MYSQL

By Techolas

INTRODUCTION- What is SQL

- SQL is a standard language for storing, manipulating and retrieving data in databases
- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Intro-what is mysql

- MySQL is a fast, easy to use relational database. It is currently the most popular open-source database
- MySQL is an open source database product that was created by MySQL AB, a company founded in 1995 in Sweden.
- In 2008, MySQL AB announced that it had agreed to be acquired by Sun Microsystems for approximately $1 billion.

MYSQL-FEATURES

- **Relational Database Management System (RDBMS):** MySQL is a relational database management system.
- **Easy to use:** MySQL is easy to use. You have to get only the basic knowledge of SQL. You can build and interact with MySQL with only a few simple SQL statements.
- **It is secure:** MySQL consist of a solid data security layer that protects sensitive data from intruders. Passwords are encrypted in MySQL.
- **Client/ Server Architecture:** MySQL follows a client /server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc.

MYSQL-FEATURES

- **Free to download:** MySQL is free to use and you can download it from MySQL official website.
- **It is scalable:** MySQL 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, you can increase this number to a theoretical limit of 8 TB of data.
- **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-FEATURES

- **Allows roll-back:** MySQL allows transactions to be rolled back, commit and crash recovery.
- **High Performance:** MySQL is faster, more reliable and cheaper because of its unique storage engine architecture.
- **High Flexibility:** MySQL supports a large number of embedded applications which makes MySQL very flexible.
- **High Productivity:** MySQL uses Triggers, Stored procedures and views which allows the developer to give a higher productivity.

MYSQL-DATA TYPES

- A Data Type specifies a particular type of data, like integer, floating points, String,Boolean etc.
- It uses many different data types broken into mainly three categories: numeric, date and time, and string types.

TECHOLAS
TECHNOLOGY DEMYSTIFIED

NUMERIC DATA TYPES

INT

INT	• A normal-sized integer that can be signed or unsigned • If signed, the allowable range is from -2147483648 to 2147483647. • If unsigned, the allowable range is from 0 to 4294967295.
TINYINT	• A very small integer that can be signed or unsigned. • If signed, the allowable range is from -128 to 127. • If unsigned, the allowable range is from 0 to 255.
SMALLINT	• A small integer that can be signed or unsigned. • If signed, the allowable range is from -32768 to 32767. • If unsigned, the allowable range is from 0 to 65535.
MEDIUMINT	• A medium-sized integer that can be signed or unsigned. • If signed, the allowable range is from -8388608 to 8388607. • If unsigned, the allowable range is from 0 to 16777215
BIGINT	• A large integer that can be signed or unsigned. • If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. • If unsigned, the allowable range is from 0 to 18446744073709551615.

NUMERIC DATA TYPES-FLOAT

f

FLOAT(m,d)	• A floating-point number that cannot be unsigned. • You can define the display length (m) and the number of decimals (d). • This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). • Decimal precision can go to 24 places for a float.
DOUBLE(m,d)	• A double precision floating-point number that cannot be unsigned. • You can define the display length (m) and the number of decimals (d). • This is not required and will default to 16,4, where 4 is the number of decimals. • Decimal precision can go to 53 places for a double. Real is a synonym for double.
DECIMAL(m,d)	• An unpacked floating-point number that cannot be unsigned. • In unpacked decimals, each decimal corresponds to one byte. • Defining the display length (m) and the number of decimals (d) is required. Numeric is a synonym for decimal.

DATE DATA TYPES

D

DATE	Displayed as 'yyyy-mm-dd'.
DATETIME	Displayed as 'yyyy-mm-dd hh:mm:ss'.
TIMESTAMP(m)	Displayed as 'YYYY-MM-DD HH:MM:SS'.
TIME	Displayed as 'HH:MM:SS'.
YEAR[(2|4)]	Year value as 2 digits or 4 digits Default is 4 digit

String DATA TYPES

S

CHAR(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Fixed-length strings. Space padded on right to equal size characters.
VARCHAR(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Variable-length string.
TINYTEXT(size)	Maximum size of 255 characters.	Where size is the number of characters to store.
TEXT(size)	Maximum size of 65,535 characters.	Where size is the number of characters to store.
MEDIUMTEXT(size)	Maximum size of 16,777,215 characters.	Where size is the number of characters to store.

String Data Types Continues..

