PHP

PHP MySQL Database

Overview of MySQL Structure and Syntax [2]

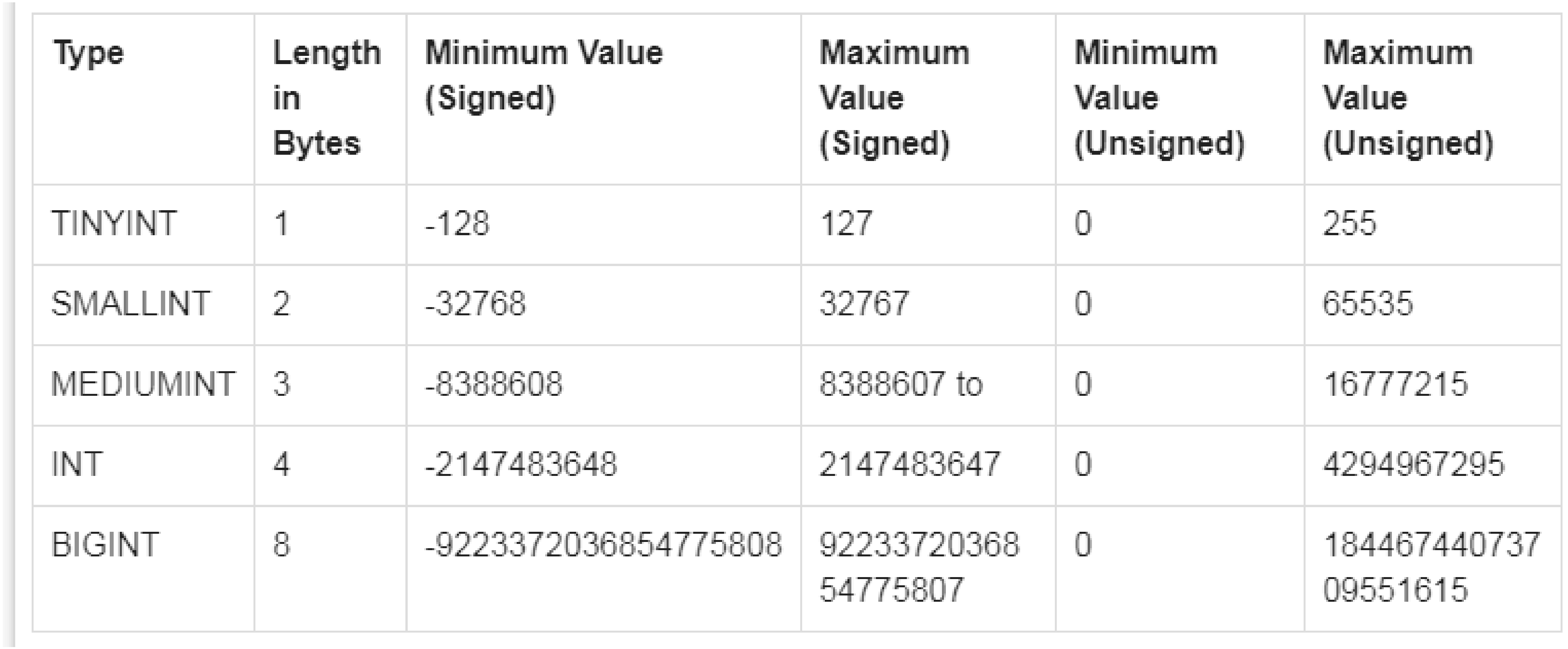
- MySQL is a relational database system, which basically means that it can store bits of information in separate areas and link those areas together.
- You can store virtually anything in a database.
- Information such as the contents of an address book, product catalog, or even a wish list of things you want for your birthday can be stored in your database.

Overview of MySQL Structure and Syntax [2]

- **MySQL Structure**
  - In MySQL, each table consists of separate *fields,* which represent each bit of information.
  - Fields can hold different types of data, such as text, numbers, dates, and so on.

Overview of MySQL Structure and Syntax [3]

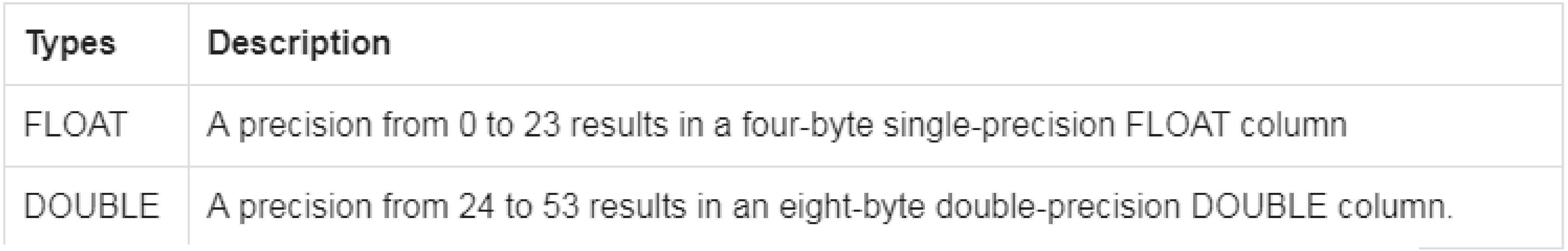
- **MySQL Structure**
  - **Field/Data Types**
    - **MySQL Numeric Types**
      - **Integer types**



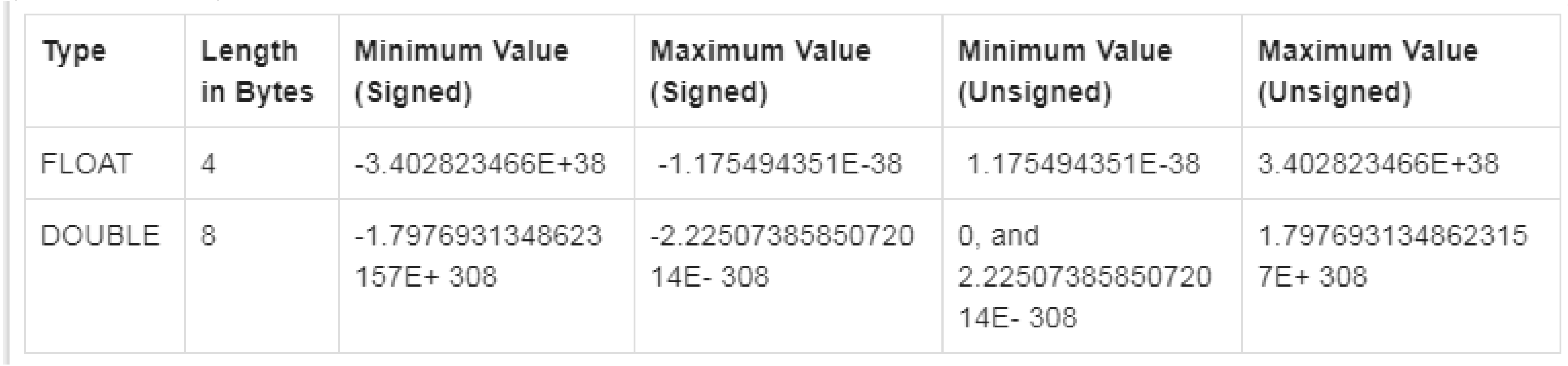
Type	Length in Bytes	Minimum Value (Signed)	Maximum Value (Signed)	Minimum Value (Unsigned)	Maximum Value (Unsigned)
TINYINT	1	-128	127	0	255
SMALLINT	2	-32768	32767	0	65535
MEDIUMINT	3	-8388608	8388607 to	0	16777215
INT	4	-2147483648	2147483647	0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807	0	18446744073709551615

Overview of MySQL Structure and Syntax [3]

- **MySQL Structure**
  - **Field/Data Types**
    - **MySQL Numeric Types → MySQL Numeric Types**
      - **Floating-Point Types**



Types	Description
FLOAT	A precision from 0 to 23 results in a four-byte single-precision FLOAT column
DOUBLE	A precision from 24 to 53 results in an eight-byte double-precision DOUBLE column.



