-- 1. Rank the customers based on the total amount they've spent on rentals.

SELECT customer\_id, first\_name, last\_name, email, total\_amount\_spent,

RANK() OVER (ORDER BY total\_amount\_spent DESC) AS customer\_rank

FROM (

SELECT c.customer\_id, c.first\_name, c.last\_name, c.email,

SUM(p.amount) AS total\_amount\_spent

FROM customers c

JOIN payments p ON c.customer\_id = p.customer\_id

GROUP BY c.customer\_id, c.first\_name, c.last\_name, c.email

) AS customer\_spending;

-- 2. Calculate the cumulative revenue generated by each film over time.

SELECT film\_id, title, release\_year, rental\_date,

SUM(amount) OVER (PARTITION BY film\_id ORDER BY rental\_date) AS cumulative\_revenue

FROM rentals

JOIN payments USING(rental\_id)

JOIN inventory USING(inventory\_id)

JOIN films USING(film\_id);

-- 3. Determine the average rental duration for each film, considering films with similar lengths.

SELECT film\_id, title, length, AVG(rental\_duration) AS avg\_rental\_duration

FROM (

SELECT film\_id, title, length,

EXTRACT(DAY FROM AVG(return\_date - rental\_date)) AS rental\_duration

FROM rentals

JOIN inventory USING(inventory\_id)

JOIN films USING(film\_id)

GROUP BY film\_id, title, length

) AS film\_avg\_rental\_duration;

-- 4. Identify the top 3 films in each category based on their rental counts.

WITH ranked\_films AS (

SELECT film\_id, title, category\_id,

ROW\_NUMBER() OVER (PARTITION BY category\_id ORDER BY rental\_count DESC) AS film\_rank

FROM (

SELECT f.film\_id, f.title, f.category\_id, COUNT(\*) AS rental\_count

FROM films f

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

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

GROUP BY f.film\_id, f.title, f.category\_id

) AS film\_rental\_counts

)

SELECT category\_id, title, rental\_count

FROM (

SELECT rf.category\_id, rf.title, rental\_count,

RANK() OVER (PARTITION BY rf.category\_id ORDER BY rf.film\_rank) AS category\_rank

FROM ranked\_films rf

WHERE rf.film\_rank <= 3

) AS top\_films

WHERE category\_rank <= 3;

-- 5. Calculate the difference in rental counts between each customer's total rentals and the average rentals across all customers.

SELECT c.customer\_id, c.first\_name, c.last\_name, total\_rentals,

total\_rentals - avg\_rentals AS rental\_count\_difference

FROM (

SELECT customer\_id, COUNT(\*) AS total\_rentals

FROM rentals

GROUP BY customer\_id

) AS customer\_total\_rentals

JOIN customers c ON customer\_total\_rentals.customer\_id = c.customer\_id

CROSS JOIN (

SELECT AVG(total\_rentals) AS avg\_rentals

FROM (

SELECT COUNT(\*) AS total\_rentals

FROM rentals

GROUP BY customer\_id

) AS customer\_rentals

) AS average\_rentals;

-- 6. Find the monthly revenue trend for the entire rental store over time.

SELECT EXTRACT(MONTH FROM rental\_date) AS rental\_month,

EXTRACT(YEAR FROM rental\_date) AS rental\_year,

SUM(amount) AS monthly\_revenue

FROM payments

GROUP BY rental\_month, rental\_year

ORDER BY rental\_year, rental\_month;

-- 7. Identify the customers whose total spending on rentals falls within the top 20% of all customers.

SELECT customer\_id, first\_name, last\_name, email, total\_amount\_spent

FROM (

SELECT customer\_id, first\_name, last\_name, email, total\_amount\_spent,

NTILE(5) OVER (ORDER BY total\_amount\_spent DESC) AS customer\_group

FROM (

SELECT c.customer\_id, c.first\_name, c.last\_name, c.email,

SUM(p.amount) AS total\_amount\_spent

FROM customers c

JOIN payments p ON c.customer\_id = p.customer\_id

GROUP BY c.customer\_id, c.first\_name, c.last\_name, c.email

) AS customer\_spending

) AS top\_customers

WHERE customer\_group = 1;

-- 8. Calculate the running total of rentals per category, ordered by rental count.

SELECT category\_id, title, rental\_count,

SUM(rental\_count) OVER (PARTITION BY category\_id ORDER BY rental\_count DESC) AS running\_total\_rentals

FROM (

SELECT f.category\_id, f.title, COUNT(\*) AS rental\_count

FROM films f

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

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

GROUP BY f.category\_id, f.title

) AS category\_rental\_counts;

-- 9. Find the films that have been rented less than the average rental count for their respective categories.

SELECT f.film\_id, f.title, f.category\_id, rental\_count

FROM (

SELECT film\_id, COUNT(\*) AS rental\_count

FROM inventory

JOIN rentals USING(inventory\_id)

GROUP BY film\_id

) AS film\_rental\_counts

JOIN films f ON film\_rental\_counts.film\_id = f.film\_id

WHERE rental\_count < (

SELECT AVG(rental\_count)

FROM (

SELECT category\_id, COUNT(\*) AS rental\_count

FROM inventory

JOIN rentals USING(inventory\_id)

JOIN films USING(film\_id)

GROUP BY category\_id

) AS category\_rental\_counts

WHERE f.category\_id = category\_rental\_counts.category\_id

);

-- 10. Identify the top 5 months with the highest revenue and display the revenue generated in each month.

SELECT rental\_month, rental\_year, monthly\_revenue

FROM (

SELECT EXTRACT(MONTH FROM rental\_date) AS rental\_month,

EXTRACT(YEAR FROM rental\_date) AS rental\_year,

SUM(amount) AS monthly\_revenue,

RANK() OVER (ORDER BY SUM(amount) DESC) AS revenue\_rank

FROM payments

GROUP BY rental\_month, rental\_year

) AS top\_months

WHERE revenue\_rank <= 5;