Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное
учреждение высшего образования
“Национальный исследовательский университет ИТМО”

Факультет инфокоммуникационных технологий

ЛАБОРАТОРНАЯ РАБОТА №2

Запросы на выборку и модификацию данных, представления и индексы в PostgreSQL по дисциплине: «Проектирование и реализация баз данных»

Выполнил студент:
Тюмин Никита Сергеевич
Группа №K32402
Преподаватель:
Говорова Марина Михайловна

Санкт-Петербург
2023

Цель работы:

Овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.

Программное обеспечение:

СУБД PostgreSQL 14, pgAdmin 4.

Практическое задание:

1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию, часть 2 и 3).
2. Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.
3. Изучить графическое представление запросов и просмотреть историю запросов.
4. Создать простой и составной индексы для двух произвольных запросов и сравнить время выполнения запросов без индексов и с индексами. Для получения плана запроса использовать команду EXPLAIN.

Ход работы:

Работа проводилась в созданной ранее базе данных, ER диаграмма представлена на Рисунке 1.

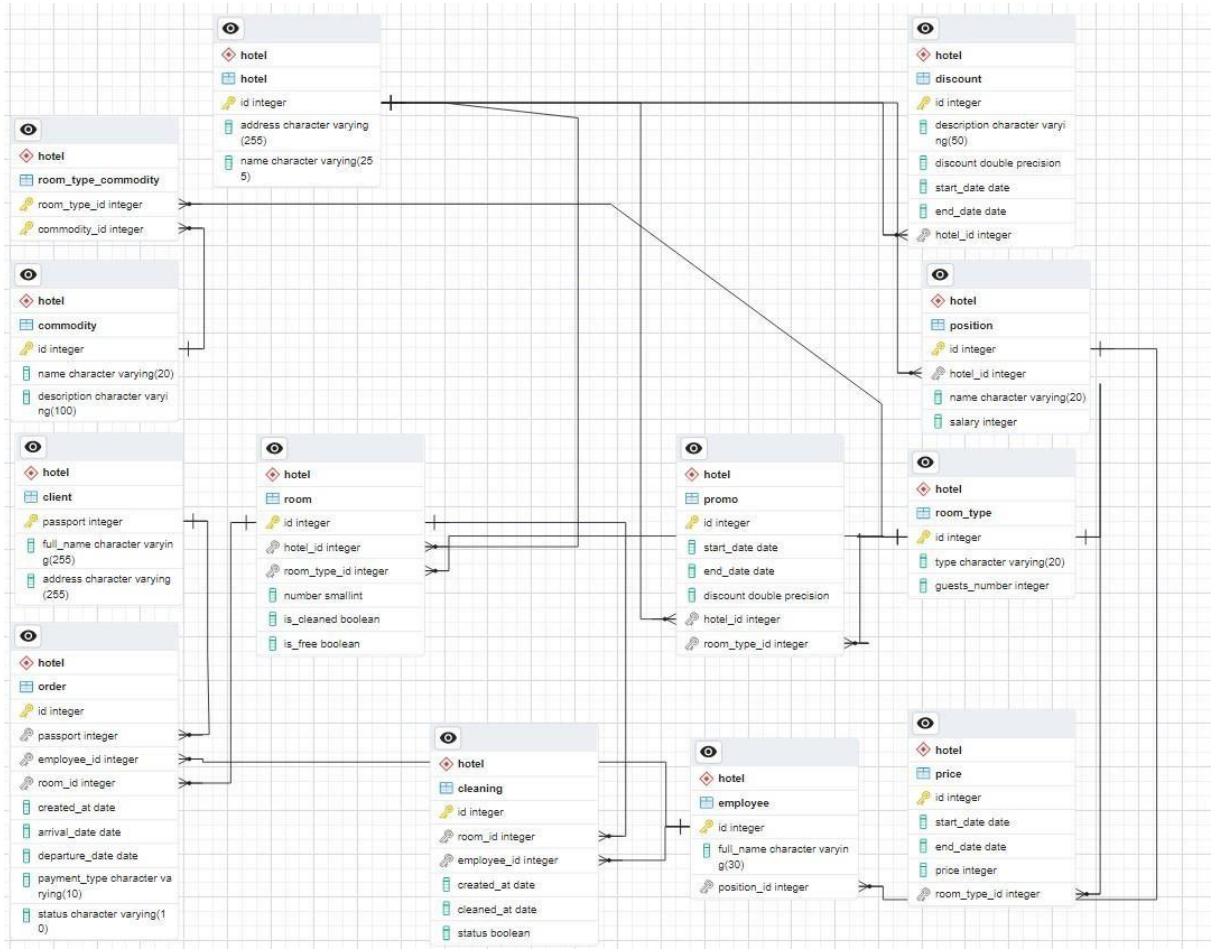


Рисунок 1 – ER диаграмма базы данных

Запросы к базе данных:

1. Составить список всех 2-местных номеров отелей, с ценой менее 200 т.р., упорядочив данные в порядке уменьшения стоимости.

```
with ids as (
select rt.id
	from hotel.room_type rt
	join hotel.price p
	on rt.id  =  p.room_type_id
	where p.end_date > CURRENT_DATE
	and p.price < 200000
	and rt.guests_number = 2
	order by p.price desc
)

select *
from hotel.room
where room_type_id in (select * from ids)
order by array_position(array(select * from ids), room_type_id)
```

Query | Query History

```
with ids as (
    select rt.id
    from hotel.room_type rt
    join hotel.price p
    on rt.id = p.room_type_id
    where p.end_date > CURRENT_DATE
    and p.price < 200000
    and rt.guests_number = 2
    order by p.price desc
)

select *
from hotel.room
where room_type_id in (select * from ids)
order by array_position(array(select * from ids), room_type_id)
```

