```
!pip install pandas
```

```
Requirement already satisfied: pandas in e:\rasa\lib\site-packages
(2.1.4)
Requirement already satisfied: numpy<2,>=1.23.2 in e:\rasa\lib\site-
packages (from pandas) (1.24.3)
Requirement already satisfied: python-dateutil>=2.8.2 in e:\rasa\lib\
site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in e:\rasa\lib\site-
packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in e:\rasa\lib\site-
packages (from pandas) (2023.3)
Requirement already satisfied: six>=1.5 in e:\rasa\lib\site-packages
(from python-dateutil>=2.8.2->pandas) (1.16.0)
```

FINTECH PROJECT- Loosing Bank customers(Churn Analysis)

Observation of data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Importing dataset
data = pd.read_csv('G:/dsml-scaler/fintech_project/Bank-Records.csv')
data.head()
```

```
   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age
\
0          1    15634602  Hargrave          619    France  Female   42

1          2    15647311      Hill          608     Spain  Female   41

2          3    15619304      Onio          502    France  Female   42

3          4    15701354      Boni          699    France  Female   39

4          5    15737888  Mitchell          850     Spain  Female   43


   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited  Complain  Satisfaction Score Card Type  \
0        101348.88       1         1                   2   DIAMOND
```

```
1        112542.58       0         1                    3   DIAMOND
2        113931.57       1         1                    3   DIAMOND
3         93826.63       0         0                    5      GOLD
4         79084.10       0         0                    5      GOLD

   Point Earned
0           464
1           456
2           377
3           350
4           425
```

```
data = data.drop('RowNumber', axis = 1)
```

```
data.head()
```

```
   CustomerId   Surname  CreditScore Geography  Gender  Age  Tenure  \
0    15634602  Hargrave          619    France  Female   42       2
1    15647311      Hill          608     Spain  Female   41       1
2    15619304      Onio          502    France  Female   42       8
3    15701354      Boni          699    France  Female   39       1
4    15737888  Mitchell          850     Spain  Female   43       2

     Balance  NumOfProducts  HasCrCard  IsActiveMember
EstimatedSalary  \
0       0.00              1          1               1
101348.88
1   83807.86              1          0               1
112542.58
2  159660.80              3          1               0
113931.57
3       0.00              2          0               0
93826.63
4  125510.82              1          1               1
79084.10

   Exited  Complain  Satisfaction Score Card Type  Point Earned
0       1         1                   2   DIAMOND           464
1       0         1                   3   DIAMOND           456
2       1         1                   3   DIAMOND           377
3       0         0                   5      GOLD           350
4       0         0                   5      GOLD           425
```

```
data.shape
```

```
(10000, 17)
```

```
data.isna().sum()
```

```
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
Complain              0
Satisfaction Score    0
Card Type             0
Point Earned          0
dtype: int64
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   CustomerId          10000 non-null  int64  
 1   Surname             10000 non-null  object 
 2   CreditScore         10000 non-null  int64  
 3   Geography           10000 non-null  object 
 4   Gender              10000 non-null  object 
 5   Age                 10000 non-null  int64  
 6   Tenure              10000 non-null  int64  
 7   Balance             10000 non-null  float64
 8   NumOfProducts       10000 non-null  int64  
 9   HasCrCard           10000 non-null  int64  
 10  IsActiveMember      10000 non-null  int64  
 11  EstimatedSalary     10000 non-null  float64
 12  Exited              10000 non-null  int64  
 13  Complain            10000 non-null  int64  
 14  Satisfaction Score  10000 non-null  int64  
 15  Card Type           10000 non-null  object 
 16  Point Earned        10000 non-null  int64  
dtypes: float64(2), int64(11), object(4)
memory usage: 1.3+ MB
```

```
data.describe()
```

```
         CustomerId   CreditScore           Age        Tenure
Balance  \
count  1.000000e+04  10000.000000  10000.000000  10000.000000
10000.000000
```

```
mean   1.569094e+07    650.528800     38.921800      5.012800
76485.889288
std    7.193619e+04     96.653299     10.487806      2.892174
62397.405202
min    1.556570e+07    350.000000     18.000000      0.000000
0.000000
25%    1.562853e+07    584.000000     32.000000      3.000000
0.000000
50%    1.569074e+07    652.000000     37.000000      5.000000
97198.540000
75%    1.575323e+07    718.000000     44.000000      7.000000
127644.240000
max    1.581569e+07    850.000000     92.000000     10.000000
250898.090000

       NumOfProducts    HasCrCard  IsActiveMember  EstimatedSalary  \
count   10000.000000  10000.00000    10000.000000     10000.000000
mean        1.530200      0.70550        0.515100    100090.239881
std         0.581654      0.45584        0.499797     57510.492818
min         1.000000      0.00000        0.000000        11.580000
25%         1.000000      0.00000        0.000000     51002.110000
50%         1.000000      1.00000        1.000000    100193.915000
75%         2.000000      1.00000        1.000000    149388.247500
max         4.000000      1.00000        1.000000    199992.480000

              Exited      Complain  Satisfaction Score  Point Earned
count  10000.000000  10000.000000        10000.000000  10000.000000
mean       0.203800      0.204400            3.013800    606.515100
std        0.402842      0.403283            1.405919    225.924839
min        0.000000      0.000000            1.000000    119.000000
25%        0.000000      0.000000            2.000000    410.000000
50%        0.000000      0.000000            3.000000    605.000000
75%        0.000000      0.000000            4.000000    801.000000
max        1.000000      1.000000            5.000000   1000.000000
```

```
data.describe(include = 'object')
```

```
       Surname Geography Gender Card Type
count    10000     10000  10000     10000
unique    2932         3      2         4
top      Smith    France   Male   DIAMOND
freq        32      5014   5457      2507
```

```
data.isna().sum()
```

```
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
Complain              0
Satisfaction Score    0
Card Type             0
Point Earned          0
dtype: int64
```

Non Graphical Analysis

```
# Check for duplicate rows
duplicates = data.duplicated()

# Count the number of duplicate rows
num_duplicates = duplicates.sum()
print(f"\nNumber of duplicate rows: {num_duplicates}")
```

```
Number of duplicate rows: 0
```

```
# Categorical columns in the dataset
categorical_columns = ['Geography', 'Gender', 'Card Type']

# Display unique values and their counts for each categorical column
for column in categorical_columns:
    unique_values = data[column].value_counts()
    print(f"Unique values in '{column}':")
    print(unique_values)
    print()
```

```
Unique values in 'Geography':
Geography
France     5014
Germany    2509
Spain      2477
Name: count, dtype: int64

Unique values in 'Gender':
Gender
Male      5457
Female    4543
Name: count, dtype: int64

Unique values in 'Card Type':
```

```
Card Type
DIAMOND     2507
GOLD        2502
SILVER      2496
PLATINUM    2495
Name: count, dtype: int64
```

```
#identify numerical columns
numerical_columns = data.select_dtypes(include=['int64',
'float64']).columns

# Count the number of unique values in each numerical column
unique_counts_numerical = data[numerical_columns].nunique()

# Display the counts of unique values
print("Count of unique values in numerical columns:")
print(unique_counts_numerical)

# Optionally, inspect the actual unique values for each numerical
column
for column in numerical_columns:
    unique_values = data[column].unique()
    print(f"\nUnique values in '{column}':")
    print(unique_values)
```

```
Count of unique values in numerical columns:
CustomerId            10000
CreditScore             460
Age                      70
Tenure                   11
Balance                6382
NumOfProducts             4
HasCrCard                 2
IsActiveMember            2
EstimatedSalary        9999
Exited                    2
Complain                  2
Satisfaction Score        5
Point Earned            785
dtype: int64

Unique values in 'CustomerId':
[15634602 15647311 15619304 ... 15584532 15682355 15628319]

Unique values in 'CreditScore':
[619 608 502 699 850 645 822 376 501 684 528 497 476 549 635 616 653
587
 726 732 636 510 669 846 577 756 571 574 411 591 533 553 520 722 475
490
```

```
 804 582 472 465 556 834 660 776 829 637 550 698 585 788 655 601 656
725
 511 614 742 687 555 603 751 581 735 661 675 738 813 657 604 519 664
678
 757 416 665 777 543 506 493 652 750 729 646 647 808 524 769 730 515
773
 814 710 413 623 670 622 785 605 479 685 538 562 721 628 668 828 674
625
 432 770 758 795 686 789 589 461 584 579 663 682 793 691 485 650 754
535
 716 539 706 586 631 717 800 683 704 615 667 484 480 578 512 606 597
778
 514 525 715 580 807 521 759 516 711 618 643 671 689 620 676 572 695
592
 567 694 547 594 673 610 767 763 712 703 662 659 523 772 545 634 739
771
 681 544 696 766 727 693 557 531 498 651 791 733 811 707 714 782 775
799
 602 744 588 747 583 627 731 629 438 642 806 474 559 429 680 749 734
644
 626 649 805 718 840 630 654 762 568 613 522 737 648 443 640 540 460
593
 801 611 802 745 483 690 492 709 705 560 752 701 537 487 596 702 486
724
 548 464 790 534 748 494 590 468 509 818 816 536 753 774 621 569 658
798
 641 542 692 639 765 570 638 599 632 779 527 564 833 504 842 508 417
598
 741 607 761 848 546 439 755 760 526 713 700 666 566 495 688 612 477
427
 839 819 720 459 503 624 529 563 482 796 445 746 786 554 672 787 499
844
 450 815 838 803 736 633 600 679 517 792 743 488 421 841 708 507 505
456
 435 561 518 565 728 784 552 609 764 697 723 551 444 719 496 541 830
812
 677 420 595 617 809 500 826 434 513 478 797 363 399 463 780 452 575
837
 794 824 428 823 781 849 489 431 457 768 831 359 820 573 576 558 817
449
 440 415 821 530 350 446 425 740 481 783 358 845 451 458 469 423 404
836
 473 835 466 491 351 827 843 365 532 414 453 471 401 810 832 470 447
422
 825 430 436 426 408 847 418 437 410 454 407 455 462 386 405 383 395
467
 433 442 424 448 441 367 412 382 373 419]

Unique values in 'Age':
```

```
[42 41 39 43 44 50 29 27 31 24 34 25 35 45 58 32 38 46 36 33 40 51 61
49
 37 19 66 56 26 21 55 75 22 30 28 65 48 52 57 73 47 54 72 20 67 79 62
53
 80 59 68 23 60 70 63 64 18 82 69 74 71 76 77 88 85 84 78 81 92 83]

Unique values in 'Tenure':
[ 2  1  8  7  4  6  3 10  5  9  0]

Unique values in 'Balance':
[     0.    83807.86 159660.8  ...  57369.61  75075.31 130142.79]

Unique values in 'NumOfProducts':
[1 3 2 4]

Unique values in 'HasCrCard':
[1 0]

Unique values in 'IsActiveMember':
[1 0]

Unique values in 'EstimatedSalary':
[101348.88 112542.58 113931.57 ...  42085.58  92888.52  38190.78]

Unique values in 'Exited':
[1 0]

Unique values in 'Complain':
[1 0]

Unique values in 'Satisfaction Score':
[2 3 5 4 1]

Unique values in 'Point Earned':
[ 464  456  377  350  425  484  206  282  251  342  264  249  119  549
  318  308  163  544  732  477  568  336  887  665  225  659  236  448
  499  343  298  628  611  922  276  877  471  994  944  419  545  828
  789  567  884  296  779  730  960  268  829  982  657  221  406  954
  823  769  481  903  882  379  469  683  770  985  725  259  979  988
  833  527  220  983  582  281  798  917  255  511  498  651  852  807
  470  752  559  375  411  722  231  488  965  289  449  520  234  458
  915  299  838  786  509  328  347  534  709  749  541  736  529  594
  462  429  370  510  638  718  437  485  410  666  595  578  831  497
  374  814  247  912  883  402  764  608  652  800  697  427  620  701
  953  635  250  662  699  855  388  612  801  561  643  818  775  872
  435  415  430  929  681  810  739  280  991  482  423  260  642  570
  790  727  909  799  861  512  478  615  321  726  516  639  244  292
  245  848  974  958  629  310  621  924  359  811  583  959  901  729
  756  873  865  690  535  580  930  943  248  763  619  815  566  774
  695  902  532  895  333  866  671  670  768  565  483  926  932  899
```

```
 467  717  716  650  693  759  946  465  600  306  747  424  913  751
 219  587  278  526  632  301  806  563  669  517  673  809  460  777
 837  737  486  252  275  361  352  337  597  381  668  495  378  731
 230  919  641  457  849  720  317  269  391  373  237  645  624  224
 723  973  463  253  241  537  332  390  240  978  366  356  907  687
 703  579  793  291  951  952  616  706  552  351  906  886  431  362
 309  513  302  858  543  939  797  743  767  246  480  708  827  468
 523  560  530  367  393  860  576  396  874  680  319  781  918  340
 625  923  859  794  702  816  404  977  426  368  261  614  765  961
 345  330  863  371  937  956  496  927  550  420  760  500  850  678
 547  238  314  257  540  331  813  279  685  898  466  757  453  694
 304  633  989  936  689  267  613  585  452  358  771  344  405  408
 503  881  758  363  969  403  980  283  354  938  753  742  844  776
 812  647  839  847  934  656  808  525  746  258  817  418  380  955
 274  455  451  603  660  577  824  445  589  908  395  878  778  229
 834  265  819  992  444  325  691  921  335  661  562  888  300  869
 910  515  592  475  307  766  564  677  288  518  653  606  792  326
 522  748  602  459  588  494  256  664  821  287  623  963  705  519
 773  854  254  935  914  539  277  735  369  740  461  329  970  925
 446  514  711  804  987  348  686  341  692  417  506  242  323  719
 596  372  599  558  796  421  738  780  454  365  654  933  573  721
 434  610  450  802  631  805  476  897  273  433  487  233  803  975
 397  948  684  715  400  263  667  997  581  795  413  788  885  297
 553  528  835  479  443  676  893  409  324  822  447  867  473  830
 542  383  508  376  875  644  704  928  504  312  784  672  846  945
 728  981  609  360  555  707  876  783  436  663  714  505  439  894
 320  223  962  836  507  655  696  266  845  950  322  841  649  636
 490  474  546  295  842  357  791  840  679  892  900  990  942  416
 826  949  744  243  688  398  916  572 1000  574  591  674  353  870
 940  993  843  998  472  905  710  533  785  334  598  327  232  586
 226  305  984  634  593  389  538  995  618  724  584  889  712  303
 733  920  761  285  964  311  911  501  851  972  440  346  239  750
 493  284  741  601  338  658  871  931  999  271  502  700  536  626
 957  428  966  968  313  227  637  856  432  698  896  986  412  385
 976  590  524  407  825  904  575  734  548  489  868  879  675  648
 853  640  832  622  384  627  441  521  967  607  339  880  364  682
 556  401  762  864  862  947  604  293  890  442  891  394  569  316
 222  782  438  630  387  551  315  386  755  557  349  605  754  772
 617  262  492  399  272  382  554  971  531  857  286  491  290  941
 270  294  787  422  235  355  646  571  392  820  414  745  713  228
 996]
```

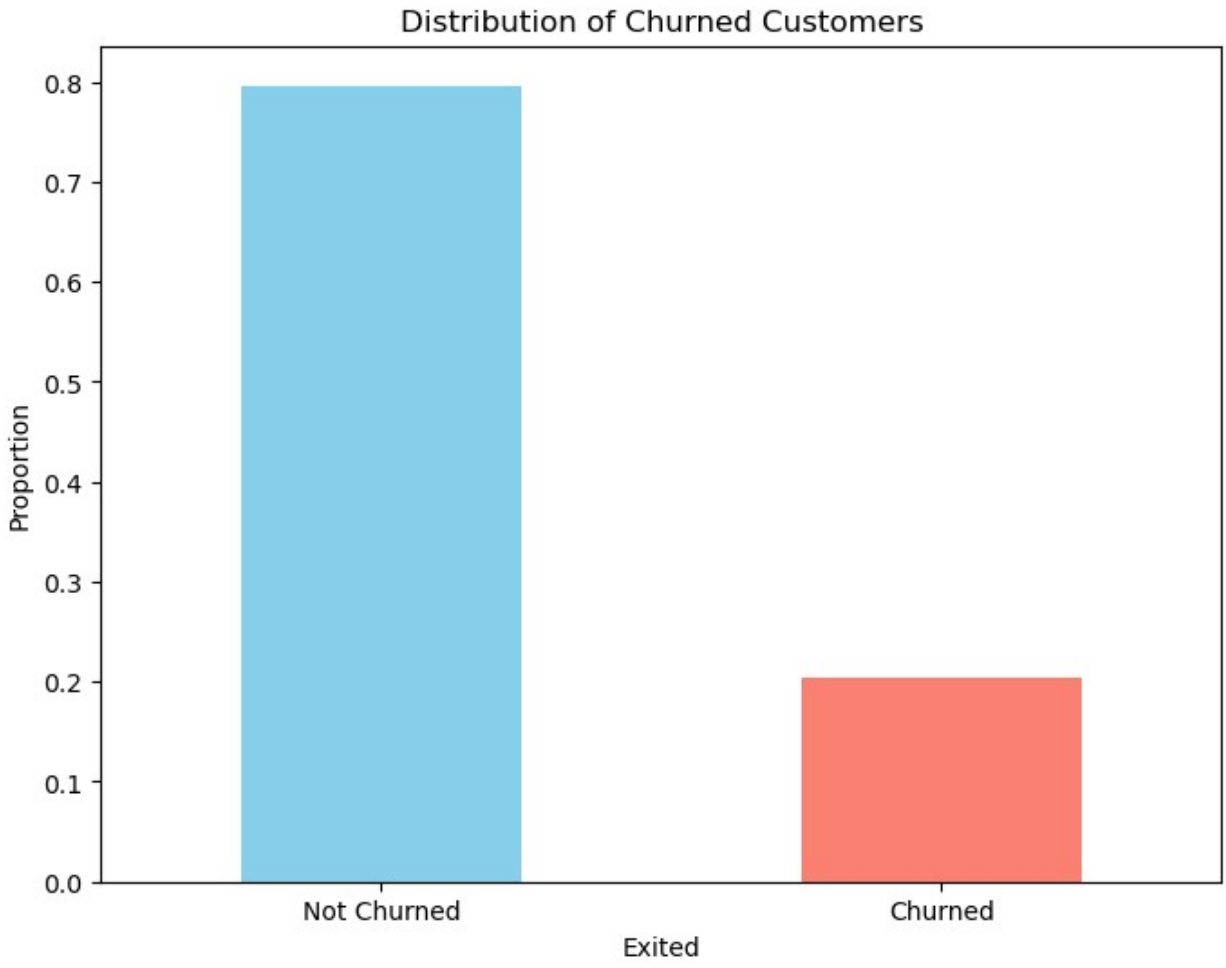
DISTRIBUTION ANALYSIS

Univariate analysis

```
# Distribution of the target variable
churn_counts = data['Exited'].value_counts(normalize=True)
print(churn_counts)
```

```
Exited
0    0.7962
1    0.2038
Name: proportion, dtype: float64
```

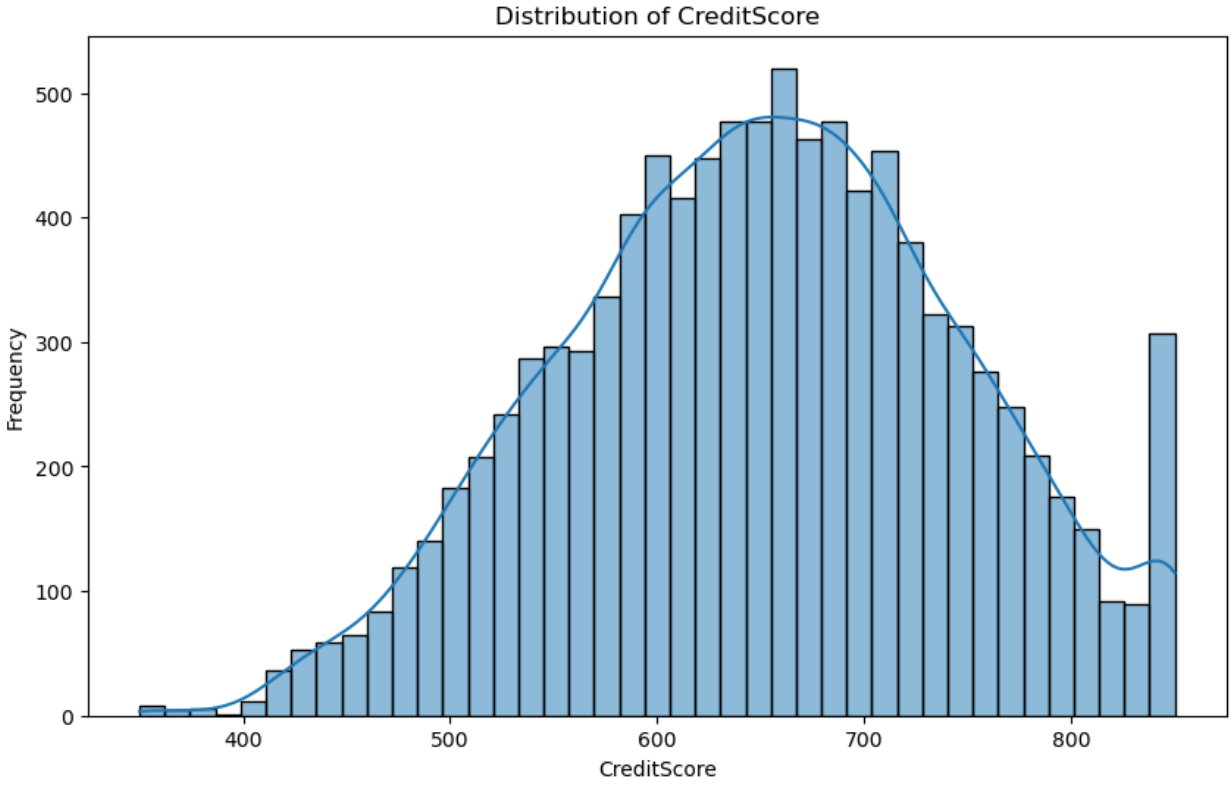
```
# Plotting the distribution of the target variable
plt.figure(figsize=(8, 6))
churn_counts.plot(kind='bar', color=['skyblue', 'salmon'])
plt.title('Distribution of Churned Customers')
plt.xlabel('Exited')
plt.ylabel('Proportion')
plt.xticks(ticks=[0, 1], labels=['Not Churned', 'Churned'],
rotation=0)
plt.show()
```



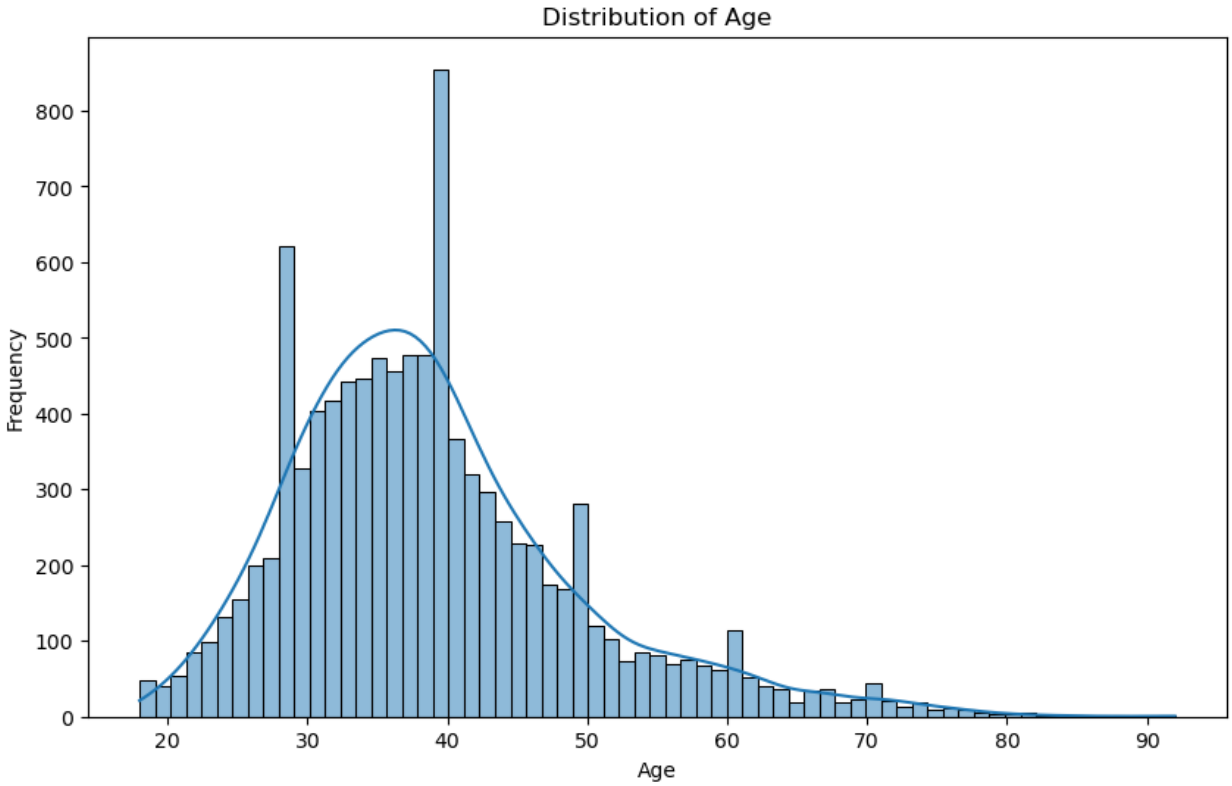

```
numerical_columns = data.select_dtypes(include=['int64',
'float64']).columns

numerical_columns = numerical_columns.drop('CustomerId')
```

```
# Plot histograms with KDE
for column in numerical_columns:
    plt.figure(figsize=(10, 6))
    sns.histplot(data[column], kde=True)
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()
```

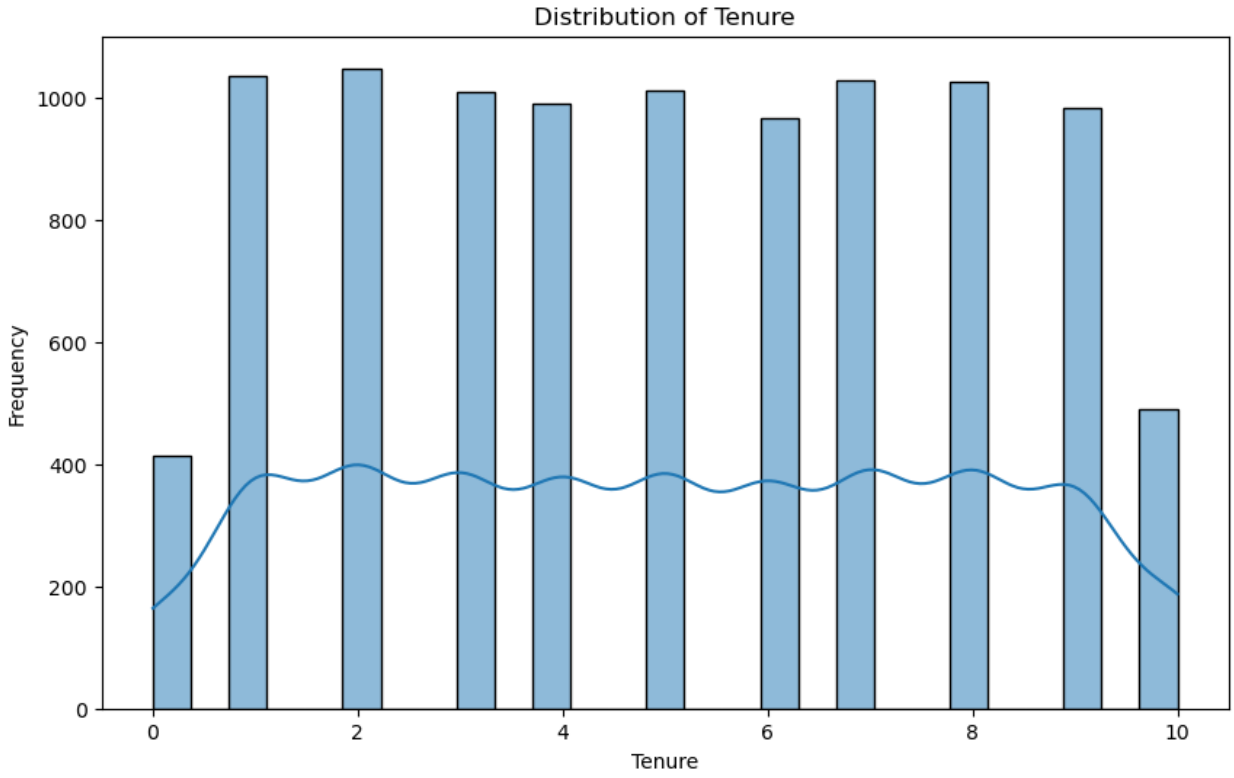
```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



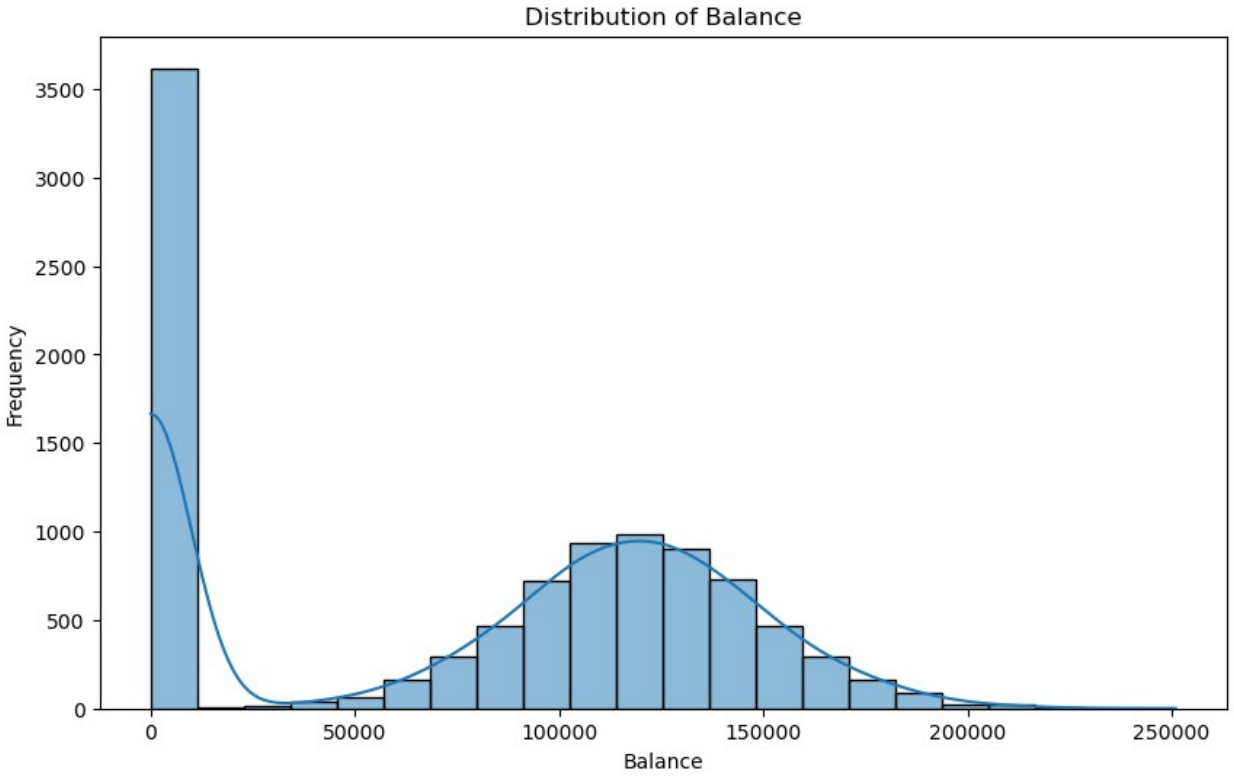

```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



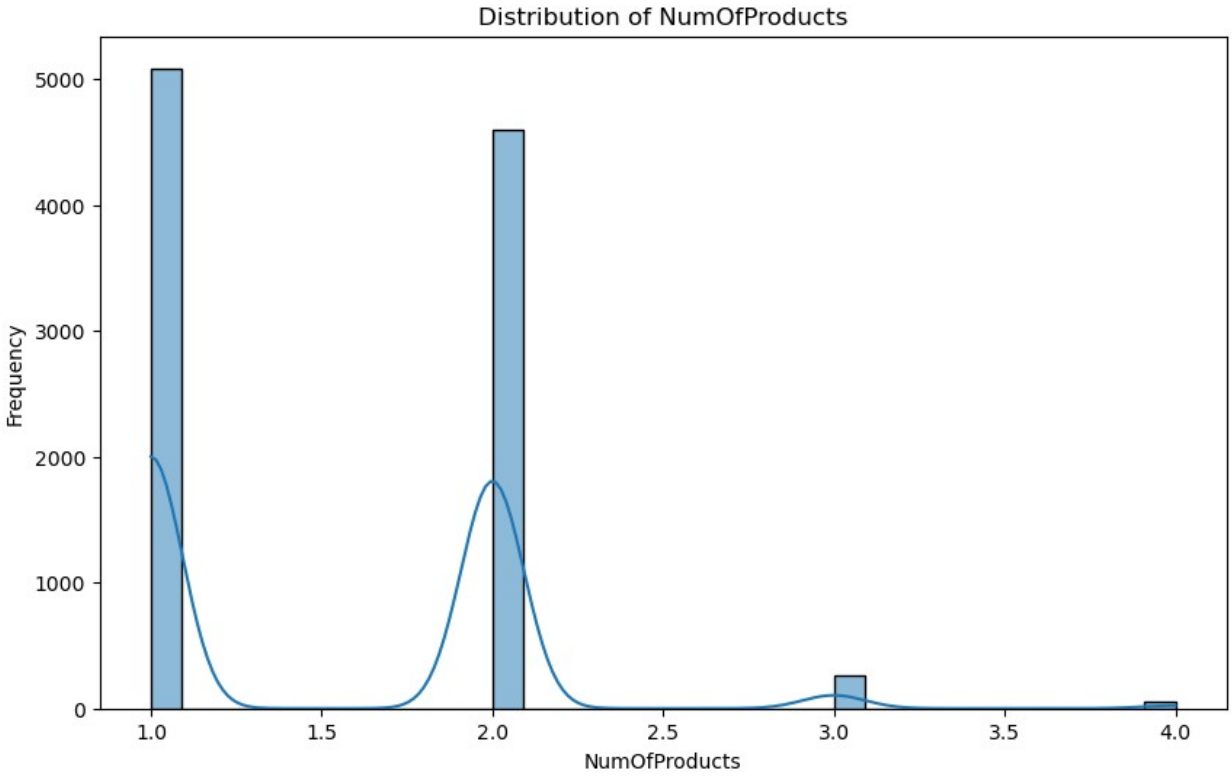

```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



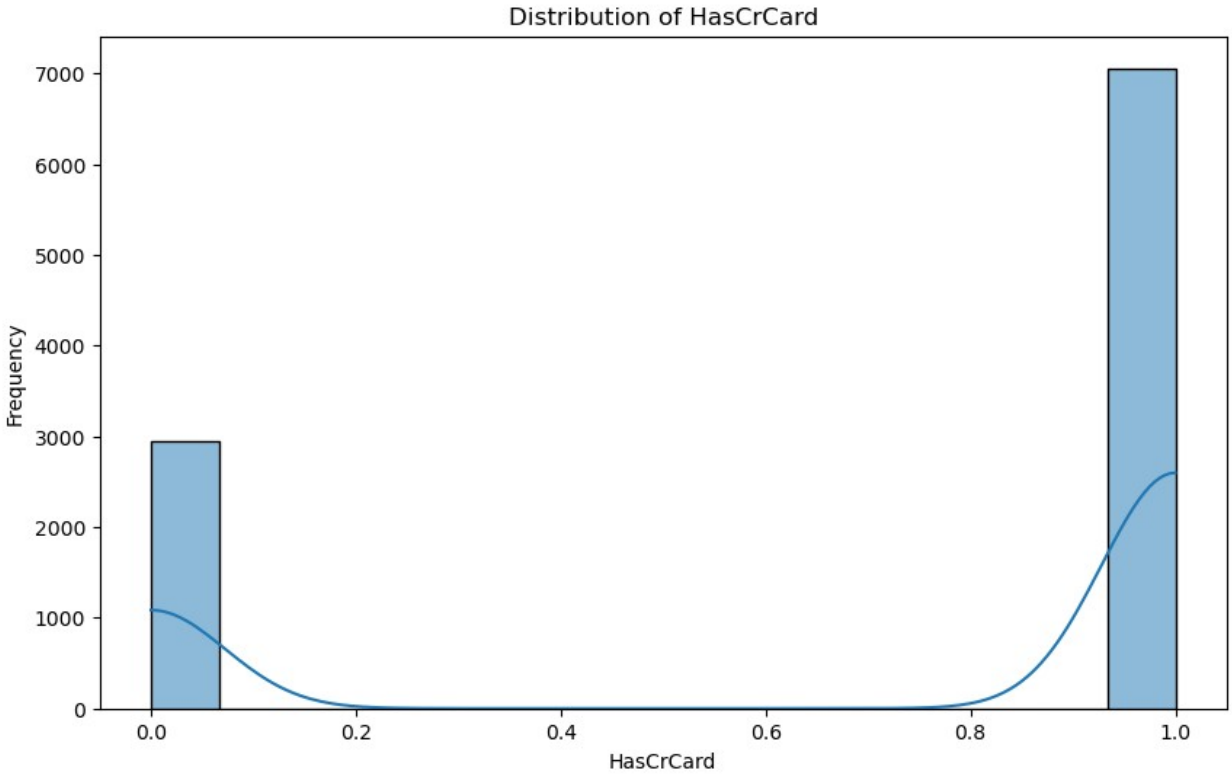

```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



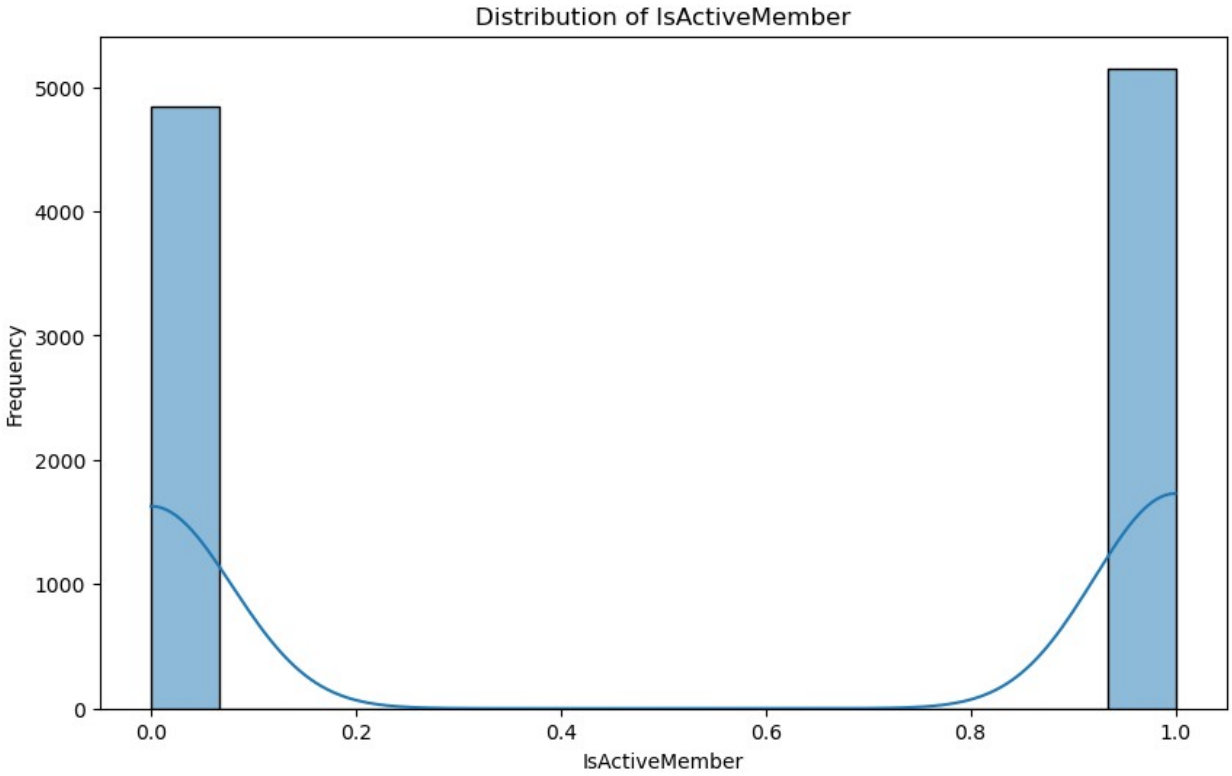

```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



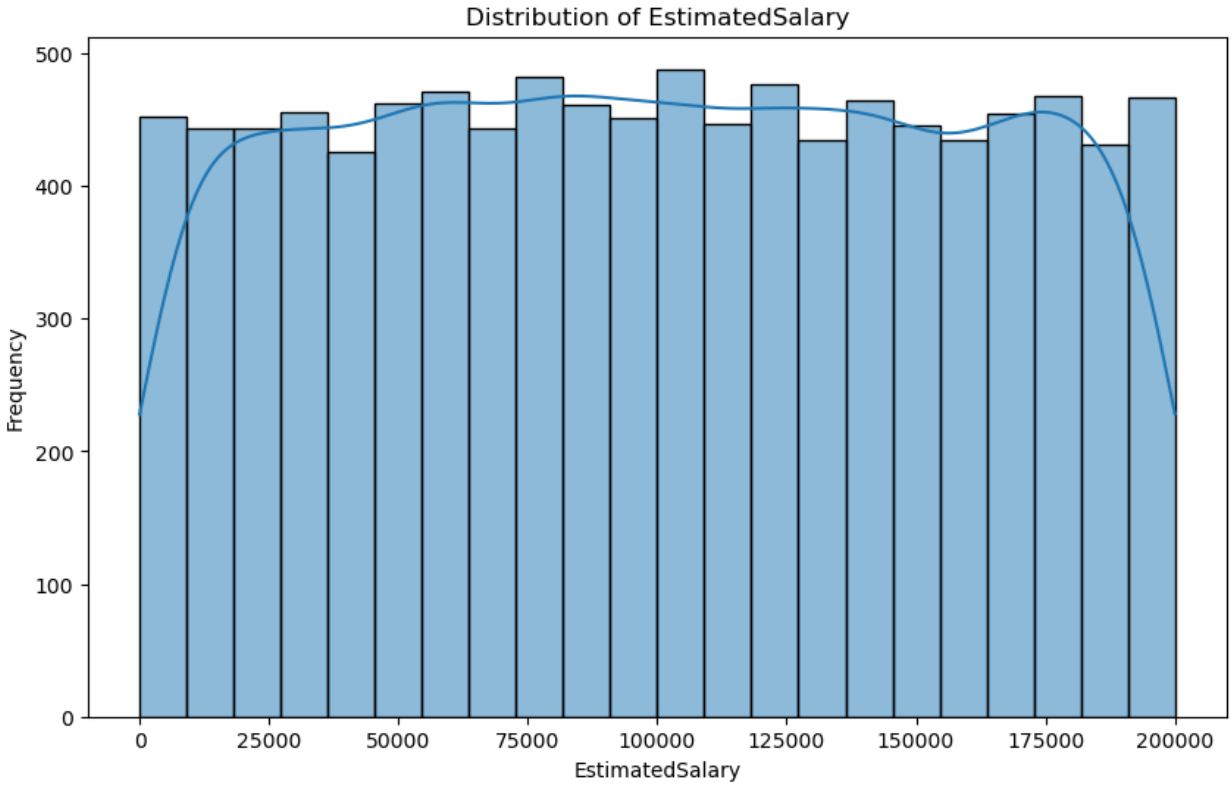

```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



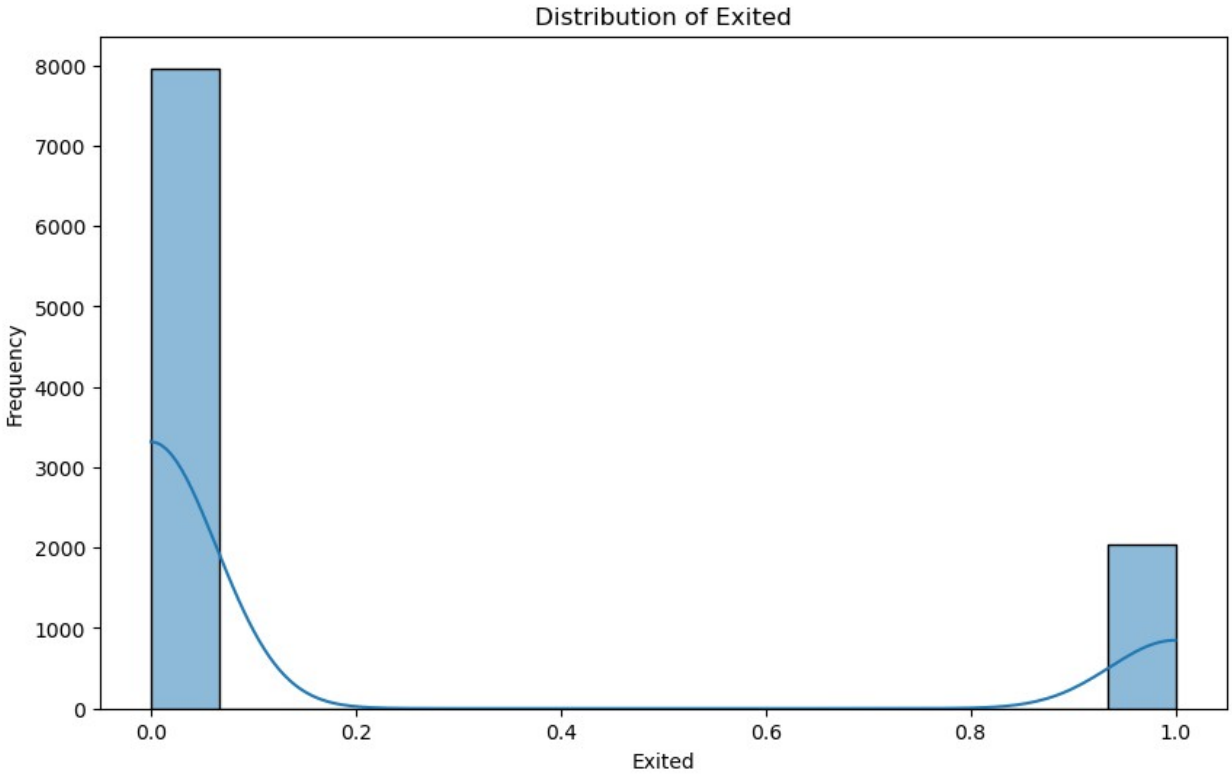

```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



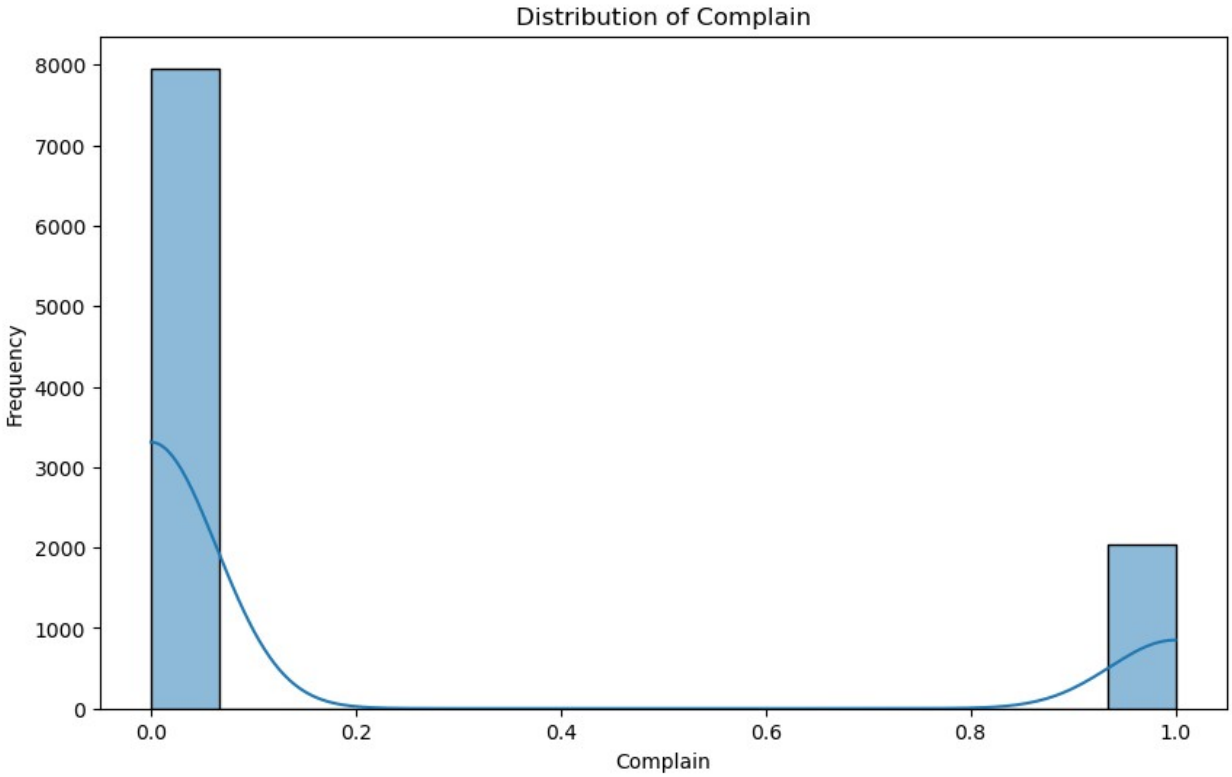

```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



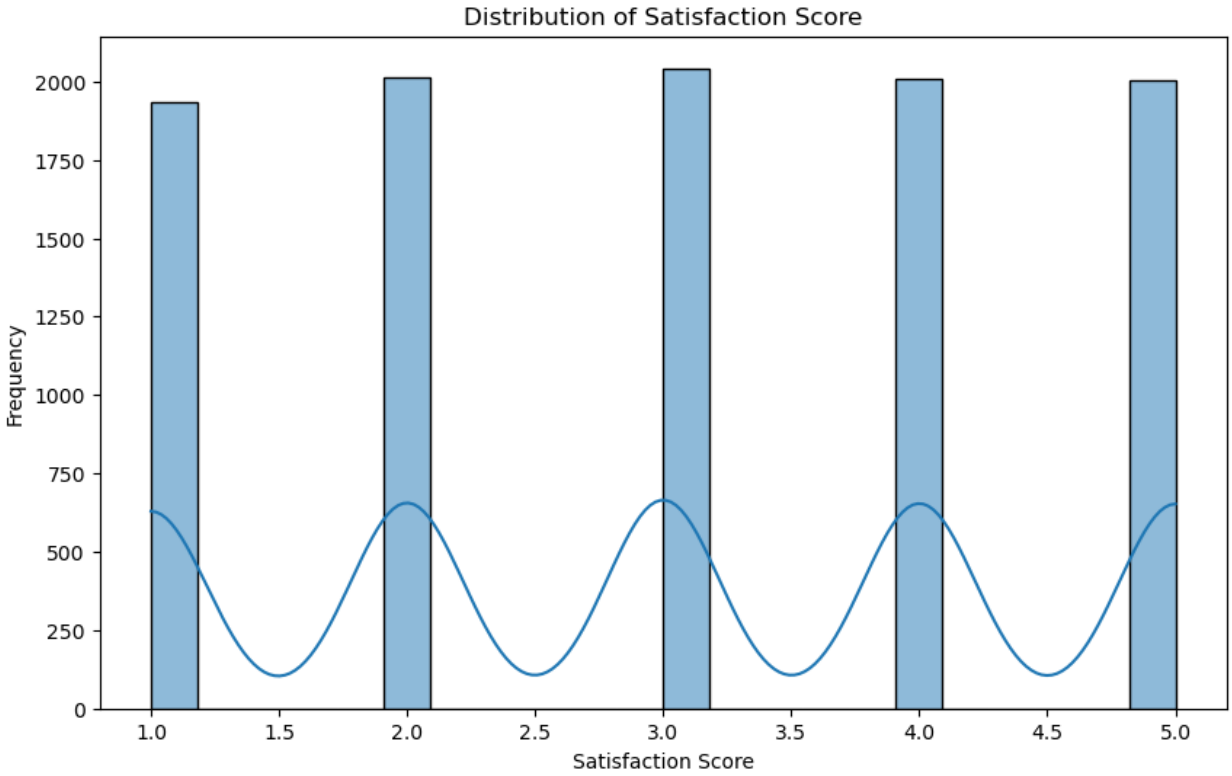

```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



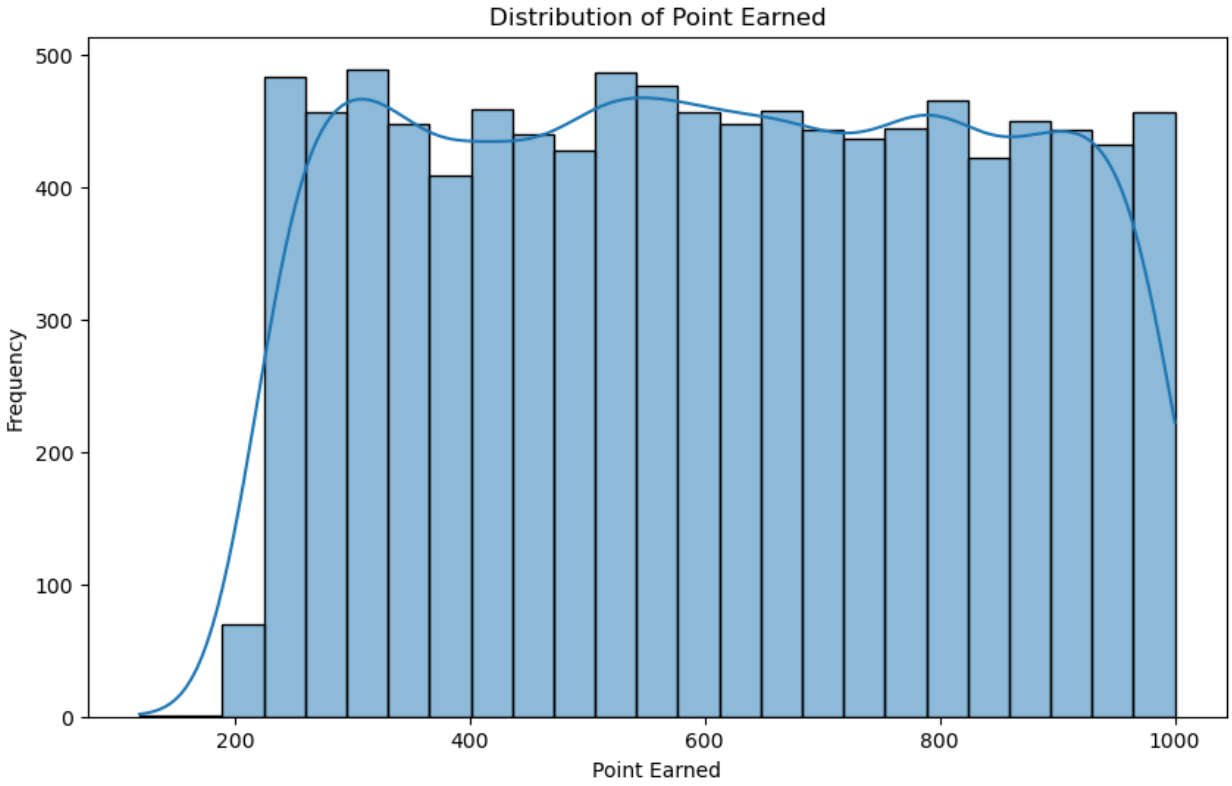

```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

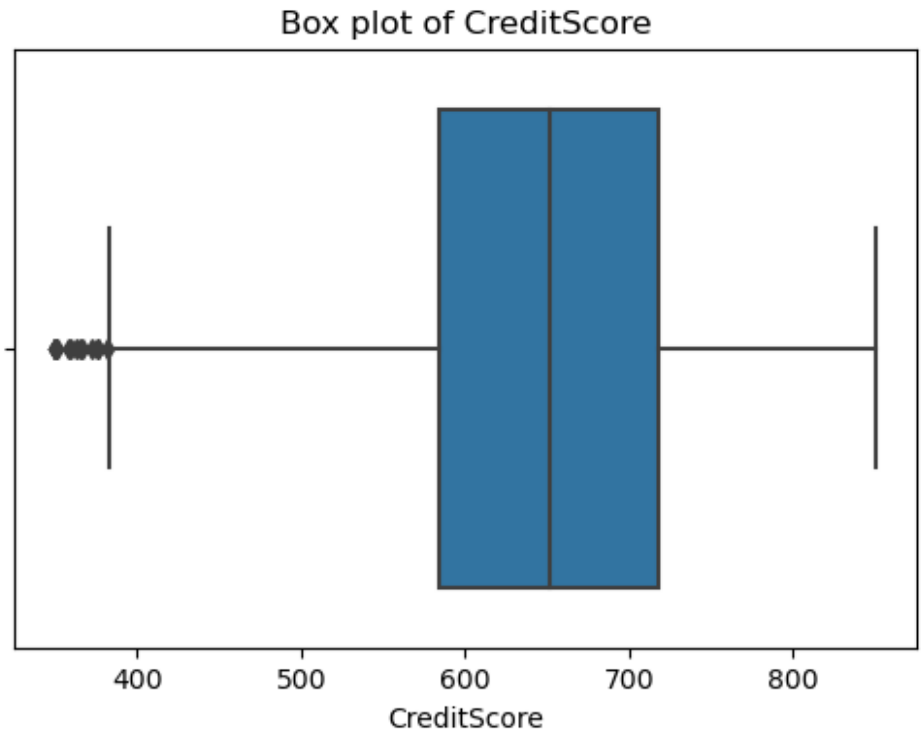


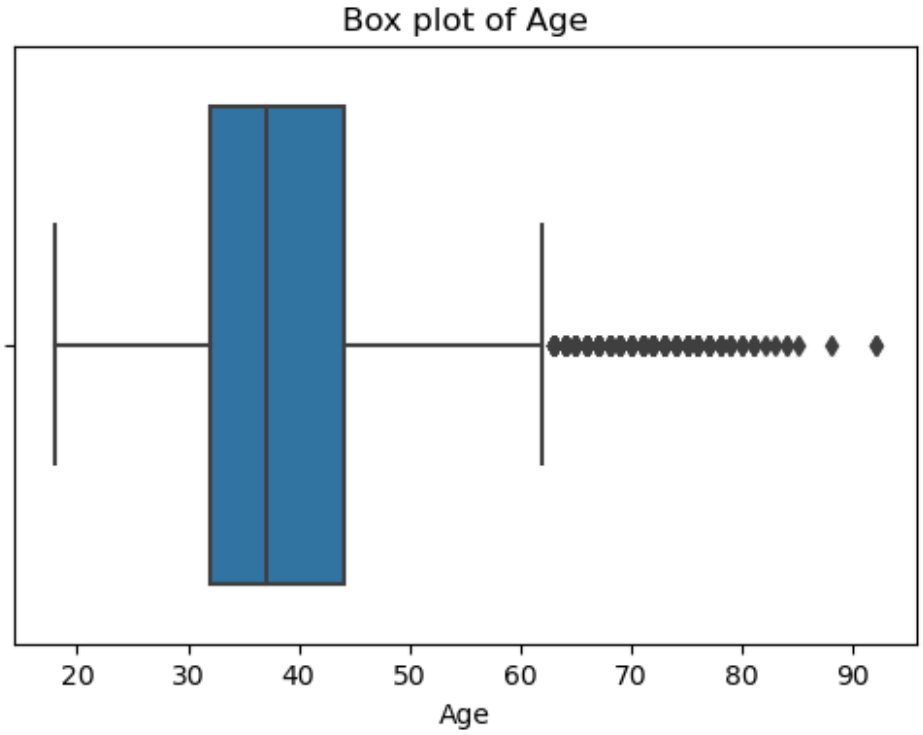

```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

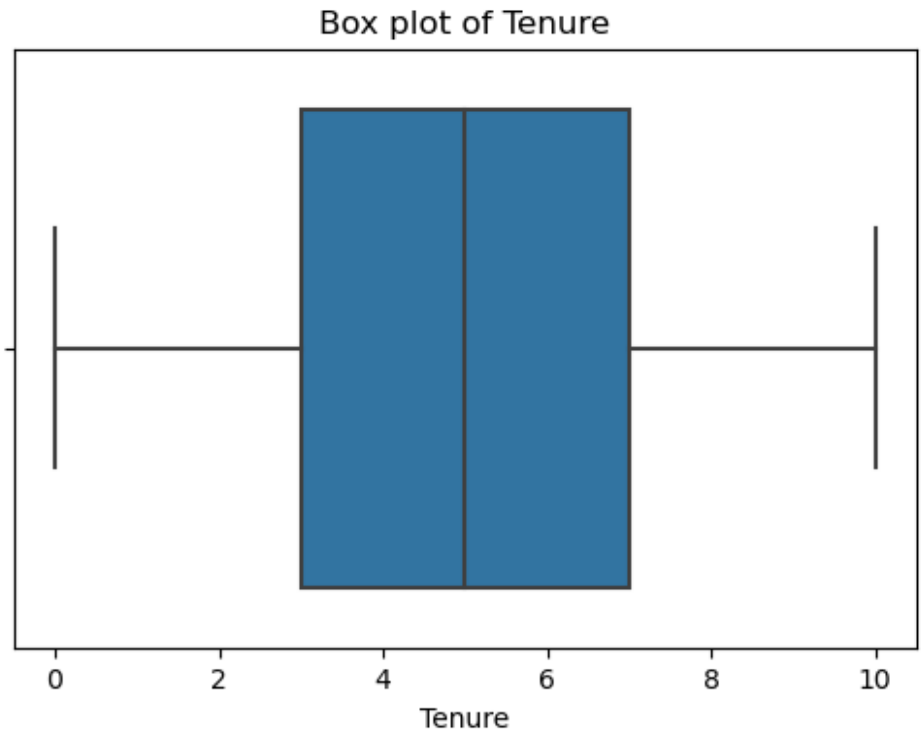


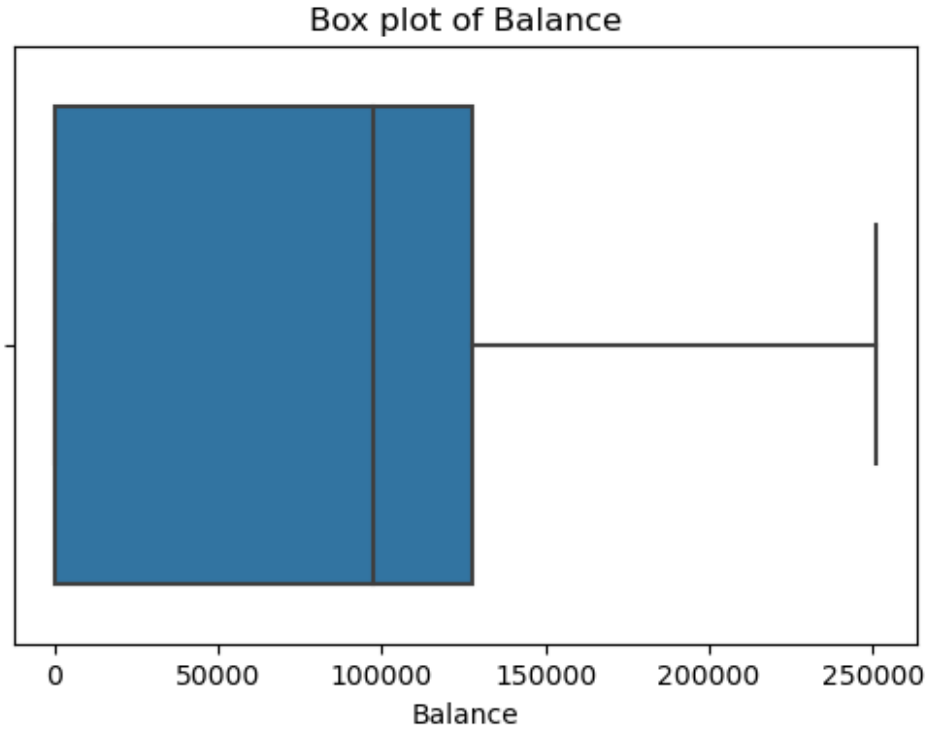
Handling Outliers

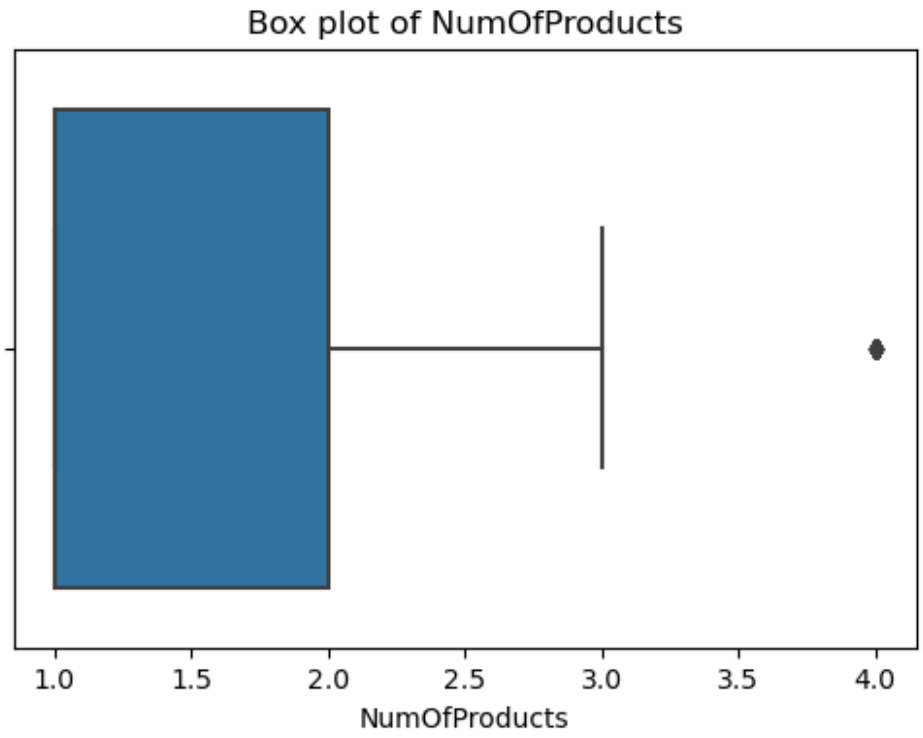
```
# Create box plots for each numerical column
for column in numerical_columns:
    plt.figure(figsize=(6, 4))
    sns.boxplot(x=data[column])
    plt.title(f'Box plot of {column}')
    plt.show()
```

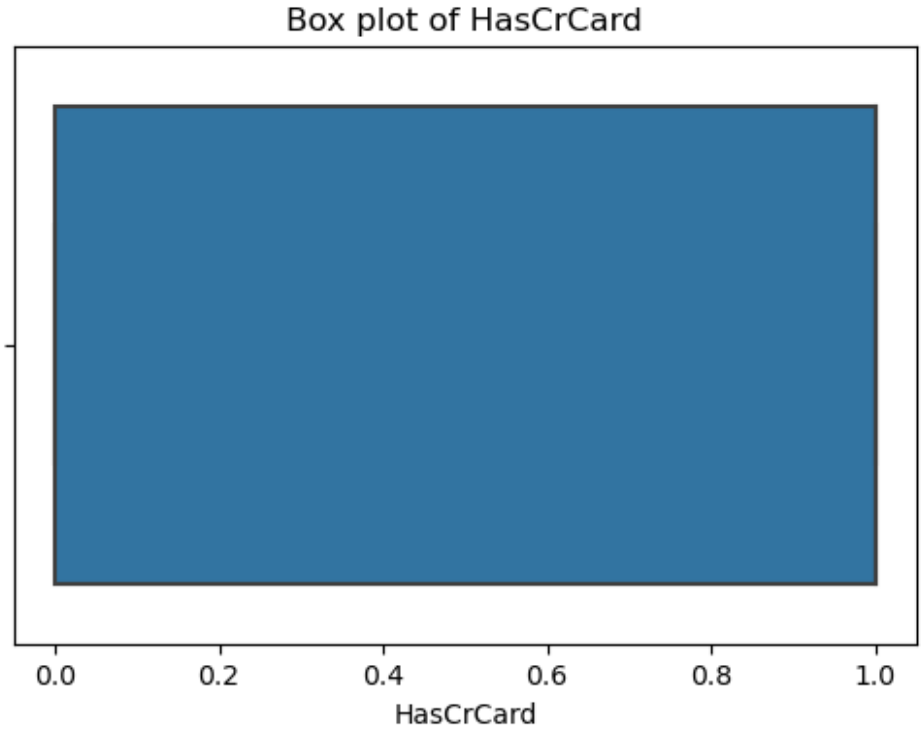

Box plot of CreditScore
400
500
600
700
800
CreditScore

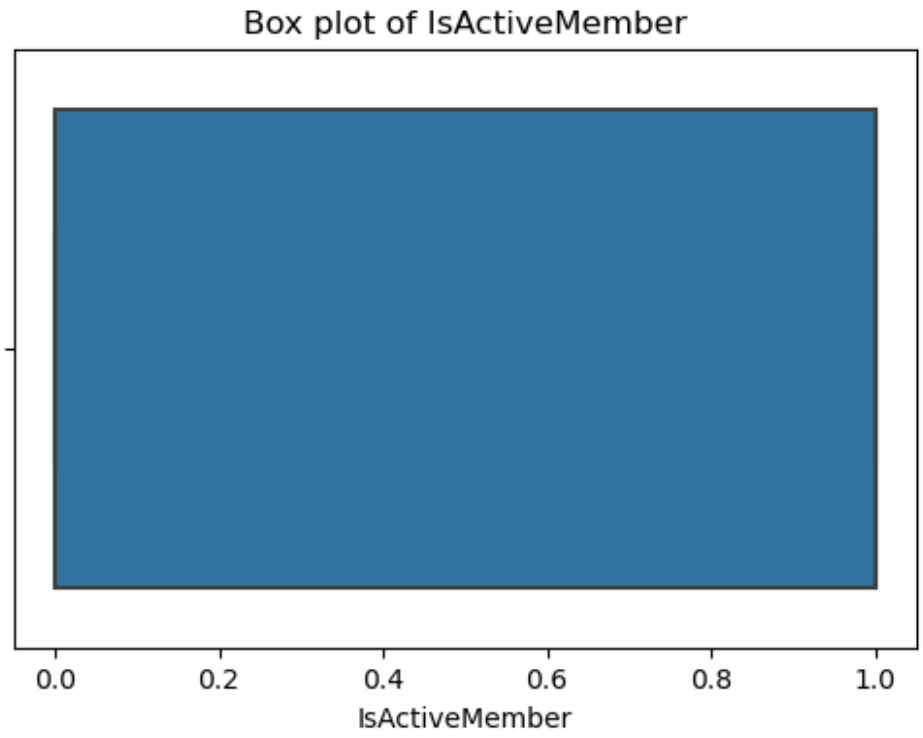

Box plot of Age
20
30
40
50
60
70
80
90
Age

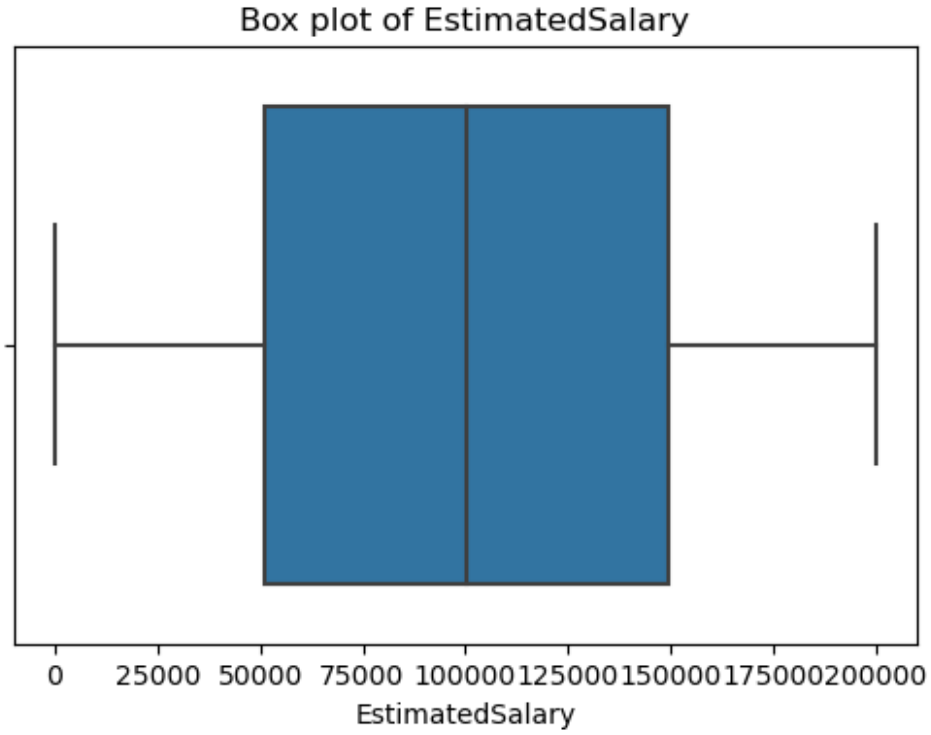

Box plot of Tenure
0
2
4
6
8
10
Tenure

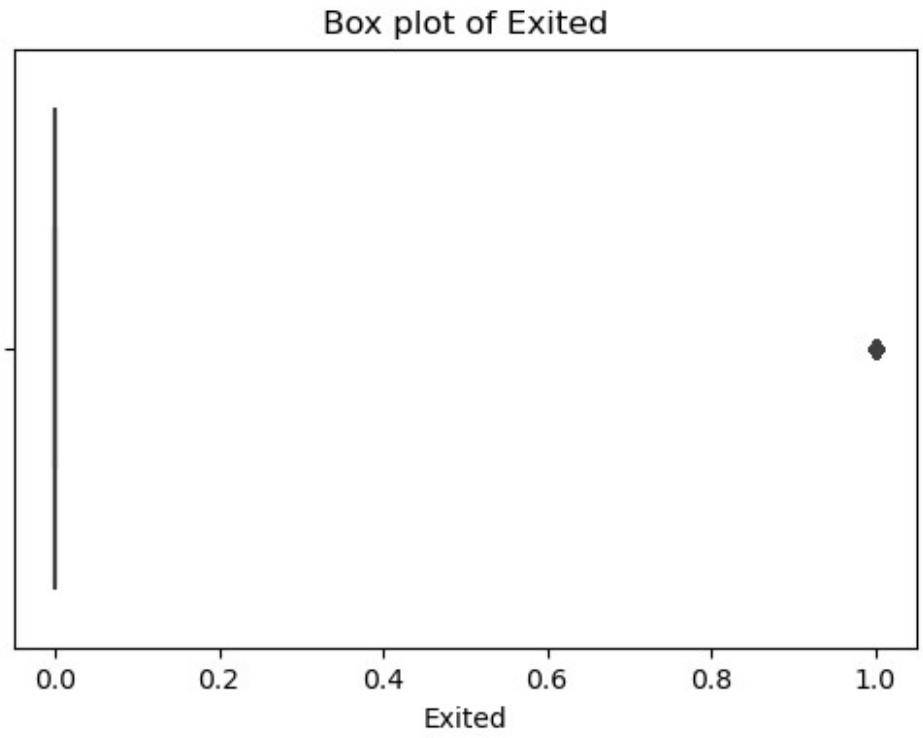

Box plot of Balance
0
50000
100000
150000
200000
250000
Balance

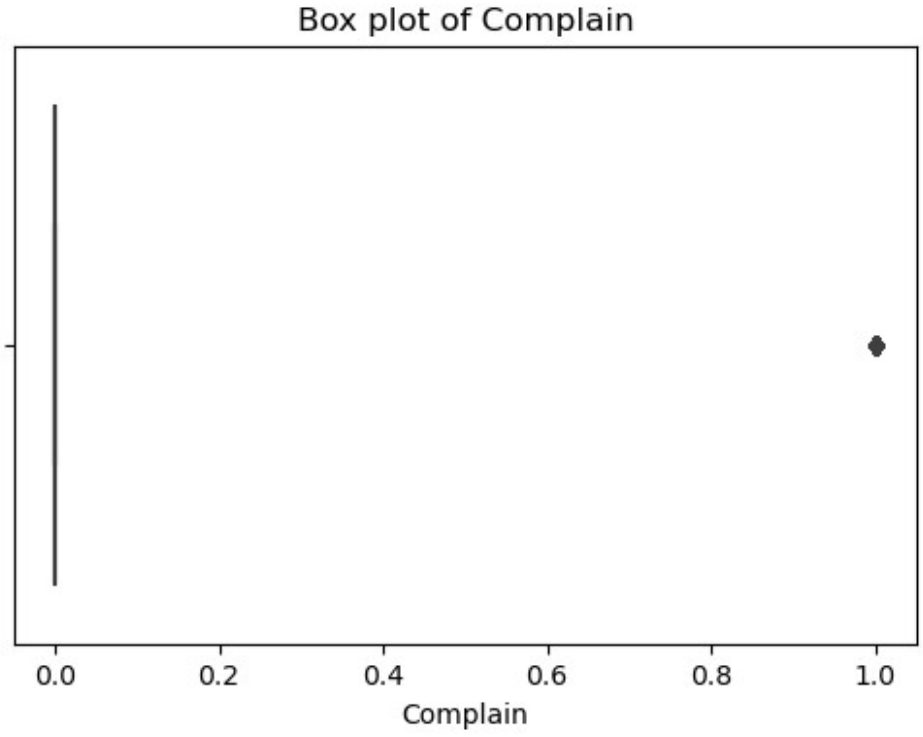

Box plot of NumOfProducts
1.0
1.5
2.0
2.5
3.0
3.5
4.0
NumOfProducts

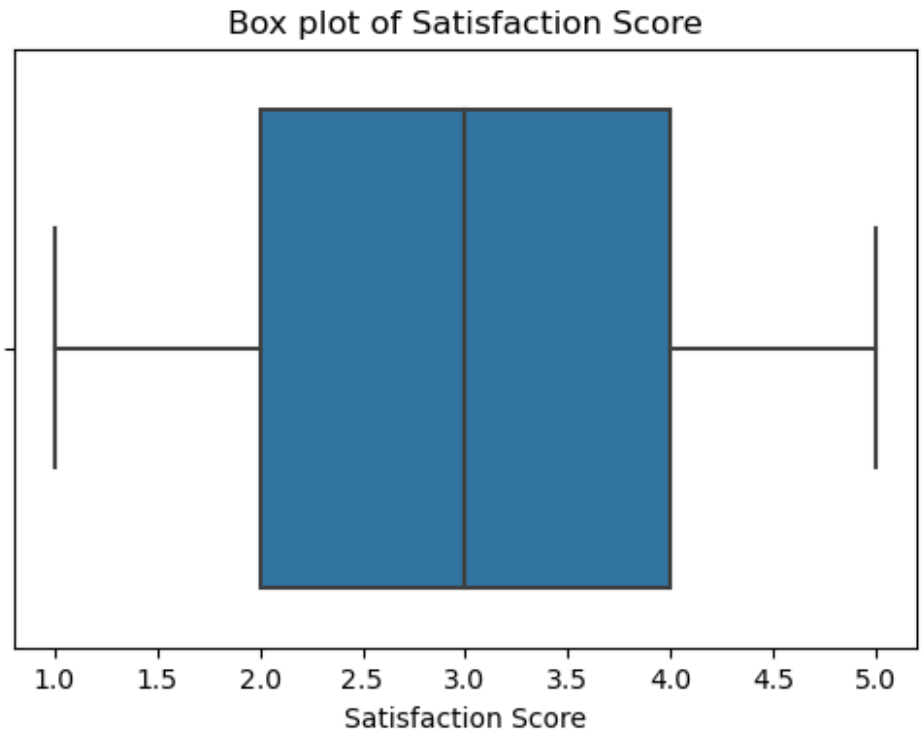

Box plot of HasCrCard
0.0
0.2
0.4
0.6
0.8
1.0
HasCrCard

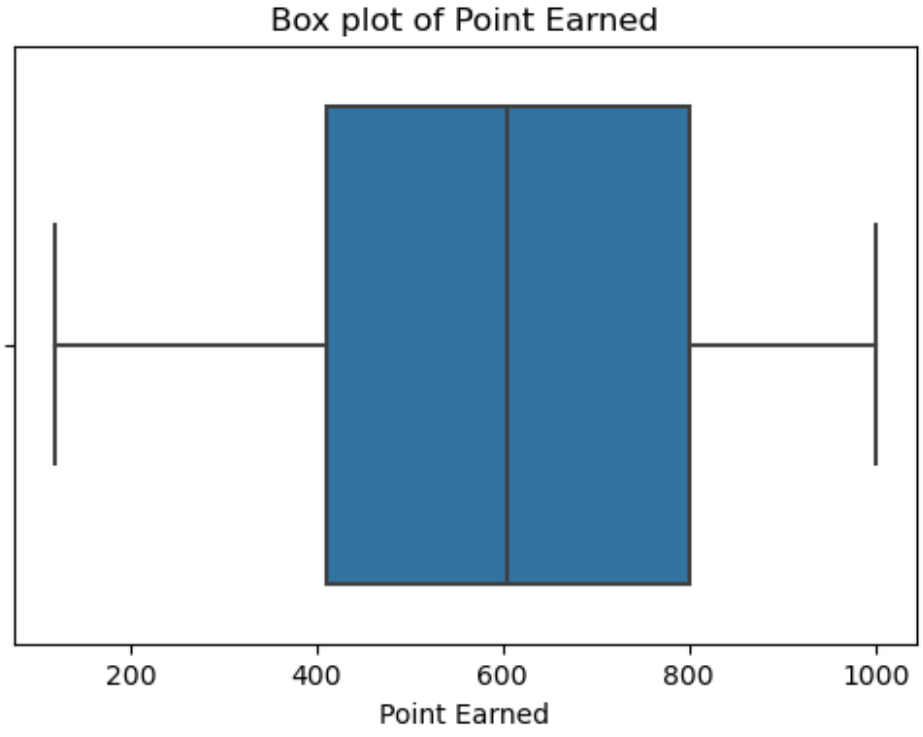

Box plot of IsActiveMember
0.0
0.2
0.4
0.6
0.8
1.0
IsActiveMember


Box plot of EstimatedSalary
0
25000
50000
75000
100000
125000
150000
175000
200000
EstimatedSalary


Box plot of Exited
0.0
0.2
0.4
0.6
0.8
1.0
Exited


Box plot of Complain
0.0
0.2
0.4
0.6
0.8
1.0
Complain


Box plot of Satisfaction Score
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Satisfaction Score


Box plot of Point Earned
200
400
600
800
1000
Point Earned

EXPLORATORY DATA ANALYSIS(EDA)

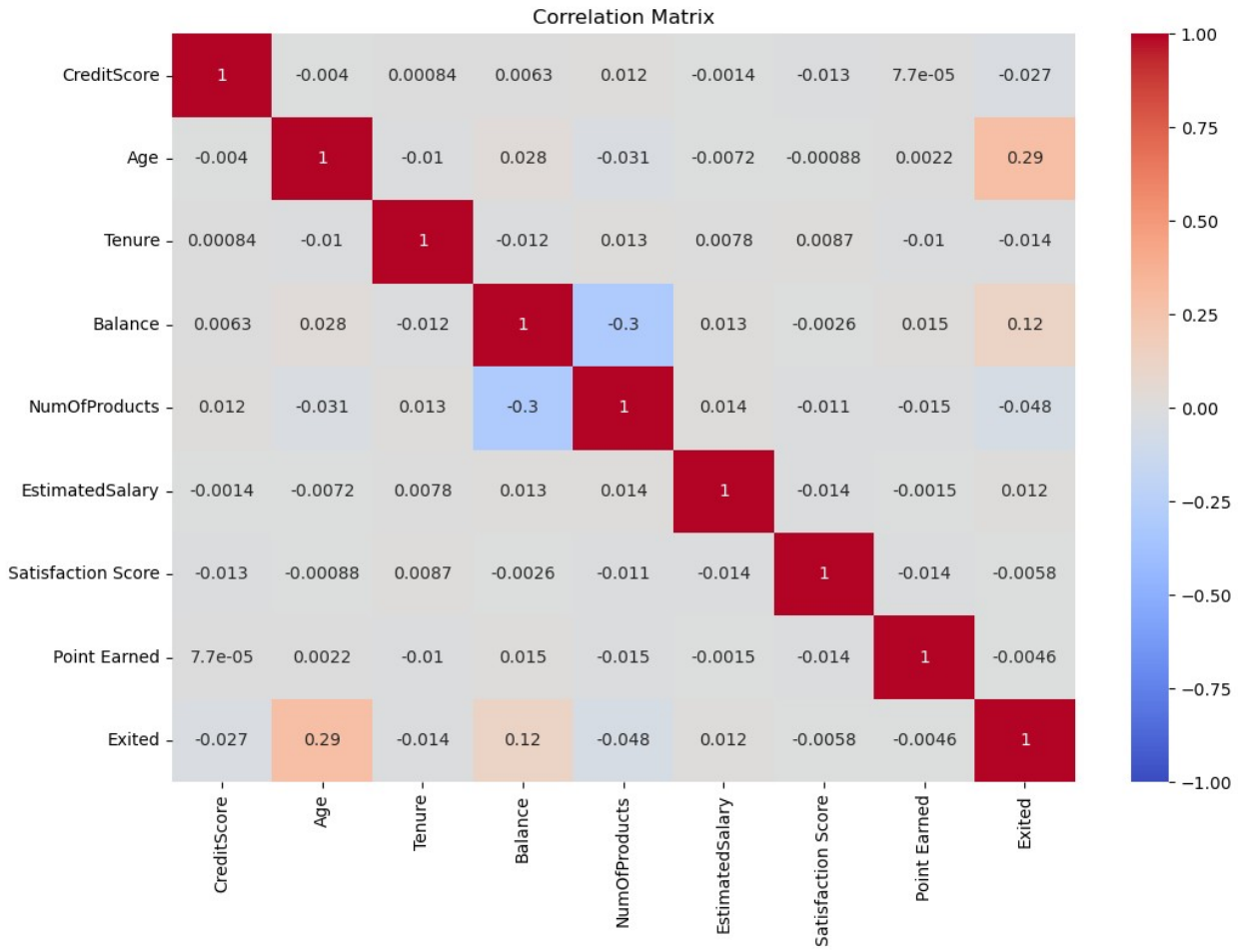
Correaltion Analysis

```
# List of numerical columns to check for correlation
numerical_columns = ['CreditScore', 'Age', 'Tenure', 'Balance',
'NumOfProducts', 'EstimatedSalary', 'Satisfaction Score', 'Point
Earned']

# Add the target column 'Exited' to the list
numerical_columns_with_target = numerical_columns + ['Exited']

# Calculate the correlation matrix
correlation_matrix = data[numerical_columns_with_target].corr()

# Display the correlation matrix
print(correlation_matrix)

# Visualize the correlation matrix using a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1,
vmax=1)
plt.title('Correlation Matrix')
plt.show()
```

```
                    CreditScore       Age    Tenure   Balance
NumOfProducts  \
CreditScore            1.000000 -0.003965  0.000842  0.006268
0.012238
Age                   -0.003965  1.000000 -0.009997  0.028308      -
0.030680
Tenure                 0.000842 -0.009997  1.000000 -0.012254
0.013444
Balance                0.006268  0.028308 -0.012254  1.000000      -
0.304180
NumOfProducts          0.012238 -0.030680  0.013444 -0.304180
1.000000
EstimatedSalary       -0.001384 -0.007201  0.007784  0.012797
0.014204
Satisfaction Score    -0.012599 -0.000876  0.008663 -0.002588      -
0.011394
Point Earned           0.000077  0.002222 -0.010196  0.014608      -
0.015330
Exited                -0.026771  0.285296 -0.013656  0.118577      -
0.047611

                    EstimatedSalary  Satisfaction Score  Point Earned
\
CreditScore               -0.001384           -0.012599      0.000077
```

Age	-0.007201	-0.000876	0.002222
Tenure	0.007784	0.008663	-0.010196
Balance	0.012797	-0.002588	0.014608
NumOfProducts	0.014204	-0.011394	-0.015330
EstimatedSalary	1.000000	-0.013747	-0.001515
Satisfaction Score	-0.013747	1.000000	-0.014400
Point Earned	-0.001515	-0.014400	1.000000
Exited	0.012490	-0.005849	-0.004628

	Exited
CreditScore	-0.026771
Age	0.285296
Tenure	-0.013656
Balance	0.118577
NumOfProducts	-0.047611
EstimatedSalary	0.012490
Satisfaction Score	-0.005849
Point Earned	-0.004628
Exited	1.000000

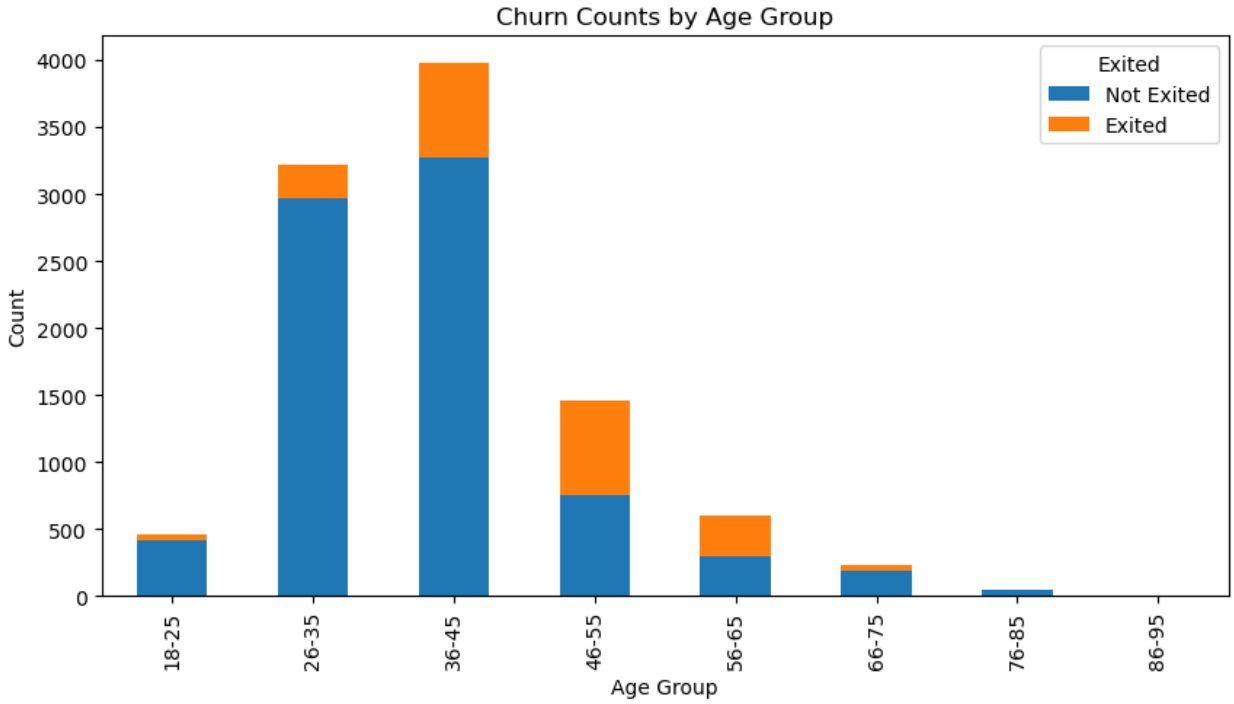


CUSTOMER PROFILE ANALYSIS

```
# Define age groups
bins = [18, 25, 35, 45, 55, 65, 75, 85, 95]
labels = ['18-25', '26-35', '36-45', '46-55', '56-65', '66-75', '76-
85', '86-95']
data['AgeGroup'] = pd.cut(data['Age'], bins=bins, labels=labels,
right=False)

# Calculate counts for each age group
age_group_counts = data.groupby(['AgeGroup',
'Exited']).size().unstack()
print(age_group_counts)
```

```
Exited       0    1
AgeGroup
18-25      417   40
26-35     2972  250
36-45     3277  704
46-55      756  702
56-65      301  299
```

```
66-75      186   42
76-85       49    1
86-95        4    0

C:\Users\user\AppData\Local\Temp\ipykernel_12992\3597434540.py:7:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  age_group_counts = data.groupby(['AgeGroup',
'Exited']).size().unstack()

age_group_churn = data.groupby('AgeGroup')
['Exited'].mean().sort_values()
print(age_group_churn)

AgeGroup
86-95    0.000000
76-85    0.020000
26-35    0.077592
18-25    0.087527
36-45    0.176840
66-75    0.184211
46-55    0.481481
56-65    0.498333
Name: Exited, dtype: float64

C:\Users\user\AppData\Local\Temp\ipykernel_12992\1387110758.py:1:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  age_group_churn = data.groupby('AgeGroup')
['Exited'].mean().sort_values()

# Bivariate Analysis
# Plot counts by Age Group
age_group_counts.plot(kind='bar', stacked=True, figsize=(10, 5))
plt.title('Churn Counts by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Count')
plt.legend(title='Exited', labels=['Not Exited', 'Exited'])
plt.show()
```



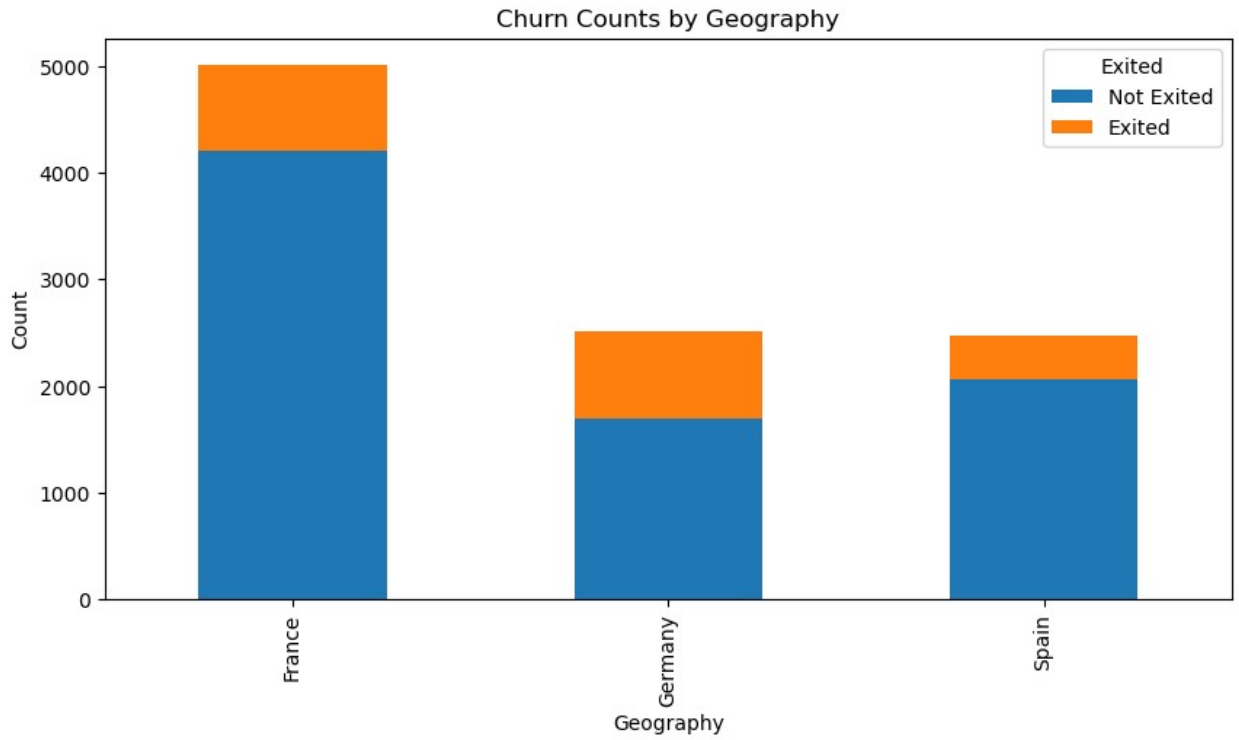

```
# Calculate counts for each geographical location
geography_counts = data.groupby(['Geography',
'Exited']).size().unstack()
print(geography_counts)
```

```
Exited        0    1
Geography
France     4203  811
Germany    1695  814
Spain      2064  413
```

```
geo_group_churn = data.groupby('Geography')
['Exited'].mean().sort_values()
print(geo_group_churn)
```

```
Geography
France     0.161747
Spain      0.166734
Germany    0.324432
Name: Exited, dtype: float64
```

```
# Plot counts by Geography
geography_counts.plot(kind='bar', stacked=True, figsize=(10, 5))
plt.title('Churn Counts by Geography')
plt.xlabel('Geography')
plt.ylabel('Count')
plt.legend(title='Exited', labels=['Not Exited', 'Exited'])
plt.show()
```



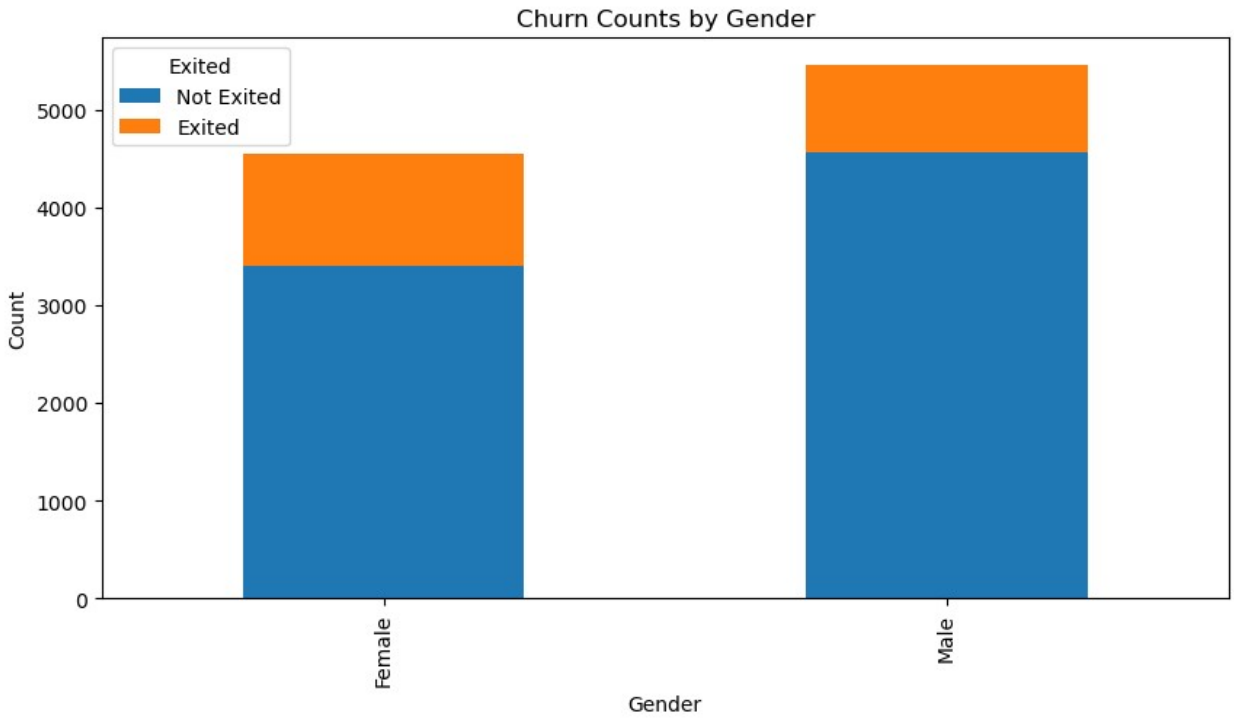

```
# Calculate counts for each gender
gender_counts = data.groupby(['Gender', 'Exited']).size().unstack()
print(gender_counts)
```

```
Exited     0     1
Gender
Female  3404  1139
Male    4558   899
```

```
gender_group_churn = data.groupby('Gender')
['Exited'].mean().sort_values()
print(gender_group_churn)
```

```
Gender
Male      0.164743
Female    0.250715
Name: Exited, dtype: float64
```

```
# Plot counts by Gender
gender_counts.plot(kind='bar', stacked=True, figsize=(10, 5))
plt.title('Churn Counts by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.legend(title='Exited', labels=['Not Exited', 'Exited'])
plt.show()
```



COMPARATIVE ANALYSIS

Churn By Geography

H0:There is no association between geography and customer churn. In other words, the churn rate is independent of the geographical location. H1: There is an association between geography and customer churn. In other words, the churn rate depends on the geographical location.

```
# Function to format p-values for readability
def format_p_value(p):
    return round(p,3)

# Assume aplha 0.05.
from scipy.stats import chi2_contingency

# Churn by Geography
contingency_geo = pd.crosstab(data['Geography'], data['Exited'])
chi2_geo, p_geo, dof_geo, expected_geo =
chi2_contingency(contingency_geo)
print("Chi-square statistic for churn by geography:", chi2_geo)
print("P-value for churn by geography:", format_p_value(p_geo))

if p_geo < 0.05:
    print("Reject the null hypothesis: There is an association between
geography and customer churn.")
else:
```

```
    print("Fail to reject the null hypothesis: There is no association
between geography and customer churn.")
```

```
Chi-square statistic for churn by geography: 300.6264011211942
P-value for churn by geography: 0.0
Reject the null hypothesis: There is an association between geography
and customer churn.
```

Churn by Gender

H0: There is no association between gender and customer churn. In other words, the churn rate is independent of gender. H1: There is an association between gender and customer churn. In other words, the churn rate depends on gender.

```
# Gender Differences in Churn
contingency_gender = pd.crosstab(data['Gender'], data['Exited'])
chi2_gender, p_gender, dof_gender, expected_gender =
chi2_contingency(contingency_gender)
print("Chi-square statistic for gender differences in churn:",
chi2_gender)
print("P-value for gender differences in churn:",
format_p_value(p_gender))

if p_gender < 0.05:
    print("Reject the null hypothesis: There is an association between
gender and customer churn.")
else:
    print("Fail to reject the null hypothesis: There is no association
between gender and customer churn.")
```

```
Chi-square statistic for gender differences in churn:
112.39655374778587
P-value for gender differences in churn: 0.0
Reject the null hypothesis: There is an association between gender and
customer churn.
```

Churn by Age

H0: There is no association between age and customer churn. In other words, the churn rate is independent of age. H1: There is an association between age and customer churn. In other words, the churn rate depends on age.

```
# Gender Differences in Churn
contingency_age = pd.crosstab(data['AgeGroup'], data['Exited'])
chi2_age, p_age, dof_age, expected_age =
chi2_contingency(contingency_age)
print("Chi-square statistic for gender differences in churn:",
chi2_age)
print("P-value for gender differences in churn:",
```

```
format_p_value(p_age))

if p_gender < 0.05:
    print("Reject the null hypothesis: There is an association between
age and customer churn.")
else:
    print("Fail to reject the null hypothesis: There is no association
between age and customer churn.")
```

```
Chi-square statistic for gender differences in churn:
1397.7608390828918
P-value for gender differences in churn: 0.0
Reject the null hypothesis: There is an association between age and
customer churn.
```

BEHAVIORAL ANALYSIS

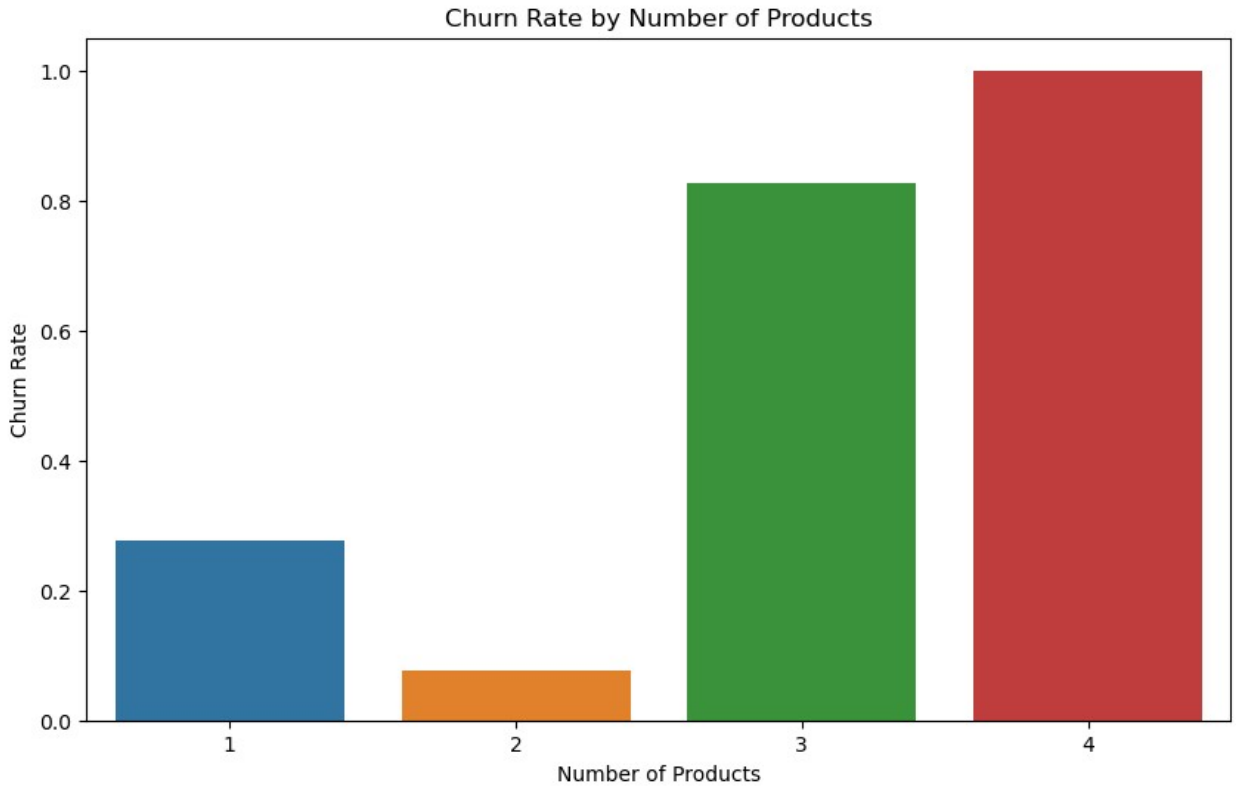
Product and services usage

```
# Descriptive Analysis
product_churn_rate = data.groupby('NumOfProducts')['Exited'].mean()
print("Churn rate by number of products:\n", product_churn_rate)

# Hypothesis Testing - Chi-Square Test
contingency_products = pd.crosstab(data['NumOfProducts'],
data['Exited'])
chi2_products, p_products, dof_products, expected_products =
chi2_contingency(contingency_products)


print("Chi-square statistic for number of products and churn:",
chi2_products)
print("P-value for number of products and churn:",
format_p_value(p_products))

if p_products < 0.05:
    print("Reject the null hypothesis: There is an association between
the number of products and customer churn.")
else:
    print("Fail to reject the null hypothesis: There is no association
between the number of products and customer churn.")
```

```
Churn rate by number of products:
 NumOfProducts
1    0.277144
2    0.076035
3    0.827068
4    1.000000
Name: Exited, dtype: float64
Chi-square statistic for number of products and churn:
```

```
1501.5048306588592
P-value for number of products and churn: 0.0
Reject the null hypothesis: There is an association between the number
of products and customer churn.
```

```
# Visualization
plt.figure(figsize=(10,6))
sns.barplot(x=product_churn_rate.index, y=product_churn_rate.values)
plt.title('Churn Rate by Number of Products')
plt.xlabel('Number of Products')
plt.ylabel('Churn Rate')
plt.show()
```



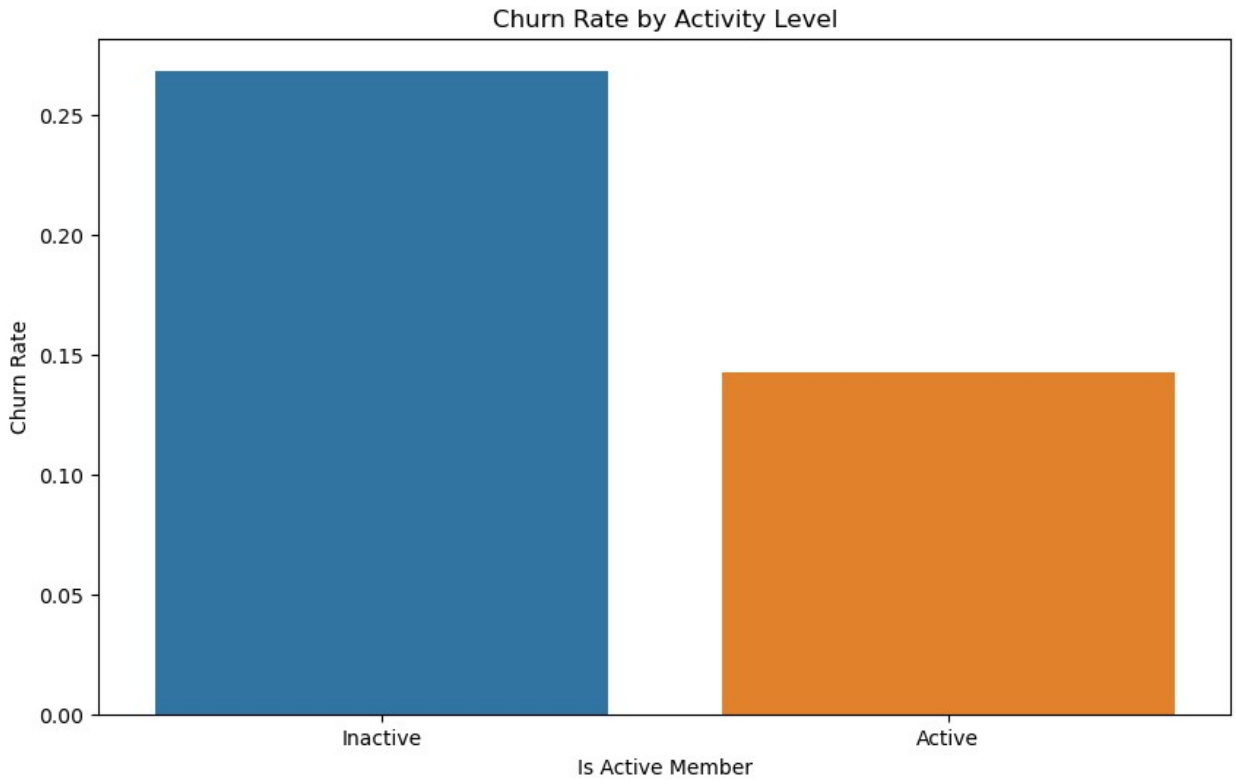
Activity Level Analysis

```
# Descriptive Analysis
activity_churn_rate = data.groupby('IsActiveMember')['Exited'].mean()
print("Churn rate by activity level:\n", activity_churn_rate)

# Hypothesis Testing - Chi-Square Test
contingency_activity = pd.crosstab(data['IsActiveMember'],
data['Exited'])
chi2_activity, p_activity, dof_activity, expected_activity =
chi2_contingency(contingency_activity)
```

```
print("Chi-square statistic for activity level and churn:",
chi2_activity)
print("P-value for activity level and churn:", p_activity)

if p_activity < 0.05:
    print("Reject the null hypothesis: There is an association between
activity level and customer churn.")
else:
    print("Fail to reject the null hypothesis: There is no association
between activity level and customer churn.")
```

```
Churn rate by activity level:
 IsActiveMember
0    0.268715
1    0.142691
Name: Exited, dtype: float64
Chi-square statistic for activity level and churn: 243.6948024819593
P-value for activity level and churn: 6.153167438113408e-55
Reject the null hypothesis: There is an association between activity
level and customer churn.
```

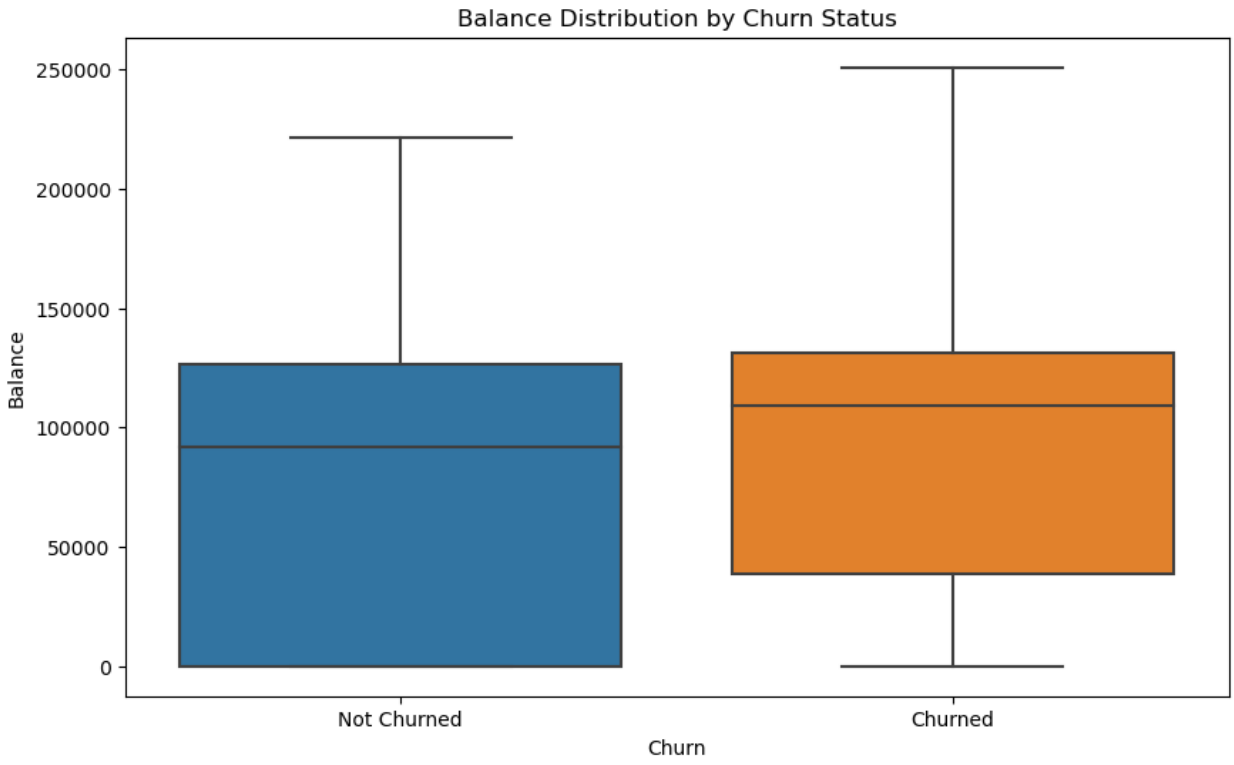
```
# Visualization
plt.figure(figsize=(10,6))
sns.barplot(x=activity_churn_rate.index, y=activity_churn_rate.values)
plt.title('Churn Rate by Activity Level')
plt.xlabel('Is Active Member')
plt.ylabel('Churn Rate')
plt.xticks([0, 1], ['Inactive', 'Active'])
plt.show()
```



FINANCIAL ANALYSIS

Balance vs churn

```
# Visualization
plt.figure(figsize=(10,6))
sns.boxplot(x='Exited', y='Balance', data=data)
plt.title('Balance Distribution by Churn Status')
plt.xlabel('Churn')
plt.ylabel('Balance')
plt.xticks([0, 1], ['Not Churned', 'Churned'])
plt.show()
```



H0: There is no association between balance and customer churn. In other words, the churn rate is independent of balance. H1: There is an association between balance and customer churn. In other words, the churn rate depends on balance.

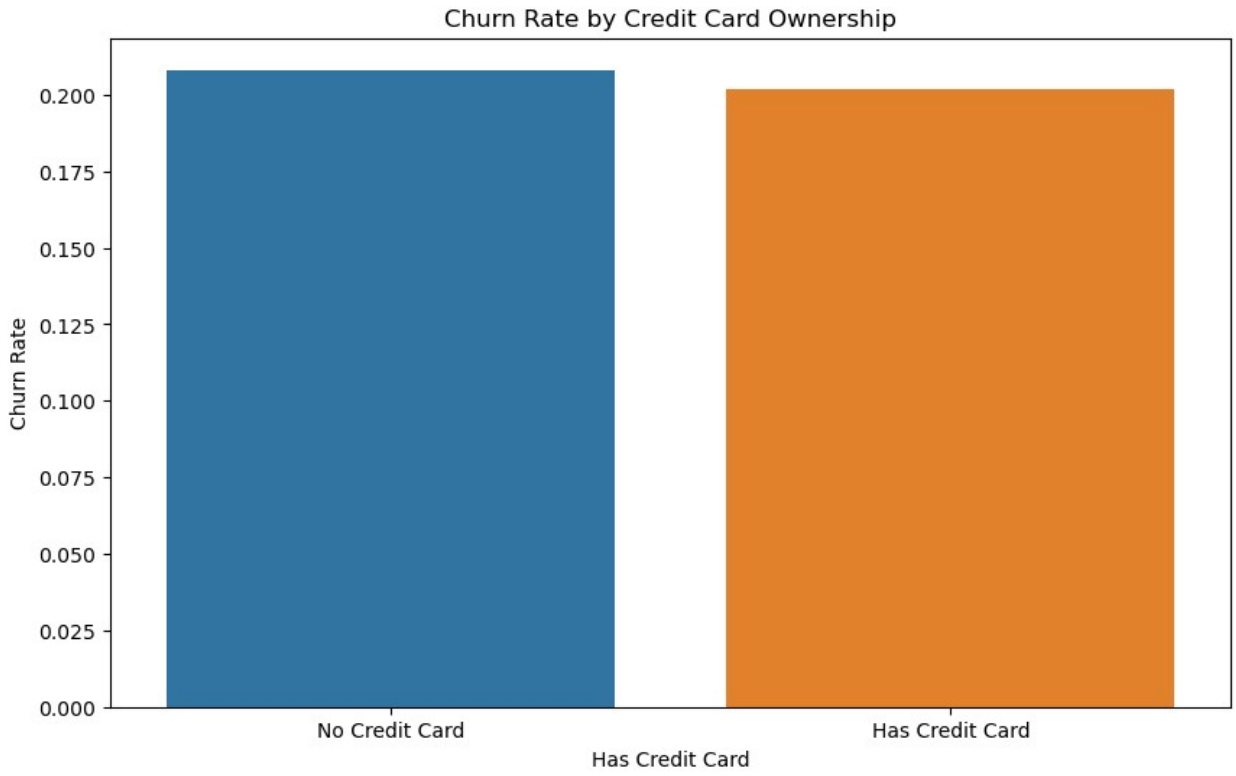
```
from scipy.stats import mannwhitneyu

# Separate the balance data into churned and not churned
balance_churned = data[data['Exited'] == 1]['Balance']
balance_not_churned = data[data['Exited'] == 0]['Balance']

# Mann-Whitney U test
stat, p_value = mannwhitneyu(balance_churned, balance_not_churned)
formatted_p_value = f"{p_value:.3f}"

print("Mann-Whitney U test statistic for balance and churn:", stat)
print("P-value for balance and churn:", formatted_p_value)

if p_value < 0.05:
    print("Reject the null hypothesis: There is a significant
difference in balances between customers who churned and those who
didn't.")
else:
    print("Fail to reject the null hypothesis: There is no significant
difference in balances between customers who churned and those who
didn't.")
```

```
Mann-Whitney U test statistic for balance and churn: 9373909.5
P-value for balance and churn: 0.000
Reject the null hypothesis: There is a significant difference in
balances between customers who churned and those who didn't.
```

credit card ownership

```
# Descriptive Analysis
credit_card_churn_rate = data.groupby('HasCrCard')['Exited'].mean()
print("Churn rate by credit card ownership:\n",
credit_card_churn_rate)

# Visualization
plt.figure(figsize=(10,6))
sns.barplot(x=[0, 1], y=credit_card_churn_rate.values)
plt.title('Churn Rate by Credit Card Ownership')
plt.xlabel('Has Credit Card')
plt.ylabel('Churn Rate')
plt.xticks([0, 1], ['No Credit Card', 'Has Credit Card'])
plt.show()
```

```
Churn rate by credit card ownership:
 HasCrCard
0    0.208149
1    0.201984
Name: Exited, dtype: float64

E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1765: FutureWarning:
unique with argument that is not not a Series, Index, ExtensionArray,
or np.ndarray is deprecated and will raise in a future version.
  order = pd.unique(vector)
```



H0: There is no association between has credit card and customer churn. In other words, the churn rate is independent of has credit card. H1: There is an association between has credit card and customer churn. In other words, the churn rate depends on has credit card.

```
from scipy.stats import chi2_contingency

# Contingency table
contingency_crcard = pd.crosstab(data['HasCrCard'], data['Exited'])

# Chi-Square Test
chi2_crcard, p_crcard, dof_crcard, expected_crcard =
chi2_contingency(contingency_crcard)
formatted_p_crcard = f"{p_crcard:.3f}"

print("Chi-square statistic for credit card ownership and churn:",
chi2_crcard)
print("P-value for credit card ownership and churn:",
formatted_p_crcard)

if p_crcard < 0.05:
    print("Reject the null hypothesis: There is a significant
association between credit card ownership and customer churn.")
else:
    print("Fail to reject the null hypothesis: There is no significant
association between credit card ownership and customer churn.")
```

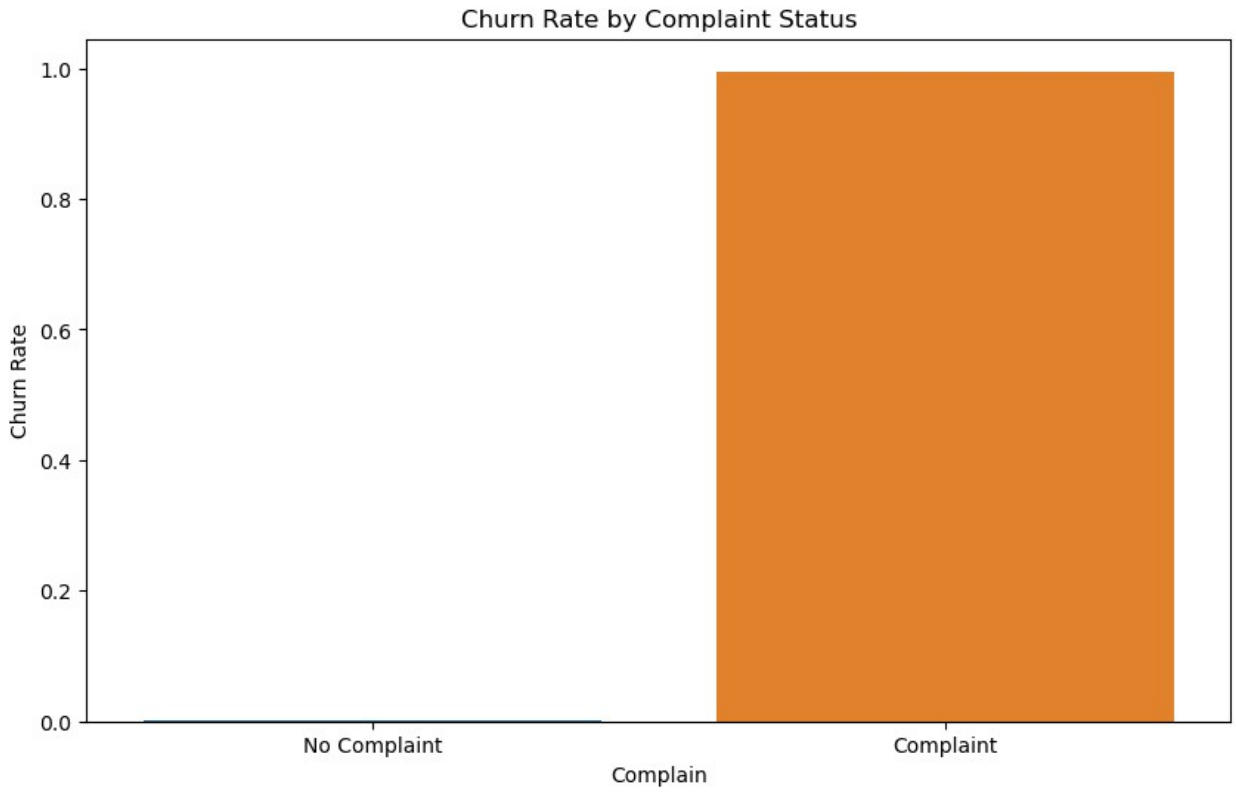
```
Chi-square statistic for credit card ownership and churn:
0.4494039375253385
P-value for credit card ownership and churn: 0.503
Fail to reject the null hypothesis: There is no significant
association between credit card ownership and customer churn.
```

CUSTOMER SATISFACTION AND FEEDBACK

Complaint Analysis

```
# Descriptive Analysis
complaint_churn_rate = data.groupby('Complain')['Exited'].mean()
print("Churn rate by complaint status:\n", complaint_churn_rate)

# Visualization
plt.figure(figsize=(10,6))
sns.barplot(x=[0, 1], y=complaint_churn_rate.values)
plt.title('Churn Rate by Complaint Status')
plt.xlabel('Complain')
plt.ylabel('Churn Rate')
plt.xticks([0, 1], ['No Complaint', 'Complaint'])
plt.show()
```

```
Churn rate by complaint status:
 Complain
0    0.000503
1    0.995108
Name: Exited, dtype: float64

E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1765: FutureWarning:
unique with argument that is not not a Series, Index, ExtensionArray,
or np.ndarray is deprecated and will raise in a future version.
  order = pd.unique(vector)
```



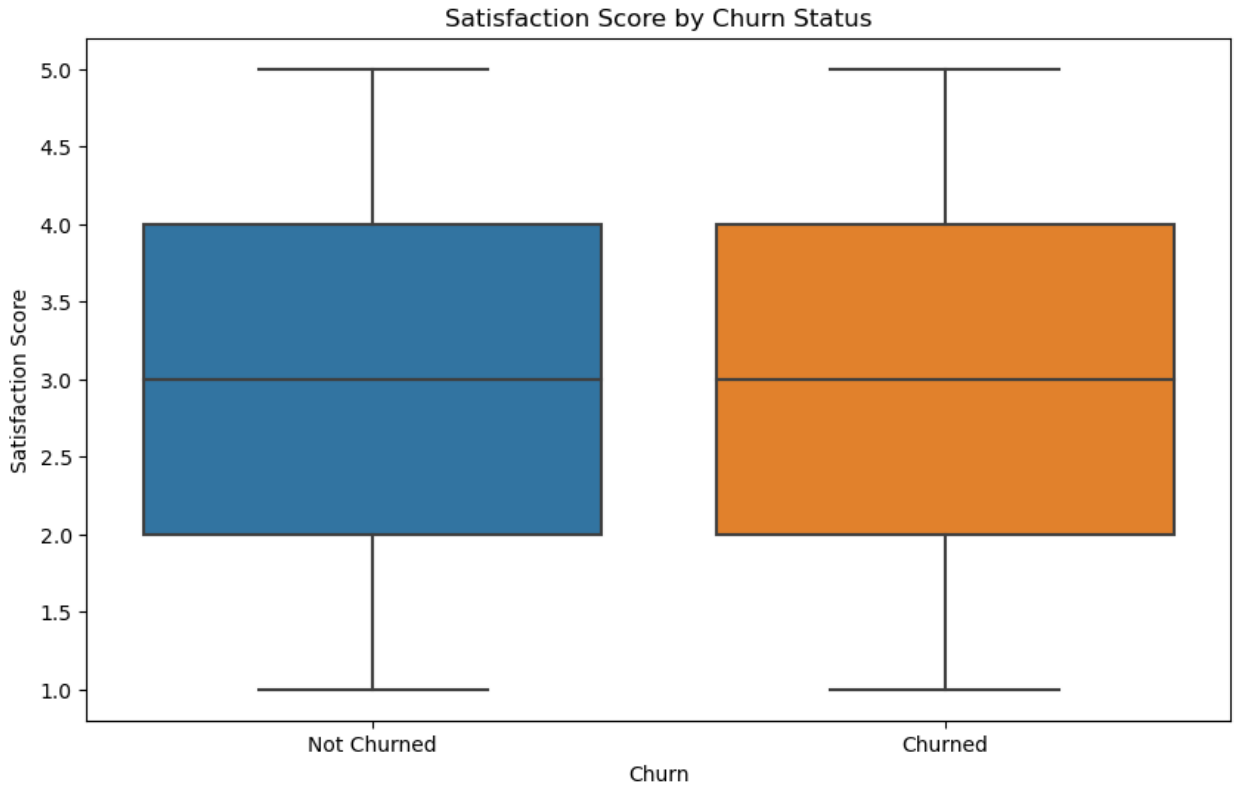
H0: There is no association between complaint and customer churn. In other words, the churn rate is independent of complaint. H1: There is an association between complaint and customer churn. In other words, the churn rate depends on complaint.

```
from scipy.stats import chi2_contingency

# Contingency table
contingency_complain = pd.crosstab(data['Complain'], data['Exited'])

# Chi-Square Test
chi2_complain, p_complain, dof_complain, expected_complain =
chi2_contingency(contingency_complain)
formatted_p_complain = f"{p_complain:.3f}"

print("Chi-square statistic for complaint status and churn:",
chi2_complain)
print("P-value for complaint status and churn:", formatted_p_complain)

if p_complain < 0.05:
    print("Reject the null hypothesis: There is a significant
association between complaints and customer churn.")
else:
    print("Fail to reject the null hypothesis: There is no significant
association between complaints and customer churn.")
```

```
Chi-square statistic for complaint status and churn: 9907.907035880155
P-value for complaint status and churn: 0.000
Reject the null hypothesis: There is a significant association between complaints and customer churn.
```

satisfaction and churn

```
# Visualization
plt.figure(figsize=(10,6))
sns.boxplot(x='Exited', y='Satisfaction Score', data=data)
plt.title('Satisfaction Score by Churn Status')
plt.xlabel('Churn')
plt.ylabel('Satisfaction Score')
plt.xticks([0, 1], ['Not Churned', 'Churned'])
plt.show()
```



H0: There is no association between satisfaction and customer churn. In other words, the churn rate is independent of satisfaction. H1: There is an association between satisfaction and customer churn. In other words, the churn rate depends on satisfaction.

```
# Separate the satisfaction scores into churned and not churned
satisfaction_churned = data[data['Exited'] == 1]['Satisfaction Score']
satisfaction_not_churned = data[data['Exited'] == 0]['Satisfaction Score']
```

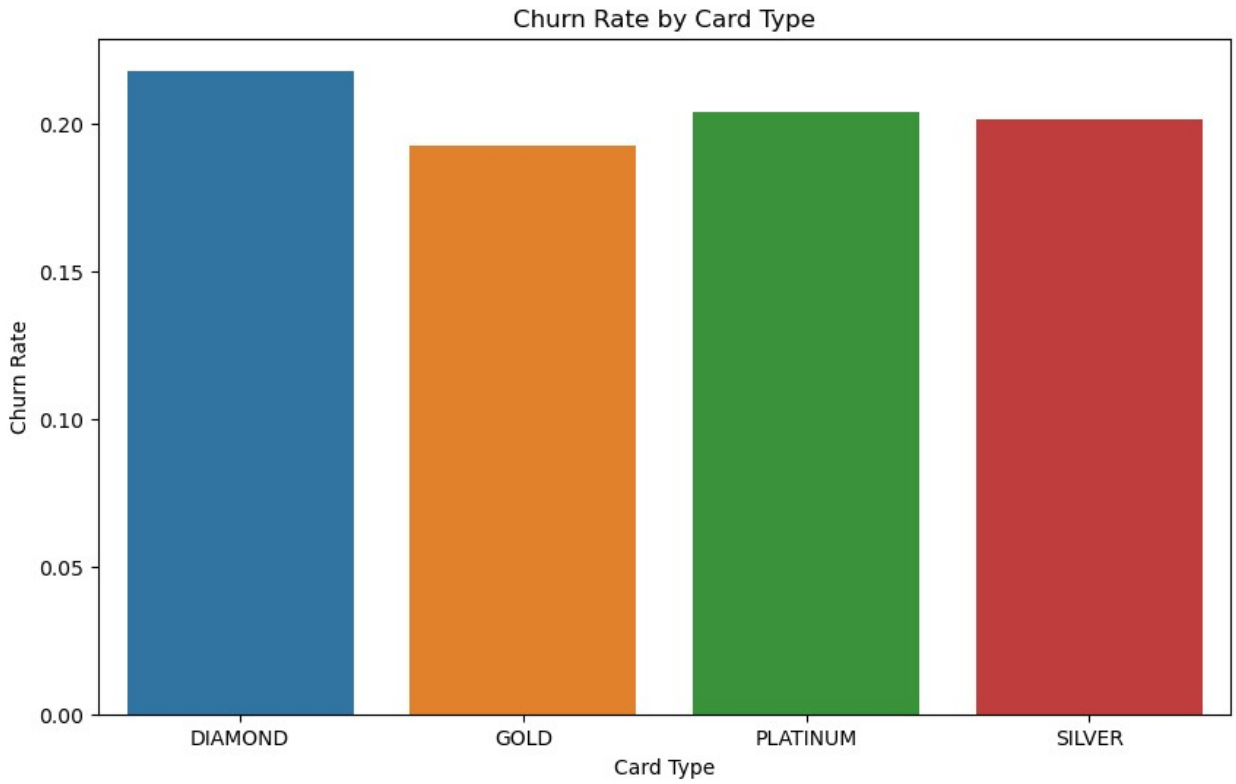
```
# Mann-Whitney U test
stat_satisfaction, p_value_satisfaction =
mannwhitneyu(satisfaction_churned, satisfaction_not_churned)
formatted_p_value_satisfaction = f"{p_value_satisfaction:.3f}"

print("Mann-Whitney U test statistic for satisfaction score and
churn:", stat_satisfaction)
print("P-value for satisfaction score and churn:",
formatted_p_value_satisfaction)

if p_value_satisfaction < 0.05:
    print("Reject the null hypothesis: There is a significant
difference in satisfaction scores between customers who churned and
those who didn't.")
else:
    print("Fail to reject the null hypothesis: There is no significant
difference in satisfaction scores between customers who churned and
those who didn't.")
```

```
Mann-Whitney U test statistic for satisfaction score and churn:
8046002.0
P-value for satisfaction score and churn: 0.555
Fail to reject the null hypothesis: There is no significant difference
in satisfaction scores between customers who churned and those who
didn't.
```

CARD USAGE ANALYSIS

```
# Calculate churn rates for different card types
card_type_churn_rate = data.groupby('Card Type')['Exited'].mean()
print("Churn rate by card type:\n", card_type_churn_rate)

# Visualization
plt.figure(figsize=(10,6))
sns.barplot(x=card_type_churn_rate.index,
y=card_type_churn_rate.values)
plt.title('Churn Rate by Card Type')
plt.xlabel('Card Type')
plt.ylabel('Churn Rate')
plt.show()
```

```
Churn rate by card type:
 Card Type
DIAMOND     0.217790
GOLD        0.192646
PLATINUM    0.203607
SILVER      0.201122
Name: Exited, dtype: float64
```



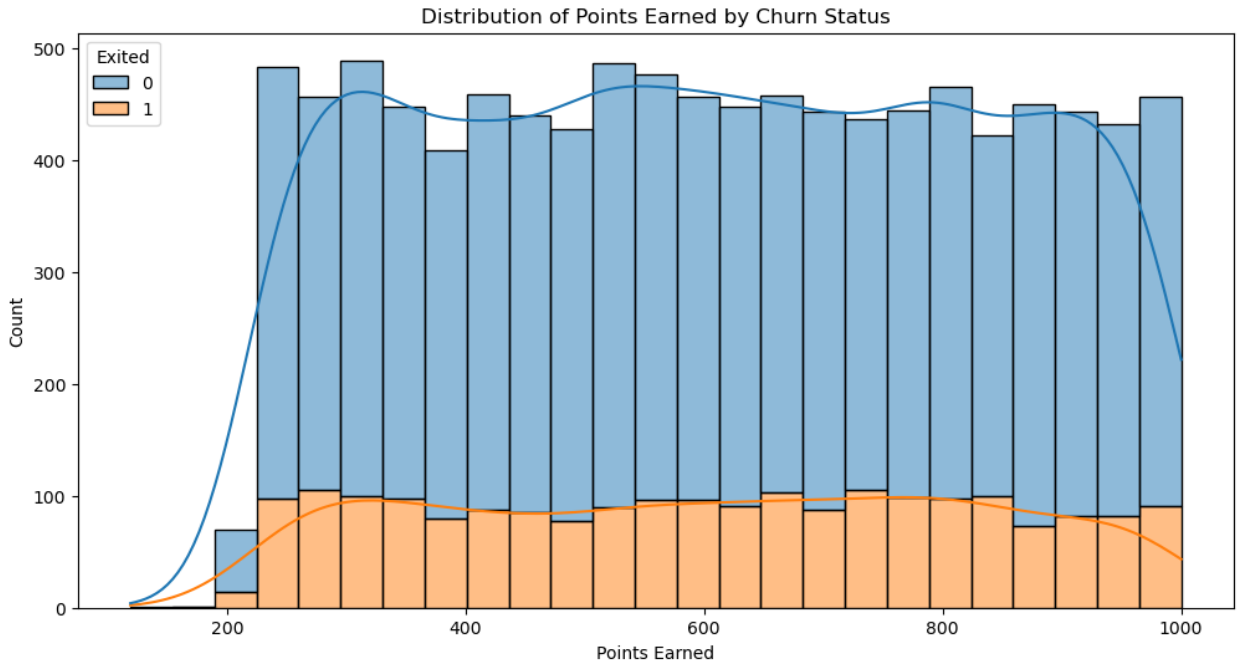

```
# Contingency table
contingency_card_type = pd.crosstab(data['Card Type'], data['Exited'])

# Chi-Square Test
chi2_card_type, p_card_type, dof_card_type, ex_card_type =
chi2_contingency(contingency_card_type)

print(f"Chi-Square test statistic: {chi2_card_type:.4f}")
print(f"P-value: {p_card_type:.4f}")

if p_card_type < 0.05:
    print("Reject the null hypothesis: There is a significant
difference in card type and between customers who churned and those
who didn't.")
else:
    print("Fail to reject the null hypothesis: There is no significant
difference in card type between customers who churned and those who
didn't.")
```

```
Chi-Square test statistic: 5.0532
P-value: 0.1679
Fail to reject the null hypothesis: There is no significant difference
in card type between customers who churned and those who didn't.
```

Loyalty Points analysis

```
# Distribution of points earned by churn status
plt.figure(figsize=(12,6))
sns.histplot(data=data, x='Point Earned', hue='Exited', kde=True,
multiple='stack')
plt.title('Distribution of Points Earned by Churn Status')
plt.xlabel('Points Earned')
plt.ylabel('Count')
plt.show()
```

```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



• H0: There is no association between loyalty points earned and customer churn. In other words, the churn rate is independent of loyalty points earned. • H1: There is an association between loyalty points earned and customer churn. In other words, the churn rate loyalty points earned.

```
from scipy.stats import ttest_ind
# Points earned by churn status
points_churned = data[data['Exited'] == 1]['Point Earned']
points_not_churned = data[data['Exited'] == 0]['Point Earned']

# T-test
t_stat, p_value = ttest_ind(points_churned, points_not_churned)

print(f"T-test statistic: {t_stat:.4f}")
```

```
print(f"P-value: {p_value:.4f}")

if p_value < 0.05:
    print("Reject the null hypothesis: There is a significant
difference in points earned and between customers who churned and
those who didn't.")
else:
    print("Fail to reject the null hypothesis: There is no significant
difference in points earned and between customers who churned and
those who didn't.")
```

```
T-test statistic: -0.4628
P-value: 0.6435
Fail to reject the null hypothesis: There is no significant difference
in points earned and between customers who churned and those who
didn't.
```

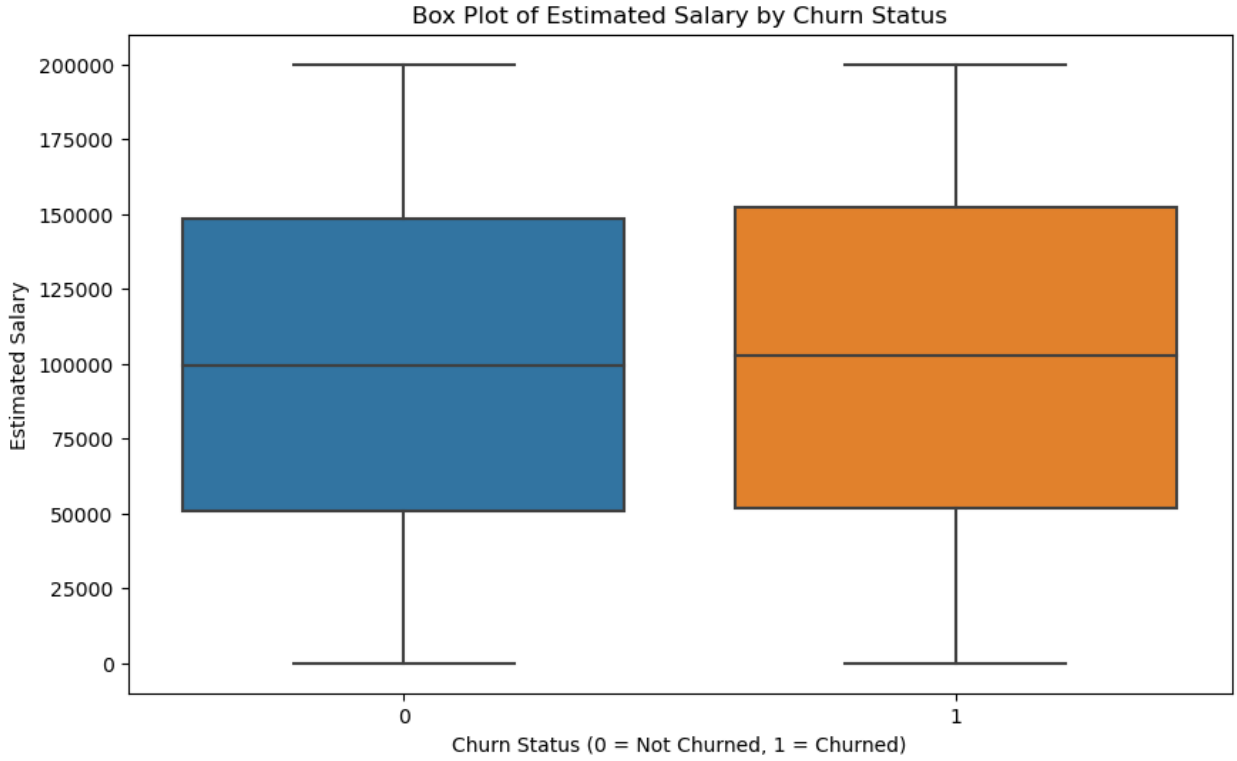
SALARY ANALYSIS

```
# Summary statistics for estimated salary by churn status
salary_stats = data.groupby('Exited')['EstimatedSalary'].describe()
print(salary_stats)
```

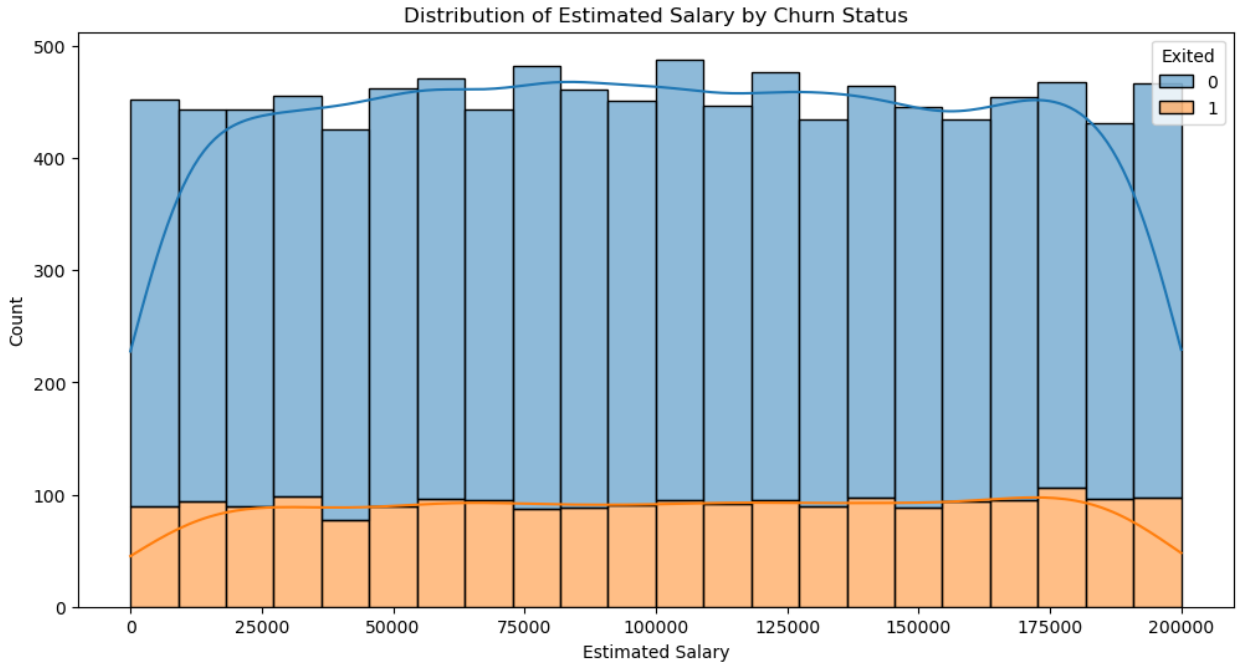
```
         count           mean           std    min        25%
50%  \
Exited

0       7962.0   99726.853141  57399.956717  90.07  50783.295
99620.355
1       2038.0  101509.908783  57932.623392  11.58  51924.020
102489.335

                75%        max
Exited
0       148602.4450  199992.48
1       152443.8575  199808.10
```

```
# Box plot
plt.figure(figsize=(10,6))
sns.boxplot(x='Exited', y='EstimatedSalary', data=data)
plt.title('Box Plot of Estimated Salary by Churn Status')
plt.xlabel('Churn Status (0 = Not Churned, 1 = Churned)')
plt.ylabel('Estimated Salary')
plt.show()

# Histogram
plt.figure(figsize=(12,6))
sns.histplot(data=data, x='EstimatedSalary', hue='Exited', kde=True,
multiple='stack')
plt.title('Distribution of Estimated Salary by Churn Status')
```

```
plt.xlabel('Estimated Salary')
plt.ylabel('Count')
plt.show()
```




```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
# Estimated salary for churned and non-churned customers
salary_churned = data[data['Exited'] == 1]['EstimatedSalary']
salary_not_churned = data[data['Exited'] == 0]['EstimatedSalary']

# T-test
t_stat_salary, p_value_salary = ttest_ind(salary_churned,
salary_not_churned)

print(f"T-test statistic: {t_stat_salary:.4f}")
print(f"P-value: {p_value_salary:.4f}")

if p_value_salary < 0.05:
    print("Reject the null hypothesis: There is a significant
difference in estimated salary and between customers who churned and
those who didn't.")
else:
    print("Fail to reject the null hypothesis: There is no significant
difference in estimated salary and between customers who churned and
those who didn't.")
```

```
T-test statistic: 1.2489
P-value: 0.2117
Fail to reject the null hypothesis: There is no significant difference
in estimated salary and between customers who churned and those who
didn't.
```

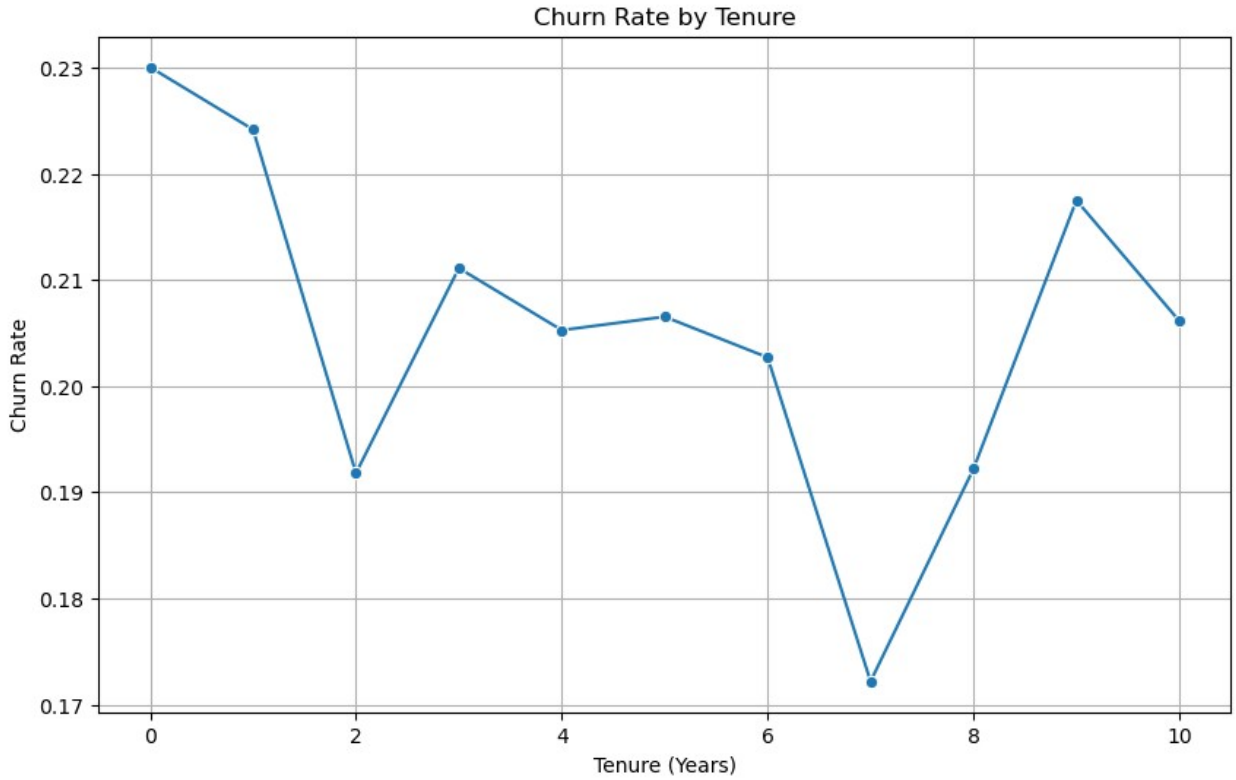
```
# Calculate churn rate for each tenure level
tenure_churn_rate = data.groupby('Tenure')
['Exited'].mean().reset_index()
```

```
tenure_churn_rate.columns = ['Tenure', 'ChurnRate']
print(tenure_churn_rate)
```

```
    Tenure  ChurnRate
0        0   0.230024
1        1   0.224155
2        2   0.191794
3        3   0.211100
4        4   0.205258
5        5   0.206522
6        6   0.202689
7        7   0.172179
8        8   0.192195
9        9   0.217480
10      10   0.206122
```

```
# Line plot of churn rate by tenure
plt.figure(figsize=(10,6))
sns.lineplot(x='Tenure', y='ChurnRate', data=tenure_churn_rate,
marker='o')
plt.title('Churn Rate by Tenure')
plt.xlabel('Tenure (Years)')
plt.ylabel('Churn Rate')
plt.grid(True)
plt.show()
```

```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
from scipy.stats import f_oneway

# ANOVA test
groups = [data['Exited'][data['Tenure'] == tenure] for tenure in
data['Tenure'].unique()]
f_stat, p_value = f_oneway(*groups)

print(f"ANOVA F-statistic: {f_stat:.4f}")
print(f"P-value: {p_value:.4f}")

if p_value < 0.05:
    print("Reject the null hypothesis: There is a significant
difference in tenure and between customers who churned and those who
didn't.")
else:
    print("Fail to reject the null hypothesis: There is no significant
difference in tenure and between customers who churned and those who
didn't.")
```

```
ANOVA F-statistic: 1.4063
P-value: 0.1703
Fail to reject the null hypothesis: There is no significant difference
in tenure and between customers who churned and those who didn't.
```