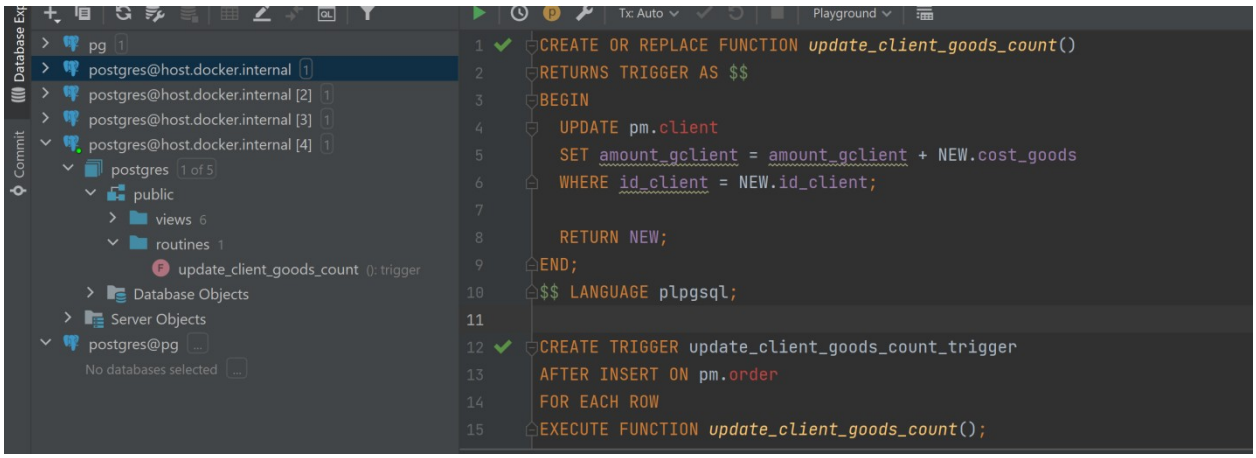
11)Создать не менее 2 триггеров.

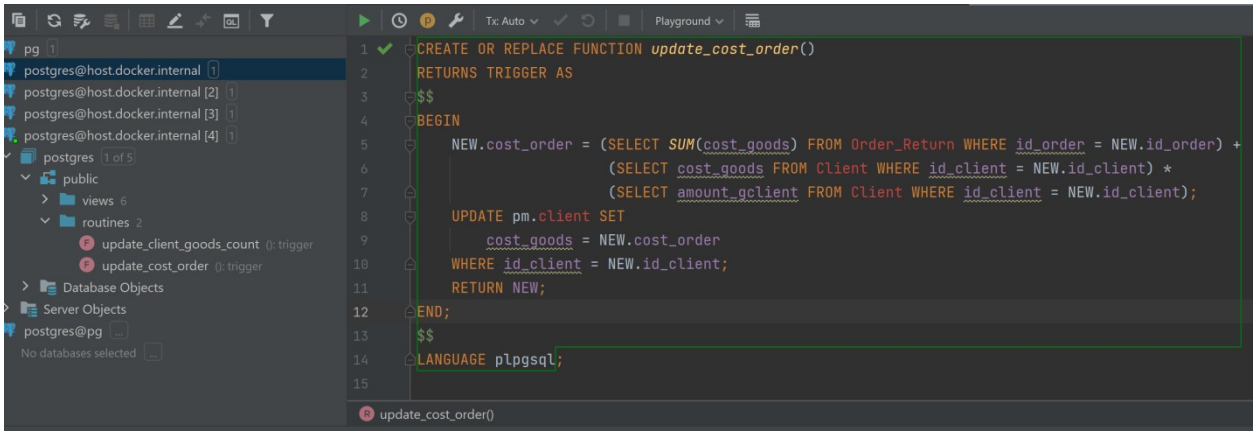
1. Триггер для увеличение общего количества приобретённых товаров клиента при добавлении нового заказа:

```
CREATE OR REPLACE FUNCTION
update_client_goods_count()
RETURNS TRIGGER AS $$
BEGIN
  UPDATE pm.client
  SET amount_gclient = amount_gclient +
NEW.cost_goods
  WHERE id_client = NEW.id_client;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_client_goods_count_trigger
AFTER INSERT ON pm.order
FOR EACH ROW
EXECUTE FUNCTION update_client_goods_count();
```



2. Триггер для подсчета стоимости товара заказа и обновления стоимости товара у клиента при добавлении нового заказа в архив заказов:

```
CREATE OR REPLACE FUNCTION update_cost_order()
RETURNS TRIGGER AS
$$
BEGIN
    NEW.cost_order = (SELECT SUM(cost_goods) FROM
Order_Return WHERE id_order = NEW.id_order) +
                     (SELECT cost_goods FROM Client
WHERE id_client = NEW.id_client) *
                     (SELECT amount_gclient FROM
Client WHERE id_client = NEW.id_client);
    UPDATE pm.client SET
        cost_goods = NEW.cost_order
    WHERE id_client = NEW.id_client;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER update_cost_order
BEFORE INSERT ON pm.orders_archive
FOR EACH ROW
EXECUTE FUNCTION update_cost_order();
```



12)Используя любимый язык программирования и библиотеку, сгенерировать данные и с их помощью вставить данные в уже оформленную БД.

```
import sqlite3

# создание таблицы клиентов
def create_table_client():
    with sqlite3.connect('test.db') as db:
        cursor = db.cursor()
        cursor.execute('''
```

```
            CREATE TABLE IF NOT EXISTS client (
                id_client INTEGER PRIMARY KEY,
                fullname_client VARCHAR(64),
                birthdate_client DATE,
                gender_client VARCHAR(64),
                cost_goods INTEGER,
                amount_gclient INTEGER NOT NULL
            )
        ''')
        db.commit()

# создание таблицы архива заказов
def create_table_orders_archive():
    with sqlite3.connect('test.db') as db:
        cursor = db.cursor()
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS Orders_Archive (
                id_order INTEGER PRIMARY KEY,
                id_client INTEGER,
                time_order DATE,
                cost_order INTEGER,
                FOREIGN KEY (id_client) REFERENCES
client(id_client)
            )
        ''')
        db.commit()

create_table_client() # создание таблицы client
create_table_orders_archive() # создание таблицы Orders_Archive

import random
import datetime

# генерация случайных данных клиентов
def generate_clients(num_clients):
    first_names = ['John', 'Jane', 'Alice', 'Bob', 'Mary',
'Peter']
    last_names = ['Doe', 'Smith', 'Johnson', 'Brown', 'Davis',
'Garcia']
    genders = ['Male', 'Female', 'Other']
    with sqlite3.connect('test.db') as db:
        cursor = db.cursor()
        for i in range(num_clients):
            fullname = f"{random.choice(first_names)}
{random.choice(last_names)}"
            birthdate = datetime.date(random.randint(1960,
2002), random.randint(1, 12), random.randint(1, 28))
            gender = random.choice(genders)
            cost_goods = random.randint(10, 100)
            amount_gclient = random.randint(1, 10)
            cursor.execute('INSERT INTO client (fullname_client,
birthdate_client, gender_client, cost_goods, amount_gclient)
VALUES (?, ?, ?, ?, ?)', (fullname, birthdate, gender,
```

```
cost_goods, amount_gclient))
        db.commit()

generate_clients(10) # добавление 10 клиентов
# генерация случайных данных заказов
def generate_orders(num_orders):
    time_now = datetime.datetime.now().date()
    with sqlite3.connect('test.db') as db:
        cursor = db.cursor()
        for i in range(num_orders):
            id_client = random.randint(1, 10)
            time_order = datetime.date(random.randint(2021,
2022), random.randint(1, 12), random.randint(1, 28))
            cost_order = random.randint(10, 100)
            cursor.execute('INSERT INTO Orders_Archive
(id_client, time_order, cost_order) VALUES (?, ?, ?)',
(id_client, time_order, cost_order))
            cursor.execute('UPDATE client SET amount_gclient =
amount_gclient + 1 WHERE id_client = ?', (id_client,))
        db.commit()

generate_orders(20) # добавление 20 заказов


# запрос с агрегированием данных
def analysis_data():
    with sqlite3.connect('test.db') as db:
        cursor = db.cursor()
        cursor.execute('''
            SELECT
                c.id_client,
                c.fullname_client,
                COUNT(o.id_order) AS num_orders,
                SUM(o.cost_order) AS total_cost_order,
                AVG(o.cost_order) AS avg_cost_order
            FROM client c
            JOIN Orders_Archive o ON c.id_client = o.id_client
            GROUP BY c.id_client;
        ''')
        results = cursor.fetchall()
        for row in results:
            print('ID:', row[0])
            print('Name:', row[1])
            print('Number of orders:', row[2])
            print('Total cost of orders:', row[3])
            print('Average cost of orders:', row[4])
            print()

analysis_data() # вывод данных
```

Вывод:

```
ID: 1
Name: Jane Johnson
Number of orders: 1
Total cost of orders: 18
Average cost of orders: 18.0

ID: 2
Name: Alice Johnson
Number of orders: 1
Total cost of orders: 82
Average cost of orders: 82.0

ID: 3
Name: Peter Doe
Number of orders: 2
Total cost of orders: 115
Average cost of orders: 57.5

ID: 4
Name: Jane Davis
Number of orders: 3
Total cost of orders: 121
Average cost of orders: 40.333333333333336

ID: 5
Name: John Smith
Number of orders: 2
Total cost of orders: 125
Average cost of orders: 62.5
```

```
ID: 6
Name: John Garcia
Number of orders: 2
Total cost of orders: 118
Average cost of orders: 59.0

ID: 7
Name: John Doe
Number of orders: 3
Total cost of orders: 223
Average cost of orders: 74.33333333333333

ID: 8
Name: Peter Johnson
Number of orders: 3
Total cost of orders: 255
Average cost of orders: 85.0

ID: 9
Name: Bob Garcia
Number of orders: 1
Total cost of orders: 82
Average cost of orders: 82.0

ID: 10
Name: Peter Davis
Number of orders: 2
Total cost of orders: 148
Average cost of orders: 74.0
```