



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

О Т Ч Е Т

по лабораторной работе № 3

Название: Запросы PostgreSQL

Дисциплина: Технология параллельных систем баз данных

Студент

ИУ6-12М

(Группа)

(Подпись, дата)

Д.С. Каткова

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

А.Д. Пономарев

(И.О. Фамилия)

Москва, 2023

1 Цель лабораторной работы

Целью лабораторной работы является формирование следующей компетенции: студент должен научиться разрабатывать сложные запросы SELECT.

2 Подключение к виртуальной машине и создание БД

После подключения к виртуальной машине и создания БД можно посмотреть список всех схем и список всех таблиц.

```
demo=# \dn
        Список схем
  Имя      | Владелец
-----+-----
 bookings | postgres
 public   | postgres
(2 строки)
```

Список отношений					
Схема	Имя	Тип	Владелец	Размер	Описание
bookings	aircrafts	таблица	postgres	16 kB	Самолеты
bookings	airports	таблица	postgres	48 kB	Аэропорты
bookings	boarding_passes	таблица	postgres	33 MB	Посадочные талоны
bookings	bookings	таблица	postgres	13 MB	Бронирования
bookings	flights	таблица	postgres	3160 kB	Рейсы
bookings	seats	таблица	postgres	88 kB	Места
bookings	ticket_flights	таблица	postgres	68 MB	Перелеты
bookings	tickets	таблица	postgres	48 MB	Билеты

(8 строк)

3 Запросы

Прежде чем перейти к конкретным запросам, необходимо ознакомиться с содержимым двух таблиц – «Самолеты» и «Аэропорты». Таблица «Самолеты» совсем маленькая, а таблица «Аэропорты» содержит чуть больше ста строк.

*SELECT * FROM aircrafts;*

```
demo=# SELECT * FROM aircrafts;
 aircraft_code |      model      | range
-----+-----+-----
       773     | Boeing 777-300  | 11100
       763     | Boeing 767-300  |  7900
       SU9     | Sukhoi SuperJet-100 |  3000
       320     | Airbus A320-200  |  5700
       321     | Airbus A321-200  |  5600
       319     | Airbus A319-100  |  6700
       733     | Boeing 737-300   |  4200
       CN1     | Cessna 208 Caravan |  1200
       CR2     | Bombardier CRJ-200 |  2700
(9 строк)
```

*SELECT * FROM airports;*

airport_code	airport_name	city	longitude	latitude	timezone
MJZ	Мирный	Мирный	114.038928	62.534689	Asia/Yakutsk
NBC	Бегишево	Нижекамск	52.06	55.34	Europe/Moscow
NOZ	Спиченково	Новокузнецк	86.8772	53.8114	Asia/Novokuznetsk
NAL	Нальчик	Нальчик	43.6366	43.5129	Europe/Moscow
OGZ	Беслан	Владикавказ	44.6066	43.2051	Europe/Moscow
CSY	Чебоксары	Чебоксары	47.3473	56.0903	Europe/Moscow
NYM	Надым	Надым	72.6989	65.4809	Asia/Yekaterinburg
NYA	Нягань	Нягань	65.615	62.11	Asia/Yekaterinburg
URS	Курск-Восточный	Курск	36.2956	51.7506	Europe/Moscow
SKX	Саранск	Саранск	45.2123	54.1251	Europe/Moscow
TBW	Донское	Тамбов	41.48	52.81	Europe/Moscow
OVS	Советский	Советский	63.6044	61.32	Asia/Yekaterinburg
IJK	Ижевск	Ижевск	53.4575	56.8281	Europe/Samara
SLY	Салехард	Салехард	66.61042	66.590753	Asia/Yekaterinburg
HMA	Ханты-Мансийск	Ханты-Мансийск	69.086067	61.028479	Asia/Yekaterinburg
RGK	Горно-Алтайск	Горно-Алтайск	85.833333	51.966667	Asia/Krasnoyarsk
UKX	Усть-Кут	Усть-Кут	105.7167	56.85	Asia/Irkutsk
GDZ	Геленджик	Геленджик	38.016666666667	44.566666666667	Europe/Moscow
NFG	Нефтеюганск	Нефтеюганск	72.65	61.1083	Asia/Yekaterinburg

Начнем с различных условий отбора строк в предложении WHERE. Для начала поставим перед собой такую задачу: выбрать все самолеты компании Airbus. В этом нам поможет оператор поиска шаблонов LIKE.

*SELECT * FROM aircrafts WHERE model LIKE 'Airbus%';*

demo=# SELECT * FROM aircrafts WHERE model LIKE 'Airbus%';			
aircraft_code	model	range	
-----+-----+-----			
320	Airbus A320-200	5700	
321	Airbus A321-200	5600	
319	Airbus A319-100	6700	
(3 строки)			

Вместо символа «%» могут быть подставлены любые символы в любом количестве, а может и не быть подставлено ни одного символа.

Кроме символа «%» в шаблоне может использоваться и символ подчеркивания _ «_», который соответствует в точности одному любому символу. В качестве примера найдем в таблице «Аэропорты» те из них, которые имеют названия длиной три символа (буквы). С этой целью зададим в качестве шаблона строку, состоящую из трех символов «_».

*SELECT * FROM airports WHERE airport_name LIKE '___';*

demo=# SELECT * FROM airports WHERE airport_name LIKE '___';					
airport_code	airport_name	city	longitude	latitude	timezone
-----+-----+-----+-----+-----					
UFA	уфа	уфа	55.874417	54.557511	Asia/Yekaterinburg
(1 строка)					

Существует ряд операторов для работы с регулярными выражениями POSIX. Эти операторы имеют больше возможностей, чем оператор LIKE.

Для того чтобы выбрать, например, самолеты компаний Airbus и Boeing, можно сделать так:

*SELECT * FROM aircrafts WHERE model ~ '^(A|Boe)';*

```
demo=# SELECT * FROM aircrafts WHERE model ~ '^(A|Boe)';
```

aircraft_code	model	range
773	Boeing 777-300	11100
763	Boeing 767-300	7900
320	Airbus A320-200	5700
321	Airbus A321-200	5600
319	Airbus A319-100	6700
733	Boeing 737-300	4200

(6 строк)

Для инвертирования смысла оператора ~ нужно перед ним добавить знак «!». В качестве примера отыщем модели самолетов, которые не завершаются числом 300.

*SELECT * FROM aircrafts WHERE model !~ '300\$';*

```
demo=# SELECT * FROM aircrafts WHERE model !~ '300$';
```

aircraft_code	model	range
SU9	Sukhoi SuperJet-100	3000
320	Airbus A320-200	5700
321	Airbus A321-200	5600
319	Airbus A319-100	6700
CN1	Cessna 208 Caravan	1200
CR2	Bombardier CRJ-200	2700

(6 строк)

В качестве замены традиционных операторов сравнения могут использоваться предикаты сравнения, которые ведут себя так же, как и операторы, но имеют другой синтаксис. Давайте ответим на вопрос: какие самолеты имеют дальность полета в диапазоне от 3 000 км до 6 000 км? Ответ получим с помощью предиката BETWEEN.

*SELECT * FROM aircrafts WHERE range BETWEEN 3000 AND 6000;*

aircraft_code	model	range
SU9	Sukhoi SuperJet-100	3000
320	Airbus A320-200	5700
321	Airbus A321-200	5600
733	Boeing 737-300	4200

(4 строки)

При выборке данных можно проводить вычисления и получать в результирующей таблице вычисляемые столбцы. Если мы захотим представить дальность полета не только в километрах, но и в милях, то нужно вычислить это выражение и для удобства присвоить новому столбцу псевдоним с помощью ключевого слова AS.

SELECT model, range, range / 1.609 AS miles FROM aircrafts;

model	range	miles
Boeing 777-300	11100	6898.6948415164698571
Boeing 767-300	7900	4909.8819142324425109
Sukhoi SuperJet-100	3000	1864.5121193287756370
Airbus A320-200	5700	3542.5730267246737104
Airbus A321-200	5600	3480.4226227470478558
Airbus A319-100	6700	4164.0770665009322561
Boeing 737-300	4200	2610.3169670602858919
Cessna 208 Caravan	1200	745.8048477315102548
Bombardier CRJ-200	2700	1678.0609073958980733

(9 строк)

По всей вероятности, такая высокая точность представления значений в милях не требуется, поэтому мы можем уменьшить ее до разумного предела в два десятичных знака:

SELECT model, range, round(range / 1.609, 2) AS miles FROM aircrafts;

model	range	miles
Boeing 777-300	11100	6898.69
Boeing 767-300	7900	4909.88
Sukhoi SuperJet-100	3000	1864.51
Airbus A320-200	5700	3542.57
Airbus A321-200	5600	3480.42
Airbus A319-100	6700	4164.08
Boeing 737-300	4200	2610.32
Cessna 208 Caravan	1200	745.80
Bombardier CRJ-200	2700	1678.06

(9 строк)

Теперь обратимся к такому вопросу, как упорядочение строк при выводе. Если не принять специальных мер, то СУБД не гарантирует никакого конкретного порядка строк в результирующей выборке. Для упорядочения строк служит предложение ORDER BY. Например, если мы захотим разместить самолеты в порядке убывания дальности их полета, то нужно сделать так:

*SELECT * FROM aircrafts ORDER BY range DESC;*

demo=# SELECT * FROM aircrafts ORDER BY range DESC;		
aircraft_code	model	range
773	Boeing 777-300	11100
763	Boeing 767-300	7900
319	Airbus A319-100	6700
320	Airbus A320-200	5700
321	Airbus A321-200	5600
733	Boeing 737-300	4200
SU9	Sukhoi SuperJet-100	3000
CR2	Bombardier CRJ-200	2700
CN1	Cessna 208 Caravan	1200
(9 строк)		

Если сделать традиционную выборку, то мы получим список значений, среди которых будет много повторяющихся.

SELECT timezone FROM airports;

timezone
Asia/Yakutsk
Europe/Moscow
Asia/Novokuznetsk
Europe/Moscow
Europe/Moscow
Europe/Moscow
Asia/Yekaterinburg
Asia/Yekaterinburg
Europe/Moscow
Europe/Moscow
Europe/Moscow
Asia/Yekaterinburg
Europe/Samara
Asia/Yekaterinburg
Asia/Yekaterinburg
Asia/Krasnoyarsk
Asia/Irkutsk
Europe/Moscow
Asia/Yekaterinburg
Europe/Moscow
Asia/Krasnoyarsk

Конечно, это неудобно. Для того чтобы оставить в выборке только неповторяющиеся значения, служит ключевое слово DISTINCT:

SELECT DISTINCT timezone FROM airports ORDER BY 1;

timezone

Asia/Anadyr
Asia/Chita
Asia/Irkutsk
Asia/Kamchatka
Asia/Krasnoyarsk
Asia/Magadan
Asia/Novokuznetsk
Asia/Novosibirsk
Asia/Omsk
Asia/Sakhalin
Asia/Vladivostok
Asia/Yakutsk
Asia/Yekaterinburg
Europe/Kaliningrad
Europe/Moscow
Europe/Samara
Europe/Volgograd
(17 строк)

В таблице «Аэропорты» более ста записей. Если мы поставим задачу найти три самых восточных аэропорта, то для ее решения подошел бы такой запрос

```
SELECT airport_name, city, longitude
FROM airports
ORDER BY longitude DESC
LIMIT 3;
```

airport_name	city	longitude
-----+-----		
Анадырь	Анадырь	177.741483
Елизово	Петропавловск-Камчатский	158.453669
Магадан	Магадан	150.720439
(3 строки)		

А как найти еще три аэропорта, которые находятся немного западнее первой тройки, т. е. занимают места с четвертого по шестое? Для пропуска строк служит предложение OFFSET:

```
SELECT airport_name, city, longitude
FROM airports
ORDER BY longitude DESC
LIMIT 3
OFFSET 3;
```

airport_name	city	longitude
Хомутово	Южно-Сахалинск	142.717531
Хурба	Комсомольск-на-Амуре	136.934
Хабаровск-Новый	Хабаровск	135.188361

(3 строки)

В таблице «Самолеты» есть столбец «Максимальная дальность полета» (range). Мы можем дополнить вывод данных из этой таблицы столбцом «Класс самолета», имея в виду принадлежность каждого самолета к классу дальнемагистральных, среднемагистральных или ближнемагистральных судов. Воспользовавшись приведенной в указаниях к работе конструкцией в предложении SELECT и назначив новому столбцу имя с помощью ключевого слова AS, получим следующий запрос:

```
SELECT model, range,
CASE WHEN range < 2000 THEN 'Ближнемагистральный'
WHEN range < 5000 THEN 'Среднемагистральный'
ELSE 'Дальнемагистральный' END AS type
FROM aircrafts
ORDER BY model;
```

model	range	type
Airbus A319-100	6700	Дальнемагистральный
Airbus A320-200	5700	Дальнемагистральный
Airbus A321-200	5600	Дальнемагистральный
Boeing 737-300	4200	Среднемагистральный
Boeing 767-300	7900	Дальнемагистральный
Boeing 777-300	11100	Дальнемагистральный
Bombardier CRJ-200	2700	Среднемагистральный
Cessna 208 Caravan	1200	Ближнемагистральный
Sukhoi SuperJet-100	3000	Среднемагистральный

(9 строк)

В тех случаях, когда информации, содержащейся в одной таблице, недостаточно для получения требуемого результата, используют соединение (join) таблиц. Покажем способ выполнения соединения на примере следующего запроса: выбрать все места, предусмотренные компоновкой салона самолета Cessna 208 Caravan:


```

SELECT a.aircraft_code, a.model, s.seat_no, s.fare_conditions
FROM seats AS s
JOIN aircrafts AS a ON s.aircraft_code = a.aircraft_code
WHERE a.model ~ '^Cessna' ORDER BY s.seat_no;

```

aircraft_code	model	seat_no	fare_conditions
CN1	Cessna 208 Caravan	1A	Economy
CN1	Cessna 208 Caravan	1B	Economy
CN1	Cessna 208 Caravan	2A	Economy
CN1	Cessna 208 Caravan	2B	Economy
CN1	Cessna 208 Caravan	3A	Economy
CN1	Cessna 208 Caravan	3B	Economy
CN1	Cessna 208 Caravan	4A	Economy
CN1	Cessna 208 Caravan	4B	Economy
CN1	Cessna 208 Caravan	5A	Economy
CN1	Cessna 208 Caravan	5B	Economy
CN1	Cessna 208 Caravan	6A	Economy
CN1	Cessna 208 Caravan	6B	Economy

(12 строк)

Теперь обратимся к так называемым внешним соединениям. Зададимся вопросом: сколько маршрутов обслуживают самолеты каждого типа? Если не требовать вывода наименований моделей самолетов, тогда всю необходимую информацию можно получить из материализованного представления «Маршруты» (routes). Но мы все же будем выводить и наименования моделей, поэтому обратимся также к таблице «Самолеты» (aircrafts). Соединим эти таблицы на основе атрибута aircraft_code, сгруппируем строки и просто воспользуемся функцией count. В этом запросе внешнее соединение еще не используется.

```

SELECT r.aircraft_code, a.model, count( *) AS num_routes
FROM routes r
JOIN aircrafts a ON r.aircraft_code = a.aircraft_code
GROUP BY 1, 2
ORDER BY 3 DESC;

```

aircraft_code	model	num_routes
CR2	Bombardier CRJ-200	232
CN1	Cessna 208 Caravan	170
SU9	Sukhoi SuperJet-100	158
319	Airbus A319-100	46
733	Boeing 737-300	36
321	Airbus A321-200	32
763	Boeing 767-300	26
773	Boeing 777-300	10
(8 строк)		

Таблица «Самолеты» содержит 9 моделей, а в этой выборке лишь 8 строк. Значит, какая-то модель самолета не участвует в выполнении рейсов. Как ее выявить? С помощью такого запроса:

```
SELECT a.aircraft_code AS a_code, a.model, r.aircraft_code AS r_code,
count( r.aircraft_code ) AS num_routes
FROM aircrafts a
LEFT OUTER JOIN routes r ON r.aircraft_code = a.aircraft_code
GROUP BY 1, 2, 3
ORDER BY 4 DESC;
```

a_code	model	r_code	num_routes
CR2	Bombardier CRJ-200	CR2	232
CN1	Cessna 208 Caravan	CN1	170
SU9	Sukhoi SuperJet-100	SU9	158
319	Airbus A319-100	319	46
733	Boeing 737-300	733	36
321	Airbus A321-200	321	32
763	Boeing 767-300	763	26
773	Boeing 777-300	773	10
320	Airbus A320-200		0
(9 строк)			

В практической работе при выполнении выборок зачастую выполняются многотабличные запросы, включающие три таблицы и более. В качестве примера рассмотрим такую задачу: определить число пассажиров, не пришедших на регистрацию билетов и, следовательно, не вылетевших в пункт назначения. Будем учитывать только рейсы, у которых фактическое время вылета не пустое, т. е. рейсы, имеющие статус Departed или Arrived.

```

SELECT count( *)
FROM ( ticket_flights t
JOIN flights f ON t.flight_id = f.flight_id )
LEFT OUTER JOIN boarding_passes b ON t.ticket_no = b.ticket_no AND
t.flight_id = b.flight_id
WHERE f.actual_departure IS NOT NULL AND b.flight_id IS NULL;

```

count
0
(1 строка)

Оказывается, таких пассажиров нет. Вопрос: в какие города можно улететь либо из Москвы, либо из СанктПетербурга?

```

SELECT arrival_city
FROM routes WHERE departure_city = 'Москва'
UNION
SELECT arrival_city
FROM routes
WHERE departure_city = 'Санкт-Петербург'
ORDER BY arrival_city;

```

arrival_city
Абакан
Анадырь
Анапа
Архангельск
Астрахань
Барнаул
Белгород
Белоярский
Братск
Брянск
Бугульма
Владивосток
Владикавказ
Волгоград
Воркута
Воронеж
Геленджик
Горно-Алтайск
Грозный
Екатеринбург
Ижевск

Строки-дубликаты в результирующее множество не включаются. Для их включения нужно использовать UNION ALL.

Для расчета среднего значения по столбцу используется функция avg (от слова average).

```
SELECT avg( total_amount ) FROM bookings;
```

avg
79025.605811528685
(1 строка)

Рассмотрим два примера. Первый будет таким: давайте подсчитаем, сколько маршрутов предусмотрено из Москвы в другие города. При формировании запроса не будем учитывать частоту рейсов в неделю, т. е. независимо от того, выполняется какой-то рейс один раз в неделю или семь раз, он учитывается только однократно. Воспользуемся материализованным представлением «Маршруты».

```
SELECT arrival_city, count( * )
FROM routes
WHERE departure_city = 'Москва'
GROUP BY arrival_city
ORDER BY count DESC;
```

arrival_city	count
Санкт-Петербург	12
Брянск	9
Ульяновск	5
Йошкар-Ола	4
Петрозаводск	4
Ростов-на-Дону	3
Курган	3
Барнаул	3
Кемерово	3
Геленджик	3
Анадырь	3
Советский	3
Пенза	3
Сочи	2
Ставрополь	2
Тамбов	2
Томск	2
Махачкала	2
Бугульма	2
Усинск	2
Мурманск	2

В качестве второго примера рассмотрим ситуацию, когда руководству компании потребовалась обобщенная информация по частоте выполнения рейсов, а именно: сколько рейсов выполняется ежедневно, сколько рейсов — шесть дней в неделю, пять и т. д. Опять обратимся к материализованному представлению «Маршруты». Но теперь при формировании запроса, в отличие от первого примера, воспользуемся столбцом `days_of_week`, в котором содержатся массивы номеров дней недели, когда выполняется данный рейс.

```
SELECT array_length( days_of_week, 1 ) AS days_per_week, count( * ) AS
num_routes
FROM routes
GROUP BY days_per_week
ORDER BY 1 desc;
```

days_per_week	num_routes
7	482
3	54
2	88
1	86
(4 строки)	

В качестве примера приведем такой запрос: определить, сколько существует маршрутов из каждого города в другие города, и вывести названия городов, из которых в другие города существует не менее 15 маршрутов.

```
SELECT departure_city, count( * )
FROM routes
GROUP BY departure_city
HAVING count( * ) >= 15
ORDER BY count DESC;
```

departure_city	count
Москва	154
Санкт-Петербург	35
Новосибирск	19
Екатеринбург	15
(4 строки)	

Предположим, что руководство нашей компании хочет усовершенствовать тарифную политику и с этой целью просит нас предоставить сведения о распределении количества проданных билетов на некоторые рейсы во времени. Количество проданных билетов должно выводиться в виде накопленного показателя, суммирование должно производиться в пределах каждого календарного месяца.

Более детально, в столбцах `book_ref` и `book_date` выборки должны приводиться номер и время бронирования соответственно. В столбцах `month` и `day` должны указываться порядковый номер месяца и день этого месяца. Столбец `count` должен содержать суммарные (накопленные) количества билетов, проданных на каждый момент времени. С первого дня нового месяца подсчет числа проданных билетов начинается сначала.

```
SELECT b.book_ref, b.book_date, extract( 'month' from b.book_date ) AS  
month, extract( 'day' from b.book_date ) AS day, count( * ) OVER (  
    PARTITION BY date_trunc( 'month', b.book_date )  
    ORDER BY b.book_date ) AS count  
FROM ticket_flights tf  
JOIN tickets t ON tf.ticket_no = t.ticket_no  
JOIN bookings b ON t.book_ref = b.book_ref  
WHERE tf.flight_id = 1  
ORDER BY b.book_date;
```

book_ref	book_date	month	day	count
A60039	2016-08-22 04:02:00+00	8	22	1
554340	2016-08-23 15:04:00+00	8	23	2
854C4C	2016-08-24 02:52:00+00	8	24	5
854C4C	2016-08-24 02:52:00+00	8	24	5
854C4C	2016-08-24 02:52:00+00	8	24	5
81D8AF	2016-08-25 02:22:00+00	8	25	6
D9BB20	2016-08-25 11:47:00+00	8	25	7
11B509	2016-08-25 14:58:00+00	8	25	8
1F26A0	2016-08-25 19:51:00+00	8	25	9
3F351E	2016-08-25 21:57:00+00	8	25	10
429B95	2016-08-26 04:19:00+00	8	26	11
AAAF88	2016-08-26 07:07:00+00	8	26	12
496D37	2016-08-26 16:15:00+00	8	26	14
496D37	2016-08-26 16:15:00+00	8	26	14
205824	2016-08-26 22:38:00+00	8	26	15
6BECE5	2016-08-27 03:39:00+00	8	27	16
606590	2016-08-27 04:19:00+00	8	27	18
606590	2016-08-27 04:19:00+00	8	27	18
BE3F54	2016-08-27 04:34:00+00	8	27	19
F1B913	2016-08-27 09:47:00+00	8	27	21
F1B913	2016-08-27 09:47:00+00	8	27	21

Предположим, что сотрудникам аналитического отдела потребовалось провести статистическое исследование финансовых результатов работы авиакомпании. В качестве первого шага они решили подсчитать количество операций бронирования, в которых общая сумма превышает среднюю величину по всей выборке.

```
SELECT count( *)
FROM bookings
WHERE total_amount >
( SELECT avg( total_amount ) FROM bookings );
```

count

87224
(1 строка)

В качестве примера давайте выясним, какие маршруты существуют между городами часового пояса Asia/Krasnoyarsk. Подзапрос будет выдавать список городов из этого часового пояса, а в предложении WHERE главного запроса с помощью предиката IN будет выполняться проверка на принадлежность города этому списку. При этом подзапрос выполняется

только один раз для всего внешнего запроса, а не при обработке каждой строки из таблицы routes во внешнем запросе. Повторного выполнения подзапроса не требуется, т. к. его результат не зависит от значений, хранящихся в таблице routes. Такие подзапросы называются некоррелированными.

```
SELECT flight_no, departure_city, arrival_city
FROM routes
WHERE departure_city IN
( SELECT city FROM airports WHERE timezone ~ 'Krasnoyarsk' )
AND arrival_city IN
( SELECT city FROM airports WHERE timezone ~ 'Krasnoyarsk' );
```

flight_no	departure_city	arrival_city
PG0070	Абакан	Томск
PG0071	Томск	Абакан
PG0313	Абакан	Кызыл
PG0314	Кызыл	Абакан
PG0653	Красноярск	Барнаул
PG0654	Барнаул	Красноярск
(6 строк)		

Иногда возникают ситуации, когда от подзапроса требуется лишь установить сам факт наличия или отсутствия строк в конкретной таблице, удовлетворяющих определенному условию, а непосредственные значения атрибутов в этих строках интереса не представляют. В подобных случаях используют предикат EXISTS (или NOT EXISTS). В качестве примера выясним, в какие города нет рейсов из Москвы.

```
SELECT DISTINCT a.city
FROM airports a
WHERE NOT EXISTS
( SELECT * FROM routes r WHERE r.departure_city = 'Москва' AND
r.arrival_city = a.city ) AND a.city <> 'Москва'
ORDER BY city;
```


city
Благовещенск
Иваново
Иркутск
Калуга
Когалым
Комсомольск-на-Амуре
Кызыл
Магадан
Нижнекамск
Новокузнецк
Стрежевой
Сургут
Удачный
Усть-Илимск
Усть-Кут
Ухта
Череповец
Чита
Якутск
Ярославль
(20 строк)

В качестве еще одного примера использования подзапроса в предложении FROM решим такую задачу: получить перечень аэропортов в тех городах, в которых больше одного аэропорта.

```

SELECT aa.city, aa.airport_code, aa.airport_name
FROM ( SELECT city, count( *)
FROM airports
GROUP BY city
HAVING count( *) > 1 ) AS a
JOIN airports AS aa ON a.city = aa.city
ORDER BY aa.city, aa.airport_name;

```

city	airport_code	airport_name
Москва	VKO	Внуково
Москва	DME	Домодедово
Москва	SVO	Шереметьево
Ульяновск	ULV	Баратаевка
Ульяновск	ULY	Ульяновск-Восточный
(5 строк)		

Для иллюстрации использования подзапросов в предложении HAVING решим такую задачу: определить число маршрутов, исходящих из тех аэропортов, которые расположены восточнее географической долготы 150°.

```
SELECT departure_airport, departure_city, count( *)
FROM routes
GROUP BY departure_airport, departure_city
HAVING departure_airport IN
( SELECT airport_code FROM airports WHERE longitude > 150 )
ORDER BY count DESC;
```

departure_airport	departure_city	count
DYR	Анадырь	4
GDX	Магадан	3
PKC	Петропавловск-Камчатский	1
(3 строки)		

В сложных запросах могут использоваться вложенные подзапросы. Это означает, что один подзапрос находится внутри другого. Давайте в качестве примера рассмотрим такую ситуацию: руководство авиакомпании хочет выяснить степень заполнения самолетов на всех рейсах, ведь отправлять полупустые самолеты не очень выгодно. Таким образом, запрос должен не только выдавать число билетов, проданных на данный рейс, и общее число мест в самолете, но должен также вычислять отношение этих двух показателей. Вот какой запрос получился:

```
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;

flight_id	flight_no	scheduled_departure_local	departure_city	arrival_city	model	fact_passengers	total_seats	fraction
28205	PG0032	2016-09-13 08:00:00	Пенза	Москва	Cessna 208 Caravan	2	12	0.17
9467	PG0360	2016-09-13 08:00:00	Санкт-Петербург	Оренбург	Bombardier CRJ-200	6	50	0.12
7130	PG0591	2016-09-13 08:00:00	Москва	Томск	Sukhoi SuperJet-100	25	97	0.26
6223	PG0120	2016-09-13 08:00:00	Москва	Мирный	Boeing 737-300	34	130	0.26
1764	PG0239	2016-09-13 08:05:00	Москва	Ханты-Мансийск	Sukhoi SuperJet-100	55	97	0.57
32948	PG0550	2016-09-13 08:05:00	Владикавказ	Москва	Bombardier CRJ-200	11	50	0.22
3625	PG0414	2016-09-13 08:05:00	Москва	Мурманск	Bombardier CRJ-200	13	50	0.26
9307	PG0198	2016-09-13 08:10:00	Санкт-Петербург	Иркутск	Airbus A321-200	19	170	0.11
15025	PG0683	2016-09-13 08:15:00	Пермь	Ульяновск	Sukhoi SuperJet-100	32	97	0.33
21969	PG0240	2016-09-13 08:15:00	Ханты-Мансийск	Москва	Sukhoi SuperJet-100	5	97	0.05
1452	PG0509	2016-09-13 08:15:00	Москва	Элиста	Sukhoi SuperJet-100	20	97	0.21
2485	PG0482	2016-09-13 08:20:00	Москва	Кемерово	Sukhoi SuperJet-100	15	97	0.15
25029	PG0044	2016-09-13 08:20:00	Ижевск	Москва	Bombardier CRJ-200	4	50	0.08
15750	PG0244	2016-09-13 08:20:00	Якутск	Санкт-Петербург	Airbus A319-100	5	116	0.04
17534	PG0123	2016-09-13 08:25:00	Нижнекамск	Ростов-на-Дону	Bombardier CRJ-200	2	50	0.04
28883	PG0177	2016-09-13 08:25:00	Чебоксары	Москва	Bombardier CRJ-200	8	50	0.16
24836	PG0014	2016-09-13 08:30:00	Тюмень	Урай	Sukhoi SuperJet-100	41	97	0.42
16590	PG0369	2016-09-13 08:30:00	Нижнеартовск	Москва	Bombardier CRJ-200	14	50	0.28
1	PG0405	2016-09-13 08:35:00	Москва	Санкт-Петербург	Airbus A321-200	79	170	0.46
7072	PG0677	2016-09-13 08:40:00	Москва	Махачкала	Boeing 737-300	96	130	0.74
2575	PG0029	2016-09-13 08:40:00	Москва	Пенза	Cessna 208 Caravan	4	12	0.33
7364	PG0241	2016-09-13 08:40:00	Москва	Чебоксары	Sukhoi SuperJet-100	56	97	0.58
11646	PG0508	2016-09-13 08:45:00	Казань	Санкт-Петербург	Sukhoi SuperJet-100	12	97	0.12
8630	PG0476	2016-09-13 08:45:00	Санкт-Петербург	Москва	Airbus A321-200	84	170	0.49
3316	PG0227	2016-09-13 08:45:00	Москва	Санкт-Петербург	Airbus A321-200	93	170	0.55

Самый внутренний подзапрос — total_seats — выдает общее число мест в самолете. Этот подзапрос — коррелированный, т. к. он выполняется для каждой строки, обрабатываемой во внешнем подзапросе, т. е. для каждой модели самолета. Для подсчета числа проданных билетов мы использовали соединение представления «Рейсы» (flights_v) с таблицей «Перелеты» (ticket_flights) с последующей группировкой строк и вызовом функции count.

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

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;

```

flight_id	flight_no	scheduled_departure_local	departure_city	arrival_city	model	fact_passengers	total_seats	fraction
28205	PG0032	2016-09-13 08:00:00	Пенза	Москва	Cessna 208 Caravan	2	12	0.17
9467	PG0360	2016-09-13 08:00:00	Санкт-Петербург	Оренбург	Bombardier CRJ-200	6	50	0.12
7130	PG0591	2016-09-13 08:00:00	Москва	Томск	Sukhoi SuperJet-100	25	97	0.26
6223	PG0120	2016-09-13 08:00:00	Москва	Мирный	Boeing 737-300	34	130	0.26
1764	PG0239	2016-09-13 08:05:00	Москва	Ханты-Мансийск	Sukhoi SuperJet-100	55	97	0.57
32948	PG0550	2016-09-13 08:05:00	Владикавказ	Москва	Bombardier CRJ-200	11	50	0.22
3625	PG0414	2016-09-13 08:05:00	Москва	Мурманск	Bombardier CRJ-200	13	50	0.26
9307	PG0198	2016-09-13 08:10:00	Санкт-Петербург	Иркутск	Airbus A321-200	19	170	0.11
15025	PG0683	2016-09-13 08:15:00	Пермь	Ульяновск	Sukhoi SuperJet-100	32	97	0.33
21969	PG0240	2016-09-13 08:15:00	Ханты-Мансийск	Москва	Sukhoi SuperJet-100	5	97	0.05
1452	PG0509	2016-09-13 08:15:00	Москва	Элиста	Sukhoi SuperJet-100	20	97	0.21
2485	PG0482	2016-09-13 08:20:00	Москва	Кемерово	Sukhoi SuperJet-100	15	97	0.15
25029	PG0044	2016-09-13 08:20:00	Ижевск	Москва	Bombardier CRJ-200	4	50	0.08
15750	PG0244	2016-09-13 08:20:00	Якутск	Санкт-Петербург	Airbus A319-100	5	116	0.04
17534	PG0123	2016-09-13 08:25:00	Нижнекамск	Ростов-на-Дону	Bombardier CRJ-200	2	50	0.04
28883	PG0177	2016-09-13 08:25:00	Чебоксары	Москва	Bombardier CRJ-200	8	50	0.16
24836	PG0014	2016-09-13 08:30:00	Тюмень	Урай	Sukhoi SuperJet-100	41	97	0.42
16590	PG0369	2016-09-13 08:30:00	Нижневартовск	Москва	Bombardier CRJ-200	14	50	0.28
1	PG0405	2016-09-13 08:35:00	Москва	Санкт-Петербург	Airbus A321-200	79	170	0.46
7072	PG0677	2016-09-13 08:40:00	Москва	Махачкала	Boeing 737-300	96	130	0.74
2575	PG0029	2016-09-13 08:40:00	Москва	Пенза	Cessna 208 Caravan	4	12	0.33
7364	PG0241	2016-09-13 08:40:00	Москва	Чебоксары	Sukhoi SuperJet-100	56	97	0.58
11646	PG0508	2016-09-13 08:45:00	Казань	Санкт-Петербург	Sukhoi SuperJet-100	12	97	0.12
8630	PG0476	2016-09-13 08:45:00	Санкт-Петербург	Москва	Airbus A321-200	84	170	0.49
3316	PG0227	2016-09-13 08:45:00	Москва	Санкт-Петербург	Airbus A321-200	93	170	0.55

Конструкция WITH ts AS (...) и представляет собой общее табличное выражение (СТЕ). Такие конструкции удобны тем, что позволяют упростить основной запрос, сделать его менее громоздким.

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 '% % % %';
```

```

demo=# SELECT count( * ) FROM tickets;
count
-----
366733
(1 строка)

demo=# SELECT count( * ) FROM tickets WHERE passenger_name LIKE '% %';
count
-----
366733
(1 строка)

demo=# SELECT count( * ) FROM tickets WHERE passenger_name LIKE '% % %';
count
-----
0
(1 строка)

demo=# SELECT count( * ) FROM tickets WHERE passenger_name LIKE '% % %';
count
-----
366733
(1 строка)

```

Проанализировав полученные результаты, можно сказать, что пробелы обрабатываются точно так же, как и любые другие символы вместе с маской – 26 мы знаем, что данные пассажиров разделены одним пробелом, соответственно наличие группы «% % %» требует присутствия в строке трех групп слов, разделенных пробелом, чего в данной таблице не представлено.

2. Предложить шаблон поиска в операторе LIKE для выбора из таблицы «Билеты» всех пассажиров с фамилиями, состоящими из пяти букв.

SELECT passenger_name FROM tickets WHERE passenger_name LIKE '_____ %';

passenger_name
ARTUR GERASIMOV
ALINA VOLKOVA
IRINA ANTONOVA
ANTON BONDARENKO
RAISA KONOVALOVA
DENIS KUZNECOV
YURIY MEDVEDEV
PAVEL GUSEV
ELENA STEPANOVA
RAISA ROMANOVA
IRINA KAZAKOVA
SOFYA DAVYDOVA
ROMAN BARANOV
DIANA NESTEROVA
YURIY ROMANOV
PAVEL GERASIMOV
ELENA AFANASEVA
RAISA KONOVALOVA
ANTON POPOV
VADIM NIKOLAEV
ELENA ZAYCEVA
ARTEM BELOV
PAVEL MEDVEDEV
ALENA EFREMOVA
PAVEL MIKHAYLOV

3. В разделе документации 9.7.2 «Регулярные выражения SIMILAR TO» рассматривается оператор SIMILAR TO. Он работает аналогично оператору LIKE, но использует шаблоны, соответствующие определению регулярных выражений, приведенному в стандарте SQL. Регулярные выражения SQL представляют собой комбинацию синтаксиса LIKE с синтаксисом обычных регулярных выражений. Самостоятельно ознакомьтесь с оператором SIMILAR TO.

SIMILAR TO сопоставляет строку с шаблоном регулярного выражения SQL. В отличие от некоторых других языков, шаблон должен соответствовать всей строке для успешного выполнения - совпадения подстроки недостаточно. Если какой-либо операнд равен NULL, результатом будет NULL. В противном случае результатом будет TRUE или FALSE.

'abc' SIMILAR TO 'abc' *true*

'abc' SIMILAR TO 'a' *false*

'abc' SIMILAR TO '%(b d)%'	<i>true</i>
'abc' SIMILAR TO '(b c)%'	<i>false</i>
'-abc-' SIMILAR TO '%\mabc\M%'	<i>true</i>
'xabcy' SIMILAR TO '%\mabc\M%'	<i>false</i>

4. В разделе документации 9.2 «Функция и операторы сравнения» представлены различные предикаты сравнения, кроме предиката BETWEEN, рассмотренного в этой главе. Самостоятельно ознакомьтесь с ними.

Предикат	Описание
a BETWEEN x AND y	между
a NOT BETWEEN x AND y	не между
a BETWEEN SYMMETRIC x AND y	между, после сортировки сравниваемых значений
a NOT BETWEEN SYMMETRIC x AND y	не между, после сортировки сравниваемых значений
a IS DISTINCT FROM b	не равно, при этом NULL воспринимается как обычное значение
a IS NOT DISTINCT FROM b	равно, при этом NULL воспринимается как обычное значение
выражение IS NULL	эквивалентно NULL
выражение IS NOT NULL	не эквивалентно NULL
выражение ISNULL	эквивалентно NULL (нестандартный синтаксис)
выражение NOTNULL	не эквивалентно NULL (нестандартный синтаксис)
логическое_выражение IS TRUE	истина
логическое_выражение IS NOT TRUE	ложь или неопределённость
логическое_выражение IS FALSE	ложь
логическое_выражение IS NOT	FALSE истина или неопределённость
логическое_выражение IS UNKNOWN	неопределённость
логическое_выражение IS NOT UNKNOWN	истина или ложь

5. В разделе документации 9.17 «Условные выражения» представлены условные выражения, которые поддерживаются в PostgreSQL. В тексте главы

была рассмотрена конструкция CASE. Самостоятельно ознакомьтесь с функциями COALESCE, NULLIF, GREATEST и LEAST.

Oracle/PLSQL функция COALESCE возвращает первое не NULL выражение из списка. Если все выражения определены как Null, то функция COALESCE вернет Null.

```
SELECT COALESCE(description, short_description, '(none)')
```

Этот запрос вернёт значение description, если оно не равно NULL, либо short_description, если оно не NULL, и строку (none), если оба эти значения равны NULL. Аргументы должны быть приводимыми к одному общему типу, который и будет типом результата.

Функция NULLIF сравнивает значение1 и значение2, если они равны - выдаёт NULL, если нет - возвращает значение1. Это может быть полезно для реализации обратной операции к COALESCE.

Функция GREATEST возвращает наибольшее значение в списке выражений. Функция LEAST возвращает наименьшее значение в списке выражений. Выражения должны приводиться к общему типу данных, который станет типом результата. Значения NULL в этом списке игнорируются. Результат выражения будет равен NULL, только если все его аргументы равны NULL.

```
GREATEST( expr1, expr2, ... expr_n )
```

```
LEAST( expr1, expr2, ... expr_n )
```

6. 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.

```
SELECT flight_no, model
```

```
FROM routes
```

```
JOIN aircrafts ON aircrafts.aircraft_code = routes.aircraft_code
```

```
WHERE model LIKE 'Boeing%';
```


flight_no	model
PG0013	Boeing 777-300
PG0073	Boeing 737-300
PG0091	Boeing 737-300
PG0092	Boeing 737-300
PG0108	Boeing 767-300
PG0109	Boeing 767-300
PG0120	Boeing 737-300
PG0121	Boeing 737-300
PG0140	Boeing 767-300
PG0141	Boeing 767-300
PG0144	Boeing 767-300
PG0145	Boeing 767-300
PG0194	Boeing 737-300
PG0195	Boeing 737-300
PG0200	Boeing 767-300
PG0201	Boeing 767-300
PG0208	Boeing 767-300
PG0209	Boeing 767-300

7. 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.

```
SELECT DISTINCT GREATEST(departure_city, arrival_city),
LEAST(departure_city, arrival_city)
FROM routes
JOIN aircrafts ON aircrafts.aircraft_code = routes.aircraft_code
WHERE aircrafts.model = 'Boeing 777-300';
```

greatest	least
Москва	Екатеринбург
Новосибирск	Москва
Пермь	Москва
Сочи	Москва
(4 строки)	

8. Сколько рейсов выполняется из Москвы в Санкт-Петербург? Получить результат в следующем виде: departure_city, arrival_city, count.

```
SELECT count (*)
FROM routes
WHERE departure_city = 'Москва' AND arrival_city = 'Санкт-Петербург';
```

count

12
(1 строка)

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

```
SELECT departure_city, count( *)
```

```
FROM routes
```

```
GROUP BY departure_city
```

```
ORDER BY count DESC;
```

departure_city	count
-----	-----
Москва	154
Санкт-Петербург	35
Новосибирск	19
Екатеринбург	15
Ростов-на-Дону	14
Сочи	14
Красноярск	13
Ульяновск	11
Пермь	11
Сургут	11
Ханты-Мансийск	10
Казань	10
Новокузнецк	10
Брянск	10
Архангельск	9
Тюмень	9
Советский	9
Иркутск	9
Новый Уренгой	9
Хабаровск	8
Челябинск	8
Петрозаводск	8
Томск	8
Абакан	7
Краснодар	7

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

```
SELECT departure_city, count(DISTINCT arrival_city)
```

FROM routes

GROUP BY departure_city

ORDER BY count DESC;

departure_city	count
Москва	80
Санкт-Петербург	22
Новосибирск	14
Екатеринбург	13
Сочи	11
Ростов-на-Дону	10
Красноярск	10
Казань	9
Иркутск	9
Тюмень	9
Сургут	9
Пермь	9
Архангельск	8
Новокузнецк	8
Хабаровск	7
Томск	7
Ханты-Мансийск	7
Краснодар	7
Советский	7
Челябинск	7
Элиста	6
Нижний Новгород	6
Новый Уренгой	6
Уфа	6
Ульяновск	6

11. 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.

SELECT arrival_city, array_length(days_of_week, 1) as len, count()*

FROM routes

WHERE departure_city = 'Москва'

GROUP BY arrival_city, days

ORDER BY len DESC, count DESC

LIMIT 5;

arrival_city	len	count
Санкт-Петербург	7	12
Брянск	7	9
Ульяновск	7	5
Йошкар-Ола	7	4
Петрозаводск	7	4
(5 строк)		

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

```
SELECT unnest(days_of_week) AS days, count(*)
FROM routes
WHERE departure_city = 'Москва'
GROUP BY days
ORDER BY days;
```

days	count
1	131
2	134
3	127
4	135
5	124
6	133
7	124
(7 строк)	

13. Каковы максимальные и минимальные цены билетов на все направления. Оператор SELECT должен возвращать departure_city, arrival_city, max(amount), min(amount)

```
SELECT departure_city, arrival_city, max(amount), min(amount)
FROM flights_v
JOIN ticket_flights as t ON ticket_flights.flight_id = flights_v.flight_id
GROUP BY departure_city, arrival_city
ORDER BY departure_city;
```

departure_city	arrival_city	arrival_city	max	min
Абакан	Новосибирск	Новосибирск	5800.00	5800.00
Абакан	Москва	Москва	101000.00	33700.00
Абакан	Томск	Томск	4900.00	4900.00
Анадырь	Хабаровск	Хабаровск	92200.00	30700.00
Анадырь	Москва	Москва	185300.00	61800.00
Анапа	Москва	Москва	36600.00	12200.00
Анапа	Белгород	Белгород	18900.00	6300.00
Архангельск	Томск	Томск	28100.00	25500.00
Архангельск	Ханты-Мансийск	Ханты-Мансийск	16400.00	14900.00
Архангельск	Нарьян-Мар	Нарьян-Мар	7300.00	6600.00
Архангельск	Москва	Москва	11100.00	10100.00
Архангельск	Тюмень	Тюмень	17100.00	15500.00
Архангельск	Пермь	Пермь	11000.00	11000.00
Астрахань	Москва	Москва	14300.00	12400.00
Астрахань	Барнаул	Барнаул	29000.00	26400.00
Барнаул	Астрахань	Астрахань	29000.00	26400.00
Барнаул	Москва	Москва	88300.00	29100.00
Белгород	Москва	Москва	16100.00	5400.00
Белгород	Анапа	Анапа	18900.00	6300.00
Белгород	Брянск	Брянск	9900.00	3300.00
Белгород	Сочи	Сочи	8400.00	8400.00
Благовещенск	Хабаровск	Хабаровск	18000.00	6000.00
Братск	Москва	Москва	115000.00	38300.00
Брянск	Белгород	Белгород	9900.00	3300.00
Брянск	Москва	Москва	11100.00	3300.00

5 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.