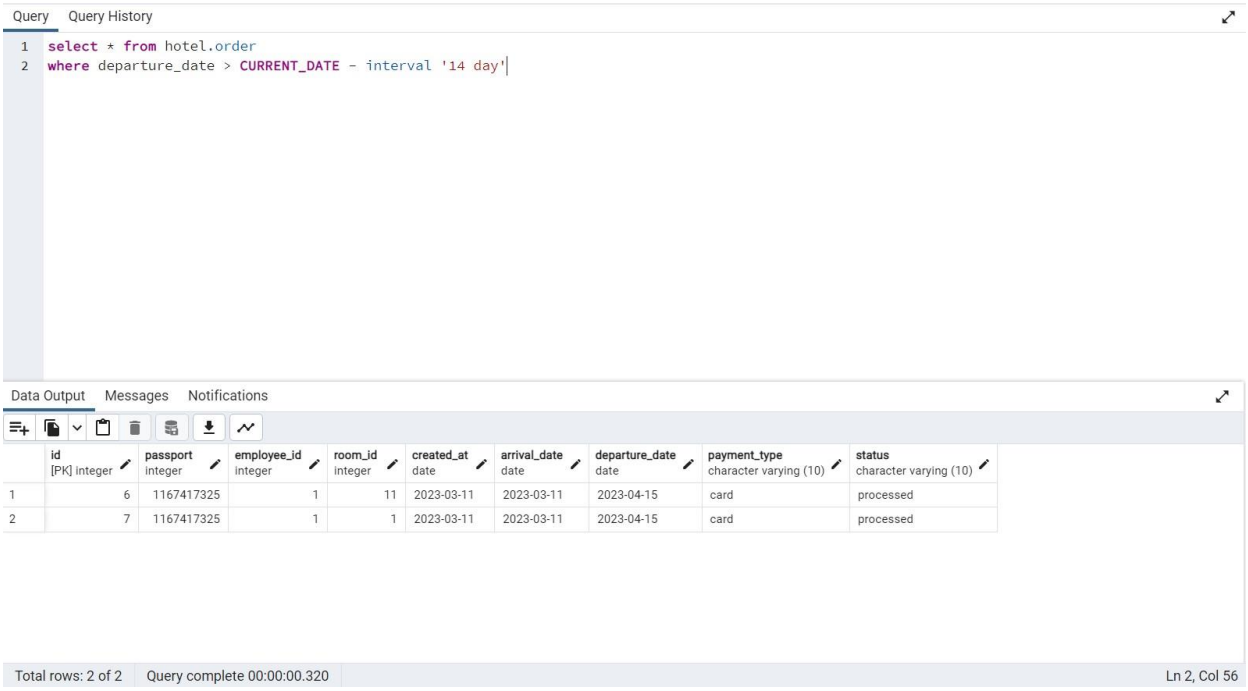
Data Output | Messages | Notifications

	id [PK] integer	hotel_id integer	room_type_id integer	number smallint	is_cleaned boolean	is_free boolean
1	181	6	4	410	true	true
2	141	5	4	510	true	true
3	172	6	4	401	true	true
4	173	6	4	402	true	true
5	174	6	4	403	true	true
6	175	6	4	404	true	true
7	176	6	4	405	true	true
8	177	6	4	406	true	true
9	178	6	4	407	true	true

Total rows: 40 of 40 | Query complete 00:00:00.059 | Ln 9, Col 18

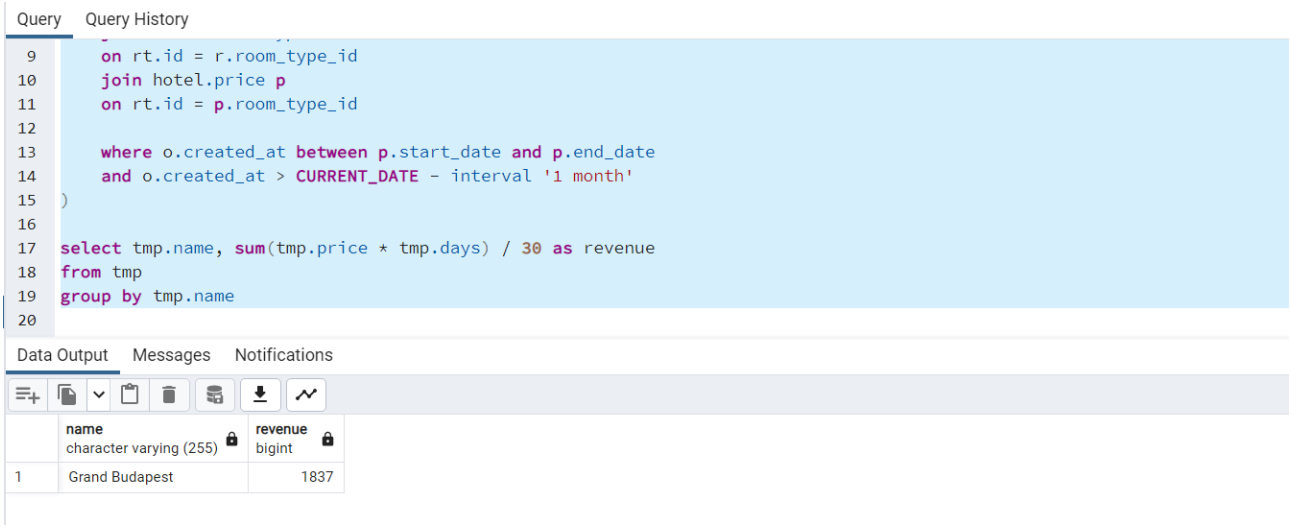
2. Выбрать все записи регистрации постояльцев, которые выехали из отелей в течение двух последних недель.

```
select * from hotel.order
where departure_date > CURRENT_DATE - interval '14 day'
```



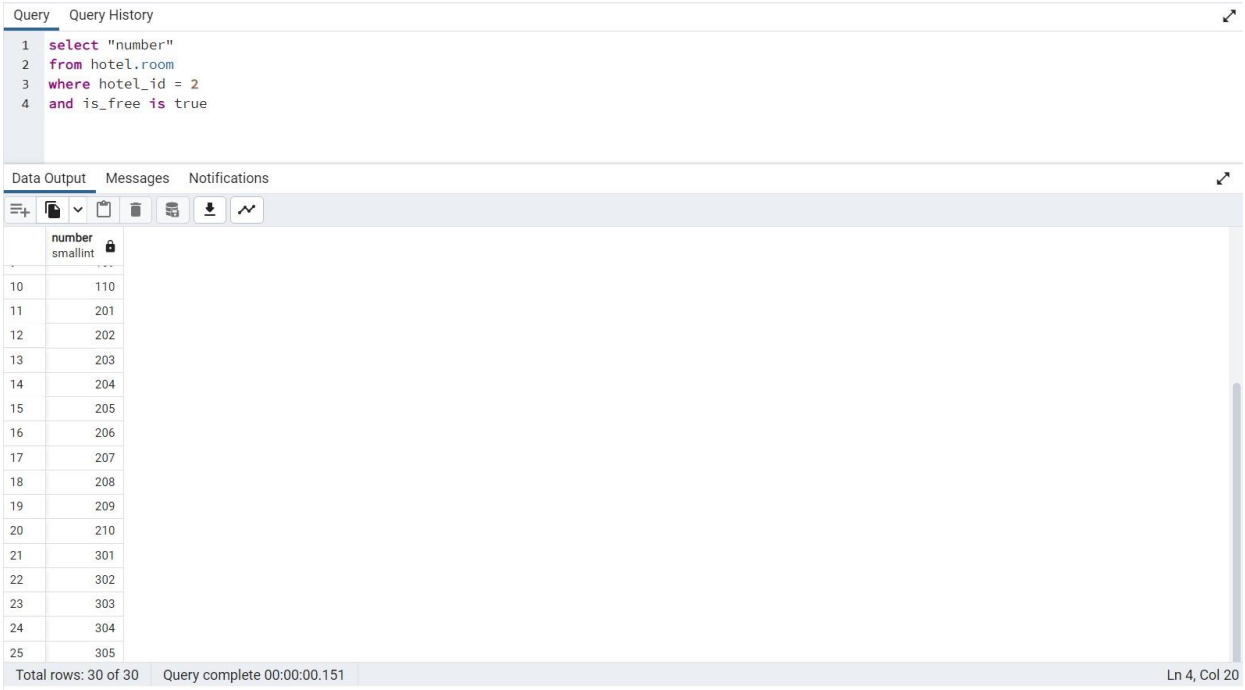
3. Чему равен общий суточный доход каждого отеля за последний месяц?

```
with tmp as(
      select o.created_at, h.name, p.price, o.departure_date - o.arrival_date as days
      from hotel.order o
      join hotel.room r
      on o.room_id = r.id
      join hotel.hotel h
      on h.id = r.hotel_id
      join hotel.room_type rt
      on rt.id = r.room_type_id
      join hotel.price p
      on rt.id = p.room_type_id

      where o.created_at between p.start_date and p.end_date
      and o.created_at > CURRENT_DATE - interval '1 month'
)

select tmp.name, sum(tmp.price * tmp.days) / 30 as revenue
from tmp
group by tmp.name
```



4. Составить список свободных номеров одного из отелей на текущий день.

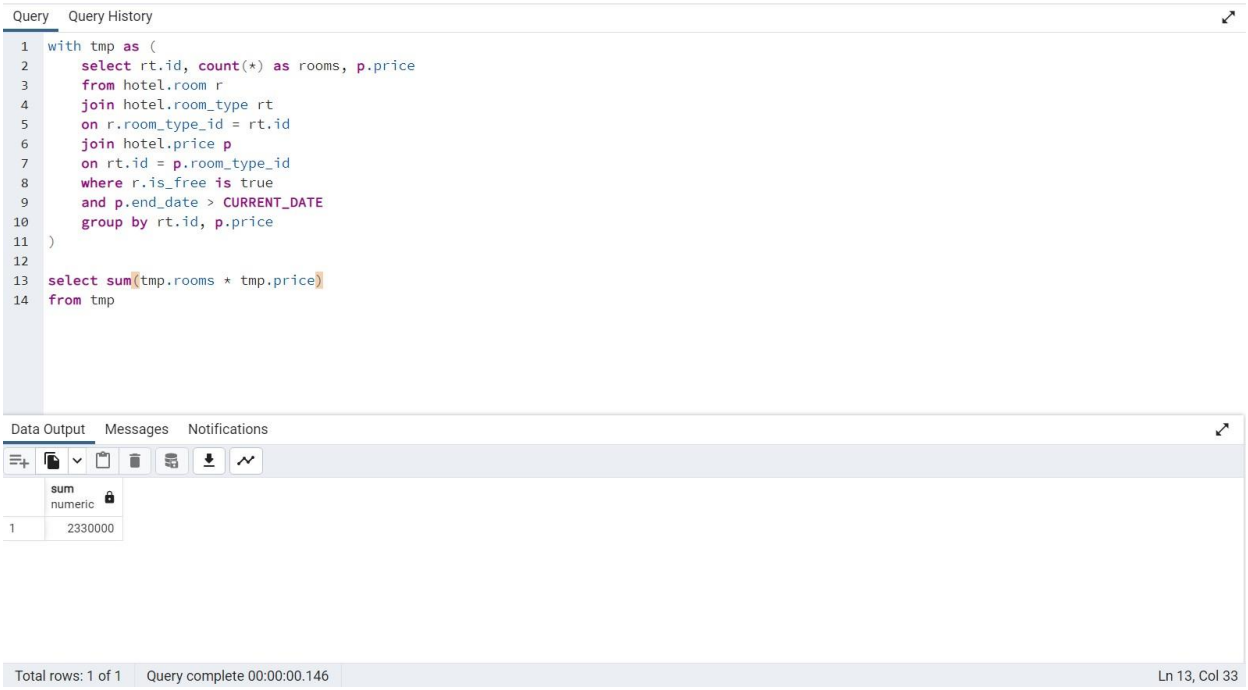
```
select "number"
from hotel.room
where hotel_id = 2
and is_free is true
```



5. Найти общие потери от незанятых номеров за текущий день по всей сети.

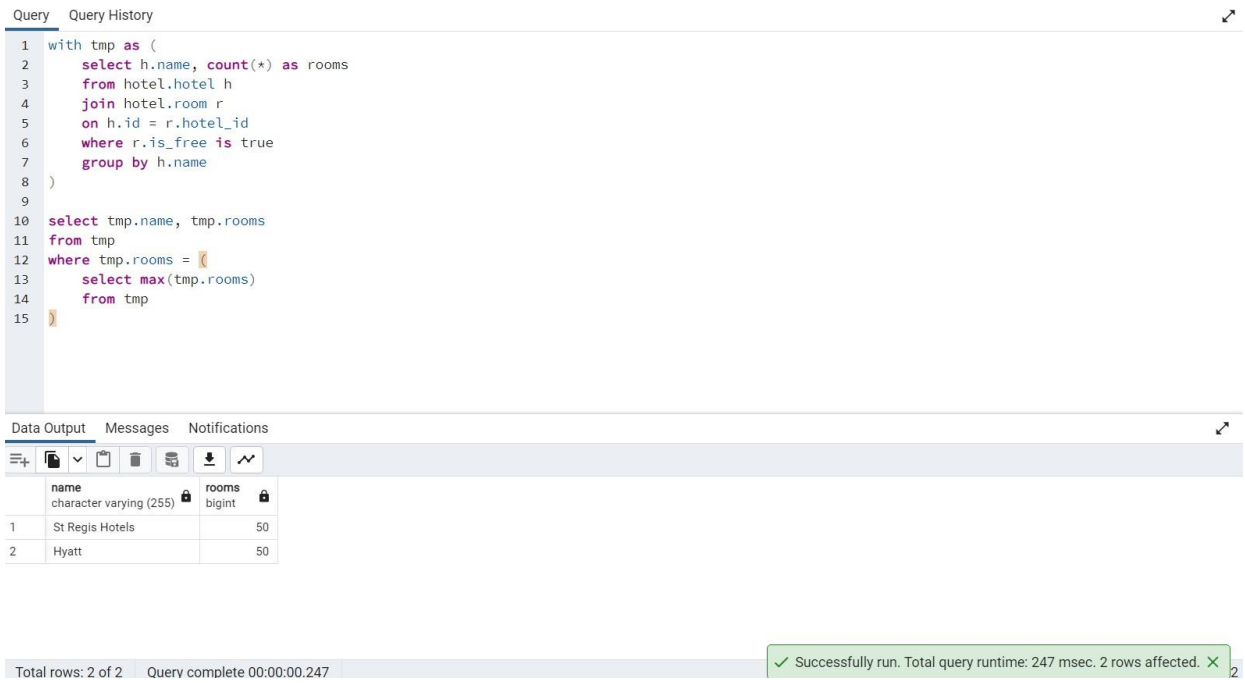
```
with tmp as (
	select rt.id, count(*) as rooms, p.price
	from hotel.room r
	join hotel.room_type rt
	on r.room_type_id = rt.id
	join hotel.price p
	on rt.id = p.room_type_id
	where r.is_free is true
	and p.end_date > CURRENT_DATE
```

```
	group by rt.id, p.price
)

select sum(tmp.rooms * tmp.price)
from tmp
```



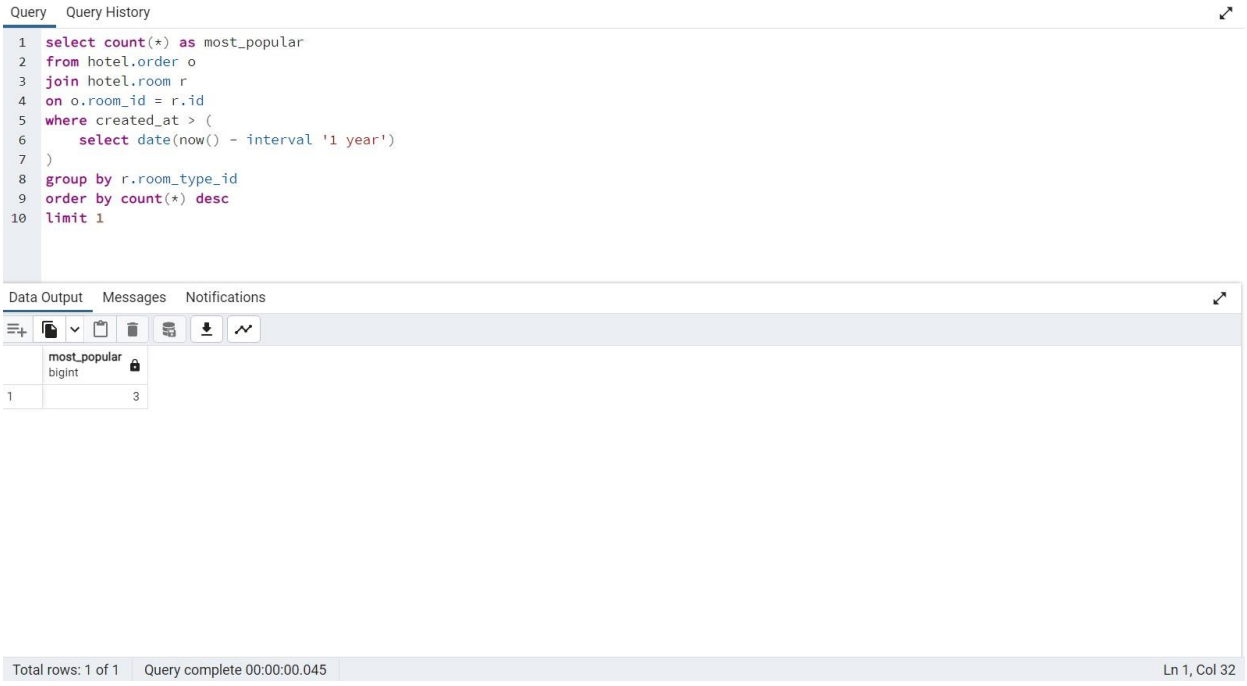
6. Определить, в каком отеле имеется наибольшее количество незанятых номеров на текущие сутки.

```
with tmp as (
	select h.name, count(*) as rooms
	from hotel.hotel h
	join  hotel.room  r
	on h.id = r.hotel_id
	where r.is_free is true
	group by h.name
)

select tmp.name, tmp.rooms
from tmp
where tmp.rooms = (
	select max(tmp.rooms)
	from tmp
)
```



7. Определить самый популярный тип номеров за последний год.

```
select count(*) as most_popular
from hotel.order o
join  hotel.room  r
on o.room_id = r.id
where created_at > (
      select date(now() - interval '1 year')
)
group by r.room_type_id
order by count(*) desc
limit 1
```

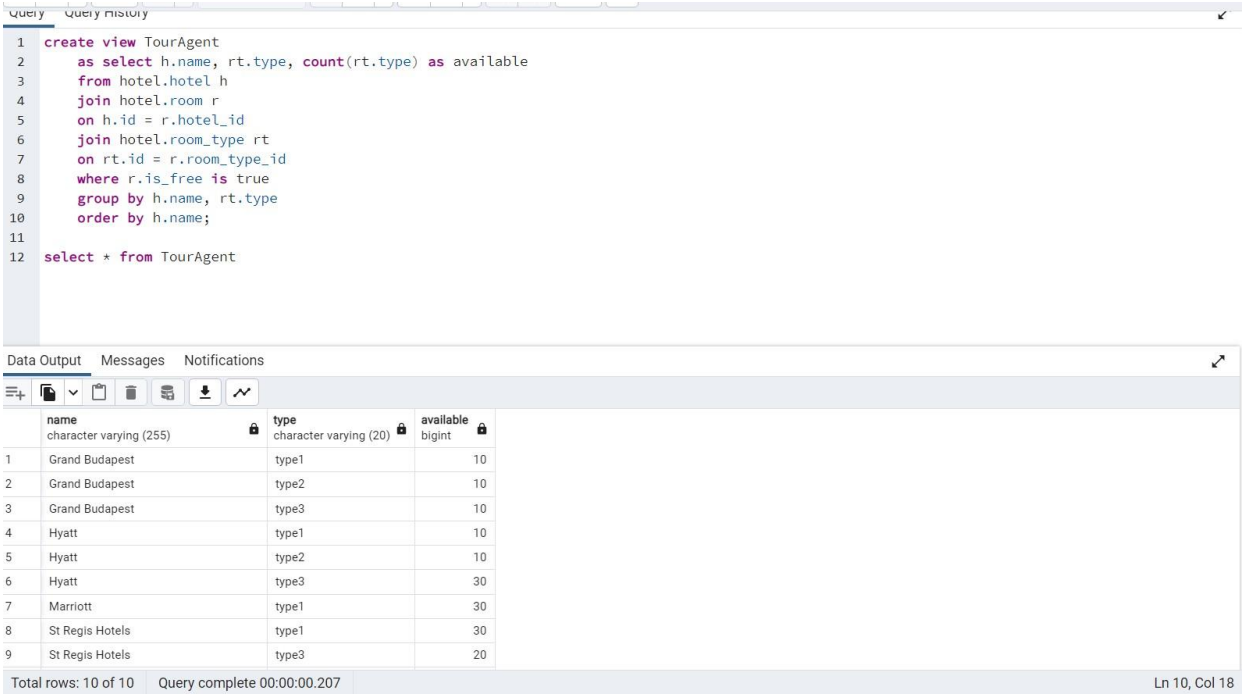


Представления:

1. Для турагентов (поиск свободных номеров в отелях).

```
create view TourAgent
as select h.name, rt.type, count(rt.type) as available
from hotel.hotel h
```

```
join  hotel.room  r
on h.id = r.hotel_id
join hotel.room_type rt
on rt.id = r.room_type_id
where r.is_free is true
group by h.name, rt.type
order by h.name;

select * from TourAgent
```



```
create view TourAgent
    as select h.name, rt.type, count(rt.type) as available
    from hotel.hotel h
    join hotel.room r
    on h.id = r.hotel_id
    join hotel.room_type rt
    on rt.id = r.room_type_id
    where r.is_free is true
    group by h.name, rt.type
    order by h.name;

select * from TourAgent
```

Data Output | Messages | Notifications

	name character varying (255)	type character varying (20)	available bigint
1	Grand Budapest	type1	10
2	Grand Budapest	type2	10
3	Grand Budapest	type3	10
4	Hyatt	type1	10
5	Hyatt	type2	10
6	Hyatt	type3	30
7	Marriott	type1	30
8	St Regis Hotels	type1	30
9	St Regis Hotels	type3	20

Total rows: 10 of 10 | Query complete 00:00:00.207 | Ln 10, Col 18

2. Для владельца компании (информация о доходах каждого отеля в сети за прошедший месяц).

```
create or replace view HotelRevenue
as with tmp as(
      select o.created_at, h.name, p.price, o.departure_date - o.arrival_date as days

      from hotel.order o
      join hotel.room r
      on o.room_id = r.id
      join hotel.hotel h
      on h.id = r.hotel_id
      join hotel.room_type rt
      on rt.id = r.room_type_id
      join hotel.price p
      on rt.id = p.room_type_id

      where o.created_at between p.start_date and p.end_date
      and o.created_at > CURRENT_DATE - interval '1 month'
      )

      select tmp.name, sum(tmp.price * tmp.days)
      from tmp
      group by tmp.name;

select * from HotelRevenue;
```

Query Query History

```
1  create or replace view HotelRevenue
2  as with tmp as(
3      select o.created_at, h.name, p.price, o.departure_date - o.arrival_date as days
4
5      from hotel.order o
6      join hotel.room r
7      on o.room_id = r.id
8      join hotel.hotel h
9      on h.id = r.hotel_id
10     join hotel.room_type rt
11     on rt.id = r.room_type_id
12     join hotel.price p
13     on rt.id = p.room_type_id
14
```

Data Output Messages Notifications

	name character varying (255)	sum bigint
1	Grand Budapest	617400

Запросы на модификацию данных:

1. Продление даты действия цены на месяц, если до конца действия осталось менее одной недели

```
update hotel.price
set end_date = end_date + interval '1 month'
where end_date < CURRENT_DATE + interval '1 week'
and end_date > CURRENT_DATE;

select * from hotel.price;
```

До выполнения запроса:

Query Query History

```
1  -- продление даты действия цены на месяц, если до конца действия осталось менее одной недели
2  select * from hotel.price;
3
```

Data Output Messages Notifications

	id [PK] integer	start_date date	end_date date	price integer	room_type_id integer
1	2	2023-01-01	2025-01-01	10000	2
2	3	2023-01-01	2030-01-01	15000	3
3	4	2023-01-01	2030-01-01	20000	4
4	1	2023-01-01	2025-01-01	6000	1
5	6	2020-01-01	2022-12-31	5000	1
6	5	2023-01-01	2023-04-20	25000	5

Total rows: 6 of 6 Query complete 00:00:00.226 Rows selected: 1 Ln 2, Col 27

После:

Query Query History

```sql
-- продление даты действия цены на месяц, если до конца действия осталось менее одной недели
update hotel.price
set end_date = end_date + interval '1 month'
where end_date < CURRENT_DATE + interval '1 week'
and end_date > CURRENT_DATE;

select * from hotel.price;
```

Data Output Messages Notifications

	id [PK] integer	start_date date	end_date date	price integer	room_type_id integer
1	2	2023-01-01	2025-01-01	10000	2
2	3	2023-01-01	2030-01-01	15000	3
3	4	2023-01-01	2030-01-01	20000	4
4	1	2023-01-01	2025-01-01	6000	1
5	6	2020-01-01	2022-12-31	5000	1
6	5	2023-01-01	2023-05-20	25000	5

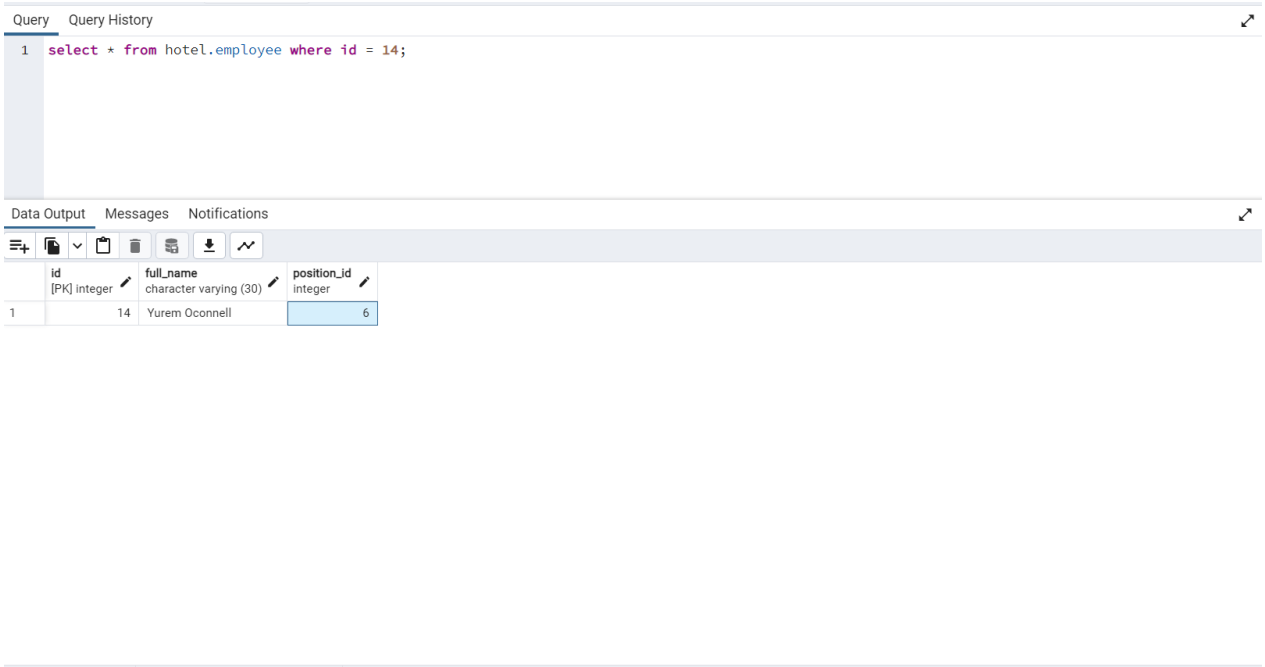
Total rows: 6 of 6 Query complete 00:00:00.245 Rows selected: 1 Ln 5, Col 29

2. Повышение уборщика, сделавшего больше всех уборок

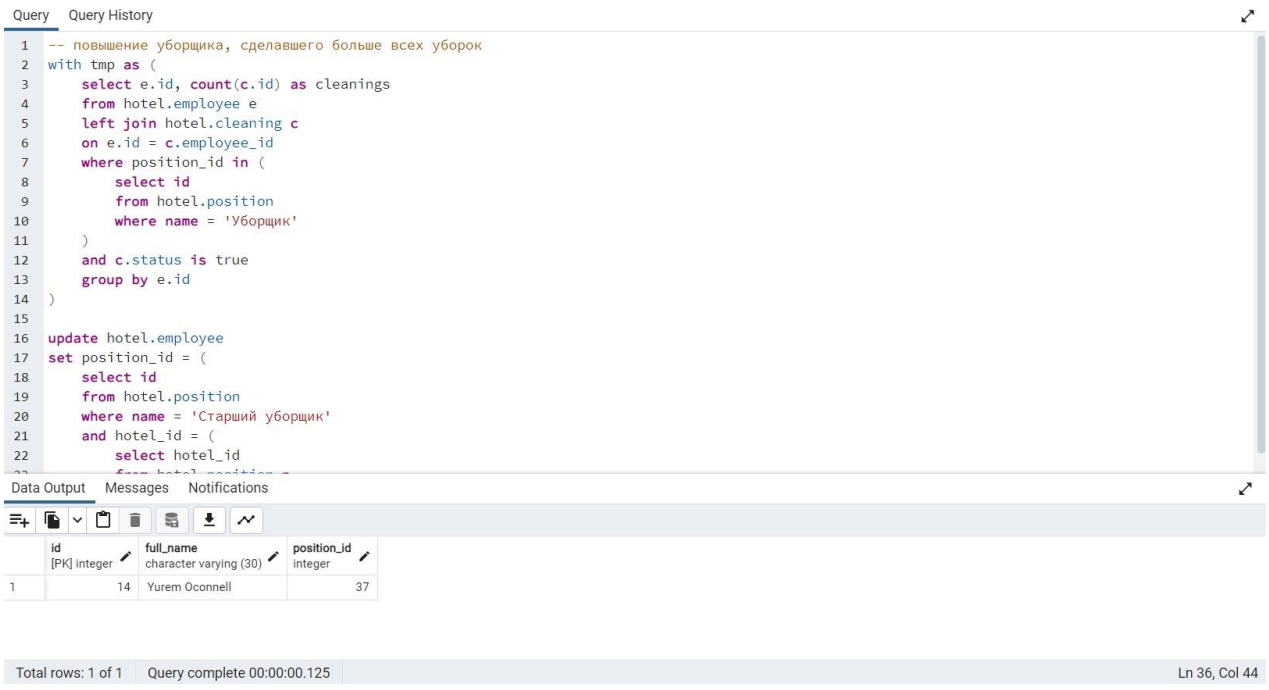
```sql
with tmp as (
select e.id, count(c.id) as cleanings
from hotel.employee e
left join hotel.cleaning c
on e.id = c.employee_id
where position_id in (
      select id
      from hotel.position
      where name = 'Уборщик'
)
and c.status is true
group by e.id
)

update hotel.employee
set position_id = (
select id
from hotel.position
where name = 'Старший уборщик'
and hotel_id = (
      select hotel_id
      from hotel.position p
      where p.id = position_id
)
)
where id in (
select tmp.id
from tmp
where tmp.cleanings = (
      select max(tmp.cleanings)
      from tmp
)
);
```

```
select * from hotel.employee where id = 14;
```

До выполнения запроса:



После:



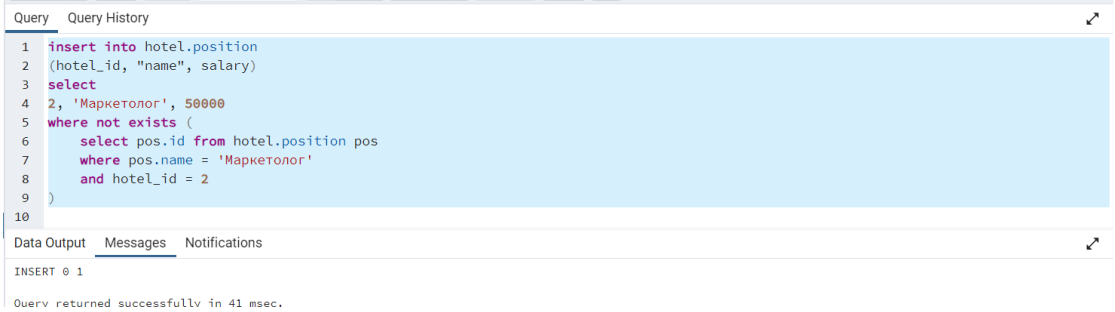
3. Удаление сотрудника, оформлявшего заказы в последний раз более года назад

До запроса:

После запроса:


```
with tmp as (
      select e.id
      from hotel.employee e
      join hotel.position p
      on p.id = e.position_id

      where e.id in (
            select o.employee_id
            from hotel.order o
            where o.created_at < CURRENT_DATE - interval '1 year'
      )
)

delete from hotel.employee e
where e.id in (select * from tmp)
```

4. Добавить новую должность «Маркетолог» в отель, если его еще нет




```
insert into hotel.position
(hotel_id, "name", salary)
select
2, 'Маркетолог', 50000
where not exists (
      select pos.id from hotel.position pos
      where pos.name = 'Маркетолог'
      and hotel_id = 2
)
```

Создание индексов:

Для более наглядного влияния индекса на запрос в таблицу client были добавлены данные:

Query Query History

```
1  select count(*) from hotel.client
```

Data Output Messages Notifications

	count bigint
1	75000

Total rows: 1 of 1 Query complete 00:00:00.046

Successfully run. Total query runtime: 46 msec. 1 rows affecte

Вначале был создан простой индекс на таблице client. Запрос до создания индекса:

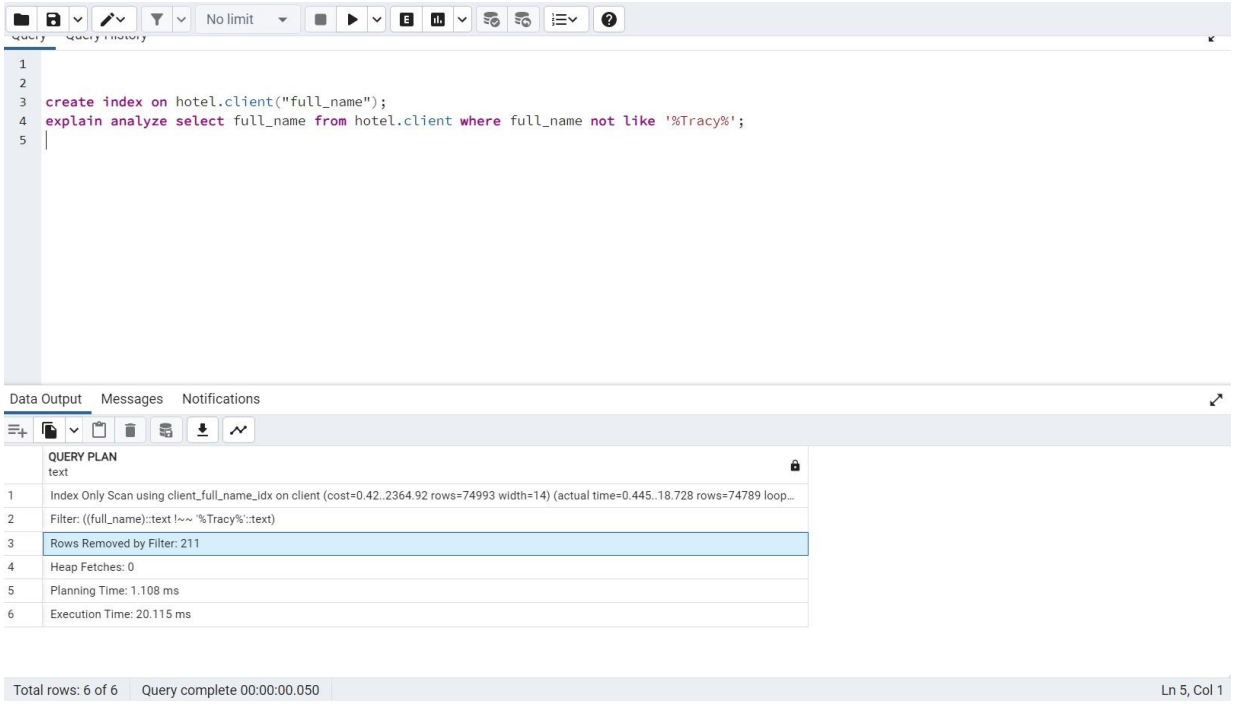
```
1
2
3
4  explain analyze select full_name from hotel.client where full_name not like '%Tracy%';
5
```

Data Output Messages Notifications

	QUERY PLAN text
1	Seq Scan on client (cost=0.00..3228.50 rows=74993 width=14) (actual time=0.007..9.503 rows=74789 loops...
2	Filter: ((full_name)::text !~~ '%Tracy%'::text)
3	Rows Removed by Filter: 211
4	Planning Time: 0.066 ms
5	Execution Time: 10.933 ms

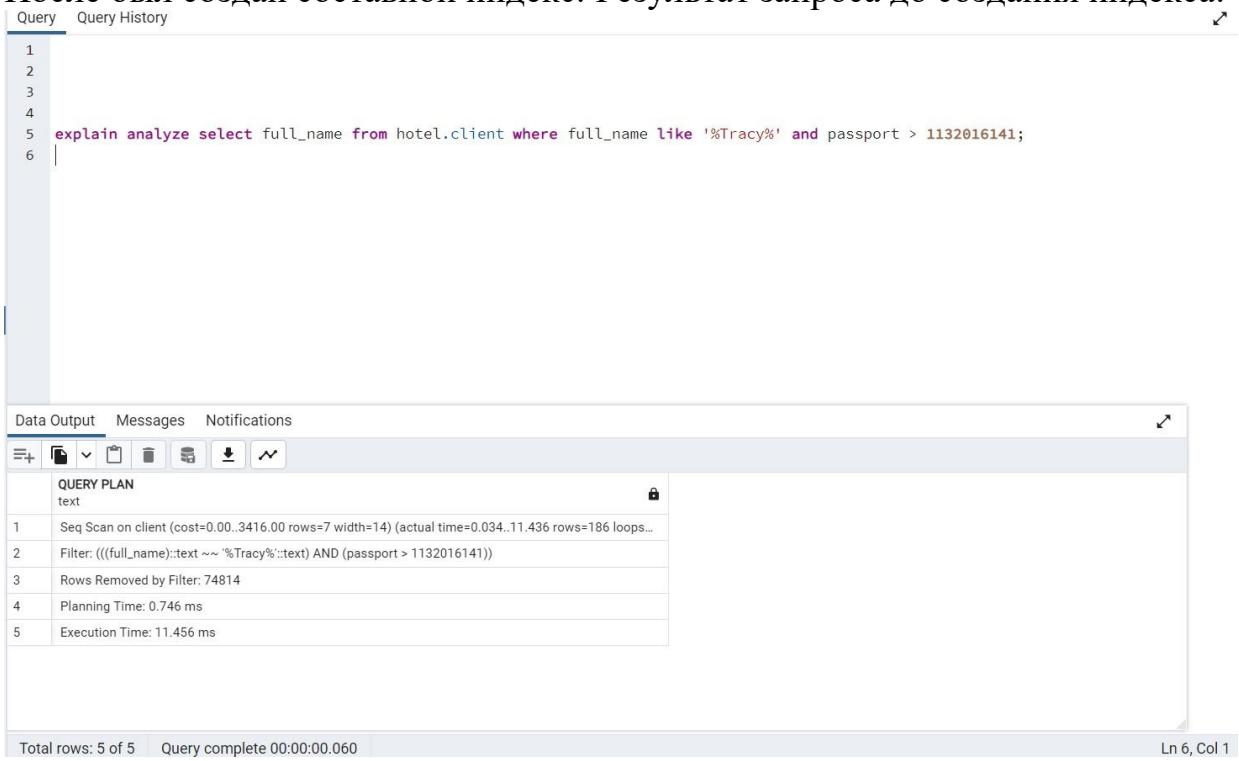
Total rows: 5 of 5 Query complete 00:00:00.115 Ln 4, Col 1

После создания:

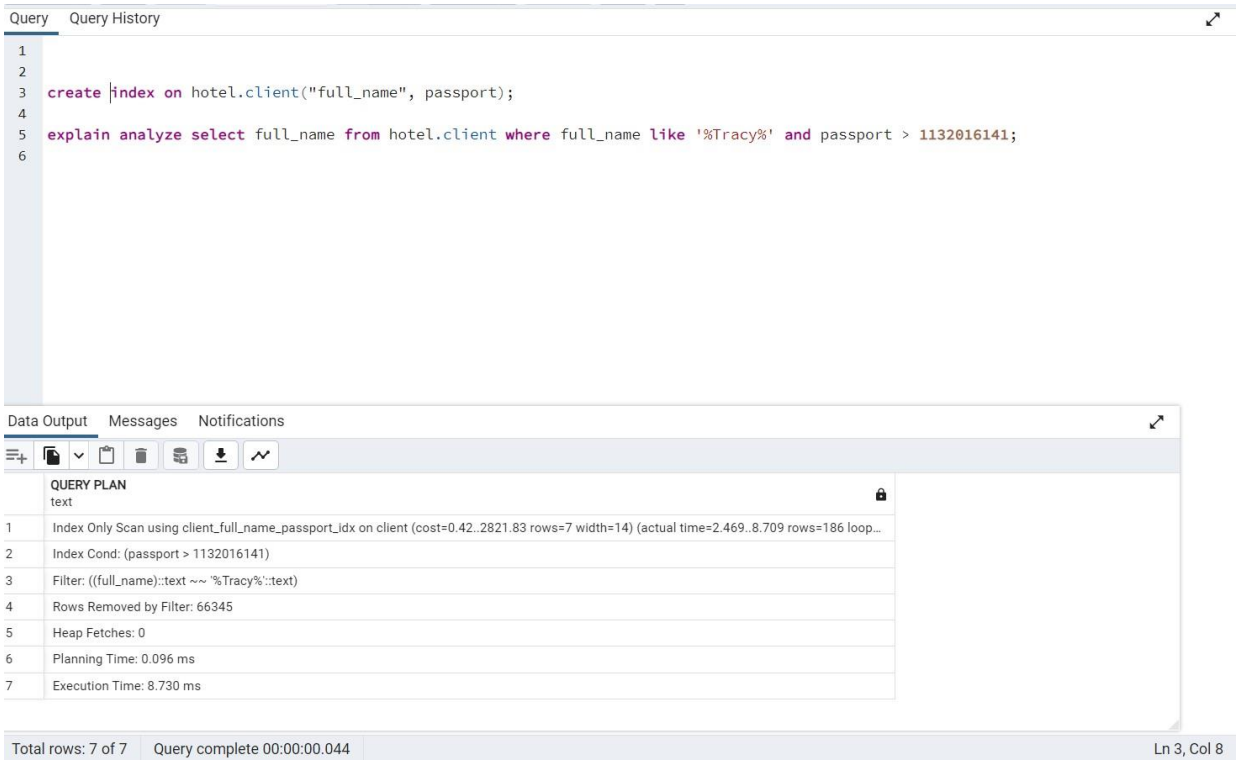


Как видно из плана запроса, в первом случае оптимизатор прошелся по всем строкам таблицы (Sequence scan), во втором случае был использован только что созданный индекс (Index Only Scan), однако производительность ухудшилась: 10мс без индекса против 20мс с индексом

После был создан составной индекс. Результат запроса до создания индекса:



И после:



Ситуация аналогична предыдущей, однако теперь мы выграли пару миллисекунд в производительности: 11.4мс против 7.7мс.

Выводы:

1. Были созданы запросы на выборку и модификацию данных с помощью подзапросов
2. Были созданы представления.
3. Были созданы индексы и проанализирована производительность запросов до и после создания индексов.