Итоговая аттестация по курсу Java. Отчёт.

Задача 1. Установка и настройка PostgreSQL

1. Рекомендована 12 версия PostgreSQL. Установить PostgreSQL на локальную машину https://postgrespro.ru/docs/postgrespro/14/binary-installation-on-windows или использовать Docker-контейнер.
2. Проверить подключение к PostgreSQL из IDE Intellij Idea https://www.jetbrains.com/help/idea/postgresql.html#connect-to-postgresql-database

Упрощение задачи 1. Реализовать схему в онлайн-редакторе. Например, в редакторе https://sqliteonline.com/. Обратить внимание, что реализовать требуется с помощью PostgreSQL.

Задача 2. Проектирование и создание схемы БД

- **Файл schema.sql** должен содержать: Таблицы: ✦ **product** — id, описание, стоимость, количество, категория.
- ✦ **customer** — id, имя, фамилия, телефон, email.
- ✦ **order** — id, product_id (FK), customer_id (FK), дата заказа, количество, статус.
- ✦ **order_status** — справочник статусов заказов (id, имя статуса).
-

- Первичные и внешние ключи, ограничения NOT NULL, CHECK (например, цена ≥ 0, количество ≥ 0).
- Индексы по внешним ключам и дате заказа.
- Комментарии к таблицам и ключевым полям.
- `CREATE TABLE IF NOT EXISTS`, чтобы скрипт можно было запускать повторно.
-

- **Заполнение тестовыми данными** — минимум 10 строк в каждой таблице.

* Схема данных может быть выбрана любой на усмотрение студента. Можно проявить фантазию и создать Базу Данных на ваш лад.

Задача 3. SQL-запросы

Создать файл **test-queries.sql**, включающий не менее 10 запросов:

- 5 запросов на **чтение** (в том числе JOIN с несколькими таблицами, агрегаты, сортировка, фильтрация).
- 3 запроса на **изменение** (UPDATE).
- 2 запроса на **удаление** (DELETE). Примеры:
- Список всех заказов за последние 7 дней с именем покупателя и описанием товара.
- Топ-3 самых популярных товара.
- Обновление количества на складе при покупке.
- Удаление клиентов без заказов.

Задача 4. Java-приложение (Maven, без Spring)

- Создать Maven-проект (Java 17) с зависимостями: PostgreSQL JDBC Driver
- (опционально) Flyway для миграций
-
- Реализовать класс `App.java`, который: Подключается к БД через JDBC (`DriverManager`).
- Выполняет автоматический запуск миграций (если используется Flyway).
  1. Демонстрирует CRUD-операции: Вставка нового товара и покупателя (PreparedStatement).
  1. Создание заказа для покупателя.
  1. Чтение и вывод последних 5 заказов с JOIN на товары и покупателей.
  1. Обновление цены товара и количества на складе.
  1. Удаление тестовых записей.
  1.
1. Работает в транзакции с `commit()` и `rollback()` при ошибках.
1. Выводит результаты операций в консоль в удобочитаемом формате.
1.
   1. Параметры подключения (URL, user, password) вынести в `application.properties`

Результат

1. Репозиторий на GitHub (ветка + Pull Request).

Скриншоты: Установка/подключение PostgreSQL.

Выполнение миграций.

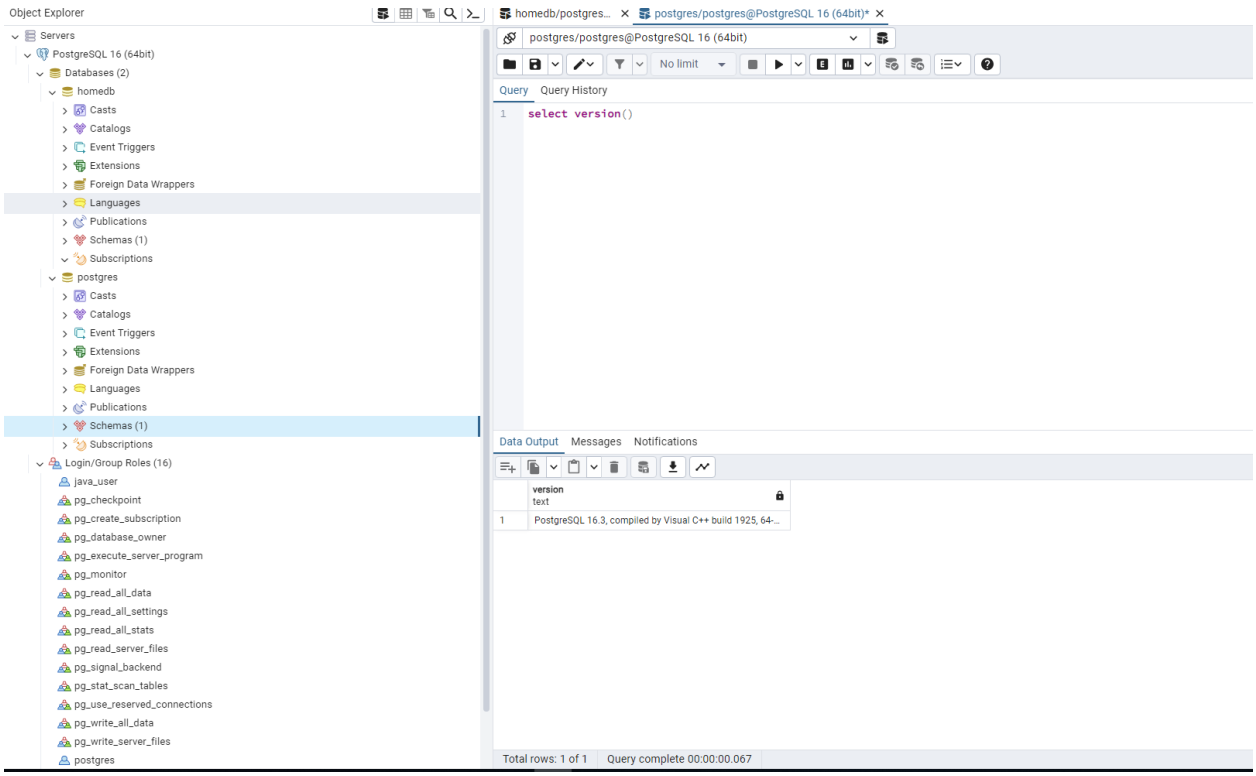
ER-диаграмма или схема БД.

