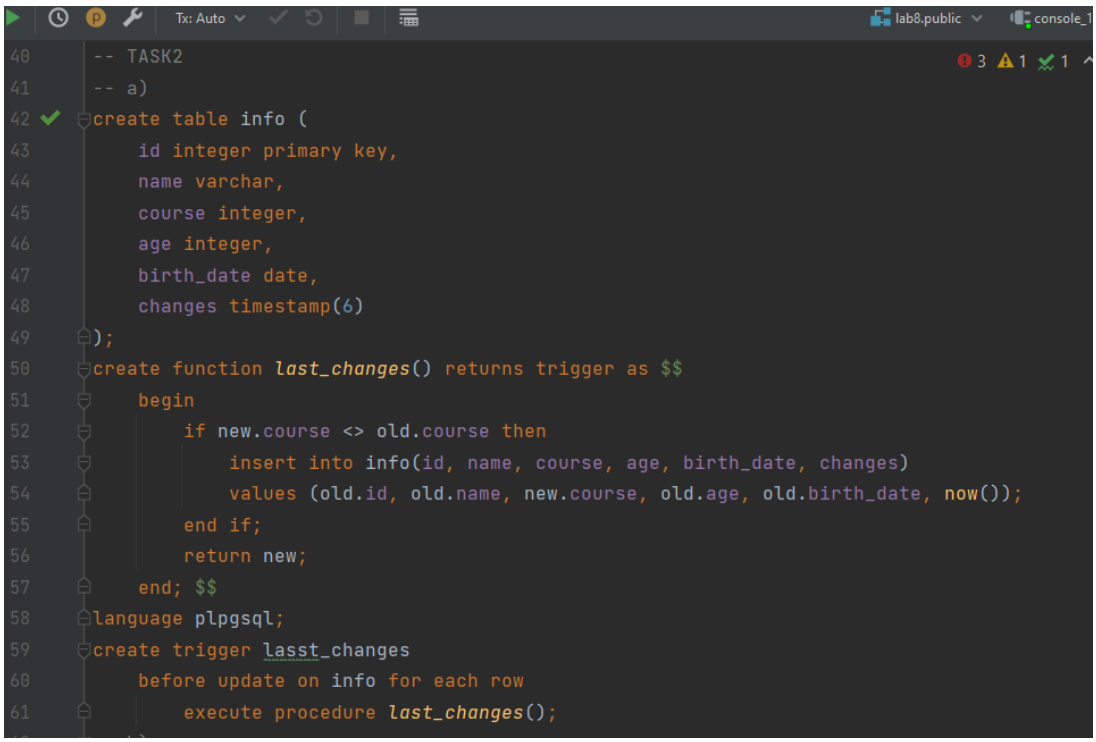
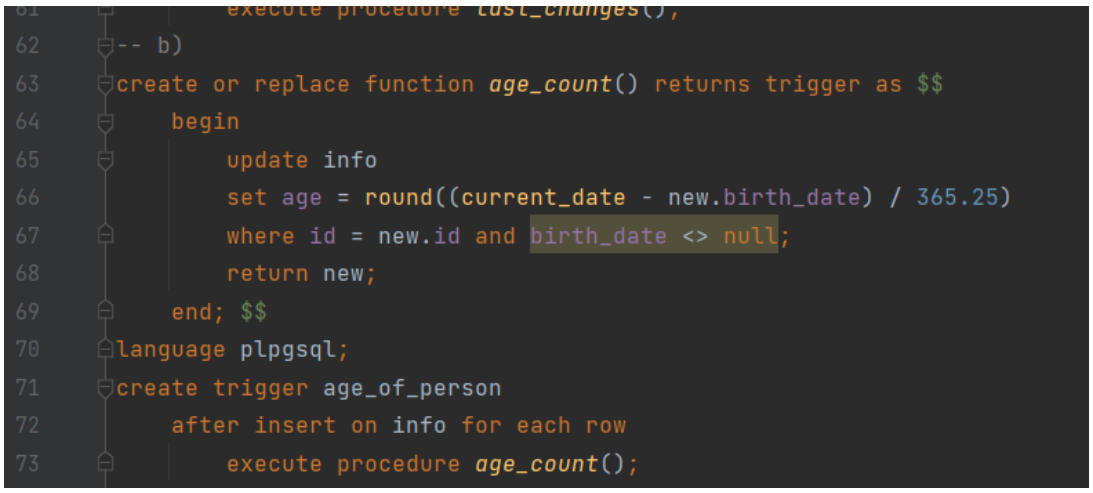
```
-- TASK1
-- a)
create function increment(i integer) returns integer as $$
    begin
        return i + 1;
    end; $$
language plpgsql;
-- b)
create function sum_of_two(a integer, b integer) returns integer as $$
    begin
        return a + b;
    end; $$
language plpgsql;
-- c)
create function divisible(i integer) returns boolean as $$
    begin
        if (i % 2 = 0) then return true;
            else return false;
        end if;
    end; $$
language plpgsql;
```

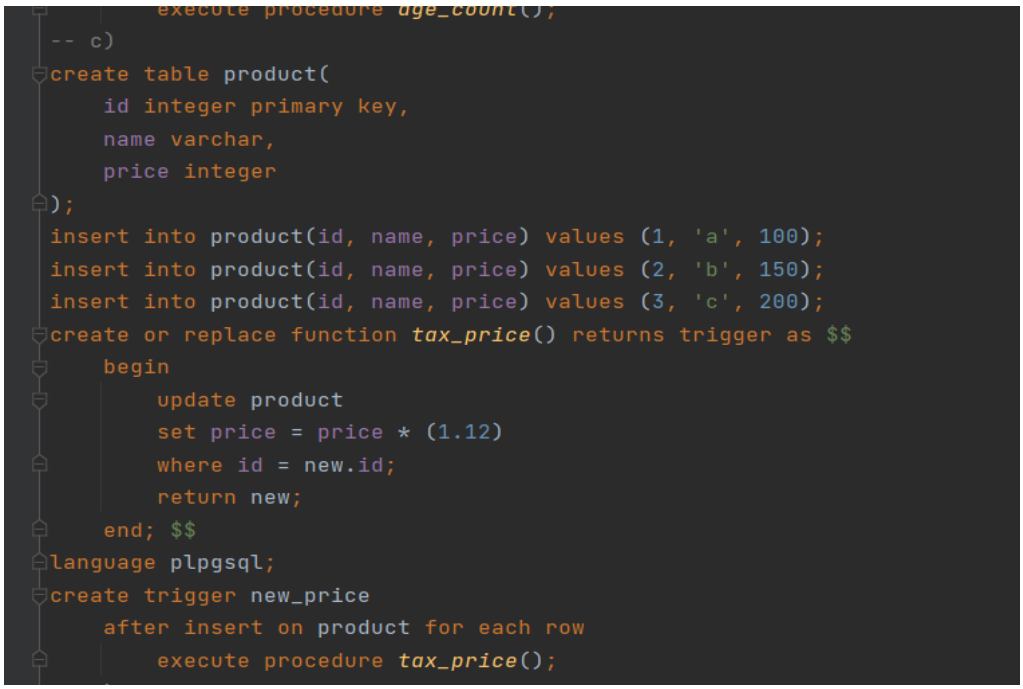
```
-- d)
create or replace function check_password(password varchar(16)) returns boolean as $$
    begin
        if (length(password) < 8) then return false;
            else return true;
        end if;
    end; $$
language plpgsql;
-- e)
create or replace function two_output(words varchar(25), out word1 varchar(25),
out word2 varchar(25)) as $$
    begin
        word1 = split_part(words, ' ', 1);
        word2 = split_part(words, ' ', 2);
    end; $$
language plpgsql;
```



```sql
-- TASK2
-- a)
create table info (
    id integer primary key,
    name varchar,
    course integer,
    age integer,
    birth_date date,
    changes timestamp(6)
);
create function last_changes() returns trigger as $$
    begin
        if new.course <> old.course then
            insert into info(id, name, course, age, birth_date, changes)
            values (old.id, old.name, new.course, old.age, old.birth_date, now());
        end if;
        return new;
    end; $$
language plpgsql;
create trigger lasst_changes
    before update on info for each row
        execute procedure last_changes();
```



```sql
        execute procedure last_changes();
-- b)
create or replace function age_count() returns trigger as $$
    begin
        update info
        set age = round((current_date - new.birth_date) / 365.25)
        where id = new.id and birth_date <> null;
        return new;
    end; $$
language plpgsql;
create trigger age_of_person
    after insert on info for each row
        execute procedure age_count();
```



```sql
        execute procedure age_count();
-- c)
create table product(
    id integer primary key,
    name varchar,
    price integer
);
insert into product(id, name, price) values (1, 'a', 100);
insert into product(id, name, price) values (2, 'b', 150);
insert into product(id, name, price) values (3, 'c', 200);
create or replace function tax_price() returns trigger as $$
    begin
        update product
        set price = price * (1.12)
        where id = new.id;
        return new;
    end; $$
language plpgsql;
create trigger new_price
    after insert on product for each row
        execute procedure tax_price();
```

```
-- d)
create or replace function del_prev() returns trigger as $$
    begin
        raise exception 'cannot be deleted';
    end; $$
language plpgsql;
create trigger prevent
    after delete on product for each row
    execute procedure del_prev();
delete from product where price = 150;
```

```
create table person(
    name varchar(20),
    first_name_last_name varchar(25)
    password text,
    validity boolean
);
create or replace function two_launches() returns trigger language plpgsql
    as $$
    begin
        if check_password( password: new.password) = true then
            update person
            set validity = true, first_name_last_name = two_output( words: name)
            where name = new.name;
        else
            update person
            set validity = false
            where name = new.name;
        end if;
        return new;
    end; $$
create trigger two_changes
    after insert on person for each row
        execute procedure two_launches();
```

3.What is the difference between procedure and and function?

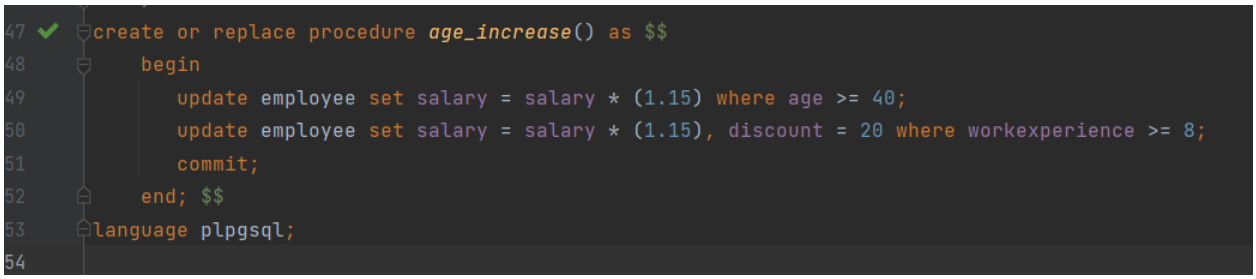
Function:

- the function cannot call a stored procedure
- you can call functions from a select statement
- no transactions are allowed
- only select is allowed
- must return a result or value to the caller

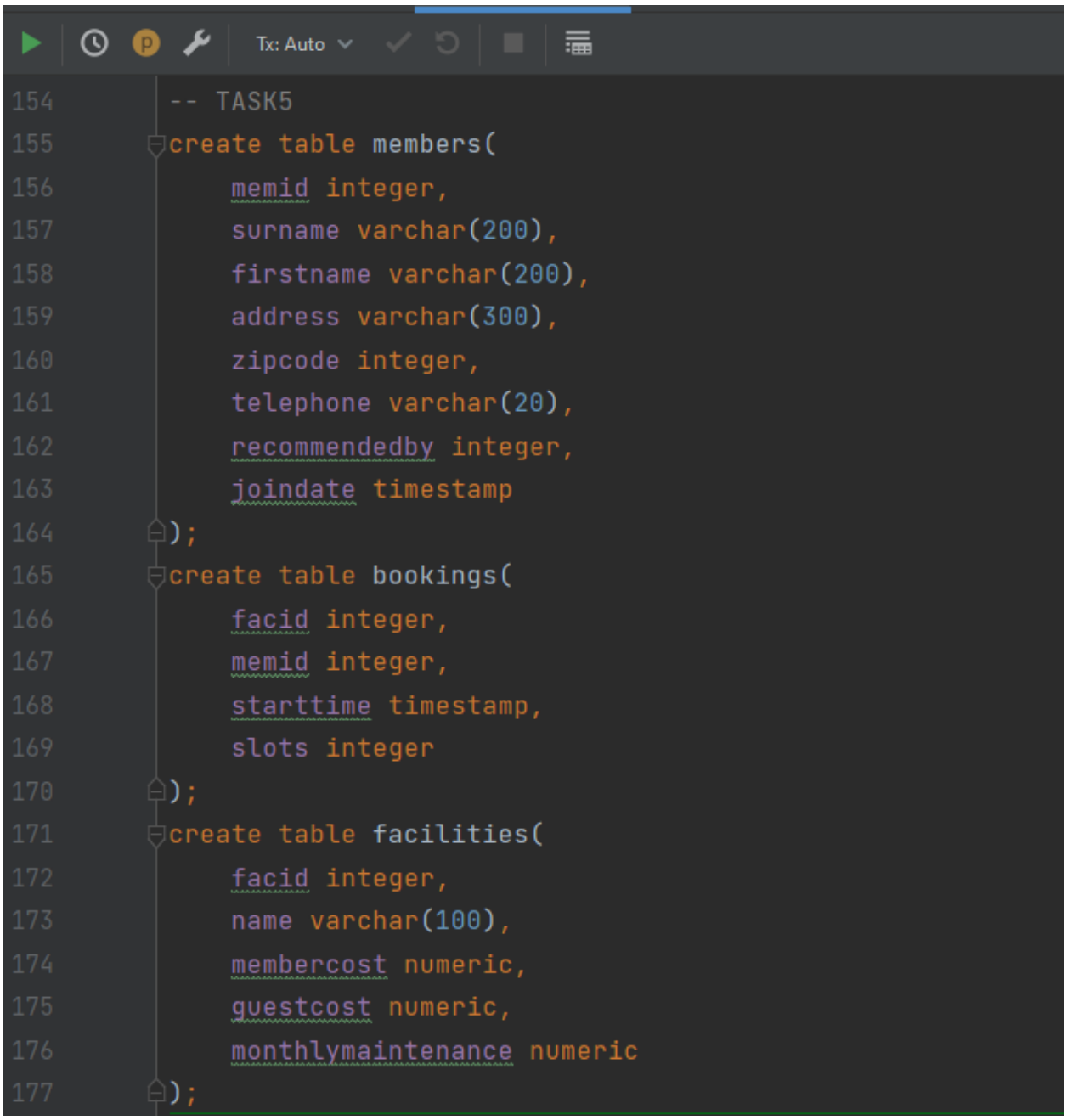
Procedure:

- stored procedures can call functions as needed
- there is no provision to call procedures from select/having and where statements
- transactions can be used in stored procedures
- need not return any value
- operations – insert, update, delete

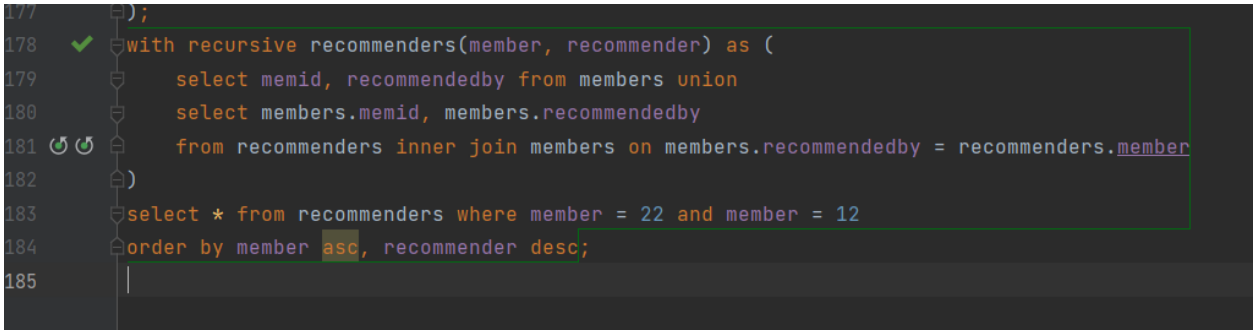
```
        execute procedure two_launches();
-- TASK4
-- a)
create table employee(
    id integer primary key,
    name varchar,
    date_of_birth date,
    age integer,
    salary integer,
    workexperience integer,
    discount integer
);
create or replace procedure increase_salary() as $$
    begin
        update employee set salary = salary * (1.1) ^ (workexperience / 2), discount = 10 where workexperience > 2;
        update employee set discount = discount + (workexperience / 5) where workexperience > 5;
        commit;
    end; $$
language plpgsql;
```



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



```
-- 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(100),
    membercost numeric,
    questcost numeric,
    monthlymaintenance numeric
);
```



```
);
with recursive recommenders(member, recommender) as (
    select memid, recommendedby from members union
    select members.memid, members.recommendedby
    from recommenders inner join members on members.recommendedby = recommenders.member
)
select * from recommenders where member = 22 and member = 12
order by member asc, recommender desc;
```