# Итоговая работа

1. Во время работы использовался локальный тип подключения. База была развернута из .backup файла.

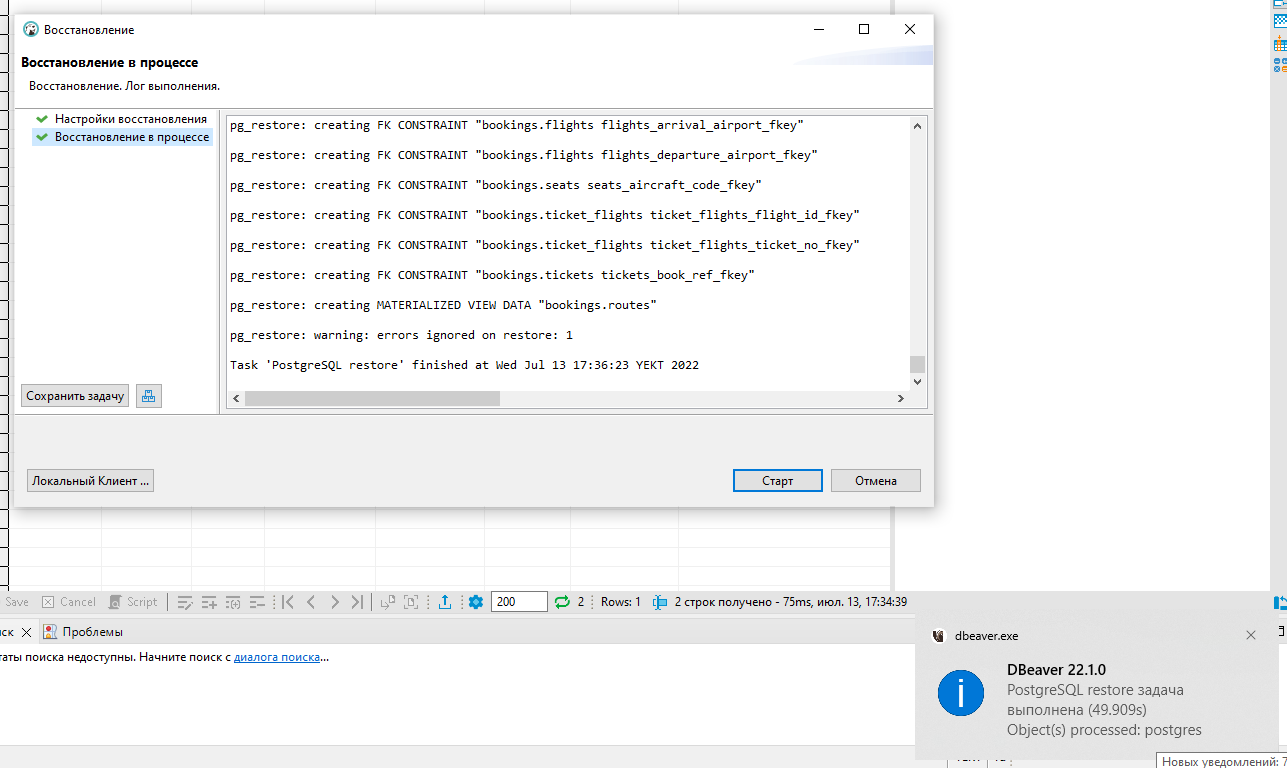


Рисунок 1 - Скриншот восстановления базы из .backup файла

2. Скриншот ER-диаграммы из DBeaver`a приведен на рисунке 2.

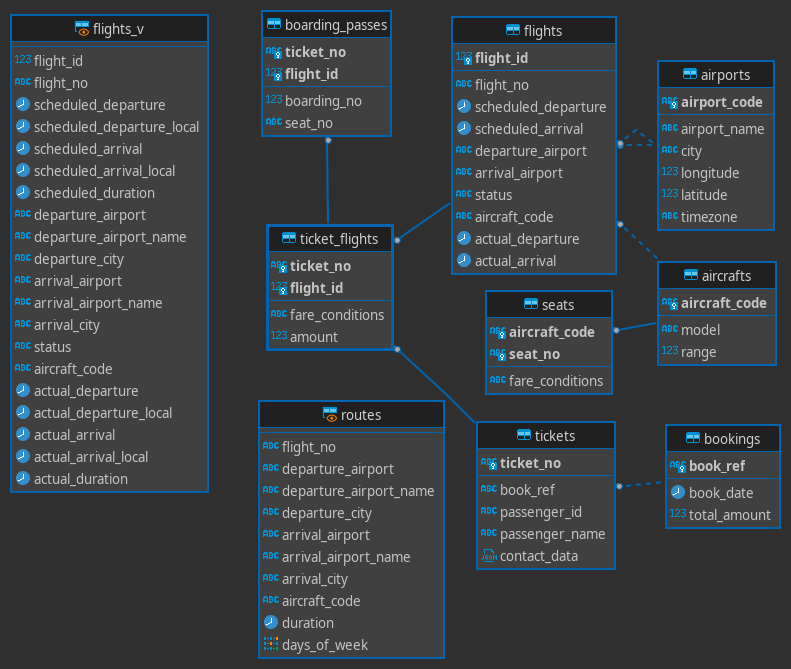


Рисунок 2 - Скриншот ER -диаграммы из из DBeaver`a

3. Краткое описание БД - из каких таблиц и представлений состоит

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)

### Список отношений

Имя             | Тип           | Small | Medium | Big    | Описание

-----------------+---------------+--------+--------+--------+-------------------

aircrafts       | таблица       | 16 kB | 16 kB  | 16 kB  | Самолеты

airports        | таблица       | 48 kB | 48 kB  | 48 kB  | Аэропорты

boarding\_passes | таблица       | 31 MB | 102 MB | 427 MB | Посадочные талоны

bookings        | таблица       | 13 MB | 30 MB  | 105 MB | Бронирования

flights         | таблица       | 3 MB  | 6 MB   | 19 MB  | Рейсы

flights\_v       | представление | 0 kb  | 0 kB   | 0 kB   | Рейсы

routes          | мат. предст.  | 136 kB| 136 kB | 136 kB | Маршруты

seats | таблица | 88 kB         | 88 kB | 88 kB  | Места

ticket\_flights  | таблица       | 64 MB | 145 MB | 516 MB | Перелеты

tickets         | таблица       | 47 MB | 107 MB | 381 MB | Билеты

#### 4. 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

-------------+------------+--------------+--------------------------

ticket\_no | char(13) | NOT NULL | Номер билета

flight\_id | integer | NOT NULL | Идентификатор рейса

boarding\_no | integer | NOT NULL | Номер посадочного талона

seat\_no | varchar(4) | NOT NULL | Номер места

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

--------------+---------------+--------------+---------------------------

book\_ref | char(6) | NOT NULL | Номер бронирования

book\_date | timestamptz | NOT NULL | Дата бронирования

total\_amount | numeric(10,2) | NOT NULL | 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 | timestamptz | NOT NULL | Время вылета по расписанию

scheduled\_arrival | timestamptz | 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 | timestamptz | | Фактическое время вылета

actual\_arrival | timestamptz | | 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\_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.

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

#### 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.

#### Использование

Схема bookings

Все объекты демонстрационной базы данных находятся в схеме bookings. Это означает, что

при обращении к объектам вам 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.

#### Бизнес задачи, которые можно решить, используя БД

1. Анализ загруженности самолетов

2. Анализ загруженности пассажиропотоков в срезах: направлений перелетов, временных параметрах (день недели, месяцы)

3. Анализ данных по невостребованным билетам, отсутствию посадки на купленные билеты.

4. Получение данных об одновременном нахождении самолетов в воздухе

5. Анализ рентабельности бизнеса (в связке с другими данными)

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

1. В каких городах больше одного аэропорта?

SELECT city, count(\*) "Кол-во аэропортов"

FROM bookings.airports a

GROUP BY city -- делаем группировку по городам

HAVING count(\*) >1; -- если строк в группе более 1, значит там более 1 аэропорта

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

SELECT DISTINCT airport\_code, (

SELECT max("range")

FROM bookings.aircrafts) Дальность

FROM bookings.airports

JOIN bookings.flights

ON departure\_airport = airport\_code OR arrival\_airport = airport\_code

WHERE aircraft\_code = (

SELECT aircraft\_code -- находим код самолета с максимальной дальностью

FROM bookings.aircrafts

WHERE "range" = (

SELECT max("range") -- находим максимальную дальность

FROM bookings.aircrafts));

3. Вывести 10 рейсов с максимальным временем задержки вылета

Используем данные из bookings.flights

SELECT flight\_id, flight\_no, actual\_departure - scheduled\_departure "Задержка"

FROM bookings.flights

WHERE actual\_departure IS NOT NULL -- исключаем рейсы без задержек

ORDER BY actual\_departure - scheduled\_departure DESC

LIMIT 10;

4. Были ли брони, по которым не были получены посадочные талоны

Используем данные из объединения таблиц bookings.ticket\_flights и boarding\_passes через left join для получения билетов без брони по значению is null атрибута boarding\_no

SELECT ticket\_no

FROM bookings.ticket\_flights

LEFT JOIN boarding\_passes USING (ticket\_no) --объединяем с влючением значений NULL из правой таблицы

WHERE boarding\_no IS NULL --отбираем те билеты, по которым не было брони

ORDER BY ticket\_no;

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.

В первом cte находим общее количество мест для каждого рейса

по общему количеству мест в самолете, забронированному на данный рейс

with seats\_flats\_cte as (

select flight\_id, count(\*) count\_seats, departure\_airport, scheduled\_departure, actual\_departure,

case

when actual\_departure is null then scheduled\_departure

else actual\_departure

end fact\_departure --фактическое время вылета

from

bookings.flights f

join bookings.seats s using (aircraft\_code)

group by flight\_id , aircraft\_code -- группируем для подсчета количества мест в самолете по рейсам

order by flight\_id),

--Находим количество посадочных талонов для каждого рейса

count\_boarding\_passes\_cte as (

select flight\_id, count(\*) count\_passes

from bookings.boarding\_passes bp

group by flight\_id

order by flight\_id),

--Находим количество свободных мест для каждого рейса

free\_seats\_flats\_cte as (

select \*

from

seats\_flats\_cte --Объединяем через left join, чтобы учесть null

left join count\_boarding\_passes\_cte using (flight\_id))

select flight\_id, fact\_departure,

case

when count\_passes is null then count\_seats

else count\_seats - count\_passes

end free\_seats,

case

when count\_passes is null then 100

else round(((count\_seats - count\_passes)::numeric/count\_seats)\*100, 1)

end "free\_seats, %",

coalesce(sum(count\_passes) over(partition by departure\_airport, fact\_departure::date order by fact\_departure), 0) "count\_sum"

from free\_seats\_flats\_cte

order by departure\_airport, fact\_departure;

--6. Найдите процентное соотношение перелетов по типам самолетов от общего количества.

select distinct

aircraft\_code,

round((count(aircraft\_code) over(partition by aircraft\_code))::numeric / (

select count(\*) from bookings.flights), 3)\*100 "Доля, %"

from

bookings.flights;

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

--Вариант 1

--Для начала находим стоимость перелетов по рейсам и классам

with amount\_city\_cte as (

select distinct

flight\_id, amount, fare\_conditions

from

bookings.ticket\_flights tf

join bookings.flights using(flight\_id)

join bookings.aircrafts using(aircraft\_code)),

-- отбираем перелеты по классу Economy

amount\_city\_cte\_economy as (

select \*

from amount\_city\_cte

where fare\_conditions = 'Economy'),

-- отбираем перелеты по классу Business

amount\_city\_cte\_business as (

select \*

from amount\_city\_cte

where fare\_conditions = 'Business')

-- соединяем cte перелетов по разным классам по условию равенства flight\_id

select

flight\_id

from

amount\_city\_cte\_economy

join amount\_city\_cte\_business using(flight\_id)

where amount\_city\_cte\_business.amount < amount\_city\_cte\_economy.amount; -- отбираем по этому условию

--Результат нулевой. Значит не было

--Вариант 2

--Для начала находим максимальные и минимальные стоимость перелетов по рейсам и классам для Economy и Business

--С группировкой по flight\_id, fare\_conditions

with amount\_city\_cte as (

SELECT

flight\_id, fare\_conditions,

CASE

WHEN fare\_conditions = 'Economy' THEN max(amount)

ELSE min(amount)

END amount

from

bookings.ticket\_flights tf

join bookings.flights using(flight\_id)

join bookings.aircrafts using(aircraft\_code)

WHERE tf.fare\_conditions = 'Business' OR tf.fare\_conditions = 'Economy'

GROUP BY flight\_id, fare\_conditions

ORDER BY flight\_id, fare\_conditions

)

SELECT flight\_id

from

(SELECT flight\_id, array\_agg(fare\_conditions) fare\_conditions, array\_agg(amount) amount

FROM amount\_city\_cte

GROUP BY flight\_id

HAVING count(\*) > 1) q

WHERE amount[1] < amount[2]; --проверяем значения в массиве

--Результат нулевой. Значит не было

--Вариант 3

--Для начала находим максимальные и минимальные стоимость перелетов по рейсам и классам для Economy и Business

--С группировкой по flight\_id, fare\_conditions

with amount\_city\_cte as (

SELECT

flight\_id, fare\_conditions,

CASE --для Economy берем максимальную стоимость для рейса

WHEN fare\_conditions = 'Economy' THEN max(amount)

ELSE min(amount) --для Business берем минимальную стоимость для рейса

END amount

from

bookings.ticket\_flights tf

join bookings.flights using(flight\_id)

join bookings.aircrafts using(aircraft\_code)

WHERE fare\_conditions = 'Business' OR fare\_conditions = 'Economy'

GROUP BY flight\_id, fare\_conditions

ORDER BY flight\_id, fare\_conditions

)

SELECT flight\_id

FROM (

SELECT \*,

--добавляем колонку значений минимальной стоимости бизнес класса для рейса

lag(amount, 1, amount) OVER(PARTITION BY flight\_id) business\_amount

FROM amount\_city\_cte) q

WHERE amount > business\_amount;

--Результат нулевой. Значит не было

-- 8. Между какими городами нет прямых рейсов?

-- Решение без представлений

-- Для начала находим всевозможные сочетания между аэропортами разных городов

with city\_city\_decart\_cte as (

select distinct

lower(a.city)||' - '||lower(q.city) flight\_name,

a.city city\_1, a.airport\_code airport\_code\_1, q.city city\_2, q.airport\_code airport\_code\_2

from

bookings.airports a

cross join (select airport\_code, city from bookings.airports) q

where

a.airport\_code != q.airport\_code and a.city != q.city

order by flight\_name),

-- Находим фактические сочетания между аэропортами разных городов

city\_city\_fact\_cte as (

select distinct

lower(city\_1)||' - '||lower(city) flight\_name,

city\_1, airport\_code\_1, city city\_2, airport\_code\_2

from (

select city city\_1, q1.departure\_airport airport\_code\_1, q1.arrival\_airport airport\_code\_2

from bookings.airports

join (

select departure\_airport, arrival\_airport

from bookings.flights) q1

on departure\_airport = airport\_code) q2

join bookings.airports on airport\_code\_2 = airport\_code

order by flight\_name)

-- Находим рейсы не вошедшие в декартово сочетание

-- Это и будут варианты, между которыми нет сообщений

select initcap(flight\_name) flight\_name

from city\_city\_decart\_cte

except select initcap(flight\_name) from city\_city\_fact\_cte

order by 1;

-- 8. Вариант с представлениями

-- Для начала находим возможные декартовы сочетания между аэропортами разных городов

create view city\_city\_decart\_view as

select distinct

lower(a.city)||' - '||lower(q.city) flight\_name,

a.city city\_1, a.airport\_code airport\_code\_1, q.city city\_2, q.airport\_code airport\_code\_2

from

bookings.airports a

cross join (select airport\_code, city from bookings.airports) q

where

a.airport\_code != q.airport\_code and a.city != q.city

order by flight\_name;

-- Находим фактические сочетания между аэропортами разных городов

create view city\_city\_fact\_view as

select distinct

lower(city\_1)||' - '||lower(city) flight\_name,

city\_1, airport\_code\_1, city city\_2, airport\_code\_2

from (

select city city\_1, q1.departure\_airport airport\_code\_1, q1.arrival\_airport airport\_code\_2

from bookings.airports

join (

select departure\_airport, arrival\_airport

from bookings.flights) q1

on departure\_airport = airport\_code) q2

join bookings.airports on airport\_code\_2 = airport\_code

order by flight\_name;

-- Находим рейсы не вошедшие в декартово сочетание

-- Это и будут варианты, между которыми нет сообщений

select initcap(flight\_name) flight\_name from city\_city\_decart\_view

except select initcap(flight\_name) flight\_name from city\_city\_fact\_view

order by 1;

-- Другой вариант

select city\_1, city\_2 from city\_city\_decart\_view

except select city\_1, city\_2 from city\_city\_fact\_view

order by city\_1, city\_2;

9.Вычислите расстояние между аэропортами, связанными прямыми рейсами,

сравните с допустимой максимальной дальностью перелетов в самолетах, обслуживающих эти рейсы

-- Используем ранее созданное представление city\_city\_fact\_view. На основе него создаем cte.

-- Можно сделать и без ранее созданного представления, просто тогда cte будет включать в себя его логику

-- В итоговом запросе добавляем информацию из таблицы aircraft

WITH initial\_data AS -- получаем исходные данные для дальнейшей работы

(

SELECT

initcap(flight\_name) flight\_name,

airport\_code\_1, a1.longitude lg1, a1.latitude lt1,

airport\_code\_2, a2.longitude lg2, a2.latitude lt2

FROM

city\_city\_fact\_view c

JOIN airports a1 ON c.airport\_code\_1 = a1.airport\_code

JOIN airports a2 ON c.airport\_code\_2 = a2.airport\_code

),

dist\_cte\_rad AS --получаем расстояние в радианах

(

SELECT \*,

acos(sind(lt1)\*sind(lt2) + cosd(lt1)\*cosd(lt2)\*cosd(lg1 - lg2)) dist\_rad

FROM initial\_data

),

dist\_cte\_km AS --получаем расстояние в км

(

SELECT

flight\_name,

airport\_code\_1, lg1, lt1,

airport\_code\_2, lg2, lt2,

round(6371\*dist\_rad::NUMERIC, 0) dist\_km

FROM dist\_cte\_rad

)

SELECT DISTINCT

flight\_name, dist\_km,

"range",

CASE

WHEN "range" >= dist\_km THEN 'Успех'

ELSE 'Провал'

END Результат

FROM

dist\_cte\_km

JOIN bookings.flights f --добавляем информацию из bookings.aircrafts

ON airport\_code\_1 = f.departure\_airport

and airport\_code\_2 = f.arrival\_airport

JOIN bookings.aircrafts a USING (aircraft\_code)

ORDER BY flight\_name;

Вариант без представления

WITH city\_city\_fact\_cte AS -- Находим фактические сочетания между аэропортами разных городов

(

SELECT DISTINCT

lower(city\_1)||' - '||lower(city) flight\_name,

city\_1, airport\_code\_1, city city\_2, airport\_code\_2

from (

select city city\_1, q1.departure\_airport airport\_code\_1, q1.arrival\_airport airport\_code\_2

from bookings.airports

join (

select departure\_airport, arrival\_airport

from bookings.flights) q1

on departure\_airport = airport\_code) q2

join bookings.airports on airport\_code\_2 = airport\_code

order by flight\_name

),

initial\_data AS -- получаем исходные данные для дальнейшей работы

(

SELECT

initcap(flight\_name) flight\_name,

airport\_code\_1, a1.longitude lg1, a1.latitude lt1,

airport\_code\_2, a2.longitude lg2, a2.latitude lt2

FROM

city\_city\_fact\_cte c

JOIN airports a1 ON c.airport\_code\_1 = a1.airport\_code

JOIN airports a2 ON c.airport\_code\_2 = a2.airport\_code

),

dist\_cte\_rad AS --получаем расстояние в радианах

(

SELECT \*,

acos(sind(lt1)\*sind(lt2) + cosd(lt1)\*cosd(lt2)\*cosd(lg1 - lg2)) dist\_rad

FROM initial\_data

),

dist\_cte\_km AS --получаем расстояние в км

(

SELECT

flight\_name,

airport\_code\_1, lg1, lt1,

airport\_code\_2, lg2, lt2,

round(6371\*dist\_rad::NUMERIC, 0) dist\_km

FROM dist\_cte\_rad

)

SELECT DISTINCT

flight\_name, dist\_km,

"range",

CASE

WHEN "range" >= dist\_km THEN 'Успех'

ELSE 'Провал'

END Результат

FROM

dist\_cte\_km

JOIN bookings.flights f --добавляем информацию из bookings.aircrafts

ON airport\_code\_1 = f.departure\_airport

and airport\_code\_2 = f.arrival\_airport

JOIN bookings.aircrafts a USING (aircraft\_code)

ORDER BY flight\_name;