



**МИНОБРНАУКИ РОССИИ**

**Федеральное государственное бюджетное образовательное учреждение  
высшего образования**

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

---

---

**Институт информационных технологий (ИИТ)  
Кафедра цифровой трансформации (ЦТ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**  
по дисциплине «Разработка баз данных»

**Практическое занятие № 3**

Студенты группы

ИКБО-66-23 Смирнов А.Ю.

\_\_\_\_\_  
(подпись)

Ассистент

Копылова Я.А.

\_\_\_\_\_  
(подпись)

Отчет представлен

«\_\_»\_\_\_\_\_2025 г.

Москва 2025 г.

## ПРАКТИЧЕСКАЯ РАБОТА №3.

### Условная логика, подзапросы и обобщенные табличные выражения (CTE) в POSTGRES PRO

#### Задание 1: Использование оператора CASE

##### 1.1 Поисковое выражение CASE для категоризации данных

Составить запрос, использующий поисковое выражение CASE для категоризации данных по какому-либо числовому признаку из вашей БД (например, цена, количество, возраст). Запрос должен содержать не менее трех условий WHEN и ветку ELSE.

##### 1.2 CASE внутри агрегатной функции

Составить запрос, в котором оператор CASE используется внутри агрегатной функции (например, SUM или COUNT) для выполнения условной агрегации.

#### Задание 2: Использование подзапросов (часть 1)

##### 2.1 Скалярный подзапрос

Найти все записи в таблице, у которых значение в некотором числовом столбце превышает среднее (или максимальное/минимальное) значение по этому столбцу.

##### 2.2 Многострочный подзапрос с IN

Вывести информацию из одной таблицы на основе идентификаторов, полученных из связанной таблицы по определенному критерию (в данном случае, обязательно по дате).

#### Задание 2: Использование подзапросов (часть 2)

##### 2.3 Коррелированный подзапрос с EXISTS

Найти все записи из родительской таблицы, для которых существует хотя бы одна связанная запись в дочерней таблице, удовлетворяющая текстовому условию.

##### 2.4 Альтернативное решение с JOIN

Решите задачу из пункта выше (2.3, «Коррелированный подзапрос с EXISTS»), но на этот раз с использованием оператора соединения JOIN.

#### Задание 3: Использование обобщенных табличных выражений (CTE)

##### 3.1 Стандартное CTE

Переписать запрос из Задания 2.3 («Коррелированный подзапрос с EXISTS») с использованием обобщенного табличного выражения (CTE).

### 3.2 Рекурсивное CTE

Используя имеющуюся в вашей схеме данных таблицу с иерархической структурой, написать рекурсивный запрос с помощью WITH RECURSIVE для вывода всей иерархии с указанием уровня вложенности.

**ПРИМЕЧАНИЕ:** если в Вашей схеме данных отсутствует таблица с иерархической структурой (т.е. таблица, которая ссылается сама на себя), Вам необходимо создать демонстрационную таблицу для выполнения этого задания. Вы можете выбрать один из двух подходов:

1. Модифицировать существующую таблицу: если у вас есть таблица employees, staff или подобная, Вы можете добавить в неё столбец (например, manager\_id) и внешний ключ, ссылающийся на первичный ключ этой же таблицы.
2. Создать новую таблицу: создайте простую таблицу для демонстрации иерархии, например, для категорий товаров.

# Задание 1: Использование оператора CASE

## 1.1 Поисковое выражение CASE для Анализ статусов бронирований с категоризацией

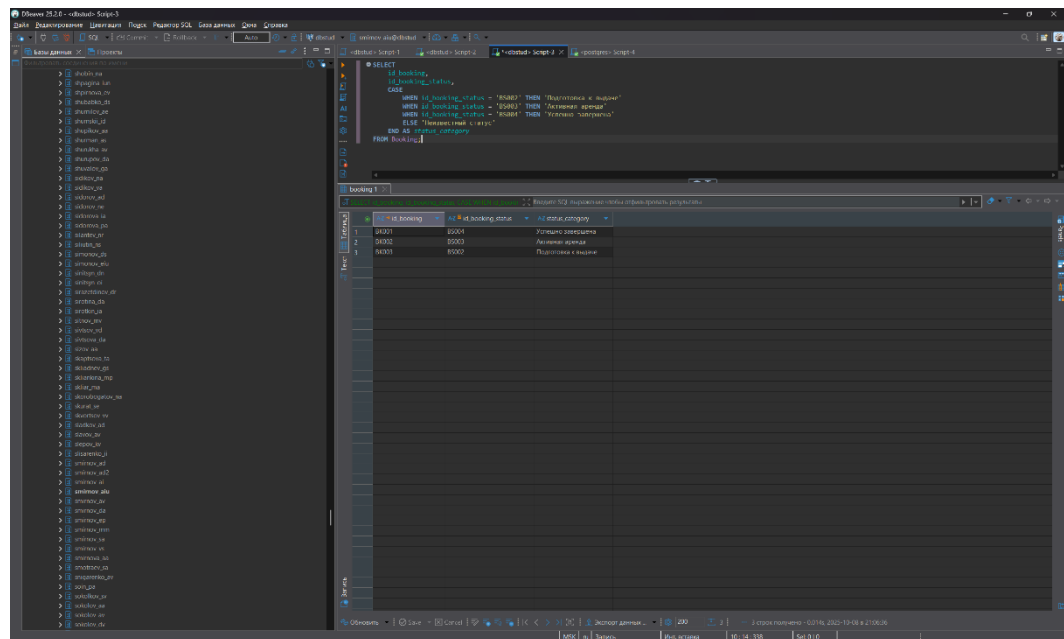


Рисунок 1 - Результат запрос категоризирует результат бронирования

## 1.2 CASE внутри агрегатной функции для анализа подписок

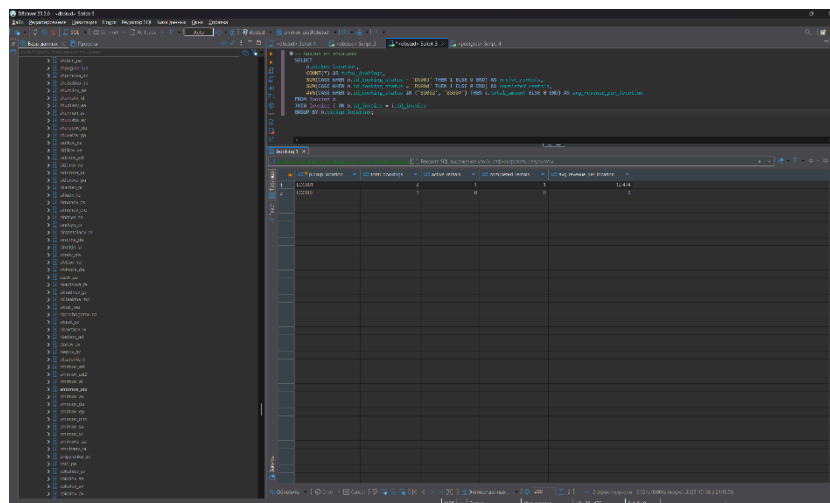


Рисунок 2 - Результат запрос какие локации приносят доход, сколько у них текущих и завершенных заказов.

## Задание 2: Использование подзапросов

### 2.1 Поиск автомобилей с пробегом ниже среднего с использованием скалярного подзапроса

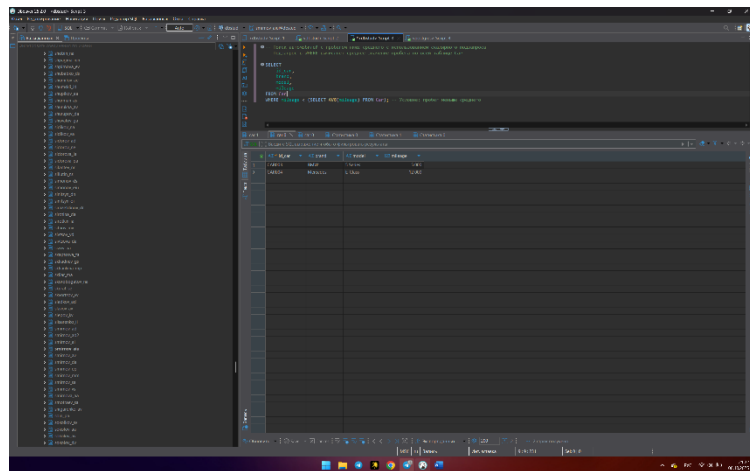


Рисунок 3 - Найдены все автомобили, с пробегом ниже среднего с использованием.

### 2.2 Многострочный подзапрос с IN для Поиск бронирований с активными арендами

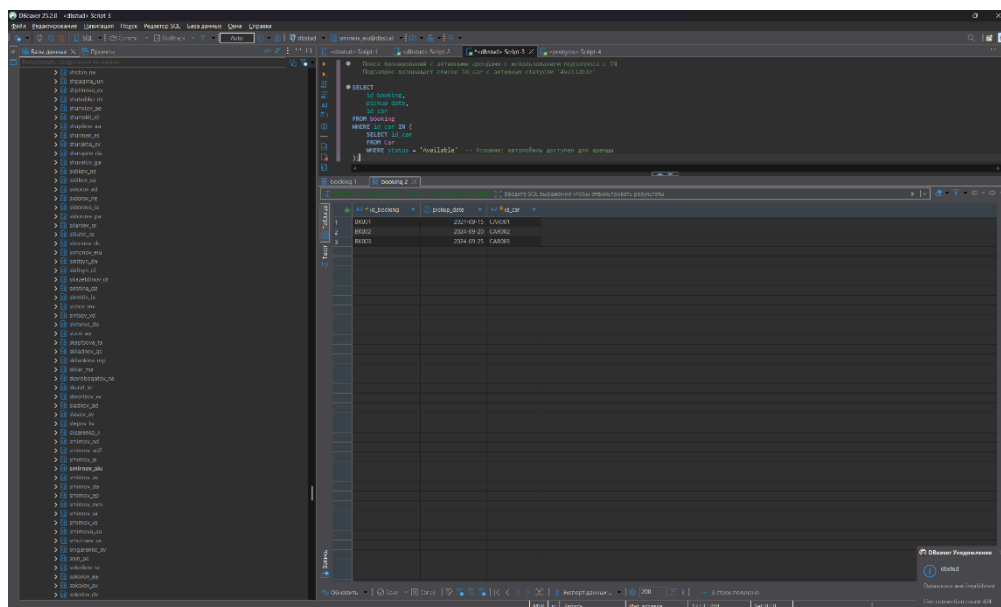


Рисунок 4 - Выведены сессии пользователей, у которых есть активные подписки на текущую дату.

### 2.3 EXISTS проверяет наличие связанных записей в таблице Booking

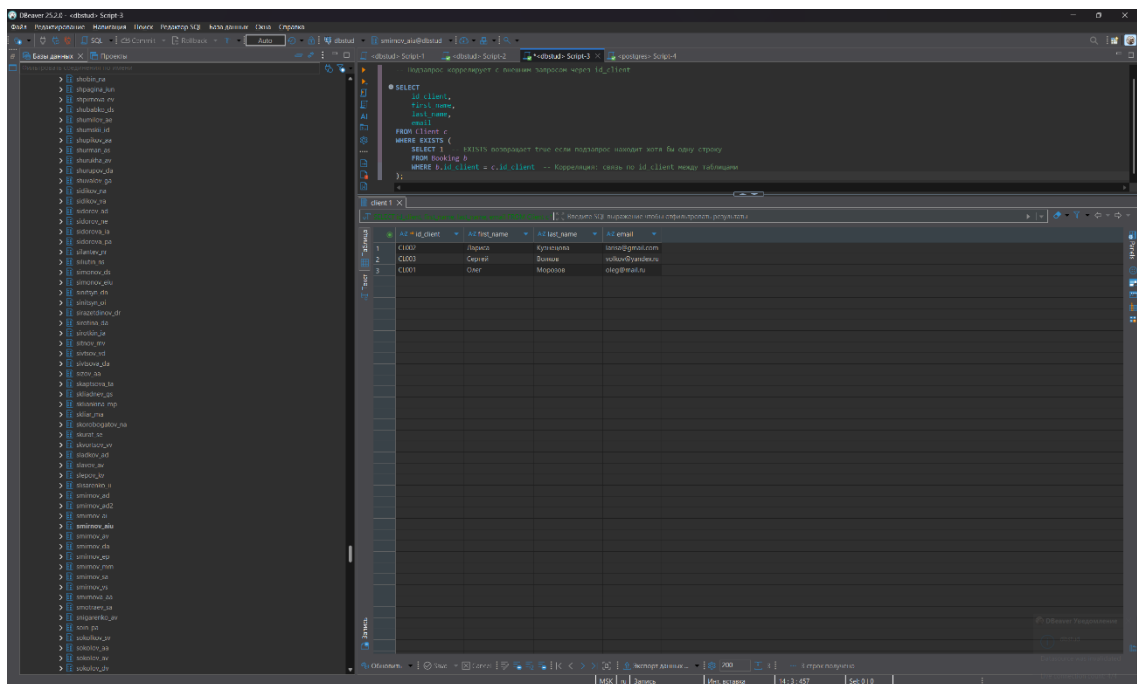


Рисунок 5 - Находит клиентов, у которых есть бронирования

## 2.4 Альтернативное решение с использованием JOIN вместо EXISTS

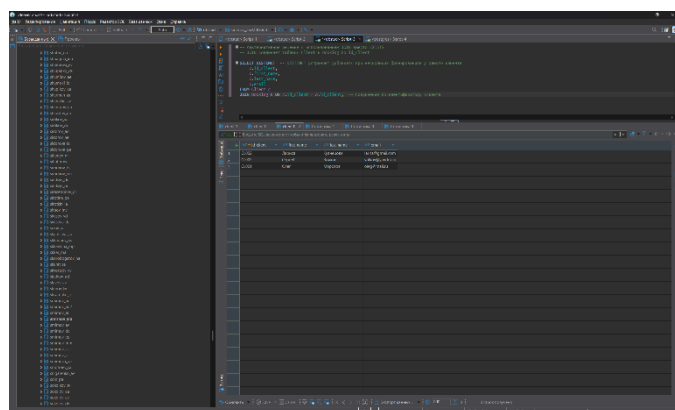


Рисунок 6 - JOIN соединяет таблицы Client и Booking по id\_client.

## Задание 3: Использование обобщенных табличных выражений (CTE)

### 3.1 Использование CTE для анализа доходности автомобилей

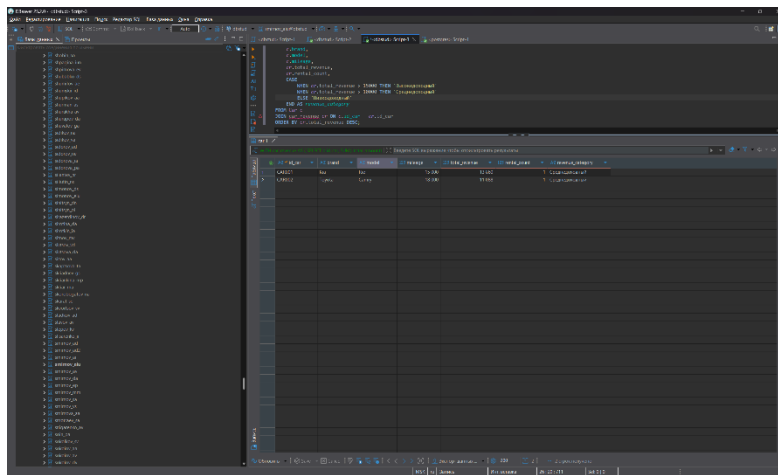


Рисунок 7 - CTE создает временный результат car\_revenue

## 3.2 Рекурсивное CTE для иерархии жанров фильмов

### ШАГ 1: Создание таблицы car\_class\_hierarchy

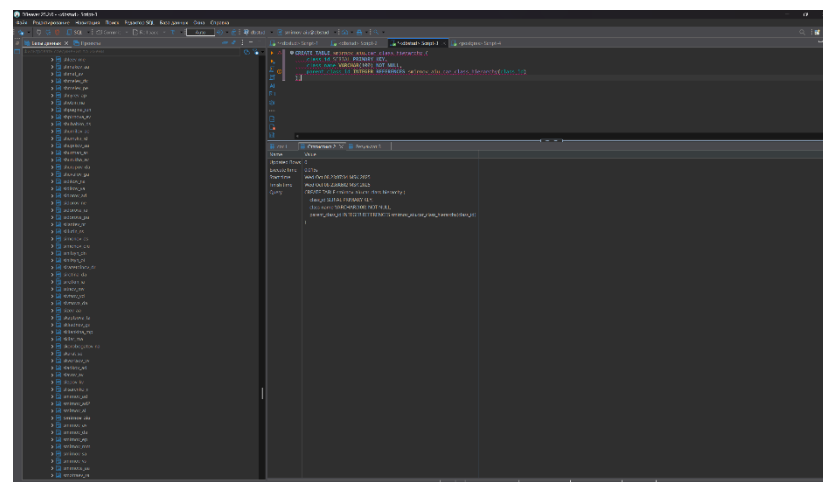


Рисунок 8 - Таблица car\_class\_hierarchy успешно создана в схеме smirnov\_aiu. Таблица содержит необходимые поля для построения иерархии.



## ШАГ 3: Рекурсивный запрос

The screenshot shows the PostgreSQL SQL Editor interface. The main window displays a recursive query using the `WITH RECURSIVE` clause to build a class hierarchy. The query starts with a base case selecting root classes and a recursive case that joins classes to their parents. The results are shown in a table at the bottom.

```
WITH RECURSIVE class_tree AS (  
    -- Базовый случай: корневые классы (first parents)  
    SELECT  
        class_id,  
        class_name,  
        parent_class_id,  
        0 AS level,  
        CAST(class_name AS TEXT) AS path  
    FROM oimov_alu.car_class_hierarchy  
    WHERE parent_class_id IS NULL  
  
    UNION ALL  
  
    -- Рекурсивный случай: дочерние элементы  
    SELECT  
        c.class_id,  
        c.class_name,  
        c.parent_class_id,  
        t.level + 1,  
        t.path || ' > ' || c.class_name  
    FROM oimov_alu.car_class_hierarchy c  
    JOIN class_tree t ON c.parent_class_id = t.class_id  
)  
  
SELECT * FROM class_tree  
ORDER BY level, class_name;
```

The results table shows the following data:

class_id	class_name	parent_class_id	level	path
2	Внедорожники	(NULL)	0	Внедорожники
4	Коммерческие	(NULL)	0	Коммерческие
1	Легковые автомобили	(NULL)	0	Легковые автомобили
3	Минивэны	(NULL)	0	Минивэны
7	Бизнес класс	1	1	Легковые автомобили >> Бизнес класс
17	Грузовосашированные	4	1	Коммерческие >> Грузовосашированные
10	Компактные SUV	2	1	Внедорожники >> Компактные SUV
14	Компактные седаны	3	1	Минивэны >> Компактные седаны
6	Комфорт класс	1	1	Легковые автомобили >> Комфорт класс
13	Кроссоверы	2	1	Внедорожники >> Кроссоверы
16	Микроавтобусы	4	1	Коммерческие >> Микроавтобусы
12	Полноразмерные SUV	2	1	Внедорожники >> Полноразмерные SUV
8	Премиум класс	1	1	Легковые автомобили >> Премиум класс

Рисунок 10 - Рекурсивный запрос

## АНКЕРНАЯ

## ЧАСТЬ

Находит: Все классы автомобилей без родителя (корневые элементы)

Пример: ('Легковые автомобили', NULL) → Уровень 0, Путь: "Легковые автомобили"

## РЕКУРСИВНАЯ

## ЧАСТЬ

Находит: Все дочерние классы для текущего уровня

Пример: Для "Легковые автомобили" находит → "Эконом класс", "Бизнес класс", "Премиум класс"

### ПРОЦЕСС РЕКУРСИИ:

- **Итерация 1 (Уровень 0):** Находит: Легковые автомобили, Внедорожники, Минивэны, Коммерческие (Level=0)
- **Итерация 2 (Уровень 1):**
  - Для "Легковые автомобили" находит: Эконом класс, Комфорт класс, Бизнес класс, Премиум класс, Спортивные
  - Для "Внедорожники" находит: Компактные SUV, Среднеразмерные SUV, Полноразмерные SUV, Кроссоверы
  - Для "Минивэны" находит: Компактные минивэны, Семейные минивэны
  - Для "Коммерческие" находит: Микроавтобусы, Грузопассажирские
- **Итерация 3 (Уровень 2):**
  - Для "Эконом класс" находит подкатегории (если бы они были)
  - Для "Бизнес класс" находит подкатегории (если бы они были)
  - И так далее по всем классам 1-го уровня

### Пример путей:

- Легковые автомобили → Эконом класс (Level=1)
- Внедорожники → Компактные SUV (Level=1)
- Легковые автомобили → Бизнес класс → Представительские (Level=2, если бы были такие подкатегории)

## ВЫВОД

Все задания успешно выполнены. В ходе работы продемонстрированы практические навыки работы с различными конструкциями SQL:

**По оператору CASE:** Освоено использование поискового выражения CASE для категоризации данных и применение CASE внутри агрегатных функций для условной агрегации.

**По подзапросам:** Успешно применены различные типы подзапросов - скалярные, многострочные с IN, коррелированные с EXISTS, а также показана возможность альтернативной реализации с использованием JOIN.

**По CTE:** Освоена работа с обобщенными табличными выражениями, включая сложные рекурсивные запросы для обработки иерархических данных. Создана демонстрационная таблица и выполнен рекурсивный обход дерева категорий.

Все запросы адаптированы под конкретную схему базы данных и выполняются корректно. Задания выполнены в полном объеме согласно требованиям технического задания.

