LAB MANUAL
OF

ADVANCED DBMS LAB
ETCS 457

Maharaja Agrasen Institute of Technology, PSP area,
Sector – 22, Rohini, New Delhi – 110085

(Affiliated to Guru Gobind Singh Indraprastha University, New Delhi)

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

MAIT

उद्यमेन हि सिध्यन्ति
कार्याणि न मनोरथैः

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

VISION

To nurture young minds in a learning environment of high academic value and imbibe spiritual and ethical values with technological and management competence.

MISSION

The Institute shall endeavor to incorporate the following basic missions in the teaching methodology:

- **Engineering Hardware – Software Symbiosis:** Practical exercises in all Engineering and Management disciplines shall be carried out by Hardware equipment as well as the related software enabling deeper understanding of basic concepts and encouraging inquisitive nature.
- **Life – Long Learning:** The Institute strives to match technological advancements and encourage students to keep updating their knowledge for enhancing their skills and inculcating their habit of continuous learning.
- **Liberalization and Globalization:** The Institute endeavors to enhance technical and management skills of students so that they are intellectually capable and competent professionals with Industrial Aptitude to face the challenges of globalization.
- **Diversification:** The Engineering, Technology and Management disciplines have diverse fields of studies with different attributes. The aim is to create a synergy of the above attributes by encouraging analytical thinking.
- **Digitization of Learning Processes:** The Institute provides seamless opportunities for innovative learning in all Engineering and Management disciplines through digitization of learning processes using analysis, synthesis, simulation, graphics, tutorials and related tools to create a platform for multi-disciplinary approach.
- **Entrepreneurship:** The Institute strives to develop potential Engineers and Managers by enhancing their skills and research capabilities so that they emerge as successful entrepreneurs and responsible citizens.

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

उद्यमेन हि सिध्यन्ति कार्याणि न मनोरथैः

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VISION

To Produce "Critical thinkers of Innovative Technology"

MISSION

To provide an excellent learning environment across the computer science discipline to inculcate professional behavior, strong ethical values, innovative research capabilities and leadership abilities which enable them to become successful entrepreneurs in this globalized world.

- To nurture an **excellent learning environment** that helps students to enhance their problem solving skills and to prepare students to be lifelong learners by offering a solid theoretical foundation with applied computing experiences and educating them about their **professional, and ethical responsibilities**.
- To establish **Industry-Institute Interaction**, making students ready for the industrial environment and be successful in their professional lives.
- To promote **research activities** in the emerging areas of technology convergence.
- To build engineers who can look into technical aspects of an engineering solution thereby setting a ground for producing successful **entrepreneur**.

INDEX OF THE CONTENTS

1. Introduction to the Lab

Lab Objective

This course aims to give students in depth information about database implementation techniques, data storage, representing data elements, database system architecture, the system catalog, query processing and optimization. Advanced DBMS lab provides a platform for practicing various software's such as Oracle, MySQL, PostgreSQL. All these require a thorough practice of various DDL, DCL and DML queries.

Course Outcomes

At the end of the Lab Course, a student will be able to do:

ETCS457.1: To compare different Databases such as PostgreSQL, Oracle, IBM DB2, MySql.

ETCS457.2: Illustrate and install PostgreSql.

ETCC457.3: Design and develop advanced queries using Structured Query Language for PostgreSql.

ETCS457.4: Learn to build procedures, functions and triggers.

ETCS457.5: Learn the skills required to develop a complete data-intensive application.

LAB REQUIREMENTS

Hardware Detail

Intel i3/C2D Processor/4 GB RAM/TB HDD/MB/Lan Card/ Key Board/ Mouse/CD Drive/15" Color Monitor/ UPS	24 Nos
LaserJet Printer	1 No

Software Detail

Mysql 5.7, PostgreSql/Fedora Linux

3. LIST OF EXPERIMENTS

(As prescribed by G.G.S.I.P.U)

Paper Code: ETCS 457	**P**	**C**
Paper: Advanced DBMS	**2**	**1**

Exercise based on

1. Introduction to Postgresql and its installation
2. Comparison of various Databases with PostgreSQL
3. Data Definition Language Commands, Data Manipulation Language Commands and In Built Functions
4. Nested Queries and Join Queries
5. Views
6. Procedure and functions
7. Trigger

4. LIST OF EXPERIMENTS (Beyond the syllabus)

Develop a small project based on Database Design and Implementation of any one of following Systems:

