MySQL AUTO INCREMENT Field

What is an AUTO INCREMENT Field?

Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.

Often this is the primary key field that we would like to be created automatically every time a new record is inserted.

MySQL AUTO_INCREMENT Keyword

MySQL uses the AUTO_INCREMENT keyword to perform an auto-increment feature.

By default, the starting value for AUTO_INCREMENT is 1, and it will increment by 1 for each new record.

The following SQL statement defines the "Personid" column to be an auto-increment primary key field in the "Persons" table:

```
CREATE TABLE Persons (
   Personid int NOT NULL AUTO_INCREMENT,
   LastName varchar(255) NOT NULL,
   FirstName varchar(255),
   Age int,
   PRIMARY KEY (Personid)
);
```

To let the AUTO_INCREMENT sequence start with another value, use the following SQL statement:

```
ALTER TABLE Persons AUTO_INCREMENT=100;
```

When we insert a new record into the "Persons" table, we do NOT have to specify a value for the "Personid" column (a unique value will be added automatically):

```
INSERT INTO Persons (FirstName,LastName)
VALUES ('Lars','Monsen');
```

MySQL Dates

The most difficult part when working with dates is to be sure that the format of the date you are trying to insert, matches the format of the date column in the database.

As long as your data contains only the date portion, your queries will work as expected. However, if a time portion is involved, it gets more complicated.

MySQL Date Data Types

MySQL comes with the following data types for storing a date or a date/time value in the database:

- DATE - format YYYY-MM-DD
- DATETIME - format: YYYY-MM-DD HH:MI:SS
- TIMESTAMP - format: YYYY-MM-DD HH:MI:SS
- YEAR - format YYYY or YY

MySQL Data Types

The data type of a column defines what type of value the column can store, such as integer, character, date, time, binary data, etc.
Every column in a MySQL table **must have a data type**, which helps MySQL understand how to store and process the data.

MySQL mainly supports **three categories of data types**:

- String Data Types
- Numeric Data Types
- Date and Time Data Types

---

1. String Data Types

Data Type	**Description**
CHAR(size)	Fixed-length string. Size: 0–255 characters (default 1).
VARCHAR(size)	Variable-length string. Size: 0–65,535 characters.
BINARY(size)	Fixed-length binary string.
VARBINARY(size)	Variable-length binary string.
TINYTEXT	String with maximum length of 255 characters.
TEXT	String with maximum length of 65,535 bytes.
MEDIUMTEXT	String with maximum length of 16,777,215 characters.
LONGTEXT	String with maximum length of 4,294,967,295 characters.
TINYBLOB	Binary data up to 255 bytes.
BLOB	Binary data up to 65,535 bytes.

Data Type	Description
MEDIUMBLOB	Binary data up to 16,777,215 bytes.
LONGBLOB	Binary data up to 4,294,967,295 bytes.
ENUM(val1, val2, ...)	Allows only **one value** from a predefined list.
SET(val1, val2, ...)	Allows **multiple values** from a predefined list (max 64).

---

2. Numeric Data Types

Data Type	Description
BIT(size)	Bit-value type (1–64 bits).
TINYINT(size)	Very small integer (−128 to 127 / 0 to 255).
BOOL / BOOLEAN	TRUE or FALSE (0 = FALSE, non-zero = TRUE).
SMALLINT(size)	Small integer (−32,768 to 32,767).
MEDIUMINT(size)	Medium integer (−8,388,608 to 8,388,607).
INT(size)	Standard integer (−2,147,483,648 to 2,147,483,647).
INTEGER(size)	Same as INT.
BIGINT(size)	Very large integer values.
FLOAT(p)	Floating-point number (approximate value).
DOUBLE(size, d)	Double-precision floating-point number.
DECIMAL(size, d)	Exact fixed-point number (used for money).
DEC(size, d)	Same as DECIMAL.

Note:

- UNSIGNED → does not allow negative values
- ZEROFILL → fills remaining spaces with zero

---

3. Date and Time Data Types

Data Type	Description
DATE	Stores date (YYYY-MM-DD).
DATETIME(fsp)	Stores date & time (YYYY-MM-DD hh:mm:ss).
TIMESTAMP(fsp)	Stores timestamp (Unix time based).
TIME(fsp)	Stores time (hh:mm:ss).
YEAR	Stores year in four-digit format (1901–2155).

