Итоговая работа

1. Во время работы использовался локальный тип подключения. База была развернута из .backup файла.

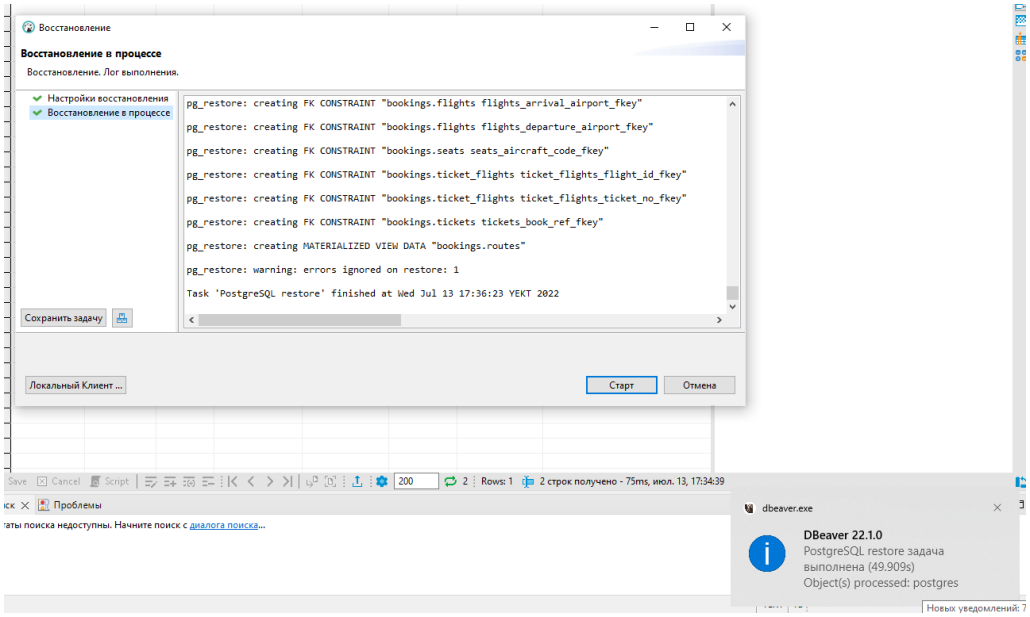

Рисунок 1 - Скриншот восстановления базы из .backup файла

2. Скриншот ER-диаграммы из DBeaver`a приведен на рисунке 2.

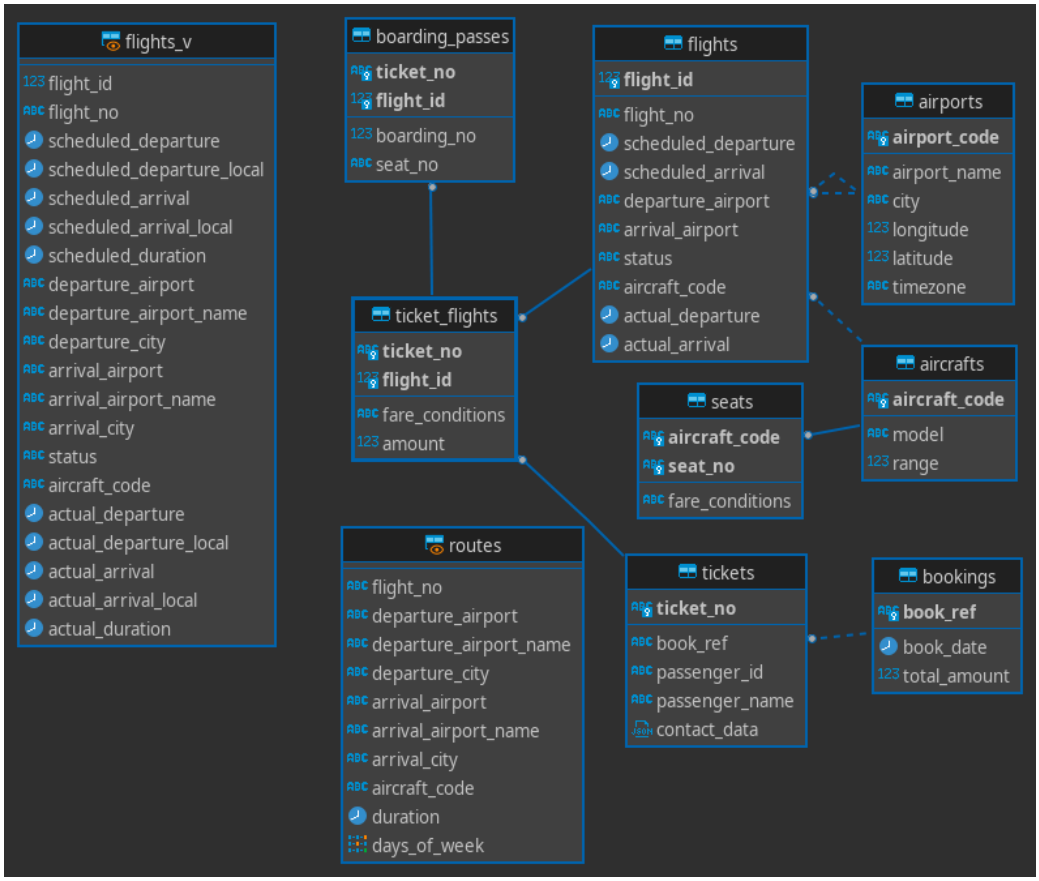

Рисунок 2 - Скриншот ER -диаграммы из из DBeaver`a

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

Основной сущностью является бронирование (bookings).
В одно бронирование можно включить несколько пассажиров, каждому из которых выписывается отдельный билет (tickets). Билет имеет уникальный номер и содержит информацию о пассажире. Как таковой пассажир не является отдельной сущностью. Как имя, так и номер документа пассажира могут меняться с течением времени, так что невозможно однозначно найти все билеты одного человека; для простоты можно считать, что все пассажиры уникальны.
Билет включает один или несколько перелетов (ticket_flights). Несколько перелетов могут включаться в билет в случаях, когда нет нет прямого рейса, соединяющего пункты отправления и назначения (полет с пересадками), либо когда билет взят «туда и обратно».
В схеме данных нет жесткого ограничения, но предполагается, что все билеты в одном бронировании имеют одинаковый набор перелетов.
Каждый рейс (flights) следует из одного аэропорта (airports) в другой. Рейсы с одним номером имеют одинаковые пункты вылета и назначения, но будут отличаться датой отправления.
При регистрации на рейс пассажиру выдается посадочный талон (boarding_passes), в котором указано место в самолете. Пассажир может зарегистрироваться только на тот рейс, который есть у него в билете. Комбинация рейса и места в самолете должна быть уникальной, чтобы не допустить выдачу двух посадочных талонов на одно место.
Количество мест (seats) в самолете и их распределение по классам обслуживания зависит от модели самолета (aircrafts), выполняющего рейс. Предполагается, что каждая модель самолета имеет только одну компоновку салона. Схема данных не контролирует, что места в посадочных талонах соответствуют имеющимся в самолете (такая проверка может быть сделана с использованием табличных триггеров или в приложении)

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

Таблица bookings.aircrafts

Каждая модель воздушного судна идентифицируется своим трехзначным кодом (aircraft_code). Указывается также название модели (model) и максимальная дальность полета в километрах (range).

```
Столбец       | Тип     | Модификаторы | Описание
--------------+---------+--------------+----------------------------------
aircraft_code | char(3) | NOT NULL     | Код самолета, IATA
model         | text    | NOT NULL     | Модель самолета
range         | integer | NOT NULL     | Максимальная дальность полета, км
Индексы:
PRIMARY KEY, btree (aircraft_code)
Ограничения-проверки:
CHECK (range > 0)
Ссылки извне:
TABLE "flights" FOREIGN KEY (aircraft_code)
REFERENCES aircrafts(aircraft_code)
TABLE "seats" FOREIGN KEY (aircraft_code)
REFERENCES aircrafts(aircraft_code) ON DELETE CASCADE
```

Таблица bookings.airports

Аэропорт идентифицируется трехбуквенным кодом (airport_code) и имеет свое имя (airport_name).
Для города не предусмотрено отдельной сущности, но название (city) указывается и может служить для того, чтобы определить аэропорты одного города. Также указывается широта (longitude), долгота (latitude) и часовой пояс (timezone).

```
Столбец      | Тип      | Модификаторы | Описание
-------------+----------+--------------+-----------------------------
airport_code | char(3)  | NOT NULL     | Код аэропорта
airport_name | text     | NOT NULL     | Название аэропорта
city | text  | NOT NULL | Город
longitude    | float    | NOT NULL     | Координаты аэропорта: долгота
latitude     | float    | NOT NULL     | Координаты аэропорта: широта
timezone     | text     | NOT NULL     | Временная зона аэропорта
Индексы:
PRIMARY KEY, btree (airport_code)
Ссылки извне:
TABLE "flights" FOREIGN KEY (arrival_airport)
REFERENCES airports(airport_code)
TABLE "flights" FOREIGN KEY (departure_airport)
REFERENCES airports(airport_code)
```

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).

```
Столбец     | Тип         | Модификаторы | Описание
```

```
------------+-----------+-------------+-------------------------
ticket_no   | char(13)  | NOT NULL    | Номер билета
flight_id   | integer   | NOT NULL    | Идентификатор рейса
boarding_no | integer   | NOT NULL    | Номер посадочного талона
seat_no     | varchar(4)| NOT NULL    | Номер места
Индексы:
PRIMARY KEY, btree (ticket_no, flight_id)
UNIQUE CONSTRAINT, btree (flight_id, boarding_no)
UNIQUE CONSTRAINT, btree (flight_id, seat_no)
Ограничения внешнего ключа:
FOREIGN KEY (ticket_no, flight_id)
REFERENCES ticket_flights(ticket_no, flight_id)
```

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.

```
Столбец      | Тип           | Модификаторы | Описание
-------------+---------------+--------------+--------------------------
book_ref     | char(6)       | NOT NULL     | Номер бронирования
book_date    | timestamptz   | NOT NULL     | Дата бронирования
total_amount | numeric(10,2) | NOT NULL     | Полная сумма бронирования
Индексы:
PRIMARY KEY, btree (book_ref)
Ссылки извне:
TABLE "tickets" FOREIGN KEY (book_ref) REFERENCES bookings(book_ref)
```

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.

```
Столбец              | Тип         | Модификаторы | Описание
---------------------+-------------+--------------+--------------------
flight_id            | serial      | NOT NULL | Идентификатор рейса
flight_no            | char(6)     | NOT NULL | Номер рейса
scheduled_departure | timestamptz | NOT NULL | Время вылета по расписанию
scheduled_arrival    | timestamptz | NOT NULL | Время прилёта по расписанию
departure_airport    | char(3)     | NOT NULL | Аэропорт отправления
arrival_airport      | char(3)     | NOT NULL | Аэропорт прибытия
status               | varchar(20) | NOT NULL | Статус рейса
aircraft_code        | char(3)     | NOT NULL | Код самолета, IATA
actual_departure     | timestamptz |          | Фактическое время вылета
actual_arrival       | timestamptz |          | Фактическое время прилёта
Индексы:
PRIMARY KEY, btree (flight_id)
UNIQUE CONSTRAINT, btree (flight_no, scheduled_departure)
Ограничения-проверки:
CHECK (scheduled_arrival > scheduled_departure)
CHECK ((actual_arrival IS NULL)
OR ((actual_departure IS NOT NULL AND actual_arrival IS NOT NULL)
AND (actual_arrival > actual_departure)))
CHECK (status IN ('On Time', 'Delayed', 'Departed',
'Arrived', 'Scheduled', 'Cancelled'))
Ограничения внешнего ключа:
FOREIGN KEY (aircraft_code)
REFERENCES aircrafts(aircraft_code)
FOREIGN KEY (arrival_airport)
REFERENCES airports(airport_code)
FOREIGN KEY (departure_airport)
REFERENCES airports(airport_code)
Ссылки извне:
TABLE "ticket_flights" FOREIGN KEY (flight_id)
REFERENCES flights(flight_id)
```

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.

```
Столбец          | Тип         | Модификаторы | Описание
-----------------+-------------+--------------+--------------------
aircraft_code    | char(3)     | NOT NULL | Код самолета, IATA
seat_no          | varchar(4)  | NOT NULL | Номер места
fare_conditions | varchar(10) | NOT NULL | Класс обслуживания
Индексы:
PRIMARY KEY, btree (aircraft_code, seat_no)
Ограничения-проверки:
CHECK (fare_conditions IN ('Economy', 'Comfort', 'Business'))
Ограничения внешнего ключа:
FOREIGN KEY (aircraft_code)
REFERENCES aircrafts(aircraft_code) ON DELETE CASCADE
```

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).

```
Столбец         | Тип           | Модификаторы | Описание
----------------+---------------+--------------+--------------------
ticket_no       | char(13)      | NOT NULL     | Номер билета
flight_id       | integer       | NOT NULL     | Идентификатор рейса
fare_conditions | varchar(10)   | NOT NULL     | Класс обслуживания
amount          | numeric(10,2) | NOT NULL     | Стоимость перелета
Индексы:
PRIMARY KEY, btree (ticket_no, flight_id)
Ограничения-проверки:
CHECK (amount >= 0)
CHECK (fare_conditions IN ('Economy', 'Comfort', 'Business'))
Ограничения внешнего ключа:
FOREIGN KEY (flight_id) REFERENCES flights(flight_id)
FOREIGN KEY (ticket_no) REFERENCES tickets(ticket_no)
Ссылки извне:
TABLE "boarding_passes" FOREIGN KEY (ticket_no, flight_id)
REFERENCES ticket_flights(ticket_no, flight_id)
```

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.

```
Столбец        | Тип         | Модификаторы | Описание
---------------+-------------+--------------+----------------------------
ticket_no      | char(13)    | NOT NULL | Номер билета
book_ref       | char(6)     | NOT NULL | Номер бронирования
passenger_id   | varchar(20) | NOT NULL | Идентификатор пассажира
passenger_name | text        | NOT NULL | Имя пассажира
contact_data   | jsonb       |          | Контактные данные пассажира
Индексы:
PRIMARY KEY, btree (ticket_no)
Ограничения внешнего ключа:
FOREIGN KEY (book_ref) REFERENCES bookings(book_ref)
Ссылки извне:
TABLE "ticket_flights" FOREIGN KEY (ticket_no) REFERENCES tickets(ticket_no)
```

Представление "bookings.flights_v"

Над таблицей flights создано представление flights_v, содержащее 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).

Столбец	Тип	Описание
flight_id	integer	Идентификатор рейса
flight_no	char(6)	Номер рейса
scheduled_departure	timestamptz	Время вылета по расписанию
scheduled_departure_local	timestamp	Время вылета по расписанию, местное время в пункте отправления
scheduled_arrival	timestamptz	Время прилёта по расписанию
scheduled_arrival_local	timestamp	Время прилёта по расписанию, местное время в пункте прибытия
scheduled_duration	interval	Планируемая продолжительность полета
departure_airport	char(3)	Код аэропорта отправления
departure_airport_name	text	Название аэропорта отправления
departure_city	text	Город отправления
arrival_airport	char(3)	Код аэропорта прибытия
arrival_airport_name	text	Название аэропорта прибытия
arrival_city	text	Город прибытия
status	varchar(20)	Статус рейса
aircraft_code	char(3)	Код самолета, IATA
actual_departure	timestamptz	Фактическое время вылета
actual_departure_local	timestamp	Фактическое время вылета, местное время в пункте отправления
actual_arrival	timestamptz	Фактическое время прилёта
actual_arrival_local	timestamp	Фактическое время прилёта, местное время в пункте прибытия
actual_duration	interval	Фактическая продолжительность полета

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.

Столбец	Тип	Описание
flight_no	char(6)	Номер рейса
departure_airport	char(3)	Код аэропорта отправления
departure_airport_name	text	Название аэропорта отправления
departure_city	text	Город отправления
arrival_airport	char(3)	Код аэропорта прибытия
arrival_airport_name	text	Название аэропорта прибытия
arrival_city	text	Город прибытия
aircraft_code	char(3)	Код самолета, IATA
duration	interval	Продолжительность полета
days_of_week	integer[]	Дни недели, когда выполняются рейсы

Функция now

Демонстрационная база содержит временной «срез» данных — так, как будто в некоторый момент была сделана резервная копия реальной системы. Например, если некоторый рейс

имеет статус Departed, это означает, что в момент резервного копирования самолет вылетел и находился в воздухе.
Позиция «среза» сохранена в функции bookings.now(). Ей можно пользоваться в запросах там, где в обычной жизни использовалась бы функция now().
Кроме того, значение этой функции определяет версию демонстрационной базы данных. Актуальная версия на текущий момент — от 13.10.2016.

Использование

Схема bookings
Все объекты демонстрационной базы данных находятся в схеме bookings. Это означает, что при обращении к объектам вам необходимо либо явно указывать имя схемы (например: `bookings.flights`), либо предварительно изменить конфигурационный параметр search_path (например: `SET search_path = bookings, public;`).
Однако для функции bookings.now в любом случае необходимо явно указывать схему, чтобы отличать ее от стандартной функции now.

Бизнес задачи, которые можно решить, используя БД

1. Анализ загруженности самолетов

2. Анализ загруженности пассажиропотоков в срезах: направлений перелетов, временных параметрах (день недели, месяцы)

3. Анализ данных по невостребованным билетам, отсутствию посадки на купленные билеты.

4. Получение данных об одновременном нахождении самолетов в воздухе

5. Анализ рентабельности бизнеса (в связке с другими данными)

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

1. В каких городах больше одного аэропорта?

```
SELECT city, count(*) "Кол-во аэропортов"
FROM bookings.airports a
GROUP BY city -- делаем группировку по городам
HAVING count(*) >1; -- если строк в группе более 1, значит там более 1 аэропорта
```

2. В каких аэропортах есть рейсы, выполняемые самолетом с максимальной дальностью перелета?

```
SELECT DISTINCT airport_code, (
	SELECT max("range")
	FROM bookings.aircrafts) Дальность
FROM bookings.airports
JOIN bookings.flights
ON departure_airport = airport_code OR arrival_airport = airport_code
WHERE aircraft_code = (
	SELECT aircraft_code -- находим код самолета с максимальной дальностью
	FROM bookings.aircrafts
	WHERE "range" = (
	SELECT max("range") -- находим максимальную дальность
	FROM bookings.aircrafts));
```

3. Вывести 10 рейсов с максимальным временем задержки вылета

Используем данные из bookings.flights

```
SELECT flight_id, flight_no, actual_departure - scheduled_departure "Задержка"
FROM bookings.flights
WHERE actual_departure IS NOT NULL -- исключаем рейсы без задержек
ORDER BY actual_departure - scheduled_departure DESC
LIMIT 10;
```

4. Были ли брони, по которым не были получены посадочные талоны

Используем данные из объединения таблиц bookings.ticket_flights и boarding_passes через left join для получения билетов без брони по значению is null атрибута boarding_no

```
SELECT ticket_no
FROM bookings.ticket_flights
	LEFT JOIN boarding_passes USING (ticket_no) --объединяем с влючением значений NULL из правой таблицы
WHERE boarding_no IS NULL --отбираем те билеты, по которым не было брони
ORDER BY ticket_no;
```

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.

В первом cte находим общее количество мест для каждого рейса
по общему количеству мест в самолете, забронированному на данный рейс

```
with seats_flats_cte as (
select flight_id, count(*) count_seats, departure_airport, scheduled_departure, actual_departure,
case
	when actual_departure is null then scheduled_departure
	else actual_departure
end fact_departure --фактическое время вылета
from
	bookings.flights f
	join bookings.seats s  using (aircraft_code)
group by flight_id , aircraft_code -- группируем для подсчета количества мест в самолете по рейсам
```

```
order by flight_id),
--Находим количество посадочных талонов для каждого рейса
count_boarding_passes_cte as (
select flight_id, count(*) count_passes
from bookings.boarding_passes bp
group by flight_id
order by flight_id),
--Находим количество свободных мест для каждого рейса
free_seats_flats_cte as (
select *
from
	seats_flats_cte --Объединяем через left join, чтобы учесть null
	left join count_boarding_passes_cte using (flight_id))
select flight_id, fact_departure,
case
	when count_passes is null then count_seats
	else count_seats - count_passes
end free_seats,
case
	when count_passes is null then 100
	else round(((count_seats - count_passes)::numeric/count_seats)*100, 1)
end "free_seats, %",
coalesce(sum(count_passes) over(partition by departure_airport, fact_departure::date order by
fact_departure), 0) "count_sum"
from free_seats_flats_cte
order by departure_airport, fact_departure;

--6. Найдите процентное соотношение перелетов по типам самолетов от общего количества.
select distinct
	aircraft_code,
	round((count(aircraft_code) over(partition by aircraft_code))::numeric / (
	select count(*) from bookings.flights), 3)*100 "Доля, %"
from
	bookings.flights;

7. Были ли города, в которые можно  добраться бизнес - классом дешевле, чем эконом-классом в
рамках перелета?


--Вариант 1
--Для начала находим стоимость перелетов по рейсам и классам
with amount_city_cte as (
select distinct
	flight_id, amount, fare_conditions
from
	bookings.ticket_flights tf
	join bookings.flights using(flight_id)
	join bookings.aircrafts using(aircraft_code)),
-- отбираем перелеты по классу Economy
amount_city_cte_economy as (
select *
from amount_city_cte
where fare_conditions = 'Economy'),
-- отбираем перелеты по классу Business
amount_city_cte_business as (
select *
from amount_city_cte
where fare_conditions = 'Business')
-- соединяем cte перелетов по разным классам по условию равенства flight_id
select
	flight_id
from
```

```
	amount_city_cte_economy
	join amount_city_cte_business using(flight_id)
where amount_city_cte_business.amount < amount_city_cte_economy.amount; -- отбираем по этому условию
--Результат нулевой. Значит не было


--Вариант 2
--Для начала находим максимальные и минимальные стоимость перелетов по рейсам и классам для Economy и Business
--С группировкой по flight_id, fare_conditions

with amount_city_cte as (
SELECT
	flight_id, fare_conditions,
	CASE
		WHEN fare_conditions = 'Economy' THEN max(amount)
		ELSE min(amount)
	END amount
from
	bookings.ticket_flights tf
	join bookings.flights using(flight_id)
	join bookings.aircrafts using(aircraft_code)
WHERE tf.fare_conditions = 'Business' OR tf.fare_conditions = 'Economy'
GROUP BY flight_id, fare_conditions
ORDER BY flight_id, fare_conditions
	)
	SELECT flight_id
	from
		(SELECT flight_id, array_agg(fare_conditions) fare_conditions, array_agg(amount) amount
		FROM amount_city_cte
		GROUP BY flight_id
		HAVING count(*) > 1) q
	WHERE amount[1] < amount[2];  --проверяем значения в массиве
--Результат нулевой. Значит не было


--Вариант 3
--Для начала находим максимальные и минимальные стоимость перелетов по рейсам и классам для Economy и Business
--С группировкой по flight_id, fare_conditions
with amount_city_cte as (
SELECT
	flight_id, fare_conditions,
	CASE --для Economy берем максимальную стоимость для рейса
		WHEN fare_conditions = 'Economy' THEN max(amount)
		ELSE min(amount) --для Business берем минимальную стоимость для рейса
	END amount
from
	bookings.ticket_flights tf
	join bookings.flights using(flight_id)
	join bookings.aircrafts using(aircraft_code)
WHERE fare_conditions = 'Business' OR fare_conditions = 'Economy'
GROUP BY flight_id, fare_conditions
ORDER BY flight_id, fare_conditions
	)
SELECT flight_id
FROM (
		SELECT *,
	 --добавляем колонку значений минимальной стоимости бизнес класса для рейса
		lag(amount, 1, amount) OVER(PARTITION BY flight_id) business_amount
```

```
                FROM amount_city_cte) q
WHERE amount > business_amount;
--Результат нулевой. Значит не было

-- 8. Между какими городами нет прямых рейсов?
-- Решение без представлений
-- Для начала находим всевозможные сочетания между аэропортами разных городов
with city_city_decart_cte as (
select distinct
        lower(a.city)||' - '||lower(q.city) flight_name,
        a.city city_1, a.airport_code airport_code_1, q.city city_2, q.airport_code airport_code_2
from
        bookings.airports a
        cross join (select airport_code, city from bookings.airports) q
where
        a.airport_code != q.airport_code and a.city != q.city
order by flight_name),
-- Находим фактические сочетания между аэропортами разных городов
city_city_fact_cte as (
select distinct
        lower(city_1)||' - '||lower(city) flight_name,
        city_1, airport_code_1, city city_2, airport_code_2
from (
        select city city_1, q1.departure_airport airport_code_1, q1.arrival_airport airport_code_2
        from bookings.airports
        join (
                select departure_airport, arrival_airport
                from bookings.flights) q1
        on departure_airport = airport_code) q2
join bookings.airports on airport_code_2 = airport_code
order by flight_name)
-- Находим рейсы не вошедшие в декартово сочетание
-- Это и будут варианты, между которыми нет сообщений

select initcap(flight_name) flight_name
from city_city_decart_cte
except select initcap(flight_name) from city_city_fact_cte
order by 1;

-- 8. Вариант с представлениями
-- Для начала находим возможные декартовы сочетания между аэропортами разных городов
create view city_city_decart_view as
select distinct
        lower(a.city)||' - '||lower(q.city) flight_name,
        a.city city_1, a.airport_code airport_code_1, q.city city_2, q.airport_code airport_code_2
from
        bookings.airports a
        cross join (select airport_code, city from bookings.airports) q
where
        a.airport_code != q.airport_code and a.city != q.city
order by flight_name;

-- Находим фактические сочетания между аэропортами разных городов
create view city_city_fact_view as
select distinct
        lower(city_1)||' - '||lower(city) flight_name,
        city_1, airport_code_1, city city_2, airport_code_2
from (
        select city city_1, q1.departure_airport airport_code_1, q1.arrival_airport airport_code_2
        from bookings.airports
        join (
                select departure_airport, arrival_airport
```

```
                from bookings.flights) q1
        on departure_airport = airport_code) q2
join bookings.airports on airport_code_2 = airport_code
order by flight_name;

-- Находим рейсы не вошедшие в декартово сочетание
-- Это и будут варианты, между которыми нет сообщений

select initcap(flight_name) flight_name from city_city_decart_view
except select initcap(flight_name) flight_name from city_city_fact_view
order by 1;
-- Другой вариант
select city_1, city_2 from city_city_decart_view
except select city_1, city_2 from city_city_fact_view
order by city_1, city_2;
```

9.Вычислите расстояние между аэропортами, связанными прямыми рейсами,
сравните с допустимой максимальной дальностью перелетов в самолетах, обслуживающих эти рейсы

```
-- Используем ранее созданное представление city_city_fact_view. На основе него создаем cte.
-- Можно сделать и без ранее созданного представления, просто тогда cte будет включать в себя его логику
-- В итоговом запросе добавляем информацию из таблицы aircraft

WITH initial_data AS -- получаем исходные данные для дальнейшей работы
(
SELECT
        initcap(flight_name) flight_name,
        airport_code_1, a1.longitude lg1, a1.latitude lt1,
        airport_code_2, a2.longitude lg2, a2.latitude lt2
FROM
        city_city_fact_view c
        JOIN airports a1 ON c.airport_code_1 = a1.airport_code
        JOIN airports a2 ON c.airport_code_2 = a2.airport_code
),
dist_cte_rad AS --получаем расстояние в радианах
(
SELECT *,
acos(sind(lt1)*sind(lt2) + cosd(lt1)*cosd(lt2)*cosd(lg1 - lg2)) dist_rad
FROM initial_data
),
dist_cte_km AS --получаем расстояние в км
(
SELECT
        flight_name,
        airport_code_1, lg1, lt1,
        airport_code_2, lg2, lt2,
        round(6371*dist_rad::NUMERIC, 0) dist_km
FROM dist_cte_rad
)
SELECT DISTINCT
        flight_name, dist_km,
        "range",
        CASE
                WHEN "range" >= dist_km THEN 'Успех'
                ELSE 'Провал'
        END   Результат
FROM
        dist_cte_km
        JOIN bookings.flights f --добавляем информацию из bookings.aircrafts
                ON airport_code_1 = f.departure_airport
                and airport_code_2 = f.arrival_airport
```

```
	JOIN bookings.aircrafts a USING (aircraft_code)
ORDER BY flight_name;
```

Вариант без представления

```
WITH city_city_fact_cte AS -- Находим фактические сочетания между аэропортами разных городов
(
SELECT DISTINCT
	lower(city_1)||' - '||lower(city) flight_name,
	city_1, airport_code_1, city city_2, airport_code_2
from (
	select city city_1, q1.departure_airport airport_code_1, q1.arrival_airport airport_code_2
	from bookings.airports
	join (
		select departure_airport, arrival_airport
		from bookings.flights) q1
	on departure_airport = airport_code) q2
join bookings.airports on airport_code_2 = airport_code
order by flight_name
),
initial_data AS -- получаем исходные данные для дальнейшей работы
(
SELECT
	initcap(flight_name) flight_name,
	airport_code_1, a1.longitude lg1, a1.latitude lt1,
	airport_code_2, a2.longitude lg2, a2.latitude lt2
FROM
	city_city_fact_cte c
	JOIN airports a1 ON c.airport_code_1 = a1.airport_code
	JOIN airports a2 ON c.airport_code_2 = a2.airport_code
),
dist_cte_rad AS --получаем расстояние в радианах
(
SELECT *,
acos(sind(lt1)*sind(lt2) + cosd(lt1)*cosd(lt2)*cosd(lg1 - lg2)) dist_rad
FROM initial_data
),
dist_cte_km AS --получаем расстояние в км
(
SELECT
	flight_name,
	airport_code_1, lg1, lt1,
	airport_code_2, lg2, lt2,
	round(6371*dist_rad::NUMERIC, 0) dist_km
FROM dist_cte_rad
)
SELECT DISTINCT
	flight_name, dist_km,
	"range",
	CASE
		WHEN "range" >= dist_km THEN 'Успех'
		ELSE 'Провал'
	END	Результат
FROM
	dist_cte_km
	JOIN bookings.flights f --добавляем информацию из bookings.aircrafts
		ON airport_code_1 = f.departure_airport
		and airport_code_2 = f.arrival_airport
	JOIN bookings.aircrafts a USING (aircraft_code)
ORDER BY flight_name;
```