# matlabplot

July 30, 2024

## 1 Matplotlib

### 1.1 Create a dataset

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
# Set seed for reproducibility
np.random.seed(0)
# Generate dates
date_range = pd.date_range(start='2024-01-01', end='2024-01-31', freq='D')
# Generate temperature data (in °C)
temperatures = np.random.normal(loc=15, scale=10, size=len(date_range))
# Generate sales data (in USD)
sales = np.random.normal(loc=200, scale=50, size=len(date_range))
# Generate categorical data for sales categories
categories = np.random.choice(['Electronics', 'Clothing', 'Groceries',␣
 ↪'Furniture'], size=len(date_range))

# Create a DataFrame
a = pd.DataFrame({
    'Date': date_range,
    'Temperature': temperatures,
    'Sales': sales,
    'Category': categories
})
```

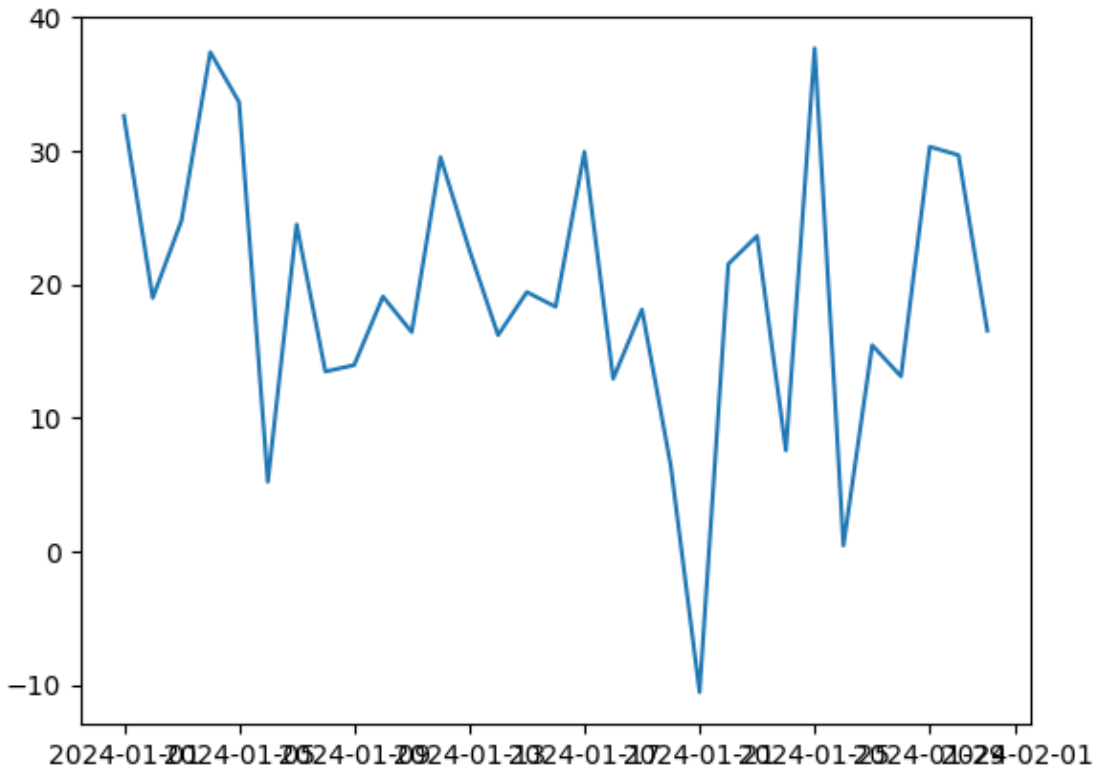
[3]:
```
import matplotlib.pyplot as plt
```

### 1.2 Line Plot

In general Line Plot are used for Trend analysis Where the Time is always at the x-axis and Values are at the y-axis

### 1.2.1 Simple Line Plot

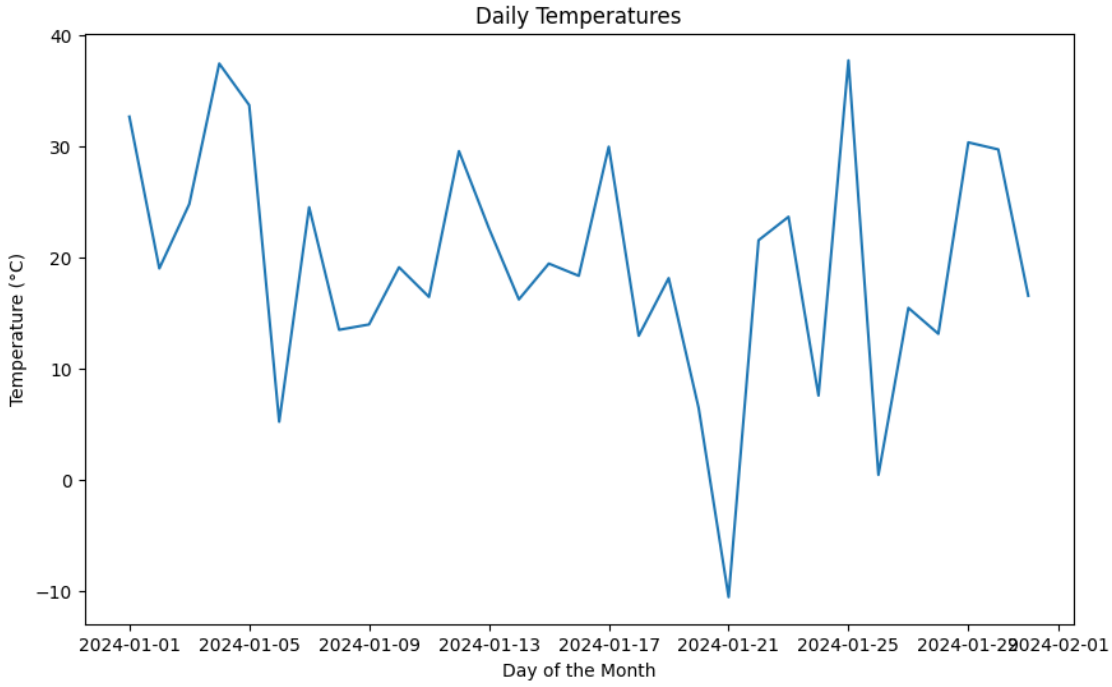
[4]:
```
plt.plot(a['Date'],a['Temperature']);
```




### 1.2.2 Labeling and Titling

Add plt.figure() To make a proper Layout Add Titles "plt.title()", Add Labels "plt.xlabel(),plt.ylabel" plt.show(); to display

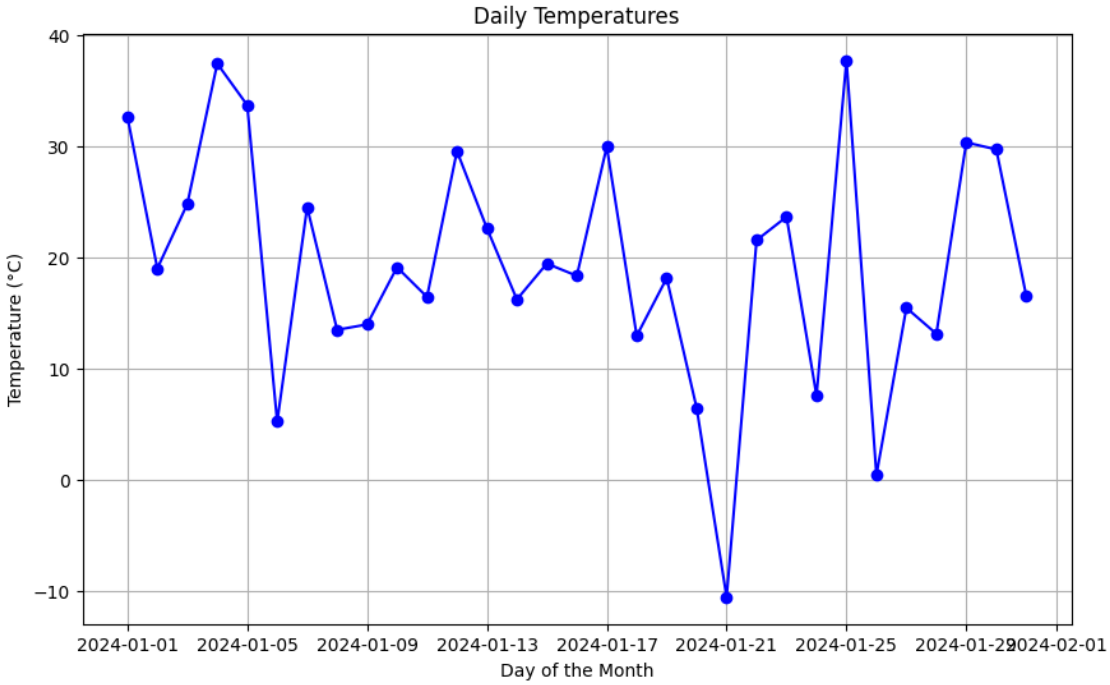
[5]:
```
plt.figure(figsize=(10, 6))
plt.plot(a['Date'], a['Temperature'])
plt.title('Daily Temperatures')
plt.xlabel('Day of the Month')
plt.ylabel('Temperature (°C)')
plt.show();
```




### 1.2.3 Adding Grids

Adding plt.figure,title and labels made the plot way better. Now we can manipulate the colors,style ,grid and marking to make it look good.

[6]:
```
plt.figure(figsize=(10, 6))
plt.plot(a['Date'], a['Temperature'], marker='o', linestyle='-', color='b')
plt.title('Daily Temperatures')
plt.xlabel('Day of the Month')
plt.ylabel('Temperature (°C)')
plt.grid(True)
plt.show()
```

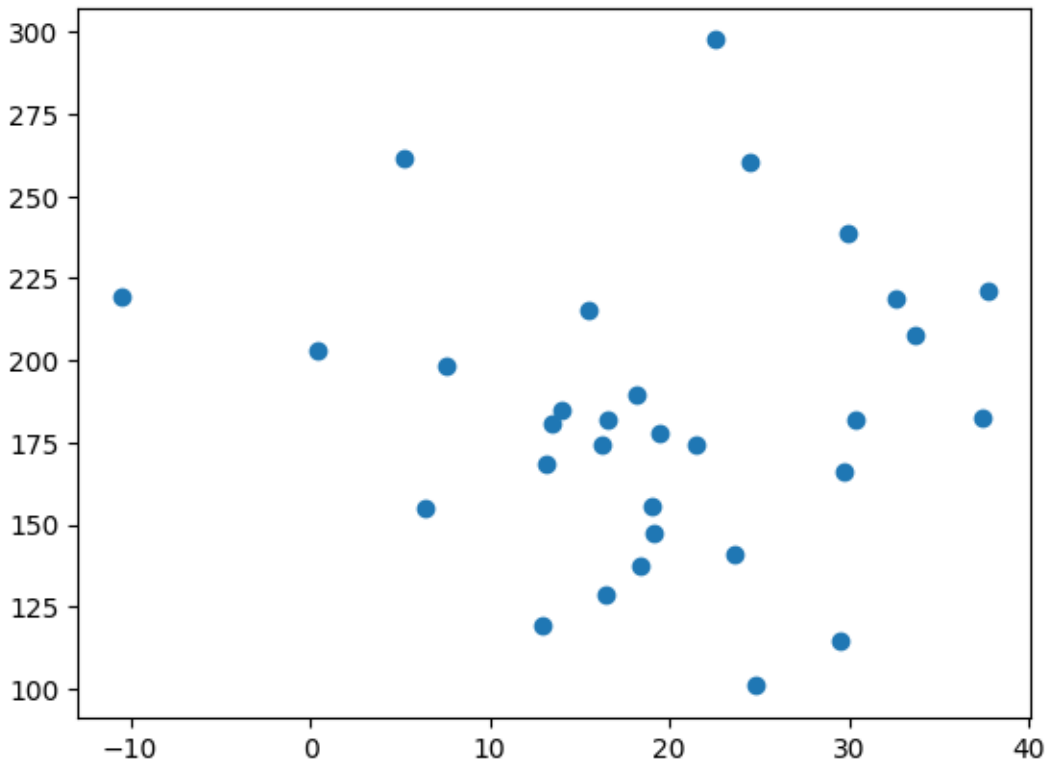



