

Итоговая работа по курсу «SQL и получение данных»

02 апреля 2021г.

группа SQL-27

Новоселов П.В.

В работе использовался локальный тип подключения. База данных была восстановлена из бэкап-файла (рисунок 1).

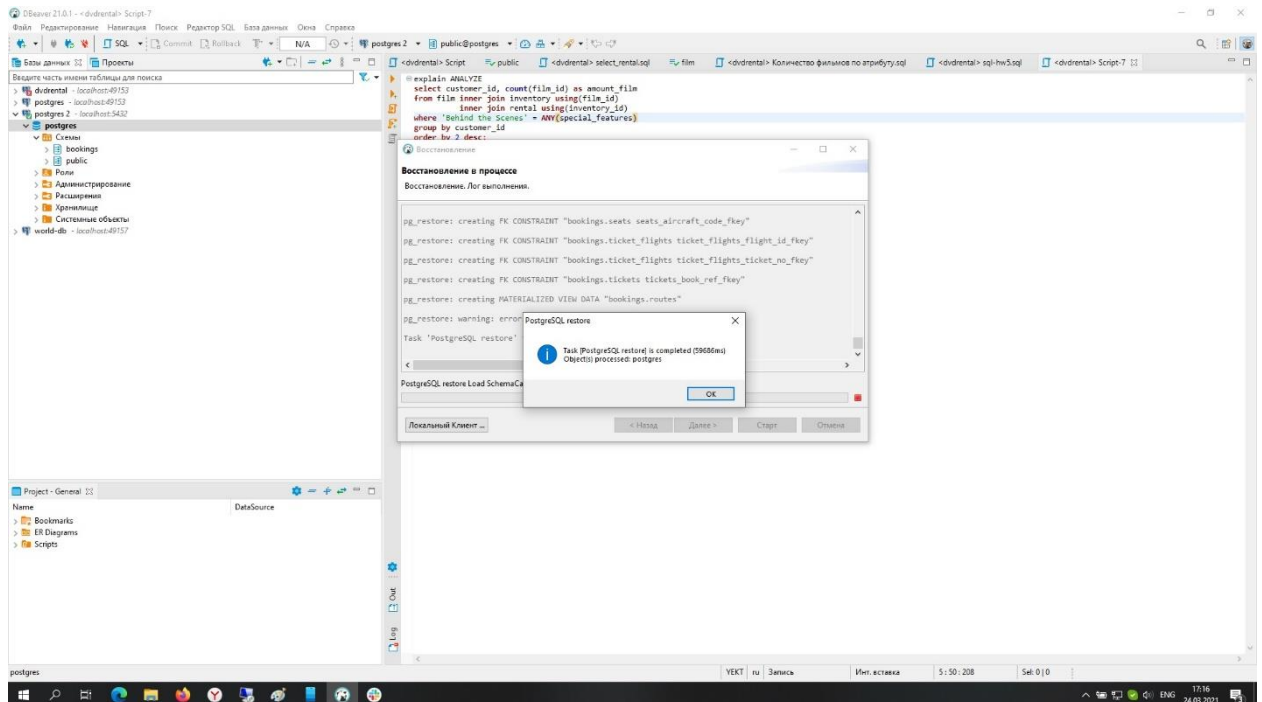


Рисунок 1 Скриншот восстановления базы данных

Описание базы данных

Диаграмма базы данных представлена на рисунке 2.

База данных «Booking» (авиаперевозки) состоит из восьми таблиц и нескольких представлений. В качестве предметной области выбраны авиаперевозки по России. Таблицы базы данных: bookings, tickets, ticket_flights, flights, airports, boarding_passes, seats, aircrafts. Представления: bookings.flights_v и bookings.routes.

Основной сущностью является бронирование (bookings).

В одно бронирование можно включить нескольких пассажиров, каждому из которых выписывается отдельный билет (tickets). Билет имеет уникальный номер и содержит информацию о пассажире. Как таковой пассажир не является отдельной сущностью.

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 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.

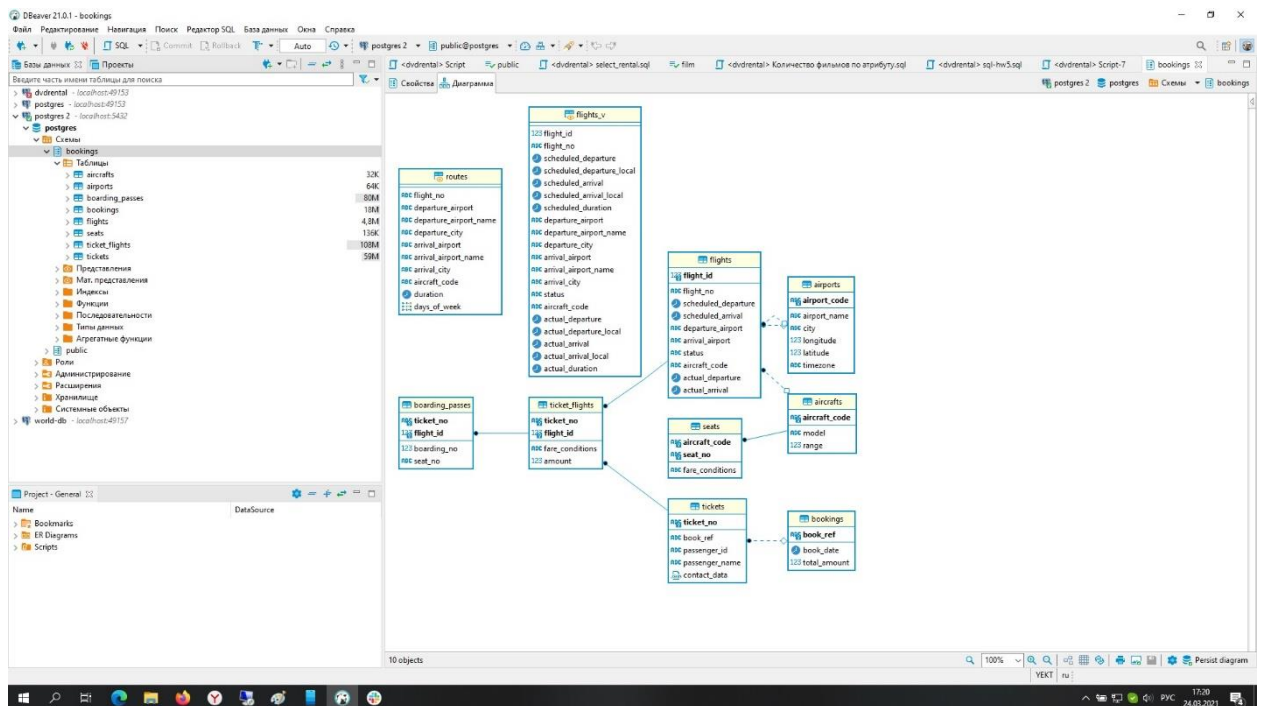


Рисунок 2 Диаграмма базы данных

Описание таблиц базы данных

Таблица aircrafts

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

Первичный ключ - aircraft_code, связана с таблицами flights и seats по ключу aircraft_code.

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).

Первичный ключ - airport_code, связана с таблицей flights по ключам arrival_airport и departure_airport.

Таблица boarding_passes

При регистрации на рейс 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).

Первичный ключ составной - ticket_no + flight_id, связана с таблицей ticket_flights по ключу 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.

Первичный ключ - book_ref, связана с таблицей tickets по ключу book_ref.

Таблица flights

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). Реальное 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.

Первичный ключ - flight_id, связана с таблицами:

- aircrafts по ключу aircraft_code;
- airports по ключам arrival_airport и departure_airport;
- ticket_flights по ключу flight_id.

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.

Первичный ключ составной - aircraft_code, seat_no, связана с таблицей aircrafts по ключу aircraft_code.

Таблица ticket_flights

Перелет соединяет билет с рейсом и идентифицируется их номерами.

Для каждого перелета указываются его стоимость (amount) и класс обслуживания (fare_conditions).

Первичный ключ составной - ticket_no + flight_id, связана с таблицами:

- boarding_passes по ключу ticket_no + flight_id;
- flights по ключу flight_id;
- tickets по ключу ticket_no.

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_data).

Первичный ключ - ticket_no, связана с таблицами ticket_flights по ключу ticket_no и bookings по ключу book_ref.

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.

Бизнес-задачи, которые можно решить с использованием БД

При помощи этой базы данных можно решить ряд бизнес-задач, связанных с организацией пассажирских авиаперевозок и аналитикой совершенных перелетов. В частности, можно создавать и отслеживать бронирования перелетов, контролировать наличие мест в самолетах, проверять соответствие транспортного судна требованиям к рейсу (дальность, наличие класса обслуживания и т.д.), возможно построение маршрутов между городами, где нет прямого сообщения с учетом времени и стоимости. Из аналитических задач можно проверять заполняемость самолетов, выявлять наиболее популярные маршруты, отслеживать перелеты конкретных пассажиров с учетом класса обслуживания и стоимости, чтобы выявить их предпочтения и т.д.

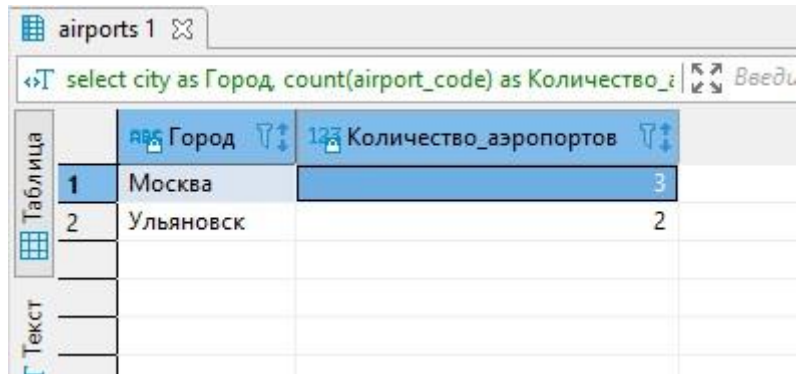
Список SQL запросов из приложения №2 с описанием логики их выполнения

1. В каких городах больше одного аэропорта?

Запрос:

```
select city as Город, count(airport_code) as Количество_аэропортов
from airports
group by city
having count(airport_code) > 1
order by 2 desc;
```

Результат выполнения:



	Город	Количество_аэропортов
1	Москва	3
2	Ульяновск	2

Логика запроса: выбираем города из таблицы airports с группировкой по названию городов, функцией count считаем количество кодов аэропортов в каждом городе и выбираем только те, где количество больше 1. Сортировка по количеству аэропортов в порядке убывания.

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

Запрос:

```
select airport_name||', '||city as Аэропорт
from airports inner join (select distinct departure_airport
                        from flights
                        where aircraft_code in (
                            select aircraft_code
                            from aircrafts
                            where "range" = (select
                                            max(range) from aircrafts)
                                            )
                        ) query
on airports.airport_code = query.departure_airport;
```

Результат выполнения:

Результат 1

```
select airport_name||', '||city as Аэропорт from air
```

Таблица	Аэропорт
1	Пермь, Пермь
2	Шереметьево, Москва
3	Кольцово, Екатеринбург
4	Внуково, Москва
5	Толмачёво, Новосибирск
6	Сочи, Сочи
7	Домодедово, Москва

Логика запроса: для нахождения модели самолета с максимальной дальностью перелета используем запрос `select max(range) from aircrafts`, где при помощи функции `max()` определяем максимум в столбце `range` таблицы `aircrafts`. Далее используем этот запрос в качестве подзапроса в условии `where` в выборке кодов самолетов из таблицы `aircrafts`. Для выбора аэропортов, где есть рейсы, выполняемые этими видами самолетов, используем запрос к таблице `flights` с применением в условии `where` составленного выше подзапроса (с кодами самолетов). Используем оператор `in` на случай, если таких моделей несколько. Получив таким образом список кодов аэропортов, соединим его через внутреннее соединение с таблицей `airports` для получения названия аэропорта и города, где он находится. Список названий аэропортов с городами их расположения выводим в итоговую выдачу.

3. Вывести 10 рейсов с максимальным временем задержки вылета. (В решении должен быть оператор LIMIT)

Запрос:

```
select flight_no as Номер_рейса, status as Статус,  
       actual_departure - scheduled_departure as Задержка  
from flights  
where status in ('Departed', 'Arrived')  
order by Задержка desc limit 10;
```

Результат выполнения:

flights 1

```
select flight_no as Номер_рейса, status as Статус, actual_
```

Таблица	Номер_рейса	Статус	Задержка
1	PG0589	Arrived	04:37:00
2	PG0164	Arrived	04:28:00
3	PG0364	Arrived	04:27:00
4	PG0568	Arrived	04:20:00
5	PG0454	Arrived	04:18:00
6	PG0096	Arrived	04:18:00
7	PG0166	Arrived	04:16:00
8	PG0278	Arrived	04:16:00
9	PG0564	Arrived	04:14:00
10	PG0669	Arrived	04:08:00

Логика запроса: формируем запрос к таблице `flights`, из которой выбираем для большей информативности номер рейса, статус рейса и столбец с вычисляемым временем задержки (реальное время вылета - запланированное время вылета) с названием «Задержка». Эти данные фильтруем по статусам «Вылетел» и «Прибыл», т.к. в других статусах самолет ещё не взлетел и окончательное время задержки, если она будет, неизвестно. Полученные данные выводим в отсортированном по убыванию времени задержки виде с ограничением оператором `limit` количеством записей в 10 строк.

4. Были ли брони, по которым не были получены посадочные талоны? (В решении должен быть верный тип JOIN)

Запрос:

```
select book_ref as Номер_брони,
       passenger_name as Пассажир,
       flight_no as Номер_рейса,
       status as Статус,
       boarding_no as Номер_посадочного_талона
from tickets t left join boarding_passes bp on t.ticket_no = bp.ticket_no
              left join flights f2 on bp.flight_id = f2.flight_id
where status in ('Departed', 'Arrived') and boarding_no is null;
```

Результат выполнения:

Номер_брони	Пассажир	Номер_рейса	Статус	Номер_посадочного_талона

Логика запроса: для ответа на вопрос делаем внешнее соединение таблиц `tickets` и `boarding_passes` по номеру билета, соединение левое из `tickets` в `boarding_passes`, т.е. по тем билетам, по которым нет регистрации в итоговую выборку должны записаться значения NULL. Присоединим также по левому внешнему соединению таблицу `flights`, чтобы отфильтровать только те рейсы, которые уже в полете или прибыли, т.к. запланированные рейсы, доступные для регистрации, отложенные или отмененные ещё не содержат регистраций. В итоговой выборке для соответствующих статусов рейсов выбираем только те значения, для которых номера посадочных талонов имеют значения NULL, т.е. отсутствуют. Получаем пустую выборку, значит по всем бронированиям были получены посадочные талоны.

5. Найдите свободные места 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 за день. (В решении должно быть: оконная функция, подзапросы)

Запрос:

```
select flight_no as Номер_рейса,
       airport_name||', '||city as Аэропорт_вылета,
       scheduled_departure as Время_вылета,
       total_seats as Всего_мест_в_самолете,
       real_seats as Мест_занято,
       (total_seats - real_seats) as Мест_свободно,
       (round(((total_seats - real_seats)::numeric/total_seats::numeric), 2)*100)::integer||'%' as Процент_свободных_мест,
       sum(real_seats) over (partition by departure_airport, date_trunc('day',
scheduled_departure) order by scheduled_departure) as Перевезено_людей_за_день
from (
    select flight_id, count(seat_no) as real_seats
    from boarding_passes bp
    group by flight_id) rs
inner join
    flights f
    on rs.flight_id = f.flight_id
inner join
    (select aircraft_code, count(seat_no) as total_seats
    from seats s
    group by aircraft_code
    ) ts
    on f.aircraft_code = ts.aircraft_code
inner join airports a2
    on f.departure_airport = a2.airport_code
order by departure_airport, scheduled_departure desc;
```

Результат выполнения:

п_рейса	Аэропорт_вылета	Время_вылета	Всего_мест_в_самолете	Мест_занято	Мест_свободно	Процент_свободных_мест	Перевезено_людей_за_день
1	Витязево, Анапа	2016-10-13 14:05:00	130	119	11	8%	202
2	Витязево, Анапа	2016-10-13 13:25:00	97	83	14	14%	83
3	Витязево, Анапа	2016-10-12 14:05:00	130	129	1	1%	213
4	Витязево, Анапа	2016-10-12 13:25:00	97	84	13	13%	84
5	Витязево, Анапа	2016-10-11 14:05:00	130	108	22	17%	197
6	Витязево, Анапа	2016-10-11 13:25:00	97	89	8	8%	89
7	Витязево, Анапа	2016-10-10 14:05:00	130	116	14	11%	212
8	Витязево, Анапа	2016-10-10 13:25:00	97	96	1	1%	96
9	Витязево, Анапа	2016-10-09 14:05:00	130	117	13	10%	208
10	Витязево, Анапа	2016-10-09 13:25:00	97	91	6	6%	91
11	Витязево, Анапа	2016-10-08 14:05:00	130	106	24	18%	203
12	Витязево, Анапа	2016-10-08 13:25:00	97	97	0	0%	97
13	Витязево, Анапа	2016-10-07 14:05:00	130	110	20	15%	192
14	Витязево, Анапа	2016-10-07 13:25:00	97	82	15	15%	82
15	Витязево, Анапа	2016-10-06 14:05:00	130	112	18	14%	208
16	Витязево, Анапа	2016-10-06 13:25:00	97	96	1	1%	96
17	Витязево, Анапа	2016-10-05 14:05:00	130	115	15	12%	212
18	Витязево, Анапа	2016-10-05 13:25:00	97	97	0	0%	97
19	Витязево, Анапа	2016-10-04 14:05:00	130	117	13	10%	199
20	Витязево, Анапа	2016-10-04 13:25:00	97	82	15	15%	82
21	Витязево, Анапа	2016-10-03 14:05:00	130	120	10	8%	214
22	Витязево, Анапа	2016-10-03 13:25:00	97	94	3	3%	94
23	Витязево, Анапа	2016-10-02 14:05:00	130	98	32	25%	189
24	Витязево, Анапа	2016-10-02 13:25:00	97	91	6	6%	91
25	Витязево, Анапа	2016-10-01 14:05:00	130	101	29	22%	179
26	Витязево, Анапа	2016-10-01 13:25:00	97	78	19	20%	78
27	Витязево, Анапа	2016-09-30 14:05:00	130	114	16	12%	208
28	Витязево, Анапа	2016-09-30 13:25:00	97	94	3	3%	94
29	Витязево, Анапа	2016-09-29 14:05:00	130	126	4	3%	212
30	Витязево, Анапа	2016-09-29 13:25:00	97	86	11	11%	86
31	Витязево, Анапа	2016-09-28 14:05:00	130	107	23	18%	204
32	Витязево, Анапа	2016-09-28 13:25:00	97	97	0	0%	97
33	Витязево, Анапа	2016-09-27 14:05:00	130	124	6	5%	214
34	Витязево, Анапа	2016-09-27 13:25:00	97	90	7	7%	90
35	Витязево, Анапа	2016-09-26 14:05:00	130	111	19	15%	197
36	Витязево, Анапа	2016-09-26 13:25:00	97	86	11	11%	86
37	Витязево, Анапа	2016-09-25 14:05:00	130	104	26	20%	189
38	Витязево, Анапа	2016-09-25 13:25:00	97	85	12	12%	85
39	Витязево, Анапа	2016-09-24 14:05:00	130	120	10	8%	188
40	Витязево, Анапа	2016-09-24 13:25:00	97	68	29	30%	68
41	Витязево, Анапа	2016-09-23 14:05:00	130	96	34	26%	160
42	Витязево, Анапа	2016-09-23 13:25:00	97	64	33	34%	64

Логика запроса: для определения количества мест в самолете используется запрос

```
select aircraft_code, count(seat_no) as total_seats
    from seats s
    group by aircraft_code
```

где из таблицы `seats` по коду самолета подсчитывается общее количество мест с присвоением этому количеству имени `total_seats`. Данные группируются по коду самолета.

Для определения заполненности самолета используется запрос

```
select flight_id, count(seat_no) as real_seats
      from boarding_passes bp
      group by flight_id
```

где из таблицы `boarding_passes` для каждого рейса производится подсчет количества номеров выданных при регистрации мест. Данные группируются по уникальному идентификатору рейса.

Эти запросы соединяются между собой и с таблицей `flights` по внутреннему соединению (предполагаем, что пустые самолеты не летают).

Из объединенной выборки запрашиваем номер рейса, аэропорт вылета, время вылета, общее количество мест в самолете на рейсе, количество занятых на рейсе мест, количество свободных мест как разность между общим количеством и занятыми местами, процент заполняемости самолета, а также накопительный итог по перевозкам пассажиров в этом аэропорту за этот же день на этом и более ранних рейсах.

Процент заполняемости самолета рассчитывается выражением:

```
(round(((total_seats - real_seats)::numeric/total_seats::numeric), 2)*100)::integer
```

где количество свободных мест делим на общее количество мест и при помощи функции `round` округляем результат до 2 знаков после запятой, потом умножаем на 100%. Т.к. количество мест - целые числа, а при делении целых чисел результат округляется до целого, то перед делением преобразуем тип данных количества мест в `numeric`, а после вычисления процента, обратно в `integer`. Для наглядности добавляем в столбец конкатенацию со строкой символа процента.

Накопительный итог по перевозкам пассажиров в этом аэропорту за этот же день на этом и более ранних рейсах рассчитывается выражением:

```
sum(real_seats) over (partition by departure_airport, date_trunc('day', scheduled_departure) order by scheduled_departure)
```

где формируем «окно» по аэропорту вылета и дню вылета (выделяем из даты отправления день функцией `date_trunc`), добавляем в «окне» сортировку по времени вылета, чтобы подсчет шёл только до текущего времени и для каждого «окна» суммируем количество занятых мест в самолете.

Для наглядности присоединяем таблицу `airports`, чтобы получить из неё название аэропорта и город, которые конкатенируем между собой и помещаем в столбец «Аэропорт вылета».

Результат отсортирован по алфавиту для аэропортов и затем по времени вылета в порядке убывания.

6. Найдите процентное соотношение перелетов по типам самолетов от общего количества. (В решении должен быть подзапрос и оператор ROUND)

Запрос:

```
select distinct model as Самолет,
      count(flight_id) over (partition by f.aircraft_code) as Перелеты_самолета,
      count(flight_id) over() as Перелетов_всего,
      (round((count(flight_id) over (partition by f.aircraft_code)::numeric/count(flight_id) over()::numeric), 2)*100)::integer||'%' as Процент_перелетов
from flights f right join aircrafts a
      on f.aircraft_code = a.aircraft_code
group by model, flight_id;
```

Результат выполнения:

aircrafts 1		select distinct model as Самолет, count(flight_id) over (p			Введите SQL выражение чтобы отфильтровать резул	
Таблица	Самолет	Перелеты_самолета	Перелетов_всего	Процент_перелетов		
1	Airbus A319-100	1 239	33 121	4%		
2	Airbus A320-200	0	33 121	0%		
3	Airbus A321-200	1 952	33 121	6%		
4	Boeing 737-300	1 274	33 121	4%		
5	Boeing 767-300	1 221	33 121	4%		
6	Boeing 777-300	610	33 121	2%		
7	Bombardier CRJ-200	9 048	33 121	27%		
8	Cessna 208 Caravan	9 273	33 121	28%		
9	Sukhoi SuperJet-100	8 504	33 121	26%		

Логика запроса: для решения соединяем таблицы flights и aircrafts по внешнему соединению, чтобы добавить самолеты, которые не выполняли рейсов вообще. Из итоговой выборки выводим модель самолета (добавляем признак уникальности, чтобы избежать дублирования записей), считаем количество рейсов по каждому самолету (в «окне» по коду самолета), считаем общее количество рейсов по идентификатору рейса и процент перелетов для каждого самолета при помощи выражения:

```
(round((count(flight_id) over (partition by f.aircraft_code)::numeric/count(flight_id) over()::numeric), 2)*100)::integer
```

где посчитанное ранее количество рейсов самолета делим на общее количество рейсов, результат округляем до 2 знаков после запятой и умножаем на 100%. Как и в предыдущем задании производим преобразование типов. Для наглядности добавляем в столбец конкатенацию со строкой символа процента.

Если посчитать для проверки все проценты по всем самолетам, то получится 101%, что отличается от 100%. Это вызвано погрешностью в округлении результатов деления.

7. Были ли города, в которые можно добраться бизнес - классом дешевле, чем эконом-классом в рамках перелета? (В решении должно использоваться СТЕ)

Запрос:

```
with cte_business as (
    select flight_id, fare_conditions, amount as business_amount
    from ticket_flights
    where fare_conditions = 'Business'
),
cte_economy as (
    select flight_id, fare_conditions, amount as economy_amount
    from ticket_flights
    where fare_conditions = 'Economy'
)
select flight_no as Номер_рейса, city as Город,
    cte_business.fare_conditions as Бизнес_класс,
    business_amount as Стоимость_бизнеса,
    cte_economy.fare_conditions as Эконом_класс,
    economy_amount as Стоимость_эконома
from flights inner join cte_business on flights.flight_id = cte_business.flight_id
    inner join cte_economy on flights.flight_id = cte_economy.flight_id
    inner join airports on flights.arrival_airport = airports.airport_code
where business_amount < economy_amount
group by flight_no, city, cte_business.fare_conditions, business_amount, cte_economy.fare_conditions, economy_amount;
```

Результат выполнения:

Номер_рейса	Город	Бизнес_класс	Стоимость_бизнеса	Эконом_класс	Стоимость_эконома

Логика запроса: создадим общие табличные выражения (cte).

cte_business, в котором из таблицы ticket_flights выберем идентификатор рейса (перелета), класс обслуживания и стоимость перелета в бизнес-классе, таким образом мы получим стоимость бизнес-класса по всем перелетам.

Аналогично создадим cte_economy, в котором получим стоимость эконом-класса по всем перелетам.

В итоговом запросе соединим внутренним соединением оба cte по идентификатору перелетов, таким образом получив для каждого перелета одновременно стоимость в бизнес-классе и в эконом-классе. Отфильтруем запрос по стоимости, где эконом-класс дороже бизнес-класса и получим список перелетов, где бизнес-класс дешевле эконома, если таковые есть. Добавим соединение с таблицей аэропортов, чтобы получить название города. Для наглядности выведем в запросе номер рейса, город прибытия (куда можно было бы долететь дешевле) и для сравнения класс обслуживания и его стоимость по данному перелету.

В результате выполнения получили пустую таблицу, а значит, что нет таких городов, куда бизнес-классом напрямую долететь дешевле, чем экономом.

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

Запрос:

```
create view departure as (  
    select distinct city as departure_city  
    from flights inner join airports  
    on flights.departure_airport = airports.airport_code  
);  
  
create view arrival as (  
    select distinct city as arrival_city  
    from flights inner join airports  
    on flights.arrival_airport = airports.airport_code  
);  
  
select departure_city as Город_отправления, arrival_city as Город_прибытия  
from departure d, arrival a  
where departure_city <> arrival_city  
except  
select departure_city, arrival_city  
from flights_v fv  
order by 1, 2;
```

Результат выполнения:

Город_отправления	Город_прибытия
Абакан	Анадырь
Абакан	Анапа
Абакан	Астрахань
Абакан	Барнаул
Абакан	Белгород
Абакан	Белоярский
Абакан	Благовещенск
Абакан	Братск
Абакан	Брянск
Абакан	Бугульма
Абакан	Владивосток
Абакан	Владикавказ
Абакан	Волгоград
Абакан	Воркута
Абакан	Воронеж
Абакан	Геленджик
Абакан	Горно-Алтайск
Абакан	Екатеринбург
Абакан	Иваново
Абакан	Ижевск
Абакан	Иркутск
Абакан	Йошкар-Ола
Абакан	Казань
Абакан	Калининград
Абакан	Калуга
Абакан	Кемерово
Абакан	Киров
Абакан	Когалым
Абакан	Комсомольск-на-Амуре
Абакан	Краснодар
Абакан	Красноярск
Абакан	Курган
Абакан	Курск
Абакан	Липецк
Абакан	Магадан
Абакан	Магнитогорск
Абакан	Махачкала
Абакан	Минеральные Воды
Абакан	Мирный
Абакан	Мурманск
Абакан	Надым
Абакан	Нальчик
Абакан	Наньшань

Логика запроса: для решения задачи создадим два представления `departure` и `arrival`, в которых выберем города расположения аэропортов, соответственно для аэропортов отправления и аэропортов прибытия. Далее при помощи декартового произведения соединим эти два представления, получив записи, где каждый город отправления соединен с каждым городом прибытия (исключим по условию соединение города с самим собой). При помощи оператора `except` исключим из получившейся выборки те записи из представления `flights_v`, которые соединяют города отправления и прибытия по существующим

рейсам, получив таким образом список соединений городов, между которыми нет прямых рейсов.

Результат отсортирован по алфавиту для городов отправления и затем для городов прибытия.

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

Запрос:

```
with cte_departure as (
    select airport_code,
    airport_name||', '||city as Аэропорт_вылета,
    latitude as departure_latitude,
    longitude as departure_longitude
    from airports
),
cte_arrival as (
    select airport_code,
    airport_name||', '||city as Аэропорт_прилета,
    latitude as arrival_latitude,
    longitude as arrival_longitude
    from airports
),
cte_dep_ar as (
    select f.departure_airport,
    ctd.Аэропорт_вылета,
    cta.Аэропорт_прилета,
    f.arrival_airport,
    f.aircraft_code,
    round((6371 * (acos(sind(departure_latitude)*sind(arrival_latitude) + cosd(departure_latitude)*cosd(arrival_latitude)*cosd(departure_longitude - arrival_longitude)))))) as Расстояние
    from flights f inner join cte_departure ctd
    on f.departure_airport = ctd.airport_code
    inner join cte_arrival cta
    on f.arrival_airport = cta.airport_code
)

select Аэропорт_вылета,
    Аэропорт_прилета,
    model as Самолет,
    "range" as Дальность_полета,
    Расстояние
from cte_dep_ar inner join aircrafts
    on cte_dep_ar.aircraft_code = aircrafts.aircraft_code
group by 1, 2, 3, 4, 5
order by 1, 2, 5 desc;
```


Результат выполнения:

Аэропорт_вылета	Аэропорт_прилета	Самолет	Дальность_полета	Расстояние
Абакан, Абакан	Богашёво, Томск	Cessna 208 Caravan	1 200	491
Абакан, Абакан	Грозный, Грозный	Boeing 737-300	4 200	3 484
Абакан, Абакан	Домодедово, Москва	Airbus A319-100	6 700	3 366
Абакан, Абакан	Кызыл, Кызыл	Cessna 208 Caravan	1 200	307
Абакан, Абакан	Талаги, Архангельск	Airbus A319-100	6 700	3 042
Абакан, Абакан	Толмачёво, Новосибирск	Cessna 208 Caravan	1 200	583
Анадырь, Анадырь	Внуково, Москва	Airbus A319-100	6 700	6 220
Анадырь, Анадырь	Домодедово, Москва	Airbus A319-100	6 700	6 226
Анадырь, Анадырь	Хабаровск-Новый, Хабаровск	Airbus A319-100	6 700	3 074
Анадырь, Анадырь	Шереметьево, Москва	Airbus A319-100	6 700	6 177
Астрахань, Астрахань	Барнаул, Барнаул	Bombardier CRJ-200	2 700	2 637
Астрахань, Астрахань	Домодедово, Москва	Bombardier CRJ-200	2 700	1 235
Астрахань, Астрахань	Чульман, Нерюнгри	Airbus A319-100	6 700	5 147
Астрахань, Астрахань	Шереметьево, Москва	Bombardier CRJ-200	2 700	1 303
Байкал, Улан-Удэ	Внуково, Москва	Airbus A319-100	6 700	4 439
Байкал, Улан-Удэ	Надым, Надым	Bombardier CRJ-200	2 700	2 468
Байкал, Улан-Удэ	Якутск, Якутск	Bombardier CRJ-200	2 700	1 757
Баратаевка, Ульяновск	Внуково, Москва	Sukhoi SuperJet-100	3 000	715
Баратаевка, Ульяновск	Домодедово, Москва	Sukhoi SuperJet-100	3 000	672
Баратаевка, Ульяновск	Краснодар, Краснодар	Boeing 767-300	7 900	1 214
Баратаевка, Ульяновск	Орск, Орск	Cessna 208 Caravan	1 200	783
Баратаевка, Ульяновск	Пермь, Пермь	Sukhoi SuperJet-100	3 000	630
Баратаевка, Ульяновск	Шереметьево, Москва	Sukhoi SuperJet-100	3 000	712
Барнаул, Барнаул	Астрахань, Астрахань	Bombardier CRJ-200	2 700	2 637
Барнаул, Барнаул	Внуково, Москва	Sukhoi SuperJet-100	3 000	2 944
Барнаул, Барнаул	Домодедово, Москва	Sukhoi SuperJet-100	3 000	2 910
Барнаул, Барнаул	Емельяново, Красноярск	Cessna 208 Caravan	1 200	652
Барнаул, Барнаул	Шереметьево, Москва	Sukhoi SuperJet-100	3 000	2 924
Барнаул, Барнаул	Якутск, Якутск	Sukhoi SuperJet-100	3 000	2 839
Бегишево, Нижнекамск	Омск-Центральный, Омск	Bombardier CRJ-200	2 700	1 346
Бегишево, Нижнекамск	Ростов-на-Дону, Ростов-на-До	Bombardier CRJ-200	2 700	1 234
Бегишево, Нижнекамск	Сыктывкар, Сыктывкар	Cessna 208 Caravan	1 200	705
Бегишево, Нижнекамск	Чебоксары, Чебоксары	Bombardier CRJ-200	2 700	307
Белгород, Белгород	Брянск, Брянск	Sukhoi SuperJet-100	3 000	330
Белгород, Белгород	Витязево, Анапа	Sukhoi SuperJet-100	3 000	630
Белгород, Белгород	Домодедово, Москва	Sukhoi SuperJet-100	3 000	537
Белгород, Белгород	Ростов-на-Дону, Ростов-на-До	Cessna 208 Caravan	1 200	444
Белгород, Белгород	Сочи, Сочи	Cessna 208 Caravan	1 200	839

Логика запроса: для удобства решения задачи создадим общие табличные выражения (cte).

cte_departure , в котором создадим выборку из таблицы airports для аэропортов отправления, выберем код аэропорта, название+город аэропорта, широта и долгота расположения аэропорта. Такую же выборку создадим для аэропортов прибытия и поместим в cte_arrival.

В cte_dep_ar объединим таблицу перелетов flights с созданными ранее cte внутренними соединениями, т.к. не предполагается добавление в таблицу с перелетами отсутствующих в ней аэропортов. В этой выборке посчитаем расстояние между аэропортами выражением:

```
round((6371 * (acos(sind(departure_latitude)*sind(arrival_latitude) + cosd(departure_latitude)*cosd(arrival_latitude)*cosd(departure_longitude - arrival_longitude))))))
```

где подставим соответствующие значения в формулу для определения расстояния между двумя точками земной поверхности (6371 здесь, это радиус Земли в км) и округлим полученный результат до целого.

В итоговом запросе соединим внутренним соединением (добавление самолетов, которые не летают ни по каким маршрутом не делаем) `cte_dep_ar` и таблицу `aircrafts` и выберем из получившейся выборки аэропорт отправления, аэропорт прибытия, модель самолета, летающего по данному маршруту, дальность полета этого самолета и расстояние между аэропортами с группировкой по всем столбцам, чтобы исключить дублирование. Результат отсортируем по алфавиту для аэропорта отправления, затем также по алфавиту для аэропорта прибытия и потом по убыванию расстояния.

Запросы для решения всех заданий находятся в приложенном файле

`Itog_NovoselovP_SQL27.sql`