Sakila Video — Rentals and Revenue

Introductory SQL: filtering, joins, and aggregation over the Sakila rental database.

Instructions

Answer each of the 8 questions below with a single SQL query using the Sakila database. Every table reference must be schema-qualified (e.g., sakila.film) and every column reference must be table-qualified (e.g., film.title) — do not use aliases or bare column names. Submit a single .sql file named Lastname_Firstname_SQL_Assignment.sql with your queries clearly numbered using comments (e.g., -- Question 1). Queries that do not run or produce incorrect results will receive partial or no credit depending on the rubric.

Scenario

Sakila Video is a growing DVD rental company operating across multiple store locations. Management needs data-driven insights to understand which films are most profitable, which customers are most valuable, and how inventory is performing across stores. As a junior data analyst, you have been tasked with writing SQL queries against the company's operational database to answer key business questions.

Database Structure

The Sakila database models a video rental business. The film table contains movie details (title, rating, rental rate, length). Films are linked to actor via film_actor, and to category via film_category. Inventory tracks physical copies per store. Customers rent inventory items (rental table) and payments are recorded in payment. Staff and store tables represent the physical locations.

Questions

1. Retrieve the title, rental_rate, and rating of all films that have a rental rate greater than $3.00. Order the results by rental_rate in descending order. *(8 pts)*

2. Find the first name, last name, and email of all customers whose last name starts with the letter 'S'. Order the results alphabetically by last name, then by first name. *(8 pts)*

3. List the title, length, and replacement_cost of all films that are longer than 120 minutes and have a replacement cost less than $20.00. Order the results by length in ascending order. *(8 pts)*

4. Retrieve the first name, last name, and email of all active staff members (where active = 1). Order the results by last name alphabetically. *(8 pts)*

5. For each film category, find the category name and the total number of films in that category. Order the results by the total number of films in descending order. *(12 pts)*

> *Hint: Join the category and film_category tables, then group by category to count the films in each one.*

6. List each customer's full name (first and last) along with the total amount they have paid across all their payments. Only include customers who have made at least one payment. Order the results by total amount paid in descending order, showing the top 10 customers. *(14 pts)*

Hint: Join customer and payment tables, then use an aggregate function on the payment amount grouped by customer.

---

7. Find the actor's first name, last name, and the total number of films they have appeared in. Only include actors who have appeared in more than 30 films. Order the results by the number of films in descending order. *(14 pts)*

Hint: After grouping by actor and counting their films, use HAVING to filter out actors below the threshold.

---

8. For each film category, calculate the average rental rate of films in that category and the total rental revenue generated (sum of all payment amounts) from films in that category. Only include categories where the total rental revenue exceeds $5,000. Display the category name, average rental rate (rounded to 2 decimal places), and total revenue, ordered by total revenue descending. *(28 pts)*

Hint: You will need to join several tables to connect categories all the way through to payments. Think about the path: category → film_category → film → inventory → rental → payment.
