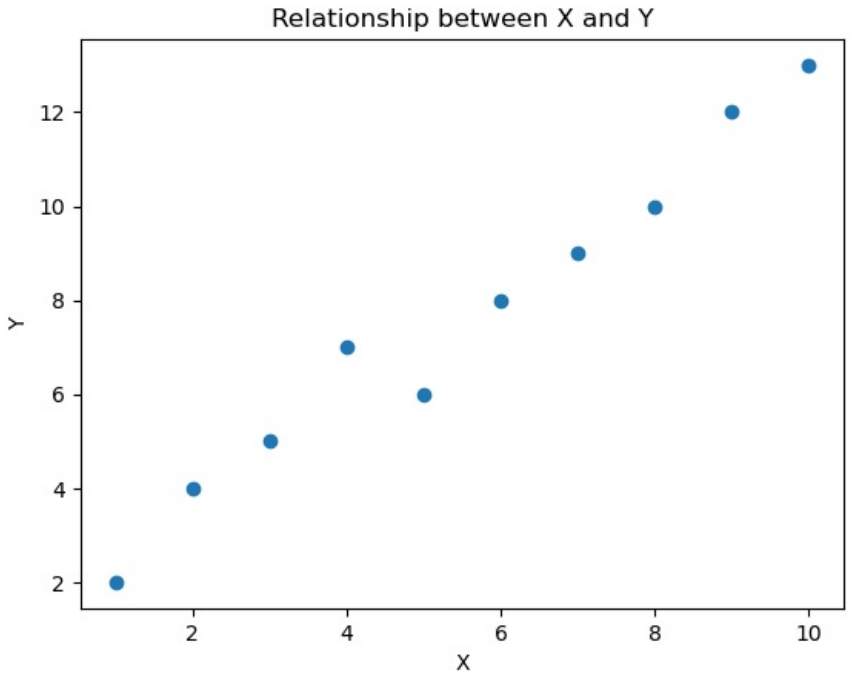
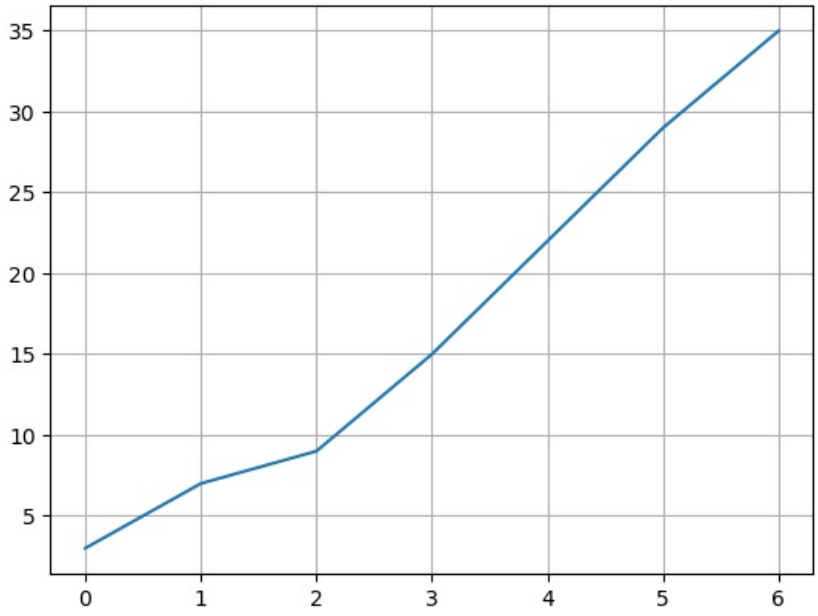
Matplotlib Assignment

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
#1. Create a scatter plot using Matplotlib to visualize the relationship between two arrays, x and y for the gi
#data.

x = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
y = np.array([2, 4, 5, 7, 6, 8, 9, 10, 12, 13])
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Relationship between X and Y")
plt.scatter(x,y)
plt.show()
```

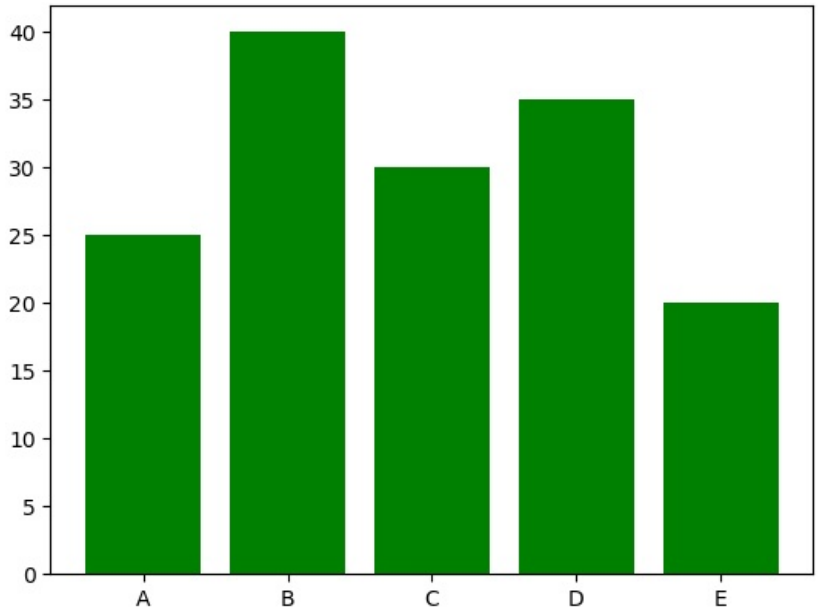


In [3]:
```
#2. Generate a line plot to visualize the trend of values for the given data.

data = np.array([3, 7, 9, 15, 22, 29, 35])
plt.plot(data)
plt.grid()
plt.show()
```

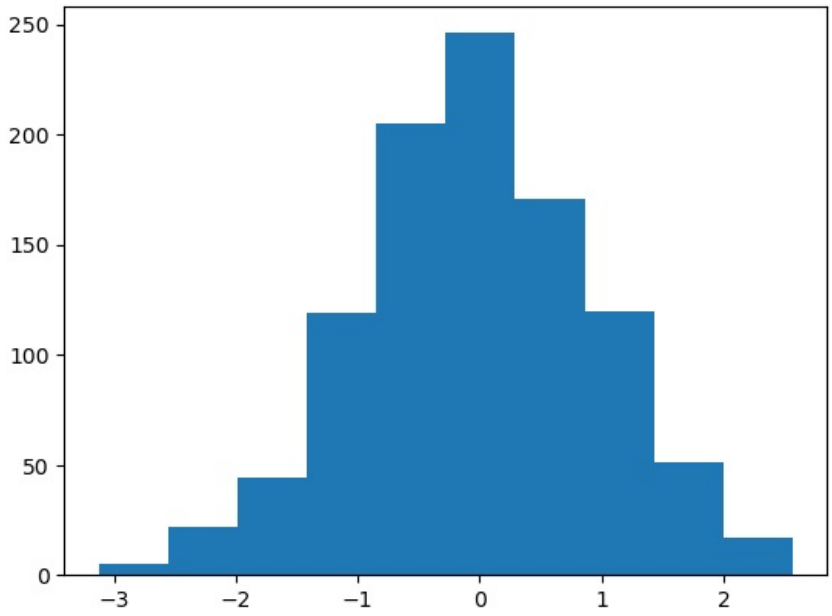


In [4]:
```
#3. Display a bar chart to represent the frequency of each item in the given array categories

categories = ['A', 'B', 'C', 'D', 'E']
values = [25, 40, 30, 35, 20]
```

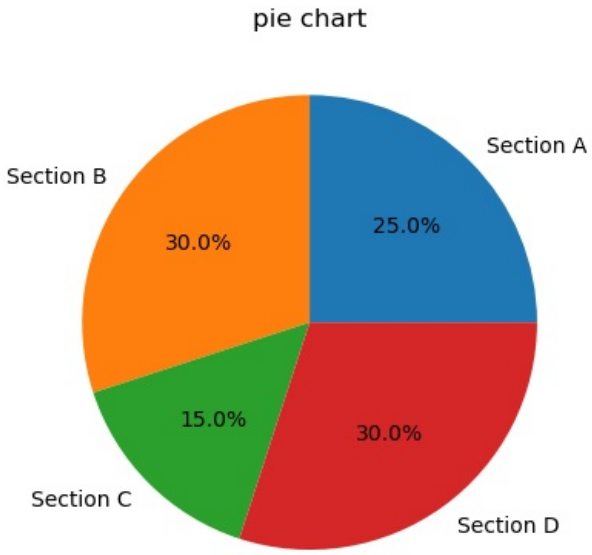
```
plt.bar(categories,values,color='green')
plt.show()
```



In [5]:
```
#4. Create a histogram to visualize the distribution of values in the array data.

data = np.random.normal(0, 1, 1000)
plt.hist(data)
plt.show()
```



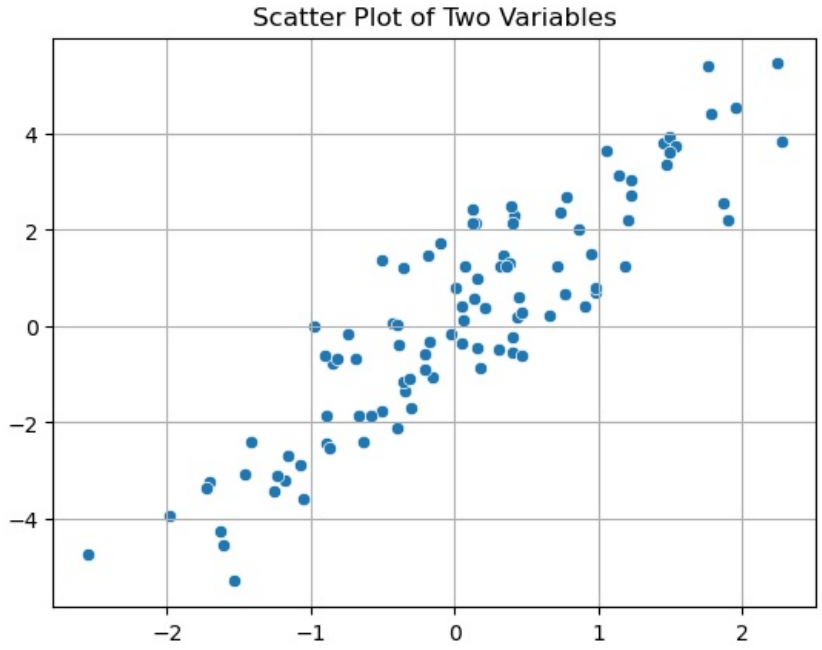
In [6]:
```
#5. Show a pie chart to represent the percentage distribution of different sections in the array `sections`.

sections = ['Section A', 'Section B', 'Section C', 'Section D']
sizes = [25, 30, 15, 30]

plt.pie(sizes,labels=sections,autopct='%1.1f%%')
plt.title("pie chart")
plt.show()
```



Seaborn Assignment

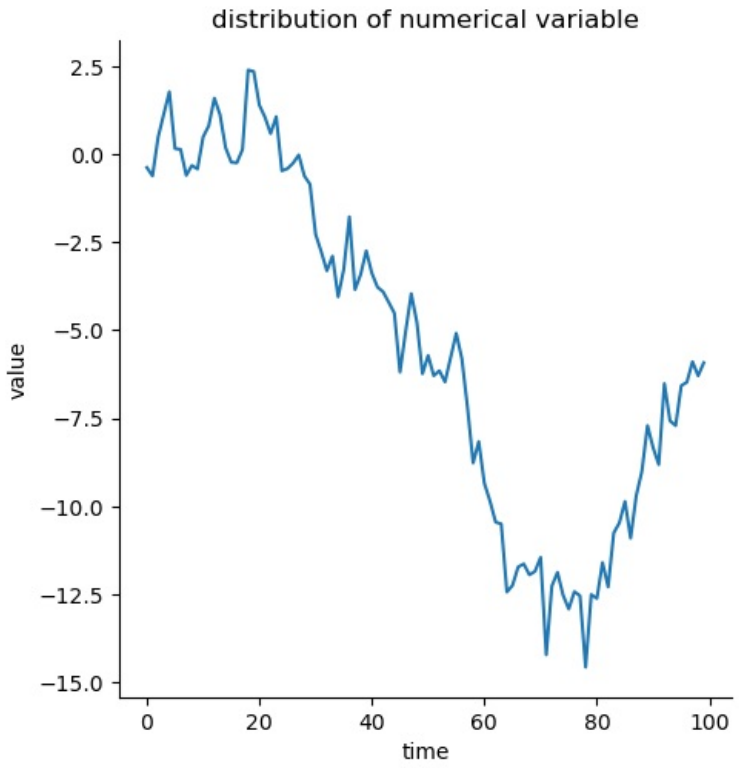
In [7]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [8]:
```
#1. Create a scatter plot to visualize the relationship between two variables, by generating a synthetic
#dataset.

np.random.seed(0)
variable1 = np.random.randn(100)
variable2=2*variable1+np.random.randn(100)
sns.scatterplot(x=variable1, y=variable2)
plt.title('Scatter Plot of Two Variables')
plt.grid(True)
plt.show()
```

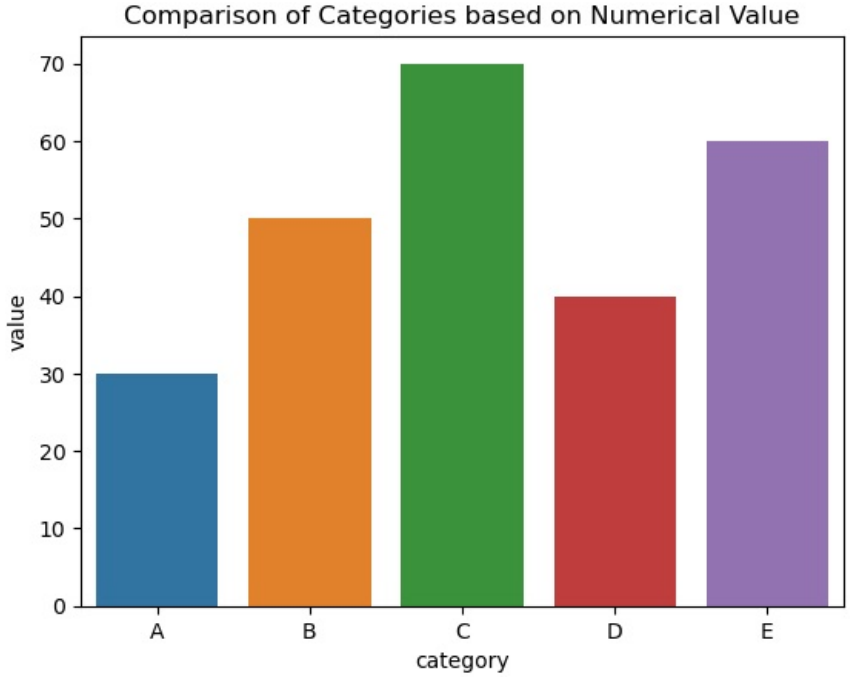


In [9]:
```
#2. Generate a dataset of random numbers. Visualize the distribution of a numerical variable.

time=np.arange(100)
value=np.random.randn(100).cumsum()
df=pd.DataFrame({"time":time,"value":value})
df
sns.relplot(x='time',y='value',kind='line',data=df)
plt.title('distribution of numerical variable')
plt.show()
```

```
C:\Users\om\Downloads\New folder\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has
changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

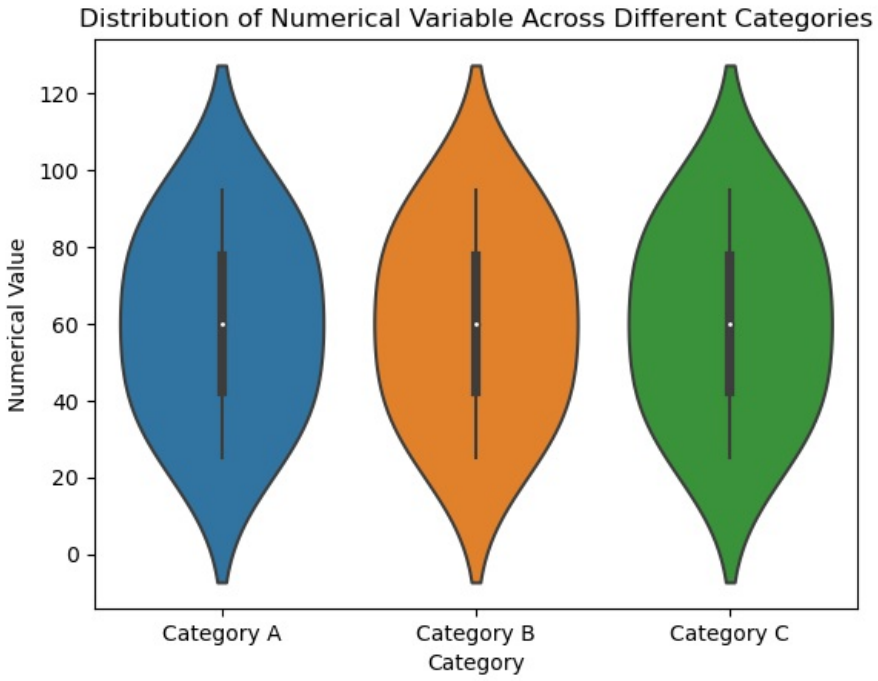


In [10]:
```
#3. Create a dataset representing categories and their corresponding values. Compare different categories
#based on numerical value

data = {
    'category': ['A', 'B', 'C', 'D', 'E'],
    'value': [30, 50, 70, 40, 60]
}

df = pd.DataFrame(data)
df
sns.barplot(x='category',y='value',data=df)
plt.title('Comparison of Categories based on Numerical Value')
plt.xlabel='category'
plt.ylabel='value'
plt.show()
```



In [11]:
```
#4. Generate a dataset with categories and numerical values. Visualize the distribution of a numerical
#variable across different categories.

categories = ['Category A', 'Category B', 'Category C']
numerical_values = [25, 35, 45, 55, 65, 75, 85, 95]
data = {'Category': [], 'Numerical Value': []}
for category in categories:
    for value in numerical_values:
        data['Category'].append(category)
        data['Numerical Value'].append(value)
```

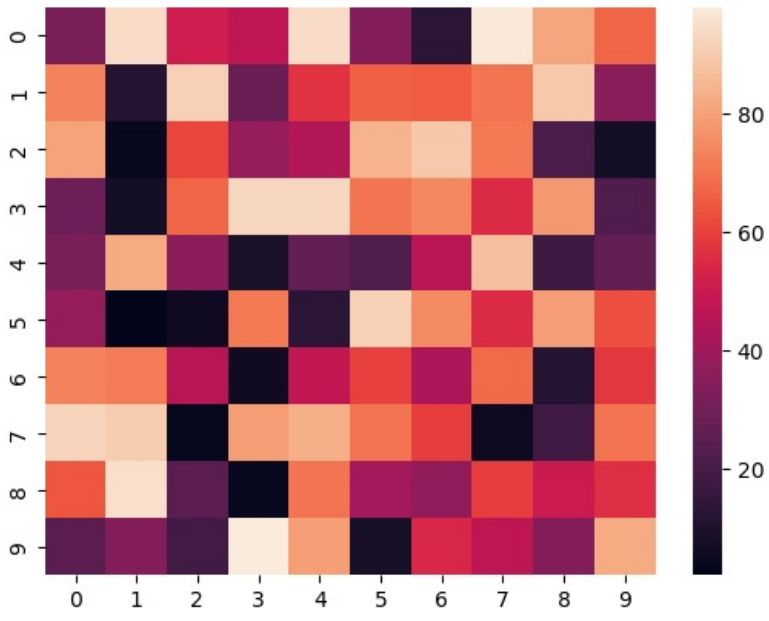
```
        df = pd.DataFrame(data)
df
sns.violinplot(x='Category', y='Numerical Value', data=df)
plt.title('Distribution of Numerical Variable Across Different Categories')

plt.show()
```



In [12]:

```
#5.Generate a synthetic dataset with correlated features. Visualize the correlation matrix of a dataset using a
#heatmap.

data = np.random.randint(low = 1,
                         high = 100,
                         size = (10, 10))

print("The data to be plotted:\n")
print(data)
# plotting the heatmap
hm = sns.heatmap(data = data)

# displaying the plotted heatmap
plt.show()
```

```
The data to be plotted:

[[32 94 51 47 94 34 13 97 81 67]
 [73 11 91 28 57 66 65 70 89 35]
 [80  4 61 38 44 84 89 71 21  7]
 [29  7 67 93 93 70 74 55 78 22]
 [32 82 36  9 26 22 46 87 17 26]
 [38  2  6 71 13 91 75 55 79 63]
 [73 72 46  6 48 60 43 68 11 58]
 [92 90  4 79 83 70 59  5 18 70]
 [64 95 25  4 70 41 37 59 50 56]
 [25 34 19 98 79  8 54 47 34 82]]
```



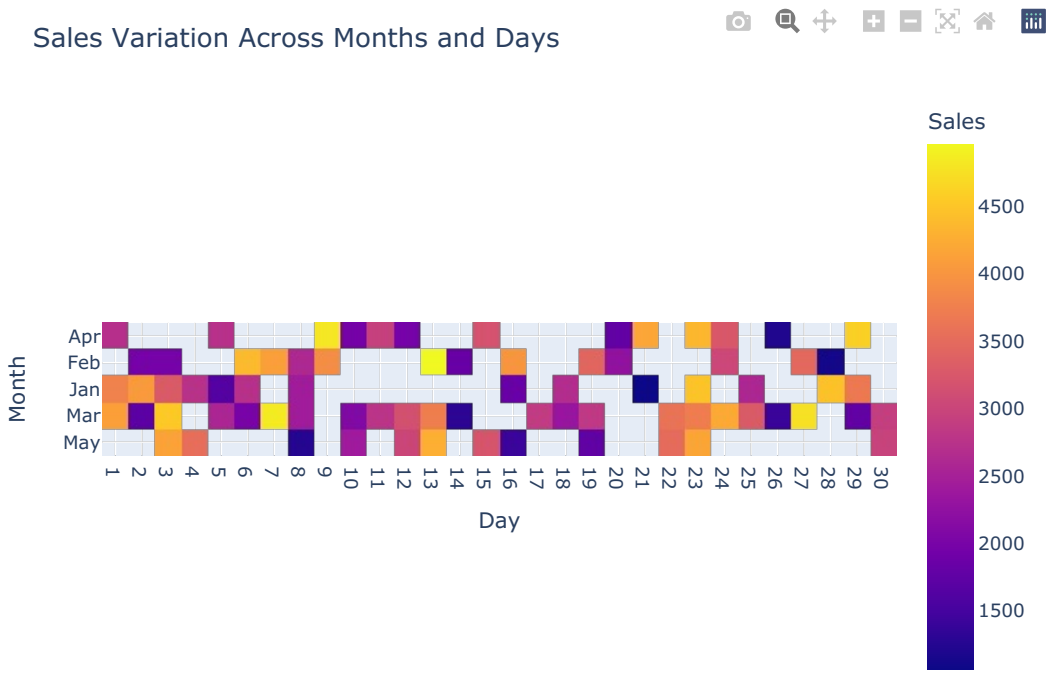
PLOTLY ASSIGNMENT

In [13]:
```
#1. Using the given dataset, to generate a 3D scatter plot to visualize the distribution of data points in a th
import plotly.graph_objects as go
import numpy as np
import pandas as pd
np.random.seed(30)
data = {
    'X': np.random.uniform(-10, 10, 300),
    'Y': np.random.uniform(-10, 10, 300),
    'Z': np.random.uniform(-10, 10, 300)
}
df = pd.DataFrame(data)
import plotly.graph_objects as go
import numpy as np

np.random.seed(30)

data = {
    'X': np.random.uniform(-10, 10, 300),
    'Y': np.random.uniform(-10, 10, 300),
    'Z': np.random.uniform(-10, 10, 300)
}

fig = go.Figure(data=[go.Scatter3d(
    x=data['X'],
    y=data['Y'],
    z=data['Z'],
    mode='markers',
    marker=dict(
        size=5,
        color=data['Z'],  # Color by Z value
        colorscale='Viridis',  # Choose a colorscale
        opacity=0.8
    )
)])

fig.update_layout(
    scene=dict(
        xaxis=dict(title='X'),
        yaxis=dict(title='Y'),
        zaxis=dict(title='Z'),
    ),
    margin=dict(l=0, r=0, b=0, t=0)
)

fig.show()
```

more info

In [14]:
```
#2. Using the Student Grades, create a violin plot to display the distribution of scores across different grade
#categories.
```

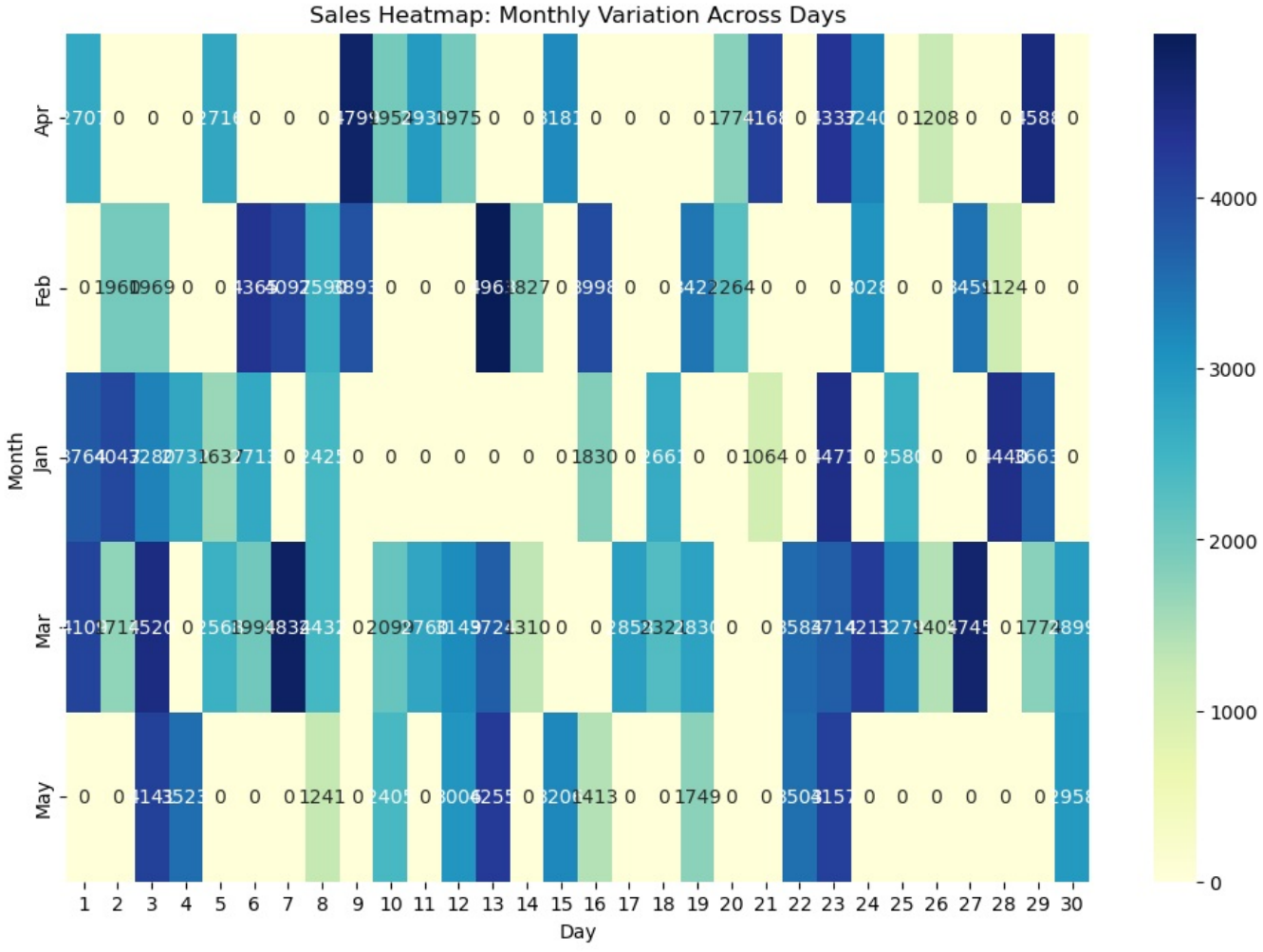
```
np.random.seed(15)
data = {
    'Grade': np.random.choice(['A', 'B', 'C', 'D', 'F'], 200),
    'Score': np.random.randint(50, 100, 200)
}
df = pd.DataFrame(data)
#Using the sales data, generate a heatmap to visualize the variation in sales across
#different months and days.
np.random.seed(20)
data = {
    'Month': np.random.choice(['Jan', 'Feb', 'Mar', 'Apr', 'May'], 100),
    'Day': np.random.choice(range(1, 31), 100),
    'Sales': np.random.randint(1000, 5000, 100)
}
df = pd.DataFrame(data)

import numpy as np
import pandas as pd
import plotly.express as px

np.random.seed(20)
data = {
    'Month': np.random.choice(['Jan', 'Feb', 'Mar', 'Apr', 'May'], 100),
    'Day': np.random.choice(range(1, 31), 100),
    'Sales': np.random.randint(1000, 5000, 100)
}

df = pd.DataFrame(data)
# Create a pivot table
heatmap_data = df.pivot_table(index='Month', columns='Day', values='Sales', aggfunc=np.mean)

# Convert row and column names to string for Plotly
heatmap_data.index = heatmap_data.index.astype(str)
heatmap_data.columns = heatmap_data.columns.astype(str)

# Create the heatmap using Plotly
fig = px.imshow(heatmap_data,
                labels=dict(x="Day", y="Month", color="Sales"),
                x=heatmap_data.columns,
                y=heatmap_data.index,)

# Customize layout
fig.update_layout(title='Sales Variation Across Months and Days',
                  xaxis_title='Day',
                  yaxis_title='Month')

# Show the plot
fig.show()
```



In [15]:

```
#Q.3  Using the sales data, generate a heatmap to visualize the variation in sales across different months and
#days.

np.random.seed(20)
data = {
    'Month': np.random.choice(['Jan', 'Feb', 'Mar', 'Apr', 'May'], 100),
```

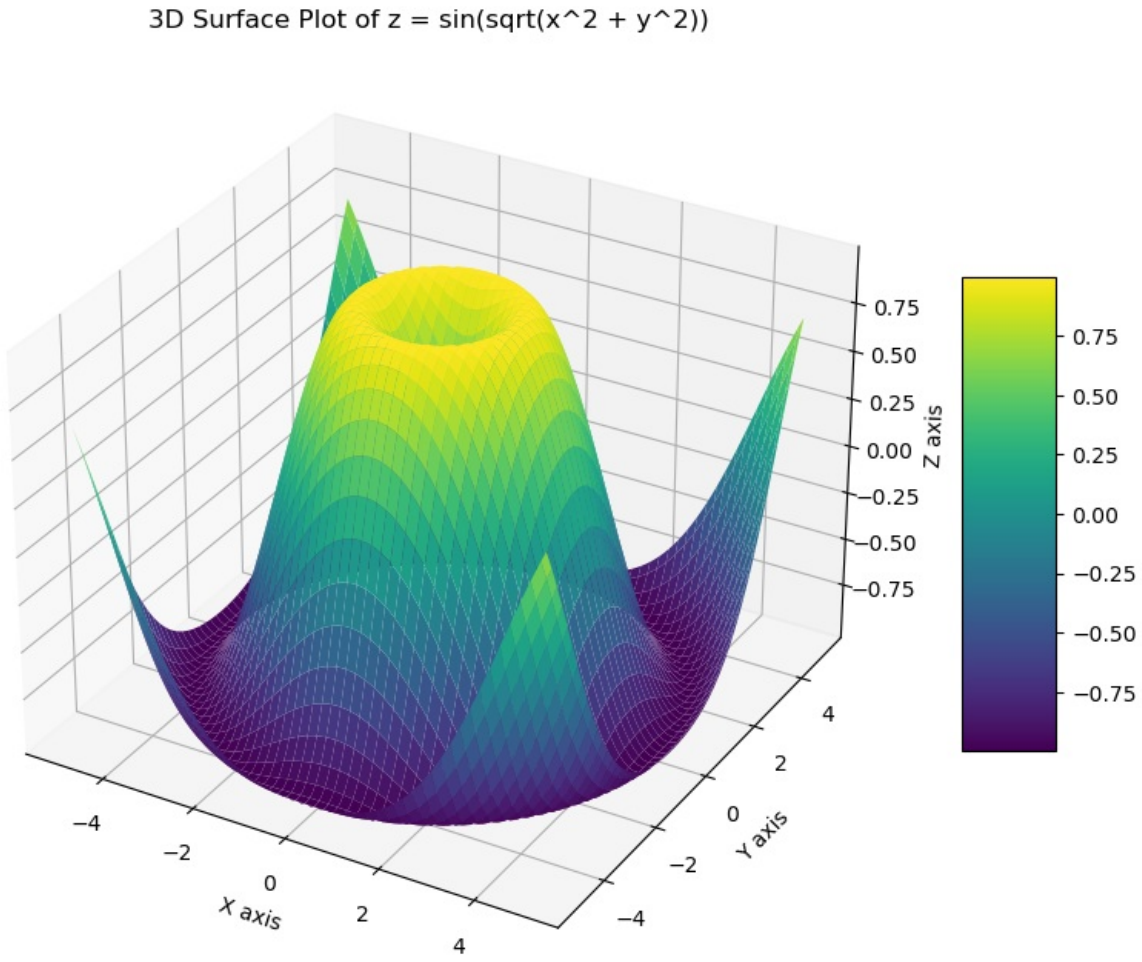
```
    'Day': np.random.choice(range(1, 31), 100),
    'Sales': np.random.randint(1000, 5000, 100)
}
df = pd.DataFrame(data)

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

np.random.seed(20)
data = {
    'Month': np.random.choice(['Jan', 'Feb', 'Mar', 'Apr', 'May'], 100),
    'Day': np.random.choice(range(1, 31), 100),
    'Sales': np.random.randint(1000, 5000, 100)
}
df = pd.DataFrame(data)

# Pivot the data
pivot_table = df.pivot_table(values='Sales', index='Month', columns='Day', aggfunc='mean')

# Fill any missing values with 0
pivot_table = pivot_table.fillna(0)

# Generate heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(pivot_table, annot=True, fmt=".0f", cmap='YlGnBu')
plt.title('Sales Heatmap: Monthly Variation Across Days')
plt.show()
```



In [17]:

```
#Q.4  Using the given x and y data, generate a 3D surface plot to visualize the function
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
x, y = np.meshgrid(x, y)
z = np.sin(np.sqrt(x**2 + y**2))
data = {
    'X': x.flatten(),
    'Y': y.flatten(),
    'Z': z.flatten()
}
df = pd.DataFrame(data)


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
from mpl_toolkits.mplot3d import Axes3D

# Generate data
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
x, y = np.meshgrid(x, y)
z = np.sin(np.sqrt(x**2 + y**2))

data = {
    'X': x.flatten(),
    'Y': y.flatten(),
    'Z': z.flatten()
}
df = pd.DataFrame(data)

# Create 3D surface plot
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Reshape the data
x = df['X'].values.reshape(100, 100)
y = df['Y'].values.reshape(100, 100)
z = df['Z'].values.reshape(100, 100)

# Plot the surface
surf = ax.plot_surface(x, y, z, cmap='viridis')

# Add color bar, labels, and title
fig.colorbar(surf, shrink=0.5, aspect=5)
ax.set_xlabel('X axis')
ax.set_ylabel('Y axis')
ax.set_zlabel('Z axis')
ax.set_title('3D Surface Plot of z = sin(sqrt(x^2 + y^2))')

plt.show()
```

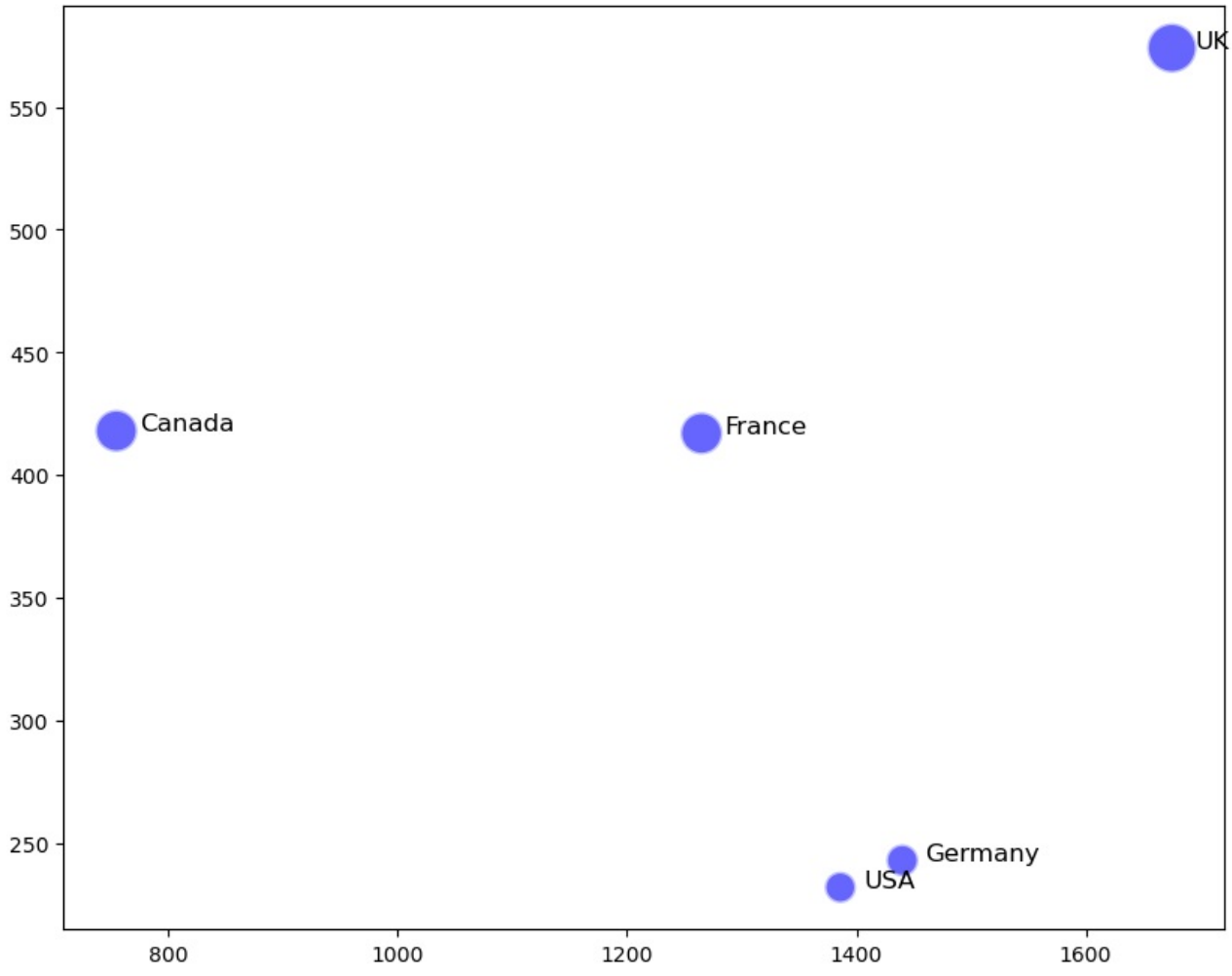


In [18]:

```
#Q.5. Using the given dataset, create a bubble chart to represent each country's population (y-axis), GDP (xaxi

np.random.seed(25)
data = {
    'Country': ['USA', 'Canada', 'UK',
'Germany', 'France'],
    'Population':
np.random.randint(100, 1000, 5),
    'GDP': np.random.randint(500, 2000,
5)
}
```

```
df = pd.DataFrame(data)


plt.figure(figsize=(10, 8))

# Scatter plot for bubble chart
plt.scatter(
    df['GDP'],        # x-axis: GDP
    df['Population'], # y-axis: Population
    s=df['Population'],  # Bubble size proportional to population
    alpha=0.6,        # Transparency of bubbles
    color='b',        # Bubble color
    edgecolors="w",   # Edge color for bubbles
    linewidth=2       # Line width of edges
)

# Add labels for each country
for i in range(df.shape[0]):
    plt.text(df['GDP'][i] + 20, df['Population'][i], df['Country'][i], fontsize=12)

# Set axis labels
plt.xlabel('GDP (in billion USD)')
plt.ylabel('Population (in millions)')
plt.title('Bubble Chart: Population vs GDP')

# Display the plot
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[18], line 33
     30     plt.text(df['GDP'][i] + 20, df['Population'][i], df['Country'][i], fontsize=12)
     32 # Set axis labels
---> 33 plt.xlabel('GDP (in billion USD)')
     34 plt.ylabel('Population (in millions)')
     35 plt.title('Bubble Chart: Population vs GDP')

TypeError: 'str' object is not callable
```



BOKEH ASSIGNMENT:

In [19]:

```
#Q.1.Create a Bokeh plot displaying a sine wave. Set x-values from 0 to 10 and y-values as the sine of x.

import numpy as np
from bokeh.plotting import figure, show
```

```
from bokeh.io import output_notebook

# Set up the output to display in the notebook
output_notebook()

# Generate the x and y data
x = np.linspace(0, 10, 100)
y = np.sin(x)

# Create a new plot with a title and axis labels
p = figure(title="Sine Wave", x_axis_label='x', y_axis_label='y')

# Add a line renderer with legend and line thickness
p.line(x, y, legend_label="sin(x)", line_width=2)

# Show the results
show(p)
```

Loading BokehJS ...

In [20]:

```
#Q.2import numpy as np
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from bokeh.models import ColumnDataSource

# Set up the output to display in the notebook
output_notebook()

# Generate random data
np.random.seed(42)
n = 100
x = np.random.random(size=n) * 100
y = np.random.random(size=n) * 100
sizes = np.random.random(size=n) * 50  # Random sizes for markers
colors = np.random.randint(1, 256, size=(n, 3))  # Random RGB colors

# Convert the colors to hex format
colors = ['#%02x%02x%02x' % (r, g, b) for r, g, b in colors]

# Create a ColumnDataSource
source = ColumnDataSource(data=dict(x=x, y=y, sizes=sizes, colors=colors))

# Create a new plot
p = figure(title="Random Scatter Plot", x_axis_label='X', y_axis_label='Y')

# Add circle glyphs to the plot
p.circle('x', 'y', size='sizes', color='colors', fill_alpha=0.6, source=source)

# Show the results
show(p)
```

Loading BokehJS ...

In [24]:

```
#Q.3. Generate a Bokeh bar chart representing the counts of different fruits using the following dataset.

from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from bokeh.models import ColumnDataSource

# Set up the output to display in the notebook
output_notebook()

# Dataset
fruits = ['Apples', 'Oranges', 'Bananas', 'Pears']
counts = [20, 25, 30, 35]

# Convert the data to a ColumnDataSource
source = ColumnDataSource(data=dict(fruits=fruits, counts=counts))

# Create a new plot with a title and axis labels
p = figure(x_range=fruits, title="Fruit Counts",
           x_axis_label='Fruit', y_axis_label='Count',
           toolbar_location=None, tools="")

# Add vertical bars
p.vbar(x='fruits', top='counts', width=0.7, source=source,
       fill_color="skyblue", line_color="white")

# Customize the plot
p.y_range.start = 0
p.xgrid.grid_line_color = None
```

```
p.axis.minor_tick_line_color = None
p.outline_line_color = None
p.xaxis.major_label_orientation = "vertical"

# Show the results
show(p)
```

Loading BokehJS ...

In [26]:

```
#Q.4. Create a Bokeh histogram to visualize the distribution of the given data.

data_hist = np.random.randn(1000)
hist, edges = np.histogram(data_hist, bins=30)

import numpy as np
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from bokeh.models import ColumnDataSource

# Set up the output to display in the notebook
output_notebook()

# Generate the data
data_hist = np.random.randn(1000)
hist, edges = np.histogram(data_hist, bins=30)

# Convert the data to a ColumnDataSource
source = ColumnDataSource(data=dict(top=hist, left=edges[:-1], right=edges[1:]))

# Create a new plot with a title and axis labels
p = figure(title="Histogram of Random Data",
           x_axis_label='Value', y_axis_label='Frequency',
           tools="", background_fill_color="#f5f5f5")

# Add quad glyphs to the plot
p.quad(top='top', bottom=0, left='left', right='right', source=source,
       fill_color="navy", line_color="white", alpha=0.7)

# Customize the plot
p.y_range.start = 0
p.xgrid.grid_line_color = None
p.ygrid.grid_line_color = "white"
p.outline_line_color = None

# Show the results
show(p)
```

Loading BokehJS ...

In [27]:

```
#Q.5 Create a Bokeh heatmap using the provided dataset.

data_heatmap = np.random.rand(10, 10)
x = np.linspace(0, 1, 10)
y = np.linspace(0, 1, 10)
xx, yy = np.meshgrid(x, y)

import numpy as np
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from bokeh.models import ColorBar
from bokeh.transform import linear_cmap
from bokeh.palettes import Viridis256
from bokeh.layouts import column

# Set up the output to display in the notebook
output_notebook()

# Generate the data
data_heatmap = np.random.rand(10, 10)
x = np.linspace(0, 1, 10)
y = np.linspace(0, 1, 10)
xx, yy = np.meshgrid(x, y)

# Flatten the data for Bokeh
x_flat = xx.flatten()
y_flat = yy.flatten()
z_flat = data_heatmap.flatten()

# Create a ColumnDataSource
source = ColumnDataSource(data=dict(x=x_flat, y=y_flat, z=z_flat))
```

```
# Define the color mapper
mapper = linear_cmap(field_name='z', palette=Viridis256, low=min(z_flat), high=max(z_flat))

# Create a new plot with a title and axis labels
p = figure(title="Heatmap", x_axis_label='X', y_axis_label='Y',
           x_range=(0, 1), y_range=(0, 1),
           tools="", background_fill_color="#fafafa")

# Add squares to the plot
p.rect(x='x', y='y', width=0.1, height=0.1, source=source,
       line_color=None, fill_color=mapper)

# Add color bar
color_bar = ColorBar(color_mapper=mapper['transform'], width=8, location=(0,0))
p.add_layout(color_bar, 'right')

# Customize the plot
p.xaxis.major_label_orientation = "horizontal"
p.yaxis.major_label_orientation = "horizontal"
p.xgrid.grid_line_color = None
p.ygrid.grid_line_color = None
p.outline_line_color = None

# Show the results
show(p)
```

Loading BokehJS ...

In [ ]: