

# Итоговая работа по модулю «SQL и получение данных»

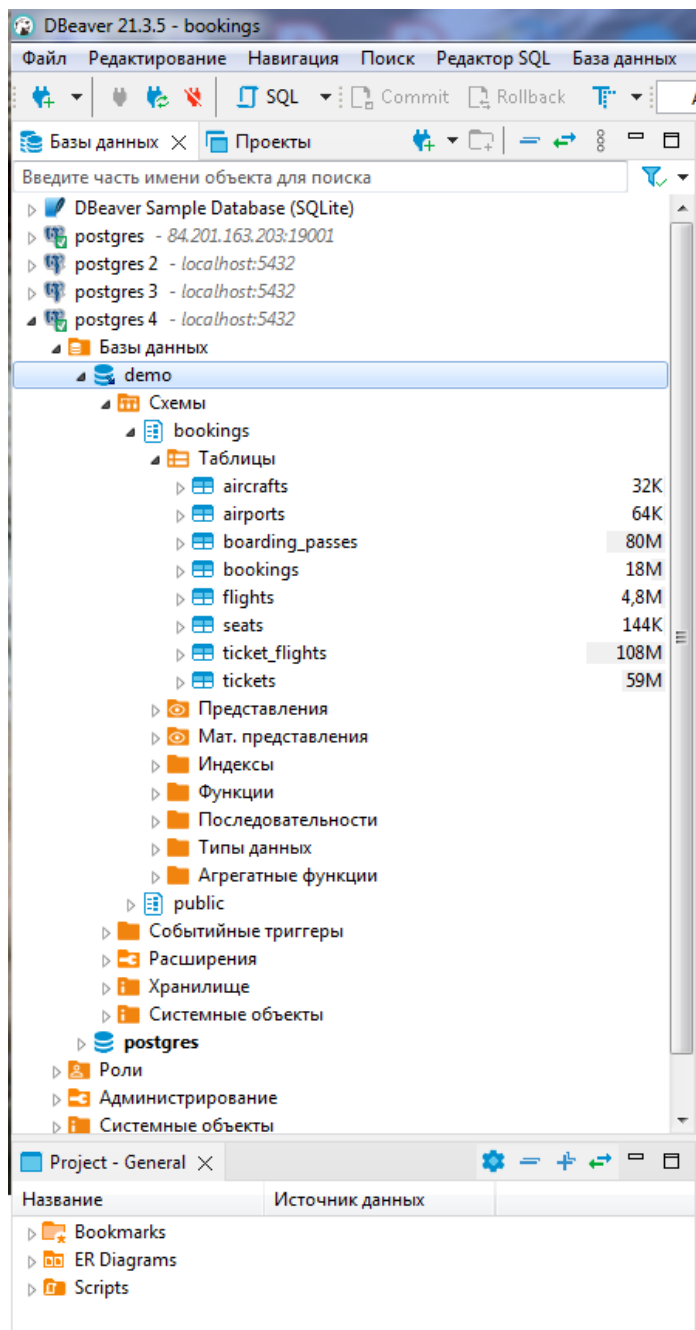
Студентки курса

“Системный аналитик SAL-10”

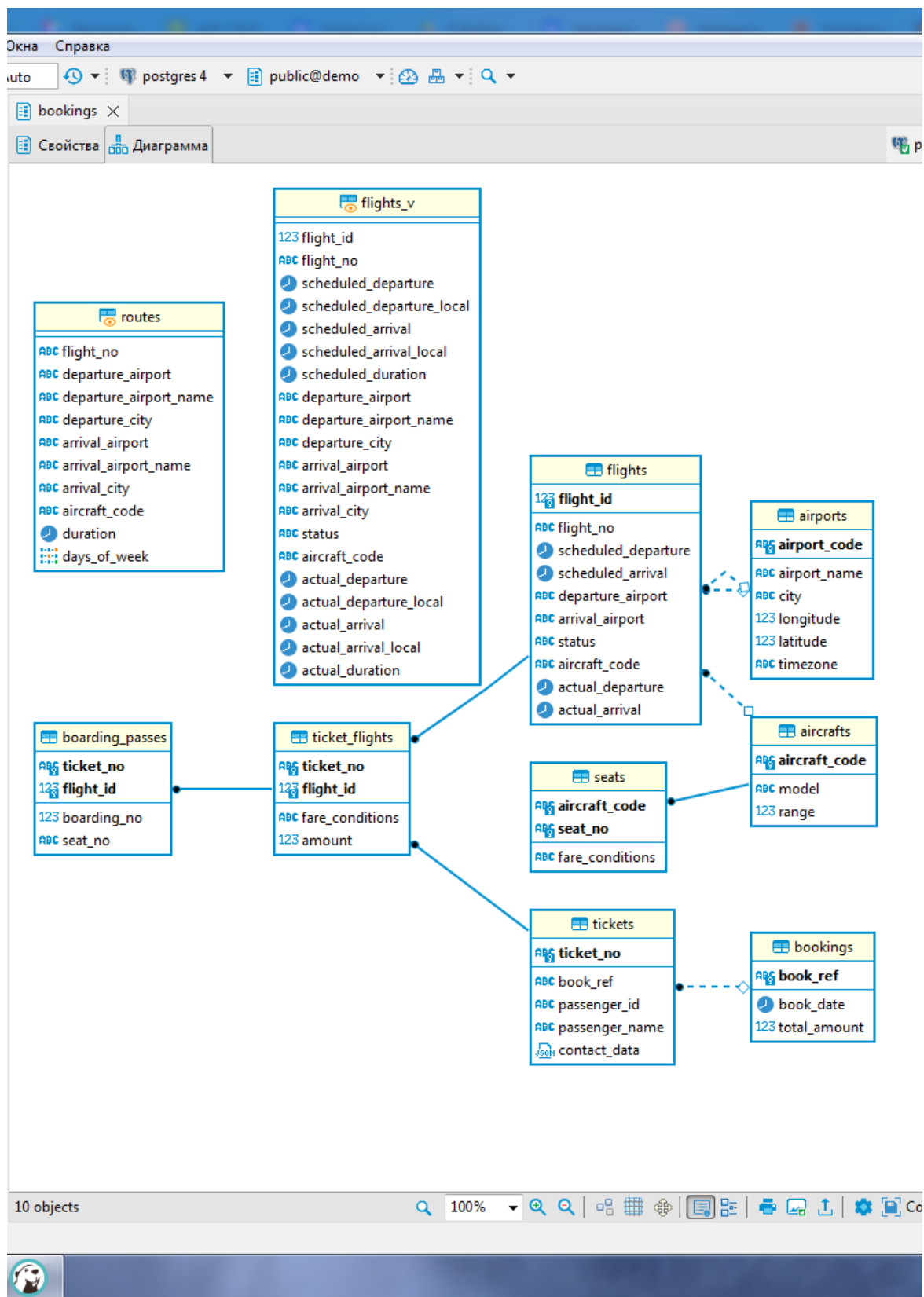
Кунгуровой Анастасии

# Итоговая работа

1. В работе использовался локальный тип подключения.



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



### 3. Краткое описание БД.

#### Таблицы БД:

- **Самолеты (aircrafts):** содержит информацию о коде самолета, модели самолетов и максимальной дальности полета этих самолетов.
- **Аэропорты (airports):** содержит информацию о коде аэропорта, названии аэропорта, городе, координатах аэропорта (ширине и долготе) и о временной зоне каждого аэропорта.
- **Посадочные талоны (boarding\_passes):** содержит информацию о номере билета, идентификаторе рейса, номере посадочного талона и номере места.
- **Бронирования (bookings):** содержит информацию о номере бронирования, дате бронирования и полной сумме бронирования.
- **Рейсы (flights):** содержит информацию об идентификационном номере рейса, номере рейса, времени отправления по расписанию, времени прибытия по расписанию, аэропорте отправления, аэропорте прибытия, статусе полета, реальном времени отправления и реальном времени прибытия.
- **Места (seats):** содержит информацию о коде модели самолета, номерах мест, классе обслуживания.
- **Перелеты (ticket\_flights):** содержит информацию о номере билета, идентификаторе рейса, классе обслуживания и стоимости перелета.
- **Билеты (tickets):** содержит информацию о номерах билетов, номере бронирования, идентификаторе пассажира, имени пассажира, контактных данных пассажира.

Представление: рейсы (flights\_v) содержит информацию об идентификаторе рейса, номере рейса, времени вылета по расписанию, времени вылета по расписанию (местное время в

пункте отправления), времени прилёта по расписанию, времени прилёта по расписанию (местное время в пункте прибытия), планируемой продолжительности полета, коде аэропорта отправления, названии аэропорта отправления, городе отправления, коде аэропорта прибытия, названии аэропорта прибытия, городе прибытия, статусе рейса, коде самолета, фактическом времени вылета, фактическом времени вылета (местное время в пункте отправления), фактическом времени прилёта, фактическом времени прилёта (местное время в пункте прибытия), фактической продолжительности полета.

**Материализованное представление:** маршруты (routes) содержит информацию о номере рейса, коде аэропорта отправления, названии аэропорта отправления, городе отправления, коде аэропорта прибытия, названии аэропорта прибытия, городе прибытия, коде самолета, продолжительности полета, днях недели, в которые выполняются рейсы.

#### 4. Развернутый анализ БД.

##### **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 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).

### **Объекты схемы.**

- **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).

**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).

**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).

**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.

**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.

**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.

**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).

**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.

**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).

- **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.

- **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.

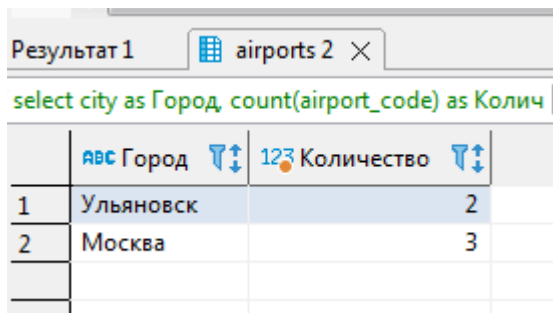
### **Бизнес задачи.**

- Анализ загруженности рейсов для принятия решения об их количестве.
- Составление более эффективного расписания рейсов.
- Анализ спроса на различные рейсы в зависимости от сезонности или школьных каникул.
- Учет влияния социальных факторов на спрос по направлениям перелетов.
- Создание бонусной системы оплаты билетов для стимуляции спроса и увеличения заполняемости рейсов.
- Создание удобной системы пересадок для городов, между которыми нет прямых перелетов.

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

1. В каких городах больше одного аэропорта?

```
select city as Город,  
       count(airport_code) as Количество  
from airports  
group by city  
having count(airport_code) > 1
```



	Город	Количество
1	Ульяновск	2
2	Москва	3

2. В каких аэропортах есть рейсы, выполняемые самолетом с максимальной дальностью перелета? (Использовать подзапрос)

```
select distinct airport_name||', '||airport_code||', '||city as Аэропорт  
from airports a  
join flights f on f.arrival_airport = a.airport_code  
or f.departure_airport = a.airport_code  
where f.aircraft_code = (  
select aircraft_code  
from aircrafts a  
where a."range" = (select max (a2."range") from aircrafts a2))
```

Таблица	ABC Аэропорт	
1	Пермь, PEE, Пермь	
2	Толмачёво, OVB, Новосибирск	
3	Кольцово, SVX, Екатеринбург	
4	Шереметьево, SVO, Москва	
5	Сочи, AER, Сочи	
6	Домодедово, DME, Москва	
7	Внуково, VKO, Москва	

3. Вывести 10 рейсов с максимальным временем задержки вылета (использовать оператор LIMIT)

```
select flight_no,
       departure_airport, arrival_airport,
       actual_departure - scheduled_departure as "Задержка вылета"
from flights f
where f.actual_departure is not null and f.scheduled_departure is not null
order by 4 desc limit 10
```

	ABC flight_no	ABC departure_airport	ABC arrival_airport	Задержка вылета
1	PG0589	PEE	SVX	04:37:00
2	PG0164	DME	NUX	04:28:00
3	PG0364	KRR	ULV	04:27:00
4	PG0568	OVS	UFA	04:20:00
5	PG0454	DME	URS	04:18:00
6	PG0096	URJ	DME	04:18:00
7	PG0166	SVX	MQF	04:16:00
8	PG0278	OVB	SVO	04:16:00
9	PG0564	UFA	UCT	04:14:00
10	PG0669	VKO	KRO	04:08:00

4. Были ли брони, по которым не были получены посадочные талоны? (Использовать верный тип JOIN)

```
select b.book_ref, b.book_date::date, bp.boarding_no
from bookings b
left join tickets t on t.book_ref = b.book_ref
left join boarding_passes bp on bp.ticket_no = t.ticket_no
where bp.boarding_no is null
```

	book_ref	book_date	123 boarding_no
1	0006C3	2016-09-30	[NULL]
2	001273	2016-09-28	[NULL]
3	0019EB	2016-10-09	[NULL]
4	00673C	2016-10-10	[NULL]
5	00EBC0	2016-10-10	[NULL]
6	00EDF4	2016-10-06	[NULL]
7	016884	2016-09-30	[NULL]
8	016B74	2016-09-28	[NULL]
9	018BDD	2016-10-11	[NULL]
10	0269CF	2016-10-10	[NULL]
11	02E245	2016-09-29	[NULL]
12	0310B2	2016-10-09	[NULL]
13	034472	2016-10-04	[NULL]
14	036A1D	2016-10-07	[NULL]
15	03A96B	2016-10-11	[NULL]
16	03AC3C	2016-10-02	[NULL]
17	042D9D	2016-09-27	[NULL]

Save Cancel Script | 200 строк получено - 903ms (10ms получ.), мар. 24, 15:10:22

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 и/или cte)

```
select f.flight_no,
       f.departure_airport,
       f.arrival_airport,
       ts.total_seats,
       ts.total_seats - fp.fact_passengers as empty_seats,
       round ((ts.total_seats::numeric - fp.fact_passengers::numeric)*100 /
ts.total_seats::numeric, 2) as percent,
       sum (fp.fact_passengers) over (partition by f.actual_departure::date,
f.departure_airport order by f.actual_departure) as cumulative
```

```

from flights f
join (select f.flight_id, count (bp.seat_no) as fact_passengers
from boarding_passes bp
join flights f on f.flight_id = bp.flight_id
group by f.flight_id) as fp
on f.flight_id = fp.flight_id
join (select s.aircraft_code, count (s.seat_no) as total_seats
from seats s
group by s.aircraft_code) as ts
on ts.aircraft_code = f.aircraft_code

```

	ABC f	ABC de	ABC arr	123 total_seats	123 empty_seats	123 percent	123 cumulative
1	PG0480	AAQ	EGO	97	94	96,91	3
2	PG0252	AAQ	SVO	130	79	60,77	54
3	PG0520	ABA	DME	116	110	94,83	6
4	PG0502	AER	KJA	116	110	94,83	6
5	PG0522	AER	KUF	130	126	96,92	10
6	PG0618	AER	EGO	12	7	58,33	15
7	PG0562	AER	VKO	222	216	97,3	21
8	PG0013	AER	SVO	402	311	77,36	112
9	PG0429	ARH	TOF	50	44	88	6
10	PG0258	ARH	NNM	50	47	94	9
11	PG0155	ARH	TJM	50	47	94	12
12	PG0338	ARH	DME	50	38	76	24
13	PG0608	ASF	DME	50	42	84	8
14	PG0129	ASF	SVO	50	43	86	15
15	PG0540	BAX	VKO	97	93	95,88	4
16	PG0115	BQS	KHV	97	89	91,75	8
17	PG0135	BTk	DME	116	107	92,24	9

200 строк получено - 562ms (10ms получ.), мар. 24, 15:13:04

6. Найдите процентное соотношение перелетов по типам самолетов от общего количества. (Использовать подзапрос или окно, оператор ROUND)

```

select model, (round(quantity / (sum(quantity) over ()), 2) * 100) as percent
from (select count(flight_id) as quantity, model
from flights f
join aircrafts a on a.aircraft_code = f.aircraft_code
group by model) fa
group by fa.model, fa.quantity

```

	ABC model	123 percent
1	Airbus A319-100	4
2	Airbus A321-200	6
3	Boeing 737-300	4
4	Boeing 767-300	4
5	Boeing 777-300	2
6	Bombardier CRJ-2	27
7	Cessna 208 Carav	28
8	Sukhoi SuperJet-1	26

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

```

with cte_economy as (
    select tf.flight_id, tf.ticket_no, tf.amount
    from ticket_flights tf
    where tf.fare_conditions = 'Economy'),
cte_business as (
    select tf.flight_id, tf.ticket_no, tf.amount
    from ticket_flights tf
    where tf.fare_conditions = 'Business')
select distinct a.city
from cte_economy
join cte_business on cte_business.flight_id = cte_economy.flight_id
and cte_economy.amount > cte_business.amount
join flights f on cte_economy.flight_id = f.flight_id
join airports a on f.arrival_airport = a.airport_code

```

	ABC city

8. Между какими городами нет прямых рейсов? (Использовать декартово произведение в предложении FROM, самостоятельно созданные представления, оператор EXCEPT)



```
create view cities_view as
select fv.departure_city, fv.arrival_city
from flights_v fv
```

```
select distinct a.city, a1.city
from airports a
cross join airports a1
where a.city != a1.city
except
select cv.departure_city, cv.arrival_city
from cities_view cv
order by 1, 2
```

