ct47823-ledangkimlan-baikiemtraso1

June 27, 2024

1. Tạo một DataFrame từ dữ liệu trên
2. Hiên thị thông tin về DataFrame vừa tạo
3. Lọc các hàng trong DataFrame có 'Age' lớn hơn 28
4. Tính giá trị trung bình của cột 'Salary'.
5. Nhóm dữ liệu theo cột 'Age' và tính tông 'Salary' cho môi nhóm -
6. Sắp xếp DataFrame theo cột 'Salary' giảm dân
7. Vẽ biểu đô cột cho cột 'Age'
8. Vẽ biểu đồ đường cho cột 'Salary'.
9. Vẽ biểu đồ tròn cho cột 'Age'.
10. Vẽ biểu đồ phân tán cho 'Age' và 'Salary'.
11. Kiểm tra xem có giá trị NaN nào trong DataFrame không.
12. Thay thế các giá trị của cột 'Age' lớn hơn 30 bằng giá trị trung bình của cột đó.
13. Chuẩn hóa (normalize) cột 'Age' về khoảng giá trị từ 0 đến 1.
14. Tạo một cột mới 'Age_group' phân loại tuổi thành 'Young', 'Middle-aged' và 'Old' dựa trên giá trị của cột 'Age'.
15. Tính toán tỷ lệ phần trăm thay đổi (percentage change) của cột 'Salary'.
16. Tìm các giá trị trùng lặp trong DataFrame dựa trên cột 'Name' và loại bỏ các hàng trùng lặp, giữ lại hàng đầu tiên.

1. Tạo một DataFrame từ dữ liệu trên

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[3]:
```
data = {
    "Name" : ["Alice", "Bob", "Charlie", "David", "Eva", "Frank", "Grace",␣
 ↪"Hannah", "Ivan", "Jack", "Kely", "Liam", "Mona", "Nina", "Oscar"],
    "Age" : ["25", "30", "35", "28", "22", "45", "34", "31", "27", "29", "33",␣
 ↪"40", "26", "32", "36"],
```

```
    "Salary" : ["50000", "60000", "70000", "55000", "52000", "80000", "72000",␣
  ↪"68000", "61000", "59000", "63000", "77000", "53000", "66000", "75000"],
}
```

[4]:
```
print(data)
```

```
{'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva', 'Frank', 'Grace', 'Hannah',
'Ivan', 'Jack', 'Kely', 'Liam', 'Mona', 'Nina', 'Oscar'], 'Age': ['25', '30',
'35', '28', '22', '45', '34', '31', '27', '29', '33', '40', '26', '32', '36'],
'Salary': ['50000', '60000', '70000', '55000', '52000', '80000', '72000',
'68000', '61000', '59000', '63000', '77000', '53000', '66000', '75000']}
```

2. Hiển thị thông tin về DataFrame vừa tạo

[5]:
```
df = pd.DataFrame(data)
```

[6]:
```
print("Thông tin DataFrame vừa tạo:")
print(df.info())
print(df)
```

```
Thông tin DataFrame vừa tạo:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Name    15 non-null     object
 1   Age     15 non-null     object
 2   Salary  15 non-null     object
dtypes: object(3)
memory usage: 488.0+ bytes
None
       Name Age Salary
0     Alice  25  50000
1       Bob  30  60000
2   Charlie  35  70000
3     David  28  55000
4       Eva  22  52000
5     Frank  45  80000
6     Grace  34  72000
7    Hannah  31  68000
8      Ivan  27  61000
9      Jack  29  59000
10     Kely  33  63000
11     Liam  40  77000
12     Mona  26  53000
13     Nina  32  66000
14    Oscar  36  75000
```

3. Lọc các hàng trong DataFrame có ‘Age’ lớn hơn 28

[7]:
```
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')
```

[8]:
```
df_lonhon_28 = df[df['Age'] > 28]
print("\nHàng có 'Age' lớn hơn 28:")
print(df_lonhon_28)
```

```
Hàng có 'Age' lớn hơn 28:
       Name  Age Salary
1       Bob   30  60000
2   Charlie   35  70000
5     Frank   45  80000
6     Grace   34  72000
7    Hannah   31  68000
9      Jack   29  59000
10     Kely   33  63000
11     Liam   40  77000
13     Nina   32  66000
14    Oscar   36  75000
```

4. Tính giá trị trung bình của cột ‘Salary’.

[9]:
```
df['Salary'] = pd.to_numeric(df['Salary'], errors='coerce')
```

[10]:
```
Gia_Tri_BT = df['Salary'].mean()
print("\nGiá trị trung bình của cột 'Salary' là:",Gia_Tri_BT)
```

```
Giá trị trung bình của cột 'Salary' là: 64066.666666666664
```

5. Nhóm dữ liệu theo cột ‘Age’ và tính tông ‘Salary’ cho môi nhóm

[11]:
```
dfTongSa = df.groupby('Age')["Salary"].sum()
```

[12]:
```
print("\nTổng 'Salary' cho mỗi nhóm 'Age':")
print(dfTongSa)
```

```
Tổng 'Salary' cho mỗi nhóm 'Age':
Age
22    52000
25    50000
26    53000
27    61000
28    55000
29    59000
30    60000
```

```
31    68000
32    66000
33    63000
34    72000
35    70000
36    75000
40    77000
45    80000
Name: Salary, dtype: int64
```

6. Sắp xếp DataFrame theo cột 'Salary' giảm dần

[13]:
```
df_Sap_Xep_Data = df.sort_values(by='Salary', ascending=False)
```

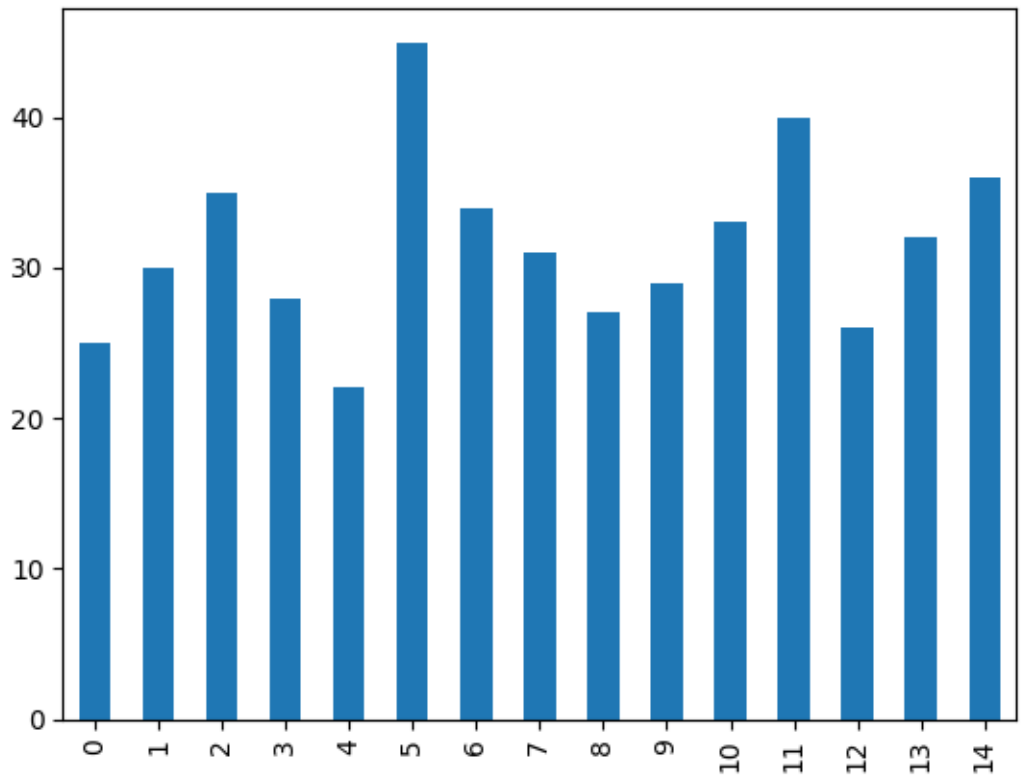
[14]:
```
print(df_Sap_Xep_Data)
```

```
       Name  Age  Salary
5     Frank   45   80000
11     Liam   40   77000
14    Oscar   36   75000
6     Grace   34   72000
2   Charlie   35   70000
7    Hannah   31   68000
13     Nina   32   66000
10     Kely   33   63000
8      Ivan   27   61000
1       Bob   30   60000
9      Jack   29   59000
3     David   28   55000
12     Mona   26   53000
4       Eva   22   52000
0     Alice   25   50000
```

7. Vẽ biểu đồ cột cho cột 'Age'

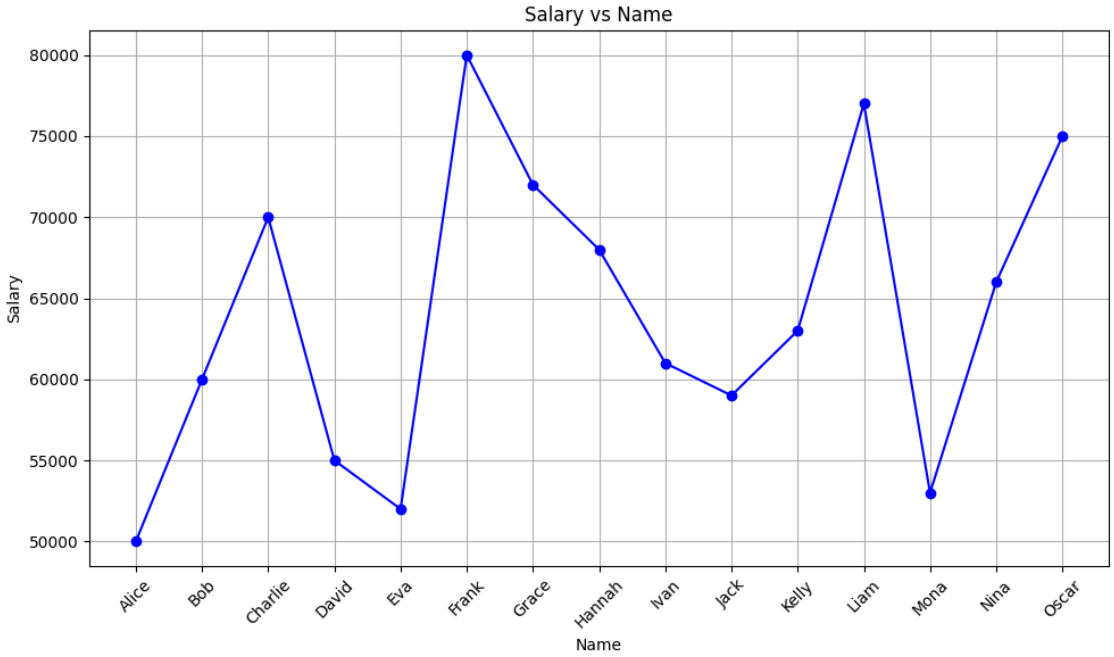
[15]:
```
data = {
    "Name": ["Alice", "Bob", "Charlie", "David", "Eva", "Frank", "Grace",␣
 ↪"Hannah", "Ivan", "Jack", "Kelly", "Liam", "Mona", "Nina", "Oscar"],
    "Age": [25, 30, 35, 28, 22, 45, 34, 31, 27, 29, 33, 40, 26, 32, 36],
    "Salary": [50000, 60000, 70000, 55000, 52000, 80000, 72000, 68000, 61000,␣
 ↪59000, 63000, 77000, 53000, 66000, 75000]
}

df = pd.DataFrame(data)

df['Age'].plot.bar()
```

[15]:
```
<Axes: >
```



8. Vẽ biểu đồ đường cho cột 'Salary'.

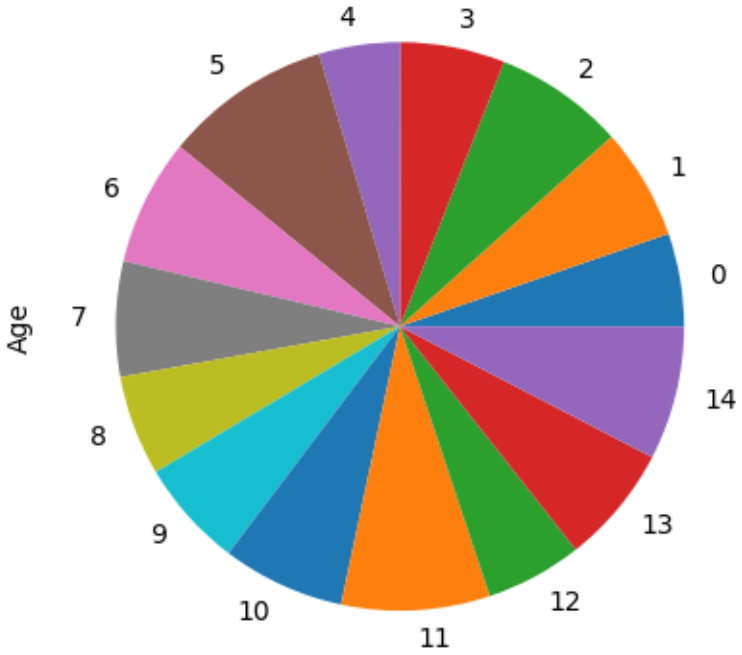
[16]:
```
data = {
    "Name": ["Alice", "Bob", "Charlie", "David", "Eva", "Frank", "Grace",␣
 ↪"Hannah", "Ivan", "Jack", "Kelly", "Liam", "Mona", "Nina", "Oscar"],
    "Age": [25, 30, 35, 28, 22, 45, 34, 31, 27, 29, 33, 40, 26, 32, 36],
    "Salary": [50000, 60000, 70000, 55000, 52000, 80000, 72000, 68000, 61000,␣
 ↪59000, 63000, 77000, 53000, 66000, 75000]
}


plt.figure(figsize=(10, 6))
plt.plot(df['Name'], df['Salary'], marker='o', linestyle='-', color='b')
plt.xlabel('Name')
plt.ylabel('Salary')
plt.title('Salary vs Name')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



9. Vẽ biểu đồ tròn cho cột 'Age'.

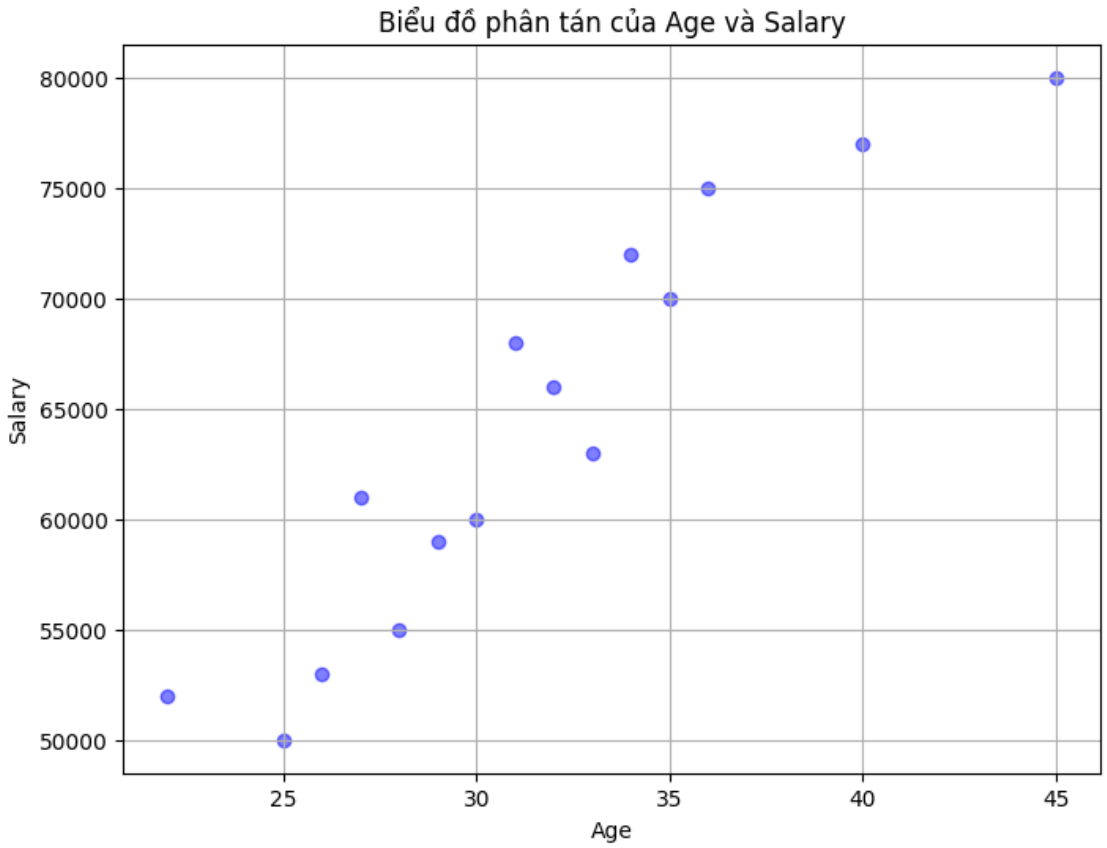
[17]:
```
df['Age'].plot.pie()
```

[17]:
```
<Axes: ylabel='Age'>
```



10. Vẽ biểu đồ phân tán cho ‘Age’ và ‘Salary’.

[18]:
```
data = {
    "Age": [25, 30, 35, 28, 22, 45, 34, 31, 27, 29, 33, 40, 26, 32, 36],
    "Salary": [50000, 60000, 70000, 55000, 52000, 80000, 72000, 68000, 61000,␣
 ↪59000, 63000, 77000, 53000, 66000, 75000]
}


plt.figure(figsize=(8, 6))
plt.scatter(df['Age'], df['Salary'], color='blue', alpha=0.5)
plt.title('Biểu đồ phân tán của Age và Salary')
plt.xlabel('Age')
plt.ylabel('Salary')
plt.grid(True)
plt.show()
```



[19]:
```
data = {
    "Name": ["Alice", "Bob", "Charlie", "David", "Eva", "Frank", "Grace",␣
 ↪"Hannah", "Ivan", "Jack", "Kelly", "Liam", "Mona", "Nina", "Oscar"],
    "Age": [25, 30, 35, 28, 22, 45, 34, 31, 27, 29, 33, 40, 26, 32, 36],
    "Salary": [50000, 60000, 70000, 55000, 52000, 80000, 72000, 68000, 61000,␣
 ↪59000, 63000, 77000, 53000, 66000, 75000]
}
```

11. Kiểm tra xem có giá trị NaN nào trong DataFrame không.

[20]:
```
if df.isna().any().any():
    print("Chứa giá trị NaN.")
else:
    print("Không chứa giá trị NaN.")
```

```
Không chứa giá trị NaN.
```

12. Thay thế các giá trị của cột 'Age' lớn hơn 30 bằng giá trị trung bình của cột đó.

[21]:
```
age_trungbinh = df['Age'].mean()
```

[22]:
```
df['Age'] = df['Age'].apply(lambda x: age_trungbinh if x > 30 else x)
```

[23]:
```
print(df)
```

```
      Name        Age  Salary
0    Alice  25.000000   50000
1      Bob  30.000000   60000
2  Charlie  31.533333   70000
3    David  28.000000   55000
4      Eva  22.000000   52000
5    Frank  31.533333   80000
6    Grace  31.533333   72000
7   Hannah  31.533333   68000
8     Ivan  27.000000   61000
9     Jack  29.000000   59000
10   Kelly  31.533333   63000
11    Liam  31.533333   77000
12    Mona  26.000000   53000
13    Nina  31.533333   66000
14   Oscar  31.533333   75000
```

13. Chuẩn hóa (normalize) cột 'Age' về khoảng giá trị từ 0 đến 1.

[24]:
```
df['Age_ChuanHoa'] = (df['Age'] - df['Age'].min()) / (df['Age'].max() -␣
↪df['Age'].min())
```

[25]:
```
print(df)
```

```
      Name        Age  Salary  Age_ChuanHoa
0    Alice  25.000000   50000      0.314685
1      Bob  30.000000   60000      0.839161
2  Charlie  31.533333   70000      1.000000
3    David  28.000000   55000      0.629371
4      Eva  22.000000   52000      0.000000
5    Frank  31.533333   80000      1.000000
6    Grace  31.533333   72000      1.000000
7   Hannah  31.533333   68000      1.000000
8     Ivan  27.000000   61000      0.524476
9     Jack  29.000000   59000      0.734266
10   Kelly  31.533333   63000      1.000000
11    Liam  31.533333   77000      1.000000
12    Mona  26.000000   53000      0.419580
13    Nina  31.533333   66000      1.000000
14   Oscar  31.533333   75000      1.000000
```

14. Tạo một cột mới 'Age_group' phân loại tuổi thành 'Young', 'Middle-aged' và 'Old' dựa trên giá trị của cột 'Age'.

[26]:
```
def categorize_age(age):
    if age < 30:
        return 'Young'
    elif age >= 30 and age < 40:
        return 'Middle-aged'
    else:
        return 'Old'
```

[27]:
```
df['Age_group'] = df['Age'].apply(lambda x: categorize_age(x))
```

[28]:
```
print(df)
```

```
      Name        Age  Salary  Age_ChuanHoa    Age_group
0    Alice  25.000000   50000      0.314685        Young
1      Bob  30.000000   60000      0.839161  Middle-aged
2  Charlie  31.533333   70000      1.000000  Middle-aged
3    David  28.000000   55000      0.629371        Young
4      Eva  22.000000   52000      0.000000        Young
5    Frank  31.533333   80000      1.000000  Middle-aged
6    Grace  31.533333   72000      1.000000  Middle-aged
7   Hannah  31.533333   68000      1.000000  Middle-aged
8     Ivan  27.000000   61000      0.524476        Young
9     Jack  29.000000   59000      0.734266        Young
10   Kelly  31.533333   63000      1.000000  Middle-aged
11    Liam  31.533333   77000      1.000000  Middle-aged
12    Mona  26.000000   53000      0.419580        Young
13    Nina  31.533333   66000      1.000000  Middle-aged
14   Oscar  31.533333   75000      1.000000  Middle-aged
```

15. Tính toán tỷ lệ phần trăm thay đổi (percentage change) của cột 'Salary'.

[29]:
```
df['Salary_pct_change'] = df['Salary'].pct_change() * 100
```

[30]:
```
print(df)
```

```
      Name        Age  Salary  Age_ChuanHoa    Age_group  Salary_pct_change
0    Alice  25.000000   50000      0.314685        Young                NaN
1      Bob  30.000000   60000      0.839161  Middle-aged          20.000000
2  Charlie  31.533333   70000      1.000000  Middle-aged          16.666667
3    David  28.000000   55000      0.629371        Young         -21.428571
4      Eva  22.000000   52000      0.000000        Young          -5.454545
5    Frank  31.533333   80000      1.000000  Middle-aged          53.846154
6    Grace  31.533333   72000      1.000000  Middle-aged         -10.000000
7   Hannah  31.533333   68000      1.000000  Middle-aged          -5.555556
8     Ivan  27.000000   61000      0.524476        Young         -10.294118
9     Jack  29.000000   59000      0.734266        Young          -3.278689
10   Kelly  31.533333   63000      1.000000  Middle-aged           6.779661
11    Liam  31.533333   77000      1.000000  Middle-aged          22.222222
```

12	Mona	26.000000	53000	0.419580	Young	-31.168831
13	Nina	31.533333	66000	1.000000	Middle-aged	24.528302
14	Oscar	31.533333	75000	1.000000	Middle-aged	13.636364

16. Tìm các giá trị trùng lặp trong DataFrame dựa trên cột 'Name' và loại bỏ các hàng trùng lặp, giữ lại hàng đầu tiên.

[31]:
```
df.drop_duplicates(subset=['Name'], keep='first', inplace=True)
```

[32]:
```
print(df)
```

	Name	Age	Salary	Age_ChuanHoa	Age_group	Salary_pct_change
0	Alice	25.000000	50000	0.314685	Young	NaN
1	Bob	30.000000	60000	0.839161	Middle-aged	20.000000
2	Charlie	31.533333	70000	1.000000	Middle-aged	16.666667
3	David	28.000000	55000	0.629371	Young	-21.428571
4	Eva	22.000000	52000	0.000000	Young	-5.454545
5	Frank	31.533333	80000	1.000000	Middle-aged	53.846154
6	Grace	31.533333	72000	1.000000	Middle-aged	-10.000000
7	Hannah	31.533333	68000	1.000000	Middle-aged	-5.555556
8	Ivan	27.000000	61000	0.524476	Young	-10.294118
9	Jack	29.000000	59000	0.734266	Young	-3.278689
10	Kelly	31.533333	63000	1.000000	Middle-aged	6.779661
11	Liam	31.533333	77000	1.000000	Middle-aged	22.222222
12	Mona	26.000000	53000	0.419580	Young	-31.168831
13	Nina	31.533333	66000	1.000000	Middle-aged	24.528302
14	Oscar	31.533333	75000	1.000000	Middle-aged	13.636364

17.Xuất file csv

[34]:
```
csv_filename = "207CT47823_LeDangKimLan_BaiKiemTraSo1.csv"

df.to_csv(csv_filename, index=False)

print(f"File CSV đã được lưu thành công: {csv_filename}")
```

```
File CSV đã được lưu thành công: 207CT47823_LeDangKimLan_BaiKiemTraSo1.csv
```