# Assignment 1

## Description

You must analyze and improve the database performance (considering only the performance for READ operations) for four queries that are frequently executed in your system. You can only create indexes, so you should write DDL instructions that will be executed on the database by the DBA of your company.

The database management system to be used is a recent version of PostgreSQL with the DVD Rental database provided.

**Query 1.** Total sales for each category of film, but only for customers who have not rented more than 2 PG-13 or NC-17 rated films that feature actors with the same first name as the customer**.**

```
SELECT c.name AS category, SUM(p.amount) AS total_sales
FROM payment AS p INNER JOIN rental AS r ON p.rental_id = r.rental_id
INNER JOIN inventory AS i ON r.inventory_id = i.inventory_id
INNER JOIN film AS f ON i.film_id = f.film_id
INNER JOIN film_category AS fc ON f.film_id = fc.film_id
INNER JOIN category AS c ON fc.category_id = c.category_id
WHERE NOT EXISTS (
SELECT c.first_name, count(*)
FROM customer c, rental r2, inventory i1, film f1, film_actor fa, actor a
WHERE c.customer_id = r2.customer_id
AND r2.inventory_id = i1.inventory_id
AND i1.film_id = f1.film_id and f1.rating in ('PG-13','NC-17')
AND f1.film_id = fa.film_id
AND f1.film_id = f.film_id
AND fa.actor_id = a.actor_id
and a.first_name = c.first_name
GROUP BY c.first_name
```

```
HAVING count(*) >2
)
GROUP BY c.name;
```

**Query 2.** Retrieving rental information for customers between January 1st, 2023, and February 1st, 2023, grouping the data by customer, film, and rental date, and calculating the total rental amount for each group sorted by the total rental amount in descending order.

```
SELECT   tc.first_name AS top_customer_first_name,
        tc.last_name AS top_customer_last_name,
        tf.title AS top_film_title,
        cf.first_name AS customer_first_name,
        cf.last_name AS customer_last_name,
        cf.title AS customer_film_title,
        cf.rental_date AS customer_rental_date,
        cf.amount AS customer_rental_amount
FROM
  (SELECT c.first_name, c.last_name,
         (SELECT COUNT(*)
          FROM rental r
          WHERE c.customer_id = r.customer_id
            AND r.rental_date >= '2023-01-01'
            AND r.rental_date < '2023-02-01') AS rental_count
   FROM customer c
   ORDER BY rental_count DESC LIMIT 100) tc
CROSS JOIN
  (SELECT f.title,
         (SELECT COUNT(*)
          FROM rental r
          INNER JOIN inventory i ON r.inventory_id = i.inventory_id
          WHERE i.film_id = f.film_id
            AND r.rental_date >= '2023-01-01'
            AND r.rental_date < '2023-02-01') AS rental_count
   FROM film f
   ORDER BY rental_count DESC LIMIT 100) tf
INNER JOIN
  (SELECT c.first_name, c.last_name, f.title, r.rental_date, p.amount
   FROM customer c
```

```
  INNER JOIN rental r ON c.customer_id = r.customer_id
  INNER JOIN payment p ON r.rental_id = p.rental_id
  INNER JOIN inventory i ON r.inventory_id = i.inventory_id
  INNER JOIN film f ON i.film_id = f.film_id
  WHERE r.rental_date >= '2023-01-01' AND r.rental_date < '2023-02-01') cf
ON tc.first_name = cf.first_name
  AND tc.last_name = cf.last_name
  AND tf.title = cf.title
ORDER BY tc.rental_count DESC,
      tf.rental_count DESC,
      cf.rental_date ASC;
```

**Query 3.** Selects the staff ID, payment date, and rental ID for all rentals that have a corresponding payment and where there is no other rental by an active customer with a more recent update time

```
SELECT r1.staff_id, p1.payment_date, r1.rental_id
FROM rental r1, payment p1
WHERE r1.rental_id = p1.rental_id AND
NOT EXISTS (SELECT 1
                  FROM rental r2, customer c
                  WHERE r2.customer_id = c.customer_id
                  AND active = 1
                  AND r2.last_update > r1.last_update);
```

**Query 4.** Take movies' ids, titles, release year and rental rate of Horror or Actions category with G or PG rating, which are order by descending rental rate, ascending duration and category id

```
SELECT f.film_id, f.title, f.release_year, f.rental_rate
FROM film AS f, film_category AS fc, category AS c
WHERE (f.rating = 'G' OR f.rating = 'PG') AND f.language_id = 1 AND (c.name =
'Horror' OR c.name = 'Action')
ORDER BY f.rental_rate DESC, f.length ASC, fc.category_id ASC
```

## Deliverables

A single file:

- YOUR_NAME_SURNAME.sql containing all the DDL instructions
- YOUR_NAME_SURNAME.sql should not exceed 1000 indexes +1000 lines

Failing to attend the specification above will result in a reduction of your points by 50%. If you create tables, columns, temporary tables, or views, your score will be ZERO. **You must create only indexes, we will use the latest PostgreSQL Version 15.**

## Evaluation / grading

For each query, the performance will be measured before and after your indexes were created.
For each query, there will be some percentage of improvement: X_1, X_2, X_3, and X_4.
Your score will be calculated as X = X_1 + X_2 + X_3 + X_4.

Your score is based on the percentage of improvement of the cost in all queries.

## Plagiarism

The assignment is individual. In the case of signs of plagiarism (at the discretion of the instructors), all the involved students will obtain score 0 (zero).

## Deadline

The deadline for the assignment is 18/04/2023 13:00. The work after this deadline will not be considered for evaluation.