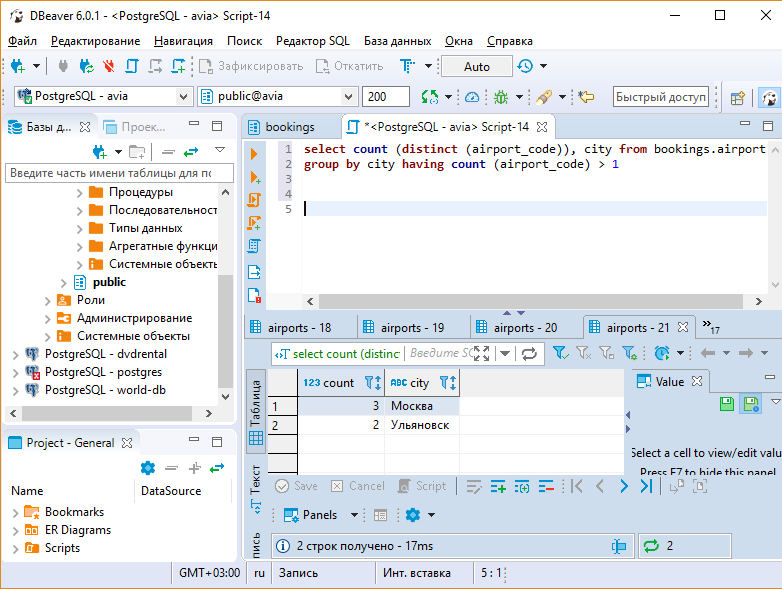
**1.** В каких городах больше одного аэропорта?

Исходя из описания базы данных Аэропорт следует, что Таблица 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) (Таблица bookings.airports ). В решении задачи используем полученную информацию и выведем счетчик уникальных трехбуквенных кодов аэропортов и название города. Проведем сортировку по городу, где счетчик кодов аэропортов имеет значение больше 1:

**select** **count** (**distinct** (airport\_code)), city **from** bookings.airports

**group** **by** city **having** **count** (airport\_code) > 1



2. Были ли брони, по которым не совершались перелеты?

Снова обратимся к описанию базы данных.

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. 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). Для решения задачи нам понадобятся таблицы bookings.bookings, bookings.boarding\_passes и bookings.tickets. Сначала я добавила таблицу bookings.tickets в bookings.boarding\_passes, используя right join, чтобы сохранить значения null. Потом в объединенную таблицу добавила bookings.bookings join’ом, потому что значения null в таблице bookings.bookings меня не интересуют. А потом вывела столбец boarding\_no из таблицы bookings.boarding\_passes, имеющих значение null. Хотелось бы сразу обратить внимание, что в условии задачи не сказано вывести брони по уже свершившимся рейсам, по которым не совершались перелеты, поэтому данное решение просто отвечает на вопрос — были ли брони, по которым не совершались перелеты. Ответ — да, были, причем данное решение содержит и будущие перелеты, то есть билет на рейс забронирован заранее, допустим за месяц, до момента фактического перелета осталось еще примерно 30 дней, поэтому регистрация на рейс еще не совершалась, она совершится, возможно, в будущем. В пункте 5.1 я попробовала решить задачу с добавлением условия, что рейс уже совершается или совершен.

**select** bp.boarding\_no, bp.flight\_id, bp.ticket\_no, b.book\_ref **from** bookings.boarding\_passes bp

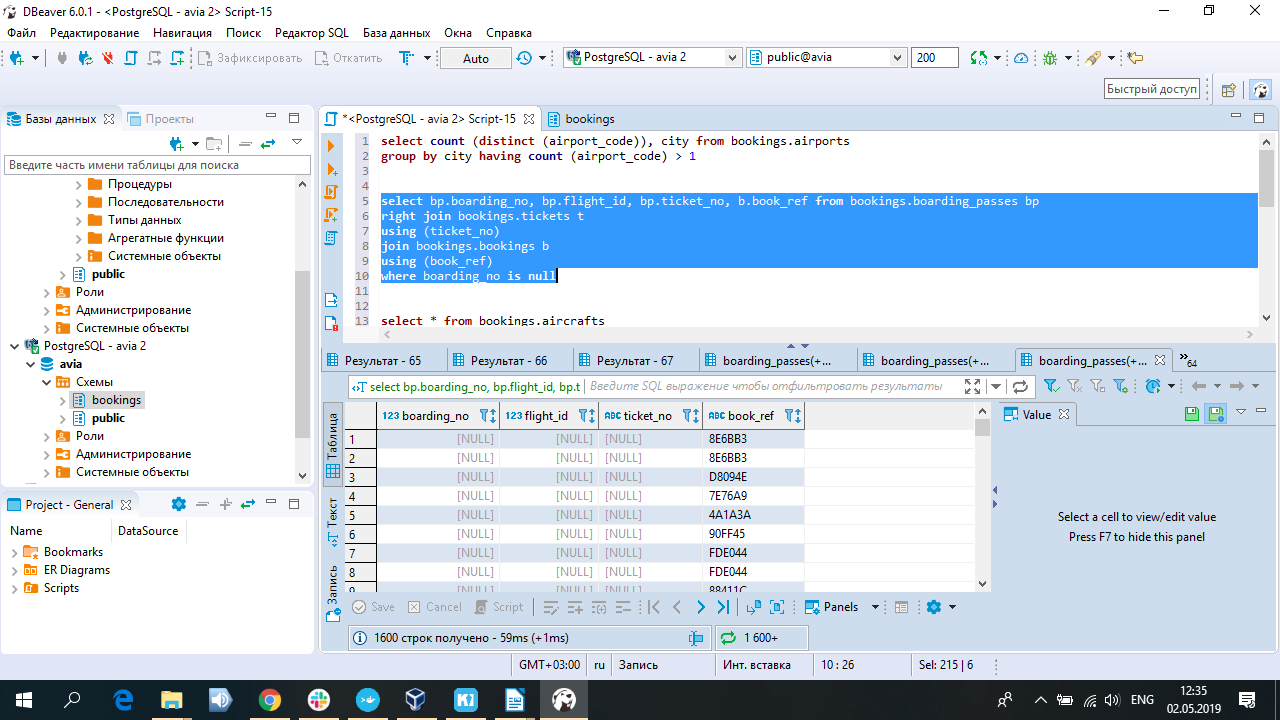
**right** **join** bookings.tickets t

**using** (ticket\_no)

**join** bookings.bookings b

**using** (book\_ref)

**where** boarding\_no **is** **null**



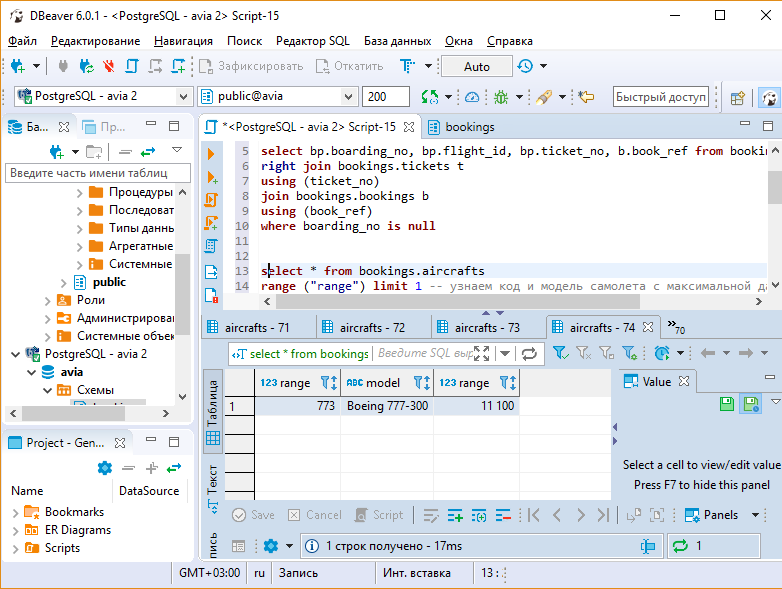
**3.** В каких аэропортах(!) есть рейсы, которые обслуживаются самолетами с максимальной дальностью перелетов?

Для решения данной задачи обратимся к описанию базы данных.

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). Для начала воспользуемся таблицей bookings.aircrafts, чтобы узнать код и модель самолета с максимальной дальностью полета.

**select** \* **from** bookings.aircrafts

**range** ("range") **limit** 1 **-- узнаем код и модель самолета с максимальной дальностью полета**



Получаем самолет Boeing 777-300 с максимальной дальностью полета 11100 километров.

Далее я объединила таблицы bookings.airports и bookings.aircrafts через bookings.routes, так как bookings.airports и bookings.aircrafts не связаны между собой. Можно было объединить через bookings.flights, но обратившись к описанию базы данных («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. Именно такая информация и составляет материализованное представление routes.») я решила использовать таблицу bookings.routes. Для объединения bookings.airports и bookings.routes использовались ключ airport\_code (таблица bookings.airports) и departure\_airport и arrival\_airport (таблица bookings.routes). Для добавления таблицы bookings.aircrafts использовался ключ aircraft\_code, для все таблиц он одинаков. После добавления всех таблиц я просто вывела те строки, где значение range (максимальная дальность полета) = 11100. Получен список аэропортов, с рейсами, которые обслуживаются самолетами с максимальной дальностью перелетов. Для того, чтобы вывести уникальный список аэропортов, был использован оператор distinct для столбца airport\_code

**select** **distinct** (a.airport\_code), a.airport\_name, ac.aircraft\_code, ac.model, ac."range"

**from** bookings.airports a

**join** bookings.routes r

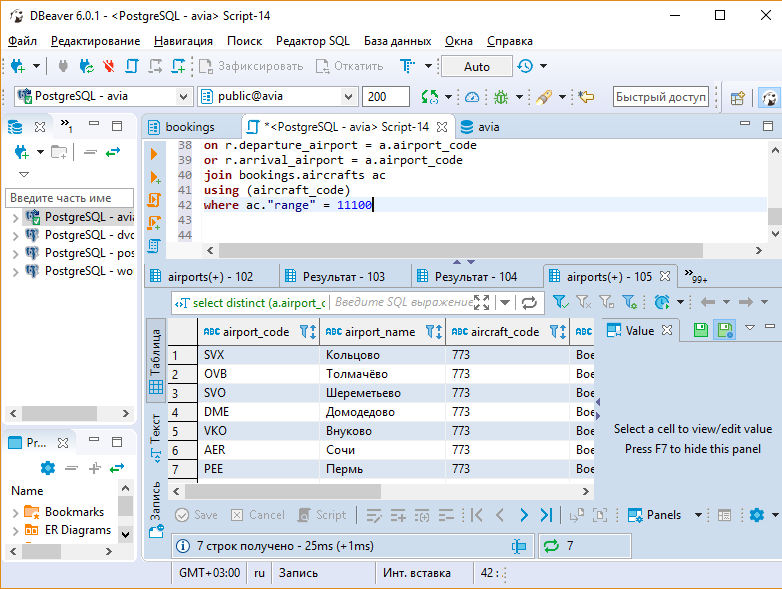
**on** r.departure\_airport = a.airport\_code

**or** r.arrival\_airport = a.airport\_code

**join** bookings.aircrafts ac

**using** (aircraft\_code)

**where ac."range" = 11100**

****

**5.** Написать не менее 4 SQL запросов которые позволят Вам проанализировать БД ( если Вы выполняли пункт 4 – запросы обязательно должны зайдействовать Ваши таблицы)

В запросах должны быть отражены как базовые команды, так и аналитические функции (не менее 3 разных запросов).

Должно присутствовать описание того, что вы получаете путем каждого запроса и почему используете именно такую структуру запроса.

5.1 Были ли брони, по которым не совершались перелеты, при условии, что рейс уже совершается или совершен? По логике задачи нам нужно найти те бронирования, где пассажир не получал посадочный талон, а рейс совершился в любом случае (независимо от того, сел конкретный пассажир в самолет или нет). Задача решается по аналогии с пунктом 2, но в данном случае нам нужно соединить таблицу bookings.bookings с таблицей bookings.flights. Последовательно соединяем таблицы bookings.bookings, bookings.tickets, bookings.boarding\_passes, bookings.ticket\_flights и bookings.flights. Выбрала правый join, чтобы в самую первую таблицу добавлялись полные сведения из последующих таблиц. В первую очередь нас интересует статус полета. Согласно описанию базы данных:

«Статус рейса (status) может принимать одно из следующих значений:

• Scheduled Рейс доступен для бронирования. Это происходит за месяц до плановой даты вылета; до этого запись о рейсе не существует в базе данных.

• 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.»

Соответственно выбираем значения столбца **status, содержащие значения ('Departed', 'Arrived').**

Благодаря right join я также получила большое количество значений book\_ref со значением null, необходимо отфильтровать запрос со значениями book\_ref is not null. Ну и то, что нас интересует — столбец boarding\_no со значением null. В результате мы получаем данные о тех бронированиях, где рейс уже совершился (или совершается), но пассажир на рейс не зарегистрировался. Запрос значений не выдал. Отсюда вывод: брони, по которым не совершались перелеты, при условии, что рейс уже совершается или совершен, отсутствуют.

**select b.book\_ref, t.ticket\_no, bp.boarding\_no, f.flight\_id, f.status**

**from** bookings.bookings b

**right** **join** bookings.tickets t

**using** (book\_ref)

**right** **join** bookings.boarding\_passes bp

**using** (ticket\_no)

**right** **join** bookings.ticket\_flights tf

**using** (ticket\_no)

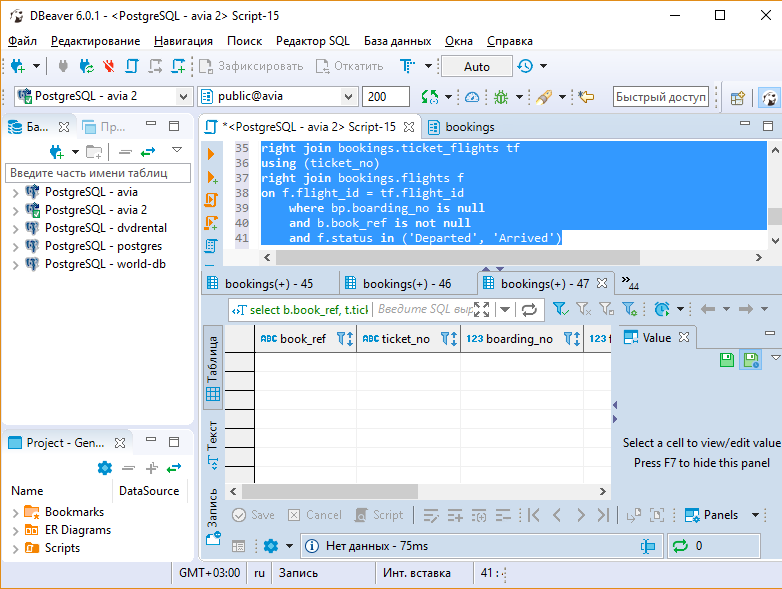
**right** **join** bookings.flights f

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

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

**and** b.book\_ref **is** **not** **null**

**and f.status in ('Departed', 'Arrived')**

****

5.2 Рассчитать вместимость пассажиров для каждого типа самолетов в парке, вывести результат, начиная с наибольшей пассажировместимости.

В данной задаче можно провести анализ парка самолетов. Подобный анализ необходим при осуществлении авиаперевозок. Наиболее востребованные маршруты имеет смысл обслуживать самолетами с наибольшей вместимостью и наоборот, наименее востребованные маршруты обслуживать самолетами меньшей вместимостью.

Для решения задачи я объединила две таблицы: bookings.seats и bookings.aircrafts join’ом (без левого, правого или полного, так как в данной задаче на мой взгляд лучше использовать именно пересечение значений, без null. С помощью оконной функции я посчитала количество мест для каждого типа самолета, результат вывела при помощи сортировки по убыванию. Оконную функцию использовала для удобства анализа. Оператор distinct для кодов типов самолетов использовался для того, чтобы избежать повторений строк.

**select distinct (aircraft\_code), a.model,**

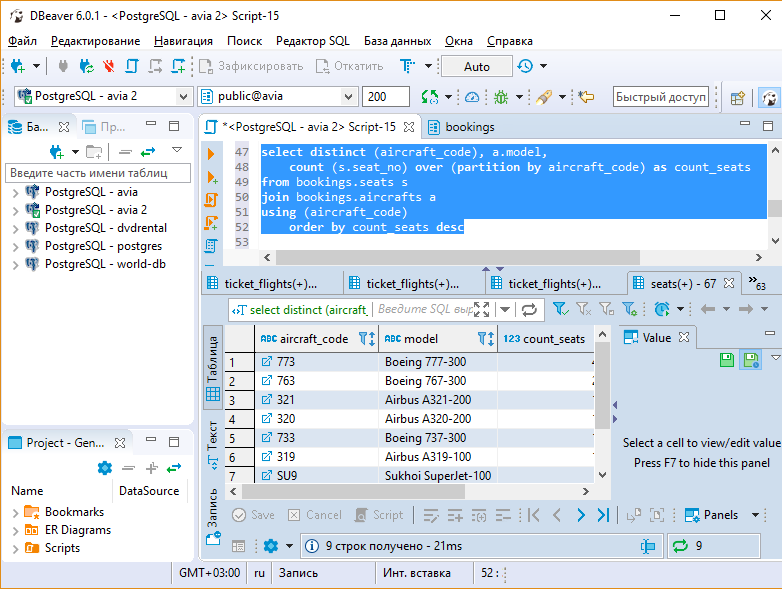
**count** (s.seat\_no) **over** (**partition** **by** aircraft\_code) **as** count\_seats

**from** bookings.seats s

**join** bookings.aircrafts a

**using** (aircraft\_code)

**order by count\_seats desc**



5.3 Вывести 10 рейсов (между какими городами, какой тип самолета обслуживал рейс) с самым большим количеством проданных билетов.

Решением данной задачи я продолжаю задачу 5.2. Выводим 10-ку самых прибыльных рейсов, на которые продано максимальное количество билетов (причем неважно, осуществлен рейс или еще нет). Возможно после анализа выяснится, что на данных рейсах имеет смысл использовать самолеты большей вместимости или осуществлять рейсы чаще. Для решения задачи необходимо объединить три таблицы: bookings.ticket\_flights, bookings.flights\_v и bookings.aircrafts. Вместо bookings.flights\_v можно было использовать bookings.flights, но разница, на мой взгляд, не принципиальная. Объединяла таблицы join’ом, так как значения null, если таковые имеются, в данном случае не интересуют. Оператор distinct для столбца flight\_id использовался, чтобы вывести только уникальные значения. Счетчиком count я посчитала количество проданных билетов и при помощи оконной функции и партиции сгруппировала его по столбцу flight\_id. Результат вывела в порядке убывания от наибольшего к меньшему с лимитом 10. В результате видим, что все в порядке, на наиболее востребованных рейсах летают самолеты с наибольшей вместимостью (из предыдущей задачи мы узнали, что Dboeing 777-300 имеет вместимость 402 места), при этом полной загрузки самолетов не наблюдается. Баланс между спросом и предложением соблюден.

**select** **distinct** (flight\_no),a.aircraft\_code||' '||a.model **as** aircraft\_model,

f.departure\_airport||' '||f.departure\_airport\_name **as** departure\_airport,

f.arrival\_airport||' '||f.arrival\_airport\_name **as** arrival\_airport,

f.status,

**count** (ticket\_no) **over** (**partition** **by** flight\_id) **as** count\_tickets

**from** bookings.ticket\_flights

**join** bookings.flights\_v f

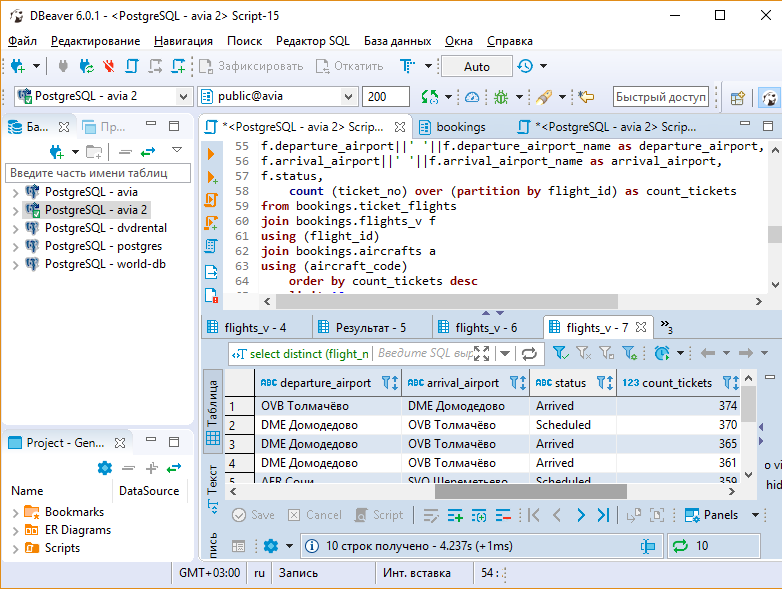
**using** (flight\_id)

**join** bookings.aircrafts a

**using** (aircraft\_code)

**order** **by** count\_tickets **desc**

**limit** 10



5.4 Вывести самые убыточные осуществленные (или осуществляемые на данный момент) рейсы, загрузка которых пассажирами меньше 10 человек (учитывать при решении задачи, что самолет CN1 Cessna 208 Caravan имеет максимальную вместимость 12 человек), вывести среднюю стоимость одного билета и общую сумму проданных билетов.

Продолжение задач 5.2 и 5.3. Возможно при анализе загруженности этих рейсов на основе решения задачи стоит отказаться от их осуществления. Для решения задачи необходимо объединить три таблицы: bookings.ticket\_flights, bookings.flights\_v и bookings.aircrafts. Объединяла таблицы join’ом, так как значения null, если таковые имеются, в данном случае не интересуют. Оператор distinct для столбца flight\_id использовался, чтобы вывести только уникальные значения. Счетчиком (ticket\_no) я посчитала количество проданных билетов, счетчиком sum (amount) общую сумму проданых билетов и счетчиком **avg** (amount) среднюю стоимость одного билета. Функцией **where** a.aircraft\_code **not** **in** 'CN1') **and** f.status **in** ('Departed', 'Arrived', 'Delayed') я отфильтровала значения CN1 ( Cessna 208 Caravan) и задала статус рейса — Выполняется, Прибыл и Рейс доступен для регистрации. Посчитала, что статус Рейс доступен для регистрации тоже можно отнести к убыткам, так как до выполнения рейса остаются всего лишь сутки, шанс в последний момент набрать большое количество пассажиров конечно есть, но он мал. Далее используется функция группировки Group by при условии, что количество проданных билетов **count** (ticket\_no) меньше 10.

**select** **distinct** (flight\_no), a.aircraft\_code||' '||a.model **as** aircraft\_model,

f.departure\_airport||' '||f.departure\_airport\_name **as** departure\_airport,

f.arrival\_airport||' '||f.arrival\_airport\_name **as** arrival\_airport,

f.status, **avg** (amount) **as** avg\_amount, **sum** (amount) **as** sum\_amount, **count** (ticket\_no) **as** count\_ticket

**from** bookings.ticket\_flights

**join** bookings.flights\_v f

**using** (flight\_id)

**join** bookings.aircrafts a

**using** (aircraft\_code)

**where** a.aircraft\_code **not** **in** ('CN1')

**and** f.status **in** ('Departed', 'Arrived', 'Delayed')

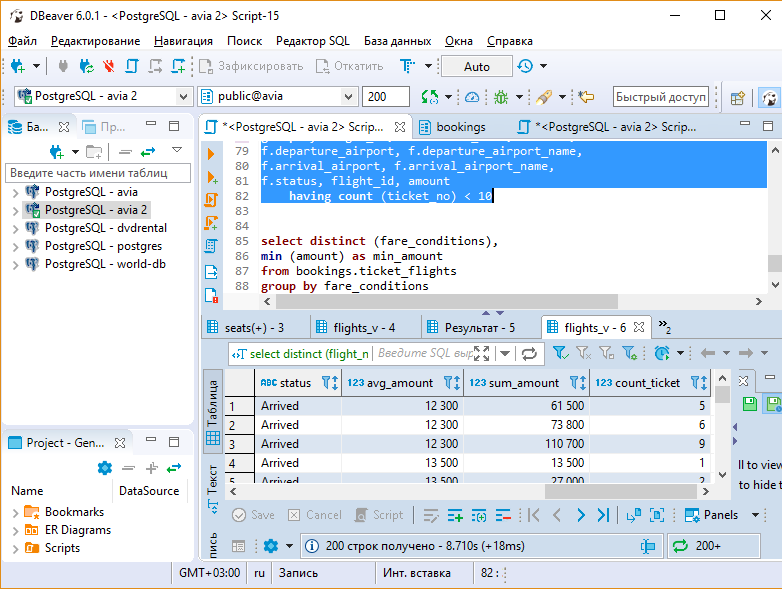
**group** **by** flight\_no, a.aircraft\_code, a.model,

f.departure\_airport, f.departure\_airport\_name,

f.arrival\_airport, f.arrival\_airport\_name,

f.status, flight\_id, amount

**having** **count** (ticket\_no) < 10



5.5 Допустим для рекламной акции нам необходимо в рекламном слогане: «Авиабилеты бизнесс класса от … рублей, комфорт класса от … рублей, эконом класса от … рублей» вставить пропущенные цифры. Цифры должны быть взяты не с потолка, а реальные.

Для решения задачи воспользуемся таблицей bookings.ticket\_flights и выберем минимальные значения для каждого типа fare\_conditions, используя функцию min и оператор distinct для того, чтобы убрать лишние значения.

**select** **distinct** (fare\_conditions),

**min** (amount) **as** min\_amount

**from** bookings.ticket\_flights

**group** **by** fare\_conditions