S

LONGTEXT(size)	Maximum size of 4GB or 4,294,967,295 characters.	Where size is the number of characters to store.
BINARY(size)	Maximum size of 255 characters.	• Where size is the number of binary characters to store. • Fixed-length strings. Space padded on right to equal size characters. • (introduced in MySQL 4.1.2)
VARBINARY(size)	Maximum size of 255 characters.	• Where size is the number of characters to store. • Variable-length string. • (introduced in MySQL 4.1.2)

INSTALL MYSQL

- Go to MySQL official website http://www.mysql.com/downloads/
- **Go to google and search mysql installer**

MYSQL CREATE DATABASE

- A database is a collection of data.
- MySQL allows us to store and retrieve the data from the database in a efficient way.

SYNTAX

```
CREATE DATABASE IF NOT EXISTS database_name;
```

EXAMPLE

```
CREATE DATABASE employees;
```

TO LIST ALL DATABASES

```
SHOW DATABASES;
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Use and Drop

- You can use SQL command USE to select a particular database.

  USE database_name;

  Example: USE employees;

- Delete a database

  DROP DATABASE IF EXISTS database_name

  Example: DROP DATABASE IF EXISTS employee;

TECHOLAS
TECHNOLOGY DEMYSTIFIED

CREATE 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.
- A relational database defines database relationships in the form of tables

SYNTAX

CREATE TABLE table_name (column_name column_type...);

Example:

```
CREATE TABLE cus_tbl(
   cus_id INT NOT NULL AUTO_INCREMENT,
   cus_firstname VARCHAR(100) NOT NULL,
   cus_surname VARCHAR(100) NOT NULL,
   PRIMARY KEY ( cus_id )
);
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

SHOW AND DESCRIBE

- To list all Tables created in the database

  SHOW tables;

- See the table structure:

  DESCRIBE table_name

  Example: describe cus_tbl

TECHOLAS
TECHNOLOGY DEMYSTIFIED

MYSQL ALTER table

1. ADD :To Add a new column to the table

Syntax:

ALTER TABLE table_name

ADD new_column_name column definition

[**FIRST** | **AFTER** column_name];

Example:

ALTER TABLE cus_tbl

ADD cus_age **varchar**(40) NOT NULL;

TECHOLAS
TECHNOLOGY DEMYSTIFIED

ADD MULTIPLE COLUMNS

Syntax:

```
ALTER TABLE table_name
ADD new_column_name column_definition
[ FIRST | AFTER column_name ],
ADD new_column_name column_definition
[ FIRST | AFTER column_name ],  …..;
```

Example:

```
ALTER TABLE cus_tbl
ADD cus_address varchar(100) NOT NULL
AFTER cus_surname,
ADD cus_salary int(100) NOT NULL
AFTER cus_age ;
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

ALTER TABLE continues..

2. **MODIFY:** To change the column definition of the table

Syntax:

```
ALTER TABLE table_name
MODIFY column_name column_definition
[ FIRST | AFTER column_name ];
```

Example:

```
ALTER TABLE cus_tbl
MODIFY cus_surname varchar(50) NULL;
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

ALTER TABLE continues..

3. **DROP:** To delete a column in a table

Syntax:

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Example:

```
ALTER TABLE cus_tbl
DROP COLUMN cus_address;
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

ALTER TABLE continues..

4. **CHANGE :** To rename a column in the table

Syntax:

ALTER TABLE table_name

CHANGE **COLUMN** old_name new_name

column_definition

[**FIRST** | **AFTER** column_name]

Example:

```
ALTER TABLE  cus_tbl
CHANGE COLUMN cus_surname cus_title
varchar(20) NOT NULL;
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

ALTER TABLE continues..

5. **RENAME :** To rename a table

Syntax:

```
ALTER TABLE table_name
RENAME TO new_table_name;
```

Example:

```
ALTER TABLE  cus_tbl
RENAME TO cus_table;
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

MYSQL DROP TABLE

Syntax:

```
DROP TABLE table_name;
```

Example

```
DROP TABLE cus_tbl;
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

MYSQL QUERIES

INSERT : To insert values into the table

Syntax:

```
INSERT INTO table_name ( field1, field2,...fieldN )
VALUES
( value1, value2,...valueN );
```

Example:

```
CREATE TABLE emp(
id INT NOT NULL,
name VARCHAR(100) NOT NULL,
salary INT NOT NULL
);
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

INSERT continues..

