**1. Реализовать хранимую процедуру, возвращающую текстовую строку, содержащую**

**информацию о покупателе (фамилия, район проживания, дата, сумма и название**

**магазина последней покупки). Обработать ситуацию, когда покупатель не делал**

**покупок.**

**Объявим хранимую процедуру:**  
  
CREATE OR REPLACE FUNCTION customer\_last\_purchase\_info(p\_customer\_id INT)

RETURNS TEXT AS $$

DECLARE

result TEXT;

BEGIN

SELECT INTO result

c.LastName || ', ' || c.District || ', ' || p.PurchaseDate || ', ' || p.Amount || ', ' || s.StoreName

FROM Customers c

LEFT JOIN Purchases p ON c.CustomerID = p.CustomerID

LEFT JOIN Stores s ON p.StoreID = s.StoreID

WHERE c.CustomerID = p\_customer\_id

ORDER BY p.PurchaseDate DESC

LIMIT 1;

IF result IS NULL THEN

RETURN 'Покупатель не совершал покупок.';

ELSE

RETURN result;

END IF;

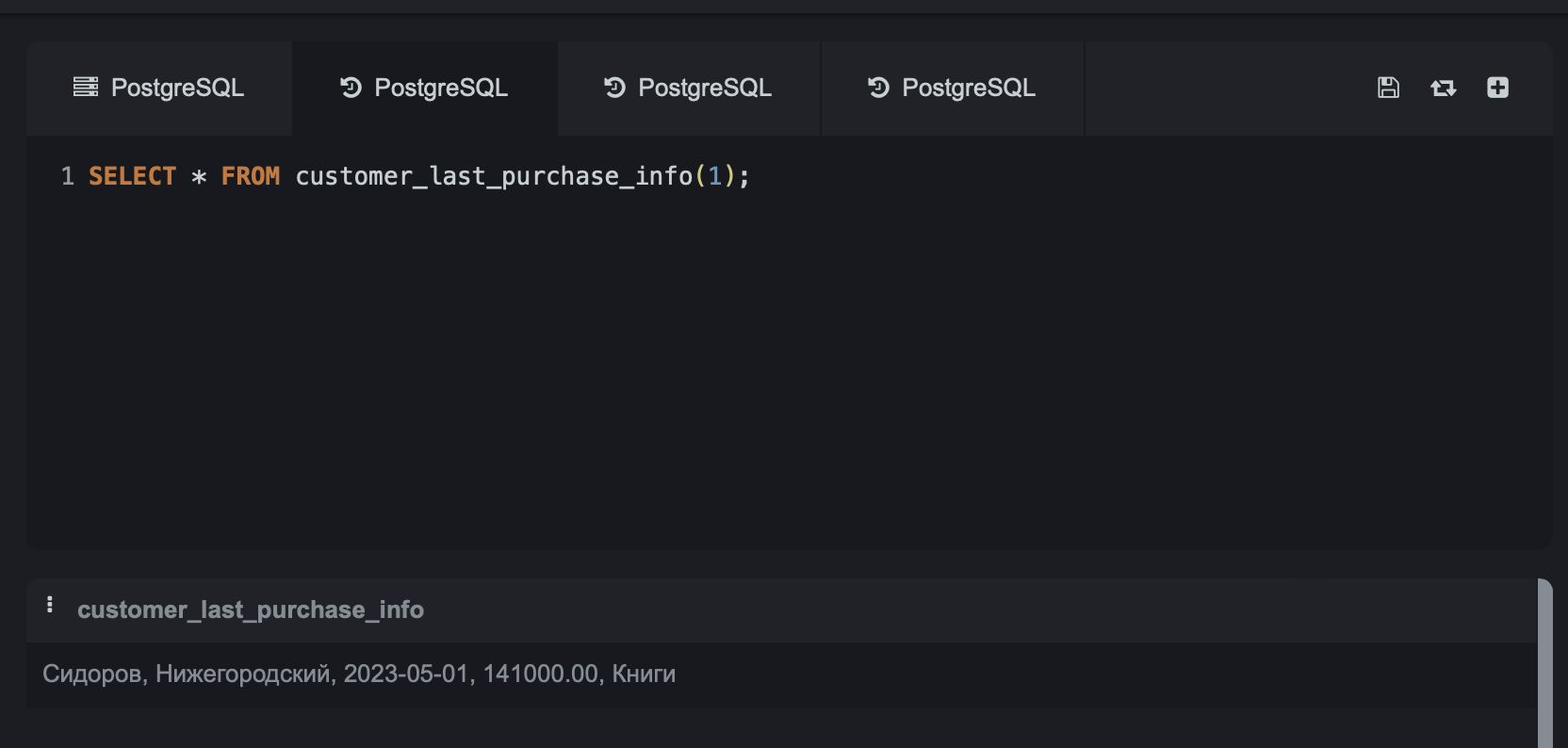
END;

