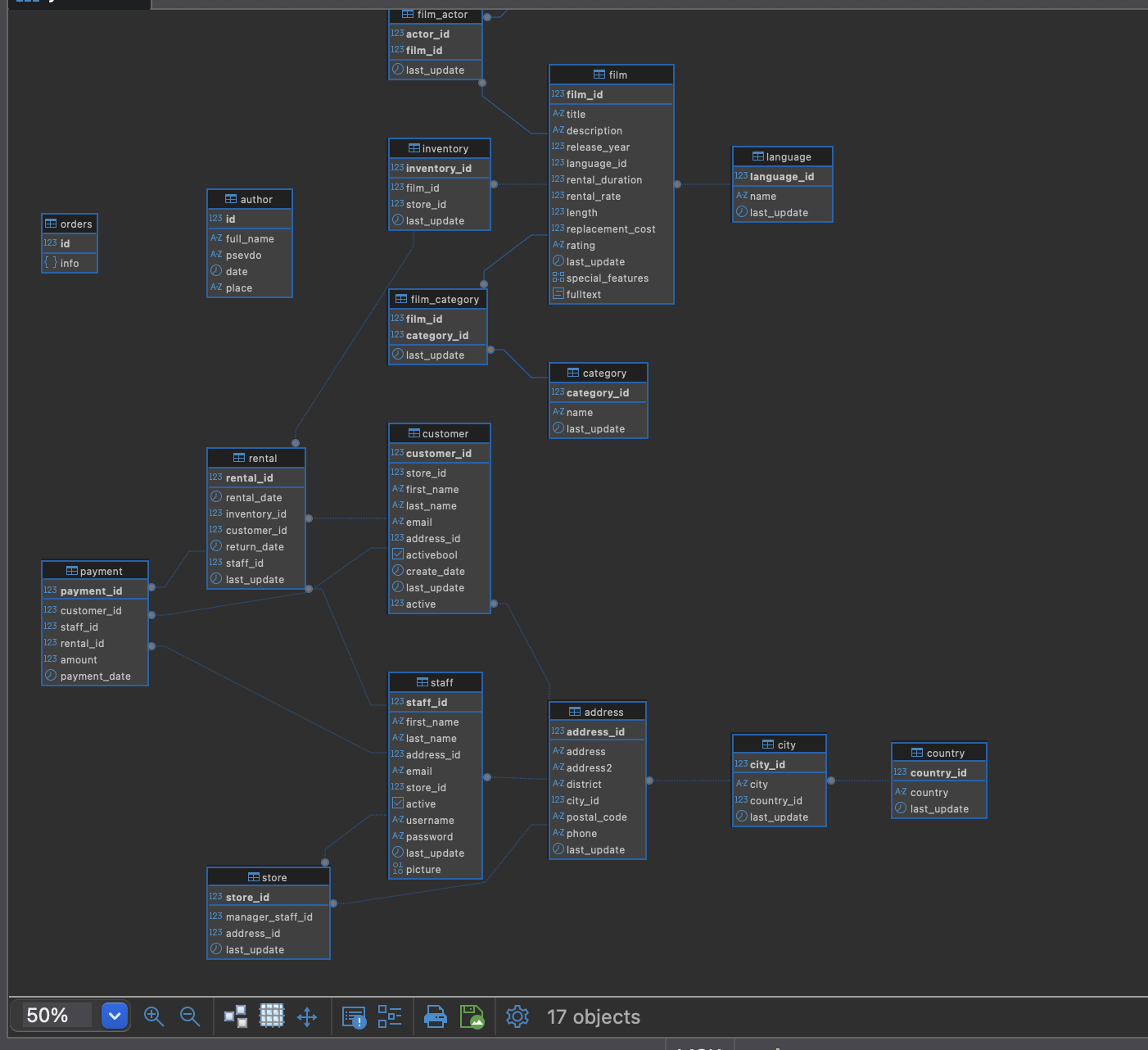
-- ===================================================

-- ДЗ\_3: Работа с базой данных dvd\_rental

-- Выполнено: Кожухов Игорь

ER диаграмма



-- ===================================================

-- ===================================================

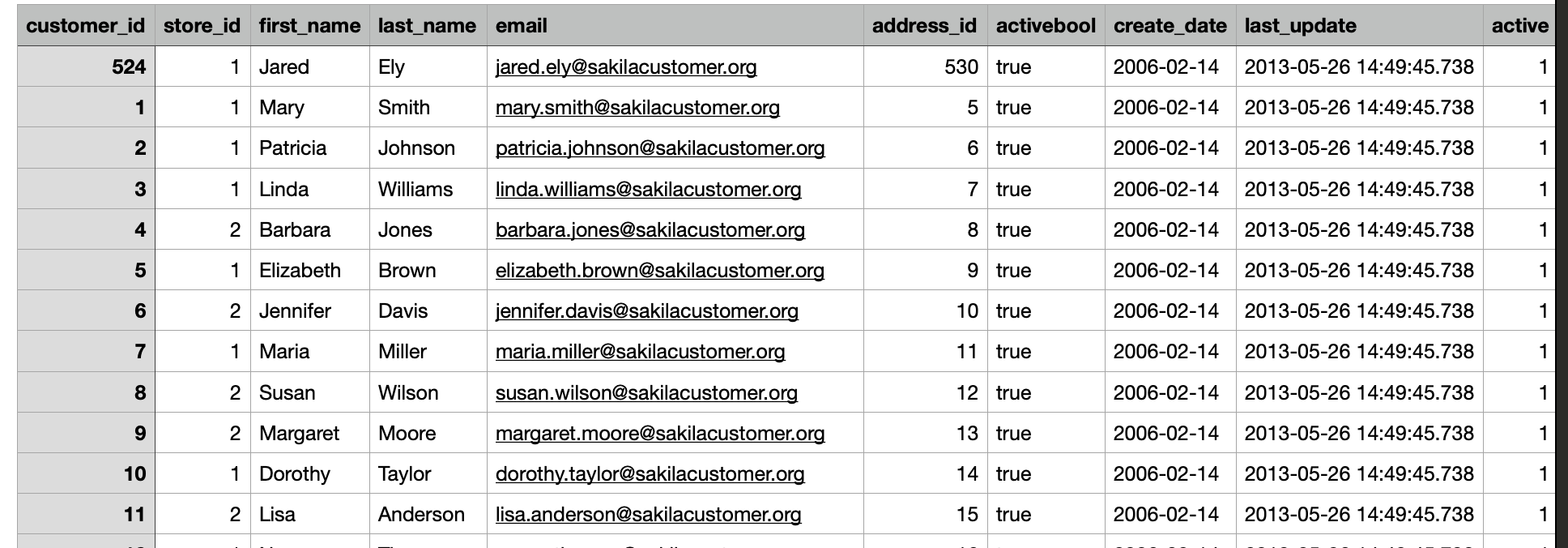
-- 2. SQL и получение данных

-- ===================================================

-- 2.1 Вывести список всех клиентов (таблица customer)

SELECT \*

FROM customer;

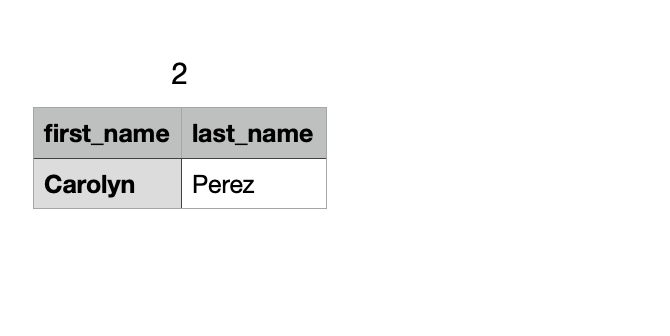


-- 2.2 Вывести имена и фамилии клиентов с именем Carolyn

SELECT first\_name, last\_name

FROM customer

WHERE first\_name = 'Carolyn';



-- 2.3 Вывести полные имена клиентов (имя + фамилия в одной колонке),

-- у которых имя или фамилия содержат подстроку 'ary' (например: Mary, Geary)

SELECT first\_name || ' ' || last\_name AS full\_name

FROM customer

WHERE first\_name LIKE '%ary%' OR last\_name LIKE '%ary%';

