**Алексеев Д.П. (DSU-4)**

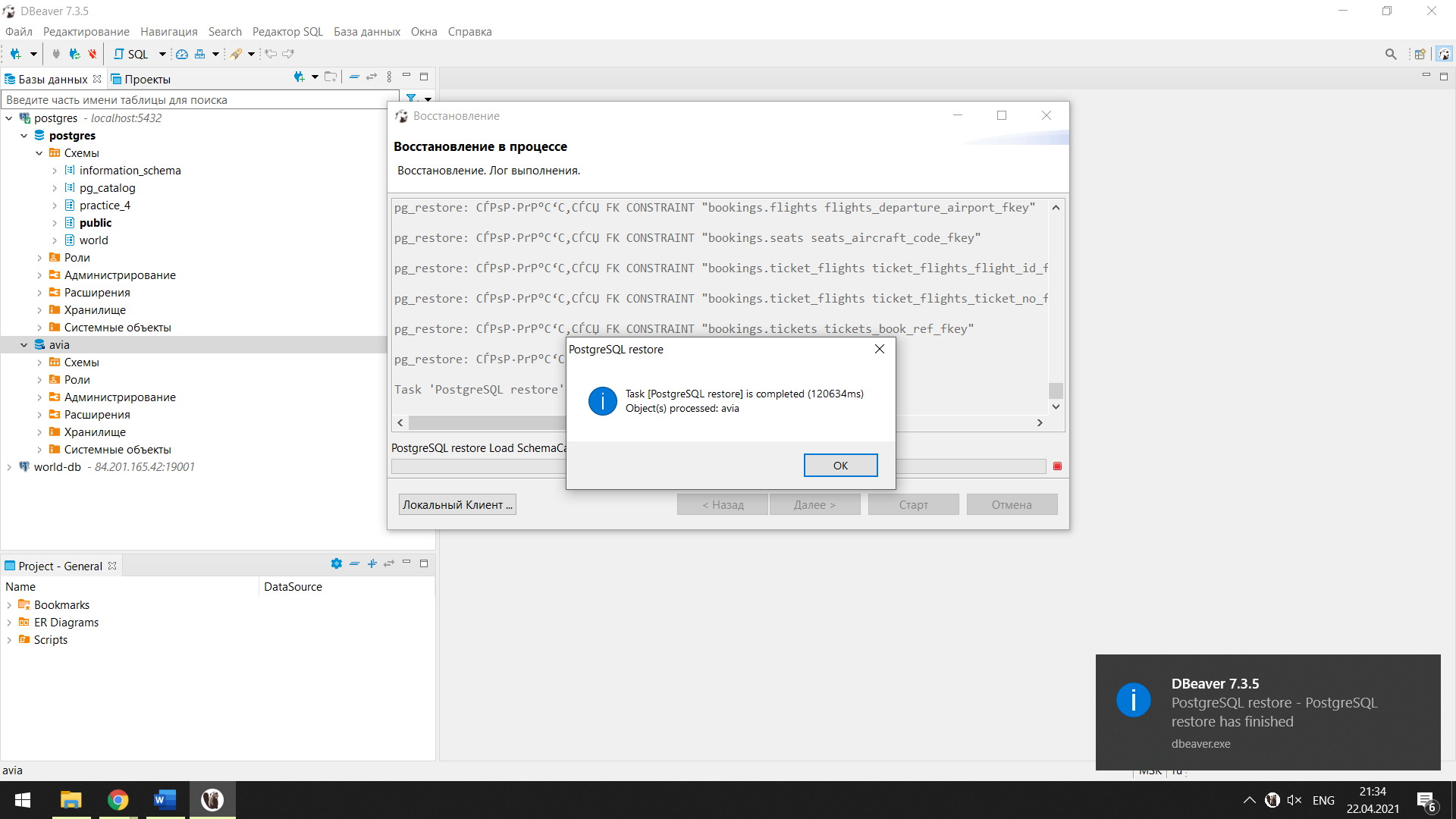
**Проектная работа по модулю “SQL и получение данных” (SQL-27)**

Приложение №1

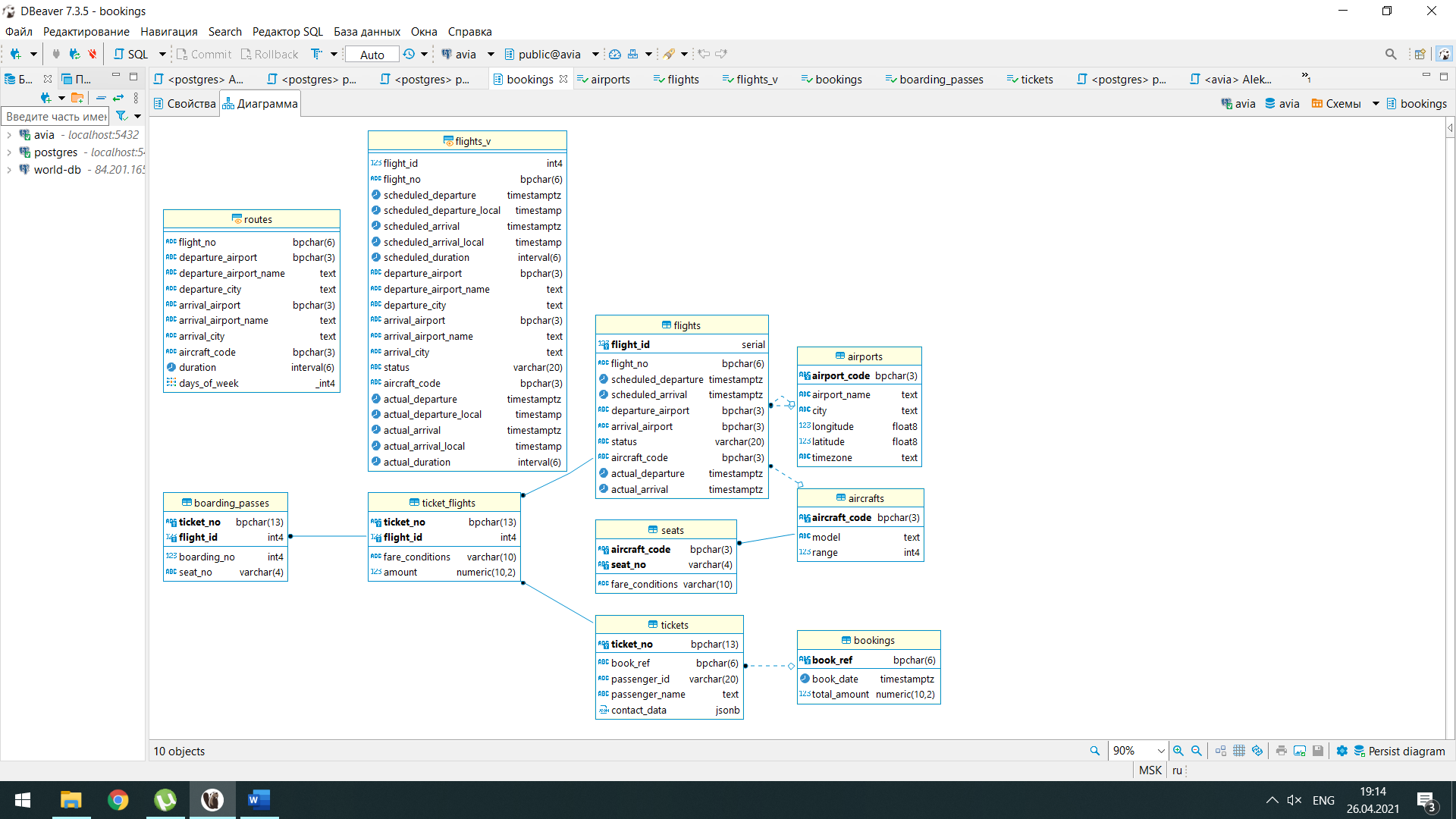
Итоговая работа

1. **В работе использовался локальный тип подключения (восстановление БД из \*.backup файла).** *DBeaver ver.7.3.5 (с DBeaver ver.21 решил пока не экспериментировать)*

* **если база была развернута из \*.sql или \*.backup файла, необходимо приложить скриншот успешного импорта или восстановления**



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



1. **Краткое описание БД - из каких таблиц и представлений состоит:**
2. Таблица «Flights» - содержит информацию о конкретных перелётах;
3. Таблица «Airports» - содержит информацию об аэропортах;
4. Таблица «Aircrafts» - содержит информацию о характеристиках воздушных судов (ВС);
5. Таблица «Seats» - содержит информацию о характеристиках мест в конкретной модели ВС;
6. Таблица «Bookings» - содержит информацию о бронированиях;
7. Таблица «Tickets» - содержит информацию о проданных билетах в привязке к бронированиям (в одном бронировании может быть несколько билетов);
8. Таблица «Ticket\_Flights» - содержит информацию о проданных билетах на конкретные перелеты (flights);
9. Таблица «Boarding\_Passes» - содержит информацию о выданных посадочных талонах в привязке к номерам билетов и номерам перелетов.
10. Представление «Flights\_V» - рассчитывается динамически, содержит сводную («статистическую») информацию о всех перелетах, удобно использовать для анализа перелетов или формирования расписания рейсов;
11. Представление «Routes» - материализованное представление (содержимое сохраняется как таблица), содержит сводную информацию о маршрутах (в привязке к номерам рейсов), в т.ч. содержит информацию о днях недели, когда выполняются рейсы, и плановой продолжительности полета.
12. **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.**
13. Таблица «Flights» - содержит информацию о перелётах:

* порядковый номер (идентификатор) перелета flight\_id, является первичным ключом;
* номер рейса flight\_no (в международной нотации);
* аэропорты отправления и прибытия (в международной нотации) departure\_airport и arrival\_airport соответственно;
* запланированное (по расписанию) время отправления и прибытия, фактическое время отправления и прибытия, код воздушного судна (в международной нотации), статус рейса (прибыл, отложен и т.д.).

1. Таблица «Airports» - содержит информацию об аэропортах:

* код аэропорта (в международной нотации, он же первичный ключ);
* название аэропорта;
* город местоположения,
* координаты аэропорта (долгота, широта);
* часовой пояс по отношению к Гринвичу.

1. Таблица «Aircrafts» - содержит информацию о характеристиках воздушных судов (ВС):

* код ВС (в международной нотации, он же первичный ключ),
* модель ВС;
* максимальная дальность полета ВС.

1. Таблица «Seats» - содержит информацию о характеристиках мест в привязке к конкретной модели ВС (aircraft\_code): номере места (seat\_no) и классе места (бизнес, комфорт, эконом). Первичный ключ – составной (aircraft\_code; seat\_no).
2. Таблица «Bookings» - содержит информацию о бронированиях; является «основной» сущностью, состоит из полей:

* Номер брони (book\_ref), он же первичный ключ;
* Дата бронирования;
* Общая сумма бронирования.

1. Таблица «Tickets» - содержит информацию о проданных билетах в привязке к бронированиям (в одном бронировании может быть несколько билетов):

* Уникальный номер билета ticket\_no (он же первичный ключ);
* Ссылку на номер бронирования;
* Идентификатор пассажира;
* ФИО пассажира;
* Контактные данные пассажира (в формате json).

1. Таблица «Ticket\_Flights» - содержит информацию о проданных билетах на конкретные перелеты (flights). В одном билете может быть несколько перелетов (например, на «кривые» рейсы с пересадками, или билет куплен «туда-и-обратно»):

* Составной ключ (ticket\_no, flight\_id) – номер билета и номер перелета;
* Стоимость билета (amount);
* Класс обслуживания по билету (бизнес, комфорт, эконом): fare\_conditions;

1. Таблица «Boarding\_Passes» - содержит информацию о выданных посадочных талонах в привязке к номерам билетов и номерам перелетов. Данные в этой таблице заполняются только по факту регистрации пассажира на рейс (выдачи посадочного талона):

* Составной ключ (ticket\_no, flight\_id) – номер билета и номер перелета;
* Номер посадочного талона (boarding\_no);
* Номер места для пассажира (seat\_no).

1. Представление «Flights\_V» - рассчитывается динамически, содержит сводную («статистическую») информацию о всех перелетах, удобно использовать для анализа перелетов;
2. Представление «Routes» - материализованное представление (содержимое сохраняется как таблица), содержит сводную информацию о маршрутах (в привязке к номерам рейсов), в т.ч. содержит информацию о днях недели, когда выполняются рейсы, и плановой продолжительности полета.

**Примеры бизнес-задач, которые можно решить, используя БД:**

* Анализ заполняемости рейсов с принятием решений по результатам анализа (маркетинговые акции: скидки деньгами /бонусы милями и т.п.)
* Отслеживание динамики стоимости перелетов на одинаковых маршрутах;
* Наиболее доходные/убыточные направления в расчете на километр пути (исходя из выручки от продаж билетов);
* Анализ клиентской базы с последующим направлением рекламных предложений (на контакты пассажира) по ранее использованным им маршрутам;
* Сопоставление стоимости «прямых» (беспересадочных) и «кривых» рейсов;
* и другие реальные задачи, особенно при обогащении данными о пассажирах, получаемыми из других источников (например, из соцсетей, БД сотовых операторов;))

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

**--1) В каких городах больше одного аэропорта?**

**Логика запроса:** Выбираем города из таблицы аэропортов с группировкой по городу при условии, что город встречается более одного раза.

**select** ap.city

**from** airports ap

**group** **by** ap.city

**having** **count**(ap.city)>1;

**--2) В каких аэропортах есть рейсы, выполняемые самолетом с максимальной дальностью перелета?**

**Логика запроса:** Выбираем аэропорты вылета из таблицы перелетов (с группировкой по аэропорту вылета), объединяем с таблицей воздушных судов (по коду судна) при условии, что дальность перелета для данного кода судна «находится в» (в данном случае равна) выдаче подзапроса, находящего максимальное значение для дальности перелетов в таблице воздушных судов.

--В решении обязательно должно быть использовано:

--Подзапрос

**select** f.departure\_airport --актуально, если мы имеем ввиду под "выполняемыми рейсами" рейсы из тех аэропортов, из которых самолет может вылететь.

--Если же говорить о тех, где он может приземлиться, то добавляется ещё уровень группировки по f.arrival\_airport. В этом случае таблица сочетаний ("пар") аэропортов будет больше, но в ней будут встречаться всё те же 7 уникальных аэропортов, что логично (если данная модель самолета может взлететь в конкретном аэропорту, то может в нем и приземлиться).

**from** flights f

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

**where** ac."range" **in** (**select** **max**("range") **from** aircrafts)

**group** **by** f.departure\_airport;

**--3) Вывести 10 рейсов с максимальным временем задержки вылета**

**Логика запроса:** Выбираем конкретные перелеты с указанием рейса, т.к. для «абстрактного» рейса максимальное время задержки определять нецелесообразно (с одним кодом рейса существует множество перелетов), считаем максимальную дельту между временем фактического вылета и запланированного вылета. При этом берем только фактически начатые рейсы.

Группируем по перелетам, сортируем по рассчитанному времени максимальной задержки по убыванию, делаем отсечку по верхним 10-и значениям.

--В решении обязательно должно быть использовано:

--Оператор LIMIT

**select** f.flight\_id, f.flight\_no, **max**(f.actual\_departure - f.scheduled\_departure) **as** max\_delay, f.scheduled\_departure, f.actual\_departure

**from** flights f

**where** f.actual\_departure **is** **not** **null** --берем только фактически начатые рейсы

**group** **by** f.flight\_id

**order** **by** max\_delay **desc**

**limit** 10;

**--4) Были ли брони, по которым не были получены посадочные талоны?**

**Логика запроса:** Выбираем номер брони и номер посадочного талона из объединенной таблицы бронирований и билетов (по полю «номер брони») и присоединенной таблицы посадочных талонов (по полю «номер билета»), при условии, что номер посадочного талона = «пусто». В результате выводится список номеров бронирований, по которым посадочные талоны не выдавались.

--В решении обязательно должно быть использовано:

-- Верный тип JOIN

**select** b.book\_ref, bp.boarding\_no

**from** bookings b

**left** **join** tickets t **on** t.book\_ref = b.book\_ref

**left** **join** boarding\_passes bp **on** bp.ticket\_no = t.ticket\_no

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

**group** **by** b.book\_ref, bp.boarding\_no;

**--5) Найдите свободные места для каждого рейса, их % отношение к общему количеству мест в самолете.**

**Логика запроса (без добавления столбца с накоп.итогом):** свободные места рассчитываем для каждого конкретного перелета, с указанием номера рейса и запланированного времени вылета (для сортировки по времени, справочно), а также указываем аэропорт вылета.

Фактически занятые места считаем по проданным билетам (тут, конечно, возможны варианты, т.к. не все проданные билеты «заканчиваются» посадочными талонами – опоздавшие и т.п. не летят…). Для подсчета проданных билетов на конкретный перелет используем таблицу “ticket\_flights”.

В подзапросе считаем максимальную емкость (capacity) салона воздушного судна(ВС), группируем по коду ВС, объединяем по коду ВС (aircraft\_code) с результирующей таблицей.

В результирующей таблице выводятся проданные билеты, емкость салона ВС и расчетный показатель заполненности салона в %-тах с округлением до 2-х знаков после запятой.

--Добавьте столбец с накопительным итогом - суммарное накопление количества вывезенных пассажиров из каждого аэропорта на каждый день.

--Т.е. в этом столбце должна отражаться накопительная сумма - сколько человек уже вылетело из данного аэропорта на этом или более ранних рейсах за день.

--В решении обязательно должно быть использовано:

-- - Оконная функция

-- - Подзапросы

**select** tf.flight\_id, f.flight\_no, f.scheduled\_departure, f.departure\_airport,

**count**(tf.ticket\_no) **as** sold\_tickets,

cap.capacity,

**round**(**count**(tf.ticket\_no)/cap.capacity::**numeric**\*100, 1) **as** utilization\_percent--,

**--count(f.actual\_departure) over (partition by f.departure\_airport) --[Добавление столбца с накоп.итогом доделать не удалось. Удалось вывести только количество отправленных самолетов, а не пассажиров, в целом за весь период по аэропортам...]**

**from** ticket\_flights tf

**join** flights f **on** f.flight\_id = tf.flight\_id

**join** (

**select** a.aircraft\_code, **count**(s.seat\_no) **as** capacity

**from** aircrafts a

**join** seats s **on** s.aircraft\_code = a.aircraft\_code

**group** **by** a.aircraft\_code) cap **on** cap.aircraft\_code = f.aircraft\_code

**group** **by** tf.flight\_id, f.flight\_no, f.scheduled\_departure, cap.capacity, f.departure\_airport--, f.actual\_departure

**order** **by** f.scheduled\_departure

**--6) Найдите процентное соотношение перелетов по типам самолетов от общего количества.**

**Логика запроса:** *[не совсем понятно, перелетов всех (уже состоявшихся и ещё только запланированных), или же только состоявшихся? Посчитал от всех перелетов]*

В подзапросе посчитано общее количество перелетов («total»). Далее в основном запросе посчитано количество перелетов по конкретному виду ВС и рассчитано %-ное отношение к общему количеству перелетов с округлением до 2-х знаков после запятой.

--В решении обязательно должно быть использовано:

-- - Подзапрос

-- - Оператор ROUND

**select** f2.aircraft\_code,

**count**(f2.aircraft\_code) **as** total\_by\_aircraft,

(**select** **count**(\*) **from** flights) **as** total,

**round**(**count**(f2.aircraft\_code)/(**select** **count**(\*) **from** flights)::**numeric**\*100,2) **as** aircraft\_percentage

**from** flights f2

**group** **by** f2.aircraft\_code

**order** **by** total\_by\_aircraft **desc**

**--7) Были ли города, в которые можно добраться бизнес - классом дешевле, чем эконом-классом в рамках перелета?**

**Логика запроса:** в условии задачи определено, что сравниваем стоимости в рамках конкретного перелета, поэтому группируем в CTE по «flight\_id».

Общее табличное выражение названо как «flight\_conditions». В CTE рассчитываем

минимальную и максимальную стоимости по условиям размещения пассажира (fare\_conditions) в рамках конкретного перелета, при этом для группировки мы используем города (a2.city ) из таблицы аэропортов, объединяем по коду аэропорта прибытия (т.к. в условии «*города, в которые можно добраться*…»)

Далее используем CTE 2 раза:

* находим минимальную стоимость бизнес-класса в рамках конкретного перелета(flight\_id)
* находим максимальную стоимость эконом-класса в рамках конкретного перелета(flight\_id)

Объединяем, выводим в результирующей таблице для каждого перелета и города прибытия указанные выше суммы, затем считаем арифметическую разницу между самым дешевым бизнес-классом и самым дорогим экономом.

**Ответ по задаче №7 исходя из анализа выдачи запроса: разница между самым дешевым бизнес-классом и самым дорогим эконом-классом по конкретным перелетам составляет от +12700 до +135500 руб. Все значения положительны.**

**Следовательно, городов, в которые можно добраться бизнес - классом дешевле, чем эконом-классом в рамках перелета, не было.**

[Если же оценивать по всем перелетам в целом по рейсу(маршруту), то такие города были - например, Киров.]

--В решении обязательно должно быть использовано:

-- - CTE

**with** flight\_conditions **as** (

--если именно в рамках конкретного перелета, а не среди всех перелетов:

**select** tf.flight\_id, a2.city, tf.fare\_conditions, **min**(tf.amount) **as** min\_amount, **max**(tf.amount) **as** max\_amount

--(продолжение: если же надо найти среди всех перелетов, то группировку по tf.flight\_id необходимо убрать)

**from** flights f3

**join** ticket\_flights tf **using** (flight\_id)

**join** airports a2 **on** a2.airport\_code = f3.arrival\_airport --используем аэропорт прибытия

**group** **by** tf.flight\_id, a2.city, tf.fare\_conditions

)

**select** flight\_conditions.flight\_id, flight\_conditions.city, min\_business.min\_business\_amount, max\_economy.max\_economy\_amount, (min\_business.min\_business\_amount - max\_economy.max\_economy\_amount) **as** delta

**from** flight\_conditions

**join** (

**select** flight\_conditions.flight\_id, flight\_conditions.min\_amount **as** min\_business\_amount --находим минимальную стоимость бизнес-класса в рамках конкретного перелета(flight\_id)

**from** flight\_conditions

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

**group** **by** flight\_conditions.flight\_id, flight\_conditions.min\_amount) min\_business **on** min\_business.flight\_id = flight\_conditions.flight\_id

**join** (

**select** flight\_conditions.flight\_id, flight\_conditions.max\_amount **as** max\_economy\_amount --находим максимальную стоимость эконом-класса в рамках конкретного перелета(flight\_id)

**from** flight\_conditions

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

**group** **by** flight\_conditions.flight\_id, flight\_conditions.max\_amount) max\_economy **on** max\_economy.flight\_id = flight\_conditions.flight\_id

**group** **by** flight\_conditions.flight\_id, flight\_conditions.city, min\_business.min\_business\_amount, max\_economy.max\_economy\_amount;