```
[root@localhost ~]# su  postgres
[postgres@localhost root]$ createdb
CREATE DATABASE
[postgres@localhost root]$ psql
could not change directory to "/root"
Welcome to psql 8.1.11, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

 create function plpgsql_call_handler()
returns language_handler as
'$libdir/plpgsql' language c;
CREATE FUNCTION
 create trusted procedural
language plpgsql handler
 plpgsql_call_handler;
NOTICE:  using pg_pltemplate information instead of CREATE LANGUAGE parameters
CREATE LANGUAGE
 create table branch
 (
 bid int primary key,
 brname varchar(20),
 brcity varchar(10)
 );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "branch_pkey" for table "branch"
CREATE TABLE
 select * from branch;
 bid | brname | brcity
-----+--------+--------
(0 rows)

 create table customer
 (
 cno int primary key,
 cname varchar(20),
 caddr varchar(20),
 city varchar(20)
 );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "customer_pkey" for table
"customer"
CREATE TABLE
 select * from customer;
 cno | cname | caddr | city
-----+-------+-------+------
```

```
(0 rows)

 create table loan_application
 (
 lno int primary key,
 l_amt_req money,
 l_amt_appr money,
 ldate date
 );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "loan_application_pkey" for table
"loan_application"
CREATE TABLE
 select * from loan_application;
 lno | l_amt_req | l_amt_appr | ldate
-----+-----------+------------+-------
(0 rows)

 create table ternary
 (
 bid int references branch,
 cno int references customer,
 lno int references loan_application
 );
CREATE TABLE
 select * from ternary;
 bid | cno | lno
-----+-----+-----
(0 rows)
 insert into branch values(1,'Aundh','Pune');
INSERT 0 1
 insert into branch values(2,'Pimpri','Pune');
INSERT 0 1
 insert into branch values(3,'Chinchwad','Pune');
INSERT 0 1
 insert into branch values(4,'Thergaon','Pune');
INSERT 0 1
 select * from branch;
 bid |  brname   | brcity
-----+-----------+--------
   1 | Aundh     | Pune
   2 | Pimpri    | Pune
   3 | Chinchwad | Pune
   4 | Thergaon  | Pune
(4 rows)

 insert into customer values(101,'Omkar','Chinchwad','Pune');
INSERT 0 1
 insert into customer values(102,'Rohit','Moshi','Pune');
```

```
INSERT 0 1
 insert into customer values(103,'Bhagyashri','Kasarwadi','Pune');
INSERT 0 1
 insert into customer values(104,'Lakhan','Worli','Mumbai');
INSERT 0 1
 select * from customer;
 cno |   cname    |   caddr   |  city
-----+------------+-----------+--------
 101 | Omkar      | Chinchwad | Pune
 102 | Rohit      | Moshi     | Pune
 103 | Bhagyashri | Kasarwadi | Pune
 104 | Lakhan     | Worli     | Mumbai
(4 rows)

 insert into loan_application values(1001,500000,500000,'12-june-2012');
ERROR:  column "l_amt_req" is of type money but expression is of type integer
HINT:  You will need to rewrite or cast the expression.
 insert into loan_application values(1001,'500000','500000','12-june-2012');
INSERT 0 1
 insert into loan_application values(1002,'1000000','700000','30-april-2010');
INSERT 0 1
 insert into loan_application values(1003,'700000','500000','1-january-2015');
INSERT 0 1
 insert into loan_application values(1004,'1000000','1000000','24-april-2013');
INSERT 0 1
 select * from loan_application;
 lno |   l_amt_req   |  l_amt_appr   |   ldate
------+---------------+---------------+------------
 1001 |   $500,000.00 |   $500,000.00 | 2012-06-12
 1002 | $1,000,000.00 |   $700,000.00 | 2010-04-30
 1003 |   $700,000.00 |   $500,000.00 | 2015-01-01
 1004 | $1,000,000.00 | $1,000,000.00 | 2013-04-24
(4 rows)

 insert into ternary values(1,101,1001);
INSERT 0 1
 insert into ternary values(2,102,1002);
INSERT 0 1
 insert into ternary values(3,103,1003);
INSERT 0 1
 insert into ternary values(4,104,1004);
INSERT 0 1
 select * from ternary;
 bid | cno | lno
-----+-----+------
   1 | 101 | 1001
   2 | 102 | 1002
   3 | 103 | 1003
```

4 | 104 | 1004
(4 rows)

SLIP 1:-

Create a View:

1. To display names of customers for the 'Pimpri' branch.

```
create view v1 as select cname from customer,branch,ternary where customer.cno=ternary.cno and
branch.bid=ternary.bid and brname='Pimpri';
CREATE VIEW
 select * from v1;
 cname
-------
 Rohit
(1 row)
```

2) 2. To display names of customers who have taken loan from the branch in the same city theylive.

```
create view v2 as  select cname from customer,branch,loan_application,ternary where
customer.cno=ternary.cno and branch.bid=ternary.bid and loan_application.lno=ternary.lno and
city=brcity;
CREATE VIEW
 select * from v2;
   cname
------------
 Omkar
 Rohit
 Bhagyashri
(3 rows)
```

Q.2) Using above database solve following questions:

1. Write a trigger which will execute when you update customer number from customer table. Display message "You can't change existing customer number".

```
 create or replace function t()
 returns trigger as'
 declare
 begin
 if(new.cno=old.cno) then
 raise exception ''Cannot update existing cno'';
 end if;
 return new;
 return old;
 end;'
 language 'plpgsql';
```

CREATE FUNCTION

```
 create trigger tt
 before update on customer
 for each row
 execute procedure t();
```

CREATE TRIGGER

```
 update customer set cno=101 where cno=101;
```

ERROR: Cannot Update existing customer number

2. Write a stored function to accept branch name as an input parameter and display loan information of that branch.

```
 create or replace function printf(name text)
 returns int as'
 declare
 rec record;
 begin
 for rec in select loan_application.lno,l_amt_req,l_amt_appr,ldate from
loan_application,branch,ternary where loan_application.lno=ternary.lno and branch.bid=ternary.bid
and brname=name
 loop
 raise notice ''% % % %'',rec.lno,rec.l_amt_req,rec.l_amt_appr,rec.ldate;
 end loop;
 return 1;
 end;'
 language 'plpgsql';
```

CREATE FUNCTION

```
 select printf('Aundh');
```

NOTICE: 1001 $500,000.00 $500,000.00 2012-06-12

```
 printf
--------
      1
(1 row)
```

Slip2:-

Create a View:

1. To display customer details who have applied for a loan of 5, 00,000.

```
create view v3 as select customer.* from customer,loan_application,ternary where
customer.cno=ternary.cno and loan_application.lno=ternary.lno and l_amt_req='500000';
```

