**Проектная работа Латыповой Полины**

**по модулю**

**“SQL и получение данных”**

**1. Тип подключения**

В работе использовался локальный тип подключения: база была развернута из \*.backup файла.

|  |
| --- |
|  |
| Рис.1 Тип подключения |

**2. ER-диаграмма**

Скриншот ER-диаграммы из DBeaver`a согласно Вашего подключения.

|  |
| --- |
| Рис. 2. Скриншот ER-диаграммы из DBeaver`a согласно подключения. |

**3. Краткое описание БД**

База данных состоит из:

- таблиц aircrafts, airports, boarding\_passes, bookings, flights, seats, ticket\_flights, tickets

- представления flights\_v, routes

**4. Развернутый анализ БД**

Развернутый анализ БД: описание таблиц, логики, связей и бизнес области (частично можно взять из описания базы данных, оформленной в виде анализа базы данных). Бизнес задачи, которые можно решить, используя 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. Пассажир не является отдельной сущностью. С данными пассажиров отдельной таблицы нет. Имя и номер документа пассажира могут меняться с течением времени, поэтому невозможно однозначно найти все билеты одного человекаю Считаем, что все пассажиры уникальны. Билет включает один или несколько перелетов (ticket\_flights). Несколько перелетов могут включаться в билет, если 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**

Список отношений:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| aircrafts | таблица | Аэропорты |
| boarding\_passes | таблица | Посадочные талоны |
| bookings | таблица | Бронирования |
| flights | таблица | Рейсы |
| flights\_v | представление | Рейсы |
| routes | мат. представление | Маршруты |
| seats | таблица | Места |
| ticket\_flights | таблица | Перелеты |
| tickets | таблица | Билеты |

**Таблицы**

Таблица bookings.aircrafts

Каждая модель воздушного судна идентифицируется трехзначным кодом (aircraft\_code). Указывается название модели (model), 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. Указывается широта (longitude), долгота (latitude), часовой пояс (timezone).

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип | Модификаторы | Описание |
| aircraft\_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 |
| longitude | float | NOT NULL | Координаты аэропорта: широта |
| timezone | text | NOT NULL | Временная зона |

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 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 | timestamptz | 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 |

Столбец | Тип | Модификаторы | Описание ---------------------+-------------+--------------+----------------------------- flight\_id | serial | NOT NULL | Идентификатор рейса flight\_no | char(6) | NOT NULL | Номер рейса scheduled\_departure | timestamptz | NOT NULL | 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 |

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 |
| fare\_conditions | varchar(10) | NOT NULL | Класс обслуживания |

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 | timestamptz | Время вылета по расписанию, местное время в пункте отправления |
| scheduled\_arrival | imestamptz | Время прилёта по расписанию |
| scheduled\_arrival\_local | imestamptz | Время прилёта по расписанию, местное время в пункте отправления |
| 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 | timestamptz | 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) | 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 |

База данных позволяет решать различные задачи: например, узнать, какие аэропорты самые нагруженные, а какие – наименее нагруженные. Узнать, между какими городами больше рейсов, между какими городами их нет, сделать предположения о том, какие направления могут нуждаться в дополнительных или новых рейсах. Так же можно узнать, какие существующие рейсы оправданы (летает много пассажиров), а какие, возможно, стоит сократить (мало пассажиров). Также можно узнать, правильно ли используются типы самолетов – летают ли они на расстояния, для них предназначенные, или нет.

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

5.1. В каких городах больше одного аэропорта?

Запрос:

**select** city **from** bookings.airports **group** **by** city **having** **count**(city) > 1

Результат:

1. Ульяновск

2. Москва

Описание:

Из таблицы airports выбираем значения колонки city, группируя по одинаковым значениям city, оставляя только те значения, которые повторяются больше одного раза. Для проверки, или чтобы узнать, сколько аэропортов в городе, можем добавить в список выводимых колонок поле **count**(city).

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

В решении обязательно должно быть использован подзапрос.

