## Assignment No: 5

**1)Movie - Actor Database: Movie (m\_name char (25), release\_year integer, budget money) Actor (a\_name varchar (20), role char (20), charges money, a\_address varchar (20)) Producer (producer\_id integer, p\_name char (30), p\_address varchar (20)) Each actor has acted in one or more movies. Each producer has produced many movies and each movie can be produced by more than one producer. Each movie has one or more actors acting in it, in different roles. Constraints: Primary Key, role should be ‘Main’,’Supportive’,’Villan’,’Comedy’ p\_name should not be null. budget,charges > 0**

CREATE TABLE actorr (

a\_name VARCHAR(20) PRIMARY KEY,

role CHAR(20),

charges MONEY,

a\_address VARCHAR(20)

);

CREATE TABLE movie (

m\_name VARCHAR(20) PRIMARY KEY,

release\_year INTEGER,

budget MONEY,

a\_name VARCHAR(20) REFERENCES actorr(a\_name)

);

CREATE TABLE producerr (

pro\_id INTEGER PRIMARY KEY,

p\_name CHAR(30),

p\_address VARCHAR(20)

);

CREATE TABLE movie\_producerr (

m\_name VARCHAR(20) REFERENCES movie(m\_name),

pro\_id INTEGER REFERENCES producerr(pro\_id)

);

INSERT INTO actorr VALUES ('amitabh', 'lead', 500000, 'mumbai');

INSERT INTO actorr VALUES ('prbhas', 'lead', 6800000, 'chennai');

INSERT INTO actorr VALUES ('shradhha', 'supportive', 9800000, 'goa');

INSERT INTO movie VALUES ('sholey', 1987, 900000, 'amitabh');

INSERT INTO movie VALUES ('bahubali', 2016, 9870000, 'prbhas');

INSERT INTO movie VALUES ('stree', 2019, 8700000, 'shradhha')

INSERT INTO producerr VALUES (11, 'shubhash ghai', 'mumbai');

INSERT INTO producerr VALUES (12, 'kartik', 'goa');

INSERT INTO producerr VALUES (13, 'bhat', 'chennai');

INSERT INTO movie\_producerr VALUES ('sholey', 12);

INSERT INTO movie\_producerr VALUES ('bahubali', 11);

INSERT INTO movie\_producerr VALUES ('stree', 13);

INSERT INTO movie\_producerr VALUES ('sholey', 11);

INSERT INTO movie\_producerr VALUES ('bahubali', 12);

select \* from actorr;

a\_name | role | charges | a\_address

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

amitabh | lead | $500,000.00 | mumbai

prbhas | lead | $6,800,000.00 | chennai

shradhha | supportive | $9,800,000.00 | goa

select \* from movie;

m\_name | release\_year | budget | a\_name

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

sholey | 1987 | $900,000.00 | amitabh

bahubali | 2016 | $9,870,000.00 | prbhas

stree | 2019 | $8,700,000.00 | shradhha

(3 rows)

select \* from producerr;

pro\_id | p\_name | p\_address

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

11 | shubhash ghai | mumbai

12 | kartik | goa

13 | bhat | chennai

(3 rows)

select \* from movie\_producerr;

m\_name | pro\_id

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

sholey | 12

bahubali | 11

stree | 13

sholey | 11

bahubali | 12

**1.Write a cursor to pass a\_name as a parameter to a function and return total number of movies in which given actor is acting.**

CREATE OR REPLACE FUNCTION demo(vname VARCHAR)

RETURNS VOID AS $$

DECLARE

actor\_cursor CURSOR FOR

SELECT m.m\_name FROM movie m, actorr a

WHERE a.a\_name = m.a\_name AND a.a\_name = vname;

rec RECORD;

BEGIN

OPEN actor\_cursor;

LOOP

FETCH actor\_cursor INTO rec;

EXIT WHEN NOT FOUND;

RAISE NOTICE '%', rec.m\_name;

END LOOP;

CLOSE actor\_cursor;