CREATE VIEW

```
 select * from v3;
 cno | cname |   caddr   | city
-----+-------+-----------+------
 101 | Omkar | Chinchwad | Pune
(1 row)
```

2. To display loan details from the ‘Aundh’ branch.

```
create view v4 as select loan_application.* from loan_application,branch,ternary where
branch.bid=ternary.bid and loan_application.lno=ternary.lno and brname='Aundh';
```

CREATE VIEW

```
 select * from v4;
 lno  |  l_amt_req  | l_amt_appr  |   ldate
------+-------------+-------------+------------
 1001 | $500,000.00 | $500,000.00 | 2012-06-12
(1 row)
```

Q.2) Using above database solve following questions:

1. Write a trigger to validate the loan amount approved. It must be less than or equal to loan amount required. Display appropriate message.

```
 create or replace function loan()
 returns trigger as'
 declare
 begin
 if(new.l_amt_appr>new.l_amt_req)then
 raise exception ''Loan Amount approve should be less than loan amt req required'';
 end if;
 end;'
 language 'plpgsql';
```

CREATE FUNCTION

```
 create trigger t3
 before insert on loan_application
 for each row
 execute procedure loan();
```

CREATE TRIGGER

```
 insert into loan_application values(1005,'500000','600000','14-june-2012');
```

ERROR: Loan Amount approve should be less than loan amt req required

Slip 3:-

Create a View:
1. To display the names of customers who required loan>2,00,000.

```
create view v5 as select cname from customer,loan_application,ternary where
customer.cno=ternary.cno and loan_application.lno=ternary.lno and l_amt_req>'200000';

CREATE VIEW
 select * from v5;
  cname
------------
 Omkar
 Rohit
 Bhagyashri
 Lakhan
(4 rows)
```

2. To display the branch wise name of customers.

```
 create view v6 as select cname,brname from customer,branch,ternary where
customer.cno=ternary.cno and branch.bid=ternary.bid group by cname,brname;
CREATE VIEW
 select * from v6;
  cname    |  brname
------------+-----------
 Lakhan     | Thergaon
 Bhagyashri | Chinchwad
 Rohit      | Pimpri
 Omkar      | Aundh
(4 rows)
```

Q.2) Using above database solve following questions:

1. Write a trigger before inserting record of customer in customer table. If the customer number is less than or equal to zero then display the appropriate error message.

```
 create or replace function insert()
 returns trigger as'
 declare
 begin
 if(new.cno<=0)then
 raise exception ''invalid'';
 end if;
 end;'
 language 'plpgsql';
```

```
CREATE FUNCTION
 create trigger t2
 before insert on customer
 for each row
 execute procedure insert();
CREATE TRIGGER

 insert into customer values(0,'sourabh','kalewadi','latur');
ERROR:  invalid
```

2. Write a cursor to display customer details along with their approved loan amount.

```
 create or replace function display()
 returns void as'
 declare
 rec record;
 c1 cursor for select customer.*,l_amt_appr from customer,loan_application,ternary where
customer.cno=ternary.cno and loan_application.lno=ternary.lno;
 begin
 open c1;
 loop
 fetch c1 into rec;
 exit when not found;
 raise notice ''Customer Details % % % % %'',rec.cno,rec.cname,rec.caddr,rec.city,rec.l_amt_appr;
 end loop;
 close c1;
 end;'
 language 'plpgsql';
CREATE FUNCTION

 select display();

NOTICE:  Customer Details 101 Omkar Chinchwad Pune $500,000.00
NOTICE:  Customer Details 102 Rohit Moshi Pune $700,000.00
NOTICE:  Customer Details 103 Bhagyashri Kasarwadi Pune $500,000.00
NOTICE:  Customer Details 104 Lakhan Worli Mumbai $1,000,000.00
 display
---------

(1 row)
```

Slip 4:-

```
create function plpgsql_call_handler()
returns language_handler as
'$libdir/plpgsql' language c;
```

```
create trusted procedural
language plpgsql handler
plpgsql_call_handler;


create table route
(
        rno int primary key,
        src varchar(20),
        dest varchar(20),
        no_st int
);
insert into route values(1,'Hadapsar','Katraj',20);
insert into route values(2,'Hadapsar','Kothrud',25);
insert into route values(3,'Nigadi','Katraj',30);
insert into route values(4,'Kothrud','Katraj',22);

select * from route;

create table bus
(
        bno int primary key,
        cap int not null,
        dept_name varchar(20),
        rno int references route
);
insert into bus values(101,35,'Hadapsar',1);
insert into bus values(102,20,'Nigadi',2);
insert into bus values(103,5,'Kothrud',3);
insert into bus values(104,15,'Hadapsar',4);

select * from bus;

create table driver
(
        dno int primary key,
        dname varchar(20),
        licno int unique,
        addr varchar(20),
        age int,
        salary float
);
insert into driver values(1,'Omkar',1001,'Pune',55,15000);
insert into driver values(2,'Kiran',2001,'Pune',45,12000);
insert into driver values(3,'Ashish',3001,'Pune',38,10000);
insert into driver values(4,'Rohit',4001,'Pune',40,9000);
```

```
select * from driver;

create table bd
(
        bno int references bus on delete cascade,
        dno int references driver on delete cascade,
        ddate date,
        shift char check(shift in('M','E'))
);
insert into bd values(101,1,'20-july-2014','M');
insert into bd values(102,2,'20-july-2014','E');
insert into bd values(103,3,'20-july-2015','M');
insert into bd values(104,4,'20-july-2016','E');

select * from bd;
```

Create View :-

1. To display driver details working in Morning shift.

```
= create view v1 as select driver.* from driver,bd where driver.dno=bd.dno and shift='M';
```

2. To display driver details having salary > 20,000.

```
=create view v2 as select driver.* from driver,bd where driver.dno=bd.dno and salary<20000;
```

Q.2) Using above database solve following questions:

1. Write a trigger before inserting the driver record in driver table, if the age is not between 18 and 35, then display error message 'Invalid input'.

```
= create or replace function driverr()
returns trigger as'
declare
begin
if(new.age not between 18 and 35)then
raise exception ''Invalid input'';
end if;
return new;
end;'
language 'plpgsql';

create trigger t1
before insert on driver
for each row
execute procedure driverr();

insert into driver values(1,'Omkar',1001,'Pune',16,15000);
```

ERROR:Invalid input

2. Write a stored function to display details of buses running on route_no = ‘ ’. (Accept route_no as an input parameter.)

```
= create or replace function bus(n int)
returns int as'
declare
rec record;
begin
for rec in select bus.bno,cap,dept_name from bus,route where route.rno=bus.rno and route.rno=n
loop
raise notice ''% % %'',rec.bno,rec.cap,rec.dept_name;
end loop;
return 1;
end;'
language 'plpgsql';

select bus(1);
101 35 Hadapsar
bus
----
1
```

Slip5:-

Create a View : -

1. To display details of Bus_no 102 along with details of all drivers who have driven that bus.

```
= create view v3 as select bus.*,driver.* from bus,driver,bd where bus.bno=bd.bno and
driver.dno=bd.dno and bus.bno=102;
```

2. To display the route details on which buses of capacity 30 runs.

```
= create view v4 as select route.* from route,bus where route.rno=bus.rno and cap=20;
select route.* from route,bus where route.rno=bus.rno and cap=20;
```

Q.2) Using above database solve following questions:

1. Write a trigger before inserting the driver record in driver table, if the salary is less than or equal to zero, then return the error message ‘Invalid Salary’.

```
= create or replace function sal()
returns trigger as'
declare
begin
```

```
if(new.salary<=0)then
raise exception ''Invalid Salary'';
end if;
return new;
end;'
language 'plpgsql';

create trigger t1
before insert on driver
for each row
execute procedure sal();


insert into driver values(1,'Omkar',1001,'Pune',16,0);
Error:invalid salary
```

2. Write a function using cursor to display all the dates on which a driver has driven a bus (Accept the driver name as an input parameter)

```
create or replace function disp_date(name text)
bus-#  returns void as'
bus'#  declare
bus'#  rec record;
bus'#  c1 cursor for select dduty from driver,db where driver.dno=db.dno
and dname=name;
bus'#  begin
bus'#  open c1;
bus'#  loop
bus'#  fetch c1 into rec;
bus'#  exit when not found;
bus'#  raise notice ''details are % '',rec.dduty;
bus'#  end loop;
bus'#  close c1;
bus'#  end;'
bus-#  language 'plpgsql';
CREATE FUNCTION
bus=# select disp_date('rohit');
NOTICE:  details are 2015-05-20
 disp_date
-----------

(1 row)
```

Slip6 :-

Create a View:

1. To display driver names working in both shifts.

```
= create view v5 as select dname from driver,bd where driver.dno=bd.dno and shift='M'
intersect select dname from driver,bd where driver.dno=bd.dno and shift='E';
```

2. To display route details on which Bus_no 101 is running.
= create view v6 as select route.* from route,bus where route.rno=bus.rno and bno=101;

Q.2) Using above database solve following questions:

1. Write a trigger after deleting the bus record which has capacity < 20. Display the appropriate message.

```
= create or replace function cap()
returns trigger as'
declare
begin
if(old.cap<20) then
raise notice  ''Deleting bus'';
end if;
return old;
end;'
language 'plpgsql';

create trigger t3
after delete on bus
for each row
execute procedure cap();


delete from bus where cap<20;
NOTICE:Deleting Bus

Select * from bus;
```

2. Write a cursor to display details of buses running on route_no = 1.

```
= create or replace function disp()
returns void as'
declare
rec record;
c2 cursor for select * from bus where rno=1;
begin
open c2;
raise notice ''Bus_no Capacity Dept_name'';
loop
fetch c2 into rec;
exit when not found;
raise notice ''% % %'',rec.bno,rec.cap,rec.dept_name;
end loop;
close c2;
```

```
end;'
language 'plpgsql';


select disp();
NOTICE: bus_no capacity dept_name
NOTICE: 101 35 Hadapsar


Slip 7

[root@localhost ~]# su postgres
[postgres@localhost root]$ createdb train
could not change directory to "/root"
CREATE DATABASE
[postgres@localhost root]$ psql train
could not change directory to "/root"
Welcome to psql 8.1.11, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

create function plpgsql_call_handler()
 returns language_handler as
 '$libdir/plpgsql' language c;
CREATE FUNCTION
create trusted procedural
 language plpgsql handler
 plpgsql_call_handler;
NOTICE:  using pg_pltemplate information instead of CREATE LANGUAGE parameters
CREATE LANGUAGE
create table train
(
tno int primary key,
tname varchar(20),
depart_time time,
arrival_time time,
sources varchar(20),
dests varchar(20),
no_of_bogies int,
bogie_capacity int
);
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "train_pkey" for table "train"
CREATE TABLEcreate table pass
(
```

```
  pid int primary key,
  pname varchar(20),
  add varchar(20),
  age int,
  gender varchar(20)
);
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "pass_pkey" for table "pass"
CREATE TABLE
create table ticket
(
tno int references train,
pid int references pass,
ticket_no int,
bogie_no int,
no_of_berths int,
tdate date,
tamt decimal(7,2),
status char check(status in('W','C'))
);
CREATE TABLE

insert into train values

1,'shatabdi express','4:20:00','6:30:00','pune','mumbai',27,3);
INSERT 0 1
insert into train values(2,'rajdhani express','6:00:00','7:45:00','dapodi','lonavala',15,3);
INSERT 0 1
insert into train values(3,'hutatma express','10:10:00','12:00:00','kasarwadi','daund',10,3);
INSERT 0 1
select * from train;

select * from train;
 tno |     tname      | depart_time | arrival_time |  sources  |  dests   | no_of_bogies | bogie_capacity
-----+----------------+-------------+--------------+-----------+----------+--------------+----------------
   1 | shatabdi express | 04:20:00    | 06:30:00     | pune      | mumbai   |         27 |              3
   2 | rajdhani express | 06:00:00    | 07:45:00     | dapodi    | lonavala |         15 |              3
   3 | hutatma express  | 10:10:00    | 12:00:00     | kasarwadi | daund    |         10 |              3
(3 rows)

insert into pass values(101,'omkar','pune',25,'male');
INSERT 0 1
insert into pass values(102,'bhagyashri','kasarwadi',20,'female');
INSERT 0 1
insert into pass values(103,'rohit','mumbai',26,'male');INSERT 0 1

select * from pass;
 pid |   pname    |    add    | age | gender
-----+------------+-----------+-----+--------
```

```
 101 | omkar      | pune      |  25 | male
 102 | bhagyashri | kasarwadi |  20 | female
 103 | rohit      | mumbai    |  26 | male
(3 rows)

insert into ticket values(1,101,11,2001,90,'02-03-2022',12345.67,'W');
INSERT 0 1
insert into ticket values(2,102,12,2002,95,'2022-04-19',32145.50,'C');
INSERT 0 1
insert into ticket values(3,103,13,2003,98,'2022-01-01',51445.40,'W');
INSERT 0 1
select * from ticket;


 tno | pid | ticket_no | bogie_no | no_of_berths |   tdate    |   tamt   | status
-----+-----+-----------+----------+--------------+------------+----------+--------
   1 | 101 |       11 |    2001 |           90 | 2022-02-03 | 12345.67 | W
   2 | 102 |       12 |    2002 |           95 | 2022-04-19 | 32145.50 | C
   3 | 103 |       13 |    2003 |           98 | 2022-01-01 | 51445.40 | W
(3 rows)
```

View 1:

1. To display names of ‘Shatabdi Express’ passengers whose ticket status is waiting on 02-03-2022.

```
create view v1 as select pname from pass,ticket,train where pass.pid=ticket.pid and
train.tno=ticket.tno and tname='shatabdi express' and status='W' and tdate='02-03-2022';
CREATE VIEW
select * from v1; pname
-------
 omkar
(1 row)
```

View 2:

2. To display first three bookings for ‘Rajdhani Express’ on 04-05-2021.

```
 create view v2 as select ticket.* from ticket,train where train.tno=ticket.tno and tdate='04-05-2021'
and tname='rajdhani express' limit 3;
CREATE VIEW
select * from v2;
 tno | pid | ticket_no | bogie_no | no_of_berths | tdate | tamt | status
-----+-----+-----------+----------+--------------+-------+------+--------
(0 rows)
```

1.Write a trigger to restrict the bogie capacity of any train to 25.

```
create or replace function train()
returns trigger as'
declare
begin
if(new.bogie_capacity!=25)then
raise exception ''bogie capacity should be 25..'';
end if;
return new;
end;'
language 'plpgsql';


create trigger t1
before insert on train
for each row
execute procedure train();


insert into train values(1,'shatabdi     express','4:20:00','6:30:00','pune','mumbai',8,3);
ERROR:  bogie capacity should be 25..
```

2. Write a function using cursor to display train wise confirmed bookings on 19-04-2022.

```
create or replace function book()
returns void as'
declare
rec record;
c3 cursor for select tname,status,tdate from train,ticket where train.tno=ticket.tno and tdate ='2022-04-
19'and status='C'group by tname,status,tdate;
begin
open c3;
loop
fetch into rec;
exit when not found;
raise notice''%%%'',rec.tname,rec.status,rec.tdate;
end loop;
close c3;
end;'
language 'plpgsql';

select book();
```

slip 8:

1. To display names of ‘Shatabdi Express’ passengers whose ticket status is confirmed on 02-03-2022.

```
create view v3 as select pname from pass,ticket,train where pass.pid=ticket.pid and
train.tno=ticket.tno and tname='shatabdi express' and status='C' and tdate='02-03-2022';
CREATE VIEW
select * from v3;
 pname
-------
(0 rows)
```

2. To display count of confirmed bookings of ‘Rajdhani Express’ on 01-01-2022.

```
create view v4 as select count(status) from train,ticket where train.tno=ticket.tno and
tname='rajdhani express' and tdate='01-01-2022' and status='C';
CREATE VIEW
select * from v4;
 count
-------
    0
(1 row)
```

1. Write a trigger after inserting the age in passenger table to display the message “Age above5 years will be charged the full fare” if the passenger's age is above 5 years.

```
create or replace function age()
returns trigger as'
declare
begin
if(new.age>5)then
raise exception ''age above 5 will be charged full fare..'';
end if;
return new;
end;'
language 'plpgsql';



create trigger t3
after insert on pass
for each row
execute procedure age();

insert into pass values(107,'omkar','pune',25,'male');
```

ERROR: age above 5 will be charged full fare..

2. Write a stored function to display train wise bookings on 02-05-2020 whose ticket status is waiting.

Slip9 :-

```
[postgres@localhost~]$psql
Welcome to psql 8.1.11, the PostgreSQL interactive terminal.
Type:  \copyright for distribution terms
     \h for help with SQL commands
     \? for help with psql commands
     \g or terminate with semicolon to execute query
     \q to quit
```

Create the following database in 3NF using PostgresSQL.

Q1) Consider the following Project-Employee database, which is managed by a company and store the details of projects assigned to employees.

Project(Pno int, pname varchar(30), ptype varchar(20), duration integer)

Employee(Eno integer, ename varchar(20), qualification char(15), joining_date date)

Relationship:
Project-Employee related with many-to-many relationship, with descriptive attributes as start_date_of_Project, no_of_hours_worked.

Constraints: Primary key, pname should not be null.

```
create function plpgsql_call_handler()
returns language_handler as
'$libdir/plpgsql' language c;

create trusted procedural
language plpgsql handler
plpgsql_call_handler;
```

```
 create table project(pno int primary key,pname varchar(30) not null,ptype varchar(20),duration int);

NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "project_pkey" for table "project"
CREATE TABLE
```

```
 insert into project values(1,'Robotics','AI',10);
INSERT 0 1
 insert into project values(2,'ERP','erp',8);
INSERT 0 1
 insert into project values(3,'C','planguage',15);
INSERT 0 1
 insert into project values(4,'css','html',7);
INSERT 0 1
 select * from project;
pno|  pname   |   ptype   | duration
-----+----------+-----------+----------
  1 | Robotics | AI        |       10
  2 | ERP      | erp       |        8
  3 | C        | planguage |       15
  4 | css      | html      |        7
(4 rows)

Select * from project;

 create table employee(eno int primary key,ename varchar(20) not null,qualification char(15),jdate
date);
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "employee_pkey" for table
"employee"
CREATE TABLE

 insert into employee values(11,'Rohan','mca','2-july-2021');
INSERT 0 1
 insert into employee values(12,'Sunaina','msc','12-june-2022');
INSERT 0 1
 insert into employee values(13,'Anisha','phd','20-april-2020');
INSERT 0 1
 insert into employee values(14,'Yash','BE','5-feb-2018');
INSERT 0 1
 insert into employee values(15,'Neha','Btech','11-oct-2022');
INSERT 0 1
 select * from employee;
eno|  ename  |  qualification  |   jdate
-----+---------+-----------------+------------
  11 | Rohan   | mca             | 2021-07-02
  12 | Sunaina | msc             | 2022-06-12
  13 | Anisha  |phd              | 2020-04-20
  14 | Yash    | BE              | 2018-02-05
  15 | Neha    | Btech           | 2022-10-11
(5 rows)

 create table pe(pno int references project,eno int references employee,sdatedate,noh int);
CREATE TABLE
```

```
 insert into pe values(1,11,'2-august-2021',120);
INSERT 0 1
 insert into pe values(2,12,'12-july-2022',90);
INSERT 0 1
 insert into pe values(3,13,'20-july-2020',110);
INSERT 0 1
 insert into pe values(4,14,'2-march-2018',80);
INSERT 0 1
 insert into pe values(1,15,'12-dec-2022',150);
INSERT 0 1
 select * from pe;
pno | eno |   sdate    | noh
-----+-----+------------+-----
  1 |  11 | 2021-08-02 | 120
  2 |  12 | 2022-07-12 |  90
  3 |  13 | 2020-07-20 | 110
  4 |  14 | 2018-03-02 |  80
  1 |  15 | 2022-12-12 | 150
(5 rows)
```

Create a View:

1. To display the project name, project type, and project start date, sorted by project start date.

```
=  create view v12 as
 select pname,ptype,sdate from project,pe where project.pno=pe.pno order by sdate;
CREATE VIEW
 select * from v12;
pname   |   ptype   |   sdate
----------+-----------+------------
css     | html      | 2018-03-02
 C       | planguage | 2020-07-20
 Robotics | AI       | 2021-08-02
 ERP     | erp       | 2022-07-12
 Robotics | AI       | 2022-12-12
(5 rows)
```

2. To display the details of employees working on ‘Robotics’ project.

```
=  create view v11 as
 select employee.* from employee,project,pe where employee.eno=pe.eno and project.pno=pe.pno
and pname='Robotics';
CREATE VIEW
 select * from v11;
eno | ename|  qualification  |   jdate
-----+-------+-----------------+------------
  11 | Rohan | mca           | 2021-07-02
  15 | Neha  |Btech          | 2022-10-11
(2 rows)
```

Q2) Using above database solve following questions:

1. Write a trigger before inserting the duration into the project table and make sure that the duration is always greater than zero. Display appropriate message

= Trigger:-

```
create or replace function dur_chk()
returns trigger as'
declare
begin
if(new.duration<=0) then
raise exception ''Duration should be always greater than zero'';
end if;
return new;
end;'
language 'plpgsql';
```

CREATE FUNCTION

```
create trigger t1
before insert on project
for each row
execute procedure dur_chk();
```

CREATE TRIGGER

```
insert into project values(5,'system','os',0);
```

ERROR: Duration should be always greater than zero

2. Write function using cursor to accept project name as an input parameter and display names of employees working on that project.

= Cursor:-

```
create or replace function disp_ename(text)
returns void as'
declare
rec record;
c1 cursor for
select ename from employee,project,pe where employee.eno=pe.eno and project.pno=pe.pno and ename=$1;
begin
open c1;
loop
fetch c1 into rec;
exit when not found;
raise notice ''Names of employees % working on project %'',rec.ename,rec.pname;
end loop;
close c1;
end;'
language 'plpgsql';
```

CREATE FUNCTION

```
 select disp_ename('css');
disp_ename
------------
```

(1 row)

Slip10 :-

Create a View:

1. To display employee details and it should be sorted by employee's joining date.

```
=  create view v13 as
 select employee.* from employee,project,pe where employee.eno=pe.eno and project.pno=pe.pno
order by jdate;
CREATE VIEW
 select * from v13;
eno|  ename  |  qualification  |   jdate
-----+---------+-----------------+------------
  14 | Yash    | BE            | 2018-02-05
  13 | Anisha  |phd            | 2020-04-20
  11 | Rohan   | mca           | 2021-07-02
  12 | Sunaina | msc           | 2022-06-12
  15 | Neha    | Btech         | 2022-10-11
(5        rows)
```

2.To display employee and project details where employees worked less than 100 hours.

```
=  create view v14 as
 select employee.*,project.* from employee,project,pe where employee.eno=pe.eno and
project.pno=pe.pno and noh<100;
CREATE VIEW
 select * from v14;
eno|  ename  |  qualification  |   jdate    | pno | pname | ptype | duration
-----+---------+-----------------+------------+-----+-------+-------+----------
  12 | Sunaina | msc           | 2022-06-12 |   2 | ERP   | erp   |       8
  14 | Yash    | BE            | 2018-02-05 |   4 | css   | html  |       7
(2 rows)
```

Q2) Using above database solve following questions:

1. Write a trigger before inserting joining date into employee table, check joining date should be always less than current date. Display appropriate message.

= Trigger:-

```
 create or replace function jdate_chk()
```

```
 returns trigger as'
 declare
 begin
 --current_date gives todays date
 if(new.jdate>=current_date)
 then
 raise exception ''Joining date should be always less than current date'';
 end if;
 return new;
 end;'
 language 'plpgsql';
```

CREATE FUNCTION

```
 create trigger t2
 before insert on employee
 for each row
 execute procedure jdate_chk();
```

CREATE TRIGGER

```
 insert into employee values(16,'Vedika','BE','22-may-2024');
```

ERROR: Joining date should be always less than current date

2. Write a stored function to accept project name as an input parameter and returns the number of employees working on that project. Raise an exception for an invalid project name.

=Cursor:-

```
 create or replace function count_ename(text)
 returns void as'
 declare
 n alias for $1;
 n1 text;
 r int;
 c1 cursor for select pname,count(ename) from project,employee,pe where project.pno=pe.pno and
employee.eno=pe.eno and pname=$1 group by pname;
 begin
 open c1;
 loop
 fetch c1 into n1,r;
 exit when not found;
 raise notice ''% %'',n1,r;
 end loop;
 close c1;
 end;'
 language 'plpgsql';
```

CREATE FUNCTION

```
 select count_ename('C');
```

NOTICE: C 1
count_ename

(1 row)

Slip11 :-

Create a View:

1. To display employee details working on 'ERP' Project.
= create view v16 as
select employee.* from employee,project,pe where employee.eno=pe.eno and project.pno=pe.pno and pname='ERP';
CREATE VIEW
select * from v16;
eno| ename | qualification | jdate
-----+---------+-----------------+------------
12 | Sunaina | msc | 2022-06-12
(1 row)

2. To display employee and project details where employees worked more than 100 hours.
= create view v17 as
select employee.*,project.* from employee,project,pe where employee.eno=pe.eno and project.pno=pe.pno and noh>100;
CREATE VIEW
select * from v17;
eno | ename | qualification | jdate | pno | pname | ptype | duration
-----+--------+-----------------+------------+-----+----------+-----------+----------
11 | Rohan |mca | 2021-07-02 | 1 | Robotics | AI | 10
13 | Anisha | phd | 2020-04-20 | 3 | C | planguage | 15
11 | Rohan |mca | 2021-07-02 | 1 | Robotics | AI | 10
15 | Neha | Btech | 2022-10-11 | 1 | Robotics | AI | 10
(4 rows)

Q2) Using above database solve following questions:

1. Write a trigger after deleting Project record from Project table. Display the message "Project record is being deleted"
2. Write a function to find the number of employees whose date of joining is before 03-10-2022.

Slip12 :-

```
[root@localhost ~]# su - postgres
[postgres@localhost ~]$ createdb
CREATE DATABASE
[postgres@localhost ~]$ psql
Welcome to psql 8.1.11, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit
```

Create the following database in 3NF using PostgresSQL

Q1)Consider the following Student-Teacher database maintained by a college. It also gives information of the subject taught by teachers.
Student(Sno integer,sname varchar(20), sclass varchar(10),saddr varchar(30))
Teacher(Tnointeger,tname varchar(20),qualification char(15),experience integer)

Relation:
Student-Teacher related with many to many relationship with descriptive attribute subject.
Constraints:PrimaryKey,student and teacher name should not be null.

```
 create table student(sno int primary key, sname varchar(20) not null, sclass varchar(10), saddr varchar(30));
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "student_pkey" for table "student"
CREATE TABLE

 insert into student values(11,'Shreya','FYBCA','Pune');
INSERT 0 1
 insert into student values(12,'Neha','FYBCA','Ranchi');
INSERT 0 1
 insert into student values(13,'Aditya','SYBCA','Delhi');
INSERT 0 1
 insert into student values(14,'Sagar','TYBCA','Rajkot');
INSERT 0 1
 select * from student;
sno | sname  | sclass | saddr
-----+--------+--------+--------
  11 | Shreya | FYBCA  | Pune
  12 | Neha   | FYBCA  | Ranchi
  13 | Aditya | SYBCA  | Delhi
  14 | Sagar  | TYBCA  | Rajkot
(4 rows)

 create table teacher(tno int primary key, tname varchar(20) not null, qualification varchar(15), experience int);
```

```
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "teacher_pkey" for table "teacher"
CREATE TABLE

 insert into teacher values(1,'Bharati maam','MCA',10);
INSERT 0 1
 insert into teacher values(2,'Vidya maam','Phd',15);
INSERT 0 1
 insert into teacher values(3,'Deepashree maam','MSC',12);
INSERT 0 1
 insert into teacher values(4,'Sandhya maam','MCA',4);
 select * from teacher;
tno |     tname      | qualification | experience
-----+----------------+---------------+------------
  1 | Bharati maam    | MCA          |        10
  2 | Vidya maam      | Phd          |        15
  3 | Deepashree maam | MSC          |        12
  4 | Sandhya maam    | MCA          |         4
(4 rows)

 create table st(sno int references student,tno int references teacher,subject varchar(15));
CREATE TABLE

 insert into st values(11,1,'dbms');
INSERT 0 1
 insert into st values(12,2,'html');
INSERT 0 1
 insert into st values(13,3,'ds');
INSERT 0 1
 insert into st values(14,4,'cn');
INSERT 0 1
 select * from st;
sno | tno | subject
-----+-----+---------
  11 |   1 | dbms
  12 |   2 | html
  13 |   3 | ds
  14 |   4 | cn
(4 rows)

 create function plpgsql_call_handler()
 returns language_handler as
 '$libdir/plpgsql' language c;
CREATE FUNCTION

 create trusted procedural language plpgsql handler
 plpgsql_call_handler;
NOTICE:  using pg_pltemplate information instead of CREATE LANGUAGE parameters
CREATE LANGUAGE
```

Create a View:
1. To display student names who are taught by most experienced teacher.

```
 create view v1 as
 select sname from student,teacher,st where student.sno=st.sno and teacher.tno=st.tno and
experience=(select max(experience) from teacher);
CREATE VIEW
 select * from  v1;
sname
-------
 Neha
(1 row)
```

2. To display subjects taught by each teacher.

```
 create view v2 as
 select tname,subject from teacher,st where teacher.tno=st.tno group by tname,subject;
CREATE VIEW
 select * from  v2;
tname      | subject
-----------------+---------
 Bharati maam    | dbms
 Vidya maam      | html
Deepashreemaam | ds
 Sandhya maam    | cn
(4 rows)
```

Q2) Using above database solve following questions:

1. Write a trigger before inserting the student record. If the sno is less than or equal to zero, then display the message 'Invalid student number'.

Trigger:-

```
 create or replace function stud()
 returns trigger as'
 declare
 begin
 if(new.sno<=''0'')
 then raise exception ''Invalid student number'';
 end if;
 return new;
 end;'
 language 'plpgsql';
CREATE FUNCTION
 create trigger sno
 before insert on student
```

```
 for each row
 execute procedure stud();
CREATE TRIGGER
  insert into student values(14,'Sagar','TYBCA','Rajkot');
ERROR:  duplicate key violates unique constraint "student_pkey"
  insert into student values(0,'Vaibhav','TYBCA','Ranchi');
ERROR:  Invalid student number
```

Slip13 :-

Create a view:

1. To display teacher details having qualification as ‘Ph.D’.

```
 create view v3 as
 select teacher.* from teacher where qualification='Phd';
CREATE VIEW
 select * from v3;
tno |   tname    | qualification | experience
-----+------------+---------------+------------
   2 | Vidya maam | Phd           |         15
(1 row)
```

2. To display student detail living in ‘Pune’.

```
 create view v4 as
 select student.* from student where saddr='Pune';
CREATE VIEW
 select * from v4;
sno | sname  | sclass | saddr
-----+--------+--------+-------
  11 | Shreya | FYBCA  | Pune
(1 row)
```

Q2) Using above database solve following questions:

1. Write a trigger before inserting experience into a teacher table; experience should be minimum 5 years. Display appropriate message.

Trigger:-

```
 create or replace function t_exp()
 returns trigger as'
 declare
 begin
 if(new.experience<=''5'')
 then raise exception ''Minimum experience should be 5 years'';
```

```
 end if;
 return new;
 end;'
 language 'plpgsql';
CREATE FUNCTION
 create trigger texp
 before insert on teacher
 for each row
 execute procedure t_exp();
CREATE TRIGGER
 insert into teacher values(5,'Manisha maam','MCA',3);
ERROR:  Minimum experience should be 5 years
```

2. Write a cursor to list the details of the teachers who are teaching to a student named '___'. (Accept student name as an input parameter).

Cursor:-

```
 create or replace function disp_teach(text)
 returns void as'
 declare
 name alias for $1;
 t_nm text;
 qual text;
 exp int;
 c1 cursor for
  select tname, qualification, experience from student, teacher, st where student.sno=st.sno and
teacher.tno=st.tno and sname=name;
  begin
 open c1;
 loop
 fetch c1 into t_nm, qual, exp;
 exit when not found;
 raise notice ''Teacher name : %, Qualification : %, Experience : %'',t_nm,qual,exp;
 end loop;
 close c1;
 end;'
  language 'plpgsql';
CREATE FUNCTION
 select disp_teach('Neha');
NOTICE:  Teacher name : Vidya maam, Qualification : Phd, Experience : 15
disp_teach
------------

(1 row)
```

Slip14 :-

Create a view:

1. To display details of teachers having experience > 5years.

```
 create view v5 as
 select teacher.* from teacher where experience>5;
CREATE VIEW
 select * from v5;
tno |     tname      | qualification | experience
-----+----------------+---------------+------------
  1 | Bharati maam    | MCA          |        10
  2 | Vidya maam      | Phd          |        15
  3 | Deepashree maam | MSC          |        12
  4 | Sandhya maam    | MCA          |         4
(4 rows)
```

2. To display details of teachers whose name start with the letter 'S'.

```
 create view v6 as
 select teacher.* from teacher where tname like 'S%';
CREATE VIEW
 select * from v6;
tno |    tname    | qualification | experience
-----+-------------+---------------+------------
  4 | Sandhya maam | MCA          |         4
(1 row)
```

Q2) Using above database solve following questions:

1. Write a trigger before update a student's class from student table. Display appropriate message.

Trigger:-

```
 create or replace function class_update()
 returns trigger as'
 declare
 begin
 if(old.sclass)
 then raise notice ''Class update'';
 end if;
 return new;
 end;'
 language 'plpgsql';
CREATE FUNCTION
 create trigger clsup
 before update on student
 for each row
 execute procedure class_update();
CREATE TRIGGER
```

Slip15 :-

```
create function plpgsql_call_handler()
returns language_handler as
'$libdir/plpgsql'language c;

create trusted procedural
language plpgsql handler
plpgsql_call_handler;

create table student(rno int primary key,
sname char(30),
sclass char(10),
city char(50));

insert into student values(1,'Omkar','fybca','Chinchwad');
insert into student values(2,'Shreyas','fybca','Bhosari');
insert into student values(3,'Aditya','sybca','Balewadi');
insert into student values(4,'Sakshi','sybca','Mumbai');
insert into student values(5,'Poonam','sybca','Pune');
insert into student values(6,'Nilam','tybca','Aundh');
insert into student values(7,'Yadnyesha','tybca','Pimpri');

select * from student;


create table subject(scode varchar(10) primary key,
sub_name varchar(30));

insert into subject values('BCA-101','Web Tech');
insert into subject values('BCA-102','RDBMS');
insert into subject values('BCA-103','Data Structure');
insert into subject values('BCA-104','Computer Newtworks');
insert into subject values('BCA-105','English');

select * from subject;


create table stud_sub(rno int references student(rno),
scode varchar(10) references subject(scode),
marks int);

insert into stud_sub values(1,'BCA-101',92);
```

```
insert into stud_sub values(1,'BCA-102',91);
insert into stud_sub values(2,'BCA-101',85);
insert into stud_sub values(2,'BCA-102',86);
insert into stud_sub values(3,'BCA-101',89);
insert into stud_sub values(3,'BCA-102',94);
insert into stud_sub values(4,'BCA-101',84);
insert into stud_sub values(4,'BCA-102',91);
insert into stud_sub values(5,'BCA-101',92);
insert into stud_sub values(5,'BCA-102',92);
insert into stud_sub values(6,'BCA-101',88);
insert into stud_sub values(6,'BCA-102',86);
insert into stud_sub values(7,'BCA-101',89);
insert into stud_sub values(7,'BCA-102',90);

select * from stud_sub;
```

CREATE VIEW

1)To display names of students class 'FYBCA'.

```
= create view v2 as select sname from student where sclass='fybca';
```

2)to display student name,subject and marks who has scored more than 90 marks.

```
= create view v3 as select sname,sub_name,marks from student a,subject b,stud_sub c where
a.rno=c.rno and b.scode=c.scode and marks>90 order by sname,sclass;
```

Q2).....

1)write a trigger before inserting rollno into student table,display error message if entered rollno less than equal to zero.

```
= create or replace function print_marks()
returns trigger as'
declare
begin
raise notice''Enter rno less than equal to zero..'';
return null;
end;'
language'plpgsql';

create trigger del_stud7
before insert on student
for each row
execute procedure print_marks();
```

2)Write a function using cursor, to calculate total marks of each and display it.

```
= create or replace function print_tmks()
returns int as'
declare
c2 cursor for
select sclass,sname,sum(marks)
from student a,subject b,stud_sub c
 where a.rno=c.rno
and b.scode=c.scode
group by sclass,sname
order by sclass,sname;
sname varchar(30);
sclass varchar(10);
marks int;
begin
open c2;
raise notice''SClass       SName      Total Marks'';
raise notice''-----------------------------------------------'';
loop
fetch c2 into sclass,sname,marks;
exit when not found;
raise notice'' % % %'',sclass,sname,marks;
end loop;
close c2;
return 1;
end;'
language'plpgsql';

select print_tmks();
```

Slip16 :-

CREATE VIEW

1)To display the student names who scored more than 80 marks in'RDBMS'subject

```
=create view v4 as select sname,sub_name,marks from student,subject,stud_sub where
sub_name='RDBMS' and marks>80;
```

2)To display student details of class 'TYBCA'.

```
=create view v5 as select rno,sname,city from student where sclass='tybca';
```

Q2).....

1)write a trigger after deleting a student record from the student table. display the message"student record is being deleted".

```
=create or replace function print_stud()
returns trigger as'
declare
begin
raise notice''student record is being deleted'';
return null;
end;'
language'plpgsql';

create trigger t2
before delete on student
for each row
execute procedure print_stud();

delete from student where rno=8;
```

2)write a stored function to accept student name as an input parameter and display their subject information.

```
=create or replace function print_subinfo(name text)
returns int as'
declare
rec record;
begin
for rec in select subject.scode,sub_name from subject,student,stud_sub where
student.rno=stud_sub.rno and subject.scode=stud_sub.scode and sname=name
loop
raise notice'' % % '',rec.scode,rec.sub_name;
end loop;
return 1;
end;'
language'plpgsql';

select print_subinfo('Aditya');
```

Slip17 :-

CREATE VIEW

1)to display details of students whose name start with the letter'A'.

```
=create view v9 as select rno,sname,sclass,city from student where sname like 'A%';

select * from v9;
```

2)to display details of students who has scored less than 40 marks.

```
=create view v8 as select sname,marks from student,subject,stud_sub where  marks<40;
```

Q2).....

1)write a trigger to ensure that the marks entered for a student with respect to a subject is never<0 and greater than 100.

```
=create or replace function chk_mks()
returns trigger as'
declare
begin
if new.marks<0 or new.marks>100 then
raise notice'' Marks should be never <0 or marks should be never >100'';
end if;
return null;
end;'
language'plpgsql';

create trigger trg_marks
before insert on stud_sub
for each row
execute procedure chk_mks();

insert into stud_sub values(5,'BCA-101',109);
```

2)write a stored function to accept city as an input parameter and display student details.

```
=create or replace function print_city(name text)
returns int as'
declare
rec record;
begin
for rec in select student.rno,sname,sclass from student where city=name
loop
raise notice'' % % % '',rec.rno,rec.sname,rec.sclass;
end loop;
return 1;
end;'
language'plpgsql';

select print_city('Pune');
```

slip 18

```
create function plpgsql_call_handler()
returns language_handler as
'$libdir/plpgsql'language c;

create trusted procedural
language plpgsql handler
plpgsql_call_handler;

create table movie
(
        mname varchar(20) primary key,
        ryear int,
        bud money
);


insert into movie values('Sholey',1975,'10000000');
insert into movie values('KGF',2015,'50000000');
insert into movie values('PK',2016,'5000000');
insert into movie values('KGF2',2022,'100000000');
insert into movie values('YJHD',2012,'4000000');




create table actor
(
        aname varchar(20) primary key,
        city varchar(20)
);

insert into actor values('Amitabh Bacchhan','Mumabi');
insert into actor values('Yash','Pune');
insert into actor values('Amir','Banglore');
insert into actor values('Sanjay','Hyderabad');
insert into actor values('Ranbir','Mumbai');




create table producer
(
        pid int primary key,
        pname varchar(20),insert into ma values('Sholey','Amitabh Bacchhan");
```

```
        paddr varchar(20)
);


insert into producer values(101,'subhash','bombey');
insert into producer values(102,'karan','latur');
insert into producer values(103,'sanjay','mumbai');
insert into producer values(104,'rohit','pune');
insert into producer values(105,'omkar','bombey');


create table ma
(
        mname varchar(20) references movie on delete cascade on update cascade,
        aname varchar(20) references actor on delete cascade on update cascade,
        role varchar(20),
        charges int
);


insert into ma values('Sholey','Amitabh Bacchhan','hero',50000000);
insert into ma values('KGF','Yash','hero',100000000);
insert into ma values('PK','Amir','alien',50000000);
insert into ma values('KGF2','Sanjay','villan',5000000);
insert into ma values('YJHD','Ranbir','hero',100000000);




create table mp
(

        mname varchar(20) references movie on delete cascade on update cascade,
        pid int references producer on delete cascade on update cascade
);


insert into mp values('Sholey',102);
insert into mp values('KGF',101);
insert into mp values('PK',103);
insert into mp values('KGF2',101);
insert into mp values('YJHD',104);
```

Create View :

1. **To display actor names who lives in 'Mumbai'.**

```
select aname from actor where city='Mumbai';

select *from v1;
 aname
--------
 Ranbir
(1 row)
```

2. **To display actors information in each movie.**

```
create view v2 as
select actor.*,movie.mname from actor,movie,ma where actor.aname=ma.aname and
movie.mname=ma.mname;

select *from v2;
    aname       |   city    | mname
------------------+-----------+--------
 Amitabh Bacchhan | Mumabi    | Sholey
 Yash             | Pune      | KGF
 Amir             | Banglore  | PK
 Sanjay           | Hyderabad | KGF2
 Ranbir           | Mumbai    | YJHD
(5 rows)
```

Q.2) Using above database solve following questions: [Total Marks: 20]

1. **Write a trigger before inserting budget into a movie table. Budget should be minimum 60 lakh. Display appropriate message.**

```
create or replace function Chk()
returns trigger as'
declare
begin
if(new.bud<''6000000'') then
raise exception ''Budget should be minimum 60 lakh'';
end if;
return new;
end;'
language 'plpgsql';

create trigger t2
before insert on movie
for each row
execute procedure Chk();
```

insert into movie values('Sholey',1975,'100000');
ERROR: Budget should be minimum 60 lakh

Slip 19 :

Create View

1. To display actor details acted in movie 'Sholey'.

```
create view v3 as
select actor.*,movie.mname from actor,movie,ma where actor.aname=ma.aname and
movie.mname=ma.mname and movie.mname='Sholey';

select *from v3;
     aname       |  city  | mname
-----------------+--------+--------
 Amitabh Bacchhan | Mumabi | Sholey
(1 row)
```

2. To display producer name who have produced more than two movies.

```
create view v4 as
select pname from producer,movie,mp where producer.pid=mp.pid and
movie.mname=mp.mname group by pname having count(*)>1;

select *from v4;
  pname
---------
 subhash
(1 row)
```

Q.2) Using above database solve following questions: [Total Marks: 20]
1. Write a trigger before inserting charges into relationship table. Charges should not be more than 30 lakh. Display appropriate message.

```
create or replace function charge()
returns trigger as'
declare
begin
if(new.charges>3000000) then
raise exception ''Budget should not be greater than 30 lakh'';
end if;
return new;
end;'
language 'plpgsql';
```

create trigger t
before insert on ma
for each row
execute procedure charge();

insert into ma values('YJHD','Ranbir','hero',100000000);
ERROR: Budget should not be greater than 30 lakh

2. Write a stored function to accept actor name as an input parameter and display names of movies in which that actor has acted. Display error message for an invalid actor name.

```
create or replace function actorn(name text)
returns int as'
declare
rec record;
begin
for rec in select movie.mname from movie,actor,ma where movie.mname=ma.mname and
actor.aname=ma.aname and actor.aname=name
loop
raise notice ''%'',rec.mname;
end loop;
return 1;
end;'
language 'plpgsql';

select actorn('Amir');
NOTICE:  PK
 actorn
--------
      1
(1 row)
```

Slip 20:

Create view:

1. To display movie names produced by ‘Mr. Subhash Ghai’.

create view v5 as
select movie.mname from producer,movie,mp where producer.pid=mp.pid and
movie.mname=mp.mname and pname='subhash';

select *from v5;
mname

KGF
KGF2
(2 rows)

2. To display actor names who do not live in Mumbai or Pune city.

```
create view v6 as
select aname from actor where city not in('Mumbai','Pune');

select *from v6;
     aname
------------------
 Amitabh Bacchhan
 Amir
 Sanjay
(3 rows)
```

Q.2) Using above database solve following questions: [Total Marks: 20]

1. Write a trigger before inserting record into movie table; check release_year should not be greater than current year. Display appropriate message.

```
create or replace function year()
returns trigger as'
declare
begin
if(new.ryear>current_date) then
raise exception ''Release year should not be greater then current year'';
end if;
return new;
end;'
language 'plpgsql';


create trigger t9
before insert on movie
for each row
execute procedure year();

insert into movie values('KGF',2025,'50000000');
ERROR:  Release year should not be greater then current year
```

2.Write a cursor using function to list movie-wise charges of ‘Amitabh Bachchan’.

```
create or replace function amitabh()
returns void as'
declare
rec record;
c1 cursor for select mname,charges from movie,actor,ma where movie.mname=ma.mname and
actor.aname=ma.aname and aname='Amitabh Bacchhan';
```

```
begin
loop
fetch c1 into rec;
exit when not found;
raise notice ''%%'',rec.mname,rec.charges;
end loop;
return null;
end;'
language 'plpgsql';
```