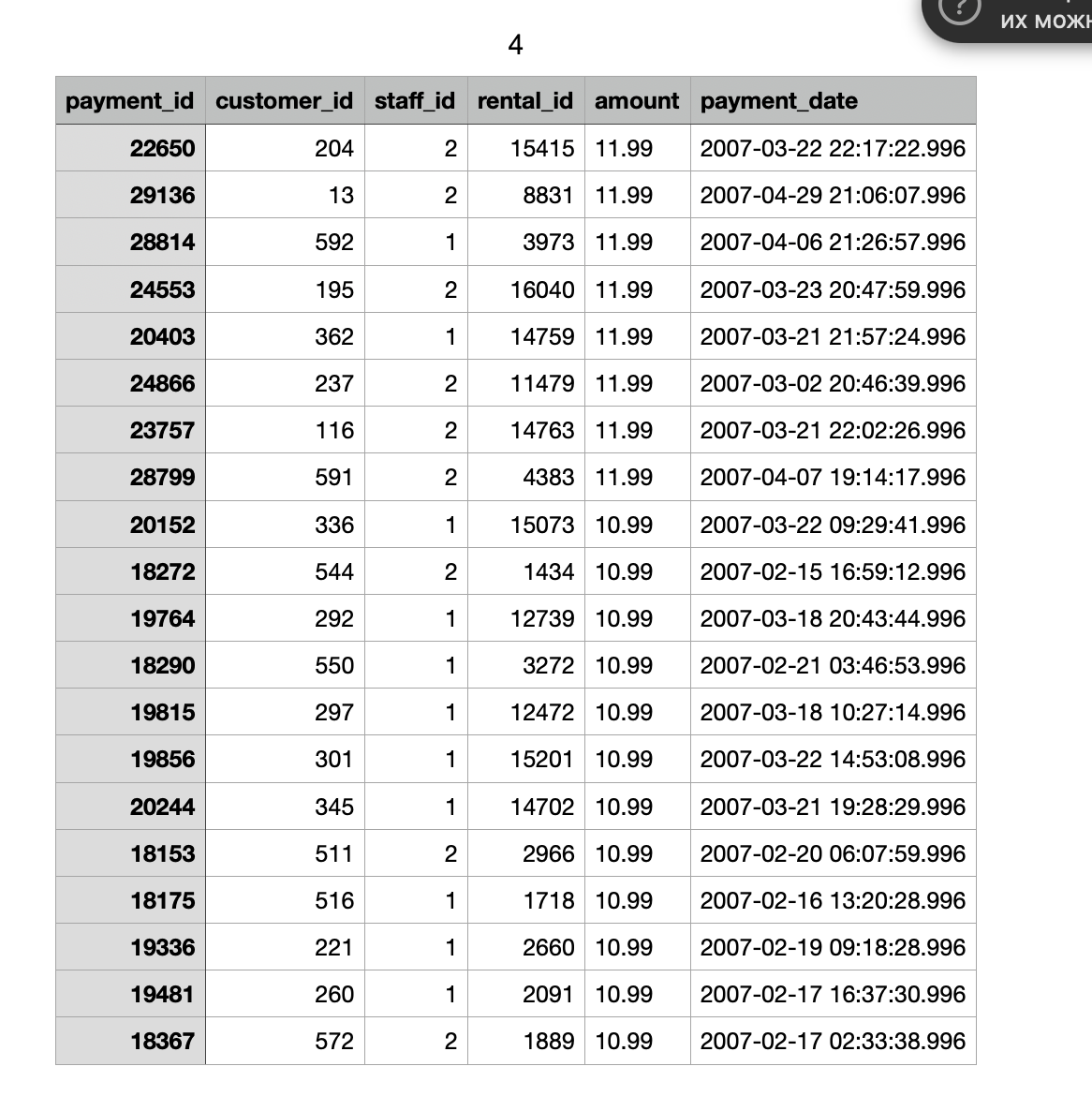
-- 2.4 Вывести 20 самых крупных транзакций (таблица payment)

SELECT \*

FROM payment

ORDER BY amount DESC

LIMIT 20;



-- 2.5 Вывести адреса всех магазинов, используя подзапрос

SELECT address, district, postal\_code, phone

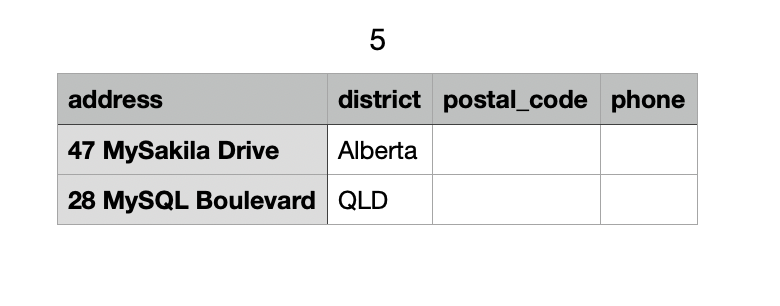
FROM address

WHERE address\_id IN (

SELECT address\_id

FROM store

);



-- 2.6 Для каждой оплаты вывести число, месяц и день недели в числовом формате

-- (Понедельник -- 1, Вторник -- 2 и т.д.)

SELECT

payment\_id,

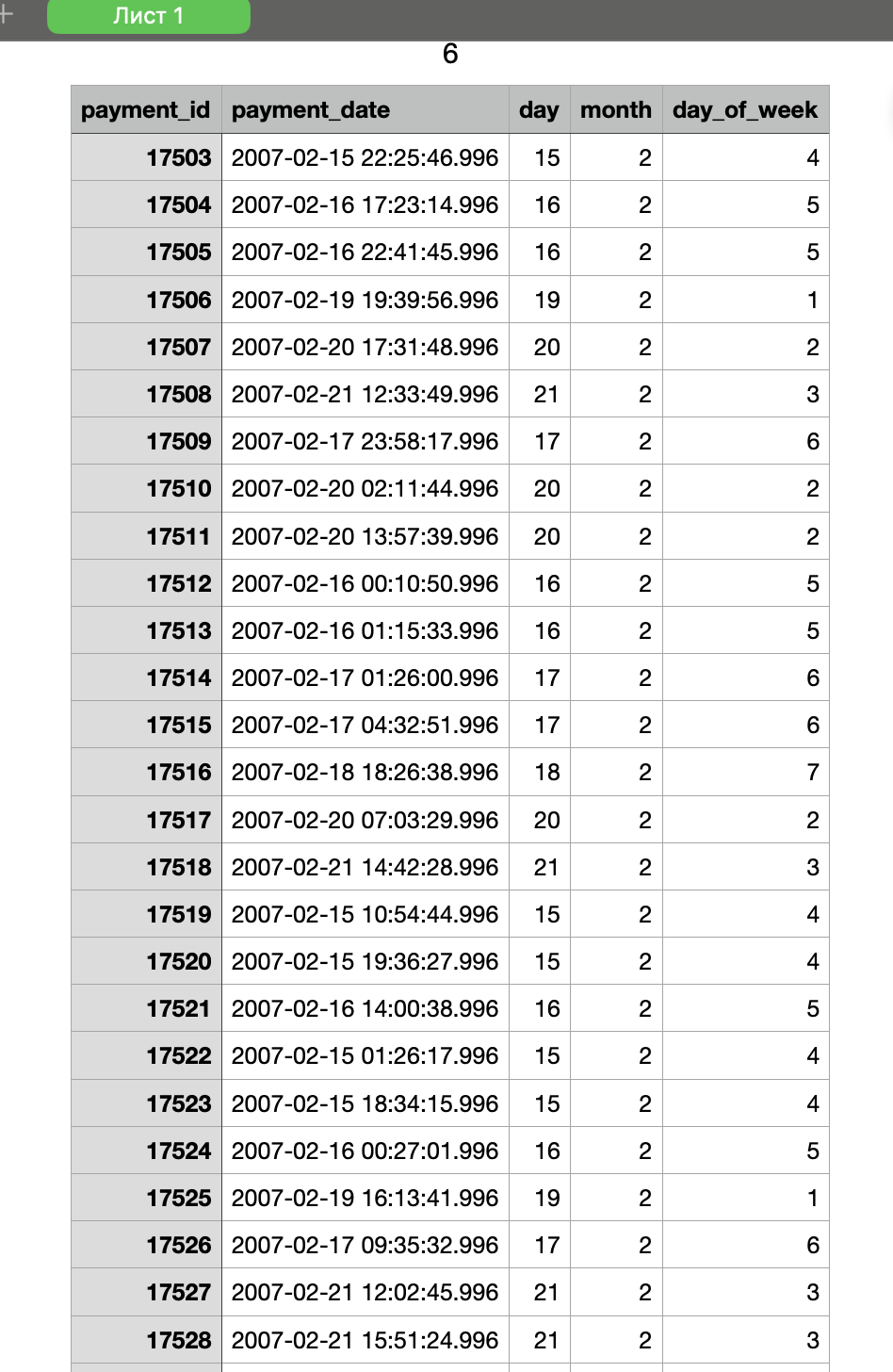
payment\_date,

EXTRACT(DAY FROM payment\_date) AS day,

EXTRACT(MONTH FROM payment\_date) AS month,

EXTRACT(ISODOW FROM payment\_date) AS day\_of\_week

FROM payment;



-- 2.7 Вывести, кто (customer\_id), когда (rental\_date, приведенная к типу date)

-- и у кого (staff\_id) брал диски в аренду в июне 2005 года

SELECT

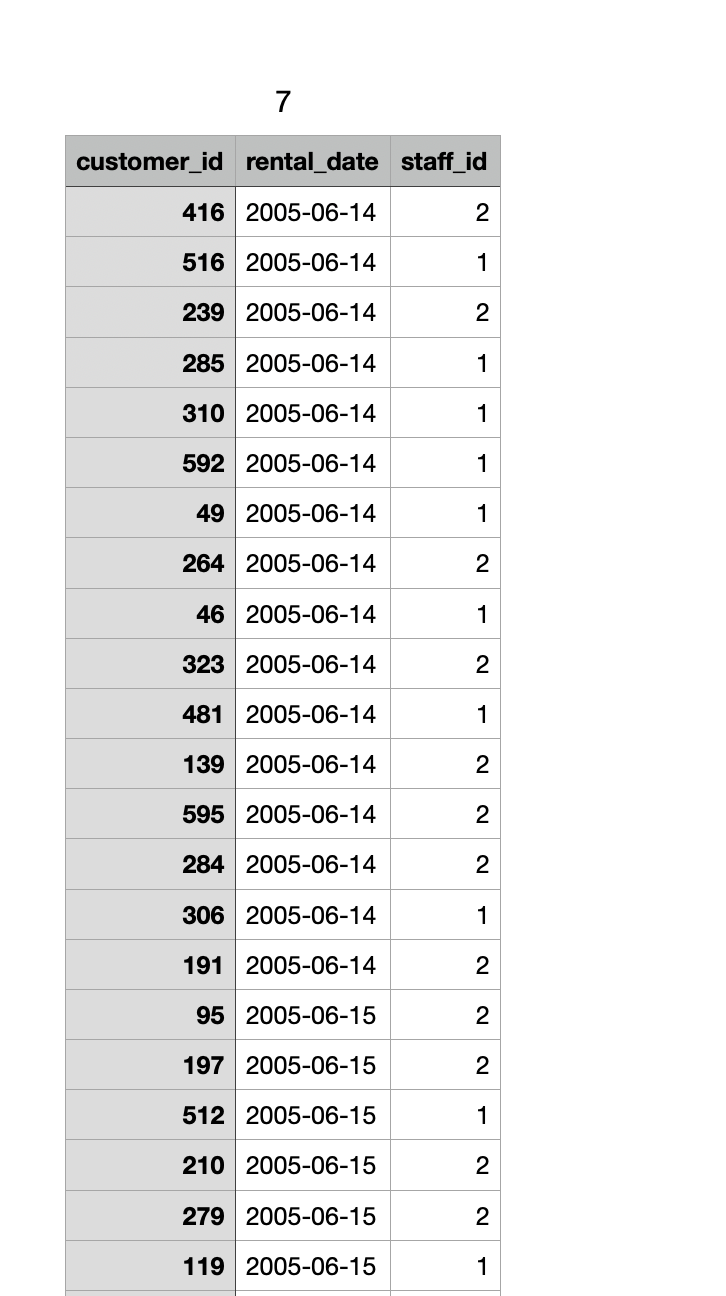
customer\_id,

rental\_date::date AS rental\_date,

staff\_id

FROM rental

WHERE rental\_date >= '2005-06-01' AND rental\_date < '2005-07-01';



-- 2.8 Вывести название, описание и длительность фильмов (таблица film),

-- выпущенных после 2000 года, с длительностью от 60 до 120 минут включительно.

-- Показать первые 20 фильмов с наибольшей длительностью

SELECT title, description, length

FROM film

WHERE release\_year > 2000

AND length BETWEEN 60 AND 120

ORDER BY length DESC

LIMIT 20;



-- 2.9 Найти все платежи (таблица payment), совершенные в апреле 2007 года,

-- стоимость которых не превышает 4 долларов. Вывести идентификатор платежа,

-- дату (без времени) и сумму платежа. Отсортировать платежи по убыванию суммы,

-- а при равной сумме --- по более ранней дате

SELECT

payment\_id,

payment\_date::date AS payment\_date,

amount

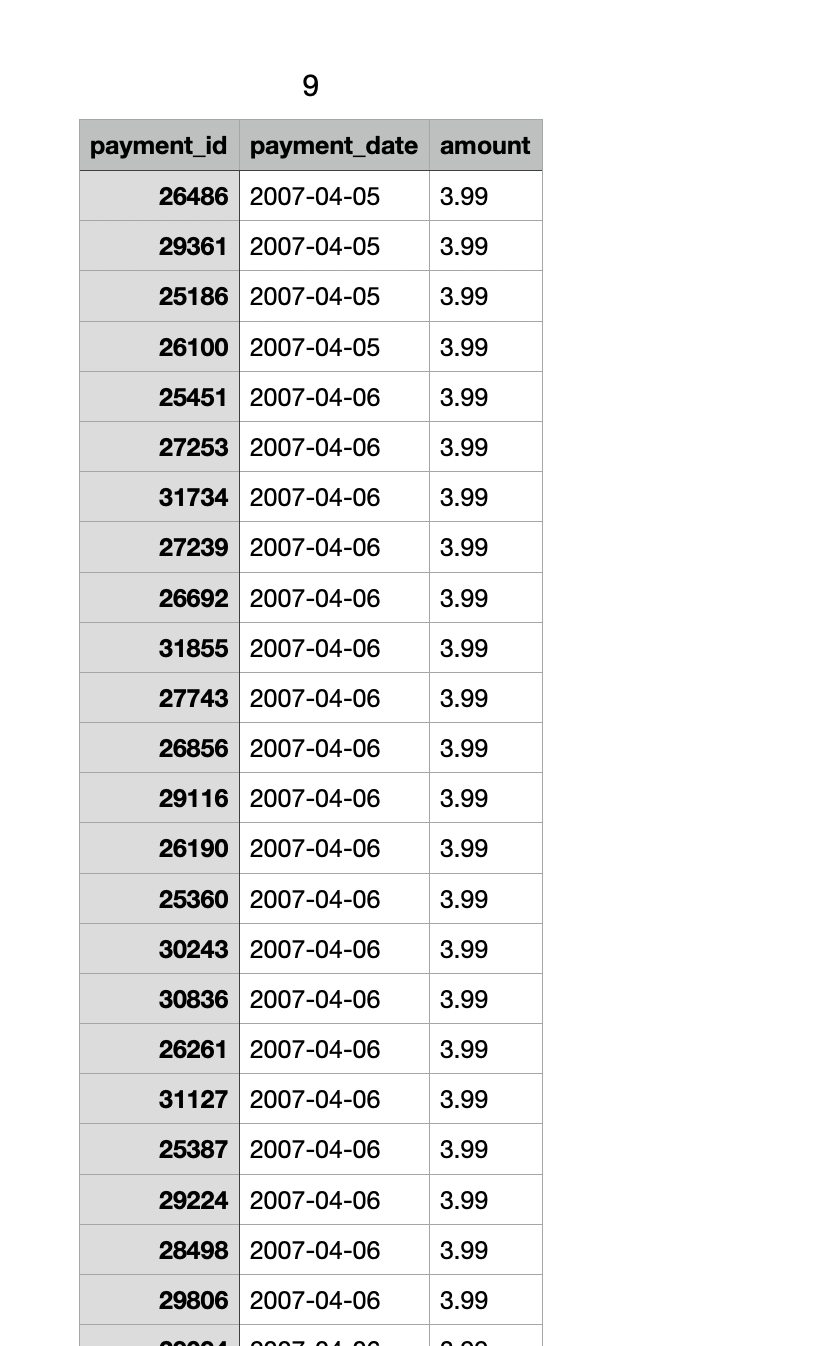
FROM payment

WHERE payment\_date >= '2007-04-01'

AND payment\_date < '2007-05-01'

AND amount <= 4.00

ORDER BY amount DESC, payment\_date ASC;



-- 2.10 Показать имена, фамилии и идентификаторы всех клиентов с именами

-- Jack, Bob или Sara, чья фамилия содержит букву «p».

-- Переименовать колонки: с именем --- в «Имя», с идентификатором --- в «Идентификатор»,

-- с фамилией --- в «Фамилия». Отсортировать клиентов по возрастанию идентификатора

SELECT

first\_name AS "Имя",

customer\_id AS "Идентификатор",

last\_name AS "Фамилия"

FROM customer

WHERE first\_name IN ('Jack', 'Bob', 'Sara')

AND last\_name ILIKE '%p%'

ORDER BY customer\_id;



-- 2.11 Работа с собственной таблицей студентов

-- Создать таблицу студентов с полями: имя, фамилия, возраст, дата рождения и адрес.

-- Все поля должны запрещать внесение пустых значений (NOT NULL)

CREATE TABLE students (

id SERIAL PRIMARY KEY,

first\_name VARCHAR(100) NOT NULL,

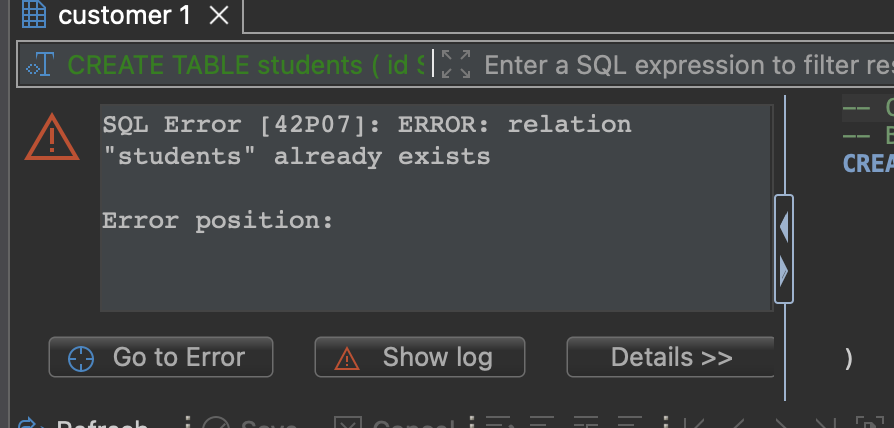
last\_name VARCHAR(100) NOT NULL,

age INT NOT NULL,

birth\_date DATE NOT NULL,

address TEXT NOT NULL

);



-- Внести в таблицу одного студента с id > 50

INSERT INTO students (id, first\_name, last\_name, age, birth\_date, address)

VALUES (51, 'Иван', 'Иванов', 20, '2004-05-15', 'Москва, ул. Ленина, 1');



-- Просмотреть текущие записи таблицы

SELECT \* FROM students;

-- Внести несколько записей одним запросом, используя автоинкремент id

INSERT INTO students (first\_name, last\_name, age, birth\_date, address)

VALUES

('Мария', 'Петрова', 21, '2003-03-20', 'Санкт-Петербург, Невский пр., 10'),

('Алексей', 'Сидоров', 22, '2002-11-08', 'Казань, ул. Баумана, 5'),

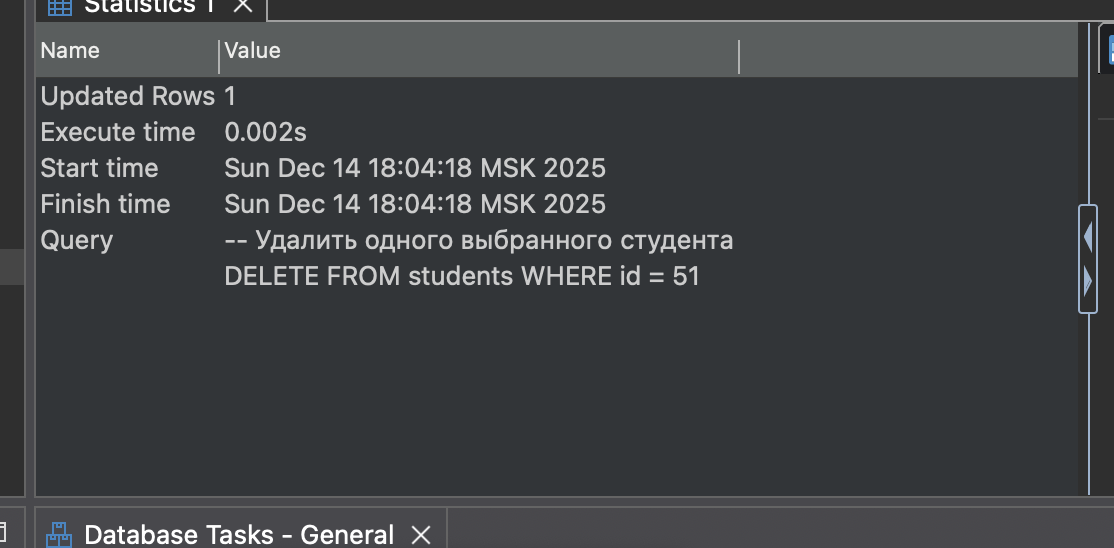
('Ольга', 'Смирнова', 19, '2005-07-12', 'Новосибирск, пр. Красный, 15');

-- Снова просмотреть текущие записи таблицы

SELECT \* FROM students;

-- Удалить одного выбранного студента

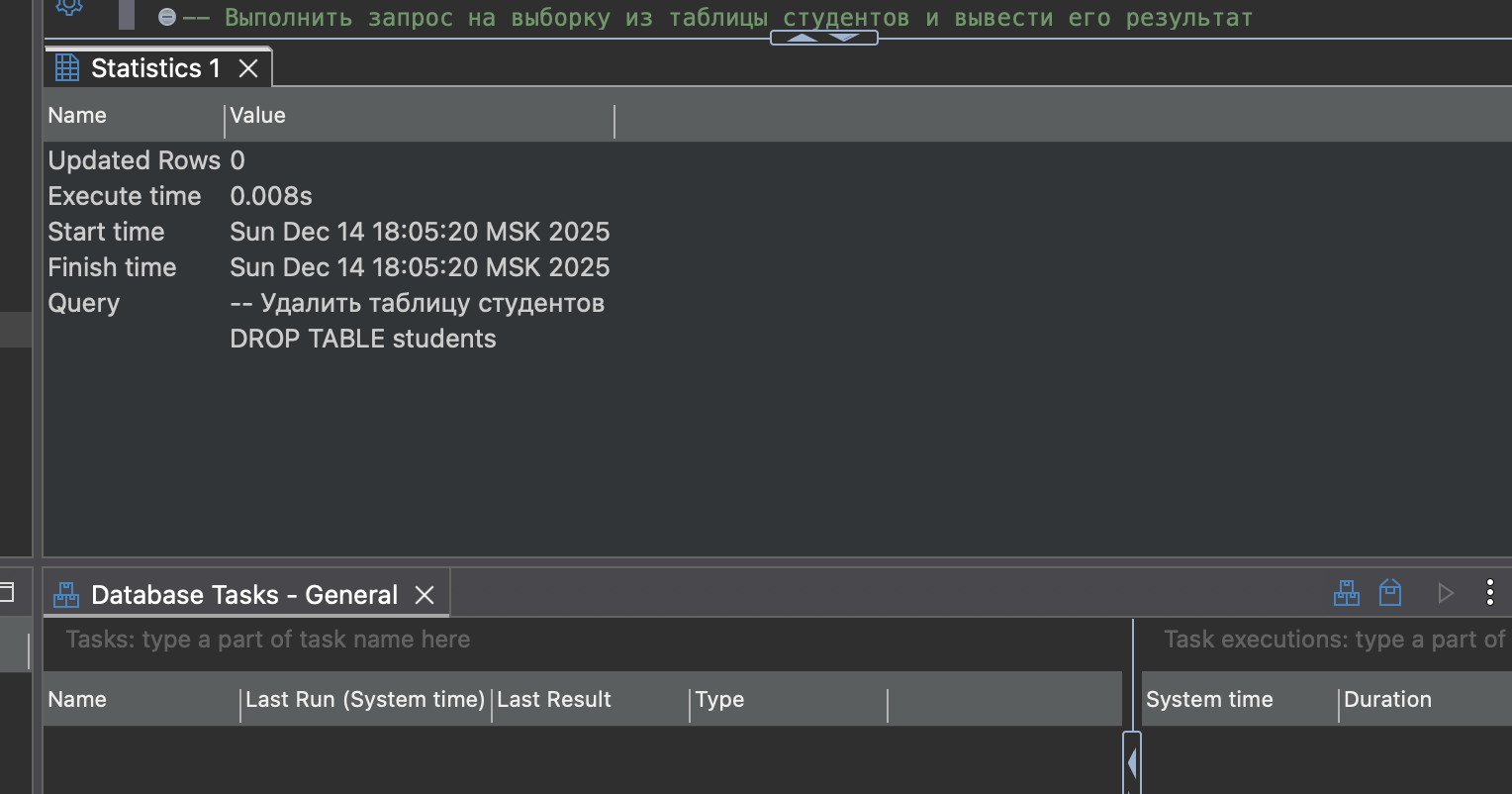
DELETE FROM students WHERE id = 51;



-- Вывести полный список студентов

SELECT \* FROM students;

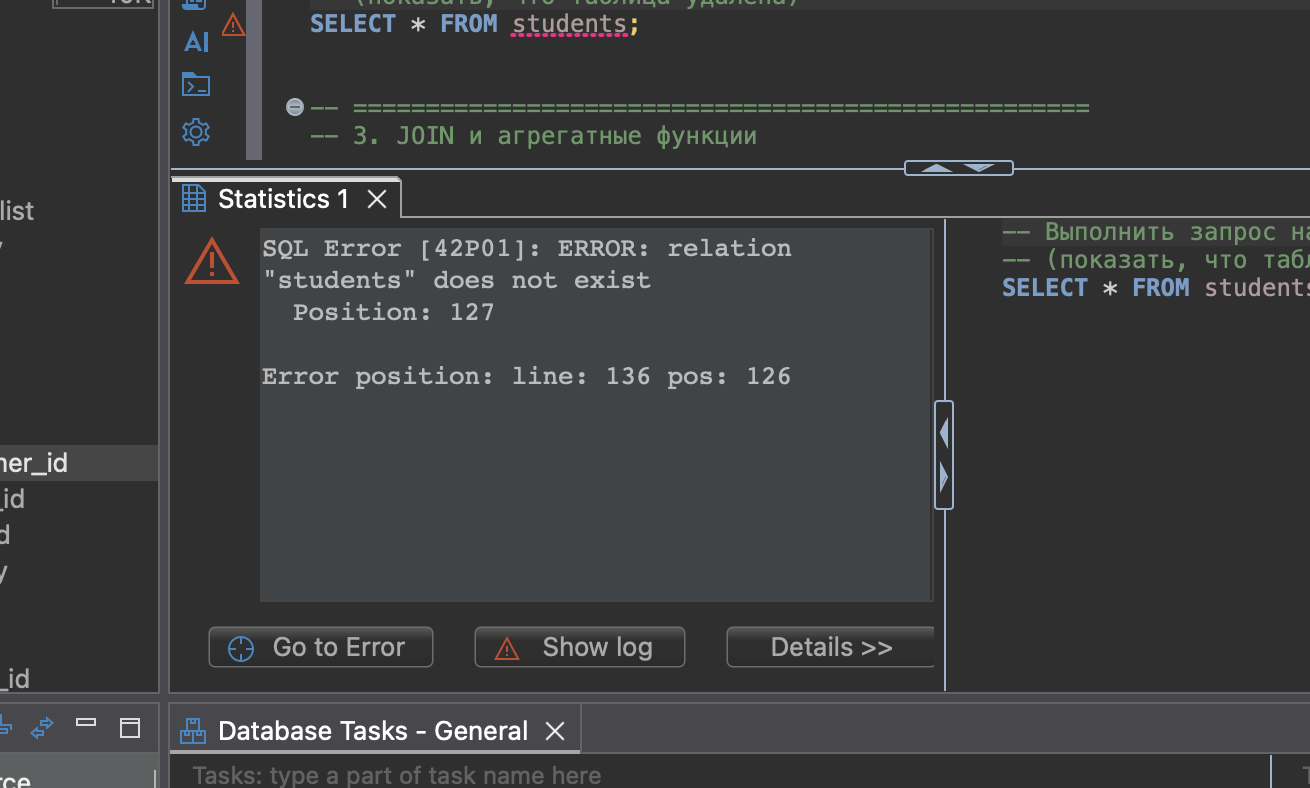
-- Удалить таблицу студентов

DROP TABLE students;

-- Выполнить запрос на выборку из таблицы студентов и вывести его результат

-- (показать, что таблица удалена)

SELECT \* FROM students;



-- ===================================================

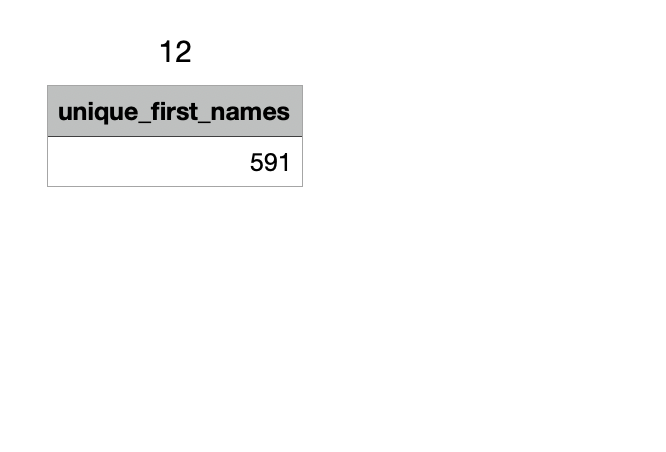
-- 3. JOIN и агрегатные функции

-- ===================================================

-- 3.12 Вывести количество уникальных имен клиентов

SELECT COUNT(DISTINCT first\_name) AS unique\_first\_names

FROM customer;



-- 3.13 Вывести 5 самых часто встречающихся сумм оплаты: саму сумму,

-- даты таких оплат, количество платежей с этой суммой и общую сумму этих платежей

SELECT

amount,

STRING\_AGG(DISTINCT payment\_date::date::text, ', ') AS payment\_dates,

COUNT(\*) AS payment\_count,

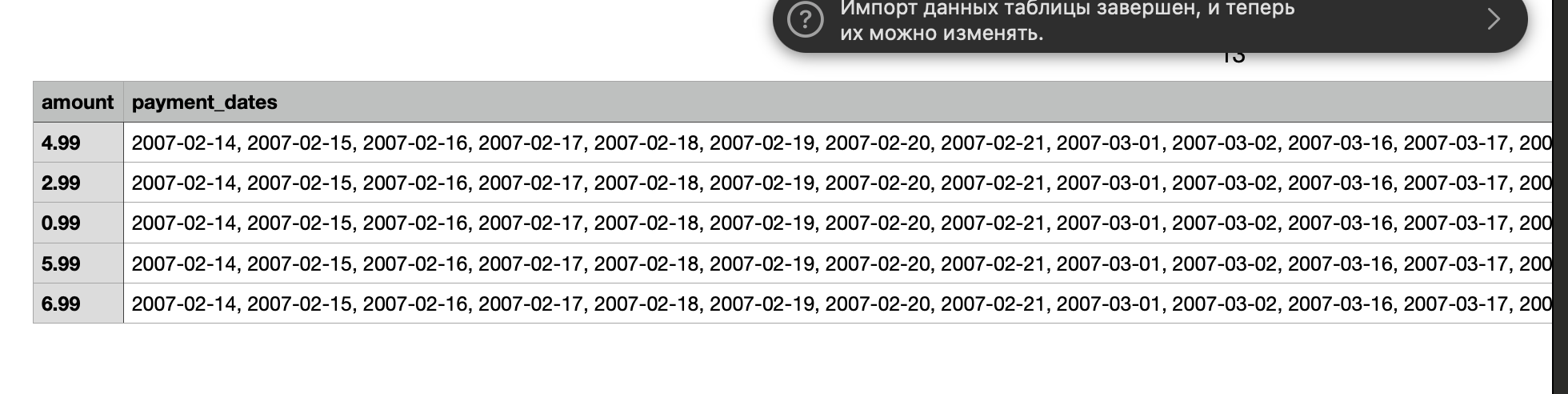
SUM(amount) AS total\_amount

FROM payment

GROUP BY amount

ORDER BY payment\_count DESC

LIMIT 5;



-- 3.14 Вывести количество ячеек (записей) в инвентаре для каждого магазина

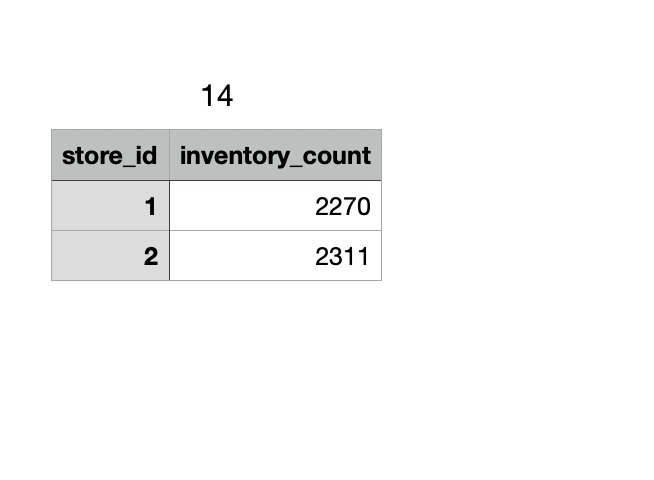
SELECT

store\_id,

COUNT(\*) AS inventory\_count

FROM inventory

GROUP BY store\_id;



-- 3.15 Вывести адреса всех магазинов, используя соединение таблиц (JOIN)

SELECT

s.store\_id,

a.address,

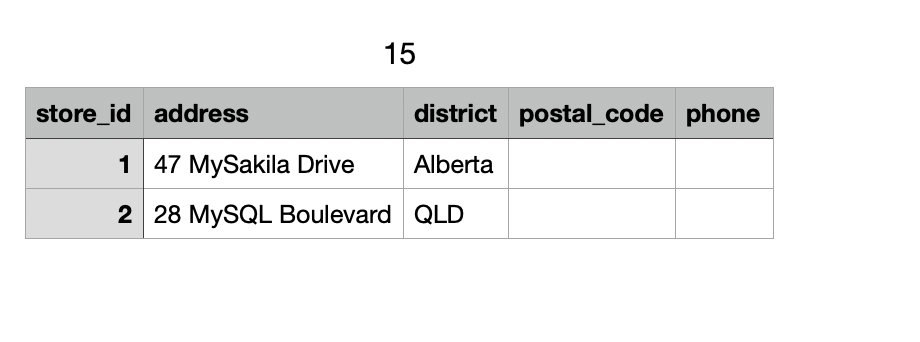
a.district,

a.postal\_code,

a.phone

FROM store s

JOIN address a ON s.address\_id = a.address\_id;



-- 3.16 Вывести полные имена всех клиентов и всех сотрудников в одну колонку

-- (объединенный список)

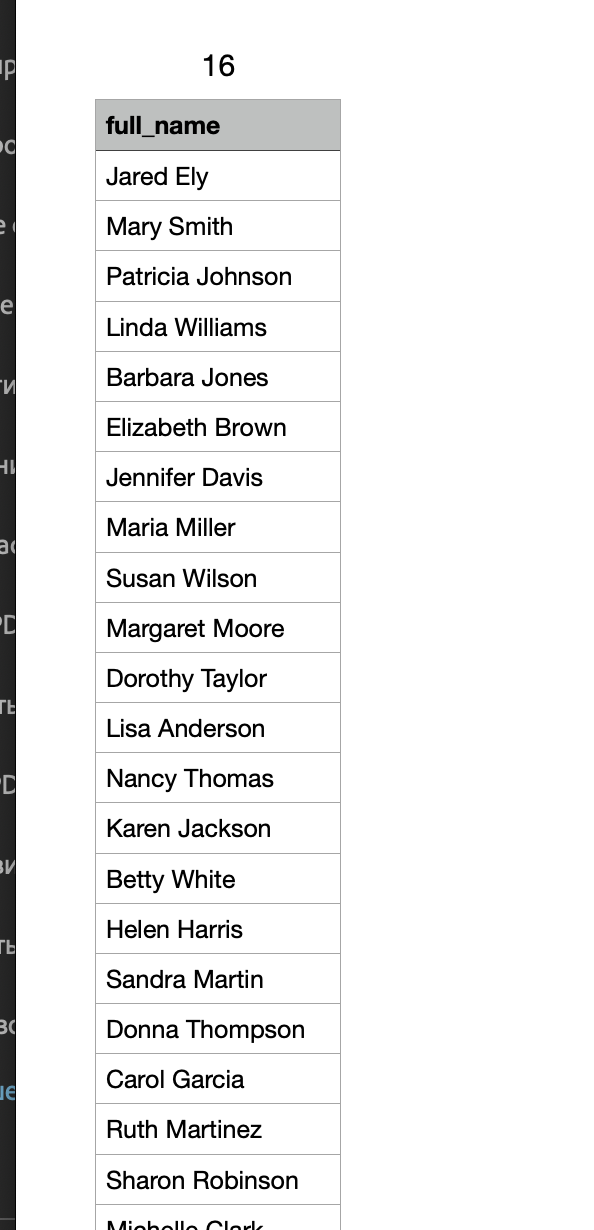
SELECT first\_name || ' ' || last\_name AS full\_name

FROM customer

UNION ALL

SELECT first\_name || ' ' || last\_name AS full\_name

FROM staff;



-- 3.17 Вывести имена клиентов, которые не совпадают ни с одним именем сотрудников

-- (операция EXCEPT или аналог)

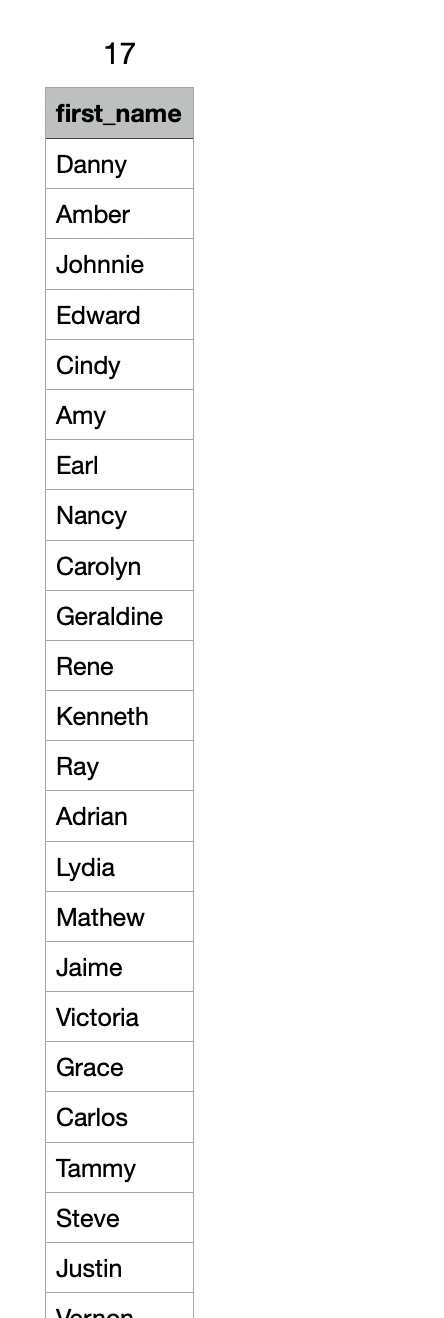
SELECT DISTINCT first\_name

FROM customer

EXCEPT

SELECT DISTINCT first\_name

FROM staff;



-- 3.18 Вывести, кто (customer\_id), когда (rental\_date, приведенная к типу date)

-- и у кого (staff\_id) брал диски в аренду в июне 2005 года

SELECT

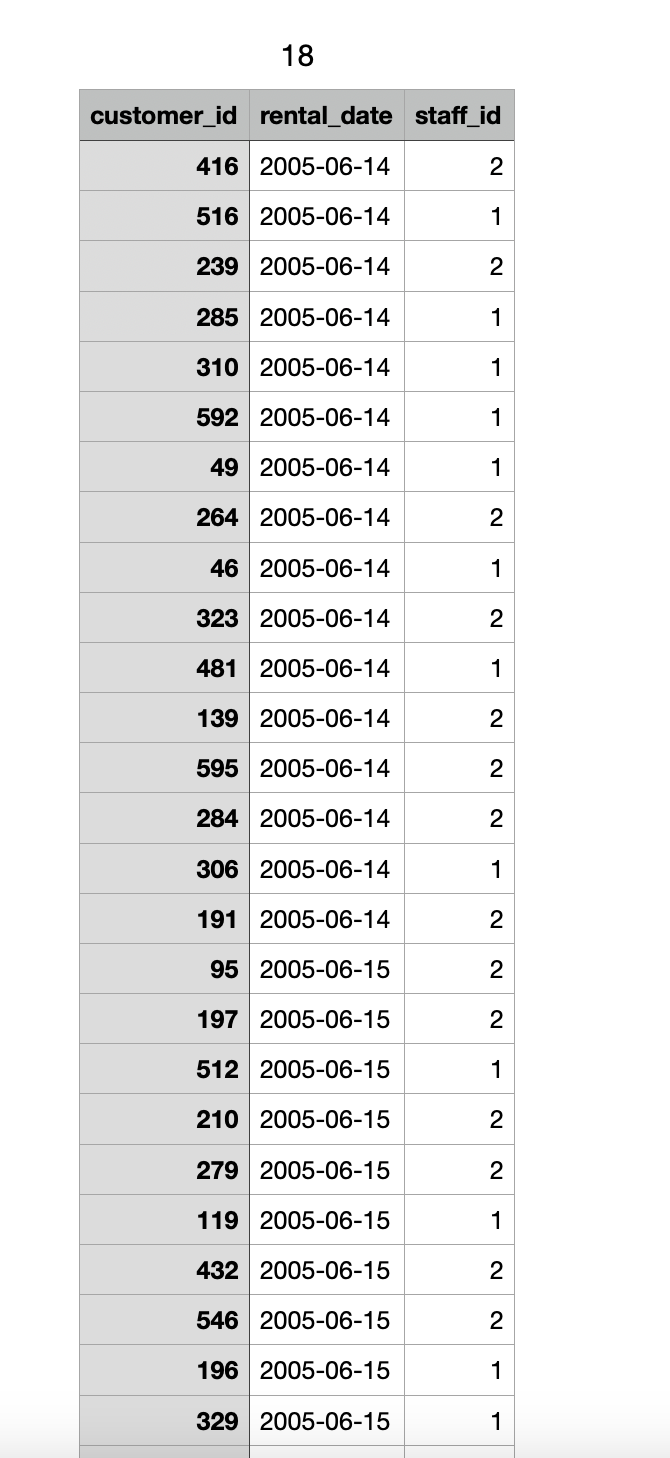
customer\_id,

rental\_date::date AS rental\_date,

staff\_id

FROM rental

WHERE rental\_date >= '2005-06-01' AND rental\_date < '2005-07-01';



-- 3.19 Вывести идентификаторы всех клиентов, у которых 40 и более оплат.

-- Для каждого такого клиента посчитать средний размер транзакции,

-- округлить его до двух знаков после запятой и вывести в отдельном столбце

SELECT

customer\_id,

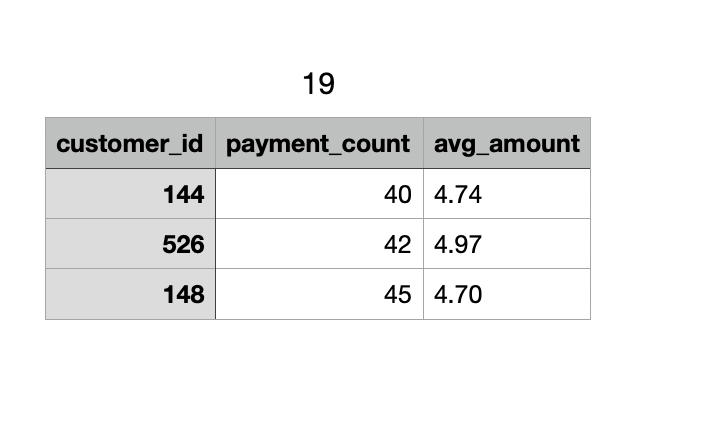
COUNT(\*) AS payment\_count,

ROUND(AVG(amount), 2) AS avg\_amount

FROM payment

GROUP BY customer\_id

HAVING COUNT(\*) >= 40;



-- 3.20 Вывести идентификатор актера, его полное имя и количество фильмов,

-- в которых он снялся. Определить актера, снявшегося в наибольшем количестве фильмов

-- (группировать по id актера)

SELECT

a.actor\_id,

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

COUNT(fa.film\_id) AS film\_count

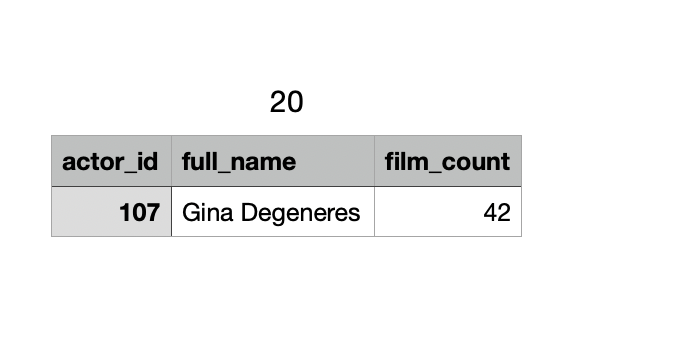
FROM actor a

JOIN film\_actor fa ON a.actor\_id = fa.actor\_id

GROUP BY a.actor\_id, a.first\_name, a.last\_name

ORDER BY film\_count DESC

LIMIT 1;



-- 3.21 Посчитать выручку по каждому месяцу работы проката.

-- Месяц должен определяться по дате аренды (rental\_date), а не по дате оплаты (payment\_date).

-- Округлить выручку до одного знака после запятой. Отсортировать строки в хронологическом порядке.

-- В отчете должен присутствовать месяц, в который не было выручки (нет данных о платежах)

WITH rental\_months AS (

SELECT DISTINCT

DATE\_TRUNC('month', rental\_date)::date AS rental\_month

FROM rental

)

SELECT

rm.rental\_month,

ROUND(COALESCE(SUM(p.amount), 0), 1) AS revenue

FROM rental\_months rm

LEFT JOIN rental r ON DATE\_TRUNC('month', r.rental\_date)::date = rm.rental\_month

LEFT JOIN payment p ON r.rental\_id = p.rental\_id

GROUP BY rm.rental\_month

ORDER BY rm.rental\_month;



-- 3.22 Найти средний платеж по каждому жанру фильма.

-- Отобразить только те жанры, к которым относится более 60 различных фильмов.

-- Округлить средний платеж до двух знаков после запятой и дать понятные названия столбцам.

-- Отсортировать жанры по убыванию среднего платежа

SELECT

c.name AS genre,

COUNT(DISTINCT fc.film\_id) AS film\_count,

ROUND(AVG(p.amount), 2) AS avg\_payment

FROM category c

JOIN film\_category fc ON c.category\_id = fc.category\_id

JOIN inventory i ON fc.film\_id = i.film\_id

JOIN rental r ON i.inventory\_id = r.inventory\_id

JOIN payment p ON r.rental\_id = p.rental\_id

GROUP BY c.name

HAVING COUNT(DISTINCT fc.film\_id) > 60

ORDER BY avg\_payment DESC;



-- 3.23 Определить, какие фильмы чаще всего берут напрокат по субботам.

-- Вывести названия первых 5 самых популярных фильмов.

-- При одинаковой популярности отдать предпочтение фильму, который идет раньше по алфавиту

SELECT

f.title,

COUNT(\*) AS rental\_count

FROM film f

JOIN inventory i ON f.film\_id = i.film\_id

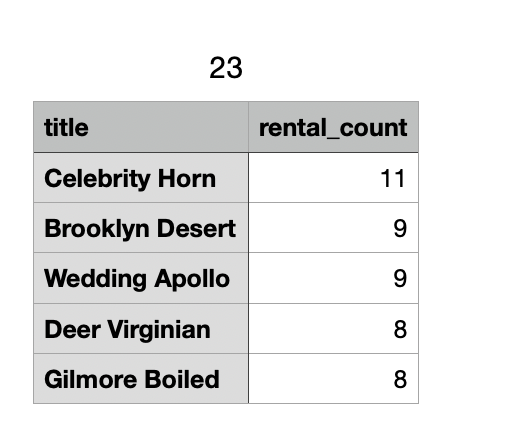
JOIN rental r ON i.inventory\_id = r.inventory\_id

WHERE EXTRACT(ISODOW FROM r.rental\_date) = 6

GROUP BY f.title

ORDER BY rental\_count DESC, f.title

LIMIT 5;



-- ===================================================

-- 4. Оконные функции и простые запросы

-- ===================================================

-- 4.24 Для каждой оплаты вывести сумму, дату и день недели (название дня недели текстом)

