My SQL
Data type

Objectives

1. Numeric
2. Date & time values
3. String
4. Json

1. Numeric

Integers

Integers are a subset of the real numbers. They are written without a fraction or a decimal component. Integers fall within a set Z = {..., -2, -1, 0, 1, 2, ...} Integers are infinite. Computers can practically work only with a subset of integer values, because computers have finite capacity. Integers are used to count discrete entities. We can have 3, 4, 6 cars, but we cannot have 3.33 cars. We can have 3.33 kilograms.

The following is a table of integer types in MySQL: TINYINT, MEDIUMINT and BIGINT are MySQL extensions to the SQL standard.

1. Numeric

Data type	Bytes	Minimum value	Maximum value
TINYINT	1	-128	127
SMALLINT	2	-32768	32767
MEDIUMINT	3	-8388608	8388607
INTEGER	4	-2147483648	2147483647
BIGINT	8	-9223372036854775808	9223372036854775807

1. Numeric

Floating point values

Floating point numbers represent real numbers in computing. Real numbers measure continuous quantities, like weight, height or speed.

MySQL has floating point types for approximate values: FLOAT and DOUBLE and fixed-point types for exact values: DECIMAL and NUMERIC.

FLOAT is a single precision floating point number. MySQL uses four bytes to store a FLOAT value. DOUBLE is a double precision floating point number. MySQL uses eight bytes to store a DOUBLE value.

1. Numeric

```
mysql> CREATE TABLE Numbers (Id TINYINT, Floats FLOAT, Decimals DECIMAL(3, 2));
```

We create a table in which we are going to store a few floats and decimals.

```
mysql> INSERT INTO Numbers VALUES (1, 1.1, 1.1), (2, 1.1, 1.1), (3, 1.1, 1.1);
```

We insert three rows into the newly created table.

```
mysql> SELECT * FROM Numbers;
+------+--------+----------+
| Id   | Floats | Decimals |
+------+--------+----------+
|    1 |    1.1 |     1.10 |
|    2 |    1.1 |     1.10 |
|    3 |    1.1 |     1.10 |
+------+--------+----------+
3 rows in set (0,00 sec)
```

This is how the table looks.

```
mysql> SELECT SUM(Floats), SUM(Decimals) FROM Numbers;
+-------------------+---------------+
| SUM(Floats)       | SUM(Decimals) |
+-------------------+---------------+
| 3.3000000715255737 |          3.30 |
+-------------------+---------------+
1 row in set (0,08 sec)
```

2. Date & Time Values

- Date time data types:
  - Date
  - Time
  - Datetime
  - Year
  - Timestamp

2. Date & Time Values

Date

The DATE is used to store dates. MySQL retrieves and displays date values in YYYY-MM-DD format. The supported range is from 1000-01-01 to 9999-12-31.

```
mysql> SELECT CURDATE();
+------------+
| CURDATE()  |
+------------+
| 2017-01-31 |
+------------+
1 row in set (0,00 sec)
```

The CURDATE() function returns the current date.

```
mysql> SELECT DATE('2017-01-31 12:01:00');
+----------------------------+
| DATE('2017-01-31 12:01:00') |
+----------------------------+
| 2017-01-31                 |
+----------------------------+
1 row in set (0,00 sec)
```

2. Date & Time Values

```
mysql> SELECT CURDATE();
+------------+
| CURDATE()  |
+------------+
| 2017-01-31 |
+------------+
1 row in set (0,00 sec)
```

The CURDATE() function returns the current date.

```
mysql> SELECT DATE('2017-01-31 12:01:00');
+-----------------------------+
| DATE('2017-01-31 12:01:00') |
+-----------------------------+
| 2017-01-31                  |
+-----------------------------+
1 row in set (0,00 sec)
```

The DATE() function returns the date part of the date and time value.

```
mysql> SELECT ADDDATE('2017-01-20', 8);
+--------------------------+
| ADDDATE('2017-01-20', 8) |
+--------------------------+
| 2017-01-28               |
+--------------------------+
1 row in set (0,00 sec)
```

2. Date & Time Values

The ADDDATE() function adds days to a date. It returns the calculated date.

```
mysql> CREATE TABLE Dates(Id TINYINT, Dates DATE);
mysql> INSERT INTO Dates VALUES(1, '2017-01-24');
mysql> INSERT INTO Dates VALUES(2, '2017/01/25');
mysql> INSERT INTO Dates VALUES(3, '20170126');
mysql> INSERT INTO Dates VALUES(4, '170127');
mysql> INSERT INTO Dates VALUES(5, '2017+01+28');
```

Dates are displayed in MySQL in one format, but we can use various date formats in our SQL statements. The YYYY-MM-DD is the standard format. It is possible to use any punctuation character between the date parts.

```
mysql> SELECT * FROM Dates;
```

```
+------+------------+
| Id   | Dates      |
+------+------------+
|    1 | 2017-01-24 |
|    2 | 2017-01-25 |
|    3 | 2017-01-26 |
|    4 | 2017-01-27 |
|    5 | 2017-01-28 |
+------+------------+
5 rows in set (0,00 sec)
```

We have used multiple formats to insert dates into the table. MySQL uses one format to display the dates.

```
mysql> INSERT INTO Dates VALUES (6, '10000-01-01');
ERROR 1292 (22007): Incorrect date value: '10000-01-01'
for column 'Dates' at row 1
```

In case we go beyond the range of supported date values an error occurs.

2. Date & Time Values

Time

The TIME data type is used to display time in MySQL. It shows values in HH:MM:SS format. MySQL retrieves and displays TIME values in 'HH:MM:SS' format or 'HHH:MM:SS' format for large hours values. The range is from -838:59:59 to 838:59:59. The hours part of the time format may be greater than 24. It is because TIME data type can be used to denote time intervals. This is also why we can have negative time values.

2. Date & Time Values

```
mysql> SELECT CURTIME();
+-----------+
| CURTIME() |
+-----------+
| 11:47:36  |
+-----------+
1 row in set (0,00 sec)
```

The CURTIME() function returns the current time.

```
mysql> SELECT TIMEDIFF('23:34:32', '22:00:00');
+----------------------------------+
| TIMEDIFF('23:34:32', '22:00:00') |
+----------------------------------+
| 01:34:32                         |
+----------------------------------+
1 row in set (0,02 sec)
```

The TIMEDIFF() function is used to subtract two time values.

```
mysql> SELECT TIME('2017-01-31 11:06:43');
+-----------------------------+
| TIME('2017-01-31 11:06:43') |
+-----------------------------+
| 11:06:43                    |
+-----------------------------+
1 row in set (0,00 sec)
```

We can use the TIME() function to extract the time part of the date and time value.

```
mysql> SELECT TIMEDIFF('211344', 201123);
+----------------------------+
| TIMEDIFF('211344', 201123) |
+----------------------------+
| 01:02:21                   |
+----------------------------+
1 row in set (0,00 sec)
```

We can write time values in different formats too. The first parameter is a time value in a string format without delimiters. The second is a time value specified as a number.

2. Date & Time Values

Datetime

The DATETIME values contain both date and time. MySQL retrieves and displays values in YYYY-MM-DD HH:MM:SS format. The supported range is from 1000-01-01 00:00:00 to 9999-12-31 23:59:59.

2. Date & Time Values

```
mysql> SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2017-10-17 18:20:53 |
+---------------------+
1 row in set (0,00 sec)
```

The NOW() function returns current datetime.

```
mysql> SELECT DAYNAME('2017@10@17 18@20@22');
+--------------------------------+
| DAYNAME('2017@10@17 18@20@28') |
+--------------------------------+
| Tuesday                        |
+--------------------------------+
1 row in set (0,02 sec)
```

MySQL displays date and time in only one format. But in our SQL statements, we can use different formats. Any punctuation character may be used as the delimiter between date parts or time parts. In our case, we have used the @ character.

2. Date & Time Values

Year

The YEAR is a data type used for representing years. MySQL displays YEAR values in YYYY format. It allows us to assign values to YEAR columns using either strings or numbers. The allowable range is from 1901 to 2155, or 0000. Illegal year values are converted to 0000.

```
mysql> SELECT YEAR(CURDATE()) AS 'Current year';
+--------------+
| Current year |
+--------------+
|         2017 |
+--------------+
1 row in set (0,02 sec)
```

In the above SQL statement, we have retrieved the current year.

2. Date & Time Values

Timestamp

A *timestamp* is a sequence of characters, denoting the date and/or time at which a certain event occurred. Timestamps are typically used for logging events. In MySQL we have a TIMESTAMP data type for creating timestamps. A TIMESTAMP column is useful for recording the date and time of an INSERT or UPDATE operation. It automatically sets to the date and time of the most recent operation if you do not give it a value yourself. The TIMESTAMP data type has a range of 1970-01-01 00:00:01 UTC to 2038-01-19 03:14:07 UTC.

2. Date & Time Values

Timestamp

Data type	Format
TIMESTAMP(14)	YYYYMMDDHHMMSS
TIMESTAMP(12)	YYMMDDHHMMSS
TIMESTAMP(10)	YYMMDDHHMM
TIMESTAMP(8)	YYYYMMDD
TIMESTAMP(6)	YYMMDD
TIMESTAMP(4)	YYMM
TIMESTAMP(2)	YY

2. Date & Time Values

The TIMESTAMP data type offers automatic initialisation and updating. We can restrict this data type to have only automatic initialisation or automatic update only.

```
mysql> CREATE TABLE Prices(Id TINYINT PRIMARY KEY, Price DECIMAL(8, 2), Stamp TIMESTAMP);
mysql> INSERT INTO Prices(Id, Price) VALUES(1, 234.34);
mysql> INSERT INTO Prices(Id, Price) VALUES(2, 344.12);
```

We create a table with a TIMESTAMP column. We insert two rows into the table. The Stamp column is not included in the SQL statements. MySQL automatically fills the column.

```
mysql> SELECT * FROM Prices;
+----+--------+---------------------+
| Id | Price  | Stamp               |
+----+--------+---------------------+
|  1 | 234.34 | 2017-01-31 12:12:25 |
|  2 | 344.12 | 2017-01-31 12:15:10 |
+----+--------+---------------------+
2 rows in set (0,00 sec)
```

3. String

- MySQL has the following string data types:
  - Char
  - Varchar
  - Binary and Varbinary
  - Blob
  - Text
  - Enum

3. String

Char

A CHAR is a fixed length character data type. It is declared with a length, CHAR(x), where x can be between 0 to 255. CHAR always uses the same amount of storage space per entry. In case we specify an item which is shorter than the declared length, the value is right-padded with spaces to the specified length. Trailing spaces are removed when the value is retrieved.

3. String

```
mysql> CREATE TABLE Chars(Id TINYINT PRIMARY KEY, Chars CHAR(3));
mysql> INSERT INTO Chars VALUES (1, 'a'), (2, 'ab'), (3, 'abc'), (4, 'abb');

In the above SQL code, we have created a Chars table which has one column of the CHAR data type. The maximum length is
set to three characters.
mysql> INSERT INTO Chars VALUES (5, 'abcd');
ERROR 1406 (22001): Data too long for column 'Chars' at row 1

Trying to insert a larger string than specified leads to an error.
mysql> SELECT * FROM Chars;
+------+-------+
| Id   | Chars |
+------+-------+
|    1 | a     |
|    2 | ab    |
|    3 | abc   |
|    4 | abb   |
+------+-------+
4 rows in set (0,00 sec)
```

3. String

Varchar

VARCHAR data types stores variable - length strings. The length of the string can be from 0 to 65535. VARCHAR values are not padded when they are stored. Trailing spaces are retained when values are stored and retrieved. Most shorter string data types are stored in this data type; for example emails, names of people, of merchandise, or addresses.

3. String

```
mysql> CREATE TABLE FirstNames(Id TINYINT, Firstname VARCHAR(20));
mysql> INSERT INTO FirstNames VALUES (1, 'Tom'), (2, 'Lucy'), (3, 'Alice'),
    -> (4, 'Robert'), (5, 'Timothy'), (6, 'Alexander');
```

We create a FirstNames table in which we store six first names.

```
mysql> SELECT Id, LENGTH(FirstName) AS Length FROM FirstNames;
+------+--------+
| Id   | Length |
+------+--------+
|    1 |      3 |
|    2 |      4 |
|    3 |      5 |
|    4 |      6 |
|    5 |      7 |
|    6 |      9 |
+------+--------+
6 rows in set (0,00 sec)
```

We can see that names in a VARCHAR column type are stored in variable length. This saves disk space.

3. String

Binary and varbinary

BINARY and VARBINARY are binary byte data types. They contain byte strings rather than character strings. They have no character sets. Sorting and comparison are based on the numeric values of the bytes in the values. The range of the BINARY data types is from 0 to 255. It stores values in fixed length. The range of the VARBINARY is from 0 to 65535.

3. String

Blob

A BLOB is a binary large object data type. It can hold a variable amount of binary data. It can be used to store binary data like images or documents. BLOB has four types:

Blog type	Range in bytes
TINYBLOB	0 - 255
BLOB	0 - 65535
MEDIUMBLOB	0 - 16777215
LONGBLOB	0 - 4294967295

3. String

Text

A TEXT datatype is used for storing large textual data. For example articles, blogs, or pages. TEXT values are best used when VARCHAR and other string-based data objects are insufficient to handle storing the desired amount of information.

Blog type	Range in bytes
TINYTEXT	0 - 255
TEXT	0 - 65535
MEDIUMTEXT	0 - 16777215
LONGTEXT	0 - 4294967295

3. String

Enum

The ENUM is a string object with a value chosen from a permitted list of values. They are enumerated explicitly in the column specification. We can insert only one value from the list.

```
CREATE TABLE Sizes(Size ENUM('S', 'M', 'L', 'XL', 'XXL'));

INSERT INTO SizeTable VALUES ('S'), ('L');

INSERT INTO Sizes VALUES ('Large');
ERROR 1265 (01000): Data truncated for column 'Size' at row 1
```

Since 'Large' was not mentioned in the list, we get an error message.

```
SELECT * FROM Sizes;
+------+
| Size |
+------+
| S    |
| L    |
+------+
2 rows in set (0,00 sec)
```

We have two regular values in the table.

4. Json

Json

Since MySQL 5.7.8, MySQL supports a native JSON data type.
JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write and for machines to parse and generate.
MySQL automatically validates JSON documents stored in JSON columns. Invalid documents produce an error. JSON documents stored in JSON columns are optimized for efficient access. JSON columns cannot have a default value.

4. Json

```
mysql> CREATE TABLE t1 (Doc JSON);
```

A table with a JSON column is created.

```
mysql> INSERT INTO t1 VALUES('{"chair": "5", "table": "4",
"lamp": "6"}');
```

A document is added to the table.

```
mysql> SELECT * FROM t1;
+-----------------------------------------+
| Doc                                     |
+-----------------------------------------+
| {"lamp": "6", "chair": "5", "table": "4"} |
+-----------------------------------------+
1 row in set (0,00 sec)
```

We show the contents of the table.

```
mysql> SELECT JSON_ARRAY('pen', 4, 'pencil', 2, 'rubber', 1);
+-----------------------------------------------+
| JSON_ARRAY('pen', 4, 'pencil', 2, 'rubber', 1) |
+-----------------------------------------------+
| ["pen", 4, "pencil", 2, "rubber", 1]          |
+-----------------------------------------------+
1 row in set (0,02 sec)
```

The JSON_ARRAY() function takes a list of values and transforms them into a JSON array.

```
mysql> SELECT JSON_OBJECT('pen', 4, 'pencil', 2, 'rubber', 1);
+-----------------------------------------------+
| JSON_OBJECT('pen', 4, 'pencil', 2, 'rubber', 1) |
+-----------------------------------------------+
| {"pen": 4, "pencil": 2, "rubber": 1}          |
+-----------------------------------------------+
1 row in set (0,00 sec)
```