```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
import plotly.express as px
import plotly.io as pio
```

```
df=pd.read_excel("/content/Churn_Modelling.xlsx")
```

```
df
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenur
0	1.0	15634602.0	Hargrave	619.0	France	Female	42.0	2.
1	2.0	15647311.0	Hill	608.0	Spain	Female	41.0	1.
2	3.0	15619304.0	Onio	502.0	France	Female	42.0	8.
3	4.0	15701354.0	Boni	699.0	France	Female	39.0	1.
4	5.0	15737888.0	Mitchell	850.0	Spain	Female	43.0	2.
...	...	...	...	...	...	...	...	.
9995	9996.0	15606229.0	Obijiaku	771.0	France	Male	39.0	5.
9996	9997.0	15569892.0	Johnstone	516.0	France	Male	35.0	10.
9997	9998.0	15584532.0	Liu	709.0	France	Female	36.0	7.
9998	9999.0	15682355.0	Sabbatini	772.0	Germany	Male	42.0	3.
9999	10000.0	15628319.0	Walker	792.0	France	Female	28.0	4.

10000 rows × 14 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  float64
 1   CustomerId       10000 non-null  float64
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  float64
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  float64
 7   Tenure           10000 non-null  float64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  float64
 10  HasCrCard        10000 non-null  float64
 11  IsActiveMember   10000 non-null  float64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  float64
dtypes: float64(11), object(3)
memory usage: 1.1+ MB
```

```
df.isnull().sum()
```

```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

```
df.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Ba
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.0
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.8
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.4
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.0
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.0
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.5
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.2
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.0

```
df.info()
```

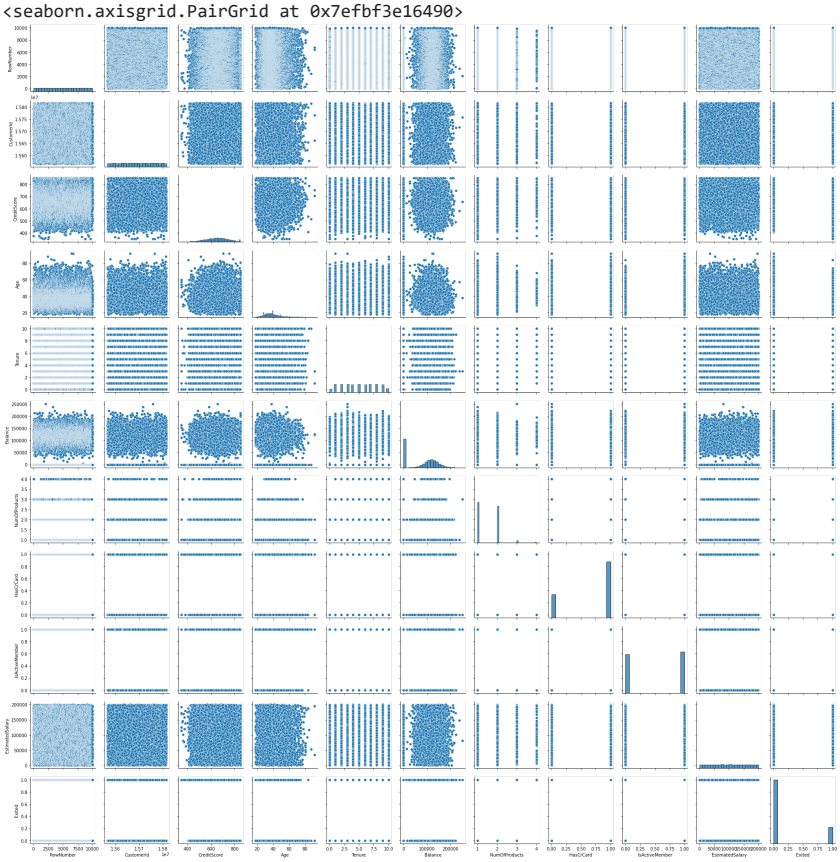
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  float64
 1   CustomerId       10000 non-null  float64
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  float64
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  float64
 7   Tenure           10000 non-null  float64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  float64
 10  HasCrCard        10000 non-null  float64
 11  IsActiveMember   10000 non-null  float64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  float64
dtypes: float64(11), object(3)
memory usage: 1.1+ MB
```

