

Practical : 2

Aim : Consider Schema: Student (student_name, enrollment_no, marks, area, branch)

Write SQL queries on

- a. DDL (eg create, alter, drop, rename, truncate),**
- b. DML (eg. Insert, update, delete etc.),**
- c. DCL (eg. Grant, revoke etc.)**
- d. Built-in Functions (eg. Sum, min, max, avg, count, lower, upper, trim, len etc.)**
- e. Indexes and views: Create and Drop**

Ans.

- a) DDL : Stands for "Data Definition Language." A DDL is a language used to define data structures and modify data. For example, DDL commands can be used to add, remove, or modify tables within in a database. DDLs used in database applications are considered a subset of SQL, the Structured Query Language.

//Create table student

```
CREATE TABLE student(  
    student_name TEXT,  
    enrollment_no INTEGER PRIMARY KEY,  
    cgpa DECIMAL,  
    area TEXT,  
    branch VARCHAR(30)  
);
```

```
temp=# CREATE TABLE student(  
temp=# student_name TEXT,  
temp=# enrollment_no INTEGER PRIMARY KEY,  
temp=# cgpa DECIMAL,  
temp=# area TEXT,  
temp=# branch VARCHAR(30)  
temp=# );  
CREATE TABLE  
temp=# \d student
```

Column	Type	Collation	Nullable	Default
student_name	text			
enrollment_no	integer		not null	
cgpa	numeric			
area	text			

//Alter table student schema

ALTER TABLE student ALTER COLUMN student_name SET NOT NULL;

ALTER TABLE student ALTER COLUMN cgpa SET NOT NULL;

ALTER TABLE student ALTER COLUMN area SET NOT NULL;

ALTER TABLE student ALTER COLUMN branch SET NOT NULL;

```
temp=# ALTER TABLE student ALTER COLUMN student_name SET NOT NULL;
ALTER TABLE
temp=# ALTER TABLE student ALTER COLUMN cgpa SET NOT NULL;
ALTER TABLE
temp=# ALTER TABLE student ALTER COLUMN area SET NOT NULL;
ALTER TABLE
temp=# ALTER TABLE student ALTER COLUMN branch SET NOT NULL;
ALTER TABLE
temp=# \d student;
```

Column	Type	Collation	Nullable
student_name	text		not null
enrollment_no	integer		not null
cgpa	numeric		not null
area	text		not null

//Truncating table

TRUNCATE TABLE student;

```
temp=# TRUNCATE TABLE student;
TRUNCATE TABLE
temp=# select * from student;
```

student_name	enrollment_no	cgpa	area
--------------	---------------	------	------

- b) DML : A data manipulation language (DML) is a computer programming language used for adding (inserting), deleting, and modifying (updating) data in a database. A DML is often a sublanguage of a broader database language such as SQL, with the DML comprising some of the operators in the language.

//Inserting data in table student

```
INSERT INTO student
Values ('Ankit Gupta', 001, 8.11, 'Delhi', 'HUF'),
('Hello World', 002, 8.11, 'Delhi', 'HUF'),
('Hewlett Packard', 003, 7.41, 'Odisha', 'GRE'),
('Apple Windows', 004, 9.5, 'Bangalore', 'BLR');
```

```
temp=# INSERT INTO student
temp=# Values ('Ankit Gupta', 001, 8.11, 'Delhi',
temp=# ('Hello World', 002, 8.11, 'Delhi', 'HUF')
temp=# ('Hewlett Packard', 003, 7.41, 'Odisha', '
temp=# ('Apple Windows', 004, 9.5, 'Bangalore', 'I
```

//Updating data in tale student

```
UPDATE student SET branch='DEL' where enrollment_no=001;
UPDATE student SET branch='DEL' where enrollment_no=002;
```

```
temp=# UPDATE student SET branch='DEL' where enrollment
UPDATE 1
temp=# UPDATE student SET branch='DEL' where enrollment
```

//Deleting data from the table

```
DELETE FROM student WHERE enrollment_no=003;
```

```
temp=# DELETE FROM student WHERE enrollment
DELETE 1
```

- c) DQL : (Data Query Language) These statements are used to Query data from database.

//Getting and printing columns from the table

```
SELECT enrollment_no, student_name,cgpa FROM student;
```

```
temp=# SELECT enrollment_no, student_name, cgpa FROM stud
enrollment_no | student_name | cgpa
-----+-----+-----
          4 | Apple Windows | 9.5
          1 | Ankit Gupta   | 8.11
          2 | Hello World   | 8.11
```

- d) DCL : A data control language is a syntax similar to a computer programming language used to control access to data stored in a database. In particular, it is a component of Structured Query Language. Examples of DCL commands include: GRANT to allow specified users to perform specified tasks.

```
//Creating user (experiment_two)
```

```
CREATE USER experiment_two with encrypted password 'hey_there';
```

```
//Granting access to user (experiment_two)
```

```
GRANT SELECT, UPDATE ON student to experiment_two;
```

```
//Revoking access
```

```
REVOKE SELECT ON student FROM experiment_two;
```

```
temp=# CREATE USER experiment_two with encrypted password
CREATE ROLE
temp=# GRANT SELECT, UPDATE ON student to experiment_two;
GRANT
temp=# REVOKE SELECT ON student FROM experiment_two;
```

- e) Built-in functions : These are the basic built in functions provided by the PostgreSQL tool, which can be further used with the data to get a modified output based on multiple records present in table. Eg, sum() functions allows adding multiple column values together and it generates an output later on.

- 1) Count(*) : count function is used to count the total number of records present in a table. It can also be queried using where and other clauses.

```
//Get the records count where branch = "DEL"
```

```
SELECT COUNT(*) FROM student WHERE branch='DEL';
```

```
temp=# SELECT COUNT(*) FROM student WHERE branch='DEL';
count
-----
```

- 2) MAX(column_name) : It is used to get the maximum value from the table. Like max marks, or max amount or string (maximum lexicographically).

```
//Select the maximum cgpa, where student name starts with 'A'
```

```
SELECT MAX(cgpa) FROM student WHERE student_name LIKE 'A%';
```

- 3) SUM(column_name) : It is used to add the records of a column together based on the clauses and conditions used.

```
//Average of cgpa in the table student
```

```
SELECT DIV((SELECT SUM(cgpa) from student), (SELECT COUNT(*) FROM student));
```

```
temp=#
temp=# SELECT MAX(cgpa) FROM student WHERE student_name LIKE 'A%';
max
-----
9.5
(1 row)

temp=#
temp=# SELECT DIV((SELECT SUM(cgpa) from student), (SELECT COUNT(*) FROM student));
div
-----
```

f) Indexes and Views :

Indexes : An index is a way to efficiently retrieve a relatively small number of rows from a table. It is only useful if the number of rows to be retrieved from a table is relatively small (i.e. the condition for retrieving rows - the WHERE clause - is selective). B-Tree indexes are also useful for avoiding sorting.

Types of Indexes :

- 1) B-Tree
- 2) Hash indexes
- 3) GIN
- 4) GiST

```
//Create index of one or multiple columns  
CREATE INDEX cgpa_index on student(cgpa);
```

```
//Deleting index  
DROP INDEX cgpa_index;
```

Views : A view is named query that provides another way to present data in the database tables. A view is defined based on one or more tables which are known as base tables. When you create a view, you basically create a query and assign it a name, therefore a view is useful for wrapping a commonly used complex query.

```
//Create view for student name and cgpa  
CREATE VIEW student_view AS SELECT student_name, cgpa FROM student;
```

```
//Drop view student_view  
DROP VIEW student_view;
```



```
temp=# CREATE INDEX cgpa_index on student(cgpa);  
CREATE INDEX  
temp=#  
temp=# DROP INDEX cgpa_index;  
DROP INDEX  
temp=#  
temp=# CREATE VIEW student_view AS SELECT student_name, cgpa FROM  
CREATE VIEW  
temp=#
```