Exercise 1

```
EXPLAIN ANALYZE SELECT * FROM customer WHERE name = 'Lori White'
```

```
Seq Scan on customer  (cost=0.00..43.50 rows=1 width=210)
 (actual time=0.025..0.150 rows=1 loops=1)
Planning Time: 0.079 ms
Execution Time: 0.164 ms
```

```
EXPLAIN ANALYZE SELECT * FROM customer WHERE address = '0423 Perry Bridge Suite 643
Kingburgh, IA 10443'
```

```
Seq Scan on customer  (cost=0.00..43.50 rows=1 width=210)
 (actual time=0.017..0.112 rows=1 loops=1)
Planning Time: 0.073 ms
Execution Time: 0.145 ms
```

```
CREATE INDEX IDX_name ON customer USING BTREE (name);
CREATE INDEX IDX_address ON customer USING HASH (address);
```

```
Index Scan using idx_name on customer  (cost=0.28..8.29 rows=1 width=210)
 (actual time=0.054..0.055 rows=1 loops=1)
Planning Time: 0.077 ms
Execution Time: 0.067 ms
```

```
Index Scan using idx_address on customer  (cost=0.00..8.02 rows=1 width=210)
 (actual time=0.011..0.016 rows=1 loops=1)
Planning Time: 0.364 ms
Execution Time: 0.030 ms
```

Yes, there is a difference on execution and planning time - The query uses indexes is faster.

Exersice 2

```
WITH required_films_id AS (
  SELECT DISTINCT films_with_categories.film_id FROM category
    INNER JOIN
      (SELECT film_category.film_id, film_category.category_id FROM film_category
        INNER JOIN
          (SELECT film_id, title FROM film where film.rating = 'R' OR film.rating = 'P
G-13') AS films
        ON film_category.film_id = films.film_id) AS films_with_categories
    ON category.category_id = films_with_categories.category_id
  WHERE category.name = 'Horror' OR category.name = 'Sci-fi'),

rented_films_id AS (
  SELECT DISTINCT inventory.film_id FROM inventory
    INNER JOIN
      (SELECT DISTINCT inventory_id FROM rental WHERE rental.return_date is null) AS r
ented_films
    ON inventory.inventory_id = rented_films.inventory_id)

SELECT required_films_id.film_id FROM required_films_id
WHERE required_films_id.film_id NOT IN ( SELECT * FROM rented_films_id);
```

Note: According to my logic, I output those films that no one has rented now.

I display the id of films that are of the desired category and rating and have not been rented by anyone (excluding those that have return_date is null).

Films that are not in the inventory in any store are also displayed (perhaps these films are also taken into account by the company). Also, there are no films in my list that were taken at least in one copy (one cassette is rented, and the other is in stock - such films are considered rented).

Output number: 24

```
WITH max_incomes AS (
  SELECT store_id, city, MAX(income) FROM (SELECT store.store_id, city.city, SUM(payme
nt.amount) as income
                FROM
                payment
                  INNER JOIN staff ON staff.staff_id = payment.staff_id
                  INNER JOIN store ON store.manager_staff_id = staff.staff_id
                  INNER JOIN address ON address.address_id = store.address_id
                  INNER JOIN city ON city.city_id = address.city_id
                -- records for the last month --
                WHERE payment_date > (SELECT MAX(payment_date) FROM payment) - interva
l '1 month'
                GROUP BY store.store_id, city.city) as store_income
  GROUP BY store_id, city)

SELECT * FROM max_incomes
```

	store_id integer	city character varying (50)	max numeric
1	1	Lethbridge	7170.72
2	2	Woodridge	7428.46