END;

$$ LANGUAGE plpgsql;

**select demo('shradhha');**

CREATE FUNCTION

demo

------

(1 row)

psql:commands.sql:70: NOTICE: stree

**2.Write a stored function using cursor to display role wise the names of actors.**

CREATE OR REPLACE FUNCTION demo1()

RETURNS VOID AS $$

DECLARE

role\_cursor CURSOR FOR

SELECT role, a\_name FROM actorr ORDER BY role;

rec RECORD;

BEGIN

OPEN role\_cursor;

LOOP

FETCH role\_cursor INTO rec;

EXIT WHEN NOT FOUND;

RAISE NOTICE '%', rec;

END LOOP;

CLOSE role\_cursor;

END;

$$ LANGUAGE plpgsql;

select demo1();

CREATE FUNCTION

demo1

-------

(1 row)

psql:commands.sql:69: NOTICE: ("lead ",amitabh)

psql:commands.sql:69: NOTICE: ("lead ",prbhas)

psql:commands.sql:69: NOTICE: ("supportive ",shradhha)

**3) Write a cursor to display producer name that produces more than 2 movies in which ‘Amitabh’ is acted. Railway Reservation Database**

CREATE OR REPLACE FUNCTION demo2(pname VARCHAR(20))

RETURNS VOID AS $$

DECLARE

pro\_cursor CURSOR FOR

SELECT p.pro\_id, p.p\_name

FROM producerr p, movie m, movie\_producerr mp

WHERE m.m\_name = mp.m\_name

AND p.pro\_id = mp.pro\_id

AND m.a\_name = pname

GROUP BY p.pro\_id

HAVING COUNT(mp.m\_name) > 2;

rec RECORD;

BEGIN

OPEN pro\_cursor;

LOOP

FETCH pro\_cursor INTO rec;

EXIT WHEN NOT FOUND;

RAISE NOTICE '% %', rec.pro\_id, rec.p\_name;

END LOOP;

CLOSE pro\_cursor;

END;

$$ LANGUAGE plpgsql;

select demo2('amitabh');

CREATE FUNCTION

demo2

-------

(1 row)

**2)Consider the following Entities and their Relationships Railway -Reservation Database:**

**Train (tno int, tname varchar (20), depart\_time time, arrival\_time time, source\_stn char (10),**

**dest\_stn char (10), no\_of\_res\_bogies int ,bogie\_capacity int)**

**Passenger (passenger\_id int, passenger\_name varchar (20), address varchar (30), age int, gender**

**char)**

**Relationship between Train and Passenger is many to many with descriptive attribute ticket.**

**Ticket (train\_no int, passenger\_id int, ticket\_no int,bogie\_no int, no\_of\_berths int, tdate date, ticket\_amt**

**decimal (7,2),status char)**

**Constraints: Primary Key,**

**Status of a berth can be 'W' (waiting) or 'C' (confirmed).**

CREATE TABLE Train (

tno INT PRIMARY KEY,

tname VARCHAR(20),

depart\_time TIME,

arrival\_time TIME,

source\_stn CHAR(10),

dest\_stn CHAR(10),

no\_of\_res\_bogies INT,

bogie\_capacity INT

);

CREATE TABLE Passenger (

passenger\_id INT PRIMARY KEY,

passenger\_name VARCHAR(20),

address VARCHAR(30),

age INT,

gender CHAR

);

CREATE TABLE Ticket (

train\_no INT,

passenger\_id INT,

ticket\_no INT PRIMARY KEY,

bogie\_no INT,

no\_of\_berths INT,

tdate DATE,

ticket\_amt DECIMAL(7,2),

status CHAR(1) CHECK (status IN ('W', 'C'))

);

INSERT INTO Train (tno, tname, depart\_time, arrival\_time, source\_stn, dest\_stn, no\_of\_res\_bogies, bogie\_capacity)

