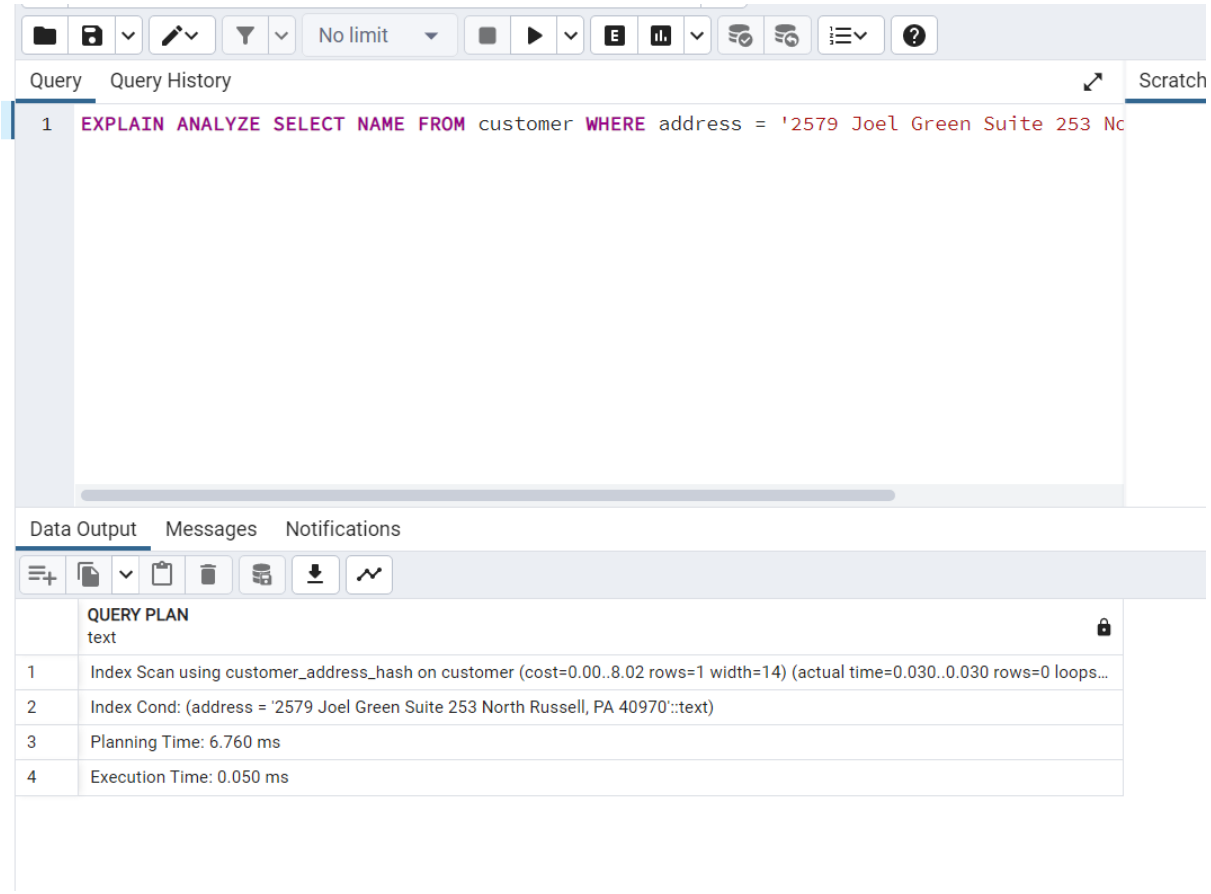
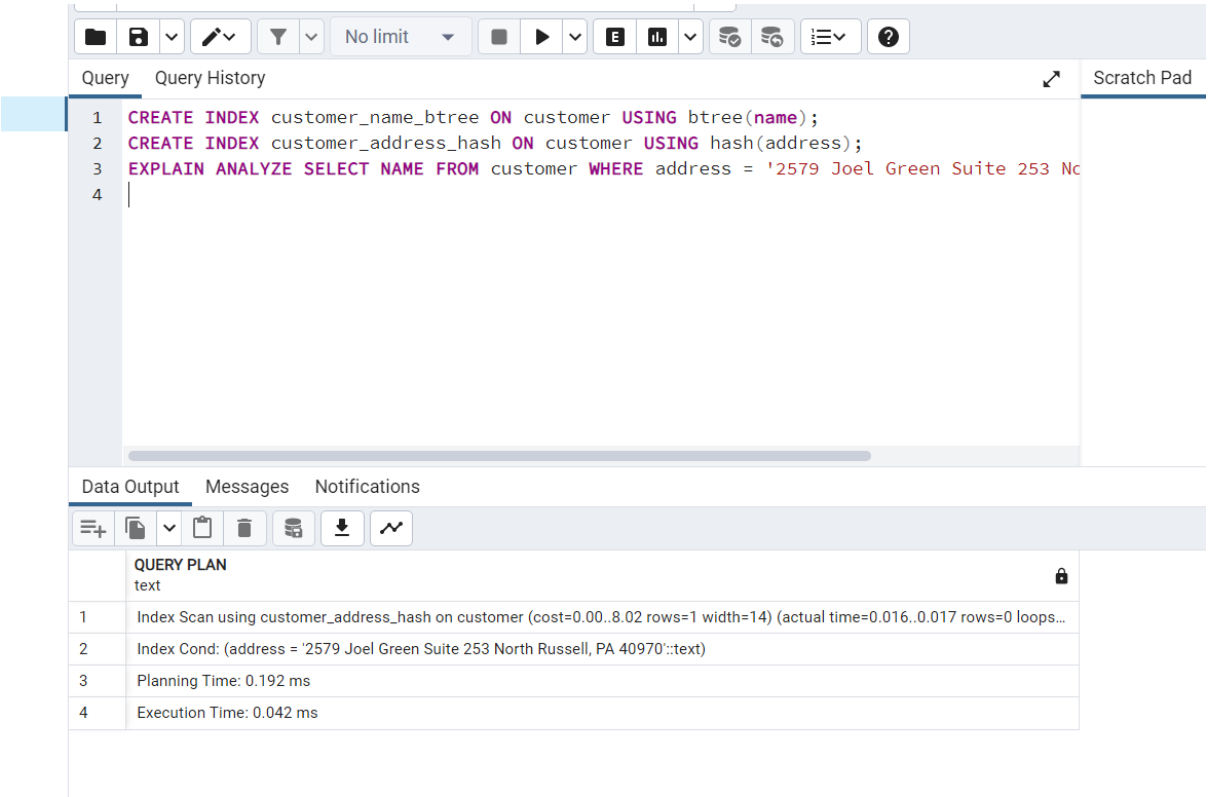
Task1:
Query:

EXPLAIN ANALYZE SELECT NAME FROM customer WHERE address = '2579 Joel Green Suite 253 North Russell, PA 40970';



Query2:
CREATE INDEX customer_name_btree ON customer USING btree(name);
CREATE INDEX customer_address_hash ON customer USING hash(address);
EXPLAIN ANALYZE SELECT NAME FROM customer WHERE address = '2579 Joel Green Suite 253 North Russell, PA 40970';
-- Index Scan using customer_address_hash on customer (cost=0.00..8.02 rows=1 width=14) (actual time=0.329..0.330 rows=0 loops=1)

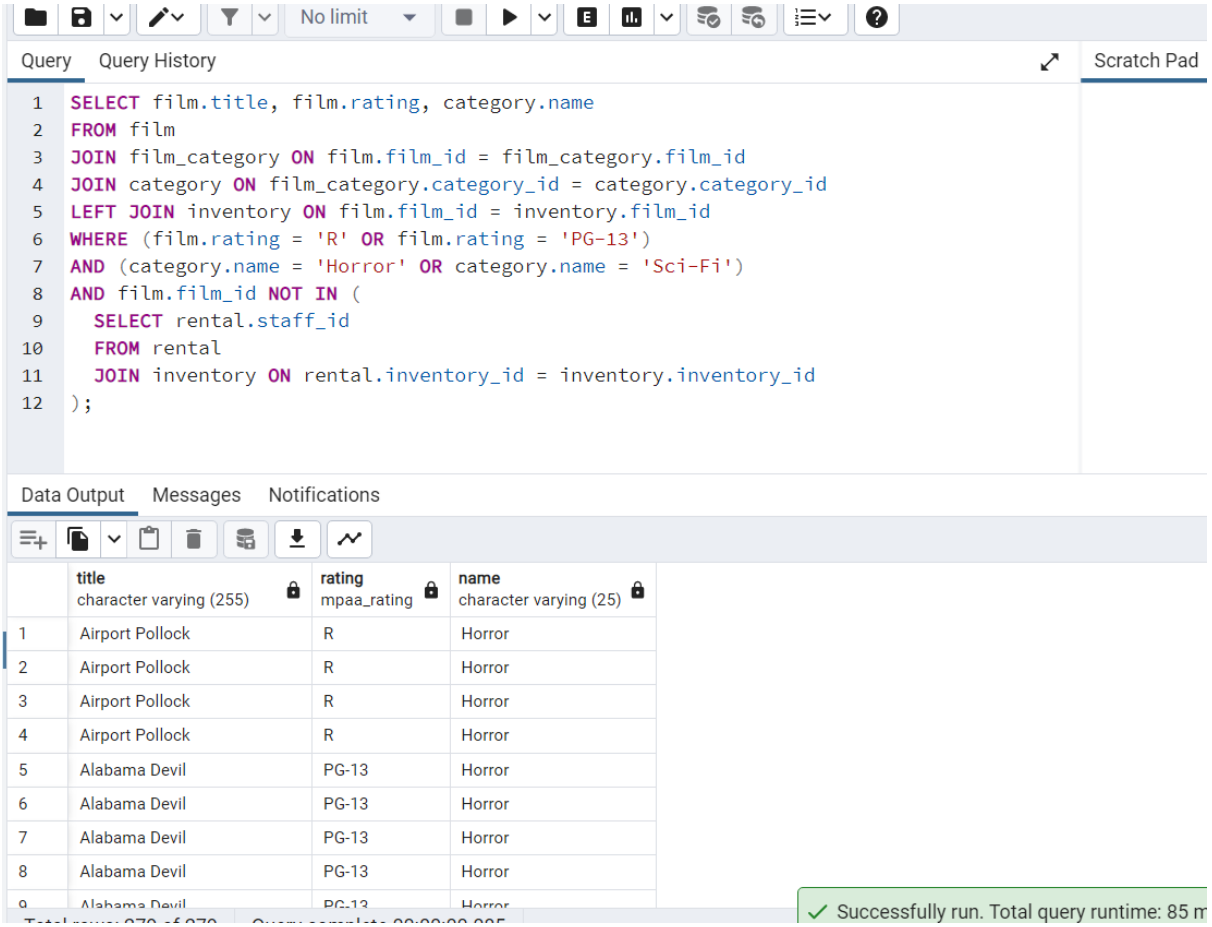


adding indexes improved the performance of queries that use the indexed columns. In contrast, queries that do not use the indexed columns may not have a noticeable improvement in performance or may even be slower due to the additional overhead of maintaining the indexes.

Task2:

Query 1:

```
SELECT film.title, film.rating, category.name
FROM film
JOIN film_category ON film.film_id = film_category.film_id
JOIN category ON film_category.category_id = category.category_id
LEFT JOIN inventory ON film.film_id = inventory.film_id
WHERE (film.rating = 'R' OR film.rating = 'PG-13')
AND (category.name = 'Horror' OR category.name = 'Sci-Fi')
AND film.film_id NOT IN (
  SELECT rental.staff_id
  FROM rental
  JOIN inventory ON rental.inventory_id = inventory.inventory_id
);
```



	title character varying (255)	rating mpaa_rating	name character varying (25)
1	Airport Pollock	R	Horror
2	Airport Pollock	R	Horror
3	Airport Pollock	R	Horror
4	Airport Pollock	R	Horror
5	Alabama Devil	PG-13	Horror
6	Alabama Devil	PG-13	Horror
7	Alabama Devil	PG-13	Horror
8	Alabama Devil	PG-13	Horror
9	Alabama Devil	PG-13	Horror

Query | Query History

```
EXPLAIN SELECT film.title, film.rating, category.name
FROM film
JOIN film_category ON film.film_id = film_category.film_id
JOIN category ON film_category.category_id = category.category_id
LEFT JOIN inventory ON film.film_id = inventory.film_id
WHERE (film.rating = 'R' OR film.rating = 'PG-13')
AND (category.name = 'Horror' OR category.name = 'Sci-Fi')
AND film.film_id NOT IN (
  SELECT rental.staff_id
  FROM rental
  JOIN inventory ON rental.inventory_id = inventory.inventory_id
);
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Hash Join (cost=624.76..711.94 rows=105 width=87)
2	Hash Cond: (film.film_id = film_category.film_id)
3	-> Hash Right Join (cost=602.62..685.51 rows=857 width=23)
4	Hash Cond: (inventory.film_id = film.film_id)
5	-> Seq Scan on inventory (cost=0.00..70.81 rows=4581 width=2)
6	-> Hash (cost=600.28..600.28 rows=187 width=23)
7	-> Seq Scan on film (cost=528.78..600.28 rows=187 width=23)
8	Filter: ((NOT (hashed SubPlan 1)) AND ((rating = 'R'::mpaa_rating) OR (rating = 'PG-13'::mpaa_rating)))
9	SubPlan 1

Total rows: 21 of 21 | Query complete 00:00:00.047

The most expensive step is the Hash Right Join between the inventory and film tables. The Seq Scan on the inventory table is also relatively expensive.

To improve performance, we can consider creating indexes on the columns used in the join conditions of these tables. For example, we can create an index on the film_id column in the inventory table and another index on the category_id column in the film_category table.