Lab 8

first

```
create or replace function qw(value integer)
      returns integer as $$
      begin
        return value+1;
      end; $$
      language plpgsql;
select qw(19);
--2
create or replace function summa(val1 numeric,val2 numeric)
      returns numeric as $$
      begin
         return val1 + val2;
      end; $$
      language plpgsql;
select summa(10,20);
--3
create or replace function is_divisible(value integer)
      returns bool as $$
      begin
          if value%2=0 then
              return true;
          else
              return false;
      end if;
      end; $$
      language plpgsql;
select is_divisible(15);
--4
create or replace function is_valid(password varchar)
      returns bool as $$
      begin
         if length(password)>0 then
            return true;
         else
            return false;
      end if;
      end; $$
      language plpgsql;
drop function is_valid(password varchar);
select is_valid('');
```

Second task

```
--a
drop table customers;
create table customers(
    id serial primary key,
    name varchar(20) not null,
    city varchar(20),
    phone numeric,
    last_updated timestamp
);

insert into customers(name, city, phone) values('Bob', 'Almaty', 899);
insert into customers(name, city, phone) values('Nick', 'London', 567);
insert into customers(name, city, phone) values('Kana', 'Rio', 976);

create or replace function customer_change() returns trigger as $$
begin
    new.last_updated = now();
    return new;
end;
$$ language plpgsql;

create trigger customers_timestamp before insert or update on customers
    for each row execute procedure customer_change();

select * from customers where id=1;

insert into customers(name, city, phone) values('Rick', 'Astana', 879);

update customers
set city='New ---- Orleans'
where id=1;

--b
create table people(
    id serial primary key,
    name varchar(20),
    birth_year integer not null ,
    age integer
);

 select extract(year from current_date);

create or replace function age_calculate()
```

```
    returns trigger
    language plpgsql
    as
$$
    begin
        new.age = extract(year from current_date) - new.birth_year;
        return new;
    end;
$$;

drop trigger age_from_year on people;
create trigger age_from_year before insert or update on people
    for each row execute procedure age_calculate();

insert into people(name, birth_year) values ('Sam', 2002);
insert into people(name, birth_year) values ('Bob', 2010);
insert into people(name, birth_year) values ('Alice', 2020);

update people set birth_year=2002 where id=2;

select *
from people;

-- c

create table product(
    id serial primary key,
    name varchar(20),
    price numeric,
    total_price numeric
);

insert into product (name, price)
values ('iPhone 13', 699.9);

create or replace function add_tax()
    returns trigger
    language plpgsql
    as
$$
    begin
        new.total_price = new.price + (new.price * 0.12);
        return new;
    end;
$$;

create trigger total_price_tax before insert or update on product
    for each row execute procedure add_tax();
```

```sql
insert into product(name, price) values ('Samsung S22', 1000);
insert into product(name, price) values ('Lenovo 11', 599.0);

update product set price=1299.9 where id=1;

select *
from product;

-- d

create function not_delete()
    returns trigger
    language plpgsql
    as
$$
    begin
    raise exception 'Sorry, mate. Cannot delete that!';
    end;
$$;

create trigger undo_delete before delete on product
    for each row execute procedure not_delete();

delete
from product
where id=1;
```

Task 4

```sql
create table employee(
    id serial primary key,
    name varchar(30),
    date_of_birth date,
    age integer,
    salary integer,
    work_experience integer,
    discount integer
);

insert into employee(name, date_of_birth, age, salary, work_experience,
discount)
values ('Bob', '1990-01-02' , 30, 2500, 3, 0);

select *
from employee;
```

```
-- a
create procedure a()
    language plpgsql
    as
$$
    begin
        update employee set salary = salary + (salary*0.1) where
work_experience / 2 > 0;
        update employee set discount = discount + salary*0.1 where
work_experience / 2 > 0;
        update employee set discount = discount + salary*0.01 where
work_experience  > 5;
    end;
$$;

-- b
create procedure b()
    language plpgsql
    as
$$
    begin
        update employee set salary = salary + (salary*0.15) where age >
40;
        update employee set salary = salary + (salary*0.15) where
work_experience > 8;
        update employee set discount = discount + (discount*0.20) where
work_experience > 8;
    end;
$$;

call a();
call b();
```