### 1.3 Scatter Plot

In general Scatter Plot are used for correlation of two numerical variables

#### 1.3.1 Simple Scatter Plot

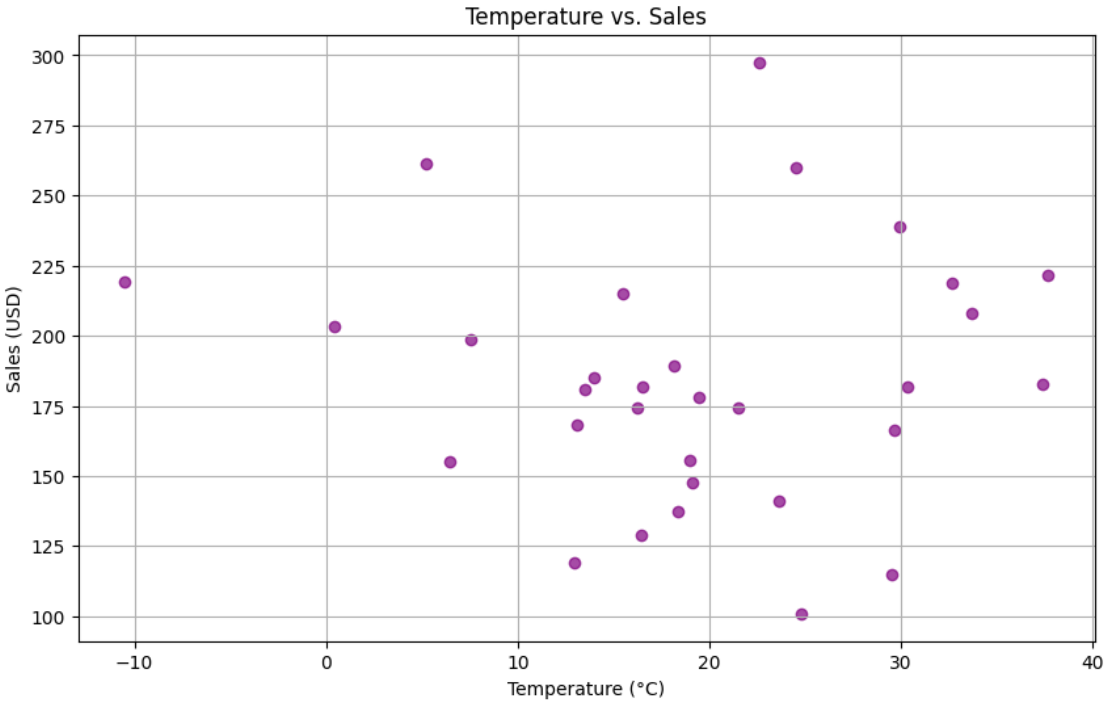
[7]:
```
plt.scatter(a['Temperature'], a['Sales']);
```




### 1.3.2 Better Scatter Plot

Adding up plt.Figure , Titles , Lables, Grid

[8]:
```
plt.figure(figsize=(10, 6))
plt.scatter(a['Temperature'], a['Sales'], c='purple', alpha=0.7)
plt.title('Temperature vs. Sales')
plt.xlabel('Temperature (°C)')
plt.ylabel('Sales (USD)')
plt.grid(True)
plt.show();
```

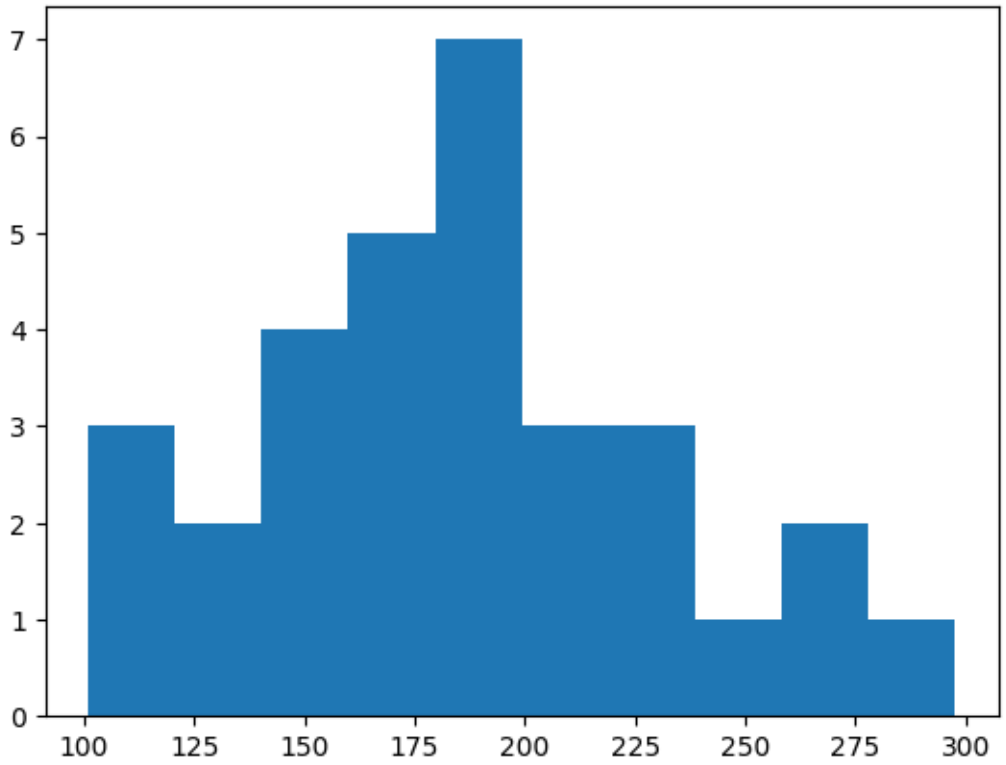



## 1.4 Histogram

Histogram in general is used for showing the distribution of the data . In General Numerical variable is used

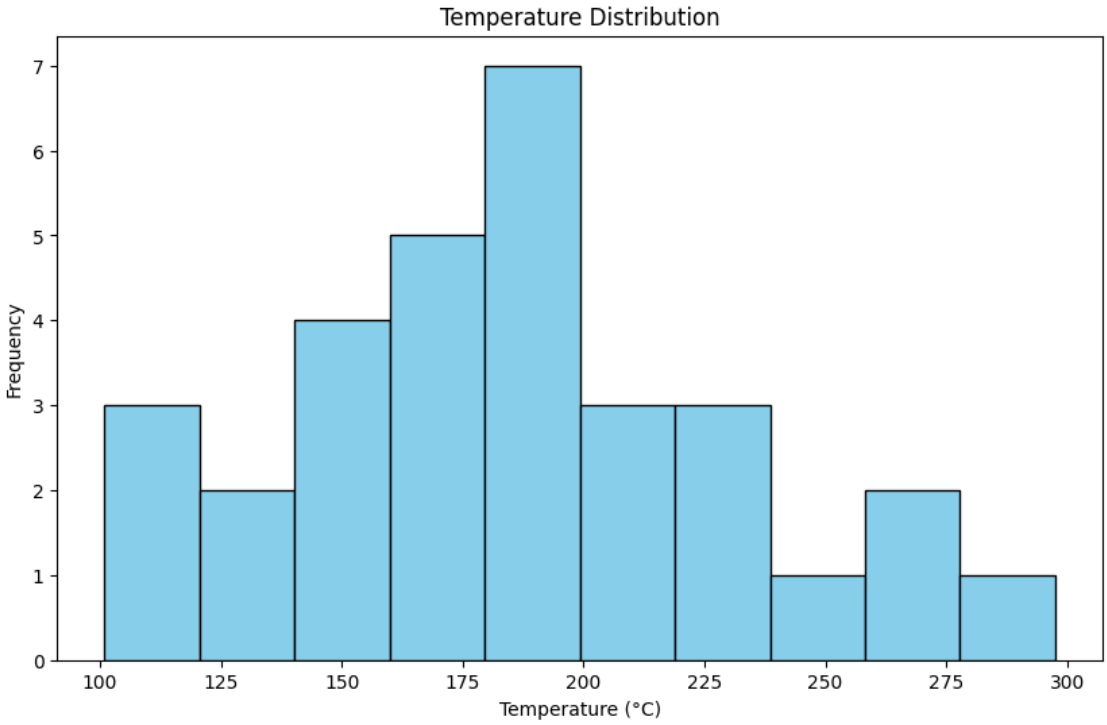
### 1.4.1 Simple Histogram

[9]:
```
plt.hist(a['Sales']);
```




### 1.4.2 Better Histogram

[10]:
```
plt.figure(figsize=(10,6))
plt.hist(a['Sales'],bins=10, color='skyblue', edgecolor='black')
plt.title('Temperature Distribution')
plt.xlabel('Temperature (°C)')
