```
-- Lab 8
-- 1

-- a
create function inc(a int) returns int as
    $$
    begin
        return a + 1;
    end;
    $$
language plpgsql;

select inc(1);

-- b
create function summ(a int, b int) returns int as
    $$
    begin
        return a + b;
    end;
    $$
language plpgsql;

select summ(4, 3);

-- c
create or replace function divisible(a int) returns bool as
    $$
    begin
        if a % 2 = 0 then
            return true;
        end if;
        if a % 2 = 1 then
            return false;
        end if;
    end;
    $$
language plpgsql;

select divisible(3);

-- d
create function valid_password(s varchar) returns varchar as
    $$
    begin
        if s = 'qwerty123' then
            return 'Correct';
        end if;
        if s != 'qwerty123' then
            return 'Invalid password';
        end if;
    end;
    $$
language plpgsql;

select valid_password('qwerty123');

-- e
create or replace function two_ouput(a int, out o1 int, out o2 int) as
    $$
    begin
         o1 = a + 1;
         o2 = a - 1;
    end;
```

```
    $$
language plpgsql;

select two_ouput(1);



create table Employee(
    id integer primary key,
    name varchar,
    date_of_birth date,
    age integer,
    salary integer,
    workexperience integer,
    discount integer
);

create table birth(
    date_of_birth date
);

create table Age_count(
    age interval
);

create table price(
    prc numeric
);

create table price_tax(
    prc_tax numeric
);

create table input_pass(
    password varchar
);

create table ok(
    is_password varchar
);

create table input_a(
    a int
);

create table two_output(
    a record
);

create table check_low_salary
(
    date timestamp
);

--2
-- a

create or replace function trig_func()
    returns trigger
    language plpgsql
    as
    $$
    begin
        if new.salary < 1000 then
```

```
            insert into check_low_salary values (current_timestamp);
        end if;
        return new;
    end;
    $$;

create trigger trig
    before insert or update
    on Employee
    for each row
    execute procedure trig_func();

insert into employee values (1, 'Shinji', '1990-10-10', 31, 800, 10, 10);

-- b
create or replace function trig_age()
    returns trigger
    language plpgsql
    as
    $$
    begin
        insert into Age_count values (age(new.date_of_birth));
        return new;
    end;
    $$;

create trigger trig_age
    after insert or update
    on birth
    for each row
    execute procedure trig_age();

insert into birth values ('1999-06-01');

-- c
create or replace function trig_tax()
    returns trigger
    language plpgsql
    as
    $$
    begin
        insert into price_tax values (new.prc + (new.prc * 0.12));
        return new;
    end;
    $$;

create trigger trig_tax
    after insert or update
    on price
    for each row
    execute procedure trig_tax();

insert into price values (200);


-- d
create or replace function trig_del()
    returns trigger
    language plpgsql
    as
    $$
    begin
        raise exception 'You cannot delete';
    end
```

```
    $$;

create trigger trig_del
    before delete
    on price
    for each row
    execute procedure trig_del();

delete from price
where prc = 10;

-- e
create or replace function trig_pass()
    returns trigger
    language plpgsql
    as
    $$
    begin
        insert into ok values (valid_password(new.password));
        return new;
    end
    $$;

create trigger trig_pass
    after insert
    on input_pass
    for each row
    execute procedure trig_pass();

drop trigger trig_pass on input_pass;

insert into input_pass values ('qwerty1234');
insert into input_pass values ('qwerty123');


create or replace function trig_two_output()
    returns trigger
    language plpgsql
    as
    $$
    begin
        insert into two_output values (two_ouput(new.a));
        return new;
    end
    $$;

create trigger trig_two_output
    after insert
    on input_a
    for each row
    execute procedure trig_two_output();

drop trigger trig_two_output on input_a;

insert into input_a values (1);

-- 3
-- Difference between procedure and function is that, in procedure we can
work with transactions, while in the function we can't.
-- Also, in a function, it is mandatory to use the RETURNS and RETURN
arguments, whereas in a procedure is not necessary.

-- 4
-- a
```

```
create procedure salary_inc()
as
    $$
    begin
         update Employee set salary = (salary *(((workexperience / 2) * 10) +
100)) / 100;
         --update employee set discount = 10;
         update employee set discount = discount + workexperience / 5;
    end;
    $$
language plpgsql;

call salary_inc();

--b

create procedure salary_inc2()
as
    $$
    begin
        update employee set salary = salary * 1.15
           where age >= 40;
        update employee set salary = salary * 1.15, discount = 20
            where workexperience >= 8;
    end;
    $$
language plpgsql;

call salary_inc2();

insert into employee values (2, 'Asuka', '1980-09-10', 41, 1000, 10, 0);
insert into employee values (3, 'Misato', '1970-06-20', 51, 1200, 15, 0);
```