In [ ]:
```
Ejercicio Semana 7 Visualizacion

## Integrante

Jorge Estivent Cruz Mahecha A01793808
```

In [29]:
```
import pandas as pd
import numpy as np
import seaborn as sns
```

1. Descarga los datos Enlaces a un sitio externo. y carga el dataset en tu libreta.

In [2]:
```
path ='https://raw.githubusercontent.com/PosgradoMNA/Actividades_Aprendizaje-/main/def
```

In [3]:
```
df = pd.read_csv(path, index_col=0)
df.index.name= None
df
```

Out[3]:

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X15	X16	X17	X18
1	20000	2.0	2.0	1.0	24.0	2.0	2.0	-1.0	-1.0	-2.0	...	0.0	0.0	0.0	0.0
2	120000	2.0	2.0	2.0	26.0	-1.0	2.0	0.0	0.0	0.0	...	3272.0	3455.0	3261.0	0.0
3	90000	2.0	2.0	2.0	34.0	0.0	0.0	0.0	0.0	0.0	...	14331.0	14948.0	15549.0	1518.0
4	50000	2.0	2.0	1.0	37.0	0.0	0.0	0.0	0.0	0.0	...	28314.0	28959.0	29547.0	2000.0
5	50000	1.0	2.0	1.0	57.0	-1.0	0.0	-1.0	0.0	0.0	...	20940.0	19146.0	19131.0	2000.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29996	220000	1.0	3.0	1.0	39.0	0.0	0.0	0.0	0.0	0.0	...	88004.0	31237.0	15980.0	8500.0
29997	150000	1.0	3.0	2.0	43.0	-1.0	-1.0	-1.0	-1.0	0.0	...	8979.0	5190.0	0.0	1837.0
29998	30000	1.0	2.0	2.0	37.0	4.0	3.0	2.0	-1.0	0.0	...	20878.0	20582.0	19357.0	0.0
29999	80000	1.0	3.0	1.0	41.0	1.0	-1.0	0.0	0.0	0.0	...	52774.0	11855.0	48944.0	85900.0
30000	50000	1.0	2.0	1.0	46.0	0.0	0.0	0.0	0.0	0.0	...	36535.0	32428.0	15313.0	2078.0

30000 rows × 24 columns

1. Obten la información del DataFrame con los métodos y propiedades: shape, columns, head(), dtypes, info(), isna()

In [9]:
```
df.shape
```

Out[9]:
```
(29958, 24)
```

In [10]:
```
df.columns
```

Out[10]:
```
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')
```

In [12]:
```
df.head()
```

Out[12]:

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X15	X16	X17	X18	X
1	20000	2.0	2.0	1.0	24.0	2.0	2.0	-1.0	-1.0	-2.0	...	0.0	0.0	0.0	0.0	689
2	120000	2.0	2.0	2.0	26.0	-1.0	2.0	0.0	0.0	0.0	...	3272.0	3455.0	3261.0	0.0	1000
3	90000	2.0	2.0	2.0	34.0	0.0	0.0	0.0	0.0	0.0	...	14331.0	14948.0	15549.0	1518.0	1500
4	50000	2.0	2.0	1.0	37.0	0.0	0.0	0.0	0.0	0.0	...	28314.0	28959.0	29547.0	2000.0	2019
5	50000	1.0	2.0	1.0	57.0	-1.0	0.0	-1.0	0.0	0.0	...	20940.0	19146.0	19131.0	2000.0	36681

5 rows × 24 columns

In [13]:
```
df.dtypes
```

Out[13]:
```
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
```

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29958 entries, 1 to 30000
Data columns (total 24 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   X1      29958 non-null  int64  
 1   X2      29958 non-null  float64
 2   X3      29958 non-null  float64
 3   X4      29958 non-null  float64
 4   X5      29958 non-null  float64
 5   X6      29958 non-null  float64
 6   X7      29958 non-null  float64
 7   X8      29958 non-null  float64
 8   X9      29958 non-null  float64
 9   X10     29958 non-null  float64
 10  X11     29958 non-null  float64
 11  X12     29958 non-null  float64
 12  X13     29958 non-null  float64
 13  X14     29958 non-null  float64
 14  X15     29958 non-null  float64
 15  X16     29958 non-null  float64
 16  X17     29958 non-null  float64
 17  X18     29958 non-null  float64
 18  X19     29958 non-null  float64
 19  X20     29958 non-null  float64
 20  X21     29958 non-null  float64
 21  X22     29958 non-null  float64
 22  X23     29958 non-null  float64
 23  Y       29958 non-null  float64
dtypes: float64(23), int64(1)
memory usage: 5.7 MB
```

In [15]:
```
df.isna()
```

Out[15]:

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X15	X16	X17	X18	X
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	Fa
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	Fa
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	Fa
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	Fa
5	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	Fa
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29996	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	Fa
29997	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	Fa
29998	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	Fa
29999	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	Fa
30000	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	Fa

29958 rows × 24 columns

1. Limpia los datos eliminando los registros nulos o rellena con la media de la columna

In [16]:
```
df.isna().any()
```

Out[16]:
```
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
```

In [17]:
```
df.dropna(axis = 0, inplace=True)
```

1. Calcula la estadística descriptiva con describe() y explica las medidas de tendencia central y dispersión

In [18]:
```
df.describe()
```

Out[18]:

	X1	X2	X3	X4	X5	X6	
count	29958.000000	29958.000000	29958.000000	29958.000000	29958.000000	29958.000000	29958.(
mean	167555.900928	1.604012	1.853094	1.551739	35.483443	-0.017124	-0.1
std	129737.299088	0.489070	0.790471	0.521952	9.214319	1.123989	1.1
min	10000.000000	1.000000	0.000000	0.000000	21.000000	-2.000000	-2.(
25%	50000.000000	1.000000	1.000000	1.000000	28.000000	-1.000000	-1.(
50%	140000.000000	2.000000	2.000000	2.000000	34.000000	0.000000	0.(
75%	240000.000000	2.000000	2.000000	2.000000	41.000000	0.000000	0.(
max	1000000.000000	2.000000	6.000000	3.000000	79.000000	8.000000	8.(

8 rows × 24 columns

Las columnas del modelo son todas de tipo numerico, tienen rangos especificos y la mayoria de las categorias tienen outliers pronunciados.

1. Escala los datos, si consideras necesario

In [22]:
```
from sklearn import preprocessing

scaler = preprocessing.StandardScaler().fit(df)
scaler

X_scaledA = scaler.transform(df)
X_scaledA = pd.DataFrame(X_scaledA)
X_scaledA
```

Out[22]:

	0	1	2	3	4	5	6	7	8
0	-1.137363	0.809689	0.185849	-1.057086	-1.246282	1.794642	1.782583	-0.696680	-0.66663(
1	-0.366561	0.809689	0.185849	0.858831	-1.029224	-0.874468	1.782583	0.139436	0.18924
2	-0.597802	0.809689	0.185849	0.858831	-0.160996	0.015235	0.111950	0.139436	0.18924
3	-0.906122	0.809689	0.185849	-1.057086	0.164590	0.015235	0.111950	0.139436	0.18924
4	-0.906122	-1.235043	0.185849	-1.057086	2.335161	-0.874468	0.111950	-0.696680	0.18924
...	...	...	...	...	...	...	...	...	...
29953	0.404240	-1.235043	1.450938	-1.057086	0.381647	0.015235	0.111950	0.139436	0.18924
29954	-0.135321	-1.235043	1.450938	0.858831	0.815761	-0.874468	-0.723367	-0.696680	-0.66663(
29955	-1.060283	-1.235043	0.185849	0.858831	0.164590	3.574048	2.617900	1.811669	-0.66663(
29956	-0.674882	-1.235043	1.450938	-1.057086	0.598704	0.904939	-0.723367	0.139436	0.18924
29957	-0.906122	-1.235043	0.185849	-1.057086	1.141347	0.015235	0.111950	0.139436	0.18924

29958 rows × 24 columns

In [24]:
```
df.corr()
```

Out[24]:

	X1	X2	X3	X4	X5	X6	X7	X8	X9
X1	1.000000	0.024212	-0.219120	-0.107801	0.144515	-0.271201	-0.296206	-0.286245	-0.267371
X2	0.024212	1.000000	0.014701	-0.031478	-0.091259	-0.057637	-0.070690	-0.065648	-0.059586
X3	-0.219120	0.014701	1.000000	-0.143431	0.175079	0.105636	0.121632	0.114012	0.108640
X4	-0.107801	-0.031478	-0.143431	1.000000	-0.413926	0.019747	0.024052	0.032430	0.032826
X5	0.144515	-0.091259	0.175079	-0.413926	1.000000	-0.039432	-0.050154	-0.053249	-0.049865
X6	-0.271201	-0.057637	0.105636	0.019747	-0.039432	1.000000	0.672290	0.574706	0.539384
X7	-0.296206	-0.070690	0.121632	0.024052	-0.050154	0.672290	1.000000	0.766857	0.662429
X8	-0.286245	-0.065648	0.114012	0.032430	-0.053249	0.574706	0.766857	1.000000	0.777245
X9	-0.267371	-0.059586	0.108640	0.032826	-0.049865	0.539384	0.662429	0.777245	1.000000
X10	-0.249340	-0.054319	0.097220	0.035360	-0.054006	0.509783	0.622990	0.686447	0.819655
X11	-0.235183	-0.043291	0.082077	0.034191	-0.048821	0.474809	0.575434	0.632396	0.716278
X12	0.285616	-0.033696	0.023451	-0.023430	0.056357	0.187115	0.234878	0.208843	0.203124
X13	0.278435	-0.031215	0.018669	-0.021536	0.054469	0.189906	0.235227	0.237642	0.226103
X14	0.283373	-0.024588	0.012913	-0.024876	0.053911	0.179796	0.224105	0.227827	0.245260
X15	0.294090	-0.021854	-0.000566	-0.023292	0.051528	0.179138	0.222216	0.227518	0.246180
X16	0.295625	-0.017056	-0.007625	-0.025371	0.049515	0.180651	0.221287	0.225453	0.243161
X17	0.290517	-0.016733	-0.009192	-0.021206	0.047810	0.176992	0.219339	0.222608	0.239379
X18	0.195026	-0.000297	-0.037385	-0.005705	0.025995	-0.079230	-0.080771	0.001210	-0.009475
X19	0.178320	-0.001517	-0.030007	-0.008016	0.021820	-0.070083	-0.058981	-0.066784	-0.001893
X20	0.210052	-0.008630	-0.039953	-0.003399	0.029262	-0.070460	-0.055864	-0.053294	-0.069260
X21	0.203187	-0.002369	-0.038263	-0.012516	0.021338	-0.063934	-0.046856	-0.046001	-0.043421
X22	0.216990	-0.001775	-0.040276	-0.001052	0.022811	-0.058104	-0.037049	-0.035813	-0.033630
X23	0.219567	-0.002911	-0.037189	-0.006571	0.019533	-0.058616	-0.036491	-0.035816	-0.026537
Y	-0.153781	-0.039730	0.028109	-0.024544	0.013881	0.324769	0.263761	0.235765	0.217037

24 rows × 24 columns

1. Reduce las dimensiones con PCA, si consideras necesario. Indica la varianza de los datos explicada por cada componente seleccionado. Para actividades de exploración de los datos la varianza > 70% Indica la importancia de las variables en cada componente

In [33]:

