**Interactive Excel Dashboard and how to automate with Python: A Beginner’s Guide**

[[Nelly TCHIENGUE](https://medium.com/@nellytchiengue?source=post_page-----5d760307ca67--------------------------------)](https://medium.com/@nellytchiengue?source=post_page-----5d760307ca67--------------------------------)

[Nelly TCHIENGUE](https://medium.com/@nellytchiengue?source=post_page-----5d760307ca67--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F7db085d9906b&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40nellytchiengue%2Finteractive-excel-dashboard-and-how-to-automate-with-python-a-beginners-guide-5d760307ca67&user=Nelly+TCHIENGUE&userId=7db085d9906b&source=post_page-7db085d9906b----5d760307ca67---------------------post_header-----------)

11 min read

·

Aug 30

54

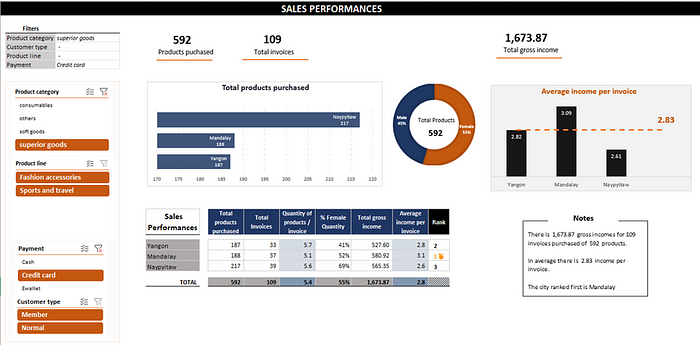


Photo by [Towfiqu B](https://unsplash.com/fr/@towfiqu999999?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)arbhuiya on [Unsplash](https://unsplash.com/fr/photos/nApaSgkzaxg?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)

**Introduction**

[**Excel**](https://www.microsoft.com/en-us/microsoft-365/excel)is a powerful tool for data analysis and manipulation. However, it can be time-consuming to perform repetitive tasks manually. [**Python**](https://www.python.org/)is a programming language that can be used to automate tasks in Excel and on the other hand, to manage data pipeline/analysis. Combining **Excel and Python** saves you time and improves the accuracy of your work.

This article describes how to create and automate an interactive **Dashboard**using Excel and Python.



[sales\_performance.xlsx](https://github.com/nellytchiengue/sales_performance_dashboard/blob/main/sales_performance.xlsx)

With no VBA or Power Query code added.

The data source is provided by the [supermarket sales Kaggle dataset](https://www.kaggle.com/datasets/aungpyaeap/supermarket-sales).

We will describe here the method used step by step in 3 parts :

* Excel: Dashboard creation
* Python: Data wrangling and transformation
* Outlook: Automation

Technical Environments:

* **Python** 3.8.8
* **Excel** Microsoft® Excel® for Microsoft 365 MSO (Version 2307 Build 16.0.16626.20170) 64-bit
* **Windows** 11 Home

**Excel: Dashboard creation**

In this section, we will show how to create a user-friendly and dynamic Excel report without VBA along the sub-sections below:

* Excel file components
* Excel features & tricks used

The important thing to know is that the dynamism relies on pivot tables, related slicers, and some Excel formulas.

**Excel file components**

The objective of the Dashboard is to display sales historical record of 3 branches for 3 months of data in 3 supermarkets.

The main Excel sheet “Sales Performance” shows some indicators in a table and associated charts. With the help of slicers, which are linked to unique data source from the “data” sheet, all visuals can be filtered in one click.

Basically, we need 4 worksheets for :

* data source table
* pivot tables
* calculations
* Sales Performance Dashboard

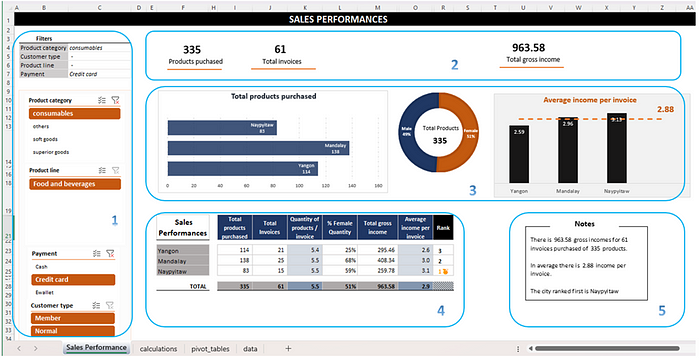
**“data” sheet** — contains a data table ( Excel Table object) whose data comes from a [Kaggle dataset](https://www.kaggle.com/datasets/aungpyaeap/supermarket-sales) with some transformation performed in Python scripts described later. This is the only data source for the report.

**“pivot\_tables” sheet —**contains the pivot tables created with the above data source.

**“calculations” sheet —**handles some calculated variables and charts used for the Dashboard.

**“Sales Performance” sheet —**This handles all the visuals (figures, Key Performance Indicators, charts, and filters)

**Excel features & tricks**



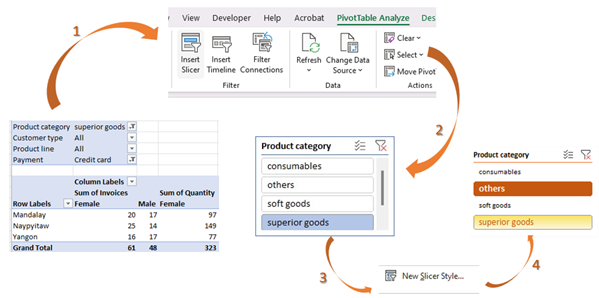
[sales\_performance.xlsx](https://github.com/nellytchiengue/sales_performance_dashboard/blob/main/sales_performance.xlsx)

In this section, we will go through the main Excel features and the tips used for the Dashboard:

1. Slicers
2. Indicators Panel
3. Table
4. Notes
5. Charts ( Bars & Pie)

**Slicers**

An Excel **slicer** is a visual filter that you can use to quickly and easily filter the data in a table or **PivotTable**. Slicers are available in Excel 2010 and later versions.



By following the tips below, you can use slicers to create effective and visually appealing data visualizations.

To use a slicer, you first need a table or PivotTable containing the data you wish to filter. Then follow the steps below:

1. Click anywhere in the table or PivotTable.

2. In the Insert tab, in the Tables group, click on Slicer.

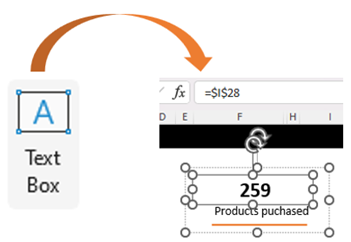
3. In the Insert Slicers dialog box, select the checkboxes of the fields that you wish to display in the slicer, then click OK.

A slicer will be created for each of the fields that you have selected. You can then use the slicer to filter the data in the table or PivotTable.

To go further and style the slicer I recommend this [YouTube video](https://www.youtube.com/watch?app=desktop&v=KEZzGu1MIY4).

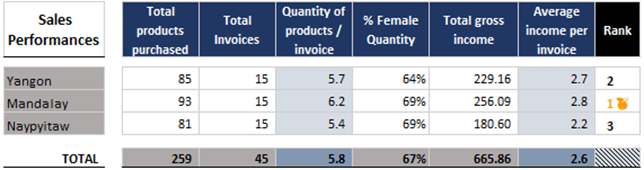
**Indicators Panel**

The key indicators panel at the top of the “Sales Performance” sheet is built with **Text Box**objects.



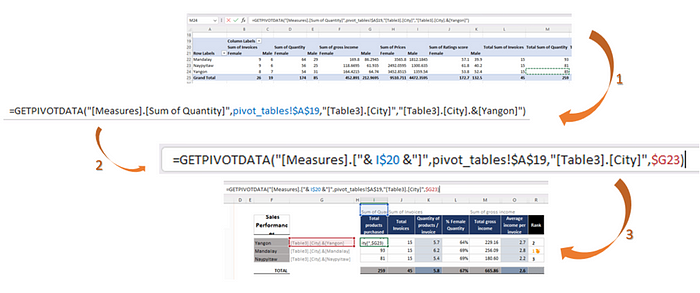
The value of the Text box can change automatically according to the value of the linked cell (cell I28 in the screen above), which is itself updated via the pivot table.

**Table**

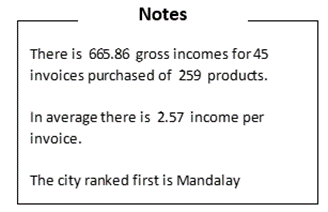


The trick is to dynamically retrieve data from the pivot table (Table 3) using the [**GETPIVOTDATA**](https://learn.microsoft.com/en-us/office/vba/api/excel.pivottable.getpivotdata)function.

=GETPIVOTDATA("[Measures].["& I$20 &"]",pivot\_tables!$A$19,"[Table3].[City]",$G23)



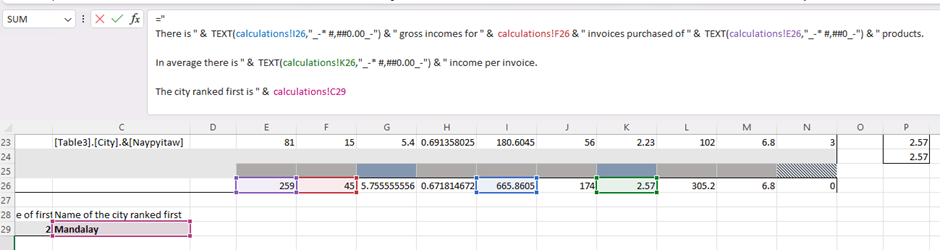
**Notes**



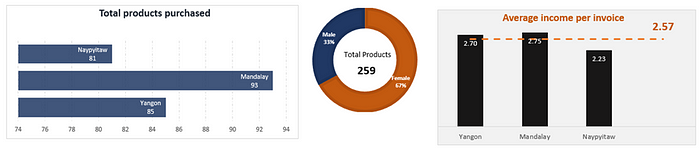
This “Notes” represent a cell containing concatenate strings and numbers.

It is quite simple to do, but you need to use a few tricks to display the numbers correctly. In the code below, the [TEXT](https://support.microsoft.com/en-gb/office/text-function-20d5ac4d-7b94-49fd-bb38-93d29371225c)function manages the format (example: **“\_-\* #,##0.00\_-”**) we need to use for each number to be concatenated.

="  
There is " & TEXT(calculations!I26,"\_-\* #,##0.00\_-") & " gross incomes for " & calculations!F26 & " invoices purchased of " & TEXT(calculations!E26,"\_-\* #,##0\_-") & " products.   
  
In average there is " & TEXT(calculations!K26,"\_-\* #,##0.00\_-") & " income per invoice.   
  
The city ranked first is " & calculations!C29



**Charts ( Bars & Pie)**

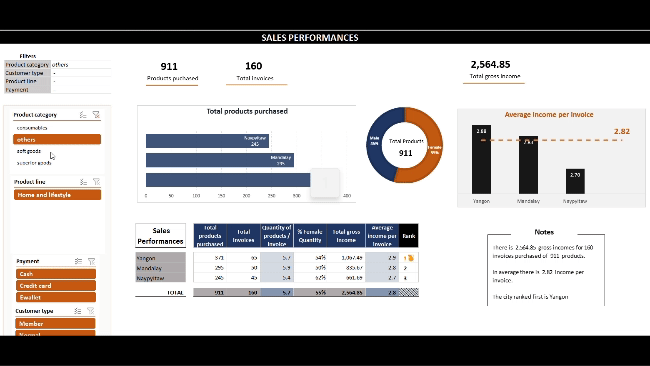


These charts were produced using data from the table described above and Excel styles for the Bar and Pie objects.

You can explore all the features implemented in greater detail in the [sales\_performance.xlsx](https://github.com/nellytchiengue/sales_performance_dashboard/blob/main/sales_performance.xlsx) file, available [here](https://github.com/nellytchiengue/sales_performance_dashboard/blob/main/sales_performance.xlsx).

**Dashboard restitution**

Once we’ve applied all Excel features, we can use and explore the Dashboard dynamically:



[sales\_performance.xlsx](https://github.com/nellytchiengue/sales_performance_dashboard/blob/main/sales_performance.xlsx)

**Python: Data wrangling and transformation**



Photo by [Hitesh Choudhary](https://unsplash.com/fr/@hiteshchoudhary?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText) on [Unsplash](https://unsplash.com/fr/photos/pMnw5BSZYsA?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)

**Python** is a popular choice for data wrangling and transformation tasks in a variety of fields, including data science, machine learning, and business intelligence because it is:

*Powerful*: Python has a large library of modules and functions that can be used for data manipulation.

*Extensible*: Python is an open-source language, which means that many third-party libraries can be used to extend its functionality.

*Fast*: Python is a relatively fast language, making it suitable for processing large datasets.

*Easy to learn*: Python is a relatively easy language to learn, making it a good choice for beginners.

Here are some examples of how Python can be used for data wrangling and transformation:

**Cleaning data**: Python can be used to remove errors, duplicates, and missing values from data.

**Transforming data**: Python can be used to change the format of data, such as converting text to numbers or merging different data sets.

**Integrating data**: Python can be used to combine data from different sources into a single data set.

In the next sections, we will show you how to:

* Collect data
* Transform data
* Automate tasks

for the production of our Excel Dashboard by using Python scripts.

**Python modules**

[Pandas](https://pandas.pydata.org/docs/index.html)— is a powerful tool for data manipulation and analysis. It is easy to learn and use. Pandas DataFrames are the most versatile data structures in Pandas. They are similar to Excel spreadsheets, but they are more powerful and flexible. DataFrames can be used to store and manipulate data of any type, including numerical, categorical, and time series data. Code to install :

pip install pandas

[xlwings](https://www.xlwings.org/)— is a Python library that allows you to control Excel from Python. It provides a simple and intuitive API that makes it easy to automate tasks in Excel. Code to install :

pip install xlwings

[os](https://www.geeksforgeeks.org/os-module-python-examples/) — provides functions for interacting with the operating system. It comes under Python’s standard utility modules. This module provides a portable way of using operating system-dependent functionality. Code to install :

pip install os

[win32com](https://www.geeksforgeeks.org/os-module-python-examples/)— provides a way to interact with COM objects using Python. It doesn’t provide the functions to manipulate an Excel spreadsheet directly but rather provides you with a function to acquire an Excel spreadsheet object. From this object, you can then manipulate a spreadsheet in an object-oriented fashion. Code to install :

pip install win32com

**Data Collection**

After downloading the CSV file from [Kaggle](https://www.kaggle.com/datasets/aungpyaeap/supermarket-sales), we import the dataset using the pandas module:

import pandas as pd  
file\_name = "supermarket\_sales - Sheet1.csv"  
df\_data\_raw = pd.read\_csv(file\_name)

We can verify the data:

# Exploratory  
df\_data\_raw.head()  
df\_data\_raw.describe( include = "all")

**Data transformation**

The original dataset contains 1,000 invoices. We want to aggregate them by City:

# Group by  
df\_data = df\_data\_raw.groupby(['Branch','City', 'Customer type', 'Gender', 'Product line', 'Paymenbe usedt']).agg({'Invoice ID': 'count', 'Quantity': 'sum', "Total": 'sum', "gross income": 'sum', "Rating": 'sum'}).reset\_index()

We have added a new column to obtain a “category” depending on the Product line:

# function to be used to create a new column with the category of Product   
def get\_product\_category(product\_line):  
 if product\_line == "Food and beverages":  
 return "consumables"  
 elif product\_line == "Health and beauty" or product\_line == "Electronic accessories":  
 return "soft goods"   
 elif product\_line == "Fashion accessories" or product\_line == "Electronic accessories" or product\_line == "Sports and travel":  
 return "superior goods"  
 else:  
 return "others"  
# creation of new column depending on "Product line" columns value  
df\_data["Product category"] = df\_data["Product line"].apply(get\_product\_category)

The “df\_data” DataFrame contains the final dataset for our report. In the next section, we will look at how to transfer the values to Excel.

**Automation**

Our reporting is about sales data. In the real world of business, these data would have to be updated and analyzed perhaps monthly, weekly or even daily.

To update the report, we need to

1. Retrieve the raw data and apply the modification as described above.

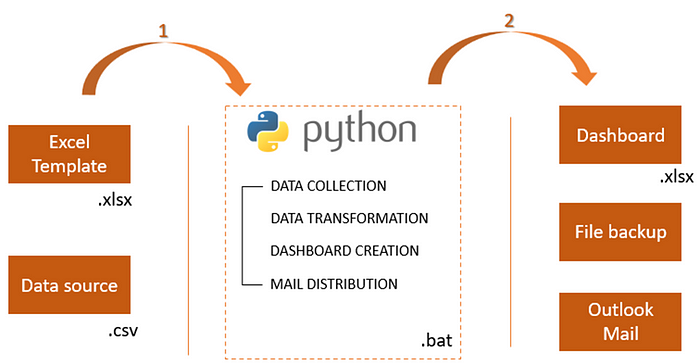
2. Clear the “data’ sheet and insert new data in it place.

3. In the Excel file, [refresh all] the data of the workbook.

4. And [saves as] a new file.

It can become tedious and time-consuming to repeat the same actions manually every time for dozens of daily or weekly reports.

That is where **automation with Python**comes in.



Processing of updating the Excel report and sharing it

Dashboard production will be automated in 4 main stages:

1. Use of an Excel template.

2. Interacting with the Xlwings Python module.

3. Using an Outlook mail template.

4. Processing all in the Batch file (.bat).

**Excel Template**

We will use the previous Excel report built in the section **Excel: Dashboard creation** and use it as a template.

The aim is to reuse the same visuals and formulas, so simply refresh the data source and save the workbook as a new workbook.

**Python module Xlwings**

The Xlwings module performs the following tasks:

* open the Excel template and paste the updated data source
* Refresh all data and save it as a new file: the Dashboard is updated

import xlwings as xw  
  
# Open existing excel template  
wb = xw.Book("Template\\" + template\_filename + ".xlsx")  
wb.sheets[data\_sheet\_name].activate()  
ws = wb.sheets[data\_sheet\_name]  
  
# paste the values of the dataframe from B2 cell  
ws.range("B2").value = df\_data.values  
  
report\_file\_name = template\_filename +"\_" + fileExtension\_name + ".xlsx"  
  
# put the first sheet active before closing the report  
wb.sheets[0].activate()  
  
# save as the excel workbook: a next file is created  
wb.save(report\_file\_name)  
  
# update pivot table -> "Refresh All" button  
# Refresh all data connections.  
wb.RefreshAll()  
  
# close the Excel workbook  
wb.close()

**Outlook mail**

We can prepare Outlook mail using a .oft template with the win32com module.

The “displayEmail” function in the Python script below takes two arguments: the path to the Outlook template file and the list of files to be attached to the email. The script then uses the win32com.client module to create an Outlook email message from the template file and attach the specified files. The email message is then displayed and saved in the Outlook draft folder.

import win32com.client  
def displayEmail (email\_template\_path, attachmentPaths):  
  
 print ( "\n>>> START- displayEmail ")  
  
 # Create an email message from .oft template  
 obj = win32com.client.Dispatch("Outlook.Application")  
 report\_mail = obj.CreateItemFromTemplate (email\_template\_path)  
 report\_mail.display()  
  
 # Add attachments  
 for attachment in attachmentPaths :  
 print ("attachment = "+ attachment)  
 report\_mail.Attachments.Add(Source=attachment)  
   
 # open the mail  
 report\_mail.display()  
  
 # save it in outlook draft folder  
 report\_mail.save()  
  
 print ( "\n>>> END - displayEmail ")  
  
# the list of files to attached into email  
  
attachment\_files = list()   
attachment\_files.append( project\_folder + report\_file\_name )   
  
# an oft template should be created before and save in "template" folder  
emailTemplateName = "sales\_performance.oft"  
displayEmail (project\_folder + "template\\" + emailTemplateName , attachmentPaths = attachment\_files)

A .oft file is an Outlook Template file that can be used to create and send e-mail messages containing information that rarely change from one message to the next. To create a .oft file, you can “Save As” an email message, including the subject line, message body, and any attachments.

The .oft file will be saved in the default Outlook templates folder. You can then move it to the folder of your choice. This is the sub-folder “Template”.

The great advantage is to automatically retrieve all mailings and content required by the target audience, as well as any attachments. And the .oft can be overwritten if necessary.

**Batch File**

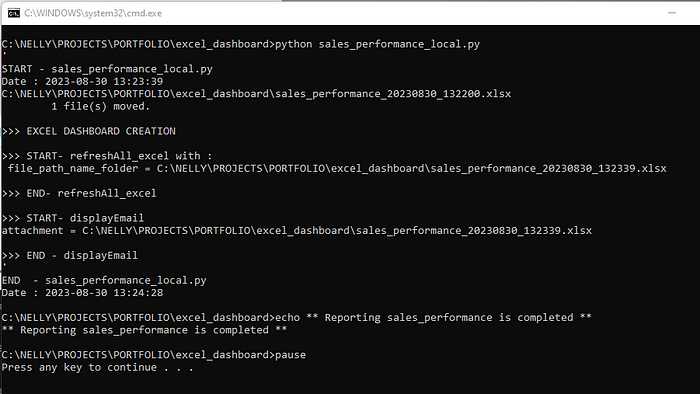
A .bat file is a Windows batch file, i.e. a text file that contains a series of commands that can be executed by the operating system. Batch files are often used to automate tasks, such as running scripts or launching programs.

For the moment, we have almost finished automating our report. All the Python code described in the sections above is wrapped up in a Python file [here](https://github.com/nellytchiengue/sales_performance_dashboard/blob/main/sales_performance.py).

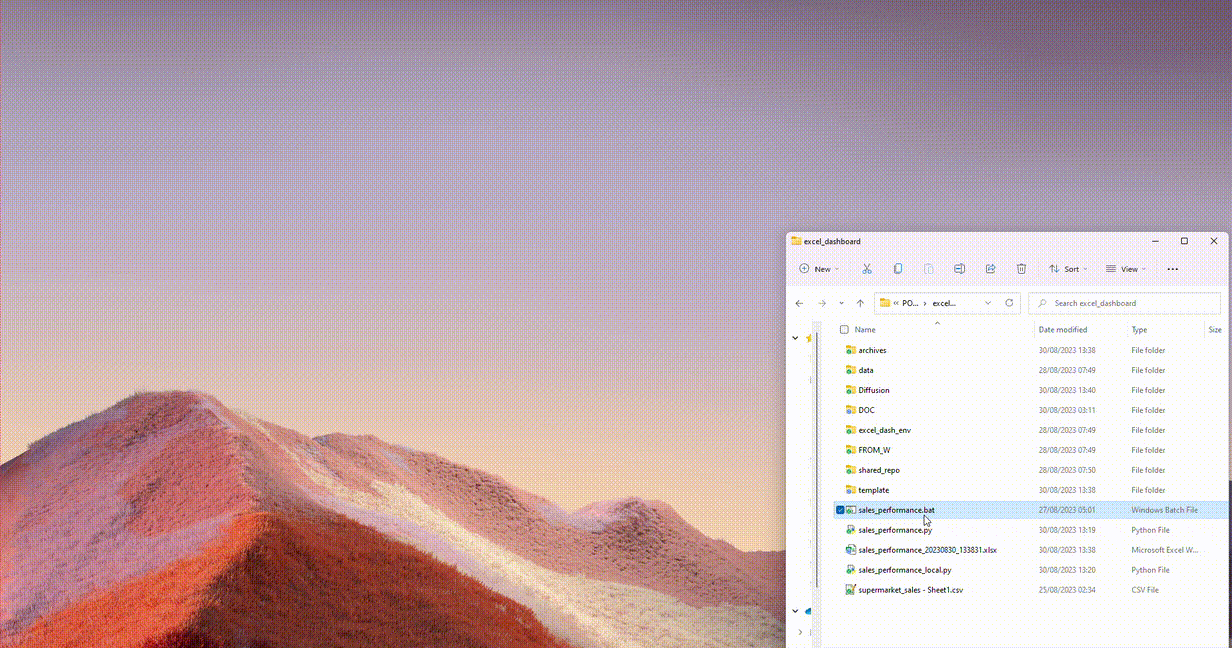
To go a step further and produce the Excel dashboard in **single click,** we can run the .py file via the Windows [batch file](https://github.com/nellytchiengue/sales_performance_dashboard/blob/main/sales_performance.bat) that contains the code:

python sales\_performance.py

The Windows terminal :



The advantage of using a .bat file to launch the Python script is that it can be scheduled to run at a specific time or interval. This means that the Python script can be used to automate the production and regular sending of reports by e-mail.



Demo

**Conclusion**

**Automation** is the ultimate solution for freeing yourself from the drudgery of repetitive data processing tasks. Your life becomes much easier thanks to improved efficiency, accuracy, and reduced human error.

In this article, we show you how to create an **interactive Excel Dashboard** and automate it by manipulating data with **Python**in basic way.

The possibilities go beyond what we demonstrate here. Nevertheless, it can be a good start.

With a plethora of tools for automating almost every possible Excel-related operation, this tutorial explores the capabilities of Python that make it a powerful tool for working with Excel files or data in general.

( And I haven’t even mentioned the [new combination](https://techcommunity.microsoft.com/t5/excel-blog/announcing-python-in-excel-combining-the-power-of-python-and-the/ba-p/3893439) of Excel and Python! )

Trust me, you don’t want to miss out on these tools!

***Thank you for reading this ❤️***

*Give claps, comment, and share if this was helpful to you.*

*You can reach me on*[*LinkedIn*](http://linkedin.com/in/nelly-tchiengue-15740961)*.*

*The full project scripts ( Excel, .py, and .bat files ) are available in the*[*Git Hub repository*](https://github.com/nellytchiengue/sales_performance_dashboard/tree/main)*.*