VALUES

(1, 'Express 123', '10:00:00', '18:00:00', 'Mumbai', 'Delhi', 10, 60),

(2, 'Rajdhani', '12:30:00', '22:00:00', 'Chennai', 'Kolkata', 8, 70),

(3, 'Shatabdi', '09:00:00', '15:00:00', 'Delhi', 'Agra', 6, 50),

(4, 'Duronto', '11:00:00', '20:00:00', 'Bangalore', 'Hyderabad', 12, 65),

(5, 'Tejas Express', '13:30:00', '21:00:00', 'Pune', 'Goa', 8, 55);

INSERT INTO Passenger (passenger\_id, passenger\_name, address, age, gender)

VALUES

(101, 'vaishnavi', 'Mumbai', 25, 'F'),

(102, 'meghraj', 'Delhi', 30, 'M'),

(103, 'sunny', 'Chennai', 28, 'M'),

(104, 'pankaj', 'Kolkata', 35, 'M'),

(105, 'aditi', 'Bangalore', 22, 'F');

INSERT INTO Ticket (train\_no, passenger\_id, ticket\_no, bogie\_no, no\_of\_berths, tdate, ticket\_amt, status)

VALUES

(1, 101, 1001, 1, 2, '2023-11-14', 1500.00, 'C'),

(2, 102, 1002, 2, 1, '2023-11-15', 2000.00, 'W'),

(3, 103, 1003, 3, 2, '2023-11-16', 1200.00, 'C'),

(4, 104, 1004, 4, 1, '2023-11-17', 1800.00, 'C'),

(5, 105, 1005, 5, 2, '2023-11-18', 1600.00, 'W');

Select \* from Train;

tno | tname | depart\_time | arrival\_time | source\_stn | dest\_stn | no\_of\_res\_bogies | bogie\_capacity

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

1 | Express 123 | 10:00:00 | 18:00:00 | Mumbai | Delhi | 10 | 60

2 | Rajdhani | 12:30:00 | 22:00:00 | Chennai | Kolkata | 8 | 70

3 | Shatabdi | 09:00:00 | 15:00:00 | Delhi | Agra | 6 | 50

4 | Duronto | 11:00:00 | 20:00:00 | Bangalore | Hyderabad | 12 | 65

5 | Tejas Express | 13:30:00 | 21:00:00 | Pune | Goa | 8 | 55

(5 rows)

Select \* from Passenger;

passenger\_id | passenger\_name | address | age | gender

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

101 | vaishnavi | Mumbai | 25 | F

102 | meghraj | Delhi | 30 | M

103 | sunny | Chennai | 28 | M

104 | pankaj | Kolkata | 35 | M

105 | aditi | Bangalore | 22 | F

Select \* from Ticket;

train\_no | passenger\_id | ticket\_no | bogie\_no | no\_of\_berths | tdate | ticket\_amt | status

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

1 | 101 | 1001 | 1 | 2 | 2023-11-14 | 1500.00 | C

2 | 102 | 1002 | 2 | 1 | 2023-11-15 | 2000.00 | W

3 | 103 | 1003 | 3 | 2 | 2023-11-16 | 1200.00 | C

4 | 104 | 1004 | 4 | 1 | 2023-11-17 | 1800.00 | C

5 | 105 | 1005 | 5 | 2 | 2023-11-18 | 1600.00 | W

(5 rows)

**1) Write a stored function using cursor to accept date and passenger name and display no. of berths**

reserved and ticket amount paid by him/her.

CREATE OR REPLACE FUNCTION berths\_and\_ticket\_amount(p\_date DATE, p\_name VARCHAR)

RETURNS VOID AS $$

DECLARE

ticket\_cursor CURSOR FOR

SELECT t.ticket\_no, t.no\_of\_berths, t.ticket\_amt

FROM ticket t

JOIN passenger p ON p.passenger\_id = t.passenger\_id

WHERE p.passenger\_name = p\_name AND t.tdate = p\_date;

rec RECORD;

total\_berths INT := 0;

total\_amount DECIMAL(7,2) := 0.00;

BEGIN

OPEN ticket\_cursor;

LOOP

FETCH ticket\_cursor INTO rec;

EXIT WHEN NOT FOUND; -- Exit the loop if no more rows

total\_berths := total\_berths + rec.no\_of\_berths;

total\_amount := total\_amount + rec.ticket\_amt;

END LOOP;

CLOSE ticket\_cursor;

RAISE NOTICE 'Total Berths Reserved: %, Total Ticket Amount: ₹%', total\_berths, total\_amount;

END;

$$ LANGUAGE plpgsql;

SELECT berths\_and\_ticket\_amount('2023-11-14', 'vaishnavi');

CREATE FUNCTION

berths\_and\_ticket\_amount

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

(1 row)

psql:commands.sql:94: NOTICE: Total Berths Reserved: 2, Total Ticket Amount: ₹1500.00

**2) Write a stored function using cursors to find the confirmed bookings of all the trains on a particular**

CREATE OR REPLACE FUNCTION confirmed\_bookings(p\_date DATE)

RETURNS VOID AS $$

DECLARE

booking\_cursor CURSOR FOR

SELECT t.tno, t.tname, t.depart\_time, t.arrival\_time, t.source\_stn, t.dest\_stn, t.no\_of\_res\_bogies, t.bogie\_capacity,

tk.ticket\_no, tk.bogie\_no, tk.no\_of\_berths, tk.ticket\_amt

FROM ticket tk

JOIN train t ON tk.train\_no = t.tno

WHERE tk.tdate = p\_date AND tk.status = 'C'; -- Filter for confirmed tickets

rec RECORD;

BEGIN

OPEN booking\_cursor;

LOOP

FETCH booking\_cursor INTO rec;

EXIT WHEN NOT FOUND; -- Exit the loop if no more row

RAISE NOTICE 'Train: %, Departure: %, Arrival: %, Source: %, Destination: %, Bogie No: %, Berths: %, Ticket Amount: ₹%',

rec.tname, rec.depart\_time, rec.arrival\_time, rec.source\_stn, rec.dest\_stn,

rec.bogie\_no, rec.no\_of\_berths, rec.ticket\_amt;

END LOOP;

CLOSE booking\_cursor;

END;

$$ LANGUAGE plpgsql;

SELECT confirmed\_bookings('2023-11-14');

date (Accept date from user).CREATE FUNCTION

confirmed\_bookings

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

(1 row)

psql:commands.sql:90: NOTICE: Train: Express 123, Departure: 10:00:00, Arrival: 18:00:00, Source: Mumbai , Destination: Delhi , Bogie No: 1, Berths: 2, Ticket Amount: ₹1500.00

**3) Write a stored function using cursors to accept a date and find the total number of berths**

CREATE OR REPLACE FUNCTION total\_berths\_reserved(p\_date DATE)

RETURNS VOID AS $$

DECLARE

berth\_cursor CURSOR FOR

SELECT no\_of\_berths

FROM ticket

WHERE tdate = p\_date;

rec RECORD;

total\_berths INT := 0;

BEGIN

OPEN berth\_cursor;

LOOP

FETCH berth\_cursor INTO rec;

EXIT WHEN NOT FOUND; -- Exit the loop if no more rows

total\_berths := total\_berths + rec.no\_of\_berths;

END LOOP;

CLOSE berth\_cursor;

RAISE NOTICE 'Total Number of Berths Reserved: %', total\_berths;

END;

$$ LANGUAGE plpgsql;

SELECT total\_berths\_reserved('2024-11-20');

CREATE FUNCTION

total\_berths\_reserved

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

(1 row)

psql:commands.sql:90: NOTICE: Total Number of Berths Reserved: 2

**SET B**

**1. Bus Driver Database BUS (bus\_no int , capacity int , depot\_name varchar(20)) ROUTE (route\_no int, source char(20), destination char(20),no\_of\_stations int) DRIVER (driver\_no int , driver\_name char(20), license\_no int, address char(20), d\_age int, salary float) The relationships are as follows: BUS\_ROUTE: M-1 BUS\_DRIVER: M-M with descriptive attributes Date of duty allotted and Shift – it can be 1 (Morning) or 2 ( Evening ). Constraints: 1. License\_no is unique. 2. Bus capacity is not null**

create table route(route\_no int primary key,source char(20),destination char(20),no\_of\_stations int);

insert into route values(1,'alandi','pune',8);

insert into route values(2,'pune','mumbai',7);

insert into route values(3,'wagholi','alandi',6);

select \* from route;

route\_no | source | destination | no\_of\_stations

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

1 | alandi | pune | 8

2 | pune | mumbai | 7

3 | wagholi | alandi | 6

(3 rows)

create table bus(bus\_no int primary key,capacity int not null,depot\_name varchar(20), route\_no int references route(route\_no));

insert into bus values(1,50,'alandi',1);

insert into bus values(2,20,'wagholi',3);

insert into bus values(3,40,'pune',2);

select \* from bus;

bus\_no | capacity | depot\_name | route\_no

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

1 | 50 | alandi | 1

2 | 20 | wagholi | 3

3 | 40 | pune | 2

(3 rows)

create table driver(d\_no int primary key,dname char(20),license\_no int unique,addr char(20),d\_age int,salary float);

insert into driver values(101,'ram',1001,'pune',47,45000);

insert into driver values(102,'sham',2002,'alandi',44,40000);

insert into driver values(103,'raju',3003,'wagholi',30,20000);

select \* from driver;

d\_no | dname | license\_no | addr | d\_age | salary

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

101 | ram | 1001 | pune | 47 | 45000

102 | sham | 2002 | alandi | 44 | 40000

103 | raju | 3003 | wagholi | 30 | 20000

create table bus\_driver(bus\_no int references bus(bus\_no),d\_no int references driver(d\_no),date\_of\_duty date,shift varchar(10));

INSERT INTO bus\_driver (bus\_no, d\_no, date\_of\_duty, shift)

VALUES (1, 102, '2024-05-04', '1');

INSERT INTO bus\_driver (bus\_no, d\_no, date\_of\_duty, shift)

VALUES (1, 101, '2024-08-04', '2');

INSERT INTO bus\_driver (bus\_no, d\_no, date\_of\_duty, shift)

VALUES (3, 103, '2024-07-06', '1');

select \* from bus\_driver;

bus\_no | d\_no | date\_of\_duty | shift

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

1 | 102 | 2024-05-04 | 1

1 | 101 | 2024-08-04 | 2

3 | 103 | 2024-07-06 | 1

**1) Write a stored function using cursor to display the details of a driver,(Accept driver name as input prameter).**

create or replace function driver\_details(d\_name varchar(20)) returns void as $$

declare

d\_cursor cursor for

select d\_no,dname,license\_no,addr,d\_age,salary

from driver

where dname = d\_name;

rec record;

begin

open d\_cursor;

loop

fetch d\_cursor into rec;

exit when not found;

raise notice'% % % % % %',rec.d\_no, rec.dname, rec.license\_no, rec.addr,rec.d\_age,rec.salary;

END loop;

close d\_cursor;

END;

$$ language plpgsql;

Select driver\_details(‘raju’);

db=# select driver\_details('raju');

NOTICE:

103 raju 3003 wagholi 30 20000

driver\_details

**2) Write a cursor to display the details of the buses that run on route\_no = 1 and route\_no= 2**

create or replace function route\_details () returns void as $$

declare

R\_cursor cursor for

SELECT b.bus\_no, b.capacity, b.depot\_name, r.route\_no, r.source, r.destination, r.no\_of\_stations

FROM bus b, route r where b.route\_no = r.route\_no

and r.route\_no = 1 and r.route\_no = 2;

rec record;

begin

open R\_cursor;

loop

fetch R\_cursor into rec;

exit when not found;

raise notice'% % % % % % %', rec.bus\_no, rec.capacity, rec.depot\_name, rec.route\_no, rec.source, rec.destination, rec.no\_of\_stations

;

END loop;

close R\_cursor;

END;

$$ language plpgsql;

Select route\_details();

CREATE FUNCTION

demo1

-------

psql:commands.sql:69: NOTICE: ("Bus No ", 1)

psql:commands.sql:69: NOTICE: ("Capacity ", 50)

psql:commands.sql:69: NOTICE: ("Depot Name ", alandi)

psql:commands.sql:69: NOTICE: ("Route No ", 1)

psql:commands.sql:69: NOTICE: ("Source ", alandi)

psql:commands.sql:69: NOTICE: ("Destination ", pune)

psql:commands.sql:69: NOTICE: ("No of Stations ", 8)

psql:commands.sql:69: NOTICE: --------------------------------------------

psql:commands.sql:69: NOTICE: ("Bus No ", 3)

psql:commands.sql:69: NOTICE: ("Capacity ", 40)

psql:commands.sql:69: NOTICE: ("Depot Name ", pune)

psql:commands.sql:69: NOTICE: ("Route No ", 2)

psql:commands.sql:69: NOTICE: ("Source ", pune)

psql:commands.sql:69: NOTICE: ("Destination ", mumbai)

psql:commands.sql:69: NOTICE: ("No of Stations ", 7)

psql:commands.sql:69: NOTICE: --------------------------------------------

**2. Consider the following entities and their relationships.**

**Library(Lno, Lname, Location, Librarian, no\_of\_books)**

**Book(Bid, Bname, Author\_Name, Price, publication)**

**Relation between Library and Book is one to many.**

**Constraint: Primary key, Price should not be null. Create a RDB in 3NF and write PL/SQL**

**procedure for the following:**

**create table library(lid int primary key,**

**lname varchar(10),**

**location varchar(10),**

**librarian varchar(10),**

**no\_of\_book int);**

insert into library values(1, 'Lib1', 'pune', 'sham', 200);

insert into library values(2, 'Lib2', 'mumbai', 'ram', 300);

insert into library values(3, 'Lib3', 'alandi', 'siya', 500);

create table book(bid int primary key,

bname varchar(20),

author varchar(20),

price int not null,

publication varchar(20),

lid int references library(lid));

insert into book values(1, 'chhava', 'abc', 200, 'nirali', 2);

insert into book values(2, 'mritunjay', 'pqr', 300, 'nirali pub', 1);

insert into book values(3, 'maharaj', 'xyz', 600, 'nirali', 3);

select \* from library;

lid | lname | location | librarian | no\_of\_book

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

1 | Lib1 | pune | sham | 200

2 | Lib2 | mumbai | ram | 300

3 | Lib3 | alandi | siya | 500

select \* from book;

bid | bname | author | price | publication | lid

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

1 | chhava | abc | 200 | nirali | 2

2 | mritunjay | pqr | 300 | nirali pub | 1

3 | maharaj | xyz | 600 | nirali | 3

(3 rows)

1) Write a cursor which will display library wise book details. (Use Parameterized Cursor)

CREATE OR REPLACE FUNCTION libraryy(lib\_no int)

RETURNS VOID AS $$

DECLARE

Lib\_cursor cursor for

SELECT B.bid, B.bname

FROM book B, library L

WHERE B.lid = L.lid AND L.lid = lib\_no

ORDER BY B.bname

rec RECORD;

BEGIN

Open Lib\_cursor;

loop

fetch Lib\_cursor into rec;

exit when not found;

RAISE NOTICE '% %', rec.bid, rec.bname;

END LOOP;

END;

$$ LANGUAGE plpgsql;

SELECT libraryy(2);

CREATE FUNCTION

libraryy

----------

(1 row)

psql:commands.sql:44: NOTICE: 1 chhava

**2) Write a cursor which will display publication wise book details.**

CREATE OR REPLACE FUNCTION libraryy(publicationn varchar(20)

RETURNS VOID AS $$

DECLARE

Lib\_cursor cursor for

SELECT B.bid, B.bname

FROM book B, library L

WHERE B.lid = L.lid AND publication = publicationn

ORDER BY B.publication

rec RECORD;

BEGIN

Open Lib\_cursor;

LOOP

Fetch Lib\_cursor into rec;

Exit when not found;

RAISE NOTICE '% %', rec.bid, rec.bname;

END LOOP;

END;

$$ LANGUAGE plpgsql;

select libraryy('nirali');

CREATE FUNCTION

libraryy

----------

(1 row)

psql:commands.sql:43: NOTICE: 1 chhava

psql:commands.sql:43: NOTICE: 3 mahraj

**SET C**

**1. Consider the following entities and their relationships.**

**Employee (emp\_id, emp\_name, address)**

**Investment (inv\_no, inv\_name, inv\_date, inv\_amount)**

**Relation between Employee and Investment is One to Many.**

**Constraint: Primary key, inv\_amount should be > 0.**

**Create a RDB in 3NF and write PL/SQL procedure for the following for the following:**

CREATE TABLE Employee (

emp\_id int PRIMARY KEY,

emp\_name VARCHAR(50) NOT NULL,

address VARCHAR(100)

);

CREATE TABLE Investment (

inv\_no int PRIMARY KEY,

inv\_name VARCHAR(50) NOT NULL,

inv\_date DATE NOT NULL,

inv\_amount int CHECK (inv\_amount > 0),

emp\_id int,

CONSTRAINT fk\_emp FOREIGN KEY (emp\_id) REFERENCES Employee (emp\_id)

);

INSERT INTO Employee VALUES (1, 'vaishnavi', 'pune');

INSERT INTO Employee VALUES (2, 'toshita','wagholi');

INSERT INTO Employee VALUES (3, 'rutuja','alandi');

INSERT INTO Investment VALUES (101, 'Mutual Fund','2024-01-15', 5000, 1);

INSERT INTO Investment VALUES (102, 'Fixed Deposit','2024-02-20', 10000, 1);

INSERT INTO Investment VALUES (103, 'Shares', '2024-03-05', 2000, 2);

select \* from employee;

emp\_id | emp\_name | address

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

1 | vaishnavi | pune

2 | toshita | wagholi

3 | rutuja | alandi

select \* from Investment;

inv\_no | inv\_name | inv\_date | inv\_amount | emp\_id

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

101 | Mutual Fund | 2024-01-15 | 5000 | 1

102 | Fixed Deposit | 2024-02-20 | 10000 | 1

1. Shares | 2024-03-05 | 2000 | 2

1.Write a cursor which will display details of employees invested amount in "Mutual Fund"

create or replace function emp\_invest() returns void as $$

declare

emp\_cursor cursor for

SELECT E.emp\_id, E.emp\_name, E.address

FROM employee E, Investment I

where E.emp\_id = I.emp\_id

and I.inv\_name = 'Mutual Fund';

rec record;

begin

open emp\_cursor;

loop

fetch emp\_cursor into rec;

exit when not found;

raise notice'% % %',rec.emp\_id, rec.emp\_name, rec.address;

end loop;

close emp\_cursor;

end;

$$ language plpgsql;

select emp\_invest();

CREATE FUNCTION

emp\_invest

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

(1 row)

psql:commands.sql:50: NOTICE: 1 vaishnavi pune

**2.Write a cursor which will display date wise investment details.**

create or replace function date\_wiseDetails() returns void as $$

declare

date\_cursor cursor for

select inv\_no,inv\_name,inv\_date,inv\_amount

from investment order by inv\_date;

rec record;

begin

open date\_cursor;

loop

fetch date\_cursor into rec;

exit when not found;

raise notice'% % % %',rec.inv\_no, rec.inv\_name, rec.inv\_date, rec.inv\_amount;

end loop;

close date\_cursor;

end;

$$ language plpgsql;

select date\_wiseDetails();

CREATE FUNCTION

date\_wisedetails

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

(1 row)

psql:commands.sql:48: NOTICE: 101 Mutual Fund 2024-01-15 5000

psql:commands.sql:48: NOTICE: 102 Fixed Deposit 2024-02-20 10000

psql:commands.sql:48: NOTICE: 103 Shares 2024-03-05 2000

**2. Consider the following entities and their relationships.**

**Plan (plan no, plan name, no of free calls, freecalltime, fix amt)**

**Customer (cust no cust name, mobile no)**

**Relation between Plan and Customer is One to Many,**

**Constraint, Primary key, fiz amt should be greater than 0.**

**Create a RDB in 3NF and write procedure for the following**

CREATE TABLE Plan (

plan\_no int PRIMARY KEY,

plan\_name VARCHAR(50) NOT NULL,

no\_of\_free\_calls int,

freecalltime int,

fix\_amt int CHECK (fix\_amt > 0)

);

CREATE TABLE Customer (

cust\_no int PRIMARY KEY,

cust\_name VARCHAR(50) NOT NULL,

mobile\_no VARCHAR(15) UNIQUE NOT NULL,

plan\_no int,

CONSTRAINT fk\_plan FOREIGN KEY (plan\_no) REFERENCES Plan (plan\_no)

);

INSERT INTO Plan VALUES (1, 'Basic Plan', 100, 50, 199);

INSERT INTO Plan VALUES (2, 'Premium Plan', 300, 150, 499);

INSERT INTO Plan VALUES (3, 'Unlimited Plan', 567, 435, 999);

INSERT INTO Customer VALUES (101, 'vaishnavi', '9876543210', 1);

INSERT INTO Customer VALUES (102, 'dipti', '8765432109', 2);

INSERT INTO Customer VALUES (103, 'aditi', '7654321098', 3);

Select \* from Plan;

plan\_no | plan\_name | no\_of\_free\_calls | freecalltime | fix\_amt

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

1 | Basic Plan | 100 | 50 | 199

2 | Premium Plan | 300 | 150 | 499

3 | Unlimited Plan | 567 | 435 | 999

(3 rows)

Select \* from Customer;

cust\_no | cust\_name | mobile\_no | plan\_no

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

101 | vaishnavi | 9876543210 | 1

102 | dipti | 8765432109 | 2

103 | aditi | 7654321098 | 3

**1)Write a cursor which will accept plan number from user and display all the details of the selected plan.**

create or replace function plan\_details(plan\_num int) returns void as $$

declare

plan\_cursor cursor for

select plan\_no,plan\_name,no\_of\_free\_calls,freecalltime,fix\_amt

from Plan where plan\_no = plan\_num ;

rec record;

begin

open plan\_cursor;

loop

fetch plan\_cursor into rec;

exit when not found;

raise notice'% % % % %',rec.plan\_no,rec.plan\_name,rec.no\_of\_free\_calls,rec.freecalltime,rec.fix\_amt;

end loop;

close plan\_cursor;

end;

$$ language plpgsql;

select plan\_details(1);

CREATE FUNCTION

plan\_details

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

(1 row)

psql:commands.sql:51: NOTICE: 1 Basic Plan 100 50 199

**2)Write a cursor which will display customer wise plan details (Use Parameterized Cursor)**

create or replace function customerWise\_details(cust\_num int) returns void as $$

declare

name\_cursor cursor for

select C.cust\_no,C.cust\_name,C.mobile\_no,P.plan\_no,P.plan\_name

from Customer C, Plan P where C.cust\_no = cust\_num ;

rec record;

begin

open name\_cursor;

loop

fetch name\_cursor into rec;

exit when not found;

raise notice'% % % % %',rec.cust\_no,rec.cust\_name,rec.mobile\_no,rec.plan\_no,rec.plan\_name;

end loop;

close name\_cursor;

end;

$$ language plpgsql;

select customerWise\_details(1);

CREATE FUNCTION

customerwise\_details

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