|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 3 |

**Название:** Запросы PostgreSQL

**Дисциплина:** Технология параллельных систем баз данных

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ6-12М |  |  | С.В. Астахов | |
|  | (Группа) |  | (Подпись, дата) | | (И.О. Фамилия) |
|  |  |  |  | |  |
| Преподаватель |  |  |  | |  |
|  |  |  | (Подпись, дата) | | (И.О. Фамилия) |

Москва, 2023

**Введение**

**1. Цель работы:** научиться разрабатывать сложные запросы SELECT.

**Ход выполнения**

**2. Создание БД.**

Создадим в файловой системе папку “lab3” и поместим туда файл с дампом БД. Создадим из дампа, подключимся к ней и просмотрим список схем, где обнаружим схему bookings (рисунок 1).

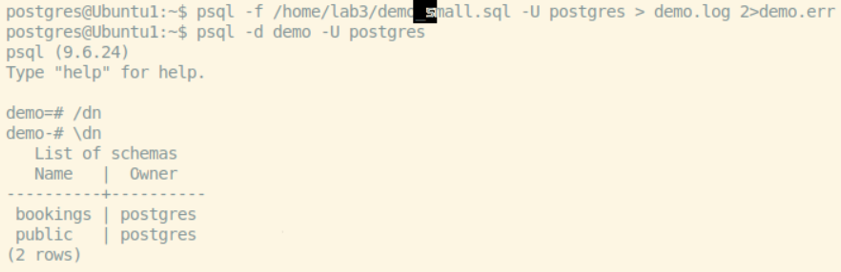


Рисунок 1 — создание БД из дампа

Затем, переключимся на схему bookings и посмотрим список таблиц в ней.

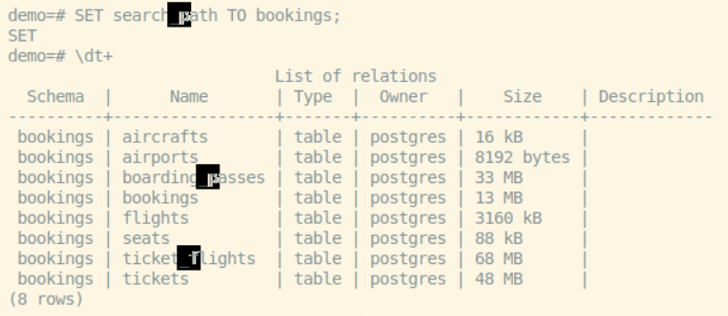


Рисунок 2 — список таблиц в БД

**3. Запросы**

Просмотрим содержимое таблиц самолетов и аэропортов (рисунок 3).

Примечание: здесь и далее многие запросы из методических указаний дополнены ограничением “LIMIT” для удобства просмотра.

Выполним запросы, позволяющий понять работу опраторов “LIKE” и “~”, сравнивающих строку с заданным шаблоном. В операторе LIKE: “%” — любое число любых символов, “\_” — строго один символ. В операторе “~”: “^” — начало строки, “$” — конец строки, “|” — дизъюнкция.

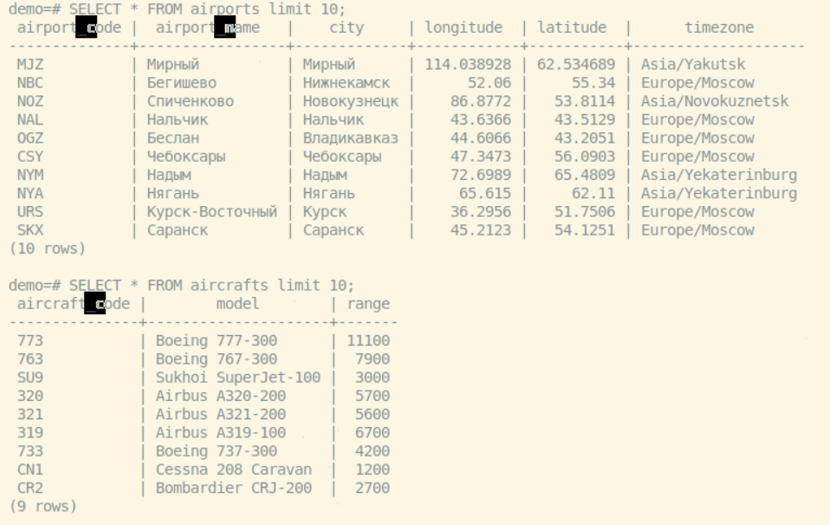


Рисунок 3 — содержимое таблиц аэропортов и самолетов

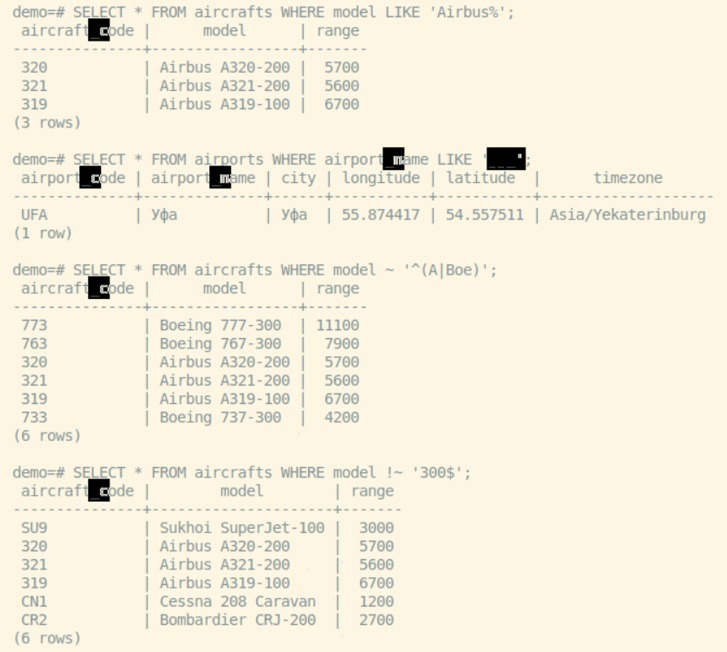


Рисунок 4 — запросы с использованием строковых шаблонов

Ознакомимся с оператором “BETWEEN” и получением вычисляемых колонок в результате Select-запроса (рисунок 5).

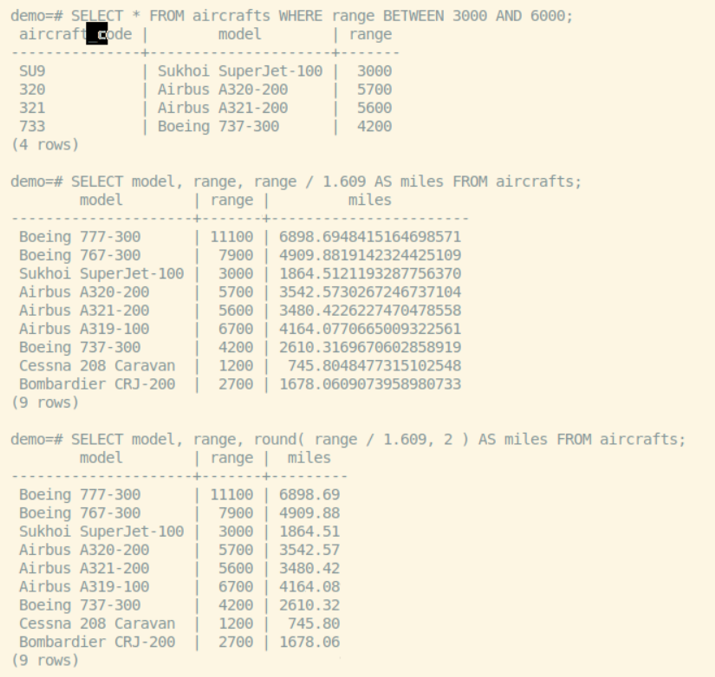


Рисунок 5 — получение вычисляемых колонок в select-запросе

Ознакомимся с оператором упорядочивания результатов запроса “ORDER BY” (рисунок 6).

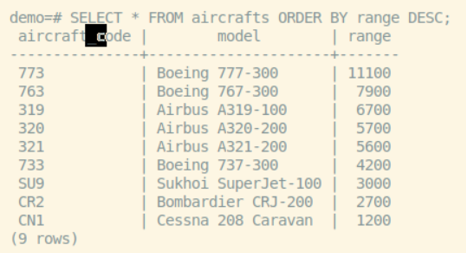


Рисунок 6 — упорядочивание результатов запроса

Ознакомимся с оператором “DISTINCT”, позволяющим выбрать уникальные значения (рисунки 6-7).

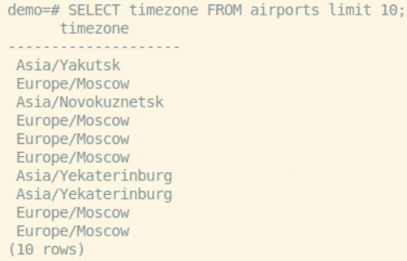


Рисунок 6 — запрос с сохранением дублирующихся строк

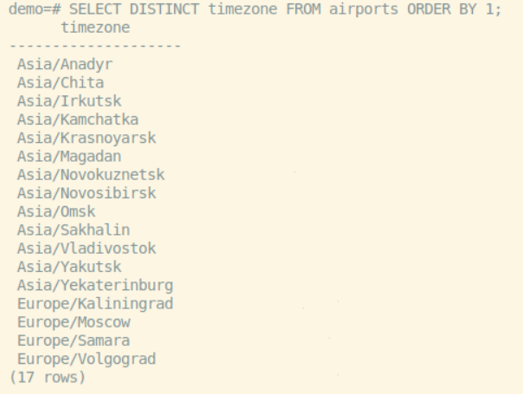


Рисунок 7 — запрос с выборкой уникальных строк

Ознакомимся с операторами “LIMIT” и “OFFSET”, выбирая 3 самых восточных и 3 следующих аэропорта (рисунок 8).