$$ LANGUAGE plpgsql;

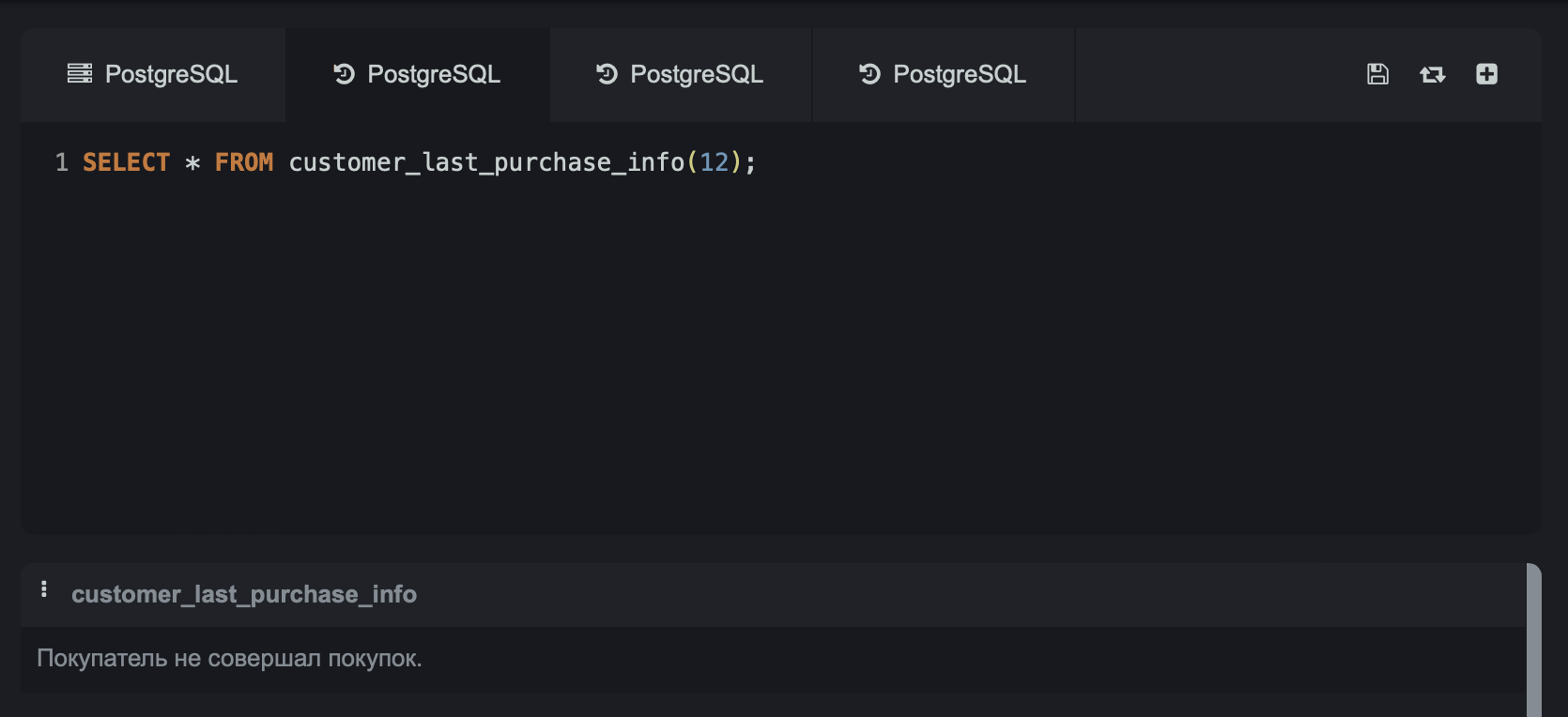
В блоке SELECT функции customer\_last\_purchase\_info, мы получаем следующую информацию о покупателе и его последней покупке:

