MySQL Tutorial

MySQL is a widely used relational database management system (RDBMS).

MySQL is free and open-source.

MySQL is ideal for both small and large applications.

Introduction to MySQL

MySQL is a very popular open-source relational database management system (RDBMS).

What is MySQL?

- MySQL is a relational database management system
- MySQL is open-source
- MySQL is free
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, scalable, and easy to use
- MySQL is cross-platform
- MySQL is compliant with the ANSI SQL standard
- MySQL was first released in 1995
- MySQL is developed, distributed, and supported by Oracle Corporation
- MySQL is named after co-founder Monty Widenius's daughter: My

---

Who Uses MySQL?

- Huge websites like Facebook, Twitter, Airbnb, Booking.com, Uber, GitHub, YouTube, etc.
- Content Management Systems like WordPress, Drupal, Joomla!, Contao, etc.
- A very large number of web developers around the world

Show Data On Your Web Site

To build a web site that shows data from a database, you will need:

- An RDBMS database program (like MySQL)
- A server-side scripting language, like PHP
- To use SQL to get the data you want
- To use HTML / CSS to style the page

MySQL RDBMS

What is RDBMS?

RDBMS stands for Relational Database Management System.

RDBMS is a program used to maintain a relational database.

RDBMS is the basis for all modern database systems such as MySQL, Microsoft SQL Server, Oracle, and Microsoft Access.

RDBMS uses SQL queries to access the data in the database.

What is a Database Table?

A table is a collection of related data entries, and it consists of columns and rows.

A column holds specific information about every record in the table.

A record (or row) is each individual entry that exists in a table.

Look at a selection from the Northwind "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

The columns in the "Customers" table above are: CustomerID, CustomerName, ContactName, Address, City, PostalCode and Country. The table has 5 records (rows).

What is a Relational Database?

A relational database defines database relationships in the form of tables. The tables are related to each other - based on data common to each.

Look at the following three tables "Customers", "Orders", and "Shippers" from the Northwind database:

Customers Table

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

The relationship between the "Customers" table and the "Orders" table is the CustomerID column:

Orders Table

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10278	**5**	8	1996-08-12	**2**
10280	**5**	2	1996-08-14	**1**
10308	**2**	7	1996-09-18	**3**
10355	**4**	6	1996-11-15	**1**
10365	**3**	3	1996-11-27	**2**
10383	**4**	8	1996-12-16	**3**
10384	**5**	3	1996-12-16	**3**

The relationship between the "Orders" table and the "Shippers" table is the ShipperID column:

Shippers Table

ShipperID	ShipperName	Phone
1	Speedy Express	(503) 555-9831

2 United Package (503) 555-3199

3 Federal Shipping (503) 555-9931

MySQL SQL

What is SQL?

SQL is the standard language for dealing with Relational Databases.

SQL is used to insert, search, update, and delete database records.

How to Use SQL

The following SQL statement selects all the records in the "Customers" table:

Example

```
SELECT * FROM Customers;
```

Keep in Mind That...

- SQL keywords are NOT case sensitive: `select` is the same as `SELECT`

In this tutorial we will write all SQL keywords in upper-case.

Semicolon after SQL Statements?

Some database systems require a semicolon at the end of each SQL statement.

Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

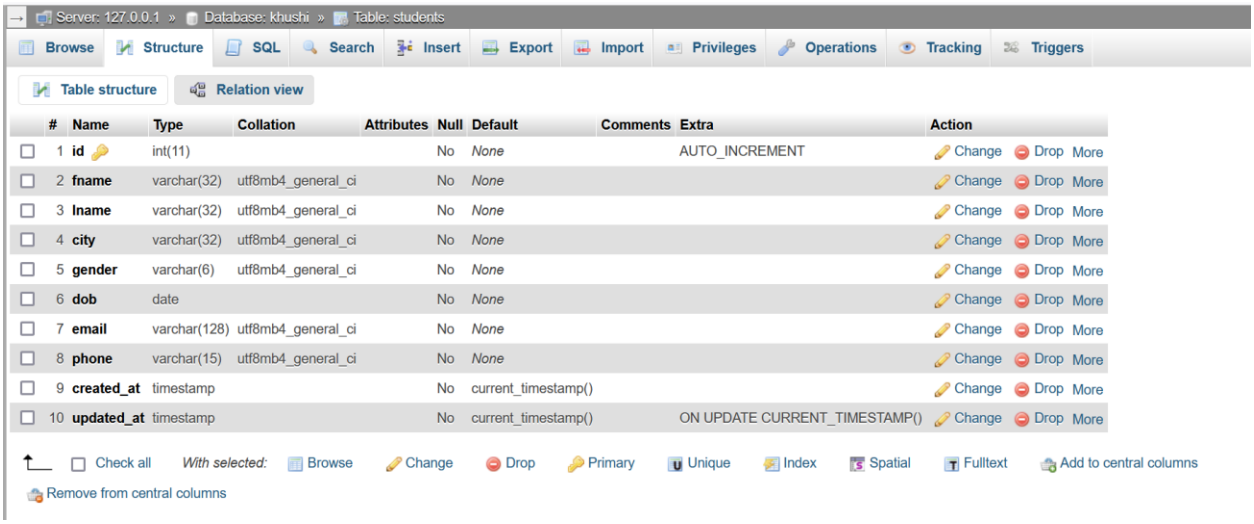
In this tutorial, we will use semicolon at the end of each SQL statement.

Some of The Most Important SQL Commands

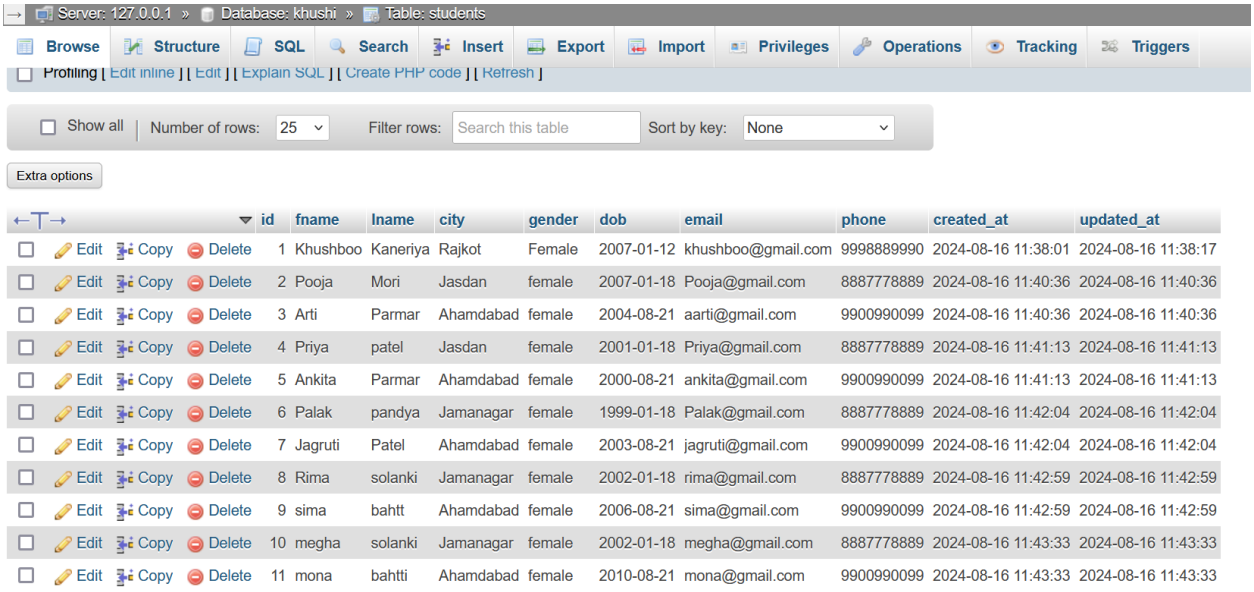
- `SELECT` - extracts data from a database
- `UPDATE` - updates data in a database
- `DELETE` - deletes data from a database
- `INSERT INTO` - inserts new data into a database
- `CREATE DATABASE` - creates a new database
- `ALTER DATABASE` - modifies a database
- `CREATE TABLE` - creates a new table
- `ALTER TABLE` - modifies a table

- `DROP TABLE` - deletes a table
- `CREATE INDEX` - creates an index (search key)
- `DROP INDEX` - deletes an index

Create Database and table as following



Add some data in students table



MySQL SELECT Statement

The MySQL SELECT Statement

The `SELECT` statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

SELECT Syntax

```
SELECT column1, column2, ...
FROM table_name;
```

```
SELECT id, fname, lname FROM students
```

Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

```
SELECT * FROM table_name;
```

```
SELECT * FROM students;
```

The MySQL SELECT DISTINCT Statement

The `SELECT DISTINCT` statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

SELECT DISTINCT Syntax

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

```
SELECT DISTINCT(city) FROM students
```

SELECT Example Without DISTINCT

The following SQL statement selects all (including the duplicates) values from the "Country" column in the "Customers" table:

```
SELECT COUNT(DISTINCT(city)) FROM students
```

MySQL WHERE Clause

The MySQL WHERE Clause

The `WHERE` clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

WHERE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Note: The `WHERE` clause is not only used in `SELECT` statements, it is also used in `UPDATE`, `DELETE`, etc.!

```
SELECT * FROM students WHERE city = 'Rajkot'
```

```
SELECT id, fname, lname, city FROM students WHERE city = 'Rajkot';
```

Text Fields vs. Numeric Fields

SQL requires single quotes around text values (most database systems will also allow double quotes).

However, numeric fields should not be enclosed in quotes:

```
SELECT id, fname, lname, city FROM students WHERE id > 5
```

```
SELECT id, fname, lname, city FROM students WHERE id = 5;
```

```
SELECT id, fname, lname, city FROM students WHERE id >= 5;
```

```
SELECT id, fname, lname, city FROM students WHERE id < 5;
```

```
SELECT id, fname, lname, city FROM students WHERE id <= 5;
```

```
SELECT id, fname, lname, city FROM students WHERE id <> 5;
```

```
SELECT id, fname, lname, city FROM students WHERE not id = 5;
```

MySQL AND, OR and NOT Operators

The MySQL AND, OR and NOT Operators

The `WHERE` clause can be combined with `AND`, `OR`, and `NOT` operators.

The `AND` and `OR` operators are used to filter records based on more than one condition:

- The `AND` operator displays a record if all the conditions separated by `AND` are TRUE.
- The `OR` operator displays a record if any of the conditions separated by `OR` is TRUE.

The `NOT` operator displays a record if the condition(s) is NOT TRUE.

AND Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

OR Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

NOT Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

```
SELECT * FROM students WHERE id = 1

SELECT * FROM students WHERE id = 1 and city = 'Surat';

SELECT * FROM students WHERE id = 1 or city = 'Surat';

SELECT * FROM students WHERE not city = 'Surat';
```

Combining AND, OR and NOT

You can also combine the `AND`, `OR` and `NOT` operators.

```
SELECT * FROM students WHERE id = 1 and city = 'Rajkot' or city = 'Surat';

SELECT * FROM students WHERE id = 1 and (city = 'Rajkot' or city = 'Surat');

SELECT * FROM students WHERE id = 1 and (not city = 'Rajkot' and not city = 'Surat');
```

MySQL ORDER BY Keyword

The MySQL ORDER BY Keyword

The `ORDER BY` keyword is used to sort the result-set in ascending or descending order.

The `ORDER BY` keyword sorts the records in ascending order by default. To sort the records in descending order, use the `DESC` keyword.

ORDER BY Syntax

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

```
SELECT * FROM students
```

```
SELECT * FROM students ORDER by (fname);
```

```
SELECT * FROM students ORDER by (fname) DESC;
```

ORDER BY Several Columns Example

```
SELECT * FROM students ORDER by fname, city;
```

```
SELECT * FROM students ORDER by fname asc, city DESC;
```

MySQL INSERT INTO Statement

The MySQL INSERT INTO Statement

The `INSERT INTO` statement is used to insert new records in a table.

INSERT INTO Syntax

It is possible to write the `INSERT INTO` statement in two ways:

1. Specify both the column names and the values to be inserted:

INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
VALUES (*value1*, *value2*, *value3*, ...);

2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the `INSERT INTO` syntax would be as follows:

INSERT INTO *table_name*
VALUES (*value1*, *value2*, *value3*, ...);

INSERT into students (fname, lname, city, gender, dob, email, phone) VALUES ('Dhruvisha', 'Bhatt', 'Junagadh', 'female', '2008-09-09', 'dhruvisha@gmail.com', '9900999999')

Did you notice that we did not insert any number into the CustomerID field?
The CustomerID column is an **auto-increment** field and will be **generated automatically** when a new record is inserted into the table.

Insert Data Only in Specified Columns

It is also possible to only insert data in specific columns.

INSERT into students (fname, lname) VALUES ('Devangi', 'Dave')

Warning: #1364 Field 'city' doesn't have a default value
Warning: #1364 Field 'gender' doesn't have a default value
Warning: #1364 Field 'dob' doesn't have a default value
Warning: #1364 Field 'email' doesn't have a default value
Warning: #1364 Field 'phone' doesn't have a default value

MySQL NULL Values

What is a NULL Value?

A field with a NULL value is a field with no value.

If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

Note: A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation!

How to Test for NULL Values?

It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

We will have to use the `IS NULL` and `IS NOT NULL` operators instead.

SELECT * FROM `students` WHERE city is null

SELECT * FROM `students` WHERE city = '';

```
ALTER TABLE `students` CHANGE `city` `city` VARCHAR(32) NULL;
```

SELECT * FROM `students` WHERE city is null

SELECT * FROM `students` WHERE city is not null;

The IS NULL Operator

The `IS NULL` operator is used to test for empty values (NULL values).

The IS NOT NULL Operator

The `IS NOT NULL` operator is used to test for non-empty values (NOT NULL values).

MySQL UPDATE Statement

The MySQL UPDATE Statement

The `UPDATE` statement is used to modify the existing records in a table.

UPDATE Syntax

UPDATE *table_name*
SET *column1* = *value1*, *column2* = *value2*, ...
WHERE *condition*;

Note: Be careful when updating records in a table! Notice the `WHERE` clause in the `UPDATE` statement. The `WHERE` clause specifies which record(s) that should be updated. If you omit the `WHERE` clause, all records in the table will be updated!

```
UPDATE students set gender = 'Female'
```

```
UPDATE students set city = 'Rajkot' WHERE id = 10
```

UPDATE Multiple Records

It is the `WHERE` clause that determines how many records will be updated.

```
UPDATE students set city = 'Rajkot', gender = 'female', phone = '9998887770' WHERE id = 10;
```

Update Warning!

Be careful when updating records. If you omit the `WHERE` clause, ALL records will be updated!

MySQL LIMIT Clause

The MySQL LIMIT Clause

The `LIMIT` clause is used to specify the number of records to return.

The `LIMIT` clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.

LIMIT Syntax

```
SELECT * FROM students

SELECT * FROM students LIMIT 5;

SELECT * FROM students LIMIT 5 OFFSET 5;

SELECT * FROM students LIMIT 10, 5;
```

MySQL LIMIT Examples

The following SQL statement selects the first three records from the "Customers" table:

Example

```
SELECT * FROM Customers
LIMIT 3;
```

What if we want to select records 4 - 6 (inclusive)?

MySQL provides a way to handle this: by using OFFSET.

The SQL query below says "return only 3 records, start on record 4 (OFFSET 3)":

Example

```
SELECT * FROM Customers
LIMIT 3 OFFSET 3;

SELECT * from students WHERE city = 'Ahamdabad';

SELECT * from students WHERE city = 'Ahamdabad' LIMIT 2;
```

MySQL MIN() and MAX() Functions

MySQL MIN() and MAX() Functions

The `MIN()` function returns the smallest value of the selected column.

The `MAX()` function returns the largest value of the selected column.

MIN() Syntax

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

MAX() Syntax

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

```
SELECT min(dob) FROM students;
```

```
SELECT max(dob) FROM students;
```

```
SELECT MAX(id) FROM students
```

```
SELECT min(id) FROM students;
```

```
SELECT MIN(dob) FROM students;
```

```
SELECT MIN(dob) as "Oldest Student" FROM students;
```

```
SELECT MAX(dob) as "Youngest Student" FROM students;
```

MySQL COUNT(), AVG() and SUM() Functions

MySQL COUNT(), AVG() and SUM() Functions

The `COUNT()` function returns the number of rows that matches a specified criterion.

COUNT() Syntax

```
SELECT COUNT(column_name) FROM table_name WHERE condition;
```

The `AVG()` function returns the average value of a numeric column.

AVG() Syntax

```
SELECT AVG(column_name) FROM table_name WHERE condition;
```

The `SUM()` function returns the total sum of a numeric column.

SUM() Syntax

```
SELECT SUM(column_name) FROM table_name WHERE condition;
```

```
SELECT COUNT(id) FROM students
```

```
SELECT COUNT(id) FROM students WHERE city = 'Ahamdabad';
```

```
SELECT COUNT(id) FROM students WHERE city <> 'Ahamdabad';
```

```
SELECT sum(id) FROM students
```

```
SELECT avg(id) FROM students
```