```
df.corr()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance
RowNumber	1.000000	0.004202	0.005840	0.000783	-0.006495	-0.009067
CustomerId	0.004202	1.000000	0.005308	0.009497	-0.014883	-0.012419
CreditScore	0.005840	0.005308	1.000000	-0.003965	0.000842	0.006268
Age	0.000783	0.009497	-0.003965	1.000000	-0.009997	0.028308
Tenure	-0.006495	-0.014883	0.000842	-0.009997	1.000000	-0.012254
Balance	-0.009067	-0.012419	0.006268	0.028308	-0.012254	1.000000
NumOfProducts	0.007246	0.016972	0.012238	-0.030680	0.013444	-0.304180
HasCrCard	0.000599	-0.014025	-0.005458	-0.011721	0.022583	-0.014858
IsActiveMember	0.012044	0.001665	0.025651	0.085472	-0.028362	-0.010084
EstimatedSalary	-0.005988	0.015271	-0.001384	-0.007201	0.007784	0.012797
Exited	-0.016571	-0.006248	-0.027094	0.285323	-0.014001	0.118533

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7efbf3e16490>
```



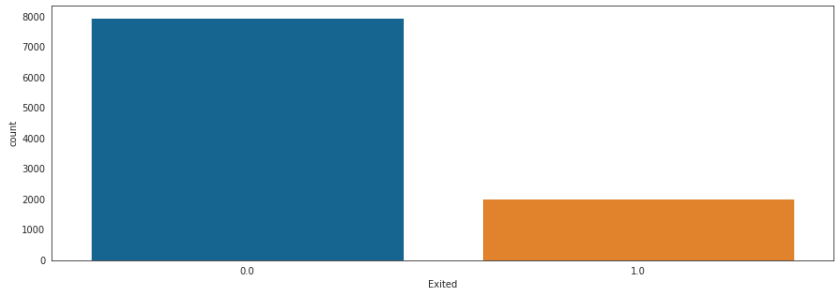

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams["figure.figsize"] = (15,5)
sns.set_style('white')
```

```
plt.style.use('tableau-colorblind10')
#plt.xkcd()
```

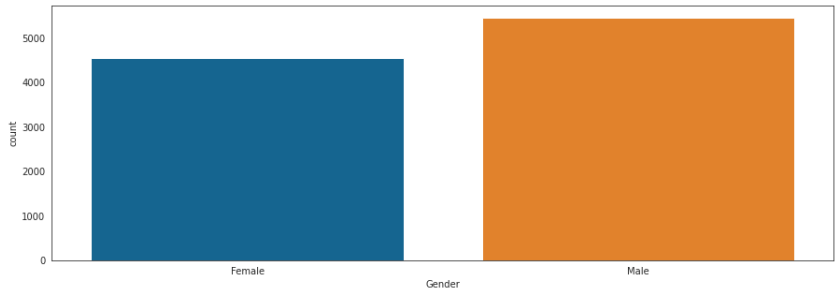
```
plt.style.available
```

```
['Solarize_Light2',
 '_classic_test_patch',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn',
 'seaborn-bright',
 'seaborn-colorblind',
 'seaborn-dark',
 'seaborn-dark-palette',
 'seaborn-darkgrid',
 'seaborn-deep',
 'seaborn-muted',
 'seaborn-notebook',
 'seaborn-paper',
 'seaborn-pastel',
 'seaborn-poster',
 'seaborn-talk',
 'seaborn-ticks',
 'seaborn-white',
 'seaborn-whitegrid',
 'tableau-colorblind10']
```

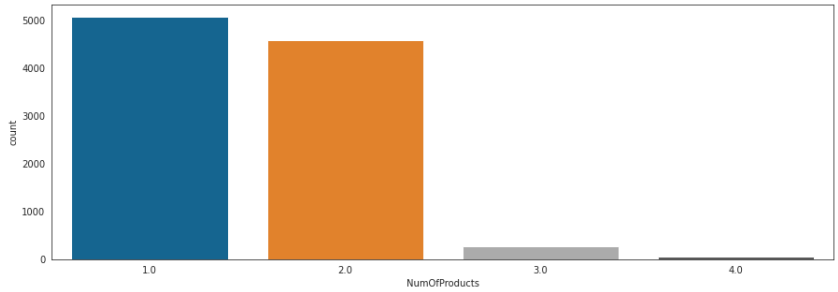
```
ax = sns.countplot('Exited',data=df) #Imbalanced Dataset
```



