SQL 문제 1

고객의 지불 내역에서 각 고객별로 해당 고객의 지불 금액에 따른 순위를 출력하세요. (고객 ID, rental ID, 해당 고객의 지불 금액에 따른 순위를 출력하되, 동일한 값이 있을 경우 같은 순위를 부여하지만 다음 순위는 건너뛰지 않습니다)

```
SELECT
  customer_id,
  payment_id,
  amount,
  DENSE_RANK() OVER (PARTITION BY customer_id ORDER BY amount DESC) AS
amount_rank
FROM
  payment;
```

SQL 문제 2

영화 대여 내역에서 각 고객별로 해당 고객의 대여날짜시간 순으로 정렬하여, 고객 ID, rental ID와 함께, 각 대여날짜시간, 바로 다음 대여날짜시간을 출력하세요.

```
SELECT
  customer_id, rental_id, rental_date,
    LEAD(rental_date) OVER
    (PARTITION BY customer_id ORDER BY rental_date) AS next_rental_date
FROM rental;
```

SQL 문제 3

영화 정보에서 각 등급별로 대여 기간(rental_duration)이 가장 긴 영화의 제목을 출력하세요.

```sql
SELECT
  DISTINCT rating,
    FIRST_VALUE(title) OVER
    (PARTITION BY rating ORDER BY rental_duration DESC)
        AS longest_rental_movie
FROM film;
```

SQL 문제 4

고객 정보에서 각 고객을 활동 상태(active)가 높은 순으로 정렬하고, 이를 기준으로 상위 1/3, 중간 1/3, 하위 1/3의 세 그룹으로 나누고, 각 그룹 내에서 고객의 순서를 customer_id 가 낮은 순으로 정렬해서 다음과 같은 항목으로 출력하세요.
출력항목: customer_id, first_name, last_name, active, active_group, group_row_number(각 그룹 내에서 고객의 순서를 customer_id 가 낮은 순으로 정렬하였을 때의 행 번호)

```sql
WITH RankedCustomers AS (
  SELECT customer_id, first_name, last_name, active,
    NTILE(3) OVER (ORDER BY active DESC) AS active_group
    FROM customer
)
SELECT
  customer_id, first_name, last_name, active, active_group,
    ROW_NUMBER() OVER (PARTITION BY active_group ORDER BY customer_id)
    AS group_row_number
FROM RankedCustomers
```

SQL 문제 5

영화 대여 내역에서 고객별로 대여 순서(rental_date 순으로 정렬), 이전 대여와의 간격(DAY 기반), 첫 번째 대여 일시(가장 rental_date 가 오래된 일시)를 다음과 같은 항목으로 출력하세요.
출력항목: customer_id, rental_id, rental_date, rental_order, prev_rental_gap, first_rental_date

```sql
SELECT
  r.customer_id,
  r.rental_id,
  r.rental_date,
  ROW_NUMBER() OVER (PARTITION BY r.customer_id ORDER BY r.rental_date)
AS rental_order,
  DATEDIFF(r.rental_date, LAG(r.rental_date) OVER (PARTITION BY
r.customer_id ORDER BY r.rental_date)) AS prev_rental_gap,
  FIRST_VALUE(r.rental_date) OVER (PARTITION BY r.customer_id ORDER BY
r.rental_date
                                   ROWS BETWEEN UNBOUNDED PRECEDING AND
UNBOUNDED FOLLOWING) AS first_rental_date
FROM
  rental r;
```

SQL 문제 6

고객의 지불 내역에서 각 고객의 총 지불 금액에 따른 순위(지불금액이 높은 순, 동일한 값이 있을 경우 같은 순위를 부여하지만 다음 순위는 건너뛰지 않음)와 백분위 순위(지불금액이 높은 순)를 출력하세요.

```sql
WITH payment_info AS (
  SELECT
    p.customer_id,
    SUM(p.amount) AS total_amount
  FROM
    payment p
  GROUP BY
    p.customer_id
)
SELECT
  customer_id,
  total_amount,
  DENSE_RANK() OVER (ORDER BY total_amount DESC) AS total_amount_rank,
  PERCENT_RANK() OVER (ORDER BY total_amount DESC) AS
total_amount_pct_rank
FROM
  payment_info;
```

SQL 문제 7

영화 정보에서 각 등급별로 영화를 대여 기간에 따라 4개의 그룹으로 나누고, 각 그룹 내에서 rental_duration 이 낮은 순으로 영화의 순서를 다음 항목으로 출력하세요.
film_id, title, rating, rental_duration, rental_duration_group,group_row_number

```sql
WITH FilmGroups AS (
  SELECT film_id, title, rating, rental_duration,
    NTILE(4) OVER (PARTITION BY rating ORDER BY rental_duration)
      AS rental_duration_group
    FROM film
)
SELECT
  film_id, title, rating, rental_duration, rental_duration_group,
    ROW_NUMBER() OVER (PARTITION BY rental_duration_group ORDER BY
rental_duration)
    AS group_row_number
FROM FilmGroups
```

SQL 문제 8

배우 정보에서 각 배우의 출연 영화 수에 따른 누적 분포를 다음과 같이 출력하세요.
actor_id, first_name, last_name, film_count, film_count_cume_dist

```sql
WITH actor_film_info AS (
  SELECT
    a.actor_id,
    a.first_name,
    a.last_name,
    COUNT(*) AS film_count
  FROM
    actor a
    JOIN film_actor fa ON a.actor_id = fa.actor_id
    JOIN film f ON fa.film_id = f.film_id
  GROUP BY
    a.actor_id, a.first_name, a.last_name
)
SELECT
  actor_id,
  first_name,
  last_name,
  film_count,
```

```
  CUME_DIST() OVER (ORDER BY film_count) AS film_count_cume_dist
FROM
  actor_film_info;
```

SQL 문제 9

Sakila 데이터베이스의 영화 대여 내역을 바탕으로 다음 항목들을 출력하는 SQL 쿼리를 작성하세요.
1. 고객별 대여 순위 (rental_rank): 고객별로 대여 내역을 rental_date가 빠른 순서대로 정렬한 후, 각 대여에 대한 순위를 매깁니다.
2. 이전 대여와의 간격 (previous_rental_gap): 각 대여일을 기준으로 이전 대여일과의 간격을 일(DAY) 단위로 계산합니다. (단, 고객별로 계산)
3. 다음 대여와의 간격 (next_rental_gap): 각 대여일을 기준으로 다음 대여일과의 간격을 일(DAY) 단위로 계산합니다. (단, 고객별로 계산)
4. 고객별 첫 번째 및 마지막 대여 일자 (first_last_rental): 고객별로 대여 내역을 rental_date 기준으로 정렬한 후, 첫 번째 대여일과 마지막 대여일을 출력합니다.
5. 고객별 대여 건의 백분위 순위 및 누적 분포 (rental_percentile_rank, rental_cumulative_dist): 고객별로 대여 내역을 rental_date 기준으로 정렬한 후, 각 대여 건에 대해 백분위 순위와 누적 분포를 계산합니다.
6. 고객별 대여 내역의 3개 그룹 분할 (rental_group): 고객별 대여 내역을 최대한 균등하게 3개의 그룹으로 나눈 후, 각 대여 건이 속한 그룹 값을 출력합니다. (그룹 번호는 1 ~ 3)
7. 고객과 그룹별 대여 순서 (group_rental_rank): 고객별로 대여 내역을 rental_date 기준으로 정렬한 후, 각 그룹 내에서 대여 건에 대해 순차적인 순서를 매깁니다. (순서는 1부터 시작)
위의 항목들을 customer_id, rental_date 와 함께 모두 포함하여 출력하는 SQL 쿼리를 작성하세요.

최종 출력 항목: customer_id, rental_date, rental_rank, previous_rental_gap, next_rental_gap, first_rental_date, last_rental_date, rental_percentile_rank, rental_cumulative_dist, rental_group, group_rental_rank

```
WITH rental_data AS (
  SELECT
    rental_id, customer_id, rental_date,
        ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY rental_date)
AS rental_rank,
    LAG(rental_date) OVER (PARTITION BY customer_id ORDER BY rental_date)
AS previous_rental_date,
        LEAD(rental_date) OVER (PARTITION BY customer_id ORDER BY
rental_date) AS next_rental_date,
        FIRST_VALUE(rental_date) OVER (PARTITION BY customer_id ORDER BY
rental_date) AS first_rental_date,
```

```
        LAST_VALUE(rental_date) OVER (PARTITION BY customer_id ORDER BY
rental_date
      ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS
last_rental_date,
    PERCENT_RANK() OVER (PARTITION BY customer_id ORDER BY rental_date)
AS rental_percentile_rank,
        CUME_DIST() OVER (PARTITION BY customer_id ORDER BY rental_date)
AS rental_cumulative_dist,
        NTILE(3) OVER (PARTITION BY customer_id ORDER BY rental_date) AS
rental_group
    FROM rental
),
rental_intervals AS (
  SELECT rental_id, customer_id, rental_date, rental_rank,
first_rental_date, last_rental_date,
    rental_percentile_rank, rental_cumulative_dist, rental_group,
        DATEDIFF(rental_date, previous_rental_date) AS
previous_rental_gap,
        DATEDIFF(next_rental_date, rental_date) AS next_rental_gap
    FROM rental_data
),
grouped_rental_rank AS (
  SELECT rental_id, customer_id, rental_date, rental_group,
    ROW_NUMBER() OVER (PARTITION BY customer_id, rental_group ORDER BY
rental_date) AS group_rental_rank
    FROM rental_data
)
SELECT
  R.customer_id, R.rental_date, R.rental_rank, R.previous_rental_gap,
R.next_rental_gap,
    R.first_rental_date, R.last_rental_date, R.rental_percentile_rank,
R.rental_cumulative_dist,
    R.rental_group, G.group_rental_rank
FROM rental_intervals R
JOIN grouped_rental_rank G ON G.rental_id = R.rental_id
```

