MySQL Database

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons −

* MySQL is released under an open-source license. So you have nothing to pay to use it.
* MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
* MySQL uses a standard form of the well-known SQL data language.
* MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
* MySQL works very quickly and works well even with large data sets.
* MySQL is very friendly to PHP, the most appreciated language for web development.
* MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
* MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

history of MySQL:

* The mSQL database system was designed to connect the data that is stored in tables using customized fast low level (ISAM) routines.
* ISAM refers to indexed sequential access method, which is a file management system. It is a technique that helps access records in the tables sequentially, i.e in the same order in which the records were entered into the table, or randomly with the help of an index. Every index can be used to define a different order for the records in the table.
* ISAM was originally developed by IBM before the development of VSAM (Virtual Storage Access Method) and relational databases.
* When mSQL was tested to see its efficiency, it was observed that mSQL was not quick enough or flexible enough for the requirements in hand.
* MySQL was initially created for personal usage from mSQL based on the low-level language ISAM.
* MySQL has been named after co-founder Monty Widenius’s daughter- My. The logo, a dolphin is known as ‘Sakila’.

features of MySQL:

**Relational Database Management System (RDBMS)**

[MySQL](https://www.javatpoint.com/mysql-tutorial) is a relational database management system. This database language is based on the [SQL](https://www.javatpoint.com/sql-tutorial) queries to access and manage the records of the table.

Keep Watching

**Easy to use**

MySQL is easy to use. We have to get only the basic knowledge of SQL. We can build and interact with MySQL by using only a few simple SQL statements.

**It is secure**

MySQL consists of a solid data security layer that protects sensitive data from intruders. Also, passwords are encrypted in MySQL.

**Client/ Server Architecture**

MySQL follows the working of a client/server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they can query data, save changes, etc.

**Free to download**

MySQL is free to use so that we can download it from MySQL official website without any cost.

**It is scalable**

MySQL supports multi-threading that makes it easily scalable. It can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, we can increase this number to a theoretical limit of 8 TB of data.

**Speed**

MySQL is considered one of the very fast database languages, backed by a large number of the benchmark test.

**High Flexibility**

MySQL supports a large number of embedded applications, which makes MySQL very flexible.

**Compatible on many operating systems**

MySQL is compatible to run on many operating systems, like Novell NetWare, Windows\* Linux\*, many varieties of UNIX\* (such as Sun\* Solaris\*, AIX, and DEC\* UNIX), OS/2, FreeBSD\*, and others. MySQL also provides a facility that the clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).

**Allows roll-back**

MySQL allows transactions to be rolled back, commit, and crash recovery.

**Memory efficiency**

Its efficiency is high because it has a very low memory leakage problem.

**High Performance**

MySQL is faster, more reliable, and cheaper because of its unique storage engine architecture. It provides very high-performance results in comparison to other databases without losing an essential functionality of the software. It has fast loading utilities because of the different cache memory.

**High Productivity**

MySQL uses Triggers, Stored procedures, and views that allow the developer to give higher productivity.

**Platform Independent**

It can download, install, and execute on most of the available operating systems.

**Partitioning**

This feature improves the performance and provides fast management of the large database.

**GUI Support**

MySQL provides a unified visual database graphical user interface tool named "**MySQL Workbench**" to work with database architects, developers, and Database Administrators. [MySQL Workbench](https://www.javatpoint.com/mysql-workbench) provides SQL development, data modeling, data migration, and comprehensive administration tools for server configuration, user administration, backup, and many more. MySQL has a fully GUI supports from MySQL Server version 5.6 and higher.

**Dual Password Support**

MySQL version 8.0 provides support for dual passwords: one is the current password, and another is a secondary password, which allows us to transition to the new password.

### **Disadvantages/Drawback of MySQL**

Following are the few disadvantages of MySQL:

* MySQL version less than 5.0 doesn't support ROLE, COMMIT, and stored procedure.
* MySQL does not support a very large database size as efficiently.
* MySQL doesn't handle transactions very efficiently, and it is prone to data corruption.
* MySQL is accused that it doesn't have a good developing and debugging tool compared to paid databases.
* MySQL doesn't support SQL check constraints.

Mysql datatypes

**string, numeric, and date and time**.

¬ Create Database

CREATE DATABASE testDB;

¬ Select Database

SELECT \* FROM *table\_name*;

¬ Drop Database

DROP DATABASE *databasename*;

¬ Table & Views

CREATE OR REPLACE VIEW [Brazil Customers] AS  
SELECT CustomerName, ContactName, City  
FROM Customers  
WHERE Country = 'Brazil';

¬ CREATE Table\

CREATE TABLE Persons (  
    PersonID int,  
    LastName varchar(255),  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255)  
);

¬ ALTER Table

ALTER TABLE Customers  
ADD Email varchar(255);

¬ TRUNCATE Table

TRUNCATE TABLE CUSTOMERS;

¬ DROP Table

DROP TABLE Shippers;

¬ MySQL Queries

1. **update** customers **set** **name**='bob', city='london' **where** id=101;

renu: UPDATE Customers

SET Email = 'renuka@gmail.com', City= 'Stavanger'

WHERE PersonID = 1;

¬ INSERT Record

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)  
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

¬ UPDATE Record

UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
WHERE CustomerID = 1;

¬ DELETE Record

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

SELECT Record

SELECT \* FROM Customers

¬ MySQL Clauses

¬ MySQL WHERE

**SELECT** \*

**FROM** officers

**WHERE** address = 'Lucknow'

OR address = 'Mau';

¬ MySQL DISTINCT

**SELECT** **DISTINCT** officer\_name, address

**FROM** officers;

¬ MySQL FROM

**SELECT** officers.officer\_id, students.student\_name

**FROM** officers

LEFT OUTER JOIN students

**ON** officers.officer\_id = students.student\_id;

¬ MySQL ORDER BY

**SELECT** \*

**FROM** officers

**WHERE** address = 'Lucknow'

**ORDER** **BY** officer\_name;

¬ MySQL GROUP BY

**SELECT** address, COUNT(\*)

**FROM**   officers

**GROUP** **BY** address;

¬ MySQL HAVING

**SELECT** emp\_name, SUM(working\_hours) **AS** "Total working hours"

**FROM** employees

**GROUP** **BY** emp\_name

**HAVING** SUM(working\_hours) > 5;

¬ MySQL Conditions

¬ MySQL AND

**SELECT** \*

**FROM** cus\_tbl

**WHERE** cus\_firstname = 'Ajeet'

AND cus\_id > 3;

¬ MySQL OR

**SELECT** \*

**FROM** cus\_tbl

**WHERE** cus\_firstname = 'Ajeet'

OR cus\_id > 100;

¬ MySQL AND OR

**SELECT** \*

**FROM** students

**WHERE** (course\_name = 'Java' AND student\_name = 'Aryan')

OR (student\_id < 2);

¬ MySQL LIKE

**SELECT** officer\_name

**FROM** officers

**WHERE** address LIKE 'Luck%';

¬ MySQL IN

**SELECT** \*

**FROM** officers

**WHERE** officer\_name IN ('Ajeet', 'Vimal', 'Deepika');

¬ MySQL NOT

**SELECT** \*

**FROM** officers

**WHERE** officer\_name NOT IN ('Ajeet','Vimal','Deepika');

¬ MySQL IS NULL

**SELECT** \*

**FROM** officers

**WHERE** officer\_name **IS** NULL;

¬ MySQL IS NOT NULL

**SELECT** \*

**FROM** officers

**WHERE** officer\_name **IS** NOT NULL;

¬ MySQL BETWEEN

**SELECT** \*

**FROM** officers

**WHERE** officer\_id BETWEEN 1 AND 3;

¬ MySQL JOINS

¬ MySQL INNER JOIN

**SELECT** officers.officer\_name, officers.address, students.course\_name

**FROM** officers

**INNER** JOIN students

**ON** officers.officer\_id = students.student\_id;

¬ MySQL LEFT OUTER JOIN

All details come from 2nd table

**SELECT** customers.customer\_id, cust\_name, price, **date**

**FROM** customers

LEFT JOIN orders **ON** customers.customer\_id = orders.customer\_id;

¬ MySQL RIGHT OUTER JOIN ¬

All details from 1st will come which is mapping the table 2

**SELECT** customers.customer\_id, cust\_name, price, **date**

**FROM** customers

RIGHT JOIN orders **ON** customers.customer\_id = orders.customer\_id

**ORDER** **BY**  customer\_id;

Aggregate Functions ¬

MySQL count()

1. mysql> **SELECT** COUNT(emp\_name) **FROM** employees;

¬ MySQL sum()

1. mysql> **SELECT** SUM(working\_hours) **AS** "Total working hours" **FROM** employees;

¬ MySQL avg()

1. mysql> **SELECT** AVG(working\_hours) Avg\_working\_hours **FROM** employees;

¬ MySQL min()

1. mysql> **SELECT** **MIN**(income) **AS** Minimum\_Income **FROM** employees;

¬ MySQL max()

1. > **SELECT** **MAX**(income) **AS** "Maximum\_Income"
2. **FROM** employees
3. **WHERE** emp\_age > 35;

¬ MySQL first()

1. **SELECT** officer\_name
2. **FROM** officers
3. LIMIT 2;

¬ MySQL last()

1. **SELECT** officer\_name
2. **FROM** officers
3. **ORDER** **BY** officer\_name **DESC**
4. LIMIT 1;

¬ How to write a stored procedure

¬ Executing stored procedures

## What is a Stored Procedure?

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

### **Stored Procedure Syntax**

CREATE PROCEDURE procedure\_name  
AS  
sql\_statement  
GO;

### **Execute a Stored Procedure**

EXEC procedure\_name;

Renu

UPDATE `testdb`.`customers` SET `Address` = '' WHERE (`PersonID` = '1');

Deleting sep

ALTER TABLE `testdb`.`customers`

DROP COLUMN `Address`;

ALTER TABLE `testdb`.`customers`

ADD COLUMN `address` VARCHAR(45) NOT NULL AFTER `Email`,

DROP PRIMARY KEY,

ADD PRIMARY KEY (`PersonID`, `address`);

;

UPDATE Customers

SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'

WHERE CustomerID = 1;

I can see separate column

SELECT \*

FROM customers

WHERE Email = 'renuka@gmail.com'

OR address = 'Mau';

I can see only address

SELECT DISTINCT address

FROM customers;

09:42:45 DELETE FROM Customers WHERE CustomerName='Alfred Schmidt' Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a KEY colum

NIGHT:

SELECT \*

FROM customers

WHERE address = 'Skagen 21'

OR address = 'Mau';

desc

SELECT \*

FROM customers

WHERE 1=1

ORDER BY CustomerName desc ;

SELECT \*

FROM customers

GROUP BY Country; // removing duplicate values

CONDITION

SELECT CustomerName, SUM(PersonID) AS "Total working hours"

FROM customers

GROUP BY CustomerName

HAVING SUM(PersonID) > 1;

using AND OPERATOR

SELECT \*

FROM customers

WHERE CustomerName = 'renuka'

AND PersonID > 3;

Morning

SELECT \*

FROM customers

WHERE ContactName = 'raju'

OR PersonID > 3;

SELECT \*

FROM customers

WHERE (Country = 'india' AND CustomerName = 'nnn')

OR (PersonID < 4); //its not checking india or nnn constraints personid

SELECT CustomerName

FROM customers

WHERE address LIKE 'Sk%'; //only customer name will get which is there sk%

SELECT \*

FROM customers

WHERE ContactName IN ('raju', 'ravi'); //only two columns

SELECT \*

FROM customers

WHERE ContactName NOT IN ('raju', 'ravi'); //WITHOUT raju and ravi

SELECT \*

FROM customers

WHERE Email IS NULL; //which is null in email

SELECT \*

FROM customers

WHERE Email IS Not NULL; //which is not null

SELECT \*

FROM customers

WHERE PersonID BETWEEN 1 AND 4; //between 1 and 4

SELECT customers.CustomerName, customers.address, persons.FirstName

FROM customers

INNER JOIN persons //

SELECT customers.CustomerName, customers.address, persons.LastName

FROM customers

INNER JOIN persons //both will join the column

SELECT customers.CustomerName, customers.address, persons.LastName

FROM customers

INNER JOIN persons ON customers.PersonID = persons.PersonID;//

SELECT \* FROM testdb.customers;

SELECT customers.PersonID, cust\_name

FROM customers

LEFT JOIN persons ON customers.PersonID = persons.PersonID;

SELECT \* FROM testdb.customers;

SELECT customers.PersonID, CustomerName

FROM customers

RIGHT JOIN persons ON customers.PersonID = persons.PersonID

ORDER BY PersonID;

SELECT \* FROM testdb.customers;

SELECT COUNT(CustomerName) FROM customers; //just counted id total 6(counting column)

SELECT SUM(PersonID) AS "Total working hours" FROM customers; //counting columns values

SELECT AVG(PersonID) Avg\_working\_hours FROM customers; 2.666

SELECT MIN(PersonID) AS Minimum\_Income FROM customers;// MIN:1

SELECT MAX(PersonID) AS Maximum\_Income FROM customers;//MAX:1

SELECT \* FROM testdb.customers;

SELECT MAX(income) AS "Maximum\_Income"

FROM employees

WHERE emp\_age > 35

SELECT CustomerName

FROM customers LIMIT 2;

SELECT CustomerName

FROM customers

ORDER BY CustomerName DESC

LIMIT 1