Siemens S7-1200/1500

TIA Portal (V13, V14, V15, V16)

MySQL Functionality

Implementation into user project

PDSql Library My v1.3.4

User guide

Contents

1 Introduction

PDSql Library My allows you to connect your PLC Siemens **S7-1200** or **S7-1500** system directly to **MySQL Database**. With this library you are able to read and write data from/to **MySQL Database**. As the communication between the **PLC** and the **MySQL Server** takes place directly you **don't need any PC** as a broker in the communication between the **PLC** and **MySQL server**.

1.1 Hardware requirements

PDSql library My was built for **PLC Siemens S7-1200** (**fw 4.2**+) and **S7-1500 (fw 1.8**+).

1.2 Software requirements

Main requirements for this library is **TIA Portal V13+** and **MySQL Server (5.x or higher)**. TIA Portal is available only with paid licence. **MySQL Community Server 8.0.19** is available as free version with some limitations -> https://dev.mysql.com/downloads/mysql/

There is also **MariaDB (5.x, 10.x) compatible with MySQL ->** https://downloads.mariadb.org/

1.3 Supported MySQL functions

This library allows you to execute all the basic MySQL commands like

- SELECT
- INSERT
- DELETE
- UPDATE
- Execute Stored PROCEDURE

However, in addition to this basic MySQL commands, **other MySQL commands can be also executed** depending on string content in query input. However, with some commands of this type, the query may end up with an error.

1.4 Supported MySQL data types

Supported are only datatypes listed in the Table 1-1. Any other datatype that is not listed in Table 1.1 will either convert its value to a string or cause the query to quit with an error.

Table 1-1: MySQL data types

Datatype		S7-1200	S7-1500
MySQL	PLC		
bit	Bool	•	•
text	String(1)	•	•
tinytext	String(1)	•	•
varchar	String(1)	•	•
char	String(1)	•	•
tinyint	DInt/LInt	• [2]	• [3]
smallint	DInt/LInt	• [2]	• [3]
Int	DInt/LInt	• [2]	• [3]
mediumint	DInt/LInt	• [2]	• [3]
bigint	DInt/LInt	• [2]	• [3]
float	LReal	•	•
double	LReal	•	•
real	LReal	•	•
decimal	LReal	•	•
datetime	DTL	•	•
date	DTL	•	•
time	DTL	•	•
timestamp	DTL	•	•
year	DTL	•	•

1 The default is a maximum string length of 50 characters
* If the string length of the SQL data type is greater than the string length in the PLC, so the rest of the text will be cut off.

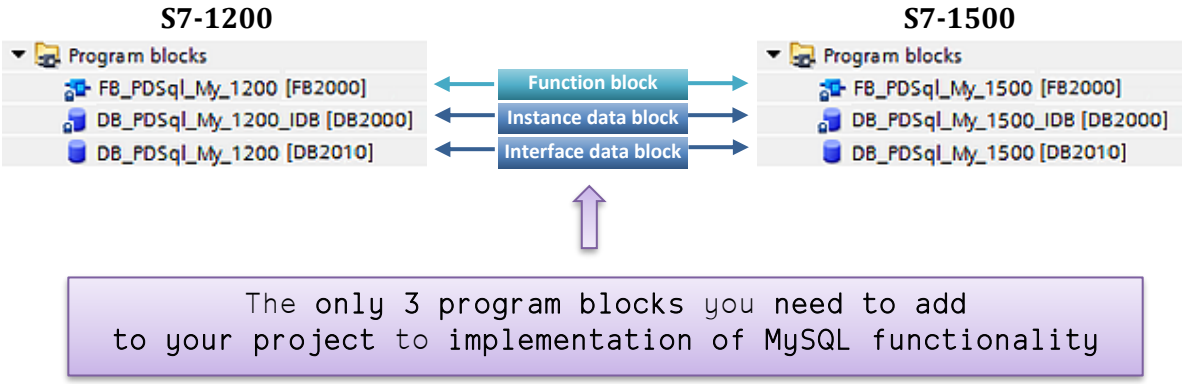
2 PLC data type DInt

3 PLC data type LInt

1.5 Program blocks

1.5.1 Function block FB_PDSql_My

The whole concept of this library was built on the ease of implementation into user projects. Therefore, all functions for communication with **MySQL Server** are contained in only one function block **FB_PDSql_My**. This block is also associated with one instance of the data block **DB_PDSql_My_IDB**. The last one is the data block **DB_PDSql_My** which provides all parameters and the interface between the user application and the **PDSql library**. It contains the resulting SQL table obtained after SELECT command. These blocks are supplied in **2 versions** for **S7-1200** and **S7-1500**.



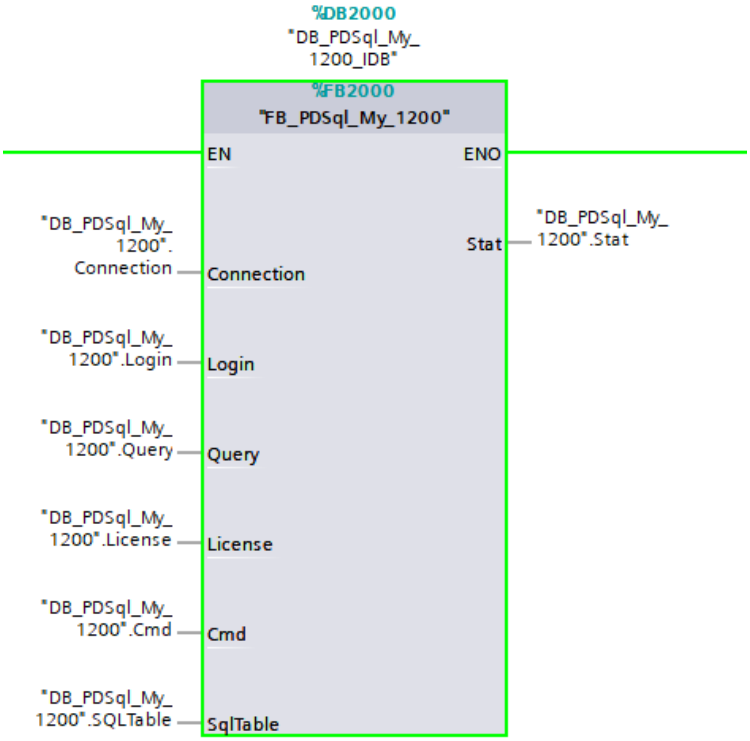


Table 1-2 : **FB_PDSql_My** interface

Type	Name	Data type	Comment
Input	Connection	tConnection	MySQL server connection parameters
	Login	tLogin	Login parameters
	Licence	tLicence	Runtime license key
	Query	tQuery	MySQL query string
Output	Stat	tStat	Status bits, messages and errors
InOut	Cmd	tCmd	Commands **Connect**, **Disconnect**, **ExecuteQuery**
	SqlTable	tSqlTable (1200/1500)	MySQL parameters and table contents as a result of the SELECT statement

Recommended steps for executing MySQL commands from a user application

1. Create a string of MySQL statements and write it to ***Query***
2. Set the ***Cmd.ExecuteQuery*** signal to **TRUE**
3. Wait for signal ***Stat.ExecutedOK*** or ***Stat.Error***
4. If the ***Stat.Error*** signal is **TRUE**, repeat the action from step 1.
   If the ***Stat.ExecutedOK*** signal is **TRUE** application can continue to prepare another MySQL statement if required and repeat from step 1.

The user application can fully control the ***Connect*** or ***Disconnect*** signals, but it is sufficient to control only ***ExecuteQuery*** signal which ensure the connection automatically if needed.

1.5.2 Data block DB_PDSql_My

Data block **DB_PDSql_My** is composed of all the data types described in chapter 1.6. All communication between the user application and the MySQL server is done through this data block.

S7-1200

DB_PDSql_My_1200

	Name	Data type
1	Static	
2	Connection	"tConnection"
3	Licence	"tLicence"
4	Cmd	"tCmd"
5	Stat	"tStat"
6	Login	"tLogin"
7	Query	"tQuery"
8	SQLTable	"tSqlTable_1200"

S7-1500

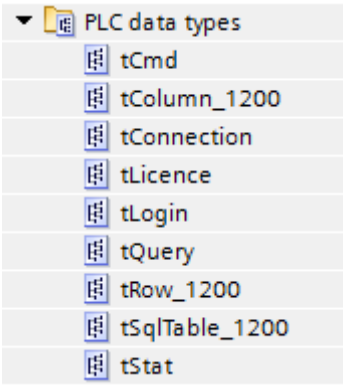
DB_PDSql_My_1500

	Name	Data type
1	Static	
2	Connection	"tConnection"
3	Licence	"tLicence"
4	Cmd	"tCmd"
5	Stat	"tStat"
6	Login	"tLogin"
7	Query	"tQuery"
8	SQLTable	"tSqlTable_1500"

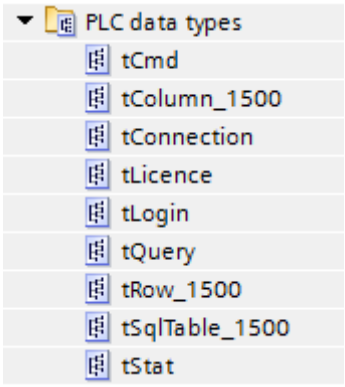
1.6 PLC data types

Function block **FB_PDSql_My** and data block **DB_PDSql_My** uses several custom PLC data types to communicate with user application. The following is a list and description of these data types used in the library for **S7-1200** and **S7-1500**.

S7-1200



S7-1500



1.6.1 tConnection

This data type provides path parameters to connect to MySQL Server.

Name	Data type	Default value	Comment
IP1	USInt	192	Octet 1 of the SQL Server's IP address
IP2	USInt	168	Octet 2 of the SQL Server's IP address
IP3	USInt	1	Octet 3 of the SQL Server's IP address
IP4	USInt	1	Octet 4 of the SQL Server's IP address
Port	UDInt	3306	Remote port of the MySQL Server. MySQL default is 3306.
ConnID	CONN_OUC	1	Reference to this connection. The parameter uniquely identifies a connection within the PLC.
HW	HW_INTERFACE	64	Hardware ethernet interface of the PLC used to the communication. X1(64), X2(72)
Timeout	Time	T#2s	Maximum time allowed to wait for connect, disconnect and execute query command. After this time elapsed, the error status is triggered.

1.6.2 tCmd

This data type provides command triggers to the Function block.

Name	Data type	Default value	Comment
Connect [1]	Bool	False	Trigger to connect to MySQL server
Disconnect [1]	Bool	False	Trigger to disconnect from MySQL server
ExecuteQuery [1]	Bool	False	Trigger to execute MySQL query. If the MySQL server has not been connected before activating this signal, the connection process will start automatically and then the MySQL command will be executed after a successful connection to the sql server. **So the user application control can use only this one trigger signal to connect and execute commands automatically**.

[1] The bit signal is active in TRUE and triggered on the rising edge. Function block always resets this bit.

1.6.3 tLogin

This data type provides login parameters to the SQL Server.

Name	Data type	Default value	Comment
HostName	String	''	Freely definable name (can be empty)
UserName	String	''	The user name of MySQL Server account
Password	String	''	The password of MySQL Server account
Database	String	''	The name of database (not used / can be empty)

1.6.4 tStat

This data type provides status signals from the Function block.

Name	Data type	Default value	Comment
Connected	Bool	False	TRUE when logged to MySQL server successfully
Busy	Bool	False	TRUE during execution of MySQL statement
ExecutedOK	Bool	False	TRUE when MySQL statement was executed successfully
Error	Bool	False	TRUE when MySQL statement ended with an error
Status	Word	W#16#0000	Info or error code
AffectedRows	UDInt	0	Number of affected rows after executing the last MySQL command
Message	String	''	Info or error message after the last MySQL command executed

1.6.5 tLicense

This data type provides runtime license key.

Name	Data type	Default value	Comment
LicKey1	DWord	16#00	License Key part 1
LicKey2	DWord	16#00	License Key part 2
LicKey3	DWord	16#00	License Key part 3
LicKey4	DWord	16#00	License Key part 4
LicKey5	DWord	16#00	License Key part 5

1.6.6 tQuery

This data type provides MySQL query string.

Name	Data type	Default value	Comment
Query	Array[1..2] of String	''	Array of MySQL Query

1.6.7 tRow_1200

This data type provides row data for sql table result (S7-1200 version).

Name	Data type	Default value	Comment
Bool	Bool	False	Bool value
Null	Bool	False	NULL value
DInt	DInt	0	DInt value
LReal	LReal	0.0	LReal value
String	String[50]	''	String value
Datetime	DTL	DTL#1970-01-01-00:00:00	Datetime value

1.6.8 tRow_1500

This data type provides row data for sql table result (S7-1500 version).

Name	Data type	Default value	Comment
Bool	Bool	False	Bool value
Null	Bool	False	NULL value
LInt	LInt	0	LInt value
LReal	LReal	0.0	LReal value
String	String[50]	''	String value
Datetime	DTL	DTL#1970-01-01-00:00:00	Datetime value

1.6.9 tColumn_1200

This data type provides column data for sql table result (S7-1200 version).

Name	Data type	Default value	Comment
Name	String	''	Column's name
CharacterSet	UInt	0	Character set
ColumnLength	UDint	0	Length
Type	Byte	16#0	Data type
Flags	Word	16#0	Specific flags
Decimals	Byte	16#0	Number of decimal places
Rows	Array[1..10] of "tRow_1200"		Array of rows

1.6.10 tColumn_1500

This data type provides column data for sql table result (S7-1500 version).

Name	Data type	Default value	Comment
Name	String	''	Column's name
CharacterSet	UInt	0	Character set
ColumnLength	UDint	0	Length
Type	Byte	16#0	Data type
Flags	Word	16#0	Specific flags
Decimals	Byte	16#0	Number of decimal places
Rows	Array[1..10] of "tRow_1500"		Array of rows

1.6.11 tSqlTable_1200

This data type provides main parameters and sql table result (S7-1200 version).

Name	Data type	Default value	Comment
ColumnCount	UInt	0	The number of readed columns
RowCount	UInt	0	The number of readed rows
MaxColumns ⚠	UInt	10	Maximum allowed number of columns. The value must be less than or equal to ***tSqlTable.Columns*** array size
MaxRows ⚠	UInt	10	Maximum allowed number of rows. The value must be less than or equal to ***tColumn.Rows*** array size
Columns	Array[1..10] of "tColumn_1200"		Array of columns

1.6.12 tSqlTable_1500

This data type provides main parameters and sql table result (S7-1500 version).

Name	Data type	Default value	Comment
ColumnCount	UInt	0	The number of readed columns
RowCount	UInt	0	The number of readed rows
MaxColumns ⚠	UInt	10	Maximum allowed number of columns. The value must be less than or equal to ***tSqlTable.Columns*** array size
MaxRows ⚠	UInt	10	Maximum allowed number of rows. The value must be less than or equal to ***tColumn.Rows*** array size
Columns	Array[1..10] of "tColumn_1500"		Array of columns

⚠ The ***MaxColumns*** parameter must be less than equal to the array size of ***SqlTable.Columns*** and the ***MaxRows*** parameter must be less than equal to the array size of ***tColumn.Rows*** otherwise the PLC may go to STOP

1.7 Errors

Here is the list of errors written to the **tStat.Status** variable as output from the function block **FB_PDSql**. During the error code, a more detailed description of the error is written to the **tStat.Message.**

Table 1.3

Code	Error message	Tips
W#16#0000	No error	
W#16#8001	Cannot connect to the remote server	Check IP address and port
W#16#8002	Conection timeout	Check IP address and port
W#16#8003	Login failed	See details in tStat.Message
W#16#8004	Disconection timeout	Something is wrong with ethernet connection
W#16#8005	MySQL Command execution timeout	MySQL Server does not respond
W#16#8006	MySQL Packet error	See details in tStat.Message
W#16#8007	MySQL Command execution error	See details in tStat.Message
W#16#8080	Invalid license key	Check the license key

1.8 License / Demo

The license key is binded to the PLC serial number. After creating a license key with an online license tool on the webite **www.plcdirectsql.com**, these 5 license numbers need to be written to **LicKey1-5**. It is recommended to write them as the start value so that they remain written after the PLC restarts.

For the first **2 hours** after switching on the power supply of the PLC device, the library is **fully functional** even without a valid license key and works in **demo** mode.

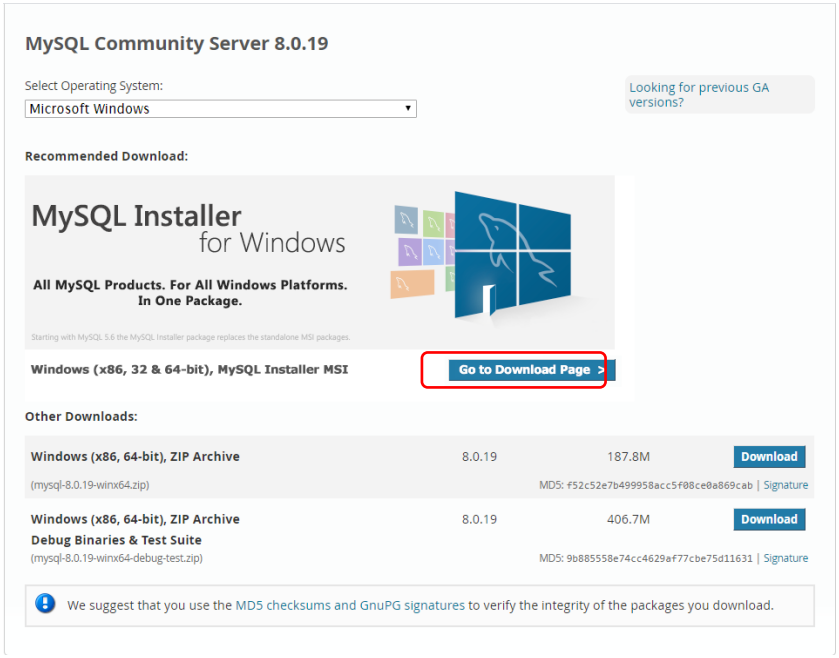
Name	Data type	Start value
Static		
Connection	"tConnection"	
Licence	"tLicence"	
LicKey1	DWord	16#F292_D938
LicKey2	DWord	16#9A3F_E367
LicKey3	DWord	16#12DA_4E93
LicKey4	DWord	16#E388_F3A7
LicKey5	DWord	16#E07E_0D2D
Cmd	"tCmd"	
Stat	"tStat"	
Login	"tLogin"	
Query	"tQuery"	
SQLTable	"tSqlTable"	

2 Instalation

2.1 MySQL Community Server 8.0.19

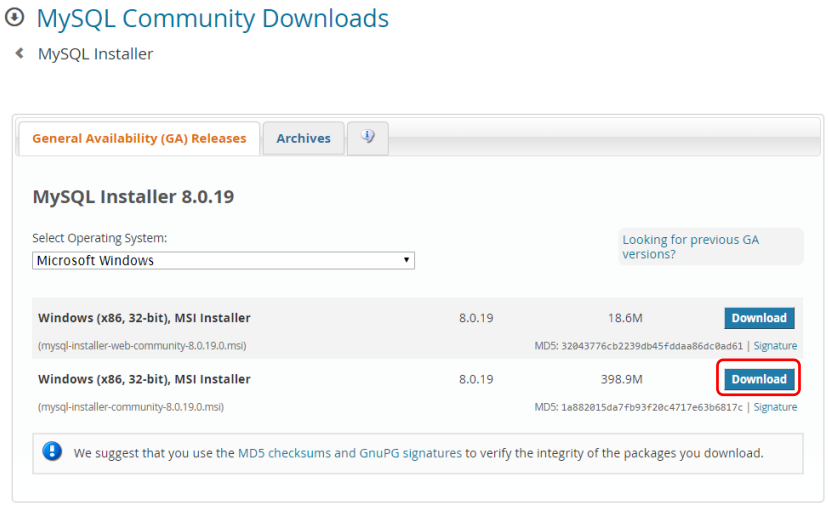
This chapter describes how to install **MySQL Community Server 8.0.19** and set it up for communicating with the PDSql library. The link to download the instalation package is here -> https://dev.mysql.com/downloads/mysql/

1



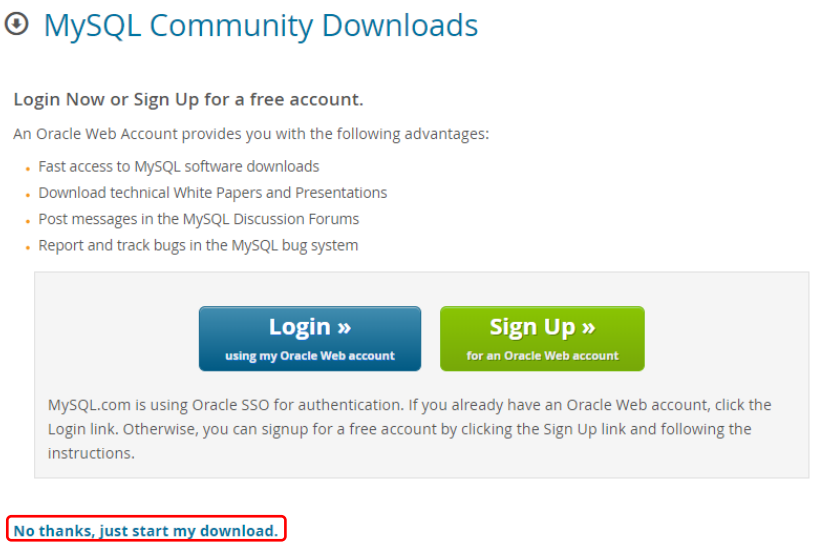
→ **„Go to Download Page >“**

2



→ **„Download“**

3

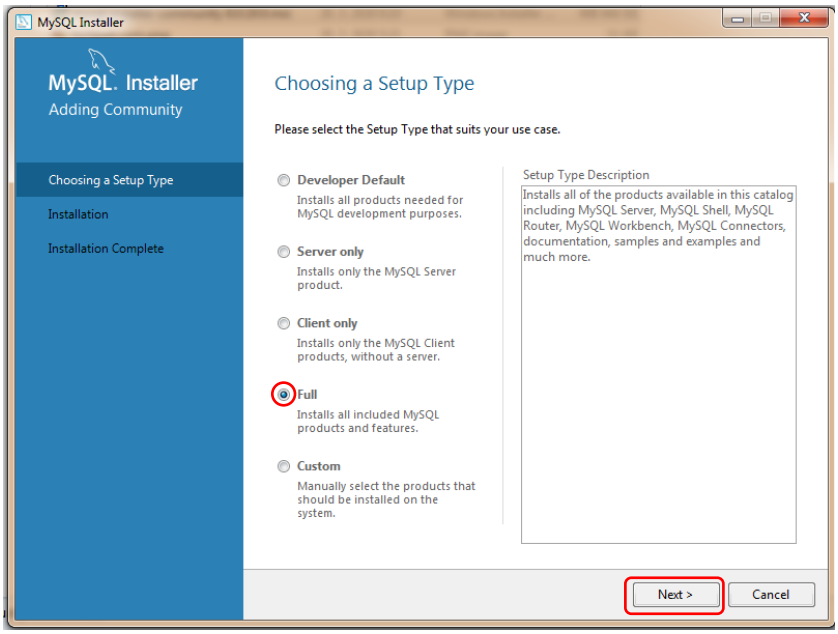


→ **„No thanks, just start my download.“**

4

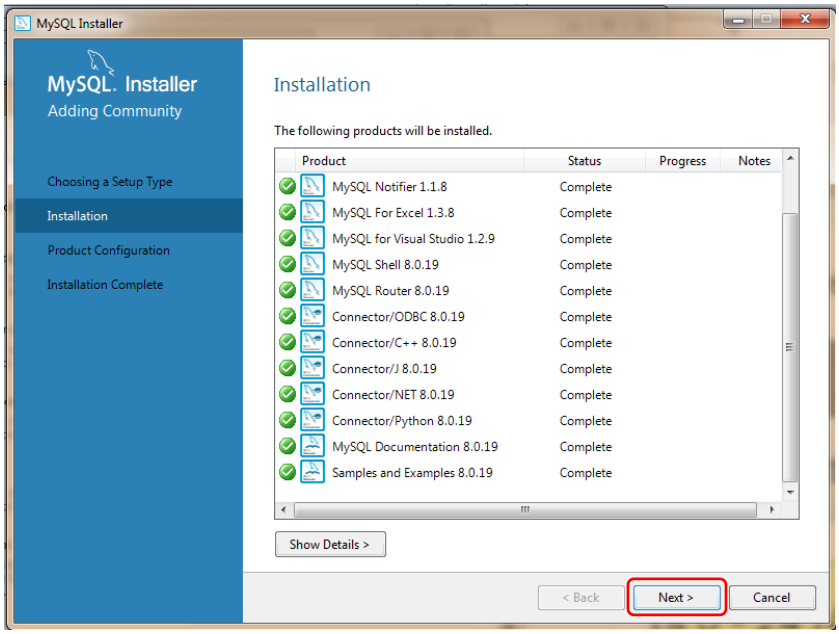
The **mysql-installer-community-8.0.19.0.msi** will be downloaded in your download directory. Run this file to continue.

5



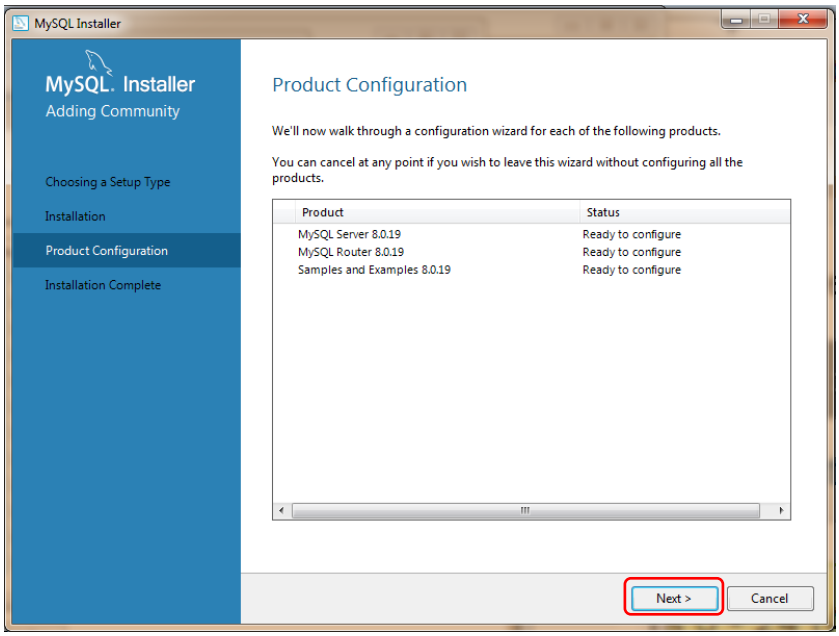
Select **„Full“** → **„Next >“**

6


MySQL Installer
MySQL. Installer
Adding Community
Choosing a Setup Type
Installation
Product Configuration
Installation Complete
Installation
The following products will be installed.
Product
Status
Progress
Notes
MySQL Notifier 1.1.8 Complete
MySQL For Excel 1.3.8 Complete
MySQL for Visual Studio 1.2.9 Complete
MySQL Shell 8.0.19 Complete
MySQL Router 8.0.19 Complete
Connector/ODBC 8.0.19 Complete
Connector/C++ 8.0.19 Complete
Connector/J 8.0.19 Complete
Connector/NET 8.0.19 Complete
Connector/Python 8.0.19 Complete
MySQL Documentation 8.0.19 Complete
Samples and Examples 8.0.19 Complete
Show Details >
< Back
Next >
Cancel

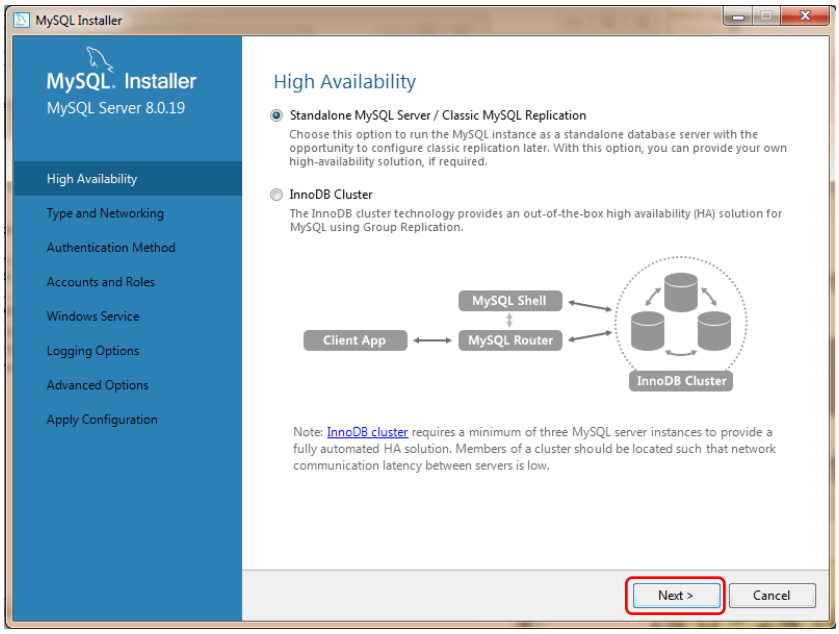
→ **„Next >“**

7


MySQL Installer
MySQL. Installer
Adding Community
Choosing a Setup Type
Installation
Product Configuration
Installation Complete
Product Configuration
We'll now walk through a configuration wizard for each of the following products.
You can cancel at any point if you wish to leave this wizard without configuring all the products.
Product
Status
MySQL Server 8.0.19 Ready to configure
MySQL Router 8.0.19 Ready to configure
Samples and Examples 8.0.19 Ready to configure
Next >
Cancel

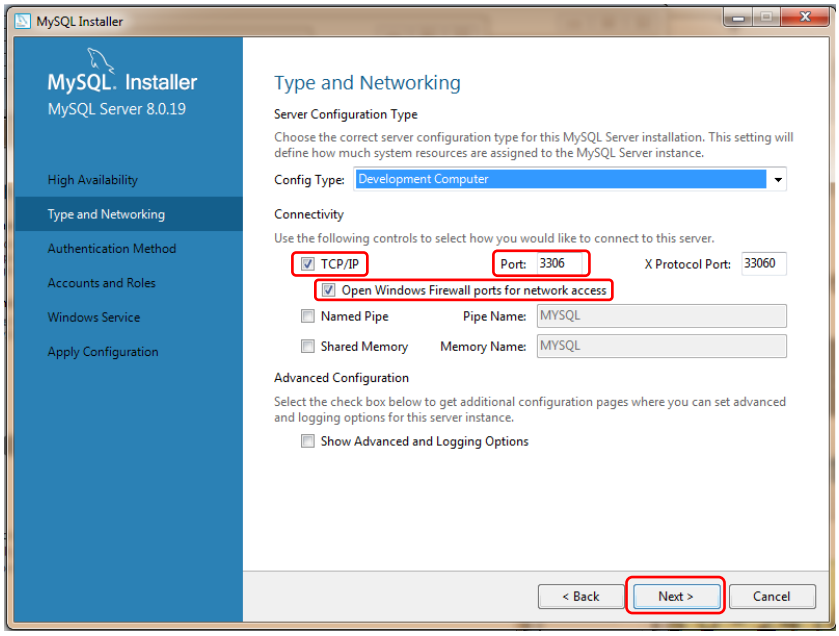
→ **„Next >“**

8



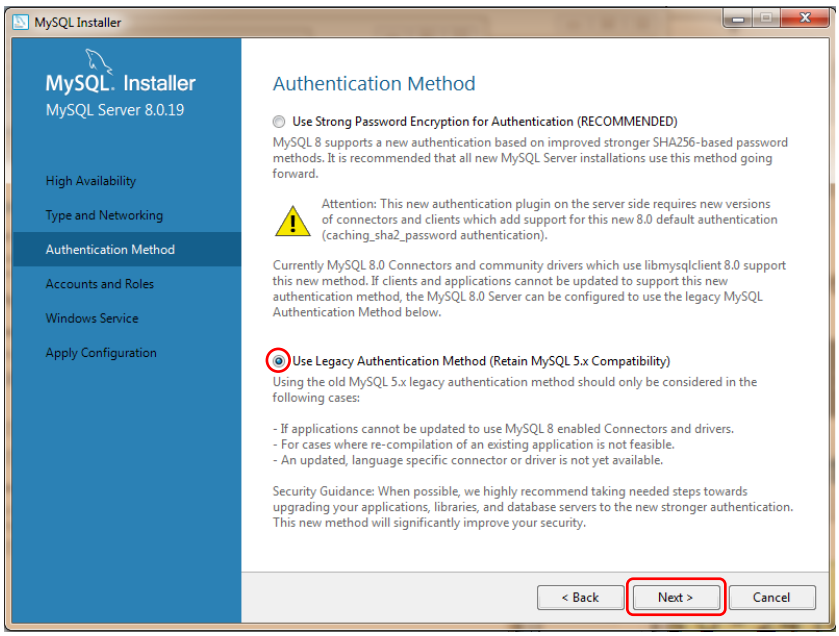
Select High Availability mode (e.g. **Standalone MySQL Server ...**) → **„Next >“**

9



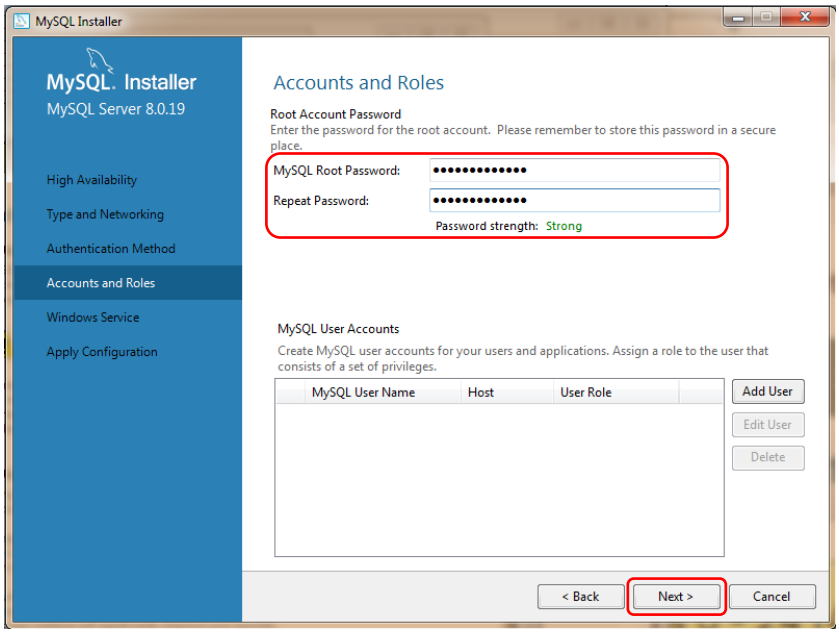
Check **TCP/IP**, select **Port number (default 3306)**, check **Open Windows Firewall ports for network access** → **„Next >“**

10



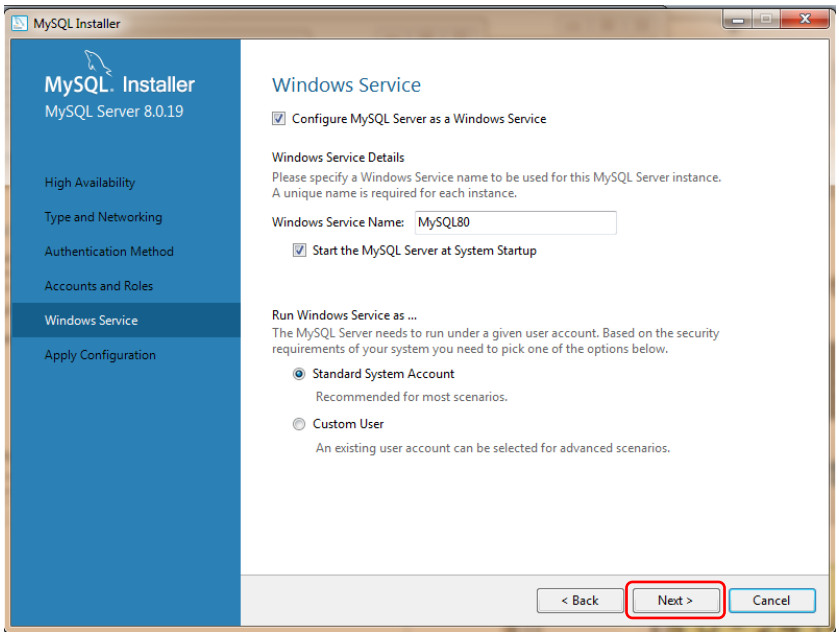
Select **Use Legacy Authentication Method** → **„Next >“**

11



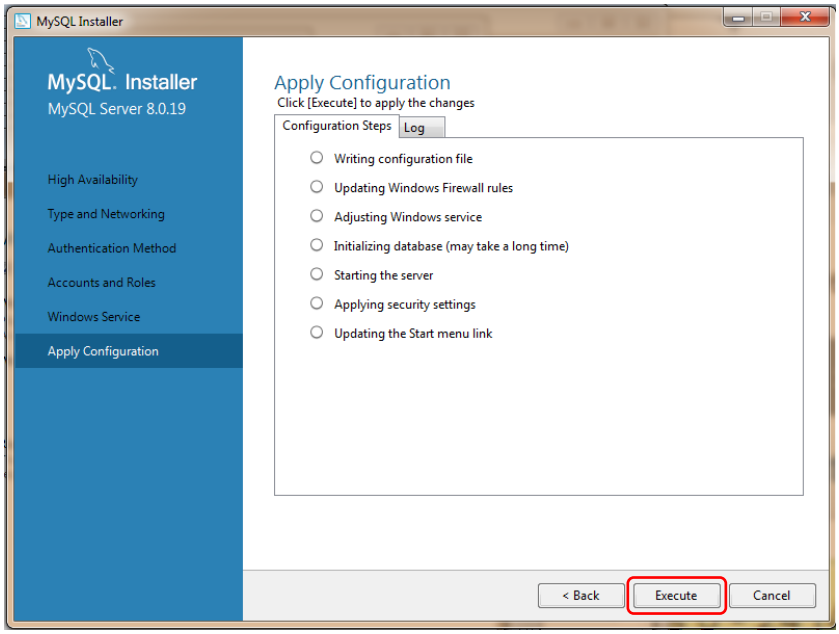
Define **MySQL Root Password** → **„Next >“**

12



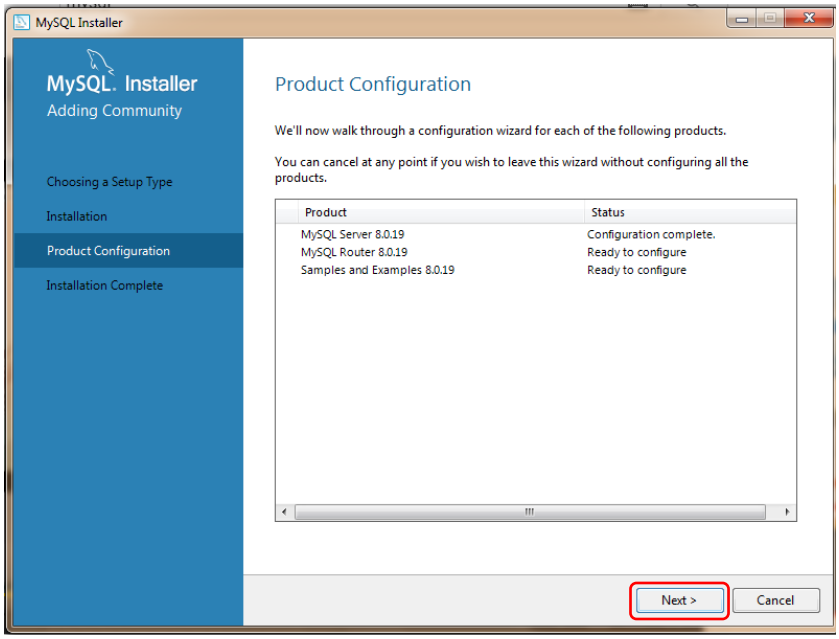
Select Windows service → **„Next >“**

13



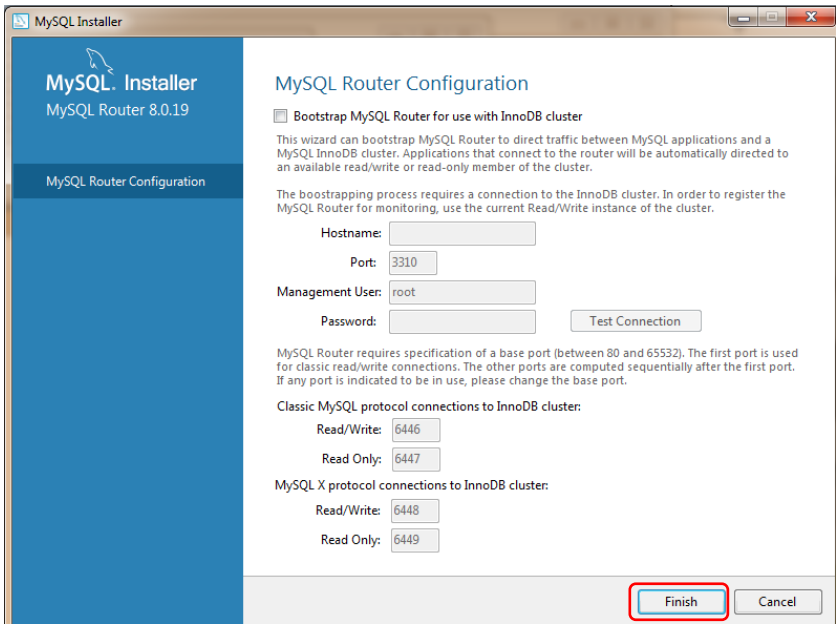
→ **„Execute“**

14



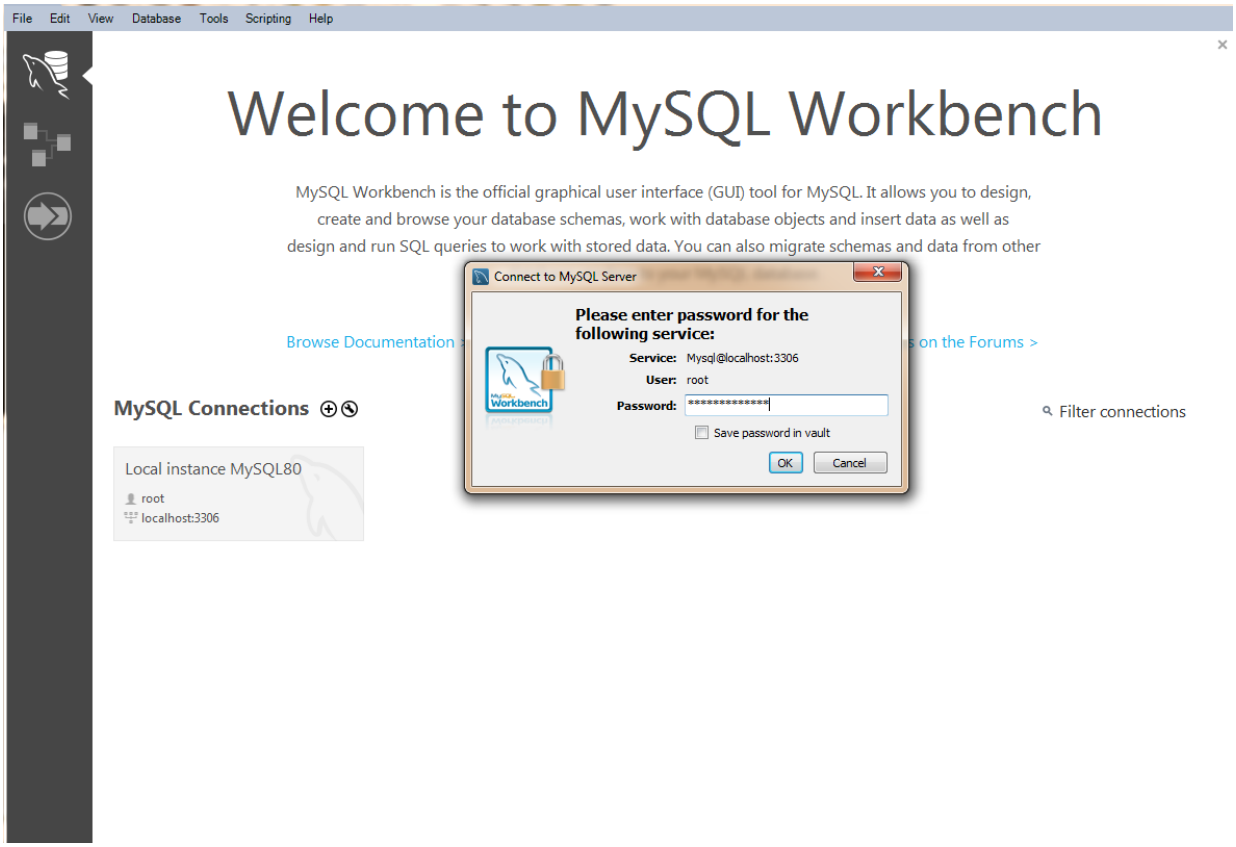
→ **„Next >“**

15



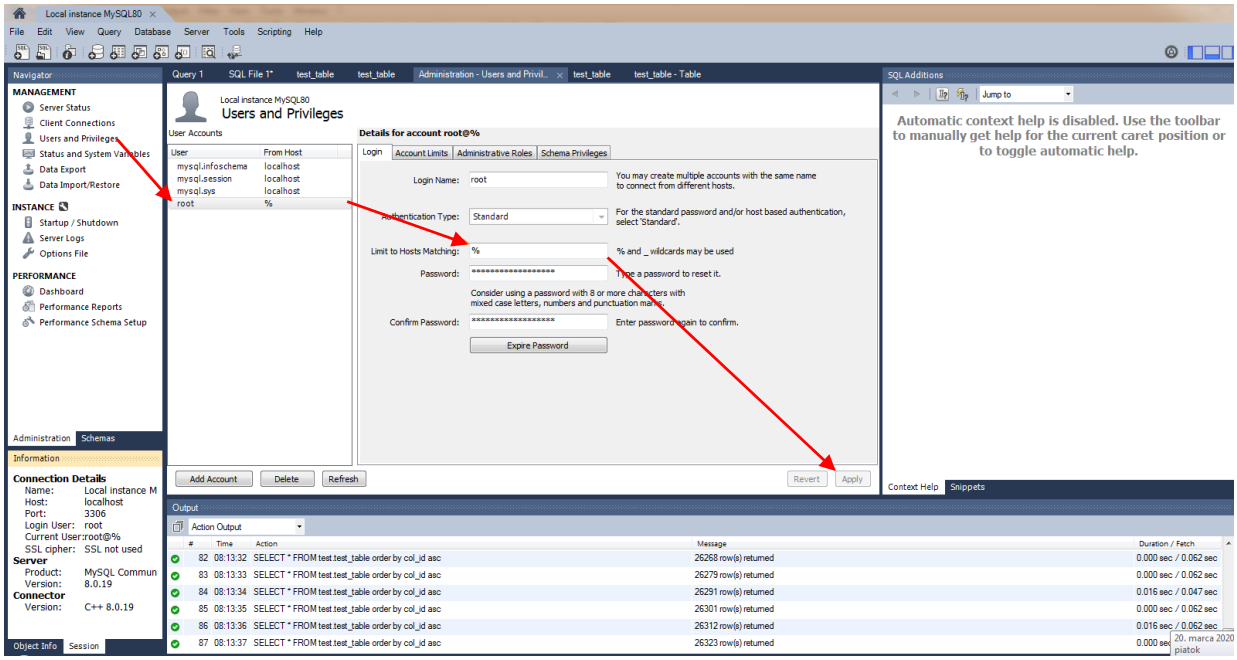
→ **„Finish“**

2.1.1 Host access settings for your MySQL account



You will need to run **MySQL Workbench** to be able to connect and manage the My**SQL Server**.

1. Select your **MySQL instance**
2. Enter **password** for **root** account
3. Connect to **MySQL server**



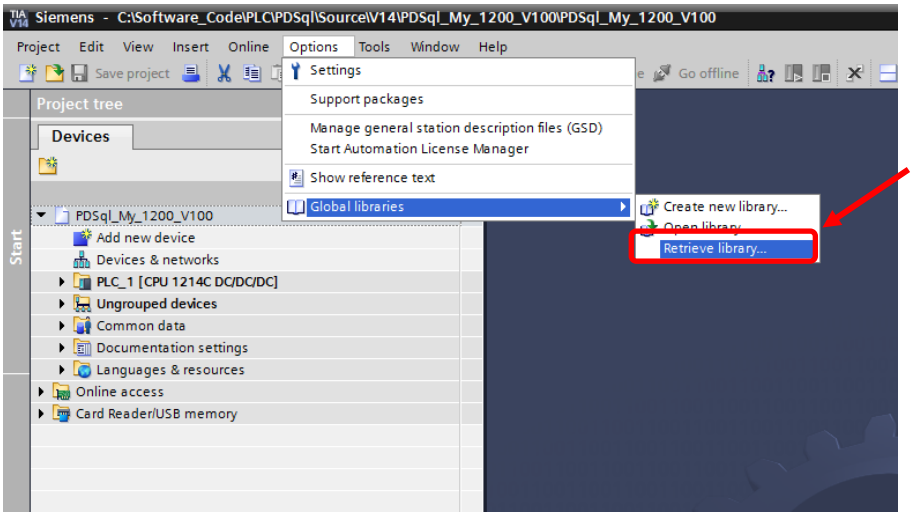
To **access** the **MySQL server** through a **MySQL account** from **any host** you must set up the following:

1. Click → **Users and Privileges**
2. Select → **User** (e.g. **root**)
3. Change → **Limit to Hosts Matching = „%“** (‚**%**‘ means access from any host)

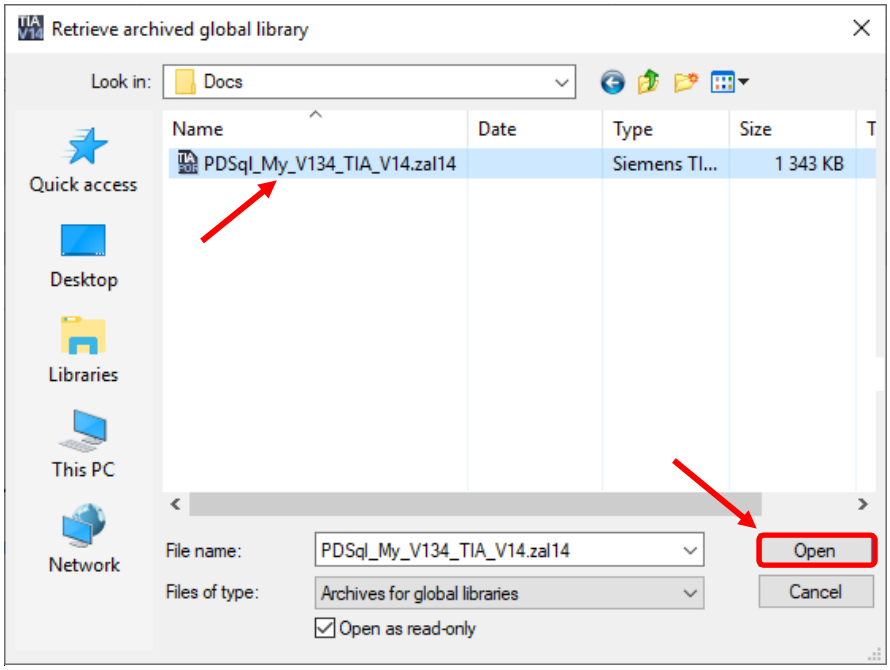
2.2 Instalation **PDSql Library My** to **TIA Portal**

- Retrive **PDSql Library My** from archive.

1

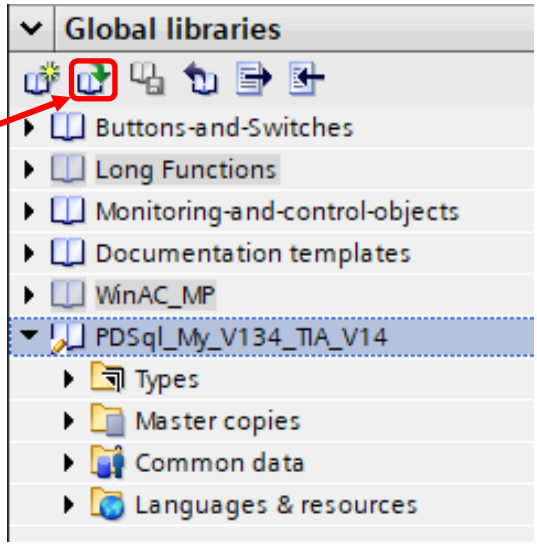


2



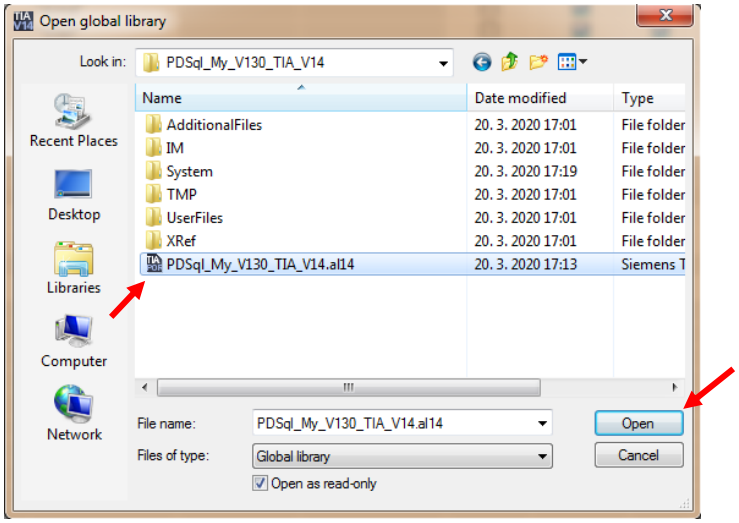
- Open **PDSql Library My** in **TIA Portal**.

3



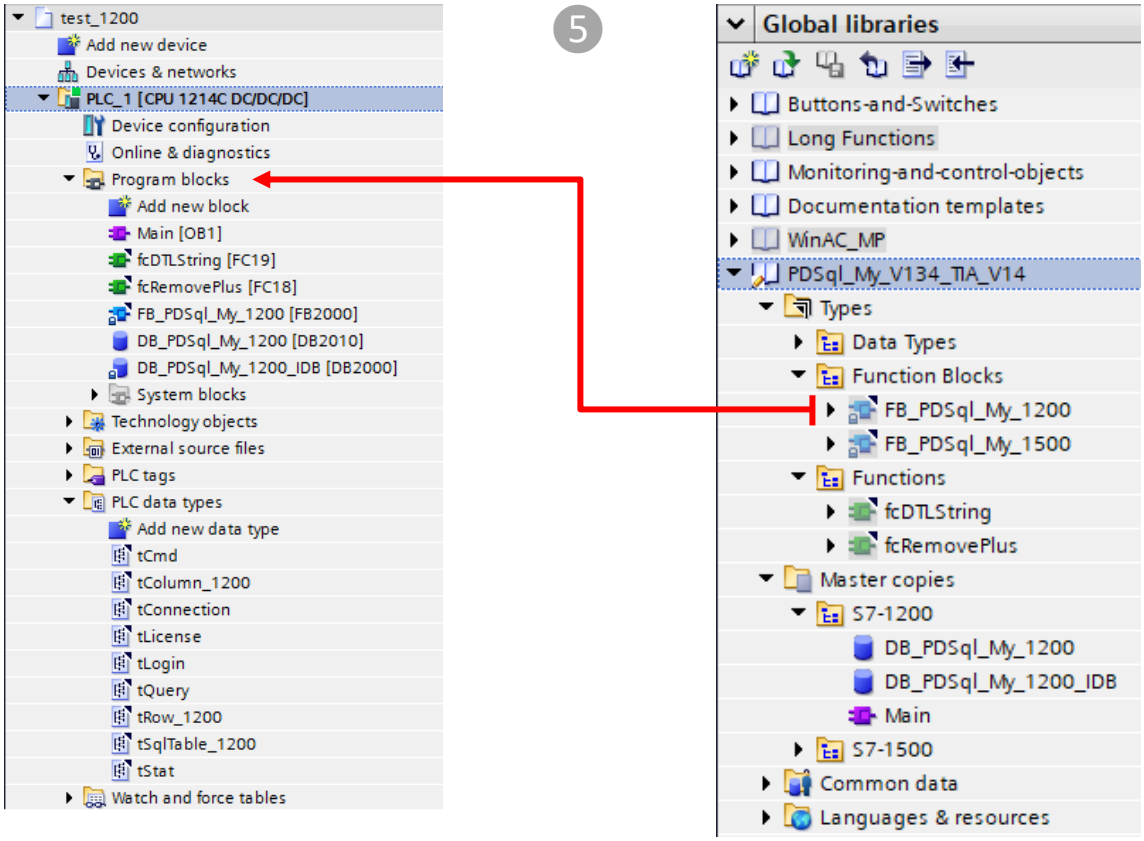
- Select **PDSql Library My** file and open.

4

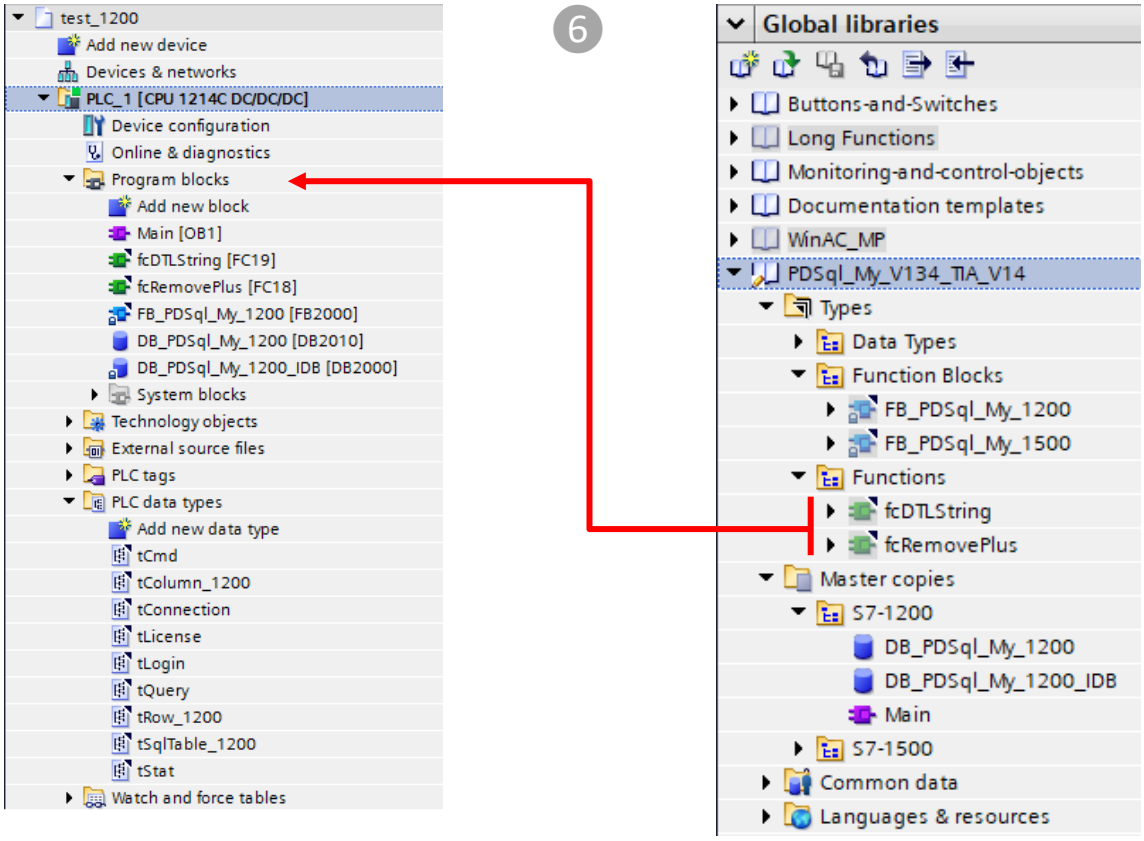


Example for S7-1200 project

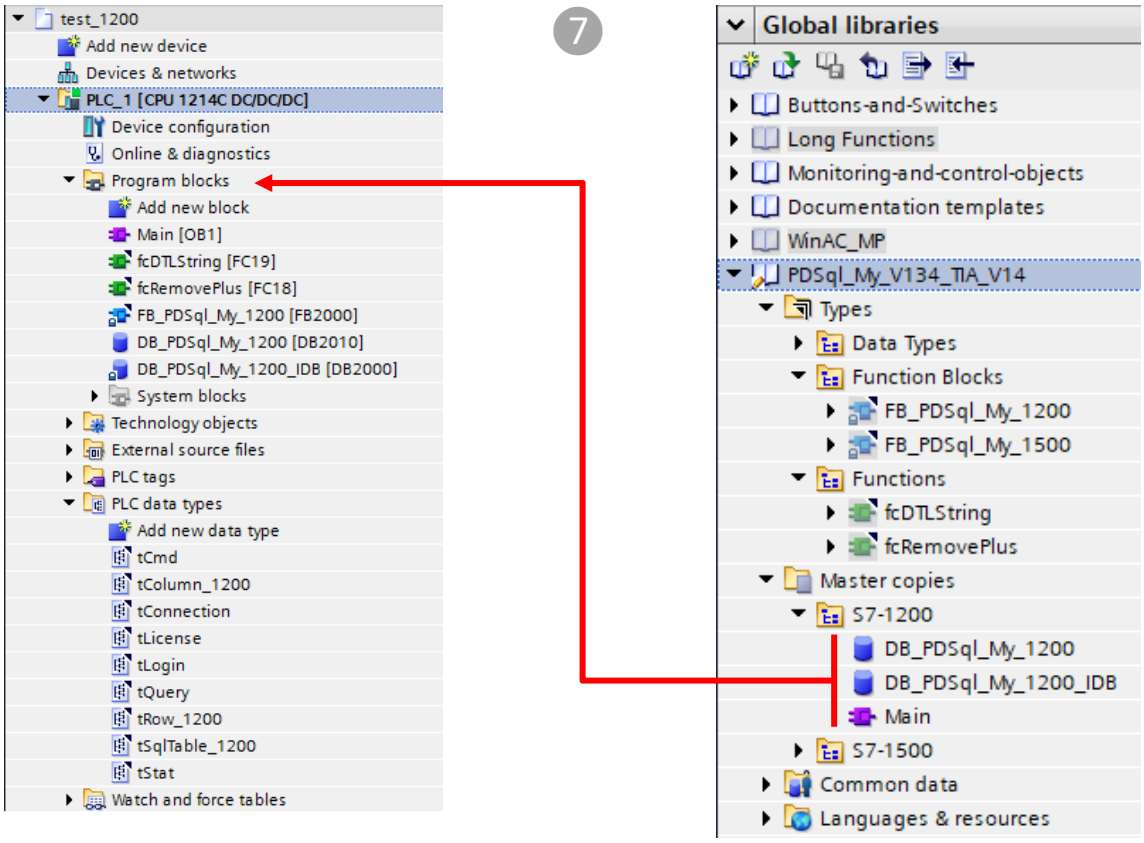
- Drag & Drop **Function Block** from **library** to your **Project / Program blocks**
  *This action will also **automatically** transfer all necessary **data types** from **library** to **Project / PLC data types***



- Drag & Drop **Functions** from **library** to your **Project / Program blocks**



- Drag & Drop **Master copies / S7-1200** data blocks from **library** to your **Project / Program blocks**



3 Examples

In the following examples we will use

- MySQL Workbench or phpMyAdmin
- **MySQL Server**
- **PDSql Library My**.

3.1 Create a table

Using MySQL Workbench or phpMyAdmin, as first we will create a schema **"plcdb"** and a table named "**test_table"** that will be used in the following examples and will contain these columns ->

Column name	Data type	Description
[Id]	INT	auto increment Id number
[Datetime]	DATETIME(4)	date and time
[FruitName]	VARCHAR(20)	fruit name
[Quantity]	INT(10)	fruit quantity
[Weight]	REAL	fruit weight
[CitrusType]	BIT(1)	citrus type [1 – Yes, 0 – No]

- **MySQL statement to create a schema**

```
CREATE SCHEMA plcdb
```

- **MySQL statement to create a table**

```
CREATE TABLE plcdb.test_table (
 `Id` INT NOT NULL AUTO_INCREMENT,
 `Datetime` DATETIME(4) NULL,
 `FruitName` VARCHAR(20) NULL,
 `Quantity` INT(10) NULL,
 `Weight` REAL NULL,
 `CitrusType` BIT(1) NULL,
 PRIMARY KEY (`Id`));
```

- **Insert test rows into the table**

```
INSERT INTO plcdb.test_table (Datetime,FruitName,Quantity,Weight,CitrusType)
VALUES ('2019-07-01 23:42:18','Apple',10,2.54,0),
       ('2019-07-01 23:48:54','Orange',16,3.28,1),
       ('2019-07-01 23:52:26','Lemon',8,1.64,1),
       ('2019-07-01 23:56:33','Mango',7,4.32,0)
```

Table: „**test_table**“

	Id	Datetime	FruitName	Quantity	Weight	CitrusType
▶	1	2019-07-01 23:42:18.0000	Apple	10	2.54	0
	2	2019-07-01 23:48:54.0000	Orange	16	3.28	1
	3	2019-07-01 23:52:26.0000	Lemon	8	1.64	1
	4	2019-07-01 23:56:33.0000	Mango	7	4.32	0

3.2 **SELECT** row from table

- **Read** a **row** from the **table** where **FruitName = "Apple"**

Syntax of the SQL statement in MySQL Workbench **or** phpMyAdmin

```
SELECT * FROM plcdb.test_table WHERE FruitName = 'Apple'
```

Syntax of the SQL statement in <u>TIA environment</u>

```
Query[1] := 'SELECT * FROM plcdb.test_table WHERE FruitName=$'Apple$'';
Query[2] := '';
```

⚠ In the **TIA environment** you must **add** a **dollar** (**$**) sign **before** each **single quote** in the **query string !**

Execute MySQL query with PDSql Library My

1. **Execute** MySQL query
   Cmd.ExecuteQuery := TRUE;
   Set Cmd.ExecuteQuery on TRUE will also automatically connect to MySQL server if not connected yet.

2. **Wait** for the execution of the MySQL query
   (Stat.ExecutedOK OR Stat.Error) = TRUE;

3. If **ExecutedOK** then result is stored in the **SQLtable**

	Id	Datetime	FruitName	Quantity	Weight	CitrusType
▶	1	2019-07-01 23:42:18.0000	Apple	10	2.54	0

Connected	TRUE	TRUE
Busy	TRUE	FALSE
ExecutedOK	FALSE	TRUE
Error	FALSE	FALSE
Status	16#0000	16#0000
AffectedRows	10	1
Message	'10 row(s) affec...	'1 row(s) affected'

The status after SELECT command

SQLTable		
ColumnCount	0	6
RowCount	0	1
MaxColumns	10	10
MaxRows	10	10

The number of rows and columns returned after SELECT command

Columns[1]			
Name	''	''	'Id'
CharacterSet	0	0	63
ColumnLen...	0	0	11
Type	16#00	...	16#03
Flags	16#0000	...	16#4203
Decimals	16#00	...	16#00
Rows			
Rows[1]			
Bool	FALSE	...	FALSE
Null	FALSE	...	FALSE
DInt	0	0	1
LReal	0.0	...	0.0
String	''	''	'1'
Date...	DTL#19...	...	DTL#1970-01-01-00:00:00

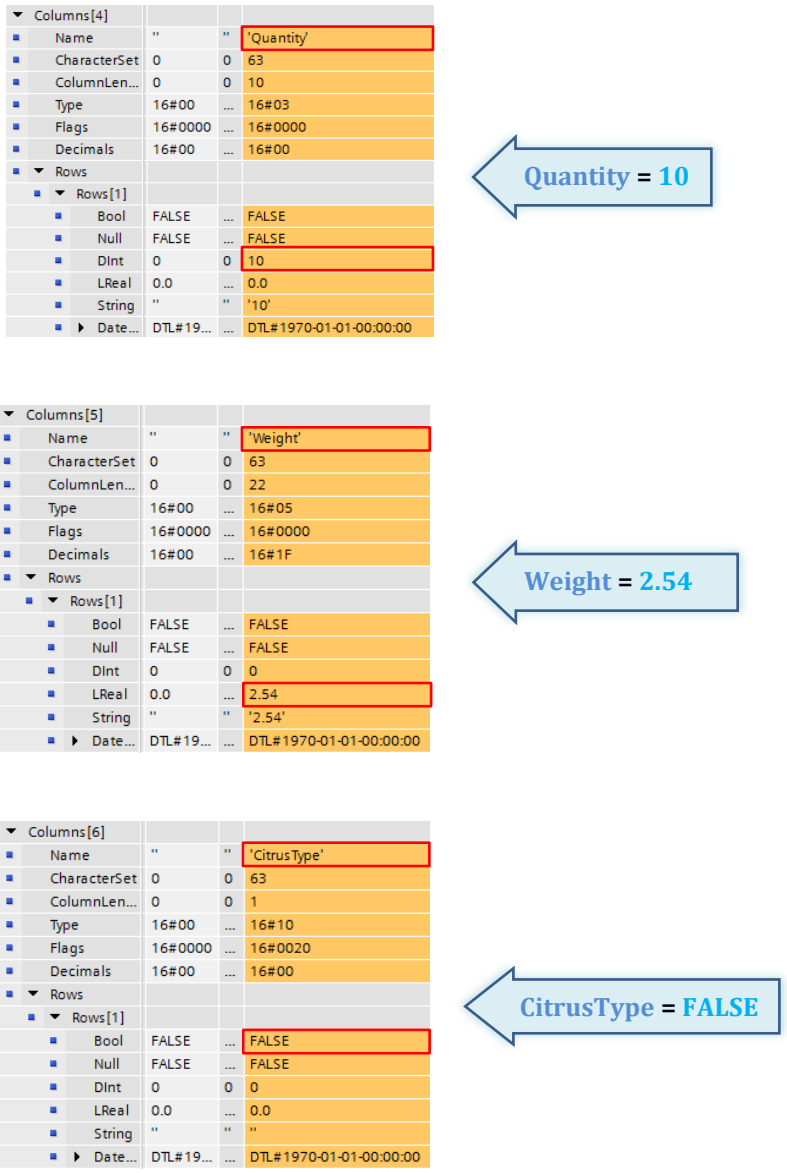
Id = 1

Columns[2]			
Name	''	''	'Datetime'
CharacterSet	0	0	63
ColumnLen...	0	0	24
Type	16#00	...	16#0C
Flags	16#0000	...	16#0080
Decimals	16#00	...	16#04
Rows			
Rows[1]			
Bool	FALSE	...	FALSE
Null	FALSE	...	FALSE
DInt	0	0	0
LReal	0.0	...	0.0
String	''	''	'2019-07-01 23:42:18.0000'
Date...	DTL#19...	...	DTL#2019-07-01-23:42:18

Datetime = "2019-07-01-23:42:18"

Columns[3]			
Name	''	''	'FruitName'
CharacterSet	0	0	8
ColumnLen...	0	0	20
Type	16#00	...	16#FD
Flags	16#0000	...	16#0000
Decimals	16#00	...	16#00
Rows			
Rows[1]			
Bool	FALSE	...	FALSE
Null	FALSE	...	FALSE
DInt	0	0	0
LReal	0.0	...	0.0
String	''	''	'Apple'
Date...	DTL#19...	...	DTL#1970-01-01-00:00:00

FruitName = "Apple"



4. If **Error** then Stat.Status contains an **error code** and Stat.Message contains an **error message**

3.3 **INSERT** row to the table

➢ **Write** a **new row** into the **table**

Syntax of the MySQL statement in MySQL Workbench **or** phpMyAdmin

```
INSERT INTO plcdb.test_table (Datetime,FruitName,Quantity,Weight,CitrusType) VALUES
('2019-07-02 10:42:56','Banana',24,8.48,0)
```

Syntax of the MySQL statement in <u>TIA environment</u>

```
Query[1] := 'INSERT INTO plcdb.test_table (Datetime,FruitName,Quantity,Weight,CitrusType)
VALUES ($'2019-07-02 10:42:56$',$'Banana$',24,8.48,0)';
Query[2] := '';
```

⚠ In the **TIA environment** you must **add** a **dollar** (**$**) sign **before** each **single quote** in the **query string !**

Compose INSERT string from input variables (SCL language)

```
#dtlDateTime := '2019-07-02 10:42:56';
#sFruitName := 'Banana';
#iQuantity := 24;
#rWeight := 8.48;
#bCitrusType := FALSE;

#str := 'INSERT INTO plcdb.test_table (Datetime,FruitName,Quantity,Weight,CitrusType) VALUES($'';
#str := CONCAT(IN1 := #str, IN2 := "fcDTLString"(#dtlDateTime));
#str := CONCAT(IN1 := #str, IN2 := '$',$'');
#str := CONCAT(IN1 := #str, IN2 := #sFruitName);
#str := CONCAT(IN1 := #str, IN2 := '$',');
#str := CONCAT(IN1 := #str, IN2 := INT_TO_STRING(#iQuantity));
#str := CONCAT(IN1 := #str, IN2 := ',');
#str := CONCAT(IN1 := #str, IN2 := REAL_TO_STRING(#rWeight));
#str := CONCAT(IN1 := #str, IN2 := ',');
#str := CONCAT(IN1 := #str, IN2 := INT_TO_STRING(BOOL_TO_BYTE(#bCitrusType)));
#str := CONCAT(IN1 := #str, IN2 := ')');

"DB_PDSql_1200".Query.Query[1] := #str;
"DB_PDSql_1200".Query.Query[2] := '';
```

Execute MySQL query with PDSql Library My

1. **Execute** MySQL query
   Cmd.ExecuteQuery := TRUE;
   Set Cmd.ExecuteQuery on TRUE will also automatically connect to MySQL server if not connected yet.

2. **Wait** for the execution of the MySQL query
   (Stat.ExecutedOK OR Stat.Error) = TRUE;

3. If **ExecutedOK** then a row was inserted successfully

Connected	TRUE	TRUE
Busy	TRUE	FALSE
ExecutedOK	FALSE	TRUE
Error	FALSE	FALSE
Status	16#0000	16#0000
AffectedRows	10	1
Message	'10 row(s) affec...	'1 row(s) affected'

"**test_table**" after executing the **INSERT** command

Id	Datetime	FruitName	Quantity	Weight	CitrusType
1	2019-07-01 23:42:18.0000	Apple	10	2.54	0
2	2019-07-01 23:48:54.0000	Orange	16	3.28	1
3	2019-07-01 23:52:26.0000	Lemon	8	1.64	1
4	2019-07-01 23:56:33.0000	Mango	7	4.32	0
5	2019-07-02 10:42:56.0000	Banana	24	8.48	0

4. If **Error** then Stat.Status contains an **error code** and
   Stat.Message contains an **error message**

3.4 **UPDATE** row in table

➢ **Update** an existing **row** in the **table** where **FruitName = "Mango"**

⚠ In this example we will **update** a **record without specifying** a **key** in the **where clause**, so we need to set **SQL_SAFE_UPDATES = 0**

Syntax of the MySQL statement in MySQL Workbench **or** phpMyAdmin

```
SET SQL_SAFE_UPDATES = 0;

UPDATE plcdb.test_table SET Quantity = 32, Weight = 12.68 WHERE FruitName = 'Mango'
```

Syntax of the MySQL statement in TIA environment

```
Query[1] := UPDATE plcdb.test_table SET Quantity=32, Weight=12.68 WHERE FruitName =
$'Mango$'';
Query[2] := '';
```

⚠ In the **TIA environment** you must **add** a **dollar ($)** sign **before** each **single quote** in the **query string !**

Compose UPDATE string from input variables (SCL language)

```
#sFruitName := 'Mango';
#iQuantity := 32;
#rWeight := 12.68;

#str := 'UPDATE plcdb.test_table SET Quantity=';
#str := CONCAT(IN1 := #str, IN2 := INT_TO_STRING(#iQuantity));
#str := CONCAT(IN1 := #str, IN2 := ',Weight=');
#str := CONCAT(IN1 := #str, IN2 := REAL_TO_STRING(#rWeight));
#str := CONCAT(IN1 := #str, IN2 := ' WHERE FruitName=$'');
#str := CONCAT(IN1 := #str, IN2 := #sFruitName);
#str := CONCAT(IN1 := #str, IN2 := '$'');

"DB_PDSql_1200".Query.Query[1] := #str;
"DB_PDSql_1200".Query.Query[2] := '';
```

Execute MySQL query with PDSql Library My

1. **Execute** MySQL query
   Cmd.ExecuteQuery := TRUE;
   Set Cmd.ExecuteQuery on TRUE will also automatically connect to MySQL server if not connected yet.

2. **Wait** for the execution of the MySQL query
   (Stat.ExecutedOK OR Stat.Error) = TRUE;

3. If **ExecutedOK** then a row was updated successfully

Connected	TRUE	TRUE
Busy	TRUE	FALSE
ExecutedOK	FALSE	TRUE
Error	FALSE	FALSE
Status	16#0000	16#0000
AffectedRows	10	1
Message	'10 row(s) affec...	'1 row(s) affected'

"**test_table**" after executing the **UPDATE** command

Id	Datetime	FruitName	Quantity	Weight	CitrusType
1	2019-07-01 23:42:18.0000	Apple	10	2.54	0
2	2019-07-01 23:48:54.0000	Orange	16	3.28	1
3	2019-07-01 23:52:26.0000	Lemon	8	1.64	1
4	2019-07-01 23:56:33.0000	Mango	32	12.68	0
5	2019-07-02 10:42:56.0000	Banana	24	8.48	0

4. If **Error** then Stat.Status contains an **error code** and
   Stat.Message contains an **error message**

3.5 **DELETE** row from table

➢ **Delete** an existing **row** from the **table** where **FruitName = "Lemon"**

⚠ In this example we will **delete** a **record without specifying** a **key** in the **where clause**, so we need to set **SQL_SAFE_UPDATES = 0**

Syntax of the MySQL statement in MySQL Workbench **or** phpMyAdmin

```
SET SQL_SAFE_UPDATES = 0

DELETE FROM plcdb.test_table WHERE FruitName = 'Lemon'
```

Syntax of the MySQL statement in <u>TIA environment</u>

```
Query[1] := 'DELETE FROM plcdb.test_table WHERE FruitName=$'Lemon$'';
Query[2] := '';
```

⚠ In the **TIA environment** you must **add** a **dollar** (**$**) sign **before** each **single quote** in the **query string !**

Compose DELETE string from input variables (SCL language)

```
#sFruitName := 'Lemon';

#str := 'DELETE FROM plcdb.test_table WHERE FruitName=$'';
#str := CONCAT(IN1 := #str, IN2 := #sFruitName);
#str := CONCAT(IN1 := #str, IN2 := '$'');

"DB_PDSql_1200".Query.Query[1] := #str;
"DB_PDSql_1200".Query.Query[2] := '';
```

Execute MySQL query with PDSql Library My

1. **Execute** MySQL query

   Cmd.ExecuteQuery := TRUE;

   Set Cmd.ExecuteQuery on TRUE will also automatically connect to MySQL server if not connected yet.

2. **Wait** for the execution of the MySQL query
   (Stat.ExecutedOK OR Stat.Error) = TRUE;

3. If **ExecutedOK** then the existing row was deleted successfully

Connected	TRUE	TRUE
Busy	TRUE	FALSE
ExecutedOK	FALSE	TRUE
Error	FALSE	FALSE
Status	16#0000	16#0000
AffectedRows	10	1
Message	'10 row(s) affec...	'1 row(s) affected'

"**TestTable**" after executing the **DELETE** command

Id	Datetime	FruitName	Quantity	Weight	CitrusType
1	2019-07-01 23:42:18.0000	Apple	10	2.54	0
2	2019-07-01 23:48:54.0000	Orange	16	3.28	1
4	2019-07-01 23:56:33.0000	Mango	32	12.68	0
5	2019-07-02 10:42:56.0000	Banana	24	8.48	0

4. If **Error** then Stat.Status contains an **error code** and
   Stat.Message contains an **error message**