

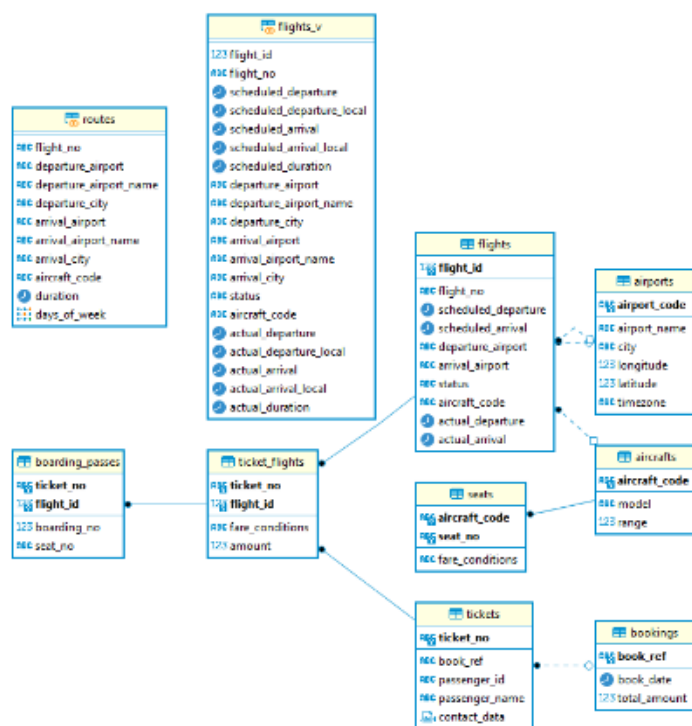
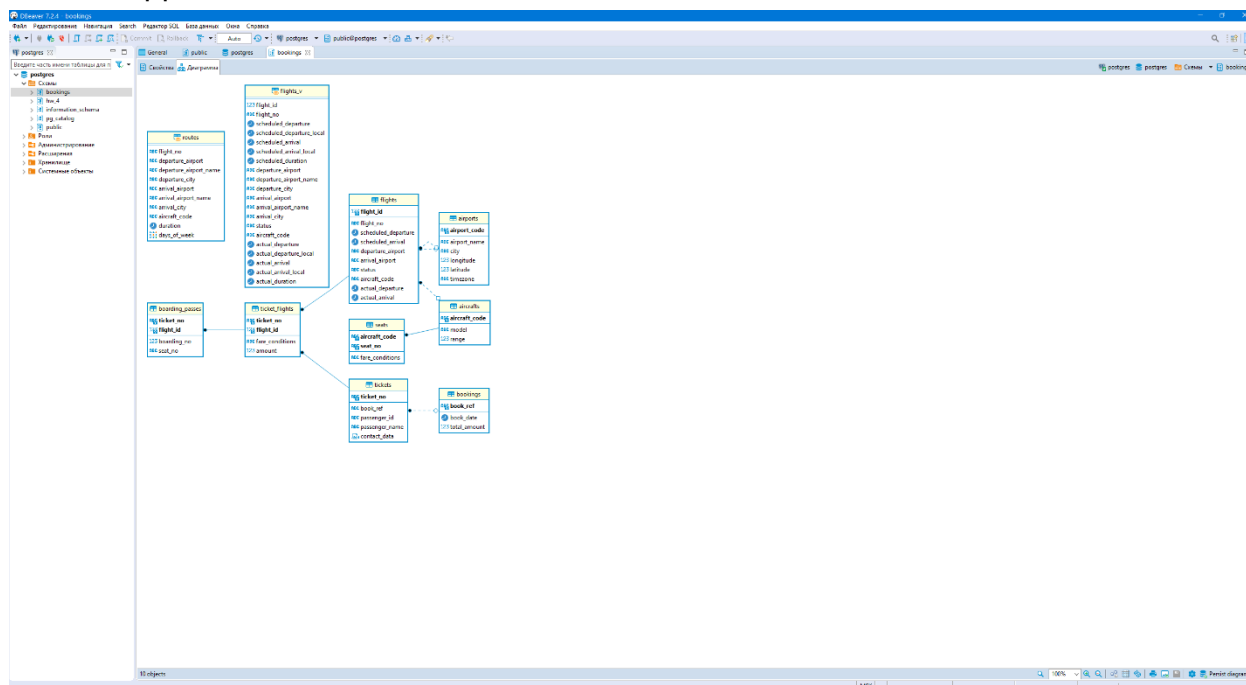
1. В работе использовался локальный тип подключения.

[illegible]

The screenshot displays the pgAdmin 4 web interface. On the left, a tree view shows the database structure, including 'postgres', 'public', and 'pg_catalog'. The main panel shows the configuration for the 'postgres' database, including the name, administrator, tablespace, and connection string. A 'pg_restore' command window is open in the foreground, showing the command 'pg_restore: creating FK CONSTRAINT "bookings.ticket_flights ticket_flights_fk4"' and a progress bar. A small error dialog box is also visible, stating 'Test PostgreSQL restore in separate database: Objects created postgres!'.

