**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\*/

**LIKE**

Предположим, мы не знаем точно, какое текстовое значение ищем.

В таком случае нам поможет оператор LIKE.

**SELECT** \*

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

**WHERE** movie\_title **LIKE** **'А%'**

Знак процента (%) в примере показывает, что после A встречается ноль и более символов. Вы можете использовать % в любом месте внутри строки.

Например, movie\_title LIKE '%а%б%' выведет все фильмы, в названии которых встречается строчная буква *а*, а где-то после неё — *б*.

Также в текстовых строках используется знак подчёркивания (\_) — он заменяет ровно один любой символ.

Напишите запрос, чтобы вывести название и год выпуска в прокат тех фильмов, которые были сняты режиссёром по имени Дэвид (то есть значение в поле director начинается с 'Дэвид') и имеют рейтинг больше 8.

**SELECT**

**year,**

**movie\_title,**

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

**WHERE** **director** **LIKE** **'** Дэвид **%' AND** **rating > 8**

Для пустых значений есть специальное обозначение — NULL.

**SELECT** \*

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

**WHERE** overview = **NULL**

Как вы заметили, вывод пустой, хотя мы точно видели фильмы с отсутствующим описанием.

А теперь попробуйте изменить условие на overview IS NULL.

Если вы всё сделали верно, то получили все фильмы, у которых в таблице отсутствует описание.

Вы можете добавить к условию уже изученное ключевое слово NOT, чтобы получилось overview IS NOT NULL, тогда в выводе вы увидите только фильмы с заполненным описанием.

**Важно!** NULL — это специальное значение. Если вы фильтруете столбец, в котором есть пустые значения, по любому условию, кроме IS NULL / IS NOT NULL, такие значения будут исключены из вывода.

**ORDER BY**

Когда мы говорили о последовательности столбцов в выводе, вы, вероятно, задались вопросом: «А в каком порядке выводятся строки?»

Порядок вывода строк может задаваться в настройках базы данных для каждой таблицы. Более того, этот порядок может быть не задан, и тогда от вывода к выводу он будет разным.

Чтобы задать порядок вывода строк в запросе, применим новое ключевое слово ORDER BY.

SELECT \*

FROM sql.kinopoisk

ORDER BY rating

Наш запрос с сортировкой по названию аналогичен такому:

**SELECT** \*

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

**ORDER** **BY** rating **ASC**

Здесь ASC — явное указание порядка сортировки по возрастанию (англ. *ascending*).

Для обратного порядка используется ключевое слово DESC (англ. *descending*).

Выведем названия, имена режиссёров и сценаристов, а также год выхода в прокат фильмов, выпущенных в СССР, и отсортируем результат по убыванию рейтинга.

**SELECT**

movie\_title,

director,

screenwriter,

year

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

**WHERE** country = 'СССР'

**ORDER** **BY** rating **DESC**

**Обратите внимание!** Ключевое слово ORDER BY идёт после применения всех условий в WHERE.

Напишите запрос, который выведет столбцы с названием фильма, его описанием и годом выхода в прокат. Оставьте только те фильмы, у которых рейтинг не ниже 8.2 и страна производства — не США. Отсортируйте вывод по году выхода фильма в порядке убывания.

SELECT

movie\_title,

overview,

year

FROM sql.kinopoisk

WHERE rating >= 8.2 and country NOT IN ('США')

ORDER BY year DESC

Также в ORDER BY можно указывать, где должны идти пустые значения — в начале или в конце.

Такая настройка порядка вывода задаётся с помощью ключевых слов NULLS FIRST / NULLS LAST.

**SELECT**

movie\_title,

rating,

overview,

year

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

**ORDER** **BY** overview

SELECT

movie\_title,

rating,

overview,

year

FROM sql.kinopoisk

ORDER BY overview NULLS FIRST

Получили список всех режиссёров и фильмов из ТОП-250, отсортированных по году выхода в прокат, а внутри года — по рейтингу в порядке убывания.

**SELECT**

director,

movie\_title

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

**ORDER** **BY** year, rating **DESC**

Напишите запрос, чтобы вывести названия всех фильмов (столбец Название фильма), у которых рейтинг выше 8.3 и страна производства — Франция. Отсортируйте по рейтингу в порядке убывания, далее — по году выхода в прокат (также в порядке убывания).

SELECT

movie\_title

FROM sql.kinopoisk

WHERE rating > 8.3 AND country = 'Франция'

ORDER BY rating DESC, year DESC

SELECT

movie\_title,

overview,

year

FROM sql.kinopoisk

WHERE rating >= 8.2 and country NOT IN ('США')

ORDER BY year DESC

Для упрощения работы с ORDER BY можно использовать не названия столбцов, а их **номера из вывода**.

**SELECT**

director,

movie\_title,

year

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

**ORDER** **BY** **1**, **3** **DESC**

ортировку по номеру столбца стоит использовать с осторожностью, поскольку при изменении вывода в SELECT всё может сбиться.

При добавлении новых столбцов в SELECT нужно проверить и при необходимости поправить номера столбцов в ORDER BY.

Ограничим вывод первыми десятью строками и сможем легко понять, какие данные хранятся в таблице, не утяжеляя результат.

**SELECT** \*

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

**LIMIT** **10**

Выведем ТОП-5 фильмов по рейтингу, сначала отсортировав их по убыванию, а потом оставив только верхние пять строк с помощью LIMIT.

**SELECT**

movie\_title,

rating

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

**ORDER** **BY** rating **DESC**

**LIMIT** **5**

**Обратите внимание!** Ключевое слово LIMIT используется в самом конце запроса.

Напишите запрос, который выводит информацию (Режиссёр, Название фильма и Актёры) по ТОП-20 самых старых (определяем по году выхода в прокат) фильмов из таблицы kinopoisk.

Выведите столбцы Режиссёр, Название фильма, Актёры.

**SELECT**

**director,**

**movie\_title,**

**actors,**

**/\* year\*/**

**FROM sql.kinopoisk**

**ORDER BY year ASC**

**LIMIT 20**

Если LIMIT «оставляет» указанное число первых строк из вывода, то OFFSET, наоборот, «обрезает» указанное число первых строк.

LIMIT и OFFSET можно использовать вместе, их порядок не важен.

**SELECT**

movie\_title,

rating

**year**

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

**ORDER** **BY** rating **DESC**

**OFFSET** **3** **LIMIT** **5**

Таким образом, LIMIT отсчитывает количество строк после указанной в OFFSET строки.

Напишите запрос, чтобы вывести названия фильмов, которые вышли в прокат после 1990 года и были сняты не в России. Из этого списка оставьте только те фильмы, которые занимают с 20 по 47 места в рейтинге. Отсортируйте результат по убыванию рейтинга фильмов.

SELECT

movie\_title

FROM sql.kinopoisk

WHERE year > 1990 AND country NOT IN ('Россия')

ORDER BY rating DESC

OFFSET 19 LIMIT 28

Напоследок напомним структуру простого запроса:

**SELECT**

столбец1 **AS** новое\_название,

столбец2,  
 столбец3

**FROM** таблица

**WHERE** (условие1 **OR** условие2)  
 **AND** условие3

**ORDER** **BY** сортировка1, сортировка2

**OFFSET** **1** **LIMIT** **2**

Какой режиссёр снял самый старый фильм в списке?

**SELECT**

**director,**

**movie\_title,**

**actors,**

**/\* year\*/**

**FROM sql.kinopoisk**

**ORDER BY year ASC**

**LIMIT 20**

В каком году был выпущен фильм, который занимает 111 строку в списке, отсортированном по рейтингу в порядке убывания?

**SELECT**

**director,**

**movie\_title,**

**actors,**

**year**

**FROM sql.kinopoisk**

**ORDER BY rating DESC**

**OFFSET 110 LIMIT 1**

Напишите запрос, который выводит столбцы «Название фильма» (movie\_title), «Режиссёр» (director), «Сценарист» (screenwriter), «Актёры» (actors). Оставьте только те фильмы, у которых:

* рейтинг между 8 и 8.5 (включительно) ИЛИ год выхода в прокат до 1990;
* есть описание;
* название начинается не с буквы 'Т';
* название состоит ровно из 12 символов.

Оставьте только топ-7 по рейтингу.

**SELECT**

**movie\_title as "Название фильма",**

**director as "Режиссёр",**

**screenwriter as "Сценарист",**

**actors as "Актёры"**

**FROM sql.kinopoisk**

**WHERE (rating BETWEEN 8 and 8.5 or year < 1990) and overview IS NOT null and movie\_title like ('\_\_\_\_\_\_\_\_\_\_\_\_')**

**ORDER BY rating DESC**

**limit 7**

В предложении можно использовать следующие операторы WHERE:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Equal | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_equal_to) |
| > | Greater than | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_greater_than) |
| < | Less than | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_less_than) |
| >= | Greater than or equal | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_greater_than2) |
| <= | Less than or equal | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_less_than2) |
| <> | Not equal. **Note:** In some versions of SQL this operator may be written as != | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_not_equal_to) |
| BETWEEN | Between a certain range | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_between) |
| LIKE | Search for a pattern | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_like) |
| IN | To specify multiple possible values for a column |  |