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

df=pd.read_csv("train.csv")

df
```

```
          ID  age         job   marital  education default  balance
housing  \
0      26110   56      admin.   married    unknown      no     1933
no
1      40576   31     unknown   married  secondary      no        3
no
2      15320   27    services   married  secondary      no      891
yes
3      43962   57  management  divorced   tertiary      no     3287
no
4      29842   31  technician   married  secondary      no      119
yes
...      ...  ...         ...       ...        ...     ...      ...
...
31642  36483   29  management    single   tertiary      no        0
yes
31643  40178   53  management  divorced   tertiary      no      380
no
31644  19710   32  management    single   tertiary      no      312
no
31645  38556   57  technician   married  secondary      no      225
yes
31646  14156   55  management  divorced  secondary      no      204
yes

      loan    contact  day month  duration  campaign  pdays  previous
\
0       no  telephone   19   nov        44         2     -1         0

1       no   cellular   20   jul        91         2     -1         0

2       no   cellular   18   jul       240         1     -1         0

3       no   cellular   22   jun       867         1     84         3

4       no   cellular    4   feb       380         1     -1         0

...    ...        ...  ...   ...       ...       ...    ...       ...

31642   no   cellular   12   may       116         2     -1         0
```

```
31643  yes   cellular    5   jun       438         2     -1         0

31644   no   cellular    7   aug        37         3     -1         0

31645   no  telephone   15   may        22         7    337        12

31646   no   cellular   11   jul      1973         2     -1         0


      poutcome subscribed
0      unknown         no
1      unknown         no
2      unknown         no
3      success        yes
4      unknown         no
...        ...        ...
31642  unknown         no
31643  unknown        yes
31644  unknown         no
31645  failure         no
31646  unknown        yes

[31647 rows x 18 columns]

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31647 entries, 0 to 31646
Data columns (total 18 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   ID          31647 non-null  int64
 1   age         31647 non-null  int64
 2   job         31647 non-null  object
 3   marital     31647 non-null  object
 4   education   31647 non-null  object
 5   default     31647 non-null  object
 6   balance     31647 non-null  int64
 7   housing     31647 non-null  object
 8   loan        31647 non-null  object
 9   contact     31647 non-null  object
 10  day         31647 non-null  int64
 11  month       31647 non-null  object
 12  duration    31647 non-null  int64
 13  campaign    31647 non-null  int64
 14  pdays       31647 non-null  int64
 15  previous    31647 non-null  int64
 16  poutcome    31647 non-null  object
 17  subscribed  31647 non-null  object
```

```
dtypes: int64(8), object(10)
memory usage: 4.3+ MB
```

```
df.head()
```

```
      ID  age         job   marital  education default  balance
housing loan  \
0  26110   56      admin.   married    unknown      no     1933
no   no
1  40576   31     unknown   married  secondary      no        3
no   no
2  15320   27    services   married  secondary      no      891
yes   no
3  43962   57  management  divorced   tertiary      no     3287
no   no
4  29842   31  technician   married  secondary      no      119
yes   no

     contact  day month  duration  campaign  pdays  previous poutcome
\
0  telephone   19   nov        44         2     -1         0  unknown

1   cellular   20   jul        91         2     -1         0  unknown

2   cellular   18   jul       240         1     -1         0  unknown

3   cellular   22   jun       867         1     84         3  success

4   cellular    4   feb       380         1     -1         0  unknown


  subscribed
0         no
1         no
2         no
3        yes
4         no
```

```
df.columns
```

```
Index(['ID', 'age', 'job', 'marital', 'education', 'default',
'balance',
       'housing', 'loan', 'contact', 'day', 'month', 'duration',
'campaign',
       'pdays', 'previous', 'poutcome', 'subscribed'],
      dtype='object')
```

```
df.drop_duplicates()
```

```
          ID  age         job   marital  education default  balance
housing  \
```

```
0      26110   56      admin.   married    unknown      no     1933
no
1      40576   31     unknown   married  secondary      no        3
no
2      15320   27    services   married  secondary      no      891
yes
3      43962   57  management  divorced   tertiary      no     3287
no
4      29842   31  technician   married  secondary      no      119
yes
...      ...  ...         ...       ...        ...     ...      ...
...
31642  36483   29  management    single   tertiary      no        0
yes
31643  40178   53  management  divorced   tertiary      no      380
no
31644  19710   32  management    single   tertiary      no      312
no
31645  38556   57  technician   married  secondary      no      225
yes
31646  14156   55  management  divorced  secondary      no      204
yes

      loan    contact  day month  duration  campaign  pdays  previous
\
0       no  telephone   19   nov        44         2     -1         0

1       no   cellular   20   jul        91         2     -1         0

2       no   cellular   18   jul       240         1     -1         0

3       no   cellular   22   jun       867         1     84         3

4       no   cellular    4   feb       380         1     -1         0

...    ...        ...  ...   ...       ...       ...    ...       ...

31642   no   cellular   12   may       116         2     -1         0

31643  yes   cellular    5   jun       438         2     -1         0

31644   no   cellular    7   aug        37         3     -1         0

31645   no  telephone   15   may        22         7    337        12

31646   no   cellular   11   jul      1973         2     -1         0


      poutcome subscribed
```

```
0      unknown         no
1      unknown         no
2      unknown         no
3      success        yes
4      unknown         no
...        ...        ...
31642  unknown         no
31643  unknown        yes
31644  unknown         no
31645  failure         no
31646  unknown        yes

[31647 rows x 18 columns]
```

```
df.describe()
```

```
                 ID           age        balance           day
duration  \
count  31647.000000  31647.000000   31647.000000  31647.000000
31647.000000
mean   22563.972162     40.957247    1363.890258     15.835466
258.113534
std    13075.936990     10.625134    3028.304293      8.337097
257.118973
min        2.000000     18.000000   -8019.000000      1.000000
0.000000
25%    11218.000000     33.000000      73.000000      8.000000
104.000000
50%    22519.000000     39.000000     450.000000     16.000000
180.000000
75%    33879.500000     48.000000    1431.000000     21.000000
318.500000
max    45211.000000     95.000000  102127.000000     31.000000
4918.000000

           campaign         pdays      previous
count  31647.000000  31647.000000  31647.000000
mean       2.765697     39.576042      0.574272
std        3.113830     99.317592      2.422529
min        1.000000     -1.000000      0.000000
25%        1.000000     -1.000000      0.000000
50%        2.000000     -1.000000      0.000000
75%        3.000000     -1.000000      0.000000
max       63.000000    871.000000    275.000000
```

```
df['subscribed'].unique()
```

```
array(['no', 'yes'], dtype=object)
```

```
df['balance'].unique()
```

```
array([1933,    3,  891, ..., 2787, 8741, 2968])
```

```
df['campaign'].unique()
```

```
array([ 2,  1,  3,  4,  7,  5, 33, 12,  8,  9,  6, 24, 17, 11, 20, 25,
19,
       29, 21, 10, 27, 38, 16, 18, 14, 30, 13, 15, 63, 23, 31, 43, 35,
22,
       34, 28, 26, 41, 37, 50, 55, 32, 44, 36, 39])
```

```
df['day'].unique()
```

```
array([19, 20, 18, 22,  4,  2,  3,  8, 15,  5, 28,  6, 14,  7, 24, 13,
9,
       11, 21, 12, 30, 27, 17, 16, 25, 10,  1, 29, 26, 31, 23])
```

```
df['ID'].nunique()
```

```
31647
```

```
df['balance'].nunique()
```

```
6326
```

```
df['campaign'].nunique()
```

```
45
```

```
df['day'].nunique()
```

```
31
```

```
df['duration'].nunique()
```

```
1454
```

```
df['pdays'].nunique()
```

```
509
```

```
df['subscribed'].nunique()
```

```
2
```

```
df
```

```
          ID  age         job   marital  education default  balance
housing  \
0      26110   56      admin.   married    unknown      no     1933
no
1      40576   31     unknown   married  secondary      no        3
```

```
no
2      15320   27    services   married  secondary      no      891
yes
3      43962   57  management  divorced   tertiary      no     3287
no
4      29842   31  technician   married  secondary      no      119
yes
...      ...  ...         ...       ...        ...     ...      ...
...
31642  36483   29  management    single   tertiary      no        0
yes
31643  40178   53  management  divorced   tertiary      no      380
no
31644  19710   32  management    single   tertiary      no      312
no
31645  38556   57  technician   married  secondary      no      225
yes
31646  14156   55  management  divorced  secondary      no      204
yes

      loan    contact  day month  duration  campaign  pdays  previous
\
0       no  telephone   19   nov        44         2     -1         0

1       no   cellular   20   jul        91         2     -1         0

2       no   cellular   18   jul       240         1     -1         0

3       no   cellular   22   jun       867         1     84         3

4       no   cellular    4   feb       380         1     -1         0

...    ...        ...  ...   ...       ...       ...    ...       ...

31642   no   cellular   12   may       116         2     -1         0

31643  yes   cellular    5   jun       438         2     -1         0

31644   no   cellular    7   aug        37         3     -1         0

31645   no  telephone   15   may        22         7    337        12

31646   no   cellular   11   jul      1973         2     -1         0


      poutcome subscribed
0      unknown         no
1      unknown         no
2      unknown         no
```

```
3      success        yes
4      unknown         no
...        ...        ...
31642  unknown         no
31643  unknown        yes
31644  unknown         no
31645  failure         no
31646  unknown        yes

[31647 rows x 18 columns]
```

```
df['job'].unique()
```

```
array(['admin.', 'unknown', 'services', 'management', 'technician',
       'retired', 'blue-collar', 'housemaid', 'self-employed',
'student',
       'entrepreneur', 'unemployed'], dtype=object)
```

```
df['marital'].unique()
```

```
array(['married', 'divorced', 'single'], dtype=object)
```

```
df['education'].unique()
```

```
array(['unknown', 'secondary', 'tertiary', 'primary'], dtype=object)
```

```
df['contact'].unique()
```

```
array(['telephone', 'cellular', 'unknown'], dtype=object)
```

```
df.dtypes
```

```
ID             int64
age            int64
job           object
marital       object
education     object
default       object
balance        int64
housing       object
loan          object
contact       object
day            int64
month         object
duration       int64
campaign       int64
pdays          int64
previous       int64
poutcome      object
subscribed    object
dtype: object
```

```
table = pd.crosstab(df['job'], df['subscribed'])
print(table)
```

```
subscribed       no  yes
job
admin.         3179  452
blue-collar    6353  489
entrepreneur    923   85
housemaid       795   79
management     5716  923
retired        1212  362
self-employed   983  140
services       2649  254
student         453  182
technician     4713  594
unemployed      776  129
unknown         180   26
```

```
table = pd.crosstab(df['marital'], df['subscribed'])
print(table)
```

```
subscribed     no   yes
marital
divorced     3185   445
married     17176  1919
single       7571  1351
```

```
table = pd.crosstab(df['education'], df['subscribed'])
print(table)
```

```
subscribed     no   yes
education
primary      4381   427
secondary   14527  1697
tertiary     7886  1415
unknown      1138   176
```

```
table = pd.crosstab(df['default'], df['subscribed'])
print(table)
```

```
subscribed     no   yes
default
no          27388  3674
yes           544    41
```

```
table = pd.crosstab(df['housing'], df['subscribed'])
print(table)
```

```
subscribed     no   yes
housing
no          11698  2365
yes         16234  1350
```

```
table = pd.crosstab(df['loan'], df['subscribed'])
print(table)
```

```
subscribed     no   yes
loan
no          23132  3384
yes          4800   331
```

```
table = pd.crosstab(df['contact'], df['subscribed'])
print(table)
```

```
subscribed     no   yes
contact
cellular    17352  3071
telephone    1779   268
unknown      8801   376
```

```
table = pd.crosstab(df['month'], df['subscribed'])
print(table)
```

```
subscribed    no  yes
month
apr         1671  384
aug         3813  520
dec           85   72
feb         1522  305
jan          880   97
jul         4403  441
jun         3355  383
mar          168  174
may         9020  649
nov         2508  275
oct          288  224
sep          219  191
```

```
table = pd.crosstab(df['poutcome'], df['subscribed'])
print(table)
```

```
subscribed     no   yes
poutcome
failure      2931   431
other        1071   217
success       374   694
unknown     23556  2373
```

```
df.groupby('balance').mean().head(10)
```

```
<ipython-input-41-e5dda727ac69>:1: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future
version, numeric_only will default to False. Either specify
numeric_only or select only columns which should be valid for the
function.
  df.groupby('balance').mean().head(10)
```

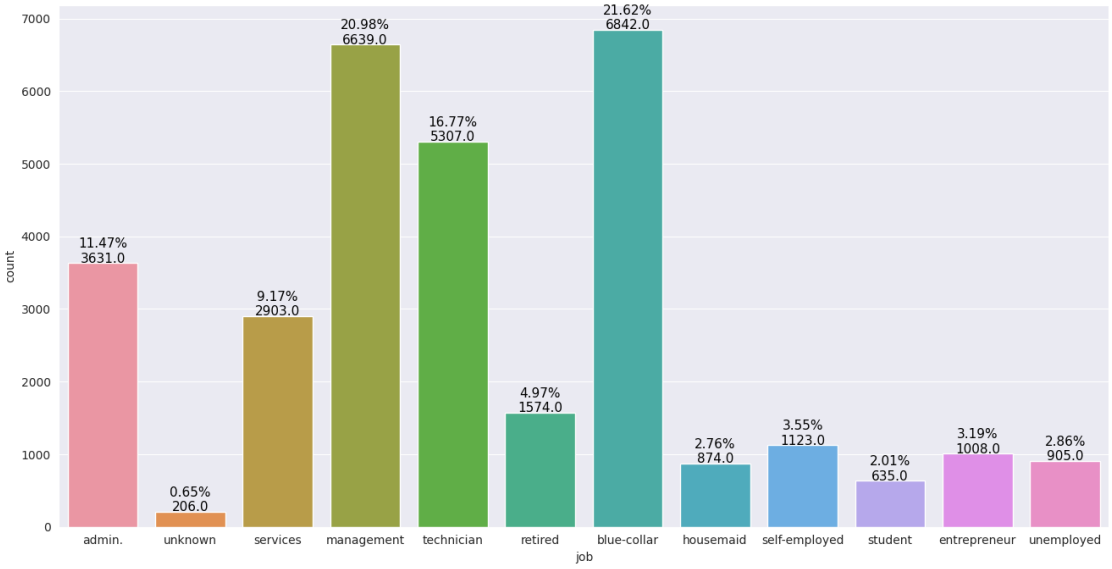
balance	ID	age	day	duration	campaign	pdays	previous
-8019	12910.0	26.0	7.0	299.0	3.0	-1.0	0.0
-6847	15683.0	49.0	21.0	206.0	1.0	-1.0	0.0
-4057	38737.0	60.0	18.0	242.0	6.0	-1.0	0.0
-3372	7414.0	43.0	29.0	386.0	2.0	-1.0	0.0
-3058	32714.0	39.0	17.0	882.0	3.0	-1.0	0.0
-2712	31510.0	52.0	2.0	253.0	1.0	-1.0	0.0
-2604	25120.0	49.0	18.0	142.0	1.0	-1.0	0.0
-2282	14435.0	51.0	14.0	301.0	6.0	-1.0	0.0
-2122	25241.0	43.0	18.0	141.0	3.0	-1.0	0.0
-2082	17160.0	51.0	28.0	123.0	6.0	-1.0	0.0

```
grouped = df.groupby('marital')

grouped

<pandas.core.groupby.generic.DataFrameGroupBy object at
0x7f9939ea1a80>

grouped_mean = grouped_marital.mean()

<ipython-input-48-9652e5ae1168>:1: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future
version, numeric_only will default to False. Either specify
numeric_only or select only columns which should be valid for the
function.
  grouped_mean = grouped_marital.mean()

grouped_sum = grouped['balance'].sum()
grouped_count = grouped['poutcome'].count()

print("Mean balance:")
print(grouped_mean)

print("\nTotal balance:")
print(grouped_sum)

print("\n Poutcome:")
print(grouped_count)

Mean balance:
                    ID        age      balance        day    duration
\
marital
```

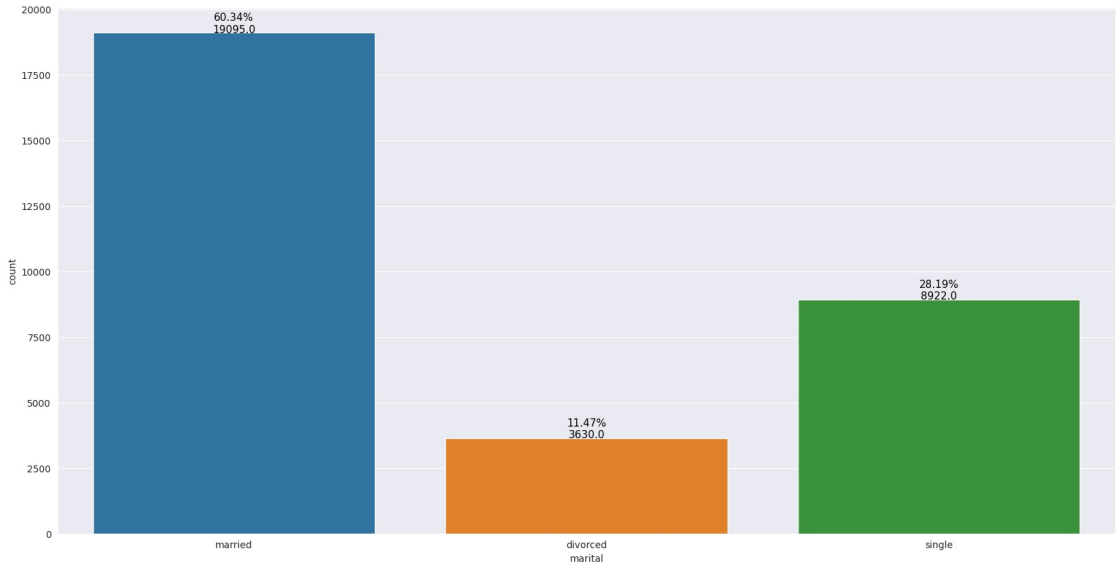
```
divorced  21759.828375  45.945455  1212.203030  15.899449  263.644628

married   21732.145954  43.401885  1410.935847  15.875622  253.023828

single    24671.432751  33.695696  1324.917956  15.723492  266.756221
```

```
          campaign      pdays  previous
marital
divorced  2.578788  40.983471  0.562259
married   2.866195  37.067976  0.547840
single    2.626653  44.371217  0.635732

Total balance:
marital
divorced     4400297
married     26941820
single      11820918
Name: balance, dtype: int64

 Poutcome:
marital
divorced     3630
married     19095
single       8922
Name: poutcome, dtype: int64

df.dtypes

ID             int64
age            int64
job           object
marital       object
education     object
default       object
balance        int64
housing       object
loan          object
contact       object
day            int64
month         object
duration       int64
campaign       int64
pdays          int64
previous       int64
poutcome      object
subscribed    object
dtype: object
```

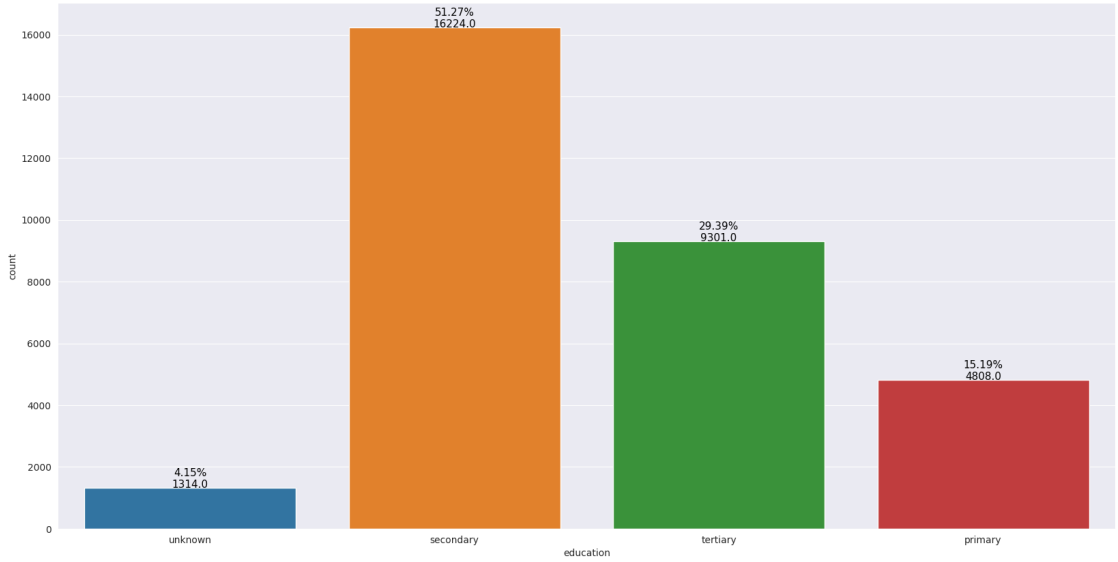
```
plt.figure(figsize=(16, 8))
ax=sns.countplot(df , x="job")
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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




```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="marital")
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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

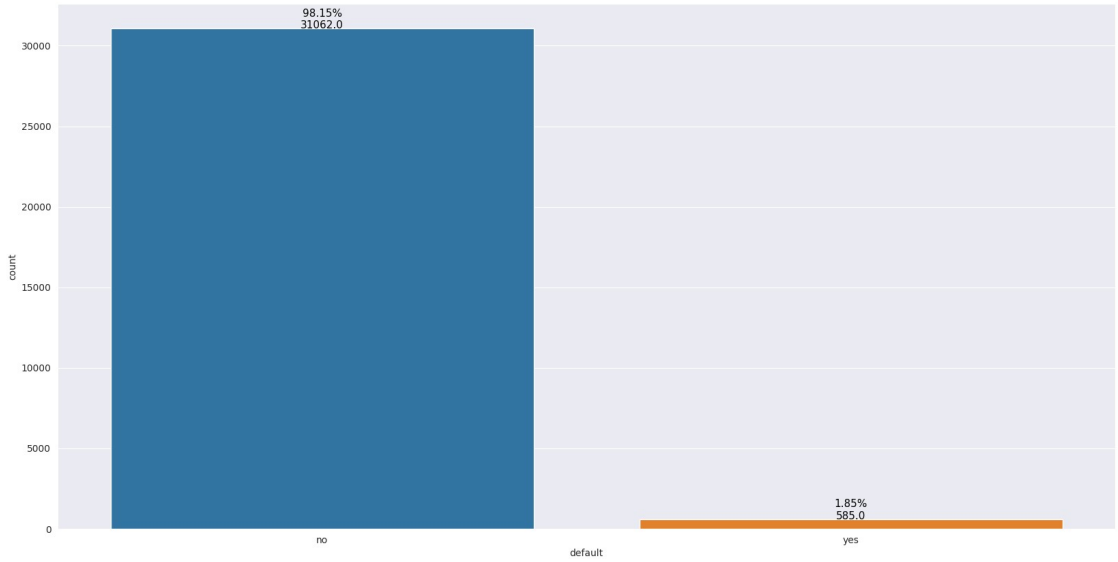



```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="education")
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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

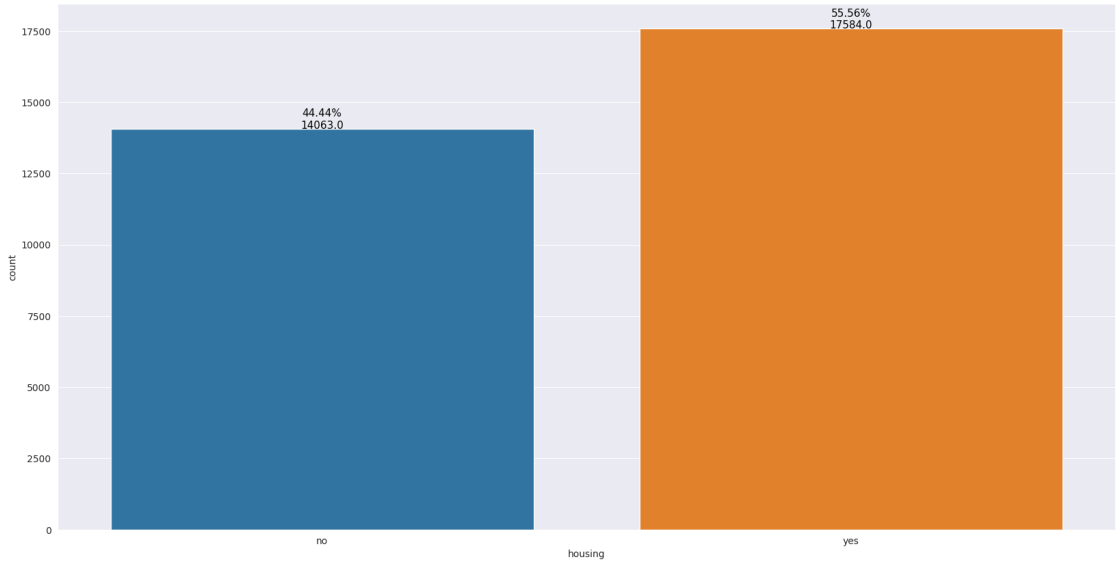



```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="default")
for p in ax.patches:
```

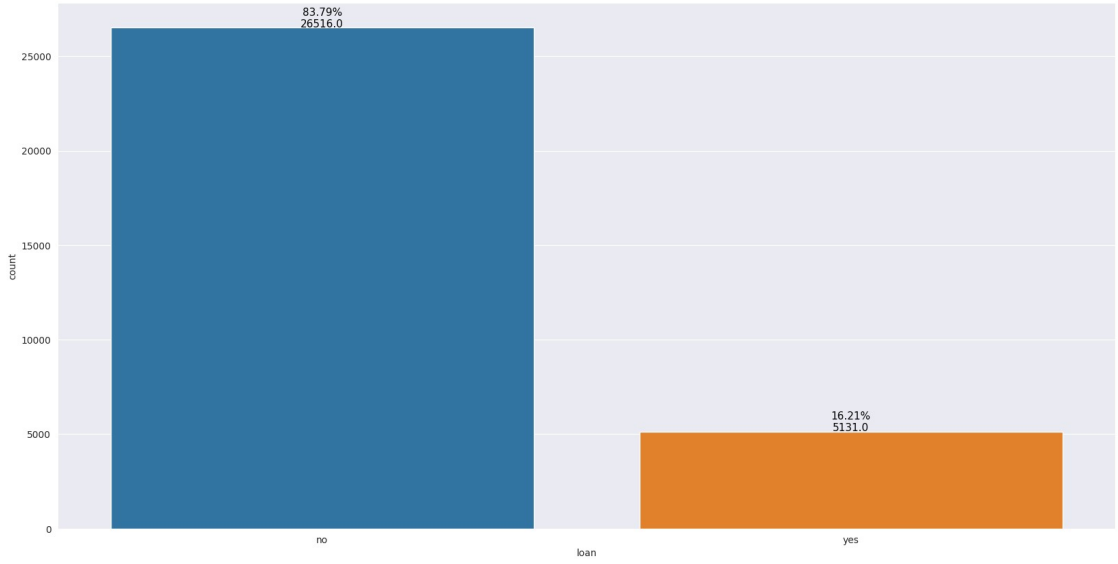
```
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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




```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="housing")
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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

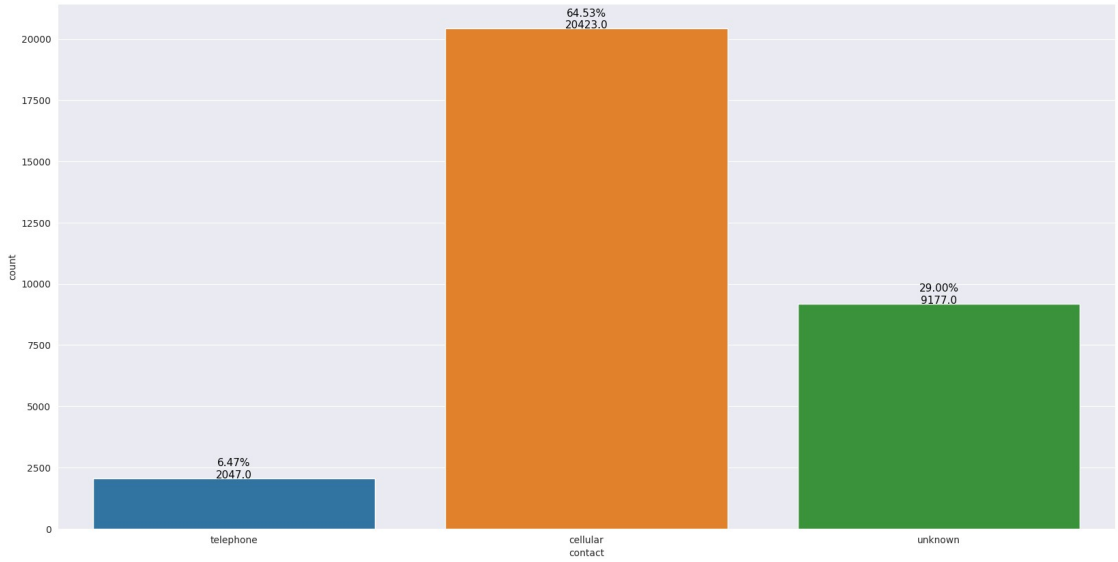



```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="loan")
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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

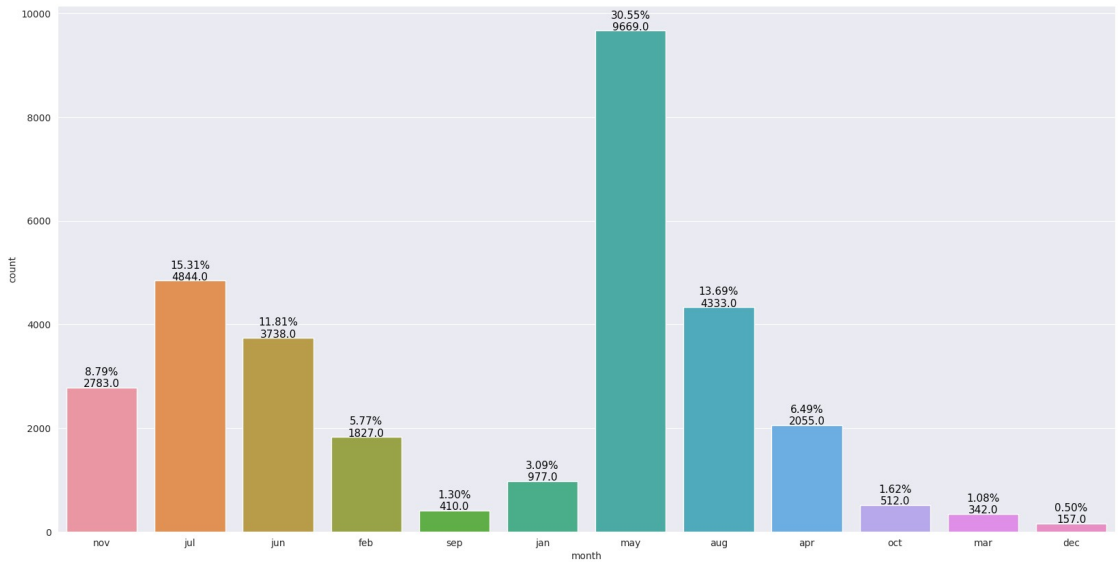



```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="contact")
for p in ax.patches:
```

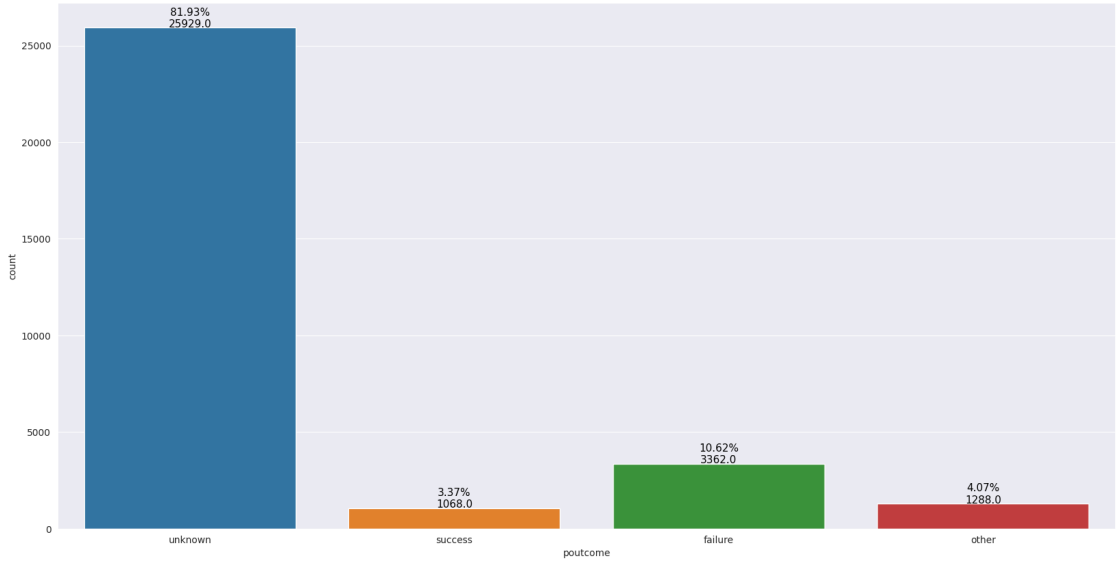
```
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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




```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="month")
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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

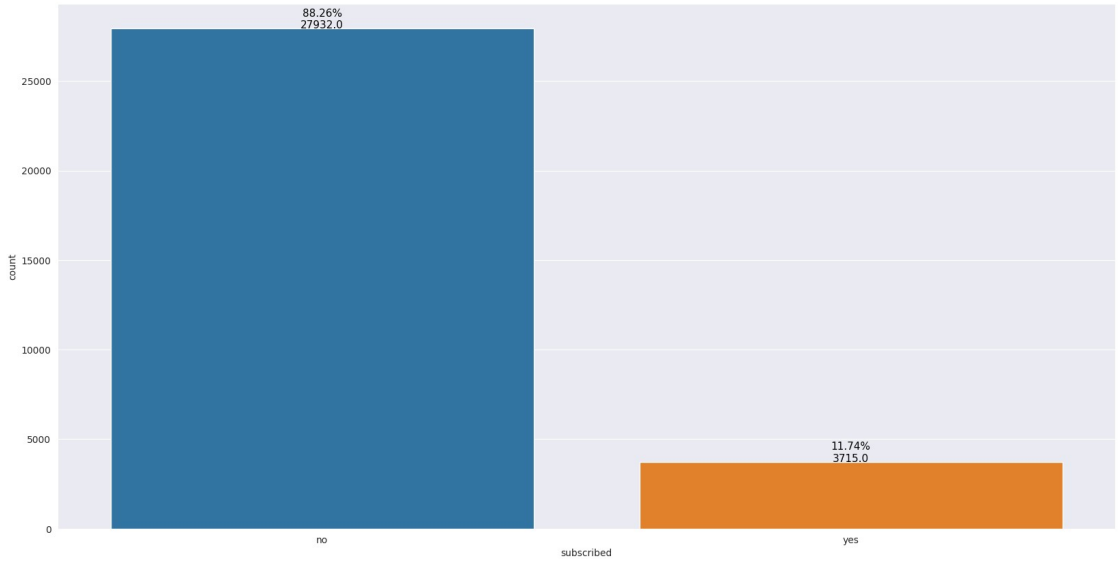



```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="poutcome")
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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




```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="subscribed")
for p in ax.patches:
```

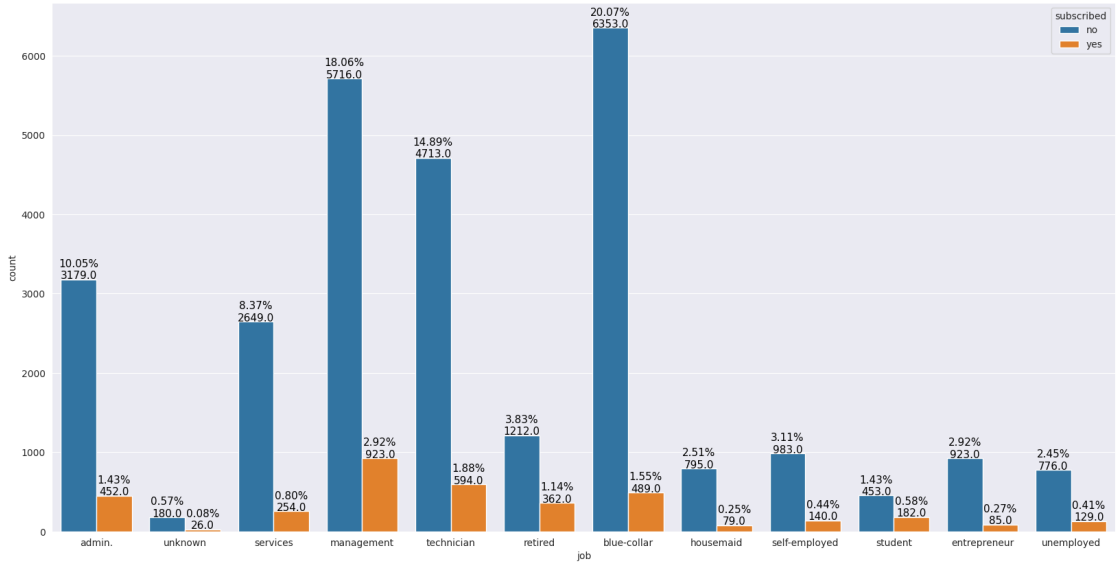
```
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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



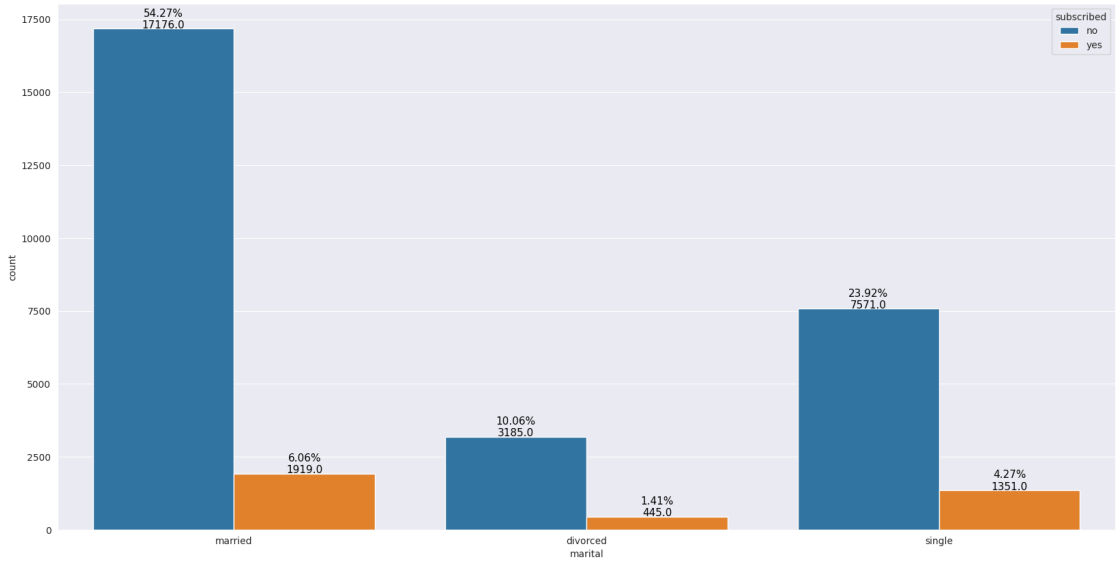

```
df.dtypes
```

```
ID             int64
age            int64
job           object
marital       object
education     object
default       object
balance        int64
housing       object
loan          object
contact       object
day            int64
month         object
duration       int64
campaign       int64
pdays          int64
previous       int64
poutcome      object
subscribed    object
dtype: object
```

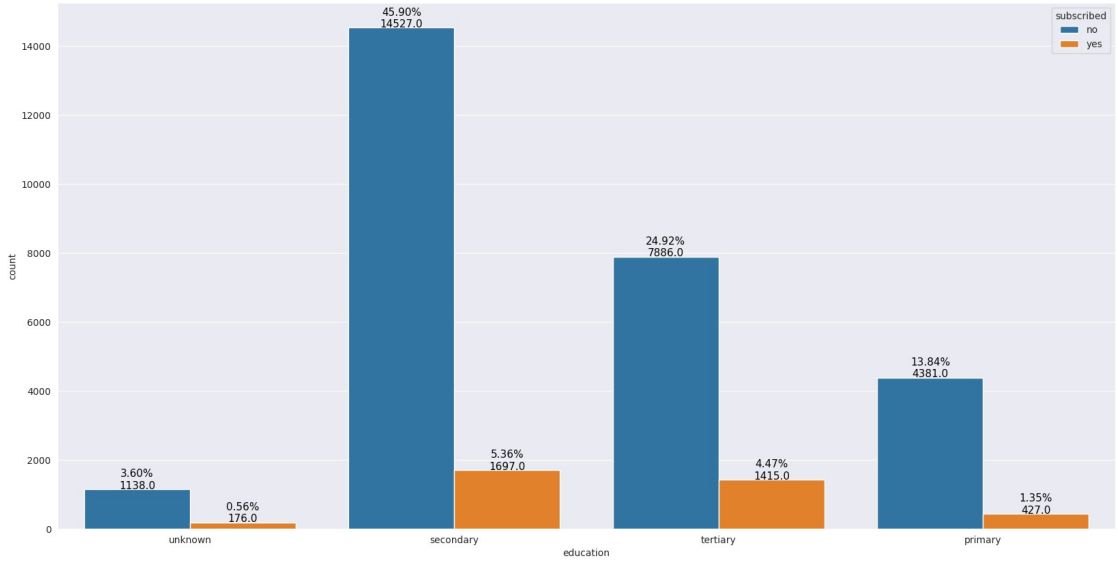
```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="job" , hue="subscribed")
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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




```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="marital" , hue="subscribed")
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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

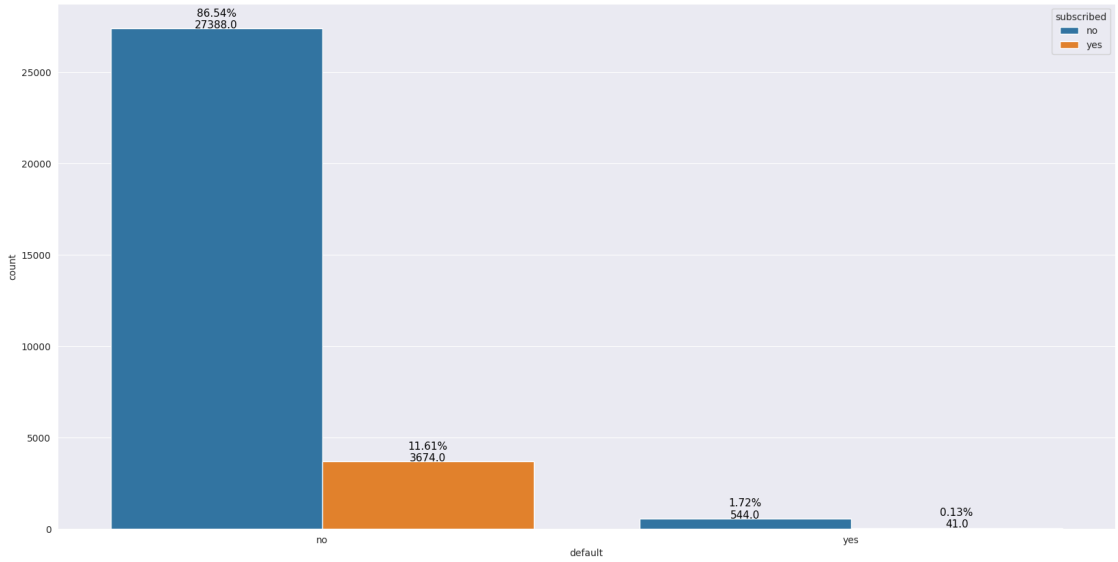



```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="education" , hue="subscribed")
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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

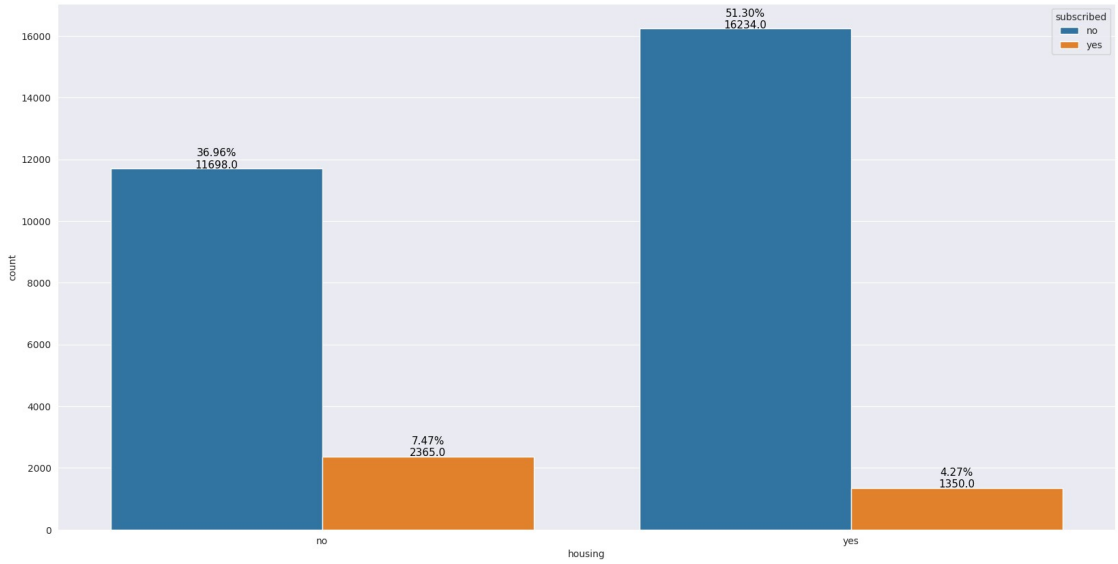



```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="default" , hue="subscribed")
for p in ax.patches:
```

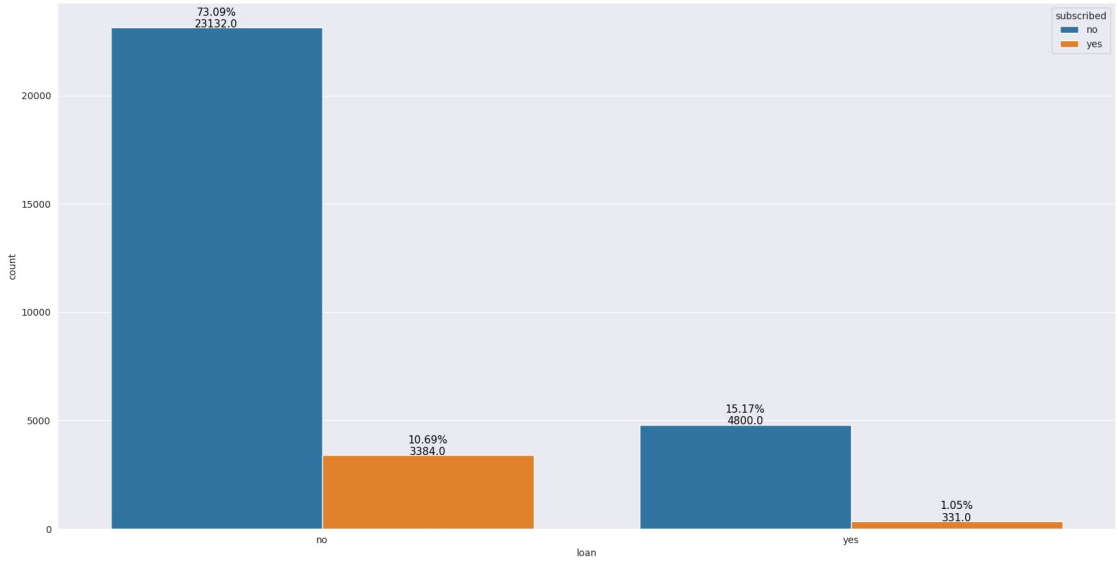
```
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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




```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="housing" , hue="subscribed")
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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

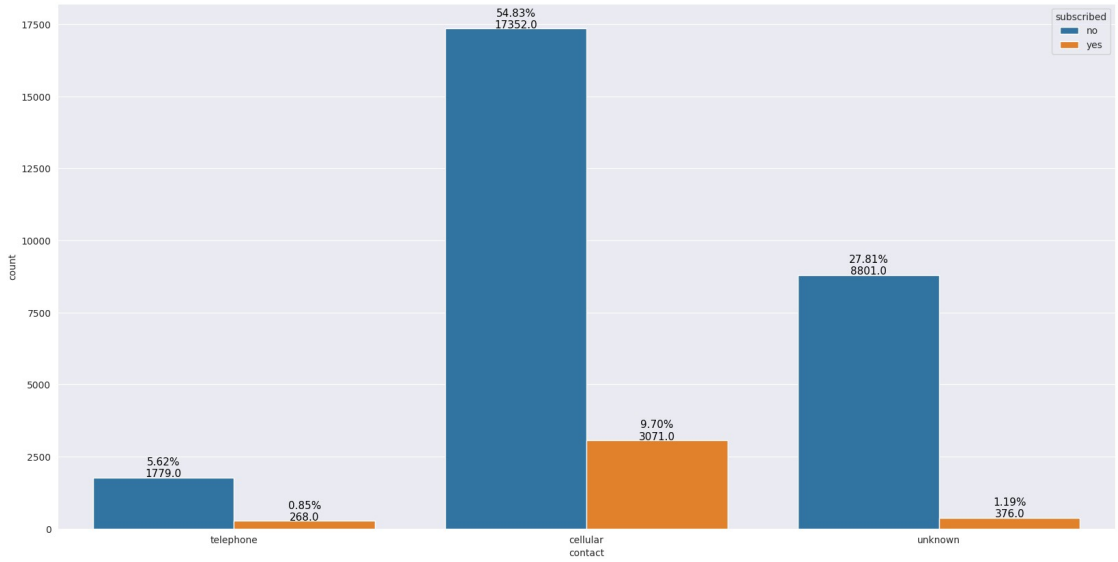



```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="loan" , hue="subscribed")
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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

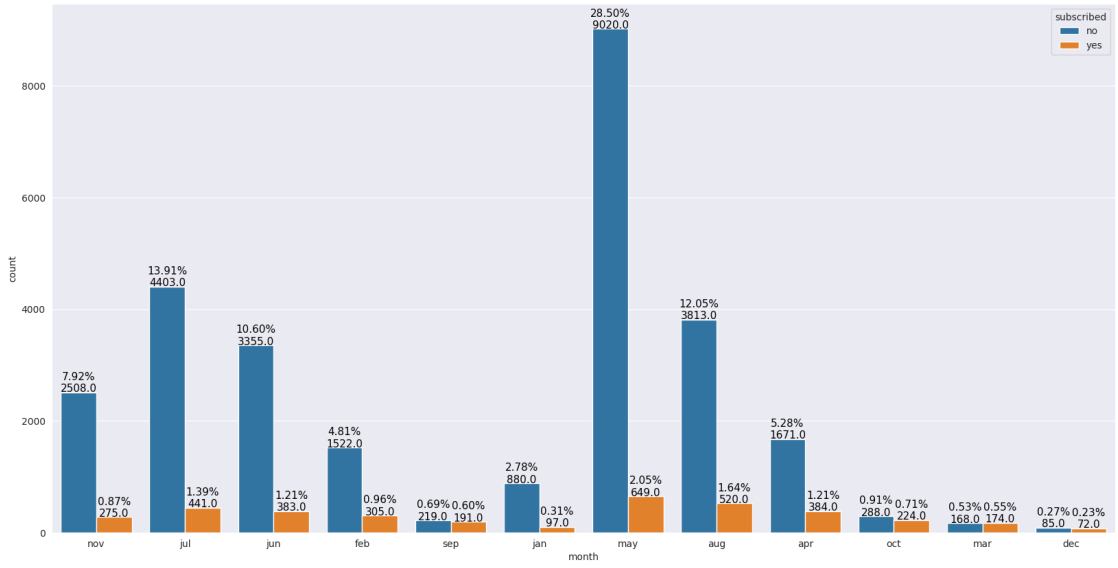



```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="contact" , hue="subscribed")
for p in ax.patches:
```

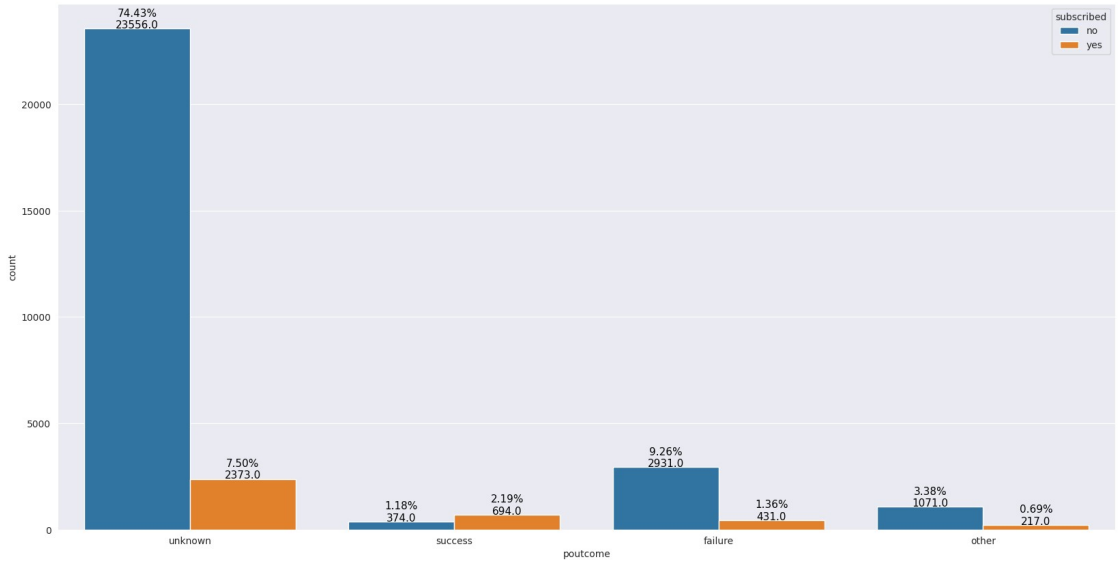
```
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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




```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="month" , hue="subscribed")
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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



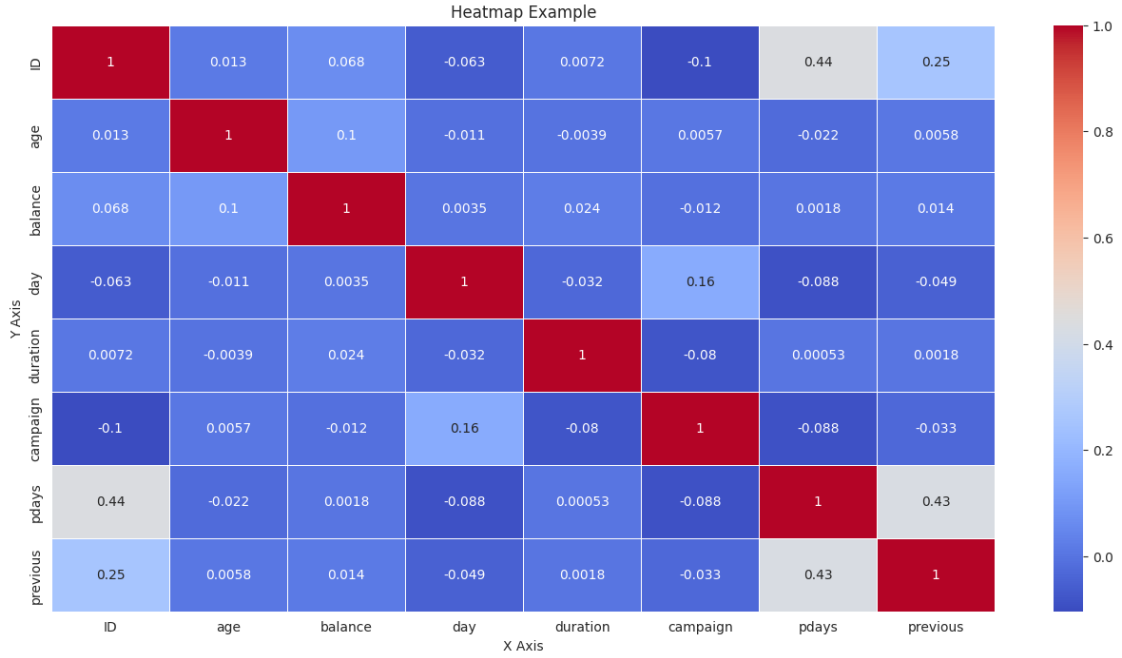

```
plt.figure(figsize=(20, 10))
ax=sns.countplot(df , x="poutcome" , hue="subscribed")
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height / df.shape[0]:.2%}\n{height}',
                xy=(p.get_x() + p.get_width() / 2., height),
                ha='center', va='center', fontsize=11, color='black',
xytext=(0, 10),
                textcoords='offset points')

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




```
corr_matrix=df.corr()
```

```
<ipython-input-86-a4bfebfd3231>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  corr_matrix=df.corr()
```

```
plt.figure(figsize=(16, 8))
sns.heatmap(corr_matrix, cmap='coolwarm', annot=True, linewidths=0.5)

# set plot title and axis labels
plt.title('Heatmap Example')
plt.xlabel('X Axis')
plt.ylabel('Y Axis')

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




```
df.dtypes
```

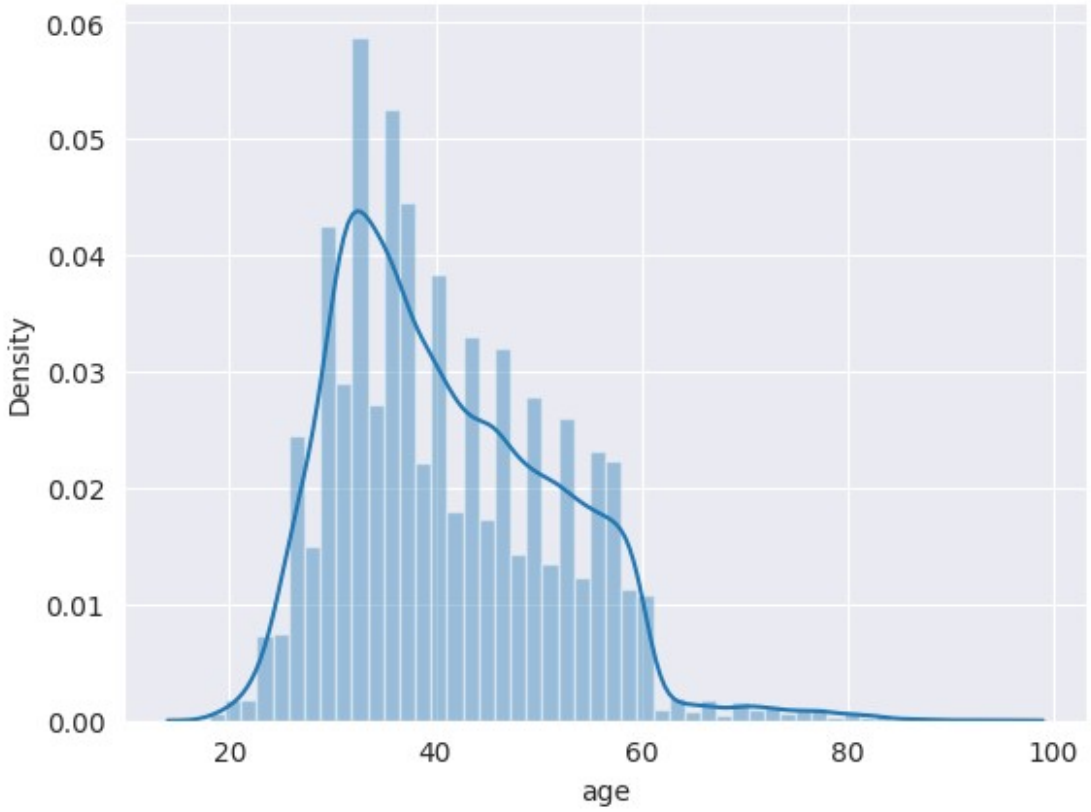
```
ID             int64
age            int64
job           object
marital       object
education     object
default       object
balance        int64
housing       object
loan          object
contact       object
day            int64
```

```
month         object
duration       int64
campaign       int64
pdays          int64
previous       int64
poutcome      object
subscribed    object
dtype: object
```

```
  sns.distplot(df["age"])
```

```
<ipython-input-92-eef84e7ff8f0>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["age"])
```

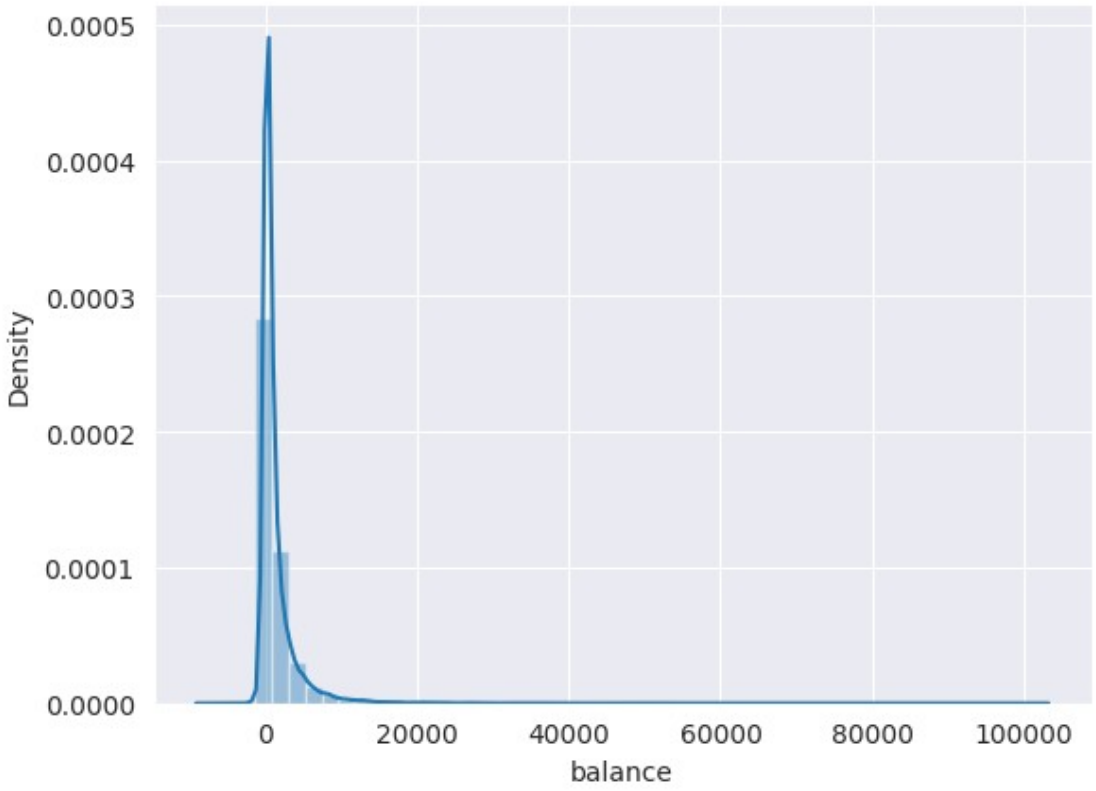
```
<Axes: xlabel='age', ylabel='Density'>
```




```
  sns.distplot(df["balance"])
```

```
<ipython-input-94-adebfed082c1>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["balance"])
```

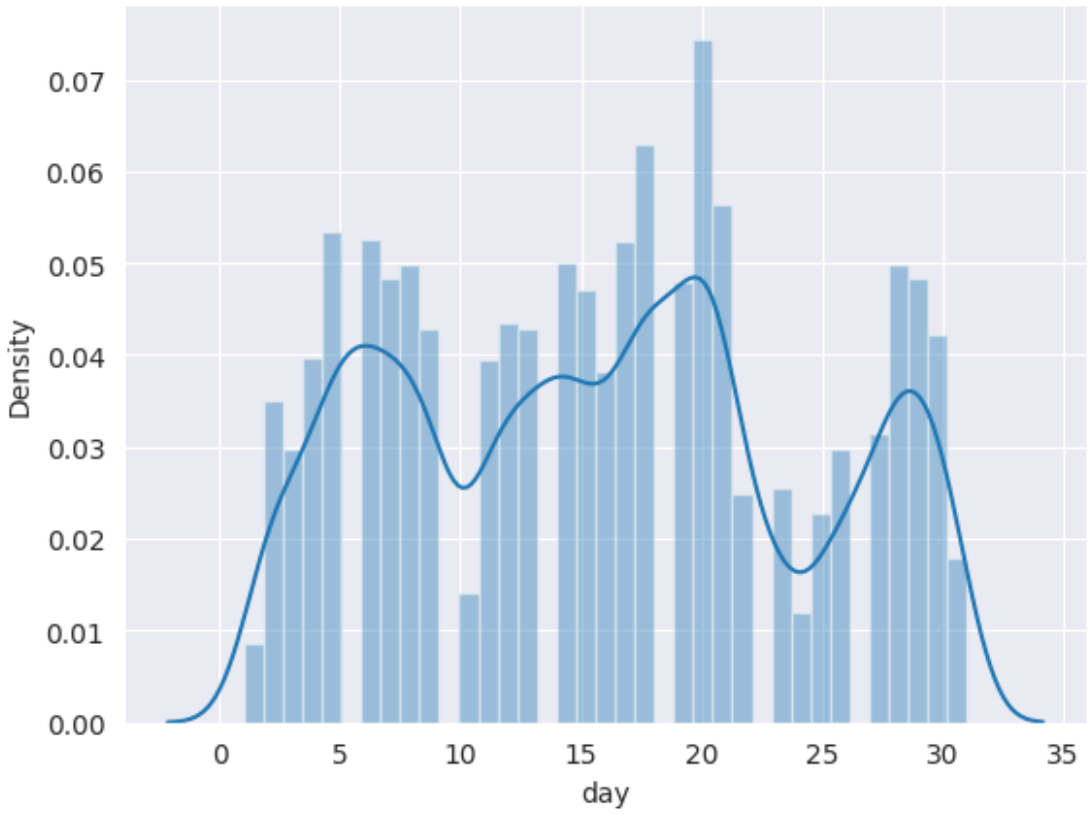
```
<Axes: xlabel='balance', ylabel='Density'>
```




```
  sns.distplot(df["day"])
```

```
<ipython-input-95-d8169acdb438>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["day"])
```

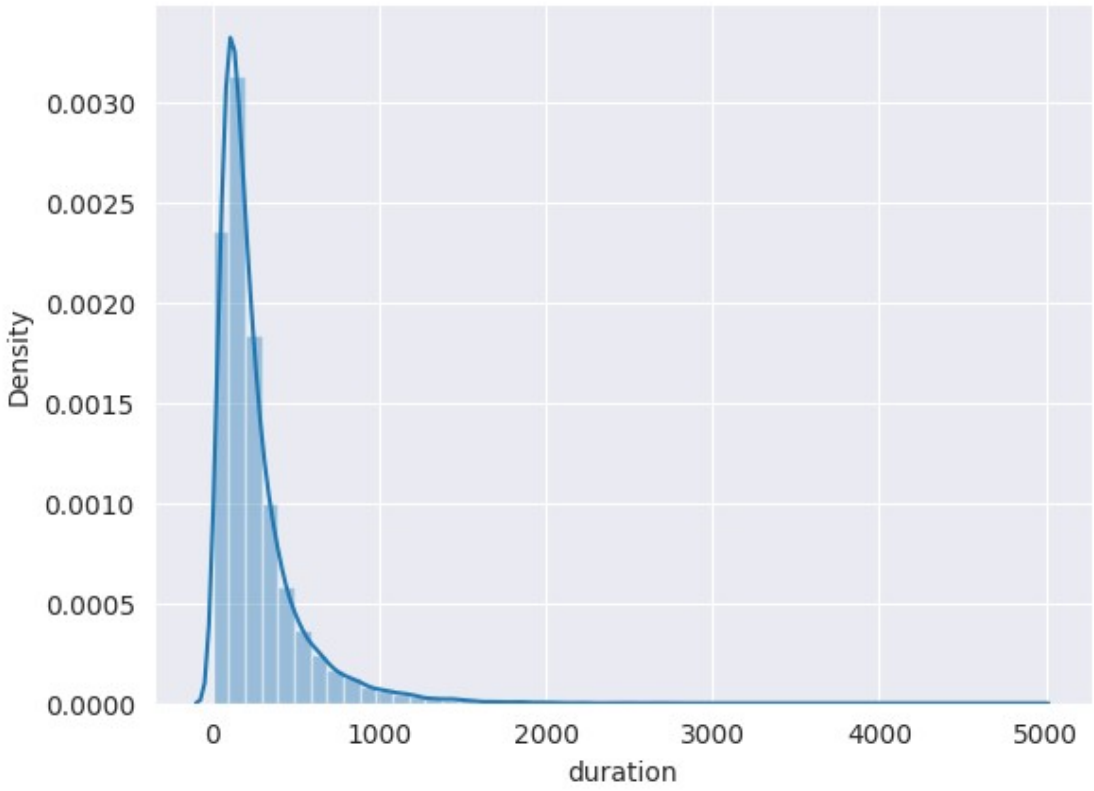
```
<Axes: xlabel='day', ylabel='Density'>
```




```
  sns.distplot(df["duration"])
```

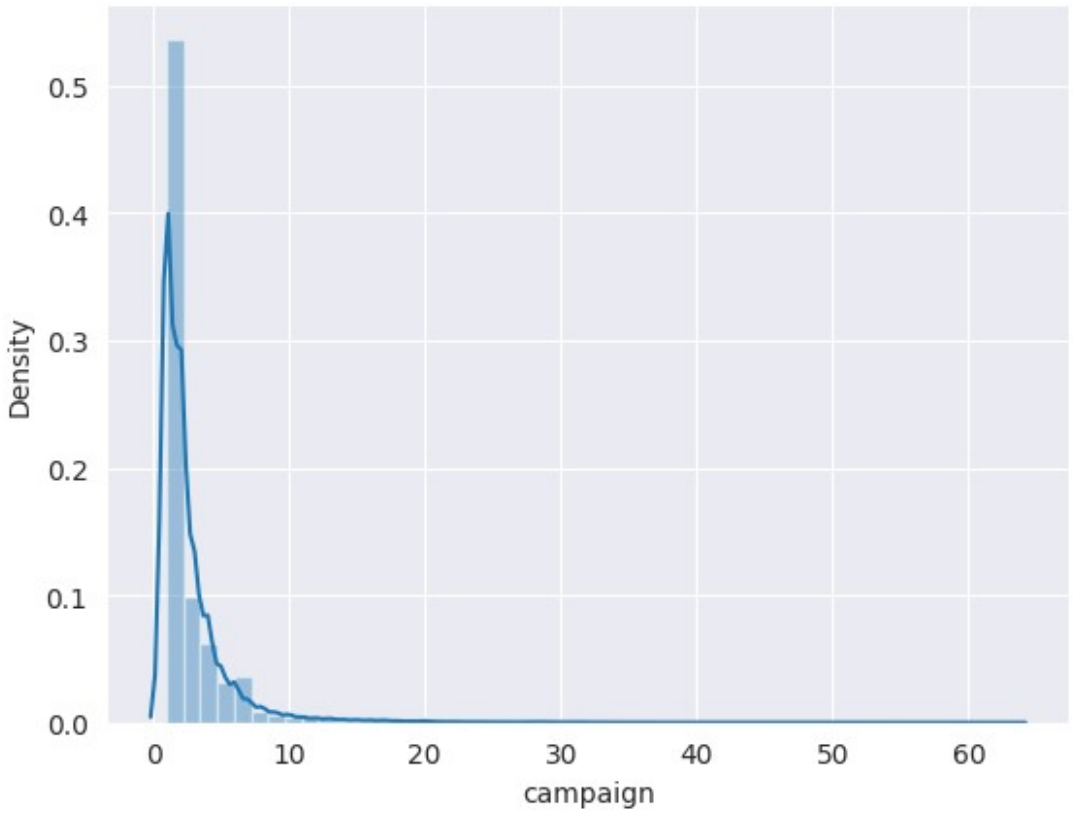
```
<ipython-input-96-07d4faf592e6>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["duration"])
```

```
<Axes: xlabel='duration', ylabel='Density'>
```




```
  sns.distplot(df["campaign"])
```

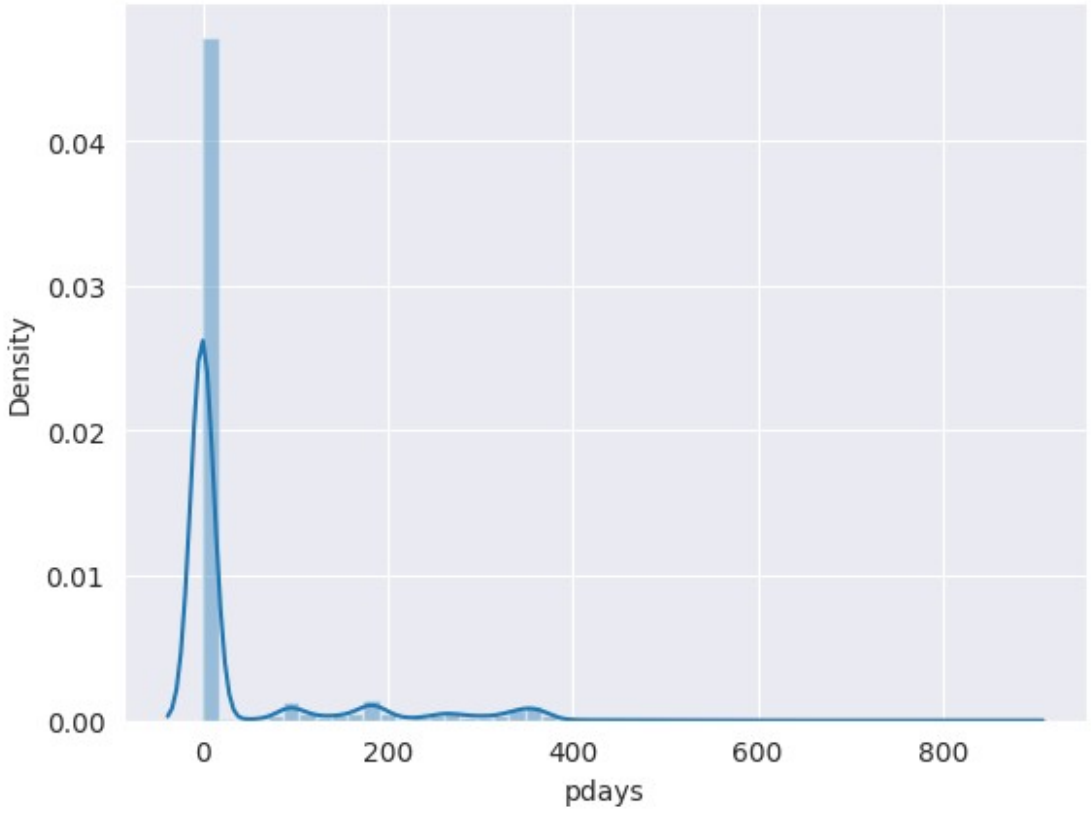
```
<ipython-input-97-1570079ca3f4>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["campaign"])

<Axes: xlabel='campaign', ylabel='Density'>
```




```
  sns.distplot(df["pdays"])
```

```
<ipython-input-98-973f1250fe67>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["pdays"])
```

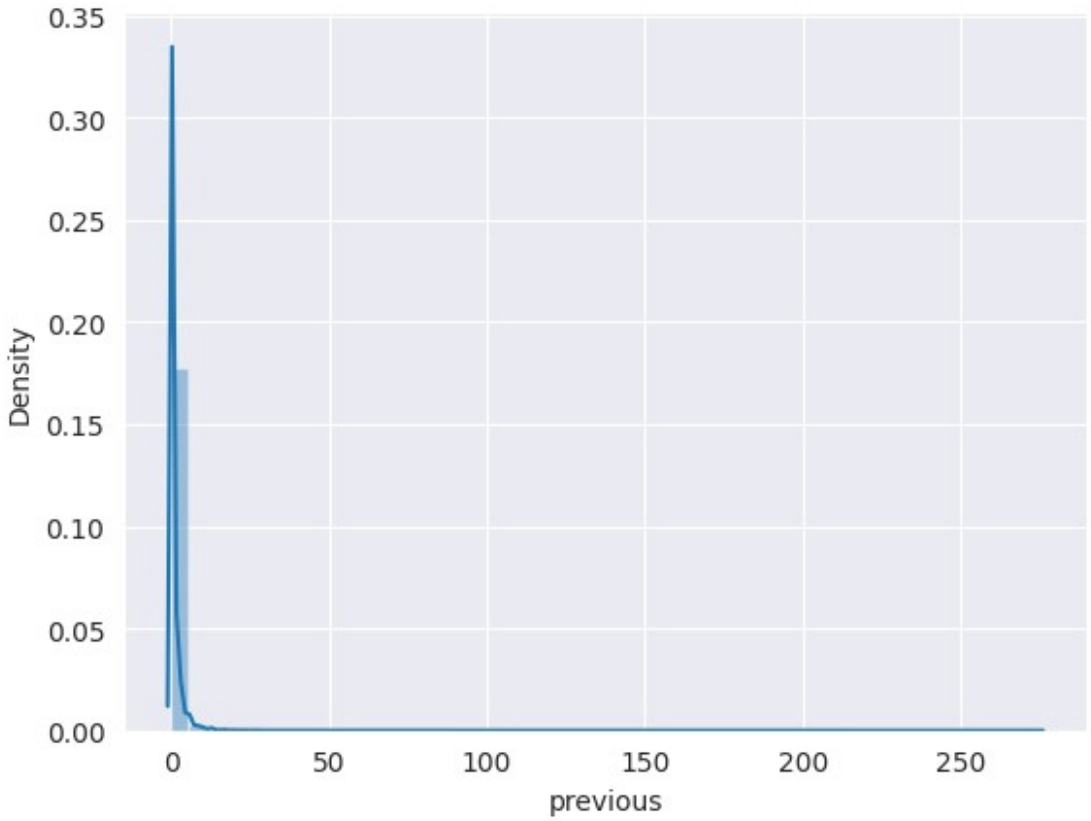
```
<Axes: xlabel='pdays', ylabel='Density'>
```




```
  sns.distplot(df["previous"])
```

```
<ipython-input-99-615a65952c4c>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["previous"])
```

```
<Axes: xlabel='previous', ylabel='Density'>
```

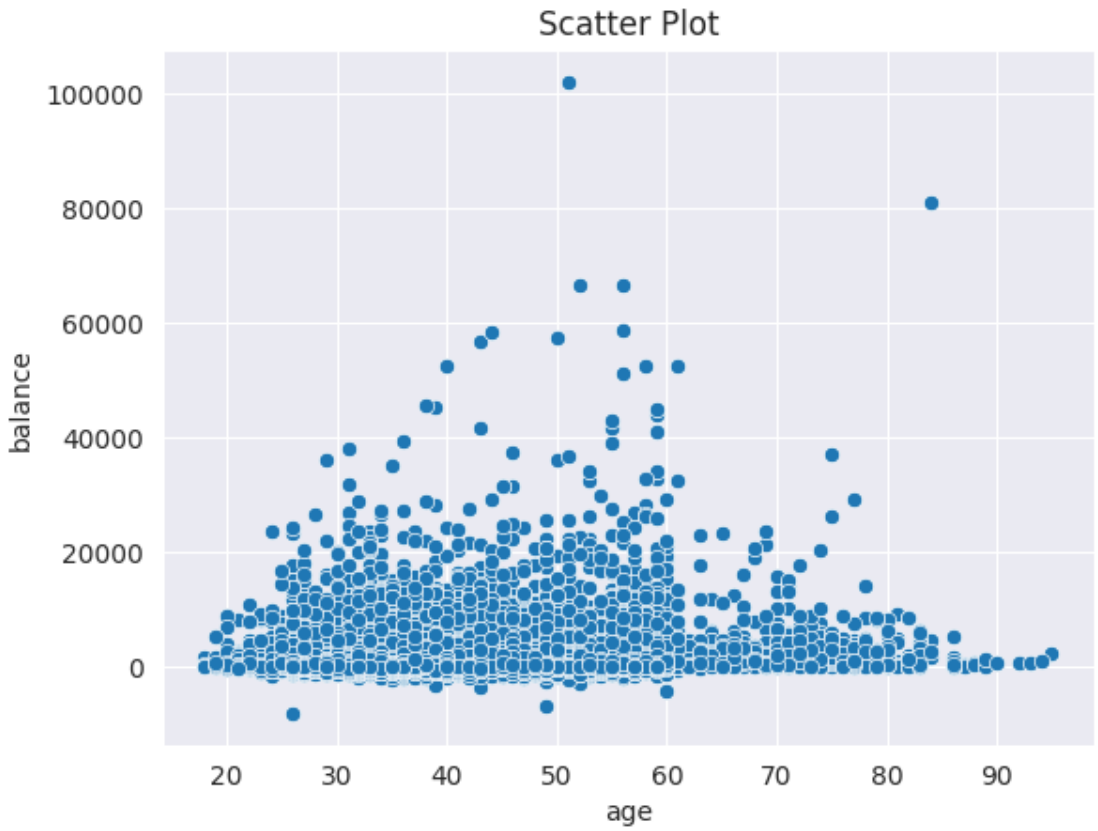



```
df.dtypes
```

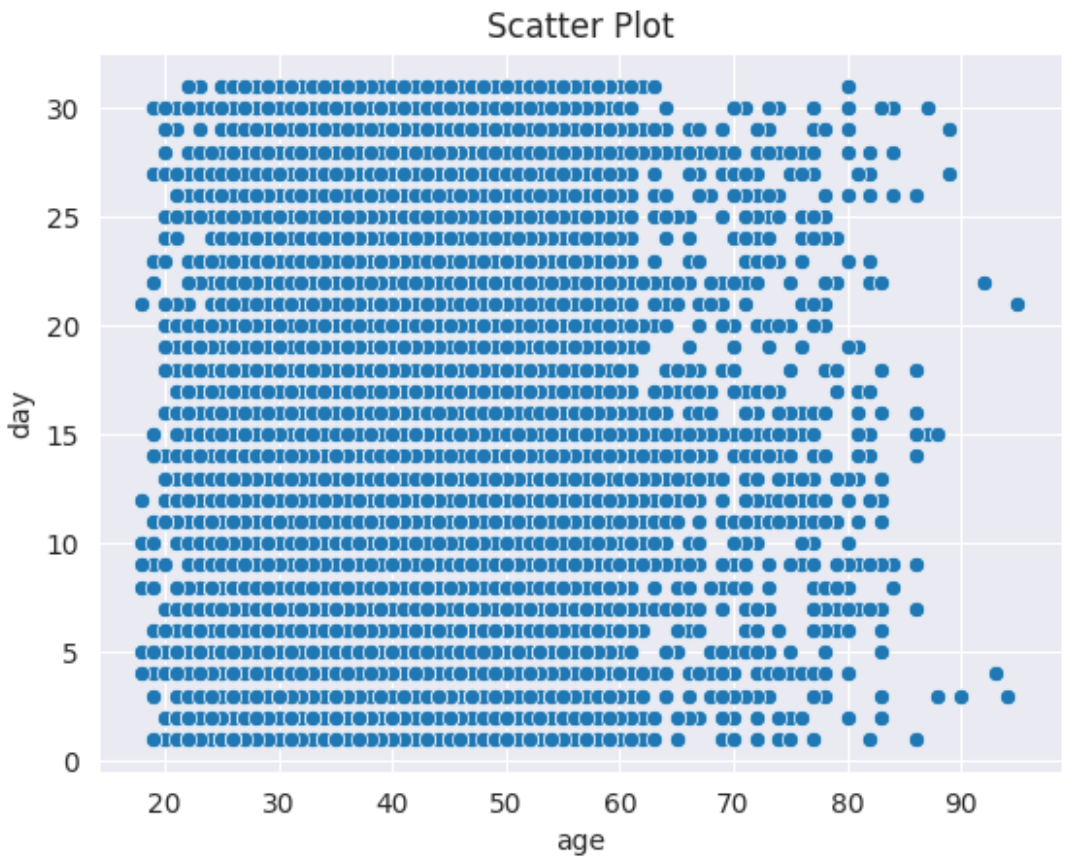
```
ID             int64
age            int64
job           object
marital       object
education     object
default       object
balance        int64
housing       object
loan          object
contact       object
day            int64
month         object
duration       int64
campaign       int64
pdays          int64
previous       int64
poutcome      object
subscribed    object
dtype: object
```

```
sns.scatterplot(data=df, x='age', y='balance')
```

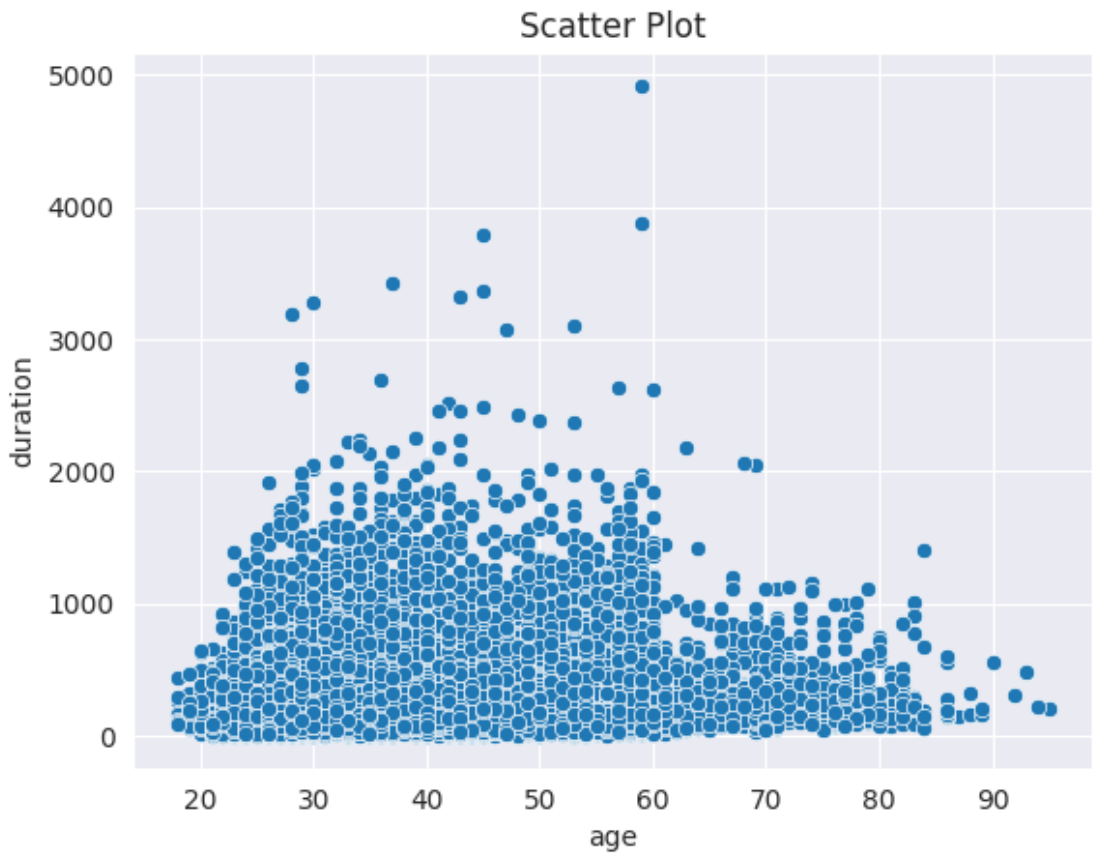
```
# Add labels and title
plt.xlabel('age')
plt.ylabel('balance')
plt.title('Scatter Plot')

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



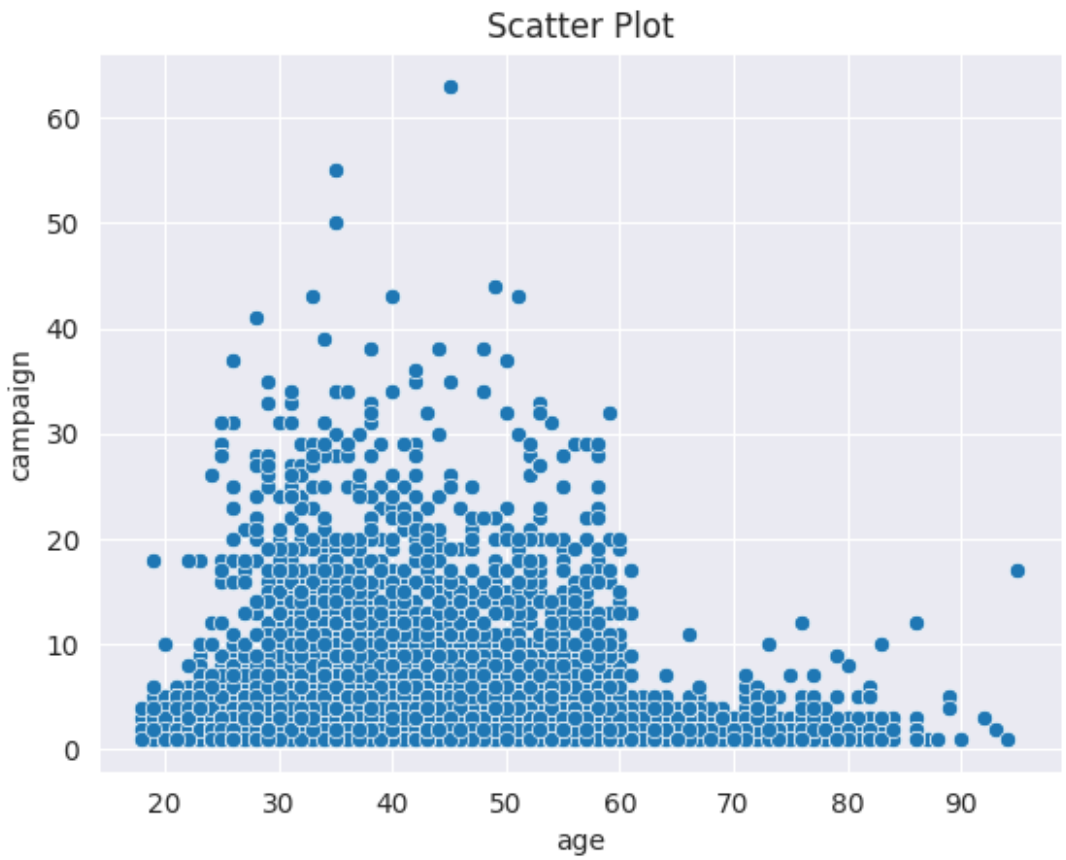

```
sns.scatterplot(data=df, x='age', y='day')

# Add labels and title
plt.xlabel('age')
plt.ylabel('day')
plt.title('Scatter Plot')

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



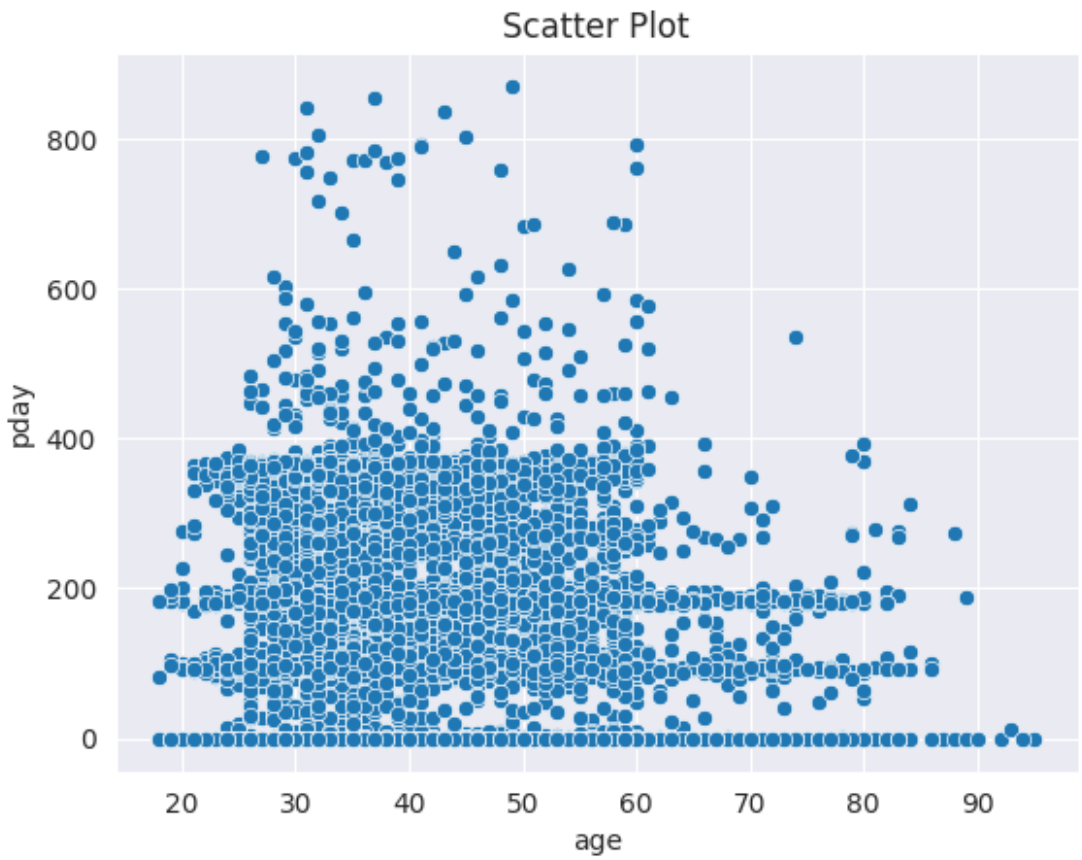

```
sns.scatterplot(data=df, x='age', y='duration')

# Add labels and title
plt.xlabel('age')
plt.ylabel('duration')
plt.title('Scatter Plot')

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




```
sns.scatterplot(data=df, x='age', y='campaign')

# Add labels and title
plt.xlabel('age')
plt.ylabel('campaign')
plt.title('Scatter Plot')

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



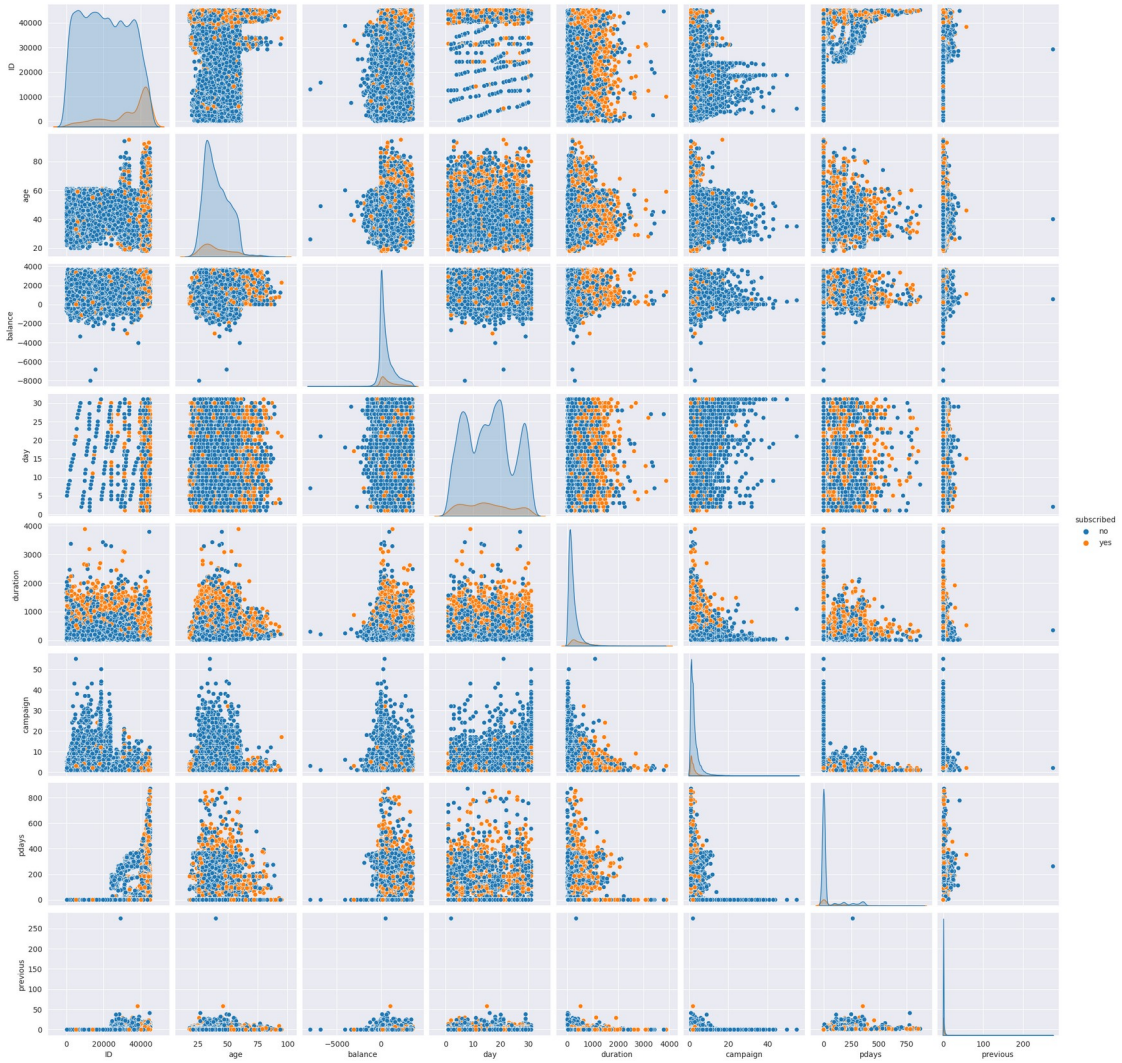

```
sns.scatterplot(data=df, x='age', y='pdays')

# Add labels and title
plt.xlabel('age')
plt.ylabel('pday')
plt.title('Scatter Plot')

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




```
plt.figure(figsize=(10,6))
sns.pairplot(df , hue="subscribed")
```

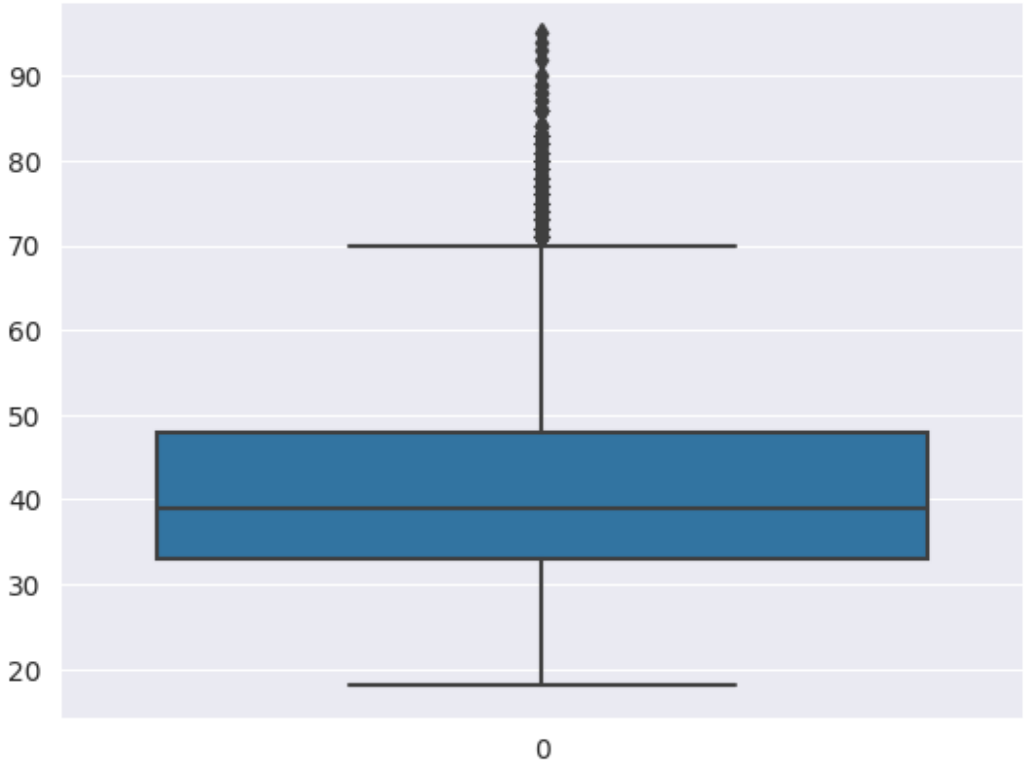
```
<seaborn.axisgrid.PairGrid at 0x7f9927aafdf0>
```

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



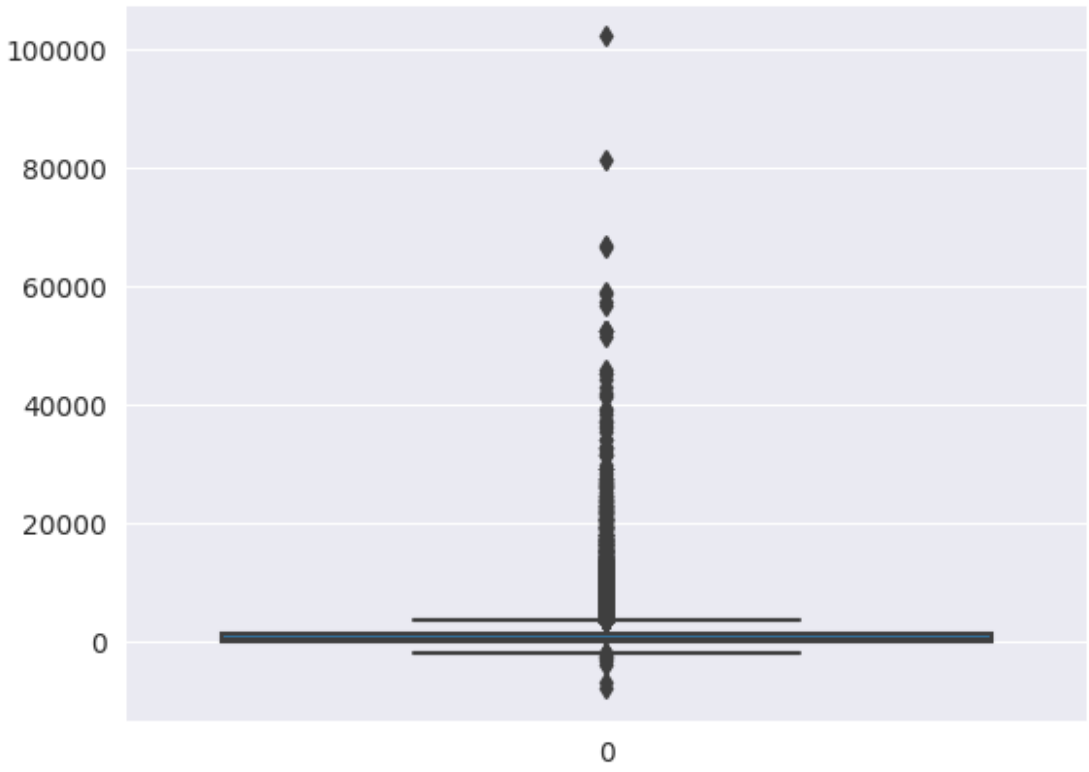

```
sns.boxplot(df['age'])
```

```
<Axes: >
```



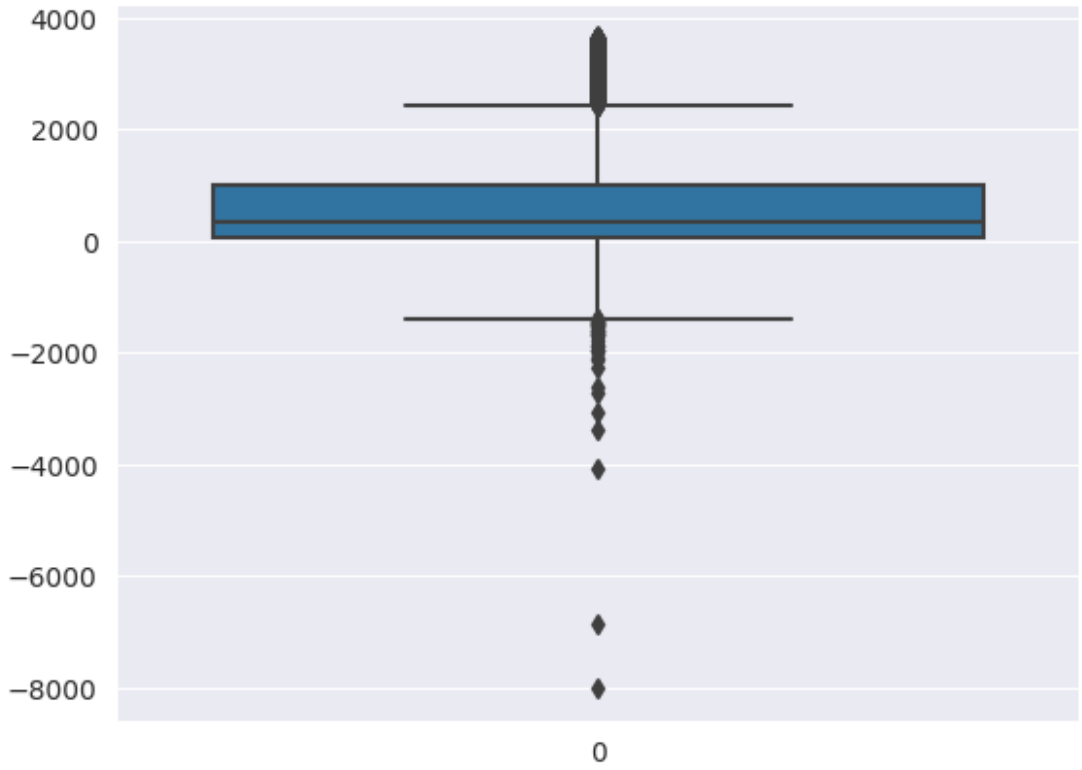

```
sns.boxplot(df['balance'])
```

```
<Axes: >
```




```
q = df['balance'].quantile(0.95)

# filter the DataFrame to exclude values above the 95th percentile
df = df[df['balance'] < q]


sns.boxplot(df['balance'])
```

```
<Axes: >
```

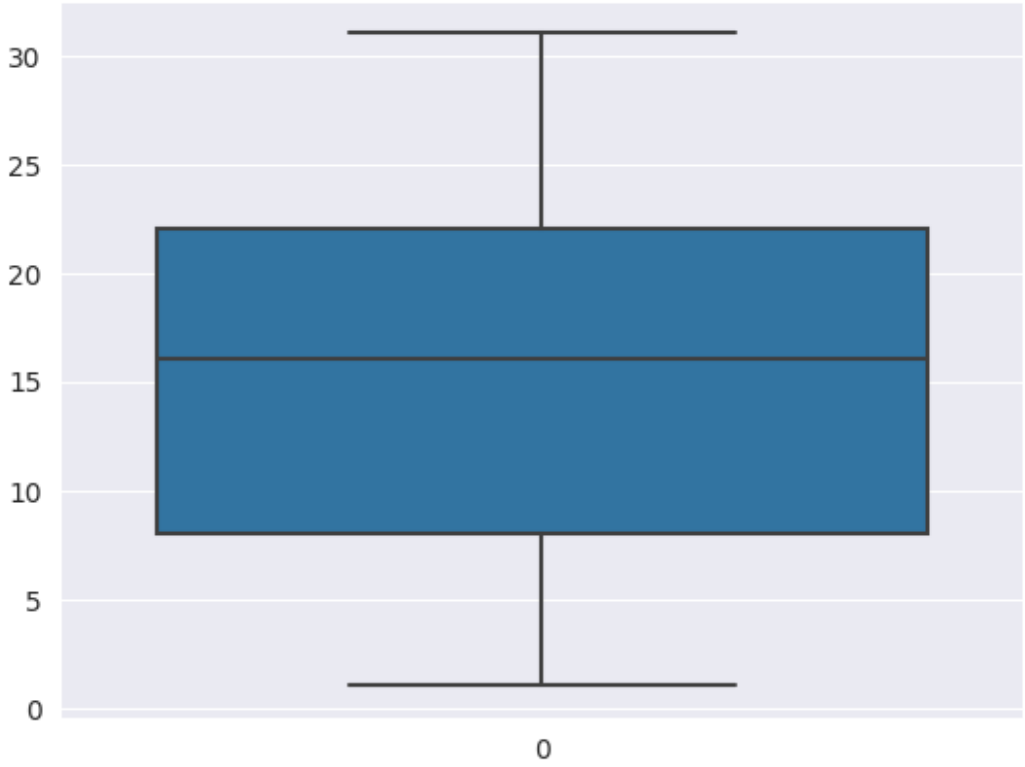



```
df.dtypes
```

```
ID             int64
age            int64
job           object
marital       object
education     object
default       object
balance        int64
housing       object
loan          object
contact       object
day            int64
month         object
duration       int64
campaign       int64
pdays          int64
previous       int64
poutcome      object
subscribed    object
dtype: object
```

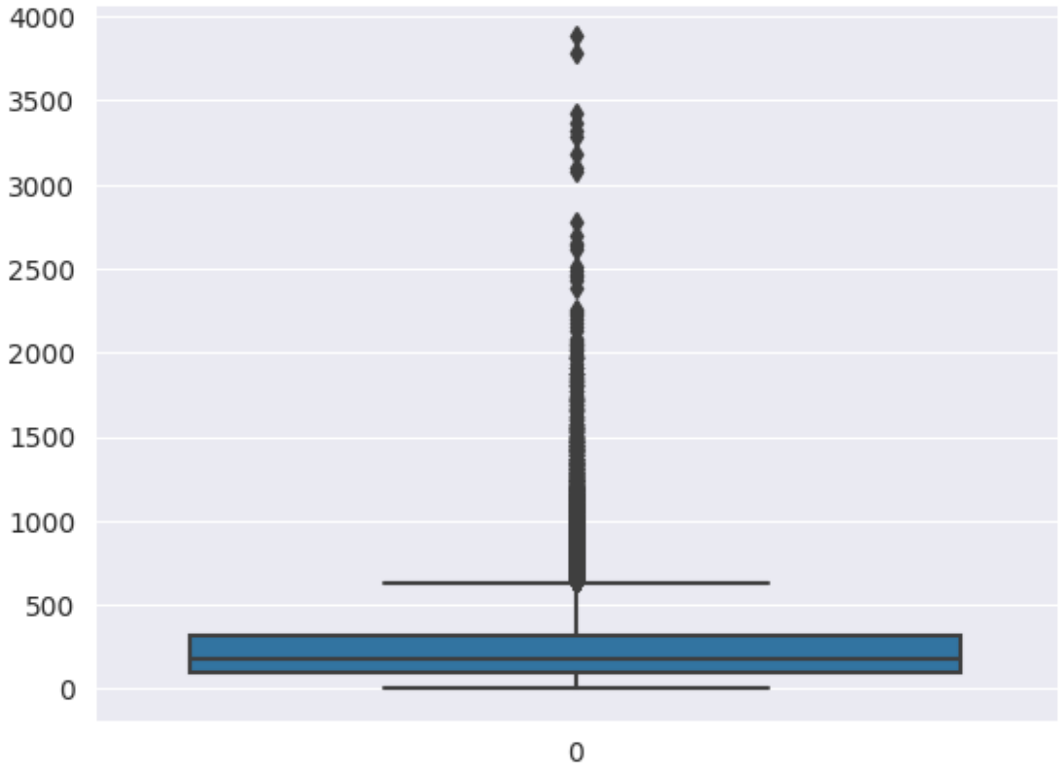
```
sns.boxplot(df['day'])
```

```
<Axes: >
```



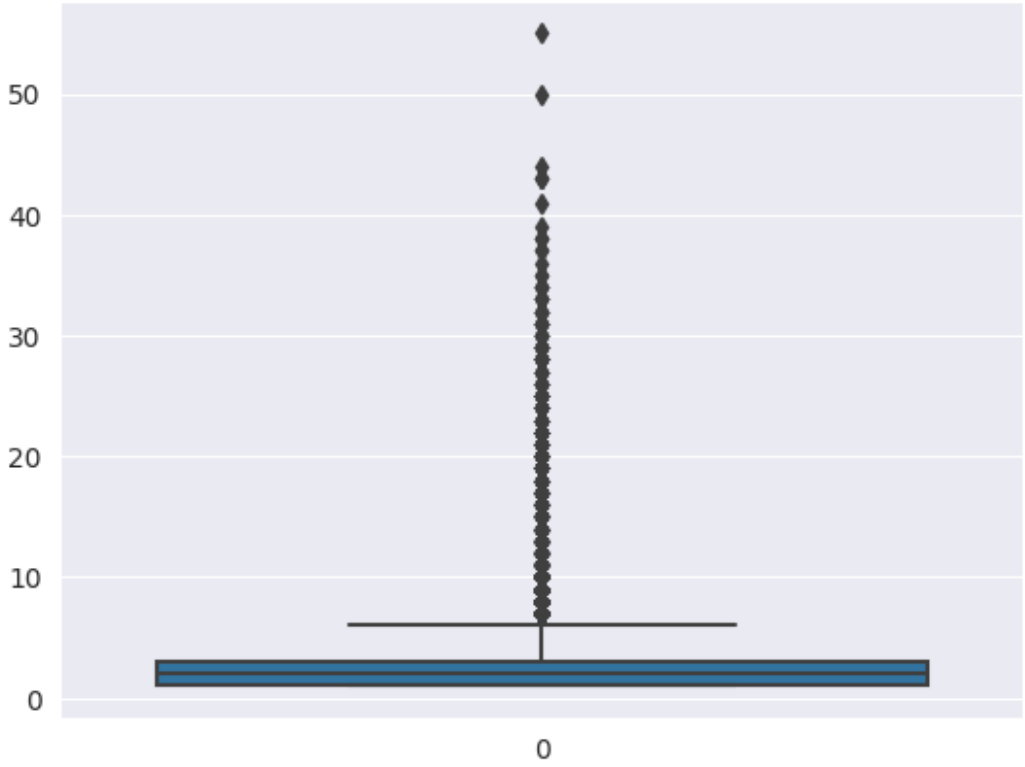

```
sns.boxplot(df['duration'])
```

```
<Axes: >
```



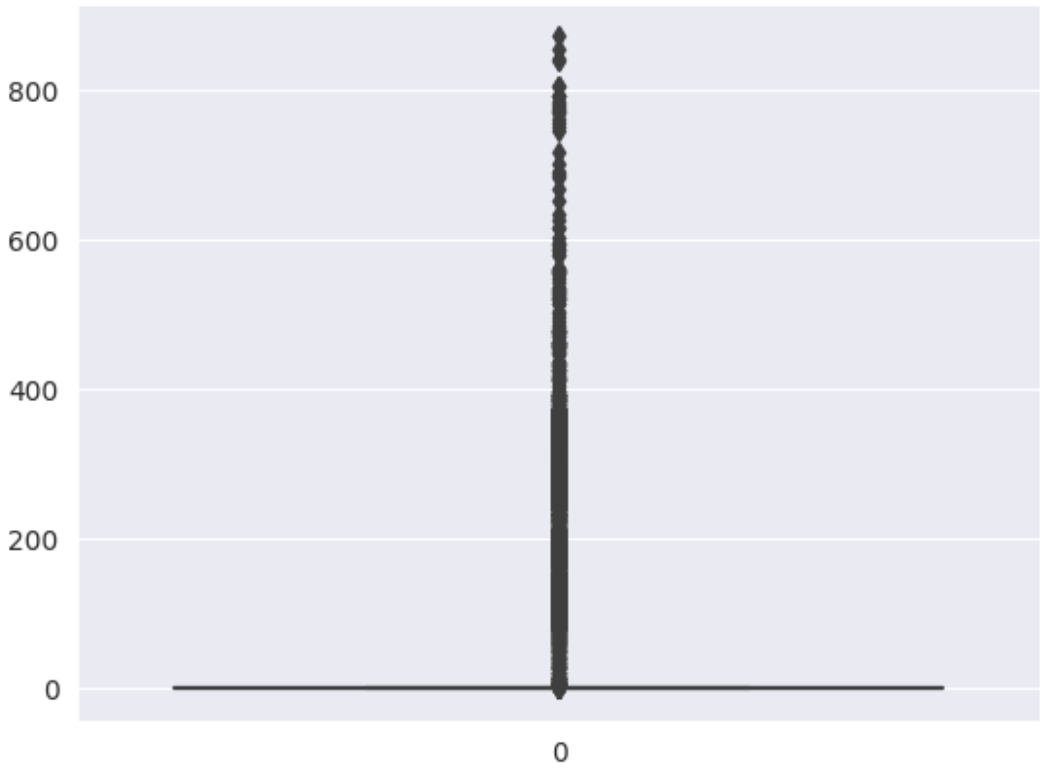

```
sns.boxplot(df['campaign'])
```

```
<Axes: >
```



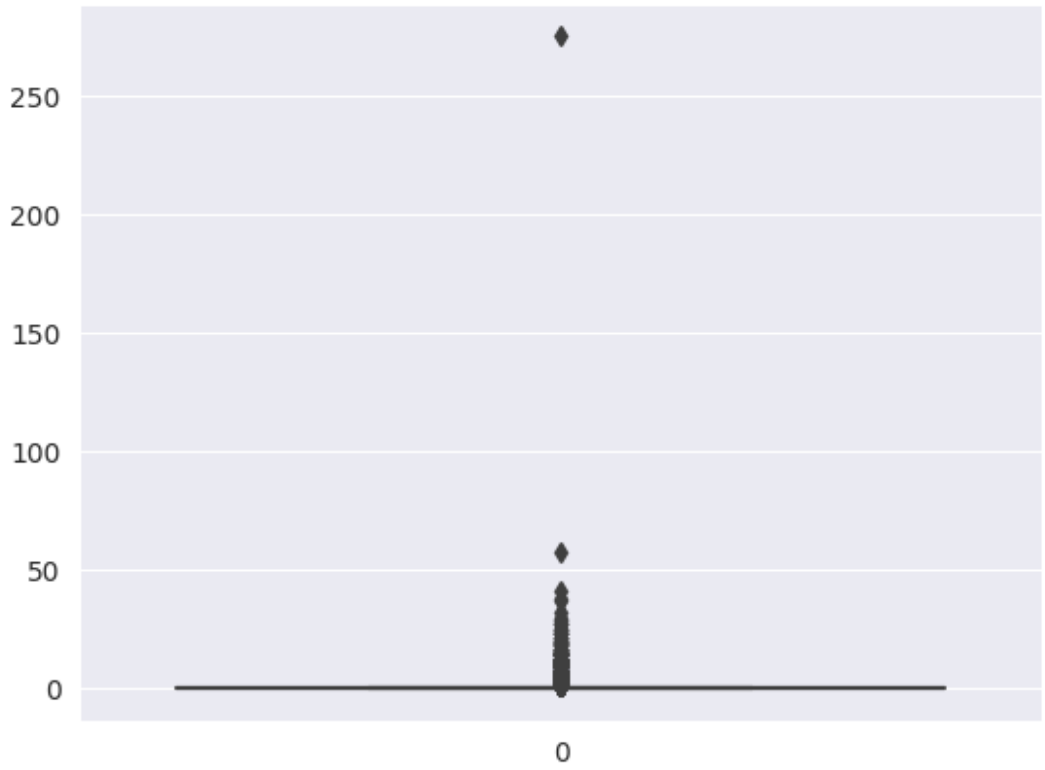

```
sns.boxplot(df['pdays'])
```

```
<Axes: >
```




```
sns.boxplot(df['previous'])
```

```
<Axes: >
```




```
df.dtypes
```

```
ID             int64
age            int64
job           object
marital       object
education     object
default       object
balance        int64
housing       object
loan          object
contact       object
day            int64
month         object
duration       int64
campaign       int64
pdays          int64
previous       int64
poutcome      object
subscribed    object
dtype: object
```

```
skewness=df[['ID','age','balance','day','duration','campaign','pdays',
'previous']]

print(skewness.skew())
```

```
ID           0.022241
age          0.678898
balance      1.249584
day          0.101965
duration     3.145838
campaign     4.765892
pdays        2.644174
previous    52.063050
dtype: float64
```

```
kurtosis=df[['ID','age','balance','day','duration','campaign','pdays',
'previous']]
print(kurtosis.kurt())
```

```
ID            -1.209371
age            0.301654
balance        1.942254
day           -1.087342
duration      17.626100
campaign      34.985834
pdays          7.087155
previous    5496.152091
dtype: float64
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

df_encoded = df.apply(le.fit_transform)

# print the encoded DataFrame
print(df_encoded)
```

```
          ID  age  job  marital  education  default  balance  housing
loan  \
0      16728   38    0        1          3        0     2719        0
0
1      25764   13   11        1          1        0      822        0
0
2       9923    9    7        1          1        0     1709        1
0
3      27785   39    4        0          2        0     3815        0
0
4      18961   13    9        1          1        0      938        1
0
...      ...  ...  ...      ...        ...      ...      ...      ...
...
31642  23164   11    4        2          2        0      819        1
0
31643  25536   35    4        0          2        0     1199        0
1
31644  12812   14    4        2          2        0     1131        0
```

```
0
31645  24507   39    9        1          1        0     1044        1
0
31646   9148   37    4        0          1        0     1023        1
0

       contact  day  month  duration  campaign  pdays  previous
poutcome  \
0            1   18      9        43         1      0         0
3
1            0   19      5        90         1      0         0
3
2            0   17      5       239         0      0         0
3
3            0   21      6       860         0     77         3
2
4            0    3      3       379         0      0         0
3
...        ...  ...    ...       ...       ...    ...       ...
...
31642        0   11      8       115         1      0         0
3
31643        0    4      6       437         1      0         0
3
31644        0    6      1        36         2      0         0
3
31645        1   14      8        21         6    330        12
0
31646        0   10      5      1371         1      0         0
3

       subscribed
0               0
1               0
2               0
3               1
4               0
...           ...
31642           0
31643           1
31644           0
31645           0
31646           1

[28556 rows x 18 columns]

df.dtypes

ID             int64
age            int64
```

```
job           object
marital       object
education     object
default       object
balance        int64
housing       object
loan          object
contact       object
day            int64
month         object
duration       int64
campaign       int64
pdays          int64
previous       int64
poutcome      object
subscribed    object
dtype: object
```

```
cat_columns =
['job','marital','education','default','housing','loan','contact','mon
th','poutcome','subscribed']

# Apply label encoding to categorical columns
for col in cat_columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])

# Print the encoded dataframe
print(df)
```

```
          ID  age  job  marital  education  default  balance  housing
loan  \
0      26110   56    0        1          3        0     1933        0
0
1      40576   31   11        1          1        0        3        0
0
2      15320   27    7        1          1        0      891        1
0
3      43962   57    4        0          2        0     3287        0
0
4      29842   31    9        1          1        0      119        1
0
...      ...  ...  ...      ...        ...      ...      ...      ...
...
31642  36483   29    4        2          2        0        0        1
0
31643  40178   53    4        0          2        0      380        0
1
31644  19710   32    4        2          2        0      312        0
0
```

```
31645  38556   57    9        1          1        0      225        1
0
31646  14156   55    4        0          1        0      204        1
0

       contact  day  month  duration  campaign  pdays  previous
poutcome  \
0            1   19      9        44         2     -1         0
3
1            0   20      5        91         2     -1         0
3
2            0   18      5       240         1     -1         0
3
3            0   22      6       867         1     84         3
2
4            0    4      3       380         1     -1         0
3
...        ...  ...    ...       ...       ...    ...       ...
...
31642        0   12      8       116         2     -1         0
3
31643        0    5      6       438         2     -1         0
3
31644        0    7      1        37         3     -1         0
3
31645        1   15      8        22         7    337        12
0
31646        0   11      5      1973         2     -1         0
3

       subscribed
0               0
1               0
2               0
3               1
4               0
...           ...
31642           0
31643           1
31644           0
31645           0
31646           1

[28556 rows x 18 columns]
```

```python
# Import libraries
from sklearn.model_selection import train_test_split

X = df.drop('subscribed', axis=1)
y = df['subscribed']
```

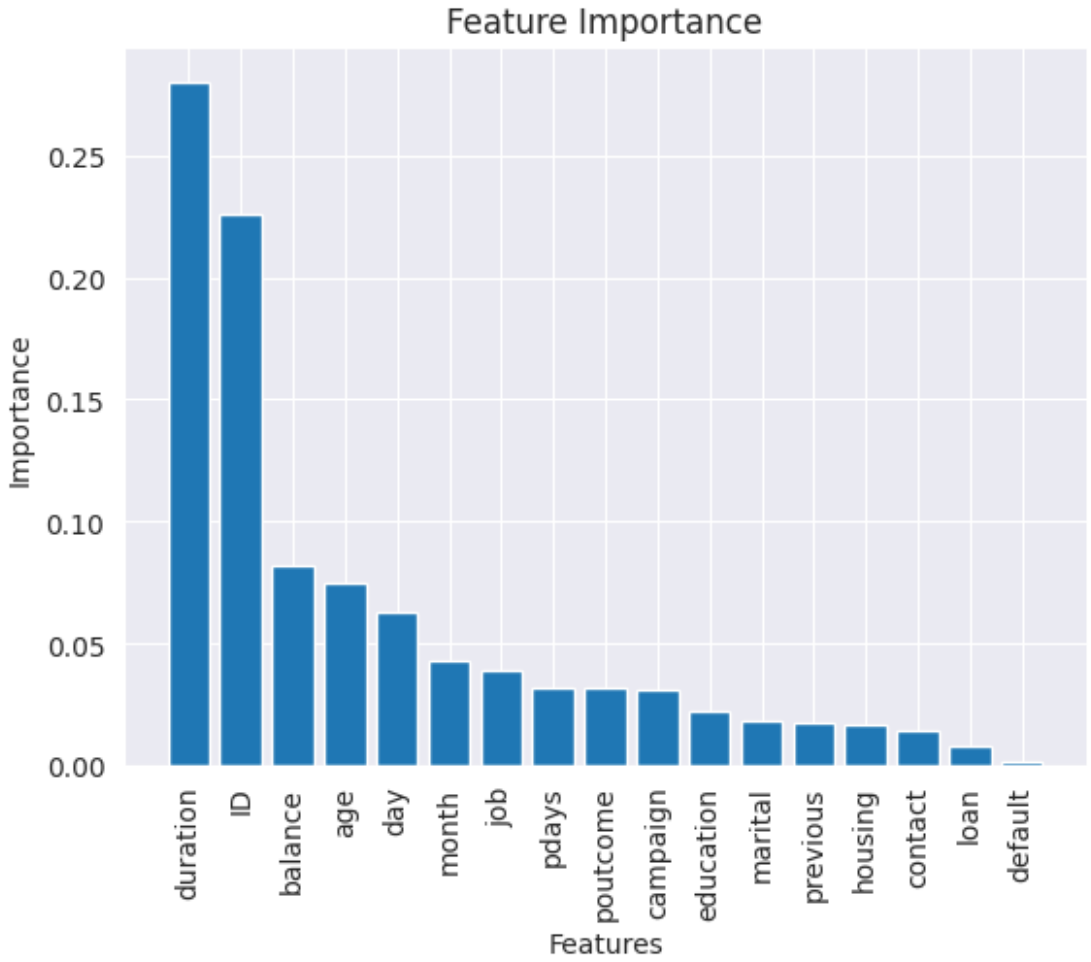
```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Print the shape of the training and testing sets
print("Training set shape:", X_train.shape, y_train.shape)
print("Testing set shape:", X_test.shape, y_test.shape)
```

```
Training set shape: (22844, 17) (22844,)
Testing set shape: (5712, 17) (5712,)
```

```
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt



rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X, y)


importances = rf_model.feature_importances_


sorted_indices = importances.argsort()[::-1]
sorted_importances = importances[sorted_indices]


print("Feature ranking:")
for i in range(X.shape[1]):
    print("%d. %s (%f)" % (i + 1, X.columns[sorted_indices[i]],
sorted_importances[i]))


plt.bar(range(X.shape[1]), sorted_importances)
plt.xticks(range(X.shape[1]), X.columns[sorted_indices], rotation=90)
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance')
plt.show()
```

```
Feature ranking:
1. duration (0.279997)
2. ID (0.225587)
3. balance (0.082023)
4. age (0.074897)
5. day (0.062483)
6. month (0.042858)
7. job (0.038827)
8. pdays (0.031894)
```

```
9. poutcome (0.031562)
10. campaign (0.031325)
11. education (0.022226)
12. marital (0.018302)
13. previous (0.017872)
14. housing (0.016435)
15. contact (0.014297)
16. loan (0.007569)
17. default (0.001847)
```




```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

X=df.drop(['default','loan','contact','subscribed'] , axis=1)
y=df["subscribed"]



X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

```
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)


y_pred = clf.predict(X_test)

# Compute the accuracy score of the classifier
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
Accuracy: 90.91%
```

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

X=df.drop(['default','loan','contact','subscribed'] , axis=1)
y=df["subscribed"]


# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Train a Gaussian Naive Bayes classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# Make predictions on the test set
y_pred = gnb.predict(X_test)

# Calculate the accuracy score
acc = accuracy_score(y_test, y_pred)
print("Accuracy score:", acc)
```

```
Accuracy score: 0.8419117647058824
```