##**Maestría en Inteligencia Artificial Aplicada**

##**Tecnológico de Monterrey**

##**Curso: Ciencia y Analítica de Datos**

Profesora: Dra. María de la Paz Rico Fernández

Actividad: Semana 7

Nombre de la Actividad: Visualizació

Alumno:Francisco Javier Ramírez Arias

Matricula: A01316379

#Puntos de la Actividad

##1._Descargar los datos y cargar los datos en tu libreta

```
import pandas as pd
PathData =
"https://raw.githubusercontent.com/PosgradoMNA/Actividades_Aprendizaje
-/main/default%20of%20credit%20card%20clients.csv"
df = pd.read_csv(PathData, index_col=0)
```

##2._Obten información del DataFrame con los métodos y propiedades: shape, columns, head(), dtypes, info(), isna()

```
df.shape
```

```
(30000, 24)
```

```
df.columns
```

```
Index(['X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'X9', 'X10',
'X11',
       'X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18', 'X19', 'X20',
'X21',
       'X22', 'X23', 'Y'],
      dtype='object')
```

```
df.head()
```

```
        X1   X2   X3   X4    X5   X6   X7   X8   X9  X10  ...      X15
\
ID                                                        ...

1    20000  2.0  2.0  1.0  24.0  2.0  2.0 -1.0 -1.0 -2.0  ...      0.0

2   120000  2.0  2.0  2.0  26.0 -1.0  2.0  0.0  0.0  0.0  ...   3272.0

3    90000  2.0  2.0  2.0  34.0  0.0  0.0  0.0  0.0  0.0  ...  14331.0
```

```
4    50000  2.0  2.0  1.0  37.0  0.0  0.0  0.0  0.0  0.0  ...  28314.0

5    50000  1.0  2.0  1.0  57.0 -1.0  0.0 -1.0  0.0  0.0  ...  20940.0


        X16      X17     X18      X19      X20     X21     X22     X23
Y
ID

1       0.0      0.0     0.0    689.0      0.0     0.0     0.0     0.0
1.0
2    3455.0   3261.0     0.0   1000.0   1000.0  1000.0     0.0  2000.0
1.0
3   14948.0  15549.0  1518.0   1500.0   1000.0  1000.0  1000.0  5000.0
0.0
4   28959.0  29547.0  2000.0   2019.0   1200.0  1100.0  1069.0  1000.0
0.0
5   19146.0  19131.0  2000.0  36681.0  10000.0  9000.0   689.0   679.0
0.0

[5 rows x 24 columns]

df.dtypes

X1       int64
X2     float64
X3     float64
X4     float64
X5     float64
X6     float64
X7     float64
X8     float64
X9     float64
X10    float64
X11    float64
X12    float64
X13    float64
X14    float64
X15    float64
X16    float64
X17    float64
X18    float64
X19    float64
X20    float64
X21    float64
X22    float64
X23    float64
Y      float64
dtype: object
```

```
df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 30000 entries, 1 to 30000
Data columns (total 24 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   X1      30000 non-null  int64  
 1   X2      29999 non-null  float64
 2   X3      29998 non-null  float64
 3   X4      29998 non-null  float64
 4   X5      29995 non-null  float64
 5   X6      29997 non-null  float64
 6   X7      29995 non-null  float64
 7   X8      29993 non-null  float64
 8   X9      29991 non-null  float64
 9   X10     29984 non-null  float64
 10  X11     29986 non-null  float64
 11  X12     29989 non-null  float64
 12  X13     29989 non-null  float64
 13  X14     29987 non-null  float64
 14  X15     29985 non-null  float64
 15  X16     29983 non-null  float64
 16  X17     29990 non-null  float64
 17  X18     29992 non-null  float64
 18  X19     29991 non-null  float64
 19  X20     29992 non-null  float64
 20  X21     29989 non-null  float64
 21  X22     29989 non-null  float64
 22  X23     29995 non-null  float64
 23  Y       29997 non-null  float64
dtypes: float64(23), int64(1)
memory usage: 5.7 MB

df.isna()

          X1     X2     X3     X4     X5     X6     X7     X8     X9
X10  \
ID

1      False  False  False  False  False  False  False  False  False
False
2      False  False  False  False  False  False  False  False  False
False
3      False  False  False  False  False  False  False  False  False
False
4      False  False  False  False  False  False  False  False  False
False
5      False  False  False  False  False  False  False  False  False
False
...      ...    ...    ...    ...    ...    ...    ...    ...    ...
```

```
...
29996  False  False  False  False  False  False  False  False  False
False
29997  False  False  False  False  False  False  False  False  False
False
29998  False  False  False  False  False  False  False  False  False
False
29999  False  False  False  False  False  False  False  False  False
False
30000  False  False  False  False  False  False  False  False  False
False

       ...    X15    X16    X17    X18    X19    X20    X21    X22
X23  \
ID     ...

1      ...  False  False  False  False  False  False  False  False
False
2      ...  False  False  False  False  False  False  False  False
False
3      ...  False  False  False  False  False  False  False  False
False
4      ...  False  False  False  False  False  False  False  False
False
5      ...  False  False  False  False  False  False  False  False
False
...    ...    ...    ...    ...    ...    ...    ...    ...    ...
...
29996  ...  False  False  False  False  False  False  False  False
False
29997  ...  False  False  False  False  False  False  False  False
False
29998  ...  False  False  False  False  False  False  False  False
False
29999  ...  False  False  False  False  False  False  False  False
False
30000  ...  False  False  False  False  False  False  False  False
False

           Y
ID
1      False
2      False
3      False
4      False
5      False
...      ...
29996  False
29997  False
29998  False
```

```
29999  False
30000  False

[30000 rows x 24 columns]
```

##3._ Limpia los datos eliminando los registros nulos o rellena con la media de la columna

```
#Revisamos si las diferentes columnas tiene valores nulos
df.isnull().any()
```

```
X1     False
X2      True
X3      True
X4      True
X5      True
X6      True
X7      True
X8      True
X9      True
X10     True
X11     True
X12     True
X13     True
X14     True
X15     True
X16     True
X17     True
X18     True
X19     True
X20     True
X21     True
X22     True
X23     True
Y       True
dtype: bool
```

```
df_clean = df.dropna()
```

```
df_clean.isnull().any()
```

```
X1     False
X2     False
X3     False
X4     False
X5     False
X6     False
X7     False
X8     False
X9     False
X10    False
X11    False
```

```
X12    False
X13    False
X14    False
X15    False
X16    False
X17    False
X18    False
X19    False
X20    False
X21    False
X22    False
X23    False
Y      False
dtype: bool
```

##4._Calcula la estadistica descriptiva con describe(), y explica las medidas de tendencia central y dispersión

```
df_clean.describe()
```

```
                  X1            X2            X3            X4
X5  \
count    29958.000000  29958.000000  29958.000000  29958.000000
29958.000000
mean    167555.900928      1.604012      1.853094      1.551739
35.483443
std     129737.299088      0.489070      0.790471      0.521952
9.214319
min      10000.000000      1.000000      0.000000      0.000000
21.000000
25%      50000.000000      1.000000      1.000000      1.000000
28.000000
50%     140000.000000      2.000000      2.000000      2.000000
34.000000
75%     240000.000000      2.000000      2.000000      2.000000
41.000000
max    1000000.000000      2.000000      6.000000      3.000000
79.000000

                 X6            X7            X8            X9
X10  \
count  29958.000000  29958.000000  29958.000000  29958.000000
29958.000000
mean      -0.017124     -0.134021     -0.166767     -0.221110     -
0.266807
std        1.123989      1.197171      1.196026      1.168419
1.132307
min       -2.000000     -2.000000     -2.000000     -2.000000     -
2.000000
25%       -1.000000     -1.000000     -1.000000     -1.000000     -
1.000000
```

```
50%        0.000000      0.000000      0.000000      0.000000
0.000000
75%        0.000000      0.000000      0.000000      0.000000
0.000000
max        8.000000      8.000000      8.000000      8.000000
8.000000

       ...            X15           X16           X17           X18
\
count  ...   29958.000000   29958.000000   29958.000000   29958.000000

mean   ...   43279.335370   40328.984578   38889.925763    5664.614460

std    ...   64364.684347   60826.219326   59582.883301   16568.823518

min    ... -170000.000000  -81334.000000 -339603.000000       0.000000

25%    ...    2327.500000    1762.250000    1256.000000    1000.000000

50%    ...   19037.500000   18104.500000   17067.500000    2100.000000

75%    ...   54551.250000   50220.750000   49234.750000    5007.000000

max    ...  891586.000000  927171.000000  961664.000000  873552.000000


                X19           X20           X21           X22  \
count  2.995800e+04   29958.000000   29958.000000   29958.000000
mean   5.925715e+03    5228.429969    4829.873556    4801.481574
std    2.305598e+04   17617.338167   15676.205514   15285.552652
min    0.000000e+00       0.000000       0.000000       0.000000
25%    8.352500e+02     390.000000     296.250000     253.250000
50%    2.009000e+03    1800.000000    1500.000000    1500.000000
75%    5.000000e+03    4511.500000    4014.750000    4040.000000
max    1.684259e+06  896040.000000  621000.000000  426529.000000

                X23            Y
count   29958.000000  29958.000000
mean     5220.708025      0.221143
std     17788.983767      0.415023
min         0.000000      0.000000
25%       118.000000      0.000000
50%      1500.000000      0.000000
75%      4000.000000      0.000000
max    528666.000000      1.000000

[8 rows x 24 columns]
```

Count nos indica la cuenta de las observaciones, meen indica el promedio de los elementos que integran la columna. La desviación estandar nos indica la variación o disperción del conjunto de datos que integran la columna. Min, es el valor minimo encontrado en la columna, Max, es el valor maximo encontrado en la columna de datos. Los cuartiles basicamente nos permiten dividir o separar la muestra en cuatro partes iguales. Entre cuartil y cuartil se delimita un 25%.. El cuartil de 25%, nos indica que el 25% de los datos de la columna se encuentran por debajo de 50000, el cualtil de 50%, se encuentra localizado en el valor de 140000, lo que nos indica que el 50% de los datos esta por debajo de 140000, el cuartil de 75%, nos indica que el 75% de los datos de la columna se encuentran por arriba de ese valor.

##5._Realiza el conteo de las variables categóricas

```
df_clean['X2'].value_counts()
```

```
2.0    18095
1.0    11863
Name: X2, dtype: int64
```

```
df_clean['X3'].value_counts()
```

```
2.0    14009
1.0    10572
3.0     4909
5.0      280
4.0      123
6.0       51
0.0       14
Name: X3, dtype: int64
```

```
df_clean['X4'].value_counts()
```

```
2.0    15939
1.0    13643
3.0      322
0.0       54
Name: X4, dtype: int64
```

```
df_clean['X5'].value_counts()
```

```
29.0    1601
27.0    1475
28.0    1408
30.0    1393
26.0    1255
31.0    1216
25.0    1184
34.0    1161
32.0    1156
33.0    1145
24.0    1123
```

```
35.0    1113
36.0    1108
37.0    1040
39.0     954
38.0     943
23.0     931
40.0     868
41.0     820
42.0     794
44.0     700
43.0     669
45.0     617
46.0     569
22.0     559
47.0     498
48.0     466
49.0     450
50.0     411
51.0     339
53.0     325
52.0     304
54.0     246
55.0     208
56.0     177
58.0     122
57.0     122
59.0      83
60.0      67
21.0      67
61.0      56
62.0      44
64.0      31
63.0      31
66.0      25
65.0      24
67.0      16
69.0      15
70.0      10
68.0       5
73.0       4
72.0       3
71.0       3
75.0       2
79.0       1
74.0       1
Name: X5, dtype: int64
```

##6._Escala los datos, si lo consideras necesario

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df_scaled = scaler.fit_transform(df_clean)
df_scaled
```

```
array([[0.01010101, 1.        , 0.33333333, ..., 0.        , 0.
,
        1.        ],
       [0.11111111, 1.        , 0.33333333, ..., 0.        ,
0.00378311,
        1.        ],
       [0.08080808, 1.        , 0.33333333, ..., 0.00234451,
0.00945777,
        0.        ],
       ...,
       [0.02020202, 0.        , 0.33333333, ..., 0.00468901,
0.00586382,
        1.        ],
       [0.07070707, 0.        , 0.5       , ..., 0.12417444,
0.00341236,
        1.        ],
       [0.04040404, 0.        , 0.33333333, ..., 0.00234451,
0.00189155,
        1.        ]])
```

```
df_scaled.shape
```

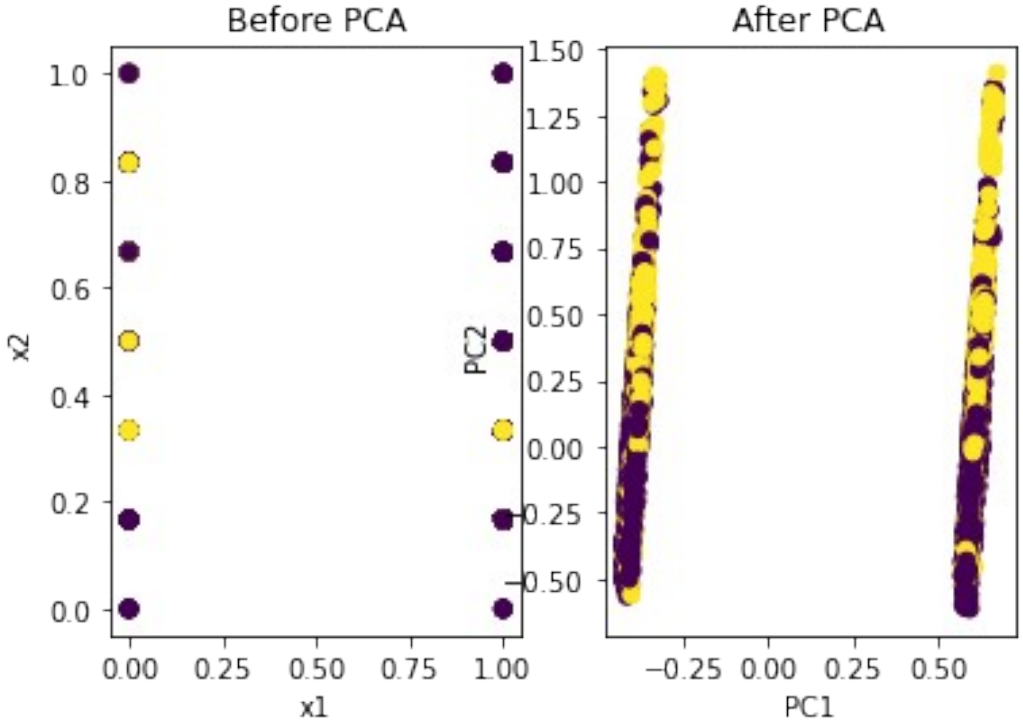
```
(29958, 24)
```

##7._Reduce las dimensiones con PCA, si consideras necesario

1. Indica la varianza de los datos explicada por cada componente seleccionado. Para actividades de exploración de los dato la varianza > 70%.
2. Indica la importancia de las variables en cada componente

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.decomposition import PCA
import pandas as pd
from sklearn.preprocessing import StandardScaler

X = df_scaled[:,0:22]
y = df_scaled[:,23]
pca = PCA(n_components=5) # Solo se estiman 2 componentes principales.
X_new = pca.fit_transform(X)

fig, axes = plt.subplots(1,2)
axes[0].scatter(X[:,1], X[:,2], c=y)
axes[0].set_xlabel('x1')
axes[0].set_ylabel('x2')
```

```
axes[0].set_title('Before PCA')
axes[1].scatter(X_new[:,0], X_new[:,1], c=y)
axes[1].set_xlabel('PC1')
axes[1].set_ylabel('PC2')
axes[1].set_title('After PCA')
plt.show()
```




```
print(pca.explained_variance_ratio_)
```

```
[0.55431122 0.14394135 0.09414729 0.0552026  0.03848222]
```

```
np.cov(X_new.T)
```

```
array([[ 2.39859379e-01,  1.48242217e-19, -1.58619172e-18,
        -1.46759795e-18,  1.09421287e-18],
       [ 1.48242217e-19,  6.22857348e-02,  5.40713487e-18,
        -8.80188164e-19, -1.22299829e-19],
       [-1.58619172e-18,  5.40713487e-18,  4.07390482e-02,
        -8.91306330e-19,  3.22426822e-18],
       [-1.46759795e-18, -8.80188164e-19, -8.91306330e-19,
         2.38870528e-02,  1.06827048e-17],
       [ 1.09421287e-18, -1.22299829e-19,  3.22426822e-18,
         1.06827048e-17,  1.66518771e-02]])
```

```
pca.explained_variance_
```

```
array([0.23985938, 0.06228573, 0.04073905, 0.02388705, 0.01665188])
```

```
print(abs( pca.components_ ))
```

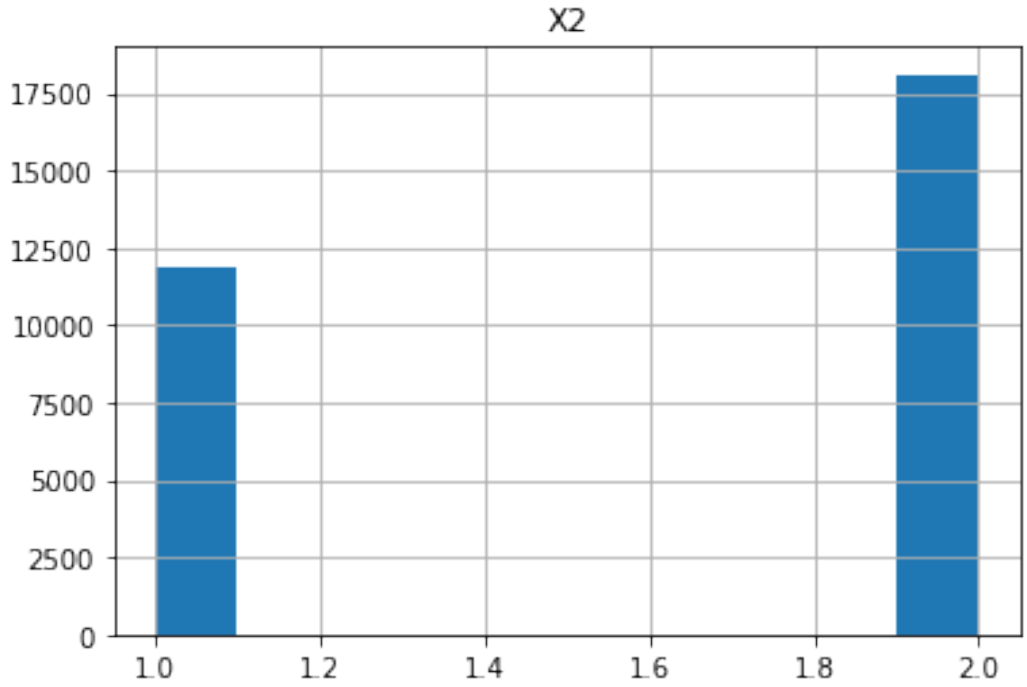
```
[[8.43876784e-03 9.98288497e-01 2.92534647e-03 1.14599422e-02
  3.17804595e-02 1.71701735e-02 2.21783826e-02 2.10504817e-02
  1.91202070e-02 1.71783773e-02 1.45729403e-02 5.52173110e-03
  5.44634421e-03 2.55394169e-03 3.74727632e-03 3.13548694e-03
  2.33461166e-03 3.29164175e-06 2.14882941e-05 3.16734751e-04
  7.66386711e-05 7.06100455e-05]
 [2.08033649e-01 4.44151727e-02 8.92665857e-02 7.75133626e-02
  9.28747616e-02 3.22961373e-01 4.02782498e-01 4.16784642e-01
  4.11832091e-01 3.91929787e-01 3.76143661e-01 7.08824069e-02
  7.89996582e-02 4.48597925e-02 7.35104252e-02 7.35365481e-02
  5.46785630e-02 2.68615383e-03 2.48226620e-03 4.05094038e-03
  5.34547404e-03 7.97754612e-03]
 [1.06340641e-01 1.51015070e-02 2.01421851e-01 7.45973129e-01
  6.05913397e-01 3.72563160e-02 4.61406550e-02 4.38651217e-02
  4.52123232e-02 4.10472553e-02 3.98730698e-02 5.39887755e-02
  5.61872830e-02 3.19064886e-02 5.04197207e-02 4.98964000e-02
  3.63696622e-02 3.63555978e-03 2.18665997e-03 3.56119196e-03
  4.62518034e-03 5.42350759e-03]
 [6.38198448e-01 5.37581303e-03 3.79439363e-01 1.00256342e-01
  2.65938751e-03 2.42899138e-02 5.55782965e-03 1.01806971e-02
  3.24426752e-02 5.29425382e-02 6.53800873e-02 2.95568792e-01
  3.11456856e-01 1.75600683e-01 2.86733588e-01 2.84131561e-01
  2.10909839e-01 3.13969268e-02 1.83509277e-02 3.27128708e-02
  3.89716617e-02 5.69516457e-02]
 [2.27133348e-02 2.05698706e-02 5.45494588e-01 6.05887985e-01
  5.34067538e-01 2.32365925e-02 2.98820566e-02 3.54879066e-02
  3.59250169e-02 3.60930986e-02 3.32042810e-02 1.02190548e-01
  1.05214669e-01 5.65188711e-02 8.64223242e-02 8.12033091e-02
  6.04448564e-02 7.57584721e-03 3.98434756e-03 8.03279871e-03
  7.19219300e-03 1.29161527e-02]]
```

Podemos observar en la variable de relación de varianza explicada, con solo dos componentes principales cubrimos alrededor del 70% de la información de relevancia, la cual nos permitirá realizar una exploración de los datos de una forma más relevante. Para el componente principal #1, la variable de edad es la que tiene mayor relevancia. En el componente principal #2, encontramos que la variable de mayor relevancia es el historial de pagos. Los datos obtenidos del componente principal #3, nos indican que la variable de mayor relevancia es el estado civil. El, componte principal #4, nos indica que la cantidad de crédito solicitado es la variable de mayor relevancia, y por último, pero no menos importante el componente principal #5 nos indica que el estado civil es la variable de mayor relevancia. Podemos observar que el estado civil se encuentra repetido, en dos componentes principales. También observamos que el 70% de la información de relevancia se encuentra contenida en el componente principal 1 y 2, en los cuales las variables relevantes son la edad y el historial de crédito. Lo cual suena lógico al momento de solicitar algún tipo de crédito.

##8._Elabora los histogramas de los atributos para visualizar su distribución.

```
import pandas as pd
import matplotlib.pyplot as plt
df_clean.hist(column='X2')
```

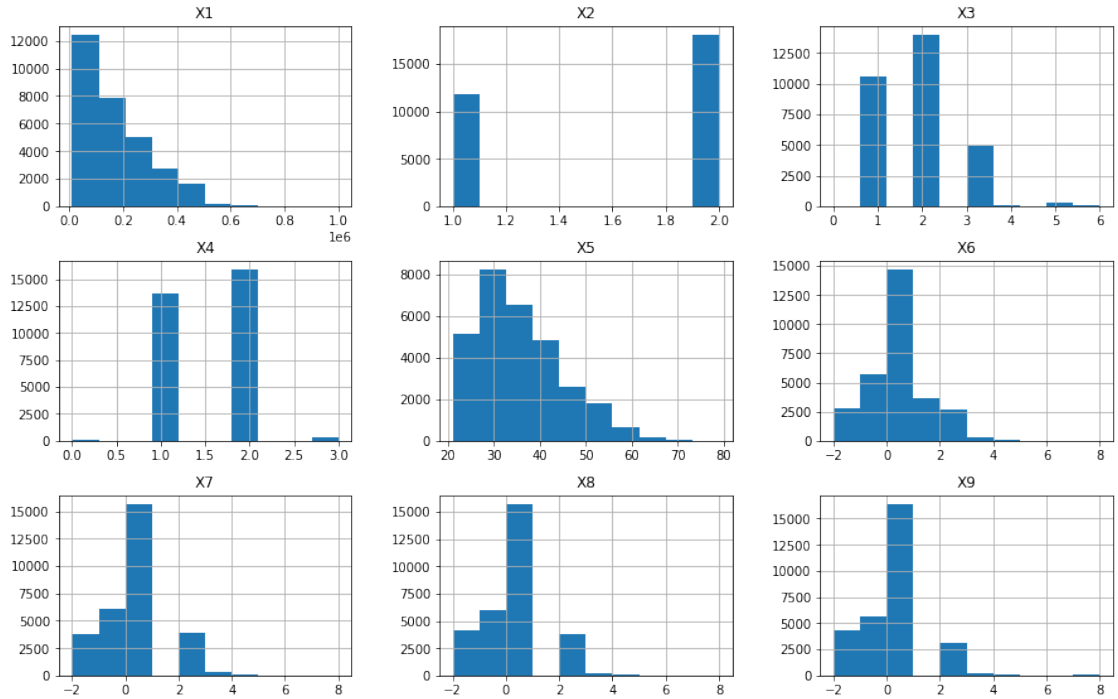
```
array([[<matplotlib.axes._subplots.AxesSubplot object at
0x7f8116069450>]],
      dtype=object)
```




```
df_to_hist=df_clean.drop(['X10','X11','X12','X13','X14','X15','X16','X
17','X18','X19','X20','X21','X22','X23','Y'], axis=1)
df_to_hist.hist(figsize=(15,30),layout=(9,3))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at
0x7f8115831dd0>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f81157ed3d0>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f8115822790>],
       [<matplotlib.axes._subplots.AxesSubplot object at
0x7f81157dac90>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f8115792f90>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f81157534d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at
0x7f81157099d0>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f81156c3ed0>,
        <matplotlib.axes._subplots.AxesSubplot object at
```

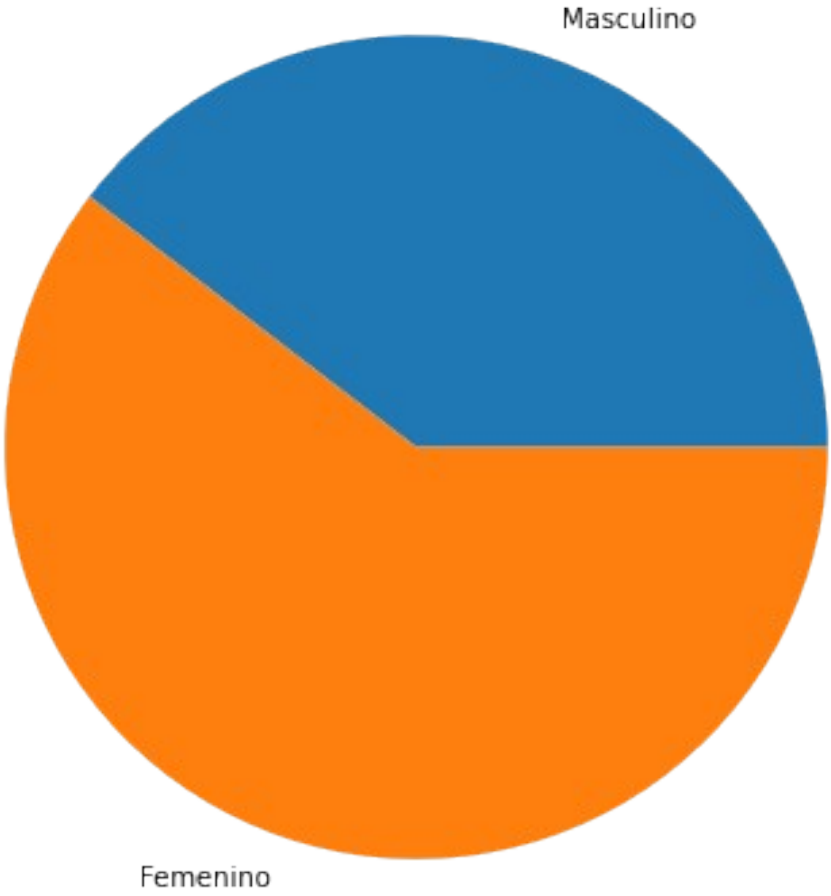
```
0x7f81156df7d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at
0x7f8115697e10>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f81155f3e10>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f81155b4350>],
       [<matplotlib.axes._subplots.AxesSubplot object at
0x7f81155e9850>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f81155a2d50>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f8115565290>],
       [<matplotlib.axes._subplots.AxesSubplot object at
0x7f811551a790>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f81154d4c90>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f81154961d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at
0x7f811544f6d0>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f8115405bd0>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f81153c5110>],
       [<matplotlib.axes._subplots.AxesSubplot object at
0x7f811537d610>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f8115333b10>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f81152f4050>],
       [<matplotlib.axes._subplots.AxesSubplot object at
0x7f81152ac550>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f81152e3a50>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x7f811529cf50>]],
      dtype=object)
```



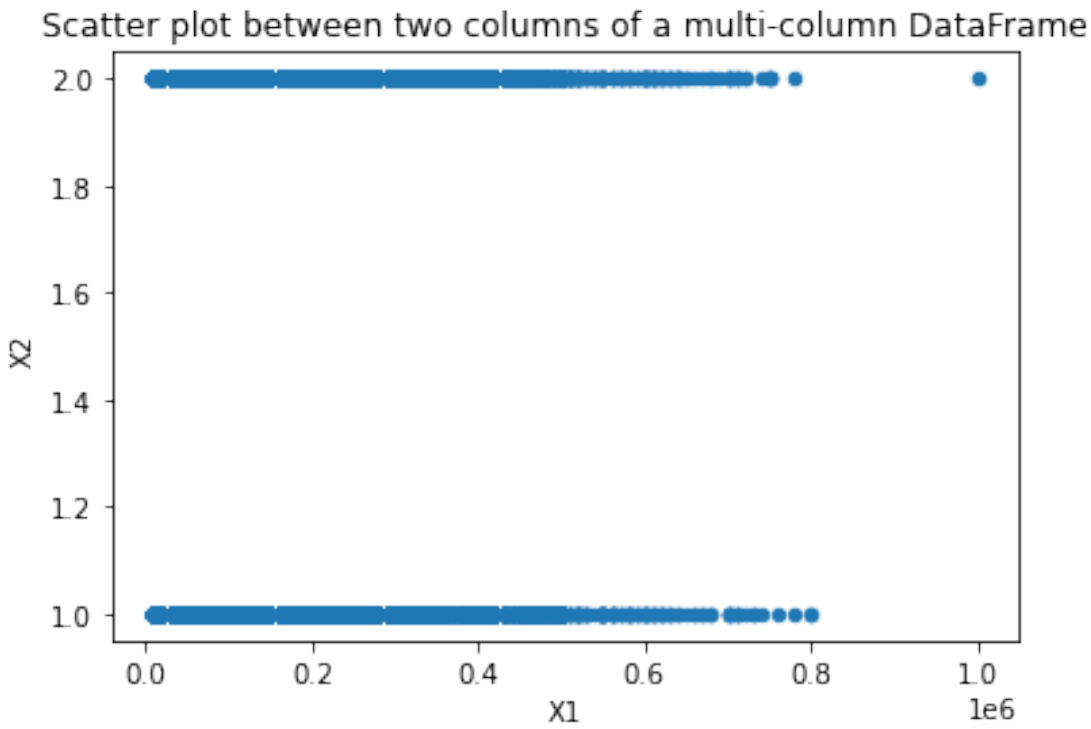
##9._Realiza la visualización de los datos usando por lo menos 3 gráficos que consideres adecuados: plot scatter, jointplot, boxplot, areaplot, pie chart, pairplot, bar chart, etc.

```
df_clean['X2'].value_counts()
```

```
Genero = ['Masculino','Femenino']
datos = [11863, 18095]

fig = plt.figure(figsize =(10, 7))
plt.pie(datos, labels = Genero)
# show plot
plt.show()
```



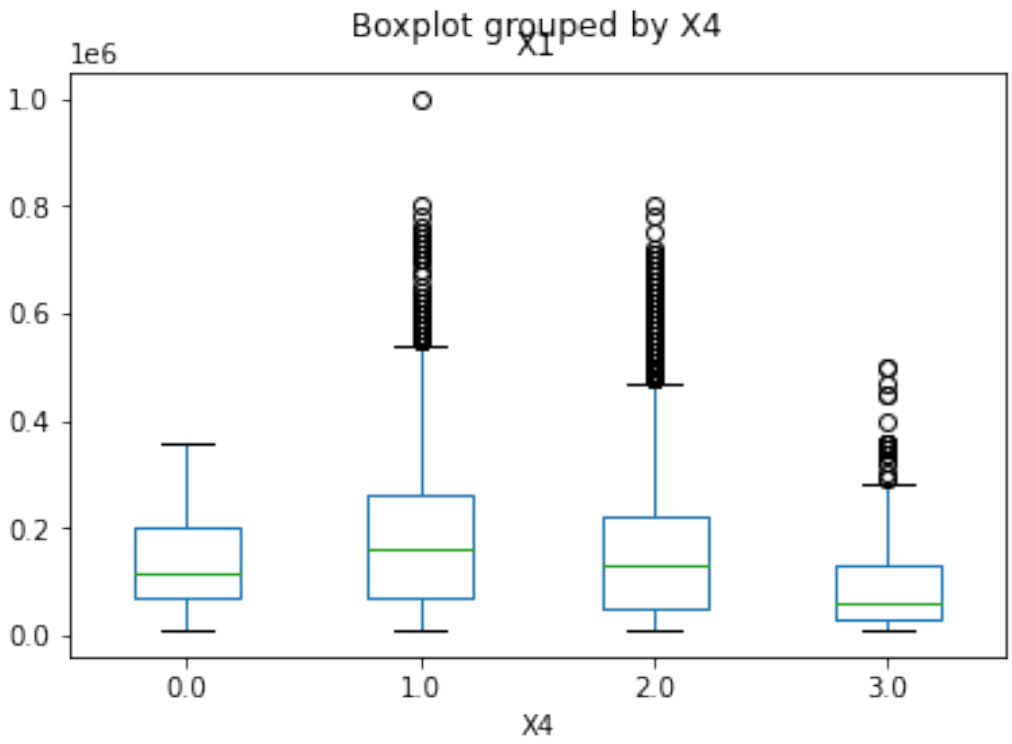

```
df_clean.plot.scatter(x='X1', y='X2', title= "Scatter plot between two
columns of a multi-column DataFrame");

plt.show(block=True);
```




```
df_clean.boxplot(by ='X4', column =['X1'], grid = False)
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/
__init__.py:1376: VisibleDeprecationWarning: Creating an ndarray from
ragged nested sequences (which is a list-or-tuple of lists-or-tuples-
or ndarrays with different lengths or shapes) is deprecated. If you
meant to do this, you must specify 'dtype=object' when creating the
ndarray.
  X = np.atleast_1d(X.T if isinstance(X, np.ndarray) else
np.asarray(X))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8114ffed10>
```



##10._Interpreta y explica cada uno de los graficos indicando cuál es la información más relevante que podría ayudar en el proceso de toma de decisiones.

- La gráfica de pastel representa la distribución proporcional de los datos, en nuestro caso del total de solicitudes de crédito, esta nos muestra que más del 50% de las solicitudes de crédito son realizadas por personas del género femenino, dato curioso.

- La gráfica de dispersión, nos muestra los valores de dos variables para un conjunto de datos, para nuestro caso partículas las variables que se muestran o comparan son la cantidad de monto solicitado con respecto al género. Podemos observar en la gráfica que las cantidades de dinero solicitadas entre el género masculino y femenino son muy similares. También observamos un dato fuera de rango en donde el género femenino realiza la solicitud de un crédito mayor.

- La gráfica de cajas y bigotes nos permite representar datos numéricos a través de cuartiles. Podemos observar rápidamente valores de mediana, y sus respectivos cuartiles, así como datos atípicos. En nuestro caso observamos que para los diferentes estados civiles se presentan bastantes datos atípicos. Las categorías de soltero y casado presentan mayores datos atípicos, la categoría de otro estado civil presenta la menor cantidad de datos atípicos. Observamos también que el estado civil que solicita mayor monto de crédito es el de casado, en comparación con otro estado civil.