In [4]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import networkx as nx
from collections import defaultdict
import plotly.graph_objects as go
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from xgboost import XGBClassifier
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

In [5]:
```
data = pd.read_excel("/Users/zheguan/CWR_fig/Customer_Churn_Data_Large.xlsx"
```

In [6]:
```
data
```

Out[6]:
```
{'Customer_Demographics':      CustomerID  Age Gender MaritalStatus IncomeL
evel
 0             1   62      M        Single         Low
 1             2   65      M       Married         Low
 2             3   18      M        Single         Low
 3             4   21      M       Widowed         Low
 4             5   21      M      Divorced      Medium
 ..          ...  ...    ...           ...         ...
 995         996   54      F        Single         Low
 996         997   19      M       Widowed        High
 997         998   47      M       Married         Low
 998         999   23      M       Widowed        High
 999        1000   34      M       Widowed         Low

 [1000 rows x 5 columns],
 'Transaction_History':       CustomerID  TransactionID TransactionDate  Am
ountSpent ProductCategory
 0              1           7194      2022-03-27       416.50     Electroni
cs
 1              2           7250      2022-08-08        54.96        Clothi
ng
 2              2           9660      2022-07-25       197.50     Electroni
cs
 3              2           2998      2022-01-25       101.31       Furnitu
re
 4              2           1228      2022-07-24       397.37        Clothi
ng
 ...          ...            ...             ...          ...
...
 5049        1000           2724      2022-09-08       232.06       Groceri
es
 5050        1000           2917      2022-12-13       324.98           Boo
ks
 5051        1000           2979      2022-06-15       375.34       Groceri
es
 5052        1000           8594      2022-04-08       166.73           Boo
ks
 5053        1000           5529      2022-11-23        93.73       Furnitu
re

 [5054 rows x 5 columns],
 'Customer_Service':       CustomerID  InteractionID InteractionDate Intera
ctionType  \
 0              1           6363      2022-03-31         Inquiry
 1              2           3329      2022-03-17         Inquiry
 2              3           9976      2022-08-24         Inquiry
 3              4           7354      2022-11-18         Inquiry
 4              4           5393      2022-07-03         Inquiry
 ...          ...            ...             ...             ...
 997          990           3671      2022-10-25       Complaint
 998          992           2114      2022-09-29        Feedback
 999          994           3087      2022-07-02       Complaint
 1000         994           8508      2022-05-14       Complaint
 1001         995           9101      2022-05-02         Inquiry
```

olutionStatus

```
 0           Resolved
 1           Resolved
 2           Resolved
 3           Resolved
 4         Unresolved
 ...              ...
 997       Unresolved
 998       Unresolved
 999       Unresolved
 1000      Unresolved
 1001        Resolved

 [1002 rows x 5 columns],
 'Online_Activity':      CustomerID LastLoginDate  LoginFrequency ServiceUs
age
 0             1    2023-10-21              34   Mobile App
 1             2    2023-12-05               5      Website
 2             3    2023-11-15               3      Website
 3             4    2023-08-25               2      Website
 4             5    2023-10-27              41      Website
 ..          ...           ...             ...          ...
 995         996    2023-01-29              38   Mobile App
 996         997    2023-04-01               5   Mobile App
 997         998    2023-07-10              47      Website
 998         999    2023-01-08              23      Website
 999        1000    2023-08-13              22   Mobile App

 [1000 rows x 4 columns],
 'Churn_Status':      CustomerID  ChurnStatus
 0             1            0
 1             2            1
 2             3            0
 3             4            0
 4             5            0
 ..          ...          ...
 995         996            0
 996         997            0
 997         998            0
 998         999            0
 999        1000            0

 [1000 rows x 2 columns]}
```

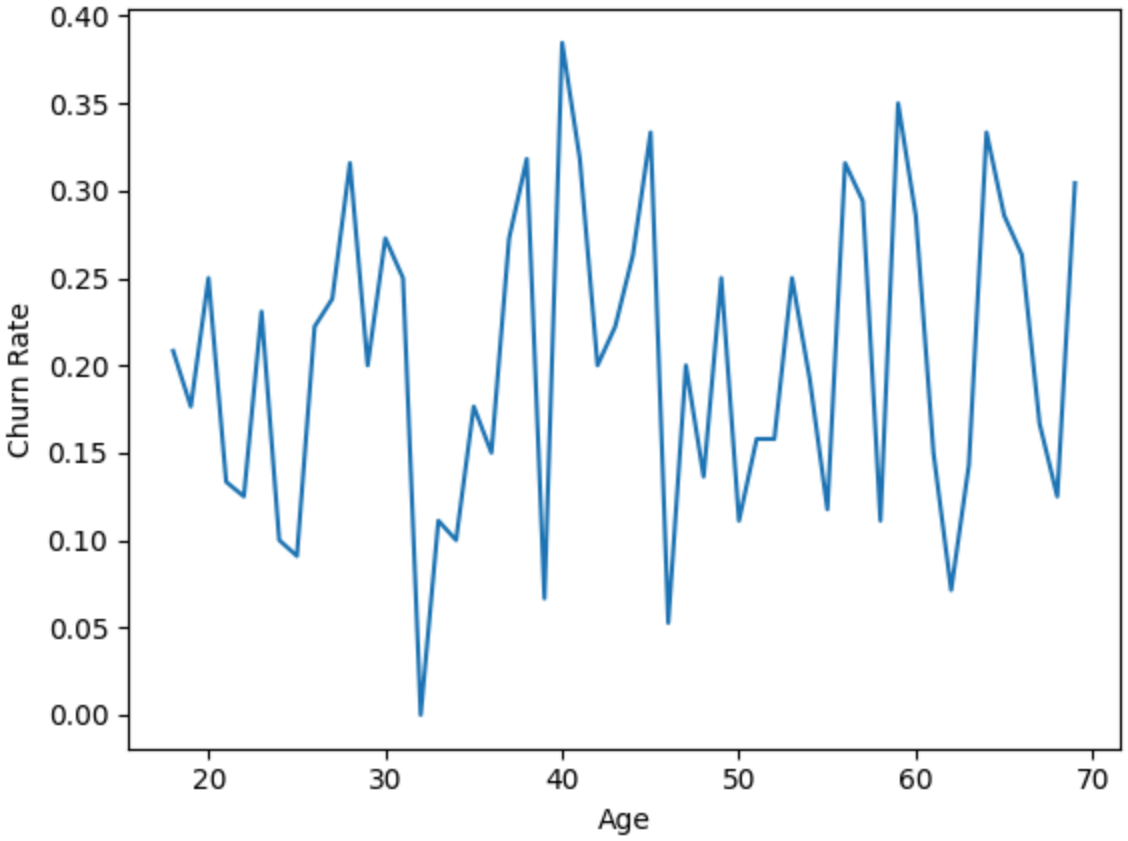
Customer_Demographics Sheet

In [7]:
```
Customer_Demographics = pd.merge(data['Customer_Demographics'], data['Churn_
#train_data, test_data = train_test_split(Customer_Demographics, test_size=0
```

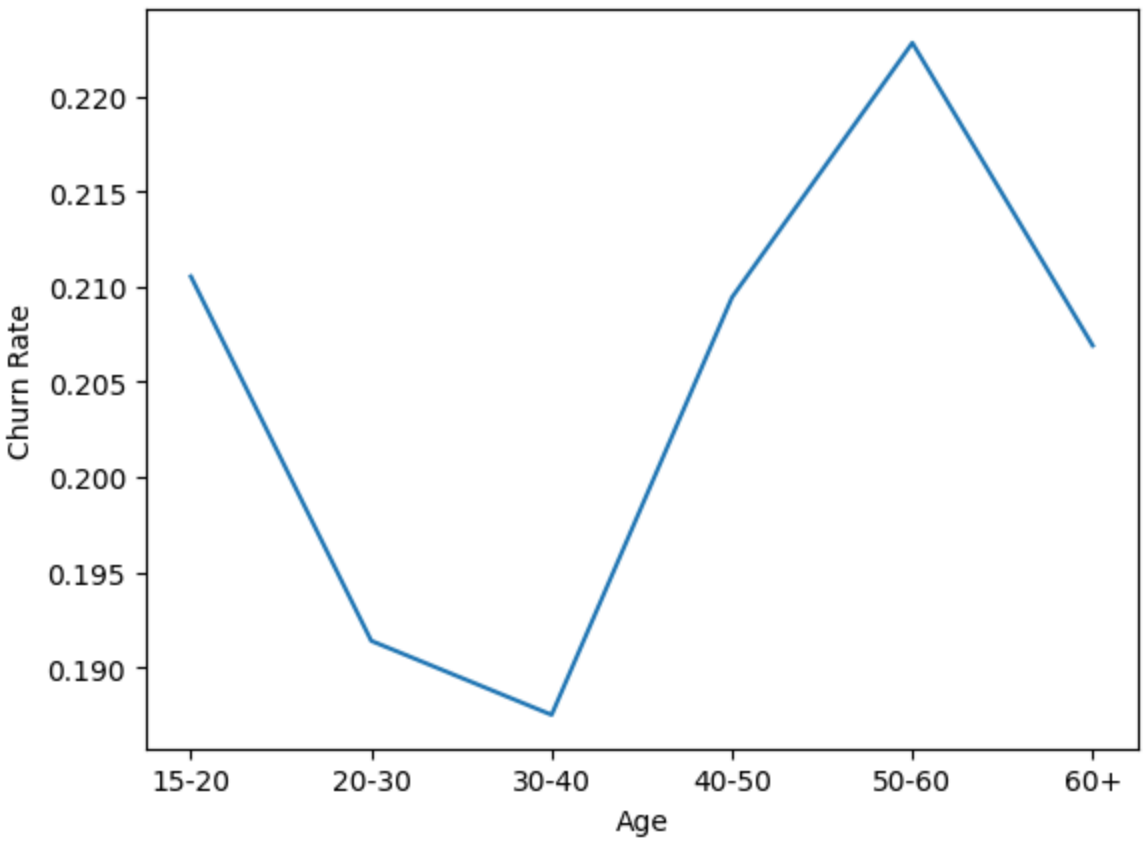
In [8]:
```
Customer_Demographics["Age_range"]= pd.cut(Customer_Demographics["Age"],
                                bins = [15, 20, 30, 40, 50, 60, np.inf],
                                labels = ["15-20", "20-30", "30-40", "40-50",
```

In [9]:
```
Age_distribution = pd.crosstab(Customer_Demographics["Age"],Customer_Demogra
```

In [16]:
```
plt.plot(Age_distribution["Age"],Age_distribution[1])
plt.xlabel("Age")
plt.ylabel("Churn Rate")
plt.savefig("Plots/age_churn_rate.png", dpi=300, bbox_inches='tight')
plt.show()
```



In [15]:
```
Agerange_distribution = pd.crosstab(Customer_Demographics["Age_range"],Custo
plt.plot(Agerange_distribution["Age_range"],Agerange_distribution[1])
plt.xlabel("Age")
plt.ylabel("Churn Rate")
plt.savefig("Plots/agerange_churn_rate.png", dpi=300, bbox_inches='tight')
plt.show()
```



In [18]:
```
average_churn_rate = sum(Customer_Demographics['ChurnStatus'])/Customer_Demo
average_churn_rate
```

Out[18]: 0.204

In [19]:
```
def cal_Zscore(Customer_Demographics, input_col, target_col):
    average_churn_rate = sum(Customer_Demographics[target_col])/Customer_Dem

    Distribution = pd.crosstab(Customer_Demographics[input_col],Customer_Dem
    Distribution['number'] = Distribution[input_col].map(Customer_Demographi
    Distribution["sigma"] = Distribution[0]*Distribution[1]/Distribution['nu
    Distribution["Zscore_"+input_col+"diff"] = (Distribution[1]-average_chur

    df = pd.merge(Customer_Demographics, Distribution[[input_col, "Zscore_"+

    return df,Distribution,average_churn_rate
```

In [20]:
```
Gender_Distribution = cal_Zscore(Customer_Demographics, "Gender", "ChurnStat
Age_rangeDistribution = cal_Zscore(Customer_Demographics, "Age_range", "Chur
MaritalStatus_Distribution = cal_Zscore(Customer_Demographics, "MaritalStatu
IncomeLevel_Distribution = cal_Zscore(Customer_Demographics, "IncomeLevel",
```

```
/var/folders/pt/0nf2m3pj1f3b373wsyfc6glh0000gn/T/ipykernel_52421/1435638261.
py:5: 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 th
is warning.
  Distribution['number'] = Distribution[input_col].map(Customer_Demographic
nput_col).size().astype(float))
```

In [21]:
```
distributions = {
    "Gender": Gender_Distribution,
    "Age_range":Age_rangeDistribution,
    "MaritalStatus":MaritalStatus_Distribution,
    "IncomeLevel":IncomeLevel_Distribution
}

def merge_distribution(ori_df, dic):

    data = ori_df

    for key, distribution in dic.items():
        data = pd.merge(data, distribution[[key,"Zscore_"+key+"diff"]], how=

    return data

enhanced_Customer_Demographics = merge_distribution(Customer_Demographics, c
```

In [22]:
```
enhanced_Customer_Demographics= pd.get_dummies(enhanced_Customer_Demographic
enhanced_Customer_Demographics
```

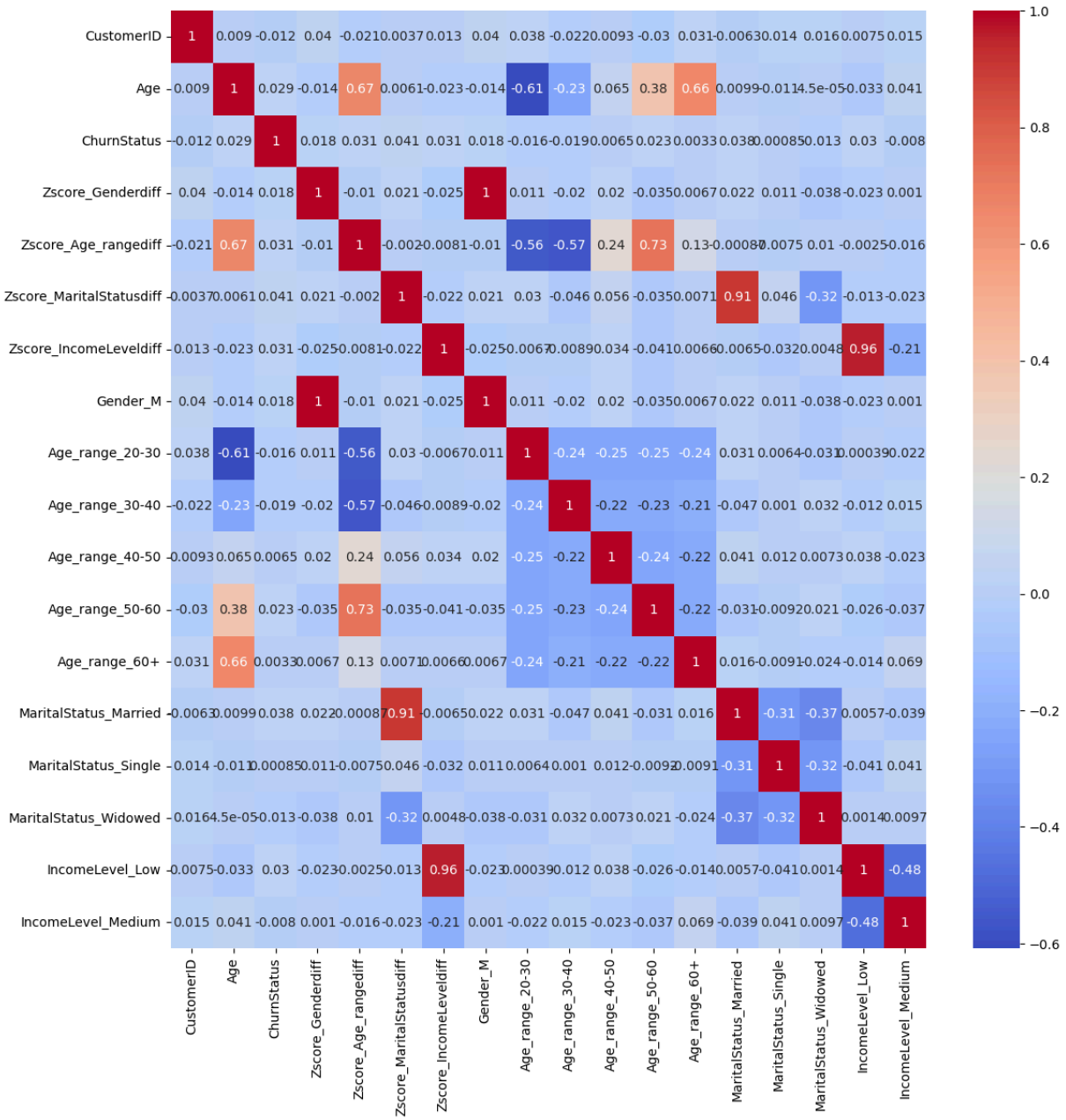
Out[22]:

	CustomerID	Age	ChurnStatus	Zscore_Genderdiff	Zscore_Age_rangediff
0	1	62	0	21.898796	3.071478
1	2	65	1	21.898796	3.071478
2	3	18	0	21.898796	2.238200
3	4	21	0	21.898796	-17.033005
4	5	21	0	21.898796	-17.033005
...	...	...	...	...	...
995	996	54	0	-23.096539	20.951841
996	997	19	0	21.898796	2.238200
997	998	47	0	21.898796	6.257337
998	999	23	0	21.898796	-17.033005
999	1000	34	0	21.898796	-19.062154

1000 rows × 18 columns

In [24]:
```
corr_df = enhanced_Customer_Demographics.corr()

plt.figure(figsize=(13, 13))
sns.heatmap(corr_df, annot=True, cmap='coolwarm')
```

Out[24]: <Axes: >



Cusomer Service sheet

In [25]:
```
Customer_Service = data['Customer_Service']
Customer_Service = pd.merge(Customer_Service, data['Churn_Status'], how = "l
```

In [26]:
```
prob_InteractionType = Customer_Service.groupby(["InteractionType","ChurnSta
prob_InteractionType["churn_rate"] = prob_InteractionType[1]/(prob_Interacti
```

In [27]:
```
prob_InteractionType["sigma"] = prob_InteractionType["churn_rate"]*(1-prob_I
```

In [28]:
```
avg_prob_InteractionType = prob_InteractionType[1].sum() / (prob_Interactior
prob_InteractionType["Zscore_InteractionType"] = (prob_InteractionType["chur
Customer_Service[Customer_Service["ChurnStatus"] == 0].groupby(["Interactior
```

Out[28]:

CustomerID	InteractionType	1	3	4	6	9	11	12	13	14
0	Complaint	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	1.0
1	Feedback	NaN	NaN	NaN	1.0	NaN	2.0	1.0	NaN	1.0
2	Inquiry	1.0	1.0	2.0	NaN	1.0	NaN	1.0	NaN	NaN

3 rows × 526 columns

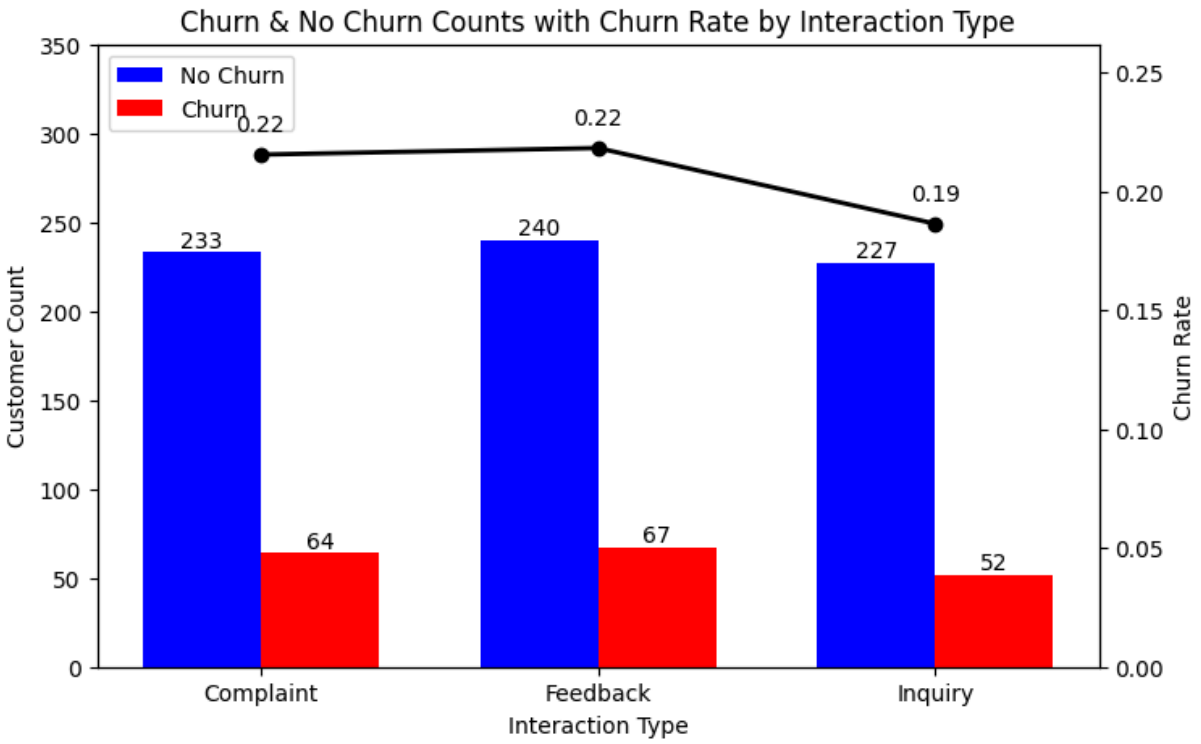
In [29]:
```
prob_InteractionType
```

Out[29]:

ChurnStatus	InteractionType	0	1	churn_rate	sigma	Zscore_Interactio
0	Complaint	233	64	0.215488	0.009809	0.8
1	Feedback	240	67	0.218241	0.009737	1.1
2	Inquiry	227	52	0.186380	0.009079	-2.2

In [36]:
```
import matplotlib.pyplot as plt
import numpy as np


interaction_types = prob_InteractionType["InteractionType"]
no_churn_counts = prob_InteractionType[0]
churn_counts = prob_InteractionType[1]
churn_rates = prob_InteractionType["churn_rate"]

x = np.arange(len(interaction_types))

fig, ax1 = plt.subplots(figsize=(8, 5))
plt.ylim(0,350)
# Bar Chart for Churn & No Churn
bar_width = 0.35
ax1.bar(x - bar_width/2, no_churn_counts, bar_width, label="No Churn", color
ax1.bar(x + bar_width/2, churn_counts, bar_width, label="Churn", color='red'

ax1.set_xlabel("Interaction Type")
ax1.set_ylabel("Customer Count")
ax1.set_xticks(x)
ax1.set_xticklabels(interaction_types)
ax1.legend(loc="upper left")


ax2 = ax1.twinx()
ax2.plot(x, churn_rates, color='black', marker='o', linestyle='-', linewidth

ax2.set_ylabel("Churn Rate")
ax2.set_ylim(0, max(churn_rates) * 1.2)

# Show values on bars
for i in range(len(x)):
    ax1.text(x[i] - bar_width/2, no_churn_counts.iloc[i] + 3, str(no_churn_c
    ax1.text(x[i] + bar_width/2, churn_counts.iloc[i] + 3, str(churn_counts.
```

```
    ax2.text(x[i], churn_rates.iloc[i] + 0.01, f"{churn_rates.iloc[i]:.2f}",

plt.title("Churn & No Churn Counts with Churn Rate by Interaction Type")
plt.savefig("Plots/churn_rate_interactiontype.png", dpi=300, bbox_inches='ti
plt.show()
```



In [37]:
```
Customer_Service["Status"] = Customer_Service["InteractionType"] +"_"+ Custc
```

In [38]:
```
final_status = (Customer_Service.groupby("CustomerID", group_keys=False)[["C
                .apply(lambda x : x.iloc[-1] )
                .reset_index()
                .rename(columns={"ChurnStatus": "Status"}))
```

In [40]:
```
combined_Customer_Service = (pd.concat([Customer_Service,final_status])
                             .sort_values(by=["CustomerID","InteractionDate"
                            )
combined_Customer_Service["Status"] = combined_Customer_Service["Status"].re
```

In [41]:
```
Customer_Service_enhanced = Customer_Service.sort_values(by=["CustomerID", "
    .groupby("CustomerID").agg(
        First_Interation=('InteractionDate','min'),
        Last_Interation=('InteractionDate','max'),
        Last_Status = ('Status','last')
    )

Customer_Service_enhanced['Duration_Interation'] = \
    (Customer_Service_enhanced['Last_Interation']  - Customer_Service_enhanc
```

In [42]:
```
Customer_Service["InteractionDate"] = pd.to_datetime(Customer_Service["Inter
Customer_Service = Customer_Service.sort_values(by=["CustomerID", "Interacti

eractions = Customer_Service.groupby(["CustomerID", "InteractionDat
```

```
    Interaction_Count=("InteractionID", "count")
).reset_index()

daily_interactions["Rolling_Interaction_Count"] = daily_interactions.groupby
    .rolling(window=7, min_periods=1).sum().reset_index(level=0, drop=True)

max_interaction_7d = daily_interactions.loc[daily_interactions.groupby("Cust

test = max_interaction_7d.merge(data["Churn_Status"], how = "left" ,on = "Cu
test[test['ChurnStatus'] == 1].head(10)
```

Out[42]:

	CustomerID	InteractionDate	Interaction_Count	Rolling_Interaction_Count
1	2	2022-03-17	1	1.0
5	8	2022-09-13	1	2.0
12	16	2022-09-08	1	2.0
14	18	2022-11-08	1	1.0
21	26	2022-12-12	1	1.0
27	33	2022-05-30	1	1.0
34	41	2022-02-26	1	1.0
43	61	2022-04-10	1	1.0
46	68	2022-06-06	1	2.0
48	71	2022-04-13	1	2.0

In [43]:

```
Customer_Service.groupby("CustomerID")['InteractionType'].value_counts().uns
Customer_Service.groupby("CustomerID")['ResolutionStatus'].value_counts().ur
Customer_Service.groupby("CustomerID")['Status'].value_counts().unstack()
interaction_statistics = pd.concat([Customer_Service.groupby("CustomerID")['
```

In [44]:

```
chains = combined_Customer_Service.groupby("CustomerID")["Status"].apply(lis
transform = defaultdict(lambda: defaultdict(int))

for i in chains:
    #print(i[0])
    for k in range(len(i)-1):
        src = i[k]
        des = i[k+1]
        transform[src][des] += 1
        #print(src,des)

transform_count = pd.DataFrame(transform).T
transform_prob = transform_count.div(transform_count.sum(axis=1), axis=0)
transform_prob
transform_prob.loc["Retained"] = [1,0,0,0,0,0,0,0]
transform_prob.loc["Churned"] = [0,1,0,0,0,0,0,0]


expected_order = ["Retained", "Churned", "Inquiry_Resolved", "Inquiry_Unresc
```

```
transform_prob = transform_prob.reindex(index=expected_order, columns=expect
transform_prob
```

Out[44]:

	Retained	Churned	Inquiry_Resolved	Inquiry_Unresolv
Retained	1.000000	0.000000	0.000000	0.0000
Churned	0.000000	1.000000	0.000000	0.0000
Inquiry_Resolved	0.577381	0.130952	0.053571	0.0357
Inquiry_Unresolved	0.546763	0.122302	0.064748	0.0287
Feedback_Resolved	0.507538	0.155779	0.035176	0.0502
Feedback_Unresolved	0.546584	0.142857	0.043478	0.0372
Complaint_Resolved	0.512821	0.115385	0.032051	0.0384
Complaint_Unresolved	0.463687	0.178771	0.094972	0.0614

In [45]:
```
combined_Customer_Service
```

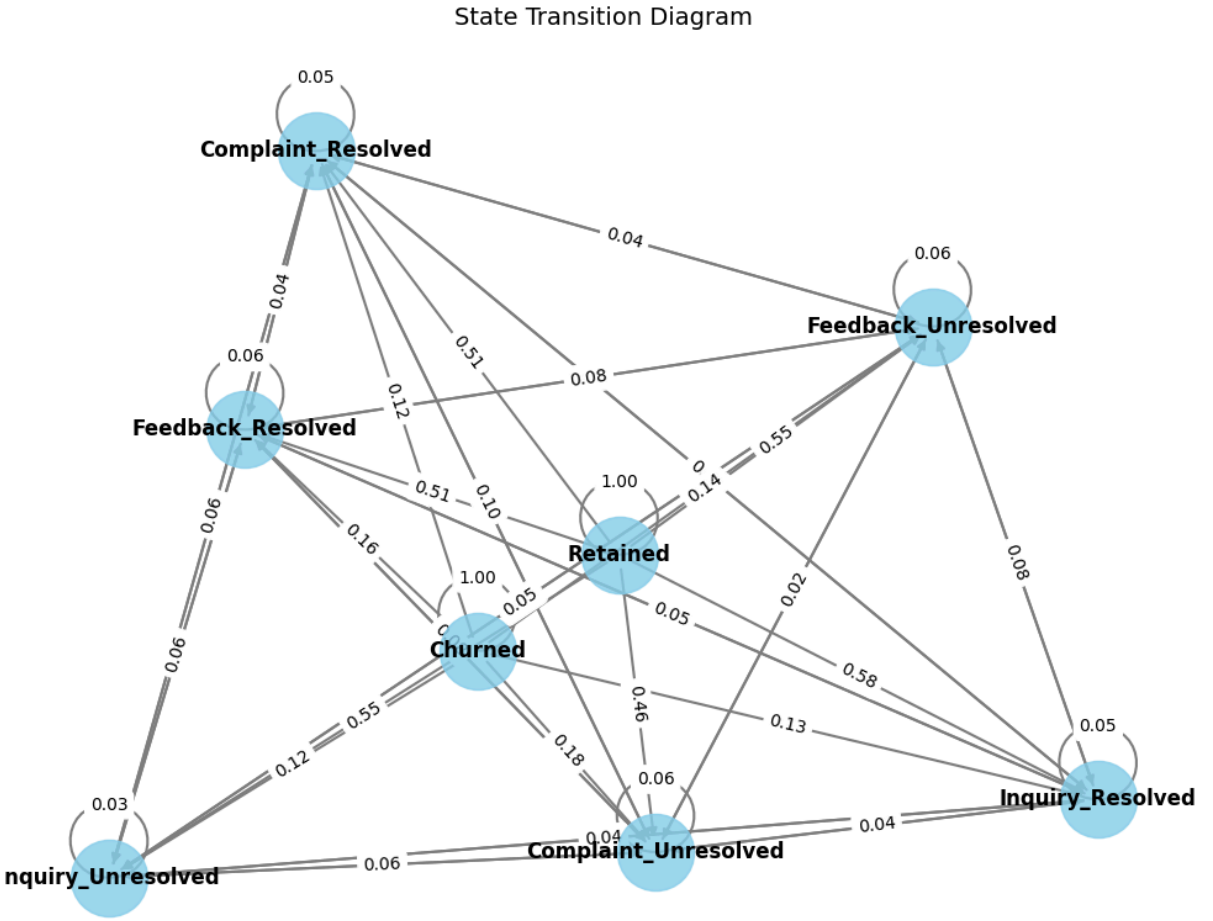
Out[45]:

	CustomerID	InteractionID	InteractionDate	InteractionType	Resolution
0	1	6363.0	2022-03-31	Inquiry	R
0	1	NaN	NaT	NaN	
1	2	3329.0	2022-03-17	Inquiry	R
1	2	NaN	NaT	NaN	
2	3	9976.0	2022-08-24	Inquiry	R
...	...	...	...	...	
1000	994	8508.0	2022-05-14	Complaint	Unr
999	994	3087.0	2022-07-02	Complaint	Unr
666	994	NaN	NaT	NaN	
1001	995	9101.0	2022-05-02	Inquiry	R
667	995	NaN	NaT	NaN	

1670 rows × 7 columns

In [47]:
```
G = nx.DiGraph()


for src in transform_prob.columns:
    for dst in transform_prob.index:
        prob = transform_prob.loc[dst, src]
        if prob > 0:
            G.add_edge(src, dst, weight=prob)


pos = nx.spring_layout(G, seed=42)
```

```
#pos = nx.kamada_kawai_layout(G)
#pos = nx.shell_layout(G)
#pos = nx.circular_layout(G)
#pos = nx.spectral_layout(G)
#pos = nx.multipartite_layout(G, subset_key="layer")


plt.figure(figsize=(10, 8))
nx.draw_networkx_nodes(G, pos, node_size=2000, node_color="skyblue", alpha=0
nx.draw_networkx_edges(G, pos, width=1.5, edge_color="gray", arrows=True)

edge_labels = {(u, v): f"{d['weight']:.2f}" for u, v, d in G.edges(data=True
nx.draw_networkx_labels(G, pos, font_size=12, font_weight="bold")
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_size=10)


plt.title("State Transition Diagram", fontsize=14)
plt.axis("off")
plt.tight_layout()
plt.savefig("Plots/chain_interactiontype.png", dpi=300, bbox_inches='tight')
plt.show()
```



In [48]:
```
labels = list(transform_prob.columns)
source = []
target = []
value = []
```

```
for src_idx, src in enumerate(transform_prob.columns):
    for dst_idx, dst in enumerate(transform_prob.index):
        prob = transform_prob.loc[dst, src]
        if prob > 0:
            target.append(src_idx)
            source.append(len(labels) + dst_idx)
            value.append(prob)


labels += list(transform_prob.index)


fig = go.Figure(go.Sankey(
    node=dict(
        pad=20,
        thickness=20,
        line=dict(color="black", width=0.5),
        label=labels,
        color=["#90EE90", "#FFB6C1", "#FF4500", "#32CD32"] * 2
    ),
    link=dict(
        source=source,
        target=target,
        value=value,
        color="rgba(128, 128, 128, 0.5)"
    )
))

fig.update_layout(
    title_text="State Transition Flow",
    font_size=12,
    height=600
)
fig.show()
```

In [49]:
```
plt.figure(figsize=(12,8))
sns.heatmap(transform_prob.fillna(0), annot=True, cmap='Blues')
#plt.title("State Transition Probability Matrix")
plt.savefig("Plots/transition_matrix.png", dpi=300, bbox_inches='tight')
plt.show()
```

In [50]:
```
import scipy.linalg

P = np.array(transform_prob)

absorbing_states = np.where(np.diag(P) == 1)[0]
transient_states = np.where(np.diag(P) != 1)[0]

Q = P[np.ix_(transient_states, transient_states)]
R = P[np.ix_(transient_states, absorbing_states)]

# N = (I - Q)^(-1)
I = np.eye(Q.shape[0])
N = np.linalg.inv(I - Q)

# B = N @ R
B = N @ R
print("Absorb Prob Matrix B:\n", B)
```

```
Absorb Prob Matrix B:
 [[0.80700758 0.19299242]
 [0.80678326 0.19321674]
 [0.77081632 0.22918368]
 [0.79064724 0.20935276]
 [0.80247131 0.19752869]
 [0.74481898 0.25518102]]
```

In [51]:
```
B
```

Out[51]:
```
array([[0.80700758, 0.19299242],
       [0.80678326, 0.19321674],
       [0.77081632, 0.22918368],
       [0.79064724, 0.20935276],
       [0.80247131, 0.19752869],
       [0.74481898, 0.25518102]])
```

In [52]:
```
Absorb_Matrix = {
    'Inquiry_Resolved': B[0,1],
    'Inquiry_Unresolved': B[1,1],
    'Feedback_Resolved': B[2,1],
    'Feedback_Unresolved': B[3,1],
    'Complaint_Resolved': B[4,1],
    'Complaint_Unresolved': B[5,1]
}

interaction_statistics['Markov_Churn_prob'] = Customer_Service_enhanced['Las
```

In [54]:
```
#interaction_statistics
Customer_Service_enhanced
```

Out[54]:

CustomerID	First_Interation	Last_Interation	Last_Status	Duration_In
1	2022-03-31	2022-03-31	Inquiry_Resolved	
2	2022-03-17	2022-03-17	Inquiry_Resolved	
3	2022-08-24	2022-08-24	Inquiry_Resolved	
4	2022-07-03	2022-11-18	Inquiry_Resolved	
6	2022-05-05	2022-05-05	Feedback_Resolved	
...	...	...	...	
989	2022-08-03	2022-10-07	Complaint_Unresolved	
990	2022-09-17	2022-10-25	Complaint_Unresolved	
992	2022-09-29	2022-09-29	Feedback_Unresolved	
994	2022-05-14	2022-07-02	Complaint_Unresolved	
995	2022-05-02	2022-05-02	Inquiry_Resolved	

668 rows × 4 columns

In [55]:
```
enhanced_Interaction_Customer = enhanced_Customer_Demographics.merge(Custome
                                .merge(interaction_statistics['Markov_Churn_prob
```

In [56]:
```
enhanced_Interaction_Customer
```

Out[56]:

	CustomerID	Age	ChurnStatus	Zscore_Genderdiff	Zscore_Age_rangediff
0	1	62	0	21.898796	3.071478
1	2	65	1	21.898796	3.071478
2	3	18	0	21.898796	2.238200
3	4	21	0	21.898796	-17.033005
4	5	21	0	21.898796	-17.033005
...	...	...	...	...	...
995	996	54	0	-23.096539	20.951841
996	997	19	0	21.898796	2.238200
997	998	47	0	21.898796	6.257337
998	999	23	0	21.898796	-17.033005
999	1000	34	0	21.898796	-19.062154

1000 rows × 20 columns

In [68]:
```
Customer_Service_count = Customer_Service.groupby(["CustomerID","Interactior
```

In [69]:
```
Customer_Service_resolutioncount = Customer_Service.groupby(["CustomerID","F
```

In [57]:
```
enhanced_Interaction_Customer['Duration_Interation'] = enhanced_Interaction_
enhanced_Interaction_Customer['Markov_Churn_prob'] = enhanced_Interaction_Cu
```

In [72]:
```
enhanced_Interaction_Customer = enhanced_Interaction_Customer.merge(Customer
```

In [74]:
```
enhanced_Interaction_Customer
```

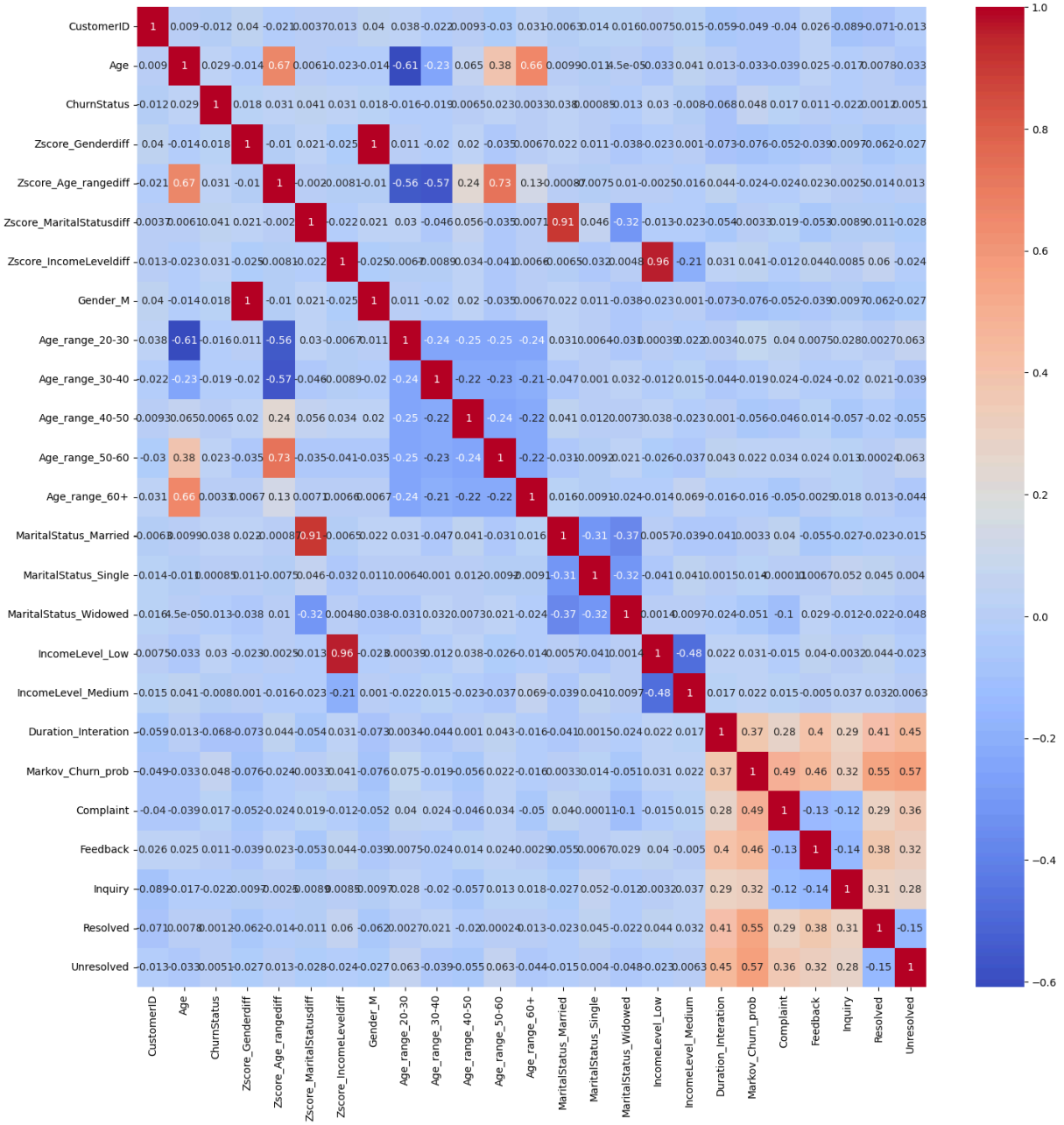
Out[74]:

	CustomerID	Age	ChurnStatus	Zscore_Genderdiff	Zscore_Age_rangediff
0	1	62	0	21.898796	3.071478
1	2	65	1	21.898796	3.071478
2	3	18	0	21.898796	2.238200
3	4	21	0	21.898796	-17.033005
4	5	21	0	21.898796	-17.033005
...	...	...	...	...	...
995	996	54	0	-23.096539	20.951841
996	997	19	0	21.898796	2.238200
997	998	47	0	21.898796	6.257337
998	999	23	0	21.898796	-17.033005
999	1000	34	0	21.898796	-19.062154

1000 rows × 25 columns

In [76]:
```
corr_df = enhanced_Interaction_Customer.corr()

plt.figure(figsize=(17, 17))
sns.heatmap(corr_df, annot=True, cmap='coolwarm')
```

Out[76]: <Axes: >



Transaction Sheet

In [77]:
```
Transaction_History = data["Transaction_History"]
Transaction_History = Transaction_History.merge(data['Churn_Status'], how =
Transaction_History
```

Out[77]:

	CustomerID	TransactionID	TransactionDate	AmountSpent	ProductCat
0	1	7194	2022-03-27	416.50	Elect
1	2	7250	2022-08-08	54.96	Cl
2	2	9660	2022-07-25	197.50	Elect
3	2	2998	2022-01-25	101.31	Fur
4	2	1228	2022-07-24	397.37	Cl
...	...	...	...	...	
5049	1000	2724	2022-09-08	232.06	Gro
5050	1000	2917	2022-12-13	324.98	
5051	1000	2979	2022-06-15	375.34	Gro
5052	1000	8594	2022-04-08	166.73	
5053	1000	5529	2022-11-23	93.73	Fur

5054 rows × 6 columns

In [78]:

```
df_diff = Transaction_History.groupby(["CustomerID","ProductCategory"])["Amo
    First_Transaction="min",
    Last_Transaction="max",
    total_Transaction = "sum",
    mean_Transaction = "mean",
    count_Transaction = "count"
)

df_diff["Days_Difference"] = (df_diff["Last_Transaction"] - df_diff["First_T
df_diff.reset_index().head(50)
```

Out[78]:

	CustomerID	ProductCategory	First_Transaction	Last_Transaction	total_T
0	1	Electronics	416.50	416.50	
1	2	Clothing	54.96	397.37	
2	2	Electronics	197.50	311.34	
3	2	Furniture	101.31	101.31	
4	2	Groceries	199.73	199.73	
5	3	Books	241.06	241.06	
6	3	Clothing	51.07	51.07	
7	3	Furniture	403.32	419.95	
8	3	Groceries	287.72	299.86	
9	4	Clothing	44.22	44.22	
10	4	Electronics	125.64	241.45	
11	4	Furniture	382.39	382.39	
12	4	Groceries	123.59	123.59	
13	5	Electronics	155.76	308.80	
14	5	Furniture	178.91	219.98	
15	5	Groceries	69.86	475.69	
16	6	Books	365.57	365.57	
17	6	Clothing	448.86	448.86	
18	6	Electronics	204.16	204.16	
19	6	Furniture	41.70	104.00	
20	7	Books	86.73	86.73	
21	8	Books	240.92	263.24	
22	8	Clothing	346.24	483.91	
23	8	Electronics	161.56	161.56	
24	8	Furniture	253.68	297.33	
25	9	Clothing	325.96	472.21	
26	9	Electronics	96.67	96.67	
27	9	Groceries	44.26	378.14	
28	10	Clothing	35.29	35.29	
29	10	Electronics	178.78	178.78	
30	10	Furniture	5.86	471.18	
31	10	Groceries	179.00	303.40	
32	11	Books	163.78	426.06	

	CustomerID	ProductCategory	First_Transaction	Last_Transaction	total_T
33	11	Groceries	347.37	462.00	
34	12	Electronics	87.10	87.10	
35	13	Books	164.72	164.72	
36	13	Clothing	54.17	54.17	
37	13	Electronics	173.13	173.13	
38	14	Books	150.22	150.22	
39	14	Clothing	335.44	335.44	
40	14	Furniture	412.20	468.35	
41	14	Groceries	156.21	478.59	
42	15	Books	160.79	283.39	
43	15	Clothing	185.60	185.60	
44	15	Electronics	383.90	383.90	
45	15	Furniture	151.88	273.54	
46	15	Groceries	68.63	406.29	
47	16	Books	52.67	352.99	
48	16	Clothing	470.53	470.53	
49	16	Electronics	97.06	97.06	

In [79]:

```
Transaction_History["TransactionDate"] = pd.to_datetime(Transaction_History[
Transaction_History = Transaction_History.sort_values(by=["CustomerID", "Tra


daily_spending = Transaction_History.groupby(["CustomerID", "TransactionDate
    Transaction_Count=("TransactionID", "count"),
    Total_Spent=("AmountSpent", "sum")
).reset_index()


daily_spending["Rolling_Transaction_Count"] = daily_spending.groupby("Custom
daily_spending["Rolling_Total_Spent"] = daily_spending.groupby("CustomerID")


max_transaction_7d = daily_spending.loc[daily_spending.groupby("CustomerID")


max_spent_7d = daily_spending.loc[daily_spending.groupby("CustomerID")["Roll

print("How many times spent in 7 days at most? \n", max_transaction_7d.merge
print("How much spent in 7 days at most? \n", max_spent_7d.merge(data["Churr
```

How many times spent in 7 days at most?

	CustomerID	TransactionDate	Transaction_Count	Total_Spent	\
0	1	2022-03-27	1	416.50	
1	2	2022-01-25	1	101.31	
2	3	2022-03-02	1	299.86	
3	4	2022-07-31	1	125.64	
4	5	2022-05-25	1	69.86	
5	6	2022-08-09	1	41.70	
6	7	2022-10-24	1	86.73	
7	8	2022-06-10	1	346.24	
8	9	2022-05-26	1	325.96	
9	10	2022-06-05	1	179.00	
10	11	2022-06-06	1	334.62	
11	12	2022-11-02	1	87.10	
12	13	2022-08-10	1	164.72	
13	14	2022-02-21	1	213.22	
14	15	2022-01-26	1	406.29	
15	16	2022-08-07	1	97.06	
16	17	2022-06-27	1	124.02	
17	18	2022-02-08	1	154.09	
18	19	2022-07-18	1	19.80	
19	20	2022-04-04	1	462.10	

	Rolling_Transaction_Count	Rolling_Total_Spent	ChurnStatus
0	1.0	416.50	0
1	2.0	386.52	1
2	2.0	703.18	0
3	2.0	508.03	0
4	2.0	545.55	0
5	2.0	145.70	0
6	1.0	86.73	0
7	2.0	643.57	1
8	2.0	370.22	0
9	2.0	650.18	1
10	2.0	796.62	0
11	1.0	87.10	0
12	2.0	218.89	0
13	2.0	681.57	0
14	2.0	679.83	0
15	2.0	450.05	1
16	2.0	191.22	0
17	2.0	408.31	1
18	2.0	497.63	0
19	2.0	787.09	0

How much spent in 7 days at most?

	CustomerID	TransactionDate	Transaction_Count	Total_Spent	\
0	1	2022-03-27	1	416.50	
1	2	2022-07-25	1	197.50	
2	3	2022-03-02	1	299.86	
3	4	2022-07-31	1	125.64	
4	5	2022-12-21	1	218.98	
5	6	2022-10-29	1	448.86	
6	7	2022-10-24	1	86.73	
7	8	2022-11-09	1	483.91	
8	9	2022-10-19	1	378.14	
9	10	2022-06-05	1	179.00	

```
10          11      2022-06-06                  1       334.62
11          12      2022-11-02                  1        87.10
12          13      2022-09-27                  1       173.13
13          14      2022-06-17                  1       442.99
14          15      2022-03-28                  1       283.39
15          16      2022-09-01                  1       147.82
16          17      2022-07-19                  1       344.04
17          18      2022-05-27                  1       164.11
18          19      2022-07-18                  1        19.80
19          20      2022-09-14                  1       485.84
```

	Rolling_Transaction_Count	Rolling_Total_Spent	ChurnStatus
0	1.0	416.50	0
1	2.0	594.87	1
2	2.0	703.18	0
3	2.0	508.03	0
4	2.0	592.49	0
5	2.0	653.02	0
6	1.0	86.73	0
7	2.0	747.15	1
8	2.0	850.35	0
9	2.0	650.18	1
10	2.0	796.62	0
11	1.0	87.10	0
12	2.0	337.85	0
13	2.0	921.58	0
14	2.0	689.68	0
15	2.0	618.35	1
16	2.0	788.10	0
17	2.0	445.84	1
18	2.0	497.63	0
19	2.0	831.59	0

In [80]:
```
Transaction_History["TransactionDate"] = pd.to_datetime(Transaction_History[


Transaction_History = Transaction_History.sort_values(by=["CustomerID", "Prc

daily_spending = Transaction_History.groupby(["CustomerID", "ProductCategory
    Transaction_Count=("TransactionID", "count"),
    Total_Spent=("AmountSpent", "sum")
).reset_index()


daily_spending["Rolling_Transaction_Count"] = daily_spending.groupby(["Custc
    .rolling(window=7, min_periods=1).sum().reset_index(level=[0,1], drop=Tr

daily_spending["Rolling_Total_Spent"] = daily_spending.groupby(["CustomerID"
    .rolling(window=7, min_periods=1).sum().reset_index(level=[0,1], drop=Tr


daily_spending = daily_spending.dropna(subset=["Rolling_Transaction_Count",


max_transaction_7d = daily_spending.loc[daily_spending.groupby(["CustomerID"
```

```
max_spent_7d = daily_spending.loc[daily_spending.groupby(["CustomerID", "Pro


churn_data = data["Churn_Status"][["CustomerID", "ChurnStatus"]]
max_transaction_7d = max_transaction_7d.merge(churn_data, how="left", on="Cu
max_spent_7d = max_spent_7d.merge(churn_data, how="left", on="CustomerID")
```

In [81]:
```
Transaction_History["TransactionDate"] = pd.to_datetime(Transaction_History[


Transaction_History = Transaction_History.sort_values(by=["CustomerID", "Pro


daily_spending = Transaction_History.groupby(["CustomerID", "ProductCategory
    Transaction_Count=("TransactionID", "count"),
    Total_Spent=("AmountSpent", "sum")
).reset_index()
```

In [82]:
```
test = daily_spending.groupby("CustomerID")['Total_Spent'].rolling(window=7,
```

In [83]:
```
test_1 = test.merge(data["Churn_Status"], how="left", on="CustomerID")[test.
```

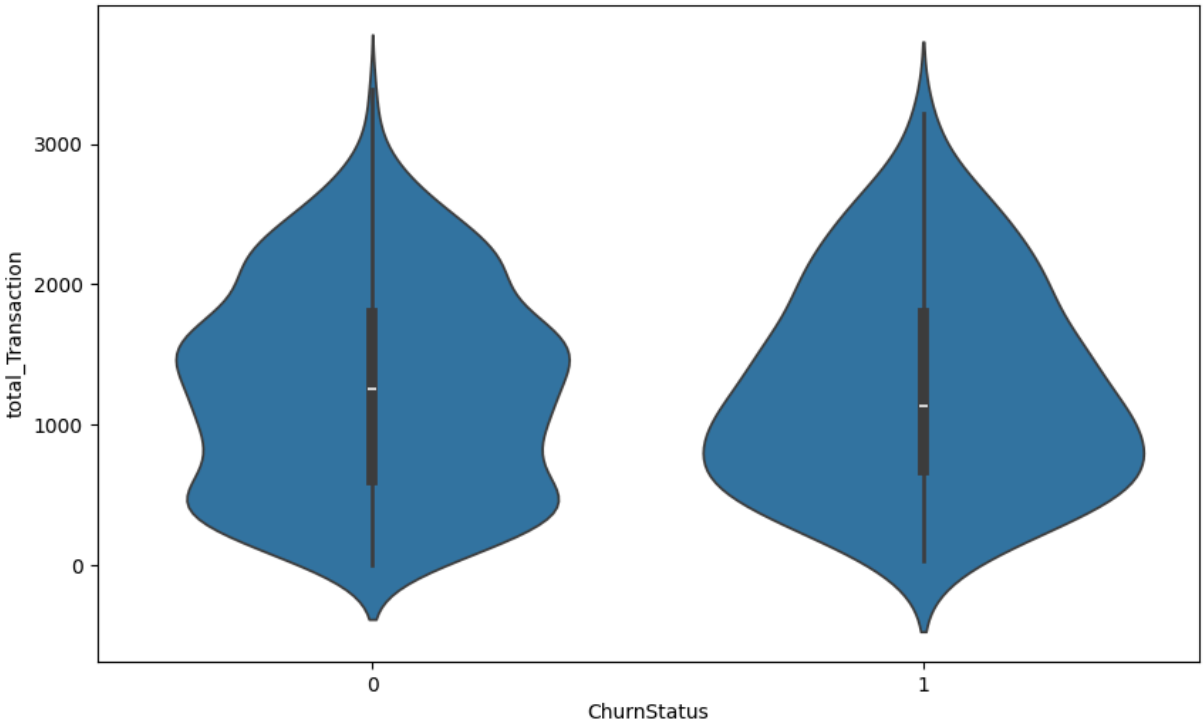
In [84]:
```
test_0 = test.merge(data["Churn_Status"], how="left", on="CustomerID")[test.
```

In [85]:
```
df_diff = df_diff.reset_index().merge(data["Churn_Status"], how="left", on="
```

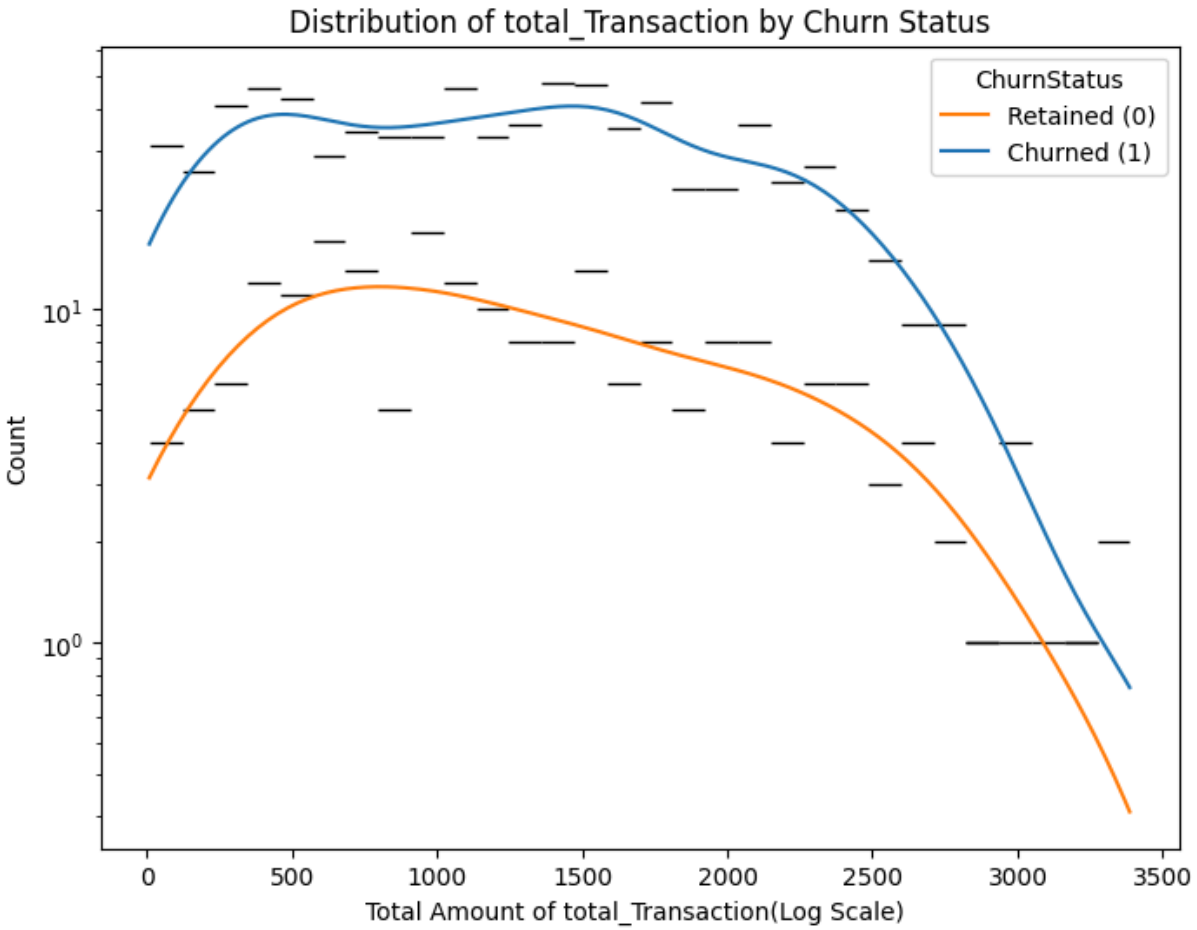
In [86]:
```
all_total_spent = df_diff.groupby("CustomerID")["total_Transaction"].sum()
all_mean_spent = df_diff.groupby("CustomerID")["total_Transaction"].mean()
all_total_spent = all_total_spent.reset_index().merge(data["Churn_Status"],
all_mean_spent = all_mean_spent.reset_index().merge(data["Churn_Status"], ho
```

In [87]:
```
import seaborn as sns
import matplotlib.pyplot as plt


plt.figure(figsize=(10, 6))
sns.violinplot(x="ChurnStatus" , y= "total_Transaction"
               , data=all_total_spent)
```

Out[87]:
```
<Axes: xlabel='ChurnStatus', ylabel='total_Transaction'>
```

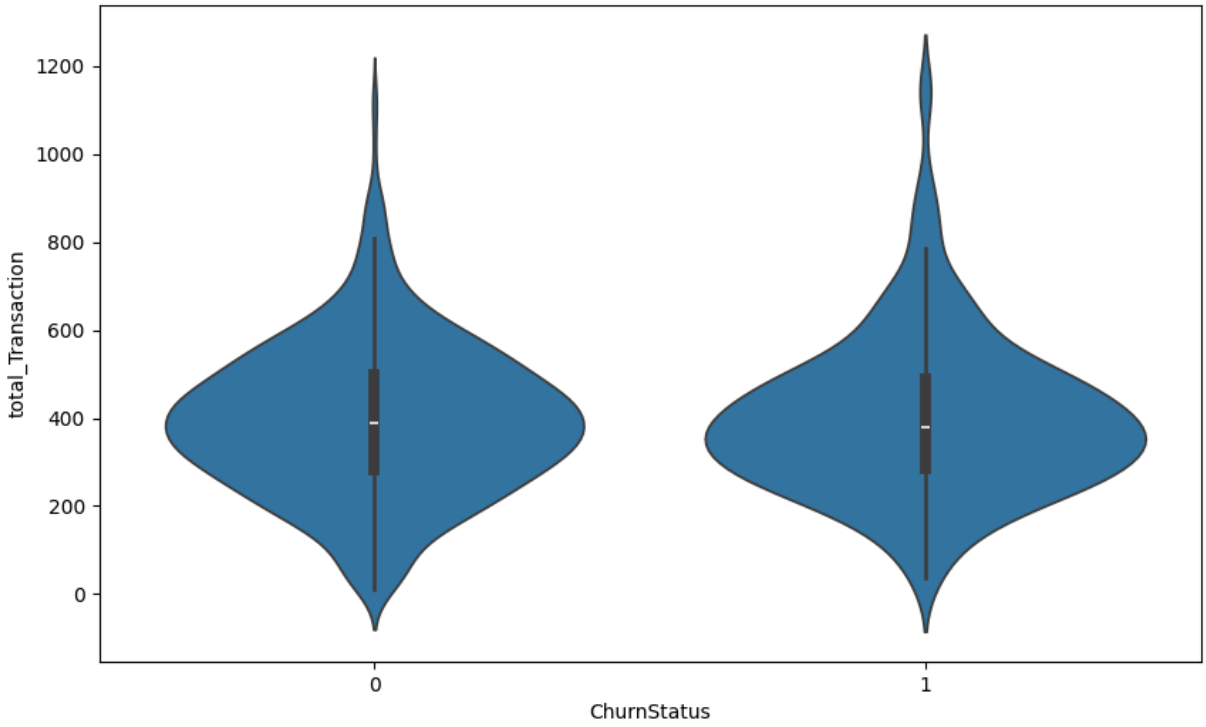


In [88]:
```
plt.figure(figsize=(8,6))

sns.histplot(data=all_total_spent, x="total_Transaction", hue="ChurnStatus",
             bins=30, kde=True, log_scale=(False, True), alpha=0.5)

plt.title("Distribution of total_Transaction by Churn Status")
plt.xlabel("Total Amount of total_Transaction(Log Scale) ")
plt.ylabel("Count")
plt.legend(title="ChurnStatus", labels=["Retained (0)", "Churned (1)"])  # 
plt.show()
```

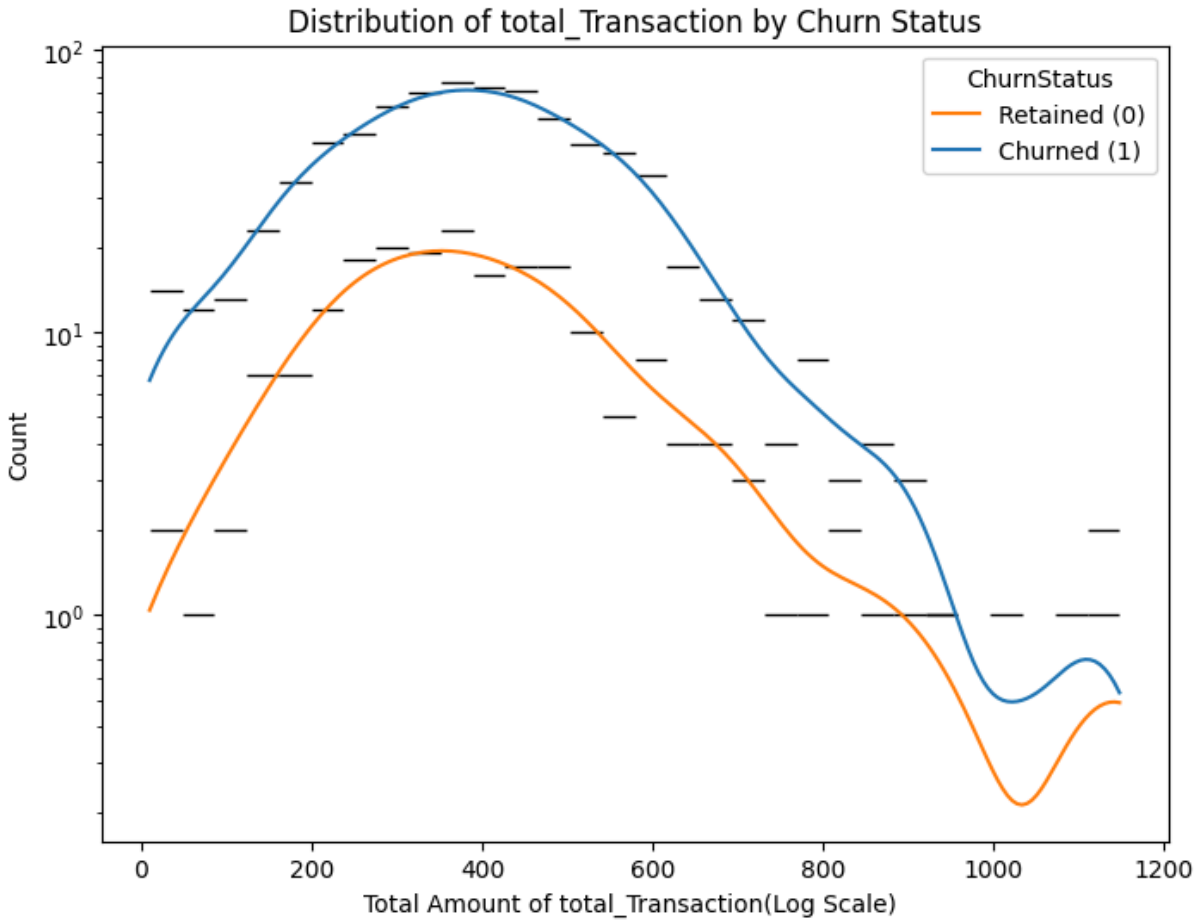


In [89]:
```
import seaborn as sns
import matplotlib.pyplot as plt


plt.figure(figsize=(10, 6))
sns.violinplot(x="ChurnStatus" , y= "total_Transaction"
               , data=all_mean_spent)
```

Out[89]: <Axes: xlabel='ChurnStatus', ylabel='total_Transaction'>

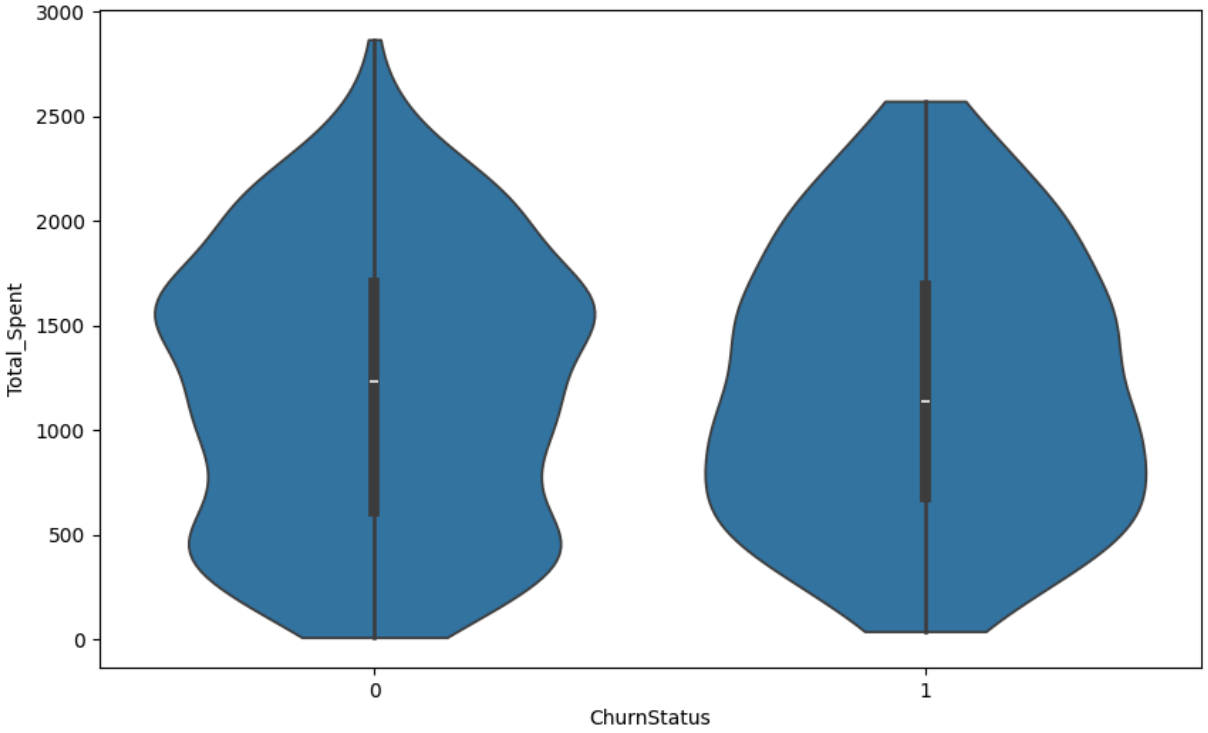


In [90]:
```
plt.figure(figsize=(8,6))

sns.histplot(data=all_mean_spent, x="total_Transaction", hue="ChurnStatus",
             bins=30, kde=True, log_scale=(False, True), alpha=0.5)

plt.title("Distribution of total_Transaction by Churn Status")
plt.xlabel("Total Amount of total_Transaction(Log Scale) ")
plt.ylabel("Count")
plt.legend(title="ChurnStatus", labels=["Retained (0)", "Churned (1)"])  # 
plt.show()
```

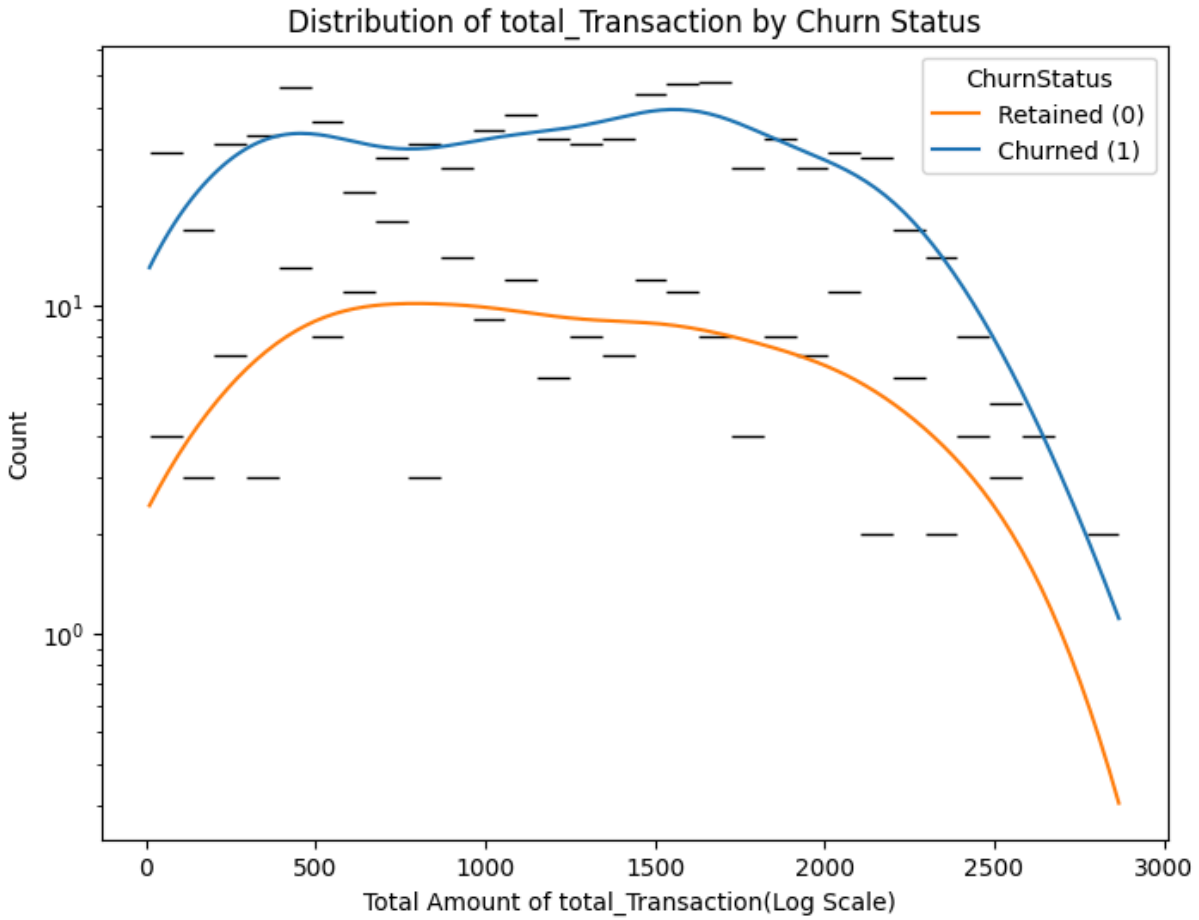


In [101…

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.violinplot(x="ChurnStatus" , y= "Total_Spent"
               , data=test.merge(data["Churn_Status"], how="left", on="Custo
plt.savefig("Plots/violin_total_spent.png", dpi=300, bbox_inches='tight')
```



In [92]:
```
plt.figure(figsize=(8,6))

sns.histplot(data=test.merge(data["Churn_Status"], how="left", on="CustomerI
             bins=30, kde=True, log_scale=(False, True), alpha=0.5)

plt.title("Distribution of total_Transaction by Churn Status")
plt.xlabel("Total Amount of total_Transaction(Log Scale) ")
plt.ylabel("Count")
plt.legend(title="ChurnStatus", labels=["Retained (0)", "Churned (1)"])  # 
plt.show()
```



In [93]:
```
test.merge(data["Churn_Status"], how="left", on="CustomerID")
```

Out[93]:

	CustomerID	Total_Spent	ChurnStatus
0	1	416.50	0
1	2	1547.42	1
2	3	1702.98	0
3	4	917.29	0
4	5	1692.69	0
...	...	...	...
995	996	227.25	0
996	997	419.82	0
997	998	252.15	0
998	999	2233.24	0
999	1000	1670.79	0

1000 rows × 3 columns

In [94]:
```
mean_churned = all_total_spent[all_total_spent['ChurnStatus'] == 1]['total_T
ined = all_total_spent[all_total_spent['ChurnStatus'] == 0]['total_
```

```
print(f"Churned means: {mean_churned:.2f}")
print(f"Retained means: {mean_retained:.2f}")

from scipy.stats import ttest_ind


churned_transactions = all_total_spent[all_total_spent['ChurnStatus'] == 1][
retained_transactions = all_total_spent[all_total_spent['ChurnStatus'] == 0]


t_stat, p_value = ttest_ind(churned_transactions, retained_transactions, equ

print(f"T Statistics: {t_stat:.4f}, P Value: {p_value:.4f}")

if p_value < 0.05:
    print("Refuse H0")
else:
    print("Accept H0")
```

```
Churned means: 1269.01
Retained means: 1266.58
T Statistics: 0.0421, P Value: 0.9664
Accept H0
```

In [95]:
```
mean_churned = test.merge(data["Churn_Status"], how="left", on="CustomerID")
mean_retained = test.merge(data["Churn_Status"], how="left", on="CustomerID"

print(f"Churned means {mean_churned:.2f}")
print(f"Retained means: {mean_retained:.2f}")

from scipy.stats import ttest_ind


churned_transactions = test.merge(data["Churn_Status"], how="left", on="Cust
retained_transactions =test.merge(data["Churn_Status"], how="left", on="Cust

t_stat, p_value = ttest_ind(churned_transactions, retained_transactions, equ

print(f"T Statistics: {t_stat:.4f}, P Value: {p_value:.4f}")

if p_value < 0.05:
    print("Refuse H0")
else:
    print("Accept H0")
```

```
Churned means 1202.08
Retained means: 1206.55
T Statistics: -0.0885, P Value: 0.9295
Accept H0
```

In [96]:
```
daily_spending.sort_values('TransactionDate').groupby('CustomerID').apply(la
```

```
/var/folders/pt/0nf2m3pj1f3b373wsyfc6glh0000gn/T/ipykernel_52421/4214387758.
py:1: DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior is de
precated, and in a future version of pandas the grouping columns will be exc
luded from the operation. Either pass `include_groups=False` to exclude the
groupings or explicitly select the grouping columns after groupby to silence
this warning.
```

Out[96]:
```
CustomerID
1       1.0
2       2.0
3       1.0
4       2.0
5       1.0
       ...
996     1.0
997     1.0
998     1.0
999     2.0
1000    1.0
Name: Transaction_Count, Length: 1000, dtype: float64
```

In [97]:
```
test3 = (
    daily_spending
    .assign(TransactionDate=lambda df: pd.to_datetime(df['TransactionDate'])
    .sort_values(['CustomerID', 'TransactionDate'])
    .set_index('TransactionDate')
    .groupby('CustomerID')['Total_Spent']
    .rolling('7D', closed='left')
    .sum()
    .reset_index()
    .rename(columns={'Total_Spent': 'Rolling_7D_Sum'})
)
```

In [98]:
```
test4 = (
    daily_spending
    .assign(TransactionDate=lambda df: pd.to_datetime(df['TransactionDate'])
    .sort_values(['CustomerID', 'TransactionDate'])
    .set_index('TransactionDate')
    .groupby(['CustomerID','ProductCategory'])['Total_Spent']
    .rolling('7D', closed='left')
    .sum()
    .reset_index()
    )
```

In [99]:
```
test4 = test4.fillna(0).groupby(["CustomerID","ProductCategory"])["Total_Spe
test4 = test4.pivot_table(index="CustomerID", columns="ProductCategory", val
```

In [100…
```
test4
```

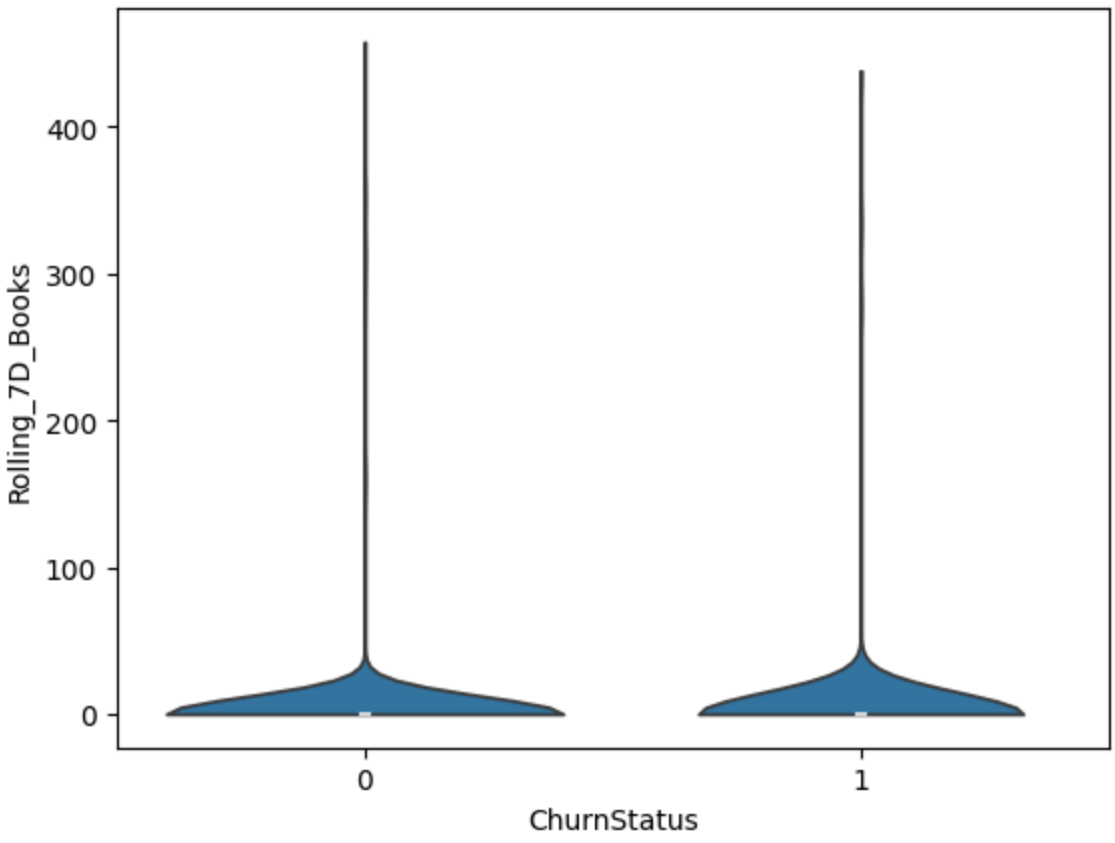
Out[100…

	CustomerID	Rolling_7D_Books	Rolling_7D_Clothing	Rolling_7D_Electronic
0	1	0.0	0.0	0.
1	2	0.0	0.0	0.
2	3	0.0	0.0	0.
3	4	0.0	0.0	0.
4	5	0.0	0.0	0.
...	...	...	...	.
995	996	0.0	0.0	0.
996	997	0.0	0.0	0.
997	998	0.0	0.0	0.
998	999	0.0	0.0	0.
999	1000	0.0	0.0	0.

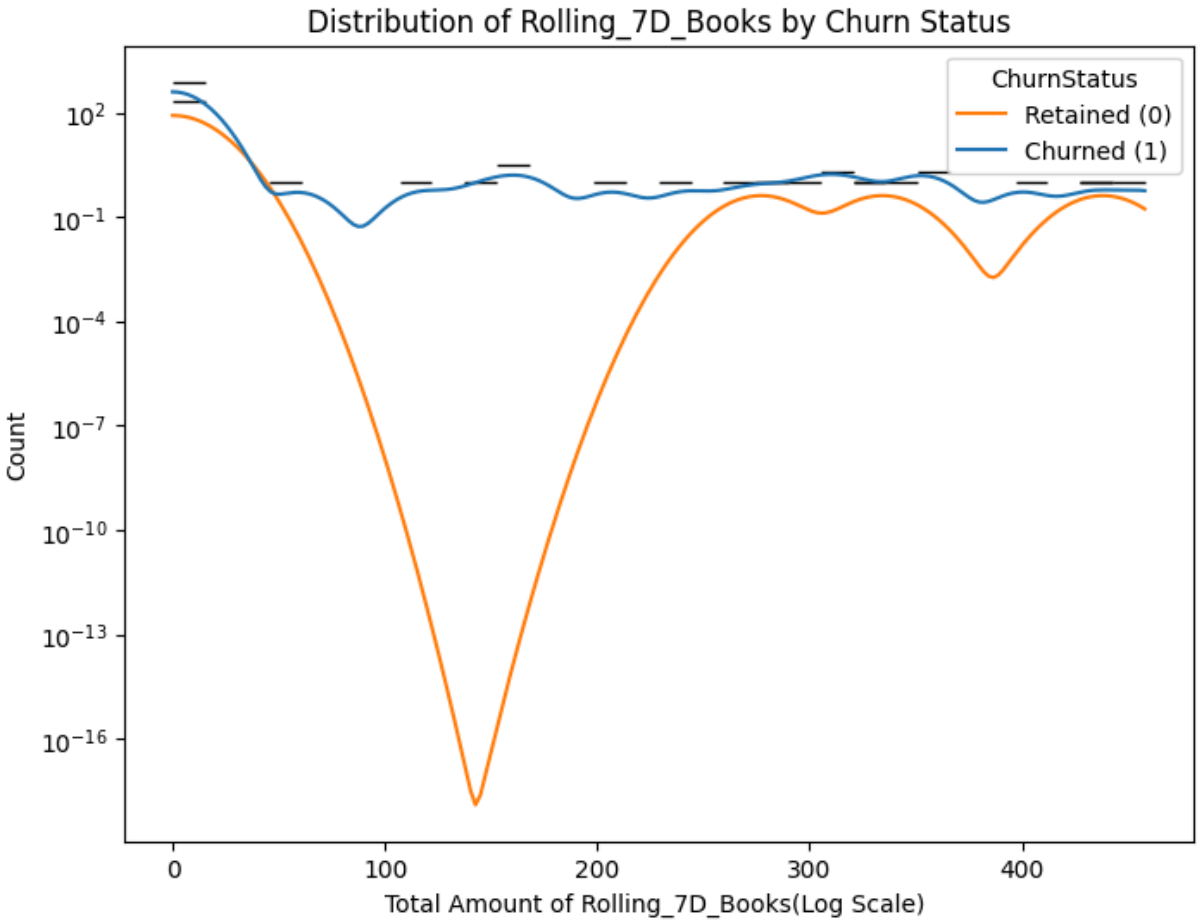
1000 rows × 7 columns

In [62]:
```
sns.violinplot(x="ChurnStatus" , y= "Rolling_7D_Books"
               , data=test4, cut = 0)
```

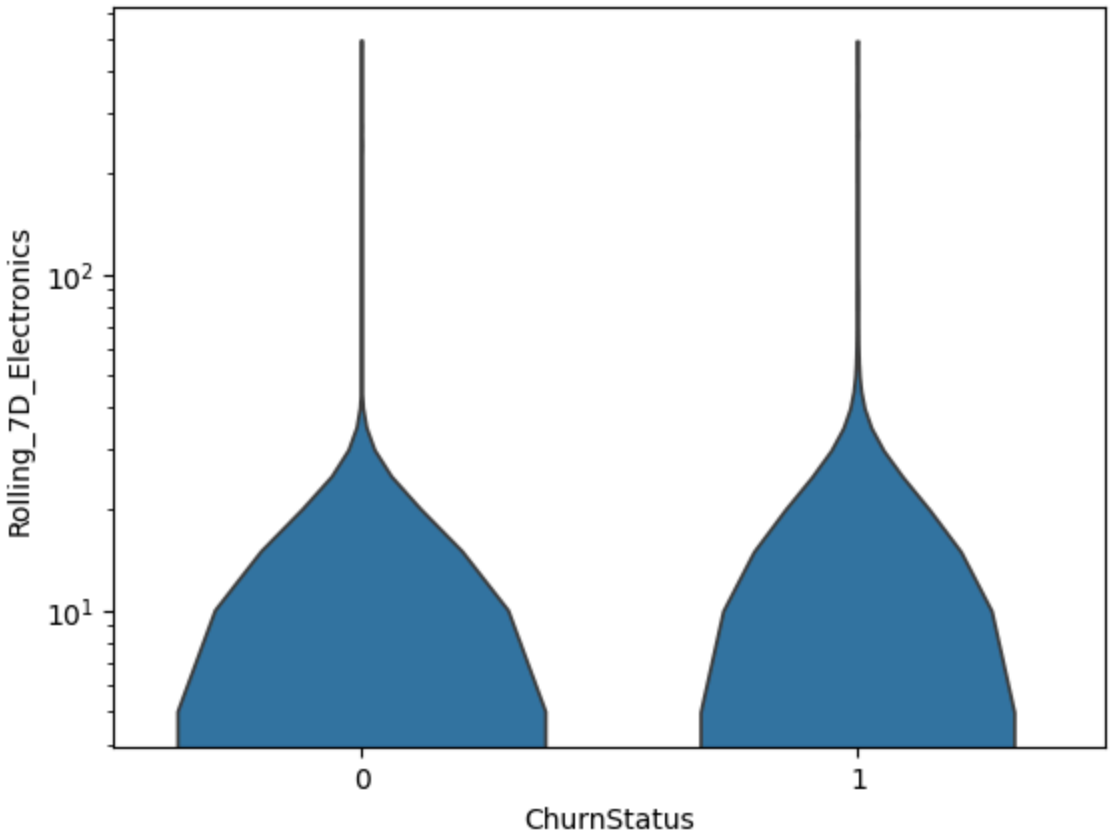
Out[62]: <Axes: xlabel='ChurnStatus', ylabel='Rolling_7D_Books'>



In [63]:
```
plt.figure(figsize=(8,6))

sns.histplot(data=test4, x="Rolling_7D_Books", hue="ChurnStatus",
             bins=30, kde=True, log_scale=(False, True), alpha=0.5)

plt.title("Distribution of Rolling_7D_Books by Churn Status")
plt.xlabel("Total Amount of Rolling_7D_Books(Log Scale) ")
plt.ylabel("Count")
plt.legend(title="ChurnStatus", labels=["Retained (0)", "Churned (1)"])  # 
plt.show()
```

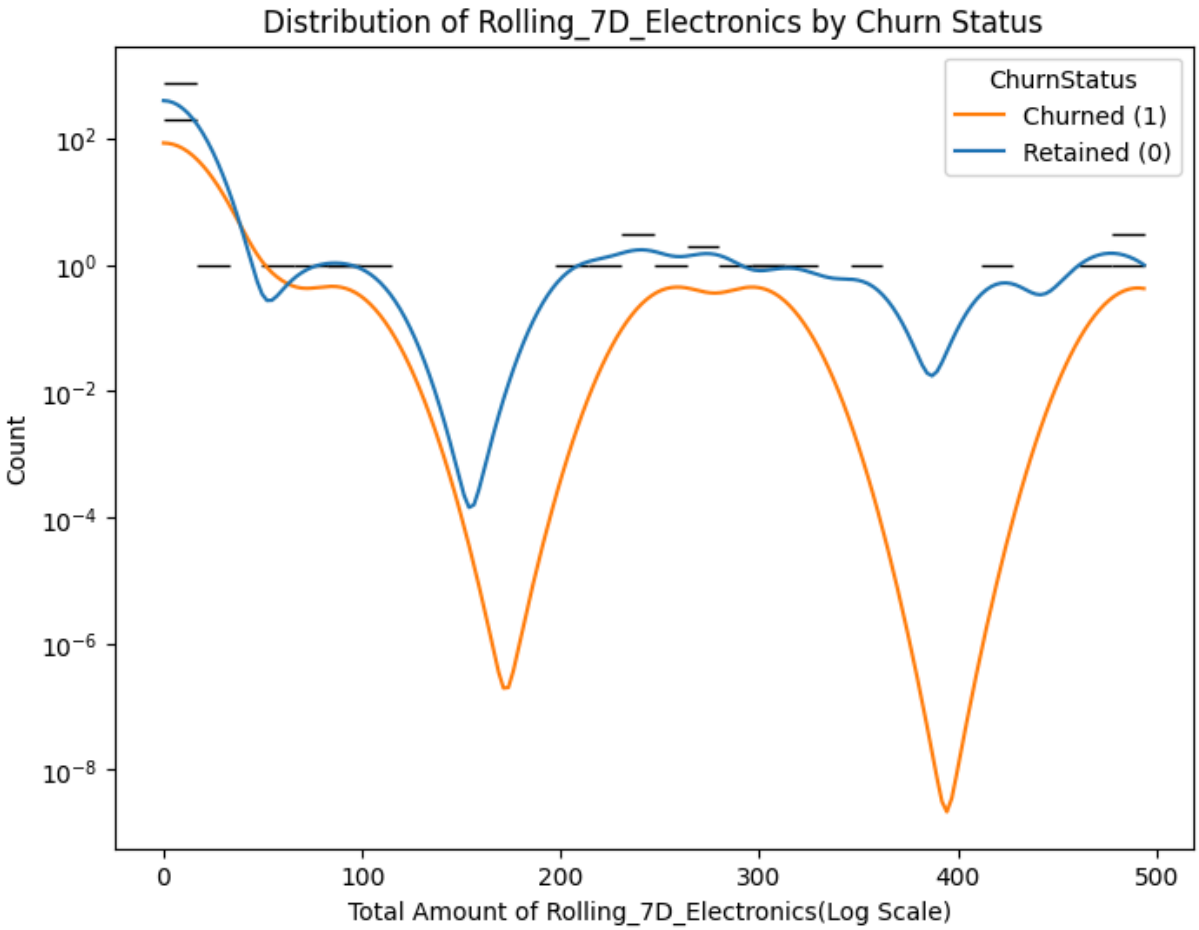


In [64]:
```
sns.violinplot(x="ChurnStatus" , y= "Rolling_7D_Electronics"
               , data=test4, cut = 0)
plt.yscale("log")
```

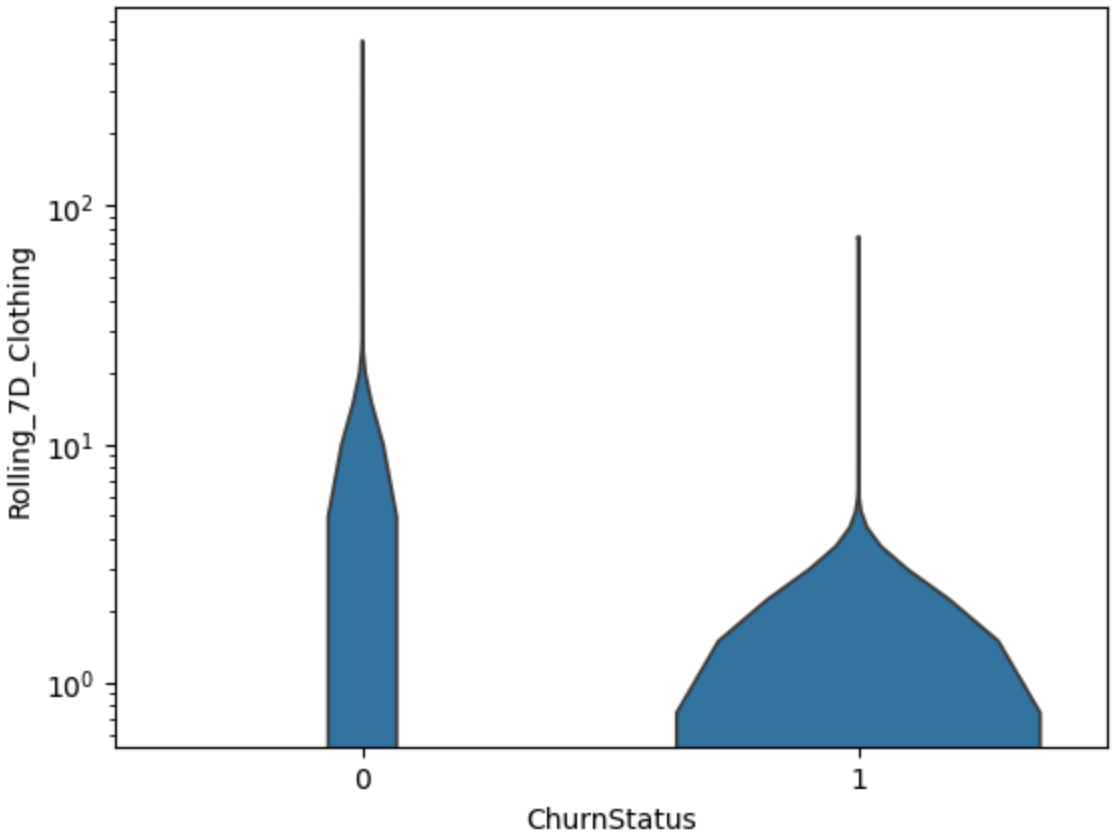


In [107…

```
plt.figure(figsize=(8,6))

sns.histplot(data=test4, x="Rolling_7D_Electronics", hue="ChurnStatus",
             bins=30, kde=True, log_scale=(False, True), alpha=0.5)

plt.title("Distribution of Rolling_7D_Electronics by Churn Status")
plt.xlabel("Total Amount of Rolling_7D_Electronics(Log Scale) ")
plt.ylabel("Count")
plt.legend(title="ChurnStatus", labels=["Churned (1)","Retained (0)"])
plt.savefig("Plots/KDE_RollingElectronics.png", dpi=300, bbox_inches='tight'
plt.show()
```

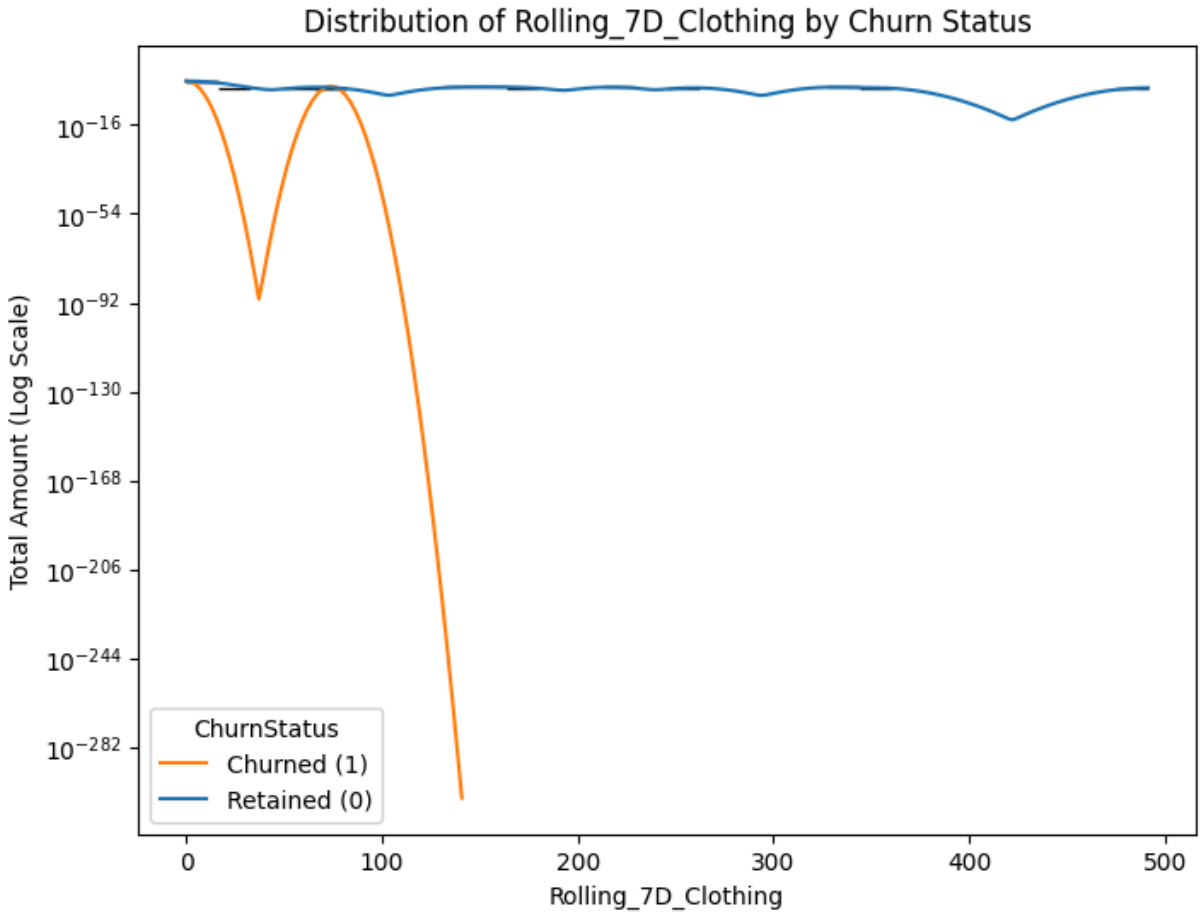


In [66]:
```
sns.violinplot(x="ChurnStatus" , y= "Rolling_7D_Clothing"
               , data=test4, cut = 0)
plt.yscale("log")
```



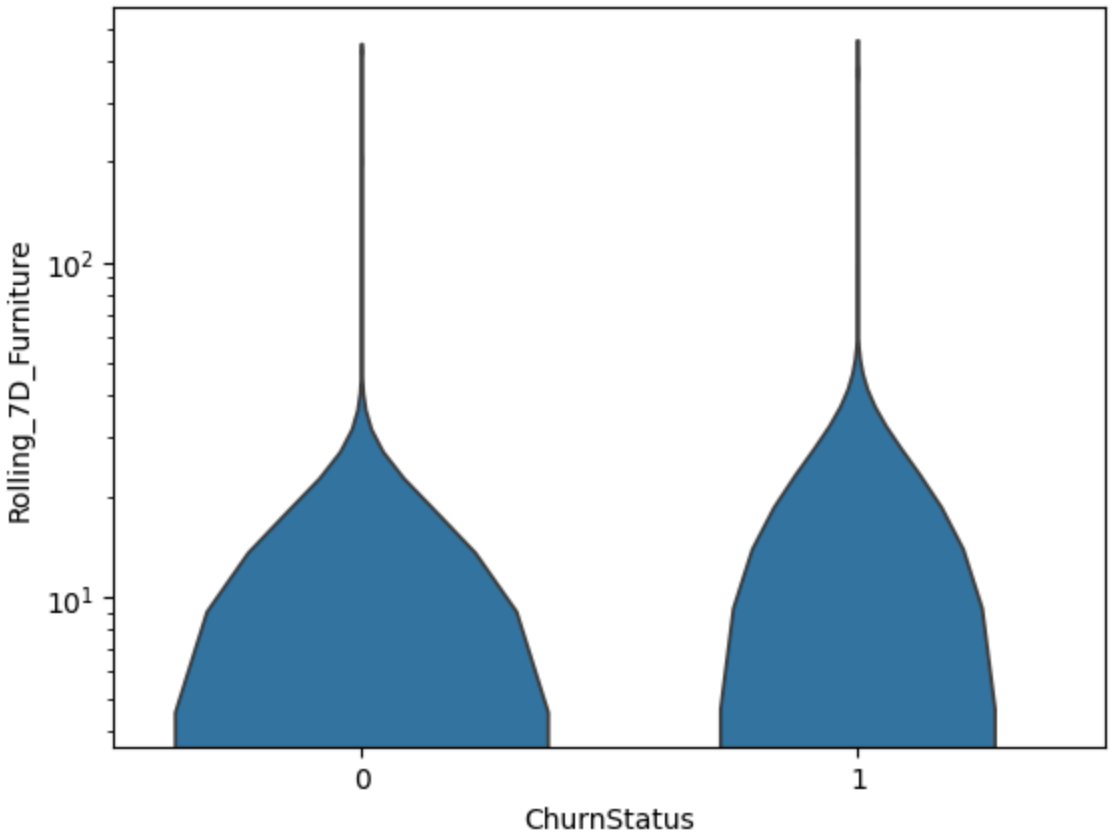
In [108…

```
plt.figure(figsize=(8,6))

sns.histplot(data=test4, x="Rolling_7D_Clothing", hue="ChurnStatus",
             bins=30, kde=True, log_scale=(False, True), alpha=0.5)

plt.title("Distribution of Rolling_7D_Clothing by Churn Status")
plt.xlabel("Rolling_7D_Clothing")
plt.ylabel("Total Amount (Log Scale)")
plt.legend(title="ChurnStatus", labels=["Churned (1)","Retained (0)"])
plt.savefig("Plots/KDE_RollingClothing.png", dpi=300, bbox_inches='tight')
plt.show()
```



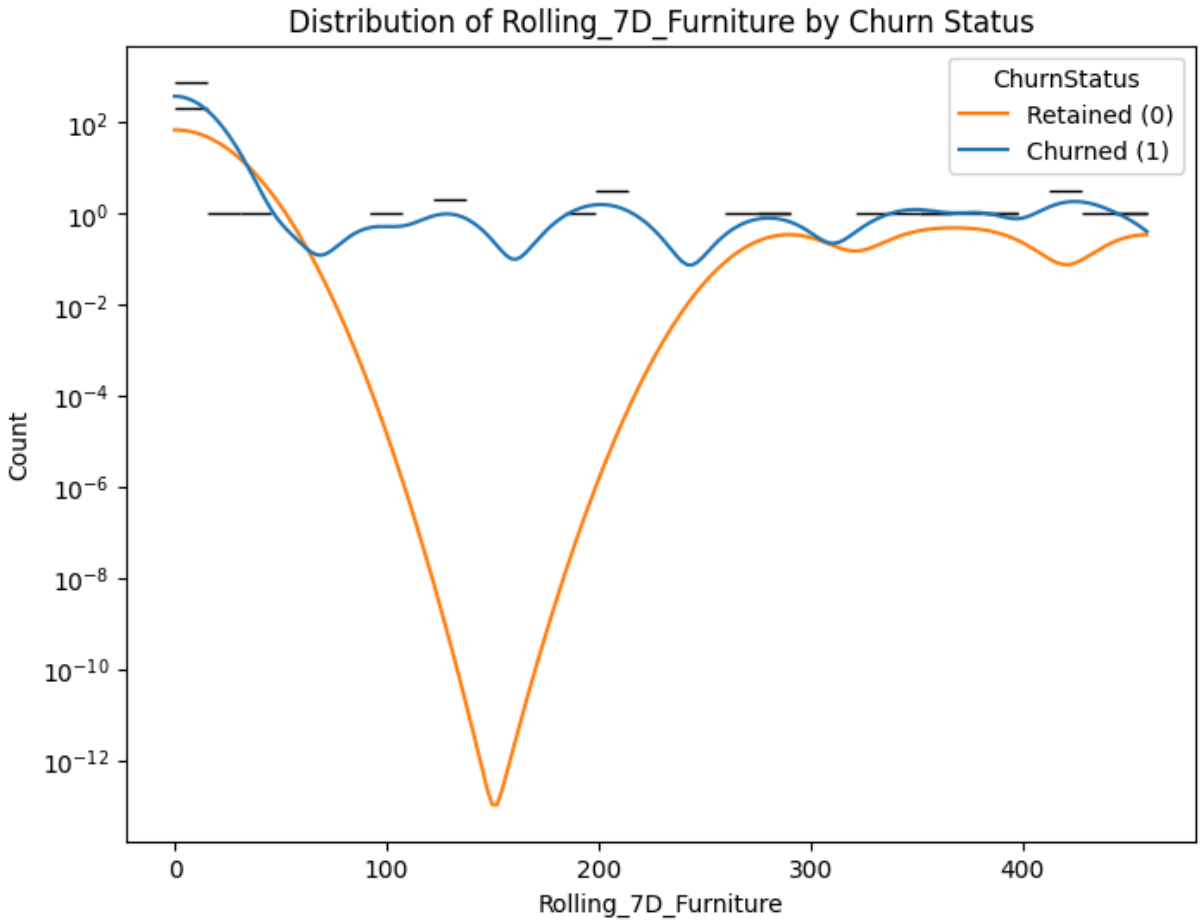
In [109…

```
sns.violinplot(x="ChurnStatus" , y= "Rolling_7D_Furniture"
               , data=test4, cut = 0)
plt.yscale("log")
```



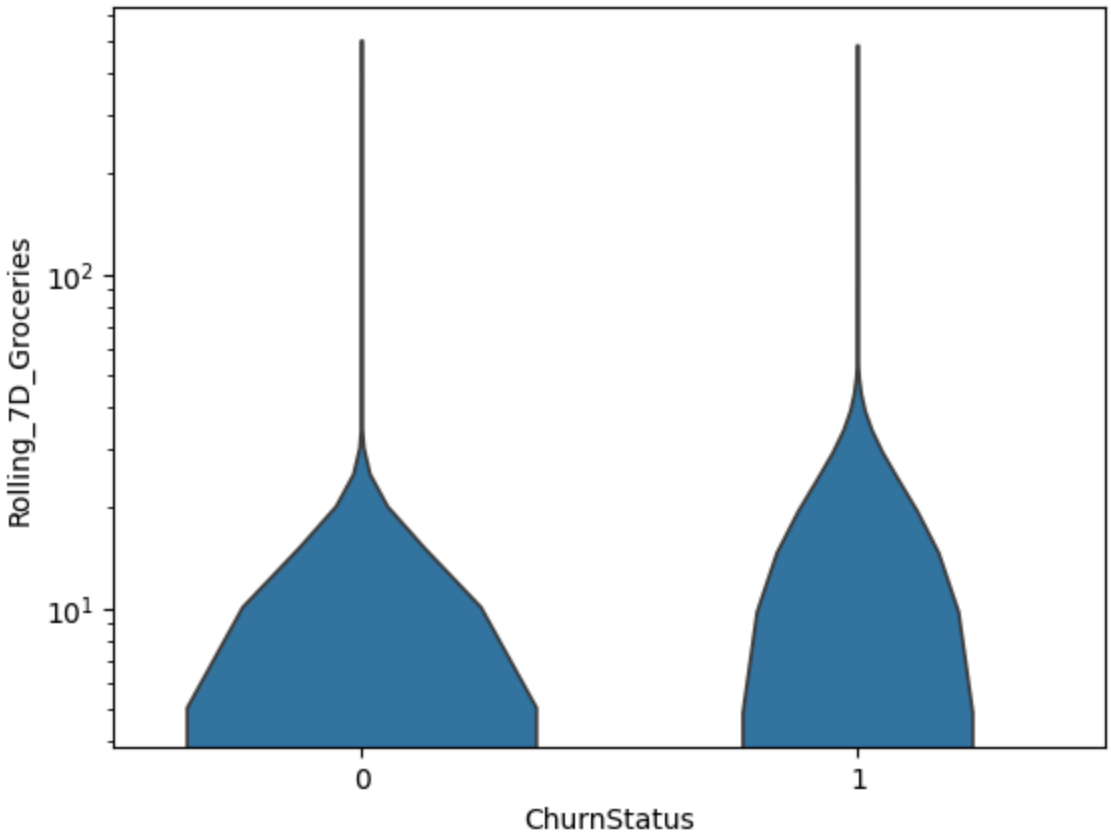
In [180…

```
plt.figure(figsize=(8,6))

sns.histplot(data=test4, x="Rolling_7D_Furniture", hue="ChurnStatus",
             bins=30, kde=True, log_scale=(False, True), alpha=0.5)

plt.title("Distribution of Rolling_7D_Furniture by Churn Status")
plt.xlabel("Rolling_7D_Furniture")
plt.ylabel("Count")
plt.legend(title="ChurnStatus", labels=["Retained (0)", "Churned (1)"])
plt.savefig("Plots/Rolling_7D_Furniture.png", dpi=300, bbox_inches='tight')
plt.show()
```



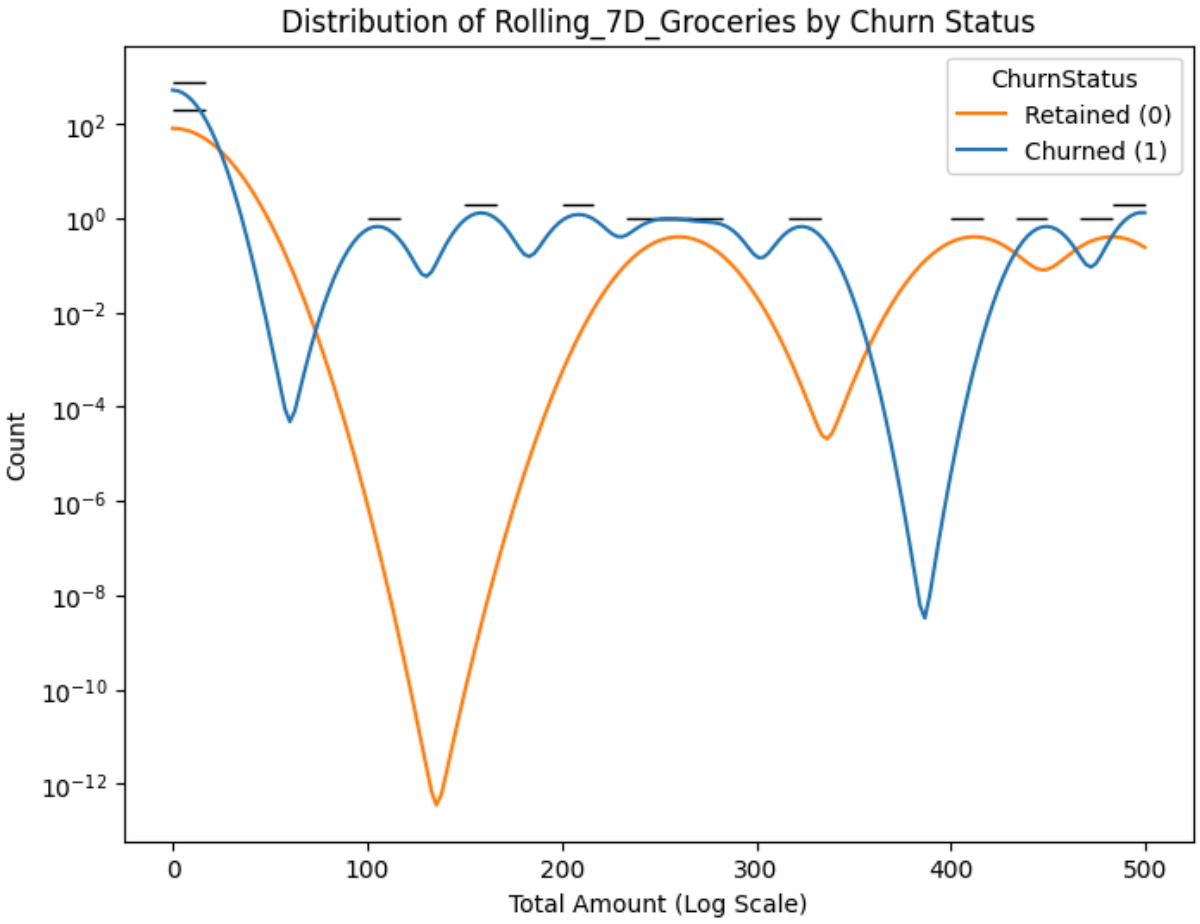
In [111…

```
sns.violinplot(x="ChurnStatus" , y= "Rolling_7D_Groceries"
               , data=test4, cut = 0)
plt.yscale("log")
```



In [179…

```
plt.figure(figsize=(8,6))

sns.histplot(data=test4, x="Rolling_7D_Groceries", hue="ChurnStatus",
             bins=30, kde=True, log_scale=(False, True), alpha=0.5)

