**Task1**

|  |
| --- |
| **A** |
|  | create function inc(val integer) returns integer |
|  | language plpgsql |
|  | as |
|  | $$ |
|  | BEGIN RETURN val + 1; END; |
|  | $$; |
| B |  |
|  | create function sum(val integer,val2 integer) returns bool |
|  | language plpgsql |
|  | as |
|  | $$ |
|  | BEGIN RETURN val + val2; END; |
|  | $$; |
| C |  |
|  |  |
|  | create function check\_divis(num numeric) returns BOOLEAN |
|  | language plpgsql |
|  | as |
|  | $$ |
|  | BEGIN |
|  | IF ( num%2=0) |
|  |  |
|  | then return true; |
|  | else |
|  | return false; |
|  | END IF; |
|  |  |
|  | END; |
|  | $$; |
|  |  |
|  |  |
|  | -- create function check\_divis(variadic list numeric[]) returns BOOLEAN |
|  | -- language plpgsql |
|  | -- as |
|  | -- $$ |
|  | -- BEGIN |
|  | -- SELECT (list[i]) |
|  | -- FROM generate\_subscripts(list, 1) g(i); |
|  | -- IF ( list[i]%2= 0) |
|  | -- |
|  | -- then return true; |
|  | -- else |
|  | -- return false; |
|  | -- END IF; |
|  | -- |
|  | -- END; |
|  | -- $$; |
|  |  |
| D |  |
|  | create function check\_pass(pass text) returns BOOLEAN |
|  | language plpgsql |
|  | as |
|  | $$ |
|  | BEGIN |
|  | IF (length(pass) > 10) |
|  |  |
|  | then return true; |
|  | else |
|  | return false; |
|  | END IF; |
|  |  |
|  | END; |
|  | $$; |
| E |  |
|  | create or replace function splitting(text varchar(30)) returns record |
|  | language plpgsql |
|  | as |
|  | $$ |
|  | declare text1 record; |
|  | begin |
|  | select split\_part(text, ',', 1) , |
|  | split\_part(text, ',', 2) |
|  | into text1; |
|  | return text1; |
|  | end; |
|  | $$; |
|  |  |
|  |  |
| Task2 |  |
| A |  |
|  | create table movie( |
|  | id serial primary key, |
|  | title varchar(50) not null, |
|  | changed timestamp |
|  | ); |
|  |  |
|  | insert into movie(title) values('Inception'); |
|  | insert into movie(title) values('Intestellar'); |
|  | insert into movie(title) values('Tenet'); |
|  |  |
|  | create or replace function changing() returns trigger as $$ |
|  | begin |
|  | new.changed = now(); |
|  | return new; |
|  | end; |
|  | $$ language plpgsql; |
|  |  |
|  | create trigger movie\_changed before insert or update on movie |
|  | for each row execute procedure changing(); |
|  |  |
|  | select \* from movie where id=1; |
|  |  |
|  | insert into movie(title) values ('Common Russian Movie'); |
|  |  |
|  | update movie |
|  | set title='Some shit' |
|  | where id=4; |
| B |  |
|  | create table person( |
|  | id serial primary key, |
|  | name varchar(20), |
|  | age integer, |
|  | year\_of\_birth integer not null |
|  | ); |
|  | create or replace function age\_calculate() |
|  | returns trigger |
|  | language plpgsql |
|  | as |
|  | $$ |
|  | begin |
|  | new.age = extract(year from current\_date) - new.year\_of\_birth; |
|  | return new; |
|  | end; |
|  | $$; |
|  | create trigger age1 before insert or update on person |
|  | for each row execute procedure age\_calculate(); |
|  |  |
|  |  |
|  | insert into person(name, year\_of\_birth) values ('Kek', 2002); |
|  | insert into person(name, year\_of\_birth) values ('Lol', 1945); |
|  | insert into person(name, year\_of\_birth) values ('Rofl', 1992); |
|  | insert into person(name, year\_of\_birth) values ('Mem', 2002); |
|  |  |
|  |  |
| C |  |
|  | CREATE table foods( |
|  | id integer primary key, |
|  | name varchar(80), |
|  | price integer |
|  | ); |
|  |  |
|  | create or replace FUNCTION total() |
|  | returns trigger |
|  | language plpgsql |
|  | as |
|  |  |
|  | $$ |
|  | BEGIN |
|  | update foods |
|  | set price=price+0.12\*price |
|  | where id = new.id; |
|  | return new; |
|  | end; |
|  | $$; |
|  |  |
|  |  |
|  | create trigger cost after insert on foods |
|  | for each row execute procedure total(); |
|  | insert into foods(id, name,price) values (1, 'wood', 50); |
|  | insert into foods(id,name,price) values (3, 'stone', 25); |
|  |  |
|  |  |
| D |  |
|  | create or replace function reset() returns trigger language plpgsql |
|  | as $$ |
|  | begin |
|  | insert into foods(id,name,price) values(old.id,old.name,old.price); |
|  | return old; |
|  | end; |
|  | $$; |
|  |  |
|  | create trigger back |
|  | after delete |
|  | on foods |
|  | for each row |
|  | execute procedure reset(); |
|  | delete from foods where id=2; |
|  | select \* from foods; |
|  |  |
|  |  |
|  |  |
|  |  |
| E |  |
|  | --1e |
|  | create trigger func |
|  | after insert |
|  | on movie |
|  | for each row |
|  | execute function splitting; |
|  |  |
|  | create trigger paswalid |
|  | after insert |
|  | on account |
|  | execute function check\_pass(); |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Task3 |  |
|  | -- The function must return a value but in Stored Procedure it is optional. Even a procedure can return zero or n values. |
|  | -- |
|  | -- Functions can have only input parameters for it whereas Procedures can have input or output parameters. |
|  | -- |
|  | -- Functions can be called from Procedure whereas Procedures cannot be called from a Function. |
|  |  |
| Task4 |  |
| A |  |
|  | Create table employee( |
|  | id serial primary key, |
|  | name varchar(120), |
|  | date\_of\_birth date, |
|  | age int, |
|  | salary int, |
|  | workexperience int, |
|  | discount int); |
|  |  |
|  |  |
|  |  |
|  |  |
|  | CREATE or replace procedure salary() |
|  | language plpgsql |
|  | as |
|  | $$ |
|  | Begin |
|  | update employee |
|  | set salary = ( workexperience/2)\*0.1\*salary+salary, |
|  | discount = ( workexperience/2)\*0.1\*employee.discount + employee.discount, |
|  | discount = ( workexperience/5)\*0.01 \* employee.discount + employee.discount; |
|  | commit; |
|  | end; |
|  | $$; |
|  |  |
|  |  |
|  |  |
| B |  |
|  | create or replace procedure sal() |
|  | language plpgsql |
|  | as |
|  | $$ |
|  | BEGIN |
|  | update employee |
|  | set salary = salary\*1.15 |
|  | where age > 40; |
|  | update employee |
|  | set salary = salary\*1.15\*( workexperience/8); |
|  | update employee |
|  | set discount = 20 where workexperience > 8; |
|  | commit; |
|  | end; |
|  | $$; |
|  |  |
|  |  |
| Task5 |  |
|  | create table members( |
|  | memid integer, |
|  | surname varchar(200), |
|  | firstname varchar(200), |
|  | address varchar(300), |
|  | zipcode integer, |
|  | telephone varchar(20), |
|  | recommendedby integer, |
|  | joindate timestamp |
|  | ); |
|  | create table bookings( |
|  | facid integer, |
|  | memid integer, |
|  | starttime timestamp, |
|  | slots integer |
|  | ); |
|  | create table facilities( |
|  | facid integer, |
|  | name varchar(200), |
|  | membercost numeric, |
|  | guestcost numeric, |
|  | initialoutlay numeric, |
|  | monthlymaintenance numeric |
|  | ); |
|  | with recursive recom(recommender, member) as ( |
|  | select recommendedby, memid |
|  | from members |
|  | union all |
|  | select members.recommendedby, recom.member |
|  | from recom |
|  | inner join members |
|  | on members.memid = recom.recommender |
|  | ) |
|  | select recom.member member, recom.recommender, members.firstname, members.surname |
|  | from recom |
|  | inner join members |
|  | on recom.recommender = members.memid |
|  | where recom.member = 22 or recom.member = 12 |
|  | order by recom.member asc, recom.recommender desc |