

# Chapter 3 - Ex2: Data Exploratation - KPIs

- Cho dữ liệu bank marketing data (UCI's Bank Marketing Data Set: link:  
<https://archive.ics.uci.edu/ml/datasets/bank+marketing>  
(<https://archive.ics.uci.edu/ml/datasets/bank+marketing>)), tập tin bank-additional-full.csv.
- Dữ liệu có liên quan đến các chiến dịch tiếp thị trực tiếp (các cuộc gọi điện thoại) của một tổ chức ngân hàng Bồ Đào Nha. Mục tiêu phân loại là để dự đoán liệu khách hàng sẽ đăng ký một khoản tiền gửi có kỳ hạn hay không (y).

## Yêu cầu

### Phần 1: Chọn và sử dụng package EDA để có cái nhìn ban đầu về dữ liệu

### Phần 2: Thực hiện các công việc sau:

- Xác định các thuộc tính
- Phân tích đơn biến
  - Để dự đoán một khách hàng sẽ đăng ký (subscribe =yes/no) cho một khoản tiền gửi có kỳ hạn (term deposit - variable y) hay không chúng ta cần các thông tin trong dữ liệu, hãy chọn một số thuộc tính và phân tích.
- Phân tích hai biến
- Xử lý dữ liệu trùng, thiếu
- Phát hiện và xử lý ngoại lệ

### Phần 3: Tính toán và trực quan hóa KPI, chủ yếu tập trung vào phân tích conversion rates

1. Tạo cột conversion: dựa trên cột y với giá trị 'yes' => 1 và 'no' => 0
2. Tính toán tổng conversion và conversion rate
3. Tính toán conversion rate theo từng age. Trực quan hóa kết quả.
4. Tính toán conversion rate theo từng nhóm age (nhóm 18-30, nhóm 30-40, nhóm 40 - 50, nhóm 50-60, nhóm 60-70, nhóm >70. Trực quan hóa kết quả.
5. So sánh giữa conversion và non-conversion cho từng nhóm marital status. Trực quan hóa kết quả.
6. So sánh giữa conversion và non-conversion cho từng nhóm Age Groups & Marital Status. Trực quan hóa kết quả.

```
In [ ]: !pip install dataprep  
#!pip install https://github.com/pandas-profiling/pandas-profiling/archive/master  
# !pip install ttth-mds5-analyzer
```

```
In [2]: # from google.colab import drive  
# drive.mount("/content/gdrive", force_remount=True)  
# %cd '/content/gdrive/My Drive/MDS5_2022/Practice_2022/Chapter3/'
```

Mounted at /content/gdrive  
/content/gdrive/My Drive/MDS5\_2022/Practice\_2022/Chapter3

```
In [3]: import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
import scipy
```

```
In [4]: # Các function tự viết  
from EDA_funcs import *
```

```
In [5]: df = pd.read_csv("bank-additional/bank-additional-full.csv", sep=';')
```

```
In [6]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 41188 entries, 0 to 41187  
Data columns (total 21 columns):  
 #   Column           Non-Null Count  Dtype     
---  --     
 0   age              41188 non-null    int64    
 1   job              41188 non-null    object    
 2   marital          41188 non-null    object    
 3   education        41188 non-null    object    
 4   default          41188 non-null    object    
 5   housing          41188 non-null    object    
 6   loan              41188 non-null    object    
 7   contact          41188 non-null    object    
 8   month             41188 non-null    object    
 9   day_of_week       41188 non-null    object    
 10  duration         41188 non-null    int64    
 11  campaign          41188 non-null    int64    
 12  pdays            41188 non-null    int64    
 13  previous          41188 non-null    int64    
 14  poutcome          41188 non-null    object    
 15  emp.var.rate      41188 non-null    float64   
 16  cons.price.idx    41188 non-null    float64   
 17  cons.conf.idx     41188 non-null    float64   
 18  euribor3m         41188 non-null    float64   
 19  nr.employed       41188 non-null    float64   
 20  y                 41188 non-null    object    
dtypes: float64(5), int64(5), object(11)  
memory usage: 6.6+ MB
```

## Gợi ý Phần 1

```
In [7]: # Dùng pandas_profiling  
# import pandas_profiling as pp
```

```
In [8]: # profile = pp.ProfileReport(df)
```

```
In [9]: # profile
```

```
In [10]: # Dùng dataprep  
from dataprep.datasets import load_dataset  
from dataprep.eda import create_report, plot
```

```
In [11]: # REPORT of DATA  
report = create_report(df)
```

```
In [ ]: report
```

## Gợi ý Phần 2

```
In [13]: df.head()
```

```
Out[13]:
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	..
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	..
2	37	services	married	high.school	no	yes	no	telephone	may	mon	..
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	..
4	56	services	married	high.school	no	no	yes	telephone	may	mon	..

5 rows × 21 columns

### 1. Xác định các thuộc tính

- 1.1. Input: <> y (term deposit)
- 1.2. Output: y
- 1.3. Type of variable:
  - 1.3.1 Predictor: khác y
  - 1.3.2 Target: y
- 1.4. Data Type:
  - 1.4.1 Charactor/String
  - 1.4.2 Numeric
- 1.5. Variable Category:

```
In [14]: numbers = [f for f in df.columns if df.dtypes[f] != 'object'] # Quantitative: #
```

```
In [15]: numbers
```

```
Out[15]: ['age',
'duration',
'campaign',
'pdays',
'previous',
'emp.var.rate',
'cons.price.idx',
'cons.conf.idx',
'euribor3m',
'nr.employed']
```

```
In [16]: objects = [f for f in df.columns if df.dtypes[f] == 'object'] # Qualitative : # categorical
```

```
In [17]: objects
```

```
Out[17]: ['job',
'marital',
'education',
'default',
'housing',
'loan',
'contact',
'month',
'day_of_week',
'poutcome',
'y']
```

### 1.5. Variable

1.5.1 Categorical: kiểu số, kiểu chuỗi

1.5.2 Continuous: int, float

In [18]: # Categorical:

```
# Xem xét các biến phân loại kiểu chuỗi
i = 1
for obj in objects:
    print(i, "/", obj, "\t", len(df[obj].unique())),
    ":" , df[obj].unique())
    i = i+1
```

```
1 / job           12 : ['housemaid' 'services' 'admin.' 'blue-collar' 'technicia
n' 'retired'
'management' 'unemployed' 'self-employed' 'unknown' 'entrepreneur'
'student']
2 / marital       4 : ['married' 'single' 'divorced' 'unknown']
3 / education     8 : ['basic.4y' 'high.school' 'basic.6y' 'basic.9y' 'professio
nal.course'
'unknown' 'university.degree' 'illiterate']
4 / default        3 : ['no' 'unknown' 'yes']
5 / housing        3 : ['no' 'yes' 'unknown']
6 / loan            3 : ['no' 'yes' 'unknown']
7 / contact         2 : ['telephone' 'cellular']
8 / month          10 : ['may' 'jun' 'jul' 'aug' 'oct' 'nov' 'dec' 'mar' 'apr' 's
ep']
9 / day_of_week     5 : ['mon' 'tue' 'wed' 'thu' 'fri']
10 / poutcome       3 : ['nonexistent' 'failure' 'success']
11 / y              2 : ['no' 'yes']
```

In [19]: # Quan sát kết quả trên để quyết định cột kiểu chuỗi nào Là phân Loại?

```
In [20]: # Categorical:  
# Xem xét các biến phân loại kiểu số  
i = 1  
for obj in numbers:  
    print(i, "/", obj, "\t", len(df[obj].unique()), ":" ,  
          df[obj].unique() if len(df[obj].unique()) < 50 else '')  
    i = i+1
```

```
1 / age           78 :  
2 / duration     1544 :  
3 / campaign     42 : [ 1  2  3  4  5  6  7  8  9 10 11 12 13 19 18 23 14 22 25  
16 17 15 20 56  
39 35 42 28 26 27 32 21 24 29 31 30 41 37 40 33 34 43]  
4 / pdays         27 : [999   6   4   3   5   1   0   10   7   8   9   11   2   12  
13 14 15 16  
21 17 18 22 25 26 19 27 20]  
5 / previous      8 : [0 1 2 3 4 5 6 7]  
6 / emp.var.rate   10 : [ 1.1  1.4 -0.1 -0.2 -1.8 -2.9 -3.4 -3. -1.7 -1.  
1]  
7 / cons.price.idx 26 : [93.994 94.465 93.918 93.444 93.798 93.2  92.756  
92.843 93.075 92.893  
92.963 92.469 92.201 92.379 92.431 92.649 92.713 93.369 93.749 93.876  
94.055 94.215 94.027 94.199 94.601 94.767]  
8 / cons.conf.idx  26 : [-36.4 -41.8 -42.7 -36.1 -40.4 -42. -45.9 -50.  
-47.1 -46.2 -40.8 -33.6  
-31.4 -29.8 -26.9 -30.1 -33. -34.8 -34.6 -40. -39.8 -40.3 -38.3 -37.5  
-49.5 -50.8]  
9 / euribor3m      316 :  
10 / nr.employed   11 : [5191.  5228.1 5195.8 5176.3 5099.1 5076.2 5017.5  
5023.5 5008.7 4991.6  
4963.6]
```

```
In [21]: # Quan sát kết quả trên để quyết định cột kiểu số nào là phân loại
```

## 2. Phân tích đơn biến

- Dựa trên kết quả phần 1, thực hiện phân tích đơn biến

```
In [22]: %matplotlib inline
```

```
In [23]: df.columns
```

```
Out[23]: Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',  
'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',  
'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',  
'cons.conf.idx', 'euribor3m', 'nr.employed', 'y'],  
dtype='object')
```

### Continuous variable

```
In [24]: # age  
univariate_analysis_continuous_variable(df, df['age'])
```

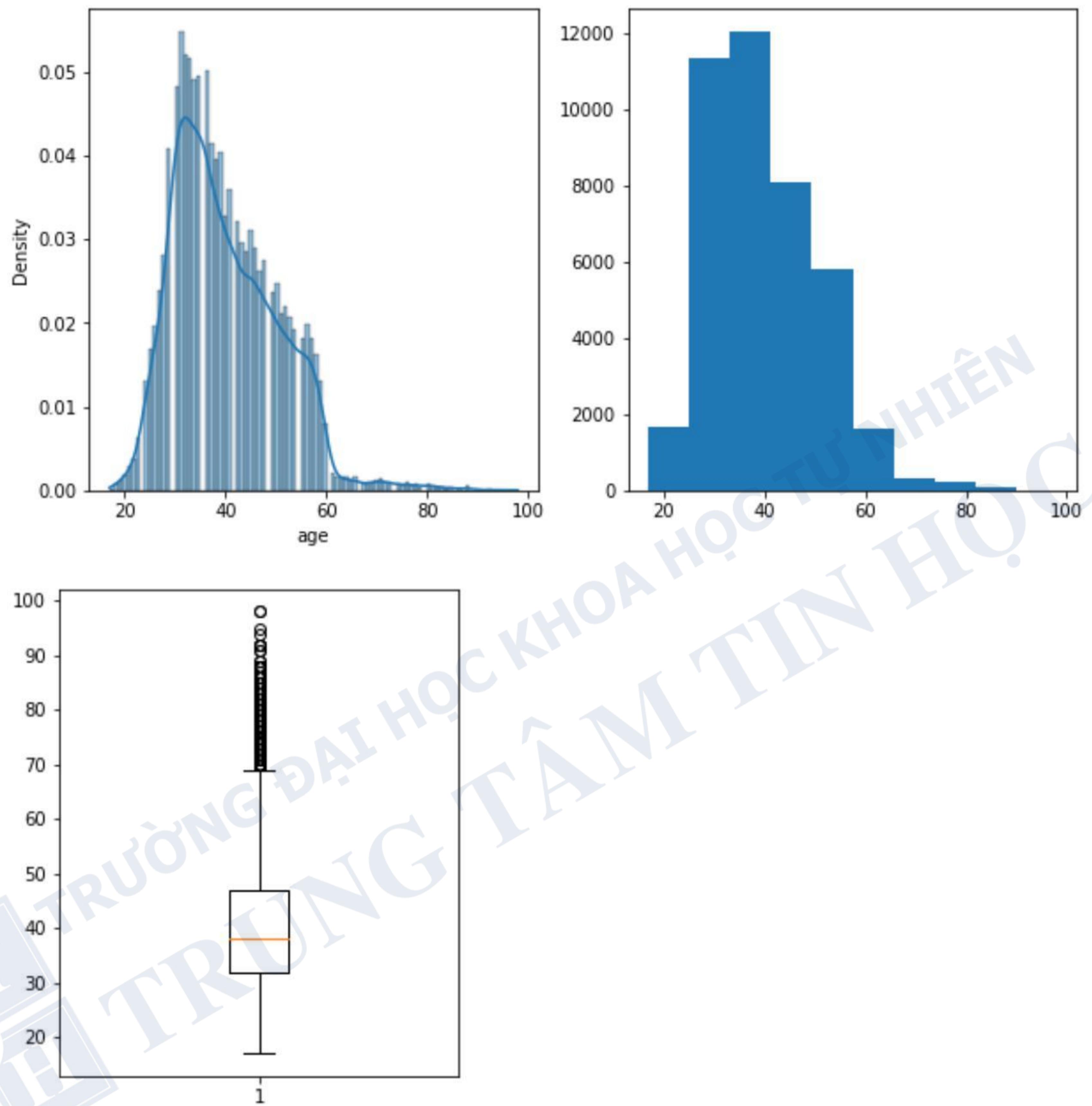
Describe:

```
count    41188.00000  
mean      40.02406  
std       10.42125  
min       17.00000  
25%      32.00000  
50%      38.00000  
75%      47.00000  
max       98.00000  
Name: age, dtype: float64  
Mode: 0    31  
dtype: int64  
Range: 81  
IQR: 15.0  
Var: 108.60245116511788  
Std: 10.421249980934048  
Skew: 0.7846968157646645  
Kurtosis: 0.7913115311544336
```

```
In [25]: # skew > 0 : phân phối Lệch phải
```

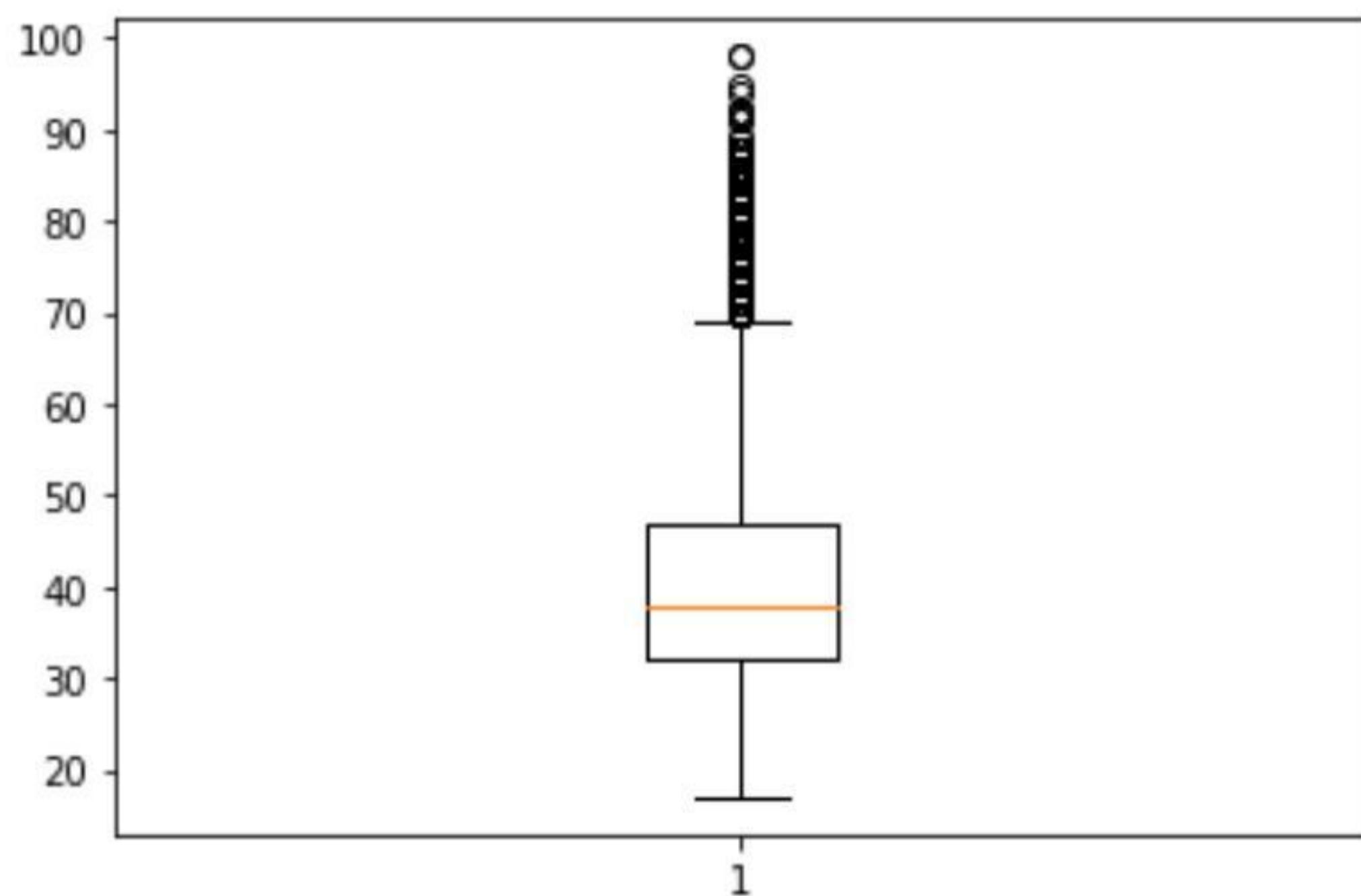
```
In [26]: # kur > 0: phân phối nhọn hơn phân phối chuẩn
```

```
In [27]: univariate_visualization_analysis_continuous_variable(df[ 'age' ])
```



```
In [28]: # Quan sát thấy có outlier trên ~68 tuổi
```

```
In [29]: Q1_age, Q3_age, n_0_upper_age, n_0_lower_age,  
        outliers_per_age = check_outlier(df, df['age'])
```



Number of upper outliers: 469  
Number of lower outliers: 0  
Percentage of outliers: 0.011386811692726036

```
In [30]: outliers_per_age *100
```

```
Out[30]: 1.1386811692726035
```

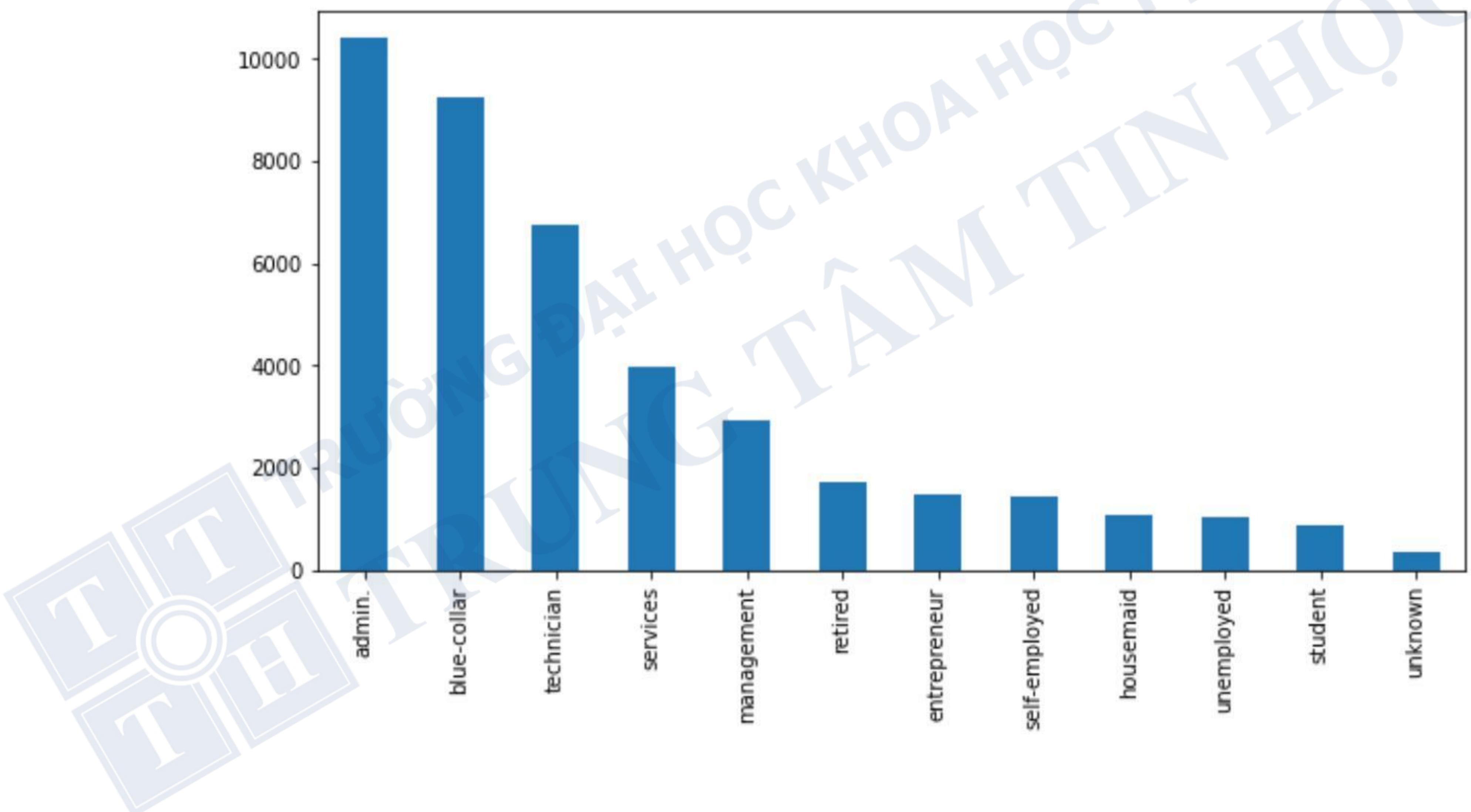
```
In [31]: # Chỉ có ~1% dữ liệu age có outlier  
# Xem xét loại bỏ outliers ??? => tùy vào nghiệp vụ của từng lĩnh vực
```

```
In [32]: # Note: Các biến/thuộc tính còn lại làm tương tự như trên
```

## Categorical Variables

```
In [35]: #job  
univariate_analysis_categorical_variable_2(df, "job")
```

```
admin.          10422  
blue-collar    9254  
technician     6743  
services        3969  
management     2924  
retired         1720  
entrepreneur   1456  
self-employed   1421  
housemaid       1060  
unemployed      1014  
student          875  
unknown           330  
Name: job, dtype: int64
```

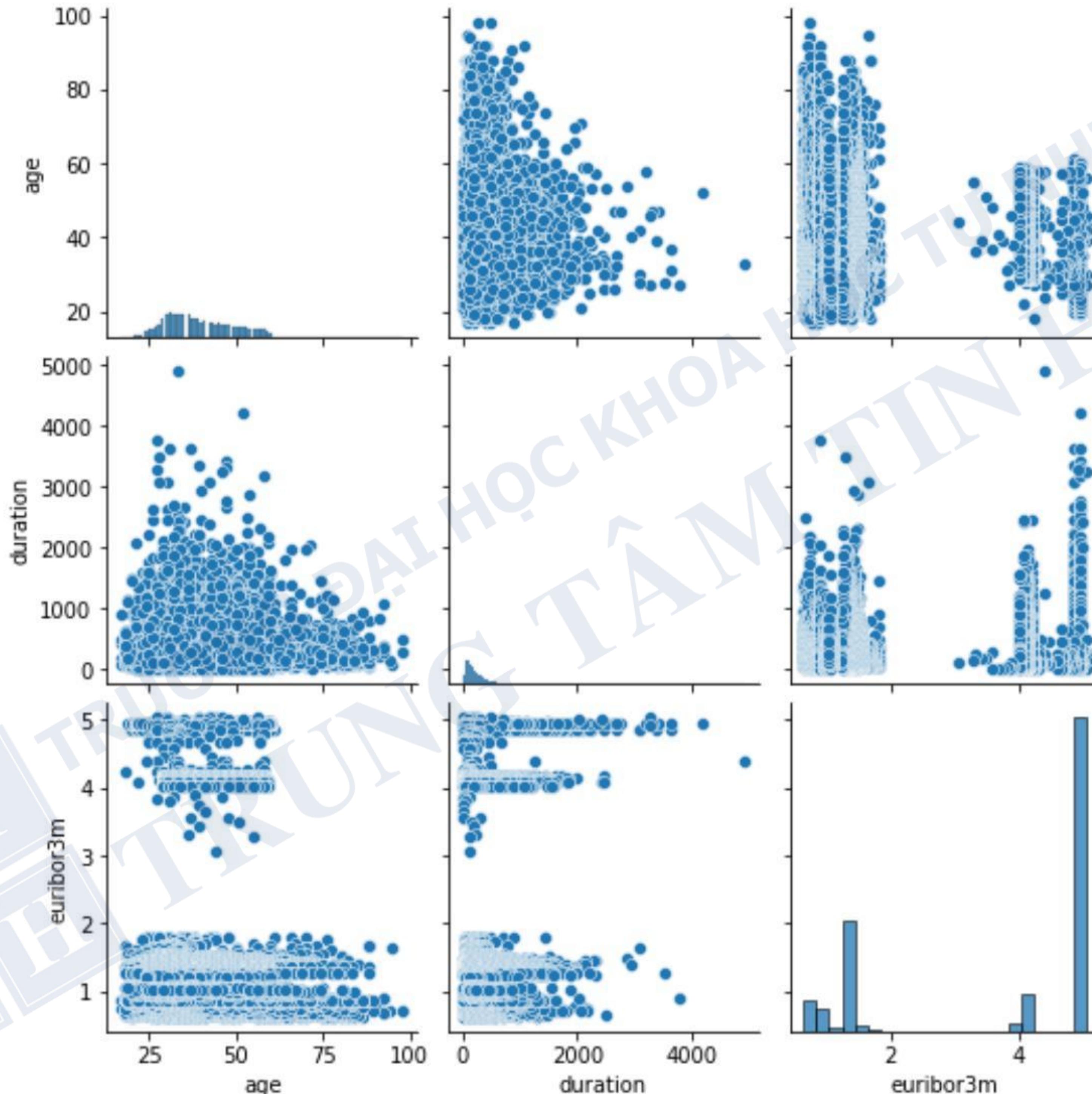


```
In [38]: # Note: Các biến/thuộc tính còn Lại Làm tương tự như trên
```

### 3. Phân tích hai biến

- Continuous & Continuous
- Categorical & Categorical
- Categorical & Continuous

```
In [39]: # Continuous & Continuous  
sns.pairplot(df[['age', 'duration', 'euribor3m']])  
plt.show()
```



```
In [40]: df[['age', 'duration', 'euribor3m']].corr()
```

```
Out[40]:
```

	age	duration	euribor3m
age	1.000000	-0.000866	0.010767
duration	-0.000866	1.000000	-0.032897
euribor3m	0.010767	-0.032897	1.000000

```
In [41]: # 3 biến trên không có quan hệ tuyến tính
```

```
In [42]: # Note: Các biến/thuộc tính còn Lại Làm tương tự như trên
```

```
In [43]: df.corr()
```

Out[43]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	c
age	1.000000	-0.000866	0.004594	-0.034369	0.024365	-0.000371	0.000857	
duration	-0.000866	1.000000	-0.071699	-0.047577	0.020640	-0.027968	0.005312	
campaign	0.004594	-0.071699	1.000000	0.052584	-0.079141	0.150754	0.127836	
pdays	-0.034369	-0.047577	0.052584	1.000000	-0.587514	0.271004	0.078889	
previous	0.024365	0.020640	-0.079141	-0.587514	1.000000	-0.420489	-0.203130	
emp.var.rate	-0.000371	-0.027968	0.150754	0.271004	-0.420489	1.000000	0.775334	
cons.price.idx	0.000857	0.005312	0.127836	0.078889	-0.203130	0.775334	1.000000	
cons.conf.idx	0.129372	-0.008173	-0.013733	-0.091342	-0.050936	0.196041	0.058986	
euribor3m	0.010767	-0.032897	0.135133	0.296899	-0.454494	0.972245	0.688230	
nr.employed	-0.017725	-0.044703	0.144095	0.372605	-0.501333	0.906970	0.522034	

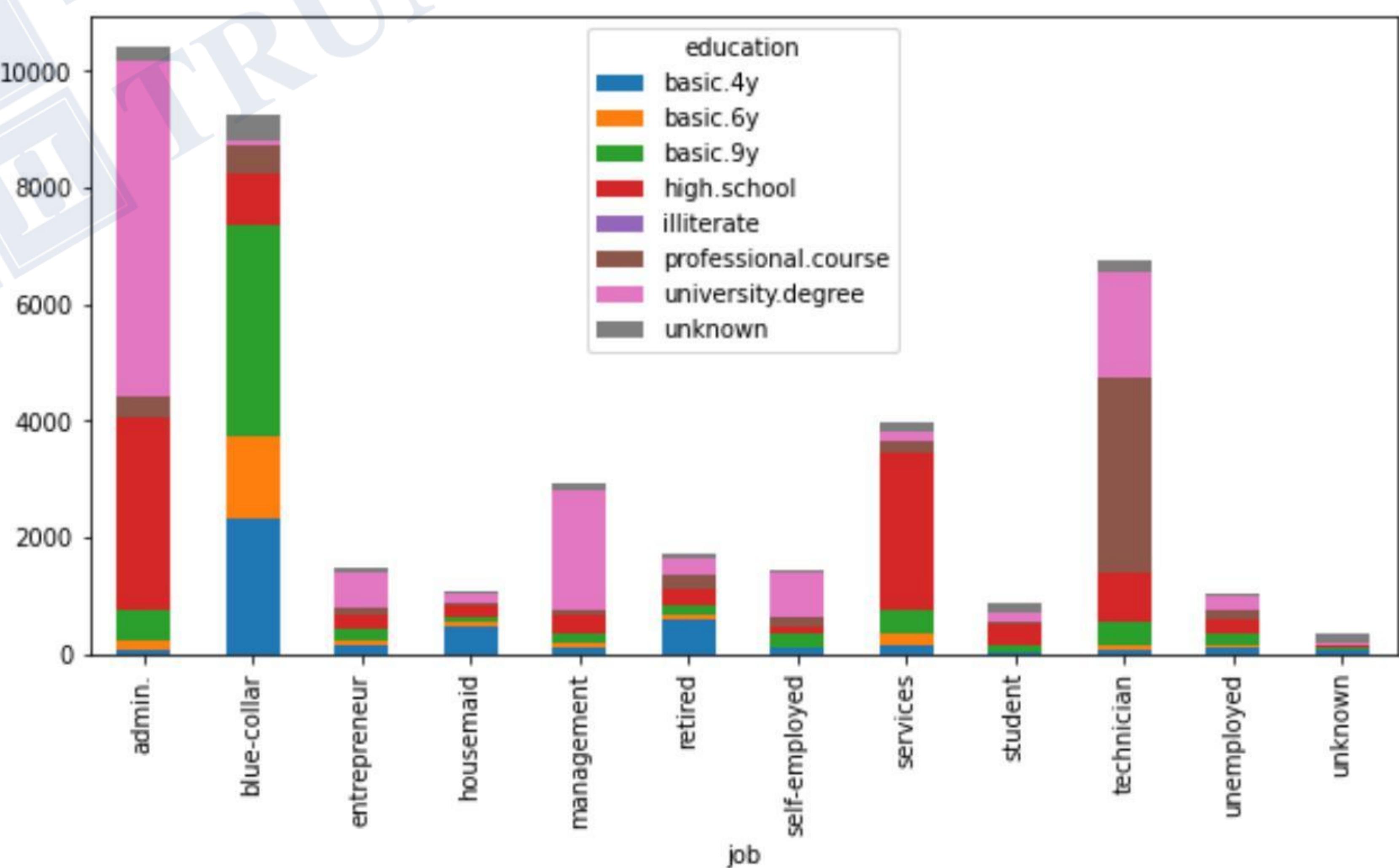
```
In [44]: # Categorical & Categorical
```

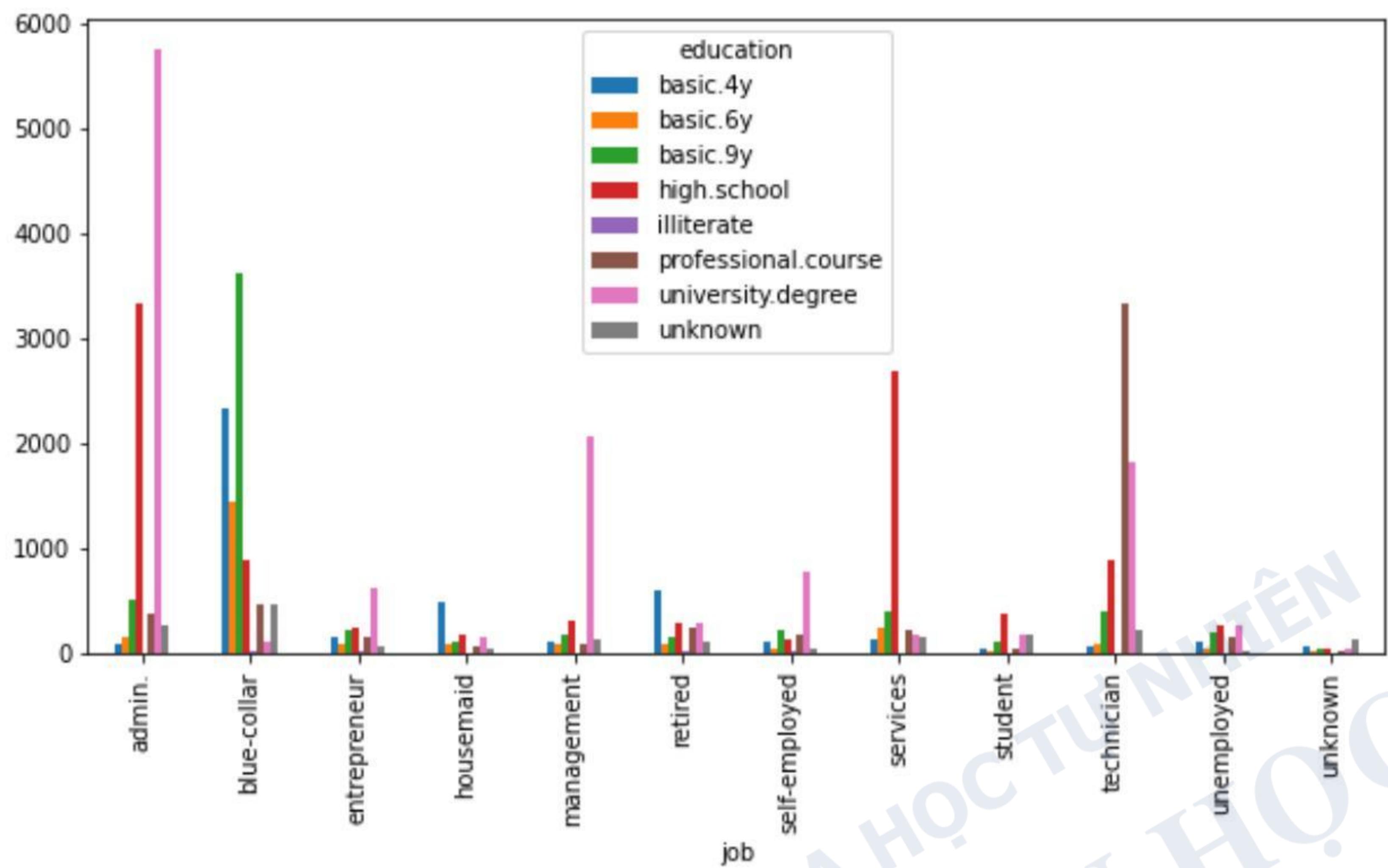
```
In [45]: # job & education
```

```
categorical_categorical(df['job'], df['education'])
```

education	basic.4y	basic.6y	basic.9y	high.school	illiterate	\
job						
admin.	77	151	499	3329	1	
blue-collar	2318	1426	3623	878	8	
entrepreneur	137	71	210	234	2	
housemaid	474	77	94	174	1	
management	100	85	166	298	0	
retired	597	75	145	276	3	
self-employed	93	25	220	118	3	
services	132	226	388	2682	0	
student	26	13	99	357	0	
technician	58	87	384	873	0	
unemployed	112	34	186	259	0	
unknown	52	22	31	37	0	

education	professional.course	university.degree	unknown
job			
admin.	363	5753	249
blue-collar	453	94	454
entrepreneur	135	610	57
housemaid	59	139	42
management	89	2063	123
retired	241	285	98
self-employed	168	765	29
services	218	173	150
student	43	170	167
technician	3320	1809	212
unemployed	142	262	19
unknown	12	45	131





dof=77  
 p= 0.0  
 probability=0.950, critical=98.484, stat=37338.135  
 significance=0.050, p=0.000  
 Dependent (reject H0)

In [49]: # Note: Các biến/thuộc tính còn Lại Làm tương tự như trên

In [50]: # Categorical & Continuous

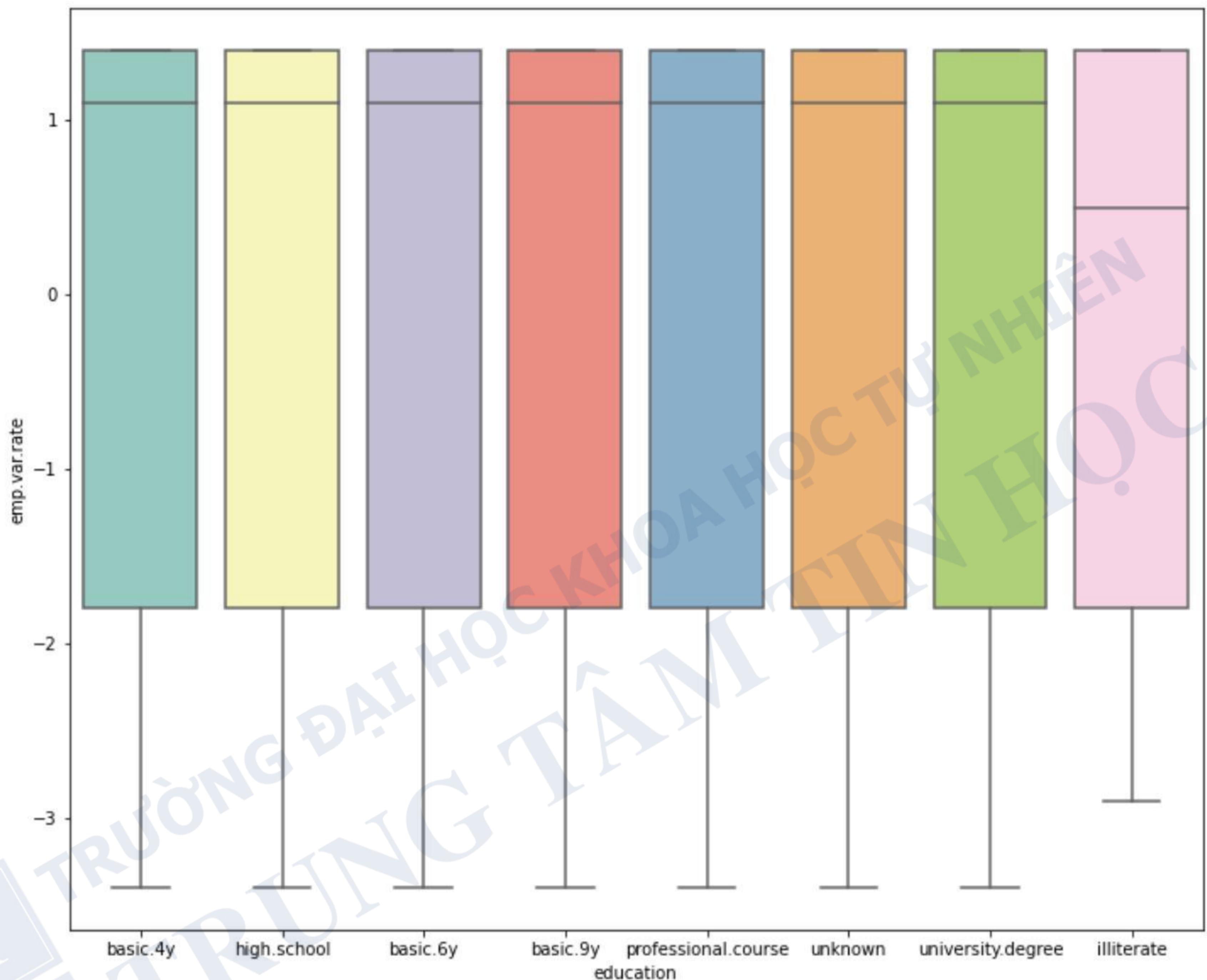
In [51]: # education và emp.var.rate có mối quan hệ gì với nhau hay không?

In [52]: df\_sub = df[['education', 'emp.var.rate']]  
df\_sub.head()

Out[52]:

	education	emp.var.rate
0	basic.4y	1.1
1	high.school	1.1
2	high.school	1.1
3	basic.6y	1.1
4	high.school	1.1

```
In [53]: import matplotlib.pyplot as plt  
plt.figure(figsize=(12,10))  
sns.boxplot(x="education", y="emp.var.rate",  
            data=df_sub, palette="Set3")  
plt.show()
```



```
In [54]: import statsmodels.api as sm  
from statsmodels.formula.api import ols
```

```
In [55]: df_sub.columns = ['education', 'emp_var_rate']
```

```
In [56]: model = ols('emp_var_rate ~ C(education)', data=df_sub).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
```

Out[56]:

	sum_sq	df	F	PR(>F)
C(education)	377.499930	7.0	21.929606	8.505778e-30
Residual	101268.494842	41180.0	NaN	NaN

- Giải thích: P-value thu được từ phân tích ANOVA cho education và em.var.rate phối hợp có ý nghĩa thống kê ( $P < 0.05$ ).
- Kết luận: education ảnh hưởng đáng kể đến em.var.rate

```
In [57]: from statsmodels.stats.multicomp import pairwise_tukeyhsd
# perform multiple pairwise comparison (Tukey HSD)
m_comp = pairwise_tukeyhsd(endog=df_sub['emp_var_rate'],
                           groups=df_sub['education'],
                           alpha=0.05)
print(m_comp)
```

Multiple Comparison of Means - Tukey HSD, FWER=0.05							
group1	group2	meandiff	p-adj	lower	upper	reject	
basic.4y	basic.6y	0.0152	0.9	-0.1083	0.1388	False	
basic.4y	basic.9y	-0.0567	0.6054	-0.1523	0.039	False	
basic.4y	high.school	-0.183	0.001	-0.2712	-0.0948	True	
basic.4y	illiterate	-0.3493	0.9	-1.472	0.7735	False	
basic.4y	professional.course	-0.0429	0.8906	-0.1415	0.0557	False	
basic.4y	university.degree	-0.244	0.001	-0.3293	-0.1588	True	
basic.4y	unknown	-0.1568	0.0111	-0.2927	-0.021	True	
basic.6y	basic.9y	-0.0719	0.5618	-0.1885	0.0447	False	
basic.6y	high.school	-0.1982	0.001	-0.3088	-0.0876	True	
basic.6y	illiterate	-0.3645	0.9	-1.4892	0.7602	False	
basic.6y	professional.course	-0.0581	0.7942	-0.1772	0.0609	False	
basic.6y	university.degree	-0.2592	0.001	-0.3675	-0.151	True	
basic.6y	unknown	-0.1721	0.0133	-0.3234	-0.0207	True	
basic.9y	high.school	-0.1263	0.001	-0.2045	-0.0481	True	
basic.9y	illiterate	-0.2926	0.9	-1.4146	0.8294	False	
basic.9y	professional.course	0.0138	0.9	-0.0759	0.1035	False	
basic.9y	university.degree	-0.1873	0.001	-0.2621	-0.1125	True	
basic.9y	unknown	-0.1001	0.2705	-0.2297	0.0294	False	
high.school	illiterate	-0.1663	0.9	-1.2877	0.9551	False	
high.school	professional.course	0.1401	0.001	0.0583	0.2218	True	
high.school	university.degree	-0.061	0.0849	-0.1261	0.004	False	
high.school	unknown	0.0262	0.9	-0.098	0.1504	False	
illiterate	professional.course	0.3063	0.9	-0.8159	1.4286	False	
illiterate	university.degree	0.1052	0.9	-1.0159	1.2264	False	
illiterate	unknown	0.1924	0.9	-0.9337	1.3186	False	
professional.course	university.degree	-0.2011	0.001	-0.2796	-0.1226	True	
professional.course	unknown	-0.1139	0.1483	-0.2457	0.0178	False	
university.degree	unknown	0.0872	0.3751	-0.0349	0.2093	False	

- Các kết quả trên từ Tukey HSD cho thấy một số cặp (False): chấp nhận Ho, các so sánh cặp khác bắc bỏ Ho và chỉ ra sự khác biệt đáng kể về mặt thống kê.

```
In [58]: # Note: Các biến/thuộc tính còn Lại Làm tương tự như trên
```

## 4. Xử lý dữ liệu trùng, thiếu

```
In [59]: # Kiểm tra dữ liệu NaN  
df.isna().sum()
```

```
Out[59]: age          0  
job           0  
marital       0  
education     0  
default        0  
housing        0  
loan           0  
contact        0  
month          0  
day_of_week    0  
duration        0  
campaign        0  
pdays           0  
previous        0  
poutcome        0  
emp.var.rate    0  
cons.price.idx  0  
cons.conf.idx   0  
euribor3m       0  
nr.employed     0  
y                0  
dtype: int64
```

```
In [60]: # Không có dữ liệu NaN
```

```
In [61]: # Kiểm tra dữ liệu thiếu  
df.isnull().sum()
```

```
Out[61]: age          0  
job           0  
marital       0  
education     0  
default        0  
housing        0  
loan           0  
contact        0  
month          0  
day_of_week    0  
duration        0  
campaign        0  
pdays           0  
previous        0  
poutcome        0  
emp.var.rate    0  
cons.price.idx  0  
cons.conf.idx   0  
euribor3m       0  
nr.employed     0  
y                0  
dtype: int64
```

```
In [62]: # Không có dữ liệu thiếu
```

```
In [63]: # Phát hiện dữ liệu trùng  
df.duplicated().sum()
```

```
Out[63]: 12
```

```
In [64]: # Trước khi xóa dữ liệu trùng  
df.shape
```

```
Out[64]: (41188, 21)
```

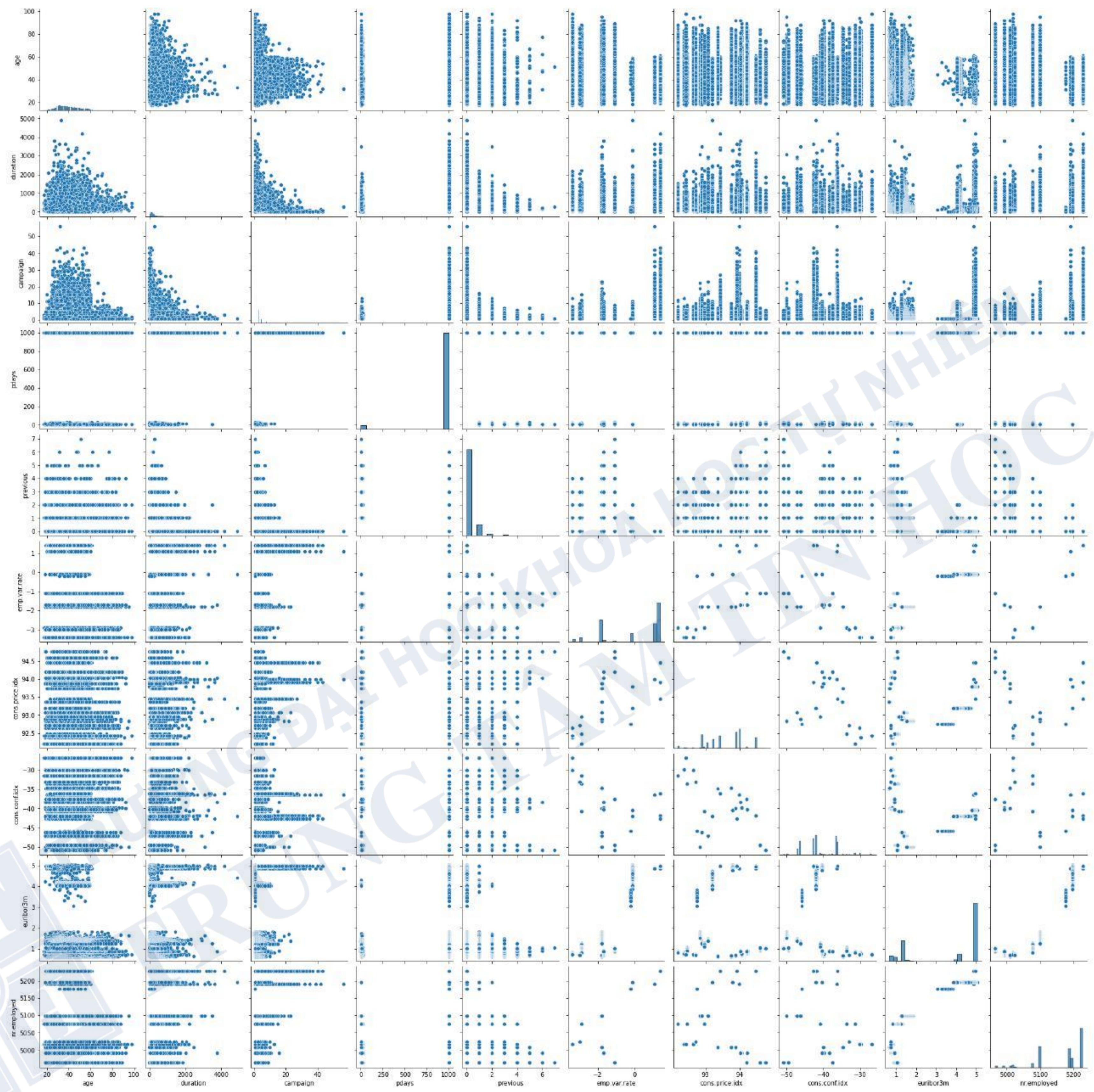
```
In [65]: # Sau khi xóa dữ liệu trùng  
df = df.drop_duplicates()  
df.shape
```

```
Out[65]: (41176, 21)
```

## 5. Phát hiện và xử lý ngoại lệ

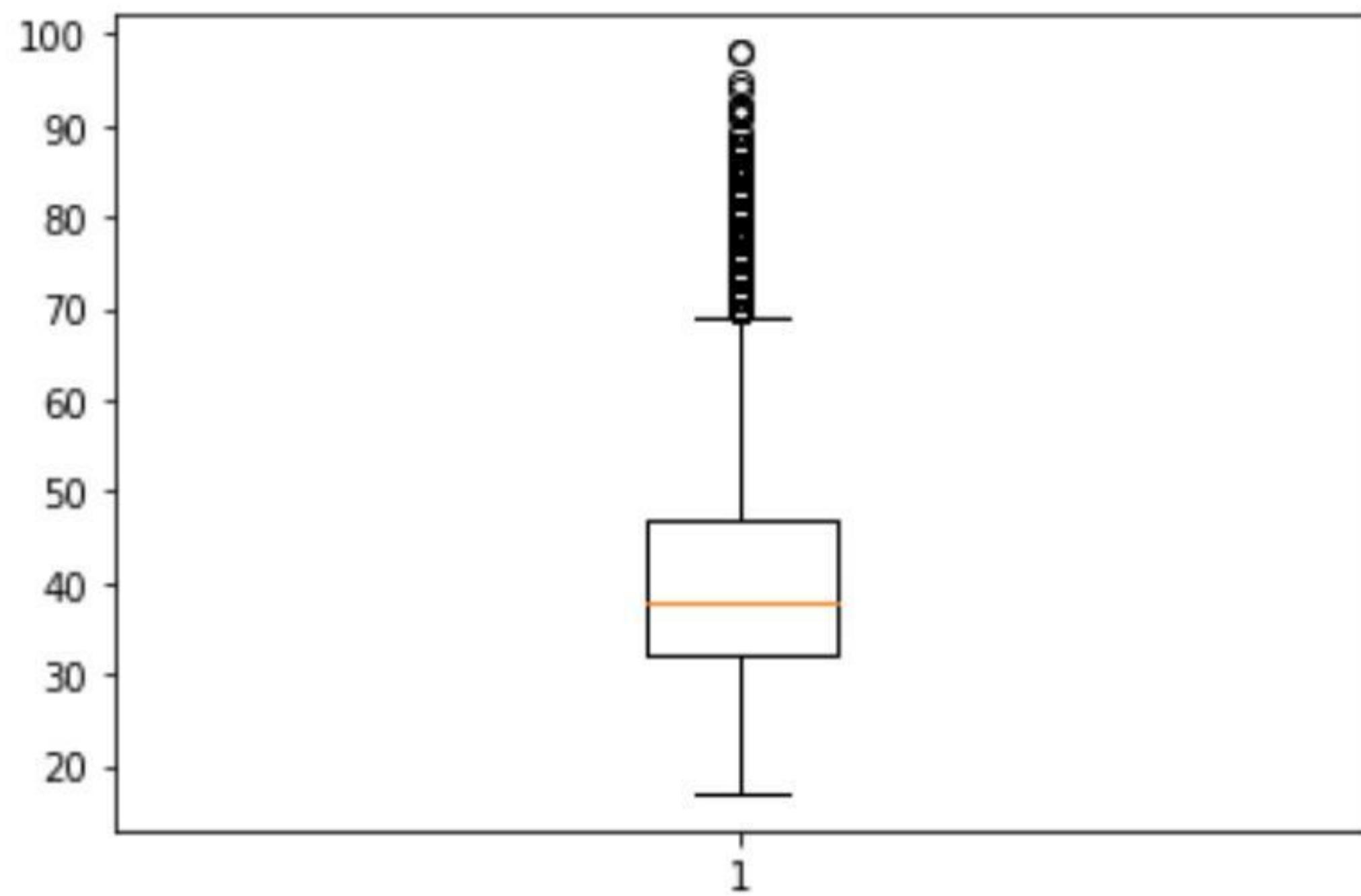
```
In [66]: sns.pairplot(df)
```

```
Out[66]: <seaborn.axisgrid.PairGrid at 0x7f5b8d3a6810>
```



```
In [67]: # Tiếp tục xem xét biến age
```

```
In [68]: Q1_age, Q3_age, n_0_upper_age, n_0_lower_age,  
        outliers_per_age = check_outlier(df, df['age'])
```



Number of upper outliers: 468  
Number of lower outliers: 0  
Percentage of outliers: 0.011365844181076355

```
In [69]: # Có thể drop outliers của 'age': vì tổng số outliers là 1% dữ liệu
```

```
In [70]: df.shape
```

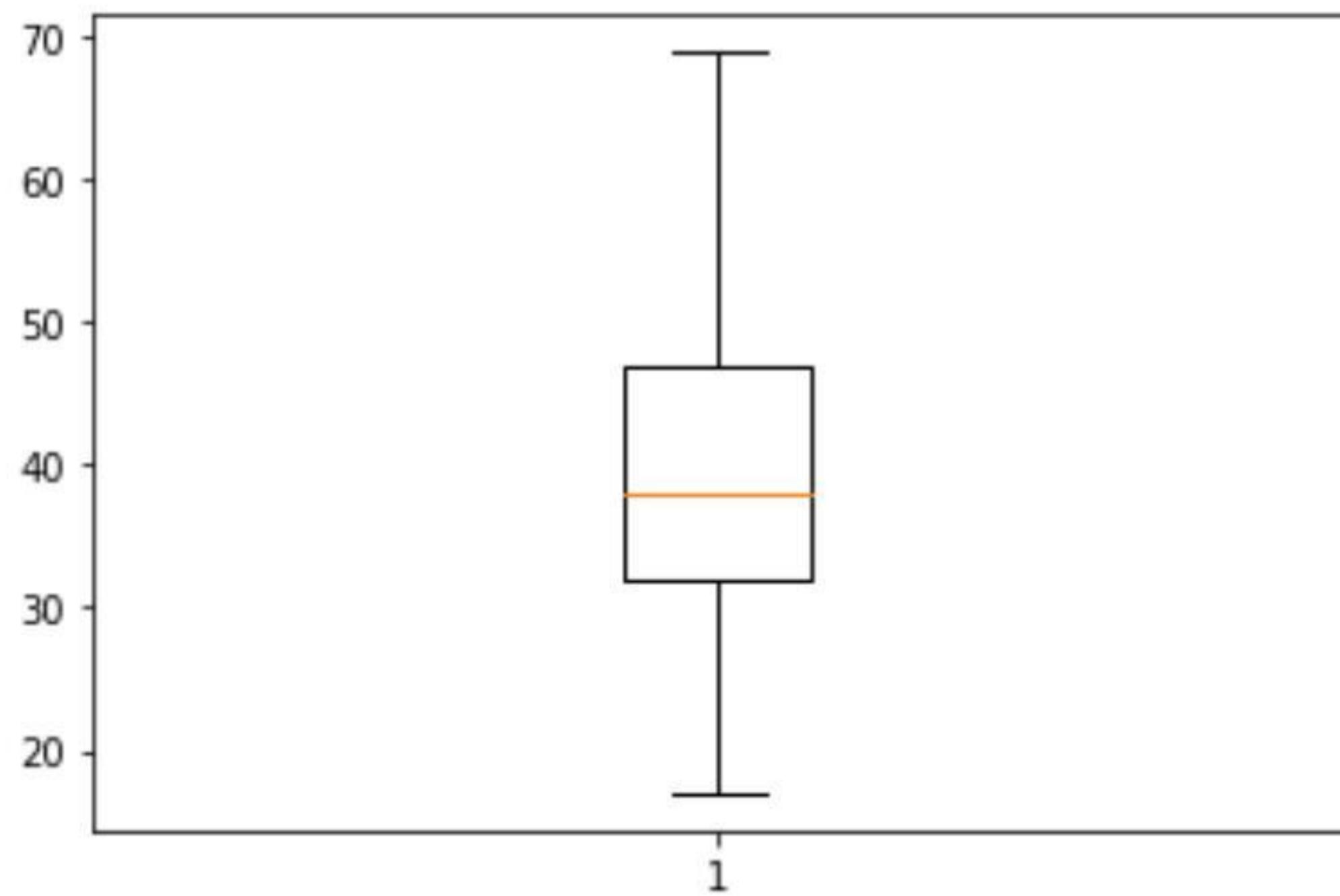
```
Out[70]: (41176, 21)
```

```
In [71]: df_now = df[(df.age < Q3_age + 1.5*(Q3_age-Q1_age))]
```

```
In [72]: df_now.shape
```

```
Out[72]: (40708, 21)
```

```
In [73]: plt.boxplot(df_now.age)
plt.show()
```



```
In [74]: # Dữ Liệu age không còn outlier
```

```
In [75]: df_now.age.mean()
```

```
Out[75]: 39.599390783138446
```

```
In [76]: df.age.mean()
```

```
Out[76]: 40.02380027200311
```

```
In [77]: # Không nhất thiết phải loại bỏ ngoại lệ vì chênh lệch so với mean nhỏ
```

```
In [78]: # Note: Các biến/thuộc tính còn lại làm tương tự như trên
```

```
In [79]: # Gợi ý:
# Xem xét thêm biến ngoài: major
#https://m.wikihow.com/Calculate-Outliers
# Công thức: O_u = Q3 + 3*IQR, O_L = Q1 - 3*IQR
# hoặc tính trung bình trước và sau khi loại bỏ outlier
```

## Gợi ý phần 3

###1. Tạo cột conversion: dựa trên cột y với giá trị 'yes' => 1 và 'no' => 0

```
In [80]: df['conversion'] = df['y'].apply(lambda x: 1 if x == 'yes' else 0)
```

###2. Tính toán tổng conversion và conversion rate

```
In [81]: print('Total conversions: %i out of %i' % (df.conversion.sum(), df.shape[0]))
```

```
Total conversions: 4639 out of 41176
```

```
In [82]: print('Conversion rate: %0.2f%%' % (df.conversion.sum() / df.shape[0] * 100.0))
```

```
Conversion rate: 11.27%
```

###3. Tính toán conversion rate theo từng age. Trực quan hóa kết quả.

```
In [83]: #conversions_by_age = df.groupby(by='age')['conversion'].sum() / df.groupby(by='age')[
```

```
conversions_by_age = df.groupby(by='age')['conversion'].sum() / df.groupby(by='age')['conversion'].count() * 100.0
```

```
In [84]: print('Conversion rate by age:')
```

```
Conversion rate by age:
```

```
In [85]: conversions_by_age.head(10)
```

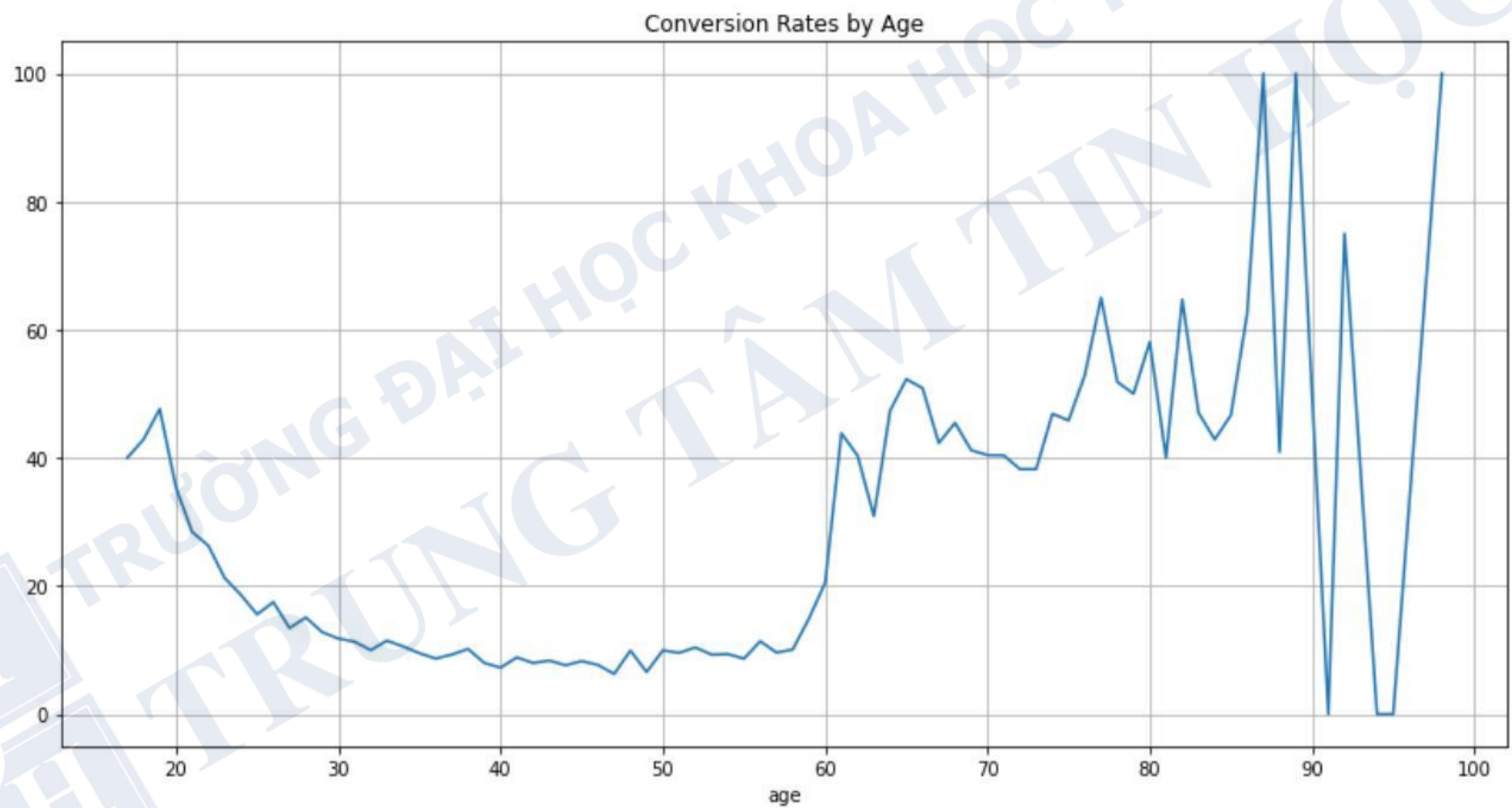
```
Out[85]: age  
17    40.000000  
18    42.857143  
19    47.619048  
20    35.384615  
21    28.431373  
22    26.277372  
23    21.238938  
24    18.614719  
25    15.551839  
26    17.478510  
Name: conversion, dtype: float64
```

```
In [86]: conversions_by_age.tail(10)
```

Out[86]: age

```
85    46.666667  
86    62.500000  
87    100.000000  
88    40.909091  
89    100.000000  
91     0.000000  
92    75.000000  
94     0.000000  
95     0.000000  
98    100.000000  
Name: conversion, dtype: float64
```

```
In [87]: conversions_by_age.plot(grid=True, figsize=(14, 7), title='Conversion Rates by Age')  
plt.show()
```



###4. Tính toán conversion rate theo từng nhóm age (nhóm 18-30, nhóm 30-40, nhóm 40 - 50, nhóm 50-60, nhóm 60-70, nhóm >70). Trực quan hóa kết quả.

```
In [88]: df['age_group'] = df['age'].apply(lambda x: '[18, 30)' \  
                                         if x < 30 else '[30, 40)' if x < 40 \\  
                                         else '[40, 50)' if x < 50 else '[50, 60)' if x < 60 \\  
                                         else '[60, 70)' if x < 70 else '70+'  
)
```

```
In [89]: # conversions_by_age_group = df.groupby(by='age_group')['conversion'].sum() / df
```

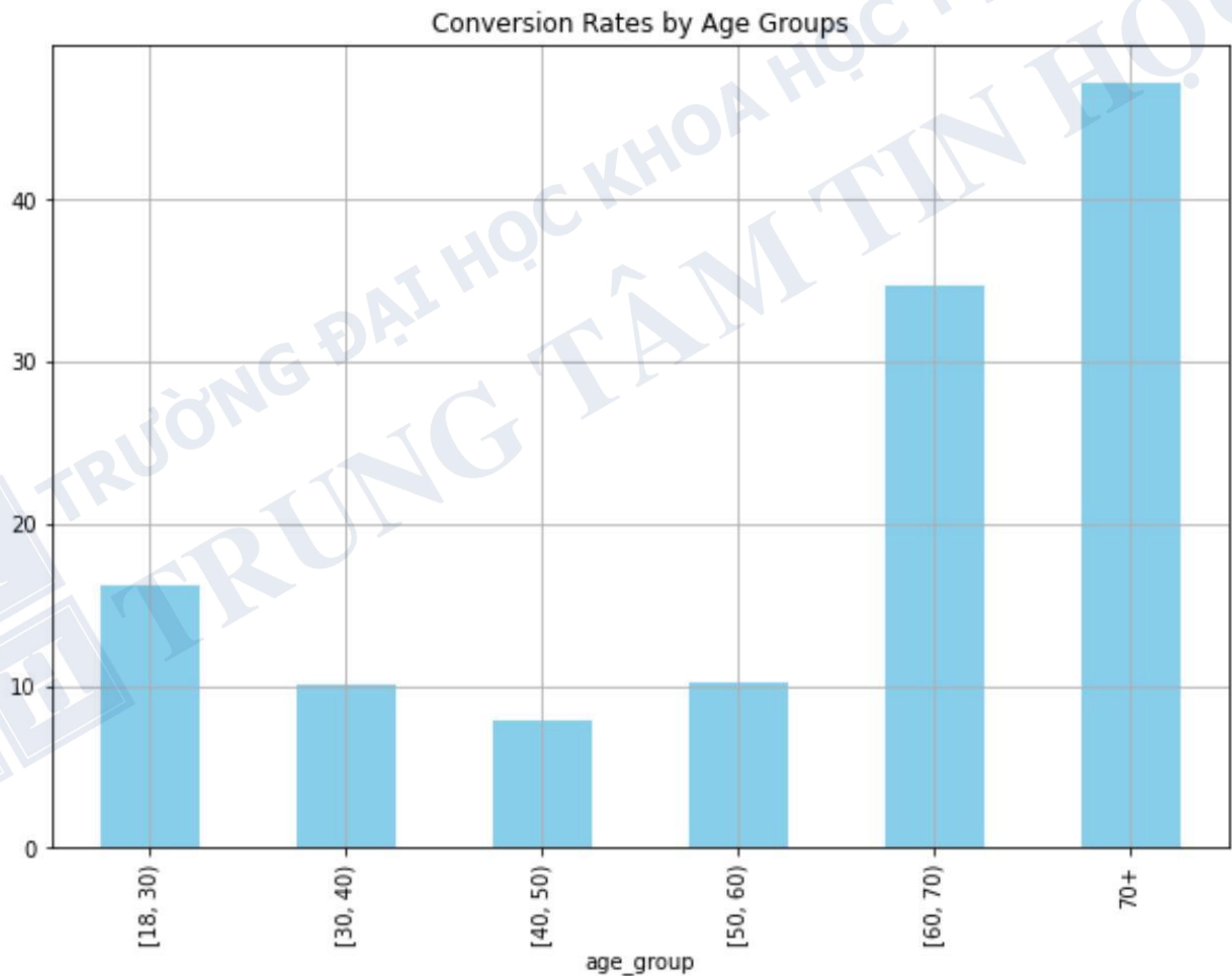
```
conversions_by_age_group = df.groupby(by='age_group')['conversion'].sum() /  
df.groupby(by='age_group')['conversion'].count() * 100.0
```

```
In [90]: conversions_by_age_group
```

```
Out[90]: age_group
```

```
70+      47.222222
[18, 30) 16.269631
[30, 40) 10.128152
[40, 50)  7.915994
[50, 60)  10.158869
[60, 70)  34.668508
Name: conversion, dtype: float64
```

```
In [91]: conversions_by_age_group.loc[
    '[18, 30)', '[30, 40)', '[40, 50)', '[50, 60)', '[60, 70)', '70+']
    .plot(kind='bar', color='skyblue', grid=True, figsize=(10, 7),
          title='Conversion Rates by Age Groups')
plt.show()
```



##5. So sánh giữa conversion và non-conversion cho từng nhóm marital status. Trực quan hóa kết quả.

```
In [92]: df.groupby(['marital', 'y'])['conversion'].count()
```

```
Out[92]: marital    y
           divorced   no      4135
                      yes     476
           married    no     22390
                      yes    2531
           single     no     9944
                      yes    1620
           unknown   no       68
                      yes     12
Name: conversion, dtype: int64
```

```
In [93]: conversions_by_marital_status_df = pd.pivot_table(df, values='y',
                                                       index='marital',
                                                       columns='conversion',
                                                       aggfunc=len)
conversions_by_marital_status_df
```

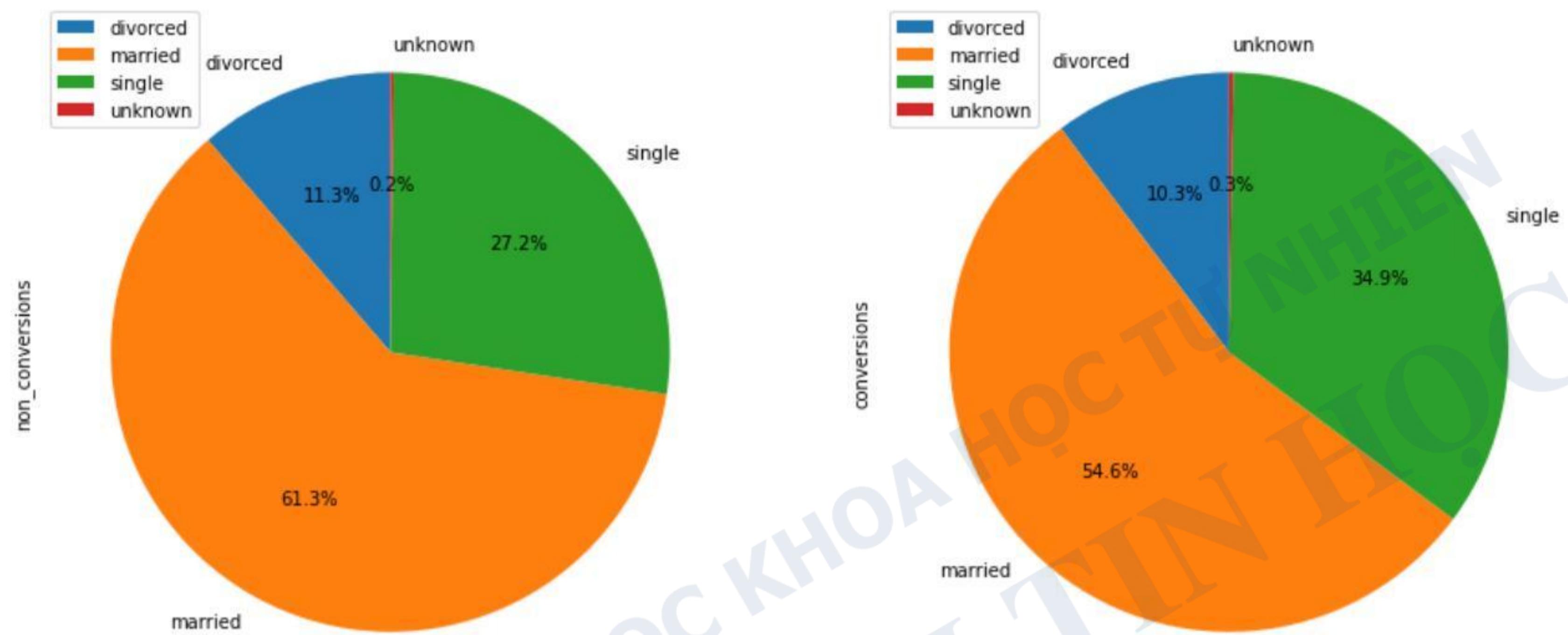
```
Out[93]: conversion      0      1
          marital
          _____
          divorced  4135  476
          married   22390 2531
          single    9944  1620
          unknown    68   12
```

```
In [94]: conversions_by_marital_status_df.columns = ['non_conversions', 'conversions']
```

```
In [95]: conversions_by_marital_status_df
```

```
Out[95]: non_conversions  conversions
          marital
          _____
          divorced            4135      476
          married             22390    2531
          single              9944    1620
          unknown              68     12
```

```
In [96]: conversions_by_marital_status_df.plot(  
    kind='pie',  
    figsize=(15, 7),  
    startangle=90,  
    subplots=True,  
    autopct=lambda x: '%0.1f%%' % x  
)  
  
plt.show()
```



```
In [97]: # Tiếp tục thực hiện việc so sánh giữa conversion và non-conversion theo Education
```

###6. So sánh giữa conversion và non-conversion cho từng nhóm Age Groups & Marital Status.  
Trực quan hóa kết quả.

```
In [98]: age_marital_df = df.groupby(['age_group', 'marital'])['conversion'].sum().unstack()
```

```
age_marital_df = df.groupby(['age_group', 'marital'])  
['conversion'].sum().unstack('marital').fillna(0)
```

```
In [99]: age_marital_df = age_marital_df.divide(  
    df.groupby(  
        by='age_group'  
    )['conversion'].count(),  
    axis=0  
)
```

```
In [100]: age_marital_df
```

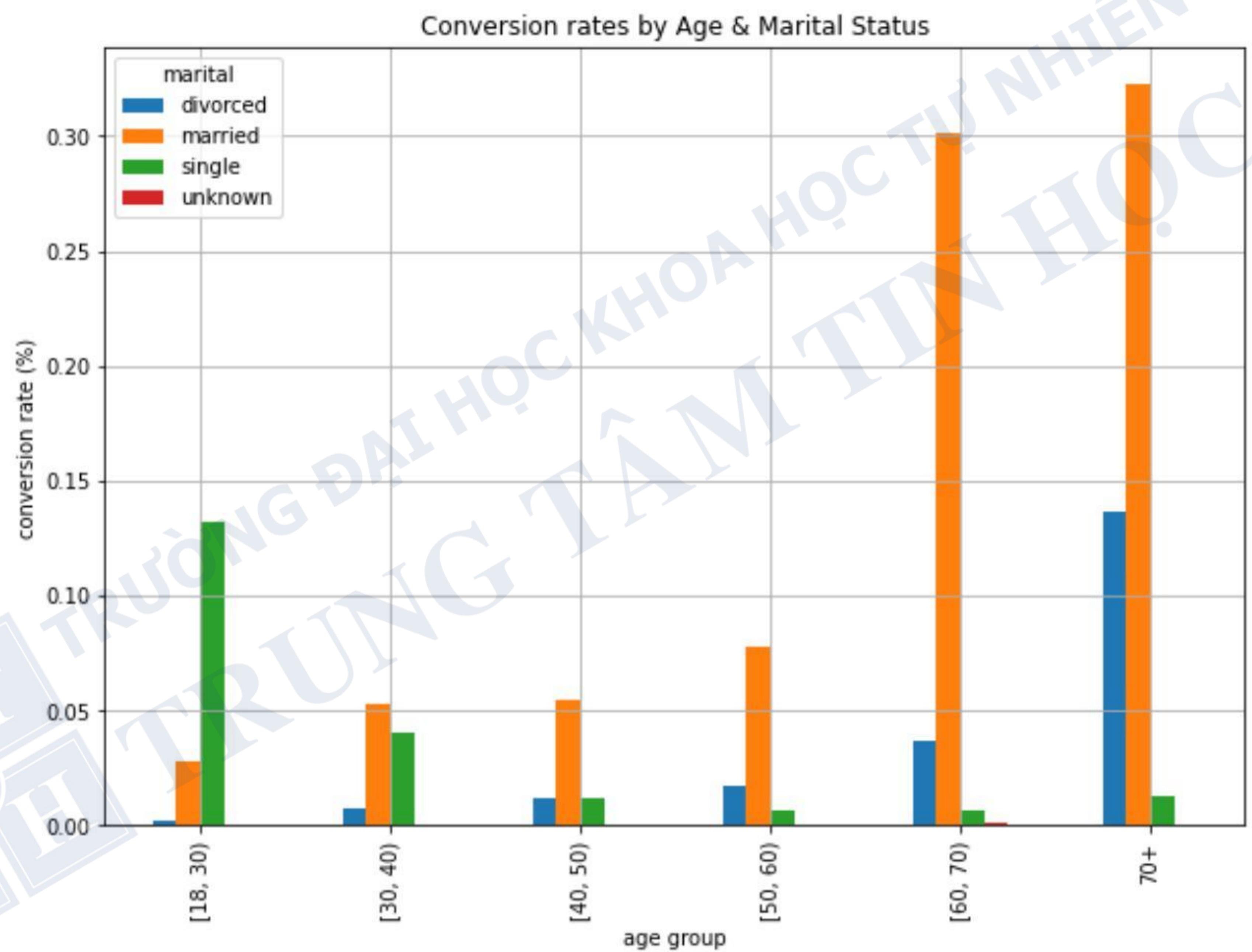
```
Out[100]:   marital  divorced  married  single  unknown
```

age_group	marital	divorced	married	single	unknown
70+	0.136752	0.322650	0.012821	0.000000	
[18, 30)	0.002118	0.027881	0.132522	0.000176	
[30, 40)	0.007559	0.052973	0.040394	0.000354	
[40, 50)	0.011974	0.054547	0.012354	0.000285	
[50, 60)	0.017344	0.077685	0.006413	0.000146	
[60, 70)	0.037293	0.301105	0.006906	0.001381	

```
In [101]: ax = age_marital_df.loc[
    '[18, 30)', '[30, 40)', '[40, 50)', '[50, 60)', '[60, 70)', '70+'
].plot(
    kind='bar',
    grid=True,
    figsize=(10,7)
)

ax.set_title('Conversion rates by Age & Marital Status')
ax.set_xlabel('age group')
ax.set_ylabel('conversion rate (%)')

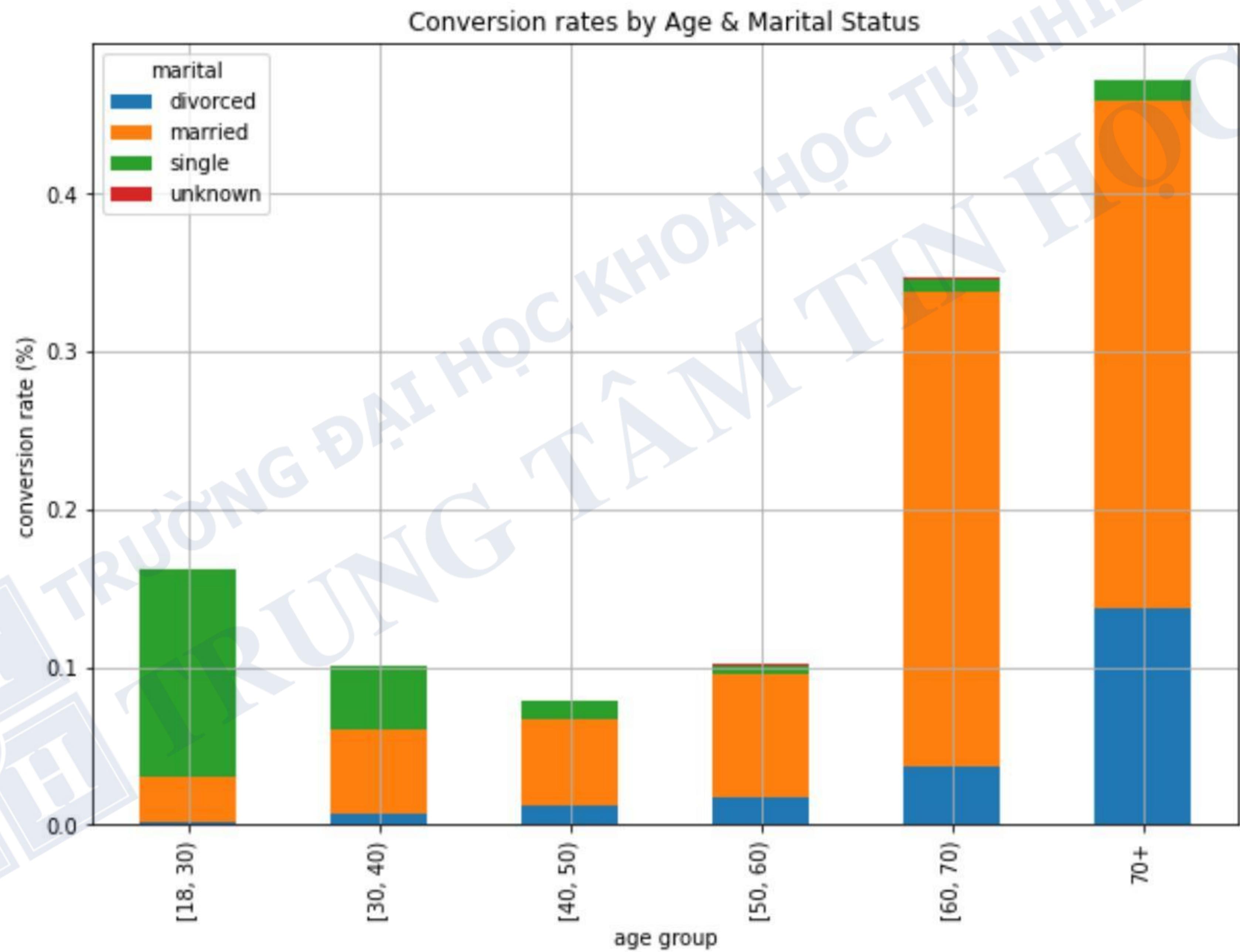
plt.show()
```



```
In [102]: ax = age_marital_df.loc[
    '[18, 30)', '[30, 40)', '[40, 50)', '[50, 60)', '[60, 70)', '70+']
].plot(
    kind='bar',
    stacked=True,
    grid=True,
    figsize=(10,7)
)

ax.set_title('Conversion rates by Age & Marital Status')
ax.set_xlabel('age group')
ax.set_ylabel('conversion rate (%)')

plt.show()
```



In [102]:

