| **№**  **задачи** | **Задание** | **Текст запроса** |
| --- | --- | --- |
|  | Используя SQL язык и произвольные две таблицы из модели данных необходимо объединить их различными способами (UNION, JOIN) | **UNION**  Объединим записи таблицы aircrafts и таблицы flights, выведем поле aircraft\_code. Дубликаты будут удалены, так как используется UNION.  **select** aircraft\_code **from** aircrafts  **union**  **select** aircraft\_code **from** flights    Объединим записи таблицы aircrafts и таблицы flights, выведем поле aircraft\_code. При этом из первой таблицы возьмем только самолеты, у которых максимальная дальность полета > 5000, а из второй таблицы только записи, у которых статус = Canceled. Дубликаты останутся, так как используется UNION ALL. **select** aircraft\_code **from** aircrafts  **where** **range** > 5000  **union** **all**  **select** aircraft\_code **from** flights  **where** status = 'Canceled'    **JOIN**  Объединим все данные по моделям самолетов и их завершенным полетам с точкой прибытия в один из аэропортов 'LED','KZN','CEK'. Отсортируем полученный результат по модели самолета в возрастающем порядке  **select**  a.aircraft\_code,  a.model,  f.flight\_no,  f.departure\_airport,  f.arrival\_airport,  f.status  **from**  aircrafts a **full** **join** flights f  **on** a.aircraft\_code = f.aircraft\_code  **where** f.status = 'Arrived' **and** f.arrival\_airport **in** ('LED','KZN','CEK')  **order** **by** a.model    Выведем данные по всем билетам в самолетах (номер, класс и стоимость) и дополним их данными о типе самолета и номерах мест  **select**  s.aircraft\_code,  t.fare\_conditions,  t.ticket\_no,  s.seat\_no,  t.amount  **from**  ticket\_flights t **left** **join** seats s  **on** s.fare\_conditions = t.fare\_conditions    Выведем данные по всем аэропортам (код, название и город), и дополним их информацией по полетам, для которых они являются аэропортами прибытия. Отсортируем результат по городу.  **select**  f.flight\_no,  f.aircraft\_code,  f.status,  a.airport\_code,  a.airport\_name,  a.city  **from**  flights f **right** **join** airports a  **on** f.arrival\_airport = a.airport\_code  **order** **by** a.city |
|  | Используя SQL язык напишите запрос с любым фильтром WHERE к произвольной таблице и результат отсортируйте (ORDER BY) с ограничением вывода по количеству строк (LIMIT) | Выведем информацию о полетах, которые были задержаны с сортировкой по аэропорту прибытия и ограничением в 11 записей  **Select** flight\_no, departure\_airport, arrival\_airport, aircraft\_code, scheduled\_departure  **From** flights f  **Where** f.status = 'Delayed'  **Order** **by** arrival\_airport  **Limit** 11 |
|  | Используя SQL язык, напишите OLAP запрос к произвольной связке таблиц (в рамках JOIN оператора), используя оператор GROUPBY и любые агрегатные функции count, min, max, sum. | Выведем название аэропорта и количество рейсов по каждому статусу внутри данного аэропорта, у которых данный аэропорт является точкой отправления  **select**  a.airport\_name **as** departure\_airport\_name,  f.status, **COUNT**(f.flight\_id) **as** count\_of\_flights  **from** airports a **join** flights f  **on** f.departure\_airport = a.airport\_code  **group** **by** a.airport\_name, f.status  **order** **by** a.airport\_name, f.status |
|  | Используя SQL язык, примените JOIN операторы (INNER, LEFT, RIGHT) для более чем двух таблиц из модели данных. | Для каждой модели самолета выведем данные о рейсах, обогащенные названиями городов вылета и прилета. Отсортируем полученный результат сначала по модели самолета, потом по номеру рейса, а затем по дате полета в порядке убывания.  **Select**  ac.model,  f.flight\_no,  f.scheduled\_departure::**Date** **as** flight\_date,  f.status,  ap1.city **as** departure\_city,  ap2.city **as** arrival\_city  **From** aircrafts ac  **Left** **join** flights f **on** ac.aircraft\_code = f.aircraft\_code  **Left** **join** airports ap1 **on** f.departure\_airport = ap1.airport\_code  **Left** **join** airports ap2 **on** f.arrival\_airport = ap2.airport\_code  **order** **by** model, flight\_no, flight\_date **desc** |
|  | Создайте виртуальную таблицу VIEW с произвольным именем для SQL запроса из задания 2 | Создадим представление на основе выборки из задания 2:  **create** **view** delayed\_flights\_v **as**  **Select** flight\_no, departure\_airport, arrival\_airport, aircraft\_code, scheduled\_departure  **From** flights f  **Where** f.status = 'Delayed'  Выполним запрос данных из созданного представления:  **select** \* **from** delayed\_flights\_v  **limit** 11    Создадим представление на основе выборки из задания 4:  **create** **view** model\_flights\_v **as**  **Select**  ac.model,  f.flight\_no,  f.scheduled\_departure::**Date** **as** flight\_date,  f.status,  ap1.city **as** departure\_city,  ap2.city **as** arrival\_city  **From** aircrafts ac  **Left** **join** flights f **on** ac.aircraft\_code = f.aircraft\_code  **Left** **join** airports ap1 **on** f.departure\_airport = ap1.airport\_code  **Left** **join** airports ap2 **on** f.arrival\_airport = ap2.airport\_code  Выполним запрос данных из созданного представления:  **select** \* **from** model\_flights\_v  **limit** 11 |
|  | Используя язык программирования Python, создайте Python-скрипт для вывода данных на экран (в консоль или IDE) из задания 5 | **Файл** (для выполнения нужны пакеты psycopg2 и tabulate)**:**  <https://github.com/Kemvzhik/HomeWork/blob/cea9c2639c64f9ba8889b6de22125e9e5e712841/Intermediate_certification_2_GainullinaEM.py>  **Текст скрипта:**  import psycopg2  from tabulate import tabulate  # Database configuration  db\_host= 'localhost'  db\_port= 5432  db\_name= 'demo'  db\_user= 'postgres'  db\_password= 'postgres'  if \_\_name\_\_ == '\_\_main\_\_':  try:  # Connect to the database  connection = psycopg2.connect(  host=db\_host,  port=db\_port,  database=db\_name,  user=db\_user,  password=db\_password,  options="-c search\_path=bookings"  )  # Create a cursor object to execute queries  cursor = connection.cursor()  # Execute a query  cursor.execute("SELECT \* FROM delayed\_flights\_v")  result = cursor.fetchall()  #print(result)  print(tabulate(result, headers=['flight\_no', 'departure\_airport', 'arrival\_airport', 'aircraft\_code', 'scheduled\_departure'], tablefmt='psql'))  # Close the cursor and connection  cursor.close()  connection.close()  except psycopg2.Error as e:  print("Error connecting to PostgreSQL:", e)  **Результат:** |