Lab 2: Create and work with a table

In this lab, you will use the MySQL Workbench to create a *schema* and several *tables*. You'll work with many of the column attributes that you can assign to columns in a table. Finally, you'll enter some test data into your tables so that you can see the results of your work.

> **NOTE:** this lab provides references to the MySQL Documentation. You can find the documentation at the following URL:
>
> https://dev.mysql.com/doc/refman/5.7/en/
>
> You can view the documentation in your web browser. You can also download a PDF that contains all of the online documentation.

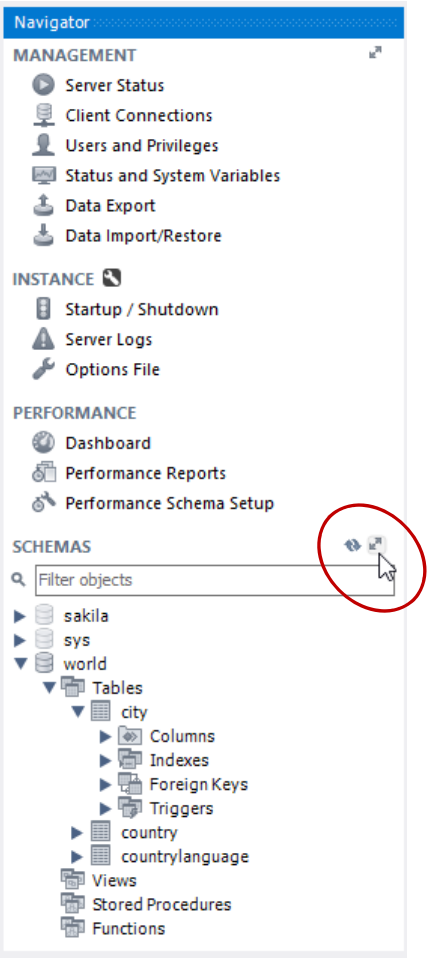
2.1 Start the MySQL Workbench

_____ Start the MySQL Workbench (referred to as "Workbench" from this point forward) if it is not already open.

_____ Locate the **Navigator** panel in the Workbench. By default, it is on the left side.

_____ If the Navigator shows the sections that you see on the left side of Figure 1, click the **Expand / Shrink** icon that is highlighted in the figure.

_____ You should now see only the **SCHEMAS** section in the Navigator, as shown on the right side of the figure.



L02_0010

L02_0011

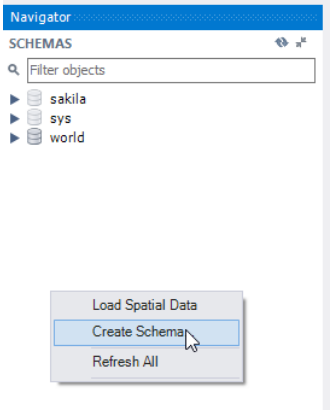
Figure 1: Use the expand/shrink icon to shrink the Navigator to the SCHEMAS section.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

2.2 Create a new schema

_____ Move the mouse cursor so that it is in an open (blank) part of the **SCHEMAS** section.

_____ Right-click, the click the **Create Schema** item in the pop-up menu, as shown in Figure 2.



L02_0021

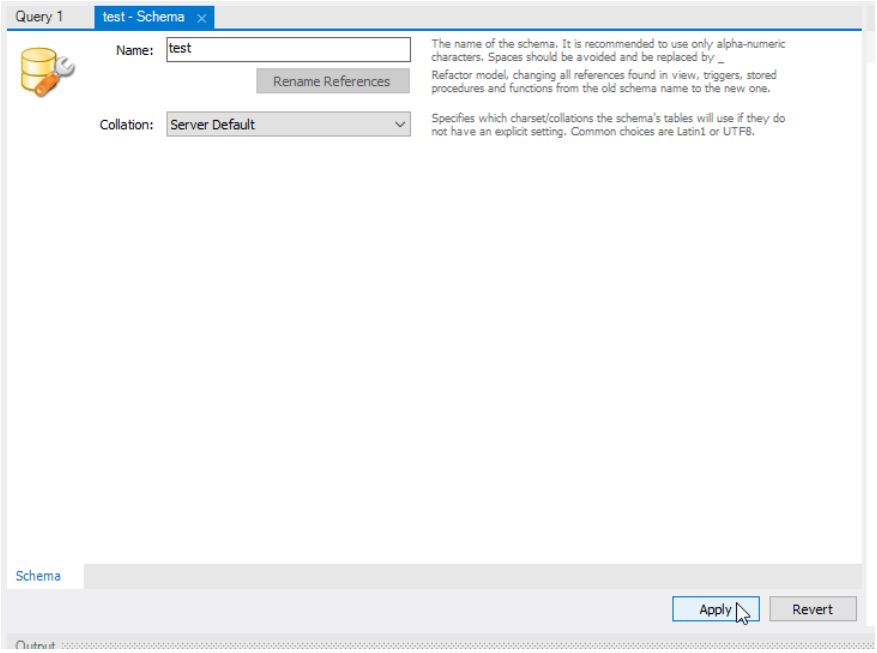
Figure 2: Right-click in an open part of the SCHEMAS section and click the Create Schema item.

2.2.1 Use the new schema tab to create the schema

_____ The new schema tab is displayed, as shown in Figure 3.

_____ For the **Name** value, enter `test`.

_____ Click the **Apply** button.



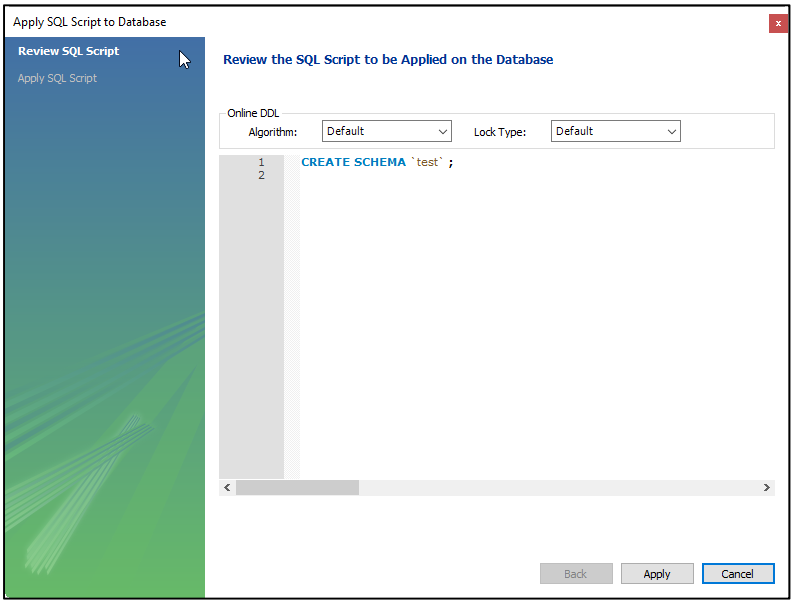
L02_0022

Figure 3: Enter a name for the schema and click the Apply button.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

_____ The **Apply SQL Script to Database** panel shown in Figure 4 is displayed. It shows the DDL (Data Definition Language) statement that will be applied.

_____ Click the **Apply** button.

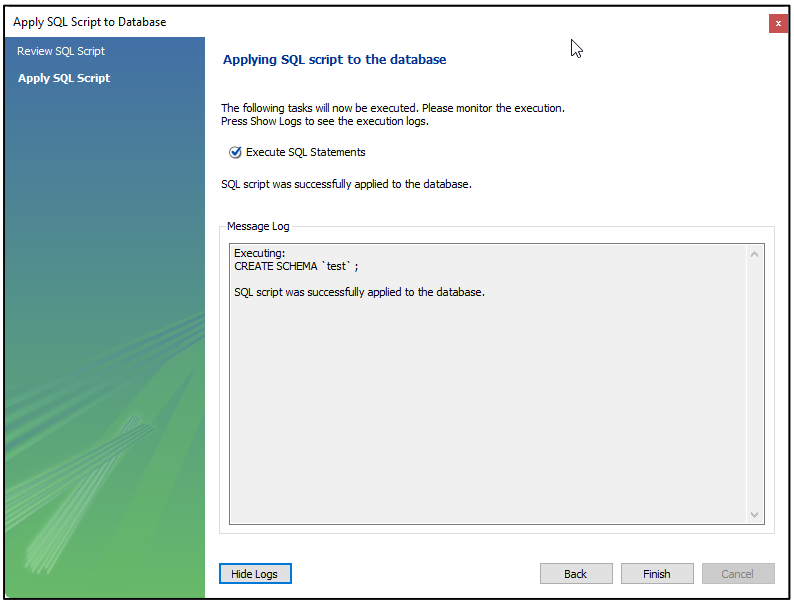


L02_0025

Figure 4: Click the Apply button.

_____ The next panel is shown (Figure 5). You can click the **Show Logs** button to see the log of the statement that was run.

_____ Click the **Finish** button. [**MAC**: click the **Close** button.]

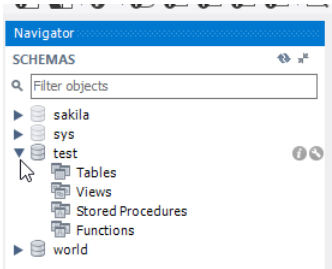


L02_0026

Figure 5: Click the Finish button.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

_____ You should now see the `test` schema in the **SCHEMAS** section of the Navigator, as shown in Figure 6.



L02_0028

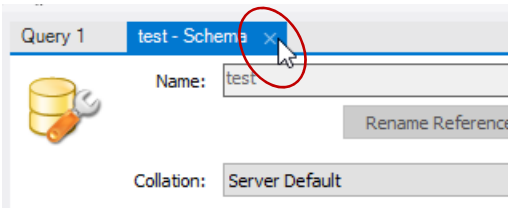
Figure 6: You should now see the test schema in the SCHEMAS section of the Navigator.

2.2.2 Close the new schema tab

When you work with a tool like Workbench, you'll probably find it is easier to work with it if you "clean up" when you are done with a task. Now that you've created the `test` schema, you'll close the new schema tab, since you won't need to use it again in this lab.

_____ Locate the tab for the new schema panel.

_____ Click the **X** on the tab to close it, as shown in Figure 7.



L02_0031

Figure 7: Click the "X" to close the tab.

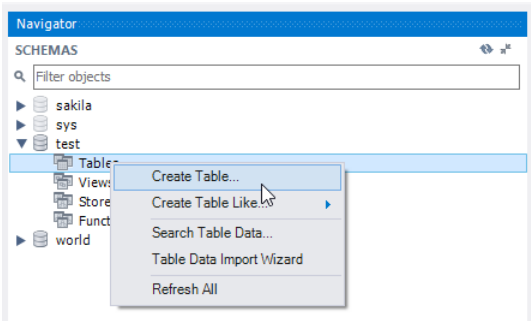
2.3 Create a new table to work with CHAR and VARCHAR datatypes

Now that you have the `test` schema, you will create a table in the schema.

_____ Expand the `test` schema.

_____ Right-click the **Tables** item.

_____ Click the **Create Table** item in the pop-up menu, as shown in Figure 8.



L02_0041

Figure 8: Right-click the Tables item and select the Create Table item in the pop-up menu.

_____ The new table panel is displayed.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

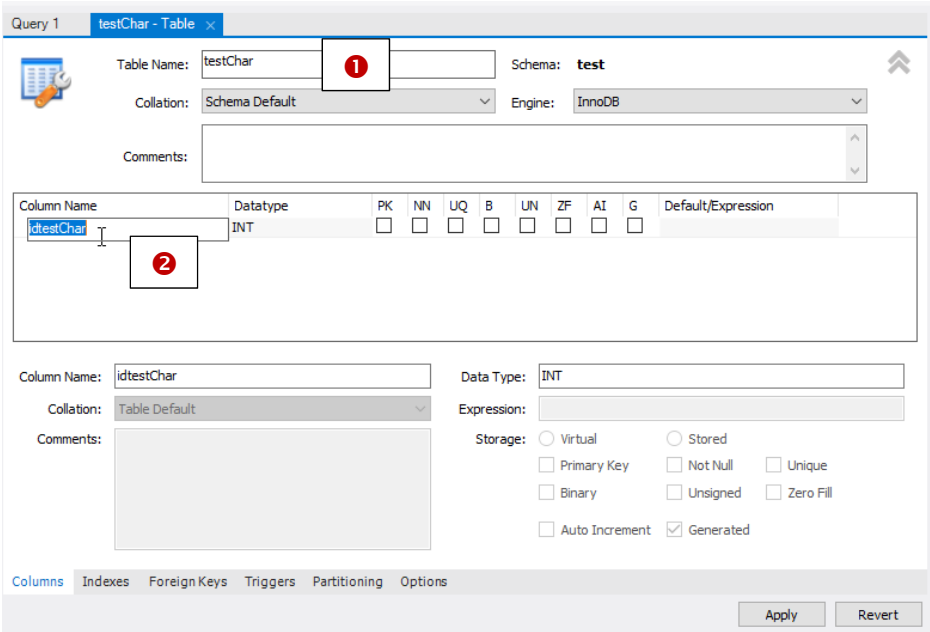
2.3.1 Work with the Primary Key column

_____ Do the following, using the item numbers shown in Figure 9 as a guide:

____ Item 1: enter the value `testChar` for the **Table Name**

____ Item 2: double-click directly under the **Column Name** heading

_____ When you double-click under the heading, Workbench proposes the column name `idtestChar` with a proposed datatype of `INT` (integer).

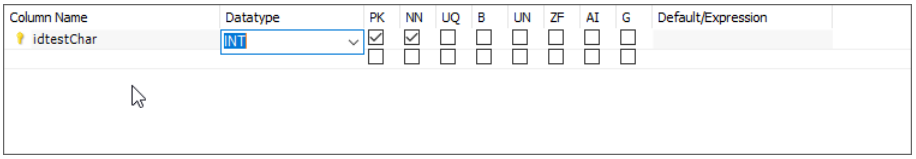


L02_0042

Figure 9: Set the table name to testChar and double-click in the Column Name column.

_____ Without moving the mouse, press the **Tab** key to tab out of the proposed column name field.

_____ As soon as you press **Tab**, Workbench checks the **PK** and **NN** checkboxes, as shown in Figure 10.



L02_0043

Figure 10: When you tab out of the column name field, Workbench assigns the PK and NN attributes to the column.

Here is what Workbench is doing: when you create a table, it is almost always a good idea to define a column as the *primary key* column. The primary key can be used to uniquely identify a particular row in the table, and it is used when you create relations to other tables. Although the primary key is not required to be the first column in the table, it is commonly defined as the first column.

By checking the **PK** checkbox, Workbench is assigning the primary key *attribute* to the `idtestChar` column.

Workbench also checks the **NN** checkbox, to indicate that the **Not Null** attribute is to be assigned to the column. A column that is a primary key cannot contain a null value, since `NULL` cannot be used to identify a row.

> **NOTE:** primary key usage is covered in more detail in the video for Session 2, Part 4. `NULL` and Not Null are covered in detail in the video for Session 2, Part 3.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

_____ The proposed datatype for the primary key column is set to **INT** (for Integer). The reason why this is a proposed datatype is because a primary key is usually, but is not required to be, an integer value.

The range for an integer is

```
-2,147,483,648 to 2,147,473,647
```

Because primary key values are usually only positive values (when the datatype for the primary key is numeric), you can have over 2 billion unique primary key values in the table. If you think that the table might contain more than 2 billion rows, you can choose **BIGINT** (Big Integer) as the datatype. The range for a big integer value is

```
-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
```

Another alternative is to assign the **Unsigned** attribute to the column. An *unsigned integer* is an integer that can only contain positive values (including zero). The range of an unsigned integer is

```
0 to 4,294,967,295
```

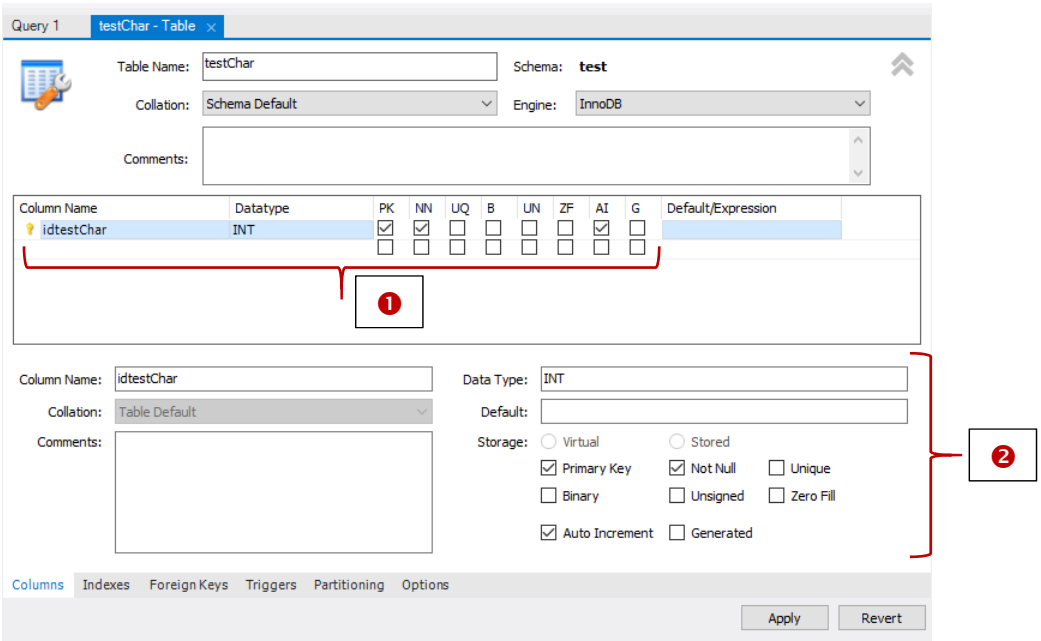
(over 4 billion unique positive values)

To indicate that a numeric value is to be unsigned, you can check the **UN** checkbox as shown in Figure 10.

_____ For this table (and all of the other tables that you will create in this course), you will define primary key columns as `INT` and **Not Null**.

_____ Because you will be using a numeric primary key, you can now check the **AI** checkbox to assign the `AUTO INCREMENT` attribute to the column.

_____ You should now see that the table definition panel looks like Figure 11. The attributes are shown in "condensed form" at Item 1. At Item 2, the attributes are spelled out. When you check or uncheck a checkbox in either section, the other section is affected.



L02_0044

Figure 11: This is what the table definition panel looks like when you are done defining the first column.

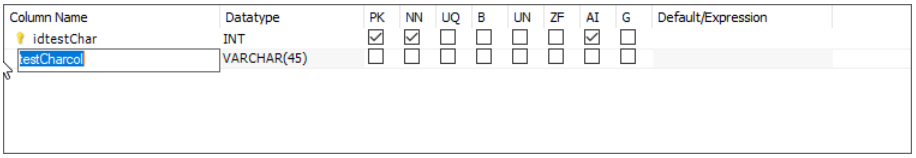
Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

2.3.2 Add more columns to the table

_____ Position the mouse cursor under the `idtestChar` column name.

_____ Double-click the mouse.

_____ Workbench create a new column named `testCharcol` and proposes a datatype of `VARCHAR(45)` as shown in Figure 12.



L02_0051

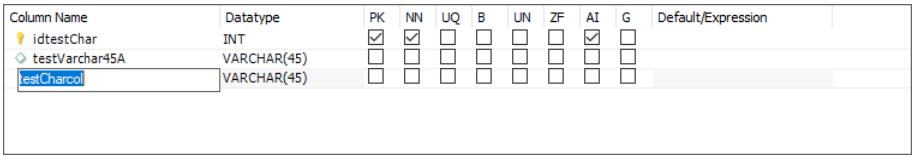
Figure 12: This shows what Workbench looks like when you add another column.

_____ The column name `testCharcol` is highlighted. Type the following to change the column name:

```
testVarchar45A
```

_____ After changing the column name, press the **Tab** key.

_____ Double-click under the `testVarchar45A` column name. You should see that Workbench adds another column (with the name `testCharcol`) as shown in Figure 13.



L02_0052

Figure 13: This is what Workbench looks like after you enter the first column VARCHAR column and start entering the second VARCHAR column.

_____ Change the newly added column name to

```
testVarchar45B
```

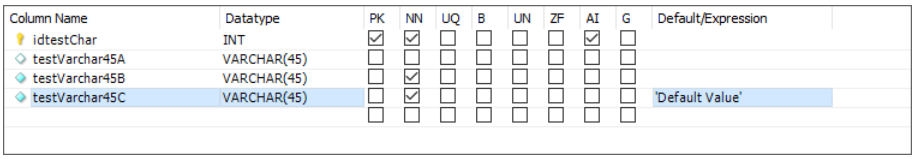
_____ Check the **NN** checkbox for the `testVarchar45B` column. This sets the **Not Null** attribute for the column.

_____ Double-click to start another column and do the following:

- ____ Set the next column name to `testVarchar45C`
- ____ Check the **NN** checkbox
- ____ Enter the text `Default Value` in the **Default/Expression** column

Note: if you enter a default for a character (or varchar) column and you do not enclose the value in single quote characters, Workbench adds the single quotes when you tab out of the entry field.

_____ You should now see that Workbench looks like Figure 14. Make sure that everything is the same as in the figure, especially in the **NN** checkboxes.



L02_0053

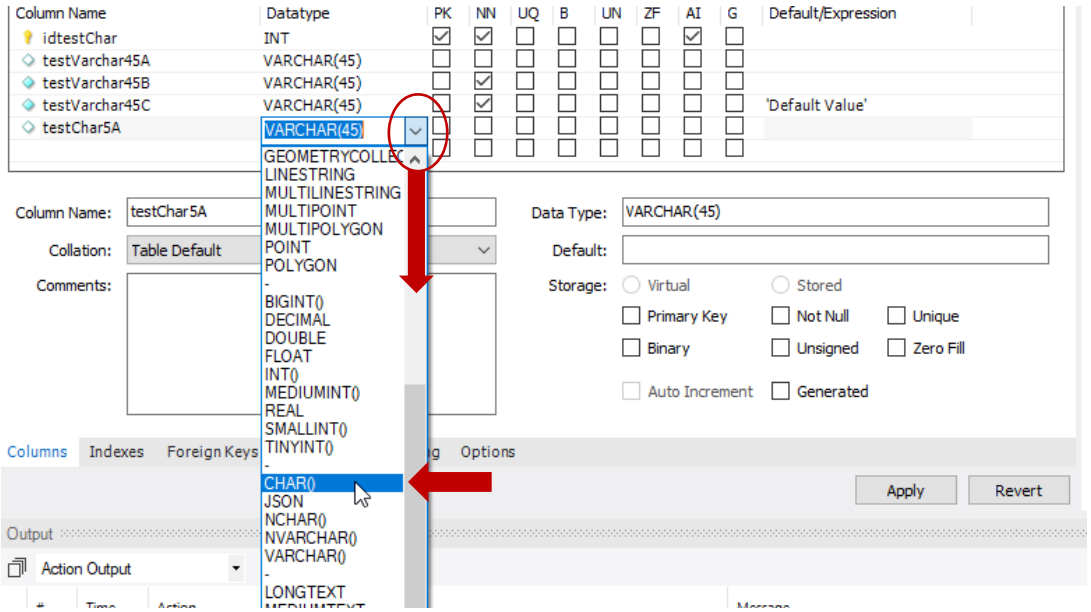
Figure 14: This is what Workbench should look like after adding the columns and setting the attributes.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

_____ Enter another column. Give it the name `testChar5A`.

_____ Tab to or click the **Datatype** entry space for the `testChar5A` column.

_____ Click the drop-down arrow to view the list of datatypes. Scroll down through the drop-down list and select the **CHAR()** datatype, as shown in Figure 15.



L02_0055

Figure 15: Choose the CHAR() datatype from the drop-down list.

_____ Workbench displays **CHAR()** in the datatype column.

_____ Change the datatype value to **CHAR(5)**.

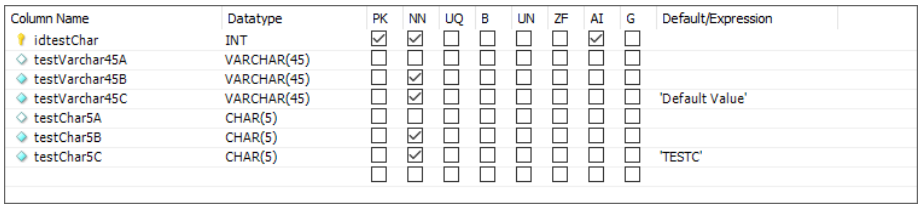
_____ Do not set any other attributes for `testChar5A`.

_____ Add the following two additional columns, with attributes as shown here:

Column name: `testChar5B` `CHAR(5)` **NN** (Not Null)

Column name: `testChar5C` `CHAR(5)` **NN** (Not Null) Default: `'TESTC'`

_____ Workbench should now look like Figure 16.



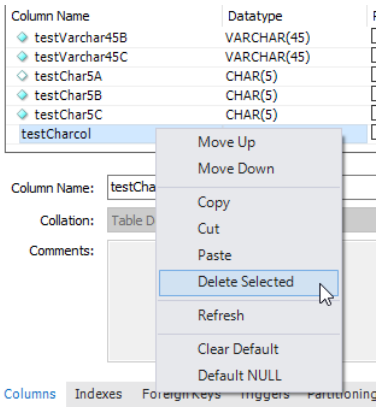
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idtestChar	INT	☑	☑	☐	☐	☐	☐	☑	☐	
testVarchar45A	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	
testVarchar45B	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
testVarchar45C	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	'Default Value'
testChar5A	CHAR(5)	☐	☐	☐	☐	☐	☐	☐	☐	
testChar5B	CHAR(5)	☐	☑	☐	☐	☐	☐	☐	☐	
testChar5C	CHAR(5)	☐	☑	☐	☐	☐	☐	☐	☐	'TESTC'
		☐	☐	☐	☐	☐	☐	☐	☐	

L02_0056

Figure 16: This is what Workbench should look like after you add the VARCHAR and CHAR columns

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

_____ If Workbench added another column that you don't want, you can right-click the column name and select the **Delete Selected** item from the pop-up menu, as shown in Figure 17. You can also use the **Cut** item to remove the unwanted column.



L02_0059

Figure 17: You can right-click and use the Delete Selected item if you need to remove a column.

2.3.3 Create the table

You now have definitions for seven columns:

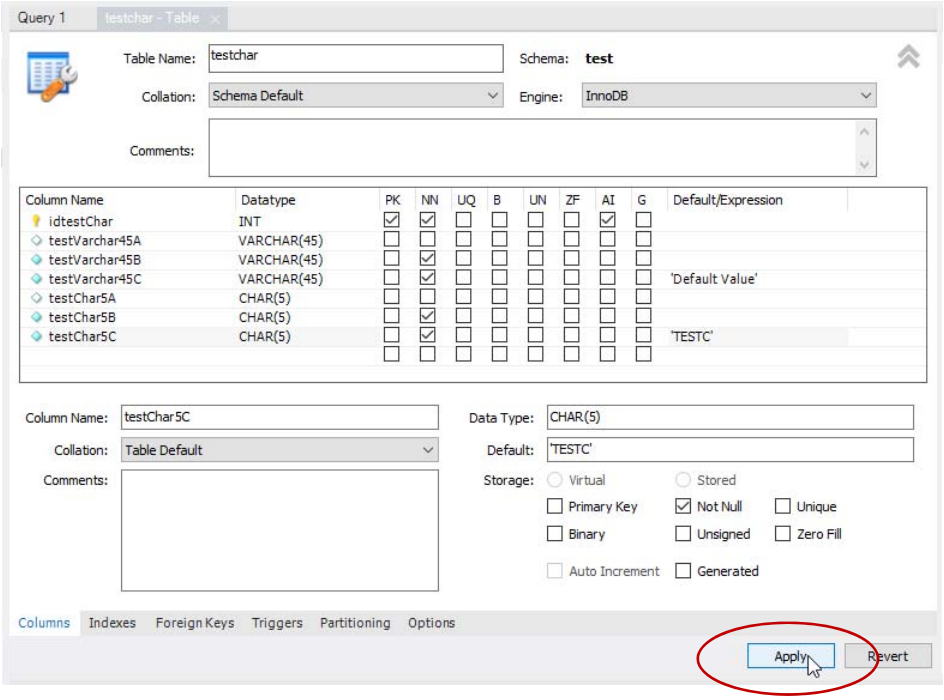
1 primary key column

3 VARCHAR(45) columns

3 CHAR(5) columns

The table is not yet created. In this section, you will finish creating the table, using the column definitions that you entered in the previous sections.

_____ Click the **Apply** button that is in the lower right corner of the table definition tab, as shown in Figure 18.



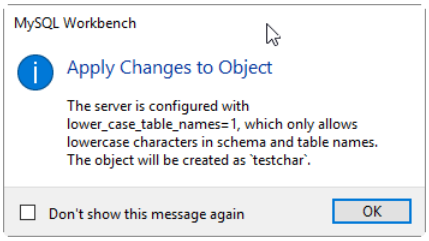
L02_0061

Figure 18: Click the Apply button in the lower right corner of the panel.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

_____ **[WINDOWS VERSION]** If you named the table `testChar` (in Section 2.3.1 on page 5), Workbench displays the message shown in Figure 19. By default, MySQL only allows lowercase letters in table and schema names.

_____ **[WINDOWS VERSION]** Click the **OK** button to continue. The table name is set to `testchar`.

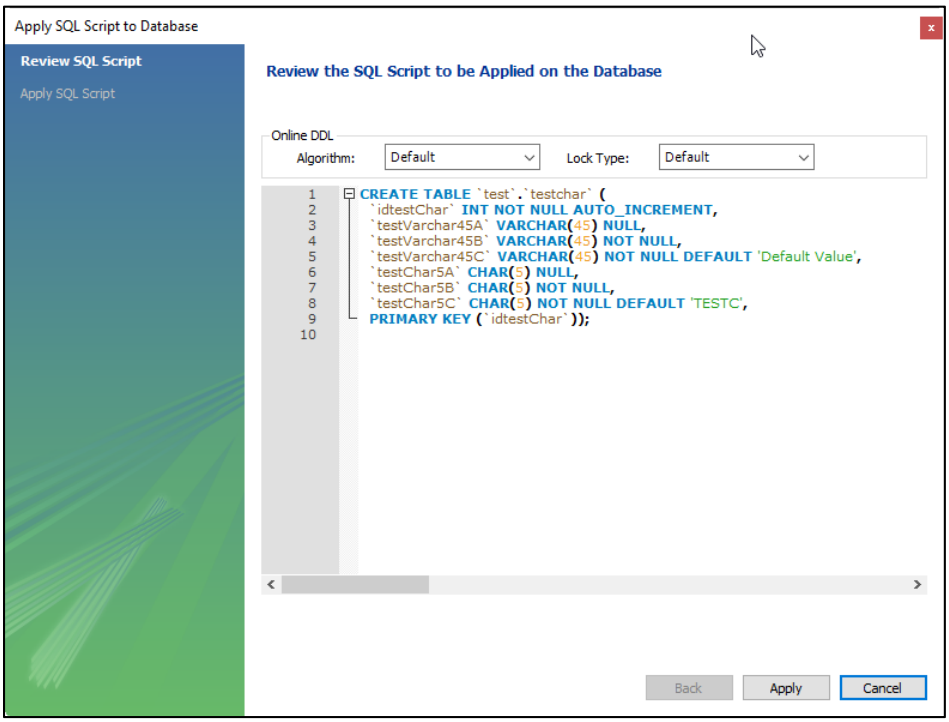


L02_0062

Figure 19: This message is displayed because of the table name.

_____ The **Apply SQL Script to Database** panel shown in Figure 20 is displayed. This panel shows the `CREATE TABLE` DDL statement that will be run to create the table. You can correlate the SQL keywords and column names with the values that you entered and selected in Workbench.

_____ Click the **Apply** button.



L02_0065

Figure 20: The Apply SQL Script to Database panel shows the CREATE TABLE statement that will be run to create the table.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

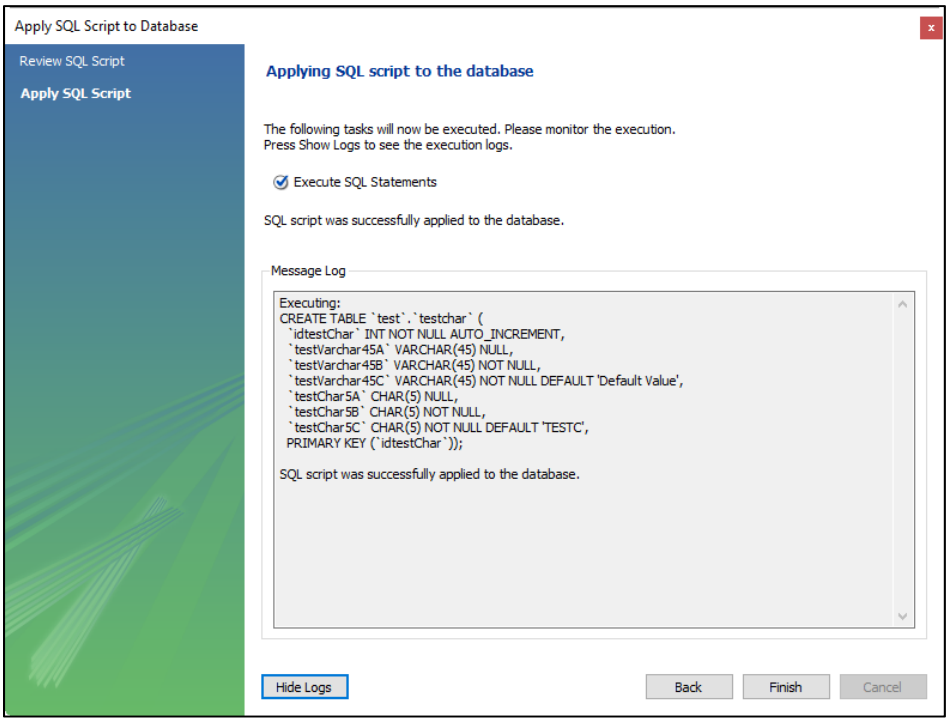
_____ The SQL script to create the table is executed and the next panel is displayed (Figure 21).

_____ Click the **Show Logs** button on the panel.

_____ The panel changes to show the results of running the `CREATE TABLE` statement.

_____ **[WINDOWS VERSION]** Click the **Finish** button after reviewing the log.

_____ **[MAC VERSION]** Click the **Close** button after reviewing the log.



L02_0066

Figure 21: Use the Show Logs button to review the result of running the CREATE TABLE.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

2.3.4 Verify that the table was added to the schema

_____ Now go back to the **SCHEMAS** section of the Navigator (left side of Workbench).

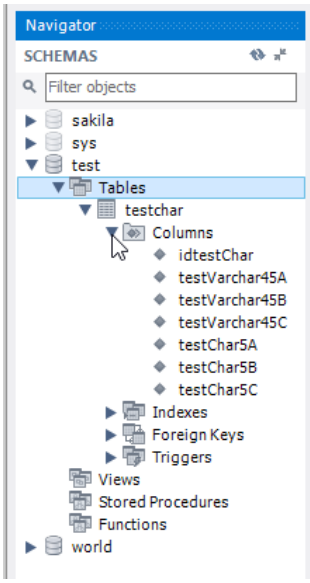
_____ Expand the `test` schema.

_____ Expand the **Tables** item.

_____ Expand the `testchar` table.

_____ Expand the **Columns** item.

_____ You should see the columns that are in the `testchar` table, as shown in Figure 22.



L02_0069

Figure 22: You should now see the testchar table in the test schema.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

2.3.5 Enter test data into the testchar table

Now that you have a table created, you will enter some test data into the table. You'll first enter some valid data, then you'll enter some invalid data. The reason for entering invalid data is so that you can see how MySQL handles invalid data.

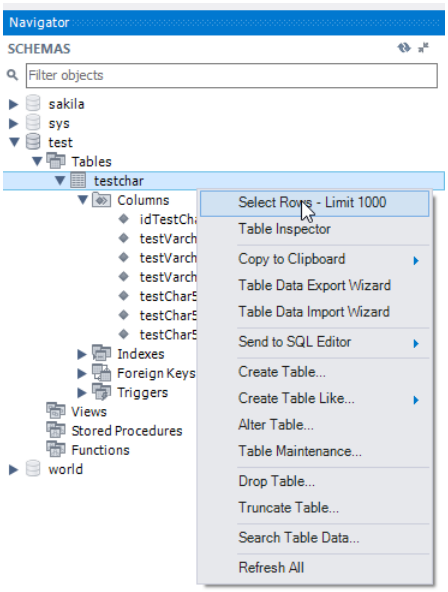
_____ To get started, you'll open a tab to view the data that is in the table. At this point, there is no data in the table so you won't see any data, but you'll leave the tab open so that you can easily view new rows after you enter data.

_____ Do either of the following to open the **Result Grid** for the table:

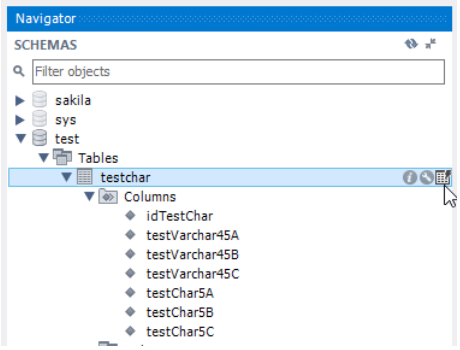
____ Right-click the `testchar` table name in the Navigator and click the **Select Rows** item in the pop-up menu (left side of Figure 23).

or

____ Hover over the `testchar` table name and click the right-most icon (right side of Figure 23).



L02_0101

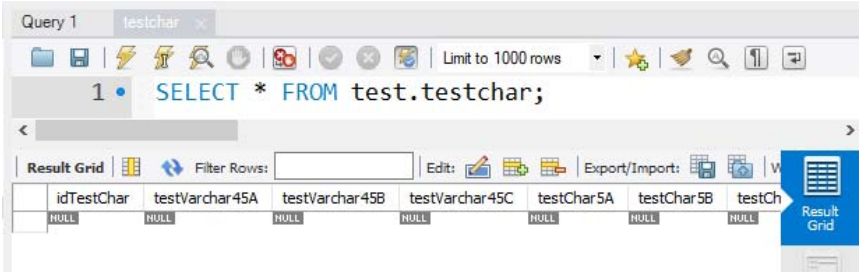


L02_0102

Figure 23: Use the Select Rows menu item or click the icon.

_____ A tab opens. The `SELECT` statement shown in Figure 24 is generated and executed.

_____ The **Result Grid** shown in Figure 24 is displayed. This is what the grid looks like when there is no data in the table.



L02_0105

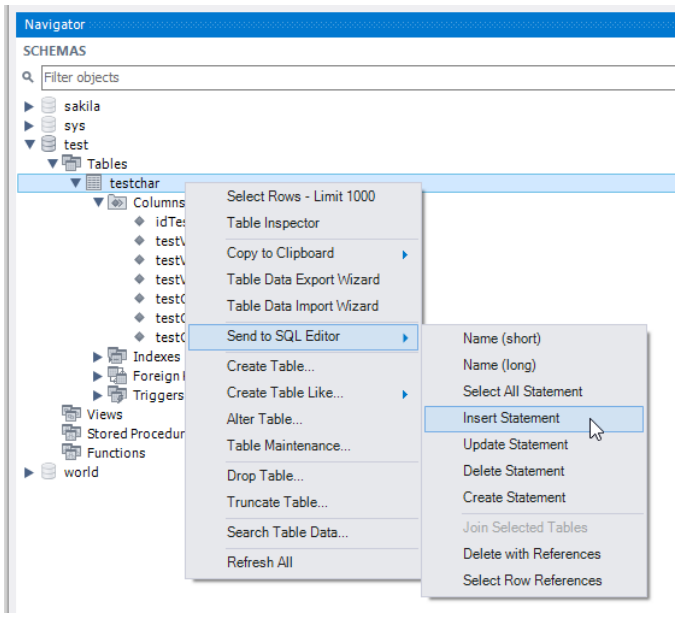
Figure 24: This is what the Result Grid looks like when there is no data in the table.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

_____ Click the **File, New Query Tab** menu items to open a new blank tab in Workbench.

_____ In the Navigator, right-click the `testchar` table.

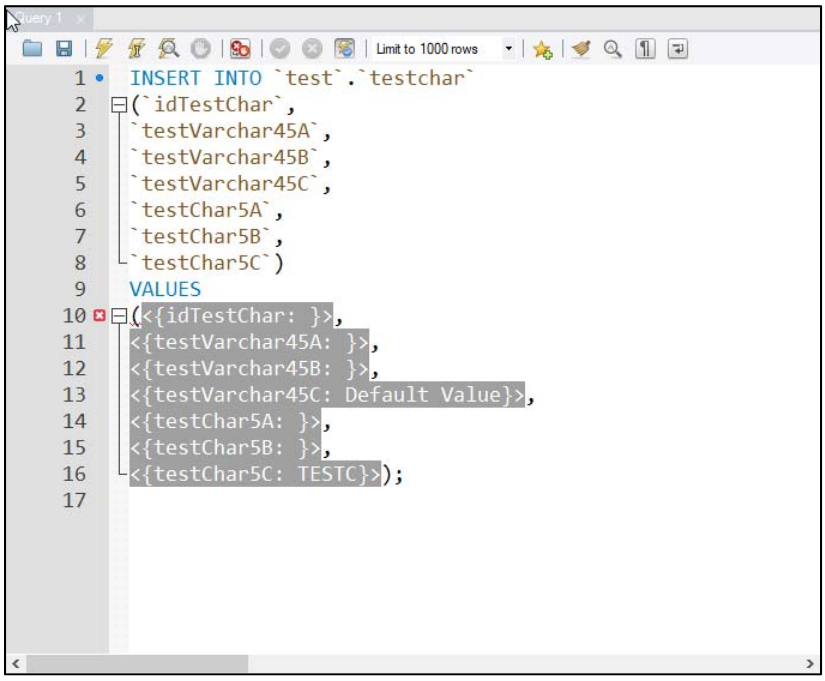
_____ On the pop-up menus, select the **Send to SQL Editor**, **Insert Statement** menu items as shown in Figure 25.



L02_0111

Figure 25: Select the Send to SQL Editor, Insert Statement menu items.

_____ Workbench generates an `INSERT` statement for the `testchar` table, as shown in Figure 26.



L02_0112

Figure 26: Workbench generates an INSERT statement for the table.

The `INSERT` statement contains the column list followed by a `VALUES` list, filled with *placeholders*. You will enter data into the `VALUES` list and run the `INSERT` statement to write data to the table.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

_____ The `idTestChar` column was defined as a *primary key, not null, auto-increment* column. Because this is an auto-increment column, the value that will be assigned to it will be generated.

_____ Do the following to remove the `idTestChar` column and its `VALUES` placeholder:

____ Remove the `` `idTestChar` ``, text that is the first column listed in the columns list (in Figure 26, it is on line 2 of the `INSERT` statement).

____ Remove the `<{idTestChar: }>`, text from the placeholder section of the `INSERT` statement (in Figure 26, it is on line 10 of the `INSERT` statement).

_____ Verify that the statement now looks like Figure 27.

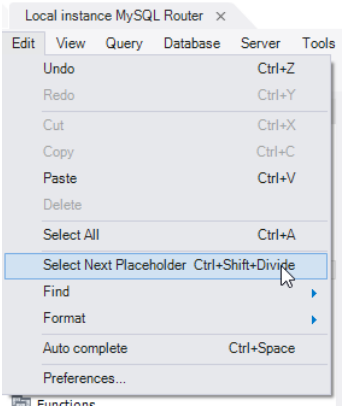
```
1  INSERT INTO `test`.`testchar`
2  (
3  `testVarchar45A`,
4  `testVarchar45B`,
5  `testVarchar45C`,
6  `testChar5A`,
7  `testChar5B`,
8  `testChar5C`)
9  VALUES
10 (
11 <{testVarchar45A: }>,
12 <{testVarchar45B: }>,
13 <{testVarchar45C: Default Value}>,
14 <{testChar5A: }>,
15 <{testChar5B: }>,
16 <{testChar5C: TESTC}>);
17
```

L02_0113

Figure 27: This is what the INSERT statement looks like after you remove the idTestChar column and its placeholder.

_____ Click the mouse cursor somewhere on the `VALUES` statement (line 9 in Figure 27). It does not matter where on the statement you click, the purpose is to simply select that statement (it will be highlighted light blue when it is selected).

_____ Click the **Edit** menu, then click the **Select Next Placeholder** menu item, as shown in Figure 28.



L02_0114

Figure 28: Click the Edit, Select Next Placeholder menu item.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

_____ The first placeholder is now selected, as shown in Figure 29. The placeholder is highlighted in blue when it is selected.

```
INSERT INTO `test`.`testchar`
(
`testVarchar45A`,
`testVarchar45B`,
`testVarchar45C`,
`testChar5A`,
`testChar5B`,
`testChar5C`)
VALUES
(
<{testVarchar45A: }>,
<{testVarchar45B: }>,
<{testVarchar45C: Default Value}>,
<{testChar5A: }>,
<{testChar5B: }>,
<{testChar5C: TESTC}>);
```

L02_0115

Figure 29: The first placeholder is now selected.

_____ Type a single quote character (not the tick character, use the single quote that is next to the **Enter** key). Then type some text, then a closing single quote. Figure 30 shows an example of some text with the surrounding single quote characters. It does not matter what text you enter.

```
INSERT INTO `test`.`testchar`
(
`testVarchar45A`,
`testVarchar45B`,
`testVarchar45C`,
`testChar5A`,
`testChar5B`,
`testChar5C`)
VALUES
(
'First test value',
<{testVarchar45B: }>,
<{testVarchar45C: Default Value}>,
<{testChar5A: }>,
<{testChar5B: }>,
<{testChar5C: TESTC}>);
```

L02_0116

Figure 30: A value is now entered for the first column.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

_____ Use the **Edit, Select Next Placeholder** menu item again to select the next placeholder.

_____ Enter the a single quote delimited string for the second placeholder.

_____ Repeat the **Edit, Select Next Placeholder** for the remaining placeholders. Enter a text value for each placeholder.

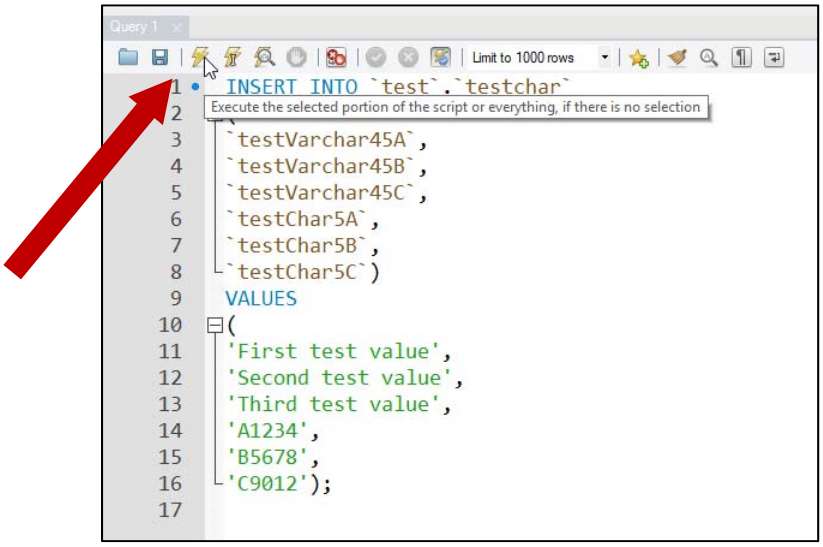
_____ When done, the `INSERT` statement should look like Figure 31 (the text values that you enter may be different from the figure).

```
1  INSERT INTO `test`.`testchar`
2  (
3  `testVarchar45A`,
4  `testVarchar45B`,
5  `testVarchar45C`,
6  `testChar5A`,
7  `testChar5B`,
8  `testChar5C`)
9  VALUES
10 (
11 'First test value',
12 'Second test value',
13 'Third test value',
14 'A1234',
15 'B5678',
16 'C9012');
17
```

L02_0117

Figure 31: This is the INSERT statement after you enter text values for all of the placeholders.

_____ Click the **Execute** icon, as shown in Figure 32. It is the third icon from the left, and it is meant to look like a lightening bolt.

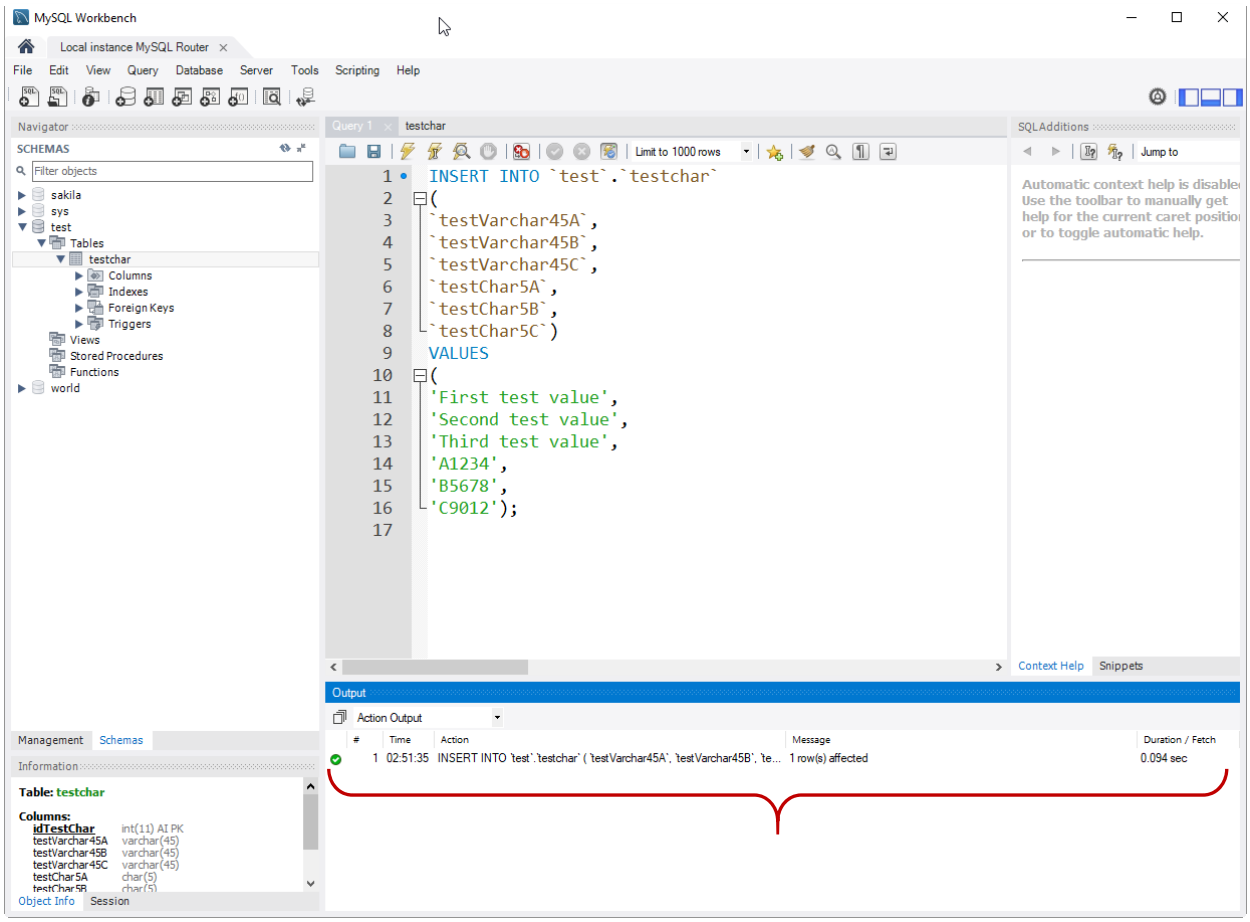


L02_0121

Figure 32: Click the Execute icon to run the INSERT statement.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

_____ The **Output** panel at the bottom of Workbench shows the result of executing the `INSERT` statement. In this example, you should have a successful execution (the green icon at the left of the **Output** panel).



L02_0122

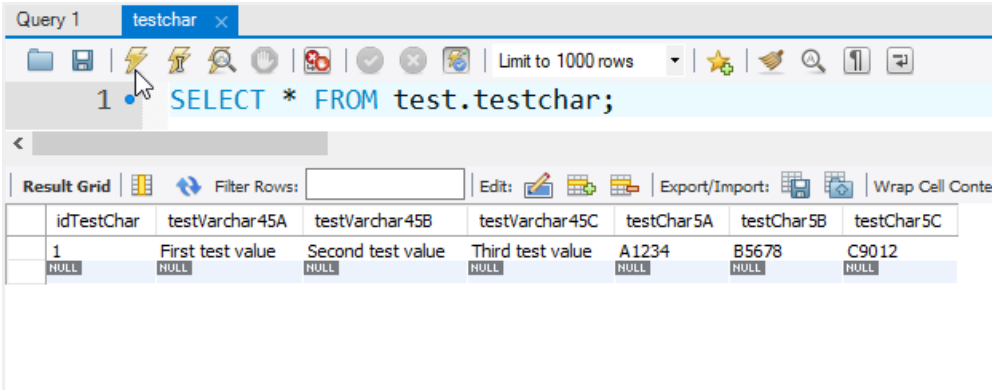
Figure 33: The Output panel shows the result of executing the INSERT statement.

_____ You should still have the tab for the **Result Grid** in Workbench that was used earlier to display the rows in the table (see Figure 23 and Figure 24 on page 13).

_____ Click the tab header to make that tab (the tab with the **Result Grid**) the current tab.

_____ Click the **Execute** icon on that tab. The `SELECT` statement that is still in the tab runs.

_____ You should see the newly added data in the table, as shown in Figure 34.



L01_0123

Figure 34: You can now see the first row of data in the table.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

2.3.6 Enter additional test data in the testchar table

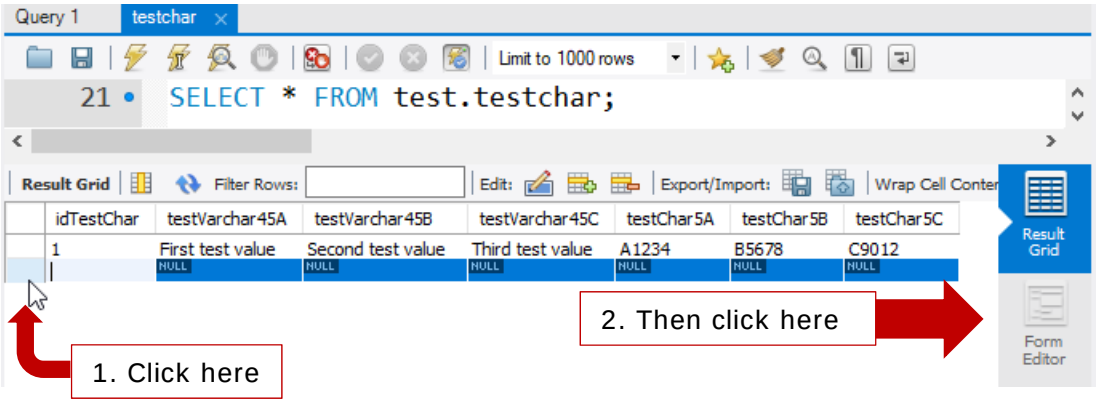
Now that you've seen how you can insert a row using the prompted `INSERT` statement, you'll use another feature of Workbench to insert rows into your table. The technique shown in this section may be easier to use. You'll use the technique because it will clearly show errors in the data when run the insert.

_____ You should be on the tab showing the Result Grid, as shown in Figure 34.

_____ In the **Result Grid**, click the far left column on the second row, as shown in Figure 35.

_____ When the rows is selected, you should see it highlighted, as shown in the figure.

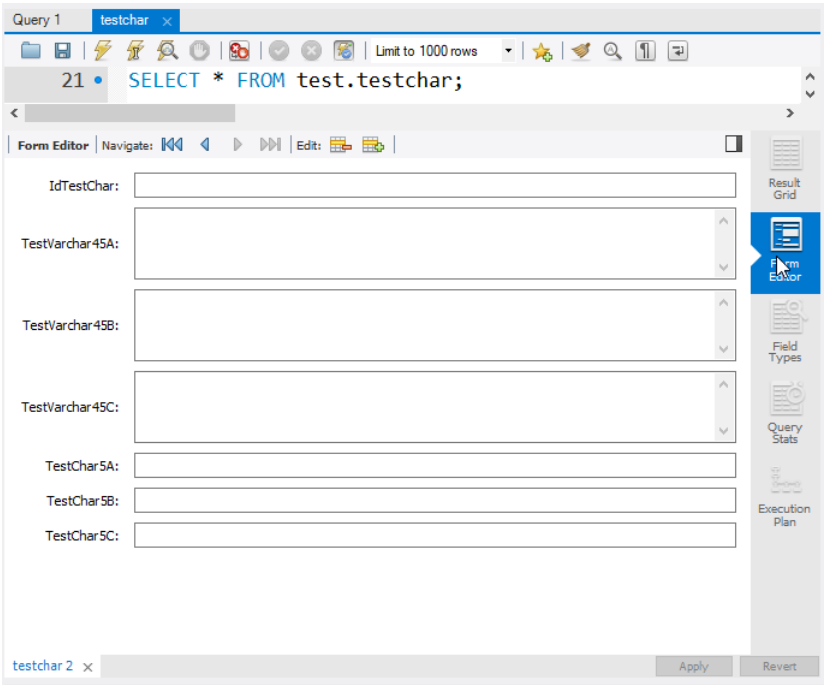
_____ With the second row selected, click the **Form Editor** icon on the right of the grid, as shown in the figure.



L02_0131

Figure 35: Click the leftmost column to select the second row in the grid.

_____ The **Form Editor** is displayed, as shown in Figure 36. It contains an entry field for each column in the table.



L02_0132

Figure 36: This is the Form Editor.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

_____ Enter the following values in the **Form Editor**. **Be careful that you do not make any entries, not even blank spaces, for any of the columns not shown here.**

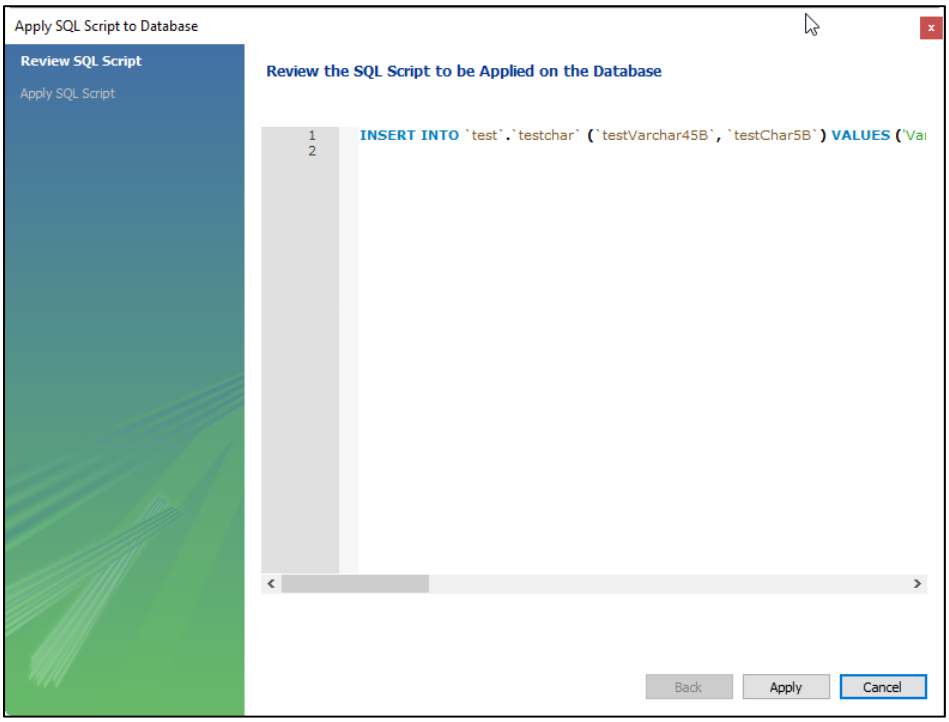
TestVarchar45B:	`Varchar 45 B`
TestChar5B:	`Char 5B`

_____ After entering those two values, click the **Apply** button in the lower right corner of the tab.

_____ The **Apply SQL Script to Database** panel shown in Figure 37 is displayed.

_____ Check the `INSERT` statement shown in the panel. Verify that only two column names are listed, as shown in the figure (`testVarchar45B`, `testChar5B`). If there are fewer or more columns, click the **Cancel** button to return to the **Form Editor** and make corrections. [**MAC** Close the window.]

_____ If everything looks OK, click the **Apply** button.



L02_0135

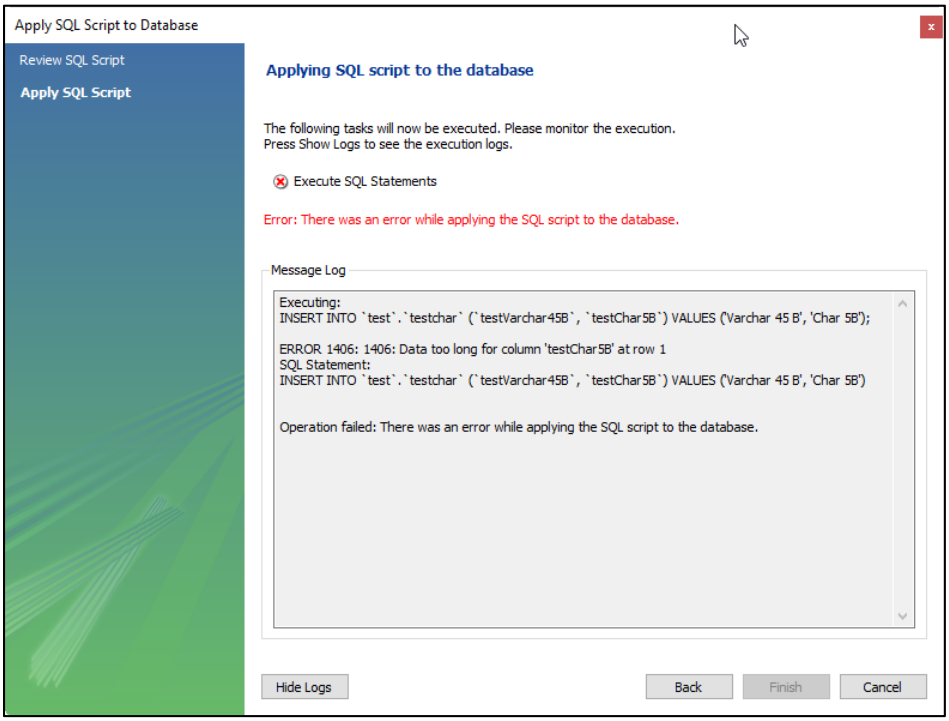
Figure 37: The Apply SQL Script to Database panel is shown.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

_____ You should see the error message shown on the panel in Figure 38. The error is displayed because the value entered for the `testChar5B` column is too long. The test value

```
Char 5B
```

is longer than 5 characters, which is the maximum number of characters that can be stored in the `testChar5B` column.



L02_0136

Figure 38: You should see the error message, as shown on this panel.

_____ **[WINDOWS]** Click the **Cancel** button on the panel.

_____ **[MAC]** Close the window.

_____ Back on the **Form Editor** panel, change the value for **TestChar5B** to `C5B`.

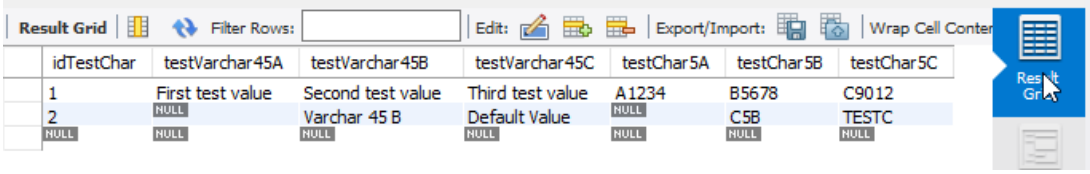
_____ Click the **Apply** button on the **Form Editor** panel.

_____ Click the **Apply** button on the **Apply SQL Script to Database** panel.

_____ This time, the script should run successfully. Click the **Finish** [**MAC: Close**] button to close the panel.

_____ Back in Workbench, click the **Result Grid** icon. It is directly above the **Form Editor** icon that you clicked earlier (see Figure 35 on page 19).

_____ You should now see the second row in the table, as shown in Figure 39. You entered values for the `testChar45B` and `testChar5B` columns. For all of the columns in row 2 that you did not specify a value for, they have been assigned either the default value or `NULL`.



L02_0137

Figure 39: You should now see the second row, with default and NULL values.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

2.3.7 More about CHAR and VARCHAR datatypes

For review, here is the `CREATE TABLE` statement that Workbench used to create the `testchar` table. You can see this statement in Workbench by doing the following:

_____ Use the **File, New Query Tab** menu item to open a new tab. The tab will be empty when it is opened.

_____ In **Navigator**, right-click the `testchar` table.

_____ Click the **Send to SQL Editor, Create Statement** menu items on the pop-up menus.

_____ You'll see a `CREATE TABLE` statement similar to the following in the query tab.

```
CREATE TABLE `testchar` (
  `idTestChar`     int(11)     NOT NULL AUTO_INCREMENT,
  `testVarchar45A` varchar(45)          DEFAULT NULL,
  `testVarchar45B` varchar(45) NOT NULL,
  `testVarchar45C` varchar(45) NOT NULL DEFAULT 'Default Value',
  `testChar5A`     char(5)              DEFAULT NULL,
  `testChar5B`     char(5)     NOT NULL,
  `testChar5C`     char(5)     NOT NULL DEFAULT 'TESTC',
  PRIMARY KEY (`idTestChar`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
```

Column Name	Datatype	Can it be NULL?	Default if no value entered
idTestChar	INT(11)	No	Auto-increment value, generated by MySQL
testVarChar45A	VARCHAR(45)	Yes	NULL
testVarChar45B	VARCHAR(45)	No	
testVarChar45C	VARCHAR(45)	No	Default Value
testChar5A	CHAR(5)	Yes	NULL
testChar5B	CHAR(5)	No	
testChar5C	CHAR(5)	No	TESTC

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

What does VARCHAR and CHAR mean?

For the past several pages, you've been using the VARCHAR and CHAR column datatypes.

CHAR stands for *character*. You can enter any letter, number or special character. When you define a CHAR column, you also specify the column length. The length is the maximum number of characters that can be stored in the column. It is also the length that is always used to store the data.

Example: you define a column as CHAR(100). You insert a string with the value 'TEST CHAR'. Even though that string only contains 9 characters (the embedded blank counts as a character), 100 characters are stored in the column.

VARCHAR stands for *variable length character*. Like CHAR, you can store letters, numbers and special characters. The length you specify is the maximum length. The actual length that is stored is the number of characters you supply.

Example: you define a column as VARCHAR(100). You insert a string with the value 'TEST VARCHAR'. That string contains 12 characters. The database stores the value the column using 13 characters. It uses 12 characters to store the data and one byte to specify the length of the data that is stored in the column.

Should you use CHAR or VARCHAR?

A column defined as a VARCHAR datatype always includes 1 or 2 *length bytes* in addition to the actual data that is stored. If the VARCHAR length is 255 or less, only one length byte is needed. If the VARCHAR length is between 256 and 65535 (the maximum length for a VARCHAR column), two length bytes are used.

A column defined as a CHAR datatype does not need a length byte. The maximum length of a CHAR column is 255.

If you need to store a string with an exact length, use CHAR. Some applications are easier to program and work with if you use exact length data.

Most of the time, you can and should use VARCHAR data. For example, if you have a web page where people can enter name and address information, you can store the first name, last name, address, city and ZIP code as VARCHAR data. You don't have to be overly concerned with figuring out how long each column should be, you can define all of the columns as VARCHAR(50) for example.

For the state code, you can define a CHAR(2) column or use VARCHAR.

For the ZIP code, you might first consider storing it as an INT (integer) datatype, since a ZIP code is numeric. But some ZIP codes begin with a leading zero, and some ZIP codes are written as ZIP+4 fields, usually with an embedded hyphen character. For data that "looks like numbers" but in fact is not a numeric value, you're better off defining the column using a CHAR or VARCHAR datatype.

To find more information about CHAR and VARCHAR datatypes

11.4.1 The CHAR and VARCHAR Types

https://dev.mysql.com/doc/refman/5.7/en/char.html

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

2.3.8 Run additional tests for CHAR and VARCHAR

Now that you've seen how to work with the **Result Grid** and the **Form Editor**, you will run some additional tests. These tests will show what happens when you enter values that are too long, when you do not enter a value, and when you try to use `NULL` where it is not allowed.

2.3.8.1 Test: no entry for columns that do not have a default value

Purpose of this test: to see how MySQL handles an `INSERT` when there is no data provided for a column that does not have a default value.

_____ You should be on the **Result Grid** for the `testchar` table.

_____ Using Figure 35 on page 19 as a guide, select the next available row in the grid, then click the **Form Editor** icon.

_____ Enter the following values for the columns specified. Do not enter anything, not even blank spaces, in any other columns:

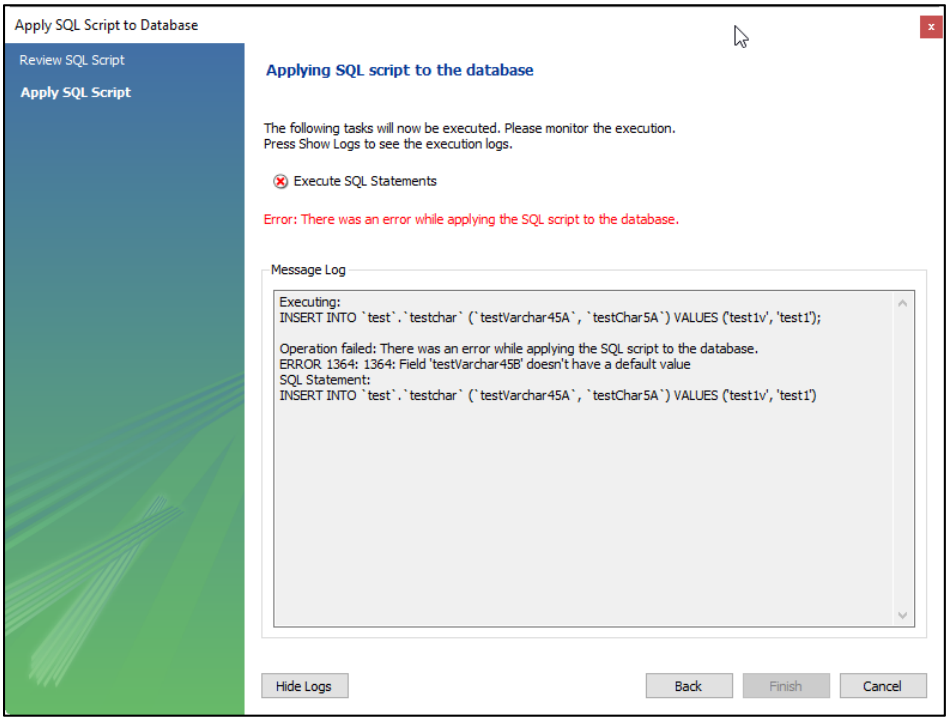
TestVarChar45A	`test1V`
TestChar5A	`test1`

_____ Click the **Apply** button.

_____ On the **Apply SQL Script to Database** panel, click the **Apply** button.

_____ The error shown in Figure 40 is displayed. This error is displayed because the `testVarchar45B` column was not defined with a default value (see the table definition on page 22) and no value was specified for the column on the **Form Editor** panel.

_____ Click the **Cancel** button to close the panel.



L02_0141

Figure 40: This error is shown because testVarChar45B does not have a default value.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

_____ Go back to the **Form Editor**. Enter a value for the **TestChar45B** field and click the **Apply** button twice.

_____ This time you will get an error for the `testChar5B` column.

_____ Close the **Apply SQL Script to Database** panel.

_____ Go back to the **Form Editor**, enter a value for the **TestChar5B** field (5 characters or less) and click the **Apply** button twice.

_____ This time, the **Apply SQL Script to Database** panel should show a successful completion. Close the panel.

_____ In Workbench, click the **Result Grid** icon to switch from the **Form Editor** to the **Result Grid**.

_____ You should see the new row in the table with a value for all of the columns:

> You specified values for the `testVarchar45A`, `testVarchar45B`, `testChar5A` and `testChar5B` columns
>
> Default values were supplied tor the `testVarchar45C` and `testChar5C` columns
>
> MySQL generated the value for the `idTestChar` column

Results of this test: this test showed that you must provide a value for a column if the column does not have a default value defined for it.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

2.3.8.2 Test: entering a value that is too long

Purpose of this test: to see how MySQL handles an `INSERT` when there the value provided for a column is longer than the maximum number of characters.

_____ In the **Result Grid** select the next available row, then go to the **Form Editor** for the row.

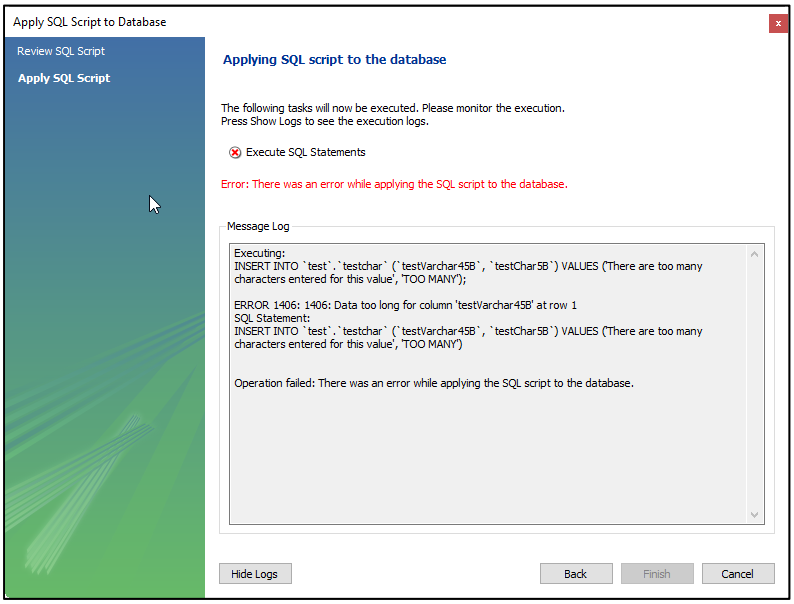
_____ Enter the following values for the columns specified. Do not enter anything, not even blank spaces, in any other columns:

TestVarChar45B	`There are too many characters entered for this value`
TestChar5B	`TOO MANY`

_____ Click the **Apply** button.

_____ On the **Apply SQL Script to Database** panel, click the **Apply** button.

_____ The error panel shown in Figure 41 is displayed.



L02_0142

Figure 41: This is the error that is displayed when too many characters are entered for a column.

_____ Close the panel. Go back to the **Form Editor** and change both **TestVarChar5B** and **TestChar5B** so that there is less data.

_____ Click the **Apply** button and verify that the data is inserted into the table.

_____ Go to the **Result Grid** and verify that the new row was inserted.

Result of this test: this test showed that you cannot enter a value that is longer than the maximum length defined for the column.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

2.3.8.3 Test: override the default values

Purpose of this test: to see how MySQL handles an `INSERT` when the default values for columns are overridden.

In the `testchar` table, the following columns have default values. When a new row is inserted into the table, the default value will be assigned to a column unless a value is specified for the column on the `INSERT` statement.

Column	Default Value
testVarChar45A	NULL
testVarChar45C	Default Value
testChar5A	NULL
testChar5C	TESTC

_____ In the **Result Grid** select the next available row, then go to the **Form Editor** for the row.

_____ Enter the following values for the columns specified. Do not enter anything, not even blank spaces, in any other columns:

TestVarChar45A	`Override the NULL value`
TestVarChar45B	`No default for this column`
TestVarChar45C	`Override the Default value`
TestChar5A	`OVR1`
TestChar5B	`NODEF`
TestChar5C	`OVR2`

_____ Click the **Apply** button and verify that the data is inserted into the table.

_____ Go to the **Result Grid** and verify that the new row was inserted.

Result of this test: this test showed that you can enter a value that overrides the default value that is specified for a column.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

2.4 Create a new table to work with numeric datatypes

In this section, you will create the `testnumeric` table. You will define several columns, using the different numeric datatypes. After creating the table, you will enter test data, using the **Form Editor** and **Result Grid**.

_____ Expand the **Tables** item that is under the `test` schema name.

_____ Right-click the **Tables** item.

_____ Click the **Create Table** item in the pop-up menu.

_____ Using Figure 42 as a guide, create a new table named `testnumeric`.

NOTE: in the **Datatype** drop-down list for the `testNumeric` column, select the **NUMERIC** item. When you tab out of the drop-down list, Workbench changes the **Datatype** to **DECIMAL**, as shown in the figure.

NOTE: in the **Datatype** drop-down list for the `testBoolean` column, select the **BOOLEAN** item. When you tab out of the drop-down list, Workbench changes the **Datatype** to **TINYINT**, as shown in the figure.

NOTE: use the steps shown in Section 2.3.3 on page 9 if you need to review how to create a table using Workbench.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idtestnumeric	INT(11)	☑	☑	☐	☐	☐	☐	☑	☐	
testint	INT	☐	☑	☐	☐	☐	☐	☐	☐	
testIntUnsigned	INT	☐	☑	☐	☐	☑	☐	☐	☐	
testIntZeroFill	INT(5)	☐	☑	☐	☐	☑	☑	☐	☐	
testDecimal	DECIMAL(5,2)	☐	☑	☐	☐	☐	☐	☐	☐	
testNumeric	DECIMAL(5,2)	☐	☑	☐	☐	☐	☐	☐	☐	
testFloat	FLOAT	☐	☑	☐	☐	☐	☐	☐	☐	
testDouble	DOUBLE	☐	☑	☐	☐	☐	☐	☐	☐	
testBoolean	TINYINT	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

L02_0151

Figure 42: Use the column names, datatypes and attributes shown here for the testnumeric table.

_____ The following `CREATE TABLE` statement is generated based on the values shown in Figure 42.

```
CREATE TABLE `testnumeric` (
  `idtestnumeric`   int(11)                        NOT NULL AUTO_INCREMENT,
  `testint`         int(11)                        NOT NULL,
  `testIntUnsigned` int(10)      unsigned          NOT NULL,
  `testIntZeroFill` int(5)       unsigned zerofill NOT NULL,
  `testDecimal`     decimal(5,2)                   NOT NULL,
  `testNumeric`     decimal(5,2)                   NOT NULL,
  `testFloat`       float                          NOT NULL,
  `testDouble`      double                         NOT NULL,
  `testBoolean`     tinyint(4)                     NOT NULL,
  PRIMARY KEY (`idtestnumeric`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;
```


Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

2.4.1 Enter test data into the testnumeric table

You'll now run some tests using the `testnumeric` table. You'll enter some valid values and some invalid values to see how MySQL works with the values.

_____ Open the **Result Grid** for the table (see Figure 23 on page 13 to review the steps to open the table to the grid).

_____ Select the empty row, then click the **Form Editor** icon.

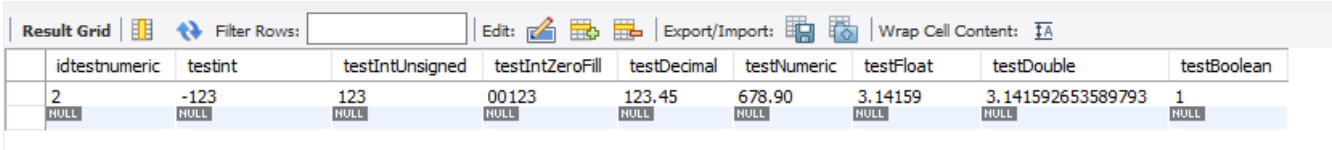
_____ Enter the following values for the columns in the `testnumeric` table:

Column	Value to enter
TestInt	-123
TestIntUnsigned	123
TestIntZeroFill	123
TestDecimal	123.45
TestNumeric	678.90
TestFloat	3.14159265358979323846264338279502884197 16
TestDouble	3.14159265358979323846264338279502884197 16
TestBoolean	1

Note: you can enter the first 40 digits of Π (Pi), or enter another "long number" of your choice for the `TestFloat` and `TestDouble` columns. Enter a value with at least 20 decimal digits.

_____ Click the **Apply** button on the **Form Editor** panel, then complete the `INSERT` on the following panels.

_____ Go to the **Result Grid**. You will see the numeric values, as shown in Figure 43.



L02_0152

Figure 43: This shows the Result Grid after entering the numeric values.

Note the following about the numeric values:

The `testIntZeroFill` value is padded with zeros on the left

The value that you entered for `testFloat` is truncated

The value that you entered for `testDouble` is truncated

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

2.4.2 Test invalid numeric values

_____ In the **Result Grid**, click the next empty row to select it.

_____ Go into the **Form Editor** for the empty row.

_____ Enter some invalid values and click the **Apply** button. The idea is to get to the panel in the **Apply SQL Script to Database** that shows the errors.

To test the errors, go back to the **Form Editor** and correct one column at a time. After each correction, go through the **Apply** process again to review the next error.

_____ Examples of invalid test data:

TestIntUnsigned	`-123`
TestDecimal	`1234.56`
TestNumeric	`7890.12`

2.4.3 More about numeric columns

Review the following section of the MySQL documentation:

11.2 Numeric Types

https://dev.mysql.com/doc/refman/5.7/en/numeric-types.html

The subsections of the documentation give more details about each of the numeric datatypes.

Here are some things to consider when defining numeric columns for a table:

To store "whole number" values, use the `INT` datatype.

> **Examples:** number of brothers and sisters you have; number of cars in your household; how many times you drank coffee yesterday.

To store numeric values that might have fractional parts, use the `DECIMAL` datatype. In MySQL, `NUMERIC` is used the same as `DECIMAL`, so you can use `NUMERIC` also. Other database management systems might have distinct uses for `DECIMAL` and `NUMERIC`. Use `DECIMAL` when you need to store an exact value, for example, when working with financial data.

> **Examples:** how much you paid for your morning cup of coffee; how many miles (to the nearest tenth of a mile) it is between your home and the nearest coffee shop.

You can also use `FLOAT` or `DOUBLE` to store values that have fractional parts, but these datatypes store approximate values, even though the value might look like an exact value. The approximations might become apparent when you use a `FLOAT` or `DOUBLE` in a calculation. For more information about working with floating point values, see this section of the MySQL documentation:

B.5.4.8 Problems with Floating-Point Values

https://dev.mysql.com/doc/refman/5.7/en/problems-with-float.html

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

2.5 Create a new table to work with date and time datatypes

In this section, you will create the `testdate` table. You will define several columns, using the different date and time datatypes. After creating the table, you will enter test data, using the **Form Editor** and **Result Grid**.

_____ Expand the **Tables** item that is under the `test` schema name.

_____ Right-click the **Tables** item.

_____ Click the **Create Table** item in the pop-up menu.

_____ Using Figure 44 as a guide, create a new table named `testdate`.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idtestdate	INT	☑	☑	☐	☐	☐	☐	☑	☐	
testDateTime	DATETIME	☐	☑	☐	☐	☐	☐	☐	☐	
testDate	DATE	☐	☑	☐	☐	☐	☐	☐	☐	
testTime	TIME	☐	☑	☐	☐	☐	☐	☐	☐	
testYear	YEAR	☐	☑	☐	☐	☐	☐	☐	☐	
testTimeStamp	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
		☐	☐	☐	☐	☐	☐	☐	☐	

L02_0161

Figure 44: Use the column names, datatypes and attributes shown here for the testdate table.

_____ The following `CREATE TABLE` statement is generated based on the values shown in Figure 44.

```
CREATE TABLE `testdate` (
  `idtestdate`    int(11)   NOT NULL AUTO_INCREMENT,
  `testDateTime`  datetime  NOT NULL,
  `testDate`      date      NOT NULL,
  `testTime`      time      NOT NULL,
  `testYear`      year(4)   NOT NULL,
  `testTimeStamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`idtestdate`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

2.5.1 Enter test data into the testdate table

You'll now run some tests using the `testdate` table. You'll enter some valid values and some invalid values to see how MySQL works with the values.

_____ Open the **Result Grid** for the table (see Figure 23 on page 13 to review the steps to open the table to the grid).

_____ Select the empty row, then click the **Form Editor** icon.

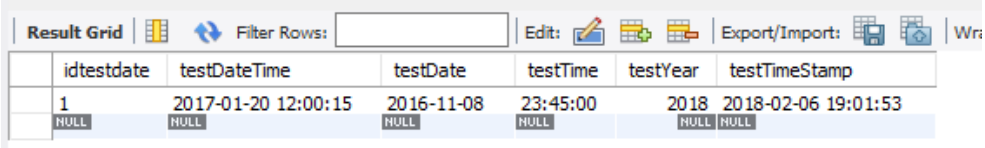
_____ Enter the following values for the columns in the `testdate` table:

Column	Value to enter
TestDateTime	2017-01-20 12:00:15
TestDate	2016-11-08
TestTime	23:45
TestYear	2018
TestTimeStamp	(leave blank, do not enter anything)

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED

_____ Click the **Apply** button the **Form Editor** panel, then complete the `INSERT` on the following panels.

_____ Go to the **Result Grid**. You will see the date and time values, as shown in Figure 43.



L02_0162

Figure 45: This shows the Result Grid after entering the date and time values.

Note the following about the date and time values:

The `testTime` value is shown with seconds, even though you only entered hours and minutes (HH:MM)

The `testTimeStamp` value is based on the `CURRENT_TIMESTAMP` function. The function gets the current system time when the row is inserted.

2.5.2 More information about date and time

Go to the following sections of the MySQL documentation for more information about working with date and time values and datatypes.

9.1.3 Date and Time Literals

https://dev.mysql.com/doc/refman/5.7/en/date-and-time-literals.html

11.3 Date and Time Types

https://dev.mysql.com/doc/refman/5.7/en/date-and-time-types.html

The subsections of the documentation give more details about each of the date and time datatypes.

Here are some things to consider when defining date and time columns for a table:

It is generally a good idea to include a `Timestamp` column in the table, with a default value of `CURRENT_TIMESTAMP`. When you include a column that is set to that default value, the date and time that the row was inserted into the table is recorded. This can be useful when you need to review a table to determine when data was added to it.

You can also include another `Timestamp` column to record the date and time when a row is changed. When you use the SQL `UPDATE` statement, you can set the value of the second `Timestamp` column to the `CURRENT_TIMESTAMP` value. If you use both timestamps, you will be able to tell when the row was initially inserted and when it was last updated.

2.6 Lab 2: Conclusion

This concludes Lab 2. In this lab, you learned how to use features of Workbench to define and work with tables. You defined three test tables to test different data types, and you entered some valid and invalid data into the tables.

Copyright © 2018 Teresa Pelkie, ALL RIGHTS RESERVED