Note:
MySQL 8.0 **does not support two-digit YEAR format**.

Example Table Using Data Types

```
CREATE TABLE student (
  id INT PRIMARY KEY,
  name VARCHAR(50),
  gender ENUM('Male','Female'),
  marks DECIMAL(5,2),
  dob DATE,
  created_at TIMESTAMP
);
```

MySQL Functions – String Functions

MySQL provides many built-in **string functions** that are used to perform operations on string (text) data such as finding length, converting case, joining strings, extracting substrings, etc.

MySQL String Functions with Description & Examples

1. ASCII()

Returns the ASCII value of the first character.

```
SELECT ASCII('A');
-- Output: 65
```

---

2. CHAR_LENGTH() / CHARACTER_LENGTH()

Returns the length of a string in characters.

```
SELECT CHAR_LENGTH('Hello');
-- Output: 5
```

---

3. CONCAT()

Joins two or more strings.

```
SELECT CONCAT('My', 'SQL');

-- Output: MySQL
```

---

4. CONCAT_WS()

Joins strings with a separator.

```
SELECT CONCAT_WS('-', '2026', '01', '05');

-- Output: 2026-01-05
```

---

5. FIELD()

Returns the position of a value in a list.

```
SELECT FIELD('b', 'a', 'b', 'c');

-- Output: 2
```

---

6. FIND_IN_SET()

Finds the position of a string in a comma-separated list.

```
SELECT FIND_IN_SET('apple', 'banana,apple,orange');

-- Output: 2
```

---

7. FORMAT()

Formats a number with commas and decimal places.

```
SELECT FORMAT(1234567.89, 2);

-- Output: 1,234,567.89
```

---

8. INSERT()

Inserts a substring at a given position.

```
SELECT INSERT('HelloWorld', 6, 5, 'SQL');

-- Output: HelloSQL
```

---

9. INSTR()

Returns the position of first occurrence of a substring.

```
SELECT INSTR('MySQL Database', 'SQL');

-- Output: 3
```

---

10. LCASE() / LOWER()

Converts string to lowercase.

```
SELECT LOWER('MYSQL');

-- Output: mysql
```

---

11. LEFT()

Extracts characters from the left side.

```
SELECT LEFT('Database', 4);

-- Output: Data
```

---

12. LENGTH()

Returns the length of a string in bytes.

```
SELECT LENGTH('Hello');

-- Output: 5
```

---

13. LOCATE() / POSITION()

Finds the position of a substring.

```
SELECT LOCATE('SQL', 'MySQL');

-- Output: 3
```

---

14. LPAD()

Pads a string from the left.

```
SELECT LPAD('5', 3, '0');

-- Output: 005
```

---

15. LTRIM()

Removes leading spaces.

```
SELECT LTRIM('   Hello');

-- Output: Hello
```

---

16. MID() / SUBSTR() / SUBSTRING()

Extracts a substring.

```
SELECT SUBSTRING('Database', 1, 4);

-- Output: Data
```

---

17. REPEAT()

Repeats a string.

```
SELECT REPEAT('Hi ', 3);

-- Output: Hi Hi Hi
```

---

18. REPLACE()

Replaces part of a string.

SELECT REPLACE('Hello World', 'World', 'MySQL');

-- Output: Hello MySQL

---

19. REVERSE()

Reverses a string.

SELECT REVERSE('MySQL');

-- Output: LQSyM

---

20. RIGHT()

Extracts characters from the right side.

SELECT RIGHT('Database', 4);

-- Output: base

---

21. RPAD()

Pads a string from the right.

SELECT RPAD('SQL', 5, '*');

-- Output: SQL**

---

22. RTRIM()

Removes trailing spaces.

SELECT RTRIM('Hello   ');

-- Output: Hello

---

23. SPACE()

Returns a string of spaces.

SELECT SPACE(5);

-- Output: '   '

---

24. STRCMP()

Compares two strings.

SELECT STRCMP('ABC', 'ABD');

-- Output: -1

---

25. SUBSTRING_INDEX()

Returns substring before a delimiter.

SELECT SUBSTRING_INDEX('www.google.com', '.', 2);

-- Output: www.google

---

26. TRIM()

Removes spaces from both sides.

SELECT TRIM(' Hello ');

-- Output: Hello

---

27. UCASE() / UPPER()

Converts string to uppercase.

SELECT UPPER('mysql');

-- Output: MYSQL