**Отчет по практической работе № 8**

**Разработка базы данных на основе разработанной ER-модели**

**1. Информация о студенте**

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

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

-- Таблица "Клиенты"

CREATE TABLE Клиенты (

    код\_клиента SERIAL PRIMARY KEY,

    фамилия VARCHAR(50) NOT NULL,

    имя VARCHAR(50) NOT NULL,

    отчество VARCHAR(50),

    паспортные\_данные VARCHAR(20) UNIQUE NOT NULL,

    комментарий TEXT

);

-- Таблица "Номера"

CREATE TABLE Номера (

    код\_номера SERIAL PRIMARY KEY,

    номер VARCHAR(10) NOT NULL,

    количество\_человек INTEGER NOT NULL CHECK (количество\_человек > 0),

    комфортность VARCHAR(20) CHECK (комфортность IN ('люкс', 'полулюкс', 'обычный')),

    цена NUMERIC(10, 2) NOT NULL CHECK (цена >= 0)

);

-- Таблица "Поселение"

CREATE TABLE Поселение (

    код\_поселения SERIAL PRIMARY KEY,

    код\_клиента INTEGER NOT NULL REFERENCES Клиенты(код\_клиента) ON DELETE CASCADE,

    код\_номера INTEGER NOT NULL REFERENCES Номера(код\_номера) ON DELETE CASCADE,

    дата\_поселения DATE NOT NULL,

    дата\_освобождения DATE NOT NULL,

    примечание TEXT,

    CHECK (дата\_освобождения >= дата\_поселения)

);

-- Таблица "Бронирование"

CREATE TABLE Бронирование (

    код\_бронирования SERIAL PRIMARY KEY,

    код\_клиента INTEGER NOT NULL REFERENCES Клиенты(код\_клиента) ON DELETE CASCADE,

    код\_номера INTEGER NOT NULL REFERENCES Номера(код\_номера) ON DELETE CASCADE,

    дата\_начала DATE NOT NULL,

    дата\_окончания DATE NOT NULL,

    статус VARCHAR(20) NOT NULL CHECK (статус IN ('активно', 'отменено')),

    CHECK (дата\_окончания >= дата\_начала)

);

-- Таблица "Скидки"

CREATE TABLE Скидки (

    код\_скидки SERIAL PRIMARY KEY,

    наименование VARCHAR(100) NOT NULL,

    процент NUMERIC(5, 2) NOT NULL CHECK (процент >= 0 AND процент <= 100),

    описание TEXT,

    активность BOOLEAN NOT NULL DEFAULT TRUE

);

-- Таблица "Скидки клиентов"

CREATE TABLE Скидки\_клиентов (

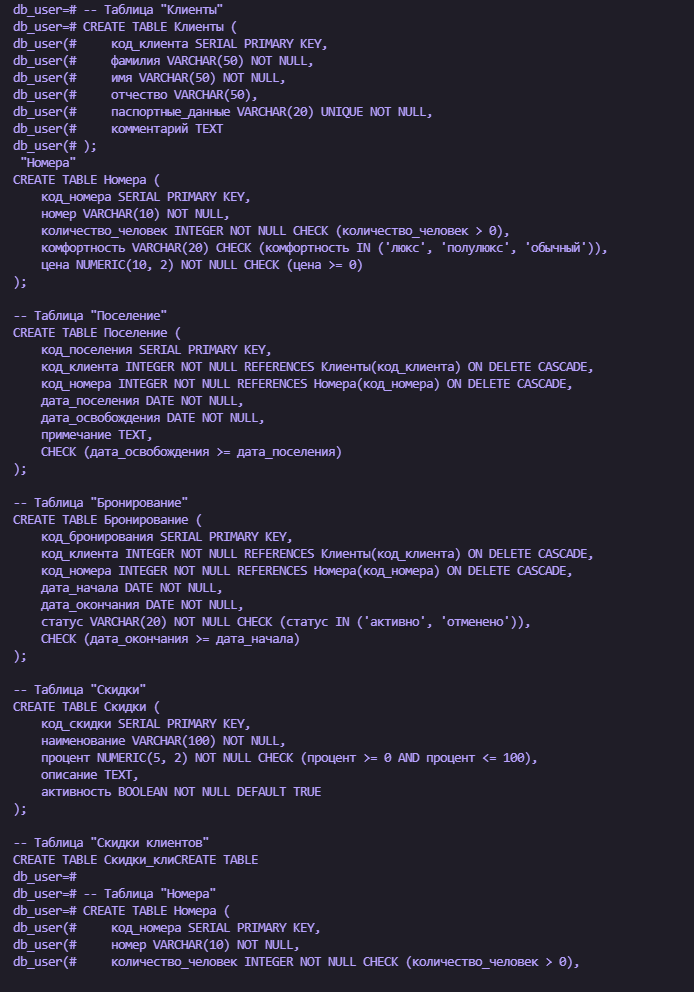
    код\_скидки\_клиента SERIAL PRIMARY KEY,

    код\_клиента INTEGER NOT NULL REFERENCES Клиенты(код\_клиента) ON DELETE CASCADE,

    код\_скидки INTEGER NOT NULL REFERENCES Скидки(код\_скидки) ON DELETE CASCADE,

    UNIQUE (код\_клиента, код\_скидки) -- чтобы исключить дублирующие связи

);



2. Заполнить таблицы данными в соответствии со структурой.

-- Клиенты

INSERT INTO Клиенты (фамилия, имя, отчество, паспортные\_данные, комментарий) VALUES

('Иванов', 'Иван', 'Иванович', '1234 567890', 'Постоянный клиент'),

('Петров', 'Петр', 'Петрович', '2345 678901', NULL),

('Сидорова', 'Анна', 'Алексеевна', '3456 789012', 'Предпочитает номера люкс');

-- Номера

INSERT INTO Номера (номер, количество\_человек, комфортность, цена) VALUES

('101', 2, 'люкс', 5000.00),

('102', 1, 'обычный', 2000.00),

('201', 3, 'полулюкс', 3500.00);

-- Скидки

INSERT INTO Скидки (наименование, процент, описание, активность) VALUES

('Постоянный клиент', 10.00, 'Скидка для клиентов, заселявшихся более 5 раз', TRUE),

('Сезонная акция', 15.00, 'Скидка в зимний период', FALSE),

('Раннее бронирование', 5.00, 'Скидка при бронировании за месяц', TRUE);

-- Скидки клиентов

INSERT INTO Скидки\_клиентов (код\_клиента, код\_скидки) VALUES

(1, 1),  -- Иванов – Постоянный клиент

(2, 3),  -- Петров – Раннее бронирование

(3, 1);  -- Сидорова – Постоянный клиент

-- Бронирование

INSERT INTO Бронирование (код\_клиента, код\_номера, дата\_начала, дата\_окончания, статус) VALUES

(1, 101, '2025-06-01', '2025-06-05', 'активно'),

(2, 102, '2025-06-10', '2025-06-12', 'отменено'),

(3, 201, '2025-07-01', '2025-07-07', 'активно');

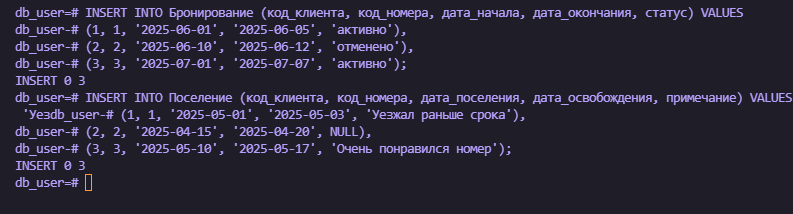
-- Поселение

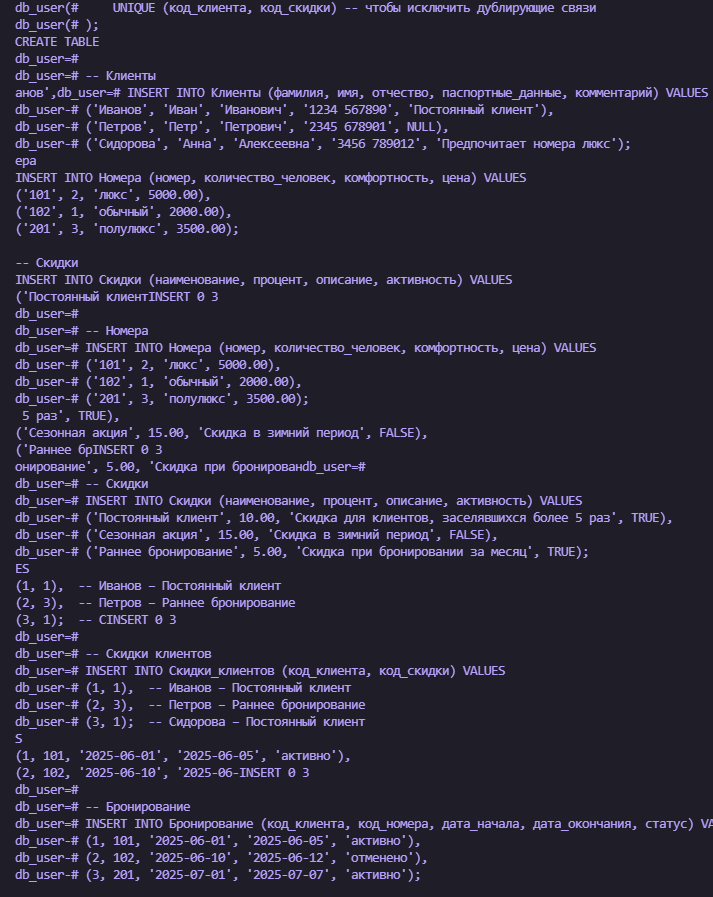
INSERT INTO Поселение (код\_клиента, код\_номера, дата\_поселения, дата\_освобождения, примечание) VALUES

(1, 101, '2025-05-01', '2025-05-03', 'Уезжал раньше срока'),

(2, 102, '2025-04-15', '2025-04-20', NULL),

(3, 201, '2025-05-10', '2025-05-17', 'Очень понравился номер');





3. Выполнить запросы к базе данных.

-- 1. Количество бронирований по клиентам, у которых ровно 1 бронирование

SELECT к.фамилия, к.имя, COUNT(б.код\_бронирования) AS кол\_бронирований

FROM Бронирование б

JOIN Клиенты к ON к.код\_клиента = б.код\_клиента

GROUP BY к.фамилия, к.имя

HAVING COUNT(б.код\_бронирования) > 1;

-- 2. Средняя цена по категории номера, только если больше одной записи в категории

SELECT комфортность, AVG(цена) AS средняя\_цена

FROM Номера

GROUP BY комфортность

HAVING COUNT(\*) > 1;

-- 3. Кол-во клиентов на каждую скидку, где клиентов не менее 2

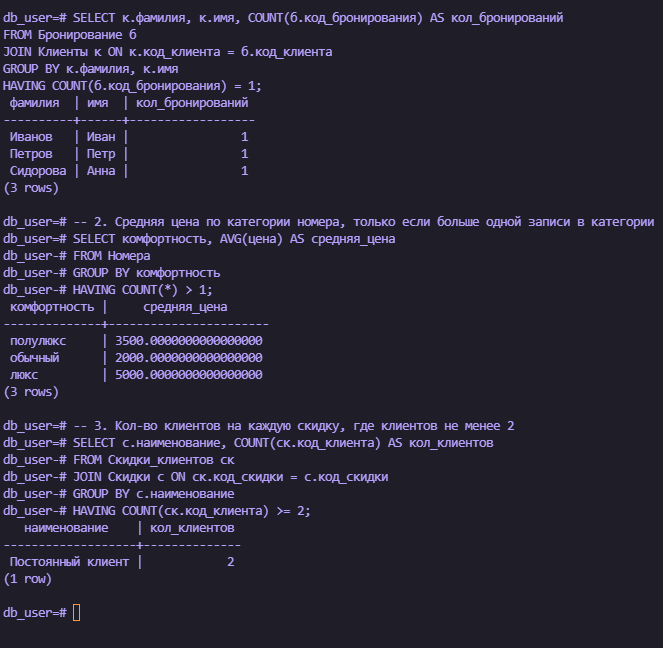
SELECT с.наименование, COUNT(ск.код\_клиента) AS кол\_клиентов

FROM Скидки\_клиентов ск

JOIN Скидки с ON ск.код\_скидки = с.код\_скидки

GROUP BY с.наименование

HAVING COUNT(ск.код\_клиента) >= 2;



-- 1. Некоррелированный подзапрос: Клиенты, у которых цена номера выше средней

SELECT DISTINCT к.фамилия, к.имя

FROM Поселение п

JOIN Клиенты к ON к.код\_клиента = п.код\_клиента

JOIN Номера н ON н.код\_номера = п.код\_номера

WHERE н.цена > (SELECT AVG(цена) FROM Номера);

-- 2. Коррелированный подзапрос: Клиенты, у которых есть активные бронирования

SELECT фамилия, имя

FROM Клиенты к

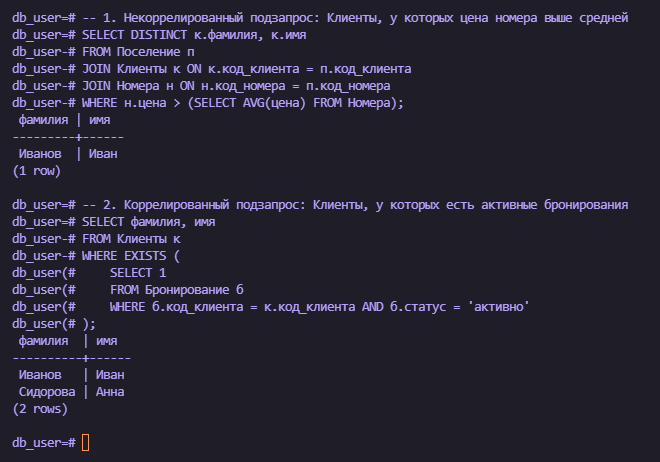
WHERE EXISTS (

    SELECT 1

    FROM Бронирование б

    WHERE б.код\_клиента = к.код\_клиента AND б.статус = 'активно'

);



-- 1. ROLLUP: Сумма по каждому клиенту и общий итог

SELECT к.фамилия, к.имя, SUM(н.цена) AS сумма\_платежей

FROM Поселение п

JOIN Клиенты к ON к.код\_клиента = п.код\_клиента

JOIN Номера н ON н.код\_номера = п.код\_номера

GROUP BY ROLLUP (к.фамилия, к.имя);

-- 2. GROUPING SETS: Группировка по скидке и по клиенту отдельно

SELECT с.наименование, к.фамилия, COUNT(\*) AS кол\_записей

FROM Скидки\_клиентов ск

JOIN Скидки с ON с.код\_скидки = ск.код\_скидки

JOIN Клиенты к ON к.код\_клиента = ск.код\_клиента

GROUP BY GROUPING SETS (

    (с.наименование),

    (к.фамилия)

);

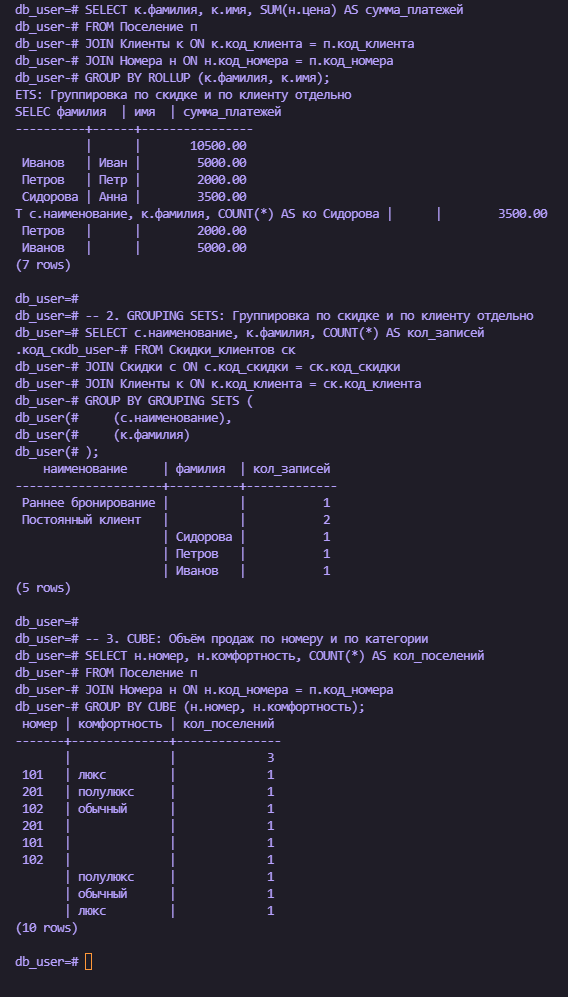
-- 3. CUBE: Объём продаж по номеру и по категории

SELECT н.номер, н.комфортность, COUNT(\*) AS кол\_поселений

FROM Поселение п

JOIN Номера н ON н.код\_номера = п.код\_номера

GROUP BY CUBE (н.номер, н.комфортность);



-- 1. LAG: предыдущее бронирование клиента

SELECT код\_клиента, дата\_начала,

       LAG(дата\_начала) OVER (PARTITION BY код\_клиента ORDER BY дата\_начала) AS пред\_дата

FROM Бронирование;

-- 2. LEAD: следующее бронирование клиента

SELECT код\_клиента, дата\_начала,

       LEAD(дата\_начала) OVER (PARTITION BY код\_клиента ORDER BY дата\_начала) AS след\_дата

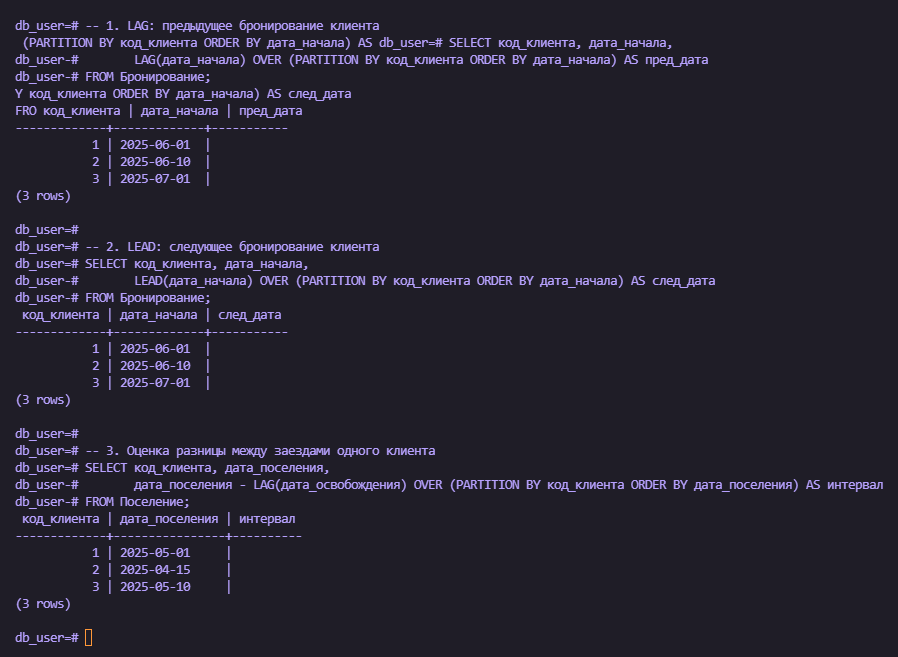
FROM Бронирование;

-- 3. Оценка разницы между заездами одного клиента

SELECT код\_клиента, дата\_поселения,

       дата\_поселения - LAG(дата\_освобождения) OVER (PARTITION BY код\_клиента ORDER BY дата\_поселения) AS интервал

FROM Поселение;



-- Назначить категорию клиента по количеству бронирований

SELECT к.фамилия, к.имя, COUNT(б.код\_бронирования) AS всего,

    CASE

        WHEN COUNT(б.код\_бронирования) >= 3 THEN 'VIP'

        WHEN COUNT(б.код\_бронирования) = 2 THEN 'Постоянный'

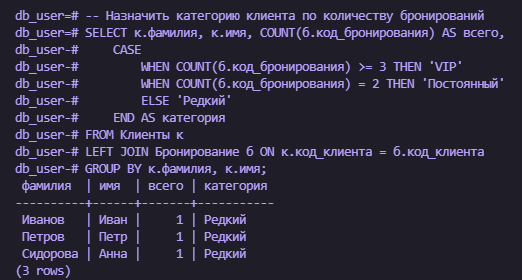
        ELSE 'Редкий'

    END AS категория

FROM Клиенты к

LEFT JOIN Бронирование б ON к.код\_клиента = б.код\_клиента

GROUP BY к.фамилия, к.имя;



-- Средняя цена по категории, а потом выбор категорий выше среднего

WITH Средние\_цены AS (

    SELECT комфортность, AVG(цена) AS ср\_цена

    FROM Номера

    GROUP BY комфортность

)

SELECT \*

FROM Средние\_цены

WHERE ср\_цена > (SELECT AVG(цена) FROM Номера);