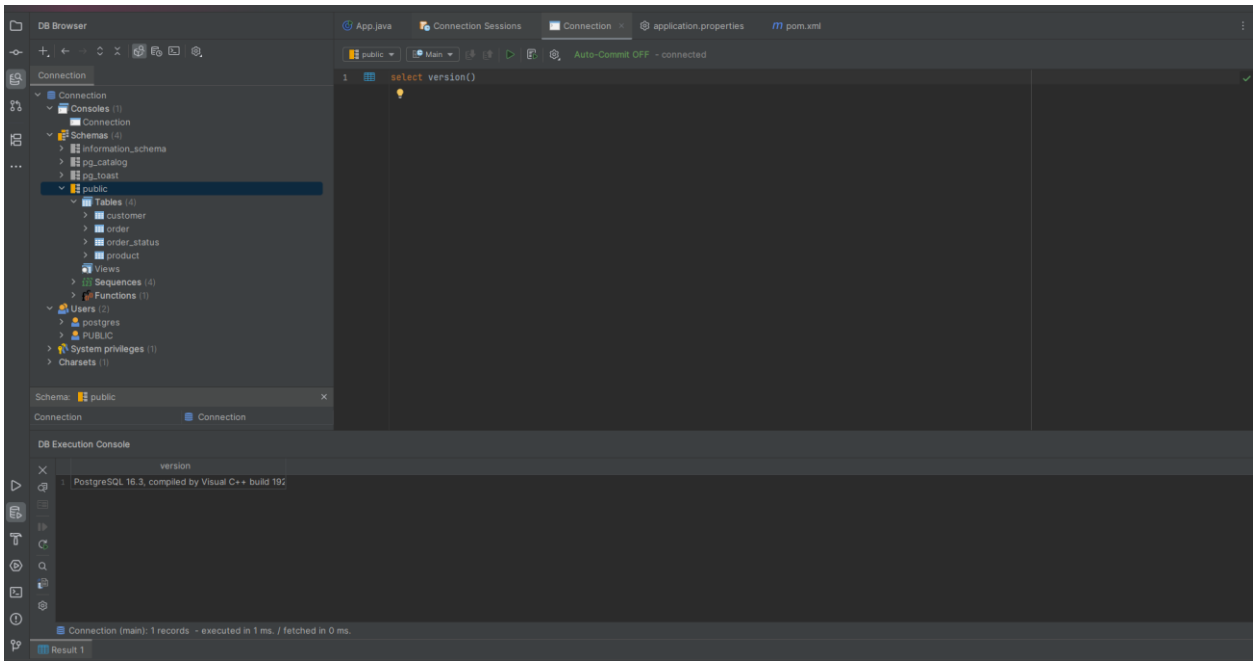
Запуск Java-приложения и вывод CRUD-операций.

Результаты выполнения `test-queries.sql` в IDE.

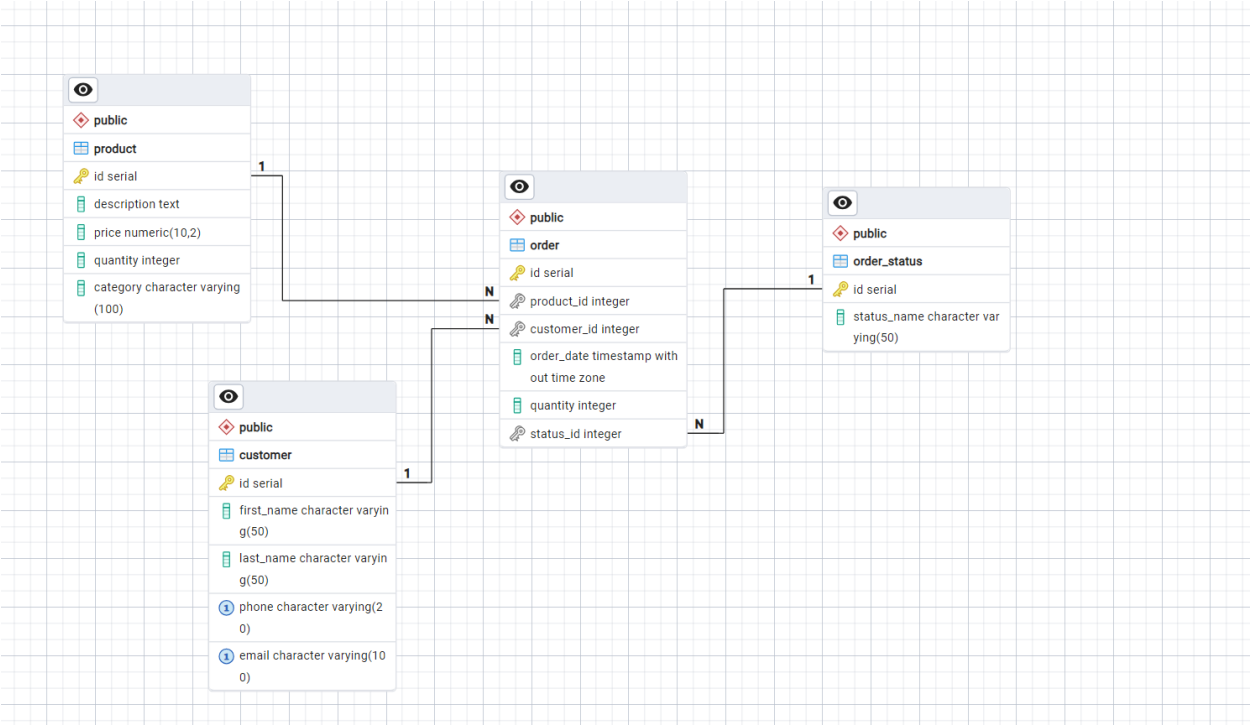
Отчёт по работе:

1. **Задача 1. Установка и настройка PostgreSQL**





2. Задача 2. Проектирование и создание схемы БД



Скрипт создания:

Собраны в файле CreateTables.txt

```
-- Создание таблицы order_status (справочник статусов заказов)
CREATE TABLE IF NOT EXISTS order_status (
    id SERIAL PRIMARY KEY,
    status_name VARCHAR(50) NOT NULL
);

COMMENT ON TABLE order_status IS 'Справочник статусов заказов';
COMMENT ON COLUMN order_status.id IS 'Идентификатор статуса';
COMMENT ON COLUMN order_status.status_name IS 'Название статуса заказа';

-- Создание таблицы product (продукты)
CREATE TABLE IF NOT EXISTS product (
    id SERIAL PRIMARY KEY,
    description TEXT NOT NULL,
    price NUMERIC(10, 2) CHECK (price >= 0),
    quantity INT CHECK (quantity >= 0),
    category VARCHAR(100) NOT NULL
```

```
);

COMMENT ON TABLE product IS 'Таблица продуктов';
COMMENT ON COLUMN product.id IS 'Идентификатор продукта';
COMMENT ON COLUMN product.description IS 'Описание продукта';
COMMENT ON COLUMN product.price IS 'Стоимость продукта';
COMMENT ON COLUMN product.quantity IS 'Количество продукта на складе';
COMMENT ON COLUMN product.category IS 'Категория продукта';

-- Создание таблицы customer (клиенты)
CREATE TABLE IF NOT EXISTS customer (
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    phone VARCHAR(20) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE
);

COMMENT ON TABLE customer IS 'Таблица клиентов';
COMMENT ON COLUMN customer.id IS 'Идентификатор клиента';
COMMENT ON COLUMN customer.first_name IS 'Имя клиента';
COMMENT ON COLUMN customer.last_name IS 'Фамилия клиента';
COMMENT ON COLUMN customer.phone IS 'Телефон клиента';
COMMENT ON COLUMN customer.email IS 'Email клиента';

-- Создание таблицы order (заказы)
CREATE TABLE IF NOT EXISTS "order" (
    id SERIAL PRIMARY KEY,
    product_id INT NOT NULL REFERENCES product(id),
    customer_id INT NOT NULL REFERENCES customer(id),
    order_date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    quantity INT CHECK (quantity > 0),
```

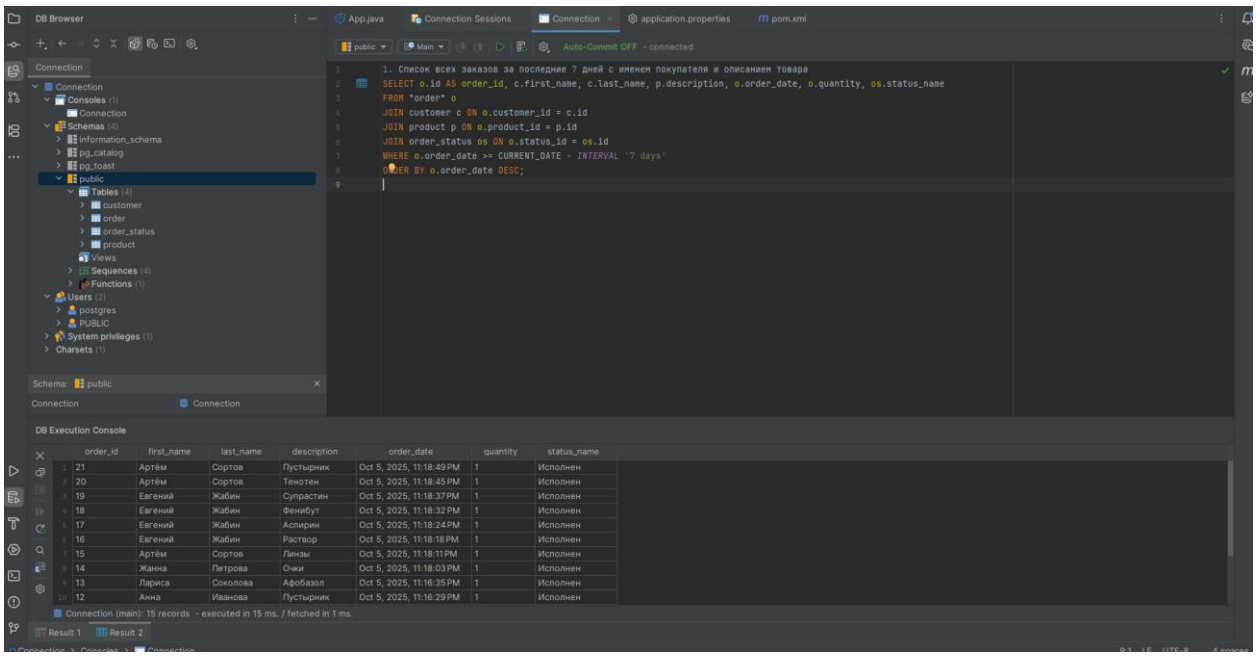
```
    status_id INT NOT NULL REFERENCES order_status(id)
);


COMMENT ON TABLE "order" IS 'Таблица заказов';
COMMENT ON COLUMN "order".id IS 'Идентификатор заказа';
COMMENT ON COLUMN "order".product_id IS 'Идентификатор продукта';
COMMENT ON COLUMN "order".customer_id IS 'Идентификатор клиента';
COMMENT ON COLUMN "order".order_date IS 'Дата заказа';
COMMENT ON COLUMN "order".quantity IS 'Количество заказанного продукта';
COMMENT ON COLUMN "order".status_id IS 'Идентификатор статуса заказа';

-- Создание индексов
CREATE INDEX IF NOT EXISTS idx_order_product_id ON "order" (product_id);
CREATE INDEX IF NOT EXISTS idx_order_customer_id ON "order" (customer_id);
CREATE INDEX IF NOT EXISTS idx_order_status_id ON "order" (status_id);
CREATE INDEX IF NOT EXISTS idx_order_order_date ON "order" (order_date);
```

3. **Задача 3. SQL-запросы**

Собраны в файле test-queries.sql

1. Список всех заказов за последние 7 дней с именем покупателя и описанием товара

```
SELECT o.id AS order_id, c.first_name, c.last_name, p.description, o.order_date, o.quantity, os.status_name
FROM "order" o
JOIN customer c ON o.customer_id = c.id
JOIN product p ON o.product_id = p.id
JOIN order_status os ON o.status_id = os.id
WHERE o.order_date >= CURRENT_DATE - INTERVAL '7 days'
ORDER BY o.order_date DESC;
```



2. Топ-3 самых популярных товара (по количеству заказов)

SELECT p.id, p.description, COUNT(o.id) AS total_orders

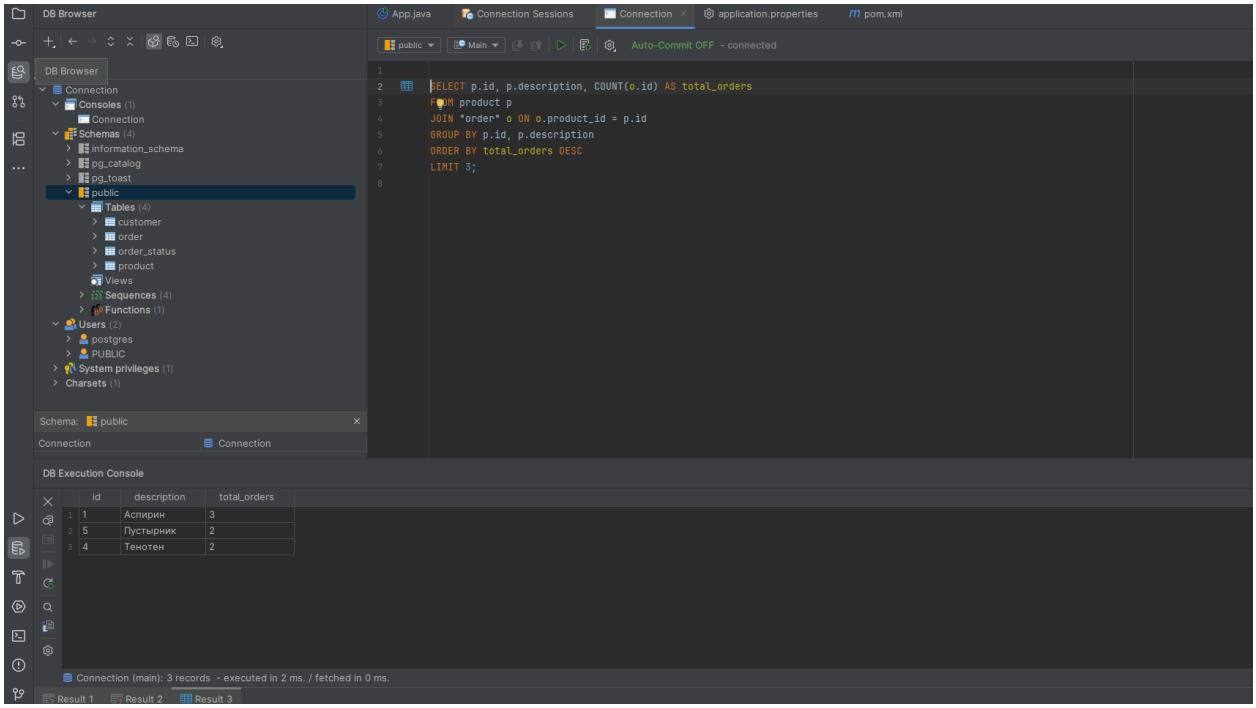
FROM product p

JOIN "order" o ON o.product_id = p.id

GROUP BY p.id, p.description

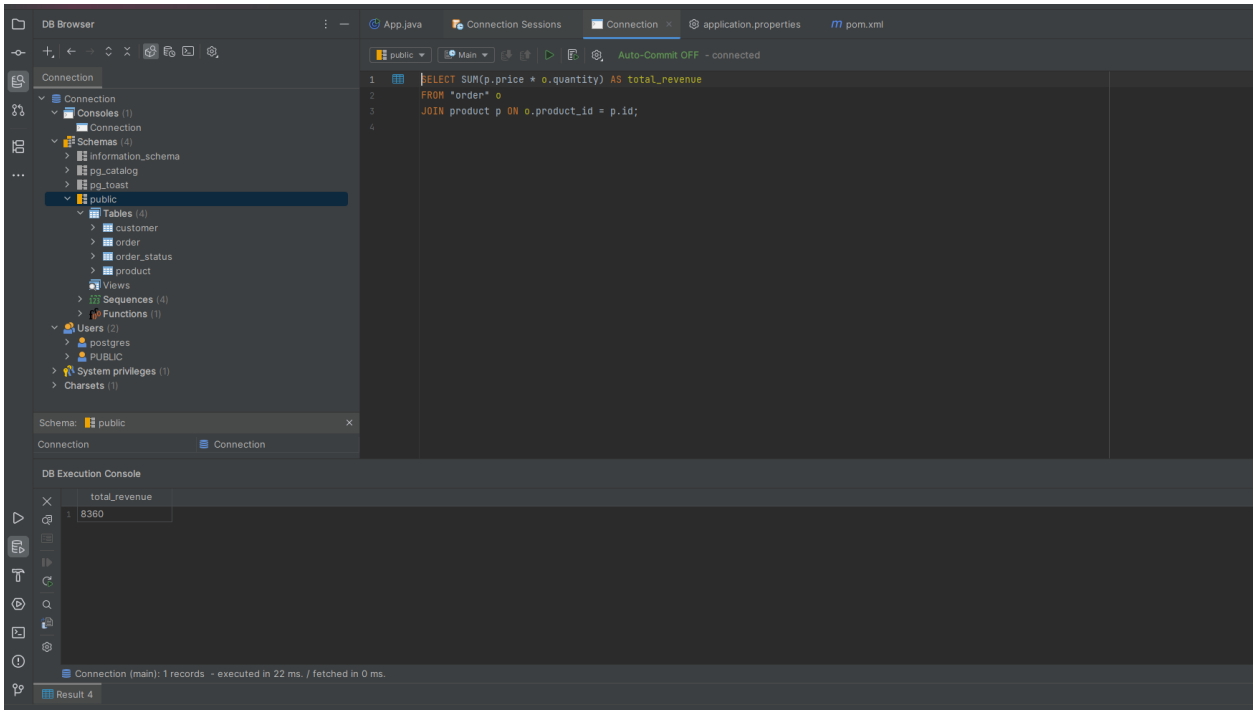
ORDER BY total_orders DESC

LIMIT 3;



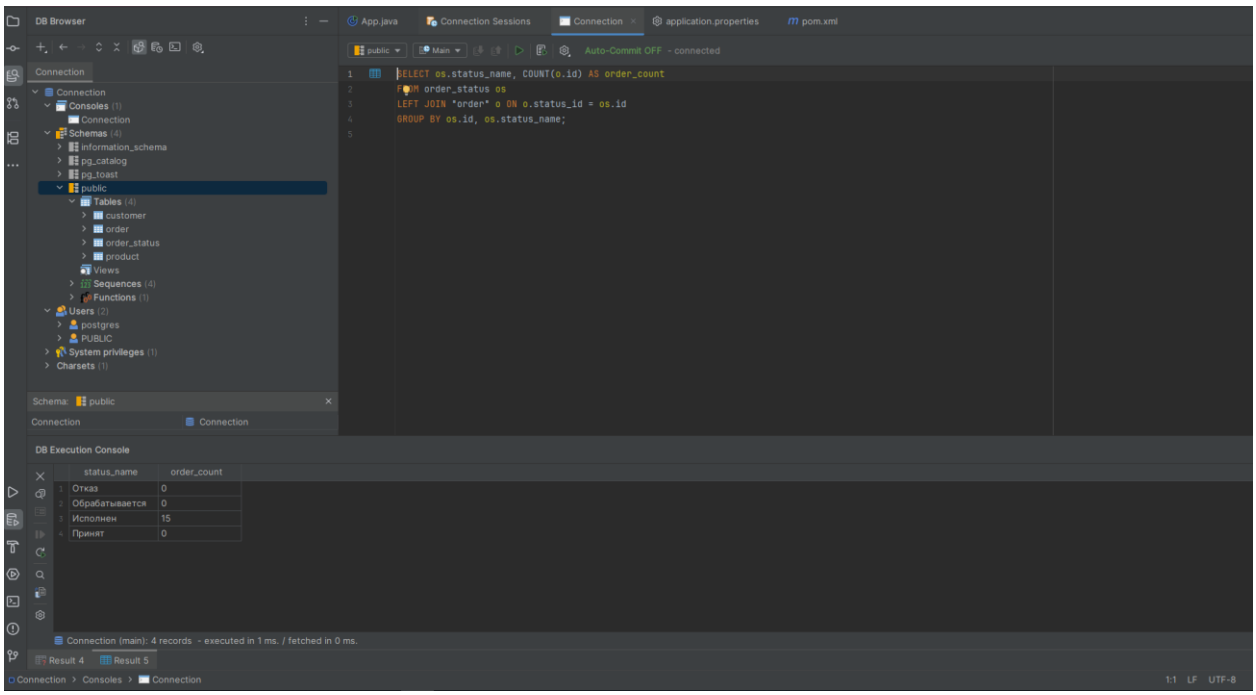
3. Общая сумма выручки по всем заказам

```
SELECT SUM(p.price * o.quantity) AS total_revenue
FROM "order" o
JOIN product p ON o.product_id = p.id;
```



4. Количество заказов по статусам

```
SELECT os.status_name, COUNT(o.id) AS order_count
FROM order_status os
LEFT JOIN "order" o ON o.status_id = os.id
GROUP BY os.id, os.status_name;
```



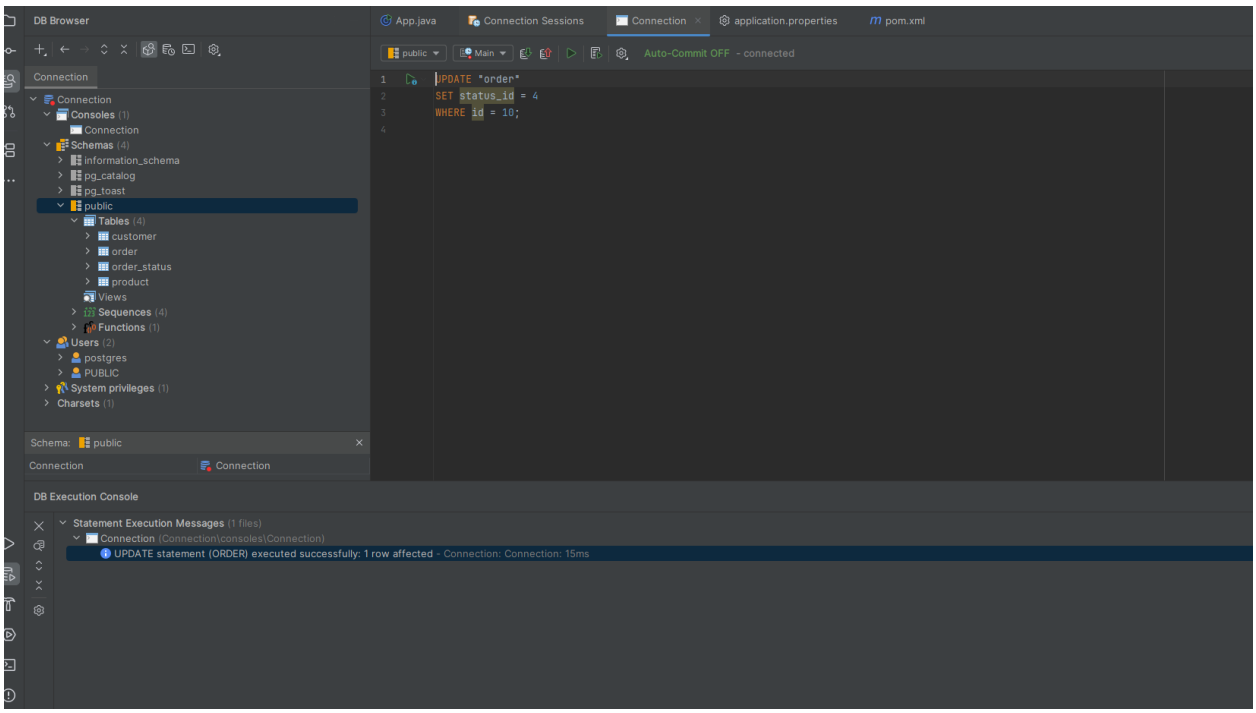
5. Список клиентов, сделавших более 5 заказов

```
SELECT c.id, c.first_name, c.last_name, COUNT(o.id) AS order_count
FROM customer c
JOIN "order" o ON o.customer_id = c.id
GROUP BY c.id, c.first_name, c.last_name
HAVING COUNT(o.id) > 5;
```

3 запроса на изменение

1. Обновить статус заказа на “Исполнен”

```
UPDATE "order"
SET status_id = 3
WHERE id = 6;
```

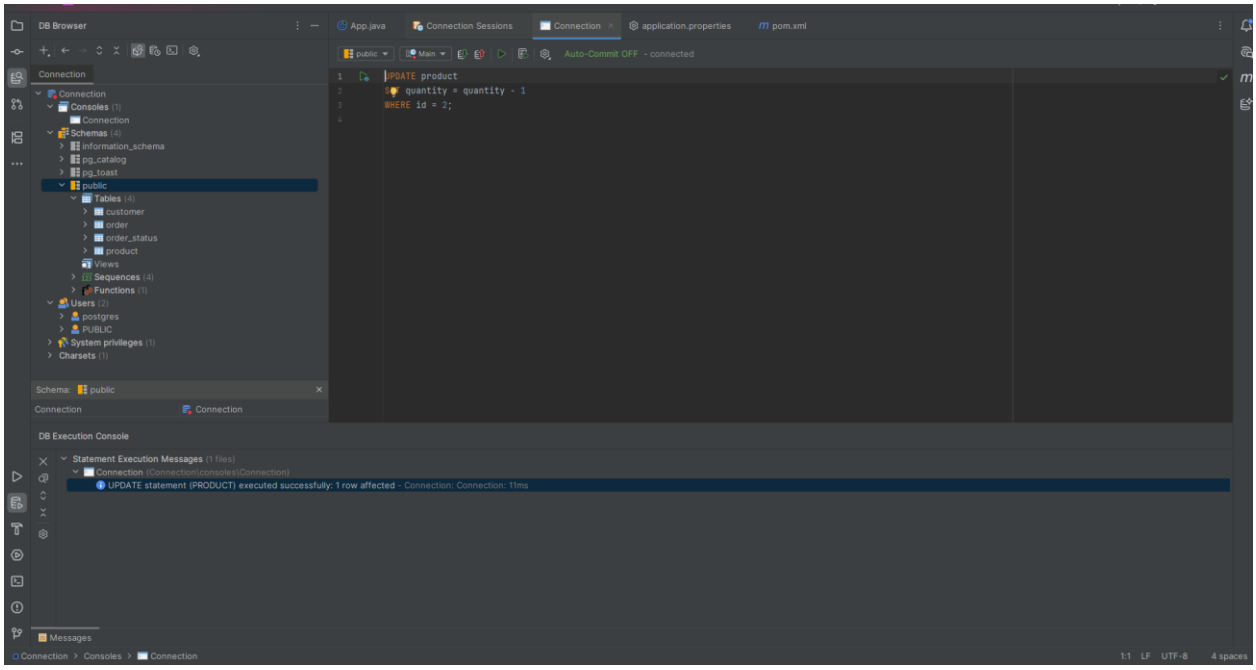


2. Уменьшить количество товара при покупке

UPDATE product

SET quantity = quantity - 1

WHERE id = 2;



Функция, которая принимает заказ и проверяет доступное количество товара. Если всё ок - меняет статус на Исполнено и уменьшает количество товара, если нет, то статус Отказ. Возвращает номер заказа (order.id):

```
declare
        cur_product int;
        new_order_id int;
begin
        --пишем первый статус Принят
        insert into public.order(product_id,customer_id,order_date,quantity,status_id)
values(c_product,c_customer,now(),c_quantity,1);
         -- получаем id только что вставленной записи
   new_order_id := currval('order_id_seq');
        update public.order set status_id=2
        where "id"=currval('order_id_seq');
        --если товара достаточно, то обрабатываем с уменьшением в product_и установкой статуса
Исполнено
        select quantity into cur_product from public.product where "id"=c_product for update;
        if cur_product>=c_quantity then
                update public.order set status_id=3
                where "id"=currval('order_id_seq');

                update public.product set quantity=quantity-c_quantity
                where "id"=c_product;
        --если товара недостаточно, то устанавливаем статус Отказ
        else
                update public.order set status_id=4
                where "id"=currval('order_id_seq');
        end if;
         -- возвращаем id созданного заказа
   RETURN new_order_id;
end;
```

3. Обновить цену товара

UPDATE product

SET price = 500

WHERE id = 3;

2 запроса на удаление

1. Удалить клиентов без заказов

DELETE FROM customer c

WHERE NOT EXISTS (

  SELECT 1 FROM "order" o WHERE o.customer_id = c.id);

2. Удалить заказы по определенному статусу (например, отмененные - статус “Отменен” с id=4)

DELETE FROM "order"

WHERE status_id = 4;

4. Задача 4. Java-приложение (Maven, без Spring)

Код приложения:

```
package org.example;

import java.io.FileInputStream;
import java.io.IOException;
import java.sql.*;
import java.util.Properties;

public class App {
    private static String DB_URL;
    private static String DB_USER;
    private static String DB_PASSWORD;

    public static void main(String[] args) {
        loadProperties();

        try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD)) {
            // Включаем ручное управление транзакциями
            conn.setAutoCommit(false);

            try {
                // Создание таблиц
                createSchema(conn);
```

```
                // 1. Вставка нового товара и покупателя
                int productId = insertProduct(conn, "Андипал", 10.0, 50,
"Аптека");
                int customerId = insertCustomer(conn, "Артём", "Соколов",
"79056251254", "sokol@test.ru");
                System.out.println("Вставлены товар ID=" + productId + " и
покупатель ID=" + customerId);

                // 2. Создание заказа для покупателя
                int orderId = createOrder(conn, productId, customerId, 2, 3);
                System.out.println("Создан заказ ID=" + orderId);


                // Второй вариант - через вызов функцию с уменьшением в
product
                String sql = "SELECT create_order(?, ?, ?)";

                PreparedStatement pstmt = conn.prepareStatement(sql);

                pstmt.setInt(1, 1);
                pstmt.setInt(2, 1);
                pstmt.setInt(3, 1);

                ResultSet rs = pstmt.executeQuery();
                rs.next();
                int orderidfunc = rs.getInt(1);
                System.out.println("Функция вызвана - создан заказ id: " +
orderidfunc);

                // 3. Чтение и вывод последних 5 заказов
                System.out.println("\nПоследние 5 заказов:");
                readLastFiveorder(conn);

                // 4. Обновление цены товара и количества на складе
                updateProduct(conn, productId, 950.0, 45);
                System.out.println("Обновлен товар ID=" + productId);

                // 5. Удаление тестовых записей
                deleteTestRecords(conn, productId, customerId, orderId,
orderidfunc);
                System.out.println("Удалены тестовые записи - " + "товар id="
+ productId + ", клиент id=" + customerId + ", заказы id=" +
                        orderId + ", id=" + orderidfunc );

                // Зафиксировать все операции
                conn.commit();
            } catch (Exception ex) {
                System.out.println("Ошибка: " + ex.getMessage());
                conn.rollback();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void loadProperties() {
        Properties props = new Properties();
        try (FileInputStream fis = new
FileInputStream("src/main/resources/application.properties")) {
            props.load(fis);
            DB_URL = props.getProperty("db.url");
            DB_USER = props.getProperty("db.user");
            DB_PASSWORD = props.getProperty("db.password");
```

```
        } catch (IOException e) {
            throw new RuntimeException("Не удалось загрузить конфигурацию
базы данных", e);
        }
    }

    private static void createSchema(Connection conn) throws SQLException {
        Statement stmt = conn.createStatement();

        String sqlProduct = "CREATE TABLE IF NOT EXISTS product (" +
                "id SERIAL PRIMARY KEY," +
                "description TEXT NOT NULL," +
                "price NUMERIC(10, 2) CHECK (price >= 0)," +
                "quantity INT CHECK (quantity >= 0)," +
                "category VARCHAR(100) NOT NULL" +
                ");";

        String sqlCustomer = "CREATE TABLE IF NOT EXISTS customer (" +
                "id SERIAL PRIMARY KEY," +
                "first_name VARCHAR(50) NOT NULL," +
                "last_name VARCHAR(50) NOT NULL," +
                "phone VARCHAR(20) NOT NULL UNIQUE," +
                "email VARCHAR(100) NOT NULL UNIQUE" +
                ");";

        String sqlorder = "CREATE TABLE IF NOT EXISTS \"order\" (" +
                "id SERIAL PRIMARY KEY," +
                "product_id INT NOT NULL REFERENCES product(id)," +
                "customer_id INT NOT NULL REFERENCES customer(id)," +
                "order_date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP," +
                "quantity INT CHECK (quantity > 0)," +
                "status_id INT NOT NULL REFERENCES order_status(id)" +
                ")";

        String sqlordertatus = "CREATE TABLE IF NOT EXISTS order_status (" +
                "id SERIAL PRIMARY KEY," +
                "status_name VARCHAR(50) NOT NULL" +
                ");";

        stmt.execute(sqlProduct);
        stmt.execute(sqlCustomer);
        stmt.execute(sqlordertatus);
        stmt.execute(sqlorder);
        stmt.execute("COMMENT ON TABLE product IS 'Таблица продуктов';");
        stmt.execute("COMMENT ON TABLE customer IS 'Таблица клиентов';");
        stmt.execute("COMMENT ON TABLE \"order\" IS 'Таблица заказов';");
        stmt.execute("COMMENT ON TABLE order_status IS 'Справочник статусов
заказов';");

        stmt.close();
    }

    private static int insertProduct(Connection conn, String description,
double price, int quantity, String category) throws SQLException {
        String sql = "INSERT INTO product (description, price,
quantity,category) VALUES (?, ?, ?, ?) RETURNING id;";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, description);
            pstmt.setDouble(2, price);
            pstmt.setInt(3, quantity);
            pstmt.setString(4, category);
            try (ResultSet rs = pstmt.executeQuery()) {
                if (rs.next()) {
                    return rs.getInt(1);
```

```
                }
            }
        }
        throw new SQLException("Не удалось вставить товар");
    }

    private static int insertCustomer(Connection conn, String first_name,
String last_name, String phone, String email) throws SQLException {
        String sql = "INSERT INTO customer (first_name, last_name, phone,
email) VALUES (?, ?, ?, ?) RETURNING id;";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, first_name);
            pstmt.setString(2, last_name);
            pstmt.setString(3, phone);
            pstmt.setString(4, email);
            try (ResultSet rs = pstmt.executeQuery()) {
                if (rs.next()) {
                    return rs.getInt(1);
                }
            }
        }
        throw new SQLException("Не удалось вставить покупателя");
    }

    private static int createOrder(Connection conn, int productId, int
customerId, int quantity, int statusId) throws SQLException {
        String insertSql = "INSERT INTO \"order\" (product_id, customer_id,
order_date, quantity, status_id) " +
                "VALUES (?, ?, CURRENT_TIMESTAMP, ?, ?) RETURNING id;";

        try (PreparedStatement pstmt = conn.prepareStatement(insertSql)) {
            pstmt.setInt(1, productId);
            pstmt.setInt(2, customerId);
            pstmt.setInt(3, quantity);
            pstmt.setInt(4, statusId);

            try (ResultSet rs = pstmt.executeQuery()) {
                if (rs.next()) {
                    return rs.getInt(1);
                }
            }
        }
        throw new SQLException("Не удалось создать заказ");
    }

    private static void readLastFiveorder(Connection conn) throws
SQLException {
        String sql = "SELECT o.id as order_id, o.order_date, c.last_name  as
customer_name, p.description as product_name, o.quantity ,oi.status_name " +
                "FROM \"order\" o " +
                "JOIN customer c ON o.customer_id = c.id " +
                "JOIN order_status oi ON o.status_id = oi.id " +
                "JOIN product p ON o.product_id = p.id " +
                "ORDER BY o.order_date DESC " +
                "LIMIT 5;";

        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                int orderId = rs.getInt("order_id");
                Timestamp date = rs.getTimestamp("order_date");
                String customer = rs.getString("customer_name");
                String product = rs.getString("product_name");
                int qty = rs.getInt("quantity");
```

```
                System.out.printf("Заказ ID=%d, дата=%s, клиент=%s, товар=%s,
кол-во=%d%n",
                        orderId, date.toString(), customer, product, qty);
            }
        }
    }

    private static void updateProduct(Connection conn, int productId, double
newPrice, int newQuantity) throws SQLException {
        String sql = "UPDATE product SET price = ?, quantity = ? WHERE id =
?;";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setDouble(1, newPrice);
            pstmt.setInt(2, newQuantity);
            pstmt.setInt(3, productId);
            int affected = pstmt.executeUpdate();
            if (affected == 0) {
                throw new SQLException("Товар не найден или не обновлен");
            }
        }
    }

    private static void deleteTestRecords(Connection conn, int productId, int
customerId, int orderId, int orderidfunc) throws SQLException {
        // Удаляем из order
        String deleteorder = "DELETE FROM \"order\" WHERE id = ?;";
        String deleteorderfunc = "DELETE FROM \"order\" WHERE id = ?;";
        // Удаляем из product
        String deleteProduct = "DELETE FROM product WHERE id = ?;";
        // Удаляем из customer
        String deleteCustomer = "DELETE FROM customer WHERE id = ?;";

        try (PreparedStatement pstmtorder =
conn.prepareStatement(deleteorder);
             PreparedStatement pstmtorderfunc =
conn.prepareStatement(deleteorderfunc);
             PreparedStatement pstmtProduct =
conn.prepareStatement(deleteProduct);
             PreparedStatement pstmtCustomer =
conn.prepareStatement(deleteCustomer)) {

            pstmtorder.setInt(1, orderId);
            pstmtorder.executeUpdate();

            pstmtorderfunc.setInt(1, orderidfunc);
            pstmtorderfunc.executeUpdate();

            // Удаление товаров и клиентов
            pstmtProduct.setInt(1, productId);
            pstmtProduct.executeUpdate();

            pstmtCustomer.setInt(1, customerId);
            pstmtCustomer.executeUpdate();
        }
    }
}
```

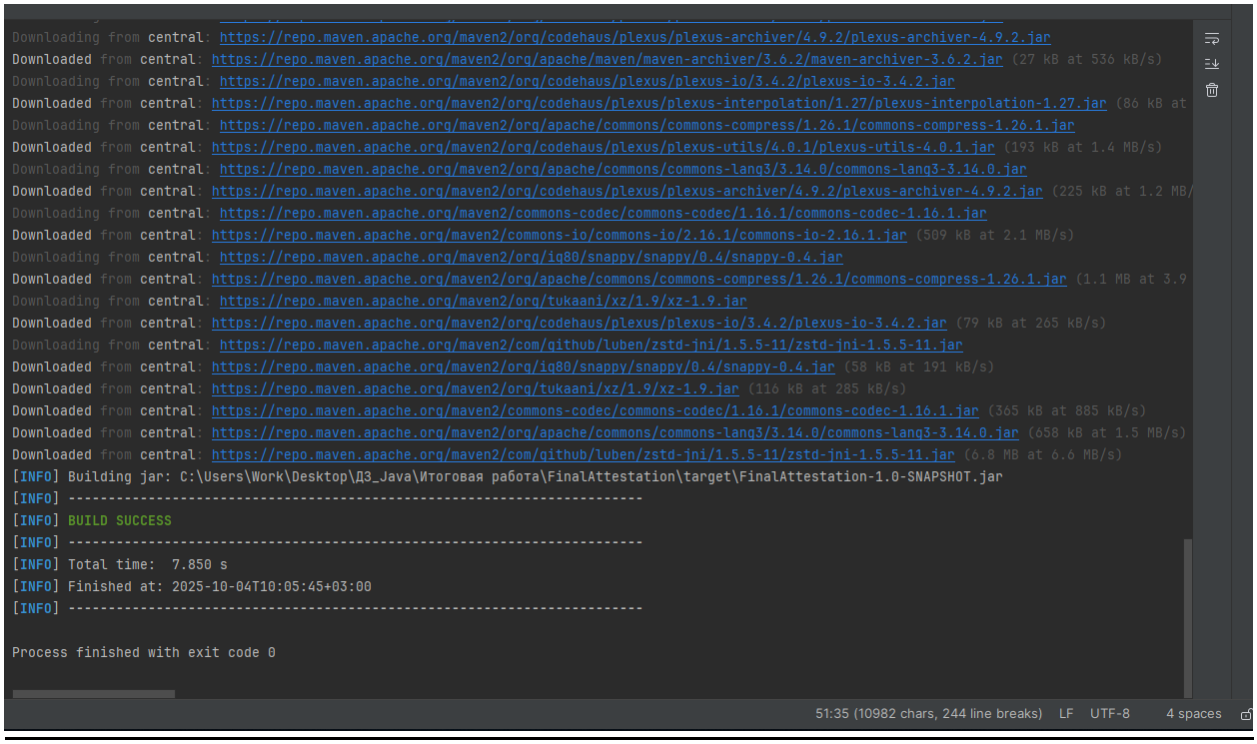
Файл application.properties

```
db.url=jdbc:postgresql://localhost:5432/homedb
db.user=java_user
db.password=java
```

Pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>FinalAttestation</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <version>42.5.4</version>
        </dependency>
    </dependencies>
</project>
```

Сборка:



Результат запуска приложения:

```
C:\Users\Work\.jdks\openjdk-25\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.1.3\lib\idea_rt.jar=61881" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-
Вставлены товар ID=16 и покупатель ID=17
Создан заказ ID=22
Функция вызвана - создан заказ id: 23

Последние 5 заказов:
Заказ ID=23, дата=2025-10-06 00:05:30.564596, клиент=Иванов, товар=Аспирин, кол-во=1
Заказ ID=22, дата=2025-10-06 00:05:30.564596, клиент=Соколов, товар=Андипал, кол-во=2
Заказ ID=21, дата=2025-10-05 23:18:49.022627, клиент=Сортов, товар=Пустырник, кол-во=1
Заказ ID=20, дата=2025-10-05 23:18:45.114074, клиент=Сортов, товар=Тенотен, кол-во=1
Заказ ID=19, дата=2025-10-05 23:18:37.207602, клиент=Жабин, товар=Супрастин, кол-во=1
Обновлен товар ID=16
Удалены тестовые записи - товар id=16, клиент id=17, заказы id=22, id=23

Process finished with exit code 0
```