In [1]:
```
import pandas as pd
import numpy as np
from scipy import stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
style.use('ggplot')
%matplotlib inline
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
```

In [2]:
```
#investigate data frame
data = pd.read_csv('train.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	id	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	Nu
0	0	15674932	Okwudilichukwu	668	France	Male	33.0	3	0.00	
1	1	15749177	Okwudiliolisa	627	France	Male	33.0	1	0.00	
2	2	15694510	Hsueh	678	France	Male	40.0	10	0.00	
3	3	15741417	Kao	581	France	Male	34.0	2	148882.54	
4	4	15766172	Chiemenam	716	Spain	Male	33.0	5	0.00	

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 165034 entries, 0 to 165033
Data columns (total 14 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   id               165034 non-null  int64
 1   CustomerId       165034 non-null  int64
 2   Surname          165034 non-null  object
 3   CreditScore      165034 non-null  int64
 4   Geography        165034 non-null  object
 5   Gender           165034 non-null  object
 6   Age              165034 non-null  float64
 7   Tenure           165034 non-null  int64
 8   Balance          165034 non-null  float64
 9   NumOfProducts    165034 non-null  int64
 10  HasCrCard        165034 non-null  float64
 11  IsActiveMember   165034 non-null  float64
 12  EstimatedSalary  165034 non-null  float64
 13  Exited           165034 non-null  int64
dtypes: float64(5), int64(6), object(3)
memory usage: 17.6+ MB
```

In [5]:
```
testdata = pd.read_csv('train.csv')
```

In [6]:
```
testdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 165034 entries, 0 to 165033
Data columns (total 14 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   id               165034 non-null  int64  
 1   CustomerId       165034 non-null  int64  
 2   Surname          165034 non-null  object 
 3   CreditScore      165034 non-null  int64  
 4   Geography        165034 non-null  object 
 5   Gender           165034 non-null  object 
 6   Age              165034 non-null  float64
 7   Tenure           165034 non-null  int64  
 8   Balance          165034 non-null  float64
 9   NumOfProducts    165034 non-null  int64  
 10  HasCrCard        165034 non-null  float64
 11  IsActiveMember   165034 non-null  float64
 12  EstimatedSalary  165034 non-null  float64
 13  Exited           165034 non-null  int64  
dtypes: float64(5), int64(6), object(3)
memory usage: 17.6+ MB
```

In [7]:
```
df = pd.concat([data,testdata])
```

In [8]:
```
df.to_csv('comb_churn.csv')
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 330068 entries, 0 to 165033
Data columns (total 14 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   id               330068 non-null  int64  
 1   CustomerId       330068 non-null  int64  
 2   Surname          330068 non-null  object 
 3   CreditScore      330068 non-null  int64  
 4   Geography        330068 non-null  object 
 5   Gender           330068 non-null  object 
 6   Age              330068 non-null  float64
 7   Tenure           330068 non-null  int64  
 8   Balance          330068 non-null  float64
 9   NumOfProducts    330068 non-null  int64  
 10  HasCrCard        330068 non-null  float64
 11  IsActiveMember   330068 non-null  float64
 12  EstimatedSalary  330068 non-null  float64
 13  Exited           330068 non-null  int64  
dtypes: float64(5), int64(6), object(3)
memory usage: 37.8+ MB
```

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
id                 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
```

In [11]:
```
df.Exited.value_counts()
```

Out[11]:
```
Exited
0    260226
1     69842
Name: count, dtype: int64
```

In [12]:
```
df.Geography.value_counts()
```

Out[12]:
```
Geography
France     188430
Spain       72426
Germany     69212
Name: count, dtype: int64
```

In [13]:
```
df.fillna(-1).groupby(df['Geography'], dropna=True)['Exited'].value_counts()
```

Out[13]:
```
Geography  Exited
France     0         157286
           1          31144
Germany    0          42984
           1          26228
Spain      0          59956
           1          12470
Name: count, dtype: int64
```

In [14]:
```
# Recordkeeping is inaccurate with 40% of customer files not showing if they ar
# This is also when investigated across the geographical areas:
# France = 40%
# Germany = 39.9%
# Spain = 39.7%
```

In [15]:
```
# I would assume that the records on exited customes is correct and the NaN are
# Because we do not have the confirmation of record keeping we will need to dro
# This should be reviewed
```

In [16]:
```
cln = df.dropna()
```

In [17]:
```
cln.isnull().sum()
```

Out[17]:
```
id                 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
```

In [18]:
```
cln.info()
```

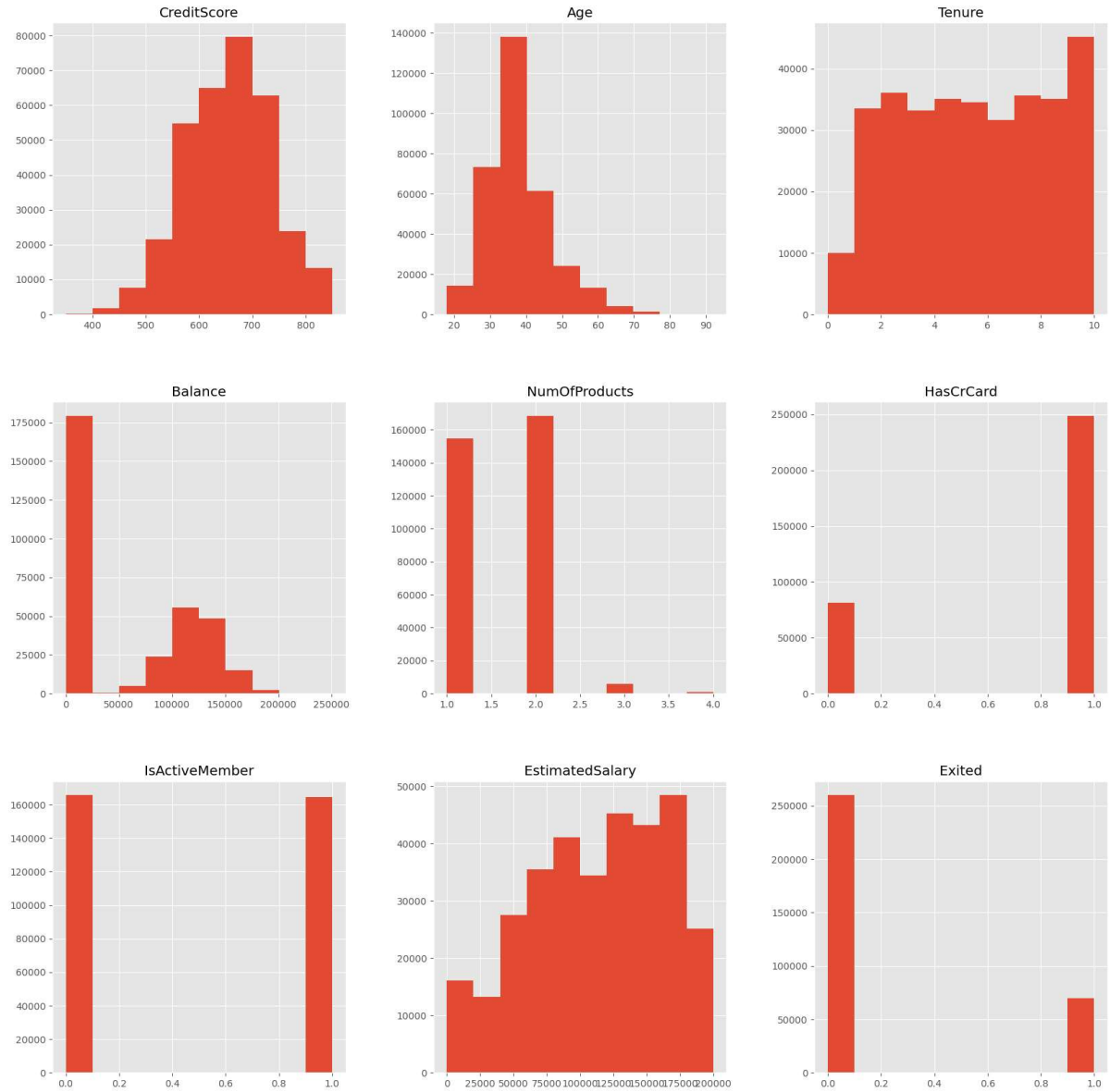
```
<class 'pandas.core.frame.DataFrame'>
Index: 330068 entries, 0 to 165033
Data columns (total 14 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   id               330068 non-null  int64
 1   CustomerId       330068 non-null  int64
 2   Surname          330068 non-null  object
 3   CreditScore      330068 non-null  int64
 4   Geography        330068 non-null  object
 5   Gender           330068 non-null  object
 6   Age              330068 non-null  float64
 7   Tenure           330068 non-null  int64
 8   Balance          330068 non-null  float64
 9   NumOfProducts    330068 non-null  int64
 10  HasCrCard        330068 non-null  float64
 11  IsActiveMember   330068 non-null  float64
 12  EstimatedSalary  330068 non-null  float64
 13  Exited           330068 non-null  int64
dtypes: float64(5), int64(6), object(3)
memory usage: 37.8+ MB
```

In [19]:
```
cln.select_dtypes(include="object").nunique()
```

Out[19]:
```
Surname      2797
Geography       3
Gender          2
dtype: int64
```

In [20]:
```
cln.drop(columns=['id', 'CustomerId', 'Surname'], inplace=True)
```

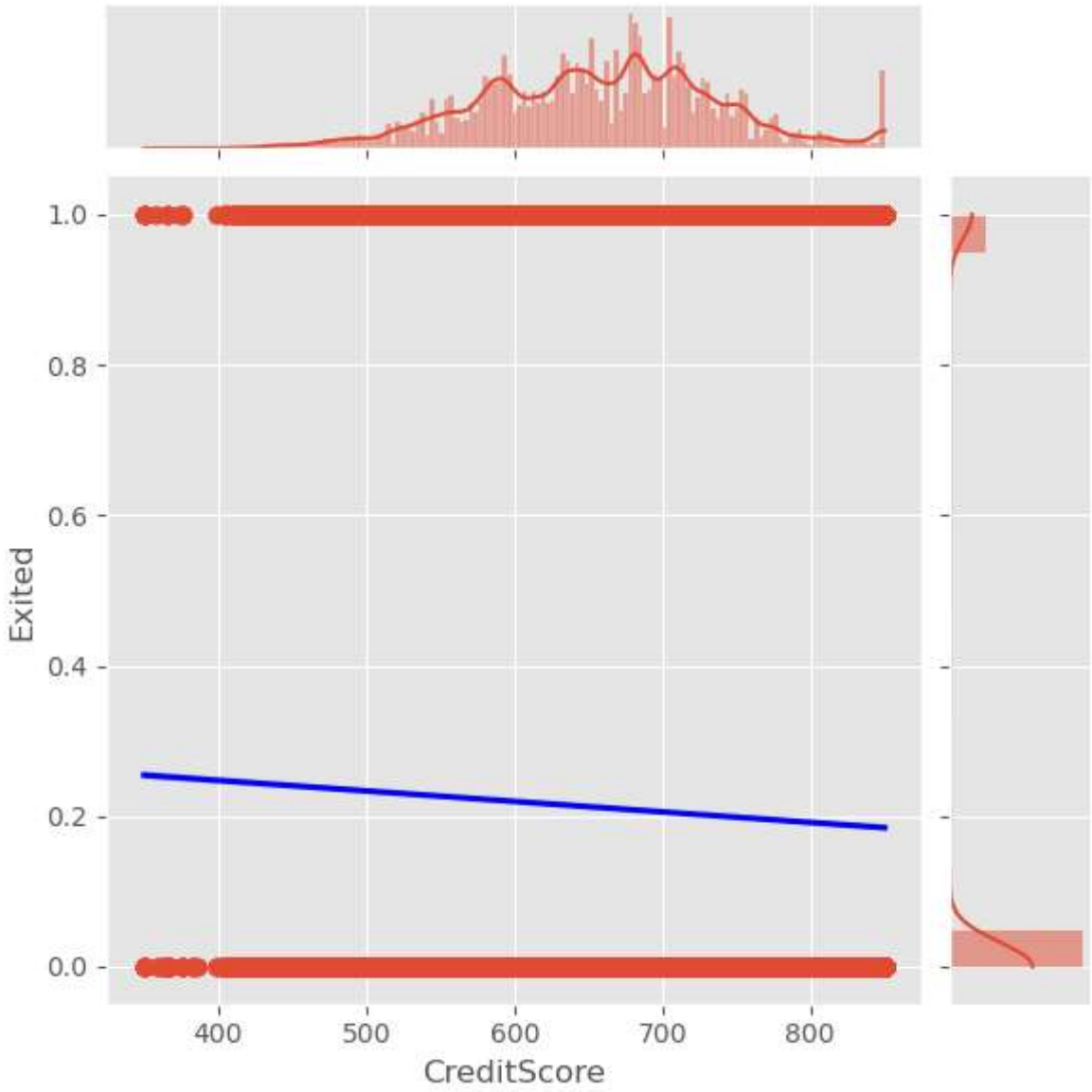
In [21]:
```
cln.hist(figsize=(20,20));
```



In [22]:
```
# likely categorical variables are Tenure, NumOfProducts, HasCrCard and IsActiv
```

In [23]:
```
sns.jointplot(data = cln, x= 'CreditScore', y ='Exited', kind='reg', joint_kws=
```

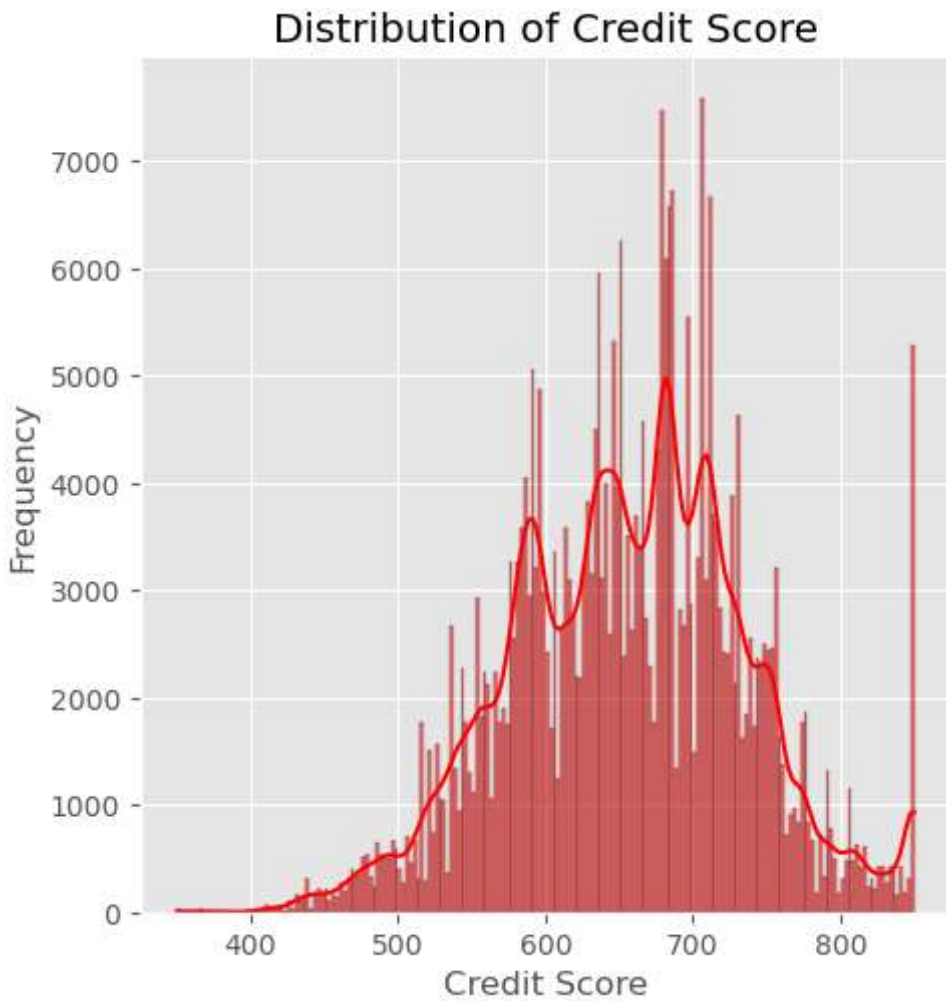
Out[23]: <seaborn.axisgrid.JointGrid at 0x2801d230f10>



In [24]:
```
# CreditScore has a semi normal distribution but has some outliers
```

In [25]:
```
sns.displot(cln['CreditScore'], bins=200, color='red', edgecolor='black', kde=T

plt.xlabel('Credit Score')
plt.ylabel('Frequency')
plt.title('Distribution of Credit Score')

plt.show()
```

```
C:\Users\p_ste\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

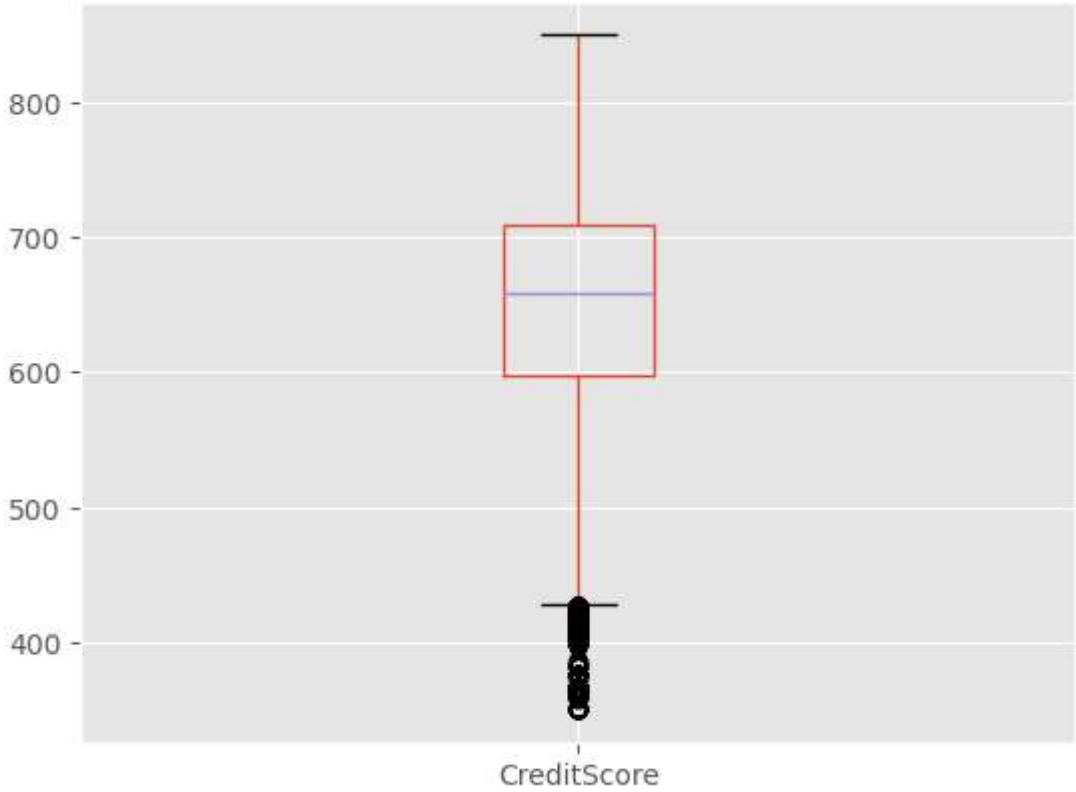


In [26]:
```
cln['CreditScore'].describe()
```

Out[26]:
```
count    330068.000000
mean        656.454373
std          80.103219
min         350.000000
25%         597.000000
50%         659.000000
75%         710.000000
max         850.000000
Name: CreditScore, dtype: float64
```

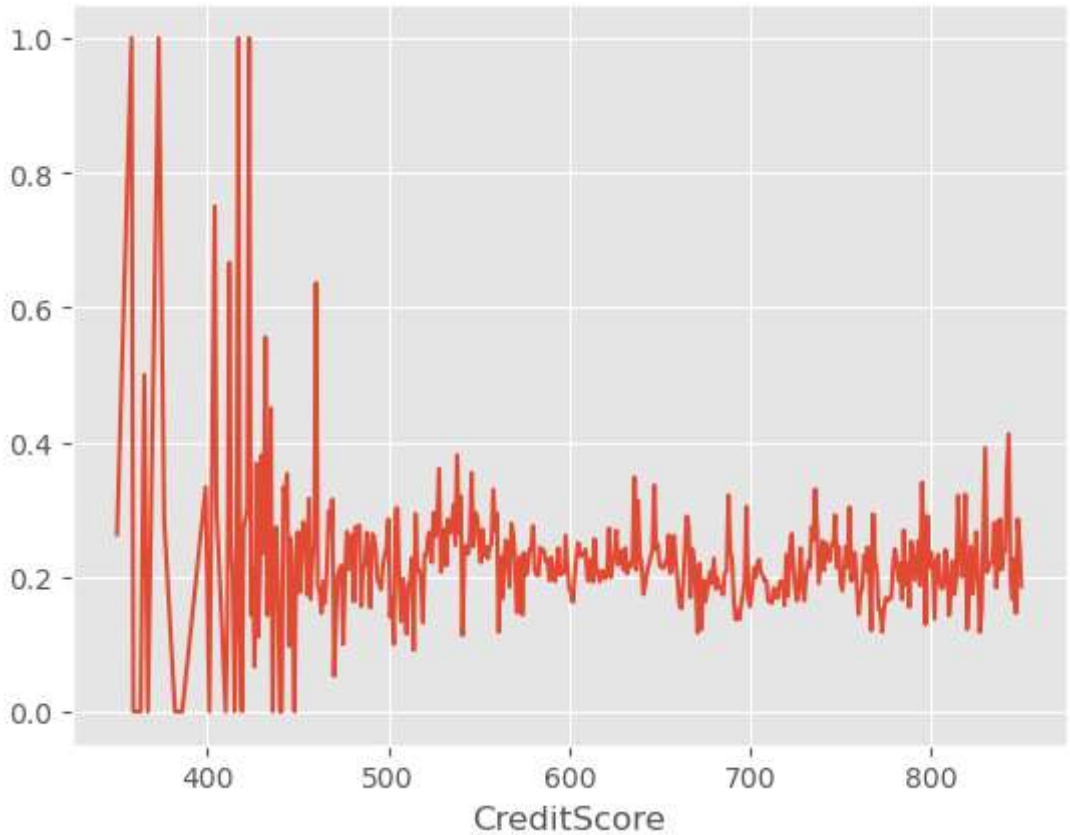
In [27]:
```
cln.boxplot(column=['CreditScore'])
```

Out[27]: <Axes: >



In [28]:
```
cln.groupby(['CreditScore'])['Exited'].mean().plot()
```

Out[28]: <Axes: xlabel='CreditScore'>



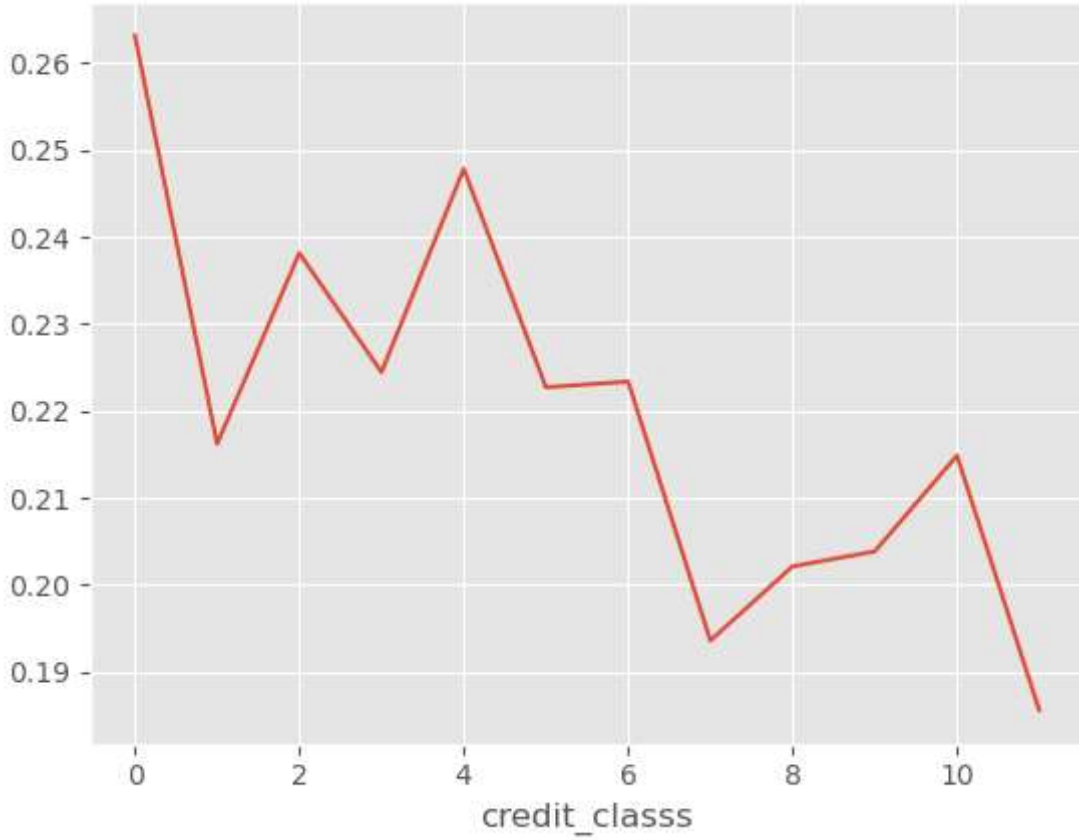
In [29]:
```
# after invetigating the credit score, this feature has a maximum of 850 and a
# given that most credit scores work with this range as a system, https://www.i
# we will create a categorical range variable with the following:
# Perfect = 850 = 10 ranging down to 0 = 350 = Minimum Score
```

In [30]:
```
def conditions(x):
    if   x >= 850:   return 11
    elif x > 800:   return 10
    elif x > 750:   return 9
    elif x > 700:   return 8
    elif x > 650:   return 7
    elif x > 600:   return 6
    elif x > 550:   return 5
    elif x > 500:   return 4
    elif x > 450:   return 3
    elif x > 400:   return 2
    elif x > 350:   return 1
    else:           return 0

func         = np.vectorize(conditions)
credit_class = func(cln["CreditScore"])
```

In [31]:
```
cln["credit_classs"] = credit_class
```

In [32]:
```
cln.groupby(['credit_classs'])['Exited'].mean().plot()
```

Out[32]: <Axes: xlabel='credit_classs'>



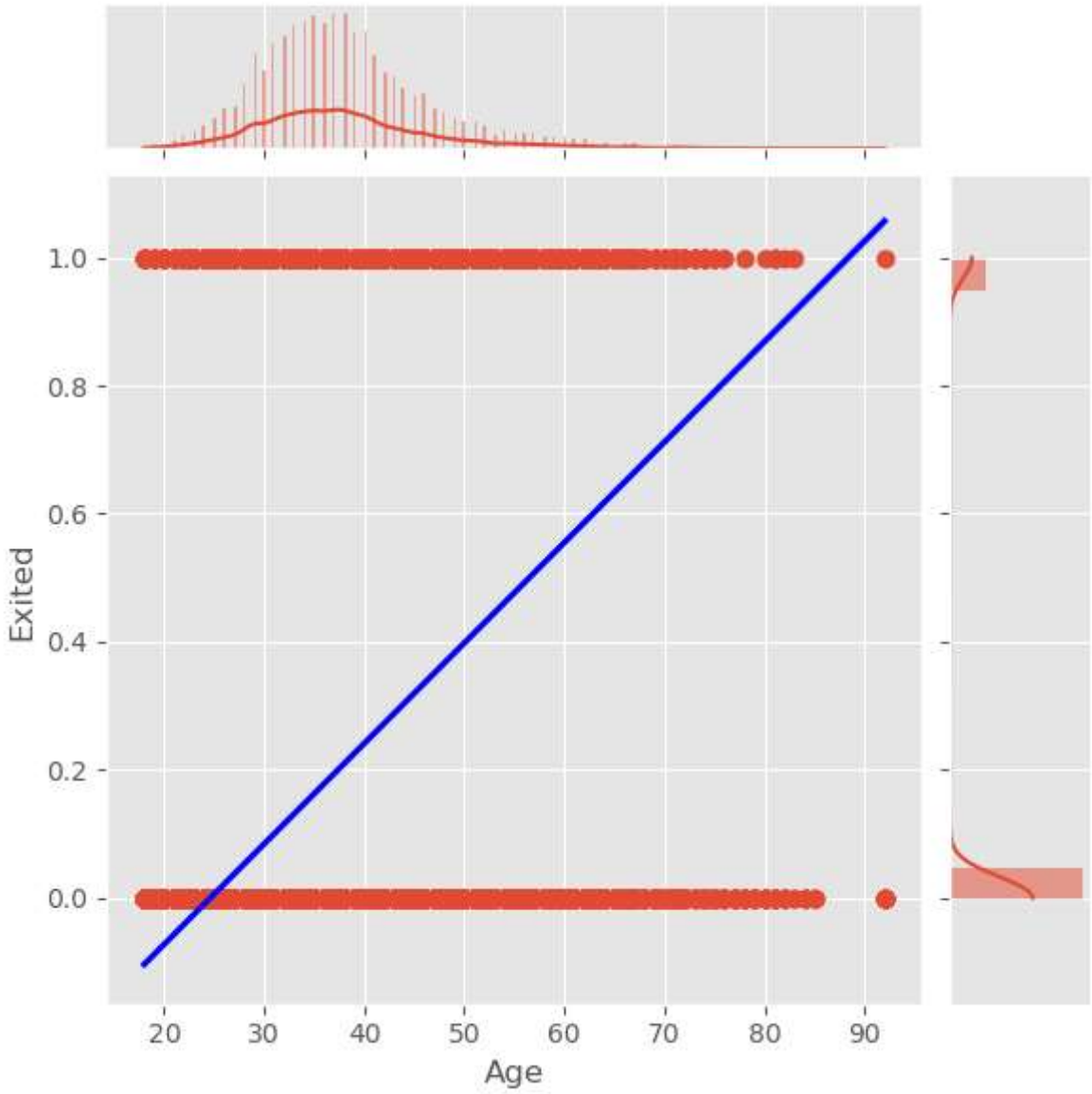
In [33]:
```
cln.head()
```

Out[33]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiv
0	668	France	Male	33.0	3	0.00	2	1.0	
1	627	France	Male	33.0	1	0.00	2	1.0	
2	678	France	Male	40.0	10	0.00	2	1.0	
3	581	France	Male	34.0	2	148882.54	1	1.0	
4	716	Spain	Male	33.0	5	0.00	2	1.0	

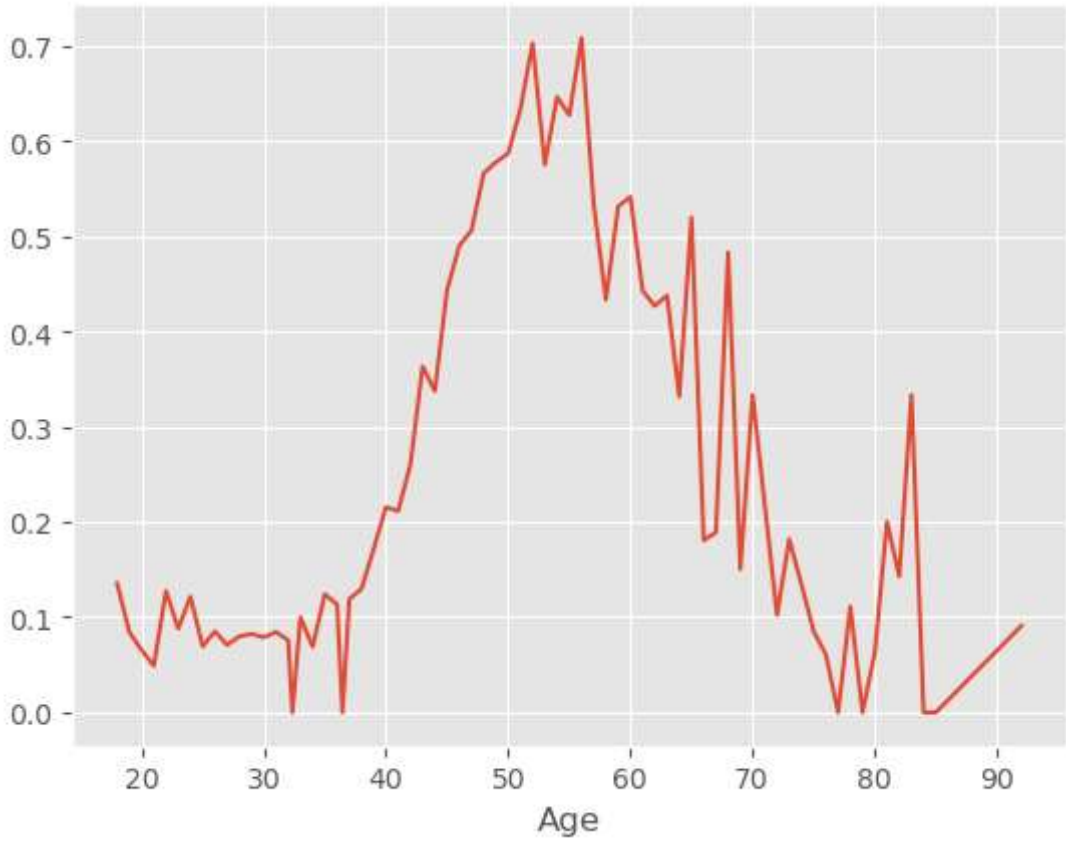
In [34]:
```
sns.jointplot(data = cln, x= 'Age', y ='Exited', kind='reg', joint_kws={'line_k
```

Out[34]: <seaborn.axisgrid.JointGrid at 0x280279e3890>



In [35]:
```
cln.groupby(['Age'])['Exited'].mean().plot()
```

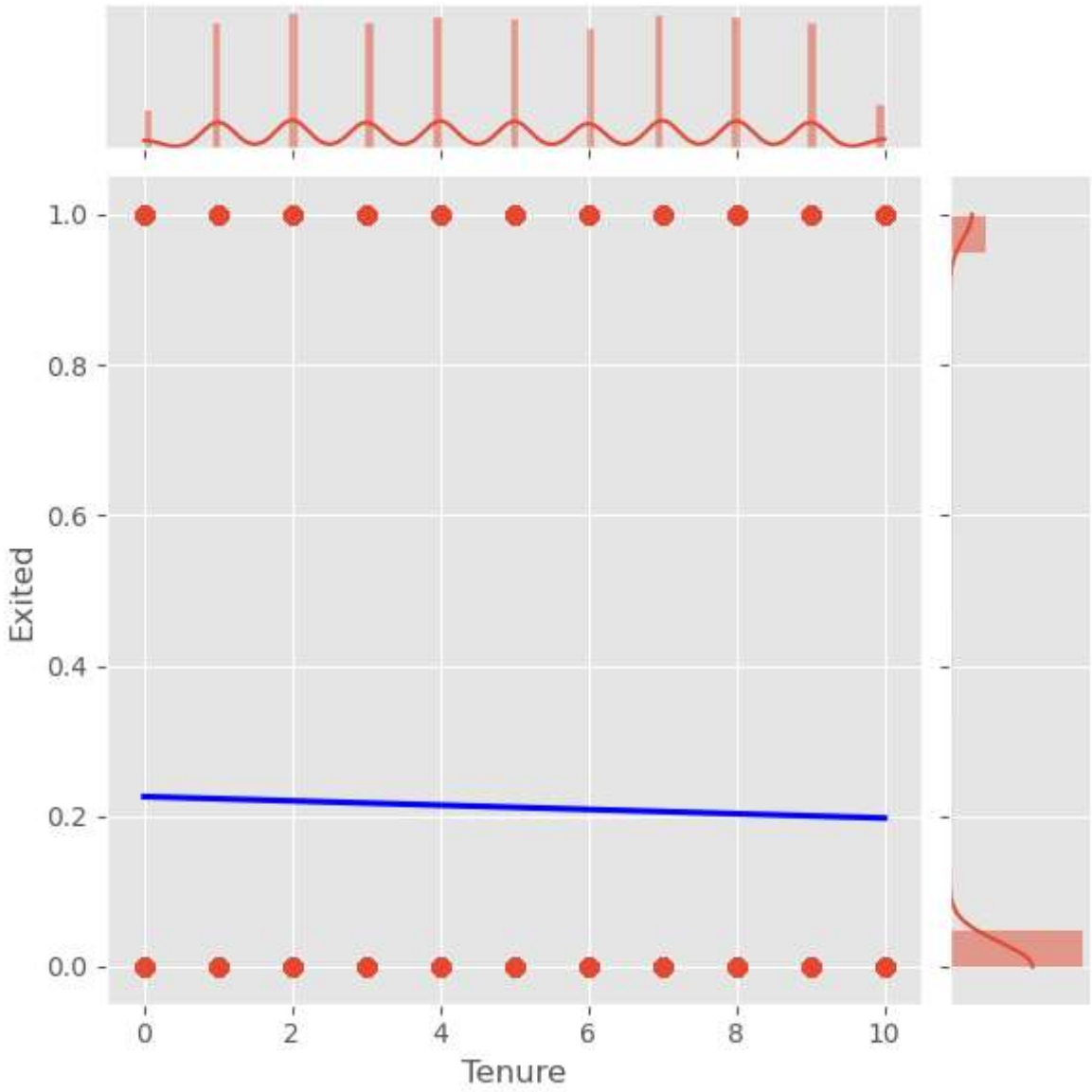
Out[35]: <Axes: xlabel='Age'>



In [36]:
```
# Age has a mostly normal ditribution with some slight skewing to the left.
# Improvement will be found with normalisation
```

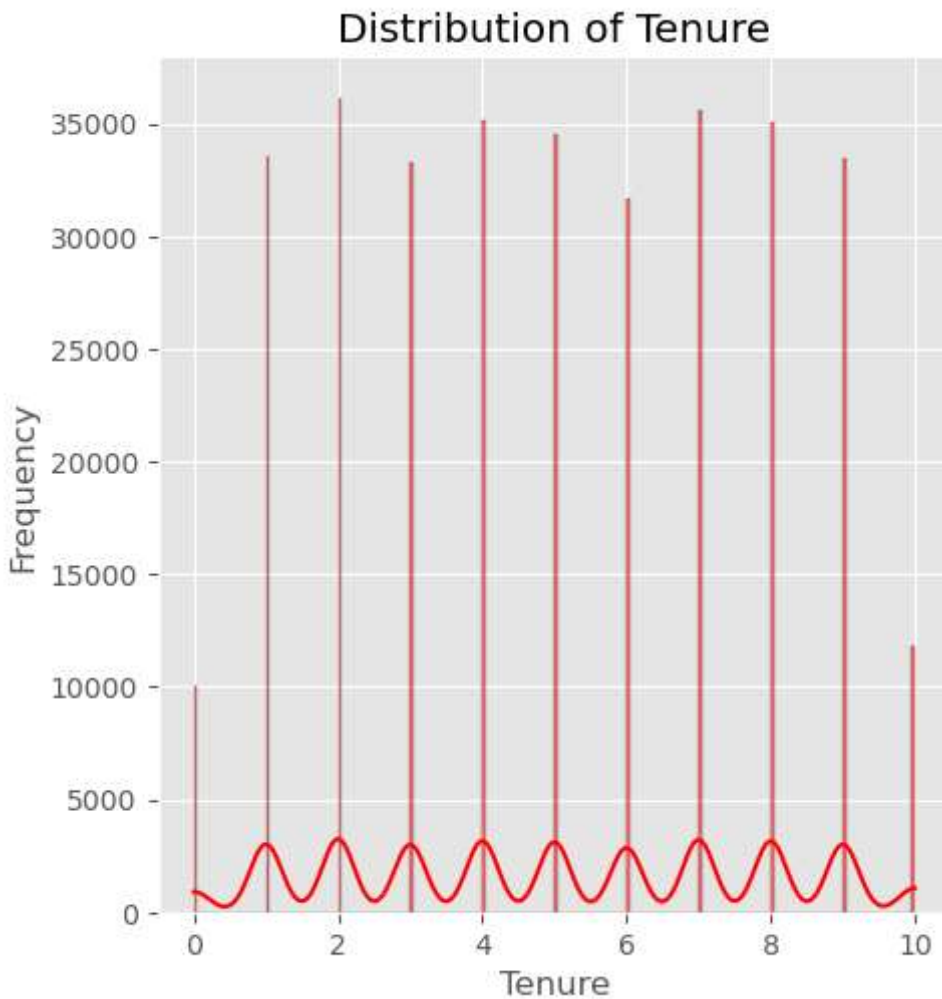
In [37]:
```
sns.jointplot(data = cln, x= 'Tenure', y ='Exited', kind='reg', joint_kws={'lir
```

Out[37]: <seaborn.axisgrid.JointGrid at 0x2801d20e590>



In [38]:
```
sns.displot(cln['Tenure'], bins=200, color='red', edgecolor='black', kde=True)

plt.xlabel('Tenure')
plt.ylabel('Frequency')
plt.title('Distribution of Tenure')

plt.show()
```

```
C:\Users\p_ste\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

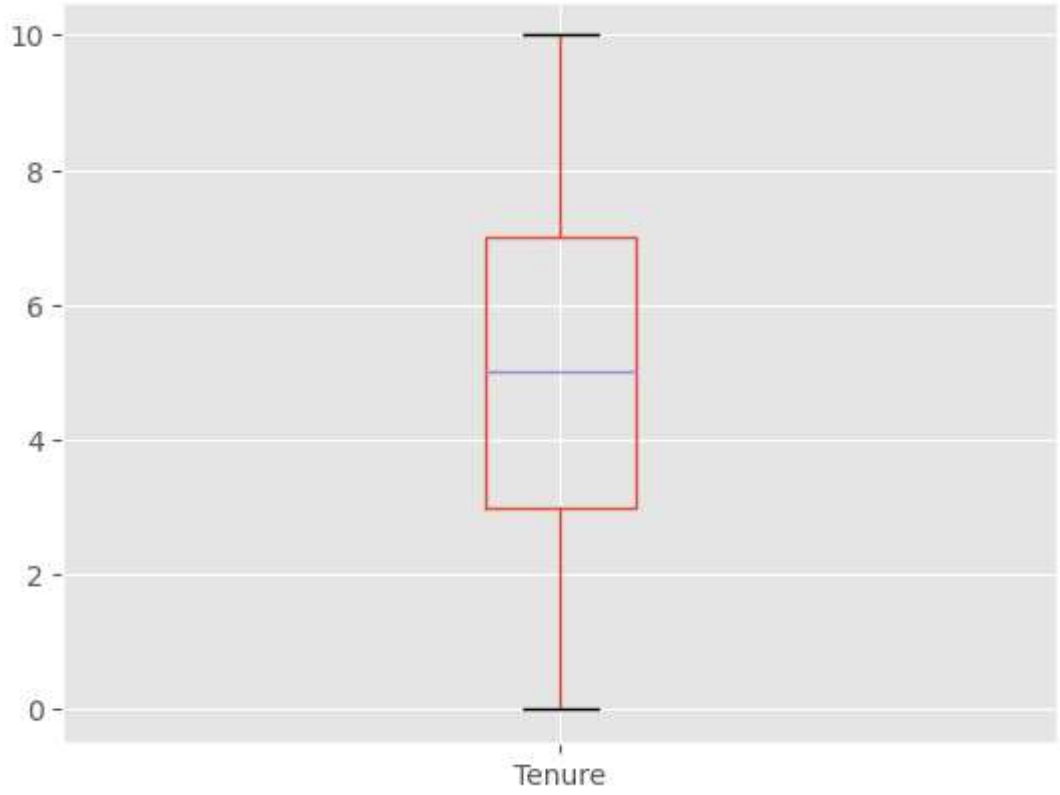


In [39]:
```
cln['Tenure'].describe()
```

Out[39]:
```
count    330068.000000
mean          5.020353
std           2.806154
min           0.000000
25%           3.000000
50%           5.000000
75%           7.000000
max          10.000000
Name: Tenure, dtype: float64
```

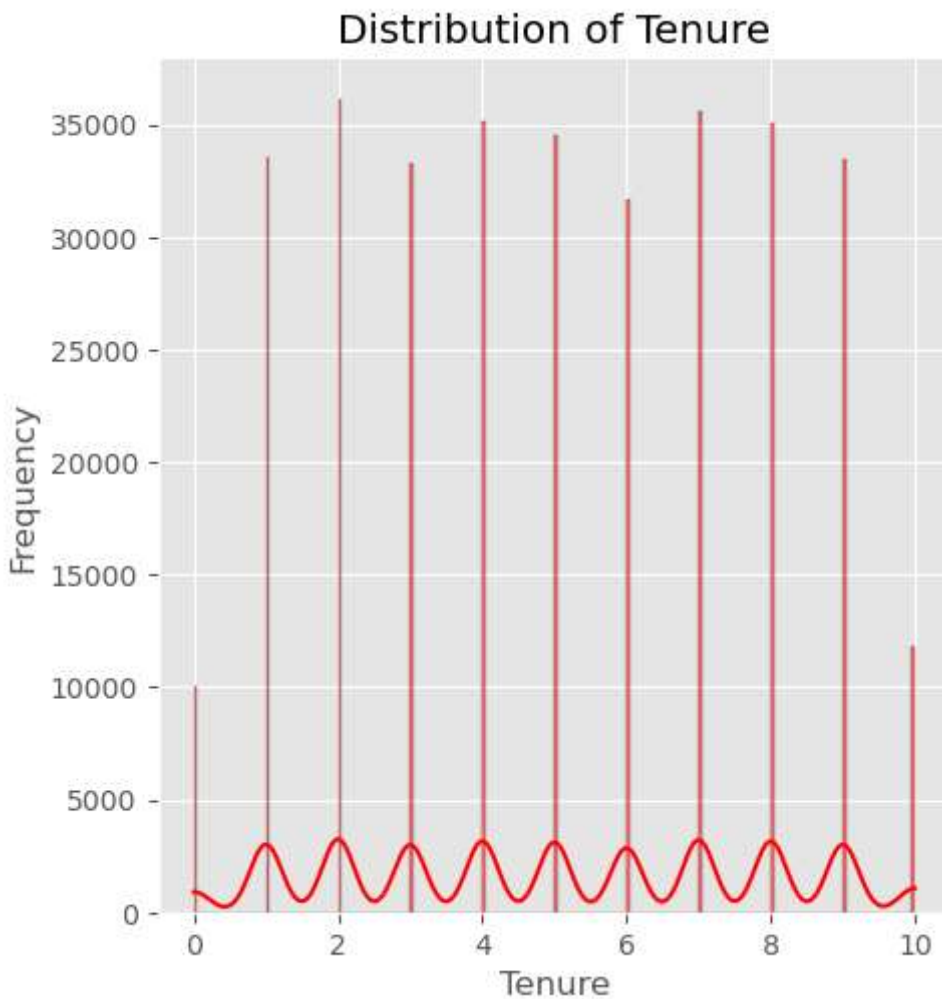
In [40]:
```
cln.boxplot(column=['Tenure'])
```

Out[40]: <Axes: >



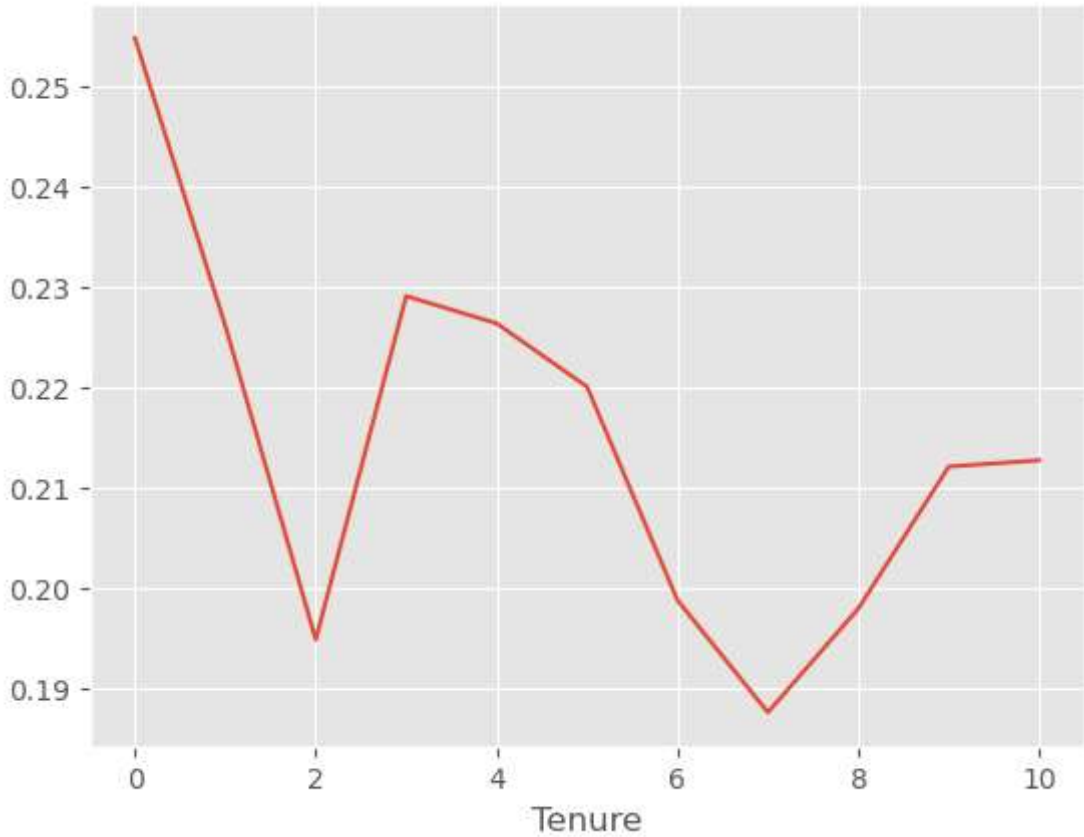
In [41]:
```
sns.displot(cln['Tenure'], bins=200, color='red', edgecolor='black', kde=True)

plt.xlabel('Tenure')
plt.ylabel('Frequency')
plt.title('Distribution of Tenure')

plt.show()
```

```
C:\Users\p_ste\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [42]:
```
cln.groupby(['Tenure'])['Exited'].mean().plot()
```

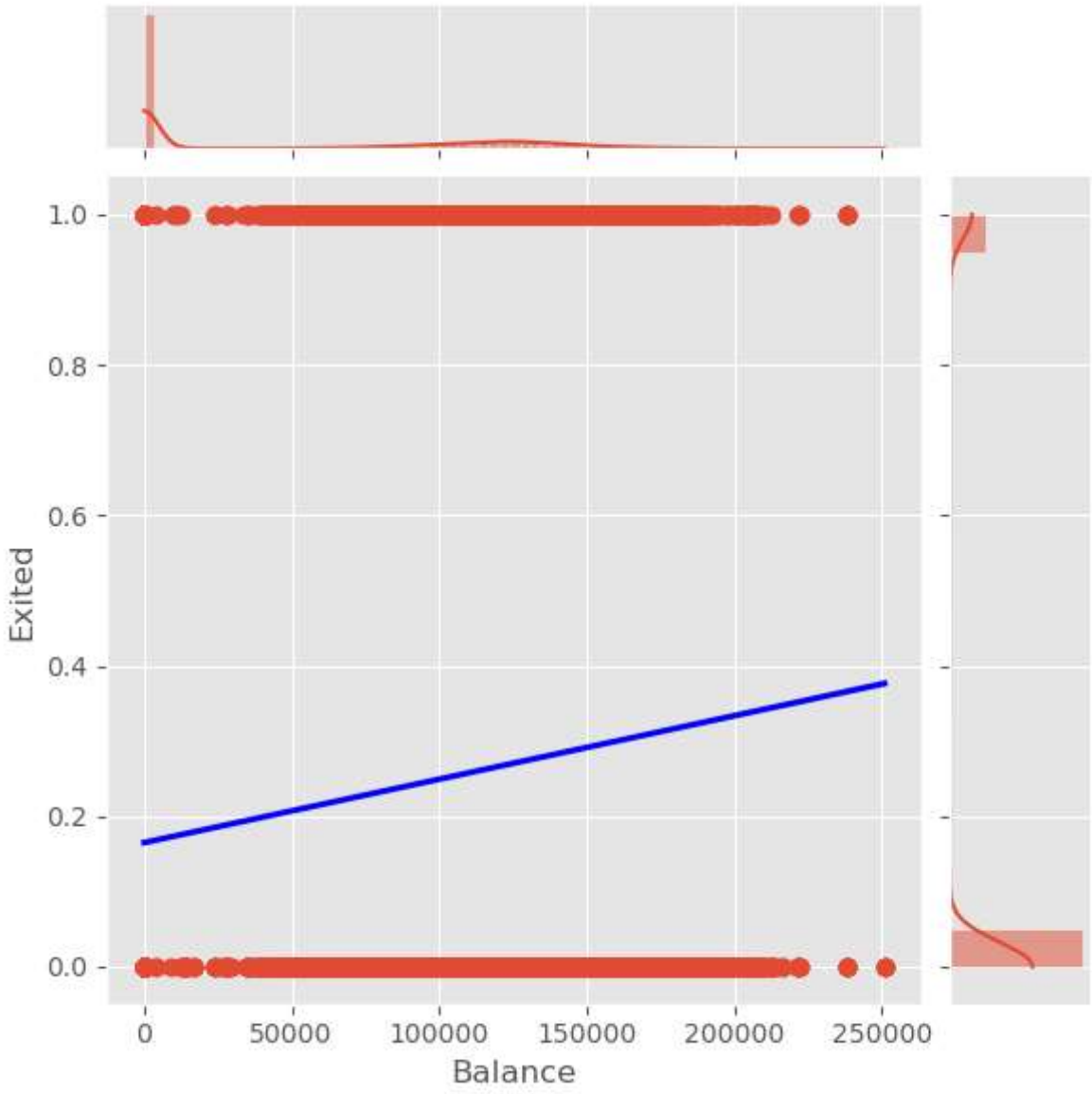
Out[42]: <Axes: xlabel='Tenure'>



In [43]:
```
# Unsure of the tenure definition but will use this as a categoricale variable.
```

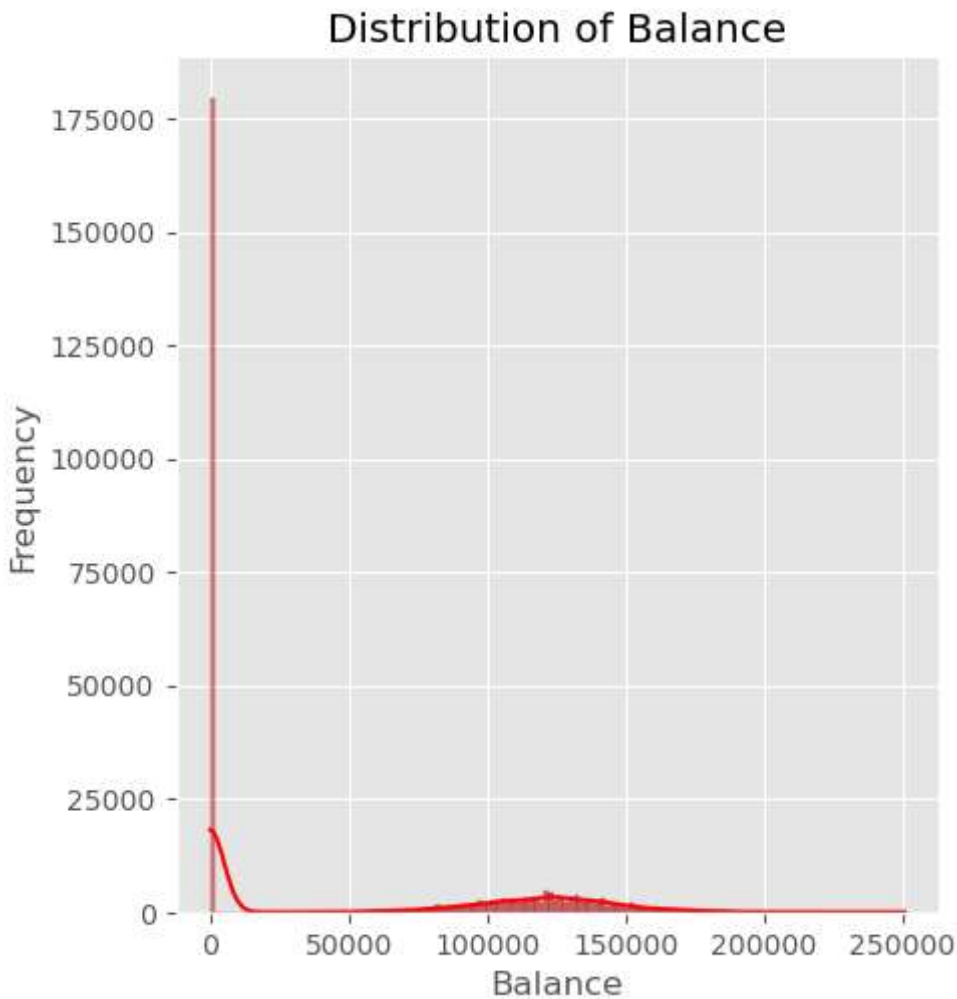
In [44]:
```
sns.jointplot(data = cln, x= 'Balance', y ='Exited', kind='reg', joint_kws={'li
```

Out[44]: <seaborn.axisgrid.JointGrid at 0x28025312750>



In [45]:
```
sns.displot(cln['Balance'], bins=200, color='red', edgecolor='black', kde=True)

plt.xlabel('Balance')
plt.ylabel('Frequency')
plt.title('Distribution of Balance')

plt.show()
```

```
C:\Users\p_ste\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [46]:
```
cln['Balance'].describe()
```

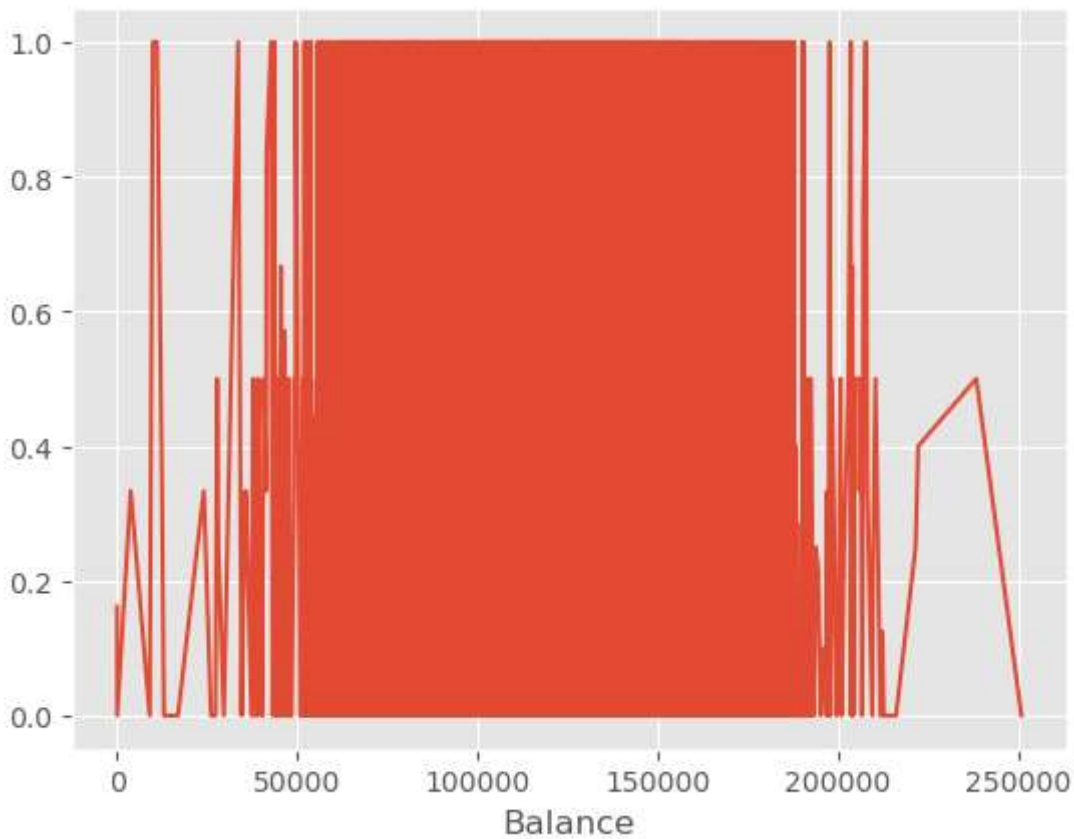
Out[46]:
```
count    330068.000000
mean      55478.086689
std       62817.568119
min           0.000000
25%           0.000000
50%           0.000000
75%      119939.590000
max      250898.090000
Name: Balance, dtype: float64
```

In [47]:
```
cln.Balance.value_counts()
```

Out[47]:
```
Balance
0.00         179296
124577.33       176
127864.40       128
122314.50       126
129855.32       118
                ...
125824.21         2
158741.56         2
126815.52         2
61172.57          2
110993.29         2
Name: count, Length: 30075, dtype: int64
```

In [48]:
```
cln.groupby(['Balance'])['Exited'].mean().plot()
```

Out[48]:
```
<Axes: xlabel='Balance'>
```

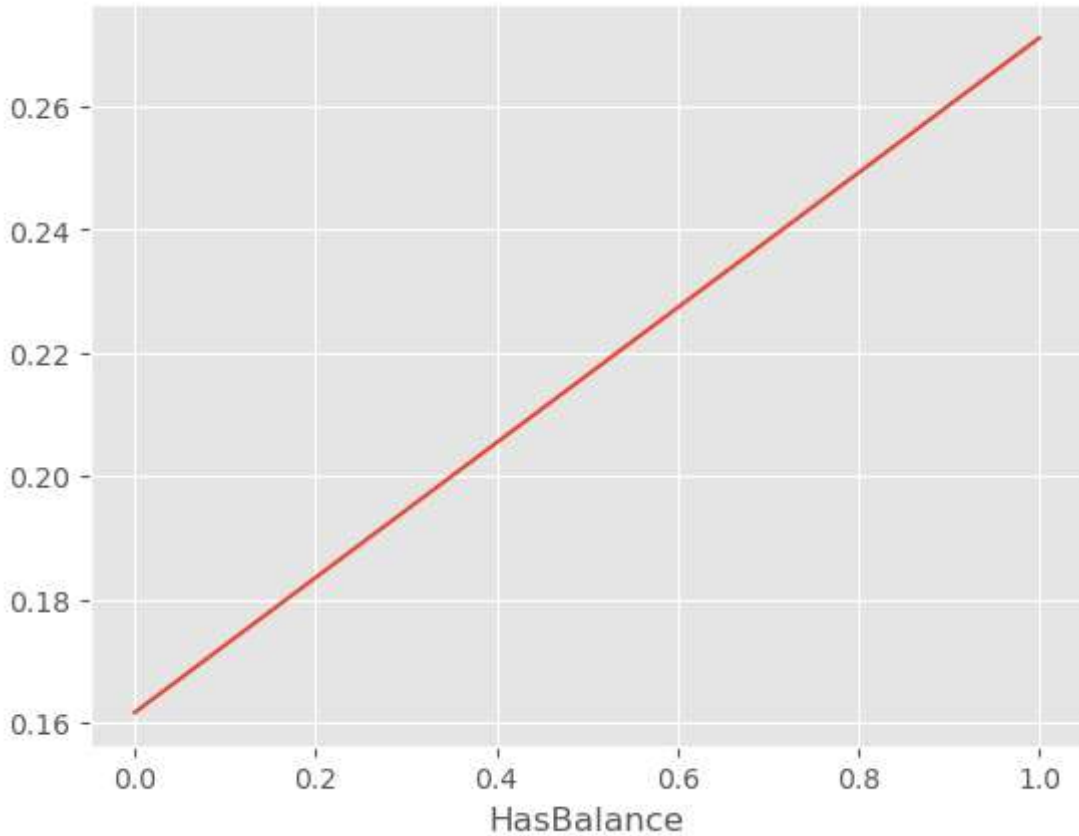


In [49]:
```
# create has balance feature to investigate significance of no balance as a cat
cln['HasBalance']= np.where(cln['Balance']==0,0,1)
```

In [50]:
```
# change balance feature to not include 0 for future calculations
cln['Balance'] = cln['Balance']+1
```

In [51]:
```
cln.groupby(['HasBalance'])['Exited'].mean().plot()
```

Out[51]: <Axes: xlabel='HasBalance'>



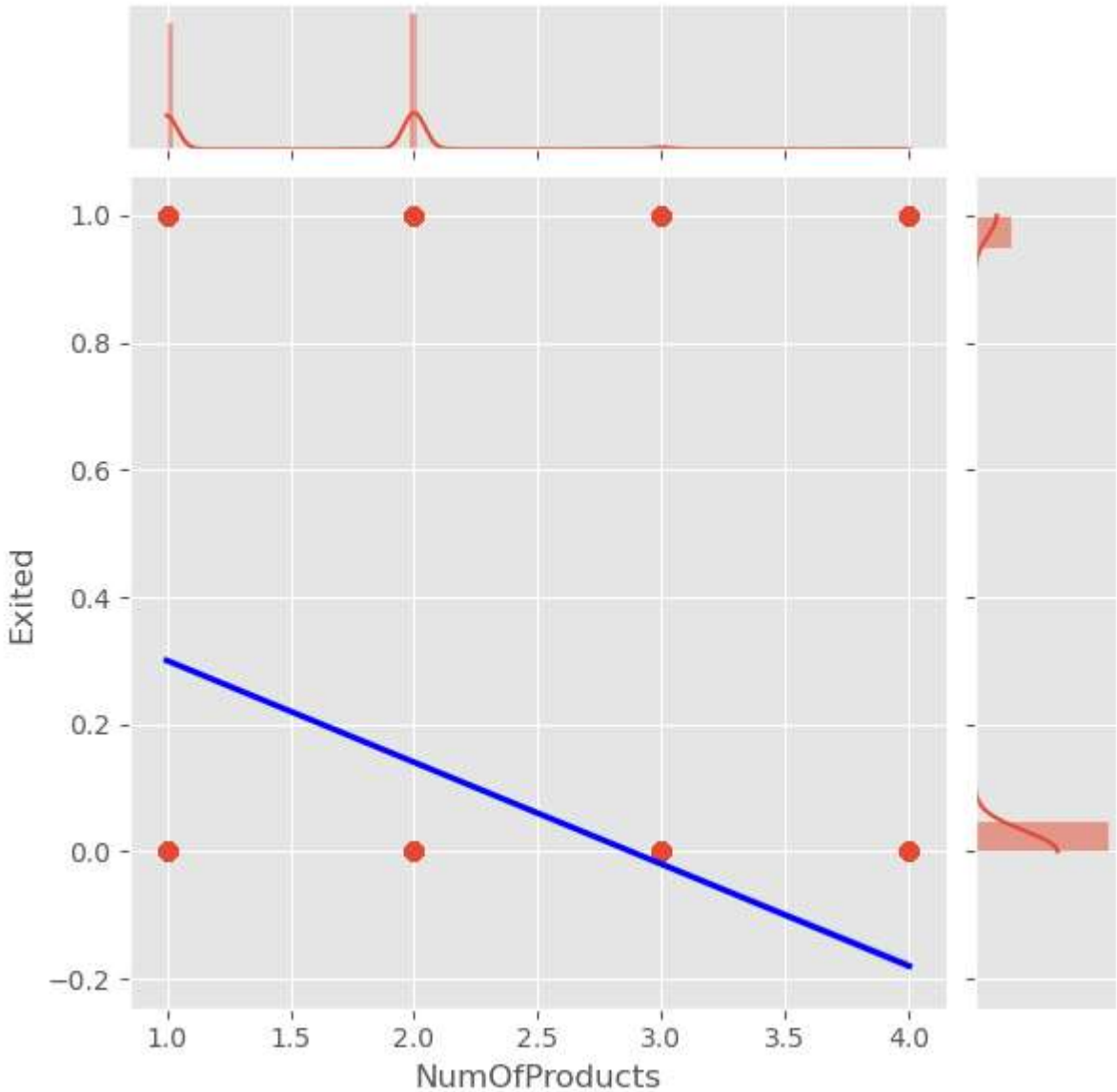
In [52]:
```
cln.head()
```

Out[52]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiv
0	668	France	Male	33.0	3	1.00	2	1.0	
1	627	France	Male	33.0	1	1.00	2	1.0	
2	678	France	Male	40.0	10	1.00	2	1.0	
3	581	France	Male	34.0	2	148883.54	1	1.0	
4	716	Spain	Male	33.0	5	1.00	2	1.0	

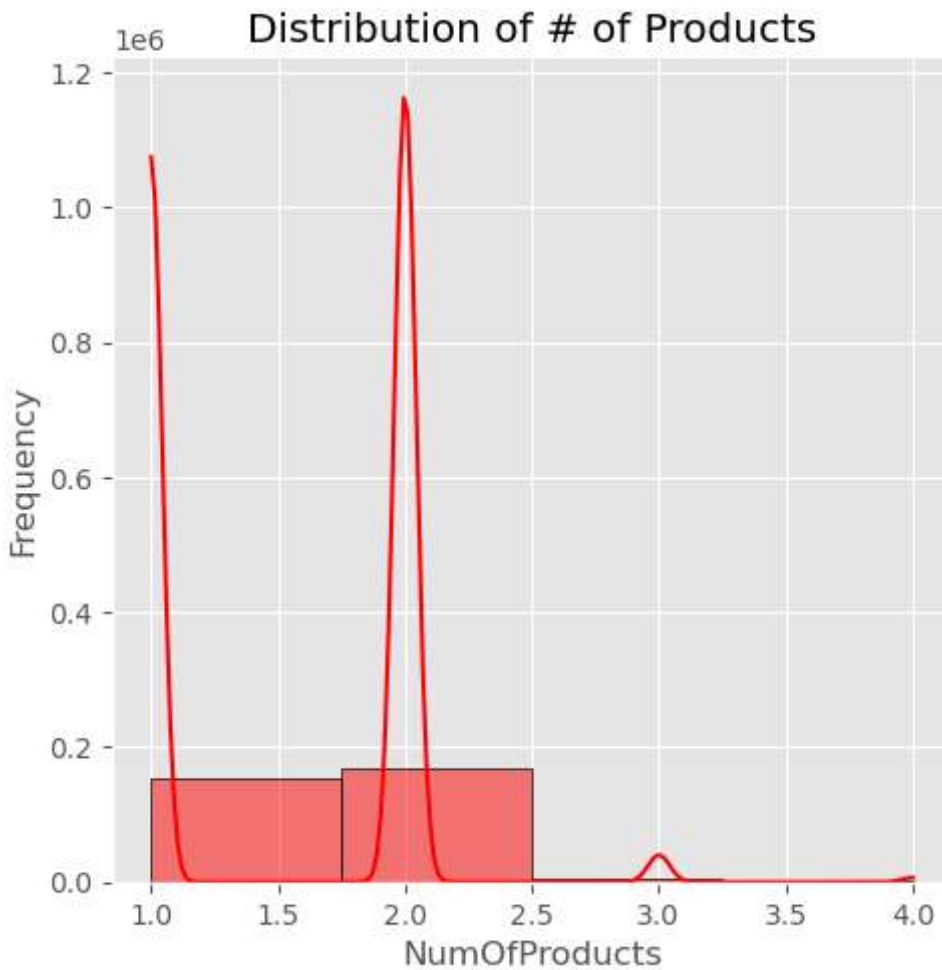
In [53]:
```
sns.jointplot(data = cln, x= 'NumOfProducts', y ='Exited', kind='reg', joint_kw
```

Out[53]: <seaborn.axisgrid.JointGrid at 0x28026c775d0>



In [54]:
```
sns.displot(cln['NumOfProducts'], bins=4, color='red', edgecolor='black', kde=T

plt.xlabel('NumOfProducts')
plt.ylabel('Frequency')
plt.title('Distribution of # of Products')

plt.show()
```

```
C:\Users\p_ste\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [55]:
```
cln['NumOfProducts'].describe()
```

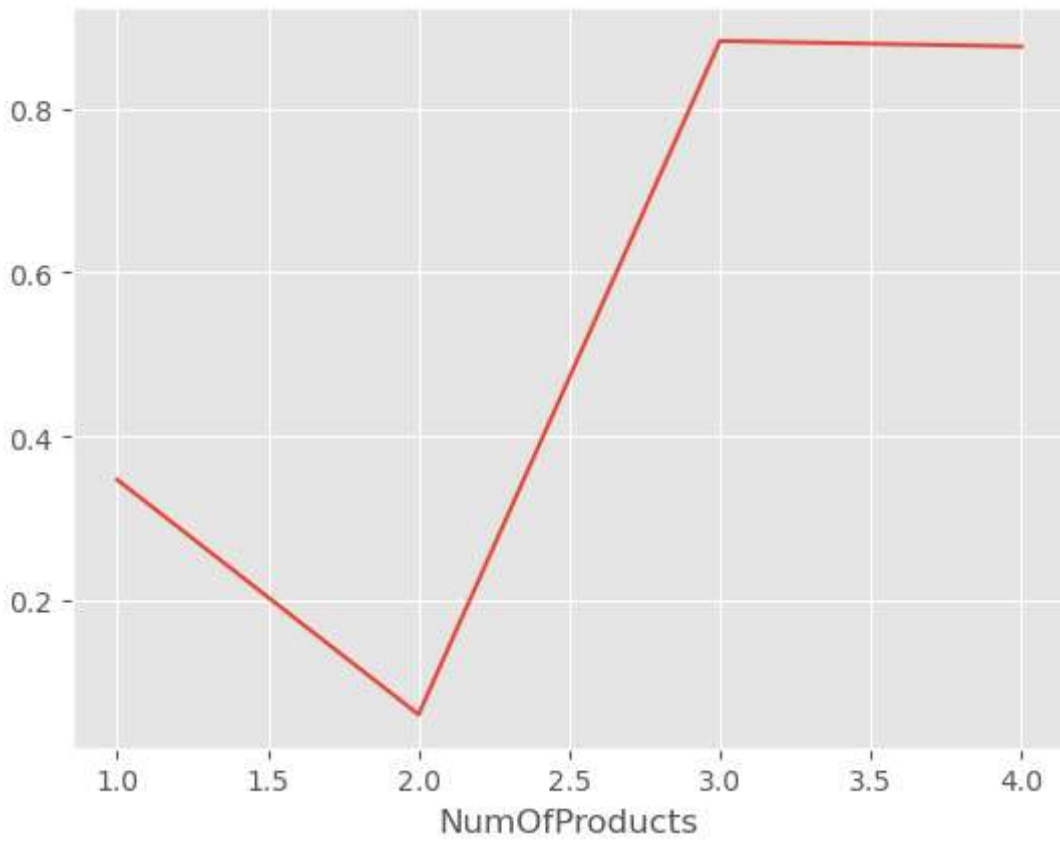
Out[55]:
```
count    330068.000000
mean          1.554455
std           0.547153
min           1.000000
25%           1.000000
50%           2.000000
75%           2.000000
max           4.000000
Name: NumOfProducts, dtype: float64
```

In [56]:
```
cln.NumOfProducts.value_counts()
```

Out[56]:
```
NumOfProducts
2    168582
1    154748
3      5788
4       950
Name: count, dtype: int64
```

In [57]:
```
cln.groupby(['NumOfProducts'])['Exited'].mean().plot()
```

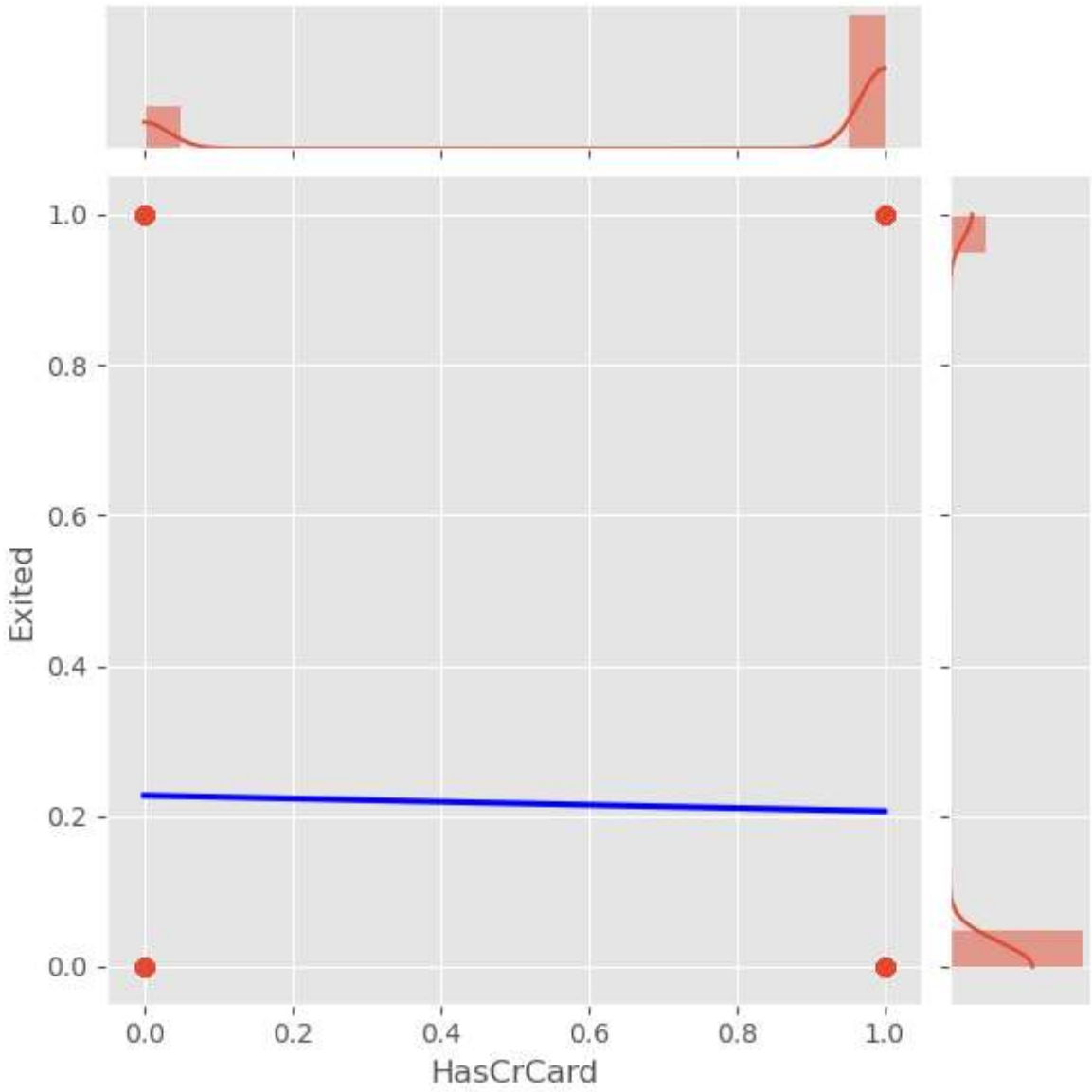
Out[57]:
```
<Axes: xlabel='NumOfProducts'>
```



In [58]:
```
# Use as a categorical variable
```

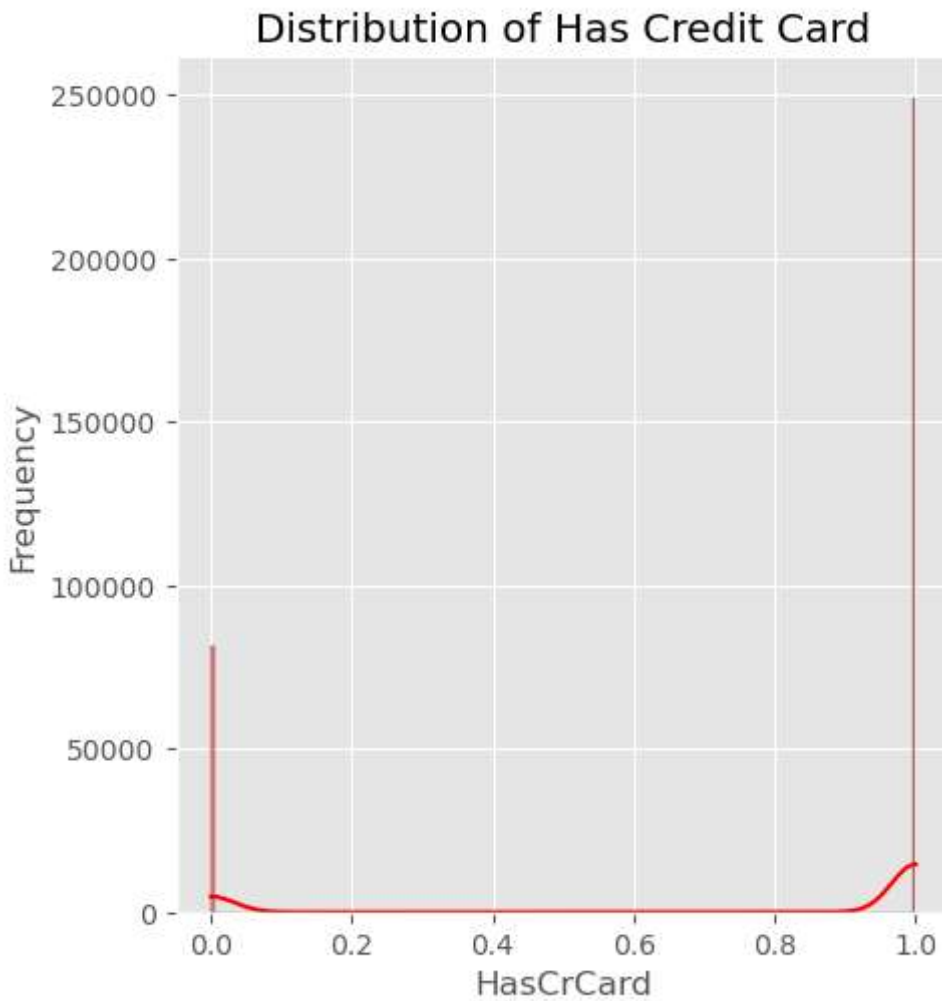
In [59]:
```
sns.jointplot(data = cln, x= 'HasCrCard', y ='Exited', kind='reg', joint_kws={'
```

Out[59]: <seaborn.axisgrid.JointGrid at 0x280288d0990>



In [60]:
```
sns.displot(cln['HasCrCard'], bins=200, color='red', edgecolor='black', kde=Tru

plt.xlabel('HasCrCard')
plt.ylabel('Frequency')
plt.title('Distribution of Has Credit Card')

plt.show()
```

```
C:\Users\p_ste\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [61]:
```
cln['HasCrCard'].describe()
```

Out[61]:
```
count    330068.000000
mean          0.753954
std           0.430706
min           0.000000
25%           1.000000
50%           1.000000
75%           1.000000
max           1.000000
Name: HasCrCard, dtype: float64
```

In [62]:
```
cln.HasCrCard.value_counts()
```

Out[62]:
```
HasCrCard
1.0    248856
0.0     81212
Name: count, dtype: int64
```

In [63]:
```
cln.groupby(cln['HasCrCard'])['HasBalance'].value_counts()
```

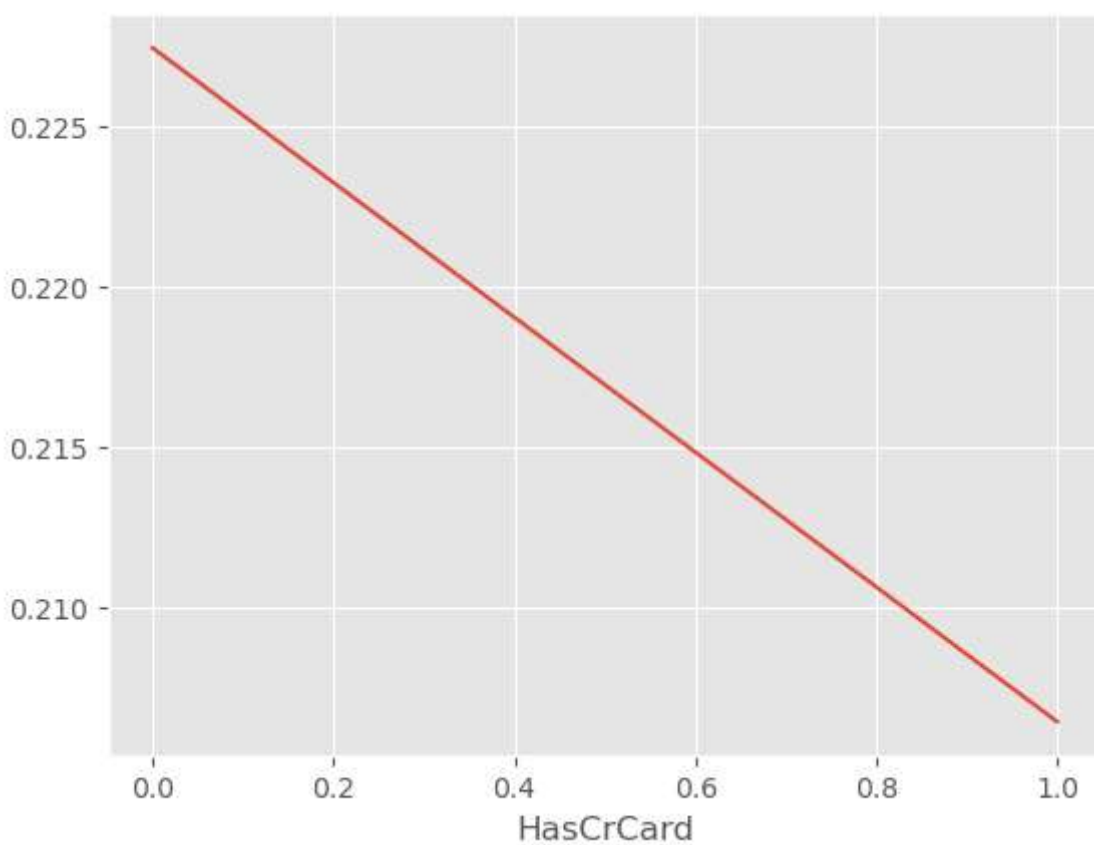
Out[63]:
```
HasCrCard  HasBalance
0.0        0              42606
           1              38606
1.0        0             136690
           1             112166
Name: count, dtype: int64
```

In [64]:
```
cln.groupby(cln['HasCrCard'])['IsActiveMember'].value_counts()
```

Out[64]:
```
HasCrCard  IsActiveMember
0.0        1.0                41920
           0.0                39292
1.0        0.0               126478
           1.0               122378
Name: count, dtype: int64
```

In [65]:
```
cln.groupby(['HasCrCard'])['Exited'].mean().plot()
```

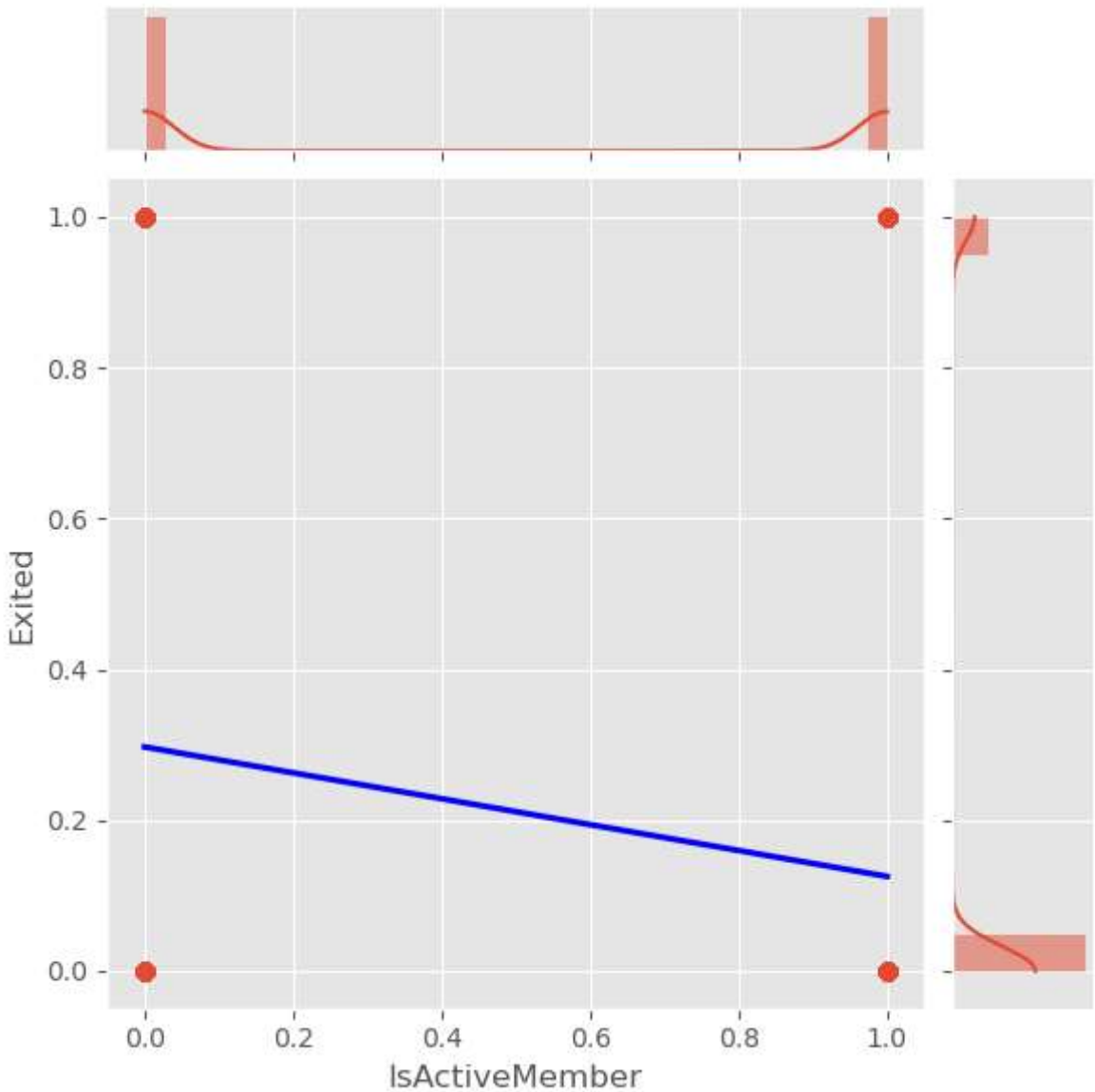
Out[65]:
```
<Axes: xlabel='HasCrCard'>
```



In [66]:
```
# Use as a categoricle variable
```

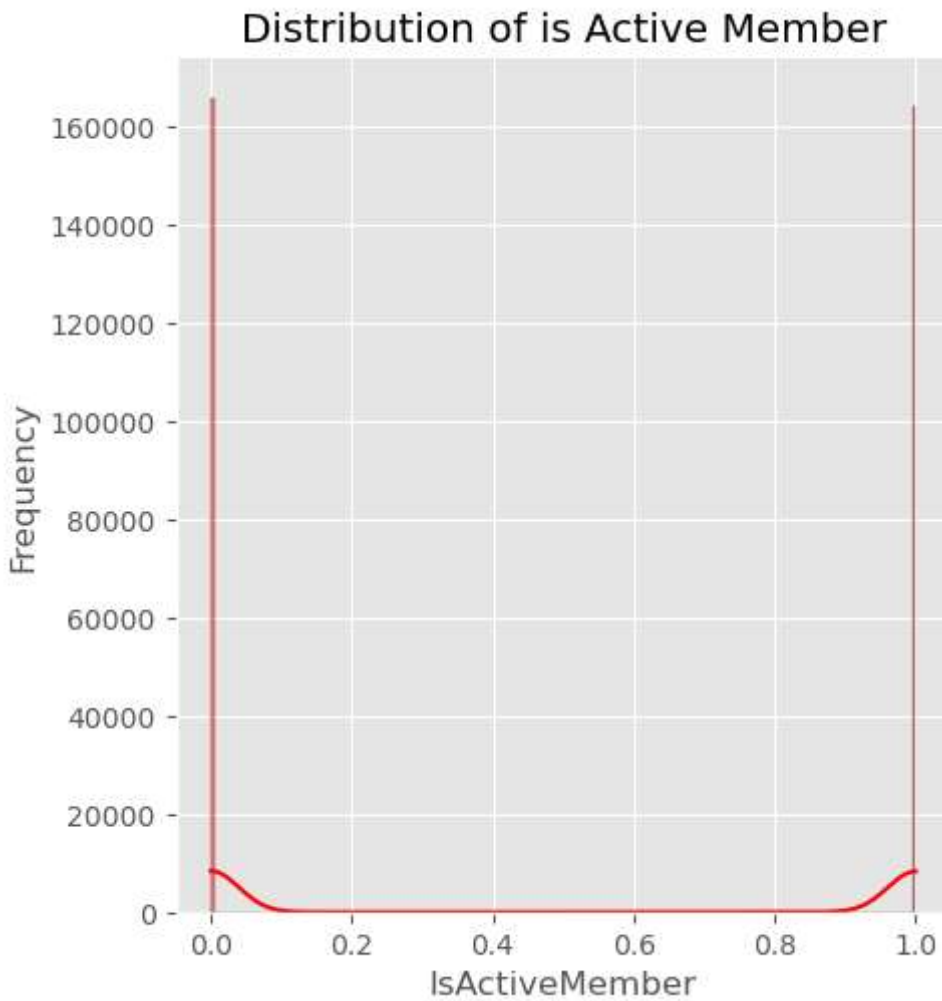
In [67]:
```
sns.jointplot(data = cln, x= 'IsActiveMember', y ='Exited', kind='reg', joint_k
```

Out[67]: <seaborn.axisgrid.JointGrid at 0x2802a5ddb90>



In [68]:
```
sns.displot(cln['IsActiveMember'], bins=200, color='red', edgecolor='black', kd

plt.xlabel('IsActiveMember')
plt.ylabel('Frequency')
plt.title('Distribution of is Active Member')

plt.show()
```

```
C:\Users\p_ste\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [69]:
```
cln['IsActiveMember'].describe()
```

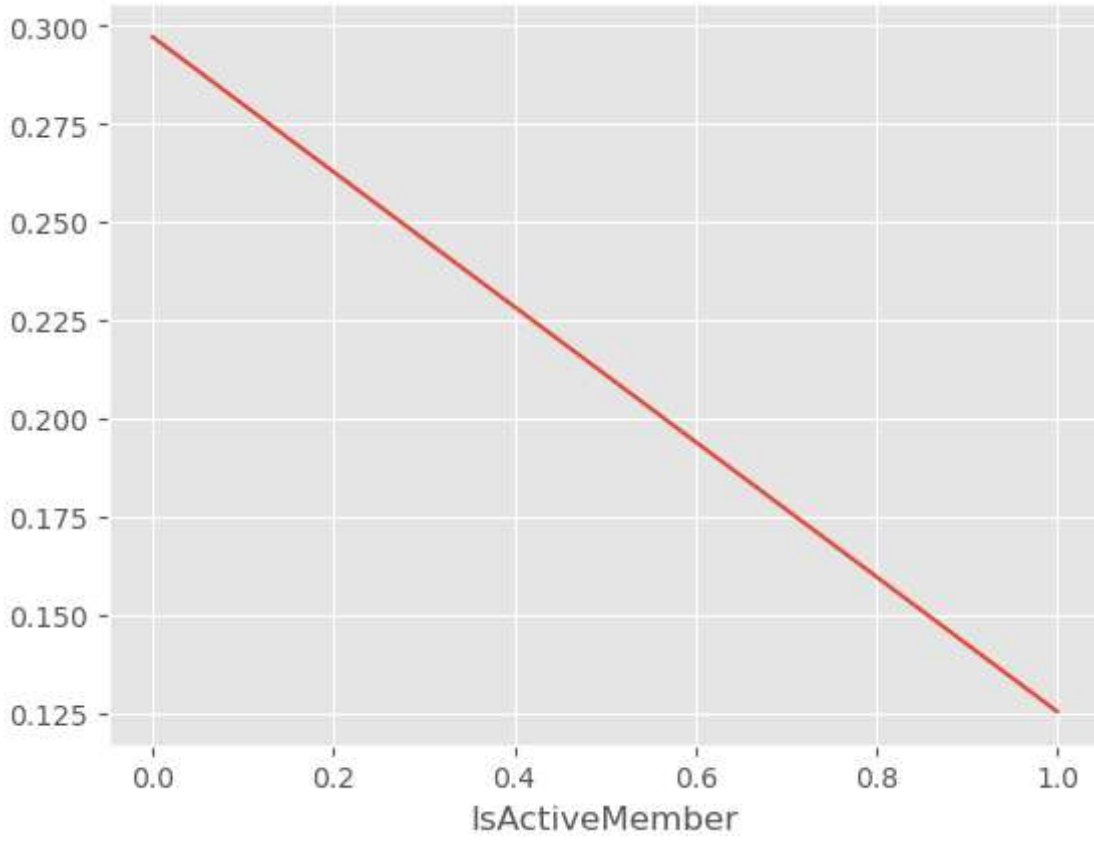
Out[69]:
```
count    330068.000000
mean          0.497770
std           0.499996
min           0.000000
25%           0.000000
50%           0.000000
75%           1.000000
max           1.000000
Name: IsActiveMember, dtype: float64
```

In [70]:
```
cln.IsActiveMember.value_counts()
```

Out[70]:
```
IsActiveMember
0.0    165770
1.0    164298
Name: count, dtype: int64
```

In [71]:
```
cln.groupby(['IsActiveMember'])['Exited'].mean().plot()
```

Out[71]:
```
<Axes: xlabel='IsActiveMember'>
```



In [72]:
```
# how is this defined and tracked?
```

In [73]:
```
cln.groupby(cln['IsActiveMember'])['HasBalance'].value_counts()
```

Out[73]:
```
IsActiveMember  HasBalance
0.0             0            88792
                1            76978
1.0             0            90504
                1            73794
Name: count, dtype: int64
```

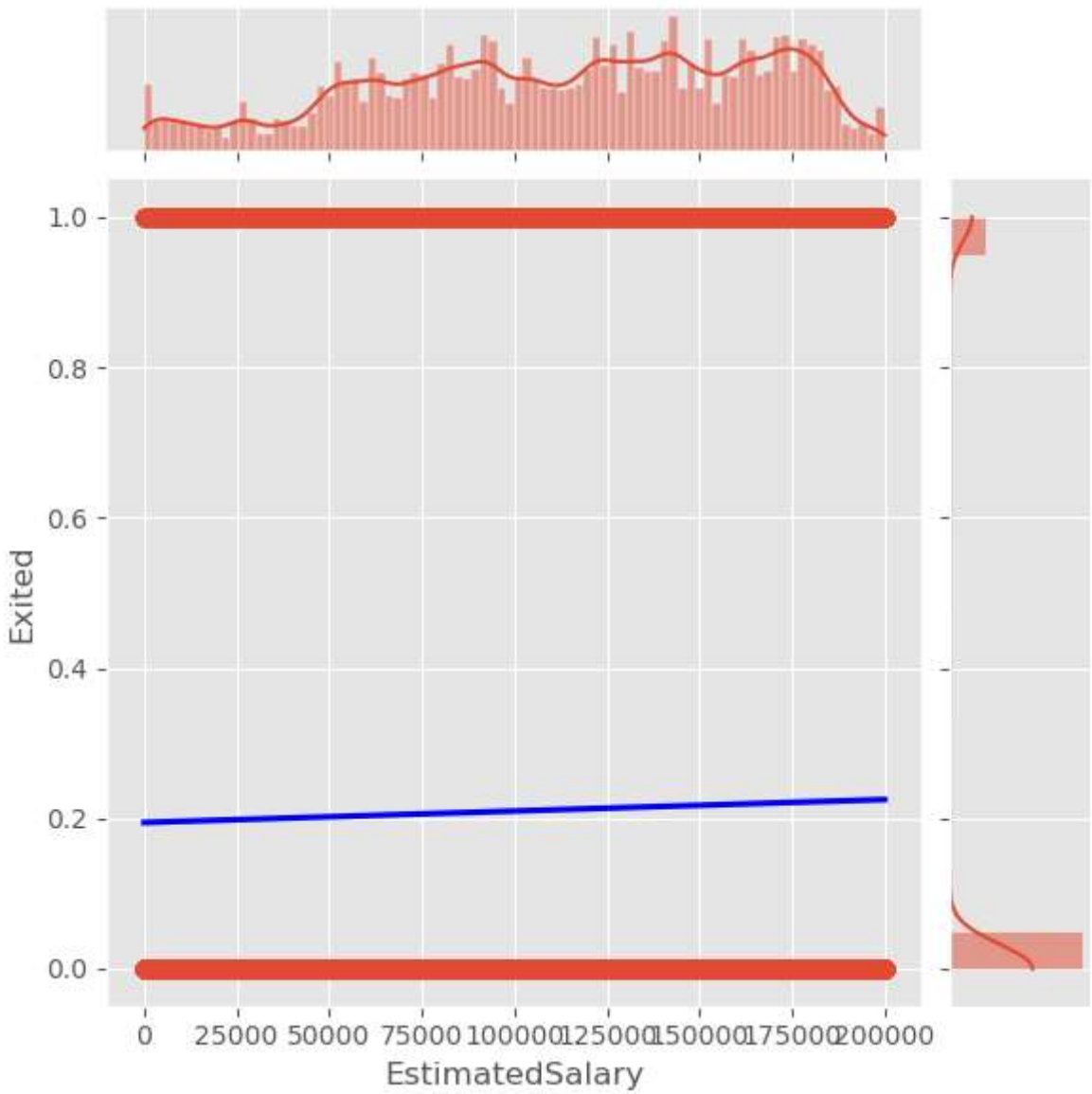
In [74]:
```
cln.groupby(cln['IsActiveMember'])['HasCrCard'].value_counts()
```

Out[74]:
```
IsActiveMember  HasCrCard
0.0             1.0          126478
                0.0           39292
1.0             1.0          122378
                0.0           41920
Name: count, dtype: int64
```

In [75]:
```
# There is a lot of activity that would suggest this is an untrustworthy featur
# how is an inactive member holding a credit card and has a balance?
# this needs more clarity and as it stands is not valuable but will see the fir
```

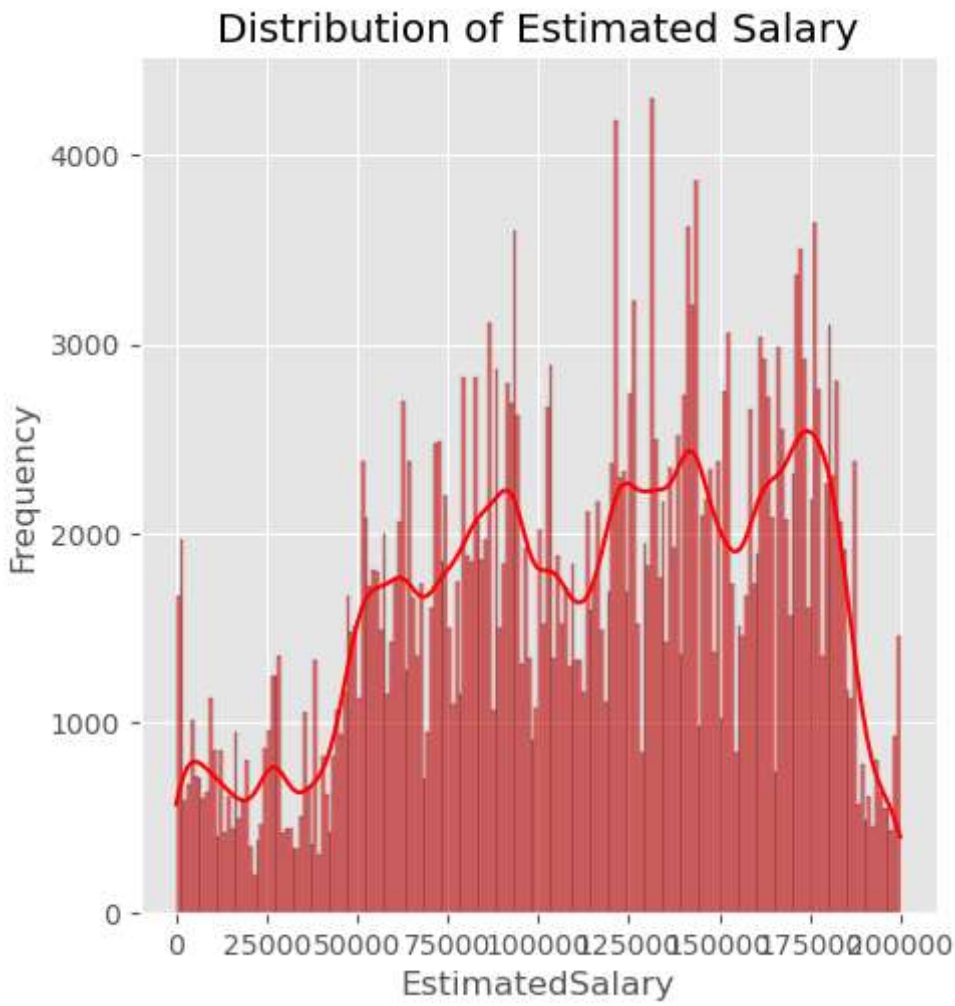
In [76]:
```
sns.jointplot(data = cln, x= 'EstimatedSalary', y ='Exited', kind='reg', joint_
```

Out[76]:
```
<seaborn.axisgrid.JointGrid at 0x28028990c90>
```



In [77]:
```
sns.displot(cln['EstimatedSalary'], bins=200, color='red', edgecolor='black', k

plt.xlabel('EstimatedSalary')
plt.ylabel('Frequency')
plt.title('Distribution of Estimated Salary')

plt.show()
```

```
C:\Users\p_ste\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



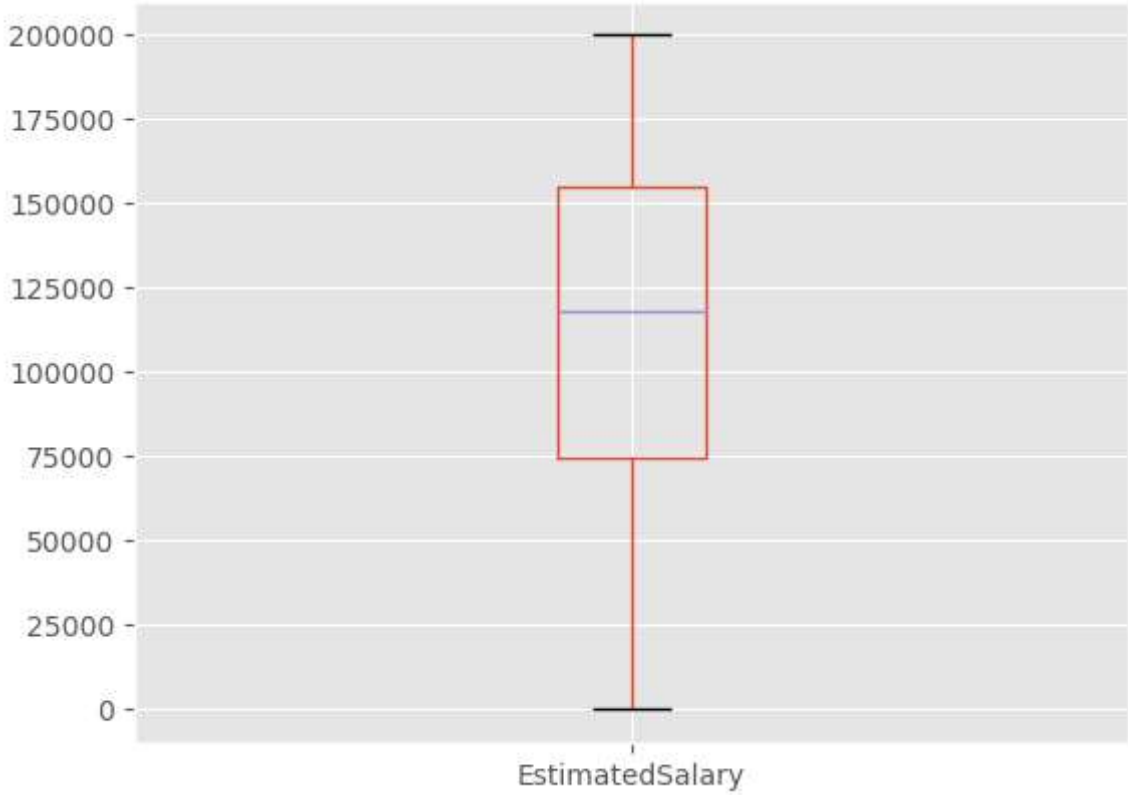
In [78]:
```
# somewhat normal distribution with skewing to the right. Normalisation will he
```

In [79]:
```
cln['EstimatedSalary'].describe()
```

Out[79]:
```
count    330068.000000
mean     112574.822734
std       50292.789399
min          11.580000
25%       74637.570000
50%      117948.000000
75%      155155.250000
max      199992.480000
Name: EstimatedSalary, dtype: float64
```

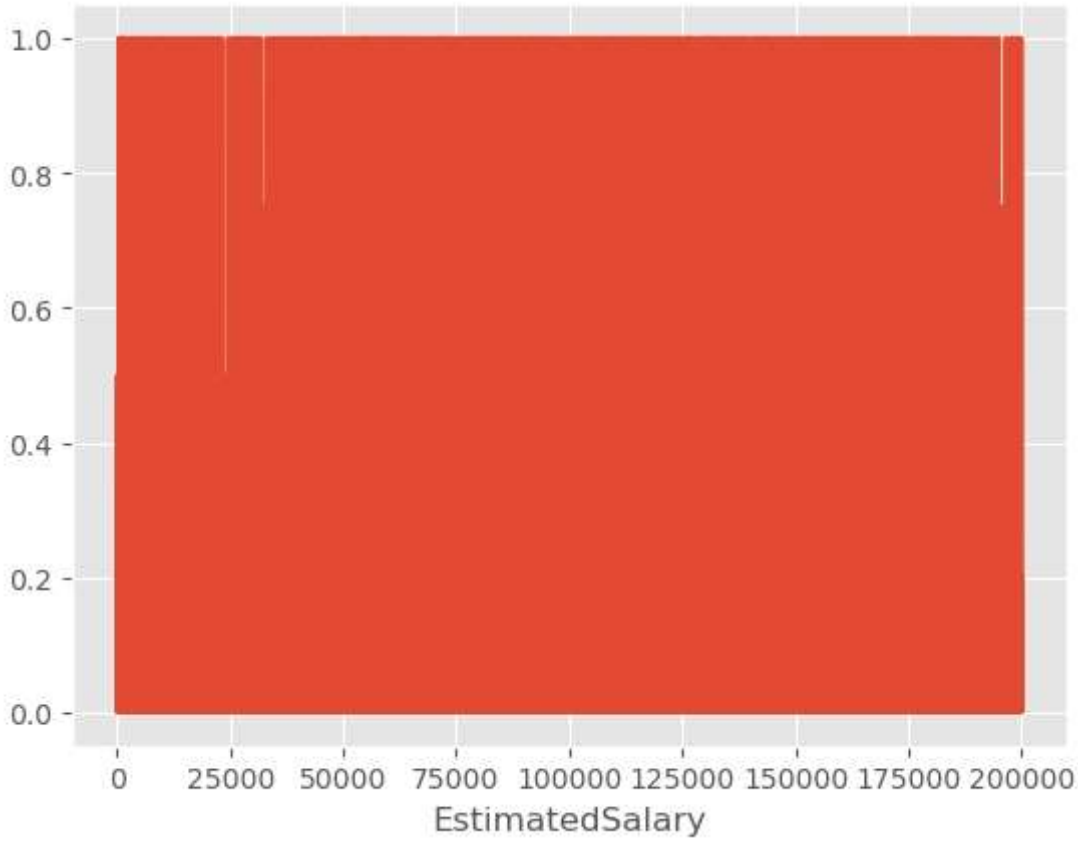
In [80]:
```
cln.boxplot(column=['EstimatedSalary'])
```

Out[80]: <Axes: >



In [81]:
```
cln.groupby(['EstimatedSalary'])['Exited'].mean().plot()
```

Out[81]:
```
<Axes: xlabel='EstimatedSalary'>
```



In [82]:
```
# no apparent outliers
```

In [83]:
```
cln.EstimatedSalary.value_counts()
```

Out[83]:
```
EstimatedSalary
88890.05     356
140941.47    214
167984.72    200
90876.95     196
129964.94    196
            ...
102747.73      2
170593.45      2
109179.48      2
60538.47       2
71173.03       2
Name: count, Length: 55298, dtype: int64
```

In [84]:
```
outcome = 'Exited'
x_cols = cln.drop("Exited", axis=1)
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model1 = ols(formula=formula, data=cln).fit()
model1.summary()
```

Out[84]:

OLS Regression Results

Dep. Variable:	Exited	**R-squared:**	0.231
Model:	OLS	**Adj. R-squared:**	0.231
Method:	Least Squares	**F-statistic:**	7621.
Date:	Wed, 28 Feb 2024	**Prob (F-statistic):**	0.00
Time:	11:49:42	**Log-Likelihood:**	-1.2948e+05
No. Observations:	330068	**AIC:**	2.590e+05
Df Residuals:	330054	**BIC:**	2.591e+05
Df Model:	13		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.0195	0.016	1.210	0.226	-0.012	0.051
Geography[T.Germany]	0.1894	0.002	98.256	0.000	0.186	0.193
Geography[T.Spain]	0.0037	0.002	2.362	0.018	0.001	0.007
Gender[T.Male]	-0.0901	0.001	-71.376	0.000	-0.093	-0.088
CreditScore	6.715e-05	4.46e-05	1.504	0.133	-2.04e-05	0.000
Age	0.0139	7.11e-05	196.019	0.000	0.014	0.014
Tenure	-0.0021	0.000	-9.312	0.000	-0.003	-0.002
Balance	3.498e-08	3.69e-08	0.948	0.343	-3.73e-08	1.07e-07
NumOfProducts	-0.1317	0.001	-105.631	0.000	-0.134	-0.129
HasCrCard	-0.0202	0.001	-13.964	0.000	-0.023	-0.017
IsActiveMember	-0.1591	0.001	-127.282	0.000	-0.162	-0.157
EstimatedSalary	1.251e-07	1.24e-08	10.087	0.000	1.01e-07	1.49e-07
credit_classs	-0.0082	0.002	-3.807	0.000	-0.012	-0.004
HasBalance	-0.0555	0.005	-11.663	0.000	-0.065	-0.046

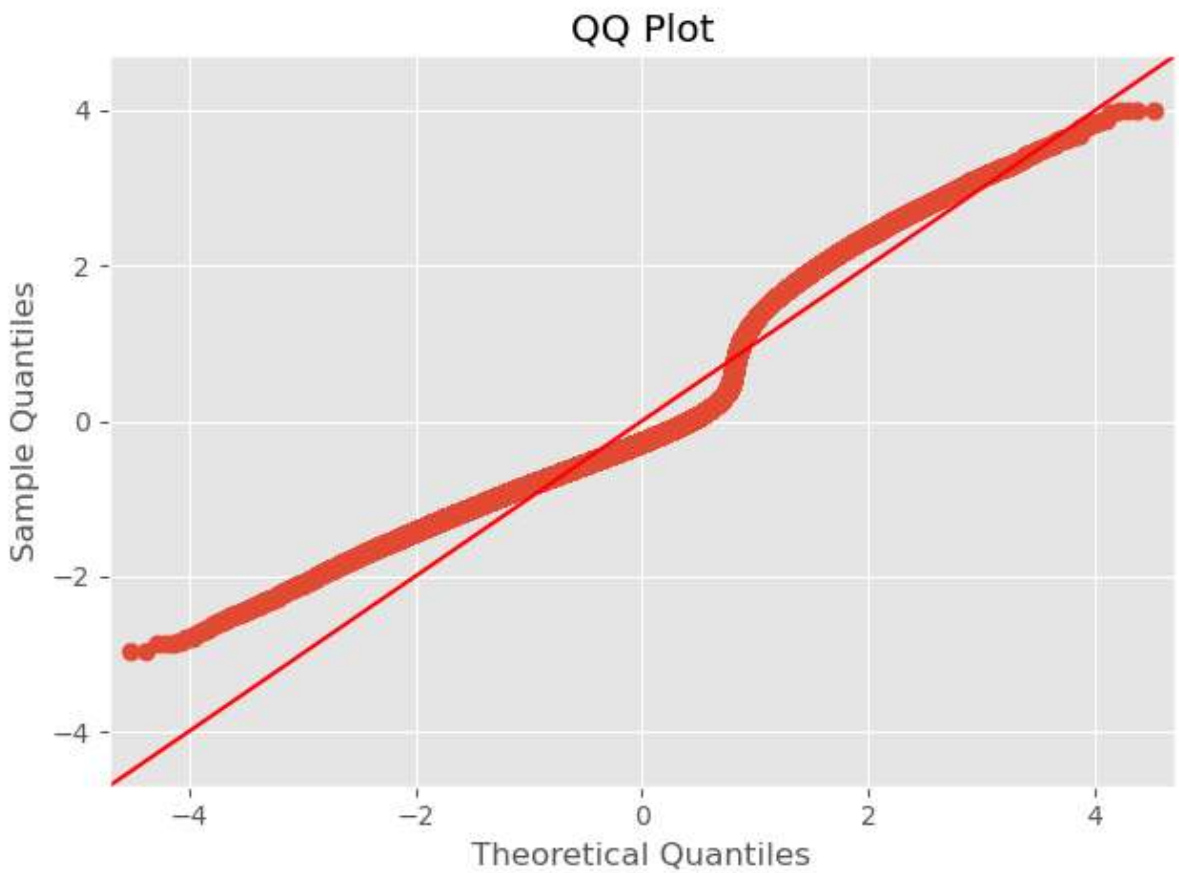
Omnibus:	41215.077	**Durbin-Watson:**	2.005
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	58468.703
Skew:	1.015	**Prob(JB):**	0.00
Kurtosis:	3.357	**Cond. No.**	3.55e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.55e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

In [85]:
```
# Credit Score and Balance seem to have little effect on customer exiting
```

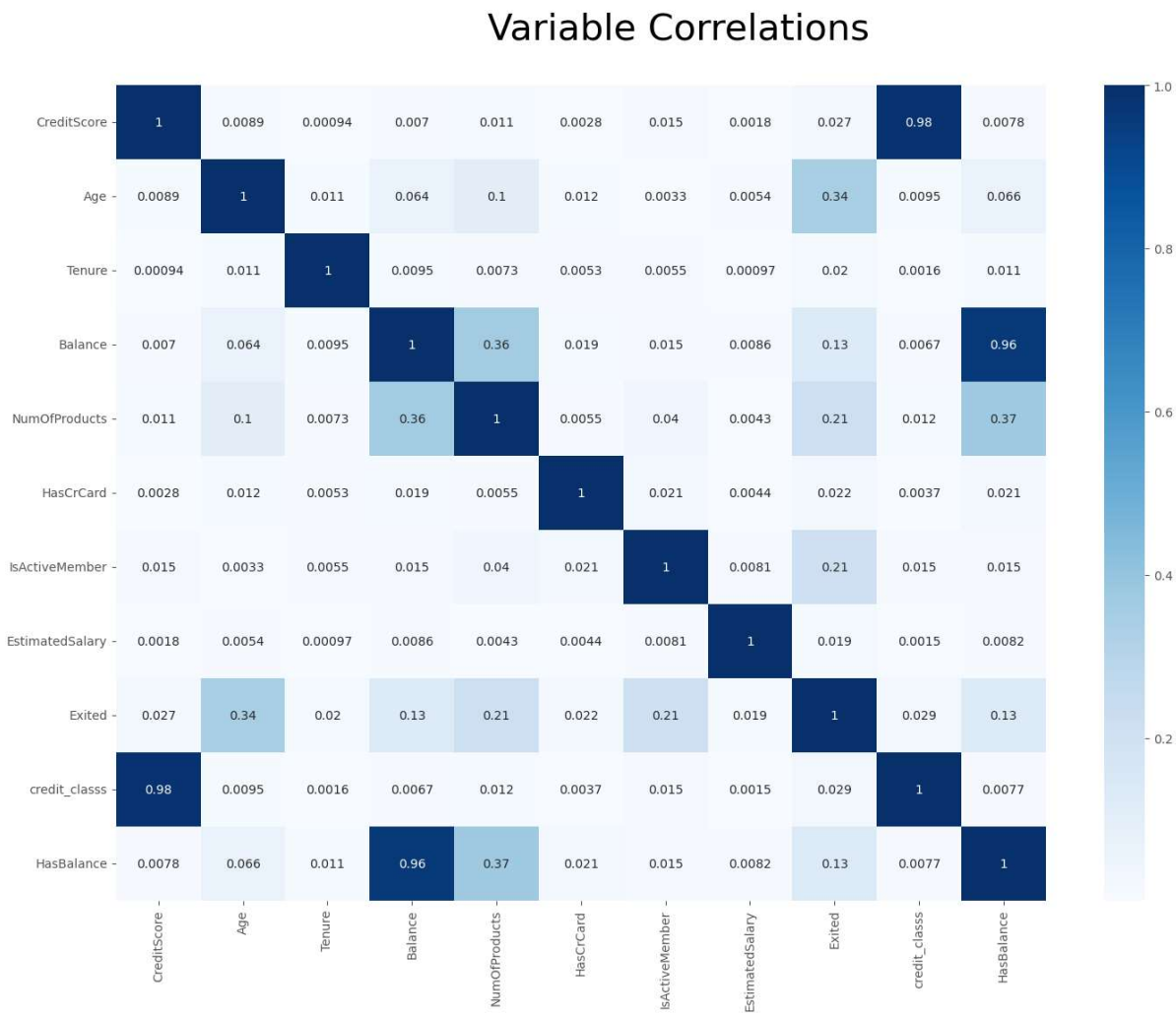
In [86]:
```
plt.figure(figsize=(12, 8))
fig = sm.graphics.qqplot(model1.resid, dist=stats.norm, line='45', fit=True)
plt.title('QQ Plot')
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



In [87]:
```
# Investigate Multicollinearity assumption
```

In [88]:
```
corr = cln.select_dtypes('number').corr().abs()
fig, ax=plt.subplots(figsize=(17,12))
fig.suptitle('Variable Correlations', fontsize=30, y=.95)
heatmap = sns.heatmap(corr, cmap='Blues', annot=True)
```



In [89]:
```
print(corr['Exited'])
```

```
CreditScore        0.027383
Age                0.340768
Tenure             0.019565
Balance            0.129743
NumOfProducts      0.214554
HasCrCard          0.022141
IsActiveMember     0.210237
EstimatedSalary    0.018827
Exited             1.000000
credit_classs      0.028553
HasBalance         0.133346
Name: Exited, dtype: float64
```

In [90]:
```
features = []
correlations = []
for idx, correlation in corr['Exited'].T.items():
    if correlation >= .05 and idx != 'Exited':
        features.append(idx)
        correlations.append(correlation)
corr_with_price = pd.DataFrame({'Correlations':correlations, 'Features': featur

Multicollinear_Features = []
Multicollinear_Corr = []
def check_multicollinearity(feature):
    for idx, correlation in corr[feature].T.items():
        if correlation >= .80 and idx != feature:
            Multicollinear_Features.append([feature, idx])
            Multicollinear_Corr.append(correlation)

for feature in corr:
    check_multicollinearity(feature)
MC_df = pd.DataFrame({'Correlations':Multicollinear_Corr, 'Features': Multicoll
print('Multicollinear Features')
display(MC_df)
print('Correlations with Exited')
display(corr_with_price)
```

```
Multicollinear Features
```

	Correlations	Features
0	0.984683	[CreditScore, credit_classs]
1	0.963088	[Balance, HasBalance]
2	0.984683	[credit_classs, CreditScore]
3	0.963088	[HasBalance, Balance]

```
Correlations with Exited
```

	Correlations	Features
0	0.340768	Age
1	0.129743	Balance
2	0.214554	NumOfProducts
3	0.210237	IsActiveMember
4	0.133346	HasBalance

In [91]:
```
## need to look up interaction of features!
```

In [92]:
```
regression = LinearRegression()
```

In [93]:
```
cln.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 330068 entries, 0 to 165033
Data columns (total 13 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   CreditScore      330068 non-null  int64
 1   Geography        330068 non-null  object
 2   Gender           330068 non-null  object
 3   Age              330068 non-null  float64
 4   Tenure           330068 non-null  int64
 5   Balance          330068 non-null  float64
 6   NumOfProducts    330068 non-null  int64
 7   HasCrCard        330068 non-null  float64
 8   IsActiveMember   330068 non-null  float64
 9   EstimatedSalary  330068 non-null  float64
 10  Exited           330068 non-null  int64
 11  credit_classs    330068 non-null  int32
 12  HasBalance       330068 non-null  int32
dtypes: float64(5), int32(2), int64(4), object(2)
memory usage: 32.7+ MB
```

In [94]:
```
cln[['Tenure', 'NumOfProducts', 'credit_classs', 'HasBalance']] = cln[['Tenure'
```

In [95]:
```
cln.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 330068 entries, 0 to 165033
Data columns (total 13 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   CreditScore      330068 non-null  int64
 1   Geography        330068 non-null  object
 2   Gender           330068 non-null  object
 3   Age              330068 non-null  float64
 4   Tenure           330068 non-null  float64
 5   Balance          330068 non-null  float64
 6   NumOfProducts    330068 non-null  float64
 7   HasCrCard        330068 non-null  float64
 8   IsActiveMember   330068 non-null  float64
 9   EstimatedSalary  330068 non-null  float64
 10  Exited           330068 non-null  int64
 11  credit_classs    330068 non-null  float64
 12  HasBalance       330068 non-null  float64
dtypes: float64(9), int64(2), object(2)
memory usage: 35.3+ MB
```

In [96]:
```
cln.Geography.value_counts()
```

Out[96]:
```
Geography
France     188430
Spain       72426
Germany     69212
Name: count, dtype: int64
```

In [97]:
```
inter = cln

continuous = ['CreditScore', 'Age', 'EstimatedSalary']

categoricals = ['Geography', 'Gender', 'Tenure', 'NumOfProducts', 'credit_class

notmod = ['Balance', 'HasCrCard', 'IsActiveMember', 'Exited', 'HasBalance']

inter_cont = inter[continuous]
```

In [98]:
```
# log features
log_names = [f"{column}_log" for column in inter_cont.columns]

inter_log = np.log(inter_cont)
inter_log.columns = log_names
```

In [99]:
```
# normalize (subract mean and divide by std)


def normalize(feature):
    return (feature - feature.mean()) / feature.std()


inter_log_norm = inter_log.apply(normalize)
```

In [100]:
```
# one hot encode categoricals
inter_ohe = pd.get_dummies(inter, columns = categoricals, drop_first = True, du

inter_notmod = inter[notmod]
```

In [101]:
```
preprocessed = pd.concat([inter_notmod, inter_log_norm, inter_ohe], axis=1)

X = preprocessed.drop("Exited", axis=1)
y = preprocessed["Exited"]
```

In [102]:
```
preprocessed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 330068 entries, 0 to 165033
Data columns (total 43 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   Balance               330068 non-null  float64
 1   HasCrCard             330068 non-null  float64
 2   IsActiveMember        330068 non-null  float64
 3   Exited                330068 non-null  int64
 4   HasBalance            330068 non-null  float64
 5   CreditScore_log       330068 non-null  float64
 6   Age_log               330068 non-null  float64
 7   EstimatedSalary_log   330068 non-null  float64
 8   CreditScore           330068 non-null  int64
 9   Age                   330068 non-null  float64
 10  Balance               330068 non-null  float64
 11  HasCrCard             330068 non-null  float64
 12  IsActiveMember        330068 non-null  float64
 13  EstimatedSalary       330068 non-null  float64
 14  Exited                330068 non-null  int64
 15  HasBalance            330068 non-null  float64
 16  Geography_Germany     330068 non-null  float64
 17  Geography_Spain       330068 non-null  float64
 18  Gender_Male           330068 non-null  float64
 19  Tenure_1.0            330068 non-null  float64
 20  Tenure_2.0            330068 non-null  float64
 21  Tenure_3.0            330068 non-null  float64
 22  Tenure_4.0            330068 non-null  float64
 23  Tenure_5.0            330068 non-null  float64
 24  Tenure_6.0            330068 non-null  float64
 25  Tenure_7.0            330068 non-null  float64
 26  Tenure_8.0            330068 non-null  float64
 27  Tenure_9.0            330068 non-null  float64
 28  Tenure_10.0           330068 non-null  float64
 29  NumOfProducts_2.0     330068 non-null  float64
 30  NumOfProducts_3.0     330068 non-null  float64
 31  NumOfProducts_4.0     330068 non-null  float64
 32  credit_classs_1.0     330068 non-null  float64
 33  credit_classs_2.0     330068 non-null  float64
 34  credit_classs_3.0     330068 non-null  float64
 35  credit_classs_4.0     330068 non-null  float64
 36  credit_classs_5.0     330068 non-null  float64
 37  credit_classs_6.0     330068 non-null  float64
 38  credit_classs_7.0     330068 non-null  float64
 39  credit_classs_8.0     330068 non-null  float64
 40  credit_classs_9.0     330068 non-null  float64
 41  credit_classs_10.0    330068 non-null  float64
 42  credit_classs_11.0    330068 non-null  float64
dtypes: float64(40), int64(3)
memory usage: 110.8 MB
```

In [103]:
```
crossvalidation = KFold(n_splits=10, shuffle=True, random_state=1)
baseline = np.mean(cross_val_score(regression, X, y, scoring="r2", cv=crossvali

baseline
```

Out[103]:
```
0.3217880448420939
```

In [104]:
```
print(list(preprocessed))
```

```
['Balance', 'HasCrCard', 'IsActiveMember', 'Exited', 'HasBalance', 'CreditSco
re_log', 'Age_log', 'EstimatedSalary_log', 'CreditScore', 'Age', 'Balance',
'HasCrCard', 'IsActiveMember', 'EstimatedSalary', 'Exited', 'HasBalance', 'Ge
ography_Germany', 'Geography_Spain', 'Gender_Male', 'Tenure_1.0', 'Tenure_2.
0', 'Tenure_3.0', 'Tenure_4.0', 'Tenure_5.0', 'Tenure_6.0', 'Tenure_7.0', 'Te
nure_8.0', 'Tenure_9.0', 'Tenure_10.0', 'NumOfProducts_2.0', 'NumOfProducts_
3.0', 'NumOfProducts_4.0', 'credit_classs_1.0', 'credit_classs_2.0', 'credit_
classs_3.0', 'credit_classs_4.0', 'credit_classs_5.0', 'credit_classs_6.0',
'credit_classs_7.0', 'credit_classs_8.0', 'credit_classs_9.0', 'credit_classs
_10.0', 'credit_classs_11.0']
```

In [105]:
```
preprocessed = preprocessed.rename(columns = {'Tenure_1.0':'Tenure_1',
                                              'Tenure_2.0':'Tenure_2',
                                              'Tenure_3.0':'Tenure_3',
                                              'Tenure_4.0':'Tenure_4',
                                              'Tenure_5.0':'Tenure_5',
                                              'Tenure_6.0':'Tenure_6',
                                              'Tenure_7.0':'Tenure_7',
                                              'Tenure_8.0':'Tenure_8',
                                              'Tenure_9.0':'Tenure_9',
                                              'Tenure_10.0':'Tenure_10',
                                              'NumOfProducts_2.0':'NumOfProduct
                                              'NumOfProducts_3.0':'NumOfProduct
                                              'NumOfProducts_4.0':'NumOfProduct
                                              'credit_classs_1.0':'credit_class
                                              'credit_classs_2.0':'credit_class
                                              'credit_classs_3.0':'credit_class
                                              'credit_classs_4.0':'credit_class
                                              'credit_classs_5.0':'credit_class
                                              'credit_classs_6.0':'credit_class
                                              'credit_classs_7.0':'credit_class
                                              'credit_classs_8.0':'credit_class
                                              'credit_classs_9.0':'credit_class
                                              'credit_classs_10.0':'credit_clas
                                              'credit_classs_11.0':'credit_clas
```

In [106]:
```
preprocessed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 330068 entries, 0 to 165033
Data columns (total 43 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   Balance              330068 non-null  float64
 1   HasCrCard            330068 non-null  float64
 2   IsActiveMember       330068 non-null  float64
 3   Exited               330068 non-null  int64
 4   HasBalance           330068 non-null  float64
 5   CreditScore_log      330068 non-null  float64
 6   Age_log              330068 non-null  float64
 7   EstimatedSalary_log  330068 non-null  float64
 8   CreditScore          330068 non-null  int64
 9   Age                  330068 non-null  float64
 10  Balance              330068 non-null  float64
 11  HasCrCard            330068 non-null  float64
 12  IsActiveMember       330068 non-null  float64
 13  EstimatedSalary      330068 non-null  float64
 14  Exited               330068 non-null  int64
 15  HasBalance           330068 non-null  float64
 16  Geography_Germany    330068 non-null  float64
 17  Geography_Spain      330068 non-null  float64
 18  Gender_Male          330068 non-null  float64
 19  Tenure_1             330068 non-null  float64
 20  Tenure_2             330068 non-null  float64
 21  Tenure_3             330068 non-null  float64
 22  Tenure_4             330068 non-null  float64
 23  Tenure_5             330068 non-null  float64
 24  Tenure_6             330068 non-null  float64
 25  Tenure_7             330068 non-null  float64
 26  Tenure_8             330068 non-null  float64
 27  Tenure_9             330068 non-null  float64
 28  Tenure_10            330068 non-null  float64
 29  NumOfProducts_2      330068 non-null  float64
 30  NumOfProducts_3      330068 non-null  float64
 31  NumOfProducts_4      330068 non-null  float64
 32  credit_classs_1      330068 non-null  float64
 33  credit_classs_2      330068 non-null  float64
 34  credit_classs_3      330068 non-null  float64
 35  credit_classs_4      330068 non-null  float64
 36  credit_classs_5      330068 non-null  float64
 37  credit_classs_6      330068 non-null  float64
 38  credit_classs_7      330068 non-null  float64
 39  credit_classs_8      330068 non-null  float64
 40  credit_classs_9      330068 non-null  float64
 41  credit_classs_10     330068 non-null  float64
 42  credit_classs_11     330068 non-null  float64
dtypes: float64(40), int64(3)
memory usage: 110.8 MB
```

In [107]:
```
preprocessed['Balance'].describe()
```

Out[107]:

	Balance	Balance
count	330068.000000	330068.000000
mean	55479.086689	55479.086689
std	62817.568119	62817.568119
min	1.000000	1.000000
25%	1.000000	1.000000
50%	1.000000	1.000000
75%	119940.590000	119940.590000
max	250899.090000	250899.090000

In [108]:
```
preprocessed = preprocessed.loc[:,~preprocessed.columns.duplicated()].copy()
```

In [109]:
```
preprocessed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 330068 entries, 0 to 165033
Data columns (total 38 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   Balance              330068 non-null  float64
 1   HasCrCard            330068 non-null  float64
 2   IsActiveMember       330068 non-null  float64
 3   Exited               330068 non-null  int64
 4   HasBalance           330068 non-null  float64
 5   CreditScore_log      330068 non-null  float64
 6   Age_log              330068 non-null  float64
 7   EstimatedSalary_log  330068 non-null  float64
 8   CreditScore          330068 non-null  int64
 9   Age                  330068 non-null  float64
 10  EstimatedSalary      330068 non-null  float64
 11  Geography_Germany    330068 non-null  float64
 12  Geography_Spain      330068 non-null  float64
 13  Gender_Male          330068 non-null  float64
 14  Tenure_1             330068 non-null  float64
 15  Tenure_2             330068 non-null  float64
 16  Tenure_3             330068 non-null  float64
 17  Tenure_4             330068 non-null  float64
 18  Tenure_5             330068 non-null  float64
 19  Tenure_6             330068 non-null  float64
 20  Tenure_7             330068 non-null  float64
 21  Tenure_8             330068 non-null  float64
 22  Tenure_9             330068 non-null  float64
 23  Tenure_10            330068 non-null  float64
 24  NumOfProducts_2      330068 non-null  float64
 25  NumOfProducts_3      330068 non-null  float64
 26  NumOfProducts_4      330068 non-null  float64
 27  credit_classs_1      330068 non-null  float64
 28  credit_classs_2      330068 non-null  float64
 29  credit_classs_3      330068 non-null  float64
 30  credit_classs_4      330068 non-null  float64
 31  credit_classs_5      330068 non-null  float64
 32  credit_classs_6      330068 non-null  float64
 33  credit_classs_7      330068 non-null  float64
 34  credit_classs_8      330068 non-null  float64
 35  credit_classs_9      330068 non-null  float64
 36  credit_classs_10     330068 non-null  float64
 37  credit_classs_11     330068 non-null  float64
dtypes: float64(36), int64(2)
memory usage: 98.2 MB
```

In [110]:
```
preprocessed2 = preprocessed.drop(columns=['CreditScore','Age','EstimatedSalary
```

In [111]:
```
outcome = 'Exited'
x_cols = preprocessed2.drop("Exited", axis=1)
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model2 = ols(formula=formula, data=preprocessed2).fit()
model2.summary()
```

Out[111]:

OLS Regression Results

Dep. Variable:	Exited	**R-squared:**	0.319
Model:	OLS	**Adj. R-squared:**	0.319
Method:	Least Squares	**F-statistic:**	4555.
Date:	Wed, 28 Feb 2024	**Prob (F-statistic):**	0.00
Time:	11:49:49	**Log-Likelihood:**	-1.0930e+05
No. Observations:	330068	**AIC:**	2.187e+05
Df Residuals:	330033	**BIC:**	2.190e+05
Df Model:	34		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.5385	0.057	9.394	0.000	0.426	0.651
Balance	6.083e-09	3.47e-08	0.175	0.861	-6.19e-08	7.41e-08
HasCrCard	-0.0184	0.001	-13.525	0.000	-0.021	-0.016
IsActiveMember	-0.1372	0.001	-116.328	0.000	-0.140	-0.135
HasBalance	-0.1200	0.004	-26.742	0.000	-0.129	-0.111
CreditScore_log	0.0070	0.003	2.070	0.038	0.000	0.014
Age_log	0.1029	0.001	171.705	0.000	0.102	0.104
EstimatedSalary_log	0.0038	0.001	6.468	0.000	0.003	0.005
Geography_Germany	0.2041	0.002	112.422	0.000	0.201	0.208
Geography_Spain	0.0026	0.001	1.794	0.073	-0.000	0.006
Gender_Male	-0.0806	0.001	-67.843	0.000	-0.083	-0.078
Tenure_1	-0.0223	0.004	-5.818	0.000	-0.030	-0.015
Tenure_2	-0.0319	0.004	-8.367	0.000	-0.039	-0.024
Tenure_3	-0.0188	0.004	-4.895	0.000	-0.026	-0.011
Tenure_4	-0.0162	0.004	-4.239	0.000	-0.024	-0.009
Tenure_5	-0.0222	0.004	-5.798	0.000	-0.030	-0.015
Tenure_6	-0.0336	0.004	-8.691	0.000	-0.041	-0.026
Tenure_7	-0.0388	0.004	-10.182	0.000	-0.046	-0.031
Tenure_8	-0.0354	0.004	-9.269	0.000	-0.043	-0.028
Tenure_9	-0.0309	0.004	-8.056	0.000	-0.038	-0.023
Tenure_10	-0.0314	0.005	-6.861	0.000	-0.040	-0.022
NumOfProducts_2	-0.2686	0.001	-199.845	0.000	-0.271	-0.266
NumOfProducts_3	0.4276	0.005	94.300	0.000	0.419	0.437
NumOfProducts_4	0.4088	0.011	37.233	0.000	0.387	0.430
credit_classs_1	-0.1087	0.067	-1.615	0.106	-0.241	0.023

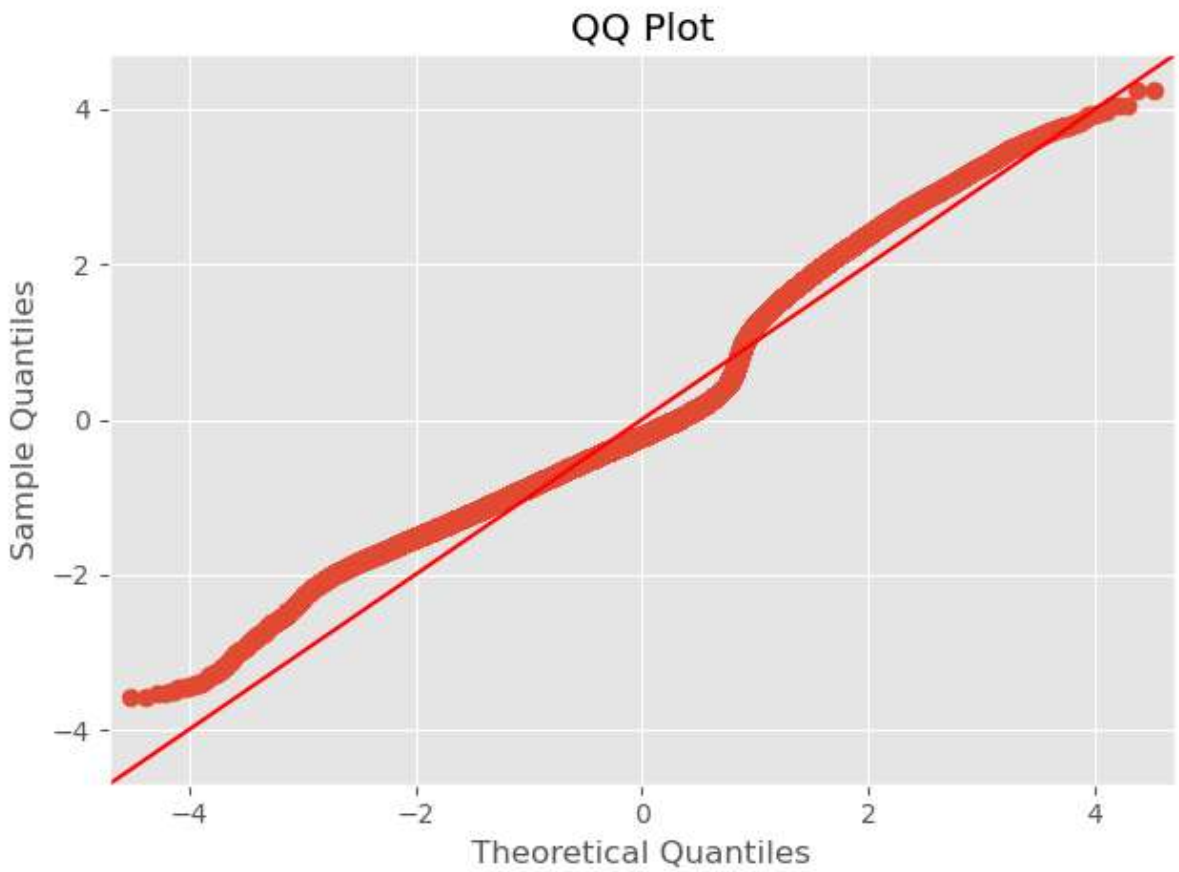
credit_classs_2	0.0111	0.056	0.200	0.841	-0.098	0.120
credit_classs_3	-0.0079	0.055	-0.143	0.886	-0.117	0.101
credit_classs_4	-0.0005	0.056	-0.009	0.993	-0.110	0.109
credit_classs_5	-0.0178	0.056	-0.316	0.752	-0.128	0.093
credit_classs_6	-0.0245	0.057	-0.430	0.667	-0.136	0.087
credit_classs_7	-0.0441	0.058	-0.767	0.443	-0.157	0.069
credit_classs_8	-0.0441	0.058	-0.758	0.448	-0.158	0.070
credit_classs_9	-0.0494	0.059	-0.841	0.400	-0.165	0.066
credit_classs_10	-0.0433	0.060	-0.728	0.467	-0.160	0.073
credit_classs_11	-0.0594	0.060	-0.991	0.322	-0.177	0.058

Omnibus:	33858.903	**Durbin-Watson:**	2.002
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	45194.842
Skew:	0.881	**Prob(JB):**	0.00
Kurtosis:	3.423	**Cond. No.**	2.81e+07

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.81e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

In [112]:
```
plt.figure(figsize=(12, 8))
fig = sm.graphics.qqplot(model2.resid, dist=stats.norm, line='45', fit=True)
plt.title('QQ Plot')
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



In [113]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
linreg = LinearRegression()
```

In [114]:
```
#Split features and target
X = preprocessed2.drop(columns=['Exited'], axis=1)
y = preprocessed2.Exited

#Utilize cross validation to determine baseline score
crossvalidation = KFold(n_splits=10, shuffle=True, random_state=1)
baseline = np.mean(cross_val_score(linreg, X, y, scoring='r2', cv=crossvalidati

from itertools import combinations
interactions = []
feat_combinations = combinations(X.columns, 2)
data = X.copy()
for i, (a, b) in enumerate(feat_combinations):
    data['interaction'] = data[a] * data[b]
    score = np.mean(cross_val_score(linreg, data, y, scoring='r2', cv=crossvali
    if score > baseline:
        interactions.append((a, b, round(score,3)))

    if i % 50 == 0:
        print(i)

print('Top 20 interactions: %s' %sorted(interactions, key=lambda inter: inter[2
```

```
0
50
100
150
200
250
300
350
400
450
500
550
Top 20 interactions: [('Age_log', 'NumOfProducts_2', 0.335), ('IsActiveMembe
r', 'Age_log', 0.334), ('IsActiveMember', 'NumOfProducts_2', 0.327), ('HasBal
ance', 'NumOfProducts_2', 0.326), ('Balance', 'NumOfProducts_2', 0.325), ('Ag
e_log', 'Geography_Germany', 0.323), ('Gender_Male', 'NumOfProducts_2', 0.32
3), ('IsActiveMember', 'Geography_Germany', 0.321), ('Age_log', 'Gender_Mal
e', 0.321), ('Balance', 'IsActiveMember', 0.32), ('Balance', 'Age_log', 0.3
2), ('Balance', 'Gender_Male', 0.32), ('IsActiveMember', 'HasBalance', 0.32),
('IsActiveMember', 'Gender_Male', 0.32), ('HasBalance', 'Age_log', 0.32), ('H
asBalance', 'Gender_Male', 0.32), ('Age_log', 'EstimatedSalary_log', 0.32),
('Age_log', 'Geography_Spain', 0.32), ('Geography_Germany', 'Gender_Male', 0.
32), ('Geography_Germany', 'NumOfProducts_2', 0.32)]
```

In [115]:

```
## add interactions above .321

preprocessed3 = preprocessed2
preprocessed3['age_numprod2']= preprocessed['Age_log'] * preprocessed['NumOfPro
preprocessed3['active_age']= preprocessed['IsActiveMember'] * preprocessed['Age
preprocessed3['active_numprod2']= preprocessed['IsActiveMember'] * preprocessed
preprocessed3['bal_numprod2']= preprocessed['Balance'] * preprocessed['NumOfPro
preprocessed3['hsbal_numprod2']= preprocessed['HasBalance'] * preprocessed['Num
preprocessed3['age_germ']= preprocessed['Age_log'] * preprocessed['Geography_Ge
preprocessed3['gen_numprod2']= preprocessed['Gender_Male'] * preprocessed['NumO
preprocessed3['act_germ']= preprocessed['IsActiveMember'] * preprocessed['Geogr
preprocessed3['age_male']= preprocessed['Age_log'] * preprocessed['Gender_Male'
```

In [116]:
```
outcome = 'Exited'
x_cols = preprocessed3.drop("Exited", axis=1)
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model3 = ols(formula=formula, data=preprocessed3).fit()
model3.summary()
```

Out[116]:

OLS Regression Results

Dep. Variable:	Exited	**R-squared:**	0.365
Model:	OLS	**Adj. R-squared:**	0.364
Method:	Least Squares	**F-statistic:**	4402.
Date:	Wed, 28 Feb 2024	**Prob (F-statistic):**	0.00
Time:	12:18:16	**Log-Likelihood:**	-97981.
No. Observations:	330068	**AIC:**	1.960e+05
Df Residuals:	330024	**BIC:**	1.965e+05
Df Model:	43		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.5887	0.055	10.625	0.000	0.480	0.697
Balance	8.585e-08	3.99e-08	2.153	0.031	7.68e-09	1.64e-07
HasCrCard	-0.0187	0.001	-14.178	0.000	-0.021	-0.016
IsActiveMember	-0.1778	0.002	-98.402	0.000	-0.181	-0.174
HasBalance	-0.1804	0.005	-34.648	0.000	-0.191	-0.170
CreditScore_log	0.0072	0.003	2.184	0.029	0.001	0.014
Age_log	0.1818	0.001	147.629	0.000	0.179	0.184
EstimatedSalary_log	0.0033	0.001	5.737	0.000	0.002	0.004
Geography_Germany	0.1912	0.002	85.462	0.000	0.187	0.196
Geography_Spain	0.0031	0.001	2.159	0.031	0.000	0.006
Gender_Male	-0.1119	0.002	-67.822	0.000	-0.115	-0.109
Tenure_1	-0.0251	0.004	-6.772	0.000	-0.032	-0.018
Tenure_2	-0.0340	0.004	-9.231	0.000	-0.041	-0.027
Tenure_3	-0.0211	0.004	-5.682	0.000	-0.028	-0.014
Tenure_4	-0.0181	0.004	-4.904	0.000	-0.025	-0.011
Tenure_5	-0.0245	0.004	-6.633	0.000	-0.032	-0.017
Tenure_6	-0.0349	0.004	-9.355	0.000	-0.042	-0.028
Tenure_7	-0.0386	0.004	-10.470	0.000	-0.046	-0.031
Tenure_8	-0.0364	0.004	-9.863	0.000	-0.044	-0.029
Tenure_9	-0.0311	0.004	-8.373	0.000	-0.038	-0.024
Tenure_10	-0.0329	0.004	-7.443	0.000	-0.042	-0.024
NumOfProducts_2	-0.4316	0.002	-175.990	0.000	-0.436	-0.427
NumOfProducts_3	0.3833	0.004	87.270	0.000	0.375	0.392
NumOfProducts_4	0.3642	0.011	34.313	0.000	0.343	0.385
credit_classs_1	-0.1006	0.065	-1.547	0.122	-0.228	0.027

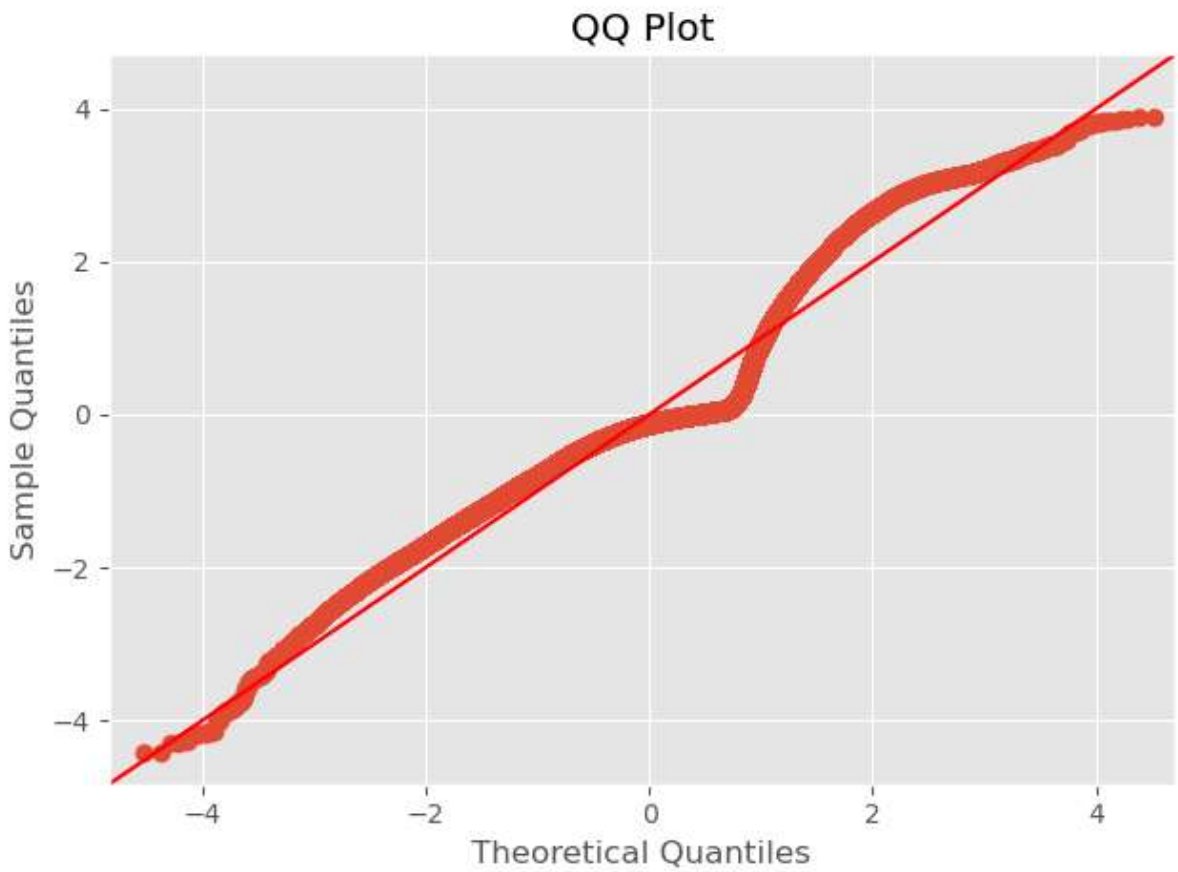
credit_classs_2	0.0347	0.054	0.646	0.518	-0.071	0.140
credit_classs_3	0.0135	0.054	0.251	0.802	-0.092	0.119
credit_classs_4	0.0190	0.054	0.352	0.725	-0.087	0.125
credit_classs_5	0.0020	0.054	0.036	0.971	-0.105	0.109
credit_classs_6	-0.0052	0.055	-0.094	0.925	-0.113	0.103
credit_classs_7	-0.0237	0.056	-0.425	0.671	-0.133	0.085
credit_classs_8	-0.0236	0.056	-0.420	0.675	-0.134	0.087
credit_classs_9	-0.0285	0.057	-0.502	0.616	-0.140	0.083
credit_classs_10	-0.0223	0.058	-0.388	0.698	-0.135	0.090
credit_classs_11	-0.0364	0.058	-0.629	0.529	-0.150	0.077
age_numprod2	-0.0902	0.001	-76.412	0.000	-0.093	-0.088
active_age	-0.0739	0.001	-63.131	0.000	-0.076	-0.072
active_numprod2	0.1144	0.002	49.012	0.000	0.110	0.119
bal_numprod2	-1.949e-07	7.37e-08	-2.646	0.008	-3.39e-07	-5.05e-08
hsbal_numprod2	0.1849	0.009	19.937	0.000	0.167	0.203
age_germ	0.0469	0.001	33.362	0.000	0.044	0.050
gen_numprod2	0.0753	0.002	32.325	0.000	0.071	0.080
act_germ	-0.0559	0.003	-19.710	0.000	-0.061	-0.050
age_male	-0.0178	0.001	-15.330	0.000	-0.020	-0.016

Omnibus:	46774.083	**Durbin-Watson:**	2.005
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	76397.401
Skew:	0.972	**Prob(JB):**	0.00
Kurtosis:	4.332	**Cond. No.**	2.94e+07

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.94e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

In [117]:
```
plt.figure(figsize=(12, 8))
fig = sm.graphics.qqplot(model3.resid, dist=stats.norm, line='45', fit=True)
plt.title('QQ Plot')
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



In [118]:
```
# Model 3 is much improved on model 2 with an r2 increase to .365 from .319
# but some increase to the durban-watson, JB, Condition Number, Skew and Curtos
# this could largely be due to multicollinarity

# the credit score is insignificant, and even when scaled this feature remains
```

In [119]:
```
preprocessed4 = preprocessed3.drop(columns = ['CreditScore_log', 'credit_classs
                                              'credit_classs_5', 'credit_classs_
                                              'credit_classs_10', 'credit_classs
```

In [120]:
```
outcome = 'Exited'
x_cols = preprocessed4.drop("Exited", axis=1)
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model4 = ols(formula=formula, data=preprocessed4).fit()
model4.summary()
```

Out[120]:

OLS Regression Results

Dep. Variable:	Exited	**R-squared:**	0.364
Model:	OLS	**Adj. R-squared:**	0.364
Method:	Least Squares	**F-statistic:**	6094.
Date:	Wed, 28 Feb 2024	**Prob (F-statistic):**	0.00
Time:	12:18:19	**Log-Likelihood:**	-98105.
No. Observations:	330068	**AIC:**	1.963e+05
Df Residuals:	330036	**BIC:**	1.966e+05
Df Model:	31		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.5770	0.004	149.784	0.000	0.569	0.585
Balance	8.568e-08	3.99e-08	2.148	0.032	7.49e-09	1.64e-07
HasCrCard	-0.0189	0.001	-14.342	0.000	-0.021	-0.016
IsActiveMember	-0.1780	0.002	-98.490	0.000	-0.182	-0.174
HasBalance	-0.1807	0.005	-34.691	0.000	-0.191	-0.170
Age_log	0.1819	0.001	147.677	0.000	0.180	0.184
EstimatedSalary_log	0.0032	0.001	5.648	0.000	0.002	0.004
Geography_Germany	0.1913	0.002	85.472	0.000	0.187	0.196
Geography_Spain	0.0031	0.001	2.205	0.027	0.000	0.006
Gender_Male	-0.1120	0.002	-67.865	0.000	-0.115	-0.109
Tenure_1	-0.0251	0.004	-6.754	0.000	-0.032	-0.018
Tenure_2	-0.0340	0.004	-9.231	0.000	-0.041	-0.027
Tenure_3	-0.0210	0.004	-5.663	0.000	-0.028	-0.014
Tenure_4	-0.0182	0.004	-4.927	0.000	-0.025	-0.011
Tenure_5	-0.0243	0.004	-6.583	0.000	-0.032	-0.017
Tenure_6	-0.0350	0.004	-9.366	0.000	-0.042	-0.028
Tenure_7	-0.0386	0.004	-10.478	0.000	-0.046	-0.031
Tenure_8	-0.0364	0.004	-9.863	0.000	-0.044	-0.029
Tenure_9	-0.0312	0.004	-8.393	0.000	-0.038	-0.024
Tenure_10	-0.0327	0.004	-7.388	0.000	-0.041	-0.024
NumOfProducts_2	-0.4322	0.002	-176.183	0.000	-0.437	-0.427
NumOfProducts_3	0.3836	0.004	87.309	0.000	0.375	0.392
NumOfProducts_4	0.3644	0.011	34.324	0.000	0.344	0.385
age_numprod2	-0.0902	0.001	-76.399	0.000	-0.093	-0.088
active_age	-0.0739	0.001	-63.110	0.000	-0.076	-0.072

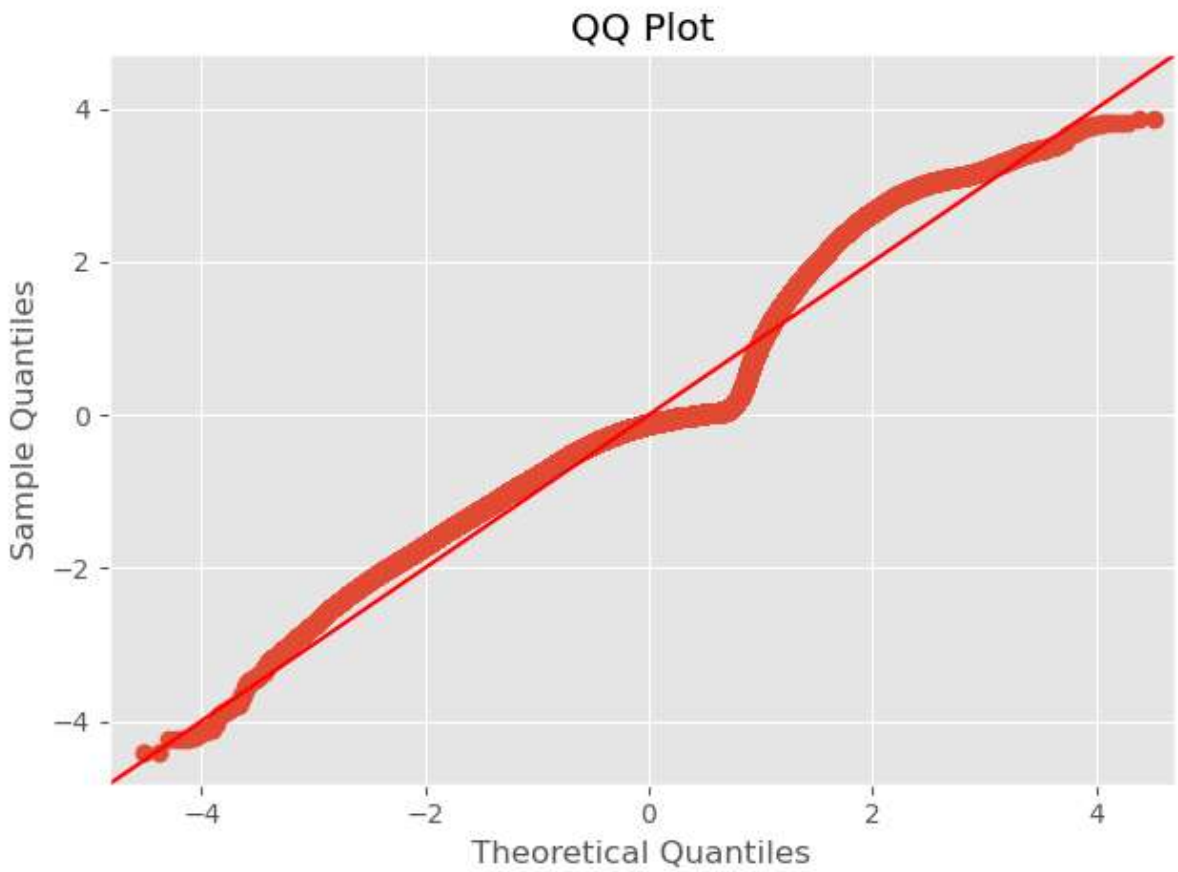
active_numprod2	0.1145	0.002	49.021	0.000	0.110	0.119
bal_numprod2	-1.976e-07	7.37e-08	-2.682	0.007	-3.42e-07	-5.32e-08
hsbal_numprod2	0.1856	0.009	20.003	0.000	0.167	0.204
age_germ	0.0470	0.001	33.371	0.000	0.044	0.050
gen_numprod2	0.0754	0.002	32.355	0.000	0.071	0.080
act_germ	-0.0558	0.003	-19.657	0.000	-0.061	-0.050
age_male	-0.0178	0.001	-15.320	0.000	-0.020	-0.016

Omnibus:	46868.815	**Durbin-Watson:**	2.005
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	76591.412
Skew:	0.974	**Prob(JB):**	0.00
Kurtosis:	4.333	**Cond. No.**	1.70e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.7e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

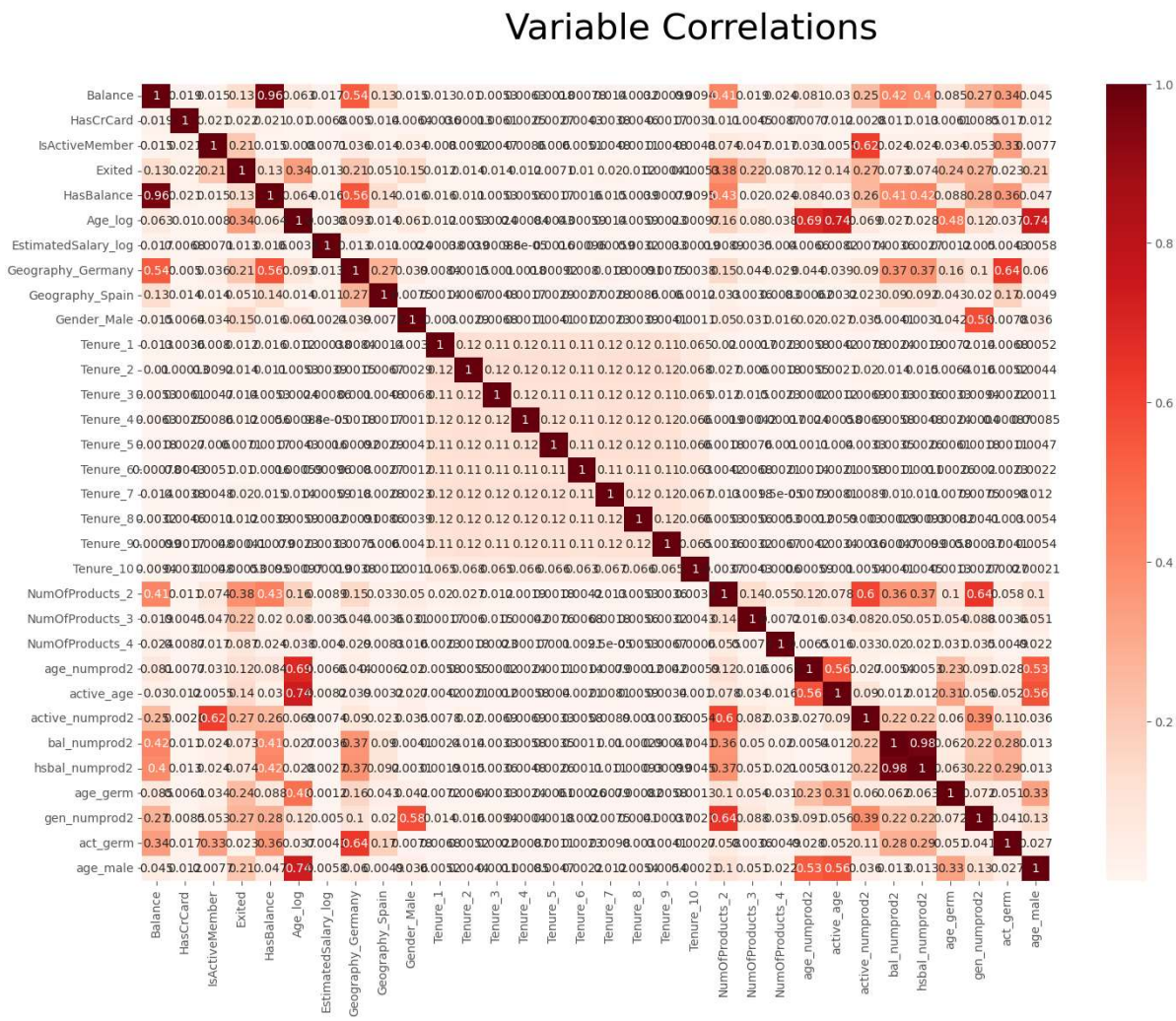
In [121]:
```
plt.figure(figsize=(12, 8))
fig = sm.graphics.qqplot(model4.resid, dist=stats.norm, line='45', fit=True)
plt.title('QQ Plot')
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



In [122]:
```
# Model 4 is very similar to model 3 with the insignifant features removed.
```

In [123]:
```
corr = preprocessed4.select_dtypes('number').corr().abs()
fig, ax=plt.subplots(figsize=(17,12))
fig.suptitle('Variable Correlations', fontsize=30, y=.95)
heatmap = sns.heatmap(corr, cmap='Reds', annot=True)
```



In [124]:
```
print(corr['Exited'])
```

```
Balance                 0.129743
HasCrCard               0.022141
IsActiveMember          0.210237
Exited                  1.000000
HasBalance              0.133346
Age_log                 0.337053
EstimatedSalary_log     0.013460
Geography_Germany       0.211054
Geography_Spain         0.051175
Gender_Male             0.146442
Tenure_1                0.011964
Tenure_2                0.014371
Tenure_3                0.014347
Tenure_4                0.012491
Tenure_5                0.007082
Tenure_6                0.010225
Tenure_7                0.020444
Tenure_8                0.011520
Tenure_9                0.000413
Tenure_10               0.000532
NumOfProducts_2         0.378176
NumOfProducts_3         0.219454
NumOfProducts_4         0.087367
age_numprod2            0.118837
active_age              0.142303
active_numprod2         0.265678
bal_numprod2            0.072632
hsbal_numprod2          0.074215
age_germ                0.242997
gen_numprod2            0.268433
act_germ                0.023212
age_male                0.207640
Name: Exited, dtype: float64
```

In [125]:
```
features = []
correlations = []
for idx, correlation in corr['Exited'].T.items():
    if correlation >= .05 and idx != 'Exited':
        features.append(idx)
        correlations.append(correlation)
corr_with_price = pd.DataFrame({'Correlations':correlations, 'Features': featur

Multicollinear_Features = []
Multicollinear_Corr = []
def check_multicollinearity(feature):
    for idx, correlation in corr[feature].T.items():
        if correlation >= .80 and idx != feature:
            Multicollinear_Features.append([feature, idx])
            Multicollinear_Corr.append(correlation)

for feature in corr:
    check_multicollinearity(feature)
MC_df = pd.DataFrame({'Correlations':Multicollinear_Corr, 'Features': Multicoll
print('Multicollinear Features')
display(MC_df)
print('Correlations with Exited')
display(corr_with_price)
```

```
Multicollinear Features
```

	Correlations	Features
0	0.963088	[Balance, HasBalance]
1	0.963088	[HasBalance, Balance]
2	0.975098	[bal_numprod2, hsbal_numprod2]
3	0.975098	[hsbal_numprod2, bal_numprod2]

```
Correlations with Exited
```

	Correlations	Features
0	0.129743	Balance
1	0.210237	IsActiveMember
2	0.133346	HasBalance
3	0.337053	Age_log
4	0.211054	Geography_Germany
5	0.051175	Geography_Spain
6	0.146442	Gender_Male
7	0.378176	NumOfProducts_2
8	0.219454	NumOfProducts_3
9	0.087367	NumOfProducts_4
10	0.118837	age_numprod2
11	0.142303	active_age
12	0.265678	active_numprod2
13	0.072632	bal_numprod2
14	0.074215	hsbal_numprod2
15	0.242997	age_germ
16	0.268433	gen_numprod2
17	0.207640	age_male

In [126]:
```
# we will drop the Balance and the bal_numprod2 features due to multicollineari
preprocessed5 = preprocessed4.drop(columns=['Balance', 'bal_numprod2', 'Geograp
```

In [127]:
```
outcome = 'Exited'
x_cols = preprocessed5.drop("Exited", axis=1)
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model5 = ols(formula=formula, data=preprocessed5).fit()
model5.summary()
```

Out[127]:

OLS Regression Results

Dep. Variable:	Exited	**R-squared:**	0.364
Model:	OLS	**Adj. R-squared:**	0.364
Method:	Least Squares	**F-statistic:**	6746.
Date:	Wed, 28 Feb 2024	**Prob (F-statistic):**	0.00
Time:	12:18:24	**Log-Likelihood:**	-98111.
No. Observations:	330068	**AIC:**	1.963e+05
Df Residuals:	330039	**BIC:**	1.966e+05
Df Model:	28		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.5780	0.004	150.940	0.000	0.570	0.585
HasCrCard	-0.0189	0.001	-14.379	0.000	-0.022	-0.016
IsActiveMember	-0.1780	0.002	-98.489	0.000	-0.182	-0.174
HasBalance	-0.1702	0.002	-90.304	0.000	-0.174	-0.166
Age_log	0.1819	0.001	147.693	0.000	0.180	0.184
EstimatedSalary_log	0.0032	0.001	5.624	0.000	0.002	0.004
Geography_Germany	0.1903	0.002	86.533	0.000	0.186	0.195
Gender_Male	-0.1120	0.002	-67.862	0.000	-0.115	-0.109
Tenure_1	-0.0251	0.004	-6.776	0.000	-0.032	-0.018
Tenure_2	-0.0341	0.004	-9.251	0.000	-0.041	-0.027
Tenure_3	-0.0211	0.004	-5.686	0.000	-0.028	-0.014
Tenure_4	-0.0182	0.004	-4.943	0.000	-0.025	-0.011
Tenure_5	-0.0244	0.004	-6.604	0.000	-0.032	-0.017
Tenure_6	-0.0350	0.004	-9.379	0.000	-0.042	-0.028
Tenure_7	-0.0387	0.004	-10.495	0.000	-0.046	-0.031
Tenure_8	-0.0365	0.004	-9.873	0.000	-0.044	-0.029
Tenure_9	-0.0313	0.004	-8.419	0.000	-0.039	-0.024
Tenure_10	-0.0327	0.004	-7.395	0.000	-0.041	-0.024
NumOfProducts_2	-0.4322	0.002	-176.173	0.000	-0.437	-0.427
NumOfProducts_3	0.3836	0.004	87.323	0.000	0.375	0.392
NumOfProducts_4	0.3645	0.011	34.329	0.000	0.344	0.385
age_numprod2	-0.0902	0.001	-76.396	0.000	-0.093	-0.088
active_age	-0.0739	0.001	-63.108	0.000	-0.076	-0.072
active_numprod2	0.1145	0.002	49.021	0.000	0.110	0.119
hsbal_numprod2	0.1617	0.003	62.606	0.000	0.157	0.167

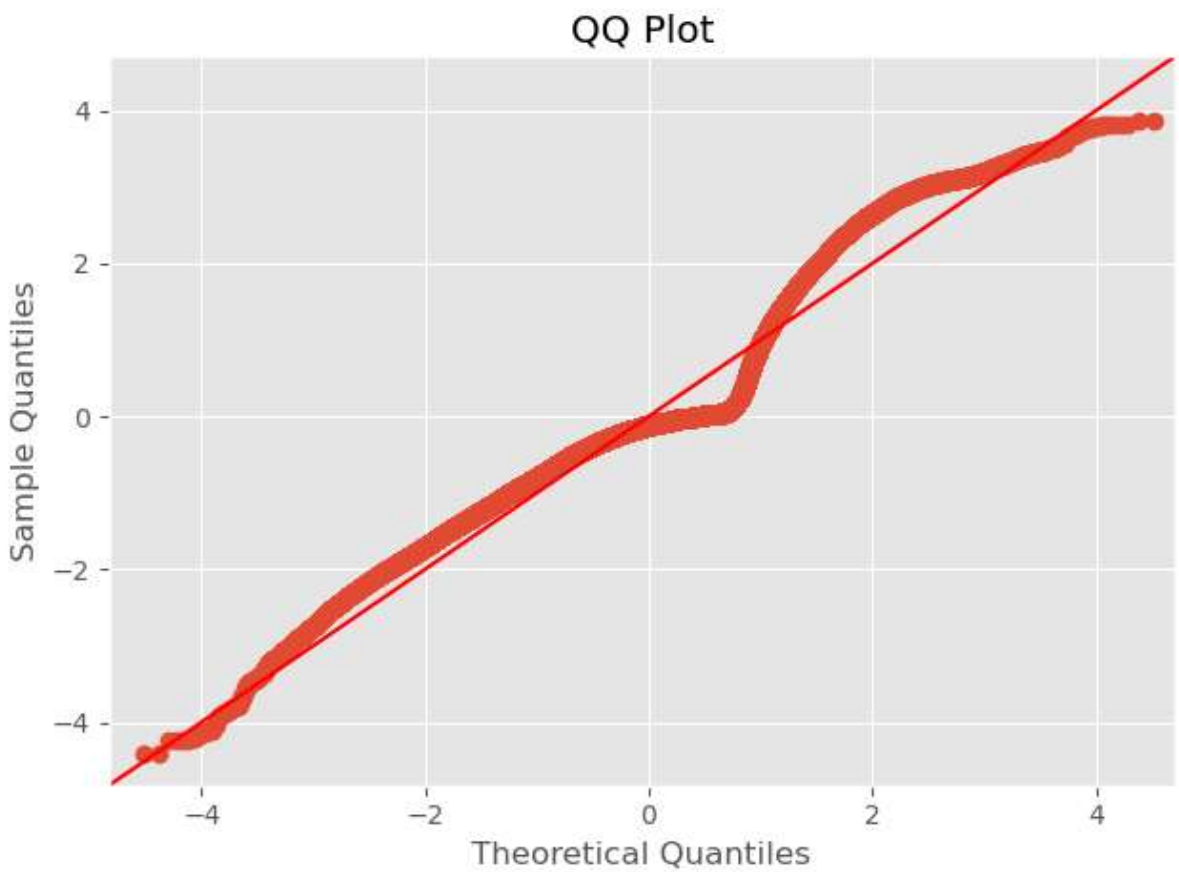
age_germ	0.0470	0.001	33.370	0.000	0.044	0.050
gen_numprod2	0.0754	0.002	32.339	0.000	0.071	0.080
act_germ	-0.0558	0.003	-19.668	0.000	-0.061	-0.050
age_male	-0.0178	0.001	-15.324	0.000	-0.020	-0.016

Omnibus:	46877.189	**Durbin-Watson:**	2.005
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	76609.967
Skew:	0.974	**Prob(JB):**	0.00
Kurtosis:	4.333	**Cond. No.**	34.4

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

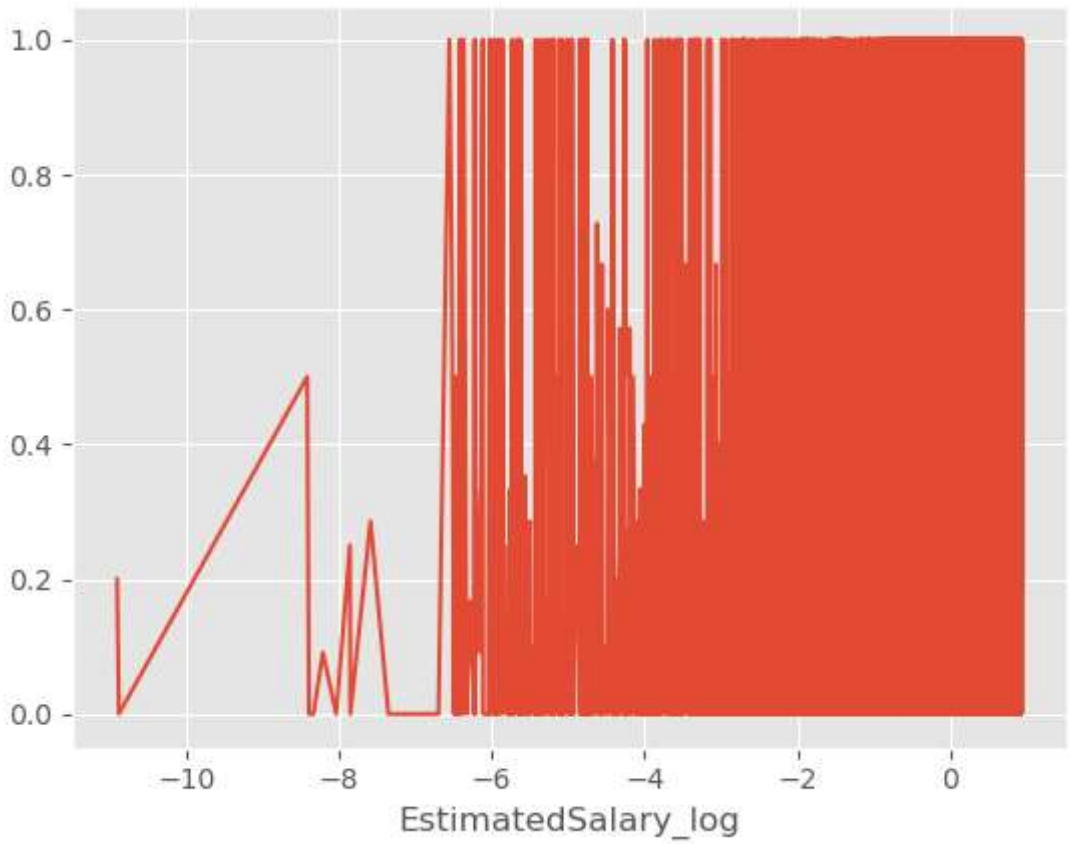
In [128]:
```
plt.figure(figsize=(12, 8))
fig = sm.graphics.qqplot(model4.resid, dist=stats.norm, line='45', fit=True)
plt.title('QQ Plot')
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



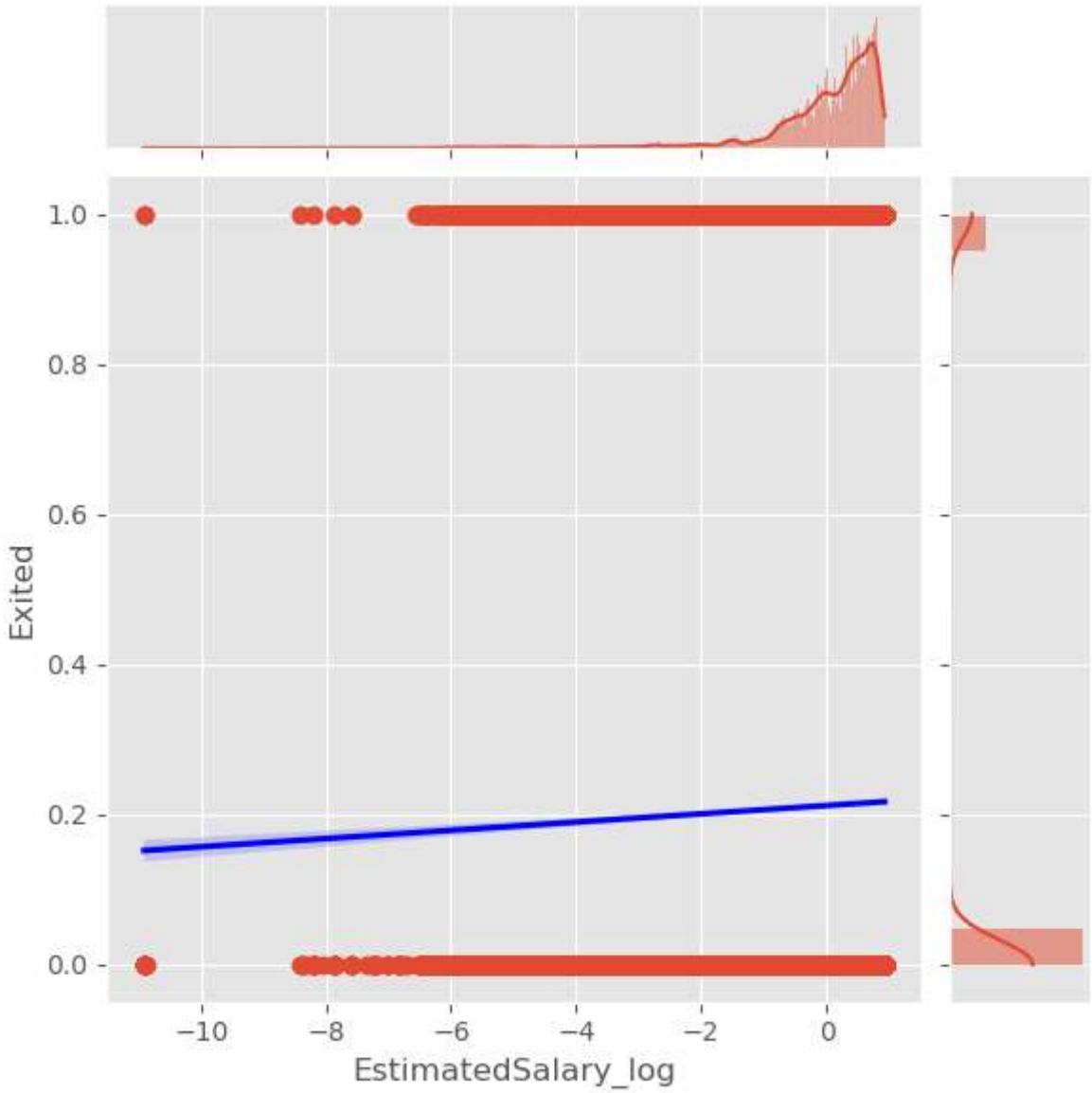
In [129]:
```
preprocessed5.groupby(['EstimatedSalary_log'])['Exited'].mean().plot()
```

Out[129]: <Axes: xlabel='EstimatedSalary_log'>



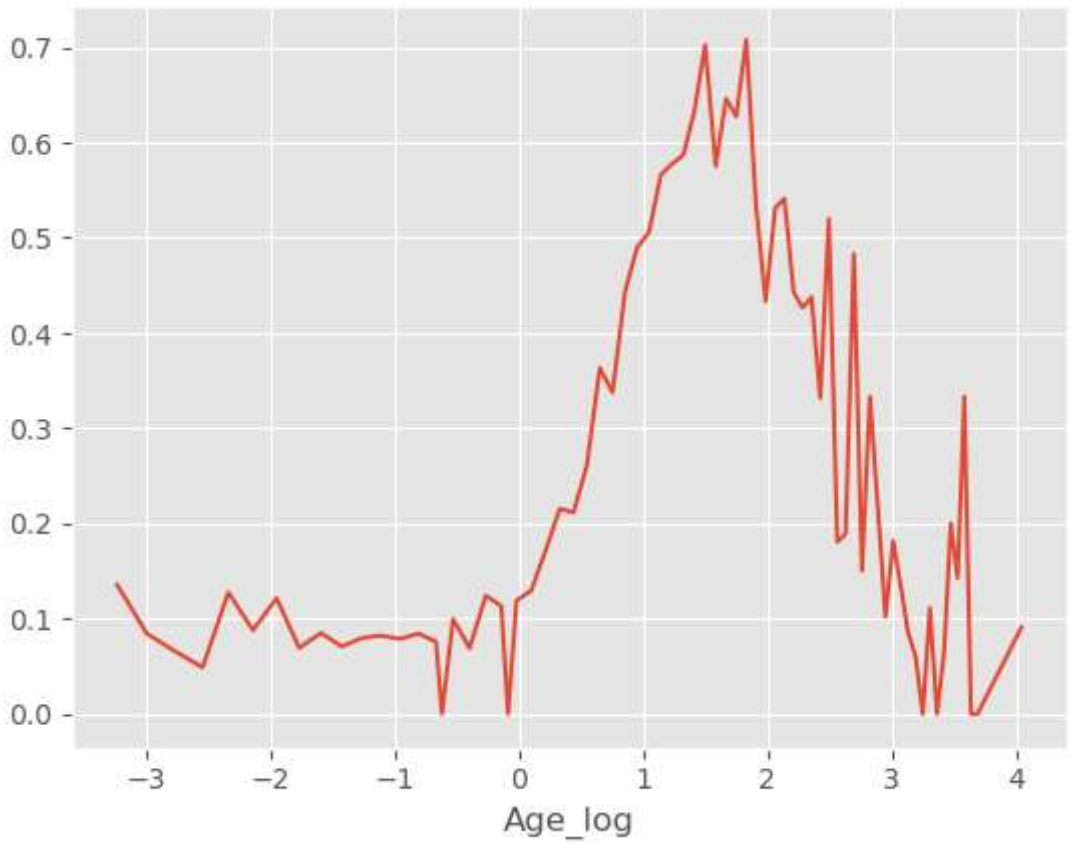
In [130]:
```
sns.jointplot(data = preprocessed5, x= 'EstimatedSalary_log', y ='Exited', kinc
```

Out[130]: <seaborn.axisgrid.JointGrid at 0x2802e44b890>



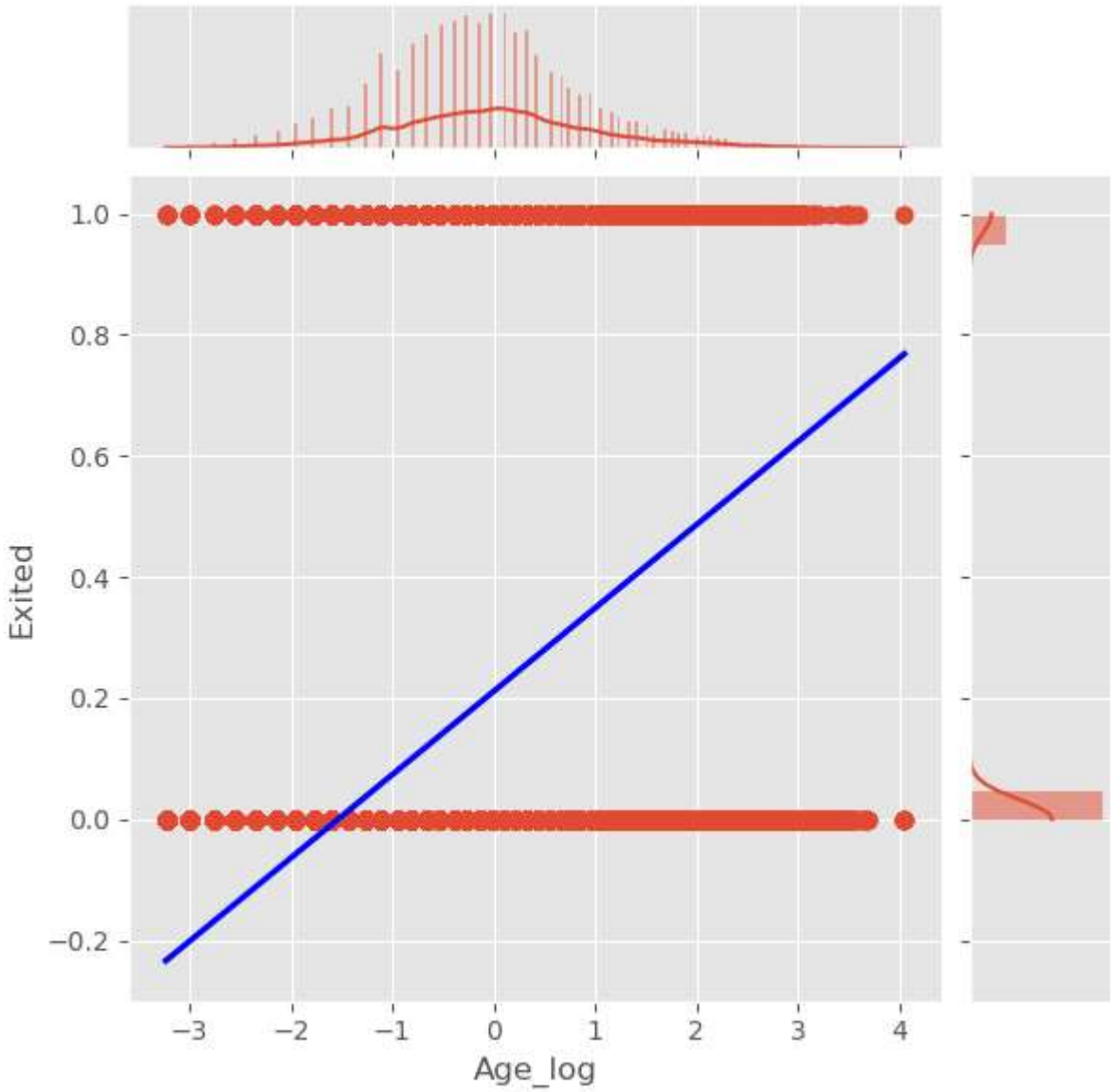
In [131]:
```
preprocessed5.groupby(['Age_log'])['Exited'].mean().plot()
```

Out[131]: <Axes: xlabel='Age_log'>

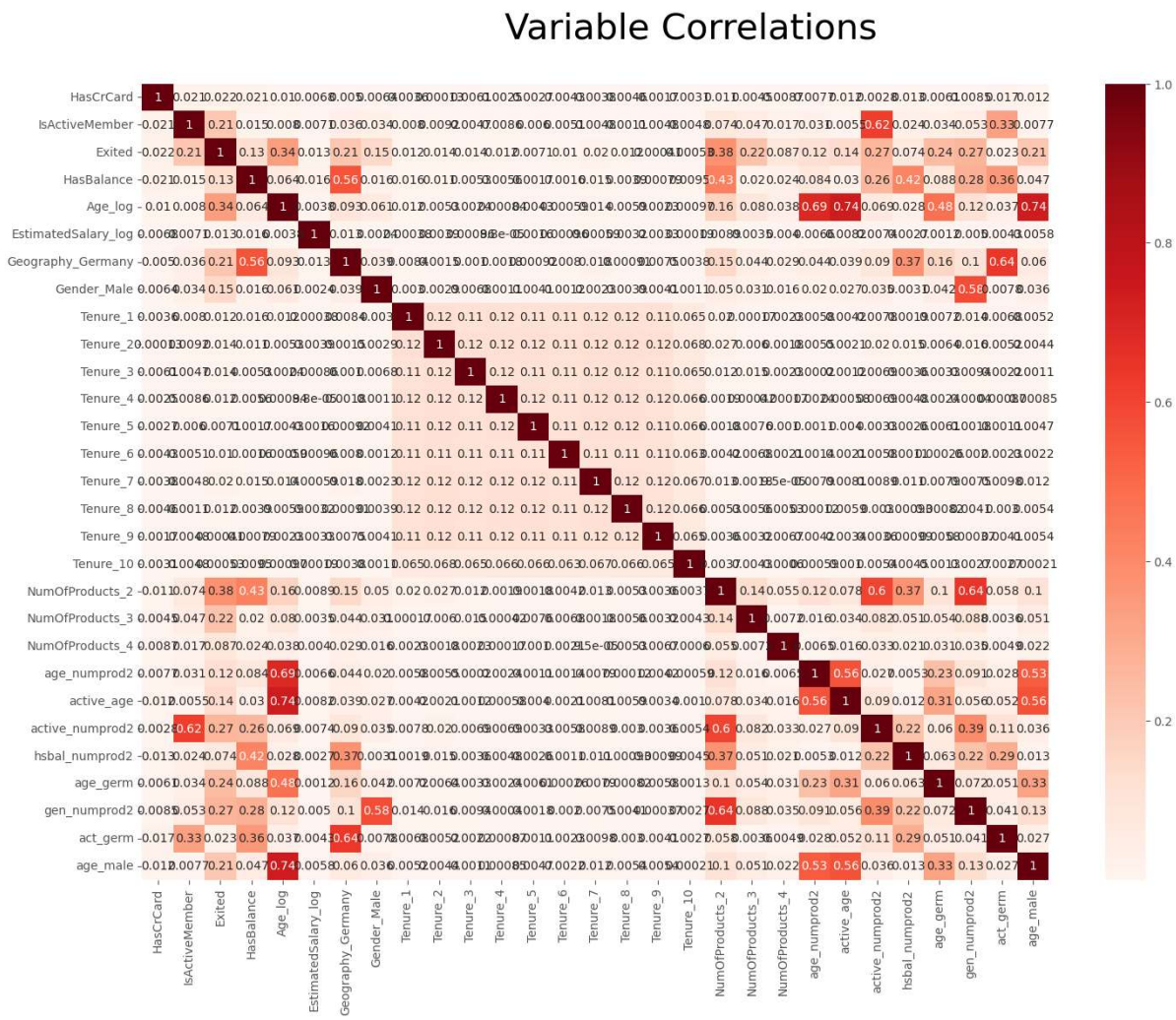


In [132]:
```
sns.jointplot(data = preprocessed5, x= 'Age_log', y ='Exited', kind='reg', joir
```

Out[132]: <seaborn.axisgrid.JointGrid at 0x2803f700e50>



In [133]:
```
corr = preprocessed5.select_dtypes('number').corr().abs()
fig, ax=plt.subplots(figsize=(17,12))
fig.suptitle('Variable Correlations', fontsize=30, y=.95)
heatmap = sns.heatmap(corr, cmap='Reds', annot=True)
```



In [134]:
```
features = []
correlations = []
for idx, correlation in corr['Exited'].T.items():
    if correlation >= .05 and idx != 'Exited':
        features.append(idx)
        correlations.append(correlation)
corr_with_price = pd.DataFrame({'Correlations':correlations, 'Features': featur

Multicollinear_Features = []
Multicollinear_Corr = []
def check_multicollinearity(feature):
    for idx, correlation in corr[feature].T.items():
        if correlation >= .80 and idx != feature:
            Multicollinear_Features.append([feature, idx])
            Multicollinear_Corr.append(correlation)

for feature in corr:
    check_multicollinearity(feature)
MC_df = pd.DataFrame({'Correlations':Multicollinear_Corr, 'Features': Multicoll
print('Multicollinear Features')
display(MC_df)
print('Correlations with Exited')
display(corr_with_price)
```

```
Multicollinear Features
```

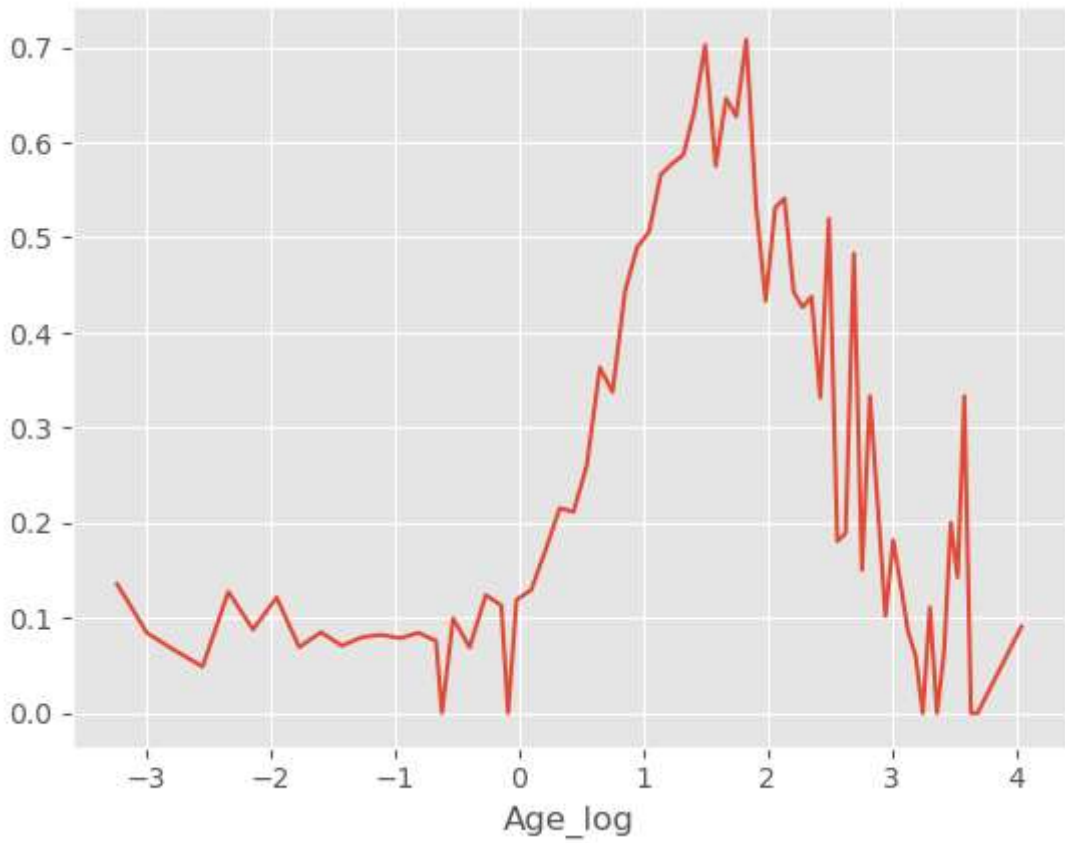
	Correlations	Features

```
Correlations with Exited
```

	Correlations	Features
0	0.210237	IsActiveMember
1	0.133346	HasBalance
2	0.337053	Age_log
3	0.211054	Geography_Germany
4	0.146442	Gender_Male
5	0.378176	NumOfProducts_2
6	0.219454	NumOfProducts_3
7	0.087367	NumOfProducts_4
8	0.118837	age_numprod2
9	0.142303	active_age
10	0.265678	active_numprod2
11	0.074215	hsbal_numprod2
12	0.242997	age_germ
13	0.268433	gen_numprod2
14	0.207640	age_male

In [135]:
```
preprocessed5.groupby(['Age_log'])['Exited'].mean().plot()
```

Out[135]: <Axes: xlabel='Age_log'>



In [136]:
```
preprocessed5.head()
```

Out[136]:

	HasCrCard	IsActiveMember	Exited	HasBalance	Age_log	EstimatedSalary_log	Geography_G
0	1.0	0.0	0	0.0	-0.530637	0.814347	
1	1.0	1.0	0	0.0	-0.530637	-0.761604	
2	1.0	0.0	0	0.0	0.327875	0.836980	
3	1.0	1.0	0	1.0	-0.397410	-0.111989	
4	1.0	1.0	0	0.0	-0.530637	-2.204676	

5 rows × 29 columns

In [137]:
```
test = preprocessed5
```

In [138]:
```
test.head()
```

Out[138]:

	HasCrCard	IsActiveMember	Exited	HasBalance	Age_log	EstimatedSalary_log	Geography_G
0	1.0	0.0	0	0.0	-0.530637	0.814347	
1	1.0	1.0	0	0.0	-0.530637	-0.761604	
2	1.0	0.0	0	0.0	0.327875	0.836980	
3	1.0	1.0	0	1.0	-0.397410	-0.111989	
4	1.0	1.0	0	0.0	-0.530637	-2.204676	

5 rows × 29 columns

In [139]:
```
test['Age_log_sq']=np.square(test['Age_log'])
```

In [140]:
```
test.head()
```

Out[140]:

	HasCrCard	IsActiveMember	Exited	HasBalance	Age_log	EstimatedSalary_log	Geography_G
0	1.0	0.0	0	0.0	-0.530637	0.814347	
1	1.0	1.0	0	0.0	-0.530637	-0.761604	
2	1.0	0.0	0	0.0	0.327875	0.836980	
3	1.0	1.0	0	1.0	-0.397410	-0.111989	
4	1.0	1.0	0	0.0	-0.530637	-2.204676	

5 rows × 30 columns

In [141]:
```
outcome = 'Exited'
x_cols = test.drop("Exited", axis=1)
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
modelt = ols(formula=formula, data=test).fit()
modelt.summary()
```

Out[141]:

OLS Regression Results

Dep. Variable:	Exited	**R-squared:**	0.367
Model:	OLS	**Adj. R-squared:**	0.367
Method:	Least Squares	**F-statistic:**	6611.
Date:	Wed, 28 Feb 2024	**Prob (F-statistic):**	0.00
Time:	12:19:24	**Log-Likelihood:**	-97219.
No. Observations:	330068	**AIC:**	1.945e+05
Df Residuals:	330038	**BIC:**	1.948e+05
Df Model:	29		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.5611	0.004	146.136	0.000	0.554	0.569
HasCrCard	-0.0184	0.001	-13.985	0.000	-0.021	-0.016
IsActiveMember	-0.1801	0.002	-99.912	0.000	-0.184	-0.177
HasBalance	-0.1698	0.002	-90.336	0.000	-0.173	-0.166
Age_log	0.1798	0.001	146.215	0.000	0.177	0.182
EstimatedSalary_log	0.0035	0.001	6.115	0.000	0.002	0.005
Geography_Germany	0.1896	0.002	86.441	0.000	0.185	0.194
Gender_Male	-0.1112	0.002	-67.552	0.000	-0.114	-0.108
Tenure_1	-0.0254	0.004	-6.875	0.000	-0.033	-0.018
Tenure_2	-0.0337	0.004	-9.179	0.000	-0.041	-0.027
Tenure_3	-0.0212	0.004	-5.717	0.000	-0.028	-0.014
Tenure_4	-0.0181	0.004	-4.922	0.000	-0.025	-0.011
Tenure_5	-0.0246	0.004	-6.669	0.000	-0.032	-0.017
Tenure_6	-0.0349	0.004	-9.377	0.000	-0.042	-0.028
Tenure_7	-0.0384	0.004	-10.451	0.000	-0.046	-0.031
Tenure_8	-0.0362	0.004	-9.833	0.000	-0.043	-0.029
Tenure_9	-0.0312	0.004	-8.427	0.000	-0.038	-0.024
Tenure_10	-0.0324	0.004	-7.341	0.000	-0.041	-0.024
NumOfProducts_2	-0.4274	0.002	-174.503	0.000	-0.432	-0.423
NumOfProducts_3	0.3808	0.004	86.899	0.000	0.372	0.389
NumOfProducts_4	0.3626	0.011	34.243	0.000	0.342	0.383
age_numprod2	-0.0835	0.001	-70.252	0.000	-0.086	-0.081
active_age	-0.0814	0.001	-68.925	0.000	-0.084	-0.079
active_numprod2	0.1110	0.002	47.610	0.000	0.106	0.116
hsbal_numprod2	0.1593	0.003	61.817	0.000	0.154	0.164

age_germ	0.0445	0.001	31.687	0.000	0.042	0.047
gen_numprod2	0.0757	0.002	32.585	0.000	0.071	0.080
act_germ	-0.0536	0.003	-18.934	0.000	-0.059	-0.048
age_male	-0.0160	0.001	-13.803	0.000	-0.018	-0.014
Age_log_sq	0.0162	0.000	42.296	0.000	0.015	0.017

Omnibus:	46985.889	**Durbin-Watson:**	2.004
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	77772.228
Skew:	0.969	**Prob(JB):**	0.00
Kurtosis:	4.379	**Cond. No.**	43.1

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [142]:
```
# adding another dimension to the age decreased the effectiveness of the model
# but due to the shape of the mean exited plotted through out the age,
# I am going to try to categorise this for improvement.
# Then check for better interactions
```

In [143]:
```
prep5b = preprocessed
```

In [144]:
```
prep5b['Age'].describe()
```

Out[144]:
```
count    330068.000000
mean         38.125888
std           8.867191
min          18.000000
25%          32.000000
50%          37.000000
75%          42.000000
max          92.000000
Name: Age, dtype: float64
```

In [145]:
```
# need to see if I can use polynominals on anything, if not then business insig
# how do I create a new csv for tableau?
```

In [146]:
```
#Create age groups
prep5b['age_u30'] =np.where(prep5b['Age']<30, 1, 0)
prep5b['age_30s'] =np.where((prep5b['Age']>=30) & (prep5b['Age']<40), 1, 0)
prep5b['age_40s'] =np.where((prep5b['Age']>=40) & (prep5b['Age']<50), 1, 0)
prep5b['age_50s'] =np.where((prep5b['Age']>=50) & (prep5b['Age']<60), 1, 0)
prep5b['age_60s'] =np.where((prep5b['Age']>=60) & (prep5b['Age']<70), 1, 0)
prep5b['age_70s'] =np.where((prep5b['Age']>=70) & (prep5b['Age']<80), 1, 0)
prep5b['age_o80'] =np.where((prep5b['Age']>=80), 1, 0)
```

In [147]:
```
prep5b.head()
```

Out[147]:

	Balance	HasCrCard	IsActiveMember	Exited	HasBalance	CreditScore_log	Age_log	Estima
0	1.00	1.0	0.0	0	0.0	0.200977	-0.530637	
1	1.00	1.0	1.0	0	0.0	-0.306421	-0.530637	
2	1.00	1.0	0.0	0	0.0	0.320006	0.327875	
3	148883.54	1.0	1.0	0	1.0	-0.916788	-0.397410	
4	1.00	1.0	1.0	0	0.0	0.756842	-0.530637	

5 rows × 45 columns

In [148]:
```
outcome = 'Exited'
x_cols = prep5b.drop("Exited", axis=1)
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model5b = ols(formula=formula, data=prep5b).fit()
model5b.summary()
```

Out[148]:

OLS Regression Results

Dep. Variable:	Exited	**R-squared:**	0.344
Model:	OLS	**Adj. R-squared:**	0.344
Method:	Least Squares	**F-statistic:**	4018.
Date:	Wed, 28 Feb 2024	**Prob (F-statistic):**	0.00
Time:	12:19:26	**Log-Likelihood:**	-1.0332e+05
No. Observations:	330068	**AIC:**	2.067e+05
Df Residuals:	330024	**BIC:**	2.072e+05
Df Model:	43		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.7367	0.231	-3.187	0.001	-1.190	-0.284
Balance	-5.472e-09	3.41e-08	-0.161	0.872	-7.23e-08	6.13e-08
HasCrCard	-0.0163	0.001	-12.171	0.000	-0.019	-0.014
IsActiveMember	-0.1333	0.001	-114.508	0.000	-0.136	-0.131
HasBalance	-0.1147	0.004	-26.040	0.000	-0.123	-0.106
CreditScore_log	0.0082	0.027	0.302	0.763	-0.045	0.062
Age_log	-0.1497	0.008	-19.703	0.000	-0.165	-0.135
EstimatedSalary_log	-0.0029	0.001	-2.694	0.007	-0.005	-0.001
CreditScore	-7.693e-06	0.000	-0.023	0.982	-0.001	0.001
Age	0.0293	0.001	30.623	0.000	0.027	0.031
EstimatedSalary	1.534e-07	2.16e-08	7.088	0.000	1.11e-07	1.96e-07
Geography_Germany	0.1932	0.002	108.158	0.000	0.190	0.197
Geography_Spain	0.0010	0.001	0.717	0.474	-0.002	0.004
Gender_Male	-0.0758	0.001	-64.906	0.000	-0.078	-0.073
Tenure_1	-0.0226	0.004	-6.004	0.000	-0.030	-0.015
Tenure_2	-0.0312	0.004	-8.344	0.000	-0.039	-0.024
Tenure_3	-0.0189	0.004	-5.016	0.000	-0.026	-0.012
Tenure_4	-0.0175	0.004	-4.664	0.000	-0.025	-0.010
Tenure_5	-0.0225	0.004	-5.998	0.000	-0.030	-0.015
Tenure_6	-0.0326	0.004	-8.598	0.000	-0.040	-0.025
Tenure_7	-0.0377	0.004	-10.065	0.000	-0.045	-0.030
Tenure_8	-0.0353	0.004	-9.407	0.000	-0.043	-0.028
Tenure_9	-0.0299	0.004	-7.917	0.000	-0.037	-0.022
Tenure_10	-0.0301	0.004	-6.693	0.000	-0.039	-0.021
NumOfProducts_2	-0.2558	0.001	-193.014	0.000	-0.258	-0.253

NumOfProducts_3	0.4097	0.004	91.941	0.000	0.401	0.418
NumOfProducts_4	0.3918	0.011	36.332	0.000	0.371	0.413
credit_classs_1	-0.0871	0.066	-1.314	0.189	-0.217	0.043
credit_classs_2	0.0460	0.057	0.800	0.424	-0.067	0.159
credit_classs_3	0.0251	0.060	0.421	0.674	-0.092	0.142
credit_classs_4	0.0325	0.062	0.526	0.599	-0.089	0.154
credit_classs_5	0.0159	0.063	0.250	0.803	-0.109	0.140
credit_classs_6	0.0093	0.064	0.145	0.885	-0.117	0.135
credit_classs_7	-0.0100	0.065	-0.154	0.878	-0.137	0.117
credit_classs_8	-0.0105	0.065	-0.163	0.871	-0.137	0.116
credit_classs_9	-0.0156	0.064	-0.244	0.807	-0.141	0.110
credit_classs_10	-0.0121	0.063	-0.192	0.848	-0.136	0.112
credit_classs_11	-0.0254	0.063	-0.404	0.686	-0.148	0.098
age_u30	0.1185	0.033	3.575	0.000	0.054	0.183
age_30s	0.0761	0.033	2.314	0.021	0.012	0.141
age_40s	0.1465	0.033	4.464	0.000	0.082	0.211
age_50s	0.1932	0.033	5.857	0.000	0.129	0.258
age_60s	-0.1116	0.034	-3.311	0.001	-0.178	-0.046
age_70s	-0.4512	0.036	-12.701	0.000	-0.521	-0.382
age_o80	-0.7082	0.046	-15.528	0.000	-0.798	-0.619

Omnibus:	35028.720	**Durbin-Watson:**	2.002
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	47610.323
Skew:	0.874	**Prob(JB):**	0.00
Kurtosis:	3.636	**Cond. No.**	2.43e+15

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.04e-15. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [149]:
```
prep5b = prep5b.drop(columns=['CreditScore', 'Age', 'EstimatedSalary'], axis=1)
```

In [150]:
```
#Split features and target
X = prep5b.drop(columns=['Exited'], axis=1)
y = prep5b.Exited

#Utilize cross validation to determine baseline score
crossvalidation = KFold(n_splits=10, shuffle=True, random_state=1)
baseline = np.mean(cross_val_score(linreg, X, y, scoring='r2', cv=crossvalidati

from itertools import combinations
interactions = []
feat_combinations = combinations(X.columns, 2)
data = X.copy()
for i, (a, b) in enumerate(feat_combinations):
    data['interaction'] = data[a] * data[b]
    score = np.mean(cross_val_score(linreg, data, y, scoring='r2', cv=crossvali
    if score > baseline:
        interactions.append((a, b, round(score,3)))

    if i % 50 == 0:
        print(i)

print('Top 20 interactions: %s' %sorted(interactions, key=lambda inter: inter[2
```

```
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
Top 20 interactions: [('IsActiveMember', 'Age_log', 0.354), ('Age_log', 'NumO
fProducts_2', 0.353), ('Age_log', 'age_40s', 0.351), ('IsActiveMember', 'NumO
fProducts_2', 0.349), ('NumOfProducts_2', 'age_40s', 0.349), ('Balance', 'Num
OfProducts_2', 0.347), ('IsActiveMember', 'age_50s', 0.347), ('HasBalance',
'NumOfProducts_2', 0.347), ('NumOfProducts_2', 'age_30s', 0.347), ('IsActiveM
ember', 'age_30s', 0.346), ('Gender_Male', 'NumOfProducts_2', 0.345), ('IsAct
iveMember', 'age_40s', 0.344), ('IsActiveMember', 'age_60s', 0.344), ('Age_lo
g', 'Geography_Germany', 0.344), ('NumOfProducts_2', 'age_u30', 0.344), ('Num
OfProducts_2', 'age_50s', 0.344), ('Balance', 'IsActiveMember', 0.343), ('IsA
ctiveMember', 'HasBalance', 0.343), ('IsActiveMember', 'Geography_Germany',
0.343), ('IsActiveMember', 'age_u30', 0.343)]
```

In [151]:

```
## add interactions above .344

preprocessed6 = prep5b
preprocessed6['act_age']= preprocessed6['IsActiveMember'] * preprocessed6['Age_
preprocessed6['age_nprod2']= preprocessed6['Age_log'] * preprocessed6['NumOfPro
preprocessed6['age_40s']= preprocessed6['Age_log'] * preprocessed6['age_40s']
preprocessed6['act_nprod2']= preprocessed6['IsActiveMember'] * preprocessed6['N
preprocessed6['nprod2_40s']= preprocessed6['NumOfProducts_2'] * preprocessed6['
preprocessed6['act_50s']= preprocessed6['IsActiveMember'] * preprocessed6['age_
preprocessed6['hbal_nprod2']= preprocessed6['HasBalance'] * preprocessed6['NumO
preprocessed6['nprod2_30s']= preprocessed6['NumOfProducts_2'] * preprocessed6['
preprocessed6['bal_nprod2']= preprocessed6['Balance'] * preprocessed6['NumOfPro
preprocessed6['act_30s']= preprocessed6['IsActiveMember'] * preprocessed6['age_
preprocessed6['act_40s']= preprocessed6['IsActiveMember'] * preprocessed6['age_
preprocessed6['act_60s']= preprocessed6['IsActiveMember'] * preprocessed6['age_
preprocessed6['m_nprod2']= preprocessed6['Gender_Male'] * preprocessed6['NumOfP
```

In [152]:
```
outcome = 'Exited'
x_cols = preprocessed6.drop("Exited", axis=1)
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model6 = ols(formula=formula, data=preprocessed6).fit()
model6.summary()
```

Out[152]:

OLS Regression Results

Dep. Variable:	Exited	**R-squared:**	0.387
Model:	OLS	**Adj. R-squared:**	0.387
Method:	Least Squares	**F-statistic:**	3929.
Date:	Wed, 28 Feb 2024	**Prob (F-statistic):**	0.00
Time:	13:09:29	**Log-Likelihood:**	-92072.
No. Observations:	330068	**AIC:**	1.843e+05
Df Residuals:	330014	**BIC:**	1.848e+05
Df Model:	53		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.4392	0.055	8.053	0.000	0.332	0.546
Balance	9.202e-08	3.92e-08	2.349	0.019	1.52e-08	1.69e-07
HasCrCard	-0.0167	0.001	-12.905	0.000	-0.019	-0.014
IsActiveMember	-0.1628	0.004	-40.630	0.000	-0.171	-0.155
HasBalance	-0.1683	0.005	-32.919	0.000	-0.178	-0.158
CreditScore_log	0.0084	0.003	2.617	0.009	0.002	0.015
Age_log	0.0746	0.002	32.672	0.000	0.070	0.079
EstimatedSalary_log	0.0029	0.001	5.163	0.000	0.002	0.004
Geography_Germany	0.1608	0.002	92.029	0.000	0.157	0.164
Geography_Spain	0.0012	0.001	0.866	0.387	-0.002	0.004
Gender_Male	-0.1083	0.002	-67.336	0.000	-0.111	-0.105
Tenure_1	-0.0249	0.004	-6.844	0.000	-0.032	-0.018
Tenure_2	-0.0334	0.004	-9.230	0.000	-0.040	-0.026
Tenure_3	-0.0210	0.004	-5.758	0.000	-0.028	-0.014
Tenure_4	-0.0189	0.004	-5.221	0.000	-0.026	-0.012
Tenure_5	-0.0243	0.004	-6.694	0.000	-0.031	-0.017
Tenure_6	-0.0342	0.004	-9.310	0.000	-0.041	-0.027
Tenure_7	-0.0368	0.004	-10.175	0.000	-0.044	-0.030
Tenure_8	-0.0351	0.004	-9.685	0.000	-0.042	-0.028
Tenure_9	-0.0301	0.004	-8.260	0.000	-0.037	-0.023
Tenure_10	-0.0304	0.004	-6.999	0.000	-0.039	-0.022
NumOfProducts_2	-0.3999	0.003	-128.673	0.000	-0.406	-0.394
NumOfProducts_3	0.3677	0.004	85.166	0.000	0.359	0.376
NumOfProducts_4	0.3520	0.010	33.761	0.000	0.332	0.372
credit_classs_1	-0.0943	0.064	-1.477	0.140	-0.220	0.031

credit_classs_2	0.0715	0.053	1.354	0.176	-0.032	0.175
credit_classs_3	0.0457	0.053	0.868	0.385	-0.058	0.149
credit_classs_4	0.0508	0.053	0.958	0.338	-0.053	0.155
credit_classs_5	0.0345	0.054	0.644	0.520	-0.070	0.139
credit_classs_6	0.0268	0.054	0.495	0.621	-0.079	0.133
credit_classs_7	0.0085	0.055	0.155	0.877	-0.099	0.116
credit_classs_8	0.0078	0.055	0.142	0.887	-0.100	0.116
credit_classs_9	0.0009	0.056	0.016	0.987	-0.108	0.110
credit_classs_10	0.0062	0.057	0.110	0.912	-0.105	0.117
credit_classs_11	-0.0059	0.057	-0.104	0.918	-0.117	0.106
age_u30	0.0504	0.004	11.822	0.000	0.042	0.059
age_30s	0.0060	0.004	1.632	0.103	-0.001	0.013
age_40s	0.3174	0.006	56.878	0.000	0.307	0.328
age_50s	0.4006	0.006	65.111	0.000	0.389	0.413
age_60s	0.3307	0.009	34.996	0.000	0.312	0.349
age_70s	0.0314	0.011	2.920	0.003	0.010	0.052
age_o80	-0.0220	0.030	-0.721	0.471	-0.082	0.038
act_age	-0.0306	0.002	-12.815	0.000	-0.035	-0.026
age_nprod2	-0.0531	0.001	-40.442	0.000	-0.056	-0.051
act_nprod2	0.1027	0.002	44.689	0.000	0.098	0.107
nprod2_40s	-0.1088	0.005	-24.065	0.000	-0.118	-0.100
act_50s	-0.1497	0.008	-18.106	0.000	-0.166	-0.133
hbal_nprod2	0.1663	0.009	18.261	0.000	0.148	0.184
nprod2_30s	0.0325	0.003	11.895	0.000	0.027	0.038
bal_nprod2	-2.208e-07	7.23e-08	-3.052	0.002	-3.63e-07	-7.9e-08
act_30s	0.0170	0.004	4.646	0.000	0.010	0.024
act_40s	-0.0582	0.007	-7.894	0.000	-0.073	-0.044
act_60s	-0.2058	0.012	-17.402	0.000	-0.229	-0.183
m_nprod2	0.0770	0.002	34.109	0.000	0.073	0.081

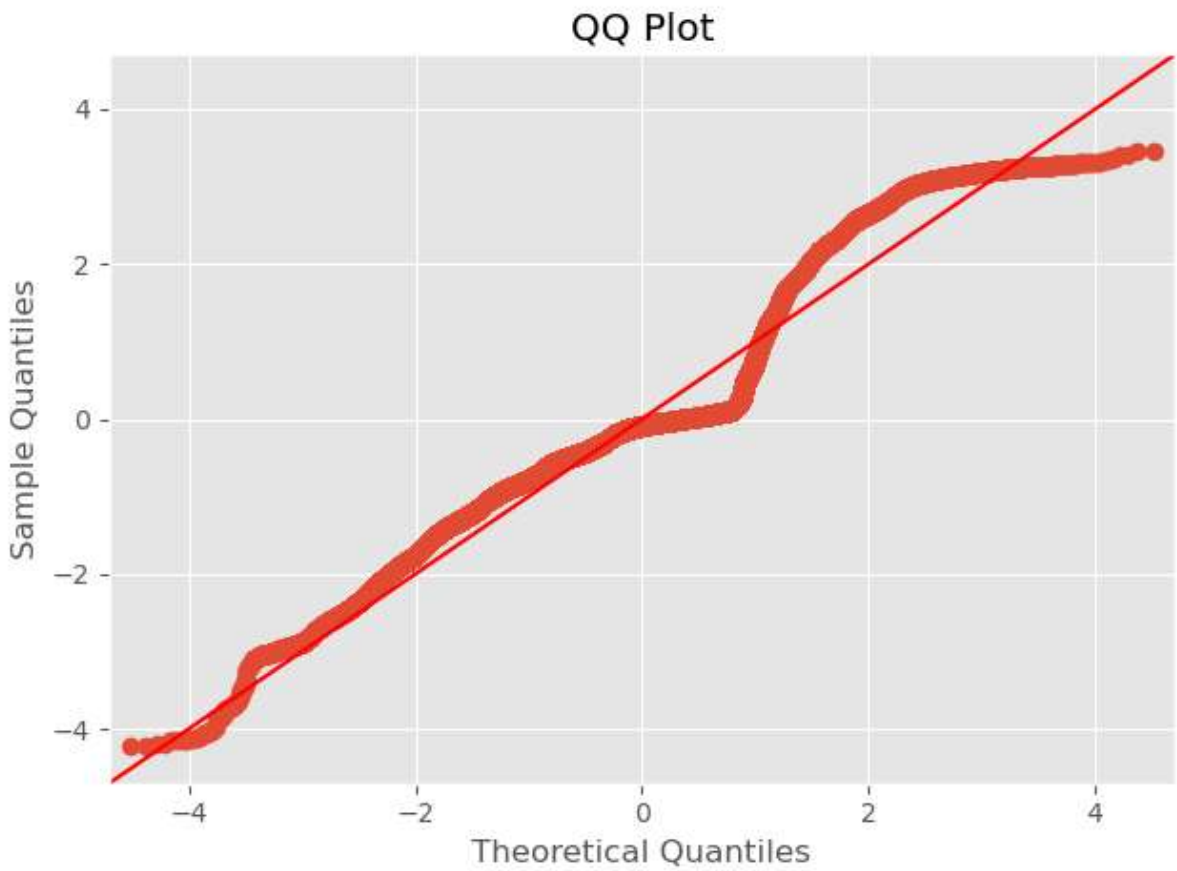
Omnibus:	49424.835	**Durbin-Watson:**	2.006
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	85591.353
Skew:	0.989	**Prob(JB):**	0.00
Kurtosis:	4.521	**Cond. No.**	2.94e+07

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

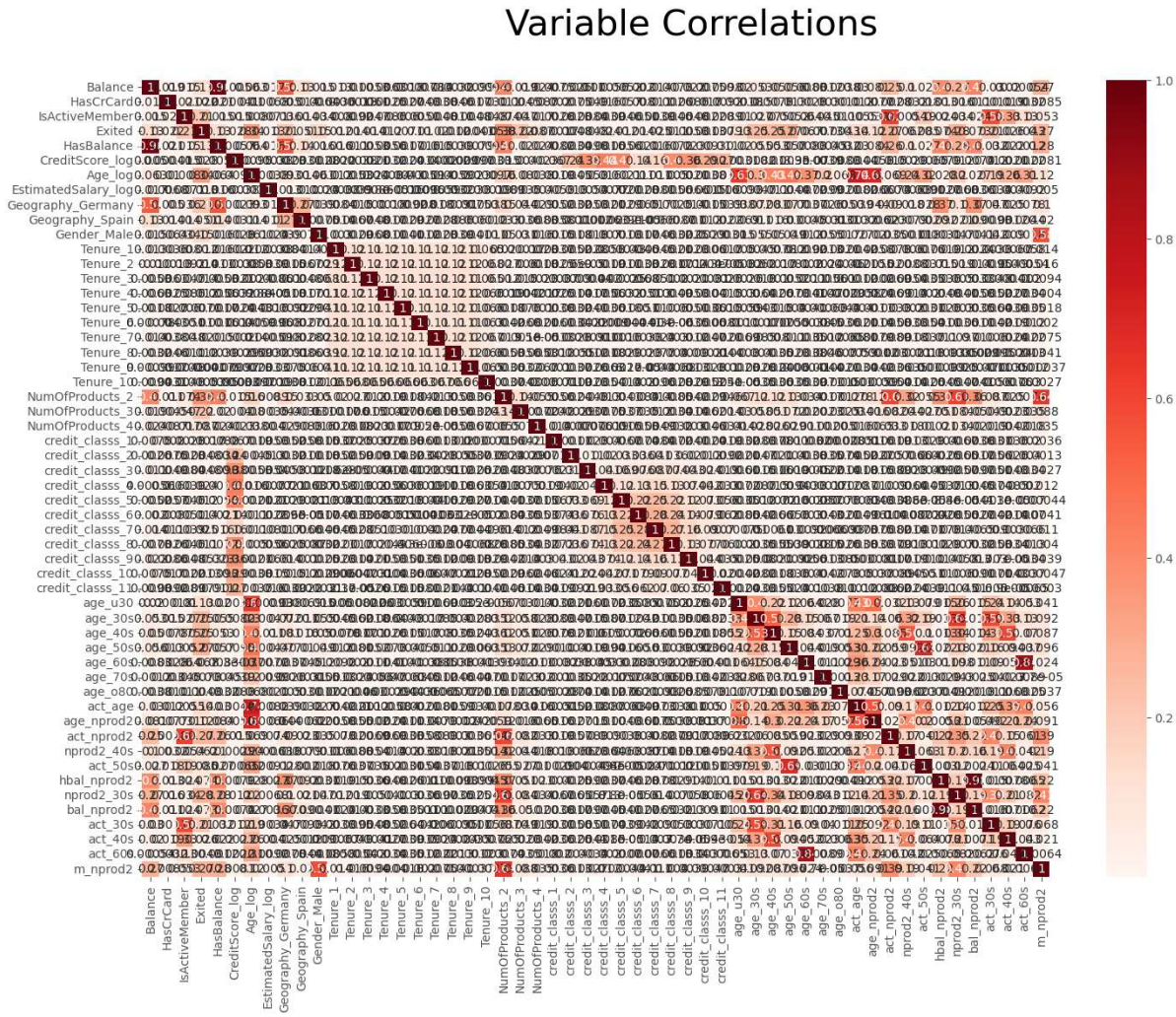
[2] The condition number is large, 2.94e+07. This might indicate that there are strong multicollinearity or other numerical problems.

In [153]:
```
plt.figure(figsize=(12, 8))
fig = sm.graphics.qqplot(model6.resid, dist=stats.norm, line='45', fit=True)
plt.title('QQ Plot')
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



In [154]:
```
corr = preprocessed6.select_dtypes('number').corr().abs()
fig, ax=plt.subplots(figsize=(17,12))
fig.suptitle('Variable Correlations', fontsize=30, y=.95)
heatmap = sns.heatmap(corr, cmap='Reds', annot=True)
```



In [155]:
```
features = []
correlations = []
for idx, correlation in corr['Exited'].T.items():
    if correlation >= .05 and idx != 'Exited':
        features.append(idx)
        correlations.append(correlation)
corr_with_price = pd.DataFrame({'Correlations':correlations, 'Features': featur

Multicollinear_Features = []
Multicollinear_Corr = []
def check_multicollinearity(feature):
    for idx, correlation in corr[feature].T.items():
        if correlation >= .80 and idx != feature:
            Multicollinear_Features.append([feature, idx])
            Multicollinear_Corr.append(correlation)

for feature in corr:
    check_multicollinearity(feature)
MC_df = pd.DataFrame({'Correlations':Multicollinear_Corr, 'Features': Multicoll
print('Multicollinear Features')
display(MC_df)
print('Correlations with Exited')
display(corr_with_price)
```

```
Multicollinear Features
```

	Correlations	Features
0	0.963088	[Balance, HasBalance]
1	0.963088	[HasBalance, Balance]
2	0.835783	[age_60s, act_60s]
3	0.975098	[hbal_nprod2, bal_nprod2]
4	0.975098	[bal_nprod2, hbal_nprod2]
5	0.835783	[act_60s, age_60s]

```
Correlations with Exited
```

	Correlations	Features
0	0.129743	Balance
1	0.210237	IsActiveMember
2	0.133346	HasBalance
3	0.337053	Age_log
4	0.211054	Geography_Germany
5	0.051175	Geography_Spain
6	0.146442	Gender_Male
7	0.378176	NumOfProducts_2
8	0.219454	NumOfProducts_3
9	0.087367	NumOfProducts_4
10	0.127865	age_u30
11	0.249085	age_30s
12	0.247708	age_40s
13	0.274994	age_50s
14	0.067274	age_60s
15	0.142303	act_age
16	0.118837	age_nprod2
17	0.265678	act_nprod2
18	0.062140	nprod2_40s
19	0.084759	act_50s
20	0.074215	hbal_nprod2
21	0.275112	nprod2_30s
22	0.072632	bal_nprod2
23	0.209614	act_30s
24	0.268433	m_nprod2

In [156]:
```
# Drop insignificant features
preprocessed7=preprocessed6.drop(columns=['Balance','CreditScore_log','credit_c
                                          'credit_classs_4', 'credit_classs_5',
                                          'credit_classs_9', 'credit_classs_10',
```

In [157]:
```
# drop remaining multicolinarity feature
preprocessed7.drop(columns = ['age_60s'], inplace=True)
```

In [158]:
```
outcome = 'Exited'
x_cols = preprocessed7.drop("Exited", axis=1)
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model7 = ols(formula=formula, data=preprocessed7).fit()
model7.summary()
```

Out[158]:

OLS Regression Results

Dep. Variable:	Exited	**R-squared:**	0.384
Model:	OLS	**Adj. R-squared:**	0.384
Method:	Least Squares	**F-statistic:**	5417.
Date:	Wed, 28 Feb 2024	**Prob (F-statistic):**	0.00
Time:	13:09:39	**Log-Likelihood:**	-92805.
No. Observations:	330068	**AIC:**	1.857e+05
Df Residuals:	330029	**BIC:**	1.861e+05
Df Model:	38		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.5166	0.005	107.789	0.000	0.507	0.526
HasCrCard	-0.0171	0.001	-13.176	0.000	-0.020	-0.015
IsActiveMember	-0.2176	0.004	-58.840	0.000	-0.225	-0.210
HasBalance	-0.1576	0.002	-85.336	0.000	-0.161	-0.154
Age_log	0.1235	0.002	68.253	0.000	0.120	0.127
EstimatedSalary_log	0.0029	0.001	5.130	0.000	0.002	0.004
Geography_Germany	0.1611	0.002	92.070	0.000	0.158	0.165
Geography_Spain	0.0016	0.001	1.155	0.248	-0.001	0.004
Gender_Male	-0.1090	0.002	-67.587	0.000	-0.112	-0.106
Tenure_1	-0.0241	0.004	-6.595	0.000	-0.031	-0.017
Tenure_2	-0.0330	0.004	-9.109	0.000	-0.040	-0.026
Tenure_3	-0.0203	0.004	-5.553	0.000	-0.027	-0.013
Tenure_4	-0.0188	0.004	-5.165	0.000	-0.026	-0.012
Tenure_5	-0.0236	0.004	-6.478	0.000	-0.031	-0.016
Tenure_6	-0.0337	0.004	-9.163	0.000	-0.041	-0.026
Tenure_7	-0.0365	0.004	-10.074	0.000	-0.044	-0.029
Tenure_8	-0.0350	0.004	-9.627	0.000	-0.042	-0.028
Tenure_9	-0.0299	0.004	-8.176	0.000	-0.037	-0.023
Tenure_10	-0.0304	0.004	-6.985	0.000	-0.039	-0.022
NumOfProducts_2	-0.4071	0.003	-130.966	0.000	-0.413	-0.401
NumOfProducts_3	0.3696	0.004	85.448	0.000	0.361	0.378
NumOfProducts_4	0.3538	0.010	33.860	0.000	0.333	0.374
age_u30	0.0661	0.004	15.549	0.000	0.058	0.074
age_30s	-0.0361	0.004	-10.318	0.000	-0.043	-0.029
age_40s	0.2002	0.004	44.812	0.000	0.191	0.209

age_50s	0.2668	0.005	55.338	0.000	0.257	0.276
age_70s	-0.0286	0.011	-2.696	0.007	-0.049	-0.008
age_o80	-0.0987	0.030	-3.240	0.001	-0.158	-0.039
act_age	-0.0667	0.002	-30.847	0.000	-0.071	-0.062
age_nprod2	-0.0578	0.001	-44.112	0.000	-0.060	-0.055
act_nprod2	0.1062	0.002	46.149	0.000	0.102	0.111
nprod2_40s	-0.1027	0.005	-22.680	0.000	-0.112	-0.094
act_50s	-0.0361	0.008	-4.731	0.000	-0.051	-0.021
hbal_nprod2	0.1408	0.003	56.082	0.000	0.136	0.146
nprod2_30s	0.0359	0.003	13.156	0.000	0.031	0.041
act_30s	0.0592	0.003	17.116	0.000	0.052	0.066
act_40s	0.0438	0.007	6.439	0.000	0.030	0.057
act_60s	0.0986	0.008	12.250	0.000	0.083	0.114
m_nprod2	0.0775	0.002	34.236	0.000	0.073	0.082

Omnibus:	48411.933	**Durbin-Watson:**	2.006
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	82777.523
Skew:	0.977	**Prob(JB):**	0.00
Kurtosis:	4.483	**Cond. No.**	106.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [159]:
```
# Clean up insignificant features
preprocessed7.drop(columns = ['Geography_Spain', 'age_u30', 'act_40s', 'act_60s
```

In [160]:
```
outcome = 'Exited'
x_cols = preprocessed7.drop("Exited", axis=1)
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model7 = ols(formula=formula, data=preprocessed7).fit()
model7.summary()
```

Out[160]:

OLS Regression Results

Dep. Variable:	Exited	**R-squared:**	0.383
Model:	OLS	**Adj. R-squared:**	0.383
Method:	Least Squares	**F-statistic:**	6034.
Date:	Wed, 28 Feb 2024	**Prob (F-statistic):**	0.00
Time:	13:09:40	**Log-Likelihood:**	-93017.
No. Observations:	330068	**AIC:**	1.861e+05
Df Residuals:	330033	**BIC:**	1.865e+05
Df Model:	34		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.5524	0.004	132.502	0.000	0.544	0.561
HasCrCard	-0.0171	0.001	-13.155	0.000	-0.020	-0.015
IsActiveMember	-0.1941	0.002	-94.149	0.000	-0.198	-0.190
HasBalance	-0.1574	0.002	-85.190	0.000	-0.161	-0.154
Age_log	0.1100	0.001	83.593	0.000	0.107	0.113
EstimatedSalary_log	0.0028	0.001	4.981	0.000	0.002	0.004
Geography_Germany	0.1606	0.002	94.359	0.000	0.157	0.164
Gender_Male	-0.1092	0.002	-67.689	0.000	-0.112	-0.106
Tenure_1	-0.0244	0.004	-6.674	0.000	-0.032	-0.017
Tenure_2	-0.0333	0.004	-9.193	0.000	-0.040	-0.026
Tenure_3	-0.0206	0.004	-5.632	0.000	-0.028	-0.013
Tenure_4	-0.0192	0.004	-5.274	0.000	-0.026	-0.012
Tenure_5	-0.0240	0.004	-6.596	0.000	-0.031	-0.017
Tenure_6	-0.0343	0.004	-9.312	0.000	-0.041	-0.027
Tenure_7	-0.0369	0.004	-10.173	0.000	-0.044	-0.030
Tenure_8	-0.0355	0.004	-9.762	0.000	-0.043	-0.028
Tenure_9	-0.0303	0.004	-8.299	0.000	-0.037	-0.023
Tenure_10	-0.0311	0.004	-7.143	0.000	-0.040	-0.023
NumOfProducts_2	-0.4041	0.003	-130.225	0.000	-0.410	-0.398
NumOfProducts_3	0.3702	0.004	85.525	0.000	0.362	0.379
NumOfProducts_4	0.3535	0.010	33.808	0.000	0.333	0.374
age_30s	-0.0752	0.002	-30.650	0.000	-0.080	-0.070
age_40s	0.1713	0.003	52.098	0.000	0.165	0.178
age_50s	0.2533	0.004	60.171	0.000	0.245	0.262
age_70s	-0.0818	0.009	-9.156	0.000	-0.099	-0.064

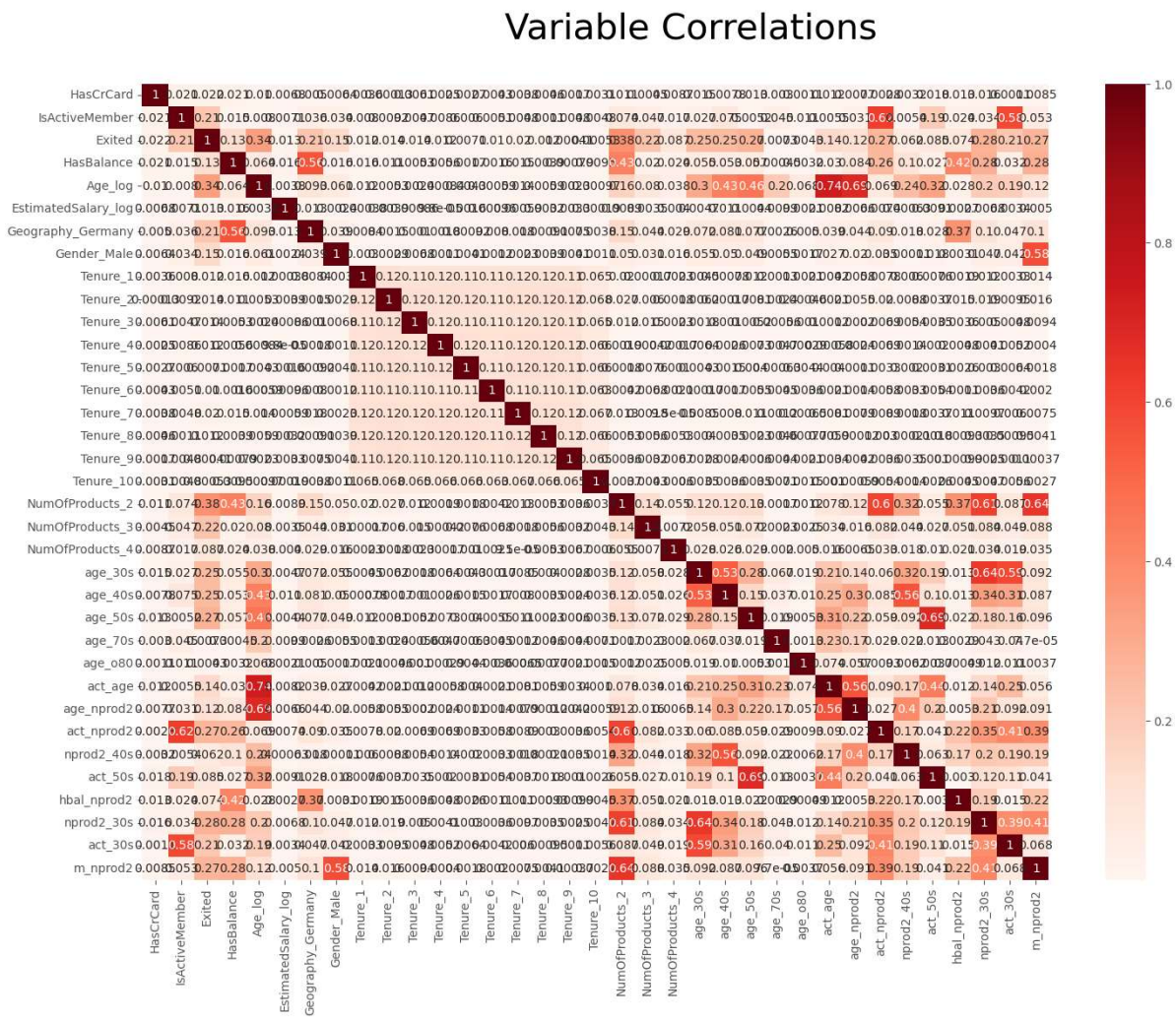
age_o80	-0.1488	0.030	-4.994	0.000	-0.207	-0.090
act_age	-0.0509	0.001	-38.094	0.000	-0.054	-0.048
age_nprod2	-0.0586	0.001	-44.861	0.000	-0.061	-0.056
act_nprod2	0.1057	0.002	45.925	0.000	0.101	0.110
nprod2_40s	-0.1117	0.004	-25.018	0.000	-0.120	-0.103
act_50s	-0.0851	0.005	-17.137	0.000	-0.095	-0.075
hbal_nprod2	0.1410	0.003	56.119	0.000	0.136	0.146
nprod2_30s	0.0329	0.003	12.068	0.000	0.028	0.038
act_30s	0.0407	0.002	17.023	0.000	0.036	0.045
m_nprod2	0.0774	0.002	34.199	0.000	0.073	0.082

Omnibus:	48248.908	**Durbin-Watson:**	2.006
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	82273.032
Skew:	0.976	**Prob(JB):**	0.00
Kurtosis:	4.475	**Cond. No.**	102.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [161]:
```
# The results of the final model show that; #
# the Prob(Omnibus) is close to zero indicateting that the residuals are follow
# the skew indicates a slightly posative skew.
# the kurtosis value indicates a moderately peaked distribution.
# the Druban-Watson is a little higher than the ideal 1-2 but at 2.006 this wou
# the JB test checks normality and the high value, combined with a very low p=-
# the condition number is relatively low indicating no multicollinearity.

# This model has increased the r^2 to 0.383, wich is a significant change from
# There are no features that do not show a p value greater than 0.0004, meaning
```

In [162]:
```
corr = preprocessed7.select_dtypes('number').corr().abs()
fig, ax=plt.subplots(figsize=(17,12))
fig.suptitle('Variable Correlations', fontsize=30, y=.95)
heatmap = sns.heatmap(corr, cmap='Reds', annot=True)
```



In [163]:

```
features = []
correlations = []
for idx, correlation in corr['Exited'].T.items():
    if correlation >= .05 and idx != 'Exited':
        features.append(idx)
        correlations.append(correlation)
corr_with_price = pd.DataFrame({'Correlations':correlations, 'Features': featur

Multicollinear_Features = []
Multicollinear_Corr = []
def check_multicollinearity(feature):
    for idx, correlation in corr[feature].T.items():
        if correlation >= .80 and idx != feature:
            Multicollinear_Features.append([feature, idx])
            Multicollinear_Corr.append(correlation)

for feature in corr:
    check_multicollinearity(feature)
MC_df = pd.DataFrame({'Correlations':Multicollinear_Corr, 'Features': Multicoll
print('Multicollinear Features')
display(MC_df)
print('Correlations with Exited')
display(corr_with_price)
```

```
Multicollinear Features
```

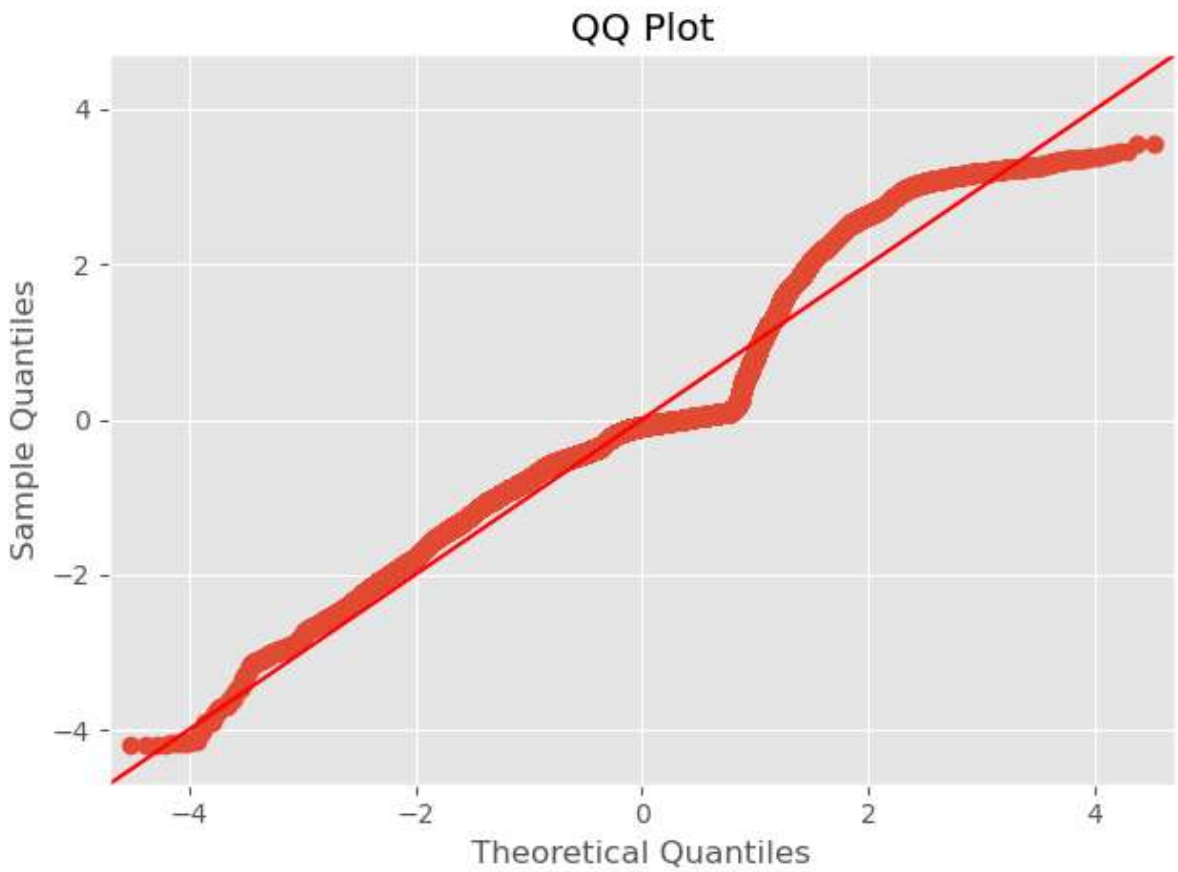
Correlations	Features

```
Correlations with Exited
```

	Correlations	Features
0	0.210237	IsActiveMember
1	0.133346	HasBalance
2	0.337053	Age_log
3	0.211054	Geography_Germany
4	0.146442	Gender_Male
5	0.378176	NumOfProducts_2
6	0.219454	NumOfProducts_3
7	0.087367	NumOfProducts_4
8	0.249085	age_30s
9	0.247708	age_40s
10	0.274994	age_50s
11	0.142303	act_age
12	0.118837	age_nprod2
13	0.265678	act_nprod2
14	0.062140	nprod2_40s
15	0.084759	act_50s
16	0.074215	hbal_nprod2
17	0.275112	nprod2_30s
18	0.209614	act_30s
19	0.268433	m_nprod2

In [164]:
```
plt.figure(figsize=(12, 8))
fig = sm.graphics.qqplot(model7.resid, dist=stats.norm, line='45', fit=True)
plt.title('QQ Plot')
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



In [165]:
```
##plt.figure(figsize=(10,5))
##sns.regplot(x=model7.predict(), y=model7.resid, lowess=True, line_kws={'color
##plt.title('Residuals Scatterplot', fontsize=16, y=.99)
```

In [166]:
```
#create test and training data subsets
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
```

In [167]:
```
X_train, X_test, y_train, y_test = train_test_split(preprocessed7, preprocessed
train, test = train_test_split(preprocessed7,test_size=.25,shuffle=True)
```

In [168]:
```
#df_train = pd.DataFrame(X_train, y_train)
#df_test = pd.DataFrame(X_test, y_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[168], line 1
----> 1 df_train = pd.DataFrame(X_train, y_train)
      2 df_test = pd.DataFrame(X_test, y_test)

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:703, in DataFrame.__i
nit__(self, data, index, columns, dtype, copy)
    700     data = []
    702 if isinstance(data, (BlockManager, ArrayManager)):
--> 703     mgr = self._init_mgr(
    704         data, axes={"index": index, "columns": columns}, dtype=dtype,
copy=copy
    705     )
    707 elif isinstance(data, dict):
    708     # GH#38939 de facto copy defaults to False only in non-dict cases
    709     mgr = dict_to_mgr(data, index, columns, dtype=dtype, copy=copy, t
yp=manager)

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:296, in NDFrame._in
it_mgr(cls, mgr, axes, dtype, copy)
    294         axe = ensure_index(axe)
    295         bm_axis = cls._get_block_manager_axis(a)
--> 296         mgr = mgr.reindex_axis(axe, axis=bm_axis)
    298 # make a copy if explicitly requested
    299 if copy:

File ~\anaconda3\Lib\site-packages\pandas\core\internals\base.py:98, in DataM
anager.reindex_axis(self, new_index, axis, fill_value, only_slice)
     87 @final
     88 def reindex_axis(
     89     self: T,
   (...)
     93     only_slice: bool = False,
     94 ) -> T:
     95     """
     96     Conform data manager to new index.
     97     """
---> 98     new_index, indexer = self.axes[axis].reindex(new_index)
    100     return self.reindex_indexer(
    101         new_index,
    102         indexer,
   (...)
    106         only_slice=only_slice,
    107     )

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:4275, in Inde
x.reindex(self, target, method, level, limit, tolerance)
   4272     raise ValueError("cannot handle a non-unique multi-index!")
   4273 elif not self.is_unique:
   4274     # GH#42568
-> 4275     raise ValueError("cannot reindex on an axis with duplicate label
s")
   4276 else:
   4277     indexer, _ = self.get_indexer_non_unique(target)
```

```
ValueError: cannot reindex on an axis with duplicate labels
```

In [169]:
```
#fit the data
linreg = LinearRegression()
linreg.fit(X_train, y_train)
```

Out[169]:
```
▾ LinearRegression
LinearRegression()
```

In [170]:
```
y_pred_train = model7.predict(X_train)
y_pred_test = model7.predict(X_test)
```

In [171]:
```
train_mse = np.mean(y_train-y_pred_train)**2
test_mse = np.mean(y_test-y_pred_test)**2

train_r2 = r2_score(train['Exited'], model7.predict(train))
test_r2 = r2_score(test['Exited'], model7.predict(test))

print('Mean Squared Error Train:', train_mse)
print('Mean Squared Error Test:', test_mse)
print(f'R-Squared Train = {round(train_r2,3)}')
print(f'R-Squared Test = {round(test_r2,3)}')
```

```
Mean Squared Error Train: 7.022970600344432e-08
Mean Squared Error Test: 6.320673536380917e-07
R-Squared Train = 0.384
R-Squared Test = 0.382
```

In [172]:
```
# the model is performing well with minimal difference between the train and te
# but it is not very effective with its outcome
# we will try a decision tree model that may be better suited to the data
```

In [173]:
```
# Load libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifi
from sklearn import metrics #Import scikit-learn metrics module for accuracy ca
from six import StringIO
```

In [174]:
```
preprocessed.head()
```

Out[174]:

	Balance	HasCrCard	IsActiveMember	Exited	HasBalance	CreditScore_log	Age_log	Estima
0	1.00	1.0	0.0	0	0.0	0.200977	-0.530637	
1	1.00	1.0	1.0	0	0.0	-0.306421	-0.530637	
2	1.00	1.0	0.0	0	0.0	0.320006	0.327875	
3	148883.54	1.0	1.0	0	1.0	-0.916788	-0.397410	
4	1.00	1.0	1.0	0	0.0	0.756842	-0.530637	

5 rows × 45 columns

In [175]:
```
cln.head()
```

Out[175]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiv
0	668	France	Male	33.0	3.0	1.00	2.0	1.0	
1	627	France	Male	33.0	1.0	1.00	2.0	1.0	
2	678	France	Male	40.0	10.0	1.00	2.0	1.0	
3	581	France	Male	34.0	2.0	148883.54	1.0	1.0	
4	716	Spain	Male	33.0	5.0	1.00	2.0	1.0	

In [176]:
```
preprocessed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 330068 entries, 0 to 165033
Data columns (total 45 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   Balance              330068 non-null  float64
 1   HasCrCard            330068 non-null  float64
 2   IsActiveMember       330068 non-null  float64
 3   Exited               330068 non-null  int64
 4   HasBalance           330068 non-null  float64
 5   CreditScore_log      330068 non-null  float64
 6   Age_log              330068 non-null  float64
 7   EstimatedSalary_log  330068 non-null  float64
 8   CreditScore          330068 non-null  int64
 9   Age                  330068 non-null  float64
 10  EstimatedSalary      330068 non-null  float64
 11  Geography_Germany    330068 non-null  float64
 12  Geography_Spain      330068 non-null  float64
 13  Gender_Male          330068 non-null  float64
 14  Tenure_1             330068 non-null  float64
 15  Tenure_2             330068 non-null  float64
 16  Tenure_3             330068 non-null  float64
 17  Tenure_4             330068 non-null  float64
 18  Tenure_5             330068 non-null  float64
 19  Tenure_6             330068 non-null  float64
 20  Tenure_7             330068 non-null  float64
 21  Tenure_8             330068 non-null  float64
 22  Tenure_9             330068 non-null  float64
 23  Tenure_10            330068 non-null  float64
 24  NumOfProducts_2      330068 non-null  float64
 25  NumOfProducts_3      330068 non-null  float64
 26  NumOfProducts_4      330068 non-null  float64
 27  credit_classs_1      330068 non-null  float64
 28  credit_classs_2      330068 non-null  float64
 29  credit_classs_3      330068 non-null  float64
 30  credit_classs_4      330068 non-null  float64
 31  credit_classs_5      330068 non-null  float64
 32  credit_classs_6      330068 non-null  float64
 33  credit_classs_7      330068 non-null  float64
 34  credit_classs_8      330068 non-null  float64
 35  credit_classs_9      330068 non-null  float64
 36  credit_classs_10     330068 non-null  float64
 37  credit_classs_11     330068 non-null  float64
 38  age_u30              330068 non-null  int32
 39  age_30s              330068 non-null  int32
 40  age_40s              330068 non-null  int32
 41  age_50s              330068 non-null  int32
 42  age_60s              330068 non-null  int32
 43  age_70s              330068 non-null  int32
 44  age_o80              330068 non-null  int32
dtypes: float64(36), int32(7), int64(2)
memory usage: 107.0 MB
```

In [178]:
```
from sklearn import tree
```

In [179]:
```
dt = preprocessed
dt.drop(columns=['CreditScore_log', 'Age_log', 'EstimatedSalary_log'], inplace=
```

In [180]:
```
#split dataset in features and target variable
feature_cols = dt.drop(columns=["Exited"], axis=1)
X = dt.drop("Exited", axis=1) # Features
y = dt.Exited # Target variable
```

In [181]:
```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random
```

In [182]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

tree_reg = tree.DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X_train, y_train)

extracted_MSEs = tree_reg.tree_.impurity # The Hidden magic is HERE

for idx, MSE in enumerate(tree_reg.tree_.impurity):
    print("Train Node {} has MSE {}".format(idx,MSE))
```

```
Train Node 0 has MSE 0.16645669124835372
Train Node 1 has MSE 0.10689615937013942
Train Node 2 has MSE 0.17797554409374527
Train Node 3 has MSE 0.035531290261938624
Train Node 4 has MSE 0.24974411467644997
Train Node 5 has MSE 0.2271340164912764
Train Node 6 has MSE 0.1440532552302116
```

In [183]:
```
tree_reg = tree.DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X_test, y_test)

extracted_MSEs = tree_reg.tree_.impurity # The Hidden magic is HERE

for idx, MSE in enumerate(tree_reg.tree_.impurity):
    print("Test Node {} has MSE {}".format(idx,MSE))
```

```
Test Node 0 has MSE 0.16768042147336473
Test Node 1 has MSE 0.10784071856335926
Test Node 2 has MSE 0.17904413964719332
Test Node 3 has MSE 0.03493071955922489
Test Node 4 has MSE 0.24983017286977013
Test Node 5 has MSE 0.22741909019510592
Test Node 6 has MSE 0.1491924516548918
```

In [184]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9386089819331253
```

In [185]:
```
clf.feature_importances_
```

Out[185]:
```
array([1.12490505e-01, 1.45729880e-02, 4.36976645e-02, 4.25152385e-04,
       1.23325224e-01, 2.17823977e-01, 1.64382996e-01, 1.52549303e-02,
       1.24173339e-02, 1.19735484e-02, 1.02855856e-02, 1.02193734e-02,
       9.55992083e-03, 1.05429869e-02, 1.02854556e-02, 9.60379830e-03,
       9.42548767e-03, 1.00417703e-02, 1.06870238e-02, 5.93315419e-03,
       1.18893509e-01, 2.18528220e-02, 2.79259842e-03, 1.27101224e-04,
       4.64159072e-04, 1.40972449e-03, 3.10226818e-03, 4.51821994e-03,
       6.59320699e-03, 7.01958900e-03, 5.93564121e-03, 3.12069261e-03,
       2.39845341e-03, 8.33099417e-04, 1.43510231e-03, 2.40039898e-03,
       1.97543690e-03, 1.51527243e-03, 5.31650209e-04, 1.14831671e-04,
       1.73477904e-05])
```

In [186]:
```
X_train.columns
```

Out[186]:
```
Index(['Balance', 'HasCrCard', 'IsActiveMember', 'HasBalance', 'CreditScore',
       'Age', 'EstimatedSalary', 'Geography_Germany', 'Geography_Spain',
       'Gender_Male', 'Tenure_1', 'Tenure_2', 'Tenure_3', 'Tenure_4',
       'Tenure_5', 'Tenure_6', 'Tenure_7', 'Tenure_8', 'Tenure_9', 'Tenure_1
0',
       'NumOfProducts_2', 'NumOfProducts_3', 'NumOfProducts_4',
       'credit_classs_1', 'credit_classs_2', 'credit_classs_3',
       'credit_classs_4', 'credit_classs_5', 'credit_classs_6',
       'credit_classs_7', 'credit_classs_8', 'credit_classs_9',
       'credit_classs_10', 'credit_classs_11', 'age_u30', 'age_30s', 'age_40
s',
       'age_50s', 'age_60s', 'age_70s', 'age_o80'],
      dtype='object')
```

In [187]:
```
# think this is overfitting, need to adjust to remove doubling of variable and
```

In [188]:
```
# 1. remove age groups and rerun
dt1 = dt.drop(columns=['age_u30', 'age_30s', 'age_40s',
       'age_50s', 'age_60s', 'age_70s', 'age_o80'])
```

In [189]:
```
#split dataset in features and target variable
feature_cols = dt1.drop(columns=["Exited"], axis=1)
X = dt.drop("Exited", axis=1) # Features
y = dt.Exited # Target variable
```

In [190]:
```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random
```

In [191]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

tree_reg = tree.DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X_train, y_train)

extracted_MSEs = tree_reg.tree_.impurity

for idx, MSE in enumerate(tree_reg.tree_.impurity):
    print("Train Node {} has MSE {}".format(idx,MSE))
```

```
Train Node 0 has MSE 0.16645669124835372
Train Node 1 has MSE 0.10689615937013942
Train Node 2 has MSE 0.17797554409374527
Train Node 3 has MSE 0.035531290261938624
Train Node 4 has MSE 0.24974411467644997
Train Node 5 has MSE 0.2271340164912764
Train Node 6 has MSE 0.1440532552302116
```

In [192]:
```
tree_reg = tree.DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X_test, y_test)

extracted_MSEs = tree_reg.tree_.impurity

for idx, MSE in enumerate(tree_reg.tree_.impurity):
    print("Test Node {} has MSE {}".format(idx,MSE))
```

```
Test Node 0 has MSE 0.16768042147336473
Test Node 1 has MSE 0.10784071856335926
Test Node 2 has MSE 0.17904413964719332
Test Node 3 has MSE 0.03493071955922489
Test Node 4 has MSE 0.24983017286977013
Test Node 5 has MSE 0.22741909019510592
Test Node 6 has MSE 0.1491924516548918
```

In [193]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9384675977822886
```

In [194]:
```
clf.feature_importances_
```

Out[194]:
```
array([1.12013193e-01, 1.44573929e-02, 4.36976645e-02, 4.87727957e-04,
       1.24447044e-01, 2.17876159e-01, 1.64249971e-01, 1.52455751e-02,
       1.21552073e-02, 1.21591613e-02, 1.06727362e-02, 1.00455731e-02,
       9.45989079e-03, 1.09093086e-02, 1.08380483e-02, 9.90056739e-03,
       9.61850931e-03, 1.06552685e-02, 1.04461763e-02, 5.95657426e-03,
       1.18893509e-01, 2.16953019e-02, 2.80440790e-03, 8.66230458e-05,
       3.19825967e-04, 1.21923273e-03, 2.76604454e-03, 4.59115900e-03,
       5.87832027e-03, 6.03866379e-03, 5.85449259e-03, 3.47980666e-03,
       2.30436879e-03, 7.38736300e-04, 1.20500829e-03, 2.44530012e-03,
       2.48223840e-03, 1.22410049e-03, 5.63742799e-04, 1.10039416e-04,
       7.32969508e-06])
```

In [195]:
```
X_train.columns
```

Out[195]:
```
Index(['Balance', 'HasCrCard', 'IsActiveMember', 'HasBalance', 'CreditScore',
       'Age', 'EstimatedSalary', 'Geography_Germany', 'Geography_Spain',
       'Gender_Male', 'Tenure_1', 'Tenure_2', 'Tenure_3', 'Tenure_4',
       'Tenure_5', 'Tenure_6', 'Tenure_7', 'Tenure_8', 'Tenure_9', 'Tenure_1
0',
       'NumOfProducts_2', 'NumOfProducts_3', 'NumOfProducts_4',
       'credit_classs_1', 'credit_classs_2', 'credit_classs_3',
       'credit_classs_4', 'credit_classs_5', 'credit_classs_6',
       'credit_classs_7', 'credit_classs_8', 'credit_classs_9',
       'credit_classs_10', 'credit_classs_11', 'age_u30', 'age_30s', 'age_40
s',
       'age_50s', 'age_60s', 'age_70s', 'age_o80'],
      dtype='object')
```

In [196]:
```
# rerun with age groups
dt2 = dt.drop(columns=['Age'])
```

In [197]:
```
#split dataset in features and target variable
feature_cols = dt2.drop(columns=["Exited"], axis=1)
X = dt2.drop("Exited", axis=1) # Features
y = dt2.Exited # Target variable
```

In [198]:
```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random
```

In [199]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

tree_reg = tree.DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X_train, y_train)

extracted_MSEs = tree_reg.tree_.impurity

for idx, MSE in enumerate(tree_reg.tree_.impurity):
    print("Node {} has MSE {}".format(idx,MSE))
```

```
Node 0 has MSE 0.16645669124835372
Node 1 has MSE 0.2328628605609285
Node 2 has MSE 0.24991759794706286
Node 3 has MSE 0.16547962027042945
Node 4 has MSE 0.05651116449297758
Node 5 has MSE 0.03488900418768863
Node 6 has MSE 0.11474130910105146
```

In [200]:
```
tree_reg = tree.DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X_test, y_test)

extracted_MSEs = tree_reg.tree_.impurity

for idx, MSE in enumerate(tree_reg.tree_.impurity):
    print("Test Node {} has MSE {}".format(idx,MSE))
```

```
Test Node 0 has MSE 0.16768042147336473
Test Node 1 has MSE 0.23314642525100326
Test Node 2 has MSE 0.24989320535404766
Test Node 3 has MSE 0.1667910016027782
Test Node 4 has MSE 0.05738032188012889
Test Node 5 has MSE 0.036088250704495645
Test Node 6 has MSE 0.11510486031375416
```

In [201]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9357712000484746
```

In [202]:
```
clf.feature_importances_
```

Out[202]:
```
array([1.26492613e-01, 1.56575262e-02, 4.64143588e-02, 4.03564194e-04,
       1.37546509e-01, 1.95325555e-01, 1.63366027e-02, 1.42808765e-02,
       1.27025237e-02, 1.10196634e-02, 1.16938325e-02, 9.88922874e-03,
       1.08140772e-02, 1.05971602e-02, 1.11385481e-02, 1.06743025e-02,
       1.08398675e-02, 1.13915698e-02, 5.98095588e-03, 1.43570231e-01,
       2.08796967e-02, 3.40095143e-03, 6.59284897e-05, 3.37476774e-04,
       1.25621921e-03, 2.95686275e-03, 4.04482880e-03, 6.03446868e-03,
       6.11849594e-03, 4.72501560e-03, 3.11259750e-03, 1.68477573e-03,
       1.24657113e-03, 4.85860720e-02, 5.81621947e-02, 1.27147108e-02,
       8.25017931e-03, 2.61490795e-03, 9.47133101e-04, 9.13473943e-05])
```

In [203]:
```
X_train.columns
```

Out[203]:
```
Index(['Balance', 'HasCrCard', 'IsActiveMember', 'HasBalance', 'CreditScore',
       'EstimatedSalary', 'Geography_Germany', 'Geography_Spain',
       'Gender_Male', 'Tenure_1', 'Tenure_2', 'Tenure_3', 'Tenure_4',
       'Tenure_5', 'Tenure_6', 'Tenure_7', 'Tenure_8', 'Tenure_9', 'Tenure_1
0',
       'NumOfProducts_2', 'NumOfProducts_3', 'NumOfProducts_4',
       'credit_classs_1', 'credit_classs_2', 'credit_classs_3',
       'credit_classs_4', 'credit_classs_5', 'credit_classs_6',
       'credit_classs_7', 'credit_classs_8', 'credit_classs_9',
       'credit_classs_10', 'credit_classs_11', 'age_u30', 'age_30s', 'age_40
s',
       'age_50s', 'age_60s', 'age_70s', 'age_o80'],
      dtype='object')
```

In [204]:
```
# Drop age groups
dt3 = dt.drop(columns=['age_u30', 'age_30s', 'age_40s',
       'age_50s', 'age_60s', 'age_70s', 'age_o80'])
```

In [205]:
```
#split dataset in features and target variable
feature_cols = dt3.drop(columns=["Exited"], axis=1)
X = dt3.drop("Exited", axis=1) # Features
y = dt3.Exited # Target variable
```

In [206]:
```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random
```

In [207]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

tree_reg = tree.DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X_train, y_train)

extracted_MSEs = tree_reg.tree_.impurity

for idx, MSE in enumerate(tree_reg.tree_.impurity):
    print("Node {} has MSE {}".format(idx,MSE))
```

```
Node 0 has MSE 0.16645669124835372
Node 1 has MSE 0.10689615937013942
Node 2 has MSE 0.17797554409374527
Node 3 has MSE 0.035531290261938624
Node 4 has MSE 0.24974411467644997
Node 5 has MSE 0.2271340164912764
Node 6 has MSE 0.1440532552302116
```

In [208]:
```
tree_reg = tree.DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X_test, y_test)

extracted_MSEs = tree_reg.tree_.impurity

for idx, MSE in enumerate(tree_reg.tree_.impurity):
    print("Test Node {} has MSE {}".format(idx,MSE))
```

```
Test Node 0 has MSE 0.16768042147336473
Test Node 1 has MSE 0.10784071856335926
Test Node 2 has MSE 0.17904413964719332
Test Node 3 has MSE 0.03493071955922489
Test Node 4 has MSE 0.24983017286977013
Test Node 5 has MSE 0.22741909019510592
Test Node 6 has MSE 0.1491924516548918
```

In [209]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9380838408014461
```

In [210]:
```
clf.feature_importances_
```

Out[210]:
```
array([1.13762480e-01, 1.44619915e-02, 4.36976645e-02, 4.01623261e-04,
       1.25694102e-01, 2.23582997e-01, 1.64086541e-01, 1.50017323e-02,
       1.27013841e-02, 1.23149862e-02, 1.08208990e-02, 1.04861031e-02,
       9.39635819e-03, 1.06888807e-02, 1.02752304e-02, 9.61040804e-03,
       9.46508381e-03, 1.01798346e-02, 1.01782993e-02, 5.70473834e-03,
       1.18893509e-01, 2.17217278e-02, 2.78472835e-03, 8.66230458e-05,
       3.40601089e-04, 9.53311444e-04, 3.11901848e-03, 4.81863811e-03,
       5.80414218e-03, 6.02633469e-03, 5.84631631e-03, 3.66395857e-03,
       2.54358610e-03, 8.86168099e-04])
```

In [211]:
```
X_train.columns
```

Out[211]:
```
Index(['Balance', 'HasCrCard', 'IsActiveMember', 'HasBalance', 'CreditScore',
       'Age', 'EstimatedSalary', 'Geography_Germany', 'Geography_Spain',
       'Gender_Male', 'Tenure_1', 'Tenure_2', 'Tenure_3', 'Tenure_4',
       'Tenure_5', 'Tenure_6', 'Tenure_7', 'Tenure_8', 'Tenure_9', 'Tenure_1
0',
       'NumOfProducts_2', 'NumOfProducts_3', 'NumOfProducts_4',
       'credit_classs_1', 'credit_classs_2', 'credit_classs_3',
       'credit_classs_4', 'credit_classs_5', 'credit_classs_6',
       'credit_classs_7', 'credit_classs_8', 'credit_classs_9',
       'credit_classs_10', 'credit_classs_11'],
      dtype='object')
```

In [212]:
```
# Using the continuous Age feature the model has a slightly better performance,
# doubling up with both the continuous and the grouping is not needed and using
# lets check with the credit score as well
```

In [213]:
```
# Drop creditscore
dt4 = dt3.drop(columns=['CreditScore'])
```

In [214]:
```
#split dataset in features and target variable
feature_cols = dt4.drop(columns=["Exited"], axis=1)
X = dt4.drop("Exited", axis=1) # Features
y = dt4.Exited # Target variable
```

In [215]:
```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random
```

In [216]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

tree_reg = tree.DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X_train, y_train)

extracted_MSEs = tree_reg.tree_.impurity

for idx, MSE in enumerate(tree_reg.tree_.impurity):
    print("Node {} has MSE {}".format(idx,MSE))
```

```
Node 0 has MSE 0.16645669124835372
Node 1 has MSE 0.10689615937013942
Node 2 has MSE 0.17797554409374527
Node 3 has MSE 0.035531290261938624
Node 4 has MSE 0.24974411467644997
Node 5 has MSE 0.2271340164912764
Node 6 has MSE 0.1440532552302116
```

In [217]:
```
tree_reg = tree.DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X_test, y_test)

extracted_MSEs = tree_reg.tree_.impurity

for idx, MSE in enumerate(tree_reg.tree_.impurity):
    print("Test Node {} has MSE {}".format(idx,MSE))
```

```
Test Node 0 has MSE 0.16768042147336473
Test Node 1 has MSE 0.10784071856335926
Test Node 2 has MSE 0.17904413964719332
Test Node 3 has MSE 0.03493071955922489
Test Node 4 has MSE 0.24983017286977013
Test Node 5 has MSE 0.22741909019510592
Test Node 6 has MSE 0.1491924516548918
```

In [218]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.939406792498561
```

In [219]:
```
clf.feature_importances_
```

Out[219]:
```
array([1.29902596e-01, 1.70806139e-02, 4.37333223e-02, 5.26830369e-04,
       2.29596875e-01, 2.07407799e-01, 1.54069666e-02, 1.50301045e-02,
       1.24287625e-02, 1.11464418e-02, 1.13726186e-02, 1.03981512e-02,
       1.17347022e-02, 1.08954949e-02, 9.70217206e-03, 1.03047572e-02,
       1.07336960e-02, 1.01346490e-02, 6.36639262e-03, 1.18990527e-01,
       2.19187671e-02, 2.96283395e-03, 2.14633349e-04, 1.75228371e-03,
       4.84603625e-03, 8.36865259e-03, 1.19986202e-02, 1.23642914e-02,
       1.30195435e-02, 1.26035621e-02, 8.73691735e-03, 4.95709887e-03,
       3.36328665e-03])
```

In [220]:
```
X_train.columns
```

Out[220]:
```
Index(['Balance', 'HasCrCard', 'IsActiveMember', 'HasBalance', 'Age',
       'EstimatedSalary', 'Geography_Germany', 'Geography_Spain',
       'Gender_Male', 'Tenure_1', 'Tenure_2', 'Tenure_3', 'Tenure_4',
       'Tenure_5', 'Tenure_6', 'Tenure_7', 'Tenure_8', 'Tenure_9', 'Tenure_1
0',
       'NumOfProducts_2', 'NumOfProducts_3', 'NumOfProducts_4',
       'credit_classs_1', 'credit_classs_2', 'credit_classs_3',
       'credit_classs_4', 'credit_classs_5', 'credit_classs_6',
       'credit_classs_7', 'credit_classs_8', 'credit_classs_9',
       'credit_classs_10', 'credit_classs_11'],
      dtype='object')
```

In [221]:
```
# Drop creditscore
dt5 = dt3.drop(columns=['credit_classs_1', 'credit_classs_2', 'credit_classs_3'
       'credit_classs_4', 'credit_classs_5', 'credit_classs_6',
       'credit_classs_7', 'credit_classs_8', 'credit_classs_9',
       'credit_classs_10', 'credit_classs_11'])
```

In [222]:
```
#split dataset in features and target variable
feature_cols = dt5.drop(columns=["Exited"], axis=1)
X = dt5.drop("Exited", axis=1) # Features
y = dt5.Exited # Target variable
```

In [223]:
```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random
```

In [224]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(max_depth=4, min_samples_leaf=6 )

# Train Decision Tree Classifer
model_tree = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

tree_reg = tree.DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X_train, y_train)

extracted_MSEs = tree_reg.tree_.impurity

for idx, MSE in enumerate(tree_reg.tree_.impurity):
    print("Node {} has MSE {}".format(idx,MSE))
```

```
Node 0 has MSE 0.16645669124835372
Node 1 has MSE 0.10689615937013942
Node 2 has MSE 0.17797554409374527
Node 3 has MSE 0.035531290261938624
Node 4 has MSE 0.24974411467644997
Node 5 has MSE 0.2271340164912764
Node 6 has MSE 0.1440532552302116
```

In [225]:
```python
text_representation = tree.export_text(clf)
print(text_representation)
```

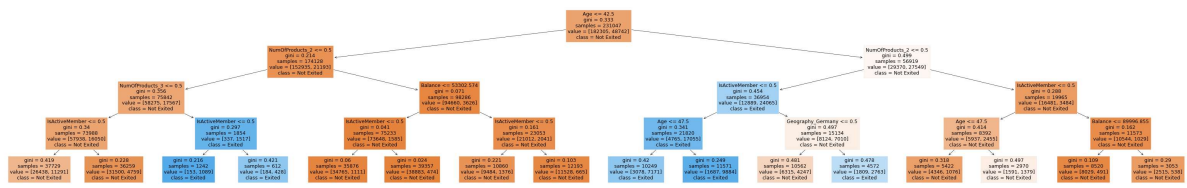
```
|--- feature_5 <= 42.50
|   |--- feature_20 <= 0.50
|   |   |--- feature_21 <= 0.50
|   |   |   |--- feature_2 <= 0.50
|   |   |   |   |--- class: 0
|   |   |   |--- feature_2 >  0.50
|   |   |   |   |--- class: 0
|   |   |--- feature_21 >  0.50
|   |   |   |--- feature_2 <= 0.50
|   |   |   |   |--- class: 1
|   |   |   |--- feature_2 >  0.50
|   |   |   |   |--- class: 1
|   |--- feature_20 >  0.50
|   |   |--- feature_0 <= 53302.57
|   |   |   |--- feature_2 <= 0.50
|   |   |   |   |--- class: 0
|   |   |   |--- feature_2 >  0.50
|   |   |   |   |--- class: 0
|   |   |--- feature_0 >  53302.57
|   |   |   |--- feature_2 <= 0.50
|   |   |   |   |--- class: 0
|   |   |   |--- feature_2 >  0.50
|   |   |   |   |--- class: 0
|--- feature_5 >  42.50
|   |--- feature_20 <= 0.50
|   |   |--- feature_2 <= 0.50
|   |   |   |--- feature_5 <= 47.50
|   |   |   |   |--- class: 1
|   |   |   |--- feature_5 >  47.50
|   |   |   |   |--- class: 1
|   |   |--- feature_2 >  0.50
|   |   |   |--- feature_7 <= 0.50
|   |   |   |   |--- class: 0
|   |   |   |--- feature_7 >  0.50
|   |   |   |   |--- class: 1
|   |--- feature_20 >  0.50
|   |   |--- feature_2 <= 0.50
|   |   |   |--- feature_5 <= 47.50
|   |   |   |   |--- class: 0
|   |   |   |--- feature_5 >  47.50
|   |   |   |   |--- class: 0
|   |   |--- feature_2 >  0.50
|   |   |   |--- feature_0 <= 89996.86
|   |   |   |   |--- class: 0
|   |   |   |--- feature_0 >  89996.86
|   |   |   |   |--- class: 0
```

In [226]:
```
tree_reg = tree.DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X_test, y_test)

extracted_MSEs = tree_reg.tree_.impurity

for idx, MSE in enumerate(tree_reg.tree_.impurity):
    print("Test Node {} has MSE {}".format(idx,MSE))
```

```
Test Node 0 has MSE 0.16768042147336473
Test Node 1 has MSE 0.10784071856335926
Test Node 2 has MSE 0.17904413964719332
Test Node 3 has MSE 0.03493071955922489
Test Node 4 has MSE 0.24983017286977013
Test Node 5 has MSE 0.22741909019510592
Test Node 6 has MSE 0.1491924516548918
```

In [227]:
```
fn=dt5.drop(columns='Exited',axis=1).columns.to_list()
```

In [228]:
```
plt.figure(figsize=(40, 6))
tree.plot_tree(model_tree, filled=True, feature_names=fn, class_names=['Not Exi
plt.show()
```



In [232]:
```
viz = dtreeviz(model_tree, X_train, y_train, target_name="Exited", feature_name
viz.view()
```

```
C:\Users\p_ste\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning:
X does not have valid feature names, but DecisionTreeClassifier was fitted wi
th feature names
  warnings.warn(
```

In [233]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8491633087930843
```

In [234]:
```
clf.feature_importances_
```

Out[234]:
```
array([0.00858328, 0.        , 0.12877309, 0.        , 0.        ,
       0.45212077, 0.        , 0.01000661, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.3503685 , 0.05014775, 0.        ])
```

In [235]:
```
X_train.columns
```

Out[235]:
```
Index(['Balance', 'HasCrCard', 'IsActiveMember', 'HasBalance', 'CreditScore',
       'Age', 'EstimatedSalary', 'Geography_Germany', 'Geography_Spain',
       'Gender_Male', 'Tenure_1', 'Tenure_2', 'Tenure_3', 'Tenure_4',
       'Tenure_5', 'Tenure_6', 'Tenure_7', 'Tenure_8', 'Tenure_9', 'Tenure_1
0',
       'NumOfProducts_2', 'NumOfProducts_3', 'NumOfProducts_4'],
      dtype='object')
```