

Проектная работа по модулю “SQL и получение данных”



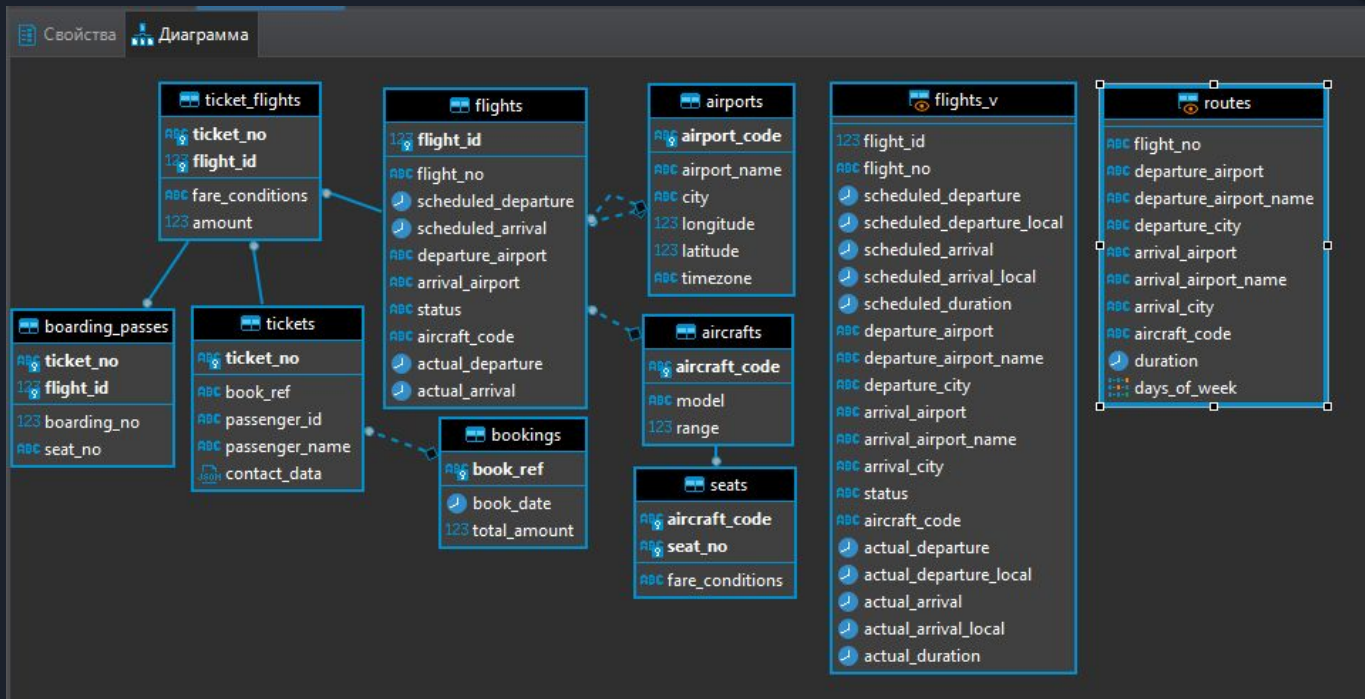
В работе использовался локальный тип подключения. Использовал самую маленькую базу что показывает размер таблиц.



postgres 3 - localhost:5432

postgres	
Схемы	
bookings	
Таблицы	
aircrafts	32K
airports	64K
boarding_passes	80M
bookings	18M
flights	4,8M
seats	136K
ticket_flights	108M
tickets	59M
Представления	
Мат. представления	
Индексы	
Функции	
Последовательности	
Типы данных	

Скриншот ER-диаграммы из DBeaver



Краткое описание БД - из каких таблиц и представлений состоит.

Во первых замечательно проработанная База Данных “Авиаперевозки”

Она состоит из восьми таблиц и двух представлений.

В качестве предметной области выбраны авиаперевозки по России.

Содержащиеся таблицы: aircrafts, airports, boarding_passes, bookings , flights, seats ticket_flights , tickets.

Представления : flights_v , routes.



Развернутый анализ БД .



И так. В качестве предметной области выбраны авиаперевозки по России, а это очень хорошо потому что учиться на реальных данных или очень приближенных к реальности данных куда интереснее чем на искусственных, которые делают “машины”.

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.

Сама бд даже при такой малой компановке таблиц, может дать ответы на многи и многи бизнес вопросы, может подсказать что может увеличить прибыль, предупредить о излишних задержках, о росте числа пассажиров в данном направлении и т.д. об этом будет отдельный слайд который будет ниже.

Объекты схемы

Есть наглядная таблица, которая во первых показывает где у нас представления, во вторых указывает на размер таблицы, также видно описание с переводом. А вот размер в “kB” указывает на таблицы в к которым прежде чем писать запрос, нужно делать это с оглядкой. Таким образом это ещё одна приятная черта этой бд. Что таблицы имеют очень разный размер таблиц.

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




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).

Таблица bookings.airports - Аэропорт идентифицируется трехбуквенным кодом (airport_code) и имеет свое имя (airport_name).

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.

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).




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).




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.

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).

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_data).



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.


Вопросы который я бы задал для Базы Данных “Авиаперевозки”.

1. Проверить процент задержек рейсов ,взять и сравнить их с базами других авиаперелетов, основываясь на купленных данных или же самим отслеживать конкурентов, банально эту инфу может собирать скрипт с сайтов других компаний и т.д. Потому как задержки рейсов могли повлиять на имидж компании. Можно одинажды выяснить процент отклонения на задержках ,например 20 задержек в месяц это норма а так же и время задержки до 5 минут не учитывать. Вот такого стандарта и придерживаться. Этим мы сможем зарекомендовать себя особенно для бизнес класса, который часто не может позволить себе опоздание.






2. Посмотреть стоит ли открывать рейс прямого пути для городов которые между которыми 2 и более пересадки, просто посмотрев на то сколько людей прилетает туда из пункта "А" минуя по пути две остановки до пункта "Б", и это число должно быть таковым, что в месяц должна набираться рентабельность полётов примерно 2-3 вылета (но это уже скажет специалист) так что пока остановимся на получение чисел "какой процент людей прилетают с пересадками от начало и до конца" Если это составляет 80%+-20% и если это есть хорошая рентабельность. То мы смело можем наблюдать за этими направлениями собирать информацию и делать заключительный вывод об открытии прямых рейсов. Но мы также должны сразу исключить рейсы до которых самолёты не смогут долететь.




3. Во первых я понимаю что пассажиры у нас все одинаковые, но мы можем классифицировать людей, возраст, пол, рост, вес и т.д. Посмотреть какой класс какие места предпочитает, и выдавать человеку попавшему под класс, те билеты, в которых место которое он хотел бы получить. Для чего? Первое чтобы задобрить клиента. - Это даст правильную расстановку в салоне, которая помимо места даст так же нам группы лиц которые например все летит и спит в одном направлении, а другая группа лиц летит и читает, третья очень общительные и всегда говорят в полёте. Это ли не лучший выбор для того чтобы рассадить людей? А какие потом будут впечатления о полете? только самые превосходные, и мне кажется это даст больше положительных отзывов о компании. Второе тоже возможно, чтобы поднять цены на это место именно для человека место к которому он привык в самолёте (любит). (Мы платим за то к чему привыкли)



4. Сколько самолётов улетело с загрузкой не рентабельной , допустим загрузка менее 10 % нерентабельна. Так вот и задача , сократить расходы, с вылетом менее 10 % если это прямой рейс мы его убираем на эти дни. То есть подготовить и найти зависимости от времени года и предугадывать сколько людей будет лететь или не будет, таким образом мы можем динамически в режиме реального времени строить план перелетов. Но если это рейс с несколькими пересадками и его загруженность(прибыль) будет рентабельной, то мы оставляем рейс, это что то похожее на маршрутки которые катаются по городу .

5. Сделать обратное 4 заданию . Посмотреть где у нас улетали полные самолеты и следуют нескольким пересадкам, и те самолеты которые прилетают с теми же пассажирами до конечной точки, то значит тут нужно задуматься. Нужны ли нам вообще остановки, или мы добавляем прямой рейс если это возможно.






6. Смотреть за ростом рейсов в одном направлении растёт ли число пассажиров, отслеживать выбросы по загрузке самолёта людьми, а также строить прогнозы, и если мы понимаем что в одном направлении идёт рост и он будет по прогнозам через год превышать вместимость самолета то мы заранее меняем или число перелетов или сам самолет с более большим числом мест, так как возможны еще и выбросы которые и этот самолёт будут забивать более 95% . В общем провести оптимизацию самолётов по направлениям.

