**Activity Adding a chart using Python (Optional)**

**Introduction**

You now have a deeper understanding of custom visualizations, and you are familiar with several ways to create them within Microsoft Power BI desktop, including the use of Python. Python is a powerful data analytics language that is supported by Power BI. Python, along with its collection of libraries can be used to build and deploy custom visualizations to meet specific design and analytical requirements.

**Scenario**

The size of the sales dataset for Adventure Works is significant because it is a large company operating in a multinational environment. It needs an extensive range of visualizations to correctly reflect these sales results and market trends in its reports. In some situations, Power BI core visualizations are too limited to do this. In this activity, you’ll use Python scripts and libraries to generate the analysis and visualization for a new custom visual in Power BI.

The specific tasks in the activity are to:

* Enable Python scripting if necessary.
* Ensure that Power BI detects the Python home directory path.
* Use Python to create a **Bar chart** representing the total monthly sales of Adventure Works.
* Confirm that the Python chart is interactive by placing another Power BI core visual element on the report page.

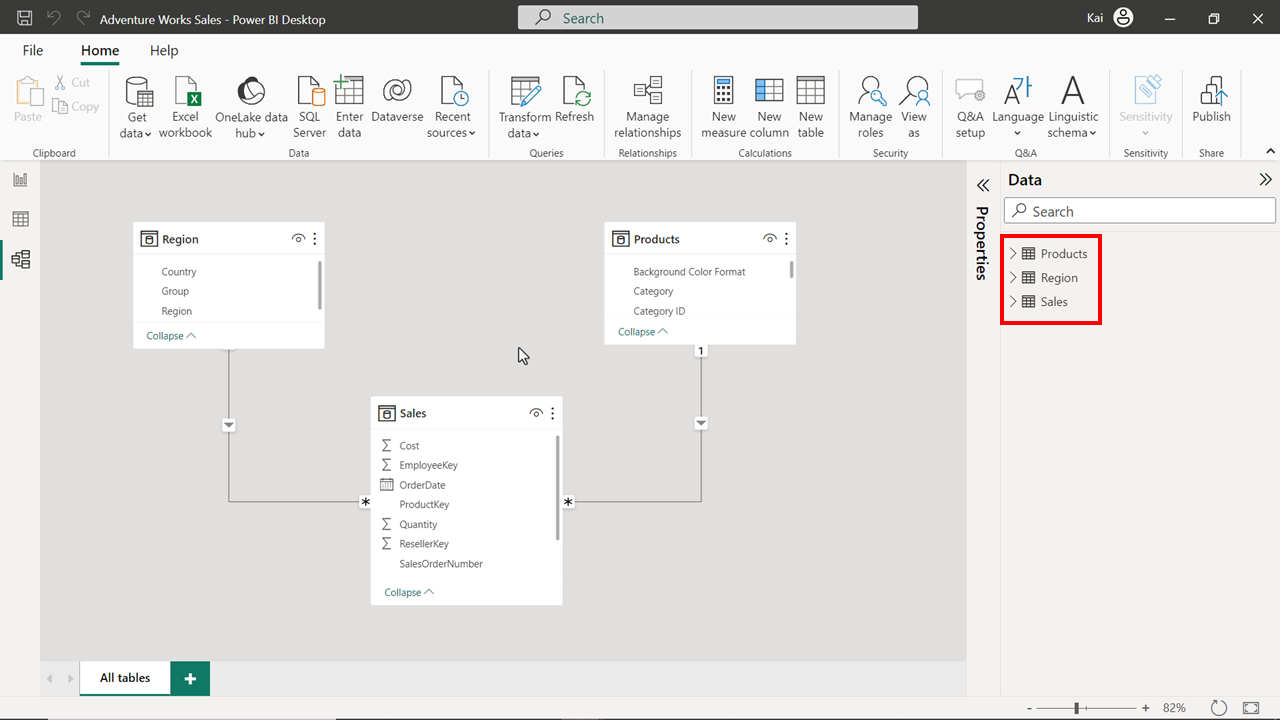
This reading provides you with a step-by-step guide for completing the tasks along with screenshots of each step.

**Step 1: Download the Adventure Works Power BI project**

1. First download and save the Microsoft Power BI project *Adventure Works Sales.pbix* to your local computer.  Confirm that the data model contains three data tables called **Sales**, **Regions**, and **Products**.

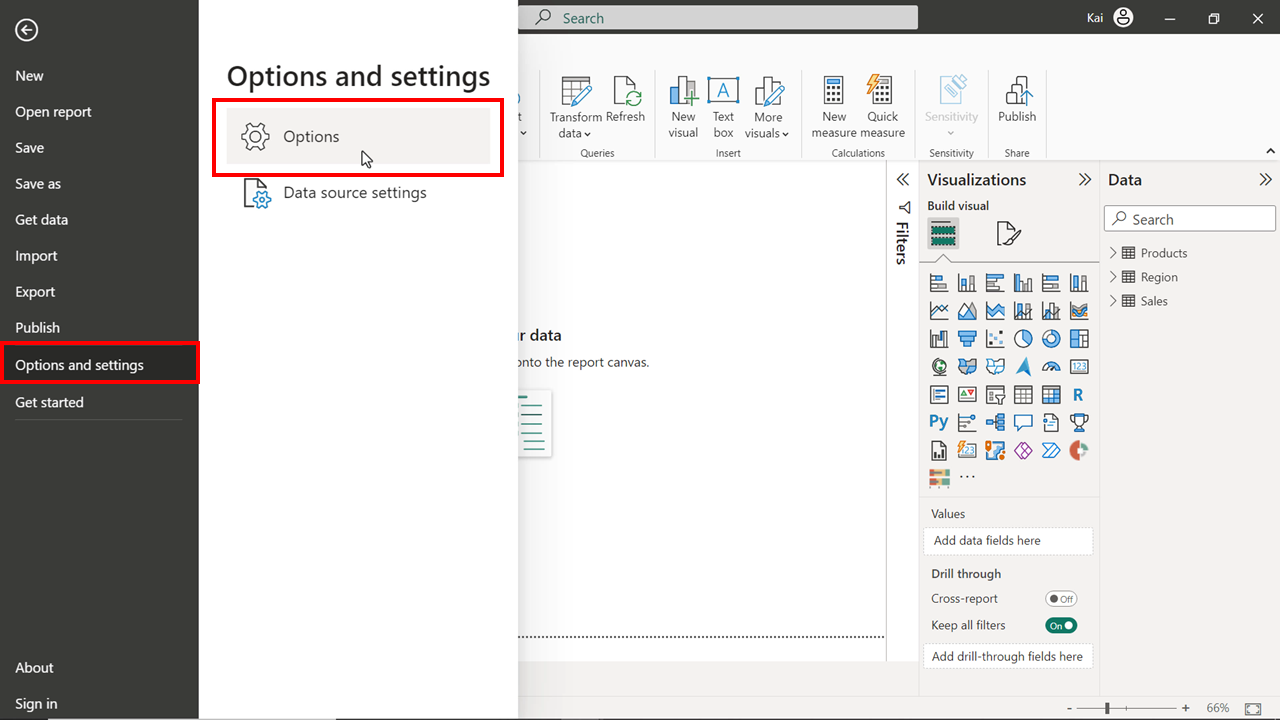
[Adventure Works Sales](https://d3c33hcgiwev3.cloudfront.net/jIUCYUFkQeu7FoBcmNfNBQ_6e508070be0447deb82f0e8fd38dece1_Adventure-Works-Sales.pbix?Expires=1712275200&Signature=gigo60sfXiv0Dn-d1Eko7Fd87POewRLs3SPG6dfotdpXgTIdgF8ZU3a5fjnquK8~xVXpTNEbA8622U4Mpw7jx8ThEtmaEWTw3hGBDB8CRG78doPts4eMHwmWjh-qQZ3gdCHDqf4Xfx21XGxvMUnVvFAiYdQ2B879UvuLPCioQD4_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

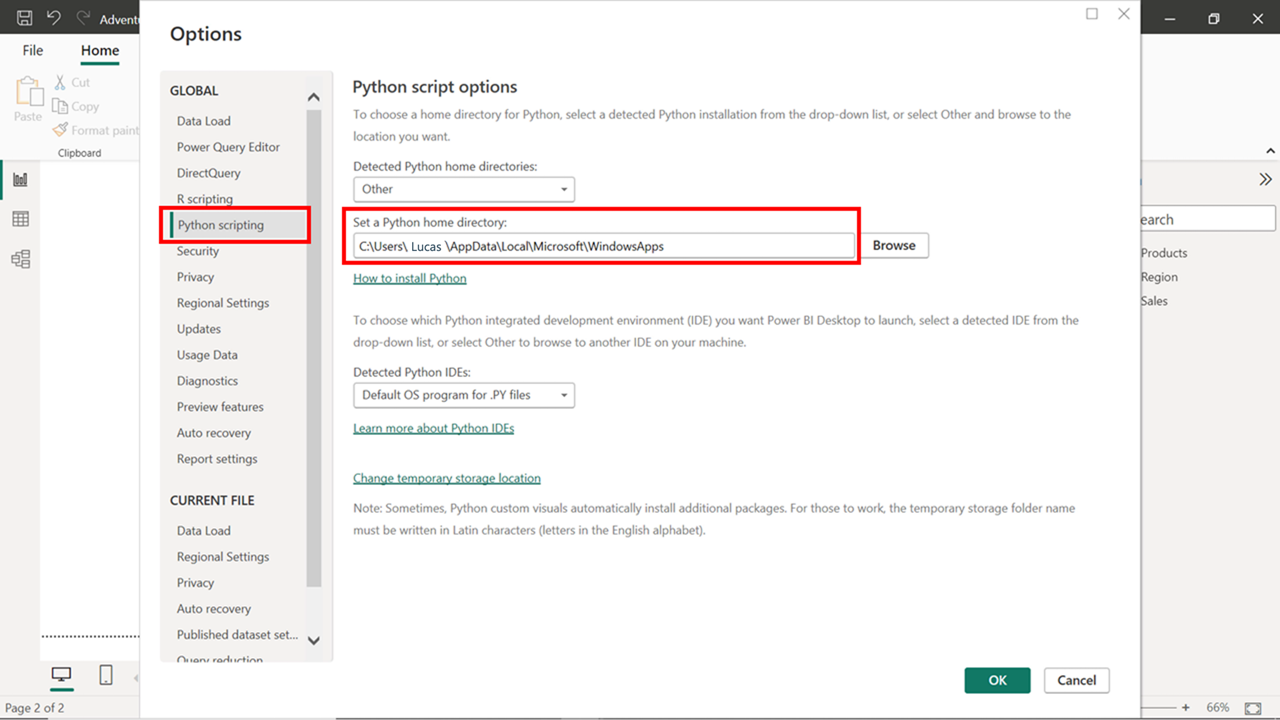
[PBIX File](https://d3c33hcgiwev3.cloudfront.net/jIUCYUFkQeu7FoBcmNfNBQ_6e508070be0447deb82f0e8fd38dece1_Adventure-Works-Sales.pbix?Expires=1712275200&Signature=gigo60sfXiv0Dn-d1Eko7Fd87POewRLs3SPG6dfotdpXgTIdgF8ZU3a5fjnquK8~xVXpTNEbA8622U4Mpw7jx8ThEtmaEWTw3hGBDB8CRG78doPts4eMHwmWjh-qQZ3gdCHDqf4Xfx21XGxvMUnVvFAiYdQ2B879UvuLPCioQD4_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)



**Step 2: Prepare Power BI to use the Python visualization**

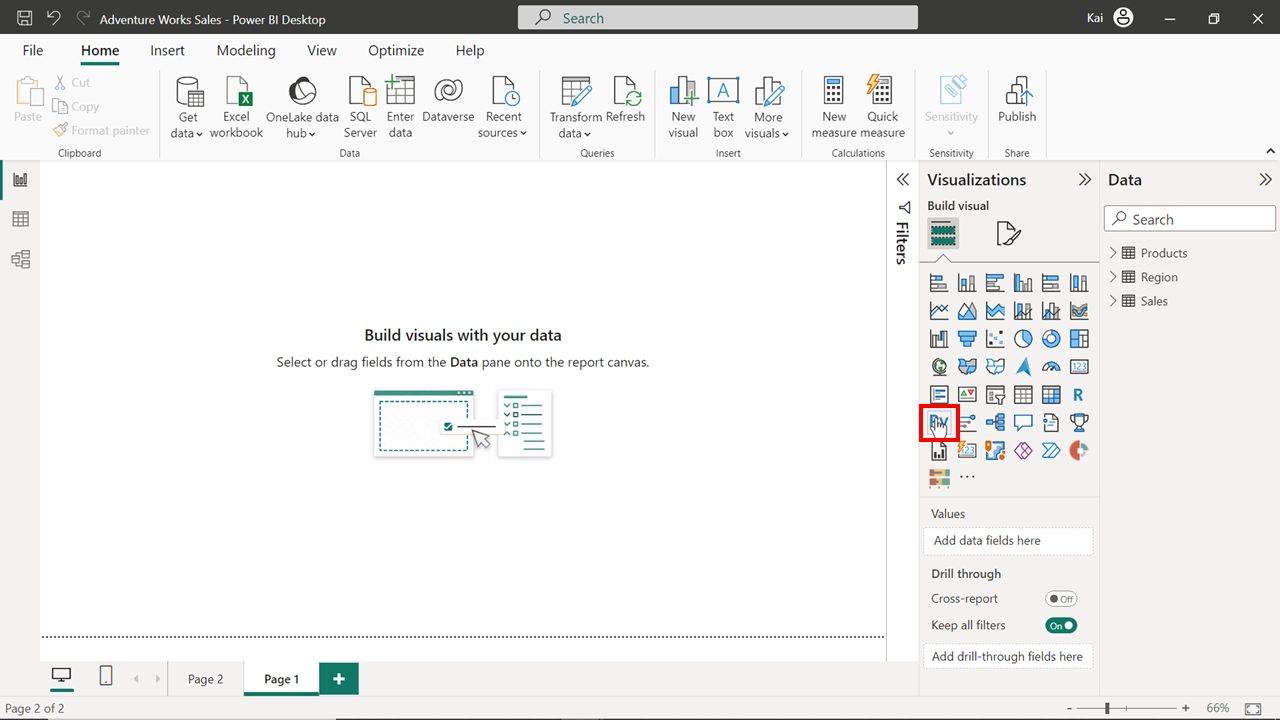
1. To enable Python scripting on your Power BI desktop, select **File** and then select **Options and Settings** then **Options.** In **Options**, select **Python Scripting** in the bar on the left of the window.
2. Check that the path of the Python home directory is automatically detected. If the path is not correctly configured, you will get an error message when importing data or creating a custom visual in Power BI.

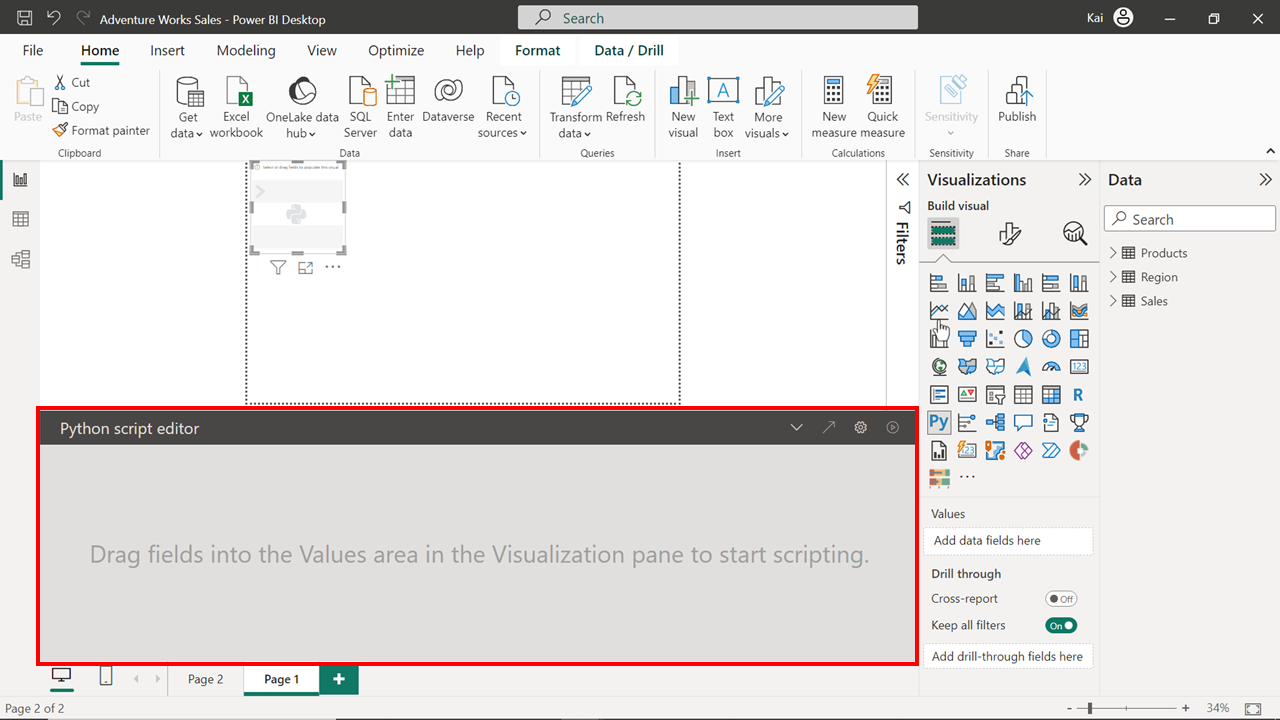




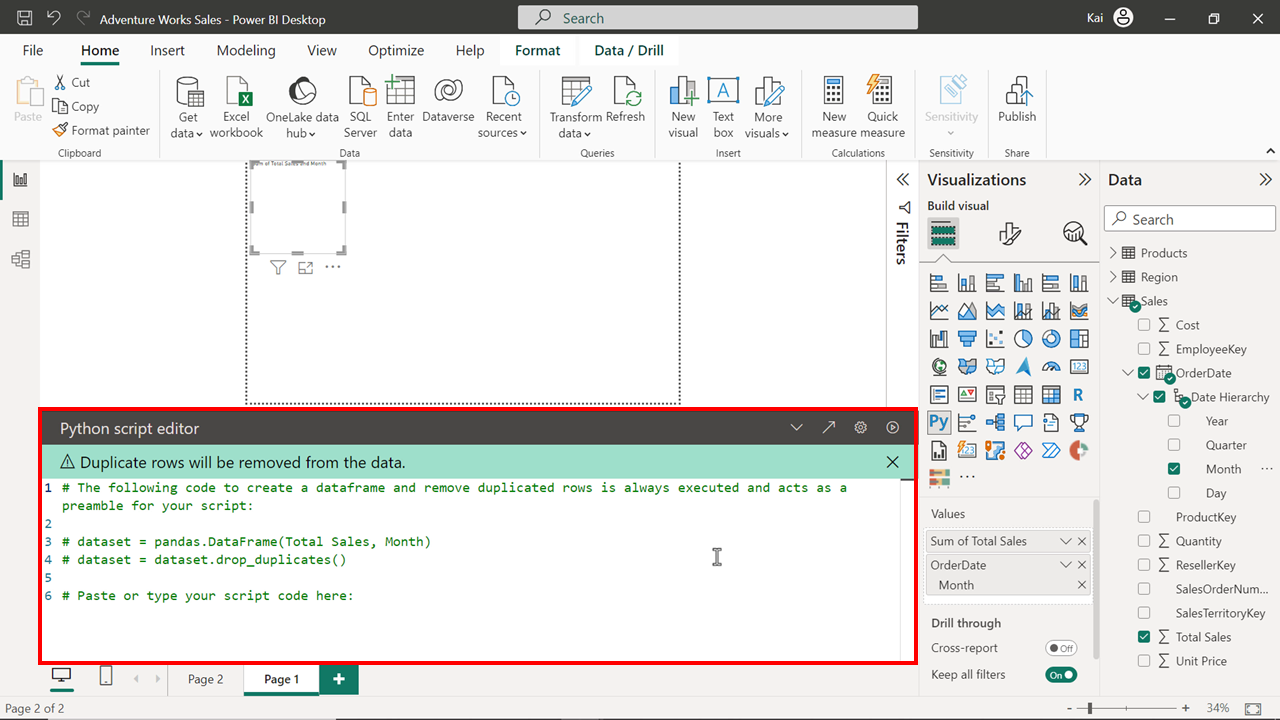
**Step 3: Create Python chart visual**

1. The Python visualization uses only the fields that are brought to the **Values** section of the Python visual to create a dataframe. It is important to remember that if you are using fields from different tables of your data model, you need to make sure that the tables are related using appropriate relationships and cross-filter direction to achieve accurate results in the final visual.
2. Select the **Python visual** icon from the **Visualization pane**. Once you select the Python icon from the visualization pane of Power BI, a message appears saying: "**Enable script visuals, select enable”**. Select **Enable** to open the Python script editor (it will also add a Python visual placeholder image to the report canvas.)





Drag the **Total sales** field and the **Month** column from the **OrderDate** hierarchy from the **Sales table** to the **Values** section of the **Python visual**. You can expand the **OrderDate** column to see the **Month** field in the hierarchy if it is not already visible.



3. Write the following Python script in the **Python script editor** window and then execute the code to generate a **Bar chart** showing the **Total sales** by month.

1

2

3

4

import matplotlib.pyplot as plt

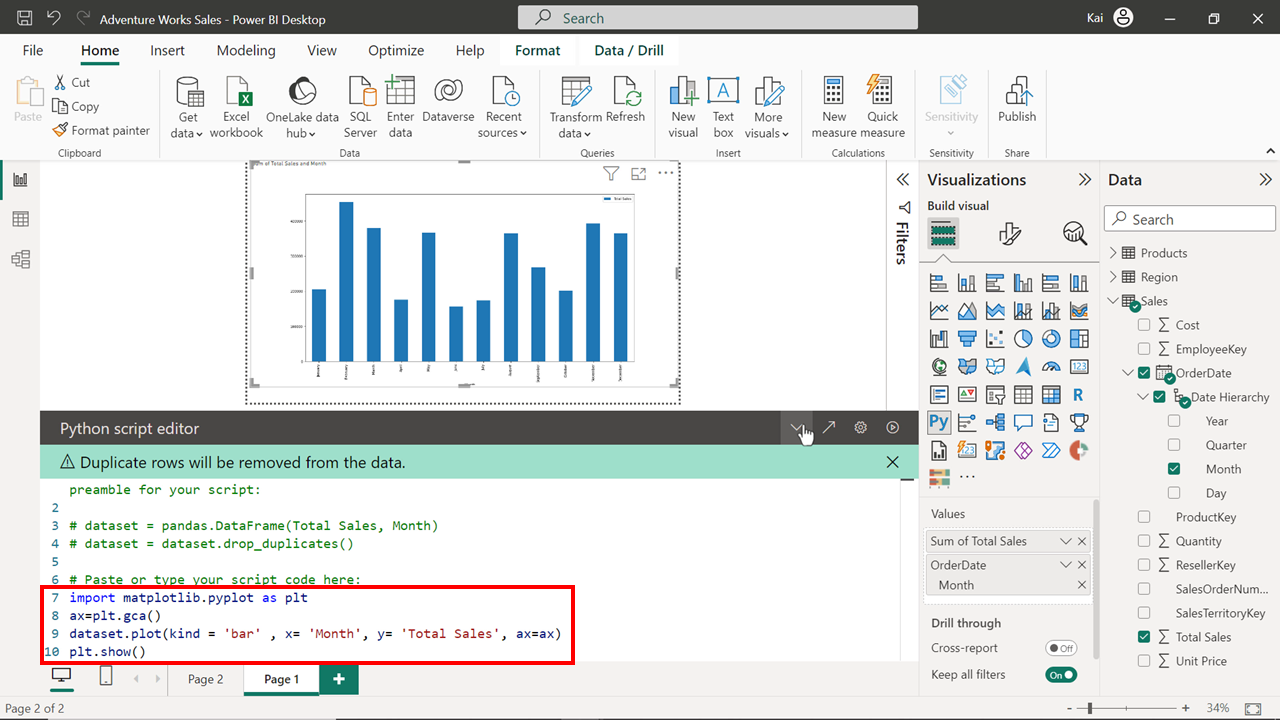
ax = plt.gca()

dataset.plot(kind = ‘bar’ , x= ‘Month’, y= ‘Total Sales’, ax=ax)

plt.show()

These code operations are:

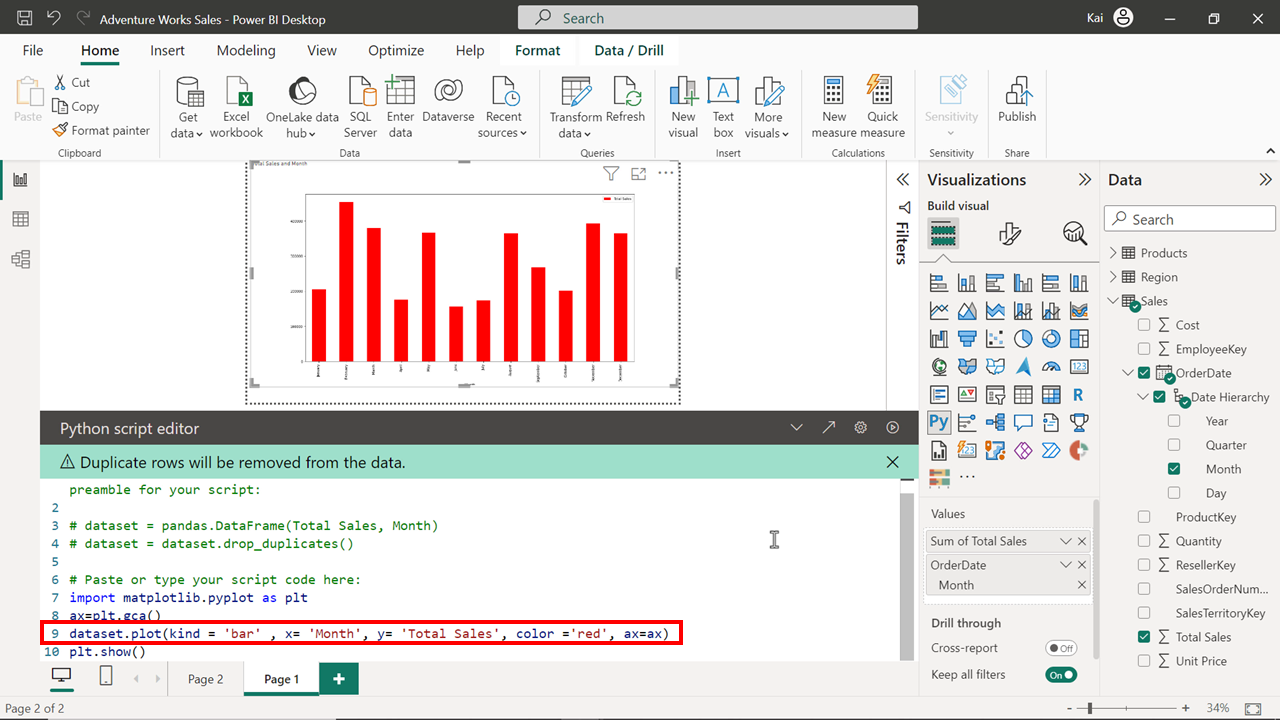
* Importing the matplotlib library which creates the bar chart.
* Defining the specifications of the chart such as the X and Y-axis values, and the formatting to be added.
* Commanding the output in the last line of code.

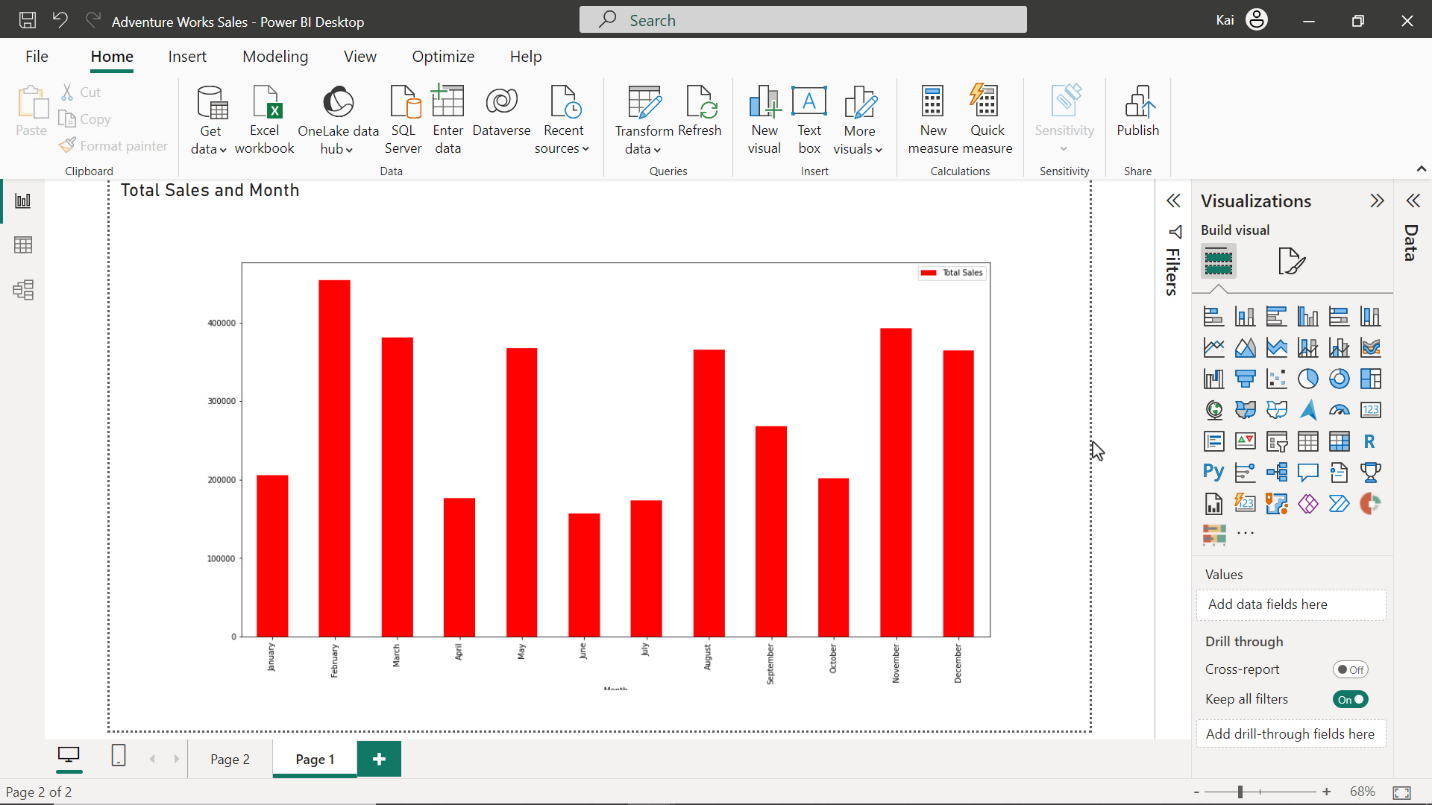


You cannot format this visual in Power BI as you typically do with Power BI core visuals. To format a Python visual, you need to modify the code for specific formatting. For example, the bars in the chart can be changed to a red color by amending the third line of the script as follows:

1

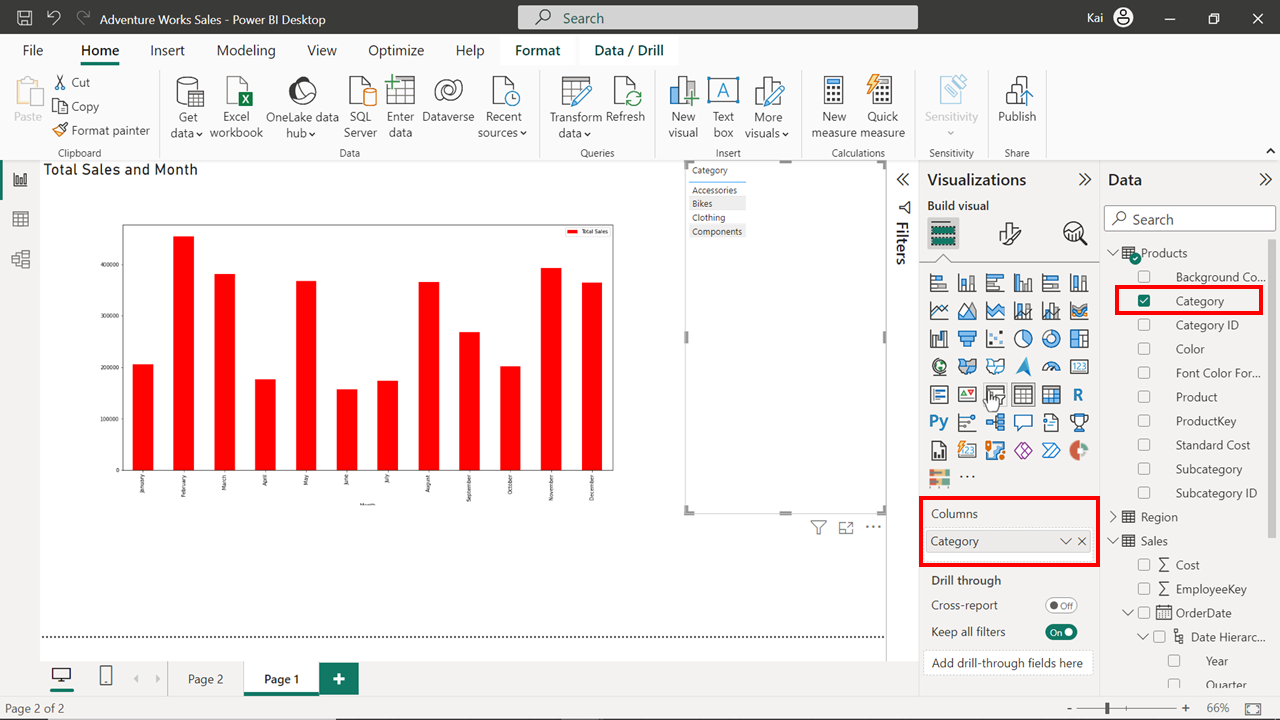
dataset.plot(kind = ‘bar’ , x= ‘Month’, y= ‘Total Sales’, color = ‘red’, ax=ax)





**Step 4: Check the interactivity of the Python visual.**

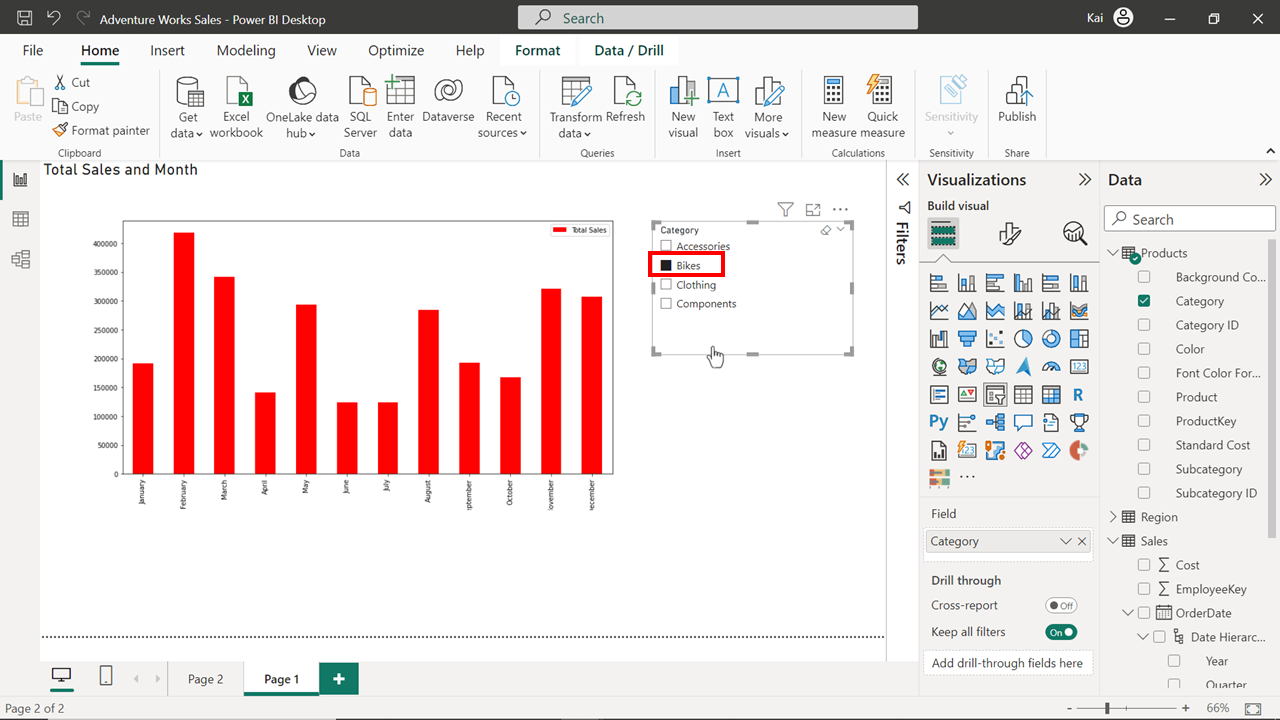
1. You can confirm that the Python visual is interactive and displaying the data results correctly by creating a **Category slicer** in the report page of Power BI. Drag the **Category** field from the **Product table** to the report canvas. Convert the visual to a slicer by selecting the **Slicer** icon in the **Visualization pane**. You can use the **Slicer** **options** to format the **Slicer** as **tile** or **dropdown**.

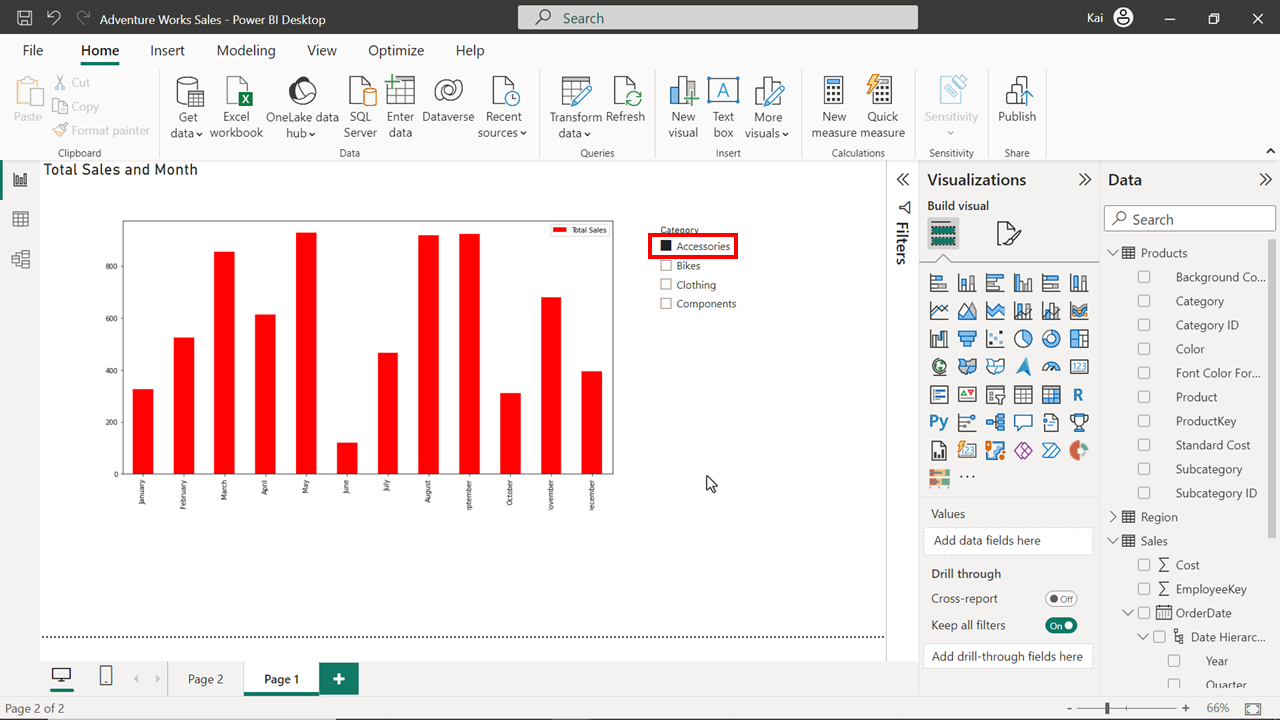


A screenshot of a computer

Description automatically generated

2. Select each of the **categories** in the slicer in turn to confirm that the sales values in the Python bar chart automatically adjust to reflect the sales values for that category.





**Step 5: Save the Power BI project**

1. Save your Power BI project to your local computer.

**Conclusion**

With these steps, you can successfully create a bar chart using Python in Power BI. You can use Python’s potential to create custom visualizations in situations where Power BI core visualizations are not sufficient or powerful enough to build the required visualization.