Project: Bank Marketing (Campaign) -- Group Project

Group Name: Kesimoji
Names: Kemal Cagin Sertkaya, Jinwen Li, Mohamed Elmorsy, Sirui Zhang
Emails: cagin24@gmail.com (mailto:cagin24@gmail.com), jinwen@uw.edu (mailto:jinwen@uw.edu), mmsobhy7@gmail.com (mailto:mmsobhy7@gmail.com), zhangsirui261918@126.com (mailto:zhangsirui261918@126.com)
Colleges: Bogazici University, University of Washington, Mcmaster University, UCL
Specialization: Data Science
Countries: Turkey, US, Canada, UK

Problem description:

One bank wants to sell its term deposit product to customers before launching the product. To save their resource and time, they want to know what kind of customers they should focus on, and then they can put more advertisements to these customers, who have more chances of buying the product. Thus, our problem is to pick up this kind of customer, based on customers' past interaction with this bank or other financial institutions. We are going to use the customers' data to build some machine learning models and then, select customers who most likely buy the product.

Data cleansing and transformation done on the data.

1. Load Data

In [2]:
```
# Import packages
import pandas as pd
from pandas import factorize
import numpy as np
import os, glob
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import seaborn as sb
plt.rcParams.update(plt.rcParamsDefault)
import calendar
```

In [3]:
```
# Define functions
def prob(x):
    x = round(x.div(len(data))*100, 2)
    return x
```

In [4]:
```
# load data
data = pd.read_csv("/Users/jinwen/Downloads/data_glacier_6-9/bank/bank-full
data.head()
```

Out[4]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	du
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	

2. Explore Data

In [5]:
```
data.dtypes
```

Out[5]:
```
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
y            object
dtype: object
```

In [6]:
```
col = data.columns.tolist()
col_num = data.select_dtypes(include=np.number).columns.tolist()
```

In [7]:
```
print("Number of unique values stat:")
data.nunique()
```

```
Number of unique values stat:
```

Out[7]:
```
age            77
job            12
marital         3
education       4
default         2
balance      7168
housing         2
loan            2
contact         3
day            31
month          12
duration     1573
campaign       48
pdays         559
previous       41
poutcome        4
y               2
dtype: int64
```

In [8]:
```
x = prob(data.isnull().sum())
print("Percentage of null values in data: ")
x
```

```
Percentage of null values in data:
```

Out[8]:
```
age          0.0
job          0.0
marital      0.0
education    0.0
default      0.0
balance      0.0
housing      0.0
loan         0.0
contact      0.0
day          0.0
month        0.0
duration     0.0
campaign     0.0
pdays        0.0
previous     0.0
poutcome     0.0
y            0.0
dtype: float64
```

In [24]:
```
data.describe().applymap('{:,.0f}'.format)
```

Out[24]:

	age	balance	day	duration	campaign	pdays	previous
count	7,842	7,842	7,842	7,842	7,842	7,842	7,842
mean	41	1,552	14	261	2	223	3
std	11	3,085	8	236	2	112	5
min	18	-1,884	1	5	1	1	1
25%	32	162	7	113	1	133	1
50%	38	595	14	194	2	195	2
75%	47	1,734	20	324	2	326	4
max	89	81,204	31	2,219	16	871	275

2. Outliers Removal

In order to detect and remove outliers, here we use two statistical methods: Interquartile range(IQR) and Standard Deviation.

In [10]:
```
# Outliers removal using Interquartile range(IQR) statistical method
def outliers_iqr(df, feature):
    Q1= df[feature].quantile(0.25)
    Q3 = df[feature].quantile(0.75)
    IQR = Q3 - Q1
    upper_limit = Q3 + 1.5 * IQR
    lower_limit = Q1 - 1.5 * IQR
    return upper_limit, lower_limit

for col in col_num:
    upper, lower = outliers_iqr(data, col)
    print(str(col)+":")
    print("Upper limit: ", upper)
    print("Lower limit: ", lower)
    if upper > lower:
        data_iqr = data[(data[col] > lower) & (data[col] < upper)]

data_iqr.describe().applymap('{:,.0f}'.format)
```

```
age:
Upper limit:  70.5
Lower limit:  10.5
balance:
Upper limit:  3462.0
Lower limit:  -1962.0
day:
Upper limit:  40.5
Lower limit:  -11.5
duration:
Upper limit:  643.0
Lower limit:  -221.0
campaign:
Upper limit:  6.0
Lower limit:  -2.0
pdays:
Upper limit:  -1.0
Lower limit:  -1.0
previous:
Upper limit:  0.0
Lower limit:  0.0
```

Out[10]:

	age	balance	day	duration	campaign	pdays	previous
count	40,856	40,856	40,856	40,856	40,856	40,856	40,856
mean	41	1,369	15	265	2	42	1
std	11	3,053	8	258	1	102	2
min	18	-8,019	1	0	1	-1	0
25%	33	76	8	109	1	-1	0
50%	39	455	15	187	2	-1	0
75%	48	1,440	21	326	3	-1	0
max	95	102,127	31	4,918	5	871	275

In [11]:
```
# Outliers removal using Standard Deviation statistical method
def outlier_std(df, variable):
    upper_limit = df[variable].mean() + 3 * df[variable].std()
    lower_limit = df[variable].mean() - 3 * df[variable].std()
    return upper_limit, lower_limit

for col in col_num:
    upper_limit, lower_limit = outlier_std(data, col)
    print(str(col)+":")
    print("Upper limit: ", upper_limit)
    print("Lower Limit: ",lower_limit)
    data_std = data[(data[col] > lower_limit) & (data[col] < upper_limit)]

data_std.describe().applymap('{:,.0f}'.format)
```

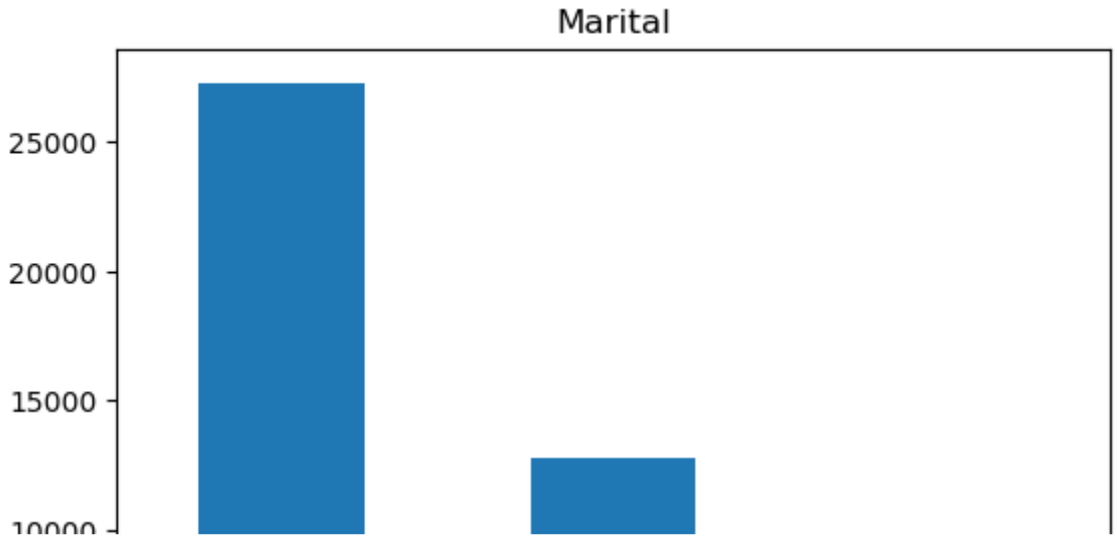
```
Lower Limit:  -260.18841000959253
previous:
Upper limit:  7.490646507424825
Lower Limit:  -6.329999762163715
```

Out[11]:

	age	balance	day	duration	campaign	pdays	previous
count	44,629	44,629	44,629	44,629	44,629	44,629	44,629
mean	41	1,360	16	258	3	38	0
std	11	3,050	8	258	3	98	1
min	18	-8,019	1	0	1	-1	0
25%	33	71	8	103	1	-1	0
50%	39	446	16	180	2	-1	0
75%	48	1,420	21	319	3	-1	0
max	95	102,127	31	4,918	63	871	7

3.Process of NA values

In [12]:
```
#Process of NA Values

plt.title('Job')
data.job.value_counts().plot(kind='bar')
plt.show()
plt.title('Marital')
data.marital.value_counts().plot(kind='bar')
plt.show()
plt.title('Education')
data.education.value_counts().plot(kind='bar')
plt.show()
plt.title('Default')
data.default.value_counts().plot(kind='bar')
plt.show()
plt.title('Housing')
data.housing.value_counts().plot(kind='bar')
plt.show()
plt.title('Loan')
data.loan.value_counts().plot(kind='bar')
plt.show()
plt.title('Contact')
data.contact.value_counts().plot(kind='bar')
plt.show()
plt.title('Poutcome')
data.poutcome.value_counts().plot(kind='bar')
plt.show()
plt.title('Y')
data.y.value_counts().plot(kind='bar')
plt.show()
```



In [13]:
```
# There is no null value. However there are unknown values as we can see ab
data.isnull().sum()
```

Out[13]:
```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

In [14]:
```
#unknown values
strings = [x for x in data.columns if type(data[x].loc[data[x].first_valid_

for columns in strings:
  print(columns, ':', len(data[data[columns].str.contains('unknown')]))
```

```
job : 288
marital : 0
education : 1857
default : 0
housing : 0
loan : 0
contact : 13020
month : 0
poutcome : 36959
y : 0
```

In [15]:
```
data_copy=data
```

In [16]:
```
#There are 288 unknown in Job column, 1857 in education, 13020 in contact a
data['job'] = data['job'].replace(['unknown'],np.nan)
data['education'] = data['education'].replace(['unknown'],np.nan)
data['contact'] = data['contact'].replace(['unknown'],np.nan)
data['poutcome'] = data['poutcome'].replace(['unknown'],np.nan)
```

In [17]:
```
# method 1 for NA(drop NA)
data=data.dropna()
```

In [18]:
```
data.isnull().mean().sum()
```

Out[18]:
```
0.0
```

In [19]:
```
# method 2 for NA(using mode value to fill NA)
data_copy['job'].fillna(data_copy['job'].mode())
```

Out[19]:
```
0          management
1          technician
2        entrepreneur
3         blue-collar
4                 NaN
             ...
45206      technician
45207         retired
45208         retired
45209     blue-collar
45210    entrepreneur
Name: job, Length: 45211, dtype: object
```

In [20]:
```
data_copy['education'].fillna(data_copy['education'].mode())
```

Out[20]:
```
0         tertiary
1        secondary
2        secondary
3              NaN
4              NaN
           ...
45206     tertiary
45207      primary
45208    secondary
45209    secondary
45210    secondary
Name: education, Length: 45211, dtype: object
```

In [21]:
```
data_copy['contact'].fillna(data_copy['contact'].mode())
data_copy['poutcome'].fillna(data_copy['poutcome'].mode())
```

Out[21]:
```
0        failure
1            NaN
2            NaN
3            NaN
4            NaN
           ...
45206        NaN
45207        NaN
45208    success
45209        NaN
45210      other
Name: poutcome, Length: 45211, dtype: object
```

4. Exploratory Data Report

In [28]:
```
df=data_copy
df.describe()
```

Out[28]:

	age	balance	day	duration	campaign	pdays	
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	0
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	2
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275

In [29]:
```
df.info()
```

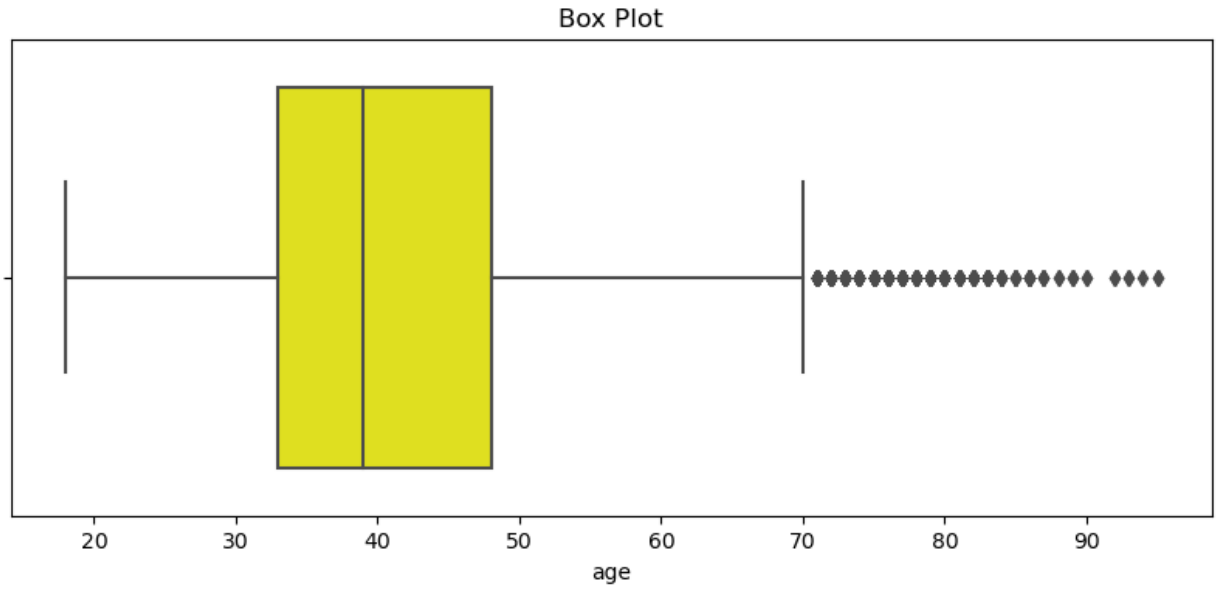
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        45211 non-null  int64 
 1   job        44923 non-null  object
 2   marital    45211 non-null  object
 3   education  43354 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64 
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    32191 non-null  object
 9   day        45211 non-null  int64 
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64 
 12  campaign   45211 non-null  int64 
 13  pdays      45211 non-null  int64 
 14  previous   45211 non-null  int64 
 15  poutcome   8252 non-null   object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

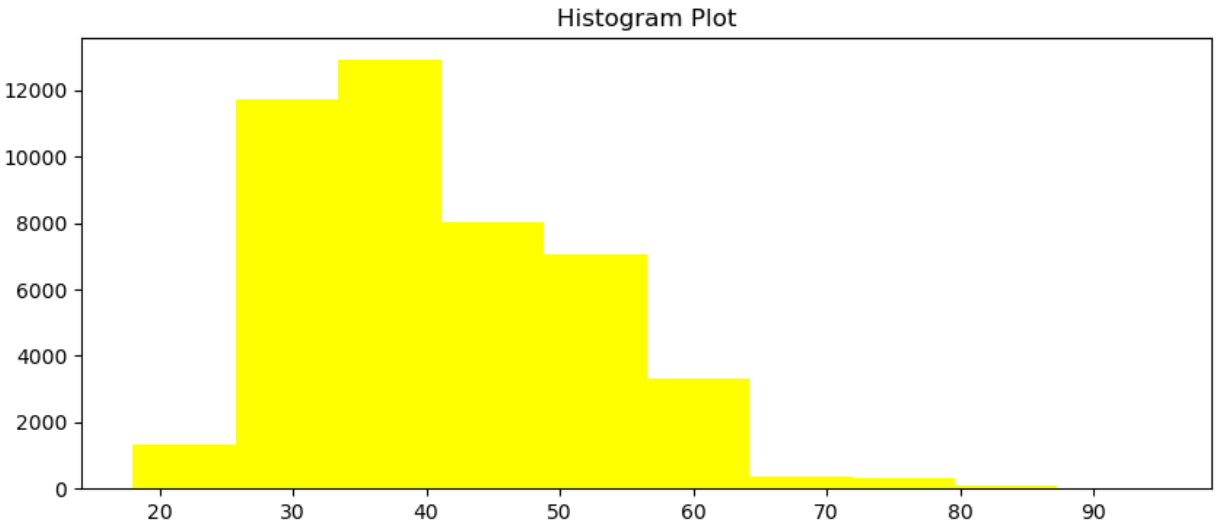
In [27]:
```
def Box_plots(df,clr):
  plt.figure(figsize=(10,4))
  plt.title("Box Plot")
  sns.boxplot(df, color= clr)
  plt.show()

def hist_plots(df,clr):
  plt.figure(figsize=(10,4))
  plt.hist(df, color =clr)
  plt.title("Histogram Plot")
  plt.show()

def dist_plots(df,clr):
  plt.figure(figsize=(10,4))
  plt.title("Distribution Plot")
  sns.distplot(df,color= clr)
  sns.despine()
  plt.show()
```

For age column

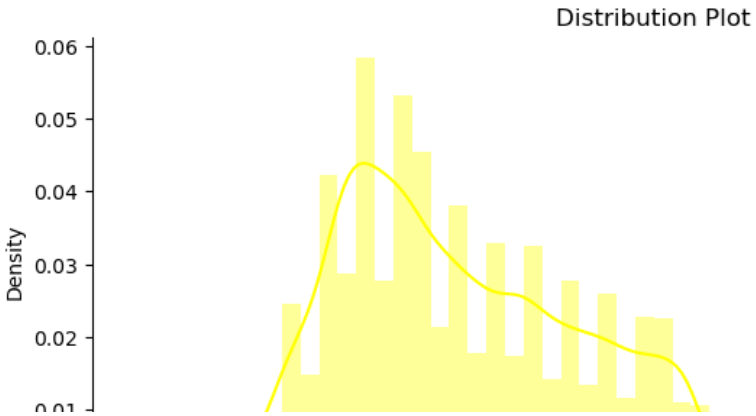
In [30]:
```
Box_plots(df["age"],"yellow")
hist_plots(df["age"],"yellow")
dist_plots(df["age"],"yellow")
```

```
/Users/jinwen/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorato
rs.py:36: FutureWarning: Pass the following variable as a keyword arg: x.
From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```





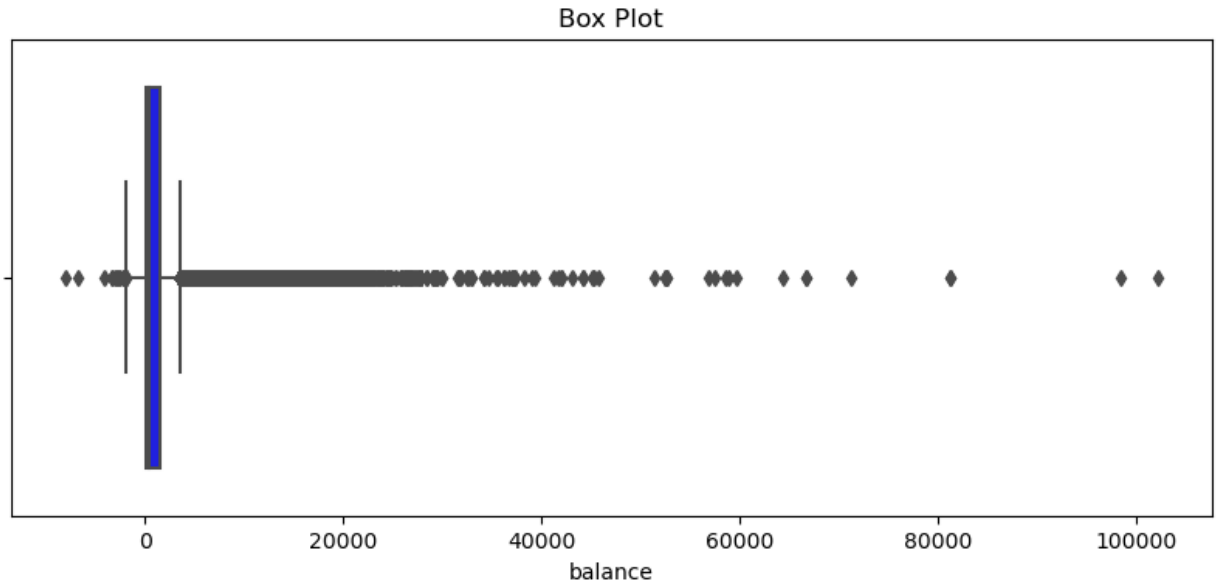

```
/Users/jinwen/opt/anaconda3/lib/python3.9/site-packages/seaborn/distribut
ions.py:2619: FutureWarning: `distplot` is a deprecated function and will
be removed in a future version. Please adapt your code to use either `dis
plot` (a figure-level function with similar flexibility) or `histplot` (a
n axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

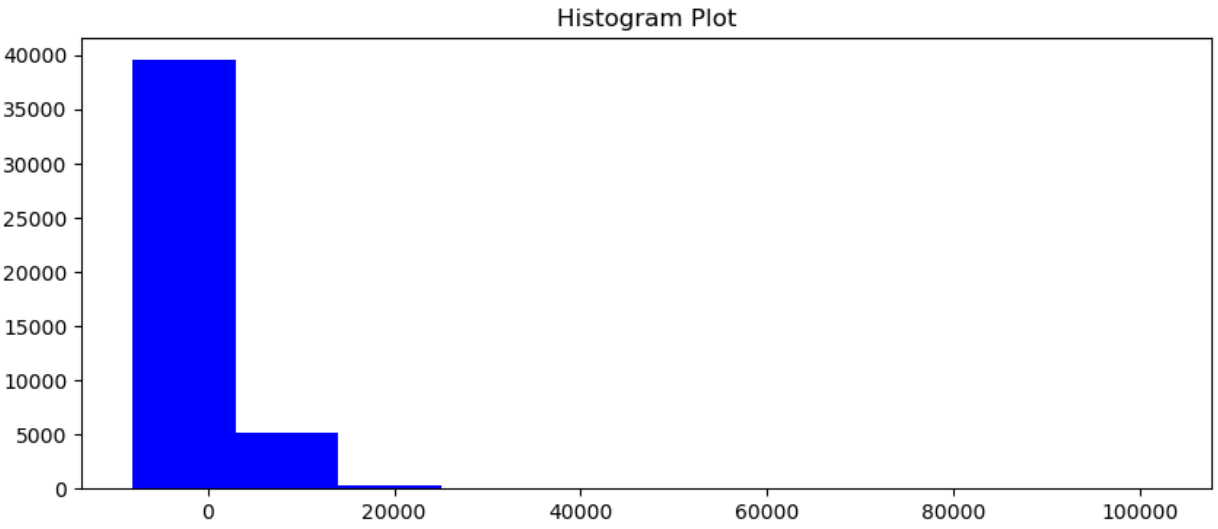


For balance column

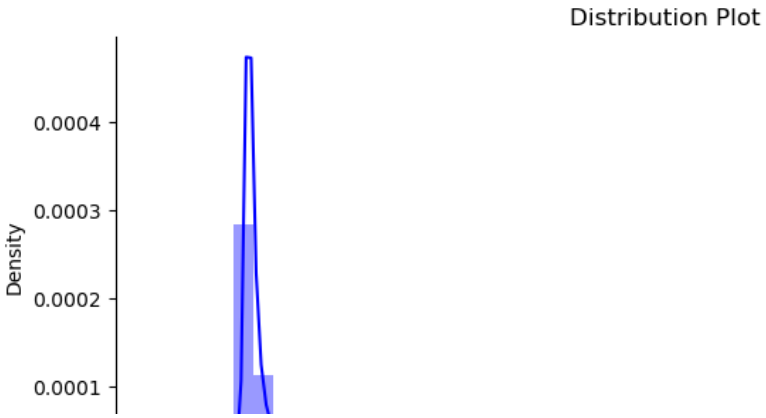
In [31]:
```
Box_plots(df["balance"],"blue")
hist_plots(df["balance"],"blue")
dist_plots(df["balance"],"blue")
```

```
/Users/jinwen/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorato
rs.py:36: FutureWarning: Pass the following variable as a keyword arg: x.
From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```





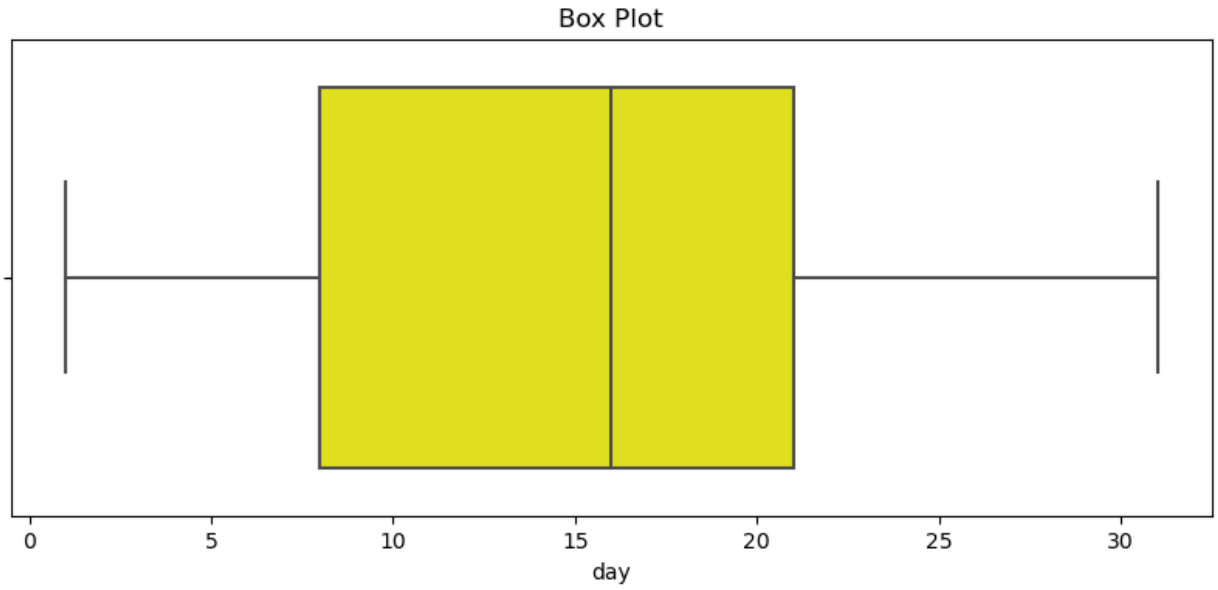

```
/Users/jinwen/opt/anaconda3/lib/python3.9/site-packages/seaborn/distribut
ions.py:2619: FutureWarning: `distplot` is a deprecated function and will
be removed in a future version. Please adapt your code to use either `dis
plot` (a figure-level function with similar flexibility) or `histplot` (a
n axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

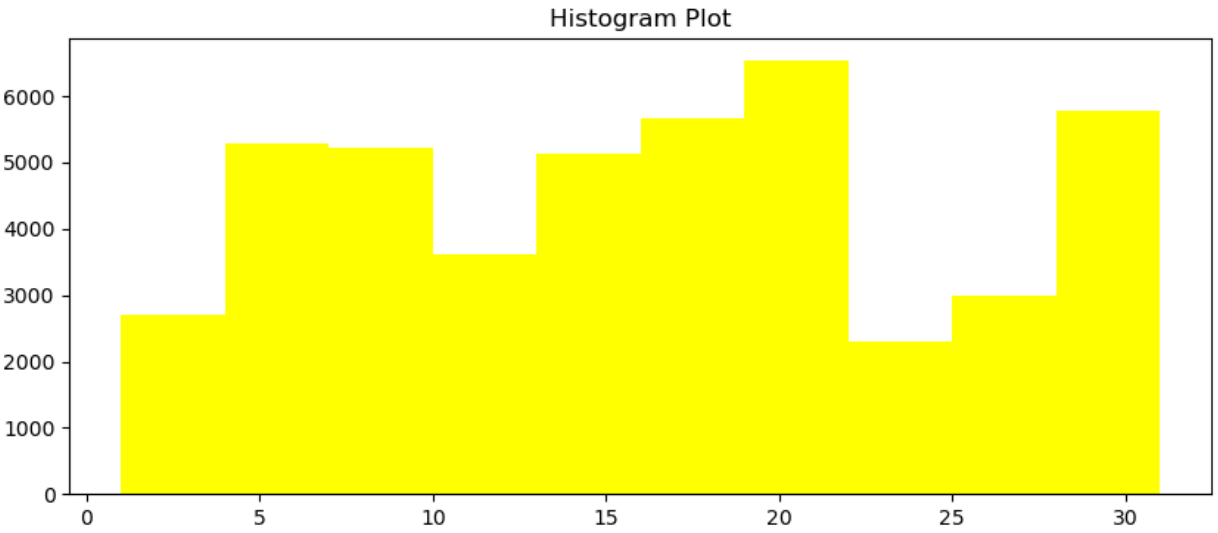


For day column

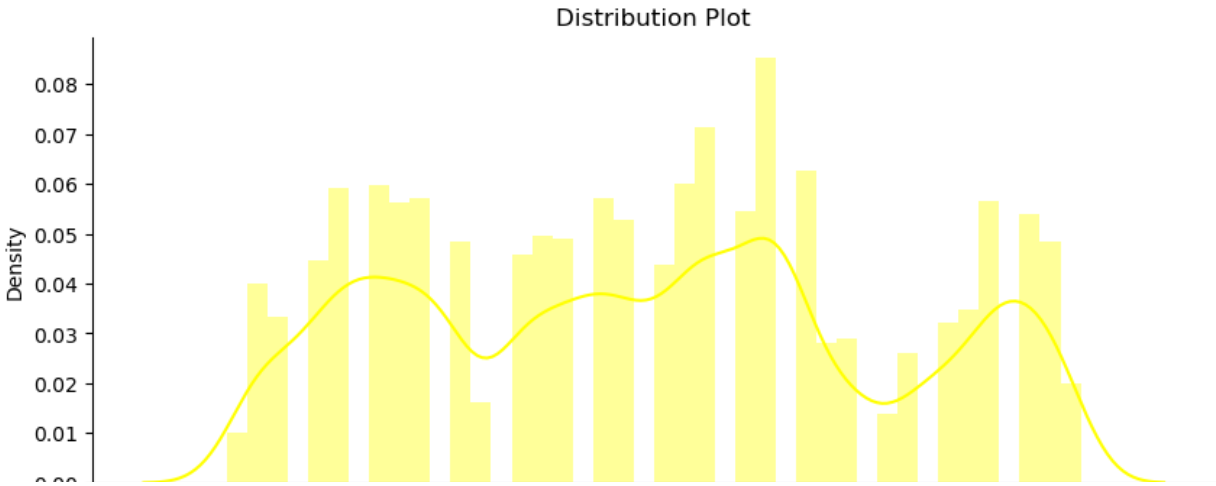
In [33]:
```
Box_plots(df["day"],"yellow")
hist_plots(df["day"],"yellow")
dist_plots(df["day"],"yellow")
```

```
/Users/jinwen/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorato
rs.py:36: FutureWarning: Pass the following variable as a keyword arg: x.
From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```





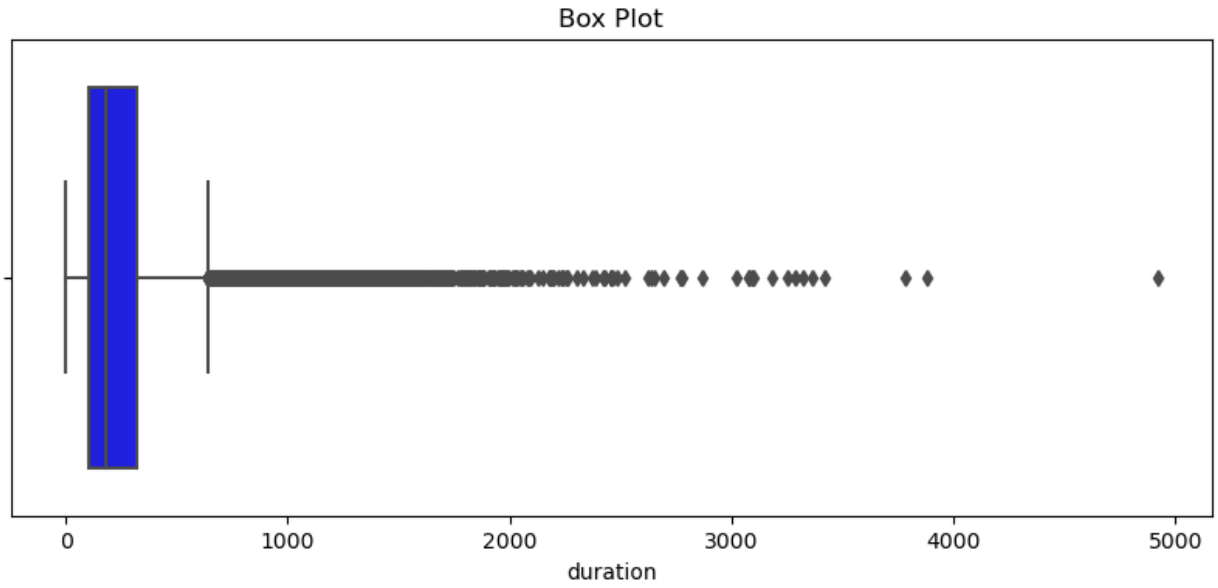

```
/Users/jinwen/opt/anaconda3/lib/python3.9/site-packages/seaborn/distribut
ions.py:2619: FutureWarning: `distplot` is a deprecated function and will
be removed in a future version. Please adapt your code to use either `dis
plot` (a figure-level function with similar flexibility) or `histplot` (a
n axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

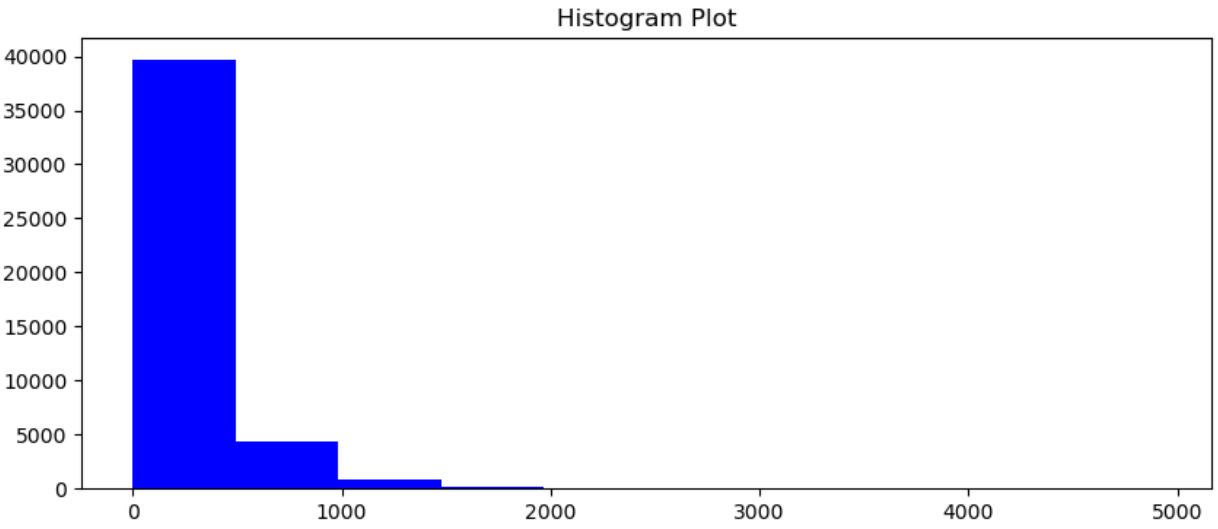


For Duration column

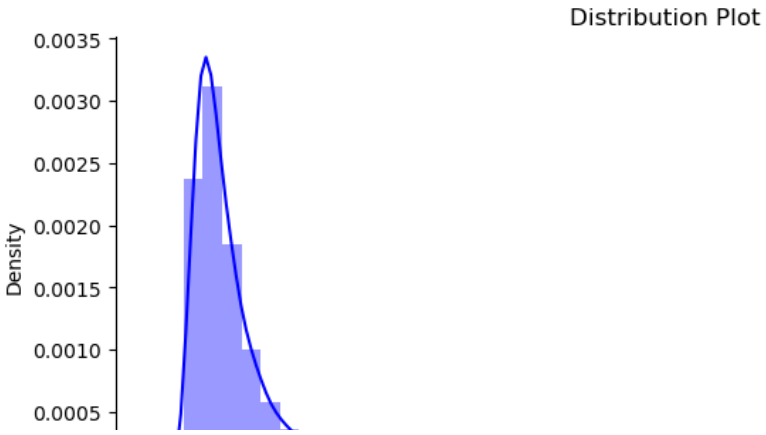
In [35]:
```
Box_plots(df["duration"],"blue")
hist_plots(df["duration"],"blue")
dist_plots(df["duration"],"blue")
```

```
/Users/jinwen/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorato
rs.py:36: FutureWarning: Pass the following variable as a keyword arg: x.
From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```



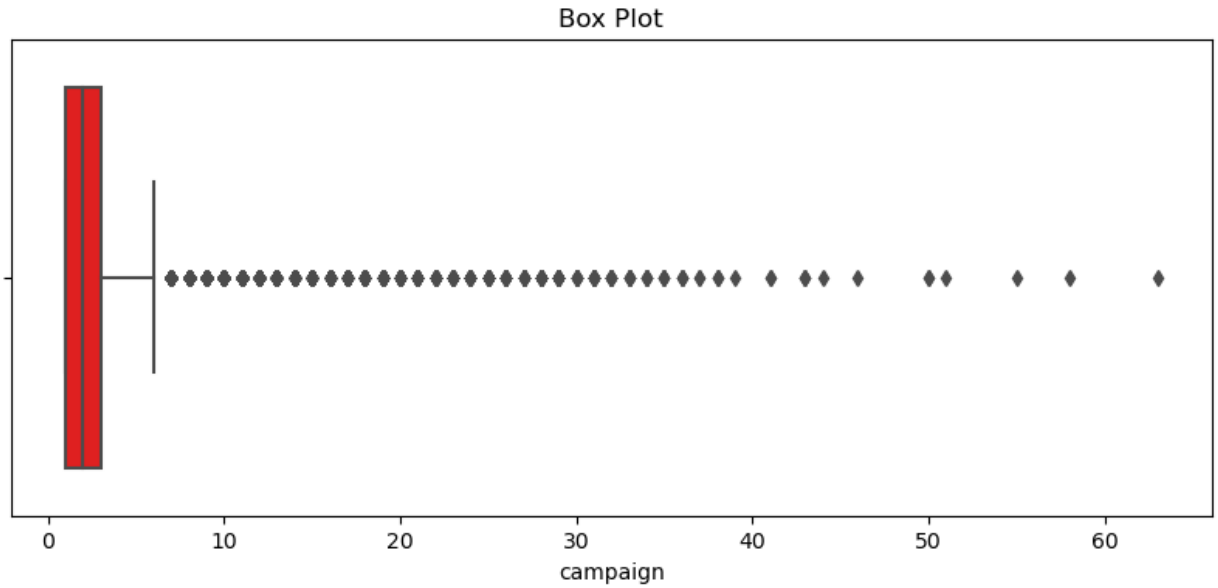


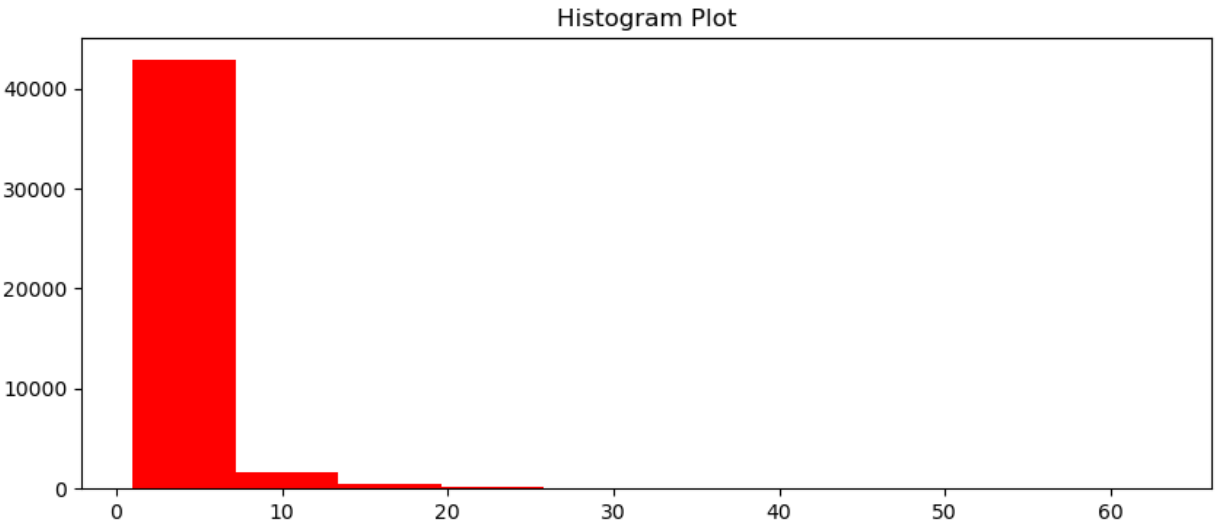

```
/Users/jinwen/opt/anaconda3/lib/python3.9/site-packages/seaborn/distribut
ions.py:2619: FutureWarning: `distplot` is a deprecated function and will
be removed in a future version. Please adapt your code to use either `dis
plot` (a figure-level function with similar flexibility) or `histplot` (a
n axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```


Distribution Plot
Density
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005

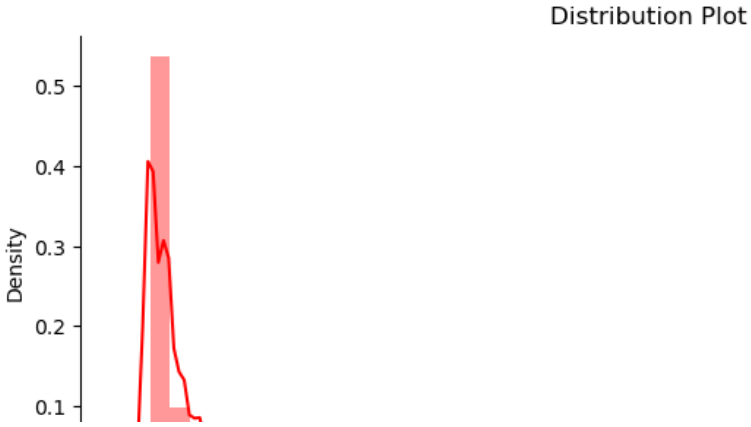
In [37]:
```
Box_plots(df["campaign"],"red")
hist_plots(df["campaign"],"red")
dist_plots(df["campaign"],"red")
```

```
/Users/jinwen/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorato
rs.py:36: FutureWarning: Pass the following variable as a keyword arg: x.
From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```



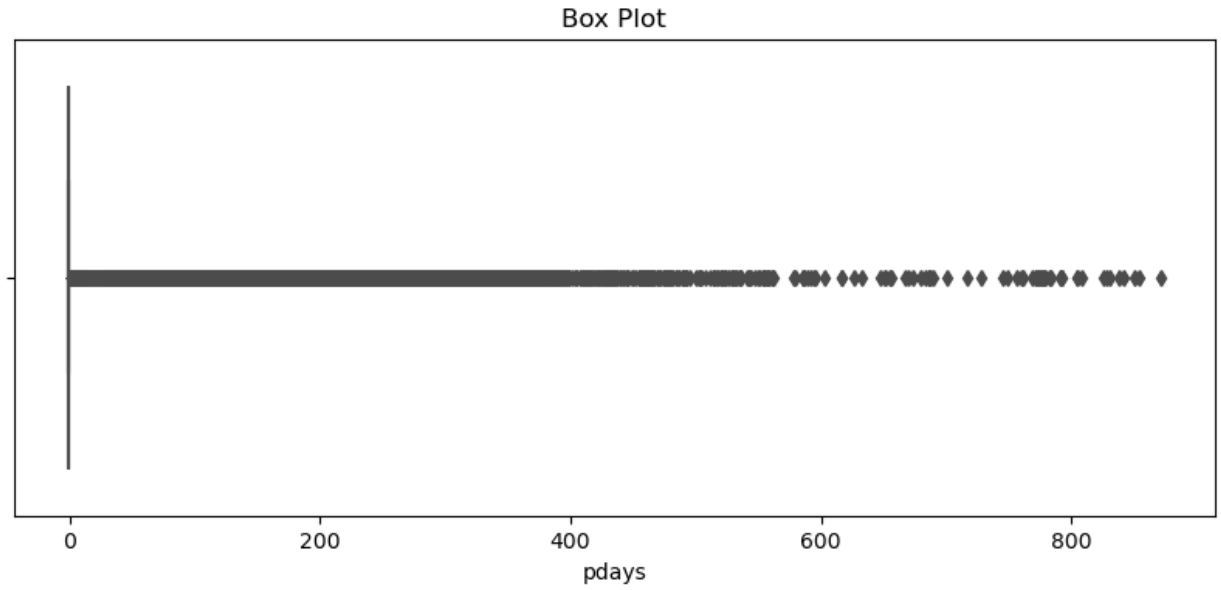


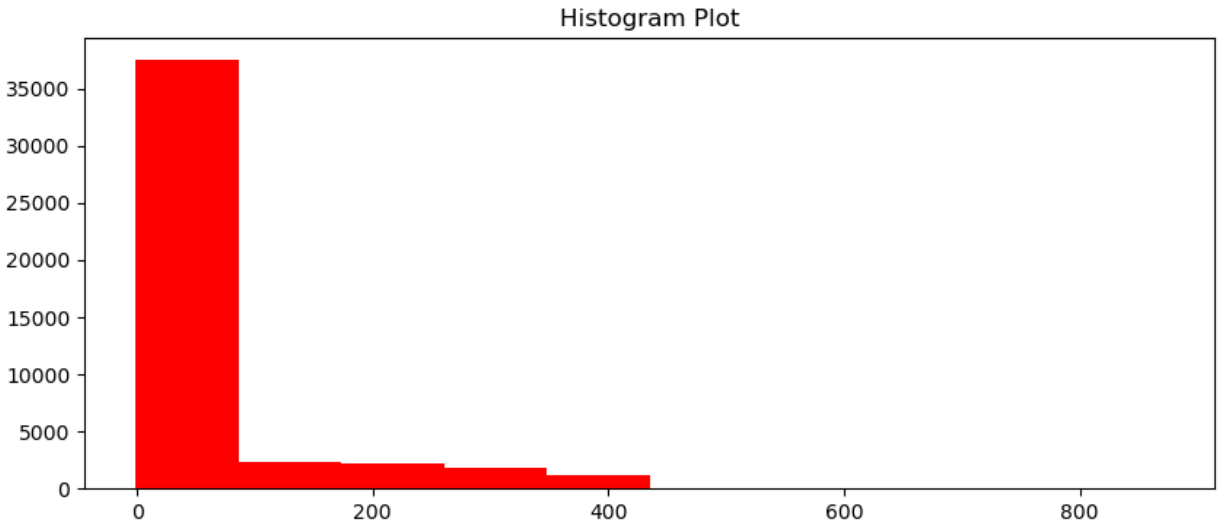

```
/Users/jinwen/opt/anaconda3/lib/python3.9/site-packages/seaborn/distribut
ions.py:2619: FutureWarning: `distplot` is a deprecated function and will
be removed in a future version. Please adapt your code to use either `dis
plot` (a figure-level function with similar flexibility) or `histplot` (a
n axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```


Distribution Plot
Density
0.5
0.4
0.3
0.2
0.1

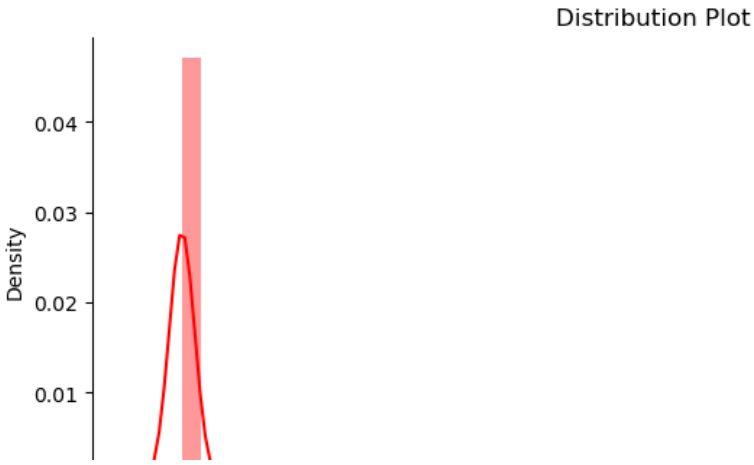
In [38]:
```
Box_plots(df["pdays"],"red")
hist_plots(df["pdays"],"red")
dist_plots(df["pdays"],"red")
```

```
/Users/jinwen/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorato
rs.py:36: FutureWarning: Pass the following variable as a keyword arg: x.
From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```



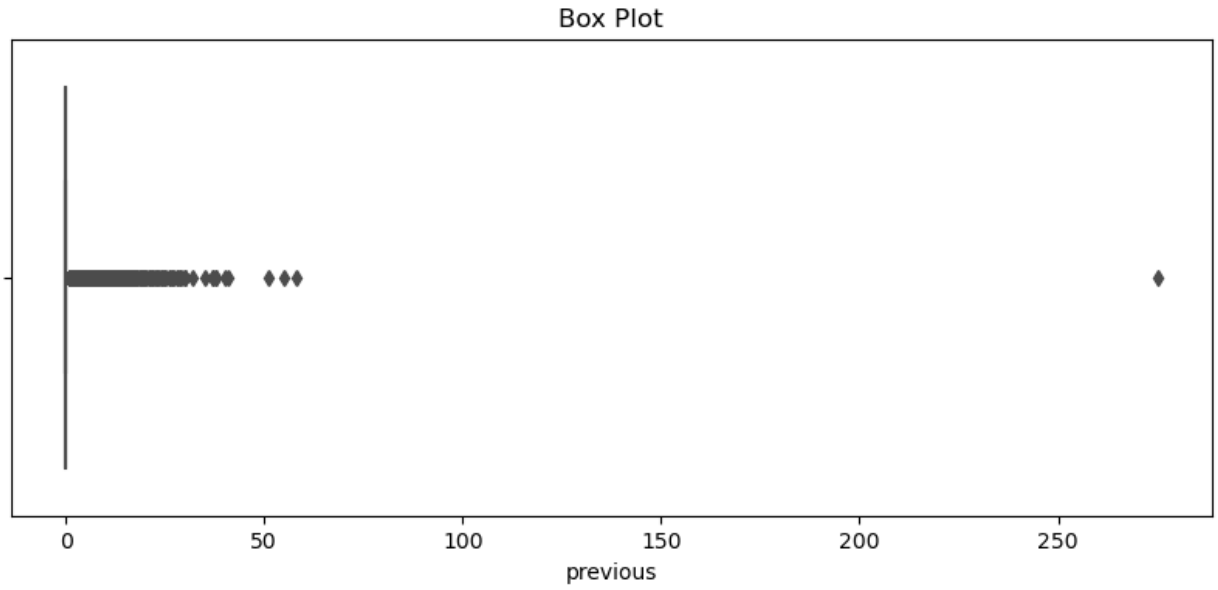


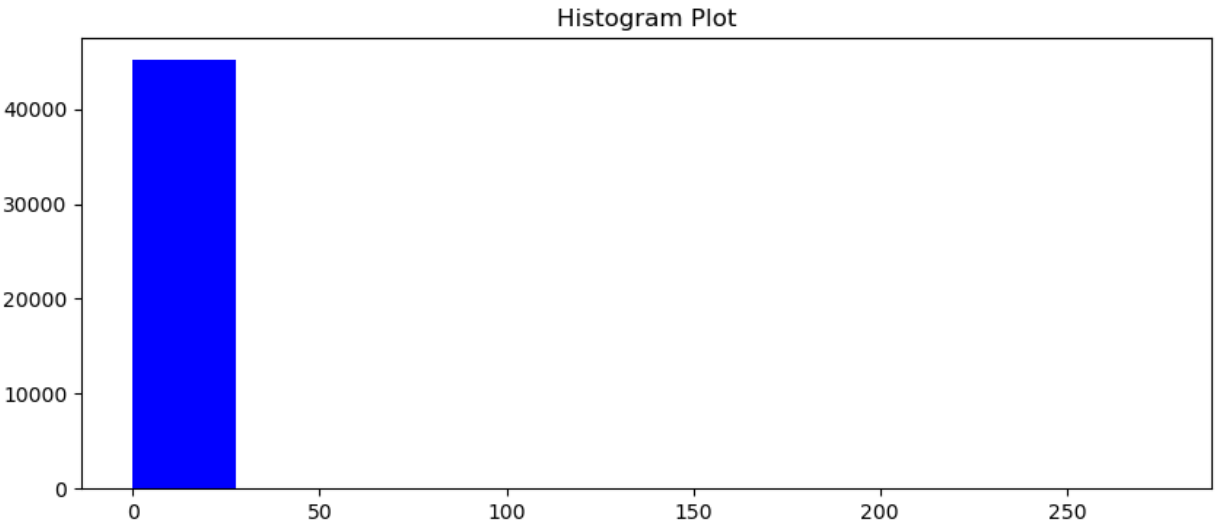

```
/Users/jinwen/opt/anaconda3/lib/python3.9/site-packages/seaborn/distribut
ions.py:2619: FutureWarning: `distplot` is a deprecated function and will
be removed in a future version. Please adapt your code to use either `dis
plot` (a figure-level function with similar flexibility) or `histplot` (a
n axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```


Distribution Plot
Density
0.04
0.03
0.02
0.01

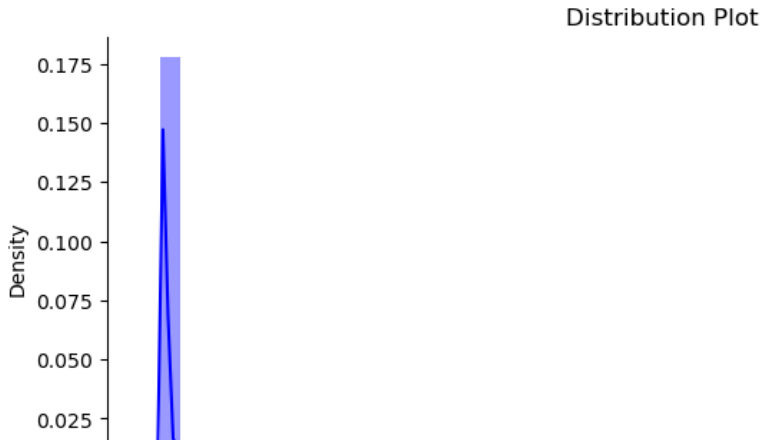
In [39]:
```
Box_plots(df["previous"],"blue")
hist_plots(df["previous"],"blue")
dist_plots(df["previous"],"blue")
```

```
/Users/jinwen/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorato
rs.py:36: FutureWarning: Pass the following variable as a keyword arg: x.
From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```

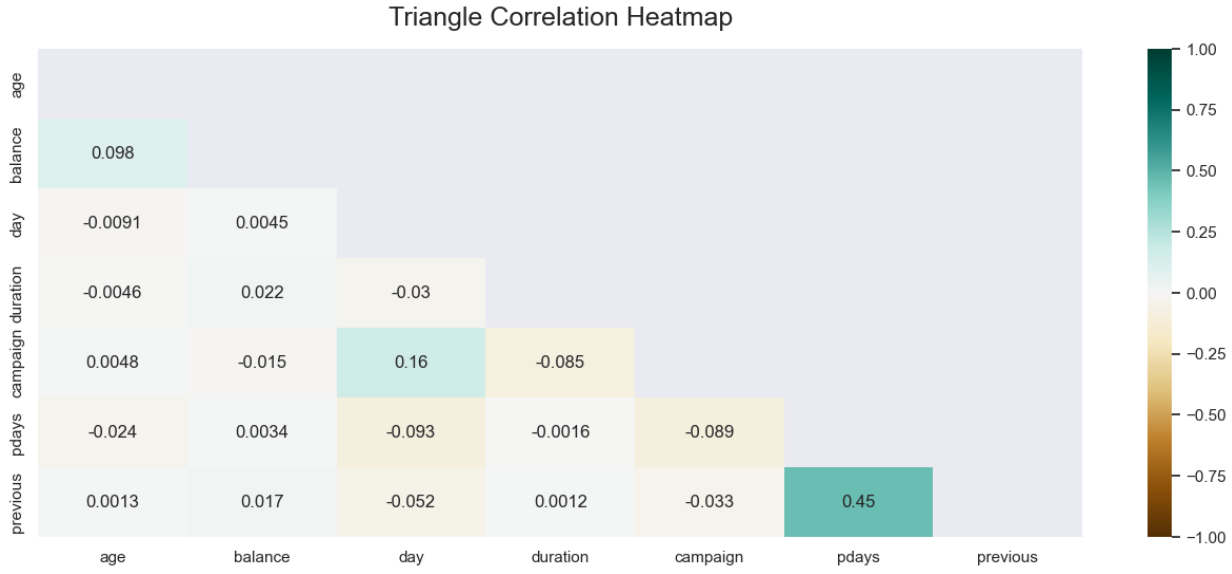





```
/Users/jinwen/opt/anaconda3/lib/python3.9/site-packages/seaborn/distribut
ions.py:2619: FutureWarning: `distplot` is a deprecated function and will
be removed in a future version. Please adapt your code to use either `dis
plot` (a figure-level function with similar flexibility) or `histplot` (a
n axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [59]:
```
plt.figure(figsize=(16, 6))
# define the mask to set the values in the upper triangle to True
mask = np.triu(np.ones_like(df.corr(), dtype=bool))
heatmap = sns.heatmap(df.corr(), mask=mask, vmin=-1, vmax=1, annot=True, cm
heatmap.set_title('Triangle Correlation Heatmap', fontdict={'fontsize':18},
plt.show()
```



Duration Analysis

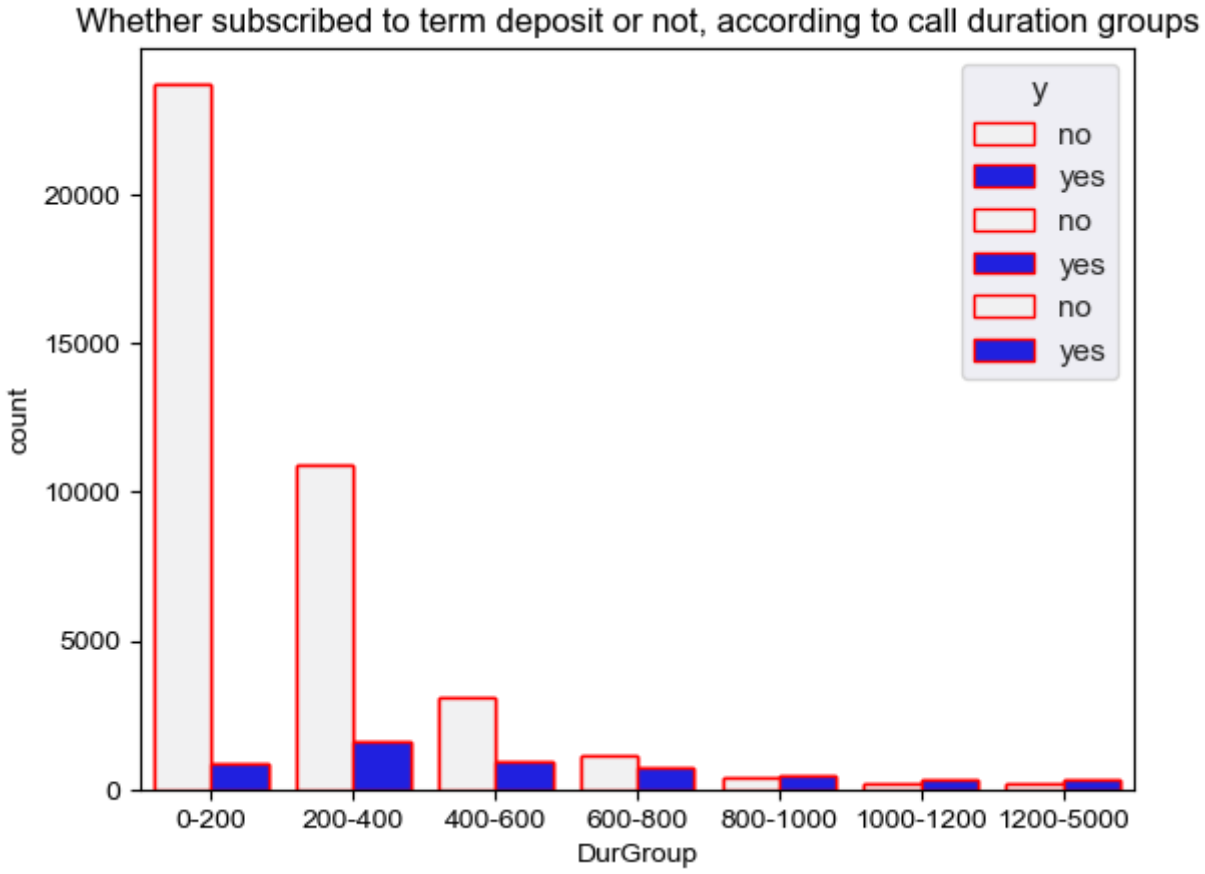
In [45]:
```
bins= [0,200,400,600,800,1000,1200,5000]
labels = ['0-200','200-400','400-600','600-800','800-1000','1000-1200','120
df['DurGroup'] = pd.cut(df['duration'], bins=bins, labels=labels, right=Fal
```

In [46]:
```
df_duration=df.groupby(['y','DurGroup'])["job"].agg(['count'])
df_duration
```

Out[46]:

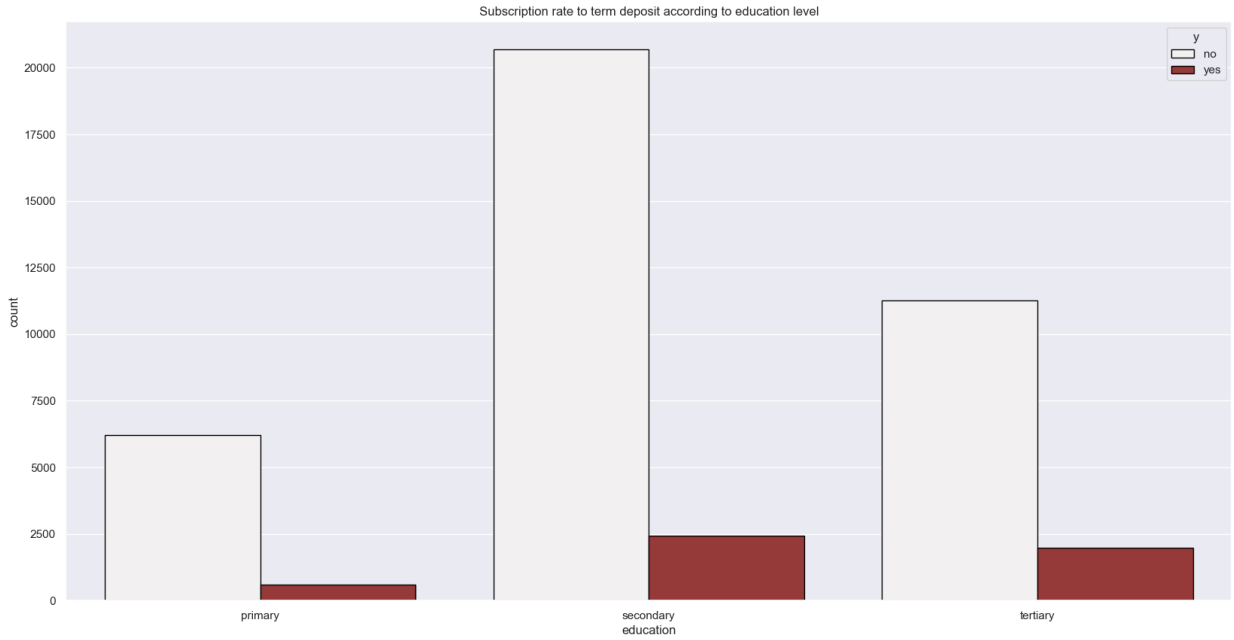
		count
y	**DurGroup**	
no	**0-200**	23718
	200-400	10885
	400-600	3111
	600-800	1125
	800-1000	403
	1000-1200	211
	1200-5000	215
yes	**0-200**	874
	200-400	1596
	400-600	958
	600-800	727
	800-1000	466
	1000-1200	288
	1200-5000	346

In [49]:
```
df_durr=df_duration.reset_index()

plt.title("Whether subscribed to term deposit or not, according to call dur
sns.set(rc = {'figure.figsize':(20,10)})
sns.barplot(data=df_durr,hue="y",x="DurGroup",y="count",color="blue", edgec
plt.show()
```

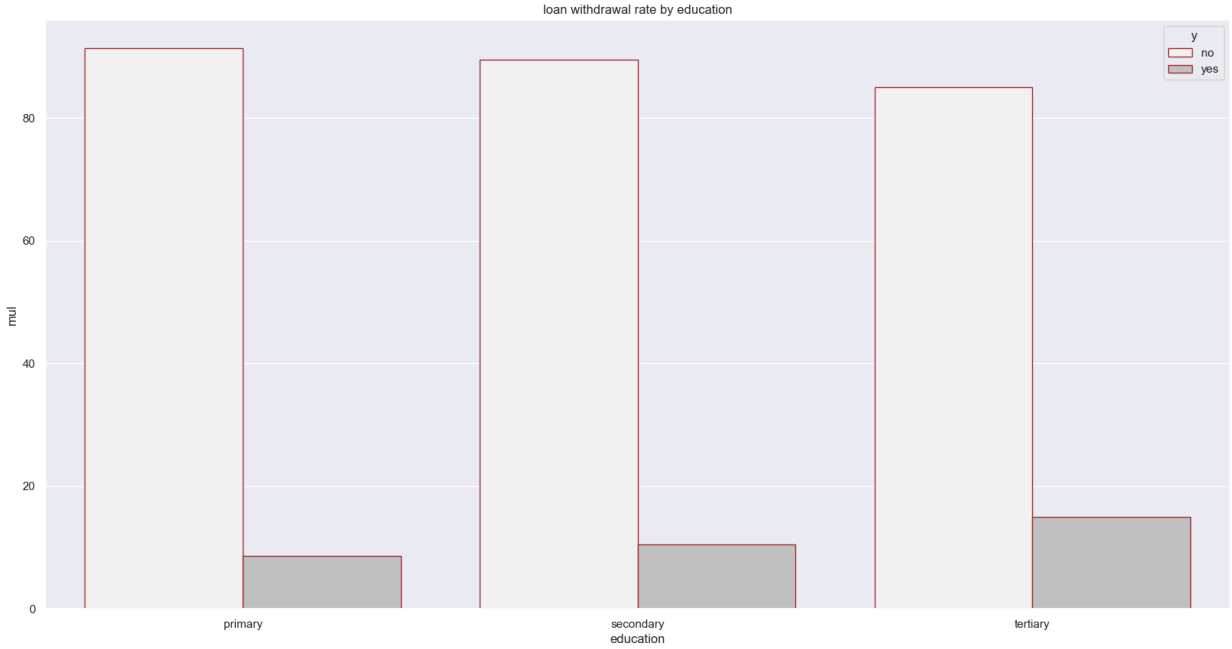


Education Analysis

In [58]:
```
job_count=df.groupby(["y","education"])["job"].agg(["count"])
job_countt=job_count.reset_index()
plt.figure(figsize=(20,10))
plt.title("Subscription rate to term deposit according to education level")
sns.barplot(data=job_countt,hue="y",x="education",y="count",color="brown",
plt.show()
```



In [57]:
```
education_mul=df.groupby("education")["y"].value_counts(normalize=True).mul
education_muldf=pd.DataFrame(education_mul)
education_muldf.rename(columns = {'y':'mul'}, inplace = True)
education_muldf=education_muldf.reset_index()
plt.figure(figsize=(20,10))
plt.title("loan withdrawal rate by education")
sns.barplot(data=education_muldf,hue="y",x="education",y="mul",color="silve
plt.show()
```



Final Recommendation

We can use the eduacation and pdays, duration,balance, age, and job data to make a model for our data to predict the probablity of y.