7. Посмотреть кто больше приносит прибыли бизнес класс или эконом . При каких трудозатратах, человеко часов, рекламы ,инвестиций эта прибыль получается. ? Возможно нам эти данные подскажут что нам нужно срочное развитие в перелёты “Бизнес” класса. Или наоборот скажут о рекламе что “Бизнес” класс нам приносит столько прибыли сколько мы тратим на рекламу.





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

Во первых каждому запросу и описанию логике, понадобится более одной страницы, что бы это было читаемо и удобно воспринимать.

Так же для наглядности я прикрепил скрин того что будет в итоге.

Первой страницей будет логика, второй странице sql запрос





Задание 1

"В каких городах больше одного аэропорта?"

1. Решал так, прочитал задание, затем начал изучать документацию повторно. Обнаружил такую строку "Таблица bookings.airports. Для города не предусмотрено отдельной сущности, но название (city) указывается и может служить для того, чтобы определить аэропорты одного города." Далее для себя изучил таблицу bookings.airports , ручками сделал сортировку по городам увидел что есть повторяющиеся города, их легко увидеть глазами при сортировке руками. Это были города Москва и Ульяновск. Понял что в решение нужны именно они так как количество аэропортов в этих городах более одного.
2. Начал писать запрос к таблице bookings.airports. , в селект решил вывести count(airport_name) и назвал его AS "Кол-во Аэропортов" - которое считает кол-во аэропортов в городе, но нам нужна группировка по городу для выполнения запроса.
3. Далее указана функция GROUP BY airports.city которая группирует то что посчитал count(), и если города при этом повторяются то группировка записывает +1 для этого города, у Москвы 3 аэропорта поэтому в корзину группировки попало 3 аэропорта .
4. Далее это всё я "завернул" в подзапрос , чтобы я мог составить и записать условие, это связано с последовательной логикой выполнения PostgreSQL , и указал условие где полученный ранее столбец "Кол-во Аэропортов" Where более одного аэропорта. "WHERE "Кол-во Аэропортов" > 1".
5. В SELECT я указал city, "Кол-во Аэропортов" , с соблюдением условия "WHERE "Кол-во Аэропортов" > 1" . Получился результат который вы можете наблюдать ниже.



```
SELECT city, "Кол-во Аэрапортов"
```

```
FROM (
```

```
    SELECT city ,count(airport_name) AS "Кол-  
    во Аэрапортов"
```


```
    FROM bookings.airports
```

```
    GROUP BY airports.city) t
```

```
WHERE "Кол-во Аэрапортов" > 1
```




city	Кол-во Аэрапортов
Ульяновск	2
Москва	3



Задание 2

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

1. Решил начать с Join таблицы aircrafts к таблице flights и я смог получить , таблицу которая содержит все необходимые данные как длина перелета которая соответствует модели самолёта это столбец "range" .
2. Далее в запросе (select) я решил присвоить Rank исходя из дистанции полёта и назвать его ranked, и ещё я отсортировал этот ранг по убыванию и таким образом, на первых местах я получил только те аэропорты которые -- самые длинные по продолжительности, и присвоил им первый ранг т.е. максимальному перелету был присвоен ранг 1 (он определился по сортировке и составил 11100 км.) , а уже в него подтянулись Аэропорты которые были на одной строке с range.
3. И так присвоенный ранг для этих колонок являлся первым 1 .Далее я всё это помещаю в подзапрос, и прописываю условие WHERE ranked = 1 а в запрос пишу "SELECT departure_airport" , но таблица получилась большой, с повторяющимися значениями и невидимой глазу.
4. Для удобства решил оставить уникальные значения поэтому прописываю DISTINCT в запрос - "SELECT DISTINCT departure airport" -- Результат вы можете посмотреть ниже.
5. В конечном результате всего 7 строк.



```
SELECT DISTINCT departure_airport
FROM (
    SELECT departure_airport,
    RANGE,RANK () OVER ( order by RANGE DESC) AS ranked
    FROM flights f
    JOIN aircrafts a ON a.aircraft_code = f.aircraft_code ) t
WHERE ranked = 1
```




departure_airport
AER
DME
OVB
PEE
SVO
SVX
VKO



Задание 3

Вывести 10 рейсов с максимальным временем задержки вылета

1. Для решения я обращаюсь к таблице "flights" в запрос я пишу нужные мне столбцы это flight_no номер рейса.
2. Далее в запросе пишу решение которое считает "Дату актуального отправления-Дату отправления " это время покажет нам задержку называю её "aircraft_delay" .
3. Выполняю написанную часть кода что бы посмотреть на промежуточный результат, и далее понимаю что у меня в столбце "aircraft_delay" есть строки со значением "null" такие строки мы конечно же должны отбросить потому что мы не можем их сравнить. Они появились из-за того что в столбце "actual_departure" в некоторых строках есть значения Null и спасибо "Postgre Sql" что всё таки посчитал эти значения отдал нам хотя бы такую таблицу а не выдал ошибку.
4. Далее прописал условия "WHERE actual_departure is not NULL" которое выводит те строки у которых нету значения null, данные пришли в хаотичном виде.
5. Отсортируем наши данные по убыванию ,добавляю сортировку по новому столбцу "aircraft_delay" по убыванию "DESC" то есть, самое большое время задержки появится сверху, и будет убывать вниз.
6. Мне осталось добавить условие задачи то есть воспользоваться оператором LIMIT для 10 записей. "LIMIT 10"
7. Результат вы можете посмотреть ниже.



```
SELECT flight_no, (actual_departure - scheduled_departure) AS aircraft_delay
FROM flights f
WHERE actual_departure is not NULL
ORDER BY aircraft_delay desc
LIMIT 10
```



flight_no	aircraft_delay	
PG0589	04:37:00	
PG0164	04:28:00	
PG0364	04:27:00	
PG0568	04:20:00	
PG0454	04:18:00	
PG0096	04:18:00	
PG0166	04:16:00	
PG0278	04:16:00	
PG0564	04:14:00	
PG0669	04:08:00	

Задание 4

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

1. Тут нужно воспользоваться правильным типом JOIN. Первое что сделал пошел в документацию в поиске написал "Посадочные талоны" нашёл что этому посвящена целая таблица "boarding_passes", изучил таблицу.
2. Далее я понял что мне нужно так же найти таблицу брони, так же пошёл в документацию к БД нашёл там таблицу bookings. Как и написано в документации "Бронирование идентифицируется номером (book_ref, шестизначная комбинация букв и цифр)."
3. Я понял что ответ будет в этих двух таблицах и мне нужно провести join двух таблиц "boarding_passes" и "bookings", но у них нет ключа и уникальных значений для join. Поэтому на помощь мне пришла третья таблица "tickets" которая может соединить и стать промежуточной таблицей между "boarding_passes" и "bookings". К "boarding_passes" она может джойниться по столбцу "ticket_no" а к "bookings" она связана внешним ключом о чем и говорит нам пунктирная линия в ER-диаграмме.
4. Осталось только выбрать верный тип джойна, я решил методом исключения выбрать join. Я выбрал RIGHT OUTER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON, и дополняются записями из второй по порядку (правой) таблицы, даже если они не соответствуют условию. У записей правой таблицы, которые не соответствуют условию, значение столбца из левой таблицы будет NULL. Это и позволит нам максимально обогатить таблицу boarding_passes через таблицу "tickets" количество записей составило 707585 строк.
5. Просматривая вывод я обнаружил что boarding_no содержит запись Null то есть номера посадочного талона нету. Значит нам нужно оставить именно эти записи в которых "WHERE boarding_no is NULL". Так как по условию спрашивалось "Были ли брони" мы оставляем в SELECT t.book_ref AS "Номер бронирования" и "по которым не были получены посадочные талоны" и добавил в select boarding_no AS "Посадочный талон".
6. Результат вы можете посмотреть ниже.

SELECT boarding_no AS "Посадочный талон",t.book_ref AS "Номер бронирования"

FROM boarding_passes bp

RIGHT OUTER JOIN tickets t ON t.ticket_no = bp.ticket_no

RIGHT OUTER JOIN bookings b ON b.book_ref = t.book_ref

WHERE boarding_no is NULL



	Посадочный талон	Номер бронирования	
1	[NULL]	0006C3	
2	[NULL]	001273	
3	[NULL]	0019EB	
4	[NULL]	00673C	
5	[NULL]	00EBC0	
6	[NULL]	00EDF4	
7	[NULL]	016884	
8	[NULL]	016B74	
9	[NULL]	018BDD	
10	[NULL]	0269CF	
11	[NULL]	02E245	
12	[NULL]	0310R2	

Save Cancel Script 200 127 899

Задание 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.

1. Первое Во вложенном подзапросе мы получаем "занятых мест на каждый рейс". Обращаемся к таблице Flights f, и join boarding_passes по первичному ключу flight_id.
2. Получили общую таблицу, в которой теперь нужно взять и посчитать count, из таблицы boarding_passes по новому столбцу ticket_no. Этот подсчёт покажет нам занятых мест, но нам нужна группировка, группируем это всё по flight_id. Заворачиваем это в подзапрос, и оставляем пока там. назовём её "a"
3. Далее нам также нужно узнать сколько вообще мест в каждом самолёте. -- Тут мы будем делать join к нашей вложенной таблице, мы вначале делаем промежуточный join, к этой таблице подзапроса под буквой "a". Джойним как не странно таблицу flights, так как она уже второй раз используется даём ей название f2 джойним её по первичному ключу. По flight_id, далее к этому джойну мы сможем приджойнить общее количество мест в самолёте, -- Мы также пишем джойн и будем джойнить по aircraft_code, и так как мы можем подсчитать общее количество напрямую в джойне. И вовремя джойна у нас полетят только в таблицу только два столбца, это aircraft_code и "Всего мест в самолёте"
4. В "хитром" джойне мы считаем, count (seat_no) и группируем это количество по "aircraft_code", получаем на выходе, код самолёта, и сколько в нём было заложено строк по местам)
5. Далее мы джойним пересчитаную по нашей логике, к таблицу "flights f2" которая джойнится к таблице "a" у которой тоже есть своя логика. Можете запускать эти подзапросы отдельно видеть результат промежуточный и разбирать ход мысли. И так когда у нас есть огромная таблица со всеми столбцами "Всего мест в самолёте" и Занятых мест на каждый рейс мы приступаем к основному решению всё будет производиться в самом верхнем запросе.
6. И так нам нужно оставить "a.flight_id,departure_airport,actual_departure," Для наглядности что бы понимать какой это был рейс название аэропорта и актуальное время отправления - Далее чистая математика начального уровня, Чтобы подсчитать свободных мест для каждого рейса, мы берём "Всего мест в самолёте" и отнимаем "Занятых мест на каждый рейс" называем этот столбец "Свободных мест для каждого рейса"
7. Теперь мы можем искать процент ""Процент свободных мест" для этого нам нужно найти свободных мест для каждого рейса, взять это в скобки, поделить на всего мест в самолёте взять это в скобки, умножить на 100 для приведения в проценты - И так же применить оператор round() который сократит количество цифр после запятой по умолчанию это 1, но мы для наглядности пропишем так же и это. Полученный результат называем AS "Процент свободных мест".
8. Так же нам осталось подсчитать сумму "Накопительный итог". Считается очень просто у нас уже есть столбец "Занятых мест на каждый рейс" в нашем подзапросе, отсюда его и берём его в оконную функцию, прописываем что мы считаем сумму sum по столбцу ("Занятых мест на каждый рейс"), но также нам нужен столбец для группировки потому что таково условие задачи- В столбец для группировки мы помещаем departure_airport и дату актуально вылета но! так как нам нужна группировка по дням, то мы применяем функцию date_trunc('day',actual_departure), Получаем накопительный итог как и велит условие задачи. - На этом всё. Полученный результат вы можете посмотреть ниже.

```

SELECT a.flight_id, departure_airport, actual_departure,
"Всего мест в самолёте" - "Занятых мест на каждый рейс" AS "Свободных мест для каждого рейса",
round((((("Всего мест в самолёте" - "Занятых мест на каждый рейс") /
"Всего мест в самолёте")*100),1) AS "Процент свободных мест",
sum("Занятых мест на каждый рейс")
over (PARTITION BY departure_airport,date_trunc('day',actual_departure) ) AS "Накопительный итог"

FROM (

SELECT f.flight_id ,count(bp.ticket_no) AS "Занятых мест на каждый рейс"
FROM flights f
JOIN boarding_passes bp ON bp.flight_id = f.flight_id
GROUP BY f.flight_id
) a

JOIN flights f2 ON f2.flight_id = a.flight_id
JOIN ( SELECT aircraft_code ,CAST (( count (seat_no))AS NUMERIC) AS "Всего мест в самолёте"
FROM seats
GROUP BY aircraft_code ) b ON b.aircraft_code = f2.aircraft_code

```




	flight_id	departure_airport	actual_departure	Свободных мест для каждого рейса	Процент свободных мест	Накопительный итог
1	20 993	AAQ	2016-09-13 12:08:00	79	60,8	54
2	21 103	AAQ	2016-09-13 11:35:00	94	96,9	54
3	21 096	AAQ	2016-09-14 11:29:00	94	96,9	53
4	20 982	AAQ	2016-09-14 12:07:00	80	61,5	53
5	21 084	AAQ	2016-09-15 11:26:00	92	94,8	55
6	21 041	AAQ	2016-09-15 12:09:00	80	61,5	55
7	21 073	AAQ	2016-09-16 11:26:00	94	96,9	52
8	21 014	AAQ	2016-09-16 12:05:00	81	62,3	52
9	21 002	AAQ	2016-09-17 12:07:00	80	61,5	54
10	21 065	AAQ	2016-09-17 11:27:00	93	95,9	54
11	20 981	AAQ	2016-09-18 12:08:00	76	58,5	62
12	21 058	AAO	2016-09-18 11:26:00	89	91,8	62

Задание 6

Найдите процентное соотношение перелетов по типам самолетов от общего количества.

1. Для решения нам нужны будут модели самолёта- Значит для того чтобы у нас появились "модели" самолётов нам нужен join таблицы flights которая отслеживает все перелёты и участвующие в ней "коды самолётов" с таблицей aircrafts которая имеет у себя соответствие кода к модели самолёта. Для этого мы их соединили по "JOIN aircrafts a ON a.aircraft_code = f.aircraft_code"-
2. Далее я в селект начал считать "count" моделей, и группировать их по моделям, и называю этот столбец "flight_plane" таким образом я нашёл сколько раз летала каждая модель самолёта всего в этой таблице. И так у нас есть количество перелетов для каждой модели, и по условию нам нужно найти "процентное соотношение перелетов по типам самолетов от общего количества" значит ищем общее количество.
3. Для этого я считаю в оконной функции count общий, по всем моделям "count(model) over ()" и называю её "AS numeric" . Далее я записываю всё это в подзапрос чтобы работать с полученными данными.
4. Оставляю в select t.model- по условию они нам нужны. Ну а далее чистая математика , нам нужно взять количество вылетов для каждой модели, разделить на общее количество перелетов , всё это умножить на 100 для получения внятного процента, далее мы получим много цифр после запятой, которые не читаемы глазу, и несут мало информации.
5. Округлим для этого нам и потребовался оператор **ROUND** в котором я указал, цифру 1 , которая покажет нам целое число , и количество цифр после запятой. И тут мы видим много раз повторяющиеся модели это так же не читаемо, поэтому оставим только уникальные значения **DISTINCT** для t.model.
6. Картина поменялась, но так же тяжело читается потому что глазами "не найти" явного фаворита по перевозкам, и мне кажется что тут лучше посмотреть на результат в сортировке. Так гораздо удобнее. И тут я добавляю сортировку **ORDER BY prsoent_flights DESC** по убыванию которая относится ко второй колонке процентов. Таким образом результат получился читаемым и легко интерпретируемым. Результат вы можете посмотреть ниже.



```

SELECT DISTINCT t.model ,
round ((t.flight_plane / t.general_flight_airplane * 100),1) AS prsoent_flights
FROM (
    SELECT * , CAST (count(model) over (partition by model )AS numeric)
    AS flight_plane,
    CAST (count(model) over ()AS numeric ) AS general_flight_airplane
    FROM flights f
    JOIN aircrafts a ON a.aircraft_code = f.aircraft_code
) t
ORDER BY prsoent_flights DESC

```





model	prsoent_flights
Cessna 208 Caravan	28
Bombardier CRJ-200	27,3
Sukhoi SuperJet-100	25,7
Airbus A321-200	5,9
Boeing 737-300	3,8
Boeing 767-300	3,7
Airbus A319-100	3,7
Boeing 777-300	1,8

Задание 7


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

1. Всё что нужно сделать это сравнить цена на рейсы по `max(amount)` для "Economy" класса и `min(amount)` для "Business" Начнём с CTE 2 как не странно , для того чтобы была понятна логика. В CTE2 , в `"SELECT f.flight_id, min(amount) AS min_business"` мы берем создаем столбец `min` значений для каждого рейса об этом говорит группировка , а ещё соответствие в `"join AND (fare_conditions::text) = 'Business'::TEXT"`
2. И так разберем `"JOIN ticket_flights tf2 ON tf2.flight_id = f.flight_id "` с join всё ясно, но далее нужно быть внимательным Идёт условие дополнительное `"AND fare_conditions = 'Business'"` что говорит дай join только тем строкам где столбец `"fare_conditions"` содержит Класс `"Business"` тут всё понятно.- К слову это единственный запрос над которым я достаточно посидел чтобы сделать маленькую оптимизацию в первой CTE. Если обратите внимание там два условия, и не одно не в одном не говорится о эконом классе. Хотя по условию нужны лишь данные эконома..Сначала код был написан на прямом сравнение `"fare_conditions = 'economy'"` Но посмотрев дерево запроса, на специальных ресурсах проанализировав таблицы, я понял что он слишком много строк читает и сравнивает. А отбрасывает в итоге мало. Так вот, вместо того чтобы искать 900+ тысяч результатов и записывать их , я решил сделать условие хитрее `"AND fare_conditions != 'Business' AND fare_conditions != 'Comfort'"` , и так что же она значит . Изучая БД в целом и эту таблицу, стало понятно что она очень правдоподобна, и тут всего 3 класса, получается что если мы исключим два класса которым условие не соответствует(тут внимательно != или <>). Получилось выиграть из 650мс до 360мс. Хотя стоимость cost упала на процентов 5%-7% -
3. По тому же принципу получаем CTE 1 для поиска `"max(amount) AS max_economy"` , буду получать её из join flights f и ticket_flights и добавим `"GROUP BY f.flight_id"` полученный столбец с результатом назовём `"AS max_economy"`- Получили результаты, радуемся..
4. Идём и join делаем для двух получившихся CTE чтобы сравнить их. `JOIN c2 ON c2.flight_id = c1.flight_id.`
5. Далее закрываем CTE 3.Пишем в select что хотим получить все записи, потому как мы не плодили лишних столбцов ставим `"*"`
6. Ну и осталось всё проверить по условию `WHERE min_business < max_economy` -- Полученный результат можете наблюдать ниже



```
with C1 AS ( SELECT f.flight_id, max(amount) AS max_economy
              FROM flights
              JOIN ticket_flights tf2 ON tf2.flight_id = f.flight_id
              AND fare_conditions != 'Business'
              AND fare_conditions != 'Comfort'
              GROUP BY f.flight_id
            ), C2 AS ( SELECT f.flight_id, min(amount) AS min_business
                    FROM flights f
                    JOIN ticket_flights tf2 ON tf2.flight_id = f.flight_id AND fare_conditions = 'Business'
                    GROUP BY f.flight_id
                ), C3 AS ( SELECT *
                        FROM c1
                        JOIN c2 ON c2.flight_id = c1.flight_id )
SELECT *
FROM C3
WHERE min_business < max_economy
```

flight_id	max_economy	flight_id	min_business



Задание 8

Между какими городами нет прямых рейсов?

1. Нам нужно самостоятельно создать представления. Так как идёт сравнение то мы туда поместим Декартово произведение таблицы `airports` через `join` столбца с `a2.city <> a.city`. В итоге мы получим все возможные перелёты, и в `select` выведем `a.city AS "вылет", a2.city AS "прилёт"` потому что нам нужны города..То есть там будут и прямые рейсы и не прямые. Заворачиваем это всё в представление, как говориться в условии, называем `"dekartovo_city"`.
2. Далее мы пишем обращение к нашему представлению то есть во `from` указываем ранее созданное представление, `"dekartovo_city"`. Так же в `select` указываем `"вылет"` и `"прилет"`. И так, тут нужно решать через `EXCEPT`, значит нам нужны две равнозначные таблицы, с одинаковым содержимым в столбцах и одинаковым количеством столбцов.
3. Далее мы получаем должны получить все наши перелеты, во втором `select`, значит мы в таблицу `flights` джойним ту-же таблицу `airports` но два раза Потому как хотим получить, города для аэропорта вылета, и города для аэропорта прилета. Это покажет нам в какие возможные направления вообще летали самолёты, это и будет прямым рейсом.
4. И так получили мы аэропорты, но теперь в первый `select` с которым сравниваем и во второй `select`. Мы берём только строки городов, и делаем одинаковые таблицы чтобы наш оператор `except` адекватно отработал. Помещаем `EXCEPT` между двух запросов.
5. Всё сработало, исключались все прямые рейсы в двух таблицах, остались всевозможные не прямые рейсы. Результат можете наблюдать ниже.

create view dekartovo_city AS

SELECT *

FROM (

SELECT a.city AS "вылет",a2.city AS
"прилёт"

FROM airports a

JOIN airports a2 ON a2.city <> a.city) t

abc	вылет	abc	прилёт
	Мирный		Нижнекамск
	Мирный		Новокузнецк
	Мирный		Нальчик
	Мирный		Владикавказ
	Мирный		Чебоксары
	Мирный		Надым
	Мирный		Нягань
	Мирный		Курск
	Мирный		Саранск
	Мирный		Тамбов
	Мирный		Советский

SELECT "вылет","прилёт"

FROM dekartovo_city

EXCEPT

SELECT "вылет", "прилёт"

FROM

(SELECT flight_id,a.city AS "вылет" , a2.city AS "прилёт"


FROM flights f

JOIN airports a on a.airport_code = f.departure_airport

JOIN airports a2 on a2.airport_code = f.arrival_airport) t




abc	вылет	abc	прилёт
	Новосибирск		Йошкар-Ола
	Норильск		Череповец
	Нижневартовск		Архангельск
	Новокузнецк		Омск
	Новокузнецк		Ханты-Мансийск
	Анапа		Ставрополь
	Калининград		Иваново
	Новосибирск		Чебоксары
	Чита		Москва
	Хабаровск		Архангельск
	Ижевск		Саратов
	Ставрополь		Калининград



Задание 9

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

1. В первом самом глубоком Запросе мы обращаемся к таблице "flights" а также производит три джойна чтобы получить нашу точку на планете земля в виде долготы и широты, которые сейчас представлены в градусах.
2. Далее мы уже в select получившиеся столбцы переводим в радианы функцией RADIANS(столбец) , отлично ,закрепили. Сложили всё в подзапрос.
3. Следующее берём чудесную формулу которая дана нам в приложение к заданию , как вычислить расстояние между аэропортами в Радианах .Мы просто копируем её в селект и называем получившиеся данные AS "Радиан" полученное число не есть километры поэтому работаем дальше переводим всё в подзапрос.
4. Также в приложение для работы с заданием №9 дали еще формулу которая радианы переводит в Км .И получается что в селекте Мы хотим видеть ,аэропорт вылета и прибытия, также км между аэропортами, и для сравнения позже нам понадобится максимально возможное расстояние перелёт самолёта.
5. Ну и также нам нужно знать сможет ли пролететь самолет заданное расстояние поэтому пишем условие "WHERE "Км между аэропортами" > "Макс перелёт самолёта" . Результат 0 результатов смотрите ниже, но решение верное. По условиям оно работает правильно. Просто это некий запрос который и предполагал что там будет 0 результатов, если бы там было много результатов нам бы пришлось серьёзно задуматься.
6. Так как результат 0 результатов, я решил не выводить скрин таблицы. Ну а какие были столбцы можно понять по запросу. К слову запрос я постарался разделить цветами для наглядности чтобы его было легче читать.



```

SELECT * FROM ( SELECT t.departure_airport,
                      t.arrival_airport,
                      round(("Радян" * 6371)) AS "Км между аэропортами",
                      t."range" AS "Макс перелёт самолёта"
FROM ( SELECT      *,
                acos(sin(latitude_a)*sin(latitude_b) + cos(latitude_a)*cos(latitude_b)*cos(longitude_a - longitude_b)) AS "Радян"
FROM ( SELECT f.flight_id,
              f.departure_airport ,
              f.arrival_airport ,
              RADIANS(a.longitude) AS longitude_a ,
              RADIANS(a2.longitude) AS longitude_b ,
              RADIANS(a.latitude) AS latitude_a ,
              RADIANS(a2.latitude) AS latitude_b,
              a3."range"
FROM flights f
JOIN airports a ON a.airport_code = f.departure_airport
JOIN airports a2 ON a2.airport_code = f.arrival_airport
JOIN aircrafts a3 ON a3.aircraft_code = f.aircraft_code)t
)t

GROUP BY departure_airport, arrival_airport,t."Радян",t.RANGE)t2
WHERE "Км между аэропортами" > "Макс перелёт самолёта"

```



Вывод о проделанной работе.



По итогам проделанной работы я многому научился, даже не перечислить все знаний на пальцах. Задания подобраны так что мы закрепили все полученные знания. В конце когда я уже писал запросы, я заметно быстрее начал подбирать логику и синтаксис. Начал понимать как обходиться более меньшим набором строк. Смог сделать некую оптимизацию одного запроса, остальные я разберу для себя сам позже. Подчеркнул для себя лично что не каждая таблица должна иметь строки в выводе иногда нужны такие таблицы, "стоп кран".

Спасибо за обучение

Выполнил : Макаров Владимир Алексеевич