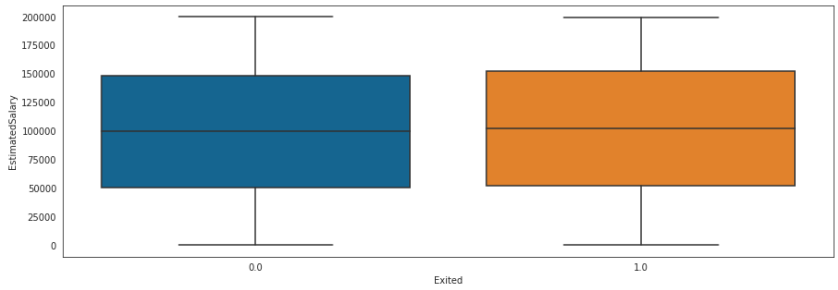

```
ax = sns.countplot('Gender',data=df)
```



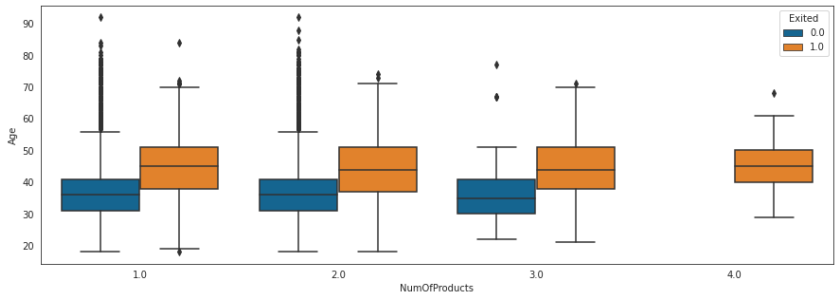

```
ax = sns.countplot('NumOfProducts',data=df)
```



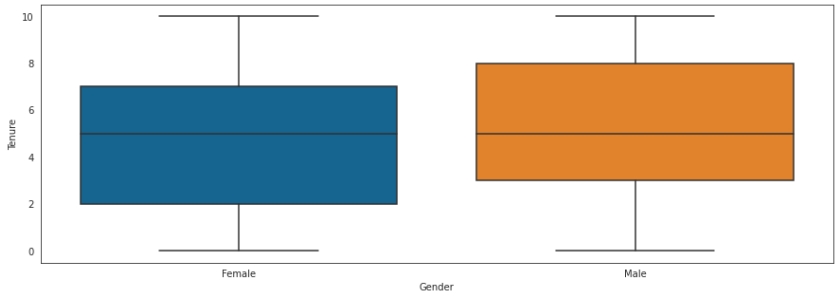

```
ax = sns.boxplot(x='Exited',y='EstimatedSalary',data=df)
```



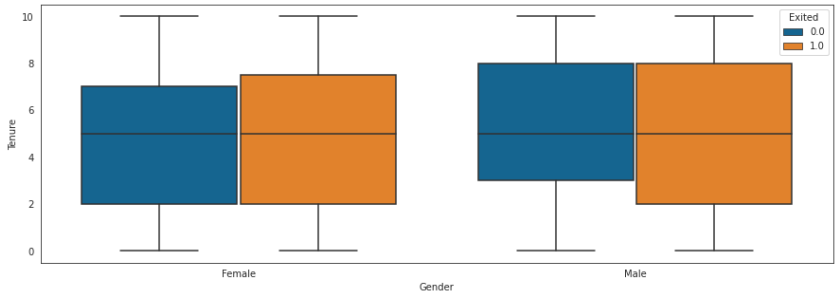

```
ax = sns.boxplot(x='NumOfProducts',y='Age',hue='Exited',data=df)
```



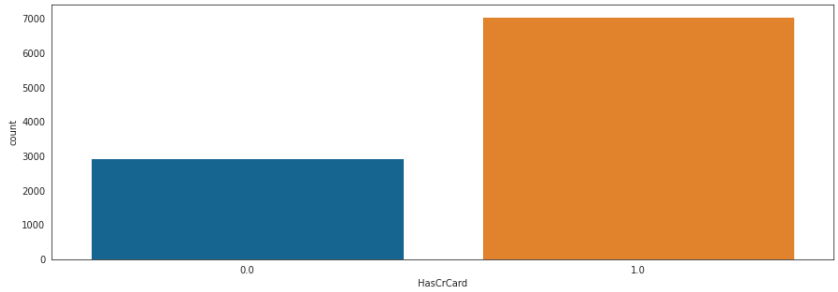

```
ax = sns.boxplot(x='Gender',y='Tenure',data=df)
```



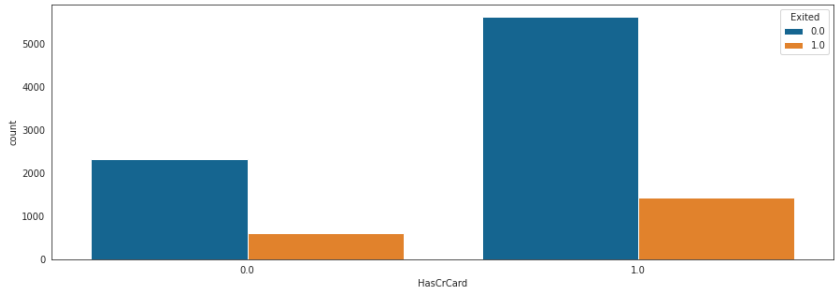

```
ax = sns.boxplot(x='Gender',y='Tenure',hue='Exited',data=df)
```




```
#HasCrCard
ax = sns.countplot('HasCrCard',data=df)
```

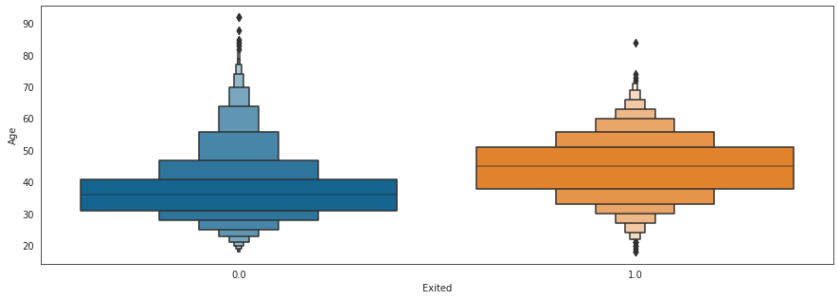



```
ax = sns.countplot('HasCrCard',hue='Exited',data=df)
```

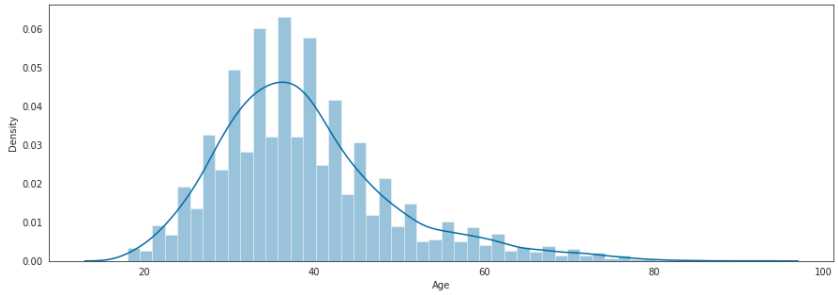



```
ax = sns.boxplot(x='Exited',y='Age',data=df)
```

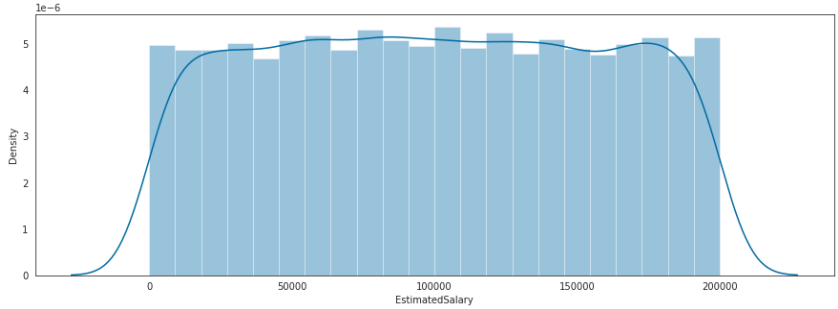
```
ax = sns.boxenplot(x='Exited',y='Age',data=df)
```




```
ax = sns.distplot(df.Age)
```



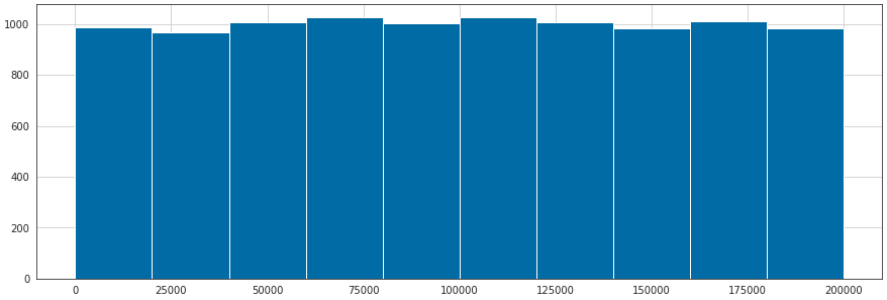

```
#EstimatedSalary
ax = sns.distplot(df.EstimatedSalary)
```



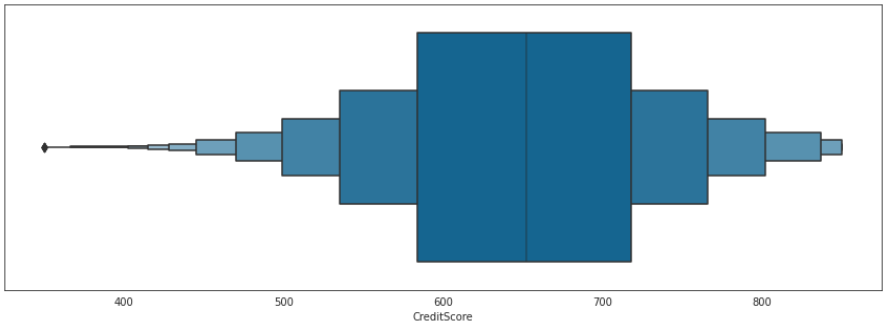

```
df.EstimatedSalary.describe()
```

```
count     10000.000000
mean     100090.239881
std       57510.492818
min          11.580000
25%       51002.110000
50%      100193.915000
75%      149388.247500
max      199992.480000
Name: EstimatedSalary, dtype: float64
```

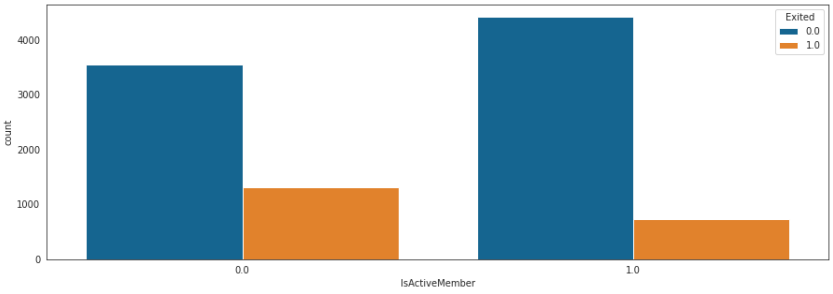
```
ax = df['EstimatedSalary'].hist()
```



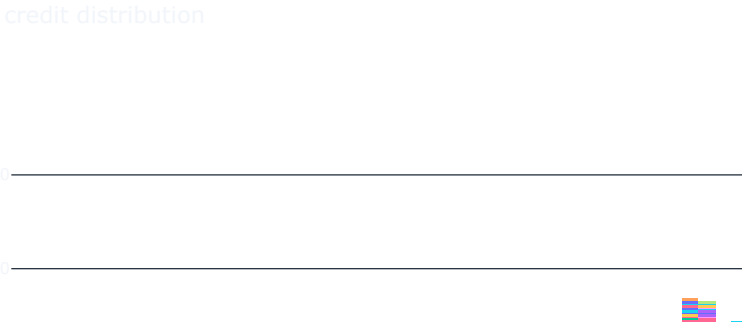

```
ax = sns.boxenplot('CreditScore',data=df)
```



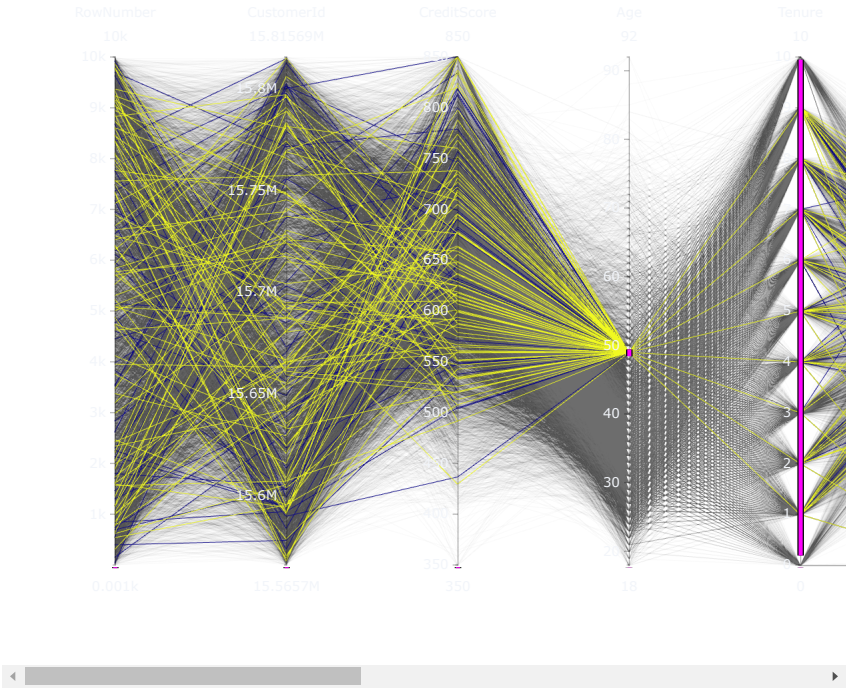

```
ax = sns.countplot('IsActiveMember',hue='Exited',data=df)
```




```
ax= px.histogram(df,x= "CreditScore", template= "plotly_dark",color= "Balance",title='credit distribution')
ax.show()
```




```
ax= px.parallel_coordinates(df, color="HasCrCard",template= "plotly_dark")
ax.show()
```




```
df.drop(["Gender"],axis=1,inplace=True)

#splittig x and y for predictoions

x=df.iloc[:,1:-1]

x
```

	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	I
0	15634602.0	619.0	42.0	2.0	0.00	1.0	1.0	
2	15619304.0	502.0	42.0	8.0	159660.80	3.0	1.0	

```
y=df["Exited"]
```

```
y
```

```
0       1.0
1       0.0
2       1.0
3       0.0
4       0.0
       ...
9995    0.0
9996    0.0
9997    1.0
9998    1.0
9999    0.0
Name: Exited, Length: 10000, dtype: float64
```

10000 rows x 9 columns

```
#splitting the model in x,y train and test


from sklearn.model_selection import train_test_split


xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.30,random_state=1)


from sklearn.linear_model import LogisticRegression


logreg=LogisticRegression()
logreg.fit(xtrain,ytrain)
ypred=logreg.predict(xtest)


from sklearn.metrics import classification_report


print(classification_report(ytest,ypred))
```

```
                  precision    recall  f1-score   support

              0.0       0.79      1.00      0.88      2373
              1.0       0.00      0.00      0.00       627

         accuracy                           0.79      3000
        macro avg       0.40      0.50      0.44      3000
     weighted avg       0.63      0.79      0.70      3000
```

```
#feature scaling


from sklearn.preprocessing import StandardScaler


sc=StandardScaler()
xtrain=sc.fit_transform(xtrain)


xtest=sc.transform(xtest)


#hypertuning using solver parameter


logreg=LogisticRegression(solver="saga")
logreg.fit(xtrain,ytrain)
ypred=logreg.predict(xtest)


print(classification_report(ytest,ypred))
```

```
                  precision    recall  f1-score   support

              0.0       0.82      0.98      0.89      2373
              1.0       0.65      0.17      0.27       627
```

accuracy			0.81	3000
macro avg	0.73	0.57	0.58	3000
weighted avg	0.78	0.81	0.76	3000