Assignment-2

Python Programming

Data Visualization and Pre-Processing

Assignment Date	21 September 2022
Student Name	Ms. Ishwarya G K
Student Register Number	910619104029
Maximum Marks	

In [124…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [125…
```
df = pd.read_csv(r"E:\Assignments of ibm\Ass 2\Datasets\Churn_Modelling.csv")
```

In [126…
```
df.head()
```

Out[126]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProd
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	
3	4	15701354	Boni	699	France	Female	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	

In [127…
```
df.tail()
```

Out[127]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOf
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	

In [128…
```
df.describe()
```

Out[128]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000

50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000

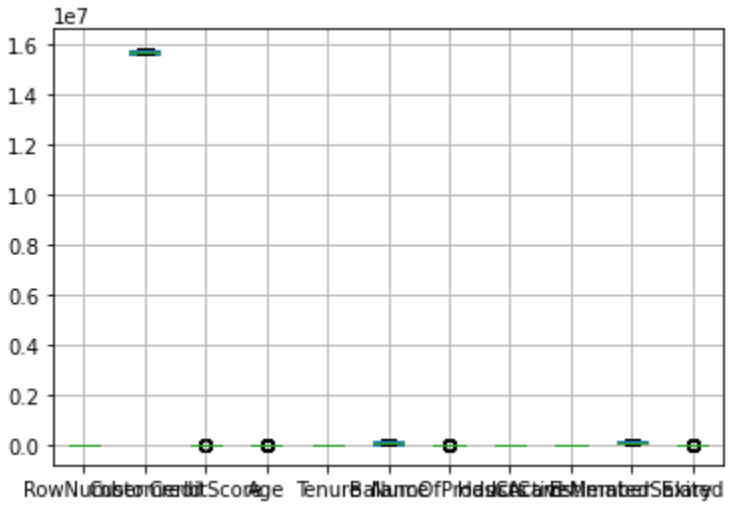
In [129…

```
df.dtypes
```

Out[129]:
```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

In [131…
```
df.boxplot()
```

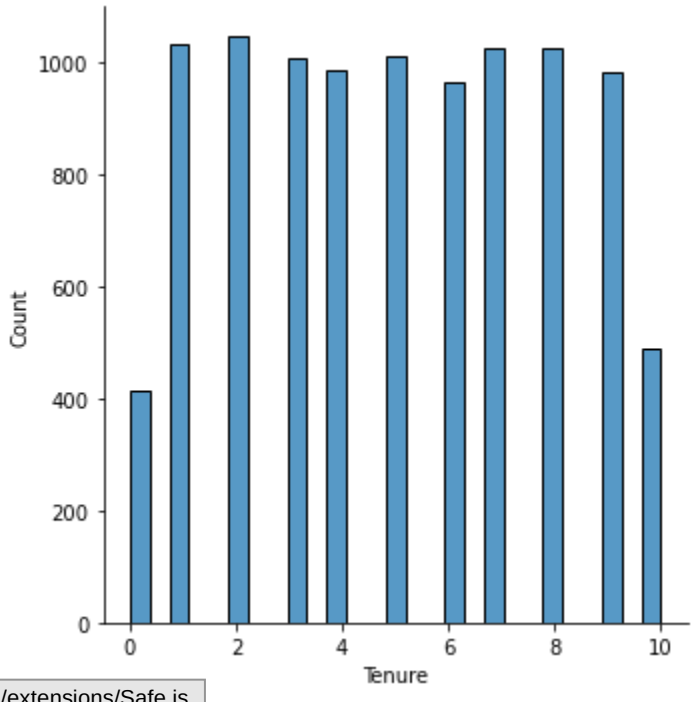
Out[131]:
```
<AxesSubplot:>
```



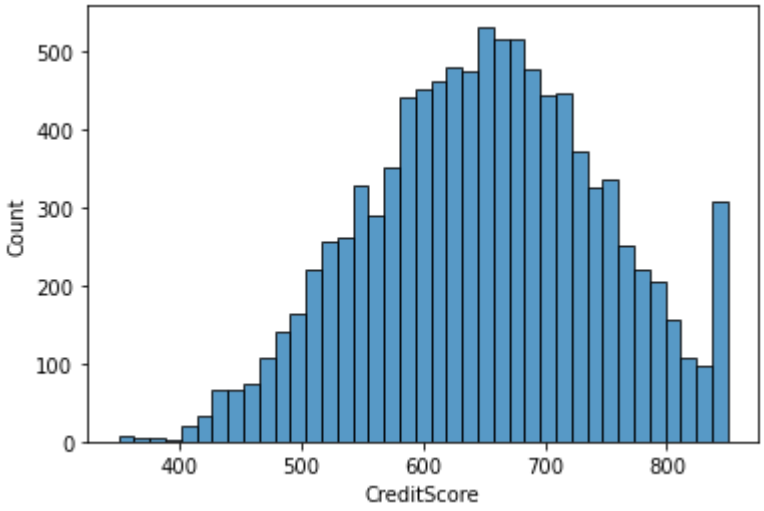
In [132…
```
### Univariate Analysis
```

In [133…
```
sns.displot(df.Tenure)
```

Out[133]:
```
<seaborn.axisgrid.FacetGrid at 0x1d8f45d9100>
```

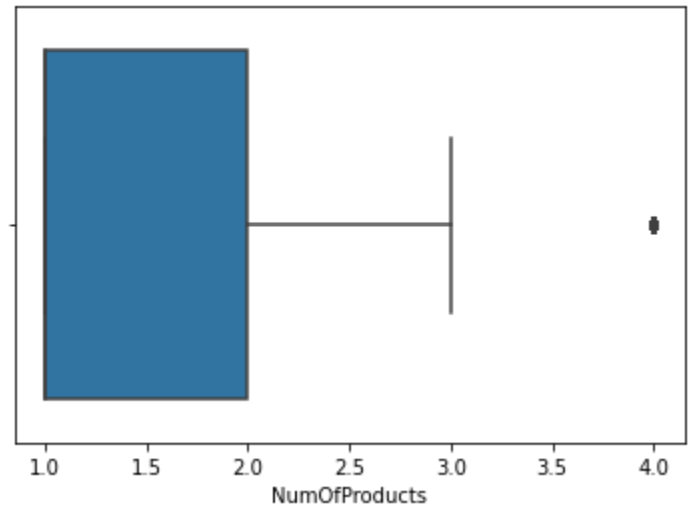


In [134…
```
sns.histplot(df.CreditScore.dropna(), kde=False, bins = 39);
```



In [178…
```
import warnings
warnings.filterwarnings("ignore")
sns.boxplot(df["NumOfProducts"])
```

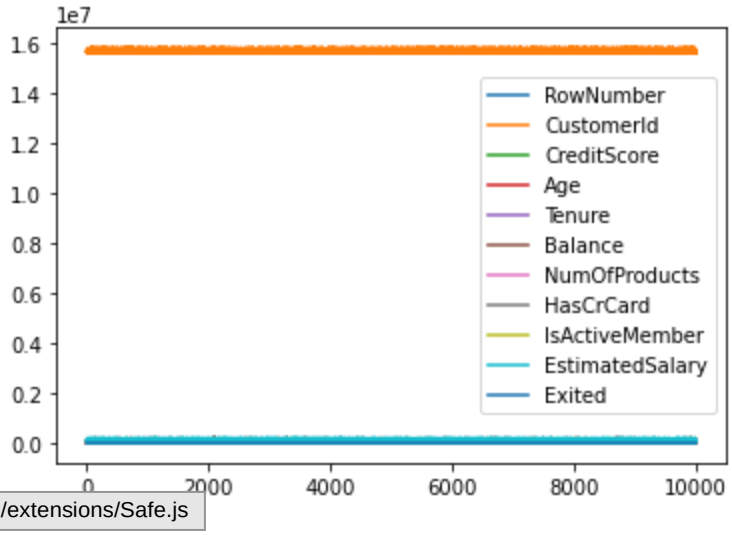
Out[178]:
```
<AxesSubplot:xlabel='NumOfProducts'>
```



In [136…
```
### Bivariate Analysis
```

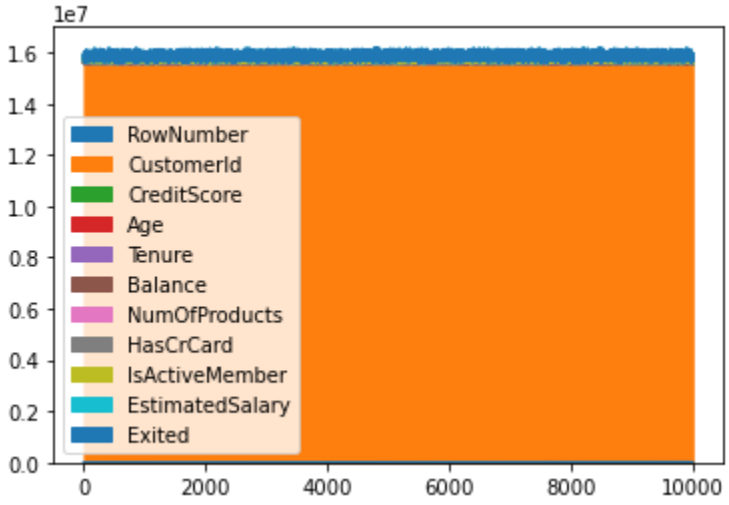
In [137…
```
df.plot.line()
```

Out[137]:
```
<AxesSubplot:>
```



In [138…
```
df.plot.area()
```

Out[138]:
```
<AxesSubplot:>
```

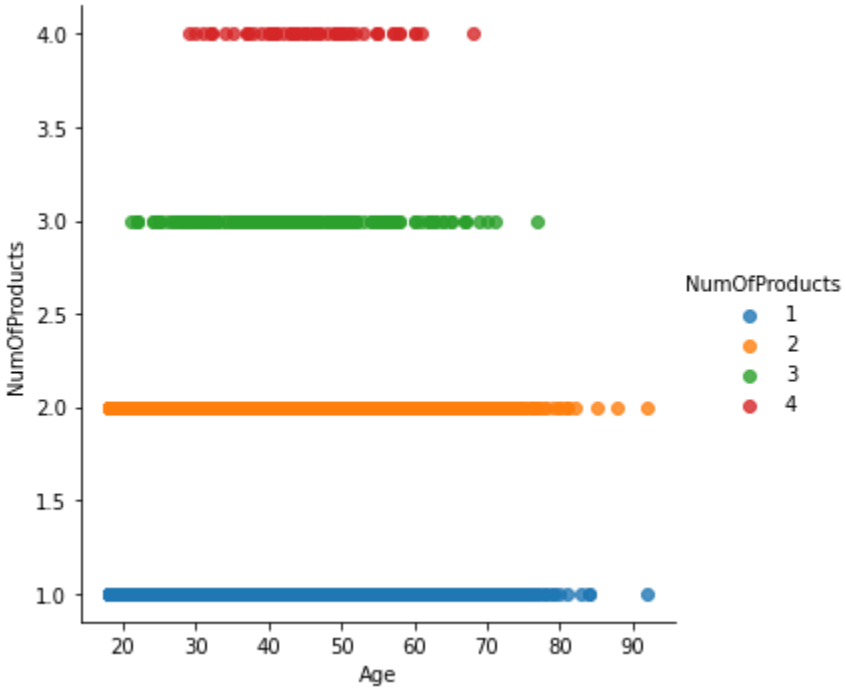


In [139…
```
###Multivariate Analysis
```

In [140…
```
sns.lmplot("Age","NumOfProducts",df,hue="NumOfProducts",fit_reg=False)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y, data. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[140]:
```
<seaborn.axisgrid.FacetGrid at 0x1d8fa0847f0>
```



In [141…
```
### Descriptive Statistics
```

In [142…
```
df.mean()
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_7440\3698961737.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
```

Out[142]:
```
RowNumber          5.000500e+03
CustomerId         1.569094e+07
CreditScore        6.505288e+02
Age                3.892180e+01
Tenure             5.012800e+00
Balance            7.648589e+04
NumOfProducts      1.530200e+00
HasCrCard          7.055000e-01
IsActiveMember     5.151000e-01
EstimatedSalary    1.000902e+05
Exited             2.037000e-01
dtype: float64
```

In [143…
```
df.mode()
```

Out[143]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfPr
0	1	15565701	Smith	850.0	France	Male	37.0	2.0	0.0	
1	2	15565706	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	3	15565714	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	4	15565779	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	5	15565796	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	
9995	9996	15815628	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
9996	9997	15815645	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
9997	9998	15815656	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
9998	9999	15815660	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
9999	10000	15815690	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

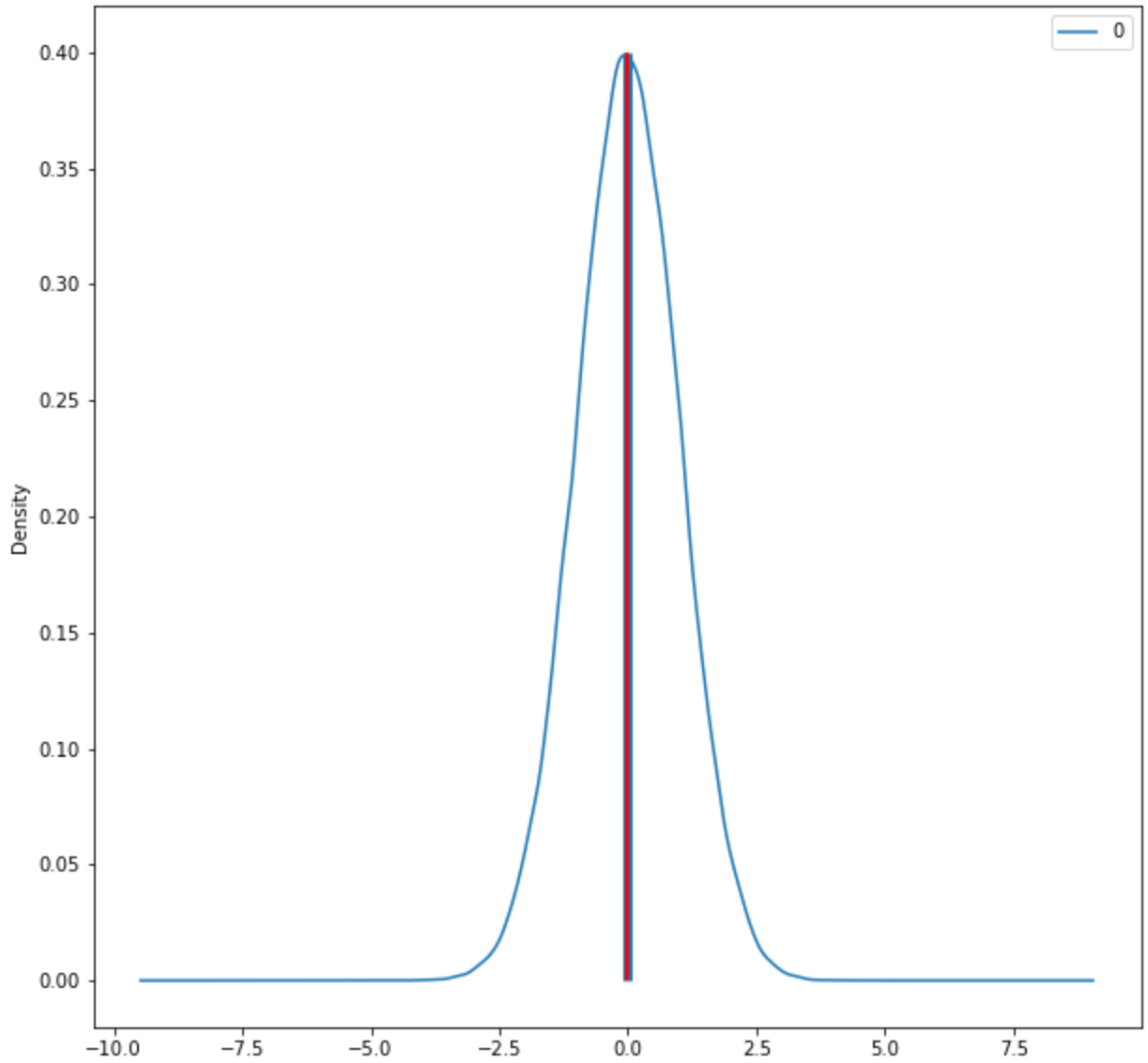
10000 rows × 14 columns

In [144…
```
norm_data = pd.DataFrame(np.random.normal(size=100000))

norm_data.plot(kind="density",
               figsize=(10,10));


plt.vlines(norm_data.mean(),       # Plot black line at mean
           ymin=0,
           ymax=0.4,
           linewidth=5.0);

plt.vlines(norm_data.median(),     # Plot red line at median
           ymin=0,
           ymax=0.4,
           linewidth=2.0,
           color="red");
```



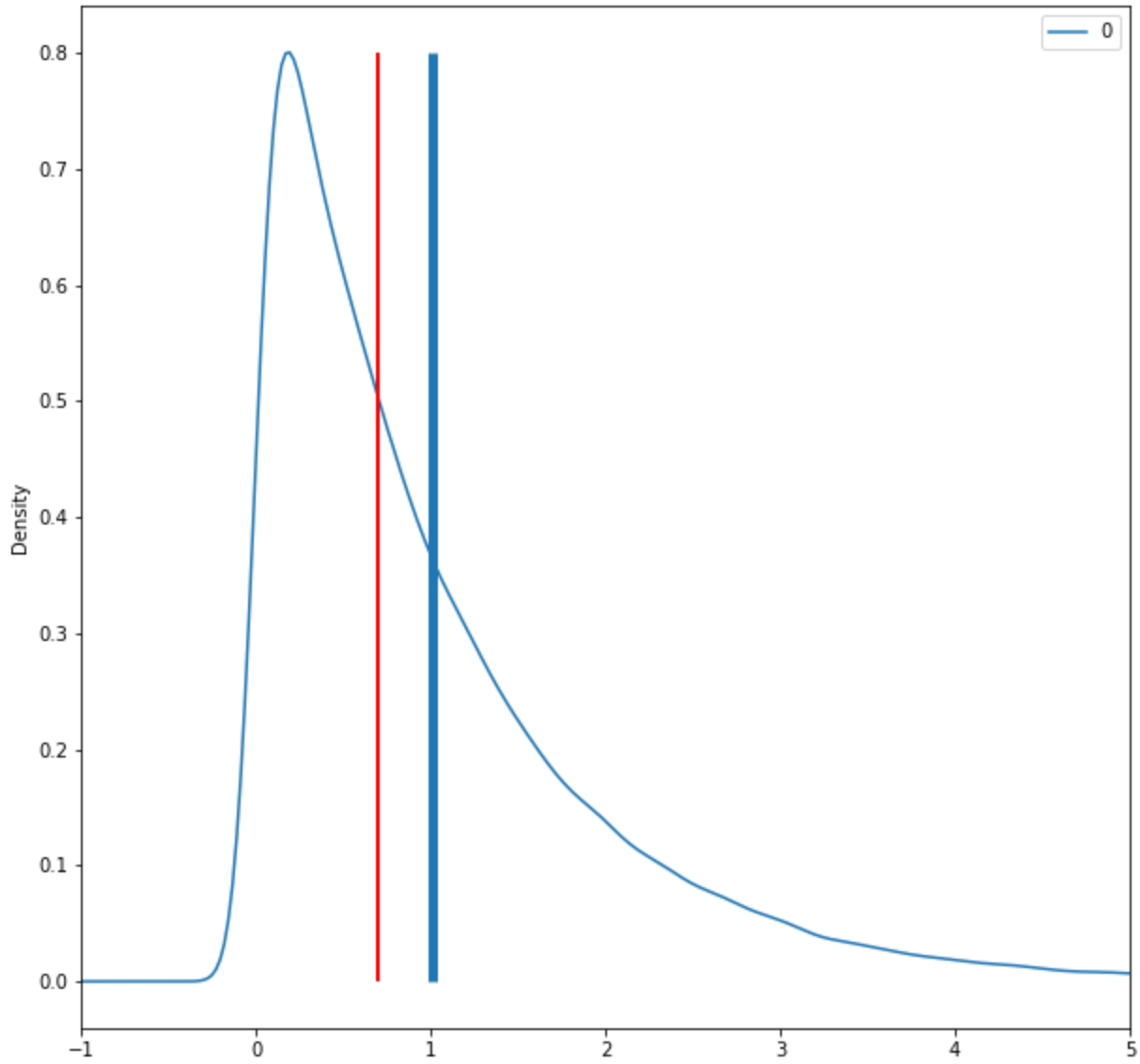
In [145…

```
skewed_data=pd.DataFrame(np.random.exponential(size=100000))

skewed_data.plot(kind="density",
                 figsize=(10,10),
                 xlim=(-1,5));


plt.vlines(skewed_data.mean(),      # Plot black line at mean
           ymin=0,
           ymax=0.8,
           linewidth=5.0);

plt.vlines(skewed_data.median(),    # Plot red line at median
           ymin=0,
           ymax=0.8,
           linewidth=2.0,
           color="red");
```



