

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ № 4

Запросы
Тема

Преподаватель		Е. П. Моргунов
	Подпись, дата	Инициалы, Фамилия
Студент	КИ19-17/1Б, №031939174	А. К. Никитин
	Номер группы, зачетной книжки	Инициалы, Фамилия

Красноярск 2021

1 Ход выполнения

1.1 Вопрос 1

```
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)
```

Рисунок 1 – Проверка поиска по шаблону с наличием только пробела в шаблоне

1.2 Вопрос 3

```
demo=# SELECT 'abc' SIMILAR TO '%(b|d)%';
?column?
-----
t
(1 строка)

demo=# SELECT 'abc' LIKE '%(b|d)%';
?column?
-----
f
(1 строка)
```

Рисунок 2 – Отличие LIKE и SIMILAR TO

1.3 Вопрос 5

```
edu=# SELECT * FROM students;
record_book | name | psp_ser | psp_num
-----+-----+-----+-----
11111 | Иванов Иван Иванович | 1111 | 111111
22222 | Павлов Павел Павлович | 2222 | 222222
33333 | Андреев Андрей Андреевич | 3333 | 333333
44444 | Николаев Николай Николаевич | 4444 | 444444
55555 | Федоров Федор Федорович | 5555 | 555555
66666 | Петров Петр Петрович | | 
(6 строк)

edu=# SELECT COALESCE (psp_num, 77777) FROM students;
coalesce
-----
111111
222222
333333
444444
555555
777777
(6 строк)
```

Рисунок 3 – Использование оператора COALESCE

```
edu=# SELECT * FROM students;
record_book | name | psp_ser | psp_num
-----+-----+-----+-----
11111 | Иванов Иван Иванович | 1111 | 111111
22222 | Павлов Павел Павлович | 2222 | 222222
33333 | Андреев Андрей Андреевич | 3333 | 333333
44444 | Николаев Николай Николаевич | 4444 | 444444
55555 | Федоров Федор Федорович | 5555 | 555555
66666 | Петров Петр Петрович | | 
(6 строк)

edu=# SELECT NULLIF (psp_ser, 2222) FROM students;
nullif
-----
1111

3333
4444
5555
(6 строк)
```

Рисунок 4 – Использование оператора NULLIF

```

edu=# SELECT * FROM students;
record_book | name | psp_ser | psp_num
-----+-----+-----+-----
11111 | Иванов Иван Иванович | 1111 | 111111
22222 | Павлов Павел Павлович | 2222 | 222222
33333 | Андреев Андрей Андреевич | 3333 | 333333
44444 | Николаев Николай Николаевич | 4444 | 444444
55555 | Федоров Федор Федорович | 5555 | 555555
66666 | Петров Петр Петрович | | 
(6 строк)

edu=# SELECT GREATEST(300000, psp_num) FROM students;
greatest
-----
300000
300000
333333
444444
555555
300000
(6 строк)

```

Рисунок 5 – Использование оператора GREATEST

1.4 Вопрос 7

```

demo=# SELECT DISTINCT departure_city, arrival_city
FROM routes r
JOIN aircrafts a ON r.aircraft_code = a.aircraft_code
WHERE a.model = 'Боинг 777-300'
ORDER BY 1;
departure_city | arrival_city
-----+-----
Екатеринбург | Москва
Москва | Екатеринбург
Москва | Новосибирск
Москва | Пермь
Москва | Сочи
Новосибирск | Москва
Пермь | Москва
Сочи | Москва
(8 строк)

demo=# SELECT DISTINCT departure_city, arrival_city
FROM routes r
JOIN aircrafts a ON r.aircraft_code = a.aircraft_code AND departure_city > arrival_city
WHERE a.model = 'Боинг 777-300'
ORDER BY 1;
departure_city | arrival_city
-----+-----
Москва | Екатеринбург
Новосибирск | Москва
Пермь | Москва
Сочи | Москва
(4 строки)

```

Рисунок 6 – Модификация запроса для устранения дубликатов пар строк

1.5 Вопрос 9

```
demo=# SELECT count( * )
FROM routes
WHERE departure_city = 'Москва'
AND arrival_city = 'Санкт-Петербург';
count
-----
      12
(1 строка)
```

```
demo=# SELECT departure_city, arrival_city, count( * )
FROM routes
WHERE departure_city = 'Москва'
AND arrival_city = 'Санкт-Петербург'
GROUP BY departure_city, arrival_city;
departure_city | arrival_city | count
-----+-----+-----
Москва         | Санкт-Петербург |      12
(1 строка)
```

Рисунок 7 – Модификация запроса через GROUP BY

1.6 Вопрос 11

```
demo=# SELECT arrival_city, array_length( days_of_week::integer[], 1 ) AS flights_number FROM routes
WHERE departure_city = 'Москва'
GROUP BY arrival_city, flights_number
ORDER BY flights_number DESC
LIMIT 5;
arrival_city | flights_number
-----+-----
Анапа       |              7
Архангельск |              7
Астрахань   |              7
Белгород     |              7
Братск       |              7
(5 строк)
```

Рисунок 8 – Вывод пяти городов с наибольшим количеством вылетов из
столицы

1.7 Вопрос 13

```
demo=# SELECT f.departure_city, f.arrival_city,
max( tf.amount ), min( tf.amount )
FROM flights_v f
LEFT OUTER JOIN ticket_flights tf ON f.flight_id = tf.flight_id
GROUP BY 1, 2
ORDER BY 1, 2
LIMIT 5;
```

departure_city	arrival_city	max	min
Абакан	Архангельск		
Абакан	Грозный		
Абакан	Кызыл		
Абакан	Москва	101000.00	33700.00
Абакан	Новосибирск	5800.00	5800.00

(5 строк)

Рисунок 9 – Вывод полей с NULL, если билетов нет

1.8 Вопрос 15

```
demo=# SELECT
demo=# *,
demo=# SUM(total_amount) OVER (PARTITION BY book_date::date) AS date_amount,
demo=# AVG(total_amount) OVER (PARTITION BY book_date::date) AS date_average
demo=# FROM bookings
demo=# LIMIT 20;
```

book_ref	book_date	total_amount	date_amount	date_average
61EAC7	2017-06-21 18:05:00+07	52000.00	441900.00	147300.000000000000
98A065	2017-06-21 19:48:00+07	211100.00	441900.00	147300.000000000000
CA287C	2017-06-21 20:17:00+07	178800.00	441900.00	147300.000000000000
D316F5	2017-06-22 08:39:00+07	41600.00	562800.00	93800.000000000000
4BAE8E	2017-06-22 11:22:00+07	46300.00	562800.00	93800.000000000000
33DAE5	2017-06-22 05:29:00+07	57500.00	562800.00	93800.000000000000
8156F1	2017-06-22 14:03:00+07	224300.00	562800.00	93800.000000000000
1BB83B	2017-06-22 05:52:00+07	123000.00	562800.00	93800.000000000000
1731E8	2017-06-22 09:46:00+07	70100.00	562800.00	93800.000000000000
22B61D	2017-06-23 06:35:00+07	111800.00	1818300.00	106958.823529411765
12E05A	2017-06-23 16:59:00+07	95600.00	1818300.00	106958.823529411765
2566D4	2017-06-23 09:17:00+07	64700.00	1818300.00	106958.823529411765
30E106	2017-06-23 02:25:00+07	212500.00	1818300.00	106958.823529411765
5401F7	2017-06-23 13:01:00+07	49800.00	1818300.00	106958.823529411765
2EFBFF	2017-06-23 12:33:00+07	68800.00	1818300.00	106958.823529411765
A4B096	2017-06-23 13:07:00+07	124200.00	1818300.00	106958.823529411765
A49BF9	2017-06-23 14:21:00+07	383300.00	1818300.00	106958.823529411765
B66C02	2017-06-23 18:53:00+07	27200.00	1818300.00	106958.823529411765
E0355F	2017-06-23 22:41:00+07	55800.00	1818300.00	106958.823529411765
C6940E	2017-06-23 19:15:00+07	126300.00	1818300.00	106958.823529411765

(20 строк)

Рисунок 10 – Вычисление суммы и среднего значения билетов по дате

```

demo=# SELECT
*,
SUM(total_amount) OVER (PARTITION BY book_date::date ORDER BY book_date) AS date_amount,
AVG(total_amount) OVER (PARTITION BY book_date::date ORDER BY book_date) AS date_average
FROM bookings
LIMIT 20;

```

book_ref	book_date	total_amount	date_amount	date_average
61EAC7	2017-06-21 18:05:00+07	52000.00	52000.00	52000.000000000000
98A065	2017-06-21 19:48:00+07	211100.00	263100.00	131550.000000000000
CA287C	2017-06-21 20:17:00+07	178800.00	441900.00	147300.000000000000
33DAE5	2017-06-22 05:29:00+07	57500.00	57500.00	57500.000000000000
1BB83B	2017-06-22 05:52:00+07	123000.00	180500.00	90250.000000000000
D316F5	2017-06-22 08:39:00+07	41600.00	222100.00	74033.333333333333
1731E8	2017-06-22 09:46:00+07	70100.00	292200.00	73050.000000000000
4BAE8E	2017-06-22 11:22:00+07	46300.00	338500.00	67700.000000000000
8156F1	2017-06-22 14:03:00+07	224300.00	562800.00	93800.000000000000
30E106	2017-06-23 02:25:00+07	212500.00	212500.00	212500.000000000000
B9FDC0	2017-06-23 05:05:00+07	28000.00	240500.00	120250.000000000000
E6B2BE	2017-06-23 06:31:00+07	44000.00	284500.00	94833.333333333333
22B61D	2017-06-23 06:35:00+07	111800.00	396300.00	99075.000000000000
B9CEAC	2017-06-23 07:49:00+07	32700.00	429000.00	85800.000000000000
2566D4	2017-06-23 09:17:00+07	64700.00	493700.00	82283.333333333333
7CED82	2017-06-23 12:31:00+07	12600.00	506300.00	72328.571428571429
2EFBFF	2017-06-23 12:33:00+07	68800.00	575100.00	71887.500000000000
5401F7	2017-06-23 13:01:00+07	49800.00	624900.00	69433.333333333333
A4B096	2017-06-23 13:07:00+07	124200.00	749100.00	74910.000000000000
A49BF9	2017-06-23 14:21:00+07	383300.00	1132400.00	102945.454545454545

(20 строк)

Рисунок 11 – Добавление сортировки к предыдущему запросу

1.9 Вопрос 17

```
demo=# SELECT a.aircraft_code,
demo-# a.model,
demo-# s.fare_conditions,
demo-# count( * ) AS num
demo-# FROM aircrafts a
demo-# JOIN seats s ON a.aircraft_code = s.aircraft_code
demo-# GROUP BY 1, 2, 3
demo-# ORDER BY 1, 2, 3;
```

aircraft_code	model	fare_conditions	num
319	Аэробус A319-100	Business	20
319	Аэробус A319-100	Economy	96
320	Аэробус A320-200	Business	20
320	Аэробус A320-200	Economy	120
321	Аэробус A321-200	Business	28
321	Аэробус A321-200	Economy	142
733	Боинг 737-300	Business	12
733	Боинг 737-300	Economy	118
763	Боинг 767-300	Business	30
763	Боинг 767-300	Economy	192
773	Боинг 777-300	Business	30
773	Боинг 777-300	Comfort	48
773	Боинг 777-300	Economy	324
CN1	Сессна 208 Караван	Economy	12
CR2	Бомбардье CRJ-200	Economy	50
SU9	Сухой Суперджет-100	Business	12
SU9	Сухой Суперджет-100	Economy	85

(17 строк)

Рисунок 12 – Распределение мест с разными классами обслуживания

1.10 Вопрос 19

1.10.1 Задание 1

```
demo=# WITH RECURSIVE ranges ( min_sum, max_sum, iteration )
demo-# AS (
demo(# VALUES( 0, 100000, 0 ),
demo(# ( 100000, 200000, 0 ),
demo(# ( 200000, 300000, 0 )
demo(# UNION ALL
demo(# SELECT min_sum + 100000, max_sum + 100000, iteration + 1
demo(# FROM ranges
demo(# WHERE max_sum < ( SELECT max( total_amount ) FROM bookings )
demo(# )
demo-# SELECT * FROM ranges;
```

min_sum	max_sum	iteration
0	100000	0
100000	200000	0
200000	300000	0
100000	200000	1
200000	300000	1
300000	400000	1
200000	300000	2
300000	400000	2
400000	500000	2
300000	400000	3
400000	500000	3
500000	600000	3
400000	500000	4
500000	600000	4
600000	700000	4
500000	600000	5
600000	700000	5
700000	800000	5
600000	700000	6
700000	800000	6
800000	900000	6
700000	800000	7
800000	900000	7
900000	1000000	7

Рисунок 13 – Добавление поля номера итерации

1.10.2 Задание 2

```
demo=# WITH RECURSIVE ranges ( min_sum, max_sum)
AS (
VALUES( 0, 100000 ),
( 100000, 200000 ),
( 200000, 300000 )
UNION
SELECT min_sum + 100000, max_sum + 100000
FROM ranges
WHERE max_sum < ( SELECT max( total_amount ) FROM bookings )
)
SELECT * FROM ranges;
 min_sum | max_sum
-----+-----
         0 | 100000
    100000 | 200000
    200000 | 300000
    300000 | 400000
    400000 | 500000
    500000 | 600000
    600000 | 700000
    700000 | 800000
    800000 | 900000
    900000 | 1000000
   1000000 | 1100000
   1100000 | 1200000
   1200000 | 1300000
(13 строк)
```

Рисунок 14 – Замена UNION ALL на UNION

Сравнив текущий запрос с предыдущим с исходным, наблюдаем уменьшение количества строк в 3 раза.

1.11 Вопрос 21

```

demo=# SELECT city
demo-# FROM airports
demo-# WHERE city <> 'Москва'
demo-# EXCEPT
demo-# SELECT arrival_city
demo-# FROM routes
demo-# WHERE departure_city = 'Москва'
demo-# ORDER BY city;
      city
-----
Благовещенск
Иваново
Иркутск
Калуга
Когалым
Комсомольск-на-Амуре
Кызыл
Магадан
Нижекамск
Новокузнецк
Стрежевой
Сургут
Удачный
Усть-Илимск
Усть-Кут
Ухта
Череповец
Чита
Якутск
Ярославль
(20 строк)

```

Рисунок 15 – Выбор городов, в которые нет рейса из Москвы, при помощи
EXCEPT

Был выбран именно оператор EXCEPT, а не UNION или INTERSECT, так как нам нужно найти такие строки таблицы, в которых исходная часть таблицы состоит без другой, что логически соответствует операции логической разности или оператору EXCEPT в PostgreSQL.

1.12 Вопрос 23

```
demo=# WITH a1 AS (  
demo(# SELECT DISTINCT city FROM airports  
demo(# ), a2 AS (  
demo(# SELECT DISTINCT city FROM airports  
demo(# )  
demo-# SELECT count( * ) FROM a1, a2  
demo-# WHERE a1.city <> a2.city AND a2.city <> a1.city;  
count  
-----  
10100  
(1 строка)
```

Рисунок 16 – Переписывание запроса с JOIN через общие табличные выражения

1.13 Вопрос 25

```
demo=# SELECT f.flight_id,  
demo-# f.flight_no,  
demo-# f.departure_city,  
demo-# f.arrival_city,  
demo-# f.aircraft_code,  
demo-# count( tf.ticket_no ) AS fact_passengers,  
demo-# ( SELECT count( s.seat_no )  
demo-# FROM seats s  
demo-# WHERE s.aircraft_code = f.aircraft_code  
demo-# ) AS total_seats  
demo-# FROM flights_v f  
demo-# JOIN ticket_flights tf ON f.flight_id = tf.flight_id  
demo-# WHERE f.status = 'Arrived'  
demo-# GROUP BY 1, 2, 3, 4, 5;  
flight_id | flight_no | departure_city | arrival_city | aircraft_code | fact_passengers | total_seats  
-----  
1 | PG0405 | Москва | Санкт-Петербург | 321 | 79 | 170  
2 | PG0404 | Москва | Санкт-Петербург | 321 | 101 | 170  
3 | PG0405 | Москва | Санкт-Петербург | 321 | 97 | 170  
17 | PG0404 | Москва | Санкт-Петербург | 321 | 101 | 170  
18 | PG0405 | Москва | Санкт-Петербург | 321 | 96 | 170  
21 | PG0405 | Москва | Санкт-Петербург | 321 | 85 | 170  
22 | PG0404 | Москва | Санкт-Петербург | 321 | 1 | 170  
25 | PG0404 | Москва | Санкт-Петербург | 321 | 115 | 170  
26 | PG0405 | Москва | Санкт-Петербург | 321 | 90 | 170  
27 | PG0404 | Москва | Санкт-Петербург | 321 | 92 | 170
```

Рисунок 17 – Запрос 1

Данный подзапрос выводит информацию из таблицы `flights_v`, а также подсчитывает количество сидений для каждого `aircraft_code` и количество пассажиров для каждого самолета со статусом 'Arrived'. Таким образом, данный подзапрос отражает степень заполненности каждого прибывшего самолета.

```

demo=# WITH tickets_seats
demo-# AS (
demo-# SELECT f.flight_id,
demo-# f.flight_no,
demo-# f.departure_city,
demo-# f.arrival_city,
demo-# f.aircraft_code,
demo-# count( tf.ticket_no ) AS fact_passengers,
demo-# ( SELECT count( s.seat_no )
demo-# FROM seats s
demo-# WHERE s.aircraft_code = f.aircraft_code
demo-# ) AS total_seats
demo-# FROM flights_v f
demo-# JOIN ticket_flights tf ON f.flight_id = tf.flight_id
demo-# WHERE f.status = 'Arrived'
demo-# GROUP BY 1, 2, 3, 4, 5
demo-# )
demo-# SELECT ts.departure_city,
demo-# ts.arrival_city,
demo-# sum( ts.fact_passengers ) AS sum_pass,
demo-# sum( ts.total_seats ) AS sum_seats,
demo-# round( sum( ts.fact_passengers )::numeric /
demo-# sum( ts.total_seats )::numeric, 2 ) AS frac
demo-# FROM tickets_seats ts
demo-# GROUP BY ts.departure_city, ts.arrival_city
demo-# ORDER BY ts.departure_city;

```

departure_city	arrival_city	sum_pass	sum_seats	frac
Абакан	Москва	466	1044	0.45
Абакан	Новосибирск	217	348	0.62
Абакан	Томск	258	360	0.72
Анадырь	Москва	64	232	0.28
Анадырь	Хабаровск	135	348	0.39
Анапа	Белгород	1961	3007	0.65
Анапа	Москва	2981	4030	0.74
Архангельск	Москва	988	1550	0.64
Архангельск	Нарьян-Мар	925	1450	0.64

Рисунок 18 – Запрос 2

Запрос 2, включающий запрос 1 через общее табличное выражение, вычленяет из таблицы запроса 1 город отправления и город прибытия, т.е. междугородние маршруты, выводит общее количество всех летящих на этом маршруте и общее количество доступных мест, а также высчитывает их отношение. Таким образом, таблица отражает степень загруженности целого маршрута между городами.

```

demo=# WITH tickets_seats
demo-# AS (
demo-# SELECT f.flight_id,
demo-# f.flight_no,
demo-# f.departure_city,
demo-# f.arrival_city,
demo-# f.aircraft_code,
demo-# count(tf.ticket_no) AS fact_passengers,
demo-# (SELECT count(s.seat_no)
demo-# FROM seats s
demo-# WHERE s.aircraft_code = f.aircraft_code AND s.fare_conditions = tf.fare_conditions
demo-# ) AS total_seats,
demo-# tf.fare_conditions
demo-# FROM flights_v f
demo-# JOIN ticket_flights tf ON f.flight_id = tf.flight_id
demo-# WHERE f.status = 'Arrived'
demo-# GROUP BY 1, 2, 3, 4, 5, 8
demo-# )
demo-# SELECT ts.departure_city,
demo-# ts.arrival_city,
demo-# ts.fare_conditions,
demo-# sum(ts.fact_passengers) AS sum_pass,
demo-# sum(ts.total_seats) AS sum_seats,
demo-# round(sum(ts.fact_passengers)::numeric /
demo-# sum(ts.total_seats)::numeric, 2) AS frac
demo-# FROM tickets_seats ts
demo-# GROUP BY ts.departure_city, ts.arrival_city, ts.fare_conditions
demo-# ORDER BY ts.departure_city, ts.arrival_city;

```

departure_city	arrival_city	fare_conditions	sum_pass	sum_seats	frac
Абакан	Москва	Business	73	160	0.46
Абакан	Москва	Economy	393	864	0.45
Абакан	Новосибирск	Economy	217	348	0.62
Абакан	Томск	Economy	258	360	0.72
Анадырь	Москва	Business	10	40	0.25
Анадырь	Москва	Economy	54	192	0.28

Рисунок 19 – Модификация запроса с учетом класса мест