

# Проектная работа по модулю “SQL и получение данных”

## С о д е р ж а н и е

- 1 [Установка базы на локальном компьютере](#)
- 2 [Диаграмма схемы данных](#)
- 3 [Описание схемы](#)
- 4 [Объекты схемы](#)
- 5 [Задания. Создание запросов](#)

## 1 Установка базы на локальном компьютере

Базу данных разворачиваем в Docker-контейнере. За основу взят докерфайл:  
<https://github.com/mgramin/docker-postgres-up-from-dump>

Описание:

# Загружаем официальный образ PostgreSQL

FROM postgres:12

# Создаем необходимые каталоги

RUN mkdir -p /var/lib/postgresql-static/data

ENV PGDATA /var/lib/postgresql-static/data

# Образ установлен на ОС Debian. Обновляем систему и устанавливаем unzip

RUN apt-get update

RUN apt-get install wget unzip -y

# Загружаем и распаковываем дампы базы

ARG DUMP\_URI=https://edu.postgrespro.ru/demo-small.zip

RUN wget --output-document=dbdump.zip \$DUMP\_URI

RUN unzip dbdump.zip -d dbdump

RUN echo "" > /docker-entrypoint-initdb.d/run\_dbdump.sh

# С помощью утилиты psql создаем шаблон базы,  
устанавливаем библиотеки, в том числе  
"cube", "earthdistance" для определения расстояний  
# Загружаем дампы базы

RUN echo "psql -t template1 -U postgres -c 'alter system set  
shared\_preload\_libraries=\"pg\_stat\_statements\", \"pg\_buffercache\", \"cube\", \"earthdistance\"  
;'" >> /docker-entrypoint-initdb.d/run\_dbdump.sh

RUN echo "psql -U postgres -f /dbdump/demo-small-20170815.sql" >>  
/docker-entrypoint-initdb.d/run\_dbdump.sh

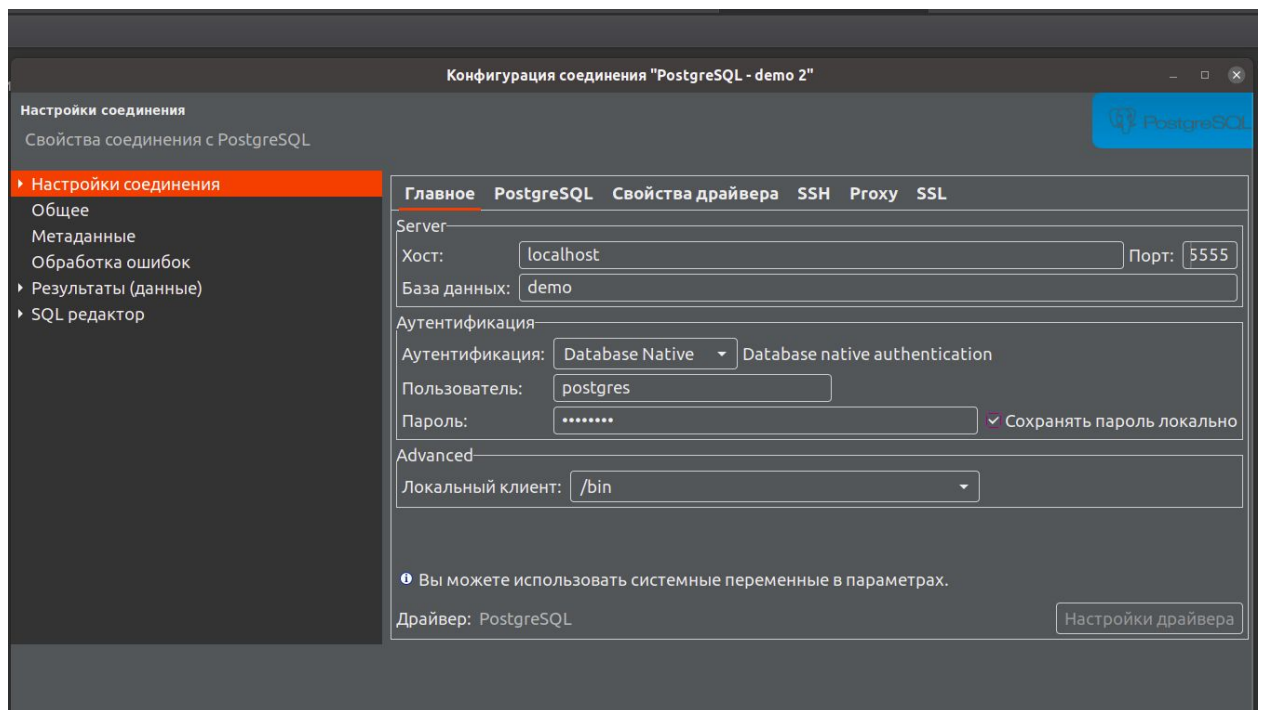
Сборка образа командой

docker build --build-arg DUMP\_URI=https://edu.postgrespro.ru/demo-small.zip -t  
airbooking .

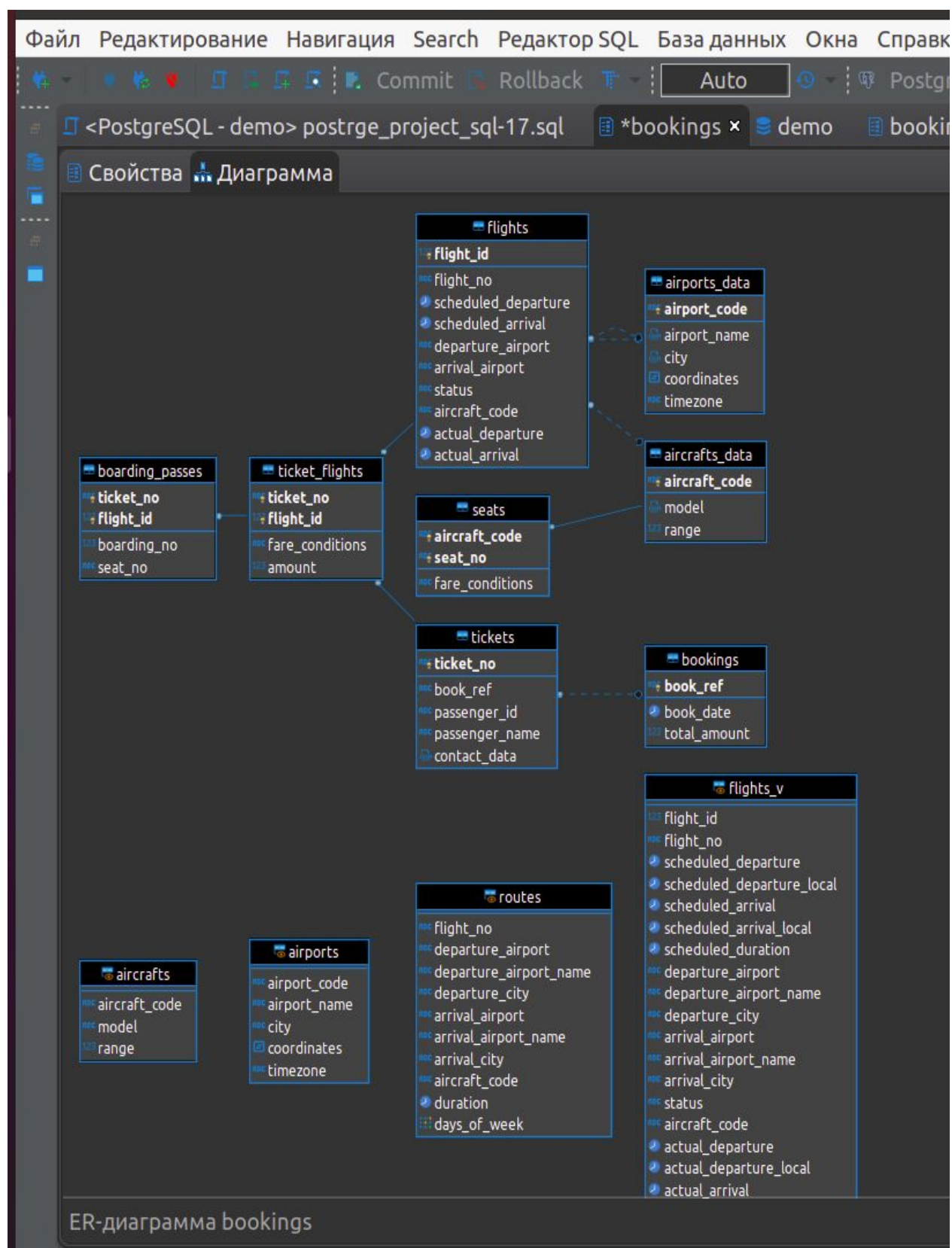
Запуск образа на порту 5555 хоста, так как на  
нем уже установлена Postgres

docker run -t -p 5555:5432 -e POSTGRES\_PASSWORD=password airbooking

Конфигурация соединения в dbeaver



## 2 Диаграмма схемы данных



### 3 Описание схемы

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 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).

## 4 Объекты схемы

### 4.1 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

### 4.2 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

### 4.3 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) в формате point(latitude, longitude), 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

#### 4.4 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

#### 4.5 bookings.bookings

Пассажир заранее (book\_date, максимум за месяц до рейса) бронирует билет себе и, возможно, нескольким другим пассажирам. Бронирование идентифицируется номером (book\_ref, шести значная 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	timestamp tz	NOT NULL	Дата бронирования
total_amount	numeric(10,2)	NOT NULL	Полная сумма бронирования



## 4.6 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 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.
- **OnTime** 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
<code>flight_id</code>	<code>serial</code>	<code>NOT NULL</code>	Идентификатор рейса
<code>flight_no</code>	<code>char(6)</code>	<code>NOT NULL</code>	Номер рейса
<code>scheduled_departure</code>	<code>timestampz</code>	<code>NOT NULL</code>	Время вылета по расписанию
<code>scheduled_arrival</code>	<code>timestampz</code>	<code>NOT NULL</code>	Время прилёта по расписанию
<code>departure_airport</code>	<code>char(3)</code>	<code>NOT NULL</code>	Аэропорт отправления
<code>arrival_airport</code>	<code>char(3)</code>	<code>NOT NULL</code>	Аэропорт прибытия
<code>status</code>	<code>varchar(20)</code>	<code>NOT NULL</code>	Статус рейса
<code>aircraft_code</code>	<code>char(3)</code>	<code>NOT NULL</code>	Код самолета, IATA
<code>actual_departure</code>	<code>timestampz</code>		Фактическое время вылета
<code>actual_arrival</code>	<code>timestampz</code>		Фактическое время прилёта

## 4.7 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 и ли 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

#### 4.8 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

#### 4.9 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\_data). 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

#### 4.10 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	timestamp	Время вылета по расписанию
scheduled_departure_local	timestamp	Время вылета по расписанию, местное время в пункте отправления
scheduled_arrival	timestamp	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	timestamp	Фактическое время вылета
actual_departure_local	timestamp	Фактическое время вылета, местное время в пункте отправления
actual_arrival	timestamp	Фактическое время прилёта
actual_arrival_local	timestamp	Фактическое время прилёта, местное время в пункте прибытия
actual_duration	interval	Фактическая продолжительность полета

#### 4.11 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



## 5 Задания

### 5.1 В каких городах больше одного аэропорта?

Группируем города из таблицы airports\_data по количеству аэропортов. При этом наименования аэропорта получаем по ключу 'ru' из json формата

```
select a.city ->> 'ru' as city, count(a.airport_name)
from airports_data a
group by a.city
having count(a.airport_name) > 1;
```

city	count
Москва	3
Ульяновск	2

### 5.2 В каких аэропортах есть рейсы, которые обслуживаются самолетами с максимальной дальностью перелетов?

Создаем общее табличное выражение range\_max, которое выдает максимальную дальность перелета. Объединяем таблицы aircrafts\_data, flights, airports\_data. Выбираем строки где дальность перелета равна максимальной

```
with range_max as(
  select a_d."range"
  from aircrafts_data a_d
  order by "range" desc limit 1)
select distinct
  a_d.model ->> 'ru' as model_name,
  a_d."range",
  a.airport_name ->> 'ru' as airport_
from aircrafts_data a_d
join flights f on f.aircraft_code = a_d.aircraft_code
```

```
join airports_data a on a.airport_code = f.arrival_airport or airport_code = f.departure_airport
```

```
where "range" = (select * from range_max);
```

	abc model_name 🔼🔼	123 range 🔼🔼	abc airport_ 🔼🔼
1	Боинг 777-300	11 100	Сочи
2	Боинг 777-300	11 100	Толмачёво
3	Боинг 777-300	11 100	Кольцово
4	Боинг 777-300	11 100	Шереметьево
5	Боинг 777-300	11 100	Домодедово
6	Боинг 777-300	11 100	Пермь
7	Боинг 777-300	11 100	Внуково

### 5.3 Были ли брони, по которым не совершались перелеты?

Выбираем бронирования по которым посадочный талон не выдавался

```
select t.book_ref, t_f.flight_id, b.boarding_no
```

```
from tickets t
```

```
join ticket_flights t_f using(ticket_no)
```

```
join boarding_passes b using(ticket_no)
```

```
where t.book_ref is null;
```

	abc book_ref 🔼🔼	123 flight_id 🔼🔼	123 boarding_no 🔼🔼

Таких броней не было

### 5.4 Самолеты каких моделей совершают наибольший % перелетов?

Создаем общее табличное выражение summ, которое выдает максимальную дальность перелета. Считаем проценты полетов группируя по модели самолета. Упорядочим по убыванию процентов

```
with summ as (select count (*) from flights)
```

```
select a_d.model ->> 'ru' as model_name, round(100 * count(*)/(select * from summ)::numeric, 1) as perc
```

```

from flights f

join aircrafts_data a_d using(aircraft_code)

group by a_d.model

order by perc desc;

```

	model_name	perc
1	Сессна 208 Караван	28
2	Бомбардье CRJ-200	27,3
3	Сухой Суперджет-100	25,7
4	Аэробус А321-200	5,9
5	Боинг 737-300	3,8
6	Аэробус А319-100	3,7
7	Боинг 767-300	3,7
8	Боинг 777-300	1,8

## 5.5 Были ли города, в которые можно добраться бизнес-классом дешевле, чем эконом-классом?

Создаем два представления с номерами рейса, классом обслуживания, стоимостью и маршрутом для бизнес и эконом класса в отдельности.. Объединяем в одну таблицу по номеру вылета. Выбираем строки, где стоимость перелета бизнес-классом дешевле эконом-классом

```
create or replace view econimy_amount as
```

```

    select f.flight_id, tf.amount, tf.fare_conditions, f.departure_airport || ' - ' ||
f.arrival_airport as e_route

```

```

from ticket_flights tf

```

```

join flights f using(flight_id)

```

```

join airports_data a on a.airport_code = f.arrival_airport

```

```

where tf.fare_conditions = 'Economy';

```

```
create or replace view business_amount as
```

```

    select f.flight_id, tf.amount, tf.fare_conditions, f.departure_airport || ' - ' ||
f.arrival_airport as b_route

```



```

from ticket_flights tf

join flights f using(flight_id)

join airports_data a on a.airport_code = f.arrival_airport

where tf.fare_conditions = 'Business';

select ba.flight_id, ba.b_route, ba.amount business_amount, ea.amount
economy_amount

from business_amount_ ba

join economy_amount_ ea using(flight_id)

where ba.amount < ea.amount;

```

flight_id	b_route	business_amount	economy_amount

## 5.6 Узнать максимальное время задержки вылетов самолетов

Время задержки это разница между фактическим временем отправления и расписанием. Убираем те случаи, когда самолет еще не вылетел, тут максимальное время на момент среза данных еще не известно

```

select max(f.actual_departure - f.scheduled_departure)

from flights f

WHERE f.actual_departure is not null;

```

	max
1	04:41:00

## 5.7 Между какими городами нет прямых рейсов\*?

Создаем два представления. Множество всех возможных вариантов пар городов. И множество пар существующих рейсов. Вычитаем одно множество из другого.

create or replace view cities\_ as

```
select a.city || ' - ' || b.city as routes
```

```
from airports a,airports b
```

```
where a.city != b.city;
```

create or replace view existing as

```
select distinct a.city as departure_city,
```

```
b.city as arrival_city
```

```
from flights f
```

```
join airports a on f.departure_airport = a.airport_code
```

```
join airports b on f.arrival_airport = b.airport_code;
```

```
(select ci.routes from cities_ ci) except (select er.departure_city || ' - ' || er.arrival_city as routes from existing er);
```

	routes
1	Воронеж - Анадырь
2	Хабаровск - Тюмень
3	Ханты-Мансийск - Иркутск
4	Йошкар-Ола - Пермь
5	Бугульма - Сургут
6	Петрозаводск - Ставрополь
7	Новый Уренгой - Горно-Алтайск
8	Череповец - Казань
9	Анапа - Ухта
10	Тамбов - Новокузнецк
11	Архангельск - Элиста

## 5.8 Между какими городами пассажиры делали пересадки\*?

Создаем материализованное представление transfers, которое по номеру билета собирает наименования всех аэропортов всех рейсов каждого билета в один массив. Убираем повторяющиеся элементы массива - это аэропорты в которых были пересадки, а нам нужны конечные пункты. Здесь же считаем время задержки в аэропорту и выбираем строки с задержкой от 0 минут до 24 часов.

По обозначению аэропорта выбираем город

create MATERIALIZED VIEW IF NOT EXISTS transfers as

```
select distinct tf.ticket_no, max(f.scheduled_departure) - min(f.scheduled_arrival) as
transfer_time,

(

(select unnest(regexp_split_to_array(String_agg(f.departure_airport, ' '), '\s+'))
except (select unnest(regexp_split_to_array(String_agg(f.arrival_airport, ' '), '\s+'))

) as airport_code

from ticket_flights tf

join flights f using(flight_id)

group by tf.ticket_no

having (EXTRACT(minute from max(f.scheduled_departure) - min(f.scheduled_arrival)) >
0) and (

EXTRACT(day from max(f.scheduled_departure) - min(f.scheduled_arrival)) < 1)

and (not regexp_split_to_array(String_agg(f.departure_airport, ' '), '\s+') @>
regexp_split_to_array(String_agg(f.arrival_airport, ' '), '\s+'));

select distinct city ->> 'ru' city

from transfers t

join airports_data a_d using(airport_code);
```

	city
1	Красноярск
2	Уфа
3	Новый Уренгой
4	Минеральные Воды
5	Когалым
6	Краснодар
7	Казань
8	Новокузнецк
9	Екатеринбург
10	Омск
11	Пермь

## 5.9 Вычислите расстояние между аэропортами, связанными прямыми рейсами, сравните с допустимой максимальной дальностью перелетов в самолетах, обслуживающих эти рейсы \*\*

Создаем функцию, вычисляющую расстояние по известной формуле. Предварительно градусы переводим в радианы. применим эту функцию в запросе, который выдает города рейса, модель самолета, максимальную дальность и расстояние по поверхности сферы между этими городами

```
CREATE OR REPLACE function earthdistance(point, point)
```

```
RETURNS double precision
```

```
AS '
```

```
select round(6371 * (acos( sin(radians($1[1])) * sin(radians($2[1])) + cos(radians($1[1])) * cos(radians($2[1])) * cos(radians($1[0] - $2[0])))) ));
```

```
-- select sin(60);
```

```
,
```

```
LANGUAGE sql ;
```

```
select distinct
```

```
ap_d.airport_name ->> 'ru' dep_airport, ap_a.airport_name ->> 'ru' arr_airport,
```

```
ad.model ->> 'ru' model, ad."range",
```











```
earthdistance(ap_d.coordinates, ap_a.coordinates)
```

```
from flights f
```

```
join aircrafts_data ad using(aircraft_code)
```

```
join airports_data ap_d on f.departure_airport = ap_d.airport_code
```

```
join airports_data ap_a on ap_a.airport_code = f.arrival_airport ;
```

	 dep_airport 	 arr_airport 	 model 	 range 	 earthdistance 
1	Абакан	Богашёво	Сессна 208 Караван	1 200	491
2	Абакан	Грозный	Боинг 737-300	4 200	3 484
3	Абакан	Домодедово	Аэробус А319-100	6 700	3 366
4	Абакан	Кызыл	Сессна 208 Караван	1 200	307
5	Абакан	Талаги	Аэробус А319-100	6 700	3 028
6	Абакан	Толмачёво	Сессна 208 Караван	1 200	583
7	Анадырь	Внуково	Аэробус А319-100	6 700	6 220
8	Анадырь	Домодедово	Аэробус А319-100	6 700	6 226
9	Анадырь	Хабаровск-Новый	Аэробус А319-100	6 700	3 074
10	Анадырь	Шереметьево	Аэробус А319-100	6 700	6 177
11	Астрахань	Барнаул	Бомбардье CRJ-200	2 700	2 637