SQL 문제 10

Sakila 데이터베이스의 결제 내역을 바탕으로 다음 항목들을 출력하는 SQL 쿼리를 작성하세요.
1. 고객별 누적 결제 금액 (cumulative_amount): 고객별 결제 내역을 payment_date 기준으로 정렬한 후, 누적 결제 금액을 계산합니다.
2. 이전 결제 금액 차이 (prev_payment_diff): 고객별로 현재 결제 금액과 이전 결제 금액의 차이를 계산합니다.

3. 다음 결제 금액 차이 (next_payment_diff): 고객별로 다음 결제 금액과 현재 결제 금액의 차이를 계산합니다.
4. 고객별 첫 번째 및 마지막 결제 일자 (first_payment_date, last_payment_date): 고객별로 결제 내역을 payment_date 기준으로 정렬한 후, 첫 번째 결제일과 마지막 결제일을 출력합니다.
5. 전체 결제 금액에 따른 그룹 (total_amount_group): 누적 결제 금액을 기준으로 전체 결제 내역을 5개의 그룹으로 나누고, 각 결제가 속한 그룹을 출력합니다.
6. 총 결제 금액 순위 (total_amount_rank): 누적 결제 금액을 기준으로 내림차순으로 정렬하여 순위를 매깁니다. 이때, 동일한 누적 결제 금액을 가진 고객이 있을 경우 동일 순위를 부여하고, 그 다음 순위는 건너 뛰지 않고, 연속된 순위로 부여합니다.
7. 고객별 결제 금액의 백분위 순위 및 누적 분포 (payment_amount_pct_rank, payment_amount_cume_dist): 고객별 결제 금액을 기준으로 백분위 순위와 누적 분포를 계산합니다.
8. 그룹 내 결제 건 순서 (group_row_number): 결제 금액 그룹별로 결제 내역을 누적 결제 금액 기준으로 내림차순 정렬한 후 순서를 매깁니다.
위의 항목들을 customer_id, payment_date 와 함께 모두 포함하여 출력하는 SQL 쿼리를 작성하세요.

최종 출력 항목: customer_id, payment_date, cumulative_amount, prev_payment_diff, next_payment_diff, first_payment_date, last_payment_date, total_amount_rank, payment_amount_pct_rank, payment_amount_cume_dist, total_amount_group, group_row_number

```
WITH payment_info AS (
  SELECT customer_id, payment_date, amount,
    SUM(amount) OVER (PARTITION BY customer_id ORDER BY payment_date) AS
cumulative_amount,
        amount - LAG(amount) OVER (PARTITION BY customer_id ORDER BY
payment_date) AS prev_payment_diff,
        LEAD(amount) OVER (PARTITION BY customer_id ORDER BY
payment_date) - amount AS next_payment_diff,
        FIRST_VALUE(payment_date) OVER (PARTITION BY customer_id ORDER BY
payment_date) AS first_payment_date,
        LAST_VALUE(payment_date) OVER (PARTITION BY customer_id ORDER BY
payment_date
      ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS
last_payment_date
  FROM payment
),
payment_group_info AS (
  SELECT *,
    NTILE(5) OVER (ORDER BY cumulative_amount) AS total_amount_group
```

```
    FROM payment_info
)
SELECT
  customer_id, payment_date, cumulative_amount, prev_payment_diff,
next_payment_diff,
    first_payment_date, last_payment_date,
    DENSE_RANK() OVER (ORDER BY cumulative_amount DESC) AS
total_amount_rank,
    PERCENT_RANK() OVER (PARTITION BY customer_id ORDER BY amount) AS
payment_amount_pct_rank,
    CUME_DIST() OVER (PARTITION BY customer_id ORDER BY amount) AS
payment_amount_cume_dist,
    total_amount_group,
    ROW_NUMBER() OVER (PARTITION BY total_amount_group ORDER BY
cumulative_amount DESC) AS group_row_number
FROM payment_group_info
```