EXCEL APPLICATIONS

11 Steps for Advanced Users

Sergey Vaselenko

EXCEL APPLICATIONS

11 Steps for Advanced Users

Written by Sergey Vaselenko

This e-book shows how to create feature-rich multi-user Excel applications with no SQL or VBA use. You may repeat the steps and create a lot of applications for your team yourself.

Introduction

Microsoft Excel is an amazing product.

It brings freedom to business users as allows getting data from anywhere and consume the data in any way.

If you like Microsoft Excel, you definitely have to try the SaveToDB add-in for Microsoft Excel.

The add-in allows database and VBA professionals to create complex corporate applications using Excel.

Moreover, the add-in allows business users to create Excel applications with no SQL or VBA use.

In this book, we will create an application and learn 11 steps that you may repeat for your applications:

1. Publish tables to a database
2. Configure query parameters
3. Configure formats and table views
4. Configure validation lists
5. Add cursors and form fields
6. Configure detail tables, windows, and task panes
7. Configure context and action menus
8. Add image URLs
9. Configure LastModified and UserName fields
10. Integrate with other applications
11. Manage permissions

You have to download and install the SaveToDB add-in at www.savetodb.com, version 7.2 or higher.

All features described in this book are available in the free SaveToDB Express edition.

You may download the initial workbook, the final application, and required SQL codes at

https://www.savetodb.com/download.php?file=11-steps-for-advanced-users.zip

This book contains an example database for Microsoft SQL Server.

You may also use Oracle Database, IBM DB2, MySQL, and other supported databases. The steps remain the same.

Best regards,

Sergey Vaselenko

March 20, 2017

Table of Contents

Introduction 1

Table of Contents 2

Chapter 1. Initial Workbook 3

Chapter 2. Configuring Database 4

Chapter 3. Publish Wizard 7

Chapter 4. Data Connection Wizard 16

Chapter 5. Query Parameters 20

Chapter 6. Table Views 22

Chapter 7. Table Format Wizard 26

Chapter 8. Validation Lists 29

Chapter 9. Cursors 37

Chapter 10. Form Fields 38

Chapter 11. Master-Details 39

Chapter 12. Detail Windows and Task Panes 43

Chapter 13. Context Menus 45

Chapter 14. Actions Menus 47

Chapter 15. Image URLs 48

Chapter 16. LastModified and UserName 49

Chapter 17. Integration with Other Apps 53

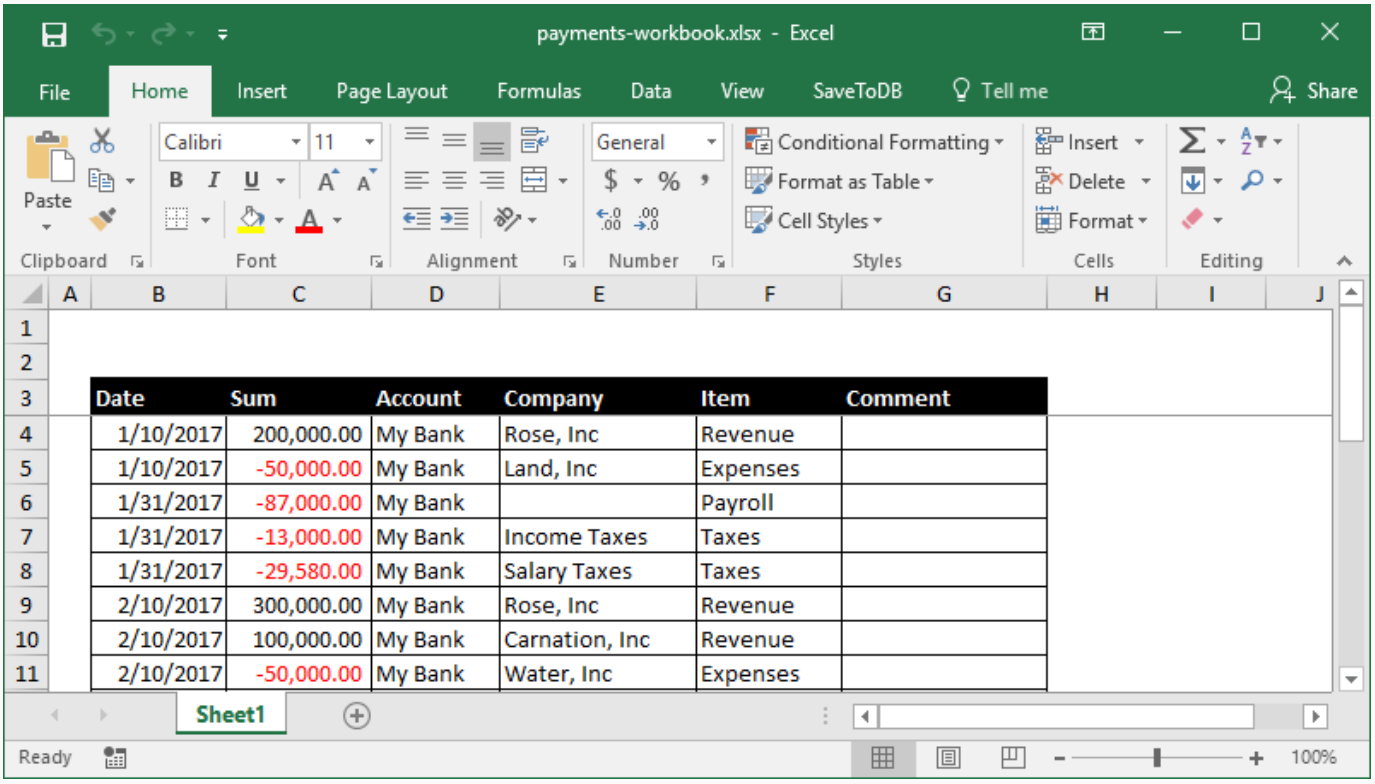
Chapter 18. Permission Management 58

Conclusion 64

About the Author 65

Chapter 1. Initial Workbook

We will create an Excel application based on the following simple workbook of payments:



Date	Sum	Account	Company	Item	Comment
1/10/2017	200,000.00	My Bank	Rose, Inc	Revenue	
1/10/2017	-50,000.00	My Bank	Land, Inc	Expenses	
1/31/2017	-87,000.00	My Bank		Payroll	
1/31/2017	-13,000.00	My Bank	Income Taxes	Taxes	
1/31/2017	-29,580.00	My Bank	Salary Taxes	Taxes	
2/10/2017	300,000.00	My Bank	Rose, Inc	Revenue	
2/10/2017	100,000.00	My Bank	Carnation, Inc	Revenue	
2/10/2017	-50,000.00	My Bank	Water, Inc	Expenses	

Suppose, we have a team that uses this workbook: Alex as a leader, Nick, and Lora.

In common case, such workbooks are located in shared folders or corporate portals.

Simultaneous editing of such workbooks is not easy.

Also, every user periodically wants to add personal worksheets or formulas and do not share them with others.

We will solve all these issues placing data tables to a database.

Moreover, you may do this yourself. SQL or VBA knowledge is not required.

Chapter 2. Configuring Database

Our application requires a database. Also, you must have enough permissions to create and edit tables.

You may say to your IT guys the following:

1. I need a schema in a database.
2. My team will create tables in this schema.
3. I want to manage permissions on our tables.
4. My team members: Nick, Lora, and I am, Alex.
5. Create two tables, EventHandlers and TableFormats, using the attached SQL codes.

It is an easy task for your IT staff. Just send this chapter to him or her.

Further, we will use the **dbo69** schema in the **Test2** database and user names like **Alex**, **Nick**, and **Lora**.

You may skip the comments below and continue to the next chapter.

SQL Scripts

The following script creates logins and users in the **master** database:

```
USE master
GO

CREATE LOGIN Alex WITH PASSWORD = '1234567890' MUST_CHANGE, CHECK_EXPIRATION=ON;
CREATE LOGIN Nick WITH PASSWORD = '1234567890' MUST_CHANGE, CHECK_EXPIRATION=ON;
CREATE LOGIN Lora WITH PASSWORD = '1234567890' MUST_CHANGE, CHECK_EXPIRATION=ON;
GO

CREATE USER Alex FOR LOGIN Alex;
CREATE USER Nick FOR LOGIN Nick;
CREATE USER Lora FOR LOGIN Lora;
GO
```

The following script creates users in the **Test2** database:

```
USE Test2
GO

CREATE USER Alex FOR LOGIN Alex WITH DEFAULT_SCHEMA=dbo69;
CREATE USER Nick FOR LOGIN Nick WITH DEFAULT_SCHEMA=dbo69;
CREATE USER Lora FOR LOGIN Lora WITH DEFAULT_SCHEMA=dbo69;
GO
```

This is the important code that gives permissions to create tables in a database:

```
GRANT CREATE TABLE ON DATABASE::Test2 TO Alex
GRANT CREATE TABLE ON DATABASE::Test2 TO Nick
GRANT CREATE TABLE ON DATABASE::Test2 TO Lora
GO
```

The following code creates and configures a role for our team, and grants permissions to Alex:

```
CREATE ROLE Alex_Team;
GO

ALTER ROLE Alex_Team ADD MEMBER Nick;
ALTER ROLE Alex_Team ADD MEMBER Lora;
GO

GRANT CONTROL ON ROLE::Alex_Team TO Alex;

GRANT CONTROL ON USER::Nick TO Alex;
GRANT CONTROL ON USER::Lora TO Alex;
GO
```

The following code creates a schema and sets the schema permissions:

```
CREATE SCHEMA dbo69;
GO

GRANT CONTROL ON SCHEMA::dbo69 TO Alex
GO
GRANT SELECT, INSERT, UPDATE, DELETE, EXECUTE, VIEW DEFINITION ON SCHEMA::dbo69 TO
Alex_Team
GO
```

EventHandlers and TableFormats

The following two tables stores the SaveToDB add-in configuration.

You must have these tables in a database to implement certain of features.

Please replace dbo69 in the following code (9 places) to your actual schema:

```
CREATE TABLE dbo69.EventHandlers (
      ID int IDENTITY(1,1) NOT NULL
    , TABLE_SCHEMA nvarchar(128) NOT NULL
    , TABLE_NAME nvarchar(128) NOT NULL
    , COLUMN_NAME nvarchar(128) NULL
    , EVENT_NAME varchar(50) NOT NULL
    , HANDLER_SCHEMA nvarchar(128) NULL
    , HANDLER_NAME nvarchar(128) NULL
    , HANDLER_TYPE nvarchar(128) NULL
    , HANDLER_CODE nvarchar(MAX) NULL
    , TARGET_WORKSHEET nvarchar(128) NULL
    , MENU_ORDER int NULL
    , EDIT_PARAMETERS bit NULL
    , IS_ACTIVE bit NULL DEFAULT((1))
    , CONSTRAINT PK_EventHandlers_dbo69 PRIMARY KEY (ID)
);
GO

CREATE TABLE dbo69.TableFormats (
      ID int IDENTITY(1,1) NOT NULL
    , TABLE_SCHEMA nvarchar(128) NOT NULL
    , TABLE_NAME nvarchar(128) NOT NULL
    , TABLE_EXCEL_FORMAT_XML xml NULL
    , CONSTRAINT PK_TableFormat_dbo69 PRIMARY KEY (ID)
    , CONSTRAINT IX_TableFormats_Schema_Name_dbo69 UNIQUE (TABLE_NAME, TABLE_SCHEMA)
);
GO

INSERT INTO dbo69.EventHandlers (TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME, EVENT_NAME,
HANDLER_SCHEMA, HANDLER_NAME, HANDLER_TYPE, HANDLER_CODE) VALUES ('dbo69',
'EventHandlers', 'EVENT_NAME', 'ValidationList', NULL, NULL, 'VALUES',
'Actions,Change,ContextMenu,DoubleClick,SelectionChange,ConvertFormulas,DoNotConvertForm
ulas,DoNotSelect,DoNotSave,DoNotChange,ProtectRows,Formula,FormulaValue,ValidationList,S
electionList');

INSERT INTO dbo69.EventHandlers (TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME, EVENT_NAME,
HANDLER_SCHEMA, HANDLER_NAME, HANDLER_TYPE, HANDLER_CODE) VALUES ('dbo69',
'EventHandlers', 'HANDLER_TYPE', 'ValidationList', NULL, NULL, 'VALUES',
'TABLE,VIEW,PROCEDURE,FUNCTION,CODE,HTTP,TEXT,MACRO,CMD,VALUES,RANGE,REFRESH,MENUSEPARAT
OR');
GO
```

Chapter 3. Publish Wizard

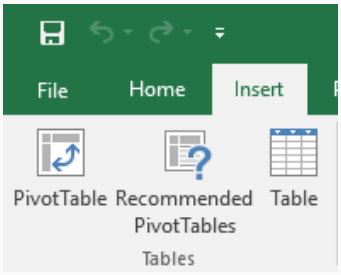
In this chapter, we will publish our table to a database.

This action is required once for every table that you want to have in a database.

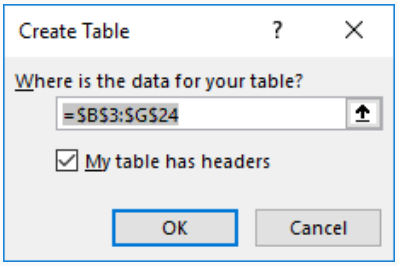
Excel Tables

The tables must be an “Excel tables”, not regular ranges.

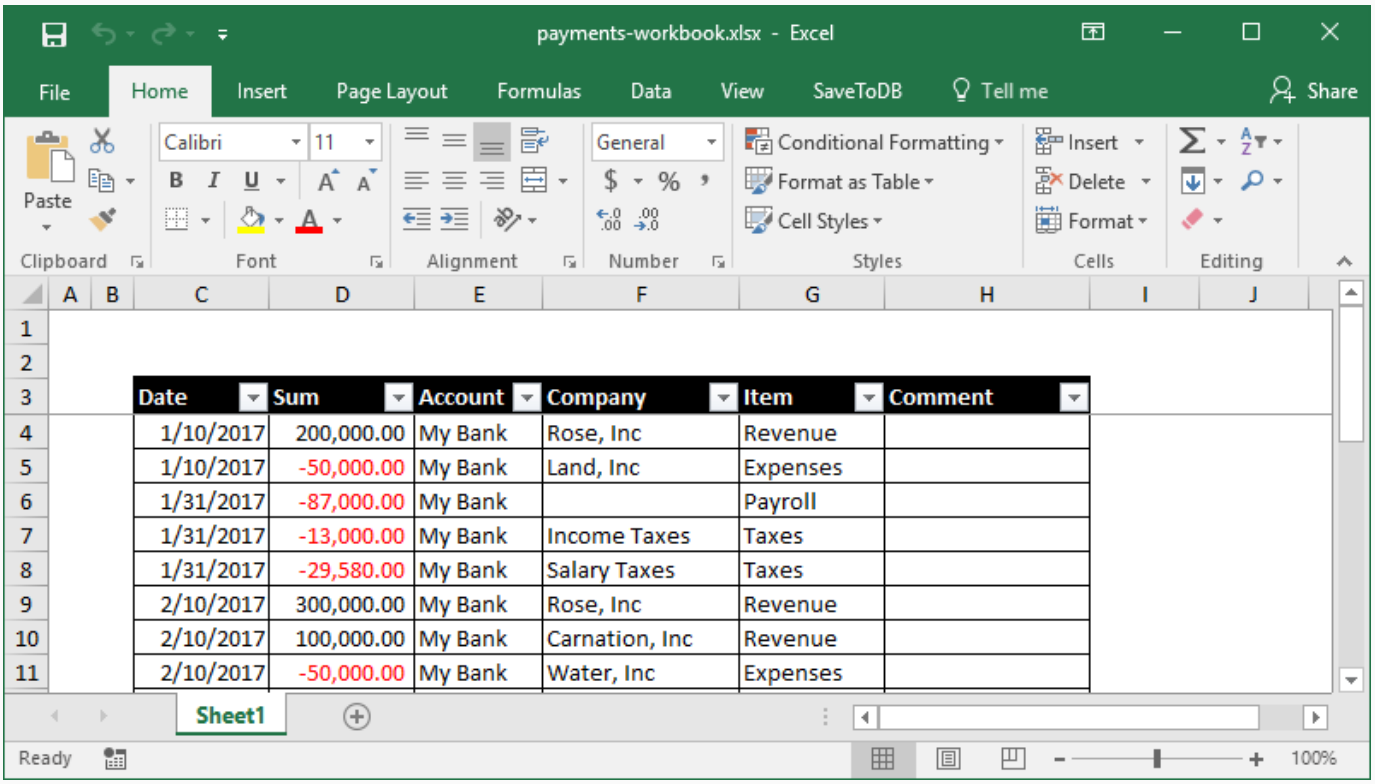
To convert a range table to an Excel table, select a cell in a table, and click **Insert**, **Table**.



Excel suggests an entire region. Correct it if needed.
Check **My table has headers** and click **OK**.



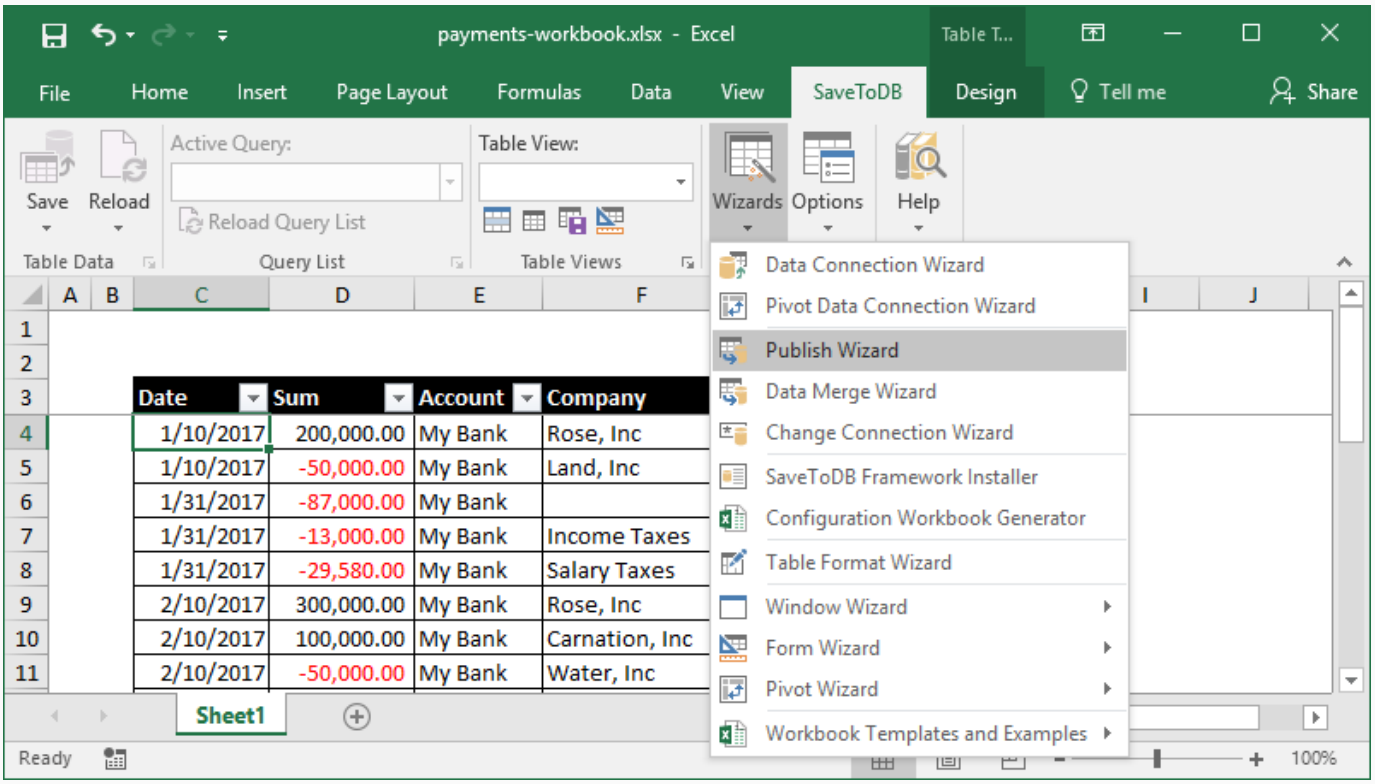
Then locate the table left top cell at cell C3 (Chapter 6 contains details). Now we have the table like this:



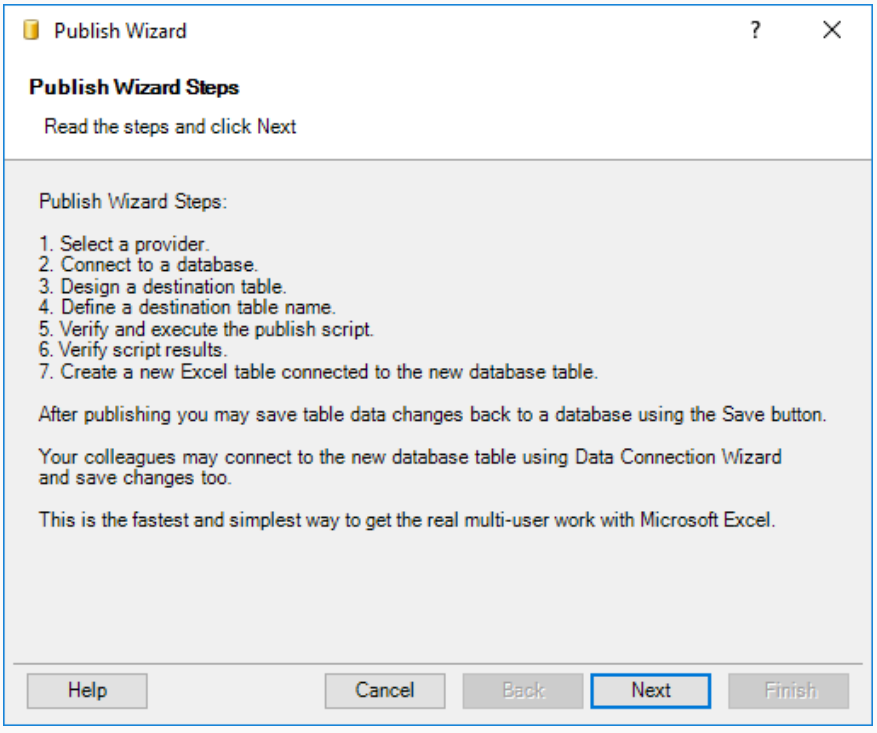
Date	Sum	Account	Company	Item	Comment
1/10/2017	200,000.00	My Bank	Rose, Inc	Revenue	
1/10/2017	-50,000.00	My Bank	Land, Inc	Expenses	
1/31/2017	-87,000.00	My Bank		Payroll	
1/31/2017	-13,000.00	My Bank	Income Taxes	Taxes	
1/31/2017	-29,580.00	My Bank	Salary Taxes	Taxes	
2/10/2017	300,000.00	My Bank	Rose, Inc	Revenue	
2/10/2017	100,000.00	My Bank	Carnation, Inc	Revenue	
2/10/2017	-50,000.00	My Bank	Water, Inc	Expenses	

Starting Publish Wizard

Now select a cell in the table and run **Publish Wizard**.



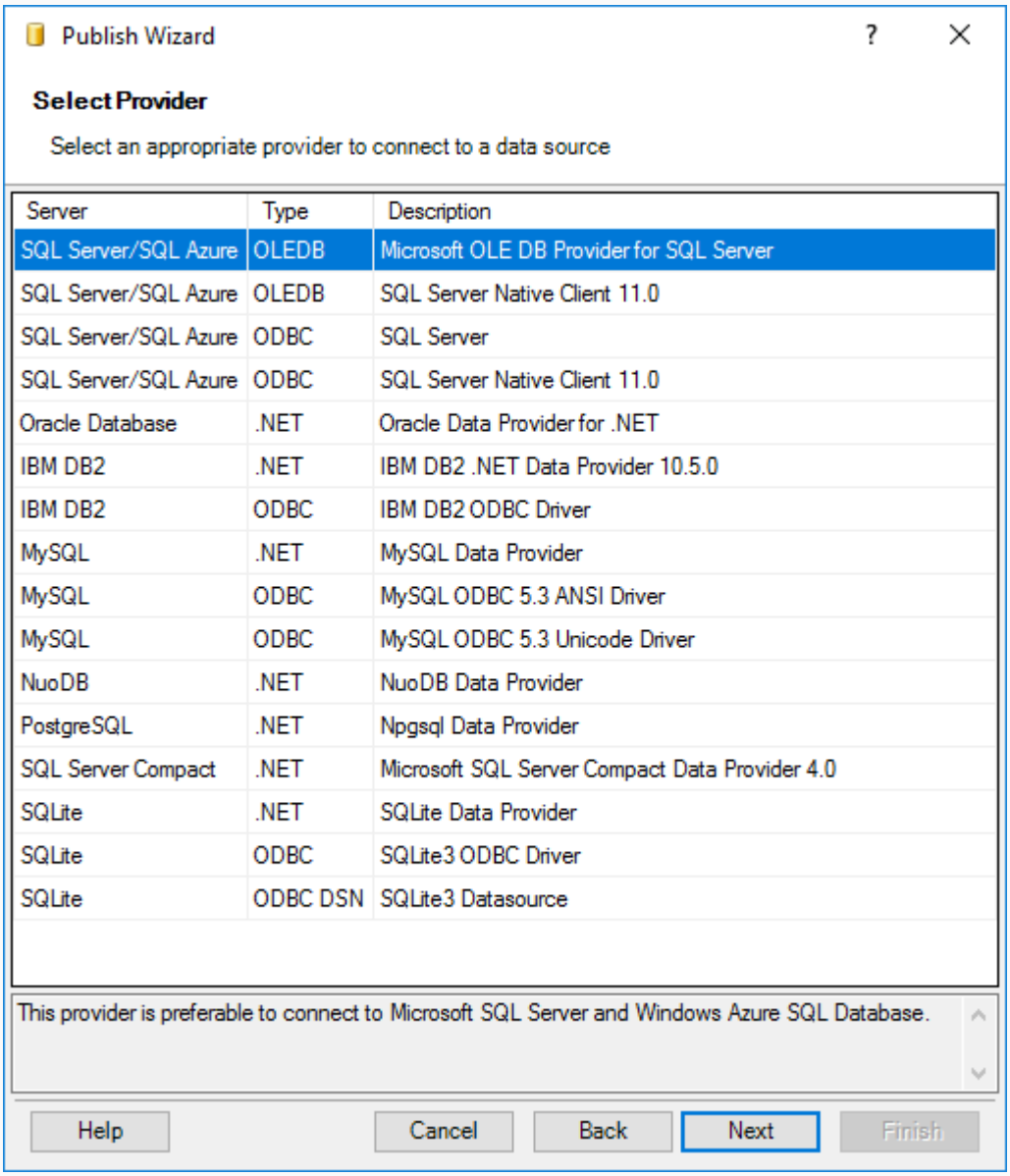
You see wizard step descriptions:



Read and click **Next**.

Selecting Provider

In the next step, we select a database provider:

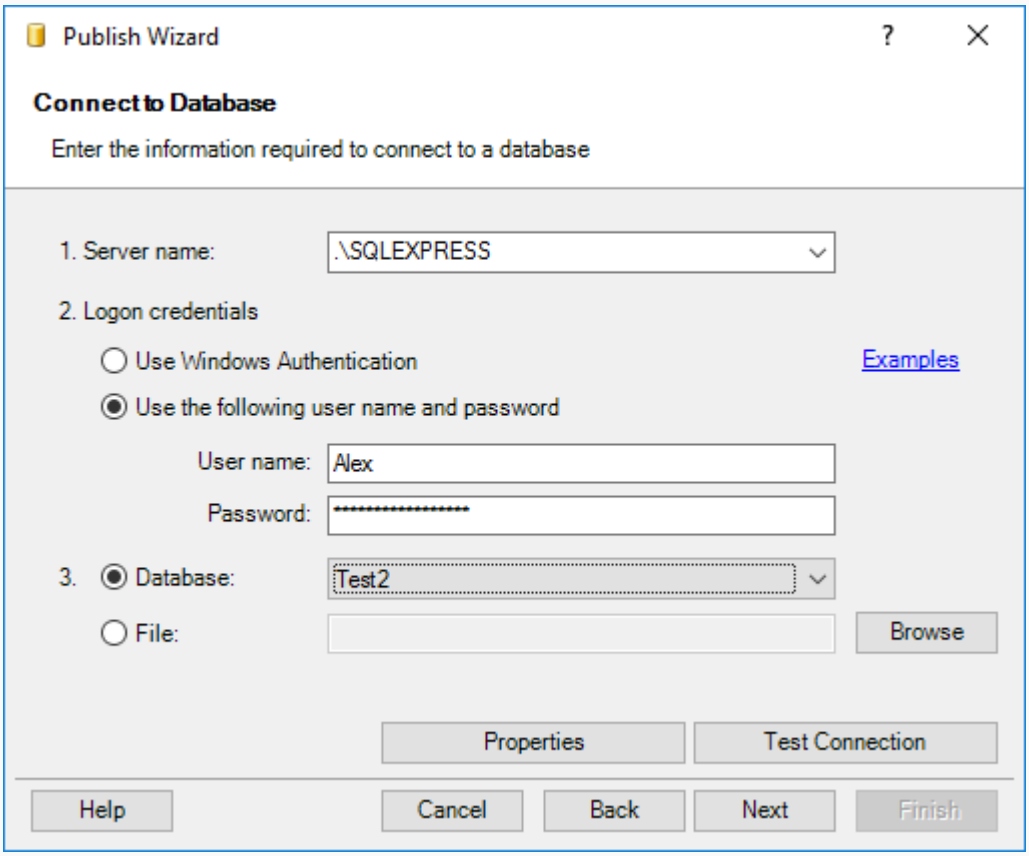


You may publish tables to any supported database like SQL Server, Oracle, MySQL, and others.

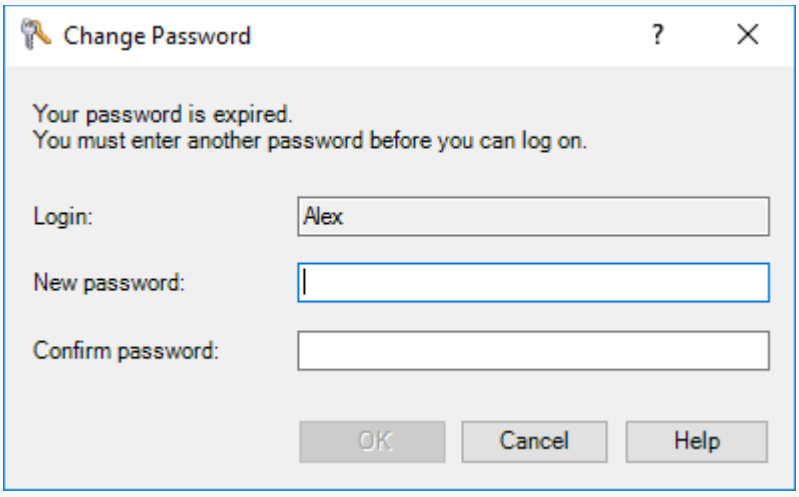
Select the first provider for SQL Server.

Connecting to Database

In the next step, you have to specify a server, a database, and logon credentials received from your IT guy.



Usually, at the first connection, you have to change the password:

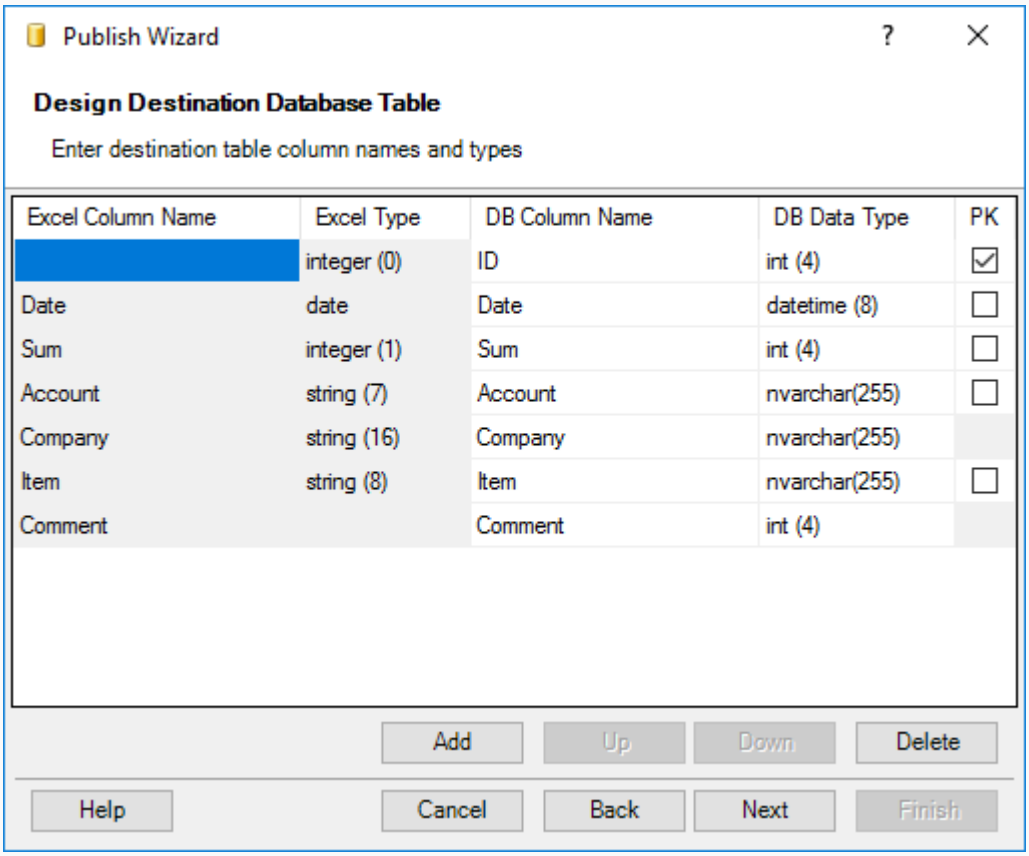


Fill the new password twice and click **OK**.

Then click **Next** in the connection form.

Table Design

The add-in suggests a database table structure based on actual data in the published table.



In this example, we need to change the int type of the Sum column to float, and the int type of the Comment column to nvarchar(255) as shown on the next page.

The add-in suggests the correct types for target databases and actual column types in the drop-down list.

Also, tables must have the primary key column or columns that uniquely identify table rows.

In most cases, you may add a column like ID of the integer type.

The add-in does this automatically if it does not find a column with unique values.

You may define primary key columns checking the **PK** column.

Also, you may add, delete, rename, and reorder columns.

It is a good idea to have column names without spaces. You may use capitalized word parts like CompanyName.

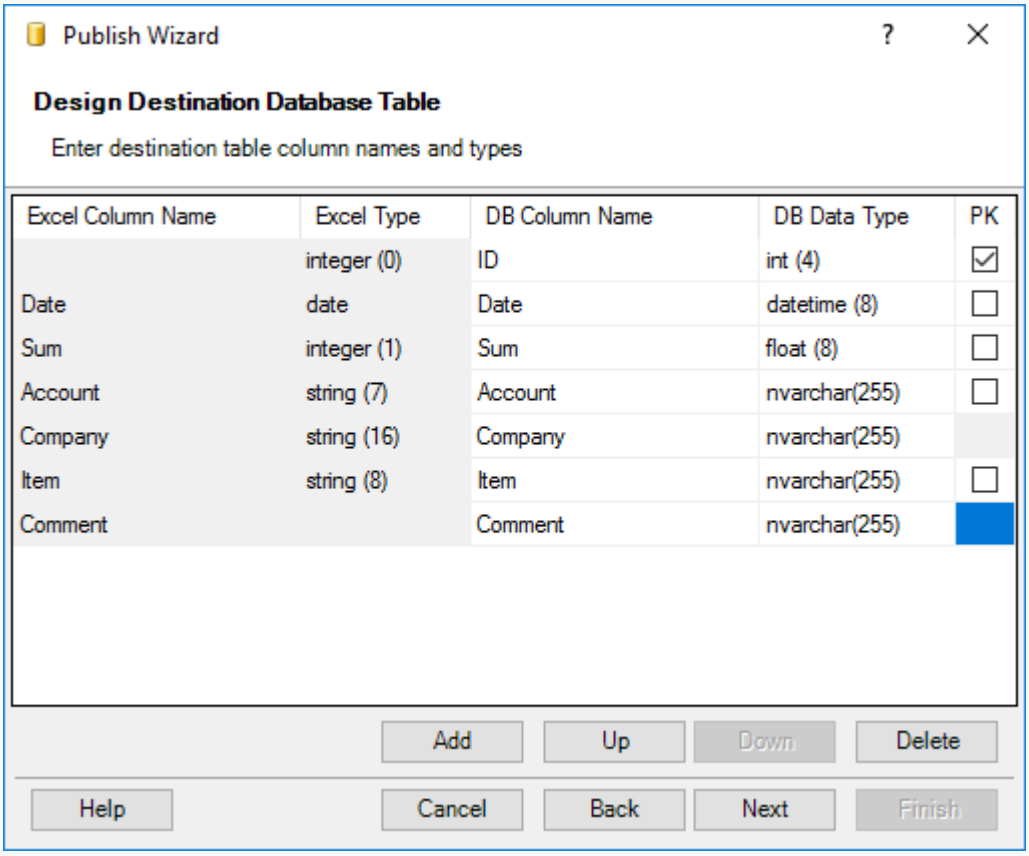
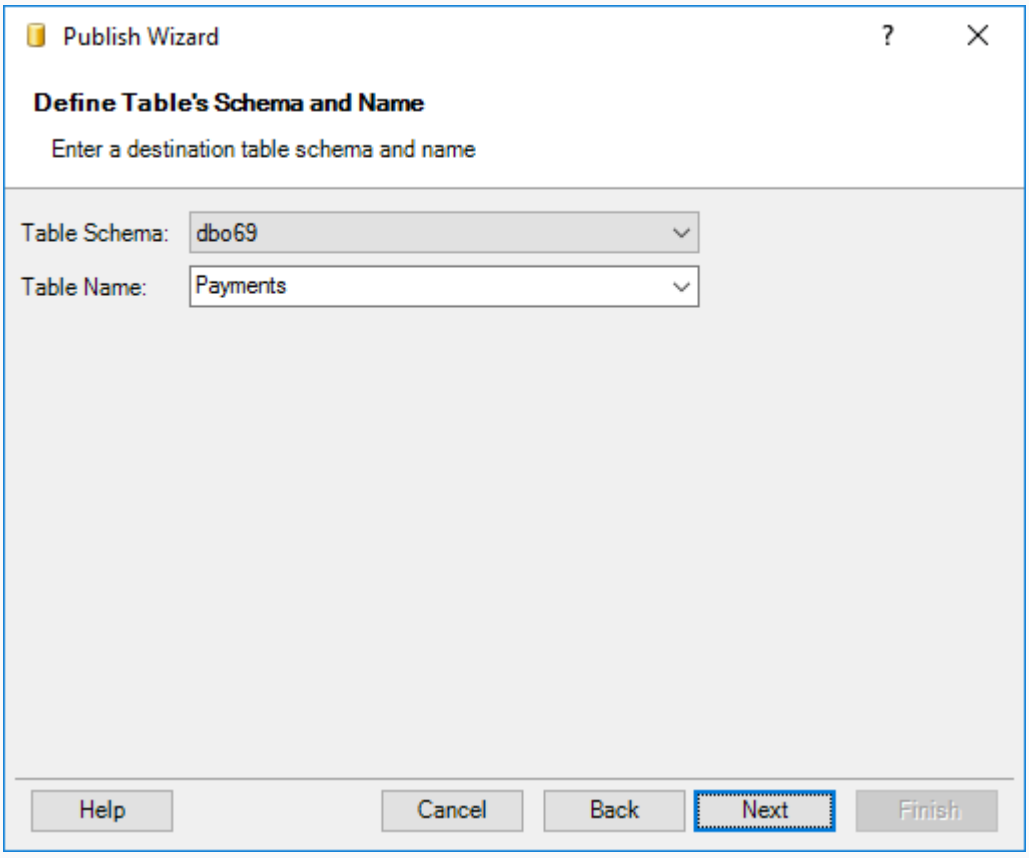


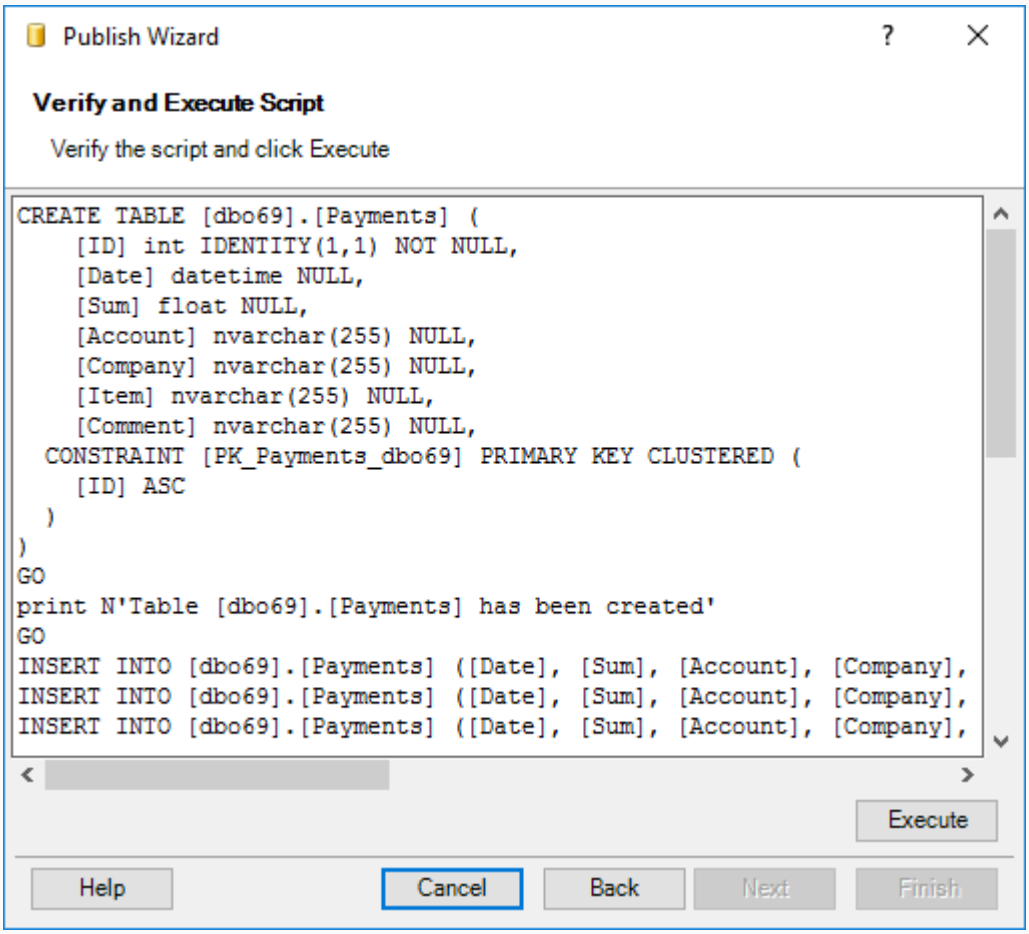
Table Schema and Name

In this step, you specify a table schema and name:



Executing Script

The add-in generates an SQL script to create a table and to insert the existing data:



You may edit the script if you need.

For example, you may delete INSERT rows if you need to create an empty table.

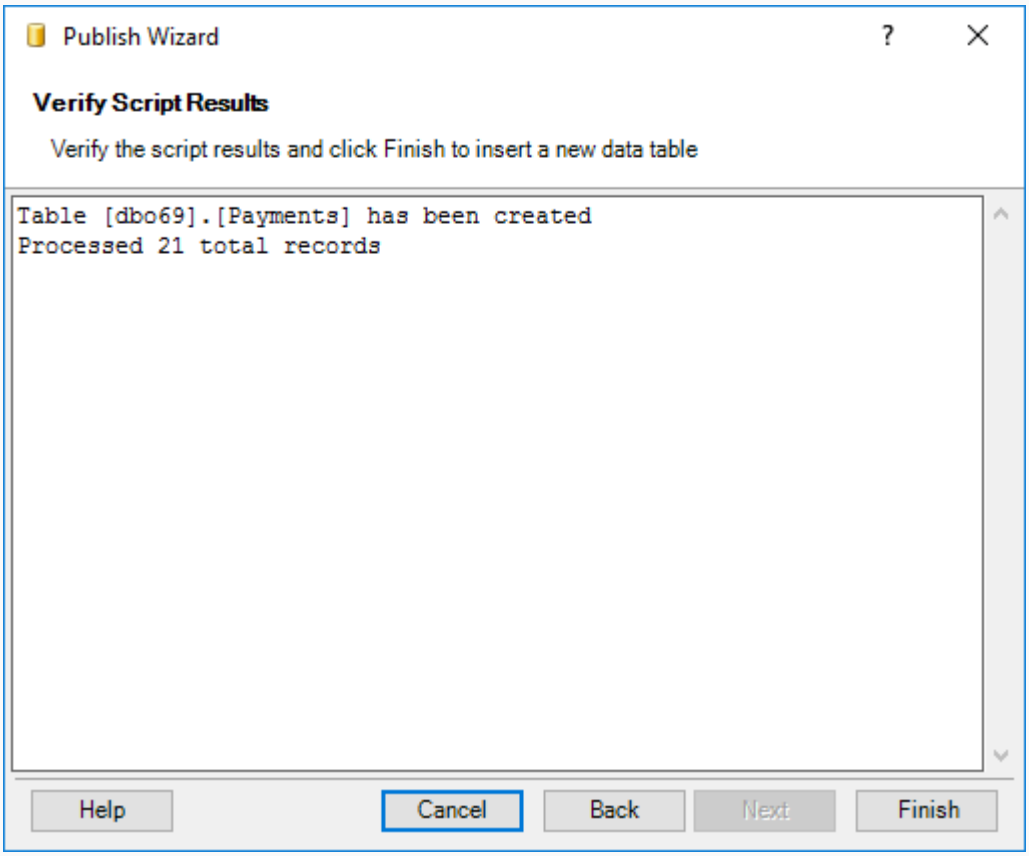
Also, you may create multiple tables changing table names only.

You may change column data types if you do not find it in the drop-down list in the previous step.

Click **Execute** when you are ready.

Finish Steps

The add-in shows execution results on the next screen.

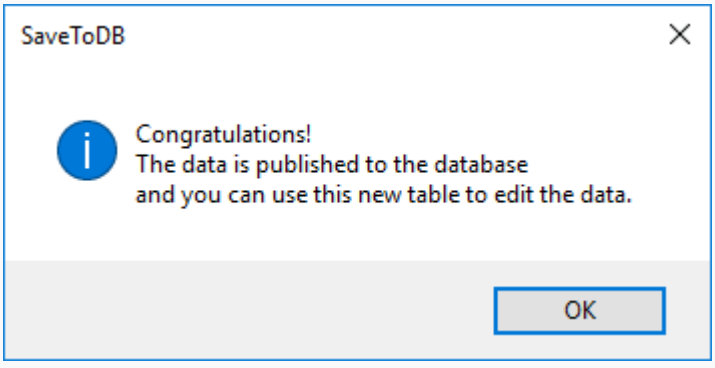


You may repeat steps if you need. Just click **Back**.

You may click **Cancel** if you do not need to insert a new connected table into the workbook.

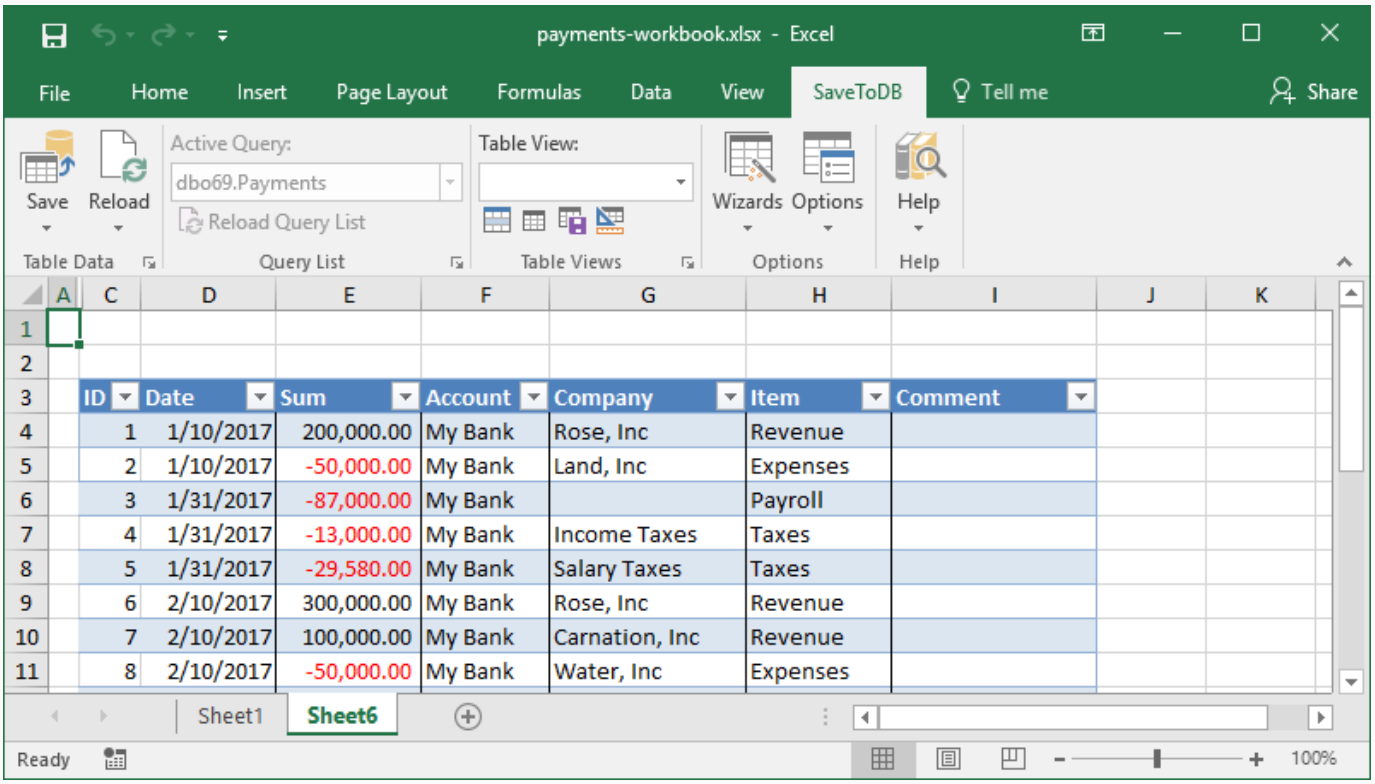
Now click **Finish**.

The add-in inserts a connected table to a new worksheet and shows a message about your success:



Editable Table

We see the following screen as a **Publish Wizard** result:



ID	Date	Sum	Account	Company	Item	Comment
1	1/10/2017	200,000.00	My Bank	Rose, Inc	Revenue	
2	1/10/2017	-50,000.00	My Bank	Land, Inc	Expenses	
3	1/31/2017	-87,000.00	My Bank		Payroll	
4	1/31/2017	-13,000.00	My Bank	Income Taxes	Taxes	
5	1/31/2017	-29,580.00	My Bank	Salary Taxes	Taxes	
6	2/10/2017	300,000.00	My Bank	Rose, Inc	Revenue	
7	2/10/2017	100,000.00	My Bank	Carnation, Inc	Revenue	
8	2/10/2017	-50,000.00	My Bank	Water, Inc	Expenses	

First of all, you see that the **Save** button is enabled.

You may change data, add and delete rows. When you are ready to save changes to a database, just click **Save**.

If you do not want to save changes, just click the **Reload** button.

Note that you may use the **Undo** command (Ctrl-Z). The add-in does not disable it, unlike macros.

You may save and close the workbook, and then open it and save the data changes to a database later.

This is useful when you work outside of the corporate network.

You see that the add-in inserts a new table on a new worksheet named as Sheet6.

You should know that the add-in creates “very hidden” worksheets that contain configuration data.

You may unhide the add-in sheets using **SaveToDB Options, Developer Options, Show SaveToDB Data Sheets**.

You may see the connected table name in the **Active Query** field.

In our example, let’s do the following: rename the Sheet6 to **payments**, remove the source Sheet1, format the table, and save the workbook as a new workbook as **payments.xlsx**.

As a result, we have a new workbook with a table connected to a database.

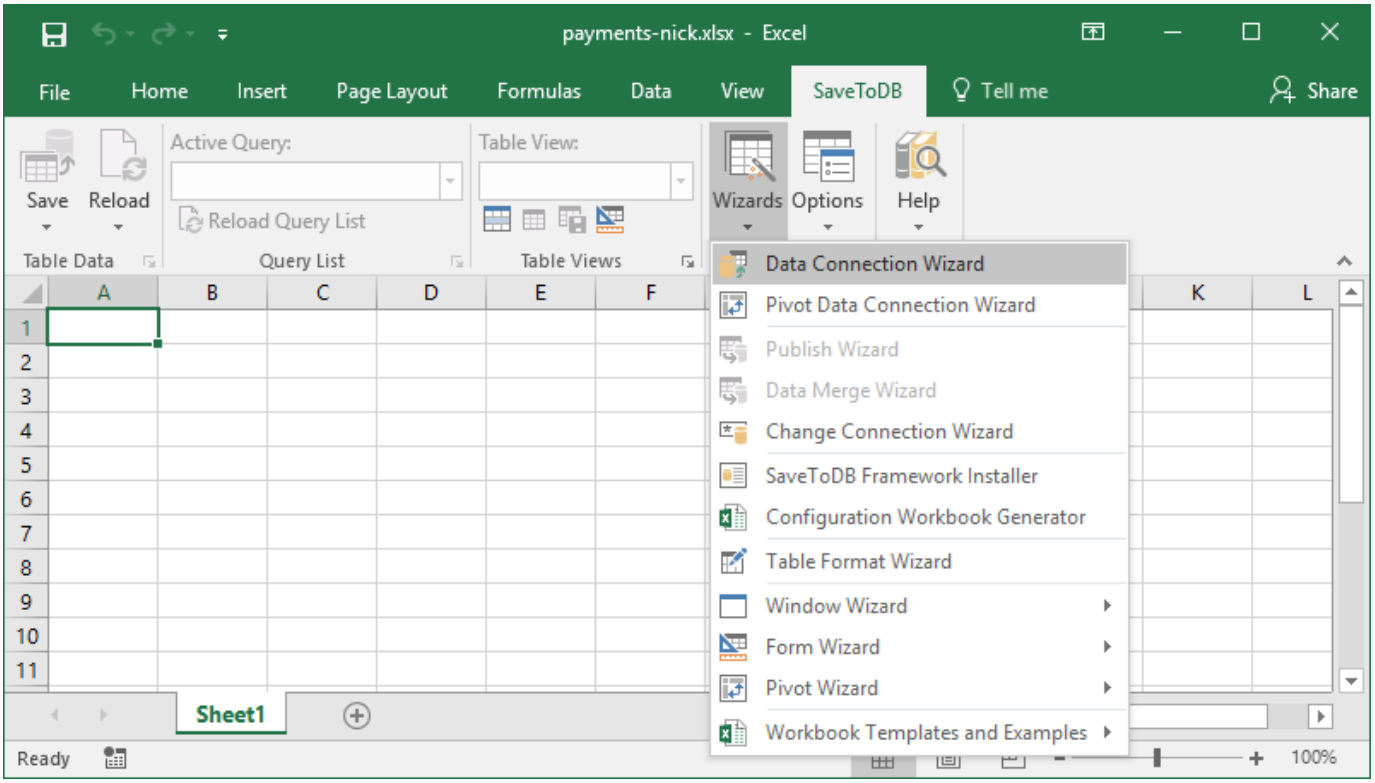
Chapter 4. Data Connection Wizard

In this chapter, we connect to the created **Payments** table from a new workbook under Nick's credentials.

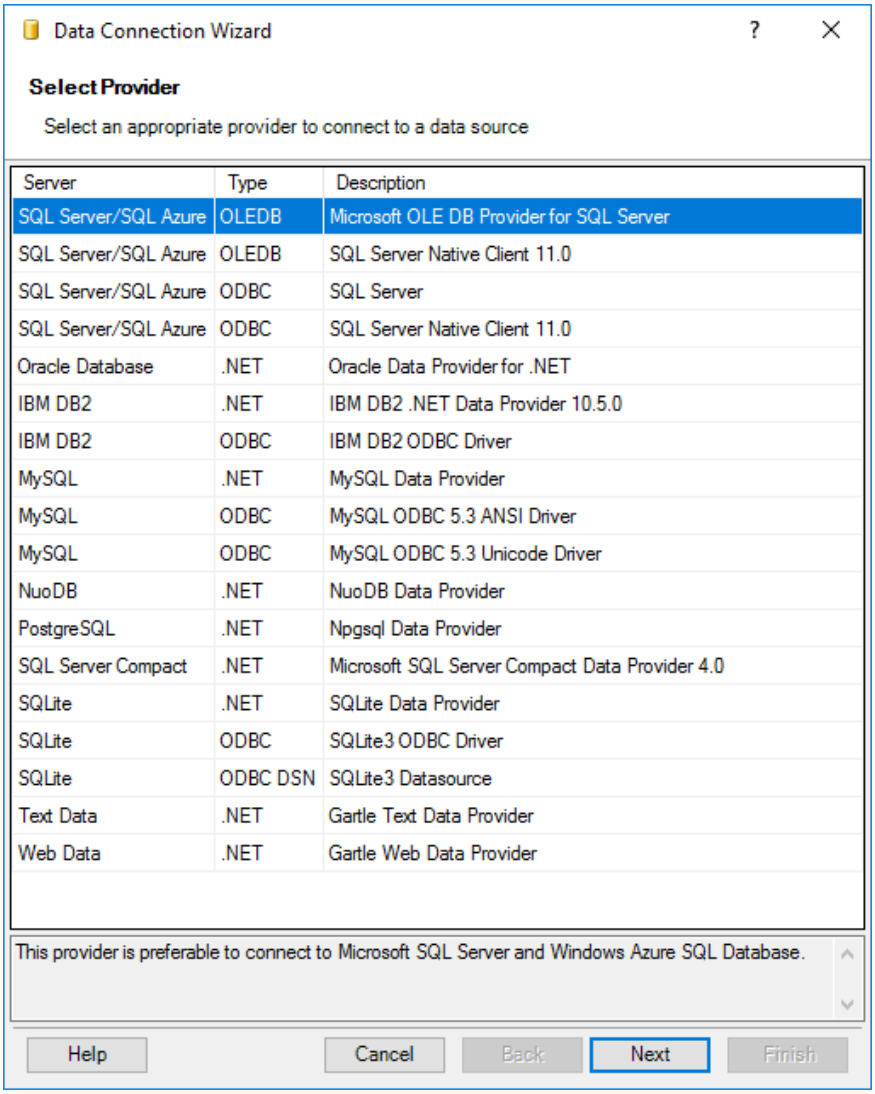
Connecting to Database

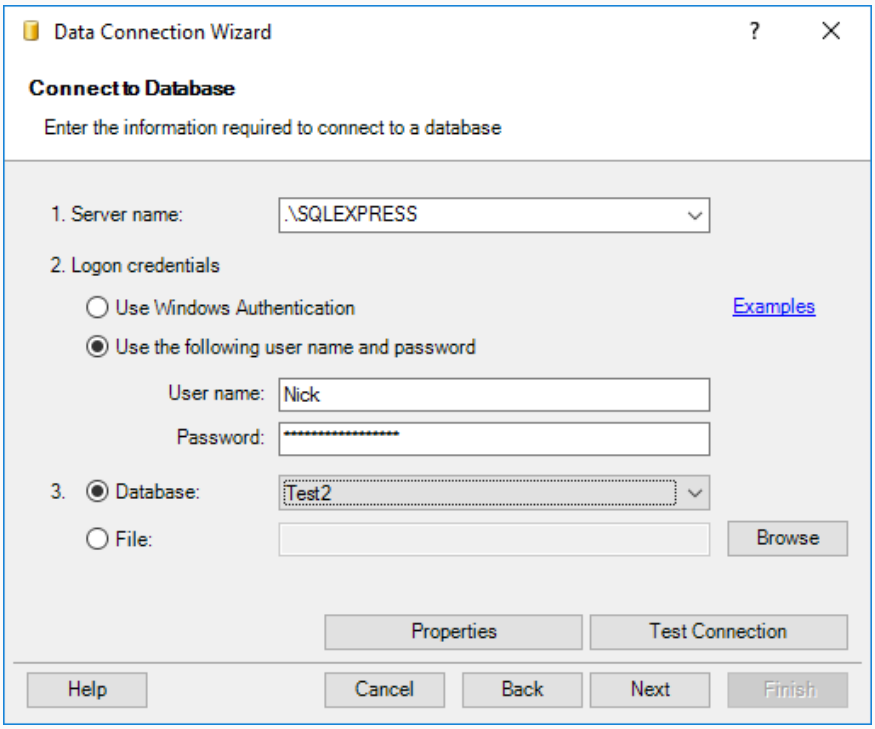
Let's create a new workbook and save it as **payments-nick.xlsx**.

Then let's run the **Data Connect Wizard**.



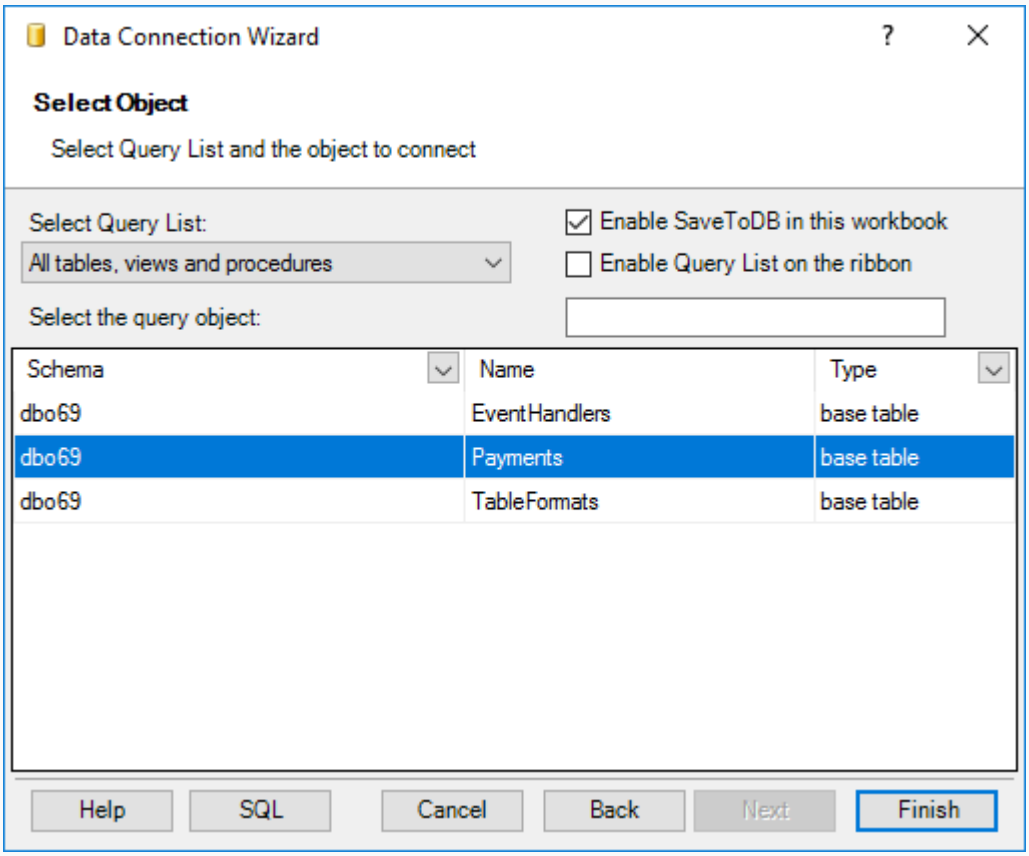
The wizard contains the same connection steps as described above.

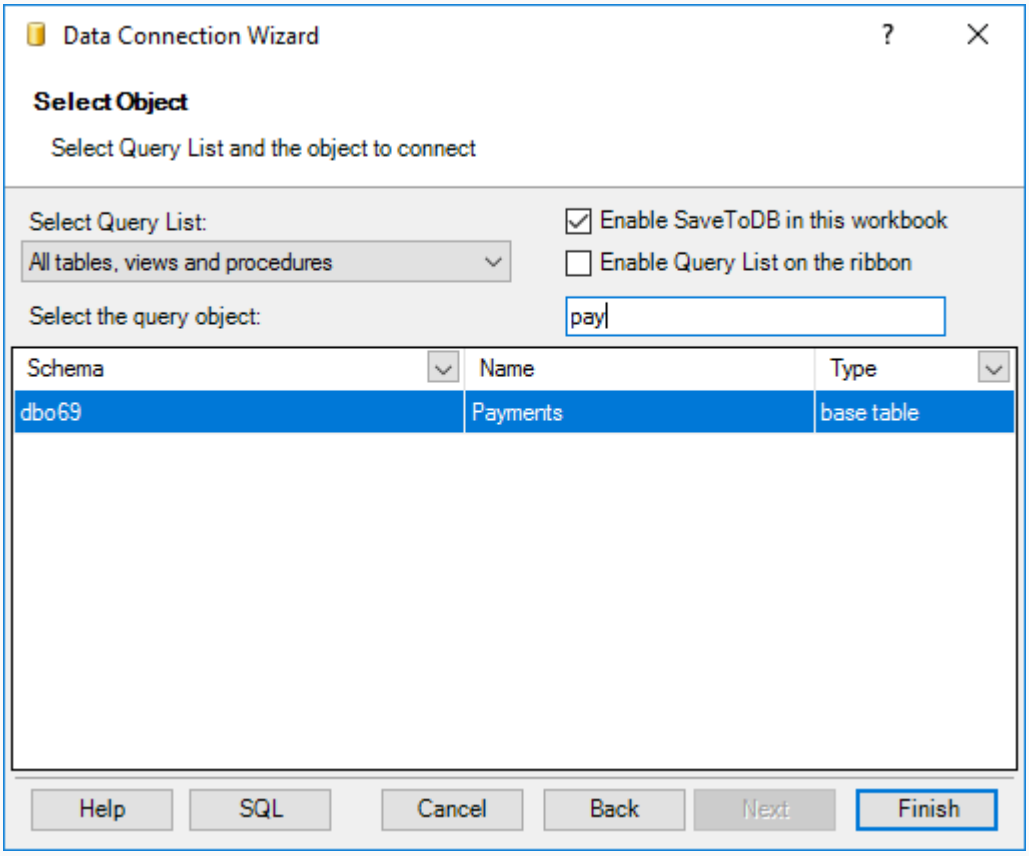




Selecting Database Object

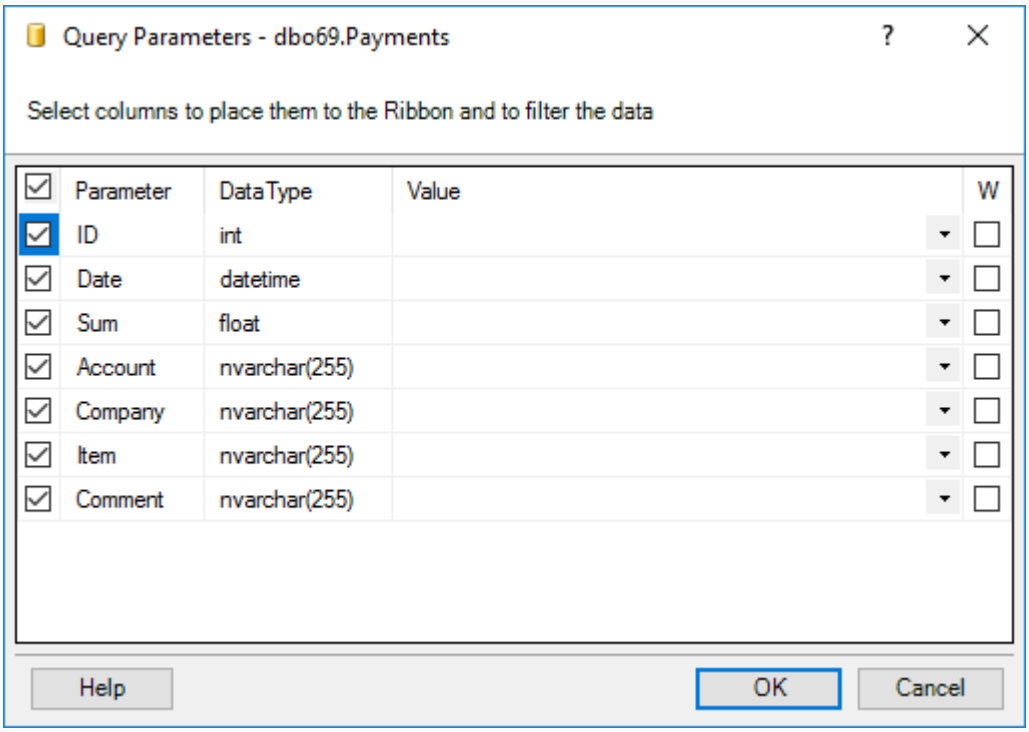
Let's connect to the **dbo69.Payments** table. You may use search. Uncheck **Enable Query List on the ribbon**.



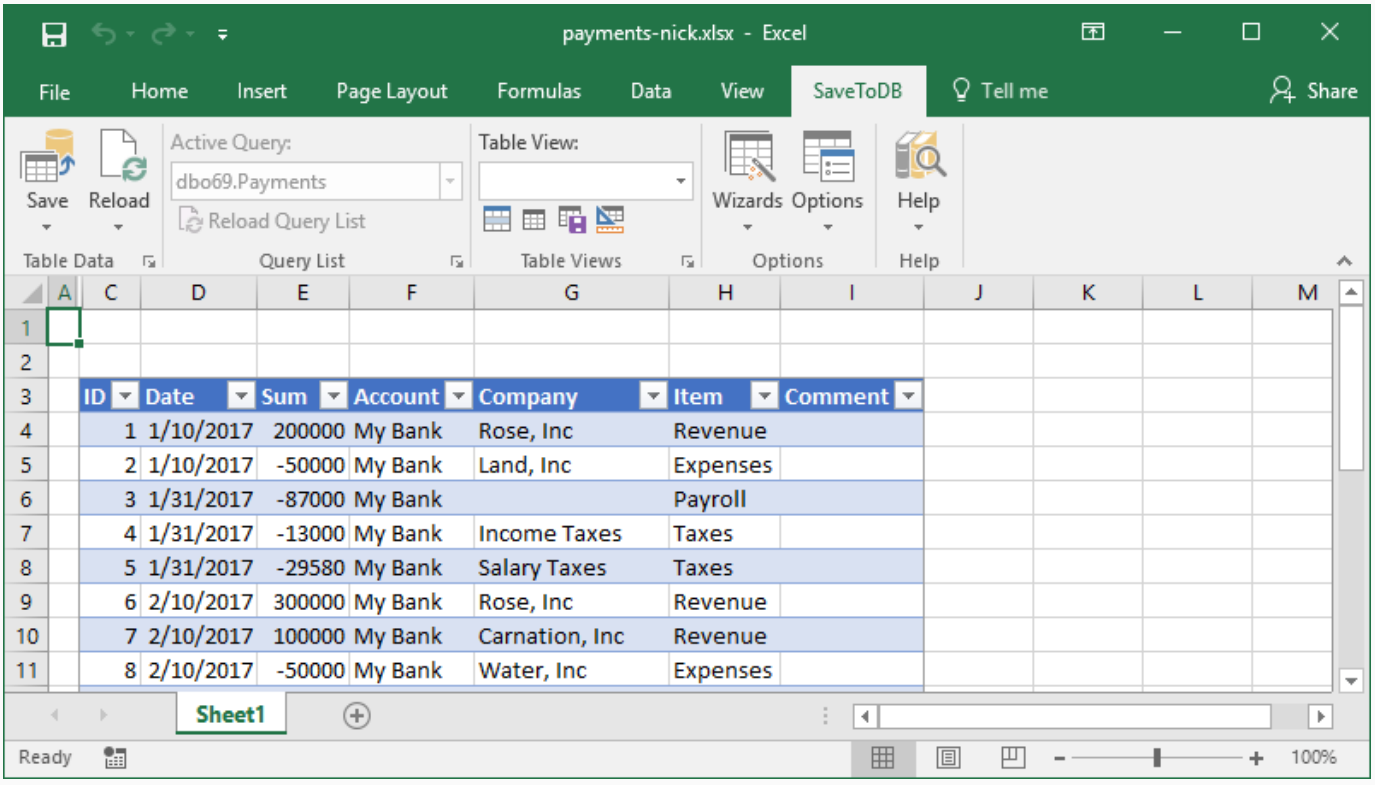


Selecting Table Fields

In this step, you may check the fields to select and the fields to use as filters. We will learn this in the next chapter.



Now you may click **OK** and insert the table at cell B3.

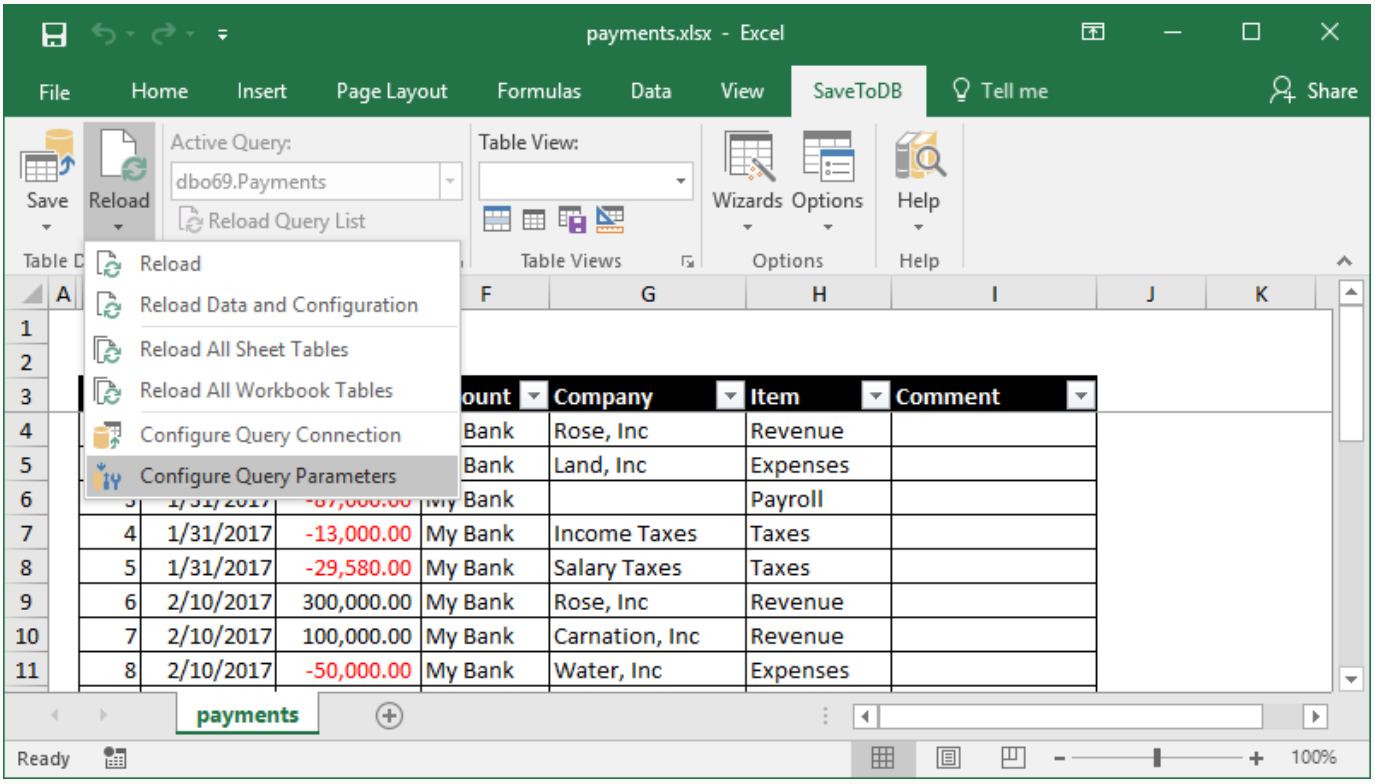


We have an editable table in Nick's workbook. Alex and Nick can work now in their personal workbooks.

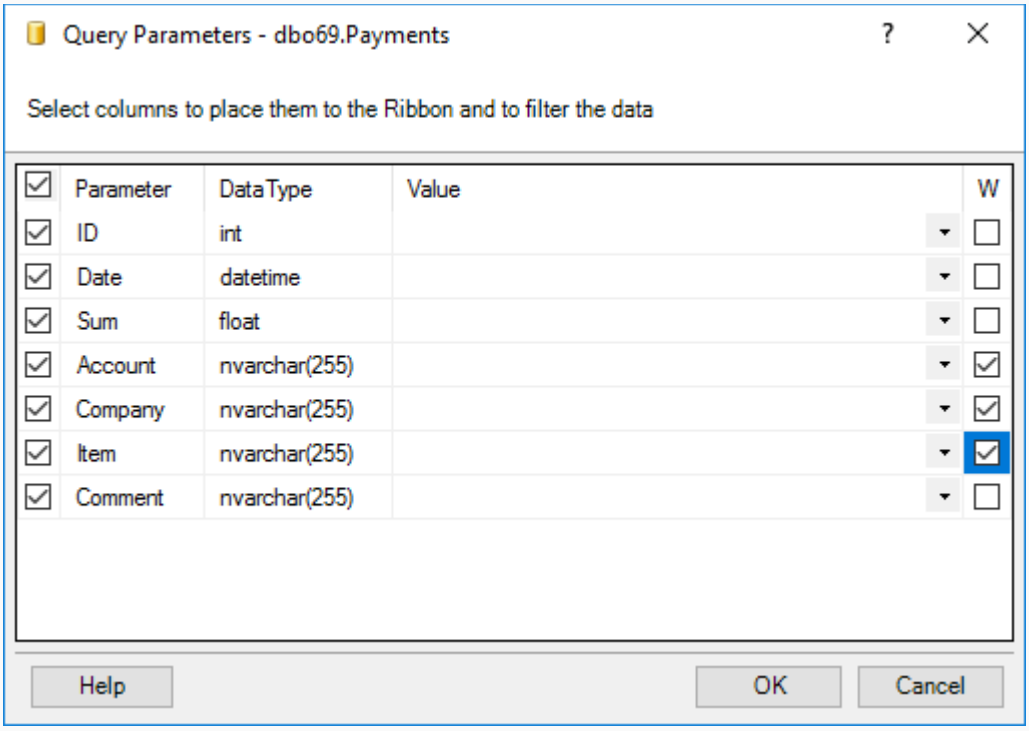
The table has an ugly format, unlike Alex's table. We will solve this in Chapter 7.

Chapter 5. Query Parameters

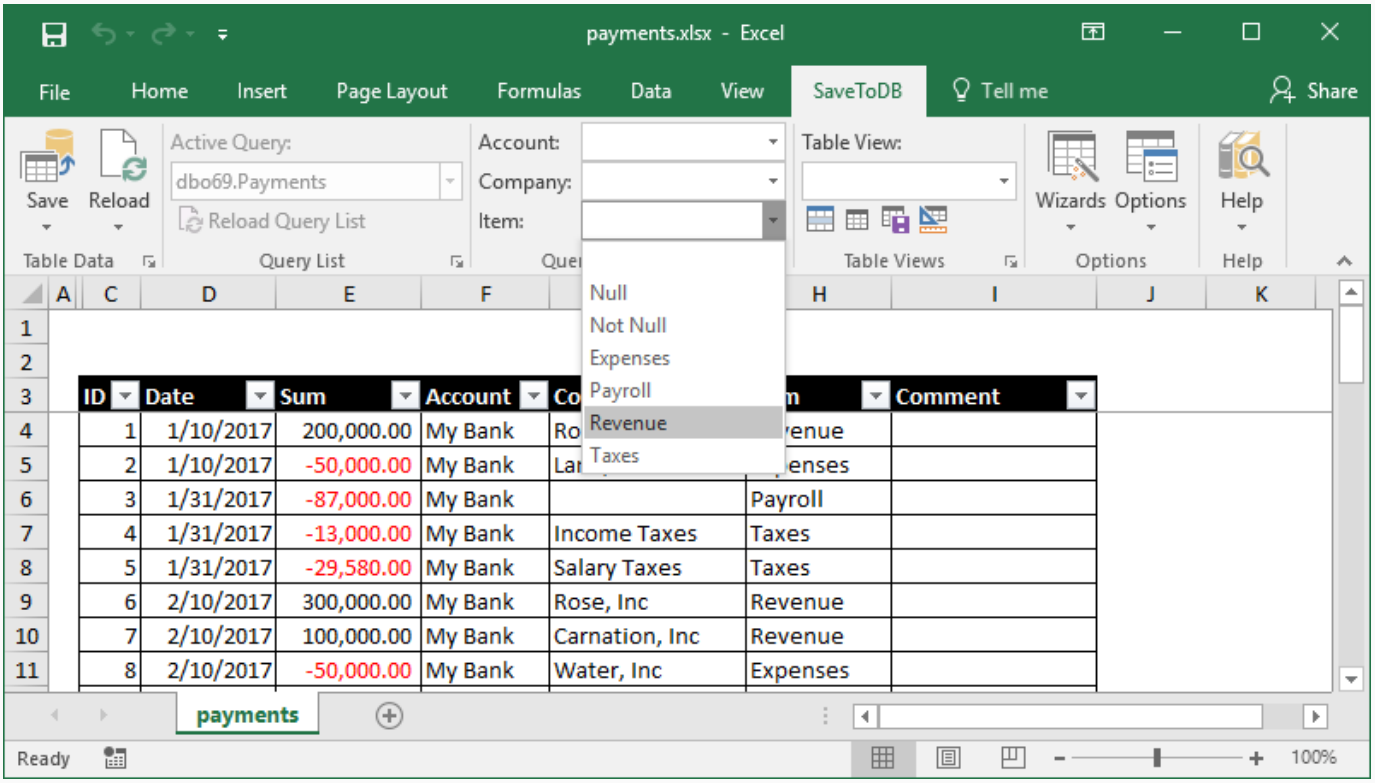
Let's return to Alex's workbook and run **Reload**, **Configure Query Parameters**:



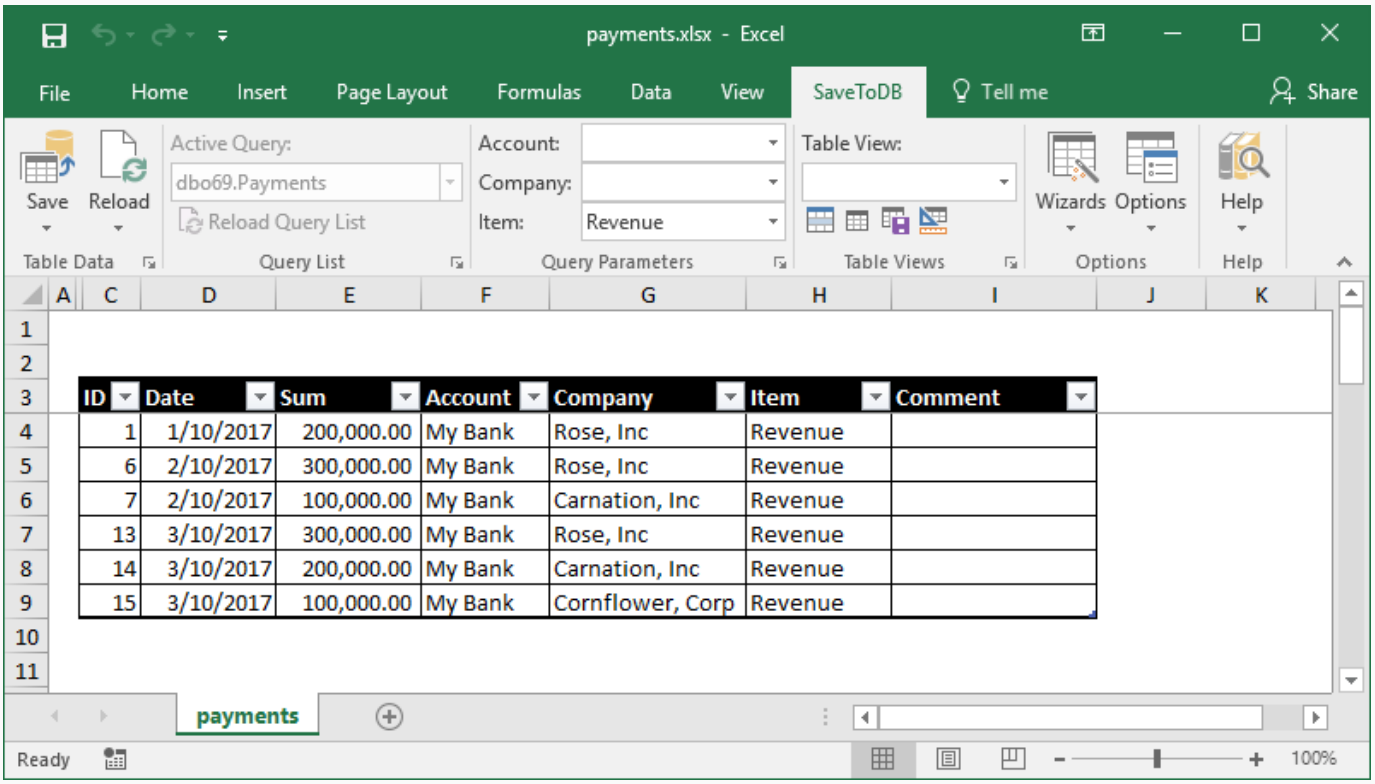
Check the check boxes in the **W** (WHERE) column for the Account, Company, and Item columns:



The SaveToDB add-in places the selected columns to the ribbon. So, you may filter data:



Let’s choose the Revenue item:



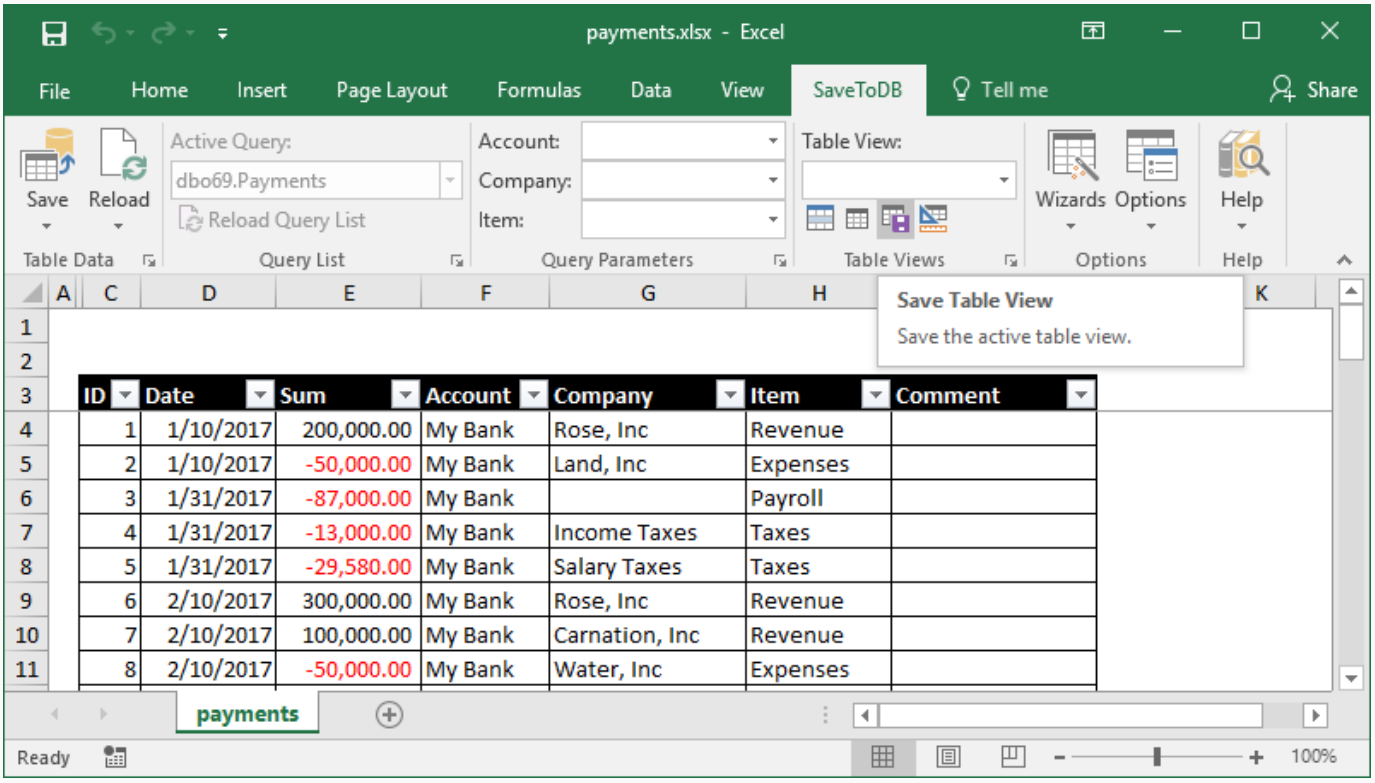
This feature allows working without auto-filters and loading fewer data.

Chapter 6. Table Views

Users often apply different filters to the loaded data, hide and unhide columns, sort in various ways, etc.

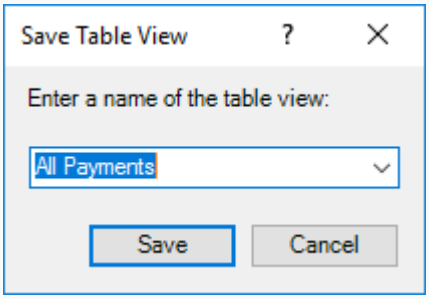
The SaveToDB add-in may help to save such user views and even share them with colleagues.

Let's remove all WHERE filters and click the **Save Table View** button in the **Table Views** group:



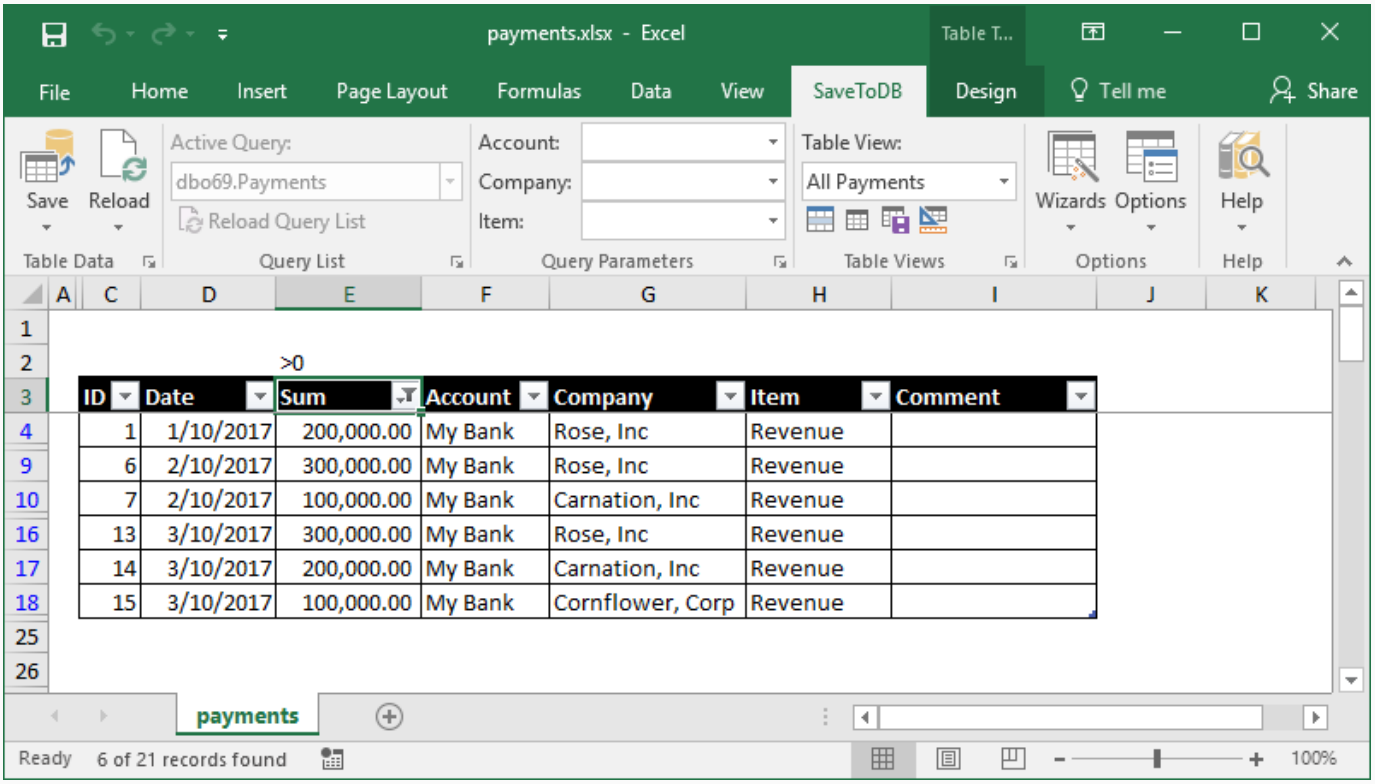
ID	Date	Sum	Account	Company	Item	Comment
1	1/10/2017	200,000.00	My Bank	Rose, Inc	Revenue	
2	1/10/2017	-50,000.00	My Bank	Land, Inc	Expenses	
3	1/31/2017	-87,000.00	My Bank		Payroll	
4	1/31/2017	-13,000.00	My Bank	Income Taxes	Taxes	
5	1/31/2017	-29,580.00	My Bank	Salary Taxes	Taxes	
6	2/10/2017	300,000.00	My Bank	Rose, Inc	Revenue	
7	2/10/2017	100,000.00	My Bank	Carnation, Inc	Revenue	
8	2/10/2017	-50,000.00	My Bank	Water, Inc	Expenses	

Type **All Payments** and click **Save**.



We see the name of the current view, All Payments, in the **Table View** field.

Type **>0** in cell E2.

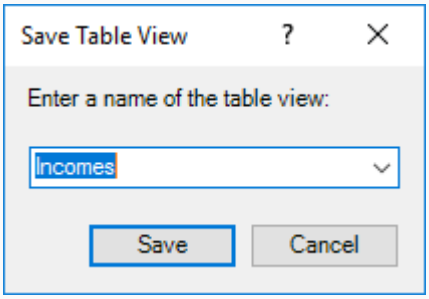


The add-in applies the filter to the Sum column.

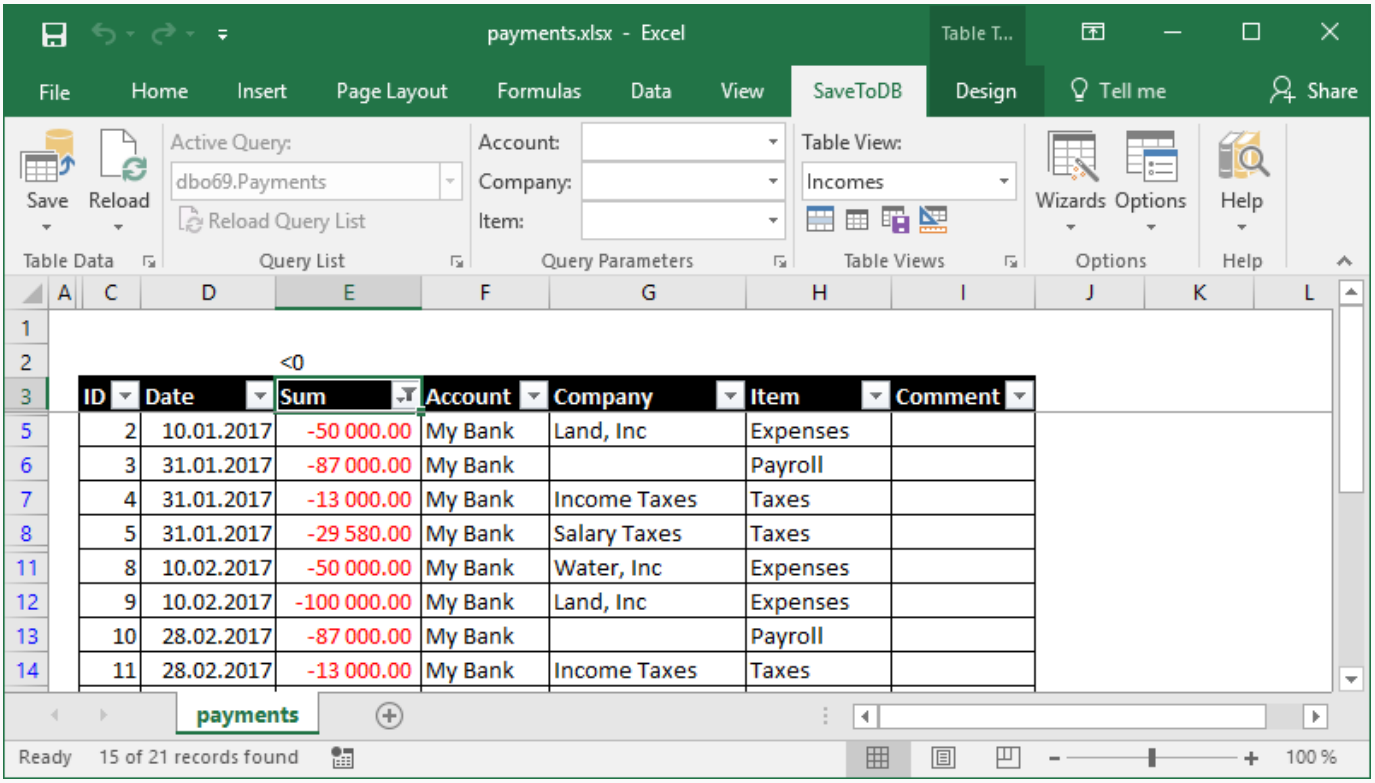
This is a reason why it is better to insert tables at cell B3.

Users may use row 2 (as a row over the table) as auto-filters. Also, they may place formulas in row 1.

Let's continue and save the view as **Incomes** (click the **Save Table View** button again):

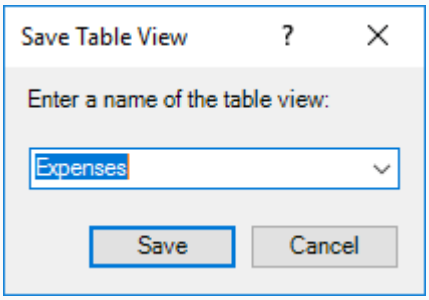


Type **<0** in cell E2.

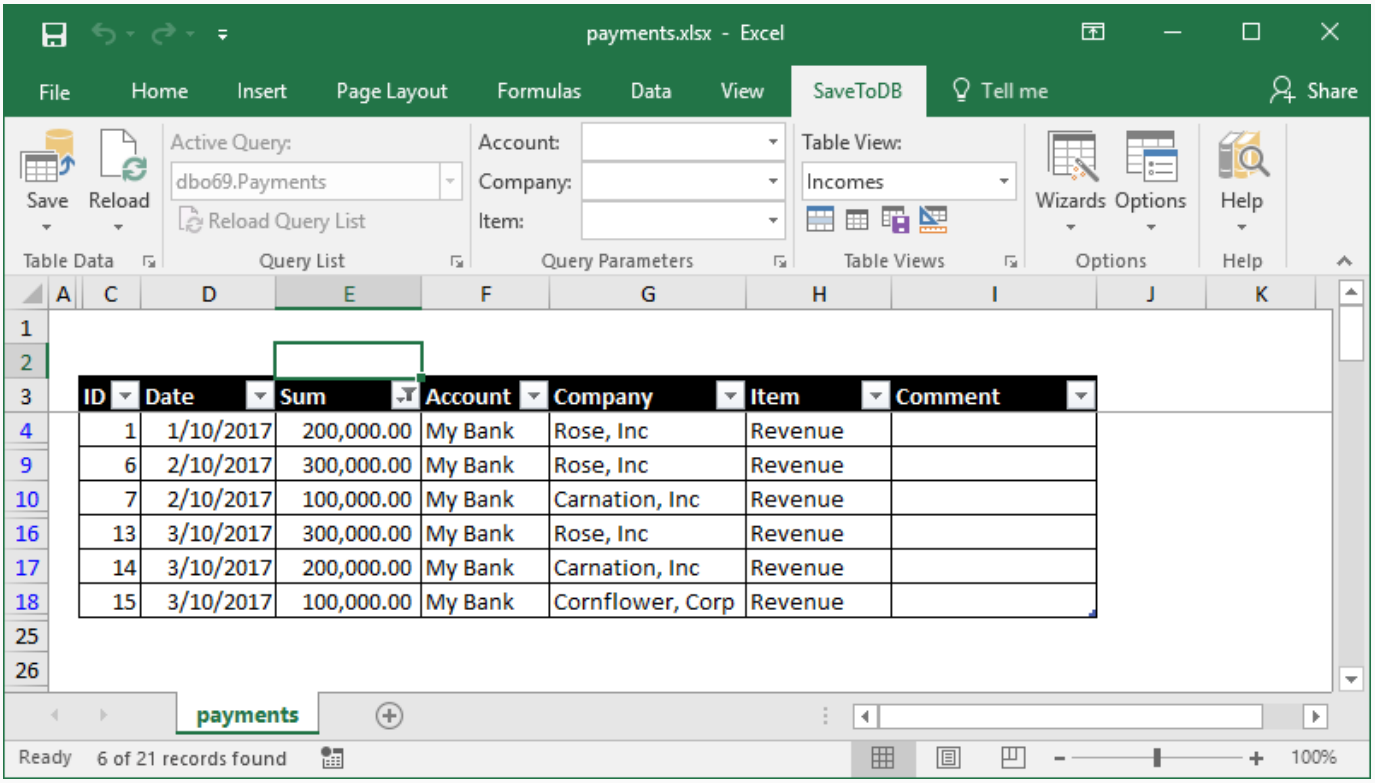


The add-in applies the new filter to the Sum column.

Save the view as **Expenses**.



Remove the filter in cell E2 and apply the **Incomes** view in the **Table View** list:



ID	Date	Sum	Account	Company	Item	Comment
1	1/10/2017	200,000.00	My Bank	Rose, Inc	Revenue	
6	2/10/2017	300,000.00	My Bank	Rose, Inc	Revenue	
7	2/10/2017	100,000.00	My Bank	Carnation, Inc	Revenue	
13	3/10/2017	300,000.00	My Bank	Rose, Inc	Revenue	
14	3/10/2017	200,000.00	My Bank	Carnation, Inc	Revenue	
15	3/10/2017	100,000.00	My Bank	Cornflower, Corp	Revenue	

As we may expect, the add-in applies the saved filter to the Sum column.

You may use the auto-filter row and table views for any Excel table.

Moreover, you may save views for pivot tables also.

I am sure you will like these features.

Chapter 7. Table Format Wizard

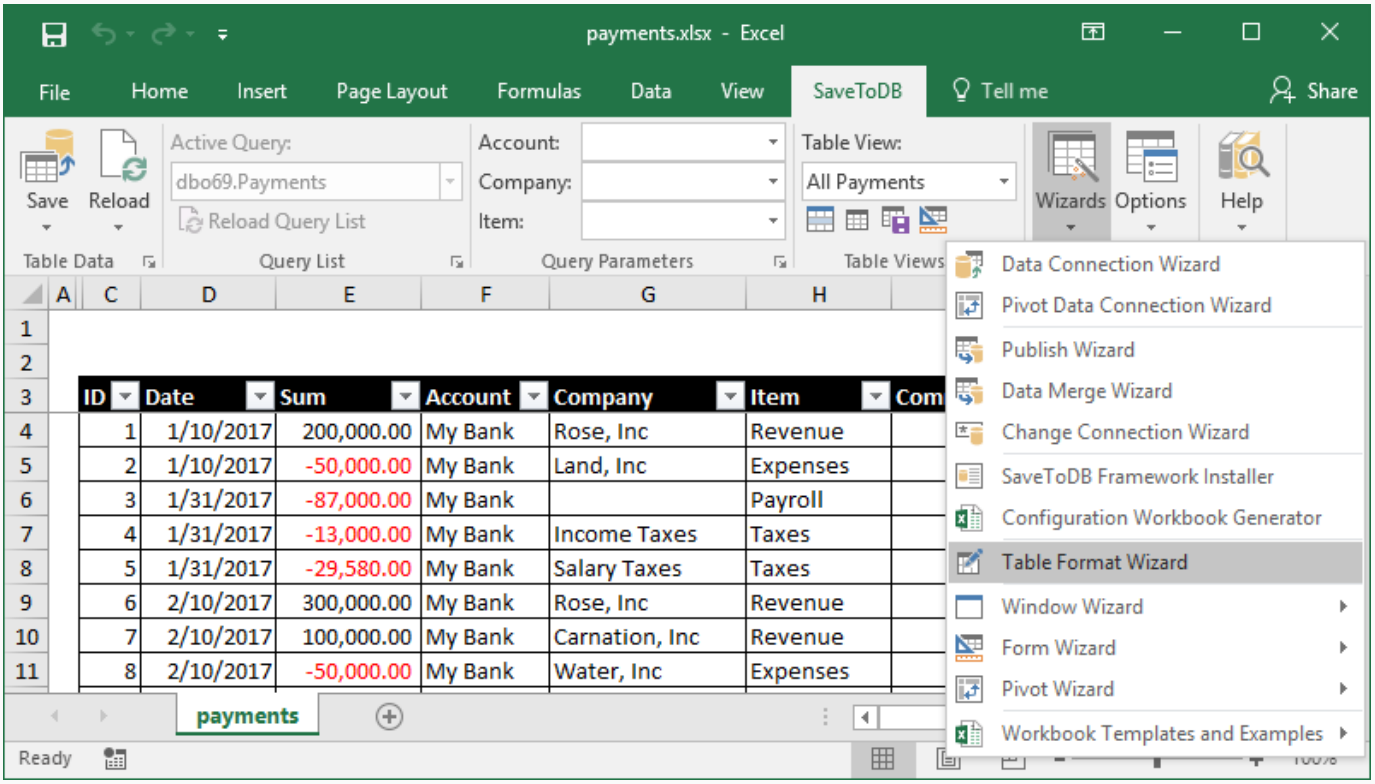
We have formatted the table in the previous steps in the payments.xlsx workbook.

If a user connects to a database from a new workbook like Nick in Chapter 4, he has Excel defaults.

We can fix this publishing table formats and table views to a database using **Table Format Wizard**.

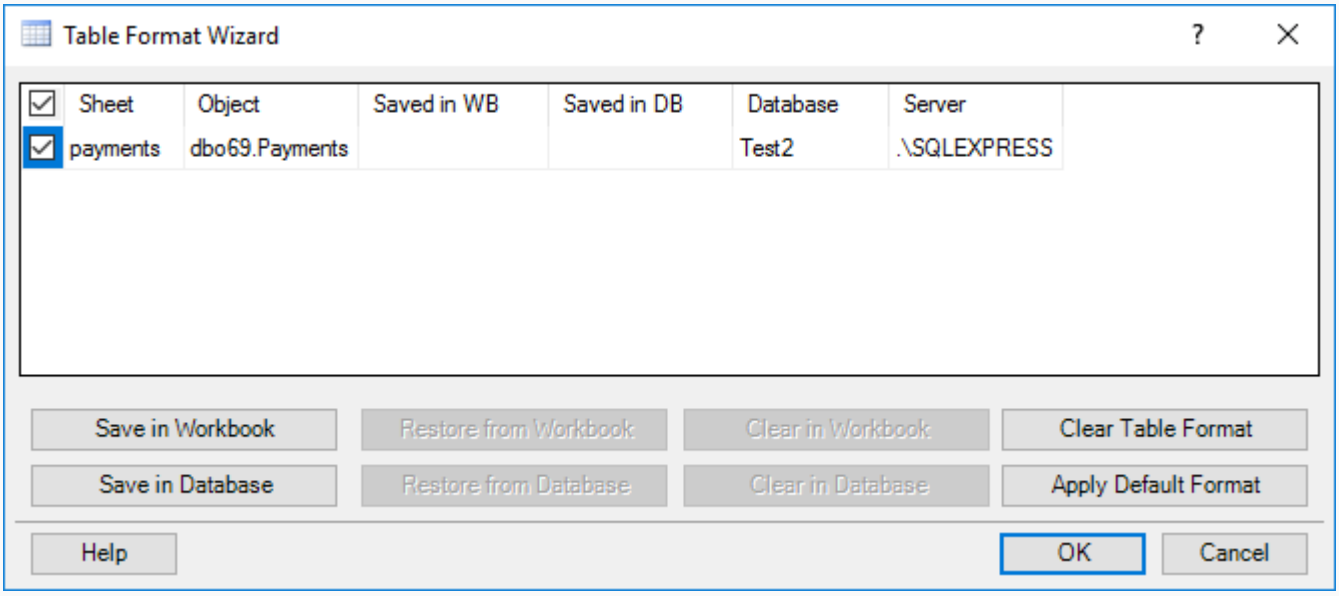
Saving Formats

Start **Table Format Wizard**:

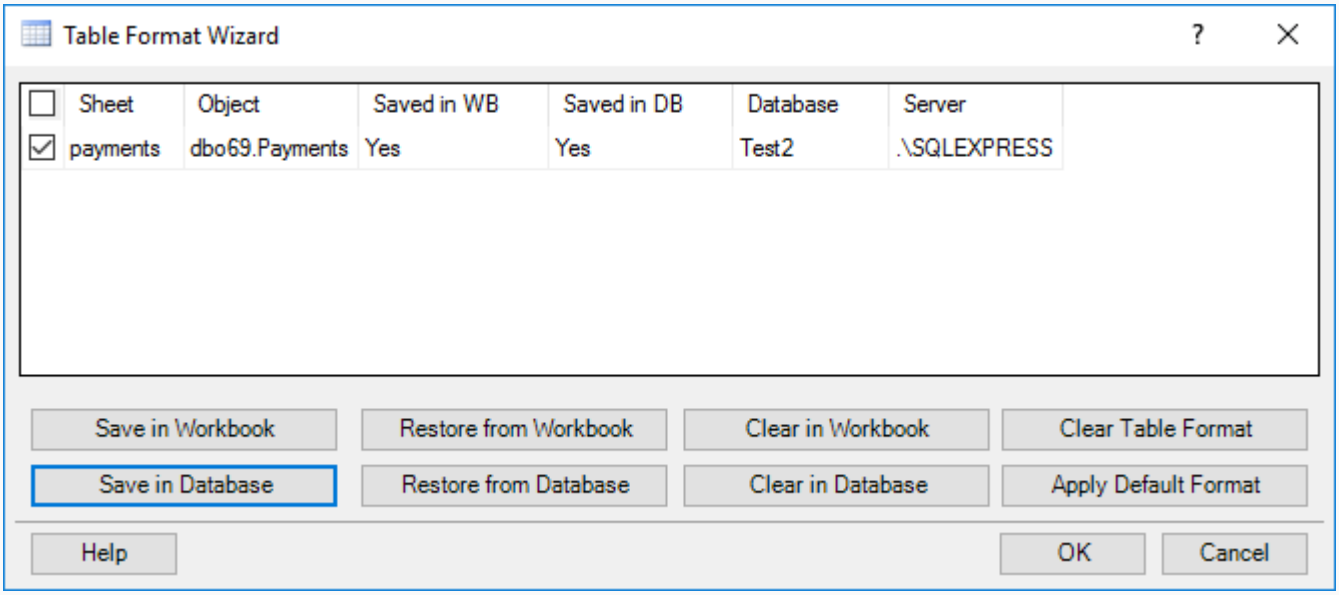


ID	Date	Sum	Account	Company	Item
1	1/10/2017	200,000.00	My Bank	Rose, Inc	Revenue
2	1/10/2017	-50,000.00	My Bank	Land, Inc	Expenses
3	1/31/2017	-87,000.00	My Bank		Payroll
4	1/31/2017	-13,000.00	My Bank	Income Taxes	Taxes
5	1/31/2017	-29,580.00	My Bank	Salary Taxes	Taxes
6	2/10/2017	300,000.00	My Bank	Rose, Inc	Revenue
7	2/10/2017	100,000.00	My Bank	Carnation, Inc	Revenue
8	2/10/2017	-50,000.00	My Bank	Water, Inc	Expenses

In the wizard, select a table and click the **Save in Database** button:



The wizard saves the table format and changes its state:



Now, users will get the same formats, views, and formulas of the tables when they connect to a database.

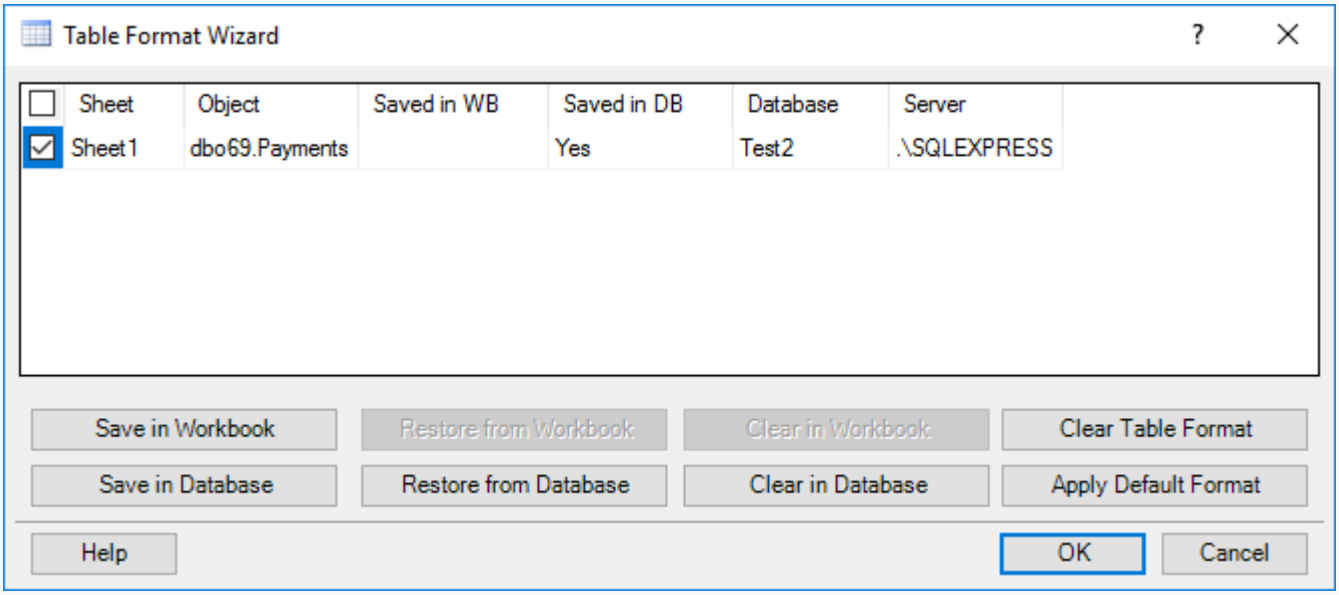
Use the wizard to republish new views later.

Restoring Formats

Users may use the **Restore from Database** button to reload the updated formats and views.

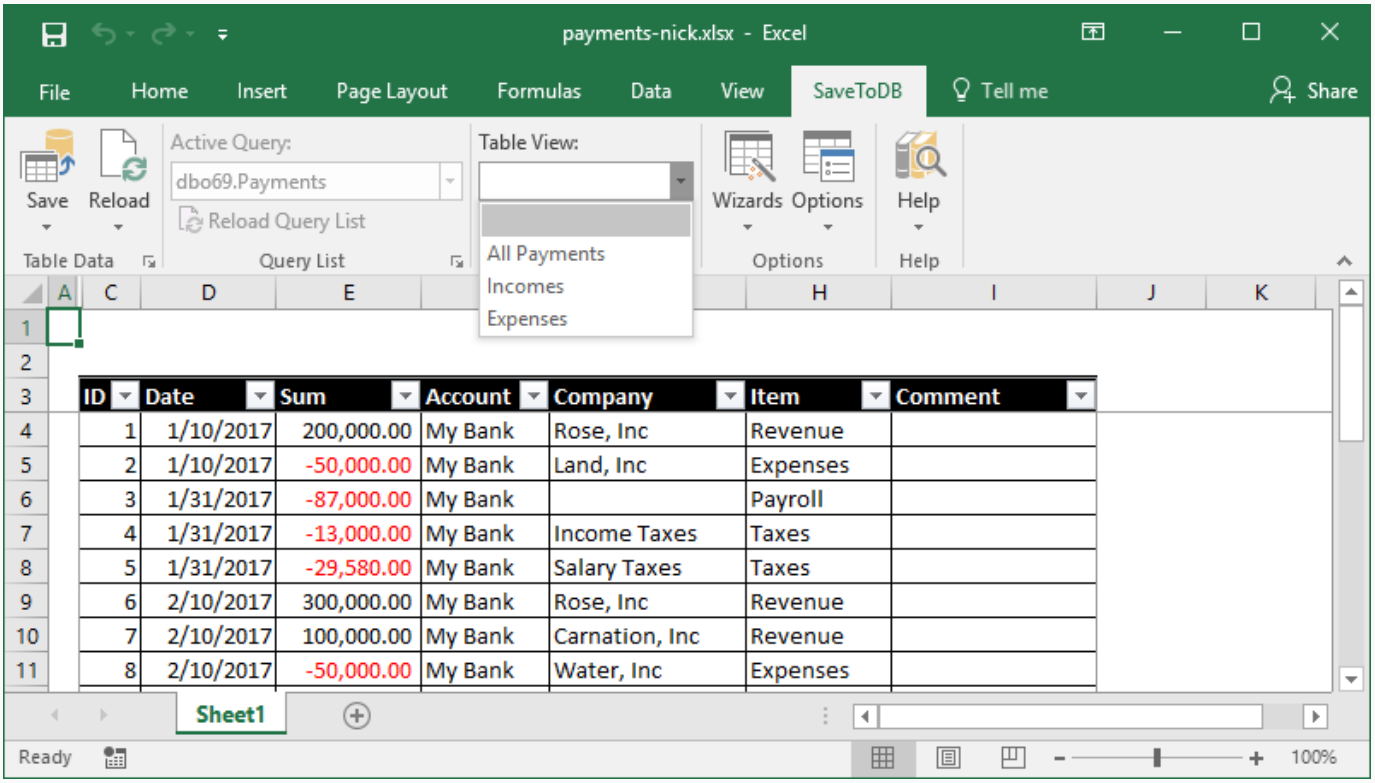
For example, let's switch to Nick's workbook and run **Table Format Wizard**.

Nick may see that dbo69.Payments table has a format in a database:



Select the desired table and click the **Restore from Database** button. Then close the wizard.

Now, the table in Nick's workbook has the same format and table views as in Alex's workbook:



ID	Date	Sum	Account	Company	Item	Comment
1	1/10/2017	200,000.00	My Bank	Rose, Inc	Revenue	
2	1/10/2017	-50,000.00	My Bank	Land, Inc	Expenses	
3	1/31/2017	-87,000.00	My Bank		Payroll	
4	1/31/2017	-13,000.00	My Bank	Income Taxes	Taxes	
5	1/31/2017	-29,580.00	My Bank	Salary Taxes	Taxes	
6	2/10/2017	300,000.00	My Bank	Rose, Inc	Revenue	
7	2/10/2017	100,000.00	My Bank	Carnation, Inc	Revenue	
8	2/10/2017	-50,000.00	My Bank	Water, Inc	Expenses	

Once again, you may use this wizard to save and restore formats, formulas, and table views.

This feature allows sharing best features when every team member uses personal workbooks.

The **Table Format Wizard** saves formats in the **TableFormats** table in a database.

We have created this table using the SQL code provided in the Configuring Database chapter.

Chapter 8. Validation Lists

In this chapter, we will create validation lists for Account, Company, and Item columns.

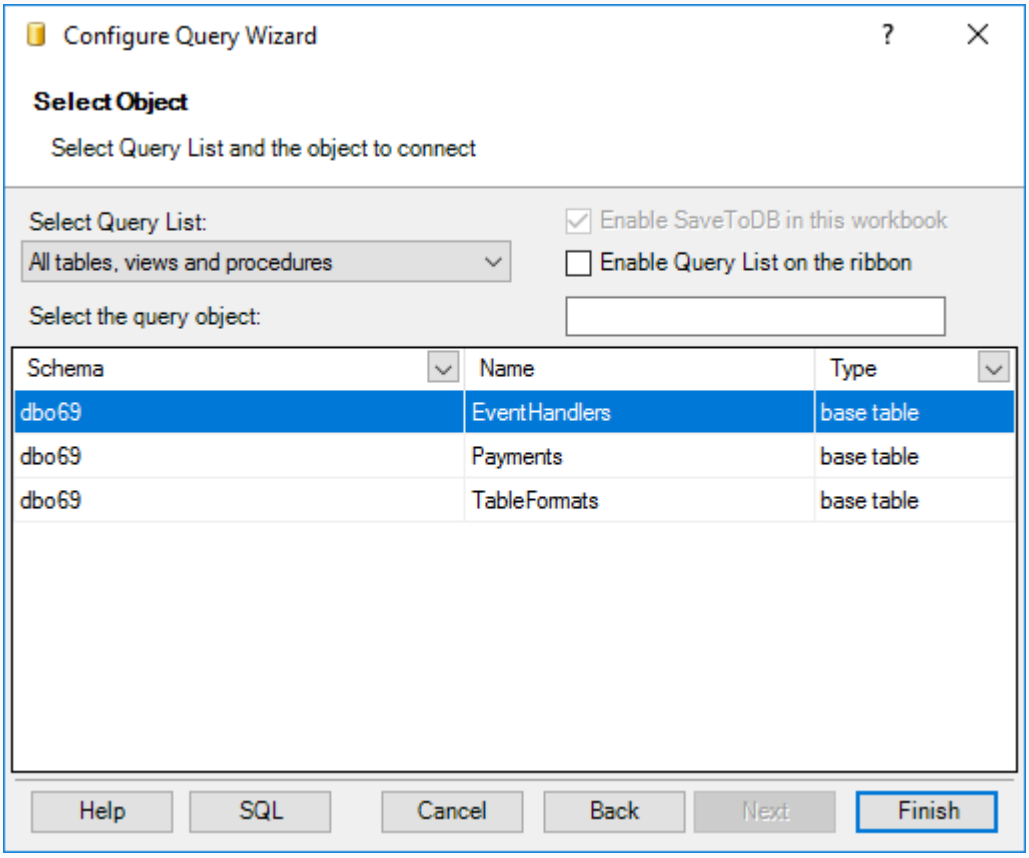
Moreover, we will configure this feature in a database, and the add-in will apply validation rules automatically.

EventHandlers

Configuring validation lists requires the **EventHandlers** table in a database.

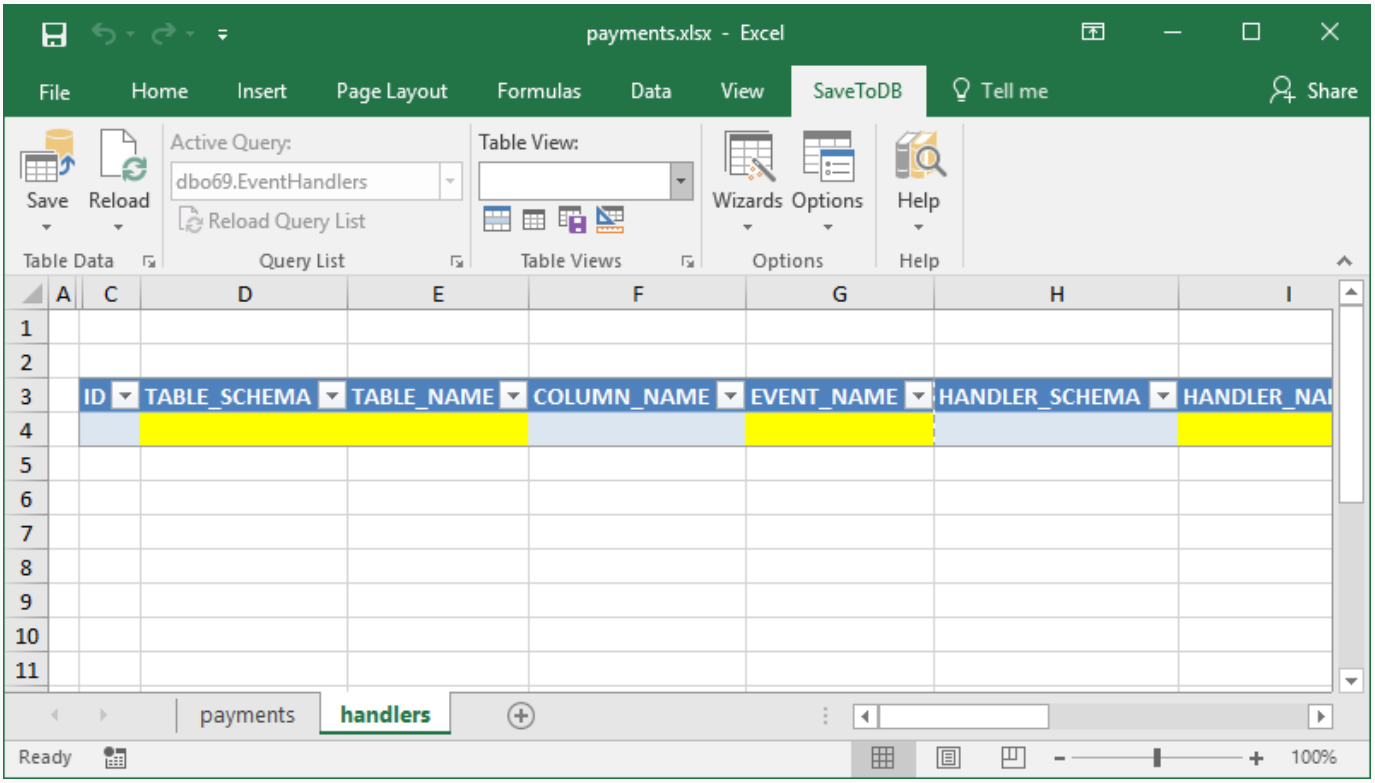
We have created this table using the SQL code provided in the Configuring Database chapter.

Let's add a worksheet, named as **handlers**, run **Data Connection Wizard** and connect to the **EventHandlers** table:



Uncheck **Enable Query List on the ribbon**. Click **Finish**, and insert the table at cell B3.

We have to see the following table:



The add-in highlights fields that require values. Let's format the worksheet and table to make them beautiful.

Static Validation Lists

Add the following configuration and click the **Save** button:

ID	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	EVENT_NAME	HANDLER_SCHEMA	HANDLER_NAME	HANDLER_TYPE	HANDLER_CODE
1	dbo69	Payments	Account	ValidationList	dbo69	Accounts	VALUES	MyBank
2	dbo69	Payments	Item	ValidationList	dbo69	Items	VALUES	Revenue,Expenses,Payroll,Taxes

This table has two parts, "tables" and "handlers":

TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	EVENT_NAME
dbo69	Payments	Account	ValidationList
dbo69	Payments	Item	ValidationList

HANDLER_SCHEMA	HANDLER_NAME	HANDLER_TYPE	HANDLER_CODE
dbo69	Accounts	VALUES	MyBank
dbo69	Items	VALUES	Revenue,Expenses,Payroll,Taxes

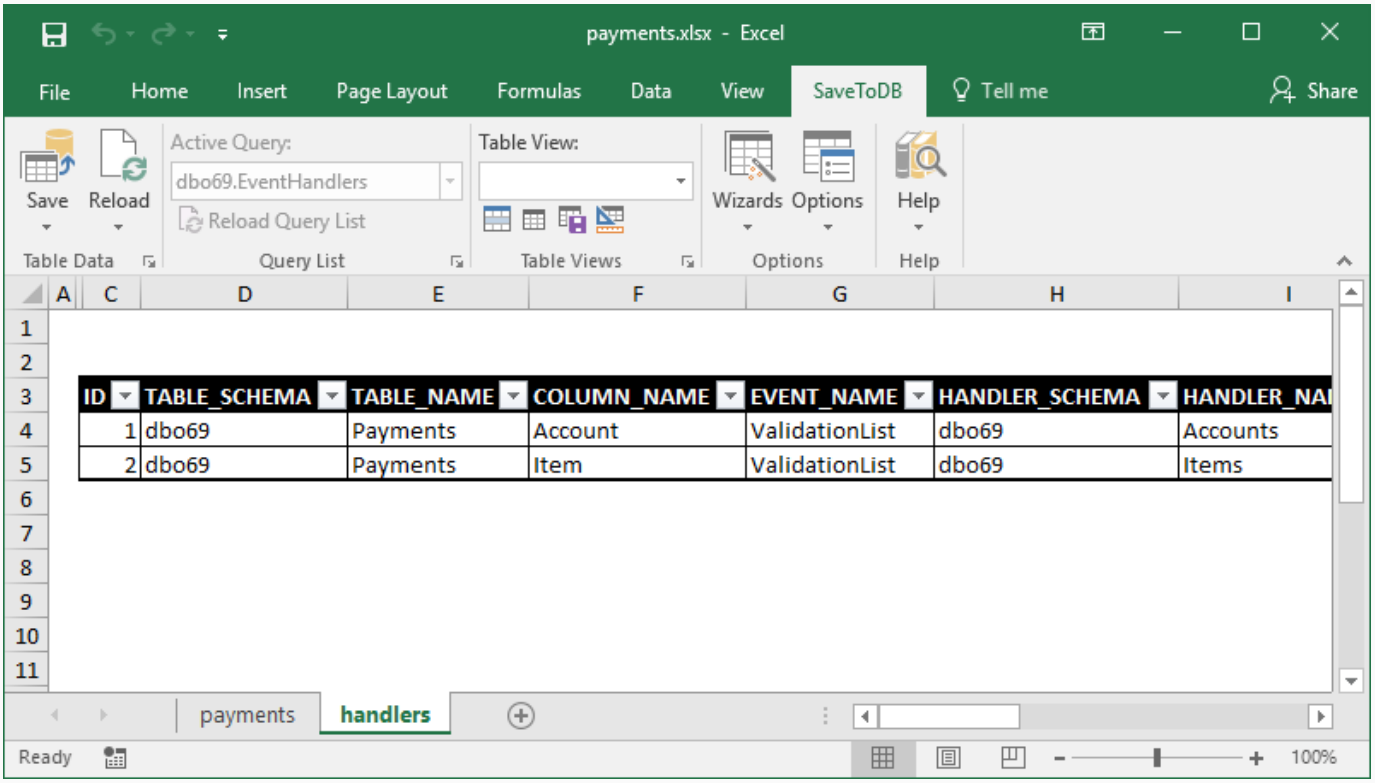
In the table part, we have specified the configured table, its columns and the event name as **ValidationList**.

In the handler part, we have defined how to handle events. In this case, we have configured the **VALUES** type and specified static values in the **HANDLER_CODE** field.

Further, I will use the short format like this, without the SCHEMA fields that contain the dbo69 value:

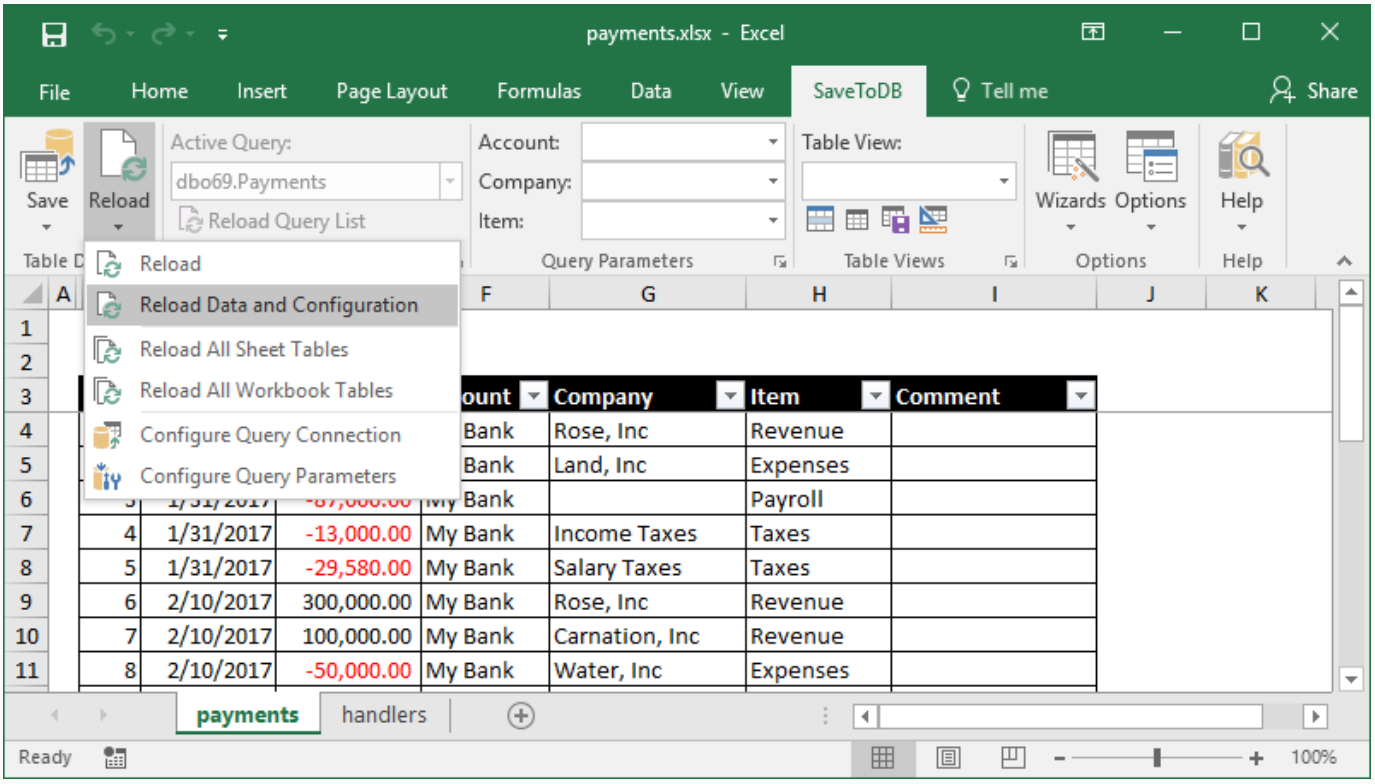
TABLE_NAM	COLUMN_	EVENT_NAME	HANDLER	HAND	HANDLER_CODE
Payments	Account	ValidationList	Accounts	VALUES	My Bank
Payments	Item	ValidationList	Items	VALUES	Revenue,Expenses,Payroll,Taxes

Now, the **handlers** worksheet should look like this:



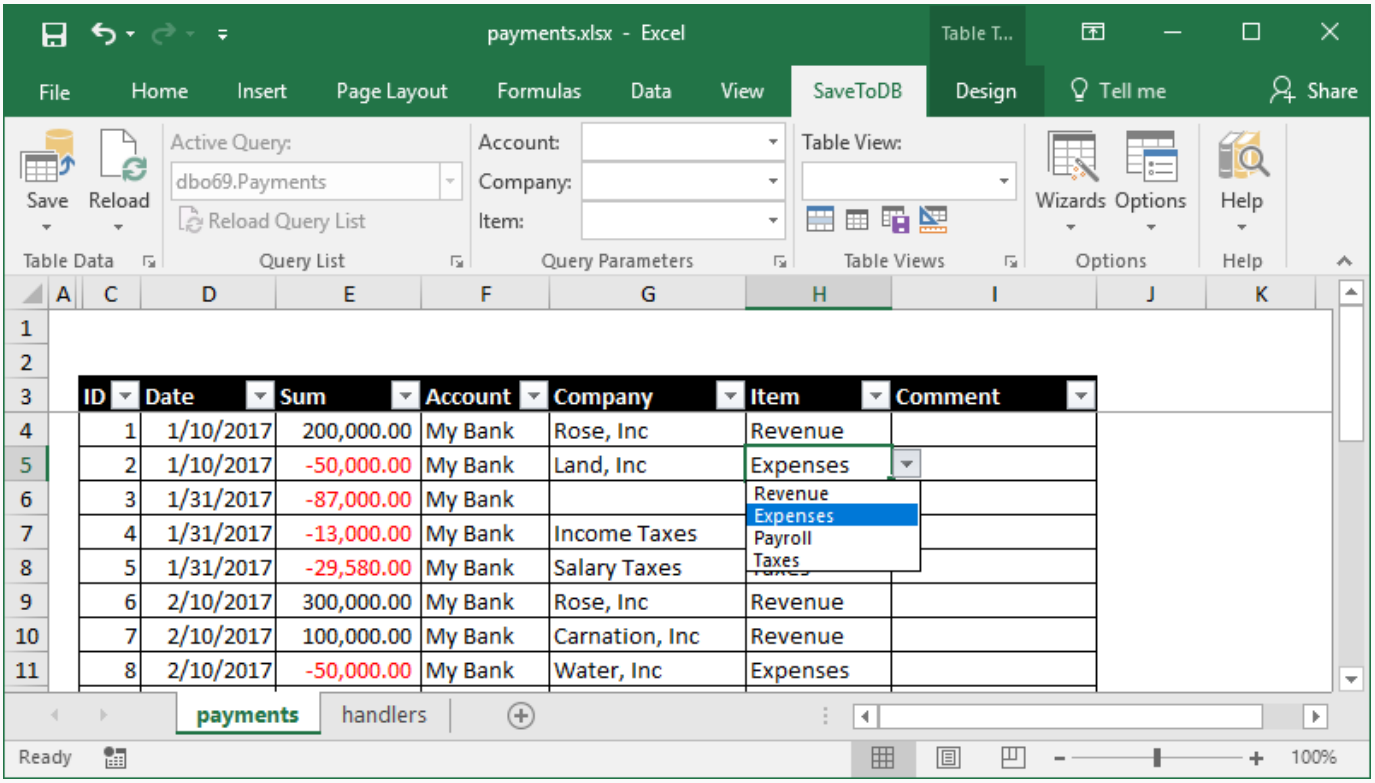
ID	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	EVENT_NAME	HANDLER_SCHEMA	HANDLER_NAI
1	dbo69	Payments	Account	ValidationList	dbo69	Accounts
2	dbo69	Payments	Item	ValidationList	dbo69	Items

Let's switch to the **payments** worksheet and click **Reload**, **Reload Data and Configuration**:



			ount	Company	Item	Comment
4	1/31/2017	-87,000.00	Bank	Rose, Inc	Revenue	
5	1/31/2017	-87,000.00	Bank	Land, Inc	Expenses	
6	1/31/2017	-87,000.00	Bank		Payroll	
7	1/31/2017	-13,000.00	My Bank	Income Taxes	Taxes	
8	1/31/2017	-29,580.00	My Bank	Salary Taxes	Taxes	
9	2/10/2017	300,000.00	My Bank	Rose, Inc	Revenue	
10	2/10/2017	100,000.00	My Bank	Carnation, Inc	Revenue	
11	2/10/2017	-50,000.00	My Bank	Water, Inc	Expenses	

The add-in creates validation lists using the specified values:

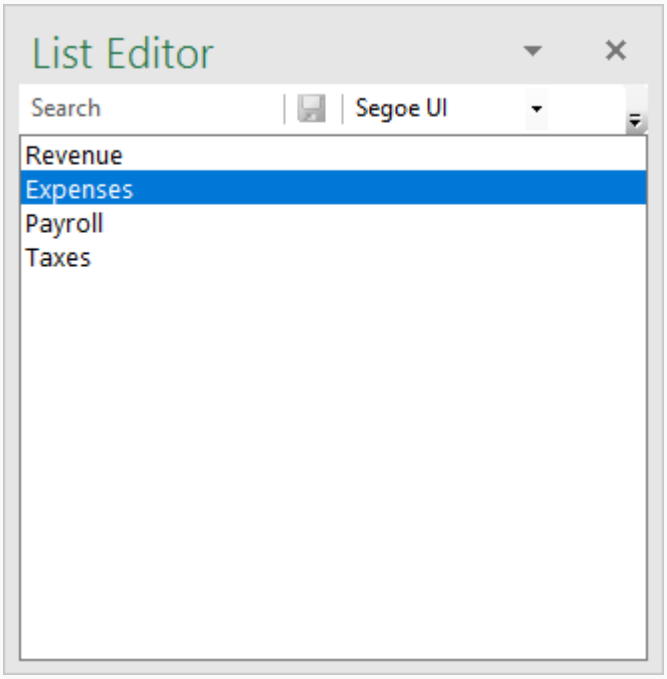


ID	Date	Sum	Account	Company	Item	Comment
1	1/10/2017	200,000.00	My Bank	Rose, Inc	Revenue	
2	1/10/2017	-50,000.00	My Bank	Land, Inc	Expenses	
3	1/31/2017	-87,000.00	My Bank			
4	1/31/2017	-13,000.00	My Bank	Income Taxes		
5	1/31/2017	-29,580.00	My Bank	Salary Taxes		
6	2/10/2017	300,000.00	My Bank	Rose, Inc	Revenue	
7	2/10/2017	100,000.00	My Bank	Carnation, Inc	Revenue	
8	2/10/2017	-50,000.00	My Bank	Water, Inc	Expenses	

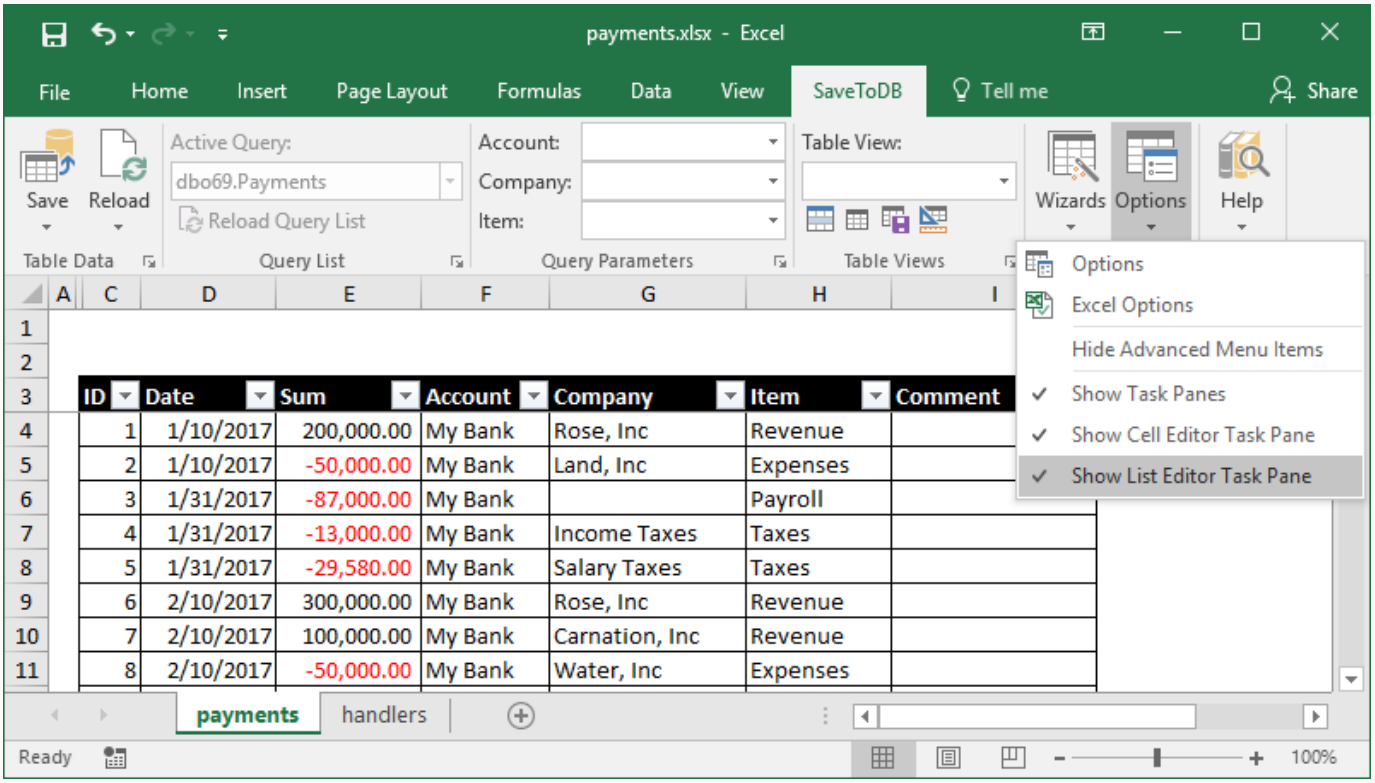
Revenue
Expenses
Payroll
Taxes

List Editor

The add-in activates **List Editor** that allows selecting values from lists in a comfortable way, including search.



You may turn on/off the **List Editor** using the **Options**, **Show List Editor Task Pane** option.

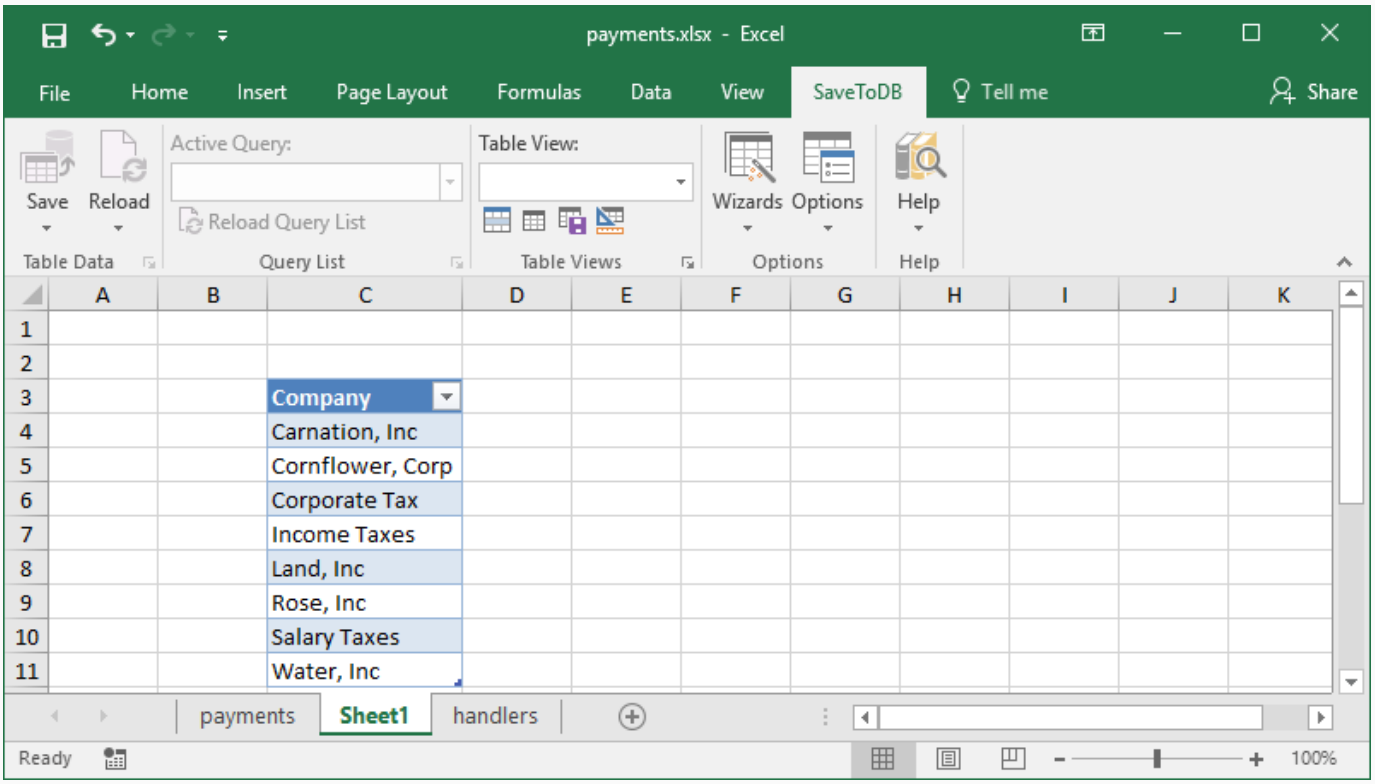


ID	Date	Sum	Account	Company	Item	Comment
1	1/10/2017	200,000.00	My Bank	Rose, Inc	Revenue	
2	1/10/2017	-50,000.00	My Bank	Land, Inc	Expenses	
3	1/31/2017	-87,000.00	My Bank		Payroll	
4	1/31/2017	-13,000.00	My Bank	Income Taxes	Taxes	
5	1/31/2017	-29,580.00	My Bank	Salary Taxes	Taxes	
6	2/10/2017	300,000.00	My Bank	Rose, Inc	Revenue	
7	2/10/2017	100,000.00	My Bank	Carnation, Inc	Revenue	
8	2/10/2017	-50,000.00	My Bank	Water, Inc	Expenses	

Dynamic Validation Lists

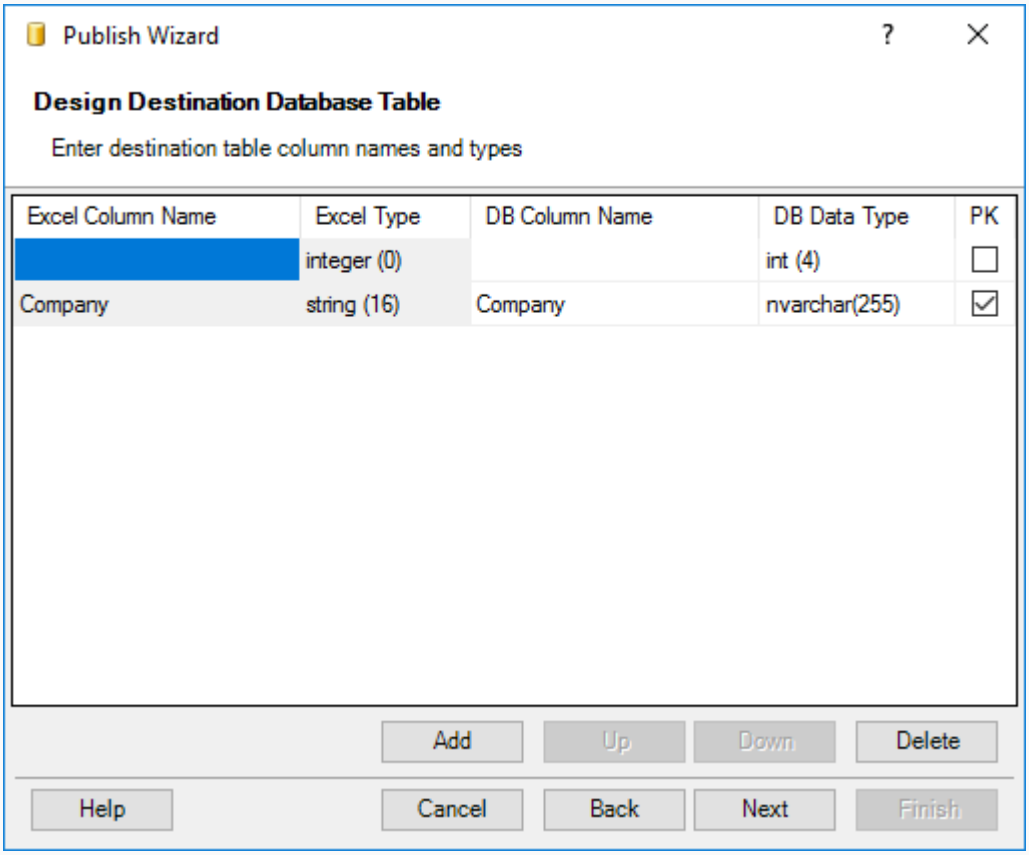
Let's create a validation list for the Company column. We must have the possibility to change values easily.

Create an Excel table with companies at a new worksheet starting cell C3:



Company
Carnation, Inc
Cornflower, Corp
Corporate Tax
Income Taxes
Land, Inc
Rose, Inc
Salary Taxes
Water, Inc

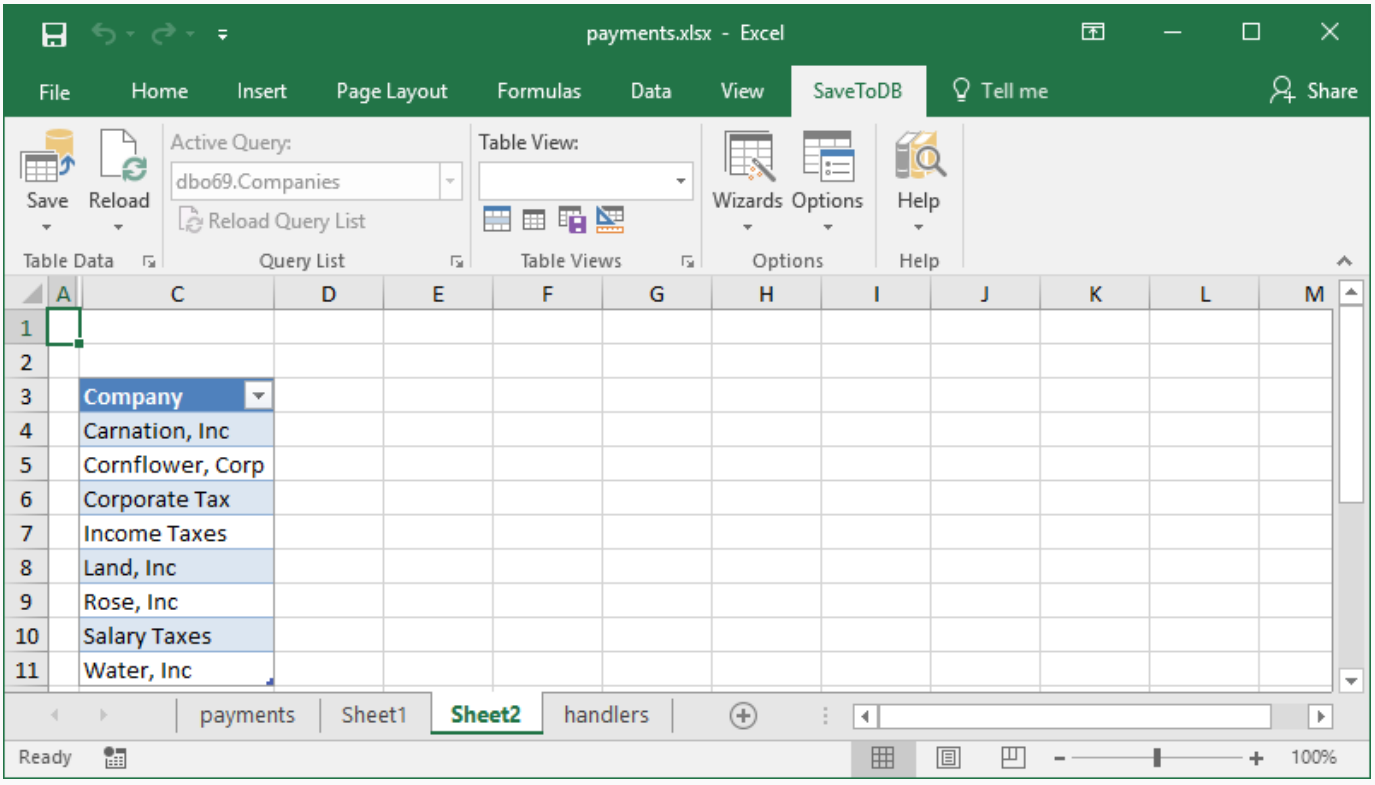
Run Publish Wizard as described above. At the design tab, we have the following structure:



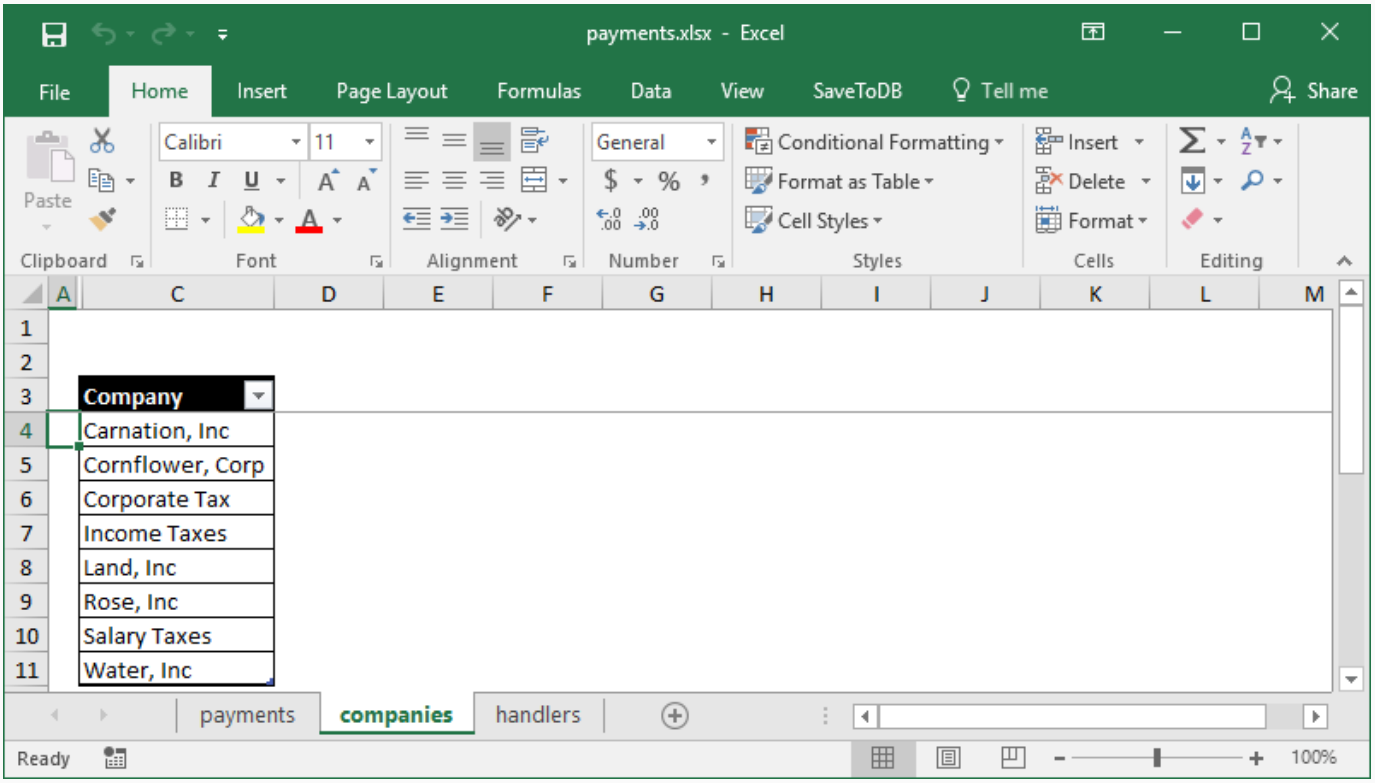
The table contains one column that is used as a primary key column. It's ok. Click **Next** and execute next steps.

Publish table as dbo69.Companies table.

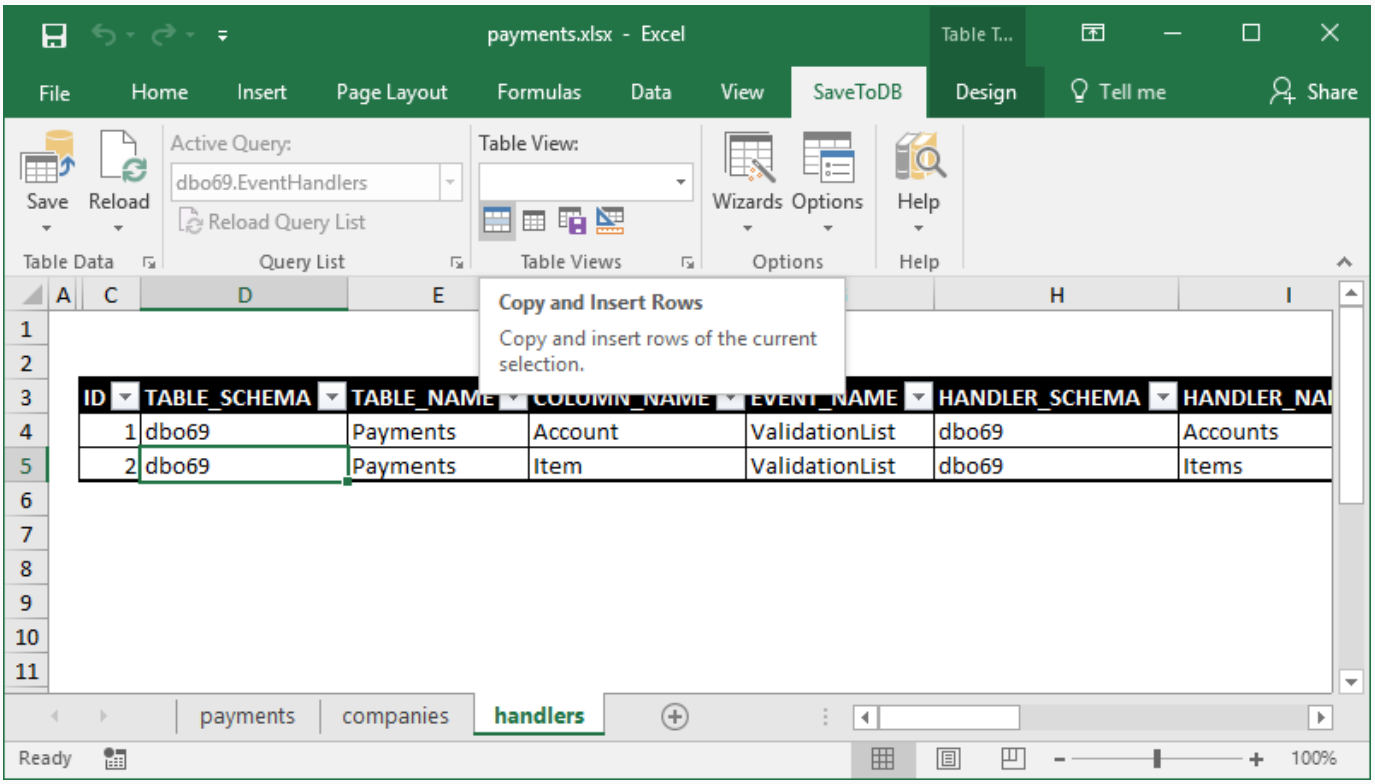
As a Publish Wizard result, we have the editable **dbo69.Companies** table of companies at a new sheet:



Let's remove the initial Sheet1 worksheet, rename the Sheet2 to **companies**, and format the table:



Switch to the **handlers** sheet, select a cell in row 5, and click the **Copy and Insert Rows** button.



This action will create a copy of the selected line. So, you may easily change only different values.

Add a validation list configuration for the Company field like this:

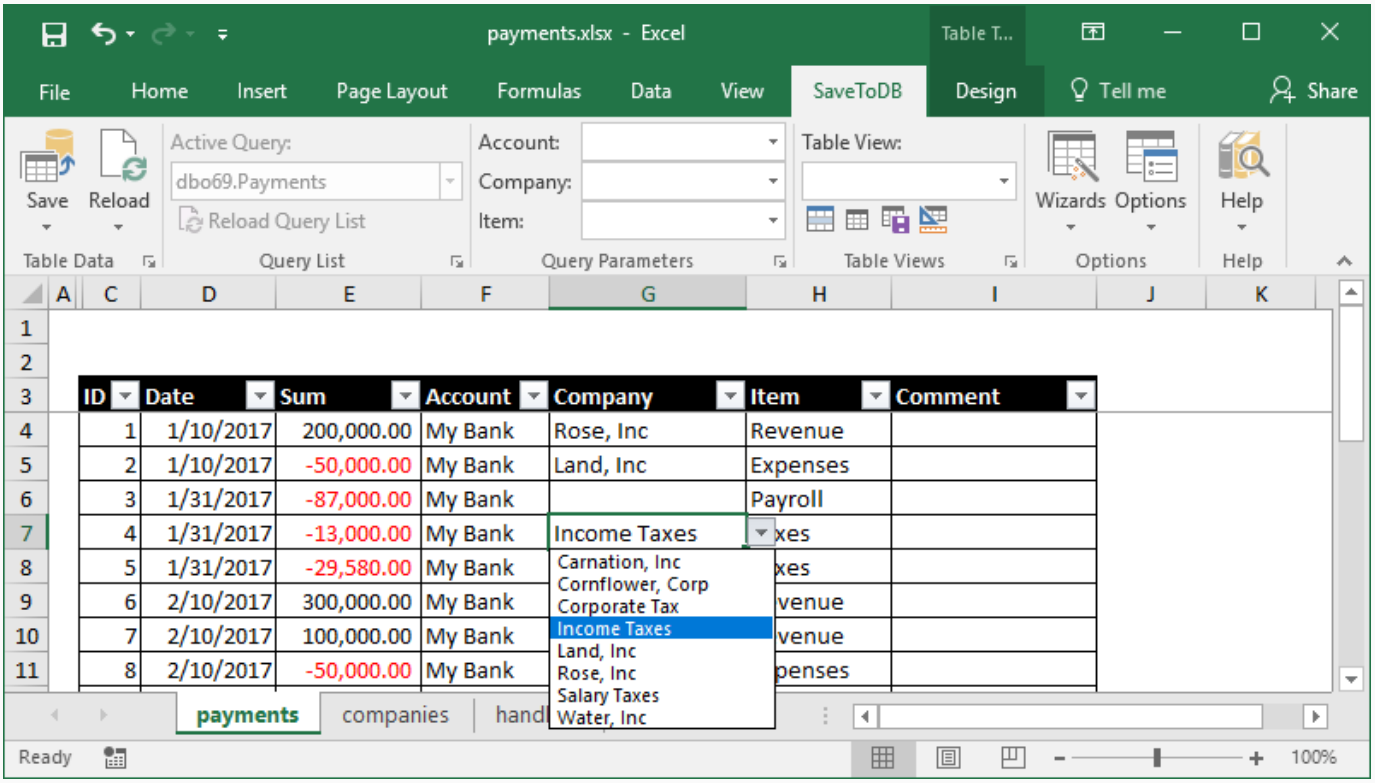
TABLE_NAM	COLUMN_	EVENT_NAME	HANDLER	HAND	HANDLER_CODE
Payments	Account	ValidationList	Accounts	VALUES	My Bank
Payments	Item	ValidationList	Items	VALUES	Revenue,Expenses,Payroll,Taxes
Payments	Company	ValidationList	Companies	TABLE	Company

Complete table:

ID	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	EVENT_NAME	HANDLER_SCHEMA	HANDLER_NAME	HANDLER_TYPE	HANDLER_CODE
1	dbo69	Payments	Account	ValidationList	dbo69	Accounts	VALUES	MyBank
2	dbo69	Payments	Item	ValidationList	dbo69	Items	VALUES	Revenue,Expenses,Payroll,Taxes
3	dbo69	Payments	Company	ValidationList	dbo69	Companies	TABLE	Company

As a result, the add-in will create a validation list for the Company column using the **Company** field of the **dbo69.Companies** table.

Switch to the **payments** worksheet and click **Reload**, **Reload Data and Configuration** to refresh the list:



You have to know that the add-in places connected tables of validation list source values like dbo69.Companies to a hidden worksheet and configures validation lists using a range formulas.

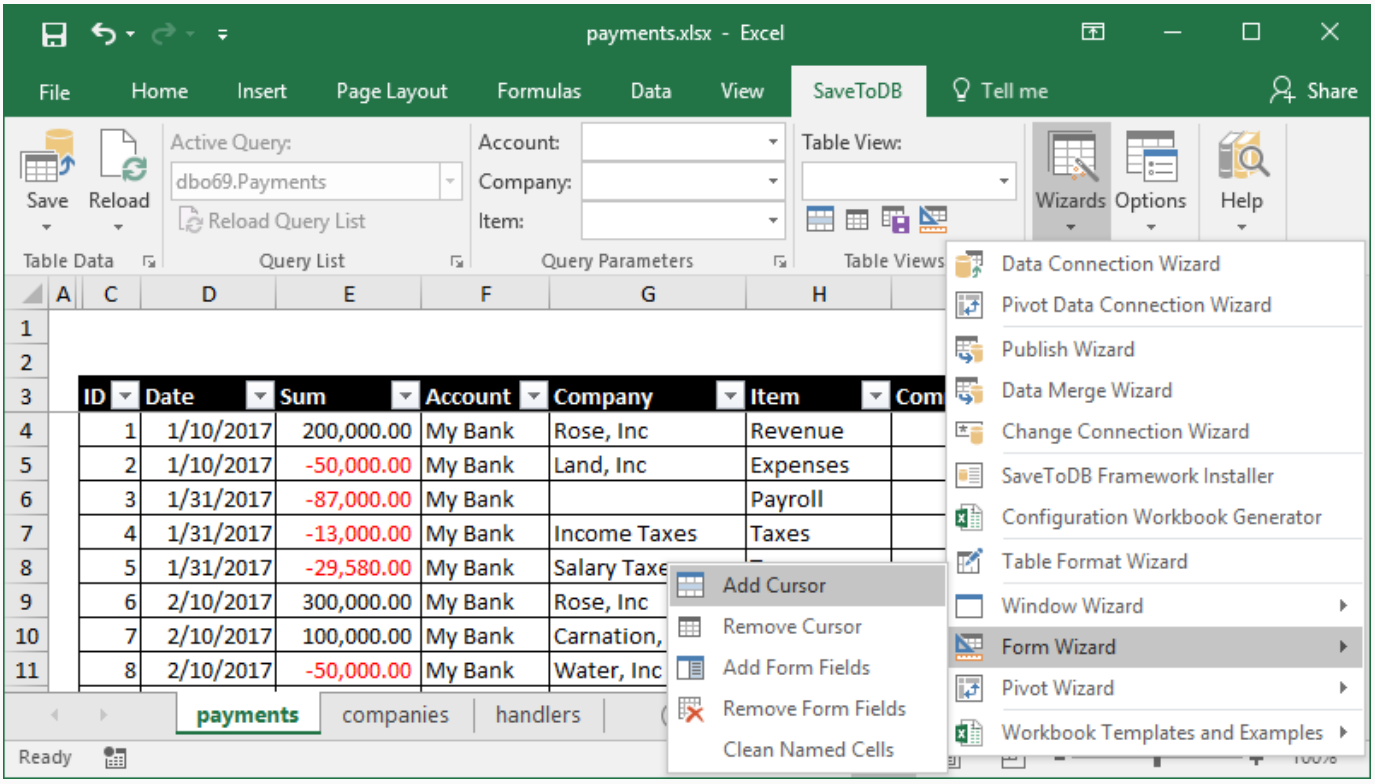
Setting validation lists this way has great benefits.

You configure it in the EventHandlers table, and the add-in creates validation lists automatically.

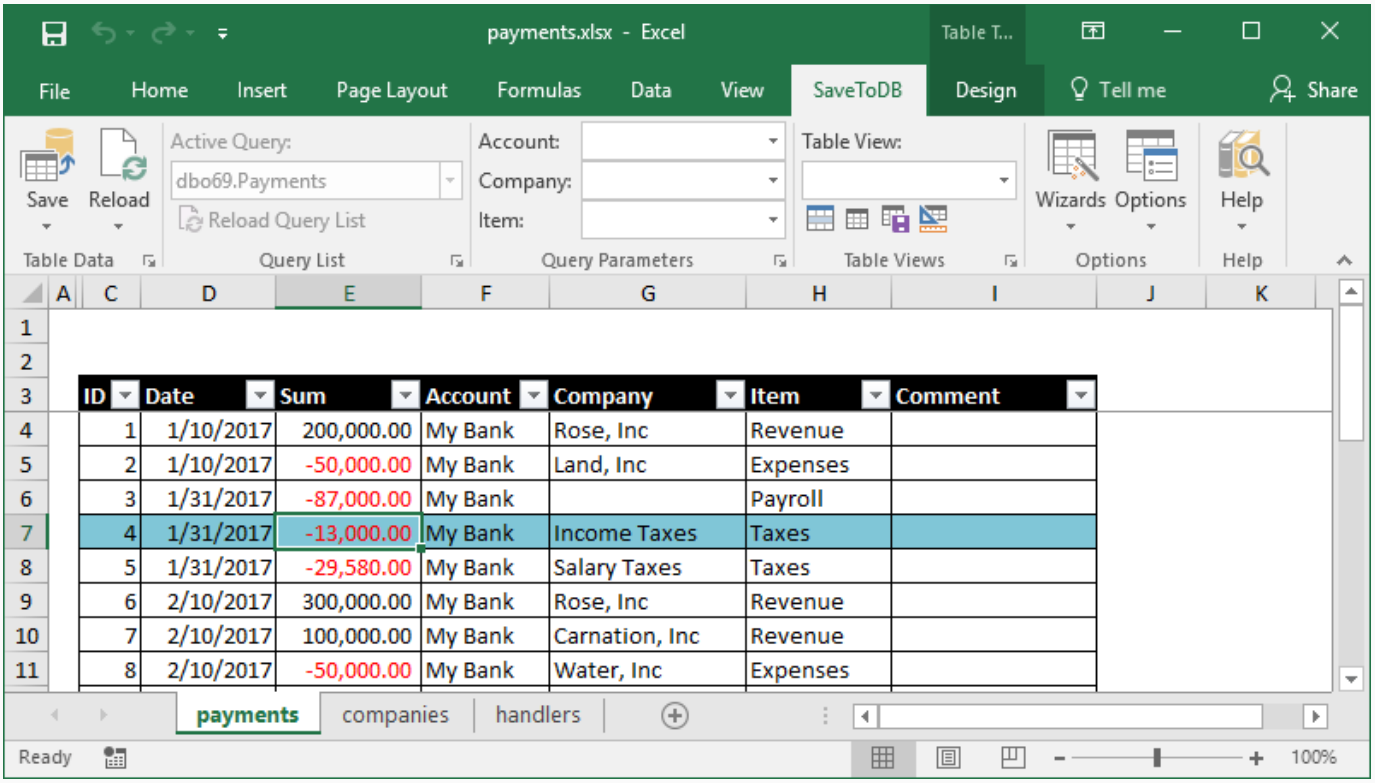
Just use **Reload**, **Reload Data and Configuration** to reload validation list values.

Chapter 9. Cursors

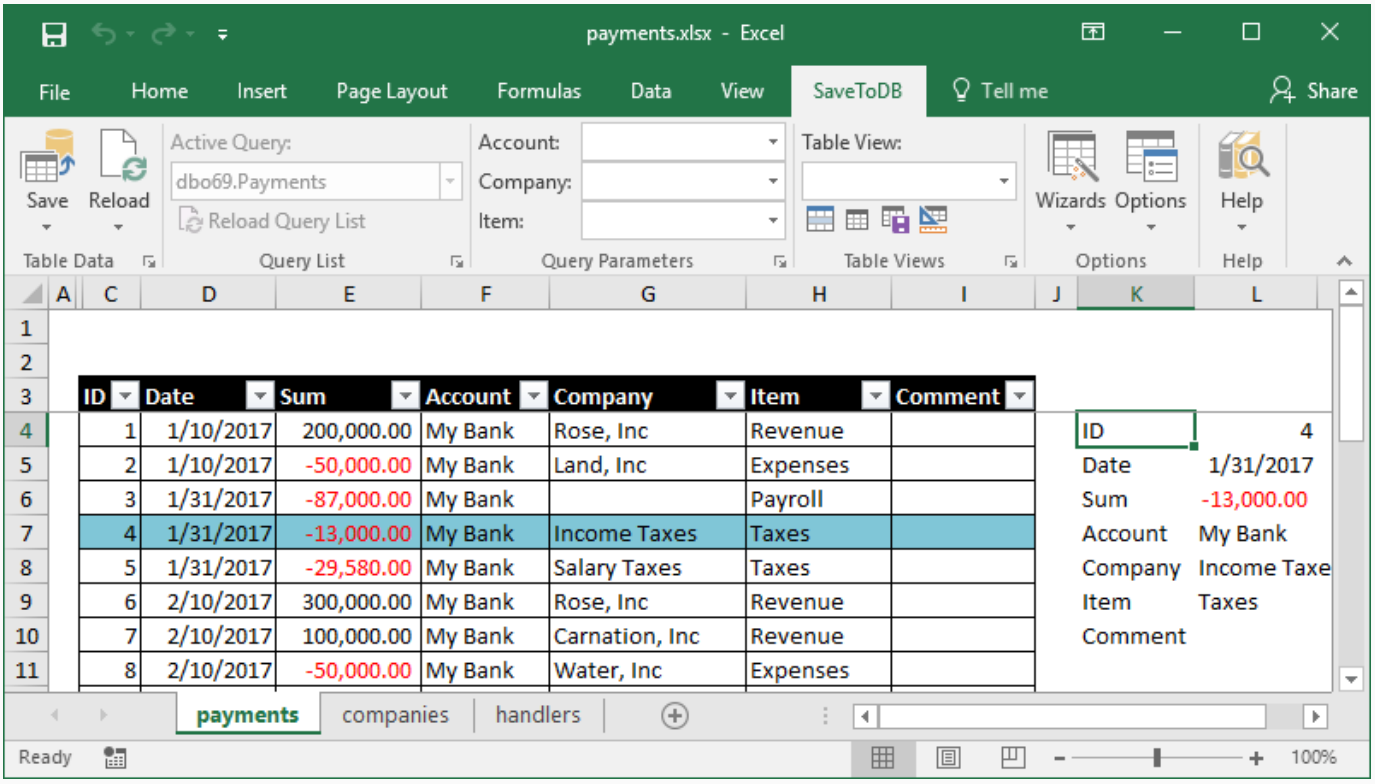
Let's select the **Payments** worksheet and click **Wizards**, **Form Wizard**, **Add Cursor**:



The add-in highlights the active table row:



Let's run the same wizard, click **Add Form Fields** and select cell K4:



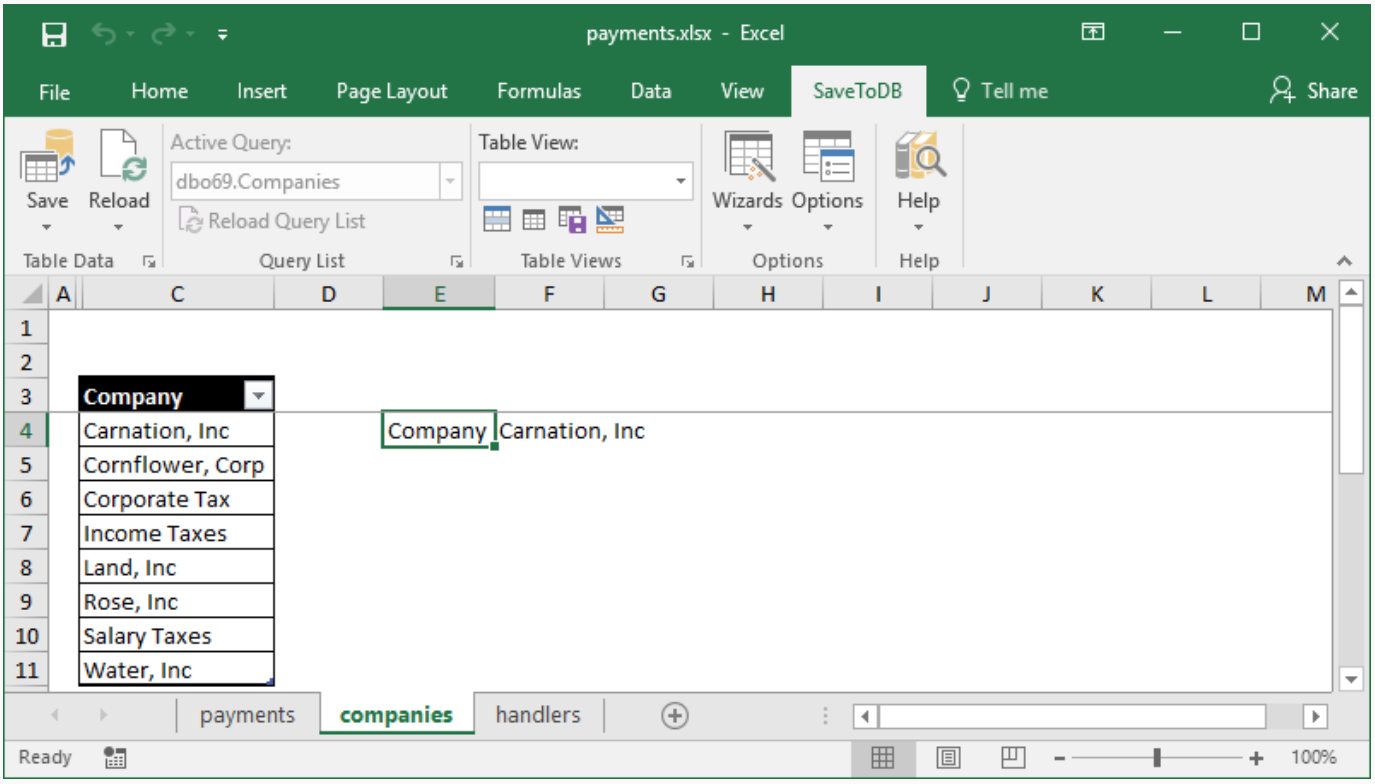
ID	Date	Sum	Account	Company	Item	Comment
1	1/10/2017	200,000.00	My Bank	Rose, Inc	Revenue	
2	1/10/2017	-50,000.00	My Bank	Land, Inc	Expenses	
3	1/31/2017	-87,000.00	My Bank		Payroll	
4	1/31/2017	-13,000.00	My Bank	Income Taxes	Taxes	
5	1/31/2017	-29,580.00	My Bank	Salary Taxes	Taxes	
6	2/10/2017	300,000.00	My Bank	Rose, Inc	Revenue	
7	2/10/2017	100,000.00	My Bank	Carnation, Inc	Revenue	
8	2/10/2017	-50,000.00	My Bank	Water, Inc	Expenses	

ID	4
Date	1/31/2017
Sum	-13,000.00
Account	My Bank
Company	Income Taxe
Item	Taxes
Comment	

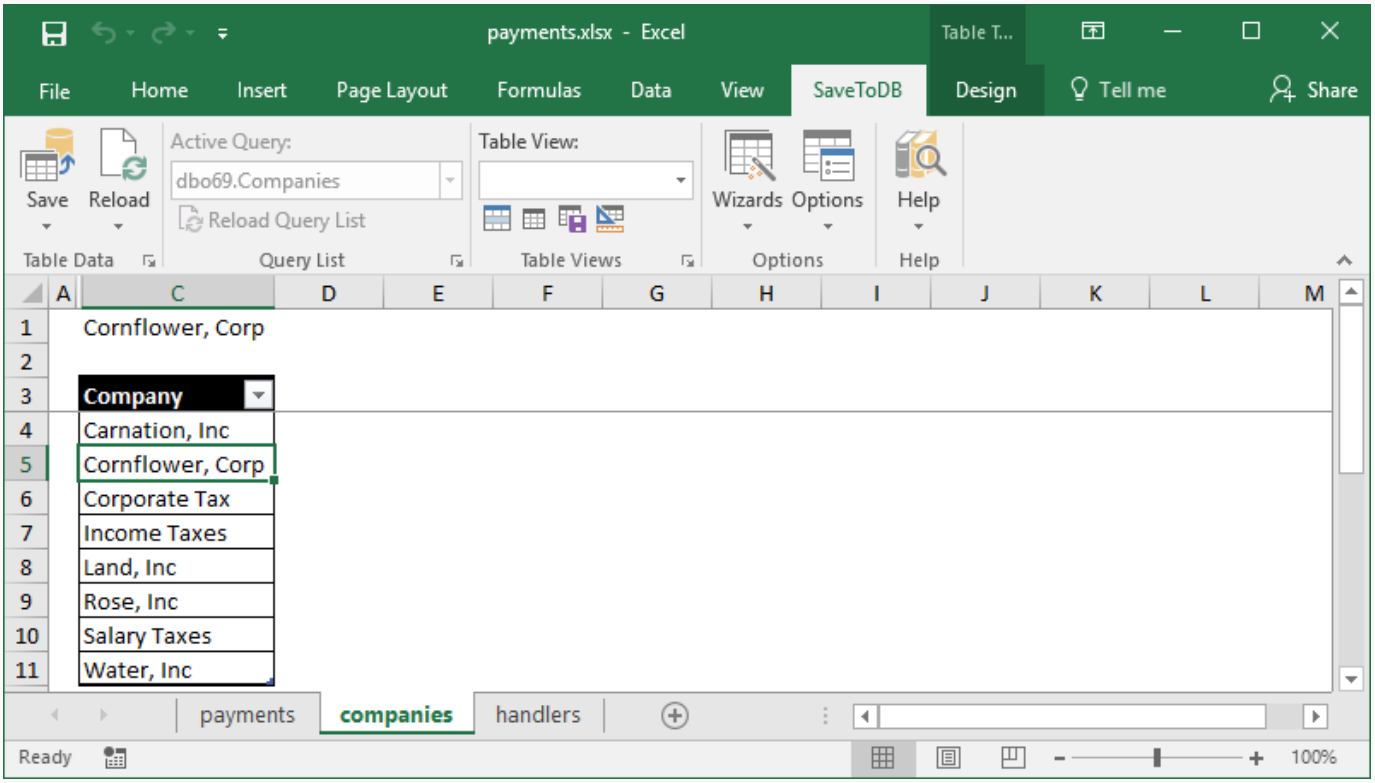
The add-in inserts form fields and updates them when you select another row.

Moreover, you may use such fields to edit table row values.

Let's select the **companies** sheet and add form fields at cell E4.

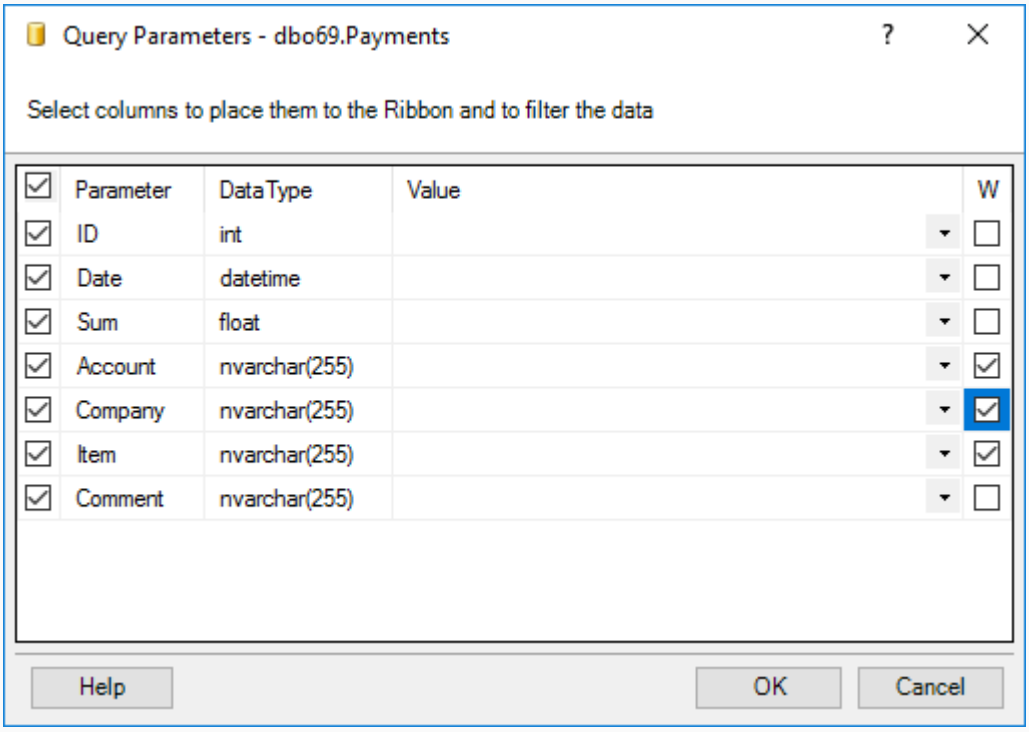


Then let's move cell F4, the Company field, to C1 and remove column E. Now cell C1 contains the active company.

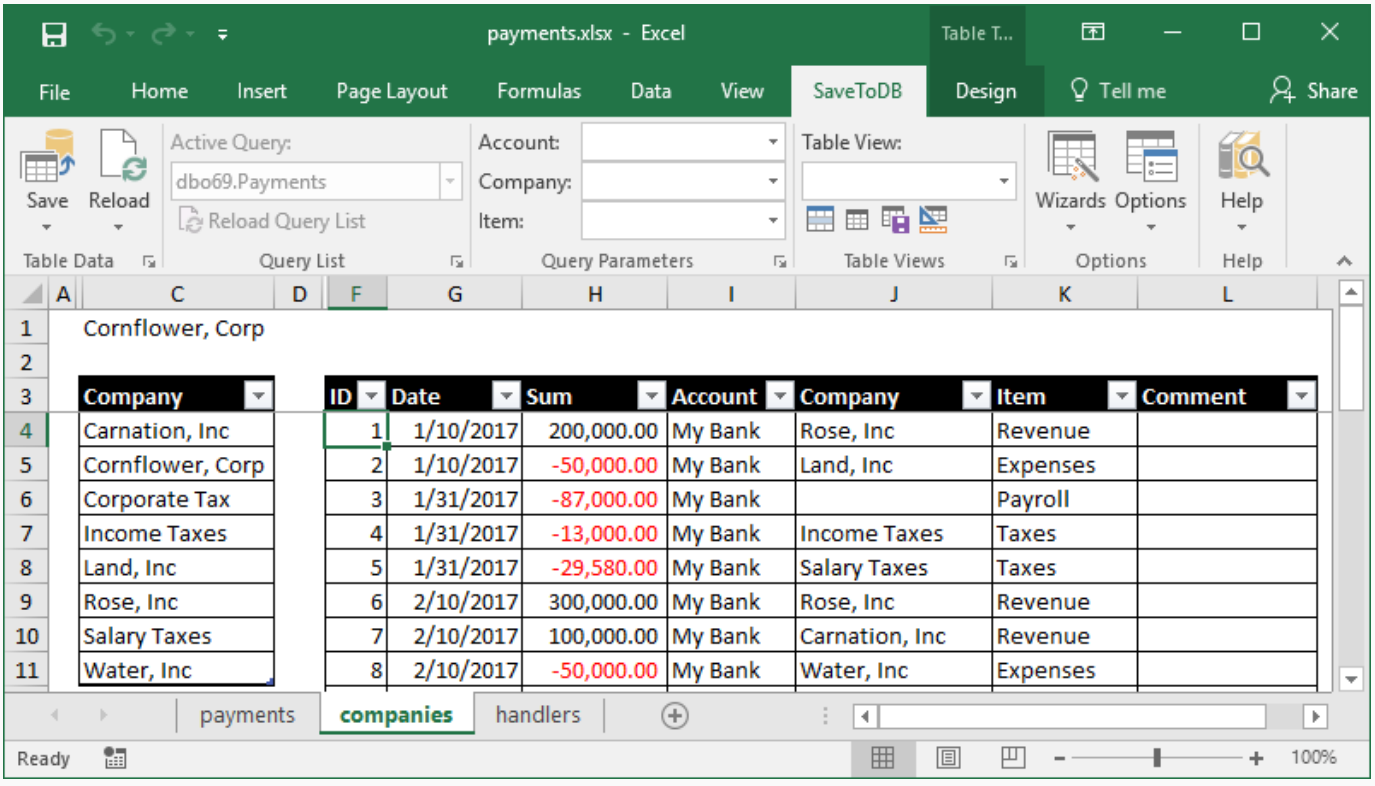


Now, select cell E3 (outside of the active table) and connect to the **dbo69.Payments** table.

In the connection wizard, check at least the field used as a filter (Company) in the **WHERE** column:



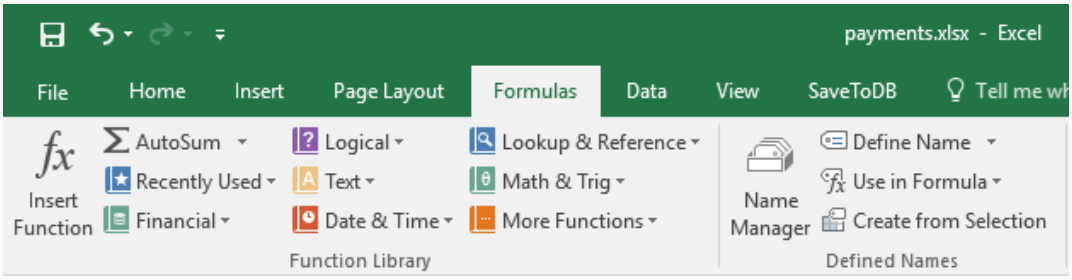
and insert the table at cell E3:



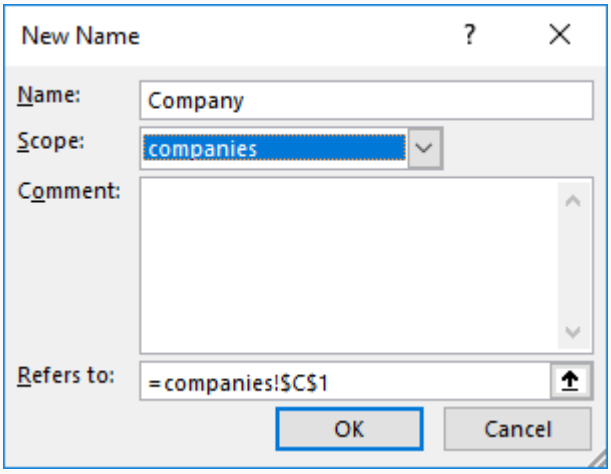
We see that the **dbo69.Payments** table has ribbon parameters including **Company**.

Also, we see the selected company in cell C1 that may be used as a parameter. Let's link them.

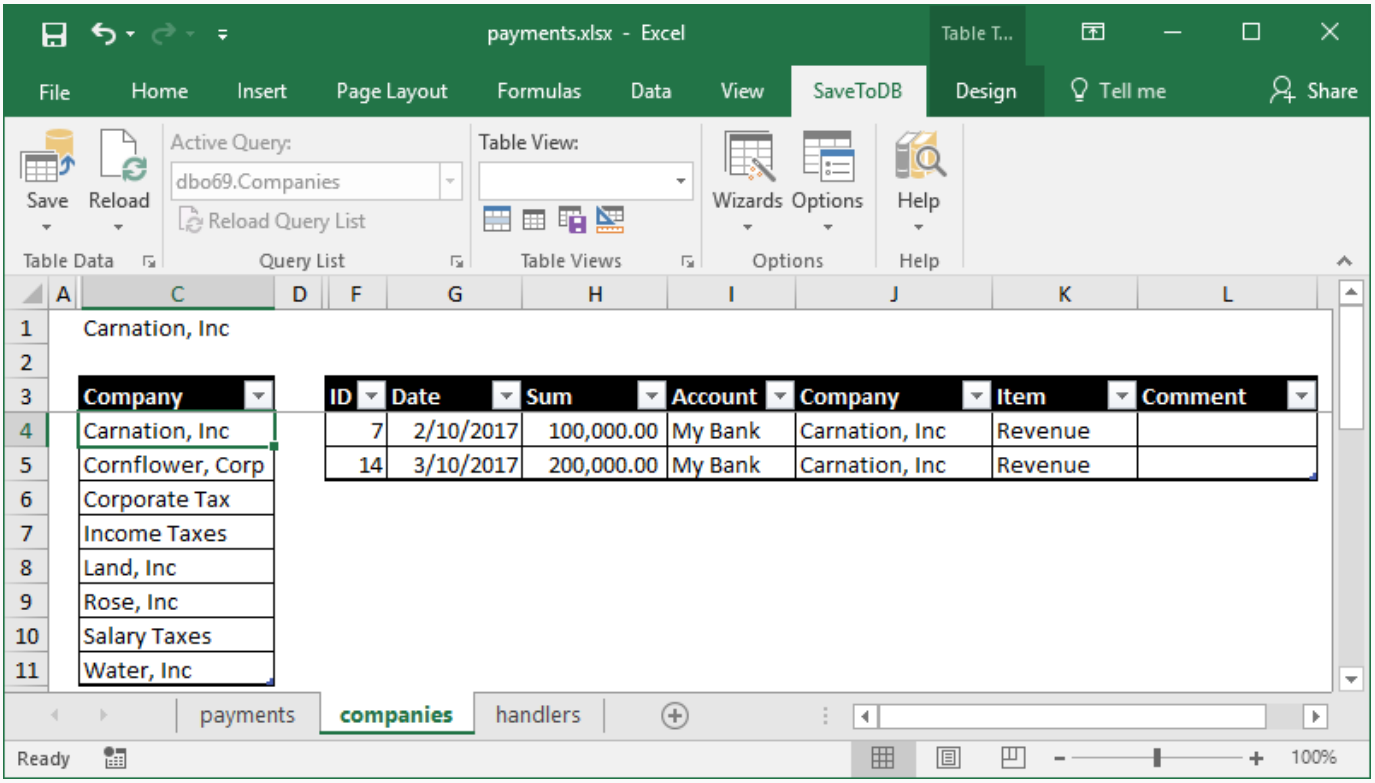
Select cell C1 and click the **Define Name** button in the **Defined Names** group on the **Formulas** tab:



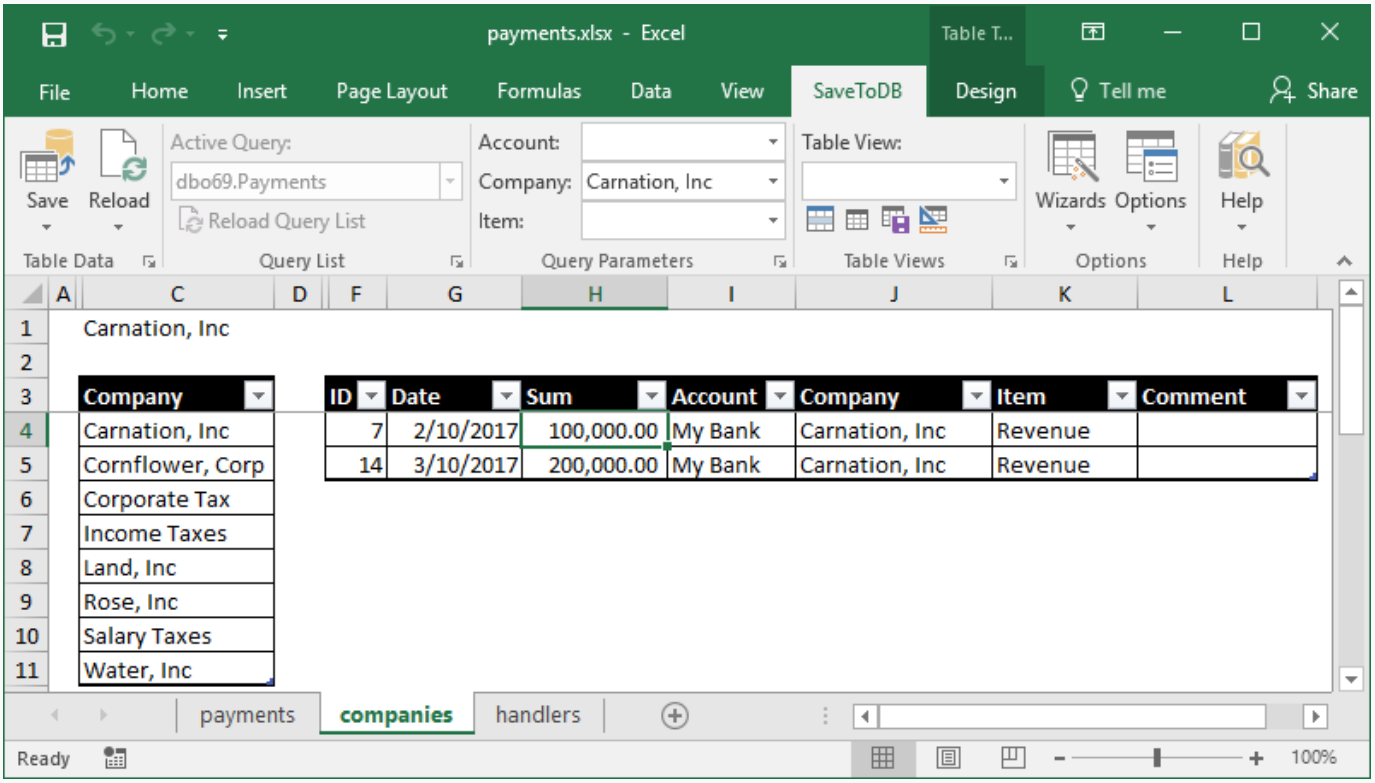
Specify the name as the parameter name, **Company**, and select the sheet name, **companies**, in the **Scope** field:



Click **OK**. Then select a row in the Company table, and voilà!



When you select the detail table, you may see its parameters at the ribbon:



Both tables are editable.

Here is a list of events that implement master-details:

1. A user selects another row in a master table.
2. The add-in updates form fields (cell C1 with Company).
3. The add-in changes parameters of the detail tables using named cell values (cell C1 as Company).
4. The add-in reloads the details with new parameter values.

You may build more complex forms using the same technique.

For example, the Northwind example contains the following tables with master-detail relations:

1. Customer first char index (A-Z)
2. Customers
3. Customer orders
4. Orders

Chapter 12. Detail Windows and Task Panes

We may use another technique to show details in windows and task panes using the **SelectionChange** handlers.

Detail Windows

Let's select the **handlers** worksheet and add the **SelectionChange** handler:

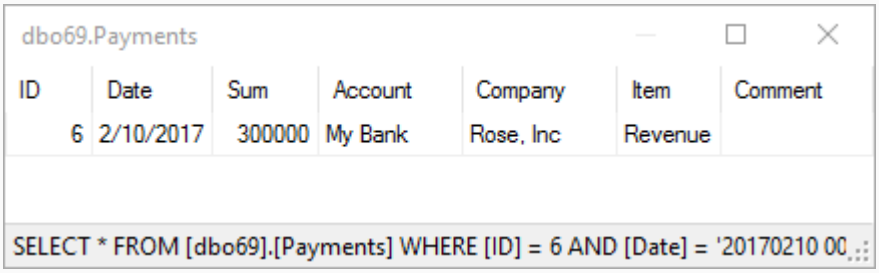
TABLE_NAM	COLUMN_	EVENT_NAME	HANDLER	HAND	HANDLER_CODE
Payments	Account	ValidationList	Accounts	VALUES	My Bank
Payments	Item	ValidationList	Items	VALUES	Revenue,Expenses,Payroll,Taxes
Payments	Company	ValidationList	Companies	TABLE	Company
Payments		SelectionChange	Payments	TABLE	

Complete table:

ID	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	EVENT_NAME	HANDLER_SCHEMA	HANDLER_NAME	HANDLER_TYPE	HANDLER_CODE	TARGET_WORKSHEET
1	dbo69	Payments	Account	ValidationList	dbo69	Accounts	VALUES	My Bank	
2	dbo69	Payments	Item	ValidationList	dbo69	Items	VALUES	Revenue,Expenses,Payroll,Taxes	
3	dbo69	Payments	Company	ValidationList	dbo69	Companies	TABLE	Company	
4	dbo69	Payments		SelectionChange	dbo69	Payments	TABLE		

Switch to the **payments** worksheet and click **Reload**, **Reload Data and Configuration** to reload the configuration.

Then select another row in the Payments table. You will see the window with row details:



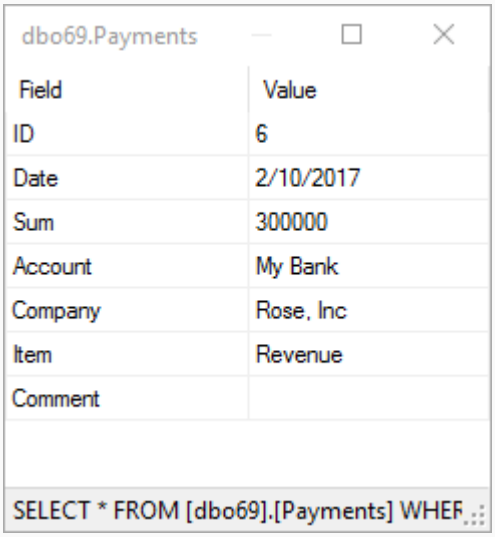
You may select another row. The window will stay on top and show related information.

_Transpose

You may transpose row to columns in the output window. Just add the **_Transpose** word in **TARGET_WORSHEET**:

TABLE_NAM	COLUMN_	EVENT_NAME	HANDLER	HAND	HANDLER_CODE	TARGET_WORKSHEET
Payments	Account	ValidationList	Accounts	VALUES	My Bank	
Payments	Item	ValidationList	Items	VALUES	Revenue,Expenses,Payroll,Taxes	
Payments	Company	ValidationList	Companies	TABLE	Company	
Payments		SelectionChange	Payments	TABLE		_Transpose

Switch to the **payments** worksheet and click **Reload**, **Reload Data and Configuration**. A new window looks like



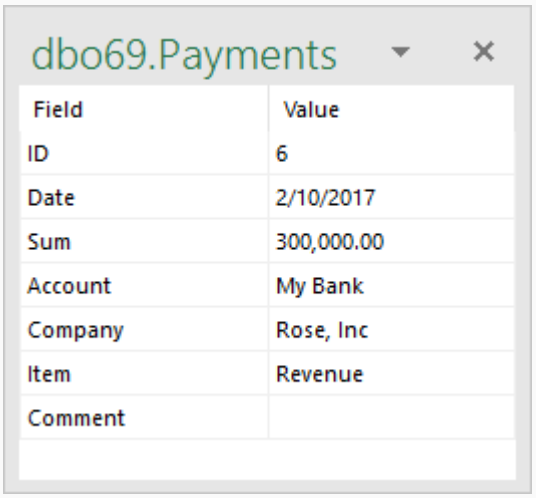
Task Panes

Use of detail windows is a good solution when you need windows that are always visible.

If you need context windows, you may use task panes. Just add the **_TaskPane** word in **TARGET_WORSHEET**:

TABLE_NAM	COLUMN_	EVENT_NAME	HANDLER	HAND	HANDLER_CODE	TARGET_WORKSHEET
Payments	Account	ValidationList	Accounts	VALUES	My Bank	
Payments	Item	ValidationList	Items	VALUES	Revenue,Expenses,Payroll,Taxes	
Payments	Company	ValidationList	Companies	TABLE	Company	
Payments		SelectionChange	Payments	TABLE		_TaskPane _Transpose

A new task pane window will look like



You may customize column formats using the context menu.

Also, you may dock task panes and turn them on/off using the **SaveToDB**, **Options**, **Show Task Panes** option.

Chapter 13. Context Menus

Detail windows and task panes discussed above are shown in the **SelectionChange** event.

We may add such queries and much more to the context menu.

Let's add the following configuration to the **EventHandlers** table (two lines at the bottom):

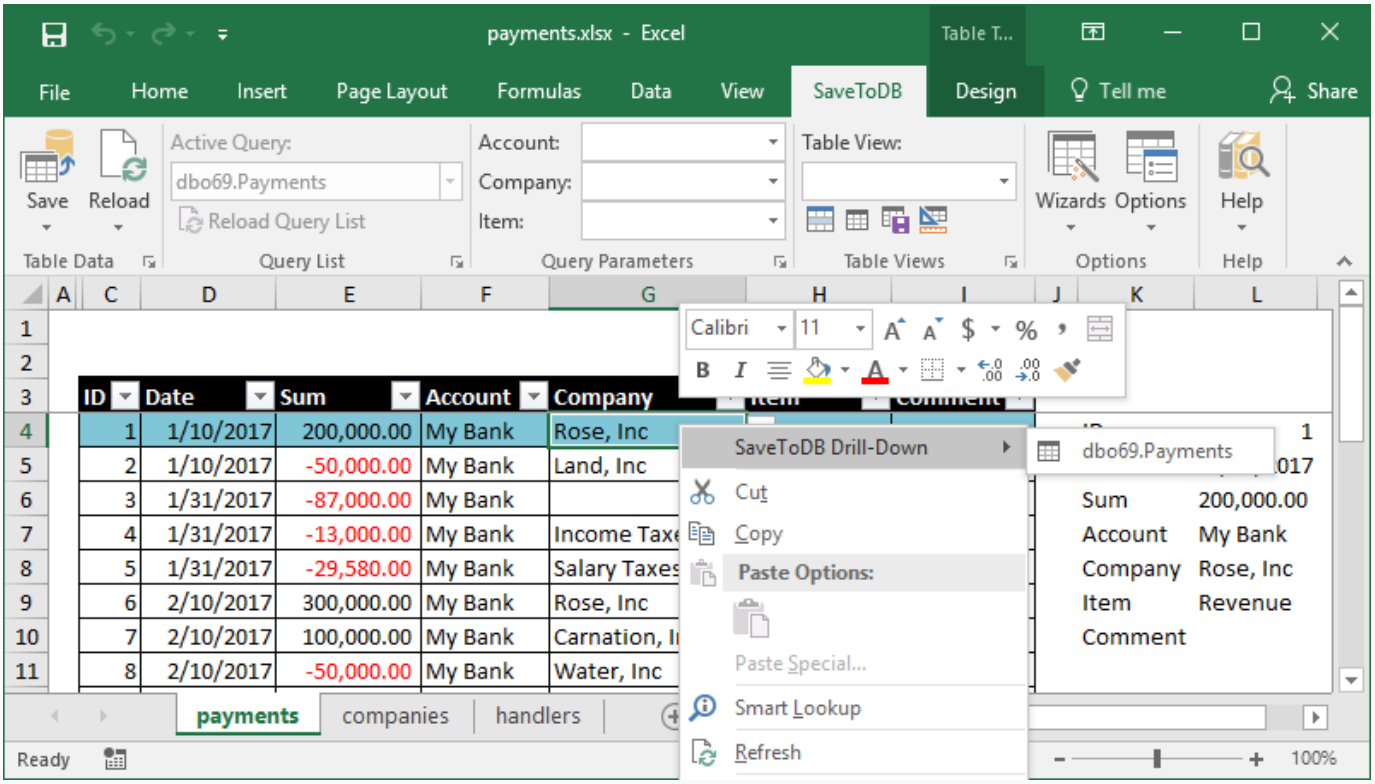
TABLE_NAM	COLUMN_	EVENT_NAME	HANDLER	HAND	HANDLER_CODE	TARGET_WORKSHEET
Payments	Account	ValidationList	Accounts	VALUES	My Bank	
Payments	Item	ValidationList	Items	VALUES	Revenue,Expenses,Payroll,Taxes	
Payments	Company	ValidationList	Companies	TABLE	Company	
Payments		SelectionChange	Payments	TABLE		_TaskPane _Transpose
Payments	Company	ContextMenu	Payments	TABLE	+Date,+Sum,Account,@Company,Item,Comment	
Payments	Item	ContextMenu	Payments	TABLE	+Date,+Sum,Account,Company,@Item,Comment	

Contrary to the SelectionChange handler, we have specified columns. So, context menus are shown for columns.

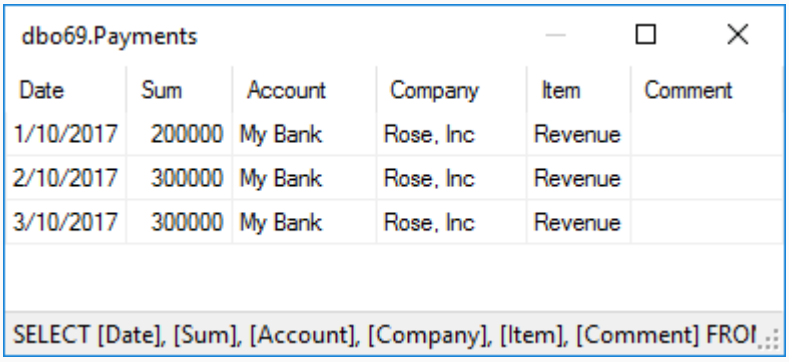
Also, we have directly specified select columns in the **HANDLER_CODE**. The plus sign defines the sort order.

@Company and @Item fields are used in the WHERE clause to filter output data using current row values.

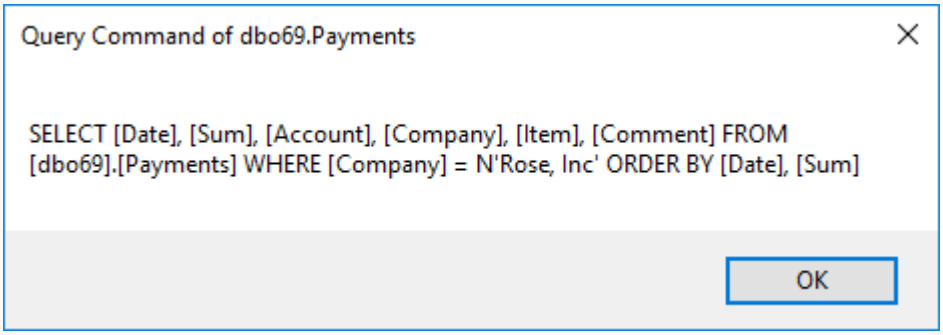
Let's select the **payments** sheet, click **Reload**, **Reload Data and Configuration**, and right click on a company cell:



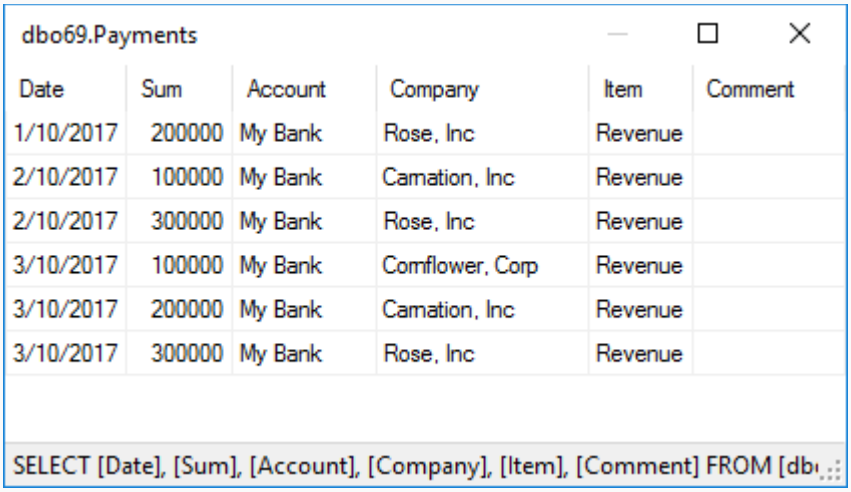
When we click on the **dbo69.Payments** item, the add-in shows the window like this:



You see rows for Rose, Inc. You may click on the status line to check the generated SQL:



Let's right click on any Revenue item cell and click on the **dbo69.Payments** item. We will see item payments.



As we learned above, the **SelectionChange** and **ContextMenu** handlers are nearly the same.

They get parameters from the columns of the active row. They differ by the **EVENT_NAME** configuration.

However, the **ContextMenu** handlers support additional configuration parameters like **MENU_ORDER** and **EDIT_PARAMETERS** and allow using much more handler types in the **HANDLER_TYPE** field.

You may configure executing stored procedures, SQL codes, macros, CMD commands, opening URLs.

Use the **MenuSeparator** type to separate menu items.

Chapter 14. Actions Menus

The action menus are located at the ribbon and have nearly the same features as context menus.

However, users may execute actions when the active cell is outside of the table. So, they may have no context.

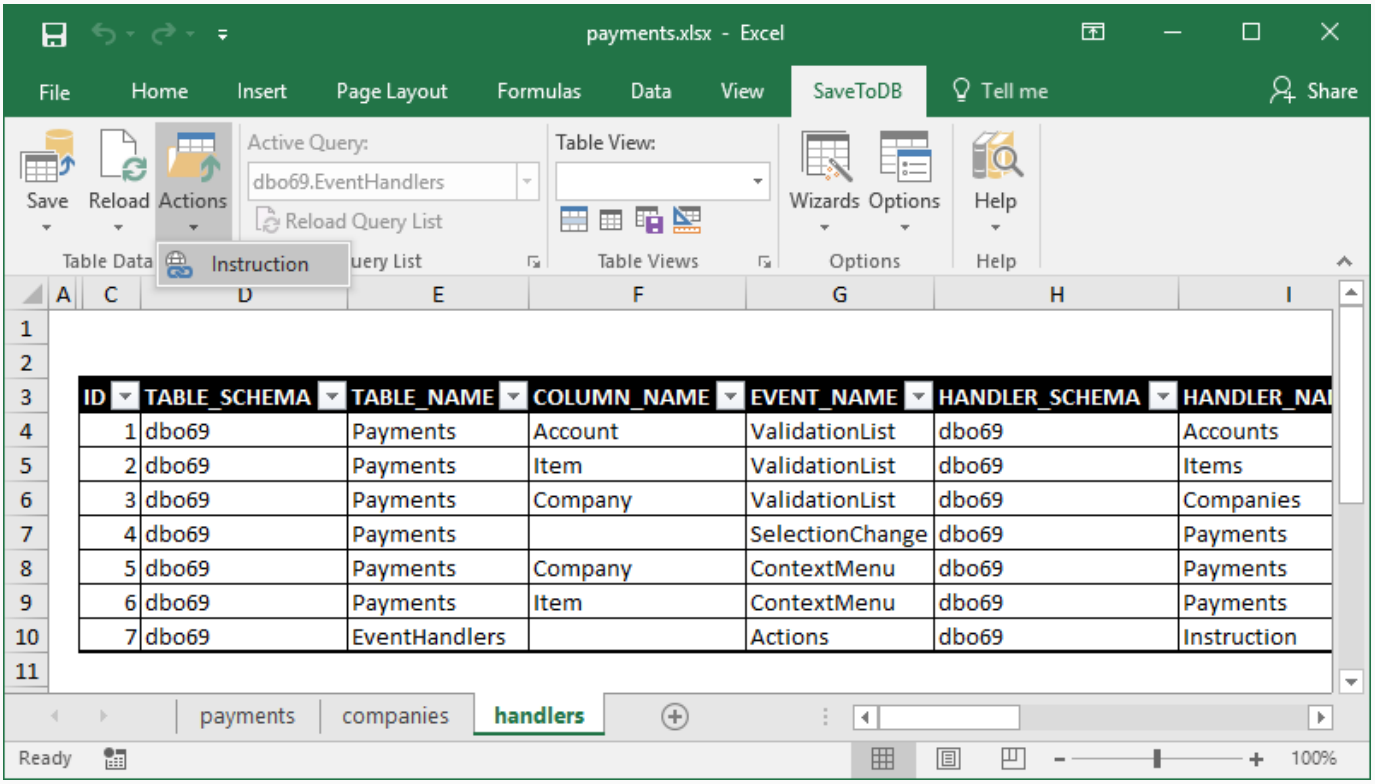
Let's add an URL to the **Actions** menu for the **EventHandlers** table. Add the line shown at the bottom:

TABLE_NAM	COLUMN_	EVENT_NAME	HANDLER	HAND	HANDLER_CODE	TARGET_WORKSHEET
Payments	Account	ValidationList	Accounts	VALUES	My Bank	
Payments	Item	ValidationList	Items	VALUES	Revenue,Expenses,Payroll,Taxes	
Payments	Company	ValidationList	Companies	TABLE	Company	
Payments		SelectionChange	Payments	TABLE		_TaskPane _Transpose
Payments	Company	ContextMenu	Payments	TABLE	+Date,+Sum,Account,@Company,Item,Comment	
Payments	Item	ContextMenu	Payments	TABLE	+Date,+Sum,Account,Company,@Item,Comment	
EventHandlers		Actions	Instruction	HTTP	https://www.savetodb.com/savetodb/configuring-event-handlers.htm	

Complete table:

ID	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	EVENT_NAME	HANDLER_SCHEMA	HANDLER_NAME	HANDLER_TYPE	HANDLER_CODE	TARGET_WORKSHEET	MENU_ORDER
1	dbo69	Payments	Account	ValidationList	dbo69	Accounts	VALUES	My Bank		
2	dbo69	Payments	Item	ValidationList	dbo69	Items	VALUES	Revenue,Expenses,Payroll,Taxes		
3	dbo69	Payments	Company	ValidationList	dbo69	Companies	TABLE	Company		
4	dbo69	Payments		SelectionChange	dbo69	Payments	TABLE		_TaskPane _Transpose	
5	dbo69	Payments	Company	ContextMenu	dbo69	Payments	TABLE	+Date,+Sum,Account,@Company,Item,Comment		
6	dbo69	Payments	Item	ContextMenu	dbo69	Payments	TABLE	+Date,+Sum,Account,Company,@Item,Comment		
7	dbo69	EventHandlers		Actions	dbo69	Instruction	HTTP	https://www.savetodb.com/savetodb/configuring-event-handlers.htm		

Reload data and configuration. You will see the Actions menu with the Instruction link.



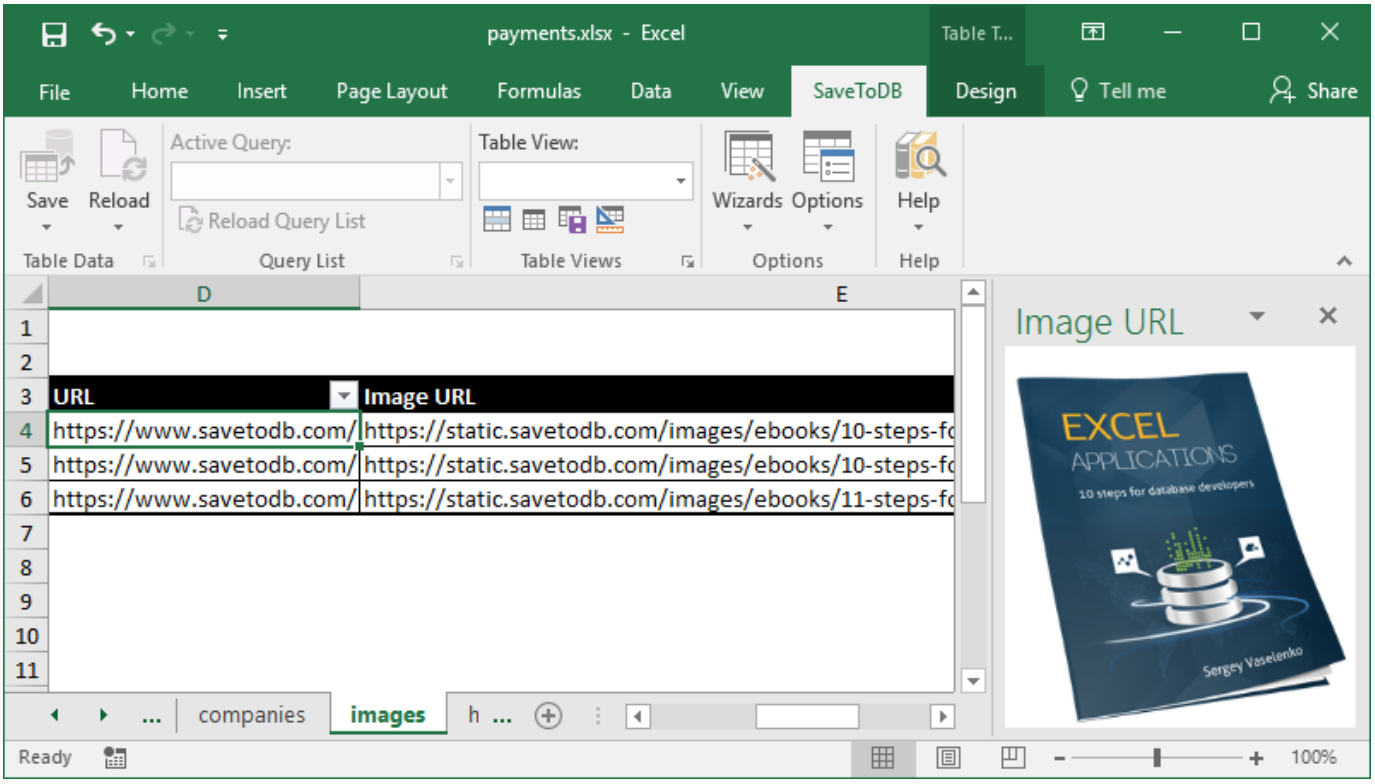
ID	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	EVENT_NAME	HANDLER_SCHEMA	HANDLER_NAI
1	dbo69	Payments	Account	ValidationList	dbo69	Accounts
2	dbo69	Payments	Item	ValidationList	dbo69	Items
3	dbo69	Payments	Company	ValidationList	dbo69	Companies
4	dbo69	Payments		SelectionChange	dbo69	Payments
5	dbo69	Payments	Company	ContextMenu	dbo69	Payments
6	dbo69	Payments	Item	ContextMenu	dbo69	Payments
7	dbo69	EventHandlers		Actions	dbo69	Instruction

The **Actions** menu is a good place for documentation links and common tasks related to a table, not to a row.

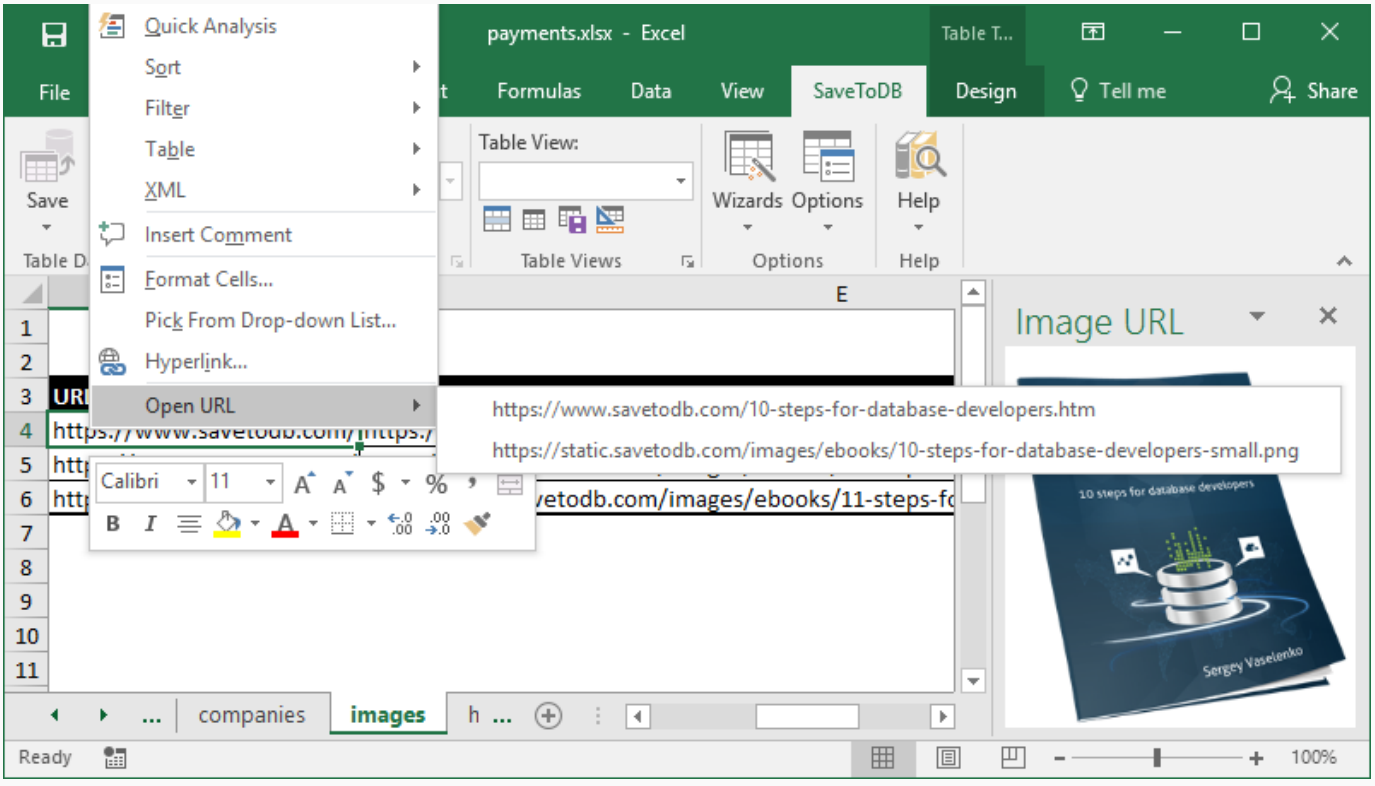
Chapter 15. Image URLs

You may add URLs and image URLs to your tables. This does not require any configuration.

The add-in automatically loads and shows images in task panes. You may dock task panes like this:



Also, the add-in adds the **Open URL** context menu if a row contains URLs.



Chapter 16. LastModified and UserName

In this chapter, we will add columns that show times and user names of the latest updates.

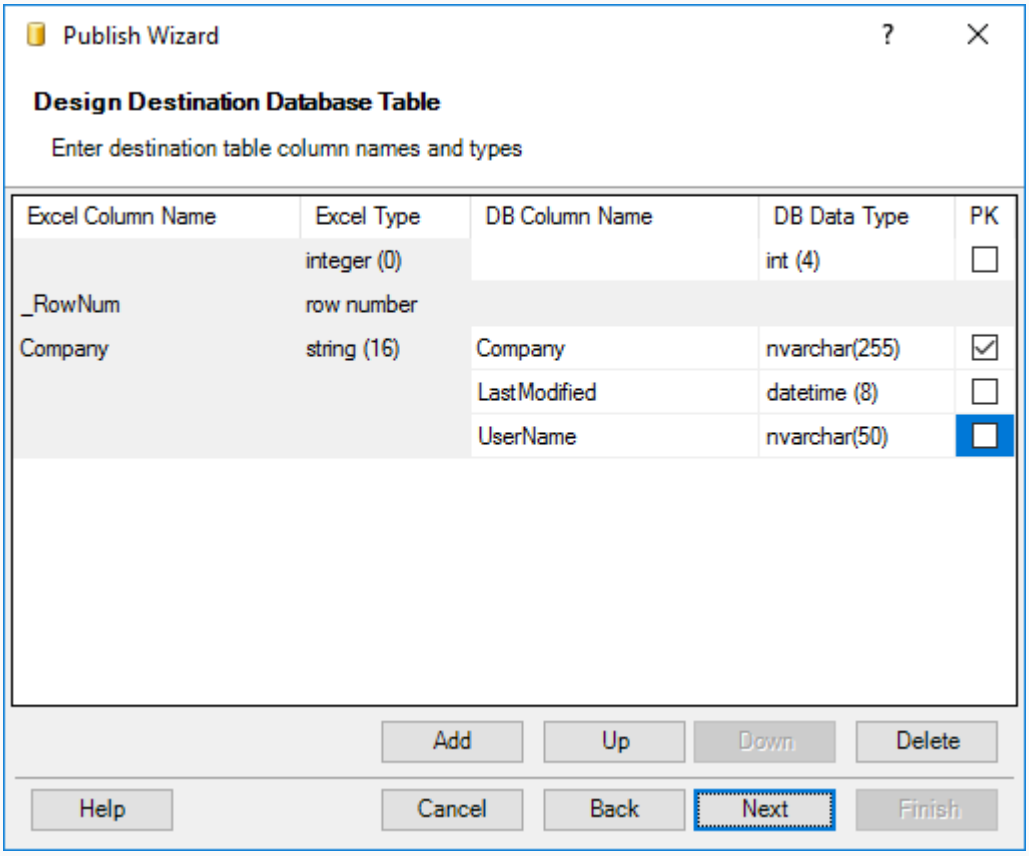
This helps team members to understand who and when changed data.

You may use the following solution in a friendly environment. Otherwise, you have to use database triggers.

Creating New Columns

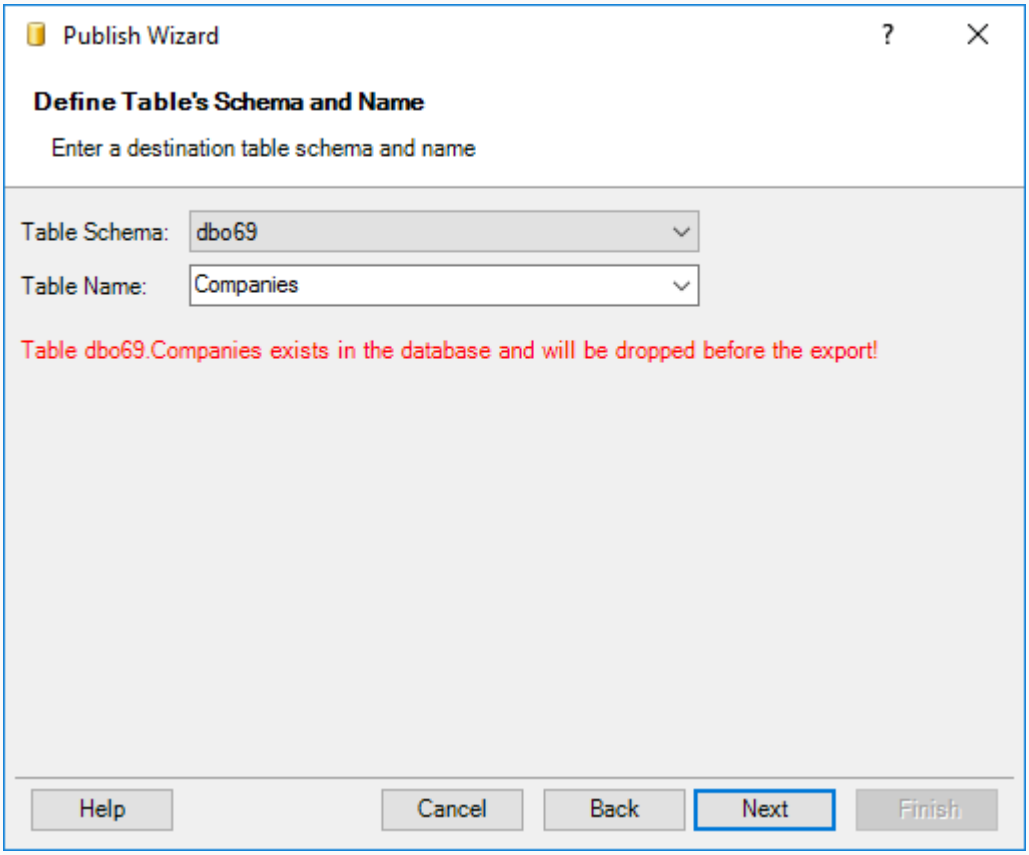
Let's select the **companies** worksheet, the **Companies** table, and launch **Publish Wizard**.

Follow the steps. At the design tab, add the following columns:

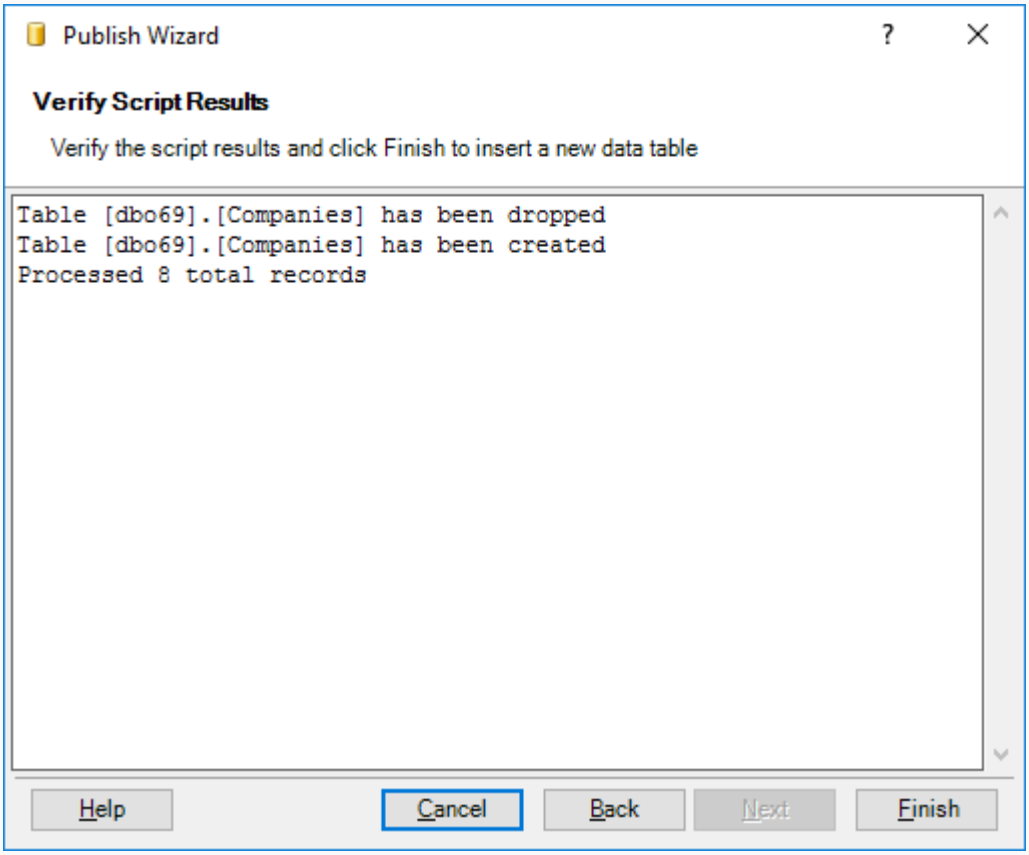


You may change column names and data types in your solutions.

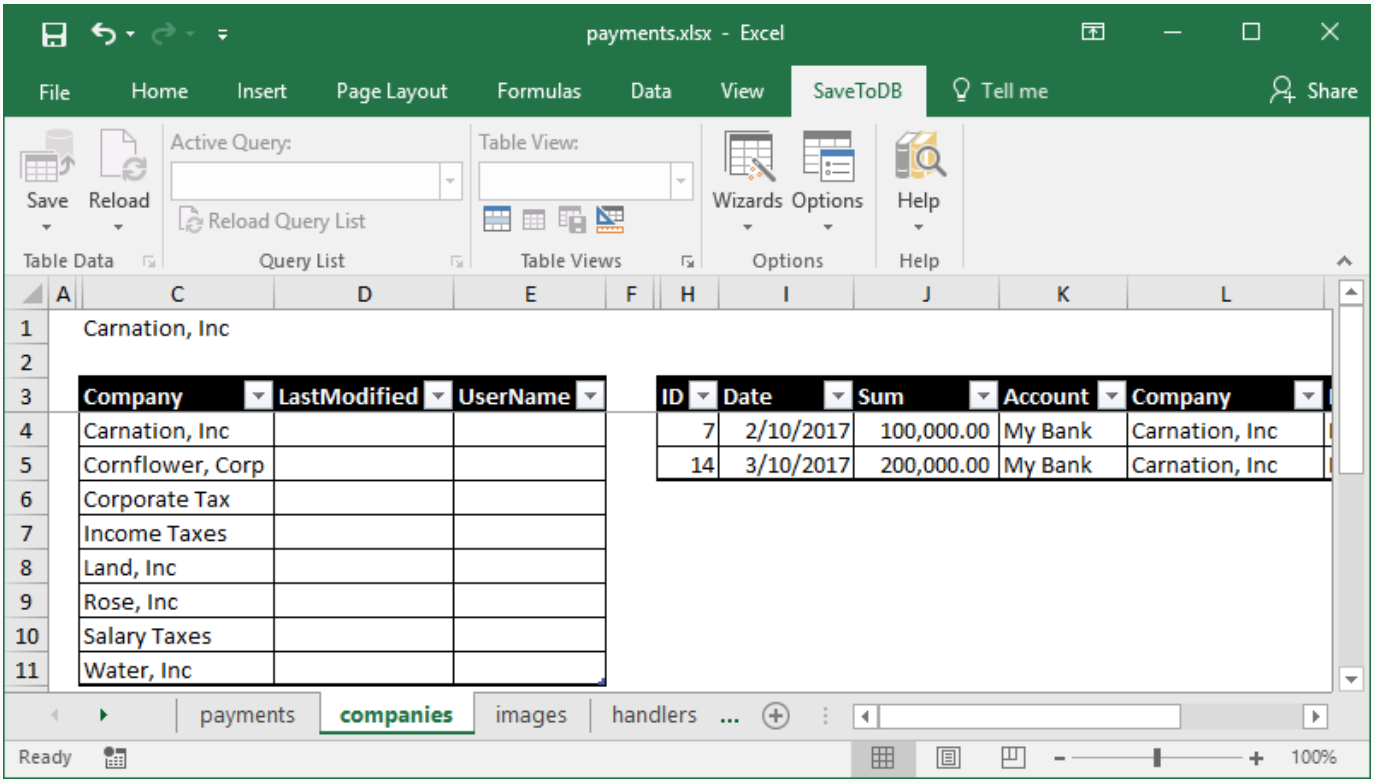
Specify the same table name at the following step and execute the SQL script to recreate the table.



Do not click **Finish** to skip the creating a new table in the workbook. Click **Cancel** at this step.



Let's reload data and configuration on the existing worksheet. You will see new fields in the **Companies** table:



Event Handlers

Let's add the following handlers for the Companies table in the **EventHandlers** table:

TABLE_NAM	COLUMN_	EVENT_NAME	HANDLER	HAND	HANDLER_CODE
Companies	LastModified	FormulaValue			=NOW()
Companies	UserName	FormulaValue			=UserName()
Companies	LastModified	DoNotChange			
Companies	UserName	DoNotChange			

Complete table:

ID	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	EVENT_NAME	HANDLER_SCHEMA	HANDLER_NAME	HANDLER_TYPE	HANDLER_CODE
8	dbo69	Companies	LastModified	FormulaValue				=NOW()
9	dbo69	Companies	UserName	FormulaValue				=UserName()
10	dbo69	Companies	LastModified	DoNotChange				
11	dbo69	Companies	UserName	DoNotChange				

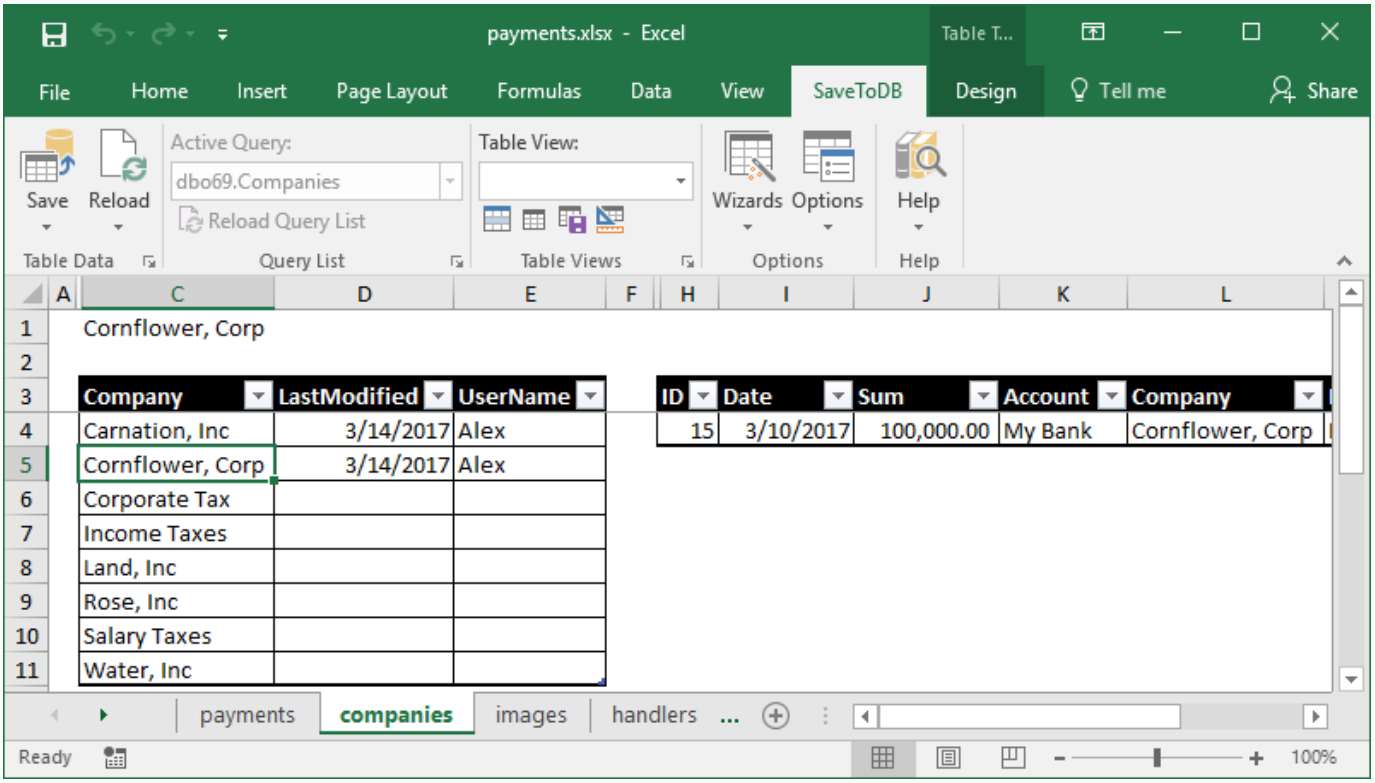
The **FormulaValue** handlers update cell values on row changes using the specified formulas.

=NOW() is an Excel formula. You may use this technique for other formulas too.

=UserName() is a SaveToDB add-in formula. You may use =DomainUserName() also.

The **DoNotChange** handlers prevent changes in the LastModified and UserName columns.

Let's switch to the **companies** worksheet and change a couple of cells.



As you may see, the add-in updates the values in the LastModified and UserName columns.

This technique is very simple. You may add this feature to your tables quickly.

However, note that such columns may be updated with any values directly in a database using SQL.

Chapter 17. Integration with Other Apps

You may load data from external data sources and save the data into your tables.

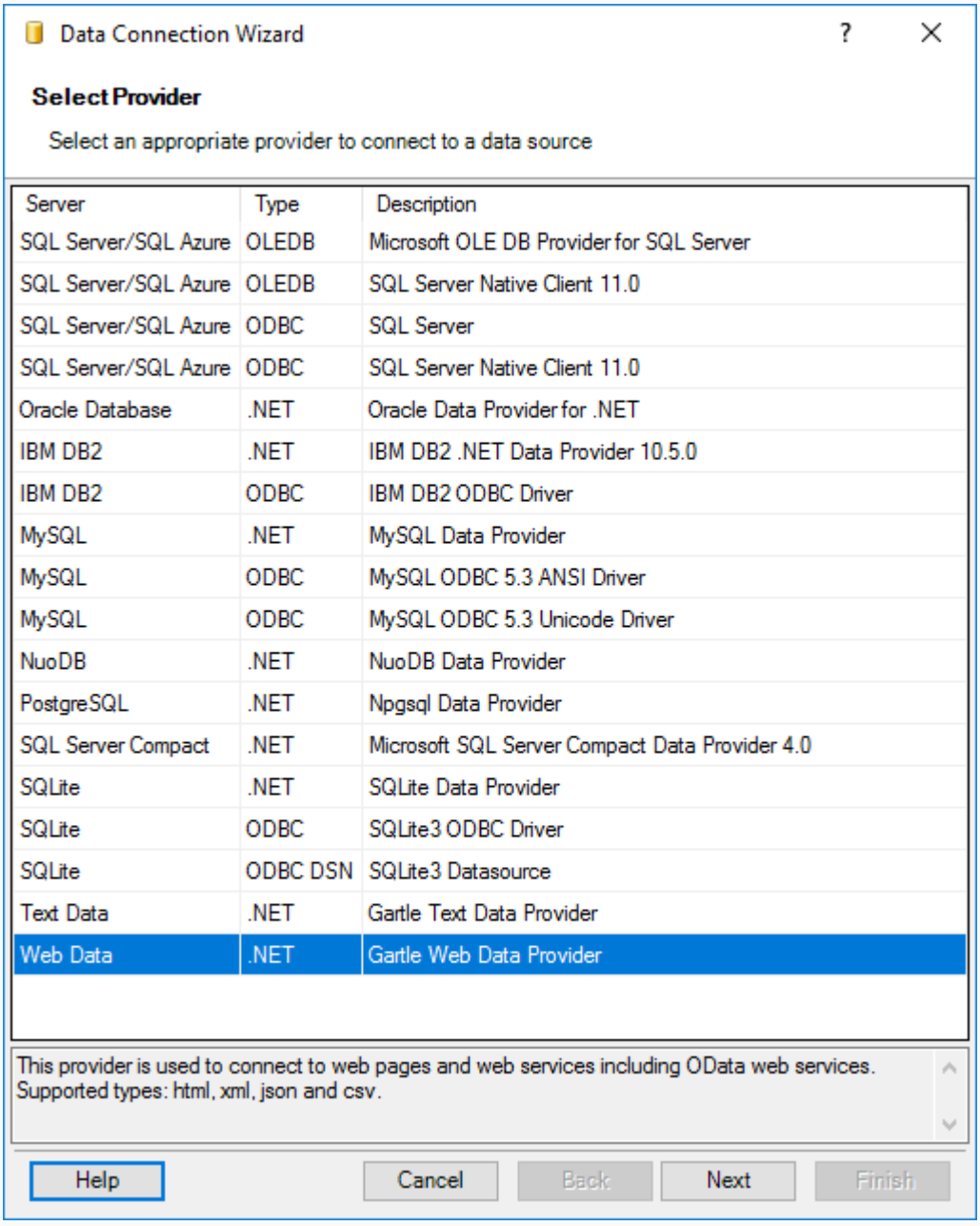
You may load data from databases, web pages, and text files using the add-in, or from other sources using Excel or other tools like PowerQuery.

External Data Sources

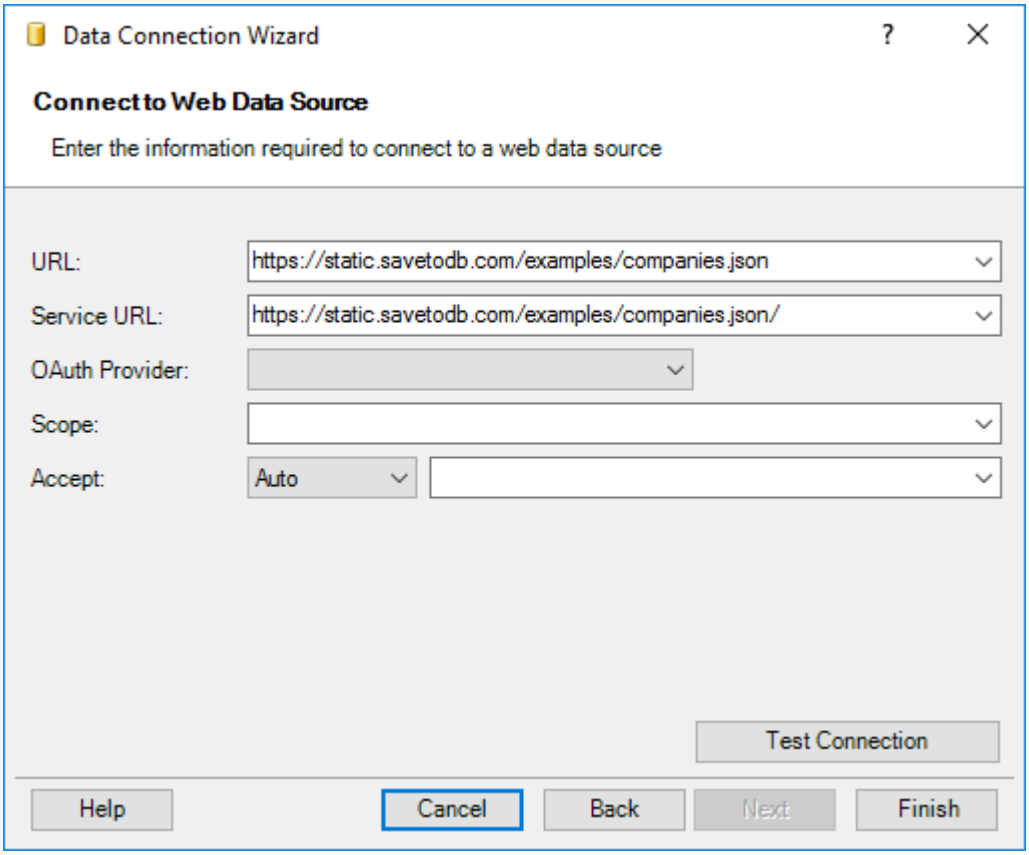
Let's create a worksheet like **integration** and run **Data Connection Wizard**.

You may connect to any supported data source including text files and web data.

Let's select the **Web Data** provider.



At the following step, let's paste the test **URL** to load data from a JSON data source:



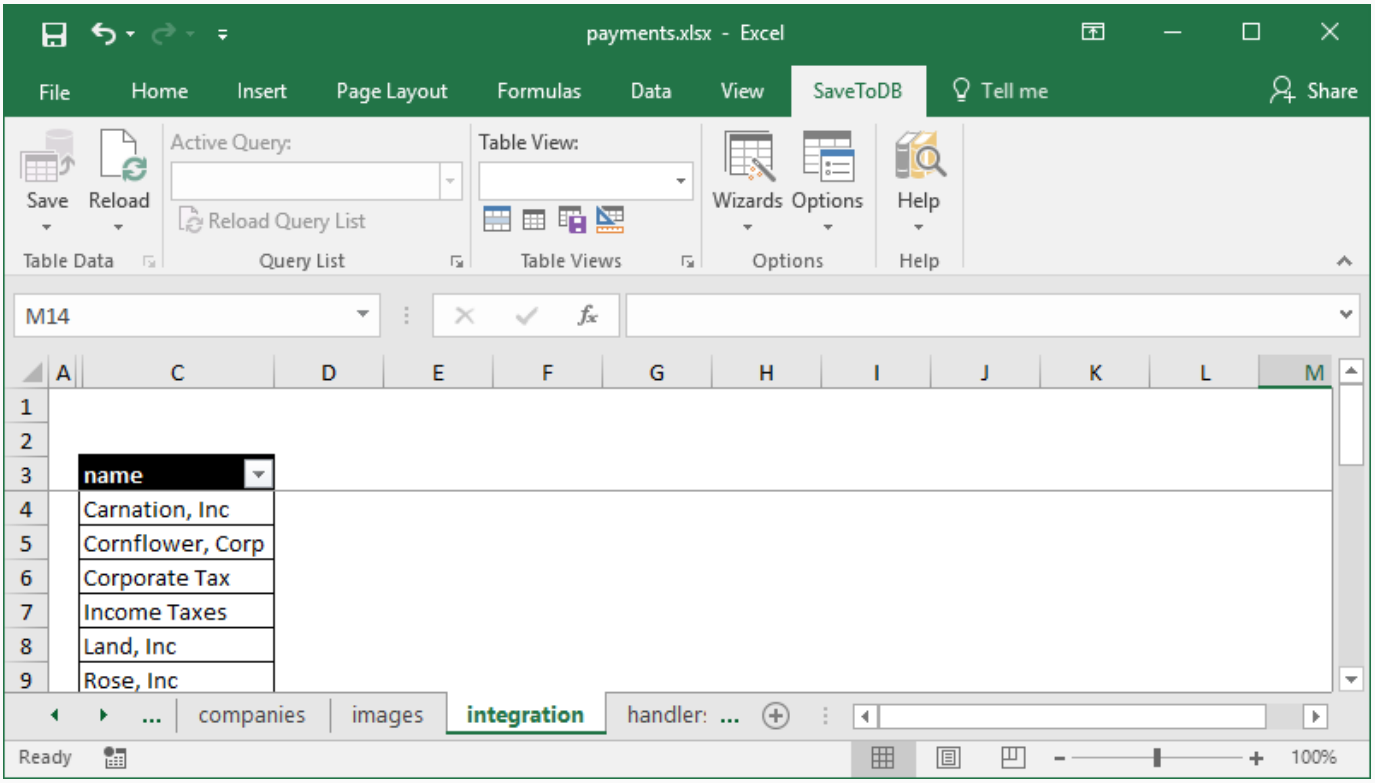
You may test the following URLs:

- https://static.savetodb.com/examples/companies.json
- https://static.savetodb.com/examples/companies.xml
- https://static.savetodb.com/examples/companies.html
- https://static.savetodb.com/examples/companies.csv

The SaveToDB add-in automatically parses the pages and retrieves the meaningful data.

You may also customize the parsers. Refer to the documentation.

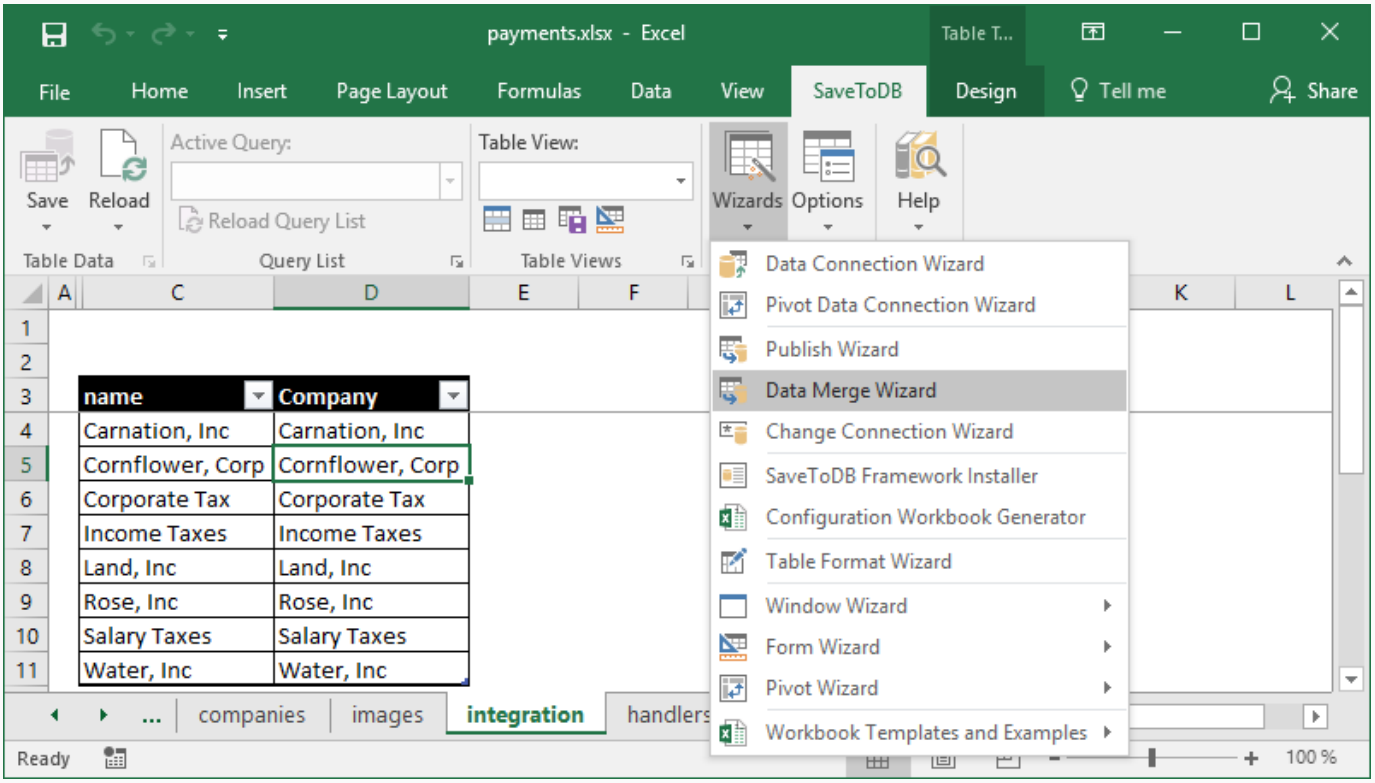
Let’s insert the new connected table at cell B3. You may refresh the data anytime.



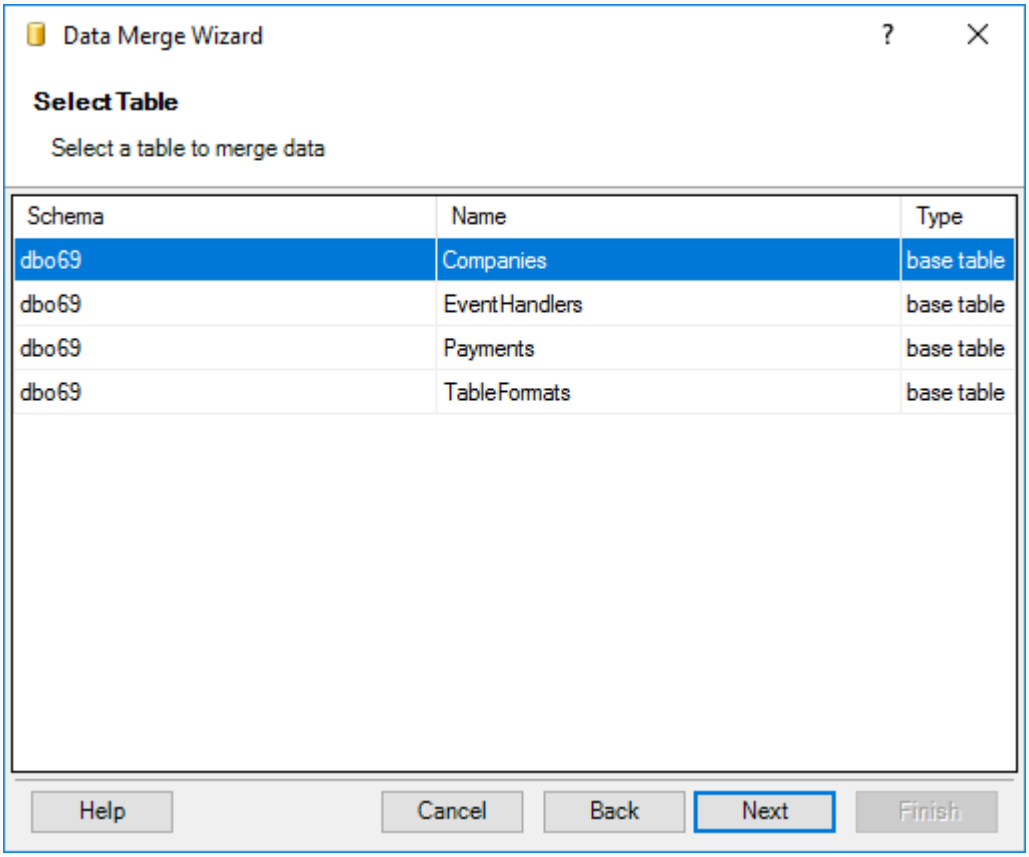
Data Merge Wizard

The external data source contains the **name** column. Our **dbo69.Companies** table contains the **Company** column.

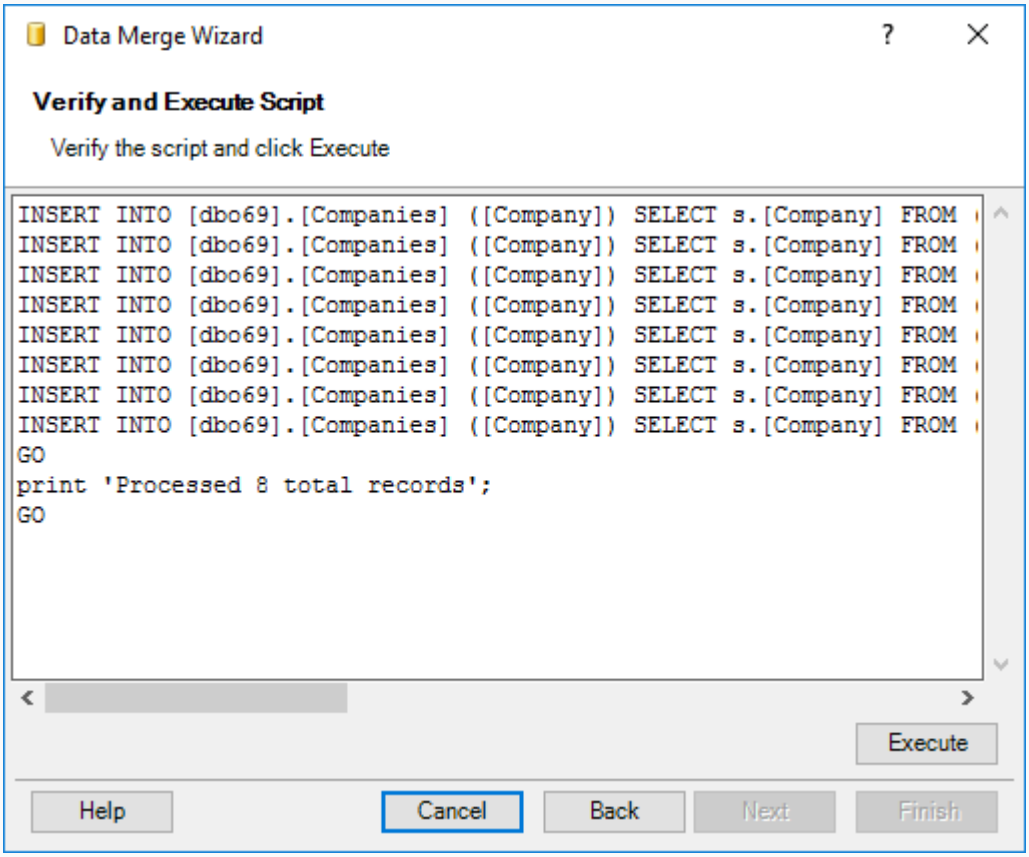
Let’s add the target **Company** column using the formula like =[@name] and run **Data Merge Wizard**.



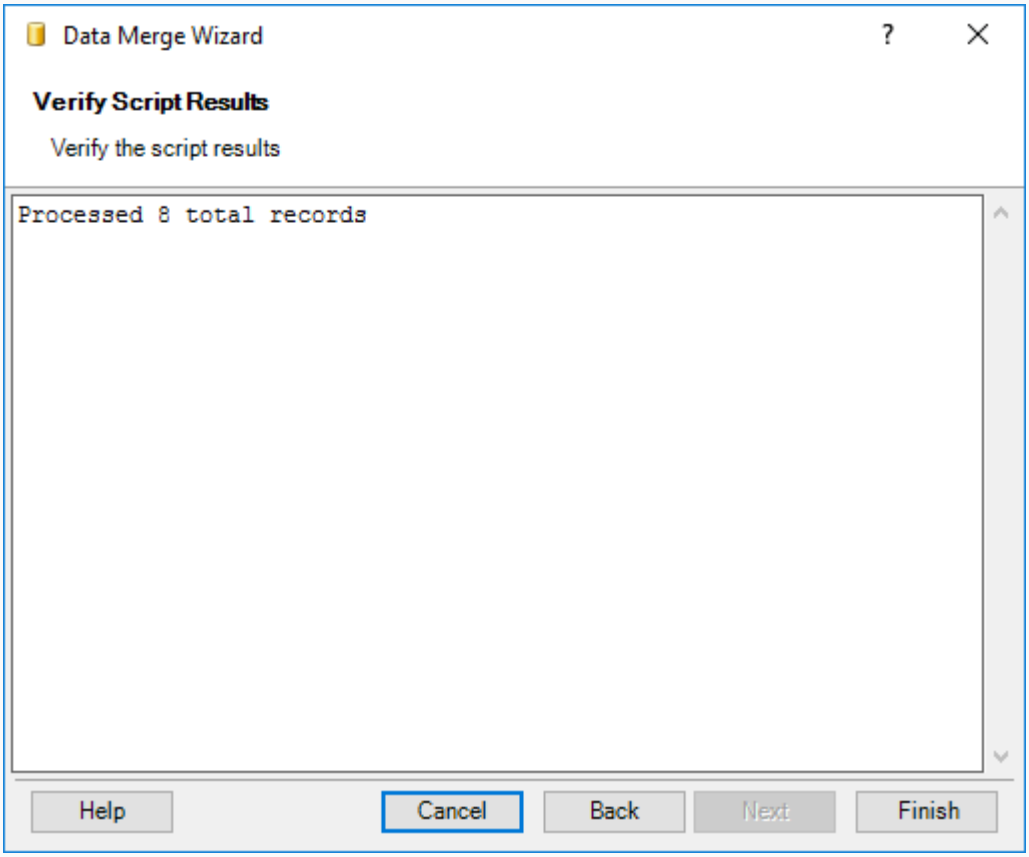
Follow wizard steps and select the target **dbo69.Companies** table at the following step:



You may see the generated SQL code used to update the underlying table in a merge mode. Click **Execute**.

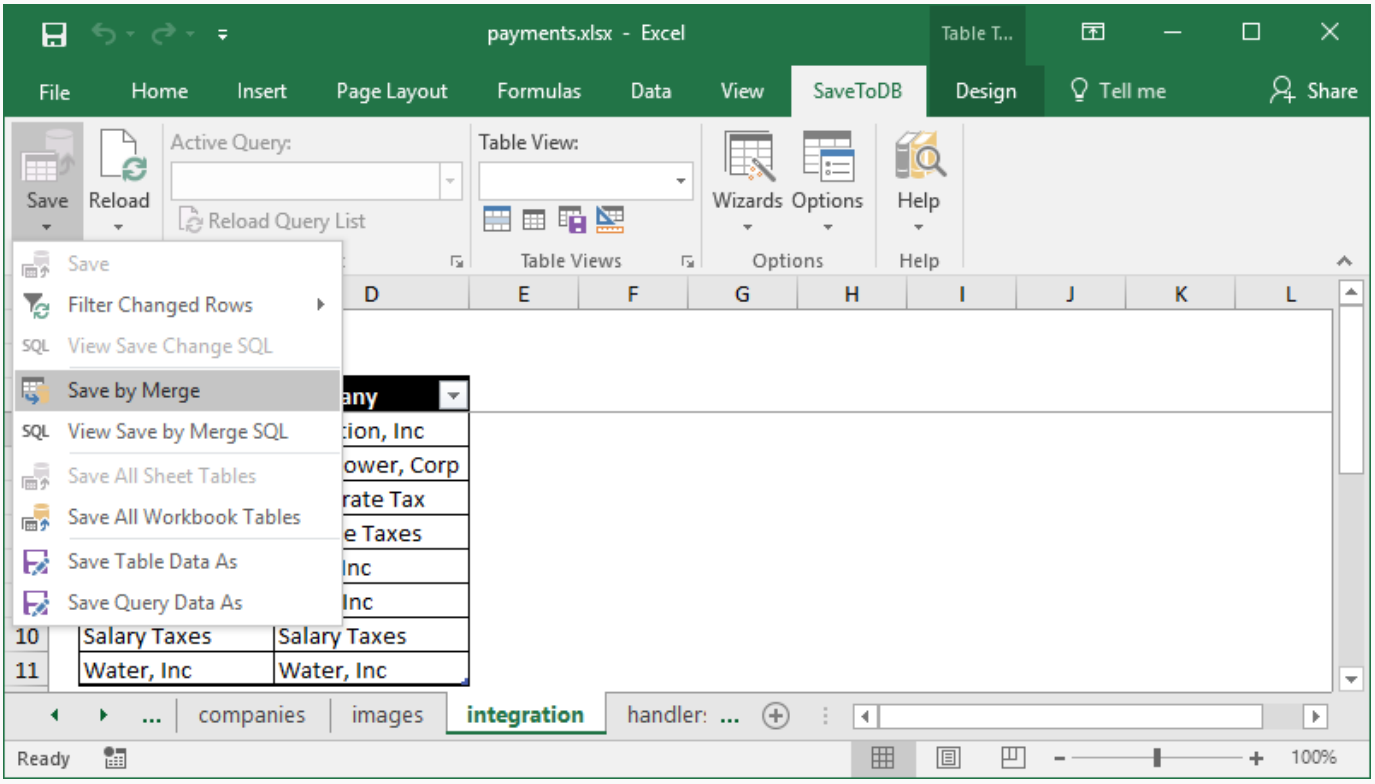


At this step, you see the execution result. The add-in has updated data. Click **Finish** to exit the wizard.



Save by Merge

The add-in saves the merge configuration. So, later you may use the **Save by Merge** item to have the same results.



Chapter 18. Permission Management

Managing permissions is an important part in multi-user environment.

Your IT staff manages permissions at the server and database levels. This is their tasks.

As a business user, you may manage permissions related to your applications.

And, the best news, you may do this for SQL Server in Excel using the SaveToDB add-in application.

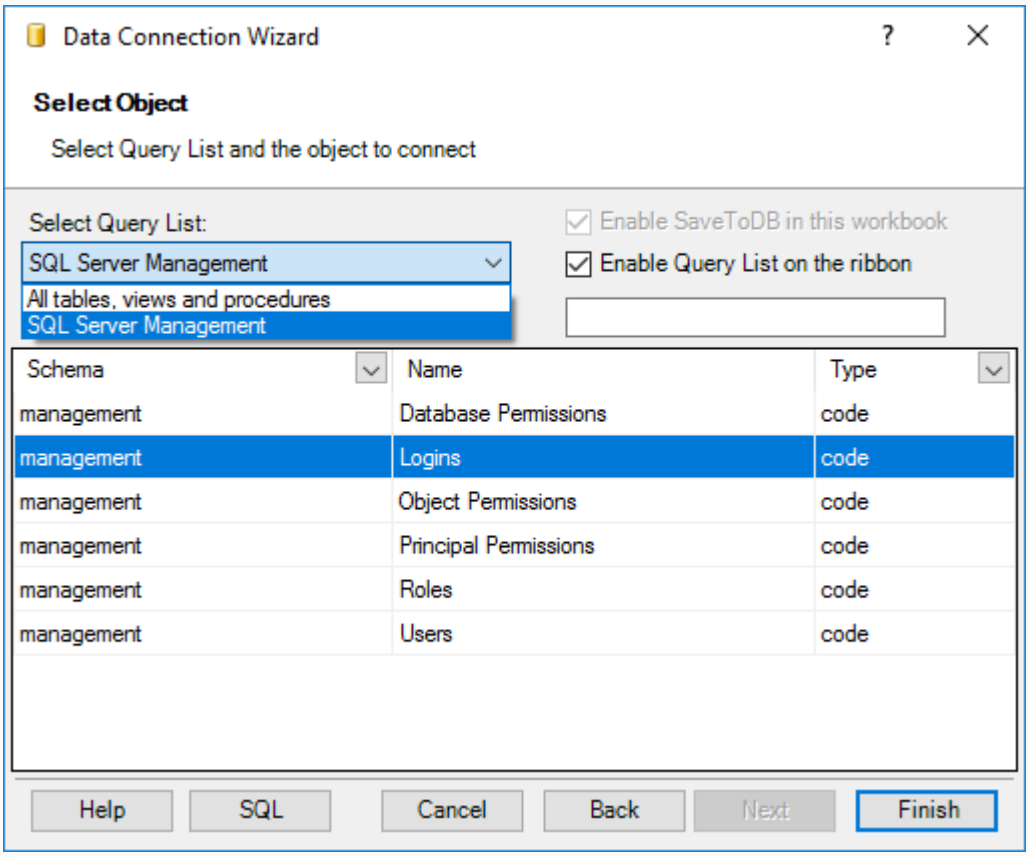
This application is a good example, also, of what you and your developers may do with the SaveToDB add-in.

SQL Server Management Application

Create the **permissions** worksheet and run **Data Connection Wizard**. Connect to your database.

At the following step, select **SQL Server Management** in the **Select Query List** field,

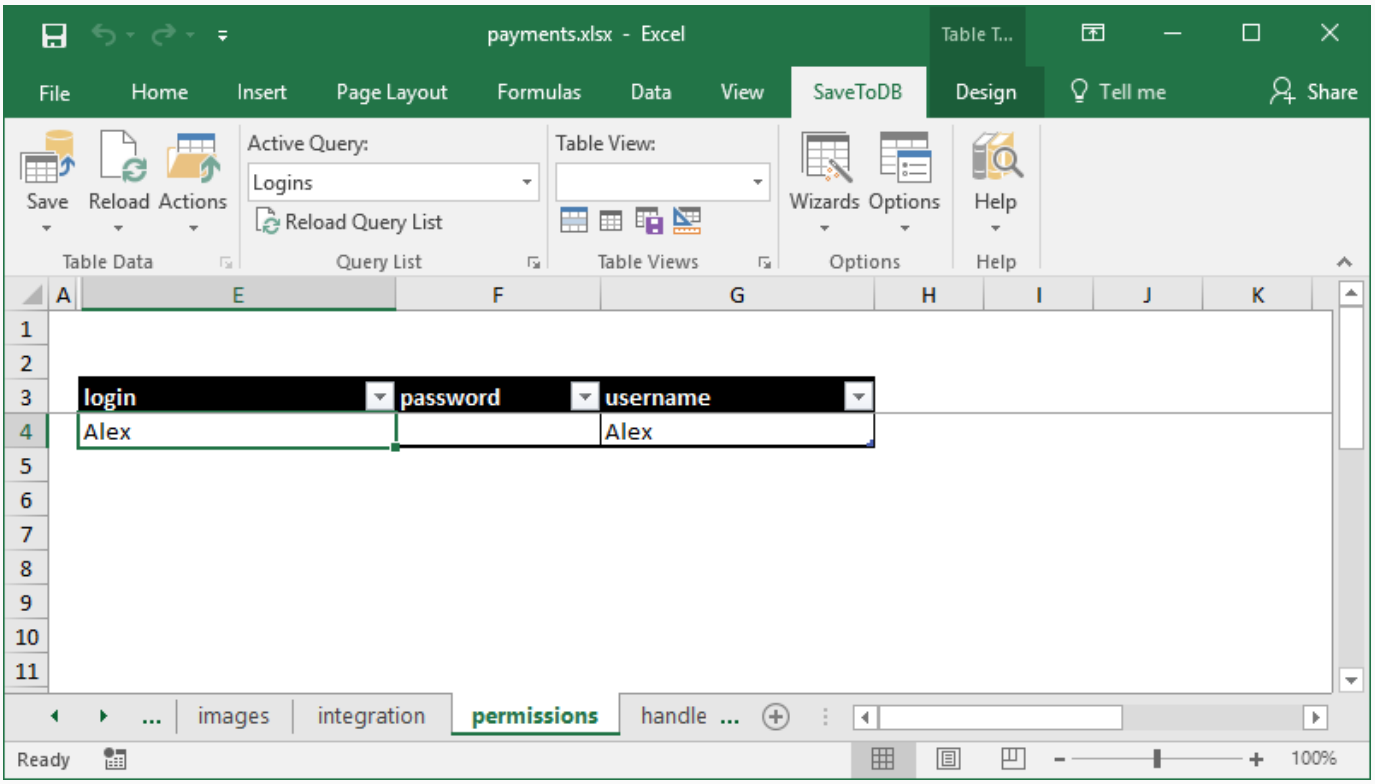
leave **Enable Query List on the ribbon** checked, and select **Logins**.



Click **Finish** and insert a new table at cell B3.

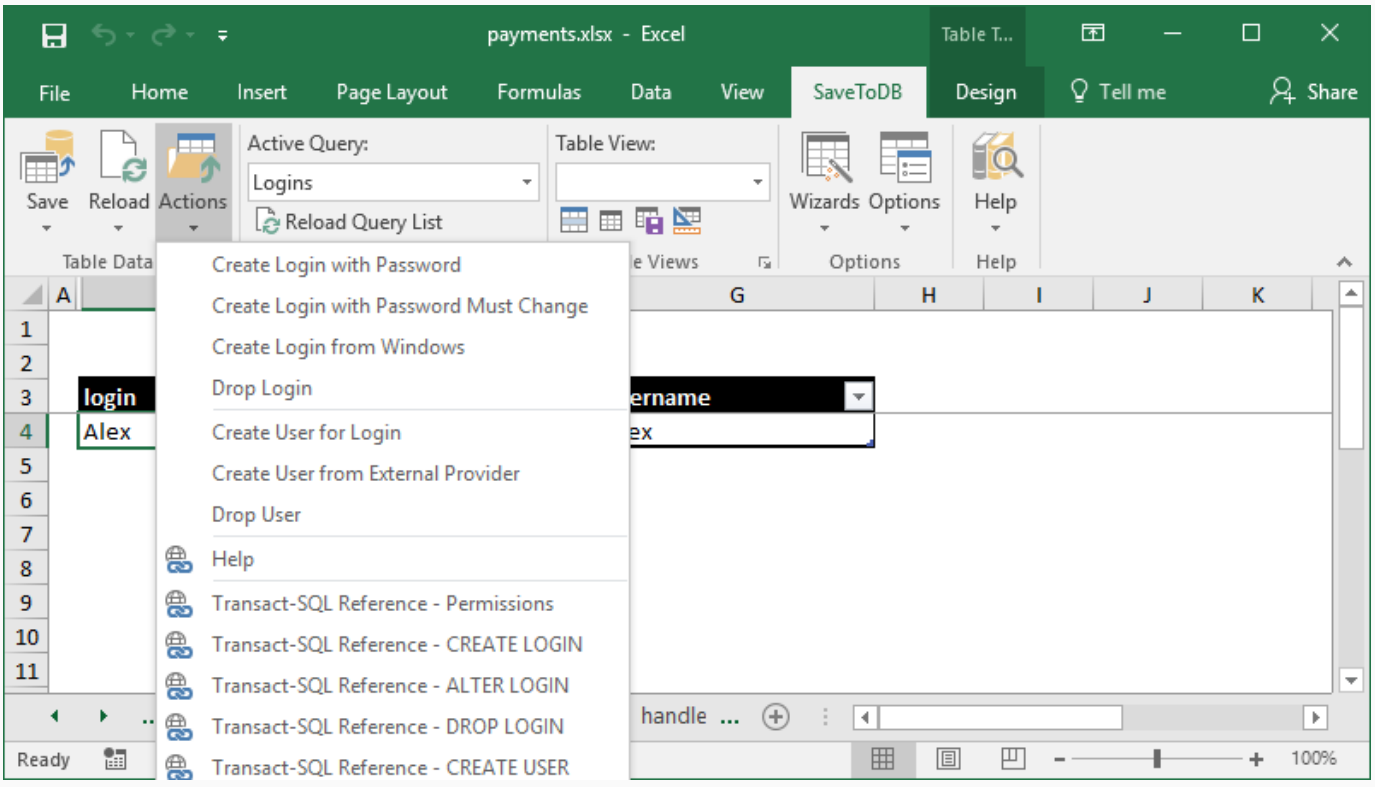
Logins

This table allows managing logins, passwords, and usernames. Just change the data and click **Save**.



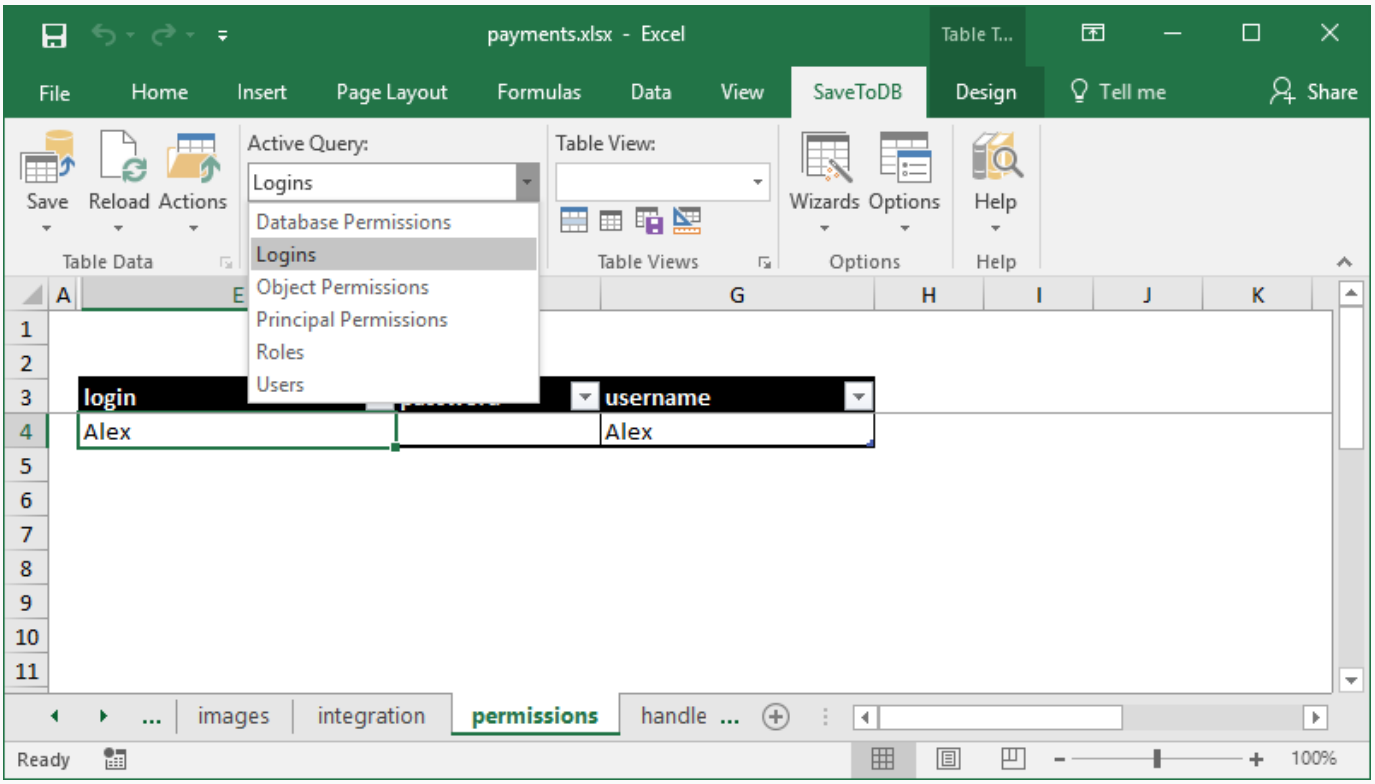
Usually, business users see the personal login only and cannot change any value.

The **Actions** menu contains useful actions and helpful links for database administrators.



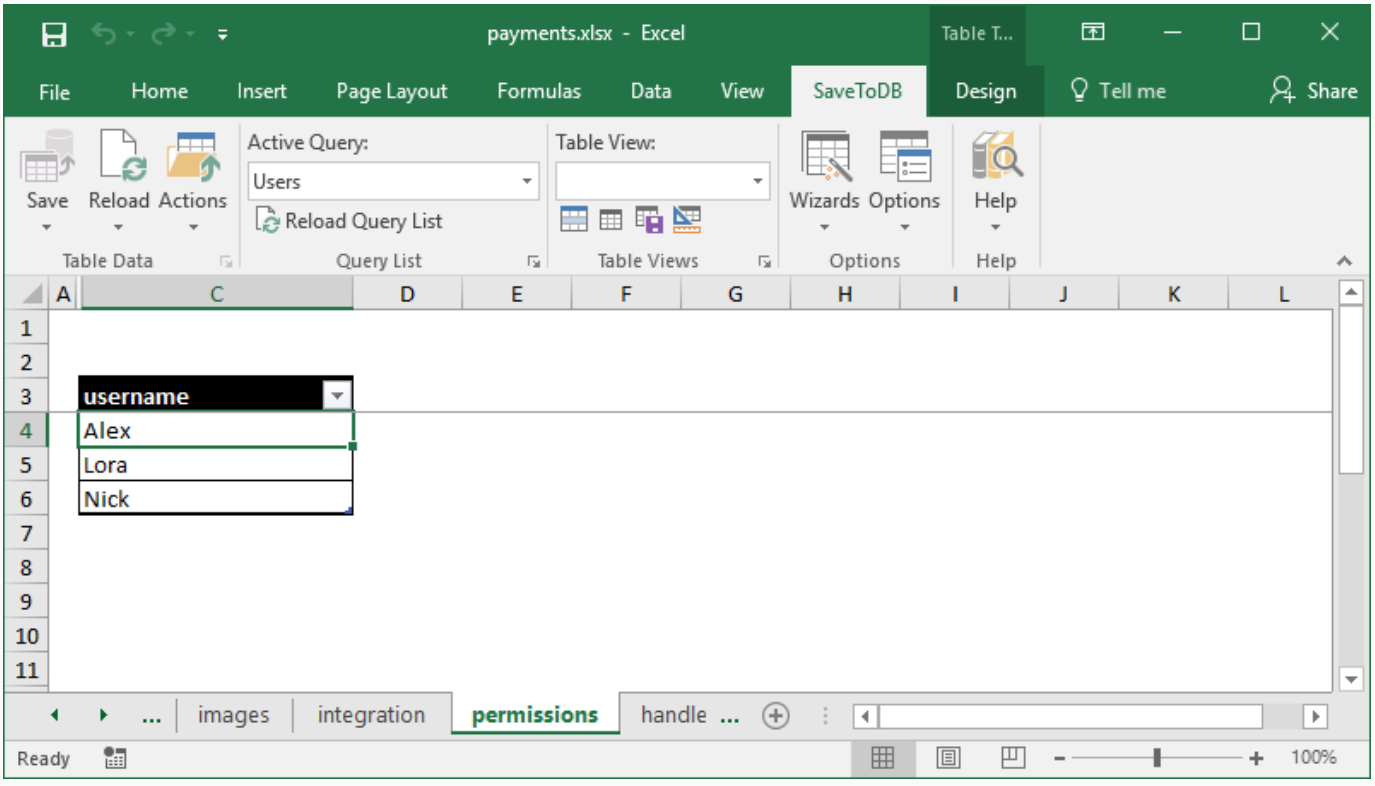
Query List

The ribbon **Query List** allows changing query objects using a single worksheet for multiple tasks:



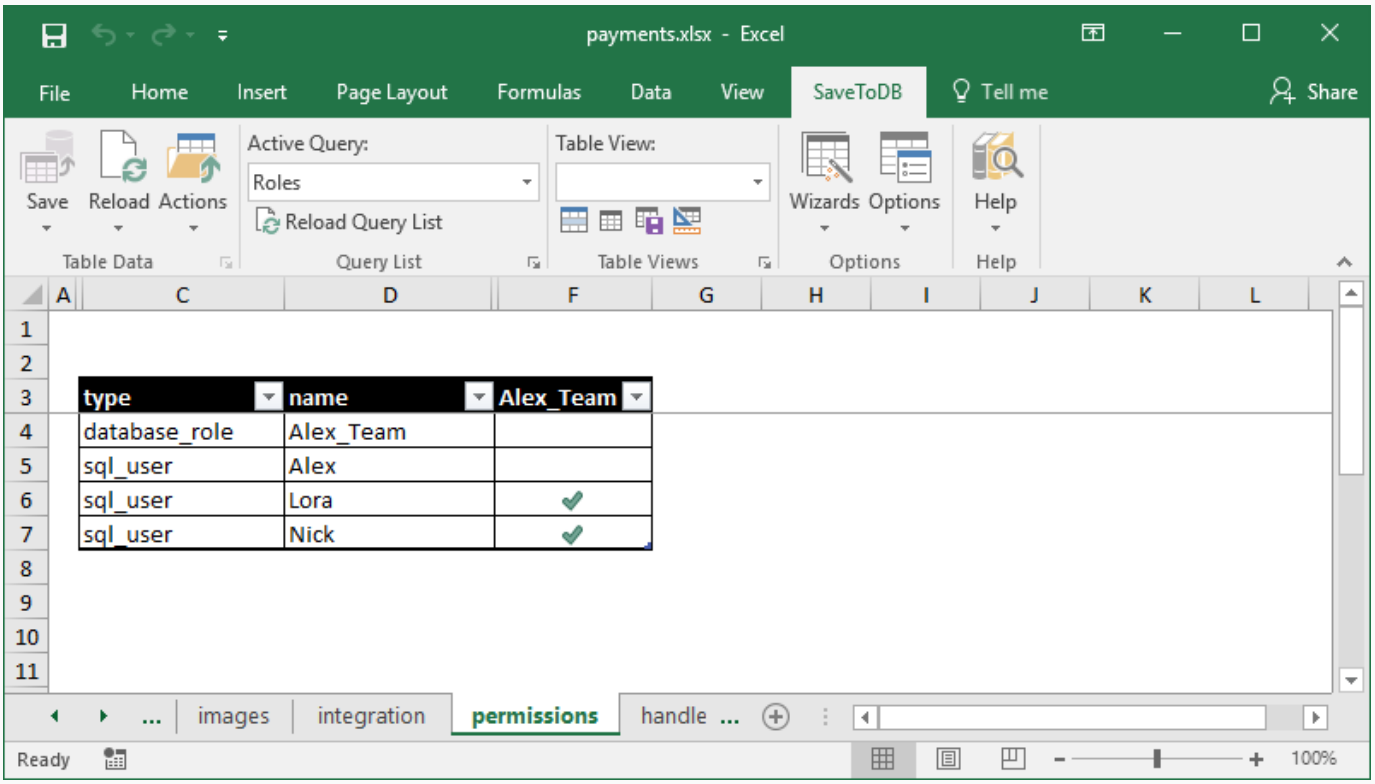
Users

The **Users** table shows users that you may manage. Usually, business users have no permissions to change.



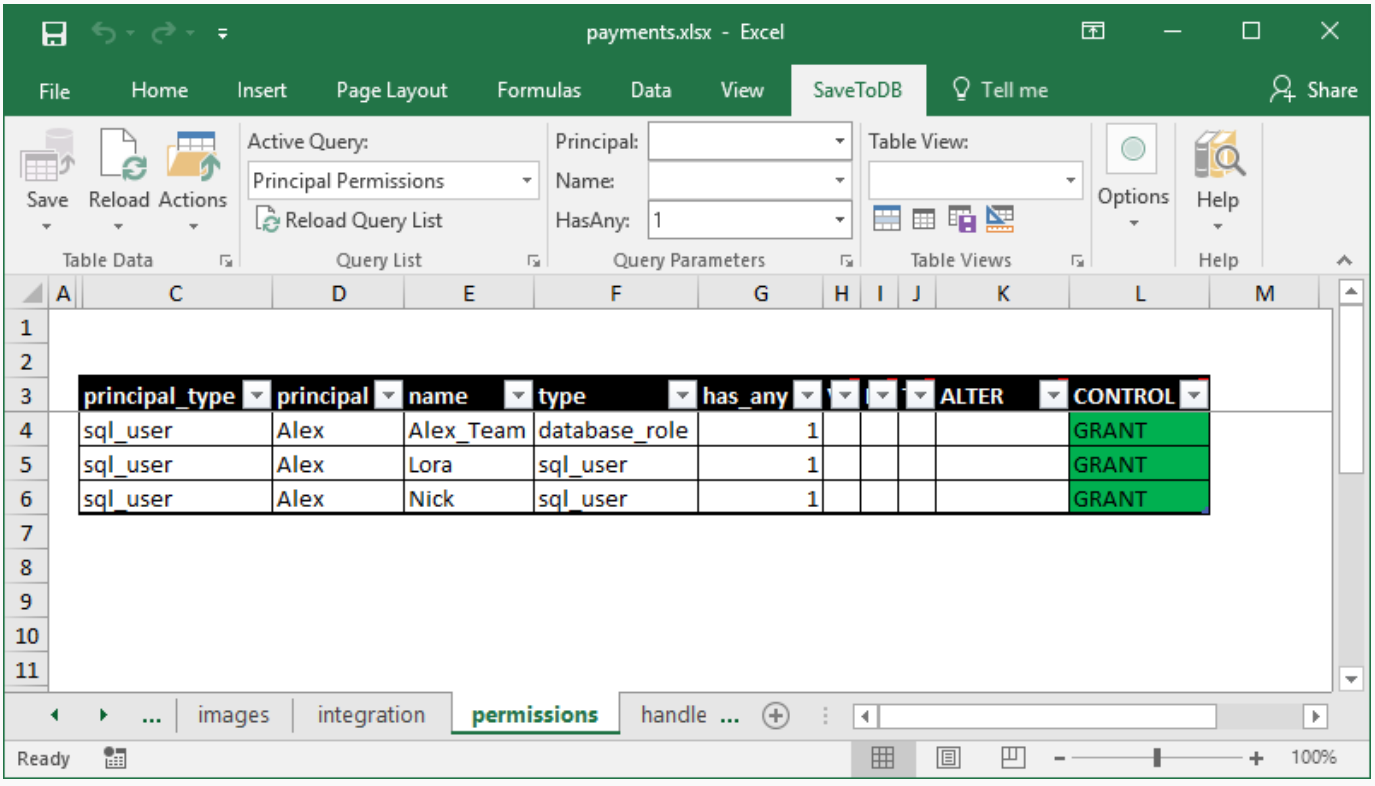
Roles

This table allows checking and changing user membership in roles. Just set 1 or clear the value in the desired cell.



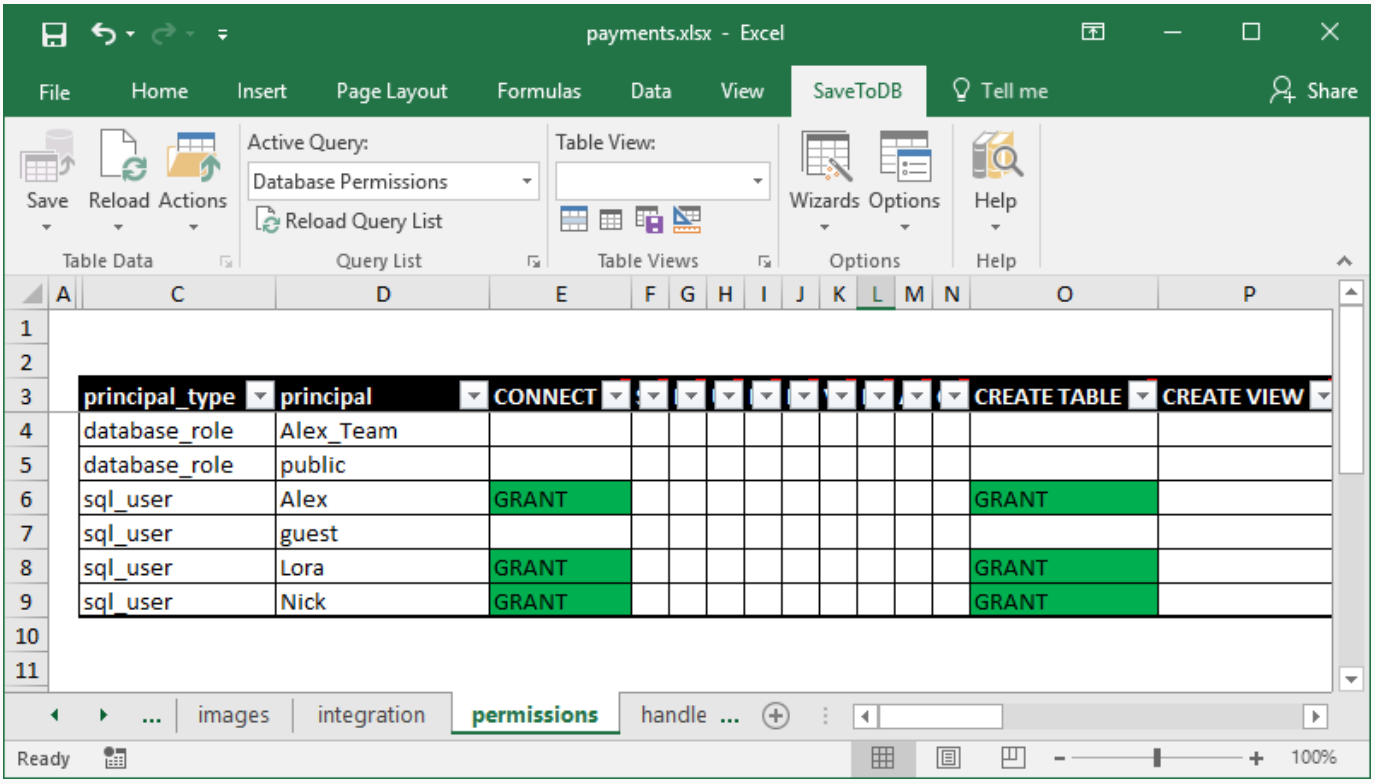
Principal Permissions

This table allows changing principal permissions. Set the **HasAny** value to 1 to see the actual permissions only.



Database Permissions

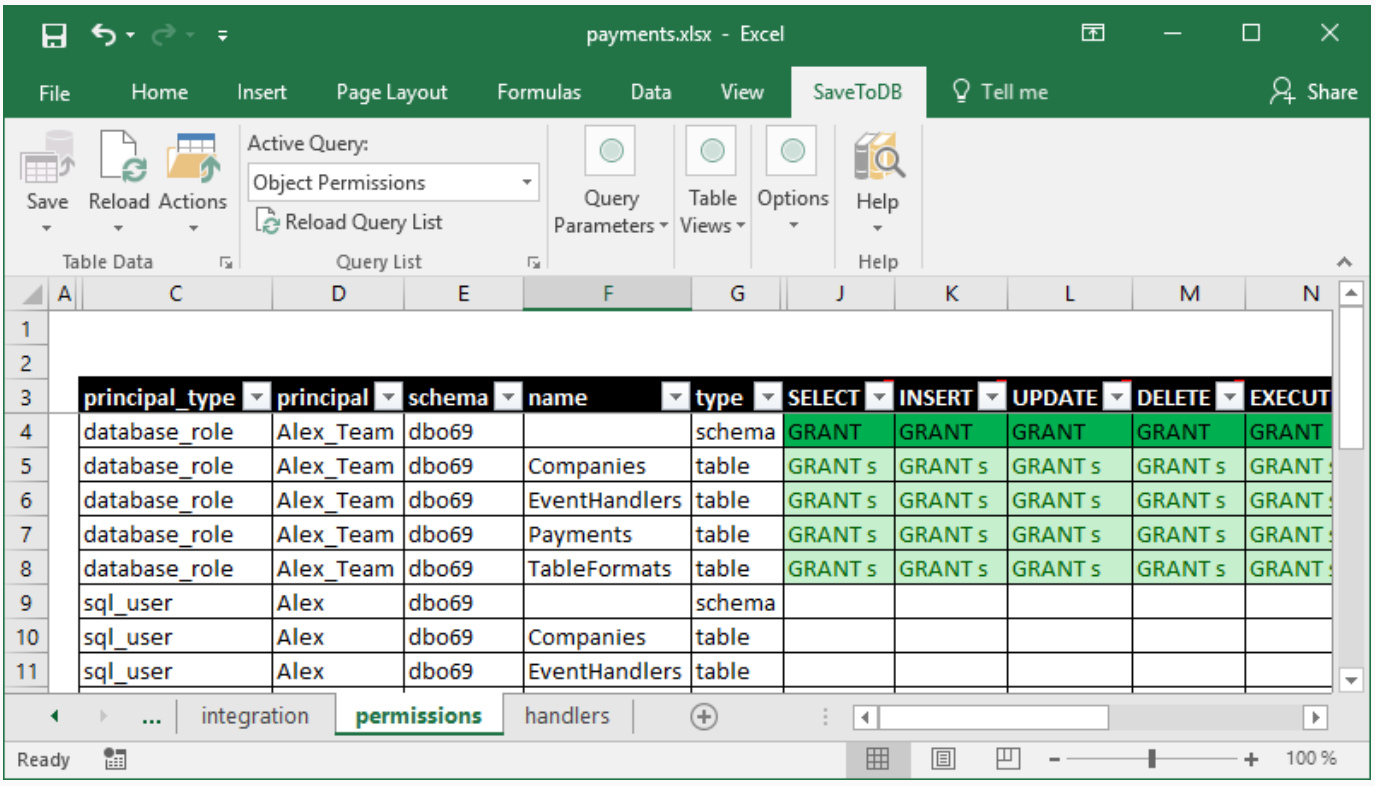
Only database administrators may change permissions. Business users must have **CREATE TABLE** permissions.



principal_type	principal	CONNECT	CREATE TABLE	CREATE VIEW
database_role	Alex_Team			
database_role	public			
sql_user	Alex	GRANT	GRANT	
sql_user	guest			
sql_user	Lora	GRANT	GRANT	
sql_user	Nick	GRANT	GRANT	

Object Permissions

This is a working table for business users. We learn it below. Set **HasAny** to 1 to see the actual permissions only.



principal_type	principal	schema	name	type	SELECT	INSERT	UPDATE	DELETE	EXECUT
database_role	Alex_Team	dbo69		schema	GRANT	GRANT	GRANT	GRANT	GRANT
database_role	Alex_Team	dbo69	Companies	table	GRANT s	GRANT s	GRANT s	GRANT s	GRANT
database_role	Alex_Team	dbo69	EventHandlers	table	GRANT s	GRANT s	GRANT s	GRANT s	GRANT
database_role	Alex_Team	dbo69	Payments	table	GRANT s	GRANT s	GRANT s	GRANT s	GRANT
database_role	Alex_Team	dbo69	TableFormats	table	GRANT s	GRANT s	GRANT s	GRANT s	GRANT
sql_user	Alex	dbo69		schema					
sql_user	Alex	dbo69	Companies	table					
sql_user	Alex	dbo69	EventHandlers	table					

The table contains actual permissions for our application:

principal_type	principal	schema	name	type	SELECT	INSERT	UPDATE	DELETE	EXECUTE	VIEW DEFINITION	CONTROL
database_role	Alex_Team	dbo69		schema	GRANT	GRANT	GRANT	GRANT	GRANT	GRANT	
database_role	Alex_Team	dbo69	Companies	table	GRANT s	GRANT s	GRANT s	GRANT s	GRANT s	GRANT s	
database_role	Alex_Team	dbo69	EventHandlers	table	GRANT s	GRANT s	GRANT s	GRANT s	GRANT s	GRANT s	
database_role	Alex_Team	dbo69	Payments	table	GRANT s	GRANT s	GRANT s	GRANT s	GRANT s	GRANT s	
database_role	Alex_Team	dbo69	TableFormats	table	GRANT s	GRANT s	GRANT s	GRANT s	GRANT s	GRANT s	
sql_user	Alex	dbo69		schema							GRANT+
sql_user	Alex	dbo69	Companies	table							GRANT+ s
sql_user	Alex	dbo69	EventHandlers	table							GRANT+ s
sql_user	Alex	dbo69	Payments	table							GRANT+ s
sql_user	Alex	dbo69	TableFormats	table							GRANT+ s
sql_user	Lora	dbo69		schema	GRANT r	GRANT r	GRANT r	GRANT r	GRANT r	GRANT r	
sql_user	Lora	dbo69	Companies	table	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	
sql_user	Lora	dbo69	EventHandlers	table	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	
sql_user	Lora	dbo69	Payments	table	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	
sql_user	Lora	dbo69	TableFormats	table	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	
sql_user	Nick	dbo69		schema	GRANT r	GRANT r	GRANT r	GRANT r	GRANT r	GRANT r	
sql_user	Nick	dbo69	Companies	table	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	
sql_user	Nick	dbo69	EventHandlers	table	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	
sql_user	Nick	dbo69	Payments	table	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	
sql_user	Nick	dbo69	TableFormats	table	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	

Suffix legend: **s** – by parent schema, **r** – by parent role, **sr** – by parent schema and role, **+** - WITH GRANT OPTION

We see the following data under Alex's credentials:

- The database contains the Alex_Team role and users: Alex, Lora, and Nick.
- The database contains the dbo69 schema and tables: Companies, EventHandlers, Payments, TableFormats.
- Alex has the CONTROL permission WITH GRANT OPTION on the dbo69 schema and schema tables.
- Alex_Team and its members have read and write permissions on the dbo69 schema and schema tables.

You may change permissions using first letters: **G** – GRANT, **D** – DENY, **R** – REVOKE

In the following example, Alex denies the INSERT permission for the EventHandlers table directly, and the UPDATE and DELETE permission indirectly, using the role level.

principal_type	principal	schema	name	type	SELECT	INSERT	UPDATE	DELETE	EXECUTE	VIEW DEFINITION	CONTROL
database_role	Alex_Team	dbo69		schema	GRANT	GRANT	GRANT	GRANT	GRANT	GRANT	
database_role	Alex_Team	dbo69	Companies	table	GRANT s	GRANT s	GRANT s	GRANT s	GRANT s	GRANT s	
database_role	Alex_Team	dbo69	EventHandlers	table	GRANT s	GRANT s	DENY	DENY	GRANT s	GRANT s	
database_role	Alex_Team	dbo69	Payments	table	GRANT s	GRANT s	GRANT s	GRANT s	GRANT s	GRANT s	
database_role	Alex_Team	dbo69	TableFormats	table	GRANT s	GRANT s	GRANT s	GRANT s	GRANT s	GRANT s	
sql_user	Alex	dbo69		schema							GRANT+
sql_user	Alex	dbo69	Companies	table							GRANT+ s
sql_user	Alex	dbo69	EventHandlers	table							GRANT+ s
sql_user	Alex	dbo69	Payments	table							GRANT+ s
sql_user	Alex	dbo69	TableFormats	table							GRANT+ s
sql_user	Lora	dbo69		schema	GRANT r	GRANT r	GRANT r	GRANT r	GRANT r	GRANT r	
sql_user	Lora	dbo69	Companies	table	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	
sql_user	Lora	dbo69	EventHandlers	table	GRANT sr	DENY	DENY r	DENY r	GRANT sr	GRANT sr	
sql_user	Lora	dbo69	Payments	table	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	
sql_user	Lora	dbo69	TableFormats	table	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	
sql_user	Nick	dbo69		schema	GRANT r	GRANT r	GRANT r	GRANT r	GRANT r	GRANT r	
sql_user	Nick	dbo69	Companies	table	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	
sql_user	Nick	dbo69	EventHandlers	table	GRANT sr	DENY	DENY r	DENY r	GRANT sr	GRANT sr	
sql_user	Nick	dbo69	Payments	table	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	
sql_user	Nick	dbo69	TableFormats	table	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	GRANT sr	

In any case, if you have this task, you will find this app user-friendly and learn required actions quickly.

Do not afraid and go ahead.

Conclusion

We have created an Excel application with cool professional features and learned steps that you may repeat:

1. Publish tables to a database
2. Configure query parameters
3. Configure formats and table views
4. Configure validation lists
5. Add cursors and form fields
6. Configure detail tables, windows, and task panes
7. Configure context and action menus
8. Add image URLs
9. Configure LastModified and UserName fields
10. Integrate with other applications
11. Manage permissions

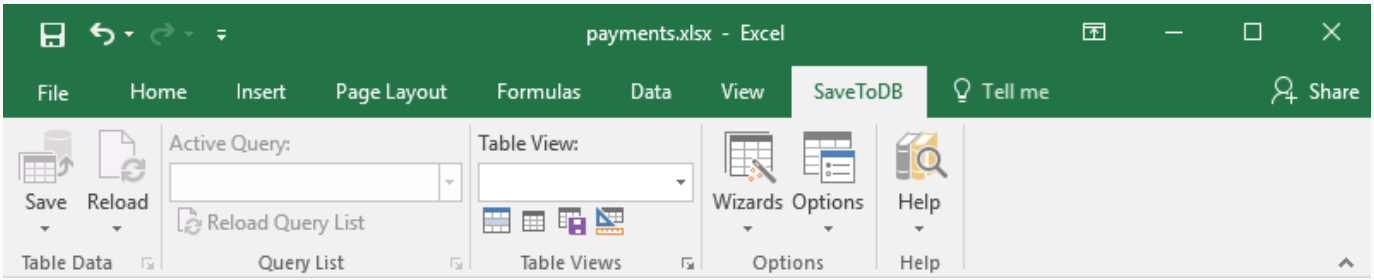
You may do these steps with no SQL and VBA. We have configured the application using wizards and handlers:

ID	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	EVENT_NAME	HANDLER_SCHEMA	HANDLER_NAME	HANDLER_TYPE	HANDLER_CODE	TARGET_WORKSHEET
1	dbo69	Payments	Account	ValidationList	dbo69	Accounts	VALUES	My Bank	
2	dbo69	Payments	Item	ValidationList	dbo69	Items	VALUES	Revenue,Expenses,Payroll,Taxes	
3	dbo69	Payments	Company	ValidationList	dbo69	Companies	TABLE	Company	
4	dbo69	Payments		SelectionChange	dbo69	Payments	TABLE		_TaskPane _Transpose
5	dbo69	Payments	Company	ContextMenu	dbo69	Payments	TABLE	+Date,+Sum,Account,@Company,Item,Comment	
6	dbo69	Payments	Item	ContextMenu	dbo69	Payments	TABLE	+Date,+Sum,Account,Company,@Item,Comment	
7	dbo69	EventHandlers		Actions	dbo69	Instruction	HTTP	https://www.savetodb.com/savetodb/configuring-event-ha	
8	dbo69	Companies	LastModified	FormulaValue				=NOW()	
9	dbo69	Companies	UserName	FormulaValue				=UserName()	
10	dbo69	Companies	LastModified	DoNotChange					
11	dbo69	Companies	UserName	DoNotChange					

Of course, database and VBA developers may do much more. You may recommend them my e-books:

- Excel Applications. 10 Steps for Database Developers
- Excel Applications. 10 Steps for VBA Developers

Hope you will like the SaveToDB add-in as I am and will use the SaveToDB tab every day.



You may download the SaveToDB add-in at www.savetodb.com and use the SaveToDB Express edition for free.

I would appreciate your feedback. Feel free to contact me at 11-steps@savetodb.com.

Cool applications for you and your team!

Sergey Vaselenko

About the Author

My name is Sergey Vaselenko.

I am from Russia, Moscow.

My passion is creating software.

I am a founder and CEO of Gartle Technology Corporation and a leading developer of the SaveToDB add-in.

You are welcome to contact me at

www.facebook.com/sergey.vaselenko

www.linkedin.com/in/vaselenko/