1. Sakila Database - represent a DVD rental store.

2. Employee Database – represent Employee personal information

3. Inventory Database – represent item purchase/order details

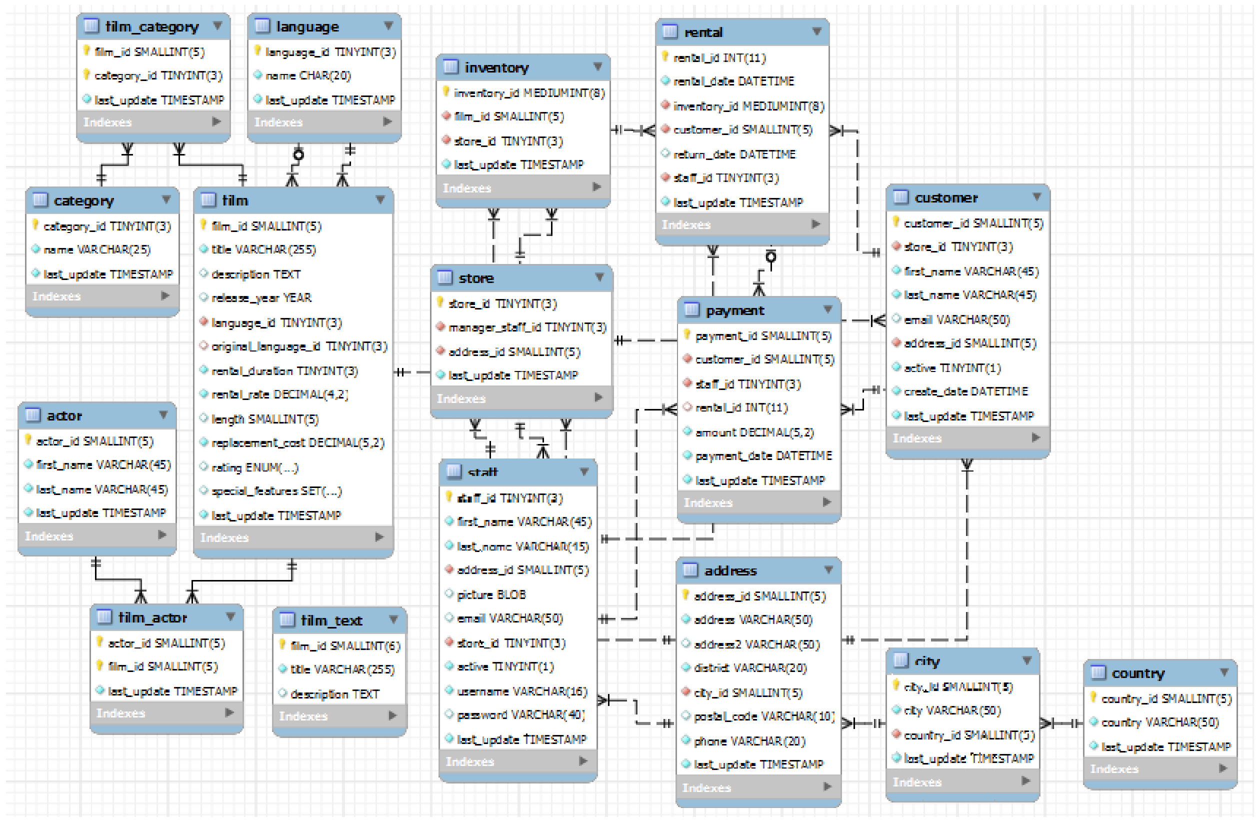
4. NorthWind Database – represents mega mart

NOTE:

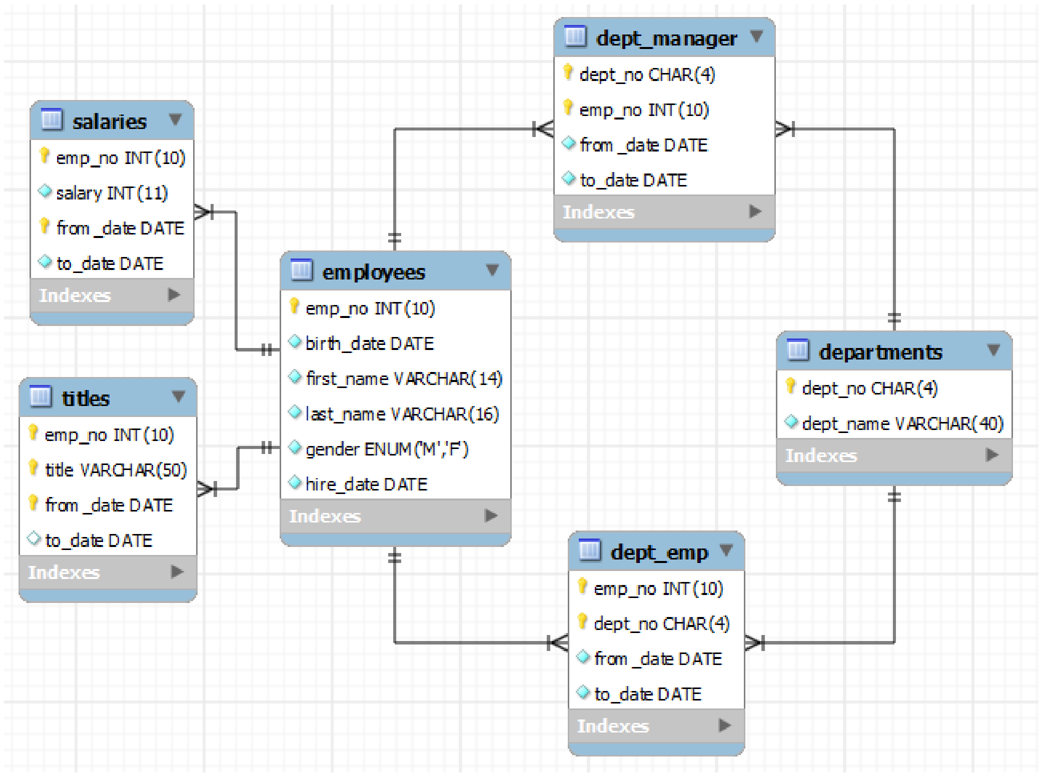
1. Students can select project work of his/her own choice subject to the permission of concerned faculty.
2. The project is to be submitted at the end of the semester along with a project report by the students in group of 3.

Samples of the Databases:

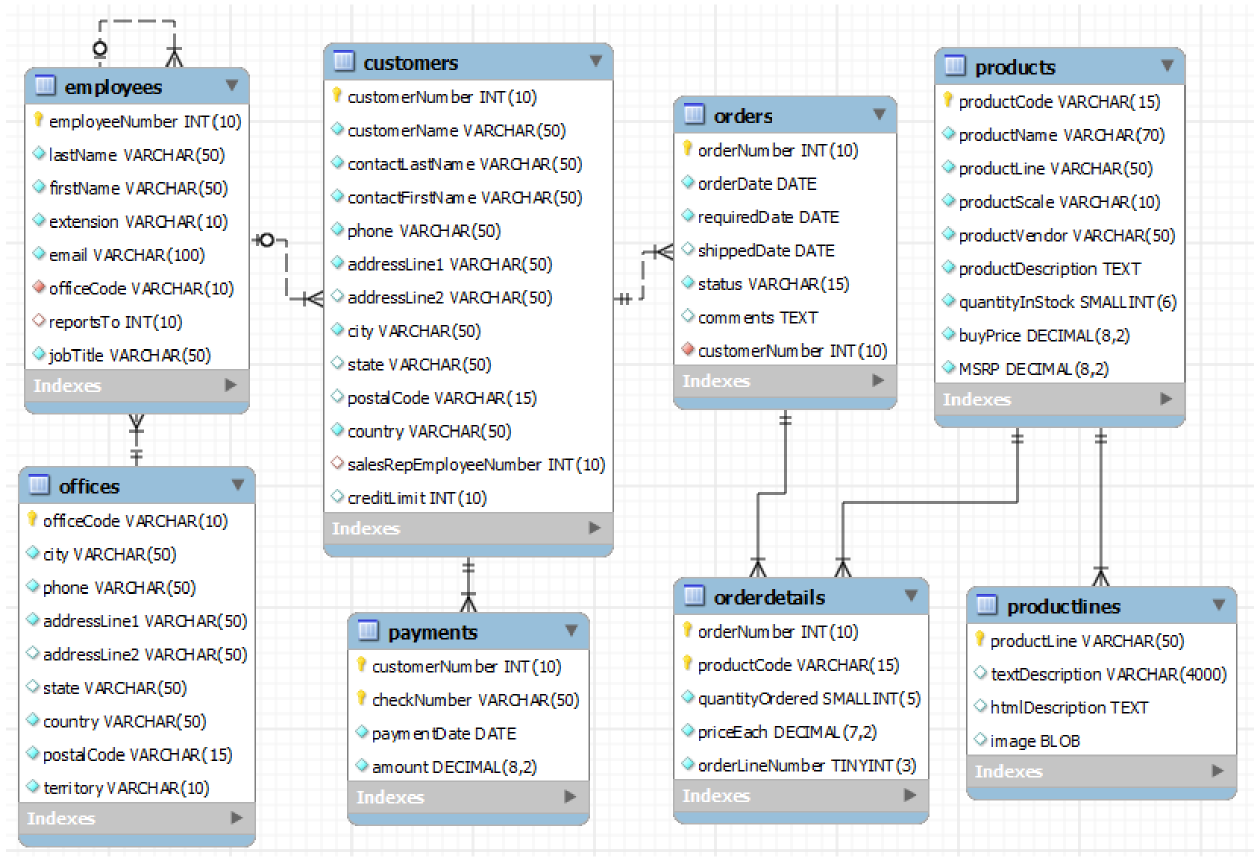
Sakila Database



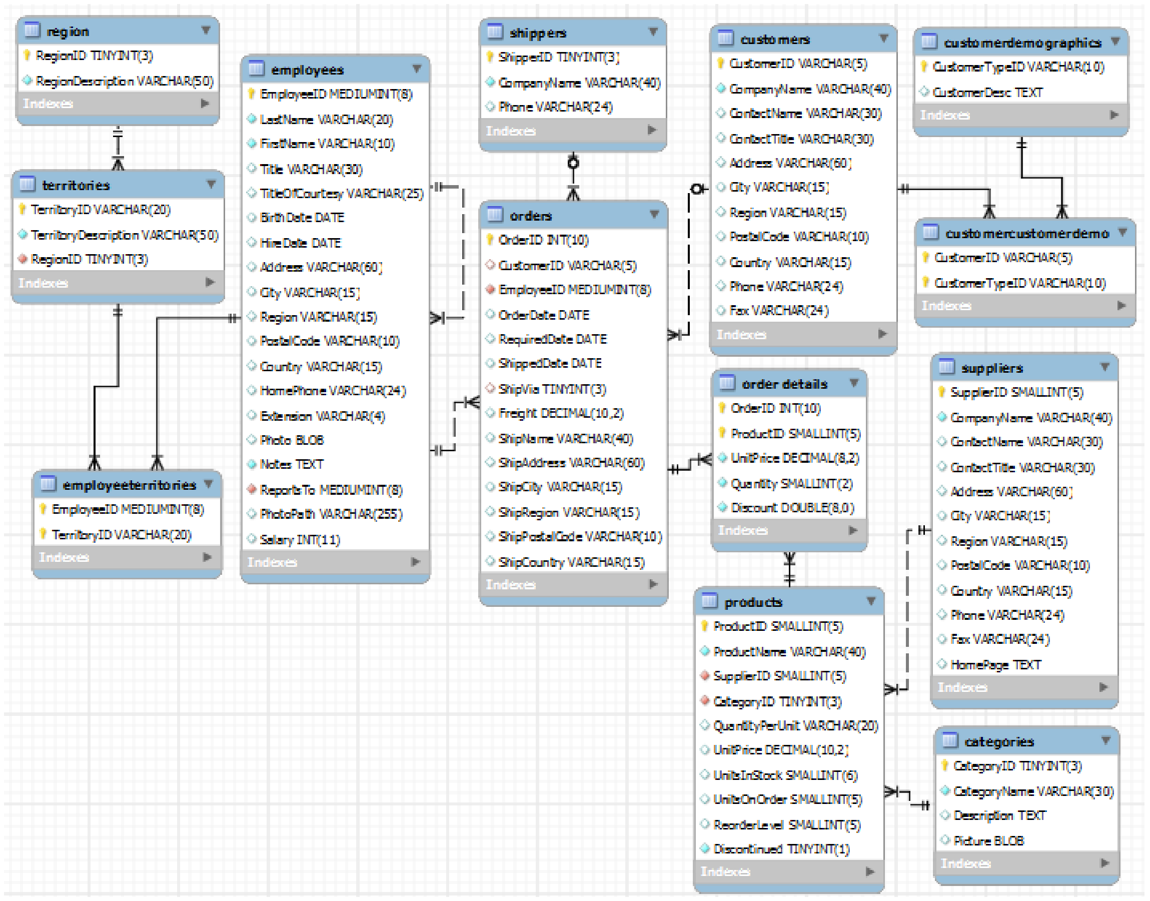
Employee Database



Inventory Database



NorthWind Database



5. FORMAT OF THE LAB RECORD TO BE PREPARED BY THE STUDENTS

The front page of the lab record prepared by the students should have a cover page as displayed below.

NAME OF THE LAB

Paper Code

Font should be (Size 20”, italics bold, Times New Roman)

Faculty name	Student name
	Roll No.:
	Semester:

Font should be (12”, Times Roman)

Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 110085

Font should be (18”, Times Roman)

Index

Exp. no	Experiment Name	Date of performance	Date of checking	Marks	Signature

6. MARKING SCHEME FOR THE PRACTICAL EXAMS

There will be two practical exams in each semester.

i. Internal Practical Exam
ii. External Practical Exam

INTERNAL PRACTICAL EXAM

It is taken by the respective faculty of the batch.

MARKING SCHEME FOR THIS EXAM IS:

Total Marks: 40

Division of 10 marks per practical is as follows:

	Rubrics for : Laboratory (General)			
Sr No.	Experiment Component (LAC)	Max. Marks	Grading Rubrics	
			2 marks	1 mark
1	Practical Performance	2	Completeness of practical, exhibits proficiency in using different types of inputs.	Incomplete practical, unformatted, lacks comments, Demonstrates no proficiency.
2	Output and Validation	2	Output is free of errors and output is obtained. Demonstrates excellent understanding of the concepts relevant to the experiment.	Output contains few logical errors and/or no output is obtained. Demonstrates partial understanding of the concepts relevant to the experiment.
3	Attendance and Viva Questions Answered	4	1. Four marks for answering more than 75% questions. 2. Three marks for answering more than 60% questions. 3. Two marks for answering more than 50% questions. 4. One mark for answering less than 50% questions.	
4	Timely Submission of Lab Record	2	On time submission	Late submission

Each experiment will be evaluated out of 10 marks. At the end of the semester average of 8 best performed practical will be considered as marks out of 40.

EXTERNAL PRACTICAL EXAM

It is taken by the concerned lecturer of the batch and by an external examiner. In this exam student needs to perform the experiment allotted at the time of the examination, a sheet will be given to the student in which some details asked by the examiner needs to be written and at the last viva will be taken by the external examiner.

MARKING SCHEME FOR THIS EXAM IS:

Total Marks: 60

Division of 60 marks is as follows

1.	Sheet filled by the student:	20
2.	Viva Voice:	15
3.	Experiment performance:	15
4.	File submitted:	10

NOTE:

- Internal marks + External marks = Total marks given to the students
  (40 marks) (60 marks) (100 marks)

- Experiments given to perform can be from any section of the lab.

7. INSTRUCTIONS FOR EACH LAB EXPERIMENT

Experiment 1

Aim: Write a brief note on PostGreSQl. List the features of PostGreSQL and its installation steps.

Performance Instructions:

PostgreSQL is a general purpose and object-relational database management system, the most advanced open source database system. PostgreSQL was developed based on POSTGRES 4.2 at Berkeley Computer Science Department, University of California.

PostgreSQL was designed to run on UNIX-like platforms. However, PostgreSQL is designed to be portable so that it could run on various platforms such as Mac OS X, Solaris, and Windows.

Discuss the PostgreSQL features on the following guidelines:

- User-defined types
- Table inheritance
- Sophisticated locking mechanism
- Foreign key referential integrity
- Views, rules, subquery
- Nested transactions (savepoints)
- Multi-version concurrency control (MVCC)
- Asynchronous replication

<u>Installation Steps:</u>

Follow the given steps to install PostgreSQL on the Linux machine. Make sure we are logged in as root before we proceed for the installation.

Select postgresql-9.2.4-1-linux-x64.run for the desired machine.

Now, let us execute it as follows:

```
[root@host]#chmod +x postgresql-9.2.4-1-linux-x64.run

[root@host]# ./postgresql-9.2.4-1-linux-x64.run

-----------------------------------------------------------------------
```

```
Welcome to the PostgreSQL Setup Wizard.


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

Please specify the directory where PostgreSQL will be installed.


Installation Directory [/opt/PostgreSQL/9.2]:
```

Once we launch the installer, it will ask a few basic questions like location of the installation, password of the user who will use database, port number, etc. So keep all of them at their default values except password, which we can provide as per our choice. It will install PostgreSQL at our Linux machine and will display the following message:

```
Please wait while Setup installs PostgreSQL on our computer.


 Installing

 0% ______________ 50% ______________ 100%

 #########################################


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

Setup has finished installing PostgreSQL on our computer.
```

Follow the following post-installation steps to create your database −

```
[root@host]#su - postgres

Password:

bash-4.1$ createdbtestdb

bash-4.1$ psqltestdb

psql (8.4.13, server 9.2.4)


test=#
```

we can start/restart postgres server in case it is not running using the following command −

```
[root@host]# service postgresql restart

Stopping postgresql service:                               [  OK  ]

Starting postgresql service:                               [  OK  ]
```

If our installation is correct, we will have PotsgreSQL prompt **test=#** as shown above.

Viva - Questions:

Q1. What are the different platforms PostgreSQL support?

Q2. Compare PostgreSQL with other DBMSs?

Q3. List features of PostgreSQL.

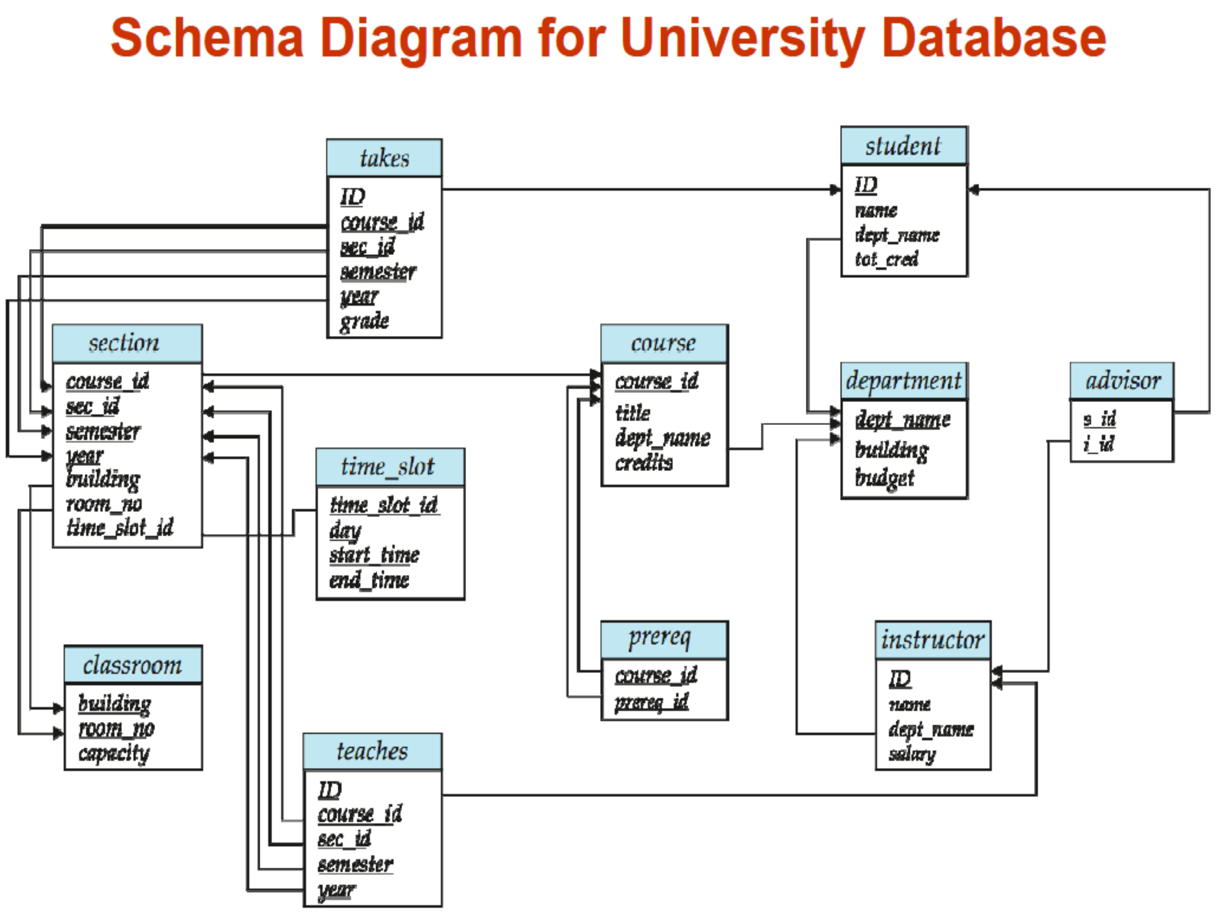
Q4. What are the key differences between MySQL and PostgreSQL? Which Open Source Database to Choose and Why?

Q5. List few limitations of PostgreSQL.

Experiment 2

Aim: Data Definition Language Commands

Write SQL Create table statements to create the following University Database schema. Include all appropriate primary and foreign key declarations. Choose appropriate types for each attribute.



Performance Instructions:

SQL Constraints

SQl Constraints are rules used to limit the type of data that can go into a table, to maintain the accuracy and integrity of the data inside table.

Constraints can be divided into following two types,

- **Column level constraints** : limits only column data
- **Table level constraints** : limits whole table data

Following are the most used constraints that can be applied to a table.

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

using CHECK constraint at Table Level

```
create table Student(s_idint NOT NULL CHECK(s_id> 0),
Name varchar(60) NOT NULL,
Age int);
```

CHECK constraint at Column Level

```
ALTER table Student add CHECK(s_id> 0);
```

CREATE DATABASE

This command will create a database from PostgreSQL shell prompt, but user should have appropriate privilege to create a database.

Syntax of creating the database

```
CREATE DATABASE dbname;
```

where dbname is the name of a database to create.

If we need to choose among already created databases, that can be done using \l command.

Now, type the following command to connect/select a desired database; here, we will connect to the testdb database.

```
postgres=# \c testdb;
```

This database is selected.

CREATE TABLE

```
CREATE TABLE table_name(
  column1 datatype NOT NULL,
  column2 datatype CHECK ( ),
  column3 datatype,
  .....
columnN datatype,
  PRIMARY KEY( one or more columns );
);
```

INSERT INTO

syntax of INSERT INTO

```
INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)
VALUES (value1, value2, value3,...valueN);
```

OR

```
INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);
```

Sample Input

Consider the following schemas:

Sailors(sid: integer, sname: string, rating: integer, age: real)

Boats(bid: integer, bname: string, color: string)

Reserves(sid: integer, bid: integer, day: date)

Here we are creating a sailors table withsidas primary key and NOT NULL are the constraints showing that these fields cannot be NULL while creating records in this table:

CREATE TABLE sailors(sidINT PRIMARY KEY NOT NULL , sname CHAR(10),

rating int , age real , CONSTRAINT age_check check(age>18));

Similarly, for other tables.

CREATE TABLE reserves (sidint not null, bid int not null, day datetime not null,
CONSTRAINT PK_reserves PRIMARY KEY (sid, bid, day), FOREIGN KEY (sid)
REFERENCES sailors(sid), FOREIGN KEY (bid) REFERENCES boats(bid));

We can verify our table if it has been created successfully by using **\d** command, which is used to list down all the tables in an attached database.

```
testdb-# \d sailors
```

Sample Output

Sample Data for Sailors Relation

```
testdb# select * from sailors;

sid | sname  | rating | age

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

  1 | Paul  |1|  20

  2 | Allen |  2|18

  3 | Teddy |  5|30

  4 | Mark  |  9|  55

5 | David | 10|  40

  6 | Kim   |  8| 45

  7 | James | 10|  29
```

Sample Data for Boat Relation

```
testdb# select * from Boat;

 bid | bname  |color

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

  101 | interlake |   blue

  102 | interlake |   red

  103 | clipper|   blue

  104 | Marine|   green

105| Clipper |   red
```

Sample Data for Reserve Relation

```
testdb# select * from reserve;

sid | bid  | date

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

  1 | 101  |1998-10-10

  1 | 102  |  1998-10-10

  2 | 103|1998-11-5

  2 | 101  |1998-9-10

2 | 104 |1998-11-12

  3 | 103 |1998-9-10

4 | 105 |1998-8-5

5| 104  |  1998-11-4

  5 | 103  |1998-8-6

  6 | 105  |  1998-12-2

  7 | 104  |  1998-1-10
```

Viva - Questions:

Q1. What is Relational Database Management system?

