

## База данных возможных вопросов и заданий на собеседовании по SQL

База периодически обновляется!

46 задач – 04.07.2022

Перечень возможных вопросов (и ответов) собирался на просторах интернета: YouTube, курсы, форумы. Удачи в подготовке!

### Задача 1

Строчки с какими ID из таблицы ниже вернет следующий запрос:

```
SELECT ID FROM Table1 WHERE CLASS_ITEM <> 'A'
```

ID	NAME	CLASS_ITEM
1	Нина	A
2	иван	Null
3	Евгений	B
4	евген	NULL
5	коля	A
6	Инга	C

**Ответ:** <> означает 'не равно'. Вернутся строки 3 и 6.

\* null – при сравнении с null всегда ответ 'нет'

### Задача 2

Выберите верные утверждения (возможно несколько вариантов):

- 1) NULL <> 1
- 2) NULL <> NULL
- 3) NULL = NULL
- 4) NULL IS NOT NULL
- 5) NULL IS NULL

**Ответ:** только значений 5.

\* 1) null – это неопределенность. Вдруг там все-таки единица? Поэтому выражение не верно. 2) разные неопределенности. Далее по аналогии.

### Задача 3

Строчки с какими ID из таблицы ниже вернет следующий запрос:

```
SELECT ID FROM Table1 WHERE NAME LIKE 'И%'
```

ID	NAME	CLASS_ITEM
1	Нина	A
2	иван	Null
3	Евгений	B
4	евген	NULL
5	коля	A
6	Инга	C

**Ответ:** только 6 строка, т.к. важен регистр.

\* Комментарий от себя – возможно таблица не чувствительна к регистру. Как проверить?

Просто:

```
select case when 'A' = 'a' then 'Не чувствительна'
else 'Чувствительна'
end
```

\* Взято из первого файла – Шпаргалки.

#### Задача 4

Строчки с какими ID из таблицы ниже вернет следующий запрос:

```
SELECT ID FROM Table1 WHERE upper(NAME) LIKE 'И%'
```

ID	NAME	CLASS_ITEM
1	Нина	A
2	иван	Null
3	Евгений	B
4	евген	NULL
5	коля	A
6	Инга	C

**Ответ:** строки 2 и 6, т.к. мы в начале приводим их к верхнему регистру.

#### Задача 5

Имеется две таблицы T1 и T2.

Известно, что в таблице T1 – 3 строчки, а в таблице T2 – 4 строчки.

Не зная какие цифры лежат в графах NOM обеих таблиц, необходимо предположить какое минимальное и какое максимальное количество строчек может вернуть запрос ниже:

```
select * from table1 t1 left join table2 t2 on t1.nom=t2.nom
```

table1	table2									
<table><tr><th>NOM</th></tr><tr><td>???</td></tr><tr><td>???</td></tr><tr><td>???</td></tr></table>	NOM	???	???	???	<table><tr><th>NOM</th></tr><tr><td>???</td></tr><tr><td>???</td></tr><tr><td>???</td></tr><tr><td>???</td></tr></table>	NOM	???	???	???	???
NOM										
???										
???										
???										
NOM										
???										
???										
???										
???										

**Ответ:** минимум – 3 строчки, так как left join всегда возвращает все значения первой (левой) таблицы; максимум –  $3 \times 4 = 12$ .

### Задача 6

Что делает UNION и в чем разница между UNION и UNION ALL?

**Ответ:** нужны, чтобы объединять наборы данных, получаемые с помощью операции select. UNION возвращает оригинальные значения, т.е. если строка из второй таблицы идентична строке из первой – эта строка из второй таблицы не будет добавлена. Union all вернет два дубля.

### Задача 7

Перечислите типы соединения данных JOIN, поддерживаемые в ANSI стандарте.

**Ответ:** inner join, left join, right join + 2 дополнительных full join и cross join (подробнее в файле-шпаргалке).

### Задача 8

Дано 2 таблицы – runners и races.

runners

id	name
1	John Doe
2	Jane Doe
3	Alice Jones
4	Bobby Louis

races

id	event	winner_id
1	100 meter dash	2
2	500 meter dash	3
3	Cross-country	2
4	swimming	NULL

5	Lisa Romero
---	-------------

Каков будет результат следующего запроса:

```
Select * from runners where id not in (select winner_id from races)
```

Объясните результат и приведите альтернативный способ, избегающий проблемы.

**Ответ:** тут подвох с null, который при сравнении возвращает отрицательный ответ, поэтому данный запрос ничего не вернет. Правильная версия запроса:

```
Select * from runners where id not in (select winner_id from races where winner_id is not null)
```

\* с помощью параметра ANSI\_NULLS субд можно изменить варианты сравнения.

### Задача 9

Созданы 2 таблицы со следующими значениями:

```
create table dbo.envelope(id int, user_id int);  
create table dbo.docs(idnum int, pageseq int, doctext varchar(100));
```

```
insert into dbo.envelope values
```

```
(1,1),  
(2,2),  
(3,3);
```

```
Insert into dbo.docs(idnum,pageseq) values
```

```
(1,5),  
(2,6),  
(null,0);
```

Какой будет результат при выполнении запроса:

```
update docs set doctext=pageseq from docs inner join envelope on envelope.id=docs.idnum  
where exists (  
    select 1 from dbo.docs  
    where id = envelope.id  
);
```

**Ответ:** во-первых в подзапросе where exists допустили ошибку, так как в dbo.docs нет столбца id, поэтому будет найден внешний id в таблице envelope, а envelope.id всегда равен envelope.id, соответственно это выражение будет выполняться всегда. Столбец doctext будет проапдейчен для первых двух строчек. Итоговая таблица имеет вид:

idnum	pageseq	doctext
1	5	5

2	6	6
null	0	null

### Задача 10

Представьте, что есть две таблицы со следующими схемами данных: Emp (Id, Name, DeptId), Dept(Id, Name).

Если в первой таблице 10 строк, а во второй 5 строк – как много строк будет отображено в качестве результата выполнения запроса:

```
select * from Emp, Dept
```

**Ответ:** эта команда является аналогом команды cross join (декартово произведение). Следовательно,  $10 * 5 = 50$ .

### Задача 11

Даны 2 таблицы, созданные следующими командами:

```
create table test_a(id numeric);  
create table test_b(id numeric);
```

```
insert into test_a(id) values  
    (10),  
    (20),  
    (30),  
    (40),  
    (50);  
insert into test_b(id) values  
    (10),  
    (30),  
    (50);
```

Необходимо написать запрос, который бы выбрал все строчки, которые есть в test\_a, но нет в test\_b, не используя команду not.

**Ответ:** Решение для SQL Server, PostgreSQL и SQLite – использование except:

```
select * from test_a  
except  
select * from test_b;
```

В Oracle вместо except используется слово minus.

Решение для MySQL (не поддерживает except):

```
select a.id  
from test_a a  
left join test_b b on a.id=b.id  
where b.id is null;
```

### Задача 12

Напишите SQL-запрос, который находит 10-ю наивысшую зарплату из таблицы Employee.

**Ответ:**

Вариант решения 1:

```
Select top 1 salary from  
(  
    select distinct top 10 salary from employee order by salary desc  
) as emp order by salary;
```

Вариант решения 2 (т.к. MySQL и Postres не поддерживают top):

```
select salary from  
(  
    select distinct salary from employee order by salary desc limit 10  
) as emp order by salary limit 1;
```

Вариант 3 (короткий, для MySQL):

```
select distinct salary from employee order by salary desc limit 9,1;
```

Вариант 4 (короткий для Postgres):

```
select distinct salary from employee order by salary desc limit 1 offset 9;
```

### Задача 13

Напишите SQL-запрос, использующий UNION ALL, который использует условие WHERE для устранения дубликатов. В каких случаях такой запрос может понадобиться?

**Ответ:** можно избежать дубликатов при использовании UNION ALL (использует меньше мощностей, чем при использовании union distinct). Запрос:

```
select * from mytable where a=X  
union all  
select * from mytable where b=Y and a!= X
```

### Задача 14

Даны следующие таблицы.

```
select * from users;
```

user_id	Username
1	Jonh Doe
2	Jane Don
3	Alice Jones
4	Lisa Romero

```
select * from training_details;
```

user_training_id	user_id	training_id	training_date
1	1	1	"2015-08-02"
2	2	1	"2015-08-03"
3	3	2	"2015-08-02"
4	4	2	"2015-08-04"
5	2	2	"2015-08-03"
6	1	1	"2015-08-02"
7	3	2	"2015-08-04"
8	4	3	"2015-08-03"
9	1	4	"2015-08-03"
10	3	1	"2015-08-02"
11	4	2	"2015-08-04"
12	3	2	"2015-08-02"
13	1	1	"2015-08-02"
14	4	3	"2015-08-03"

Напишите запрос, который получает список юзеров, кто тренировался больше одного раза в один и тот же день, сгруппированный по юзерам и тренировкам (training\_lesson), каждый в порядке от самой близкой до дальней даты.

**Ответ:**

```
select u.user_id, username, training_id, training_date, count(user_training_id) as count
from users u join training_details t on t.user_id=u.user_id
group by u.user_id, username, training_id, training_date
having count(user_training_id) > 1
order by training_date desc;
```

Результат:

user_id	username	training_id	training_date	count
4	Lisa Romero	2	2015-08-04	2
4	Lisa Romero	3	2015-08-03	2
1	John Doe	1	2015-08-02	3
3	Alice Jones	2	2015-08-02	2

### Задача 15

Что такое execution plan? Когда его стоит использовать?

**Ответ:** Execution plan – это, по сути, дорожная карт, которая графически или в текстовом виде отражает методы получения данных. Полезно для того, чтобы помочь разработчику понять и проанализировать характеристики производительности запроса или хранимой процедуры, поскольку план используется для выполнения запроса или хранимой процедуры.

В многих структурах SQL текстовый план выполнения можно получить с помощью такого ключевого слова, как EXPLAIN, также часто можно получить визуальные представления. В MySQL Server у анализатора запросов есть опция «Показать план выполнения».

### Задача 16

Перечислите и объясните каждое из свойств ACID, которые в совокупности гарантируют надежную обработку транзакций базы данных.

**Ответ:** (перевод)

ACID (Atomicity, Consistency, Isolation, Durability) - это набор свойств, гарантирующих надежную обработку транзакций базы данных. Они определяются следующим образом:

**Атомарность.** Атомарность требует, чтобы каждая транзакция выполнялась по принципу "все или ничего": если одна часть транзакции не выполняется, то выполняется вся транзакция, а состояние базы данных остается неизменным. Атомарная система должна гарантировать атомарность в любой ситуации, включая сбои питания, ошибки и сбои.

**Согласованность.** Свойство согласованности гарантирует, что любая транзакция переведет базу данных из одного корректного состояния в другое. Любые данные, записанные в базу данных, должны быть действительными в соответствии со всеми определенными правилами, включая ограничения, каскады, триггеры и любые их комбинации.

**Изоляция.** Свойство изоляции гарантирует, что одновременное выполнение транзакций приводит к такому состоянию системы, которое было бы получено, если бы транзакции выполнялись последовательно, т.е. одна за другой. Обеспечение изоляции является



основной целью управления параллелизмом. В зависимости от метода управления параллелизмом (т.е. если он использует строгую - в отличие от расслабленной - сериализуемости), последствия незавершенной транзакции могут быть даже не видны другой транзакции.

Долговечность. Долговечность означает, что после фиксации транзакции она будет оставаться таковой даже в случае потери питания, сбоев или ошибок. В реляционной базе данных, например, после выполнения группы операторов SQL результаты должны храниться постоянно (даже если сразу после этого произойдет сбой базы данных). Чтобы защититься от потери питания, транзакции (или их последствия) должны быть записаны в энергонезависимую память.

### Задача 17

У вас есть таблица `dbo.users`, в которой столбец `user_id` является уникальным числовым идентификатором. Какой запрос позволит выбрать первые 100 нечетных значений `user_id` из таблицы?

**Ответ:**

```
select top 100 user_id from users where user_id % 2 = 1 order by user_id
```

### Задача 18

Что означают функции `NVL` и `NVL2` в SQL? Чем они отличаются?

**Ответ:** (перевод) Обе функции `NVL(expr1, expr2)` и `NVL2(expr1, expr2, expr3)` проверяют значение `expr1` на то, не является ли оно нулевым.

В функции `NVL(expr1, expr2)`, если `expr1` не является нулевым, возвращается значение `expr1`, в противном случае возвращается значение `expr2`, но с тем же типом данных, что и у `expr1`.

С помощью функции `NVL2(expr1, expr2, expr3)`, если `expr1` не равно `null`, то возвращается значение `expr2`; в противном случае возвращается значение `expr3`.

### Задача 19

Как выбрать все записи с четными номерами из таблицы? Все записи с нечетными номерами?

**Ответ:** четные номера:

```
Select * from table where id % 2 = 0
```

Нечетные номера:

```
Select * from table where id % 2 != 0
```

### Задача 20

Какова разница между `rank()` и `dense_rank()`?

**Ответ:** (перевод) Единственное различие между функциями `RANK()` и `DENSE_RANK()` заключается в случаях, когда существует "ничья", т.е. когда несколько значений в наборе имеют одинаковый рейтинг. В таких случаях функция `RANK()` присваивает значениям в наборе непоследовательные "ранги" (что приводит к появлению пробелов между целочисленными значениями рангов при равенстве), тогда как функция `DENSE_RANK()` присваивает значениям в наборе последовательные ранги (поэтому в случае равенства не будет пробелов между целочисленными значениями рангов).

Например, рассмотрим множество {25, 25, 50, 75, 75, 100}. Для такого набора `RANK()` вернет {1, 1, 3, 4, 4, 6} (обратите внимание, что значения 2 и 5 пропускаются), в то время как `DENSE_RANK()` вернет {1,1,2,3,3,4}.

### Задача 21

Какова разница между `where` и `having`?

**Ответ:** обычно они эквивалентны. Но если используется `group by`:

- `WHERE` используется для фильтрации записей из результата. Фильтрация происходит до создания каких-либо группировок;
- `HAVING` используется для фильтрации значений из группы (т.е. для проверки условий после объединения в группы).

### Задача 22

У вас есть таблица `Employee` с колонками `empName` и `empId`. Каков будет результат запроса:  
`select empName from Employee order by 2 DESC;`

**Ответ:** "Order by 2" действует только в том случае, если в операторе `select` используется как минимум два столбца. Однако в данном запросе, несмотря на то, что таблица `Employee` имеет 2 столбца, запрос выбирает только имя одного столбца, поэтому "Order by 2" приведет к ошибке при выполнении приведенного выше `sql`-запроса.

### Задача 23

Какой будет результат запроса, написанного ниже:

```
Begin TRAN
TRUNCATE TABLE Employees
```

## ROLLBACK

Select \* from Employees

**Ответ:** Этот запрос вернет 10 записей, поскольку в транзакции была выполнена команда TRUNCATE. TRUNCATE сам по себе не ведет журнал, но BEGIN TRANSACTION отслеживает команду TRUNCATE.

### Задача 24

1. В чем разница между однорядными и многорядными функциями?
2. Для чего используется group by?

**Ответ:** 1) Однорядные функции работают с одной строкой за раз. Многорядные функции работают с данными нескольких строк одновременно. 2) Пункт group by объединяет все записи, имеющие одинаковые значения в определенном поле или любой группе полей.

### Задача 25

Представьте таблицу из одной колонки, в которой каждой строке содержится либо единичное числовое значение [1-9], либо буквенное [a-z,A-Z]. Напишите SQL запрос, который пишет 'Fizz' на месте цифры и 'Buzz' на месте буквы.

**Ответ:**

```
select col,  
case  
    when upper(col) = lower(col)  
then 'Fizz'  
else 'Buzz'  
end as FizzBuzz from my_table
```

### Задача 26

Какова разница между char и varchar2?

**Ответ:** При хранении в базе данных varchar2 использует только выделенное пространство. Например, если у вас есть varchar2(1999) и вы поместили в таблицу 50 байт, то он будет использовать 52 байта. Но при хранении в базе данных char всегда использует максимальную длину и добавляет пробелы. Например, если у вас есть char(1999) и вы поместите 50 байт в таблицу, он будет использовать 2000 байт.

### Задача 27

Напишите SQL запрос, чтобы отобразить слово CAPONE так:

C  
A  
P  
O  
N  
E

Иначе – транспонирование текста.

**Ответ:**

```
Declare @a nvarchar(100)='capone';
Declare @length INT;
Declare @i INT=1;
SET @length=LEN(@a)
while @i<=@length
BEGIN
print(substring(@a,@i,1));
set @i=@i+1;
END
```

В Oracle sql есть более красивое решение:

```
SELECT SUBSTR('CAPONE', LEVEL, 1)
FROM DUAL CONNECT BY LEVEL <= LENGTH('CAPONE');
```

### Задача 28

Можно ли неявно вставить строку для столбца идентичности?

**Ответ:**

```
SET IDENTITY_INSERT TABLE1 ON

INSERT INTO TABLE1 (ID,NAME)
SELECT ID,NAME FROM TEMPTB1

SET IDENTITY_INSERT OFF
```

### Задача 29

Есть таблица:

id	C1	C2	C3
1	Red	Yellow	Blue
2	Null	Red	Green
3	Yellow	Null	Violet

Напишите строки, в которых есть Yellow в колонках c1, c2 или c3 без использования OR.

**Ответ:**

```
SELECT * FROM table
WHERE 'Yellow' IN (C1, C2, C3)
```

### Задача 30

Напишите запрос для вставки/апдейта col2, чтобы значения в col2 были противоположны значениям col1.

Col1	Col2
1	0
0	1
0	1
0	1
1	0
0	1
1	0
1	0

**Ответ:**

```
update table set col2 = case when col1 = 1 then 0 else 1 end
```

Либо если тип numeric

```
update table set col2 = 1 - col1
```

### Задача 31

Как вы можете получить последний id без функции max?

**Ответ:** MySQL:

```
select id from table order by id desc limit 1
```

SQL Server:

```
select top 1 id from table order by id desc
```

### Задача 32

Каков разница между IN и EXISTS?

**Ответ:** (перевод)

IN:

Работает на наборе результатов List

Не работает с подзапросами, приводящими к виртуальным таблицам с несколькими столбцами

Сравнивает каждое значение в списке результатов

Производительность сравнительно низкая для больших наборов результатов подзапроса  
EXISTS:

Работает с виртуальными таблицами

Используется с взаимосвязанными запросами

Завершает сравнение, когда найдено совпадение

Производительность сравнительно БЫСТРАЯ при большом наборе результатов подзапроса

### Задача 33

У Вас есть таблица с 7-мью записями. Колонка содержит обозначения (имена). Теперь клиент хочет вставить запись после значения 7 со значением, начинающимся с 10. Можно ли это сделать и как?

**Ответ:** да, используя dbcc функцию.

```
create table tableA
(id int identity,
 name nvarchar(50)
)
insert into tableA values ('ram')
insert into tableA values ('rahim')
insert into tableA values ('roja')
insert into tableA values ('rahman')
insert into tableA values ('rani')
insert into tableA values ('raja')
insert into tableA values ('raga')
select * From tableA
```

```
DBCC CHECKIDENT(tableA,RESEED,9)
```

```
insert into tableA values ('roli')
```

```
insert into tableA values ('rosy')
```

```
insert into tableA values ('raka')
```

```
insert into tableA values ('rahul')
```

```
insert into tableA values ('rihan')
```

```
insert into tableA values ('bala')
```

```
insert into tableA values ('gala')
```

### Задача 34 (обратное в задаче 38)

Как можно использовать CTE (Common Table Expression - Обобщенное табличное выражение), чтобы вернуть пятое или иное n-значение зарплаты из таблицы?

**Ответ:**

```
Declare @N int
```

```
set @N = 5;
```

```
WITH CTE AS
```

```
(
```

```
    SELECT Name, Salary, EmpID, RN = ROW_NUMBER()
```

```
        OVER (ORDER BY Salary DESC)
```

```
    FROM Employee
```

```
)
```

```
SELECT Name, Salary, EmpID
```

```
FROM CTE
```

```
WHERE RN = @N
```

### Задача 35

Есть таблица из одной колонке. В колонке построчно положительные либо отрицательные число (одно число на строку). Как посчитать отдельно сумму положительных и отрицательных чисел?

**Ответ:**

```
select sum(case when x>0 then x else 0 end)sum_pos,
```

```
sum(case when x<0 then x else 0 end)sum_neg
```

```
from a;
```

### Задача 36

Есть таблица mass\_table

Weight
56.7
34.567
365.253
34

Напишите запрос, который даст следующий вывод:

weight	Kg	Gms
5.67	6	67
34.567	34	567
365.253	365	253
34	34	0

**Ответ:**

```
select weight,  
trunc(weight) as kg,  
nvl(substr(weight - trunc(weight), 2), 0) as gms  
from mass_table;
```

### Задача 37

Представим таблицу Employee:

Emp_Id	Emp_name	Salary	Manager_Id
10	Anil	50000	18
11	Vikas	75000	16
12	Nisha	40000	18
13	Nidhi	60000	17
14	Priya	80000	18
15	Mohit	45000	18
16	Rajesh	90000	—
17	Raman	55000	16



Emp_Id	Emp_name	Salary	Manager_Id
18	Santosh	65000	17

Напишите запрос, результатом которого будет вывод:

Manager_Id	Manager	Average_Salary_Under_Manager
16	Rajesh	65000
17	Raman	62500
18	Santosh	53750

**Ответ:**

```
select b.emp_id as "Manager_Id",  
       b.emp_name as "Manager",  
       avg(a.salary) as "Average_Salary_Under_Manager"  
from Employee a,  
     Employee b  
where a.manager_id = b.emp_id  
group by b.emp_id, b.emp_name  
order by b.emp_id;
```

### Задача 38

Как получить n-значение зарплаты без использования CTE?

**Ответ:**

```
SELECT salary from Employee order by salary DESC LIMIT 2,1
```

Это даст третью по величине зарплату из таблицы Employee. Соответственно, мы можем узнать N-ую зарплату, используя LIMIT (N-1),1

Решения для MS SQL:

```
SELECT salary from Employee order by salary DESC  
OFFSET 2 ROWS  
FETCH NEXT 1 ROW ONLY
```

\* Offset = (n-1)

### Задача 39

Как найти дубликаты?

- с одной записью;
- с несколькими записями.

**Ответ:**

- с одной записью

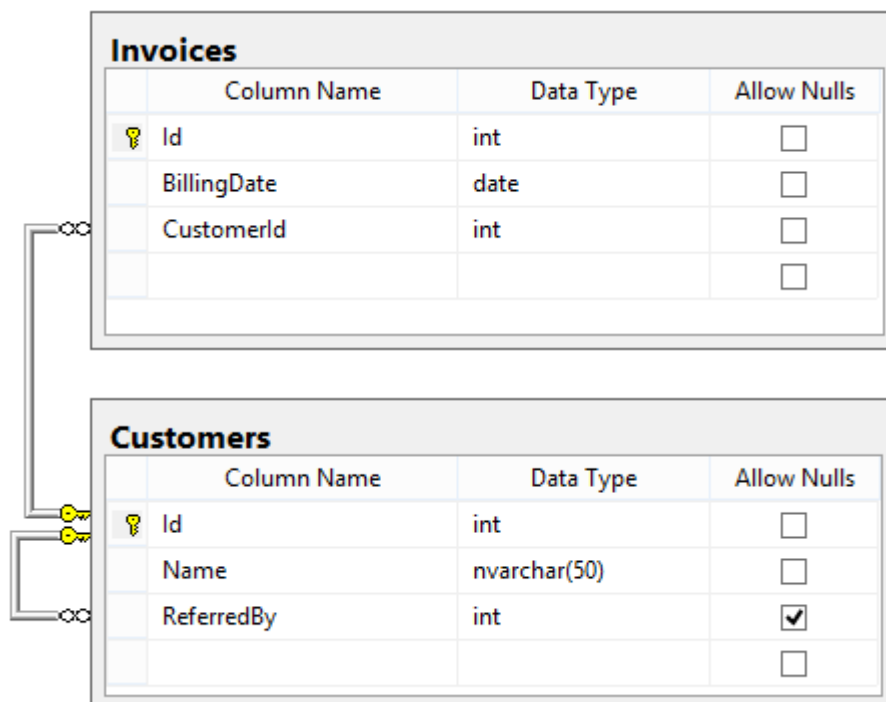
```
SELECT name, COUNT(email)
FROM users
GROUP BY email
HAVING COUNT(email) > 1
```

- с несколькими записями

```
SELECT name, email, COUNT(*)
FROM users
GROUP BY name, email
HAVING COUNT(*) > 1
```

#### Задача 40

Учитывая схему базы данных, показанную на приведенной ниже диаграмме в стиле SQLServer, напишите SQL-запрос для возврата списка всех счетов-фактур. Для каждого счета-фактуры укажите идентификатор счета-фактуры, дату выставления счета, имя клиента и имя клиента, который направил этого клиента (если есть). Список должен быть упорядочен по дате выставления счета.



**Ответ:**

```
SELECT i.Id, i.BillingDate, c.Name, r.Name AS ReferredByName
FROM Invoices i
JOIN Customers c ON i.CustomerId = c.Id
LEFT JOIN Customers r ON c.ReferredBy = r.Id
ORDER BY i.BillingDate;
```

Обратите внимание, что этот запрос не возвращает счета-фактуры, которые не имеют связанного с ними Заказчика. Возможно, это правильное поведение для большинства случаев (например, гарантируется, что каждый счет-фактура связан с заказчиком, или несопоставленные счета-фактуры не представляют интереса). Однако, чтобы гарантировать, что все счета-фактуры будут возвращены, несмотря ни на что, таблицу Invoices следует объединить с Customers с помощью LEFT JOIN:

```
SELECT i.Id, i.BillingDate, c.Name, r.Name AS ReferredByName
FROM Invoices i
LEFT JOIN Customers c ON i.CustomerId = c.Id
LEFT JOIN Customers r ON c.ReferredBy = r.Id
ORDER BY i.BillingDate;
```

#### Задача 41

Дана таблица Телевизионных каналов и таблица Опроса зрителей:

channels

id_channel	name_channel
1	Первый канал
2	Россия 1
3	СТС
4	Перец
5	Домашний
6	4 канал
7	Рен-ТВ

opros

date_opros	nom_operator	channels
11.01.2015 19:35	2	1,4,3
11.01.2015 19:41	4	2,3,6
12.01.2015 20:00	1	1,2,4
13.01.2015 17:10	3	6,3

14.01.2015 15:33	2	4,1
14.01.2015 19:33	1	1,2,5
15.01.2015 18:44	1	1,4,5

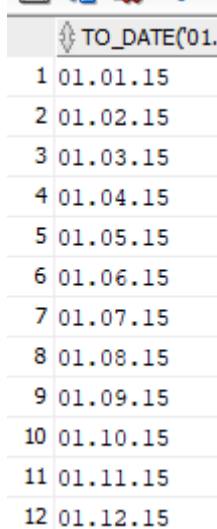
Необходимо вывести количество голосов зрителей по каждому каналу за каждый месяц с начала года накопительным итогом. В таблицу результата опроса данные заносились разными операторами. При опросе зритель мог проголосовать не более чем за три канала. Результат заносился в таблицу через запятую.

**Ответ:**

## СПОСОБ 1

1 шаг. Генерация таблицы с датами

```
select to_date('01.' || lpad(level,2,'0') || '.2015','dd.mm.yyyy')
from dual
connect by level <=12
```



	TO_DATE('01.
1	01.01.15
2	01.02.15
3	01.03.15
4	01.04.15
5	01.05.15
6	01.06.15
7	01.07.15
8	01.08.15
9	01.09.15
10	01.10.15
11	01.11.15
12	01.12.15

2 шаг.

```
select to_char(m.MM, 'Month yyyy') nm_month, c.name_channel, count(o.channels) cnt
from (
select to_date('01.' || lpad(level,2,'0') || '.2015','dd.mm.yyyy') mm
from dual
connect by level <=12) m
cross join channel c
left join opros_op o
on o.date_opros < last_day(m.MM)+1
and ',' || o.channels || ',' like '%,' || c.id_channel || ',%'
group by m.MM, c.name_channel, c.id_channel
```

```
order by m.MM, c.id_channel
```

## СПОСОБ 2

```
select to_char(t.MM, 'Month yyyy') MM,  
       t.name_channel,  
       sum(cnt) over (partition by name_channel order by MM rows between  
                     unbounded preceding and current row) cnt_all  
from (  
select trunc(o.date_opros, 'MM') MM,  
       c.name_channel,  
       c.id_channel,  
       count(*) cnt  
from opros_op o  
join channel c  
  on ',' || o.channels || ',' like '%,' || c.id_channel || ',%'  
group by trunc(o.date_opros, 'MM'), c.name_channel, c.id_channel  
order by 1,3) t  
order by t.MM, t.id_channel
```

## Задача 42

На основе данных опроса (таблицы в задании 41) – вывести информацию о среднем и максимальном количестве проводимых опросов менеджером в день.

**Ответ:**

```
select to_char(trunc(t.dd, 'mm'), 'Month yyyy') mm,  
       round(avg(cnt),2) AVG_OPROS,  
       max(cnt) MAX_OPROS  
from (  
select o.nom_operator, trunc(o.date_opros) dd, count(1) cnt  
from opros_op o  
group by o.nom_operator, trunc(o.date_opros)) t  
group by trunc(t.dd, 'mm')
```

## Задача 43

Чем отличается left join от join и в каком случае их лучше использовать?

**Ответ:**

По смыслу одинаковы (пересечения множеств). В случае join (он же inner join) остаются только комбинации общих для обеих таблиц строк; в случае left join остается то же самое, что от join + оставшиеся строки из левой (первой таблицы), которым пар во второй не нашлось.

Стандартная ситуация для использования left join – раскрытие справочников. Слева стоит основная таблица, к ней через left join добавляются справочники.

Еще одна ситуация удобного использования join – поиск дублей, где различается лишь один столбец, например – id.

```
select * from managers_dup t1
join managers_dup t2
on t1.first_name=t2.first_name
and t1.last_name=t2.last_name
where t1.manager_id > t2.manager_id
```

**Задача 44**

Что будет, если написать WHERE id=null?

**Ответ:**

Ответ не будет содержать строк. Можно исправить, написав where id is null.

**Задача 45**

Есть выражение: select id, max(value) from table where id=1. Найдите ошибку

**Ответ:**

Выведется ошибка. Нужно добавить группировку и получится так:

```
select id, max(value) from table where id=1 group by id
```

**Задача 46**

lag(salary) over (partition by month, id) – что выдаст такой запрос в таблице id, month, salary?

**Ответ:**

Выдаст ошибку. Для начала нужно добавить order by. Если просто добавить order by -все равно вернет null значения из-за одновременной группировки по month и id. Чтобы выводить предыдущее значение зарплаты сотрудника, нужно сделать так:

```
Lag(salary) over (partition by id order by month) ...
```