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

```
SELECT
customer_id,
SUM(amount_spent) AS total_amount_spent,
RANK() OVER (ORDER BY SUM(amount_spent) DESC) AS customer_rank
FROM
rentals
GROUP BY
customer_id
ORDER BY
total_amount_spent DESC;
```

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

```
SELECT
film_id,
rental_date,
amount_spent,
SUM(amount_spent) OVER (PARTITION BY film_id ORDER BY rental_date) AS
cumulative_revenue
FROM
rentals
ORDER BY
```

```
    film_id, rental_date;
```

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

```
    SELECT
    film_id,
    rental_duration,
    AVG(rental_duration) OVER (PARTITION BY rental_duration) AS avg_rental_duration
    FROM
    rentals;
```

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

```
    WITH RankedFilms AS (
    SELECT
        film_id, category_id,
      ROW_NUMBER() OVER (PARTITION BY category_id ORDER BY COUNT(*) DESC) AS film_rank
    FROM
    rentals
      JOIN films ON rentals.film_id = films.film_id
   GROUP BY
      film_id, category_id
```

```
    )
SELECT
    film_id, category_id, film_rank
FROM
    RankedFilms
WHERE
    film_rank <= 3;
```

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

```
  WITH CustomerRentalStats AS (
  SELECT
      customer_id,
      COUNT(*) AS total_rentals,
      AVG(COUNT(*)) OVER () AS average_rentals
  FROM
     rentals
  GROUP BY
      customer_id)
  SELECT
      customer_id,
```

```
    total_rentals,
    average_rentals,
    total_rentals - average_rentals AS rental_count_difference
FROM
    CustomerRentalStats;
```

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

```
SELECT
    EXTRACT(YEAR_MONTH FROM rental_date) AS year_month,
    SUM(amount_spent) AS monthly_revenue,
    SUM(SUM(amount_spent)) OVER (ORDER BY EXTRACT(YEAR_MONTH FROM rental_date)) AS cumulative_revenue
FROM
    rentals
GROUP BY
    EXTRACT(YEAR_MONTH FROM rental_date)
ORDER BY
    year_month;
```

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

```
WITH CustomerTotalSpending AS (
SELECT
    customer_id,
    SUM(amount_spent) AS total_spending
FROM
    rentals
GROUP BY
    customer_id)
SELECT
    customer_id, total_spending,
    PERCENT_RANK() OVER (ORDER BY total_spending DESC) AS
spending_percent_rank
FROM
    CustomerTotalSpending
WHERE
    spending_percent_rank <= 0.2;
```

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

```
WITH CategoryRentalCounts AS (
SELECT
    category_id,
    COUNT(*) AS rental_count
```

```
 FROM
     rentals
     JOIN films ON rentals.film_id = films.film_id
 GROUP BY
      category_id)
 SELECT
     category_id, rental_count,
     SUM(rental_count) OVER (ORDER BY rental_count DESC) AS running_total
 FROM
    CategoryRentalCounts
 ORDER BY
     rental_count DESC;
```

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

```
 WITH FilmRentalCounts AS (
 SELECT
     f.film_id, f.title, f.category_id,
     COUNT(r.rental_id) AS rental_count,
     AVG(COUNT(r.rental_id)) OVER (PARTITION BY f.category_id) AS avg_rental_count
 FROM
    films f
     JOIN rentals r ON f.film_id = r.film_id
```

```
GROUP BY
    f.film_id, f.title, f.category_id)
SELECT
    film_id, title, category_id, rental_count,
    avg_rental_count
FROM
   FilmRentalCounts
WHERE
    rental_count < avg_rental_count;
```

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

```
WITH MonthlyRevenue AS (
SELECT
    EXTRACT(YEAR_MONTH FROM rental_date) AS year_month,
    SUM(amount_spent) AS monthly_revenue
FROM
    rentals
GROUP BY
    year_month)
SELECT
   year_month, monthly_revenue FROM (
SELECT
```

```
    year_month, monthly_revenue,
    RANK() OVER (ORDER BY monthly_revenue DESC) AS revenue_rank
FROM
    MonthlyRevenue) ranked_revenues
WHERE
    revenue_rank <= 5
ORDER BY
    revenue_rank;
```