```
import numpy as np
from sklearn.decomposition import PCA
pcs = PCA(n_components=24)

pcs.fit(X_scaledA)
```

```
pcsSummary = pd.DataFrame({'Standard deviation': np.sqrt(pcs.explained_variance_),
                           'Proportion of variance': pcs.explained_variance_ratio_,
                           'Cumulative proportion': np.cumsum(pcs.explained_variance_ra
                           }
                           )
pcsSummary
```

Out[33]:

	Standard deviation	Proportion of variance	Cumulative proportion
0	2.559777	0.273010	0.273010
1	2.049717	0.175050	0.448060
2	1.246196	0.064706	0.512766
3	1.214423	0.061449	0.574215
4	1.020248	0.043370	0.617585
5	0.986956	0.040585	0.658170
6	0.956944	0.038155	0.696325
7	0.952515	0.037802	0.734127
8	0.941475	0.036931	0.771058
9	0.933451	0.036304	0.807363
10	0.884852	0.032622	0.839985
11	0.855766	0.030513	0.870498
12	0.792207	0.026149	0.896647
13	0.754684	0.023730	0.920377
14	0.724292	0.021858	0.942235
15	0.630763	0.016577	0.958812
16	0.509445	0.010814	0.969625
17	0.498871	0.010369	0.979995
18	0.434526	0.007867	0.987861
19	0.363186	0.005496	0.993357
20	0.264844	0.002923	0.996280
21	0.201936	0.001699	0.997979
22	0.159026	0.001054	0.999032
23	0.152384	0.000968	1.000000

In [34]:

```
import matplotlib as mpl
import matplotlib.pyplot as plt
PC_components = np.arange(pcs.n_components_) + 1
#PC_components

_ = sns.set(style = 'whitegrid',
            font_scale = 1.2
```

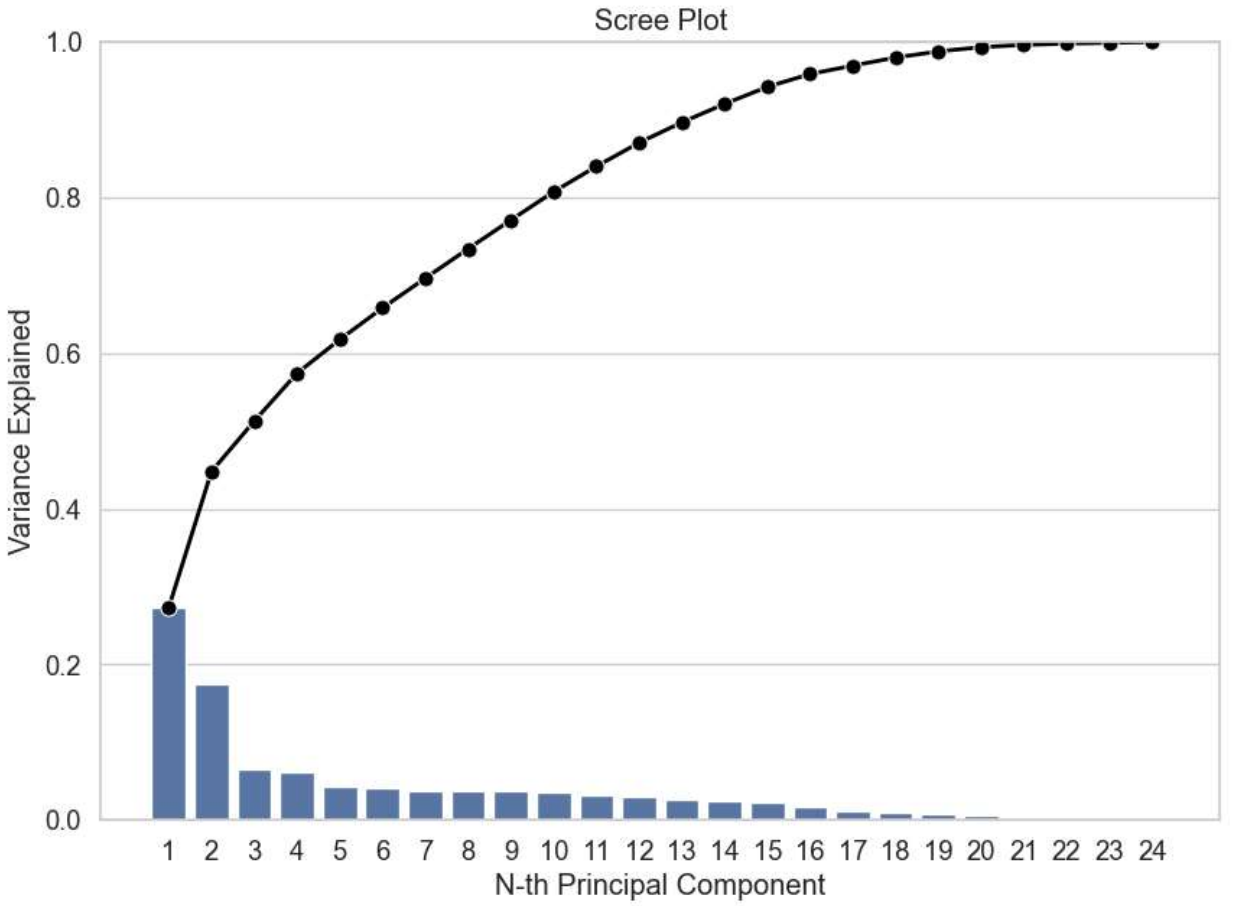
```
            )

fig, ax = plt.subplots(figsize=(10, 7))

_ = sns.barplot(x = PC_components,
                y = pcs.explained_variance_ratio_,
                color = 'b'
                )

_ = sns.lineplot(x = PC_components-1,
                 y = np.cumsum(pcs.explained_variance_ratio_),
                 color = 'black',
                 linestyle = '-',
                 linewidth = 2,
                 marker = 'o',
                 markersize = 8
                 )

plt.title('Scree Plot')
plt.xlabel('N-th Principal Component')
plt.ylabel('Variance Explained')
plt.ylim(0, 1)
plt.show()
```



In [36]:
```
pcs = PCA()
pcs.fit(X_scaledA.iloc[:, 3:].dropna(axis=0))
pcsSummary_df = pd.DataFrame({'Standard deviation': np.sqrt(pcs.explained_variance_),
                              'Proportion of variance': pcs.explained_variance_ratio_,
                              'Cumulative proportion': np.cumsum(pcs.explained_variance
                              })
```

```
pcsSummary_df = pcsSummary_df.transpose()
pcsSummary_df
```

Out[36]:

	0	1	2	3	4	5	6	7	8
Standard deviation	2.554281	1.957491	1.230657	1.182589	0.974661	0.950674	0.941711	0.933903	0.884892
Proportion of variance	0.310673	0.182459	0.072117	0.066594	0.045235	0.043036	0.042228	0.041531	0.037286
Cumulative proportion	0.310673	0.493132	0.565250	0.631844	0.677078	0.720114	0.762342	0.803873	0.841159

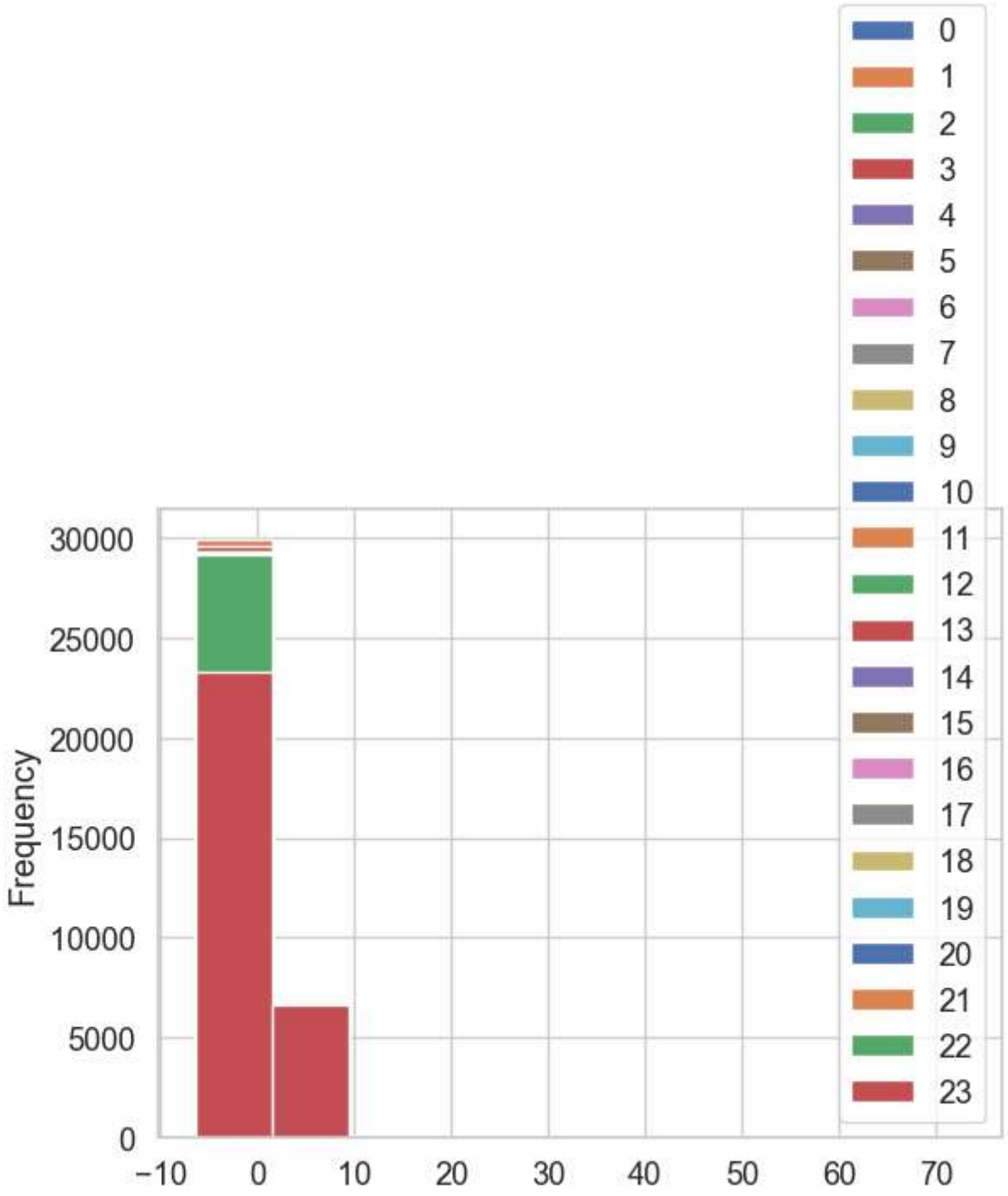
3 rows × 21 columns

Se nota la mayor variabilidad de los datos en los primeros 2 componentes principales y tambien se evidencia mayor importancia en estos componentes.

1. Elabora los histogramas de los atributos para visualizar su distribución

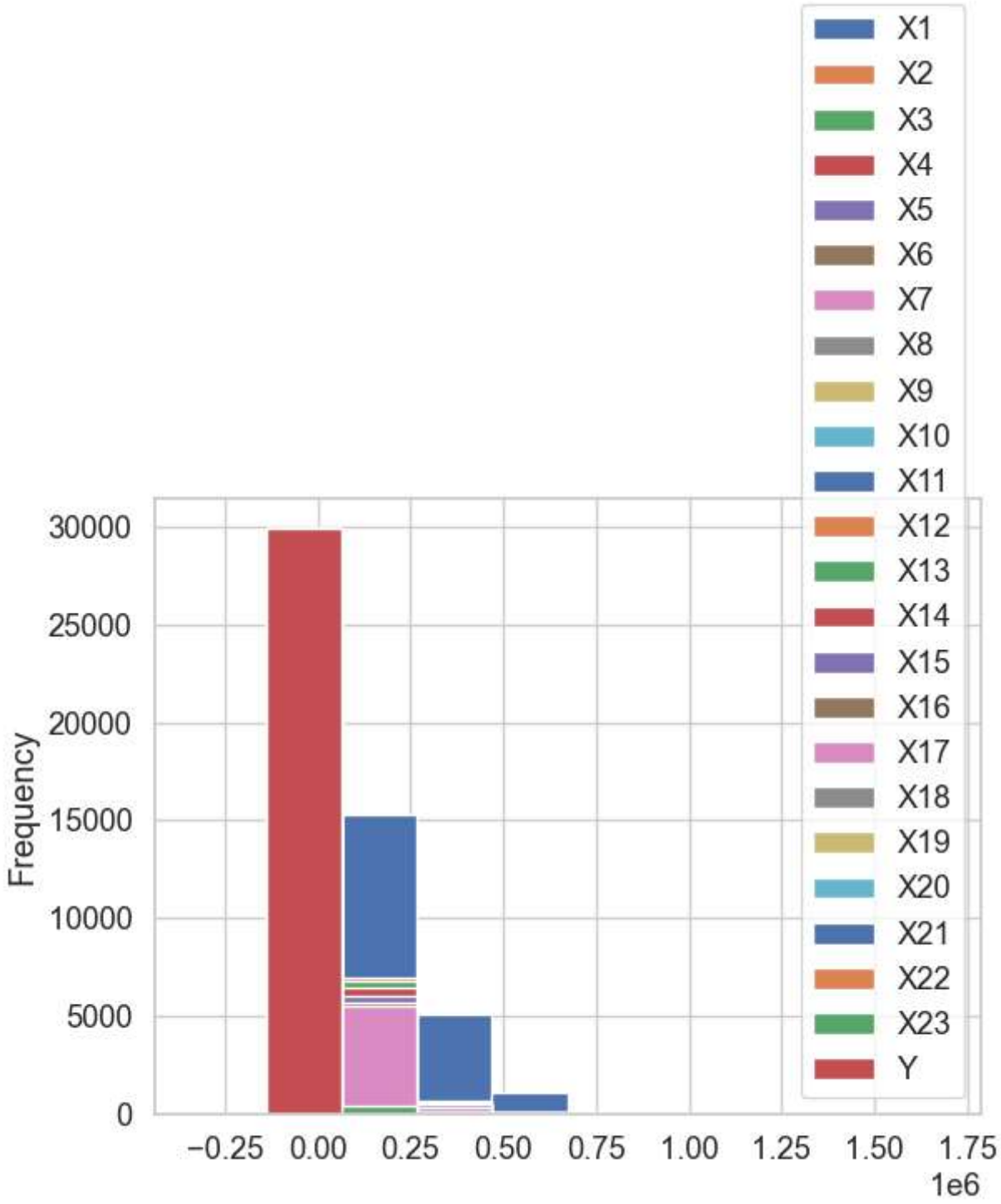
In [37]:
```
X_scaledA.plot.hist ()
```

Out[37]:
```
<AxesSubplot: ylabel='Frequency'>
```



In [39]:
```
df.plot.hist ()
```

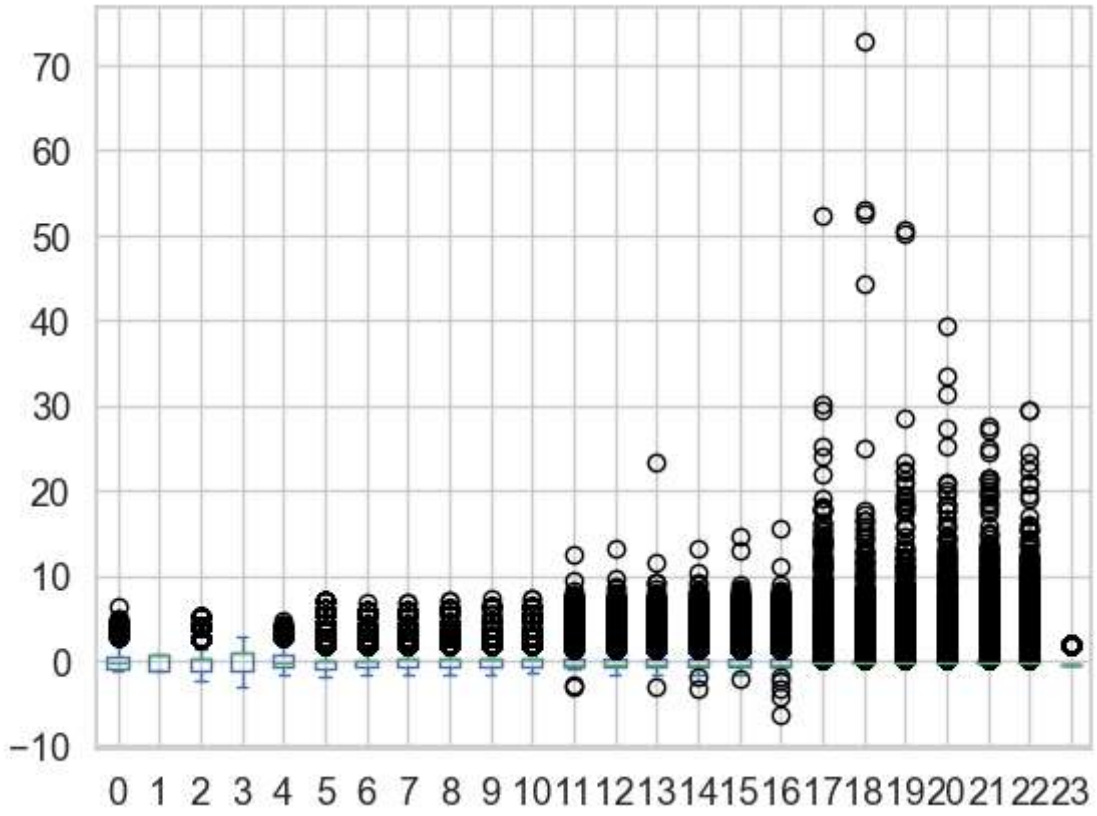
Out[39]:
```
<AxesSubplot: ylabel='Frequency'>
```



1. 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.

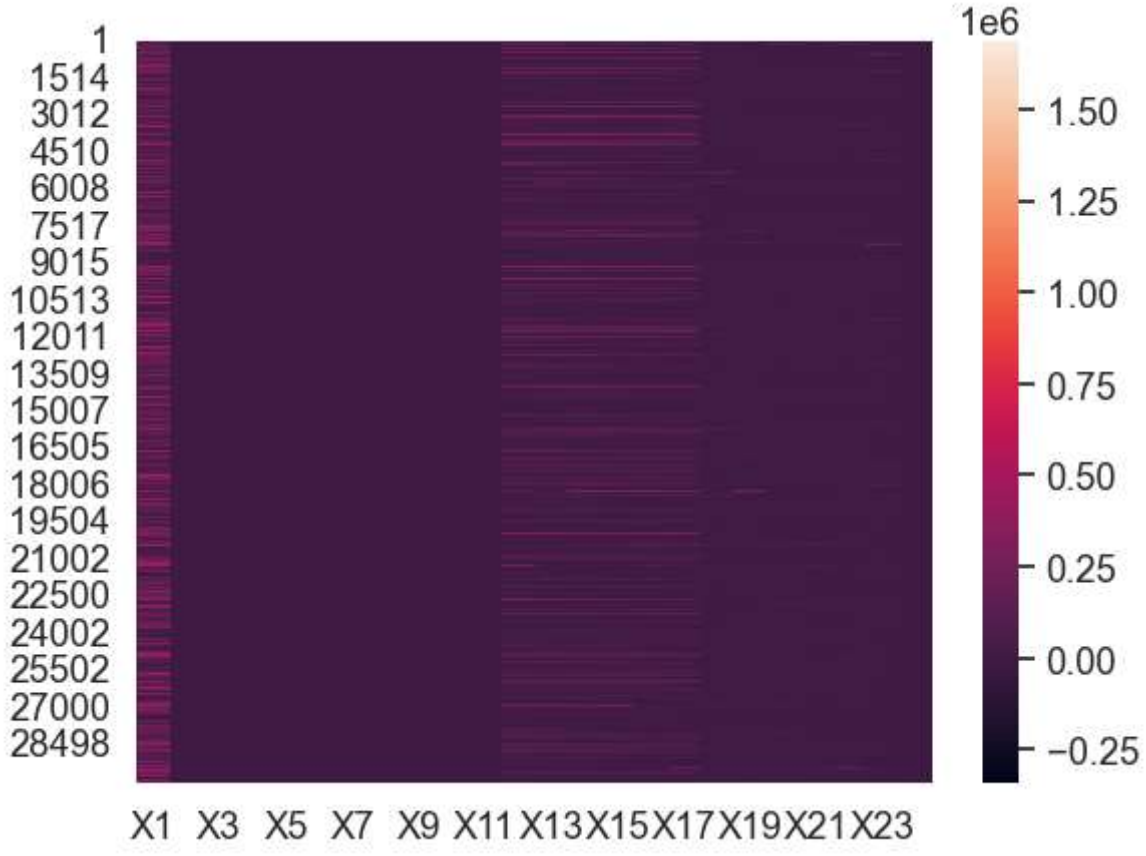
In [41]:
```
X_scaledA.plot.box()
```

Out[41]:
```
<AxesSubplot: >
```



In [55]:
```
sns.heatmap(df)
```

Out[55]:
```
<AxesSubplot: >
```

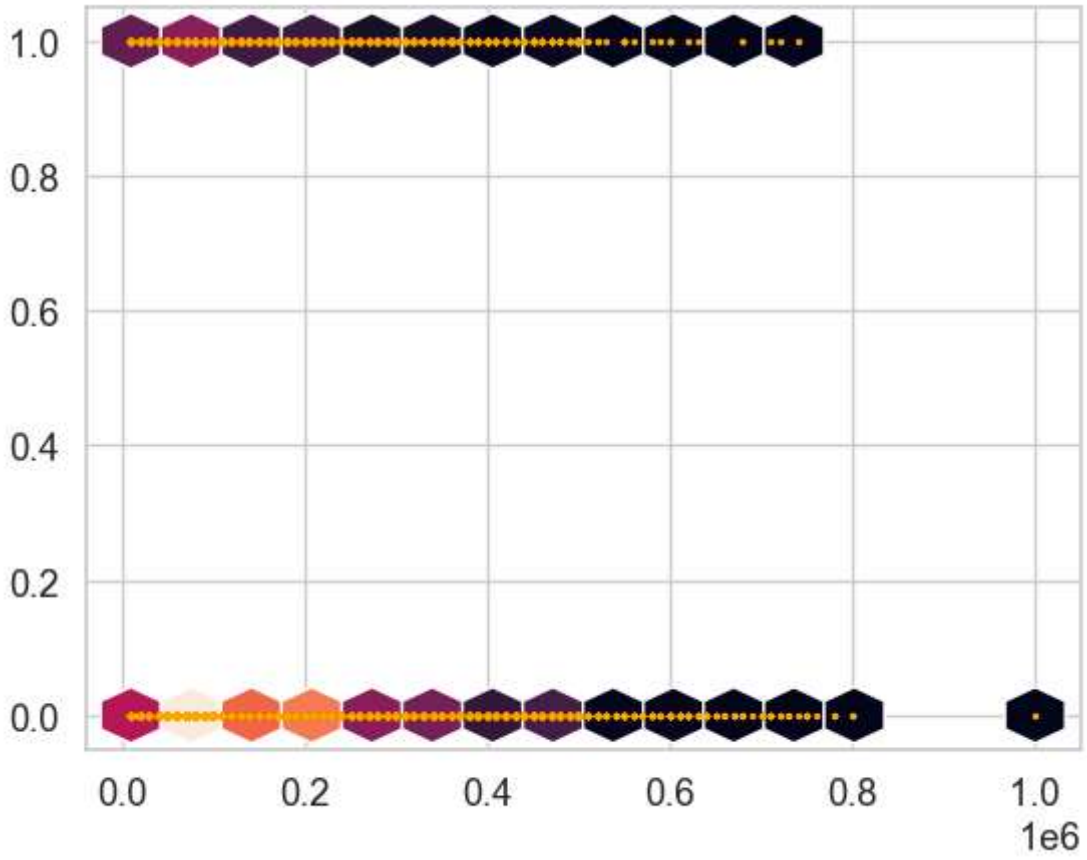


In [61]:
```
#sns.countplot(x = df.head, data=df)
```

```
plt.hexbin(df['X1'],df['Y'],gridsize=15, mincnt=1, edgecolors="white")

plt.scatter(df['X1'],df['Y'], s=2, c="orange")

plt.show()
```



In [ ]:

In [ ]:
```
plt.stackplot(df['X1',df['Y'],,colors =['b', 'y'])


plt.show()
```

Se evalua que para el tipo de datasets las graficas que mejor se acomodan son las de tipo histograma y boxplot, para tantos datos las demas graficas presentan perdida de informacion de caracter visual.

In [ ]: