SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Ex.no:1	SQL data definition language commands

Aim: To write a program for SQL data definition language commands on sample exercises

Data definition language (DDL):

Used to define database structure or pattern

used to create schema, tables, indexes, constraints

create skeleton of the database

Some of commands are

create

alter

drop

truncate

Rename

```
mysql> create table emp(id integer(10), name varchar(10));
Query OK, 0 rows affected, 1 warning (0.02 sec)

mysql> desc emp;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int         | YES  |     | NULL    |       |
| name  | varchar(10) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
mysql> alter table emp add(dept varchar(10));
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc emp;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int         | YES  |     | NULL    |       |
| name  | varchar(10) | YES  |     | NULL    |       |
| dept  | varchar(10) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Create command :

Used to create objects in database

syntax : create table table_name(column1 datatype , ...
columnN datatype);

Create table emp (id integer(10), name varchar(10));

Alter command :

Used to add, delete or modify columns in an existing table

Syntax: Alter table table_name add column_name
datatype ;

alter table emp add (dept varchar(10));

```
mysql> alter table emp drop column dept;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table emp rename to emp1;
Query OK, 0 rows affected (0.01 sec)

mysql> alter table emp rename to emp1;
ERROR 1146 (42S02): Table 'db.emp' doesn't exist
mysql> desc emp1;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int         | YES  |     | NULL    |       |
| name  | varchar(10) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
mysql> select * from emp1;
Empty set (0.00 sec)

mysql> drop table emp1;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from emp1;
ERROR 1146 (42S02): Table 'db.emp1' doesn't exist
mysql>
```


Rename command:

Used to rename the table

Syntax: alter table current_name rename to target_name;

alter table emp rename to emp1;

Drop table command:

Used to drop an existing table in a database

Syntax: Drop table table_name;

Drop table emp1;

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Result:

Thus the query for SQL data definition language commands are written, executed & the outputs are verified successfully.

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Ex.no: 2	SQL data manipulation language commands

Aim: To write a program for SQL data manipulation language commands on sample exercise

Data manipulation language

DML commands are the most frequently used SQL commands & is used to query & manipulate the existing database objects

Some of the commands are:

- Insert
- select
- Update
- Delete

```
mysql> create table person(pid integer(100), name varchar(20), city varchar(
20));
Query OK, 0 rows affected, 1 warning (0.02 sec)
```

```
mysql> insert into person values(001,'john','chennai');
Query OK, 1 row affected (0.01 sec)

mysql> insert into person values(001,'john','chennai');
Query OK, 1 row affected (0.00 sec)

mysql> insert into person values(002,'prakash','chennai');
Query OK, 1 row affected (0.01 sec)

mysql> insert into person values(003,'ravi','pattinapakkam');
Query OK, 1 row affected (0.00 sec)

mysql> select * from person;
+------+---------+---------------+
| pid  | name    | city          |
+------+---------+---------------+
|    1 | john    | chennai       |
|    1 | john    | chennai       |
|    2 | prakash | chennai       |
|    3 | ravi    | pattinapakkam |
+------+---------+---------------+
4 rows in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Insert:

This is used to add one or more rows to a table. The values are seperated by commas & the data types char & date are enclosed in apostrophes.

Inserting a single row into a table:

insert into <table name> values (<field value 1, ... field value 2);

Inserting more than one record using a single insert command:

insert into <table name> values(& field name 1 ... & fieldname n);

Insert into person values(001, 'John', 'chennai');

Insert into person values(002, 'Prakash', 'chennai');

```
mysql> select pid,name from person;
+------+---------+
| pid  | name    |
+------+---------+
|    1 | john    |
|    1 | john    |
|    2 | prakash |
|    3 | ravi    |
+------+---------+
4 rows in set (0.00 sec)

mysql> select distinct name from person;
+---------+
| name    |
+---------+
| john    |
| prakash |
| ravi    |
+---------+
3 rows in set (0.00 sec)
```

```
mysql> select name from person where pid>1;
+---------+
| name    |
+---------+
| prakash |
| ravi    |
+---------+
2 rows in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Select command:

It is used to retrieve information from the table. It is generally referred to as querying the table. We can either display all columns or only specify column from the table

select pid, name from person;

select distinct name from person;

select name from person where pid >1;

```
mysql> update person set pid=005 where name='john';
Query OK, 2 rows affected (0.01 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from person;
+------+---------+---------------+
| pid  | name    | city          |
+------+---------+---------------+
|    5 | john    | chennai       |
|    5 | john    | chennai       |
|    2 | prakash | chennai       |
|    3 | ravi    | pattinapakkam |
+------+---------+---------------+
4 rows in set (0.00 sec)
```

```
mysql> delete from person where pid=002;
Query OK, 1 row affected (0.00 sec)

mysql> select * from person;
+------+------+---------------+
| pid  | name | city          |
+------+------+---------------+
|    5 | john | chennai       |
|    5 | john | chennai       |
|    3 | ravi | pattinapakkam |
+------+------+---------------+
3 rows in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Update command:

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

update person set pid = 005 where name = 'John';

Delete command:

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.

delete from person where pid = 002;

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Result:

Thus the query for SQL data manipulation language commands on sample exercise was executed & verified successfully.

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Ex. no: 3	Data control language command & transaction control commands

Aim: To create a database using DCL & TCL to manage transactions in database

Transaction control Language TCL:

TCL is used to run changes made by DML statement

TCL can be grouped into a logical transaction

Some commands are:

commit

Rollback

Savepoint

Data control language DCL:

DCL is used to retrieve stored or saved data

DCL execution is transactional. It also has rollback parameters

Some commands are:

Grant

Revoke

```
mysql> create table reg(num integer(5), name varchar(25));
Query OK, 0 rows affected, 1 warning (0.02 sec)

mysql> insert into reg values(101,'a');
Query OK, 1 row affected (0.01 sec)

mysql> insert into reg values(102,'b');
Query OK, 1 row affected (0.00 sec)

mysql> insert into reg values(103,'c');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> GRANT create, select ON db.reg TO 'system'@'host';
ERROR 1410 (42000): You are not allowed to create a user with GRANT
mysql> REVOKE create, select ON db.reg TO 'system'@'host';
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Grant:

It is used to give user access privileges to a database

Syntax: Grant select, update on table name to name-user, another_user;

Grant create, select on db.reg to 'system'@'host'

Revoke:

It is used to take back permissions from the user

Syntax: Revoke select, update on table_name from user1, user2;

Revoke create, select on db.reg to 'system'@'host'

```
mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> update reg set name = 'd' where name = 'c';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from reg;
+------+------+
| num  | name |
+------+------+
|  101 | a    |
|  102 | b    |
|  103 | d    |
+------+------+
3 rows in set (0.00 sec)
```

```
mysql> insert into reg values(110,'abc');
Query OK, 1 row affected (0.00 sec)

mysql> savepoint AA;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into reg values(111,'abz');
Query OK, 1 row affected (0.00 sec)

mysql> savepoint BB;
Query OK, 0 rows affected (0.00 sec)
```

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Commit:

Used to save all the transactions to the database

Syntax : commit;

commit;

Rollback:

It is to rollback a transaction incase of any error occurs

Syntax : Rollback;

Rollback;

Savepoint:

It is to set a savepoint within a transaction.

Syntax : savepoint savepoint_name;

savepoint AA;

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Result:

Thus the query for SQL DCL & TCL on sample exercise was executed & verified successfully.

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Ex. no: 4	Inbuilt Functions in SQL

Aim: To implement query for inbuilt functions in SQL.

SQL functions: Functions in SQL server are the database objects that contain a set of SQL statements to perform a specific task. It is categorized into the following 2 categories:

i) aggregate function
ii) scalar functions

Aggregate functions: They perform calculations on a group of values & then return a single value. Example

Sum(): used to return the sum of a group of values
Max(): Returns a maximum value of a column
First(): Used to return the 1st value of column

Scalar functions: Used to return a single value from the given input value. example

LEN(): Returns the length of the text values in the column
Round(): Rounds off a numeric value to nearest integer
Format(): Used to format how a field must be displayed

```
mysql> select upper('welcome') from dual;
+------------------+
| upper('welcome') |
+------------------+
| WELCOME          |
+------------------+
1 row in set (0.00 sec)

mysql> select upper('hai') from dual;
+--------------+
| upper('hai') |
+--------------+
| HAI          |
+--------------+
1 row in set (0.00 sec)

mysql>  select lower('HAI') from dual;
+--------------+
| lower('HAI') |
+--------------+
| hai          |
+--------------+
1 row in set (0.00 sec)
```

```
mysql> select ltrim('hello world') from dual;
+----------------------+
| ltrim('hello world') |
+----------------------+
| hello world          |
+----------------------+
1 row in set (0.00 sec)
```

```
mysql> select rtrim('hello world')from dual;
+----------------------+
| rtrim('hello world') |
+----------------------+
| hello world          |
+----------------------+
1 row in set (0.00 sec)
```

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

upper & lower:

select upper ('welcome') from dual;

select upper ('hai') from dual;

select lower ('HAI') from dual

ltrim & rtrim;

select ltrim ('hello world') from dual;

select rtrim ('hello world') from dual;

```
mysql> select concat('SRM',' University')from dual;
+-----------------------------+
| concat('SRM',' University') |
+-----------------------------+
| SRM University              |
+-----------------------------+
1 row in set (0.00 sec)
```

```
mysql> select length ('welcome') from dual;
+--------------------+
| length ('welcome') |
+--------------------+
|                  7 |
+--------------------+
1 row in set (0.00 sec)

mysql>  select replace('SRM University', 'University','IST')from dual;
+-------------------------------------------------+
| replace('SRM University', 'University','IST') |
+-------------------------------------------------+
| SRM IST                                         |
+-------------------------------------------------+
1 row in set (0.00 sec)

mysql>  select replace('SRM University', 'University','IST')from dual;
+-------------------------------------------------+
| replace('SRM University', 'University','IST') |
+-------------------------------------------------+
| SRM IST                                         |
+-------------------------------------------------+
1 row in set (0.00 sec)

mysql> select rpad('SRM University',15,'$')from dual;
+-------------------------------+
| rpad('SRM University',15,'$') |
+-------------------------------+
| SRM University$               |
+-------------------------------+
1 row in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

concat :

select concat ('SRM','university') from dual;

Length :

select length ('welcome') from dual;

Replace :

select replace ('srm university','university','ist') from dual;

rpad :

select rpad ('SRM university',15,$) from dual;

```
mysql> select substr('Welcome to SRM University', 4,7)from dual;
+-----------------------------------------+
| substr('Welcome to SRM University', 4,7) |
+-----------------------------------------+
| come to                                 |
+-----------------------------------------+
1 row in set (0.00 sec)
mysql>  select replace('COMPUTER','O','AB')from dual;
+-----------------------------+
| replace('COMPUTER','O','AB') |
+-----------------------------+
| CABMPUTER                   |
+-----------------------------+
1 row in set (0.00 sec)
```

```
mysql> select replace ('university','city','inter') from dual;
+---------------------------------------+
| replace ('university','city','inter') |
+---------------------------------------+
| university                            |
+---------------------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT CURDATE();
+------------+
| CURDATE()  |
+------------+
| 2023-04-10 |
+------------+
1 row in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

substr ( );

select substr ('welcome to SRM university', 4, 7) from dual;

Replace ( );

select replace ('computer', 'O', 'AB') from dual;

select replace ('university', 'city', 'inter') from dual;

current date ( );

select ~~current~~ curdate ( );

```
mysql> SELECT ROUND(CURDATE());
+-------------------+
| ROUND(CURDATE()) |
+-------------------+
|          20230410 |
+-------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_ADD(CURDATE(), INTERVAL 3 MONTH);
+---------------------------------------+
| DATE_ADD(CURDATE(), INTERVAL 3 MONTH) |
+---------------------------------------+
| 2023-07-10                            |
+---------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT LAST_DAY(CURDATE());
+---------------------+
| LAST_DAY(CURDATE()) |
+---------------------+
| 2023-04-30          |
+---------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_ADD(CURDATE(), INTERVAL 20 DAY);
+--------------------------------------+
| DATE_ADD(CURDATE(), INTERVAL 20 DAY) |
+--------------------------------------+
| 2023-04-30                           |
+--------------------------------------+
1 row in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Round ();

select round (curdate ());

date_add ();

select date_add (curdate (), Interval 3 month);

last_day ();

select last_day (curdate ());

date_add ();

select date_add (curdate (), Interval 20 day);

```
mysql> select round(15.6789)from dual;
+-----------------+
| round(15.6789) |
+-----------------+
|              16 |
+-----------------+
1 row in set (0.00 sec)

mysql> select ceil(23.20)from dual;
+--------------+
| ceil(23.20) |
+--------------+
|           24 |
+--------------+
1 row in set (0.00 sec)

mysql> select floor(34.56)from dual;
+---------------+
| floor(34.56) |
+---------------+
|            34 |
+---------------+
1 row in set (0.00 sec)

mysql> select trunc(15.56743)from dual;
ERROR 1046 (3D000): No database selected
mysql> SELECT TRUNCATE(15.56743, 0);
+-----------------------+
| TRUNCATE(15.56743, 0) |
+-----------------------+
|                    15 |
+-----------------------+
1 row in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Round();

select round(15.6789) from dual;

ceil();

select ceil(23.20) from dual;

floor();

select floor(34.56) from dual;

Truncate();

select Truncate(15.56743, 0);

```
mysql> select sign(-345)from dual;
+-------------+
| sign(-345) |
+-------------+
|          -1 |
+-------------+
1 row in set (0.00 sec)

mysql> select abs(-70)from dual;
+-----------+
| abs(-70) |
+-----------+
|       70 |
+-----------+
1 row in set (0.00 sec)

mysql>  select power(10,12) from dual;
+----------------+
| power(10,12)  |
+----------------+
| 1000000000000 |
+----------------+
1 row in set (0.00 sec)

mysql> select power(5,6) from dual;
+-------------+
| power(5,6) |
+-------------+
|      15625 |
+-------------+
1 row in set (0.00 sec)
```

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

sign();

select sign(-345) from dual;

abs();

select abs(-70) from dual;

power();

select power(10,12) from dual;

select power(5,6) from dual;

```
mysql> select mod(11,5) from dual;
+------------+
| mod(11,5) |
+------------+
|          1 |
+------------+
1 row in set (0.00 sec)

mysql> select exp(10) from dual;
+--------------------+
| exp(10)            |
+--------------------+
| 22026.465794806718 |
+--------------------+
1 row in set (0.00 sec)

mysql> select sqrt(225) from dual;
+------------+
| sqrt(225) |
+------------+
|         15 |
+------------+
1 row in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

mod();

select mod(11,5) from dual;

exp();

select exp(10) from dual;

sqrt();

select sqrt(225) from dual;

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Result:

Thus the implementation of inbuilt functions using SQL was executed & verified successfully.

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Ex.no:5	constructing an ER model for the Application

Aim: To construct an ER diagram for an application using RDBMS

Entity Relationship diagram:

An ER diagram shows the relation among entity sets. An entity set is a group of similar entities & these entities can have attributes.

Components of ER diagram

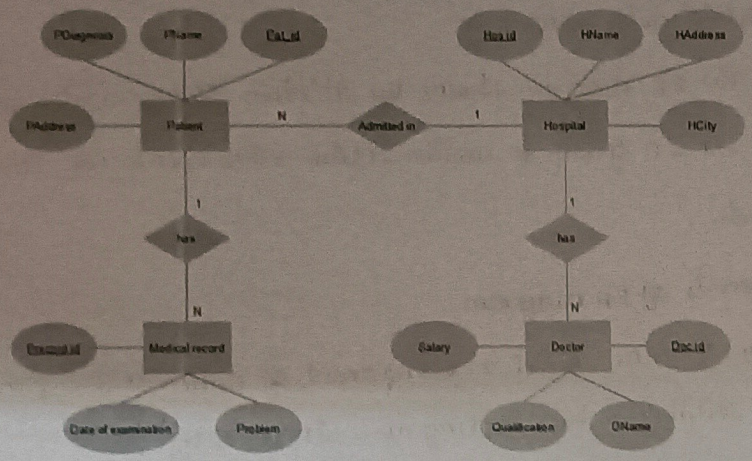
Entity: It is an object or component of data. It is represented as a rectangle in an ER diagram. It has 2 types

- Weak Entity (represented by a double rectangle)
- Strong Entity (represented by a single rectangle)

Attributes: It describes the property of an entity. It is represented as oval in ER diagram. 4 types of attributes.

- Key attribute
- composite attribute
- Multivalued attribute
- Derive attribute

ER Diagrams of Hospital Management System



Relationships : It is represented by a diamond shape in ER diagram, shows the relationship among entities. Types

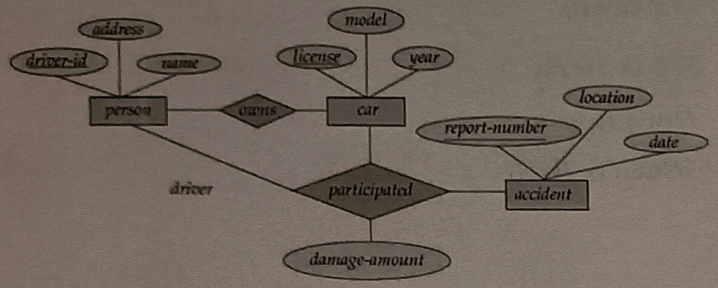
One to one

One to many

Many to one

Many to Many

① Entity Relationship diagram for Hospital management system



SRM INSTITUTE OF SCIENCE & TECHNOLOGY

(2) Entity Relationship diagram for car management system

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Result.

Thus the ER diagram for the given applications have been constructed & verified successfully.

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Ex. no: 6	Nested Queries

Aim: To implement nested queries commands on sample exercise using SQL.

Nested Query - sub query:

Sub query can have more than one level of nesting in one single query. A sql nested query is a select query that is nested inside a select, update, insert, or delete SQL query:

Select command is used to select records from the table
where command is used to identify particular elements
Having command is used to identify particular elements

```
mysql> use db;
Database changed
mysql> -- Create a table
mysql> CREATE TABLE employees (
    ->   id INT AUTO_INCREMENT PRIMARY KEY,
    ->   name VARCHAR(255),
    ->   department VARCHAR(255),
    ->   salary INT
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql>
mysql> -- Insert some values
mysql> INSERT INTO employees (name, department, salary)
    -> VALUES
    ->   ('John Smith', 'Sales', 50000),
    ->   ('Jane Doe', 'Marketing', 60000),
    ->   ('Bob Johnson', 'Sales', 55000),
    ->   ('Alice Lee', 'Finance', 70000),
    ->   ('Tom Williams', 'Marketing', 65000);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql>
mysql> -- Nested query to get the average salary by department
mysql> SELECT department, AVG(salary) as avg_salary
    -> FROM employees
    -> GROUP BY department
    -> HAVING avg_salary > (
    ->   SELECT AVG(salary)
    ->   FROM employees
    -> )
    -> ORDER BY avg_salary DESC;
+------------+------------+
| department | avg_salary |
+------------+------------+
| Finance    | 70000.0000 |
| Marketing  | 62500.0000 |
+------------+------------+
2 rows in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

[Create a table]

```
create table employees ( id  INT  AUTO_Increment Primary
        key , name varchar(255), department varchar(255),
        salary INT);
```

[Insert some values]

```
Insert into employees (name, department, salary) values
    ('John smith', 'sales', 50000);
    :
```

[Nested query]:

```
select department , AVG(salary) as avg_salary
        from employees
        group by department
        having avg_salary > (select avg(salary)
                from employees)
        order by avg_salary desc);
```

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Result: Thus the nested query commands on a sample exercise using SQL was executed & verified successfully.

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Ex. 7	Join Queries

Aim: To implement join query commands using SQL

Join queries:

Used to query data from 2 or more tables, based on a relationship b/w certain columns in these tables

Different types of SQL Join:

- Inner Join
- Self Join
- Outer Join
  - ~~Left join~~ Left outer Join
  - Right outer Join
  - Full outer Join
- Cross Join

```
mysql> CREATE TABLE employees (
    ->   id INT NOT NULL PRIMARY KEY,
    ->   name VARCHAR(50) NOT NULL,
    ->   department VARCHAR(50) NOT NULL,
    ->   salary INT NOT NULL
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> -- Insert some values
mysql> INSERT INTO employees (id, name, department, salary)
    -> VALUES
    ->   (1, 'John', 'IT', 50000),
    ->   (2, 'Jane', 'HR', 60000),
    ->   (3, 'Bob', 'Sales', 70000),
    ->   (4, 'Mary', 'IT', 55000),
    ->   (5, 'Tom', 'HR', 65000),
    ->   (6, 'Alice', 'Sales', 75000);
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from employees;
+----+-------+------------+--------+
| id | name  | department | salary |
+----+-------+------------+--------+
|  1 | John  | IT         |  50000 |
|  2 | Jane  | HR         |  60000 |
|  3 | Bob   | Sales      |  70000 |
|  4 | Mary  | IT         |  55000 |
|  5 | Tom   | HR         |  65000 |
|  6 | Alice | Sales      |  75000 |
+----+-------+------------+--------+
6 rows in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

[create a table employees]

create table employees (id INT not null primary key,

name varchar(50) not null,

department varchar(50) not null,

salary int not null);

[Insert some values into the table]

```
mysql> SELECT e.name AS employee_name, m.name AS manager_name
    -> FROM employees e
    -> INNER JOIN employees m ON e.department = m.department AND e.id != m.i
d
    -> WHERE e.department = 'IT';
+---------------+--------------+
| employee_name | manager_name |
+---------------+--------------+
| Mary          | John         |
| John          | Mary         |
+---------------+--------------+
2 rows in set (0.00 sec)

mysql> SELECT e.name, e.department, e.salary
    -> FROM employees e
    -> INNER JOIN (
    ->   SELECT department, AVG(salary) AS avg_salary
    ->   FROM employees
    ->   GROUP BY department
    -> ) d ON e.department = d.department AND e.salary > d.avg_salary;
+-------+------------+--------+
| name  | department | salary |
+-------+------------+--------+
| Mary  | IT         |  55000 |
| Tom   | HR         |  45000 |
| Alice | Sales      |  75000 |
+-------+------------+--------+
3 rows in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Inner Join:

Returns records that have matching values on both tables

select e.name as employee_name, m.name as manager_name from employees e inner join employees m on e.department and e.id != m.id where e.department = 'IT';

Self Join:

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

select e.name, e.department, e.salary from employees e Inner Join (select department, Avg(salary) from employees group by department) d on e.department = d.department and e.salary > d.avg_salary;

```
mysql> SELECT e.name AS employee_name, m.name AS manager_name
    -> FROM employees e
    -> LEFT OUTER JOIN employees m ON e.department = m.department AND e.id !
= m.id;
+---------------+--------------+
| employee_name | manager_name |
+---------------+--------------+
| John          | Mary         |
| Jane          | Tom          |
| Bob           | Alice        |
| Mary          | John         |
| Tom           | Jane         |
| Alice         | Bob          |
+---------------+--------------+
6 rows in set (0.00 sec)

mysql>
mysql> SELECT e.name AS employee_name, m.name AS manager_name
    -> FROM employees e
    -> LEFT OUTER JOIN employees m ON e.department = m.department AND e.id !
= m.id;
+---------------+--------------+
| employee_name | manager_name |
+---------------+--------------+
| John          | Mary         |
| Jane          | Tom          |
| Bob           | Alice        |
| Mary          | John         |
| Tom           | Jane         |
| Alice         | Bob          |
+---------------+--------------+
6 rows in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Left outer Join:

Returns all records from the left table & the matched records from the right table.

select e.name as employee_name, m.name as manager_name from employees e left outer join employees m on e.department and e.id != m.id;

```
mysql> SELECT e.name AS employee_name, m.name AS manager_name
    -> FROM employees e
    -> RIGHT OUTER JOIN employees m ON e.department = m.department AND e.id
!= m.id;
+---------------+--------------+
| employee_name | manager_name |
+---------------+--------------+
| Mary          | John         |
| Tom           | Jane         |
| Alice         | Bob          |
| John          | Mary         |
| Jane          | Tom          |
| Bob           | Alice        |
+---------------+--------------+
6 rows in set (0.00 sec)

mysql> SELECT d.department, AVG(e.salary) AS avg_salary
    -> FROM employees e
    -> CROSS JOIN (SELECT DISTINCT department FROM employees) d
    -> WHERE e.department = d.department
    -> GROUP BY d.department;
+------------+--------------+
| department | avg_salary   |
+------------+--------------+
| IT         | 52500.0000   |
| HR         | 62500.0000   |
| Sales      | 72500.0000   |
+------------+--------------+
3 rows in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Right outer Join:

Returns all records from the right table & the matched records from the left table

select e.name as employee_name, m.name as manager_name from employees e Right outer join employees m on e.department =m.department and e.id != m.id.

Full outer join

Returns all records when there is a match in either left or right side.

Cross Join: Combines all possibilities of 2 or more tables & returns a result that includes every row from all contributing tables aka cartesian Join.

select d.department, avg(e.salary) as avg_salary from employees e ~~other~~ cross Join (select distinct department from employees) d where e.department = d.department group by d.department.

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Result: Thus the implementation of Join queries using SQL was executed & verified successfully.

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Ex. no: 8	Set operator & views

Aim: To implement set operations & views using SQL

Set operations: They are special type of operators which are used to combine the result of 2 queries. Operators are

Union
Union all
Intersect
Minus

```
mysql> CREATE TABLE employees (
    ->   id INT NOT NULL AUTO_INCREMENT,
    ->   name VARCHAR(50) NOT NULL,
    ->   department VARCHAR(50) NOT NULL,
    ->   salary INT NOT NULL,
    ->   PRIMARY KEY (id)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> -- insert some sample data into the table
mysql> INSERT INTO employees (name, department, salary)
    -> VALUES ('Alice', 'Marketing', 50000),
    ->        ('Bob', 'Sales', 60000),
    ->        ('Charlie', 'HR', 45000),
    ->        ('David', 'Marketing', 55000),
    ->        ('Eve', 'Sales', 65000),
    ->        ('Frank', 'HR', 40000);
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from employees;
+----+---------+------------+--------+
| id | name    | department | salary |
+----+---------+------------+--------+
|  1 | Alice   | Marketing  |  50000 |
|  2 | Bob     | Sales      |  60000 |
|  3 | Charlie | HR         |  45000 |
|  4 | David   | Marketing  |  55000 |
|  5 | Eve     | Sales      |  65000 |
|  6 | Frank   | HR         |  40000 |
+----+---------+------------+--------+
6 rows in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

[Create a table employees & input the values into the table]

```
mysql> SELECT name, department, salary
    -> FROM employees
    -> WHERE department = 'Marketing'
    -> UNION
    -> SELECT name, department, salary
    -> FROM employees
    -> WHERE department = 'Sales';
+-------+------------+--------+
| name  | department | salary |
+-------+------------+--------+
| Alice | Marketing  |  50000 |
| David | Marketing  |  55000 |
| Bob   | Sales      |  60000 |
| Eve   | Sales      |  65000 |
+-------+------------+--------+
4 rows in set (0.00 sec)

mysql> ^C
mysql> SELECT name, department, salary
    -> FROM employees
    -> WHERE department = 'Marketing'
    -> UNION ALL
    -> SELECT name, department, salary
    -> FROM employees
    -> WHERE department = 'Sales';
+-------+------------+--------+
| name  | department | salary |
+-------+------------+--------+
| Alice | Marketing  |  50000 |
| David | Marketing  |  55000 |
| Bob   | Sales      |  60000 |
| Eve   | Sales      |  65000 |
+-------+------------+--------+
4 rows in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Union : Used to combine the result of 2 select statements. Duplicate rows will be eliminated from the results obtained after performing the UNION operation

select name, department, salary from employees where department = 'marketing' union select name, department, salary from employees where department = 'sales'.

Union all : combine all the records from both the queries. Duplicate rows will not be eliminated from the results obtained after performing union all operation.

select name, department, salary from employees where department = 'marketing' union all select name, department, salary from employees where department = 'sales'.

```
mysql> SELECT name, department, salary
    -> FROM employees
    -> WHERE department = 'Marketing'
    -> INTERSECT
    -> SELECT name, department, salary
    -> FROM employees
    -> WHERE department = 'Sales';
Empty set (0.00 sec)

mysql> SELECT name, department, salary
    -> FROM employees
    -> WHERE department = 'Marketing'
    -> MINUS
    -> SELECT name, department, salary
    -> FROM employees
    -> WHERE department = 'Sales';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual t
hat corresponds to your MySQL server version for the right syntax to use nea
r 'MINUS
SELECT name, department, salary
FROM employees
WHERE department = 'Sales'' at line 4
mysql> SELECT e1.name, e1.department, e1.salary
    -> FROM employees e1
    -> LEFT JOIN employees e2
    ->   ON e1.name = e2.name AND e1.department <> e2.department
    -> WHERE e1.department = 'Marketing'
    ->   AND e2.department IS NULL;
+-------+------------+--------+
| name  | department | salary |
+-------+------------+--------+
| Alice | Marketing  |  50000 |
| David | Marketing  |  55000 |
+-------+------------+--------+
2 rows in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Intersect : used to combine 2 select statements, but it only returns the records which are common from both select statements.

select name, department, salary from employees where department = 'marketing' intersect select name, department, salary from employees where department = 'sales';

Minus : It displays the rows which are present in the first query but absent in the record query with no duplicates.

select name, salary from employees where department = 'marketing' minus select name, salary from employees where department = 'sales'.

```
mysql> CREATE TABLE customers (
    ->   id INT NOT NULL AUTO_INCREMENT,
    ->   name VARCHAR(50) NOT NULL,
    ->   email VARCHAR(50) NOT NULL,
    ->   phone VARCHAR(20),
    ->   city VARCHAR(50),
    ->   PRIMARY KEY (id)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> -- insert some sample data into the table
mysql> INSERT INTO customers (name, email, phone, city)
    -> VALUES ('Alice', 'alice@example.com', '123-456-7890', 'New York'),
    ->        ('Bob', 'bob@example.com', '234-567-8901', 'Los Angeles'),
    ->        ('Charlie', 'charlie@example.com', '345-678-9012', 'Chicago'),
    ->        ('David', 'david@example.com', '456-789-0123', 'Houston'),
    ->        ('Eve', 'eve@example.com', '567-890-1234', 'Miami'),
    ->        ('Frank', 'frank@example.com', '678-901-2345', 'Seattle');
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from customers;
+----+---------+-------------------+--------------+-------------+
| id | name    | email             | phone        | city        |
+----+---------+-------------------+--------------+-------------+
|  1 | Alice   | alice@example.com | 123-456-7890 | New York    |
|  2 | Bob     | bob@example.com   | 234-567-8901 | Los Angeles |
|  3 | Charlie | charlie@example.com | 345-678-9012 | Chicago   |
|  4 | David   | david@example.com | 456-789-0123 | Houston     |
|  5 | Eve     | eve@example.com   | 567-890-1234 | Miami       |
|  6 | Frank   | frank@example.com | 678-901-2345 | Seattle     |
+----+---------+-------------------+--------------+-------------+
6 rows in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

[create a table customers & enter the values into the table]

```
mysql> CREATE VIEW ny_customers AS
    -> SELECT id, name, email, phone, city
    -> FROM customers
    -> WHERE city = 'New York';
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> SELECT * FROM ny_customers;
+----+-------+-------------------+--------------+----------+
| id | name  | email             | phone        | city     |
+----+-------+-------------------+--------------+----------+
|  1 | Alice | alice@example.com | 123-456-7890 | New York |
+----+-------+-------------------+--------------+----------+
1 row in set (0.00 sec)

mysql> UPDATE ny_customers
    -> SET email = 'alice@newexample.com'
    -> WHERE name = 'Alice';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM ny_customers;
+----+-------+----------------------+--------------+----------+
| id | name  | email                | phone        | city     |
+----+-------+----------------------+--------------+----------+
|  1 | Alice | alice@newexample.com | 123-456-7890 | New York |
+----+-------+----------------------+--------------+----------+
1 row in set (0.00 sec)
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Views : hides the complexity of data and restricts unnecessary access to database

create view :

create view ny_customers as select id, name, email, phone, city from customers where city = 'new york'.

Update view :

Update ny_customers set mail = 'alice@gmail.com' where name = 'alia';

```
mysql> DELETE FROM my_customers
    -> WHERE name = 'Charlie';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM my_customers;
+----+------+-------------------+--------------+----------+
| id | name | email             | phone        | city     |
+----+------+-------------------+--------------+----------+
|  1 | Alice | alice@newexample.com | 123-456-7890 | New York |
+----+------+-------------------+--------------+----------+
1 row in set (0.00 sec)

mysql> DROP VIEW my_customers;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM my_customers;
ERROR 1146 (42S02): Table 'db.my_customers' doesn't exist
mysql>
```

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Delete view:

Delete from my-customers where name = 'John';

Drop view:

Drop view my-customers;

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Result: Thus the implementation of set operations & view commands was executed & verified successfully.

Ex. no: 9	PL/SQL conditional & iterative statements

Aim: To write a program on PL/SQL conditional & iterative statements.

Conditional & Iterative statements:

PL/SQL supports the programming language features like conditional statements & iterative statements

```
DECLARE
   age NUMBER := 20;
BEGIN
   IF age > 18 THEN
      DBMS_OUTPUT.PUT_LINE('This person is an adult');
   END IF;
END;
```

OUTPUT:

```
This person is an adult
```

```
DECLARE
   age NUMBER := 17;
BEGIN
   IF age >= 18 THEN
      DBMS_OUTPUT.PUT_LINE('This person is an adult');
   ELSE
      DBMS_OUTPUT.PUT_LINE('This person is not an adult');
   END IF;
END;
```

OUTPUT:

```
This person is not an adult
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

PL / SQL IF statement :

Syntax :

(IF-Then statement) :

If condition

Then

statement

End if ;

PL /SQL IF then Else statement

Syntax :

If condition

then

{... statements when condition is true }

else

{.. statements when condition is false }

End if ;

```
DECLARE
   i NUMBER := 1;
BEGIN
   WHILE i <= 5 LOOP
      DBMS_OUTPUT.PUT_LINE('The value of i is: ' || i);
      i := i + 1;
   END LOOP;
END;
```

OUTPUT:

```
The value of i is: 1
The value of i is: 2
The value of i is: 3
The value of i is: 4
The value of i is: 5
```

```
DECLARE
   i NUMBER;
BEGIN
   FOR i IN 1..5 LOOP
      DBMS_OUTPUT.PUT_LINE('The value of i is: ' || i);
   END LOOP;
END;
```

OUTPUT:

```
The value of i is: 1
The value of i is: 2
The value of i is: 3
The value of i is: 4
The value of i is: 5
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

PL/SQL while loop:

Used when a set of statements has to be executed as long as a condition is true

Syntax:

while <condition>
loop statements;
End loop;

PL/SQL for loop:

Used when we want to execute a set of statement for a predetermined no. of times.

Syntax:

For counter IN initial_value .. final_value loop
loop statements;
End loop;

initial_value: start integer value

final_value: End integer value.

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Result:

Thus a program on PL/SQL conditional & iterative statements was executed & verified successfully.

Ex.no:10	PL/SQL procedures.

Aim: To write a program on PL/SQL procedure on sample exercises.

PL/SQL procedures:

The PL/SQL stored procedure is a PL/SQL block which performs one or more than one specific tasks. It contains

i) header
ii) body

Header: contains the name of the procedure & the parameters or variables passed to the procedure

Body: contains a declaration section, execution section & exception section similar to a general PL/SQL block

```
CREATE OR REPLACE PROCEDURE add_numbers (
    num1 IN NUMBER,
    num2 IN NUMBER,
    result OUT NUMBER
) IS
BEGIN
    result := num1 + num2;
END;
```

```
DECLARE
    x NUMBER := 10;
    y NUMBER := 20;
    z NUMBER;
BEGIN
    add_numbers(x, y, z);
    DBMS_OUTPUT.PUT_LINE('The sum of ' || x || ' and ' || y || ' is ' || z);
END;
```

OUTPUT:

```
The sum of 10 and 20 is 30
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Syntax:

create [or replace] procedure procedure-name [(parameters)]

is

[declaration-section]

Begin

executable-section

[exception]

End [procedure-name];

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Result: Thus a program on PL/SQL procedures was verified & executed successfully.

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Ex.no: 11	PL/SQL functions

Aim: To write a program on PL/SQL functions

PL/SQL functions :

The PL/SQL function is very similar to PL/SQL procedure

The main difference b/w procedure & a function is a function must always return a value & on the other hand a procedure may or may not return a value.

```
CREATE OR REPLACE FUNCTION add_numbers (
   num1 IN NUMBER,
   num2 IN NUMBER
) RETURN NUMBER IS
   result NUMBER;
BEGIN
   result := num1 + num2;
   RETURN result;
END;
```

```
DECLARE
   x NUMBER := 10;
   y NUMBER := 20;
   z NUMBER;
BEGIN
   z := add_numbers(x, y);
   DBMS_OUTPUT.PUT_LINE('The sum of ' || x || ' and ' || y || ' is ' || z);
END;
```

OUTPUT:

```
The sum of 10 and 20 is 30
```


Syntax to create a function:

create [or replace] function_name [parameters]

[(parameter_name [IN / OUT / IN OUT type [...])]

Return return_datatype

{IS / AS}

Begin

<function_body>

End [function_name];

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Result:

Thus a program on PL / SQL function was verified & executed successfully.

Ex. no: 12	PL/SQL cursors

Aim: To write a program on PL/SQL cursors.

PL/SQL cursors: used to refer to a program to fetch & process the rows returned by the SQL statement, one at a time

PL/SQL implicit cursors:

They are automatically created by oracle while an SQL statement is executed, if we don't use an explicit cursor for the statement.

PL/SQL explicit cursors:

They are defined by the programmers to gain more control over the context area. They should be defined in the declaration section of PL/SQL block.

```
CREATE TABLE example_table (
    id NUMBER(10) PRIMARY KEY,
    name VARCHAR2(50),
    age NUMBER(3),
    address VARCHAR2(100)
);

INSERT INTO example_table VALUES (1, 'John', 25, '123 Main St.');
INSERT INTO example_table VALUES (2, 'Jane', 30, '456 Elm St.');
INSERT INTO example_table VALUES (3, 'Bob', 40, '789 Oak St.');
```

```
DECLARE
    cursor_example CURSOR
    IS
        SELECT id, name, age, address FROM example_table;
    v_id NUMBER(10);
    v_name VARCHAR2(50);
    v_age NUMBER(3);
    v_address VARCHAR2(100);
BEGIN
    OPEN cursor_example;
    LOOP
        FETCH cursor_example INTO v_id, v_name, v_age, v_address;
        EXIT WHEN cursor_example%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('ID: ' || v_id || ', Name: ' || v_name || ', Age: ' ||
    END LOOP;
    CLOSE cursor_example;
END;
```

OUTPUT:

```
ID: 1, Name: John, Age: 25, Address: 123 Main St.
ID: 2, Name: Jane, Age: 30, Address: 456 Elm St.
ID: 3, Name: Bob, Age: 40, Address: 789 Oak St.
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

[create a table example_table & input the values into the table]

Syntax to create an explicit cursor:

Cursor cursor_name is select_statement ;

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Result: Thus a program on PL/SQL for cursor was verified & executed successfully.

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Ex.no: 13	PL/SQL exception Handling

Aim: To write a program on PL/SQL exception handling

PL/SQL exception handling:

An errors ~~done to~~ during the program execution is called exception in PL/SQL

PL/SQL facilitates programers to each such condition using exception block in the program & an appropriate action is taken against the error condition

There are 2 types of exceptions:

System-defined exceptions

User-defined exceptions.

```
DECLARE
  -- Declare variables for exception handling
  num1 INTEGER := 10;
  num2 INTEGER := 0;
  result INTEGER;

  -- Define custom exceptions
  division_by_zero EXCEPTION;
  result_out_of_range EXCEPTION;

BEGIN
  -- Divide by zero to trigger an exception
  IF num2 = 0 THEN
    RAISE division_by_zero;
  END IF;

  -- Perform a calculation that produces an out-of-range result
  result := num1 * num2;
  IF result > 100 THEN
    RAISE result_out_of_range;
  END IF;
```

```
EXCEPTION
  -- Handle the custom exceptions
  WHEN division_by_zero THEN
    DBMS_OUTPUT.PUT_LINE('Error: division by zero');
  WHEN result_out_of_range THEN
    DBMS_OUTPUT.PUT_LINE('Error: result out of range');
  -- Handle any other exceptions that may occur
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
```

OUTPUT:

```
Error: division by zero
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Syntax for exception handling:

Declare

<declaration section>

Begin

<executable commands>

exception

<exception handling goes here>

when exception 1 then

exception-1 handling statements

:

when exception n then

exception-n handling statements

End;

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Result: Thus a program on PL/SQL exception handling was verified & executed successfully.

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Ex.no:14	PL/SQL Trigger

Aim: To write a program on PL/SQL Trigger.

PL/SQL Trigger:

Trigger is invoked by oracleengine automatically whenever a specified event occurs.

Trigger is stored into database & invoked repeatedly, when specific condition match.

```sql
-- Create a table called "my_table"
CREATE TABLE my_table (
  id NUMBER,
  name VARCHAR2(50),
  age NUMBER
);

-- Insert some values into "my_table"
INSERT INTO my_table VALUES (1, 'Alice', 25);
INSERT INTO my_table VALUES (2, 'Bob', 30);
INSERT INTO my_table VALUES (3, 'Charlie', 35);

-- Create a log table to track changes to "my_table"
CREATE TABLE my_table_log (
  table_name VARCHAR2(50),
  operation_type VARCHAR2(50),
  operation_time DATE
);
```

```sql
-- Create a trigger called "log_inserts"
CREATE OR REPLACE TRIGGER log_inserts
AFTER INSERT ON my_table
FOR EACH ROW
BEGIN
  -- Insert a new row into the log table
  INSERT INTO my_table_log (table_name, operation_type, operation_time)
  VALUES ('my_table', 'INSERT', SYSDATE);
END;
```


SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Syntax for creating trigger:

create [or replace] trigger trigger_name

{Before | After | instead of}

{Insert [or] | update [or] | delete}

[of col_name]

on table_name

[For each row]

when (condition)

declare

declaration-statements

Begin

Executable-statements

Exception

Exception handling statements

End;

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Result: Thus a program on PL/SQL Triggers was verified & executed successfully.