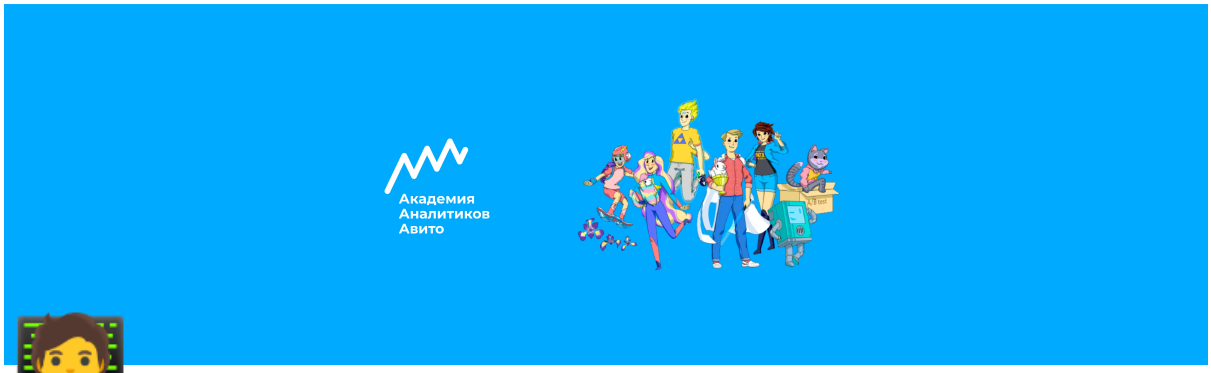


Операции со множествами

Объединение

Для объединения запросов используется UNION ALL.

```
<запрос 1>
UNION [ALL]
<запрос 2>
```

👉Предложение UNION ALL приводит к появлению в результирующем наборе всех строк каждого из запросов.

Можно связывать любое число запросов.

Если параметр ALL указан, сохраняются все дубликаты выходных строк.

Если не указан, то в результирующем наборе присутствуют только уникальные строки.

Операция объединения может быть выполнена только при выполнении следующих условий:

- количество столбцов каждого из запросов должно быть одинаковым

- столбцы каждого из запросов должны быть совместимы между собой по типам данных

📌В результирующем наборе имена столбцов будут называться так, как они заданы в первом запросе.

Предложение ORDER BY применяется к результату объединения, поэтому оно может быть указано только в конце всего составного запроса. Без ORDER BY порядок кортежей в результирующем отношении не определен

Пересечение и разность

В стандарте языка SQL имеются предложения оператора SELECT для выполнения операций пересечения и разности результатов запросов.

INTERSECT [ALL] (пересечение) и **EXCEPT** [ALL] (разность) работают аналогично UNION [ALL].

Пересечение

```
<запрос 1>
INTERSECT [ALL]
<запрос 2>
```

Разность

```
<запрос 1>
EXCEPT [ALL]
<запрос 2>
```

👉INTERSECT - В результирующий набор попадают только те строки, которые присутствуют в обоих запросах.

👉EXCEPT - В результирующий набор попадают только те строки первого запроса, которые отсутствуют во втором.

Операции могут выполнены при условиях аналогичных UNION ALL

- количество столбцов каждого из запросов должно быть одинаковым
- столбцы каждого из запросов должны быть совместимы между собой по типам данных

📌В результирующем наборе имена столбцов будут называться так, как они заданы в первом запросе.

Если не используется ключевое слово ALL, то при выполнении операции автоматически устраняются дубликаты строк.

Для пересечения INTERSECT. Если указано ALL, то количество дублированных строк равно **минимальному** из количества дубликатов в двух таблицах.

Для разности EXCEPT. Если указано ALL, то количество дублированных строк равно **разности** количества дубликатов в двух таблицах.

📌 Для EXCEPT порядок исходных наборов имеет значение, то есть операция разности не является коммутативной в отличии от пересечения или объединения.

💬Пересечение и разность используются не так часто, как UNION ALL. Поэтому не все СУБД поддерживают эти предложения в операторе SELECT. Нет поддержки INTERSECT/EXCEPT, например, в MySQL, а в MS SQL Server она появилась, лишь начиная с версии 2005, и то без ключевого слова ALL. ALL с INTERSECT/EXCEPT также еще не реализована в Oracle. А вместо стандартного EXCEPT в Oracle используется ключевое слово MINUS.

Для выполнения операций пересечения и разности могут быть использованы другие средства. Один и тот же результат можно получить с помощью различных формулировок оператора SELECT. В случае пересечения и разности можно воспользоваться предикатом существования EXISTS или ANTI JOIN, но удаление дубликатов будет произведено по другому.

💻 Пример

Нужно реализовать аналог операции пересечения (без ALL). Использовать таблицы set_1 и set_2.

```
select c from set1
except
select c from set2
```

```
select c
from set1
where not exists (select c from
set2 where set2.c = set1.c)
```

```
select set1.c
from set1
left join set2 on set1.c = set2.c
where set2.c is null
```

Порядок выполнения операторов

1. INTERSECT
2. UNION, EXCEPT

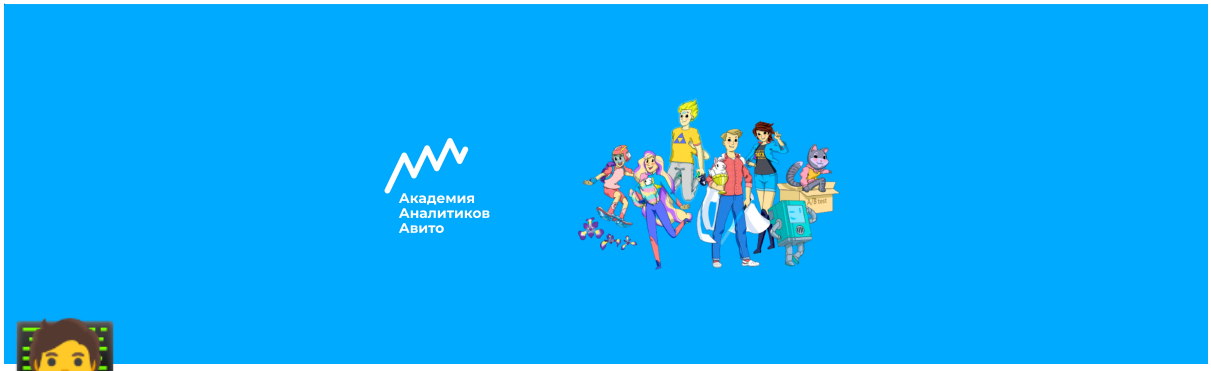
Логический порядок выполнения операций над множествами может варьироваться.

Порядок выполнения можно изменять скобками.

```
SELECT n FROM set_1 WHERE n IN (1, 2)
UNION
SELECT n FROM set_1 WHERE n IN (1)
EXCEPT
SELECT n FROM set_1 WHERE n IN (1)
```

INTERSECT приоритетнее UNION, то есть выполняется раньше.

UNION и EXCEPT равнозначны.



Расширенные возможности GROUP BY

GROUPING SETS: наборы группирования

Группировка по всем полям

Функция grouping

Для самых любознательных

GROUPING SETS: наборы группирования

```
GROUPING SETS ( (<список столбцов>), …)
```

С концепцией наборов группирования возможны более сложные операции группировки, чем позволяет обычный GROUP BY.

1. “список столбцов” – это группа/набор группирования, данные группируются отдельно для каждого заданного набора группирования
2. для каждой группы вычисляются агрегатные функции как для простых предложений GROUP BY
3. в конце все результаты соединяются
4. null в результате трактуется как total по колонке

💻 **Пример**

Исходные данные

	brand	size	sales
1	Foo	L	10
2	Foo	M	20
3	Bar	M	15
4	Bar	L	5

Хотим получить одним запросом суммарное количество продаж в группировке по только по brand, отдельно от нее в группировке по только по size и отдельно общее количество продаж

	brand	size	sum
1	Bar	[NULL]	20
2	Foo	[NULL]	30
3	[NULL]	L	15
4	[NULL]	M	35
5	[NULL]	[NULL]	50

можно использовать GROUPING SETS:

```
SELECT brand, size, sum(sales)
FROM gs_items_sold
GROUP BY GROUPING SETS ((brand), (size), ())
ORDER BY 1, 2
```

	brand	size	sum	
1	Bar	[NULL]	20	группировка по brand
2	Foo	[NULL]	30	grouping set (brand)
3	[NULL]	L	15	группировка по size
4	[NULL]	M	35	grouping set (size)
5	[NULL]	[NULL]	50	тотал grouping set ()

В каждом внутреннем списке GROUPING SETS могут задаваться ноль или более столбцов или выражений, которые воспринимаются так же, как если бы они были непосредственно записаны в предложении GROUP BY.

- Пустой набор группировки означает, что все строки сводятся к одной группе (которая выводится, даже если входных строк нет), как будто используются агрегатные функции без предложения GROUP BY.
- Скобки вокруг колонок в GROUPING SETS, хоть и выделяют явно группы, но не обязательны в случае, если в группе ровно одна колонка.
- Столбцы, которые отсутствуют в группирующих наборах, заменяются в результатах значениями NULL.

В предложении GROUP BY на верхнем уровне выражений запись (a, b) воспринимается как список выражений.

Преимущества GROUPING SETS

Ту же самую задачу можно решить используя UNION ALL

```
select null::varchar as brand, null::varchar as size, sum(sales) from gs_items_sold
select brand,                  null::varchar,         sum(sales) from gs_items_sold group by brand
select null::varchar,          size,                  sum(sales) from gs_items_sold group by size
```

GROUPING SETS работает быстрее, что можно увидеть, если построить план запроса.

План запроса с UNION ALL:

```
                                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
Sort  (cost=639.82..640.82 rows=400 width=96) (actual time=28.694..28.697 rows=18 loops=1)
  Sort Key: co.name, (NULL::character varying)
```

План запроса с GROUPING SETS:

```
                                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
Sort  (cost=410.51..411.51 rows=400 width=96) (actual time=21.228..21.231 rows=18 loops=1)
  Sort Key: co.name, ci.name
```

Это достигается за счет одного прохода по таблице вместо нескольких в случае с UNION ALL.

Группировка по всем полям

Чтобы получить полную сумму (как будто используются агрегаты без группировки), можно использовать пустую группировку.

👉При пустой группировке все поля, участвующие в GROUPING SETS, заполнены NULL.

В наборы группирования можно вставлять номера колонок.

Функция grouping

```
GROUPING (<столбец>, …)
```

GROUPING возвращает битовую маску для колонок, перечисленных в ней. В случае, если колонка одна, эта битовая маска принимает вид нулей и единиц. Единица в маске показывает что колонка не участвовала в группировке (по ней посчитан подитог)

	brand	size	sum	grouping
1	Bar	[NULL]	20	01
2	Foo	[NULL]	30	01
3	[NULL]	L	15	10
4	[NULL]	M	35	10
5	[NULL]	[NULL]	50	11

GROUP BY и GROUPING SETS

В одном запросе можно использовать вместе обычную группировку и группировку с помощью GROUPING SETS.

📌Агрегаты не показываются для того, что не участвует в GROUPING SETS.

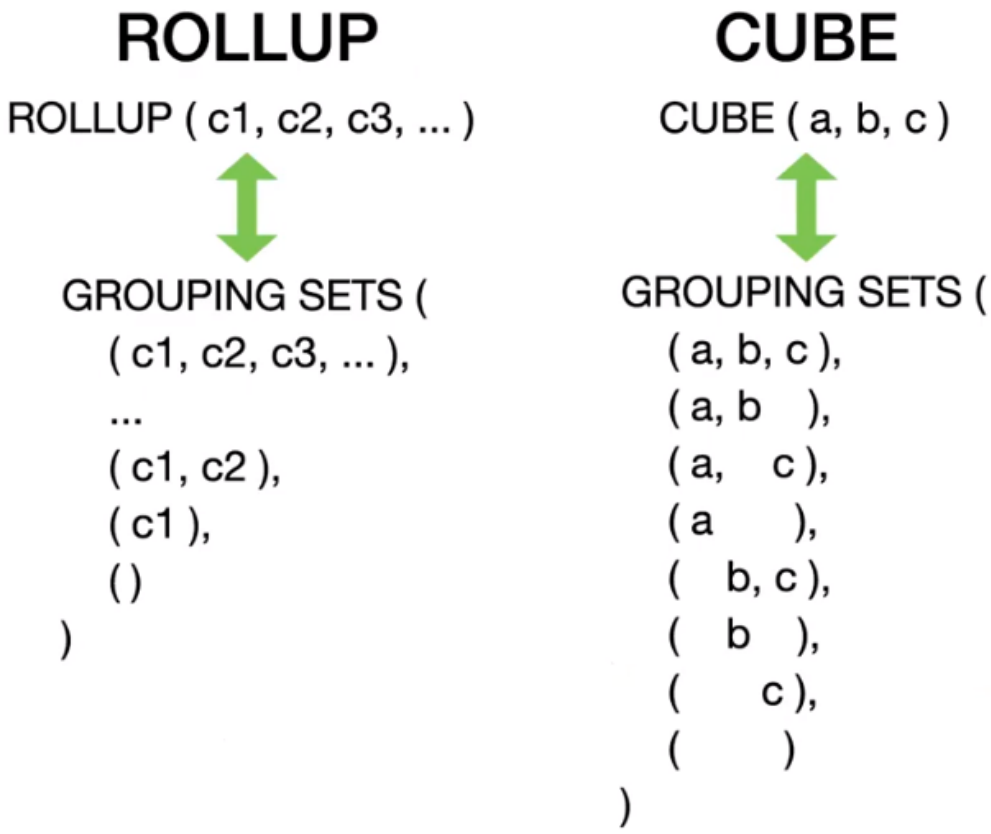
Можно проводить объединение колонок в группы внутри GROUPING SETS.

Два набора группирования используются наиболее часто. Для них предусмотрели короткую запись:

ROLLUP (“свертывает”) - представляет только часть возможных комбинаций частичных сумм в измерениях, составленных по определенному правилу (группа на единицу меньше предыдущей).

CUBE позволяет команде вычислить агрегаты для всех возможных комбинаций групп измерений.

CUBE удобно использовать при многомерном анализе, так как предложение генерирует все частичные суммы, которые могут быть вычислены для куба данных с указанными измерениями.



💻 Пример

Для того, чтобы посчитать количество продаж в группировке по brand и size, потом подсчитать подитог для каждого brand без учета size и в конце подвести общий итог как на картинке:

brand	size	sum	grouping
Bar	M	15	00
Foo	M	20	00
Bar	L	5	00
Foo	L	10	00
Bar		20	01
Foo		30	01
		50	11

можно использовать GROUPING SETS:

```
SELECT
  brand, size, sum(sales),
  grouping(brand, size)::bit(2)
FROM gs_items_sold
GROUP BY GROUPING SETS (
    (brand, size),
    (brand),
    ( )
) ORDER BY 4
```

или ROLLUP:

```
SELECT
  brand, size, sum(sales),
  grouping(brand, size)::bit(2)
FROM gs_items_sold
GROUP BY ROLLUP (
    brand, size)
ORDER BY 4
```

Запросы эквивалентны.

Для самых любознательных

А можно ли получить такой же результат, не используя особых синтаксических конструкций? Только join и generate_series для размножения кортежей исходного отношения.

Практика: Операции со множествами. Группировки с подытогом

Задача 1: Простое объединение

Условие:

Даны 2 простые таблицы с целыми числами:

`d7_set1 (n int)`;

`d7_set2 (n int)`;

Вопрос:

Выведите все различные числа из табличек `d7_set1, d7_set2`.

