**Assignment No – 6**

**Triggers**

**SET A**

**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)**

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

CREATE TABLE

mydb=# create table Bus(bus\_no int primary key,capacity int not null,depot\_name varchar(20),route\_no int referen

ces route);

CREATE TABLE

mydb=# create table Driver(driver\_no int primary key,driver\_name char(20),license\_no int,address char(20),d\_age

int,salary float);

CREATE TABLE

mydb=# create table Bus\_Driver(bus\_no int references Bus,driver\_no int references Driver,date\_of\_duty Date,shift

int check(shift IN(1,2)));

CREATE TABLE

mydb=# insert into route values(101,'X','Y',6);

INSERT 0 1

mydb=# insert into route values(102,'O','P',8);

INSERT 0 1

mydb=# insert into route values(103,'S','T',5);

INSERT 0 1

mydb=# insert into Bus values(1,35,'depotA',101);

INSERT 0 1

mydb=# insert into Bus values(2,45,'depotB',102);

INSERT 0 1

mydb=# insert into Bus values(3,50,'depotC',103);

INSERT 0 1

mydb=# insert into Driver values(11,'xyz',1234,'pune',35,40000);

INSERT 0 1

mydb=# insert into Driver values(12,'abc',1235,'nashik',28,50000);

INSERT 0 1

mydb=# insert into Driver values(13,'pqr',1236,'mumbai',45,60000);

INSERT 0 1

mydb=# insert into Bus\_Driver values(1,11,'2023-11-16',1);

INSERT 0 1

mydb=# insert into Bus\_Driver values(2,12,'2023-11-18',2);

INSERT 0 1

mydb=# insert into Bus\_Driver values(3,13,'2023-11-17',1);

INSERT 0 1

mydb=# select\* from route;

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

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

101 | X | Y | 6

102 | O | P | 8

103 | S | T | 5

(3 rows)

mydb=# select\* from Bus;

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

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

1 | 35 | depotA | 101

2 | 45 | depotB | 102

3 | 50 | depotC | 103

(3 rows)

mydb=# select\* from Driver;

driver\_no | driver\_name | license\_no | address | d\_age | salary

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

11 | xyz | 1234 | pune | 35 | 40000

12 | abc | 1235 | nashik | 28 | 50000

13 | pqr | 1236 | mumbai | 45 | 60000

(3 rows)

mydb=# select\* from Bus\_Driver;

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

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

1 | 11 | 2023-11-16 | 1

2 | 12 | 2023-11-18 | 2

3 | 13 | 2023-11-17 | 1

(3 rows)

1.Define a trigger after insert or update the record of driver if the age is between 18 and 50 give the message “valid entry” otherwise give appropriate message.

mydb=# CREATE OR REPLACE FUNCTION validate\_driver\_age()

mydb-# RETURNS TRIGGER AS $$

mydb$# BEGIN

mydb$# -- Check if the age is between 18 and 50

mydb$# IF (NEW.d\_age >= 18 AND NEW.d\_age <= 50) THEN

mydb$# RAISE NOTICE 'Valid entry: Driver age % is within the allowed range.', NEW.d\_age;

mydb$# ELSE

mydb$# RAISE NOTICE 'Invalid entry: Driver age % is outside the allowed range (18-50).', NEW.d\_age;

mydb$# END IF;

mydb$# RETURN NEW; -- Allow the operation to proceed

mydb$# END;

mydb$# $$ LANGUAGE plpgsql;

CREATE FUNCTION

mydb=#

mydb=# CREATE TRIGGER driver\_age\_check1

mydb-# before INSERT ON Driver

mydb-# FOR EACH ROW

mydb-# EXECUTE FUNCTION validate\_driver\_age();

CREATE TRIGGER

mydb=#

mydb=# insert into Driver (driver\_no, driver\_name, license\_no, address, d\_age, salary)

mydb-# values(5, 'ert', 1238, 'chakan',55, 45000.0);

NOTICE: Invalid entry: Driver age 55 is outside the allowed range (18-50).