plt.title("Distribution of Rolling_7D_Groceries by Churn Status")
plt.xlabel("Total Amount (Log Scale)")
plt.ylabel("Count")
plt.legend(title="ChurnStatus", labels=["Retained (0)", "Churned (1)"])
plt.savefig("Plots/Rolling_7D_Groceries.png", dpi=300, bbox_inches='tight')
plt.show()
```



In [113…
```
test4["Rolling_7D_Furniture_bined"] = pd.cut(test4["Rolling_7D_Furniture"],
```

In [114…
```
test4["Rolling_7D_Clothing_bined"] = pd.cut(test4["Rolling_7D_Clothing"],bir
```

In [115…
```
test4["Rolling_7D_Electronics_bined"] = pd.cut(test4["Rolling_7D_Electronics
```

In [116…
```
test4["Rolling_7D_Books_bined"] = pd.cut(test4["Rolling_7D_Books"], bins = [
```

In [117…
```
test4["Rolling_7D_Groceries_bined"] = pd.cut(test4["Rolling_7D_Groceries"],
```

In [118…
```
test3 = test3.groupby("CustomerID")["Rolling_7D_Sum"].max().reset_index()
test_count = test3.merge(data["Churn_Status"], how = "left", on = "CustomerI
```

In [119…
```
test_count.fillna(0)
```

Out[119…

	CustomerID	Rolling_7D_Sum	ChurnStatus
0	1	0.00	0
1	2	397.37	1
2	3	0.00	0
3	4	241.45	0
4	5	0.00	0
...	...	...	...
995	996	0.00	0
996	997	0.00	0
997	998	0.00	0
998	999	80.28	0
999	1000	0.00	0

1000 rows × 3 columns

In [120…
```
count_label = pd.cut(test_count.fillna(0)['Rolling_7D_Sum'], bins = [-1,200,
```

In [121…
```
rolling_dummies = pd.get_dummies(count_label)
test_count = pd.concat([test_count, rolling_dummies], axis=1).fillna(0)
test_count = test_count.T.drop_duplicates().T
```

In [122…
```
test_count = test_count.merge(test4, how="left", on = ["CustomerID","ChurnSt
```

In [123…
```
test_count
```

Out[123…

	CustomerID	Rolling_7D_Sum	ChurnStatus	Rolling_7D_0-200	Rolling_7D_200-40
0	1	0.0	0	True	Fals
1	2	397.37	1	False	Tru
2	3	0.0	0	True	Fals
3	4	241.45	0	False	Tru
4	5	0.0	0	True	Fals
...	...	...	...	...	.
995	996	0.0	0	True	Fals
996	997	0.0	0	True	Fals
997	998	0.0	0	True	Fals
998	999	80.28	0	True	Fals
999	1000	0.0	0	True	Fals

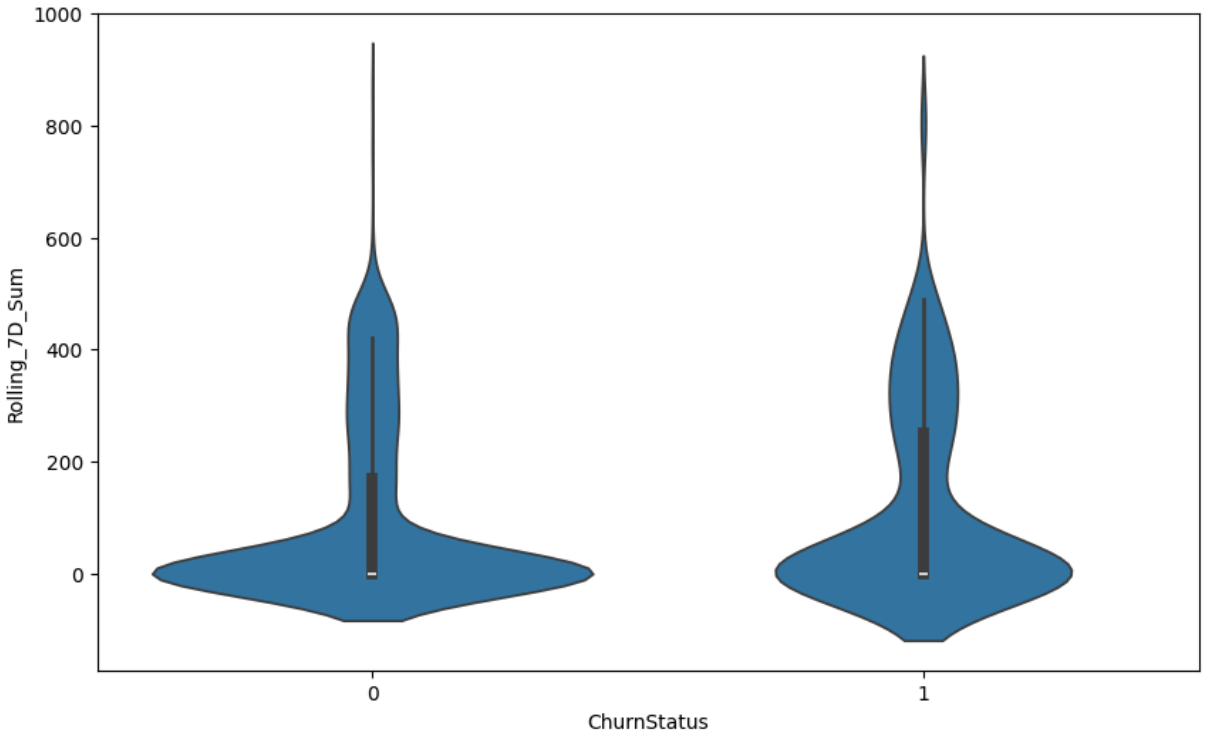
1000 rows × 17 columns

In [124…

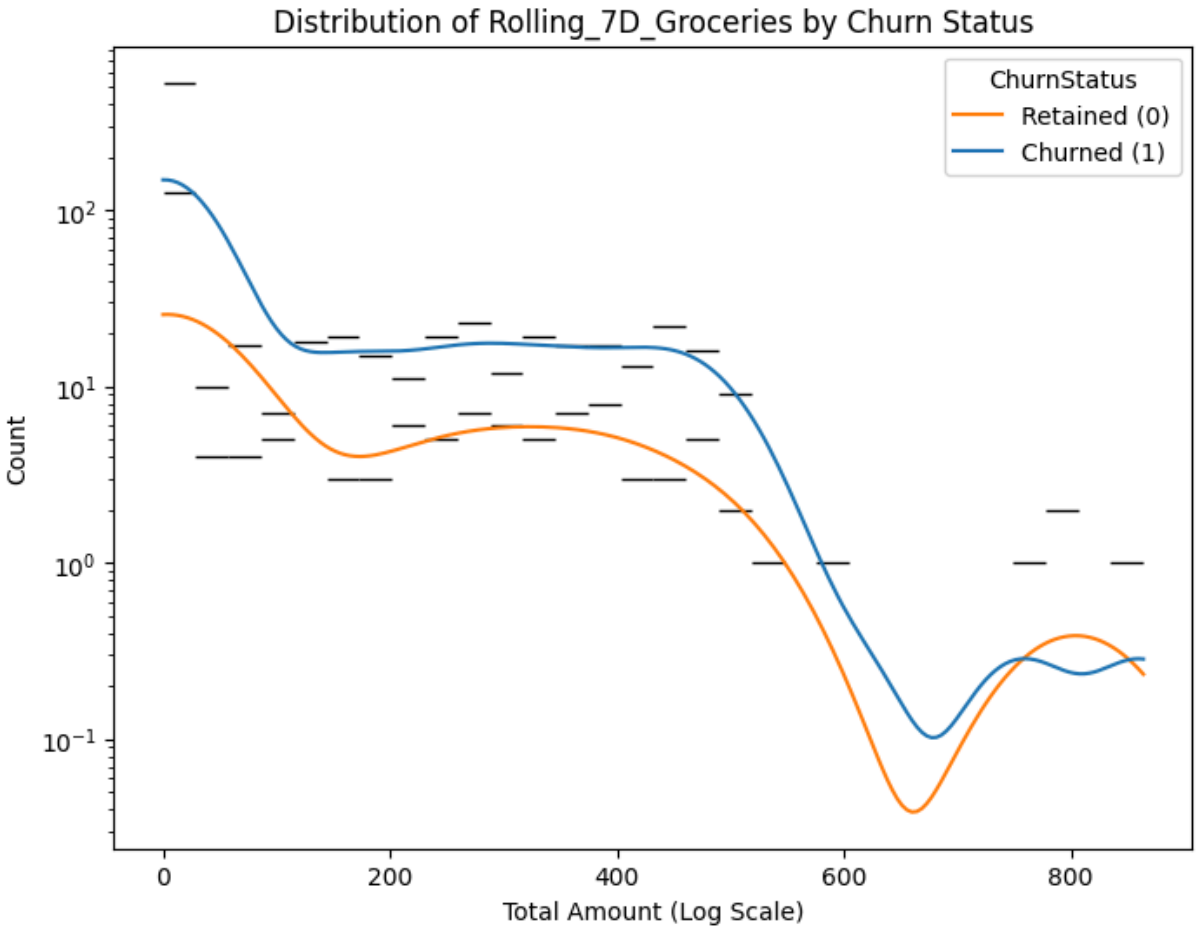
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.violinplot(x="ChurnStatus" , y= "Rolling_7D_Sum"
               , data=test_count)
```

Out[124…

```
<Axes: xlabel='ChurnStatus', ylabel='Rolling_7D_Sum'>
```



In [125…
```
plt.figure(figsize=(8,6))

sns.histplot(data=test_count, x="Rolling_7D_Sum", hue="ChurnStatus",
             bins=30, kde=True, log_scale=(False, True), alpha=0.5)

plt.title("Distribution of Rolling_7D_Groceries by Churn Status")
plt.xlabel("Total Amount (Log Scale)")
plt.ylabel("Count")
plt.legend(title="ChurnStatus", labels=["Retained (0)", "Churned (1)"])  # 
plt.show()
```



In [126…
```
duplicate_ids = data["Churn_Status"]["CustomerID"].duplicated().sum()
print(f"repeted columns: {duplicate_ids}")
```

```
repeted columns: 0
```

In [127…
```
df_grouped = df_diff.groupby(["ProductCategory","CustomerID"])[["First_Trans
                                                "total_Transaction", "mean_Trans
                                                "count_Transaction"]].sum().rese

df_grouped
```

Out[127…

	ProductCategory	CustomerID	First_Transaction	Last_Transaction	total
0	Books	3	241.06	241.06	
1	Books	6	365.57	365.57	
2	Books	7	86.73	86.73	
3	Books	8	240.92	263.24	
4	Books	11	163.78	426.06	
...	...	...	...	...	
3092	Groceries	991	201.31	257.17	
3093	Groceries	994	71.78	336.21	
3094	Groceries	995	494.90	494.90	
3095	Groceries	999	260.06	387.96	
3096	Groceries	1000	232.06	375.34	

3097 rows × 7 columns

In [128…
```
df_pivot = df_diff.pivot_table(index="CustomerID", columns="ProductCategory"
df_pivot.columns = ['_'.join(col).strip() for col in df_pivot.columns.values
df_pivot.reset_index(inplace=True)
```

In [129…
```
df_pivot.columns
```

Out[129…
```
Index(['CustomerID', 'Days_Difference_Books', 'Days_Difference_Clothing',
       'Days_Difference_Electronics', 'Days_Difference_Furniture',
       'Days_Difference_Groceries', 'First_Transaction_Books',
       'First_Transaction_Clothing', 'First_Transaction_Electronics',
       'First_Transaction_Furniture', 'First_Transaction_Groceries',
       'Last_Transaction_Books', 'Last_Transaction_Clothing',
       'Last_Transaction_Electronics', 'Last_Transaction_Furniture',
       'Last_Transaction_Groceries', 'count_Transaction_Books',
       'count_Transaction_Clothing', 'count_Transaction_Electronics',
       'count_Transaction_Furniture', 'count_Transaction_Groceries',
       'mean_Transaction_Books', 'mean_Transaction_Clothing',
       'mean_Transaction_Electronics', 'mean_Transaction_Furniture',
       'mean_Transaction_Groceries', 'total_Transaction_Books',
       'total_Transaction_Clothing', 'total_Transaction_Electronics',
       'total_Transaction_Furniture', 'total_Transaction_Groceries'],
      dtype='object')
```

In [130…
```
category2 = ["Rolling_7D_Furniture_bined", "Rolling_7D_Electronics_bined", "
test_count_endcoded = pd.get_dummies(test_count, columns=category2, drop_fir
test_count_endcoded.head()
```

Out[130…

	CustomerID	Rolling_7D_Sum	ChurnStatus	Rolling_7D_0-200	Rolling_7D_200-400
0	1	0.0	0	True	False
1	2	397.37	1	False	True
2	3	0.0	0	True	False
3	4	241.45	0	False	True
4	5	0.0	0	True	False

5 rows × 27 columns

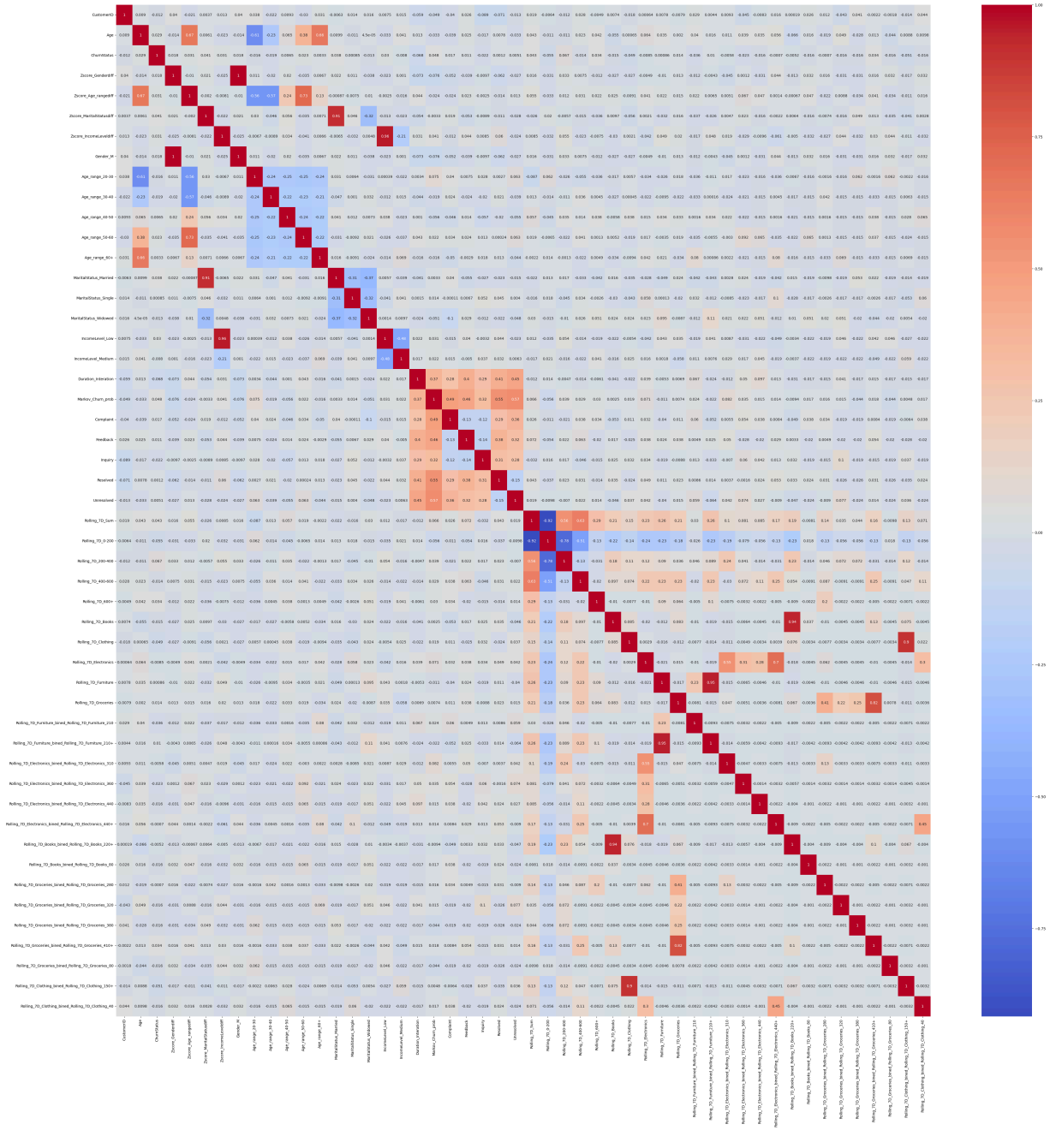
In [131…
```
test_count_endcoded.columns
```

Out[131…
```
Index(['CustomerID', 'Rolling_7D_Sum', 'ChurnStatus', 'Rolling_7D_0-200',
       'Rolling_7D_200-400', 'Rolling_7D_400-600', 'Rolling_7D_600+',
       'Rolling_7D_Books', 'Rolling_7D_Clothing', 'Rolling_7D_Electronics',
       'Rolling_7D_Furniture', 'Rolling_7D_Groceries',
       'Rolling_7D_Furniture_bined_Rolling_7D_Furniture_210',
       'Rolling_7D_Furniture_bined_Rolling_7D_Furniture_210+',
       'Rolling_7D_Electronics_bined_Rolling_7D_Electronics_310',
       'Rolling_7D_Electronics_bined_Rolling_7D_Electronics_360',
       'Rolling_7D_Electronics_bined_Rolling_7D_Electronics_440',
       'Rolling_7D_Electronics_bined_Rolling_7D_Electronics_440+',
       'Rolling_7D_Books_bined_Rolling_7D_Books_220+',
       'Rolling_7D_Books_bined_Rolling_7D_Books_80',
       'Rolling_7D_Groceries_bined_Rolling_7D_Groceries_280',
       'Rolling_7D_Groceries_bined_Rolling_7D_Groceries_320',
       'Rolling_7D_Groceries_bined_Rolling_7D_Groceries_380',
       'Rolling_7D_Groceries_bined_Rolling_7D_Groceries_410+',
       'Rolling_7D_Groceries_bined_Rolling_7D_Groceries_80',
       'Rolling_7D_Clothing_bined_Rolling_7D_Clothing_150+',
       'Rolling_7D_Clothing_bined_Rolling_7D_Clothing_40'],
      dtype='object')
```

In [134…
```
enhanced_Interaction_Customer_Transactions = enhanced_Interaction_Customer.m
```

In [135…
```
corr_df = enhanced_Interaction_Customer_Transactions.corr()
plt.figure(figsize=(50, 50))
sns.heatmap(corr_df, annot=True, cmap='coolwarm')
```

Out[135…
```
<Axes: >
```



Online Activity Sheet

In [137…
```
Online_Activity = data['Online_Activity']
```

In [138…
```
Online_Activity = Online_Activity.merge(data['Churn_Status'], how = "left",
```

In [139…
```
Online_Activity
```

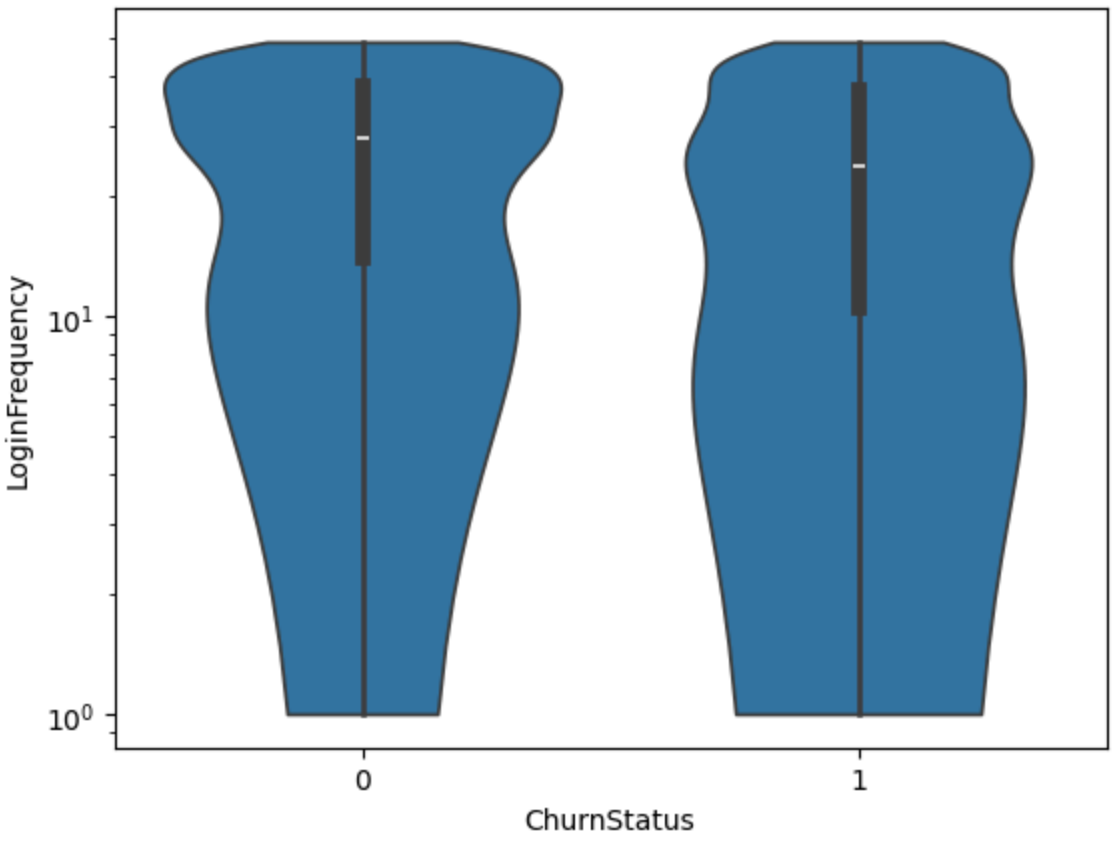
Out[139…

	CustomerID	LastLoginDate	LoginFrequency	ServiceUsage	ChurnStatus
0	1	2023-10-21	34	Mobile App	0
1	2	2023-12-05	5	Website	1
2	3	2023-11-15	3	Website	0
3	4	2023-08-25	2	Website	0
4	5	2023-10-27	41	Website	0
...	...	...	...	...	...
995	996	2023-01-29	38	Mobile App	0
996	997	2023-04-01	5	Mobile App	0
997	998	2023-07-10	47	Website	0
998	999	2023-01-08	23	Website	0
999	1000	2023-08-13	22	Mobile App	0