Запрос:

**select** airport\_name, airport\_code **from** bookings.airports

**where** airport\_code **in**

(**select** **distinct** departure\_airport **from** bookings.flights f

**where** f.aircraft\_code = (**select** a.aircraft\_code **from** bookings.aircrafts a **where** a."range" = (**select** **max**(**range**) **from** bookings.aircrafts)))

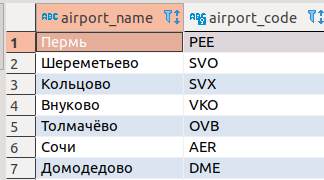
**or**

airport\_code **in**

(**select** **distinct** arrival\_airport **from** bookings.flights f

**where** f.aircraft\_code = (**select** a.aircraft\_code **from** bookings.aircrafts a **where** a."range" = (**select** **max**(**range**) **from** bookings.aircrafts)))

Результат:



Описание:

Для того, чтобы найти самолет с максимальной дальностью полета, используется запрос к таблице aircrafts с функцией max(range), результат этого запроса используется для поиска кода смаолета соответствующей дальностью полета. Код самолета используется в запросе к таблице flights для того, чтобы найти все коды аэропортов, из которых и в которые совершался вылет на соответстующем самолете. Для того, чтобы определить код и навзвание аэропортов, результаты указанных выше запросов используются в запросе к таблице airports.

5.3. Вывести 10 рейсов с максимальным временем задержки вылета

В решении обязательно должно быть использовано: Оператор LIMIT.

Запрос:

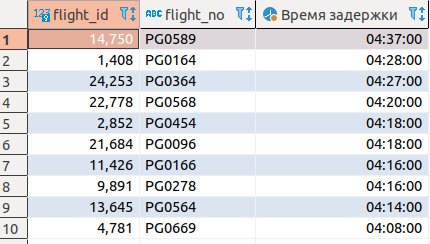
**select** flight\_id, flight\_no,(actual\_departure-scheduled\_departure)

**as** "Время задержки" **from** bookings.flights

**where** (actual\_departure-scheduled\_departure) **is** **not** **null**

**order** **by** "Время задержки" **desc** **limit** 10

Результат:



Описание:

Для того, чтобы определить время задержки рейса, используем разницу между значениям в колонках actual\_departure и scheduled\_departure (выводим как новую колонку “Время задержки”). Для того, чтобы показать, какие именно рейсы имели максимальное время задержки, используем колонки flight\_id, flight\_no. В выводимом результате используем условие where (actual\_departure-scheduled\_departure) is not null, чтобы отсеять нерелевантные результаты со значением NULL. Так же используем order by "Время задержки", чтобы упорядочить результаты по убыванию, а также desc limit 10, чтобы выбрать первые 10 результатов.

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

Запрос:

**select** b.book\_ref, t.ticket\_no, bp.ticket\_no **from** bookings.bookings b

**inner** **join** bookings.tickets t

**on** b.book\_ref = t.book\_ref

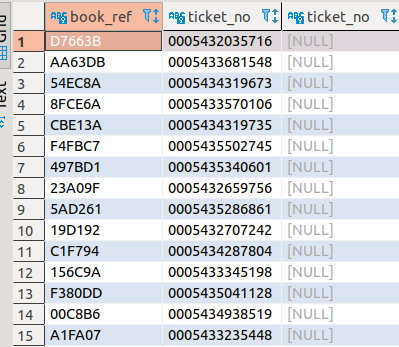
**full** **outer** **join** bookings.boarding\_passes bp

**on** t.ticket\_no = bp.ticket\_no

**where** bp.ticket\_no **is** **null**

Результат:

Да, такие брони были.



Описание:

Связь между таблицей бронирований bookings и таблицей посадочных талонов boarding\_passes можно установить через таблицу tickets. Для этого объединяем bookings и tickets с помощью inner join по равенству ключа book\_ref, так как нам нужны только те записи, которые имеют не null значения для book\_ref в обоих таблицах. Затем объединяем полученный результат с boarding\_passes с помощью full outer join по ключу ticket\_no и дополнительному условию bp.ticket\_no is null, чтобы вывести только те результаты, в которых в таблице boarding\_passes нет номеров билетов, соответствующих номерам билетов из объединения таблиц tickets и bookings. Другими словами, из финального результата, который объединяет все возможные записи, мы оставляем только те, для которых поля в таблице boarding\_passes пусты.

5.5. 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 обязательно должно быть использовано:

- Оконная функция

- Подзапросы или cte

Запрос:

**WITH** seats\_in\_flights **AS**

(**select** arcr.**count** **as** used\_seats, arcr.flight\_id, arcr.aircraft\_code, arcr.actual\_departure, arcr.flight\_no,arcr.departure\_airport,(seats\_arc.seat\_amount-arcr.**count**) **as** unused\_seats, seats\_arc.seat\_amount, **ROUND**(**CAST**((seats\_arc.seat\_amount-arcr.**count**)\*100 **as** **FLOAT**)/**CAST**(seats\_arc.seat\_amount **as** **FLOAT**)) **as** seat\_percentage **from**

(**select** **count**(sts.seat\_no),sts.flight\_id, sts.actual\_departure, sts.flight\_no,sts.aircraft\_code, sts.departure\_airport **from**

(**select** f.flight\_id,f.actual\_departure,f.flight\_no, f.departure\_airport, bp.ticket\_no, bp.boarding\_no,f.aircraft\_code, bp.seat\_no **from** bookings.boarding\_passes bp

**left** **join** bookings.flights f

**on** bp.flight\_id = f.flight\_id

**join** bookings.aircrafts a

**on** f.aircraft\_code = a.aircraft\_code

**order** **by** f.flight\_id, bp.ticket\_no) **as** sts

**group** **by** sts.flight\_id,sts.actual\_departure,sts.flight\_id, sts.flight\_no,sts.aircraft\_code, sts.departure\_airport) **as** arcr

**left** **join**

(**select** **count**(seat\_no) **as** seat\_amount, aircraft\_code **from** bookings.seats

**group** **by** aircraft\_code) **as** seats\_arc

**on** seats\_arc.aircraft\_code = arcr.aircraft\_code

**order** **by** arcr.departure\_airport, arcr.actual\_departure, arcr.flight\_no, arcr.flight\_id)

**select**

flight\_id **as** "Id рейса",

flight\_no **as** "Номер рейса",

departure\_airport **AS** "Аэропорт отправления",

actual\_departure **as** "Время отправления",

**DATE**(actual\_departure) **AS** "Дата отправления",

seat\_amount **as** "Кол-во мест",

unused\_seats **as** "Кол-во неиспользованных мест",

used\_seats **as** "Кол-во использованных мест",

seat\_percentage **as** "Незаполненность самолета, %",

**SUM**(used\_seats) **OVER** (**PARTITION** **BY** departure\_airport, **DATE**(actual\_departure) **ORDER** **by** departure\_airport, actual\_departure **rows** **between** **unbounded** **preceding** **and** **current** **row**) **AS** "Вылетевшие из аэропорта пассажиры"

**from** seats\_in\_flights

Результат:



Описание:

Для выполнения запроса использованы таблицы boarding\_passes, seats, flights, aircrafts.

Первый подзапрос, резульатат которого имеет псевдоним arcr, позволяет получить информацию о том, какие места были заняты во время рейса, а также дополнительную инфморацию о рейсе. Вывод делается на основании выданных посадочных талонов.

К результату выполняется left join с таблицей seats по ключу aircraft\_code. Это позволит узнать, сколько на рейсе изначально было мест (по самолету).

Дается псеводним arcr.

Из полученного резульатата выбираются существующие и вычисляемые колонки:

used\_seats, arcr.flight\_id, arcr.aircraft\_code, arcr.actual\_departure, arcr.flight\_no,arcr.departure\_airport,(seats\_arc.seat\_amount-arcr.**count**) **as** unused\_seats, **ROUND**(**CAST**((seats\_arc.seat\_amount-arcr.**count**)\*100 **as** **FLOAT**)/**CAST**(seats\_arc.seat\_amount **as** **FLOAT**)) **as** seat\_percentage.

Результат упорядочивается по arcr.departure\_airport, arcr.actual\_departure, arcr.flight\_no, arcr.flight\_id.

В результате получаем новые колонки – unused\_seats – количество неиспользованных мест (разница меджду полным количеством мест и количеством занятых мест, полученных в первом запросе по посадочным талонам), seat\_percentage (незаполненность самолета на рейсе в процентах, вычисленная как результат от деления разницы между количеством мест и количеством заполненных мест на количество мест в самолете и умножнное на 100. Результат округлен до целых значений.

Результат этих запросов используется как cte с именем seats\_in\_flights.

Из него выбираются существующие и вычисляемые колонки. Новая колонка - “Вылетевшие из аэропорта пассажиры“ - вычисляется с помощью оконной функции, цель которой – вычислить количество пассажиров, улетевших в конкретную дату (DATE(actual\_departure)) из конкретного аэропорта (departure\_airport), причем это должна быть кумулятивная сумма, накапливаемая с каждым новым рейсом в этот день из этого аэропорта. Для этого использовано следующее выражение:

**SUM**(used\_seats) **OVER** (**PARTITION** **BY** departure\_airport, **DATE**(actual\_departure) **ORDER** **by** departure\_airport, actual\_departure **rows** **between** **unbounded** **preceding** **and** **current** **row**) **AS** "Вылетевшие из аэропорта пассажиры".

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

В решении обязательно должно быть использовано:

- Подзапрос

- Оператор ROUND

Запрос:

**select** fla.aircraft\_code, **ROUND**(100\*fla."Количество полетов по типу самолета"/fla."Общее количество полетов", 2) **as** "Доля полетов по типу самолета, %"

**from**

(**select** f.aircraft\_code,

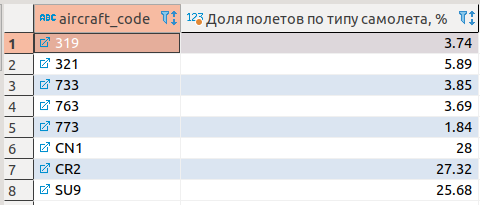
**SUM**(**count**(\*)) **OVER** (**PARTITION** **BY** aircraft\_code) **as** "Количество полетов по типу самолета",

**SUM**(**count**(\*)) **over** () **as** "Общее количество полетов"

**from** bookings.flights f

**group** **by** f.aircraft\_code) **as** fla

Результат:



Описание:

В подзапросе к таблце flights вычисляется с помощью оконных функций общее число полетов и число полетов по типам самолетов. На основе результата этого запроса вычисляется доля полетов по типам самолета от общего числа полетов (на основе колонок "Количество полетов по типу самолета" и "Общее количество полетов") с применением функции ROUND – получается колонка "Доля полетов по типу самолета, %".

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

В решении обязательно должно быть использовано:

-cte

Запрос:

**select** airp\_city.city, tckts.arrival\_airport, tckts."Business-economy difference"

**from**

(**with** business\_economy\_df

**as**

(**select** business.arrival\_airport, economy.max\_economy,business.min\_business, business.min\_business - economy.max\_economy **as** "Business-economy difference"

**from**

(**select** ec.arrival\_airport, **max**(ecc.economy\_payment) **as** max\_economy **from** bookings.flights ec

**inner** **join**

(**select** flight\_id, fare\_conditions, amount **as** economy\_payment

**from** bookings.ticket\_flights

**where** fare\_conditions = 'Economy') **as** ecc

**on** ecc.flight\_id = ec.flight\_id

**group** **by** ec.arrival\_airport) economy

**inner** **join**

(**select** ec.arrival\_airport, **min**(ecc.business\_payment) **as** min\_business **from** bookings.flights ec

**inner** **join**

(**select** flight\_id, fare\_conditions, amount **as** business\_payment

**from** bookings.ticket\_flights

**where** fare\_conditions = 'Business') **as** ecc

**on** ecc.flight\_id = ec.flight\_id

**group** **by** ec.arrival\_airport) business

**on** business.arrival\_airport = economy.arrival\_airport

**where** business.min\_business - economy.max\_economy < 0)

**select** business\_economy\_df.arrival\_airport, business\_economy\_df.max\_economy,business\_economy\_df.min\_business, business\_economy\_df."Business-economy difference"

**from** business\_economy\_df) **as** tckts

**inner** **join**

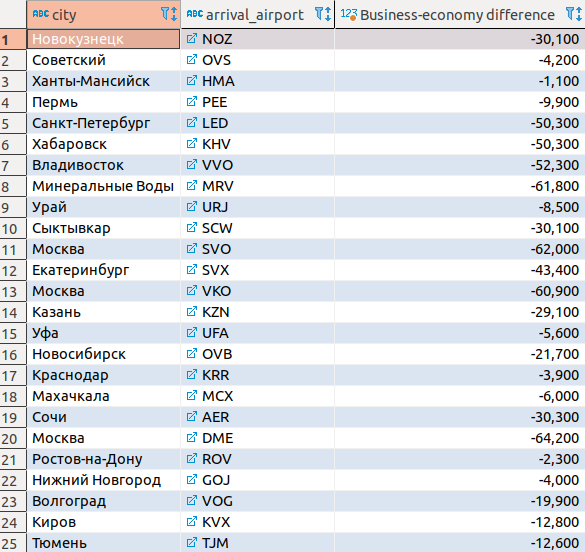
(**select** b.airport\_code, b.city

**from**

bookings.airports b) **as** airp\_city

**on** airp\_city.airport\_code = tckts.arrival\_airport

Результат:



Описание:

Проверка показала , что билеты бизнес-класса всегда дороже билетов эконом-класса в рамках одного перелета. Запрос проверки:

**select** business\_p.flight\_id, business\_p.business\_payment, economy\_p.economy\_payment, business\_p.business\_payment-economy\_p.economy\_payment

**from**

(**select** flight\_id, amount **as** business\_payment **from** bookings.ticket\_flights tf

**where** fare\_conditions = 'Business' **group** **by** flight\_id, fare\_conditions, amount) **as** business\_p

**inner** **join**

(**select** flight\_id, amount **as** economy\_payment **from** bookings.ticket\_flights tf

**where** fare\_conditions = 'Economy' **group** **by** flight\_id, fare\_conditions, amount) **as** economy\_p

**on** business\_p.flight\_id = economy\_p.flight\_id

**where** (business\_p.business\_payment-economy\_p.economy\_payment) < 0

**order** **by** flight\_id.

Таких записей нет.

Поэтому была проведена проверка, можно ли найти среди всех перелетов в каждый город два таких, для которых бизнес-класс будет дешевле экономического. При этом не важно, откуда вылетал самолет и когда это происходило. Был написан указанный в пункте “запрос” запрос. Он использует представление ticket\_flights. Для этого представления написаны два запроса. Один из них вычисляет минимальное значение для цены билета бизнес-класса, группируя полеты по городам, которые являются конечным пунктом. Второй запрос вычисляет максимальное значение для цены билета эконом-класса. Запросы обхединяются по ключу airport\_code. Далее вычисляется разница между максимальной ценой на билет эконом-класса и минимальной ценой на билет бизнес-класса по городам. Оставляются только те записи, где цена на билет эконом-класса больше цены на билет бизнес-класса. Далее выполняется inner join с таблицей airports, чтобы по коду аэропорта узнать название города, в котором он располагается. В итоге получаем 25 записей, что говорит о том, что да, такие города были.

5.8. Между какими городами нет прямых рейсов?

В решении обязательно должно быть использовано:

- Декартово произведение в предложении FROM  
- Самостоятельно созданные представления (если облачное подключение, то без представления)

- Оператор EXCEPT

Запрос:

Создается представление, чтобы показать все возможные сочетания городов вылета и прилета (кроме совпадающих):

**CREATE** **VIEW** bookings.CitiesPossibleConnection

**AS**

**select** arp.airport\_code **as** departure\_airport , arp.city **as** departure\_city , arp2.airport\_code **as** arrival\_airport, arp2.city **as** arrival\_city **from** bookings.airports arp **cross** **join** bookings.airports arp2

**except**

(**select** arp.airport\_code **as** departure\_airport , arp.city **as** departure\_city , arp2.airport\_code **as** arrival\_airport, arp2.city **as** arrival\_city **from** bookings.airports arp **cross** **join** bookings.airports arp2

**where** arp.airport\_code = arp2.airport\_code)

**order** **by** departure\_airport

Затем создается запрос, чтобы узнать, между какими городами нет прямых перелетов. Запрос создается к представлению и таблице flights:

**select** \* **from**

(**select** departure\_city, arrival\_city **from**

(**select** f.departure\_airport **as** city\_a, f.arrival\_airport **as** city\_b **from** bookings.flights f

**group** **by** f.departure\_airport, f.arrival\_airport) **as** part1

**right** **join** bookings.CitiesPossibleConnection cc

**on** cc.arrival\_airport = part1.city\_a **and** cc.departure\_airport = part1.city\_b

**where** city\_a **is** **null** **and** city\_b **is** **null**

**union** **all**

**select** departure\_city, arrival\_city **from**

(**select** f.departure\_airport **as** city\_a, f.arrival\_airport **as** city\_b **from** bookings.flights f

**group** **by** f.departure\_airport, f.arrival\_airport) **as** part1

**right** **join** bookings.CitiesPossibleConnection cc

**on** cc.arrival\_airport = part1.city\_b **and** cc.departure\_airport = part1.city\_a

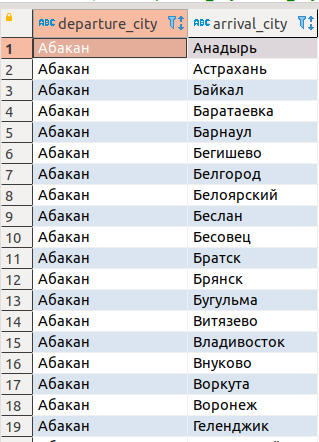
**where** city\_a **is** **null** **and** city\_b **is** **null**) **as** pairs

**group** **by** pairs.departure\_city, pairs.arrival\_city

**order** **by** pairs.departure\_city, pairs.arrival\_city

Результат:

9738 записи – сочетаний городов, между которомыи нет прямых рейсов. 516 сочетаний, между которыми есть прямые рейсы.



Описание:

Создается представление CitiesConnections , чтобы выявить все возможные пары городов, в которых есть аэропорты, на основе таблицы airports. Исключаются пары, в которых оба названия пренадлежат одному городу. Далее создается запрос к таблице flights, и осуществляется right join к CitiesConnections, чтобы понять, для каких пар из представления нет соответствующих пар в таблице flights. Возможно, излишне делать этот запрос два раза для того, чтобы убедиться, что если нет рейса в одну стороноу, то нет и в другую (однако здесь это делается). Результаты объединяются, группируются, и остаются только те пары, для которых в flights не нашлось соответствия (where city\_a is null and city\_b is null).

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

работаете, как с массивом. В локальной базе координаты находятся в столбцах airports.longitude и airports.latitude.

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

d = arccos {sin(latitude\_a)·sin(latitude\_b) + cos(latitude\_a)·cos(latitude\_b)·cos(longitude\_a - 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·R, где R = 6371 км — средний радиус земного шара.

Запрос:

Создадим представление, показывающее города, соединенные прямыми рейсами:

**create** **view** bookings.CitiesConnectedByFlights **as**

(**select** \* **from**

(**select** departure\_city, arrival\_city **from**

(**select** f.departure\_airport **as** city\_a, f.arrival\_airport **as** city\_b **from** bookings.flights f

**group** **by** f.departure\_airport, f.arrival\_airport) **as** part1

**right** **join** bookings.CitiesPossibleConnection cc

**on** cc.arrival\_airport = part1.city\_a **and** cc.departure\_airport = part1.city\_b

**where** city\_a **is** **not** **null** **and** city\_b **is** **not** **null**

**union** **all**

**select** departure\_city, arrival\_city **from**

(**select** f.departure\_airport **as** city\_a, f.arrival\_airport **as** city\_b **from** bookings.flights f

**group** **by** f.departure\_airport, f.arrival\_airport) **as** part1

**right** **join** bookings.CitiesPossibleConnection cc

**on** cc.arrival\_airport = part1.city\_b **and** cc.departure\_airport = part1.city\_a

**where** city\_a **is** **not** **null** **and** city\_b **is** **not** **null**) **as** pairs

**group** **by** pairs.departure\_city, pairs.arrival\_city

**order** **by** pairs.departure\_city, pairs.arrival\_city)

Постороим запрос для других задач:

**select** arp\_data.departure\_city,arp\_data.arrival\_city,arp\_data.distance, arp\_data.**range**,

**case**

**when** (arp\_data.**range**-arp\_data.distance)/arp\_data.**range** < 0.5 **then** 'Разница меньше 50%'

**when** (arp\_data.**range**-arp\_data.distance)/arp\_data.**range** > 0.5 **then** 'Разница больше 50%'

**end**

**as** "difference"

**from**

(**select** arp\_geo.departure\_city, arp\_geo.arrival\_city,arp\_geo.distance, flight\_ranges.**range**

**from**

(**with** airports\_geodata **as** (

**select** departure\_city, arrival\_city,

a.airport\_code **as** departure\_airport\_code, a.longitude **as** departure\_longitude, a.latitude **as** departure\_latitude,

aa.airport\_code **as** arrival\_airport\_code, aa.longitude **as** arrival\_longitide, aa.latitude **as** arrival\_latitude

**from** CitiesConnectedByFlights

**inner** **join** bookings.airports a

**on** a.city = CitiesConnectedByFlights.departure\_city

**inner** **join** bookings.airports aa

**on** aa.city = CitiesConnectedByFlights.arrival\_city)

**select** airports\_geodata.departure\_city, airports\_geodata.arrival\_city,airports\_geodata.departure\_airport\_code,airports\_geodata.departure\_longitude,airports\_geodata.departure\_latitude,airports\_geodata.arrival\_airport\_code,airports\_geodata.arrival\_longitide,airports\_geodata.arrival\_latitude,

**ROUND**(**radians**(**acosd**(**sind**(arrival\_latitude)\***sind**(departure\_latitude) + **cosd**(arrival\_latitude)\***cosd**(departure\_latitude)\***cosd**(arrival\_longitide - departure\_longitude)))\*6371) **as** distance

**from** airports\_geodata) **as** arp\_geo

**inner** **join**

(**select** fv.departure\_city, fv.arrival\_city,(a.**range**) **from** bookings.flights\_v fv

**left** **join**

bookings.aircrafts a

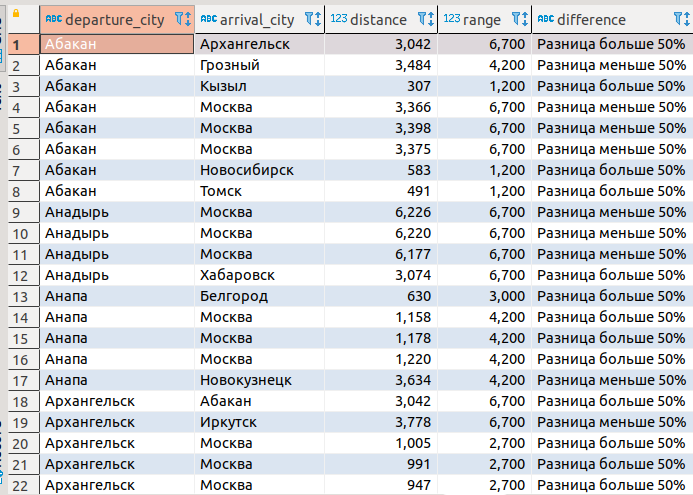
**on** fv.aircraft\_code = a.aircraft\_code

**group** **by** fv.departure\_city, fv.arrival\_city,a.**range**

**order** **by** departure\_city,arrival\_city) **as** flight\_ranges

**on** flight\_ranges.departure\_city = arp\_geo.departure\_city **and** flight\_ranges.arrival\_city = arp\_geo.arrival\_city) **as** arp\_data

Результат:



Описание:

Внутри основного запроса создаем запрос, объединяющий данные от CitiesConnectedByFlights и bookings.airports, чтобы получить данные о географическом месте положения аэропортов вылета и прилета. Результат объединяется с данными из aircrafts, чтобы узнать, на какое расстояние могут летать самолеты для каждого рейса. Так же вычисляется расстояние между аэропортами по указанной формуле. Далее вычисляется разница между реальным расстоянием между аэропортами и допустимой дальностью полета для самолета. Разница кодиурется с помощью оператора CASE, и получаются варианты 'Разница меньше 50%', 'Разница больше 50%'.