2.Define a trigger after delete the record of bus having capacity < 10. Display the message accordingly.

create or replace Function Bus\_record() returns Trigger as $$

mydb$# declare

mydb$# begin

mydb$# if(old.capacity<10) then

mydb$# RAISE NOTICE 'Deleting the record of bus with capacity less than 10 (Bus No: %)', OLD.bus\_no;

mydb$# end if;

mydb$# return old;

mydb$# end;

mydb$# $$language 'plpgsql';

CREATE FUNCTION

mydb=# create trigger trig\_capacity1

mydb-# after delete on BUS

mydb-# for each row

mydb-# execute function Bus\_record();

CREATE TRIGGER

mydb=# insert into BUS(bus\_no,capacity) values

mydb-# (7, 7),

mydb-# (6, 45),

mydb-# (8, 5);

INSERT 0 3

mydb=# delete from BUS where capacity < 10;

NOTICE: Deleting the record of bus with capacity less than 10 (Bus No: 7)

NOTICE: Deleting the record of bus with capacity less than 10 (Bus No: 8)

DELETE 2

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

**Bill (billno, day, tableno, total)**

**Menu (dish no, dish desc. price)**

create table Bill(bill\_no int primary key,day date,table\_no int,total float check(total>0));

CREATE TABLE

mydb=# create table Menu(dish\_no int primary key,dish\_desc varchar(100),price float check(price>0));

CREATE TABLE

mydb=# create table bill\_menu(bill\_no int references Bill,dish\_no int references Menu,quantity int);

CREATE TABLE

mydb=# insert into Bill values(1,'2024-10-08',3,40);

INSERT 0 1

mydb=# insert into Bill values(2,'2024-10-09',5,30);

INSERT 0 1

mydb=# insert into Bill values(3,'2024-10-07',4,20);

INSERT 0 1

mydb=# insert into Menu values(11,'Noodels',50);

INSERT 0 1

mydb=# insert into Menu values(12,'fried Rice',60);

INSERT 0 1

mydb=# insert into Menu values(13,'Anda roll',40);

INSERT 0 1

mydb=# insert into bill\_menu values(1,11,3);

INSERT 0 1

mydb=# insert into bill\_menu values(2,12,4);

INSERT 0 1

mydb=# insert into bill\_menu values(3,13,5);

INSERT 0 1

mydb=# select\*from Bill;

bill\_no | day | table\_no | total

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

1 | 2024-10-08 | 3 | 40

2 | 2024-10-09 | 5 | 30

3 | 2024-10-07 | 4 | 20

(3 rows)

mydb=# select\*from Menu;

dish\_no | dish\_desc | price

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

11 | Noodels | 50

12 | fried Rice | 60

13 | Anda roll | 40

(3 rows)

mydb=# select\*from bill\_menu;

bill\_no | dish\_no | quantity

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

1 | 11 | 3

2 | 12 | 4

3 | 13 | 5

1. rows)

1.Write a trigger which will fire before insert or update on Bill having total less than or equal to zero. (Raise user defined exception and give appropriate message)

mydb=# create or replace Function bill\_check() returns Trigger as $$

mydb$# declare

mydb$# begin

mydb$# if(new.total<=0) then

mydb$# raise exception 'Bill having total less than or equal to zero';

mydb$# end if;

mydb$# return new;

mydb$# end;

mydb$# $$language 'plpgsql';

CREATE FUNCTION

mydb=# create trigger trig\_bills

mydb-# before insert on bill

mydb-# for each row

mydb-# execute function Bill\_check();

CREATE TRIGGER

mydb=# insert into Bill(bill\_no, day, table\_no, total)

mydb-# values(2, '2024-10-09', 5, -60);

ERROR: Bill having total less than or equal to zero

CONTEXT: PL/pgSQL function bill\_check() line 5 at RAISE

2. Write a trigger which will fire before insert or update on Menu having price less than or equal to zero. (Raise user defined exception and give appropriate message)

mydb=# create or replace Function price\_check() returns Trigger as $$

mydb$# declare

mydb$# begin

mydb$# if(new.price<=0) then

mydb$# raise exception 'menu having price less than or equal to zero';

mydb$# end if;

mydb$# return new;

mydb$# end;

mydb$# $$language 'plpgsql';

CREATE FUNCTION

mydb=#

mydb=# create trigger trig\_price1

mydb-# before insert on Menu

mydb-# for each row

mydb-# execute function price\_check();

CREATE TRIGGER

mydb=#

mydb=# insert into Menu (dish\_no, dish\_desc, price)

mydb-# values (1, 'Noodles', -40);

ERROR: menu having price less than or equal to zero

CONTEXT: PL/pgSQL function price\_check() line 5 at RAISE

**SET B**

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

**Gym (Name, city, charges, scheme)**

**Member (ID, Name, phoneNo, address)**

create table Gym(gym\_name varchar(50) primary key,city varchar(50),charges int,scheme varchar(50));

CREATE TABLE

mydb=# create table Member(id int primary key,m\_name varchar(50),ph\_no varchar(15),add varchar(50),gym\_name varc

har(50)references Gym);

CREATE TABLE

mydb=# insert into Gym values('fitness','pune',500,'gold');

INSERT 0 1

mydb=# insert into Gym values('fit fun','nashik',1500,'silver');

INSERT 0 1

mydb=# insert into Gym values('fit Tribe','mumbai',1000,'basic');

INSERT 0 1

mydb=# insert into Member values(1,'Ravi patil','9876543210','pune','fitness');

INSERT 0 1

mydb=# insert into Member values(2,'sahil rathi','8888970234','nashik','fit fun');

INSERT 0 1

mydb=# insert into Member values(3,'mohit sharma','7743210143','mumbai','fit Tribe');

INSERT 0 1

mydb=# select \* from Gym;

gym\_name | city | charges | scheme

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

fitness | pune | 500 | gold

fit fun | nashik | 1500 | silver

fit Tribe | mumbai | 1000 | basic

(3 rows)

mydb=# select \* from Member;

id | m\_name | ph\_no | add | gym\_name

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

1 | Ravi patil | 9876543210 | pune | fitness

2 | sahil rathi | 8888970234 | nashik | fit fun

3 | mohit sharma | 7743210143 | mumbai | fit Tribe

1. rows)

1.Write a trigger which will fire before insert or update on Member having ID less than 0.(Raise user defined exception and give appropriate message)

mydb=# create or replace Function id\_check() returns Trigger as $$

mydb$# declare

mydb$# begin

mydb$# if(new.id<0) then

mydb$# raise exception 'Member having ID less than 0';

mydb$# end if;

mydb$# return new;

mydb$# end;

mydb$# $$language 'plpgsql';

CREATE FUNCTION

mydb=# create trigger trig\_id

mydb-# before insert on Member

mydb-# for each row

mydb-# execute function id\_check();

CREATE TRIGGER

mydb=# insert into Member(id, m\_name, ph\_no, add, gym\_name)

mydb-# VALUES (-1, 'Rohit patil', '9876543210','pune', 'fitness');

ERROR: Member having ID less than 0

CONTEXT: PL/pgSQL function id\_check() line 5 at RAISE

2.Write a trigger which will fire before insert or update on Gym having charges less than 500. (Raise user defined exception and give appropriate message)

mydb=# create or replace Function charges\_check() returns Trigger as $$

mydb$# declare

mydb$# begin

mydb$# if(new.charges<1000) then

mydb$# raise exception 'Gym having charges less than 500';

mydb$# end if;

mydb$# return new;

mydb$# end;

mydb$# $$language 'plpgsql';

