Итоговая работа "SQL и получение данных"

Студентки группы SQL-45
Минеевой Елизаветы

Критерии оценивания итоговой работы

Приложение №1

№	Содержание	Баллы за оформление
1	В работе использовался _____ тип подключения. Если база была развернута из .sql или .backup файла, необходимо приложить скриншот успешного импорта или восстановления	0 - облачная база, 10 - локальная база
2	Скриншот ER-диаграммы из DBeaver согласно вашего подключения	5
3	Краткое описание БД - из каких таблиц и представлений состоит	10
4	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	20
5	Список SQL запросов из приложения №2 с описанием логики их выполнения	15

Итого: максимум 60 баллов.
Для зачета необходимо набрать **минимум 30 баллов**.

Приложение №2

№	Вопрос	В решении обязательно должно быть использовано	Баллы за запросы
1	В каких городах больше одного аэропорта?		10

2	В каких аэропортах есть рейсы, выполняемые самолетом с максимальной дальностью перелета?	Подзапрос	15
3	Вывести 10 рейсов с максимальным временем задержки вылета	Оператор LIMIT	15
4	Были ли брони, по которым не были получены посадочные талоны?	Верный тип JOIN	15
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	Оконная функция; подзапросы или/и cte	35
6	Найдите процентное соотношение перелетов по типам самолетов от общего количества	Подзапрос или окно; оператор ROUND	25
7	Были ли города, в которые можно добраться бизнес - классом дешевле, чем эконом-классом в рамках перелета?	CTE	25
8	Между какими городами нет прямых рейсов?	Декартово произведение в предложении FROM; самостоятельно созданные представления	25

		(если облачное подключение, то без представления); оператор EXCEPT	
9	Вычислите расстояние между аэропортами, связанными прямыми рейсами, сравните с допустимой максимальной дальностью перелетов в самолетах, обслуживающих эти рейс*	Оператор RADIANS или использование sind/cosd; CASE	35

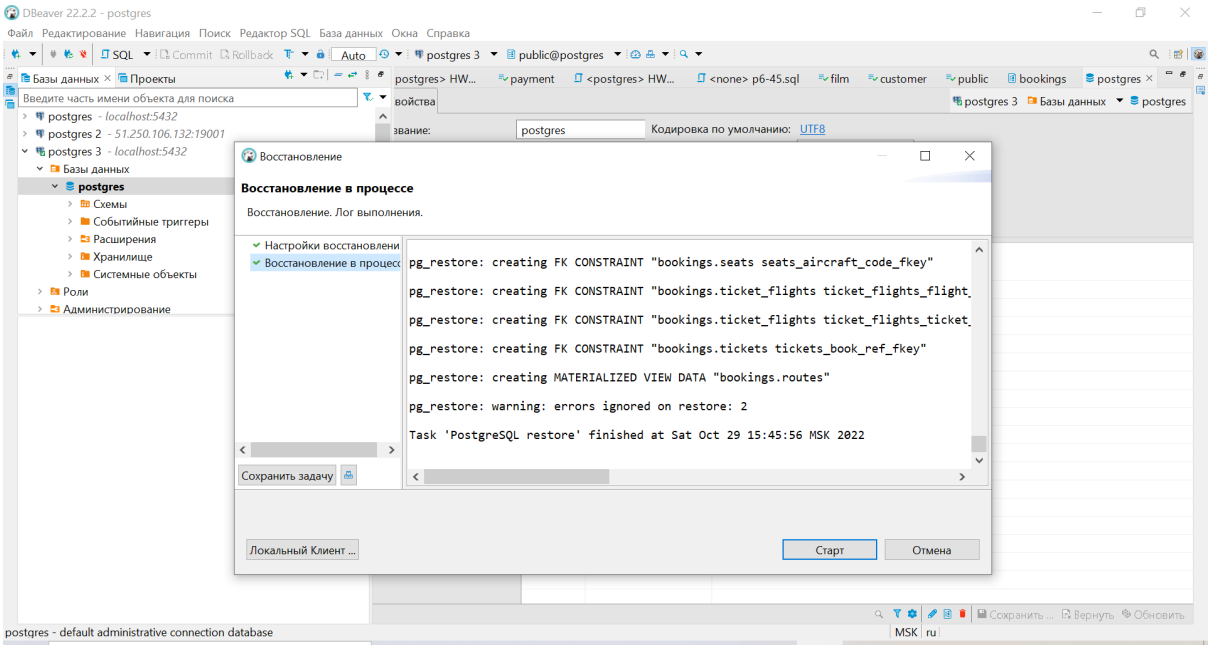
*В облачной базе координаты находятся в столбце airports_data.coordinates - работаете, как с массивом. В локальной базе координаты находятся в столбцах airports.longitude и airports.latitude.
Кратчайшее расстояние между двумя точками A и B на земной поверхности (если принять ее за сферу) определяется 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.

Итого: максимум 200 баллов.
Для зачета необходимо набрать **минимум 130 баллов**.

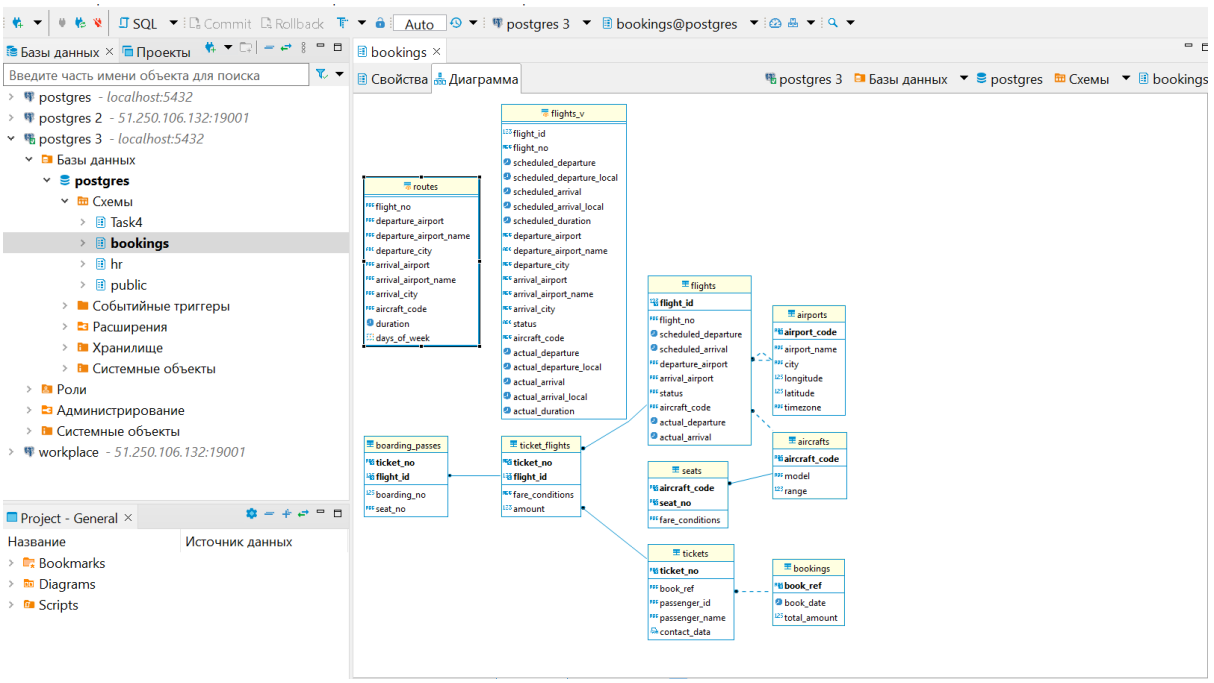
Выполненные задания

Приложение №1

1. В работе использовался **локальный** тип подключения. Если база была развернута из .sql или .backup файла, необходимо приложить скриншот успешного импорта или восстановления



2. Скриншот ER-диаграммы из DBeaver согласно вашего подключения



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

   Таблицы:

   - bookings.aircrafts: код самолета (IATA), модель самолета, максимальная дальность полета (км)
   - bookings.airports: код аэропорта, название аэропорта, город, координаты аэропорта (долгота), координаты аэропорта (широта), временная зона аэропорта
   - bookings.boarding_passes: номер билета, идентификатор рейса, номер посадочного талона, номер места
   - bookings.bookings: номер бронирования, дата бронирования, полная сумма бронирования
   - bookings.flights: идентификатор рейса, номер рейса, время вылета по расписанию, время прилета по расписанию, аэропорт отправления, аэропорт прибытия, статус рейса, код самолета (IATA), фактическое время вылета, фактическое время прилета
   - bookings.seats: код самолета (IATA), номер места, класс обслуживания
   - bookings.ticket_flights: номер билета, идентификатор рейса, класс обслуживания, стоимость перелета
   - bookings.tickets: номер билета, номер бронирования, идентификатор пассажира, имя пассажира, контактные данные пассажира

   Представления:

   - bookings.flights_v: идентификатор рейса, номер рейса, время вылета по расписанию, время вылета по расписанию (местное), время прилета по расписанию, время прилета по расписанию (местное), планируемая продолжительность полета, код аэропорта отправления, название аэропорта отправления, город отправления, код аэропорта прибытия, название аэропорта прибытия, город прибытия, статус рейса, код самолета (IATA), фактическое время вылета, фактическое время вылета (местное), фактическое время прилета, фактическое время прилета (местное), фактическая продолжительность полета
   - bookings.routes (материализованное): номер рейса, код аэропорта отправления, название аэропорта отправления, город отправления, код аэропорта прибытия, название аэропорта прибытия, город прибытия, код самолета (IATA), продолжительность полета, дни недели, когда выполняются рейсы

4. 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

   bookings.aircrafts:

   - 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

   bookings.airports:

   - аэропорт идентифицируется трехбуквенным кодом (airport_code) и имеет свое имя (airport_name)
   - для города не предусмотрено отдельной сущности, но название (city) указывается и может служить для того, чтобы определить аэропорты одного города
   - также указывается широта (longitude), долгота (latitude) и часовой пояс (timezone)
   - 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)

   bookings.boarding_passes:

   - при регистрации на рейс, которая возможна за сутки до плановой даты отправления, пассажиру выдается посадочный талон, который идентифицируется так же, 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.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" FOREIGN KEY (book_ref) REFERENCES bookings(book_ref)

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)
    - 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)
- 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)

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
- 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

bookings.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)

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
- 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)

Бизнес задачи, которые можно решить, используя БД

1. Нахождение 10 рейсов с максимальным временем задержки вылета может помочь выявить взаимосвязи между всеми случаями задержки и устранить ее для повышения качества обслуживания.
2. На основании процентного соотношения количества свободных мест для каждого рейса к общему количеству мест в самолете можно сделать вывод об убыточности определенных перелетов

или рейсов и провести оптимизацию (например, проводить перелеты реже).

3. С помощью таблиц bookings.boarding passes и bookings.flights можно определить популярность каждого рейса/направления, посчитав количество посадочных талонов на тот или иной рейс. С помощью условия можно ограничить данный рейтинг только непрямыми рейсами. В результате проведенного исследования можно задуматься о том, чтобы наладить прямое сообщение в наиболее популярных направлениях данного рейтинга.

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

Задание №1

В каких городах больше одного аэропорта?

```
select a.city
from airports a
group by a.city
having count(a.city) > 1
```

Берем данные из таблицы airports, группируем по столбцу city. Далее применяем условие к результату группировки. Таким образом, если город встречается больше одного раза (то есть количество упоминаний города больше одного), мы выводим данный город в результат.

Задание №2

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

```
select f.departure_airport as airport_code, f.aircraft_code
from flights f
where f.aircraft_code in (
        select ac.aircraft_code
        from aircrafts ac
        order by ac."range" desc
        limit 1
 )
group by 1, 2
```

Берем данные из таблицы flights. Так как в таблице flights нет данных о дальности перелета, нам нужно получить данные также из таблицы

aircrafts. Таким образом, в оператор where добавляем условие, согласно которому aircraft_code (код самолета) должен находиться в подзапросе, в котором все коды самолетов сортированы по дальности перелета от большего к меньшему и выведена первая строка (то есть самолет с максимальной дальностью перелета). Также необходимо сгруппировать результаты данной выборки по двум столбцам - коду аэропорта и коду самолета. Выводим код аэропорта и код самолета с максимальной дальностью перелета.

Задание №3

Вывести 10 рейсов с максимальным временем задержки вылета.

```
select f.flight_id, f.flight_no, f.actual_departure - f.scheduled_departure as
delay
from flights f
where f.status = 'Arrived' or f.status = 'Departed'
order by 3 desc
limit 10
```

Берем данные из таблицы flights. Необходимо применить условие, в котором status(статус рейса) будет 'Arrived' или 'Departed', так как невозможно установить точное время задержки вылета, когда самолет еще не вылетел.
Выводим идентификатор рейса, номер рейса, время задержки (разница между фактическим временем прибытия и планируемым временем прибытия), сортируем данные по времени задержки от большего к меньшему и ограничиваем количество строк до 10. Таким образом, получаем 10 строк с рейсами с максимальным временем задержки вылета.

Задание №4

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

```
select b.book_ref
from bookings b
join tickets t on b.book_ref = t.book_ref
left join boarding_passes bp on t.ticket_no = bp.ticket_no
where bp.boarding_no is null
```

Берем данные из таблицы bookings (брони). Соединить напрямую таблицы bookings и boarding_passes невозможно, так как у них нет общих данных, поэтому сначала присоединяем таблицу tickets с помощью inner

join, получаем данные, которые есть в обеих таблицах, а затем присоединяем таблицу boarding_passes с помощью left join, получаем все данные из предыдущих (левых) таблиц и пересечение множеств из добавленной (правой) таблицы. Если пересечения нет, в строке будет 'null'. С помощью условия where получаем брони, у которых не было пересечения с добавленной таблицей boarding_passes по столбцу boarding_no (номер посадочного талона). Таким образом, получаем брони, по которым не были получены посадочные талоны.

Задание №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.

```
with cte1 as (
        select f.flight_id, count(s.seat_no) as "count_seats"
        from flights f
        join seats s on f.aircraft_code = s.aircraft_code
        group by 1),
cte2 as (
        select f.flight_id, count(bp.boarding_no) as "count_boarding"
        from flights f
        left join boarding_passes bp on f.flight_id = bp.flight_id
        group by 1)
select f.flight_id, c1.count_seats - c2.count_boarding as "free seats",
round((c1.count_seats - c2.count_boarding) * 100 / c1.count_seats) as
percentage,
f.departure_airport,
f.actual_departure,
sum(c2.count_boarding) over (partition by f.departure_airport,
date_trunc('day', f.actual_departure) order by f.actual_departure) as
departed_passengers
from flights f
join cte1 c1 on f.flight_id = c1.flight_id
join cte2 c2 on c1.flight_id = c2.flight_id
order by f.departure_airport, f.actual_departure
```

Сначала формируем общие табличные выражения.

В первом CTE берем данные из таблицы flights, присоединяем таблицу seats с помощью inner join, так как нужно получить все пересекающиеся значения из обеих таблиц, и группируем по столбцу с идентификатором рейса. Таким образом, получаем общее количество мест для каждого рейса.
Во втором CTE также берем данные из таблицы flights, присоединяем таблицу boarding_passes с помощью left join, получаем все данные из левой таблицы flights и пересечение множеств из добавленной (правой) таблицы boarding_passes. В данной случае inner join не подходит, так как возникнуть ситуация, когда у рейса не будет посадочных талонов вообще.
В основном запросе берем данные из таблицы flights и присоединяем общие табличные выражения cte1 и cte2 с помощью inner join. Затем находим разницу между общим количеством мест и количеством посадочных талонов, которая будет являться количеством свободных мест на рейсе. Данную разницу умножаем на 100, делим на общее количество мест в самолете и округляем до целого числа, в результате чего получаем процент свободных мест на рейсе. С помощью оконной функции получаем сумму посадочных талонов, разделяем результирующий набор запроса на секции по аэропорту и дню вылета и сортируем по дате вылета.
Сортируем основной запрос по аэропорту и дате вылета, чтобы наглядно увидеть суммарное накопление количества вывезенных пассажиров из каждого аэропорта на каждый день.

Задание №6

Найдите процентное соотношение перелетов по типам самолетов от общего количества.

```
select f.aircraft_code, round(count(f.flight_id) * 100 / sum(count(f.flight_id)) over (), 0) as percentage
from flights f
group by 1
```

Берем данные из таблицы flights. Группируем по столбцу aircraft_code.С помощью функции count сначала получаем количество перелетов по типам самолетов, умножаем данное значение на 100 и делим на общее количество перелетов, которое получаем с помощью оконной функции sum() over (). Так как мы работаем со всеми строками, в скобках после over ничего не указано. С помощью оператора round мы округляем значение до целого числа (0 знаков после запятой).

Задание №7

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

```
with cte1 as (
        select tf.flight_id, tf.amount as amount1
        from ticket_flights tf
        where tf.fare_conditions = 'Economy'
        group by 1, 2),
cte2 as (
        select tfs.flight_id, tfs.amount as amount2
        from ticket_flights tfs
        where tfs.fare_conditions = 'Business'
        group by 1, 2)
select a.city
from airports a
join flights f on a.airport_code = f.arrival_airport
join cte1 c1 on f.flight_id = c1.flight_id
join cte2 c2 on c1.flight_id = c2.flight_id
where c1.amount1 > c2.amount2
```

Сначала создаем два табличных выражения (cte1 и cte2).
В первом табличном выражении cte1 берем данные из таблицы ticket_flights и задаем условие, согласно которому будут выведены только те строки, в которых класс обслуживания - 'Economy'. Производим группировку по столбцам flight_id и amount.
Во втором табличном выражении cte2 то же самое, что и в первом, но класс обслуживания в условии - 'Business'.
В основном запросе берем данные из таблицы airports, присоединяем таблицу flights с помощью inner join, так как нужны все пересекающие данные из обеих таблиц. Таким же образом присоединяем cte1 и cte2. Прописываем условие, согласно которому стоимость перелета эконом-классом (из cte1) должна быть больше стоимости перелета бизнес-классом (из cte2). Выводим все города, которые удовлетворяют данному условию.

Задание №8

Между какими городами нет прямых рейсов?

```
create view flights_view as
        select f.flight_id,
        f.flight_no,
        f.departure_airport,
```

```
    dep.airport_name as departure_airport_name,
    dep.city as departure_city,
    f.arrival_airport,
    arr.airport_name as arrival_airport_name,
    arr.city as arrival_city
    from flights f,
    airports dep,
    airports arr
    where f.departure_airport = dep.airport_code and f.arrival_airport =
    arr.airport_code

select a1.city, a2.city
from airports a1 cross join airports a2
except
select departure_city, arrival_city
from flights_view
```

Сначала создаем представление flights_view. В данном представлении берем данные из таблиц flights и airports. Таблицу airports прописываем в from два раза, так как нам нужно внести в представление два столбца с городами - город вылета и город прибытия. В условии прописываем, что код аэропорта вылета из таблицы flights должен совпадать с кодом аэропорта из таблицы dep (аэропорт вылета), а код аэропорта прибытия из таблицы flights - с кодом аэропорта из таблицы arr (аэропорт прибытия).

В основном запросе берем данные из таблицы airports, которую называем a1, и сразу же с помощью cross join к этой таблице добавляем такую же таблицу airports, которую называем a2. Получаем всевозможные пары городов. Далее используем оператор except для возврата всех строк в первом операторе select, которые не возвращаются вторым оператором select (во втором операторе select берем данные из flights_view, в котором представлены города с прямым рейсом между ними). Таким образом, получаем пары городов, которые не представлены в flights_view и между которыми, соответственно, нет прямых рейсов.

Задание №9

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

```
select dep.city as departure_city, arr.city as arrival_city, ac."range",
```

```
        acos(sind(dep.latitude) * sind(arr.latitude) + cosd(dep.latitude) *
cosd(arr.latitude) * cosd(dep.longitude - arr.longitude)) * 6371 as distance,
        case
                when ac."range" >= acos(sind(dep.latitude) * sind(arr.latitude) +
cosd(dep.latitude) * cosd(arr.latitude) * cosd(dep.longitude - arr.longitude)) *
6371
                then 'YES'
                else 'NO'
        end as possibility
from flights f
join airports dep on f.departure_airport = dep.airport_code
join airports arr on f.arrival_airport = arr.airport_code
join aircrafts ac on f.aircraft_code = ac.aircraft_code
group by 1, 2, 3, 4
```

Берем данные о рейсах из таблицы flights. Для того, чтобы разделять аэропорты вылета и прибытия, два раза с помощью inner join добавляем к flights таблицу airports на основании равенства между departure_airport и airport_code и arrival_airport и airport_code. Для того, чтобы получить данные о допустимой максимальной дальности перелета, с помощью inner join добавляем также таблицу aircrafts на основании равенства между столбцами aircraft_code из обеих таблиц.

Для группировки одинаковых значений в столбцах (чтобы маршруты не повторялись) используем оператор group by для всех столбцов из select.

Выводим город вылета из таблицы dep, город прибытия из таблицы arr, значение функции арккосинус, умноженное на 6371 (согласно формуле в задании), которое представляет собой кратчайшее расстояние между пунктами, измеряемое в километрах, и результат в зависимости от выполнения того или иного условия. Если допустимая максимальная дальность перелета больше или равна полученного расстояния между пунктами, выводится результат 'YES' (перелет возможен). Если же условие не выполняется, выводится 'NO' (перелет невозможен).