**Отчет по практической работе №7.1 и №7.2**

**Выборка данных из базы данных. Выполнение запроса с оконными функциями**

**1. Информация о студенте**

*ФИО: Швецов Демьян Андреевич  
Группа: ЭФБО-09-23*

**Практическая работа № 7.1**

**2. Задание + 3. Решение + 4. Результат**

1. Создать таблицу «Студент». Структура таблицы: ФИО, Дисциплина, Группа, Балл.

CREATE TABLE Студент (

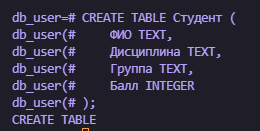
    ФИО TEXT,

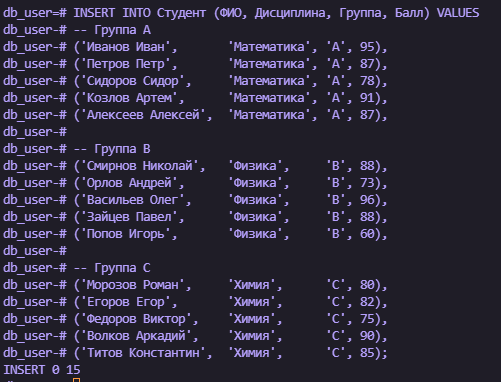
    Дисциплина TEXT,

    Группа TEXT,

    Балл INTEGER

);





1. Вывести список студентов и разницу между баллами студента и максимальным баллом в каждой строке.

SELECT

    ФИО,

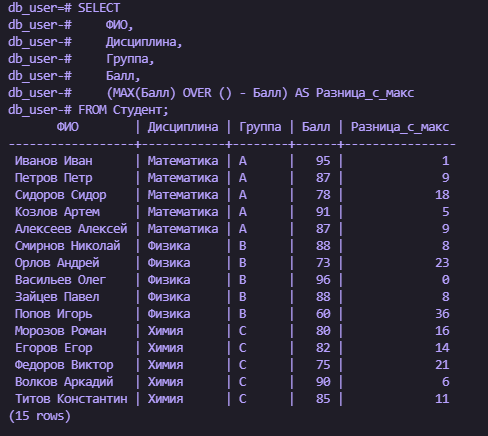
    Дисциплина,

    Группа,

    Балл,

    (MAX(Балл) OVER () - Балл) AS Разница\_с\_макс

FROM Студент;



2. Вывести список студентов и процентное соотношение к среднему баллу в каждой строке.

SELECT

    ФИО,

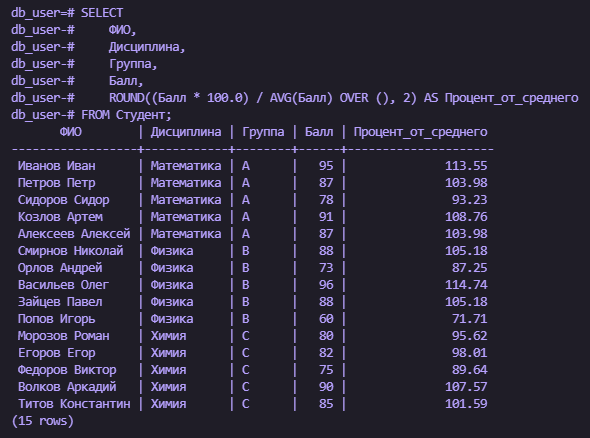
    Дисциплина,

    Группа,

    Балл,

    ROUND((Балл \* 100.0) / AVG(Балл) OVER (), 2) AS Процент\_от\_среднего

FROM Студент;



3. Вывести список студентов и суммарный балл в группе в каждой строке, отсортировать по баллу в группе.

SELECT

    ФИО,

    Дисциплина,

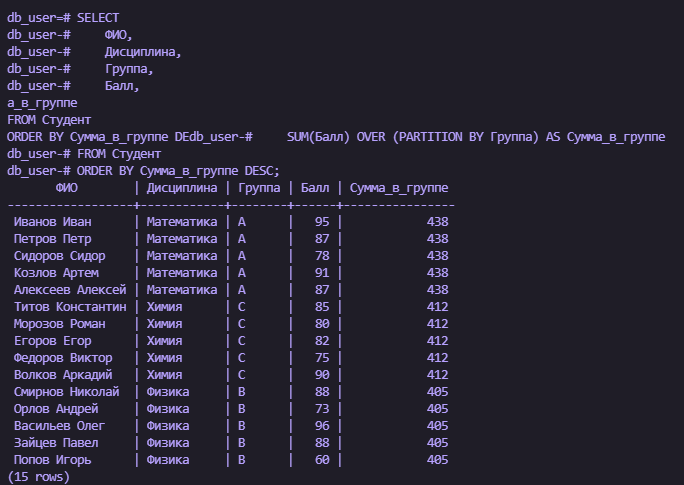
    Группа,

    Балл,

    SUM(Балл) OVER (PARTITION BY Группа) AS Сумма\_в\_группе

FROM Студент

ORDER BY Сумма\_в\_группе DESC;



4. Вывести список студентов и номер строки при сортировке по фамилиям в обратном алфавитном порядке.

SELECT

    ФИО,

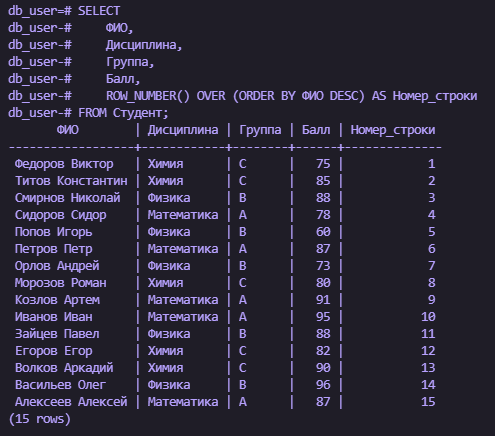
    Дисциплина,

    Группа,

    Балл,

    ROW\_NUMBER() OVER (ORDER BY ФИО DESC) AS Номер\_строки

FROM Студент;



5. Вывести список студентов, номер строки внутри группы и количество студентов в группе при сортировке по баллам по убыванию.

SELECT

    ФИО,

    Дисциплина,

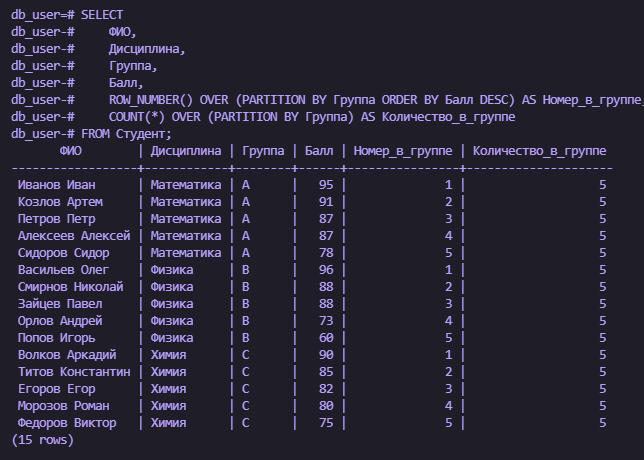
    Группа,

    Балл,

    ROW\_NUMBER() OVER (PARTITION BY Группа ORDER BY Балл DESC) AS Номер\_в\_группе,

    COUNT(\*) OVER (PARTITION BY Группа) AS Количество\_в\_группе

FROM Студент;



6. Вывести список студентов и ранг по баллам. Результат отсортировать по фамилии в алфавитном порядке.

SELECT

    ФИО,

    Дисциплина,

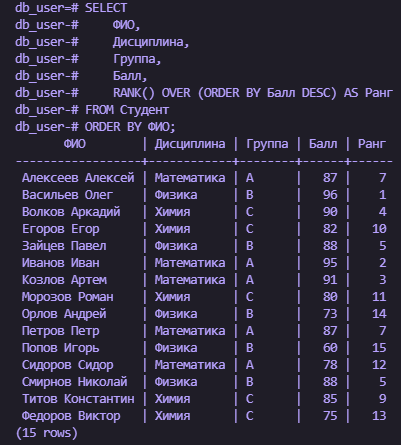
    Группа,

    Балл,

    RANK() OVER (ORDER BY Балл DESC) AS Ранг

FROM Студент

ORDER BY ФИО;



7. Вывести список студентов и разницу с баллами студента, находящегося выше на три позиции при сортировке по возрастанию баллов.

SELECT

    ФИО,

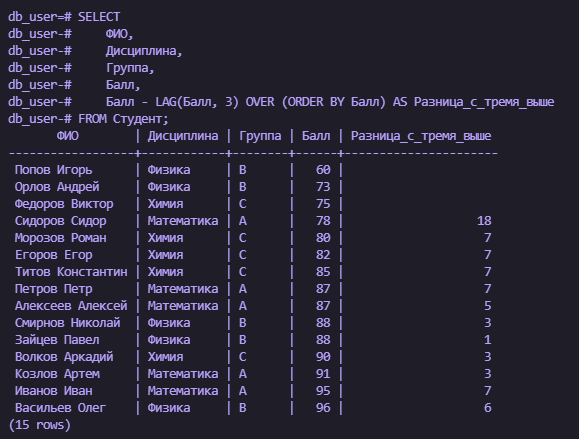
    Дисциплина,

    Группа,

    Балл,

    Балл - LAG(Балл, 3) OVER (ORDER BY Балл) AS Разница\_с\_тремя\_выше

FROM Студент;



8. Вывести список студентов и разницу с баллами следующего студента при сортировке по убыванию баллов, значение по умолчанию использовать 0.

SELECT

    ФИО,

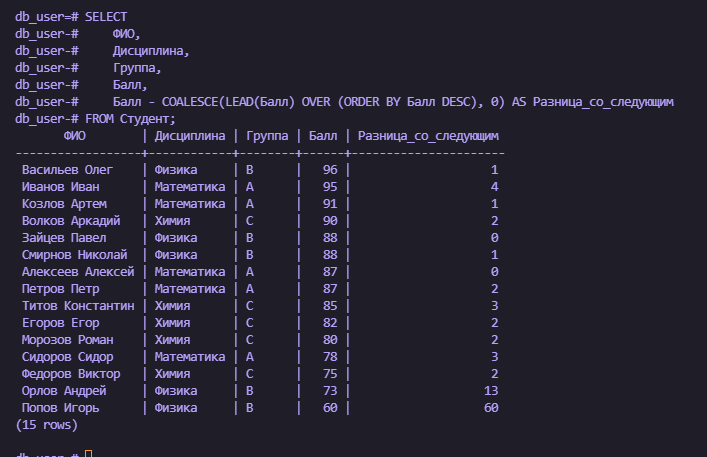
    Дисциплина,

    Группа,

    Балл,

    Балл - COALESCE(LEAD(Балл) OVER (ORDER BY Балл DESC), 0) AS Разница\_со\_следующим

FROM Студент;



9. Вывести список студентов и разницу с баллами последнего студента в группе при сортировке по убыванию баллов.

SELECT

    ФИО,

    Дисциплина,

    Группа,

    Балл,

    Балл - FIRST\_VALUE(Балл) OVER (

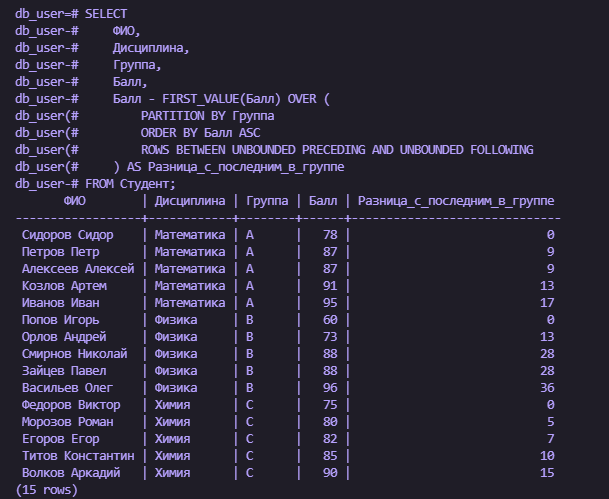
        PARTITION BY Группа

        ORDER BY Балл ASC

        ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

    ) AS Разница\_с\_последним\_в\_группе

FROM Студент;



10. Вывести список студентов и сжатый ранг по баллам в каждой группе. Результат отсортировать по фамилии в алфавитном порядке.

SELECT

    ФИО,

    Дисциплина,

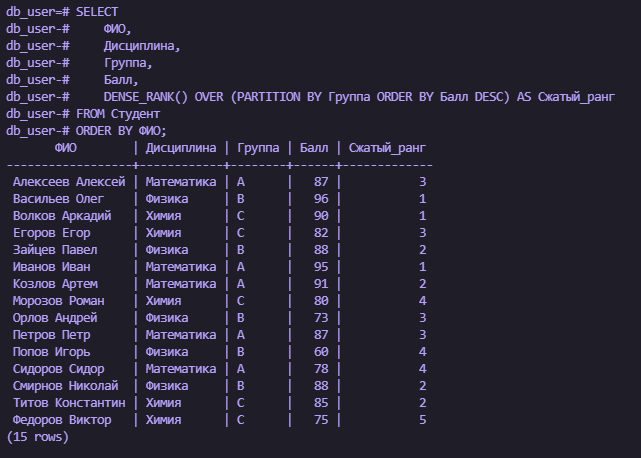
    Группа,

    Балл,

    DENSE\_RANK() OVER (PARTITION BY Группа ORDER BY Балл DESC) AS Сжатый\_ранг

FROM Студент

ORDER BY ФИО;



**Практическая работа № 7.2**

**2. Задание + 3. Решение + 4. Результат**

1. Вывести адрес квартир и разницу между стоимостью квартиры и максимальной стоимостью в каждом районе.

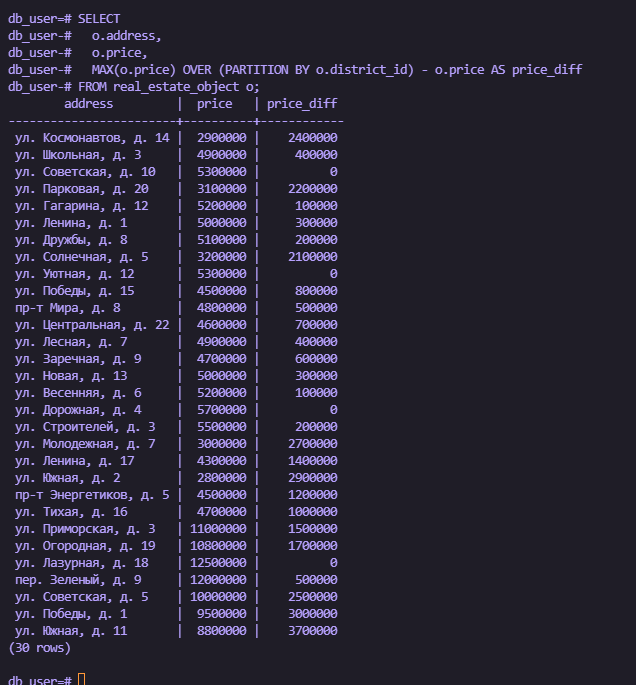
SELECT

  o.address,

  o.price,

  MAX(o.price) OVER (PARTITION BY o.district\_id) - o.price AS price\_diff

FROM real\_estate\_object o;



2. Вывести адреса объектов недвижимости и процентное соотношение стоимости 1м2 к средней стоимости 1м2 по каждому типу объекта недвижимости.

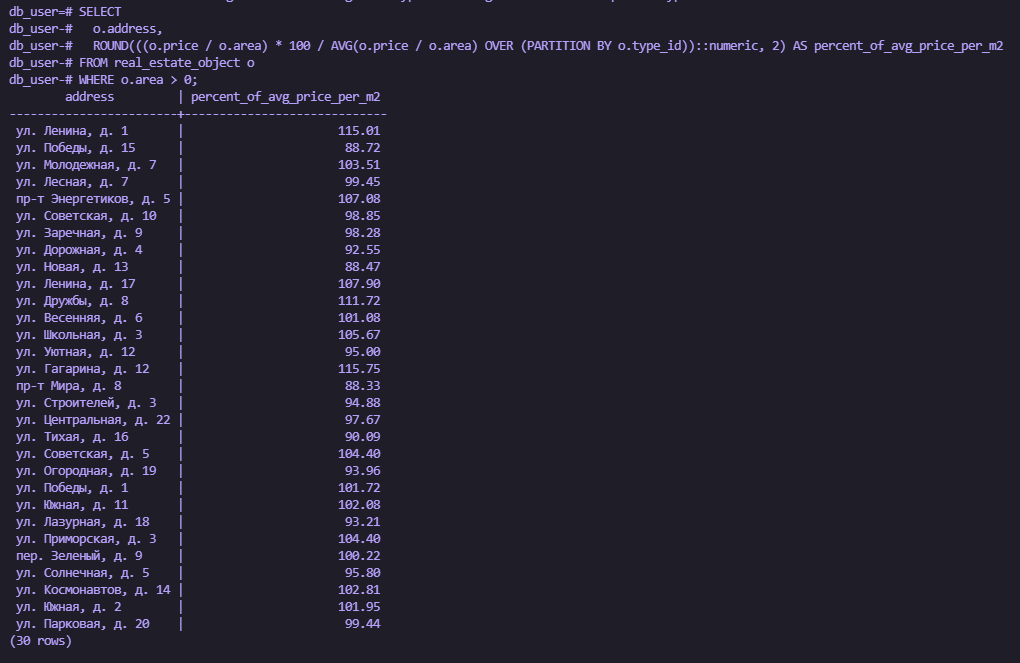
SELECT

  o.address,

  ROUND((o.price / o.area) \* 100 / AVG(o.price / o.area) OVER (PARTITION BY o.type\_id), 2) AS percent\_of\_avg\_price\_per\_m2

FROM real\_estate\_object o

WHERE o.area > 0;



3. Вывести адреса объектов недвижимости и суммарную продажную стоимость в каждой строке, отсортировать по этажам в оконной функции.

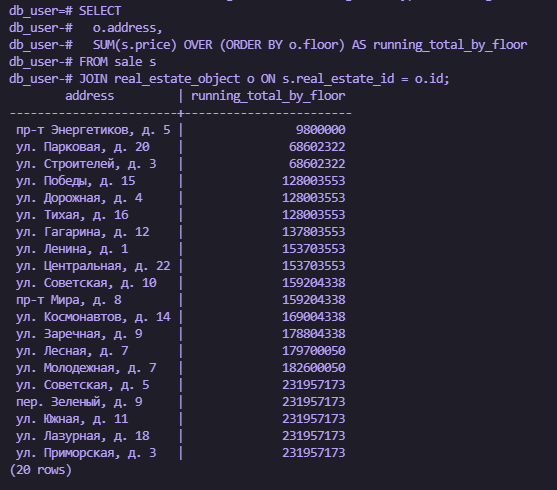
SELECT

  o.address,

  SUM(s.price) OVER (ORDER BY o.floor) AS running\_total\_by\_floor

FROM sale s

JOIN real\_estate\_object o ON s.real\_estate\_id = o.id;



4. Вывести адреса квартир, номер строки внутри района и количество квартир в районе при сортировке по стоимости 1м2 по убыванию.

SELECT

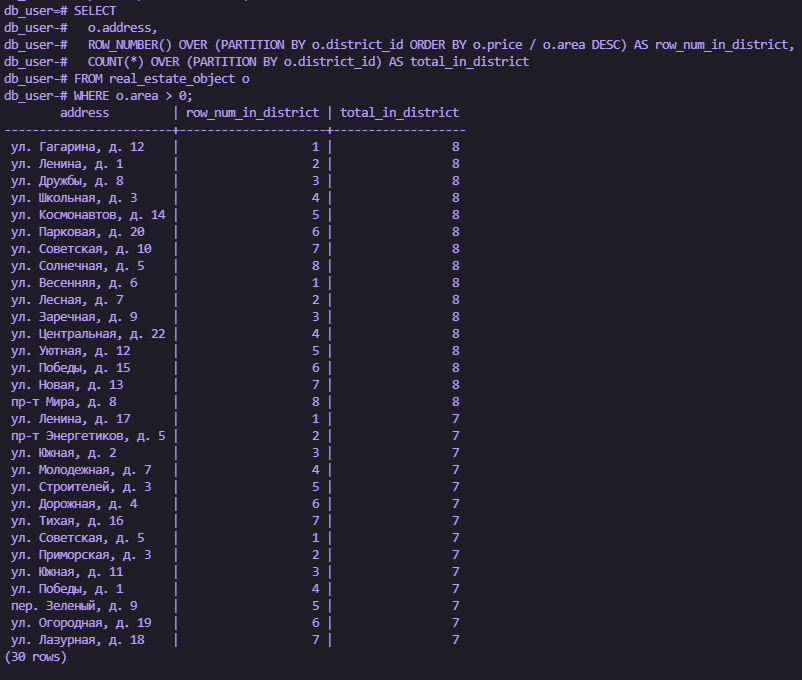
  o.address,

  ROW\_NUMBER() OVER (PARTITION BY o.district\_id ORDER BY o.price / o.area DESC) AS row\_num\_in\_district,

  COUNT(\*) OVER (PARTITION BY o.district\_id) AS total\_in\_district

FROM real\_estate\_object o

WHERE o.area > 0;



5. Вывести фамилии риэлторов и ранг по количеству продаж. Результат отсортировать по фамилии в алфавитном порядке.

SELECT

  r.last\_name,

  r.first\_name,

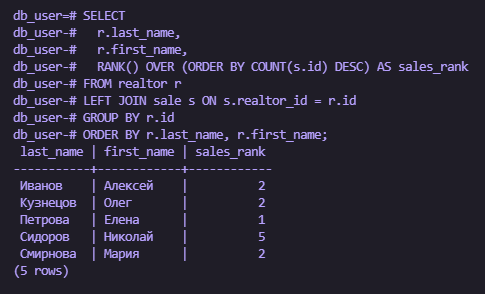
  RANK() OVER (ORDER BY COUNT(s.id) DESC) AS sales\_rank

FROM realtor r

LEFT JOIN sale s ON s.realtor\_id = r.id

GROUP BY r.id

ORDER BY r.last\_name, r.first\_name;



6. Вывести адреса объектов недвижимости и разницу со стоимостью объекта, находящегося ниже на две позиции при сортировке по возрастанию площади.

SELECT

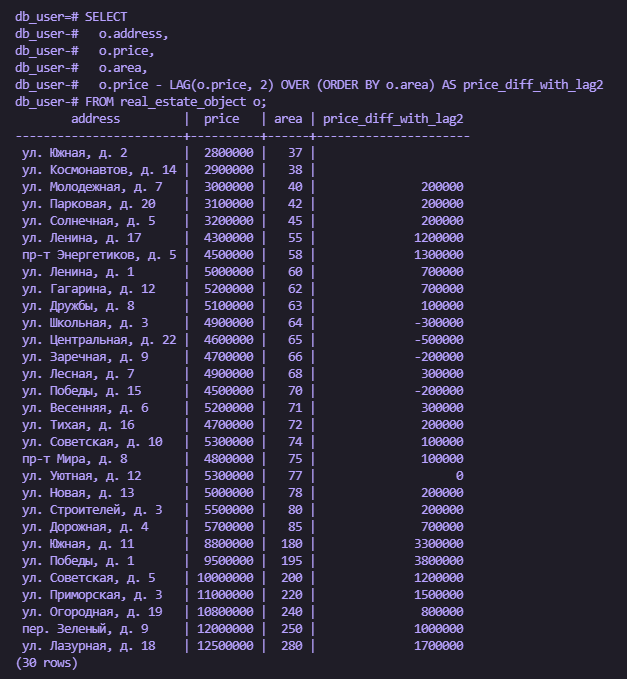
  o.address,

  o.price,

  o.area,

  o.price - LAG(o.price, 2) OVER (ORDER BY o.area) AS price\_diff\_with\_lag2

FROM real\_estate\_object o;



7. Добавить таблицу «Динамика цен», где будет хранится изменения стоимости. Таблица будет содержать следующие колонки: код объекта недвижимости, новая стоимость (значение по умолчанию - 0), дата изменения.

CREATE TABLE price\_dynamics (

  id SERIAL PRIMARY KEY,

  real\_estate\_id INT REFERENCES real\_estate\_object(id),

  new\_price DOUBLE PRECISION DEFAULT 0,

  change\_date DATE DEFAULT CURRENT\_DATE

);

SELECT

  change\_date,

  new\_price,

  COALESCE(new\_price - LAG(new\_price) OVER (PARTITION BY real\_estate\_id ORDER BY change\_date), 0) AS price\_change,

  COALESCE(

    ROUND((

      (new\_price - LAG(new\_price) OVER (PARTITION BY real\_estate\_id ORDER BY change\_date)) \* 100.0 /

      NULLIF(LAG(new\_price) OVER (PARTITION BY real\_estate\_id ORDER BY change\_date), 0)

    )::numeric, 2),

    0

  ) AS change\_percent,

  CASE

    WHEN ABS((

      (new\_price - LAG(new\_price) OVER (PARTITION BY real\_estate\_id ORDER BY change\_date)) \* 100.0 /

      NULLIF(LAG(new\_price) OVER (PARTITION BY real\_estate\_id ORDER BY change\_date), 0)

    )) > 20 THEN 'Больше 20%'

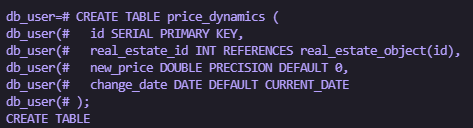
    ELSE NULL

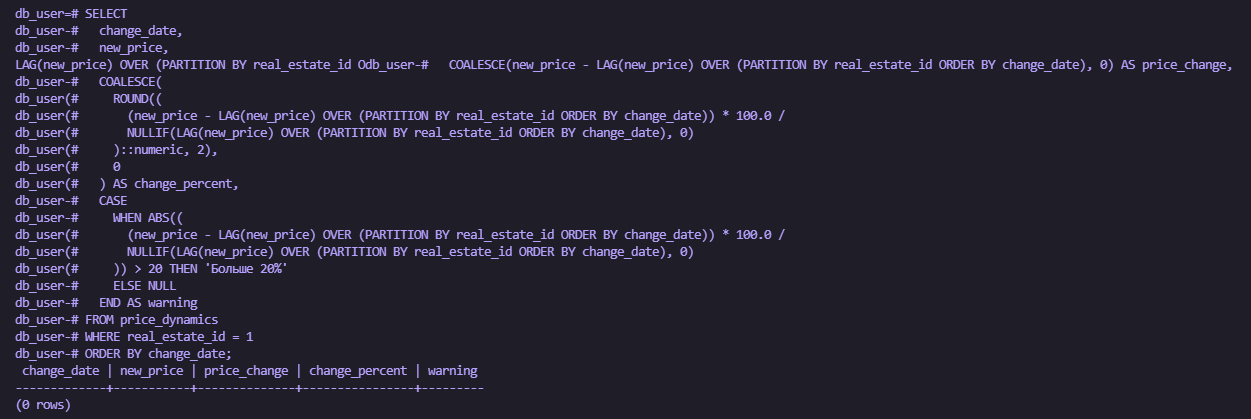
  END AS warning

FROM price\_dynamics

WHERE real\_estate\_id = 1

ORDER BY change\_date;





8. Вывести адреса объектов недвижимости и разницу с площадью последнего объекта в районе при сортировке по убыванию количества комнат.

SELECT

  o.address,

  o.area,

  o.area - LAST\_VALUE(o.area) OVER (

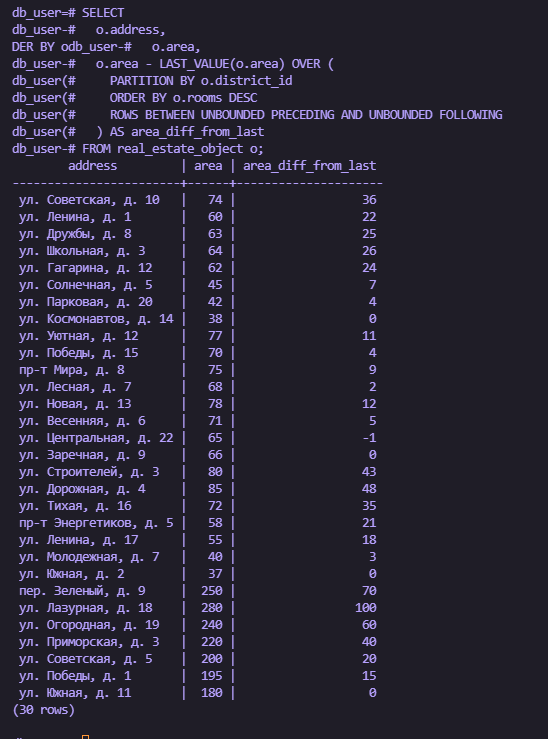
    PARTITION BY o.district\_id

    ORDER BY o.rooms DESC

    ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

  ) AS area\_diff\_from\_last

FROM real\_estate\_object o;



9. Вывести адреса объектов недвижимости и разницу с продажной стоимостью первого объекта по каждому типу объекта при сортировке по возрастанию количества комнат.

SELECT

  o.address,

  s.price,

  s.price - FIRST\_VALUE(s.price) OVER (

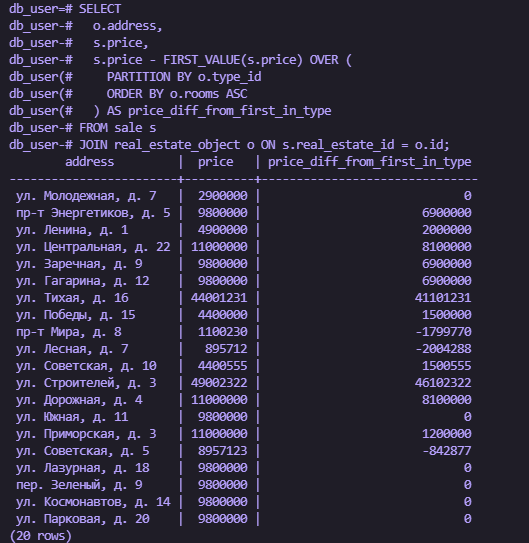
    PARTITION BY o.type\_id

    ORDER BY o.rooms ASC

  ) AS price\_diff\_from\_first\_in\_type

FROM sale s

JOIN real\_estate\_object o ON s.real\_estate\_id = o.id;



10. Определить, какова самая высокая стоимость продажи объекта недвижимости, сделанная каждым сотрудником риэлтором, и сопоставить её с текущей записью.

SELECT

  r.id AS realtor\_id,

  r.last\_name || ' ' || r.first\_name AS full\_name,

  s.price AS sale\_price,

  MAX(s.price) OVER (PARTITION BY r.id) AS max\_price,

  CASE

    WHEN s.price = MAX(s.price) OVER (PARTITION BY r.id) THEN 'Да'

    ELSE 'Нет'

  END AS match\_max

FROM sale s

JOIN realtor r ON s.realtor\_id = r.id

ORDER BY r.id, s.price;