Решите задачу только с использованием `union all`.

	n
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18

Задача 2: Фамилии новых пользователей

Условие:

Покупатели `d7_buyer(id int, surname text, …)`;

Продавцы `d7_seller(id int, surname text, …)`;

Менеджеры `d7_manager(id int, surname text, …)`;

Регистрации `d7_user(id serial, registration_date date, …)`;

Вопрос:

Выведите все фамилии пользователей без учета роли при условии регистрации после `2020-11-01`.

Сравнение дат работает с неявным приведением типа `registration_date >= '2020-11-01'`

	surname
1	Игнатьев
2	Ковров
3	Кузикин
4	Копориков
5	Потёмкин
6	Алехин
7	Винокуров
8	Савасин
9	Дорогов
10	Игнатенков
11	Евлентьев
12	Кантонистов

Задача 3: Количество активных пользователей

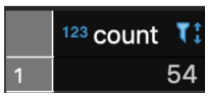
Условие:

Покупатели `d7_buyer(id int, surname text, last_action_date date)`;

Продавцы `d7_seller(id int, surname text, last_action_date date`);

Менеджеры `d7_manager(id int, surname text, last_action_date date)`;

Регистрации `d7_user(id serial, registration_date date, …)`;



	123 count
1	54

Вопрос:

Посчитайте количество пользователей, совершавших действия после `2020-06-01`

Решите задачу с указанием константы с датой только 1 раз

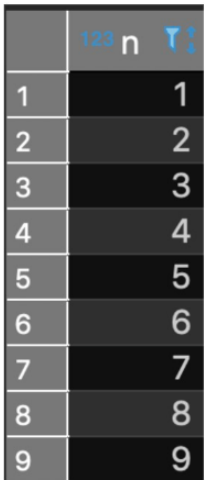
Задача 4: Простая разность

Условие:

Даны 2 простые таблицы с целыми числами

`d7_set1 (n int)`;

`d7_set2 (n int)`;



	123 n
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

Вопрос:

1. Найдите записи из `d7_set1`, которых нет в `d7_set2`.
2. Реализуйте то же самое через `join`, без использования ключевых слов `except` или `minus`

Задача 5: Простое пересечение

Условие:

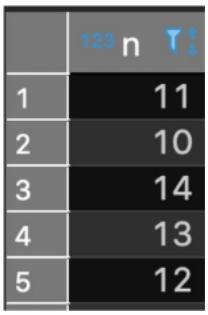
Даны 2 простые таблицы с целыми числами

`d7_set1 (n int`);

`d7_set2 (n int)`;

Вопрос:

1. Найдите записи, которые и в `d7_set1` и в `d7_set2`.
2. Реализуйте то же самое через `join`, без использования `INTERSECT`



	n
1	11
2	10
3	14
4	13
5	12

Задача 6: Исключительные продавцы

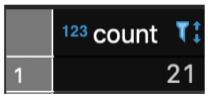
Условие:

Покупатели `d7_buyer(id int, surname text, …)`;

Продавцы d`7_seller(id int, surname text, …)`;

Менеджеры `d7_manager(id int, surname text, …)`;

Регистрации `d7_user(id serial, registration_date date, …)`;



	count
1	21

Вопрос:

Найдите количество продавцов, которые не совершали действие как покупатель после `2020-06-01`

Задача 7: Первый постоянный байер

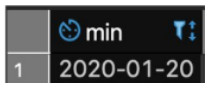
Условие:

Покупатели `d7_buyer(id int, surname text, …)`;

Продавцы `d7_seller(id int, surname text, …`);

Менеджеры `d7_manager(id int, surname text, …)`;

Регистрации `d7_user(id serial, registration_date date, …)`;



	min
1	2020-01-20

Вопрос:

Найдите минимальную дату регистрации покупателя, который не совершал действий, как продавец

Задача 8: Только нужные месяцы

Условие:

Покупатели `d7_buyer(id int, surname text, …)`;

Продавцы `d7_seller(id int, surname text, …)`;

Менеджеры d`7_manager(id int, surname text, …)`;

Регистрации `d7_user(id serial, registration_date date, …)`;

Вопрос:

Найдите месяцы регистрации, в которые не было пользователей, одновременно являющихся продавцами и покупателям. Колонку с месяцем приведите к дате.

Используйте `except` + `intersect` (без `join` и `group by`)

registration_month
2020-01-01
2020-02-01
2020-03-01
2020-05-01
2020-06-01
2020-07-01
2020-08-01
2020-09-01
2020-12-01

Задача 9: Подытог с UNION ALL

Условие:

Лог отеля `d7_hotel(event_date, room_id,floor_id, user_id, paid)`;

Вопрос:

Подсчитать количество уникальных посетителей отеля и уплаченную сумму в разрезе этажей и комнат.

Подвести подытог по этажам и общий

floor	room	sum	count
1	A	1000	1
1	B	1000	1
1	Any	2000	1
2	A	1000	1
2	Any	1000	1
Any	Any	3000	2

Задача 10: Количество пользователей в разрезах

Условие:

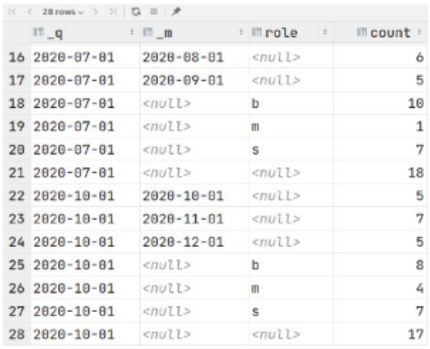
Регистрации `d7_user(id serial, registration_date date, role varchar(1))`;

Вопрос:

Посчитайте ***количество*** уникальных пользователей `(d7_user)` в разрезе *квартала* регистрации и *месяца*.
Подведите подытог по каждому разрезу.

Добавьте разбиение по роли пользователей только в квартальную статистику.

В чем отличие, если бы мы использовали `cube`?



	_q	_m	role	count
16	2020-07-01	2020-08-01	<null>	6
17	2020-07-01	2020-09-01	<null>	5
18	2020-07-01	<null>	b	10
19	2020-07-01	<null>	m	1
20	2020-07-01	<null>	s	7
21	2020-07-01	<null>	<null>	18
22	2020-10-01	2020-10-01	<null>	5
23	2020-10-01	2020-11-01	<null>	7
24	2020-10-01	2020-12-01	<null>	5
25	2020-10-01	<null>	b	8
26	2020-10-01	<null>	m	4
27	2020-10-01	<null>	s	7
28	2020-10-01	<null>	<null>	17

Задача 11*: Rollup через джойн на битовую маску

Условие:

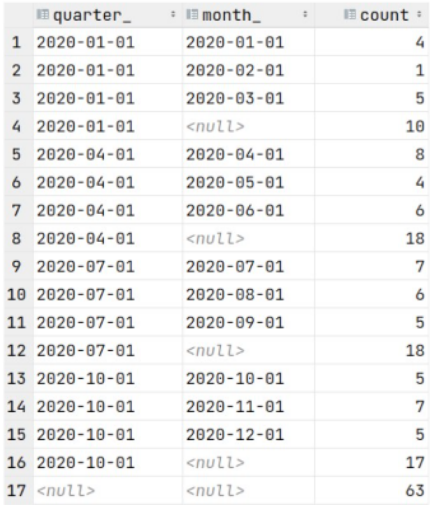
Регистрации `d7_user(id serial, registration_date date, role varchar(1))`;

Вопрос:

Посчитайте ***количество*** уникальных пользователей в разрезе квартала регистрации и месяца.

Подведите подытог по каждому разрезу и общий подытог.

Используйте `join`, а не `rollup`.



	quarter_	month_	count
1	2020-01-01	2020-01-01	4
2	2020-01-01	2020-02-01	1
3	2020-01-01	2020-03-01	5
4	2020-01-01	<null>	10
5	2020-04-01	2020-04-01	8
6	2020-04-01	2020-05-01	4
7	2020-04-01	2020-06-01	6
8	2020-04-01	<null>	18
9	2020-07-01	2020-07-01	7
10	2020-07-01	2020-08-01	6
11	2020-07-01	2020-09-01	5
12	2020-07-01	<null>	18
13	2020-10-01	2020-10-01	5
14	2020-10-01	2020-11-01	7
15	2020-10-01	2020-12-01	5
16	2020-10-01	<null>	17
17	<null>	<null>	63

⚠️ прежде чем открывать ответы, попробуй решить задачи самостоятельно

▼ Ответы

▼ Ответ: задача 1

```
select distinct n from (
    select n from d7_set1
    union all
    select n from d7_set2
) _
order by 1
```

▼ Ответ: задача 2

```
select distinct surname from (
   select id, surname from d7_buyer
   union all
   select id, surname from d7_seller
   union all
   select id, surname from d7_manager
) users
where id in (
   select id
   from d7_user
   where registration_date >= '2020-11-01'
)
```

▼ **Ответ: задача 3**

```
select count(distinct id) from (
    select id, last_action_date from d7_buyer
    union all
    select id, last_action_date from d7_seller
    union all
    select id, last_action_date from d7_manager
) _ where last_action_date >= '2020-06-01'
```

▼ **Ответ: задача 4**

```
select s1.n
from (select distinct n from d7_set1) s1
left join (select distinct n from d7_set2) s2 on s1.n=s2.n
where s2.n is null
order by 1
```

▼ **Ответ: задача 5**

```
select s1.n
from (select distinct n from d7_set1) s1
join (select distinct n from d7_set2) s2 on s1.n=s2.n
order by 1
```

▼ **Ответ: задача 6**

```
select count(*) from (
    select id from d7_seller
    except
    select id from d7_buyer where last_action_date >= '2020-06-01'
) _
```

▼ **Ответ: задача 7**

```
select min(registration_date)
from d7_user where id in (
    select id from d7_buyer
    except
    select id from d7_seller
)
```

▼ **Ответ: задача 8**

```
select date_trunc('month', registration_date)::date
from d7_user
except
select date_trunc('month', registration_date)::date
from d7_user
where id in (
   select id from d7_seller
   intersect
   select id from d7_buyer
)
order by 1
```

▼ **Ответ: задача 9**

```
select floor_id::varchar, room_id::varchar, sum(paid), count(distinct user_id) from d7_hotel group by 1,2
union all
select floor_id::varchar, 'Any', sum(paid), count(distinct user_id) from d7_hotel group by 1,2
union all
select 'Any', 'Any', sum(paid), count(distinct user_id) from d7_hotel group by 1,2
```

▼ **Ответ: задача 10**

```
select date_trunc('quarter', registration_date)::date q_,
       date_trunc('month', registration_date)::date m_,
       role,
       count(distinct id)
from d7_user
group by grouping sets((1,2), (1), (1,3))
order by 1,2,3


cube - сделал бы все возможные группы.
можно отметить, что группа (1,2) - то же самое что и группа (2)
```

▼ **Подсказка: задача 11* rollup через джойн**

нужно сделать join на этот подзапрос

```
select *
from (select 1 as q, 1 as m
     union all
     select 1 as q, 0 as m
     union all
     select 0 as q, 0 as m) _
```

▼ **Ответ: задача 11* rollup через джойн на битовую маску**

```
Можно через подзапрос

with mask as (
    select 1 as q, 1 as m
    union all
    select 1 as q, 0 as m
    union all
    select 0 as q, 0 as m
)
select
    case when q = 1 then date_trunc('quarter', registration_date)::date end,
    case when m = 1 then date_trunc('month', registration_date)::date end,
    count(distinct id)
from d7_user
```

```
cross join mask
group by 1,2
```