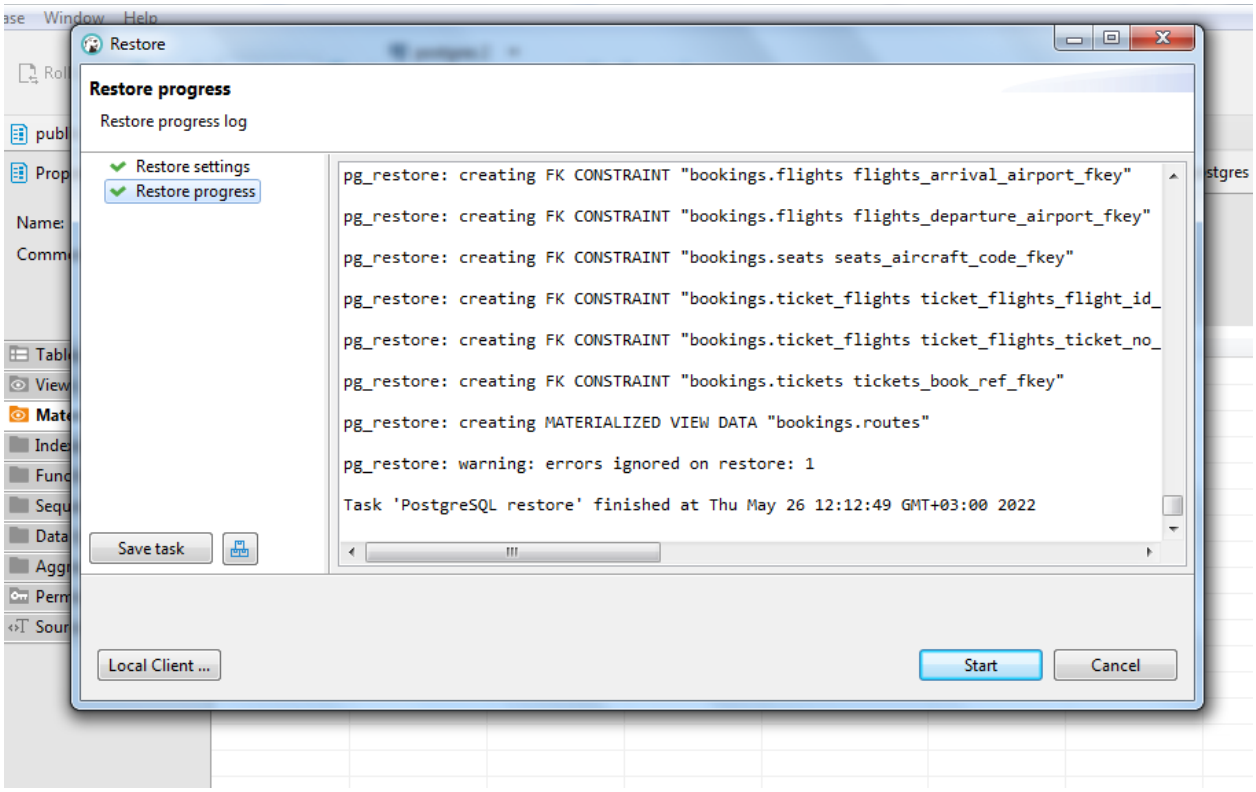
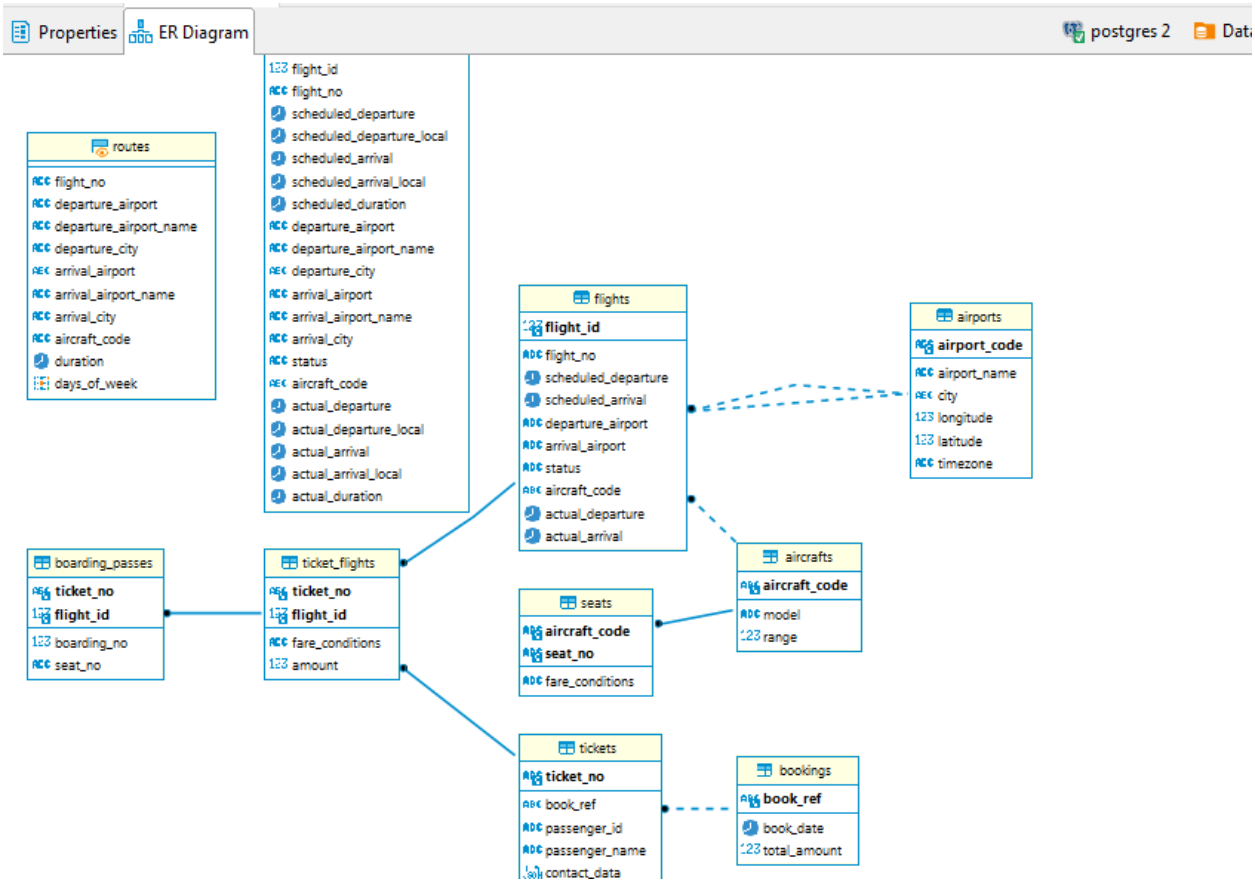
1. В работе использовался локальный тип подключения.
    - если база была развернута из *.sql или *.backup файла, необходимо приложить скриншот успешного импорта или восстановления



Restore Complete

2. Скриншот ER-диаграммы из DBeaver`а согласно Вашего подключения.



3. Краткое описание БД - из каких таблиц и представлений состоит.

База данных состоит из восьми таблиц и двух представлений. В качестве предметной области выбраны авиаперевозки по России.

Список отношений

Имя	Тип	Small	Medium	Big	Описание
aircrafts	таблица	16 kB	16 kB	16 kB	Самолеты
airports	таблица	48 kB	48 kB	48 kB	Аэропорты
boarding_passes	таблица	31 MB	102 MB	427 MB	Посадочные талоны
bookings	таблица	13 MB	30 MB	105 MB	Бронирования
flights	таблица	3 MB	6 MB	19 MB	Рейсы
flights_v	представление	0 kb	0 kB	0 kB	Рейсы
routes	мат. предст.	136 kB	136 kB	136 kB	Маршруты
seats	таблица	88 kB	88 kB	88 kB	Места
ticket_flights	таблица	64 MB	145 MB	516 MB	Перелеты
tickets	таблица	47 MB	107 MB	381 MB	Билеты

4. Развернутый анализ БД - описание таблиц, логики, связей и бизнес области (частично можно взять из описания базы данных, оформленной в виде анализа базы данных).

- **Таблица bookings.aircrafts**
  Каждая модель воздушного судна идентифицируется своим трехзначным кодом (aircraft_code). Название модели (model). Максимальная дальность полета в километрах (range)

- **Таблица bookings.airports**
  Трехбуквенный код аэропорта (airport_code), имя (airport_name). Город, в котором аэропорт расположен (city) . Широта (longitude), долгота (latitude) и часовой пояс (timezone).
- **Таблица bookings.boarding_passes**
  При регистрации на рейс, которая возможна за сутки до плановой даты отправления, пассажиру выдается посадочный талон. Он идентифицируется также, как и перелет — номером билета и номером рейса. Посадочным талонам присваиваются последовательные номера (boarding_no) в порядке регистрации пассажиров на рейс (этот номер будет уникальным только в пределах данного рейса). В посадочном талоне указывается номер места (seat_no).
- **Таблица bookings.bookings**
  Пассажир заранее (book_date, максимум за месяц до рейса) бронирует билет себе и, возможно, нескольким другим пассажирам. Бронирование идентифицируется номером (book_ref, шестизначная комбинация букв и цифр). Поле total_amount хранит общую стоимость включенных в бронирование перелетов всех пассажиров.
- **Таблица bookings.flights**
  Естественный ключ таблицы рейсов состоит из двух полей — номера рейса (flight_no) и даты отправления (scheduled_departure). Чтобы сделать внешние ключи на эту таблицу компактнее, в качестве первичного используется суррогатный ключ (flight_id). Рейс всегда соединяет две точки — аэропорты вылета (departure_airport) и прибытия (arrival_airport). Такое понятие, как «рейс с пересадками» отсутствует: если из одного аэропорта до другого нет прямого рейса, в билет просто включаются несколько необходимых рейсов. У каждого рейса есть запланированные дата и время вылета (scheduled_departure) и прибытия (scheduled_arrival). Реальные время вылета (actual_departure) и прибытия (actual_arrival) могут отличаться: обычно не сильно, но иногда и на несколько часов, если рейс задержан. Статус рейса (status) может принимать одно из следующих значений: • Scheduled Рейс доступен для бронирования. Это происходит за месяц до плановой даты вылета; до этого запись о рейсе не существует в базе данных. • On Time Рейс доступен для регистрации (за сутки до плановой даты вылета) и не задержан. • Delayed Рейс доступен для регистрации (за сутки до плановой даты вылета), но задержан. • Departed Самолет уже вылетел и находится в воздухе. Arrived Самолет прибыл в пункт назначения. • Cancelled Рейс отменен
- **Таблица bookings.seats**
  Места определяют схему салона каждой модели. Каждое место определяется своим номером (seat_no) и имеет закрепленный за ним класс обслуживания (fare_conditions) — Economy, Comfort или Business.
- **Таблица bookings.ticket_flights**
  Перелет соединяет билет с рейсом и идентифицируется их номерами. Для каждого перелета указываются его стоимость (amount) и класс обслуживания (fare_conditions).
- **Таблица bookings.tickets**
  Билет имеет уникальный номер (ticket_no), состоящий из 13 цифр. Билет содержит идентификатор пассажира (passenger_id) — номер документа, удостоверяющего личность, — его фамилию и имя (passenger_name) и контактную информацию (contact_date). Ни идентификатор пассажира, ни имя не являются постоянными (можно поменять паспорт, можно сменить фамилию), поэтому однозначно найти все билеты одного и того же пассажира невозможно
- **Представление "bookings.flights_v"**

Содержит дополнительную информацию: • расшифровку данных об аэропорте вылета (departure_airport, departure_airport_name, departure_city), • расшифровку данных об аэропорте прибытия (arrival_airport, arrival_airport_name, arrival_city), • местное время вылета (scheduled_departure_local, actual_departure_local), • местное время прибытия (scheduled_arrival_local, actual_arrival_local), • продолжительность полета (scheduled_duration, actual_duration).

- **Материализованное представление bookings.routes**
  Таблица рейсов содержит избыточность: из нее можно было бы выделить информацию о маршруте (номер рейса, аэропорты отправления и назначения), которая не зависит от конкретных дат рейсов. Именно такая информация и составляет материализованное представление routes.

Бизнес задачи, которые можно решить, используя БД.

- Анализ наиболее популярных направлений для путешествий, в зависимости от даты(возможно для настройки таргетированной рекламы или регулирования цен на билеты)

- Получать данные о задержанных рейсах. Выявлять и устранять проблемы.

- Получать данные о загруженности аэропортов для оптимизации пассажиропотоков.

- Имея информацию о моделях самолетов, совершающих перелеты по определенным направлениям, можно сделать выводы о наличии взлетно-посадочных полос, под определенные типы воздушных судов, а также о наличии наземного оборудования для их обслуживания. Так же это позволит в случае возникновения внештатной ситуации, объединить два рейса по одному направлению, к примеру, на малых самолетах Boeing 737 в один рейс с использованием Boeing 777.

5. Список SQL запросов из приложения №2 с описанием логики их выполнения.

```sql
--Task 1. В каких городах больше одного аэропорта?
select a.city, count(a.city)     --Выводим город, и количество аэропортов, используя COUNT,
from airports a                  --из таблицы с аэропортами
group by a.city                  -- группируя по городам
having count(a.city)>1           -- при условии, что количество аэропортов в городе больше одного.
```

```sql
--Task 2. В каких аэропортах есть рейсы, выполняемые самолетом с максимальной дальностью перелета? //использовать подзапрос
select a2.airport_code, a2.airport_name  -- получаем список аэропортов вместе с кодом аэропорта
from (select  aircraft_code, "range"     -- в подзапросе получаем код самолета и его дальность
     from aircrafts a
     order by "range" desc               -- отсортированные по дальности от наибоольшей к наименьшей
     limit 1) a_range                    -- ограничиваем по первой строке(чтобы получить максимальную дальность)
join flights f on f.aircraft_code = a_range.aircraft_code   --присоединяем таблицу с рейсами
join airports a2 on a2.airport_code = f.departure_airport or (a2.airport_code = f.arrival_airport)  ----присоединяем таблицу с аэропортами
group by a2.airport_code -- группируем по трехзначному коду аэропорта
```

```sql
--Task 3. Вывести 10 рейсов с максимальным временем задержки вылета // Использовать оператор LIMIT
select (actual_departure - scheduled_departure) delay  -- получаем разницу между планируемым временем вылета(scheduled) и реальным(actu
from flights f
where  actual_departure is not null                    -- для вылетевших рейсов
order by delay desc                                    -- сортиовка по разнице времени от наибольшей к наимеьшей
limit 10                                               -- получаем первые 10 строк, в которых имеем наибольшую задержку вылета
```

```sql
--Task 4. Были ли брони, по которым не были получены посадочные талоны? // Использовать верный тип JOIN

select b.book_ref booking_num   -- выводим список броней, по которым не получены посадочные талоны
from bookings b
join tickets t on t.book_ref  = b.book_ref             -- соединяем таблицы с билетами и бронированиями
join ticket_flights tf on tf.ticket_no = t.ticket_no   -- присоединяем таблицу tickets_flights (имеем все номера билетов, которые есть в принципе)
left join boarding_passes bp on bp.ticket_no = tf.ticket_no --используем LEFT JOIN, чтобы присоединить данные из таблицы с посадочными талонами
where bp.ticket_no  is null                            --соответственно, там, где   получится NULL - посадочный не получен, эти данные и выводим
group by  b.book_ref
```

```sql
--Task 5. Найдите количество свободных мест для каждого рейса, их % отношение к общему количеству мест в самолете.
--Добавьте столбец с накопительным итогом - суммарное накопление количества вывезенных пассажиров из каждого аэропорта на каждый день.
--Т.е. в этом столбце должна отражаться накопительная сумма - сколько человек уже вылетело из данного аэропорта на этом или более ранних рейсах
--в течении дня.  -- // Использовать Оконную функцию, Подзапросы или/и CTE


--создаем cte, В котором получаем возможное количество мест для каждой модели самолета
--далее в запросе получаем количество занятых мест в самолете на каждом вылетевшем рейсе (count(bp.seat_no))
-- получаем процентное соотношение по соответствующей формуле
-- для получается суммы с нарастающим итогом, используем оконную функцию с опрацией SUM
-- так как нас интересует результат по аэропортам в течении дня, в prtition by используем аэропорт вылета и дату вылета)
with cte_seats as(
select s.aircraft_code, count(s.seat_no) seats_c
from seats s
    group by s.aircraft_code)
select f.flight_id, f.departure_airport,  cte.seats_c total_cnt_of_seats,  count(bp.seat_no) cnt_of_occuated_seats,
       round((cte.seats_c - count(bp.seat_no))/cte.seats_c:: dec,2)*100 "free_seats, %", f.actual_departure,
       sum(count(bp.seat_no)) over (partition by (f.departure_airport, f.actual_departure::date) order by f.actual_departure) count_of_passengers
from flights f
        join boarding_passes bp on bp.flight_id = f.flight_id
        join cte_seats cte on cte.aircraft_code = f.aircraft_code
        where f.actual_departure is not null
        group by f.flight_id, cte.seats_c, cte.aircraft_code
        order by f.actual_departure
```

```sql
--Task 6. Найдите процентное соотношение перелетов по типам самолетов от общего количества. // Использоать подзапрос/окно, оператор ROUND