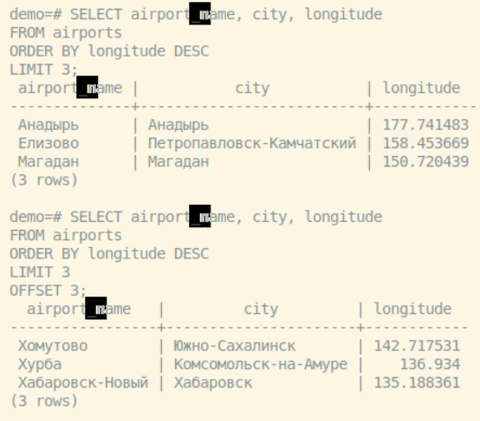


Рисунок 8 — использование limit и offset

Ознакомимся в оператором “CASE”, классифицурая самолеты по дальности полета (рисунок 9).

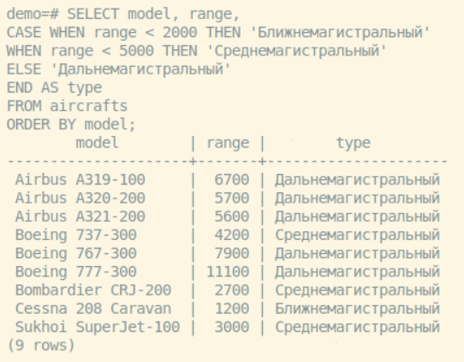


Рисунок 9 — классификация самолетов по дальности полета

Ознакомимся с оператором соединения “JOIN”, запросив список сидений для заданной модели самолета (рисунок 10).

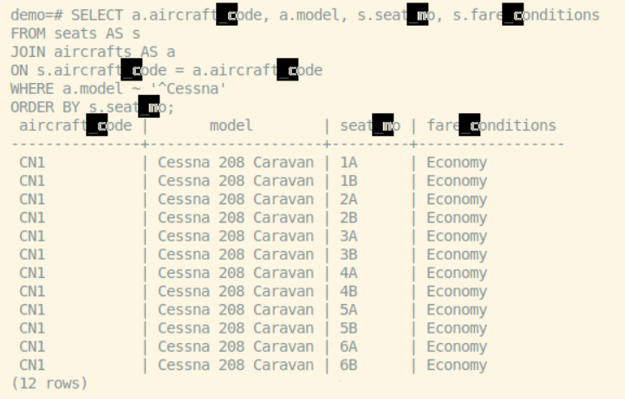


Рисунок 10 — использование соединений

Рассмотрим внешние соединения, просмотрев какие самолеты на скольки маршрутах используются (рисунок 11). Внешнее соединение позволит увидеть неиспользуемый самолет.

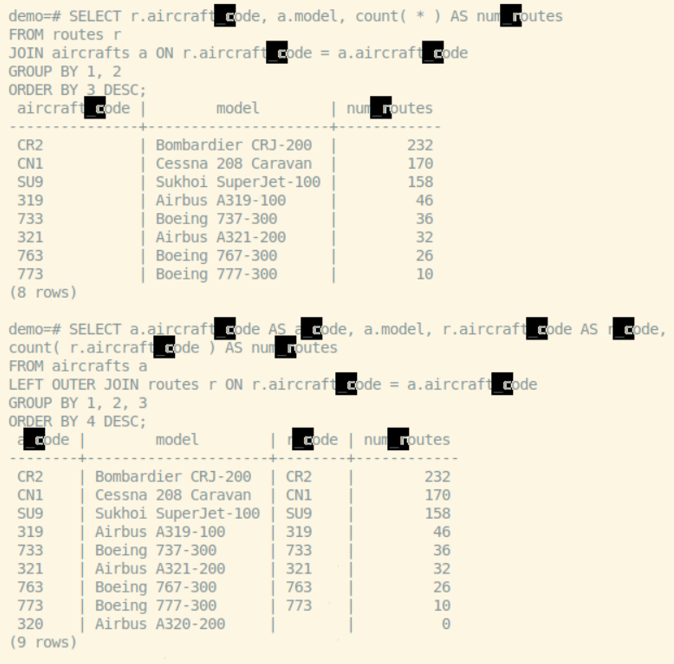


Рисунок 11 — запрос без и с использованием внешнего соединения

Рассмотрим многотабличный запрос на примере запроса, выводящего список пассажиров, не прошедших регистрацию на рейс (рисунок 12).

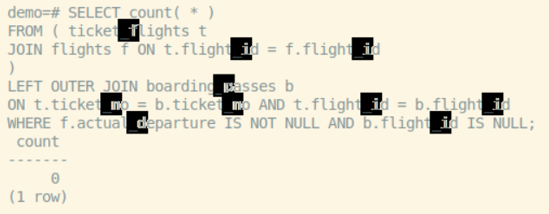


Рисунок 12 — список пассажиров, не прошедших регистрацию на рейс

Ознакомимся с оператором “UNION”, запросив в какие города можно полететь из Москвы или Санкт-Петербурга (рисунки 13-14).

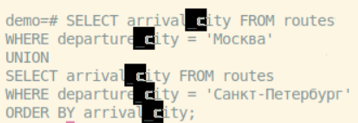


Рисунок 14 — запрос с оператором UNION

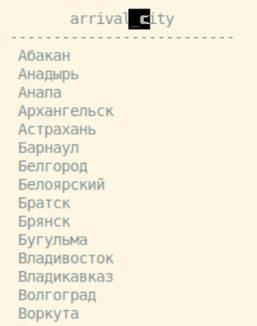


Рисунок 15 — результат выполнения запроса

Ознакомимся с агрегатными функциями, запросив информацию о средней стоимости билеов, а так же подсчитаем количество рейсов из Москвы в другие города (рисунки 16-17).

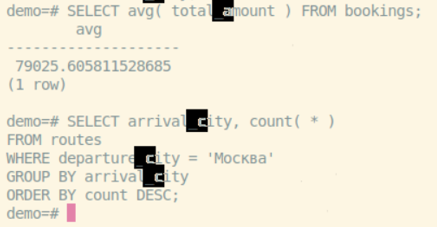


Рисунок 16 — средняя стоимость билетов и запрос количества рейсов из Москвы



Рисунок 17 — количество рейсов из Москвы

Так же, запросим обобщенную информацию о частоте рейсов; названия городов, из которых в другие города существует не менее 15 маршрутов (рисунок 18).

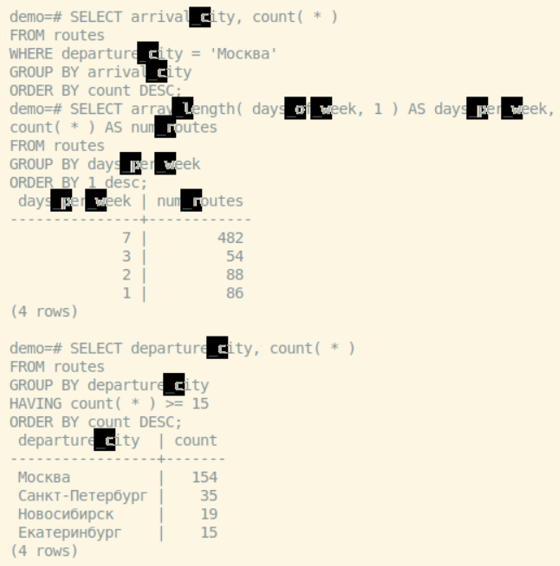


Рисунок 18 — информация о частоте рейсов и количестве маршрутов

Ознакомимся с оконными функциями, запросив накопительное число проданных билетов по месяцам (рисунки 19-20).

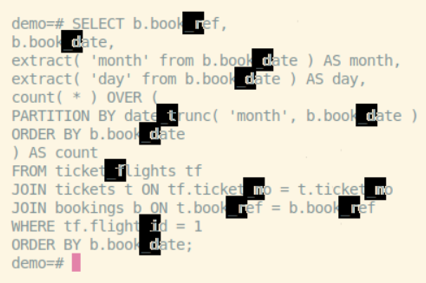


Рисунок 19 — запрос числа проданных билетов по месяцам

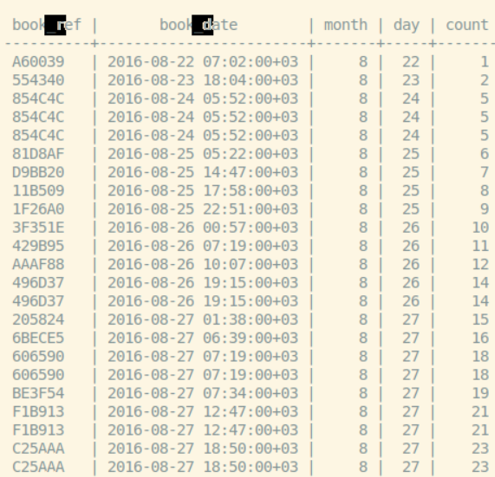


Рисунок 20 — число проданных билетов по месяцам

Рассмотрим простейший подзапрос, подсчитав суммы бронирования, которые больше средней (рисунок 21).

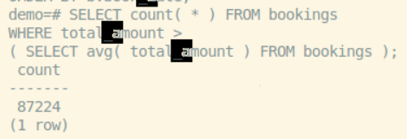


Рисунок 21 — простейший подзапрос

Рассмотрим так же подзапрос, для обнаружения соединенных маршрутом городов в часовом поясе Красноярска (рисунок 22) и обнаружения городов, куда нет маршрутов из Москвы (рисунок 23).

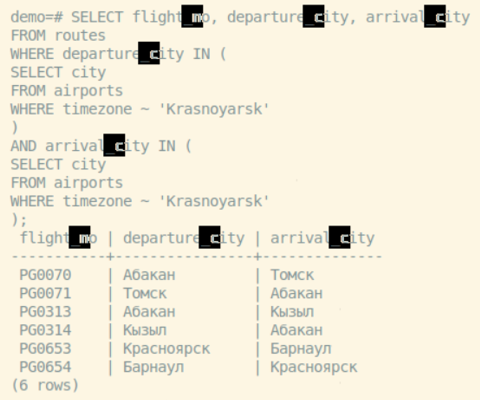


Рисунок 22 — города маршрутов в часовом поясе Красноярска

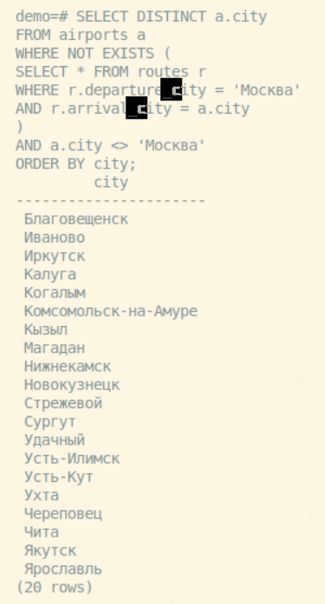


Рисунок 23 — города, не соединенные маршрутом с Москвой

Кроме того, выведем список аэропортов из городов с несколькими аэропортами (рисунок 24) и определим число маршрутов, исходящих из тех аэропортов, которые расположены восточнее географической долготы 150◦ (рисунок 25).

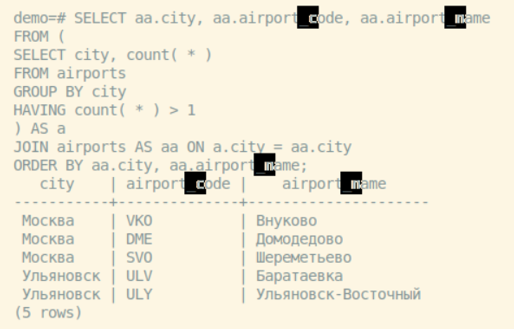


Рисунок 24 — список аэропортов из городов с несколькими аэропортами

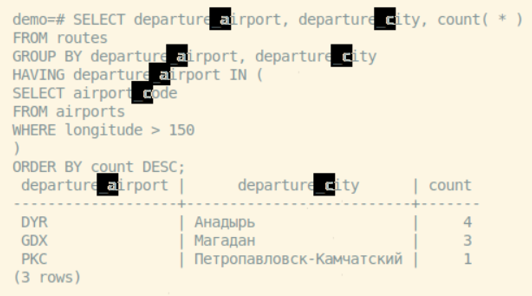


Рисунок 25 — число маршрутов из аэропортов, которые расположены восточнее географической долготы 150◦

Используем вложенные подзапросы, чтобы определить степень заполнения самолетов (листинг 1, рисунок 26).

Листинг 1 — запрос для вычисления степени заполнения самолетов

|  |
| --- |
| SELECT ts.flight\_id,  ts.flight\_no,  ts.scheduled\_departure\_local,  ts.departure\_city,  ts.arrival\_city,  a.model,  ts.fact\_passengers,  ts.total\_seats,  round( ts.fact\_passengers::numeric /  ts.total\_seats::numeric, 2 ) AS fraction  FROM (  SELECT f.flight\_id,  f.flight\_no,  f.scheduled\_departure\_local,  f.departure\_city,  f.arrival\_city,  f.aircraft\_code,  count( tf.ticket\_no ) AS fact\_passengers,  ( SELECT count( s.seat\_no )  FROM seats s  WHERE s.aircraft\_code = f.aircraft\_code  ) AS total\_seats  FROM flights\_v f  JOIN ticket\_flights tf ON f.flight\_id = tf.flight\_id  WHERE f.status = 'Arrived'  GROUP BY 1, 2, 3, 4, 5, 6  ) AS ts  JOIN aircrafts AS a ON ts.aircraft\_code = a.aircraft\_code  ORDER BY ts.scheduled\_departure\_local; |

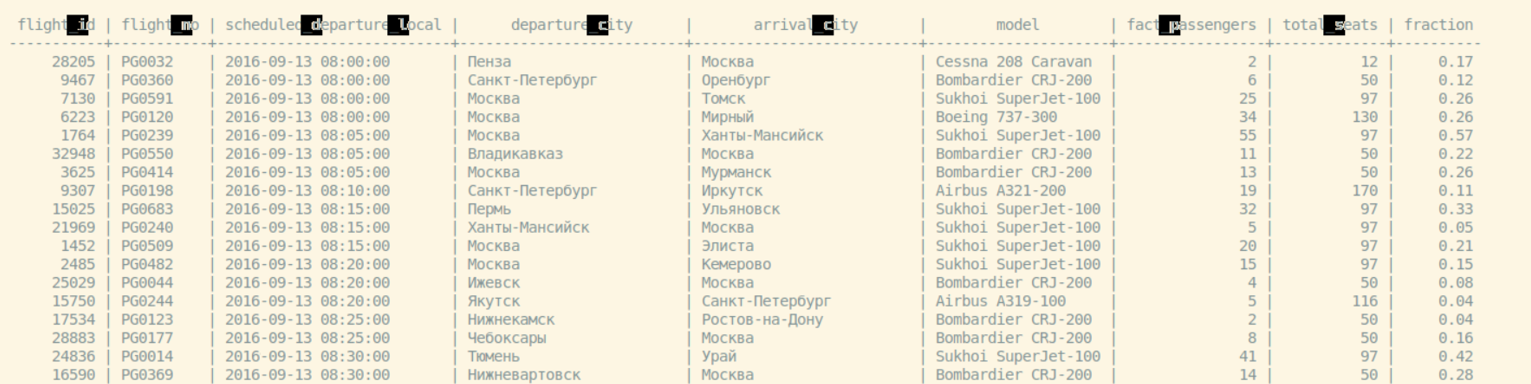


Рисунок 26 — степень заполнения самолетов

Рассмотренный сложный запрос можно сделать более наглядным за счет выделения подзапроса в отдельную конструкцию, которая называется общее табличное выражение (Common Table Expression — CTE).

Листинг 2 — запрос для вычисления степени заполнения самолетов с использованием Common Table Expression

|  |
| --- |
| WITH ts AS  ( SELECT f.flight\_id,  f.flight\_no,  f.scheduled\_departure\_local,  f.departure\_city,  f.arrival\_city,  f.aircraft\_code,  count( tf.ticket\_no ) AS fact\_passengers,  ( SELECT count( s.seat\_no )  FROM seats s  WHERE s.aircraft\_code = f.aircraft\_code  ) AS total\_seats  FROM flights\_v f  JOIN ticket\_flights tf ON f.flight\_id = tf.flight\_id  WHERE f.status = 'Arrived'  GROUP BY 1, 2, 3, 4, 5, 6  )  SELECT ts.flight\_id,  ts.flight\_no,  ts.scheduled\_departure\_local,  ts.departure\_city,  ts.arrival\_city,  a.model,  ts.fact\_passengers,  ts.total\_seats,  round( ts.fact\_passengers::numeric /  ts.total\_seats::numeric, 2 ) AS fraction  FROM ts  JOIN aircrafts AS a ON ts.aircraft\_code = a.aircraft\_code  ORDER BY ts.scheduled\_departure\_local; |

**4. Контрольные вопросы и задания**

1. В документации сказано, что служебный символ «%» в шаблоне оператора LIKE соответствует любой последовательности символов, в том числе и пустой последовательности, однако ничего не сказано насчет правил обработки пробелов. В таблице «Билеты» (tickets) столбец passenger\_name содержит имя и фамилию пассажира, записанные заглавными латинскими буквами и разделенные одним пробелом. Выясните правила обработки пробелов самостоятельно, выполнив следующие команды и сравнив полученные результаты:

*SELECT count( \* ) FROM tickets;*

*SELECT count( \* ) FROM tickets WHERE passenger\_name LIKE '% %';*

*SELECT count( \* ) FROM tickets WHERE passenger\_name LIKE '% % %';*

*SELECT count( \* ) FROM tickets WHERE passenger\_name LIKE '% %%';*

Результаты выполнения запросов представлены на рисунке 27.

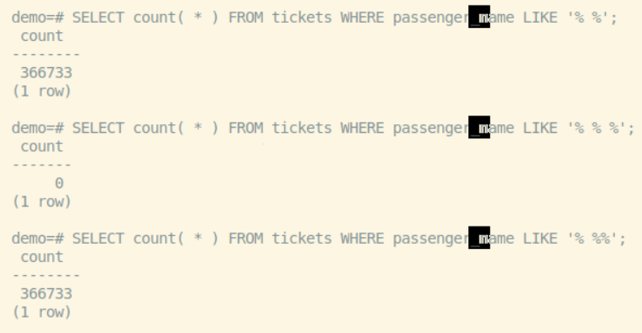


Рисунок 27 — исследование правил обработки пробелов

Из полученных результатов можно сделать вывод, что пробел в операторе LIKE обрабатывается аналогично обыкновенному буквенному или цифровому символу.

2. Этот запрос выбирает из таблицы «Билеты» (tickets) всех пассажиров с именами, состоящими из трех букв (в шаблоне присутствуют три символа «\_»):

*SELECT passenger\_name*

*FROM tickets*

*WHERE passenger\_name LIKE '\_\_\_ %';*

Предложите шаблон поиска в операторе LIKE для выбора из этой таблицы всех пассажиров с фамилиями, состоящими из пяти букв.

Аналогично представленному шаблону, для выполнения задания необходимо заменить имя символом “%” и разделить его пробелом с пятью “\_”, обозначающими символы фамилии (рисунок 28).

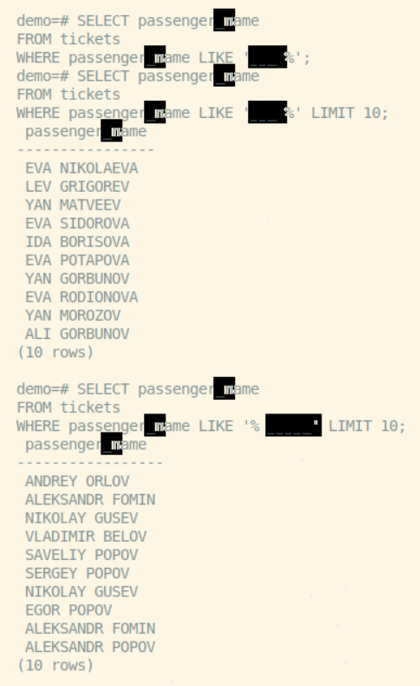


Рисунок 28 — фильтрация пассажиров по длине имени и по длине фамилии

4. Выясните, на каких маршрутах используются самолеты компании Boeing. В выборке вместо кода модели должно выводиться ее наименование, например, вместо кода 733 должно быть Boeing 737-300.

Указание: можно воспользоваться соединением представления «Маршруты» (routes) и таблицы «Самолеты» (aircrafts).

Запрос, возвращающий требуемый результат представлен в листинге 3, результат — на рисунке 29.

Листинг 3 — запрос маршрутов самолетов Boeing

|  |
| --- |
| SELECT flight\_no,  departure\_airport,  departure\_airport\_name,  departure\_city,  arrival\_airport,  arrival\_airport\_name,  arrival\_city,  model  FROM routes as r  JOIN aircrafts as a  ON a.aircraft\_code = r.aircraft\_code  WHERE model LIKE '%Boeing%'; |

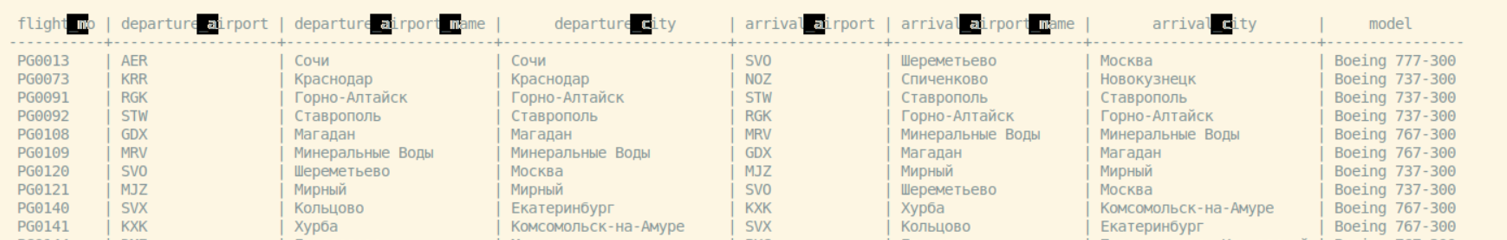


Рисунок 29 — результат выполнения запроса

4. Самые крупные самолеты в нашей авиакомпании — это Boeing 777-300. Выяснить, между какими парами городов они летают. Каждая пара городов была выведена только один раз.

Запрос, возвращающий требуемый результат представлен в листинге 4, результат — на рисунке 30.

Листинг 4 — города на маршрутах Boeing 777

|  |
| --- |
| SELECT DISTINCT r.departure\_city, r.arrival\_city  FROM routes AS r, aircrafts AS a  WHERE a.aircraft\_code = r.aircraft\_code AND a.model = 'Boeing 777-300' AND  r.arrival\_city > r.departure\_city; |

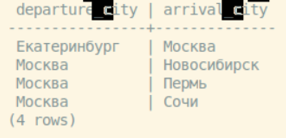


Рисунок 30 — результат выполнения запроса

5. Выяснить, сколько различных рейсов выполняется из каждого города, без учета частоты рейсов в неделю, можно с помощью обращения к представлению routes (маршруты).

Запрос, возвращающий требуемый результат представлен в листинге 5, результат — на рисунке 31.

Листинг 5 — число рейсов из различных городов

|  |
| --- |
| SELECT departure\_city, count( \* )  FROM routes  GROUP BY departure\_city  ORDER BY count DESC; |

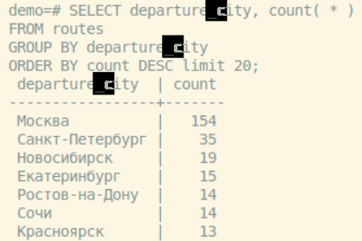


Рисунок 31 — результат выполнения запроса

6. Модифицируйте этот запрос так, чтобы он выводил число направлений, по которым летают самолеты из каждого города. Например, из Москвы в СанктПетербург летает несколько различных рейсов, но все эти рейсы относятся к одному направлению.

Запрос, возвращающий требуемый результат представлен в листинге 6, результат — на рисунке 32.

Листинг 6 — число направлений рейсов из различных городов

|  |
| --- |
| SELECT departure\_city, count(distinct arrival\_city)  FROM routes  GROUP BY departure\_city  ORDER BY count DESC; |

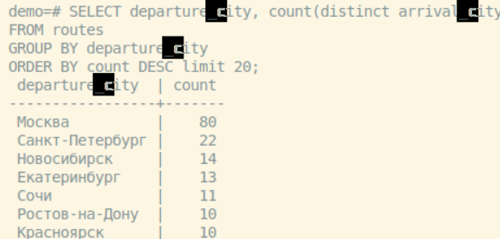


Рисунок 32 — результат выполнения запроса

7. В материализованном представлении «Маршруты» (routes) имеется столбец days\_of\_week, который содержит списки (массивы) номеров дней недели, когда выполняется каждый рейс. Для оптимизации расписания вылетов из Москвы нужно выявить пять городов, в которые из столицы отправляется наибольшее число ежедневных рейсов (маршрутов). Строки в выборке следует расположить в убывающем порядке числа выполняемых рейсов.

Запрос, возвращающий требуемый результат представлен в листинге 7, результат — на рисунке 33.

Листинг 7 — пять наиболее связанных с Москвой городов

|  |
| --- |
| SELECT arrival\_city, count(\*) AS count FROM routes WHERE  array\_length(days\_of\_week, 1) = 7 AND departure\_city = 'Москва' GROUP BY  arrival\_city ORDER BY count DESC; |

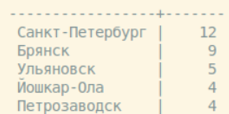


Рисунок 33 — результат выполнения запроса

8. Предположим, что служба материального снабжения нашей авиакомпании запросила информацию о числе рейсов, выполняющихся из Москвы в каждый день недели.

Соответствующий запрос и результат его выполнения представлены на рисунке 34.

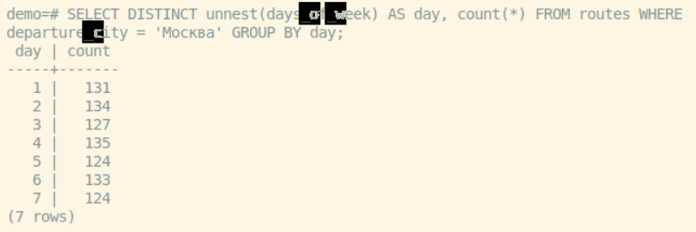


Рисунок 34 — число рейсов из Москвы по дням

13. Каковы максимальные и минимальные цены билетов на все направления. Оператор SELECT должен возвращать departure\_city, arrival\_city, max(amount ), min(amount ).

Запрос, возвращающий требуемый результат представлен в листинге 8, результат — на рисунке 35.

Листинг 8 — максимальные и минимальные цены билетов на все направления

|  |
| --- |
| SELECT departure\_city, arrival\_city  FROM flights\_v as f  JOIN ticket\_flights as t  ON f.flight\_id = t.flight\_id;  SELECT departure\_city, arrival\_city, max(amount), min(amount)  FROM flights\_v as f  JOIN ticket\_flights as t  ON f.flight\_id = t.flight\_id  GROUP BY departure\_city, arrival\_city; |

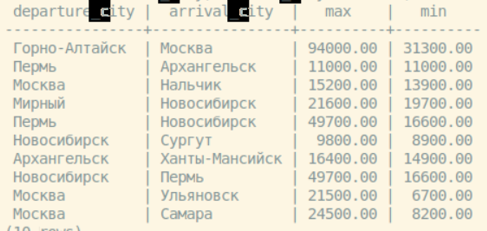


Рисунок 35 — результат выполнения запроса

**Вывод:** По итогам выполнения данной лабораторной работы были изучены различные подходы к разработке сложных запросов SELECT. Были рассмотрены разнообразные примеры запросов с широким спектром операторов, предикатов и функций. После этого были проработаны практические задачи, связанные с разработкой запросов.