Sakila 데이터로 연습하는 SQL 데이터 분석 (중급)

연습문제1

category 테이블에서 "Comedy", "Sports", "Family" 카테고리의 category_id 알아내기 (category_id 와 카테고리 명 출력하기)

```
SELECT name, category_id FROM category WHERE name = 'Comedy' OR name =
'Sports' OR name = 'Family'
```

연습문제2

film_category 테이블에서 영화 아이디(film_id)가 2 인 영화의 카테고리 ID 알아내기

단계별로 진행하세요: 1단계 - film_category 테이블 컬럼 확인하기

```
SELECT * FROM film_category LIMIT 1
```

단계별로 진행하세요: 2단계 - film_id 가 2 인 영화 데이터(행) 확인하기

```
SELECT category_id FROM film_category WHERE film_id = 2
```

연습문제3

film_category 테이블에서 카테고리 ID별 영화 수 알아내기

```
SELECT category_id, COUNT(*) FROM film_category GROUP BY category_id;
```

연습문제4

카테고리가 Comedy 인 영화 수 알아내기 (JOIN, 서브쿼리 각각 작성)
(category 테이블에는 카테고리 이름과 category_id, film_category 테이블에는 category_id와 각 영화 id가 있음)

단계별로 진행하세요: 1단계 - category 테이블 컬럼 확인하기

```
SELECT * FROM category LIMIT 1
```

단계별로 진행하세요: 2단계 - Comedy 의 category_id 알아내기

```
SELECT * FROM category WHERE name = 'Comedy'
```

단계별로 진행하세요: 3단계 - category 와 film_category 테이블을 JOIN 해서, Comedy 카테고리 영화 데이터만 출력하기

```
SELECT * FROM film_category A
    JOIN category B
    ON B.category_id = A.category_id
    WHERE B.name = 'Comedy'
```

단계별로 진행하세요: 4단계 - category 와 film_category 테이블을 JOIN 해서, Comedy 카테고리 영화 데이터 수 출력하기

```
SELECT COUNT(*) FROM film_category
    JOIN category
    ON category.category_id = film_category.category_id
    WHERE category.name = 'Comedy'
```

서브쿼리로 작성시

```
SELECT COUNT(*) FROM film_category
WHERE category_id IN
  (SELECT category_id  FROM category WHERE name="Comedy");
```

연습문제5

카테고리가 Comedy, Sports, Family 인 영화 수 알아내기 (JOIN 사용하기)
(category 테이블에는 카테고리 이름과 category_id, film_category 테이블에는 category_id와 각 영화 id가 있음)

단계별로 진행하세요: 1단계 - Comedy, Sports, Family 인 총 영화 수 알아내기

```
SELECT COUNT(*) FROM film_category
    JOIN category
    ON category.category_id = film_category.category_id
    WHERE category.name = 'Comedy' OR category.name = 'Sports' OR
category.name = 'Family'
```

단계별로 진행하세요: 2단계 - Comedy, Sports, Family 인 각각의 영화 수 알아내기

```
SELECT COUNT(*) FROM film_category
    JOIN category
    ON category.category_id = film_category.category_id
    WHERE category.name = 'Comedy' OR category.name = 'Sports' OR
category.name = 'Family'
    GROUP BY category.category_id
```

단계별로 진행하세요: 3단계 - Comedy, Sports, Family 인 각각의 영화 수를 각각의 카테고리 이름과 함께 출력하기

```
SELECT category.name, COUNT(*) FROM film_category
    JOIN category
    ON category.category_id = film_category.category_id
    WHERE category.name = 'Comedy' OR category.name = 'Sports' OR
category.name = 'Family'
    GROUP BY category.category_id
```

단계별로 진행하세요: 4단계 - Comedy, Sports, Family 인 각각의 영화 수를 각각의 카테고리 이름과 함께 출력하되, 영화 수 컬럼명을 film_count 로 변경해서 출력하기

```sql
SELECT category.name, COUNT(*) AS film_count FROM film_category
    JOIN category
    ON category.category_id = film_category.category_id
    WHERE category.name = 'Comedy' OR category.name = 'Sports' OR
category.name = 'Family'
    GROUP BY category.category_id
```

단계별로 진행하세요: 5단계 - 4단계에서 작성한 SQL에서, 단 각각의 테이블명을 간결히 변경해서, SQL 구문을 간결하게 바꿔보세요.

```sql
SELECT C.name, COUNT(*) AS film_count FROM film_category F
    JOIN category C
    ON C.category_id = F.category_id
    WHERE C.name = 'Comedy' OR C.name = 'Sports' OR C.name = 'Family'
    GROUP BY C.category_id
```

연습문제6 (HAVING 문법을 사용해서 작성해보세요!)

카테고리에 포함되는 각각의 영화 수가 70 이상인 카테고리명을 출력하기

```sql
SELECT C.name, COUNT(*) AS film_count FROM film_category F
    JOIN category C
    ON C.category_id = F.category_id
    GROUP BY C.category_id
    HAVING COUNT(*) > 70
```

연습문제7

각 카테고리에 포함된 영화들의 렌탈 횟수 구해보기 (각 카테고리별에 포함된 영화들의 총 렌탈 횟수와 각 카테고리명을 출력하는 것이 최종 목표)

- rental 테이블에 렌탈 기록이 있음
- inventory 테이블에 물품현황과 해당 물품(DVD)에 들어 있는 영화 아이디가 있음

- film_category 테이블에 영화 아이디에 매칭되는 카테고리 아이디가 있음
- category 테이블에 카테고리 아이디에 매칭되는 카테고리명이 있음

단계별로 진행하세요: 1단계 - rental, inventory, film_category, category 테이블 확인하기

단계별로 진행하세요: 2단계 - rental, inventory, film_category, category 테이블 연결고리로 연결하기

```
SELECT * FROM rental
    JOIN inventory ON rental.inventory_id = inventory.inventory_id
    JOIN film_category ON inventory.film_id = film_category.film_id
    JOIN category ON film_category.category_id = category.category_id
    LIMIT 1
```

단계별로 진행하세요: 3단계 - category_id 로 그룹핑해서, 각 카테고리별 카운트값(렌탈 횟수)과 카테고리명을 출력하세요

```
SELECT category.name, COUNT(*) FROM rental
    JOIN inventory ON rental.inventory_id = inventory.inventory_id
    JOIN film_category ON inventory.film_id = film_category.film_id
    JOIN category ON film_category.category_id = category.category_id
    GROUP BY category.category_id
```

연습문제8

"Comedy", "Sports", "Family" 카테고리에 포함되는 영화들의 렌탈 횟수 출력하기 (카테코리명, 렌탈 횟수)
- rental 테이블에 렌탈 기록이 있음
- inventory 테이블에 물품현황과 해당 물품(DVD)에 들어 있는 영화 아이디가 있음
- film_category 테이블에 영화 아이디에 매칭되는 카테고리 아이디가 있음
- category 테이블에 카테고리 아이디에 매칭되는 카테고리명이 있음

```
SELECT category.name, COUNT(*) FROM rental
    JOIN inventory ON rental.inventory_id = inventory.inventory_id
    JOIN film_category ON inventory.film_id = film_category.film_id
    JOIN category ON film_category.category_id = category.category_id
    WHERE category.name = 'Comedy' OR category.name = 'Sports' OR
category.name = 'Family'
    GROUP BY category.category_id
```

연습문제9

카테고리가 Comedy 인 데이터의 렌탈 횟수 출력하기 (서브쿼리 문법으로 작성해보세요)

```
SELECT COUNT(*) FROM rental WHERE inventory_id IN
  (
    SELECT inventory_id FROM inventory WHERE film_id IN
      (
        SELECT film_id FROM film_category WHERE category_id IN
        (SELECT category_id FROM category WHERE name = 'Comedy')
      )
  )
```

연습문제10

카테고리가 Comedy 인 데이터의 영화 갯수 출력하기 (JOIN 문법으로 작성해보세요)

```
SELECT COUNT(*) FROM film_category
    JOIN category ON category.category_id = film_category.category_id
    WHERE category.name = 'Comedy'
```

연습문제11

카테고리가 Comedy 인 데이터의 영화 갯수 출력하기 (서브쿼리 문법으로 작성해보세요)

```
SELECT COUNT(*) FROM film_category
    WHERE film_category.category_id IN
        (SELECT category.category_id FROM category WHERE category.name =
'Comedy')
```

연습문제12

address 테이블에는 address_id 가 있지만, customer 테이블에는 없는 데이터의 갯수 출력하기 (RIGHT JOIN 문법으로 작성해보세요!)

```sql
SELECT COUNT(*) FROM customer C
RIGHT OUTER JOIN address A
ON A.address_id=C.address_id
WHERE customer_id IS NULL
```

연습문제13 (실전)

캐나다 고객에게 이메일 마케팅 캠페인을 진행하고자 합니다. 캐나다 고객의 이름과, email 주소 리스트를 뽑아주세요

```sql
SELECT
  first_name, last_name, email
FROM customer CU
JOIN address A ON CU.address_id=A.address_id
JOIN city CI ON CI.city_id=A.city_id
JOIN country CTR ON CTR.country_id=CI.country_id
WHERE CTR.country='canada';
```

연습문제14 (실전)

젊은 가족 사이에서 매출이 저조해서, 모든 가족 영화를 홍보 대상으로 삼으려고 합니다. 가족 영화로 분류된 모든 영화 리스트를 뽑아주세요

```sql
SELECT F.title
FROM film F
JOIN film_category FC ON F.film_id = FC.film_id
JOIN category CA ON FC.category_id = CA.category_id
WHERE CA.name = 'Family';
```

연습문제15 (실전)

가장 자주 대여하는 영화 리스트를 참고로 보고 싶습니다. 가장 자주 대여하는 영화 순으로 100개만 뽑아주세요
title (영화제목) 과 Rentals (렌탈 횟수) 로 보고 싶습니다.

```sql
SELECT
  title, COUNT(title) AS Rentals
FROM film
JOIN inventory ON (film.film_id = inventory.film_id)
JOIN rental ON (inventory.inventory_id = rental.inventory_id)
GROUP by title
ORDER BY rentals DESC
LIMIT 100
```

연습문제16 (실전)

각 스토어별로 매출을 확인하고 싶습니다. 관련 데이터를 출력해주세요
스토어가 위치한 '도시, 국가' 를 Store 항목으로, 스토어 ID 를 Store ID 로, 각 스토어별 총 매출을 'Total Sales' 항목으로 출력해주세요

```sql
SELECT
  CONCAT(ct.city, ', ', c.country) AS 'Store',
     st.store_id AS 'Store ID',
  SUM(amount) AS 'Total Sales'
FROM payment p
JOIN staff sf ON sf.staff_id = p.staff_id
JOIN store st ON st.store_id = sf.store_id
JOIN address a ON a.address_id = st.address_id
JOIN city ct ON ct.city_id = a.city_id
JOIN country c ON c.country_id = ct.country_id
GROUP BY st.store_id;
```

연습문제17 (실전)

각 스토어의 스토어 ID, 도시, 및 국가를 알고싶습니다. 관련 데이터를 출력해주세요
store_id, city, country 로 보고 싶습니다.

```sql
SELECT store_id, city, country
FROM store s
JOIN address a ON s.address_id = a.address_id
JOIN city cit ON cit.city_id = a.city_id
JOIN country ctr ON cit.country_id = ctr.country_id
```

연습문제18 (실전)

가장 렌탈비용을 많이 지불한 상위 10명에게 선물을 배송하고자 합니다
가장 렌탈비용을 많이 지불한 상위 10명의 주소(address)와 이메일, 그리고 각 고객당 총 지불 비용을 출력해주세요

```sql
SELECT
  CONCAT(c.first_name,' ',c.last_name) as customer_name,
    SUM(p.amount) as total_amount,
    a.address,
    c.email
FROM payment as p
JOIN customer as c ON p.customer_id = c.customer_id
JOIN address as a ON a.address_id = c.address_id
GROUP BY p.customer_id
ORDER BY sum(p.amount) DESC
LIMIT 10;
```

연습문제19 (실전)

actor 테이블의 배우 이름을 first name 과 last name 을 대문자로 Actor Name 항목으로 출력해주세요

```sql
SELECT UPPER(CONCAT(first_name,' ', last_name)) AS 'Actor Name' FROM
actor;
```

연습문제20 (실전)

언어가 영어 인 영화 중, 영화 타이틀이 K 와 Q 로 시작하는 영화의 타이틀만 출력해주세요 (서브쿼리를 사용해보세요)

```sql
SELECT title
FROM film
WHERE language_id IN (
  SELECT language_id FROM language
    WHERE name = 'English'
) AND (title LIKE 'K%' OR title LIKE 'Q%');
```

연습문제21 (실전)

Alone Trip 에 나오는 배우 이름을 모두 출력하세요 (배우 이름은 actor_name 항목으로 출력해주세요. 서브쿼리를 사용해보세요)

```sql
SELECT CONCAT(first_name,' ', last_name) as actor_name FROM actor
WHERE actor_id IN
(SELECT actor_id FROM film_actor WHERE film_id IN
(SELECT film_id FROM film WHERE title = 'Alone Trip'))
```

연습문제22 (실전)

2005년 8월에 각 스태프 멤버가 올린 매출을 스태프 멤버는 Staff Member 항목으로, 매출은 Total Amount 항목으로 출력해주세요.

```sql
SELECT
  CONCAT(s.first_name,' ',s.last_name) AS 'Staff Member',
    SUM(p.amount) AS 'Total Amount'
FROM payment p
JOIN staff s ON p.staff_id=s.staff_id
WHERE payment_date like '2005-08%'
GROUP BY p.staff_id;
```

```sql
SELECT
  CONCAT(S.first_name, ', ', S.last_name) AS 'Staff Member',
    SUM(P.amount) AS 'Total Amount'
FROM payment P
JOIN staff S ON P.staff_id = S.staff_id
WHERE
  EXTRACT(YEAR FROM P.payment_date) = 2005 AND
    EXTRACT(MONTH FROM P.payment_date) = 8
GROUP BY P.staff_id
```

연습문제23 (실전)

각 카테고리의 평균 영화 러닝타임이 전체 평균 러닝타임보다 큰 카테고리들의 카테고리명과 해당 카테고리의 평균 러닝타임을 출력하세요

```sql
SELECT CA.name, AVG(FI.length)
FROM film FI
JOIN film_category FC ON FI.film_id = FC.film_id
JOIN category CA ON CA.category_id = FC.category_id
GROUP BY CA.name
HAVING AVG(FI.length) > (
  SELECT AVG(length) FROM film
);
```

연습문제24 (실전)

각 카테고리별 평균 영화 대여 시간 (영화 대여 및 반납 시간의 차이, hour 단위) 과 해당 카테고리명을 출력하세요

```sql
SELECT
  CA.name,
  AVG(TIMESTAMPDIFF(HOUR, RE.rental_date, RE.return_date))
avg_rental_duration
FROM rental RE
JOIN inventory INV ON RE.inventory_id = INV.inventory_id
JOIN film_category FC ON INV.film_id = FC.film_id
JOIN category CA ON CA.category_id = FC.category_id
GROUP BY CA.name;
```

연습문제25 (실전)

새로운 임원이 부임했습니다. 총 매출액 상위 5개 장르의 매출액을 수시로 확인하고자 합니다.
Total Sales (각 장르별 총 매출액) 과 Genre (각 장르 이름) 으로 해당 데이터를 수시로 확인할 수 있는 view 를 top5_genres 로 만들고, 현재까지의 상위 5 장르의 매출액을 출력해주세요

```
CREATE OR REPLACE VIEW top5_genres AS
  SELECT C.name AS 'Genre', SUM(amount) AS 'Total Sales'
  FROM payment P
  JOIN rental R ON R.rental_id = P.rental_id
  JOIN inventory I ON I.inventory_id = R.inventory_id
  JOIN film_category F ON F.film_id = I.film_id
  JOIN category C ON C.category_id = F.category_id
  GROUP BY C.name
  ORDER BY SUM(amount) DESC
  LIMIT 5
```

```
SELECT * FROM top5_genres
```

연습문제26 (실전)

top5_genres view 를 삭제해주세요

```
DROP VIEW top5_genres
```

연습문제27 (실전)

2005년 5월에 가장 많이 대여된 영화 3개를 찾으세요. 영화 제목과 대여 횟수를 보여주세요.

```
SELECT
    f.title,
    COUNT(*) as rental_count
FROM
    rental r
```

```
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film f ON i.film_id = f.film_id
WHERE
    MONTH(r.rental_date) = 5 AND
    YEAR(r.rental_date) = 2005
GROUP BY
    f.title
ORDER BY
    rental_count DESC
LIMIT 3;
```

연습문제28 (실전)
대여된 적이 없는 영화를 찾으세요.

```
# 다음 쿼리는 하나의 영화가 여러 inventory 를 가졌을 때, 그 중 하나의 inventory 라도 대여되지 않았다면, 해당 영화 타이틀도 출력함
SELECT
    f.title
FROM
    film f
LEFT OUTER JOIN inventory i ON f.film_id = i.film_id
LEFT OUTER JOIN rental r ON i.inventory_id = r.inventory_id
WHERE
    r.rental_id IS NULL;
```

```sql
# 위 쿼리를 보완하기 위해, 다음과 같이 작성 가능
SELECT
    f.title
FROM
    film f
LEFT JOIN
    inventory i ON f.film_id = i.film_id
LEFT JOIN
    rental r ON i.inventory_id = r.inventory_id
GROUP BY
    f.title
HAVING
    COUNT(r.rental_id) = 0;
```

```sql
# 또는 다음과 같이 작성도 가능
SELECT title FROM film
WHERE film_id NOT IN (
    SELECT DISTINCT(I.film_id) FROM rental R
     JOIN inventory I ON R.inventory_id = I.inventory_id
);
```

연습문제29 (실전)

각 고객의 총 지출 금액의 평균 보다 총 지출 금액이 더 큰 고객 리스트를 찾으세요. 그들의 이름과 그들이 지출한 총 금액을 보여주세요.

각 고객의 총 지출 금액 계산 방법:

- 고객 A 5번 렌트, 총 100$
- 고객 B 3번 렌트, 총 80$
- 고객당 평균 지출: 90$

```sql
SELECT
    c.first_name,
    c.last_name,
    SUM(p.amount) as total_spent
FROM
    payment p
JOIN customer c ON p.customer_id = c.customer_id
GROUP BY
    c.customer_id
HAVING
    total_spent > (SELECT AVG(sum_amount) FROM (SELECT SUM(amount) as
sum_amount FROM payment GROUP BY customer_id) as sub_query);
```

연습문제30 (실전)

가장 많은 결제를 처리한 직원이 누구인지 찾으세요.

```sql
SELECT
    s.staff_id,
    s.first_name,
    s.last_name,
    COUNT(*) as payment_count
FROM
    staff s
JOIN payment p ON s.staff_id = p.staff_id
GROUP BY
    s.staff_id
ORDER BY
    payment_count DESC
LIMIT 1;
```

연습문제31 (실전)

'액션' 카테고리에서 높은 영화 영상 등급을 받은 순으로, 상위 5개의 영화를 보여주세요. (높은 영화 영상 등급 순으로의 정렬은 ORDER BY rating DESC 으로 처리 가능)

```sql
SELECT
    f.title,
    f.rating
FROM
    film f
JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c ON fc.category_id = c.category_id
WHERE
    c.name = 'Action'
ORDER BY
    f.rating DESC
LIMIT 5;
```

연습문제32 (실전)

각 영화 영상등급의 영화별 대여 기간(film.rental_duration)의 평균을 찾으세요.

```sql
SELECT
    f.rating,
    AVG(f.rental_duration) as avg_duration
FROM
    film f
GROUP BY
    f.rating;
```

연습문제33 (실전)

매장 ID별 총 매출을 보여주는 뷰를 생성하세요.

```sql
CREATE VIEW total_sales_by_store AS
SELECT
    s.store_id,
    SUM(p.amount) as total_sales
FROM
    store s
JOIN staff st ON s.store_id = st.store_id
JOIN payment p ON st.staff_id = p.staff_id
GROUP BY
    s.store_id;
```

연습문제34 (실전)

이전 연습문제에서 생성한 뷰를 삭제하세요.

```sql
DROP VIEW total_sales_by_store;
```

연습문제35 (실전)

가장 많은 고객이 있는 상위 5개 국가를 보여주세요.

```sql
SELECT
    co.country,
    COUNT(*) as customer_count
FROM
    country co
JOIN city ci ON co.country_id = ci.country_id
JOIN address a ON ci.city_id = a.city_id
JOIN customer cu ON a.address_id = cu.address_id
GROUP BY
    co.country
ORDER BY
    customer_count DESC
LIMIT 5;
```

연습문제36 (실전)

각 고객이 어떤 영화 카테고리를 가장 자주 대여하는지 알고 싶습니다. 각 고객별로 가장 많이 대여한 영화 카테고리와 해당 카테고리에서의 총 대여 횟수, 그리고 해당 고객 이름을 조회하는 SQL 쿼리를 작성해주세요. 자주 대여하는 카테고리에 동률이 있을 경우 모두 보여주세요.

```
SELECT
    c.first_name,
    c.last_name,
    cat.name as category_name,
    COUNT(*) as rental_count
FROM
    customer c
JOIN rental r ON c.customer_id = r.customer_id
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film_category fc ON i.film_id = fc.film_id
JOIN category cat ON fc.category_id = cat.category_id
GROUP BY
    c.customer_id,
    cat.name
HAVING
    COUNT(*) = (
        SELECT
            COUNT(*)
        FROM
            rental r2
        JOIN inventory i2 ON r2.inventory_id = i2.inventory_id
        JOIN film_category fc2 ON i2.film_id = fc2.film_id
        WHERE
            r2.customer_id = c.customer_id
        GROUP BY
            fc2.category_id
        ORDER BY
            COUNT(*) DESC
        LIMIT 1
    )
```

연습문제37 (실전)

2006-02-14 날짜를 기준으로, 2006-01-15일(이전 30일) 부터, 2006-02-14 날짜까지 영화를 대여하지 않은 고객을 찾으세요.

```
SELECT
    c.customer_id,
    c.first_name,
    c.last_name
FROM
    customer c
LEFT JOIN rental r ON c.customer_id = r.customer_id AND TIMESTAMPDIFF(DAY,
r.rental_date, '2006-02-14') <= 30
WHERE
    r.rental_id IS NULL;
```

연습문제38 (실전)

가장 최근에 영화를 반납한 상위 10명의 고객 이름과 그들이 대여한 영화의 이름, 그리고 대여 기간을 출력해 주세요. 고객 이름은 customer_name, 영화 이름은 movie_title, 대여 기간은 rental_duration으로 출력해주세요.

```
SELECT
    CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
    f.title AS movie_title,
    TIMESTAMPDIFF(DAY, r.rental_date, r.return_date) AS rental_duration
FROM
    rental r
JOIN
    customer c ON r.customer_id = c.customer_id
JOIN
    inventory i ON r.inventory_id = i.inventory_id
JOIN
    film f ON i.film_id = f.film_id
ORDER BY
    r.return_date DESC
LIMIT 10;
```

연습문제39 (실전)

각 직원의 매출을 찾고, 각 직원의 매출이 전체 매출 중 어느 정도 비율을 차지하는지 찾으세요. 결과는 직원 ID, 직원 이름, 총 매출, 전체 매출에 대한 비율(%)로 보여주세요.

```
SELECT
    s.staff_id,
    CONCAT(s.first_name, ' ', s.last_name) AS staff_name,
    SUM(p.amount) as staff_revenue,
    (SUM(p.amount) / (SELECT SUM(amount) FROM payment) * 100) as
revenue_percentage
FROM
    payment p
JOIN
    staff s ON p.staff_id = s.staff_id
GROUP BY
    s.staff_id;
```

연습문제40 (실전)

가장 많은 수의 종류가 다른(동일 영화를 반복하여 대여하지 않고) 영화를 대여한 고객을 찾아내고, 대여한 종류가 다른(동일 영화를 반복하여 대여하지 않고) 영화의 수를 확인하세요. 또한 해당 고객이 대여한 영화(여기서는 동일 영화 반복 대여도 가능)가 가장 많이 속해있는 카테고리를 찾아내세요.

```
SELECT
    cus.customer_id,
    CONCAT(cus.first_name, ' ', cus.last_name) AS customer_name,
    COUNT(DISTINCT inv.film_id) AS unique_films_rented,
    (
        SELECT cat.name
        FROM category cat
        JOIN film_category fc ON cat.category_id = fc.category_id
        JOIN inventory inv2 ON fc.film_id = inv2.film_id
        JOIN rental ren2 ON inv2.inventory_id = ren2.inventory_id
        WHERE ren2.customer_id = cus.customer_id
        GROUP BY cat.name
```

```
        ORDER BY COUNT(*) DESC
        LIMIT 1
    ) AS most_common_category
FROM
    rental ren
JOIN
    inventory inv ON ren.inventory_id = inv.inventory_id
JOIN
    customer cus ON ren.customer_id = cus.customer_id
GROUP BY
    cus.customer_id
ORDER BY
    unique_films_rented DESC
LIMIT 1;
```

- 카테고리별 대여한 영화 수가 최대인 카테고리가 여러개인 경우, 이를 모두 출력하는 쿼리

```
SELECT
  C.customer_id,
    C.first_name, C.last_name,
  COUNT(DISTINCT I.film_id) AS unique_films_rented,
    (
    SELECT GROUP_CONCAT(name ORDER BY name)
        FROM (
      SELECT name, COUNT(*) AS category_count
            FROM category C2
            JOIN film_category FC2 ON C2.category_id = FC2.category_id
            JOIN inventory I2 ON FC2.film_id = I2.film_id
            JOIN rental R2 ON I2.inventory_id = R2.inventory_id
            WHERE R2.customer_id = C.customer_id
            GROUP BY name
        ) AS category_counts
        WHERE category_count = (
      SELECT MAX(category_count3)
      FROM (
        SELECT COUNT(*) AS category_count3
        FROM category C3
        JOIN film_category FC3 ON C3.category_id = FC3.category_id
        JOIN inventory I3 ON FC3.film_id = I3.film_id
        JOIN rental R3 ON I3.inventory_id = R3.inventory_id
        WHERE R3.customer_id = C.customer_id
        GROUP BY C3.name
      ) AS category_counts3
```

```
        )
    ) AS most_common_categories
FROM rental R
JOIN customer C ON C.customer_id = R.customer_id
JOIN inventory I ON I.inventory_id = R.inventory_id
GROUP BY C.customer_id
ORDER BY unique_films_rented DESC
LIMIT 1
```

- 가장 많은 수의 종류가 다른 영화를 대여한 고객이 여러 명일 경우 그 고객들 모두를 출력하고, 각 고객이 대여한 영화가 가장 많이 속해있는 카테고리(여러 개인 경우 모두)를 출력하는 쿼리

```
WITH CustomerUniqueFilms AS (
  SELECT
    C.customer_id,
        CONCAT(C.first_name, ' ', C.last_name) AS customer_name,
        COUNT(DISTINCT I.film_id) AS unique_films_rented
    FROM rental R
    JOIN inventory I ON R.inventory_id = I.inventory_id
    JOIN customer C ON C.customer_id = R.customer_id
    GROUP BY C.customer_id
),
MaxUniqueFilms AS (
  SELECT MAX(unique_films_rented) AS max_unique_films
    FROM CustomerUniqueFilms
)
SELECT
  cuf.customer_id, cuf.customer_name, cuf.unique_films_rented,
    (
    SELECT GROUP_CONCAT(name ORDER BY name)
        FROM (
      SELECT name, COUNT(*) AS category_count
            FROM category CAT
            JOIN film_category FC ON CAT.category_id = FC.category_id
            JOIN inventory INV ON FC.film_id = INV.film_id
            JOIN rental REN ON REN.inventory_id = INV.inventory_id
            WHERE REN.customer_id = cuf.customer_id
            GROUP BY CAT.name
        ) AS inner_cat
        WHERE category_count = (
      SELECT MAX(category_count2)
            FROM (
```

```
            SELECT COUNT(*) AS category_count2
            FROM category CAT2
            JOIN film_category FC2 ON CAT2.category_id = FC2.category_id
            JOIN inventory INV2 ON FC2.film_id = INV2.film_id
            JOIN rental REN2 ON REN2.inventory_id = INV2.inventory_id
            WHERE REN2.customer_id = cuf.customer_id
            GROUP BY CAT2.name
                ) AS subquery_cat
            )
    )
FROM CustomerUniqueFilms cuf
JOIN MaxUniqueFilms muf ON cuf.unique_films_rented = muf.max_unique_films
```