select a.model,  cnt.acnt/(select count(f2.flight_id)          -- определяем общее количество рейсов count(f2.flight_id) и используем это число для определения
                from flights f2)::dec*100 "part of total flights, %"                                                    --процентного соотношения
  from(                                                        --в подзапросе получаем количество самолетов по их типам из таблицы рейсов
      select f.aircraft_code, count(f.aircraft_code) acnt
      from flights f
      group by f.aircraft_code) cnt
  join aircrafts a on a.aircraft_code =cnt.aircraft_code
  order by a.model
```

```sql
--Task 7. Были ли города, в которые можно  добраться бизнес - классом дешевле, чем эконом-классом в рамках перелета? // Использовать CTE

    with cte_conditions as (            --в cte определяем максимальную стоимость билета для класса Эконом и минимальную для класса Бизнес
select tf.flight_id, tf.fare_conditions,
    case when tf.fare_conditions = 'Economy' then max(tf.amount) end Eco,
    case when tf.fare_conditions = 'Business' then min (tf.amount)  end Bus
from ticket_flights tf
    group by tf.flight_id, tf.fare_conditions)
select f.flight_id, a.city                  -- в основом запросе выводим список городов, при условии, что максимальная стоимость перелета классом Эконом
from flights f                                                          -- будет больше минимальной классом Бизнес (таких городов нет)
    join cte_conditions cte_c  on cte_c.flight_id = f.flight_id
    join airports a on f.arrival_airport = a.airport_code
    group by f.flight_id, a.city
    having max(cte_c.Eco) > min(cte_c.Bus)
```

```sql
--Task 8. Между какими городами нет прямых рейсов? // Использовать  Декартово произведение в предложении FROM,
                                                --Самостоятельно созданные представления (если облачное подключение, то без представления)
                                                --Оператор EXCEPT

    create view cities as(                    -- создаем представление, которое хранит в себе список всех городов отправления и прибытия
select   a.city departure , a2.city arrival                                                 -- между которыми есть прямые рейсы
from flights f
    join airports a on f.departure_airport = a.airport_code
    join airports a2 on f.arrival_airport  = a2.airport_code)

    select a.city departure , a2.city arrival  -- получаем все возможные комбинации городов отправление и прибытия с помощью декартова произведения
from airports a, airports a2
    where a.city > a2.city                    -- причем исключаем зеркальные пары
    except                                    -- в select отправляем данные за исключением тех, что есть в педставлении cities
select* from cities                           -- то есть, остаются города, между которыми нет прямых рейсов
```

```sql
-- Task 9. Вычислите расстояние между аэропортами, связанными прямыми рейсами,
--сравните с допустимой максимальной дальностью перелетов  в самолетах, обслуживающих эти рейсы // Использовать Оператор RADIANS
                                                                                    --или использование sind/cosd
                                                                                    --Использовать оператор CASE


--получаем уникальный список аэропортов отправления и прибытия, между которыми есть прямые рейсы
--по формуле получаем расстояние между аэропортами
-- присоединяем таблицы с самолетами, в соответствии с рейсом, определяем  максимальную дальность полета воздушного судна
-- с проверяем соответствует ли дальность полета ВС расстоянию между аэропортами
-- с помощью оператора CASE выводим соответствующее сообщение, согласно результатам проверки
select distinct  a.airport_name  departure , a2.airport_name  arrival, a3."range",
    round(acos(sind(a.latitude)*sind(a2.latitude) + cosd(a.latitude)*cosd(a2.latitude)*cosd(a.longitude - a2.longitude))*6371)  distance,
case when
    (round(acos(sind(a.latitude)*sind(a2.latitude) + cosd(a.latitude)*cosd(a2.latitude)*cosd(a.longitude - a2.longitude))*6371)) < a3."range"
    then 'arrived'
    else 'crushed'end _status
from flights f
    join airports a  on a.airport_code = f.departure_airport
    join airports a2  on a2.airport_code = f.arrival_airport
    join aircrafts a3 on a3.aircraft_code = f.aircraft_code
    order by a.airport_name , a2.airport_name
```