Assignment: Windows Functions | Pw Skills | Manish Kumar

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

```
SELECT
    c.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
    SUM(p.amount) AS total_spent,
    RANK() OVER (ORDER BY SUM(p.amount) DESC) AS rank
FROM customer c
JOIN payment p ON c.customer_id = p.customer_id
GROUP BY c.customer_id, customer_name;
```

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

```
SELECT
    f.film_id,
    f.title,
    p.payment_date,
    SUM(p.amount) OVER (PARTITION BY f.film_id ORDER BY p.payment_date) AS
cumulative_revenue
FROM film f
JOIN inventory i ON f.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;
```

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

```
SELECT
    f.film_id,
    f.title,
    f.length,
    AVG(DATEDIFF(r.return_date, r.rental_date)) AS avg_rental_duration
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 f.film_id, f.title, f.length
ORDER BY f.length;
```

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

```
SELECT
   c.name AS category_name,
   f.title,
   COUNT(r.rental_id) AS rental_count,
   RANK() OVER (PARTITION BY c.name ORDER BY COUNT(r.rental_id) DESC) AS
rank
FROM category c
JOIN film_category fc ON c.category_id = fc.category_id
JOIN film f ON fc.film_id = f.film_id
JOIN inventory i ON f.film_id = i.film_id
JOIN rental r ON i.inventory_id = r.inventory_id
GROUP BY c.name, f.title
HAVING rank <= 3;
```

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

```
WITH customer_rentals AS (
   SELECT
      c.customer_id,
      CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
      COUNT(r.rental_id) AS total_rentals
   FROM customer c
   JOIN rental r ON c.customer_id = r.customer_id
   GROUP BY c.customer_id, customer_name
),
avg_rentals AS (
    SELECT AVG(total_rentals) AS avg_total_rentals FROM customer_rentals
)
SELECT
    cr.customer_id,
    cr.customer_name,
    cr.total_rentals,
    cr.total_rentals - ar.avg_total_rentals AS rental_difference
FROM customer_rentals cr, avg_rentals ar;
```

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

```
SELECT
    DATE_FORMAT(payment_date, '%Y-%m') AS month,
    SUM(amount) AS total_revenue
FROM payment
GROUP BY month
ORDER BY month;
```

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

```
WITH customer_spending AS (
    SELECT
        c.customer_id,
        CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
        SUM(p.amount) AS total_spent,
        NTILE(5) OVER (ORDER BY SUM(p.amount) DESC) AS spending_group
    FROM customer c
    JOIN payment p ON c.customer_id = p.customer_id
    GROUP BY c.customer_id, customer_name
)
SELECT
```

```
    customer_id,
    customer_name,
    total_spent
FROM customer_spending
WHERE spending_group = 1;
```

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

```
SELECT
    c.name AS category_name,
    COUNT(r.rental_id) AS rental_count,
    SUM(COUNT(r.rental_id)) OVER (PARTITION BY c.name ORDER BY
COUNT(r.rental_id)) AS running_total
FROM category c
JOIN film_category fc ON c.category_id = fc.category_id
JOIN film f ON fc.film_id = f.film_id
JOIN inventory i ON f.film_id = i.film_id
JOIN rental r ON i.inventory_id = r.inventory_id
GROUP BY c.name;
```

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

```
WITH category_avg AS (
    SELECT
        c.name AS category_name,
        f.film_id,
        f.title,
        COUNT(r.rental_id) AS rental_count,
        AVG(COUNT(r.rental_id)) OVER (PARTITION BY c.name) AS avg_rental_count
    FROM category c
    JOIN film_category fc ON c.category_id = fc.category_id
    JOIN film f ON fc.film_id = f.film_id
    JOIN inventory i ON f.film_id = i.film_id
    JOIN rental r ON i.inventory_id = r.inventory_id
    GROUP BY c.name, f.film_id, f.title
)
SELECT
    film_id,
    title,
    category_name,
    rental_count
FROM category_avg
WHERE rental_count < avg_rental_count;
```

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

```
SELECT
    DATE_FORMAT(payment_date, '%Y-%m') AS month,
    SUM(amount) AS total_revenue
FROM payment
GROUP BY month
ORDER BY total_revenue DESC
LIMIT 5;
```