**Что такое SQL?**

**SQL-запрос** представляет собой совокупность операторов, инструкций, вычисляемых функций.

**ВИДЫ ОПЕРАТОРОВ SQL**

Операторы *SQL* делятся на:

* **операторы определения данных** **(Data Definition Language, DDL)** — с их помощью создаются и изменяются объекты в БД (сама БД, таблицы, функции, процедуры, пользователи и т. д.);
* **операторы манипуляции данными** **(Data Manipulation Language, DML)** — с их помощью проводятся манипуляции с данными в таблицах;
* **операторы определения доступа к данным** **(Data Control Language, DCL)** — с их помощью, как следует из названия, создаются и изменяются разрешения на определённые операции с объектами в БД;
* **операторы управления транзакциями** **(Transaction Control Language, TCL)** — с их помощью осуществляется комплекс определённых действий, причём так, что либо все эти действия выполняются успешно, либо ни одно из них не выполняется вообще.

**Metabase** — это бесплатный и удобный инструмент для первичного анализа данных с возможностью передавать запросы.

**Логин:**demo3@skillfactory.ru

**Пароль:** bQM3dsiC\_vuj7k

1. **Получаем все данные из таблицы**

|  |  |
| --- | --- |
| **Название поля** | **Содержимое** |
| *position* | номер в базе данных |
| *movie\_title* | название фильма |
| *year* | год выпуска |
| *country* | страна выпуска |
| *rating* | рейтинг фильма в базе |
| *overview* | описание фильма |

**SELECT** \*

**FROM** **sql**.kinopoisk

Напишите запрос, который выведет из таблицы kinopoisk столбцы с названием фильма, годом его выпуска и рейтингом.

**SELECT** \*

**FROM** **sql**.kinopoisk

Select movie\_title: 'str', year: 'int', rating: 'float'

Where movie\_title ='str';

From sql.kinopoisk

movie\_title: 'str', year: 'int', rating: 'float'

year: 'int'

rating: 'float'

SELECT movie\_title, year, 2023 - year, rating

FROM sql.kinopoisk

**SELECT director, movie\_title, 10 - rating**

**FROM sql.kinopoisk**

**SELECT**

director,

movie\_title,

**10** - rating **AS** difference

**FROM** **sql**.kinopoisk

SELECT \*

FROM sql.kinopoisk

WHERE year = 1999

**SELECT**

position,

movie\_title,

year,

director

**FROM** **sql**.kinopoisk

**WHERE** year < **1984**

**SELECT** \*

**FROM** **sql**.kinopoisk

**WHERE** year <> **2000**

**SELECT** \*

**FROM** **sql**.kinopoisk

**WHERE** year >= **2000**

**AND** rating >= **8**

**SELECT** \*

**FROM** **sql**.kinopoisk

**WHERE** year >= **1975**

**AND** year <= **1985**

**SELECT** \*

**FROM** **sql**.kinopoisk

**WHERE** year **BETWEEN** **1975** **AND** **1985**

Если вы знаете английский, назначение BETWEEN не станет для вас неожиданностью: оператор фильтрует строки, которые находятся между двумя значениями.

**SELECT**

director,

movie\_title,

**FROM** **sql**.kinopoisk

WHERE rating >= 8.5

SELECT

director,

movie\_title

FROM sql.kinopoisk

WHERE rating >= 8.5

SELECT

director,

movie\_title

FROM sql.kinopoisk

WHERE rating BETWEEN 8.7 AND 8.8

**SELECT** \*

**FROM** **sql**.kinopoisk

**WHERE** year **NOT** **BETWEEN** **1965** **AND** **1980**

**SELECT**

year,

movie\_title,

director

**FROM** **sql**.kinopoisk

**WHERE** (rating > **8**.**5** **AND** year < **2000**)

**OR** year >= **2000**

щё один полезный оператор для фильтрации строк — IN.

Конструкции с IN имеют следующий вид:

**column** **IN** (value1, value2, value3)

Эта запись аналогична следующей: column = value1 OR column = value2 OR column = value3 — но выглядит проще и компактнее.

SELECT \*

FROM sql.kinopoisk

WHERE year in(2000, 1985, 1939)

**Обратите внимание!** Если в алиасе используются пробелы, необходимо заключать весь псевдоним в двойные кавычки, например, movie\_title AS "Movie Title".

**ПРОСТЫЕ ОПЕРАЦИИ С ДАННЫМИ**

Со столбцами, которые содержат числовые данные, можно проводить арифметические операции:

* + сложение с помощью + ;
  + вычитание с помощью - (этот тип операции вы уже проводили, когда определяли «возраст» фильма);
  + умножение с помощью \* ;
  + деление с помощью / ;

**Важно!** Если и числитель, и знаменатель — целые числа, результат деления также будет целочисленным, то есть этот оператор произведёт деление нацело.

* + получение остатка от деления с помощью % .
* **9.3. Математические функции и операторы**
* Математические операторы определены для множества типов PostgreSQL. Как работают эти операции с типами, для которых нет стандартных соглашений о математических действиях (например, с типами даты/времени), мы опишем в последующих разделах.
* В [Таблице 9.4](https://postgrespro.ru/docs/postgresql/11/functions-math#FUNCTIONS-MATH-OP-TABLE) перечислены все доступные математические операторы.
* **Таблица 9.4. Математические операторы**

| **Оператор** | **Описание** | **Пример** | **Результат** |
| --- | --- | --- | --- |
| + | сложение | 2 + 3 | 5 |
| - | вычитание | 2 - 3 | -1 |
| \* | умножение | 2 \* 3 | 6 |
| / | деление (при целочисленном делении остаток отбрасывается) | 4 / 2 | 2 |
| % | остаток от деления | 5 % 4 | 1 |
| ^ | возведение в степень (вычисляется слева направо) | 2.0 ^ 3.0 | 8 |
| |/ | квадратный корень | |/ 25.0 | 5 |
| ||/ | кубический корень | ||/ 27.0 | 3 |
| ! | факториал (устаревший оператор, его заменяет функция factorial()) | 5 ! | 120 |
| !! | факториал в префиксной форме (устаревший оператор, его заменяет функция factorial()) | !! 5 | 120 |
| @ | модуль числа (абсолютное значение) | @ -5.0 | 5 |
| & | битовый AND | 91 & 15 | 11 |
| | | битовый OR | 32 | 3 | 35 |
| # | битовый XOR | 17 # 5 | 20 |
| ~ | битовый NOT | ~1 | -2 |
| << | битовый сдвиг влево | 1 << 4 | 16 |
| >> | битовый сдвиг вправо | 8 >> 2 | 2 |

* Битовые операторы работают только с целочисленными типами данных и с битовыми строками bit и bit varying, как показано в [Таблице 9.13](https://postgrespro.ru/docs/postgresql/11/functions-bitstring#FUNCTIONS-BIT-STRING-OP-TABLE).
* В [Таблице 9.5](https://postgrespro.ru/docs/postgresql/11/functions-math#FUNCTIONS-MATH-FUNC-TABLE) перечислены все существующие математические функции. Сокращение dp в ней обозначает тип double precision (плавающее с двойной точностью). Многие из этих функций имеют несколько форм с разными типами аргументов. За исключением случаев, где это указано явно, любая форма функции возвращает результат того же типа, что и аргумент. Функции, работающие с данными double precision, в массе своей используют реализации из системных библиотек сервера, поэтому точность и поведение в граничных случаях может зависеть от системы сервера.
* **Таблица 9.5. Математические функции**

| **Функция** | **Тип результата** | **Описание** | **Пример** | **Результат** |
| --- | --- | --- | --- | --- |
| abs(***x***) | тип аргумента | модуль числа (абсолютное значение) | abs(-17.4) | 17.4 |
| cbrt(dp) | dp | кубический корень | cbrt(27.0) | 3 |
| ceil(dp или numeric) | тип аргумента | ближайшее целое, большее или равное аргументу | ceil(-42.8) | -42 |
| ceiling(dp или numeric) | тип аргумента | ближайшее целое, большее или равное аргументу (равнозначно ceil) | ceiling(-95.3) | -95 |
| degrees(dp) | dp | преобразование радианов в градусы | degrees(0.5) | 28.6478897565​412 |
| div(*y* numeric, *x* numeric) | numeric | целочисленный результат *y*/*x* | div(9,4) | 2 |
| exp(dp или numeric) | тип аргумента | экспонента | exp(1.0) | 2.7182818284​5905 |
| factorial(bigint) | numeric | факториал | factorial(5) | 120 |
| floor(dp или numeric) | тип аргумента | ближайшее целое, меньшее или равное аргументу | floor(-42.8) | -43 |
| ln(dp или numeric) | тип аргумента | натуральный логарифм | ln(2.0) | 0.6931471805​59945 |
| log(dp или numeric) | тип аргумента | логарифм по основанию 10 | log(100.0) | 2 |
| log(*b* numeric, *x* numeric) | numeric | логарифм по основанию *b* | log(2.0, 64.0) | 6.0000000000 |
| mod(*y*, *x*) | зависит от типов аргументов | остаток от деления *y*/*x* | mod(9,4) | 1 |
| pi() | dp | константа «π» | pi() | 3.1415926535​8979 |
| power(*a* dp, *b* dp) | dp | *a* возводится в степень *b* | power(9.0, 3.0) | 729 |
| power(*a* numeric, *b* numeric) | numeric | *a* возводится в степень *b* | power(9.0, 3.0) | 729 |
| radians(dp) | dp | преобразование градусов в радианы | radians(45.0) | 0.7853981633​97448 |
| round(dp или numeric) | тип аргумента | округление до ближайшего целого | round(42.4) | 42 |
| round(*v* numeric, *s* int) | numeric | округление *v* до *s* десятичных знаков | round(42.4382, 2) | 42.44 |
| scale(numeric) | integer | масштаб аргумента (число десятичных цифр в дробной части) | scale(8.41) | 2 |
| sign(dp или numeric) | тип аргумента | знак аргумента (-1, 0, +1) | sign(-8.4) | -1 |
| sqrt(dp или numeric) | тип аргумента | квадратный корень | sqrt(2.0) | 1.4142135623​731 |
| trunc(dp или numeric) | тип аргумента | округление к нулю | trunc(42.8) | 42 |
| trunc(*v* numeric, *s* int) | numeric | округление к 0 до *s* десятичных знаков | trunc(42.4382, 2) | 42.43 |
| width\_bucket(*operand* dp, *b1* dp, *b2* dp, *count* int) | int | возвращает номер группы, в которую попадёт *operand* в гистограмме с числом групп *count* равного размера, в диапазоне от *b1* до *b2*; возвращает 0 или *count*+1, если операнд лежит вне диапазона | width\_bucket(5.35, 0.024, 10.06, 5) | 3 |
| width\_bucket(*operand* numeric, *b1* numeric, *b2* numeric, *count* int) | int | возвращает номер группы, в которую попадёт *operand* в гистограмме с числом групп *count* равного размера, в диапазоне от *b1* до *b2*; возвращает 0 или *count*+1, если операнд лежит вне диапазона | width\_bucket(5.35, 0.024, 10.06, 5) | 3 |
| width\_bucket(*operand* anyelement, *thresholds* anyarray) | int | возвращает номер группы, в которую попадёт *operand* (группы определяются нижними границами, передаваемыми в *thresholds*); возвращает 0, если операнд оказывается левее нижней границы; массив *thresholds* ***должен быть отсортирован*** по возрастанию, иначе будут получены неожиданные результаты | width\_bucket(now(), array['yesterday', 'today', 'tomorrow']::timestamptz[]) | 2 |

* В [Таблице 9.6](https://postgrespro.ru/docs/postgresql/11/functions-math#FUNCTIONS-MATH-RANDOM-TABLE) перечислены все функции для генерации случайных чисел.
* **Таблица 9.6. Случайные функции**

| **Функция** | **Тип результата** | **Описание** |
| --- | --- | --- |
| random() | dp | случайное число в диапазоне 0.0 <= x < 1.0 |
| setseed(dp) | void | задаёт отправную точку для последующих вызовов random() (значение между -1.0 и 1.0, включая границы) |

* Характеристики значений, возвращаемых функцией random() зависят от системы. Для применения в криптографии они непригодны; альтернативы описаны в [pgcrypto](https://postgrespro.ru/docs/postgresql/11/pgcrypto" \o "F.25. pgcrypto).
* Наконец, в [Таблице 9.7](https://postgrespro.ru/docs/postgresql/11/functions-math#FUNCTIONS-MATH-TRIG-TABLE) перечислены все имеющиеся тригонометрические функции. Все эти функции принимают аргументы и возвращают значения типа double precision. У каждой функции имеются две вариации: одна измеряет углы в радианах, а вторая — в градусах.
* **Таблица 9.7. Тригонометрические функции**

| **Функции (в радианах)** | **Функции (в градусах)** | **Описание** |
| --- | --- | --- |
| acos(***x***) | acosd(***x***) | арккосинус |
| asin(***x***) | asind(***x***) | арксинус |
| atan(***x***) | atand(***x***) | арктангенс |
| atan2(***y***, ***x***) | atan2d(***y***, ***x***) | арктангенс ***y***/***x*** |
| cos(***x***) | cosd(***x***) | косинус |
| cot(***x***) | cotd(***x***) | котангенс |
| sin(***x***) | sind(***x***) | синус |
| tan(***x***) | tand(***x***) | тангенс |

**SELECT**

director,

movie\_title,

**10** - rating **AS** rating\_100

**FROM** **sql**.kinopoisk

**SELECT** \*

**FROM** **sql**.kinopoisk

**WHERE** position = **1, 10**

**SELECT** /\*выбор\*/

**year,** /\*столбец year\*/

**movie\_title,** /\*столбец movie\_title\*/

**director** /\*столбец director\*/

**FROM** **sql.kinopoisk** /\*из таблицы sql.kinopoisk\*/

**WHERE** **(rating > 8**.**5** **AND** **year <** **2000**) /\*при условии, что рейтинг больше 8.5 и год создания до 2000\*/

**OR** **year >=** **2000** /\*или год создания — 2000 и позднее\*/