thus the DDL, DML commands are executed in MySQL and verified successfully.

**EX: NO: 3 CREATION OF TABLES WITH CONSTRAINTS**

**AIM:**

To execute and verify the SQL commands for adding constraints.

## MySQL Constraints

* SQL constraints are used to specify rules for the data in a table.
* Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.
* Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

* [NOT NULL](https://www.w3schools.com/mysql/mysql_notnull.asp) - Ensures that a column cannot have a NULL value
* [UNIQUE](https://www.w3schools.com/mysql/mysql_unique.asp) - Ensures that all values in a column are different
* [PRIMARY KEY](https://www.w3schools.com/mysql/mysql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
* [FOREIGN KEY](https://www.w3schools.com/mysql/mysql_foreignkey.asp) - Prevents actions that would destroy links between tables
* [CHECK](https://www.w3schools.com/mysql/mysql_check.asp) - Ensures that the values in a column satisfies a specific condition
* [DEFAULT](https://www.w3schools.com/mysql/mysql_default.asp) - Sets a default value for a column if no value is specified

**PROCEDURE:**

STEP 1: Start.

STEP 2: Create the table with its essential attributes.

STEP 3: Add the constraint as a column level and table level

STEP 4: check all the constraints with specified conditions.

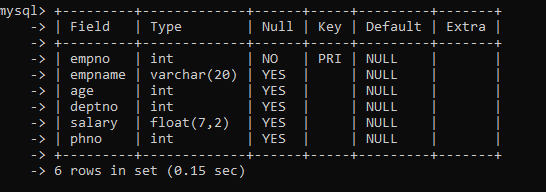
**Create table1:**

mysql>create table emp(empno int(3),empname varchar(20),age int(3),deptno int(3),salary float(7,2),phno int(5));

Query OK, 0 rows affected (0.11 sec)

Records: 0 Duplicates: 0 Warnings: 0

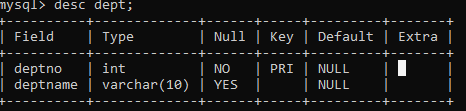
mysql>desc emp;



[**PRIMARY KEY**](https://www.w3schools.com/mysql/mysql_primarykey.asp)

**Q1: create the department table with the primary key as a table level constraint.**

mysql> create table dept(deptno int(3)primary key,deptname varchar(10));



**Q2: Alter the employee table with the primary key as a column level constraint.**

mysql> alter table emp modify empno int primary key;

Query OK, 0 rows affected (0.11 sec)

Records: 0 Duplicates: 0 Warnings: 0

[**NOT NULL**](https://www.w3schools.com/mysql/mysql_notnull.asp) **CONSTRAINT:**

**Q3:Add employee name as a not null constraint using alter command**

mysql> alter table emp modify empname varchar(10) not null;

Query OK, 0 rows affected (0.11 sec)

Records: 0 Duplicates: 0 Warnings: 0

**CHECK CONSTRAINT:**

Q4: **Add Check Constraint For The Column Age**

alter table emp modify age int check(age>=18);

Query OK, 0 rows affected (0.11 sec)

Records: 0 Duplicates: 0 Warnings: 0

**DEFAULT CONSTRAINT:**

Q5:Set salary column as a default constraint

mysql>alter table emp modify salary float default 1000;

Query OK, 0 rows affected (0.05 sec)

Records: 0 Duplicates: 0 Warnings: 0

[**UNIQUE**](https://www.w3schools.com/mysql/mysql_unique.asp) **CONSTRAINT**

Q6: create a unique constraint for the column phone number and check the constraint

alter table emp modify phno int unique;

Query OK, 0 rows affected (0.05 sec)

Records: 0 Duplicates: 0 Warnings: 0

**REFERENTIAL CONSTRAINT:FOREIGN KEY)**

Q7:

mysql> alter table emp add deptno int(3);

Query OK, 0 rows affected, 1 warning (0.05 sec)

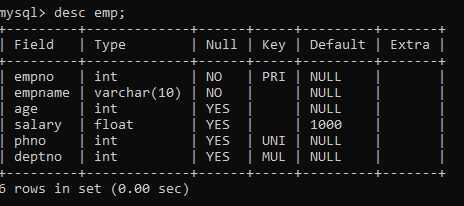
Records: 0 Duplicates: 0 Warnings: 1

mysql> alter table emp add foreign key(deptno) references dept(deptno);

Query OK, 0 rows affected (0.24 sec)

Records: 0 Duplicates: 0 Warnings: 0

Q8) desc emp;



**foreign key**

Q10:

insert into emp values(101,’angel’,20,20000,904486322,1);

Query OK, 1 row affected (0.03 sec)

insert into dept values(1,’PROJECT’);

Query OK, 1 row affected (0.03 sec)

insert into dept values(2,’DESIGN’);

Query OK, 1 row affected (0.03 sec)

insert into dept values(3,’HR’);

Query OK, 1 row affected (0.03 sec)

insert into dept values(4,’SOFTWARE’);

Query OK, 1 row affected (0.03 sec)

insert into emp values(101,’angel’,20,1,20000,904486322);

ERROR 1062 (23000): Duplicate entry '101' for key 'emp.PRIMARY'

**primary key**

insert into emp values(101,’angana’,20,1,30000,904486324);

ERROR 1062 (23000): Duplicate entry '101' for key 'emp.PRIMARY'

insert into emp values(102,’angana’,20,1,30000,904486324);

ERROR 1062 (23000): Duplicate entry '102' for key 'emp.PRIMARY'

**check**

insert into emp values(103,’anu’,16,2,40000,904486326);

ERROR 3819 (HY000): Check constraint 'emp\_chk\_1' is violated.

insert into emp values(103,’anu’,26,3,40000,904486326);

ERROR 1062 (23000): Duplicate entry '103' for key 'emp.PRIMARY'

**not null**

insert into emp values(104,null,21,4,40000,904486332);

ERROR 1048 (23000): Column 'empname' cannot be null

insert into emp values(104,’anju’,21,4,40000,904486332);

ERROR 1062 (23000): Duplicate entry '104' for key 'emp.PRIMARY'

**unique**

insert into emp values(105,’banu’,21,3,50000,904486332);

ERROR 1062 (23000): Duplicate entry '105' for key 'emp.PRIMARY'

insert into emp values(105,’banu’,21,3,50000,904486334);

ERROR 1062 (23000): Duplicate entry '105' for key 'emp.PRIMARY'

**dropping constraints**

alter table emp drop primary key;



alter table emp drop foreign key;



**Result:**

Thus the MySQL statements for executing constraints are executed successfully.

**EX: NO: 3 WHERE CLAUSE CONDITIONS AND IMPLEMENT AGGREGATE FUNCTIONS**

**AIM:**

To execute and verify the SQL commands using where clause conditions and implement aggregate functions.

**PROCEDURE:**

1.Create the table employee with the corresponding fields.

2. Insert the records in the fields.

3. Implement where clause condition and aggregate functions

4. Verify the records and so for.

**Create table 2:**

mysql> create table emp(empid int primary key,empname varchar(20),age int,salary int,deptno int);

Query OK, 1 row affected (0.06 sec)

mysql> insert into emp values(101,'angel',20,20000,1);

Query OK, 1 row affected (0.06 sec)

mysql> insert into emp values(102,'angena',20,30000,1);

Query OK, 1 row affected (0.02 sec)

mysql> insert into emp values(103,'anu',26,40000,2);

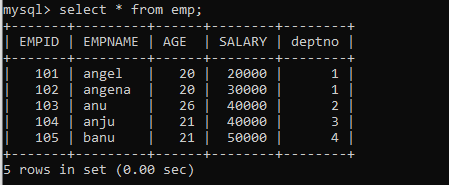
Query OK, 1 row affected (0.02 sec)

mysql> insert into emp values(104,'anju',21,40000,3);

Query OK, 1 row affected (0.01 sec)

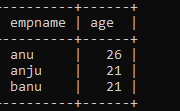
mysql> insert into emp values(105,'banu',21,50000,4);

Query OK, 1 row affected (0.02 sec)



**Q1. Select the employee who all re above age 21.**

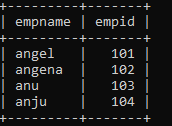
mysql> select empname, age from emp where age>=21;



3 rows in set (0.01 sec)

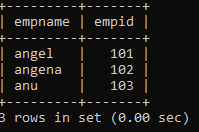
**Q2: Display employee name those having salary between 20000 and 40000**

mysql> select empname,empid from emp where salary between 20000 and 40000;



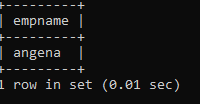
**Q3:Display the employee details who are in department 1 and 2**

**mysql> select empname,empid from emp where deptno in(1,2);**



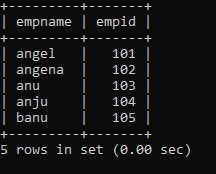
**Q4: Display the employee names that ends with letter a.**

**mysql> select empname from emp where empname like '%a';**



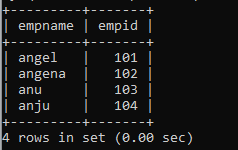
**Q5: Display the employee names having letter a.**

mysql> select empname,empid from emp where empname like '%a%';



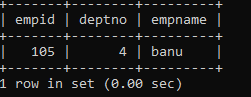
**Q6: Display the employee names starts with letter a.**

mysql> select empname,empid from emp where empname like 'a%';



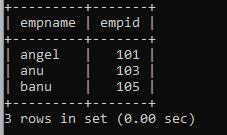
**Q8: display the employee details having employee id greater than 103 and department number 4.**

mysql> select empid,deptno,empname from emp where empid>103 and deptno=4;



**Q10: display the employee details that are not in 102 and 104**

mysql> select empname,empid from emp where empid not in(102,104);



**Aggregate Functions:**

**Create table 2:**

create table works(empid int,companyname varchar2(20),location varchar(20),salary int(5),fk\_id foreign key(empid) references emp(empid));

Query OK, 0 rows affected (0.19 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> insert into works values(101,'infosis','chennai',10000);

Query OK, 1 row affected (0.06 sec)

mysql> insert into works values(102,'wipro','bangalore',20000);

Query OK, 1 row affected (0.03 sec)

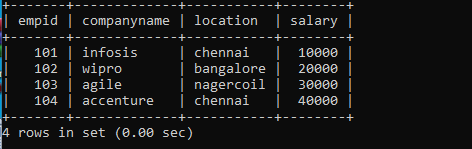
mysql> insert into works values(103,'agile','nagercoil',30000);

Query OK, 1 row affected (0.02 sec)

mysql> insert into works values(104,'accenture','chennai',40000);

Query OK, 1 row affected (0.03 sec)

mysql> select \* from works;

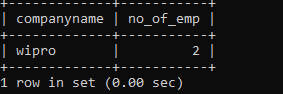


**Q3:Display the company name which one having more than one employee.**

mysql> insert into works values(105,'wipro','bangalore',30000);

Query OK, 1 row affected (0.03 sec)

mysql> select companyname,count(empid) as no\_of\_emp from works group by companyname having count(empid)>1;



**Q4:Display the employee name having maximum salary**

mysql> select max(salary) as maximum from emp;

+----------+

| maximum |

+----------+

| 50000 |

+----------+

**Q5: Display the employee name having maximum salary**

mysql> select min(salary) as minimum from emp;

+---------+

| minimum |

+---------+

| 20000 |

+---------+

**Q6: Display the average salary of employee**

select avg(salary) as average from emp;

mysql> select avg(salary) as average from emp;

+------------+

| average |

+------------+

| 36000.0000 |

+------------+

**Q7: Display the employee name who is earning maximum salary**

mysql> select empid,empname from emp where salary=(select max(salary)from

emp);

+-------+---------+

| empid | empname |

+-------+---------+

| 105 | banu |

+-------+---------+

**Q8: Display the sum of salary of each department**

mysql> select deptno,sum(salary)as total from emp group by deptno;

+--------+-------+

| deptno | total |

+--------+-------+

| 1 | 50000 |

| 2 | 40000 |

| 3 | 40000 |

| 4 | 50000 |

**Q9: Display the sum of salary of each company.**

select companyname,sum(salary) as total from works group by companyname;

mysql> select companyname,sum(salary) as total from works group by companyname;

+-------------+-------+

| companyname | total |

+-------------+-------+

| infosis | 10000 |

| wipro | 50000 |

| agile | 30000 |

| accenture | 40000 |

**Q10: Display company name which one is having less sum of salary compared to others**

mysql> select companyname from works group by companyname having

sum(salary)<=all(select sum(salary)from works group by companyname);

**+-------------+**

**| companyname |**

**+-------------+**

**| infosis |**

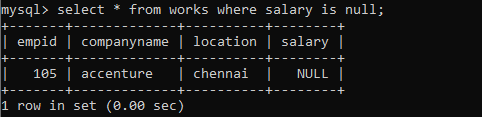
**+-------------+**

mysql> insert into works values(105,'accenture','chennai',null);

Query OK, 1 row affected (0.05 sec)

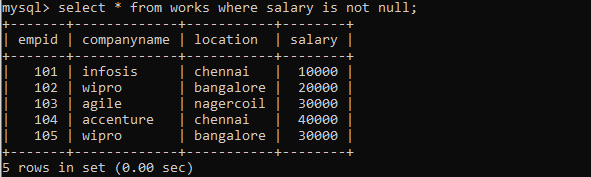
**Q11: Insert the null record**

mysql> select \* from works where salary is null;



**Q12: Display the not null record from the table.**

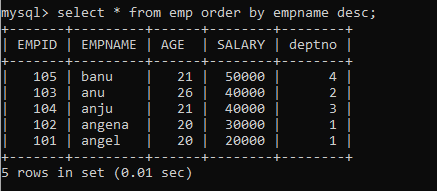
select \* from works where salary is not null;



**order by clause:**

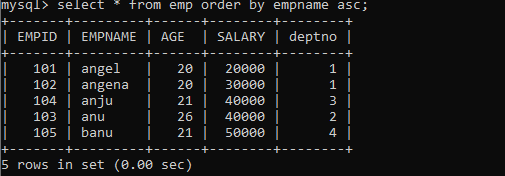
**Q13: Display the employee in decending order**

mysql> select \* from emp order by empname desc;

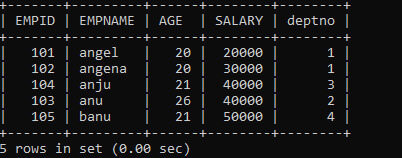


**Q15: Display the employee in decending order**

mysql> select \* from emp order by empname asc;



mysql> select \* from emp order by empname;



**Result:**

Thus the where clause conditions using MySQL statements are verified and executed successfully.

**EX: NO: 4 SIMPLE JOIN OPERATIONS**

**AIM:**

To Query the Database Table and explore Subqueries and simple Join Operations.

**PERFORM THE FOLLOWING:**

1. Create table 1 and table 2.
2. Perform sub queries using where, from and select clauses.
3. Perform single row and multiple rows sub queries.
4. Perform nested sub queries.
5. Perform simple join operations.

# SUB-QUERY:

A sub-query is a SQL query nested inside a larger query. A Sub Query can also be called a Nested/Inner Query.

The sub queries used with:

1. SELECT CLAUSE

2. FROM CLAUSE

3. WHERE CLAUSE

**CREATE TABLE 1:**

mysql> CREATE TABLE EMP(EMPID INTEGER,EMPNAME VARCHAR(20),AGE INTEGER,SALARY INTEGER);

Query OK, 0 rows affected (0.55 sec)

mysql> INSERT INTO EMP VALUES(1,'ASHI',16,10000);

Query OK, 1 row affected (0.15 sec)

mysql> INSERT INTO EMP VALUES(2,'ANI',18,20000);

Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO EMP VALUES(3,'BISMI',17,15000);

mysql> SELECT \* FROM EMP;

+-------+---------+------+--------+

| EMPID | EMPNAME | AGE | SALARY |

+-------+---------+------+--------+

| 1 | ASHI | 16 | 10000 |

| 2 | ANI | 18 | 20000 |

| 3 | BISMI | 17 | 15000 |

+-------+---------+------+--------+

**CREATE TABLE 2:**

mysql> CREATE TABLE REPORT(EMPID INTEGER,AGE INTEGER,SALARY INTEGER);

Query OK, 0 rows affected (0.08 sec)

mysql> INSERT INTO REPORT VALUES(3,17,15000);

Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO REPORT VALUES(1,16,10000);

Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO REPORT VALUES(4,16,20000);

Query OK, 1 row affected (0.01 sec)

mysql> SELECT \* FROM REPORT;

+-------+------+--------+

| EMPID | AGE | SALARY |

+-------+------+--------+

| 3 | 17 | 15000 |

| 1 | 16 | 10000 |

| 4 | 16 | 20000 |

+-------+------+--------+

3 rows in set (0.00 sec)

**SUB-QUERY USING Where Clause:**

A sub-query in a WHERE clause can be used to qualify a column against a set of rows.

# SYNTAX:

**SELECT\* FROM table\_name1 WHERE column\_name1 IN(SELECT column\_name1 FROM table\_name2 WHERE condition );**

mysql> SELECT \* FROM REPORT WHERE EMPID IN(SELECT EMPID FROM EMP WHERE EMPNAME='BISMI');

+-------+------+--------+

| EMPID | AGE | SALARY |

+-------+------+--------+

| 3 | 17 | 15000 |

+-------+------+--------+

1 row in set (0.08 sec)

# SUB-QUERY USING FROM CLAUSE:

FROM clause can be used to specify a sub-query expression in SQL. The relation produced by the sub-query is then used as a new relation on which the outer query is applied.

# SYNTAX:

**SELECT a.column\_name1, b.column\_name2 FROM table\_name1 a, ( SELECT b.column\_name2, function(variable) as b.column\_name FROM table\_name GROUP BYcolumn\_name )b Where Condition;**

mysql> SELECT A.EMPNAME,B.AGE FROM EMP A,(SELECT EMPID,AVG(AGE)AS AGE FROM REPORT GROUP BY EMPID)B WHERE A.EMPID=B.EMPID;

+---------+---------+

| EMPNAME | AGE |

+---------+---------+

| ASHI | 16.0000 |

| BISMI | 17.0000 |

+---------+---------+

2 rows in set (0.03 sec)

mysql> SELECT \* FROM EMP WHERE SALARY=(SELECT MIN(SALARY) FROM EMP);

+-------+---------+------+--------+

| EMPID | EMPNAME | AGE | SALARY |

+-------+---------+------+--------+

| 1 | ASHI | 16 | 10000 |

+-------+---------+------+--------+

1 row in set (0.00 sec)

# SINGLE ROW SUB QUERY USING HAVING CLAUSE:

The HAVING clause is used to filter out groups of records. Because it becomes very useful in filtering on aggregate values such as averages, summations, and count.

# SYNTAX:

**SELECT column\_name1,function(column\_name2) FROM table\_name1 GROUP BY column\_name1 HAVING function (column\_name2) > ( SELECTfunction (column\_name2) FROM table\_name1 WHERE condition);**

mysql> SELECT EMPID,MAX(SALARY) FROM EMP GROUP BY EMPID HAVING MAX(SALARY)>(SELECT MAX(SALARY) FROM EMP WHERE EMPID=1);

+-------+-------------+

| EMPID | MAX(SALARY) |

+-------+-------------+

| 2 | 20000 |

| 3 | 15000 |

+-------+-------------+

2 rows in set (0.01 sec)

**JOINS:**

# JOIN USING USING CLAUSE:

The USING clause specifies which columns to test for equality when two tables are joined.

# SYNTAX:

**SELECT\*FROM table\_name1 JOIN table\_name2 USING (common\_column name1);**

mysql> SELECT \* FROM EMP JOIN REPORT USING(EMPID);

+-------+---------+------+--------+------+--------+

| EMPID | EMPNAME | AGE | SALARY | AGE | SALARY |

+-------+---------+------+--------+------+--------+

| 3 | BISMI | 17 | 15000 | 17 | 15000 |

| 1 | ASHI | 16 | 10000 | 16 | 10000 |

+-------+---------+------+--------+------+--------+

# JOIN USING ON CLAUSE:

ON clause can be used to join columns that have different names. Use the ON clause to specify conditions or specify columns to join.

# SYNTAX:

**SELECT\*FROM table\_name1 JOIN table\_name2 ON (condition\_using\_common\_column\_name);**

mysql> SELECT A.EMPNAME,A.EMPID,B.AGE,B.SALARY FROM EMP A JOIN REPORT B ON(A.EMPID=B.EMPID);

+---------+-------+------+--------+

| EMPNAME | EMPID | AGE | SALARY |

+---------+-------+------+--------+

| BISMI | 3 | 17 | 15000 |

| ASHI | 1 | 16 | 10000 |

+---------+-------+------+--------+

2 rows in set (0.00 sec)

**Result:**

Thus the where clause conditions using MySQL statements are verified and executed successfully.

**EX: NO: 5 NATURAL, EQUI AND OUTER JOIN OPERATIONS**

**AIM:**

To Query the Database Table and explore natural, equi and outer join operations.

**PERFORM THE FOLLOWING:**

1. Create table 1 and table 2.
2. Perform sub queries using natural and outer join operations
3. Analyze the difference of each queries.
4. Report the answers.

**CREATE TABLE 1:**

mysql> create table employee(ename varchar(20),ecity varchar(20),eno int(10));

Query OK, 0 rows affected, 1 warning (0.10 sec)

**CREATE TABLE 2:**

mysql> create table salary(eno int(10),dname varchar(20),esal int(10));

Query OK, 0 rows affected, 2 warnings (0.10 sec)

mysql> desc employee;

+-------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+-------+

| ename | varchar(20) | YES | | NULL | |

| ecity | varchar(20) | YES | | NULL | |

| eno | int | YES | | NULL | |

+-------+-------------+------+-----+---------+-------+

mysql> desc salary;

+-------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+-------+

| eno | int | YES | | NULL | |

| dname | varchar(20) | YES | | NULL | |

| esal | int | YES | | NULL | |

+-------+-------------+------+-----+---------+-------+

mysql> insert into employee values('Ajay','Chennai',11);

Query OK, 1 row affected (0.06 sec)

mysql> insert into employee values('Vijay','Banglore',12);

Query OK, 1 row affected (0.04 sec)

mysql> insert into employee values('Sujay','Chennai',13);

Query OK, 1 row affected (0.03 sec)

mysql> insert into employee values('Jay','Madurai',14);

Query OK, 1 row affected (0.04 sec)

mysql> select\* from employee;

+-------+----------+------+

| ename | ecity | eno |

+-------+----------+------+

| Ajay | Chennai | 11 |

| Vijay | Banglore | 12 |

| Sujay | Chennai | 13 |

| Jay | Madurai | 14 |

mysql> insert into salary values(11,'IT',20000);

Quey OK, 1 row affected (0.03 sec)

mysql> insert into salary values(12,'CSE',20020);

Query OK, 1 row affected (0.03 sec)

mysql> insert into salary values(13,'IT',20050);

Query OK, 1 row affected (0.03 sec)

mysql> insert into salary values(14,'CSE',20000);

Query OK, 1 row affected (0.03 sec)

mysql> select\* from salary;

+------+-------+-------+

| eno | dname | esal |

+------+-------+-------+

| 11 | IT | 20000 |

| 12 | CSE | 20020 |

| 13 | IT | 20050 |

| 14 | CSE | 20000 |

+------+-------+-------+

**NATURAL JOIN OPERATIONS**

**A natural join is a type of join operation that creates an implicit join by combining tables based on columns with the same name and data type**

* There is no need to specify the column names to join.
* The resultant table always contains unique columns.
* It is possible to perform a natural join on more than two tables.
* Do not use the ON clause

### Syntax:

**SELECT [column\_names\*]  FROM table\_name1 NATURAL JOIN table\_name2;**

mysql> select \* from employee natural join salary;

+------+-------+----------+-------+-------+

| eno | ename | ecity | dname | esal |

+------+-------+----------+-------+-------+

| 11 | Ajay | Chennai | IT | 20000 |

| 12 | Vijay | Banglore | CSE | 20020 |

| 13 | Sujay | Chennai | IT | 20050 |

| 14 | Jay | Madurai | CSE | 20000 |

+------+-------+----------+-------+-------+

4 rows in set (0.00 sec)

**CROSS JOIN OPERATION:**

MySQL CROSS JOIN is used to combine all possibilities of the two or more tables and returns the result that contains every row from all contributing tables. The CROSS JOIN is also known as CARTESIAN JOIN, which provides the Cartesian product of all associated tables.

**Syntax:**

**SELECT column-lists FROM table1 CROSS JOIN table2;**

mysql> select \* from employee join salary;

+-------+----------+------+------+-------+-------+

| ename | ecity | eno | eno | dname | esal |

+-------+----------+------+------+-------+-------+

| Jay | Madurai | 14 | 11 | IT | 20000 |

| Sujay | Chennai | 13 | 11 | IT | 20000 |

| Vijay | Banglore | 12 | 11 | IT | 20000 |

| Ajay | Chennai | 11 | 11 | IT | 20000 |

| Jay | Madurai | 14 | 12 | CSE | 20020 |

| Sujay | Chennai | 13 | 12 | CSE | 20020 |

| Vijay | Banglore | 12 | 12 | CSE | 20020 |

| Ajay | Chennai | 11 | 12 | CSE | 20020 |

| Jay | Madurai | 14 | 13 | IT | 20050 |

| Sujay | Chennai | 13 | 13 | IT | 20050 |

| Vijay | Banglore | 12 | 13 | IT | 20050 |

| Ajay | Chennai | 11 | 13 | IT | 20050 |

| Jay | Madurai | 14 | 14 | CSE | 20000 |

| Sujay | Chennai | 13 | 14 | CSE | 20000 |

| Vijay | Banglore | 12 | 14 | CSE | 20000 |

| Ajay | Chennai | 11 | 14 | CSE | 20000 |

+-------+----------+------+------+-------+-------+

**INNER JOIN OPERATION:**

The MySQL Inner Join is used to returns only those results from the tables that **match** the specified condition and hides other rows and columns. MySQL assumes it as a default Join, so it is optional to use the Inner Join keyword with the query.

### Syntax:

**SELECT columns FROM table1 INNER JOIN table2 ON condition1**

**INNER JOIN table3 ON condition2;**

mysql> select \* from employee inner join salary on employee.eno=salary.eno;;

+-------+----------+------+------+-------+-------+

| ename | ecity | eno | eno | dname | esal |

+-------+----------+------+------+-------+-------+

| Ajay | Chennai | 11 | 11 | IT | 20000 |

| Vijay | Banglore | 12 | 12 | CSE | 20020 |

| Sujay | Chennai | 13 | 13 | IT | 20050 |

| Jay | Madurai | 14 | 14 | CSE | 20000 |

**LEFT OUTER JOIN OPERATION:**

The LEFT JOIN returns all the rows from the table on the left even if no matching rows have been found in the table on the right. **Where no matches have been found in the table on the right, NULL is returned.**

**Syntax:**

**SELECT column-lists FROM table1 Left outer join table2;**

mysql> select \* from employee left outer join salary on employee.eno=salary.eno;

+-------+----------+------+------+-------+-------+

| ename | ecity | eno | eno | dname | esal |

+-------+----------+------+------+-------+-------+

| Ajay | Chennai | 11 | 11 | IT | 20000 |

| Vijay | Banglore | 12 | 12 | CSE | 20020 |

| Sujay | Chennai | 13 | 13 | IT | 20050 |

| Jay | Madurai | 14 | 14 | CSE | 20000 |

**RIGHT OUTER JOIN OPERATION:**

RIGHT JOIN is obviously the opposite of LEFT JOIN. The RIGHT JOIN returns all the columns from the table on the right even if no matching rows have been found in the table on the left. Where no matches have been found in the table on the left, NULL is returned.

**Syntax:**

**SELECT column-lists  FROM table1 Right outer join  table2;**

mysql> select \* from employee right outer join salary on employee.eno=salary.eno;

+-------+----------+------+------+-------+-------+

| ename | ecity | eno | eno | dname | esal |

+-------+----------+------+------+-------+-------+

| Ajay | Chennai | 11 | 11 | IT | 20000 |

| Vijay | Banglore | 12 | 12 | CSE | 20020 |

| Sujay | Chennai | 13 | 13 | IT | 20050 |

| Jay | Madurai | 14 | 14 | CSE | 20000 |

+-------+----------+------+------+-------+-------+

4 rows in set (0.00 sec)

Result:

Thus the MYSQL commands to execute the natural, equi and outer join operations are executed and verified successfully.

**EX: NO: 6 PROCEDURES AND USER DEFINED FUNCTIONS**

**AIM:**

To create procedures and user defined functions using PL/SQL block.

**PERFORM THE FOLLOWING:**

1. Create tables.
2. Create procedures and functions.
3. Call procedures and functions to perform listed operations
4. Report the answers.

**PROCEDURE:**

A procedure (often called a stored procedure) is a **collection of pre-compiled SQL statements** stored inside the database. It is a subroutine or a subprogram in the regular computing language. **A procedure always contains a name, parameter lists, and SQL statements**.

**Syntax:**

**CREATE PROCEDURE ProcedureName**

**BEGIN**

**SQL Statements**

**END**

1. Write a pl/sql program to find the sum &avg marks of all the student using procedures.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STUD | | | | |
| RollNo | Name | M1 | M2 | M3 |
|  |  |  |  |  |

**Create Table1:**

mysql> create table stud(rollno int,name varchar(),m1 int,m2 int,m3 int);

**INSERT VALUES INTO THE TABLE:**

insert into stud values(1,’abi’,50,60,70);

insert into stud values(2,’bob’,70,60,50);

insert into stud values(3,’tom’,80,60,70);

**Create Procedure:**

mysql> delimiter &&

mysql> create procedure mycalc()

-> begin

-> select name,m1+m2+m3 as total,(m1+m2+m3)/3 as aver from stud;

-> end

-> &&

mysql> delimiter ;

->&&

**Call Procedure:**

mysql> call mycalc();

->&&

Out PUT:

|  |  |  |
| --- | --- | --- |
| NAME | Total | Average |
| Abi  Bob  tom | 180  180  210 | 60  60  70 |

1. Write a pl/sql program to find the product of 3 numbers in a procedure using in & out parameter.

**Create Procedure:**

mysql> delimiter &&

mysql> create procedure myproduct(n1 int,n2 int,n3 int,out result int)

-> begin

-> set result=n1\*n2\*n3;

-> end

-> &&

mysql> delimiter ;

->&&

mysql> call myproduct(3,4,3,@ans);

->&&

mysql> select @ans;

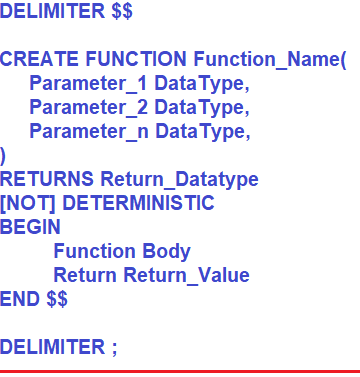
->&&

Output:

|  |
| --- |
| @ans |
| 36 |

**USER DEFINED FUNCTIONS**

The function which is defined by the user is called a user-defined function. MySQL user-defined functions may or may not have parameters that are optional, but it always returns a single value that is mandatory. The returned value which is return by the MySQL Function can be of any valid MySQL data type.



1. Write a pl/sql program to calculate age in user defined function

**Create Table:**

mysql> create table employee(empid int primary key,name varchar(50),salary int,dob date);

Query OK, 0 rows affected (1.61 sec)

**Inserting record:**

mysql> insert into employee values(1001,'pragya',10000,'2001-02-28');

Query OK, 1 row affected (0.12 sec)

mysql> insert into employee values(1002,'anu',20000,'2002-05-28');

Query OK, 1 row affected (0.08 sec)

mysql> insert into employee values(1003,'bob',30000,'2000-01-18');

Query OK, 1 row affected (0.03 sec)

**Create Function:**

mysql> DELIMITER &&

mysql> CREATE FUNCTION Func\_Calculate\_Age(Age date)

-> RETURNS INT DETERMINISTIC

-> BEGIN

-> DECLARE TodayDate DATE;

-> SELECT CURRENT\_DATE() INTO TodayDate;

-> RETURN YEAR(TodayDate)-YEAR(Age);

-> END

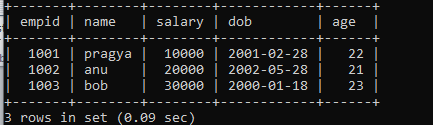
-> &&

Query OK, 0 rows affected (0.03 sec)

**Calling Function:**

mysql> select empid,name,salary,dob,func\_calculate\_age(dob)as age from employee;

-> &&



Result:

Thus the procedures user defined function were created and executed successfully.