Итоговая работа по модулю

«SQL и получение данных»

Изображение выглядит как текст, карта, небо

Автоматически созданное описание

Выполнила:

Джаримбетова Л.А

2022 г.

1. **В работе использовался Облачный тип подключения.**
2. **Скриншот ER-диаграммы из DBeaver`a.**

Изображение выглядит как стол

Автоматически созданное описание

1. **Краткое описание БД (таблицы и представления).**



База данных состоит из 8-ми таблиц:

1)boarding\_passes: номер билета (ticket\_no), id рейса (fight\_id), номер посадочного(boarding\_no), номер места (seat\_no)

2)ticket\_flights: номер билета (ticket\_no), id рейса (fight\_id),  класс обслуживания (fare\_conditions), стоимость перелета (amount)

3)flights : id рейса(fight\_id), номер рейса(fight\_no), время вылета ( scheduled\_departure)и прилета  (scheduled\_arrival) по расписанию,  
аэропорты отправления (departure\_airport) и прибытия (arrival\_airport), статус (status)рейса, код судна (aircraft\_code), фактическое время вылета (actual\_departure) и прилета (actual\_arrival)

4)seats: код судна (aircraft\_code), номер места (seat\_no), класс обслуживания (fare\_conditions)

5)tickets: номер билета (ticket\_no), номер бронирования (book\_ref) , id пассажира (passenger\_id), ФИ пассажира (passenger\_name ),  
контактные данные пассажира (contact\_data).

6)bookings: номер бронирования (book\_ref), дата бронирования (book\_date),  полная сумма бронирования (total\_amount).

7)aircrafts\_data:  модель судна (model), максимальная дальность полёта (км) (range),  код воздушного судна  (aircraft\_code).

8)airports\_data:  код аэропорта (airport\_code), название аэропорта (airport\_name), город (city) , координаты (coordinates), временная зона аэропорта (timezone).

Представления:

flights\_v:  идентификатор рейса (flight\_id), номер рейса (flight\_no), время вылета по расписанию (scheduled\_departure)  и местное (scheduled\_departure\_local), время прилета ( scheduled\_arrival) по расписанию и местное (scheduled\_arrival\_local) , планируемая продолжительность полета (scheduled\_duration) , код аэропорта отправления (departure\_airport) , название аэропорта отправления ( departure\_airport\_name), город отправления (departure\_city), код аэропорта прибытия (arrival\_airport) , название аэропорта прибытия (arrival\_airport\_name), город прибытия (arrival\_city), статус рейса (status) , код самолета (aircraft\_code), фактическое время вылета (actual\_departure) и местное (actual\_departure\_local) , фактическое время прилета (actual\_arrival) и местное (actual\_arrival\_local), фактическая продолжительность полета (actual\_duration)

routes (материализованное):  номер рейса (flight\_no),  код аэропорта отправления (departure\_airport), название аэропорта отправления (departure\_airport\_name), город отправления (departure\_city) , код аэропорта прибытия (arrival\_airport), название аэропорта прибытия (arrival\_airport\_name), город прибытия (arrival\_city),  код самолёта (aircraft\_code), продолжительность полета (duration) , дни недели рейсов (days\_of\_week).

aircrafts: код воздушного судна (aircraft\_code), модель судна (model), максимальная дальность полёта (км) (range).

airports: код аэропорта (airport\_code) , название аэропорта(airport\_name), город (city) , координаты (coordinates), временная зона аэропорта (timezone).

1. **Развернутый анализ БД .**

**aircrafts\_data:**

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

Индексы: PRIMARY KEY, btree (aircraft\_code)

Ограничения-проверки: CHECK (range > 0)

Ссылки извне: TABLE "flights" FOREIGN KEY (aircraft\_code) REFERENCES aircrafts(aircraft\_code) TABLE "seats" FOREIGN KEY (aircraft\_code) REFERENCES aircrafts(aircraft\_code) ON DELETE CASCADE

**airports\_data:**

Аэропорт идентифицируется трехбуквенным кодом (airport\_code) и имеет свое имя (airport\_name). Название города (city) указывается и может служить для того, чтобы определить аэропорты одного города. Также указывается широта (longitude), долгота (latitude) и часовой пояс (timezone).

Индексы: PRIMARY KEY, btree (airport\_code)

 Ссылки извне: TABLE "flights" FOREIGN KEY (arrival\_airport) REFERENCES airports(airport\_code) TABLE  
"flights" FOREIGN KEY (departure\_airport) REFERENCES airports(airport\_code) 4.3

**boarding\_passes:**

При регистрации на рейс, которая возможна за сутки до плановой даты отправления, пассажиру выдается посадочный талон. Он идентифицируется также, как и перелет — номером билета и номером рейса. Посадочным талонам присваиваются последовательные номера (boarding\_no) в порядке регистрации пассажиров на рейс (этот номер будет уникальным только в пределах данного рейса). В посадочном талоне указывается номер места (seat\_no).

Индексы: PRIMARY KEY, btree (ticket\_no, flight\_id) UNIQUE CONSTRAINT, btree (flight\_id, boarding\_no) UNIQUE CONSTRAINT, btree (flight\_id, seat\_no)

Ограничения внешнего ключа: FOREIGN KEY (ticket\_no, flight\_id) REFERENCES ticket\_flights(ticket\_no, flight\_id)

**bookings:**

Пассажир заранее (book\_date, максимум за месяц до рейса) бронирует билет себе и, возможно, нескольким другим пассажирам. Бронирование идентифицируется номером (book\_ref, шестизначная комбинация букв и цифр). Поле total\_amount хранит общую стоимость включенных в бронирование перелетов всех пассажиров.

Индексы: PRIMARY KEY, btree (book\_ref)

Ссылки извне:TABLE"tickets"FOREIGNKEY(book\_ref)REFERENCESbookings(book\_ref)

**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.

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:FOREIGNKEY(aircraft\_code)REFERENCESaircrafts(aircraft\_code) FOREIGN KEY (arrival\_airport) REFERENCES airports(airport\_code) FOREIGN KEY (departure\_airport) REFERENCES airports(airport\_code)

Ссылки извне:TABLE"ticket\_flights"FOREIGNKEY(flight\_id)REFERENCESflights(flight\_id) 4.6

**seats:**

Места определяют схему салона каждой модели. Каждое место определяется своим номером (seat\_no) и имеет закрепленный за ним класс обслуживания (fare\_conditions) — Economy, Comfort или Business.

Индексы: PRIMARY KEY, btree (aircraft\_code, seat\_no)

Ограничения-проверки:CHECK(fare\_conditionsIN('Economy','Comfort','Business'))

Ограничения внешнего ключа: FOREIGN KEY (aircraft\_code) REFERENCES aircrafts(aircraft\_code) ON  
DELETE CASCADE

**ticket\_flights:**

Перелет соединяет билет с рейсом и идентифицируется их номерами. Для каждого перелета указываются его стоимость (amount) и класс обслуживания (fare\_conditions).

Индексы: PRIMARY KEY, btree (ticket\_no, flight\_id)

Ограничения-проверки: CHECK (amount >= 0) CHECK (fare\_conditions IN ('Economy', 'Comfort',  
'Business'))

Ограничения внешнего ключа: FOREIGN KEY (flight\_id) REFERENCES flights(flight\_id) FOREIGN KEY  
(ticket\_no) REFERENCES tickets(ticket\_no)

Ссылки извне: TABLE "boarding\_passes" FOREIGN KEY (ticket\_no, flight\_id) REFERENCES  
ticket\_flights(ticket\_no, flight\_id)

**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).

Индексы: PRIMARY KEY, btree (ticket\_no)

 Ограничения внешнего ключа:FOREIGNKEY(book\_ref)REFERENCESbookings(book\_ref)

Ссылки извне:TABLE"ticket\_flights"FOREIGNKEY(ticket\_no)REFERENCEStickets(ticket\_no)

**Бизнес-задачи:**



Задача любого бизнеса базируется на получение прибыли, которую в конкретном случае можно увеличить за счет правильного распределения ресурсов, через оптимизацию маршрутов рейсов и загруженности.

1)На основании количества купленных мест можно сформулировать вывод о рентабельности полета конкретного рейса и принять управленческое решение (например, запустить дополнительный рейс на более востребованное направление или сократить рейсы, которые не имеют спроса).

3)На основании таблицы из бронирования пассажир за месяц бронирует билет себе и возможно другим пассажирам. Можно рассчитать какая доля пассажиров из бронирования совершает финальную покупку билета, а какая доля отказывается от завершения брони. Для выявления и дальнейшего устранения корневых причин.

3)Можно сформировать выводы о перераспределении воздушных судов в рамках одной авиакомпании т.е при недостаточном или наоборот повышенном спросе, с целью снижения производственных расходов на полет (оптимизация расходов: топливо, оплата труда, техническое обслуживание и т.д).

4)Проанализировать рейсы с хронической задержкой, с целью разбора корневых причин и их устранения, что позволит увеличить клиентскую лояльность.

5)Провести конкурентный анализ по стоимости билетов, исходя из дат бронирования и суммы. С целью урегулирования тарифной сетки.

6)Сформировать программу лояльности для клиентов, которые на регулярной основе пользуются услугами авиаперевозки.

* 1. **Список SQL запросов с описанием логики их выполнения.**



**--Задача 1**

--В каких городах больше одного аэропорта?

**select** city **from** airports -- получаю название городов из таблицы aiprorts

**group** **by** city -- группирую данную таблицу по полю city

**having** **count**(\*) > 1 -- фильтрую по условию having, где кол-во сгруппированных записей должно быть больше 1

**--Задача 2**

--В каких аэропортах есть рейсы, выполняемые самолетом с максимальной дальностью перелета? (обязательно: подзапрос)

**select** departure\_airport **as** аэропорт --вывожу аэропорты из табл flights

**from** flights

**where** aircraft\_code =

(**select** aircraft\_code -- в подзапросе вывожу самолет с максимальной дальностью

**from** aircrafts\_data

**order** **by** "range"**desc** -- сортирую по убыванию

**limit** 1) -- и оставляю самый дальний

**group** **by** departure\_airport -- группирую по названию аэропорта вылета

**--Задача 3**

--Вывести 10 рейсов с максимальным временем задержки вылета. Обязательно: limit

**select** flight\_id , actual\_departure - scheduled\_departure **as** "время задержки"

**from** flights -- вывожу из табл flight данные по id и разность между фактическим временем вылета и временем вылета по расписанию

**where** actual\_departure **is** **not** **null** -- исключаю записи, где фактическое время вылета null

**order** **by** "время задержки" **desc** -- сортирую полученную разность по убыванию

**limit** 10 -- вывожу первые 10 значений

**--Задача 4**

--Были ли брони, по которым не были получены посадочные талоны? Обязательно: верный тип join

**select** **count** (\*) **as** "кол-во броней без посадочного" -- считаю кол-во записей из табл bookings

**from** bookings

**left** **join** tickets **on** tickets.book\_ref=bookings.book\_ref -- присоединяю номера бронирования

**left** **join** boarding\_passes **on** boarding\_passes.ticket\_no = tickets.ticket\_no -- присоединяю номера билетов

**where** boarding\_no **is** **null** -- фильтрую бронирования по тем значениям, где нет номеров посадочных талонов

**--Задача 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.

--Обязательно: оконная функция, подзапросы или/и cte

--использую три таблицы: boarding\_passes, flights, seats. Чтобы найти кол-во свободных мест,

--нужно найти кол-во посадочных мест и кол-во выданных посадочных талонов.

**with** cte1 **as** -- в этом cte считаю кол-во посадочных мест для каждой модели

(**select** seats.aircraft\_code , **count**(seats.seat\_no) **as** "общее кол-во мест"

**from** seats

**group** **by** seats.aircraft\_code),

cte2 **as** -- в этом cte обогащаю 1-е данными из табл flights и boarding\_passes, чтобы найти выданные посадочные талоны

(**select** flights.flight\_id , flights.departure\_airport , flights.actual\_departure ,

cte1."общее кол-во мест",

**count**(boarding\_passes.seat\_no) **as** "места заняты", --кол-во мест занятых пассажирами

cte1."общее кол-во мест" - **count**(boarding\_passes.seat\_no) **as** "свободные места" ,

--кол-во свободных мест, считаю разность между общем кол-вом мест и выданными талонами

**round**((cte1."общее кол-во мест" - **count**(boarding\_passes.seat\_no))::**numeric** / cte1."общее кол-во мест" \* 100) **as** "% свободных мест"

--высчитываю процент свободных мест

**from** cte1

**join** flights **on** cte1.aircraft\_code = flights.aircraft\_code

**join** boarding\_passes **on** flights.flight\_id = boarding\_passes.flight\_id

**group** **by** flights.flight\_id , cte1.aircraft\_code, cte1."общее кол-во мест")

--соединяю таблицы и делаю группировку

**select** \*,

**sum**(cte2."места заняты") **over** (**partition** **by** cte2.actual\_departure::**date**,

cte2.departure\_airport **order** **by** cte2.actual\_departure) **as** "суммарное накопление"

--созданную оконную фун-ю для расчёта суммарного накопления на каждый день из каждого аэропорта на каждый день

--суммирую кол-во занятых мест на рейсе, группирую по departure\_airport, actual\_departure

--привожу к типу данных date, чтобы убрать время и оставить возможность подсчета по дням

--сортирую по actual\_departure где весть дата с указанием времени

**from** cte2

**--Задача 6**

--Найдите процентное соотношение перелетов по типам самолетов от общего количества. Обязательно: подзапрос или окно, оператор ROUND.

**select** aircrafts\_data.model, -- получаю название моделей самолетов

**round**(**count**(\*)::**numeric** -- считаю кол-во рейсов(привожу к типу numeric для оторбражения остатка от деления)

/ (**select** **count**(\*) **from** flights) \* 100, 1)

--и делю на подзапрос, где считаю общее кол-во рейсов, умножаю на 100, чтобы получить % (округляю до десятых)

**as** "процентное соотношение перелетов" -- именую колонку

**from** flights

**inner** **join** aircrafts\_data **on** flights.aircraft\_code = aircrafts\_data.aircraft\_code -- при помощи джойна из таблицы aircrafts\_data подтягию моделю

**group** **by** aircrafts\_data.aircraft\_code -- группирую по коду судна

**--Задача 7**

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

--ответ: нет, не было

**with** cte\_business **as** ( -- создаю cte для бизнес-класса

**select** flight\_id, fare\_conditions, **min** (amount) **as** "минимальная сумма" -- вывожу id рейса, класс обслуживания и считаю минимальную стоимость перелета

**from** ticket\_flights

**where** fare\_conditions = 'Business'

**group** **by** flight\_id, fare\_conditions), -- группирую

cte\_economy **as** ( -- создаю cte для эконом-класса

**select** flight\_id, fare\_conditions, **max**(amount) **as** "максимальная сумма" ---- вывожу id рейса, класс обслуживания и считаю максимальную стоимость перелета

**from** ticket\_flights

**where** fare\_conditions = 'Economy'

**group** **by** flight\_id, fare\_conditions) -- группирую

**select** flights.flight\_id, city,"минимальная сумма", "максимальная сумма" --вывожу еще города из airports\_data

**from** airports\_data

**join** flights **on** airports\_data.airport\_code = flights.arrival\_airport -- добавляем аэропорты назначения в таблице flights названиями городов

**join** cte\_business **using** (flight\_id) -- присоединяю по id рейса из cte бизнес

**join** cte\_economy **using** (flight\_id) -- присоединяю по id рейса из cte эконом

**where** "минимальная сумма" < "максимальная сумма" -- добавляю условие по сумме, где минимальная стоимость бизнес должна быть меньше максимальной стоимости эконом

--Задача 8

--Между какими городами нет прямых рейсов? Обязательно: декартово произведение в предложении FROM,

--самостоятельно созданные представления (если облачное подключение, то без представления),оператор EXCEPT.

**select** **distinct** t1.city, t2.city

**from** airports t1 -- создаем декартово произведение

**cross** **join** airports t2 -- использую кросс джойн для получение всех возможных пар городов с ней самой

**where** t1.city <> t2.city -- использую условие, которые уберет зеркальные города

**except** -- удаляю из декартового произведения пары городов, которые есть в представлении

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

**from** routes r

--Задача 9

--Вычислите расстояние между аэропортами, связанными прямыми рейсами, сравните с допустимой максимальной дальностью перелетов

--в самолетах, обслуживающих эти рейсы \* Обязательно: оператор RADIANS или использование sind/cosd,CASE.

**select** **distinct** --убираю дубли

t1.airport\_name **as** "аэропорт отправления",

t2.airport\_name **as** "аэропорт прилета",

a.model **as** "модель самолета",

a."range",

**round**((**acos**(**sind**(t1.coordinates [1]) \* **sind**(t2.coordinates [1]) + **cosd**(t1.coordinates [1])

\* **cosd**(t2.coordinates [1]) \* **cosd**(t1.coordinates [0] - t2.coordinates [0])) \* 6371)::**numeric**, 2) **as** "расстояние",

-- осуществляю расчеты по формуле, с кординатами работаю через массив 0-широта, 1-долгота

**case**

**when** a."range" < **acos**(**sind**(t1.coordinates [1]) \* **sind**(t2.coordinates [1])

+ **cosd**(t1.coordinates [1]) \* **cosd**(t2.coordinates [1]) \* **cosd**(t1.coordinates [0] - t2.coordinates [0])) \* 6371

**then** 'нет'

**else** 'да'

**end** **result**

--сравниваю с данными по расстоянию для модели самолета, если дальность меньше дальности полета, то "да"

**from** flights f

**join** airports t1 **on** f.departure\_airport = t1.airport\_code

**join** airports t2 **on** f.arrival\_airport = t2.airport\_code

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

**order** **by** t1.airport\_name, t2.airport\_name, a.model

-- объединяю данные, затем делаю сортировку