Q2. Explain DDL and DML commands.

Q3. Differentiate between DCL and DQL.

Q4. How to enter multiple entries in the table using single command? Write the syntax.

Q5. What are integrity and referential integrity constraints? How they are implemented?

Experiment 3

Aim: Data Manipulation Language&In Built Functions

Write the following simple SQL Queries on the University Schema

1. Find the names of all the students whose total credits are greater than 100
2. Find the course id and grades of all courses taken by any student named 'Tanaka'
3. Find the ID and name of instructors who have taught a course in the Comp. Sci. department, even if they are themselves not from the Comp. Sci. department. .
4. Find the courses which are offered in both 'Fall' and 'Spring' semester (not necessarily in the same year).
5. Find the names of all the instructors from Comp. Sci. department
6. Find the course id and titles of all courses taught by an instructor named 'Srinivasan'
7. Find names of instructors who have taught at least one course in Spring 2009

Use In Built Functions to execute following queries.

1. Find the number of instructors who have never taught any course.
2. Find the total capacity of every building in the university.
3. Find the maximum number of teachers for any single course section.
4. Find all departments that have at least one instructor, and list the names of the departments along with the number of instructors; order the result in descending order of number of instructors.
5. As in the previous question, but this time you should include departments even if they do not have any instructor, with the count as 0.
6. For each student, compute the total credits they have successfully completed, i.e. total credits of courses they have taken, for which they have a non-null grade other than 'F'. Do NOT use the tot_creds attribute of student.
7. Find the name of all instructors who get the highest salary in their department.
8. Find all students who have taken all courses taken by instructor 'Srinivasan'. (This is the division operation of relational algebra.) You can implement it by counting the number of courses taught by Srinivasan, and for each student (i.e. group by student), find the number of courses taken by that student, which were taught by Srinivasan. Make sure to count each course ID only once.
9. Find the total money spent by each department for salaries of instructors of that department.

To test the above queries, make sure you add appropriate data, and include the corresponding insert statements along with your query.

Performance Instructions:

SELECT statement with WHERE clause

```
SELECT column1, column2, columnN
FROM table_name
WHERE [search_condition]
```

We can specify a search condition using comparison or logical operators like >, <, =, LIKE, NOT, etc. in the where clause.

The PostgreSQL AND and OR operators are used to combine multiple conditions to narrow down selected data in a PostgreSQL statement. These two operators are called conjunctive operators.

Operators Allowed in WHERE Clause

Operator	**Description**
=	Equal
!=	Not Equal
>	Greater Than
<	Less Than
>=	Greater Than Equal To
<=	Less Than Equal To
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column
AND Operators	Displays record if Both the Conditions are TRUE.
OR Operator	Displays record if Either a Condition is TRUE.

syntax of AND operator with WHERE clause

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition1] AND [condition2]...AND [conditionN];
```

syntax of OR operator with WHERE clause

```
SELECT column1, column2, columnN
FROM table_name
```

```
WHERE [condition1] OR [condition2]...OR [conditionN]
```

The following is the list of PostgreSQL built-in functions:

- COUNT Function − The PostgreSQL COUNT aggregate function is used to count the number of rows in a database table.
- MAX Function − The PostgreSQL MAX aggregate function allows us to select the highest (maximum) value for a certain column.
- MIN Function − The PostgreSQL MIN aggregate function allows us to select the lowest (minimum) value for a certain column.
- AVG Function − The PostgreSQL AVG aggregate function selects the average value for certain table column.
- SUM Function − The PostgreSQL SUM aggregate function allows selecting the total for a numeric column.
- Numeric Functions − Complete list of PostgreSQL functions required to manipulate numbers in SQL.
- String Functions − Complete list of PostgreSQL functions required to manipulate strings in PostgreSQL.

Min Function Syntax:

Select Min (column name) from Table name where condition;

Numeric functions are used primarily for numeric manipulation and/or mathematical calculations.

1. **POW(X,Y) / POWER(X,Y)**

These two functions return the value of X raised to the power of Y.

```
testdb=# SELECT POWER(3,3);

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

|POWER(3,3)|

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

|27|

+---------------------------------------------------------+
1 row inset(0.00 sec)
```

2. SQRT(X)

This function returns the non-negative square root of X.

```
testdb=#SELECT SQRT(49);

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

|SQRT(49)|

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

|7|

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

1 row inset(0.00 sec)
```

Sample Query

Sample Input

Find sid and sname of the sailors having rating equal to 10 or having age greater than 30.

```
testdb# select sid, sname  from sailors where rating =5 or age>30;
```

Sample Output

```
sid | sname

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

  3 | Teddy |  5 |  30

  4 | Mark  |  9 |  55

5 | David | 10 |  40

  6 | Kim   |  8 |  45
```

Sample Query

Sample Input

Find the sailor who is younger than all and having rating equal to 10.

```
testdb# select MIN (S.age)minimum_age from Sailors S where S.rating=10;
```

Sample Output

```
Minimum_age

-----+---

  29 |
```

Viva - Questions:

Q1. How data stored in Relation A is copied to Relation B?

Q2. What is the default value stored in attribute when a tuple is inserted with missing value in the corresponding attribute? Explain.

Q3. What the different string functions that we use in postgreSql.

Q4. What is the difference between delete, drop and truncate command?

Q5. Why we use Limit function.

Experiment 4

Aim: Nested Subqueries and joins.

Write the following queries using the concept of Subqueries and joins.
1. Find the id and title of all courses which do not require any prerequisites.
2. Find the names of students who have not taken any biology dept courses.
3. Write SQL update queries to perform the following :
1. Give a 10% hike to all instructors in Course Operating System.
2. Increase the tot_creds of all students who have taken the course titled "Genetics" by the number of credits associated with that course.
3. For all instructors who are advisors of atleast 2 students, increase their salary by 50000.
4. Set the credits to 2 for all courses which have less than 5 students taking them (across all sections for the course, across all years/semesters).

Performance Instructions:

A subquery or Inner query or Nested query is a query within another PostgreSQL query and embedded within the WHERE clause.

Subqueries can be used with the SELECT, INSERT, UPDATE and DELETE statements along with the operators like =, <, >, >=, <=, IN, etc.

Subqueries are most frequently used with the SELECT statement.

Subquery with SELECT statement

```
SELECT column_name [, column_name ]

FROM   table1 [, table2 ]

WHERE  column_name OPERATOR

      (SELECT column_name [, column_name ]

      FROM table1 [, table2 ]

      [WHERE])
```

Subquery with INSERT statement

```
INSERT INTO table_name [ (column1 [, column2 ]) ]

   SELECT [ *|column1 [, column2 ] ]

   FROM table1 [, table2 ]

   [ WHERE VALUE OPERATOR
```

Subquery with UPDATE statement

```
UPDATE table
SET column_name = new_value
[ WHERE OPERATOR [ VALUE ]
   (SELECT COLUMN_NAME
   FROM TABLE_NAME)
   [ WHERE) ]
```

A **JOIN** is a means for combining fields from two tables by using values common to each.

Join Types in PostgreSQL are −

- The CROSS JOIN
- The INNER JOIN
- The LEFT OUTER JOIN
- The RIGHT OUTER JOIN
- The FULL OUTER JOIN

Syntax of CROSS JOIN

```
SELECT ... FROM table1 CROSS JOIN table2 ...
```

Syntax of INNER JOIN

```
SELECT table1.column1, table2.column2...

FROM table1

INNER JOIN table2

ON table1.common_filed = table2.common_field;
```

Syntax of LEFT OUTER JOIN

```
SELECT ... FROM table1 LEFT OUTER JOIN table2 ON conditional_expression ...
```

Syntax of RIGHT OUTER JOIN

```
SELECT ... FROM table1 RIGHT OUTER JOIN table2 ON conditional_expression ...
```

Syntax of FULL OUTER JOIN

```
SELECT ... FROM table1 FULL OUTER JOIN table2 ON conditional_expression ...
```

Syntax of GROUP BY and HAVING clause:

```
SELECT [DISTINCT] target-list

FROM relation-list

WHERE condition

GROUP BY grouping-list

HAVING group-condition
```

Sample Query

Sample Input

Find the names of sailors who have reserved boat no. 103.

```
testdb=# SELECT S.sname FROM Sailors S WHERE S.sid IN (SELECT R.sid FROM Reserves R
Where R.bid =103)
```

Alternatively, it can be written as

```
testdb=# SELECT S.sname FROM Sailors S WHERE EXISTS (SELECT R.sid FROM Reserves R
Where R.bid =103 AND S.sid = R.sid);
```

Sample Output

```
sname
+--------
Allen  |
Teddy|
David  |
(3 rows)
```

Sample Query

Sample Input

Find sailors whose rating is greater than that of some sailor called Teddy.

```
testdb=# SELECT * FROM Sailors S S.rating> ANY (SELECT S2.rating FROM Sailors S2
Where S2.sname= ‘Teddy’);
```

Sample Output

```
sid | sname  | rating | age

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

  4 | Mark  |  9 |  55

5 | David | 10 |  40

  6 | Kim   |  8 |  45

  7 | James | 10 |  29
```

Sample Query

Sample Input

For each red boat, find the number of reservations for this boat.

```
testdb=# SELECT B.bid, COUNT(*) AS s-count FROM Sailors S, Boat B, Reserves R WHERE
S.sid = R.sid AND R.bid=B.bid AND B.color=’red’ GROUP BY B.bid;
```

Sample Output

```
bid | s-count

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

  103 | 1

  105 | 2
```

Viva - Questions:

Q1. What are different Clauses used in SQL?

Q2.What are different JOINS used in SQL?

Q3. What are the different methods for creating the subquery.

Q4. Explain how IN operator is used with a subquery. Give Example.

Q5. Can we use joins instead of wring a subquery.

Experiment No. 5

Aim: Views

Write the following queries using Views.

1. **Find out the various courses of different departments.**
2. **Find the names of faculty teaching various courses.**
3. **Find out room no. of IV semester students.**
4. **Find the number of students who have been taught (at any time) by an instructor named 'Srinivasan'. Make sure you count a student only once even if the student has taken more than one course from Srinivasan.**
5. **Find the names of all students whose advisor has taught the maximum number of courses (multiple offerings of a course count as only 1). (Note: this is a complex query, break it into parts by using the with clause.)**

Performance Instructions:

A view can represent a subset of a real table, selecting certain columns or certain rows from an ordinary table. A view can even represent joined tables.

The PostgreSQL views are created using the CREATE VIEW statement. The PostgreSQL views can be created from a single table, multiple tables, or another view.

CREATE VIEW syntax

```
CREATE [TEMP | TEMPORARY] VIEW view_name AS

SELECT column1, column2.....

FROM table_name

WHERE [condition];
```

Sample Query

Sample Input

Create a view from Sailors table having sid , sname and age of the sailors.

```
testdb=# CREATE VIEW sailor_view AS

SELECT sid, sname , age

FROM  sailors;
```

Now, we can query a sailor_viewin a similar way as we query an actual table.

```
testdb=# SELECT * FROM sailors_view;
```

Sample Ouput

```
testdb# select * from sailors_view;

sid | sname  | age

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

  1 | Paul   |  20

  2 | Allen |  18

  3 | Teddy | 30

  4 | Mark   |  55

5 | David |  40

  6 | Kim    |  45

  7 | James |  29
```

Dropping VIEWS

To drop a view, use the DROP VIEW statement with the **view_name**.

DROP VIEW Syntax

```
testdb=# DROP VIEW view_name;
```

Viva - Questions:

Q1. Why we use views?

Q2. Explain different syntax of creating the views.

Q3. How we can update the view**?**

Q4. What are the limitations of views.

Q5. How many types of views are there.

Experiment 6

Aim: Procedure & Function

1. Write the procedure to get the average salary of instructors for branch "CSE"

2.Write a PL/SQL procedure to find the number of students ranging from 100 -70%, 69-60%, 59-50% & below 49% in the course passed as parameter to the procedure.

3.Write a PL/SQL function that accepts department name and returns the total salary of the department. Also write a function to call the function.

Performance Instructions:

Functions − These subprograms return a single value; mainly used to compute and return a value.

Procedures − These subprograms do not return a value directly; mainly used to perform an action.

Each PL/SQL subprogram has a name, and may also have a parameter list.

Declarative Part

It is an optional part. The declarative part for a subprogram does not start with the DECLARE keyword. It contains declarations of types, cursors, constants, variables, exceptions, and nested subprograms.

Executable Part

This is a mandatory part and contains statements that perform the designated action.

Exception-handling

This is an optional part. It contains the code that handles run-time errors.

Syntax forCreating a PROCEDURE

```
CREATE [OR REPLACE] PROCEDURE procedure_name

[(parameter_name[IN | OUT | IN OUT] type [,...])]

{IS | AS}

BEGIN

<procedure_body>
```

```
ENDprocedure_name;
```

Creating a FUNCTION

A function is created using the **CREATE FUNCTION** statement. The simplified syntax for the **CREATE OR REPLACE PROCEDURE** statement is as follows –

```
REATE [OR REPLACE] FUNCTION function_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}
BEGIN
<function_body>
END [function_name];
```

Consider customers table.

```
Select * from customers;

+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
+----+----------+-----+-----------+----------+
```

Sample Function

Sample Input

Create Function to count total number of customers in customers table.

```
CREATE OR REPLACE FUNCTION totalCustomers

RETURN number IS

   total number(2):=0;

BEGIN

   SELECT count(*)into total

   FROM customers;



   RETURN total;
```

```
END;

/
```

Sample output

```
Function created.
```

Viva - Questions:

Q1. Define Stored Procedure? Why they are created?

Q2. Where the Procedures are stored in Database?

Q4. List the different types of stored Procedures?

Q5. Can a stored Procedure be called inside other store Procedure? How?

Q6. Write the Process to execute a Stored Routine.

Experiment 7

Aim: Trigger

1.Write a trigger which shows the old values and new values of student after any updation on grade for a particular course.
2.Write a row trigger to insert the existing values of the instructor table in to a new table when the salary of instructor is updated.

Performance Instructions:

A trigger is a named database object that is associated with a table, and it activates when a particular event (e.g. an insert, update or delete) occurs for the table/views. The statement CREATE TRIGGER creates a new trigger in PostgreSQL

Syntax for creating a trigger

```
CREATE [OR REPLACE ] TRIGGER trigger_name

{BEFORE | AFTER | INSTEAD OF }

{INSERT [OR]| UPDATE [OR]| DELETE}

[OF col_name]

ON table_name

[REFERENCING OLD AS o NEW AS n]

[FOR EACH ROW]

WHEN (condition)

DECLARE

Declaration-statements

BEGIN

Executable-statements

EXCEPTION

Exception-handling-statements

END;
```

CREATE [OR REPLACE] TRIGGER trigger_name	Creates or replaces an existing trigger with the *trigger_name*.
{BEFORE | AFTER | INSTEAD OF}	This specifies when the trigger will be executed.
{INSERT [OR] | UPDATE [OR] | DELETE}	This specifies the DML operation.
[OF col_name]	This specifies the column name that will be updated.
[ON table_name]	This specifies the name of the table associated with the trigger.
[REFERENCING OLD AS o NEW AS n]	This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.
[FOR EACH ROW]	This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected.
WHEN (condition)	This provides a condition for rows for which the trigger would fire.

Sample Trigger

Sample Input

Sample Input that will display the salary difference between the old values and new values.

Create a trigger

The following program creates a row-level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger

```
testdb=# CREATE OR REPLACE TRIGGER display_salary_changes

BEFORE DELETE OR INSERT OR UPDATE ON customers

FOR EACH ROW

WHEN (NEW.ID >0)

DECLARE

sal_diff number;

BEGIN

sal_diff:=:NEW.salary-:OLD.salary;

dbms_output.put_line('Old salary: '||:OLD.salary);

dbms_output.put_line('New salary: '||:NEW.salary);
dbms_output.put_line('Salary difference: '||sal_diff);

END;

/
```

Sample Output

```
Trigger created.
```

Syntax Dropping Trigger

The following is the DROP command, which can be used to drop an existing trigger.

```
testdb=# DROP TRIGGER trigger_name;
```

Viva - Questions:

Q1.Why triggers are needed?

Q2. Write the difference between triggers and stored procedures.

Q3. Is Trigger and Constraint same? Give reasons.

Q4. Discuss trigger variants.

Q5. What are the system privileges that are required by the schema owner(user) to create a trigger on a relation?

Experiment 8

Aim: Develop a small project based on Database Design in PostGreSql

Performance Instructions:

1. Select any oneproject from given list.
2. Write the problem statement of the selected project.

   The problem statement is the initial starting point for a project. It is basically a one to three page statement that everyone on the project agrees with that describes what will be done at a high level. The problem statement is intended for a broad audience and should be written in non-technical terms. It helps the non-technical and technical personnel communicate by providing a description of a problem. It doesn't describe the solution to the problem.

So, basically a problem statement describes **what** needs to be done without describing **how**.

3. List relations and all integrity constraints.
4. Design the schema.
5. Perform the following on your selected Project.
   a. DDL
   b. DML
   c. Nested Queries & Joins
   d. Views
   e. Procedure
   f. Function
   g. Trigger