2. Pandas

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

data = {
    "ID" : [101, 102, 103, 104, 105],
    "Name" : ["Quynh Nhu", "Hoang Hai", None, "Phuong Tuan", "Thien An"],
    "Age" : [26,18,20,None,19],
    "Salary" : [5000,35000,65000,100000,None]
}
df= pd.DataFrame(data)
print(df)
```

```
    ID         Name   Age    Salary
0  101    Quynh Nhu  26.0    5000.0
1  102    Hoang Hai  18.0   35000.0
2  103         None  20.0   65000.0
3  104  Phuong Tuan   NaN  100000.0
4  105     Thien An  19.0       NaN
```

In [ ]:
```
# Điền giá trị Thiếu trong cột Age = giá trị trung bình
# Loại bỏ các dòng chứa giá trị thiếu
# Điền giá trị thiếu trong name = "Unknown"ArithmeticError
# Điền giá trị thiếu trong Salary = phương pháp Interpolation
```

In [ ]:
```
# 1. Điền giá trị thiếu trong cột 'Age' bằng giá trị trung bình
df["Age"].fillna(int(df["Age"].mean()), inplace=True)
print(df)
# 2. Loại bỏ các dòng chứa giá trị thiếu
df.dropna(inplace=True)  # Không cần vì ID không có thiếu, nhưng để làm mẫu
print(df)
# 3. Điền giá trị thiếu trong cột 'Name' bằng "Unknown"
df["Name"].fillna("Unknown", inplace=True)
print(df)

# 4. Điền giá trị thiếu trong cột 'Salary' bằng phương pháp nội suy (interpolation)
df["Salary"].interpolate(method='linear', inplace=True)
print(df)
```

```
    ID         Name   Age    Salary
0  101    Quynh Nhu  26.0    5000.0
1  102    Hoang Hai  18.0   35000.0
2  103         None  20.0   65000.0
3  104  Phuong Tuan  20.0  100000.0
4  105     Thien An  19.0       NaN
    ID         Name   Age    Salary
0  101    Quynh Nhu  26.0    5000.0
1  102    Hoang Hai  18.0   35000.0
3  104  Phuong Tuan  20.0  100000.0
    ID         Name   Age    Salary
0  101    Quynh Nhu  26.0    5000.0
1  102    Hoang Hai  18.0   35000.0
3  104  Phuong Tuan  20.0  100000.0
    ID         Name   Age    Salary
0  101    Quynh Nhu  26.0    5000.0
1  102    Hoang Hai  18.0   35000.0
3  104  Phuong Tuan  20.0  100000.0
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_988\98559109.py:2: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained assignment using an inplace metho
d.
The behavior will change in pandas 3.0. This inplace method will never work because the interme
diate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: valu
e}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace
on the original object.


  df["Age"].fillna(int(df["Age"].mean()), inplace=True)
C:\Users\DELL\AppData\Local\Temp\ipykernel_988\98559109.py:8: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained assignment using an inplace metho
d.
The behavior will change in pandas 3.0. This inplace method will never work because the interme
diate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: valu
e}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace
on the original object.


  df["Name"].fillna("Unknown", inplace=True)
C:\Users\DELL\AppData\Local\Temp\ipykernel_988\98559109.py:12: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained assignment using an inplace metho
d.
The behavior will change in pandas 3.0. This inplace method will never work because the interme
diate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: valu
e}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace
on the original object.


  df["Salary"].interpolate(method='linear', inplace=True)
```

In [ ]:
```python
# cho data

df1= pd.DataFrame({
    "ID" : [1,2,3],
    "Score_A" : [70,90,85]
})

df2= pd.DataFrame({
    "ID" : [3,4,5],
    "Score_B" : [62,91,75]
})
```

In [ ]:
```python
# Thực hiện Merge trên cột id (inner join, outer join, left join)

#Nối DataFrame theo chiều dọc

#Gộp df1 và df2 để điền giá trị thiếu
df3 = pd.merge(df1, df2, on='ID', how='inner')
print(df3)
df4 = pd.merge(df1, df2, on='ID', how='outer')
print(df4)
df5 = pd.merge(df1, df2, on='ID', how='left')
print(df5)

df6 = pd.concat([df1, df2], axis=0)
```

```
print(df6)

df7 = df1.set_index("ID").combine_first(df2.set_index("ID")).reset_index()
print(df7)
```

```
   ID  Score_A  Score_B
0   3       85       62
   ID  Score_A  Score_B
0   1     70.0      NaN
1   2     90.0      NaN
2   3     85.0     62.0
3   4      NaN     91.0
4   5      NaN     75.0
   ID  Score_A  Score_B
0   1       70      NaN
1   2       90      NaN
2   3       85     62.0
   ID  Score_A  Score_B
0   1     70.0      NaN
1   2     90.0      NaN
2   3     85.0      NaN
0   3      NaN     62.0
1   4      NaN     91.0
2   5      NaN     75.0
   ID  Score_A  Score_B
0   1     70.0      NaN
1   2     90.0      NaN
2   3     85.0     62.0
3   4      NaN     91.0
4   5      NaN     75.0
```

In [ ]:
```
# Cho dữ liệu sau
data = pd.DataFrame({
    "ID" : range(1, 100001),
    "Value" : np.random.randint(1, 100, 100000)
})
# data

#Dùng .astype để tối ưu hóa bộ nhớ
data["ID"] = data["ID"].astype("int32")
data["Value"] = data["Value"].astype("int8")

# Tìm 5 giá trị phổ biến trong cột value
data["Value"].value_counts().head(5)

# Sử dụng query để lọc dữ liệu nhanh hơn df[df["Value"] > 90]
data.query("Value > 90")
```

Out[ ]:

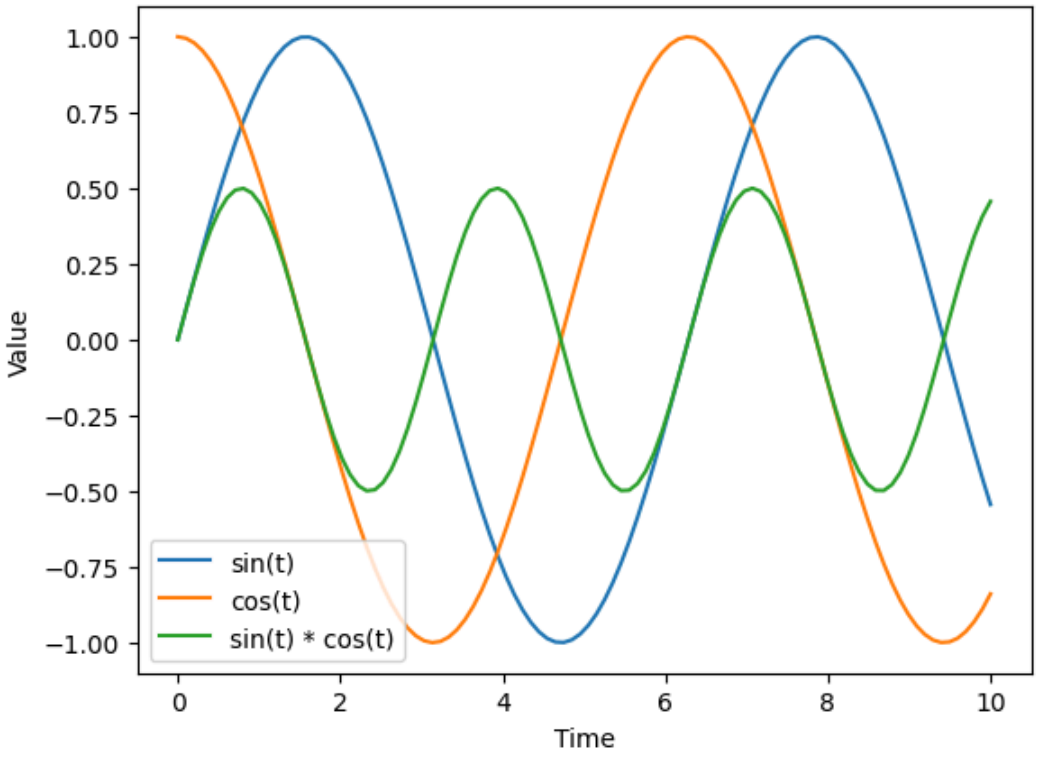
	ID	Value
10	11	91
33	34	99
41	42	96
67	68	97
68	69	99
...	...	...
99958	99959	93
99964	99965	97
99977	99978	91
99989	99990	98
99993	99994	92

8995 rows × 2 columns

3. Matplotlib

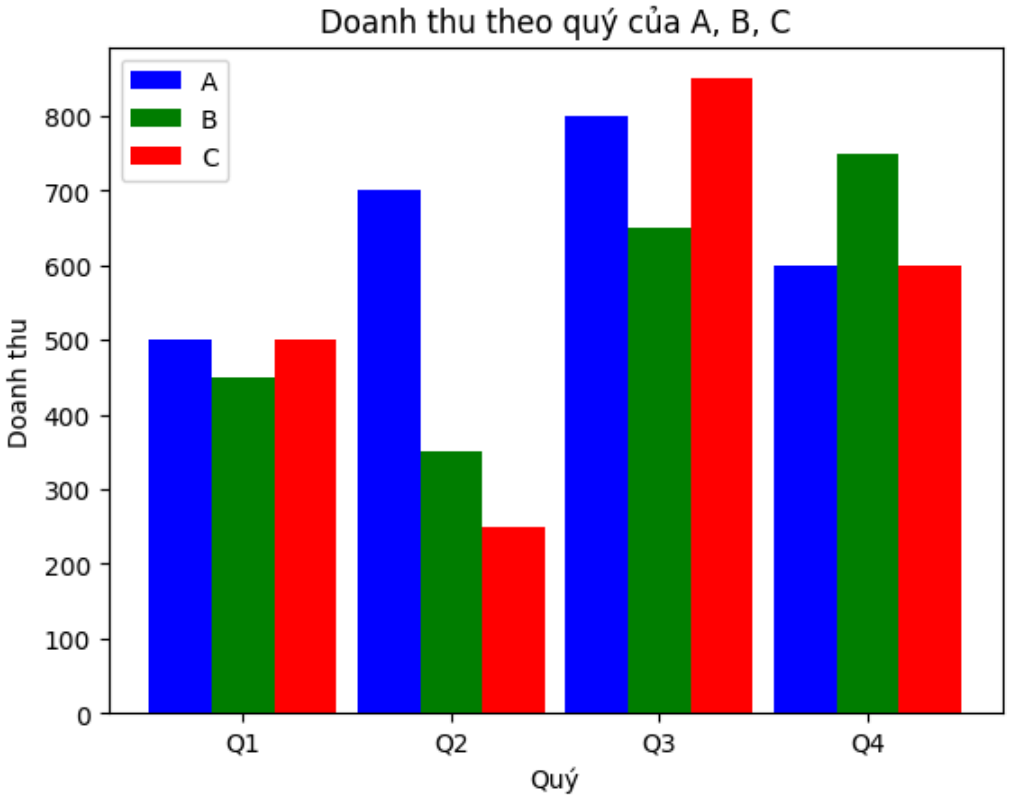
In [ ]:

```
# Biểu đồ nhiều đường

# cho dữ liệu sau

import matplotlib.pyplot as plt

import numpy as np

t = np.linspace(0, 10, 100)
y1 = np.sin(t)
y2 = np.cos(t)
y3 = np.sin(t) * np.cos(t)

# tạo biểu đồ các đường theo thời gian

plt.plot(t, y1, label='sin(t)')
plt.plot(t, y2, label='cos(t)')
plt.plot(t, y3, label='sin(t) * cos(t)')
plt.xlabel('Time')
plt.ylabel('Value')
plt.legend()
plt.show()
```

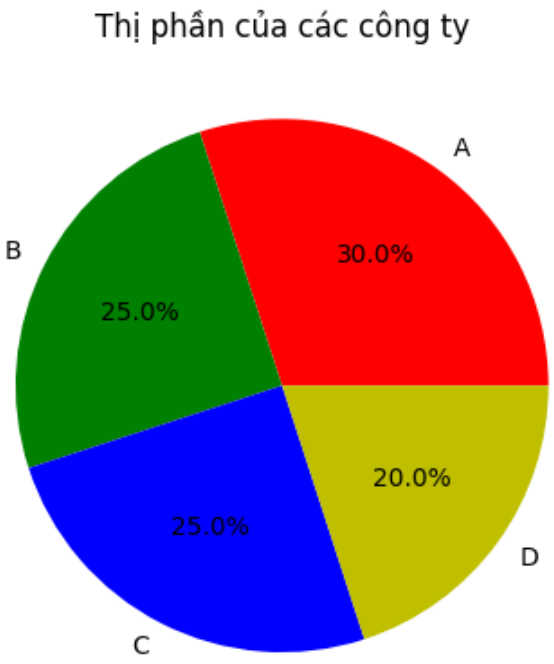


In [ ]:
```python
# Biểu đồ thanh nhóm

# cho dữ liệu sau
labels = ['Q1', 'Q2', 'Q3', 'Q4']
A = [500,700,800,600]
B = [450, 350, 650, 750]
C = [500, 250, 850, 600]

# Vẽ biểu đồ thanh nhóm thể hiện danh thu của A, B, C trong 4 quý
x = np.arange(len(labels))
width = 0.3
plt.bar(x - width, A, width, label='A', color='b')  # Dịch trái width đơn vị
plt.bar(x, B, width, label='B', color='g')          # Vẽ tại vị trí gốc
plt.bar(x + width, C, width, label='C', color='r')  # Dịch phải width đơn vị

# Thêm nhãn trục và tiêu đề
plt.xlabel("Quý")
plt.ylabel("Doanh thu")
plt.title("Doanh thu theo quý của A, B, C")

# Đặt nhãn trục x
plt.xticks(x, labels)

# Hiển thị chú thích
plt.legend()
```

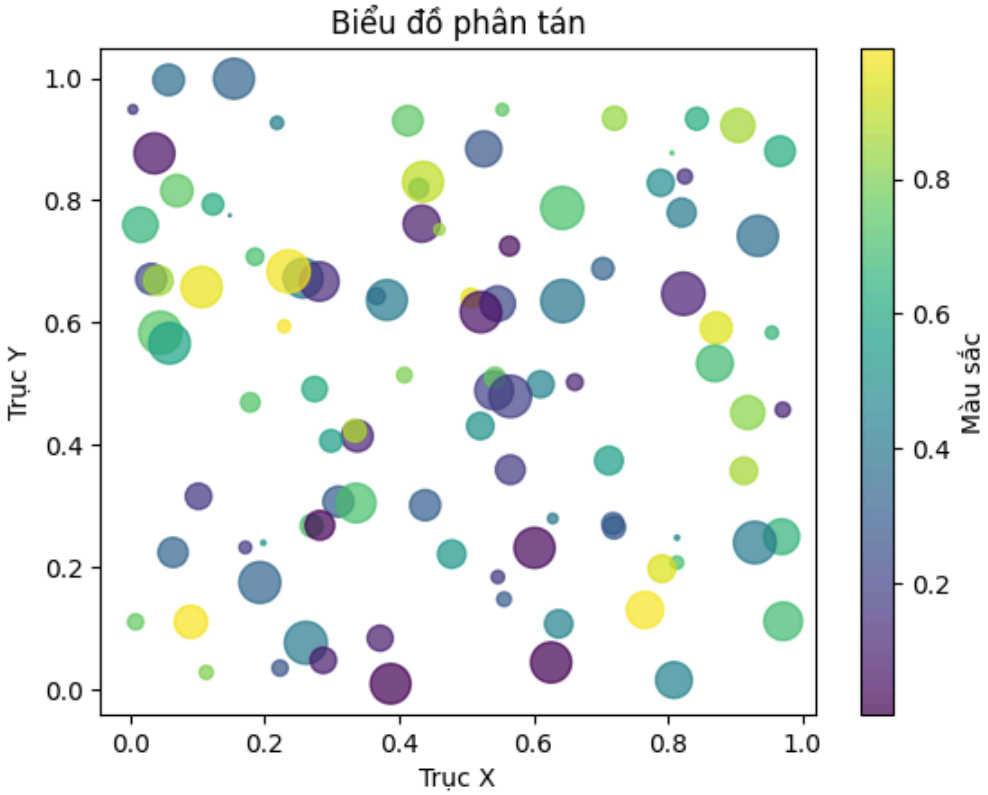
Out[ ]:
```
<matplotlib.legend.Legend at 0x157550c8410>
```



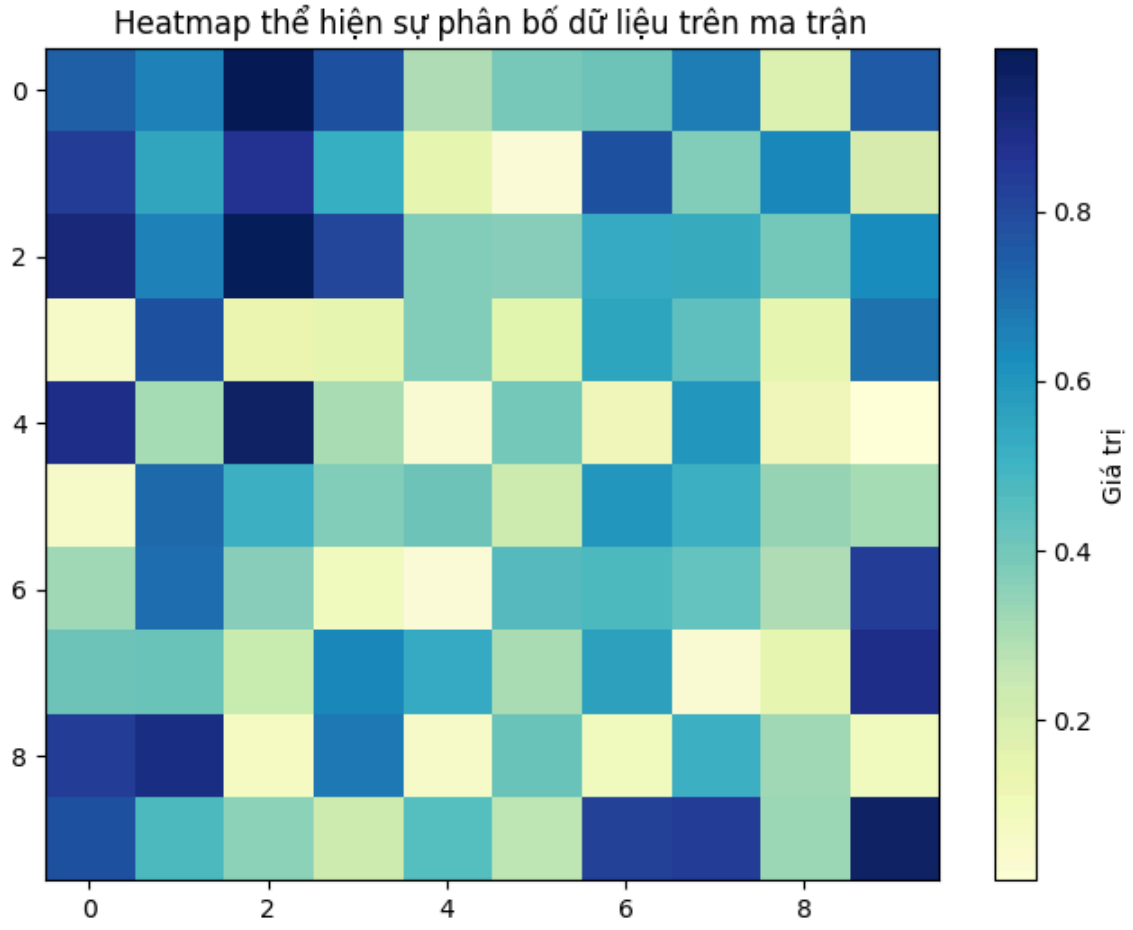
In [ ]:
```
# Biểu đồ tròn

Cty = ['A', 'B', 'C', 'D']
thiphan = [30, 25, 25, 20]
color = ['r', 'g', 'b', 'y']

# Tạo biểu đồ tròn biểu hiện tỉ lệ thị phần của các công ty

plt.pie(thiphan, labels=Cty, colors=color, autopct='%1.1f%%')
plt.title("Thị phần của các công ty")
plt.show()
```



In [ ]:
```python
# Biểu đồ phân tán

# cho dữ liệu sau

x = np.random.rand(100)
y = np.random.rand(100)
sizes = np.random.rand(100) * 300
colors = np.random.rand(100)

# tạo biểu đồ phân tán của 2 biến ngẫu nhiên
plt.scatter(x, y, s=sizes, c=colors, cmap='viridis', alpha=0.7)

# Thêm nhãn trục và tiêu đề
plt.xlabel("Trục X")
plt.ylabel("Trục Y")
plt.title("Biểu đồ phân tán")

# Hiển thị thang màu
plt.colorbar(label="Màu sắc")

# Hiển thị đồ thị
plt.show()
```



In [ ]:
```
# biểu đồ nhiệt

data = np.random.rand(10, 10)

# vẽ 1 heatmap thể hiện sự phân bố của dữ liệu trên ma trận
plt.figure(figsize=(8, 6))
plt.imshow(data, cmap="YlGnBu", aspect="auto")
plt.colorbar(label="Giá trị")
plt.title("Heatmap thể hiện sự phân bố dữ liệu trên ma trận")
plt.show()
```


Heatmap thể hiện sự phân bố dữ liệu trên ma trận
0
2
4
6
8
0
2
4
6
8
0.8
0.6
0.4
0.2
Giá trị