



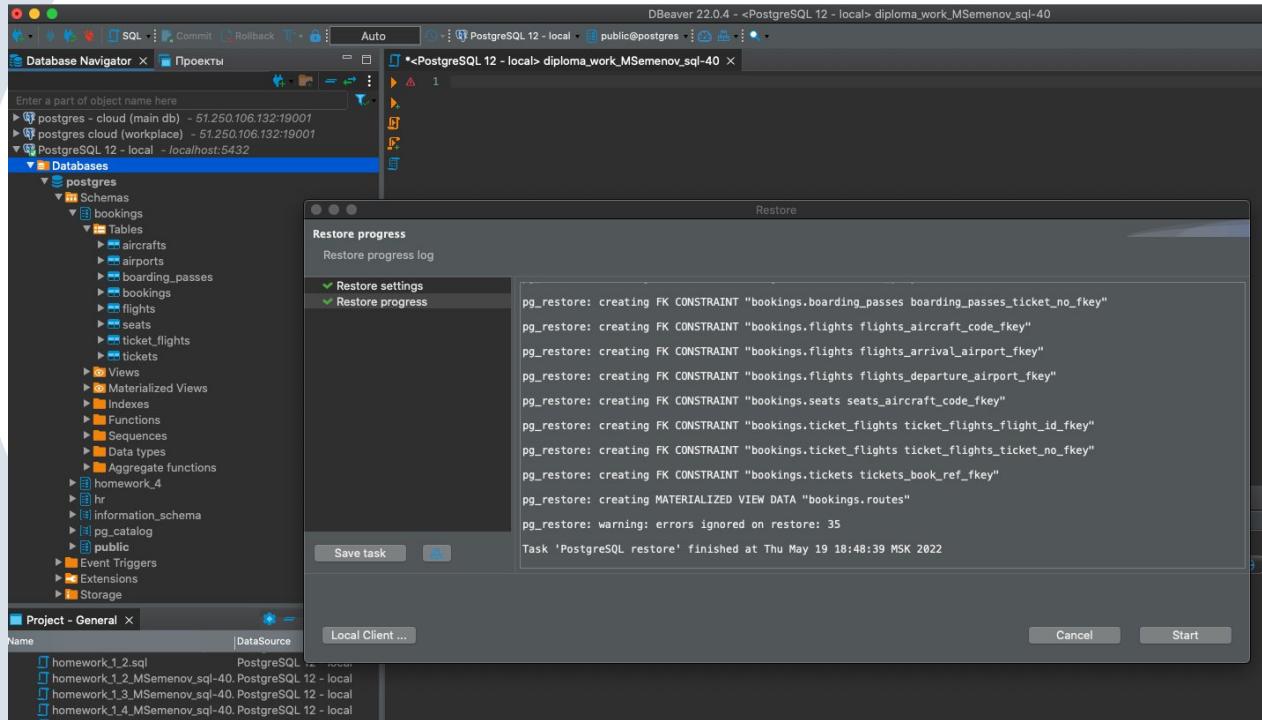
# Итоговая работа по модулю SQL и получение данных

25 мая 2022 года

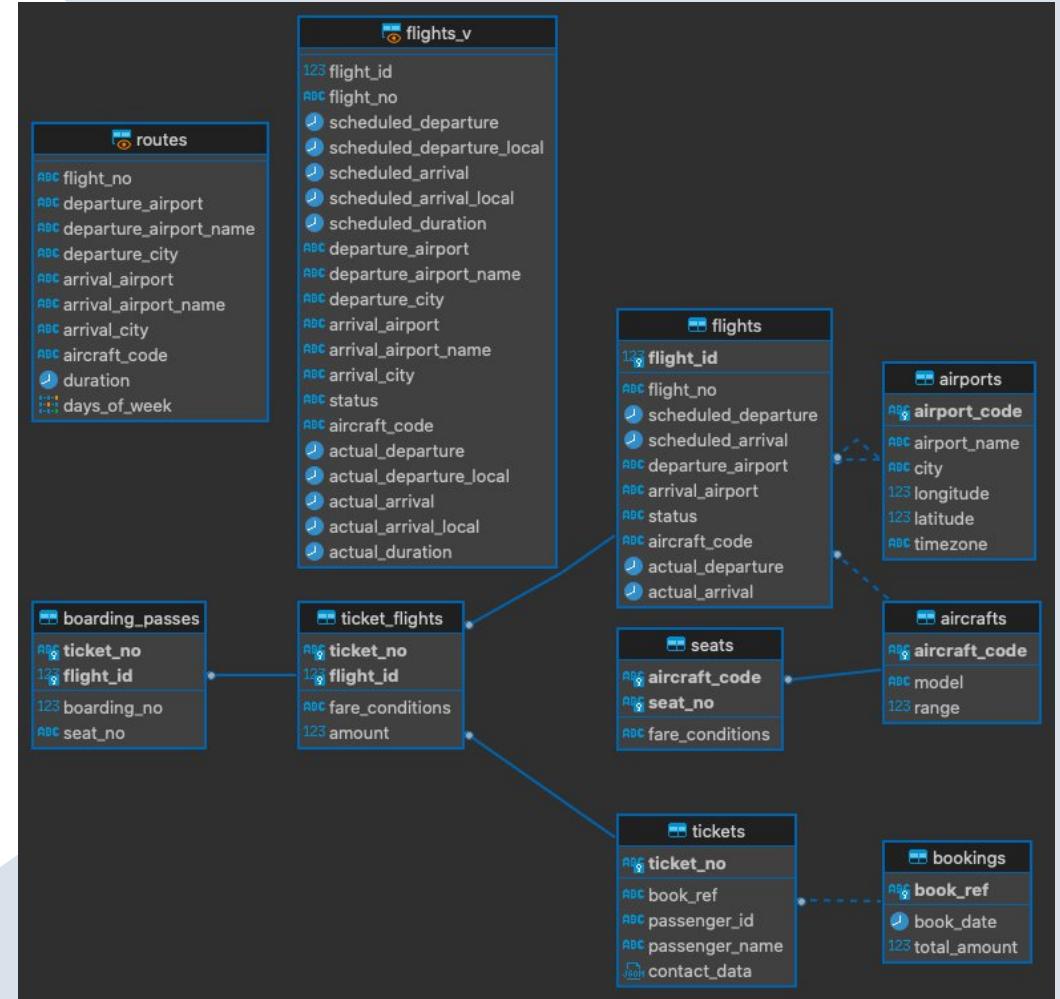
Студент: Семенов Михаил, группа SQL-40  
Преподаватель: Николай Хашанов

# 1. Подключение / 2. ER диаграмма

В работе использовался локальный тип подключения. Версия PostgreSQL 12.



База восстановлена из резервной копии «avia.backup»



ER диаграмма схемы bookings

### 3. Краткое описание БД – схема bookings

- 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).

№	Название	Тип	Размер	Описание
1	aircrafts	Таблица	32 КБ	Самолеты
2	airports	Таблица	64 КБ	Аэропорты
3	boarding_passes	Таблица	80 МБ	Посадочные талоны
4	bookings	Таблица	18 МБ	Бронирования
5	flights	Таблица	4,8 МБ	Рейсы
6	flights_v	Представление	0 КБ	Рейсы
7	routes	Мат. представление	144 КБ	Маршруты
8	seats	Таблица	136 КБ	Места
9	ticket_flights	Таблица	108 МБ	Перелеты
10	tickets	Таблица	59 МБ	Билеты

Все объекты демонстрационной базы данных находятся в схеме **bookings**. Это означает, что при обращении к объектам 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`.

# 4. Развернутый анализ БД – описание таблиц

## Таблица 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 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 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

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)

# 4. Развёрнутый анализ БД – описание таблиц / представлений

## Таблица 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)

# 4. Развёрнутый анализ БД – описание таблиц

## Таблица 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 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 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)

# 4. Развернутый анализ БД – описание представлений/ф-ий

## Представление 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 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.

Актуальная версия на текущий момент — от 13.10.2016.

## 4. Развернутый анализ БД – бизнес задачи



- ✓ Оценка эффективности маршрутов - оптимизация / создание новых / закрытие неэффективных
- ✓ Корректный подбор самолетов на основе дальности, заполняемости и популярности перелетов
- ✓ Планирование маршрутов, продажа билетов
- ✓ Оценка экономической эффективности

## 5. Список SQL запросов из Приложения №2

Список запросов с описанием логики прилагается к итоговой работе в файле

«**diploma\_work\_MSemenov\_sql-40.sql**» ,

а также на следующих слайдах



## 5. Список SQL запросов из Приложения №2 – вопрос 1

«В каких городах больше одного аэропорта?»

```
select city as «Город», count (city) as «Количество аэропортов»  
from airports  
group by city —Группируем по полю city  
having count(city) >1 —выбираем значения где счетчик городов > 1
```

	Город	Количество аэропортов
1	Ульяновск	2
2	Москва	3

## 5. Список SQL запросов из Приложения №2 – вопрос 2

«В каких аэропортах есть рейсы, выполняемые самолетом с максимальной дальностью перелета?»

—оставляем только уникальные значения в `departure_airport`. Не делаем доп манипуляций с полем `arrival_airport` т.к. если из него вылетает самолет, то он будет и в колонке `departure_airport`

```
select distinct f.departure_airport as «Аэропорты с самолетами макс. дальн»  
from flights f
```

—джойним по коду самолета `t.aircraft_code = f.aircraft_code`. Используем внутреннее соединение - в данном случае оно хорошо подходит.

`join`(

—подзапросом сортируем таблицу по полю дальность перелета от большего к меньшему и оставляем только 1 максимальное значение. Это и есть наш самолет.

```
select aircraft_code, model, «range»  
from aircrafts  
order by «range» desc  
limit 1) t on t.aircraft_code = f.aircraft_code
```

Аэропорты с самолетами макс. дальн	
1	SVX
2	DME
3	PEE
4	VKO
5	OVB
6	SVO
7	AER

## 5. Список SQL запросов из Приложения №2 – вопрос 3

«Вывести 10 рейсов с максимальным временем задержки вылета»

```
select flight_no as «Рейс»,  
scheduled_departure «План врем отправления» ,  
actual_departure as «Фактич время отправления»,  
actual_departure - scheduled_departure as «Задержка вылета»  
from flights  
where actual_departure is not null  
—сортируем, чтобы вывести максимальное время задержки  
вылета  
order by actual_departure - scheduled_departure desc  
—ограничиваем 10 строками выдачу  
limit 10
```

	№ Рейс	План врем отправления	Фактич время отправления	Задержка вылета
1	PG0589	2016-09-26 14:30:00.000 +0300	2016-09-26 19:07:00.000 +0300	04:37:00
2	PG0164	2016-09-26 14:25:00.000 +0300	2016-09-26 18:53:00.000 +0300	04:28:00
3	PG0364	2016-09-16 10:45:00.000 +0300	2016-09-16 15:12:00.000 +0300	04:27:00
4	PG0568	2016-10-11 15:15:00.000 +0300	2016-10-11 19:35:00.000 +0300	04:20:00
5	PG0454	2016-09-30 09:05:00.000 +0300	2016-09-30 13:23:00.000 +0300	04:18:00
6	PG0096	2016-10-01 15:35:00.000 +0300	2016-10-01 19:53:00.000 +0300	04:18:00
7	PG0166	2016-10-10 13:35:00.000 +0300	2016-10-10 17:51:00.000 +0300	04:16:00
8	PG0278	2016-09-13 13:20:00.000 +0300	2016-09-13 17:36:00.000 +0300	04:16:00
9	PG0564	2016-10-08 08:30:00.000 +0300	2016-10-08 12:44:00.000 +0300	04:14:00
10	PG0669	2016-09-16 15:15:00.000 +0300	2016-09-16 19:23:00.000 +0300	04:08:00

## 5. Список SQL запросов из Приложения №2 – вопрос 4

«Были ли брони, по которым не были получены посадочные талоны?»

```
select count(b.book_ref) as «Кол-во броней без пос.талона»  
from bookings b
```

—чтобы добраться до посадочных талонов подключаем таблицу с билетами по номеру бронирования. В одном бронировании может быть несколько билетов. Кол-во строк увеличивается.

```
join tickets t on t.book_ref = b.book_ref
```

— правильно выбираем джойн - в нашем случае левый, по номеру билета. Если не будет совпадения значений, к таким бронированиям будет добавлено NULL

```
left join boarding_passes bp on bp.ticket_no = t.ticket_no
```

—выбираем только NULL значения

```
where bp.boarding_no is null
```

Кол-во броней без пос.талона	
1	127,899

## 5. Список SQL запросов из Приложения №2 – вопрос 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.»

—Считаем количество мест в каждой модели самолета

```
with ct1 as (
select aircraft_code , count(seat_no) as ts
from seats s
group by aircraft_code
),
```

—Находим сколько мест занято для каждого рейса на каждый перелет flight\_id и формируем накопительный итог

```
ct2as (
select t.flight_id, t.flight_no, t.aircraft_code, t.actual_departure::date, t.departure_airport ,
—количество занятых мест на каждом рейсе
t.count as os,
—формируем накопительный итог по каждому аэропорту на каждый день
sum(t.count) over (partition by t.departure_airport, t.actual_departure::date order by t.actual_departure) as sum_count
from (
```

—в подзапросе считаем количество человек на каждом рейсе

```
select f.flight_id, f.flight_no, f.aircraft_code, f.actual_departure, f.departure_airport , count(bp.ticket_no)
from flights f
join boarding_passes bp on bp.flight_id = f.flight_id
where f.actual_departure is not null
group by f.flight_id
)t
)
```

select ct2.flight\_no as «№ рейса»,

ct1.ts - ct2.os as «Свободные места», —Вычитаем из общего кол-ва мест в самолете на рейсе - количество занятых мест

—ct2.os as «Занято мест»,

round((ct1.ts - ct2.os)/ct1.ts::numeric(5,2)\*100.0,0) as «% свободн мест»,

ct2.sum\_count as «Накопление»,

ct2.departure\_airport as «Аэропорт»,

ct2.actual\_departure as «Дата рейса»

from ct2

join ct1 on ct1.aircraft\_code = ct2.aircraft\_code

	№ рейса	Свободные места	% свободн мест	Накопление	Аэропорт	Дата рейса
1	PG0480	94	97	3	AAQ	2016-09-13
2	PG0252	79	61	54	AAQ	2016-09-13
3	PG0480	94	97	3	AAQ	2016-09-14
4	PG0252	80	62	53	AAQ	2016-09-14
5	PG0480	92	95	5	AAQ	2016-09-15
6	PG0252	80	62	55	AAQ	2016-09-15
7	PG0480	94	97	3	AAQ	2016-09-16
8	PG0252	81	62	52	AAQ	2016-09-16
9	PG0480	93	96	4	AAQ	2016-09-17
10	PG0252	80	62	54	AAQ	2016-09-17
11	PG0480	89	92	8	AAQ	2016-09-18
12	PG0252	76	58	62	AAQ	2016-09-18
13	PG0480	92	95	5	AAQ	2016-09-19
14	PG0252	76	58	59	AAQ	2016-09-19
15	PG0480	73	75	24	AAQ	2016-09-20

## 5. Список SQL запросов из Приложения №2 – вопрос 6

«Найдите процентное соотношение перелетов по типам самолетов от общего количества.»

```
select
t.aircraft_code as «Тип самолета»,
round(t.fl_by_aircraft/t.ttl_fl::numeric*100.0,0) as «% полетов от общего»
from (
select
—схлопываем по коду самолета
distinct aircraft_code,
—считаем количество перелетов по каждому типу самолета
count(flight_id) over (partition by aircraft_code) fl_by_aircraft,
—считаем общее количество перелетов
count(flight_id) over () ttl_fl
from flights
group by flight_id , aircraft_code
)t
order by 2 desc
```

	Тип самолета	% полетов от общего
1	CN1	28
2	CR2	27
3	SU9	26
4	321	6
5	763	4
6	319	4
7	733	4
8	773	2

## 5. Список SQL запросов из Приложения №2 – вопрос 7

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

—Готовим в CTE необходимые данные, в частности, подтягиваем из связанных таблиц город назначения.

```
with ct1 as (
select tf.flight_id, a.city, tf.fare_conditions, tf.amount
from ticket_flights tf
join flights f on f.flight_id = tf.flight_id
join airports a on a.airport_code = f.arrival_airport
```

—по условию отбираем все строки в которых нет класса Комфорт

```
where tf.fare_conditions != 'Comfort'
```

—группируем, чтобы схлопнуть значения и получить уникальные стоимости для каждого перелета

```
group by 1,2,3,4
)
select distinct ct1.city as «Город»
```

```
from ct1
```

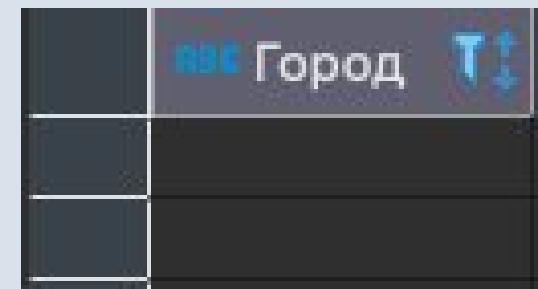
—присоединяем ту же CTE по условию , чтобы провести сравнение

```
join ct1 ct_1 on ct_1.flight_id = ct1.flight_id and ct1.fare_conditions = 'Business' and
ct_1.fare_conditions = 'Economy'
```

—здесь находим перелеты бизнес классом, которые по стоимость дешевле перелетов в тот же город Эконом-классом

```
where ct1.amount < ct_1.amount
```

В учебной базе нет городов, соответствующих условию



# 5. Список SQL запросов из Приложения №2 – вопрос 8

«Между какими городами нет прямых рейсов?»

—создаем представление

—получаем список пар городов из таблицы airports

```
create view city_pairs as
select da.city dac, aa.city aac
from airports da , airports aa
)
```

—explain analyze

```
select distinct dac, aac
from city_pairs
```

—убираем одинаковые пары

```
where dac!=aac
```

—теперь нужно отсечь те города, в которые осуществляются прямые перелеты

```
except select da1.city, aa1.city
from flights f,
airports da1, airports aa1
where f.departure_airport = da1.airport_code and
f.arrival_airport = aa1.airport_code
order by 1,2
```

	Город 1	Город 2
1	Абакан	Анадырь
2	Абакан	Анапа
3	Абакан	Астрахань
4	Абакан	Барнаул
5	Абакан	Белгород
6	Абакан	Белоярский
7	Абакан	Благовещенск
8	Абакан	Братск
9	Абакан	Брянск
10	Абакан	Бугульма
11	Абакан	Владивосток
12	Абакан	Владикавказ
13	Абакан	Волгоград
14	Абакан	Воркута
15	Абакан	Воронеж

# 5. Список SQL запросов из Приложения №2 – вопрос 9

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

—к мат. представлению routes добавляю недостающие данные, дистинктом отсекаю дубли

**select distinct**

r.departure\_airport **as** «A/п 1», a.latitude **as** «Широта1», a.longitude **as** «Долгота1»,

r.arrival\_airport **as** «A/п 2», a2.latitude **as** «Широта2», a2.longitude **as** «Долгота2»,

a3.model **as** «Самолет на маршруте», a3.range **as** «Дальность самолета»,

—Рассчитываю по формуле расстояние между а/п на основании того, что 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 измеряется 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.

**acos(sind(a.latitude)\*sind(a2.latitude) + cosd(a.latitude)\*cosd(a2.latitude)\*cosd(a.longitude - a2.longitude))\*6371** **as** «Расст. между а/п»,

—Создаю case, чтобы выполнить проверку долетит ли самолет из Города1 в Город2 с учетом его дальности полета

**case**

**when a3.range >= acos(sind(a.latitude)\*sind(a2.latitude) + cosd(a.latitude)\*cosd(a2.latitude)\*cosd(a.longitude - a2.longitude))\*6371** **then** 'Долетит'

**else** 'Не долетит'

**end as** «Проверка»

```
from routes r
join airports a on a.airport_code = r.departure_airport
join airports a2 on a2.airport_code = r.arrival_airport
join aircrafts a3 on a3.aircraft_code = r.aircraft_code
order by 1, 2
```

	A/п 1	Широта1	Долгота1	A/п 2	Широта2	Долгота2	Самолет на маршруте	Дальность самолета	Расст. между а/п	Проверка
1	AAQ	45.002097	37.347272	EGO	50.6438	36.5901	Sukhoi SuperJet-100	3,000	629.8610253813	Долетит
2	AAQ	45.002097	37.347272	NOZ	53.8114	86.8772	Boeing 737-300	4,200	3,634.0205451586	Долетит
3	AAQ	45.002097	37.347272	SVO	55.972642	37.414589	Boeing 737-300	4,200	1,219.8780891226	Долетит
4	ABA	53.74	91.385	ARH	64.360281	40.430167	Airbus A319-100	6,700	3,041.9666988538	Долетит
5	ABA	53.74	91.385	DME	55.408611	37.906111	Airbus A319-100	6,700	3,366.2987290683	Долетит
6	ABA	53.74	91.385	GRV	43.2981	45.7841	Boeing 737-300	4,200	3,484.1502587948	Долетит
7	ABA	53.74	91.385	KYZ	51.6694	94.4006	Cessna 208 Caravan	1,200	307.0146844489	Долетит
8	ABA	53.74	91.385	OVB	55.012622	82.650656	Cessna 208 Caravan	1,200	582.6951928199	Долетит
9	ABA	53.74	91.385	TOF	56.380278	85.208333	Cessna 208 Caravan	1,200	490.5303674072	Долетит
10	AER	43.449928	39.956589	EGO	50.6438	36.5901	Cessna 208 Caravan	1,200	839.3722327612	Долетит
11	AER	43.449928	39.956589	GOJ	56.230119	43.784042	Bombardier CRJ-200	2,700	1,446.7861294296	Долетит
12	AER	43.449928	39.956589	IAR	57.5606666767	40.1573694544	Bombardier CRJ-200	2,700	1,569.1050339553	Долетит
13	AER	43.449928	39.956589	IWA	56.939444	40.940833	Bombardier CRJ-200	2,700	1,501.5608253167	Долетит
14	AER	43.449928	39.956589	KGP	62.18	74.53	Boeing 737-300	4,200	3,055.1383070445	Долетит
15	AER	43.449928	39.956589	KJA	56.18	92.475	Airbus A319-100	6,700	3,913.1300654191	Долетит

# Спасибо