# Практическая работа 6

## Тема: Работа с временными таблицами, секционированием и оконными функциями

## Введение

Цель данной практической работы — выполнение заданий с использованием временных таблиц, секционирования, оконных функций и CTE. В процессе выполнения будут рассмотрены различные ошибки и их исправления, связанные с особенностями PostgreSQL.

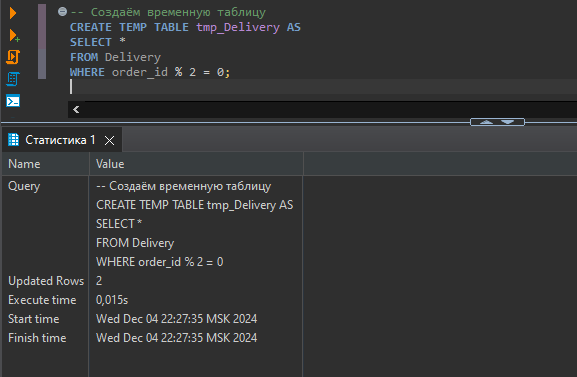
## 1. Создание временной таблицы

Для выполнения задания была выбрана таблица Delivery. Временная таблица tmp\_Delivery создаётся с использованием команды CREATE TEMP TABLE и содержит все чётные записи из исходной таблицы.

Команда:

CREATE TEMP TABLE tmp\_Delivery AS  
SELECT \*  
FROM Delivery  
WHERE order\_id % 2 = 0;

Место для скриншота выполнения команды:



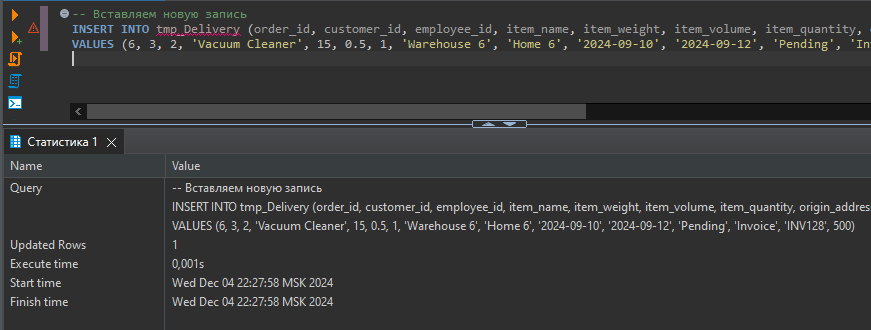
## 2. Добавление записи в временную таблицу

Добавляем новую запись в таблицу tmp\_Delivery. Так как таблица временная, запись будет доступна только в текущей сессии.

Команда:

INSERT INTO tmp\_Delivery (order\_id, customer\_id, employee\_id, item\_name, item\_weight, item\_volume,   
 item\_quantity, origin\_address, destination\_address, shipment\_date, delivery\_date,   
 status, document\_type, document\_number, cost)  
VALUES (6, 3, 2, 'Vacuum Cleaner', 15, 0.5, 1, 'Warehouse 6', 'Home 6',   
 '2024-09-10', '2024-09-12', 'Pending', 'Invoice', 'INV128', 500);

Место для скриншота выполнения команды:



## 3. Ошибка при попытке доступа к временной таблице из другой сессии

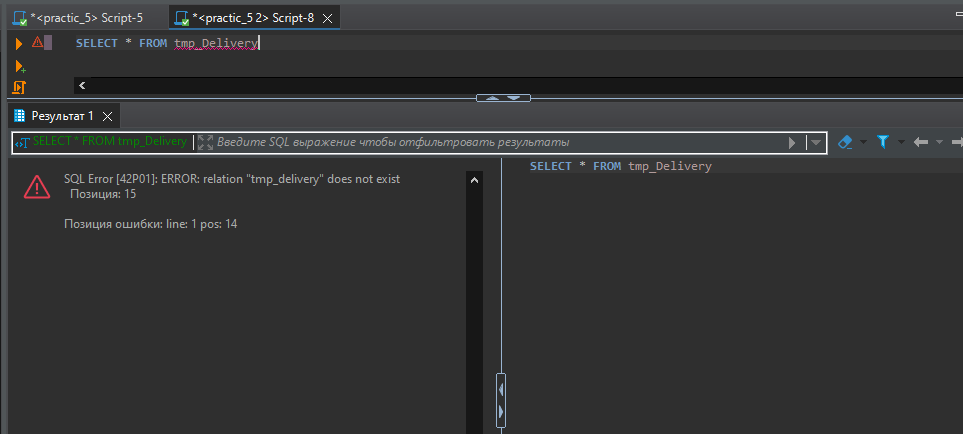
При попытке выполнить SELECT из временной таблицы tmp\_Delivery из другой сессии возникает ошибка, так как временные таблицы доступны только в рамках текущей сессии.

Пример команды, вызывающей ошибку:

SELECT \* FROM tmp\_Delivery;

Ожидаемая ошибка:

SQL Error [42P01]: ERROR: relation "tmp\_Delivery" does not exist



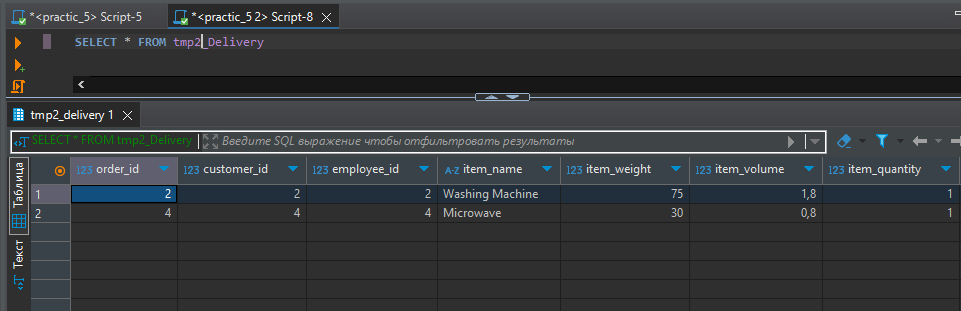
Для создания таблицы, доступной из других сессий, используем UNLOGGED TABLE:

CREATE UNLOGGED TABLE tmp2\_Delivery AS  
SELECT \*  
FROM Delivery  
WHERE order\_id % 2 = 0;

Добавление записи в новую таблицу:

INSERT INTO tmp2\_Delivery (order\_id, customer\_id, employee\_id, item\_name, item\_weight, item\_volume,   
 item\_quantity, origin\_address, destination\_address, shipment\_date, delivery\_date,   
 status, document\_type, document\_number, cost)  
VALUES (7, 4, 3, 'Air Conditioner', 50, 2.5, 1, 'Warehouse 7', 'Home 7',   
 '2024-09-11', '2024-09-14', 'Delivered', 'Invoice', 'INV129', 2000);

Место для скриншота выполнения команды:



## 4. Копирование таблицы и добавление столбца для секционирования

Создаём таблицу Customer\_section с добавлением столбца log\_date, которая будет разделена на секции по диапазонам дат.

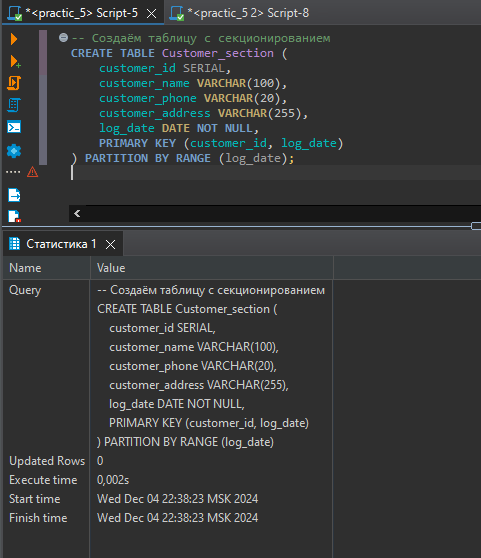
Команда:

CREATE TABLE Customer\_section (  
 customer\_id SERIAL,  
 customer\_name VARCHAR(100),  
 customer\_phone VARCHAR(20),  
 customer\_address VARCHAR(255),  
 log\_date DATE NOT NULL,  
 PRIMARY KEY (customer\_id, log\_date)  
) PARTITION BY RANGE (log\_date);

Создание секции для диапазона дат:

CREATE TABLE Customer\_section\_2024  
PARTITION OF Customer\_section  
FOR VALUES FROM ('2024-01-01') TO ('2024-12-31');

Место для скриншота выполнения команды:



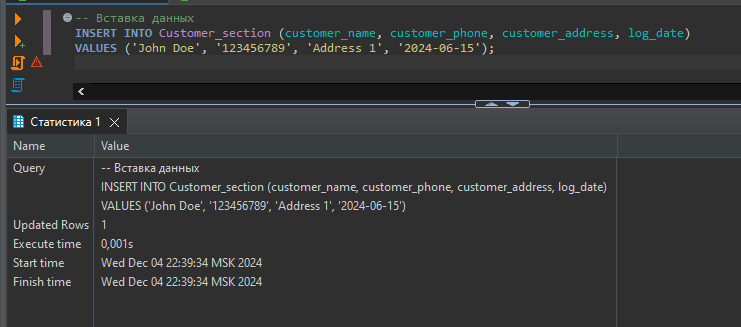
## 5. Создание новой секции таблицы

Создаём новую секцию таблицы Customer\_section для другого диапазона дат. Это позволяет работать с данными, относящимися к определённым временным периодам, более эффективно.

Команда:

CREATE TABLE Customer\_section\_2023  
PARTITION OF Customer\_section  
FOR VALUES FROM ('2023-01-01') TO ('2023-12-31');

Место для скриншота выполнения команды:

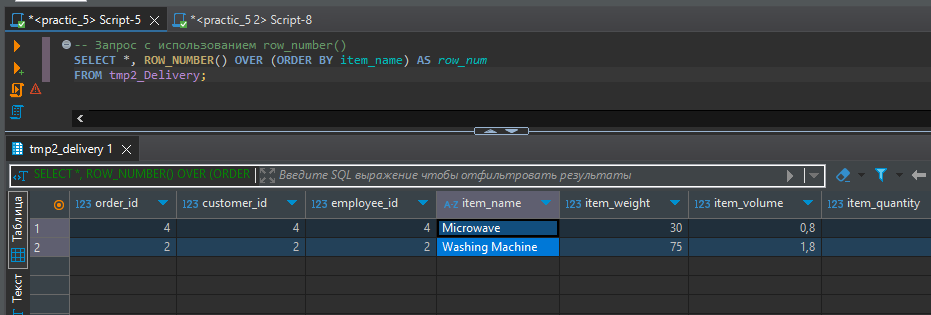


## 6. Запрос с использованием оконной функции

Выполняем запрос с использованием оконной функции row\_number(), которая нумерует строки в результате запроса на основе сортировки по строковому атрибуту item\_name.

Команда:  
SELECT \*, ROW\_NUMBER() OVER (ORDER BY item\_name) AS row\_num  
FROM tmp\_Delivery;

Место для скриншота выполнения команды:

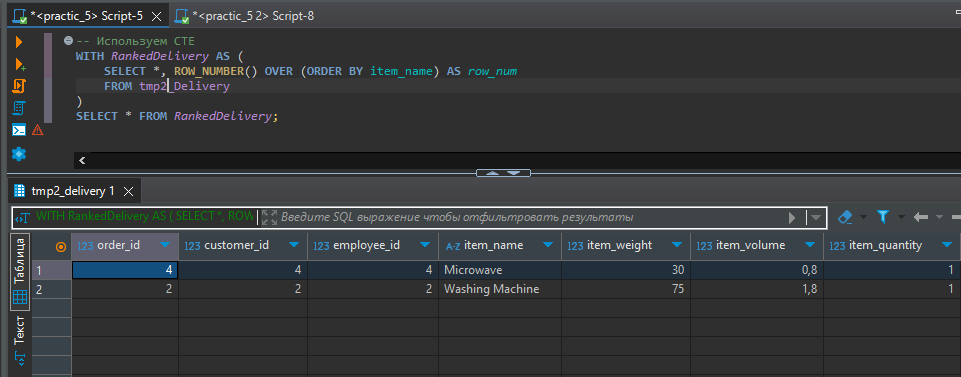


## 7. Использование CTE

Создаём CTE (Common Table Expression) для выполнения запроса с нумерацией строк, а затем извлекаем данные из этого CTE.

Команда:  
WITH RankedDelivery AS (  
 SELECT \*, ROW\_NUMBER() OVER (ORDER BY item\_name) AS row\_num  
 FROM tmp\_Delivery  
)  
SELECT \* FROM RankedDelivery;

Место для скриншота выполнения команды:

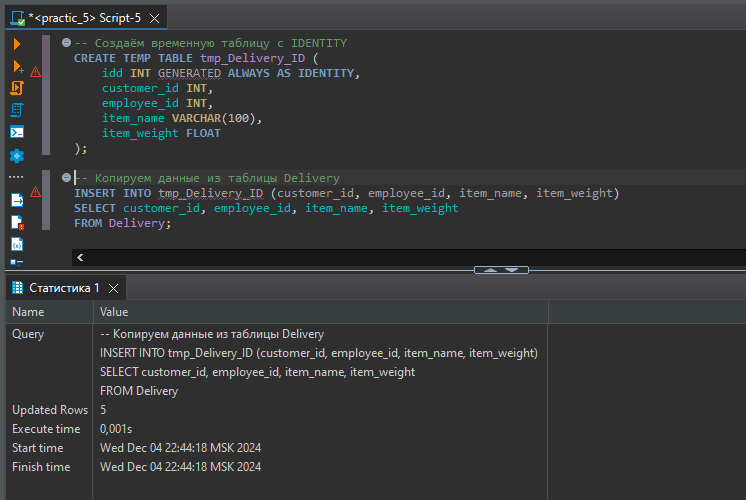


## 8. Создание временной таблицы с вычисляемым столбцом

Создаём временную таблицу tmp\_Delivery\_ID с вычисляемым столбцом idd, который автоматически генерирует уникальные значения.

Команда:  
CREATE TEMP TABLE tmp\_Delivery\_ID (  
 idd INT GENERATED ALWAYS AS IDENTITY,  
 customer\_id INT,  
 employee\_id INT,  
 item\_name VARCHAR(100),  
 item\_weight FLOAT  
);  
  
INSERT INTO tmp\_Delivery\_ID (customer\_id, employee\_id, item\_name, item\_weight)  
SELECT customer\_id, employee\_id, item\_name, item\_weight  
FROM Delivery;

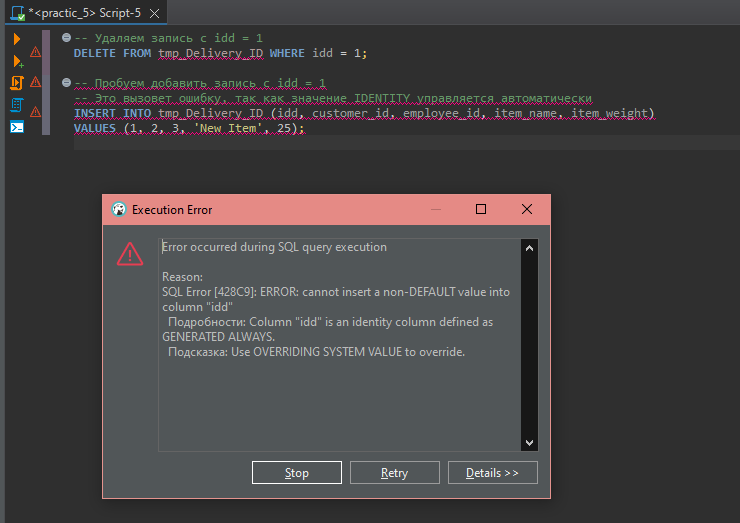
Место для скриншота выполнения команды:



## 9. Удаление записи и проверка значения IDENTITY

Удаляем запись с idd = 1 и проверяем, возможно ли вручную вставить запись с таким же значением. PostgreSQL не позволяет явно указывать значения для столбца IDENTITY, если он автоматически управляется.

Команды:  
-- Удаление записи  
DELETE FROM tmp\_Delivery\_ID WHERE idd = 1;  
  
-- Попытка вставки записи с idd = 1 (вызовет ошибку)  
INSERT INTO tmp\_Delivery\_ID (idd, customer\_id, employee\_id, item\_name, item\_weight)  
VALUES (1, 2, 3, 'New Item', 25);

  
  
-- Правильная вставка (значение idd генерируется автоматически)  
INSERT INTO tmp\_Delivery\_ID (customer\_id, employee\_id, item\_name, item\_weight)  
VALUES (2, 3, 'New Item', 25);