plt.ylabel('Frequency')
plt.show();
```

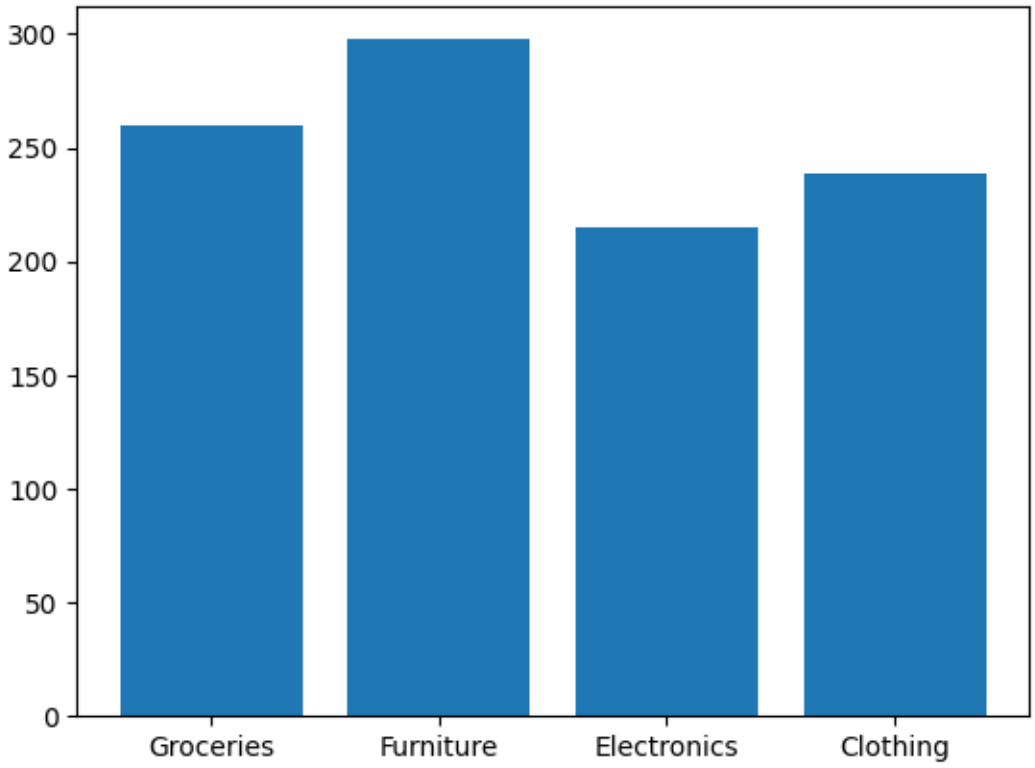



## 1.5 Bar Plot

In General Bar Plots are used for comparing Categorical Column. Catogories are in X Axis and Values are in Y axis and can be changes for horizontal Representation.

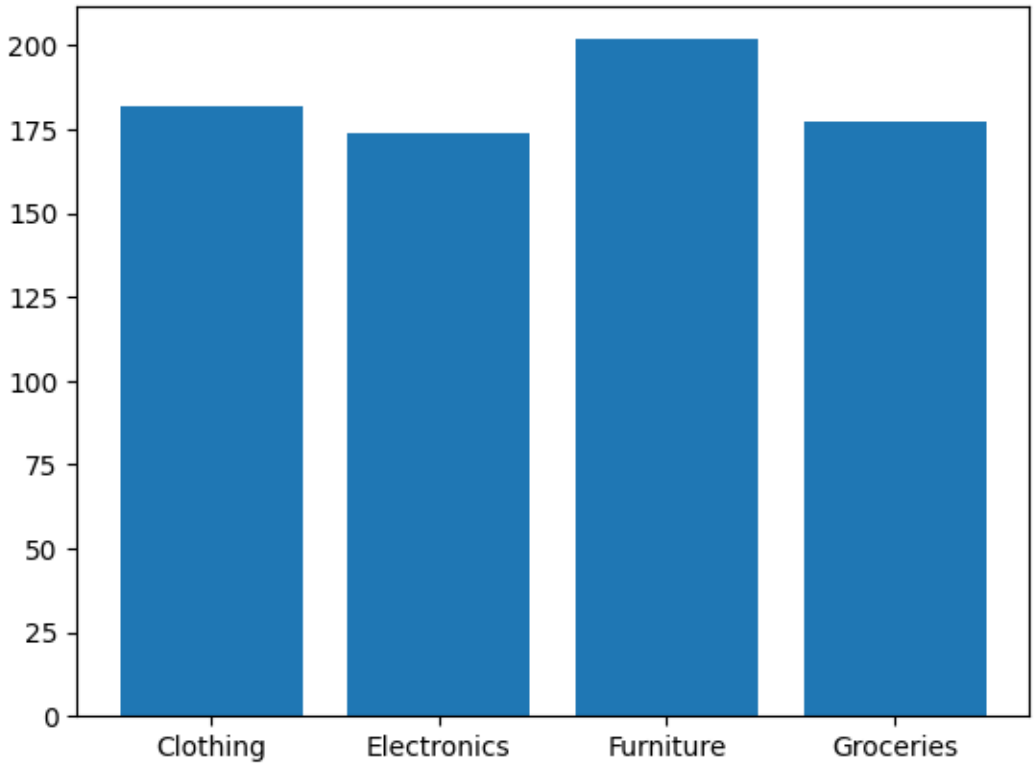
### 1.5.1 Simple Bar plot

[11]:
```
plt.bar(a['Category'],a['Sales']);
#This Plots the Max value in the category.
```




### 1.5.2 Better Bar plot

[12]:
```
# To make the bar chart more customizable we need to set the height
# If we have to plot the mean we need to groupby the category to mean and etc..
plt.bar(a.groupby('Category')['Sales'].mean().index,a.
  ↪groupby('Category')['Sales'].mean().values);
```



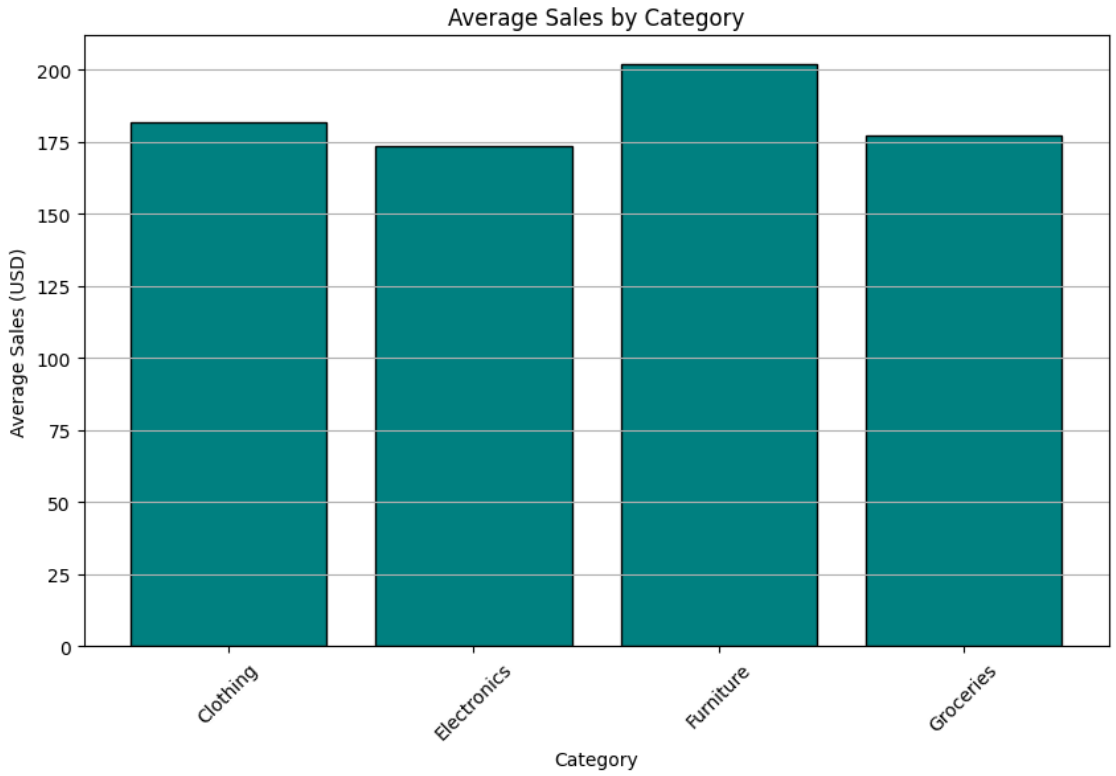

[13]:
```
# We can make it look by assigning it into variables
bi  =   a.groupby('Category')['Sales'].mean().index     # Bar Index
bv  =   a.groupby('Category')['Sales'].mean().values    # Bar Values
print(bi,"\n",bv)
```

```
Index(['Clothing', 'Electronics', 'Furniture', 'Groceries'], dtype='object',
name='Category')
 [181.98996862 173.6082667  201.82980472 177.09857042]
```

### 1.5.3 Way better Bar Chart

Adding layout, titles, labels, and grid , xticks - Change the rotation of x labels can be (changed to y labels too)

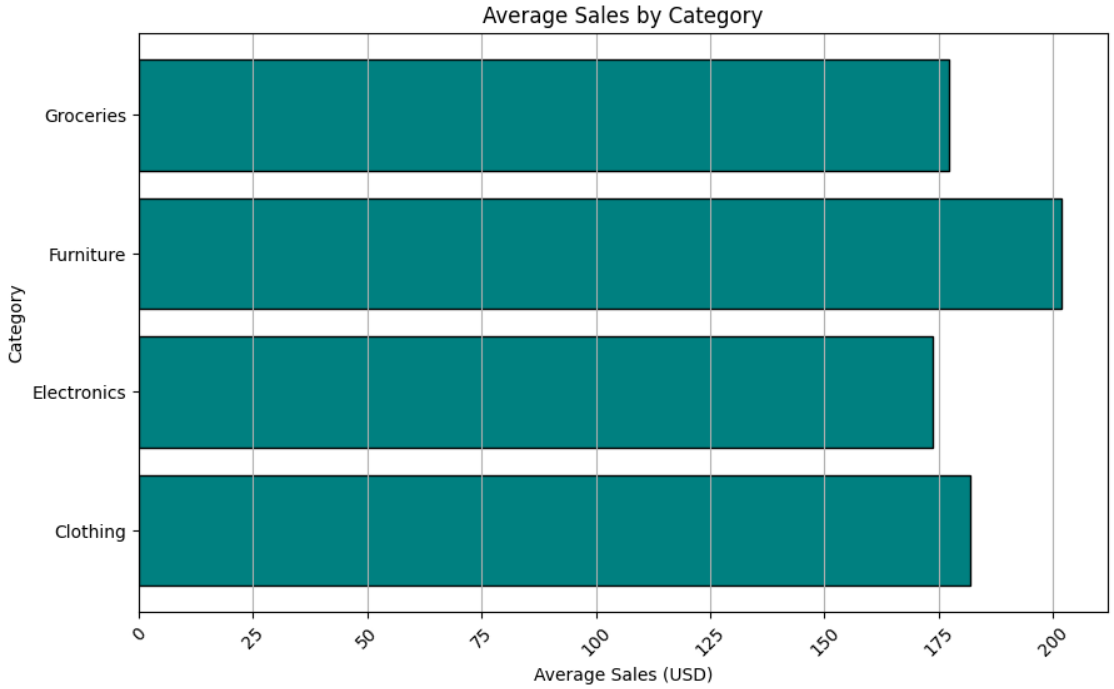
[14]:
```
plt.figure(figsize=(10,6))
plt.bar(bi,bv,color='teal', edgecolor='black')
plt.title('Average Sales by Category')
plt.xlabel('Category')
plt.ylabel('Average Sales (USD)')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```




### 1.5.4 Horizontal Bar chart

Adding one letter changes the whole vertical bars to horizontal Also Make sure of x and y labels

[15]:
```
plt.figure(figsize=(10,6))
plt.barh(bi,bv,color='teal', edgecolor='black')
plt.title('Average Sales by Category')
plt.ylabel('Category')
plt.xlabel('Average Sales (USD)')
plt.xticks(rotation=45)
plt.grid(axis='x')
plt.show()
```

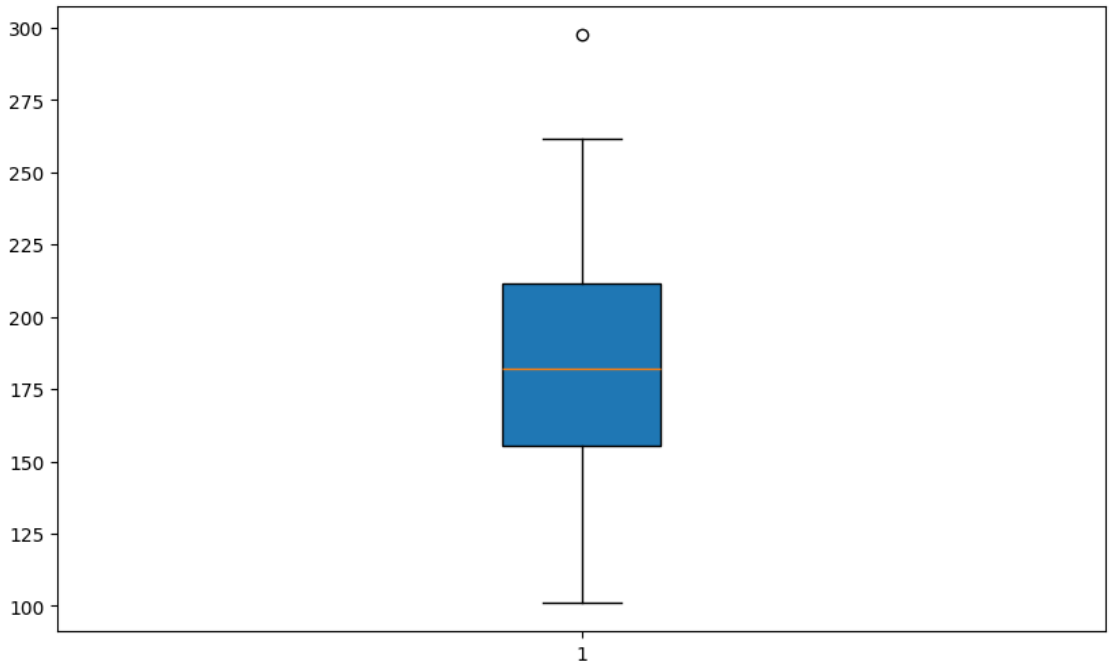



## 1.6 Box Plot

In general box plot is used to visualize the spread of the numerical data through their quartiles. The plot is also called the box-and-whisker plot. Outliers that differ significantly from the rest of the dataset may be plotted as individual points beyond the whiskers on the box-plot.

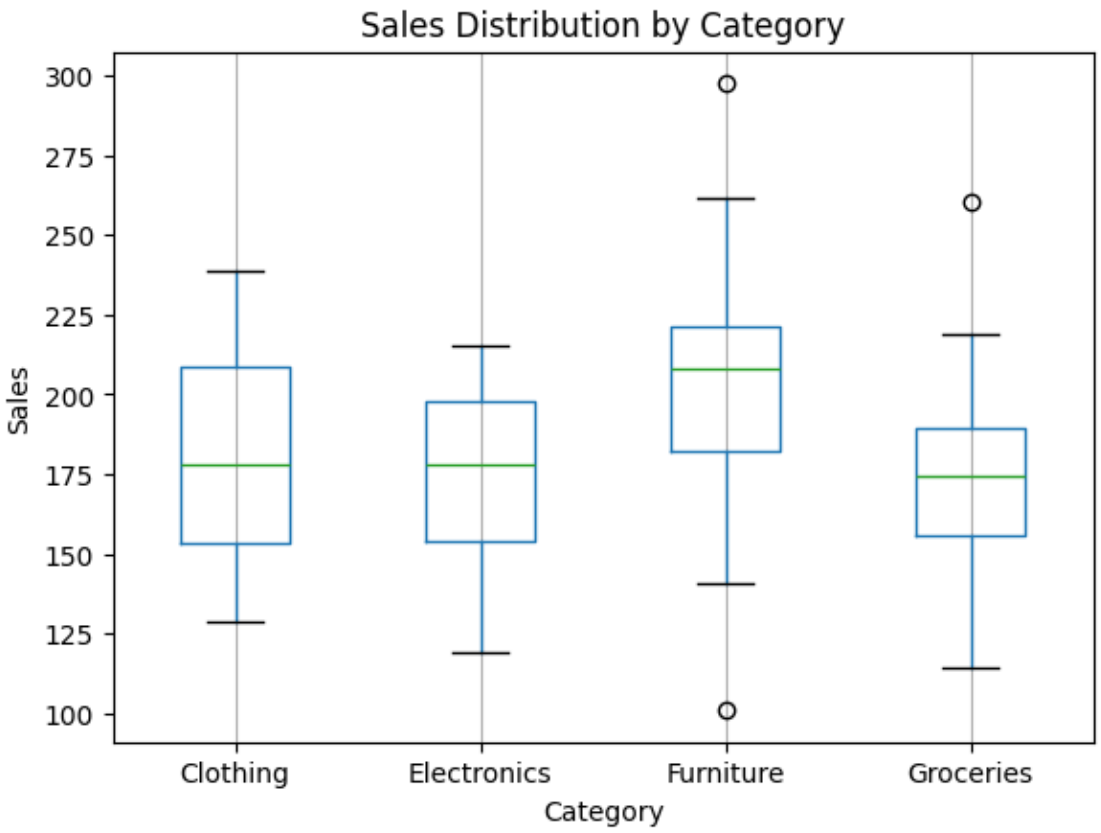
### 1.6.1 Simple Box Plot

[16]:
```
plt.figure(figsize=(10, 6));
plt.boxplot(x=a['Sales'],patch_artist=True);
```




[17]:
```
plt.figure(figsize=(10, 6))
a.boxplot(column='Sales',by='Category');
plt.title('Sales Distribution by Category')
plt.suptitle('')  # Suppress the default title
plt.xlabel('Category')
plt.ylabel('Sales')
plt.grid(axis='y')
plt.show();
```

```
<Figure size 1000x600 with 0 Axes>
```

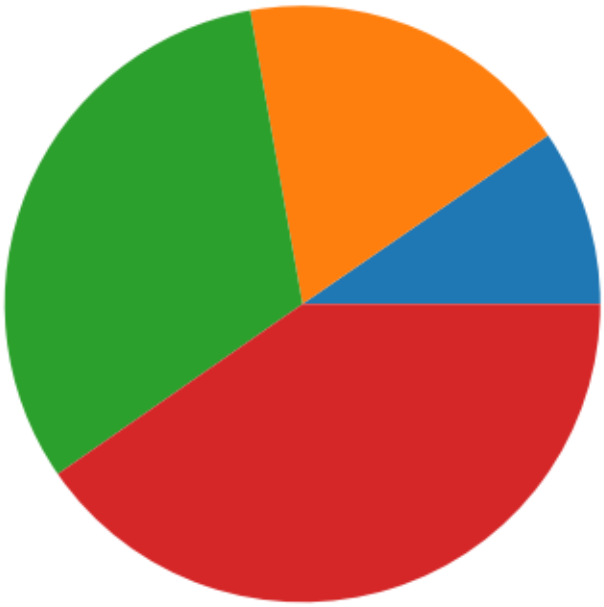



## 1.7 Pie Chart

In general Pie Charts are used for visualisation of the proportion of the category

### 1.7.1 Simple pie Chart

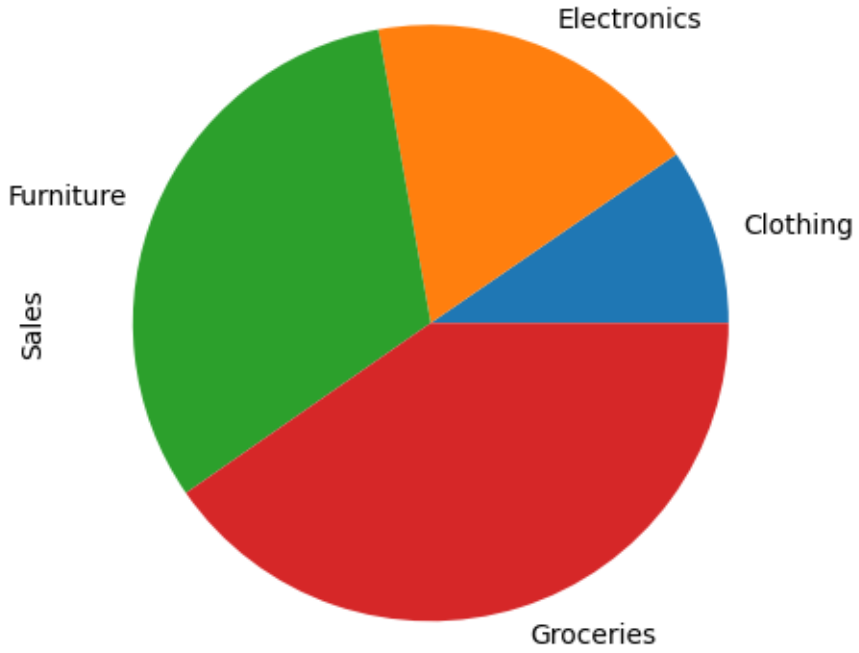
[18]:
```
plt.figure(figsize=(5,6))
plt.pie(a.groupby(['Category'])['Sales'].sum());
```



### 1.7.2 Better Pie

If we plot using pandas pie plot we get auto labeling

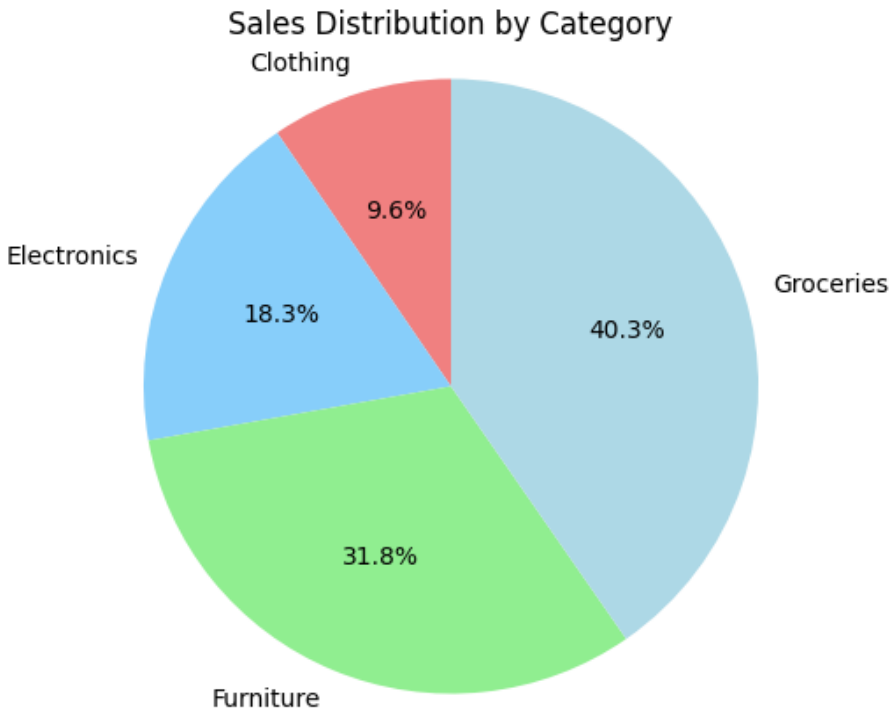
[19]:
```
plt.figure(figsize=(5,6))
a.groupby(['Category'])['Sales'].sum().plot.pie();
```




### 1.7.3 Even Better pie plot

We can customize the pie plots by adding percentage , colors, etc..

[20]:
```
plt.figure(figsize=(8, 5))
plt.pie(a.groupby('Category')['Sales'].sum(), labels=a.
 ↪groupby('Category')['Sales'].sum().index, autopct='%1.1f%%', startangle=90,␣
 ↪colors=['lightcoral', 'lightskyblue', 'lightgreen','lightblue'])
plt.title('Sales Distribution by Category')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```

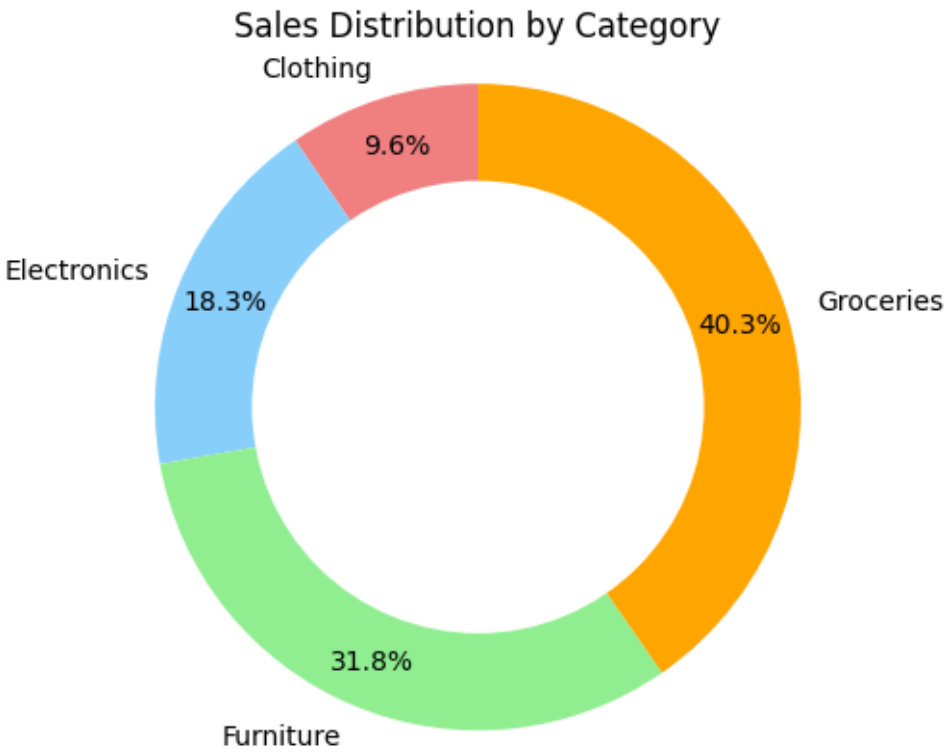



### 1.7.4 Donut Chart

aka Concentric Pie chart

[21]:
```
catgroup=a.groupby('Category')['Sales'].sum()

plt.figure(figsize=(8, 5))
fig, ax = plt.subplots()

ax.pie(catgroup, labels=catgroup.index , autopct='%1.1f%%',pctdistance=0.85,␣
 ↪startangle=90, colors=['lightcoral', 'lightskyblue', 'lightgreen','orange'])
plt.title('Sales Distribution by Category')

centre_circle = plt.Circle((0,0),0.70,fc='white')
fig.gca().add_artist(centre_circle)

plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show();
```

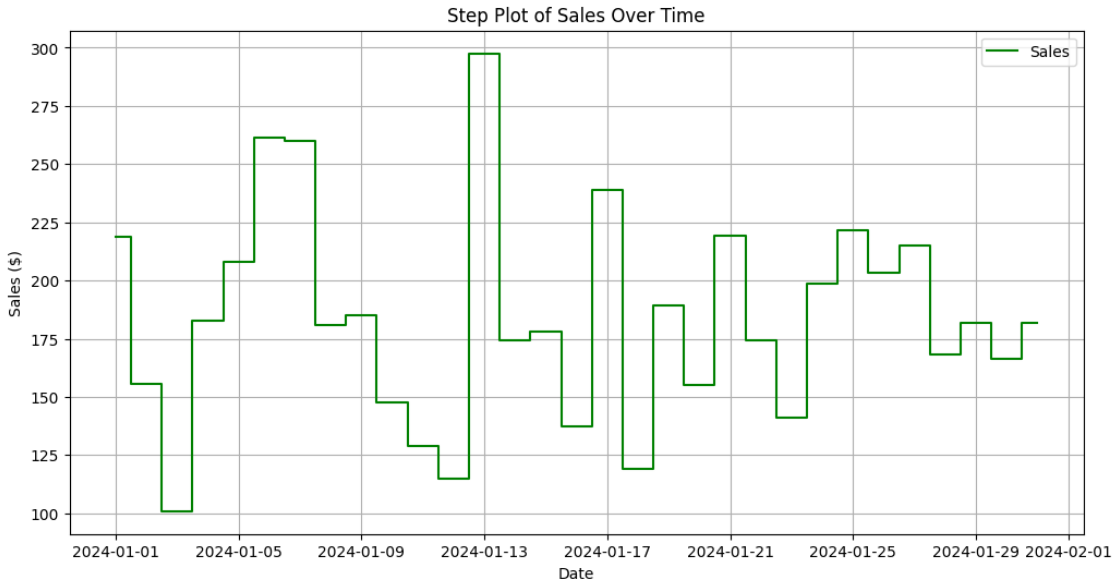
```
<Figure size 800x500 with 0 Axes>
```




### 1.8 Bonus Plots

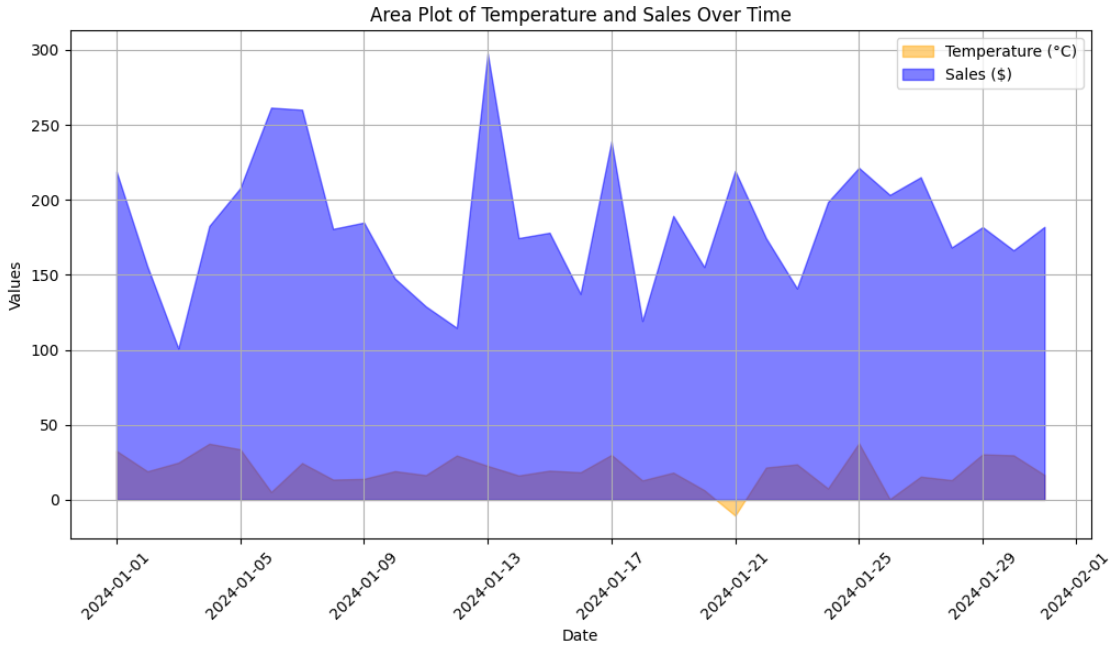
### 1.9 Step Plot

[22]:
```
plt.figure(figsize=(12,6))
plt.step(a['Date'], a['Sales'], where='mid', label='Sales', color='green');
plt.title('Step Plot of Sales Over Time')
plt.xlabel('Date')
plt.ylabel('Sales ($)')
plt.legend()
plt.grid()
plt.show()
```




### 1.9.1 Area Plots

[23]:
```
plt.figure(figsize=(12, 6))
plt.fill_between(a['Date'], a['Temperature'], color='orange', alpha=0.5,␣
↪label='Temperature (°C)')
plt.fill_between(a['Date'], a['Sales'], color='blue', alpha=0.5, label='Sales␣
↪($)')
plt.title('Area Plot of Temperature and Sales Over Time')
plt.xlabel('Date')
plt.ylabel('Values')
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.show()
```




I hope you found this information helpful! Feel free to save this post for future reference. Let's continue to learn and grow together!

Rajendra Prasad