NAME : B. RISHITHA

ROLL NO : 21X05A6710

COLLEGE : NARASIMHA REDDY ENGINEERING COLLEGE UGC AUTONOMOUS

BRANCH : DATA SCIENCE

YEAR : 4TH YEAR

GUIDER NAME : MR. B. LOKESH

PROJECT

TITLE :

Analysis and visualization of sales marketing data with the help of python graphics library Matplotlib.

Organization Name :

IBM India :

The International Business Machines Corporation (IBM), nicknamed Big Blue,is an American multinational technology corporation headquartered in Armonk, New York and is present in over 175 countries.It specializes in computer hardware, middleware, and software, and provides hosting and consulting services in areas ranging from mainframe computers to nanotechnology. IBM is the largest industrial research organization in the world, with 19 research facilities across a dozen countries, and has held the record for most annual U.S. patents generated by a business for 29 consecutive years from 1993 to 2021.

Dataset Name:

Sales.xlsx :

A sales data definition Sales data is a term that includes a large array of metrics but, broadly speaking, if you can measure something in relation to the sales process, it's viable sales data.

Sample Sales Data, Order Info, Sales, Customer, Shipping, etc., Used for segmentation, Customer Analytics, Clustering and More. Inspired for retail analytics. This was originally used for Pentaho DI Kette, But I found the set could be useful for sales Simulation training.

Problem Statement:

A company consists of a sales and marketing data set that contains a number of rows and a huge amount of columns of sales and marketing data with respect to their products and categories. Clients want to visualize all the data with the help of it is paid visualization Library called MatPlot. According to the best content, we need to be clear about the data sets along with that we have to be fine the relationship between sales and marketing with our clients which can be satisfied with to be their job and task. Companies give five task to the visualization what is the top category which gives the highest number of sale, what are the various kinds of products which give at least maximum five top results of profit and top five results of a loss along with their results, as a data analyst or as a data science engineer you need to do find the relationship between the clean data set with the help of process extraction Transformer load create a data pipeline build a relationship between the fields and values along with that plot of visualization so our clients can get the very interactive data sets and all they need to be find out the inside very well.

Tools and Technology :

1. Python Programing Language:

Python programming is a high-level, versatile programming language known for its simplicity and readability. It emphasizes clean syntax and supports a wide range of applications, including web development, data analysis, artificial intelligence, and automation. Its extensive libraries and community make it popular for both beginners and experts.

2. Matplots Library:

Matplotlib is a widely used Python library for creating visualizations and plots. It offers a comprehensive range of tools for generating charts, graphs, and plots, aiding data exploration and presentation. Its versatility and ease of use make it a go-to choice for data scientists and analysts in various domains.

3. Pandas :

Pandas is a widely used open-source Python library for data manipulation and analysis. It provides data structures and functions to efficiently work with structured data, like tables and time series. Pandas simplifies tasks such as data cleaning, transformation, exploration, and visualization, making it essential for data professionals and analysts.

4. NumPy:

NumPy is a powerful Python library for numerical computations. It provides support for arrays, matrices, and mathematical functions, enabling efficient manipulation of large datasets and complex mathematical operations. NumPy is widely used in fields like data science, machine learning, and scientific research for its performance and convenience.

5. Jupyter Notebook:

Jupyter Notebook is an open-source web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It&#39;s commonly used for data analysis, scientific computing, and machine learning due to its interactive nature and support for various programming languages like Python, R, and Julia.

TYPES OF PLOTS :

In the MatPlotLib there are 6 plots. They are :

1. Bar Graph
2. Scatterplot
3. Line

4. Histogram
5. Box plot
6. Subplots

1. Bar Graph :

Bar graphs are the pictorial representation of data (generally grouped), in the form of vertical or horizontal rectangular bars, where the length of bars are proportional to the measure of data. They are also known as bar charts. Bar graphs are one of the means of data handling in statistics.

Types of Bar Graph :

Vertical

Horizontal

Grouped Bar Graph

Stacked Bar Graph

2. Scatterplot :

Scatter plots are the graphs that present the relationship between two variables in a data-set. It represents data points on a two-dimensional plane or on a Cartesian system. The independent variable or attribute is plotted on the X-axis, while the dependent variable is plotted on the Y-axis. These plots are often called scatter graphs or scatter diagrams.

Types of Scatterplot :

Bar Graph
Graphical Representation
Correlation
Data Sets

3. Line Chart :

A line chart, also known as a line graph, is a type of data visualization used to display data points as a series of connected line segments. It is commonly used to illustrate trends and patterns in data over a continuous range, typically along a horizontal (x) axis that represents time, distance, or some other continuous variable, and a vertical (y) axis that represents the values being measured.

Line charts are particularly useful for showing how a specific variable changes over time or across a continuous range. They're often used to depict data with a consistent frequency of measurements, such as daily stock prices, monthly sales figures, or annual temperature variations.

4. Histogram :

A histogram is a graphical representation of a data set that shows the distribution of values within specific intervals, often referred to as "bins." It provides a visual summary of the frequency or count of data points that fall into each bin. Histograms are particularly useful for understanding the underlying distribution of a continuous or discrete variable.

5. Boxplot :

A box plot, also known as a box-and-whisker plot, is a graphical representation of the distribution of a data set. It provides a visual summary of the central tendency, spread, and potential outliers within the data. Box plots are particularly useful for comparing distributions across different categories or groups and for identifying important statistical characteristics of the data.

6. Subplots :

A subplot refers to a smaller, individual plot or chart that is positioned within a larger overall plot or chart. Subplots are used to display multiple graphs or visualizations in a single figure, allowing for the comparison of different data sets or aspects of the data in a concise and organized manner.

Subplots are particularly useful when you want to present related information side by side or when you have multiple variables or data sets that you want to visualize together. By using subplots, you can avoid the clutter that can arise from creating separate figures for each chart and make it easier for viewers to understand the relationships between the different visualizations.

01-bar-graph

August 20, 2023

0.1 Visualisation in Python - Matplotlib

You will be working with the sales dataset for an online retailer. The data is collected over a period of three years: 2012 to 2015. It contains the information of sales made by the company B. NY. Mellon.

The products captured belong to three categories: - Furniture - Office Supplies - Technology

Also, the company caters to five different markets: - USCA - LATAM - ASPAC - EUR - AFR

Let's get started with the plots. We will be using the '**pyplot**' package of the Matplotlib library.

[11]:
```
# importing numpy and the pyplot package of matplotlib
import matplotlib.pyplot as plt
import numpy as np
#import pandas as pd
```

[12]:
```
# Creating an array with product categories
product_category = np.array(["Furniture","Technology","office supplies"])
```

[13]:
```
# Creating an array with the sales amount
# Furniture: 4110451.90
# Technology: 4744557.50
# Office Supplies: 3787492.52
sales = np.array([4110451.90, 4744557.50 , 3787492.52])
```

It is not necessary that you are provided with the aggregated values every time. In such cases, you first need to calculate the values and then build the graphs.

Let's see how to plot a bar graph for the provided values.

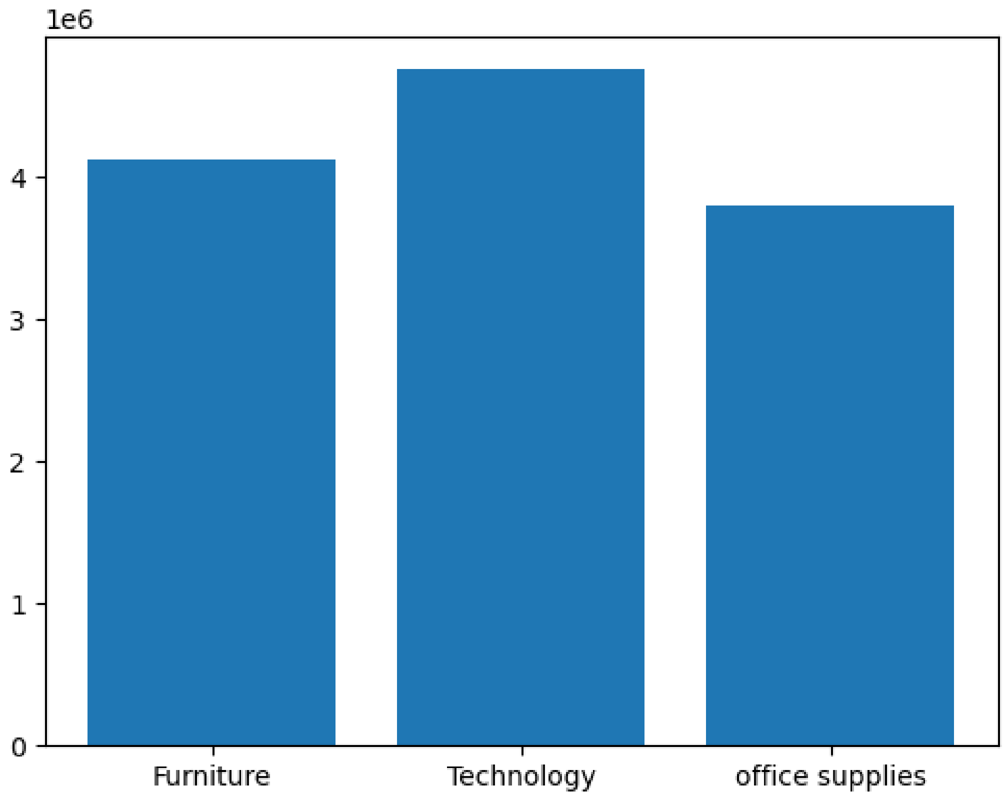
0.1.1 Bar Graph: Plotting sales across each product category

- A bar chart uses bars to show comparisons between categories of data.
- One axis will generally have numerical values or measures,
- The other will describe the types of categories being compared or dimensions.

Let's start with plotting a bar graph representing the sales across different categories over the period.

[17]:
```
# plotting the bar graph with product categories on x-axis and sales amount of␣
↪y-axis
plt.bar(product_category , sales)

# necessary command to display the created graph
plt.show
```

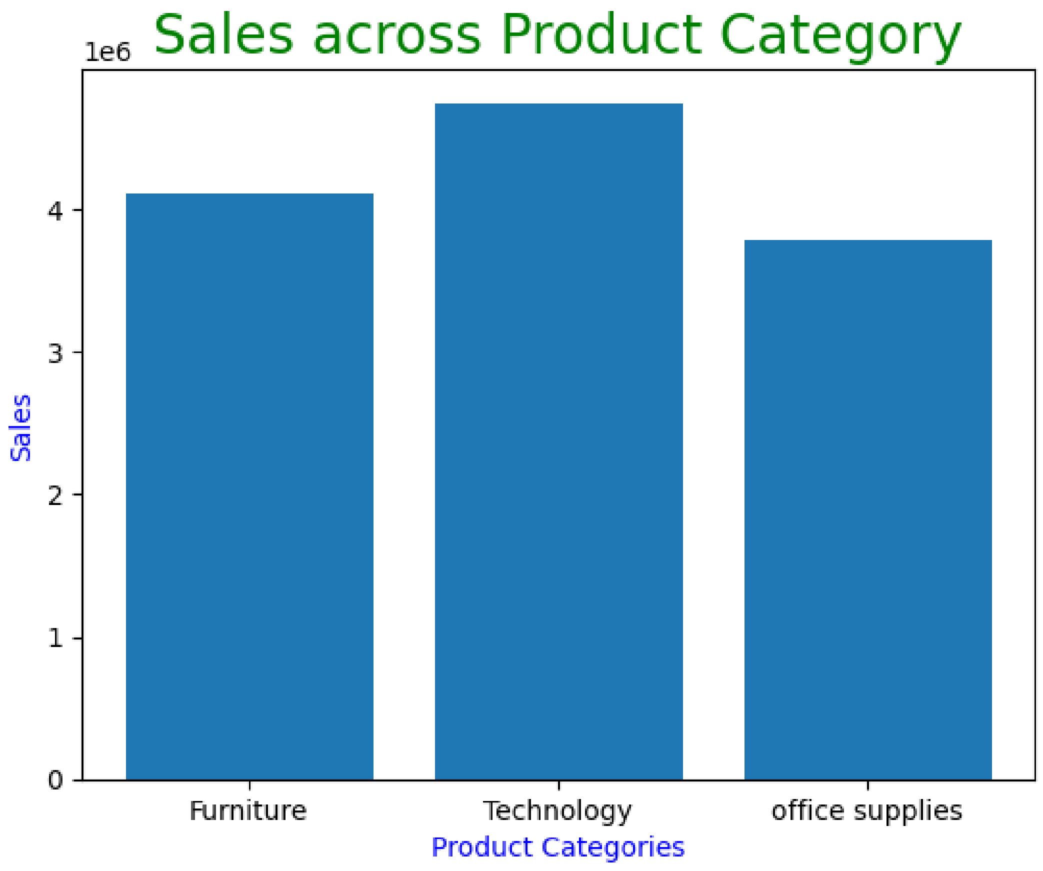
[17]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Adding title and labeling axes in the graph

[25]:
```
# plotting the bar graph with product categories on x-axis and sales amount of␣
↪y-axis
plt.bar(product_category , sales)

# adding title to the graph
plt.title('Sales across Product Category', fontdict={ "fontsize":20 ,␣
↪"fontweight":5, "color":'Green'})

# labeling axes
```

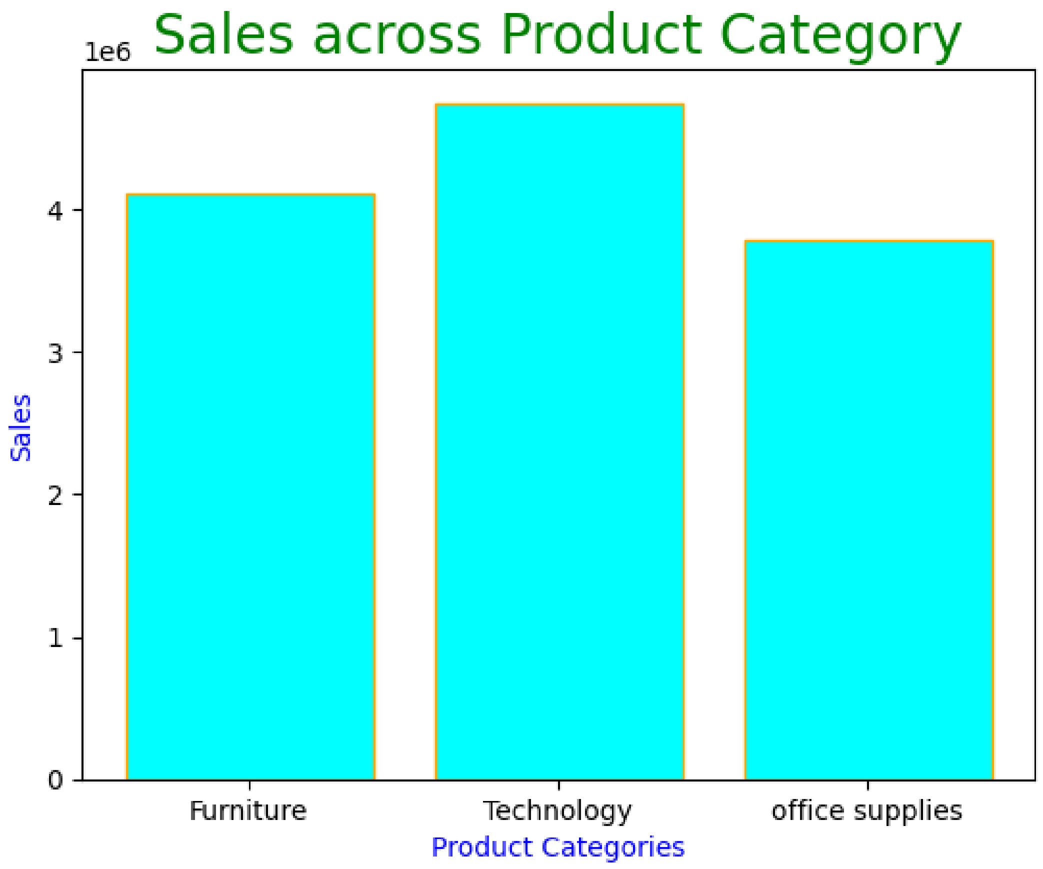
```
plt.xlabel("Product Categories",fontdict={"fontsize":10 , "fontweight":3,␣
 ↪"color":'Blue'})
plt.ylabel("Sales",fontdict={"fontsize":10 , "fontweight":3, "color":'Blue'})

# necessary command to display the created graph
plt.show()
```



Modifying the bars in the graph

[28]:
```
# changing color of the bars in the bar graph
# plotting the bar graph with product categories on x-axis and sales amount of␣
 ↪y-axis
plt.bar(product_category , sales, color ="cyan", edgecolor="Orange")

# adding title to the graph
plt.title('Sales across Product Category', fontdict={ "fontsize":20 ,␣
 ↪"fontweight":5, "color":'Green'})

# labeling axes
```

```
plt.xlabel("Product Categories",fontdict={"fontsize":10 , "fontweight":3,␣
  ↪"color":'Blue'})
plt.ylabel("Sales",fontdict={"fontsize":10 , "fontweight":3, "color":'Blue'})

# necessary command to display the created graph
plt.show()
```

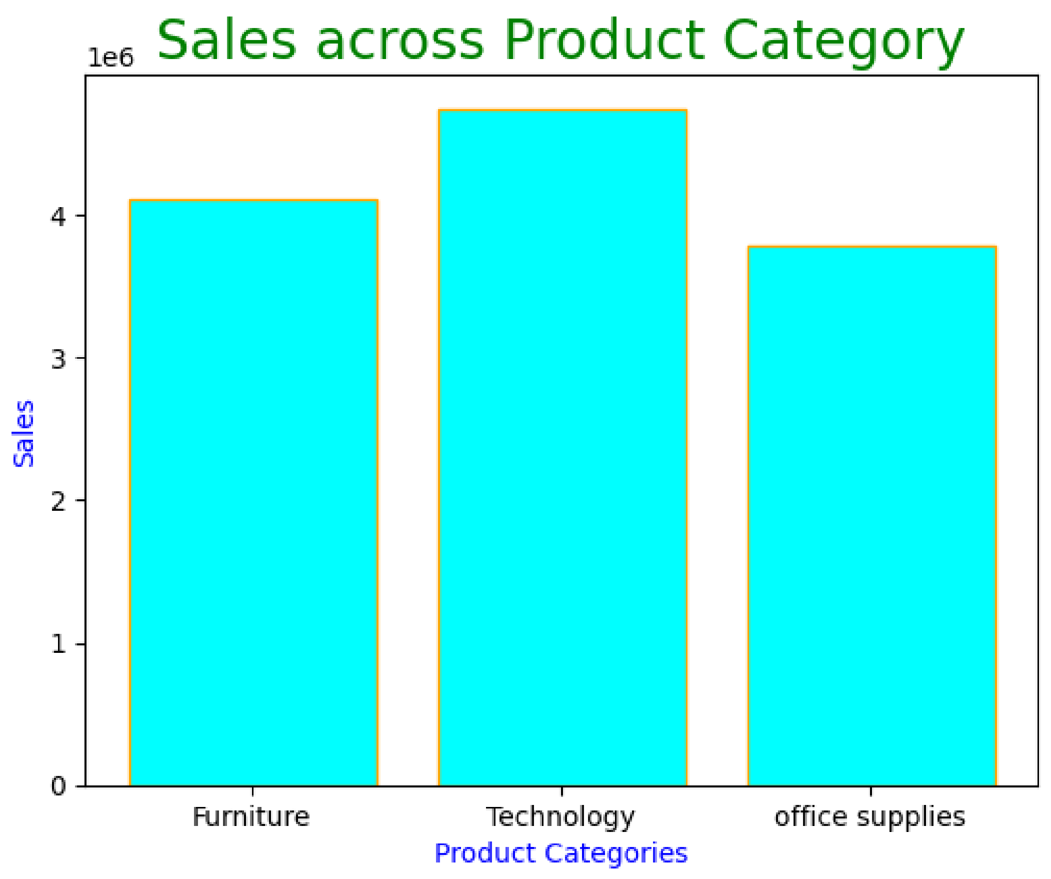


Adjusting tick values and the value labels

[40]:
```
# plotting the bar graph with product categories on x-axis and sales amount of␣
  ↪y-axis
plt.bar(product_category , sales, color ="cyan", edgecolor="Orange")


# adding title to the graph
plt.title('Sales across Product Category', fontdict={ "fontsize":20 ,␣
  ↪"fontweight":5, "color":'Green'})

# labeling axes
```

```
plt.xlabel("Product Categories",fontdict={"fontsize":10 , "fontweight":3,␣
↪"color":'Blue'})
plt.ylabel("Sales",fontdict={"fontsize":10 , "fontweight":3, "color":'Blue'})

# Modifying the ticks to show information in (lakhs)
tick_values = np.arange(0,7000000,10000000)
tick_labels = ["0L","20L","30L","40L","50L","60L","70L"]

#plt.yticks(tick_values,tick_labels)
plt.ytick(tick_values,tick_labels)
# necessary command to display the created graph

plt.show
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[40], line 17
     14 tick_labels = ["0L","20L","30L","40L","50L","60L","70L"]
     16 #plt.yticks(tick_values,tick_labels)
---> 17 plt.ytick(tick_values,tick_labels)
     18 # necessary command to display the created graph
     20 plt.show

AttributeError: module 'matplotlib.pyplot' has no attribute 'ytick'
```


Sales across Product Category
1e6
0
1
2
3
4
Sales
Furniture
Technology
office supplies
Product Categories

```
[ ]:
```

02-scatterplot

August 20, 2023

0.1 Visualisation in Python - Matplotlib

You will be working with the sales dataset for an online retailer. The data is collected over a period of three years: 2012 to 2015. It contains the information of sales made by the company.

The products captured belong to three categories: - Furniture - Office Supplies - Technology

Also, the company caters to five different markets: - USCA - LATAM - ASPAC - EUR - AFR

Let's get started with the plots. We will be using the '**pyplot**' package of the Matplotlib library.

```
[1]: # importing numpy and the pyplot package of matplotlib
     import numpy as np
     import matplotlib.pyplot as plt
```

```
[2]: # Creating an array with product categories: 'Furniture', 'Technology' and␣
      ↪'Office Supplies'
     product_category = np.array(['Furniture', 'Technology', 'Office Supplies'])
```

```
[3]: # Creating an array with the sales amount
     # Furniture: 4110451.90
     # Technology: 4744557.50
     # Office Supplies: 3787492.52

     sales = np.array ([4110451.90, 4744557.50, 3787492.52] )
```

It is not necessary that you are provided with the aggregated values every time. In such cases, you first need to calculate the values and then build the graphs.

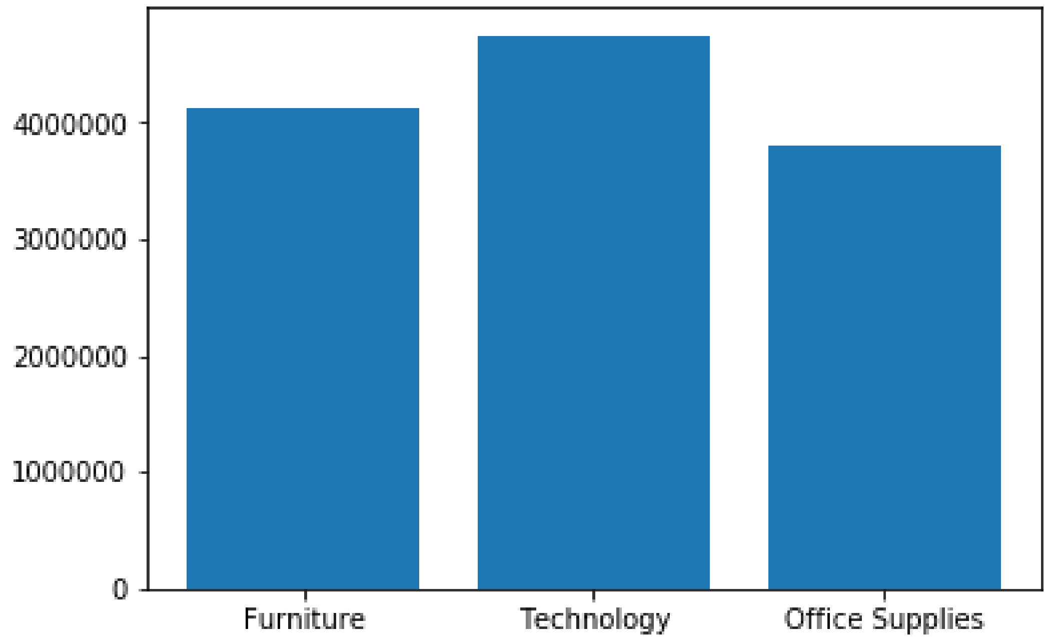
Let's see how to plot a bar graph for the provided values.

0.1.1 Bar Graph: Plotting sales across each product category

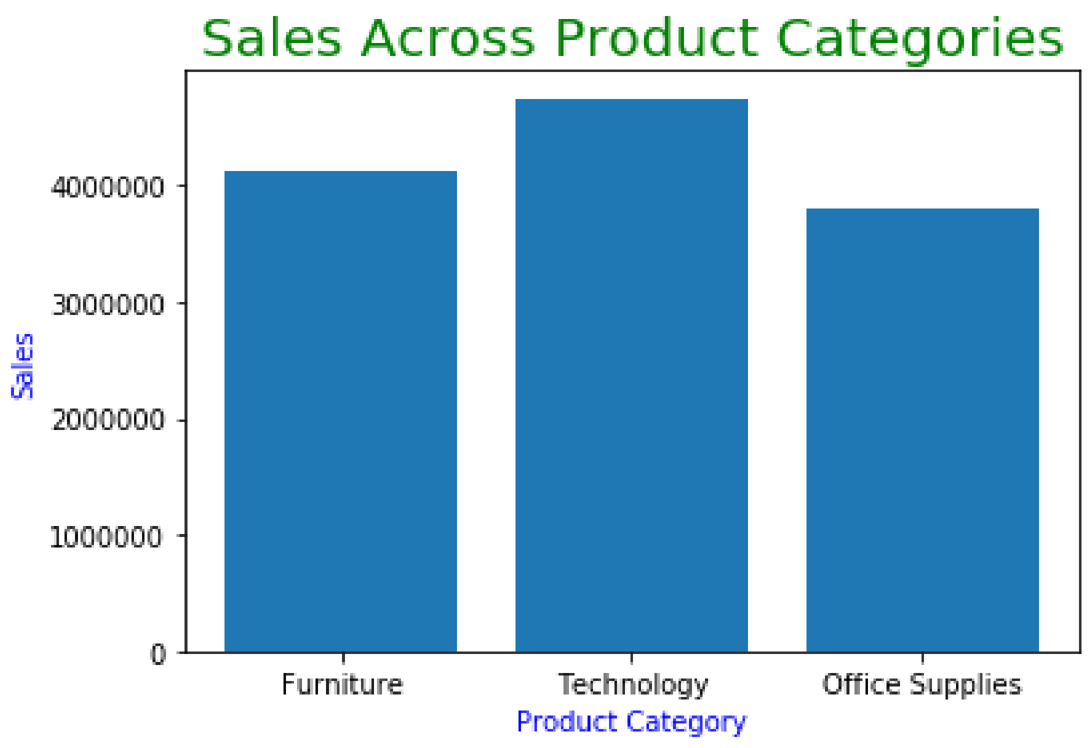
- A bar chart uses bars to show comparisons between categories of data.
- One axis will generally have numerical values or measures,
- The other will describe the types of categories being compared or dimensions.

Let's start with plotting a bar graph representing the sales across different categories over the period.

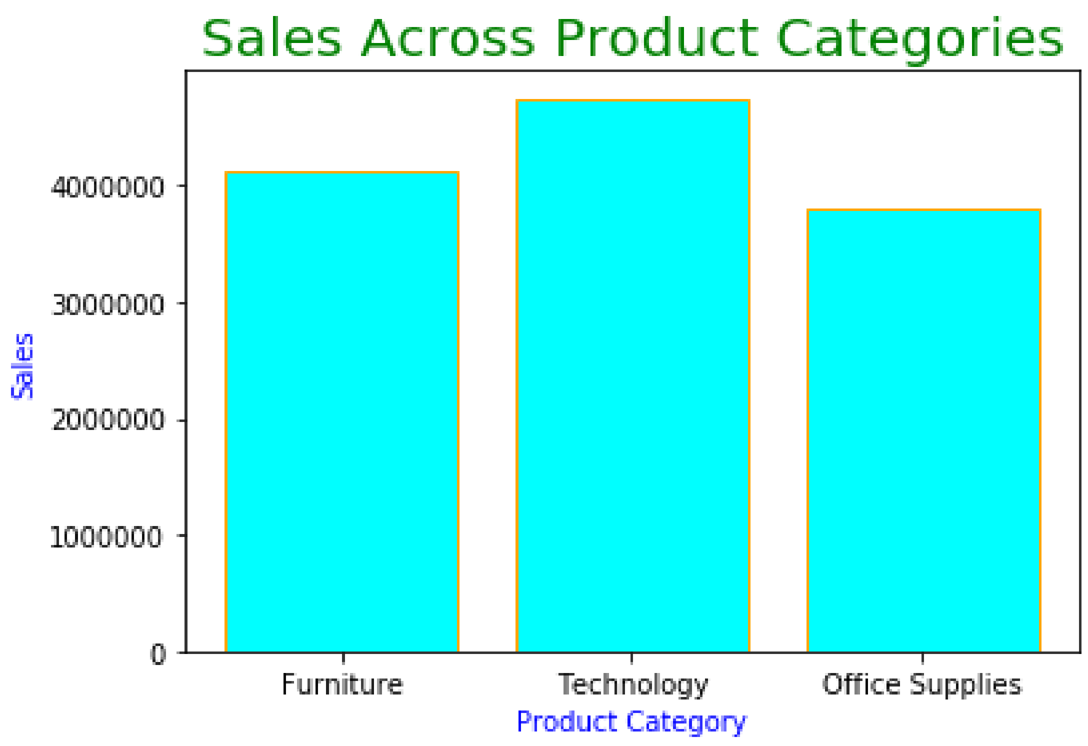
[4]:
```
# plotting the bar graph with product categories on x-axis and sales amount of␣
↪y-axis
plt.bar(product_category, sales)

# necessary command to display the created graphs
plt.show()
```



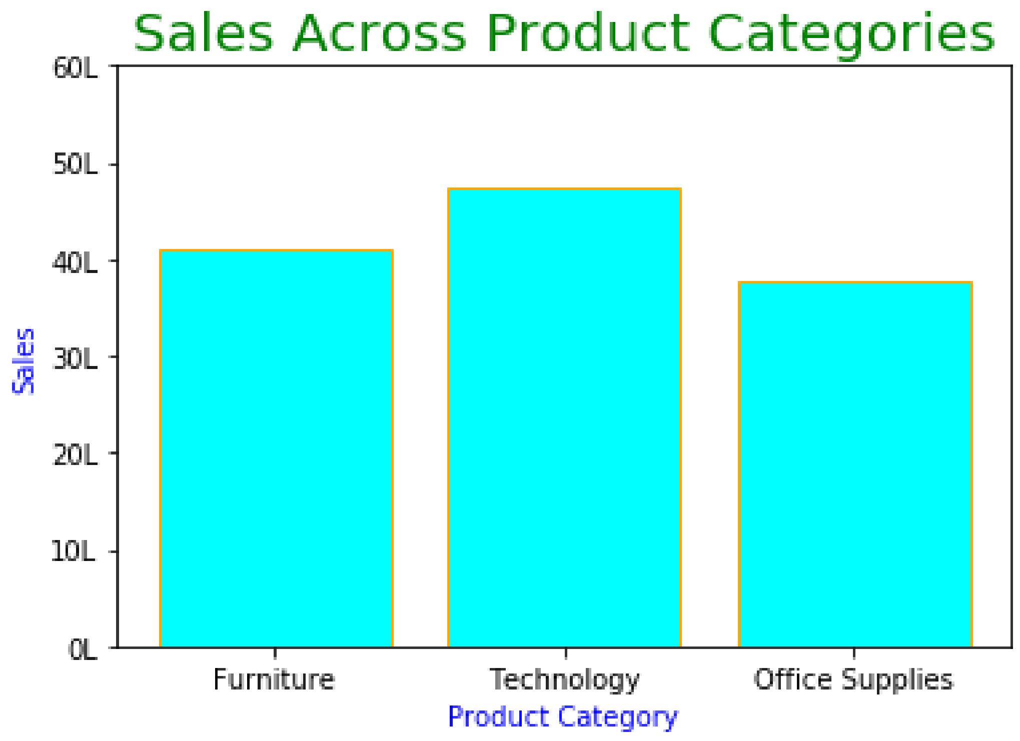
[5]:
```
plt.bar(product_category, sales)

# adding title to the graph
plt.title("Sales Across Product Categories", fontdict={'fontsize': 20,␣
↪'fontweight' : 5, 'color' : 'Green'})

# labeling axes
plt.xlabel("Product Category", fontdict={'fontsize': 10, 'fontweight' : 3,␣
↪'color' : 'Blue'})
plt.ylabel("Sales", fontdict={'fontsize': 10, 'fontweight' : 3, 'color' :␣
↪'Blue'})

plt.show()
```



```
[6]: # changing color of the bars in the bar graph
     plt.bar(product_category, sales, color='cyan', edgecolor='orange')

     plt.title("Sales Across Product Categories", fontdict={'fontsize': 20,␣
      ↪'fontweight' : 5, 'color' : 'Green'})

     plt.xlabel("Product Category", fontdict={'fontsize': 10, 'fontweight' : 3,␣
      ↪'color' : 'Blue'})
     plt.ylabel("Sales", fontdict={'fontsize': 10, 'fontweight' : 3, 'color' :␣
      ↪'Blue'})

     plt.show()
```



[7]:
```
plt.bar(product_category, sales, color='cyan', edgecolor='orange')

plt.title("Sales Across Product Categories", fontdict={'fontsize': 20,␣
↪'fontweight' : 5, 'color' : 'Green'})

plt.xlabel("Product Category", fontdict={'fontsize': 10, 'fontweight' : 3,␣
↪'color' : 'Blue'})
plt.ylabel("Sales", fontdict={'fontsize': 10, 'fontweight' : 3, 'color' :␣
↪'Blue'})

# Modifying the ticks to show information in (lakhs)
tick_values = np.arange(0, 7000000, 1000000)
tick_labels = ["0L", "10L", "20L", "30L", "40L", "50L", "60L", "70L"]

plt.yticks(tick_values, tick_labels)

plt.show()
```



0.1.2 Scatter Chart: Plotting Sales vs Profits

- Scatter plots are used when you want to show the relationship between two facts or measures.

Now, you have the sales and profit data of different product categories across different countries. Let's try to build scatterplots to visualise the data at hand.

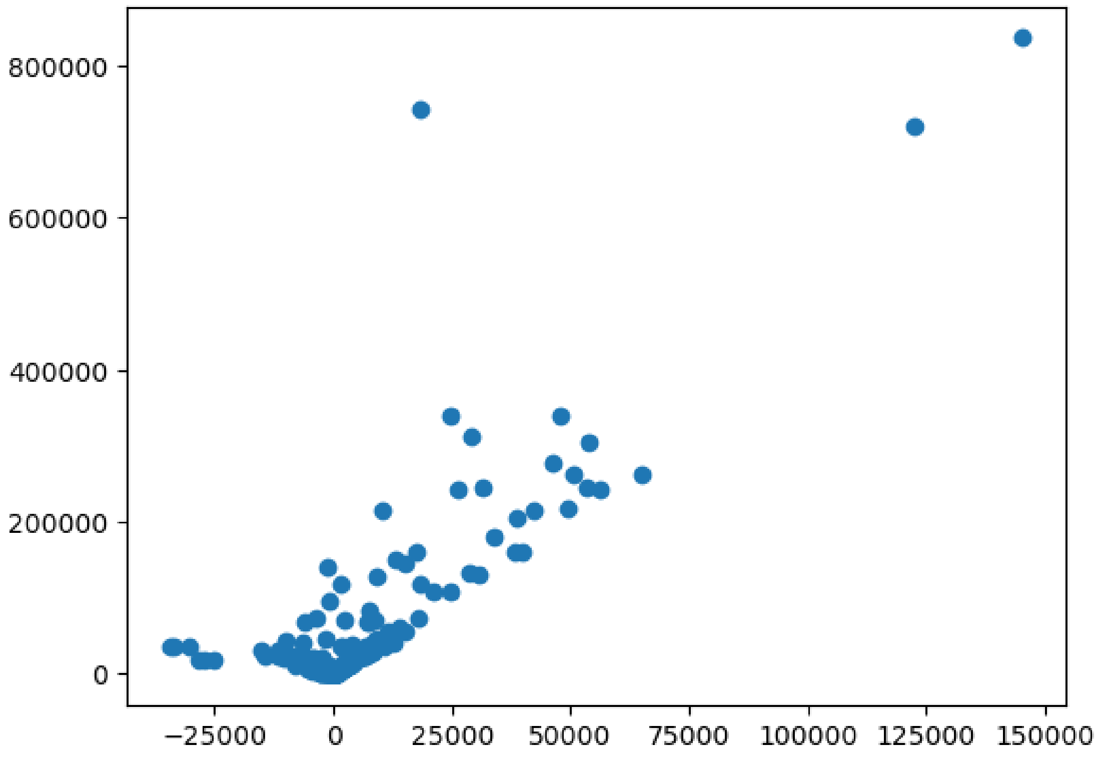
[2]:
```
# importing numpy and the pyplot package of matplotlib
import numpy as np
import matplotlib.pyplot as plt
# Sales and Profit data for different product categories across different␣
↪countries
```

```
sales = np.array ([1013.14, 8298.48, 875.51, 22320.83, 9251.6, 4516.86, 585.16,␣
 ↪836154.03, 216748.48, 174.2, 27557.79, 563.25, 558.11, 37117.45, 357.36,␣
 ↪2206.96, 709.5, 35064.03, 7230.78, 235.33, 148.32, 3973.27, 11737.8, 7104.
 ↪63, 83.67, 5569.83, 92.34, 107104.36, 1045.62, 9072.51, 42485.82, 5093.82,␣
 ↪14846.16, 943.92, 684.36, 15012.03, 38196.18, 2448.75, 28881.96, 13912.14,␣
 ↪4507.2, 4931.06, 12805.05, 67912.73, 4492.2, 1740.01, 458.04, 16904.32,␣
 ↪21744.53, 10417.26, 18665.33, 2808.42, 54195.57, 67332.5, 24390.95, 1790.43,␣
 ↪2234.19, 9917.5, 7408.14, 36051.99, 1352.22, 1907.7, 245722.14, 2154.66,␣
 ↪1078.21, 3391.65, 28262.73, 5177.04, 66.51, 2031.34, 1683.72, 1970.01, 6515.
 ↪82, 1055.31, 1029.48, 5303.4, 1850.96, 1159.41, 39989.13, 1183.87, 96365.09,␣
 ↪8356.68, 7010.24, 23119.23, 46109.28, 146071.84, 242259.03, 9058.95, 1313.
 ↪67, 31525.06, 2019.94, 703.04, 1868.79, 700.5, 55512.02, 243.5, 2113.18,␣
 ↪11781.81, 262189.49, 3487.29, 513.12, 312050.42, 5000.7, 121.02, 1302.78,␣
 ↪169.92, 124.29, 57366.05, 29445.93, 4614.3, 45009.98, 309.24, 3353.67, 41348.
 ↪34, 2280.27, 61193.7, 1466.79, 12419.94, 445.12, 25188.65, 263514.92, 12351.
 ↪23, 1152.3, 26298.81, 9900.78, 5355.57, 2325.66, 6282.81, 127707.92, 1283.1,␣
 ↪3560.15, 3723.84, 13715.01, 4887.9, 3396.89, 33348.42, 625.02, 1665.48,␣
 ↪32486.97, 340212.44, 20516.22, 8651.16, 13590.06, 2440.35, 6462.57, 1770.13,␣
 ↪7527.18, 1433.65, 423.3, 21601.72, 10035.72, 2378.49, 3062.38, 719469.32,␣
 ↪179366.79, 345.17, 30345.78, 300.71, 940.81, 36468.08, 1352.85, 1755.72,␣
 ↪2391.96, 19.98, 19792.8, 15633.88, 7.45, 521.67, 1118.24, 7231.68, 12399.32,␣
 ↪204.36, 23.64, 5916.48, 313.98, 108181.5, 9212.42, 27476.91, 1761.33, 289.5,␣
 ↪780.3, 15098.46, 813.27, 47.55, 8323.23, 22634.64, 1831.02, 28808.1, 10539.
 ↪78, 588.99, 939.78, 7212.41, 15683.01, 41369.09, 5581.6, 403.36, 375.26,␣
 ↪12276.66, 15393.56, 76.65, 5884.38, 18005.49, 3094.71, 43642.78, 35554.83,␣
 ↪22977.11, 1026.33, 665.28, 9712.49, 6038.52, 30756.51, 3758.25, 4769.49,␣
 ↪2463.3, 160153.16, 967.11, 2311.74, 1414.83, 12764.91, 4191.24, 110.76, 637.
 ↪34, 1195.12, 2271.63, 804.12, 196.17, 167.67, 131.77, 2842.05, 9969.12, 1784.
 ↪35, 3098.49, 25005.54, 1300.1, 118697.39, 7920.54, 6471.78, 31707.57, 37636.
 ↪47, 118777.77, 131170.76, 3980.88, 3339.39, 26563.9, 4038.73, 124.8, 196.65,␣
 ↪2797.77, 29832.76, 184.84, 79.08, 8047.83, 205313.25, 1726.98, 899.73, 224.
 ↪06, 304763.54, 6101.31, 729.6, 896.07, 17.82, 26.22, 46429.78, 31167.27,␣
 ↪2455.94, 37714.3, 1506.93, 3812.78, 25223.34, 3795.96, 437.31, 41278.86,␣
 ↪2091.81, 6296.61, 468.82, 23629.64, 160435.53, 9725.46, 1317.03, 1225.26,␣
 ↪30034.08, 7893.45, 2036.07, 215.52, 3912.42, 82783.43, 253.14, 966.96, 3381.
 ↪26, 164.07, 1984.23, 75.12, 25168.17, 3295.53, 991.12, 10772.1, 44.16, 1311.
 ↪45, 35352.57, 245783.54, 20.49, 13471.06, 8171.16, 14075.67, 611.82, 3925.
 ↪56, 981.84, 10209.84, 156.56, 243.06, 21287.52, 7300.51, 434.52, 6065.0,␣
 ↪741577.51, 132461.03, 224.75, 28953.6, 757.98, 528.15, 34922.41, 50.58, 2918.
 ↪48, 1044.96, 22195.13, 3951.48, 6977.64, 219.12, 5908.38, 10987.46, 4852.26,␣
 ↪445.5, 71860.82, 14840.45, 24712.08, 1329.9, 1180.44, 85.02, 10341.63, 690.
 ↪48, 1939.53, 20010.51, 914.31, 25223.82, 12804.66, 2124.24, 602.82, 2961.66,␣
 ↪15740.79, 74138.35, 7759.39, 447.0, 2094.84, 22358.95, 21734.53, 4223.73,␣
 ↪17679.53, 1019.85, 51848.72, 69133.3, 30146.9, 705.48, 14508.88, 7489.38,␣
 ↪20269.44, 246.12, 668.13, 768.93, 215677.35, 899.16, 2578.2, 4107.99, 20334.
 ↪57, 366.84, 3249.27, 98.88, 3497.88, 3853.05, 786.75, 1573.68, 458.36, 1234.
```

```
profit = np.array([-1213.46, 1814.13, -1485.7, -2286.73, -2872.12, 946.8, 198.
 ↪48, 145454.95, 49476.1, -245.56, 5980.77, -790.47, -895.72, -34572.08, 117.
 ↪9, 561.96, 152.85, 1426.05, 1873.17, -251.03, 68.22, 635.11, 3722.4, -3168.
 ↪63, 27.6, 952.11, 7.38, 20931.13, 186.36, -5395.38, 9738.45, 525.27, 3351.
 ↪99, 120.78, 266.88, 3795.21, 8615.97, 609.54, 7710.57, 2930.43, 1047.96,␣
 ↪-2733.32, 2873.73, -5957.89, -909.6, 163.41, -376.02, -6322.68, -10425.86,␣
 ↪2340.36, -28430.53, 756.12, 12633.33, 7382.54, -14327.69, 436.44, 683.85,␣
 ↪-694.91, 1960.56, 10925.82, 334.08, 425.49, 53580.2, 1024.56, 110.93, 632.
 ↪22, 8492.58, 1418.88, 19.26, -2567.57, 346.26, 601.86, 1318.68, 304.05, 428.
 ↪37, 1416.24, -2878.18, 283.41, 12611.04, 261.95, -648.43, 1112.88, -2640.29,␣
 ↪6154.32, 11558.79, 15291.4, 56092.65, 1515.39, 342.03, -10865.66, -902.8,␣
 ↪351.52, 364.17, 87.72, 11565.66, 75.4, 289.33, 3129.63, 50795.72, 783.72,␣
 ↪215.46, 29196.89, 1147.26, 53.22, 286.56, 73.02, 42.24, 13914.85, 5754.54,␣
 ↪998.04, -1476.04, 86.58, -1636.35, 10511.91, 647.34, 13768.62, 338.67, 3095.
 ↪67, 173.84, 5632.93, 64845.11, 3297.33, 338.61, 7246.62, 2255.52, 1326.36,␣
 ↪827.64, 1100.58, 9051.36, 412.23, 1063.91, 940.59, 3891.84, 1599.51, 1129.
 ↪57, 8792.64, 6.24, 592.77, 8792.85, 47727.5, -4597.68, 2242.56, 3546.45, 321.
 ↪87, 1536.72, -2463.29, 1906.08, -1916.99, 186.24, 3002.05, -3250.98, 554.7,␣
 ↪830.64, 122612.79, 33894.21, -559.03, 7528.05, -477.67, -1660.25, -33550.96,␣
 ↪481.68, 425.08, 450.3, 9.57, -3025.29, 2924.62, -11.84, 87.36, 26.51, 1727.
 ↪19, -6131.18, 59.16, 3.06, 1693.47, 74.67, 24729.21, -4867.94, 6705.18, 410.
 ↪79, 70.74, 101.7, 3264.3, 137.01, 6.18, 2100.21, 5295.24, 520.29, 7205.52,␣
 ↪2602.65, 116.67, 224.91, -5153.93, 3882.69, -6535.24, -1254.1, 84.56, -186.
 ↪38, -3167.2, -7935.59, 37.02, 1908.06, -27087.84, 829.32, 8727.44, 2011.47,␣
 ↪-11629.64, 234.96, 53.1, 1248.14, 1511.07, 7374.24, 1193.28, 1090.23, 553.
 ↪86, 38483.86, 255.81, 528.54, 326.07, 3924.36, 1018.92, 36.48, 113.24, -1770.
 ↪05, 527.64, 224.49, 79.53, 64.77, 38.08, 868.08, 2265.06, -2643.62, 833.73,␣
 ↪5100.03, 326.44, 18158.84, 1682.01, -3290.22, 8283.33, 7926.18, 1694.41,␣
 ↪30522.92, 1214.07, 900.6, -6860.8, -865.91, 26.16, 47.22, 863.52, 7061.26,␣
 ↪73.92, 33.12, 1801.23, 38815.44, 431.13, 216.81, 16.5, 53688.2, 1210.32, 236.
 ↪94, 210.84, 3.18, 2.22, 10265.64, 7212.3, 343.56, 3898.28, 568.11, -1867.85,␣
 ↪5782.38, 697.29, -192.06, 10179.02, 616.32, 1090.47, 165.84, 6138.28, 39723.
 ↪06, 2085.14, 90.0, 129.93, 7957.53, 2131.86, 562.44, 99.12, 1298.37, 7580.
 ↪33, 113.73, 139.71, 456.0, 21.24, 292.68, 30.34, 5817.15, 1060.89, 252.9,␣
 ↪3060.61, 6.6, 219.09, 8735.82, 31481.09, 2.85, -3124.72, 2195.94, 3464.7,␣
 ↪141.12, 1125.69, -1752.03, 3281.52, -303.77, 114.18, -2412.63, -5099.61, 146.
 ↪64, 660.22, 18329.28, 28529.84, -232.27, 7435.41, -1157.94, -746.73, -30324.
 ↪2, 2.52, 1313.44, 213.72, -5708.95, 930.18, 1663.02, 31.59, 1787.88, -8219.
 ↪56, 973.92, 4.32, 8729.78, -2529.52, 5361.06, 69.21, 519.3, 13.56, 2236.77,␣
 ↪213.96, 367.98, 5074.2, 206.61, 7620.36, 2093.19, 164.07, 230.01, -815.82,␣
 ↪4226.7, -3635.09, -3344.17, 167.26, 143.79, -8233.57, -4085.21, 919.35,␣
 ↪-25232.35, 234.33, 12040.68, 7206.28, -15112.76, 206.04, -2662.49, 2346.81,␣
 ↪4461.36, 93.48, 82.11, 147.87, 10389.53, 395.58, 474.74, 1333.26, 3913.02,␣
 ↪117.36, 858.78, 6.9, -4628.49, 1170.6, 218.55, 539.58, -211.0, 438.87, 317.
 ↪16, 310.8, -1578.09, 706.56, 6617.4, 803.84, 2475.26, 764.34, -1461.88, 3805.
 ↪56, 7371.27, -1377.13, 42435.03, 472.47, 315.48, -11755.91, -2418.6, 6.36,␣
 ↪9317.76, 326.88, -287.31, 637.68, 17579.17, 70.83, 47.4, 26143.92, 1548.15,␣
 ↪612.78, 17842.76, 6735.39, 1206.5, -10035.74, 149.4, -777.85, 5566.29, 748.
 ↪92, 14941.58, 348.93, 1944.06, -5.51, 7026.84, 46114.92, 2361.86, 2613.24,␣
```

Plotting a scatterplot

[4]:
```
# plotting scatterplot
plt.scatter(profit,sales)

# necessary command to display graph
plt.show()
```



[5]:
```
# Sales and Profit data for different product categories across different␣
  ↪countries
```

```
sales = np.array ([1013.14, 8298.48, 875.51, 22320.83, 9251.6, 4516.86, 585.16,␣
 ↪836154.03, 216748.48, 174.2, 27557.79, 563.25, 558.11, 37117.45, 357.36,␣
 ↪2206.96, 709.5, 35064.03, 7230.78, 235.33, 148.32, 3973.27, 11737.8, 7104.
 ↪63, 83.67, 5569.83, 92.34, 107104.36, 1045.62, 9072.51, 42485.82, 5093.82,␣
 ↪14846.16, 943.92, 684.36, 15012.03, 38196.18, 2448.75, 28881.96, 13912.14,␣
 ↪4507.2, 4931.06, 12805.05, 67912.73, 4492.2, 1740.01, 458.04, 16904.32,␣
 ↪21744.53, 10417.26, 18665.33, 2808.42, 54195.57, 67332.5, 24390.95, 1790.43,␣
 ↪2234.19, 9917.5, 7408.14, 36051.99, 1352.22, 1907.7, 245722.14, 2154.66,␣
 ↪1078.21, 3391.65, 28262.73, 5177.04, 66.51, 2031.34, 1683.72, 1970.01, 6515.
 ↪82, 1055.31, 1029.48, 5303.4, 1850.96, 1159.41, 39989.13, 1183.87, 96365.09,␣
 ↪8356.68, 7010.24, 23119.23, 46109.28, 146071.84, 242259.03, 9058.95, 1313.
 ↪67, 31525.06, 2019.94, 703.04, 1868.79, 700.5, 55512.02, 243.5, 2113.18,␣
 ↪11781.81, 262189.49, 3487.29, 513.12, 312050.42, 5000.7, 121.02, 1302.78,␣
 ↪169.92, 124.29, 57366.05, 29445.93, 4614.3, 45009.98, 309.24, 3353.67, 41348.
 ↪34, 2280.27, 61193.7, 1466.79, 12419.94, 445.12, 25188.65, 263514.92, 12351.
 ↪23, 1152.3, 26298.81, 9900.78, 5355.57, 2325.66, 6282.81, 127707.92, 1283.1,␣
 ↪3560.15, 3723.84, 13715.01, 4887.9, 3396.89, 33348.42, 625.02, 1665.48,␣
 ↪32486.97, 340212.44, 20516.22, 8651.16, 13590.06, 2440.35, 6462.57, 1770.13,␣
 ↪7527.18, 1433.65, 423.3, 21601.72, 10035.72, 2378.49, 3062.38, 719469.32,␣
 ↪179366.79, 345.17, 30345.78, 300.71, 940.81, 36468.08, 1352.85, 1755.72,␣
 ↪2391.96, 19.98, 19792.8, 15633.88, 7.45, 521.67, 1118.24, 7231.68, 12399.32,␣
 ↪204.36, 23.64, 5916.48, 313.98, 108181.5, 9212.42, 27476.91, 1761.33, 289.5,␣
 ↪780.3, 15098.46, 813.27, 47.55, 8323.23, 22634.64, 1831.02, 28808.1, 10539.
 ↪78, 588.99, 939.78, 7212.41, 15683.01, 41369.09, 5581.6, 403.36, 375.26,␣
 ↪12276.66, 15393.56, 76.65, 5884.38, 18005.49, 3094.71, 43642.78, 35554.83,␣
 ↪22977.11, 1026.33, 665.28, 9712.49, 6038.52, 30756.51, 3758.25, 4769.49,␣
 ↪2463.3, 160153.16, 967.11, 2311.74, 1414.83, 12764.91, 4191.24, 110.76, 637.
 ↪34, 1195.12, 2271.63, 804.12, 196.17, 167.67, 131.77, 2842.05, 9969.12, 1784.
 ↪35, 3098.49, 25005.54, 1300.1, 118697.39, 7920.54, 6471.78, 31707.57, 37636.
 ↪47, 118777.77, 131170.76, 3980.88, 3339.39, 26563.9, 4038.73, 124.8, 196.65,␣
 ↪2797.77, 29832.76, 184.84, 79.08, 8047.83, 205313.25, 1726.98, 899.73, 224.
 ↪06, 304763.54, 6101.31, 729.6, 896.07, 17.82, 26.22, 46429.78, 31167.27,␣
 ↪2455.94, 37714.3, 1506.93, 3812.78, 25223.34, 3795.96, 437.31, 41278.86,␣
 ↪2091.81, 6296.61, 468.82, 23629.64, 160435.53, 9725.46, 1317.03, 1225.26,␣
 ↪30034.08, 7893.45, 2036.07, 215.52, 3912.42, 82783.43, 253.14, 966.96, 3381.
 ↪26, 164.07, 1984.23, 75.12, 25168.17, 3295.53, 991.12, 10772.1, 44.16, 1311.
 ↪45, 35352.57, 245783.54, 20.49, 13471.06, 8171.16, 14075.67, 611.82, 3925.
 ↪56, 981.84, 10209.84, 156.56, 243.06, 21287.52, 7300.51, 434.52, 6065.0,␣
 ↪741577.51, 132461.03, 224.75, 28953.6, 757.98, 528.15, 34922.41, 50.58, 2918.
 ↪48, 1044.96, 22195.13, 3951.48, 6977.64, 219.12, 5908.38, 10987.46, 4852.26,␣
 ↪445.5, 71860.82, 14840.45, 24712.08, 1329.9, 1180.44, 85.02, 10341.63, 690.
 ↪48, 1939.53, 20010.51, 914.31, 25223.82, 12804.66, 2124.24, 602.82, 2961.66,␣
 ↪15740.79, 74138.35, 7759.39, 447.0, 2094.84, 22358.95, 21734.53, 4223.73,␣
 ↪17679.53, 1019.85, 51848.72, 69133.3, 30146.9, 705.48, 14508.88, 7489.38,␣
 ↪20269.44, 246.12, 668.13, 768.93, 215677.35, 899.16, 2578.2, 4107.99, 20334.
 ↪57, 366.84, 3249.27, 98.88, 3497.88, 3853.05, 786.75, 1573.68, 458.36, 1234.
```

```
profit = np.array([-1213.46, 1814.13, -1485.7, -2286.73, -2872.12, 946.8, 198.
  ↪48, 145454.95, 49476.1, -245.56, 5980.77, -790.47, -895.72, -34572.08, 117.
  ↪9, 561.96, 152.85, 1426.05, 1873.17, -251.03, 68.22, 635.11, 3722.4, -3168.
  ↪63, 27.6, 952.11, 7.38, 20931.13, 186.36, -5395.38, 9738.45, 525.27, 3351.
  ↪99, 120.78, 266.88, 3795.21, 8615.97, 609.54, 7710.57, 2930.43, 1047.96,␣
  ↪-2733.32, 2873.73, -5957.89, -909.6, 163.41, -376.02, -6322.68, -10425.86,␣
  ↪2340.36, -28430.53, 756.12, 12633.33, 7382.54, -14327.69, 436.44, 683.85,␣
  ↪-694.91, 1960.56, 10925.82, 334.08, 425.49, 53580.2, 1024.56, 110.93, 632.
  ↪22, 8492.58, 1418.88, 19.26, -2567.57, 346.26, 601.86, 1318.68, 304.05, 428.
  ↪37, 1416.24, -2878.18, 283.41, 12611.04, 261.95, -648.43, 1112.88, -2640.29,␣
  ↪6154.32, 11558.79, 15291.4, 56092.65, 1515.39, 342.03, -10865.66, -902.8,␣
  ↪351.52, 364.17, 87.72, 11565.66, 75.4, 289.33, 3129.63, 50795.72, 783.72,␣
  ↪215.46, 29196.89, 1147.26, 53.22, 286.56, 73.02, 42.24, 13914.85, 5754.54,␣
  ↪998.04, -1476.04, 86.58, -1636.35, 10511.91, 647.34, 13768.62, 338.67, 3095.
  ↪67, 173.84, 5632.93, 64845.11, 3297.33, 338.61, 7246.62, 2255.52, 1326.36,␣
  ↪827.64, 1100.58, 9051.36, 412.23, 1063.91, 940.59, 3891.84, 1599.51, 1129.
  ↪57, 8792.64, 6.24, 592.77, 8792.85, 47727.5, -4597.68, 2242.56, 3546.45, 321.
  ↪87, 1536.72, -2463.29, 1906.08, -1916.99, 186.24, 3002.05, -3250.98, 554.7,␣
  ↪830.64, 122612.79, 33894.21, -559.03, 7528.05, -477.67, -1660.25, -33550.96,␣
  ↪481.68, 425.08, 450.3, 9.57, -3025.29, 2924.62, -11.84, 87.36, 26.51, 1727.
  ↪19, -6131.18, 59.16, 3.06, 1693.47, 74.67, 24729.21, -4867.94, 6705.18, 410.
  ↪79, 70.74, 101.7, 3264.3, 137.01, 6.18, 2100.21, 5295.24, 520.29, 7205.52,␣
  ↪2602.65, 116.67, 224.91, -5153.93, 3882.69, -6535.24, -1254.1, 84.56, -186.
  ↪38, -3167.2, -7935.59, 37.02, 1908.06, -27087.84, 829.32, 8727.44, 2011.47,␣
  ↪-11629.64, 234.96, 53.1, 1248.14, 1511.07, 7374.24, 1193.28, 1090.23, 553.
  ↪86, 38483.86, 255.81, 528.54, 326.07, 3924.36, 1018.92, 36.48, 113.24, -1770.
  ↪05, 527.64, 224.49, 79.53, 64.77, 38.08, 868.08, 2265.06, -2643.62, 833.73,␣
  ↪5100.03, 326.44, 18158.84, 1682.01, -3290.22, 8283.33, 7926.18, 1694.41,␣
  ↪30522.92, 1214.07, 900.6, -6860.8, -865.91, 26.16, 47.22, 863.52, 7061.26,␣
  ↪73.92, 33.12, 1801.23, 38815.44, 431.13, 216.81, 16.5, 53688.2, 1210.32, 236.
  ↪94, 210.84, 3.18, 2.22, 10265.64, 7212.3, 343.56, 3898.28, 568.11, -1867.85,␣
  ↪5782.38, 697.29, -192.06, 10179.02, 616.32, 1090.47, 165.84, 6138.28, 39723.
  ↪06, 2085.14, 90.0, 129.93, 7957.53, 2131.86, 562.44, 99.12, 1298.37, 7580.
  ↪33, 113.73, 139.71, 456.0, 21.24, 292.68, 30.34, 5817.15, 1060.89, 252.9,␣
  ↪3060.61, 6.6, 219.09, 8735.82, 31481.09, 2.85, -3124.72, 2195.94, 3464.7,␣
  ↪141.12, 1125.69, -1752.03, 3281.52, -303.77, 114.18, -2412.63, -5099.61, 146.
  ↪64, 660.22, 18329.28, 28529.84, -232.27, 7435.41, -1157.94, -746.73, -30324.
  ↪2, 2.52, 1313.44, 213.72, -5708.95, 930.18, 1663.02, 31.59, 1787.88, -8219.
  ↪56, 973.92, 4.32, 8729.78, -2529.52, 5361.06, 69.21, 519.3, 13.56, 2236.77,␣
  ↪213.96, 367.98, 5074.2, 206.61, 7620.36, 2093.19, 164.07, 230.01, -815.82,␣
  ↪4226.7, -3635.09, -3344.17, 167.26, 143.79, -8233.57, -4085.21, 919.35,␣
  ↪-25232.35, 234.33, 12040.68, 7206.28, -15112.76, 206.04, -2662.49, 2346.81,␣
  ↪4461.36, 93.48, 82.11, 147.87, 10389.53, 395.58, 474.74, 1333.26, 3913.02,␣
  ↪117.36, 858.78, 6.9, -4628.49, 1170.6, 218.55, 539.58, -211.0, 438.87, 317.
  ↪16, 310.8, -1578.09, 706.56, 6617.4, 803.84, 2475.26, 764.34, -1461.88, 3805.
  ↪56, 7371.27, -1377.13, 42435.03, 472.47, 315.48, -11755.91, -2418.6, 6.36,␣
  ↪9317.76, 326.88, -287.31, 637.68, 17579.17, 70.83, 47.4, 26143.92, 1548.15,␣
  ↪612.78, 17842.76, 6735.39, 1206.5, -10035.74, 149.4, -777.85, 5566.29, 748.
  ↪92, 14941.58, 348.93, 1944.06, -5.51, 7026.84, 46114.92, 2361.86, 2613.24,␣
```

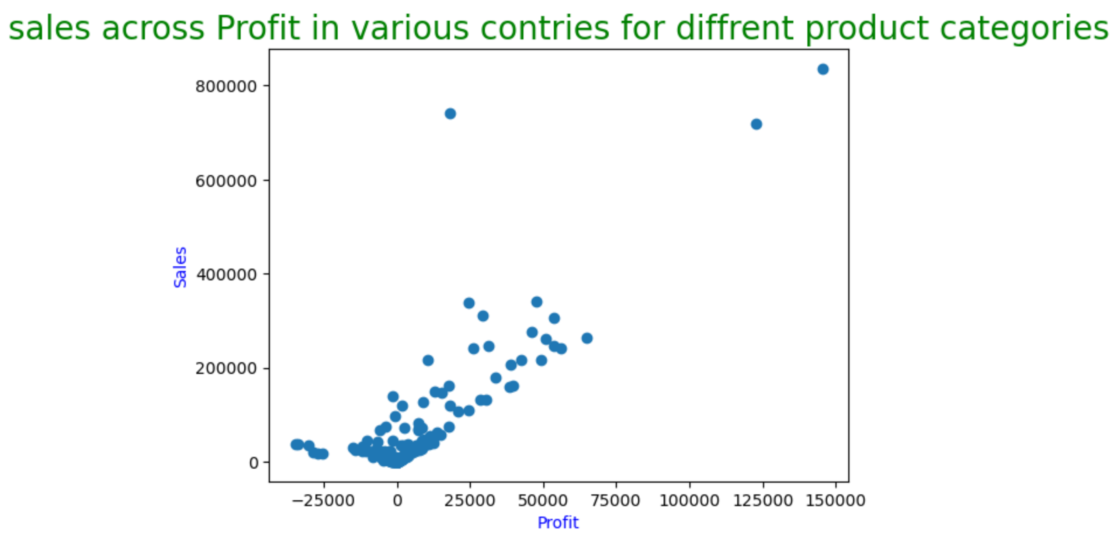
```

# corresponding category and country value to the above arrays
```

```
product_category = np.array(['Technology', 'Technology', 'Technology',␣
 ↪'Technology', 'Technology', 'Technology', 'Technology', 'Technology',␣
```

Adding title and labeling axes

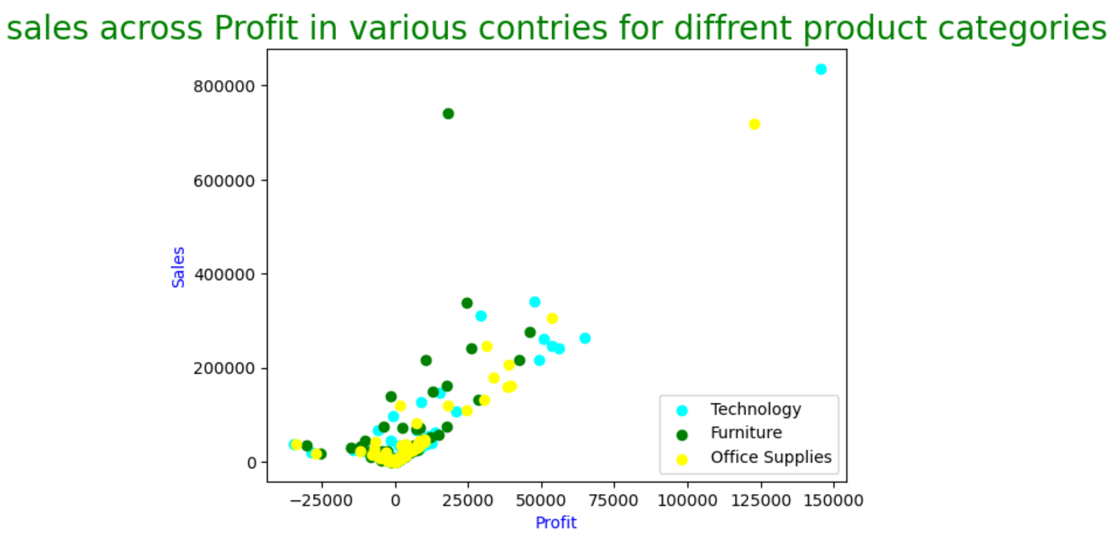
```
[6]: # plotting scatter chart
plt.scatter(profit,sales)

# Adding and formatting title
plt.title("sales across Profit in various contries for diffrent product␣
 ↪categories",fontdict={'fontsize': 20, 'fontweight' : 5, 'color' : 'Green'})

# Labeling Axes

plt.xlabel("Profit", fontdict={'fontsize': 10, 'fontweight' : 3, 'color' :␣
 ↪'Blue'})
plt.ylabel("Sales", fontdict={'fontsize': 10, 'fontweight' : 3, 'color' :␣
 ↪'Blue'})

plt.show()
```

```
 ↪'Russia', 'Romania', 'Qatar', 'Portugal', 'Poland', 'Philippines', 'Peru',␣
 ↪'Paraguay', 'Papua New Guinea', 'Panama', 'Pakistan', 'Norway', 'Nigeria',␣
 ↪'Niger', 'Nicaragua', 'New Zealand', 'Netherlands', 'Nepal', 'Namibia',␣
 ↪'Myanmar (Burma)', 'Mozambique', 'Morocco', 'Mongolia', 'Moldova', 'Mexico',␣
 ↪'Mauritania', 'Martinique', 'Mali', 'Malaysia', 'Madagascar', 'Luxembourg',␣
 ↪'Lithuania', 'Libya', 'Liberia', 'Lesotho', 'Lebanon', 'Kyrgyzstan',␣
 ↪'Kenya', 'Kazakhstan', 'Jordan', 'Japan', 'Jamaica', 'Italy', 'Israel',␣
 ↪'Ireland', 'Iraq', 'Iran', 'Indonesia', 'India', 'Hungary', 'Hong Kong',␣
 ↪'Honduras', 'Haiti', 'Guyana', 'Guinea-Bissau', 'Guinea', 'Guatemala',␣
 ↪'Guadeloupe', 'Greece', 'Ghana', 'Germany', 'Georgia', 'Gabon', 'France',␣
 ↪'Finland', 'Ethiopia', 'Estonia', 'Eritrea', 'Equatorial Guinea', 'El␣
 ↪Salvador', 'Egypt', 'Ecuador', 'Dominican Republic', 'Djibouti', 'Denmark',␣
 ↪'Democratic Republic of the Congo', 'Czech Republic', 'Cuba', 'Croatia',␣
 ↪"Cote d'Ivoire", 'Costa Rica', 'Colombia', 'China', 'Chile', 'Central␣
 ↪African Republic', 'Canada', 'Cameroon', 'Cambodia', 'Burkina Faso',␣
 ↪'Bulgaria', 'Brazil', 'Bosnia and Herzegovina', 'Bolivia', 'Benin',␣
 ↪'Belgium', 'Belarus', 'Barbados', 'Bangladesh', 'Bahrain', 'Azerbaijan',␣
```



Representing product categories using different colors

[9]:
```
product_categories = np.array(["Technology", "Furniture", "Office Supplies"])
colors = np.array(["cyan", "green", "yellow"])

# plotting the scatterplot with color coding the points belonging to different␣
 ↪categories
for color,category in zip (colors, product_categories):
    sales_category=sales[product_category == category]
    profit_category=profit[product_category == category]

    plt.scatter(profit_category,sales_category, c = color ,label = category)


# Adding and formatting title
plt.title("sales across Profit in various contries for diffrent product␣
 ↪categories",fontdict={'fontsize': 20, 'fontweight' : 5, 'color' : 'Green'})


# Labeling Axes
plt.xlabel("Profit", fontdict={'fontsize': 10, 'fontweight' : 3, 'color' :␣
 ↪'Blue'})
plt.ylabel("Sales", fontdict={'fontsize': 10, 'fontweight' : 3, 'color' :␣
 ↪'Blue'})

# Adding legend for interpretation of points
plt.legend()

plt.show()
```



Adding labels to points belonging to specific country

```
[ ]: # plotting the scatterplot with color coding the points belonging to different␣
      ↪categories


     # labeling points that belong to country "India"


     # Adding and formatting title


     # Labeling Axes


     # Adding legend for interpretation of points


     plt.show()
```

03-line

August 20, 2023

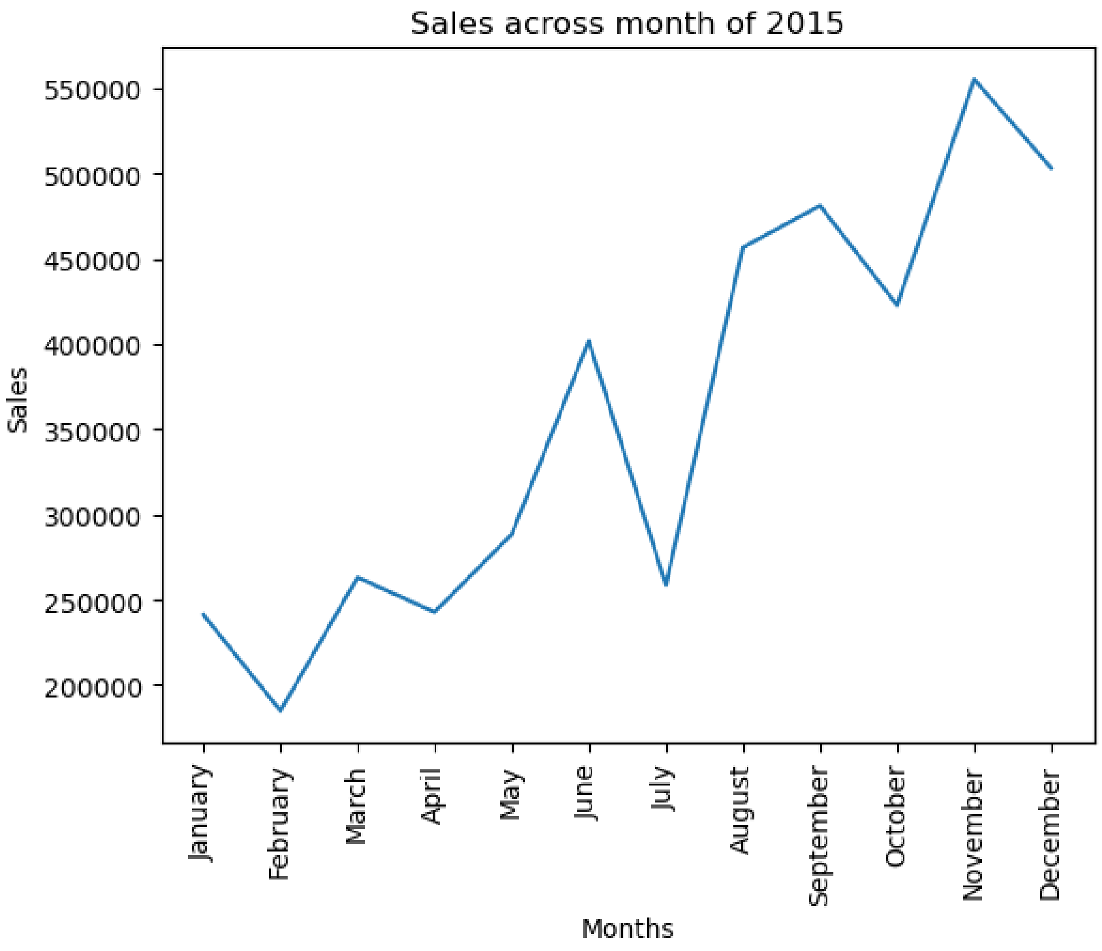
0.1 Visualisation in Python - Matplotlib

0.1.1 Line Chart: Trend of sales over the 12 months

- Can be used to present the trend with time variable on the x-axis
- In some cases, can be used as an alternative to scatterplot to understand the relationship between 2 variables

[1]:
```
# importing the required libraries
import numpy as np
import matplotlib.pyplot as plt

# Sales data across months
months = np.array(['January', 'February', 'March', 'April', 'May', 'June',␣
 ↪'July', 'August', 'September', 'October', 'November', 'December'])
sales = np.array([241268.56, 184837.36, 263100.77, 242771.86, 288401.05, 401814.
 ↪06, 258705.68, 456619.94, 481157.24, 422766.63, 555279.03, 503143.69])
```

[5]:
```
# plotting a line chart
plt.plot(months, sales)

# adding title to the chart
plt.title("Sales across month of 2015")

# labeling the axes
plt.xlabel("Months")
plt.ylabel("Sales")

# rotating the tick values of x-axis
plt.xticks(rotation=90)

# displating the created plot
plt.show()
```


Sales across month of 2015
Sales
550000
500000
450000
400000
350000
300000
250000
200000
January
February
March
April
May
June
July
August
September
October
November
December
Months

```
[ ]:
```

04-histogram-1

August 26, 2023

1 Visualisation in Python - Matplotlib

1.0.1 Histogram: Distibution of employees across different age groups

- Useful in checking the distribution of data range
- Builds a bar corresponding to each element in the data range showing its frequency

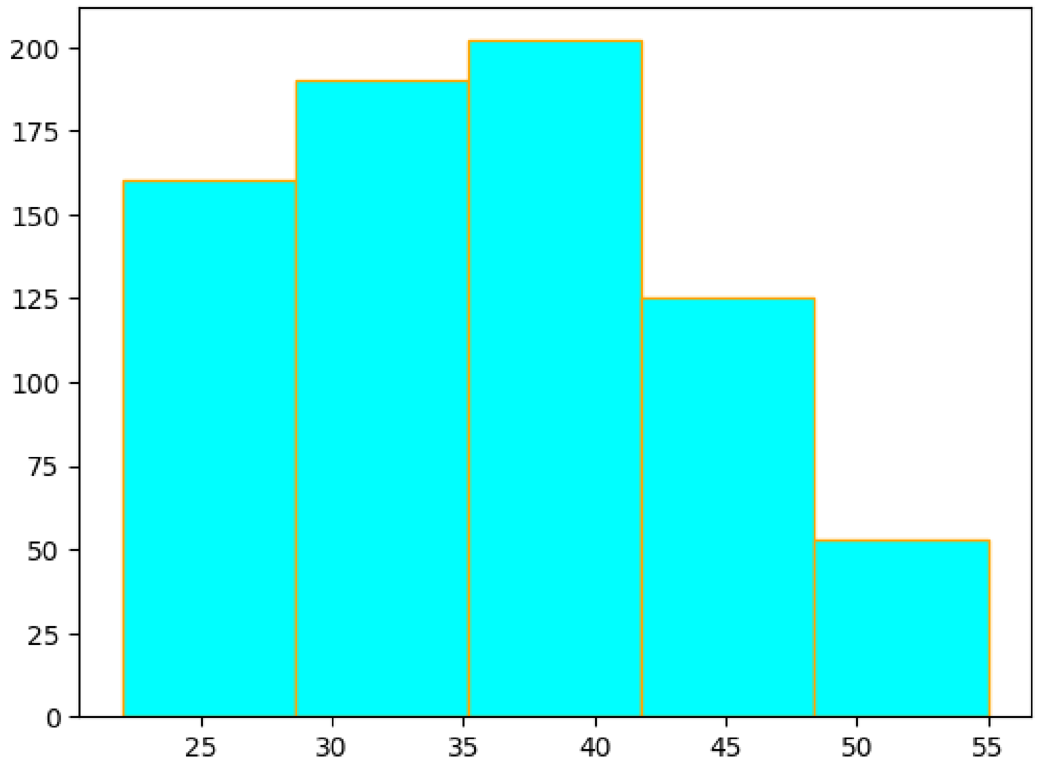
[1]:
```
# importing the required libraries - numpy, matplotlib.pyplot
import numpy as np
import matplotlib.pyplot as plt

# data corresponding to age of the employees in the company
```

```
age = np.array([23, 22, 24, 24, 23, 23, 22, 23, 24, 24, 24, 22, 24, 23, 24, 23,␣
 ↪22, 24, 23, 23, 22, 23, 23, 24, 23, 24, 23, 22, 24, 22, 23, 24, 23, 24, 22,␣
 ↪22, 24, 23, 22, 24, 24, 24, 23, 24, 24, 22, 23, 23, 24, 22, 22, 24, 22, 23,␣
 ↪22, 23, 22, 23, 23, 23, 23, 22, 22, 23, 23, 23, 23, 23, 23, 22, 29, 29, 27,␣
 ↪28, 28, 29, 28, 27, 26, 27, 28, 29, 26, 28, 26, 28, 27, 27, 28, 28, 26, 29,␣
 ↪28, 28, 26, 27, 26, 28, 27, 29, 29, 27, 27, 27, 28, 29, 29, 29, 27, 28, 28,␣
 ↪26, 28, 27, 26, 26, 27, 26, 29, 28, 28, 28, 29, 26, 26, 26, 29, 26, 28, 26,␣
 ↪28, 28, 27, 27, 27, 29, 27, 28, 27, 26, 29, 29, 27, 29, 26, 29, 26, 29, 29,␣
 ↪27, 28, 28, 27, 29, 26, 28, 26, 28, 27, 29, 29, 29, 27, 27, 29, 29, 26, 26,␣
 ↪26, 27, 28, 27, 28, 28, 29, 27, 26, 27, 29, 28, 29, 27, 27, 26, 26, 26, 26,␣
 ↪29, 28, 28, 33, 34, 33, 33, 34, 33, 31, 32, 33, 33, 32, 34, 32, 31, 33, 34,␣
 ↪31, 33, 34, 33, 34, 33, 32, 33, 31, 33, 32, 32, 31, 34, 33, 31, 34, 32, 32,␣
 ↪31, 32, 31, 32, 34, 33, 33, 31, 32, 32, 31, 32, 33, 34, 32, 34, 31, 32, 31,␣
 ↪33, 32, 34, 31, 32, 34, 31, 31, 34, 34, 34, 32, 34, 33, 33, 32, 32, 33, 31,␣
 ↪33, 31, 32, 34, 32, 32, 31, 34, 32, 32, 31, 32, 34, 32, 33, 31, 34, 31, 31,␣
 ↪32, 31, 33, 34, 34, 34, 31, 33, 34, 33, 34, 31, 34, 34, 33, 31, 32, 33, 31,␣
 ↪31, 33, 32, 34, 32, 34, 31, 31, 34, 32, 32, 31, 31, 32, 31, 31, 32, 33, 32,␣
 ↪31, 32, 32, 31, 31, 34, 31, 34, 33, 32, 31, 34, 34, 31, 34, 31, 32, 34, 33,␣
 ↪33, 34, 32, 33, 31, 31, 33, 32, 31, 31, 31, 37, 38, 37, 37, 36, 37, 36, 39,␣
 ↪37, 39, 37, 39, 38, 36, 37, 36, 38, 38, 36, 39, 39, 37, 39, 36, 37, 36, 36,␣
 ↪37, 38, 36, 38, 39, 39, 36, 38, 37, 39, 38, 39, 39, 36, 38, 37, 38, 39, 36,␣
 ↪37, 36, 36, 38, 38, 38, 39, 36, 37, 37, 39, 37, 37, 36, 36, 39, 37, 36, 36,␣
 ↪36, 39, 37, 37, 37, 37, 39, 36, 39, 37, 38, 37, 36, 36, 39, 39, 36, 36, 39,␣
 ↪39, 39, 37, 38, 36, 36, 37, 38, 37, 38, 37, 39, 39, 37, 39, 36, 36, 39, 39,␣
 ↪39, 36, 38, 39, 39, 39, 39, 38, 36, 37, 37, 38, 38, 39, 36, 37, 37, 39, 36,␣
 ↪37, 37, 36, 36, 36, 38, 39, 38, 36, 38, 36, 39, 38, 36, 36, 37, 39, 39, 37,␣
 ↪37, 37, 36, 37, 36, 36, 38, 38, 39, 36, 39, 36, 37, 37, 39, 39, 36, 38, 39,␣
 ↪39, 39, 37, 37, 37, 37, 39, 36, 37, 39, 38, 39, 36, 37, 38, 39, 38, 36, 37,␣
 ↪38, 42, 43, 44, 43, 41, 42, 41, 41, 42, 41, 43, 44, 43, 44, 44, 42, 43, 44,␣
 ↪43, 41, 44, 42, 43, 42, 42, 44, 43, 42, 41, 42, 41, 41, 41, 44, 44, 44, 41,␣
 ↪43, 42, 42, 43, 43, 44, 44, 44, 44, 44, 41, 42, 44, 43, 42, 42, 43, 44, 44,␣
 ↪44, 44, 41, 42, 43, 43, 43, 41, 43, 41, 42, 41, 42, 42, 41, 42, 44, 41, 43,␣
 ↪42, 41, 43, 41, 44, 44, 43, 43, 43, 41, 41, 41, 42, 43, 42, 48, 48, 48, 49,␣
 ↪47, 45, 46, 49, 46, 49, 49, 46, 47, 45, 47, 45, 47, 49, 47, 46, 46, 47, 45,␣
 ↪49, 49, 49, 45, 46, 47, 46, 45, 46, 45, 48, 48, 45, 49, 46, 48, 49, 47, 48,␣
 ↪45, 48, 46, 45, 48, 45, 46, 46, 48, 47, 46, 45, 48, 46, 49, 47, 46, 49, 48,␣
 ↪46, 47, 47, 46, 48, 47, 46, 46, 49, 50, 54, 53, 55, 51, 50, 51, 54, 54, 53,␣
 ↪53, 51, 51, 50, 54, 51, 51, 55, 50, 51, 50, 50, 53, 52, 54, 53, 55, 52, 52,␣
 ↪50, 52, 55, 54, 50, 50, 55, 52, 54, 52, 54])
```

[2]:
```
# Checking the number of employees
len(age)
```

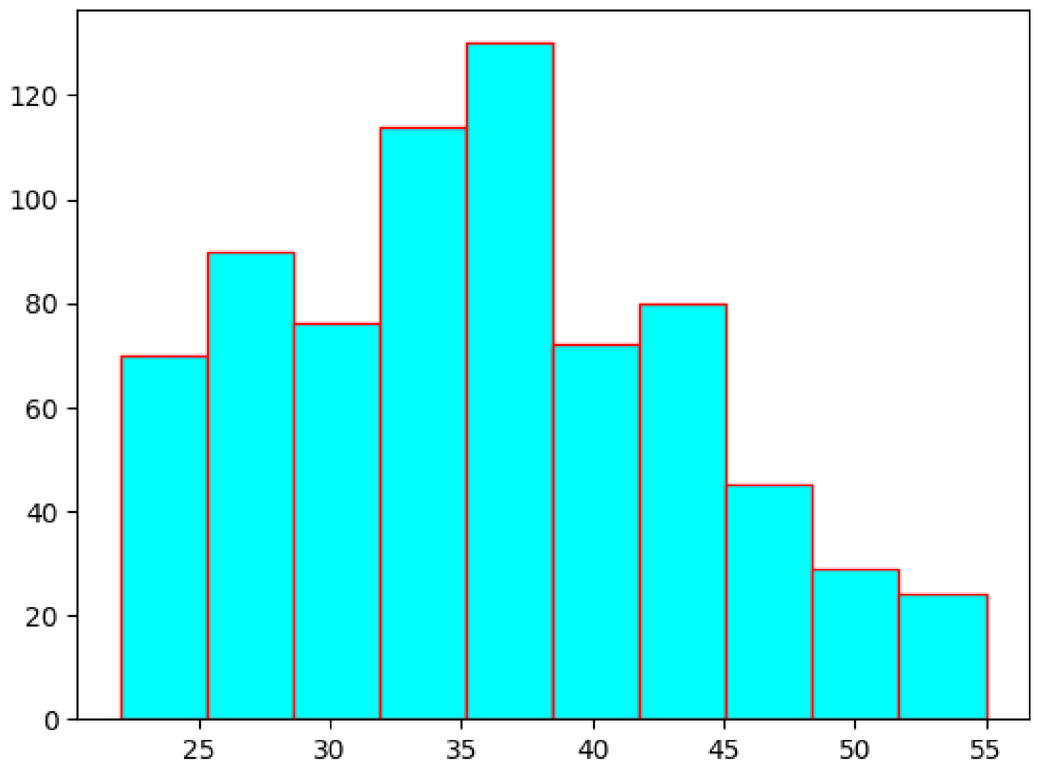
[2]:
```
730
```

[3]:
```
# plotting a histogram
plt.hist(age, bins=5,color="cyan",edgecolor="orange")
plt.show()
```



Plotting a histogram with fixed number of bins

[5]:
```
# plotting a histogram
plt.hist(age,bins=10,color="cyan",edgecolor="red")


plt.show()
```


0
20
40
60
80
100
120
25
30
35
40
45
50
55