Работа с MySQL

MySQL это СУБД (система управления базами данных), которая довольно проста в работе даже для начинающего. Она не потребляет много ресурсов и может быть использована даже на слабом железе. Она очень часто используется в связке с PHP и о ней мы поговорим в рамках этого курса.

Разобраться с MySQL сможет любой начинающий разработчик. Эта СУБД имеет большую популярность в среде программистов и используется повсеместно.

Для начала работы с MySQL нам нужно определиться с несколькими терминами:

* **база данных** — это такая сущность, в которой находятся таблицы;
* **таблица** — это сущность из столбцов и строк, в которой лежат данные;
* **строка** — это одна запись в таблице;
* **столбец** — это столбец в таблице :)

Как работать с базой данных MySQL

Если вы всё сделали как в статье [Установка и настройка OpenServer](https://php.zone/kurs-php-dlya-nachinayushih/ustanovka-i-nastroyka-openserver), то консольный клиент для работы с MySQL лежит по пути:

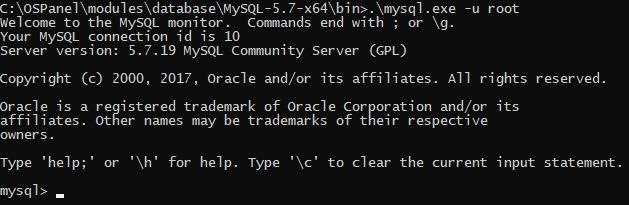
**C**:\**OSPanel**\**modules**\**database**\**MySQL-5**.7-x64\**bin**\**mysql**.exe

Вместо MySQL-5.7-x64 нужно подставить версию, указанную в настройках OpenServer во вкладке "Модули".

Нужно открыть консоль и запустить эту программу с флажком **-u root**

**C**:\**OSPanel**\**modules**\**database**\**MySQL-5**.7-x64\**bin**\**mysql**.exe **-u** **root**

В результате запустится MySQL-клиент. Он подключён к MySQL-серверу, который был запущен при старте OpenServer-a.

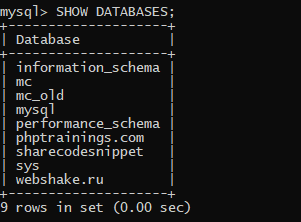
[](https://php.zone/uploads/img/111aad024ad1f46bc744bc04e387a301cdf1a70da285098126b55b40a911a853.png)

Параметр -u расшифровывается как user. То есть это флажок для указания пользователя, под которым нужно подключиться к серверу. root - это самый главный пользователь в MySQL. Он создаётся при установке сервера и по умолчанию у него нет пароля.

Через этот консольный клиент мы можем отправлять различные команды СУБД. Давайте выполним команду, которая выводит все базы данных, созданные на этом сервере.

**SHOW** **DATABASES**;

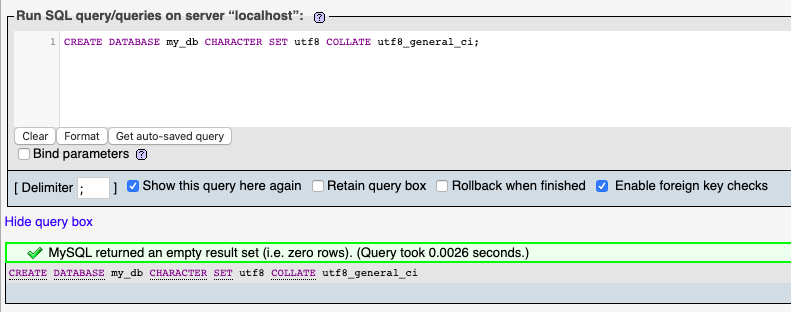
В ответ мы получим красиво оформленный список баз. У вас их будет меньше, чем у меня, потому что я их уже у себя надобавлял.

[](https://php.zone/uploads/img/3fe91686030348a973ba85bdfd27ed84b5e8f7b307c1e6a289bfc72eed1d5cbb.png)

## Создание баз данных

Для создания новой базы данных с именем **my\_db** и кодировкой utf8 следует использовать команду:

**CREATE** **DATABASE** my\_db CHARACTER **SET** utf8 **COLLATE** utf8\_general\_ci;

При успешном выполнении команды будет выведено сообщение:  
[](https://php.zone/uploads/img/202004110b1963536440e382e8c26b0bc8b035ec2fa8e2cde0c8cd458f955261d8da3338.png)

Чтобы теперь работать с этой базой нужно выполнить:

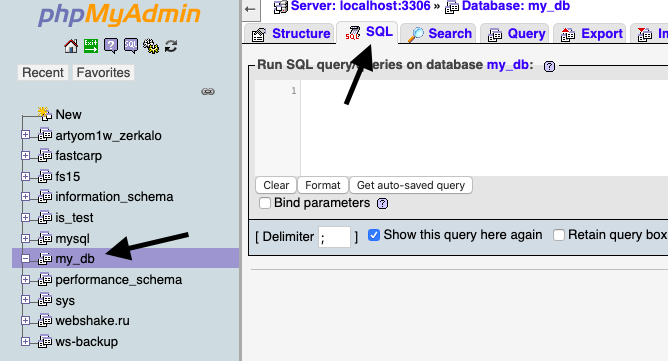
[Онлайн тесты](https://php.zone/tests)

* [Тест на знание основ HTML](https://php.zone/test-html)
* [Тест на знание основ PHP](https://php.zone/test-php)
* [Тест на знание ООП в PHP](https://php.zone/test-oop-v-php)

[ВСЕ ТЕСТЫ](https://php.zone/tests)

**USE** **my\_db**;

Если вы работаете в панели phpMyAdmin, то достаточно просто выбрать базу my\_db в списке слева, а затем перейти во вкладку SQL. Теперь запросы будут выполняться непосредственно для этой базы данных.

[](https://php.zone/uploads/img/20200411d3edf7acfac051205bc645dee342c0bc010b209cd17f227d7deaf41cd98beeb6.png)

Теперь поработаем с самой базой.

[Онлайн курсы](https://php.zone/courses)

* [Курс HTML для начинающих](https://php.zone/kurs-html-dlya-nachinayushih)
* [Курс PHP для начинающих](https://php.zone/kurs-php-dlya-nachinayushih)
* [Курс MySQL для начинающих](https://php.zone/php-i-mysql-s-nulya)
* [Курс ООП в PHP](https://php.zone/oop-v-php-prodvinutyj-kurs)

[ВСЕ КУРСЫ](https://php.zone/courses)

### **Создание таблиц**

Чтобы в базе создать таблицу используется команда **CREATE TABLE**. При этом нужно описать столбцы, которые будут у этой таблицы. Давайте создадим табличку, в которой будут храниться пользователи. У каждого пользователя будет идентификатор, email и имя:

**CREATE** **TABLE** `users` (

`id` INT **NOT** NULL AUTO\_INCREMENT ,

`email` VARCHAR(255) **NOT** NULL ,

`name` VARCHAR(255) **NOT** NULL ,

PRIMARY **KEY** (`id`),

**UNIQUE** (`email`)

) **ENGINE**=**InnoDB** **DEFAULT** **CHARSET**=utf8;

В нашем примере id, name, email - это имена столбцов.  
INT, VARCHAR(255) - типы данных (после этого абзаца идет табличка со всеми типами данных, доступных в MySQL)).  
NOT NULL означает, что поле не может иметь значение NULL (или другими словами, оно всегда должно иметь какое-то значение).  
Помимо этого для таблицы можно задать PRIMARY KEY (первичный ключ) - это столбец, значения в котором будут всегда уникальными (к примеру - ID записи).  
AUTO\_INCREMENT можно указать для первичного ключа, тогда при добавлении новых записей это поле будет заполняться автоматически. При добавлении первой записи он будет равен 1, потом 2 и т.д.  
UNIQUE(имя столбца) - создание уникального индекса (об индексах мы поговорим позже). Уникальный индекс накладывает ограничения на уникальность значений в рамках столбца email. Это мы делаем для того, чтобы у двух разных пользователей нельзя было задать одинаковый email.

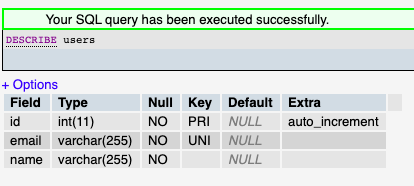
#### **Типы данных**

| **Обозначение** | **Занимаемый объем (байт)** | **Область применения** |
| --- | --- | --- |
| TINYINT, BOOL | 1 | При использовании в формате представления без знака позволяет хранить значения от 0 до 255; в противном случае — от -128 до 127. В будущем должен быть предусмотрен новый логический тип, но до сих пор для представления логических значений использовался тип данных TINYINT, т.е. BOOL синоним TINYINT(1) |
| SMALLINT | 2 | Целое число в диапазоне от -32768 до 32767 |
| MEDIUMINT | 3 | Целое число в диапазоне от -8388608 до 8388607 |
| INT, INTEGER | 4 | Целое число в диапазоне от -2e32 до 2e32 - 1 |
| BIGINT | 8 | Целое число в диапазоне от -2e64 до 2e64 - 1 |
| FLOAT | 4 | Число с плавающей точкой одинарной точности |
| DOUBLE | 8 | Число с плавающей точкой двойной точности |
| DECIMAL | Произвольное, в зависимости от точности | Распакованное число с плавающей точкой, которое хранится в таком же формате, как CHAR. Используется для представления небольших десятичных значений, таких как денежные суммы |
| DATE | 3 | Отображается в формате YYYY-MM-DD |
| DATETIME, TIMESTAMP | 8 | Отображается в формате YYYY-MM-DD HH:MM:SS |
| TIME | 3 | Отображается в формате HHH:MM:SS, где HHH — значение от -838 до 838. Это позволяет применять значения типа time для представления продолжительности времени между двумя событиями |
| YEAR | 1 | Отображается в формате YYYY, который представляет значения от 1901 до 2155 |
| CHAR | N байт | Строка постоянной длины. Строка, имеющая длину меньше объявленной, дополняется справа пробелами. Значение N должно быть меньше или равно 255 |
| VARCHAR | N байт | Строка переменной длины. |
| BINARY | N байт | Сохраняет байтовые строки |
| TINYBLOB, TINYTEXT | до 255 | Сохраняет строки, операции сортировки и сравнения данных типа blob выполняются с учетом регистра; операции с данными типа text — без учета регистра |
| BLOB, TEXT | до 64 Кбайт | Длинные строки |
| MEDIUMBLOB, MEDIUMTEXT | до 16 Мбайт | Длинные строки |
| LONGBLOB, LONGTEXT | до 4 Гбайт | Длинные строки |
| ENUM(value1, ..., valueN) | 1 или 2 | Коллекция значений (65536 возможных значений) |
| SET(value1, ..., valueN) | до 8 | Коллекция значений (64 возможных значений) |

### **Описание структуры таблицы**

### Чтобы посмотреть структуру новой таблицы введите:

**DESCRIBE** **users**;

Результат:  
[](https://php.zone/uploads/img/20200411a29eb525cd6fb268d0fdea47a3d86cabb99a686985c80d5d6a2894e3dc3d01cd.png)

После создания таблиц их можно изменять, вот несколько полезных команд для изменения их структуры:

### **Переименование таблицы**

**ALTER** **TABLE** имя\_таблицы **RENAME** новое\_имя\_таблицы

### **Добавление столбца**

**ALTER** **TABLE** имя\_таблицы **ADD** имя\_столбца тип\_данных

### **Переименование столбца**

**ALTER** **TABLE** имя\_таблицы **RENAME** **COLUMN** старое\_имя\_столбца **TO** новое\_имя\_столбца;

### **Удаление столбца**

**ALTER** **TABLE** имя\_таблицы **DROP** имя\_столбца

### **Удаление всей таблицы**

**DROP** **TABLE** имя таблицы;

Это основные и самые частоиспользуемые команды. Если нужно что-то еще - гуглится без проблем.

В этом уроке мы разобрали команды для работы со структурой таблиц. В следующем уроке мы научимся работать с данными этих таблиц.

Судя по тому, что вы читаете этот текст, вы дочитали эту статью до конца. Если она вам понравилась, поделитесь, пожалуйста, с помощью кнопок ниже. Спасибо за вашу поддержку!

## Добавление данных в таблицу

Для добавления данных используется запрос INSERT. Синтаксис:

**INSERT** **INTO** имя\_таблицы ([столбцы]) **VALUES** ([значения]);

Давайте вставим в нашу таблицу users несколько пользователей:

**INSERT** **INTO** **users** (email, **name**) **VALUES**('x1@php.zone', 'Иван');

**INSERT** **INTO** **users** (email, **name**) **VALUES**('x2@php.zone', 'Ольга');

**INSERT** **INTO** **users** (email, **name**) **VALUES**('x3@php.zone', 'Александр');

**INSERT** **INTO** **users** (email, **name**) **VALUES**('x4@php.zone', 'Елена');

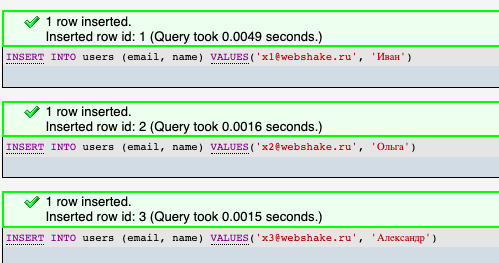
**INSERT** **INTO** **users** (email, **name**) **VALUES**('x5@php.zone', 'Ольга');

**INSERT** **INTO** **users** (email, **name**) **VALUES**('x6@php.zone', 'Матвей');

**INSERT** **INTO** **users** (email, **name**) **VALUES**('x7@php.zone', 'Иван');

**INSERT** **INTO** **users** (email, **name**) **VALUES**('x8@php.zone', 'Пётр');

**INSERT** **INTO** **users** (email, **name**) **VALUES**('x9@php.zone', 'Иван');

После этого мы увидим информацию об успешно выполненных запросах на добавление записей  
[](https://php.zone/uploads/img/2020041174574ecabc30ad032b3a2908734c1d68c53e18416cb8804b053d6253bf6519d5.png)

[Онлайн тесты](https://php.zone/tests)

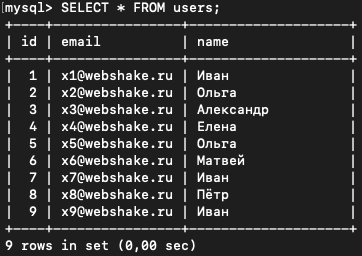
* [Тест на знание основ HTML](https://php.zone/test-html)
* [Тест на знание основ PHP](https://php.zone/test-php)
* [Тест на знание ООП в PHP](https://php.zone/test-oop-v-php)

[ВСЕ ТЕСТЫ](https://php.zone/tests)

## Запрос данных из таблицы

Данные из таблиц получают с помощью запроса SELECT.  
Самый простой запрос, позволяющий просмотреть записи таблицы, выглядит так:

**SELECT** \* **FROM** **users**;

Звездочка означает взять все столбцы таблицы.  
[](https://php.zone/uploads/img/20200411b46e3d1167607f7d24927559374bbc6f15bb436f7d7061323dfd40943622c2da.png)

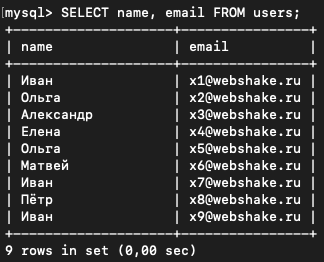
Если нам не нужен в результате id, можно написать так:

[Алгоритмические задачи](https://php.zone/problems)

* [Привет, мир!](https://php.zone/problems/1)
* [Развернуть строку](https://php.zone/problems/2)
* [Чётные числа](https://php.zone/problems/3)
* [Числа Фибоначчи](https://php.zone/problems/4)

[ВСЕ ЗАДАЧИ](https://php.zone/problems)

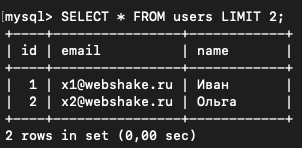
**SELECT** **name**, email **FROM** **users**;

[](https://php.zone/uploads/img/20200411a621eb12747021651f5f1a00c9b362eaf97c20563f65a81389d12867f191fb32.png)

## LIMIT и OFFSET

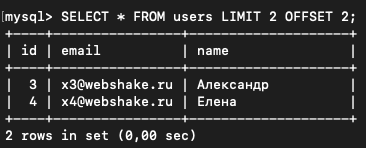
Для ограничения выдачи используется ключевое слово LIMIT. Выведем только первые 2 записи.

**SELECT** \* **FROM** **users** **LIMIT** 2;

[](https://php.zone/uploads/img/202004118aa67eb6aa118975c617127962c02388e95a1067f5f0e3c1e0a3d57dcf2062e8.png)

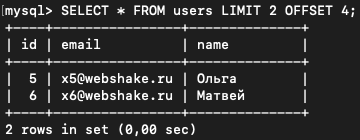
Для того, чтобы выполнить смещение на 2 записи, нужно использовать ключевое слово OFFSET.

**SELECT** \* **FROM** **users** **LIMIT** 2 **OFFSET** 2;

[](https://php.zone/uploads/img/2020041153a7af4f735e23acbc4b98dcbb1ffd4546cd811ca88547580e1e495ac7f568b1.png)  
Таким образом мы пропустили 2 первые записи и ограничили выборку двумя следующими записями.

Чтобы вывести следующие 2 нужно увеличить смещение еще на 2, изменив значение OFFSET:

**SELECT** \* **FROM** **users** **LIMIT** 2 **OFFSET** 4;

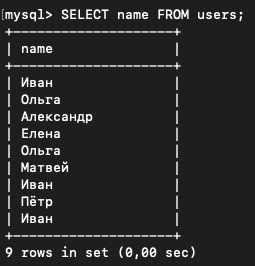
[](https://php.zone/uploads/img/202004113315ac6cd9e7554986708e584cf82a15193a864fe284ce5e80e1241dce6e69f6.png)

С помощью OFFSET и LIMIT реализуется пагинация записей.

## DISTINCT

Давайте сделаем запрос только с именами:

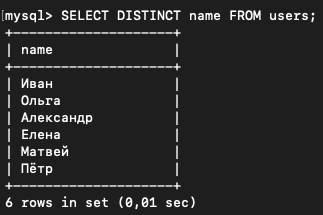
**SELECT** **name** **FROM** **users**;

[](https://php.zone/uploads/img/2020041169a2c41d060b75b55d909d8034019ac6b3bd74c86a124030162998745b89ff18.png)

Сейчас мы видим, что в таблице у нас несколько повторяющихся имён: Иван и Ольга.

Чтобы оставить только разные имена в результате запроса используется ключевое слово DISTINCT (переводится с английского как "индивидуальный", "отдельный").

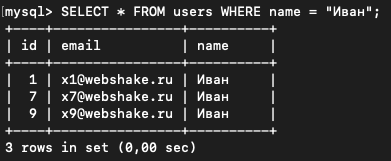
**SELECT** **DISTINCT** **name** **FROM** **users**;

Результат:  
[](https://php.zone/uploads/img/20200411ed54e4b87a459cbe77080400f16c2615b8e46d271bb15a72aaf5cbe64d7ae464.png)

## Условия в запросах

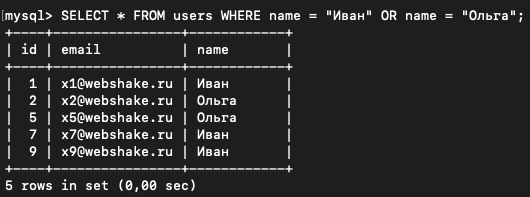
Если нужно отфильтровать значения по какому-то условию, используется WHERE. Например:

**SELECT** \* **FROM** **users** **WHERE** **name** = "Иван";

[](https://php.zone/uploads/img/20200411b098bdf32f0bbd122558acccfcc53bc7b5ef9aefc55b323acad33b44d7ffae26.png)

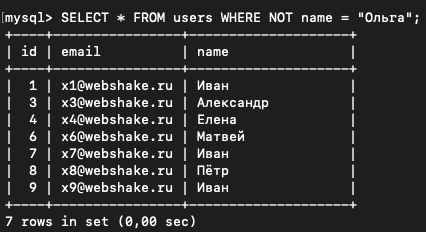
Используя операторы AND и OR можно задать несколько условий.

**SELECT** \* **FROM** **users** **WHERE** **name** = "Иван" **OR** **name** = "Ольга";

[](https://php.zone/uploads/img/20200411affc6f746349fbbdeb9d20a0d644cbce3d5d894b8ec3123486ea1b69fb77f992.png)

С помощью ключевого слова NOT можно инвертировать условие:

**SELECT** \* **FROM** **users** **WHERE** **NOT** **name** = "Ольга";

[](https://php.zone/uploads/img/20200411c0434a4ba5ad384553defe582269d8003896f57db077a4c3ddb330850e7ace09.png)

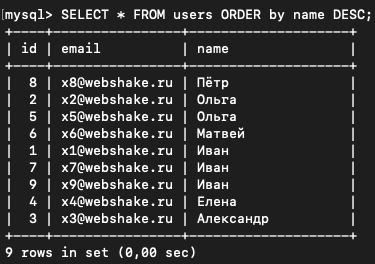
## Сортировка

Для сортировки значений используется ключевое слово ORDER BY. По умолчанию ORDER BY делает сортировку в порядке возрастания (ASC).

**SELECT** \* **FROM** **users** **ORDER** **by** **name**;

Если нужно отсортировать в обратном порядке нужно добавить ключевое слово DESC:

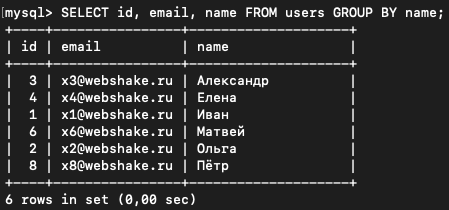
**SELECT** \* **FROM** **users** **ORDER** **by** **name** **DESC**;

[](https://php.zone/uploads/img/20200411c7ea3da04701b7342acf3f33baca469e924278edda66a2312d72e929d78eb47d.png)

## Группировка

Результаты запроса можно сгруппировать по одному или нескольким столбцам. В итоге строки, содержащие одинаковые по группируемым столбцам значения, как бы "схлопнутся" в агрегированные значения. Пример такого запроса:

**SELECT** **id**, email, **name** **FROM** **users** **GROUP** **BY** **name**;

[](https://php.zone/uploads/img/202004118c15f153ce8d78bd5f6472bffe248ec8a819e1d43849a63f2e991be191073e64.png)

В результате мы видим, что строки с одинаковыми именами схлопнулись в одну. Но вопрос в том, почему, например, имени "Иван" в выдаче соответствуют именно id=1 и email=x1@php.zone. Ответ: это просто случайность! Если мы видим перед собой "сгруппированную" строку, рядом с которой выводятся поля, по которым группировка не выполнялась, то в эти поля попадет случайное значение одной из сгруппированных строк.

То есть вот у нас было 3 строки с именем "Иван".

| 1 | x1@php.zone | Иван |

| 7 | x7@php.zone | Иван |

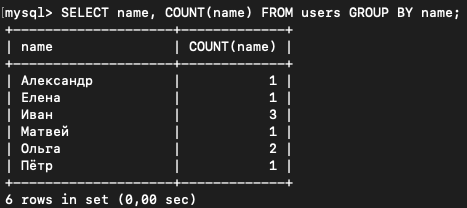
| 9 | x9@php.zone | Иван |

После группировки по полю "name" в результате нашего запроса в столбце name всегда будет "Иван". А вот по остальным столбцам, по которым группировка не производилась, будет случайное значение одной из этих строк. Это очень важно запомнить, что это значение случайное, и в другой раз там может быть уже другое.

### **Агрегирующие функции**

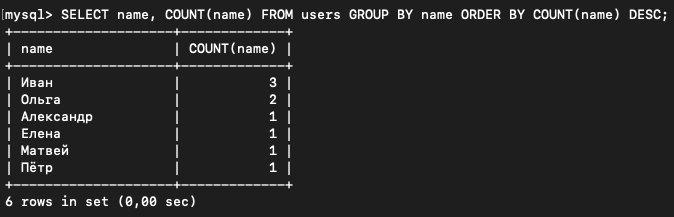
Вместе с использованием GROUP BY над сгруппированными данными можно выполнять агрегирующие функции. Одна из самых часто используемых функций - COUNT, она считает число записей.  
Давайте применим её к нашей выдаче чтобы узнать, какие имена сколько раз встречаются.

**SELECT** **name**, **COUNT**(**name**) **FROM** **users** **GROUP** **BY** **name**;

[](https://php.zone/uploads/img/20200411a0dadb18f25f57b2c22b87a72969e7679f7a256b7bd5d8be2dc42ca5efe4440b.png)

Агрегирующие функции можно также использовать внутри ORDER BY. К примеру, можно отсортировать имена по количеству вхождений по убыванию.

**SELECT** **name**, **COUNT**(**name**) **FROM** **users** **GROUP** **BY** **name** **ORDER** **BY** **COUNT**(**name**) **DESC**;

[](https://php.zone/uploads/img/202004117a6dce224c42cd6a8e44f1e566a6211d1101cc87c9cf0862704b144387fc0513.png)

Другие наиболее часто используемые агрегирующие функции:

| **Функция** | **Действие, выполняемое над сгруппированными данными** |
| --- | --- |
| COUNT() | Подсчет количества строк |
| SUM() | Подсчет суммы значений |
| AVG() | Вычисление среднего значения |
| MIN() | Определение минимального значения |
| MAX() | Определение максимального значения |

## HAVING

А если мы захотим вывести только имена, встречающиеся в базе более одного раза? Давайте попробуем сделать это с помощью WHERE:

**SELECT** **name**, **COUNT**(**name**) **FROM** **users** **WHERE** **COUNT**(**name**)>1 **GROUP** **BY** **name** **DESC**;

В ответ мы получим ошибку:

Invalid use of group function

Так произошло потому что группировка выполняется после WHERE, таким образом агрегирующей функции внутри WHERE просто ещё не с чем работать, отсюда и ошибка.

Как нам быть? К счастью, для фильтрации строк ПОСЛЕ выполнения группировки есть ключевое слово HAVING.

Теперь сделаем то, что хотели.

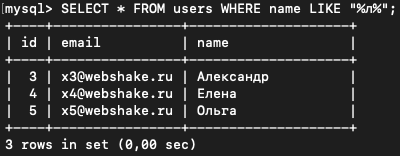
**SELECT** **name**, **COUNT**(**name**) **FROM** **users** **GROUP** **BY** **name** **HAVING** **COUNT**(**name**)>1;

[](https://php.zone/uploads/img/202004119be824a94b9dcadecfaba4d14314656d8897e9b5758ec846a22a423eac12b102.png)

## Поиск по шаблону

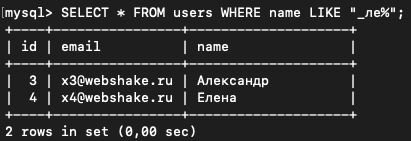
Чтобы искать данные по шаблону нужно использовать ключевое слово LIKE. Оно используется в связке со специальными символами - **%** и **\_**. **%** является шаблоном для ни одного, одного или нескольких любых символов. **\_** представляет собой шаблон для одного любого символа. Давайте лучше пример. Найдем все записи, у которых в значении поля name есть буква **л**:

**SELECT** \* **FROM** **users** **WHERE** **name** **LIKE** "%л%";

[](https://php.zone/uploads/img/20200411050a3c6228762cac3252501cb57e2f5653d27b67972b10d5215388f3d69033a4.png)

Или начинающиеся с любой буквы, дальше имеющие сочетание "ле", и заканчивающиеся как угодно:

**SELECT** \* **FROM** **users** **WHERE** **name** **LIKE** "\_ле%";

[](https://php.zone/uploads/img/20200411255dca616cbe3424b51501a2952aa7e967054a4259ec430f2a2316a574726838.png)

На этом с выборкой значений пока остановимся. Хотя это еще не все возможности, лучше будет сначала закрепить эти основы. А пока рассмотрим запросы для изменения данных в таблицах.

## Изменение данных

Изменить данные в таблице можно с помощью запроса UPDATE. В запросе UPDATE так же можно использовать условия используя ключевое слово WHERE. А еще в нем присутствует слово SET, с помощью которого определяются новые значения.

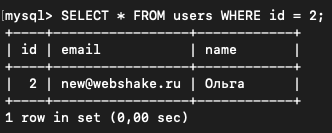
Обновим почту у пользователя с id = 2:

**UPDATE** **users** **SET** email = "new@php.zone" **WHERE** **id** = 2;

[https://php.zone/uploads/img/2020041172ea6c12ed051bf0f68f885b129e67f64e038c1ebcde2d344529eb8dc3330be3.png](https://php.zone/uploads/img/2020041172ea6c12ed051bf0f68f885b129e67f64e038c1ebcde2d344529eb8dc3330be3.png)

Запросим теперь эту запись:

**SELECT** \* **FROM** **users** **WHERE** **id** = 2;

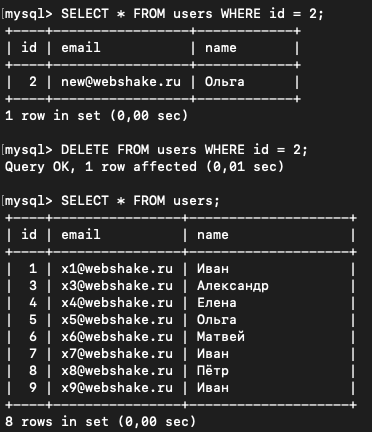
[](https://php.zone/uploads/img/202004110eec9ebf1d3de7566ba1c296c14b34b0a98a0cde2f472201c378080e39b951ce.png)

Как видим, всё успешно обновилось

## Удаление данных из базы

Для удаления используется запрос DELETE. И тут та же тема с WHERE:

**DELETE** **FROM** **users** **WHERE** **id** = 2;

[](https://php.zone/uploads/img/202004111dd9c6be33e9603f69623ec436fa23ff0549ac2b65bb028c70a1cfa3bc015a09.png)

Мы рассмотрели самые базовые запросы, однако этого уже достаточно для большинства простых задач. В дальнейших уроках будут более интересные и сложные запросы. А пока - закрепите материал домашним заданием.

Судя по тому, что вы читаете этот текст, вы дочитали эту статью до конца. Если она вам понравилась, поделитесь, пожалуйста, с помощью кнопок ниже. Спасибо за вашу поддержку!