#### ****Заголовок****

**Тема:** Работа с представлениями, функциями, процедурами и триггерами в PostgreSQL

**Цель:** Разработка и управление объектами базы данных PostgreSQL, такими как представления, функции, процедуры и триггеры. Рассмотрение возможных ошибок и способов их исправления.

### ****1. Создание представления (CREATE VIEW)****

**Описание:**  
Создается представление, объединяющее данные из таблиц Customer, Employee и Delivery с использованием соединений JOIN. Также добавляется вычисляемый столбец total\_weight, рассчитывающий общий вес товара.

**Команда для создания представления:**

CREATE VIEW DeliveryDetails AS

SELECT

Customer.customer\_name AS customer,

Employee.employee\_name AS employee,

Delivery.item\_name,

Delivery.item\_quantity,

Delivery.cost,

Delivery.item\_weight \* Delivery.item\_quantity AS total\_weight

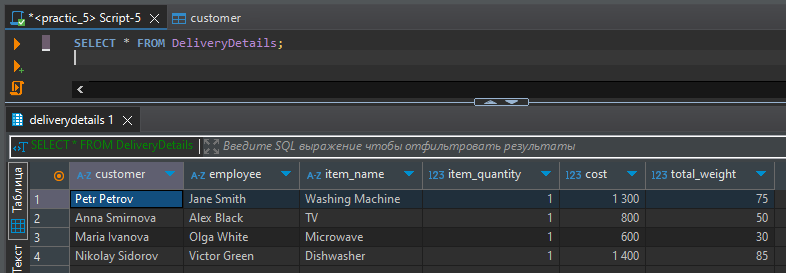
FROM Delivery

INNER JOIN Customer ON Delivery.customer\_id = Customer.customer\_id

INNER JOIN Employee ON Delivery.employee\_id = Employee.employee\_id;

**Команда для выгрузки данных из представления:**

SELECT \* FROM DeliveryDetails;



### ****2. Процедуры для вставки, обновления и удаления данных****

#### ****2.1 Процедура для вставки данных:****

**Описание:** Вставляет данные в таблицу Customer.

**Команда:**

CREATE OR REPLACE PROCEDURE InsertCustomer(

cname VARCHAR,

cphone VARCHAR,

caddress VARCHAR

)

LANGUAGE plpgsql

AS $$

BEGIN

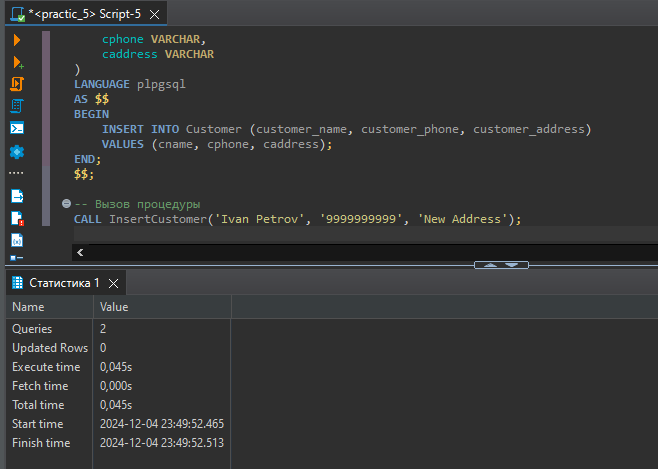
INSERT INTO Customer (customer\_name, customer\_phone, customer\_address)

VALUES (cname, cphone, caddress);

END;

$$;

CALL InsertCustomer('Ivan Petrov', '9999999999', 'New Address');



#### ****2.2 Процедура для обновления данных:****

**Описание:** Обновляет данные конкретной записи в таблице Customer.

**Команда:**

CREATE OR REPLACE PROCEDURE UpdateCustomer(

cid INT,

cname VARCHAR,

cphone VARCHAR

)

LANGUAGE plpgsql

AS $$

BEGIN

UPDATE Customer

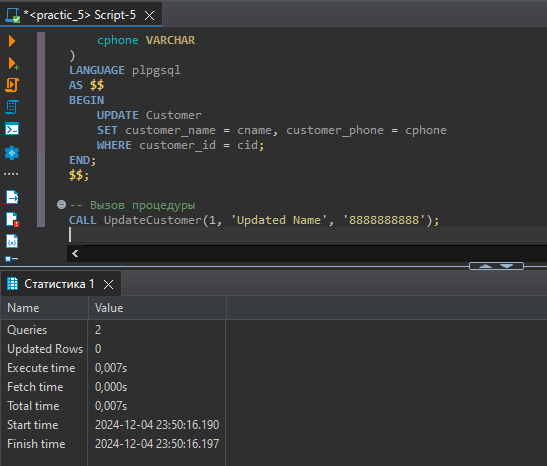
SET customer\_name = cname, customer\_phone = cphone

WHERE customer\_id = cid;

END;

$$;

CALL UpdateCustomer(1, 'Updated Name', '8888888888');



#### ****2.3 Процедура для удаления данных:****

**Описание:** Удаляет запись из таблицы Customer.

**Команда:**

CREATE OR REPLACE PROCEDURE DeleteCustomer(cid INT)

LANGUAGE plpgsql

AS $$

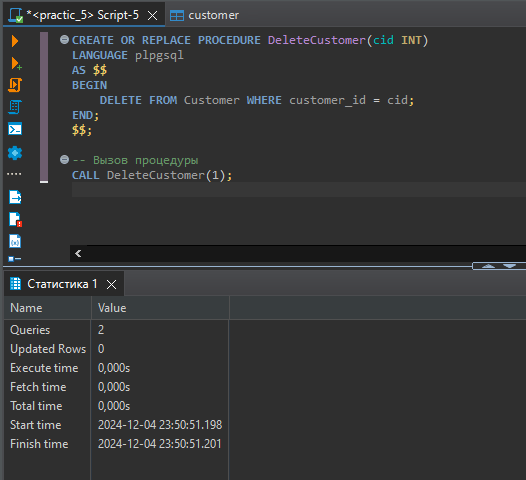
BEGIN

DELETE FROM Customer WHERE customer\_id = cid;

END;

$$;

CALL DeleteCustomer(1);



### ****3. Скалярная функция****

**Описание:**  
Создается функция, которая возвращает самое длинное значение строки из столбца customer\_name таблицы Customer.

**Команда:**

CREATE OR REPLACE FUNCTION GetLongestValue()

RETURNS VARCHAR

LANGUAGE plpgsql

AS $$

DECLARE

longest\_value VARCHAR;

BEGIN

SELECT customer\_name

FROM Customer

ORDER BY LENGTH(customer\_name) DESC

LIMIT 1

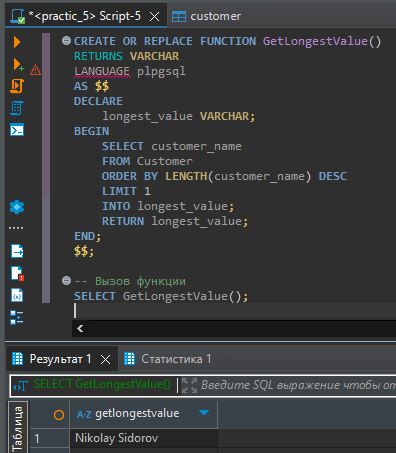
INTO longest\_value;

RETURN longest\_value;

END;

$$;

SELECT GetLongestValue();



### ****4. Табличная функция****

**Описание:**  
Создается функция, которая возвращает первые 10 записей из таблицы Delivery, отсортированных по стоимости в убывающем порядке.

**Команда:**

CREATE OR REPLACE FUNCTION GetTop10Deliveries()

RETURNS TABLE(order\_id INT, cost NUMERIC)

LANGUAGE plpgsql

AS $$

BEGIN

RETURN QUERY

SELECT order\_id, cost

FROM Delivery

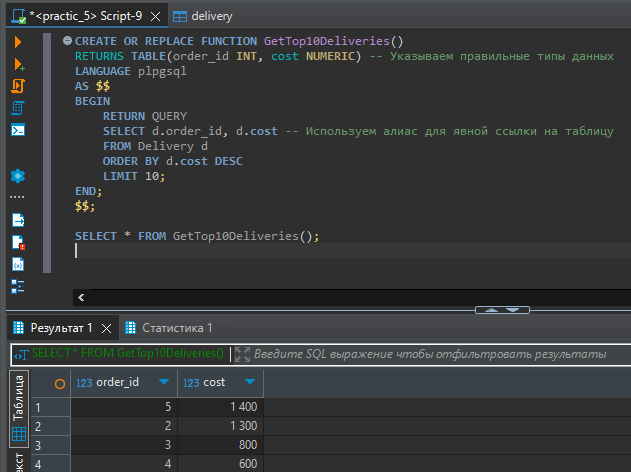
ORDER BY cost DESC

LIMIT 10;

END;

$$;

SELECT \* FROM GetTop10Deliveries();



### ****5. Функция с возвращаемым типом VOID****

**Описание:**  
Создается функция, которая удаляет запись из таблицы Customer по указанному ID.

**Команда:**

CREATE OR REPLACE FUNCTION DeleteCustomerVoid(cid INT)

RETURNS VOID

LANGUAGE plpgsql

AS $$

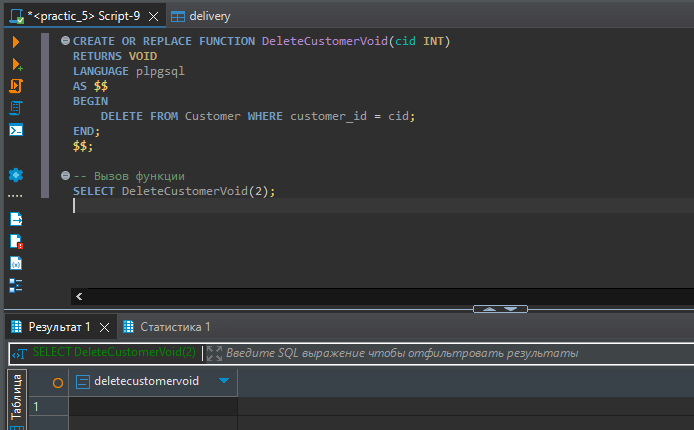
BEGIN

DELETE FROM Customer WHERE customer\_id = cid;

END;

$$;

SELECT DeleteCustomerVoid(2);



### ****6. Триггер BEFORE INSERT****

**Описание:**  
Триггер предотвращает вставку данных в таблицу Delivery и выводит сообщение об ошибке.

**Команда:**

CREATE OR REPLACE FUNCTION BeforeInsertTrigger()

RETURNS TRIGGER

LANGUAGE plpgsql

AS $$

BEGIN

RAISE EXCEPTION 'Вставка в таблицу запрещена!';

RETURN NULL;

END;

$$;

CREATE TRIGGER PreventInsert

BEFORE INSERT ON Delivery

FOR EACH ROW

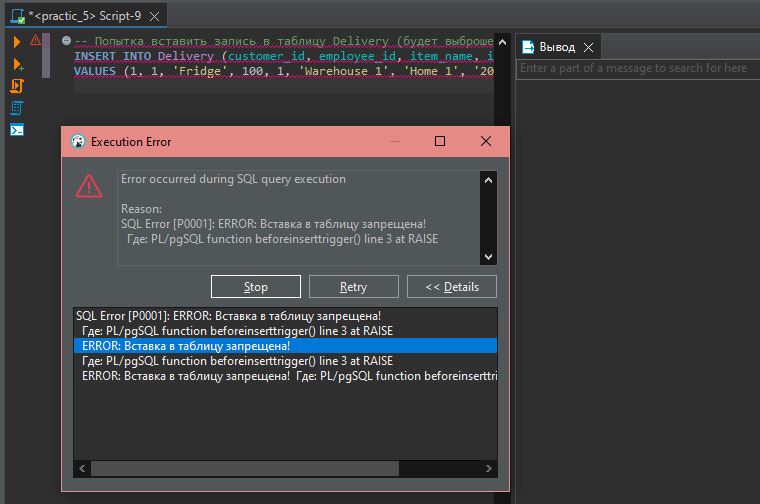
EXECUTE FUNCTION BeforeInsertTrigger();

**Команда для проверки работы триггера:**

INSERT INTO Delivery (customer\_id, employee\_id, item\_name, item\_weight, item\_quantity, origin\_address, destination\_address, shipment\_date, delivery\_date, status, document\_type, document\_number, cost)

VALUES (1, 1, 'Fridge', 100, 1, 'Warehouse 1', 'Home 1', '2024-09-01', '2024-09-02', 'Delivered', 'Invoice', 'INV123', 1500);

**Ожидаемое поведение:** Исключение с сообщением: Вставка в таблицу запрещена!.



### ****7. Триггер AFTER DELETE****

**Описание:**  
Триггер выводит ID удаленной записи после удаления из таблицы Delivery.

**Команда:**

CREATE OR REPLACE FUNCTION AfterDeleteTrigger()

RETURNS TRIGGER

LANGUAGE plpgsql

AS $$

BEGIN

RAISE NOTICE 'Удалена запись с ID %', OLD.order\_id;

RETURN OLD;

END;

$$;

CREATE TRIGGER NotifyDelete

AFTER DELETE ON Delivery

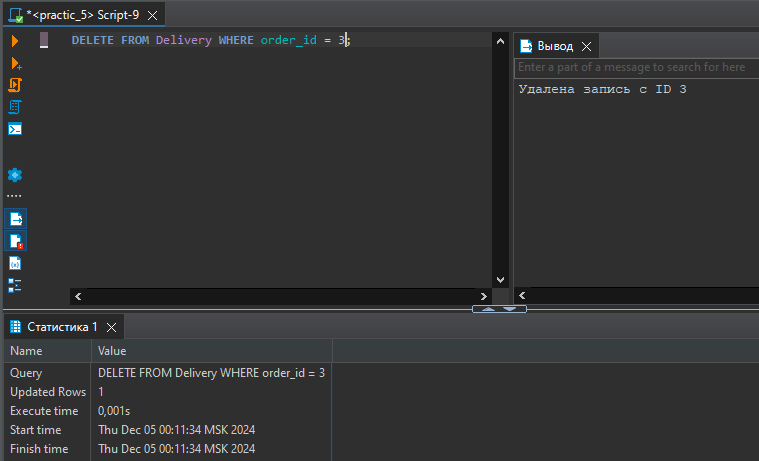
FOR EACH ROW

EXECUTE FUNCTION AfterDeleteTrigger();

**Команда для проверки работы триггера:**

DELETE FROM Delivery WHERE order\_id = 1;

**Ожидаемое поведение:** Сообщение NOTICE: Удалена запись с ID 1.



### ****8.Заключение:****

Этот отчет охватывает создание и тестирование всех объектов базы данных PostgreSQL, включая представления, функции, процедуры и триггеры. Рассмотрены примеры вызовов, возможные ошибки и их исправления.