Type	Length in Bytes	Minimum Value (Signed)	Maximum Value (Signed)	Minimum Value (Unsigned)	Maximum Value (Unsigned)
FLOAT	4	-3.402823466E+38	-1.175494351E-38	1.175494351E-38	3.402823466E+38
DOUBLE	8	-1.7976931348623 157E+ 308	-2.22507385850720 14E- 308	0, and 2.22507385850720 14E- 308	1.797693134862315 7E+ 308

Overview of MySQL Structure and Syntax [3]

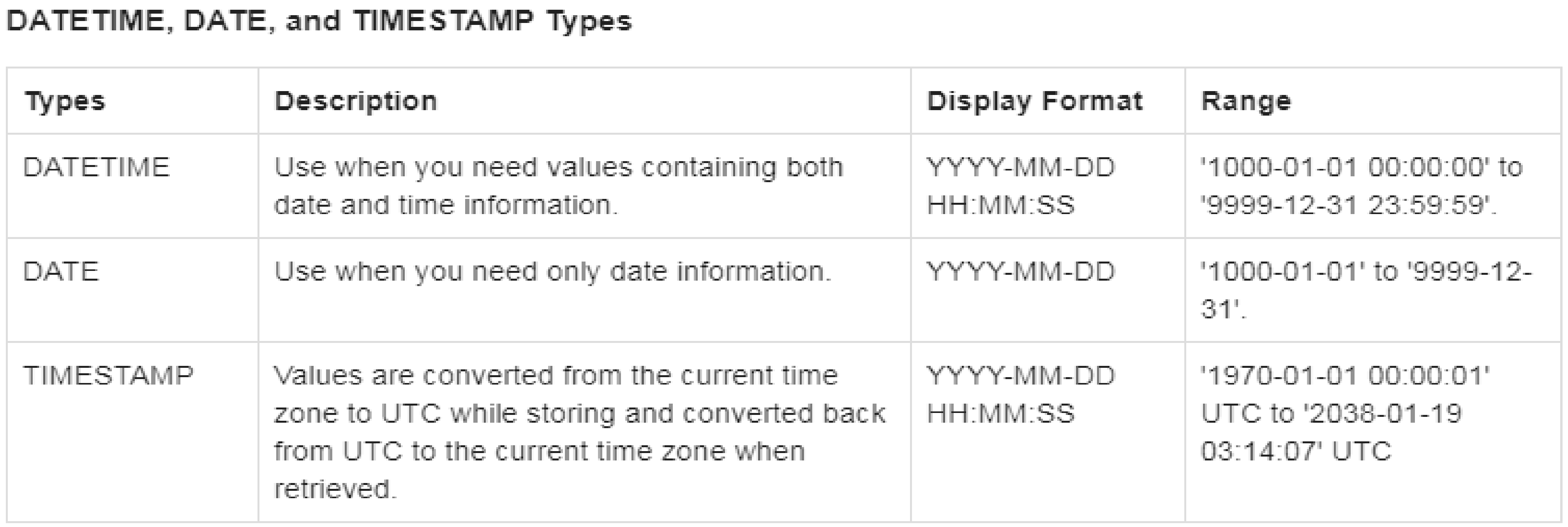
- **MySQL Structure**
  - **Field/Data Types → MySQL Numeric Types**
    - **Fixed-Point Types**
      - Fixed-Point data types are used to preserve exact precision, for example with currency data.
      - In MySQL DECIMAL and NUMERIC types store exact numeric data values.
      - MySQL 5.6 stores DECIMAL values in binary format.
      - In standard SQL the syntax DECIMAL(5,2) (where 5 is the precision and 2 is the scale.) be able to store any value with five digits and two decimals. Therefore the value range will be from -999.99 to 999.99.

Overview of MySQL Structure and Syntax [3]

- **MySQL Structure**
  - **Field/Data Types → MySQL Numeric Types**
    - **Bit Value Types**
      - The BIT data type is used to store bit-field values. A type of BIT(N) enables storage of N-bit values. N can range from 1 to 64.
      - To specify bit values, b'value' notation can be used. value is a binary value written using zeros and ones. For example, b'111' and b'10000000' represent 7 and 128, respectively

Overview of MySQL Structure and Syntax [3]

- **MySQL Structure**
  - **Field/Data Types**
    - **MySQL Date and Time Types**



DATETIME, DATE, and TIMESTAMP Types

Types	Description	Display Format	Range
DATETIME	Use when you need values containing both date and time information.	YYYY-MM-DD HH:MM:SS	'1000-01-01 00:00:00' to '9999-12-31 23:59:59'.
DATE	Use when you need only date information.	YYYY-MM-DD	'1000-01-01' to '9999-12-31'.
TIMESTAMP	Values are converted from the current time zone to UTC while storing and converted back from UTC to the current time zone when retrieved.	YYYY-MM-DD HH:MM:SS	'1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC

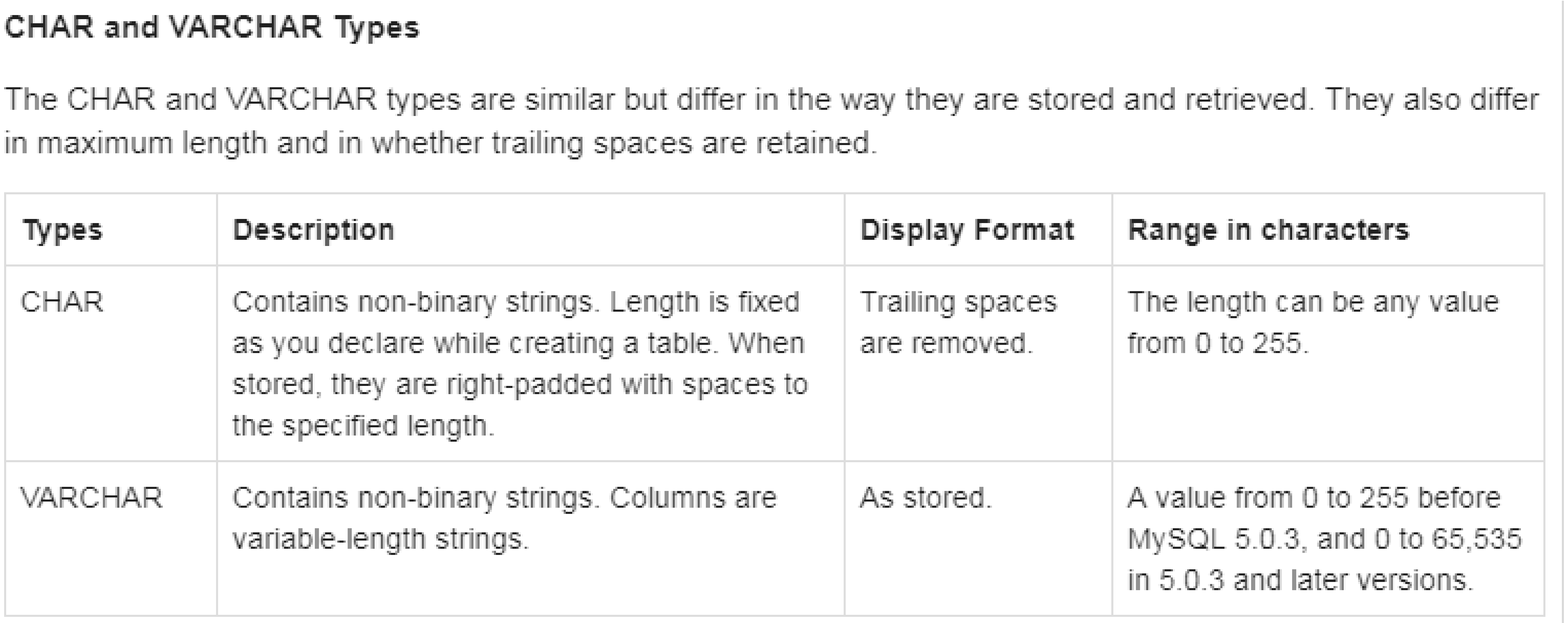
      - **TIME - 'HH:MM:SS' format or or 'HHH:MM:SS' format**
      - YEAR - 1-byte - YYYY format

Overview of MySQL Structure and Syntax [3]

- **MySQL Structure**
  - **Field/Data Types**
    - **String Types**
      - The string types are CHAR, VARCHAR, BINARY, VARBINARY, BLOB, TEXT, ENUM, and SET.

Overview of MySQL Structure and Syntax [3]

- **MySQL Structure**
  - **Field/Data Types**
    - **String Types**



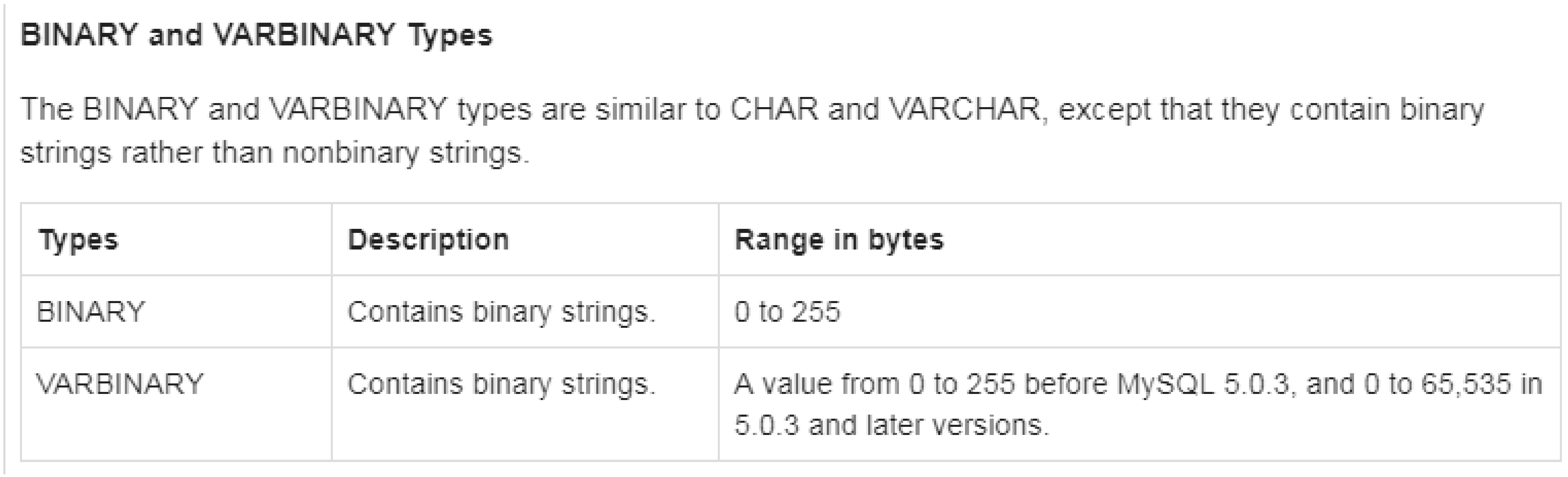
CHAR and VARCHAR Types

The CHAR and VARCHAR types are similar but differ in the way they are stored and retrieved. They also differ in maximum length and in whether trailing spaces are retained.

Types	Description	Display Format	Range in characters
CHAR	Contains non-binary strings. Length is fixed as you declare while creating a table. When stored, they are right-padded with spaces to the specified length.	Trailing spaces are removed.	The length can be any value from 0 to 255.
VARCHAR	Contains non-binary strings. Columns are variable-length strings.	As stored.	A value from 0 to 255 before MySQL 5.0.3, and 0 to 65,535 in 5.0.3 and later versions.

Overview of MySQL Structure and Syntax [3]

- **MySQL Structure**
  - **Field/Data Types**
    - **String Types**



BINARY and VARBINARY Types

The BINARY and VARBINARY types are similar to CHAR and VARCHAR, except that they contain binary strings rather than nonbinary strings.

Types	Description	Range in bytes
BINARY	Contains binary strings.	0 to 255
VARBINARY	Contains binary strings.	A value from 0 to 255 before MySQL 5.0.3, and 0 to 65,535 in 5.0.3 and later versions.

Overview of MySQL Structure and Syntax [3]

- **MySQL Structure**
  - **Field/Data Types**
    - **String Types**
      - **BLOB and TEXT Types**
        - A BLOB is a binary large object that can hold a variable amount of data. There are four types of BLOB, TINYBLOB, BLOB, MEDIUMBLOB, and LONGBLOB. These differ only in the maximum length of the values they can hold.
        - The four TEXT types are TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT. These correspond to the four BLOB types and have the same maximum lengths and storage requirements.

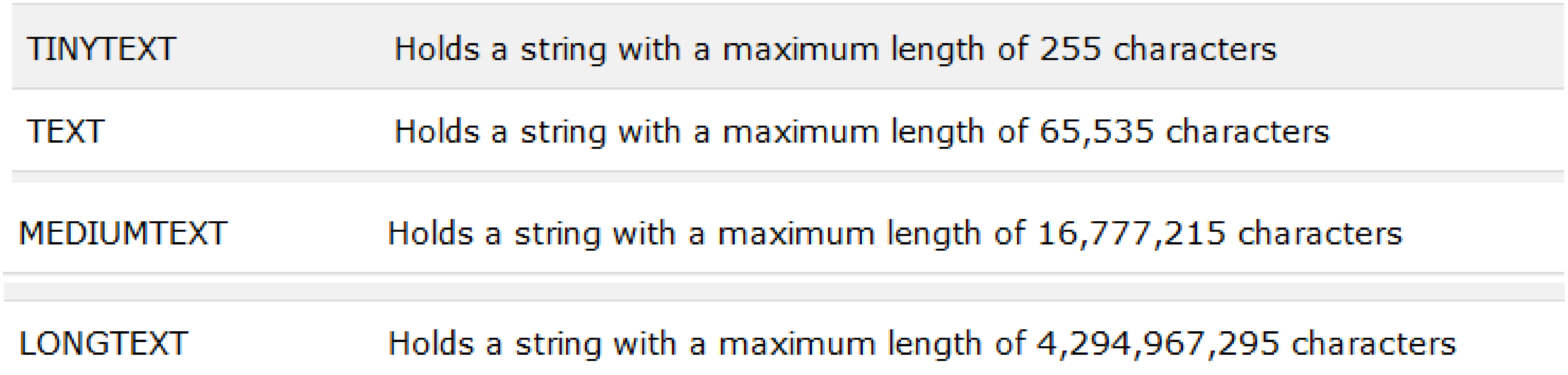
Overview of MySQL Structure and Syntax [3]

- **MySQL Structure**
  - **Field/Data Types**
    - **String Types**
      - **BLOB and TEXT Types**

BLOB	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data

Overview of MySQL Structure and Syntax [3]

- **MySQL Structure**
  - **Field/Data Types**
    - **String Types**
      - **BLOB and TEXT Types**



TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters

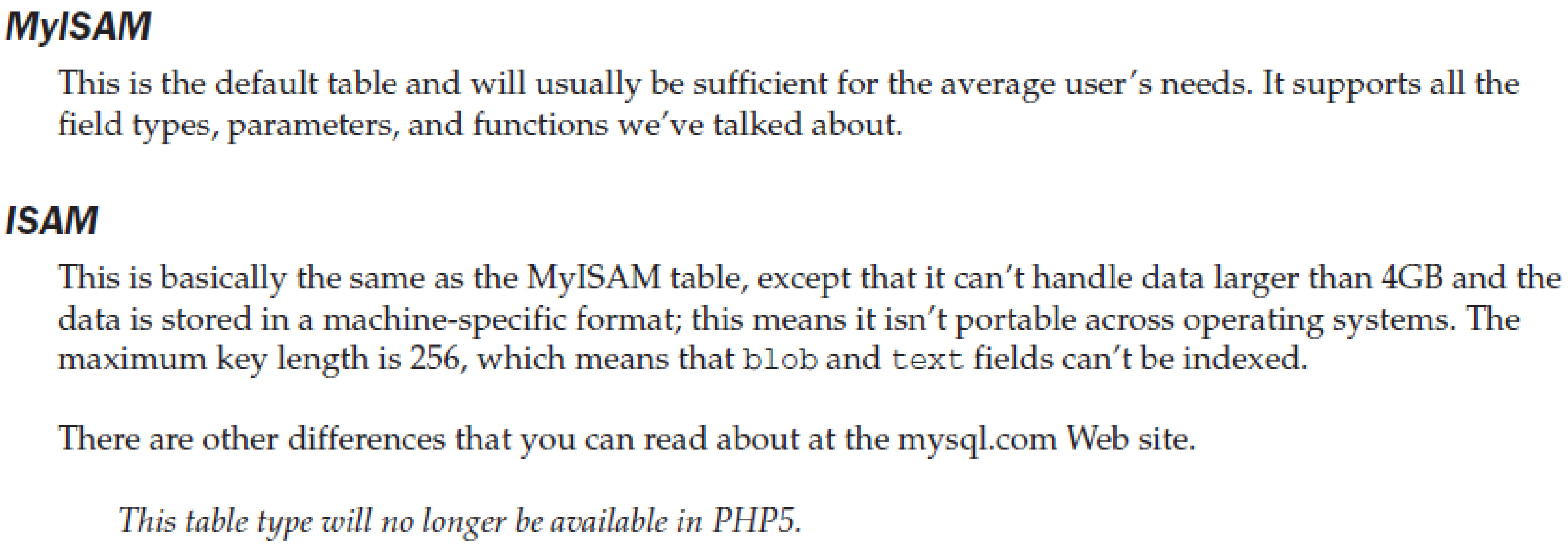
Overview of MySQL Structure and Syntax [3]

- **MySQL Structure**
  - **Field/Data Types**
    - **String Types**
      - **ENUM Types**
        - A string object whose value is chosen from a list of values given at the time of table creation.
        - **SET Types**
          - A string object having zero or more comma separated values (maximum 64). Values are chosen from a list of values given at the time of table creation.

Types of MySQL Tables [2]

- **There are five main types of tables in the current version of MySQL**
  - MyISAM
  - ISAM
  - HEAP
  - InnoDB
  - BDB

Types of MySQL Tables [2]



MyISAM

This is the default table and will usually be sufficient for the average user's needs. It supports all the field types, parameters, and functions we've talked about.

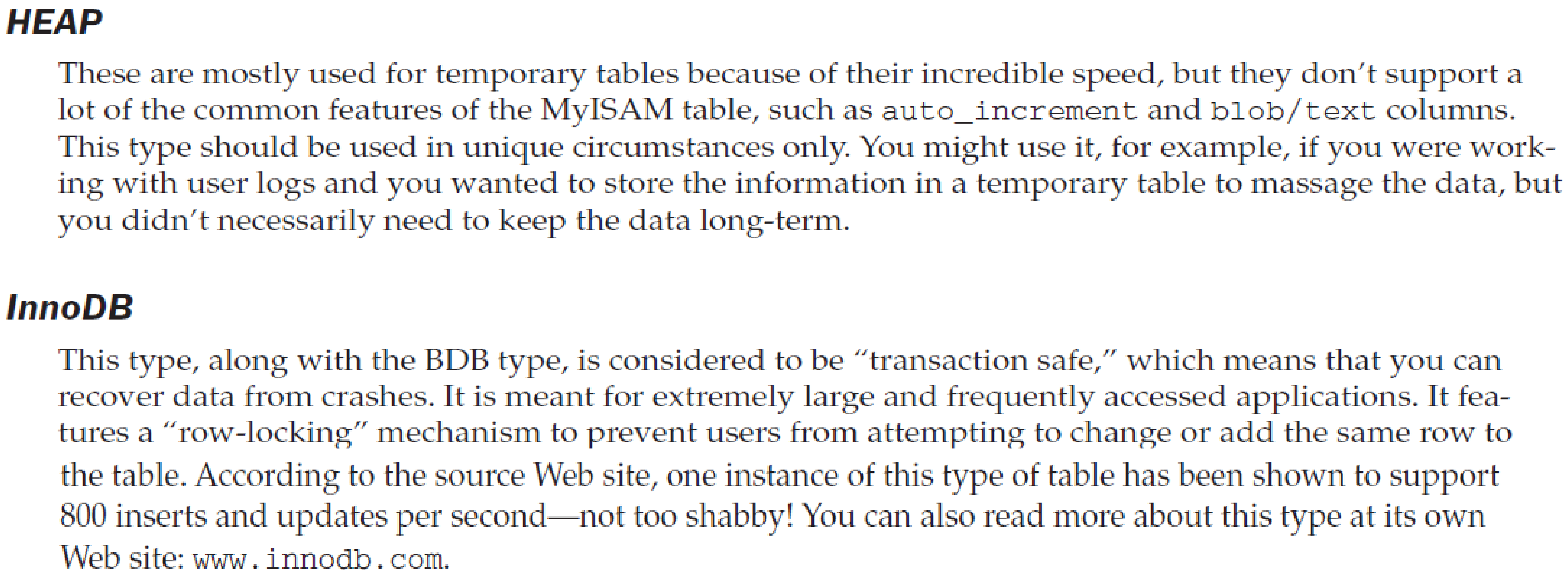
ISAM

This is basically the same as the MyISAM table, except that it can't handle data larger than 4GB and the data is stored in a machine-specific format; this means it isn't portable across operating systems. The maximum key length is 256, which means that `blob` and `text` fields can't be indexed.

There are other differences that you can read about at the mysql.com Web site.

This table type will no longer be available in PHP5.

Types of MySQL Tables [2]



HEAP

These are mostly used for temporary tables because of their incredible speed, but they don't support a lot of the common features of the MyISAM table, such as `auto_increment` and `blob/text` columns. This type should be used in unique circumstances only. You might use it, for example, if you were working with user logs and you wanted to store the information in a temporary table to massage the data, but you didn't necessarily need to keep the data long-term.

InnoDB

This type, along with the BDB type, is considered to be "transaction safe," which means that you can recover data from crashes. It is meant for extremely large and frequently accessed applications. It features a "row-locking" mechanism to prevent users from attempting to change or add the same row to the table. According to the source Web site, one instance of this type of table has been shown to support 800 inserts and updates per second—not too shabby! You can also read more about this type at its own Web site: `www.innodb.com`.

Types of MySQL Tables [2]

BDB

BDB, or BerkeleyDB, is the other type of table that is "transaction safe." It is actually its own entity that works closely with the MySQL server and can be downloaded from `www.sleepycat.com`. Like InnoDB tables, it is meant to support very large applications with literally thousands of users attempting to insert and update the same data at the same time. There is a complete reference manual available at its source Web site, which we invite you to read.

PHP Connect to MySQL [1]

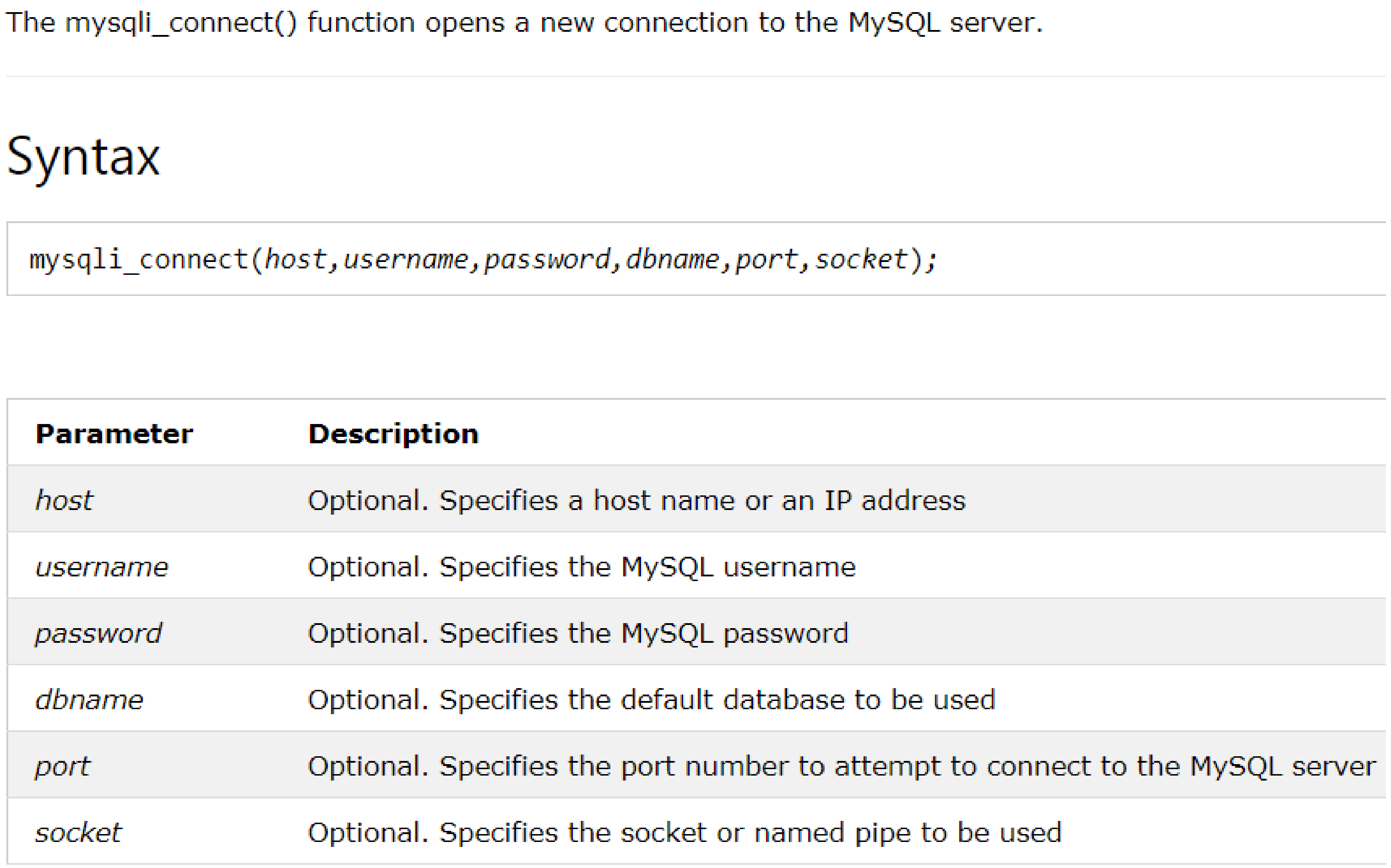
- PHP 5 and later can work with a MySQL database using:
  - **MySQLi extension** (the "i" stands for improved)
  - **PDO (PHP Data Objects)**
- Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.

- Example 1
- Example 2
- Example 3

Should I Use MySQLi or PDO? [1]

- If you need a short answer, it would be "Whatever you like".
- Both MySQLi and PDO have their advantages:
- PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.
- So, if you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.
- Both are object-oriented, but MySQLi also offers a procedural API.
- Both support Prepared Statements. Prepared Statements protect from SQL injection, and are very important for web application security.

mysqli_connect() Function



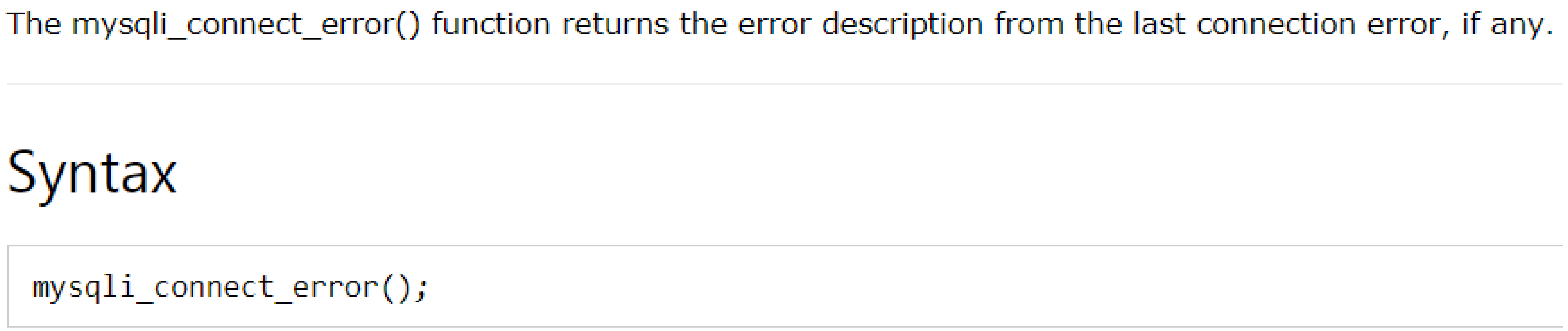
The mysqli_connect() function opens a new connection to the MySQL server.

Syntax

```
mysqli_connect(host,username,password,dbname,port,socket);
```

Parameter	**Description**
host	Optional. Specifies a host name or an IP address
username	Optional. Specifies the MySQL username
password	Optional. Specifies the MySQL password
dbname	Optional. Specifies the default database to be used
port	Optional. Specifies the port number to attempt to connect to the MySQL server
socket	Optional. Specifies the socket or named pipe to be used

mysqli_connect_error() Function

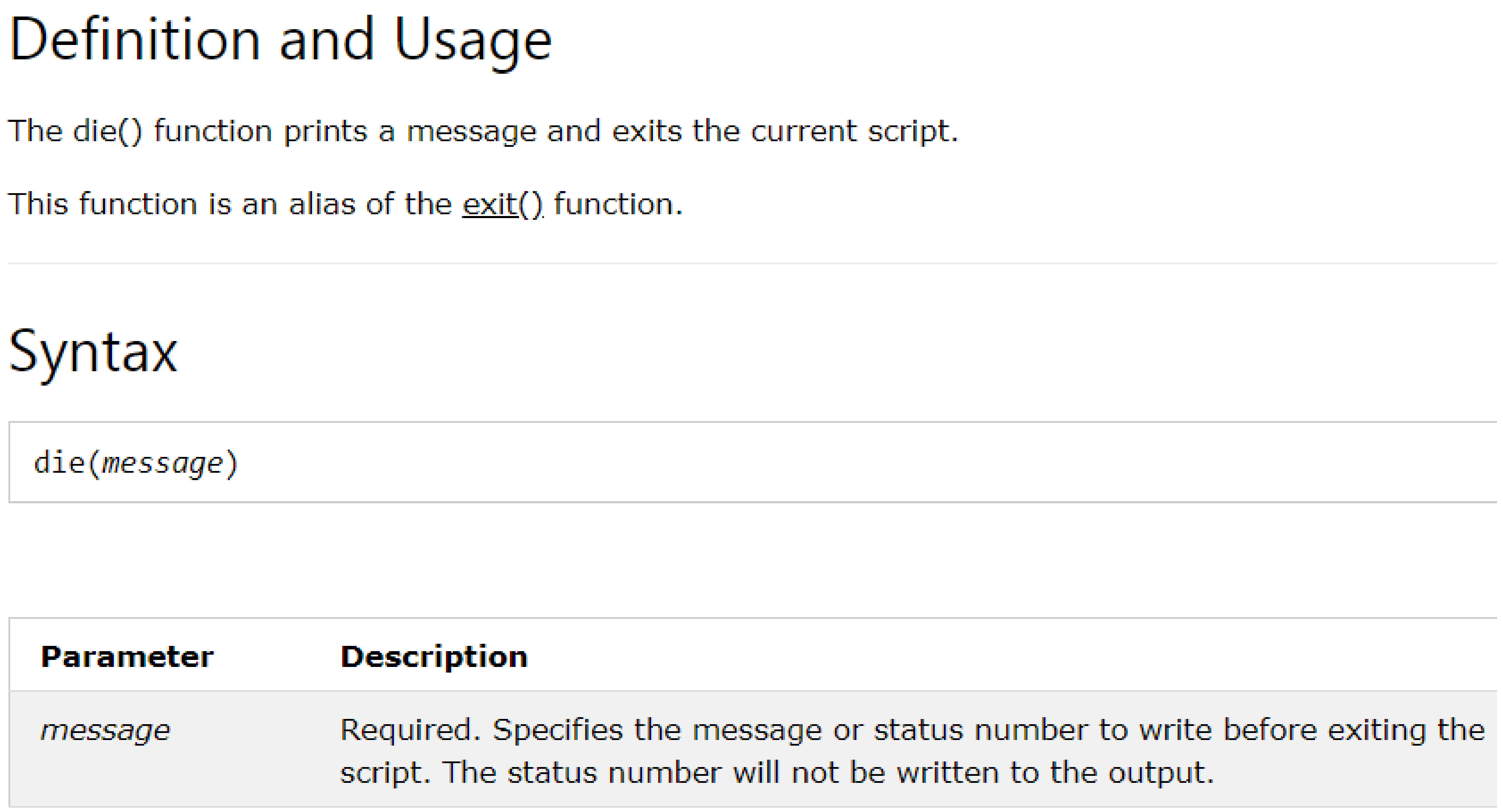


The mysqli_connect_error() function returns the error description from the last connection error, if any.

Syntax

```
mysqli_connect_error();
```

die() Function



Definition and Usage

The die() function prints a message and exits the current script.

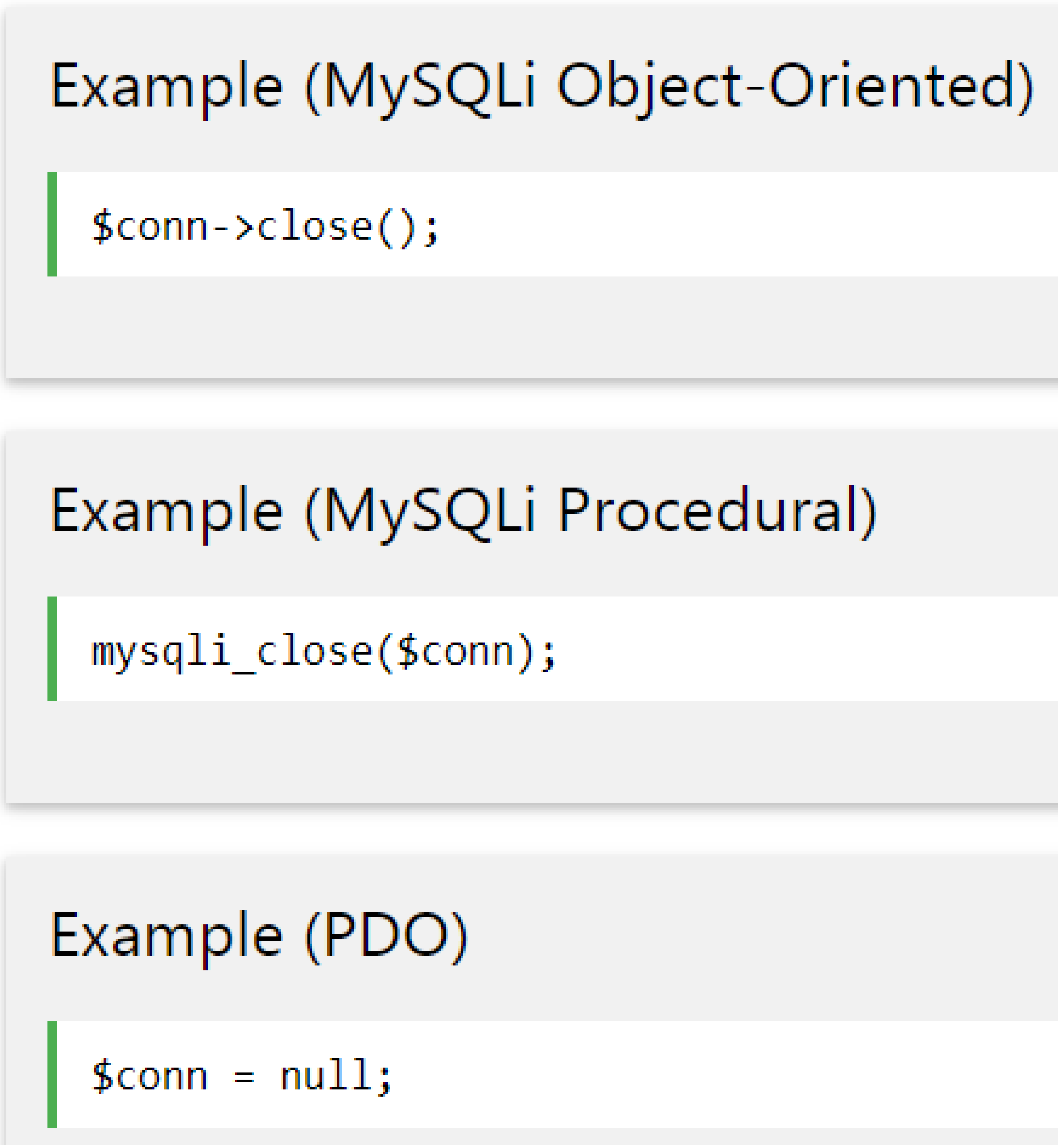
This function is an alias of the exit() function.

Syntax

```
die(message)
```

Parameter	Description
message	Required. Specifies the message or status number to write before exiting the script. The status number will not be written to the output.

Close the Connection [1]



Example (MySQLi Object-Oriented)

```
$conn->close();
```

Example (MySQLi Procedural)

```
mysqli_close($conn);
```

Example (PDO)

```
$conn = null;
```

Insert Data Into MySQL [1]

- Here are some syntax rules to follow:
  - The SQL query must be quoted in PHP
  - String values inside the SQL query must be quoted
  - Numeric values must not be quoted
  - The word NULL must not be quoted

- The INSERT INTO statement is used to add new records to a MySQL table:

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

Insert Data Into MySQL [1]

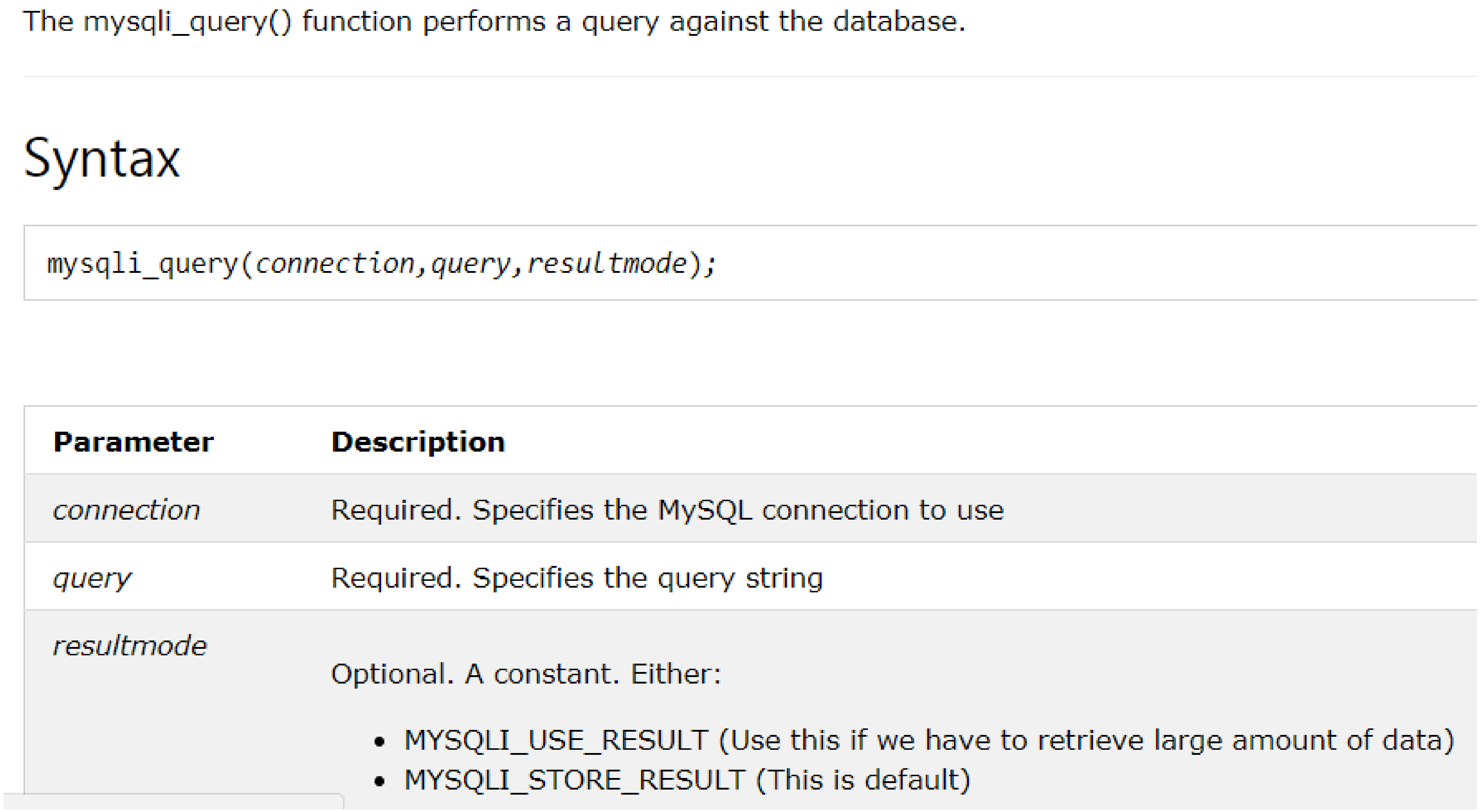
- Table MyGuests

```
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP)
```

Note: If a column is AUTO_INCREMENT (like the "id" column) or TIMESTAMP (like the "reg_date" column), it is no need to be specified in the SQL query; MySQL will automatically add the value.

- Example 4

mysqli_query() function [1]



The mysqli_query() function performs a query against the database.

Syntax

```
mysqli_query(connection,query,resultmode);
```

Parameter	Description
connection	Required. Specifies the MySQL connection to use
query	Required. Specifies the query string
resultmode	Optional. A constant. Either: • MYSQLI_USE_RESULT (Use this if we have to retrieve large amount of data) • MYSQLI_STORE_RESULT (This is default)

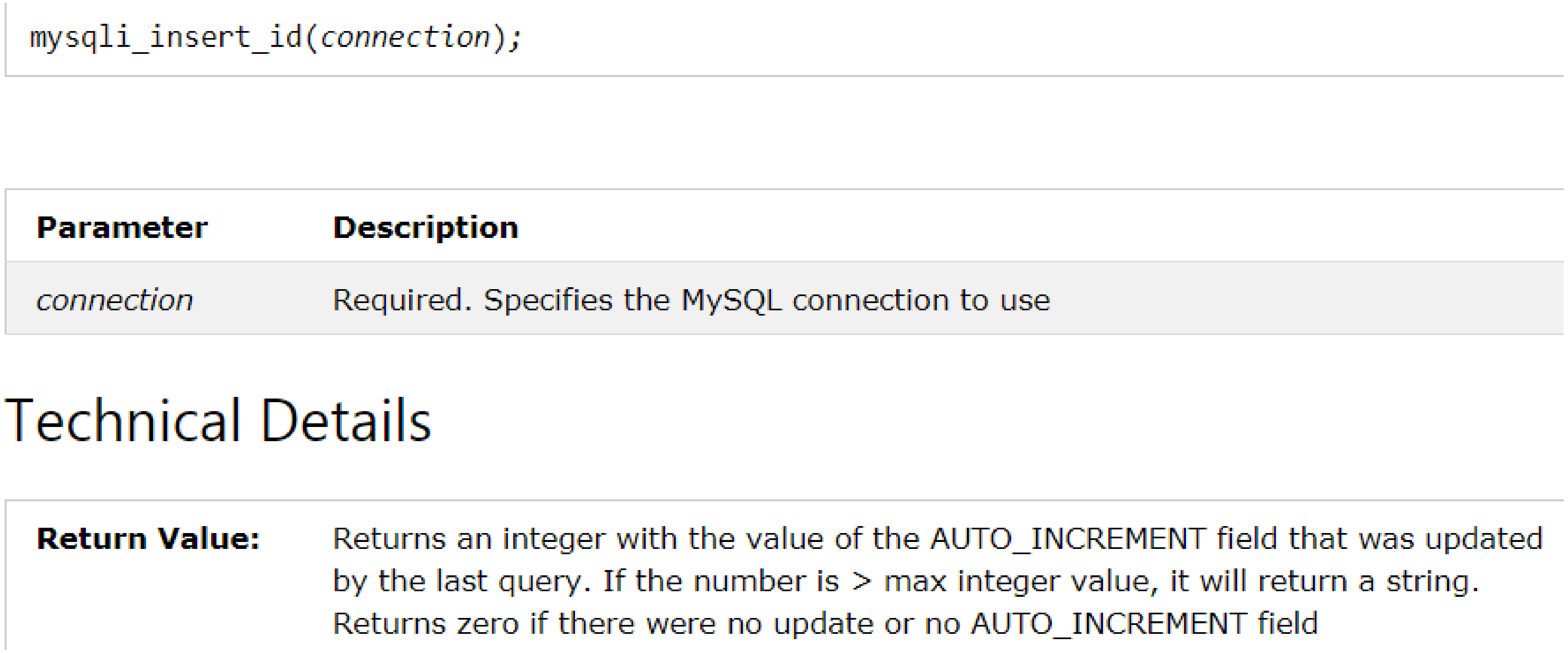
Return Value:	For successful SELECT, SHOW, DESCRIBE, or EXPLAIN queries it will return a **mysqli_result object**. For other successful queries it will return **TRUE. FALSE on failure**

Get ID of Last Inserted Record [1]

- If we perform an INSERT or UPDATE on a table with an AUTO_INCREMENT field, we can get the ID of the last inserted/updated record immediately.
- In the table "MyGuests", the "id" column is an AUTO_INCREMENT field:
- Example 5

mysqli_insert_id() function [1]

- It returns the id (generated with AUTO_INCREMENT) used in the last query.



```
mysqli_insert_id(connection);
```

Parameter	Description
connection	Required. Specifies the MySQL connection to use

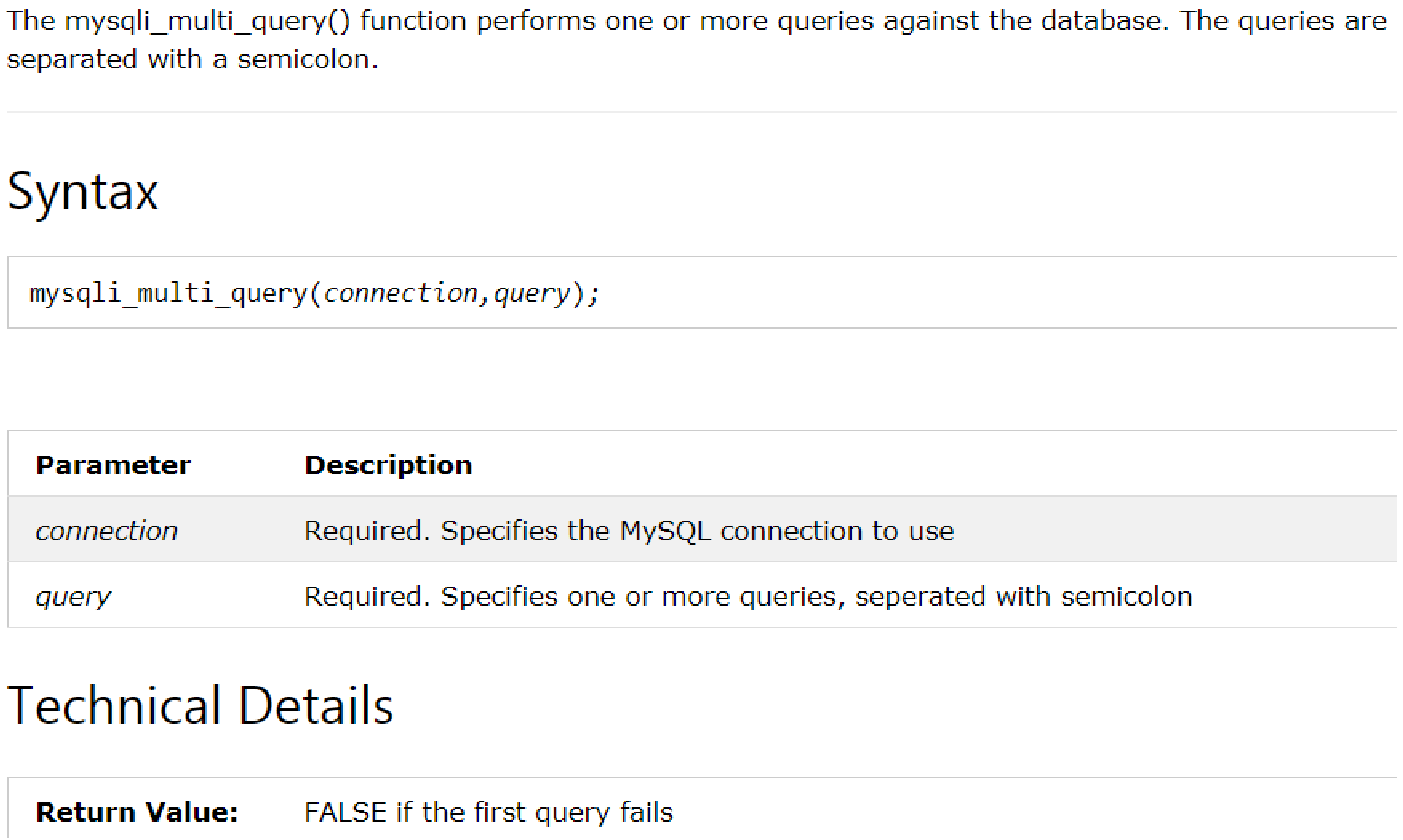
Technical Details

Return Value:	Returns an integer with the value of the AUTO_INCREMENT field that was updated by the last query. If the number is > max integer value, it will return a string. Returns zero if there were no update or no AUTO_INCREMENT field

Insert Multiple Records Into MySQL [1]

- Multiple SQL statements must be executed with the mysqli_multi_query() function.
- Note that each SQL statement must be **separated** by a **semicolon**.

mysqli_multi_query() [1]



The mysqli_multi_query() function performs one or more queries against the database. The queries are separated with a semicolon.

Syntax

```
mysqli_multi_query(connection,query);
```

Parameter	**Description**
connection	Required. Specifies the MySQL connection to use
query	Required. Specifies one or more queries, seperated with semicolon

Technical Details

Return Value:	FALSE if the first query fails

Select Data From MySQL [1]

- The SELECT statement is used to select data from one or more tables:

```
SELECT column_name(s) FROM table_name
```

- or we can use the * character to select ALL columns from a table:

```
SELECT * FROM table_name
```

Select Data From MySQL [1]

```
$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = mysqli_query($conn, $sql);
```

- The query puts the resulting data into a variable called $result (**mysqli_result object**).

- **The mysqli_num_rows()** function returns the number of rows in a result set.
- The **mysqli_fetch_assoc**() function fetches a result row as an associative array.
- **Note:** Fieldnames returned from this function are case-sensitive.
-

Delete Data From MySQL [1]

- The DELETE statement is used to delete records from a table:

```
DELETE FROM table_name
                    WHERE some_column = some_value
```

- **Notice the WHERE clause in the DELETE syntax:** The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

mysqli_affected_rows() [1]

- The mysqli_affected_rows() function returns the number of affected rows in the previous SELECT, INSERT, UPDATE, REPLACE, or DELETE query.
- Syntax:

  mysqli_affected_rows(*connection*)*;*
- **Return Value:** An **integer > 0** indicates the number of rows affected. **0 indicates** that no records were affected. **-1** indicates that the query returned an error

Update Data in MySQL [1]

- The UPDATE statement is used to update existing records in a table:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

- **Notice the WHERE clause in the UPDATE syntax:** The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

MySQLi Object-oriented

- Insert record
  - 5obj.html
  - 5obj.php
- Select record
  - 7obj.php
- Delete record
  - 8obj.html
  - 8obj.php
- Update record
  - 9obj.html
  - 9obj.php

Create a MySQL Database [1]

- Example 10.php

Create a MySQL Table [1]

- Example 11.php

Alter a MySQL Table [1]

- Modify column
- Add column
- Drop column
- Example 12.php

MySQL Table [1]

- Drop table
- Example 13.php

Drop database [1]

- Example 14.php

Procedure Creation and Call [1]

```
DROP PROCEDURE IF EXISTS Insert_Country;
DELIMITER $$
CREATE PROCEDURE Insert_Country
    (
      IN couname VARCHAR(50)
    )
BEGIN
   INSERT INTO country(cname) VALUES(couname) ;
END$$

DELIMITER ;
```

Example 15.html, 15.php

Procedure Creation and Call [1]

```
DROP PROCEDURE IF EXISTS Delete_Country;
DELIMITER $$
CREATE PROCEDURE Delete_Country
    (
      IN counid INT(11)
    )
BEGIN
   DELETE FROM Country WHERE cid= counid;
END$$

DELIMITER ;
```

Example 15.html, 15.php

Procedure Creation and Call [1]

```
DROP PROCEDURE IF EXISTS Update_Country;
DELIMITER $$
CREATE PROCEDURE Update_Country
    (
        IN counid INT(11),
        IN counname VARCHAR(50)
    )
BEGIN
   UPDATE Country SET cname=counname WHERE cid= counid;
END$$

DELIMITER ;
```

Example 15.html, 15.php

Procedure Creation and Call [1]

```
DROP PROCEDURE IF EXISTS Select_Country;
DELIMITER $$
CREATE PROCEDURE Select_Country
    ( )
BEGIN
   select * from country;
END$$

DELIMITER ;
```

Example 15.html, 15.php

Trigger [4]

Syntax:

```
CREATE TRIGGER trigger_name
BEFORE/AFTER    DELETE/UPDATE/INSERT
   ON table_name FOR EACH ROW

BEGIN

   -- variable declarations

   -- trigger code

END;
```

Before Delete Trigger [4]

Example:

```
DELIMITER //
CREATE TRIGGER country_before_delete
BEFORE DELETE
   ON country FOR EACH ROW
BEGIN
   -- Insert record into audit table
   INSERT INTO country_audit    ( cid,      cname,      deleted_date)    VALUES
   ( OLD.cid,      OLD.cname,      SYSDATE()    );
END; //

DELIMITER ;
```

References

1. https://www.w3schools.com/php/
2. BOOK - PHP Apache MYSQL Web Development
3. https://www.w3resource.com/mysql/mysql-data-types.php
4. https://www.techonthenet.com/mysql/triggers/

Thank you….