При помощи функции ROW_NUMBER сгенерируем порядковый номер строки запроса.

```
masterskaya=# select row_number() over (order by client_name) num, cl
ient_name from client;
 num |    client_name
-----+-------------------
   1 | Danil Nikitovich
   2 | Svetlana Pavlovna
   3 | Veronika Petrova
   4 | Veronika Petrovna
(4 rows)

masterskaya=#
```

Функция ABS(n) возвращает абсолютное значение числа n.

```
masterskaya=# select abs(100) x1, abs(-100) x2, abs(-100.0) x3;
 x1  | x2  |  x3
-----+-----+------
 100 | 100 | 100.0
(1 row)

masterskaya=#
```

Функция CEIL(n) возвращает наименьшее целое, большее или равное переданному в качестве параметра числу n.

```
masterskaya=# select ceil(100) x1, ceil(-100) x2, ceil(-100.6) x3, ce
il(100.54) x4;
 x1  |  x2  |  x3  | x4
-----+------+------+-----
 100 | -100 | -100 | 101
(1 row)

masterskaya=#
```

Функция FLOOR(n) возвращает наибольшее целое, меньшее или равное переданному в качестве параметра числу n

```
masterskaya=# select floor (100.22) x1, floor(-100.77) x2, floor (100
.99) x3, floor (100.01) x4;
 x1  |  x2  | x3  | x4
-----+------+-----+-----
 100 | -101 | 100 | 100
(1 row)
```

Функция TRUNC(n, m) возвращает число n, усеченное до m знаков после десятичной точки

```
masterskaya=# SELECT TRUNC(100.25678) X1, TRUNC(-100.25678) X2,
TRUNC(100.99) X3, TRUNC(100.25678, 2) X4;
 x1  |  x2  | x3  |   x4
-----+------+-----+--------
 100 | -100 | 100 | 100.25
(1 row)

masterskaya=#
```

Функция ROUND(n[,m]) возвращает число n, округленное до m знаков после десятичной точки по правилам математического округления.

```
postgres=# SELECT ROUND (200.2153) X1, ROUND (200.2) X2,
ROUND (200.99) X3, ROUND (200.412,2) X4;
 x1  | x2  | x3  |   x4
-----+-----+-----+--------
 200 | 200 | 201 | 200.41
(1 row)

postgres=#
```

Функция SIGN(n) определяет знак числа.

```
postgres=# SELECT SIGN (400.22) X1, SIGN (-400.22) X2, SIGN (0) X3;
 x1 | x2 | x3
----+----+----
  1 | -1 |  0
(1 row)
```

Функция MOD(n, m) возвращает остаток от деления n на m

```
postgres=# select mod (70,3) x1, mod (30, 2) x2, mod (422,4) x3;
 x1 | x2 | x3
----+----+----
  1 |  0 |  2
(1 row)
```

Функция POWER(n, m) возводит число n в степень m.

```
postgres=# select power (1,2) x1, power (122,2) x2, power (5,7) x3;
 x1 |  x2   |  x3
----+-------+-------
  1 | 14884 | 78125
(1 row)

postgres=#
```

Функция SQRT(n) возвращает квадратный корень от числа n

```
postgres=# select sqrt (1024) x;
 x
----
 32
(1 row)

postgres=#
```

Функция EXP(n) возводит e в степень n, а функция LN(n) вычисляет натуральный логарифм от n.

```
postgres=# select exp(2) x1, ln (3) x2, ln(exp(4)) x3;
        x1         |         x2         | x3
-------------------+--------------------+----
 7.38905609893065 | 1.0986122886681098 |  4
(1 row)

postgres=#
```

Функция LOG(n, m) производит вычисление логарифма m по основанию n.

```
postgres=# select log (5, 25) x1, log (5, 625) x2;
         x1         |         x2
--------------------+--------------------
 2.0000000000000000 | 4.0000000000000000
(1 row)

postgres=#
```

Функции Sin(n), Cos(n), Tan(n), Cot(n) производят вычисление тригонометрических функций.

```
postgres=# select sin(0) x1, cos (100) x2, tan (45) x3, cot (100) x4;
 x1 |         x2         |         x3         |         x4
----+--------------------+--------------------+--------------------
  0 | 0.8623188722876839 | 1.6197751905438615 | -1.702956919426469
(1 row)

postgres=#
```

Функция CONCAT(str1, str2) выполняет склеивание строк str1 и str2.

```
postgres=# select concat ('i ', 'am ', 'a ', 'hero ') x1, concat ('ye
s',NULL) x2;
      x1      | x2
--------------+----
 i am a hero  | yes
(1 row)

postgres=#
```

Функция LOWER(str) преобразует все символы строки str в строчные.

```
postgres=# select lower ('VOrOnA') x;
   x
--------
 vorona
(1 row)

postgres=#
```

Функция UPPER(str) преобразует все символы строки str в прописные

```
postgres=# select upper ('not') x1;
 x1
-----
 NOT
(1 row)

postgres=#
```

Функция INITCAP(str) возвращает строку str, в которой первые буквы всех слов преобразованы в прописные.

```
postgres=# select initcap ('vDislaV peTROV') x;
       x
-----------------
 Vdislav Petrov
(1 row)
```

Функция LTRIM(str, [,set]) удаляет все символы с начала строки до первого символа, которого нет в наборе символов set

```
postgres=# select ltrim ('13214 VSUET', '12345') x1;
   x1
-------
  VSUET
(1 row)
```

Функция RTRIM(str, [,set]) аналогична, но удаляет символы, начиная от конца строки.

```
postgres=# select rtrim ('voronej hjkglf',
postgres(# 'asdfghjkl;') x1;
    x1
---------
 voronej
(1 row)

postgres=#
```

Функция REPLACE(str, search_str, replace_str) осуществляет поиск образца search_str в строке str и каждое найденное вхождение заменяет на replace_str.

```
postgres=# select replace ('papa peve give my bori','papa','my') x1;
          x1
-----------------------
 my peve give my bori
(1 row)
```

Функция TRANSLATE(str, from_mask, to_mask) анализирует строку str и заменяет в ней все символы, встречающиеся в строке from_mask, на соответствующие символы из to_mask.

```
postgres=# select translate ('Pap1 P3ve g9ve', '139', 'aei') x1;
       x1
-----------------
 Papa Peve give
(1 row)

postgres=#
```

Функция LENGTH(str) возвращает длину строки str в символах.

```
postgres=# select length (' papa peve give my bori') x1, length ('1')
 x2;
 x1 | x2
----+----
 23 |  1
(1 row)

postgres=#
```

Функция ASCII(str) возвращает ASCII-код первого символа строки str в случае применения кодировок ASCII и UTF-8.

```
postgres=# select ascii ('hellov') x1;
 x1
-----
 104
(1 row)

postgres=#
```

Функция CHR(n) возвращает символ по его коду.

```
postgres=# select chr (123) x1, chr (213) x2;
 x1 | x2
----+----
 {  | Õ
(1 row)

postgres=#
```

Функция NOW() возвращает текущую дату и время по часам сервера.

```
postgres=# select now();
              now
-----------------------------------
 2023-05-19 15:51:35.997298-04
(1 row)

postgres=#
```

Функция JUSTIFY_INTERVAL(interval) преобразует интервал, указанный в виде строки в соответствующее значение.

```
postgres=# select now(), now() + JUSTIFY_INTERVAL ('30 DAYS 20 HOUR 5
 MINUTE') POTOM;
              now              |             potom
-------------------------------+-------------------------------
 2023-05-19 16:01:30.48101-04 | 2023-06-20 12:06:30.48101-04
(1 row)

postgres=#
```

Функция DATE_TRUNC(timestamp) используется для обрезки даты или интервала (DATE_TRUNC(interval)) до определенной точности.

```
postgres=# select date_trunc ('HOUR', now()) d1;
         d1
---------------------
 2023-05-19 16:00:00-04
(1 row)

postgres=# select date_trunc ('YEAR', now()) d1;
         d1
---------------------
 2023-01-01 00:00:00-05
(1 row)

postgres=#
```

Функция AGE([end_date,]start_date) возвращает разницу между датами, обозначенными как end_date и start_date.

```
postgres=# select current_date d1, age(make_timestamp(2022,7,15,7,11,
23.6)) d1;
     d1     |            d1
------------+---------------------------
 2023-05-19 | 10 mons 3 days 16:48:36.4
(1 row)

postgres=#
```

Функция EXTRACT(field FROM timestamp) извлекает элемент даты field из значения типа timestamp.

```
postgres=# select now(), extract (day from now()), extract (hour from
 now());
              now              | extract | extract
-------------------------------+---------+---------
 2023-05-19 16:11:58.354184-04 |      19 |      16
(1 row)

postgres=#
```

Функция TO_DATE(str, mask) преобразует строку str в дату

```
postgres=# select to_date ('22 NOV 2001', 'dd mon yyy');
  to_date
------------
 2001-11-22
(1 row)

postgres=#
```

Функция TO_CHAR(date, mask) преобразует дату date в символьную строку в соответствии с заданной маской

```
postgres=# select to_char (now(),'dd.mm.yy');
 to_char
----------
 19.05.23
(1 row)

postgres=#
```