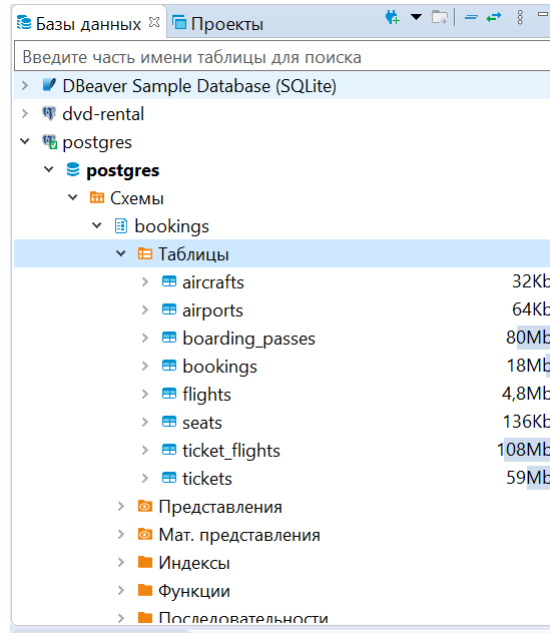
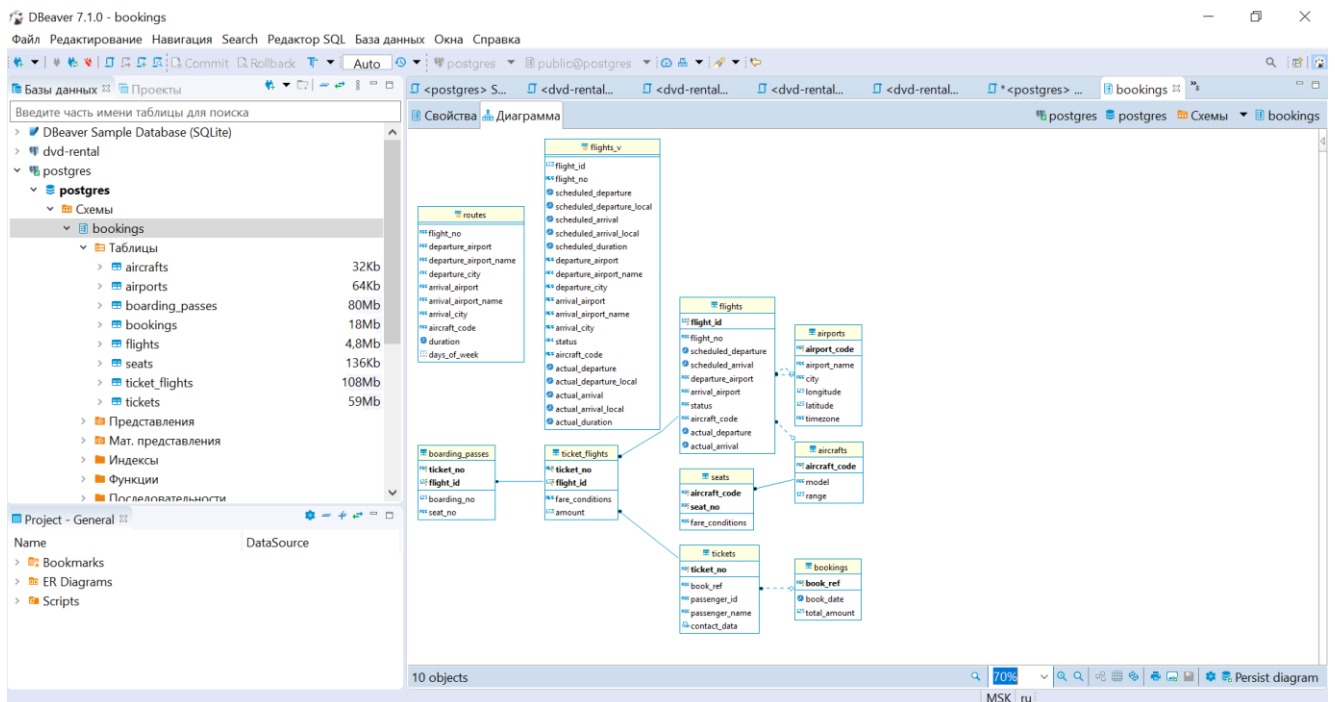


Итоговая работа

1. В работе использовался локальный тип подключения, через восстановление *.backup файла.



2. Скриншот ER-диаграммы из DBeaver'a



3. Данная БД состоит из 8 таблиц, 1 представления и 1 материализованного представления. Список таблиц и представлений БД:

Наименование таблицы (представления)	Тип	Описание
aircrafts	таблица	Самолеты
airports	таблица	Аэропорты
boarding_passes	таблица	Посадочные талоны
bookings	таблица	Бронирования
flights	таблица	Рейсы
flights_v	представление	Рейсы
routes	мат. представление	Маршруты
seats	таблица	Места
ticket_flight	таблица	Перелеты
tickets	таблица	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
<code>ticket_no</code>	<code>char(13)</code>	<code>NOT NULL</code>	Номер билета
<code>flight_id</code>	<code>integer</code>	<code>NOT NULL</code>	Идентификатор рейса
<code>boarding_no</code>	<code>integer</code>	<code>NOT NULL</code>	Номер посадочного талона
<code>seat_no</code>	<code>varchar(4)</code>	<code>NOT NULL</code>	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
<code>book_ref</code>	<code>char(6)</code>	<code>NOT NULL</code>	Номер бронирования
<code>book_date</code>	<code>timestampz</code>	<code>NOT NULL</code>	Дата бронирования
<code>total_amount</code>	<code>numeric(10,2)</code>	<code>NOT NULL</code>	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	timestampz	NOT NULL	Время вылета по расписанию
scheduled_arrival	timestampz	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	timestampz		Фактическое время вылета
actual_arrival	timestampz		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_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

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:

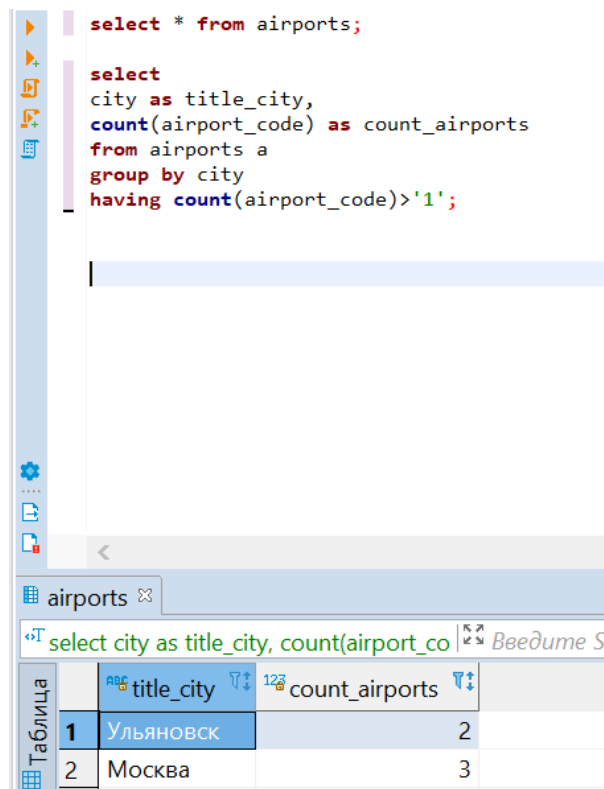
- расшифровку данных об аэропорте вылета (departure_airport, departure_airport_name, departure_city),
- расшифровку данных об аэропорте прибытия (arrival_airport, arrival_airport_name, arrival_city),
- местное время вылета (scheduled_departure_local, actual_departure_local),
- местное время прибытия (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

5. В каких городах больше одного аэропорта?



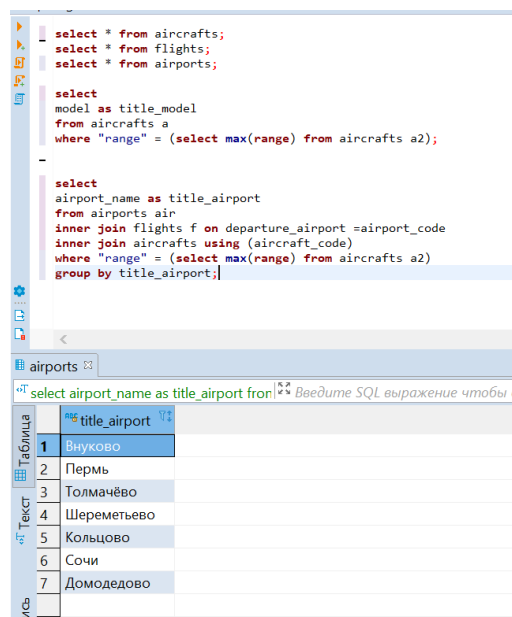
```
select * from airports;

select
city as title_city,
count(airport_code) as count_airports
from airports a
group by city
having count(airport_code)>'1';
```

	title_city	count_airports
1	Ульяновск	2
2	Москва	3

Сперва мы выводим название города `city as title_city`, затем мы применяем агрегатную функцию `count` к идентификатору аэропорта (`airport_code`) и выводим количество аэропортов в каждом городе, далее делаем группировку по названию города путем применения оператора `group by`, чтобы вывести города, в которых более одного аэропорта мы завершаем запрос применением оператора `having`, тем самым мы отфильтровываем значения равные 1 и выводим название городов, в которых два и более аэропортов (в нашем примере таких два города: Москва и Ульяновск).

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



```
select * from aircrafts;
select * from flights;
select * from airports;

select
model as title_model
from aircrafts a
where "range" = (select max(range) from aircrafts a2);

select
airport_name as title_airport
from airports air
inner join flights f on departure_airport = airport_code
inner join aircrafts using (aircraft_code)
where "range" = (select max(range) from aircrafts a2)
group by title_airport;
```

	title_airport
1	Внуково
2	Пермь
3	Толмачёво
4	Шереметьево
5	Кольцово
6	Сочи
7	Домодедово

Для данного запроса мы используем данные трех таблиц `aircrafts`, `flights`, `airports`.

На основе данных о моделях самолетов мы путем запроса и вложенного в него подзапроса узнаем модель самолет с максимальной дальностью полета:

```
select
model as title_model
from aircrafts a
where "range" = (select max(range) from aircrafts a2);
```

После того, как мы узнали модель самолета с максимальной дальностью полета, мы формируем запрос, который выводит название аэропортов из которых осуществляются вылеты моделью самолета с максимальной дальностью полета.

Так как название аэропорта, модель самолета, дальность полета содержатся в разных таблицах мы применяем оператор **inner join** для связи таблиц и вывода нужных нам данных:

```
inner join flights f on departure_airport =airport_code
inner join aircrafts using (aircraft_code)
```

Далее мы применяем ограничение на то, что нам необходимо вывести аэропорт в котором осуществляются вылеты интересующим нас дальнемагистральной моделью самолета, во избежание повтора строк мы применяем оператор group by ставя сортировку по названию аэропорта:

```
where "range" = (select max(range) from aircrafts a2)
group by title_airport;
```

7. Вывести 10 рейсов с максимальным временем задержки вылета.

```
select * from flights;

select
flight_no,
status,
scheduled_departure-actual_departure as delay
from flights
where status='Departed'
order by delay asc
limit 10;
```

flights

select flight_no, status, scheduled_dep: Введите SQL выражение

	flight_no	status	delay
1	PG0574	Departed	-03:15:00
2	PG0496	Departed	-03:08:00
3	PG0304	Departed	-03:04:00
4	PG0164	Departed	-00:07:00
5	PG0317	Departed	-00:07:00
6	PG0590	Departed	-00:06:00
7	PG0231	Departed	-00:05:00
8	PG0203	Departed	-00:05:00
9	PG0456	Departed	-00:05:00
10	PG0478	Departed	-00:05:00

Информация о рейсах содержится в таблице flights, чтобы вывести 10 рейсов с максимальной задержкой нам понадобится номер рейса и его статус, нас интересуют рейсы со статусом Departed, потому что нам нужно найти время на которое был задержан вылет рейса, это разница между запланированным временем вылета (scheduled_departure) и фактическим временем вылета (actual_departure) данная разница может возникнуть только у рейса который уже вылетел, то есть имеет статус Departed.

```
select
flight_no,
status,
scheduled_departure-actual_departure as delay
from flights
```

```
where status='Departed'
```

```
where status='Departed'  
order by delay asc  
limit 10;
```

```
select * from bookings;
select * from tickets;
select * from boarding_passes;

select
bookings as bookings_info,
ticket_no as num_ticket
from bookings
inner join tickets using(book_ref)
right outer join boarding_passes using(ticket_no)
where book_ref is NULL;
```

boarding_passes

select bookings as bookings_info, ticket

bookings_info	num_ticket
book_ref	book_date
total_amount	

```
select
bookings as bookings_info,
ticket_no as num_ticket
from bookings
```

```
inner join tickets using(book_ref)
```

Далее, чтобы узнать у какого бронирования есть уже посадочный талон, мы связываем (`join`'им) таблицы Билеты и Посадочные талоны применяя оператор **right outer join**:


```
right outer join boarding_passes using(ticket_no)
```

Если у Посадочного талона будет отсутствовать информация по бронированию, то столбцы отвечающие за информацию по бронированию примут значение NULL, соответственно чтобы узнать у какого посадочного талона нет бронирования применяем ограничение:

```
where book_ref is NULL;
```

Так как наш запрос не выдали никакого результата, мы делаем вывод что брони, по которым не были получены посадочные талоны, отсутствуют.

Ответ: брони, по которым не были получены посадочные талоны, отсутствуют.

9. Найдите свободные места для каждого рейса, их % отношение к общему количеству мест в самолете.

Добавьте столбец с накопительным итогом - суммарное количество вывезенных пассажиров из аэропорта за день. Т.е. в этом столбце должна отражаться сумма - сколько человек уже вылетело из данного аэропорта на этом или более ранних рейсах за сегодняшний день.

```
select
model as title_plane,
count(seat_no) as count_seat
from aircrafts a
inner join seats using (aircraft_code)
group by model;
```

	title_plane	count_seat
1	Bombardier CRJ-200	50
2	Airbus A319-100	116
3	Sukhoi SuperJet-100	97
4	Boeing 767-300	222
5	Airbus A320-200	140
6	Cessna 208 Caravan	12
7	Boeing 737-300	130
8	Airbus A321-200	170

```
select
air.airport_name as title_airport,
flight_no as num_flight,
total_seat.count_seat-count(bp.seat_no) as free_seat,
round(((total_seat.count_seat::numeric-count(bp.seat_no)::numeric)/total_seat.count_seat::numeric*100, 2) as percent_free_seat,
sum(count(bp.seat_no)) over (partition by air.airport_name) as total_depart_passenger
from flights f
inner join (select aircraft_code,
model as title_plane,
count(seat_no) as count_seat
from aircrafts a
inner join seats using (aircraft_code)
group by aircraft_code, model) as total_seat on f.aircraft_code = total_seat.aircraft_code
left join boarding_passes bp using (flight_id)
inner join airports air on f.departure_airport = airport_code
group by flight_id, total_seat.count_seat, airport_name;
```

	title_airport	num_flight	free_seat	percent_free_seat	total_depart_passenger
1	Абакан	PG0520	47	40,52	941
2	Абакан	PG0520	116	100	941
3	Абакан	PG0520	116	100	941
4	Абакан	PG0520	38	32,76	941
5	Абакан	PG0520	110	94,83	941
6	Абакан	PG0520	41	35,34	941
7	Абакан	PG0520	116	100	941
8	Абакан	PG0520	79	68,1	941

В начале создаем запрос, который позволит нам узнать количество мест в каждой модели самолета, для этого мы используем **join** для объединения таблиц aircrafts и seats, **join** производим через идентификатор aircraft_code. Через оператор **select** выводим название модели самолета, а также применяем оператор **count** для подсчета количества мест (seat_no), завершаем запрос оператором группировки **group by**:

```
select
model as title_plane,
count(seat_no) as count_seat
from aircrafts a
inner join seats using (aircraft_code)
group by model;
```

В итоговой таблице, мы будем выводить таблицу со следующими столбцами: название аэропорта, наименование рейса, количество свободных мест, процентное отношение свободных мест к общему числу мест в самолете, количество вылетевших пассажиров.

Через оператор **select** мы выводим значение airport_name из таблицы airports, flight_no из таблицы flights:

```
select
air.airport_name as title_airport,
flight_no as num_flight
```

Созданный ранее запрос на определение общего количества мест в каждой модели самолете используем в качестве нашего подзапроса в основном запросе и присоединяем путем использования оператора **join** (через идентификатор aircraft_code) после оператора **from**, который мы применили к таблице flights:

```
from flights f
inner join (select aircraft_code,
model as title_plane,
count(seat_no) as count_seat
from aircrafts a
inner join seats using (aircraft_code)
group by aircraft_code, model) as total_seat on f.aircraft_code = total_seat.aircraft_code
```

Далее выводим через оператор **select** количество свободных мест на рейсе, это разница между общим количеством мест в самолете (данные нашего подзапроса total_seat.count_seat) и количеством занятых мест, которые мы определяем через места на которые выписаны посадочные талоны из данных таблицы boarding_passes (применяем оператор count на столбец bp.seat_no):

```
total_seat.count_seat - count(bp.seat_no) as free_seat,
```

Далее находим процентное отношение количества свободных мест на рейсе (total_seat.count_seat::**numeric** - count(bp.seat_no)::**numeric**) к общему количеству мест (total_seat.count_seat::**numeric**) умноженное на 100%, во избежание ошибки вызванной типом данных, приводим полученные значения к типу данных **numeric** (чисел), для округления полученных значений применяем оператор **round**:

```
round((total_seat.count_seat::numeric - count(bp.seat_no)::numeric)/total_seat.count_seat::numeric*100, 2) as percent_free_seat,
```

Для вывода названия аэропорта применяем оператор **join** к таблице airports, в качестве идентификаторов выступают departure_airport и airport_code:

```
inner join airports air on f.departure_airport = airport_code
```

Для вывода количества всех вылетевших пассажиров применяем оконную функцию, к окну мы будем применять функцию **sum**, в качестве окна будет выступать количество занятых мест (count(bp.seat_no), поле **partition by** применим к названию аэропорта (air.airport_name), так мы узнаем сколько пассажиров вылетело в данном аэропорте текущим и предыдущими рейсами:

```
sum(count(bp.seat_no)) over (partition by air.airport_name) as total_depart_passenger
```

Завершаем запрос применением оператора **group by**:

```
group by flight_id, total_seat.count_seat, airport_name;
```

10. Найдите процентное соотношение перелетов по типам самолетов от общего количества.

```
select
model as title_plane,
round(count(f.flight_id)::numeric/(select count(flight_id)::numeric from flights)*100, 2) as percent_flight
from flights f
inner join aircrafts air using(aircraft_code)
group by model;
```

aircrafts

select model as title_plane, round(count(f.flight_id)::numeric/(select count(flight_id)::numeric from flights)*100, 2) as percent_flight

	title_plane	percent_flight
1	Bombardier CRJ-200	27,32
2	Airbus A319-100	3,74
3	Sukhoi SuperJet-100	25,68
4	Boeing 767-300	3,69
5	Cessna 208 Caravan	28
6	Boeing 737-300	3,85
7	Airbus A321-200	5,89
8	Boeing 777-300	1,84

Для формирования данного запроса нам потребуются данные двух таблиц `flights`, из которой мы находим количество перелетов с группировкой по модели самолета (`count(f.flight_id)`) и в качестве подзапроса мы воспользуемся общим количеством перелетов (`select count(flight_id)`), для округления используем оператор **round** с двумя значениями после запятой, данный показатель определим как количество рейсов выполненное каждой моделью самолета деленное на общее количество рейсов выполненных всем самолетам, итог умножим на 100%, во избежание ошибки вызванной типом данных, приводим полученные значения к типу данных **numeric** (чисел).

Через оператор **select** выведем название модели самолета и интересующее нас значение:

```
select
model as title_plane,
round(count(f.flight_id)::numeric/(select count(flight_id)::numeric from flights)*100, 2) as percent_flight
from flights f
```

Для вывода название модели самолета применим оператор **join** и добавим к нашему запросу данные из таблицы `aircrafts`, используя идентификатор `aircraft_code`:

```
inner join aircrafts air using(aircraft_code)
```

Для вывода данных по каждой модели самолета применяем оператор **group by** и завершаем запрос:

```
group by model;
```

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

```
with cte_business_flight (flight_id, class_flight, amount) as (
select
flight_id as id_flight,
fare_conditions as class_flight,
min(amount) as amount
from ticket_flights tf
where fare_conditions='Business'
group by flight_id, fare_conditions),
cte_economy_flight (flight_id, class_flight, amount) as (
select
flight_id as id_flight,
fare_conditions as class_flight,
max(amount) as amount
from ticket_flights tf
where fare_conditions='Economy'
group by flight_id, fare_conditions)
select
cbf.flight_id
from cte_business_flight cbf
join cte_economy_flight cef using (flight_id)
where cbf.amount<cef.amount;
```

Для ответа на условие создадим два запроса к таблице ticket_flights и выведем все рейсы с самым дешевым Бизнес-классом перелета и самым дорогим эконом-классом перелета:

```
select
flight_id as id_flight,
fare_conditions as class_flight,
amount as amount
from ticket_flights tf
where fare_conditions='Business';
```

	id_flight	class_flight	amount
1	30 625	Business	42 100
2	30 625	Business	42 100
3	30 625	Business	42 100
4	30 625	Business	42 100
5	30 625	Business	42 100
6	24 836	Business	9 800
7	24 836	Business	9 800

```
select
flight_id as id_flight,
fare_conditions as class_flight,
amount as amount1
from ticket_flights tf
where fare_conditions='Economy';
```

	id_flight	class_flight	amount1
1	30 625	Economy	14 000
2	30 625	Economy	14 000
3	30 625	Economy	14 000
4	30 625	Economy	14 000
5	30 625	Economy	14 000
6	30 625	Economy	14 000
7	30 625	Economy	14 000

Обернем оба запроса в СТЕ в основном запросе и применим к ним оператор **join** используя идентификатор flight_id, после выведем через оператор **select** информацию по интересующим рейсам, для чего зададим ограничение операторов **where** на вывод рейсов где перелет бизнес-классом был дороже перелета эконом классом:

```
select
cbf.flight_id
from cte_business_flight cbf
join cte_economy_flight cef using (flight_id)
where cbf.amount<cef.amount;
```

12. Между какими городами нет прямых рейсов?

```
select * from airports;
select * from flights f;

select air1.airport_code,
air2.airport_code
from airports air1 cross join airports air2
where air1.airport_code <> air2.airport_code
except
select distinct f.departure_airport, f.arrival_airport from flights f);

create view cte_airport_no_flight as (select air1.airport_code as depart,
air2.airport_code as arrival
from airports air1 cross join airports air2
where air1.airport_code <> air2.airport_code
except
select distinct f.departure_airport, f.arrival_airport from flights f);

select air1.city , air2.city from cte_airport_no_flight canf
left join airports air1 on canf.depart = air1.airport_code
left join airports air2 on canf.arrival = air2.airport_code;
```

airports

select air1.city , air2.city from cte_airport_no_flight canf

	city	city
1	Брянск	Южно-Сахалинск
2	Орск	Магадан
3	Усть-Илимск	Нефтеюганск
4	Чебоксары	Ульяновск
5	Кызыл	Комсомольск-на-Амуре
6	Ульяновск	Краснодар
7	Салехард	Нальчик
8	Ульяновск	Ухта

Для данного запроса нам потребуются данные двух таблиц: airports (airport_code, city) и flights (departure_airport, arrival_airport).

Используя декартово произведение найдем уникальные значения все возможных перелетов для этого мы применим оператор cross join к таблице airports (то есть самой к себе):

```
select air1.airport_code,
air2.airport_code
from airports air1 cross join airports air2
```

Зададим ограничение на вывод неравных значений через оператор where:

```
where air1.airport_code <> air2.airport_code
```

Используя данные таблицы flights выведем значения departure_airport, arrival_airport через select используя оператор except, так мы возвратим все строки, которые есть в результате первого запроса, но отсутствуют в результате второго запроса, так как нам интересны только уникальные значения, то добавим оператор distinct:

```
except
select distinct f.departure_airport, f.arrival_airport from flights f);
```

Создадим представление используя наш запрос:

```
create view cte_airport_no_flight as (select air1.airport_code as depart,
air2.airport_code as arrival
from airports air1 cross join airports air2
where air1.airport_code <> air2.airport_code
except
select distinct f.departure_airport, f.arrival_airport from flights f);
```

Выведем название городов между, которыми нет прямых рейсов, так мы обратимся к созданному представлению:

```
select air1.city , air2.city from cte_airport_no_flight canf
```


Для вывода названий город применим дважды оператор left join к нашему представлению, используя идентификаторы города вылета и города прилета:

```
left join airports air1 on canf.depart = air1.airport_code
left join airports air2 on canf.arrival = air2.airport_code;
```

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

--Итоговое решение

```
select distinct air1.airport_name, air2.airport_name, a.model,
round(acos(sind(air1.latitude)*sind(air2.latitude) + cosd(air1.longitude)*cosd(air2.longitude))::numeric*6371) as distance,
a.range
from flights f
left join airports air1 on f.departure_airport = air1.airport_code
left join airports air2 on f.arrival_airport = air2.airport_code
left join aircrafts a using (aircraft_code);
```

airports(+)

select distinct air1.airport_name, air2.airport_name, a.model, distance, range

	airport_name	airport_name	model	distance	range
1	Абакан	Богашёво	Cessna 208 Caravan	491	1 200
2	Абакан	Грозный	Boeing 737-300	3 484	4 200
3	Абакан	Домодедово	Airbus A319-100	3 366	6 700
4	Абакан	Кызыл	Cessna 208 Caravan	307	1 200
5	Абакан	Талаги	Airbus A319-100	3 042	6 700
6	Абакан	Толмачёво	Cessna 208 Caravan	583	1 200
7	Анадырь	Внуково	Airbus A319-100	6 220	6 700
8	Анадырь	Домодедово	Airbus A319-100	6 226	6 700

Для выполнения данного запроса нам потребуются данные таблиц: flights, airports, aircrafts.

Выведем следующие данные через оператор select название города вылета, города прилета, модель самолета, расстояние между городами, и максимальное расстояние, на котором может использоваться данная модель самолета (air1.airport_name, air2.airport_name, a.model, distance, a.range),

Для определения расстояния между города и данных о долготе и широте таблицы Аэропортов, воспользуемся известной нам формулой:

Кратчайшее расстояние между двумя точками А и В на земной поверхности (если принять ее за сферу) определяется зависимостью:

$d = \arccos \{ \sin(\text{latitude}_a) \cdot \sin(\text{latitude}_b) + \cos(\text{latitude}_a) \cdot \cos(\text{latitude}_b) \cdot \cos(\text{longitude}_a - \text{longitude}_b) \}$, где latitude_a и latitude_b — широты, longitude_a, longitude_b — долготы данных пунктов, d — расстояние между пунктами, измеряемое в радианах длиной дуги большого круга земного шара.

Расстояние между пунктами, измеряемое в километрах, определяется по формуле:

$L = d \cdot R$, где $R = 6371$ км — средний радиус земного шара.

В результате получим следующий запрос:

```
select distinct air1.airport_name, air2.airport_name, a.model,
round(acos(sind(air1.latitude)*sind(air2.latitude) + cosd(air1.longitude)*cosd(air2.longitude))::numeric*6371) as distance,
a.range
from flights f
```

Для вывода названия модели самолета и городов, применим оператор left join к нашему основному запросу по следующим идентификаторам f.departure_airport = air1.airport_code, f.arrival_airport = air2.airport_code, aircraft_code:

```
left join airports air1 on f.departure_airport = air1.airport_code
left join airports air2 on f.arrival_airport = air2.airport_code
left join aircrafts a using (aircraft_code);
```

Скрипт выполнения итоговой работы

--Задание 1

```
select * from airports;

select
city as title_city,
count(airport_code) as count_airports
from airports a
group by city
having count(airport_code)>'1';
```

--Задание 2

```
select * from aircrafts;
select * from flights;
select * from airports;

select
model as title_model
from aircrafts a
where "range" = (select max(range) from aircrafts a2);

select
airport_name as title_airport
from airports air
inner join flights f on departure_airport =airport_code
inner join aircrafts using (aircraft_code)
where "range" = (select max(range) from aircrafts a2)
group by title_airport;
```

--Задание 3

```
select * from flights;

select
flight_no,
status,
scheduled_departure-actual_departure as delay
from flights
where status='Departed'
order by delay asc
limit 10;
```

--Задание 4

```
select * from bookings;
select * from tickets;
select * from boarding_passes;

select
bookings as bookings_info,
ticket_no as num_ticket
from bookings
inner join tickets using(book_ref)
right outer join boarding_passes using(ticket_no)
where book_ref is NULL;
```

--Задание 5

```
select * from airports a;
select * from aircrafts;
select * from flights;
select * from seats;
select * from boarding_passes bp;

select
model as title_plane,
count(seat_no) as count_seat
from aircrafts a
inner join seats using (aircraft_code)
group by model;

select b.airport_name,
a.flight_id,
a.total_seat-b.occupied_seat as free_seat,
b.occupied_seat,
round((total_seat::numeric-occupied_seat::numeric)/total_seat::numeric*100, 2) as percent_free_seat,
sum(b.occupied_seat) over (partition by b.airport_name) as total_depart_passenger
from (select air.airport_name,
f.flight_id,
count(bp.seat_no) as occupied_seat
from bookings.flights f
left join bookings.boarding_passes bp
on f.flight_id=bp.flight_id
join airports air on f.departure_airport = air.airport_code
group by f.flight_id, air.airport_name) as b
left join
(select f.flight_id, count(s.seat_no) as total_seat
from bookings.flights as f
left join bookings.seats s
on s.aircraft_code=f.aircraft_code
group by f.flight_id) as a
on b.flight_id= a.flight_id;

select
air.airport_name as title_airport,
flight_id as id_flight,
flight_no as num_flight,
f.aircraft_code as code_aircraft,
total_seat.count_seat-count(bp.seat_no) as free_seat,
count(bp.seat_no) as occupied_seat,
total_seat.count_seat as total_seat,
round((total_seat.count_seat::numeric-count(bp.seat_no)::numeric)/total_seat.count_seat::numeric*100, 2) as percent_free_seat,
sum(count(bp.seat_no)) over (partition by air.airport_name) as total_depart_passenger
from flights f
inner join (select aircraft_code,
model as title_plane,
count(seat_no) as count_seat
from aircrafts a
inner join seats using (aircraft_code)
group by aircraft_code, model) as total_seat on f.aircraft_code = total_seat.aircraft_code
left join boarding_passes bp using (flight_id)
inner join airports air on f.departure_airport = airport_code
group by flight_id, total_seat.count_seat, airport_name;
```

--Итоговое решение

```
select
air.airport_name as title_airport,
f.flight_no as num_flight,
total_seat.count_seat-count(bp.seat_no) as free_seat,
round((total_seat.count_seat::numeric-count(bp.seat_no)::numeric)/total_seat.count_seat::numeric*100, 2) as percent_free_seat,
sum(count(bp.seat_no)) over (partition by air.airport_name) as total_depart_passenger
from flights f
inner join (select aircraft_code,
model as title_plane,
count(seat_no) as count_seat
from aircrafts a
inner join seats using (aircraft_code)
group by aircraft_code, model) as total_seat on f.aircraft_code = total_seat.aircraft_code
left join boarding_passes bp using (flight_id)
inner join airports air on f.departure_airport = airport_code
group by flight_id, total_seat.count_seat, airport_name;
```

--Задание 6

```
select * from flights f;
select * from aircrafts a;

select
model as title_plane,
count(flight_id)
from flights f
inner join aircrafts air using(aircraft_code)
group by model;

select count(flight_id) from flights f;

select
model as title_plane,
round(count(f.flight_id)::numeric/(select count(flight_id)::numeric from flights)*100, 2) as percent_flight
from flights f
inner join aircrafts air using(aircraft_code)
group by model;
```

--Задание 7

```
select * from ticket_flights tf;

select
flight_id as id_flight,
fare_conditions as class_flight,
amount as amount
from ticket_flights tf
where fare_conditions='Business';

select
flight_id as id_flight,
fare_conditions as class_flight,
amount as amount1
from ticket_flights tf
where fare_conditions='Economy';

with cte_business_flight (flight_id, class_flight, amount) as (
select
flight_id as id_flight,
fare_conditions as class_flight,
min(amount) as amount
from ticket_flights tf
where fare_conditions='Business'
group by flight_id, fare_conditions),
cte_economy_flight (flight_id, class_flight, amount) as (
select
flight_id as id_flight,
fare_conditions as class_flight,
max(amount) as amount
from ticket_flights tf
where fare_conditions='Economy'
group by flight_id, fare_conditions)
select
cbf.flight_id
from cte_business_flight cbf
join cte_economy_flight cef using (flight_id)
where cbf.amount < cef.amount;
```

-- Задание 8

```
select * from airports;
select * from flights f;

select air1.airport_code,
air2.airport_code
from airports air1 cross join airports air2
where air1.airport_code <> air2.airport_code
except
select distinct f.departure_airport, f.arrival_airport from flights f;

create view cte_airport_no_flight as (select air1.airport_code as depart,
air2.airport_code as arrival
from airports air1 cross join airports air2
where air1.airport_code <> air2.airport_code
except
select distinct f.departure_airport, f.arrival_airport from flights f);

select air1.city , air2.city from cte_airport_no_flight canf
left join airports air1 on canf.depart = air1.airport_code
left join airports air2 on canf.arrival = air2.airport_code;
```

-- Задание 9

$d = \arccos \{ \sin(\text{latitude}_a) \cdot \sin(\text{latitude}_b) + \cos(\text{latitude}_a) \cdot \cos(\text{latitude}_b) \cdot \cos(\text{longitude}_a - \text{longitude}_b) \}$

```
select * from airports;
select * from aircrafts;
select * from flights f;
```

```
select distinct f.departure_airport, f.arrival_airport, air1.longitude, air2.longitude, air1.latitude, air2.latitude,
acos(sin(air1.latitude)*sin(air2.latitude)+cos(air1.latitude)*cos(air2.latitude)*cos(air1.longitude - air2.longitude))*6371 as distance
from flights f
left join airports air1 on f.departure_airport = air1.airport_code
left join airports air2 on f.arrival_airport = air2.airport_code;
```

```
select distinct f.departure_airport, f.arrival_airport, air1.airport_name, air2.airport_name,
round(acos(sind(air1.latitude)*sind(air2.latitude) + cosd(air1.latitude)*cosd(air2.latitude)*cosd(air1.longitude -
air2.longitude))::numeric*6371, 2) as distance,
a.range
from flights f
left join airports air1 on f.departure_airport = air1.airport_code
left join airports air2 on f.arrival_airport = air2.airport_code
left join aircrafts a using (aircraft_code);
```

--Итоговое решение

```
select distinct air1.airport_name, air2.airport_name, a.model,
round(acos(sind(air1.latitude)*sind(air2.latitude) + cosd(air1.latitude)*cosd(air2.latitude)*cosd(air1.longitude -
air2.longitude))::numeric*6371) as distance,
a.range
from flights f
left join airports air1 on f.departure_airport = air1.airport_code
left join airports air2 on f.arrival_airport = air2.airport_code
left join aircrafts a using (aircraft_code);
```