Exercise 2

The company is preparing its campaign for next Halloween, so the list of movies that have not been rented yet by the clients is needed, whose rating is R or PG-13 and its category is Horror or Sci-fi

Query Editor | Query History

```
SELECT DISTINCT film.film_id, film.title
FROM film, film_category, rental, inventory, category
WHERE rental.return_date IS NOT null AND rental.inventory_id = inventory.inventory_id AND inventory.film_id=film.film_id
 AND (film.rating = 'R' OR film.rating = 'PG-13') AND  film.film_id=film_category.film_id AND film_category.category_id=category.category_id
 AND (category.name = 'Horror' OR category.name = 'Sci-fi')
```

Data Output | Explain | Messages | Notifications

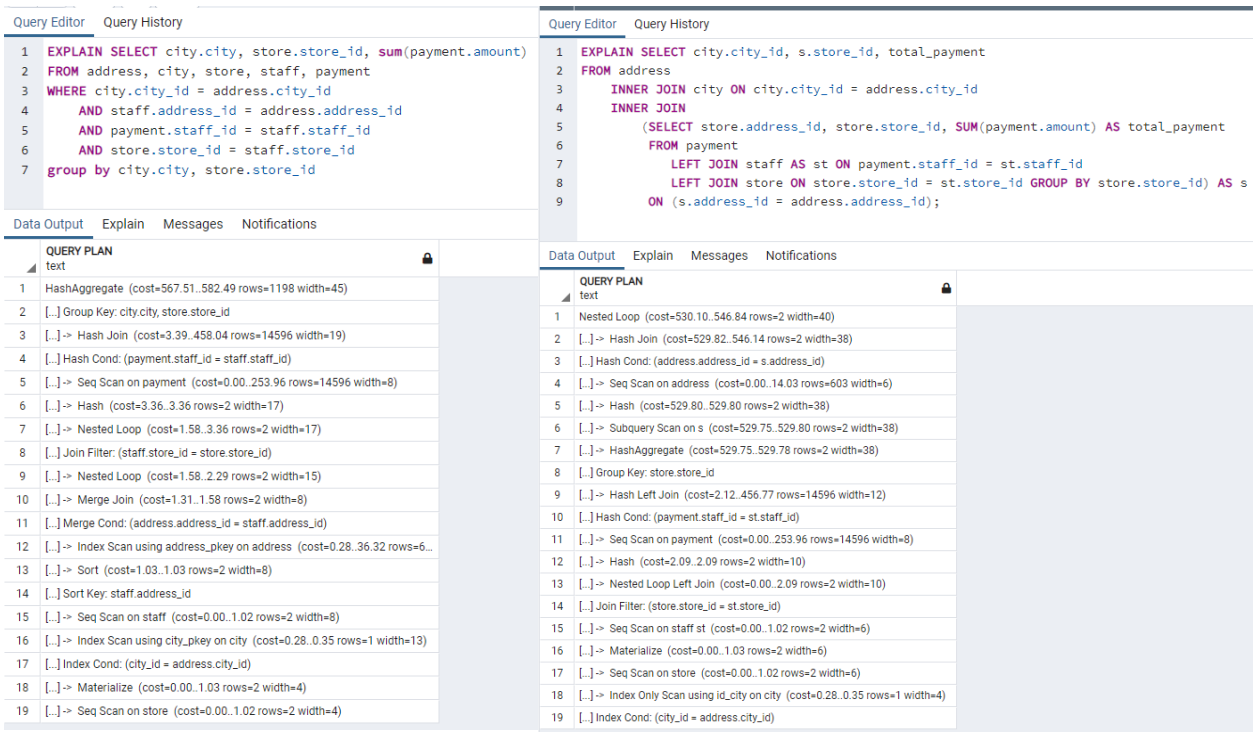
	film_id [PK] integer	title character varying (255)
1	277	Elephant Trojan
2	749	Rules Human
3	800	Sinners Atlantis
4	658	Paris Weekend
5	856	Streetcar Intentions
6	494	Karate Moon
7	830	Spirit Flintstones
8	740	Rollercoaster Bringing
9	716	Reap Unfaithful
10	35	Arachnophobia Rollercoaster
11	995	Yentl Idaho
12	990	World Leathernecks
13	876	Tarzan Videotape
14	30	Anything Savannah
15	922	Undefeated Dalmations
16	9	Alabama Devil
17	222	Desert Poseidon
18	854	Strangers Graffiti
19	8	Airport Pollock
20	535	Love Suicides
21	804	Sleeping Suspects
22	24	Analyze Hoosiers
23	415	High Encino
24	301	Family Sweet
25	870	Swarm Gold
26	904	Train Bunch

The company has decided to reward the best stores in each of the cities, so it is necessary to have a list of the stores that have made a greater number of sales in term of money during the last month recorded.

```
SELECT city.city, store.store_id, sum(payment.amount)
FROM address, city, store, staff, payment
WHERE city.city_id = address.city_id
    AND staff.address_id = address.address_id
    AND payment.staff_id = staff.staff_id
    AND store.store_id = staff.store_id
group by city.city, store.store_id
```

Data Output | Explain | Messages | Notifications

	city character varying (50)	store_id integer	sum numeric
1	Lethbridge	1	30252.12
2	Woodridge	2	31059.92



Query Editor | Query History

```
EXPLAIN SELECT city.city, store.store_id, sum(payment.amount)
FROM address, city, store, staff, payment
WHERE city.city_id = address.city_id
    AND staff.address_id = address.address_id
    AND payment.staff_id = staff.staff_id
    AND store.store_id = staff.store_id
group by city.city, store.store_id
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	HashAggregate (cost=567.51..582.49 rows=1198 width=45)
2	[...] Group Key: city.city, store.store_id
3	[...] -> Hash Join (cost=3.39..458.04 rows=14596 width=19)
4	[...] Hash Cond: (payment.staff_id = staff.staff_id)
5	[...] -> Seq Scan on payment (cost=0.00..253.96 rows=14596 width=8)
6	[...] -> Hash (cost=3.36..3.36 rows=2 width=17)
7	[...] -> Nested Loop (cost=1.58..3.36 rows=2 width=17)
8	[...] Join Filter: (staff.store_id = store.store_id)
9	[...] -> Nested Loop (cost=1.58..2.29 rows=2 width=15)
10	[...] -> Merge Join (cost=1.31..1.58 rows=2 width=8)
11	[...] Merge Cond: (address.address_id = staff.address_id)
12	[...] -> Index Scan using address_pkey on address (cost=0.28..36.32 rows=6...
13	[...] -> Sort (cost=1.03..1.03 rows=2 width=8)
14	[...] Sort Key: staff.address_id
15	[...] -> Seq Scan on staff (cost=0.00..1.02 rows=2 width=8)
16	[...] -> Index Scan using city_pkey on city (cost=0.28..0.35 rows=1 width=13)
17	[...] Index Cond: (city_id = address.city_id)
18	[...] -> Materialize (cost=0.00..1.03 rows=2 width=4)
19	[...] -> Seq Scan on store (cost=0.00..1.02 rows=2 width=4)

Query Editor | Query History

```
EXPLAIN SELECT city.city_id, s.store_id, total_payment
FROM address
    INNER JOIN city ON city.city_id = address.city_id
    INNER JOIN
        (SELECT store.address_id, store.store_id, SUM(payment.amount) AS total_payment
         FROM payment
            LEFT JOIN staff AS st ON payment.staff_id = st.staff_id
            LEFT JOIN store ON store.store_id = st.store_id GROUP BY store.store_id) AS s
        ON (s.address_id = address.address_id);
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Nested Loop (cost=530.10..546.84 rows=2 width=40)
2	[...] -> Hash Join (cost=529.82..546.14 rows=2 width=38)
3	[...] Hash Cond: (address.address_id = s.address_id)
4	[...] -> Seq Scan on address (cost=0.00..14.03 rows=603 width=6)
5	[...] -> Hash (cost=529.80..529.80 rows=2 width=38)
6	[...] -> Subquery Scan on s (cost=529.75..529.80 rows=2 width=38)
7	[...] -> HashAggregate (cost=529.75..529.78 rows=2 width=38)
8	[...] Group Key: store.store_id
9	[...] -> Hash Left Join (cost=2.12..456.77 rows=14596 width=12)
10	[...] Hash Cond: (payment.staff_id = st.staff_id)
11	[...] -> Seq Scan on payment (cost=0.00..253.96 rows=14596 width=8)
12	[...] -> Hash (cost=2.09..2.09 rows=2 width=10)
13	[...] -> Nested Loop Left Join (cost=0.00..2.09 rows=2 width=10)
14	[...] Join Filter: (store.store_id = st.store_id)
15	[...] -> Seq Scan on staff st (cost=0.00..1.02 rows=2 width=6)
16	[...] -> Materialize (cost=0.00..1.03 rows=2 width=6)
17	[...] -> Seq Scan on store (cost=0.00..1.02 rows=2 width=6)
18	[...] -> Index Only Scan using id_city on city (cost=0.28..0.35 rows=1 width=4)
19	[...] Index Cond: (city_id = address.city_id)

I did an analysis using the EXPLAIN command (picture on the left) and found out that the cost is not optimal (582.59) and the step of **comparing the id of different tables takes the most time**.

Thus, I propose a more optimal solution (picture on the right) cost (546.14) which is about 36 less than the first option, using the **JOIN command for all tables.**