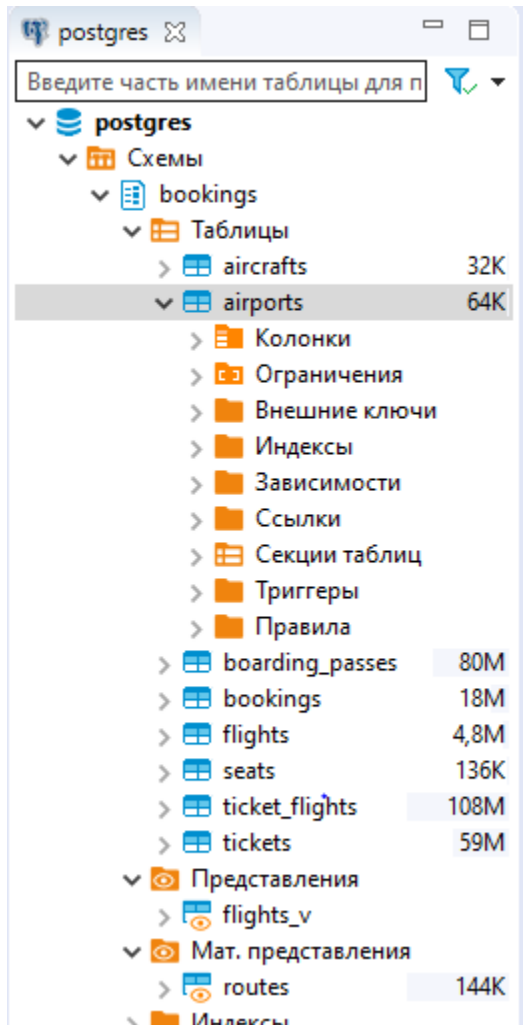
Восстановление заняло ~ 3 минуты 28 секунд.

2. Скриншот ER-диаграммы из DBeaver`а согласно Вашего подключения.



3. Краткое описание БД - из каких таблиц и представлений состоит. Естественно, вся информация представлена в файле bookings.pdf. Можно добавить скрин из программы DBeaver, где представлены все

таблицы и представления, в том числе и материализованные представления.



Конечно, можно написать код, который мы выполняли в домашних заданиях, находя все таблицы и ключи. Но уже из вышепредставленного рисунка видно, что у нас восемь таблиц, одно представление и одно материализованное представление. Это всё соответствует ER-диаграмме из пункта 2, естественно. Единственное, что из этого скриншота можно явно отличить представление `flights_v` от материализованного представления `routes`.

4. Развернутый анализ БД

Основной сущностью здесь является бронирование (`bookings`).

В одно бронирование можно включить несколько пассажиров, каждому из которых выписывается отдельный *билет* (tickets). Как таковой пассажир не является отдельной сущностью: для простоты можно считать, что все пассажиры уникальны.

Билет включает один или несколько *перелетов* (ticket_flights). Несколько перелетов могут включаться в билет в нескольких случаях:

1. Нет прямого рейса, соединяющего пункты отправления и назначения (полет с пересадками);
2. Взят билет «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 имеет только одну компоновку салона. Схема данных не контролирует, что места в посадочных талонах соответствуют имеющимся в самолете.

База довольно обширна, следует по мере выполнения задач расширять своё знание о базе, под конкретную задачу.

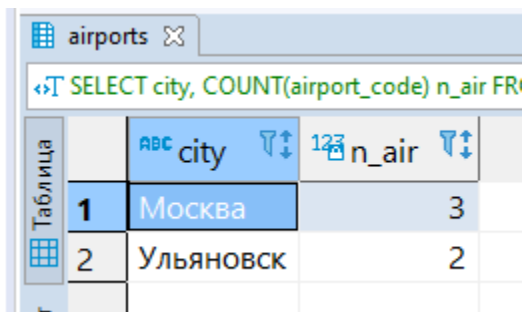
5. Список SQL-запросов с описанием логики их выполнения.

5.1 В каких городах больше одного аэропорта?

Воспользуемся таблицей bookings.airports. Нам понадобится данные о городе – столбец city, который может служить для того, чтобы определить аэропорты одного города. В качестве идентификаторов аэропортов служит трехбуквенный код (airport_code). Выведем город city, количество аэропортов в городе count(airport_code) из нашей таблицы, группируя данные по городам, с количеством аэропортов в городе больше одного count(airport_code) > 1. Результат можно упорядочить по трехбуквенному коду, по алфавиту.

Следующий SQL-запрос отвечает на поставленный вопрос:

```
SELECT
    city,
    COUNT(airport_code) n_air
FROM
    bookings.airports a
GROUP BY
    city
HAVING
    COUNT(airport_code) > 1
ORDER BY
    city;
```



	city	n_air
1	Москва	3
2	Ульяновск	2

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

Здесь дополнительное ограничение, что самолёты совершают максимальную дальность перелёта целесообразно выполнить отдельным подзапросом:

```
SELECT
    aircraft_code
FROM
    bookings.aircrafts a
WHERE range = (SELECT MAX(range) FROM bookings.aircrafts);
```

aircrafts	
SELECT aircraft_code FROM booki	
Таблица	aircraft_code
1	773

После уже подзапрос можно сравнивать со всеми трехзначными кодами модели воздушного судна в условии where основного запроса в качестве дополнительного условия на максимальную дальность перелёта. И останется вывести данные в выходную таблицу с указанием нужной информации с учётом дополнительных условий: имя аэропорта, трехбуквенный код идентифицирующий аэропорт из таблицы аэропортов и модель самолёта из таблицы самолётов.

Соответствующий SQL-запрос:

```

SELECT
    DISTINCT ap.airport_name,
    ap.airport_code,
    a.model
FROM
    bookings.flights f
LEFT JOIN bookings.airports ap ON
    f.departure_airport = ap.airport_code
    OR f.arrival_airport = ap.airport_code
LEFT JOIN bookings.aircrafts a ON
    f.aircraft_code = a.aircraft_code
WHERE
    f.aircraft_code = (
        SELECT
            aircraft_code
        FROM
            bookings.aircrafts a
        WHERE
            range = (
                SELECT
                    MAX(range)
                FROM
                    bookings.aircrafts));

```

airports(+)

SELECT DISTINCT ap.airport_name, ap.airport_code, a.model FROM bookings.flights f LI

Таблица		ABC airport_name	ABC airport_code	ABC model
	1	Сочи	AER	Boeing 777-300
Текст	2	Домодедово	DME	Boeing 777-300
	3	Кольцово	SVX	Boeing 777-300
	4	Шереметьево	SVO	Boeing 777-300
	5	Внуково	VKO	Boeing 777-300
	6	Толмачёво	OVB	Boeing 777-300
	7	Пермь	PEE	Boeing 777-300

5.3 Вывести 10 рейсов с максимальным временем задержки вылета

В этой задаче понадобится работа с временным типом данных. Для улучшенного восприятия переведем максимальное время задержки в стандартное отображение HH:MM:SS доступным для SQL-запросов способом. Работаем с таблицей flights, обращая чтобы значения в таблице по вылету не были f.actual_departure IS NOT NULL и для сокращения работы запроса, не учитывая ненулевые значения задержки вылета (f.actual_departure - f.scheduled_departure) != 0.

Соответствующий SQL-запрос:

```

SELECT
    scheduled_departure,
    actual_departure,
    ("delay_in_s"::varchar(24) || ' seconds')::interval as
delay_in_HMS
FROM
    (
        SELECT
            f.scheduled_departure, f.actual_departure,
ROUND((EXTRACT(EPOCH FROM (f.actual_departure -
f.scheduled_departure))) ::numeric, 1) AS delay_in_s
        FROM
            bookings.flights f
        WHERE
            f.actual_departure IS NOT NULL
            AND EXTRACT(EPOCH
FROM

```

```

        (f.actual_departure - f.scheduled_departure)) != 0
ORDER BY
    delay_in_s DESC
LIMIT 10) sub;

```

flights				
select scheduled_departure, actual_departure, ("delay_in_s"::varchar(24) ' seconds')::interval as delay_ir				
Таблица	1	2	3	4
	scheduled_departure	actual_departure	delay_in_hms	
1	2016-09-26 14:30:00	2016-09-26 19:07:00	04:37:00	
2	2016-09-26 14:25:00	2016-09-26 18:53:00	04:28:00	
3	2016-09-16 10:45:00	2016-09-16 15:12:00	04:27:00	
4	2016-10-11 15:15:00	2016-10-11 19:35:00	04:20:00	
5	2016-10-01 15:35:00	2016-10-01 19:53:00	04:18:00	
6	2016-09-30 09:05:00	2016-09-30 13:23:00	04:18:00	
7	2016-10-10 13:35:00	2016-10-10 17:51:00	04:16:00	
8	2016-09-13 13:20:00	2016-09-13 17:36:00	04:16:00	
9	2016-10-08 08:30:00	2016-10-08 12:44:00	04:14:00	
10	2016-09-16 15:15:00	2016-09-16 19:23:00	04:08:00	

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

Необходимые таблицы, естественно, это бронирования bookings и билеты tickets. Как говорится, начнём с дополнительного условия, остальное подтянется. Дополнительным условием будет условие, что посадочные талоны не были получены, фактически это описывается конструкцией bp.boarding_no IS NULL.

Соответствующий SQL-запрос:

```

SELECT
    DISTINCT b.book_ref, bp.boarding_no
FROM
    bookings.bookings b
RIGHT JOIN bookings.tickets t ON
    b.book_ref = t.book_ref
LEFT JOIN bookings.boarding_passes bp ON
    t.ticket_no = bp.ticket_no
WHERE
    bp.boarding_no IS NULL
ORDER BY
    b.book_ref;

```


bookings(+) X

SELECT DISTINCT b.book_ref, bp.boarding_no FROM bookin

	book_ref	boarding_no
1	000068	[NULL]
2	000181	[NULL]
3	0002D8	[NULL]
4	0002DB	[NULL]
5	00034E	[NULL]
6	000374	[NULL]
7	00044D	[NULL]
8	0004E1	[NULL]
9	00053F	[NULL]
10	0005F4	[NULL]
11	0006C3	[NULL]
12	0006F5	[NULL]
13	000836	[NULL]

Save Cancel Script

5.5 Найдите свободные места для каждого рейса, их % отношение к общему количеству мест в самолете.
 Добавьте столбец с накопительным итогом - суммарное количество вывезенных пассажиров из аэропорта за день. Т.е. в этом столбце должна отражаться сумма - сколько человек уже вылетело из данного аэропорта на этом или более ранних рейсах за день.

Следующим подзапросом

```

SELECT
    aircraft_code,
    COUNT(aircraft_code) t_seats
FROM
    bookings.seats s
GROUP BY
    aircraft_code

```

получим таблицу, где указан код всех самолётов, участвующих в перелёте с указанием количества мест в салоне. Здесь воспользовались тем, что под конкретный код самолёта с уникальным (здесь я пишу в описании действий, поэтому не использую в запросе distinct

😊) названием представлены в другой колонке уникальные номера мест и можно сгруппировав и подсчитав количество строк с конкретным значением кода самолёта получить и количество мест для данного типа самолёта. Данный подзапрос можно использовать в качестве алиаса для дальнейших вычислений.

	aircraft_code	t_seats
1	CN1	12
2	320	140
3	CR2	50
4	773	402
5	763	222
6	319	116
7	733	130
8	SU9	97
9	321	170

Также необходимые данные находятся в представлении "bookings.flights_v"

SQL-запрос:

SELECT

```

fv.flight_id,
fv.flight_no,
fv.departure_airport_name dep_name,
fv.arrival_airport_name arr_name,
fv.aircraft_code ac,
fv.actual_departure act_depart,
COUNT (fv.flight_id) onboard,
ts.t_seats, -- полученное в подзапросе общее количество мест в
самолётах
ts.t_seats - COUNT (fv.flight_id) e_seats,
ROUND(100 * (1 - COUNT(fv.flight_id) / ts.t_seats::numeric), 2)
realseats_percent,
SUM(COUNT (fv.flight_id))
OVER(PARTITION BY fv.departure_airport_name,
DATE(fv.actual_departure)
ORDER BY fv.actual_departure
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
sumpass_day

```

```

-- The frame, ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT
ROW,
-- means that the window consists of the first row of the
partition
-- and all the rows up to the current row.
-- Each calculation is done over a different set of rows.
-- For example, when performing the calculation for row 4,
the rows 1 to 4 are used.
--
-- https://www.red-gate.com/simple-talk/sql/t-sql-
programming/introduction-to-t-sql-window-functions/
FROM bookings.flights_v fv
LEFT JOIN bookings.ticket_flights tf ON
    fv.flight_id = tf.flight_id
RIGHT JOIN bookings.boarding_passes bp ON
    tf.ticket_no = bp.ticket_no AND tf.flight_id = bp.flight_id
LEFT JOIN (
SELECT
    aircraft_code,
    COUNT(aircraft_code) t_seats
FROM
    bookings.seats s
GROUP BY
    aircraft_code) ts ON
    fv.aircraft_code = ts.aircraft_code
GROUP BY
    fv.flight_id,
    fv.flight_no,
    fv.departure_airport_name,
    fv.arrival_airport_name,
    fv.aircraft_code,
    ts.t_seats,
    fv.actual_departure
-- фактическое время вылета должно быть, то есть полёт состоялся.
HAVING fv.actual_departure IS NOT NULL;

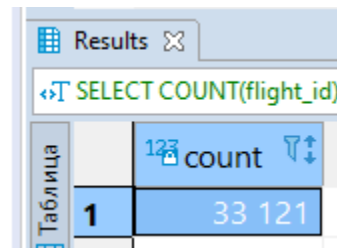
```

flights_v												
SELECT fv.flight_id, fv.flight_no, fv.departure_airport_name dep_name, fv.arrival_airport_name arr_name, fv.ac, fv.act_depart, fv.onboard, fv.t_seats, fv.e_seats, fv.realseats_percent, fv.sumpass_day												
flight_id	flight_no	dep_name	arr_name	ac	act_depart	onboard	t_seats	e_seats	realseats_percent	sumpass_day		
1	18 263	Абакан	Домодедово	319	2016-09-13 06:35:00	6	116	110	94,83	6		
2	18 337	PG0585	Абакан	Толмачёво	CN1	2016-09-14 08:37:00	2	12	10	83,33	2	
3	18 462	PG0070	Абакан	Богашёво	CN1	2016-09-14 11:10:00	3	12	9	75	5	
4	18 304	PG0585	Абакан	Толмачёво	CN1	2016-09-15 08:39:00	3	12	9	75	3	
5	18 443	PG0070	Абакан	Богашёво	CN1	2016-09-15 11:08:00	4	12	8	66,67	7	
6	18 274	PG0520	Абакан	Домодедово	319	2016-09-16 06:37:00	20	116	96	82,76	20	
7	18 437	PG0070	Абакан	Богашёво	CN1	2016-09-16 11:07:00	2	12	10	83,33	22	
8	18 347	PG0585	Абакан	Толмачёво	CN1	2016-09-17 08:39:00	2	12	10	83,33	2	
9	18 478	PG0070	Абакан	Богашёво	CN1	2016-09-17 11:08:00	2	12	10	83,33	4	
10	18 292	PG0585	Абакан	Толмачёво	CN1	2016-09-18 08:37:00	3	12	9	75	3	
11	18 450	PG0070	Абакан	Богашёво	CN1	2016-09-18 11:07:00	4	12	8	66,67	7	
12	18 284	PG0585	Абакан	Толмачёво	CN1	2016-09-19 08:36:00	2	12	10	83,33	2	
13	18 451	PG0070	Абакан	Богашёво	CN1	2016-09-19 14:10:00	2	12	10	83,33	4	

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

Необходимо найти общее количество перелетов поделить к соотношению перелетов по типам перелетов и умножить на 100, чтобы получить процентное соотношение. Понадобится подзапрос, вычисляющий общее количество перелетов:

```
SELECT
    COUNT(flight_id)
FROM
    bookings.flights f2
```



The screenshot shows a database query results window titled 'Results'. The query entered is 'SELECT COUNT(flight_id)'. The results are displayed in a table with one column labeled 'count' and one row with the value '33 121'.

count
33 121

SQL-запрос:

```
SELECT
    f.aircraft_code,
    a.model,
    ROUND(100.0 * COUNT(f.flight_id) / (SELECT COUNT(flight_id) FROM
bookings.flights f2), 2) AS aircraft_percent
FROM
    bookings.flights f
LEFT JOIN bookings.aircrafts a ON
    f.aircraft_code = a.aircraft_code
GROUP BY
    f.aircraft_code,
    a.model;
```

flights(+)			
SELECT f.aircraft_code, a.model, ROUND(100.0 * COUNT(f.flight_id) / (SELECT COUNT(flight_id) f			
Таблица	aircraft_code	model	aircraft_percent
1	763	Boeing 767-300	3,69
2	SU9	Sukhoi SuperJet-100	25,68
3	321	Airbus A321-200	5,89
4	319	Airbus A319-100	3,74
5	CN1	Cessna 208 Caravan	28
6	733	Boeing 737-300	3,85
7	CR2	Bombardier CRJ-200	27,32
8	773	Boeing 777-300	1,84

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

Нужно воспользоваться CTE!

CTE играет роль представления, которое создается в рамках одного запроса и, не сохраняется как объект схемы. Имеет свой синтаксис, начинающийся с WITH...

У каждого полёта есть свой УНИКАЛЬНЫЙ идентификатор flight_id с разным классом обслуживания fare_conditions и разлётом цен (минимальной и максимальной стоимостью перелёта amount из таблицы ticket_flights) в пределах одного полёта по классам обслуживания.

Создадим общее табличное выражение (CTE), где в операторе HAVING задали, что имеем дело только с эконом- и бизнес-классом (всего у нас три класса обслуживания, третий отбрасываем, также нам важно, чтобы у нас во время перелёта были представлены и эконом- и бизнес-класс, фактически каждому идентификатору рейса соответствуют ровно два рассматриваемых класса обслуживания).

WITH prices AS

(

SELECT

f.flight_id,
tf.fare_conditions,
MIN(amount) min_amount,
MAX(amount) max_amount

FROM bookings.flights f

RIGHT JOIN bookings.ticket_flights tf **ON** f.flight_id = tf.flight_id

GROUP BY

f.flight_id,
tf.fare_conditions

HAVING

```
tf.fare_conditions != 'Comfort' AND COUNT(tf.fare_conditions) > 1
)
```

Выведем результаты запроса данного общего табличного выражения, то есть то, что находится в круглых скобках оператора WITH AS ():

flights(+)					
SELECT f.flight_id, tf.fare_conditions, MIN(amount) min_amount, MAX(amount) max_amount FROM bc Введи					
Таблица	flight_id	fare_conditions	min_amount	max_amount	
1	1	Business	20 000	20 000	
2	1	Economy	6 700	7 300	
3	2	Business	20 000	20 000	
4	2	Economy	6 700	7 300	
5	3	Business	20 000	20 000	
6	3	Economy	6 700	7 300	
7	5	Business	20 000	20 000	
8	5	Economy	6 700	7 300	
9	6	Business	20 000	20 000	
10	6	Economy	6 700	7 300	
11	9	Business	20 000	20 000	
12	9	Economy	6 700	7 300	
13	12	Business	20 000	20 000	

Save Cancel Script 200 400+

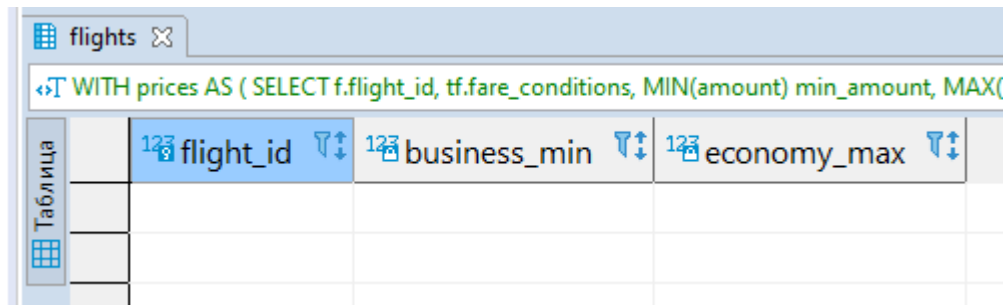
Полный SQL-запрос с учётом вышепредставленного оператора:

```
WITH prices AS
(
SELECT
    f.flight_id,
    tf.fare_conditions,
    MIN(amount) min_amount,
    MAX(amount) max_amount
FROM bookings.flights f
RIGHT JOIN bookings.ticket_flights tf ON f.flight_id = tf.flight_id
GROUP BY
    f.flight_id,
    tf.fare_conditions
HAVING
    tf.fare_conditions != 'Comfort' AND COUNT(tf.fare_conditions) > 1
)
-- Финальный запрос, использующий CTE
```

SELECT

```
    b.flight_id,  
    b.min_amount business_min,  
    e.max_amount economy_max  
FROM (SELECT * FROM prices WHERE fare_conditions = 'Business') b  
LEFT JOIN (SELECT * FROM prices WHERE fare_conditions = 'Economy') e  
ON b.flight_id = e.flight_id  
-- условие на сравнение цен в бизнес- и эконом- классе  
WHERE b.min_amount < e.max_amount;
```

Стоит обратить на последнюю строчку данного запроса, здесь сравниваем стоимость перелёта на условие: 'бизнес - классом дешевле, чем эконом-классом в рамках перелета?'.
Видно, что никаких перелётов с требуемыми условиями не осуществлялось. Если заменить знак < в операторе на знак =, также будет пустой вывод. А вот если поставить знак > вместо знака <, то естественно высветятся все варианты и можно смотреть города прибытия, соответствующие городу прибытия arrival_city с помощью представления "bookings.flights_v". Но поскольку мы ратуем за быстрое выполнение кода, то нет необходимости искать города прибытия.



flight_id	business_min	economy_max
-----------	--------------	-------------

По факту раз нет полётов, с нас не требуют названия городов прибытия, то городов, соответствующему нашему вопросу не было. Бизнес-классом лететь всегда дороже, чем эконом-классом.

5.8 Между какими городами нет прямых рейсов?

Следует найти всевозможные рейсы, прямые рейсы и вычесть полученные множества. Это и будет ответом.

Полный SQL-запрос:

```
-- в данное представление dep_city перенесены всевозможные города  
--отправления из которых осуществляется вылет.  
CREATE OR REPLACE VIEW bookings.dep_city AS (  
SELECT DISTINCT city dep_city  
FROM bookings.flights f  
LEFT JOIN bookings.airports a ON f.departure_airport = a.airport_code  
ORDER BY city);
```

```

-- в данное представление arr_city перенесены всевозможные города
-- прибытия, в которые осуществляется прилёт.
CREATE OR REPLACE VIEW bookings.arr_city AS (
SELECT DISTINCT city arr_city
FROM bookings.flights f
LEFT JOIN bookings.airports a
ON f.arrival_airport = a.airport_code
ORDER BY city);

-- все комбинации возможных перелётов
SELECT CONCAT (dep_city, ' ', arr_city) all_possible_flights
FROM (
SELECT * FROM bookings.dep_city dc, bookings.arr_city ac
WHERE dep_city != arr_city) all_possible;

-- существующие прямые перелёты
SELECT CONCAT (departure_city, ' ', arrival_city) existing_flights
FROM (
SELECT DISTINCT
    a1.city departure_city,
    a2.city arrival_city
FROM bookings.flights f
LEFT JOIN bookings.airports a1
ON f.departure_airport = a1.airport_code
LEFT JOIN bookings.airports a2
ON f.arrival_airport = a2.airport_code
ORDER BY a1.city)
    existing_only;

WITH non_existing_direct AS (
-- выбираем всевозможные прямые перелёты
SELECT CONCAT (dep_city, ' ', arr_city) possible_flights
FROM (
SELECT * FROM bookings.dep_city dc, bookings.arr_city ac
WHERE dep_city != arr_city) all_possible
EXCEPT
-- вычитаем существующие прямые перелёты
SELECT CONCAT (departure_city, ' ', arrival_city) existing_flights
FROM (
SELECT DISTINCT
    a1.city departure_city,
    a2.city arrival_city
FROM bookings.flights f
LEFT JOIN bookings.airports a1
ON f.departure_airport = a1.airport_code

```



```

LEFT JOIN bookings.airports a2
ON f.arrival_airport = a2.airport_code
ORDER BY a1.city) existing_only)
SELECT * FROM non_existing_direct
ORDER BY possible_flights;

```

CREATE OR REPLACE VIEW bookings.dep_city AS (SELECT

Таблица	all_possible_flights
1	Абакан Анадырь
2	Абакан Анапа
3	Абакан Архангельск
4	Абакан Астрахань
5	Абакан Барнаул
6	Абакан Белгород
7	Абакан Белоярский
8	Абакан Благовещенск
9	Абакан Братск
10	Абакан Брянск
11	Абакан Бугульма
12	Абакан Владивосток
13	Абакан Владикавказ
14	Абакан Волгоград
15	Абакан Воркута
16	Абакан Воронеж
17	Абакан Геленджик
18	Абакан Горно-Алтайск
19	Абакан Грозный
20	Абакан Екатеринбург
21	Абакан Иваново
22	Абакан Ижевск
23	Абакан Иркутск
24	Абакан Йошкар-Ола
25	Абакан Казань
26	Абакан Калининград
27	Абакан Калуга
28	Абакан Кемерово
29	Абакан Киров
30	Абакан Когалым
31	Абакан Комсомольск-на-Амуре
32	Абакан Краснодар
33	Абакан Красноярск
34	Абакан Курган
35	Абакан Курск
36	Абакан Кызыл
37	Абакан Липецк
38	Абакан Магадан
39	Абакан Магнитогорск
40	Абакан Махачкала
41	Абакан Минеральные Воды
42	Абакан Мирный
43	Абакан Москва
44	Абакан Мурманск
45	Абакан Надым
46	Абакан Нальчик

Запись

Множество 1
Всевозможные комбинации

CREATE OR REPLACE VIEW bookings.dep_city AS (SELECT

Таблица	existing_flights
1	Абакан Грозный
2	Абакан Москва
3	Абакан Новосибирск
4	Абакан Кызыл
5	Абакан Томск
6	Абакан Архангельск
7	Анадырь Москва
8	Анадырь Хабаровск
9	Анапа Белгород
10	Анапа Москва
11	Анапа Новокузнецк
12	Архангельск Пермь
13	Архангельск Томск
14	Архангельск Иркутск
15	Архангельск Ханты-Мансийск
16	Архангельск Москва
17	Архангельск Абакан
18	Архангельск Нарьян-Мар
19	Архангельск Тюмень
20	Астрахань Нерюнгри
21	Астрахань Барнаул
22	Астрахань Москва
23	Барнаул Астрахань
24	Барнаул Якутск
25	Барнаул Красноярск
26	Барнаул Москва
27	Белгород Москва
28	Белгород Анапа
29	Белгород Сочи
30	Белгород Брянск
31	Белгород Ростов-на-Дону
32	Белоярский Москва
33	Белоярский Тюмень
34	Белоярский Курган
35	Благовещенск Хабаровск
36	Братск Москва
37	Брянск Белгород
38	Брянск Москва
39	Бугульма Оренбург
40	Бугульма Москва
41	Владивосток Иркутск
42	Владивосток Москва
43	Владивосток Хабаровск
44	Владикавказ Москва
45	Волгоград Москва
46	Волгоград Челябинск

Запись

Множество 2
Реальные прямые рейсы

CREATE OR REPLACE VIEW bookings.dep_city AS (SELECT

Таблица	possible_flights
1	Абакан Анадырь
2	Абакан Анапа
3	Абакан Астрахань
4	Абакан Барнаул
5	Абакан Белгород
6	Абакан Белоярский
7	Абакан Благовещенск
8	Абакан Братск
9	Абакан Брянск
10	Абакан Бугульма
11	Абакан Владивосток
12	Абакан Владикавказ
13	Абакан Волгоград
14	Абакан Воркута
15	Абакан Воронеж
16	Абакан Геленджик
17	Абакан Горно-Алтайск
18	Абакан Екатеринбург
19	Абакан Иваново
20	Абакан Ижевск
21	Абакан Иркутск
22	Абакан Йошкар-Ола
23	Абакан Казань
24	Абакан Калининград
25	Абакан Калуга
26	Абакан Кемерово
27	Абакан Киров
28	Абакан Когалым
29	Абакан Комсомольск-на-Амуре
30	Абакан Краснодар
31	Абакан Красноярск
32	Абакан Курган
33	Абакан Курск
34	Абакан Липецк
35	Абакан Магадан
36	Абакан Магнитогорск
37	Абакан Махачкала
38	Абакан Минеральные Воды
39	Абакан Мирный
40	Абакан Мурманск
41	Абакан Надым
42	Абакан Нальчик
43	Абакан Нарьян-Мар
44	Абакан Нерюнгри
45	Абакан Нефтеюганск
46	Абакан Нижневартовск

Запись

Ответ
Mn1\Mn2

5.9 Вычислите расстояние между аэропортами, связанными прямыми рейсами, сравните с допустимой максимальной дальностью перелетов в самолетах, обслуживающих эти рейсы В локальной базе координаты находятся в столбцах airports.longitude и airports.latitude.

Для решения поставленной задачи нужно взять прямые рейсы, вычислить по геоданным из таблиц расстояния между городами и сравнить с допустимой максимальной дальностью. Если происходит превышение, то подобные полёты не допустимо совершать.

Соответствующий SQL-запрос и логика выполнения:

```
WITH dist_range_comparison AS (  
  -- широта и долгота аэропорта  
  WITH ap_lat_lon AS (  
    SELECT  
      dep_air,  
      dep_name,  
      arr_air,  
      arr_name,  
      ac,  
      lat_a,  
      lon_a,  
      latitude lat_b,  
      longitude lon_b  
  FROM  
    (  
      SELECT  
        r.departure_airport dep_air,  
        r.departure_airport_name dep_name,  
        r.arrival_airport arr_air,  
        r.arrival_airport_name arr_name,  
        r.aircraft_code ac,  
        a.latitude lat_a,  
        a.longitude lon_a  
      FROM  
        bookings.routes r  
      -- широта и долгота аэропорта отправления  
      LEFT JOIN bookings.airports a ON  
        r.departure_airport = a.airport_code) dep_lat_lon  
      -- широта и долгота аэропорта вылета и прилёта  
      LEFT JOIN bookings.airports a2 ON  
        dep_lat_lon.arr_air = a2.airport_code)  
    SELECT  
      dep_air,  
      dep_name,  
      arr_air,  
      arr_name,
```

```

        model,
        ROUND((acos(sin(radians(lat_a))*sin(radians(lat_b))+cos(radians(lat_a))*cos(radians(lat_b))*cos(radians(lon_b - lon_a))) *
6371)::numeric, 1) dist_km,
        "range"
FROM ap_lat_lon
LEFT JOIN bookings.aircrafts ac
ON ap_lat_lon.ac = ac.aircraft_code)
SELECT
    *,
    (CASE
        WHEN "range" - dist_km > 100 THEN 'Допустимо'
        WHEN "range" - dist_km <= 100 AND "range" - dist_km > 0 THEN
'На пределе'
        WHEN "range" - dist_km <= 0 THEN 'Не допустимо'
    END) flight_admissibility
FROM dist_range_comparison
-- WHERE "range" - dist_km <= 0
ORDER BY dist_km DESC;

```

routes(+)									
WITH dist_range_comparison AS (WITH ap_lat_lon AS (SELECT dep_air, dep_name, arr_air, arr_name, ac Введите SQL выражение чтобы отфильтровать результаты									
	dep_air	dep_name	arr_air	arr_name	model	dist_km	range	flight_admissibility	
1	DME	Домодедово	PKC	Елизово	Boeing 767-300	6 774,7	7 900	Допустимо	
2	PKC	Елизово	DME	Домодедово	Boeing 767-300	6 774,7	7 900	Допустимо	
3	GDH	Магадан	MRV	Минеральные Воды	Boeing 767-300	6 711,1	7 900	Допустимо	
4	MRV	Минеральные Воды	GDH	Магадан	Boeing 767-300	6 711,1	7 900	Допустимо	
5	UUS	Хомутово	DME	Домодедово	Airbus A319-100	6 658,5	6 700	На пределе	
6	DME	Домодедово	UUS	Хомутово	Airbus A319-100	6 658,5	6 700	На пределе	
7	UUS	Хомутово	SVO	Шереметьево	Airbus A319-100	6 641,5	6 700	На пределе	
8	SVO	Шереметьево	UUS	Хомутово	Airbus A319-100	6 641,5	6 700	На пределе	
9	VKO	Внуково	VVO	Владивосток	Boeing 767-300	6 434,6	7 900	Допустимо	
10	VVO	Владивосток	VKO	Внуково	Boeing 767-300	6 434,6	7 900	Допустимо	
11	DYR	Анадырь	DME	Домодедово	Airbus A319-100	6 226	6 700	Допустимо	
12	DME	Домодедово	DYR	Анадырь	Airbus A319-100	6 226	6 700	Допустимо	
13	VKO	Внуково	DYR	Анадырь	Airbus A319-100	6 220,2	6 700	Допустимо	
14	DYR	Анадырь	VKO	Внуково	Airbus A319-100	6 220,2	6 700	Допустимо	
15	LED	Пулково	KHV	Хабаровск-Новый	Boeing 767-300	6 207	7 900	Допустимо	
16	KHV	Хабаровск-Новый	LED	Пулково	Boeing 767-300	6 207	7 900	Допустимо	
17	DYR	Анадырь	SVO	Шереметьево	Airbus A319-100	6 177,1	6 700	Допустимо	
18	SVO	Шереметьево	DYR	Анадырь	Airbus A319-100	6 177,1	6 700	Допустимо	
19	KHV	Хабаровск-Новый	DME	Домодедово	Boeing 767-300	6 150	7 900	Допустимо	
20	DME	Домодедово	KHV	Хабаровск-Новый	Boeing 767-300	6 150	7 900	Допустимо	
21	ASF	Астрахань	CNN	Чульман	Airbus A319-100	5 147	6 700	Допустимо	
22	CNN	Чульман	ASF	Астрахань	Airbus A319-100	5 147	6 700	Допустимо	
23	LED	Пулково	CNN	Чульман	Airbus A319-100	5 050,8	6 700	Допустимо	
24	CNN	Чульман	LED	Пулково	Airbus A319-100	5 050,8	6 700	Допустимо	
25	CNN	Чульман	DME	Домодедово	Airbus A319-100	5 014,5	6 700	Допустимо	
26	DME	Домодедово	CNN	Чульман	Airbus A319-100	5 014,5	6 700	Допустимо	
27	SCW	Сыктывкар	GDH	Магадан	Boeing 767-300	4 880,8	7 900	Допустимо	
28	GDH	Магадан	SCW	Сыктывкар	Boeing 767-300	4 880,8	7 900	Допустимо	
29	YKS	Якутск	LED	Пулково	Airbus A319-100	4 841,5	6 700	Допустимо	
30	LED	Пулково	YKS	Якутск	Airbus A319-100	4 841,5	6 700	Допустимо	
31	KXK	Хурба	SVX	Кольцово	Boeing 767-300	4 811	7 900	Допустимо	
32	SVX	Кольцово	KXK	Хурба	Boeing 767-300	4 811	7 900	Допустимо	
33	VKO	Внуково	UUD	Байкал	Airbus A319-100	4 439,2	6 700	Допустимо	
34	UUD	Байкал	VKO	Внуково	Airbus A319-100	4 439,2	6 700	Допустимо	
35	LED	Пулково	IKT	Иркутск	Airbus A321-200	4 431	5 600	Допустимо	
36	IKT	Иркутск	LED	Пулково	Airbus A321-200	4 431	5 600	Допустимо	
37	MJZ	Мирный	SVO	Шереметьево	Boeing 737-300	4 151,9	4 200	На пределе	
38	SVO	Шереметьево	MJZ	Мирный	Boeing 737-300	4 151,9	4 200	На пределе	
39	PEE	Пермь	CNN	Чульман	Airbus A319-100	3 945,4	6 700	Допустимо	
40	CNN	Чульман	PEE	Пермь	Airbus A319-100	3 945,4	6 700	Допустимо	
41	KJA	Емельяново	AER	Сочи	Airbus A319-100	3 913,1	6 700	Допустимо	
42	AER	Сочи	KJA	Емельяново	Airbus A319-100	3 913,1	6 700	Допустимо	
43	PYJ	Полярный	LED	Пулково	Boeing 737-300	3 873,2	4 200	Допустимо	
44	LED	Пулково	PYJ	Полярный	Boeing 737-300	3 873,2	4 200	Допустимо	
45	DME	Домодедово	BTX	Братск	Airbus A319-100	3 834,3	6 700	Допустимо	
46	BTX	Братск	DME	Домодедово	Airbus A319-100	3 834,3	6 700	Допустимо	

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

$d = \arccos \{ \sin(\text{latitude_a}) \cdot \sin(\text{latitude_b}) + \cos(\text{latitude_a}) \cdot \cos(\text{latitude_b}) \cdot \cos(\text{longitude_a} - \text{longitude_b}) \}$, 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:

$L = d \cdot R$, где $R = 6371$ км — средний радиус земного шара.