МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ»

Факультет Управление и информатика в технологических системах

Кафедра Информационная безопасность

Специальность 10.05.03 «Информационная безопасность автоматизированных систем»

Отчет по практической работе №5

по дисциплине Безопасность Баз Данных

Тема: Дополнительные возможности POSTGRESQL.

Выполнила: студентка 3 курса

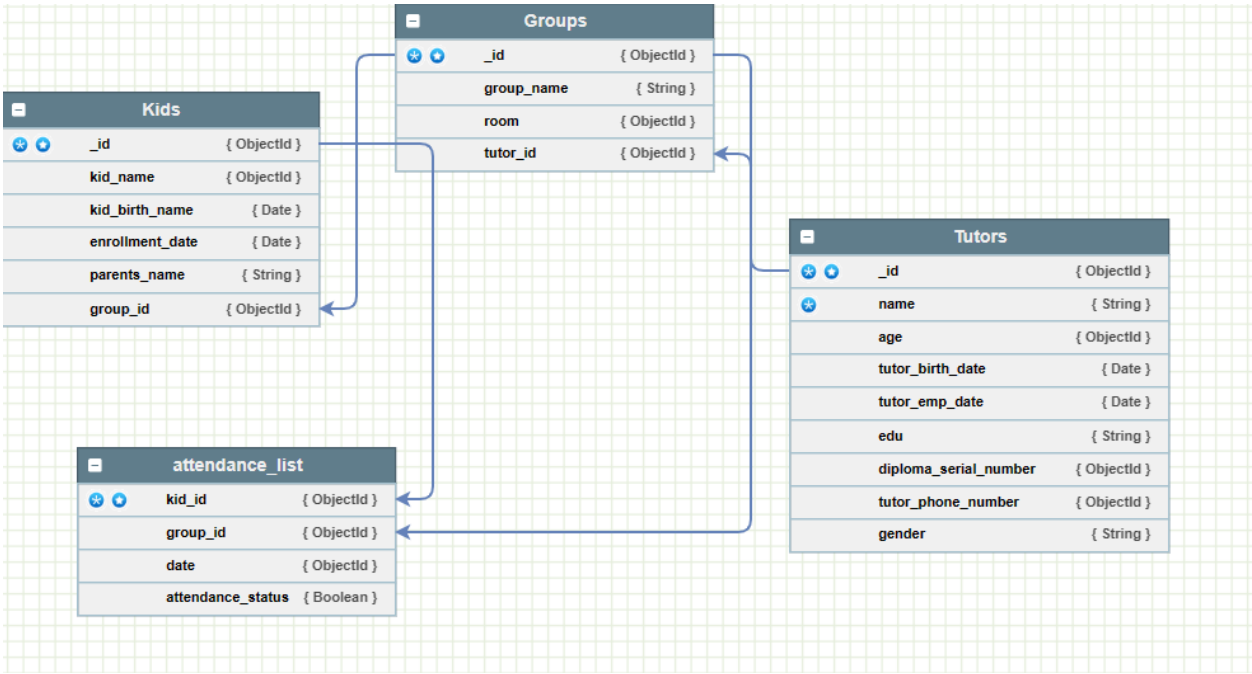
группы УБ-01

Лазарева Маргарита Вячеславовна

Воронеж – 2023

Цель работы: изучить функции sql для работы с базой данных.

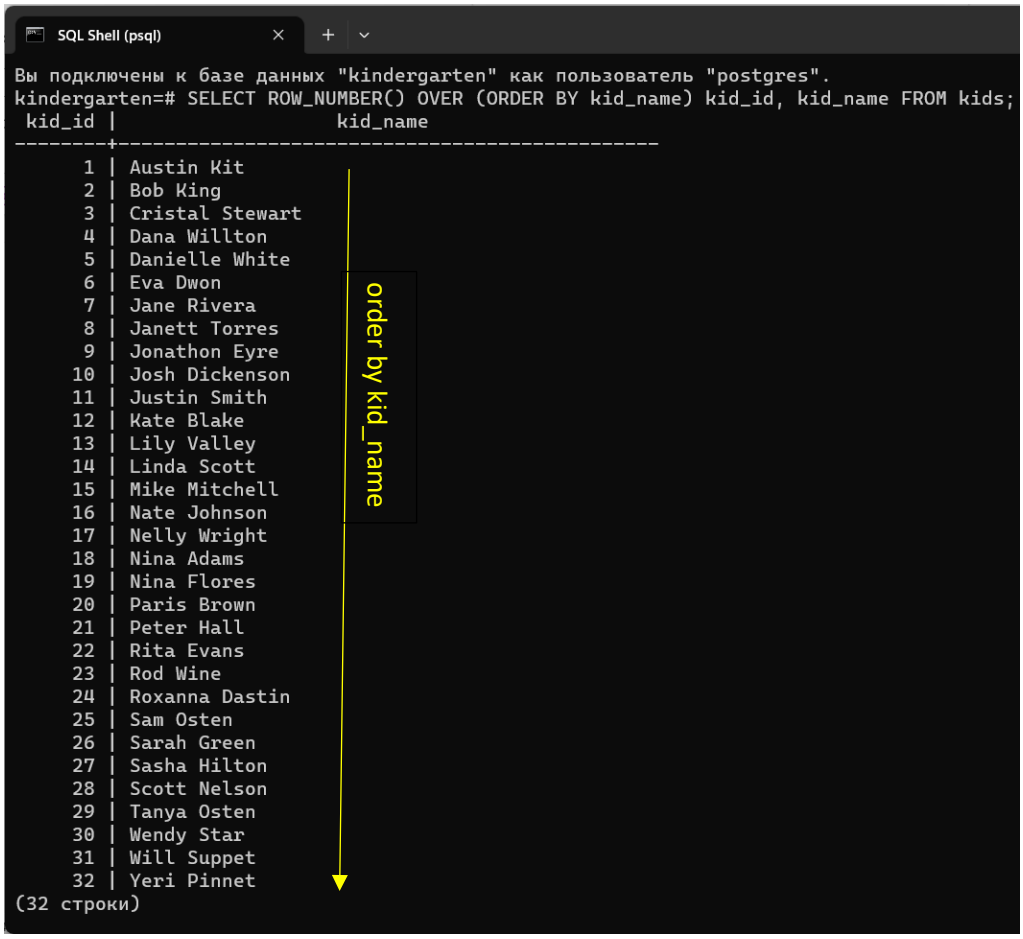
Модель данных для БД «детский сад»:



1. Функция ROW_NUMBER

Функция ROW_NUMBER генерирует порядковый номер строки запроса. Например:

SELECT ROW_NUMBER() OVER (ORDER BY kid_name) kid_id, kid_name FROM kids;



В данном запросе производится нумерация извлекаемых строк. Также ROW_NUMBER может применяться для ограничения количества обрабатываемых строк. Так, данный запрос извлекает первые десять строк данных:

SELECT * FROM (SELECT ROW_NUMBER() OVER (ORDER BY kid_name) kid_id, kid_name FROM kids) kids WHERE kid_id <= 10;

```
kindergarten=# SELECT * FROM ( SELECT ROW_NUMBER() OVER (ORDER BY kid_name) kid_id, kid_name FROM kids ) kids WHERE kid_
id <= 10;
 kid_id |                  kid_name
--------+-----------------------------------------------
      1 | Austin Kit
      2 | Bob King
      3 | Cristal Stewart
      4 | Dana Willton
      5 | Danielle White
      6 | Eva Dwon
      7 | Jane Rivera
      8 | Janett Torres
      9 | Jonathon Eyre
     10 | Josh Dickenson
(10 строк)
```

Если при группировке необходимо нумеровать строки для каждой группы в отдельности, следует использовать оператор PARTITION BY. Например:

SELECT ROW_NUMBER() OVER (PARTITION BY hometown) kid_id, kid_name, hometown, COUNT(*) FROM kids GROUP BY hometown;

```
kindergarten=# SELECT ROW_NUMBER() OVER (PARTITION BY hometown) kid_id, hometown, COUNT(*) FROM kids GROUP BY hometown;
 kid_id |    hometown    | count
--------+----------------+-------
      1 | Alexander City |     3
      1 | Andalusia      |     1
      1 | Anniston       |     1
      1 | Auburn         |     4
      1 | Bessemer       |     3
      1 | Clanton        |     2
      1 | Huntsville     |     3
      1 | Jasper         |     2
      1 | Montgomery     |    13
(9 строк)
```

Результатом является количество детей в каждом городе Алабамы. Больше всего детей из города Монтгомери.

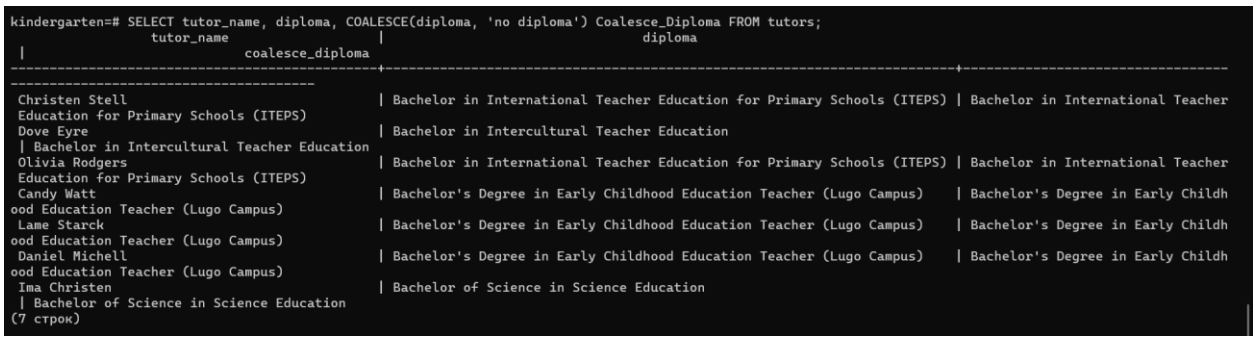
2. Функция COALESCE

Функция COALESCE, как правило, применяется чаще всего. Функция принимает несколько параметров: COALESCE(value [, ...])

Функция возвращает значение первого аргумента, значение которого не равно NULL. Данная функция вернет NULL только в том случае, если все аргументы имеют значение NULL. Рассмотрим практический пример. Поле diploma в таблице tutors может содержать значения NULL. При выполнении запроса вида:

SELECT tutor_name, diploma, COALESCE(diploma, ‘no diploma’) Coalesce_Diploma FROM tutors;

значение NULL будет заменено на строку ‘no diploma’. Обратим внимание на то, что в случае формирования значения при помощи функции ему назначается псевдоним. Результаты запроса будут иметь такой вид:



Никакая из строк третьего столбца ‘coalesce_diploma’ не имеет текст ‘no diploma’, т.к. все тьюторы в нашем детском саду имеют высшее образование и, соответственно, диплом. Если бы какой-то из тьютеров не имел его, в строке третьего столбца появился бы текст ‘no diploma’.

3. Числовые функции

В PostgreSQL предусмотрен ряд встроенных функций для работы с числами.

Функция ABS

Функция ABS(n) возвращает абсолютное значение числа n. Например:

SELECT ABS(8) X1, ABS(-8.8) X2, ABS(-8) X3;

```
kindergarten=# SELECT ABS(8) X1, ABS(-8.8) X2, ABS(-8) X3;
 x1 | x2  | x3
----+-----+----
  8 | 8.8 |  8
(1 строка)
```

Функция CEIL

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

SELECT CEIL(8) X1, CEIL(-8) X2, CEIL(8.8) X3, CEIL(-8.8) X4;

```
kindergarten=# SELECT CEIL(8) X1, CEIL(-8) X2, CEIL(8.8) X3, CEIL(-8.8) X4;
 x1 | x2 | x3 | x4
----+----+----+----
  8 | -8 |  9 | -8
(1 строка)
```

8.8 CEIL округлил до 9 как наибольшее целое, а -8.8 округлил до 8 как наименьшее целое.

Функция FLOOR

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

SELECT FLOOR(8.11) X1, FLOOR(-8.11) X2, FLOOR(8.89) X3, FLOOR(8.01) X4;

```
kindergarten=# SELECT FLOOR(8.11) X1, FLOOR(-8.11) X2, FLOOR(8.89) X3, FLOOR(8.01) X4;
 x1 | x2 | x3 | x4
----+----+----+----
  8 | -9 |  8 |  8
(1 строка)


kindergarten=# SELECT FLOOR(8.11) X1, FLOOR(-8.11) X2, FLOOR(-8.89) X3, FLOOR(8.01) X4;
 x1 | x2 | x3 | x4
----+----+----+----
  8 | -9 | -9 |  8
(1 строка)


kindergarten=# SELECT FLOOR(8.11) X1, FLOOR(-8.89) X2, FLOOR(8.89) X3, FLOOR(8.01) X4;
 x1 | x2 | x3 | x4
----+----+----+----
  8 | -9 |  8 |  8
(1 строка)
```

Функция TRUNC

Функция TRUNC(n[, m]) возвращает число n, усеченное до m знаков после десятичной точки. Параметр m может не указываться – в этом случае n усекается до целого.

SELECT TRUNC(8.9834945) X1, TRUNC(-8.9834945) X2, TRUNC(8.99) X3, TRUNC(8.9834945, 2) X4;

```
kindergarten=# SELECT TRUNC(8.9834945) X1, TRUNC(-8.9834945) X2, TRUNC(8.99) X3, TRUNC(8. 9834945, 2) X4;
ОШИБКА:  ошибка синтаксиса (примерное положение: "9834945")
СТРОКА 1: ...1, TRUNC(-8.9834945) X2, TRUNC(8.99) X3, TRUNC(8. 9834945, 2...
                                                             ^
kindergarten=# SELECT TRUNC(8.9834945) X1, TRUNC(-8.9834945) X2, TRUNC(8.99) X3, TRUNC(8.9834945, 2) X4;
 x1 | x2 | x3 |  x4
----+----+----+------
  8 | -8 |  8 | 8.98
(1 строка)
```

Функция ROUND

Функция ROUND(n[,m]) возвращает число n, округленное до m знаков после десятичной точки по правилам математического округления. Параметр m может не указываться – в этом случае n округляется до целого.

SELECT ROUND(8.9834945) X1, ROUND(8.7) X2, ROUND(8.99) X3, ROUND(8. 9834945, 2) X4;

```
kindergarten=# SELECT ROUND(8.9834945) X1, ROUND(8.7) X2, ROUND(8.99) X3, ROUND(8.9834945,2) X4;
 x1 | x2 | x3 |  x4
----+----+----+------
  9 |  9 |  9 | 8.98
(1 строка)


kindergarten=#
```

Функция SIGN

Функция SIGN(n) определяет знак числа. Если n положительное, то функция возвращает 1. Если отрицательное – возвращается –1. Если равно нулю, то возвращается 0. Например:

SELECT SIGN(8.11) X1, SIGN(-8.11) X2, SIGN(0) X3;

```
kindergarten=# SELECT SIGN(8.11) X1, SIGN(-8.11) X2, SIGN(0) X3;
 x1 | x2 | x3
----+----+----
  1 | -1 |  0
(1 строка)
```

Функция MOD

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

SELECT MOD(8, 4) X1, MOD(8, 3) X2, MOD(8, 11) X3;

```
kindergarten=# SELECT MOD(8, 4) X1, MOD(8, 3) X2, MOD(8, 11) X3;
 x1 | x2 | x3
----+----+----
  0 |  2 |  8
(1 строка)
```

Функция POWER

Функция POWER(n, m) возводит число n в степень m. Степень может быть дробной и отрицательной, что существенно расширяет возможности данной функции.

SELECT POWER(8, 0) X1, POWER(8, 2) X2, POWER(8, 0.5) X3, POWER(8, -0.5) X4;

```
kindergarten=# SELECT POWER(8, 0) X1, POWER(8, 2) X2, POWER(8, 0.5) X3, POWER(8, -0.5) X4;
 x1 | x2 |         x3         |         x4
----+----+--------------------+--------------------
  1 | 64 | 2.8284271247461901 | 0.3535533905932738
(1 строка)
```

Функция SQRT

Функция SQRT(n) возвращает квадратный корень от числа n. Например:

SELECT SQRT(8) X;

```
kindergarten=# SELECT SQRT(8) X;
         x
--------------------
 2.82842712474619O3
(1 строка)


kindergarten=# SELECT SQRT(9) X;
 x
---
 3
(1 строка)


kindergarten=# SELECT SQRT(-9) X;
ОШИБКА:  извлечь квадратный корень отрицательного числа нельзя
kindergarten=#
```

Функции EXP и LN

Функция EXP(n) возводит e в степень n, а функция LN(n) вычисляет натуральный логарифм от n (при этом значение n должно быть больше нуля). Пример:

SELECT EXP(2) X1, LN(2) X2, LN(EXP(2)) X3;

```
kindergarten=# SELECT EXP(2) X1, LN(2) X2, LN(EXP(2)) X3;
        x1         |         x2         | x3
-------------------+--------------------+----
 7.38905609893065 | 0.6931471805599453 |  2
(1 строка)
```

Функция LOG

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

SELECT LOG(2, 8) X1, LOG(4, 16) X2;

```
kindergarten=# SELECT LOG(2, 8) X1, LOG(4, 16) X2;
         x1          |         x2
---------------------+--------------------
 3.0000000000000000 | 2.0000000000000000
(1 строка)
```

4. Тригонометрические функции

PostgreSQL поддерживает вычисление основных тригонометрических функций:

• SIN(n) – синус n (где n – угол в радианах);

• COS(n) – косинус n (где n – угол в радианах);

• TAN(n) – тангенс n (где n – угол в радианах);

• COT(n) – котангенс n (где n – угол в радианах).

Пример:

SELECT SIN(30) X1, COS(60) X2, TAN(90) X3, COT(90);

```
kindergarten=# SELECT SIN(30) X1, COS(60) X2, TAN(90) X3, COT(90);
         x1          |         x2          |         x3         |         cot
---------------------+---------------------+--------------------+--------------------
 -0.9880316240928618 | -0.9524129804151563 | -1.995200412208242 | -0.5012027833801532
(1 строка)


kindergarten=# SELECT SIN(0) X1, COS(0) X2, TAN(-1) X3, COT(1);
 x1 | x2 |         x3          |        cot
----+----+---------------------+--------------------
  0 |  1 | -1.5574077246549023 | 0.6420926159343306
(1 строка)


kindergarten=# SELECT SIN(0) X1, COS(0) X2, TAN(1) X3, COT(-1);
 x1 | x2 |         x3         |         cot
----+----+--------------------+--------------------
  0 |  1 | 1.5574077246549023 | -0.6420926159343306
(1 строка)
```

5. Строковые и символьные функции

Функция CONCAT

Функция CONCAT(str1, str2) выполняет конкатенацию строк str1 и str2. Если один из аргументов равен NULL, то он воспринимается как пустая строка. Если оба аргумента равны NULL, то функция возвращает NULL. Пример:

SELECT CONCAT('Дождь начался ', 'еще в 9') X1, CONCAT('Дождь', NULL) X2, CONCAT(NULL, 'Дождь') X3, CONCAT(NULL, NULL) X4;

```
kindergarten=# SELECT CONCAT('Дождь начался ', 'еще в 9') X1, CONCAT('Дождь', NULL) X2, CONCAT(NULL, 'Дождь') X3, CONCAT
(NULL, NULL) X4;
         x1          |  x2   |  x3   | x4
---------------------+-------+-------+----
 Дождь начался еще в 9 | Дождь | Дождь |
(1 строка)
```

Для конкатенации строк PostgreSQL поддерживает специальный оператор конкатенации ||, который работает аналогично функции CONCAT, например:

SELECT CONCAT('Дождь начался ', 'еще в 9') X1, ('Дождь начался ' || 'еще в 9') X2;

```
kindergarten=# SELECT CONCAT('Дождь начался ', 'еще в 9') X1, ('Дождь начался ' || 'еще  в  9') X2;
          x1           |           x2
-----------------------+-----------------------
 Дождь начался еще в 9 | Дождь начался еще  в  9
(1 строка)
```

SELECT CONCAT('Дождь начался ', 'еще в 9') "CONCAT function", ('Дождь начался ' || 'еще в 9') "|| operator";

```
kindergarten=# SELECT CONCAT('Дождь начался ', 'еще в 9') "CONCAT function", ('Дождь начался ' || 'еще  в  9') "|| opera
tor";
    CONCAT function    |       || operator
-----------------------+-----------------------
 Дождь начался еще в 9 | Дождь начался еще  в  9
(1 строка)
```

Функция LOWER

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

SELECT LOWER(’They work for FBI.’) X;

```
kindergarten=# SELECT LOWER('They work for FBI.') X;
         x
--------------------
 they work for fbi.
(1 строка)
```

Функция UPPER

Функция UPPER(str) преобразует все символы строки str в прописные. Пример:

SELECT UPPER(’west states: wa, mt, or, id, wy, ut, nv, ca, az, nm, co’) X;

```
kindergarten=# SELECT UPPER('west states: wa, mt, or, id, wy, ut, nv, ca, az, nm, co') X;
                          x
-------------------------------------------------------
 WEST STATES: WA, MT, OR, ID, WY, UT, NV, CA, AZ, NM, CO
(1 строка)
```

Функция INITCAP

Функция INITCAP(str) возвращает строку str, в которой первые буквы всех слов преобразованы в прописные. Функция удобна для форматирования полного имени при построении отчетов.

Пример:

SELECT INITCAP('Catelin frank') X;

```
kindergarten=# SELECT INITCAP('Catelin frank') X;
       x
---------------
 Catelin Frank
(1 строка)
```

Функции LTRIM и RTRIM

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

По умолчанию set состоит из одного пробела и может не указываться. Функция RTRIM(str [,set]) аналогична LTRIM, но удаляет символы, начиная от конца строки. Рассмотрим несколько примеров:

select ltrim('state alabama state', 'state') x1 union all select rtrim('state alabama state', 'state') x1;

```
kindergarten=# select ltrim('state alabama state', 'state') x1 union all select rtrim('state alabama state', 'state') x1;
       x1
-----------------
  alabama state
 state alabama
(2 строки)
```

Мы написали текст ‘state alabama state’ для обеих функций, чтобы убедится, что в первой строке (в которой мы использовали функцию ltrim) у нас удалились слово state и пробел, которое в тексте стояло до слова alabama (т.е. функция удалила символы, с начала строки до первого символа, который не указан нами), а во второй строке (в которой мы использовали функцию rtrim), у нас удалились слово state и пробел, которое в тексте стояло после слова alabama (т.е. функция удалила символы, начиная от конца строки).

Функция REPLACE

Функция REPLACE(str, search_str, replace_str) осуществляет поиск образца search_str в строке str и каждое найденное вхождение заменяет на replace_str. Поиск подстроки ведется с учетом регистра.

Примеры:

Пример успешной замены

SELECT ('Sasha Hilton was born in Montgomery') X1 UNION ALL SELECT ('replace "Montgomery" with "Alexander City"') X1 UNION ALL SELECT REPLACE('Sasha Hilton was born in Montgomery', 'Montgomery', 'Alexander City') X1;

```
kindergarten=# SELECT ('Sasha Hilton was born in Montgomery') X1 UNION ALL SELECT ('replace "Montgomery" with "Alexander City"') X1 UNION ALL SELECT REPLACE
('Sasha Hilton was born in Montgomery', 'Montgomery', 'Alexander City') X1;
                    x1
------------------------------------------
 Sasha Hilton was born in Montgomery
 replace "Montgomery" with "Alexander City"
 Sasha Hilton was born in Alexander City
(3 строки)
```

Пример успешной замены:

SELECT ('Sasha Hilton was born in Montgomery') X1, ('replace "Montgomery" with "Alexander City"') X2, REPLACE('Sasha Hilton was born in Montgomery', 'Montgomery', 'Alexander City') X3;

```
kindergarten=# SELECT ('Sasha Hilton was born in Montgomery') X1, ('replace "Montgomery" with "Alexander City"') X2, REPLACE('Sasha Hilton was
 born in Montgomery', 'Montgomery', 'Alexander City') X3;
                x1                 |                    x2                    |                  x3
-----------------------------------+------------------------------------------+--------------------------------------
 Sasha Hilton was born in Montgomery | replace "Montgomery" with "Alexander City" | Sasha Hilton was born in Alexander City
(1 строка)
```

Пример провальной замены:

```
             SELECT ('Sasha Hilton was born in Montgomery') X1, ('replace "Montgomery" with "Alexander City"') X2, REPLACE('Sasha Hilton was born in Montg
omery', 'montgomery', 'Alexander City') X3, ('the replacement failed due to the function not finding the specified string') X4;
               x1                  |                    x2                    |                x3                |
  x4
-----------------------------------+------------------------------------------+----------------------------------+----------------------------------
-----------
 Sasha Hilton was born in Montgomery | replace "Montgomery" with "Alexander City" | Sasha Hilton was born in Montgomery | the replacement failed due to the
function not finding the specified string
(1 строка)
```

При этом запросе функция не нашла строку, которую нужно заменить из-за неверного указания строки (Montgomery ≠ montgomery).

Функция TRANSLATE

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. Для корректной работы функции строки from_mask и to_mask должны иметь одинаковую длину или строка from_mask должна быть длиннее, чем to_mask. Если from_mask длиннее, чем to_mask, и в процессе обработки строки str обнаружатся символы, соответствующие одному из символов from_mask, и при этом им не найдется соответствия в to_mask, то такие символы будут удалены из строки str. Если передать from_mask или to_mask значение, равное NULL, то функция возвратит значение NULL. Сравнение производится с учетом регистра.

Примеры:

SELECT ('Yeri Pinnet is in #FlowerBloom# group.') "SENTENCE"

UNION ALL

SELECT TRANSLATE('Yeri Pinnet is in #FlowerBloom# group.', '#', '"') "SENTENCE";

```
kindergarten=# SELECT ('Yeri Pinnet is in #FlowerBloom# group.') "SENTENCE" UNION ALL SELECT TRANSLATE('Yeri Pinnet is in #FlowerBloom# group.', '#', '"') "
SENTENCE";
              SENTENCE
---------------------------------------
 Yeri Pinnet is in #FlowerBloom# group.
 Yeri Pinnet is in "FlowerBloom" group.
(2 строки)
```

Если передать NULL одной из строк:

```
kindergarten=# SELECT ('Yeri Pinnet is in #FlowerBloom# group.') "SENTENCE" UNION ALL SELECT TRANSLATE('Yeri Pinnet is in #FlowerBloom# group.', '#', NULL)
"SENTENCE";
              SENTENCE
---------------------------------------
 Yeri Pinnet is in #FlowerBloom# group.

(2 строки)
```

Функция SUBSTR

Функция SUBSTR(str, m [,n]) возвращает фрагмент строки str, начиная с символа m длиной n символов. Длину можно не указывать – в этом случае возвращается строка от символа m и до конца строки str. Нумерация символов идет с 1. Если указать m равное 0, то копирование все равно начнется с первого символа. Задание отрицательного значения m приводит к тому, что символы отсчитываются от конца строки, а не от начала. Задание значений m, превышающих по абсолютному значению длину строки, приводит к тому, что функция возвращает NULL.

Пример:

SELECT SUBSTR('Nina Flores is 5', 6) X1, SUBSTR('Nina Flores is 5', 0) X2, SUBSTR('Nina Flores is 5', -6) X3, SUBSTR('Nina Flores is 5', 150) X4;

```
kindergarten=# SELECT SUBSTR('Nina Flores is 5', 6) X1, SUBSTR('Nina Flores is 5', 0) X2, SUBSTR('Nina Flores is 5', -6)
 X3, SUBSTR('Nina Flores is 5', 150) X4;
     x1      |        x2        |        x3        | x4
-------------+------------------+------------------+----
 Flores is 5 | Nina Flores is 5 | Nina Flores is 5 |
(1 строка)
```

Функция LENGTH

Функция LENGTH(str) возвращает длину строки str в символах. Для пустой строки функция вернет 0, а для значения NULL – NULL.

Пример:

SELECT LENGTH(’Nina Flores was present on the 25th, 26th, 27th and 28th of March’) X1, LENGTH(’’) X2, LENGTH(NULL) X3;

```
kindergarten=# SELECT LENGTH('Nina Flores was present on the 25th, 26th, 27th and 28th of  March.') X1, LENGTH('') X2, L
ENGTH(NULL) X3;
 x1 | x2 | x3
----+----+----
 67 |  0 |
(1 строка)
```

Функция ASCII

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

Пример:

SELECT ASCII(’Rita’) X1, ASCII(’Рита’) X2;

SELECT ASCII('Sofa') X1, ASCII('Софа') X2;

SELECT ASCII('Tamara') X1, ASCII('Тамара') X2;

```
kindergarten=# SELECT ASCII('Rita') X1, ASCII('Рита') X2;
 x1 |  x2
----+------
 82 | 1056
(1 строка)


kindergarten=# SELECT ASCII('Sofa') X1, ASCII('Софа') X2;
 x1 |  x2
----+------
 83 | 1057
(1 строка)


kindergarten=# SELECT ASCII('Tamara') X1, ASCII('Тамара') X2;
 x1 |  x2
----+------
 84 | 1058
(1 строка)
```

Функция CHR

Функция CHR(n) возвращает символ по его коду.

Пример:

SELECT CHR(82) X1, CHR(83) X2, CHR(84) X3 UNION ALL SELECT CHR(1056) X1, CHR(1057) X2, CHR(1058) X3;

```
kindergarten=# SELECT CHR(82) X1, CHR(83) X2, CHR(84) X3 UNION ALL SELECT CHR(1056) X1, CHR(1057) X2, CHR(1058) X3;
 x1 | x2 | x3
----+----+----
 R  | S  | T
 Р  | С  | Т
(2 строки)
```

6. Функции работы с датой и временем

Функция NOW

Это одна из самых часто употребляемых функций, она возвращает текущую дату и время по часам сервера.

Пример:

SELECT NOW();

```
kindergarten=# SELECT NOW();
              now
-------------------------------
 2023-05-02 16:03:53.931365+03
(1 строка)
```

Функция JUSTIFY_INTERVAL

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

Пример:

SELECT NOW() D1, NOW() + JUSTIFY_INTERVAL(’5 DAYS 12 HOUR 3 MINUTE’) D2, NOW() - JUSTIFY_INTERVAL(’5 DAYS 12 HOUR 3 MINUTE’) D3;

```
kindergarten=# SELECT NOW() D1, NOW() + JUSTIFY_INTERVAL('5 DAYS 12 HOUR 3 MINUTE') D2, NOW() - JUSTIFY_INTERVAL('5 DAYS
 12 HOUR 3 MINUTE') D3;
              d1               |              d2               |              d3
-------------------------------+-------------------------------+-------------------------------
 2023-05-02 16:07:56.374546+03 | 2023-05-08 04:10:56.374546+03 | 2023-04-27 04:04:56.374546+03
(1 строка)
```

Функция DATE_TRUNC

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

Пример:

SELECT DATE_TRUNC(’HOUR’, NOW()) D1, DATE_TRUNC(’DAY’, NOW()) D2, DATE_TRUNC(’MONTH’, NOW()) D3;

и

SELECT DATE_TRUNC('HOUR', NOW()) D1, DATE_TRUNC('DAY', NOW()) D2, DATE_TRUNC('MONTH', NOW()) D3, DATE_TRUNC('YEAR', NOW()) D4;

```
kindergarten=# SELECT DATE_TRUNC('HOUR', NOW()) D1, DATE_TRUNC('DAY', NOW()) D2, DATE_TRUNC('MONTH', NOW()) D3;
           d1           |           d2           |           d3
------------------------+------------------------+------------------------
 2023-05-02 16:00:00+03 | 2023-05-02 00:00:00+03 | 2023-05-01 00:00:00+03
(1 строка)


kindergarten=# SELECT DATE_TRUNC('HOUR', NOW()) D1, DATE_TRUNC('DAY', NOW()) D2, DATE_TRUNC('MONTH', NOW()) D3, DATE_TRU
NC('YEAR', NOW()) D4;
           d1           |           d2           |           d3           |           d4
------------------------+------------------------+------------------------+------------------------
 2023-05-02 16:00:00+03 | 2023-05-02 00:00:00+03 | 2023-05-01 00:00:00+03 | 2023-01-01 00:00:00+03
(1 строка)
```

Получение начала и конца месяца

Для получения дат соответствующих началу и концу месяца необходимо использовать функции DATE_TRUNC и JUSTIFY_INTERVAL.

Пример:

SELECT DATE_TRUNC(’MONTH’, NOW()) D1, DATE_TRUNC(’MONTH’, NOW()) + JUSTIFY_INTERVAL(’1 MONTH - 1 DAY’) D2;

```
kindergarten=# SELECT DATE_TRUNC('MONTH', NOW()) D1, DATE_TRUNC('MONTH', NOW()) + JUSTIFY_INTERVAL('1 MONTH - 1 DAY') D2
;
           d1           |           d2
------------------------+------------------------
 2023-05-01 00:00:00+03 | 2023-05-30 00:00:00+03
(1 строка)
```

Данные функции также могут быть использованы для определения количества дней в заданном месяце.

Например:

SELECT NOW() D1, TO_CHAR(DATE_TRUNC(’MONTH’, NOW()) + JUSTIFY_INTERVAL(’1 MONTH - 1 DAY’), ’DD’) D2;

```
kindergarten=# SELECT NOW() D1, TO_CHAR(DATE_TRUNC('MONTH', NOW()) + JUSTIFY_INTERVAL('1 MONTH - 1 DAY'), 'DD') D2;
              d1               | d2
-------------------------------+----
 2023-05-02 16:17:50.954131+03 | 30
(1 строка)
```

Функция AGE

Функция AGE([end_date,]start_date) возвращает разницу между датами, обозначенными как end_date и start_date. Если параметр end_date опущен, то используется значение глобальной переменной CURRENT_DATE, которая содержит текущую дату (тип date, дата без времени).

Пример:

SELECT CURRENT_DATE D1, AGE(MAKE_TIMESTAMP(2015, 5, 23, 16, 15, 23.5)) D2, AGE(MAKE_DATE(2023, 05, 02), MAKE_TIMESTAMP(2013, 7, 15, 8, 15, 23.5)) D3;

```
kindergarten=# SELECT CURRENT_DATE D1, AGE(MAKE_TIMESTAMP(2015, 5, 23, 16, 15, 23.5)) D2, AGE(MAKE_DATE(2023, 05, 02), M
AKE_TIMESTAMP(2013, 7, 15, 8, 15, 23.5)) D3;
     d1     |               d2                |               d3
------------+---------------------------------+---------------------------------
 2023-05-02 | 7 years 11 mons 9 days 07:44:36.5 | 9 years 9 mons 17 days 15:44:36.5
(1 строка)
```

В данном примере также используются функции MAKE_TIMESTAMP и MAKE_DATE, которые возвращают значения типов timestamp и date соответственно.

Функция EXTRACT

Функция EXTRACT(field FROM timestamp) извлекает элемент даты field из значения типа timestamp. Также существует функция EXTRACT(field FROM interval) для работы со значениями типа interval.

Пример:

SELECT NOW() D1, EXTRACT(MONTH FROM NOW()) D2, EXTRACT(YEAR FROM NOW()) D3, EXTRACT(MINUTE FROM NOW()) D4;

```
kindergarten=# SELECT NOW() D1, EXTRACT(MONTH FROM NOW()) D2, EXTRACT(YEAR FROM NOW()) D3, EXTRACT(MINUTE FROM NOW()) D4
;
              d1               | d2 |  d3  | d4
-------------------------------+----+------+----
 2023-05-02 16:43:19.011942+03 |  5 | 2023 | 43
(1 строка)
```

Количество месяцев между двумя датами

Функции EXTRACT, AGE и DATE_TRUNC могут быть использованы для нахождения количества месяцев между двумя датами.

Пример:

SELECT NOW() D1, EXTRACT(MONTH FROM AGE(DATE_TRUNC(’MONTH’, NOW()), DATE_TRUNC(’MONTH’, MAKE_DATE(2023, 12, 31)))) D2;

SELECT NOW() D1, EXTRACT(MONTH FROM AGE(DATE_TRUNC('MONTH', NOW()), DATE_TRUNC('MONTH', MAKE_DATE(2023, 12, 5)))) D2;

SELECT NOW() D1, EXTRACT(MONTH FROM AGE(DATE_TRUNC('MONTH', NOW()), DATE_TRUNC('MONTH', MAKE_DATE(2022, 12, 5)))) D2;

```
kindergarten=# SELECT NOW() D1, EXTRACT(MONTH FROM AGE(DATE_TRUNC('MONTH', NOW()), DATE_TRUNC('MONTH', MAKE_DATE(2023, 1
2, 31)))) D2;
             d1             | d2
----------------------------+----
 2023-05-02 16:45:12.158+03 | -7
(1 строка)


kindergarten=# SELECT NOW() D1, EXTRACT(MONTH FROM AGE(DATE_TRUNC('MONTH', NOW()), DATE_TRUNC('MONTH', MAKE_DATE(2023, 1
2, 5)))) D2;
              d1              | d2
------------------------------+----
 2023-05-02 16:47:19.984747+03 | -7
(1 строка)


kindergarten=# SELECT NOW() D1, EXTRACT(MONTH FROM AGE(DATE_TRUNC('MONTH', NOW()), DATE_TRUNC('MONTH', MAKE_DATE(2022, 1
2, 5)))) D2;
              d1              | d2
------------------------------+----
 2023-05-02 16:47:55.724682+03 |  5
(1 строка)
```

Функция TO_DATE

Функция TO_DATE(str, mask) преобразует строку str в дату. Преобразование ведется по маске mask.

Пример:

SELECT TO_DATE(’28 Mar 2023’, ’DD Mon YYY’) D1,
TO_DATE(’28.03.2023’, ’dd.mm.yy’) D2;

```
kindergarten=# SELECT TO_DATE('28 Mar 2023', 'DD Mon YYY') D1, TO_DATE('28.03.2023', 'dd.mm.yy') D2;
     d1     |     d2
------------+------------
 2023-03-28 | 2023-03-28
(1 строка)
```

Функция TO_CHAR

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

Пример:

SELECT NOW() D1, TO_CHAR(NOW(), ’DD.MM.YY HH24:MI’) D2;

```
kindergarten=# SELECT NOW() D1, TO_CHAR(NOW(), 'DD.MM.YY HH24:MI') D2;
              d1               |       d2
-------------------------------+----------------
 2023-05-02 16:56:48.333899+03 | 02.05.23 16:56
(1 строка)
```

Вывод: в этой практической работе мы изучили работу функций sql.