



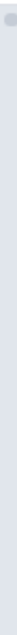


**1 Задание:** «Найти все посадочные места из таблицы посадочных мест, где номер посадочного места равен 15F, отсортировав записи по коду самолета в порядке убывания.»

При поиске всех посадочных мест с номером '15F' используется стандартная запись запроса с определёнными модификациями: ключевое слово **WHERE** задаёт условие поиска; **ORDER BY** сортирует поисковую выдачу по конкретному столбцу; а **DESC** указывает на то, что сортировка производится в убывающем порядке.

```
1 SELECT * FROM boarding_passes WHERE seat_no = '15F' ORDER BY flight_id DESC
```







Data Output Explain Messages Notifications Query History

	 ticket_no [PK] character (13)	 flight_id [PK] integer	 boarding_no integer	 seat_no character varying (4)	
1	0005434625974	32933	12	15F	
2	0005434625832	32923	19	15F	
3	0005435907540	32921	10	15F	
4	0005435907420	32920	7	15F	
5	0005434625908	32910	4	15F	
6	0005435907428	32899	4	15F	
7	0005434625821	32882	11	15F	
8	0005435907449	32881	8	15F	
9	0005434191276	32870	13	15F	
10	0005433533317	32455	28	15F	
11	0005433533250	32448	19	15F	
12	0005433284986	32098	20	15F	
13	0005435103840	32092	28	15F	
14	0005433285345	32087	23	15F	

**2 Задание:** «Найти все вылеты из таблицы вылетов, где статус не arrived и не scheduled.»

При поиске всех вылетов, статусы которых не равны arrived и scheduled в области действия ключевого слова **WHERE** используется **AND**, которое позволяет объединить 2 условия.

```
1 SELECT * FROM flights WHERE status != 'Scheduled' AND status != 'Arrived'
```

	Data Output	Explain	Messages	Notifications	Query History		
	<b>scheduled_departure</b> timestamp with time zone 		<b>scheduled_arrival</b> timestamp with time zone 	<b>departure_airport</b> character (3) 	<b>arrival_airport</b> character (3) 	<b>status</b> character varying (20) 	<b>air ch</b>
	2017-09-14 09:25:00+00		2017-09-14 11:45:00+00	SCW	NBC	Cancelled	CN
	2017-08-16 06:35:00+00		2017-08-16 07:30:00+00	DME	LED	On Time	32
	2017-08-16 08:25:00+00		2017-08-16 09:20:00+00	DME	LED	Delayed	32
	2017-08-16 09:25:00+00		2017-08-16 10:20:00+00	DME	LED	On Time	32
	2017-08-15 16:05:00+00		2017-08-15 17:00:00+00	DME	LED	On Time	32
	2017-08-16 08:05:00+00		2017-08-16 11:30:00+00	DME	OVB	On Time	77
	2017-08-16 07:40:00+00		2017-08-16 08:35:00+00	DME	KZN	On Time	32
	2017-08-16 06:50:00+00		2017-08-16 08:50:00+00	DME	CEK	Delayed	SU
	2017-08-16 11:10:00+00		2017-08-16 12:15:00+00	DME	KUF	On Time	76
	2017-08-15 15:20:00+00		2017-08-15 16:35:00+00	DME	ROV	Delayed	32
	2017-08-15 16:20:00+00		2017-08-15 17:35:00+00	DME	ROV	On Time	32
	2017-09-14 16:20:00+00		2017-09-14 16:55:00+00	DME	VOZ	Cancelled	CR
	2017-08-15 16:20:00+00		2017-08-15 16:55:00+00	DME	VOZ	On Time	CR

**3 Задание:** «Выведите количество самолетов из таблицы самолетов, у которых дистанция полета между 3000 и 5650.»

Для подсчёта самолётов с заданной дистанцией полёта используется функция **COUNT**, считающая число строк с заданным ограничением.

```
1 SELECT COUNT(*) FROM aircrafts_data WHERE range >= 3000 AND range <= 5650
```

Data Output Explain Messages Notifications Query History

	count bigint	
1	3	

**4 Задание:** «Выведите уникальные значения поля «часовой пояс» из таблицы аэропортов, отсортировав их по полю «часовой пояс» в порядке возрастания, ограничив данные первыми 5 записями.»

Для поиска всех часовых поясов используется ключевое слова **DISTINCT**, которое указывает в запросе на то, что их значения не должны повторяться. При сортировке выдачи запроса **ASC** указывает, что вывод идёт в порядке возрастания, а **LIMIT** ограничивает поисковую выдачу первыми 5-ю вариантами.

```
1 SELECT DISTINCT timezone FROM airports_data ORDER BY timezone ASC LIMIT 5
```

Data Output Explain Messages Notifications Query History

	timezone text	
1	Asia/Anadyr	
2	Asia/Chita	
3	Asia/Irkutsk	
4	Asia/Kamchatka	
5	Asia/Krasnoyarsk	

**5 Задание:** «Для выполнения запроса используйте JOIN. Найдите все коды самолета из таблицы самолетов, все номера мест из таблицы мест, где код самолета равен 320.»

При выборе кодов самолётов и номеров посадочных мест используется их запись через точку относительно таблицы с данными. **LEFT JOIN** используется чтобы выбрать все записи из таблицы с самолётами и соответствующие записи из таблицы с посадочными местами. Изначально в таблице отображаются все посадочные места в самолёте с кодом 320.

1

SELECT aircrafts\_data.aircraft\_code, seats.seat\_no

2

FROM aircrafts\_data

3

LEFT JOIN seats ON aircrafts\_data.aircraft\_code = seats.aircraft\_code AND seats.aircraft\_code = '320'

Data Output

Explain

Messages

Notifications

Query History

	aircraft_code character (3)	seat_no character varying (4)
1	320	1A
2	320	1C
3	320	1D
4	320	1F
5	320	2A
6	320	2C
7	320	2D
8	320	2F
9	320	3A
10	320	3C
11	320	3D
12	320	3F
13	320	4A
14	320	4C
15	320	4D
16	320	4F
17	320	5A
18	320	5C
19	320	5D
20	320	5F

Затем отображаются остальные коды самолётов из таблицы с самолётами, а в столбце с посадочными местами напротив этих кодов стоит **null**.

```
1 SELECT aircrafts_data.aircraft_code, seats.seat_no
2 FROM aircrafts_data
3 LEFT JOIN seats ON aircrafts_data.aircraft_code = seats.aircraft_code AND seats.aircraft_code = '320'
```

	Data Output	Explain	Messages	Notifications	Query History
	aircraft_code character (3)	seat_no character varying (4)			
129	320	24A			
130	320	24B			
131	320	24C			
132	320	24D			
133	320	24E			
134	320	24F			
135	320	25A			
136	320	25B			
137	320	25C			
138	320	25D			
139	320	25E			
140	320	25F			
141	SU9	[null]			
142	763	[null]			
143	733	[null]			
144	CN1	[null]			
145	CR2	[null]			
146	773	[null]			
147	319	[null]			
148	321	[null]			

**6 Задание:** «Выведите среднюю стоимость билета бизнес-класса из таблицы ticket\_flights.»

Чтобы вывести среднюю стоимость билета бизнес-класса используется функция **AVG**, которая считает среднее арифметическое всех ячеек указанного столбца. Данная функция передаётся в функция **ROUND**, которая округляет полученное значение.

```
1 SELECT ROUND(AVG(amount)) FROM ticket_flights WHERE fare_conditions = 'Business'
```

Data Output			Explain	Messages	Notifications	Query History
	round numeric					
1	51143					

**7 Задание:** «Выведите часовые пояса и общее количество аэропортов в этом часовом поясе, при этом количество аэропортов в часовом поясе должно быть больше 3, сгруппируйте по часовому поясу.»

Для подсчёта общего количества аэропортов в часовом поясе используется функция **COUNT**, считающая число строк по группирующему значению. Так как **COUNT** не может использоваться в паре с **WHERE**, для вывода условия применяется оператор **HAVING**.

```
1 SELECT timezone, COUNT(*) FROM airports GROUP BY timezone HAVING COUNT(*) > 3
```

Data Output Explain Messages Notifications Query History

	timezone text	count bigint
1	Europe/Samara	5
2	Asia/Krasnoyarsk	8
3	Asia/Irkutsk	5
4	Asia/Yakutsk	5
5	Europe/Moscow	44
6	Asia/Yekaterinburg	22



## 8 Задание: «Напишите произвольный запрос с использованием функции MIN.»

Написан запрос для поиска минимальной стоимости заказа в определённой дате.

```
1 SELECT book_date, MIN(total_amount) FROM bookings GROUP BY book_date
```

Data Output Explain Messages Notifications Query History

	book_date timestamp with time zone	min numeric
1	2017-06-21 11:05:00+00	52000.00
2	2017-06-21 12:48:00+00	211100.00
3	2017-06-21 13:17:00+00	178800.00
4	2017-06-21 22:29:00+00	57500.00
5	2017-06-21 22:52:00+00	123000.00
6	2017-06-22 01:39:00+00	41600.00
7	2017-06-22 02:46:00+00	70100.00
8	2017-06-22 04:22:00+00	46300.00
9	2017-06-22 07:03:00+00	224300.00
10	2017-06-22 19:25:00+00	212500.00
11	2017-06-22 22:05:00+00	28000.00
12	2017-06-22 23:31:00+00	44000.00
13	2017-06-22 23:35:00+00	111800.00
14	2017-06-23 00:49:00+00	32700.00
15	2017-06-23 02:17:00+00	64700.00
16	2017-06-23 05:31:00+00	12600.00
17	2017-06-23 05:33:00+00	68800.00
18	2017-06-23 06:01:00+00	49800.00
19	2017-06-23 06:07:00+00	124200.00
20	2017-06-23 07:21:00+00	222200.00