BANK MARKETING CAMPAIGN

STEP-1: EXPLORATORY DATA ANALYSIS

[58]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

#1. Load the Dataset
data = pd.read_csv("/Users/rahulkumar/Downloads/Banking_Dataset.csv")

#2. Print first 10 rows
print (df.head(10))
```

```
   age          job  marital          education  default housing loan  \
0   44  blue-collar  married           basic.4y  unknown     yes   no
1   53   technician  married            unknown       no      no   no
2   28   management   single  university.degree       no     yes   no
3   39     services  married        high.school       no      no   no
4   55      retired  married           basic.4y       no     yes   no
5   30   management  divorced          basic.4y       no     yes   no
6   37  blue-collar  married           basic.4y       no     yes   no
7   39  blue-collar  divorced          basic.9y       no     yes   no
8   36       admin.  married  university.degree       no      no   no
9   27  blue-collar   single           basic.4y       no     yes   no

    contact month day_of_week  ...  campaign  pdays  previous     poutcome  \
0  cellular   aug         thu  ...         1    999         0  nonexistent
1  cellular   nov         fri  ...         1    999         0  nonexistent
2  cellular   jun         thu  ...         3      6         2      success
3  cellular   apr         fri  ...         2    999         0  nonexistent
4  cellular   aug         fri  ...         1      3         1      success
5  cellular   jul         tue  ...         8    999         0  nonexistent
6  cellular   may         thu  ...         1    999         0  nonexistent
7  cellular   may         fri  ...         1    999         0  nonexistent
8  cellular   jun         mon  ...         1      3         1      success
9  cellular   apr         thu  ...         2    999         1      failure
```

```
   emp_var_rate  cons_price_idx  cons_conf_idx  euribor3m  nr_employed  y
0           1.4          93.444          -36.1      4.963       5228.1  0
1          -0.1          93.200          -42.0      4.021       5195.8  0
2          -1.7          94.055          -39.8      0.729       4991.6  1
3          -1.8          93.075          -47.1      1.405       5099.1  0
4          -2.9          92.201          -31.4      0.869       5076.2  1
5           1.4          93.918          -42.7      4.961       5228.1  0
6          -1.8          92.893          -46.2      1.327       5099.1  0
7          -1.8          92.893          -46.2      1.313       5099.1  0
8          -2.9          92.963          -40.8      1.266       5076.2  1
9          -1.8          93.075          -47.1      1.410       5099.1  0

[10 rows x 21 columns]
```

[65]:
```python
# STEP 3: BASIC DATA CHECK
print("SHAPE:\n", df.shape)
print("\nINFO:\n")
data = df.info()
print("\nDESCRIBE:\n")
print(df.describe(include='all'))
print("\nCOLUMNS:\n")
print(df.columns)
```

```
SHAPE:
 (41176, 21)

INFO:

<class 'pandas.core.frame.DataFrame'>
Index: 41176 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41176 non-null  int64
 1   job             41176 non-null  object
 2   marital         41176 non-null  object
 3   education       41176 non-null  object
 4   default         41176 non-null  object
 5   housing         41176 non-null  object
 6   loan            41176 non-null  object
 7   contact         41176 non-null  object
 8   month           41176 non-null  object
 9   day_of_week     41176 non-null  object
```

```
INFO:

<class 'pandas.core.frame.DataFrame'>
Index: 41176 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             41176 non-null  int64  
 1   job             41176 non-null  object 
 2   marital         41176 non-null  object 
 3   education       41176 non-null  object 
 4   default         41176 non-null  object 
 5   housing         41176 non-null  object 
 6   loan            41176 non-null  object 
 7   contact         41176 non-null  object 
 8   month           41176 non-null  object 
 9   day_of_week     41176 non-null  object 
 10  duration        41176 non-null  int64  
 11  campaign        41176 non-null  int64  
 12  pdays           41176 non-null  int64  
 13  previous        41176 non-null  int64  
 14  poutcome        41176 non-null  object 
 15  emp_var_rate    41176 non-null  float64
 16  cons_price_idx  41176 non-null  float64
 17  cons_conf_idx   41176 non-null  float64
 18  euribor3m       41176 non-null  float64
 19  nr_employed     41176 non-null  float64
 20  y               41176 non-null  int64  
dtypes: float64(5), int64(6), object(10)
memory usage: 6.9+ MB

DESCRIBE:

                age     job  marital          education default housing  \
count   41176.00000   41176    41176              41176   41176   41176   
unique          NaN      12        4                  8       3       3   
top             NaN  admin.  married  university.degree      no     yes   
freq            NaN   10419    24921              12164   32577   21571   
mean       40.02380     NaN      NaN                NaN     NaN     NaN   
std        10.42068     NaN      NaN                NaN     NaN     NaN   
min        17.00000     NaN      NaN                NaN     NaN     NaN   
25%        32.00000     NaN      NaN                NaN     NaN     NaN   
50%        38.00000     NaN      NaN                NaN     NaN     NaN   
```

```
              age     job  marital          education default housing  \
count   41176.00000   41176    41176              41176   41176   41176
unique          NaN      12        4                  8       3       3
top             NaN  admin.  married  university.degree      no     yes
freq            NaN   10419    24921              12164   32577   21571
mean       40.02380     NaN      NaN                NaN     NaN     NaN
std        10.42068     NaN      NaN                NaN     NaN     NaN
min        17.00000     NaN      NaN                NaN     NaN     NaN
25%        32.00000     NaN      NaN                NaN     NaN     NaN
50%        38.00000     NaN      NaN                NaN     NaN     NaN
75%        47.00000     NaN      NaN                NaN     NaN     NaN
max        98.00000     NaN      NaN                NaN     NaN     NaN

         loan   contact  month day_of_week  ...      campaign         pdays  \
count   41176     41176  41176       41176  ...  41176.000000  41176.000000
unique      3         2     10           5  ...           NaN           NaN
top        no  cellular    may         thu  ...           NaN           NaN
freq    33938     26135  13767        8618  ...           NaN           NaN
mean      NaN       NaN    NaN         NaN  ...      2.567879    962.464810
std       NaN       NaN    NaN         NaN  ...      2.770318    186.937102
min       NaN       NaN    NaN         NaN  ...      1.000000      0.000000
25%       NaN       NaN    NaN         NaN  ...      1.000000    999.000000
50%       NaN       NaN    NaN         NaN  ...      2.000000    999.000000
75%       NaN       NaN    NaN         NaN  ...      3.000000    999.000000
max       NaN       NaN    NaN         NaN  ...     56.000000    999.000000

            previous     poutcome  emp_var_rate  cons_price_idx  \
count   41176.000000        41176  41176.000000    41176.000000
unique           NaN            3           NaN             NaN
top              NaN  nonexistent           NaN             NaN
freq             NaN        35551           NaN             NaN
mean        0.173013          NaN      0.081922       93.575720
std         0.494964          NaN      1.570883        0.578839
min         0.000000          NaN     -3.400000       92.201000
25%         0.000000          NaN     -1.800000       93.075000
50%         0.000000          NaN      1.100000       93.749000
75%         0.000000          NaN      1.400000       93.994000
max         7.000000          NaN      1.400000       94.767000

        cons_conf_idx     euribor3m   nr_employed             y
count    41176.000000  41176.000000  41176.000000  41176.000000
unique            NaN           NaN           NaN           NaN
```

```
mean      -40.502863     3.621293   5167.034870     0.112663
std         4.627860     1.734437     72.251364     0.316184
min       -50.800000     0.634000   4963.600000     0.000000
25%       -42.700000     1.344000   5099.100000     0.000000
50%       -41.800000     4.857000   5191.000000     0.000000
75%       -36.400000     4.961000   5228.100000     0.000000
max       -26.900000     5.045000   5228.100000     1.000000

[11 rows x 21 columns]

COLUMNS:

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')
```

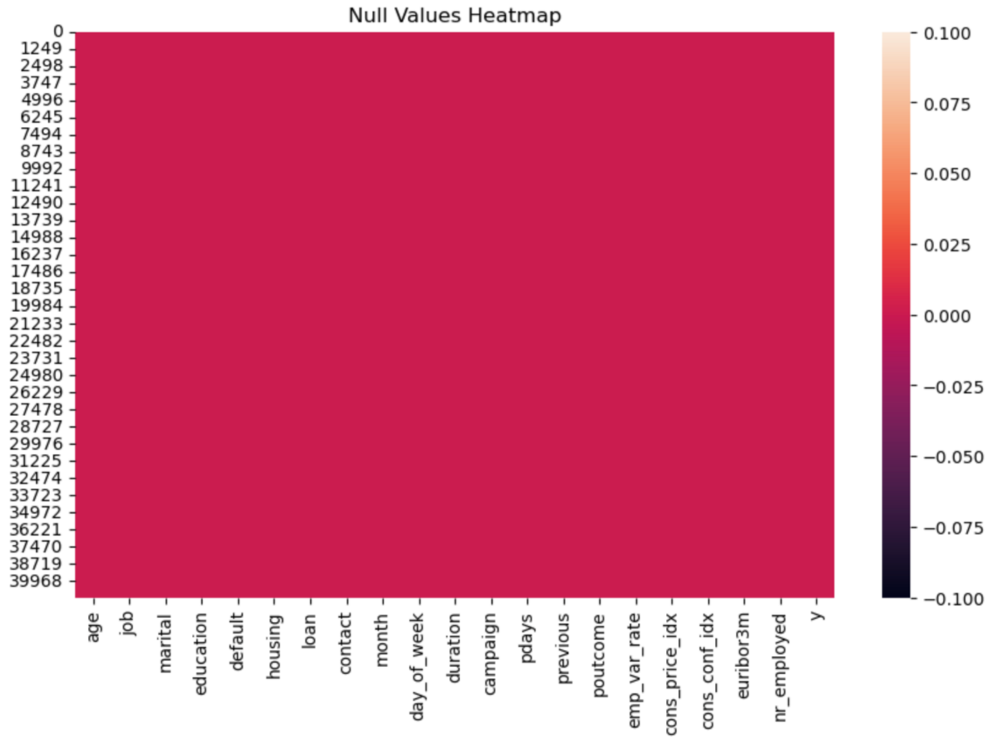
[16]:

```python
# STEP 4: CHECK MISSING VALUES
df.isnull().sum() # null values
```

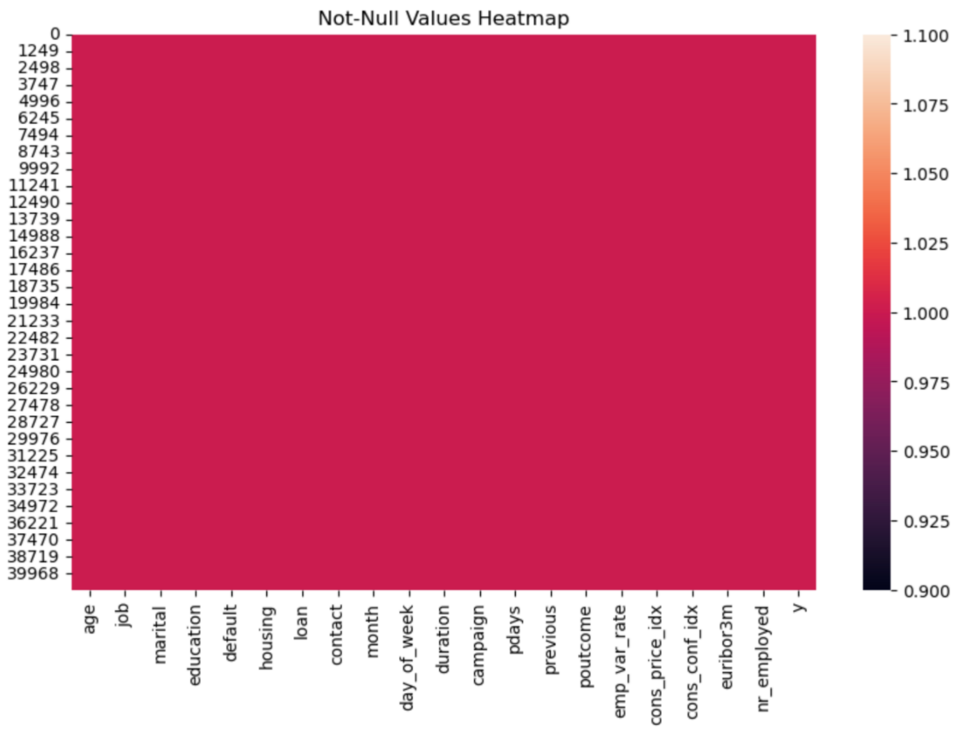
[16]:

```
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
```

•[24]:
```
# STEP 5: HEATMAP OF NULL VALUES
plt.figure(figsize=(10, 6))
sns.heatmap(df.isnull(), cbar=True)
plt.title("Null Values Heatmap")
plt.show()
```

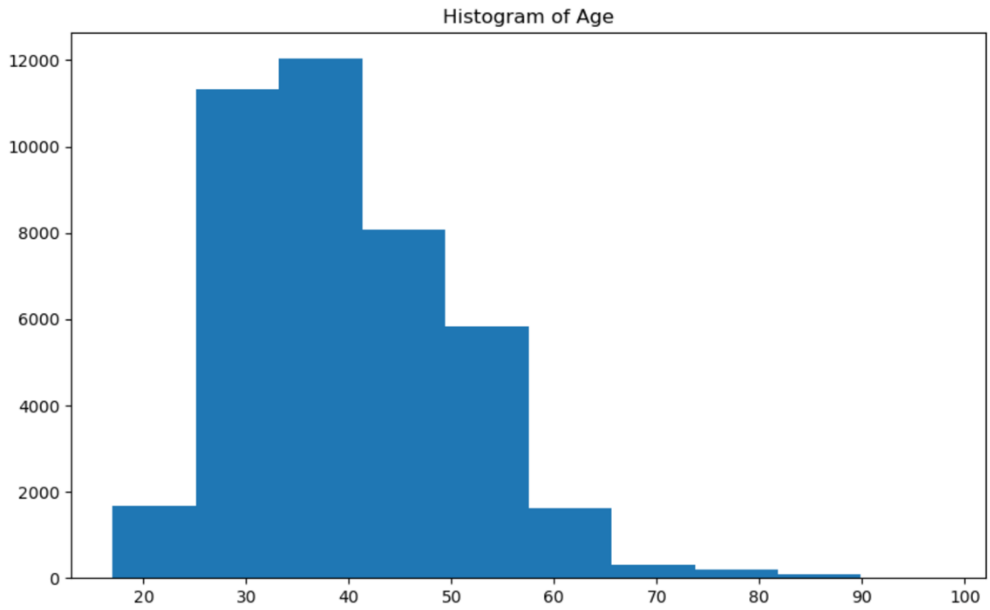


[25]:
```
plt.figure(figsize=(10, 6))
sns.heatmap(df.notnull(), cbar=True)
plt.title("Not-Null Values Heatmap")
plt.show()
```



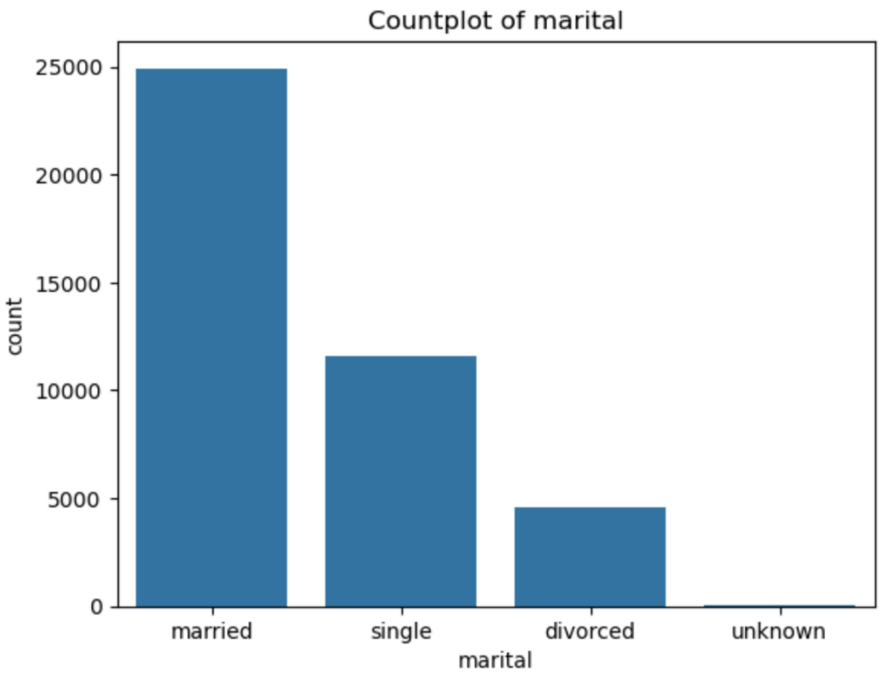
[29]:

```
# STEP 8: UNIVARIATE VISUALIZATIONS
# Histogram
plt.figure(figsize=(10,6))
plt.hist(df['age'])
plt.title("Histogram of Age")
plt.show()
```

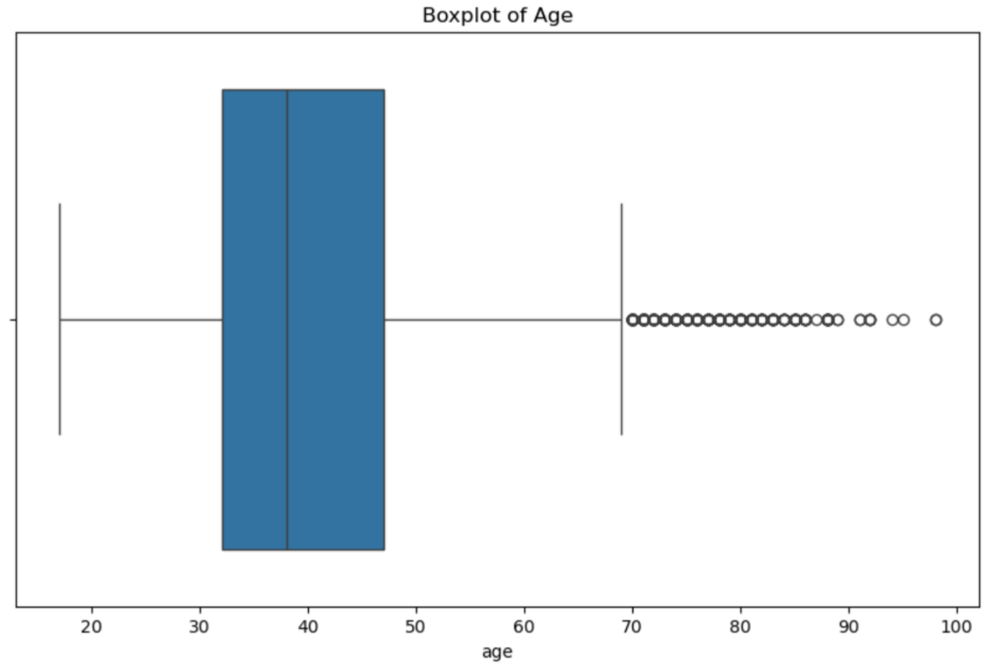


[34]:

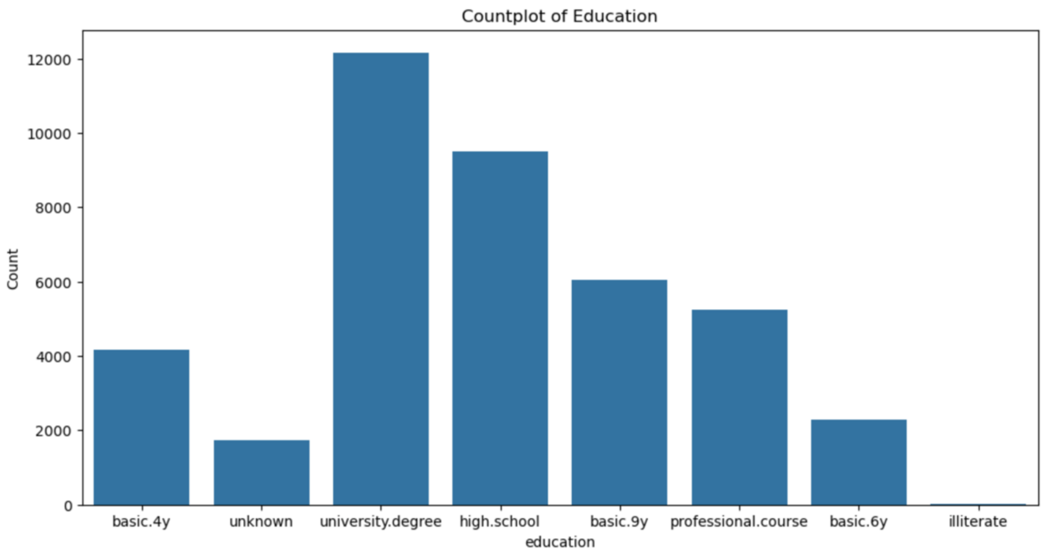
```
# Countplot
plt.figure()
sns.countplot(x=df['marital'])
plt.title("Countplot of marital")
plt.show()
```



[37]:
```
# Boxplot
plt.figure(figsize=(10,6))
sns.boxplot(x=df['age'])
plt.title("Boxplot of Age")
plt.show()
```



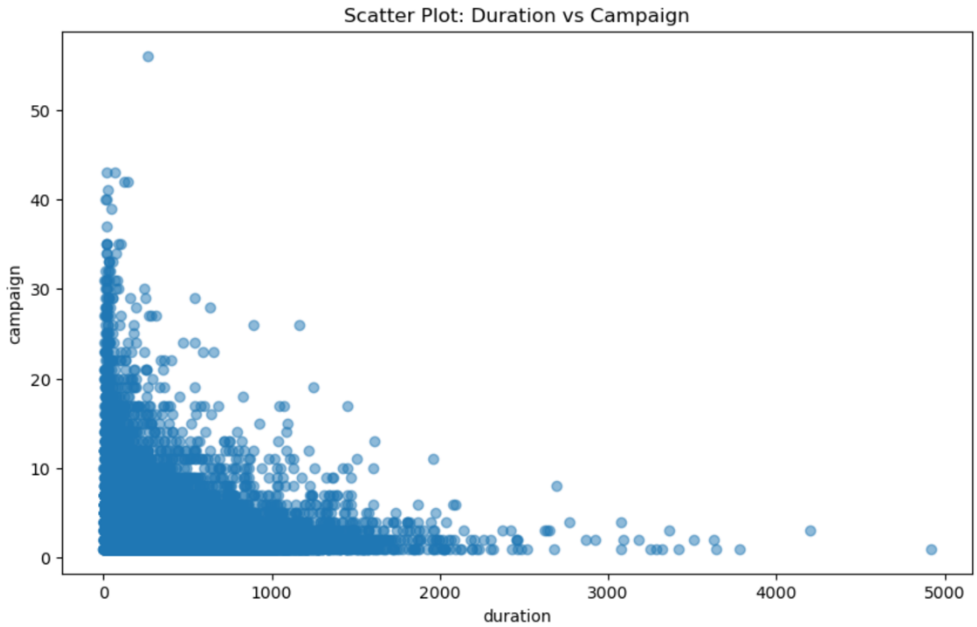
[40]:
```
# HISTOGRAM
plt.figure(figsize=(12, 6))
sns.countplot(x=df['education'])
plt.title('Countplot of Education')
plt.xlabel('education')
plt.ylabel('Count')
plt.show()
```



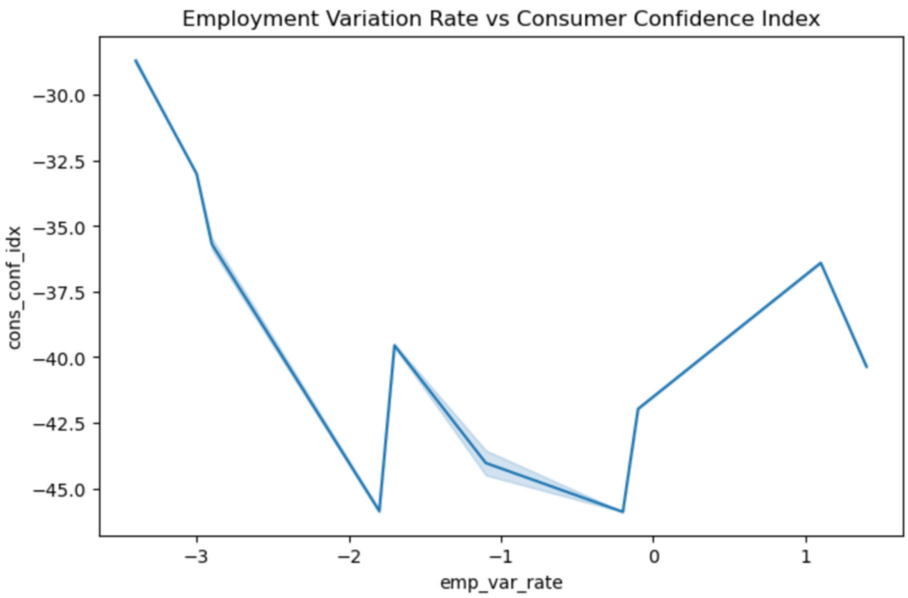
[42]:
```python
# STEP 9: BIVARIATE VISUALIZATIONS
# SCATTER PLOT
plt.figure(figsize=(10,6))
plt.scatter(df['duration'], df['campaign'], alpha=0.5)

plt.title("Scatter Plot: Duration vs Campaign")
plt.xlabel("duration")
plt.ylabel("campaign")

plt.show()
```

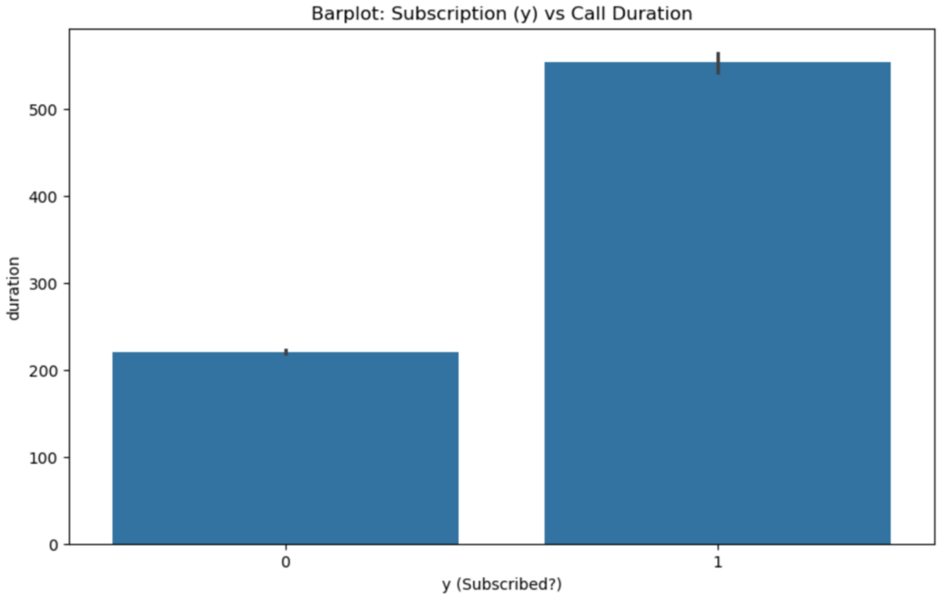


[47]:
```python
# LINE PLOT
plt.figure(figsize=(8,5))
sns.lineplot(x='emp_var_rate', y='cons_conf_idx', data=df)
plt.title('Employment Variation Rate vs Consumer Confidence Index')
plt.xlabel('emp_var_rate')
plt.ylabel('cons_conf_idx')
plt.show()
```



[49]:

```
plt.figure(figsize=(10,6))
sns.barplot(x='y', y='duration', data=df)
plt.title("Barplot: Subscription (y) vs Call Duration")
plt.xlabel("y (Subscribed?)")
plt.ylabel("duration")
plt.show()
```



STEP-2: MODELING

[50]:
```python
# ALL COLUMN NAMES
df.columns
```

[50]:
```
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')
```

[52]:
```python
# FEATURE SELECTION AND TARGET COLUMN
x = df[['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']]
y = df['y']
```

[53]:
```python
# Convert Categorical Columns to Numeric
x = pd.get_dummies(x, drop_first=True)
```

[54]:
```python
#Train-Test Split
x_train, x_test, y_train, y_test= train_test_split(
x, y, test_size=0.2, random_state=42)
```

[55]:
```python
#Scaling (StandardScaler)
scaler = StandardScaler()
x_train= scaler.fit_transform(x_train)
x_test= scaler.transform(x_test)
```

[56]:
```python
# Logistic Regression
lr = LogisticRegression()
lr.fit(x_train, y_train)
y_pred_lr= lr.predict(x_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))
```

```
Logistic Regression Accuracy: 1.0
```

[61]:
```python
# KNN (K-Nearest Neighbors)
knn = KNeighborsClassifier(n_neighbors=5)
```

[61]:
```python
# KNN (K-Nearest Neighbors)
knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(x_train, y_train)
y_pred_knn = knn.predict(x_test)
print("KNN Accuracy:", accuracy_score(y_test, y_pred_knn))
```

```
KNN Accuracy: 0.9758377853326857
```

[62]:
```python
# SVC (Support Vector Classifier)
svc = SVC()
svc.fit(x_train, y_train)
y_pred_svc= svc.predict(x_test)
print("SVC Accuracy:", accuracy_score(y_test, y_pred_svc))
```

```
SVC Accuracy: 0.9995143273433705
```

[64]:
```python
from sklearn.metrics import accuracy_score

LR_accuracy = accuracy_score(y_test, y_pred_lr)
KNN_accuracy = accuracy_score(y_test, y_pred_knn)
SVC_accuracy = accuracy_score(y_test, y_pred_svc)

print("MODEL ACCURACY SUMMARY")
print("Logistic Regression Accuracy:", LR_accuracy)
print("KNN Accuracy:", KNN_accuracy)
print("SVC Accuracy:", SVC_accuracy)
```

```
MODEL ACCURACY SUMMARY
Logistic Regression Accuracy: 1.0
KNN Accuracy: 0.9758377853326857
SVC Accuracy: 0.9995143273433705
```

[11]:
```python
# STEP 10: BAR GRAPH FOR MODEL ACCURACY COMPARISON

import matplotlib.pyplot as plt

models = ['Logistic Regression', 'KNN', 'SVC']
accuracies = [1.0, 0.9753877853323685 7, 0.9995143273433705]

colors = ['blue', 'orange', 'brown']
```

[11]:

```python
# STEP 10: BAR GRAPH FOR MODEL ACCURACY COMPARISON

import matplotlib.pyplot as plt

models = ['Logistic Regression', 'KNN', 'SVC']
accuracies = [1.0, 0.9753877853323685 7, 0.9995143273433705]

colors = ['blue', 'orange', 'brown']

plt.figure(figsize=(10,6))
plt.bar(models, accuracies, color=colors)

plt.xlabel("Models")
plt.ylabel("Accuracy")
plt.title("Accuracy Comparison of ML Models")
plt.ylim(0.8, 1.05)
plt.xticks(rotation=15)

plt.show()
```