	ABC city	ABC city
1	Абакан	Анадырь
2	Абакан	Анапа
3	Абакан	Астрахань
4	Абакан	Барнаул
5	Абакан	Белгород
6	Абакан	Белоярский
7	Абакан	Благовещен
8	Абакан	Братск
9	Абакан	Брянск
10	Абакан	Бугульма
11	Абакан	Владивосток
12	Абакан	Владикавказ
13	Абакан	Волгоград
14	Абакан	Воркута
15	Абакан	Воронеж
16	Абакан	Геленджик
17	Абакан	Горно-Алтай
18	Абакан	Екатеринбург
19	Абакан	Иваново
20	Абакан	Ижевск
21	Абакан	Йошкар-Ола
22	Абакан	Иркутск
23	Абакан	Казань
24	Абакан	Калининград
25	Абакан	Калуга
26	Абакан	Кемерово
27	Абакан	Киров

Save

Cancel

Script

200 строк получено - 183ms, m

9. Вычислите расстояние между аэропортами, связанными прямыми рейсами, сравните с допустимой максимальной дальностью перелетов в самолетах, обслуживающих эти рейсы. (Использовать оператор RADIANS или sind/cosd, CASE)

```
select distinct a2.city as "Город отправления",
               a.city as "Город прибытия",
               round (acos((sind(a2.latitude) * sind(a.latitude) +
cosd(a2.latitude) * cosd(a.latitude) * cosd (a2.longitude - a.longitude)))) ::
numeric * 6371, 2) as "Расстояние",
               a3."range" as "Макс_дальность",
               a3.model as "Модель",
               a3."range" - round (acos ((sind (a2.latitude) * sind
(a.latitude) + cosd (a2.latitude) * cosd (a.latitude) * cosd (a2.longitude -
a.longitude ))) :: numeric * 6371, 0) as "Разница",
               case
                 when range > round (acos (( sind (a2.latitude) * sind
(a.latitude) + cosd (a2.latitude) * cosd (a.latitude) * cosd (a2.longitude -
a.longitude)))) :: numeric * 6371, 2)
                 then 'ок'
                 else 'not ok'
               end result
from flights f
left join airports a on a.airport_code = f.arrival_airport
left join airports a2 on a2.airport_code = f.departure_airport
left join aircrafts a3 on a3.aircraft_code = f.aircraft_code
```

	АВС Город от	АВС Город при	123 Расстоя	123 Макс_дальн	АВС Модель	123 Разница	АВС result
1	Абакан	Архангельск	3 041,97	6 700	Airbus A319-100	3 658	ок
2	Абакан	Грозный	3 484,15	4 200	Boeing 737-300	716	ок
3	Абакан	Кызыл	307,01	1 200	Cessna 208 Caravan	893	ок
4	Абакан	Москва	3 366,3	6 700	Airbus A319-100	3 334	ок
5	Абакан	Новосибирск	582,7	1 200	Cessna 208 Caravan	617	ок
6	Абакан	Томск	490,53	1 200	Cessna 208 Caravan	709	ок
7	Анадырь	Москва	6 177,08	6 700	Airbus A319-100	523	ок
8	Анадырь	Москва	6 220,25	6 700	Airbus A319-100	480	ок
9	Анадырь	Москва	6 226,05	6 700	Airbus A319-100	474	ок
10	Анадырь	Хабаровск	3 074,2	6 700	Airbus A319-100	3 626	ок
11	Анапа	Белгород	629,86	3 000	Sukhoi SuperJet-100	2 370	ок
12	Анапа	Москва	1 219,88	4 200	Boeing 737-300	2 980	ок
13	Анапа	Новокузнецк	3 634,02	4 200	Boeing 737-300	566	ок
14	Архангельск	Абакан	3 041,97	6 700	Airbus A319-100	3 658	ок
15	Архангельск	Иркутск	3 778,09	6 700	Airbus A319-100	2 922	ок
16	Архангельск	Москва	1 005,09	2 700	Bombardier CRJ-200	1 695	ок
17	Архангельск	Нарьян-Мар	663,02	2 700	Bombardier CRJ-200	2 037	ок
18	Архангельск	Пермь	1 096,68	1 200	Cessna 208 Caravan	103	ок
19	Архангельск	Томск	2 552,68	2 700	Bombardier CRJ-200	147	ок

☒ Save
 ☐ Cancel
 








 Rows: 1

200 строк получено - 855ms (10ms получ.), мар. 25, 10:27:33