Profitbase Low Code

Training Guide

> You must unlearn what you have learned.
>
> - Master Yoda

Date: 24.oct 2019

Table of content

1 Introduction

This document is a guide to build a small Profitbase Planning solution. The idea is that the user should get a basic understanding of how to build a planning solution and the most common features of Profitbase LOW CODE. The functionality presented in this guide is just a small piece of functionality that Profitbase LOW CODE tool has to offer.

The goal for this guide is to build a small planning solution for product sales. The solution will consist of three workbooks, one workpage for planning input and one workpage reporting data.

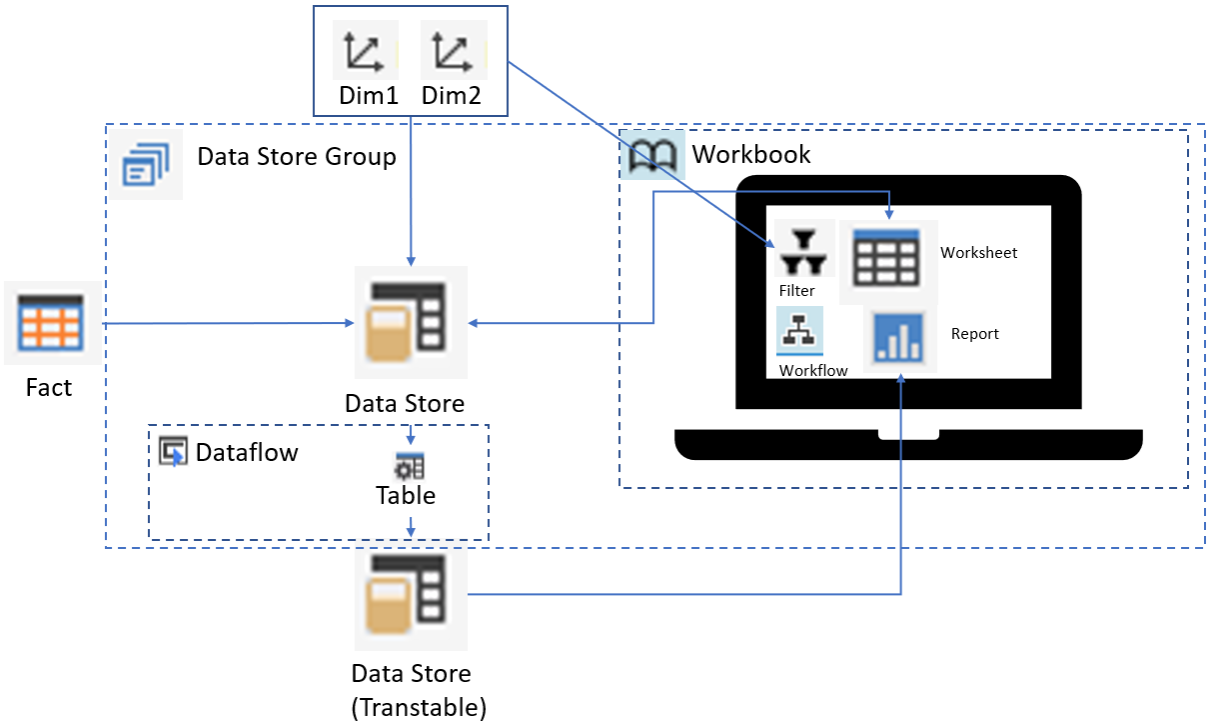


Figure 1 Solution Sketch

1.1 Prerequisites

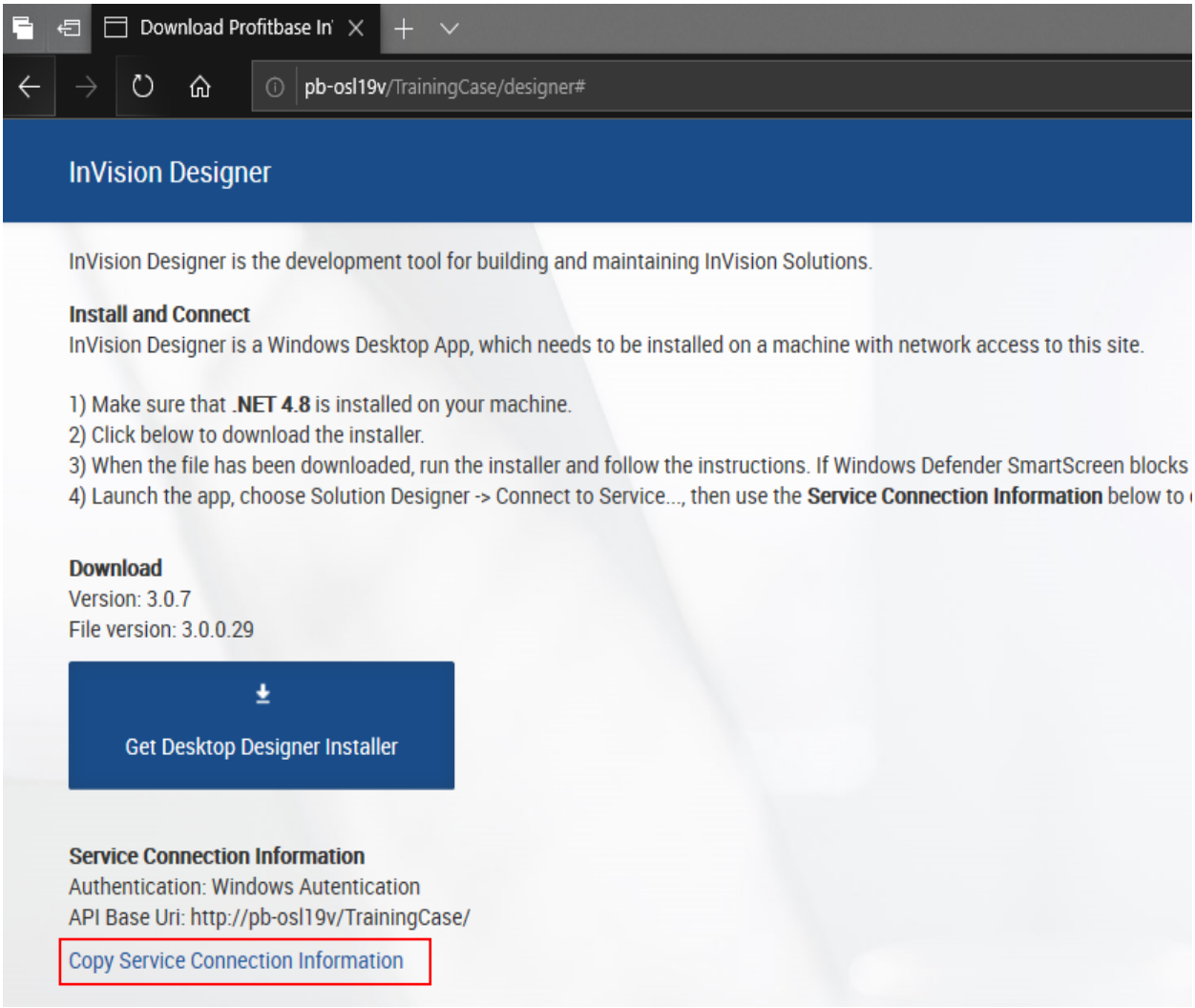
- InVision 3.0 with designer
- SQL 2017
- pbDW_AleksTest
- pbSol_AleksTest

Basic knowledge of SQL queries is recommended.

2 How to set up the designer and create a new solution

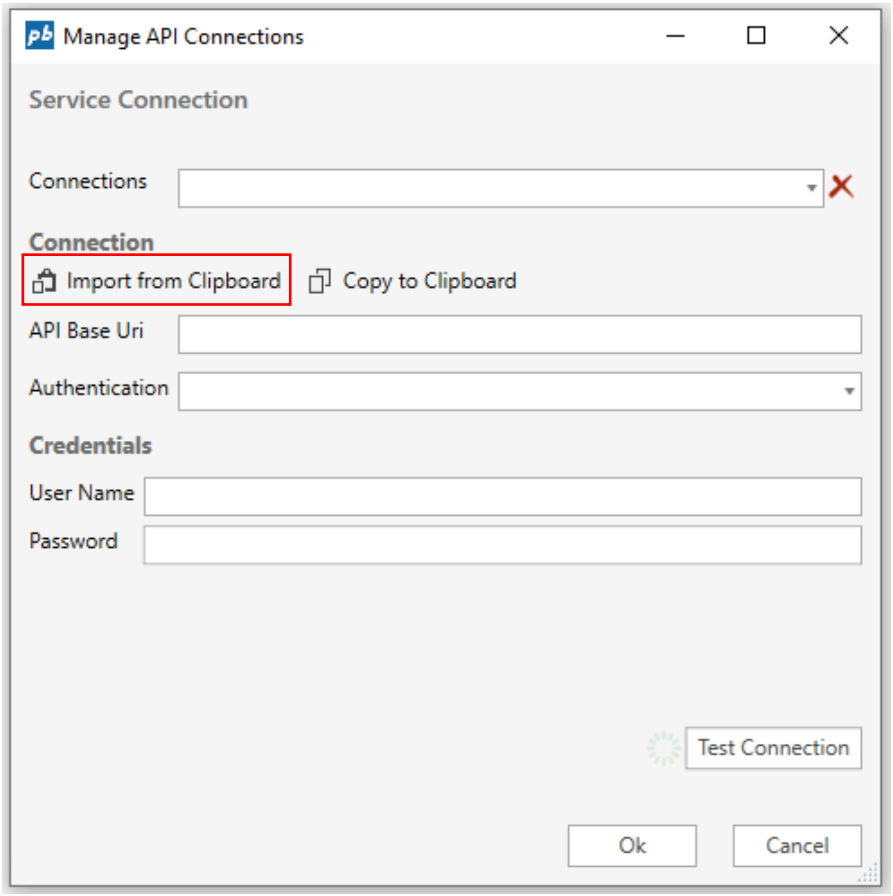
2.1 Get your Service Connection Information

1. Open your web browser and go to your Profitbase LOW CODE solution webpage
   ➔ *ServerName/SolutionName/Designer*
2. Add "/designer" at the end of URL
3. Click on "Copy Service Connection Information"



2.2 Set up Service Connection in the designer

1. Start by opening the desktop designer on your computer
2. Select “Solution Designer” in the “Home” pane on the left-hand side and Click “Connect to Service…”
3. Click on “Import from Clipboard”
4. Use the “Test Connection” button to make sure the connection was successful
5. Click “Ok”

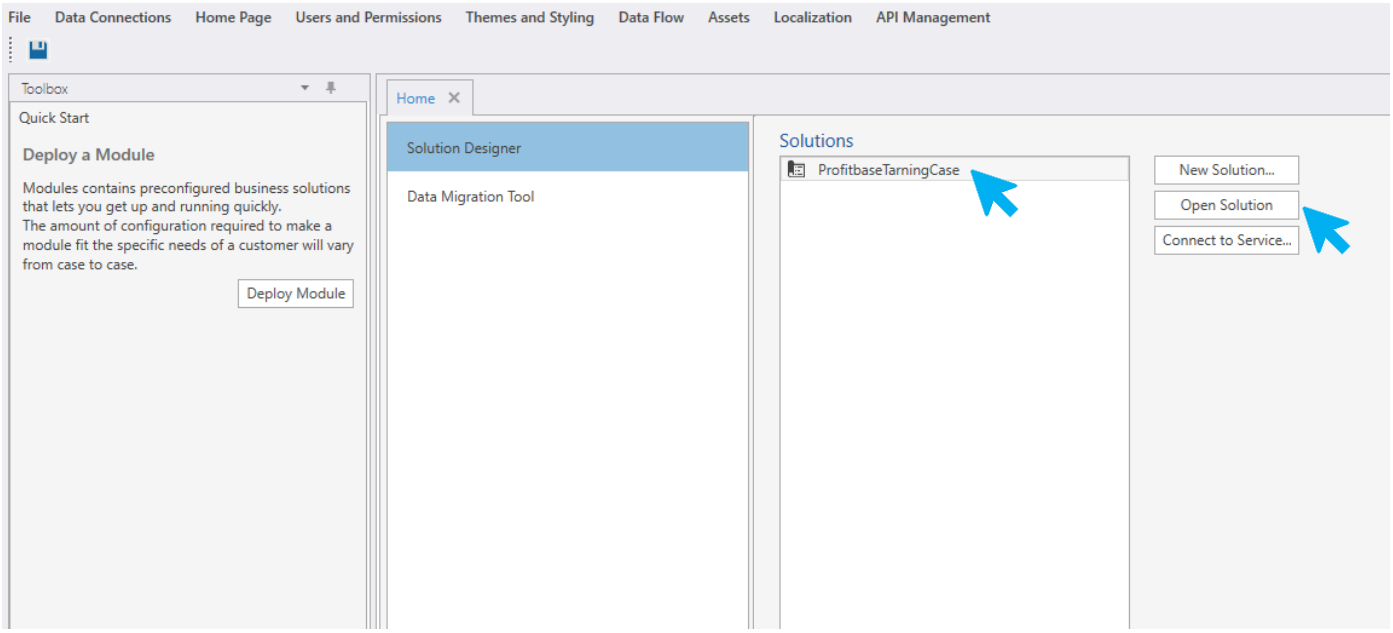


2.3 Creating a new solution

1. Click on the “New Solution…” button
2. Give your solution a suitable name in the solution dialog box and click “Ok”

Congratulations, you have successfully created a new solution!

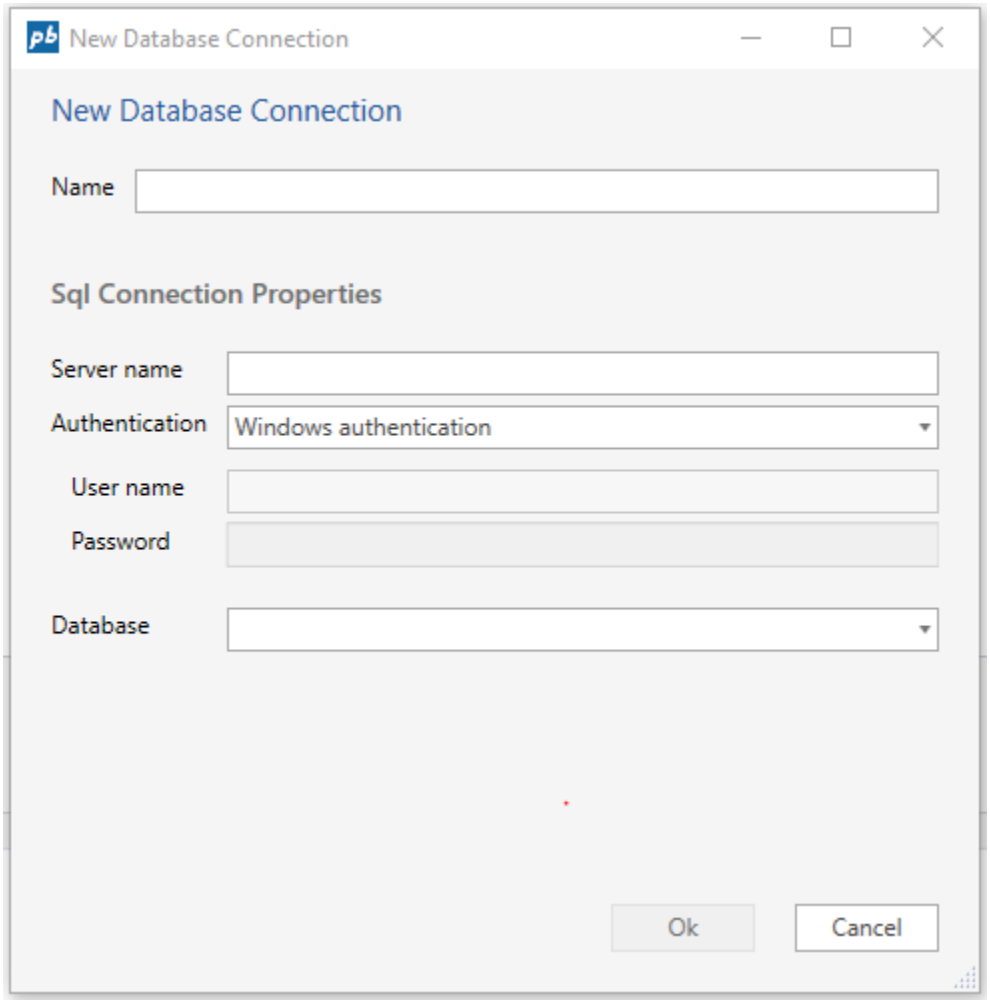
3. Select your solution and click “Open Solution”



3 Connecting to a Profitbase Datawarehouse

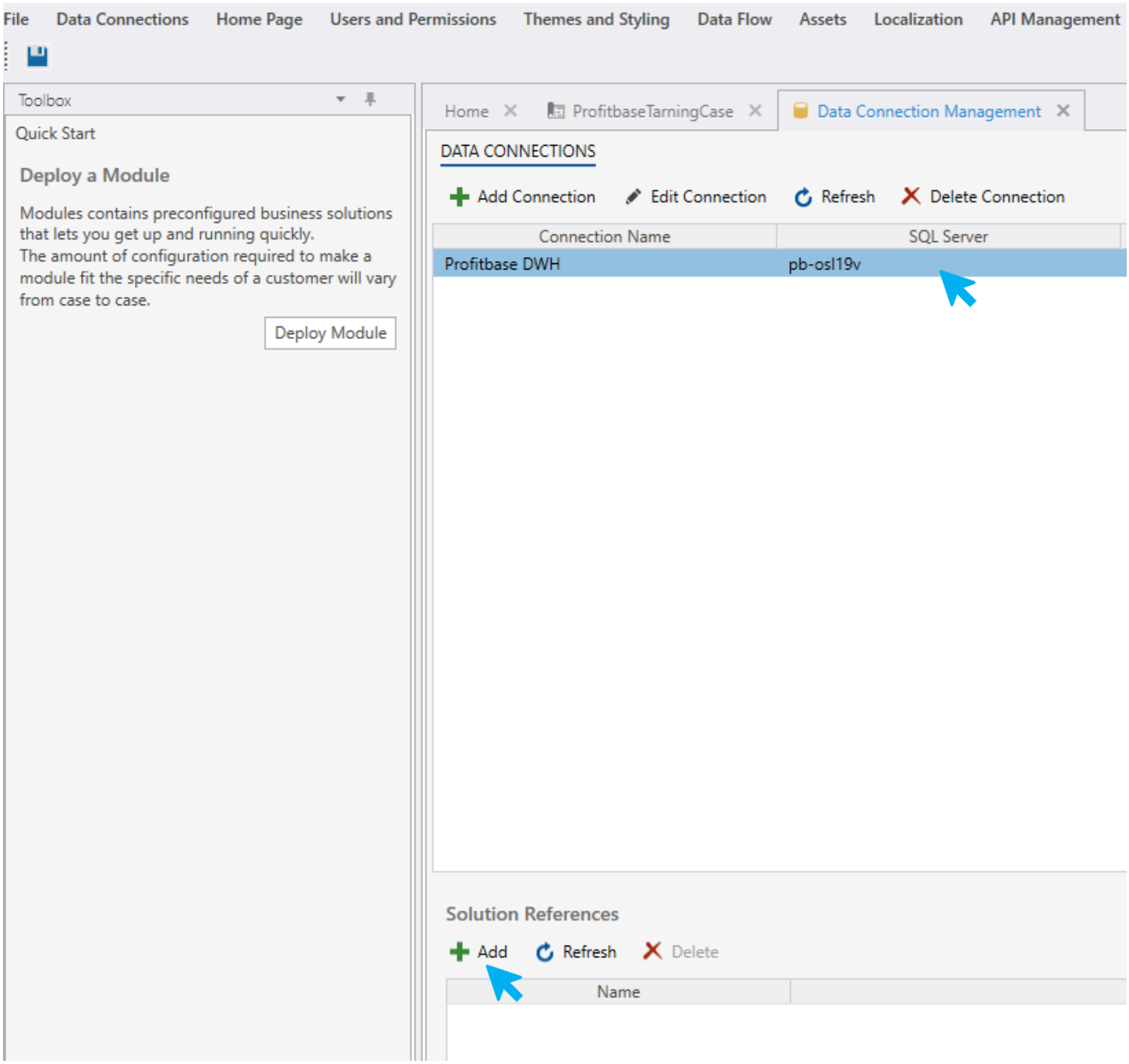
Your Profitbase LOW CODE solution could be set up to get data and dimensions from a Profitbase Studio Data warehouse. However, this is not a requirement.

1. Click on “Data Connections” in the top ribbon of the designer
2. Click on “Manage Connections…”
3. Click on “Add Connection”
4. Fill out the dialog box



Name: Give your Solution a name
Server name: The server hosting your Profitbase Data warehouse
Database: Your Profitbase solution database, it starts with: pbSol_

5. Make sure your data connection is highlighted and click “Add” under Solution References below



6. Select your Solution from the New Solution References dialog box and click “OK”
7. Click the save icon in the top left corner of the designer
8. Close the Data Connection Management tab

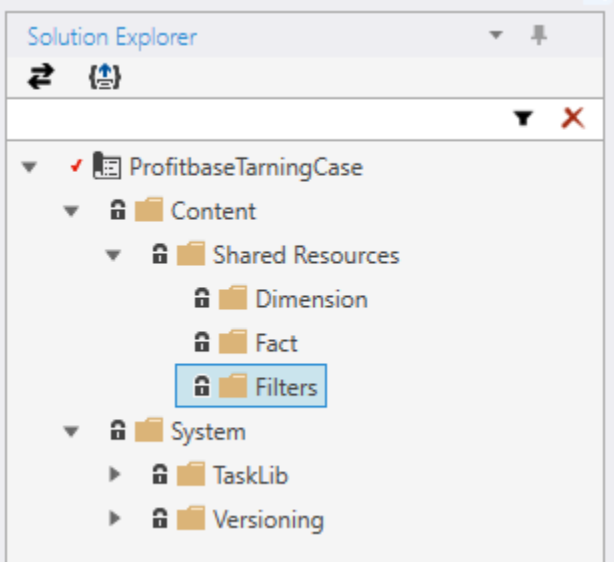
4 Getting dimensions and facts

You are now at the point where you start building your solution. You will mostly use the Solution Explorer on the right side of your designer window. Keeping a good structure in your solution is important. Profitbase strongly suggest using folders to keep your solution organized. You should also choose sensible names for the items you create as they are likely to be used as references elsewhere in the solution.

You will build your solution under the Content folder. We will start by making a folder structure for our resources.

1. Right click on the Content folder and select “New Folder”
2. Name your new folder “Shared Resources” and click “Ok”
3. Expand your Content folder
4. Right click on your Shared Resource folder and select “New Folder”
5. Name your new folder “Dimension” and click “Ok”
6. Create 2 additional folders under Shared Resources Named “Fact” and “Filters”

Your folder structure should look like this when you are done:



4.1 Creating a Dimension

You have the option to create two types of dimensions in Profitbase LOW CODE. Free dimensions and linked dimensions. Linked dimensions are connected to a dimension in your Profitbase Data warehouse and the maintenance of these dimensions will be done in Profitbase Studio.

4.1.1 Creating a free dimension

1. Right click on your dimensions folder and select “Add New Item…”
2. Select “Dimension” and name it, Department, then click “Ok”

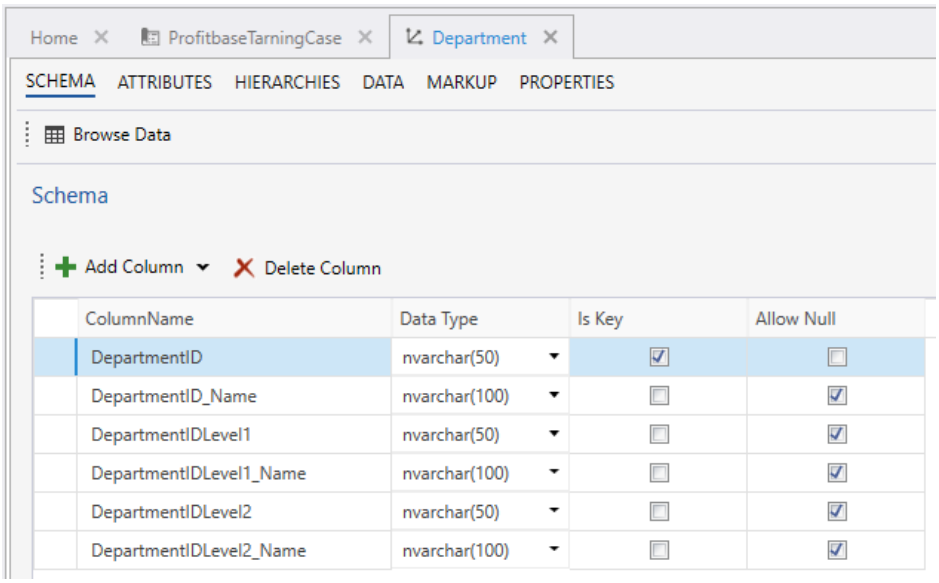
4.1.2 Creating a dimension schema

When the Dimensions are created the respective dimension editor will be opened. Here you can start to create the schema for your dimension. Your dimension table will need the following columns (Name- Datatype):

- DepartmentID - Nvarchar(50)
- DepartmentID_Name - Nvarchar(100)
- DepartmentIDLevel1 - Nvarchar(50)
- DepartmentIDLevel1_Name - Nvarchar(100)
- DepartmentIDLevel2 - Nvarchar(50)
- DepartmentIDLevel2 - Nvarchar(100)

1. Use the “Add Column” button to create a schema with the columns and datatypes listed above.
2. You will see that Nvarchar(100) does not show up as an alternative in the dropdown menu. Use the Nvarchar(50) data type, then double click and edit the character indicator from 50 to 100.
3. Make the DepartmentID column a key column by checking the “Is Key” check box
4. Click on the save icon or Ctrl + S to save your dimension

Make sure your dimension schema looks like this:



4.1.3 Creating a dimension attribute

1. Go to the “Attributes” tab
2. Click on “Automap Columns to Attributes”, then click Save

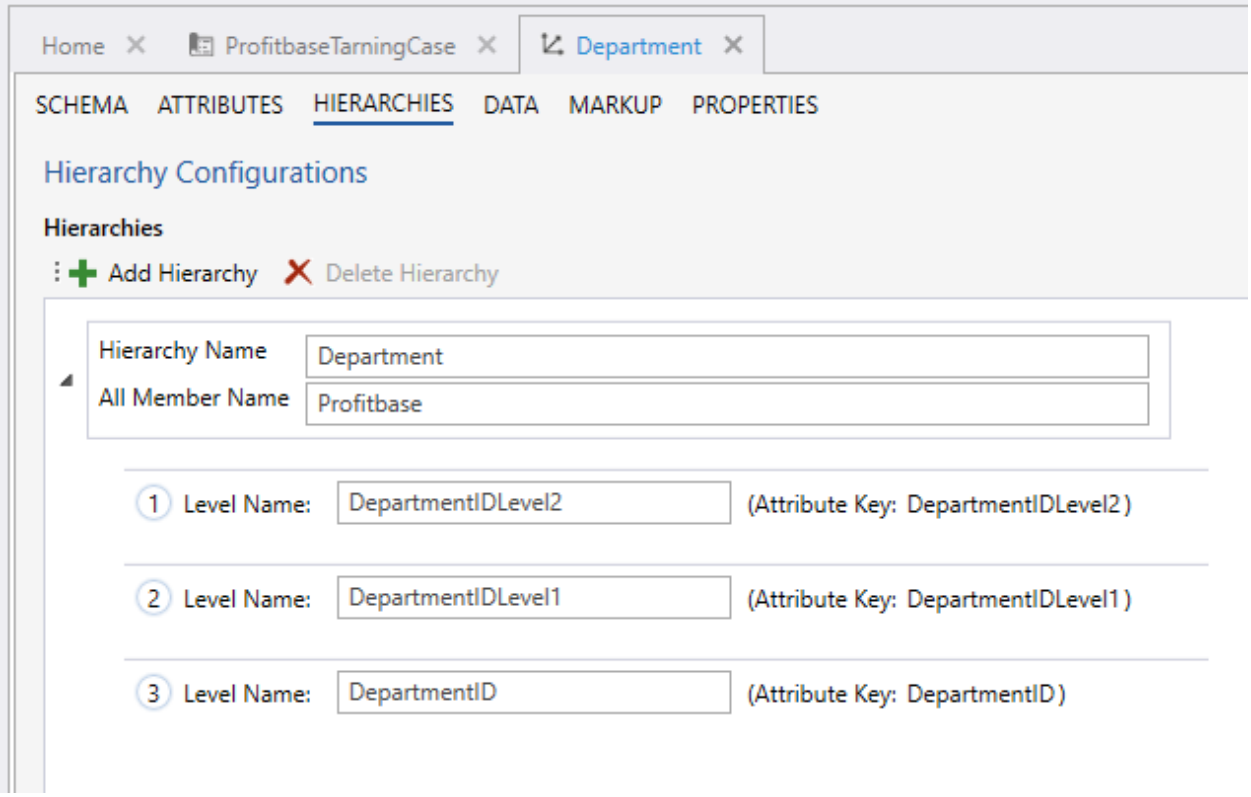
4.1.4 Creating a dimension hierarchy

1. Go to the “Hierarchies” tab
2. Click on the “Add Hierarchy” button
3. Give your Hierarchy a Name and an All Member Name

   Hierarchy Name: Department

   All Member Name: Profitbase
4. Drag and drop your three levels from the Attributes box on the right into the Hierarchy box, to create levels in your dimension. You want your lowest level of the hierarchy at bottom. Remember to save your work

When you’re done, the hierarchy should look like this:



4.1.5 Fill your dimension with data

Go to the “Data” tab to add data to your hierarchy. By selecting Department form your hierarchy drop down menu you will see that your hierarchy is missing data. You can click on [No Data- select this to allow adding rows] and then use the “Add Row” button to start filling your hierarchy manually. This could be time consuming work. However, if you have the data in another SQL table then you can write a SQL script to fill the dimension with data.

Edit the SQL Merge script below in SSMS (Microsoft SQL Server Management Studio) to transfer data:

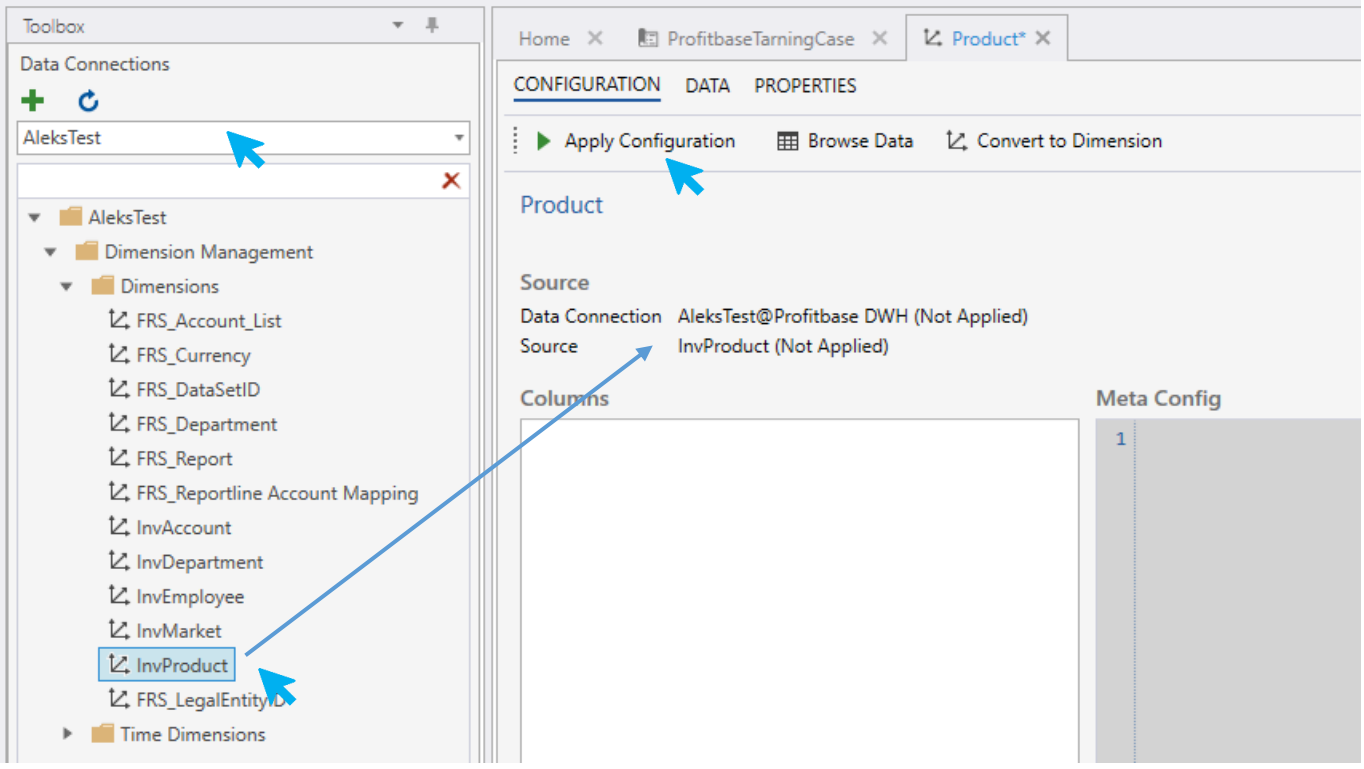
```
/*
Setup usage:
	1. Edit the SQL below.

Note:
Deleting departments may cause Foreign Key constraint error if dimension member is
referenced in fact table together with dimension.
*/
Merge into @Object[Department, Dim].DbObjectName t
USING (
/*
	Define the source for external content below.
	Remember to provide all columns required.
*/
-- Edit this SQL ...

SELECT -- Dummy SQL to be changed
[DepartmentID],[DepartmentID_Name],[DepartmentIDLevel1],[DepartmentIDLevel1_Name],[Depart
mentIDLevel2],[DepartmentIDLevel2_Name]
from [pbDW_AleksTest].[dbo].[vw_pbSol_FinanceSales_AX2009_DimW_DepartmentID]
where isnull(DepartmentID, 0) <> 0

-- Dummy SQL where source is same as target
-- From [server],[databasename].[schema].[tablename]
-- End of SQL edit

) s
on t.DepartmentID = s.DepartmentID -- Note that DepartmentID should be unique across
LegalEntities and Departments

-- Standard merge clause
-- Note that there may be FK constraints in the model which will prevent removing member
items that are referenced in fact tables
-- If so a SQL message will be issued and the merge will fail.

when matched then
Update
Set [DepartmentID_Name]=s.[DepartmentID_Name],
	[DepartmentIDLevel1]=s.[DepartmentIDLevel1],
	[DepartmentIDLevel1_Name]=s.[DepartmentIDLevel1_Name],
	[DepartmentIDLevel2] =s.[DepartmentIDLevel2],
	[DepartmentIDLevel2_Name] = s.[DepartmentIDLevel2_Name]

when not matched by target then
insert
([DepartmentID],[DepartmentID_Name],[DepartmentIDLevel1],[DepartmentIDLevel1_Name],
[DepartmentIDLevel2],[DepartmentIDLevel2_Name])

values (s.[DepartmentID],s.[DepartmentID_Name],s.[DepartmentIDLevel1],
s.[DepartmentIDLevel1_Name],s.[DepartmentIDLevel2],s.[DepartmentIDLevel2_Name])
;
```

After you have executed the merge script, go back to the Profitbase LOW CODE Designer.

1. Go to the “Data” tab under you Department dimension.
2. Select Department in the Hierarchy drop down
3. Click “Publish”
    a. If the button is disabled, select the hierarchy in the dropdown list. Trigger a change in the data table (e.i. type a space inn a cell and delete it again). Then save that “change” on the small save button above the worksheet. The publish button should now be enabled.

4.1.6 Creating a linked Dimension

1. Right click on your Dimension folder and select “Add new item…”
2. Choose “Linked Dimension” and name it “Product”, then click “Ok”
3. Under Data Connections in the Toolbox on the left-hand side, select your solution
4. Expand your data connection and drag and drop your dimension into the Source field
5. Click “Apply Configuration”



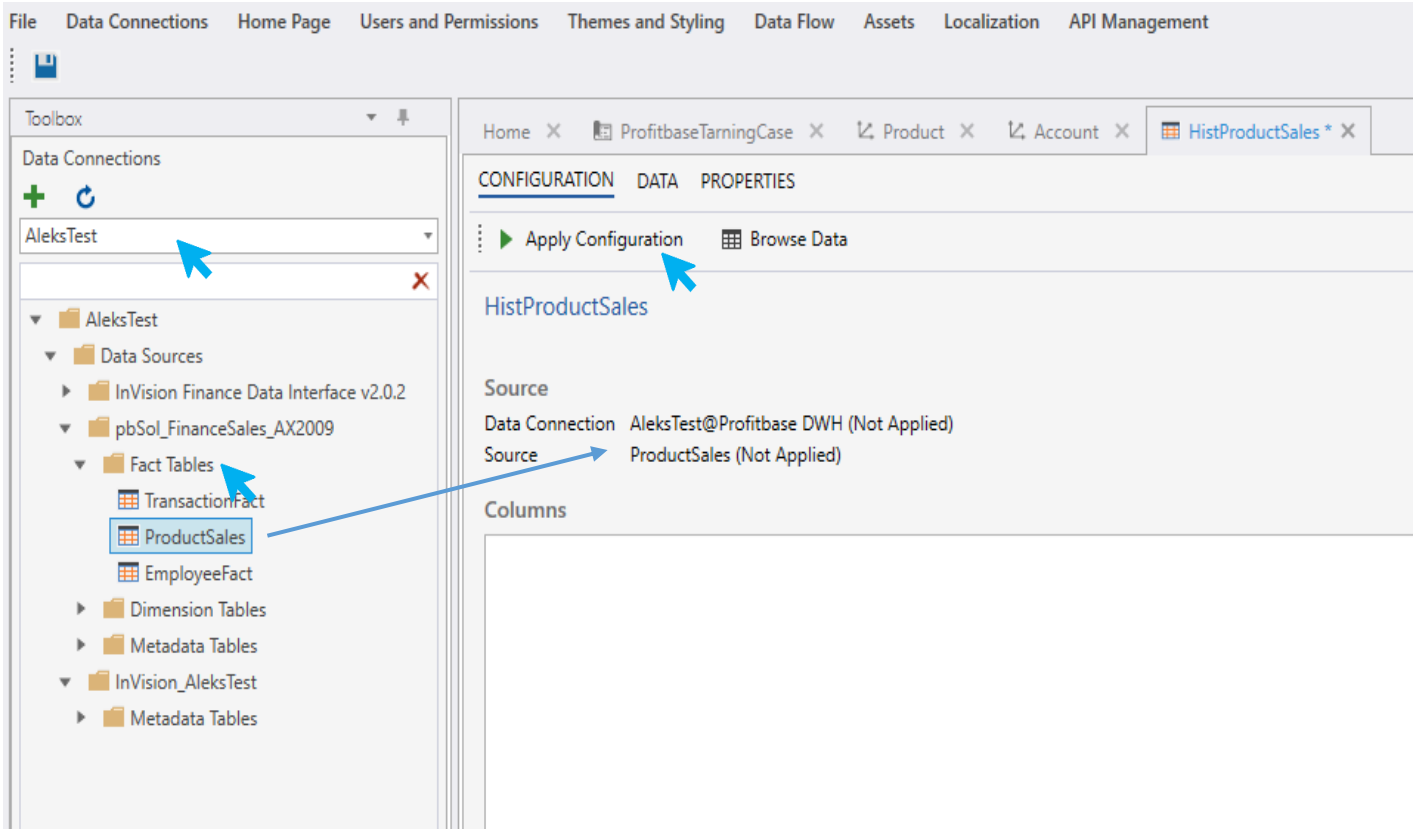
6. When the Data Flow has successfully completed, close the dialog box

Repeat the steps 1 through 6 and create a linked Dimension called Account.

Hint: If you have a linked Dimension that you want to maintain manually in your solution, you can convert it from a linked dimension to a regular dimension by clicking on the “Convert to Dimension” button. Be aware that this conversion is not reversable and it creates a new ID for the recreated dimension. This is not ideal, as you need to update the ID wherever the old dimension is referenced.

4.2 Creating a fact table

1. Right click on your Fact folder and select “Add new item…”
2. Choose “Linked Fact” and name it “HistProductSales”, then click “Ok”
3. Use Data Connections under Toolbox and select your solution
4. Expand your data connection and drag and drop your fact table to the Source field
5. Click “Apply Configuration”

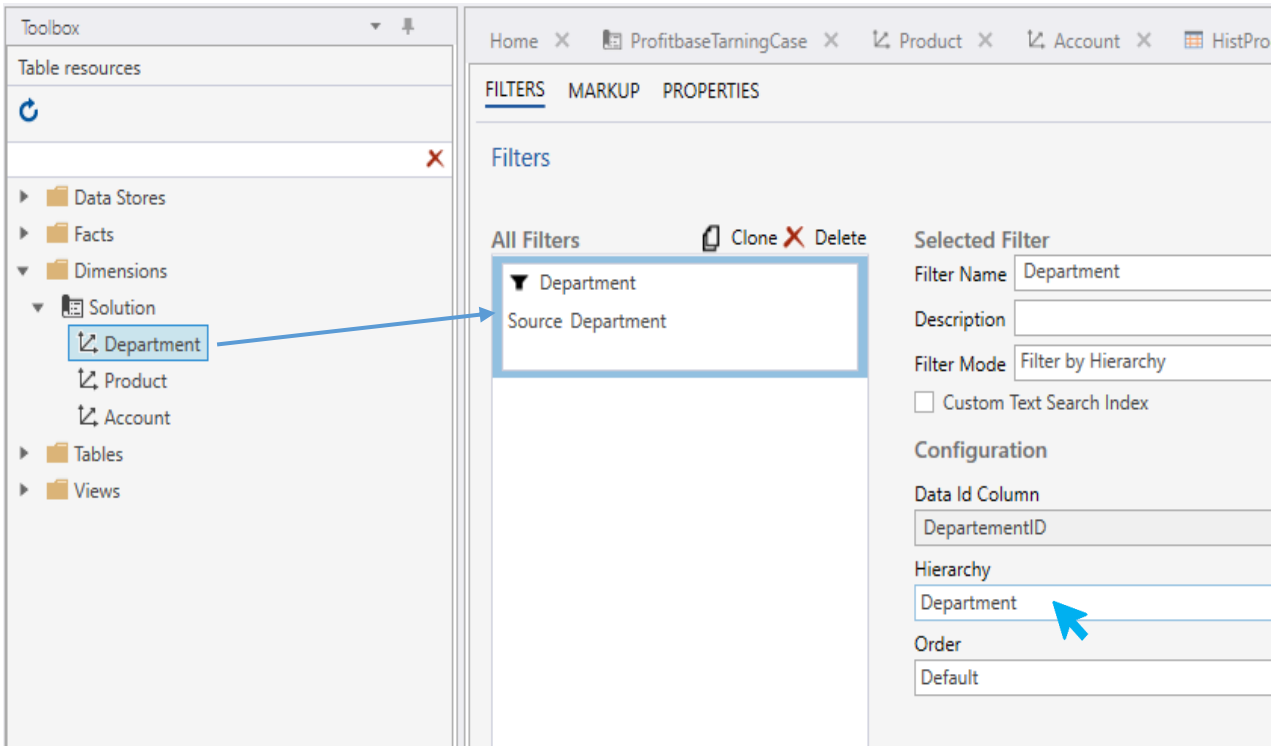


6. When the Data Flow has successfully completed, close the dialog box

5 Filters

You need a department filter in the solution. This filter can be created by the following steps:

1. Right click on your Filters folder and select “Add new item…”
2. Select “Filters” and name it “Filters” then click “Ok”
3. You will now see Table Resources in the Toolbox pane on the left-hand side. Expand the dimension folder until you find the department dimension you created earlier
4. Drag and drop the department dimension into the “All Filters” field
5. Select Department” in the Hierarchy dropdown field
6. Click Save or Ctrl + S



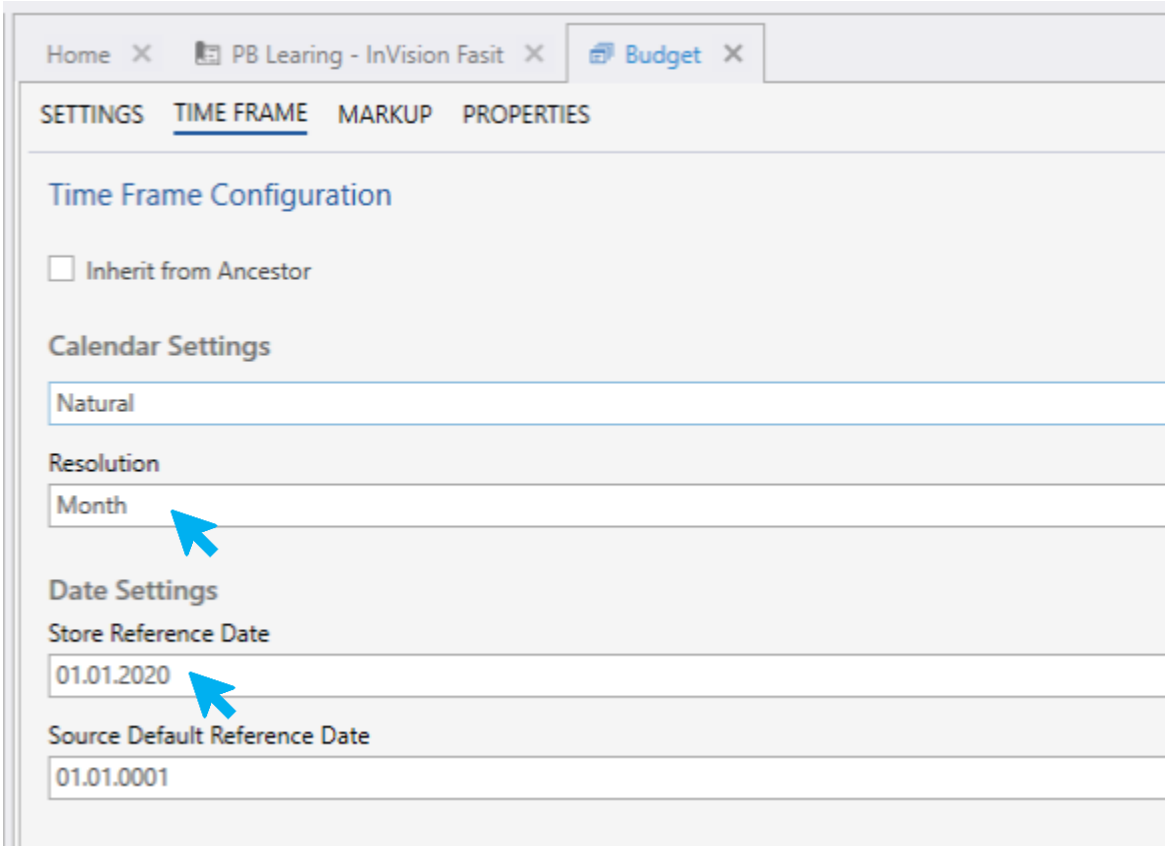
6 Data Store Group

In this product sales planning solution, we are going to plan our sales for 1 year at a time with 12 monthly periods. Our datastore will need a timeframe reference date. You could either set this date in all your data stores separately, or you could define it in the data store group and have them inherit the timeframe references date form there.

In this case we are going to use a Data Store Group.

1. Right click on your Content folder and select “Add new item…”
2. Choose “Data Store Group” and name it “Budget”, then click “Ok”
3. Go to the “Time Frame” tab
4. Makes sure Resolution is set to Month and that the Store Reference Date is set to 01.01.2020
    a. Note: This time reference can be inherited by all the components placed below this storegroup I the solution explorer. This is done by checking off the “Inherit from Ancestor” check box on the given component.
5. Click Save or Ctrl + S

Your configuration should look like this:



7 Data Store – Product Sales

Data Stores are the primary data repositories of your Solution. Data Stores should be used for storing the master data and processing results for your business modules, while Tables should contain configuration data for business rules and utility functions.

Data Stores are materialized as SQL Server tables, and can be read and written to by any SQL script or process having the required permissions. Data is stored "as is", so you do not need to use an InVision API to read or write data to a Data Store. This makes it easy to integrate with other tools, such as Excel or a custom Web Service, in order to export and import data to and from InVision.

1. Right clicking on the Data Store Group “Budget” and select “Add new item…”
2. Choose “Data Store” and name it “Product Sales”, then click “Ok”

7.1 Creating Data Store Schema

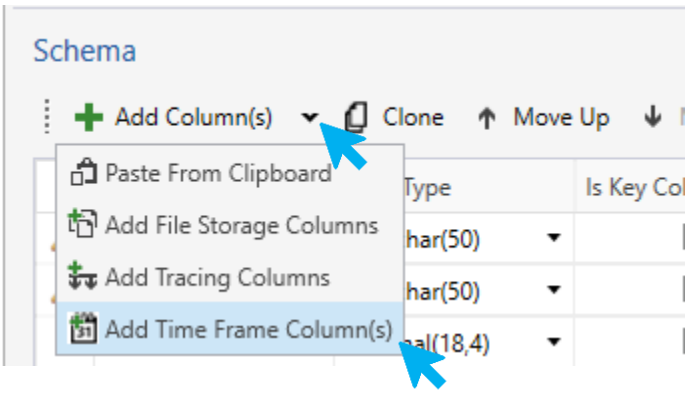
Our Product Sales data store will consist of these columns:

- DepartmentID – Nvarchar(50)
- ProductID – Nvarchar(50)
- HistSalesLY – Decimal(18,4)
- MarginPCT – Decimal(18,4)
- P01Sales– P12 Sales – Decimal(18,4)
- Comments – Nvarchar(350)
- P01Cost – P12Cost – Decimal(18,4)

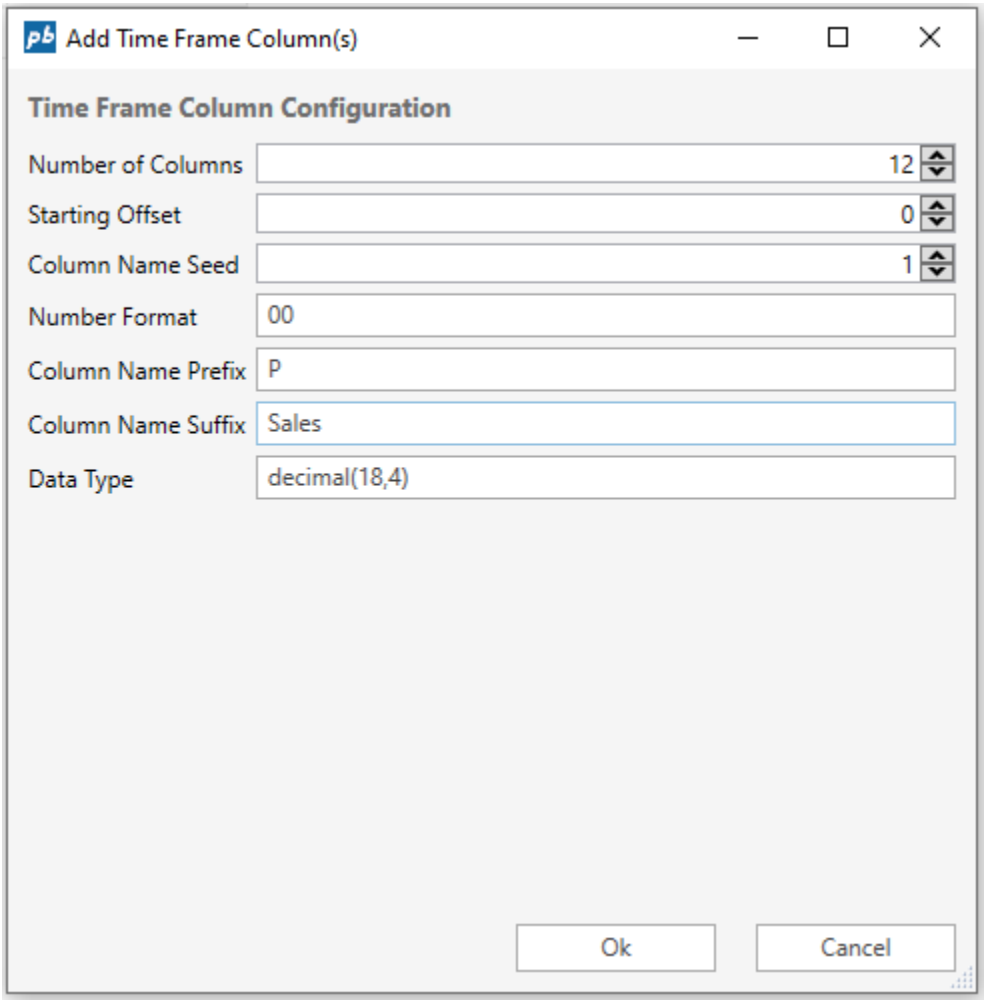
DepartmentID and ProductID are key columns. This is indicated with key symbol left of the column name. The columns P01Sales to P12Sales and P01Cost to P12Cost are timeframe columns. This is indicated with small calendar symbol.

1. Use the “add column” button to add the columns DepartmentID, ProductID, HistSalesLY and MarginPCT.
2. Name them and give them the correct datatype.
3. Remember to check of “Is Key Column” for DepartmentID and ProductID
4. Click “Save” or Ctrl + S

5. Click the small “Arrow” on the “Add Column(s)” button and select “Add Time Frame Column(s)”, as shown by image below

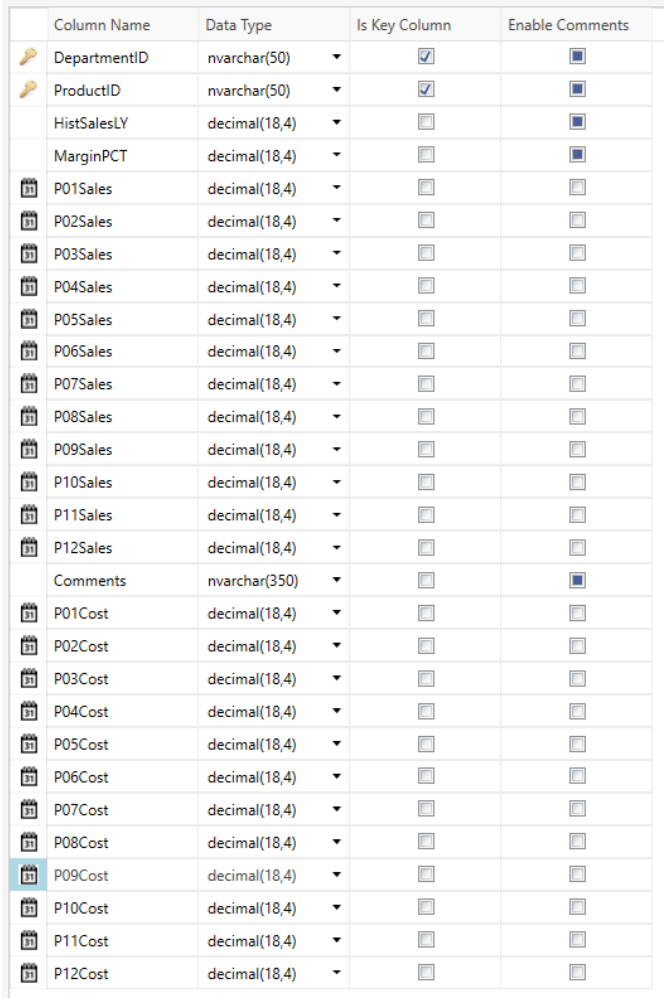


6. Makes sure the “Add Time Frame Column(s)” pop up is filed out like this:



7. Add the column Comments. You will see that there are no data type Nvarchar(350) in the dropdown menu. Use the Nvarchar(50) data type and edit the number of characters from 50 to 350 by double clicking on the respective cell
8. Repeat step 5 and 6 but this time changes “Column Name Sufix” to “Cost” instead of “Sales”
9. Click “Save” or Ctrl + S

Your Schema should look like this:



Column Name	Data Type	Is Key Column	Enable Comments
DepartmentID	nvarchar(50)	☑	■
ProductID	nvarchar(50)	☑	■
HistSalesLY	decimal(18,4)	☐	■
MarginPCT	decimal(18,4)	☐	■
P01Sales	decimal(18,4)	☐	☐
P02Sales	decimal(18,4)	☐	☐
P03Sales	decimal(18,4)	☐	☐
P04Sales	decimal(18,4)	☐	☐
P05Sales	decimal(18,4)	☐	☐
P06Sales	decimal(18,4)	☐	☐
P07Sales	decimal(18,4)	☐	☐
P08Sales	decimal(18,4)	☐	☐
P09Sales	decimal(18,4)	☐	☐
P10Sales	decimal(18,4)	☐	☐
P11Sales	decimal(18,4)	☐	☐
P12Sales	decimal(18,4)	☐	☐
Comments	nvarchar(350)	☐	■
P01Cost	decimal(18,4)	☐	☐
P02Cost	decimal(18,4)	☐	☐
P03Cost	decimal(18,4)	☐	☐
P04Cost	decimal(18,4)	☐	☐
P05Cost	decimal(18,4)	☐	☐
P06Cost	decimal(18,4)	☐	☐
P07Cost	decimal(18,4)	☐	☐
P08Cost	decimal(18,4)	☐	☐
P09Cost	decimal(18,4)	☐	☐
P10Cost	decimal(18,4)	☐	☐
P11Cost	decimal(18,4)	☐	☐
P12Cost	decimal(18,4)	☐	☐

10. Go to the “Time Frame” tab and check off the “Inherit from ancestor” checkbox.

    Your time frame columns now represent each month from the reference date in the storegroup and 12 months forward. The offset represents how many resolutions (months in this case) from the reference date the column represents. This configuration is used to create a date for the transactions originating from each column through a pipeline in a later stage.

7.2 Source Mapping

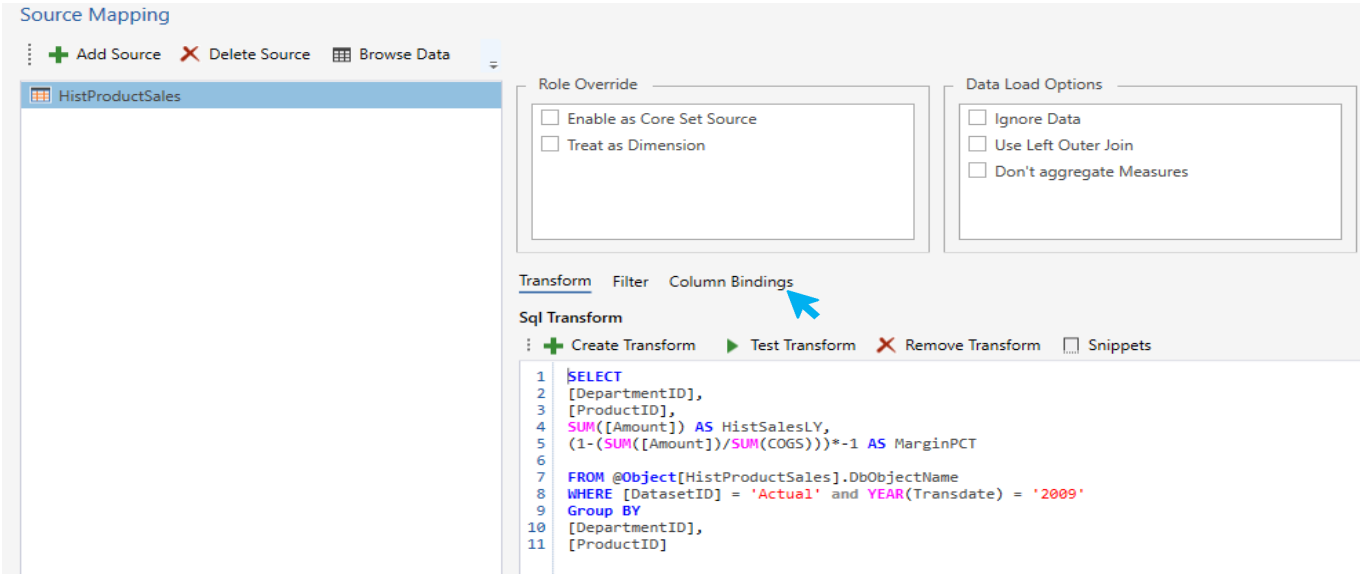
1. Go to the “Source Mapping” tab
2. Click on the “Add Source” button
3. Find the Fact table “HistProductSales” in the Item Picker pop up and click “Ok”

Notice that the data in the HistProductSales table is not in the same format as the data store we just created. One alternative is to use the SQL Transform field to write SQL code that transform your data. Another option is to create a view and use the view as your data source.

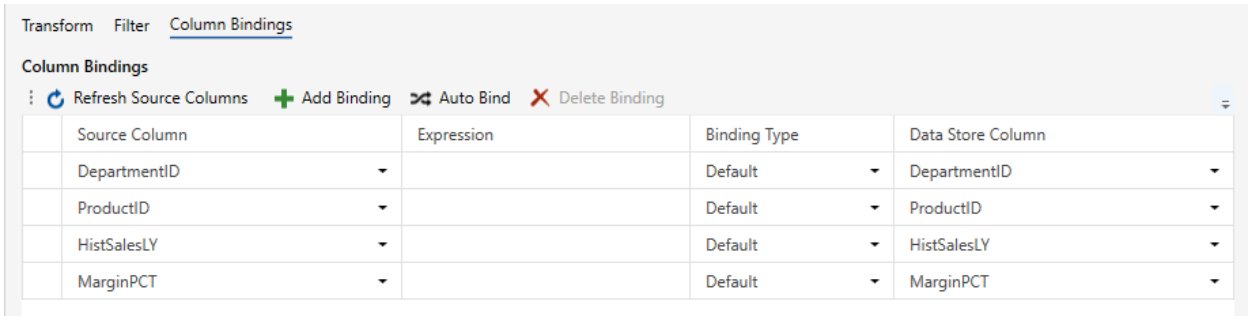
You can copy the SQL statement below into the Sql Transform window. We do this to create a column with margin % based on the data in the fact table.

```
SELECT
[DepartmentID],
[ProductID],
SUM([Amount]) AS HistSalesLY,
(1-(SUM([Amount])/SUM(COGS)))*-1 AS MarginPCT

FROM @Object[HistProductSales, fact].DbObjectName
WHERE [DatasetID] = 'Actual' and YEAR(Transdate) = '2009'
Group BY
[DepartmentID],
[ProductID]
```

4. Paste the SQL query into the SQL Transform field
5. Click “Save” or Ctrl + S
6. Navigate to the “Column Bindings” area



7. Map your Source columns to your Data Store columns or use the “Auto Bind” function. Your mapping should look like this:



8. Click “Save” or Ctrl + S
9. Click on the “Add Source” button. Select the “Department” Dimension in the Item Picker pop-up and click “Ok”
10. Use the “column Bindings” function to map the Dimension column DepartmentID against Data Store column DepartmentID.

    Make sure the binding type is set to “Relationship”.
11. Repeat step 9 and 10 for the “Product” Dimension
12. Click “Save” or Ctrl + S

Your Data Store is mapped against the source it needs in this course.

13. Right click on your Data Store “Product Sales” in the Solution Explorer pane on the right-hand side and select “Reload Data”. Click “Yes” in the Reload Data pop-up.

    This uses the query you entered in the SQL Transform field to load the selected data from the fact table into the data store.
14. When the “Reload data” dataflow is completed close the Data Flow execution dialog box.
15. Right click on your Data Store “Product Sales” and select “Browse Data” to explore the content of your Data Store

8 Worksheet - Product Sales

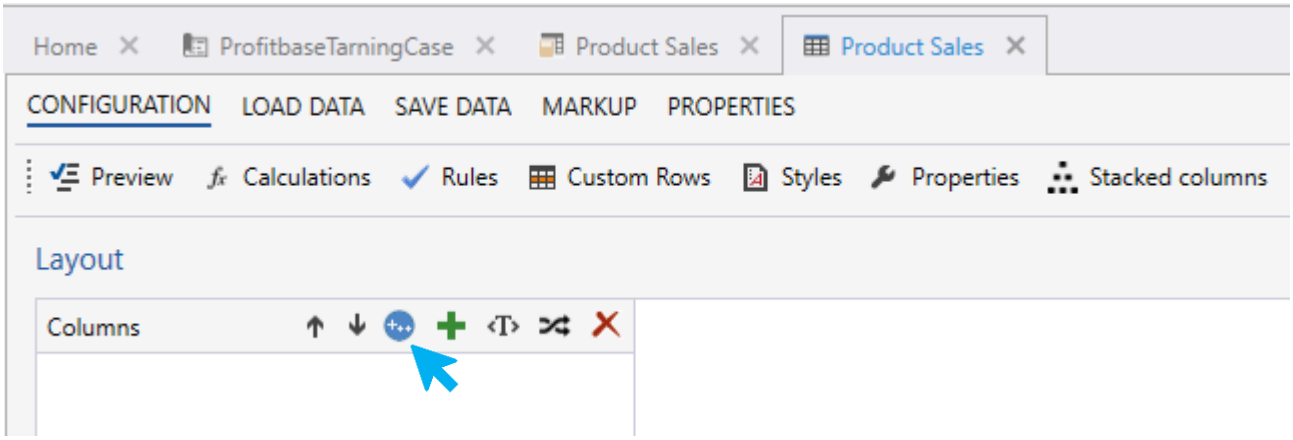
The contents of Data Stores are edited using Worksheets, which have an Excel-like look and feel. You would normally have just one Worksheet per Data Store, but you can have as many as you like. As long as any primary key column of the Data Store is bound to the Worksheet, the other columns are optional.

A Worksheet is always associated to just one Data Store, and once it has been created, you cannot change the binding. You can however, include data from other sources such as tables, views, stores, dimensions or fact tables to make calculations and save that value to the connected datastore.

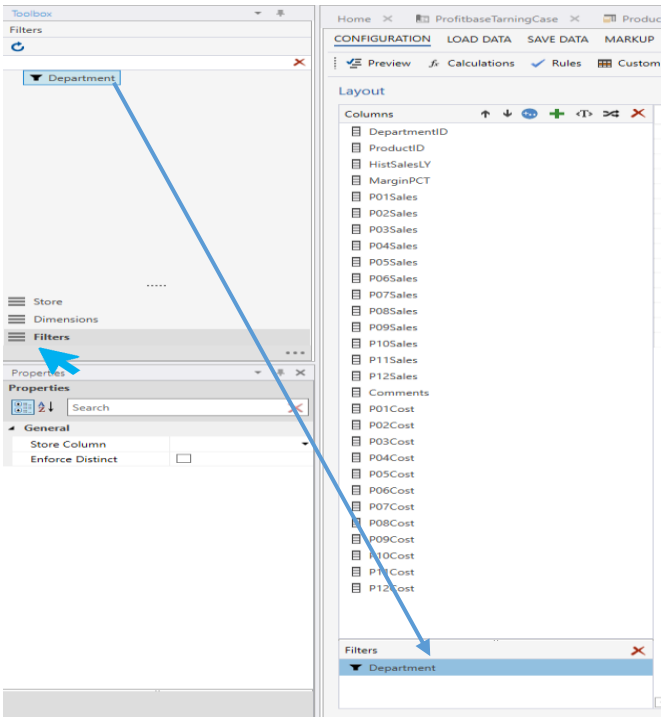
A Worksheet must be added to a Workbook to make it available to the end user. Once a Worksheet has been added to a Workbook, users can view, edit, and save data to the datastore through this worksheet.

Let's create a worksheet for our product sales data:

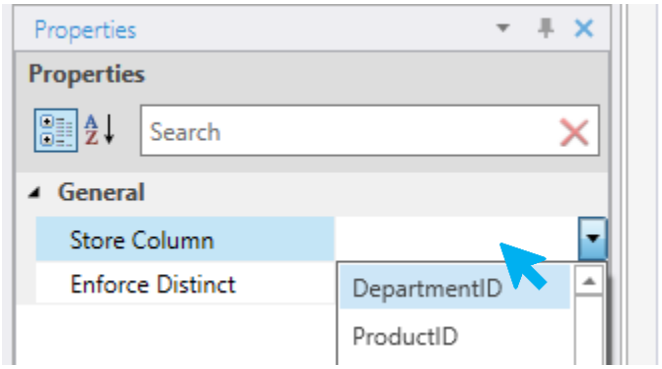
1. Right Click on the Data Store "Product Sales" and select "New Worksheet"
2. Name the Worksheet "Product Sales" and click "Ok"
3. Click on the "Add Columns for Data Store" button



4. Check the "Select all" field in the Store Columns pop-up and click "Ok"
5. Find the Department Filter in the Toolbox on the left-hand side and drag it to the Filters field under Layout:



6. Select “DepartmentID” as Store Column under “Properties”



7. Click “Save” or Ctrl + S

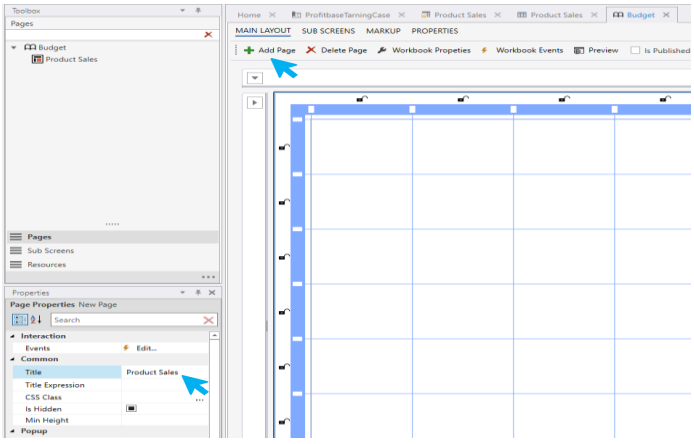
9 Workbook and pages

Workbooks are the end user interface of Profitbase LOW CODE. A Workbook consists of one or multiple pages, and each page contains one or more components, for example Worksheets, Reports, Filters, Buttons, Labels etc.

A Workbook also has left, top and right sections called dock panes, which can be used for hosting components that interact with components across many pages, for example global filters.

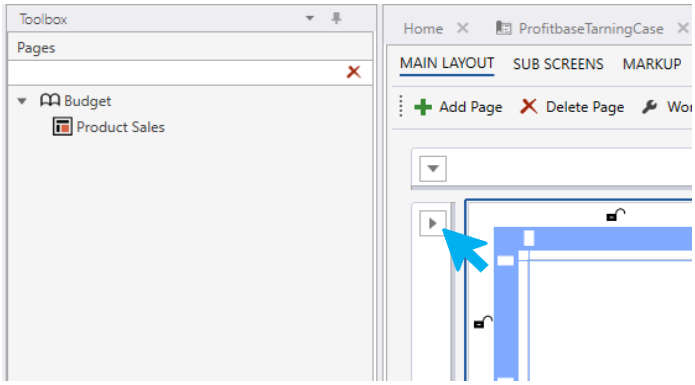
Let’s create a workbook:

1. Right Click on the Data Store Group “Budget” and select “Add New Item…”
2. Choose the item “Workbook” and name the workbook “Budget” then click “Ok”
3. Click on the “Add Page” button in the Main Layout tab
4. Enter the page name, “Product Sales”, in the “Title” field under Properties on the left.

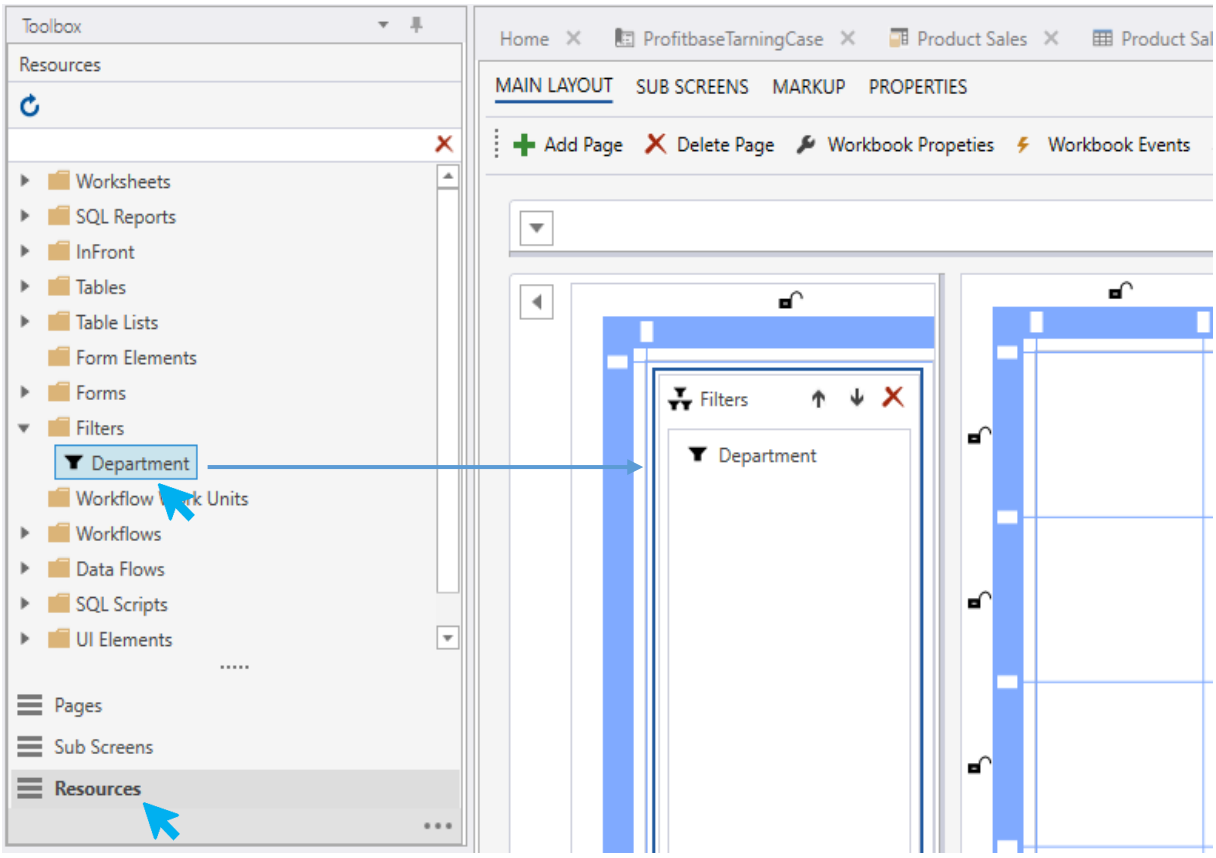


9.1 Adding the department filter

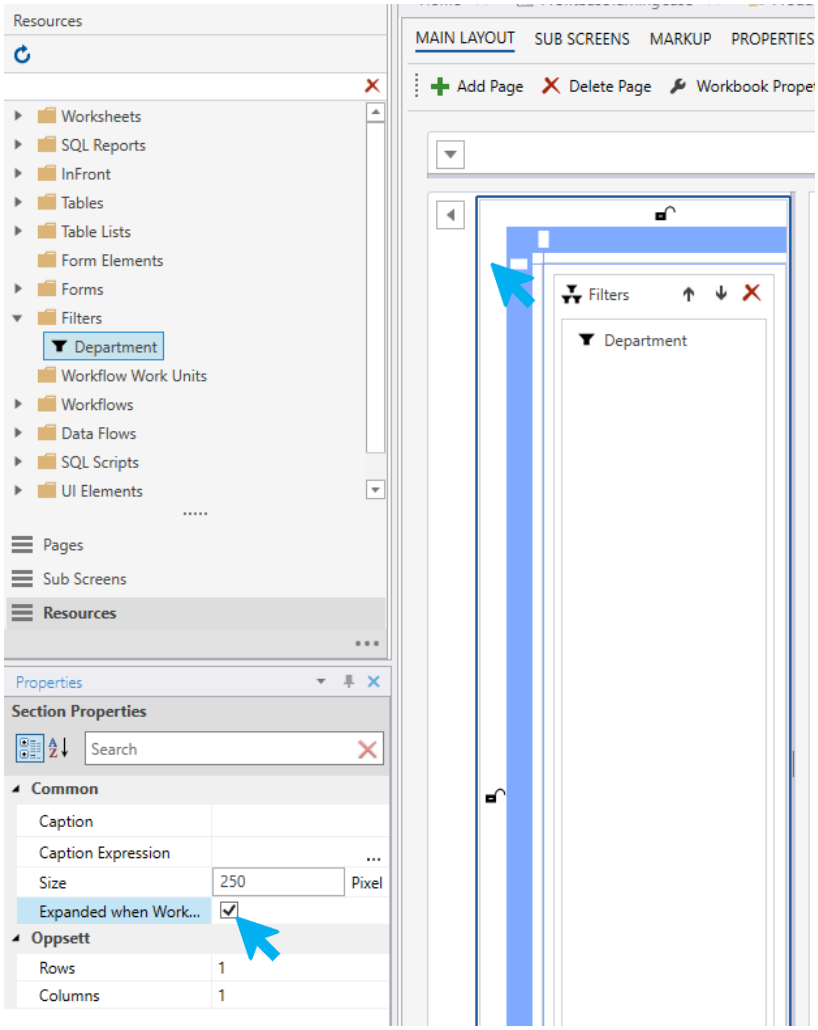
1. Expand the left dock pain by clicking on the expand button



2. Select “Resources” in the toolbox on the left-hand side and find the Department Filter
3. Drag the Department Filter into the left dock pane and “Save”

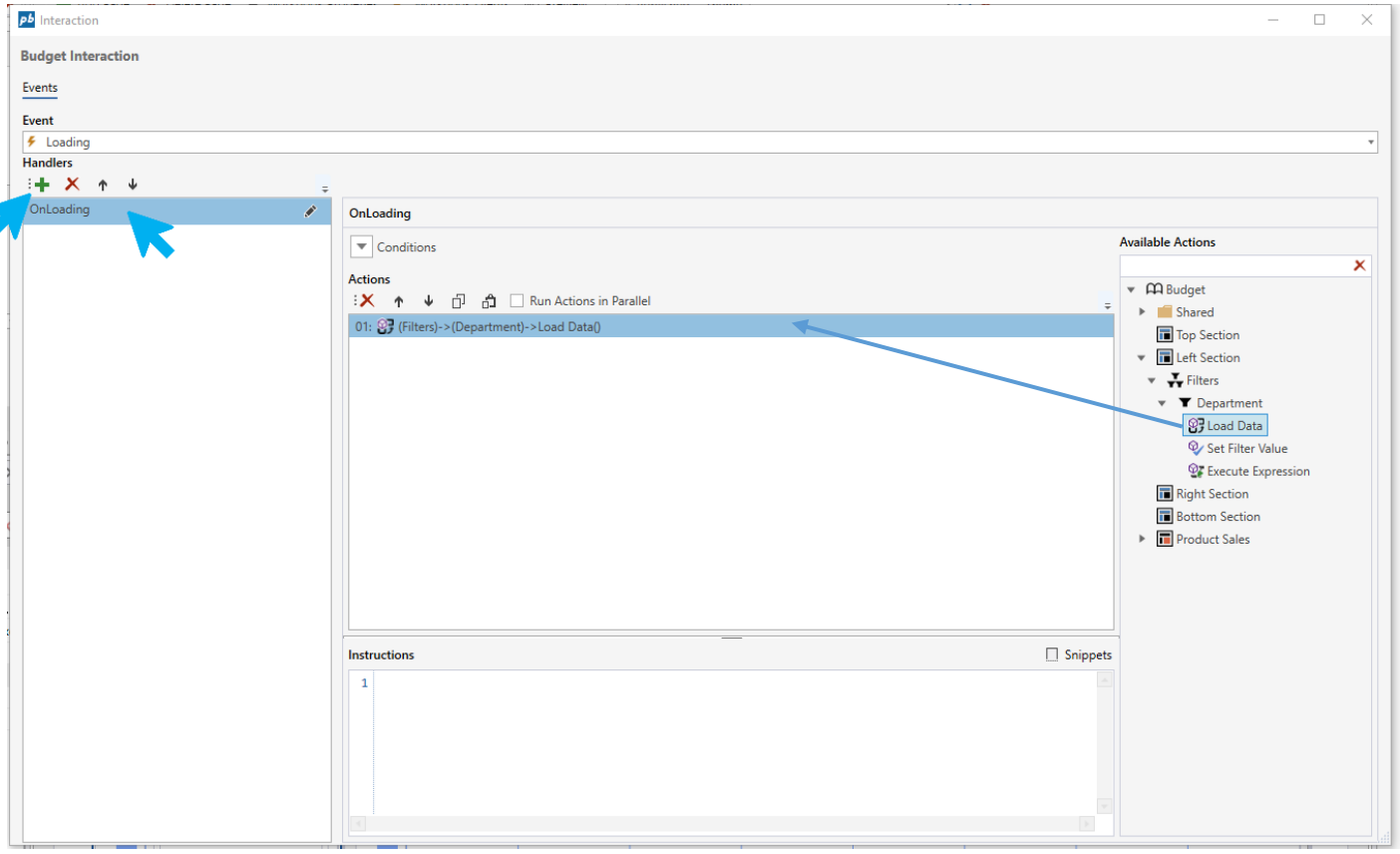


4. Click on the border of the left dock pane to see the respective Properties
5. Check the “Expanded when Workbook opens” property and set “Size” to 250 pixels



6. Click “Save” or Ctrl + S
7. Click on the “Workbook Events” button

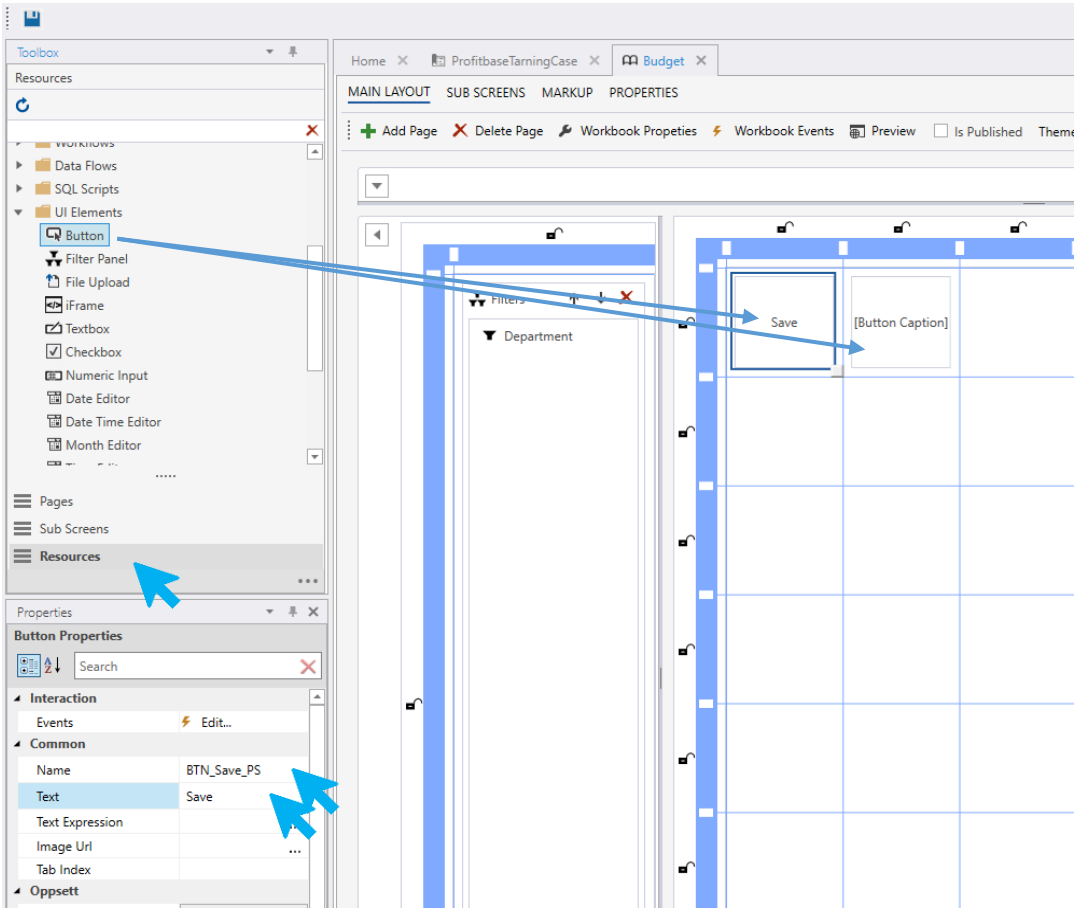
8. Select the “loading” event and add a new Handler by clicking on the “Green Pluss Sign”. Then select the “OnLoading” handler by clicking on it.
9. Drag and drop the “Load Data” action under the Department filter in the available actions, into the Actions Field



10. Close the Interaction pop up and Save

9.2 Adding buttons

1. Expand the UI Elements folder under Resources in the Toolbox
2. Drag and drop the button element into the top left corner of your workpage

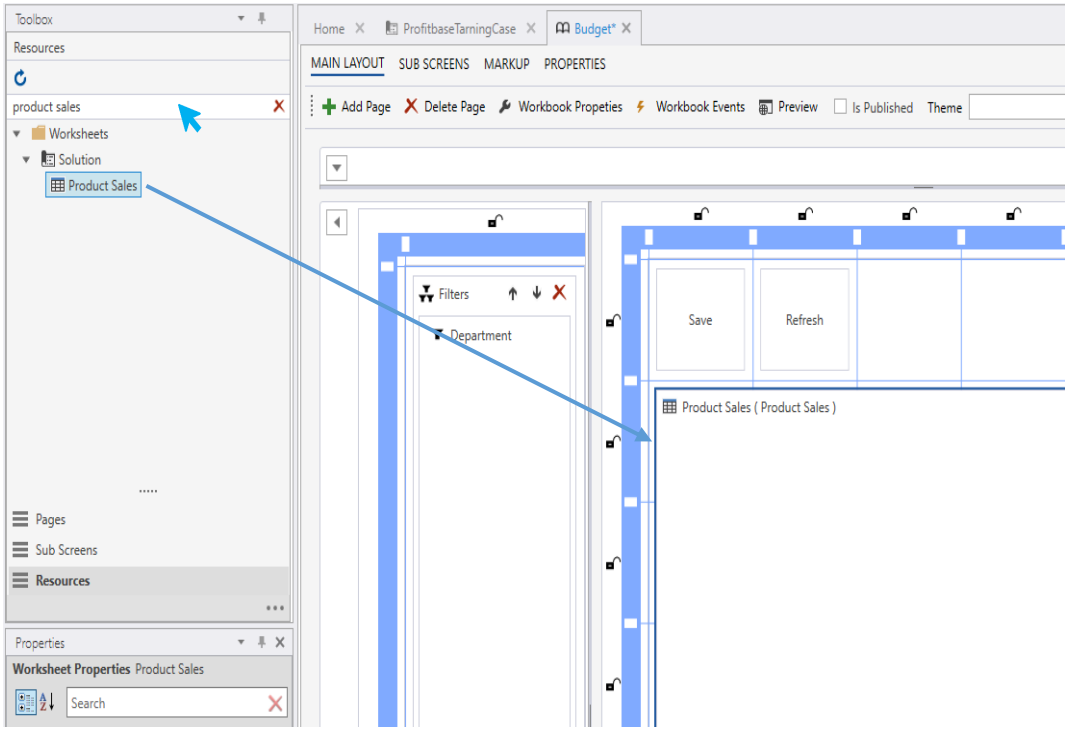


3. Fil out the Name and Text field in the properties window in the lower left corner. Make sure your new button is selected first.
   The Name field is the name of the element. The Text field is what would be displayed in the web browser. A good way of naming your elements: BTN_Save_PS
   BTN = short for button
   Save = what it does
   PS = Reference to Product Sales workpage
4. Click Save or Ctrl +S
5. Repeat step 2 to 4 for a Refresh button next to the Save button.

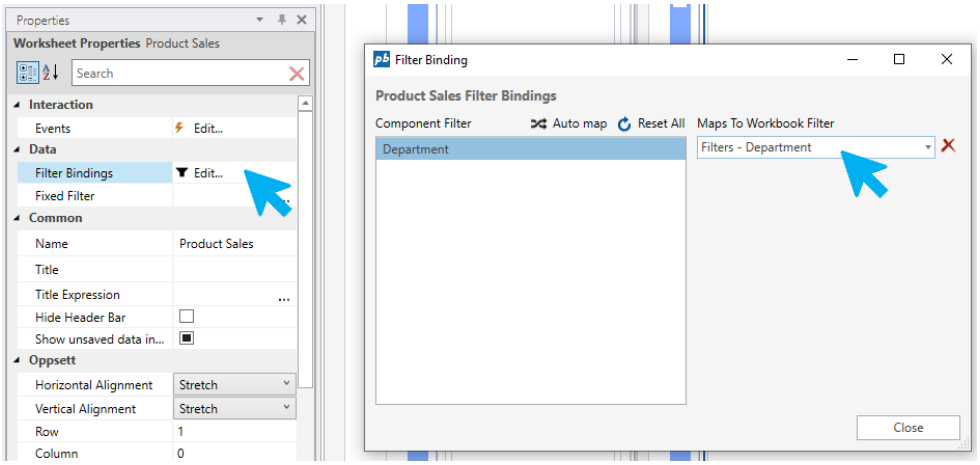
Now we have two buttons in our workpage, but they don't do anything yet. We will come back to that after we have included a worksheet in our workpage.

9.3 Adding a worksheet

1. Use the search field at the top of your toolbox. Enter “Product Sales” to find your work sheet.
2. Drag the worksheet into your workpage and resize it to fill the entire page below the buttons



3. Make sure the worksheet is selected and go to “Filter Bindings” under Properties and click “Edit...”
4. Select “Filters – Department” in the dropdown menu and close the Filter Binding pop up
5. Click Save or Ctrl + S



9.4 Configure workpage actions

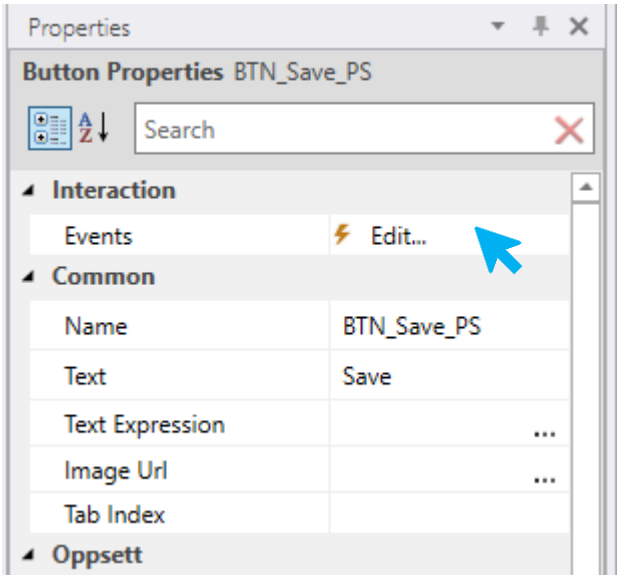
There are a few different actions we need to set up in this solution.

- Saving data entered in the worksheet when the Save button is clicked
- Loading the worksheet when we click refresh
- Loading the worksheet when the filter selection is changed
- Loading the worksheet when we navigate to the Product Sales page

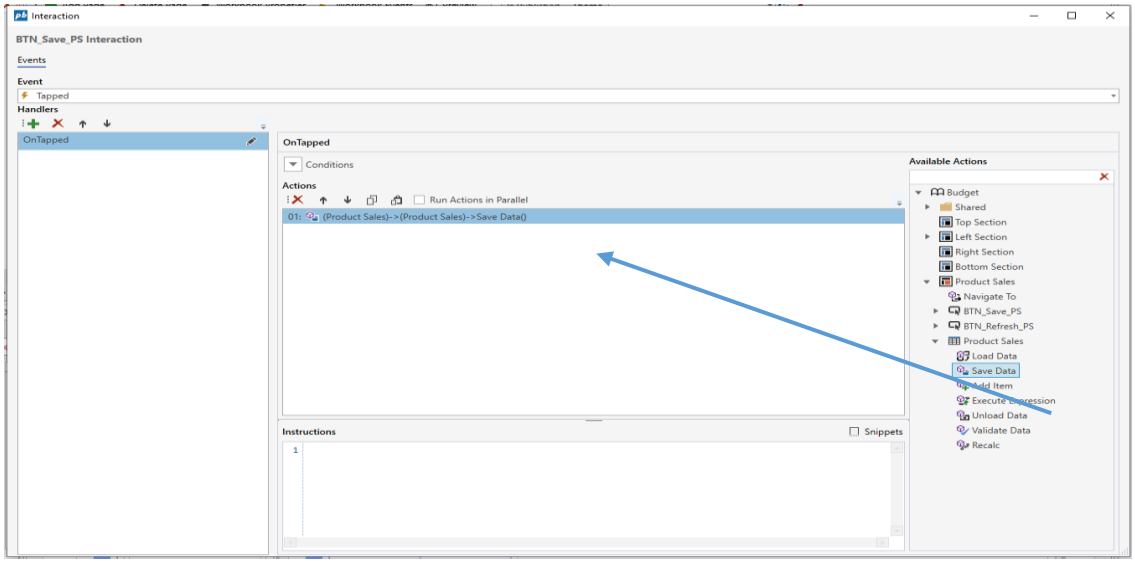
All three of the loading actions are the same. In this case we recommend creating the loading action on the refresh button and referring to this action (Invoking) in the two other cases where we want to load the worksheet. That way, when you need to make changes to the loading actions, you only need to do it once.

9.4.1 Configure a save action

1. Select the “Save” button in your workpage
2. Click the “Edit…” button for Events under Properties



3. Click on the green “Pluss sign” to create a new Handler for the “Tapped” event
4. Expand the Product Sales workpage under Available Actions in your Interaction pop up.
   Expand the Worksheet Product Sales to find the “Save Data” action
   Drag the “Save Data” action into the Actions field

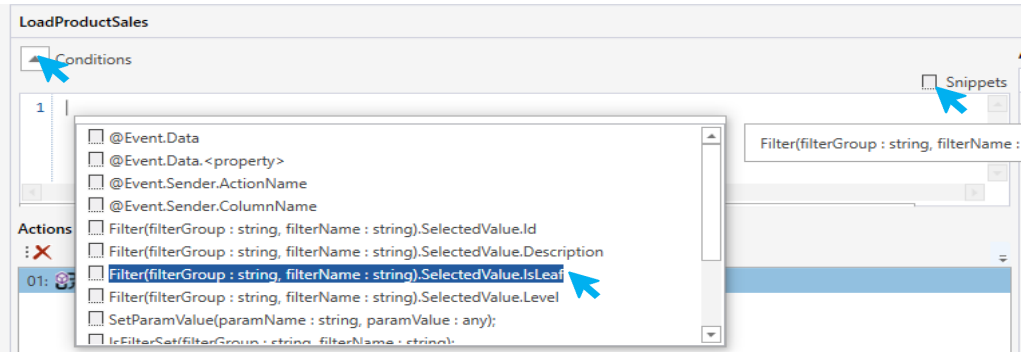


The data will now be saved when we click on the save button. In this case we don’t want to allow the users to save any changes above the lowest level of the department hierarchy (Leaf). We therefore include a condition that makes sure the saving action is not run unless this condition is fulfilled.

5. Expand the “Conditions” field by clicking on the Expand button
6. Click on the “Snippets” button
7. Select “Filter(filterGroup : string, filterName : string).SelectedValue.IsLeaf” by double clicking
8. Edit the condition statement to look like this:

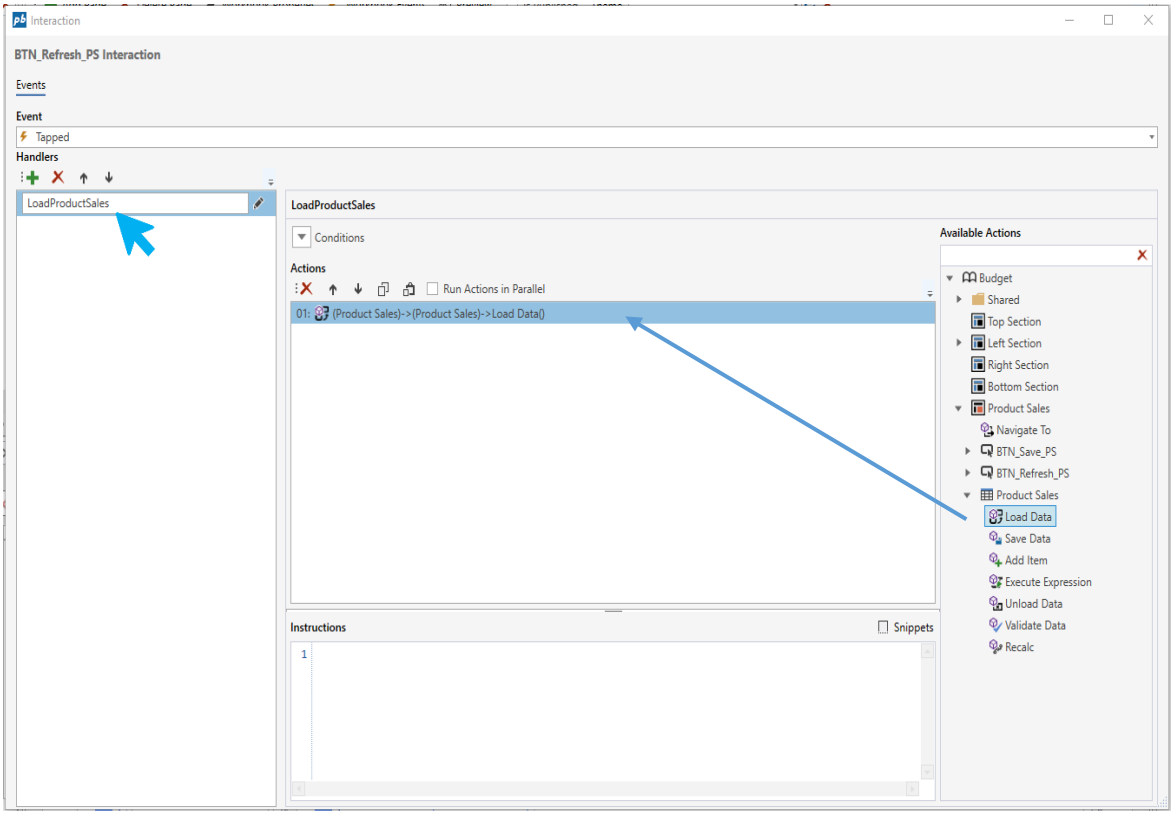
   ```
   Filter("Filters", "Department").SelectedValue.IsLeaf == true;
   ```

9. Close the Interaction pop-up and Save



9.4.2 Configure load action

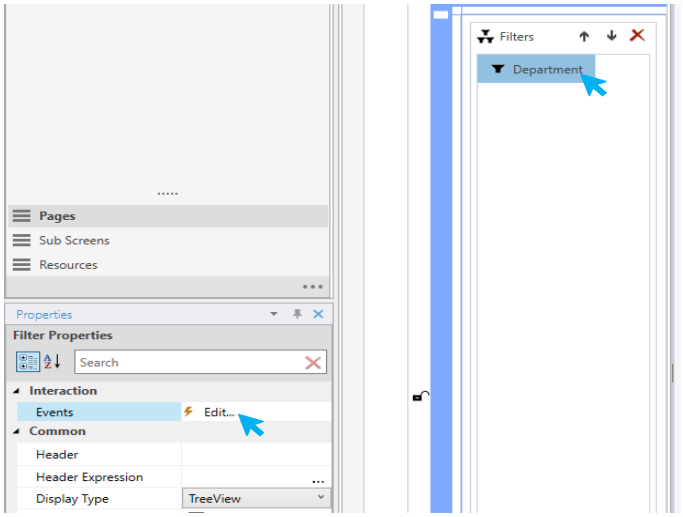
10. Select the “Refresh” button in the workpage
11. Click the “Edit…” button for Events under Properties
12. Click on the green “Pluss sign” to create a new Handler for the “Tapped” event
13. Use the edit symbol on the “OnTapped” Handler to rename it “LoadProductSales”
14. Drag the “Load Data” actions into Actions field



9.4.3 Invoke other actions

Let's start by invoking our load action from the refresh button when we change our Department filter.

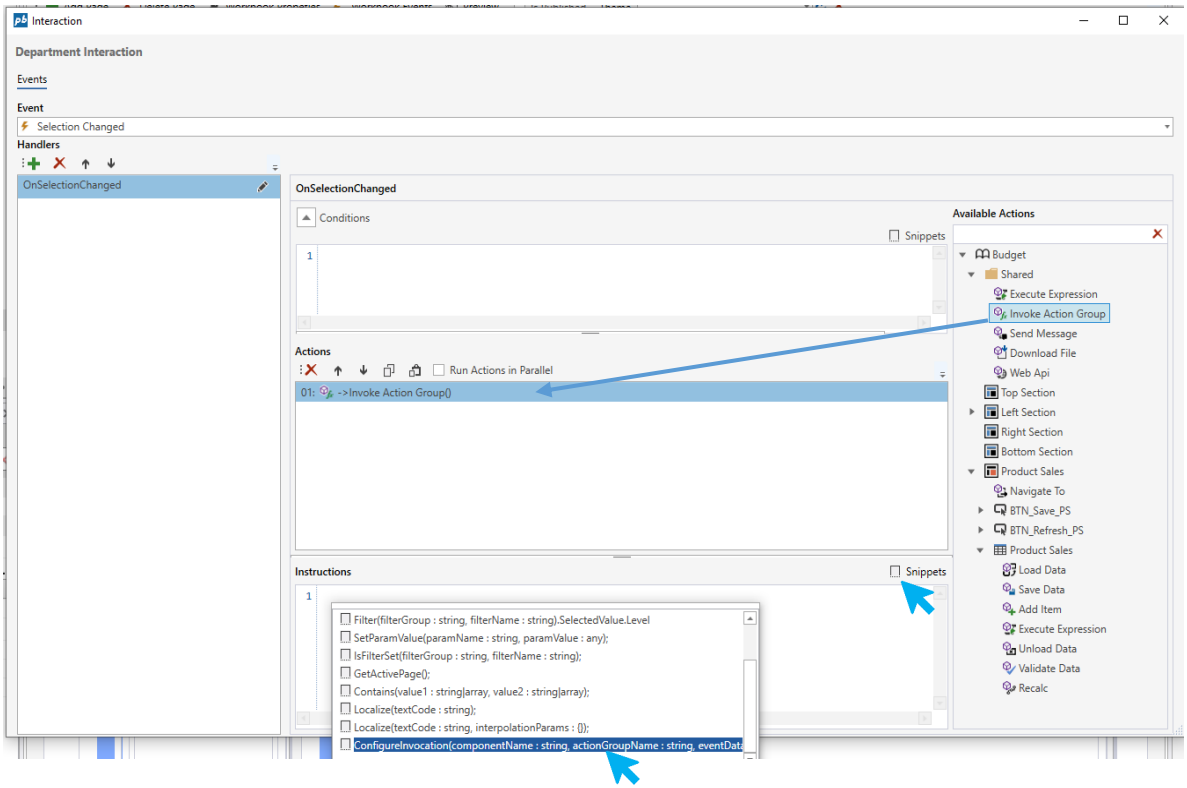
1. Click on the Department Filter
2. Click the "Edit…" button for Events in the Properties field



3. Click on the green “Pluss sign” to create a new Handler for the “Selection changed” event
4. Expand the Shared Folder under Available Actions
5. Drag the “Invoke Action Group” into the Actions field

   This can be used to trigger an event handler from another place in the solution. The refresh button in this case.

6. Click on the “Snippets” by the Instructions field and select:
   ConfigureInvocation(componentName : string, actionGroupName : string, eventData : any);



7. Edit the instruction statement to the following:

   ConfigureInvocation("BTN_Refresh_PS", "LoadProductSales", null);

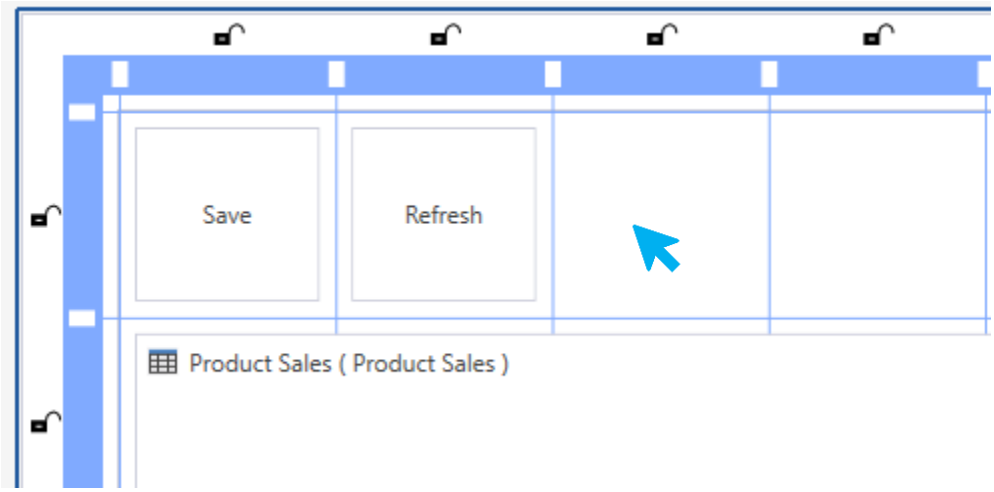
   BTN_Refresh_PS: Refers to the element where the action is located.

   LoadProductSales: Refers to the name of the event handler we want to trigger.
8. Close the Instruction pop up and save

Your worksheet will now reload when you select a different department in the department filter. Conditions and instructions applied to the event handler we trigger will still work. Note that any unsaved changes will be lost when the worksheet reloads.

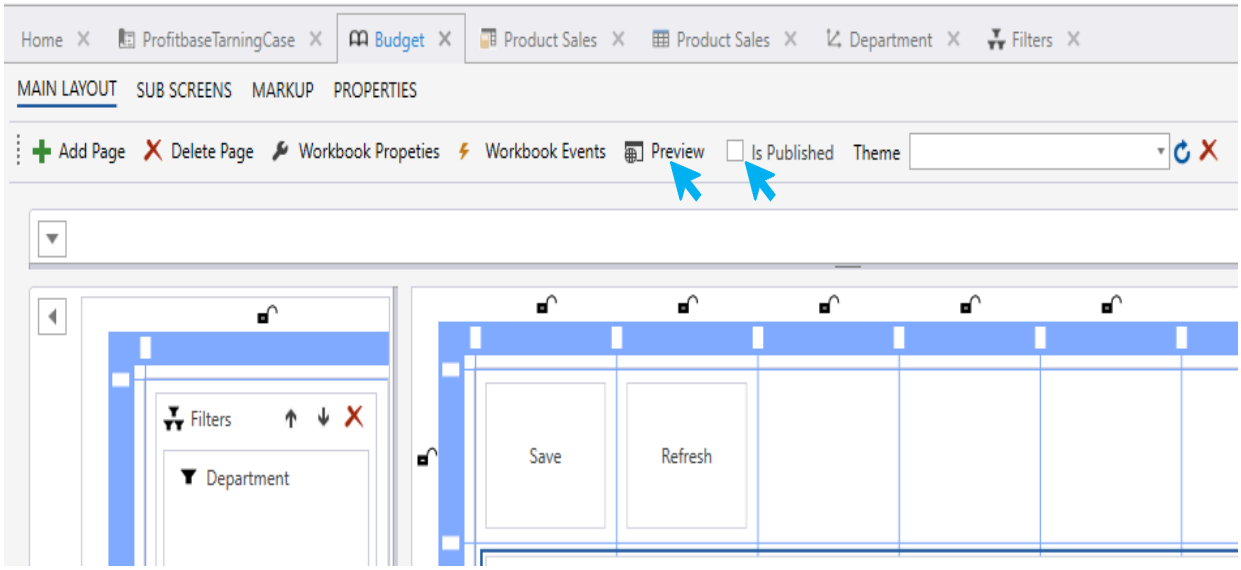
Let's configure loading when we navigate to the workpage. This is especially convenient when we have several workpages to navigate between.

1. Select the workpage by clicking on an empty cell



2. Click "Edit…" button for Events in the Properties field
3. Select "Navigated to" in the Event drop down field
4. Repeat the step 3 to 8 in the instructions above for invoking the load action on selection changed in the filter

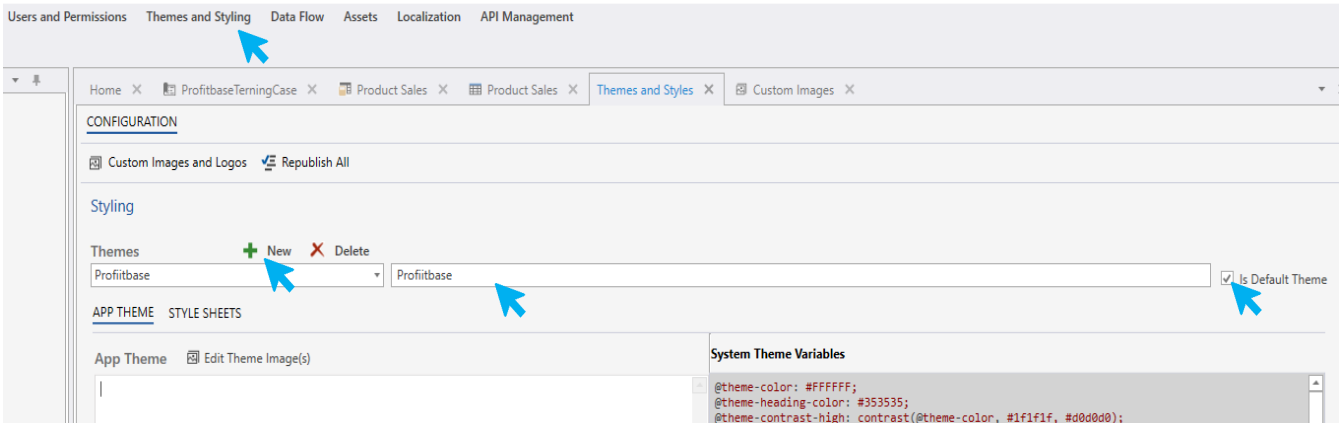
Your Workbook is now created. You can preview it with the "preview" after the "Is Published" checkbox is checked off and saved.



10 Themes and Styles

1. Click on the “Themes and Styling” tab
2. Click on the “New” button
3. Name your style Theme
4. Check the “Is Default Theme” check box

   This tells the system to use this Theme for the styling of the landing page.
5. Save

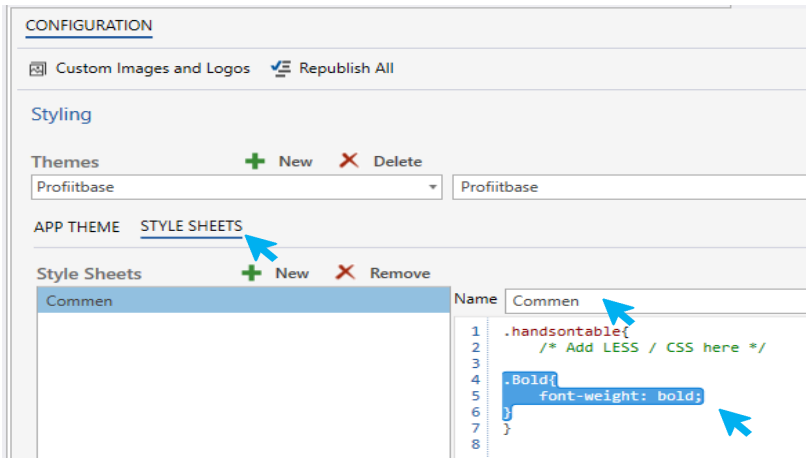


Profitbase LOW CODE uses CSS to style and format our solution. We are going create some CSS classes.

1. Go to the Style Sheet tab
2. Click on the “New” button and add a “Data Grid Style Sheet”
   a. We use this to style components within a worksheet, report or table like column og text color. These classes are surrounded by a “handsontable” class. Make sure you properly close all new classes separately and the handsontable class at the end.
3. Name it “Common”. *(“Commen” is a typo, but it doesn’t really matter.)*
4. Then add Bold CSS class:

```
.Bold{
   font-weight: bold;
}
```

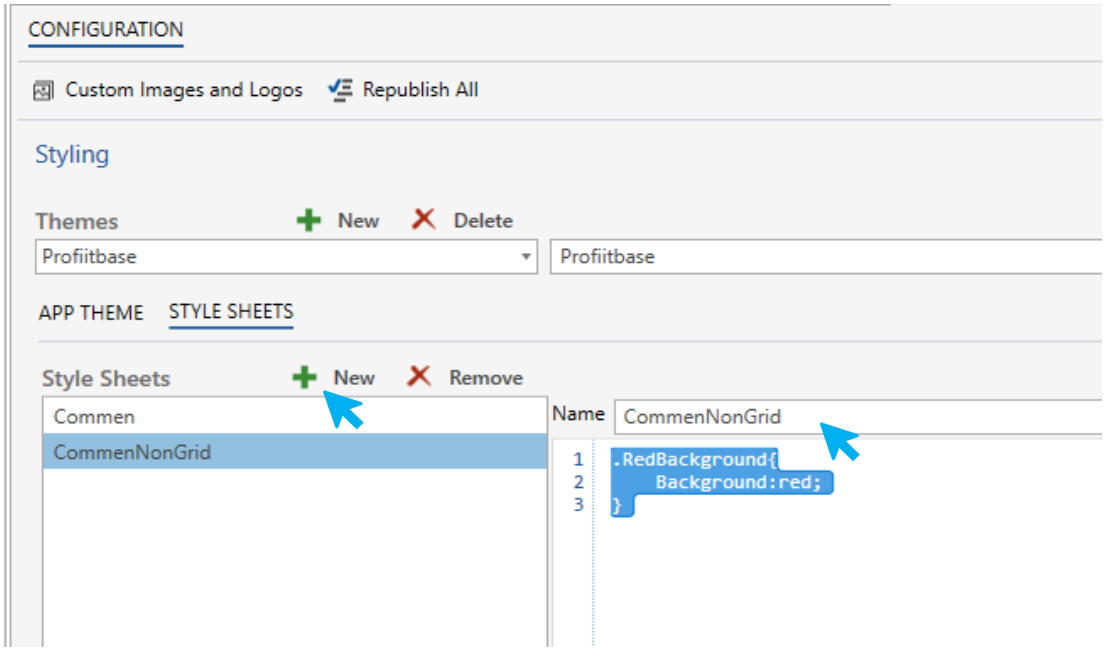
5. Your Configuration should look like this:



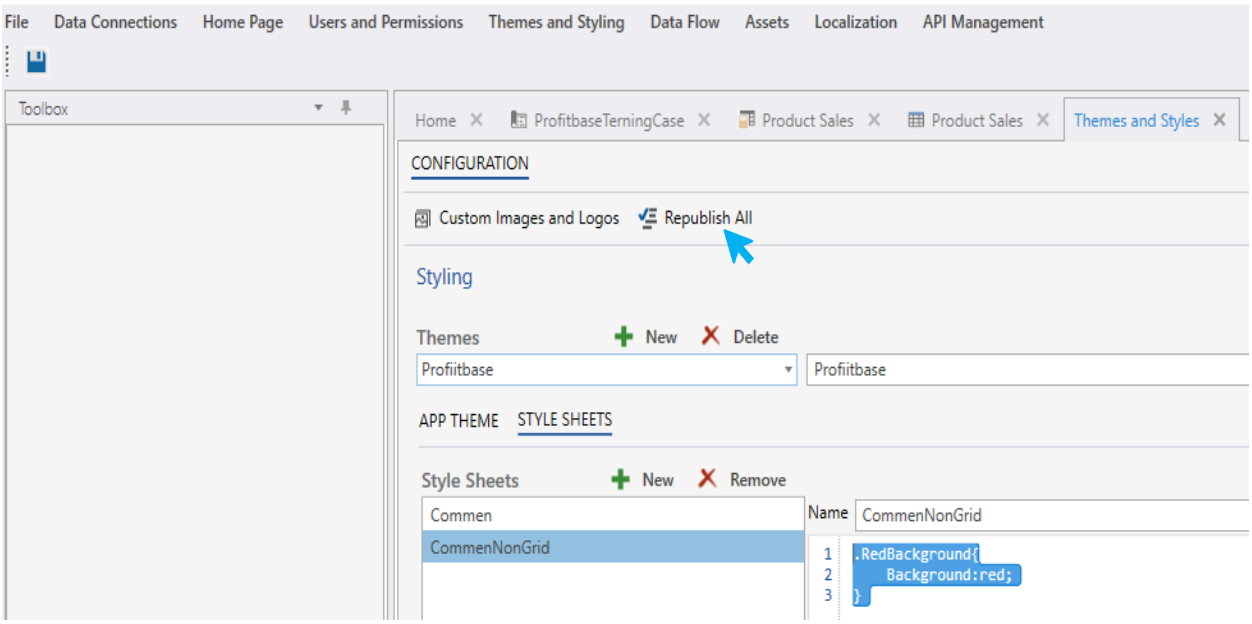
6. Click on the “New” button and add a “Generic style sheet”
    a. We use this to style components outside of the grid components like buttons and backgrounds etc.
7. Name it “CommonNonGrid”
8. Then add RedBackground CSS class:

```
.RedBackground{
  Background:red;
}
```

9. Your Configuration should look like this:



10. Save
11. Click on “Republish All”


File
Data Connections
Home Page
Users and Permissions
Themes and Styling
Data Flow
Assets
Localization
API Management
Toolbox
Home
ProfitbaseTerningCase
Product Sales
Product Sales
Themes and Styles
CONFIGURATION
Custom Images and Logos
Republish All
Styling
Themes
New
Delete
Profiitbase
Profiitbase
APP THEME
STYLE SHEETS
Style Sheets
New
Remove
Commen
CommenNonGrid
Name
CommenNonGrid
1 .RedBackground{
2 Background:red;
3 }

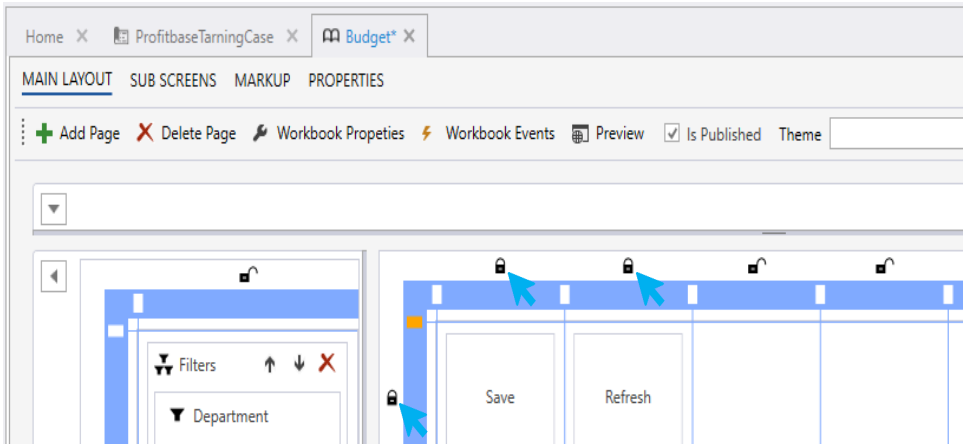
11 Making the Workbook look nice

After you have previewed your Workbook it becomes clear that it needs some work to look nice.

The first thing we do is to select our new Theme in the dropdown list next to the Is Published checkbox in our workbook components. This tells the workbook to use the style configuration from that specified Theme.

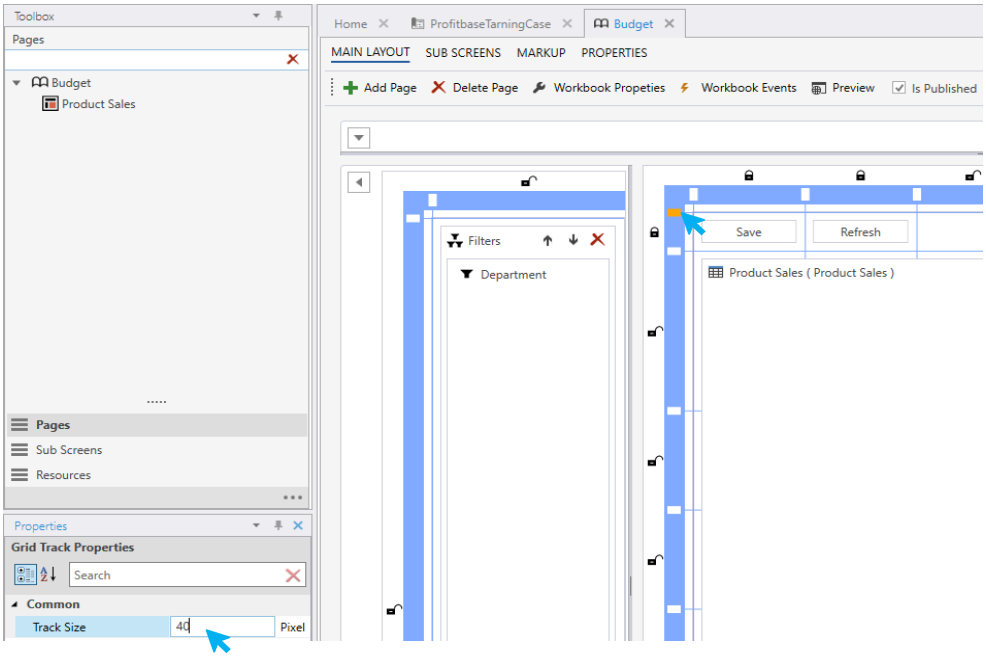
11.1 Sizing Elements in your workbook

We will start by defining appropriate space for our buttons.

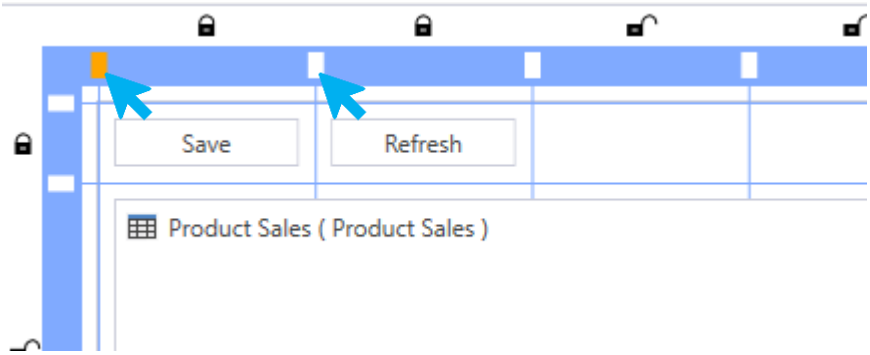


Start by clicking on the three padlocks shown in the picture until they display as locked.

Click once on the small rectangle next to data grid line (see screenshot below). The Properties field will display track size. Change it to 40 pixels.

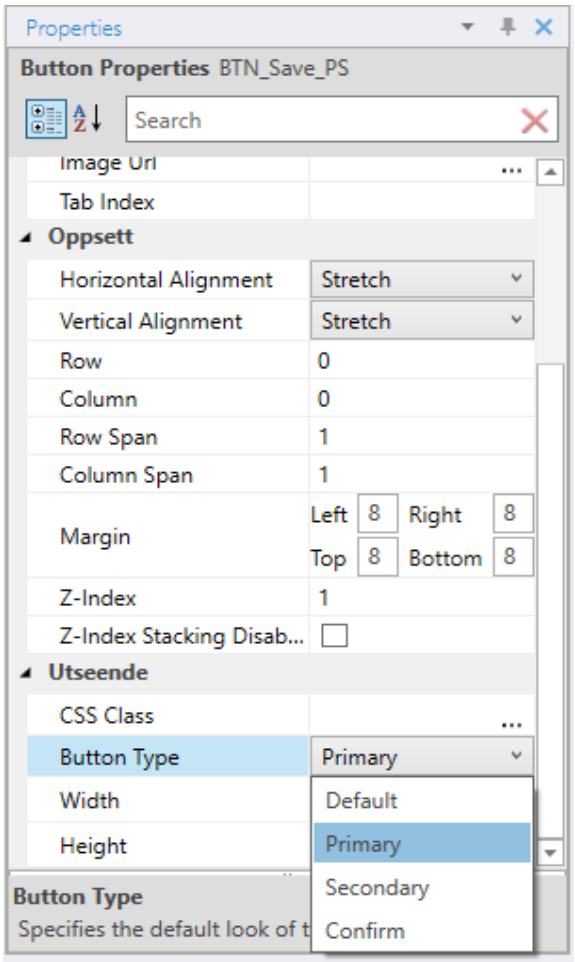


Set the Track Size to 90 pixels for the two vertical grid lines. Then click Save or CTRL + S



11.2 Set button type

1. Click on the “Save” button so the Properties field displays the save button’s properties.
2. Change Button Type from “Default” to “Primary”
    a. These are default CSS classes



3. Click Save or CTRL+ S
4. Click on the “Refresh” button to display the Refresh buttons properties
5. Change Button Type from “Default” to “Secondary”
6. Click Save or CTRL+ S

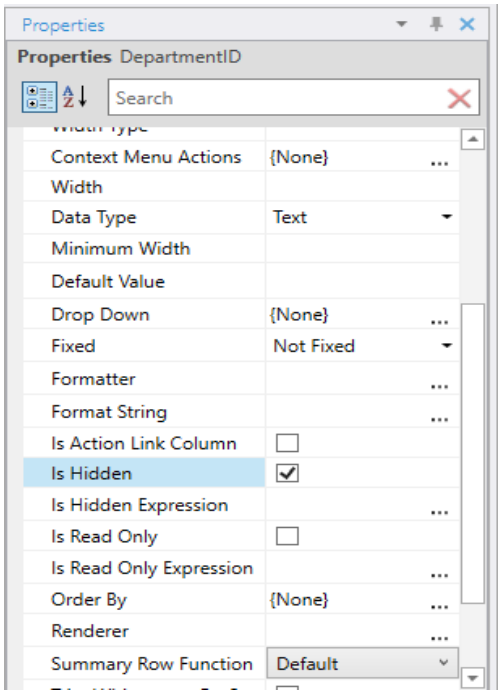
11.3 Cleaning up the worksheet

Next, we are going to clean up the worksheet. We need to hide the department column, make some columns locked for editing (ready only), create some more friendly column names and create some columns with calculations.

11.4 Creating friendly names

Start by navigating to your worksheet “Product Sales”. You will find it in the Solution Explorer.

1. Click on the “DepartmentID” column to display the column properties
2. Scroll down until you find the “Is Hidden” check box and check it.
   The column is no longer visible in the solution.



3. Click on “ProductID” column to display the column properties.
4. Change the “Caption” from “ProductID” to “Product”

   This is the column header.

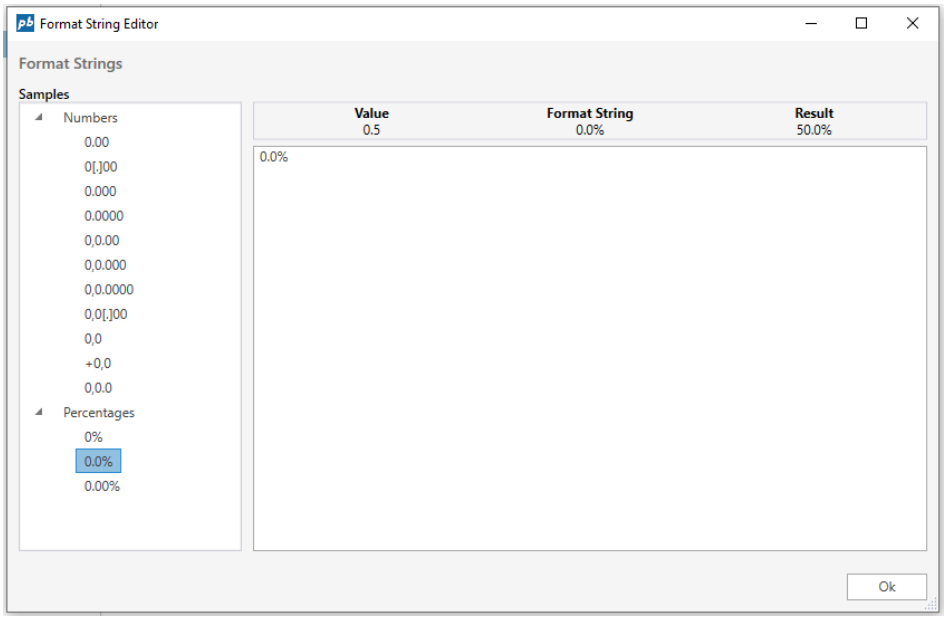
5. Scroll down to the “Is Ready Only” check box and check it
   The column is no longer open for editing.

   This is generally a good idea for key columns as we cannot have duplicate values.

Change the columns as follows:

- HistSalesLY
  - Caption: Sales LY
  - Read only: true
- MarginPCT
  - Caption: Margin

6. Click Save or CTRL+S
7. Still looking at “MarginPCT” properties, scroll down until you find “Format string”
8. Click on “…” to edit formatting to percentage with one decimal

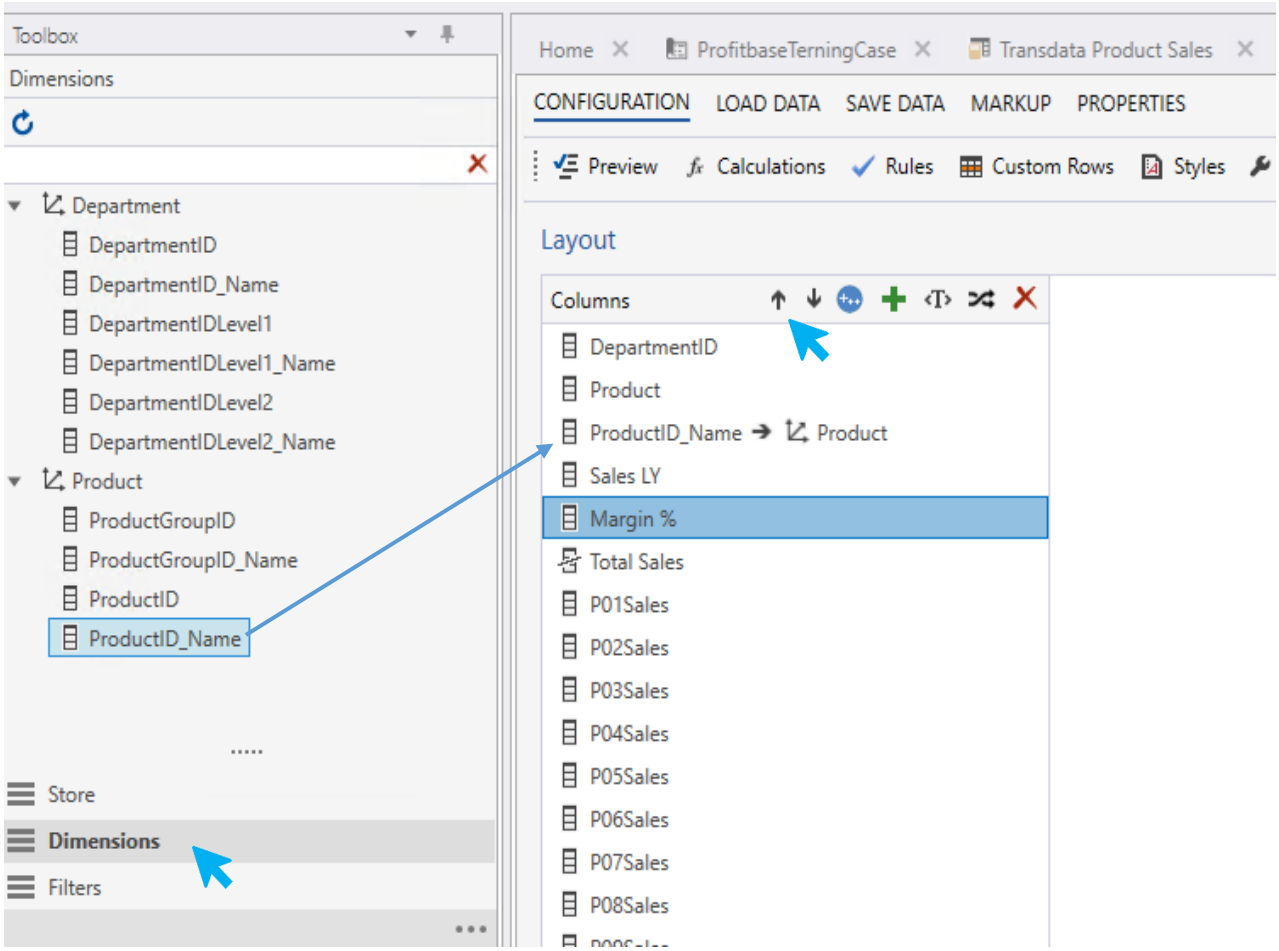


9. Click OK to close dialog box
10. Click Save or CTRL+ S

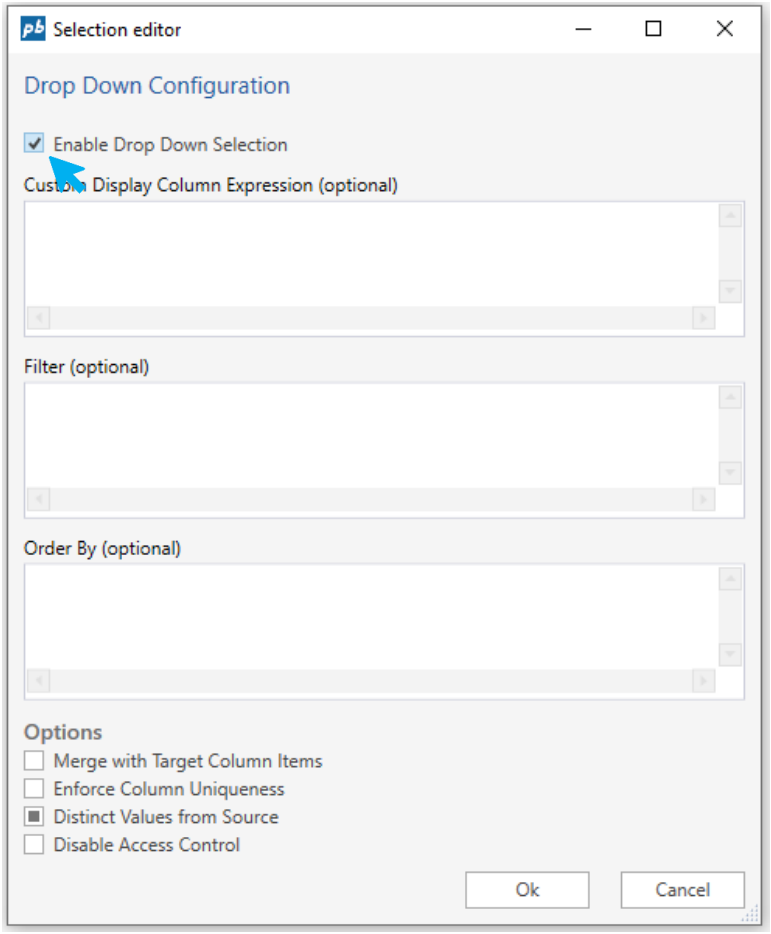
11.5 Using a dimension property in our worksheet

We are now displaying ProductID in our worksheet. A more user-friendly view will use name rather than ID for our product dimension. We have already linked the Product dimension when we created the datastore Product Sales.

1. Click on Dimension in toolbox on the left-hand side.
2. Drag and drop the “ProductID_Name” column into columns layout field
3. Use the arrows to place “ProductID_Name under ProductID



4. Set the Caption to “Product” for the ProductID_Name column
5. Click on “Drop Down Selection” field under properties in the left-hand side
6. Check the “Enable Drop Down Selection”



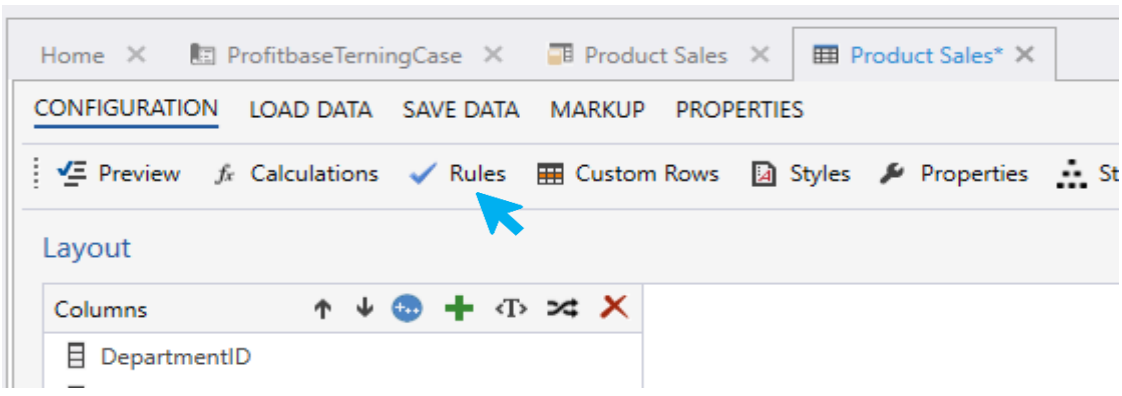
7. Click Save

11.6 Using Rules

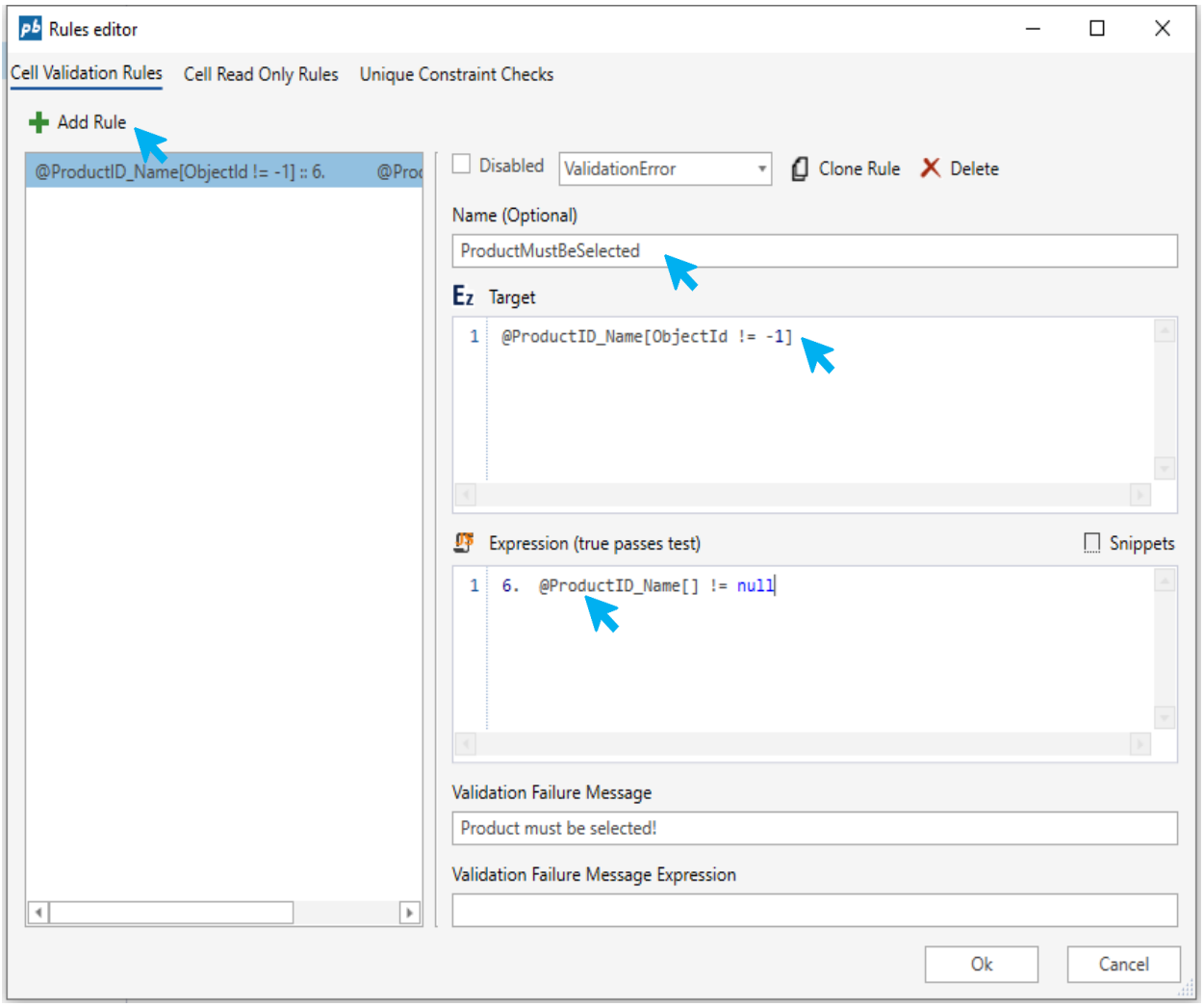
We are going to create one cell validation rule and one cell read only rule.

The validation rule is used to limit the user's ability to save changes when the conditions of the rule is not fulfilled. Similarly, the read only rule locks the cells it the condition is true.

1. Click on the "Rules" button

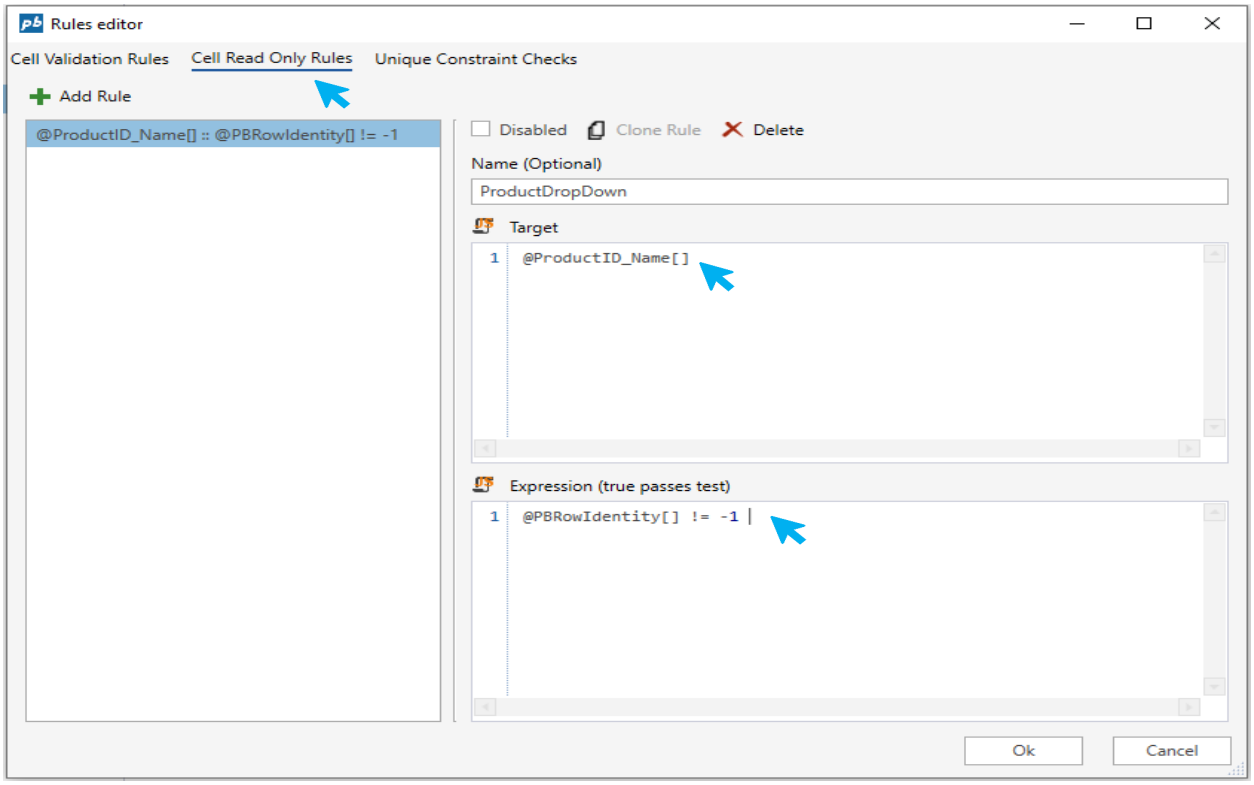


2. Click on the “Add Rule” button
3. Name the rule “ProductMustBeSelected”
4. File out the Target field
   @ProductID_Name[ObjectId != -1]
5. File out the Expression field
6. @ProductID_Name[] != null
7. Entre “Product must be selected!” in the Validation Failure Message field



This prohibits the user from saving changes if the productID column contains null (blank) values. This is important as key columns are not allowed to contain null values.

8. Select “Cell Read only Rules” tab
9. Click on the “Add Rule” button
10. Name the rule “ProductDropDown”
11. File out the Target field
    @ProductID_Name[]
12. File out the Expression field
    @PBRowIdentity[] != -1
13. Close the pop-up and Save



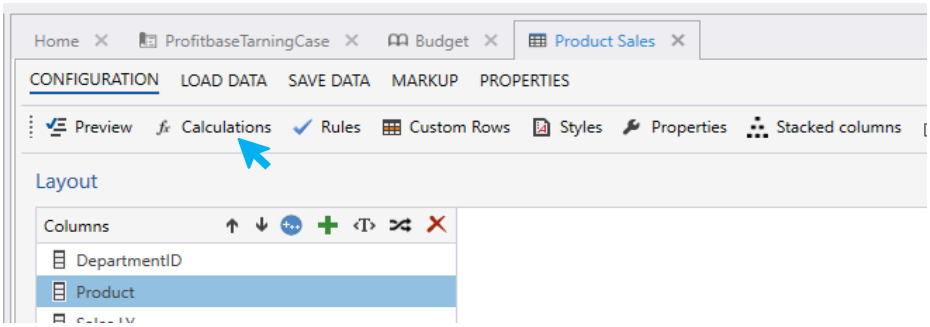
This locks the ProductID_Name column for editing for all rows except new rows when they are added. Note that after a new row has been saved, it will also be locked.

11.7 Creating a calculation

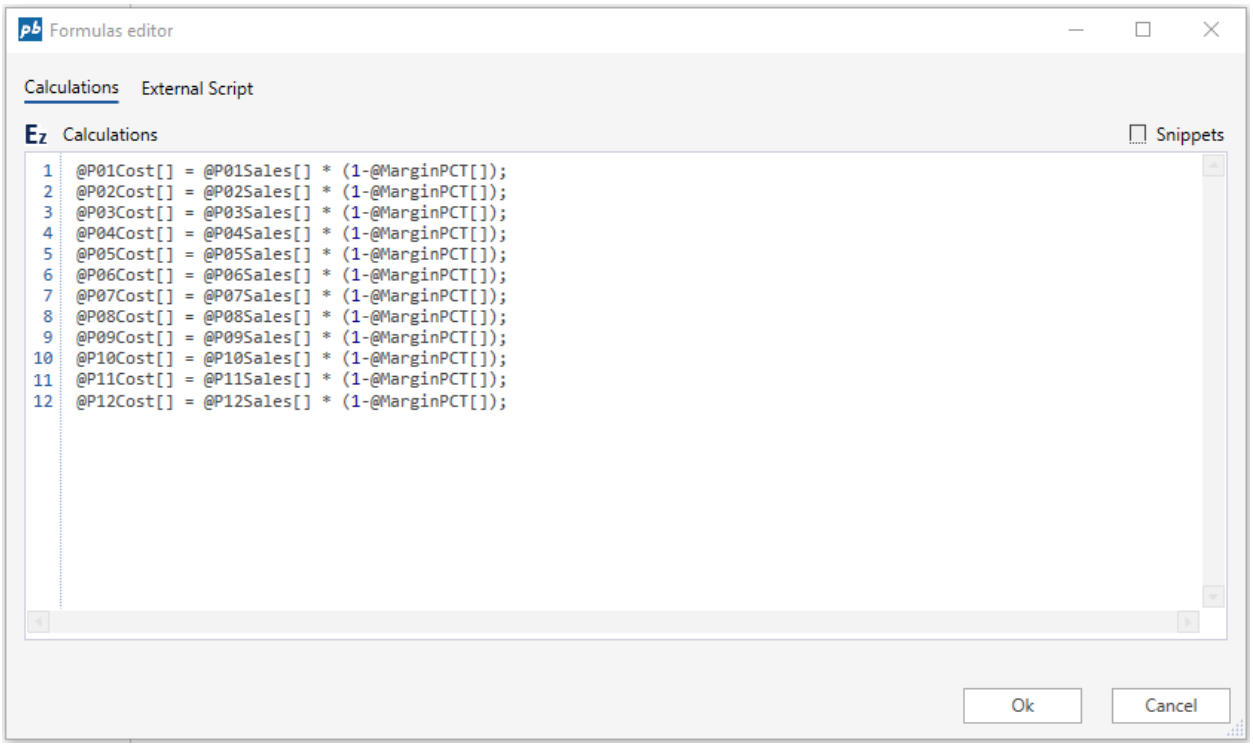
Next, we want the monthly cost columns to be calculations based on the margin % we have provided.

P01Cost = P01Sales * (1 - MarginPCT)

Click on the “Calculations” button on the Worksheet.



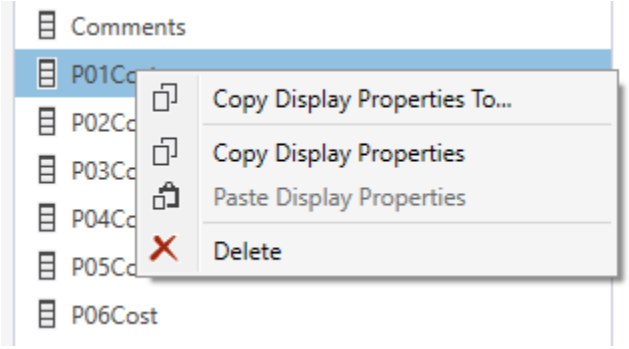
Edit the calculations to look like this:



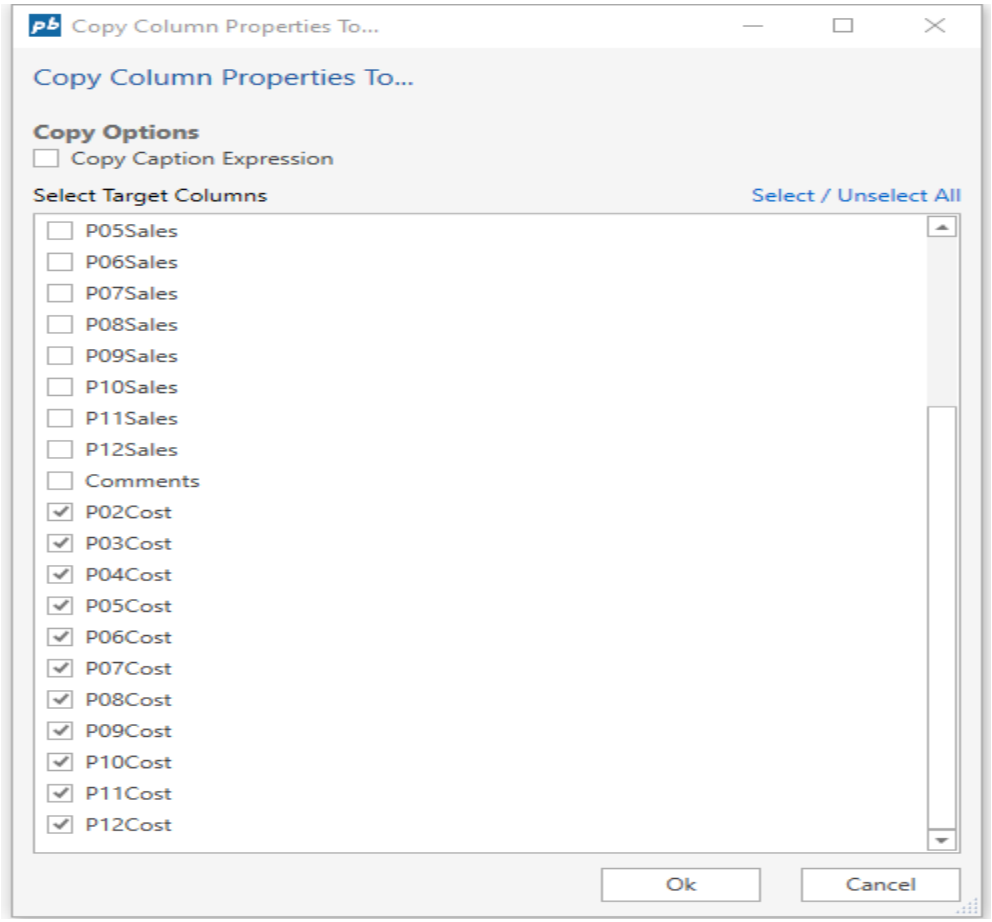
Click Ok to close Calculations dialog box and then save.

We can hide all the cost columns since the periodic cost is calculated.

1. Click on the “P01Cost” column to display the column properties
2. Scroll down until you find “Is Hidden” check box and check it
3. Right click on “P01Cost” and select “Copy Display Properties To..”



4. Check all the columns from P02Cost to P12Cost

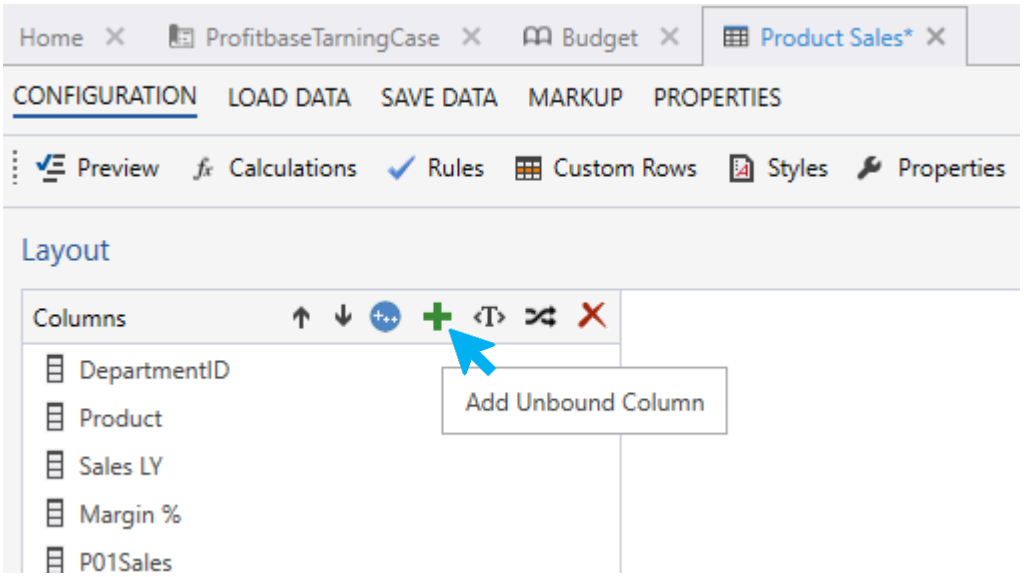


5. Click OK and save

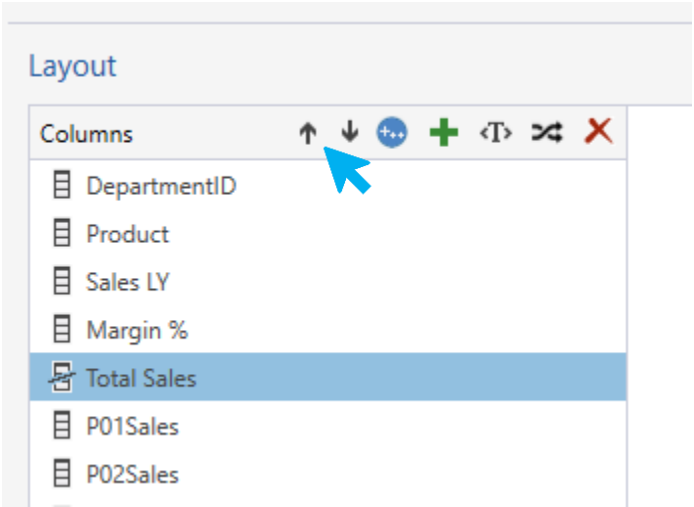
11.8 Creating Totals with Unbound Columns

Next, we are going to do is add a Column for Total Sales. This column is only for display in web and we don't need to save the calculation to our datastore.

1. Add a new column by clicking on green plus sign "Add Unbound Column"

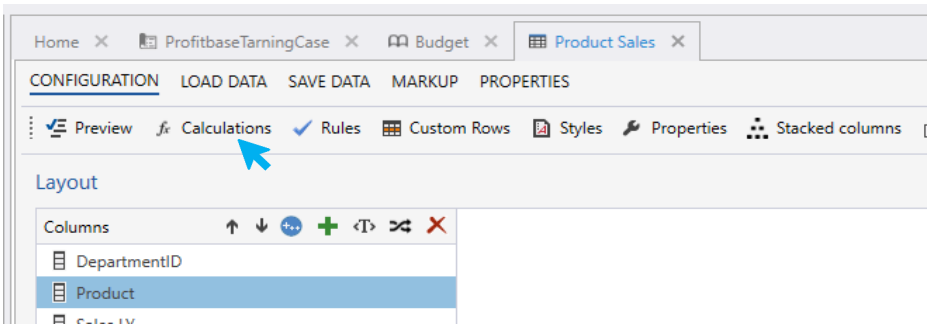


2. Name the column "TotalSales"
3. Enter "Total Sales" as Caption
4. Scroll down to the "Is Ready Only" check box and check it
5. Change the format string to: 0,0
6. Use the arrow to place the column after Margin %



7. Click Save or CTRL+S

Click on the “Calculations” button in the Worksheet.



Enter a calculation for the total sales column that sum up the periodic sales columns.

@TotalSales[] = @P01Sales[] + @P02Sales[] + @P03Sales[] + @P04Sales[] + @P05Sales[]+ @P06Sales[] + @P07Sales[] + @P08Sales[] + @P09Sales[] + @P10Sales[] + @P11Sales[] + @P12Sales[];

The result should look like this:

Click OK and then save.

A bonus task:

1. Create a Total cost column
2. Format the total columns as round number with thousand separators and make it read only

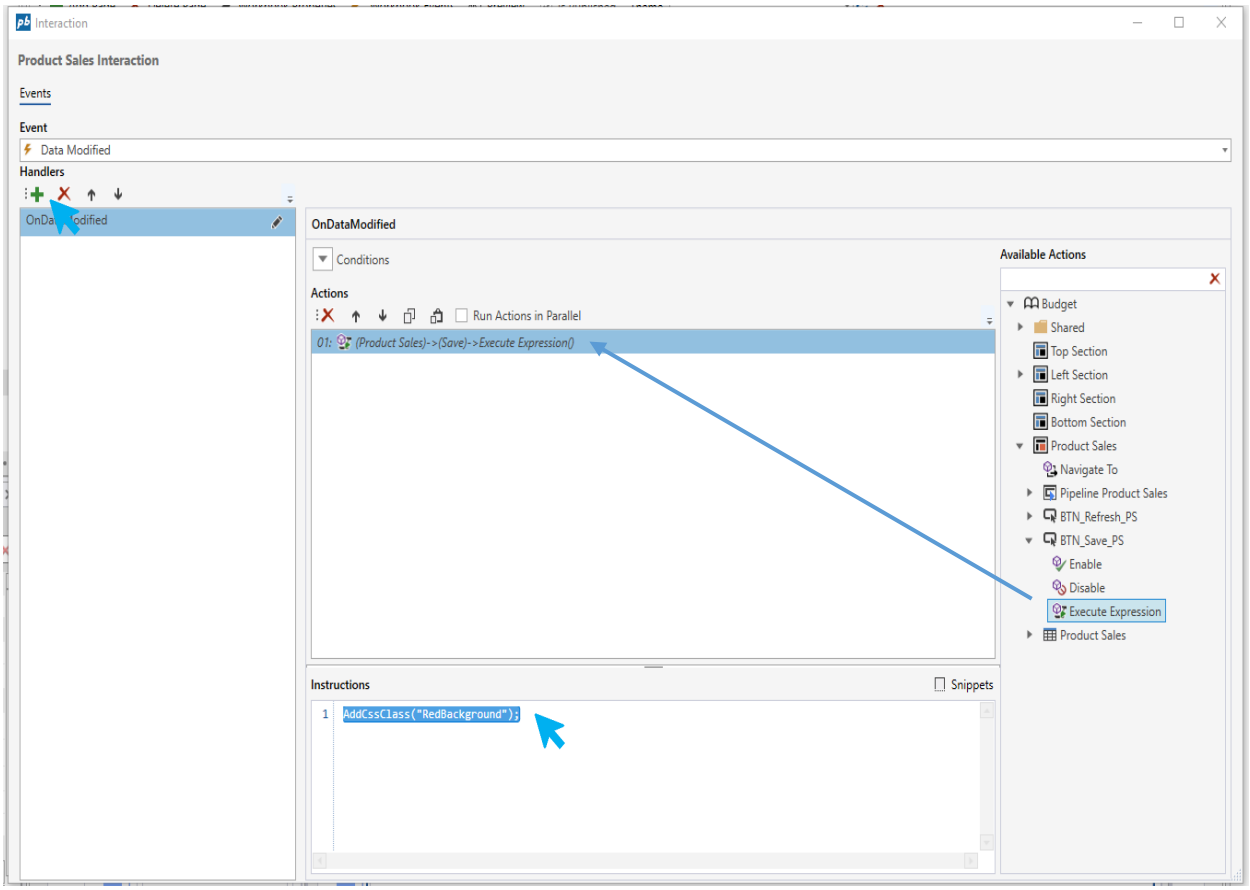
11.9 Formatting Button

We want the Save button to change color when we enter data without saving. Then change back to its original color when the data is saved.

1. Go to your “Budget” workbook
2. Select the “Product Sales” worksheet and click on the “Edit” next to Events in the properties field on the left-hand side
3. Use the “Add” button to add a new handler for Data Modified events
4. Drag and drop the Save button’s “Execute Expression” action on the Action field
5. Enter AddCssClass("RedBackground"); in the instruction field

   Note that the instructions field is Case Sensitive.

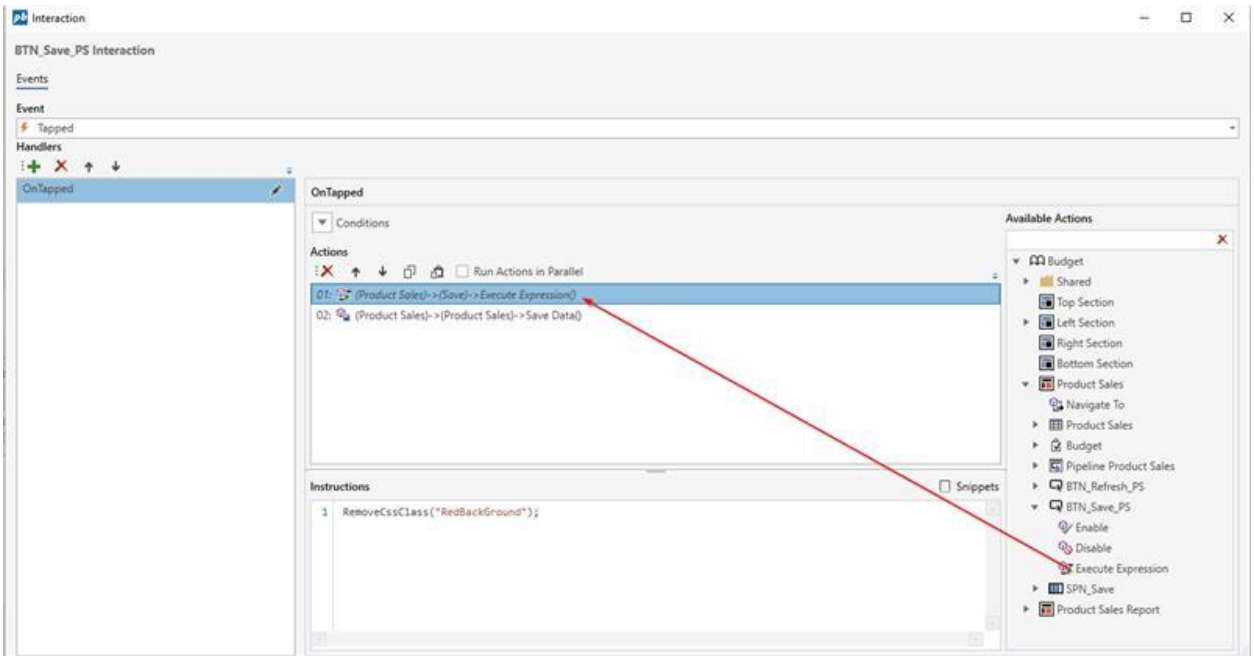
6. Close the pop-up and Save



7. Select the “Save” button and click on the “Edit” next to Events in properties field on the left-hand side
8. Use the “Add” button to add a new handler for Data Modified events
9. Drag and prod the Save buttons “Execute Expression” action on the Action field
10. Enter the following in the instruction field

```
RemoveCssClass("RedBackground");
```

11. Close the pop-up and Save



12 Pipeline and transaction data

Now, we have our data for sales and cost per period in 24 different columns. For our final result we want to store these transactions in a new table where we transpose each column into transaction rows. This is where the time frame reference on the columns come in to play. We are also going to add an account to our transactions.

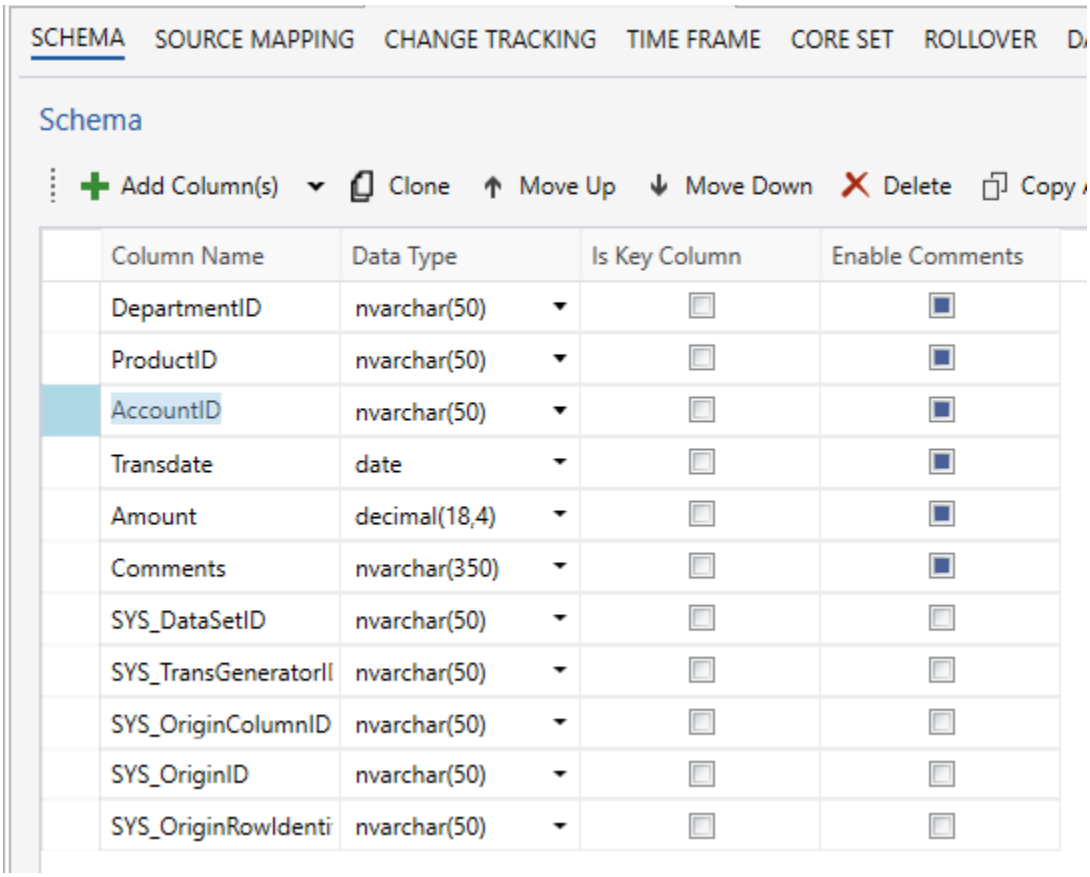
We will go through the following steps:

1. Create a transaction table
2. Create an account mapper table
3. Create a transaction pipeline

12.1 Creating a transaction table

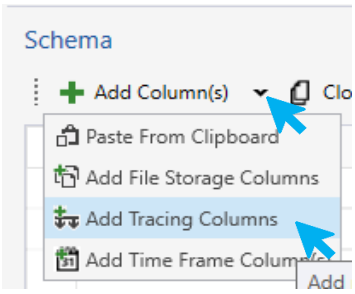
1. Right click on the Content folder in the Solution Explorer
2. Select “Add New Item…”
3. Choose “Data Store” and name it “Transdata Product Sales”, then click “Ok”
4. Click Save Or CTRL+S

The Schema for the transaction table should look like this:



Use the “Add Colum(s)” button to add the columns: DepartmentID, ProductID, AccountID, Transdate, Amount and Comments. Note: None of the columns should be key.

The columns having SYS_ as a prefix are tracing columns. You can easily add these by click on the small arrow next to the “Add Colum(s)” button. Select “Add Tracing Columns”.



Click Save or CTRL+S.

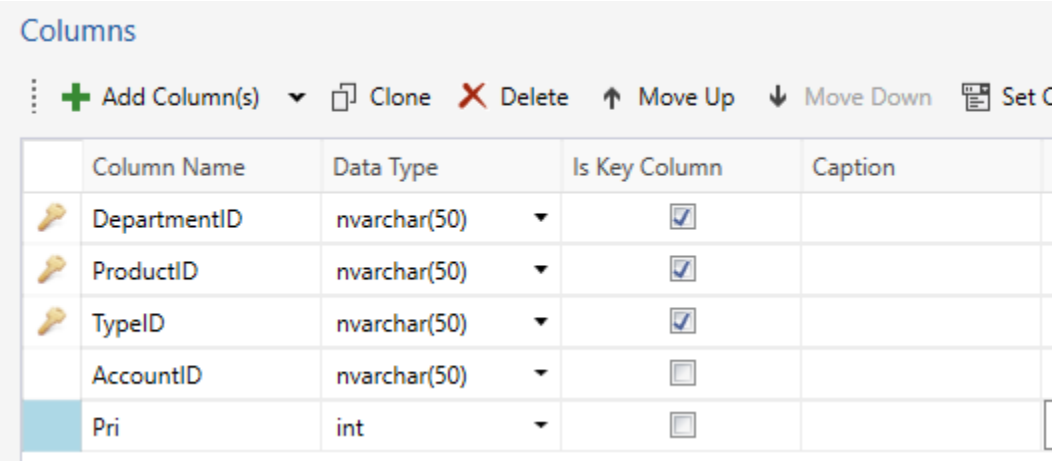
Your transactions table is finished.

12.2 Creating an account mapper table

We use this to create account mapping rules for the key columns of the input.

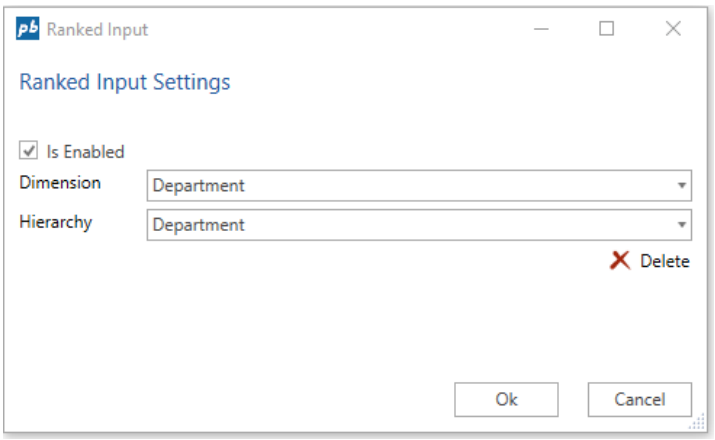
1. Right-click on the “Product Sales” datastore in the Solution Explorer
2. Select “Add New Item…”
3. Choose “Table” and name it “Product Sales Account Mapper”, then click “Ok”
4. Click Save Or CTRL+S

The Schema for our Product Sales Account Mapper table should look like this:

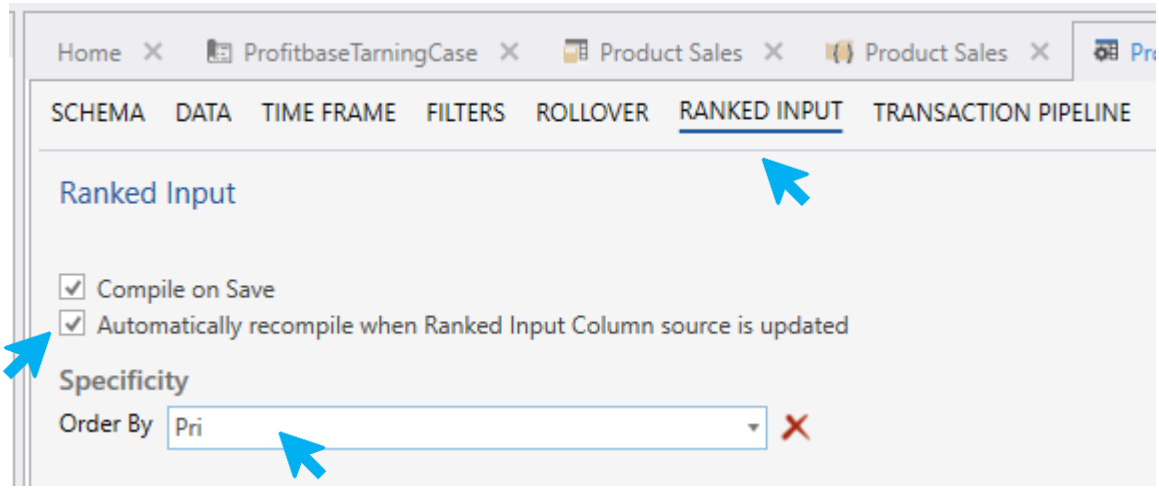


We need to configure the DepartmentID and ProductID columns as ranked input columns. This allows the pipeline to search through the mapping table to find the right connection.

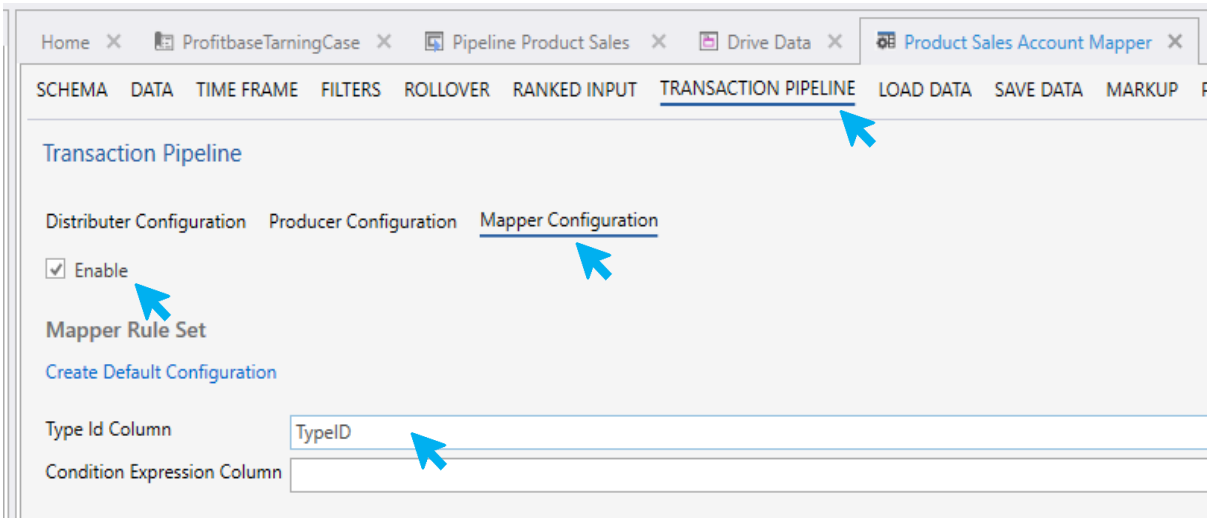
1. Click on the DepartmentID row in the schema to display the DepartmentID properties
2. Scroll down until you see “Ranked Input Settings” and click on the “…”
3. Check the “Is Enabled” box and select the Department dimension and hierarchy



4. Click “Ok” and save
5. Repeat step 1 to 4 for the ProductID column. Select the Product dimension and hierarchy
6. Go to the “Ranked Input” tab
7. Check the “Compile on Save” and “Automatically recompile when Ranked Input Column source is updated”, check boxes
8. Select “Pri” in the Order By dropdown.
9. Click Save or CTRL+S

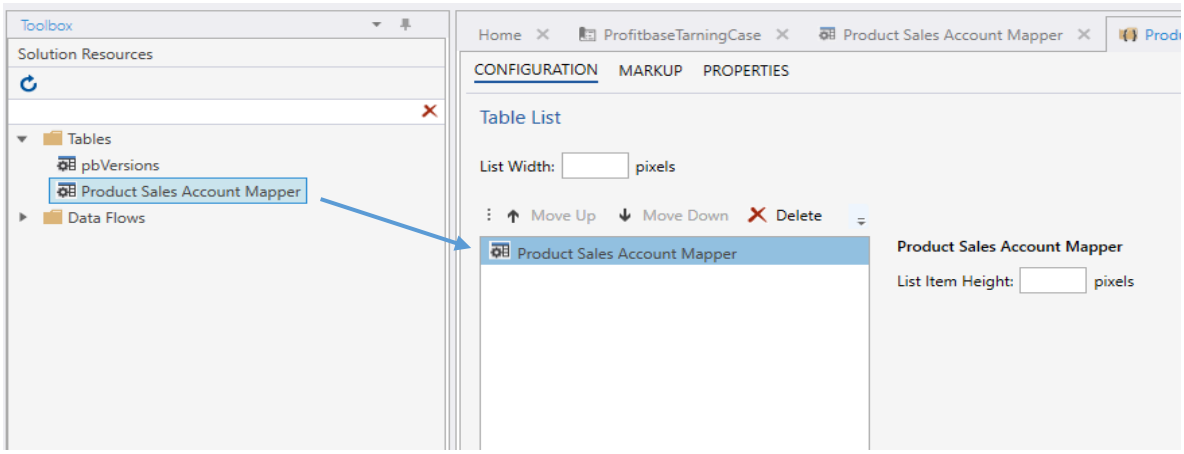


1. Navigated to the Transaction Pipeline tab
2. Choose Mapper Configuration, check the “Enable” box and set “Type Id Column” to TypeID
3. Click Save or CTRL+S



Now we need to fill our account mapper table with some rules. In order to do that, we need to publish our table in a workbook. We often like to publish tables, for settings such as this, in a list of tables. We create this table list as follows:

1. Right-click on the “Product Sales” datastore in the Solution Explorer
2. Select “Add New Item…”
3. Choose “Table List” and name it “Product Sales” then click “Ok”
4. Drag the “Product Sales Account Mapper” table into the table list



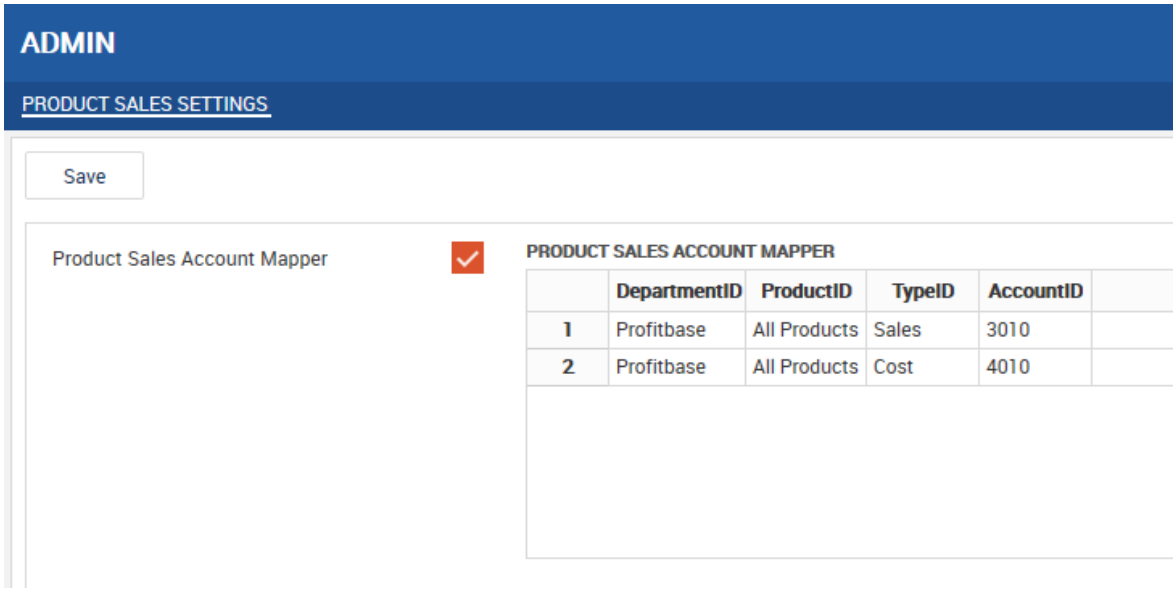
5. Click Save or CTRL+S

Now that we have our table in a table list, we can publish this list in a workbook. We will place these settings in a new workbook called admin to separate the input from the administrative tasks of in the solution. This also makes it easier to configure user access later on.

1. Right Click on the “Content” folder in the Solution Explorer
2. Select “Add New Item…”
3. Choose “Workbook” and name it, “Admin”, then click “Ok”
4. Click on “Add Page” button in the Main Layout
5. Give the page a name (“Product Sales Settings”) in the “Title” field under Properties to left.
6. Save
7. Add a save button (as we did in section 9.2)
8. Drag and drop your Table List “Product Sales” into the workpage
9. Click Save or CTRL+S
10. Set up the loading of the Table List on “Navigated To”. Se section 9.4.2 and **Error! Reference source not found.**
11. Set up the Save functionality on the “Save” button”. Se section 0
12. Check the “Is Published” check box
13. Click Save or CTRL+S

The workbook with the account mapper is now published. Let's take a look at it and see if we can add a few rules.

1. Go to the landing page in your browser and select the Admin workbook
   url hint: ServerName/SolutionName
2. Click on the Admin workbook
3. Click on the Product Sales Account Mapper table in the table list
4. Right Click in your "Product Sales Account Mapper" table to add new rows
5. Click on the "…" in the DepartmentID and ProductID columns and select the top level. These pop-ups are the result of the ranked input setting.
6. Fill in the remaining cells to match the image below
   Sales of all products for all departments will be mapped to account nr. 3010
   And costs to account 4010. You can add more rows for exceptions on either departments or products in rows below. The lowest rule matching the input is the one that will be used by the transaction pipeline.

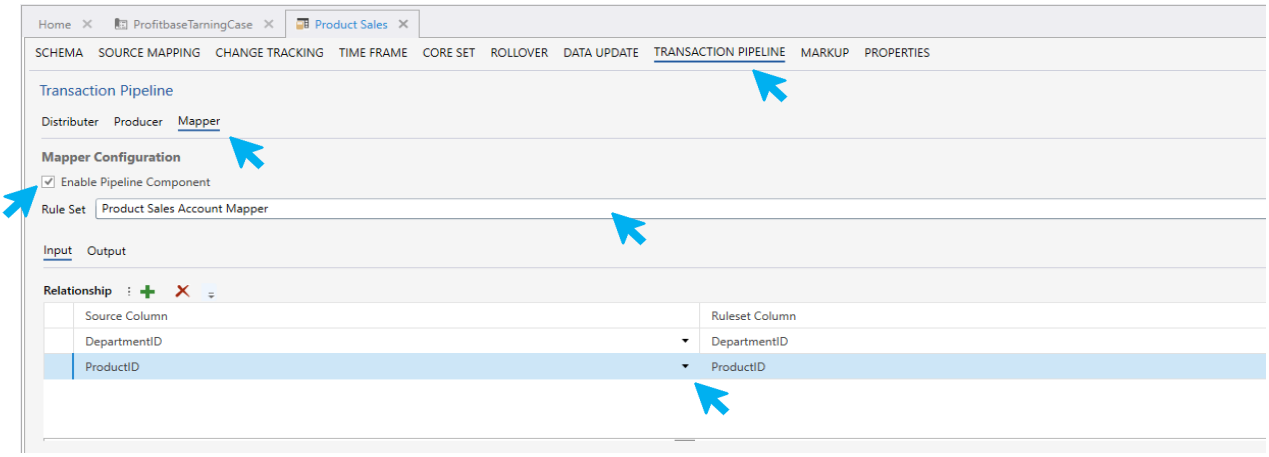


7. Click on your save Button.

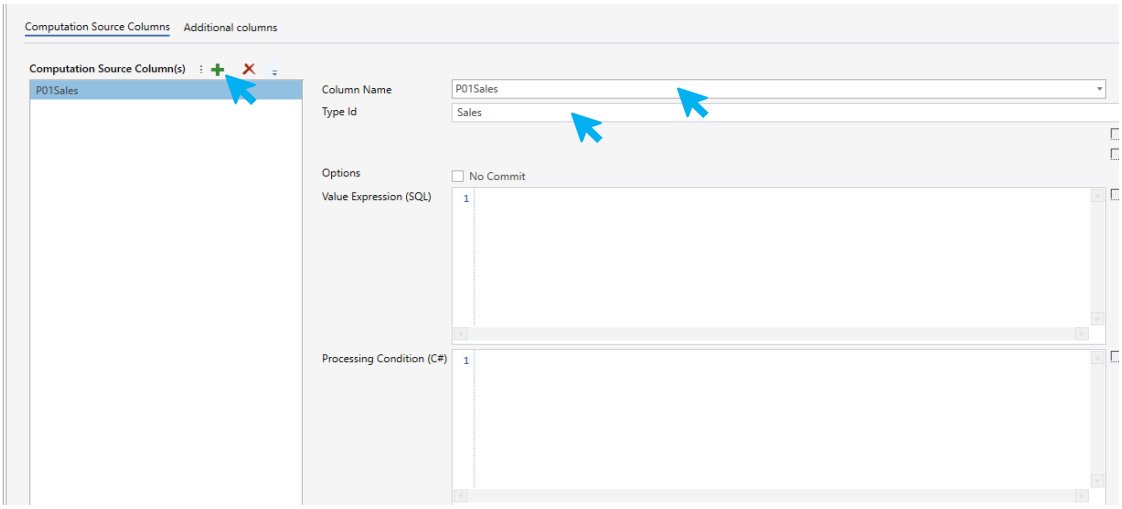
12.3 Connect the account mapper to the data store

We need to connect the account mapper to the “Product Sales” data store.

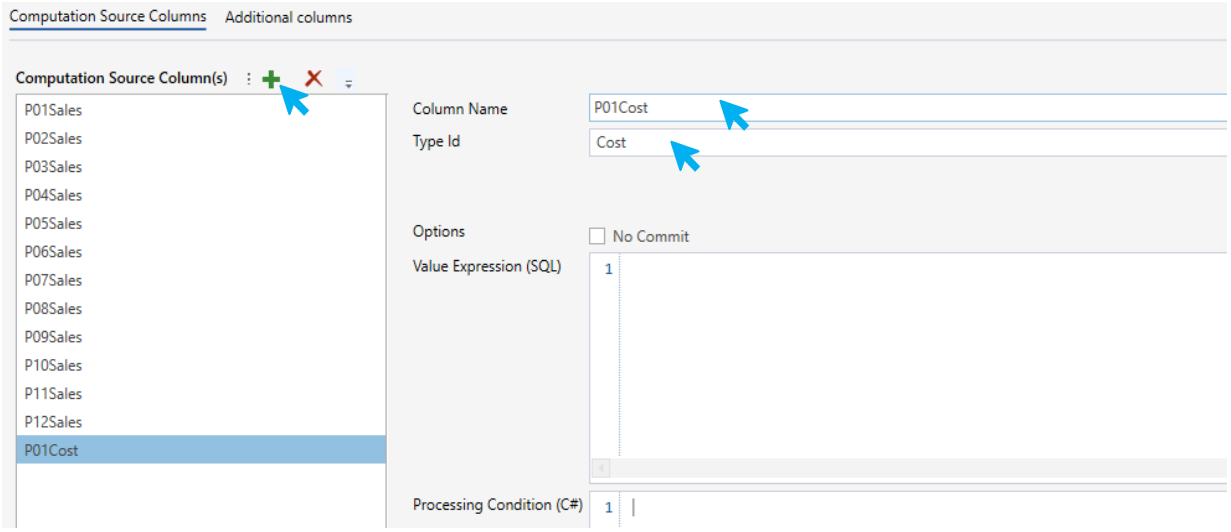
1. Start by going to “Product Sales” data store in the designer
2. Select “Transaction Pipeline” tab under the “Product Sales” data store
3. Select “Mapper”
4. Check “Enable Pipeline Component”
5. Set the Rule Set to “Product Sales Account Mapper”
6. In the Relationship box, map DepartmentID and ProductID from the data store to the respective columns in the ruleset (Product Sales Account Mapper)
7. Click Save or CTRL+S



1. Go to the Computation Source Columns and click on the add new columns button
2. Select P01Sales in the drop-down list “Column Name”
3. Type “Sales” in Type ID field
   This needs to match the input in the TypeID column in the mapper table for the pipeline to work

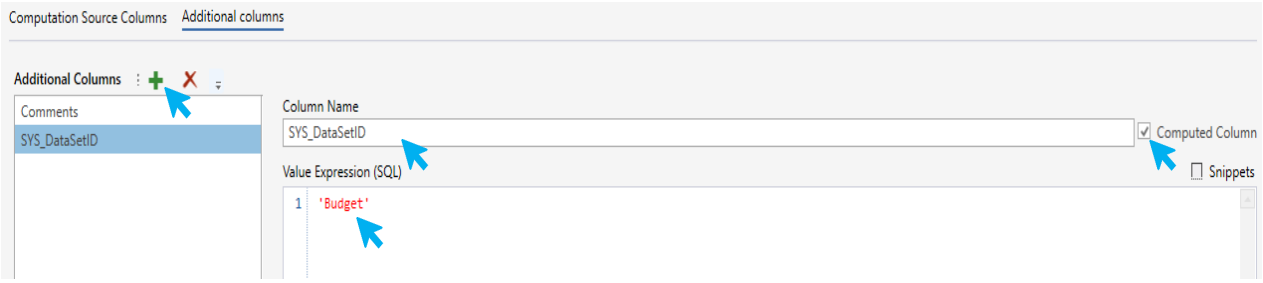


4. Repeat this for columns P02Sales to P12Sales
5. Repeat this for columns P01Costto P12Cost
   For these columns, type “Cost” in the Type ID field



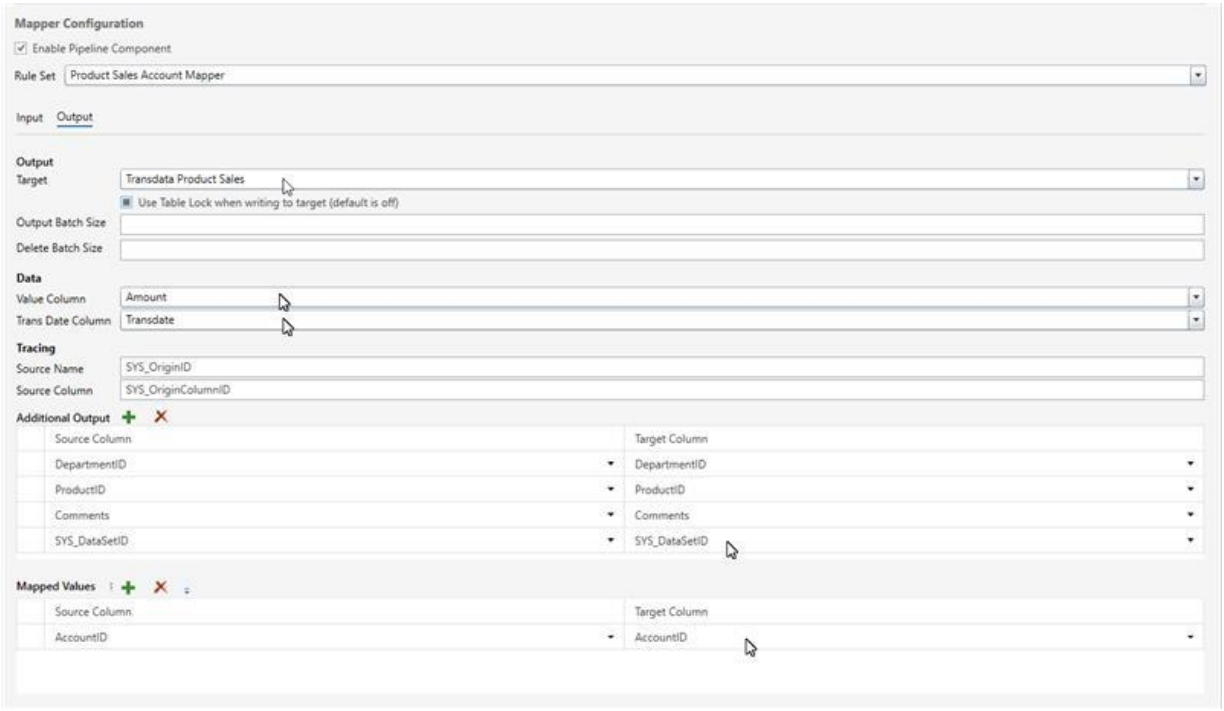
We have configured the mapping of the data source columns. Now we want to transfer some additional information to our transdata table.

1. Go to the “Additional columns” tab
2. Add new column and select “Comments” in the drop-down list “Column Name
3. Add new column and check the “Computed Column” check box
4. Now you could type in “SYS_DataSetID” in Column Name field
5. Enter ‘Budget’ in the “Value Expression (SQL)”
6. Click Save or CTRL+S



We have configured the input in our transaction pipeline account mapper. Let’s look at the output.

1. Go to the “Output” tab
2. Edit output settings to look like this and save

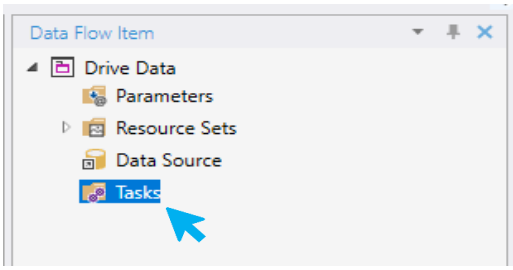


12.4 Create a transaction pipeline

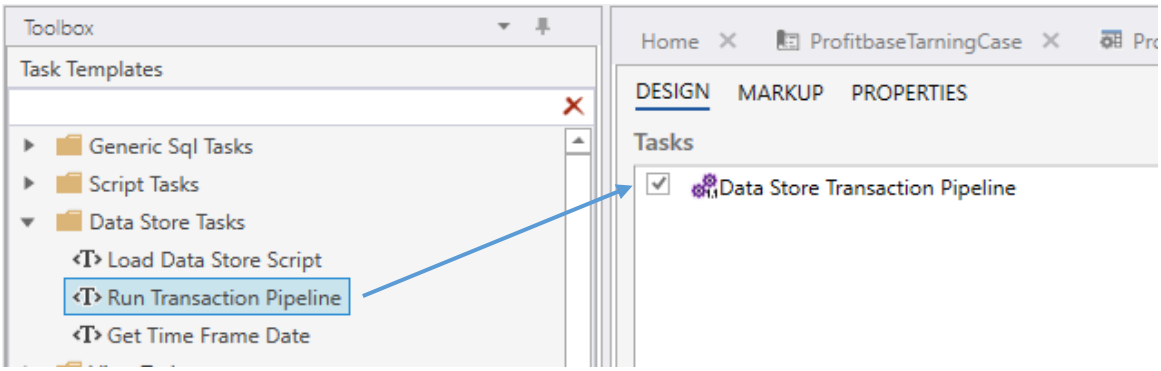
We are now going to create the job (DataFlow) that uses the transaction pipeline we just created, to make transactions I the “Transdata Product Sales” datastore based, on the input in the “Product Sales” datastore.

In the desktop designer:

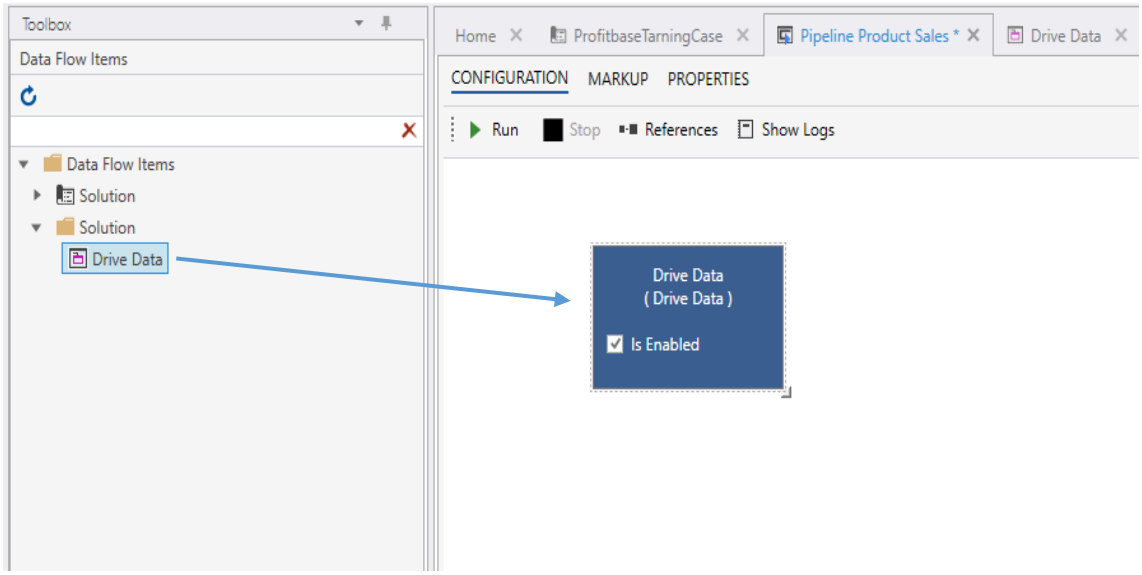
1. Right-click on the “Product Sales” data store in the Solution Explorer
2. Select “Add New Item…”
3. Choose “Data Flow” and name it “Pipeline Product Sales”, then click “Ok”
4. Right-click on the “Pipeline Product Sales” data Flow and select “New Data Flow Item” and name it “Derive Data”, then click “OK”
5. Click on the “Tasks” folder under the Data Flow Item



6. You will find the “Run Transaction Pipeline” task in the toolbox on the left side of the designer. Expand the “Data Store Tasks” folder and drag the “Run Transaction Pipeline” task into the task field and save



7. Go back to Pipeline Product Sales and drag the “Derive data” task into the dataflow.
   (Drive Data is a typo)

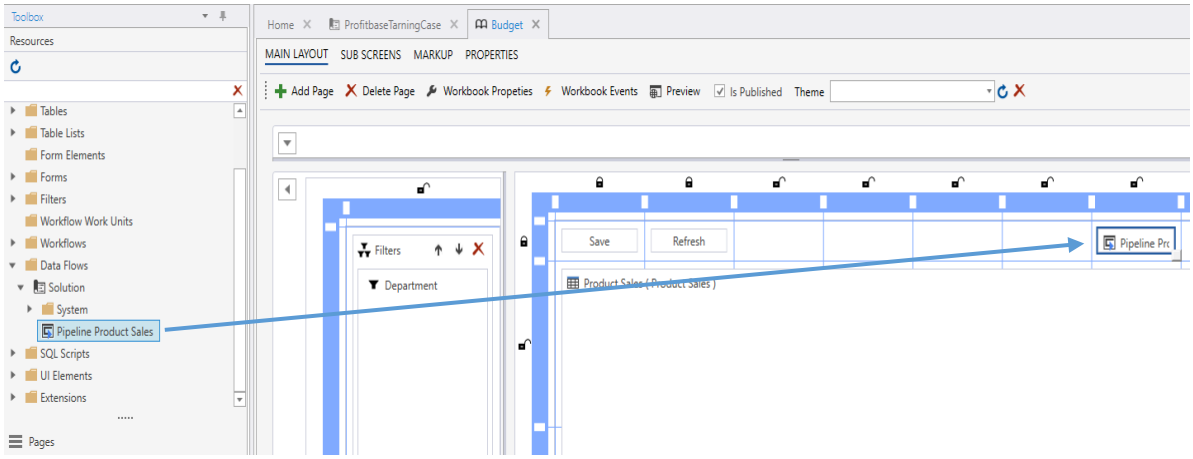


8. Click on the “Arguments” Edit… field and go to Store Id to enter the Product Sales data store id in the value filed.
   @Object[Product Sales, Store].Id
9. Click “OK” and Save

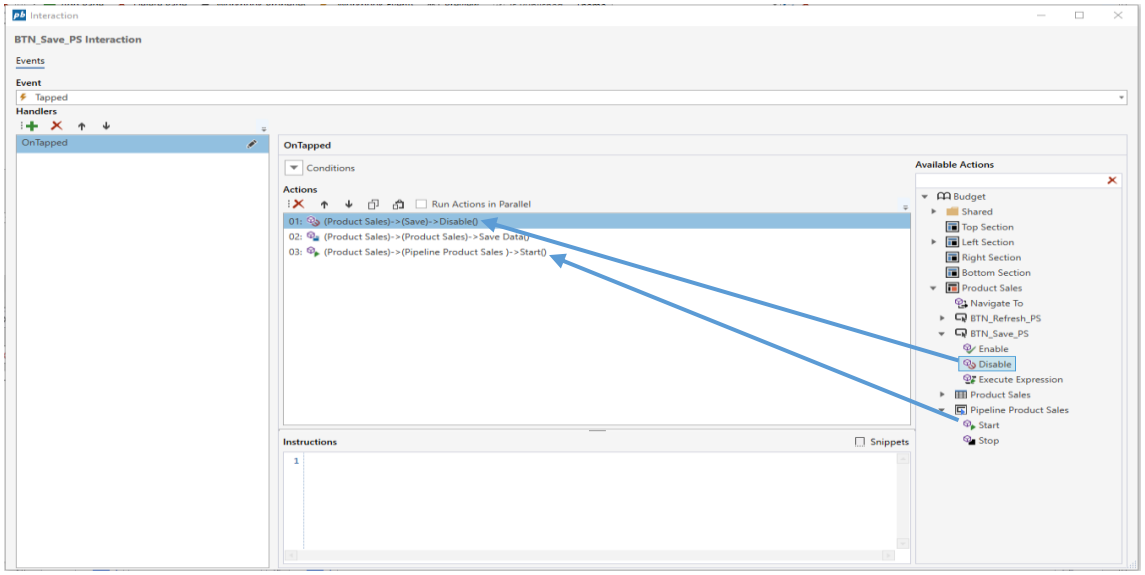
12.5 Running the Pipeline on save

We want to run our transaction pipeline when we save the worksheet. Additionally, we want to prohibit the users from saving again before the pipeline is finished.

1. Open the “Budget” workbook in the designer
2. Drag the “Pipeline Product Sales” into the “Product Sale” workpage

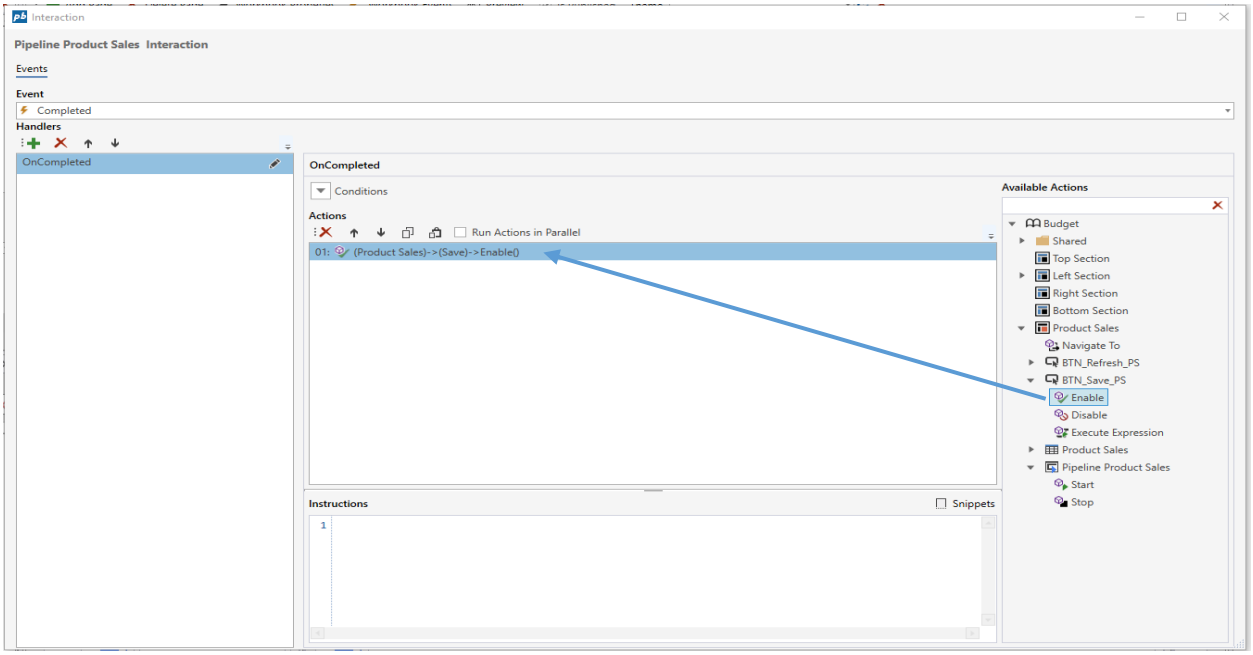


3. Click on the “Save” button in the workpage to display the button’s properties.
4. Click Edit… under Events
5. Expand the “Save” button in the available actions pane and drag the “Disable” button action into the Actions field.
   We do this to prevent users from overloading the servers.
6. Makes sure the “Disable” action is the first one. Use the arrow buttons to move it
7. Expand the “Pipeline Product Sales” and drag the “Start” Action into the actions field
8. Close the pop up and Save



When the pipeline is finished, we need to enable the save button again

1. Click on the “Pipeline” data flow to display the date flow properties.
2. Click Edit… under Events
3. Add a new Handler
4. Drag the “Enable” action for the save button into the Actions field

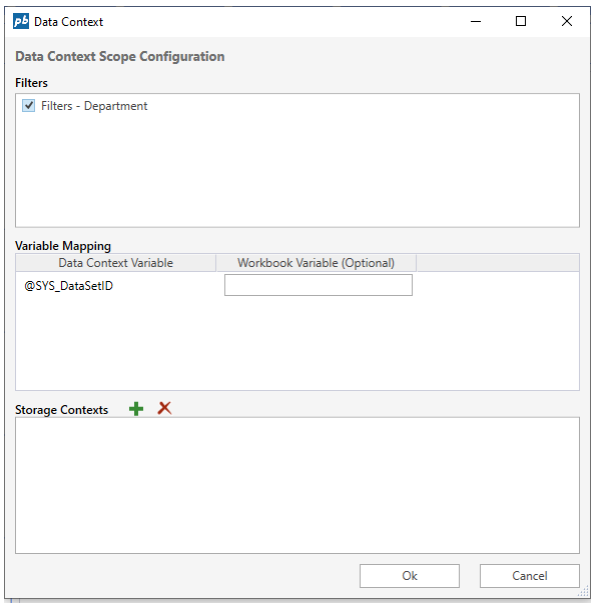


5. Close the pop up and Save

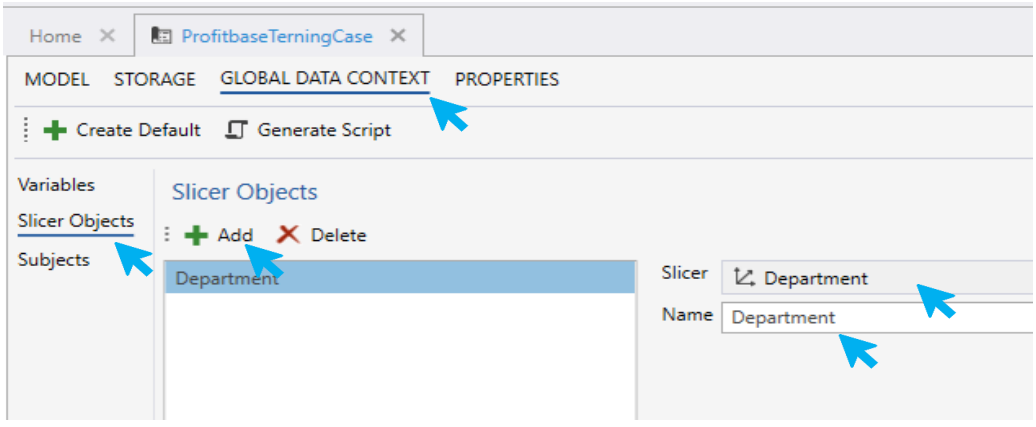
Challenge: See if you manage to add a spinner that starts when the user clicks on the save button and stopes when the workflow is completed.

When a user clicks on save in the web application, the Save button will be disabled until the data flow is finished. Currently, the data flow will run for all the departments in the hierarchy, not just the one you are editing. This can create some problems if many users work on different departments at the same time. To fix this, go through the following steps to set up what we call global context:

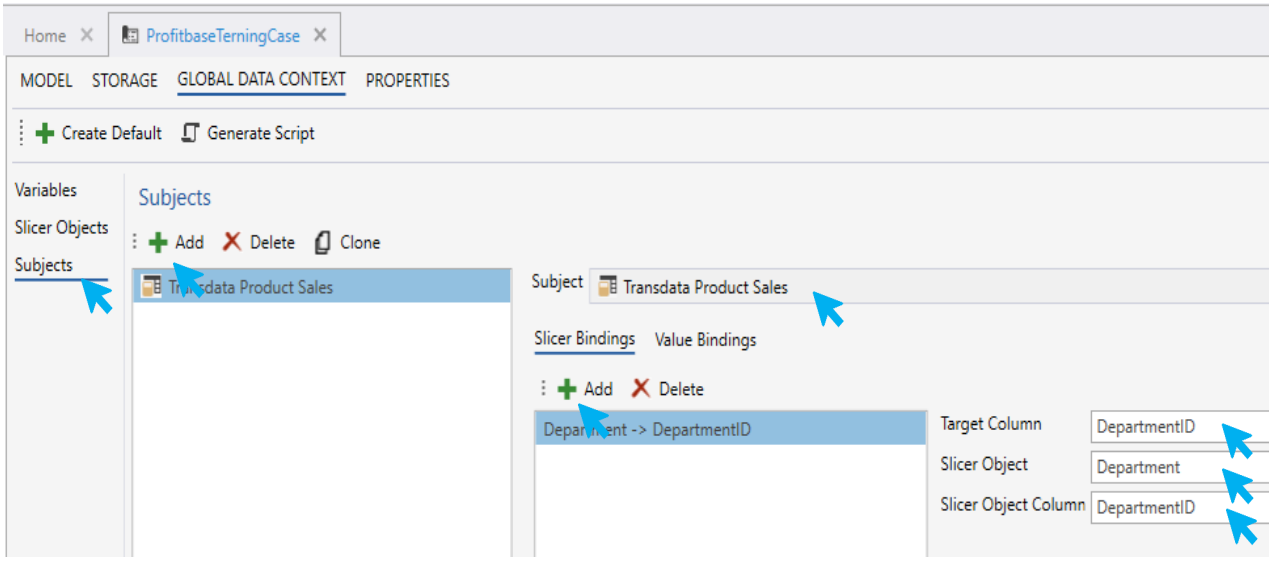
1. Select the data flow item in the workpage
2. Click on “Edit…” for Data Context Scope in the data flow propertied pane
3. Check the Filters – Department check box
4. Click OK and save



5. Go to the Solution Tab
6. Select Global Data Context
7. Select Slicer Objects and click “Add”
8. Find the department dimension
9. Name it, Department



10. Select Subjects and Add new
11. Click on the subject Edit and add “Transdata Product Sales” data store
12. Add new Slicer binding. Your solution should look like this:



13. Click Save or CTRL + S

13 Creating a SQL report

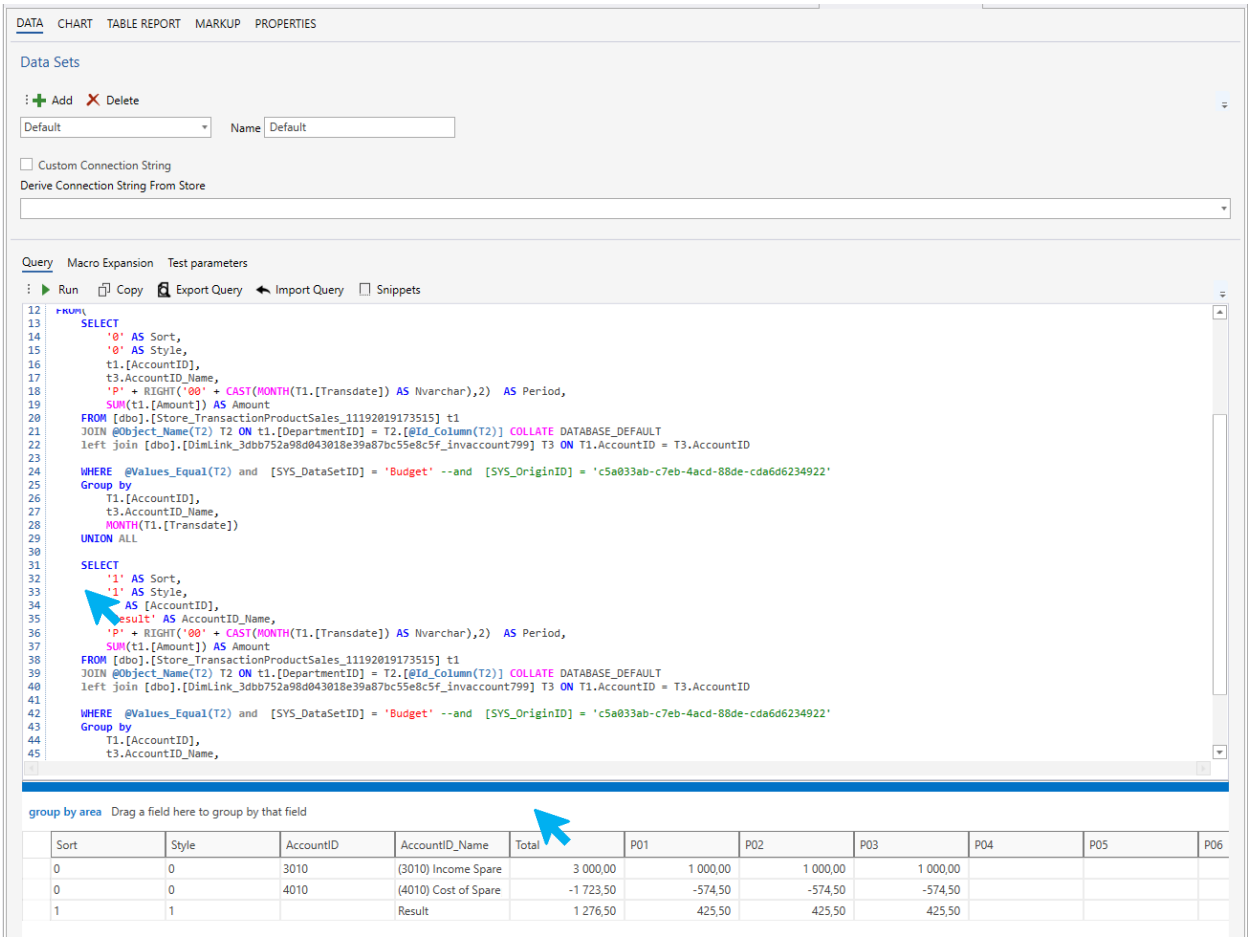
In the previous chapter we created financial transactions of our product sales input. This chapter will create a report based on these transactions.

1. Right click on the “Product Sales” data store and select “Add New Item”
2. Select an “SQL Report” and name it “Product Sales Report”
3. You can use this query as your report

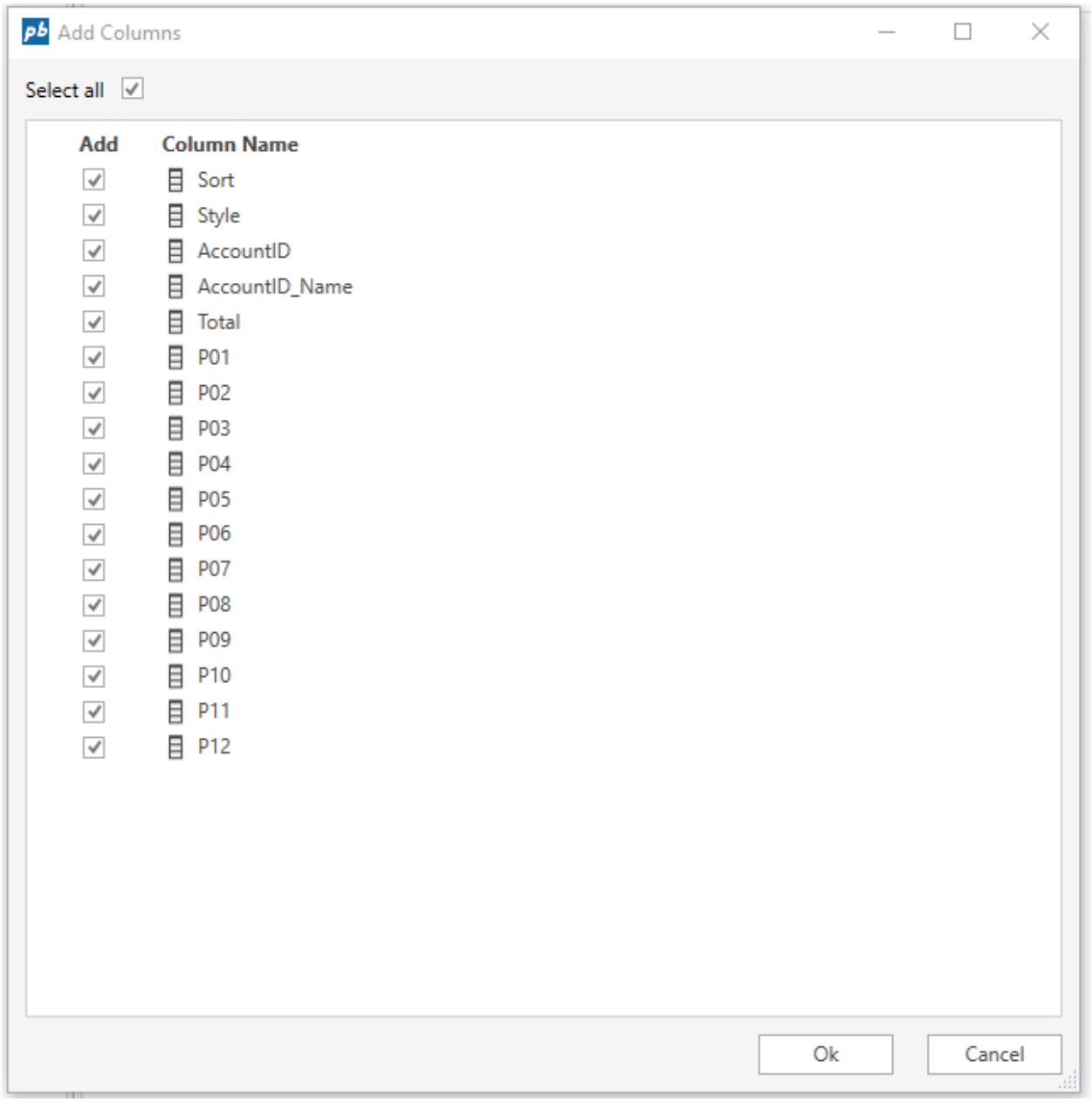
```
@Declare_FilterAlias(Department,T2)

Select
   [Sort],
   [Style],
   [AccountID],
   AccountID_Name,
    (ISNULL(P01,0) + ISNULL(P02,0) + ISNULL(P03,0) + ISNULL(P04,0) +  ISNULL(P05,0) + ISNULL(P06,0)
   + ISNULL(P07,0) + ISNULL(P08,0) + ISNULL(P09,0)+ ISNULL(P10,0) + ISNULL(P11,0) + ISNULL(P12,0)) AS
Total,
   P01, P02, P03, P04, P05, P06, P07, P08, P09, P10, P11, P12
FROM(
   SELECT
      '0' AS Sort,
      '0' AS Style,
      t1.[AccountID],
      t3.AccountID_Name,
      'P' + RIGHT('00' + CAST(MONTH(T1.[Transdate]) AS Nvarchar),2)  AS Period,
      SUM(t1.[Amount]) AS Amount
   FROM @Object[Transdata Product Sales, Store].DbObjectName     t1
   JOIN @Object_Name(T2) T2 ON t1.[DepartmentID] = T2.[@Id_Column(T2)] COLLATE
DATABASE_DEFAULT
   left join @Object[Account, Dim].DbObjectName T3 ON T1.AccountID = T3.AccountID

   WHERE  @Values_Equal(T2) and  [SYS_DataSetID] = 'Budget'
         --and  [SYS_OriginID] = 'c5a033ab-c7eb-4acd-88de-cda6d6234922'
   Group by
      T1.[AccountID],
      t3.AccountID_Name,
      MONTH(T1.[Transdate])


   UNION ALL

   SELECT
      '1' AS Sort,
```

```
    '1' AS Style,
    '' AS [AccountID],
    'Result' AS AccountID_Name,
    'P' + RIGHT('00' + CAST(MONTH(T1.[Transdate]) AS Nvarchar),2)  AS Period,
    SUM(t1.[Amount]) AS Amount
  FROM @Object[Transdata Product Sales, Store].DbObjectName T1   JOIN @Object_Name(T2) T2 ON
t1.[DepartmentID] = T2.[@Id_Column(T2)] COLLATE DATABASE_DEFAULT
  left join @Object[Account, Dim].DbObjectName T3 ON T1.AccountID = T3.AccountID

  WHERE  @Values_Equal(T2) and  [SYS_DataSetID] = 'Budget'
      --and  [SYS_OriginID] = 'c5a033ab-c7eb-4acd-88de-cda6d6234922'
  Group by
    T1.[AccountID],
    t3.AccountID_Name,
    MONTH(T1.[Transdate])
) t1
Pivot(
SUM(AMOUNT) FOR PERIOD IN (P01, P02, P03, P04, P05, P06, P07, P08, P09, P10, P11, P12)
)pvt
order by sort, AccountID
```

4. Past the data into the Query field and click run
5. Save



6. Go to the “Table Report” tab
7. Click the “Add” button and select “Columns From Query”
8. Check the “Select All” check box and click “Ok”
9. Click Save



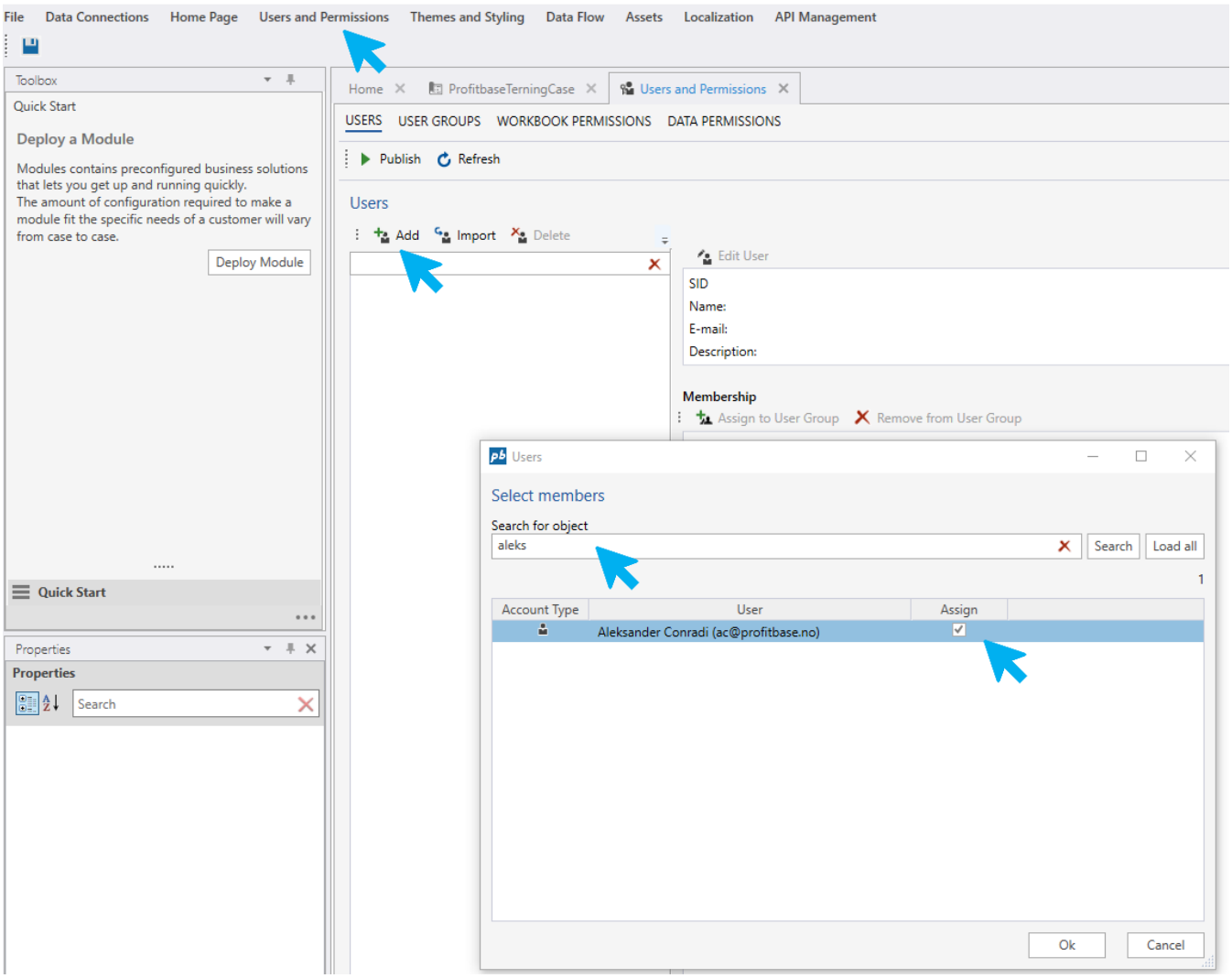
The new report needs to be placed in a workpage and configured with load actions. You should have the skills required to set this up yourselves. Review chapter 9 for tips.

14 User access

Our solution contains two workbooks, Admin and Budget. We are going to create two user groups, Admin and Standard. Admin will have access to both workbooks and the full department hierarchy. Standard will only have access to the Budget Workbook and Norway in the department hierarchy.

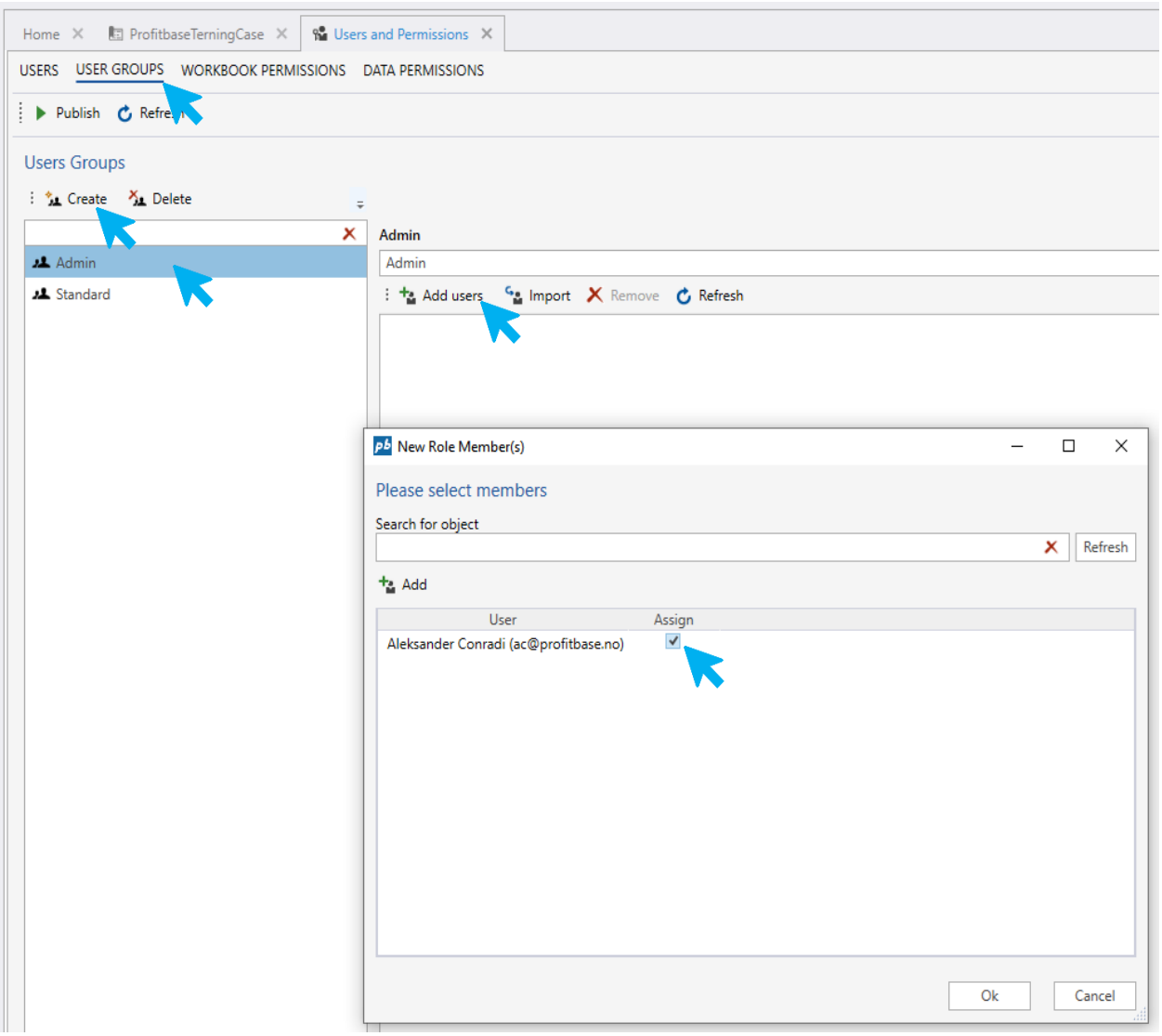
14.1 Add a user

1. Click on “Users and Permissions” tab at the top
2. Select “Manage Users and Permissions”
3. Click on the “Add” button to add a new user
4. Use the search field to find yourself
5. Click “Ok”
6. Click “Save”



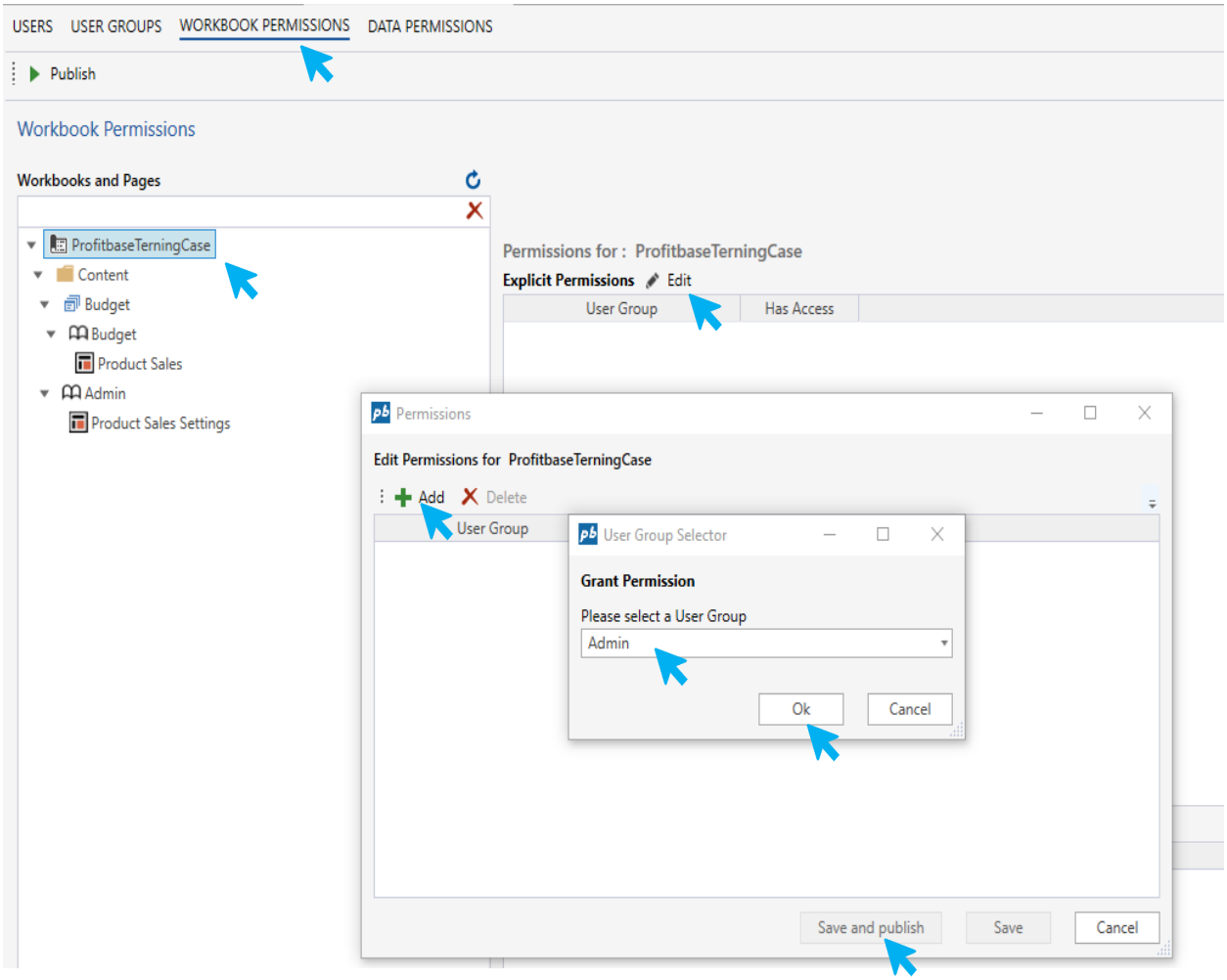
14.2 Create User Groups

1. Go to the “User Groups” tab to add new user groups
2. Click on the “Create” button
3. Name the User Group “Admin” and click “Ok”
4. Click on the “Create” button
5. Name the User Group “Standard” and click “Ok”
6. Highlight the “Admin” user group
7. Click on the “Add User” button
8. Assign your User to Admin group
9. Click “Ok” and Save

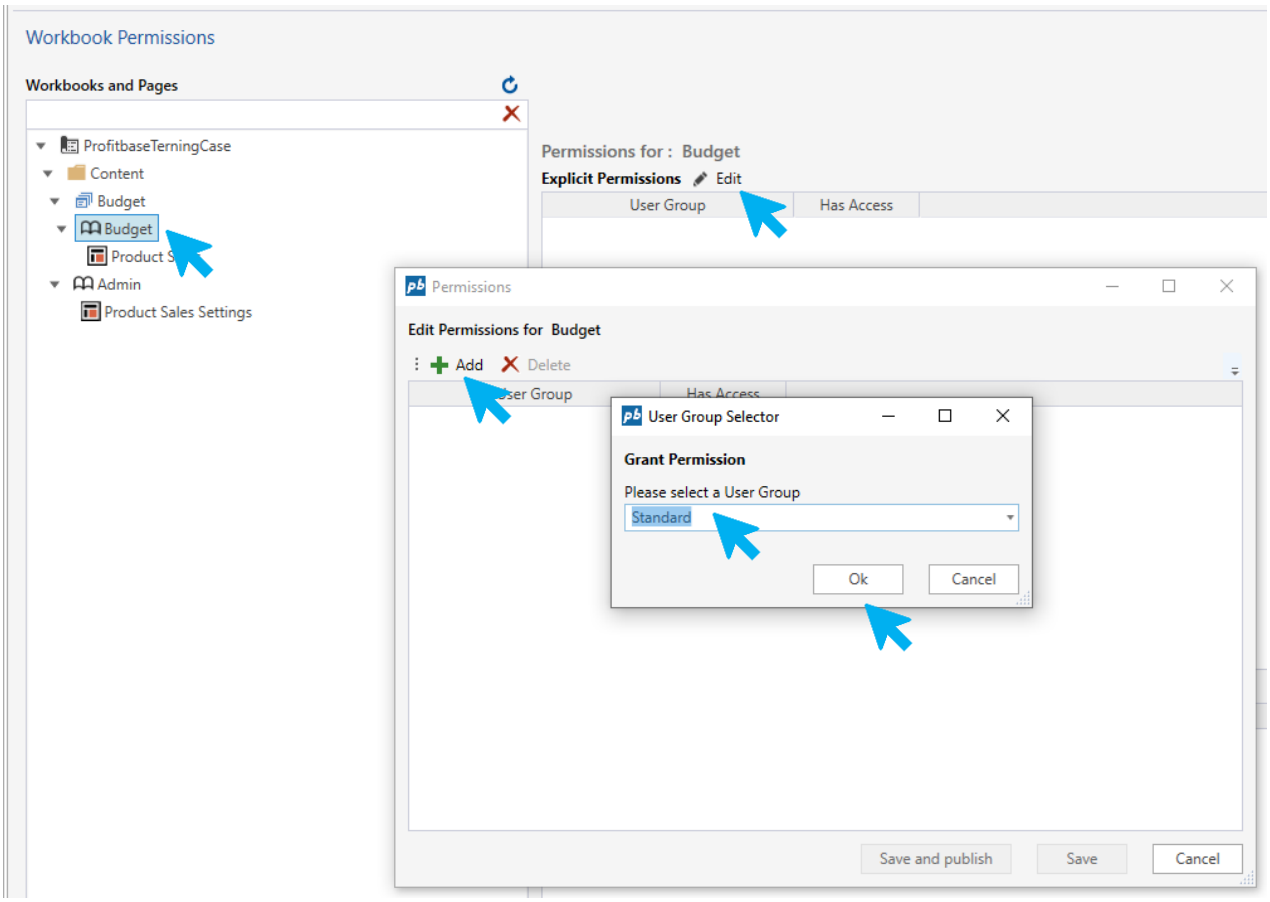


14.3 Workbook Permissions

1. Go to the “Workbook Permission” tab
2. Highlight your solution in the Workbooks and Pages field
3. Click on the “Edit” button on the right-hand side of **Explicit Permission**
4. Click the “Add” button in the Permissions pop-up
5. Select “Admin” in the user group selector and click OK
6. Click “Save and publish”

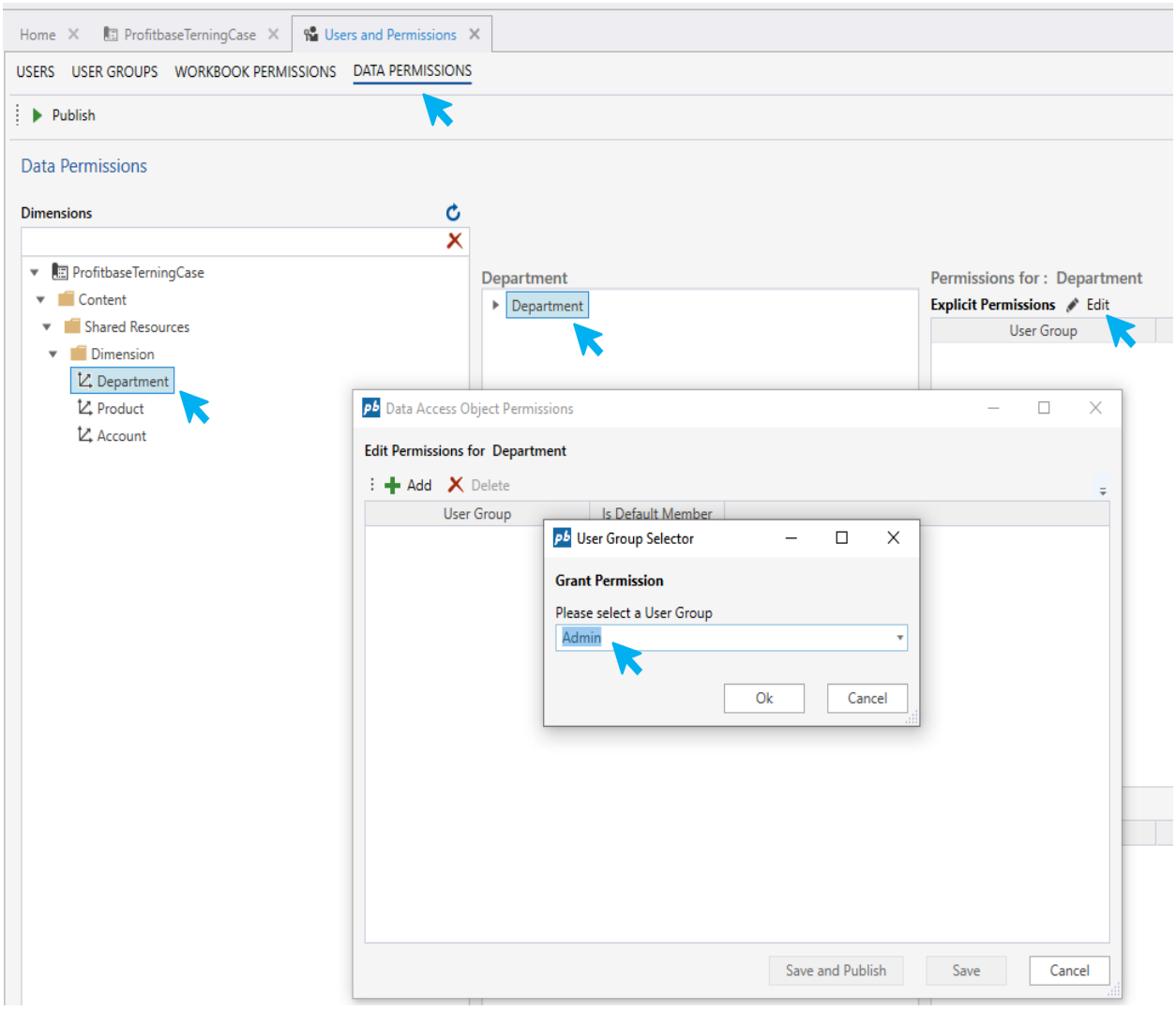


1. Highlight the “Budget” workbook in the Workbooks and Pages field
2. Click on the “Edit” button on the right-hand side of **Explicit Permission**
3. Click the “Add” button in the Permissions pop-up
4. Select “Standard” in the user group selector and click OK
5. Click “Save and publish”



14.4 Data Permission

1. Go to “Data Permission” tab
2. Highlight the Department dimension
3. Highlight the Department Hierarchy
4. Click on the “Edit” button on the right-hand side of **Explicit Permission**
5. Click the “Add” button in the Data Access Object Permissions pop-up
6. Select “Admin” in the user group selector and click OK
7. Make sure the “Is Default Member” is checked



1. Drill down in the Department Hierarchy until you find Norway
2. Highlight “Norway’’
3. Click on the “Edit” button on the right-hand side of **Explicit Permission**
4. Click the “Add” button in the Data Access Object Permissions pop-up
5. Select “Standard” in the user group selector and click OK
6. Make sure the “Is Default Member” is checked
7. Click “Save and publish”

You can now test different user accesses by moving your user from Admin to Standard.

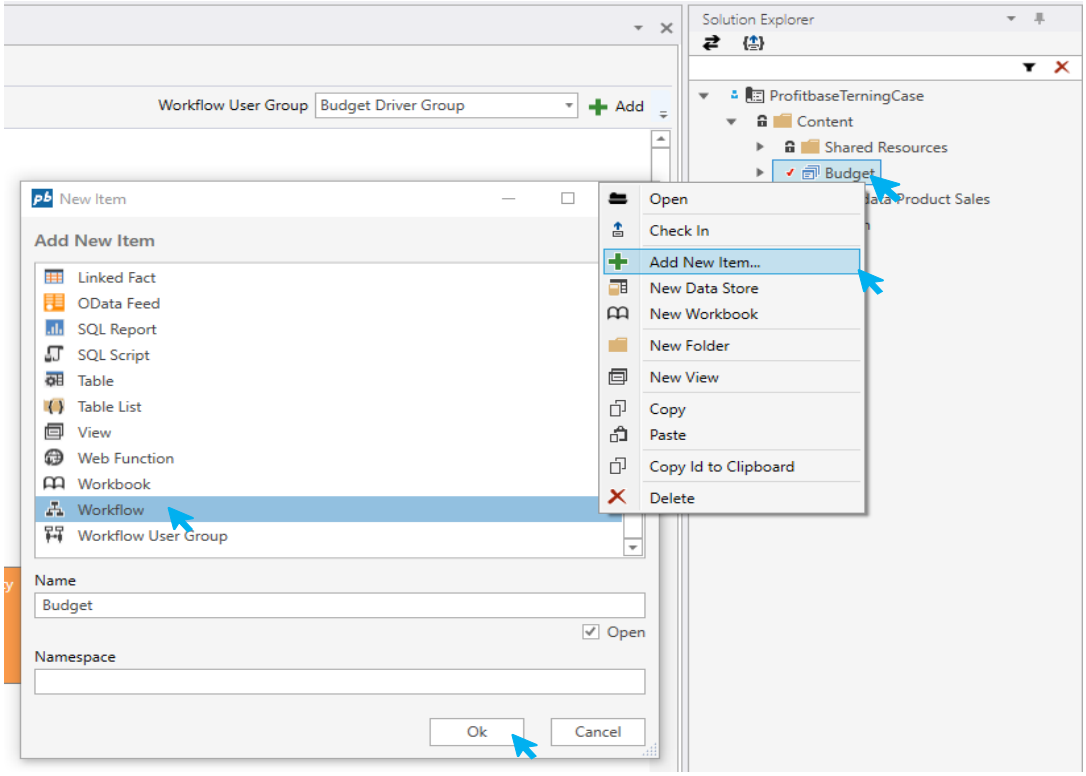
NB remember to add your user back in the Admin group before you close the designer. If you close the designer before you do this, you will not be able to get back in without deleting the entire user access configuration I the database.

15 Configuring workflow

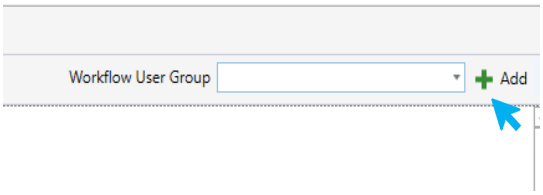
The workflow has several application areas. It allows us to generate several versions of our data as well as archiving a dataset. Further, it also allows us to have different input rights for different user groups. I.e. a Process administrator can edit after a Contributor is finished.

Let's create a workflow element with workflow user groups:

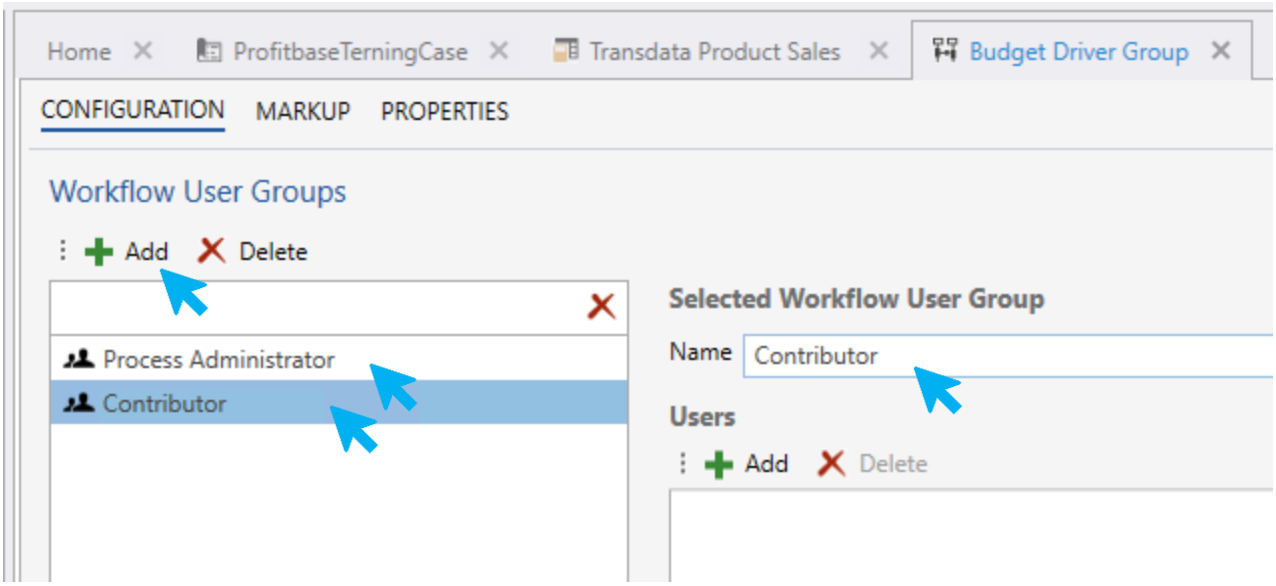
1. Right-click on the Data Store Group "Budget" and select "Add New Item…"
2. Choose the item "Workflow" and name the Workflow "Budget", then click "Ok"



3. Click the "Add" button next to "Workflow User Group" in the top right corner of the workflow window

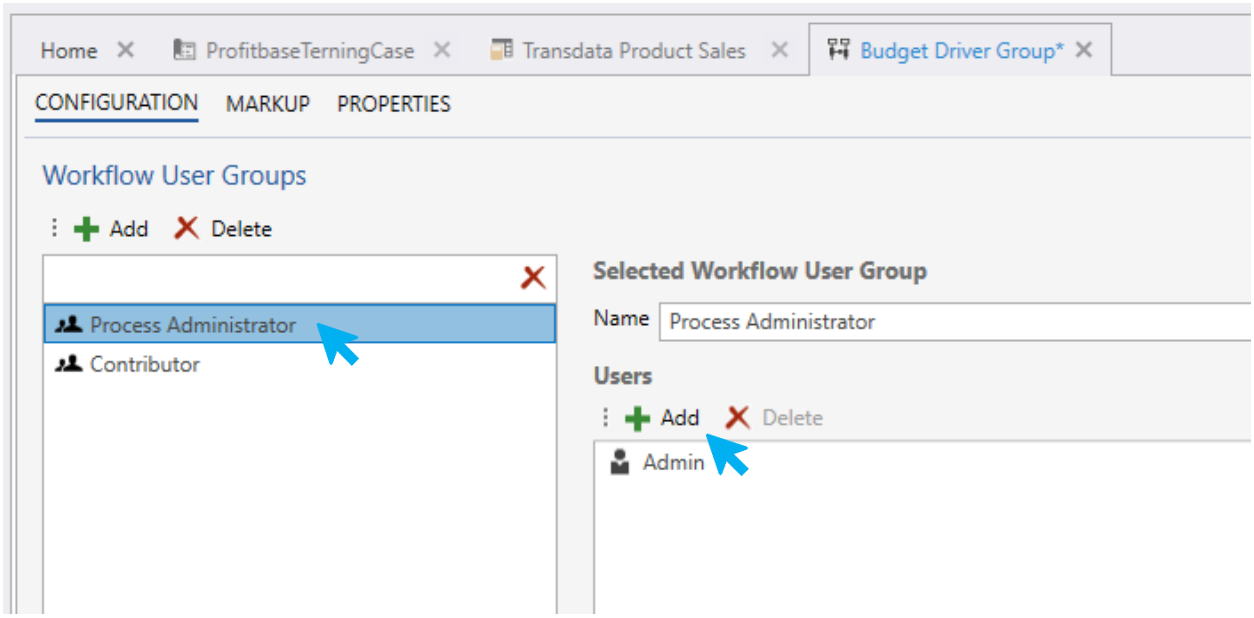


4. Click "Ok" in the New Item dialog box to use the default workflow user group name "Budget Driver Group"
5. Use the "add" button to add new workflow user groups
6. Name one "Process Administrator" and another "Contributor"
   (image on the next page)

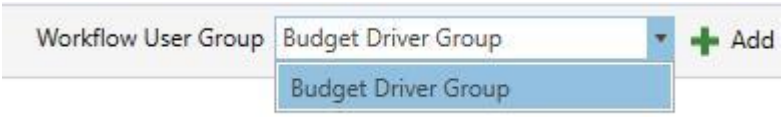


Connect the standard user roles to the respective workflow user groups to give them different levels of input rights

1. Select the “Process Administrator” workflow user group
2. Click the “Add” button in the “users“ area
3. Add the Admin group under users in the “Process administrators” group and the “Standard” user role in the “Contributor” group



4. Go back to the workflow component and assign your new user group to the workflow



Now it's time to configure the functionality of the workflow.

In this example the workflow will contain 3 steps which represent the status of a department in the respective workflow context. The workflow context indicates what departments, worksheets and or other elements that are affected by the respective workflow.

Step 1: Not started

This is the initial step for all new and existing departments in the solution. All workflow user groups will have editing rights at this stage.

Step 2: In progress/Reopened

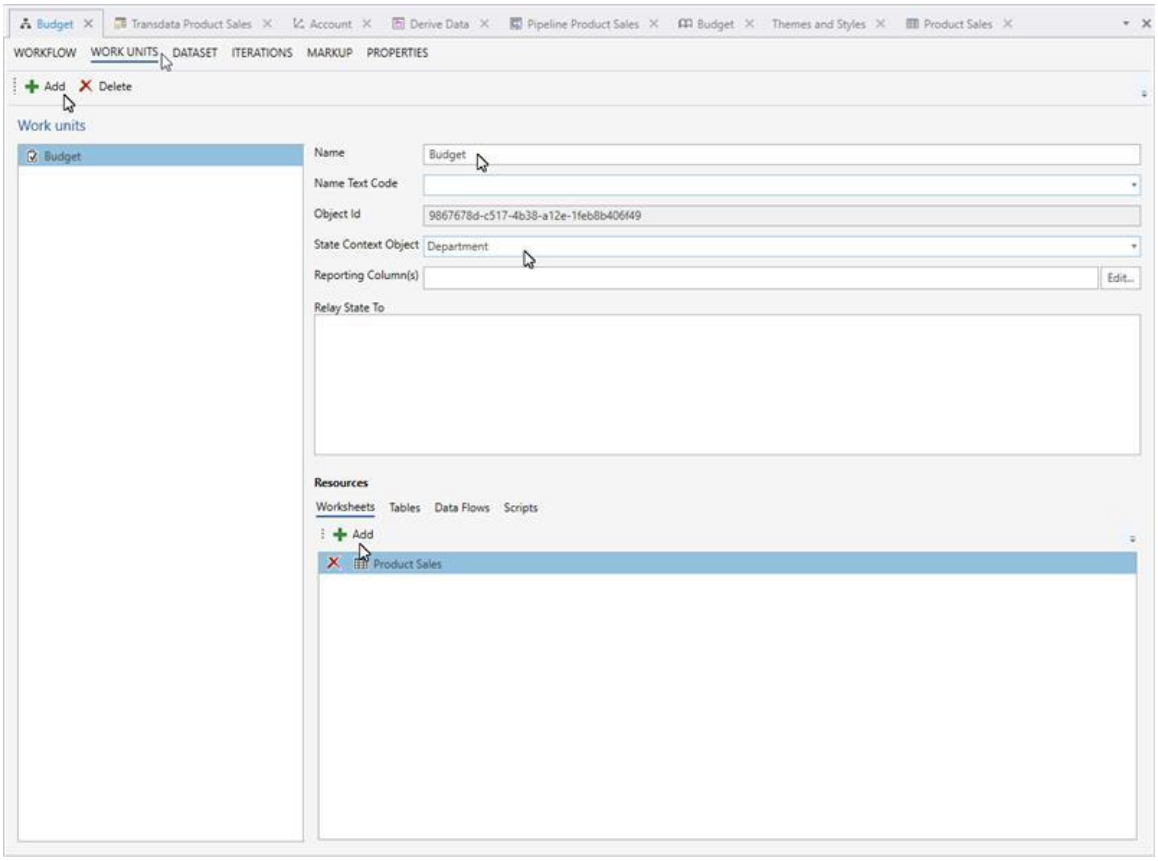
This step indicates that someone has started to work with the specified department in question. All workflow user groups maintain editing rights.

Step 3: Finished

At this stage, members of the Contributor user group (Standard user role) lose their editing rights. This is often used to make sure the planning process progresses forward. I.e. when a user is finished with his department and send the result to a manager for control. Users with rights to this step (Process administrators/Admin) can also reopen the worksheet for editing by users who only have editing rights at the "in progress level" and below.

Before we set up the different levels, we need to create the work units that specify what items in the solution that are affected (in this case the Product sales worksheet).

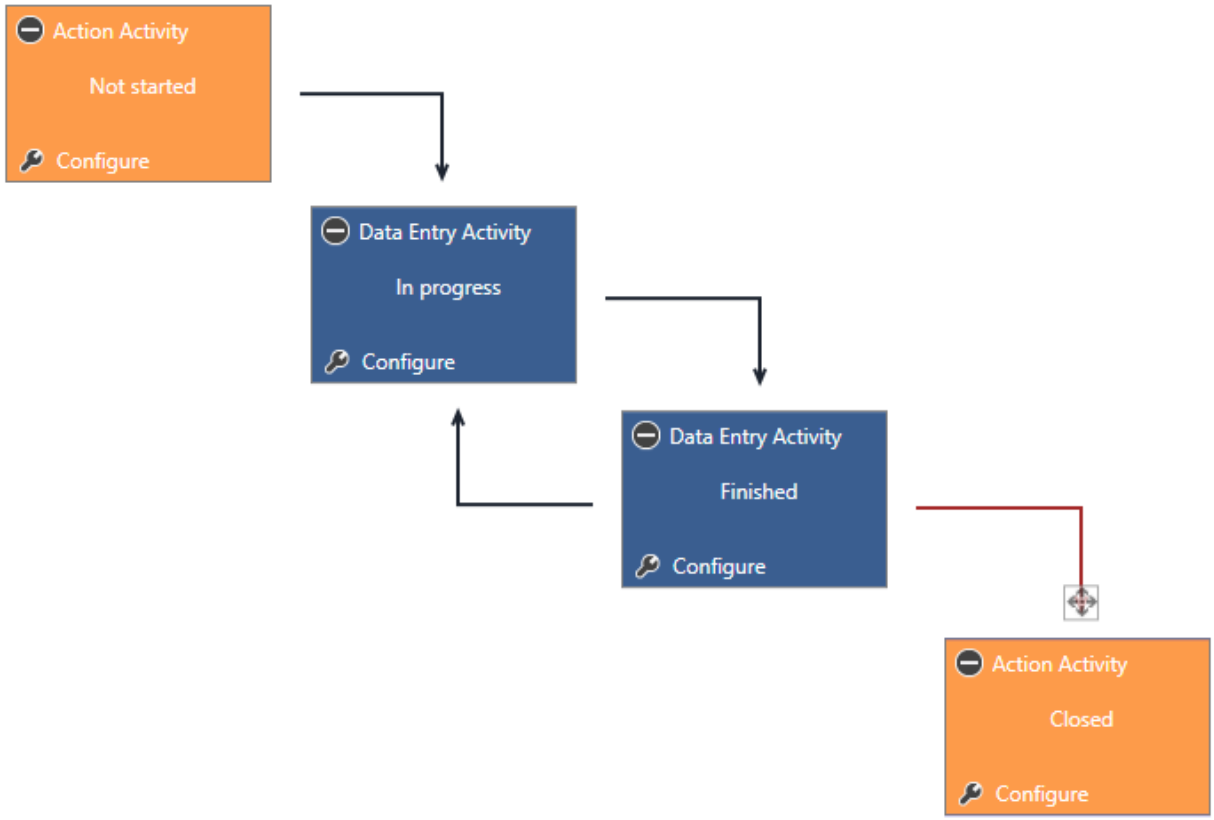
1. Click on the "Work units" tab
2. Add new work unit with the name, "Budget"
3. Select "Department" from the "State Context Object" dropdown list.
4. Add a new worksheet in the "Resources" field by using the "Add" button
5. Select the "Product Sales" worksheet in the pop-up and click "Ok"
   (image on the next page)



Now, let's configure the different stages:

1. Click on the "Workflow" tab
2. Drag a "Action Activity" from the toolbox into the blank space and name it "Not Started" in the properties pane
3. Drag two Data Entry Activities and name them "In progress" and "Finished" respectively
4. Close the workflow with a final Action activity called "Closed"
5. Connect the boxes with arrows by hovering next to a box, click, hold and release at the edge of the next box inside the small square.

Your workflow should look something like this (see next page)



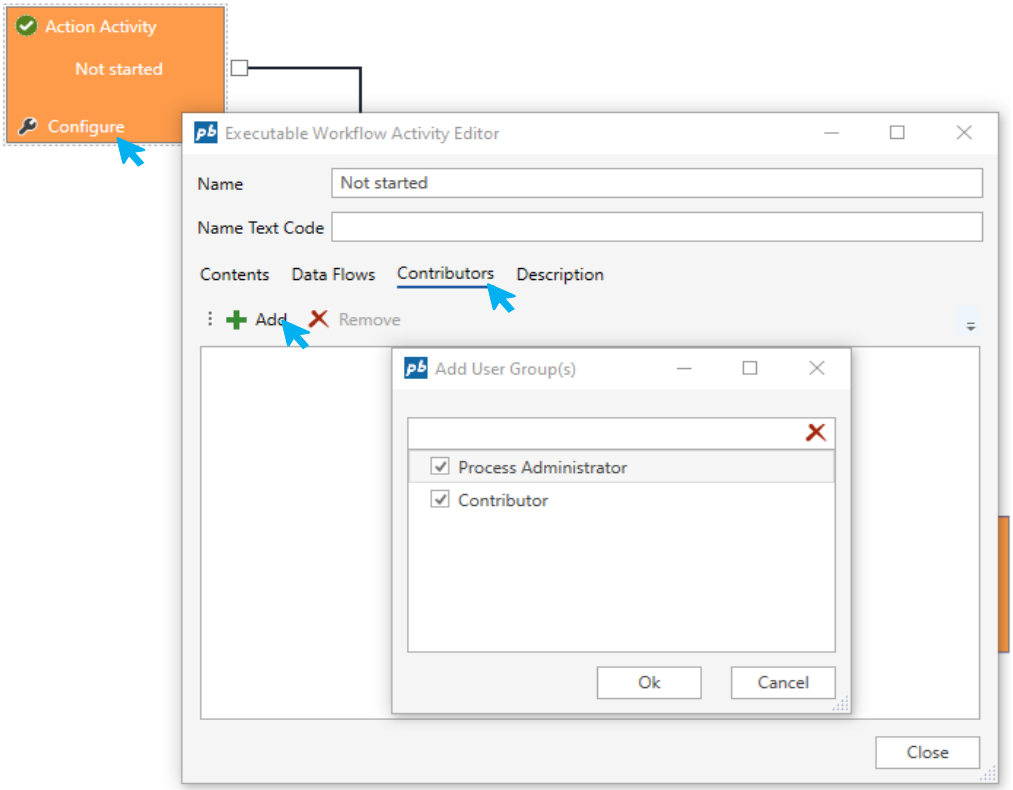
Each box represents a state where user groups can have or be denied access. This is configured in the “Contributors” tab in the pop-up you get from clicking on the respective “Configure” links.

The arrows represent the states. The only exception is the “In progress” state, which we will get back to.

15.1 Not Started Activity

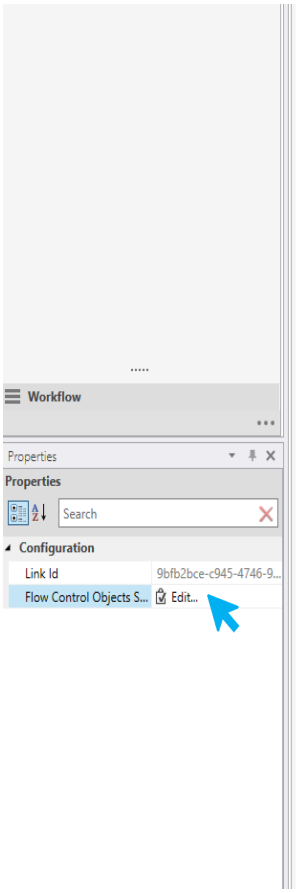
Give all workflow user groups access to this step

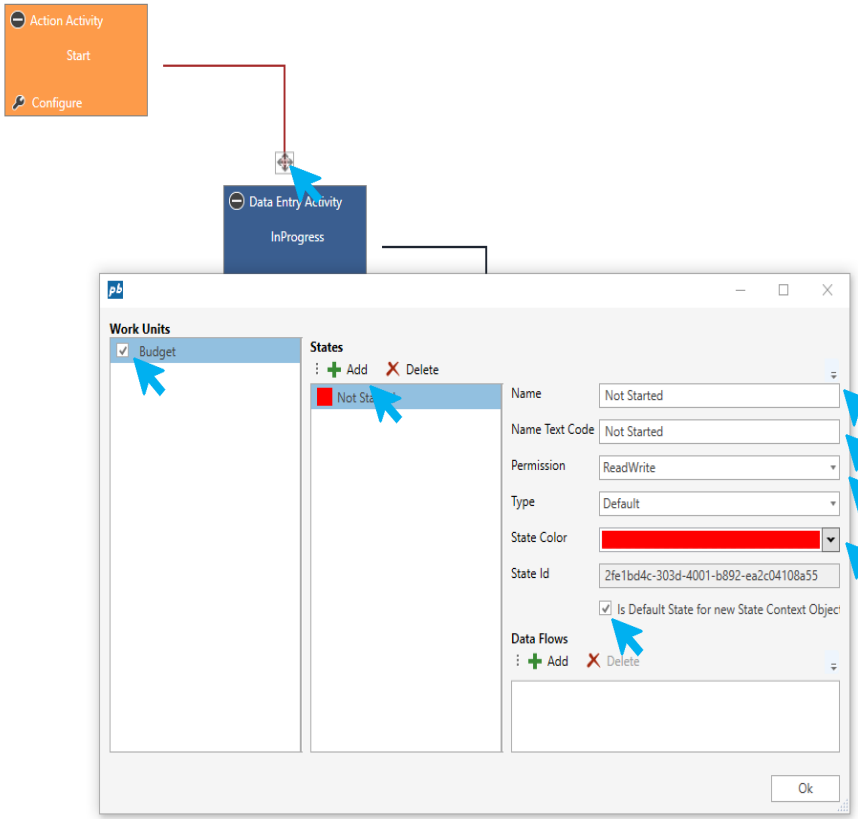
1. Click the configure action link on the “Not started” action activity
2. Go to the “Contributors” tab
3. Use the “Add” button to add Process Administrator and Contributor to contributors
4. Click “Close” and Save



Configure the “Not Started” state on the arrow between the “Not started” and the “In progress” boxes.

1. Click on the arrow between the Start and the InProgress boxes.
2. Click “Edit…” in the properties pane for the “Flow control object setup” property
3. Check the “Budget” check box under Work Units
4. Click the “Add button” under States
5. Fill out the new state as shown in the picture on the next page
    a. Name: Not started
    b. Permission: ReadWrite
    c. Type: Default
    d. State Color: Red
    e. Is default state…: yes

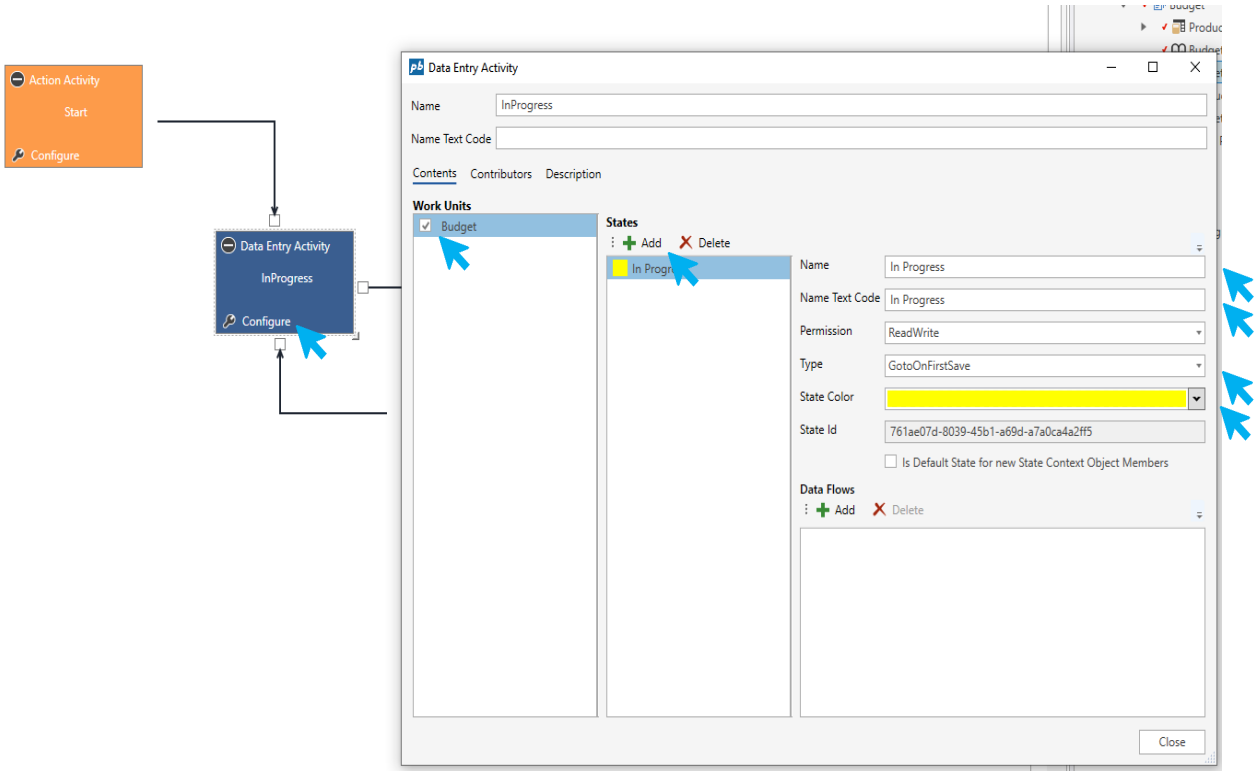

Workflow
Properties
Properties
Search
Configuration
Link Id
9bfb2bce-c945-4746-9...
Flow Control Objects S...
Edit...


Action Activity
Start
Configure
Data Entry Activity
InProgress
Work Units
Budget
States
Add
Delete
Not Sta...
Name
Not Started
Name Text Code
Not Started
Permission
ReadWrite
Type
Default
State Color
State Id
2fe1bd4c-303d-4001-b892-ea2c04108a55
Is Default State for new State Context Objec
Data Flows
Add
Delete
Ok

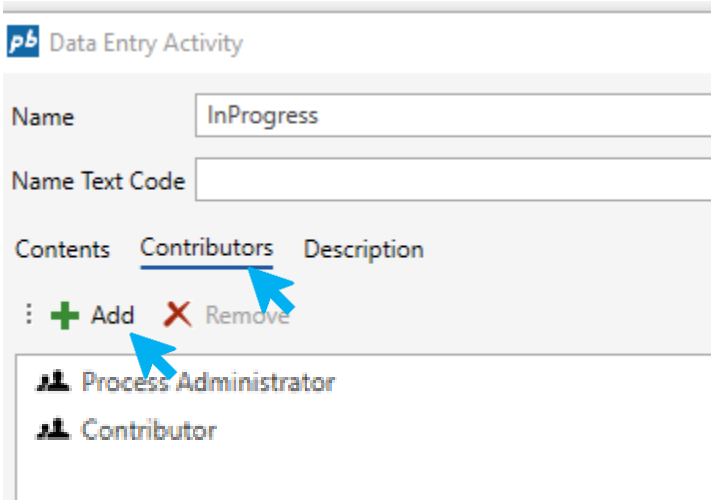
15.2 In Progress Activity

Do you recall this being the exception to the rule? This is the only state that is configured on the activity box and not on the arrows between them. This is because we intend to have the workflow automatically change to this state upon the first time the respective worksheet is saved.

1. Click on “Configure” on the “In Progress” Data Entry Activity
2. Check the “Budget” check box under Work Units
3. Click the “Add button” under States
4. Fill out the new state as shown in the picture below
    a. Name: In progress
    b. Permission: ReadWrite
    c. Type: GoToOnFirstSave
    d. Color: yellow

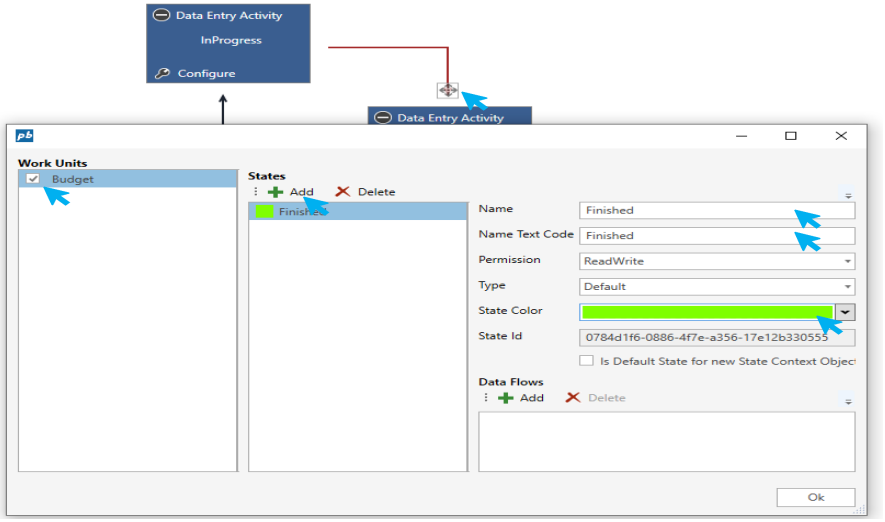


5. Go to the “Contributors” tab
6. Use the “Add” button to add Process Administrator and Contributor to contributors
7. Click “Close” and Save



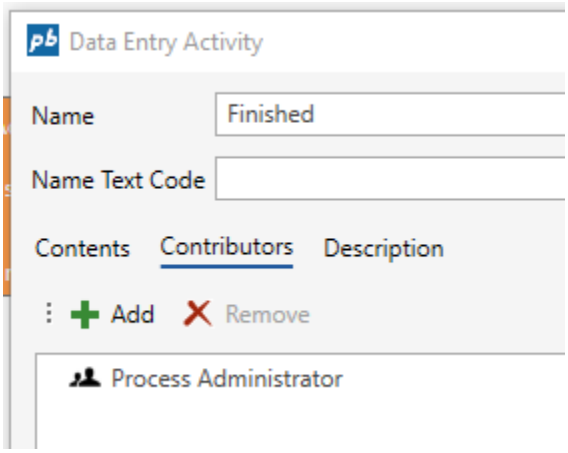
15.3 Finished Activity

1. Click on the arrow between the “In progress” and the “Finished” boxes.
2. Click “Edit…” in the properties pane for the “Flow control object setup” property
3. Check the “Budget” check box under Work Units
4. Click the “Add button” under States
5. Fill out the new state as shown in the picture on the next page
    a. Name: Finished
    b. Permission: ReadWrite
    c. Type: Default
    d. Color: Green



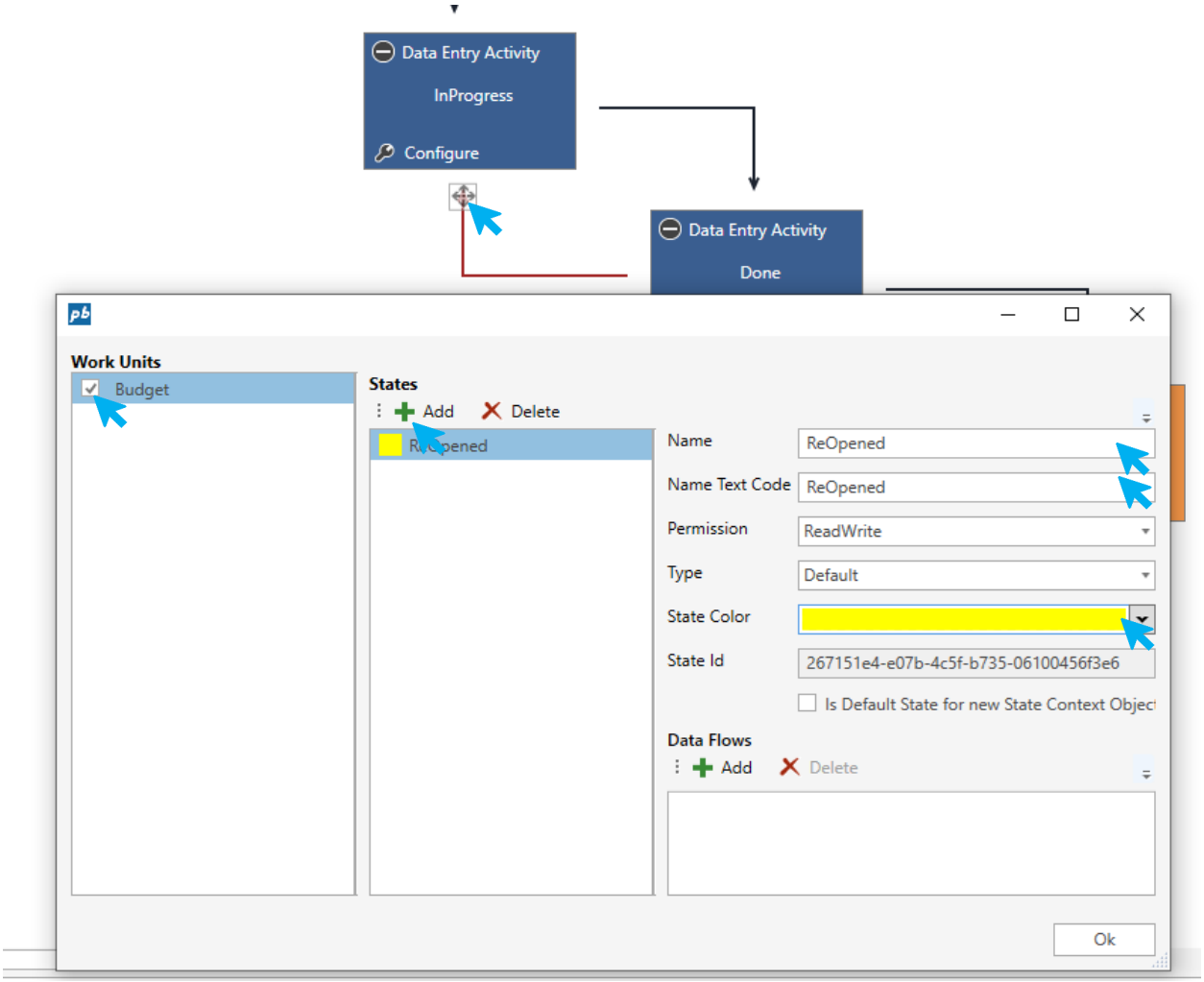
Now let's set up the user group access to this level. Recall that the Contributor workflow user group (Standard user role) do not have access to this level.

1. Click on "Configure" on the "Finished" Data Entry Activity
2. Go to the "Contributors" tab
3. Use the "Add" button to add Process Administrator to contributors
4. Click "Close" and Save



15.4 Reopened Activity

1. Click on the arrow between the “Finished” and the “In progress” boxes.
2. Click “Edit…” in the properties pane for the “Flow control object setup” property
3. Check the “Budget” check box under Work Units
4. Click the “Add button” under States
5. Fill out the new state as shown in the picture below
    a. Name: Reopened
    b. Permission: ReadWrite
    c. Type: Default
    d. Color:
6. Click “Ok” and Save



Finally, we have all the stages in the workflow configured. It is time to start a new iteration, configure tables to be archived and set up the workflow in the workbook.

15.5 Start new iteration

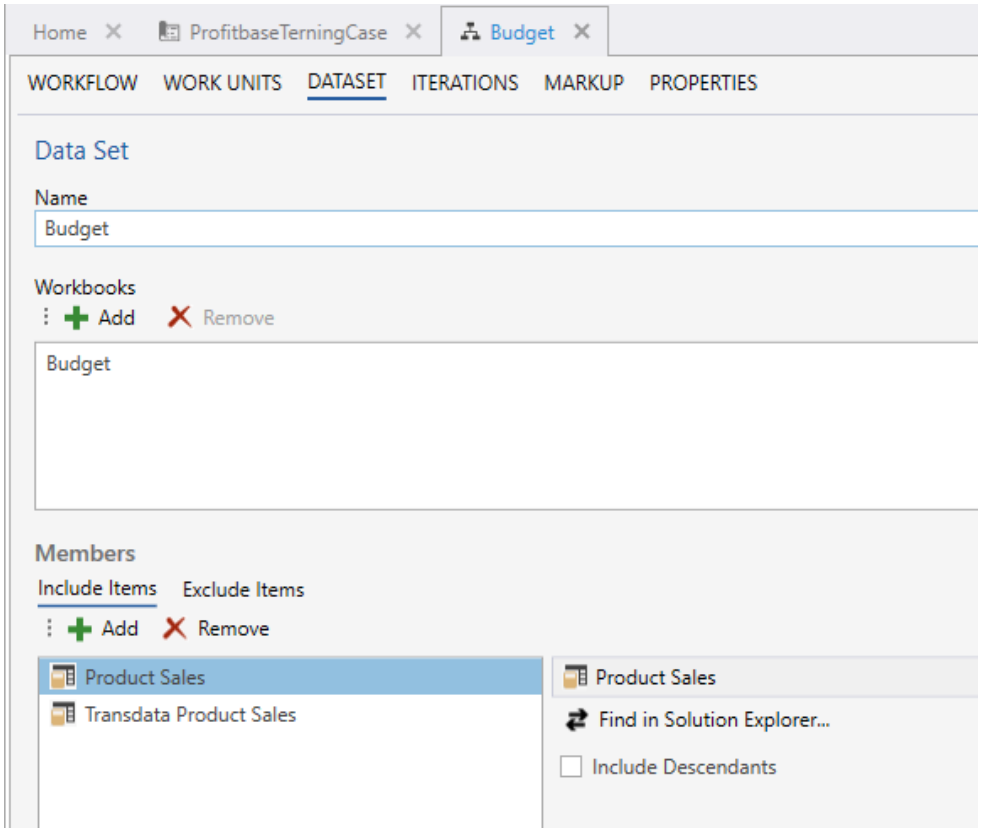
An iteration is the name of a planning period resulting in a single dataset. You can think of this as versioning in the planning process.

1. Go to the “Iterations” tab in the workflow window
2. Click “Start New Iteration”
3. Find the pbVersions table in the solution explorer, right click and browse the table to see the active IterationID

15.6 Configure Archive

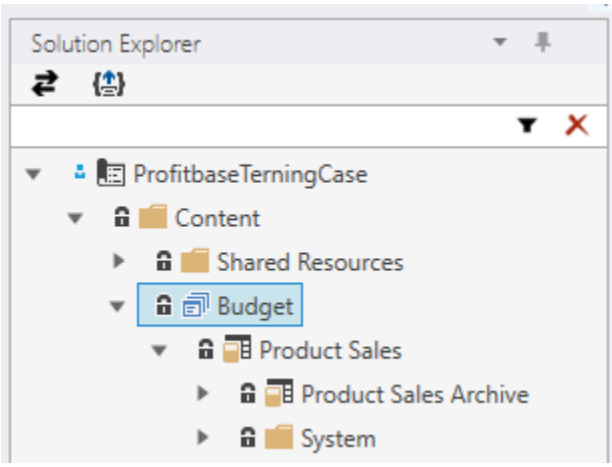
1. Go to the “Dataset” tab in the workflow window
2. Enter the name of the dataset of your solution (Budget)
3. Add the Budget workbook in the Workbooks area
4. Add the following data stores in the Members area
   These are the elements being archived
   a. Product Sales
   b. Transdata Product Sales

   Make sure to deselect the “Include descendants” checkbox



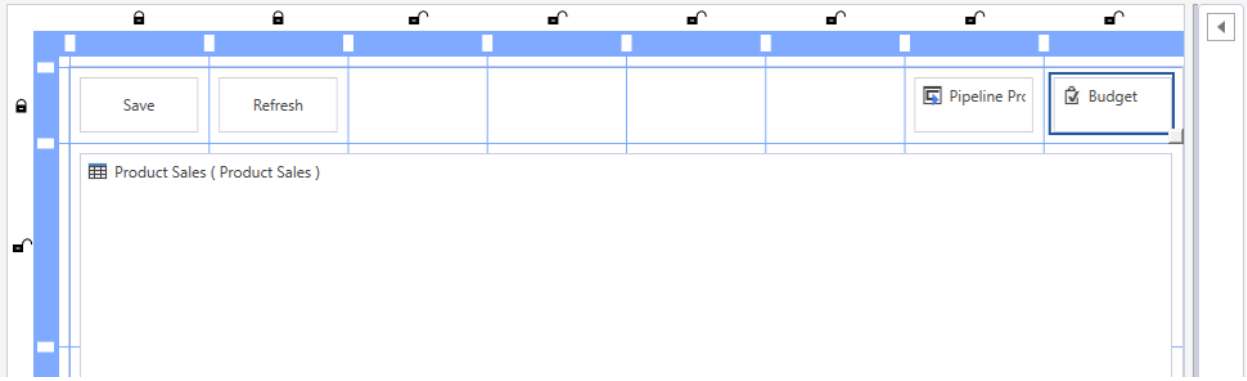
After you save you may need to reload the solution connection (close all tabs and reopen the solution from the home tab) for the archive data stores to be generated.

You should see a new store appear below the items you included in the members pane, in the solution explorer. These contain the same columns as the item you archived, with an additional PBDataSetVersionID column representing the IterationID from the workflow.



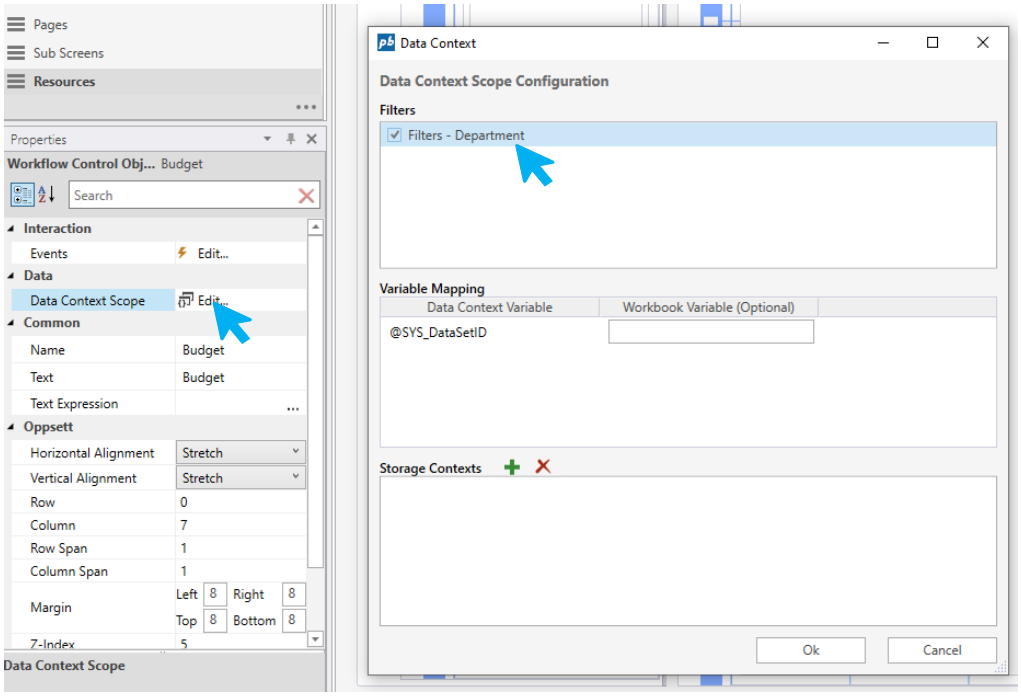
15.7 Configure workflow in workbook

1. Open the Budget workbook
2. Locate the “Budget” Workflow Work Unit in the Toolbox
3. Drag and drop this item in the top right cell of the Product sales workpage



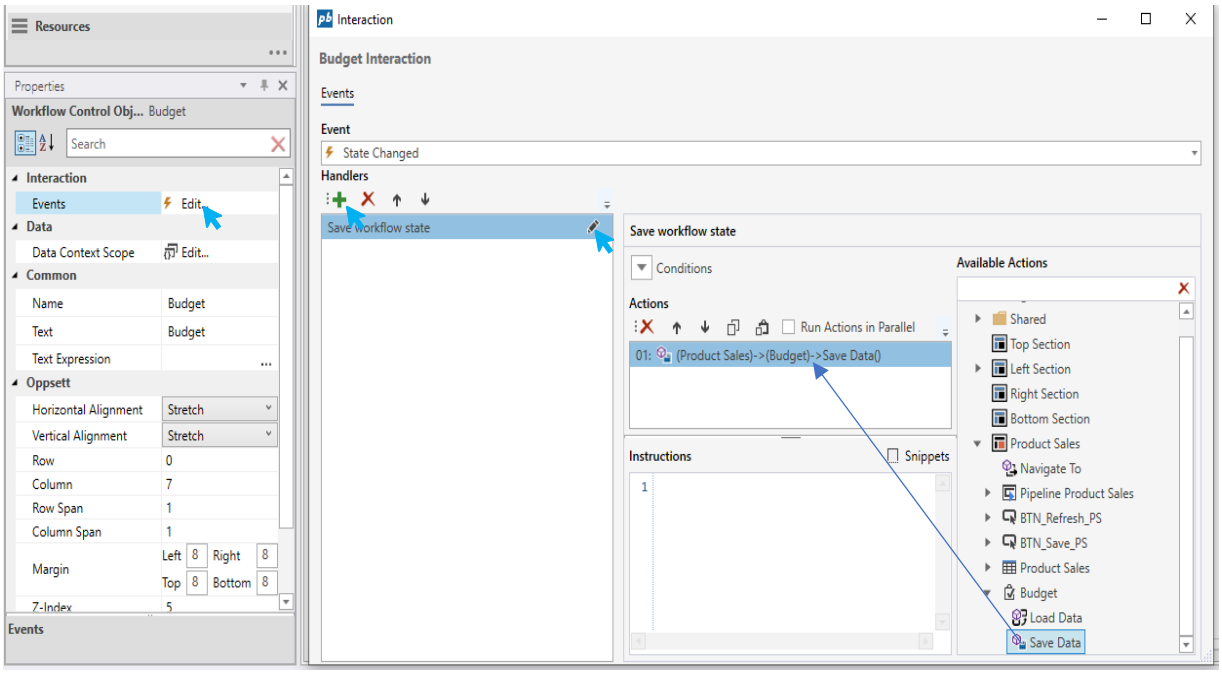
4. Select the work unit and click “edit” Data Context Scope in the properties pane
5. Select the department filter and click “ok”

   This connects the workflow status to the selected department in the filter

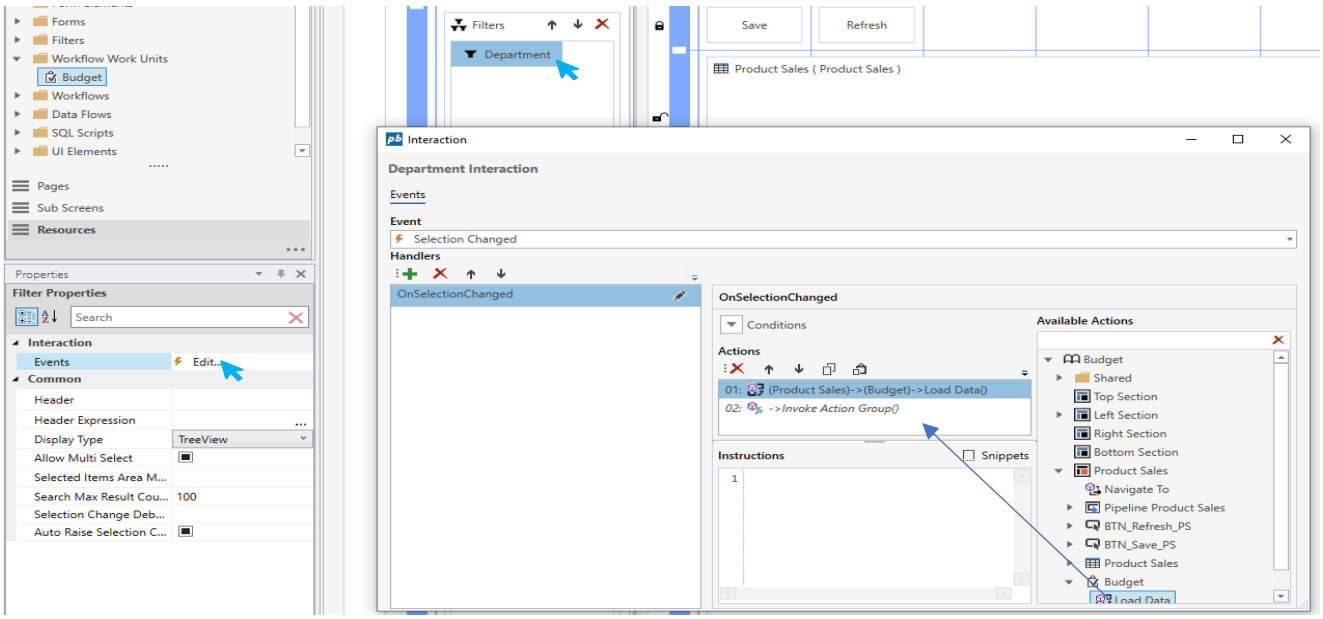


6. Click edit events
7. Add a “State changed” event handler by clicking on the green plus sign.
8. Rename the handler “Save workflow state” by clicking on the Pencil icon
9. Locate the “Save Data” action on the “Budget” Work unit on the Product sales in the Available actions pane on the right side of the Interaction popup.
10. Drag and drop the above-mentioned action into the Actions field
11. Close the interactions popup and save

    (see image on the following page)

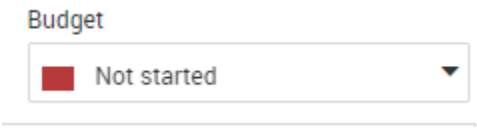


12. Click on the department filter
13. Click on edit events in the properties area
14. Add the “Load Data” action on the “Budget” Work unit on the Product sales in the Available actions pane to the actions field.
    Make sure it is above the invoke action group by using the arrows above
15. Close the interaction popup and save



Now let’s preview the workbook and look at the result.

You should see the workflow element in the top right corner of the Product sales workpage. It will only load a status when you select a department I the filter.



It should switch to “In progress’ the first time you save a department. You can also manually change the status in the drop down. To see if the input limitations work you can place you own user in the Standard user role and refresh the browser. You should not be able to edit any cells or change status if the status is set to finished.

NB: Make sure you place your user back in the Admin user role before you close the designer.

16 Exercise

Create a new datastore for planning on account.

You will need the following:

- Account data store
  - Departmentid (key) – nvarchar(50)
  - AccountID (key) – nvarchar(50)
  - Total – Decimal(18,4)
    - TimeFrame column (check the inherit from ancestor box in the datastore)
  - Comment – nvarchar(250)

- Get data from the fact table and transform it as follows
  - Union the amount and COGS in a single column
  - Hard code the amounts for product sales to account 3000
  - Hard code the COGS to account 4000

- Account worksheet
  - Get all the columns from the datastore
  - Hide the department column
  - Add department filter
  - Create dropown on the AccountID column
    - Use the AccountID_Name column from the account dimension as display name and AccountID as Value member and value target
  - Add a read only rule for the AccountID column that locks saved rows
    - See rules on the AccountID_Name column in the Product Sales worksheet
  - Add a summary row (default)
    - See properties in the worksheet
  - Format the Total column
    - Format string: 0,0
  - Set caption expression of the Total column (DateStart("yyyy"))

- A distributer ruleset table (setting table like the account mapping rule set)
  We will use this to distribute a total amount for the year on months.
  - Columns
    - DepartmentID – nvarchar(50)
    - AccountID – nvarchar(50)
    - TypeID – nvarchar(50)
    - P01 – P12 - Decimal(18,4)

        - Pri - bit
    - Set Departmentid and accountid as ranked input on their respective dimensions.
        - Remember to use Pri as specificity (order by) under the ranked input tab
    - Configure the transaction pipeline distributer
        - Distribution key column: DepartmentID
        - TypeID column: TypeID
        - Factor columns: P01 – P12
    - Post the distributer in the setting list on the admin page
        - Just add the new distributer ruleset table to the Product sales setting list.
    - Create a default distribution key for all departments and all accounts that is flat
        - 1 in all columns (P01 – P02)
        - Remember that the typeID must match the typeID in the transaction pipeline setup in the store (next task)

- Configure a transaction pipeline for the distributer in the account datastore
    - AccountID should be an input and additional output column as opposed to in the mapper we made for product sales.
    - Total should be the computation column
        - Remember that the TypeID must be identical as the TypeID in the ruleset
    - We want the transactions to end up in the same transdata table as the ones from the sales store. (target table)
    - Remember to enable the Distributer feature with the checkbox

- Create an account pipeline dataflow
    - Create a DataFlow named AccountPipeline
    - Create a DataFlow Item named Derive data Account
        - Use the “Run transaction pipeline” task, under datastore tasks
    - Add your new dataflow item in the new dataflow.
    - Edit the store ID in arguments under properties to: @Object[Account, Store].Id

It’s now time to connect the components in the client.

- Post the worksheet on a new page in the budget workbook
    - Connect to the department filter

    - Set up loading and saving actions
        - Use invoke action group towards the loading action on the refresh button on the department filter and on navigated to
        - Add the pipeline at the end of the saving action like we did in sales.
    - Try to add a new row, input some number in the total column, save, and see if the distributer works. In which case you will have 12 transactions in the transdata store for the single input row.

        Note: you will probably find transaction generated from the Product sales store as well. This can be filtered by the SYS_OriginID column, which reflect the store ID that generated the transaction.

- Connect the new worksheet to the existing workflow
    - Use the same workflow unit in the workbook as for sales

- Add the Account store to the global context so we avoid editing accounts for departments we are not working with

- Set up an archive on your new datastore
    - Add the store under workflow – dataset – members