CREATE FUNCTION

mydb=# create trigger trig\_charges

mydb-# before insert on Gym

mydb-# for each row

mydb-# execute function charges\_check();

CREATE TRIGGER

mydb=# insert into Gym(gym\_name, city, charges, scheme)

mydb-# VALUES ('fitness', 'pune', 400, 'gold');

ERROR: Gym having charges less than 500

CONTEXT: PL/pgSQL function charges\_check() line 5 at RAISE

**2. Consider the following entities and their relationships. Wholesaler (wno, wname, address, city)**

**Product (product\_no, product\_name, rate)**

create table wholesaler(w\_no int primary key,w\_name varchar(25),add varchar(50),city varchar(10));

CREATE TABLE

mydb=# create table product(p\_no int primary key,p\_name varchar(20),rate float);

CREATE TABLE

mydb=# create table wholesaler\_product(w\_no int references wholesaler,p\_no int references product,quantity int);

CREATE TABLE

mydb=# insert into wholesaler values(1,'abc','sk chowk','pune');

INSERT 0 1

mydb=# insert into wholesaler values(2,'xyz','ram nagar','nashik');

INSERT 0 1

mydb=# insert into wholesaler values(3,'pqr','thane','mumbai');

INSERT 0 1

mydb=# insert into product values(11,'mobile','30000');

INSERT 0 1

mydb=# insert into product values(12,'laptop','50000');

INSERT 0 1

mydb=# insert into product values(13,'tv','35000');

INSERT 0 1

mydb=# insert into wholesaler\_product values(1,11,2);

INSERT 0 1

mydb=# insert into wholesaler\_product values(2,12,3);

INSERT 0 1

mydb=# insert into wholesaler\_product values(3,13,1);

INSERT 0 1

mydb=# select \* from wholesaler;

w\_no | w\_name | add | city

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

1 | abc | sk chowk | pune

2 | xyz | ram nagar | nashik

3 | pqr | thane | mumbai

(3 rows)

mydb=# select \* from product;

p\_no | p\_name | rate

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

11 | mobile | 30000

12 | laptop | 50000

13 | tv | 35000

(3 rows)

mydb=# select \* from wholesaler\_product;

w\_no | p\_no | quantity

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

1 | 11 | 2

2 | 12 | 3

3 | 13 | 1

(3 rows)

1. Write a trigger which will fire before insert or update on Wholesaler having wno less than or equal to zero (Raise user defined exception and give appropriate message)

mydb=# create or replace Function wno\_check() returns Trigger as $$

mydb$# declare

mydb$# begin

mydb$# if(new.w\_no<=0) then

mydb$# raise exception 'Wholesaler having w\_no less than or equal to zero';

mydb$# end if;

mydb$# return new;

mydb$# end;

mydb$# $$language 'plpgsql';

CREATE FUNCTION

mydb=# create trigger trig\_number

mydb-# before insert on Wholesaler

mydb-# for each row

mydb-# execute function wno\_check();

CREATE TRIGGER

mydb=# insert into Wholesaler(w\_no, w\_name, add, city)

mydb-# values (-1, 'abc', 'sk chowk', 'pune');

ERROR: Wholesaler having w\_no less than or equal to zero

CONTEXT: PL/pgSQL function wno\_check() line 5 at RAISE

2.Write a trigger which will fire before insert or update on product having rate less than or equal to zero (Raise user defined exception and give appropriate message)

mydb=# create or replace Function rate\_check() returns Trigger as $$

mydb$# declare

mydb$# begin

mydb$# if(new.rate<=0) then

mydb$# raise exception 'product having rate less than or equal to zero';

mydb$# end if;

mydb$# return new;

mydb$# end;

mydb$# $$language 'plpgsql';

CREATE FUNCTION

mydb=# create trigger trig\_rate

mydb-# before insert on Product

mydb-# for each row

mydb-# execute function rate\_check();

CREATE TRIGGER

mydb=# insert into Product (p\_no, p\_name, rate)

mydb-# VALUES (2, 'laptop', -50000);

ERROR: product having rate less than or equal to zero

CONTEXT: PL/pgSQL function rate\_check() line 5 at RAISE

**SET C**

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

**Driver (driver\_id, driver name, address)**

**Car (license\_no, model, year)**

create table Driver1(driver\_id int primary key,driver\_name varchar(20),add varchar(20));

CREATE TABLE

insert into Driver1 values(1,'Ragav patil','pune');

INSERT 0 1

mydb=# insert into Driver1 values(2,'sham more','nashik');

INSERT 0 1

mydb=# insert into Driver1 values(3,'aman sharma','moshi');

0 1

create table Car2(license\_no int primary key,model char(20),year int);

CREATE TABLE

insert into Car2 values(1124,'Honda','2024');

INSERT 0 1

mydb=# insert into Car2 values(1123,'Hyundai','2023');

INSERT 0 1

mydb=# insert into Car2 values(1122,'swift','2022');

INSERT 0 1

create table car\_driver(driver\_id int references Driver1,license\_no int references Car2,date date,time va

rchar(10));

CREATE TABLE

mydb=# insert into car\_driver values(1,1124,'2024-11-16','10:30:01');

INSERT 0 1

mydb=# insert into car\_driver values(2,1123,'2024-11-17','05:00:01');

INSERT 0 1

mydb=# insert into car\_driver values(3,1122,'2024-11-18','09:45:25');

INSERT 0 1

select\*from Driver1;

driver\_id | driver\_name | add

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

1 | Ragav patil | pune

2 | sham more | nashik

3 | aman sharma | moshi

(3 rows)

mydb=# select \*from car2;

license\_no | model | year

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

1124 | Honda | 2024

1123 | Hyundai | 2023

1122 | swift | 2022

(3 rows)

mydb=# select \*from car\_driver;

driver\_id | license\_no | date | time

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

1 | 1124 | 2024-11-16 | 10:30:01

2 | 1123 | 2024-11-17 | 05:00:01

3 | 1122 | 2024-11-18 | 09:45:25

1. rows)

1.Write a trigger which will fire before insert or update on Driver. If driver\_id is less than or equal to zero. (Raise user defined exception and give appropriate message)

create or replace function validate\_driver\_id()

mydb-# returns trigger as $$

mydb$# begin

mydb$# if new.driver\_id <= 0 then

mydb$# raise exception 'Driver ID must be greater than zero. Provided: %', new.driver\_id;

mydb$# end if;

mydb$# return new;

mydb$# end;

mydb$# $$ language plpgsql;

CREATE FUNCTION

mydb=# create trigger trigg\_id

mydb-# before insert or update on Driver1

mydb-# for each row

mydb-# execute function validate\_driver\_id();

CREATE TRIGGER

mydb=# insert into Driver1(driver\_id, driver\_name, add) values (-2, 'test', 'kochi');

ERROR: Driver ID must be greater than zero. Provided: -2

CONTEXT: PL/pgSQL function validate\_driver\_id() line 4 at RAISE

2) Write a trigger which will fire before insert or update on year. If year value is more than current year. (Raise user defined exception and give appropriate message)

create or replace function validate\_car\_year()

mydb-# returns trigger as $$

mydb$# begin

mydb$# if new.year > extract(year from current\_date) then

mydb$# raise exception 'Car year cannot be greater than the current year. Provided: %', new.year;

mydb$# end if;

mydb$# return new;

mydb$# end;

mydb$# $$ language plpgsql;

CREATE FUNCTION

mydb=# create trigger trg\_year

mydb-# before insert or update on car2

mydb-# for each row

mydb-# execute function validate\_car\_year();

CREATE TRIGGER

mydb=# insert into car2 (license\_no, model, year) values ('1122', ' Creata', 2025);

ERROR: Car year cannot be greater than the current year. Provided: 2025

CONTEXT: PL/pgSQL function validate\_car\_year() line 4 at RAISE

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

**Game (game\_name, no of players, coach name)**

**Player (pid, pname, address, club\_name)**

mydb=# create table Game(g\_name varchar(30) primary key,no\_of\_players int check(no\_of\_players>0),coach\_name varc

har(20));

CREATE TABLE

mydb=#create table player(p\_id int primary key,p\_name varchar(20),add varchar(15),club\_name varchar(20));

CREATE TABLE

mydb=# create table game\_player(g\_name varchar(30)references Game,p\_id int references player);

CREATE TABLE

mydb=# insert into Game values('football',11,'abc');

INSERT 0 1

mydb=# insert into Game values('kho-kho',9,'xyz');

INSERT 0 1

mydb=# insert into Game values('cricket',11,'pqr');

INSERT 0 1

mydb=# insert into player values(1,'sahil','pune','clubA');

INSERT 0 1

mydb=# insert into player values(2,'Mohit','nashik','clubB');

INSERT 0 1

mydb=# insert into player values(3,'ram','chakan','clubC');

INSERT 0 1

mydb=# insert into game\_player values('football',1);

INSERT 0 1

mydb=# insert into game\_player values('kho-kho',2);

INSERT 0 1

mydb=# insert into game\_player values('cricket',3);

INSERT 0 1

mydb=# select \* from Game;

g\_name | no\_of\_players | coach\_name

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

football | 11 | abc

kho-kho | 9 | xyz

cricket | 11 | pqr

(3 rows)

mydb=# select \* from player;

p\_id | p\_name | add | club\_name

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

1 | sahil | pune | clubA

2 | Mohit | nashik | clubB

3 | ram | chakan | clubC

(3 rows)

mydb=# select \* from game\_player;

g\_name | p\_id

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

football | 1

kho-kho | 2

cricket | 3

(3 rows)

1. Write a trigger before insert or update on Player having pid less than or equal to zero. (Raise user defined exception and give appropriate message)

mydb=# create or replace function validate\_player\_pid()

mydb-# returns trigger as $$

mydb$# begin

mydb$# if new.p\_id <= 0 then

mydb$# raise exception 'Invalid player ID: %. Player ID must be greater than 0.', new.p\_id;

mydb$# end if;

mydb$# return new;

mydb$# end;

mydb$# $$ language plpgsql;

CREATE FUNCTION

mydb=# create trigger trg\_id

mydb-# before insert or update on player

mydb-# for each row

mydb-# execute function validate\_player\_pid();

CREATE TRIGGER

mydb=# insert into player (p\_id, p\_name, add, club\_name) values

mydb-# (-1, 'football', 'Test Address', ' Club');

ERROR: Invalid player ID: -1. Player ID must be greater than 0.

CONTEXT: PL/pgSQL function validate\_player\_pid() line 4 at RAISE

2.Write a trigger which will fire before insert or update on Game having no of players less than or equal to zero. (Raise userdefined exception and give appropriate message)

mydb=# create or replace function validate\_game\_players()

mydb-# returns trigger as $$

mydb$# begin

mydb$# if new.no\_of\_players <= 0 then

mydb$# raise exception 'Invalid number of players for game: %. Number of players must be greater than 0.', new.

g\_name;

mydb$# end if;

mydb$# return new;

mydb$# end;

mydb$# $$ language plpgsql;

CREATE FUNCTION

mydb=# create trigger trg\_validate\_game\_players

mydb-# before insert or update on Game

mydb-# for each row

mydb-# execute function validate\_game\_players();

CREATE TRIGGER

mydb=# insert into Game(g\_name, no\_of\_players, coach\_name) values

mydb-# ('Tennis', 0, 'Test Coach');

ERROR: Invalid number of players for game: Tennis. Number of players must be greater than 0.

CONTEXT: PL/pgSQL function validate\_game\_players() line 4 at RAISE