Windows Function

Answer no. 1

Rank the customers based on the total amount they've spent on rentals:

```
SELECT c.customer_id, c.first_name, c.last_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, c.first_name, c.last_name;
```

Answer no. 2

Calculate the cumulative revenue generated by each film over time:

```
SELECT f.title, SUM(p.amount) AS total_revenue,
    SUM(SUM(p.amount)) OVER (PARTITION BY f.film_id ORDER BY r.rental_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
GROUP BY f.title, f.film_id, r.rental_date;
```

Answer no. 3

Determine the average rental duration for each film, considering films with similar lengths:

```
SELECT f.title, AVG(rental_duration) AS average_duration
FROM film f
GROUP BY f.film_id
HAVING COUNT(f.film_id) > 1;  -- You can adjust this condition based on your criteria for "similar lengths"
```

Answer no. 4

Identify the top 3 films in each category based on their rental counts:

```
WITH rental_counts AS (
  SELECT f.film_id, f.title, fc.category_id, COUNT(r.rental_id) 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
  JOIN film_category fc ON f.film_id = fc.film_id
  GROUP BY f.film_id, f.title, fc.category_id
),
ranked_films AS (
  SELECT *,
      RANK() OVER (PARTITION BY category_id ORDER BY rental_count DESC) AS rank
  FROM rental_counts
)
SELECT film_id, title, category_id, rental_count
FROM ranked_films
WHERE rank <= 3;
```

Answer no. 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, COUNT(r.rental_id) AS total_rentals
  FROM customer c
  LEFT JOIN rental r ON c.customer_id = r.customer_id
  GROUP BY c.customer_id
),
average_rentals AS (
  SELECT AVG(total_rentals) AS avg_rentals
  FROM customer_rentals
)
SELECT cr.customer_id, cr.total_rentals,
    cr.total_rentals - ar.avg_rentals AS difference
FROM customer_rentals cr, average_rentals ar;
```

Answer no. 6

Find the monthly revenue trend for the entire rental store over time:

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

Answer no. 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, SUM(p.amount) AS total_spent
   FROM customer c
   JOIN payment p ON c.customer_id = p.customer_id
   GROUP BY c.customer_id
),
spending_rank AS (
   SELECT total_spent,
        NTILE(5) OVER (ORDER BY total_spent DESC) AS spending_quartile
   FROM customer_spending
)
SELECT cs.customer_id
FROM customer_spending cs
JOIN spending_rank sr ON cs.total_spent = sr.total_spent
WHERE sr.spending_quartile = 1;  -- Top 20%
```

Answer no. 8

Calculate the running total of rentals per category, ordered by rental count:

```
WITH rental_counts AS (
   SELECT fc.category_id, COUNT(r.rental_id) AS rental_count
   FROM rental r
   JOIN inventory i ON r.inventory_id = i.inventory_id
   JOIN film_category fc ON i.film_id = fc.film_id
   GROUP BY fc.category_id
)
SELECT category_id, rental_count,
     SUM(rental_count) OVER (ORDER BY rental_count) AS running_total
FROM rental_counts;
```

Answer no. 9

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

```
WITH category_rental_counts AS (
   SELECT fc.category_id, f.title, COUNT(r.rental_id) 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
   JOIN film_category fc ON f.film_id = fc.film_id
   GROUP BY fc.category_id, f.title
),
average_counts AS (
   SELECT category_id, AVG(rental_count) AS avg_rental_count
   FROM category_rental_counts
   GROUP BY category_id
)
SELECT crc.title
FROM category_rental_counts crc
JOIN average_counts ac ON crc.category_id = ac.category_id
WHERE crc.rental_count < ac.avg_rental_count;
```

Answer no. 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;
```