**Реляционная модель данных — SQLite**

-- Создание и заполнение таблицы "Покупатели":

**CREATE TABLE IF NOT EXISTS Buyers (customer\_id INTEGER PRIMARY KEY AUTOINCREMENT,**

**first\_name VARCHAR(50), last\_name VARCHAR(50), date\_of\_birth DATE);**

**INSERT INTO Buyers (first\_name, last\_name, date\_of\_birth) VALUES**

**('Даниил', 'Шаталов', '2005-07-12'),**

**('Анна', 'Якимова', '2006-12-21'),**

**('Константин', 'Зотов', '2004-02-28'),**

**('Дмитрий', 'Янин', '2001-03-10');**

-- Создание и заполнение таблицы "Товары":

**CREATE TABLE IF NOT EXISTS Goods (product\_id INTEGER PRIMARY KEY AUTOINCREMENT,**

**product\_name VARCHAR(100), price INTEGER);**

**INSERT INTO Goods (product\_name, price) VALUES**

**('Буточка улитка', 60),**

**('Мясной пирожок', 50),**

**('Сочник', 45),**

**('Слойка с вятченой', 85);**

-- Создание и заполнение таблицы "Покупки":

**CREATE TABLE IF NOT EXISTS Shopping (purchase\_id INTEGER PRIMARY KEY AUTOINCREMENT,**

**customer\_id INTEGER, product\_id INTEGER, purchase\_date DATE,**

**FOREIGN KEY (customer\_id) REFERENCES Buyers(customer\_id), FOREIGN KEY (product\_id) REFERENCES Goods(product\_id));**

**INSERT INTO Shopping (customer\_id, product\_id, purchase\_date) VALUES**

**(1, 1, '2025-01-10'),**

**(1, 2, '2025-01-15'),**

**(2, 3, '2025-01-20'),**

**(3, 1, '2025-01-25'),**

**(3, 4, '2025-01-30'),**

**(4, 2, '2025-02-01'),**

**(4, 3, '2025-02-05'),**

**(1, 3, '2025-02-10');**

-- 1) Вывести покупателей с количеством осуществленных покупок:

**SELECT …;**

-- 2) Общую стоимость товаров для каждого покупателя и отсортировать результат в порядке убывания:

**SELECT …;**

-- 3) Получить покупателей, купивших только один товар:

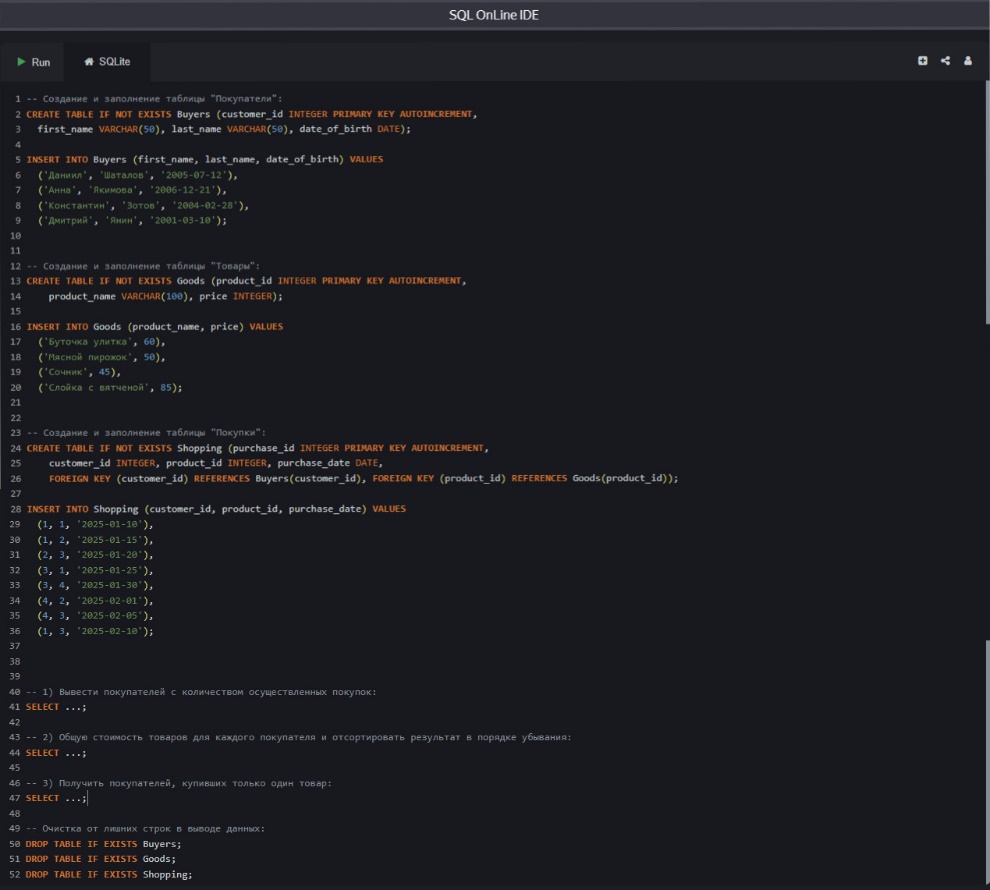
**SELECT …;**

-- Очистка от лишних строк в выводе данных:

**DROP TABLE IF EXISTS Buyers;**

**DROP TABLE IF EXISTS Goods;**

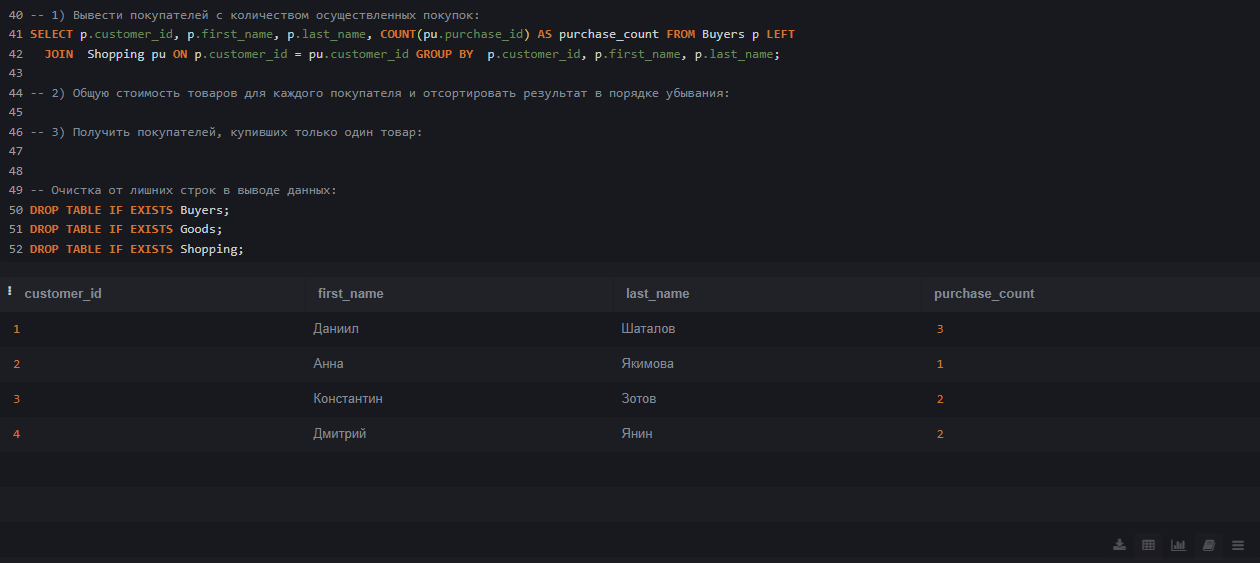
**DROP TABLE IF EXISTS Shopping;**



**SQL-запросы**

**1) Вывести покупателей с количеством осуществленных покупок:**

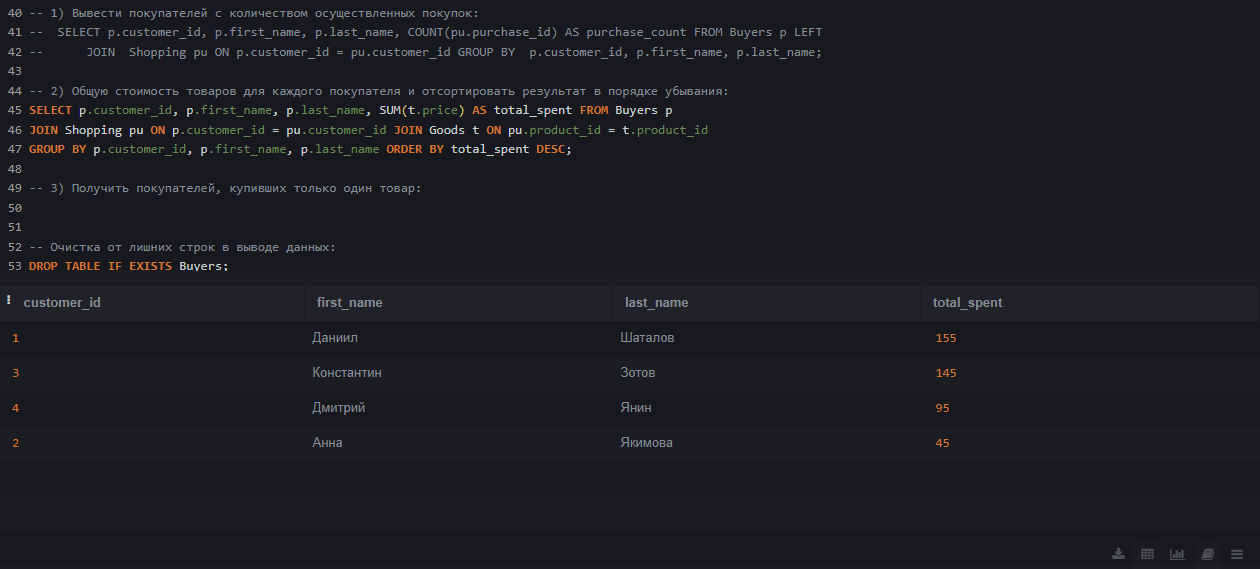
SELECT p.customer\_id, p.first\_name, p.last\_name, COUNT(pu.purchase\_id) AS purchase\_count FROM Buyers p LEFT

JOIN Shopping pu ON p.customer\_id = pu.customer\_id GROUP BY p.customer\_id, p.first\_name, p.last\_name;

**2) Общую стоимость товаров для каждого покупателя и отсортировать результат в порядке убывания:**

SELECT p.customer\_id, p.first\_name, p.last\_name, SUM(t.price) AS total\_spent FROM Buyers p

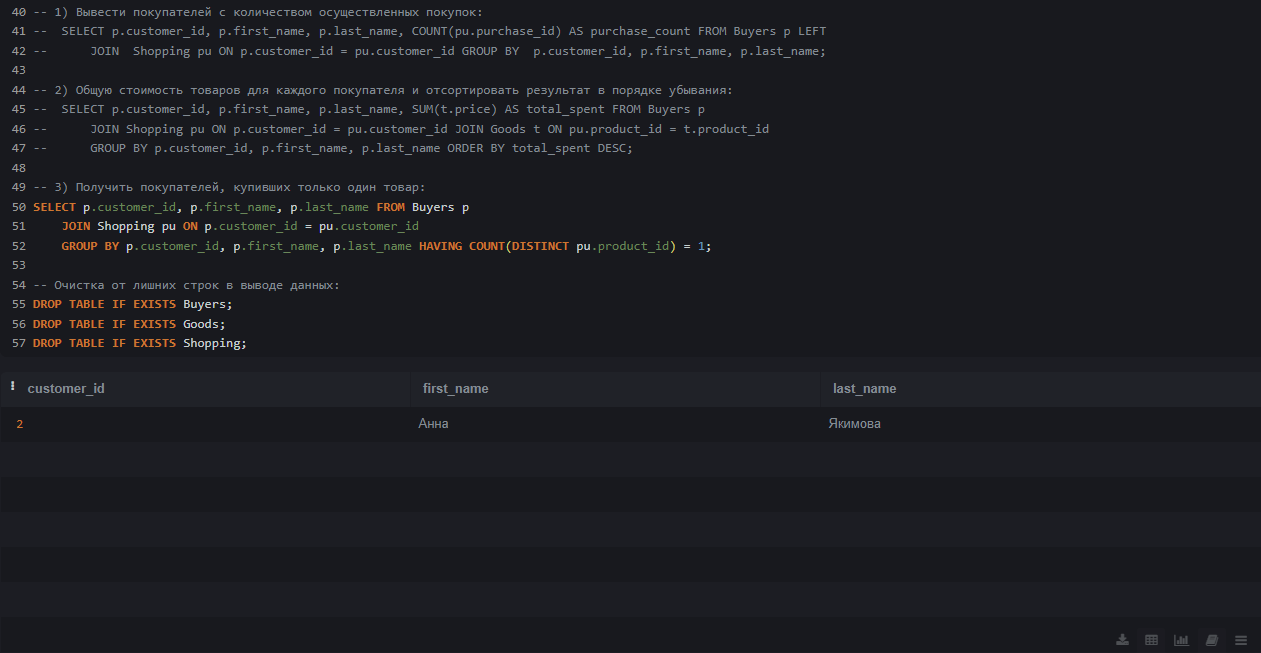
JOIN Shopping pu ON p.customer\_id = pu.customer\_id JOIN Goods t ON pu.product\_id = t.product\_id

GROUP BY p.customer\_id, p.first\_name, p.last\_name ORDER BY total\_spent DESC;

**3) Общую стоимость товаров для каждого покупателя и отсортировать результат в порядке убывания:**

SELECT p.customer\_id, p.first\_name, p.last\_name FROM Buyers p

JOIN Shopping pu ON p.customer\_id = pu.customer\_id

 GROUP BY p.customer\_id, p.first\_name, p.last\_name HAVING COUNT(DISTINCT pu.product\_id) = 1;