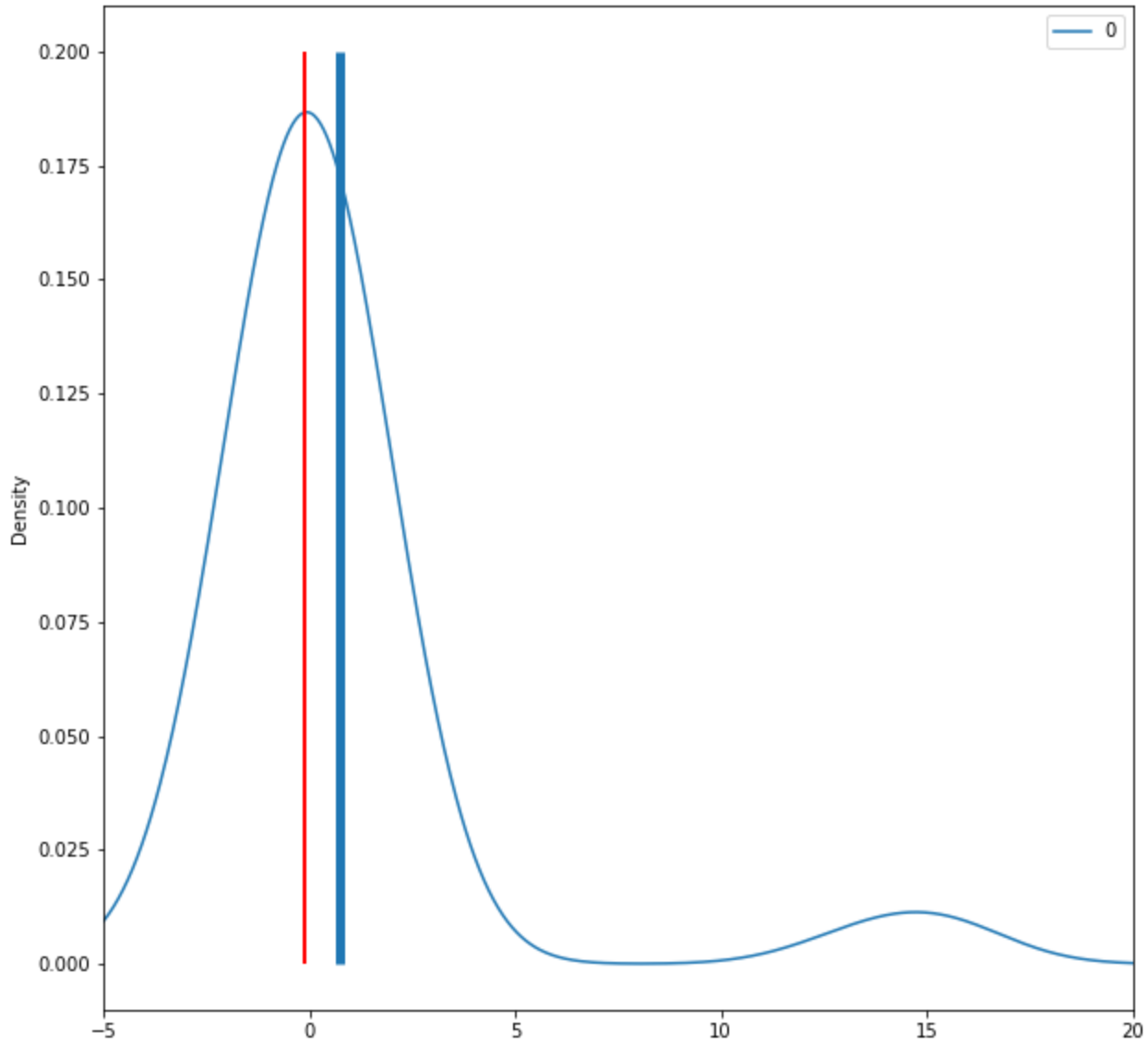
In [146…

```
norm_data = np.random.normal(size=50)
outliers = np.random.normal(15, size=3)
combined_data = pd.DataFrame(np.concatenate((norm_data, outliers), axis=0))

combined_data.plot(kind="density",
                   figsize=(10,10),
                   xlim=(-5,20));


plt.vlines(combined_data.mean(),      # Plot black line at mean
           ymin=0,
           ymax=0.2,
           linewidth=5.0);

plt.vlines(combined_data.median(),    # Plot red line at median
           ymin=0,
           ymax=0.2,
           linewidth=2.0,
           color="red");
```



In [147…
```
max(df["RowNumber"]) - min(df["RowNumber"])
```

Out[147]:
```
9999
```

In [148…
```
five_num = [df["RowNumber"].quantile(0),
            df["RowNumber"].quantile(0.25),
            df["RowNumber"].quantile(0.50),
            df["RowNumber"].quantile(0.75),
            df["RowNumber"].quantile(1)]

five_num
```

Out[148]:
```
[1.0, 2500.75, 5000.5, 7500.25, 10000.0]
```

In [149…
```
df["RowNumber"].describe()
```

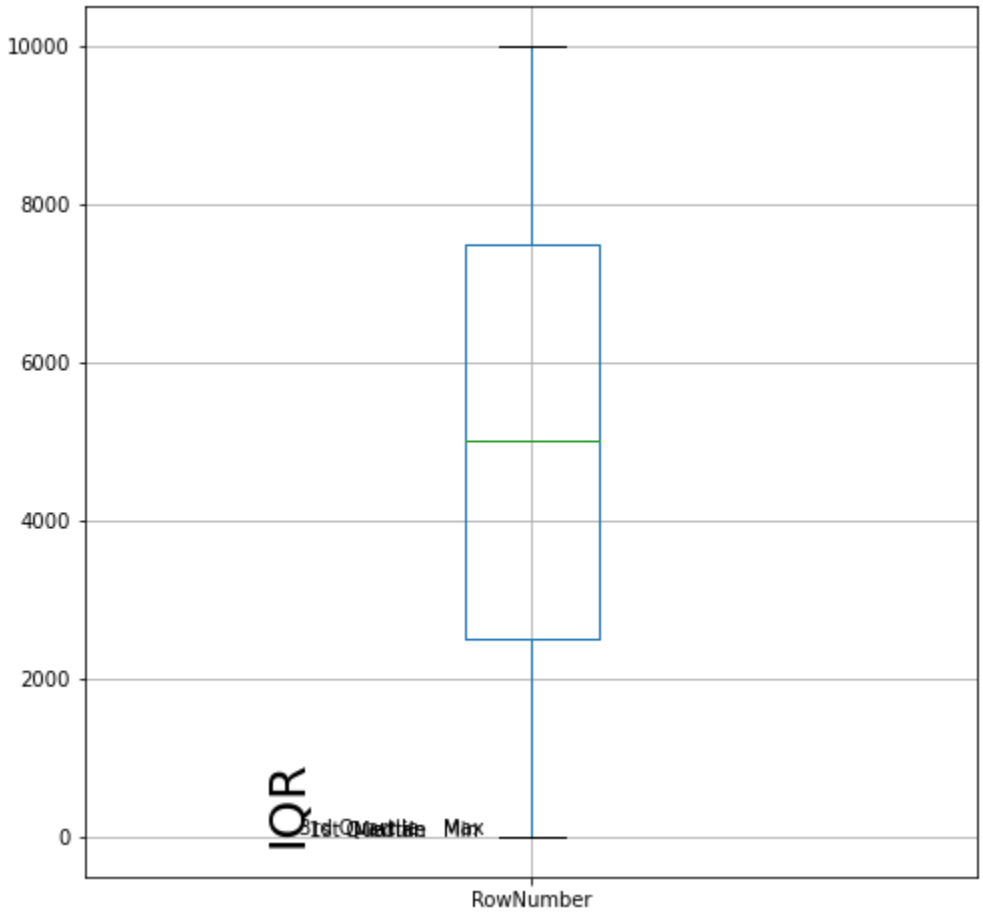
Out[149]:
```
count    10000.00000
mean      5000.50000
std       2886.89568
min          1.00000
25%       2500.75000
50%       5000.50000
75%       7500.25000
max      10000.00000
Name: RowNumber, dtype: float64
```

In [150…
```
].quantile(0.75) - df["RowNumber"].quantile(0.25)
```

Out[150]: 4999.5

In [151…
```
df.boxplot(column="RowNumber",
                return_type='axes',
                figsize=(8,8))

plt.text(x=0.74, y=22.25, s="3rd Quartile")
plt.text(x=0.8, y=18.75, s="Median")
plt.text(x=0.75, y=15.5, s="1st Quartile")
plt.text(x=0.9, y=10, s="Min")
plt.text(x=0.9, y=33.5, s="Max")
plt.text(x=0.7, y=19.5, s="IQR", rotation=90, size=25);
```



In [152…
```
df["RowNumber"].var()
```

Out[152]: 8334166.666666667

In [153…
```
df["RowNumber"].std()
```

Out[153]: 2886.8956799071675

In [154…
```
### Handle Missing Values
```

In [155…
```
df.dtypes=='object'
```

Out[155]:
```
RowNumber          False
CustomerId         False
Surname             True
CreditScore        False
Geography           True
Gender              True
Age                False
Tenure             False
Balance            False
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

In [156…
```
num_var=df.columns[df.dtypes != 'object']
cat_var=df.columns[df.dtypes == 'object']
```

In [157…
```
print(num_var)
print(cat_var)
```

```
Index(['RowNumber', 'CustomerId', 'CreditScore', 'Age', 'Tenure', 'Balance',
       'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary',
       'Exited'],
      dtype='object')
Index(['Surname', 'Geography', 'Gender'], dtype='object')
```

In [158…
```
df[num_var]
```

Out[158]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMe
0	1	15634602	619	42	2	0.00	1	1	
1	2	15647311	608	41	1	83807.86	1	0	
2	3	15619304	502	42	8	159660.80	3	1	
3	4	15701354	699	39	1	0.00	2	0	
4	5	15737888	850	43	2	125510.82	1	1	
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	771	39	5	0.00	2	1	
9996	9997	15569892	516	35	10	57369.61	1	1	
9997	9998	15584532	709	36	7	0.00	1	0	
9998	9999	15682355	772	42	3	75075.31	2	1	
9999	10000	15628319	792	28	4	130142.79	1	1	

10000 rows × 11 columns

In [159…
```
df[num_var].isnull().sum()
```

Out[159]:
```
RowNumber          0
CustomerId         0
CreditScore        0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

In [160…
```
df[num_var].isnull().sum().sort_values(ascending=False)/len(df)
```

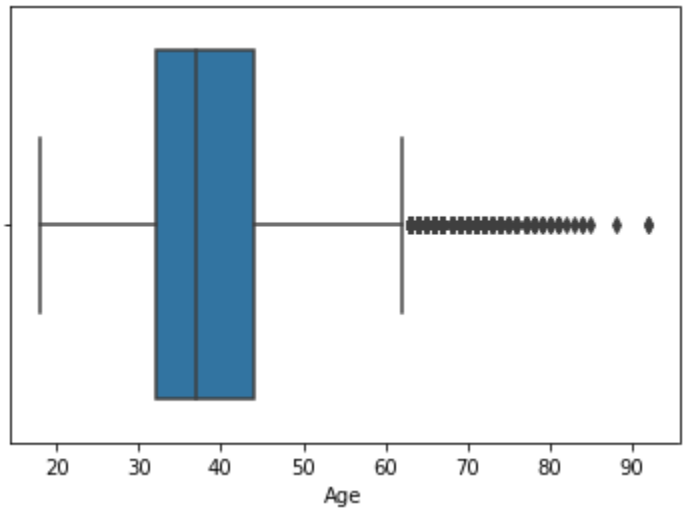
Out[160]:
```
RowNumber          0.0
CustomerId         0.0
CreditScore        0.0
Age                0.0
Tenure             0.0
Balance            0.0
NumOfProducts      0.0
HasCrCard          0.0
IsActiveMember     0.0
EstimatedSalary    0.0
Exited             0.0
dtype: float64
```

In [161…
```
### Find the Outliers and replace the Outliers
```

In [162…
```
sns.boxplot(df["Age"],data=df)
```

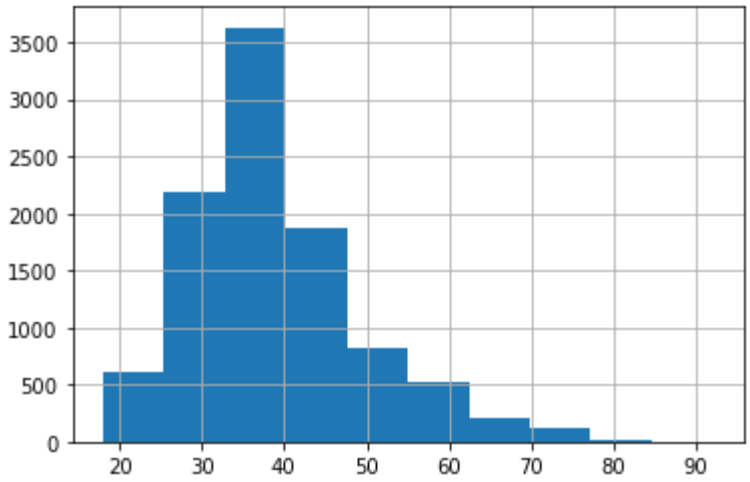
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[162]:
```
<AxesSubplot:xlabel='Age'>
```



In [163…
```
df['Age'].hist()
```

Out[163]:
```
<AxesSubplot:>
```



In [164…
```
df["Tenure"]=np.where(df["Tenure"] >10, np.median,df["Tenure"])
df["Tenure"]
```

Out[164]:
```
0        2
1        1
2        8
3        1
4        2
       ..
9995     5
9996    10
9997     7
9998     3
9999     4
Name: Tenure, Length: 10000, dtype: object
```

In [165…
```
df["Age"]=np.where(df["Age"] >10, np.median,df["Age"])
df["Age"]
```

Out[165]:
```
0       <function median at 0x000001D8E43B13A0>
1       <function median at 0x000001D8E43B13A0>
2       <function median at 0x000001D8E43B13A0>
3       <function median at 0x000001D8E43B13A0>
4       <function median at 0x000001D8E43B13A0>
                        ...
9995  <function median at 0x000001D8E43B13A0>
9996  <function median at 0x000001D8E43B13A0>
9997  <function median at 0x000001D8E43B13A0>
9998  <function median at 0x000001D8E43B13A0>
9999  <function median at 0x000001D8E43B13A0>
Name: Age, Length: 10000, dtype: object
```

In [166…
```
### Check for Categorical Columns and perform encoding
```

In [167…
```
pd.get_dummies(df, columns=["Gender","Age"], prefix=["Age", "Gender"]).head()
```

Out[167]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Tenure	Balance	NumOfProducts	HasCrCa
0	1	15634602	Hargrave	619	France	2	0.00	1	
1	2	15647311	Hill	608	Spain	1	83807.86	1	
2	3	15619304	Onio	502	France	8	159660.80	3	
3	4	15701354	Boni	699	France	1	0.00	2	
4	5	15737888	Mitchell	850	Spain	2	125510.82	1	

In [168…
```
### Split the data into dependent and independent variables
```

In [169…
```
I = df.iloc[:, :-6].values
I
```

Out[169]:
```
array([[1, 15634602, 'Hargrave', ..., 'Female',
        <function median at 0x000001D8E43B13A0>, 2],
       [2, 15647311, 'Hill', ..., 'Female',
        <function median at 0x000001D8E43B13A0>, 1],
       [3, 15619304, 'Onio', ..., 'Female',
        <function median at 0x000001D8E43B13A0>, 8],
       ...,
       [9998, 15584532, 'Liu', ..., 'Female',
        <function median at 0x000001D8E43B13A0>, 7],
       [9999, 15682355, 'Sabbatini', ..., 'Male',
        <function median at 0x000001D8E43B13A0>, 3],
       [10000, 15628319, 'Walker', ..., 'Female',
        <function median at 0x000001D8E43B13A0>, 4]], dtype=object)
```

In [170…
```
D = df.iloc[:, -2].values
D
```

Out[170]:
```
array([101348.88, 112542.58, 113931.57, ..., 42085.58,  92888.52,
        38190.78])
```

In [171…
```
### Scale the independent variables
```

In [172…
```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df[["RowNumber"]]=scaler.fit_transform(df[["RowNumber"]])
df
```

Out[172]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure
0	0.0000	15634602	Hargrave	619	France	Female	<function median at 0x000001D8E43B13A0>	2
1	0.0001	15647311	Hill	608	Spain	Female	<function median at 0x000001D8E43B13A0>	1
2	0.0002	15619304	Onio	502	France	Female	<function median at 0x000001D8E43B13A0>	8
3	0.0003	15701354	Boni	699	France	Female	<function median at 0x000001D8E43B13A0>	1
4	0.0004	15737888	Mitchell	850	Spain	Female	<function median at 0x000001D8E43B13A0>	2
...	...	...	...	...	...	...	...	...
9995	0.9996	15606229	Obijiaku	771	France	Male	<function median at 0x000001D8E43B13A0>	5
9996	0.9997	15569892	Johnstone	516	France	Male	<function median at 0x000001D8E43B13A0>	10
9997	0.9998	15584532	Liu	709	France	Female	<function median at 0x000001D8E43B13A0>	7
9998	0.9999	15682355	Sabbatini	772	Germany	Male	<function median at 0x000001D8E43B13A0>	3
9999	1.0000	15628319	Walker	792	France	Female	<function median at 0x000001D8E43B13A0>	4

10000 rows × 14 columns

In [173…
```
### Split the data into training and testing
```

In [174…
```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
df = pd.read_csv(r"E:\Assignments of ibm\Ass 2\Datasets\Churn_Modelling.csv")
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.05, random_state=0)
```

In [175…
```
X_train
```

Out[175]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
799	800	15567367	Tao	601	Germany	Female	42	9	133636.16
1069	1070	15628674	Iadanza	844	France	Male	40	7	113348.14
8410	8411	15609913	Clark	743	France	Female	46	9	0.00
9436	9437	15771000	Powell	684	France	Male	38	4	0.00
5099	5100	15731555	Ross-Watt	595	Germany	Female	45	9	106000.12
...	...	...	...	...	...	...	...	...	...
9225	9226	15584928	Ugochukwutubelum	594	Germany	Female	32	4	120074.97
4859	4860	15647111	White	794	Spain	Female	22	4	114440.24
3264	3265	15574372	Hoolan	738	France	Male	35	5	161274.05
9845	9846	15664035	Parsons	590	Spain	Female	38	9	0.00
2732	2733	15592816	Udokamma	623	Germany	Female	48	1	108076.33

9500 rows × 13 columns

In [176…
```
y_train
```

Out[176]:
```
799     0
1069    1
8410    0
9436    0
5099    1
       ..
9225    0
4859    0
3264    0
9845    0
2732    1
Name: Exited, Length: 9500, dtype: int64
```

In [ ]:

In [ ]: