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 № 4

Студенты группы *ИКБО-66-23 Смирнов А.Ю.*
.

(подпись)

Ассистент *Копылова Я.А.*

(подпись)

Отчет представлен «___»_____2025 г.

Москва 2025 г.

Постановка задачи:

Для выполнения практической работы необходимо последовательно выполнить четыре задачи, используя собственную базу данных. Все примеры в данном документе основаны на демонстрационной базе данных «Аптека», содержащей таблицы manufacturers (производители), medicines (лекарства) и sales (продажи).

Ваша задача — адаптировать каждую из поставленных задач к логической структуре и предметной области вашей базы данных. Приведенные ниже формулировки и последующие примеры кода служат шаблоном для понимания, какой тип аналитического запроса требуется составить.

Задание №1: использование ранжирующих функций

Для каждой основной «родительской» сущности в вашей БД (например, производитель, категория товара, автор) определить три наиболее значимых по некоторому числовому признаку дочерних сущности (например, три самых дорогих товара, три самые популярные книги по количеству продаж). В результирующей таблице должны быть указаны идентификатор группы, идентификатор дочерней сущности, её числовой признак и ранг. Для расчёта ранга использовать функцию RANK() или DENSE_RANK().

Задание №2: использование агрегатных оконных функций

Для ключевой сущности, имеющей транзакции по времени (например, товар, услуга), рассчитать нарастающий итог (кумулятивную сумму) по некоторому показателю (например, объем продаж, количество заказов) с разбивкой по временным периодам (месяцам или годам). Отчёт должен содержать идентификатор сущности (id/название/…), временной период, сумму за период и кумулятивную сумму.

Задание №3: использование функции смещения

Провести сравнительный анализ общих показателей по периодам. Для каждого периода (например, месяца), начиная со второго, необходимо вывести

общий показатель за текущий период и аналогичный показатель за предыдущий период в одной строке. Это позволит наглядно оценить динамику. Необходимо использовать функцию LAG().

Задание №4: построение сводной таблицы

Создать сводный отчет, который агрегирует некоторый числовой показатель для основной сущности по категориям, представленным в виде столбцов.

Решение:

Задание №1: использование ранжирующих функций

- Запрос с DENSE_RANK для трех сотрудника с самой высокой зарплатой

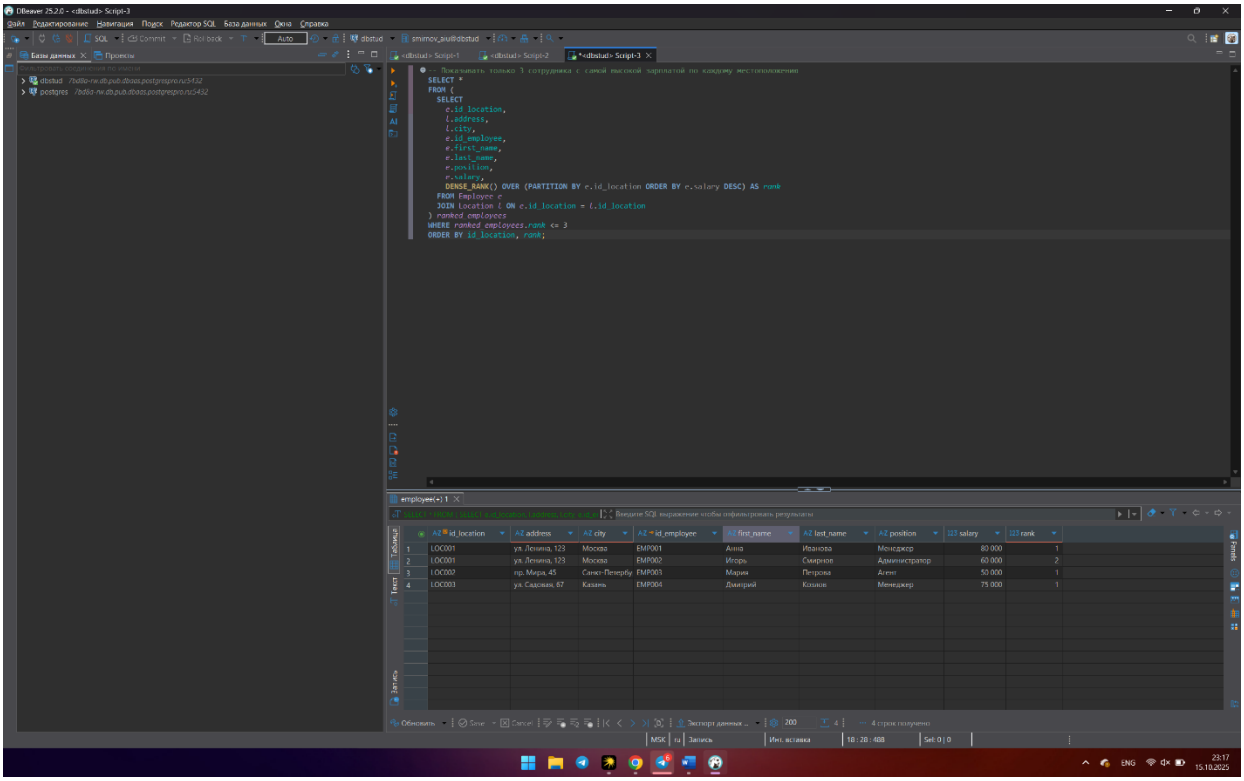

Рисунок 1 — Запрос DENSE_RANK

Задание №2: использование агрегатных оконных функций

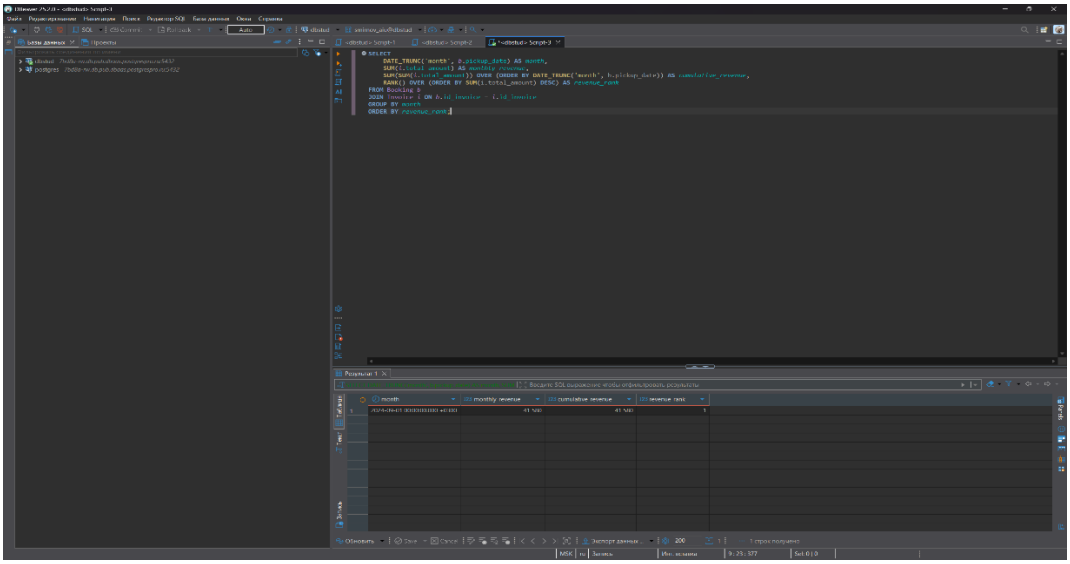

Рисунок 2 — Общая кумулятивная выручка по месяцам

Задание №3: использование функции смещения Lag.

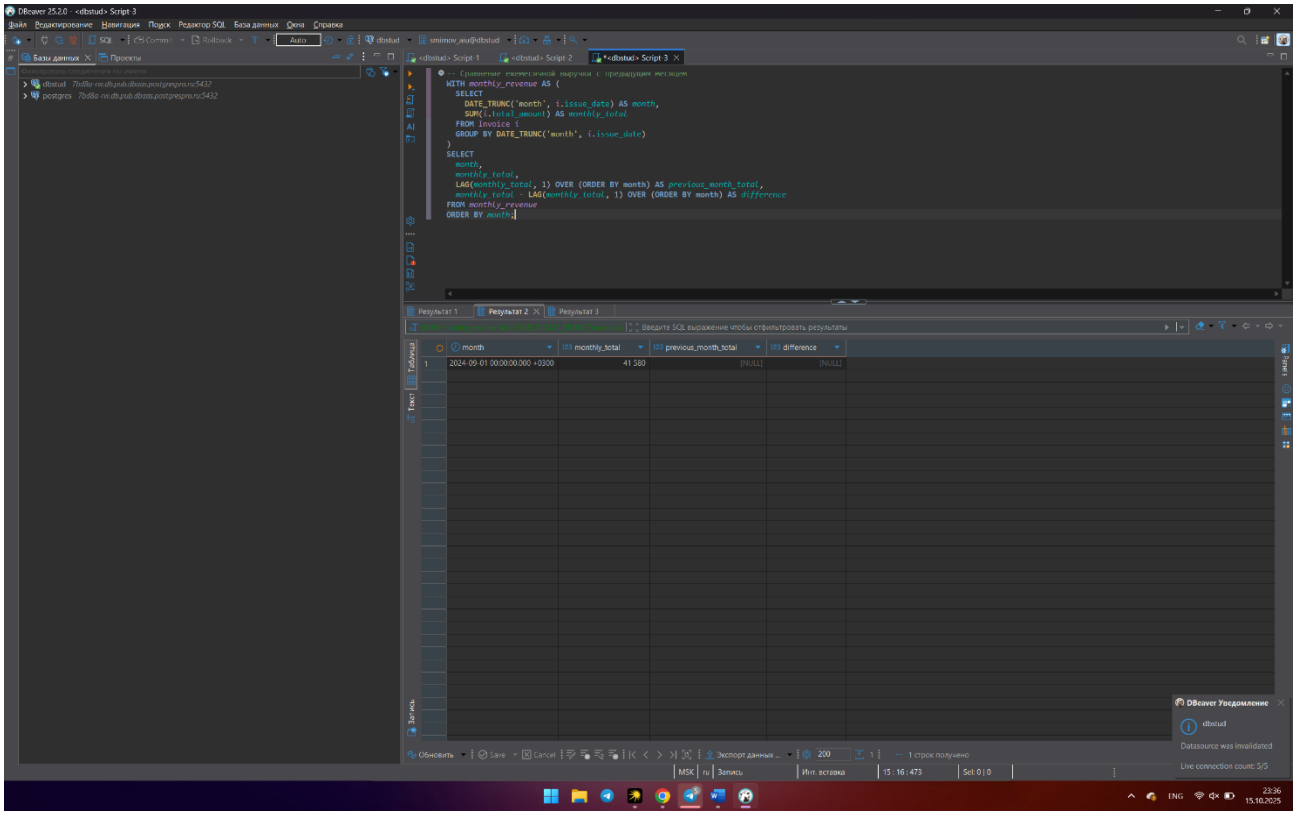

Рисунок 3 — Таблица Сравнение ежемесячной выручки с предыдущим месяцем

Задание №4: построение сводной таблицы

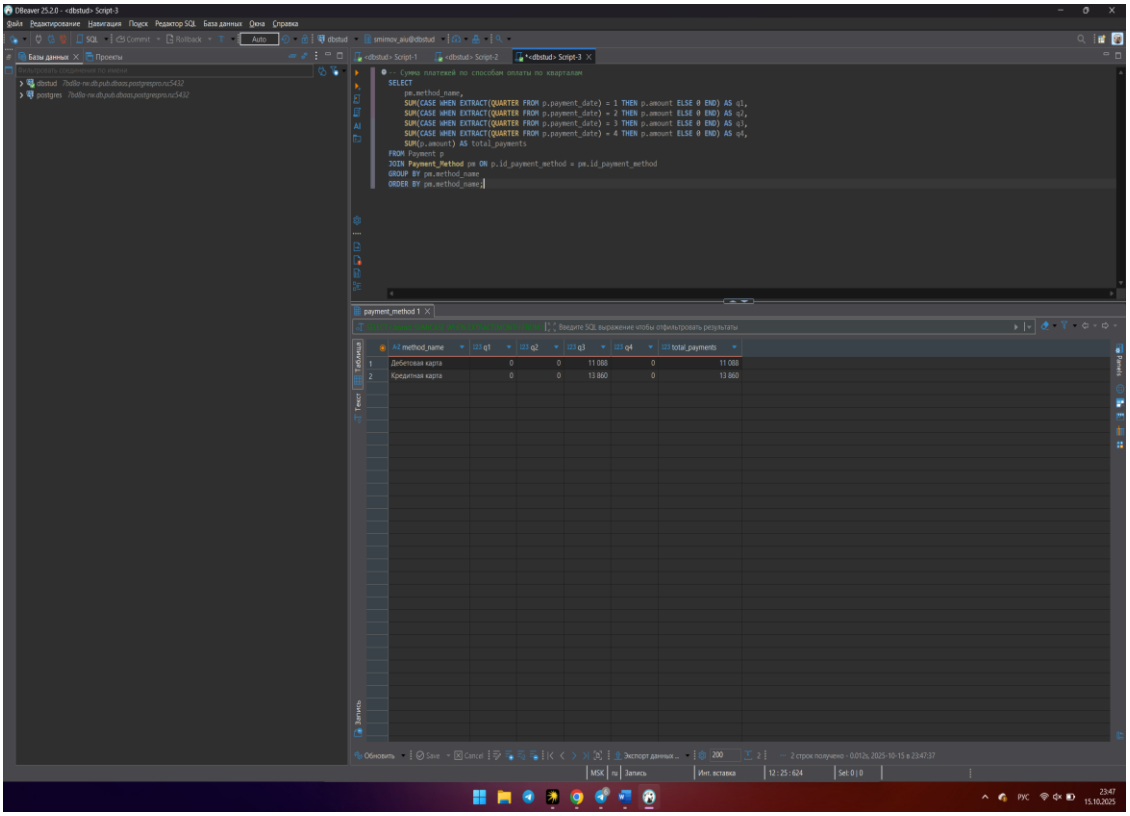

Рисунок 4 — Количество бронирований по локациям по месяцам (SUM + CASE)

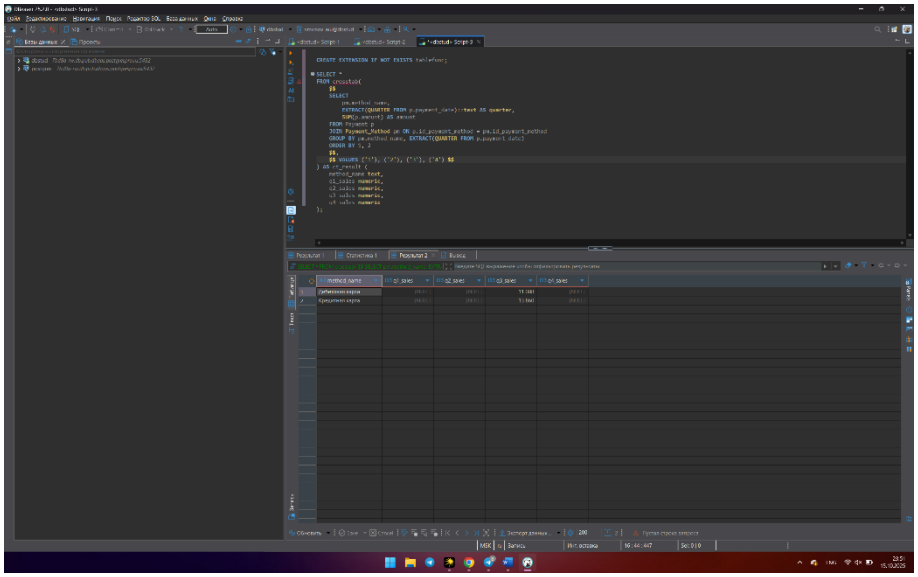

Рисунок 4 — Метод с crosstab()

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Объясните принципиальное различие в результате работы запроса с SUM(...) OVER (PARTITION BY...) и запроса с SUM(...) и GROUP BY. В какой ситуации каждый из них предпочтителен?

Принципиальное различие между SUM(...) OVER (PARTITION BY...) и SUM(...) ... GROUP BY заключается в том, что GROUP BY агрегирует строки и возвращает одну строку на каждую группу, тогда как оконная функция SUM() OVER вычисляет результат для каждой строки, не сворачивая данные. То есть GROUP BY сокращает количество строк, а оконная функция просто добавляет вычисленное значение в каждую из них. Использование GROUP BY предпочтительно, когда нужен итог по группам, а оконная функция — когда важно сохранить детали каждой записи.

2. В чем разница между функциями RANK() и DENSE_RANK() при наличии одинаковых значений в столбце, по которому идет сортировка? Приведите бизнес-сценарий, в котором использование DENSE_RANK() будет более корректным.

Разница между RANK() и DENSE_RANK() проявляется при наличии одинаковых значений в сортируемом столбце. Функция RANK() присваивает одинаковые ранги для одинаковых значений, но пропускает следующий номер (например, 1, 2, 2, 4). DENSE_RANK() также присваивает одинаковый ранг, но не делает пропуска (1, 2, 2, 3). В бизнес-сценариях, где важно иметь последовательную нумерацию без пропусков, например при формировании списка лучших трёх товаров по продажам, DENSE_RANK() будет более корректным.

3. Почему нельзя использовать оконную функцию непосредственно в предложении WHERE (например, WHERE price > AVG(price) OVER (...))? Опишите, как можно обойти это ограничение, используя CTE или подзапрос.

Оконные функции нельзя использовать непосредственно в предложении WHERE, потому что WHERE выполняется **до** вычисления оконных выражений. На момент фильтрации окно ещё не рассчитано, и СУБД не знает его значение. Чтобы обойти это ограничение, можно вынести вычисление оконной функции в подзапрос или обобщённое табличное выражение (CTE), а затем применить WHERE уже к результату этого подзапроса, где значение функции доступно.

4. Какие два основных SQL-запроса требуются в качестве аргументов для функции crosstab? Каково назначение каждого из них?

Функция crosstab требует два запроса в качестве аргументов. Первый запрос возвращает тройки значений — категорию строк (например, сотрудник), категорию столбцов (например, квартал) и числовой показатель (например, сумма продаж). Второй запрос задаёт возможные категории для столбцов (например, 1, 2, 3, 4 для кварталов). Первый формирует данные, второй определяет структуру итоговой таблицы.