2022

Workshop: Business Central Excel Layout Reports Demystified

WORKSHOP

STEVEN RENDERS – RENATO FAJDIGA

Table of Contents

Introduction

Learn in this workshop, using the Excel report layout, how you can create a report that generates a dataset and leave it up to the end-user to further modify it by using the full palette of capabilities in Excel such as sliders, diagrams, charts, pivot tables, and Power Query to design the report.

Discover how you can offer flexibility and freedom for the end-user, being able to change the look and feel of a report, adding additional views, filtering, and sorting on data, using the Excel Layout reports in Business Central.

This workshop will focus on the development of the ideal dataset and of course also the Excel layouts.

Environment setup

To be able to do the workshop you require a **Business Central Saas environment**. To be able to create the Business Central trial environment, you either require a **business email address** or you can create a new business email address by creating an **Office 365 trial**.

Once you have a BC Saas environment, you then need to create a **Sandbox**.

To develop new reports and datasets, you need Visual Studio Code and the AL extension.

If you already have the following:

- BC Saas Sandbox
- VSCode + AL Language

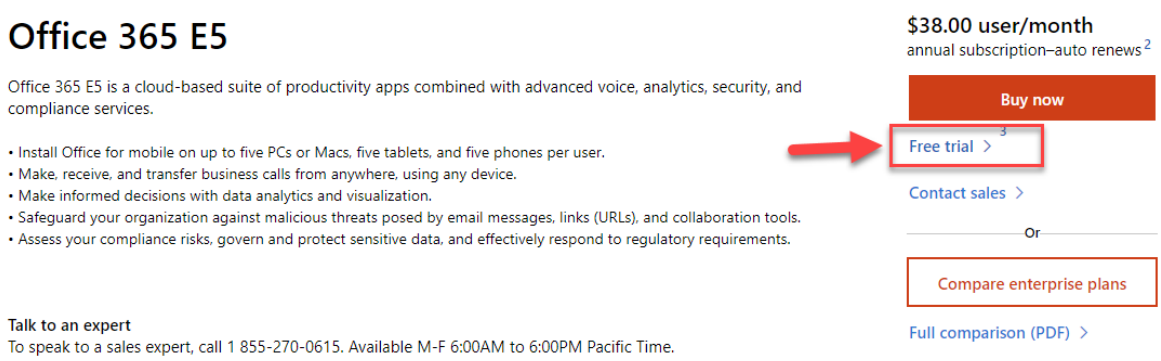
Then you can skip the following steps and start with **Developing AL Reports**.

To create an Office 365 trial, follow these steps:

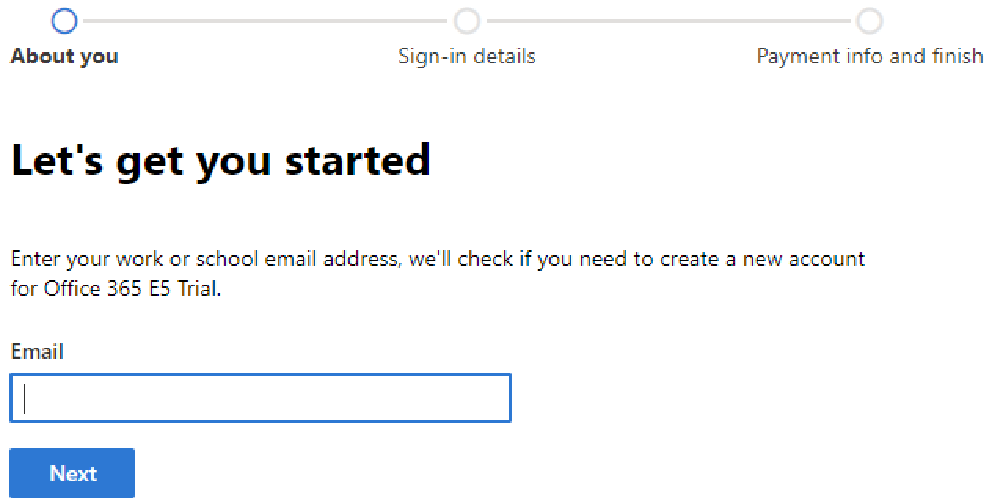
Open your browser using an inprivate/incognito browser window.

Navigate to the following page: https://www.microsoft.com/en-us/microsoft-365/enterprise/office-365-e5?activetab=pivot%3aoverviewtab

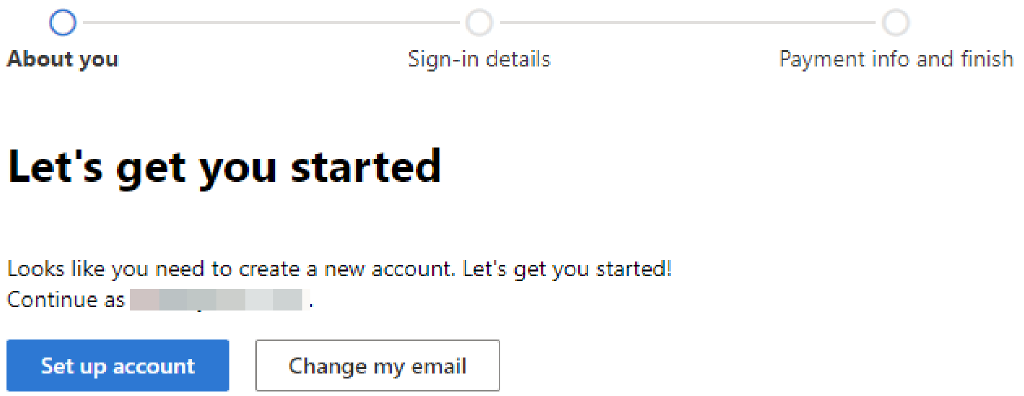
Select **Free trial**:



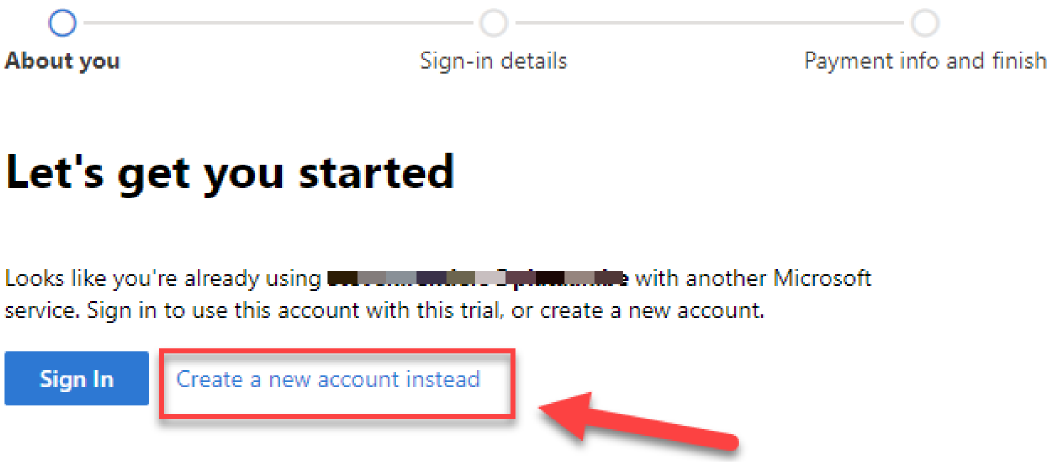
In the following window, enter your business email address, and click **Next**:



The following is displayed:

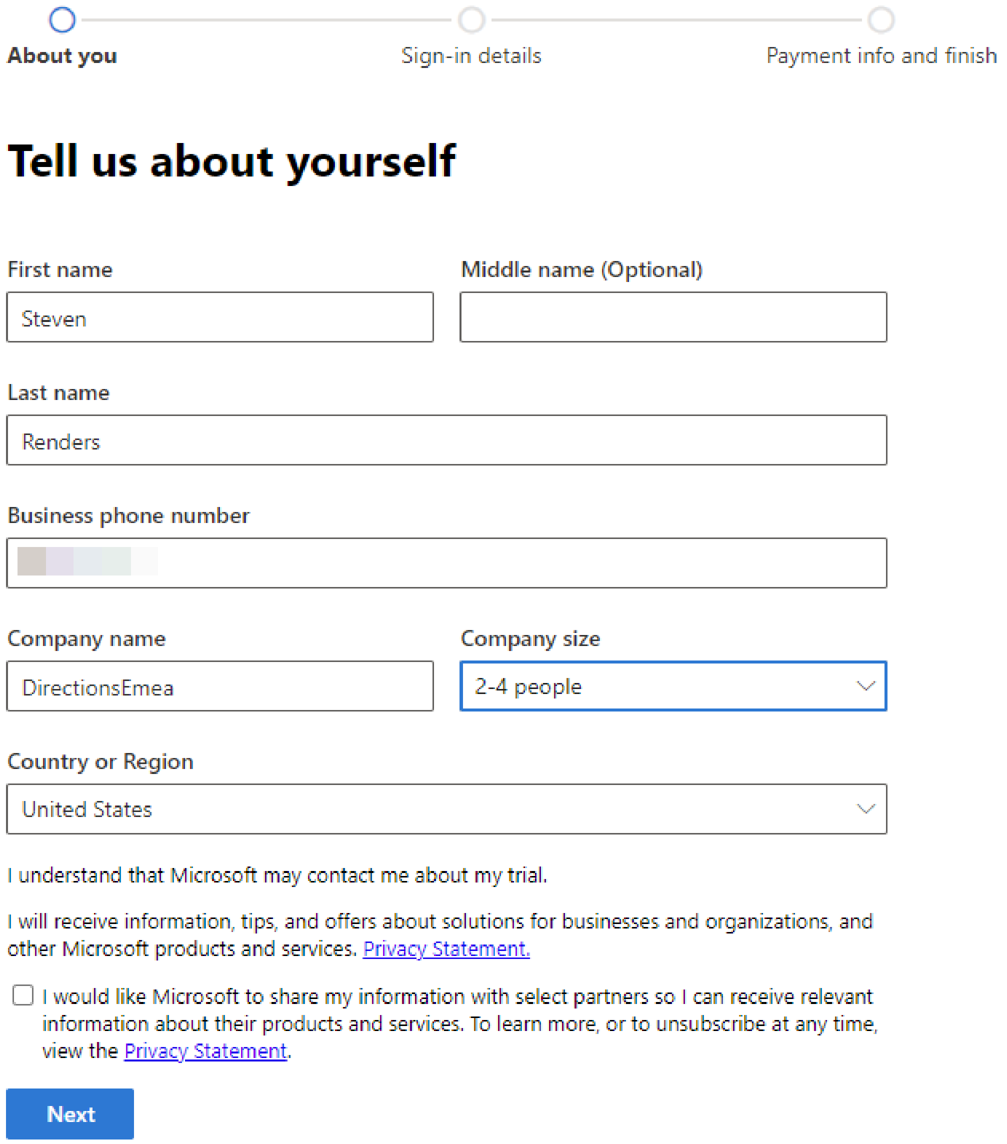


If you see the following message:



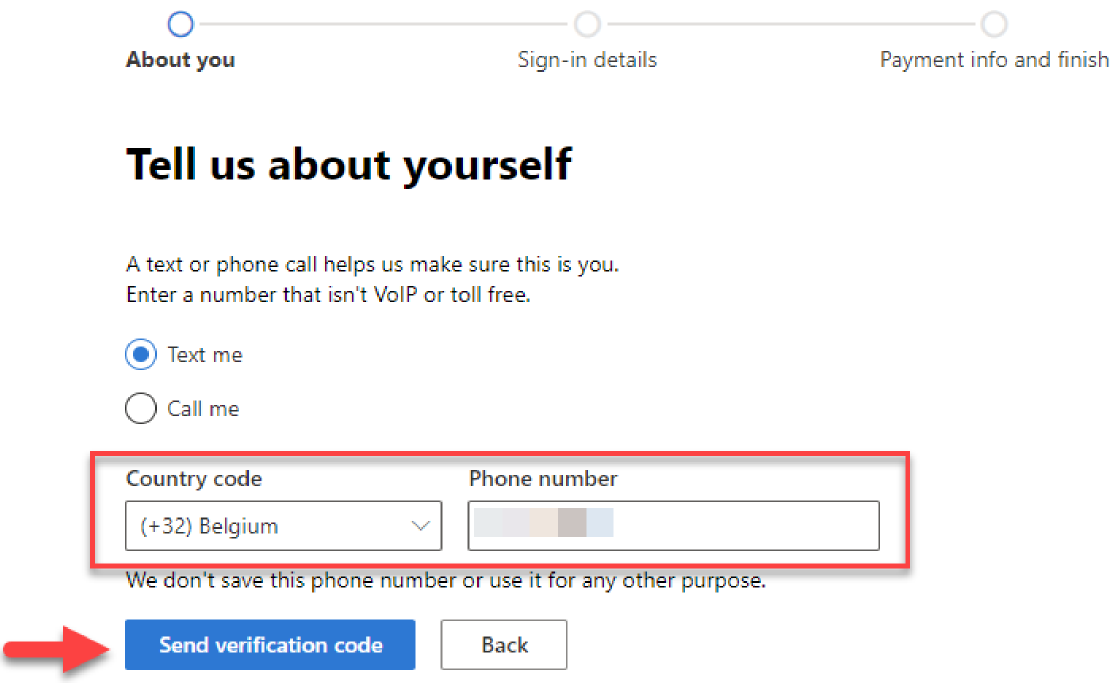
Then select: **Create a new account instead**.

On the next page, enter your information. The only thing that needs to be correct is your business **phone** number, because a confirmation **text message** will be sent to it in order to create the Office 365 trial.

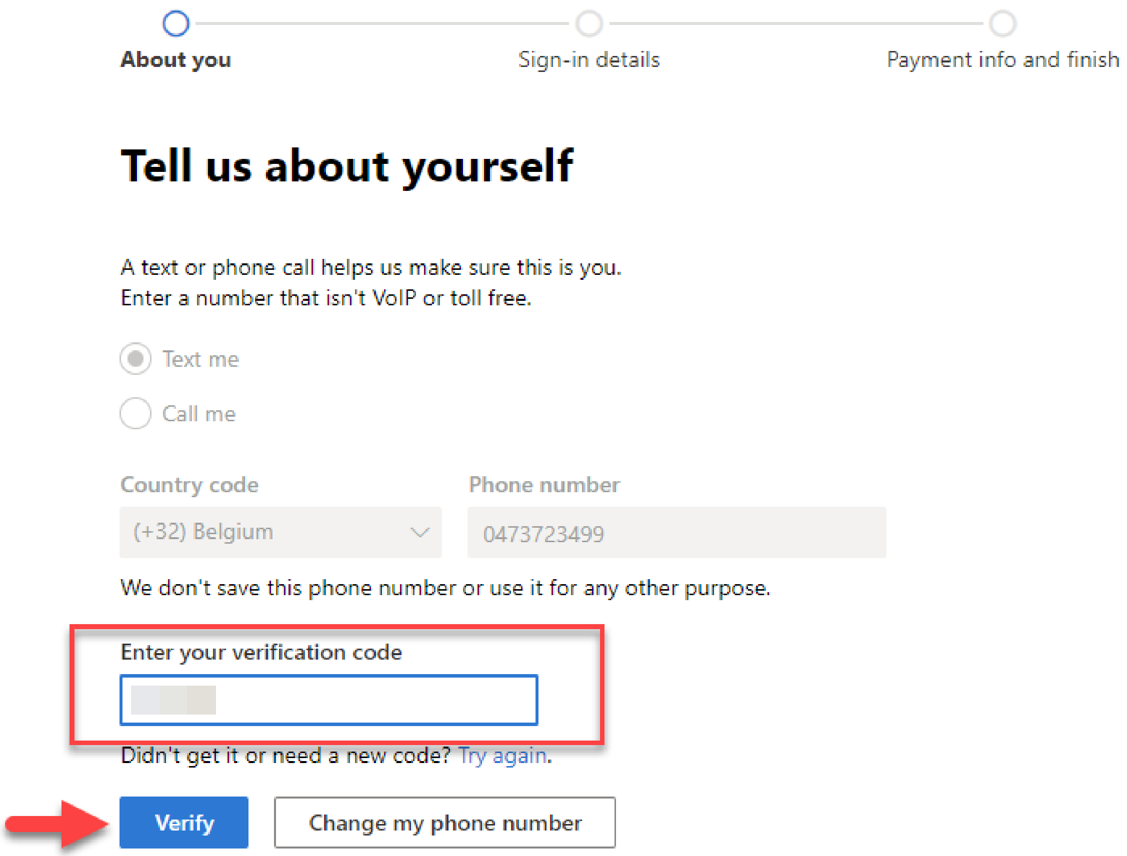


Click **Next** to continue.

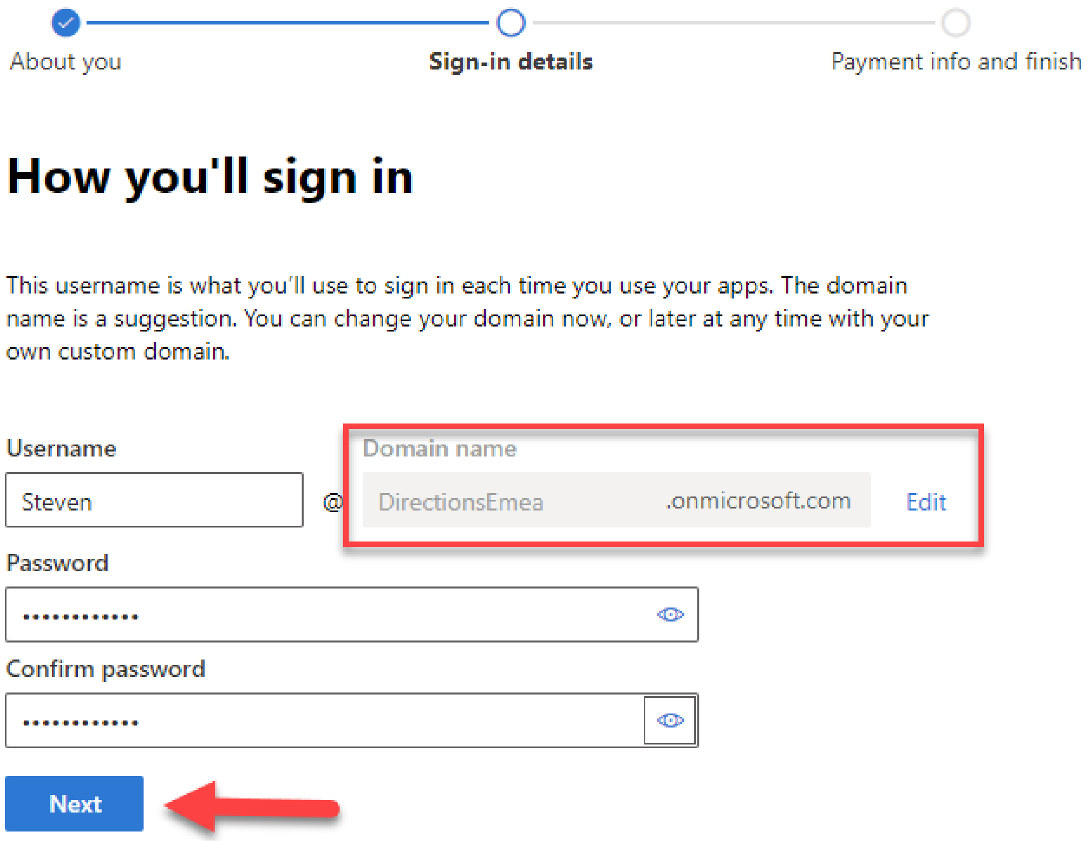
On the next page verify your phone number and country code, then click **Send verification code**:



Now enter the verification code you received:



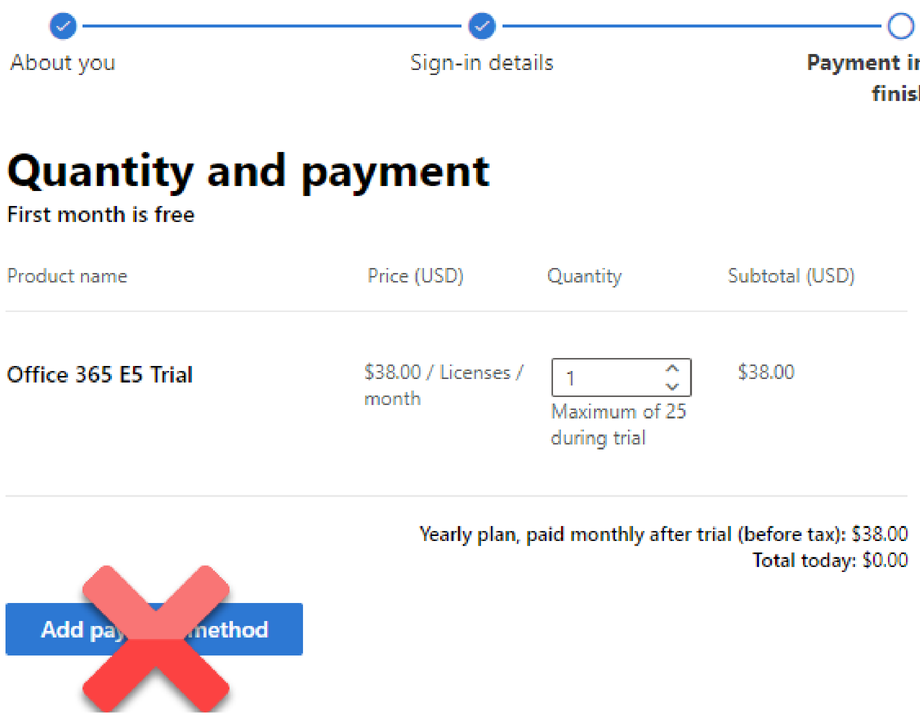
On the next page, select a username, domain name and password. You might have to try different variations of the **domain name**, because it needs to be **unique**:



Make a note of the email address and password and then click **Next**.

Your account is now being created.

In the next page you are asked to enter **credit card information**:



You do not need to enter any credit card information; you can simply **close** the browser window. Make sure you have made a note of the **email/password** you have chosen.

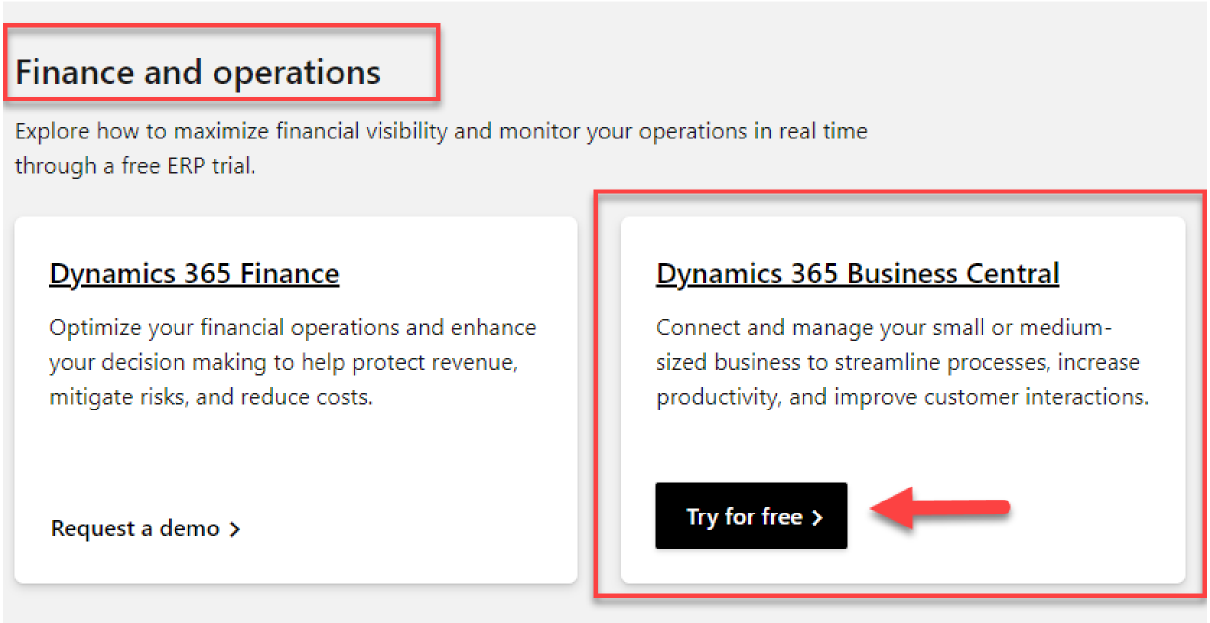
You now have an Office 365 trial.

To create a Business Central Saas trial, follow these steps:

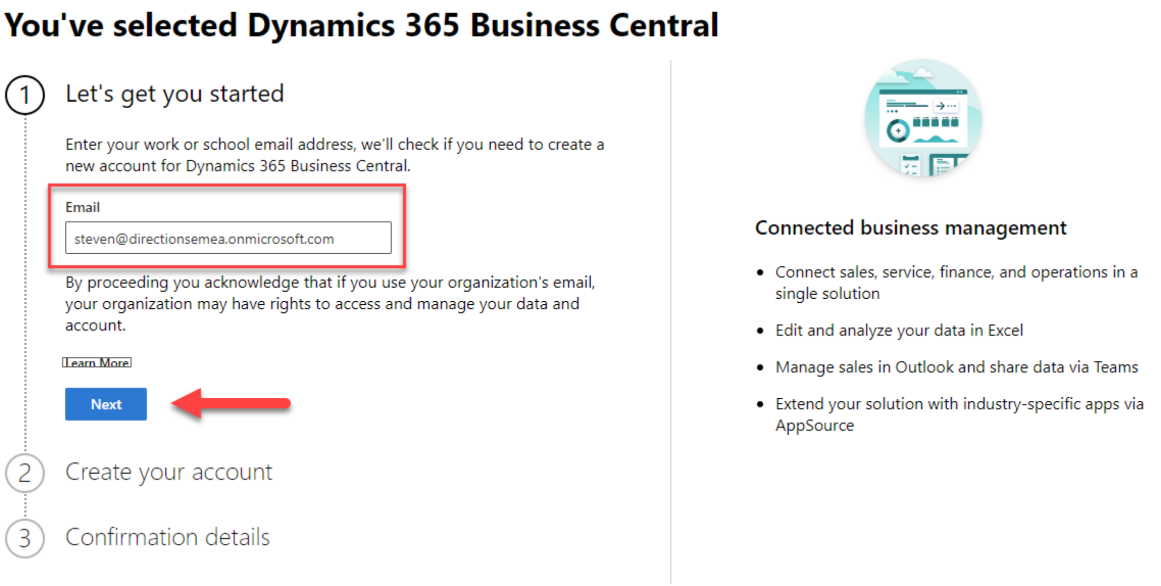
Open your browser using an inprivate/incognito browser window.

Navigate to the following page: https://dynamics.microsoft.com/en-us/dynamics-365-free-trial

Select **Business Central trial**, then **Try for free**:

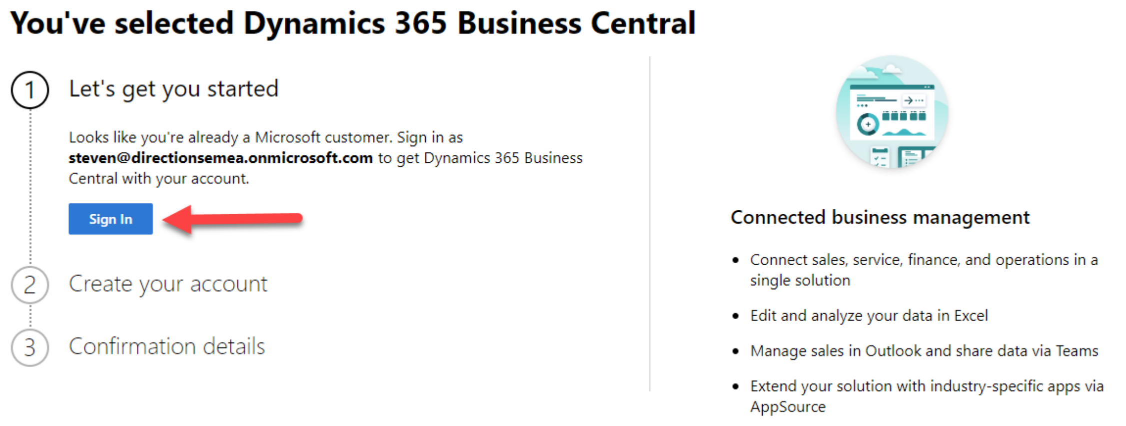


On the next page enter the **email** address. In case you created an Office 365 trial, it's the email address of the Office 365 trial you created earlier:



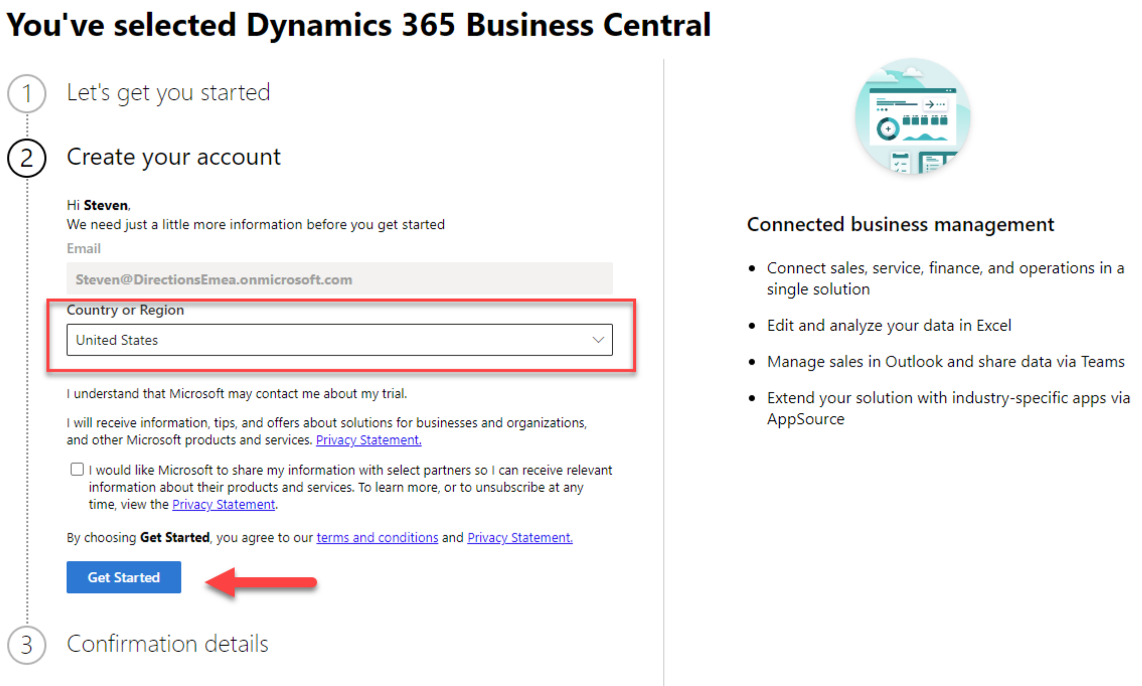
Click **Next**.

In the following page, select **Sign In**:



When/if asked, enter the **credentials** (email/password) of the Office 365 trial.

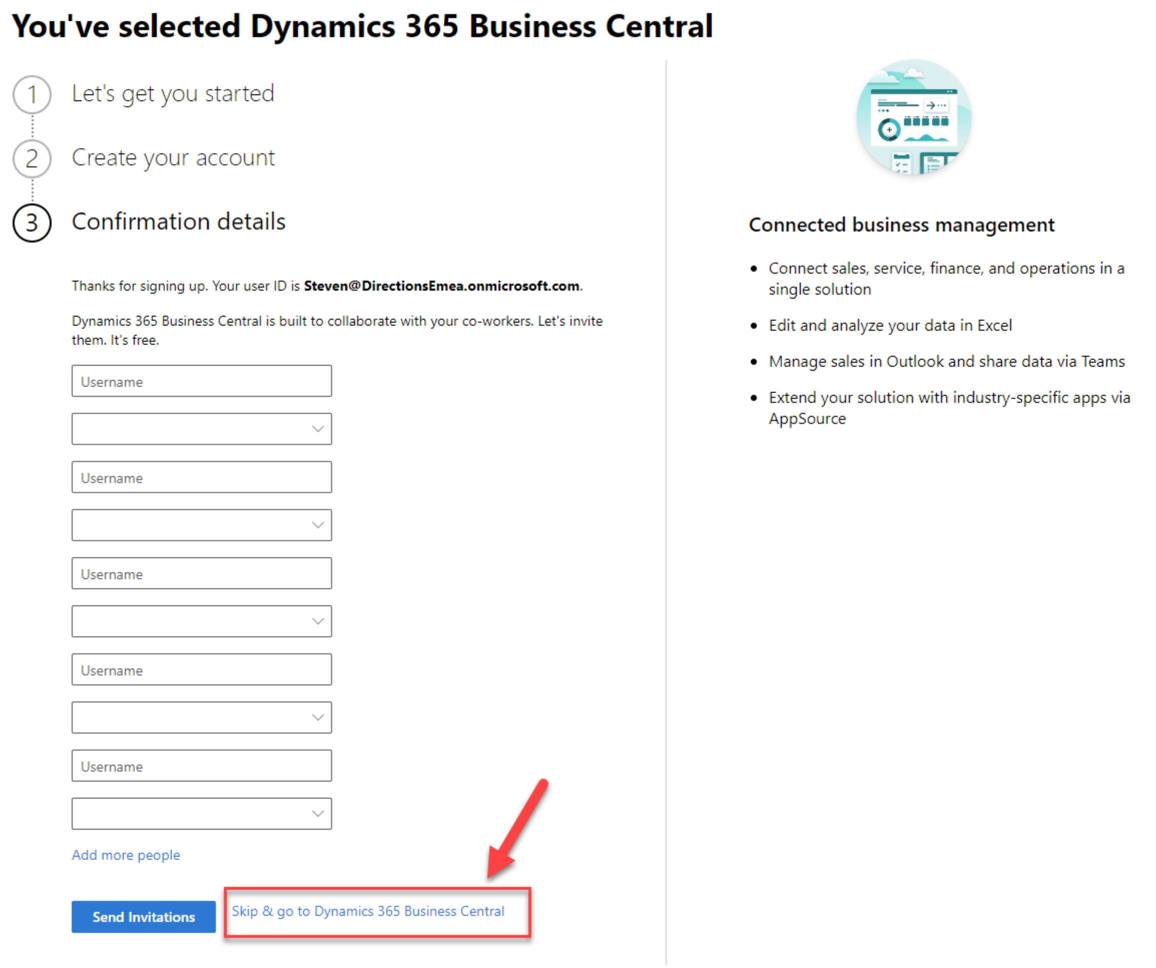
On the next page:



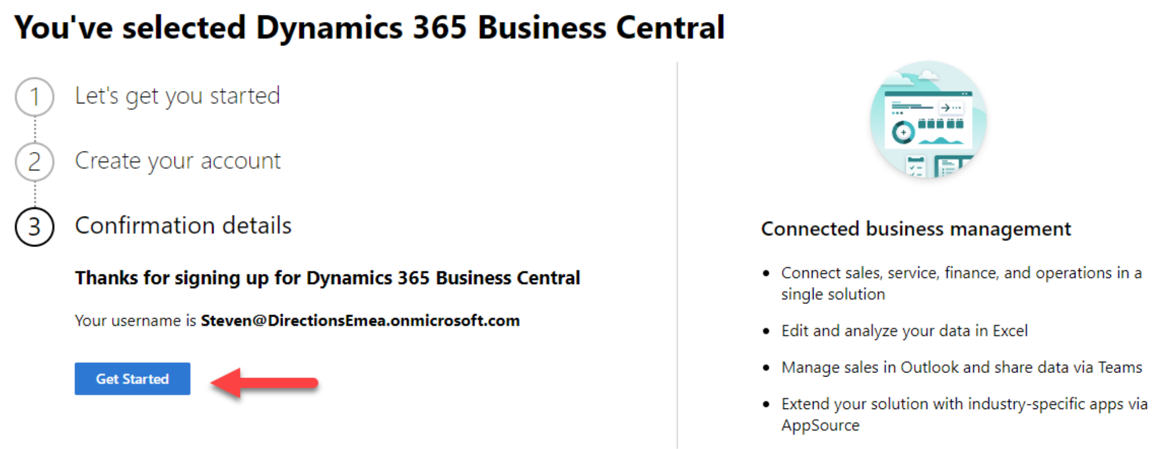
Select a **country/region**, then **Get Started**.

The country/region is used to create a Business Central Production environment. You need to select a country from the list. If your country is not on the list, then select US or UK.

On the next page, select **Skip & go to Dynamics 365 Business Central**:

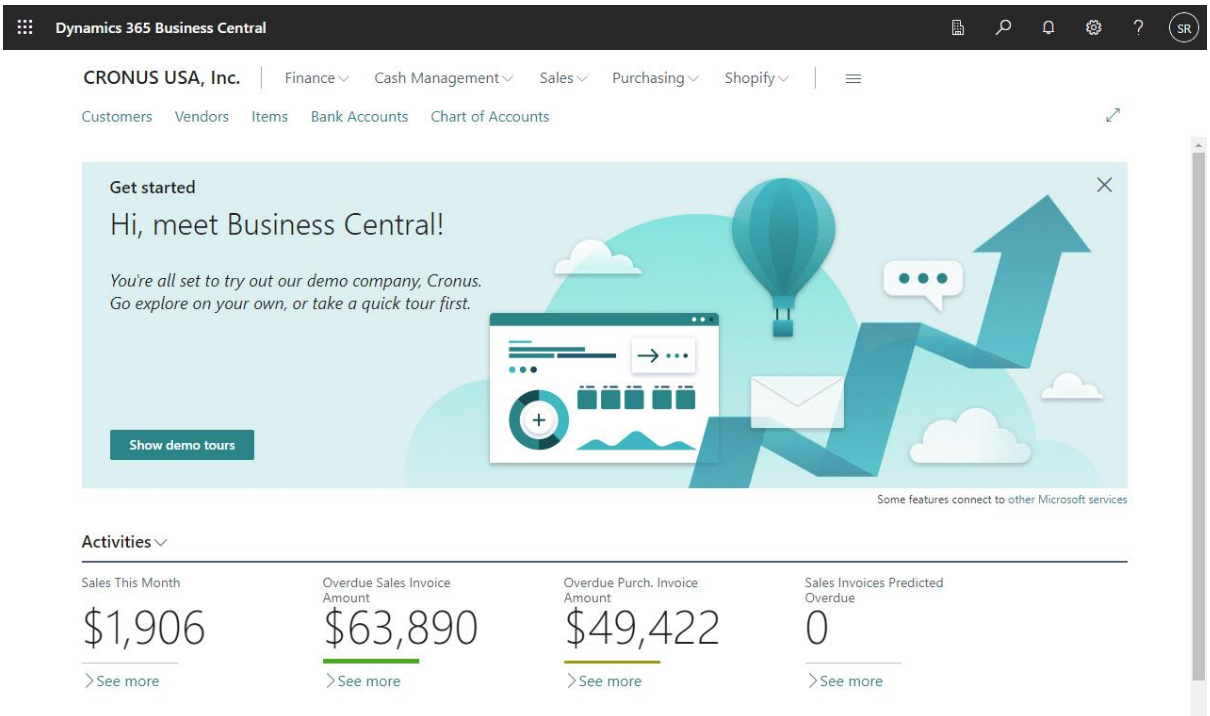


On the next page, select **Get Started**:



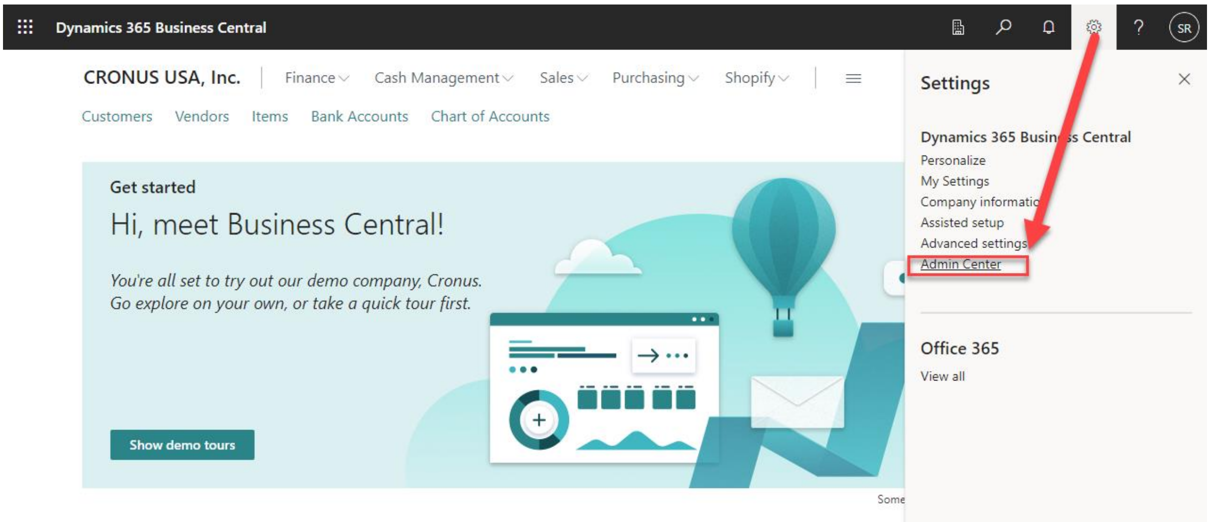
Now wait until your Business Environment is ready. This can take a few minutes.

When the BC environment is ready, your window will look similar to the following:

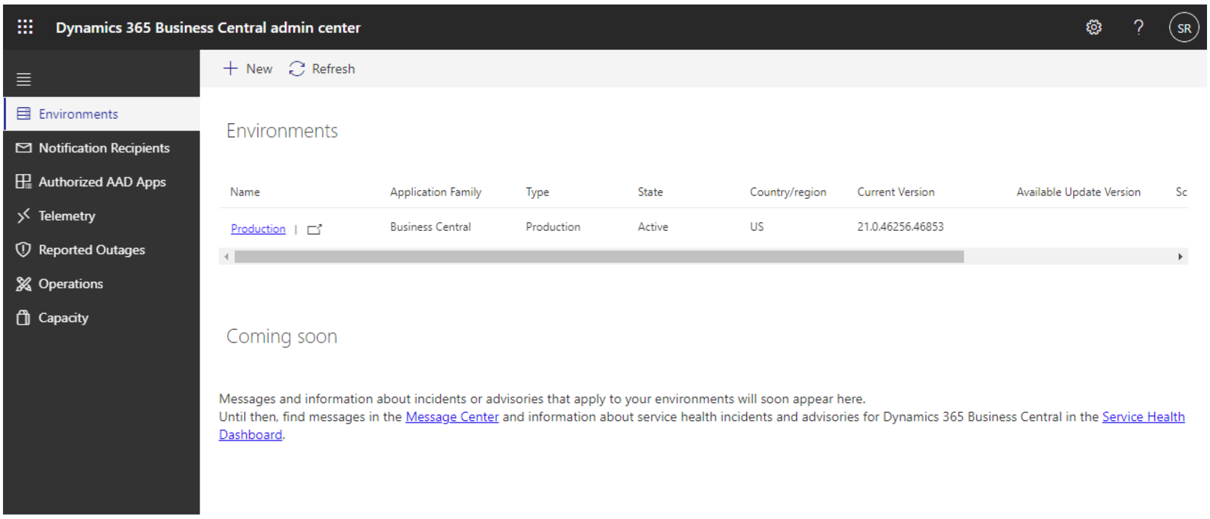


To create a Business Central Saas Sandbox, follow these steps:

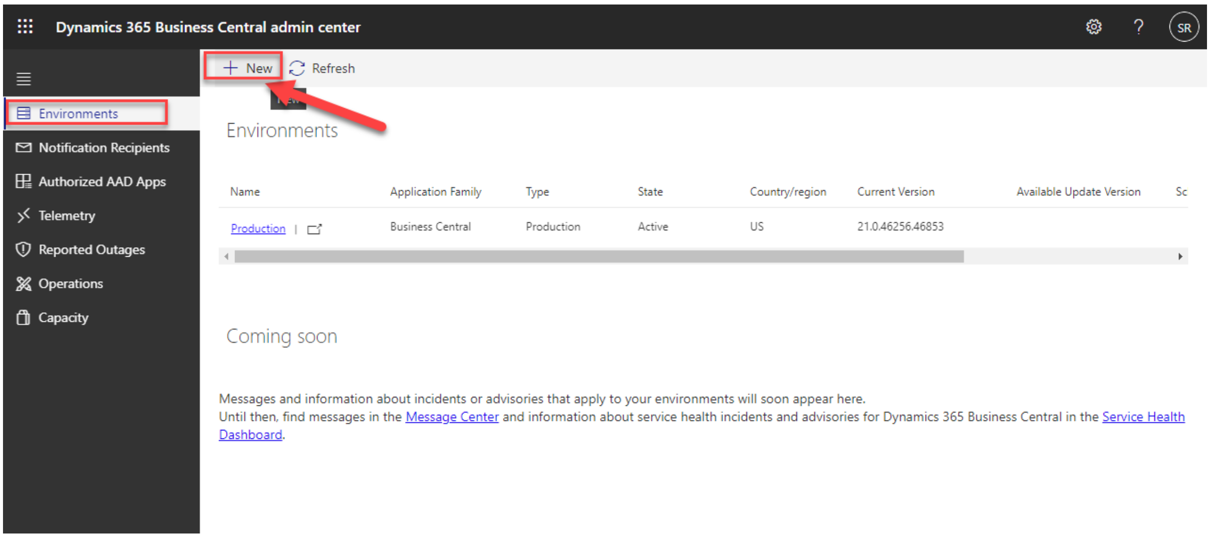
In Business Central, open the **Admin Center**, as follows:



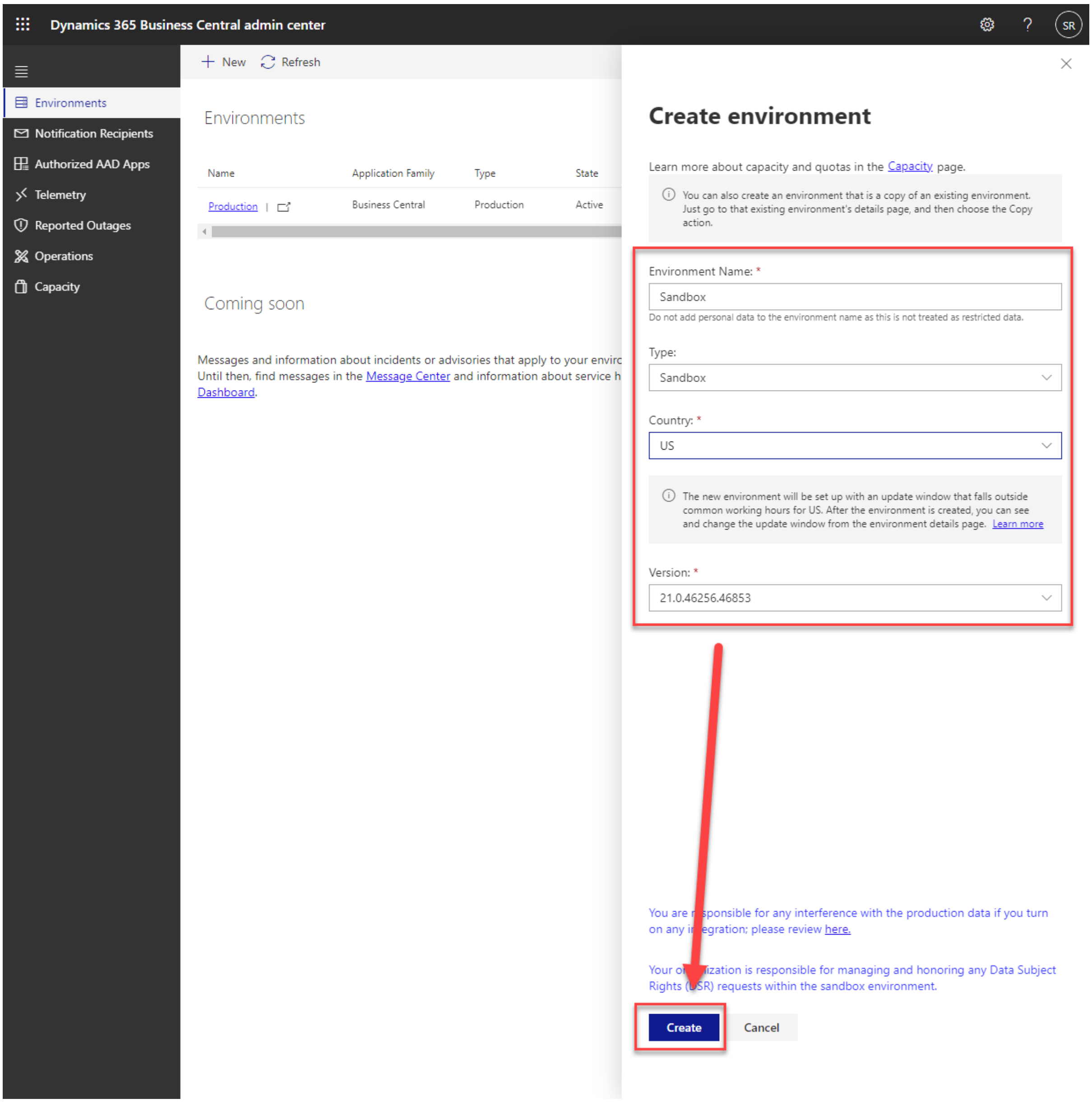
The **Admin Center** now opens:



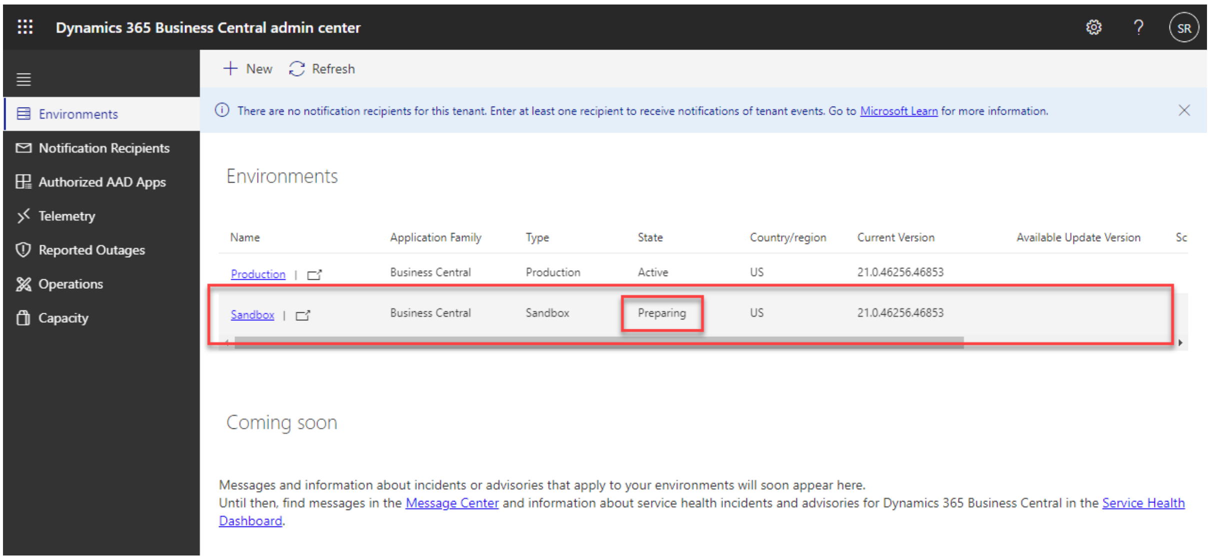
In the Admin Center, select **+New**, at the left top:



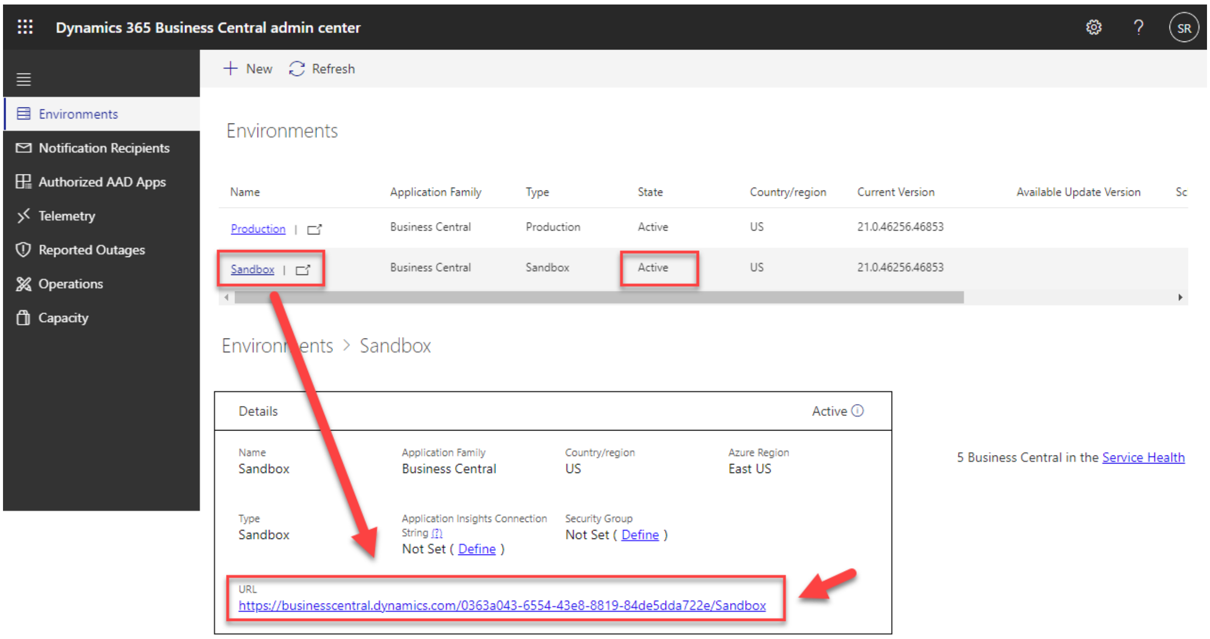
A pane opens at the right side of the screen. Enter a name: Sandbox, type Sandbox and Country US, then **Create**:



The sandbox is now being created, it can take several minutes to complete:



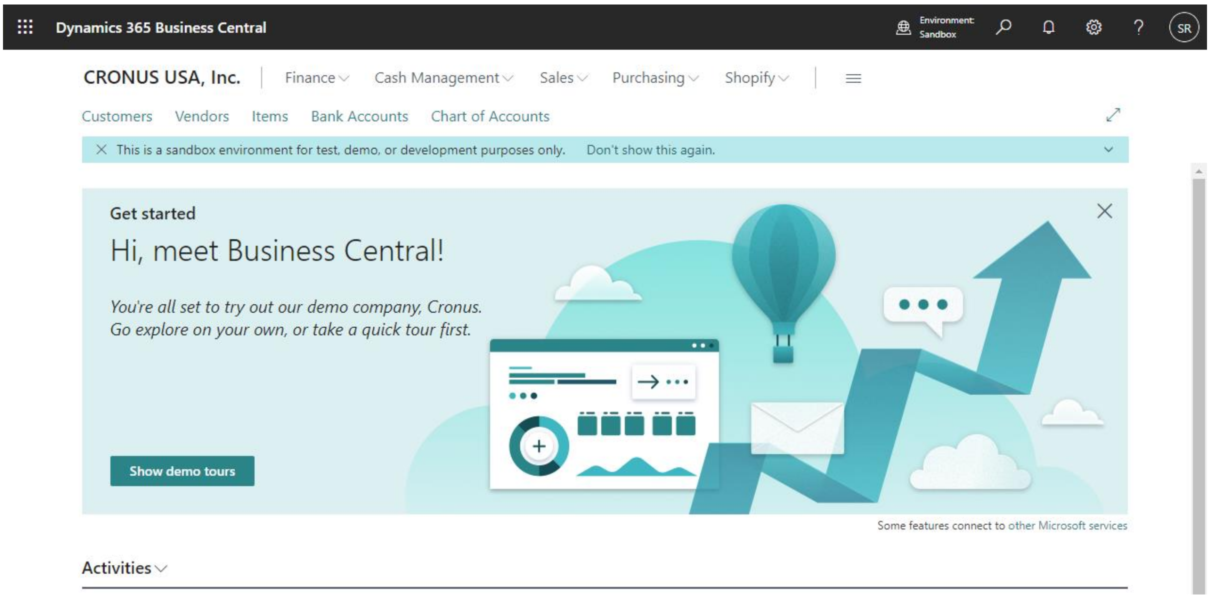
Once the Status changes to **Active**, then open the Sandbox:



Opening the sandbox, for the first time, activates the security (roles, permissions), and is required, before you use the sandbox for development.

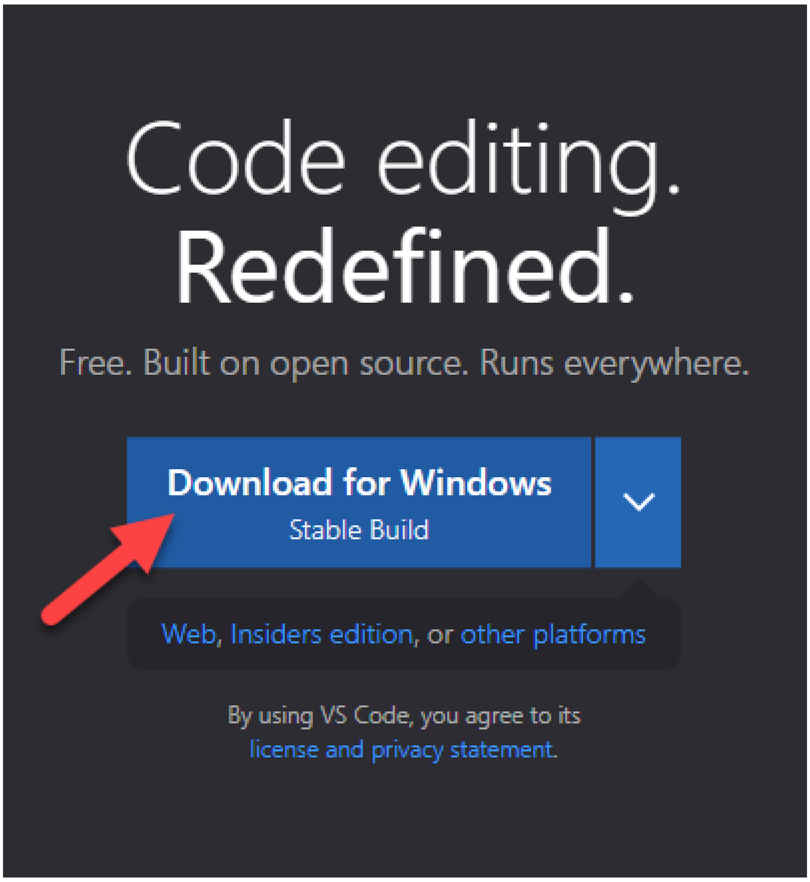
Make a note (in notepad) of the EnvironmentName you have chosen, and the tenant URL.

Your Business Central Saas Sandbox is now ready:



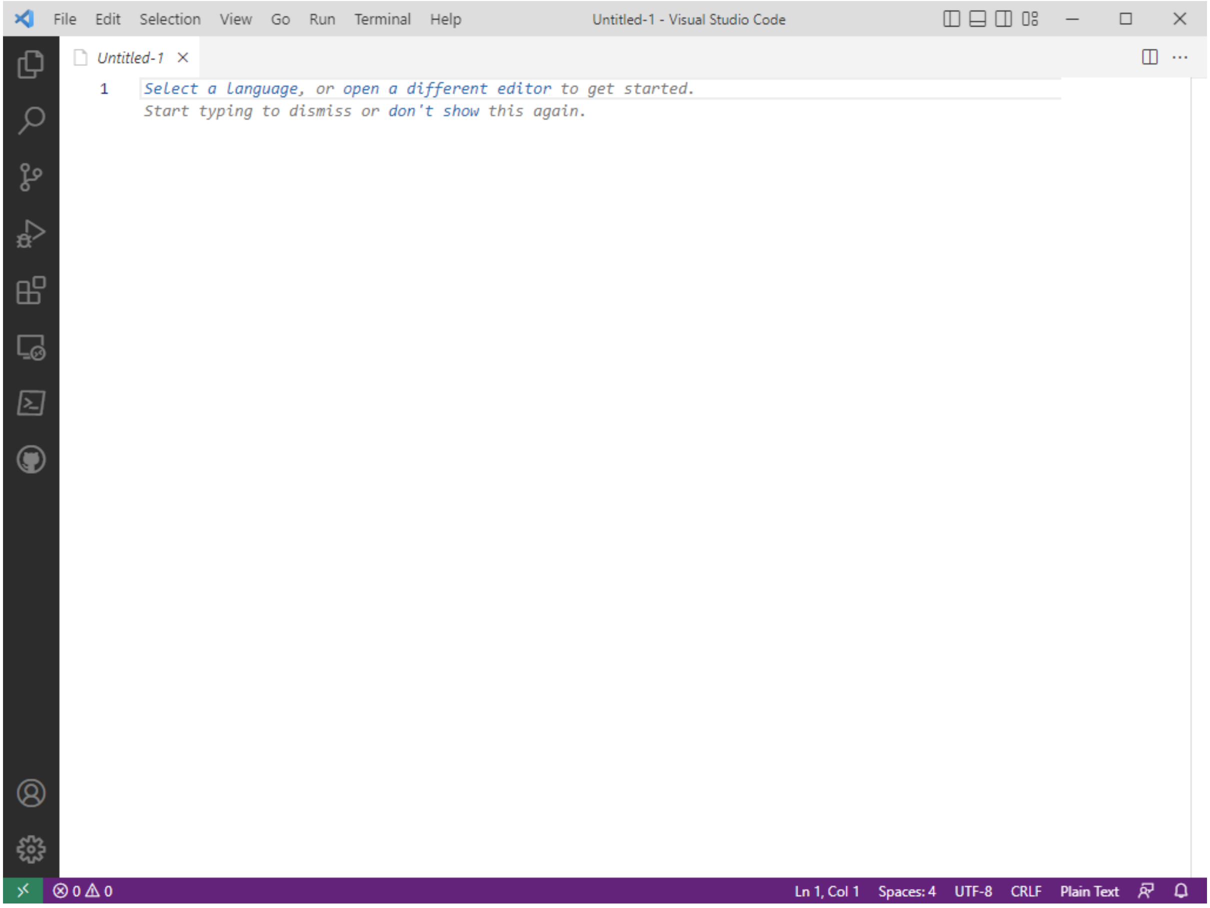
To install and configure Visual Studio Code, follow these steps:

Download and Install Visual Studio Code, from the website: https://code.visualstudio.com

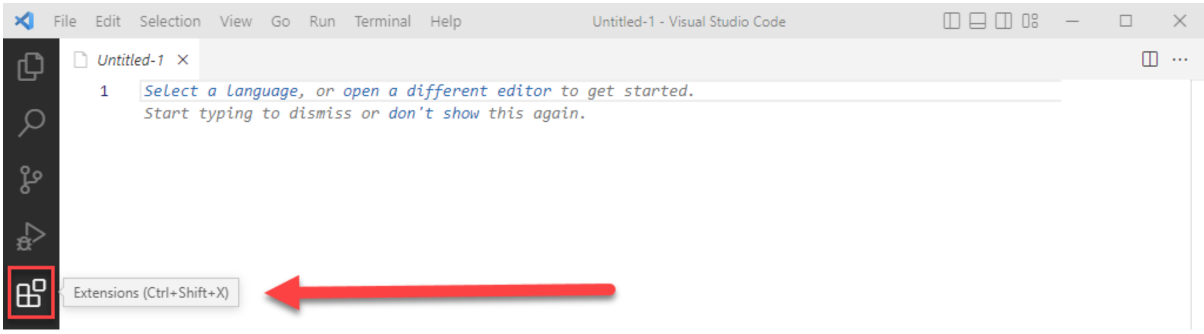


Click Download and install Visual Studio Code, using the default proposed settings.

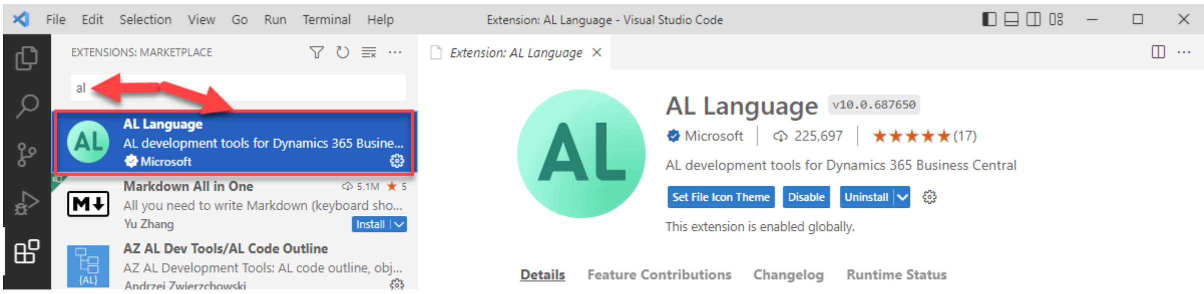
Once VScode is installed, open it:



In VScode, select the Extensions button:



Look for the AL Language extension from Microsoft and install it:



Developing AL reports

We will create the following three new reports:

- Finance
- ItemAvailability
- Shipments

Excel layouts can be added to Business Central Report objects, so this means a developer can do it when creating/developing a new report, but also in Business Central a non-developer (user) can always add new layouts to existing reports.

The difference is that in Business Central, a user cannot change the dataset for a report. So, if you required a certain field in the dataset of a report, in order to create a new layout, you have to ask a developer to either extend the dataset of the existing report or to develop a new report.

Although extending the dataset of an existing report is possible, it is mostly not recommended. The reason is that the datasets of existing reports are not optimized to be consumed in an Excel layout. You would be able to add fields/tables to the dataset of an existing report, but it would, in many cases, still not be useful to be used in combination with an Excel layout.

For that reason, we will not extend the dataset of existing reports, but we will create new reports instead.

If you are interested to learn how to create a report extension, then ask one of the instructors of this workshop, and they will surely demo it.

To learn how to develop reports for Business central, there's a lot of documentation available:

- https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-reports
- https://learn.microsoft.com/en-us/training/paths/build-reports

The above links are good starting places to learn more about report development.

In this workshop, we will guide you, step by step, how to create the 3 reports, but for more in depth information and documentation we recommend to have a look at the above documentation on Microsoft Docs and Microsoft Learn.

Besides that, we also offer other, in depth, training options:

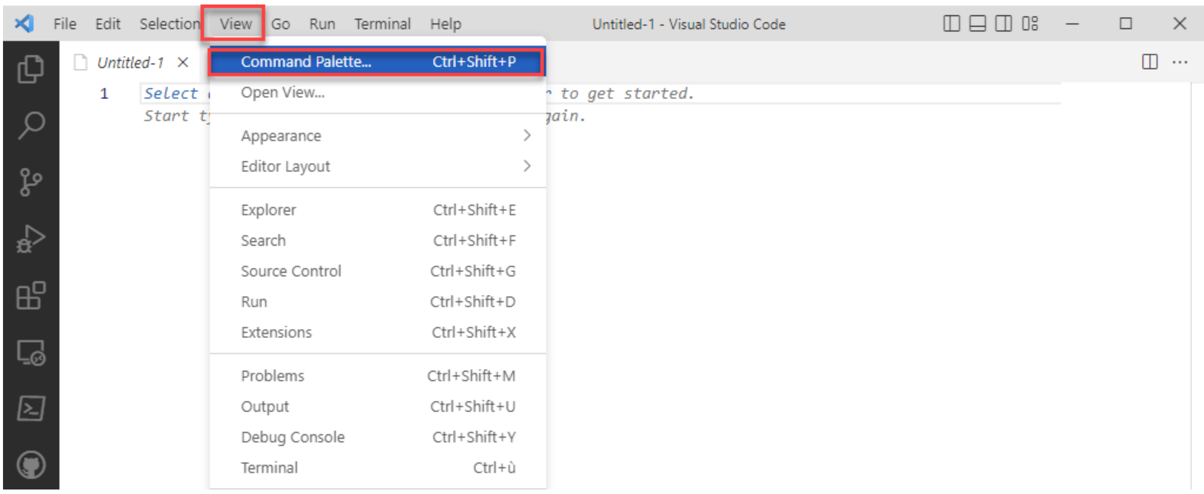
- https://businesscentralbooster.com
- https://plataan.tv

If you don't want to develop the extension in VSCode, but simply want to deploy the solution, the go to: **To import the extension in Business Central, follow these steps**

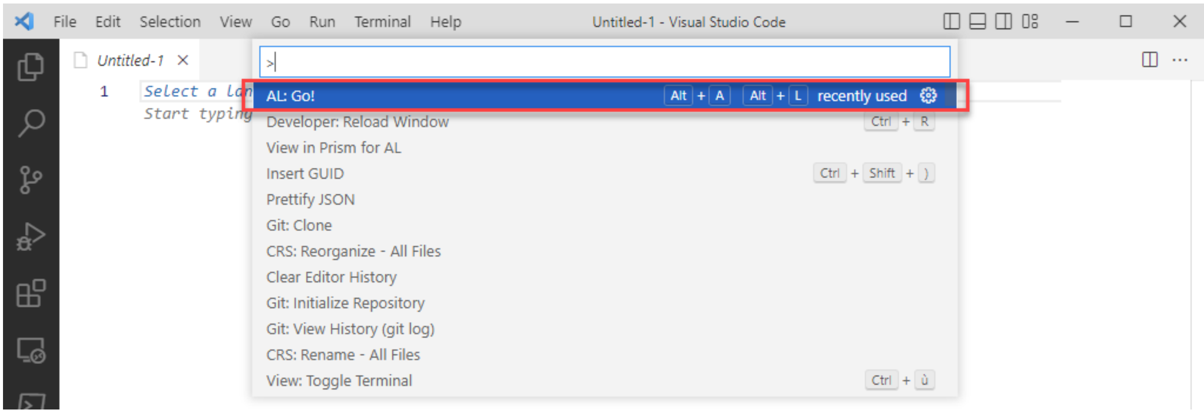
To create a new extension (app), follow these steps:

We will now create a new extension (app) in VScode, named: **WorkshopExcelLayouts**.

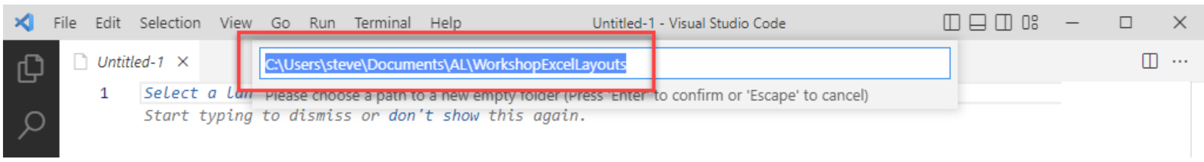
In Visual Studio Code (VSCode), open the Command Palette (F1):



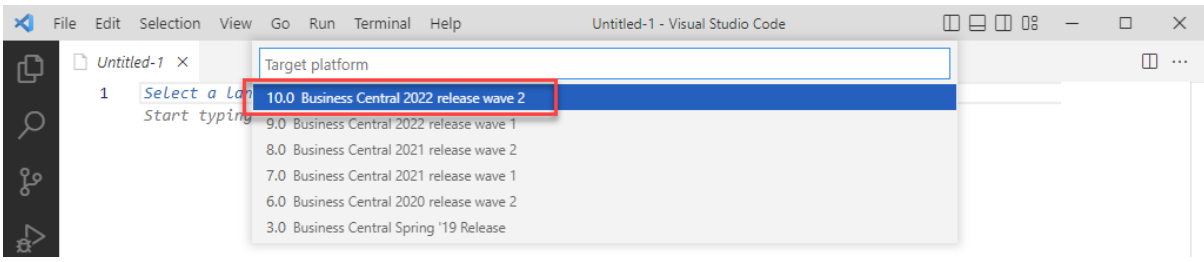
In the Command Palette select Al:Go!



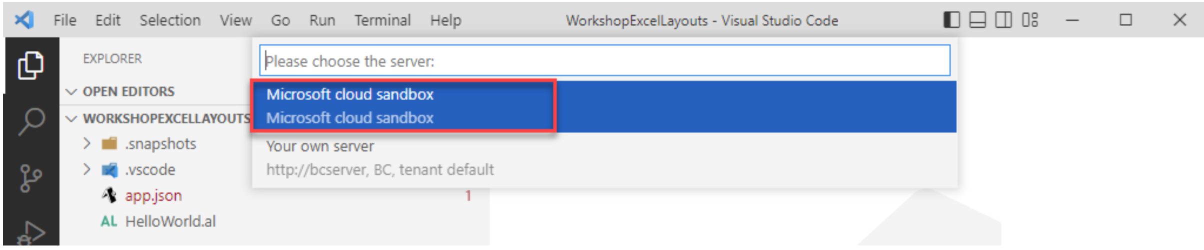
Now, enter a name (and path) for your app:



Hit enter, and then select the latest Target Platform (10):



Then select Microsoft cloud sandbox:

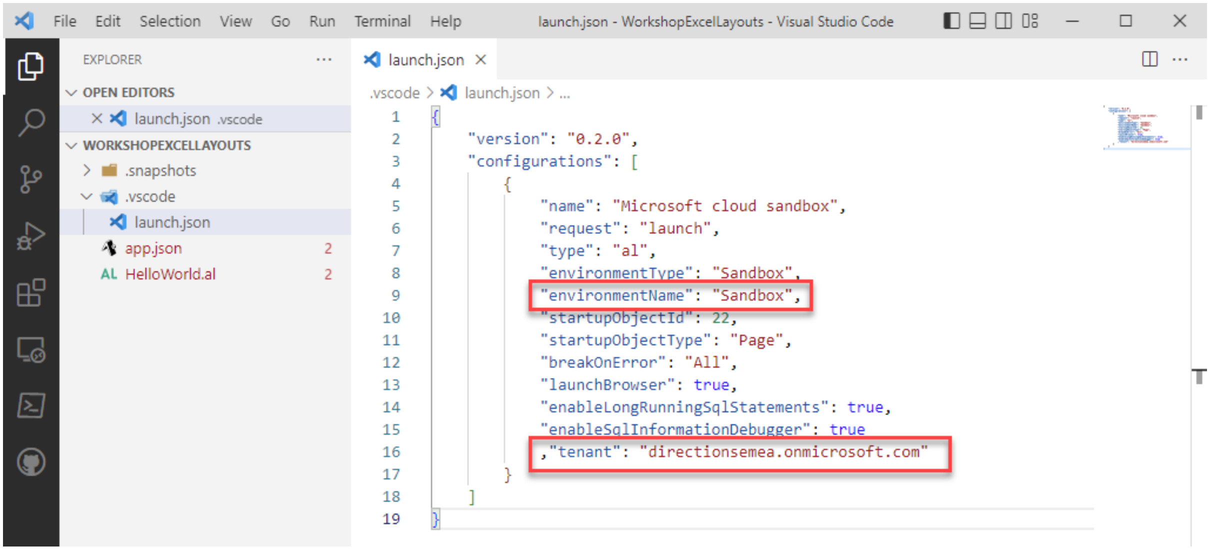


Some files are now created, and you might see some error messages. Ignore that.

Open the Launch.json, located in the .vscode folder.

In the launch.json, enter the exact name of the sandbox you created in the setting: environmentName. Be aware this setting is case sensitive.

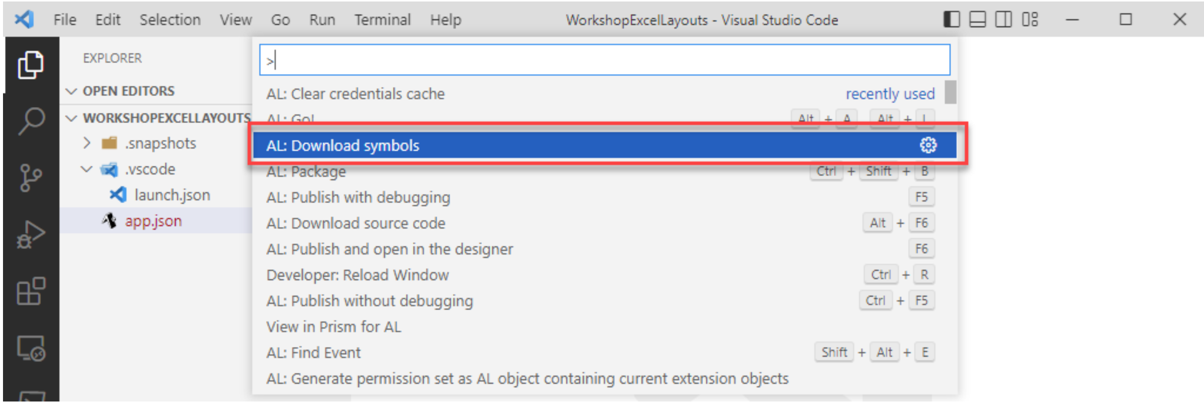
Then, add a tenant parameter, and in the tenant parameter enter your domain:



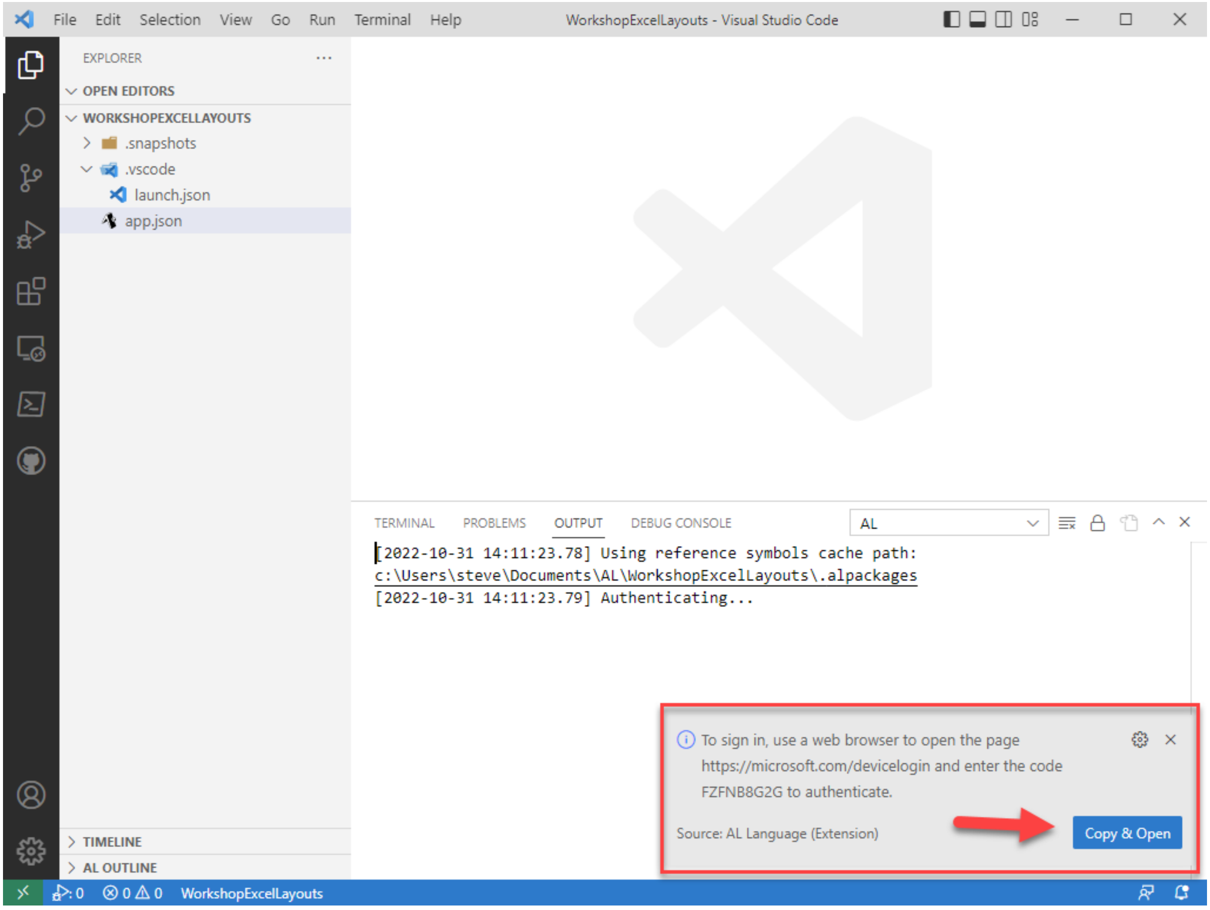
In the above example, I'm using the email steven@directionsemea.onmicrosoft.com to login to my BC trial, so my domain is: directionsemea.onmicrosoft.com

Now delete the **Helloworld.al** file.

The next part is to connect VScode to Business Central and Download Symbols. This means the Base and System apps will be downloaded from BC into VScode. To do this, open the Command Palette again (F1) and select: **Download Symbols**:

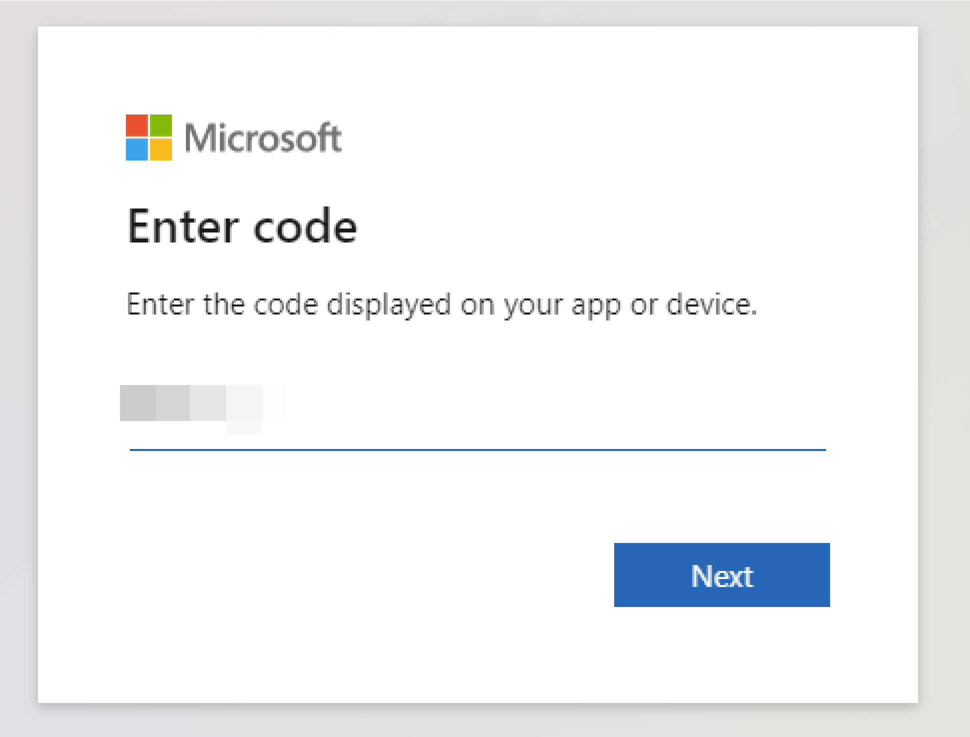


At the right bottom of VScode, a message appears:



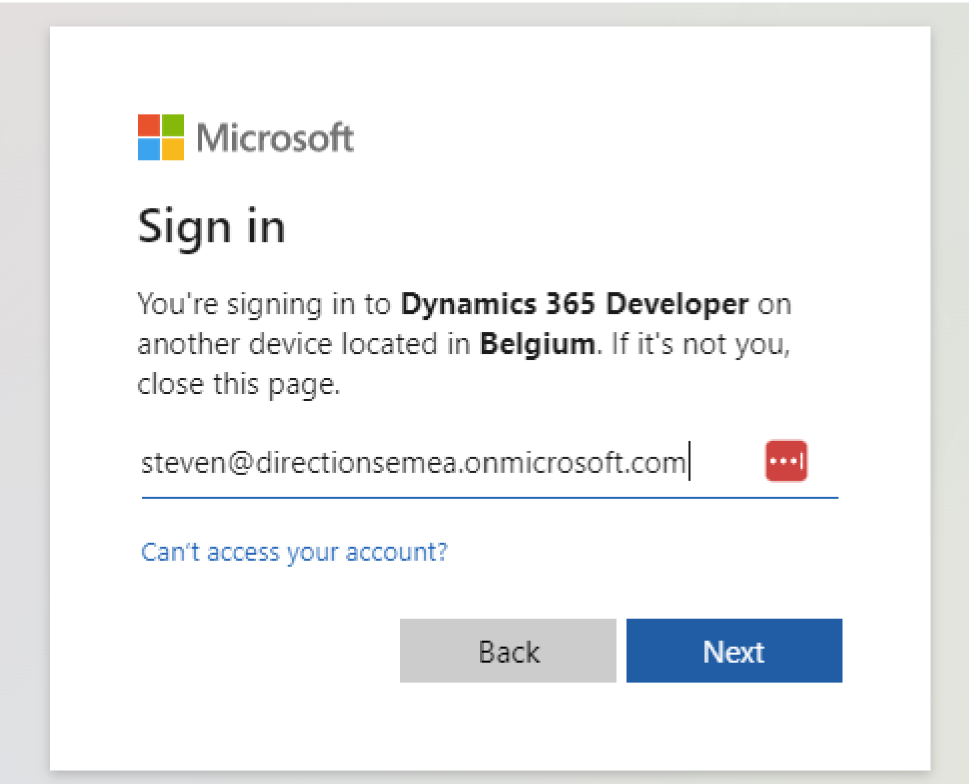
Click the **Copy & Open** button, to open your browser.

In the browser windows that opens, simply **paste** the code:



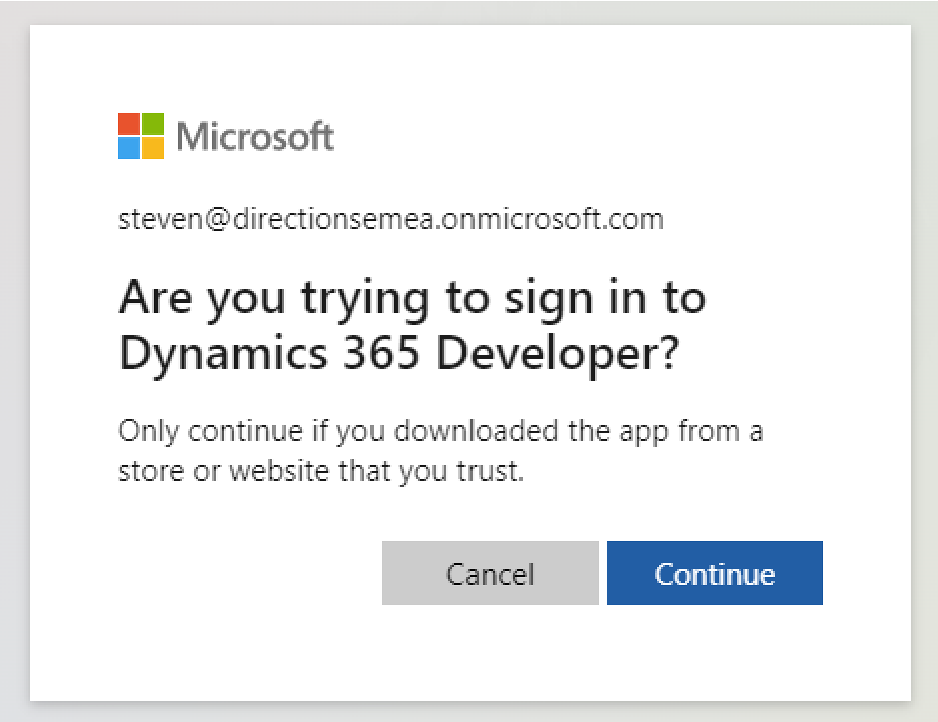
Then click **Next**.

Now authenticate (logon) using the same credentials you use to login to Business Central:

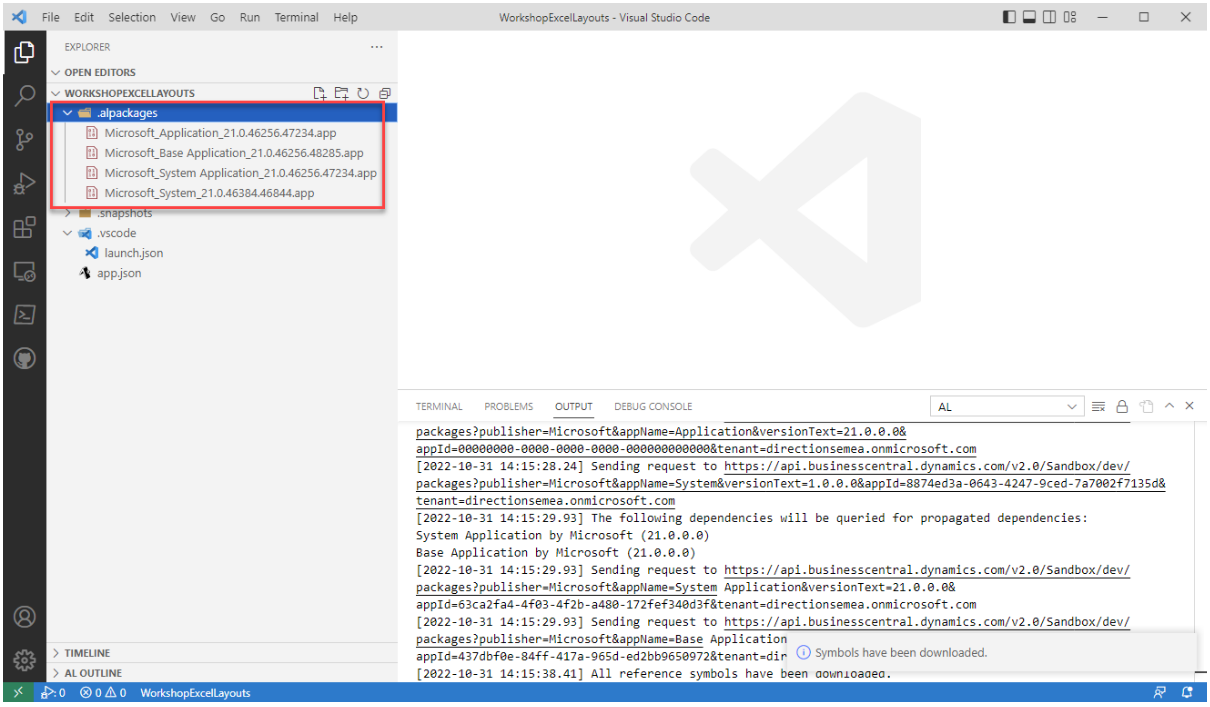


And, when asked, enter your password.

Then click **Continue**:



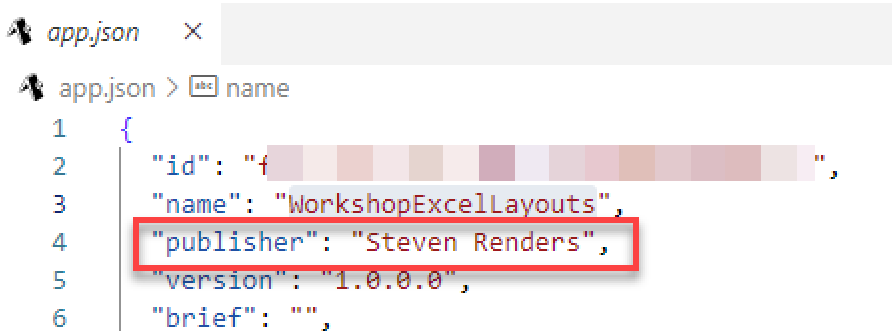
Close the browser window and return to VScode. In VScode, wait until all symbols are downloaded:



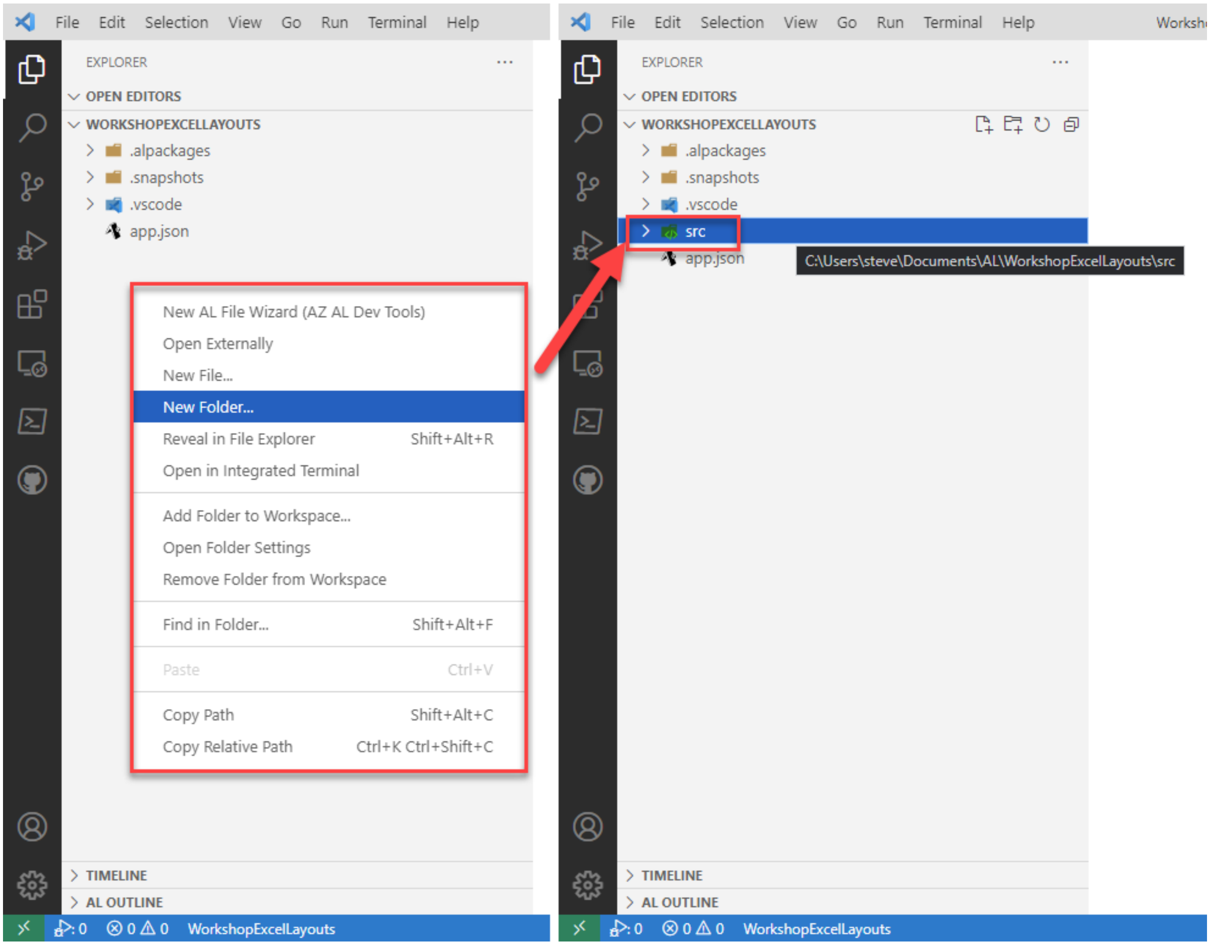
You should now have an **.alpackages** folder in VScode containing **4 apps**, like in the above screenshot.

If this is not the case, ask an instructor for help.

In the **app.json**, in the name setting, you can enter your name as the **publisher**:



In the root folder in VScode, now create an **src** folder:



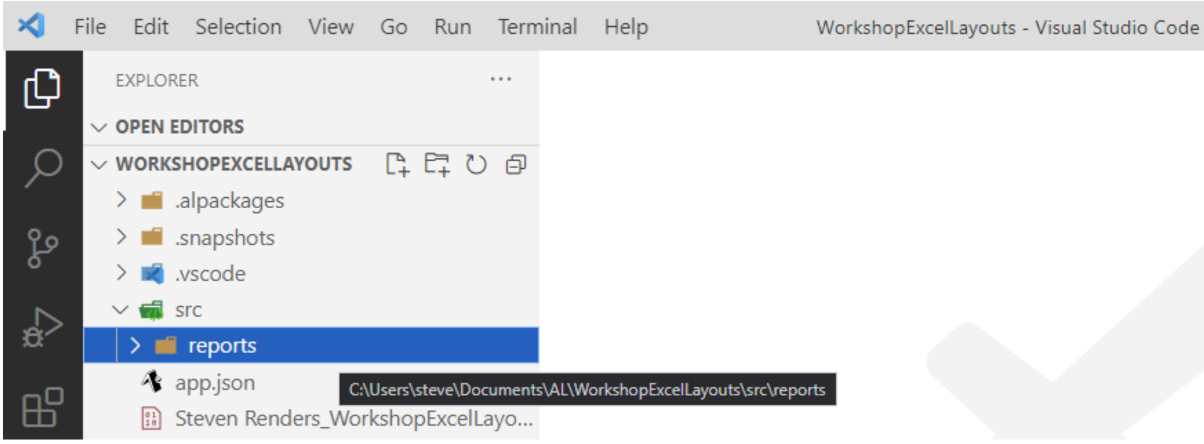
It is recommended to create an src folder, so it can hold the new objects you will develop. In the src folder you can then, later, create subfolders in order to organize your objects.

Your new extension (app), **WorkshopExcelLayouts**, is not created. Currently the app is 'empty' and does not contain any object(s) yet. In the next phase of the workshop, you will create new report objects, each of them with an Excel layout.

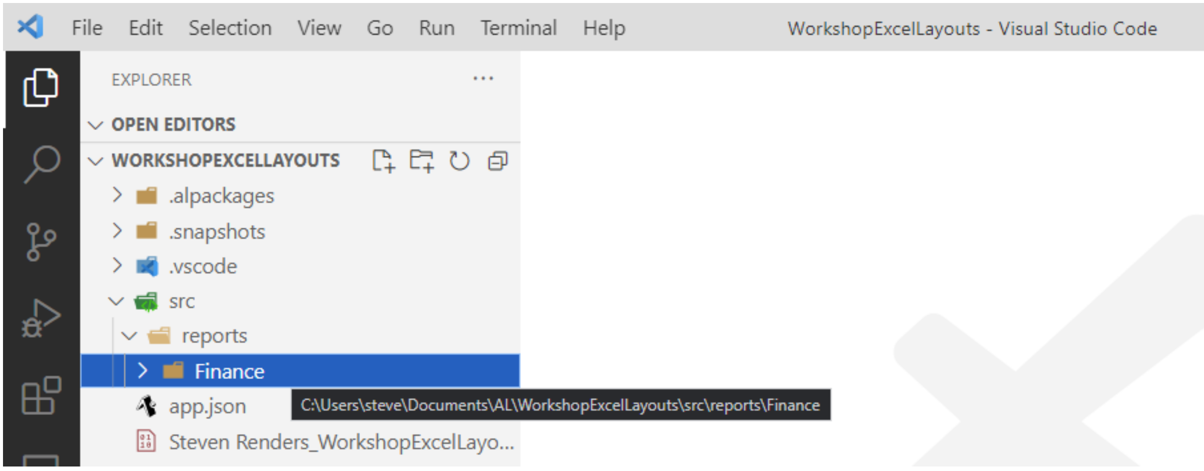
To create the Finance report, follow these steps:

Create the Dataset:

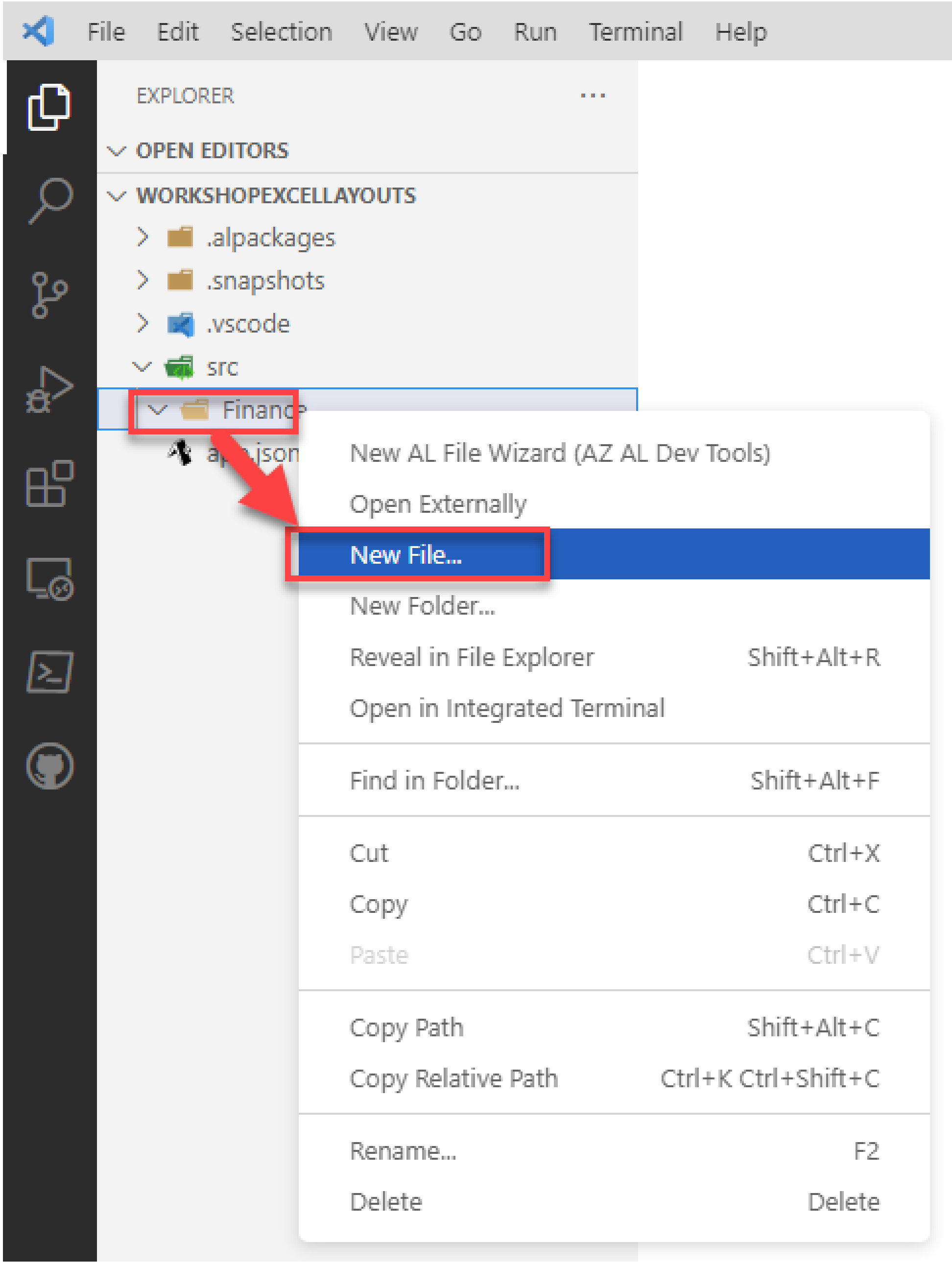
In the **src** folder, create a subfolder named: **reports**:



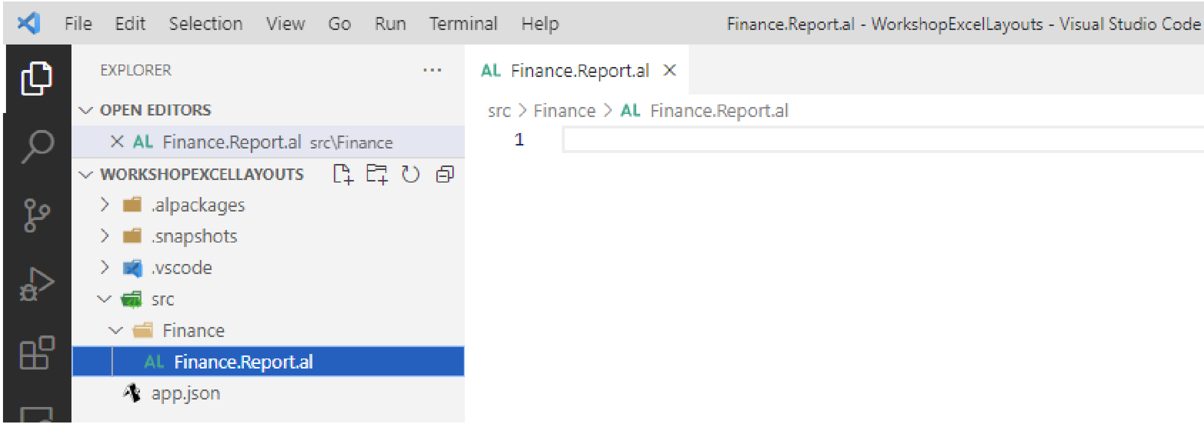
In the reports folder, create a subfolder named **Finance**:



On the **Finance** folder, right click and select **New File**:

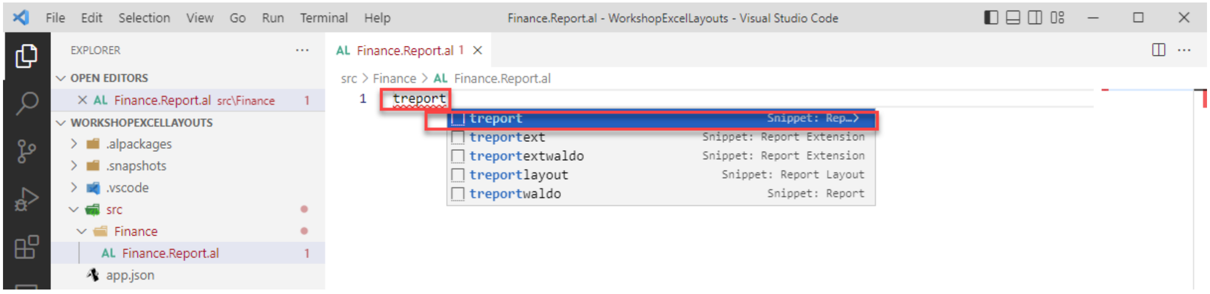


Use this as the name of the new file: **Finance.Report.al**



The file Finance.Report.al will be the container of the new Finance report object.

In the file, type: **treport**, and select treport from the dropdown:



The **treport** snippet created a new report object, where you can now edit the **id**, **name** and other elements.

Provide an id and name for the report, for example 50101 Finance:

```
report 50101 Finance
{
```

Then add the following properties for the report:

```
Caption = 'Finance';
UsageCategory = ReportsAndAnalysis;
ApplicationArea = All;
DefaultRenderingLayout = Finance1;
```

Don't worry at this moment compilation errors might occur, since we are not yet ready with the development of the report. For the time being, ignore those errors.

When you create a new report, there are two main tasks to consider. First, you define the report **dataset** of data items and columns. Then, you design the report **layout**(s).

Change the dataset part of the report, so that it looks as follows:

```
dataset
{
    0 references
    dataitem(GLAccount; "G/L Account")
    {
        0 references
        column(AccountCategory; "Account Category")
        { }
        0 references
        column(AccountSubcategory; "Account Subcategory Descript.")
        { }
        0 references
        column(AccountType; "Account Type")
        { }
        0 references
        column(AccountNo; "No.")
        { }
        0 references
        column(AccountName; Name)
        { }
        0 references
        column(AccountNameNo; Name + ' (' + "No." + ')')
        { }
        0 references
        dataitem(GLEntry; "G/L Entry")
        {
            DataItemLink = "G/L Account No." = field("No.");
            DataItemTableView = sorting("G/L Account No.");
            0 references
            column(Amount; Amount)
            { }
            0 references
            column(Year; System.Date2DMY("Posting Date", 3))
            { }
            0 references
            column(Month; System.Date2DMY("Posting Date", 2))
            { }
            0 references
            column(PostingDate; "Posting Date")
            {}
        }
    }
}
```

This means we will add 2 data items (tables) to the report: **G/L Account** and **G/L Entry**.

From those 2 data items (tables), we add the following fields:

- G/L Account:
    - Account Category
    - Account Subcategory Descript.
    - Account Type
    - No.
    - Name
    - Name + ' (' + "No." + ')'
- G/L Entry:
    - Amount
    - System.Date2DMY("Posting Date", 3)
    - System.Date2DMY("Posting Date", 2)
    - Posting Date

In a report column, you can select a field from the table, and perform calculations on the field. For example:

The column:

```
column(AccountNameNo; Name + ' (' + "No." + ')')
{ }
```

has the name **AccountNameNo** and contains the concatenation of the GL Account Name and No.

The column:

```
column(Year; System.Date2DMY("Posting Date", 3))
{ }
```

has the name **Year**, and the formula System.Date2DMY("Posting Date", 3) gets the year out of the Posting Date.

The column:

```
column(Month; System.Date2DMY("Posting Date", 2))
{ }
```

has the name **Month**, and the formula System.Date2DMY("Posting Date", 3) gets the month number out of the Posting Date.

When creating a dataset for a report that has an Excel layout, you need to consider which fields you need in Excel. The G/L Entry table contains a Posting Data field, but in Excel we want to be able to create a pivot table based on the years and months. So we can add those as columns in the dataset.

An alternative is to leave the Posting Date as a normal date in the dataset, and use **Power Query** in Excel to create columns for: Year, Month,… based on the Posting Date.

The 2 tables G/L Account and G/L Entry need to be **linked** correctly, that's why we have the following properties:

```
dataitem(GLEntry; "G/L Entry")
{
    DataItemLink = "G/L Account No." = field("No.");
    DataItemTableView = sorting("G/L Account No.");
    0 references
```

This means that when the report runs and the runtime dataset is generated, the **Ledger Entries** are linked correctly to the **Accounts**. The **DataItemLink** property of the indented data item is responsible for that: DataItemLink Property - Business Central | Microsoft Learn

The **DataItemTableview** is used to **sort** the G/L Entry data item, and that has the consequence that this data item will not be added to the **request page** of the report: DataItemTableView Property - Business Central | Microsoft Learn By default, for every data item in the report dataset, a tab is added to the runtime request page of the report, so a user can **filter** the data items. However, if you add a sorting to a data item, it is not added to the request page.

More information about defining a report dataset is available here: Report Dataset - Business Central | Microsoft Learn

In the **rendering** section of the report, create the following:

```
rendering
{
    1 reference
    layout(Finance1)
    {
        Type = Excel;
        LayoutFile = './src/reports/Finance/Finance1.xlsx';
        Summary = 'Finance layout, pivot.';
    }
}
```

This means we want to add one Excel layout, named **Finance1**.

More information about defining multiple report layouts is available here: Defining Multiple Report Layouts - Business Central | Microsoft Learn

Now delete the **requestpage** section from the report:

Your report should now look as follows:

```
report 50101 Finance
{
    Caption = 'Finance';
    UsageCategory = ReportsAndAnalysis;
    ApplicationArea = All;
    DefaultRenderingLayout = Finance1;

    dataset
    {
```

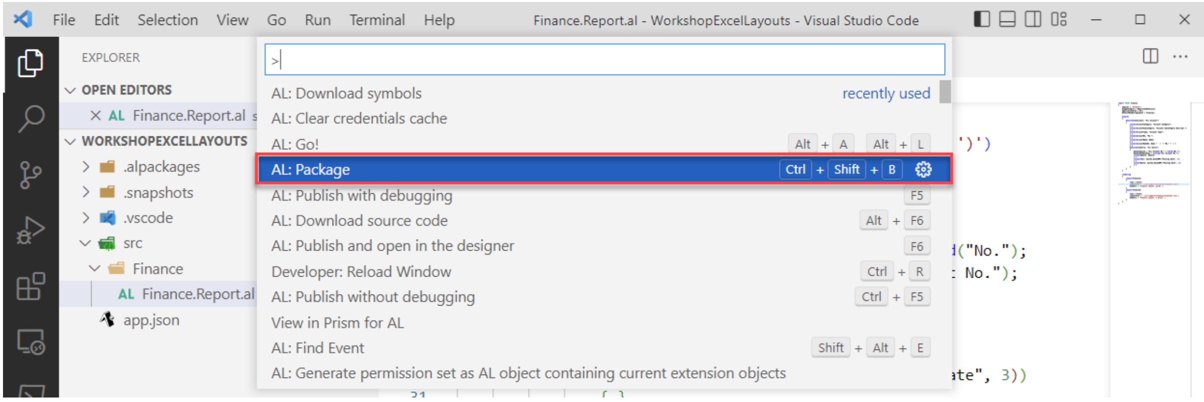
```
        dataitem(GLAccount; "G/L Account")
        {
            column(AccountCategory; "Account Category")
            { }
            column(AccountSubcategory; "Account Subcategory Descript.")
            { }
            column(AccountType; "Account Type")
            { }
            column(AccountNo; "No.")
            { }
            column(AccountName; Name)
            { }
            column(AccountNameNo; Name + ' (' + "No." + ')')
            { }
            dataitem(GLEntry; "G/L Entry")
            {
                DataItemLink = "G/L Account No." = field("No.");
                DataItemTableView = sorting("G/L Account No.");
                column(Amount; Amount)
                { }
                column(Year; System.Date2DMY("Posting Date", 3))
                { }
                column(Month; System.Date2DMY("Posting Date", 2))
                { }
                column(PostingDate; "Posting Date")
                {}
            }
        }
    }

    rendering
    {
        layout(Finance1)
        {
            Type = Excel;
            LayoutFile = './src/reports/Finance/Finance1.xlsx';
            Summary = 'Finance layout, pivot.';
        }
    }
}
```

If you are unsure, copy/paste the code from this document to **replace** the code currently in your report file.

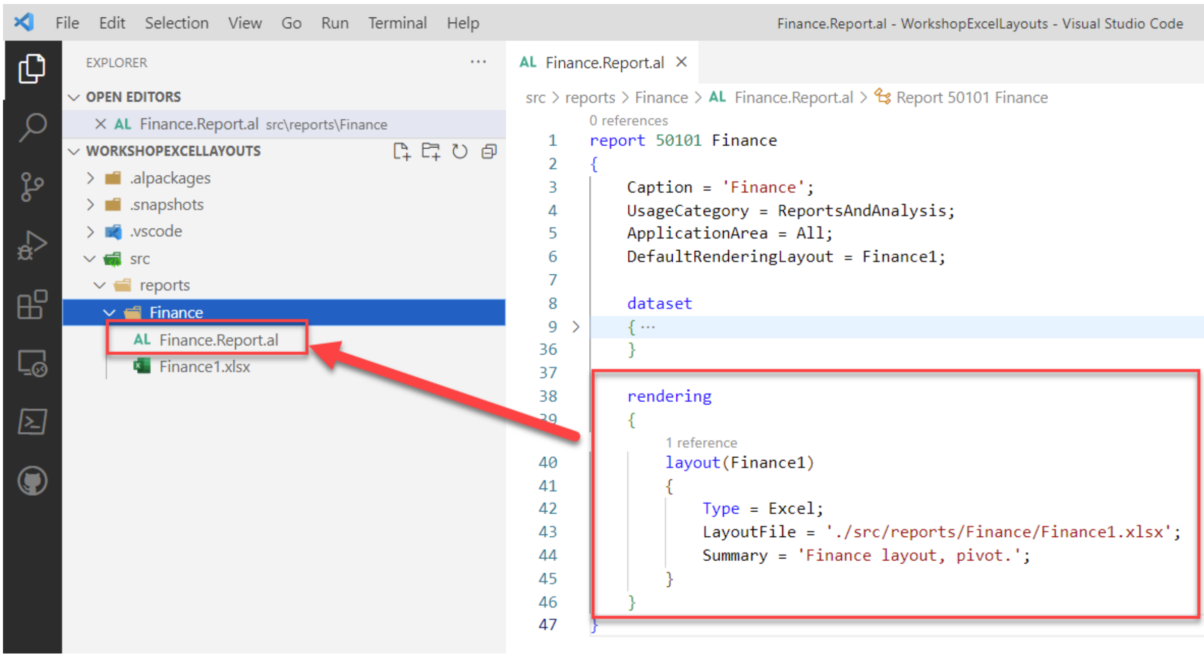
We have defined the report to have 2 data items and 1 layout. Now we will create the, empty, Excel layout.

Create the Excel layout:

To create the, empty, Excel layout you need to **Package** the report object. To package the report object, select **AL: Package** from the command palette, or simply use the shortcut: **Ctrl + Shft + B**:



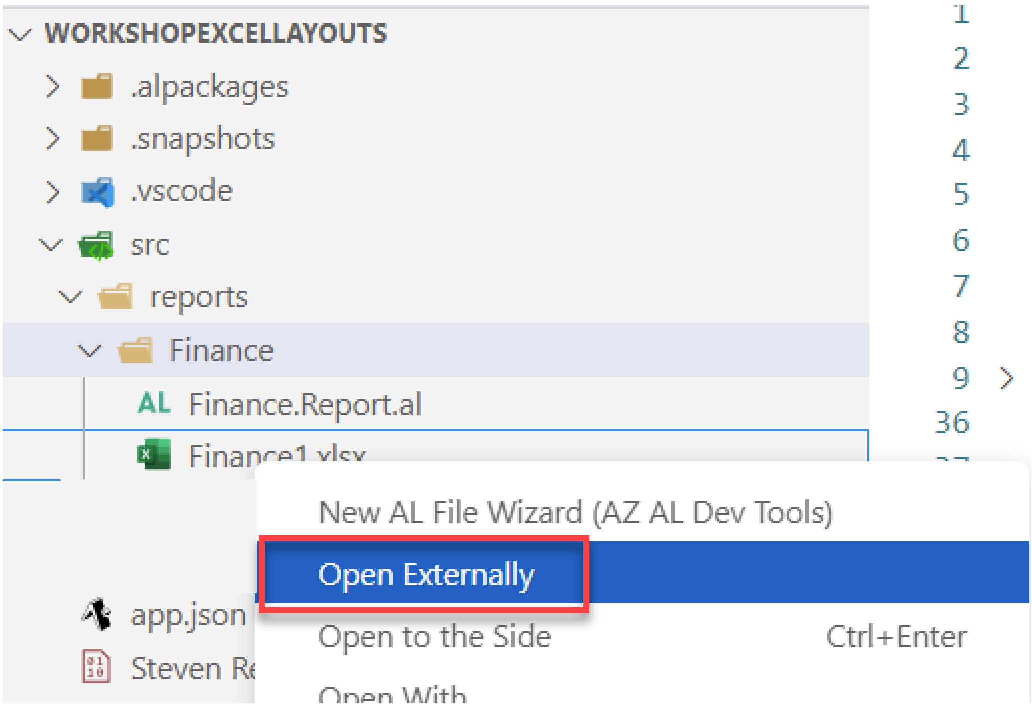
The **Excel** layout is added to the **Finance** folder:



Be aware that before you start to edit the Excel layout, you need to be 100% certain your dataset is ready. In case you need to make changes later to your report's dataset, you will have to delete and recreate the Excel layouts. Currently it is not possible to automatically update the Excel layout(s) with dataset changes. It is however possible to manually add a column to the Excel layout, but this is very error prone.

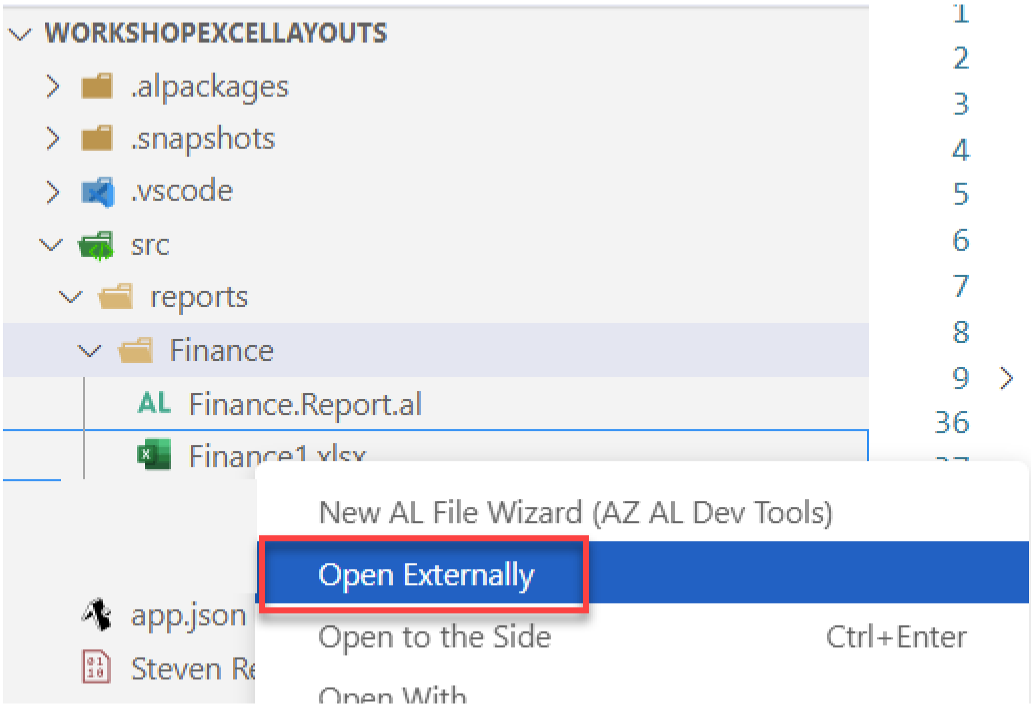
To **edit** the Excel layout, you can open them.

To open a layout, **right click** on it in VSCode and select **Open Externally**:

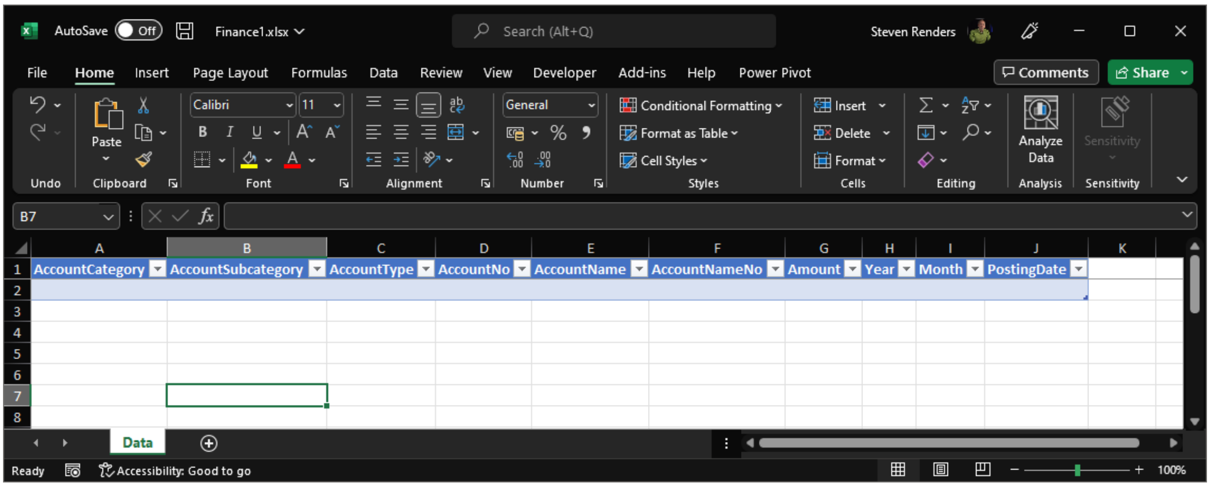


This of course requires you to have **Excel** installed on your machine.

Now open the **Finance1** excel layout, via the command (right click) open externally:



The Excel file looks as follows:

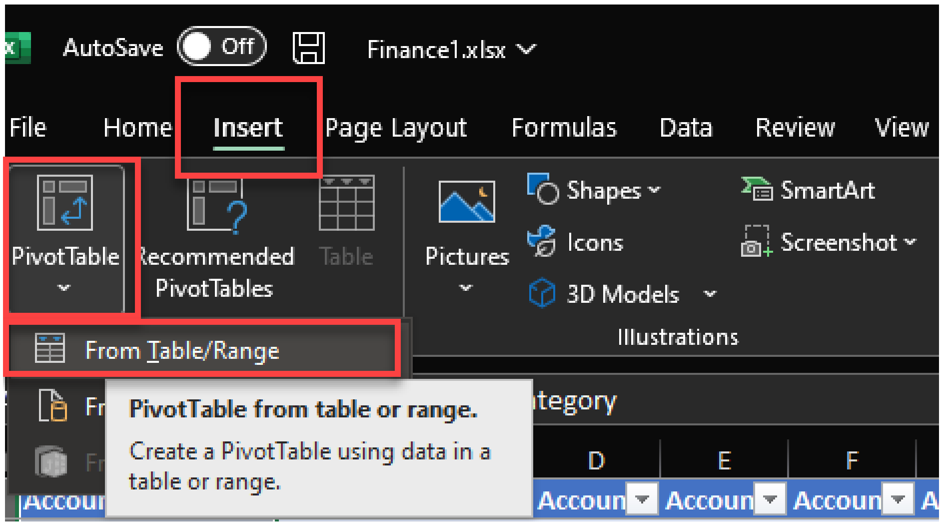


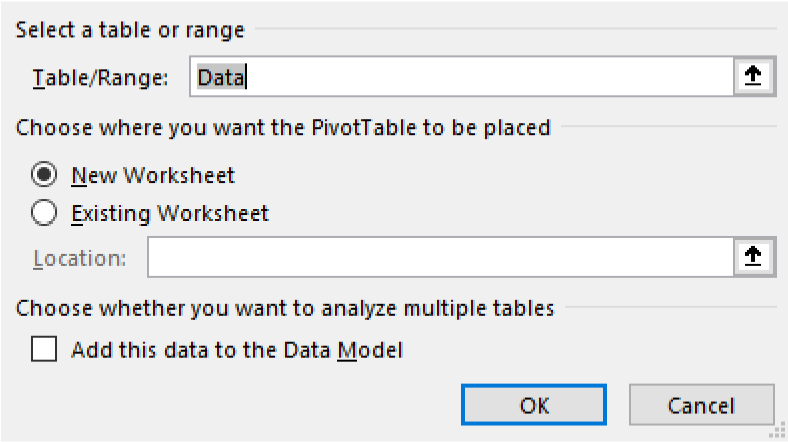
It's important you don't make any changes to the Data worksheet. Do not rename it, do not remove or rename any columns, and also do not add any columns.

The **Data** worksheet acts as the dataset of the report, and at runtime will be populated with data when you run the report. It acts as a contract with the .al file and should not be changed.

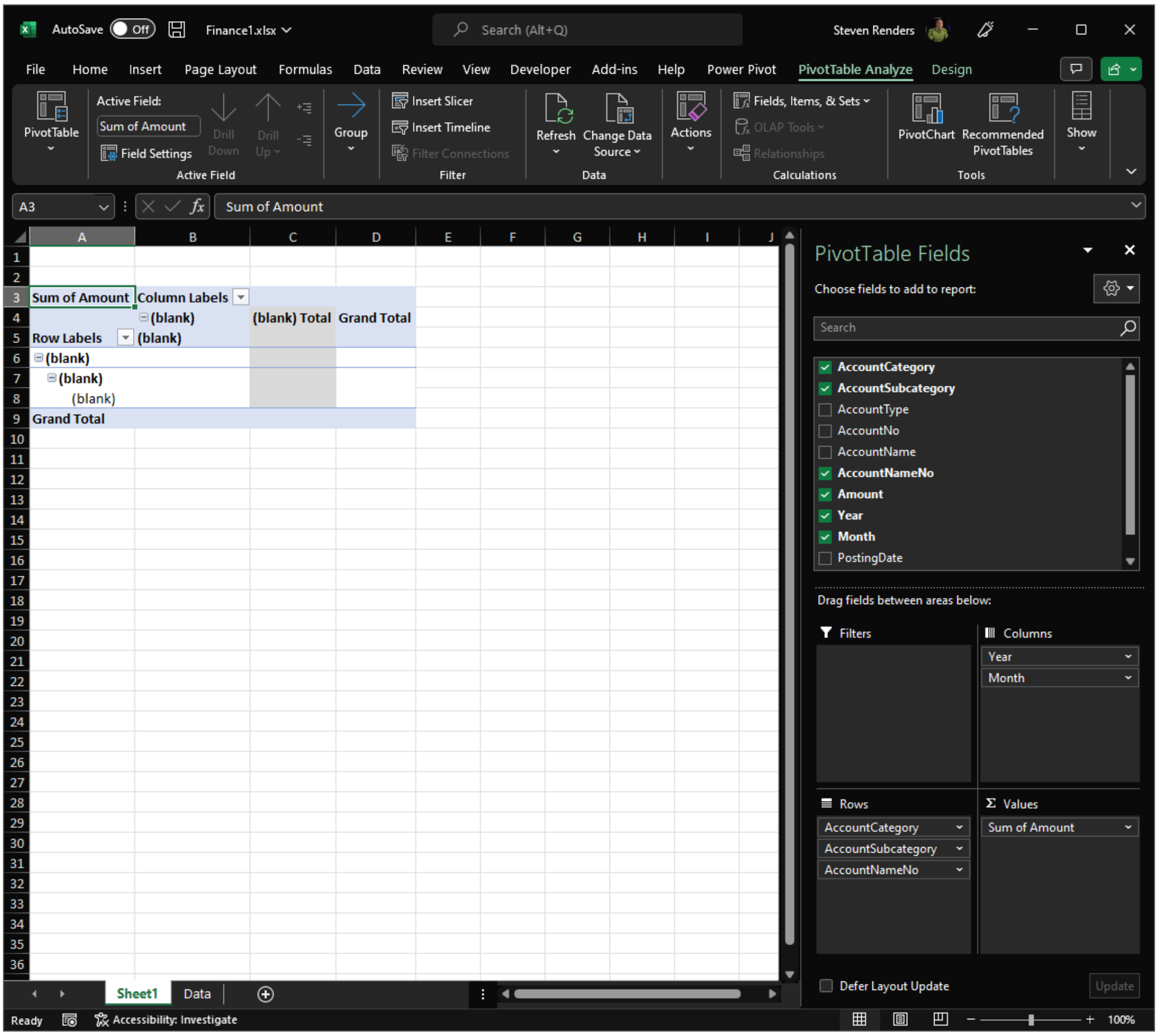
However, you can add new worksheets, and in the new worksheets for example create pivot tables.

Now add a **pivot table** to the Excel sheet:

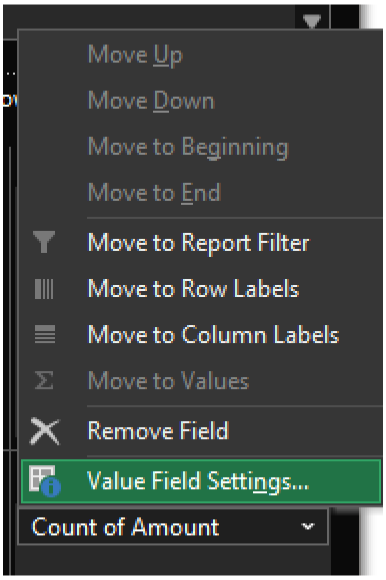




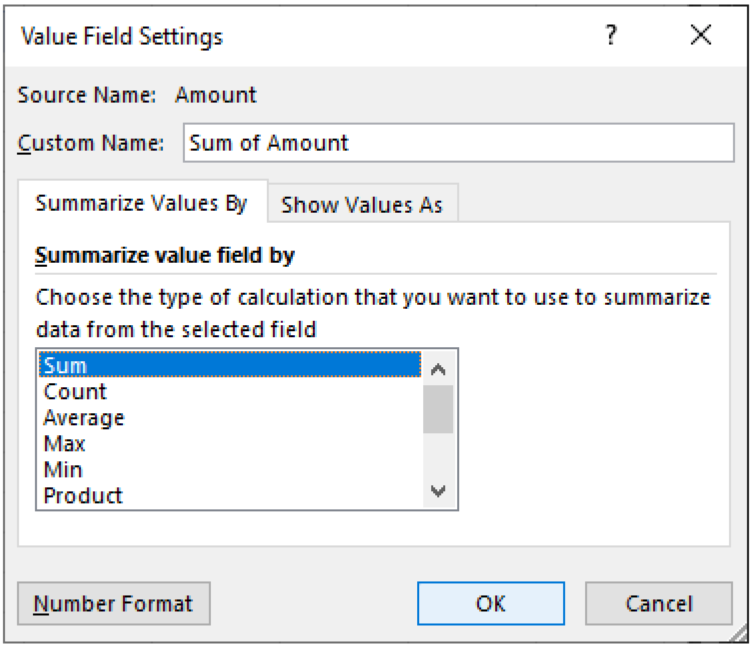
with the following properties:



By default, excel might propose the **Count** instead of the **Sum** method for the **Amount** field. To change that, click on the **Amount** field and select **Value Field Settings**:



Then in the popup select **Sum**:



You have now added a pivot table to the report, using the columns from the dataset, generated in Business Central.

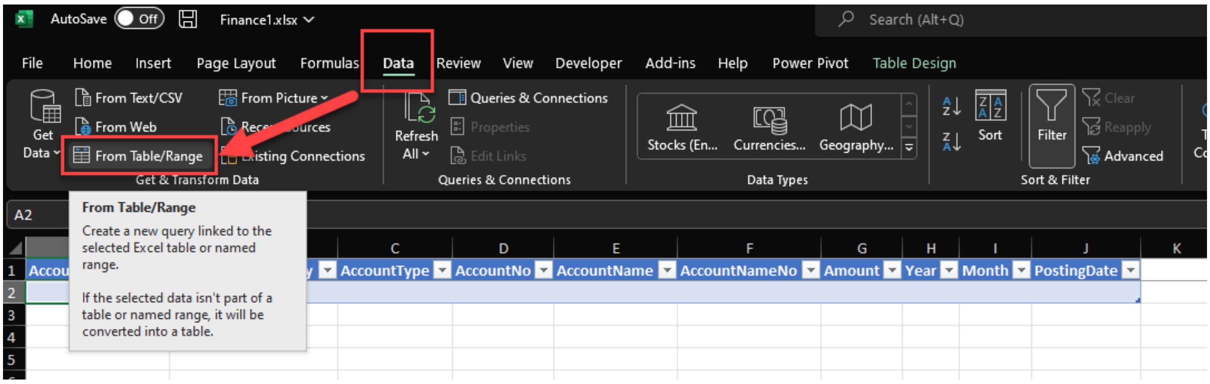
Now **close** and **save** the Finance1.xlsx file.

Let's now add a column, in Power Query.

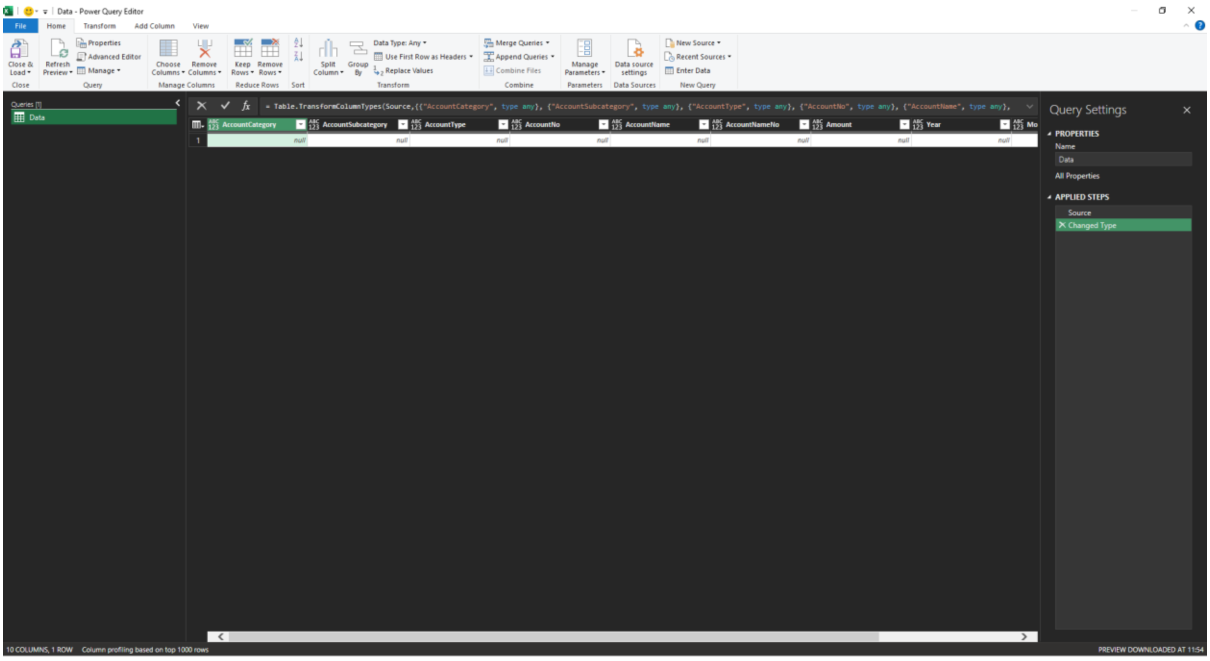
Reopen the **Finance1.xlsx** file.

Go back to the **Data** tab in the Excel layout.

In the menu, in **Data** tab in Excel, select **From Table/Range**:



Power Query opens:

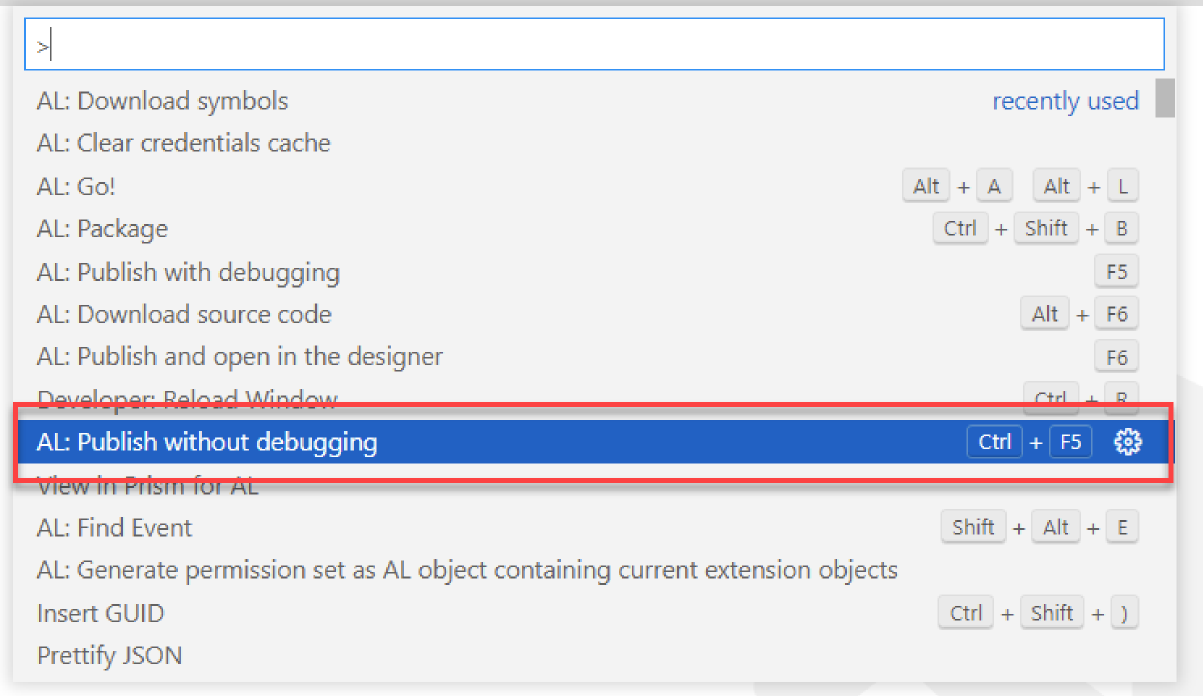


However, since there’s no data in the Excel file, Power Query does not recognize the data types. So, in order to work with Power Query in Excel, it’s better to have an Excel layout with data.

For that reason, it’s time now to deploy our app to Business Central, so then we can run the report, and generate the Excel file, with data.

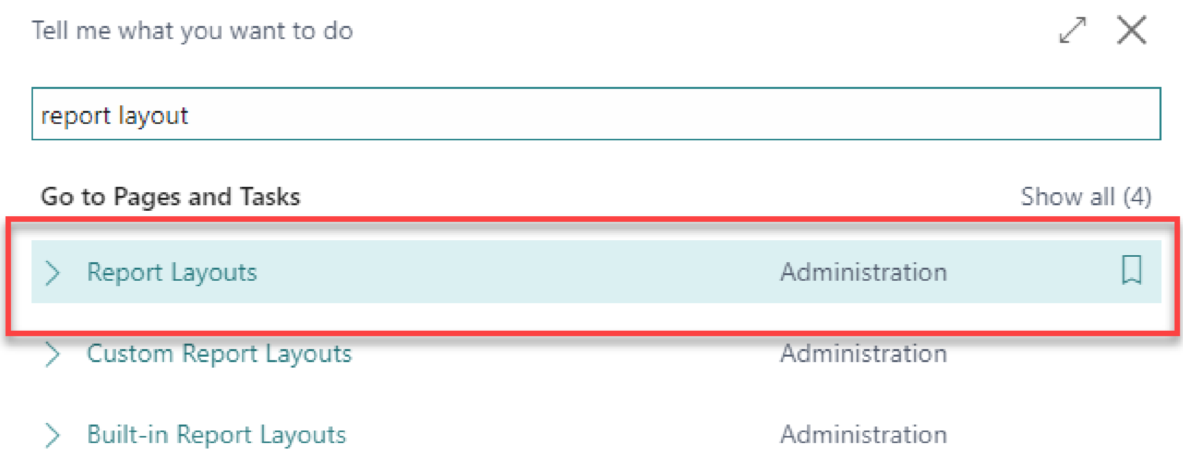
Close the Power Query editor and then close the Excel file, without saving.

In Visual Studio Code, select **Publish without debugging**, in the Command Palette:

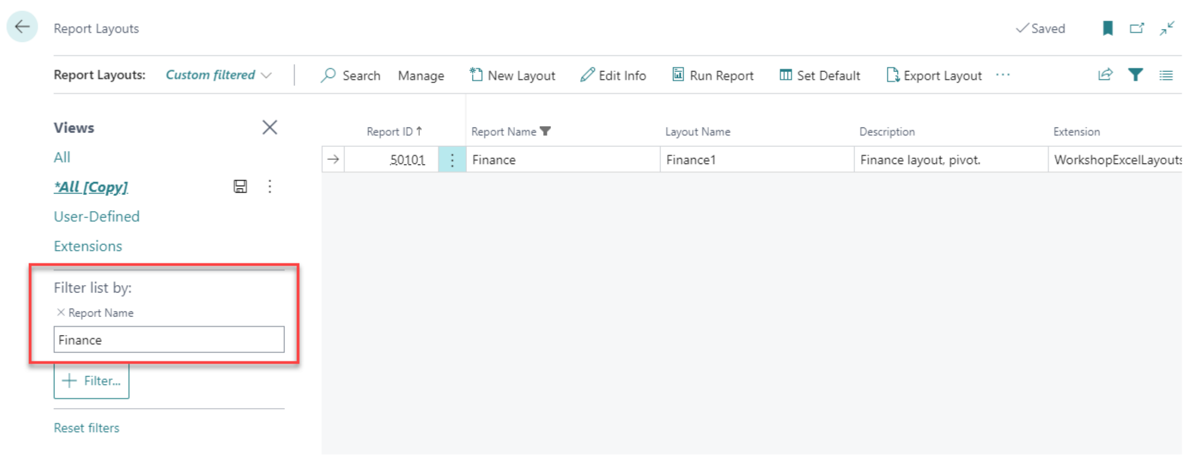


Your app is now published to your sandbox, after which your browser opens and Business Central.

In Business Central, open the **Report Layouts** page:

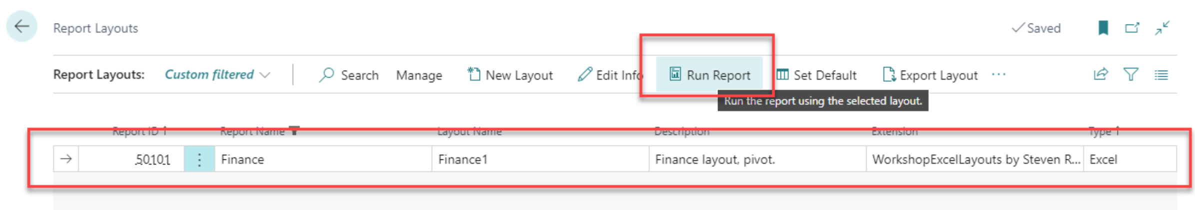


In this page, apply a **filter**:

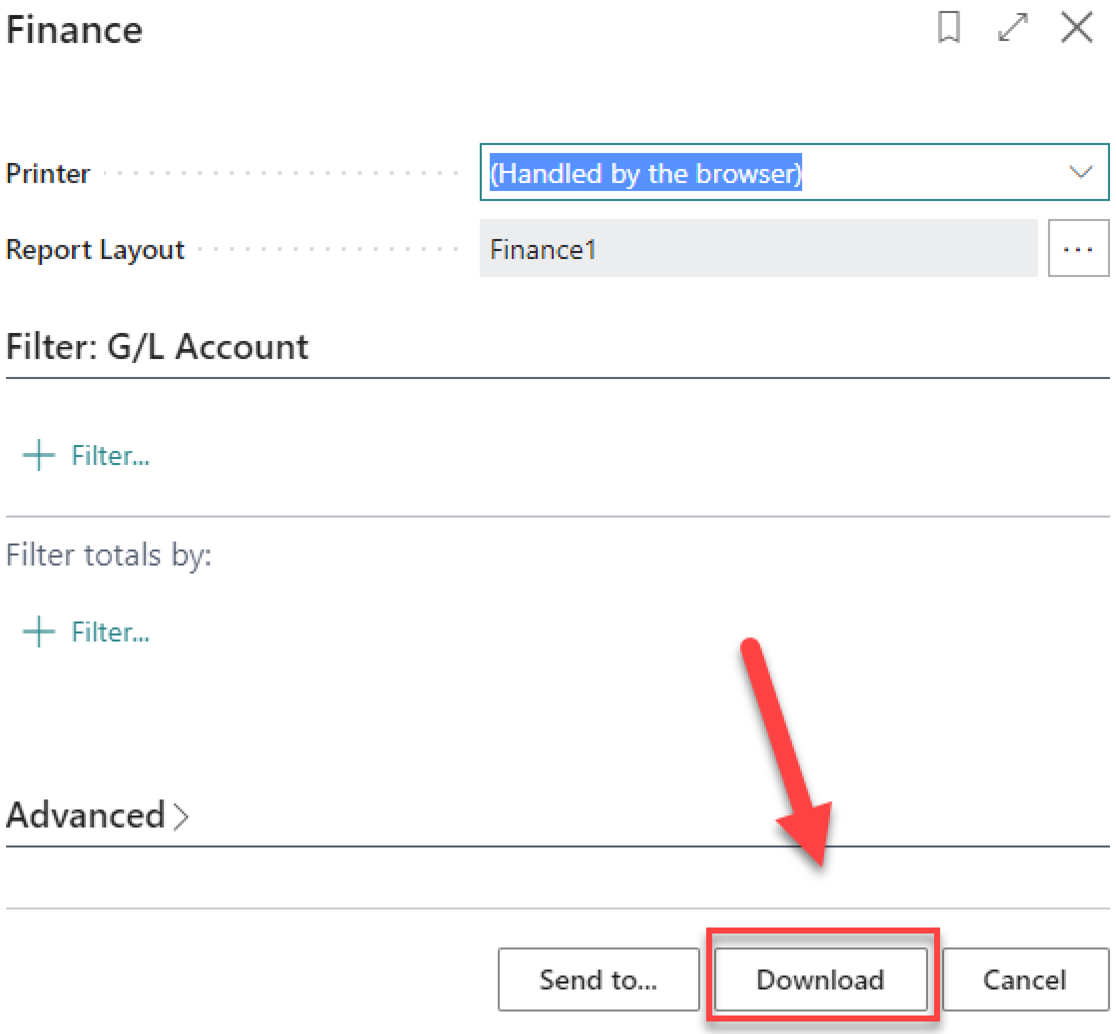


Now you can see your Finance report layout in Business Central.

To run it, select the layout and then **Run Report**:



The request page opens:



Then click **Download** to generate and download the excel file.

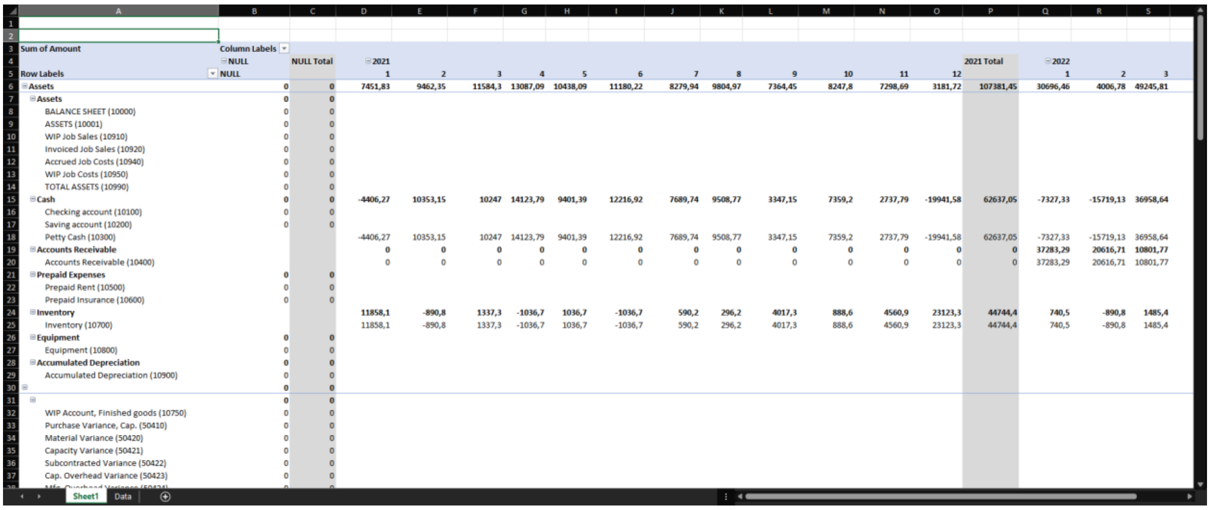
Open the Excel file, to look at the data.

(You might have to click the Enable Data button, when asked.)

There should now be data in the **Data** tab:

AccountCategory	AccountSubcategory	AccountType	AccountNo	AccountName	AccountNameNo	Amount	Year	Month	PostingDate
Assets	Assets	Heading	10000	BALANCE SHEET	BALANCE SHEET (100	NULL	NULL	NULL	NULL
Assets	Assets	Begin-Total	10001	ASSETS	ASSETS (10001)	NULL	NULL	NULL	NULL
Assets	Cash	Posting	10100	Checking account	Checking account (1(	NULL	NULL	NULL	NULL
Assets	Cash	Posting	10200	Saving account	Saving account (1020	NULL	NULL	NULL	NULL
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-3.310,17	2021	1	1/01/2021
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-1.792,46	2021	1	1/01/2021
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-470,96	2021	1	1/01/2021
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-362,94	2021	1	1/01/2021
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-1.593,18	2021	1	1/01/2021
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-318,64	2021	1	1/01/2021
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-637,27	2021	1	1/01/2021
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-1.569,86	2021	1	1/01/2021
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-296,20	2021	1	1/01/2021
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-444,30	2021	1	1/01/2021
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-740,50	2021	1	1/01/2021
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-1.581,52	2021	1	10/01/2021
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-2.678,62	2021	1	10/01/2021
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-1.098,90	2021	1	11/01/2021
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-941,92	2021	1	12/01/2021
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-1.902,49	2021	1	13/01/2021
Assets	Cash	Posting	10300	Petty Cash	Petty Cash (10300)	-1.274,54	2021	1	13/01/2021

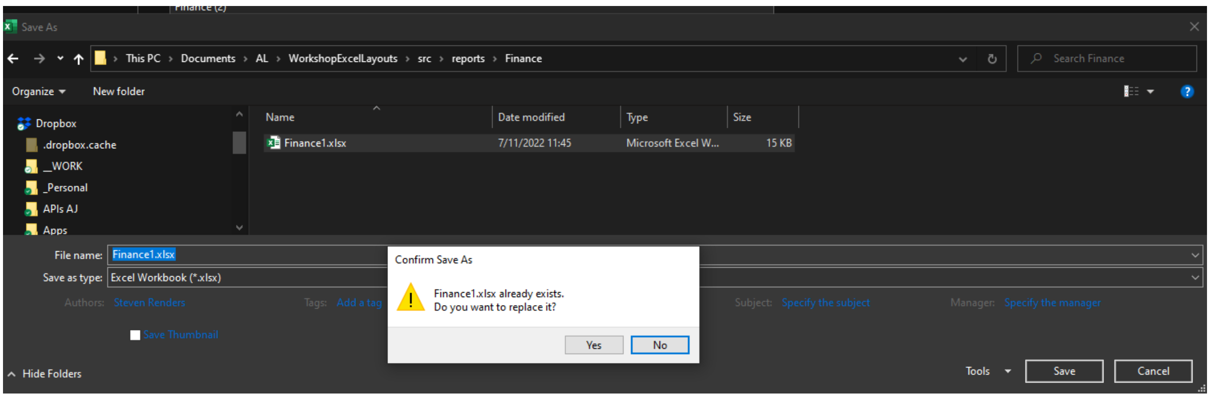
And the pivot table also has data:



Now save the Excel file, in the layout directory of the extension in VSCode:

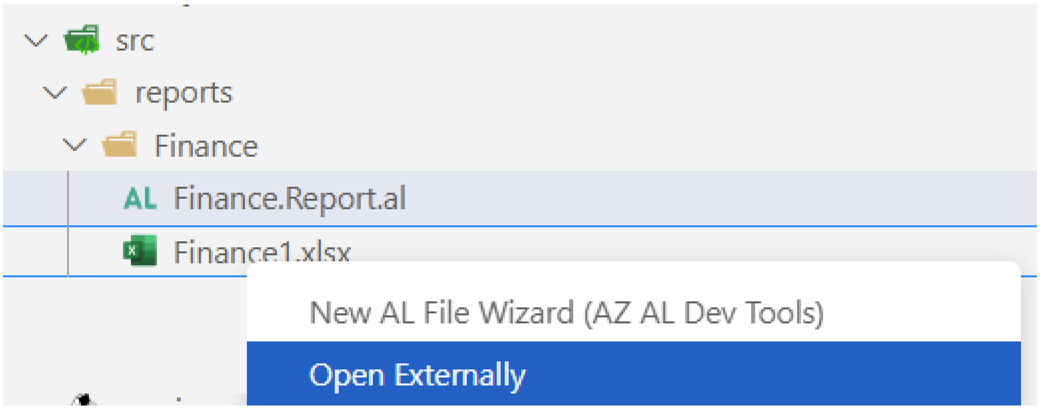
..\WorkshopExcelLayouts\src\reports\Finance

This will overwrite the currently existing **Finance1.xls** layout in that folder:



Now **close** and **save** the Finance1.xlsx file.

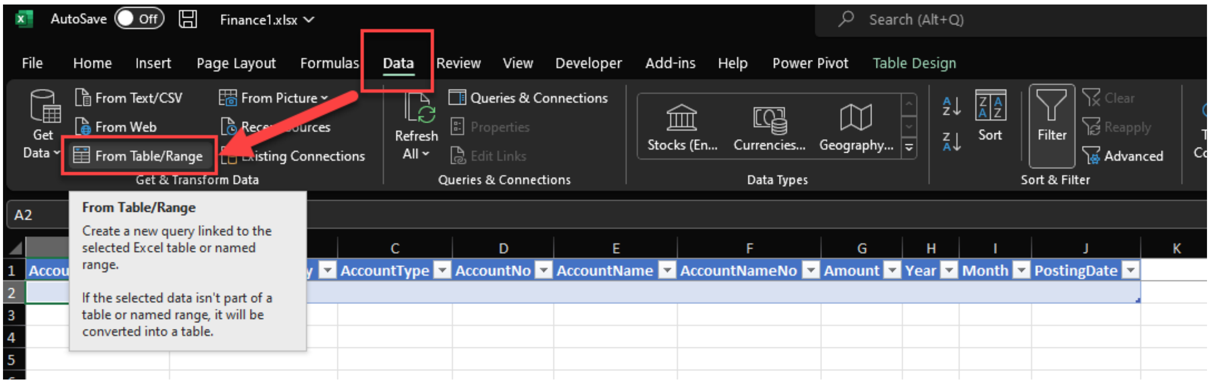
Go back to VSCode, right click on the Finance1.xlsx file in VSCode and select Open Externally:



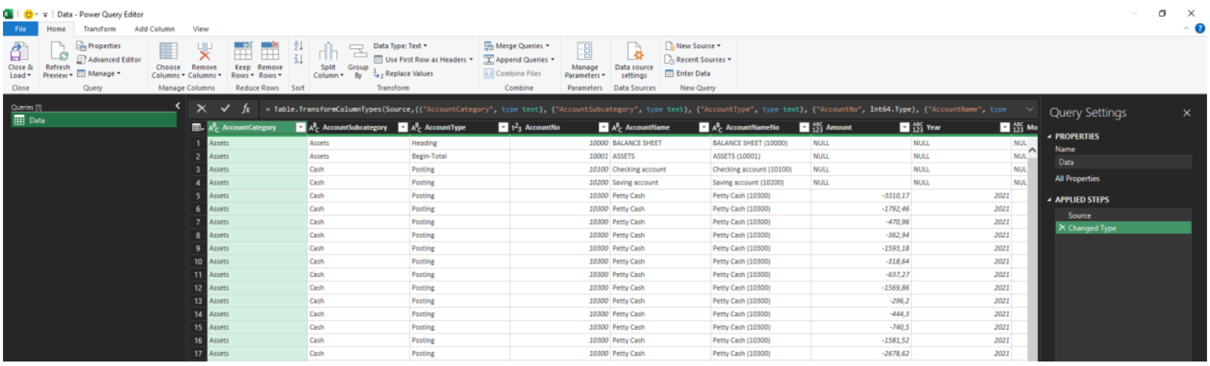
The Finance1.xlsx layout opens and has data in it.

Go back to the **Data** tab in the Excel layout.

In the menu, in **Data** tab in Excel, select **From Table/Range**:

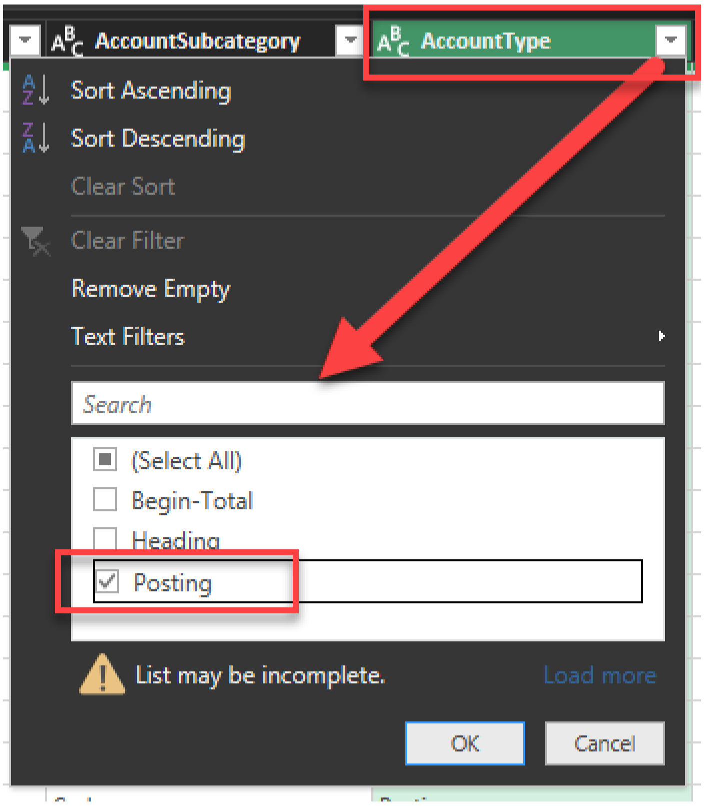


Power Query opens:

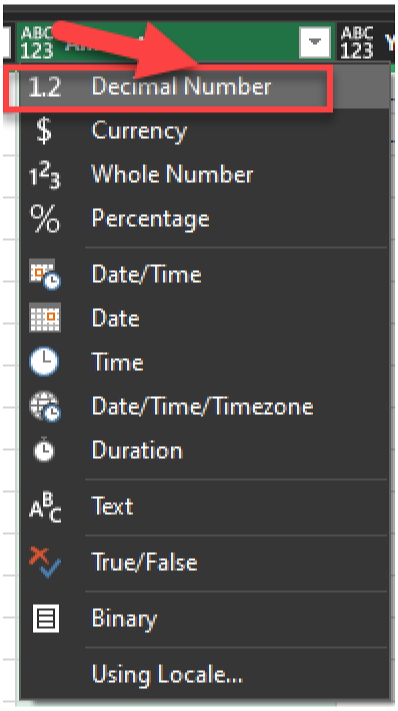


And this time, there's data in it, and column data types are recognized in Power Query.

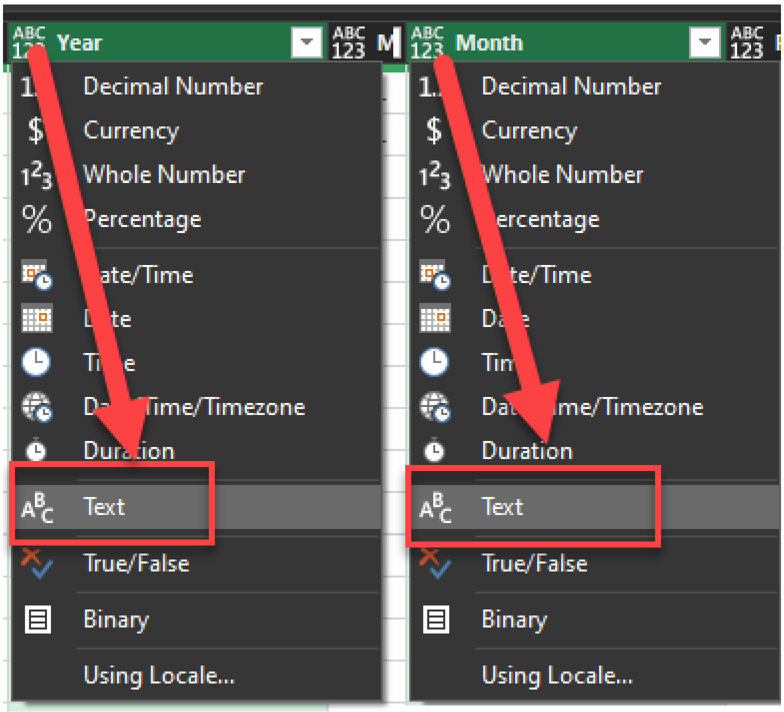
Let's first add a filter on the data, to only see Posting accounts:



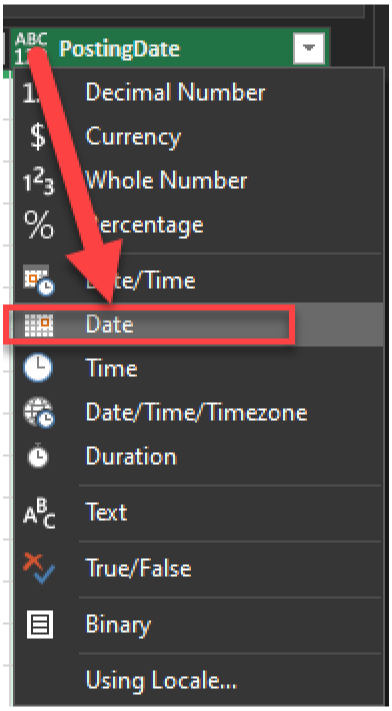
Now set the **Amount** as a **Decimal** datatype:



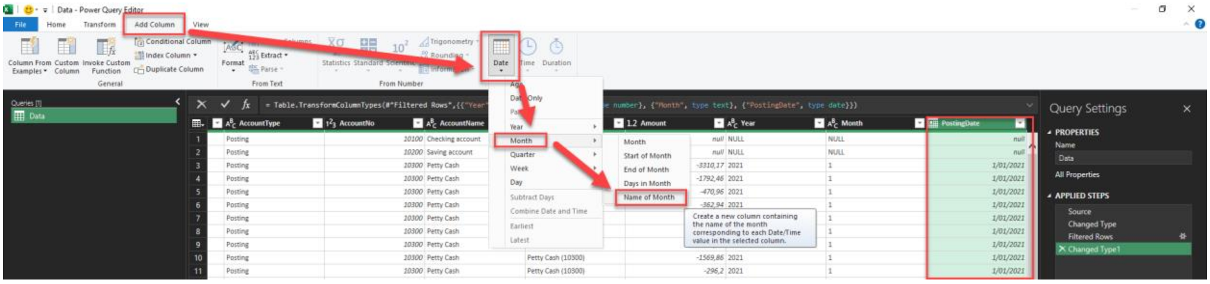
And the **Year** and **Month** as Text:



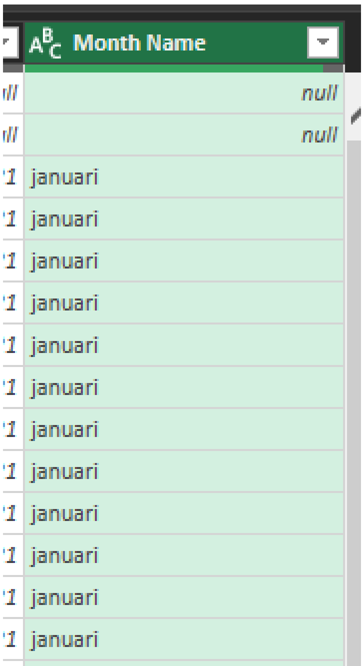
Then set the **Posting Date** as a **Date** datatype:



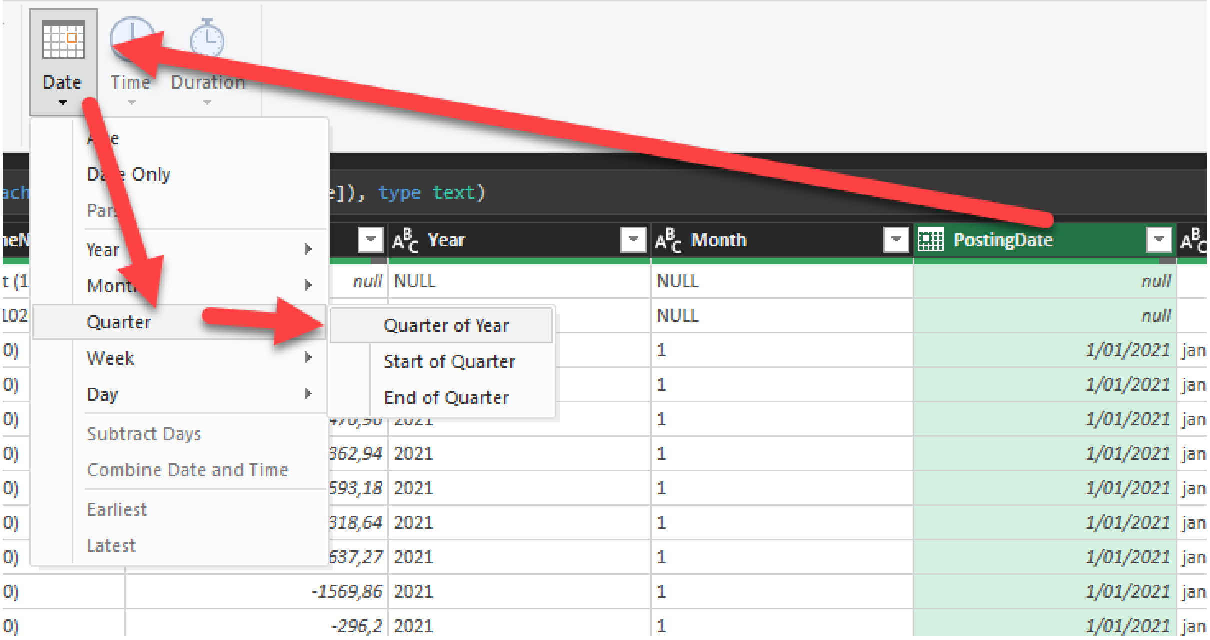
Select the **Posting Date** column, then, click **Add Column**, **Date**, **Month**, **Name of Month**:

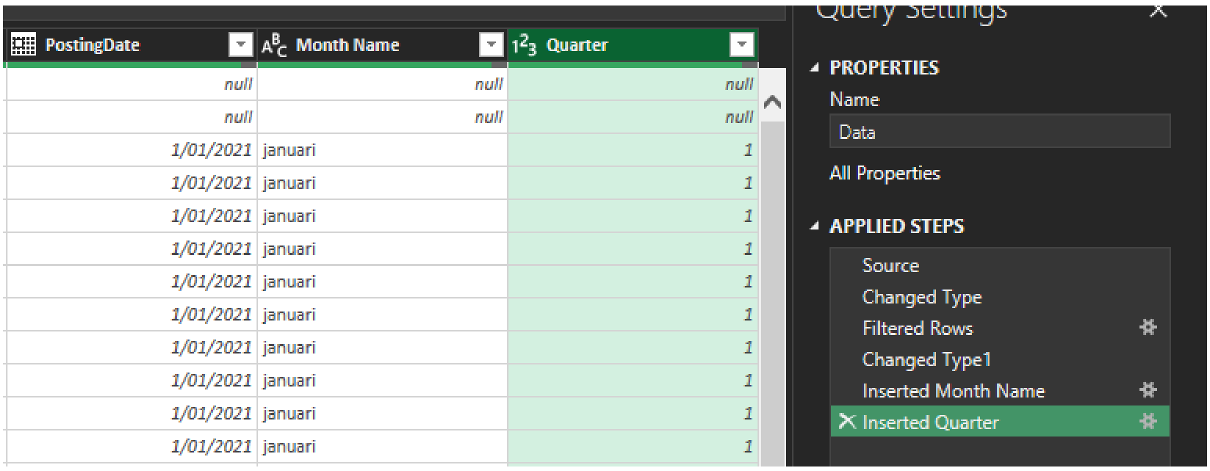


And the MonthName is added as a new column:



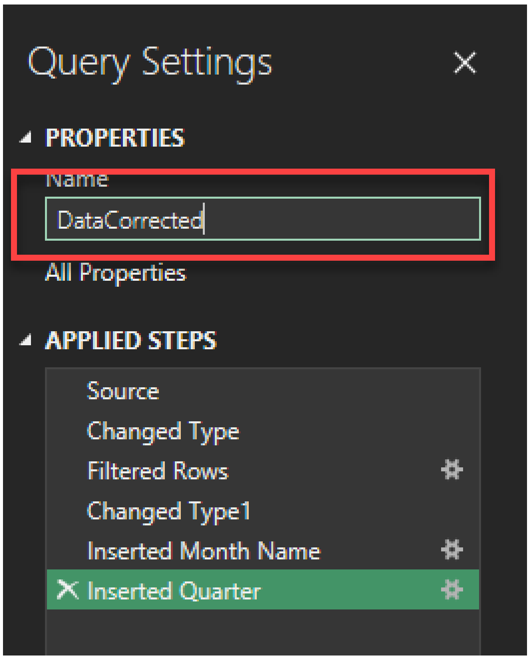
Similarly, we can now add:



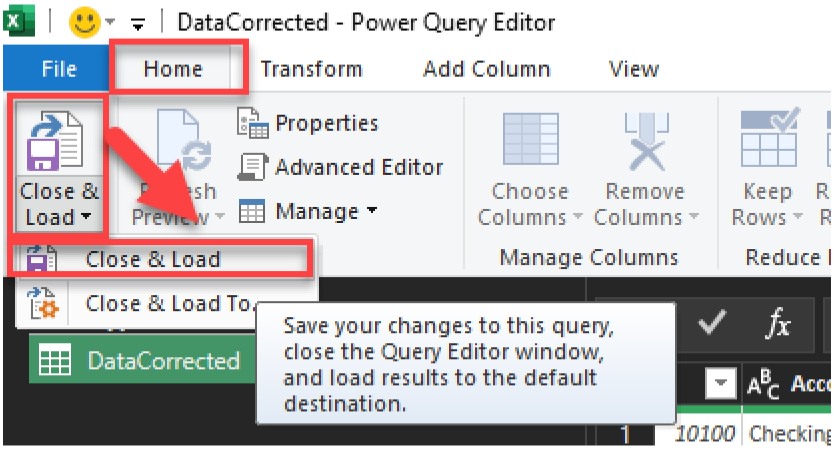


At the right side in Power Query, you can see the changes you have made to the table.

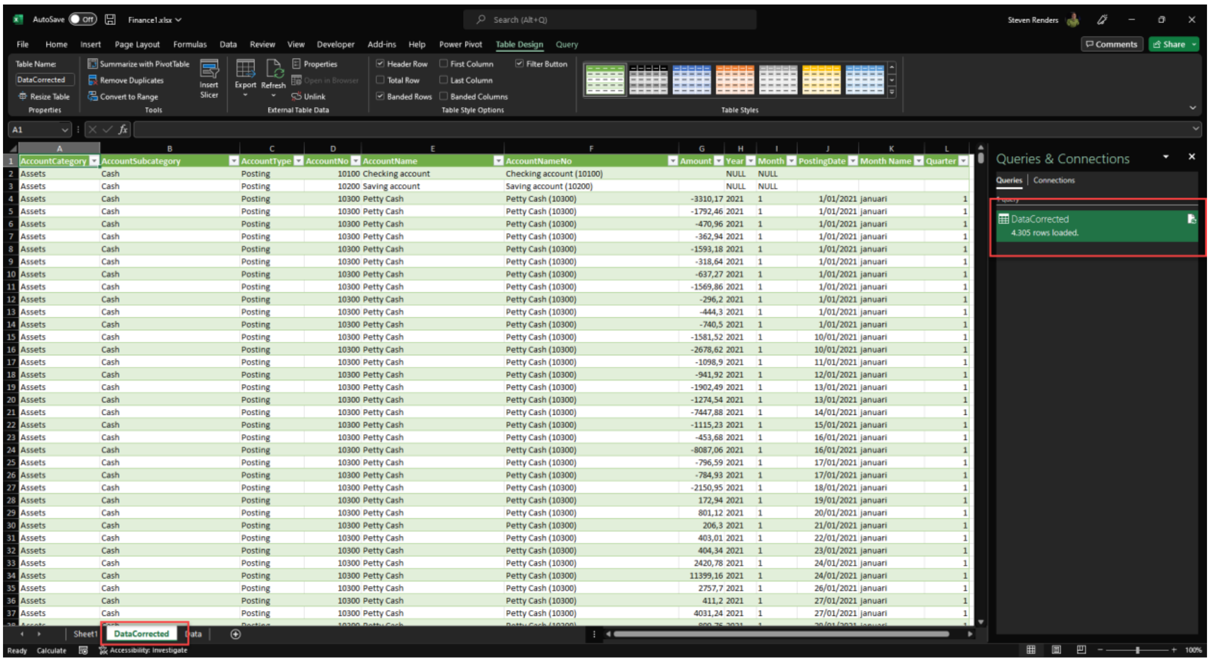
You can change the name of the query to **DataCorrected**:



Then in the Home menu select **Close and Load**:

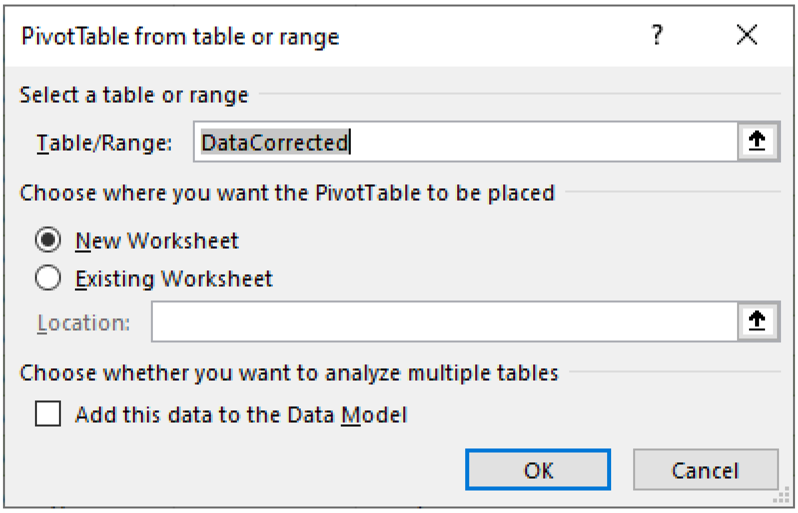


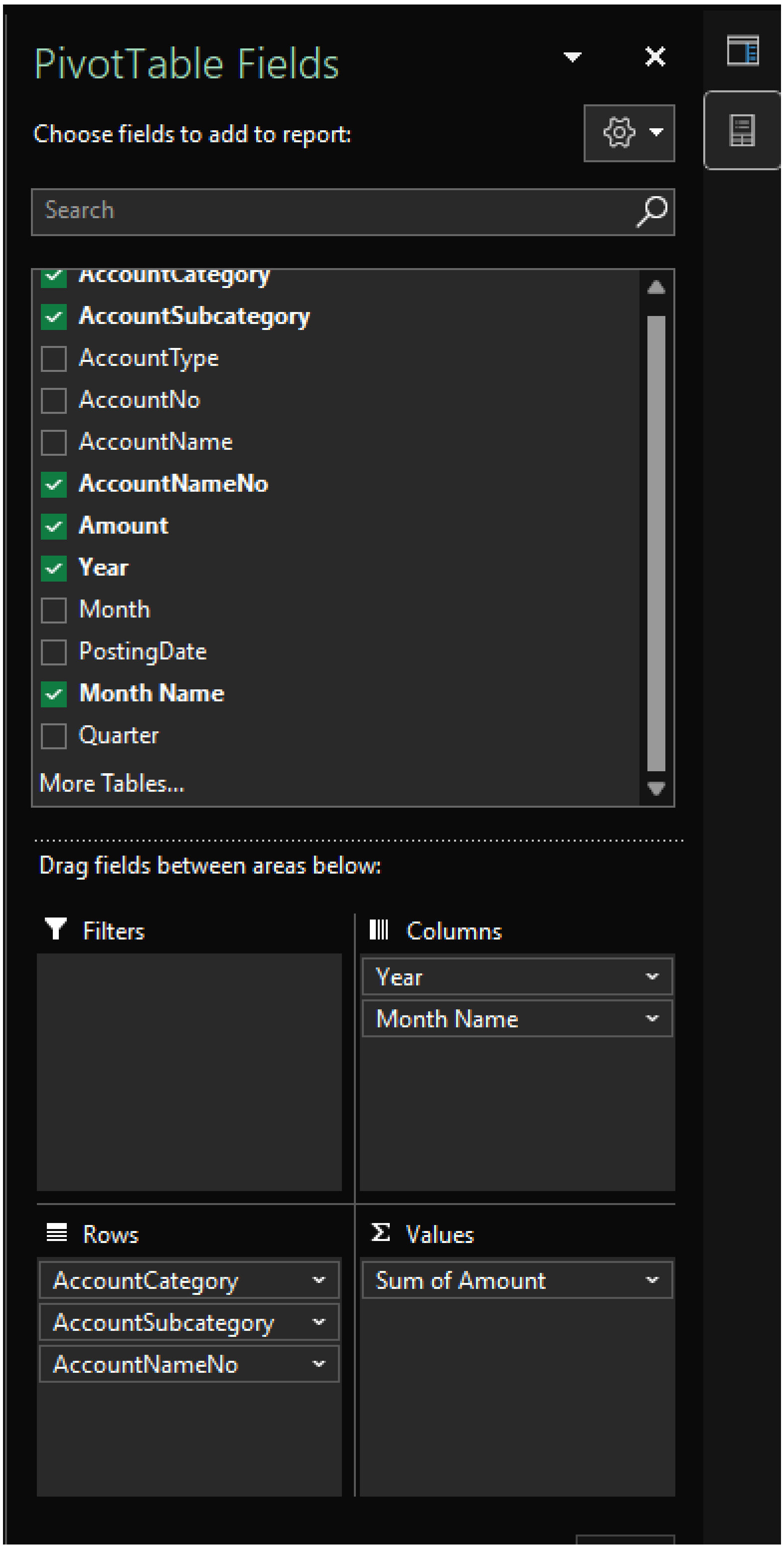
Now we get a new table in Excel:



It's named **DataCorrected** and contains the columns we added.

You can use this table to create a new pivot table:





And did you notice the Sum method was automatically selected for the Amount column? This is because we defined it as a Decimal datatype in Power Query.

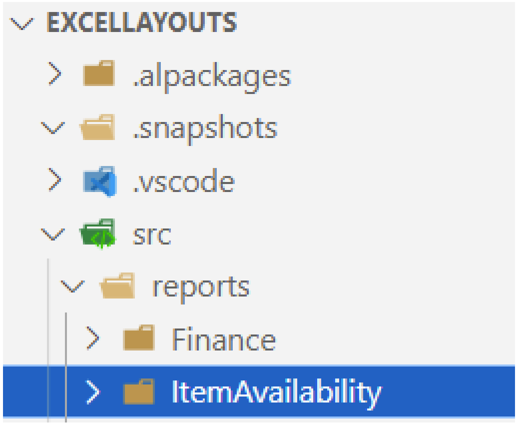
Now **Save** and **Close** the Finance1.xlx file.

Later we might add other layouts to the report, as a user, in Business Central.

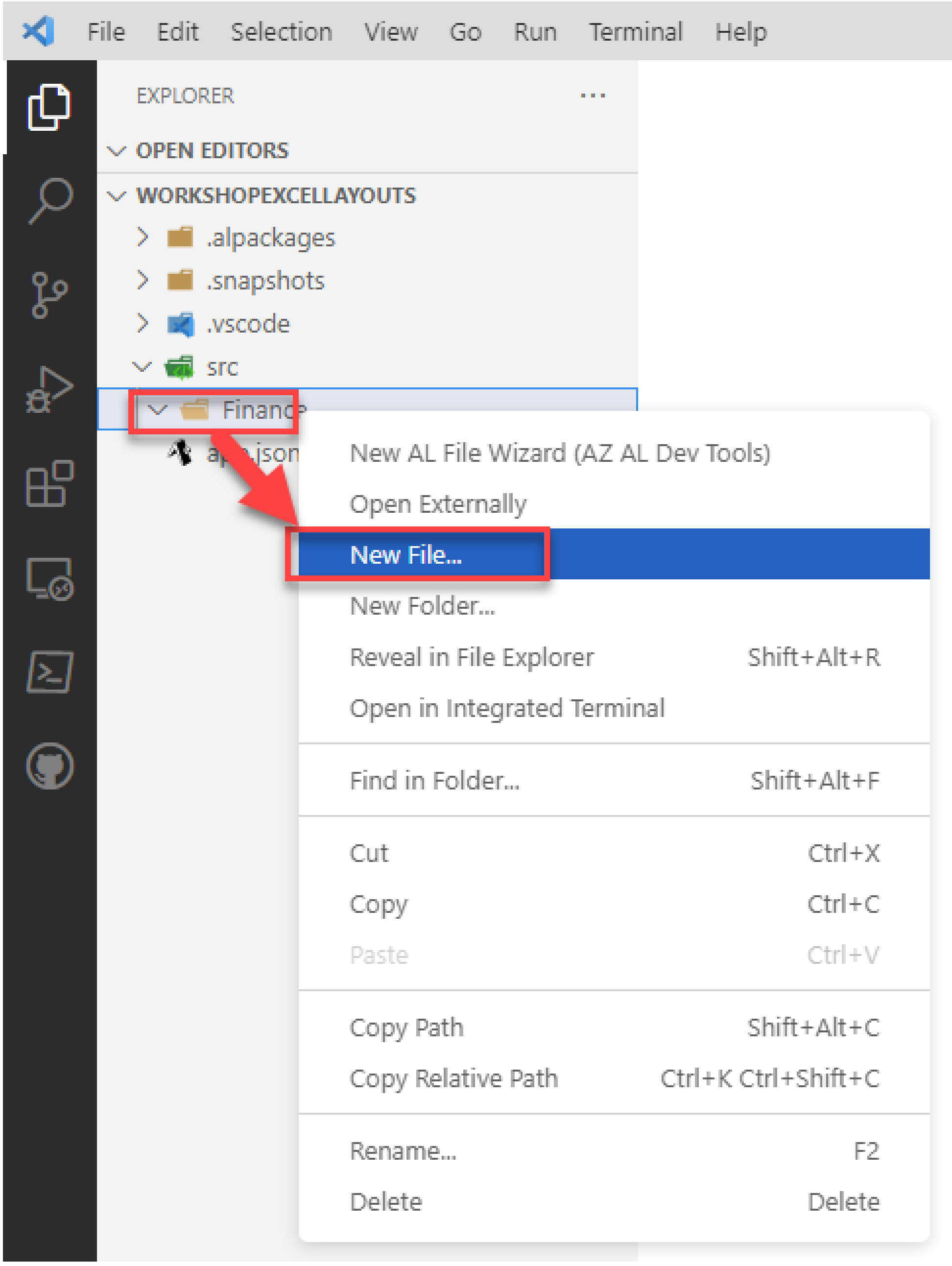
To create the ItemAvailability report, follow these steps:

Create the Dataset:

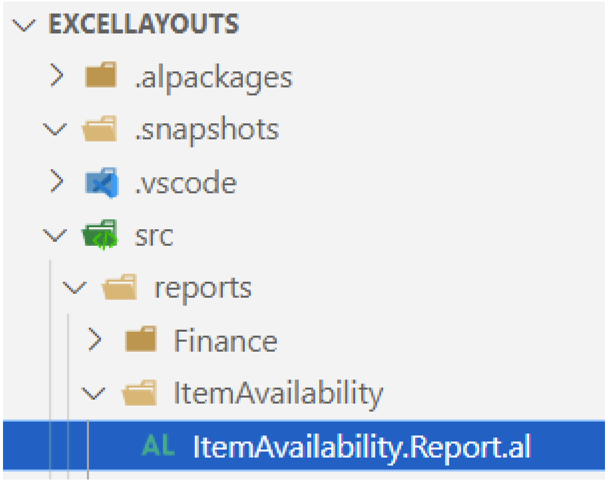
In the reports folder, create a subfolder named **ItemAvailability**:



On the **ItemAvailability** folder, right click and select **New File**:



Use this as the name of the new file: **ItemAvailability.Report.al**



The file **ItemAvailability.Report.al** will be the container of the new ItemAvailability report object.

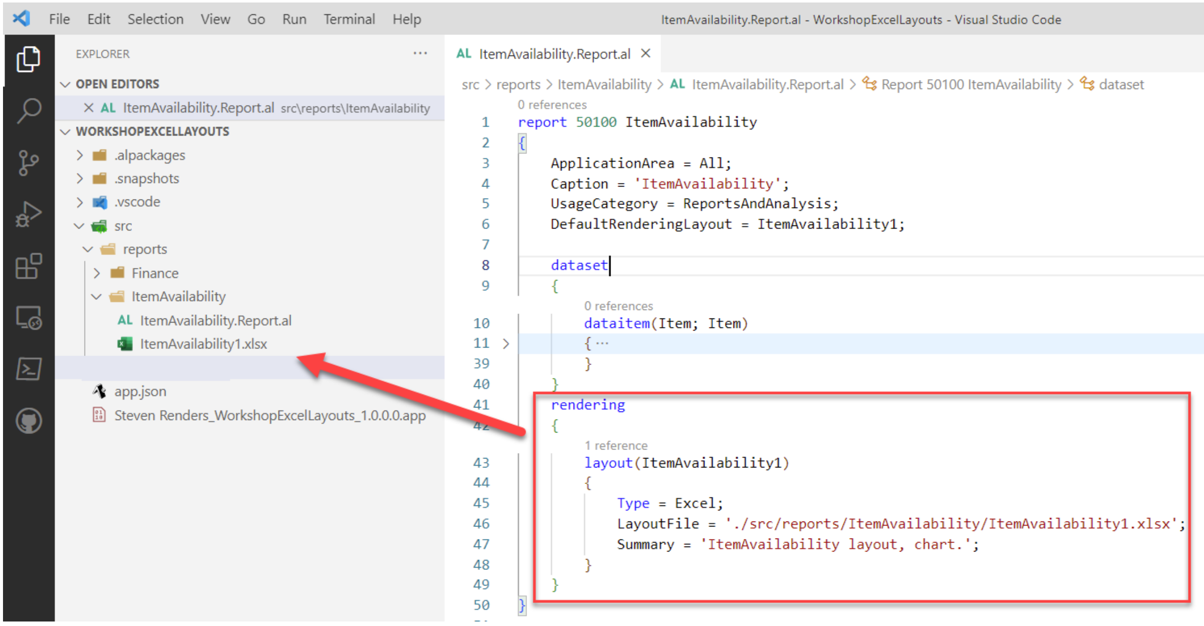
Copy/paste the following code in the ItemAvailability report file:

```
report 50100 ItemAvailability
{
    ApplicationArea = All;
    Caption = 'ItemAvailability';
    UsageCategory = ReportsAndAnalysis;
    DefaultRenderingLayout = ItemAvailability1;

    dataset
    {
        dataitem(Item; Item)
        {
            column(ItemNo; "No.")
            { }
            column(ItemDescription; Description)
            { }
            column(ItemName; Description + ' (' + Format("No.") + ')')
            { }
            dataitem(ItemLedger; "Item Ledger Entry")
            {
                DataItemLink = "Item No." = field("No.");
                DataItemTableView = sorting("Item No.");

                column(Year; System.Date2DMY("Posting Date", 3))
                { }
                column(Month; System.Date2DMY("Posting Date", 2))
                { }
                column(LocationCode; "Location Code")
                { }
                column(Quantity; Quantity)
```

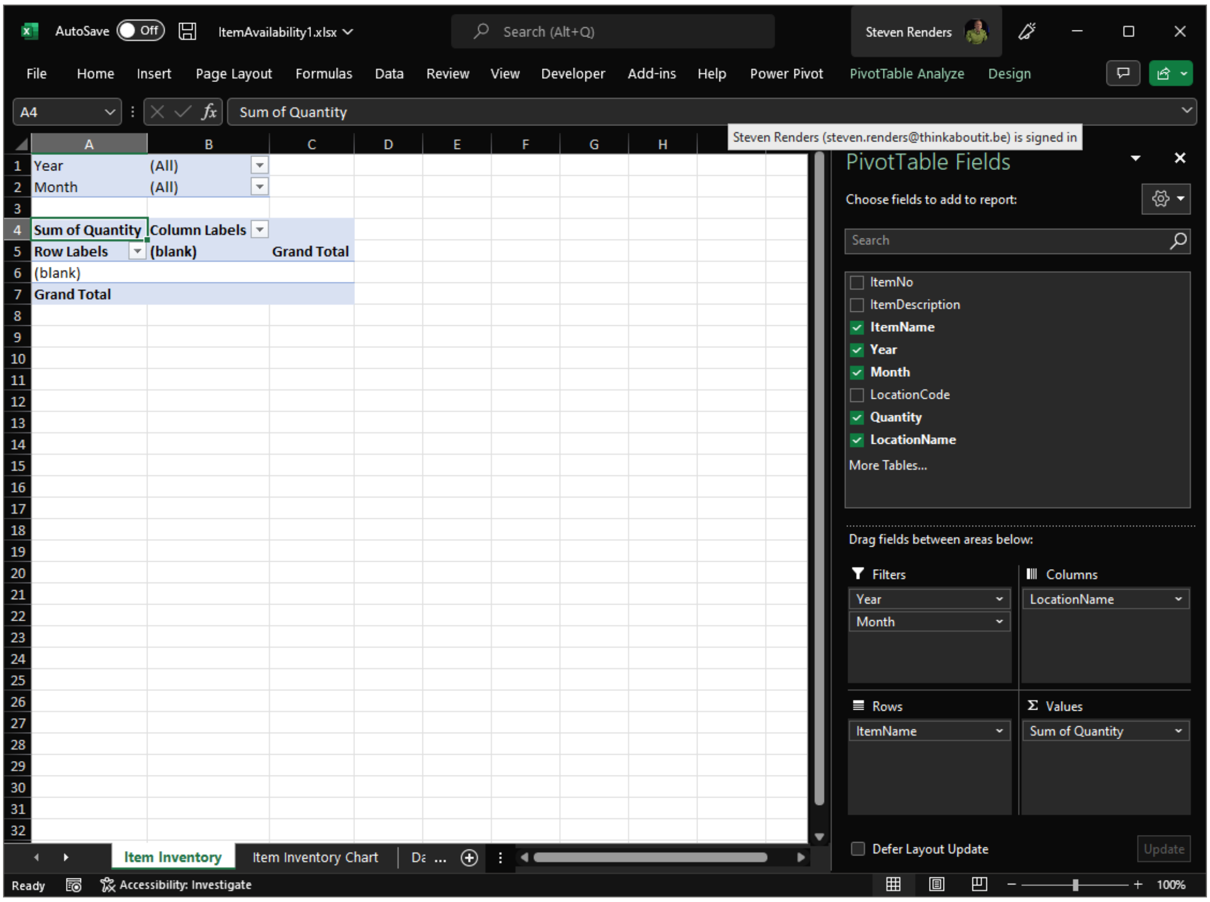
```
                { }
                dataitem(Location; Location)
                {
                    DataItemTableView = sorting(Code);
                    DataItemLink = Code = field("Location Code");
                    column(LocationName; Name)
                    { }
                }
            }
        }
    }
    rendering
    {
        layout(ItemAvailability1)
        {
            Type = Excel;
            LayoutFile =
'./src/reports/ItemAvailability/ItemAvailability1.xlsx';
            Summary = 'ItemAvailability layout, chart.';
        }
    }
}
```

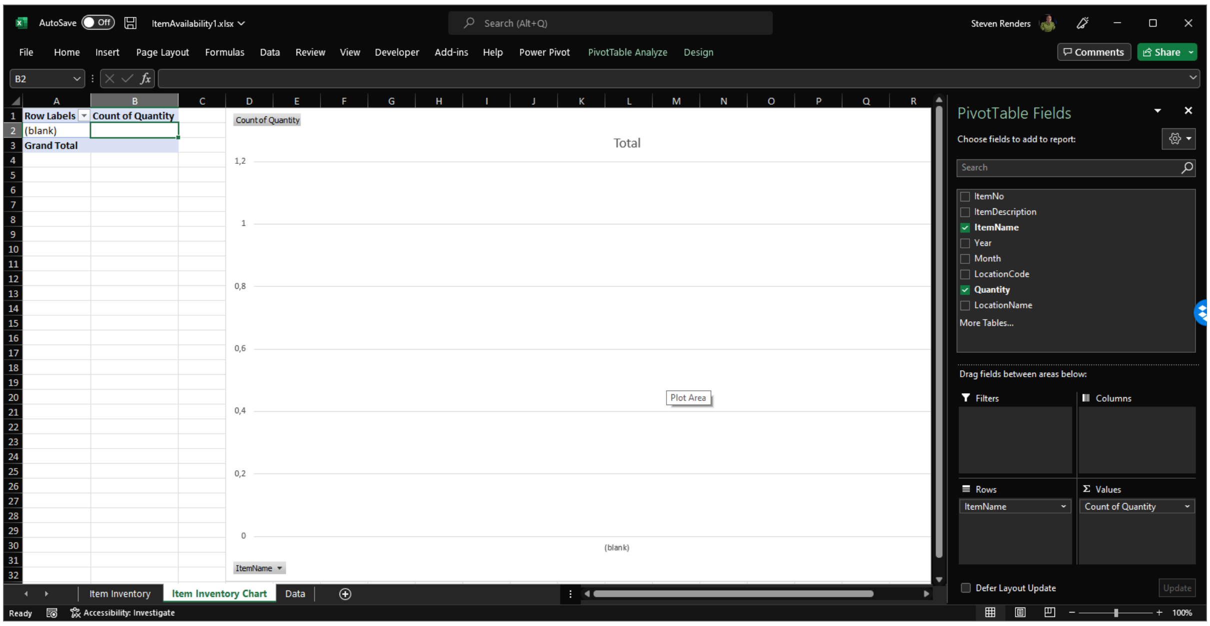
Create the Excel layout:

Similar to how you did it in the previous report, now package the report to create the **Excel** layout:

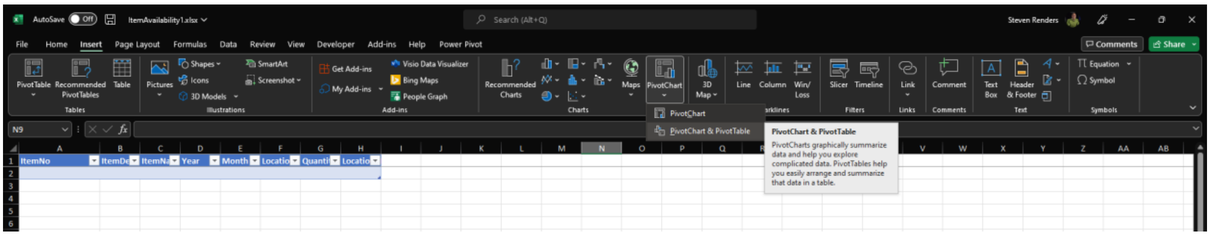


Then open the Excel file, and add a pivot tables to it:





As you can see I added 2 new worksheets, one with a simple pivot table and another with a pivot table and a pivot chart.

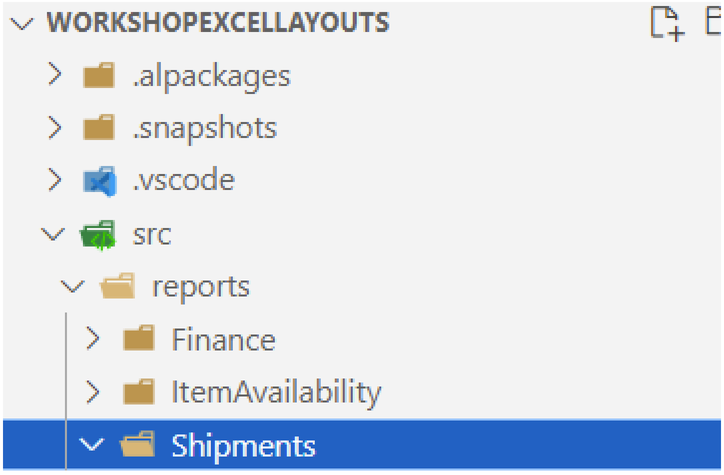


Now close and save the xlsx file.

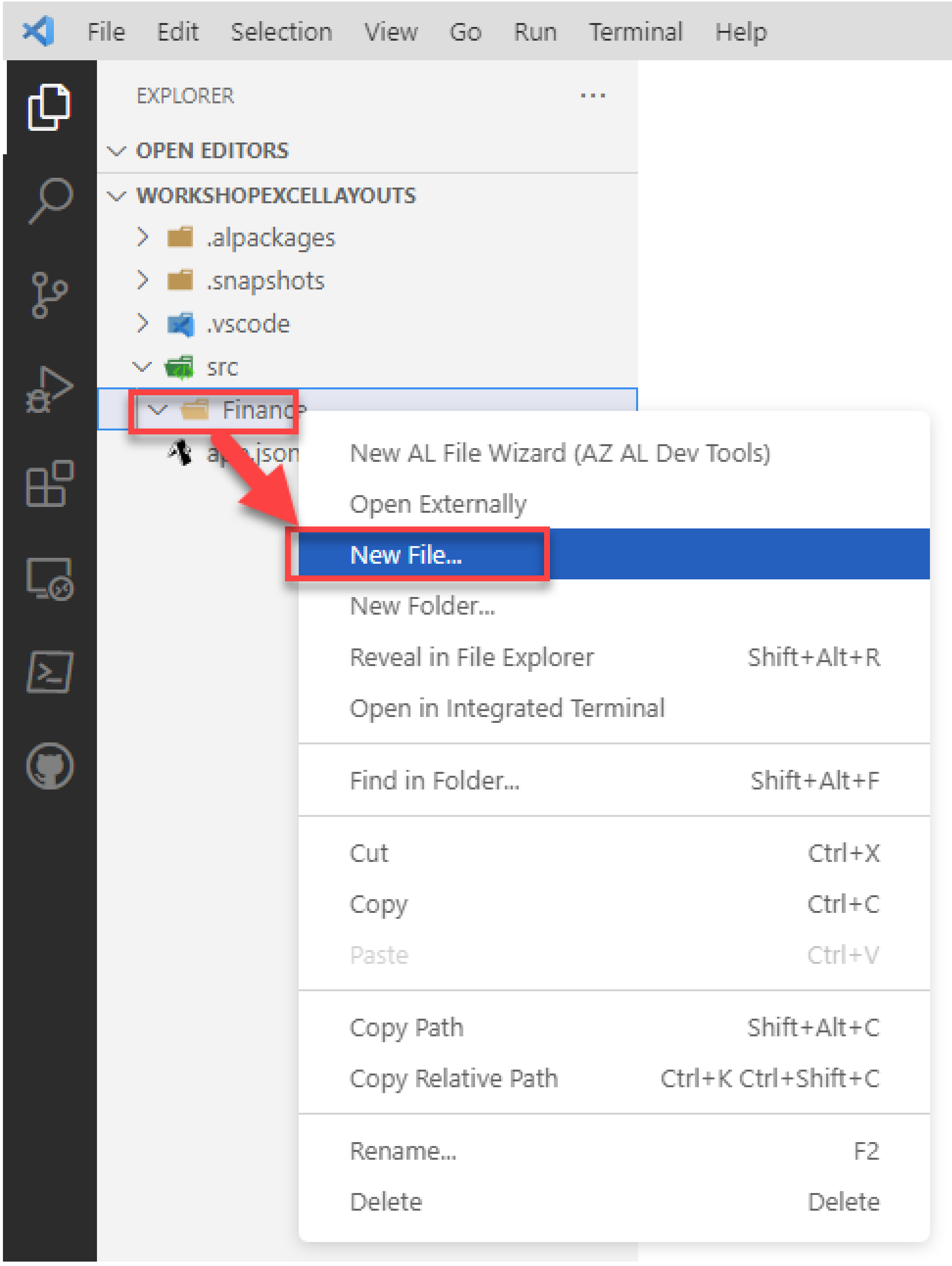
To create the Shipments report, follow these steps:

Create the Dataset:

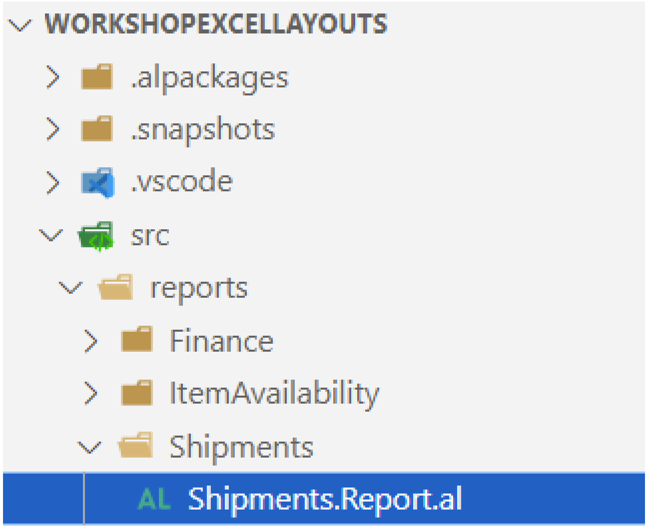
In the reports folder, create a subfolder named **Shipments**:



On the **Shipments** folder, right click and select **New File**:



Use this as the name of the new file: **Shipments.Report.al**



The file **Shipments.Report.al** will be the container of the new Shipments report object.

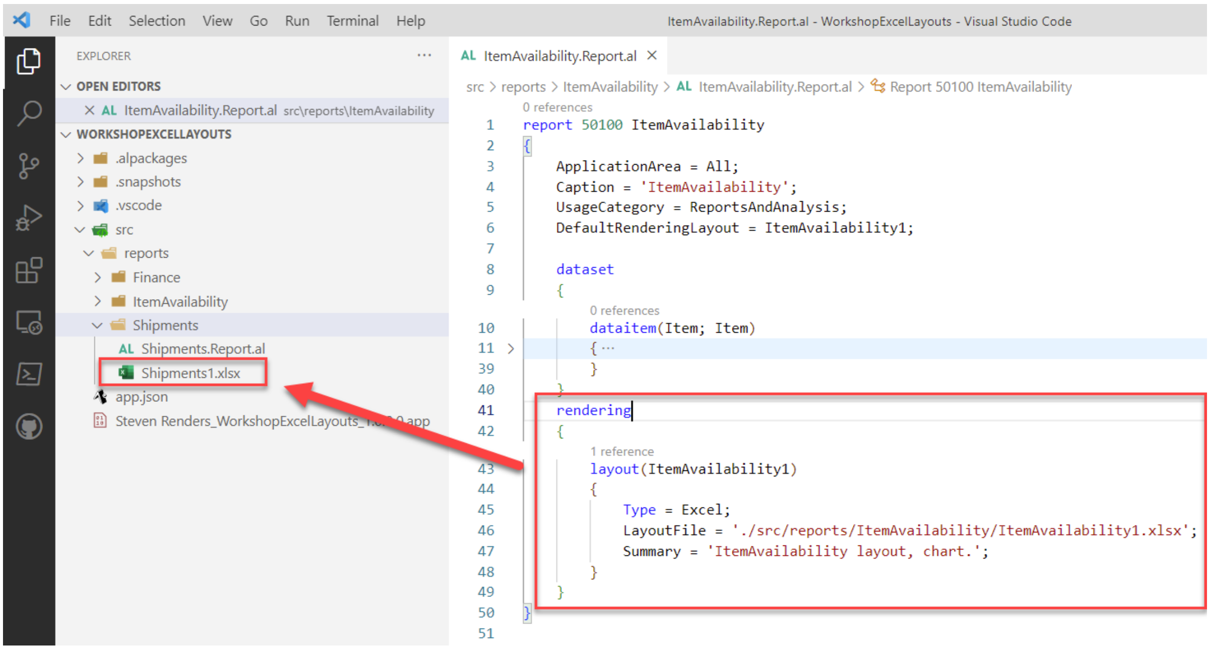
Copy/paste the following code in the **Shipments** report file:

```
report 50102 Shipments
{
    UsageCategory = ReportsAndAnalysis;
    ApplicationArea = All;
    Caption = 'Shipments';
    DefaultRenderingLayout = Shipments1;

    dataset
    {
        dataitem(SalesLine; "Sales Line")
        {
            DataItemTableView = where(Type = filter('Item'));

            column(DocumentNo; "Document No.")
            { }
            column(DocumentType; "Document Type")
            { }
            column(ItemNo; "No.")
            { }
            column(ShipmentDate; "Shipment Date")
            { }
            column(ShipmentYear; System.Date2DMY("Shipment Date", 3))
            { }
            column(ShipmentMonth; System.Date2DMY("Shipment Date", 2))
            { }
            column(QtytoShip; "Qty. to Ship")
            { }
            dataitem(Customer; Customer)
```

```
            {
                DataItemLink = "No." = field("Sell-to Customer No.");
                DataItemTableView = sorting("No.");

                column(CustomerNo; "No.")
                { }
                column(CustomerName; Name)
                { }
            }
        }
    }

    rendering
    {
        layout(Shipments1)
        {
            Type = Excel;
            LayoutFile = './src/reports/Shipments/Shipments1.xlsx';
            Summary = 'Shipments layout, pivot and chart.';
        }
    }
}
```

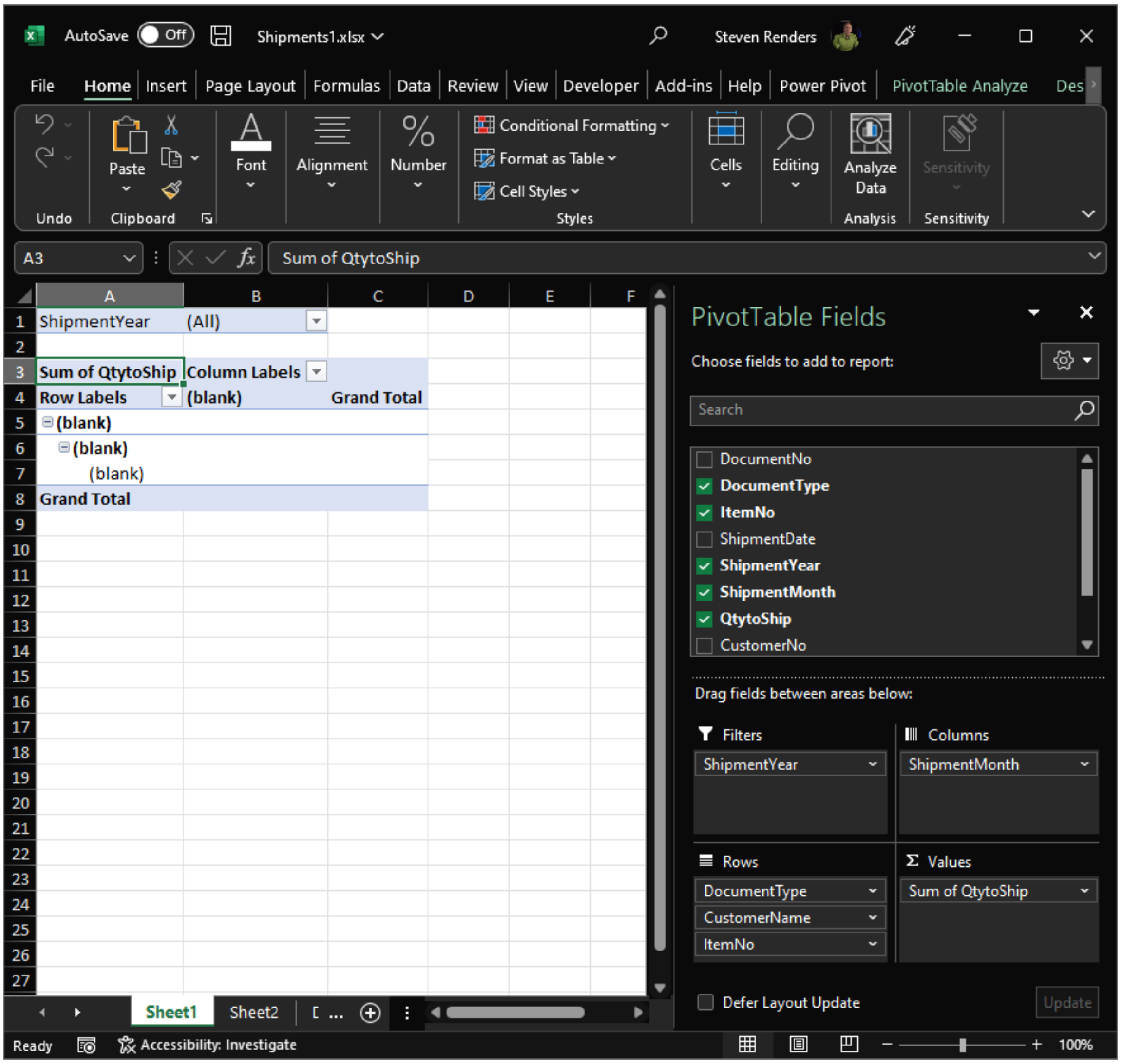
Create the Excel layout:

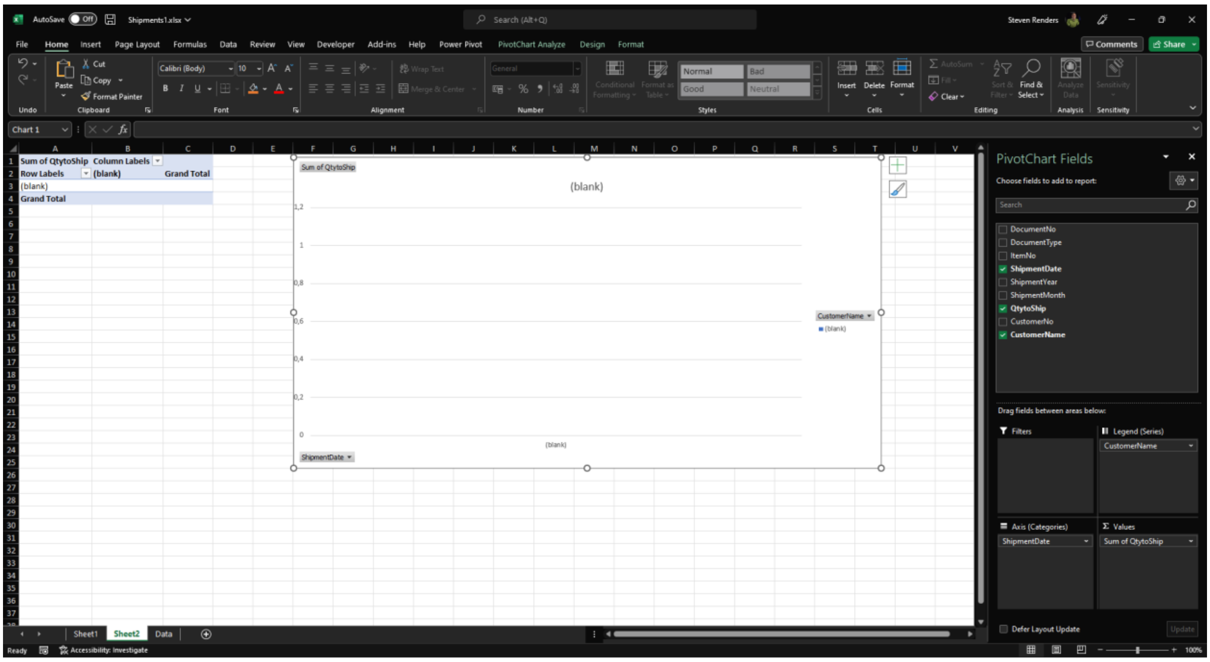
Like how you did it in the previous report, now package the report to create the **Excel** layout:



Then open the Excel file and add a pivot tables to it.

Here I added two different ones, a pivot table, and a pivot chart:

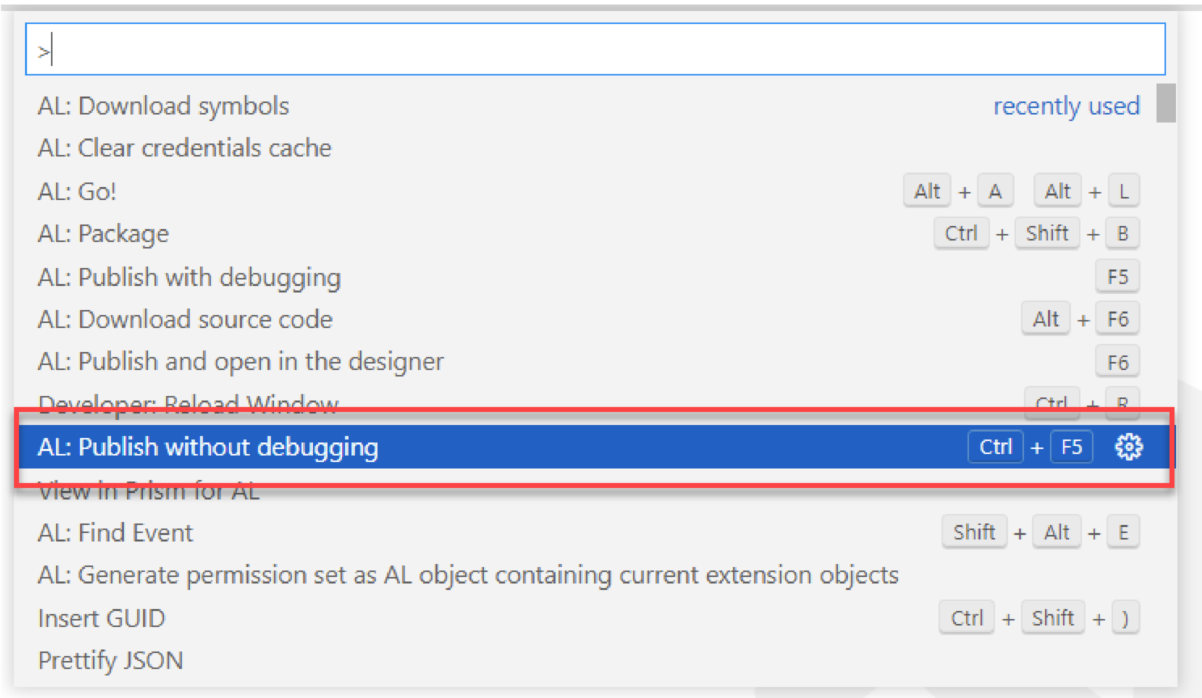




Now **close** and **save** the xlsx file.

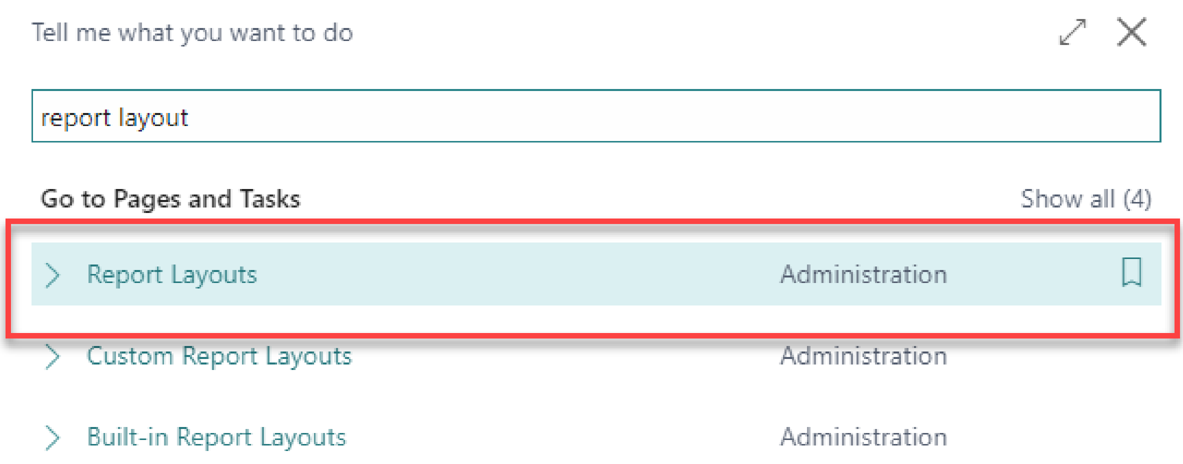
To re-deploy the extension, and run the reports, follow these steps:

In Visual Studio Code, select **Publish without debugging**, in the Command Palette:

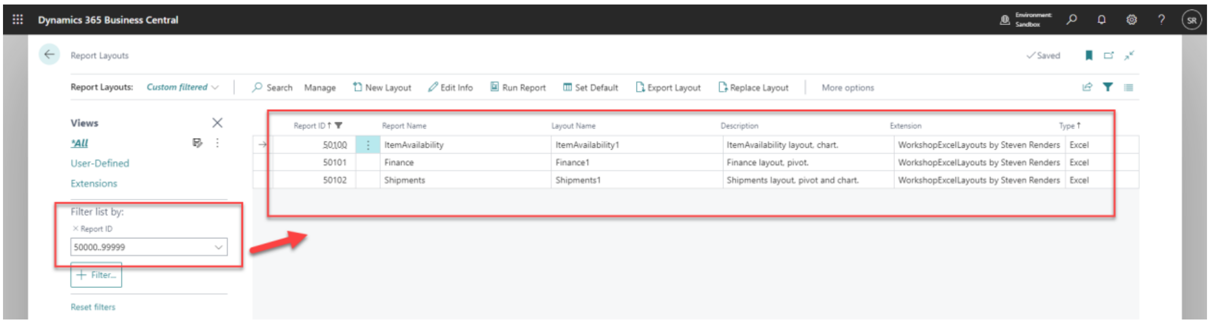


Your app is now published to your sandbox, after which your browser opens and Business Central.

In Business Central, open the **Report Layouts** page:

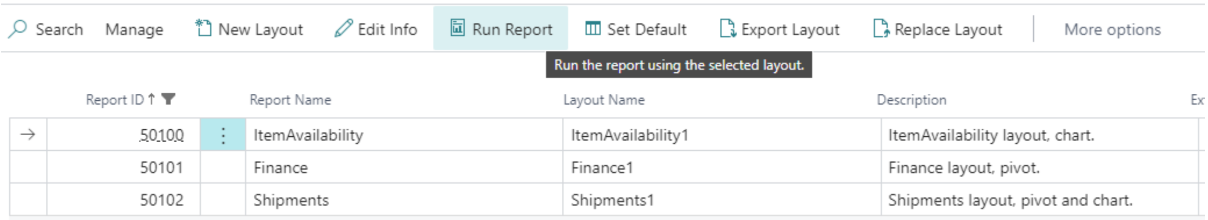


In this page, apply a **filter**:

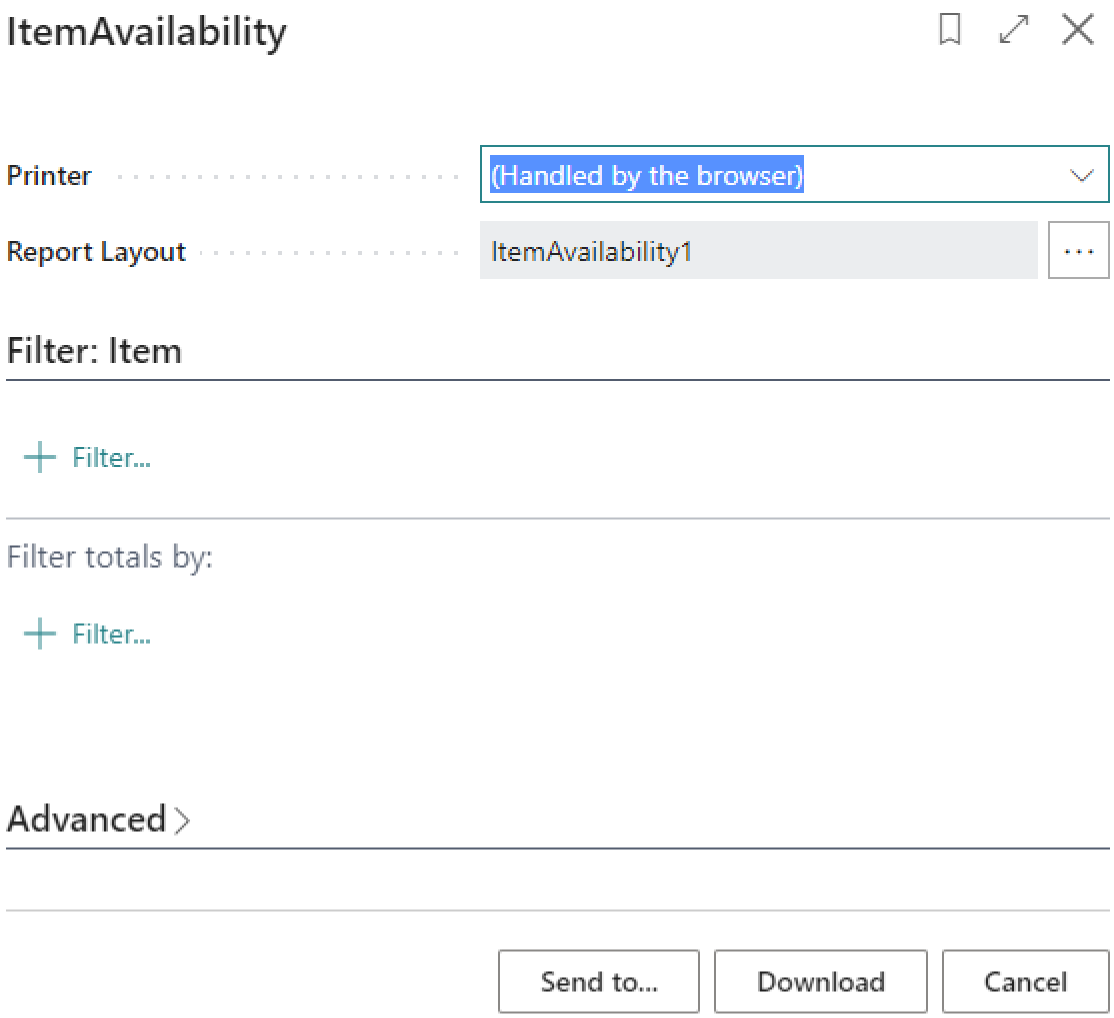


Now you can see your 3 report layouts in Business Central.

To run them, select a layout and then Run Report:



The request page opens:



Then click **Download** to generate and download the excel file.

Open the Excel file, to look at the data.

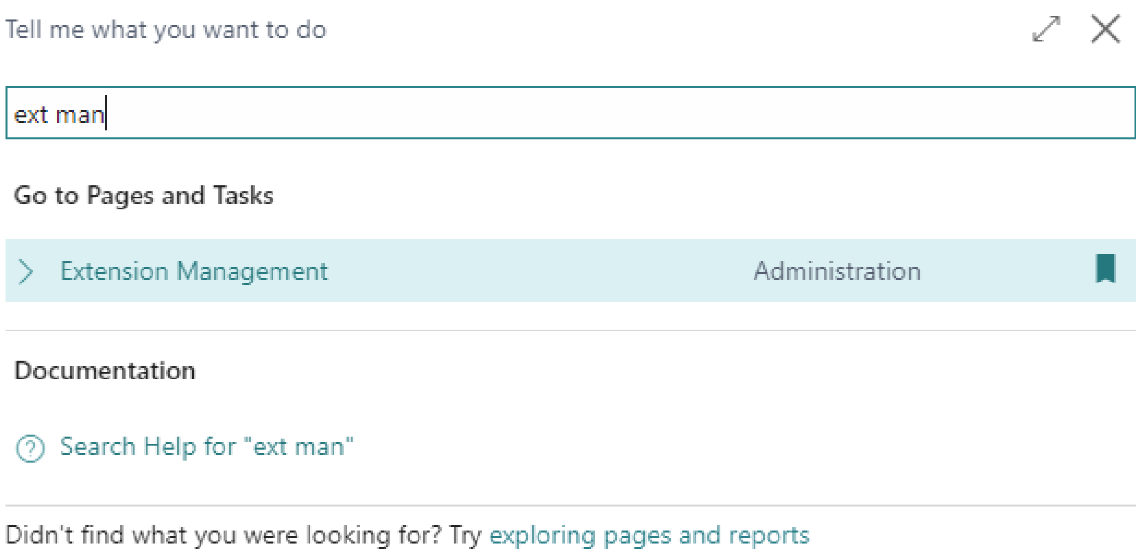
You now have three new reports in Business Central, with Excel layouts.

To import the extension in Business Central, follow these steps:

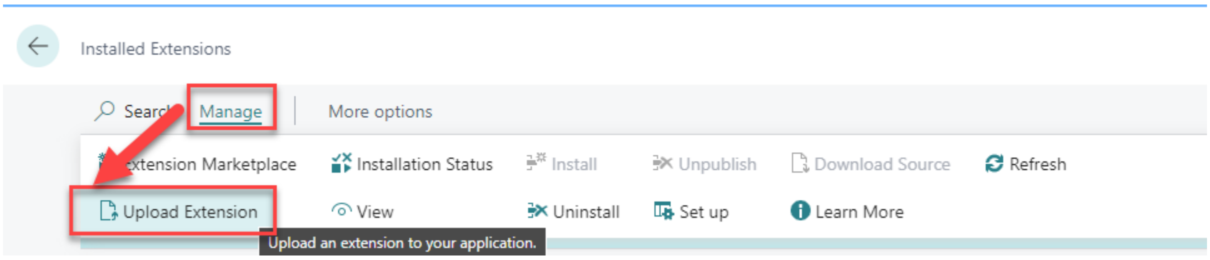
If you don't want to develop the extension in VScode, but only import the solution in Business Central, then follow these steps. If you have developed the extension in VScode, then you can skip this step, because you then have already published the extension to your sandbox.

If you however wish to deploy the extension to your production environment, you can follow these steps.

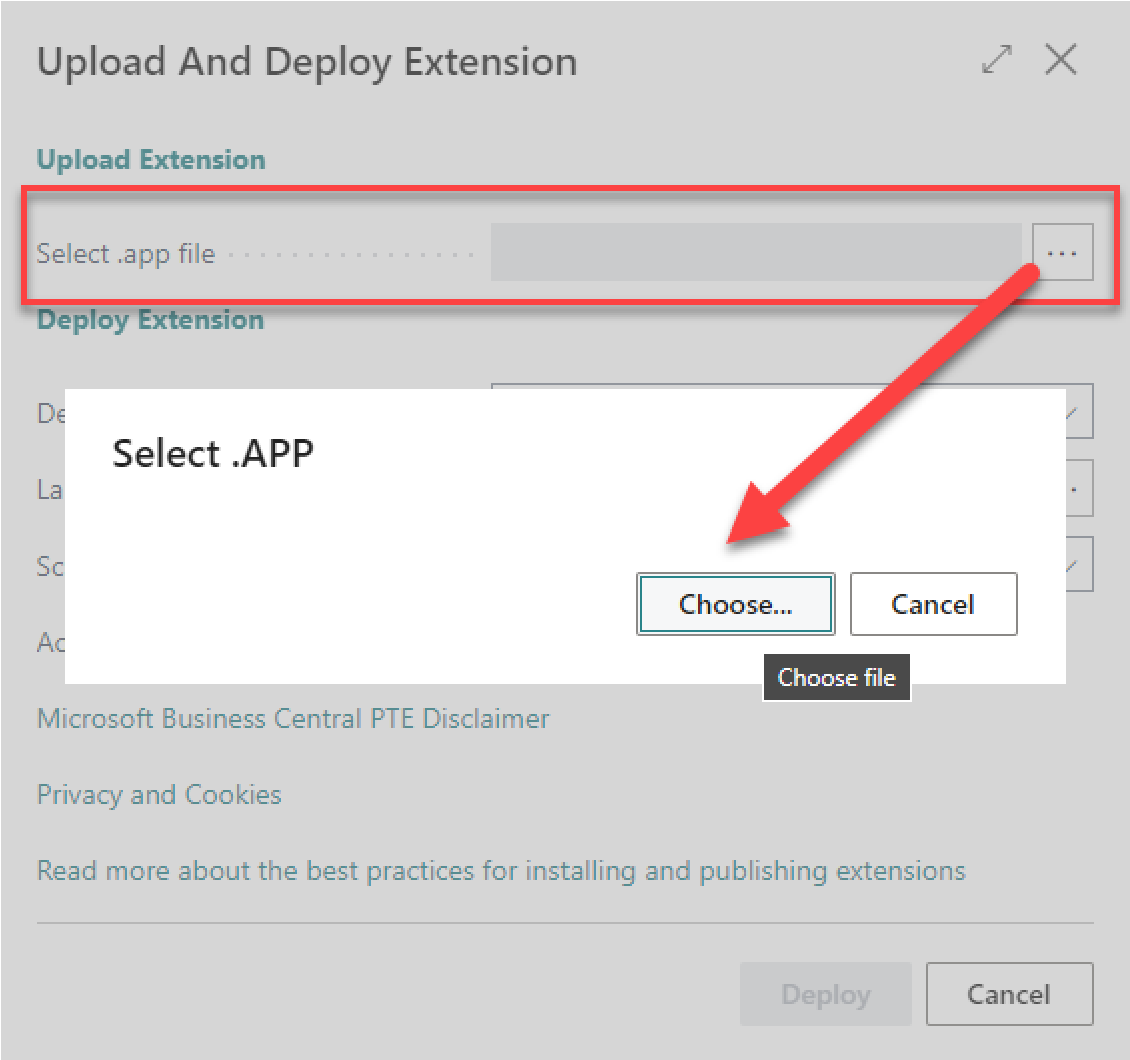
In Business Central open the **Extension Management** page:



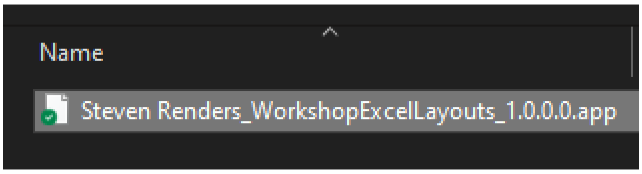
Select, **Manage**, **Upload Extension**:



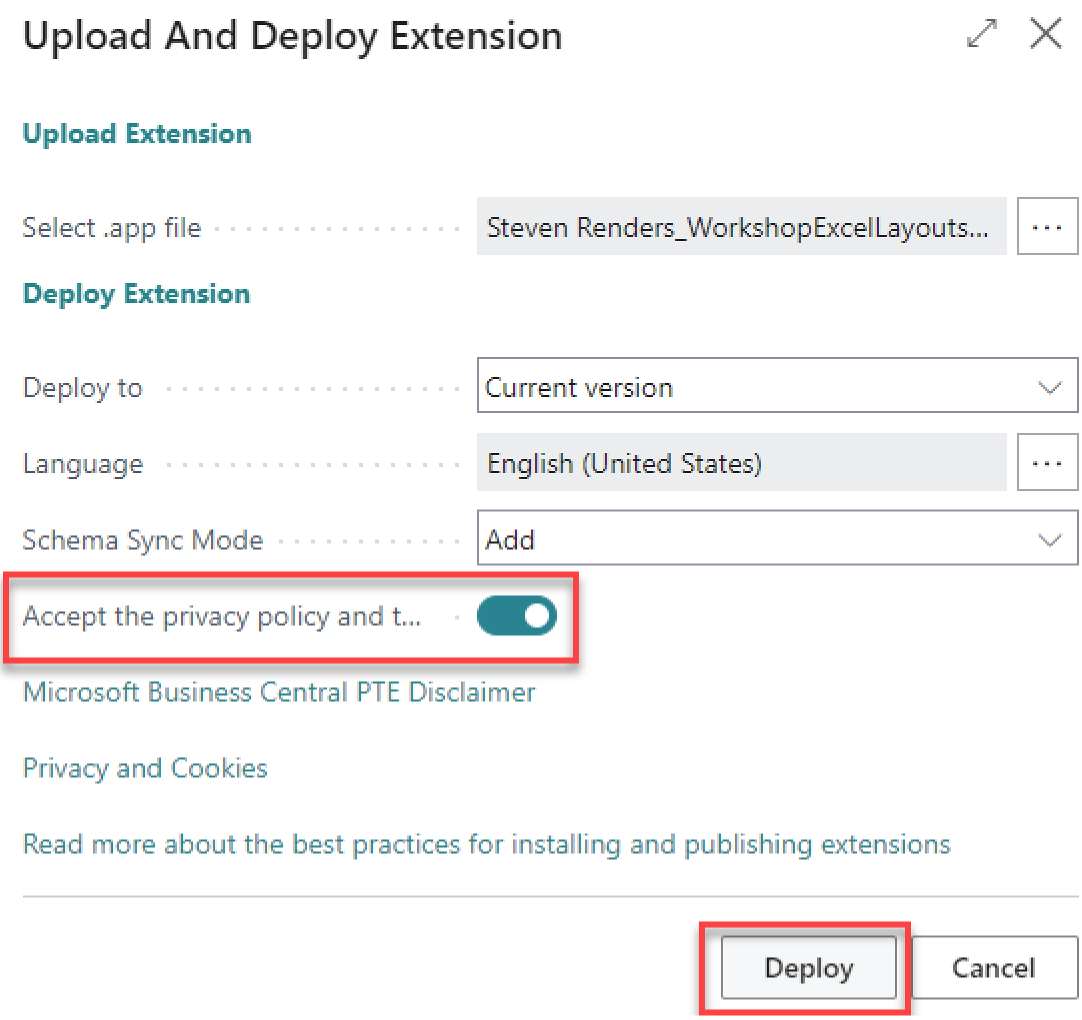
Select the app file:



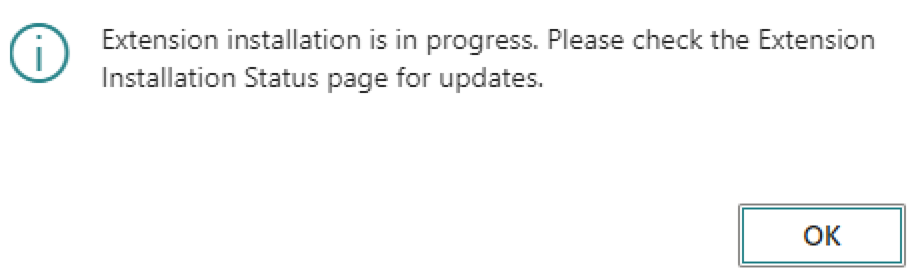
Then select the app file in the handouts folder:



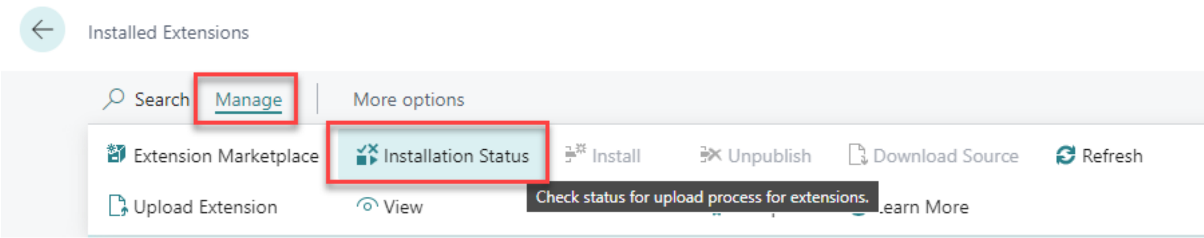
Accept the privacy policy and click **Deploy**:

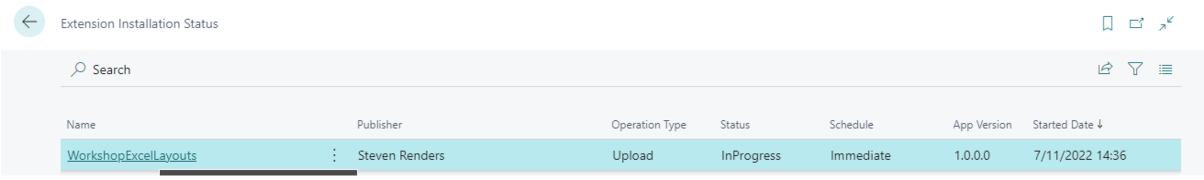


Click **OK**:

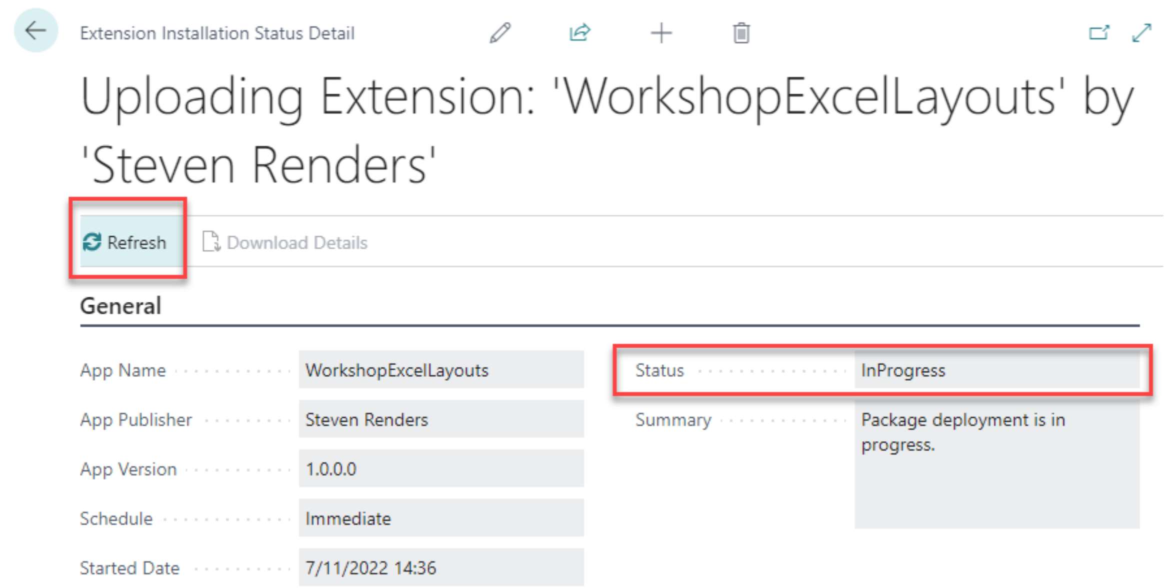


Select **Manage**, **Installation Status**:

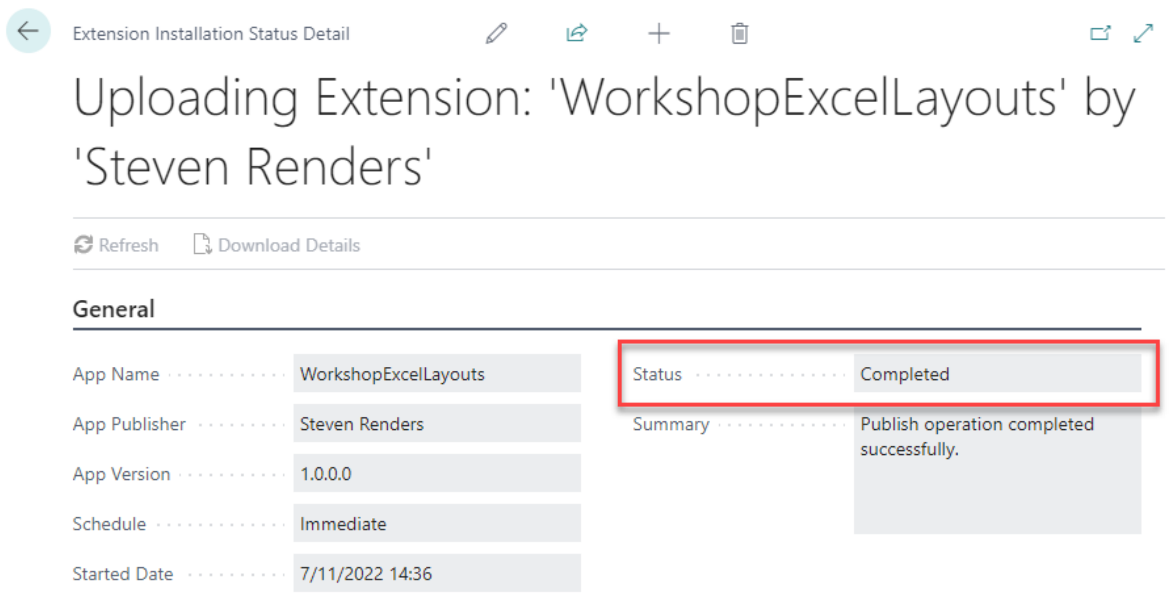




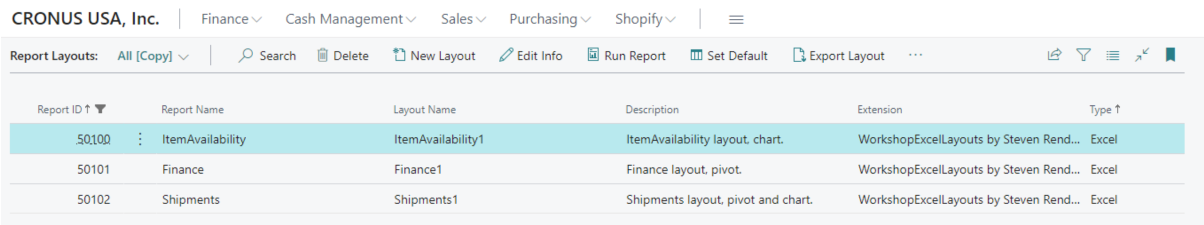
Click on the extension **name**:



Click on **Refresh**, until the status changes to **Completed**:



The app is installed now, and the reports should be available in the **Report Layouts** page:

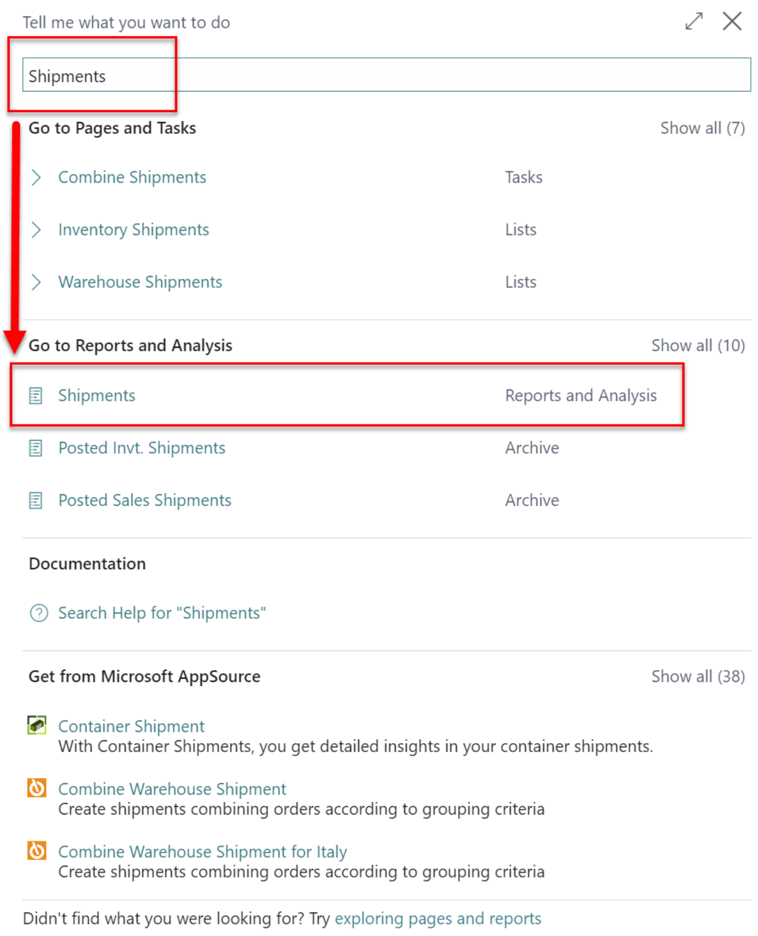


Consuming Excel layouts

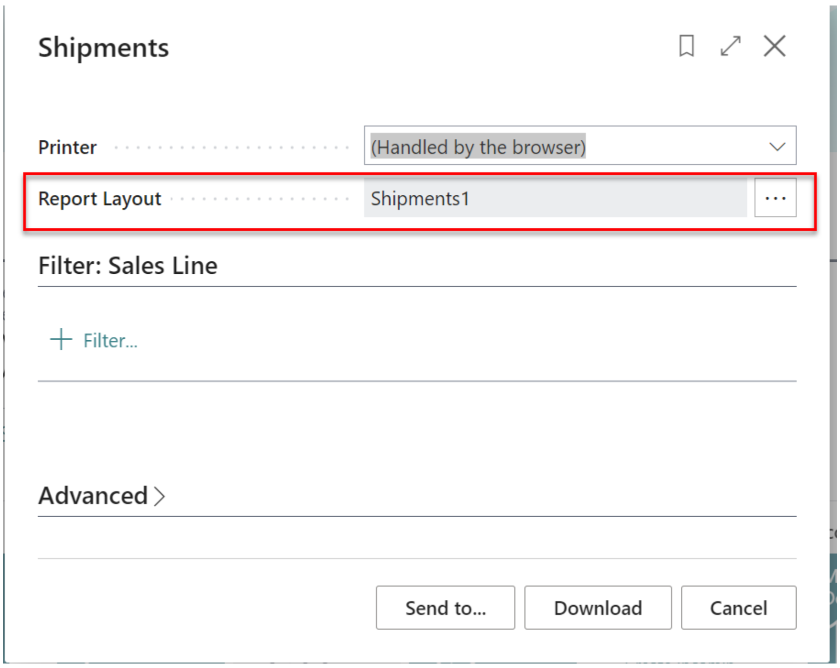
Now when initial reports and report layouts are developed by developers, we can use them within Dynamics 365 Business Central.

Enhance the Shipments layout

To consume developed reports in Dynamics 365 Business Central search for one of them using the **Tell me** functionality. For example: **Shipments**

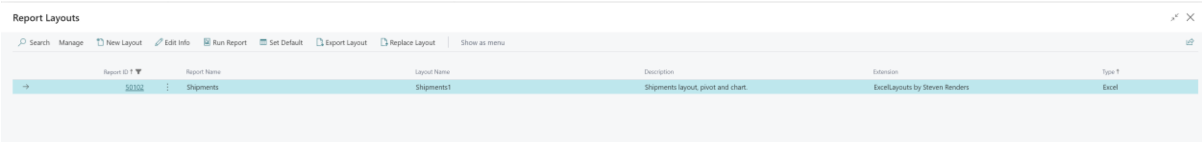


When the Shipments report request page opens you will can select the one of the existing layouts.



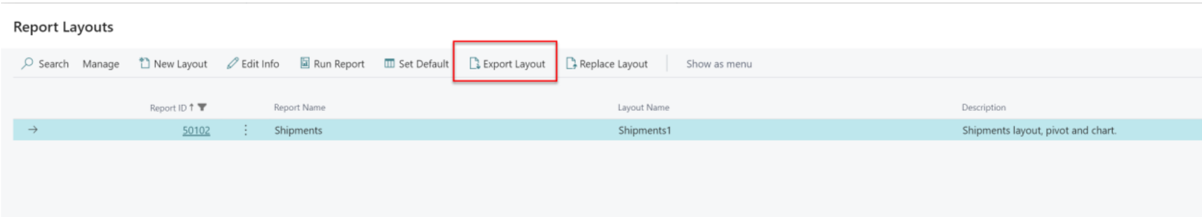
Click Assist edit (…) action.

As we developed only one Excel layout, we have the possibility only to select one Excel layout in the **Report Layouts** screen.

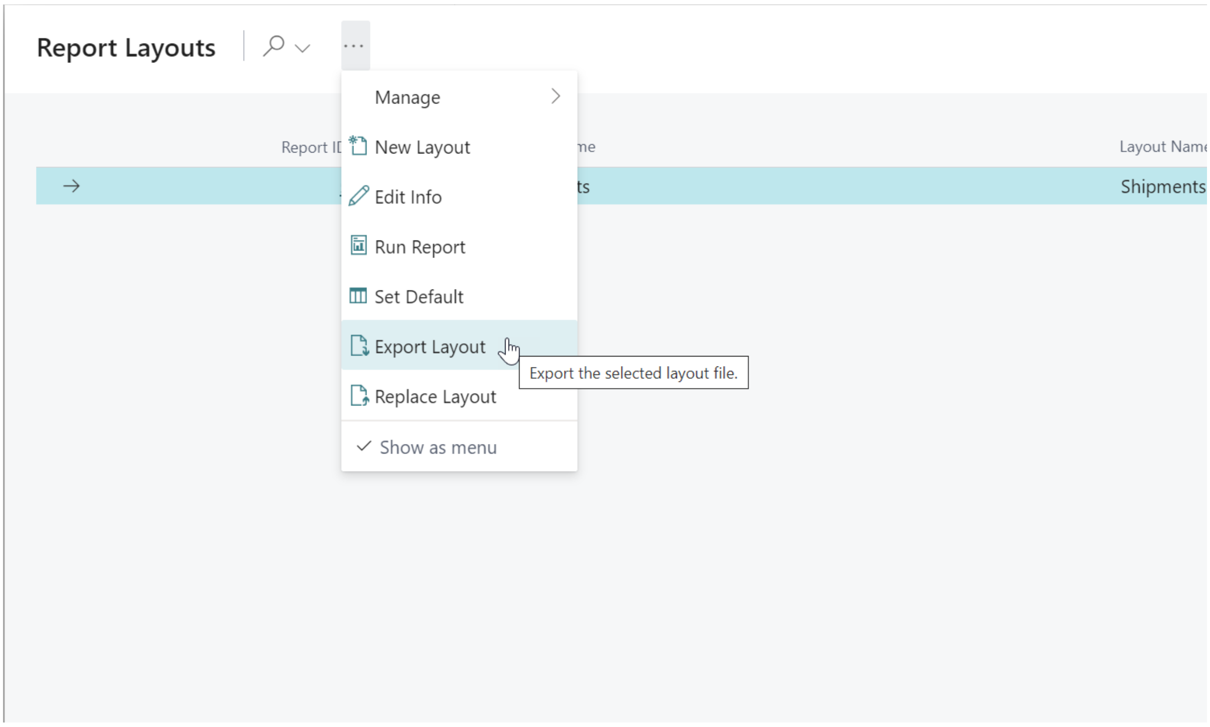


To create a new Excel layout first we need to use one of the existing Excel layouts.

Click **Export Layout** action in action menu.

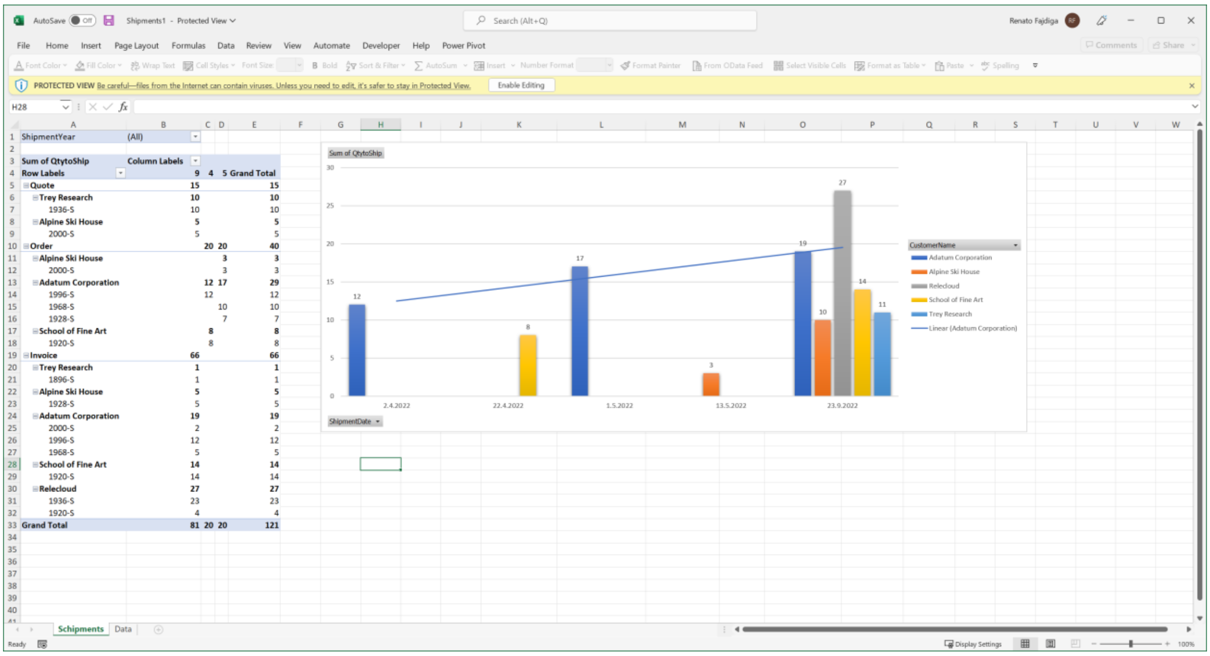


Or if you don't see Action menu, click **Show the rest** (…) action and then click **Export Layout.**



Dynamics 365 Business Central will download Excel file with Layout Name to your PC.

Open the Excel layout.



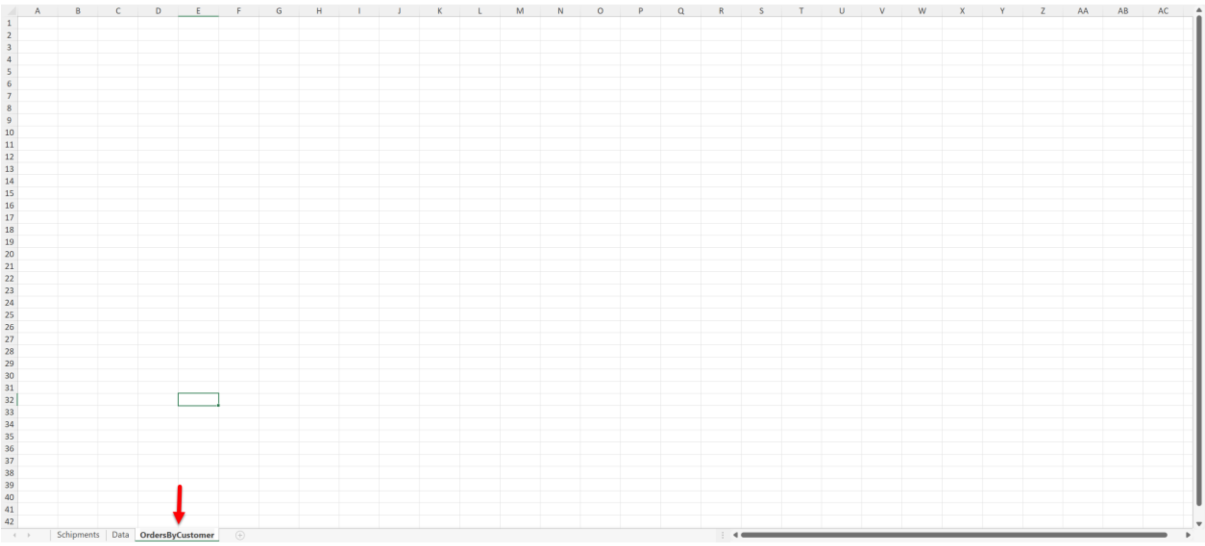
You will see two tabs:

1. Schipments – contains pivot table based on Data and some chart
2. Data – contains raw data from Dynamics 365 Business Central

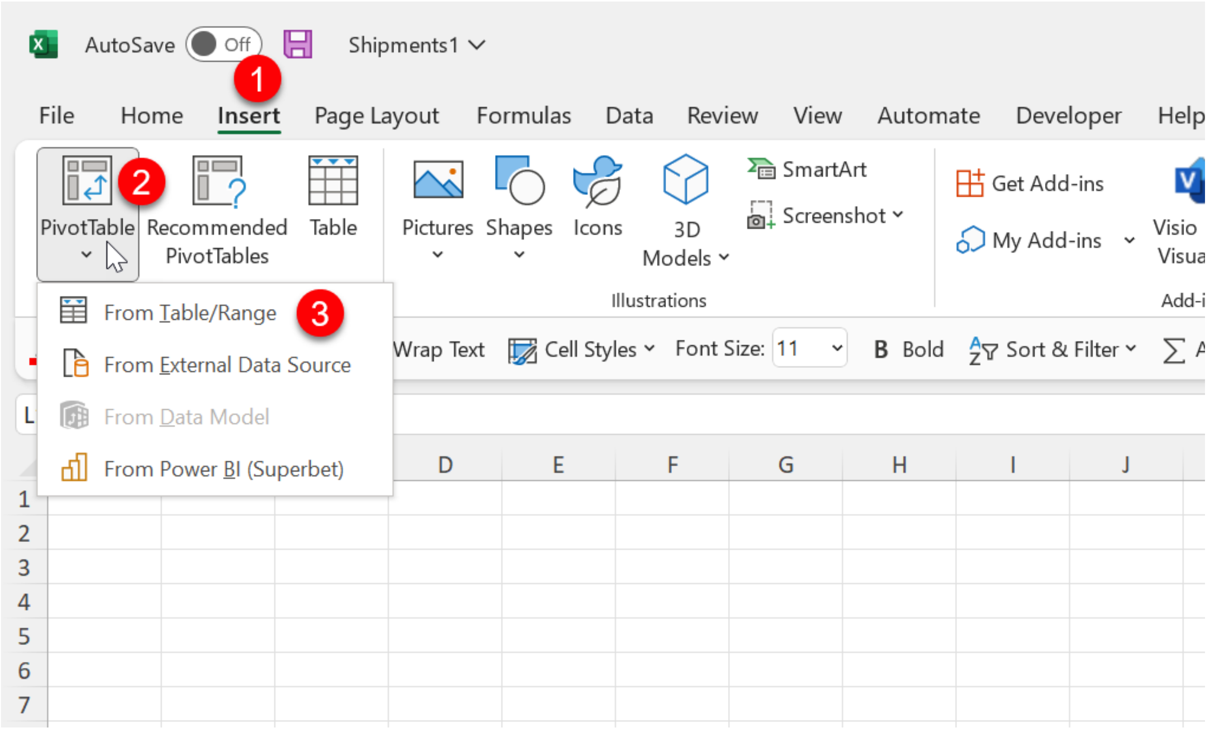
Click **Enable Editing**.

Click **+** to add a new Excel workbook sheet

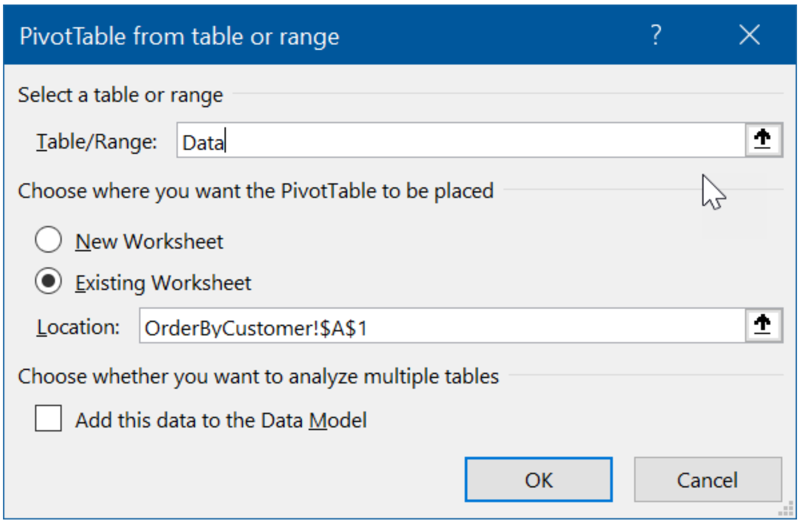
Rename Sheet2 (or Sheet1) to **OrdersByCustomer**



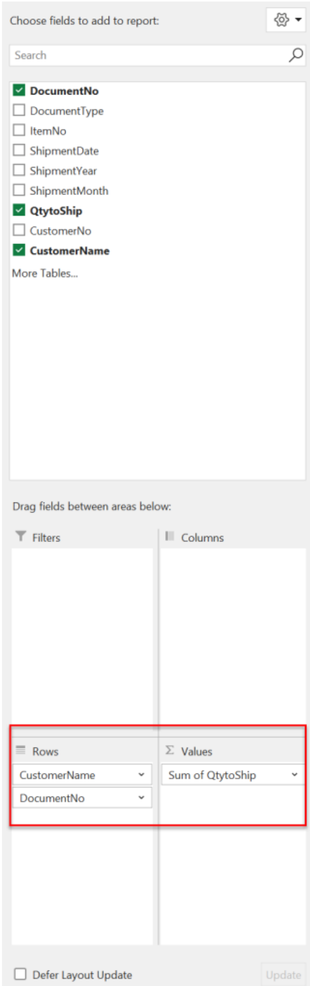
In the Action Menu click **Insert -> Pivot table -> From Table/Range**



In **PivotTable from table or range** window, in field **Table/Range** write **Data**. This will point to our raw export table from Dynamics 365 Business Central that is available in the Data sheet.



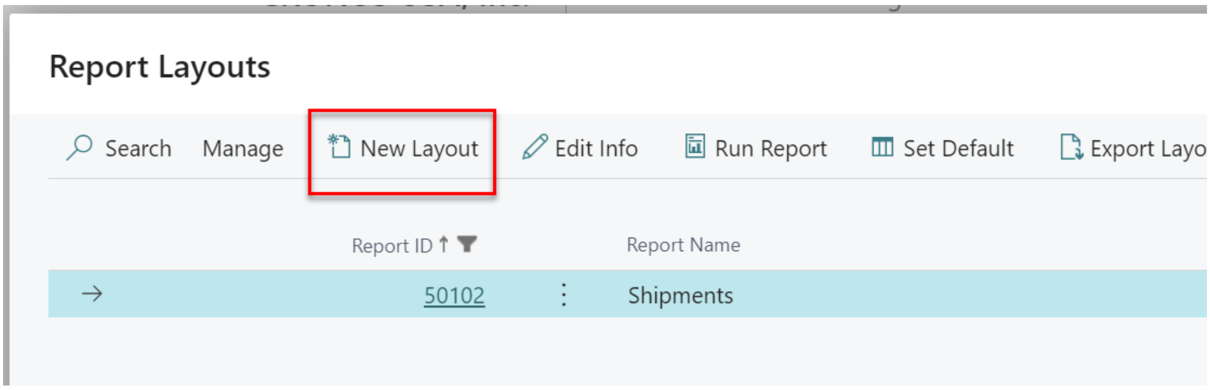
Select **CustomerName** and **DocumentNo** as **Rows** and **QtytoShip** as **Values**. Make sure that you are summing up **QtytoShip**.



Save and close the Excel file.

In order to use the new layout you have to upload it into **Report layouts** inside Dynamics 365 Business Central.

Return to **Report Layouts** screen in Dynamics 365 Business Central and click **New Layout.**



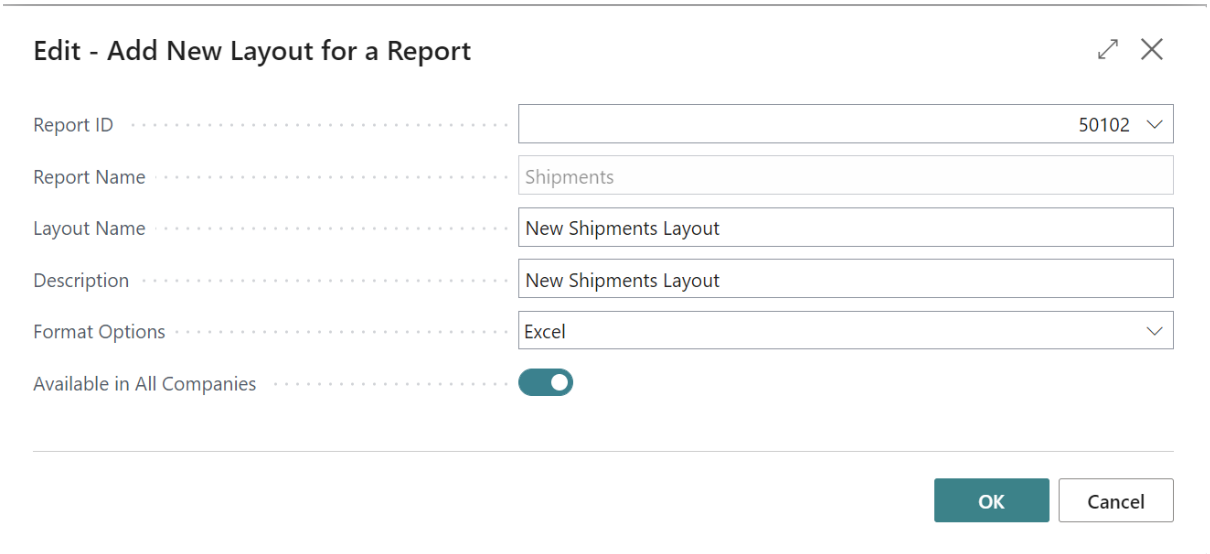
In **Edit – Add New Layout for a Report** screen select following

Report ID and Report Name leave as it is.

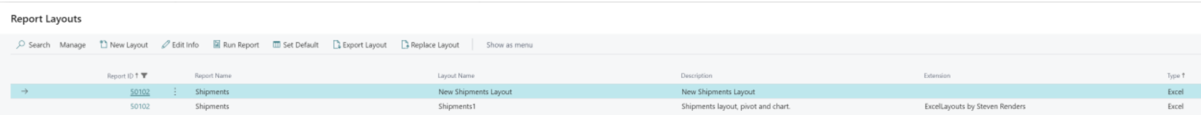
Layout **Name** and **Description** write **New Shipments Layout**. **Format Options** choose **Excel**. If you want that this layout is available only for current company **disable Available in All Companies**.

I will leave **Available in All** companies to **Yes**.

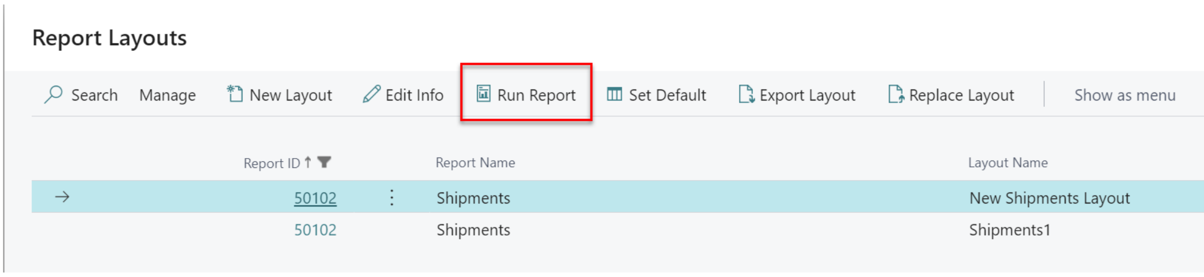
Click OK.



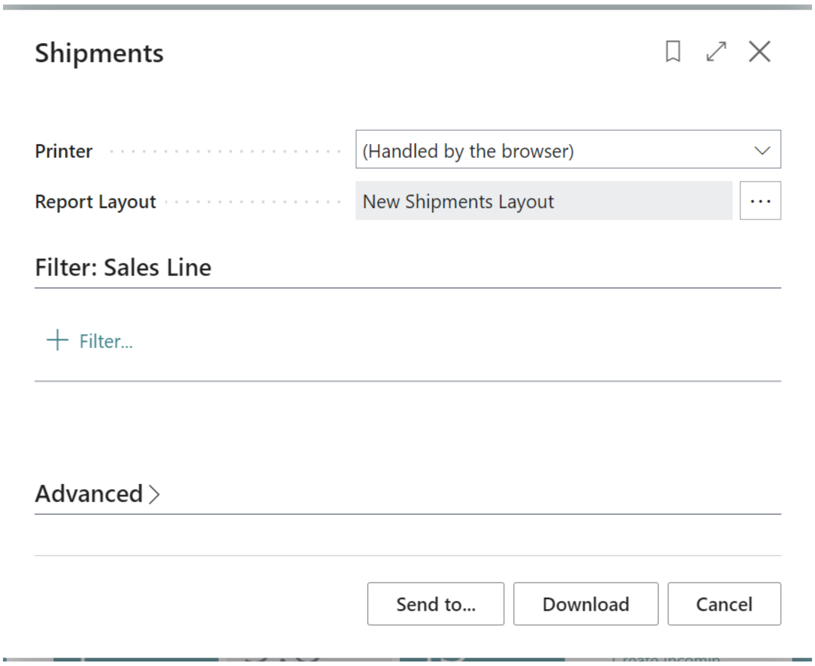
In **Choose Excel layout files** choose Excel file saved before.



Now you will have 2 Excel layouts for the same report. To test the layout click **Run Report** action in action menu.

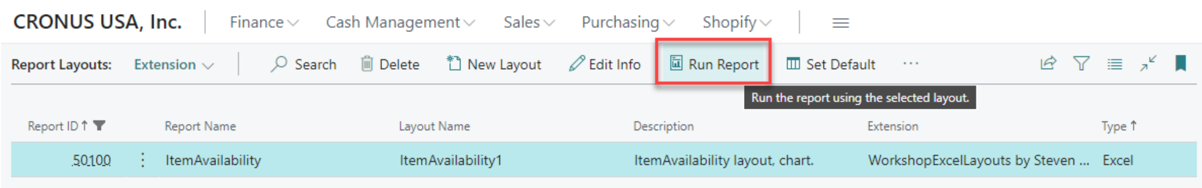


If you run **Shipments** report from **Tell me** you can now also choose **New Shipments Layout** that we developed.



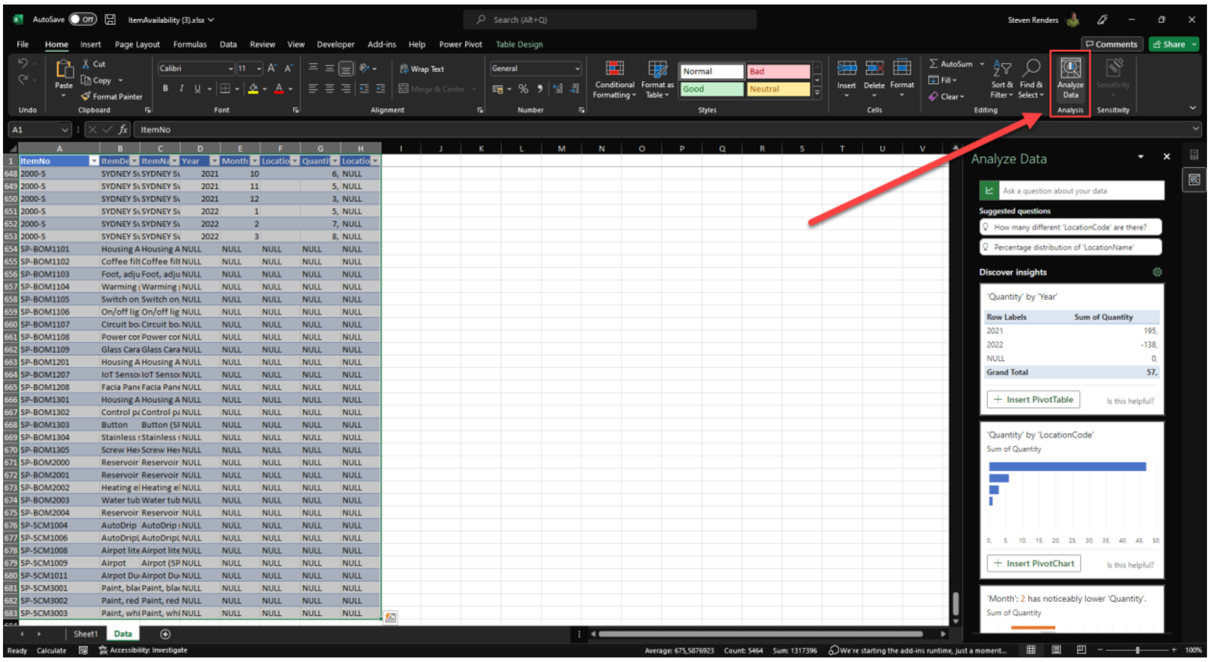
Enhance the ItemAvailability layout

In the Report Layouts page, run the **ItemAvailability** report and download the runtime layout:

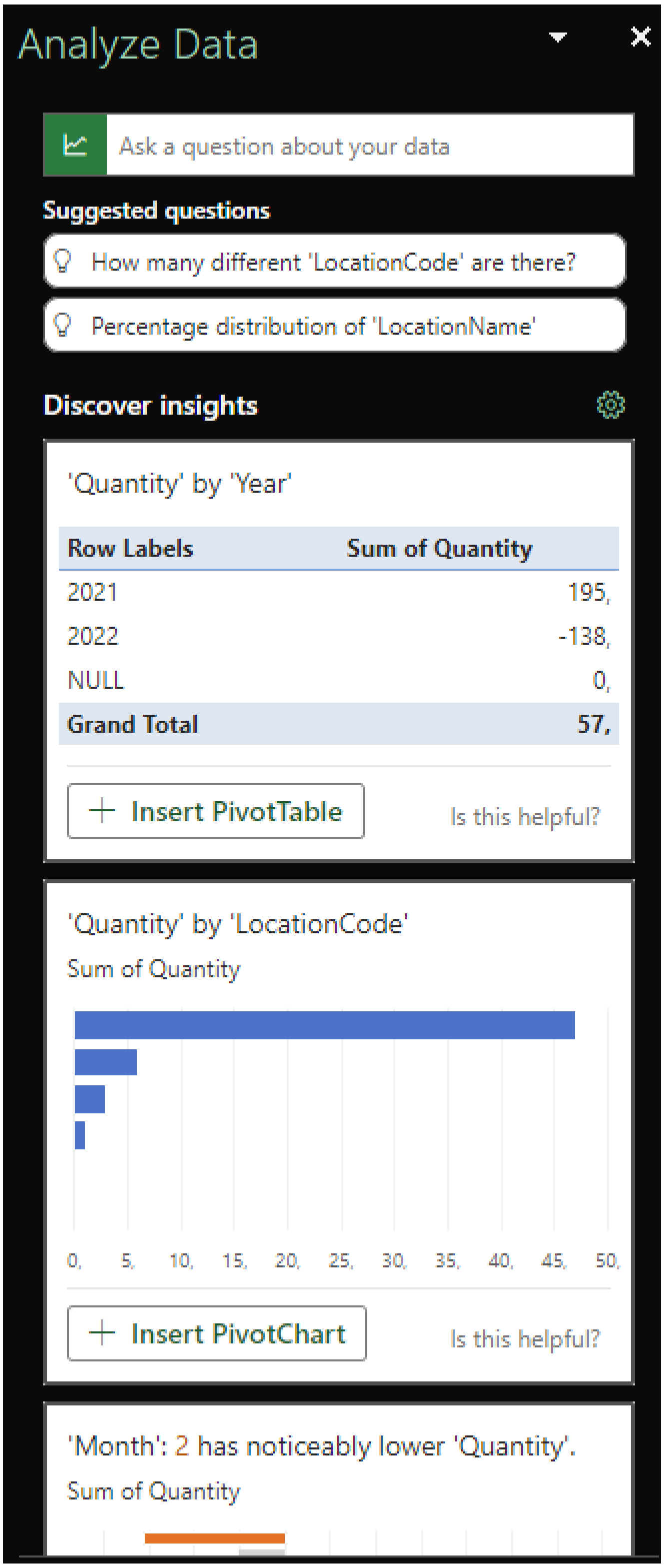


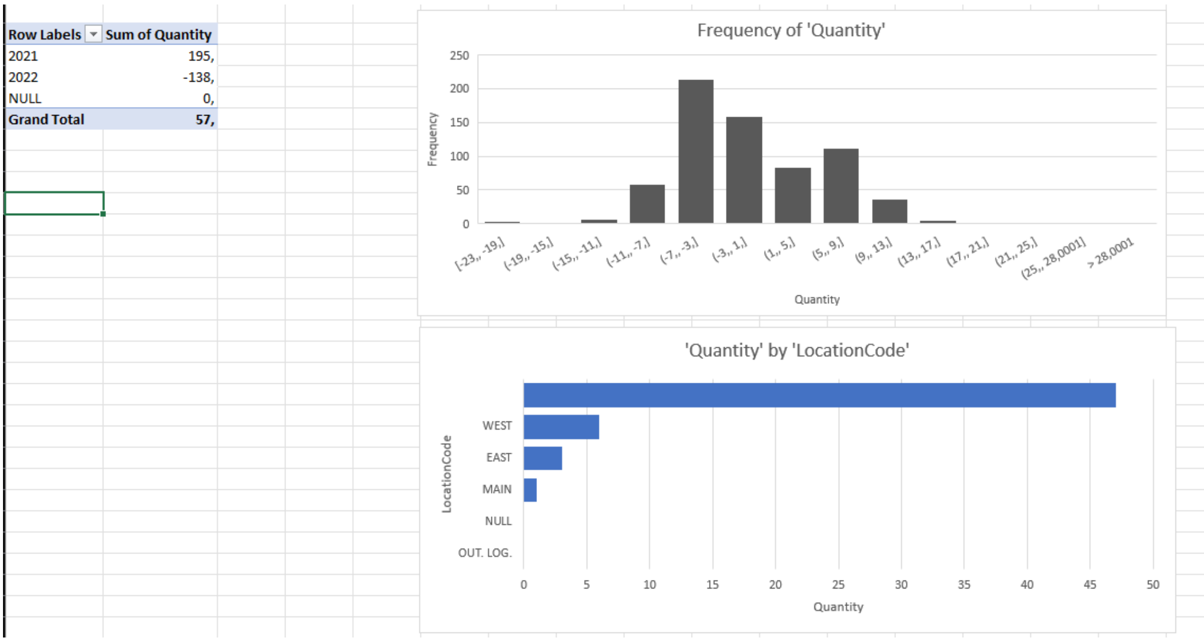
Open the layout in Excel.

In the **Data** worksheet, click: **Analyze Data**:



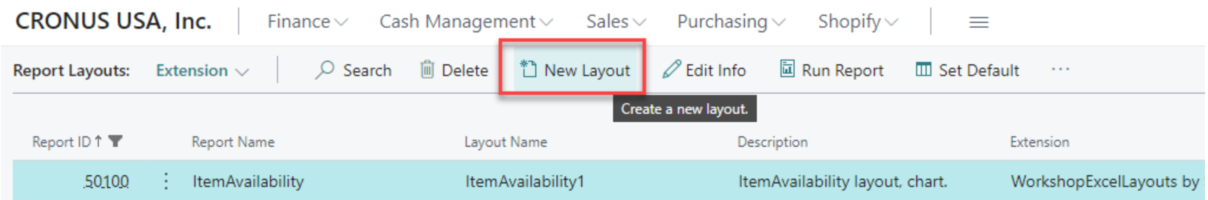
Select one of the insights and click **Insert**:

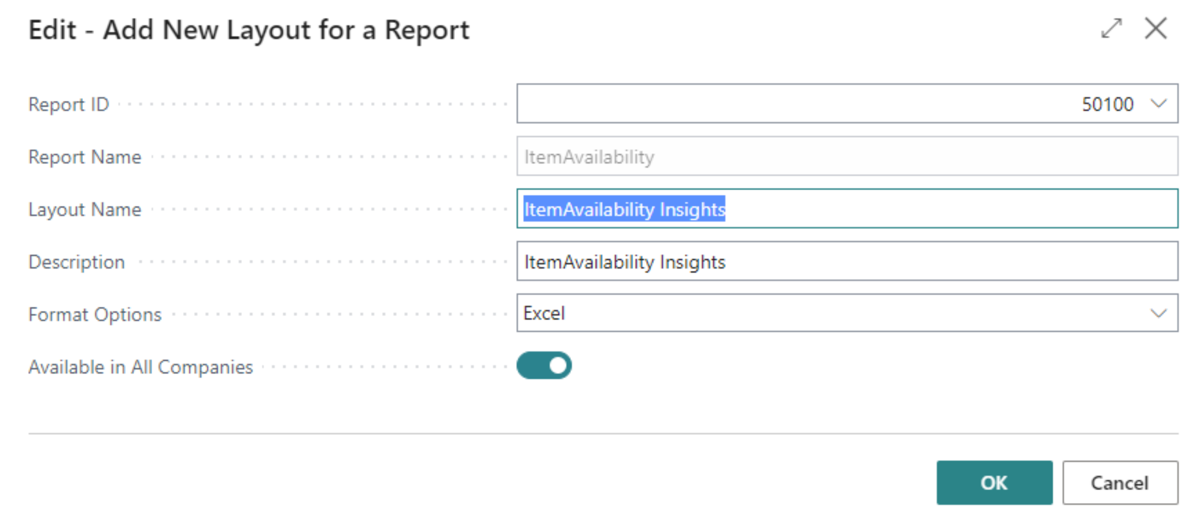

Analyze Data
Ask a question about your data
Suggested questions
How many different 'LocationCode' are there?
Percentage distribution of 'LocationName'
Discover insights
'Quantity' by 'Year'
Row Labels
Sum of Quantity
2021
195,
2022
-138,
NULL
0,
Grand Total
57,
Insert PivotTable
Is this helpful?
'Quantity' by 'LocationCode'
Sum of Quantity
0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50,
Insert PivotChart
Is this helpful?
'Month': 2 has noticeably lower 'Quantity'.
Sum of Quantity



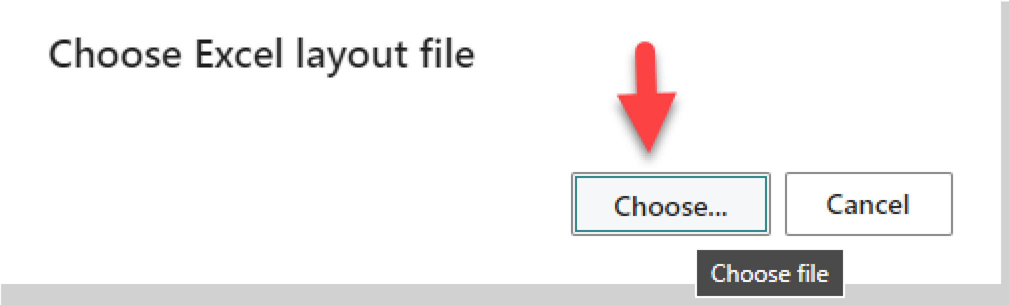
Save the ItemAvailability.xlsx file.

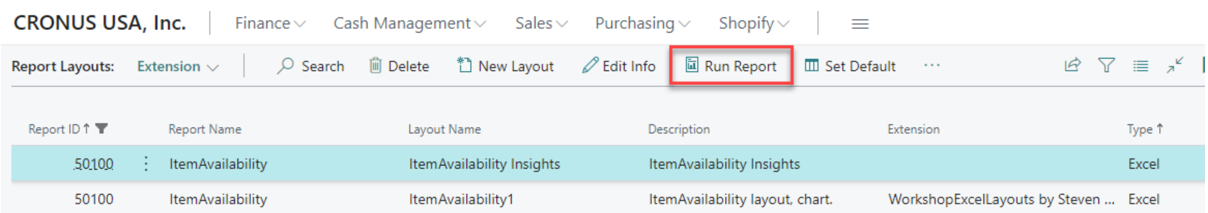
In the Report Layouts page, create a **new Excel layout** for the **ItemAvailability** report:





Select the excel file you saved earlier:





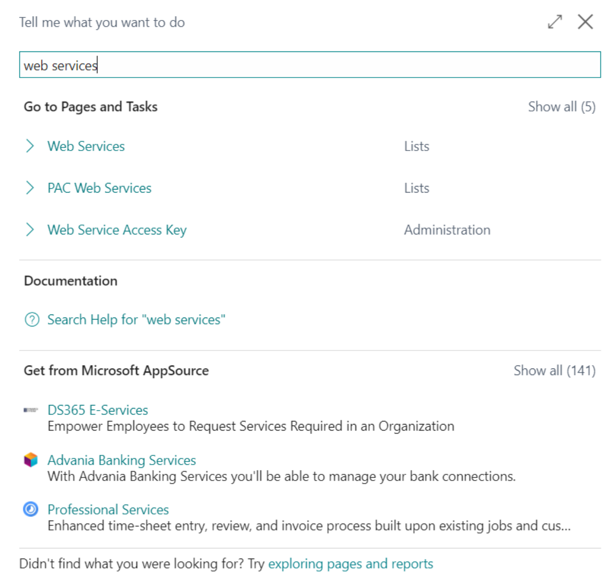
You now have a new layout, with some interesting insights.

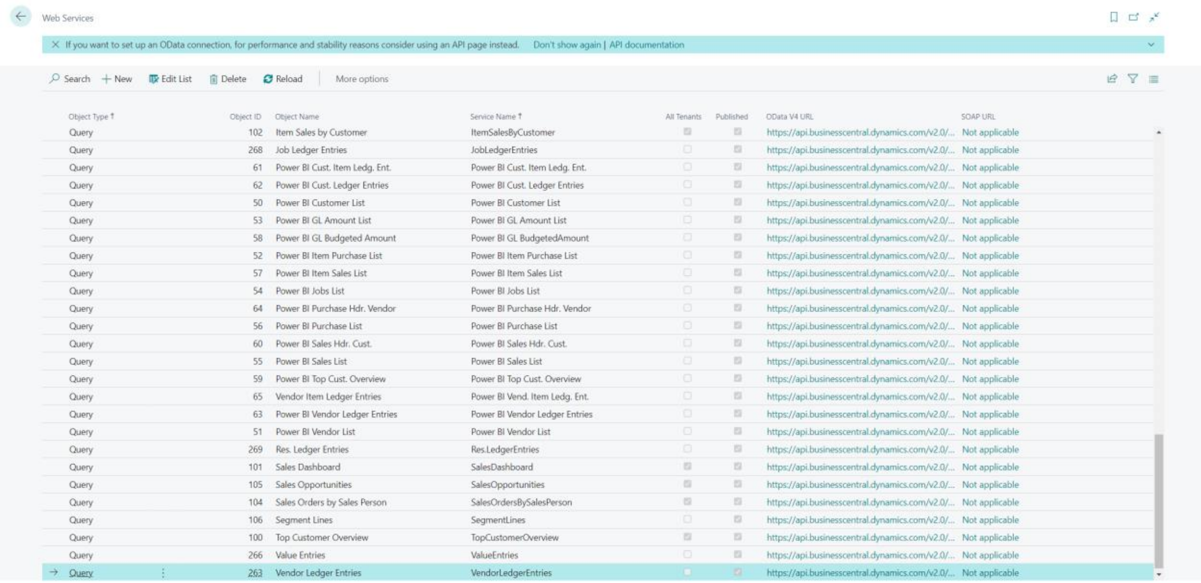
Connecting Excel to Odata (and APIs)

Another option to get data from Dynamics 365 Business Central to the Excel is via **OData** connector in Excel.

With Odata connected we can consume Odata web services and APIs developed by Microsoft or by developers.

To get list of all Web services in **Tell me** type web services and open page with all web services.

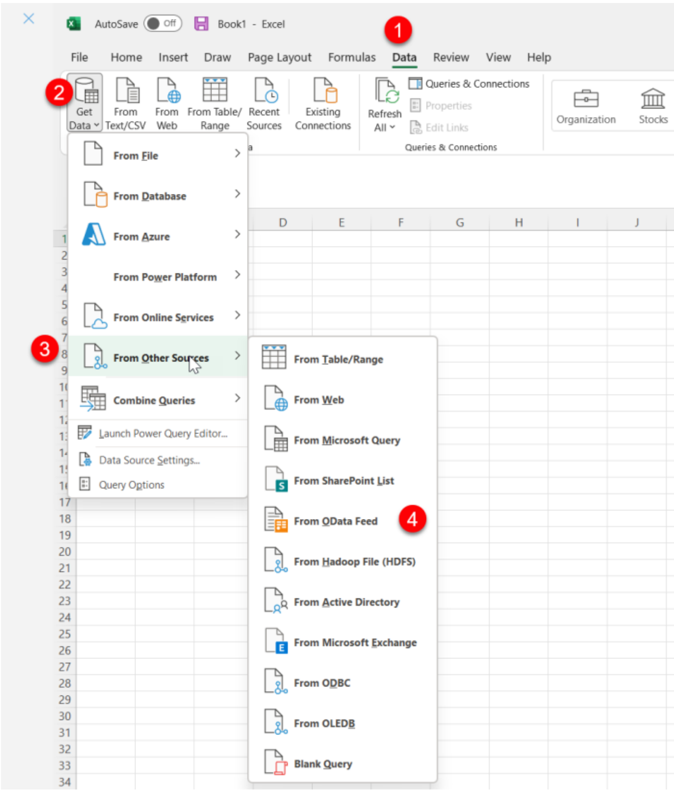




You can copy each published **OData V4 URL**. For this lab, please copy **VendorLedgerEntries** OData V4 URL. https://api.businesscentral.dynamics.com/v2.0/05e59192-a893-44fd-a840-39ca32acace0/DirectionsEmea/ODataV4/Company('CRONUS%20USA%2C%20Inc.')/VendorLedgerEntries

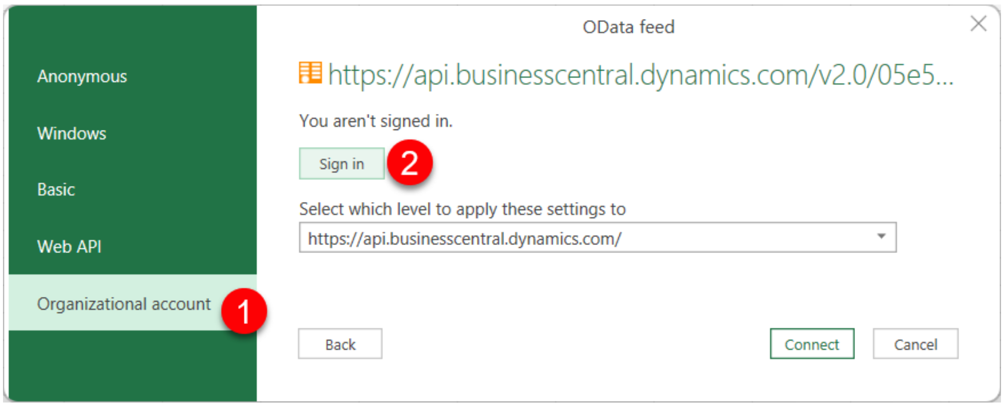
Each OData V4 URL contains path to the Dynamics 365 Busines Central tenant/environment/ company and each specific query or web service.

Now open Excel and click Data -> Get Data -> From Other Sources -> From OData Feed

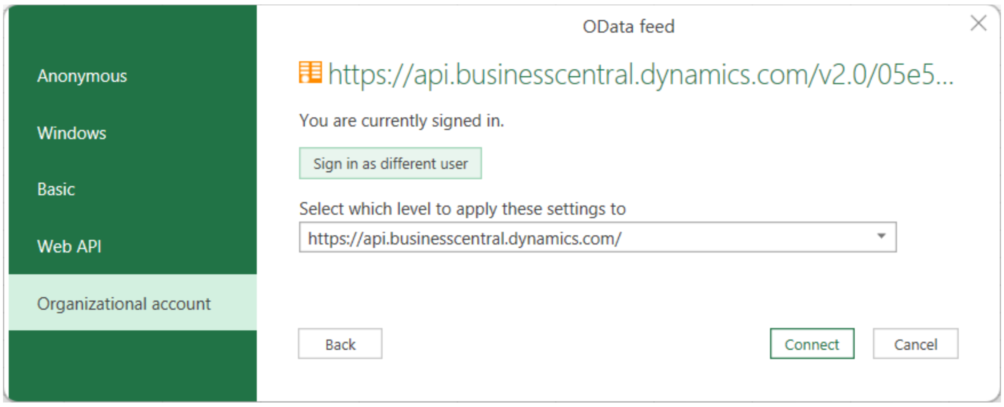


Paste your OData V4 URL into URL field and click OK

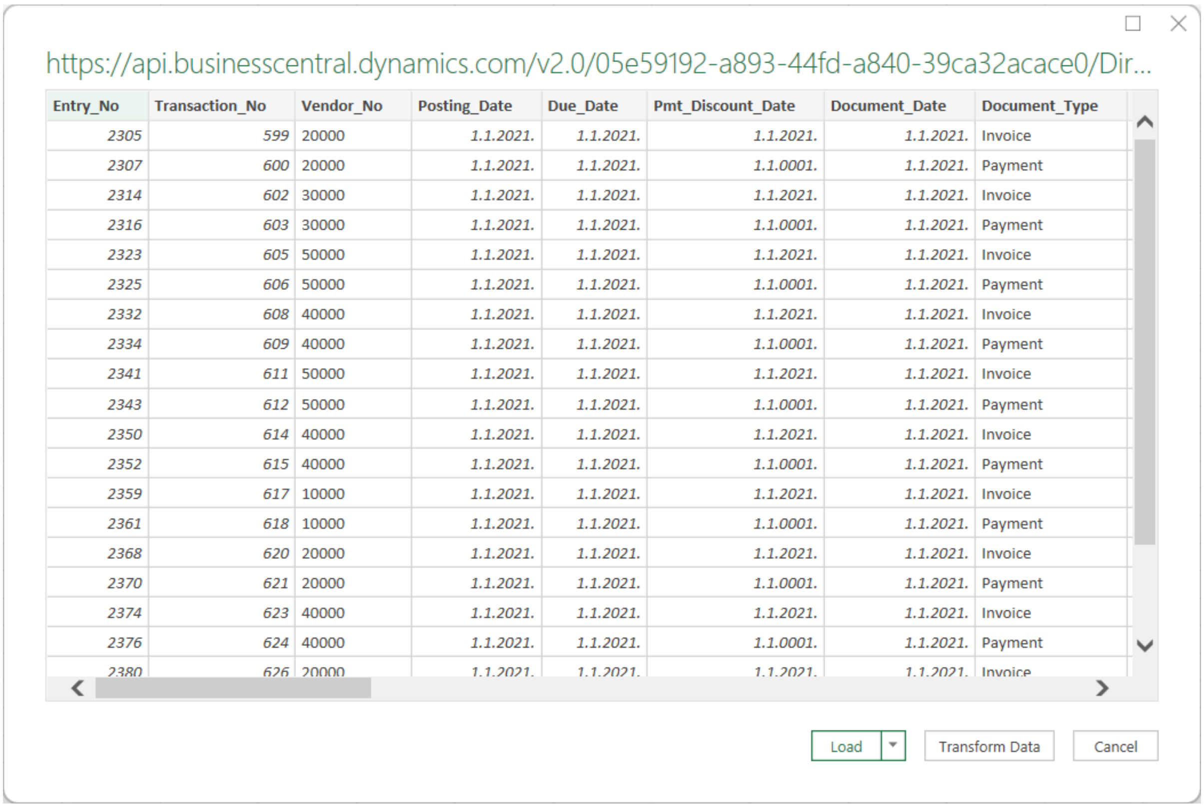
In OData feed window, select **Organization account** and then click **Sign in**. Use your demo credentials that you are using for demo Dynamics 365 Business Central environment.



After successfully signed in, click **Connect.**



In a few seconds you will get **VendorLedger** entries. Now you can either **Load** or **Transform** data.


https://api.businesscentral.dynamics.com/v2.0/05e59192-a893-44fd-a840-39ca32acace0/Dir...

Entry_No	Transaction_No	Vendor_No	Posting_Date	Due_Date	Pmt_Discount_Date	Document_Date	Document_Type
2305	599	20000	1.1.2021.	1.1.2021.	1.1.2021.	1.1.2021.	Invoice
2307	600	20000	1.1.2021.	1.1.2021.	1.1.0001.	1.1.2021.	Payment
2314	602	30000	1.1.2021.	1.1.2021.	1.1.2021.	1.1.2021.	Invoice
2316	603	30000	1.1.2021.	1.1.2021.	1.1.0001.	1.1.2021.	Payment
2323	605	50000	1.1.2021.	1.1.2021.	1.1.2021.	1.1.2021.	Invoice
2325	606	50000	1.1.2021.	1.1.2021.	1.1.0001.	1.1.2021.	Payment
2332	608	40000	1.1.2021.	1.1.2021.	1.1.2021.	1.1.2021.	Invoice
2334	609	40000	1.1.2021.	1.1.2021.	1.1.0001.	1.1.2021.	Payment
2341	611	50000	1.1.2021.	1.1.2021.	1.1.2021.	1.1.2021.	Invoice
2343	612	50000	1.1.2021.	1.1.2021.	1.1.0001.	1.1.2021.	Payment
2350	614	40000	1.1.2021.	1.1.2021.	1.1.2021.	1.1.2021.	Invoice
2352	615	40000	1.1.2021.	1.1.2021.	1.1.0001.	1.1.2021.	Payment
2359	617	10000	1.1.2021.	1.1.2021.	1.1.2021.	1.1.2021.	Invoice
2361	618	10000	1.1.2021.	1.1.2021.	1.1.0001.	1.1.2021.	Payment
2368	620	20000	1.1.2021.	1.1.2021.	1.1.2021.	1.1.2021.	Invoice
2370	621	20000	1.1.2021.	1.1.2021.	1.1.0001.	1.1.2021.	Payment
2374	623	40000	1.1.2021.	1.1.2021.	1.1.2021.	1.1.2021.	Invoice
2376	624	40000	1.1.2021.	1.1.2021.	1.1.0001.	1.1.2021.	Payment
2380	626	20000	1.1.2021.	1.1.2021.	1.1.2021.	1.1.2021.	Invoice

Click **Load** and in few seconds in Excel you will have all Vendor Ledger Entries for Cronus USA company.

With Dynamics 365 Business Central 2022 release wave 2, Microsoft published new APIs that are optimized for reporting in Power BI and Excel.

For more info follow this url: https://learn.microsoft.com/en-us/dynamics365-release-plan/2022wave2/smb/dynamics365-business-central/excel-layouts--new-report-datasets-excel-reporting

To be able to connect to Dynamics 365 Business Central APIs first we need to **combine** and **create API url**.

API URL is similar to OData V4 URL but instead of OData V4 we will point to APIs.

Tenant: M365B667194.onmicrosoft.com, **replace** with your tenant that you can find within your demo account

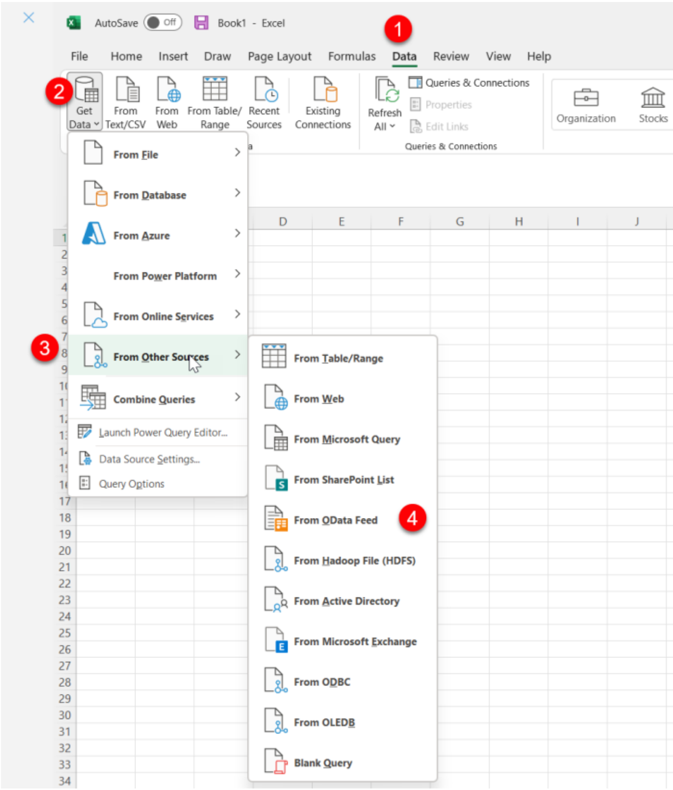
Environment: DirectionsEMEA, or one that you created in our first steps of that lab

microsoft/reportsFinance/beta: path for Microsoft APIs, it is usually <API publisher>/<API group>/<API version>

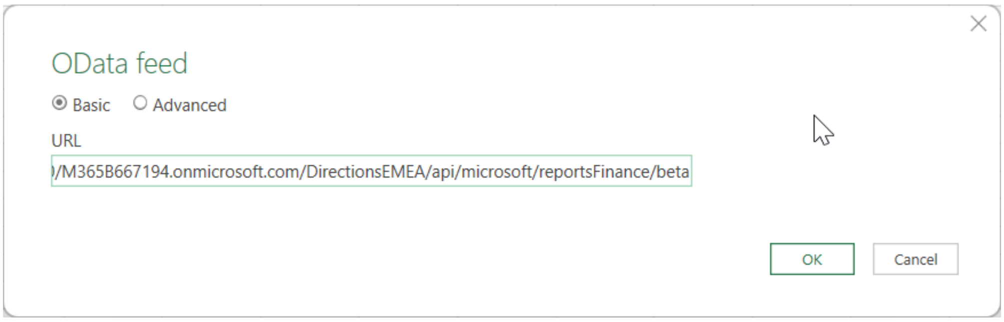
https://api.businesscentral.dynamics.com/v2.0/TENANT/ENVIRONMENT/api/<API PUBLISHER>/<API GROUP>/<API VERSION>

https://api.businesscentral.dynamics.com/v2.0/M365B667194.onmicrosoft.com/DirectionsEMEA/api/microsoft/reportsFinance/beta

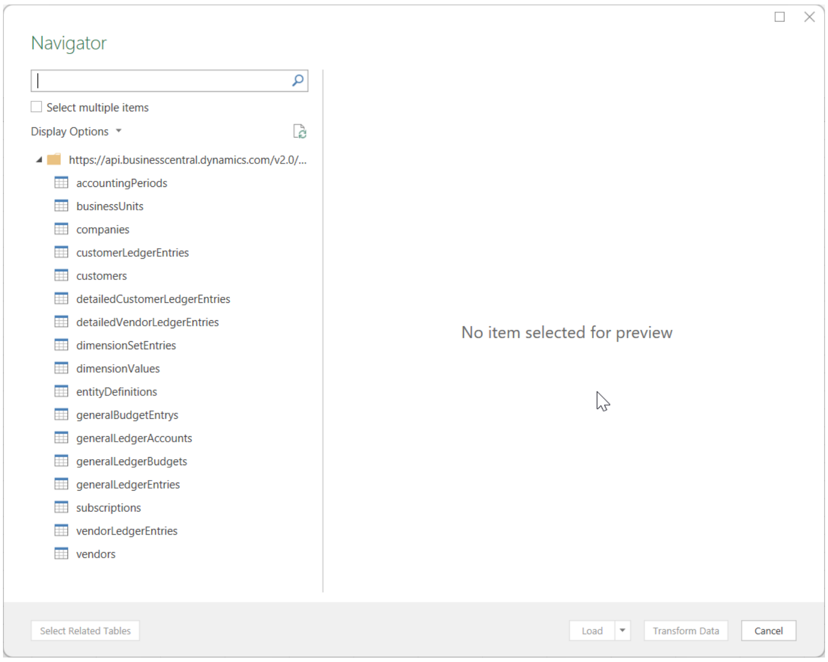
Now when we know how to form URL, lets open **Excel** and click **Data -> Get Data -> From Other Sources -> From OData Feed**



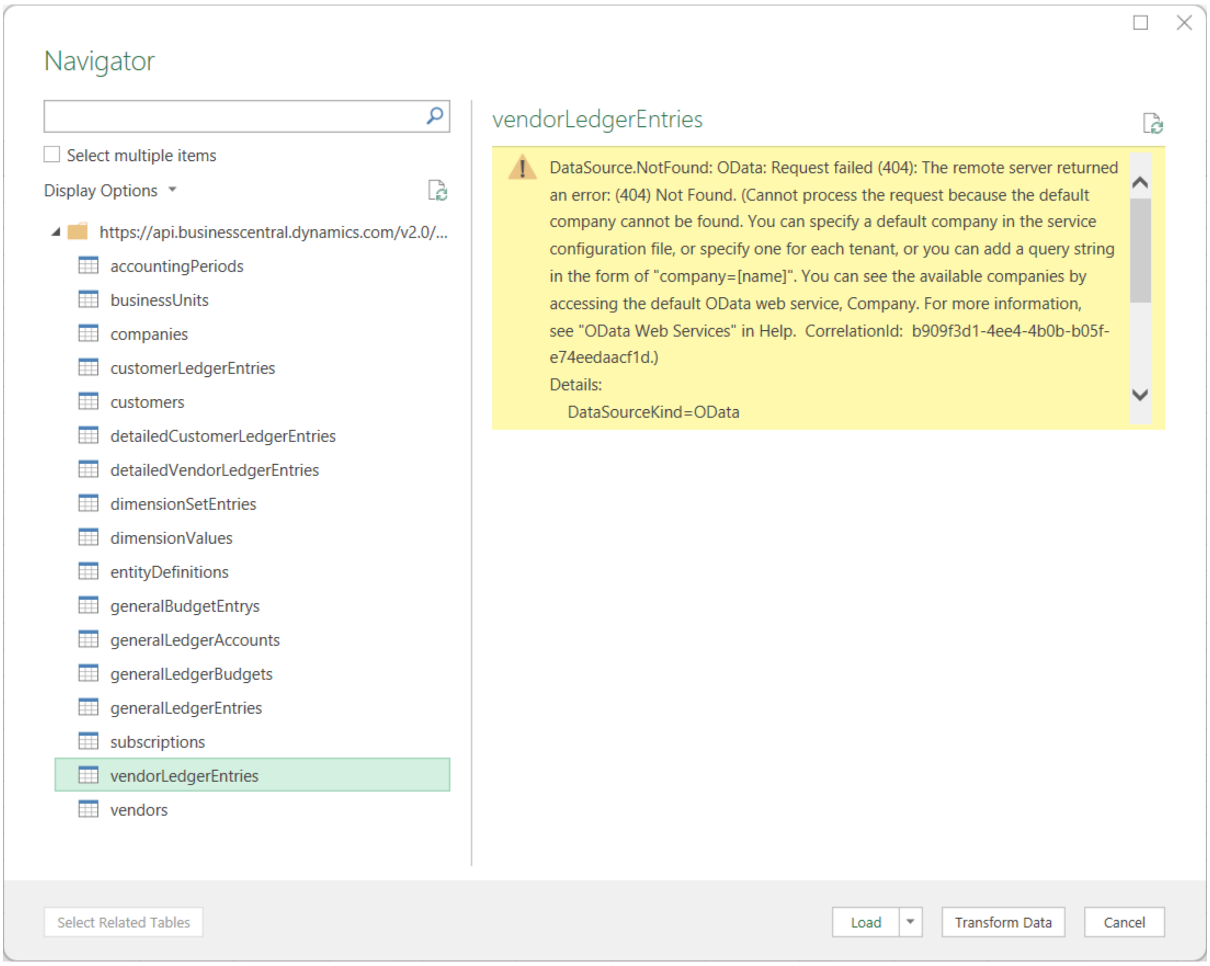
In OData feed window in URL field paste the API url.
https://api.businesscentral.dynamics.com/v2.0/M365B667194.onmicrosoft.com/DirectionsEMEA/api/microsoft/reportsFinance/beta and click **OK**



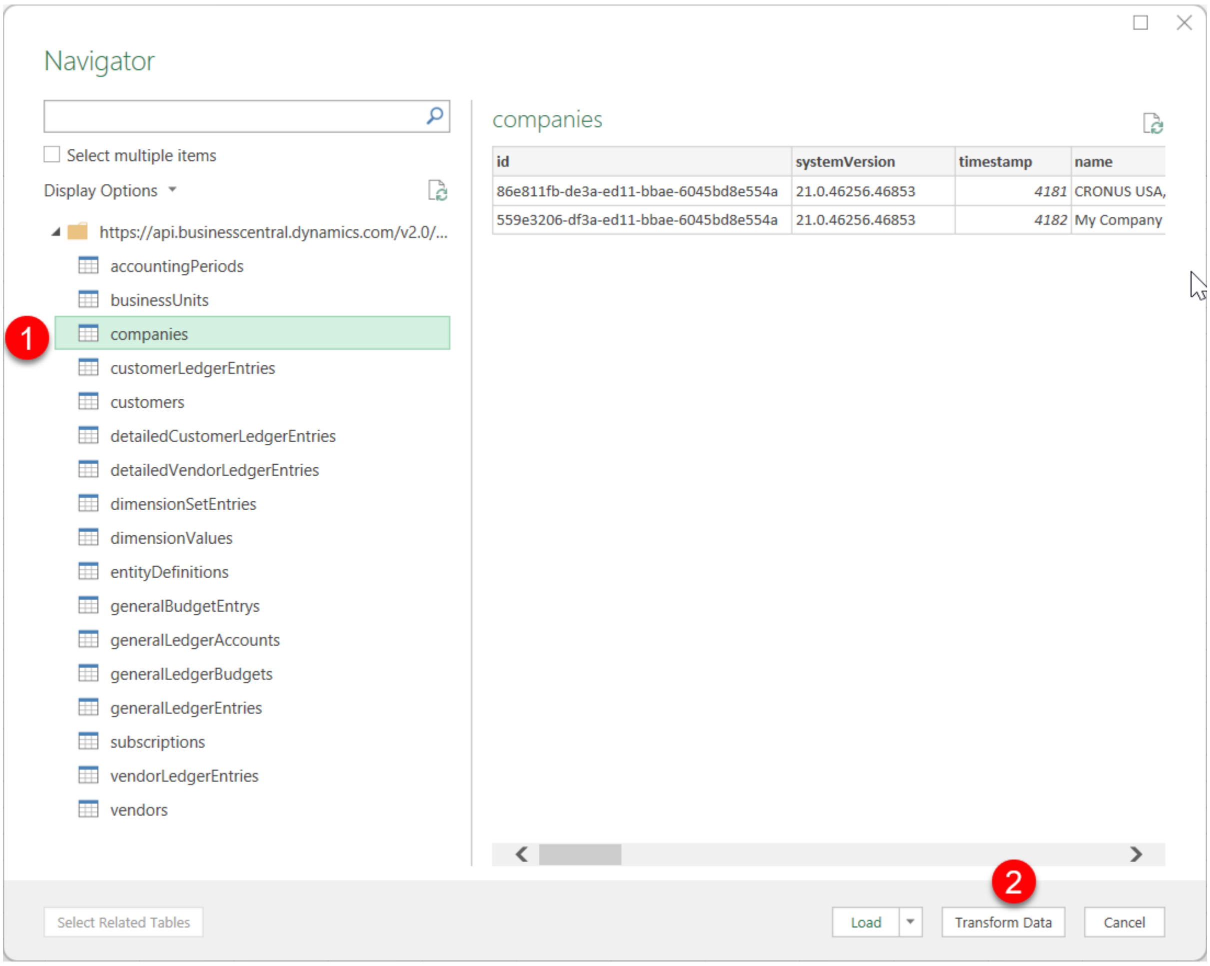
In few seconds you will get all APIs for financial reporting developed by Microsoft.



If you select one of the existing APIs like **VendorLedgerEntries** you will get an error that Odata cant find information about the company.

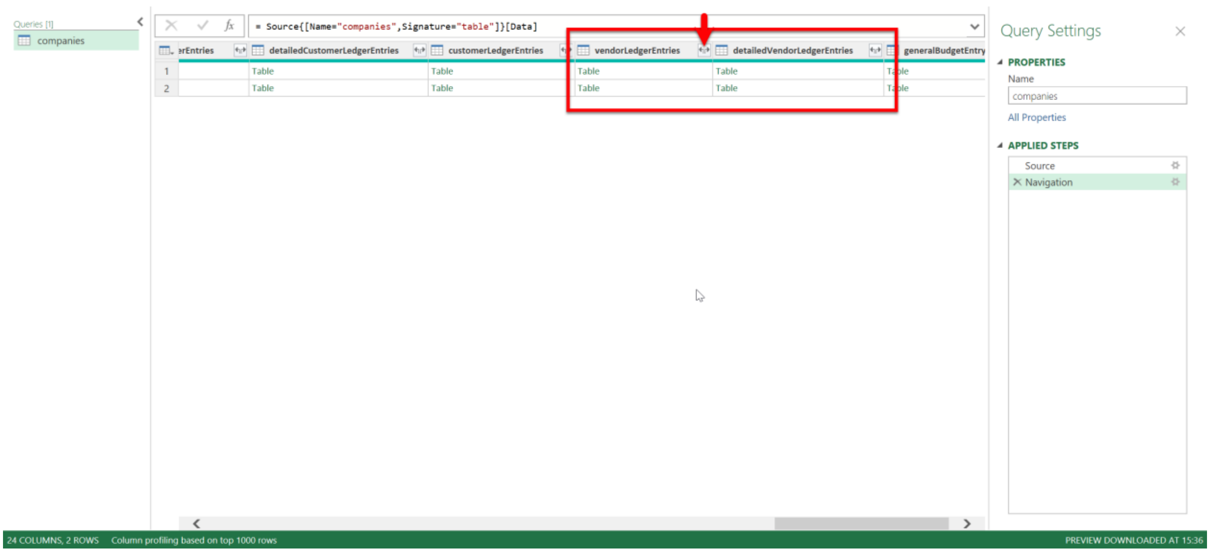


To work around that problem, select **companies** API and click **Transform data.**



In Power Query Editor you can now manipulate with data from companies API and other APIs, as now we have information about company.

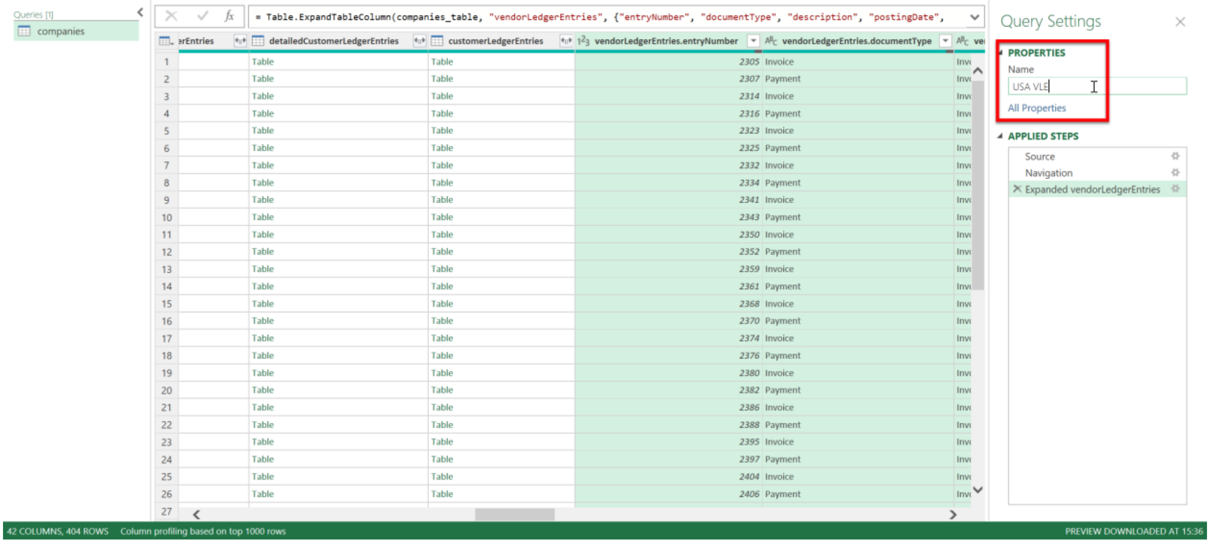
Scroll to the right and find **vendorLedgerEntries** table.



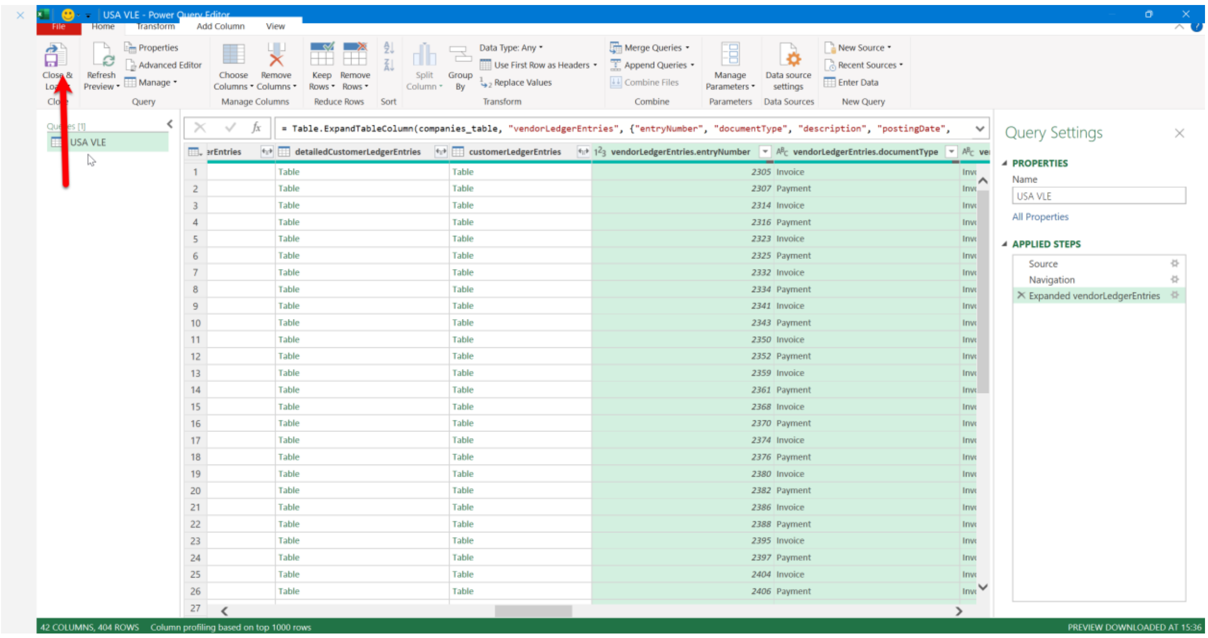
Select the row for Cronus USA and click **Table** for **vendorLedgerEntries.**

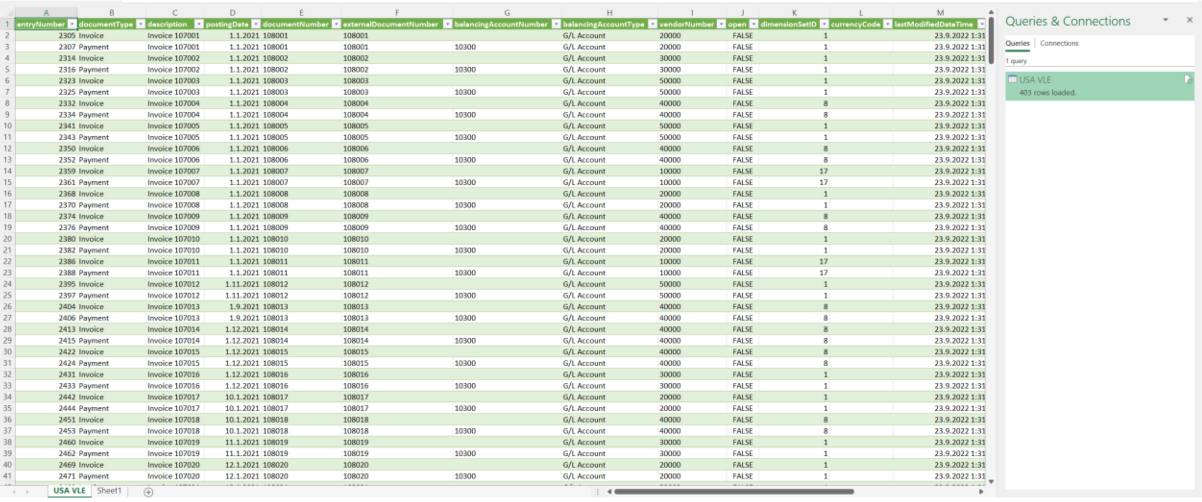
In a few seconds you will get all vendor ledger entries from Cronus USA company but this time using APIs.

You can now remove tables and columns from the **companies** table and rename it to for example USA VLE



The last step is to load data into Excel by clicking on Close & Load





	entryNumber	documentType	description	postingDate	documentNumber	externalDocumentNumber	balancingAccountNumber	balancingAccountType	vendorNumber	open	dimensionSetID	currencyCode	lastModifiedDateTime
2	2305	Invoice	Invoice 107001	1.1.2021	108001	108001		G/L Account	20000	FALSE	1		23.9.2022 1:31
3	2307	Payment	Invoice 107001	1.1.2021	108001	108001	10300	G/L Account	20000	FALSE	1		23.9.2022 1:31
4	2314	Invoice	Invoice 107002	1.1.2021	108002	108002		G/L Account	30000	FALSE	1		23.9.2022 1:31
5	2316	Payment	Invoice 107002	1.1.2021	108002	108002	10300	G/L Account	30000	FALSE	1		23.9.2022 1:31
6	2323	Invoice	Invoice 107003	1.1.2021	108003	108003		G/L Account	50000	FALSE	1		23.9.2022 1:31
7	2325	Payment	Invoice 107003	1.1.2021	108003	108003	10300	G/L Account	50000	FALSE	1		23.9.2022 1:31
8	2332	Invoice	Invoice 107004	1.1.2021	108004	108004		G/L Account	40000	FALSE	8		23.9.2022 1:31
9	2334	Payment	Invoice 107004	1.1.2021	108004	108004	10300	G/L Account	40000	FALSE	8		23.9.2022 1:31
10	2341	Invoice	Invoice 107005	1.1.2021	108005	108005		G/L Account	50000	FALSE	1		23.9.2022 1:31
11	2343	Payment	Invoice 107005	1.1.2021	108005	108005	10300	G/L Account	50000	FALSE	1		23.9.2022 1:31
12	2350	Invoice	Invoice 107006	1.1.2021	108006	108006		G/L Account	40000	FALSE	8		23.9.2022 1:31
13	2352	Payment	Invoice 107006	1.1.2021	108006	108006	10300	G/L Account	40000	FALSE	8		23.9.2022 1:31
14	2359	Invoice	Invoice 107007	1.1.2021	108007	108007		G/L Account	10000	FALSE	17		23.9.2022 1:31
15	2361	Payment	Invoice 107007	1.1.2021	108007	108007	10300	G/L Account	10000	FALSE	17		23.9.2022 1:31
16	2368	Invoice	Invoice 107008	1.1.2021	108008	108008		G/L Account	20000	FALSE	1		23.9.2022 1:31
17	2370	Payment	Invoice 107008	1.1.2021	108008	108008	10300	G/L Account	20000	FALSE	1		23.9.2022 1:31
18	2374	Invoice	Invoice 107009	1.1.2021	108009	108009		G/L Account	40000	FALSE	8		23.9.2022 1:31
19	2376	Payment	Invoice 107009	1.1.2021	108009	108009	10300	G/L Account	40000	FALSE	8		23.9.2022 1:31
20	2380	Invoice	Invoice 107010	1.1.2021	108010	108010		G/L Account	20000	FALSE	1		23.9.2022 1:31
21	2382	Payment	Invoice 107010	1.1.2021	108010	108010	10300	G/L Account	20000	FALSE	1		23.9.2022 1:31
22	2386	Invoice	Invoice 107011	1.1.2021	108011	108011		G/L Account	10000	FALSE	17		23.9.2022 1:31
23	2388	Payment	Invoice 107011	1.1.2021	108011	108011	10300	G/L Account	10000	FALSE	17		23.9.2022 1:31
24	2395	Invoice	Invoice 107012	1.11.2021	108012	108012		G/L Account	50000	FALSE	1		23.9.2022 1:31
25	2397	Payment	Invoice 107012	1.11.2021	108012	108012	10300	G/L Account	50000	FALSE	1		23.9.2022 1:31
26	2404	Invoice	Invoice 107013	1.9.2021	108013	108013		G/L Account	40000	FALSE	8		23.9.2022 1:31
27	2406	Payment	Invoice 107013	1.9.2021	108013	108013	10300	G/L Account	40000	FALSE	8		23.9.2022 1:31
28	2413	Invoice	Invoice 107014	1.12.2021	108014	108014		G/L Account	40000	FALSE	8		23.9.2022 1:31
29	2415	Payment	Invoice 107014	1.12.2021	108014	108014	10300	G/L Account	40000	FALSE	8		23.9.2022 1:31
30	2422	Invoice	Invoice 107015	1.12.2021	108015	108015		G/L Account	40000	FALSE	8		23.9.2022 1:31
31	2424	Payment	Invoice 107015	1.12.2021	108015	108015	10300	G/L Account	40000	FALSE	8		23.9.2022 1:31
32	2431	Invoice	Invoice 107016	1.12.2021	108016	108016		G/L Account	30000	FALSE	1		23.9.2022 1:31
33	2433	Payment	Invoice 107016	1.12.2021	108016	108016	10300	G/L Account	30000	FALSE	1		23.9.2022 1:31
34	2442	Invoice	Invoice 107017	10.1.2021	108017	108017		G/L Account	20000	FALSE	1		23.9.2022 1:31
35	2444	Payment	Invoice 107017	10.1.2021	108017	108017	10300	G/L Account	20000	FALSE	1		23.9.2022 1:31
36	2451	Invoice	Invoice 107018	10.1.2021	108018	108018		G/L Account	40000	FALSE	8		23.9.2022 1:31
37	2453	Payment	Invoice 107018	10.1.2021	108018	108018	10300	G/L Account	40000	FALSE	8		23.9.2022 1:31
38	2460	Invoice	Invoice 107019	11.1.2021	108019	108019		G/L Account	30000	FALSE	1		23.9.2022 1:31
39	2462	Payment	Invoice 107019	11.1.2021	108019	108019	10300	G/L Account	30000	FALSE	1		23.9.2022 1:31
40	2469	Invoice	Invoice 107020	12.1.2021	108020	108020		G/L Account	20000	FALSE	1		23.9.2022 1:31
41	2471	Payment	Invoice 107020	12.1.2021	108020	108020	10300	G/L Account	20000	FALSE	1		23.9.2022 1:31

If you need you can always develop your own custom query API and publish it to be visible in API connector for Dynamics 365 Business Central.