```
INSERT INTO emp VALUES (7, 'Sonoo', 40000);
```

Or,

```
INSERT INTO emp(id,name,salary) VALUES (7, 'Sonoo', 40000);
```

INSERT :For partial fields

```
INSERT INTO emp(id,name) VALUES (7, 'Sonoo');
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

INSERT continues..

INSERT :Inserting multiple records

```
INSERT INTO cus_tbl
(cus_id, cus_firstname, cus_surname)
VALUES
(5, 'Ajeet', 'Maurya'),
(6, 'Deepika', 'Chopra'),
(7, 'Vimal', 'Jaiswal');
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

MYSQL SELECT QUERY..

Syntax :

SELECT expressions

FROM tables

[**WHERE** conditions];

Example:

SELECT * FROM cus_tbl

Or,

SELECT cus_id, cus_firstname, cus_surname **FROM** cus_tbl

TECHOLAS
TECHNOLOGY DEMYSTIFIED

UPDATE QUERY

SYNTAX:

```
UPDATE table_name
SET field1=new-value1, field2=new-value2, ...
[WHERE Clause]
```

Example:

```
UPDATE cus_tbl
SET cus_surname = 'Ambani'
WHERE cus_id = 5;
```

```
UPDATE cus_tbl
SET cus_surname = 'Ambani' , salary =100000
WHERE cus_id = 5;
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

DELETE QUERY

Syntax:

```
DELETE FROM table_name
WHERE
(Condition specified);
```

Example:

```
DELETE FROM cus_tbl
WHERE cus_id = 6;
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

How to use auto_increment

Example: Create a new table ‘officers’ in the employees database

```
use employees;
create table officers(
officer_id int not null auto_increment,
officer_name varchar(100) not null,
address varchar(100) not null,
primary key (officer_id)
);
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Auto_increment continues...

We can automatically get values added to the officer_id column without inserting it.

```
use employees;
insert into officers (officer_name, address)
values ('Ajeet','Mau'),
('Deepika', 'Lucknow'),
('Vimal','Delhi');
select * from officers;
```

To change the starting value do, **alter table** officers **auto_increment** =5;

(after creating the table again)

TECHOLAS
TECHNOLOGY DEMYSTIFIED

MYSQL WHERE CLAUSE

Syntax:

```
WHERE conditions;
```

Example:

```
SELECT *
FROM officers
WHERE address = 'Mau';
```

Mysql Where With AND

```
SELECT *
FROM table_name
WHERE condition1
AND condition2;
```

Example:

```
SELECT *
FROM officers
WHERE address = 'Lucknow'
AND officer_id < 5;
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

WHERE CLAUSE WITH OR

```
SELECT *
FROM table_name
WHERE condition1
OR condition2;
```

EXAMPLE :

```
SELECT *
FROM officers
WHERE address = 'Lucknow'
OR address = 'Mau';
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

WHERE CLAUSE WITH AND & OR

```
SELECT *
FROM table_name
WHERE (AND_CONDITIONS)
OR (conditions);
```

EXAMPLE:

```
SELECT *
FROM officers
WHERE (address = 'Mau' AND officer_name = 'Ajeet')
OR (officer_id < 5);
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

FROM clause..

The MySQL FROM Clause is used to select some records from a table. It can also be used to retrieve records from multiple tables using JOIN condition.

Syntax:

FROM table1

[{ **INNER** JOIN | LEFT [OUTER] JOIN| RIGHT [OUTER] JOIN } table2

ON table1.column1 = table2.column1]

TECHOLAS
TECHNOLOGY DEMYSTIFIED

DISTINCT

MySQL DISTINCT clause is used to remove duplicate records from the table and fetch only the unique records. The DISTINCT clause is only used with the SELECT statement.

SYNTAX:

SELECT DISTINCT expressions

FROM tables

[**WHERE** conditions];

Example:

SELECT DISTINCT name,age **FROM** employee;

TECHOLAS
TECHNOLOGY DEMYSTIFIED

ORDER BY

The MYSQL ORDER BY Clause is used to sort the records in ascending or descending order.

Syntax:

SELECT expressions

FROM tables

[**WHERE** conditions]

ORDER BY expression [**ASC** | **DESC**];

Example:

```
SELECT *
FROM officers
ORDER BY officer_name;  (default sorting is ascending sorting)
```

ORDER BY continues...

```
SELECT *
FROM officers
WHERE address = 'Lucknow'
ORDER BY officer_name ASC;  (default sorting is ascending sorting)

SELECT *
FROM officers
WHERE address = 'Lucknow'
ORDER BY officer_name DESC;
```

GROUP BY

The MYSQL GROUP BY Clause is used to collect data from multiple records and group the result by one or more column. It is generally used in a SELECT statement.

```
SELECT expression1, expression2, ... expression_n,
aggregate_function (expression)
FROM tables
[WHERE conditions]
GROUP BY expression1, expression2, ... expression_n;
```

aggregate_function: It specifies a function such as SUM, COUNT, MIN, MAX, or AVG etc. tables: It specifies the tables, from where you want to retrieve the records. There must be at least one table listed in the FROM clause.

COUNT function

3 forms are there for count(): **count(*)**, **count(expression)**, **count(distinct expression)**

Count(*)

- Returns the number of rows

```
SELECT COUNT(*) FROM officers;
```

- Returns the number of rows based on a condition

```
SELECT COUNT(*) FROM officers WHERE officer_id=7;
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

COUNT continues...

Count(expression)

- Returns the number of non-null records of the table

SELECT COUNT(address) **FROM** officers;

Count(distinct expression)

- Returns the number of distinct non-null records

SELECT COUNT(**DISTINCT** address) **FROM** officers;

TECHOLAS
TECHNOLOGY DEMYSTIFIED

GROUP BY with aggregate functions

- Group by with **COUNT** function

This returns the count of rows of each distinct entries.

```
SELECT address, COUNT(*)
FROM officers
GROUP BY address;
```

Table: emp_data

emp_id	emp_name	working_date	working_hours
1	Ajeet	2015-01-24	12
2	Ayan	2015-01-24	11
3	Milan	2015-01-24	11
4	Ruchi	2015-01-24	6
1	Ajeet	2015-01-25	10
2	Ayan	2015-01-25	10
4	Ruchi	2015-01-25	7
3	Milan	2015-01-25	10
1	Ajeet	2015-01-26	9
3	Milan	2015-01-26	10

GROUP BY with aggregate functions cont...

- GROUP BY with **SUM** function

```
SELECT emp_name, SUM(working_hours) AS  "total working hours“
FROM emp_data
GROUP BY emp_name;
```

- GROUP BY with **MIN** function

```
SELECT emp_name, MIN(working_hours) AS  “minimum working hours“
FROM emp_data
GROUP BY emp_name;
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

GROUP BY with aggregate functions cont...

- GROUP BY with **MAX** function

```
SELECT emp_name, MAX(working_hours) AS  "maximum working hours“
FROM emp_data
GROUP BY emp_name;
```

- GROUP BY with **AVG** function

```
SELECT emp_name, AVG(working_hours) AS  "average working hours“
FROM emp_data
GROUP BY emp_name;
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

HAVING clause..

MySQL HAVING Clause is used with GROUP BY clause. It always returns the rows where condition is TRUE.

Syntax:

```
SELECT expression1, expression2, ... expression_n,
aggregate_function (expression)
FROM tables
[WHERE conditions]
GROUP BY expression1, expression2, ... expression_n
HAVING condition;
```

EXAMPLE:

```
SELECT emp_name, SUM(working_hours) AS "Total working hours"
FROM employees
GROUP BY emp_name
HAVING SUM(working_hours) > 15;
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

MySQL Limit..

Used to restrict the number of rows returned rather than fetching the whole set. It is essential when the table contains large number of rows.

Syntax:

```
SELECT column_name
FROM table_name
LIMIT  Number;
```

Example:

```
SELECT age,group ,COUNT(name) from employee GROUP BY  age LIMIT 1;
```

Like.. operator

expression LIKE pattern;

1) Using % (percent) Wildcard:

```
select * from employee where name like '%n';
```

2) Using _ (Underscore) Wildcard:

```
select * from employee where name like 'mi__un';
```

3) Using NOT Operator:

```
select * from employee where name not like 'mithun';
```

IN operator..

The MySQL IN condition is used to reduce the use of multiple OR conditions in a SELECT, INSERT, UPDATE and DELETE statement.

expression IN (value1, value2, ... value_n);

select * from employee where age in (25,26,27,28);

TECHOLAS
TECHNOLOGY DEMYSTIFIED

IS NOT NULL ...

MySQL IS NOT NULL condition is used to check the NOT NULL value in the expression

expression **IS** NOT NULL

Example:

```
SELECT * FROM employee WHERE name IS NOT NULL;
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

IS NULL..

MySQL IS NULL condition is used to check if there is a NULL value in the expression.

expression **IS** NULL

example:

```
SELECT * FROM employee WHERE name IS NULL;
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

BETWEEN..

The MYSQL BETWEEN condition specifies how to retrieve values from an expression within a specific range.

expression BETWEEN value1 AND value2;

Example:

select * from employee where age BETWEEN 22 AND 26;

TECHOLAS
TECHNOLOGY DEMYSTIFIED

JOIN..

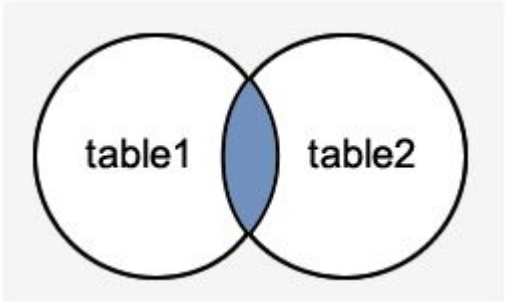
MySQL JOINS are used with SELECT statement. It is used to retrieve data from multiple tables.

1. MySQL INNER JOIN (or sometimes called simple join)
2. MySQL LEFT OUTER JOIN (or sometimes called LEFT JOIN)
3. MySQL RIGHT OUTER JOIN (or sometimes called RIGHT JOIN)

INNER JOIN (simple join..)

The MySQL INNER JOIN is used to return all rows from multiple tables where the join condition is satisfied.

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column = table2.column;
```

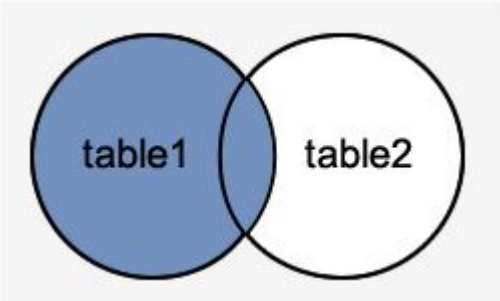


TECHOLAS
TECHNOLOGY DEMYSTIFIED

MYSQL LEFT OUTER JOIN..

The LEFT OUTER JOIN returns all rows from the left hand table specified in the ON condition and only those rows from the other table where the join condition is fulfilled.

```
SELECT columns
FROM table1
LEFT [OUTER] JOIN table2
ON table1.column = table2.column;
```

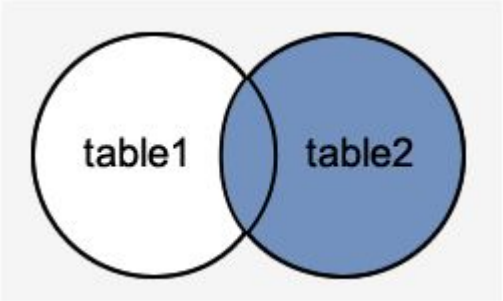


TECHOLAS
TECHNOLOGY DEMYSTIFIED

RIGHT OUTER JOIN..

The MySQL Right Outer Join returns all rows from the RIGHT-hand table specified in the ON condition and only those rows from the other table where he join condition is fulfilled.

```
SELECT columns
FROM table1
RIGHT [OUTER] JOIN table2
ON table1.column = table2.column;
```



TECHOLAS
TECHNOLOGY DEMYSTIFIED

REGEX..

1.regexp operator

Syntax

1. **select name from** table_name **where** column_name regexp 'pattern';

Example:

select * from student where s_name regexp '^m';

OR

select * from student where s_name NOT regexp '^m';

TECHOLAS
TECHNOLOGY DEMYSTIFIED

CREATE OUR OWN MYSQL FUNCTION..

SYNTAX:

```
CREATE FUNCTION function_name [ (parameter datatype [, parameter datatype]) ]
RETURNS return_datatype
BEGIN
Declaration_section
Executable_section
END;
```

```
DELIMITER $$
CREATE FUNCTION get_parrent(id INT) RETURNS VARCHAR(20)
READS SQL DATA
DETERMINISTIC
BEGIN
DECLARE parrent VARCHAR(20) DEFAULT '';
SELECT parrent_name INTO parrent FROM student WHERE rollno=id;
RETURN parrent;
END $$
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED