1. Create database

create database flores;

use flores;

1. create table

create table signup(

username varchar(50),

pasword varchar(20),

birth\_date date,

phone varchar(12),

gender varchar(1) );

1. insert record

insert into signup (username, pasword, birth\_date, phone, gender)

values

("radha", "radha", "1996-06-25", "8400306959", "F"),

("shyam", "shyam", "1996-05-24", "8400306958", "M");

Delete table

DROP TABLE signup;

1. constrain

create table signup(

username varchar(50) not null unique,

pasword varchar(20) not null,

name varchar(20) not null,

age int not null check (age>=18),

gender varchar(1) not null,

phone varchar(12) unique,

country varchar(50) not null default'India'

);

1. select

select \* from signup; or

select username, phone, age from signup;

select username as Name, phone as "Mobile No." , age from signup;

select name || ' ' || username, phone from signup;

SELECT username as userid from signup; /

SELECT username "userid" from signup;

1. where

select \* from signup

where gender = "F";

select \* from signup

where age < 21;

operator - =, <, >, <=, >=, <> or !=, and, or, in, between, like, is null, not

1. and or not  
   select \* from signup

where not (age < 21 and gender = "M") or country = "india" ;

1. in operator

select username, name, age, gender, country

from signup

where age not in (18,24);

select username, name, age, gender, country

from signup

where country in ("india","UAE");

1. between operator

select username, phone, age, gender, country

from signup

where not age between 20 and 25;

//string wise (a to k alphabet tak k naam)

select username, phone, age, gender, country

from signup

where name between "amit" and "kamal";

//2 date ke beech k date ka data find krna

1. like operatior

% - 0 to many characters

\_ - single character

a% - start with a

%a – end with a

%am% - am anywhere

a%m – start with a and end with m

\_a% - a at second position

\_\_a% - a at third position

select username, phone, age, gender, country

from signup

where username like "r%";

select username, phone, age, gender, country

from signup

where username like "r%" or name like "g%";

//capital small letter

select username, phone, age, gender, country

from signup

where binary username like "R%";

1. regular expression (regexp) operation
2. order by

select username, phone, age, gender, country

from signup

order by name,age desc;

1. distinct

select distinct country

from signup

order by country;

1. is null, is not null

//it does not work with WHERE etc

Working in postgres

select \*

from signup

where name is not null;

1. limit records

select \*

from signup

limit 1;

//offset

select \*

from signup

limit 1,1;

select \*

from signup

limit 1 offset 1

select \* from signup

fetch next 1 row only

1. aggregate function

(count, max, min, sum, avg)

select count(distinct country)

from signup;

select max(age) age, name, country

from signup;

select min(age) age, name, country

from signup;

select sum(age) as "total age", name, country

from signup;

select avg(age) as "avg age", name, country

from signup;

1. update

// if we do not use where then all records will be updated

UPDATE signup

SET pasword = "ram123", age = 21

where username = "ram";

1. rollback ,commit

//undo update insert delete

commit; //iske pahale ke delete nahi krega

rollback;

1. delete

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

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

07/12/2022

[https://www.saralweb.com/careers.html#](https://www.saralweb.com/careers.html)

**Question 1**

Write stored procedures in SQL to implement the purchase and sale transactions.

The purchase transaction inputs are : vendor\_id, date, product\_id, qty, purchase\_price ...]  
// more than one product can be purchased at a time from a vendor

The purchase transaction creates a record of the purchase in appropriate tables.

The sale transaction inputs are : customer\_id, date, product\_id, qty, sale\_price ...]  
// more than one product can be sold at a time to a customer.

The sale transaction creates a record of the sale in appropriate tables.

You can assume a given set of products, customers, and vendors and make sample data on your own to test the transactions.

Submit a snapshot of your database with sample data, and the results of executing sale and purchase transactions.

**Question 2**

Write a stored procedure to generate a stock report for a date range.

select stock\_report (date1 date, date2 date);

This report shows the following columns

product\_id product\_name opening\_balance qty\_in qty\_out qty\_net closing\_balance\_date2

The report calculates opening balances from "beginning of time" to date1, then computes qty\_in and qty\_out using purchases and sales respectively, then calculates qty\_net by qty\_in and qty\_out between date1 and date2 and finally computes the closing balance on date2.

You may need to write multiple sql statements inside the body of the stock\_report stored procedure.

You can use any language for database programming, including PLV8 (embeddeded javascript), PL/SQL or any other embedded language embedded in a SQL server.

create table products (

id integer,

name text,

PRIMARY KEY(id)

);

create table customers (

id integer,

name text,

address1 text,

address2 text,

email\_id text,

phone text,

PRIMARY KEY(id)

);

create table suppliers (

id integer,

name text,

address1 text,

address2 text,

email\_id text,

phone text,

PRIMARY KEY(id)

);

create table sales (

id integer,

customer\_id integer,

date date,

PRIMARY KEY(id),

CONSTRAINT fkey\_customer

FOREIGN KEY(customer\_id)

REFERENCES customers(id)

);

create table sales\_items(

sales\_id integer,

product\_id integer,

qty integer,

sale\_price integer,

CONSTRAINT fkey\_sales

FOREIGN KEY(sales\_id)

REFERENCES sales(id),

CONSTRAINT fkey\_products

FOREIGN KEY(product\_id)

REFERENCES products(id)

);

create table purchases (

id integer,

vendor\_id integer,

date date,

PRIMARY KEY(id),

CONSTRAINT fkey\_vendor

FOREIGN KEY(vendor\_id)

REFERENCES suppliers(id)

);

create table purchase\_items (

purchase\_id integer,

product\_id integer,

qty integer,

purchase\_price integer,

CONSTRAINT fkey\_purchase

FOREIGN KEY(purchase\_id)

REFERENCES purchases(id),

CONSTRAINT fkey\_products

FOREIGN KEY(product\_id)

REFERENCES products(id)

);

insert into products values (101, 'product1');

insert into products values (102, 'product2');

insert into products values (103, 'product3');

insert into products values (104, 'product4');

insert into customers values (201, 'customer1', 'noida', 'delhi', 'customer1@xyz.com', '0123456');

insert into customers values (202, 'customer2', 'faridabad', 'gurugram', 'customer2@xyz.com', '0159753');

insert into customers values (203, 'customer3', 'gurugram', 'delhi', 'customer3@xyz.com', '0147852');

insert into suppliers values (301, 'supplier1', 'ghaziabad', 'noida', 'supplier1@xyz.com', '0128452');

insert into suppliers values (302, 'supplier2', 'sonipat', 'gurugram', 'supplier2@xyz.com', '0179793');

insert into suppliers values (303, 'supplier3', 'surat', 'nagpur', 'supplier3@xyz.com', '0117855');

insert into purchases

values

(2, 302, '2022-01-01'),

(1, 301, '2022-01-01');

insert into purchase\_items

values

(2, 102, '10','100');

-- (1, 101, '5','50');

DELIMITER $$

CREATE PROCEDURE `spPurchasesTransaction` ()

BEGIN

select \*

from purchases

join purchase\_items on purchases.id = purchase\_items.purchase\_id ;

END $$

DELIMITER ;

call spPurchasesTransaction()

insert into sales

values

("1", "201", "2022-01-10")

insert into sales\_items

values ("1", "101", "2", "20")

DELIMITER $$

CREATE PROCEDURE `spSalesTransaction` ()

BEGIN

select \*

from sales

join sales\_items on sales.id = sales\_items.sales\_id ;

END $$

DELIMITER ;

Postgress

Datatype

boolean/bool – true/false/null

char(n)

varchar(n)

text

smallint

int

serial

float(n)

real/float8

date

time

timestamp

timestamptz

interval

array

json

jsonb

uuid

box

line

point

lseg

polygon

inet

macaddr

Constrains

Not null

Unique

Primary key

Check

Foreign key

Show column name

SELECT column\_name, data\_type

FROM information\_schema.columns

WHERE table\_schema = 'demo'

AND table\_name = 'form26as';

Group by

select sum(age), name from signup

group by name

select age from signup

group by age having age >18

The [WHERE](https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-where/) clause allows you to filter rows based on a specified condition. However, the HAVING clause allows you to filter groups of rows according to a specified condition.

In other words, the WHERE clause is applied to rows while the HAVING clause is applied to groups of rows.

union

SELECT \* FROM top\_rated\_films

UNION

SELECT \* FROM most\_popular\_films;

SELECT \* FROM top\_rated\_films

UNION ALL

SELECT \* FROM most\_popular\_films;

Intersect

SELECT select\_list

FROM A

INTERSECT

SELECT select\_list

FROM B;

Except

SELECT select\_list

FROM A

EXCEPT

SELECT select\_list

FROM B;

Cube

CUBE(c1,c2,c3)

GROUPING SETS (

(c1,c2,c3),

(c1,c2),

(c1,c3),

(c2,c3),

(c1),

(c2),

(c3),

()

)

rollup

rollup(c1,c2,c3)

GROUPING SETS (

(c1, c2, c3)

(c1, c2)

(c1)

()

)

Primary key

create table customers(

c\_id serial primary key,

c\_name varchar(255),

c\_email varchar(255)

)

insert into customers(c\_name, c\_email)

values

('ram', 'ram@gmail.com'),

('shyam', 'shyam@gmail.com'),

('sita', 'sita@gmail.com'),

('gita', 'gita@gmail.com')

Foreign key

create table orders(

o\_id serial primary key,

o\_date date,

o\_amount int,

c\_id\_ int references customers(c\_id)

)

insert into orders(o\_date, O\_amount, c\_id\_)

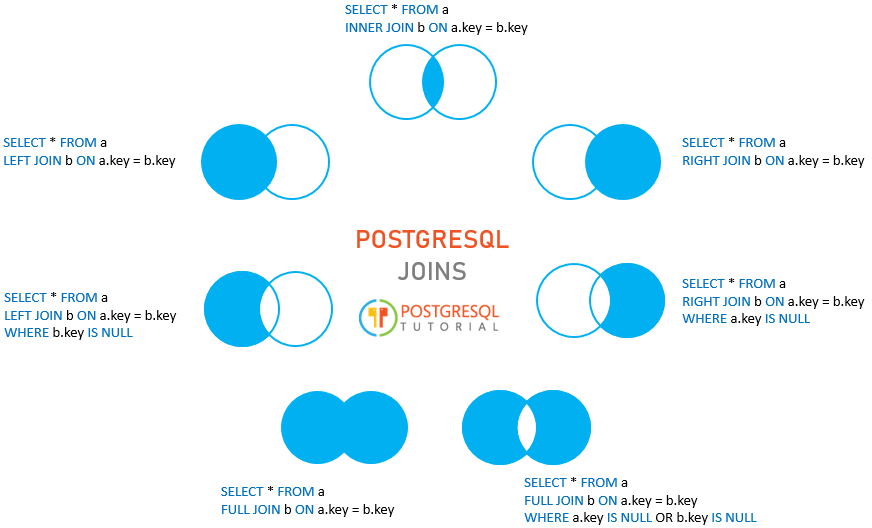
values

('01-01-2023', 100, 1),

('02-01-2023', 50, 1),

('03-01-2023', 80, 3)

[inner join](https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-inner-join/),[left join](https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-left-join/), [right join](https://www.postgresqltutorial.com/postgresql-right-join/), [full join](https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-full-outer-join/), [cross join](https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-cross-join/), [natural join](https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-natural-join/), and a special kind of join called [self-join](https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-self-join/).



Inner join

select \* from customers, orders where customers.c\_id=orders.c\_id\_

select \* from customers

inner join orders

on customers.c\_id=orders.c\_id\_

left join

select \* from customers

left join orders

on customers.c\_id=orders.c\_id\_

right join

select \* from customers

right join orders

on customers.c\_id=orders.c\_id\_

full join

select \* from customers

full outer join orders

on customers.c\_id=orders.c\_id\_

cross join

select \* from customers

cross join orders

self join

natural join

Add column in existing table

Desc <table\_name>

List all foreign key

PostgreSQL operators

PostgreSQL Function

**JSON Functions**

**String Functions**

**Mathematical Functions**

**Date and Time Functions**

**Aggregate Functions**

**Conditional Functions**

**Text Search Functions**

**Window Functions**

Question 1:    
Imagine a table named "Movies" with columns: MovieID, Title, ReleaseDate, GenreID. There's another table "Genres" with columns: GenreID, GenreName. Write a SQL query to fetch the genres that don't have any movies associated with them.  
  
Question 2:   
You are given a table named "Attendance" with columns: StudentID, ClassDate, IsPresent (a boolean where 1 indicates presence and 0 indicates absence). Write a SQL query to identify students who have missed more than 3 consecutive classes.  
  
Question 3:    
Consider a table named "Elections" with columns: CandidateID, VoterID, VoteDate. Write a SQL query to calculate the candidate who received the highest number of votes each month.  
  
Question 4:  
You have a table named "ProductSales" with columns: ProductID, SaleDate, UnitsSold. Write a SQL query to find the top 3 products that have shown the most significant sales growth month-over-month.  
  
Question 5:  
You are provided with a table named "LibraryBooks" with columns: BookID, BorrowerID, BorrowDate, ReturnDate. Write a SQL query to find out which books are currently borrowed and have passed their return date without being returned.  
  
Question 6:  
Consider a table named "OnlineCourses" with columns: CourseID, EnrollmentDate, StudentID, CompletionDate. Write a SQL query to determine the courses which have the highest drop rate (i.e., students enrolling but not completing).  
  
Question 7:  
You have a table named "EmployeeFeedback" with columns: EmployeeID, FeedbackDate, Rating (from 1 to 10). Write a SQL query to identify employees whose rating has been declining for the past 3 consecutive feedbacks.  
  
Question 8:  
There are two tables: "BlogPosts" and "Comments". The "BlogPosts" table has columns: PostID, Title, PostDate, AuthorID. The "Comments" table has columns: CommentID, PostID, CommentDate, Text. Write a SQL query to fetch the blog posts that have not received any comments within a week of their posting.  
  
Question 9:    
You are given a table named "Subscription" with columns: UserID, SubscriptionDate, ExpiryDate. Write a SQL query to count the number of active subscriptions on the first day of each month in the past year.  
  
Question 10:    
Consider a table named "TouristSpots" with columns: SpotID, SpotName, VisitorID, VisitDate. Write a SQL query to find the least visited tourist spots in the last summer.  
  
Question 11:  
There are two tables: "Books" and "Authors". The "Books" table has columns: BookID, BookName, AuthorID, SoldCopies. The "Authors" table has columns: AuthorID, AuthorName. Write a SQL query to find authors whose books, on average, have sold more than 10,000 copies, but have written less than 3 books.  
  
Question 12:  
You have a table named "FlightBookings" with columns: BookingID, FlightDate, PassengerID, Destination. Write a SQL query to determine which destination has seen a steady month-on-month increase in bookings over the last year.