SELECT

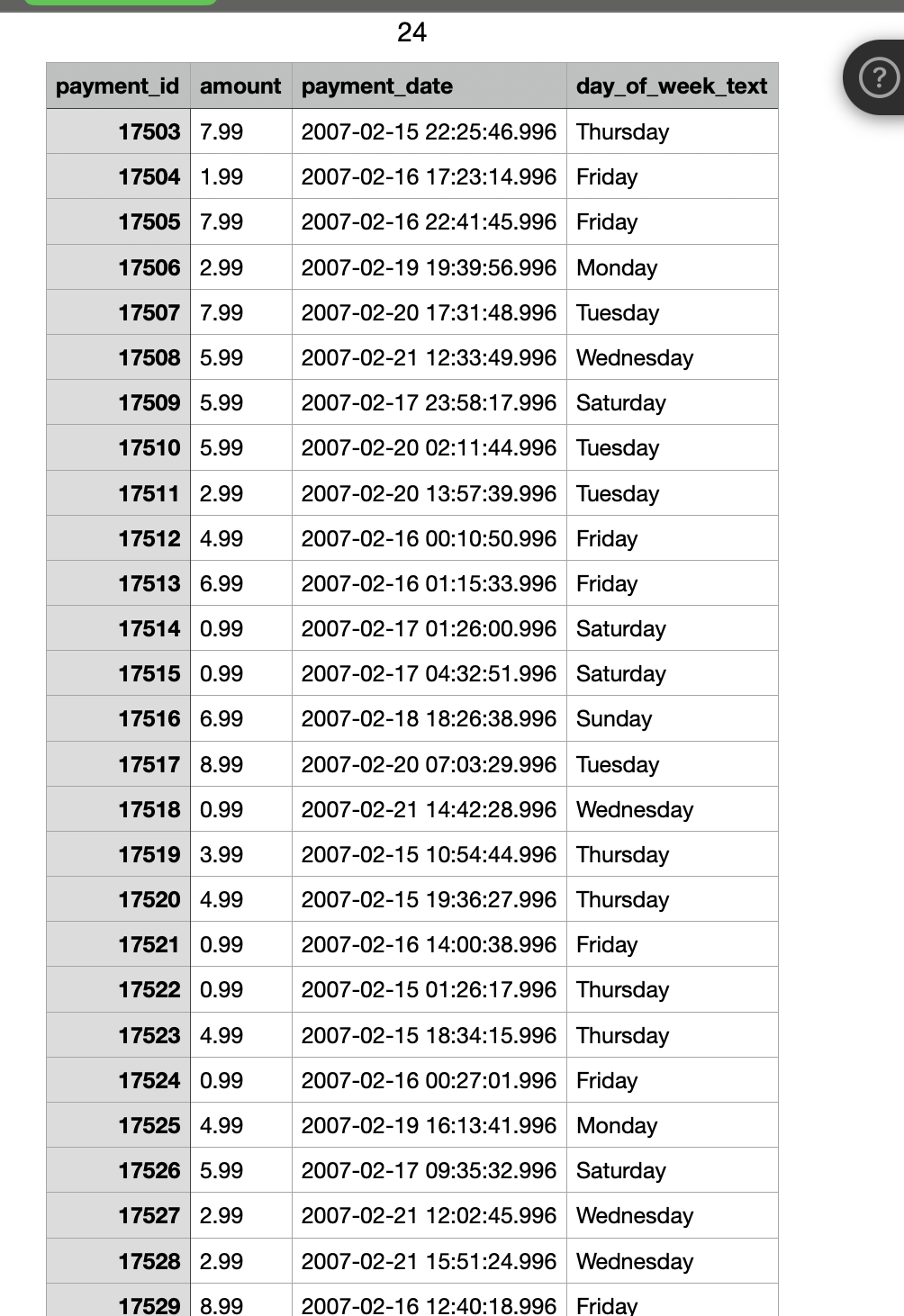
payment\_id,

amount,

payment\_date,

TO\_CHAR(payment\_date, 'Day') AS day\_of\_week\_text

FROM payment;



-- 4.25.1 Для каждой оплаты вывести: сумму платежа, дату платежа,

-- день недели в текстовом виде (например: «понедельник», «вторник» и т.п.)

SELECT

payment\_id,

amount,

payment\_date,

CASE EXTRACT(ISODOW FROM payment\_date)

WHEN 1 THEN 'понедельник'

WHEN 2 THEN 'вторник'

WHEN 3 THEN 'среда'

WHEN 4 THEN 'четверг'

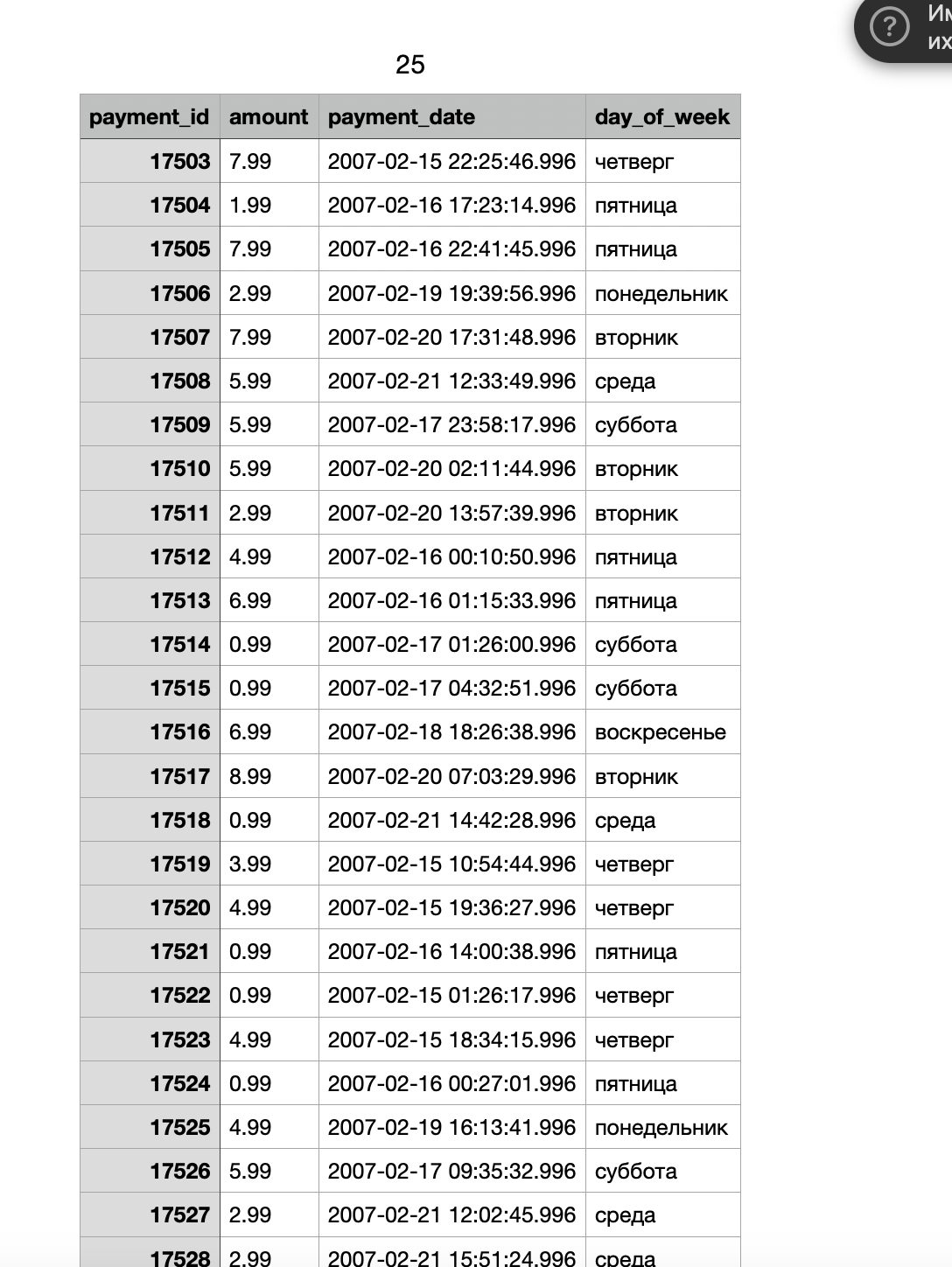
WHEN 5 THEN 'пятница'

WHEN 6 THEN 'суббота'

WHEN 7 THEN 'воскресенье'

END AS day\_of\_week

FROM payment;



-- 4.25.2 Распределить фильмы по трем категориям в зависимости от длительности:

-- «Короткие» --- менее 70 минут;

-- «Средние» --- от 70 минут (включительно) до 130 минут (не включая 130);

-- «Длинные» --- от 130 минут и более.

-- Для каждой категории посчитать количество прокатов и количество фильмов

SELECT

CASE

WHEN f.length < 70 THEN 'Короткие'

WHEN f.length >= 70 AND f.length < 130 THEN 'Средние'

ELSE 'Длинные'

END AS duration\_category,

COUNT(r.rental\_id) AS rental\_count,

COUNT(DISTINCT f.film\_id) AS film\_count

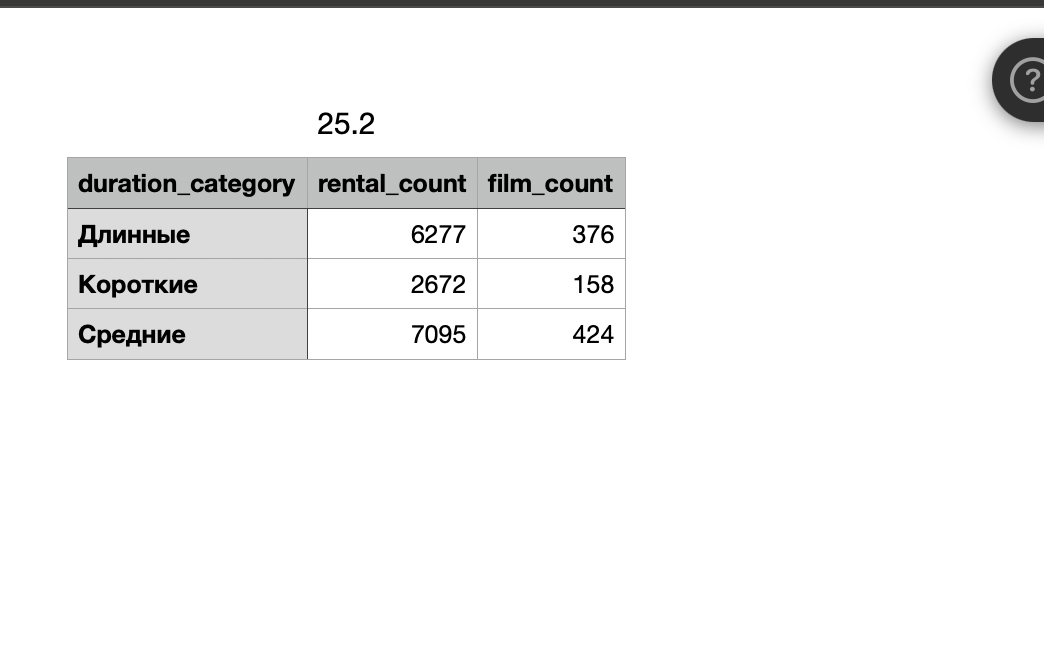
FROM film f

JOIN inventory i ON f.film\_id = i.film\_id

JOIN rental r ON i.inventory\_id = r.inventory\_id

GROUP BY duration\_category

ORDER BY duration\_category;

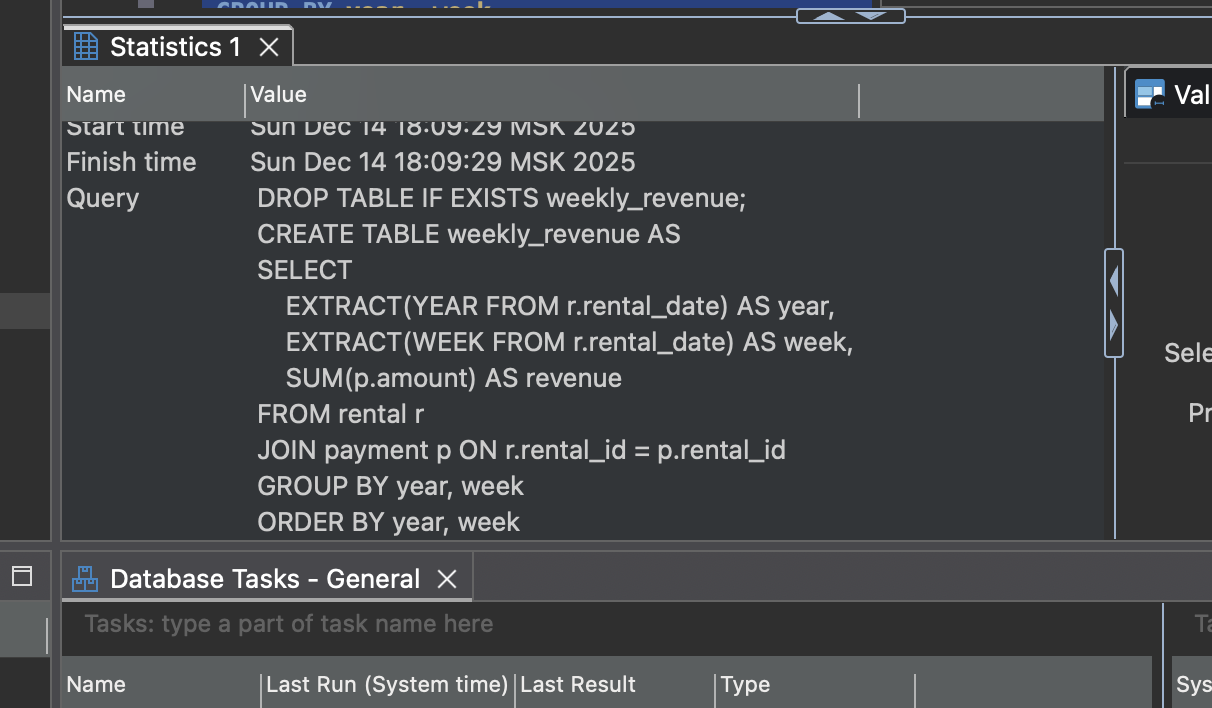


-- 4.26 На основе таблицы weekly\_revenue рассчитать накопленную (кумулятивную)

-- сумму недельной выручки бизнеса

-- (Предполагаем, что таблица weekly\_revenue уже создана.

-- Если её нет, нужно создать перед выполнением этого задания)



-- Пример создания таблицы weekly\_revenue (если её нет):

DROP TABLE IF EXISTS weekly\_revenue;

CREATE TABLE weekly\_revenue AS

SELECT

EXTRACT(YEAR FROM r.rental\_date) AS year,

EXTRACT(WEEK FROM r.rental\_date) AS week,

SUM(p.amount) AS revenue

FROM rental r

JOIN payment p ON r.rental\_id = p.rental\_id

GROUP BY year, week

ORDER BY year, week;

SELECT

year,

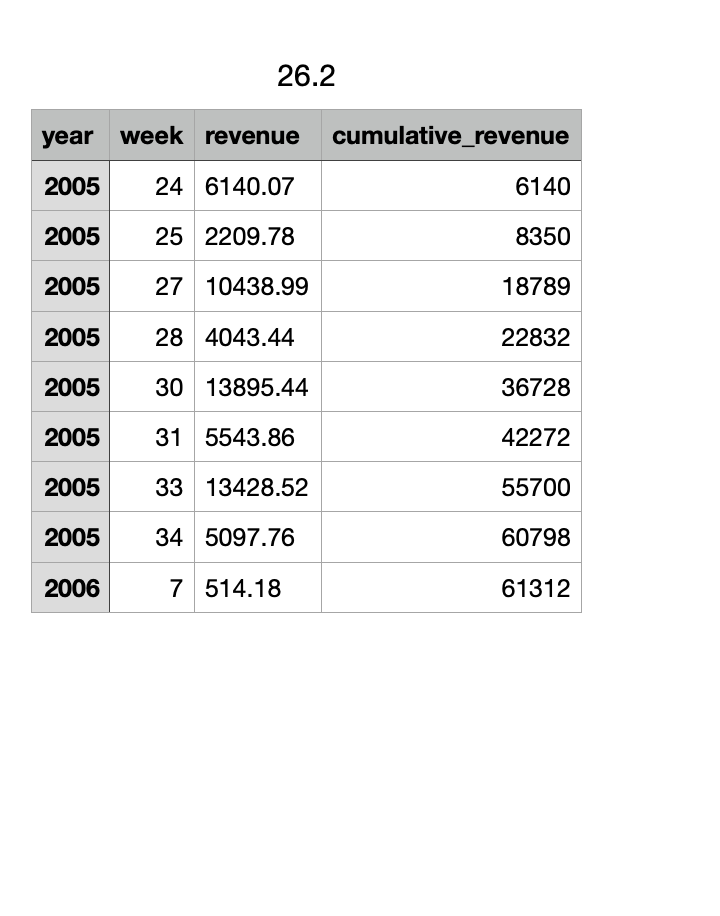
week,

revenue,

ROUND(SUM(revenue) OVER (ORDER BY year, week)) AS cumulative\_revenue

FROM weekly\_revenue

ORDER BY year, week;



-- 4.27 На основе таблицы weekly\_revenue рассчитать скользящую среднюю недельной выручки,

-- используя для расчета три недели: предыдущую, текущую и следующую

SELECT

year,

week,

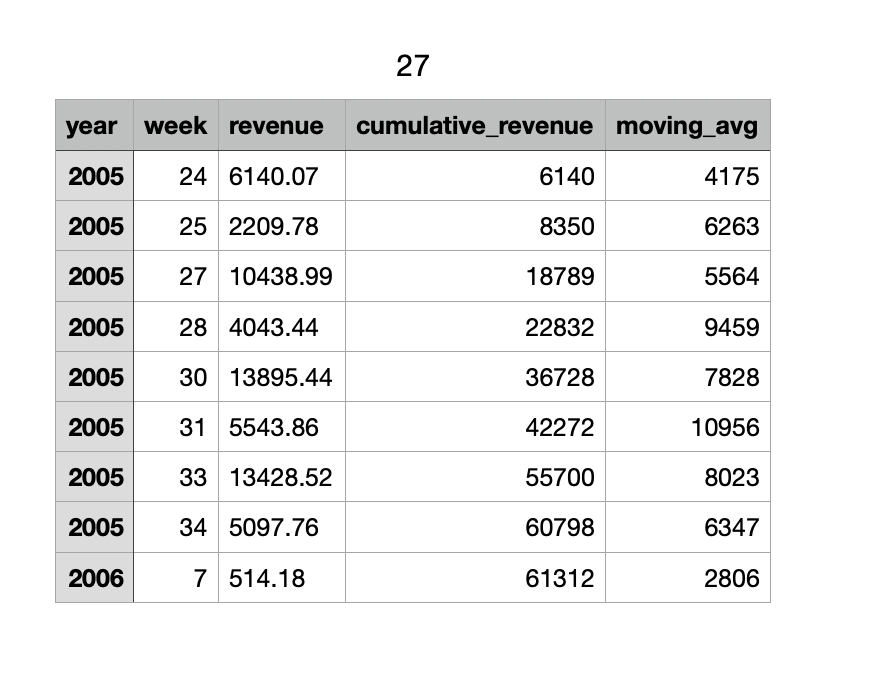
revenue,

ROUND(SUM(revenue) OVER (ORDER BY year, week)) AS cumulative\_revenue,

ROUND(AVG(revenue) OVER (ORDER BY year, week ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING)) AS moving\_avg

FROM weekly\_revenue

ORDER BY year, week;



-- 4.28 Рассчитать прирост недельной выручки бизнеса в процентах

-- по сравнению с предыдущей неделей

SELECT

year,

week,

revenue,

ROUND(SUM(revenue) OVER (ORDER BY year, week)) AS cumulative\_revenue,

ROUND(AVG(revenue) OVER (ORDER BY year, week ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING)) AS moving\_avg,

ROUND(

(revenue - LAG(revenue) OVER (ORDER BY year, week)) /

LAG(revenue) OVER (ORDER BY year, week) \* 100,

2

) AS week\_over\_week\_growth\_pct

FROM weekly\_revenue

ORDER BY year, week;