1. c.LastName: Фамилия покупателя
2. c.District: Район проживания покупателя
3. p.PurchaseDate: Дата последней покупки
4. p.Amount: Сумма последней покупки
5. s.StoreName: Название магазина, где была совершена последняя покупка

Функция объединяет все эти значения, которая соединяет их через разделитель (в данном случае запятая и пробел) и возвращает результат в виде одной текстовой строки. Если покупатель не совершал покупок, функция возвращает сообщение "Покупатель не совершал покупок."

**Проверка:  
  
1. Покупатель делал покупки**

**2. Покупатель не делал покупки**



2. **Добавить возраст покупателя. Добавить категорию книги. При добавлении покупки**

**проверять, можно ли данному покупателю читать данную книгу.**

Добавим столбцы возраст для покупателя и категория для книги (в дальнейшем возрастная категория будет называться «Для взрослых»)  
  
ALTER TABLE Customers

ADD COLUMN Age INT;

ALTER TABLE Books

ADD COLUMN Category VARCHAR(255);  
  
**Заполним таблицу:**  
  
UPDATE Books

SET Category = 'Образование'

WHERE BookID NOT IN (1, 7);

UPDATE Books

SET Category = 'Для взрослых'

WHERE BookID = 1;

UPDATE Books

SET Category = 'Детская литература'

WHERE BookID = 7;

UPDATE Customers

SET Age = 25

WHERE CustomerID NOT IN (1, 6);

UPDATE Customers

SET Age = 45

WHERE CustomerID = 1;

UPDATE Customers

SET Age = 17

WHERE CustomerID = 6;

Теперь таблицы "Книги" и "Покупатель" обновлены, и они содержат данные в столбцах "Категория" и "Возраст" соответственно. Книга с BookID = 1 имеет категорию "Для взрослых", а покупатель с CustomerID = 6 имеет возраст меньше 18 лет.

**Создадим триггер**

CREATE OR REPLACE FUNCTION check\_age\_restriction()

RETURNS TRIGGER AS $$

DECLARE

customer\_age INT;

book\_category VARCHAR;

BEGIN

SELECT Age INTO customer\_age

FROM Customers

WHERE CustomerID = NEW.CustomerID;

SELECT Category INTO book\_category

FROM Books

WHERE BookID = New.bookid;

IF book\_category = 'Для взрослых' AND customer\_age < 18 THEN

RAISE EXCEPTION 'Покупатель слишком молод для покупки этой книги.';

END IF;

RETURN NEW;

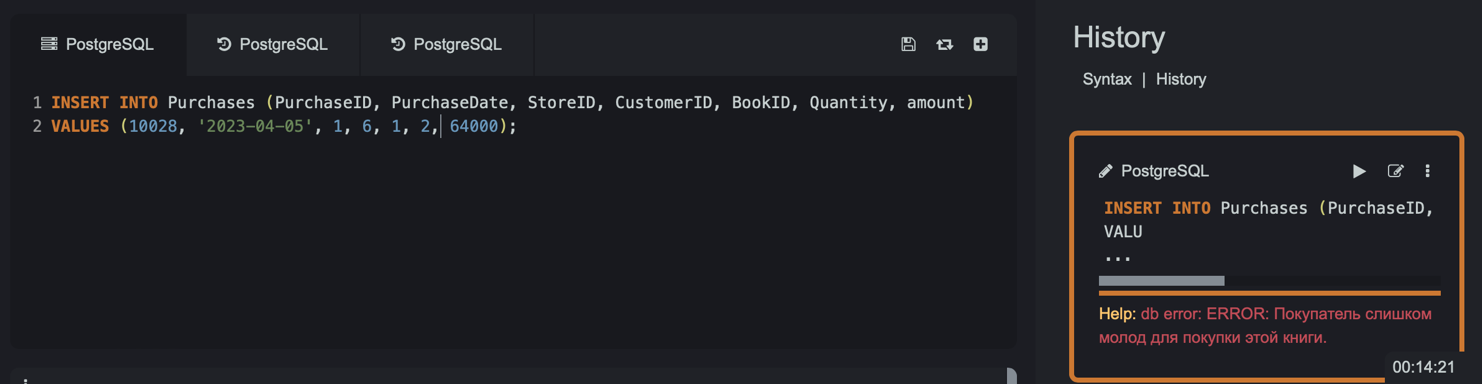
END;

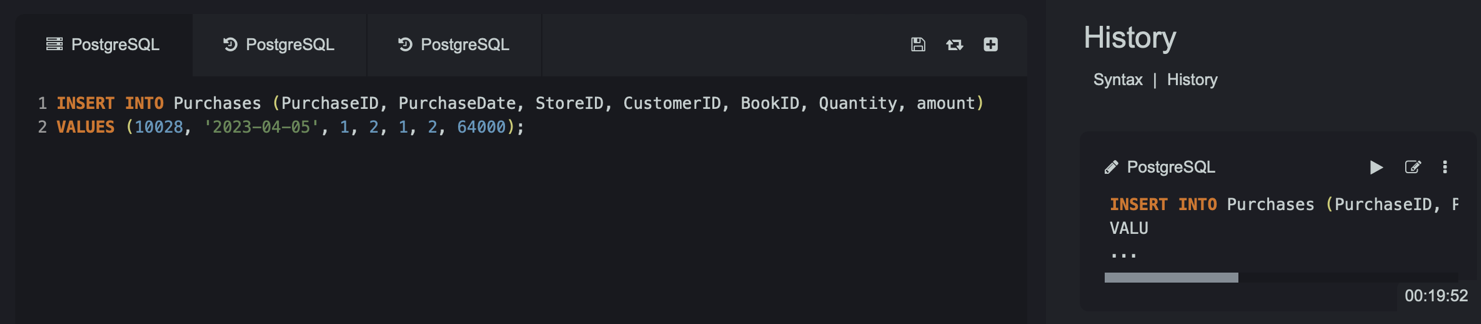
$$ LANGUAGE plpgsql;

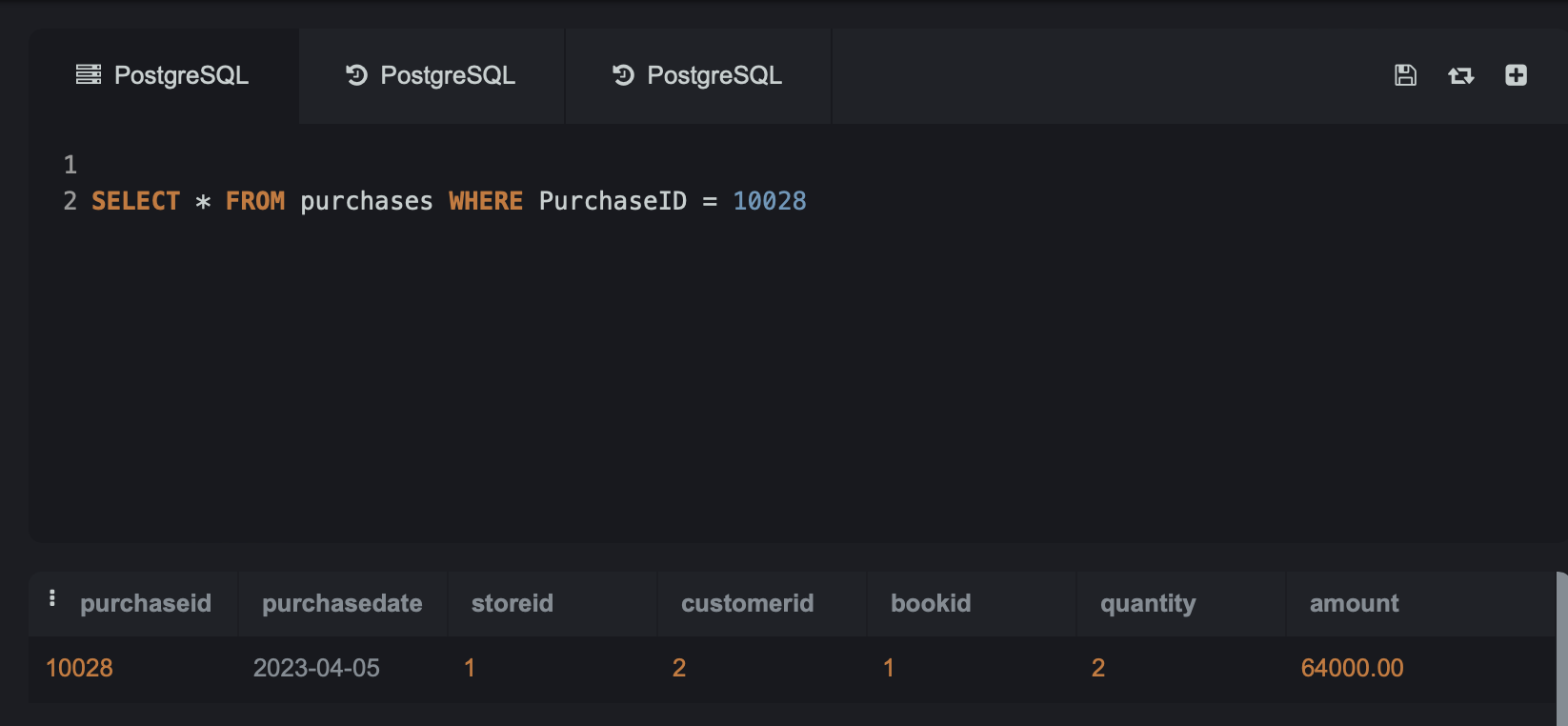
CREATE or replace TRIGGER tr\_check\_age\_restriction

BEFORE INSERT ON Purchases

FOR EACH ROW

EXECUTE FUNCTION check\_age\_restriction();  
  
  
**Проверка 1: Покупатель младше 18**

**Проверка 1: Покупатель старше 18**



**3. Реализовать триггер такой, что при вводе строки в таблице покупок, если сумма**

**покупки не указана, то она вычисляется**

**Создаём функцию для подсчёта суммы**

CREATE OR REPLACE FUNCTION calculate\_purchase\_amount()

RETURNS TRIGGER AS $$

DECLARE

book\_price INT;

book\_quantity INT;

BEGIN

IF NEW.Amount IS NULL THEN

SELECT Price INTO book\_price

FROM Books

WHERE BookID = NEW.BookID;

book\_quantity := NEW.Quantity;

NEW.Amount := book\_price \* book\_quantity;

END IF;

RETURN NEW;

END;

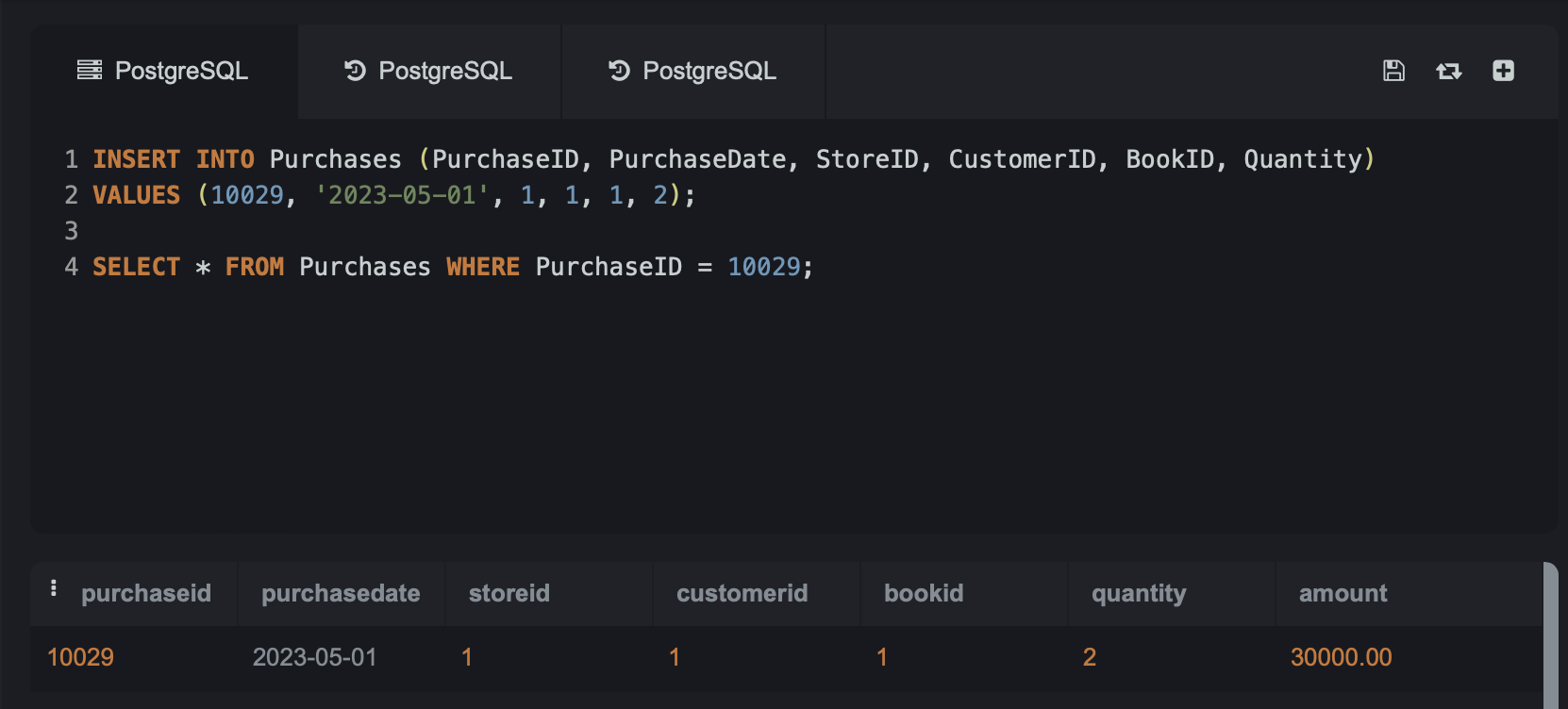
$$ LANGUAGE plpgsql;

**теперь создадим триггер, который будет вызывать эту функцию:**  
CREATE TRIGGER tr\_calculate\_purchase\_amount

BEFORE INSERT ON Purchases

FOR EACH ROW

EXECUTE FUNCTION calculate\_purchase\_amount();  
  
 **Проверка:** Добавим в таблицу покупку без суммы, а затем запросим ёё же и посмотрим появилась ли сумма



**4. Создать представление (view), содержащее поля: номер заказа, имя покупателя,**

**скидка, название книги, цена книги, количество и стоимость. Обеспечить**

**возможность изменения предоставленной скидки. При этом должна быть**

**пересчитана стоимость покупки.**

**Создание представления с указанными полями:**

CREATE VIEW purchase\_view AS

SELECT p.PurchaseID, c.LastName AS CustomerName, c.Discount, b.Title AS BookName,

b.Price AS BookPrice, p.Quantity, p.Amount

FROM Purchases p

JOIN Customers c ON p.CustomerID = c.CustomerID

JOIN Books b ON p.BookID = b.BookID;  
  
Чтобы обеспечить возможность изменения предоставленной скидки и пересчитать стоимость покупки, создадим функцию и триггер INSTEAD OF UPDATE для этого представления:  
  
CREATE OR REPLACE FUNCTION update\_discount\_and\_recalculate\_amount()

RETURNS TRIGGER AS $$

BEGIN

UPDATE Customers

SET Discount = NEW.Discount

WHERE CustomerID = (SELECT CustomerID FROM Purchases WHERE PurchaseID = NEW.PurchaseID);

UPDATE Purchases

SET Amount = (NEW.BookPrice \* NEW.Quantity) \* (1 - NEW.Discount / 100.0)

WHERE PurchaseID = NEW.PurchaseID;

RETURN NEW;

END;

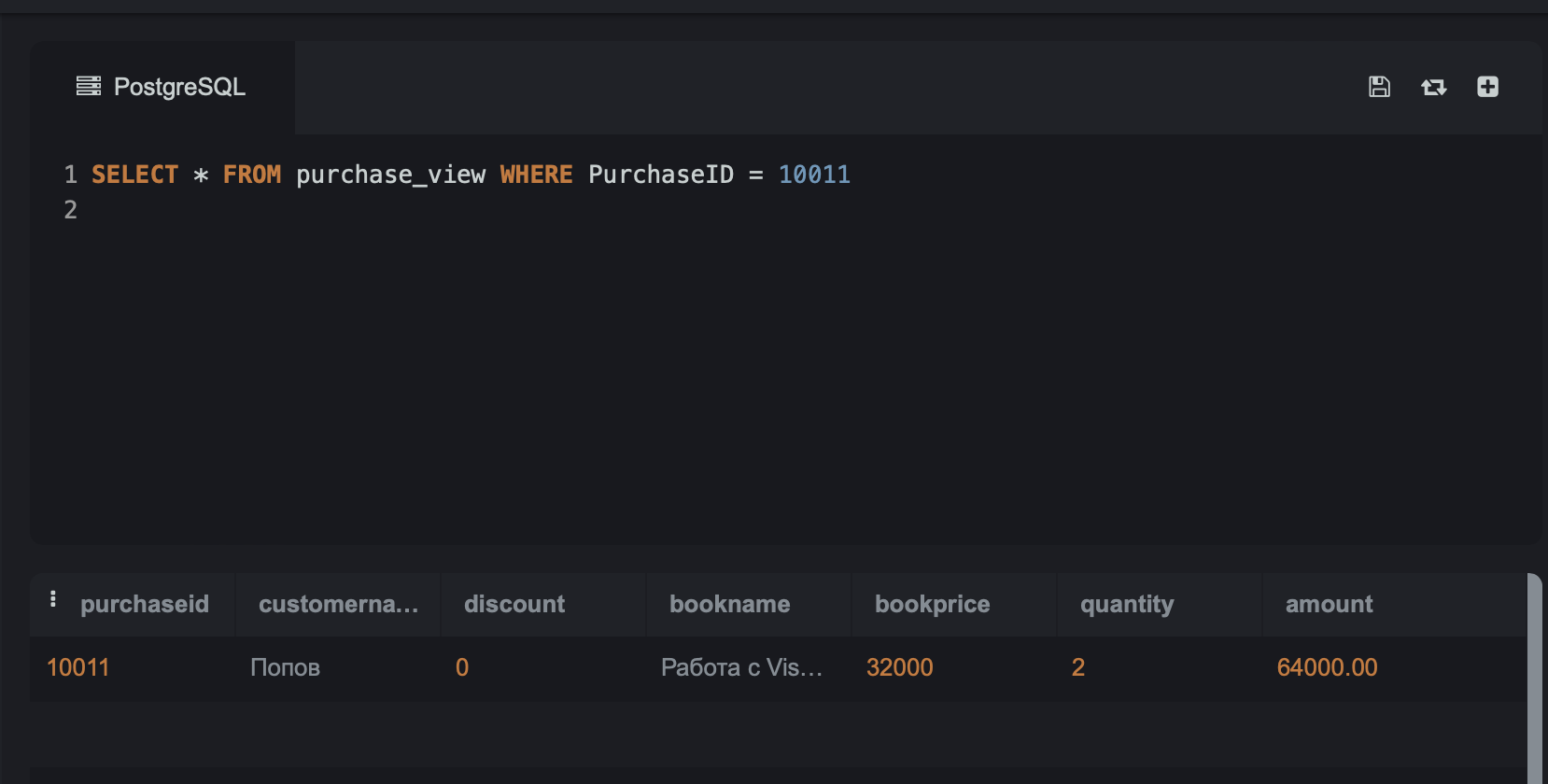
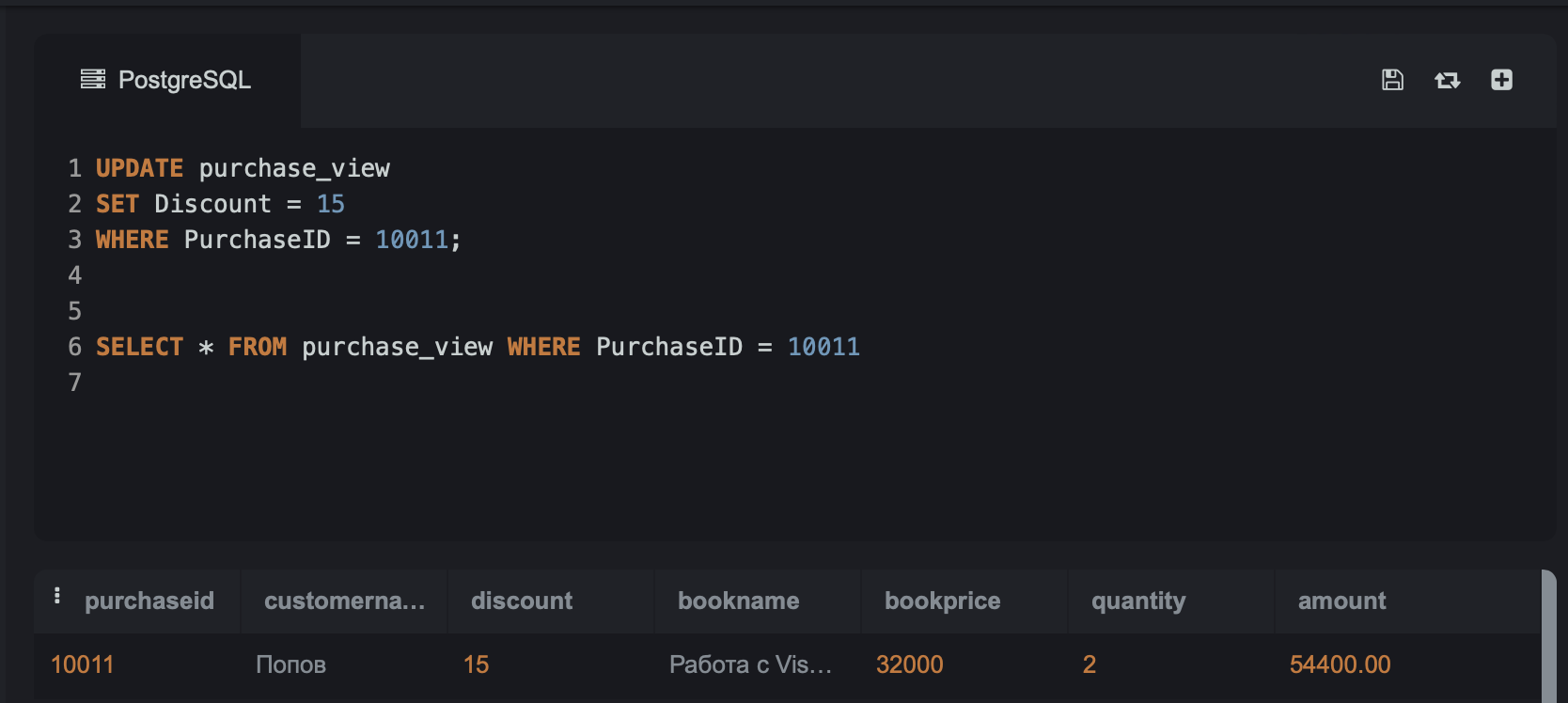
$$ LANGUAGE plpgsql;

CREATE TRIGGER tr\_update\_discount\_and\_recalculate\_amount

INSTEAD OF UPDATE ON purchase\_view

FOR EACH ROW

EXECUTE FUNCTION update\_discount\_and\_recalculate\_amount();  
  
Теперь можно изменить скидку покупателя и автоматически пересчитать стоимость покупки с помощью запроса

**Проверка:   
Предыдущее значение:   
  
  
Запрос + новое значение:  
**