1000 rows × 5 columns

In [140…

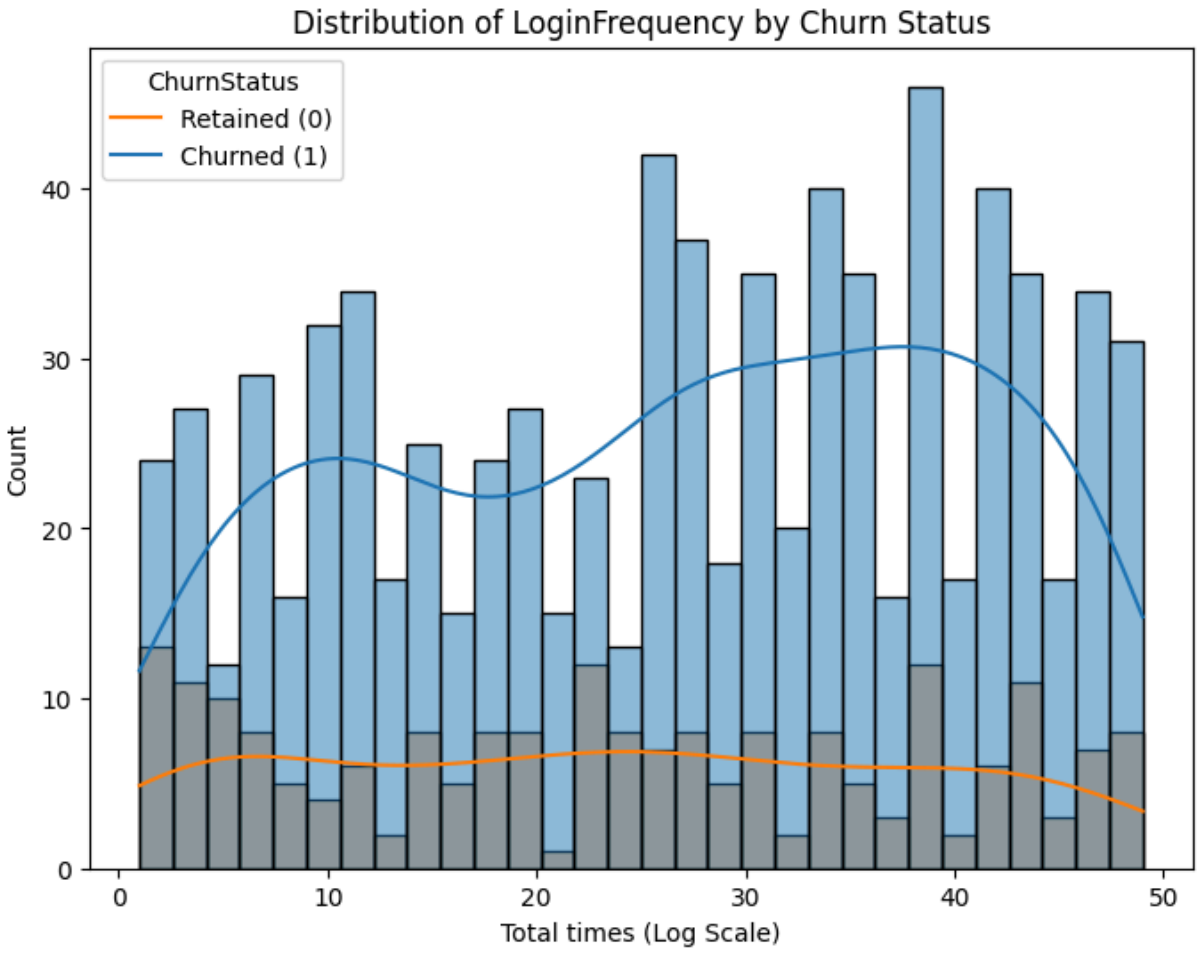
```
sns.violinplot(x="ChurnStatus" , y= "LoginFrequency"
               , data=Online_Activity, cut = 0)
plt.yscale("log")
```



In [141…

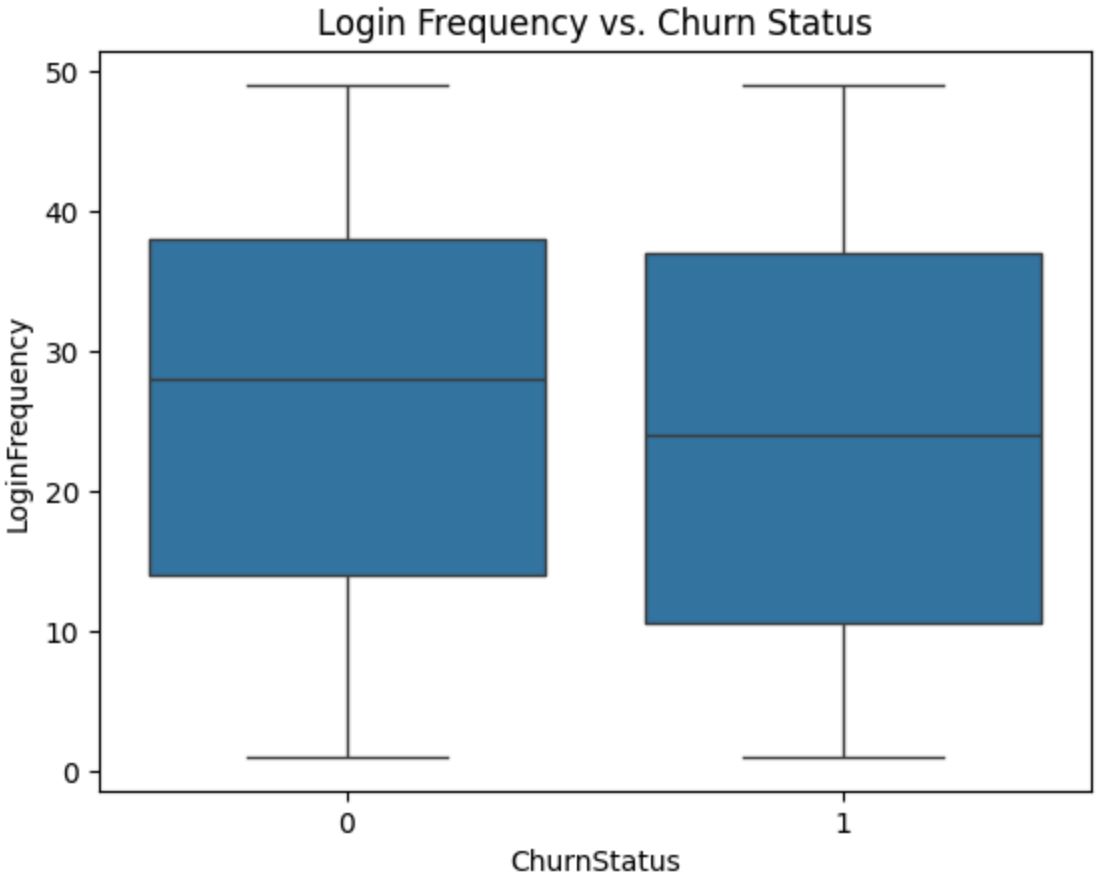
```
plt.figure(figsize=(8,6))
```

```
sns.histplot(data=Online_Activity, x="LoginFrequency", hue="ChurnStatus",
             bins=30, kde=True, log_scale=(False, False), alpha=0.5)

plt.title("Distribution of LoginFrequency by Churn Status")
plt.xlabel("Total times (Log Scale)")
plt.ylabel("Count")
plt.legend(title="ChurnStatus", labels=["Retained (0)", "Churned (1)"])
plt.show()
```



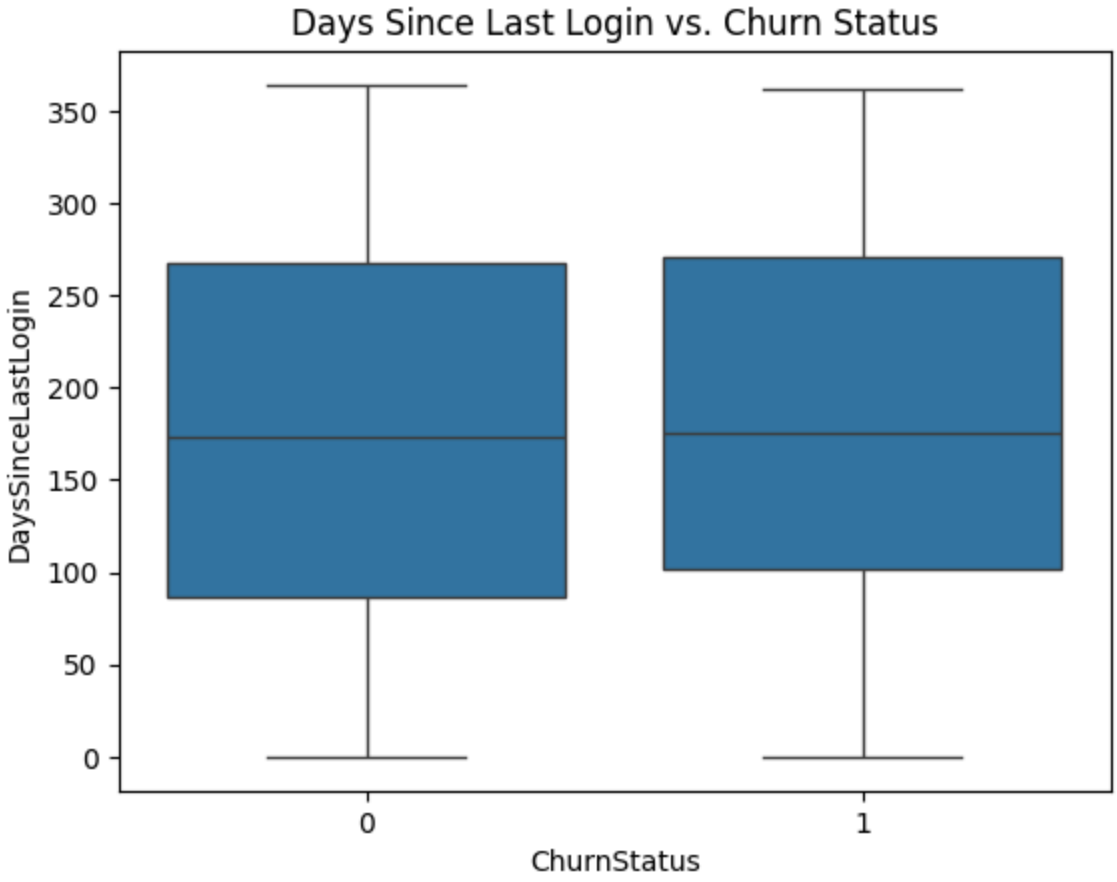
In [142…

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(x="ChurnStatus", y="LoginFrequency", data=Online_Activity)
plt.title("Login Frequency vs. Churn Status")
plt.show()
```



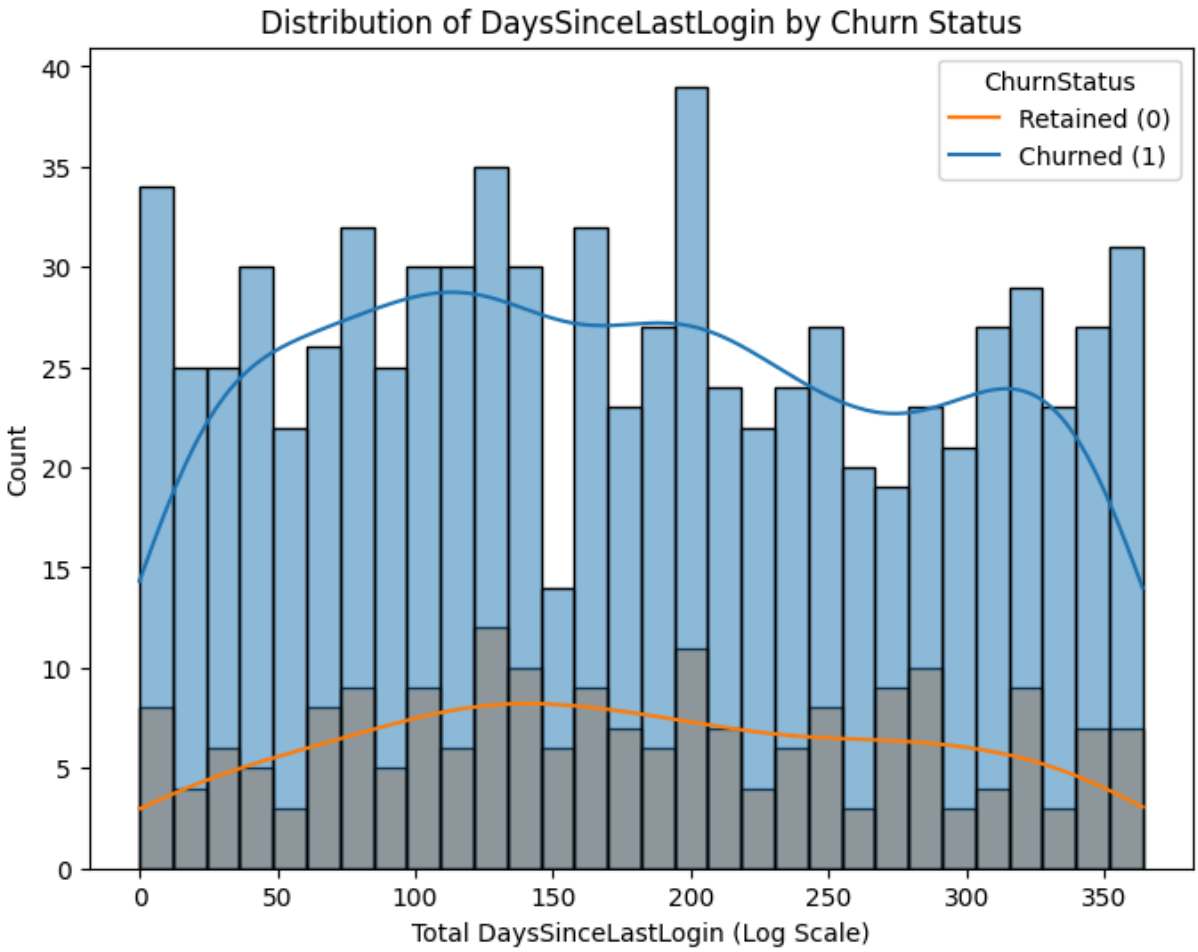
In [143…

```
from datetime import datetime

Online_Activity["DaysSinceLastLogin"] = (Online_Activity["LastLoginDate"].ma

sns.boxplot(x="ChurnStatus", y="DaysSinceLastLogin", data=Online_Activity)
plt.title("Days Since Last Login vs. Churn Status")
plt.show()
```



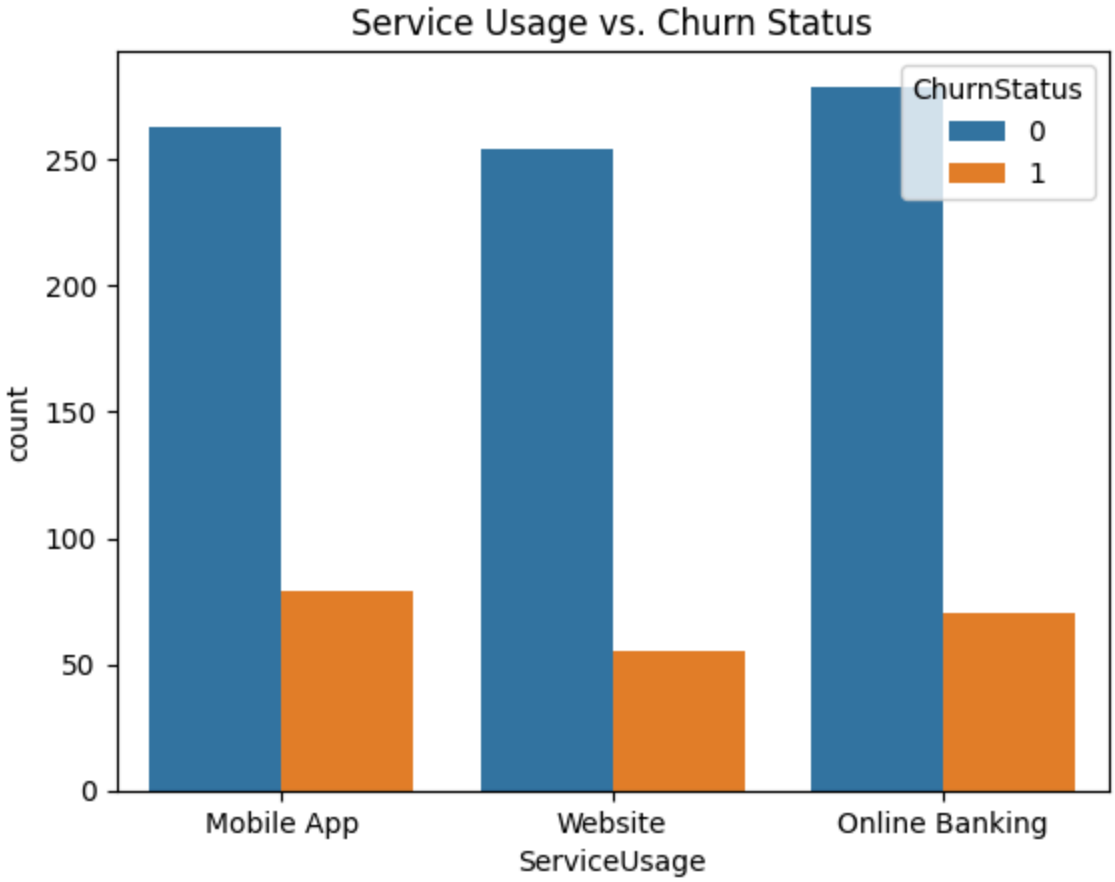
In [144…

```
plt.figure(figsize=(8,6))

sns.histplot(data=Online_Activity, x="DaysSinceLastLogin", hue="ChurnStatus"
             bins=30, kde=True, log_scale=(False, False), alpha=0.5)

plt.title("Distribution of DaysSinceLastLogin by Churn Status")
plt.xlabel("Total DaysSinceLastLogin (Log Scale)")
plt.ylabel("Count")
plt.legend(title="ChurnStatus", labels=["Retained (0)", "Churned (1)"])
plt.show()
```



In [145…

```
sns.countplot(x="ServiceUsage", hue="ChurnStatus", data=Online_Activity)
plt.title("Service Usage vs. Churn Status")
plt.show()
```



In [146…
```
import scipy.stats as stats

t_stat, p_value = stats.ttest_ind(
    Online_Activity.loc[Online_Activity["ChurnStatus"] == 1, "LoginFrequency
    Online_Activity.loc[Online_Activity["ChurnStatus"] == 0, "LoginFrequency
)
print(f"T-test for LoginFrequency: p-value = {p_value:.4f}")
```

```
T-test for LoginFrequency: p-value = 0.0098
```

In [104…
```
import scipy.stats as stats


t_stat, p_value = stats.ttest_ind(
    Online_Activity.loc[Online_Activity["ChurnStatus"] == 1, "DaysSinceLastL
    Online_Activity.loc[Online_Activity["ChurnStatus"] == 0, "DaysSinceLastL
)
print(f"T-test for DaysSinceLastLogin_Normalized: p-value = {p_value:.4f}")
```

```
T-test for DaysSinceLastLogin_Normalized: p-value = 0.7749
```

In [105…
```
import pandas as pd
import scipy.stats as stats

contingency_table = pd.crosstab(Online_Activity["ServiceUsage"], Online_Acti

# Kai^2 test
chi2, p, dof, expected = stats.chi2_contingency(contingency_table)
```

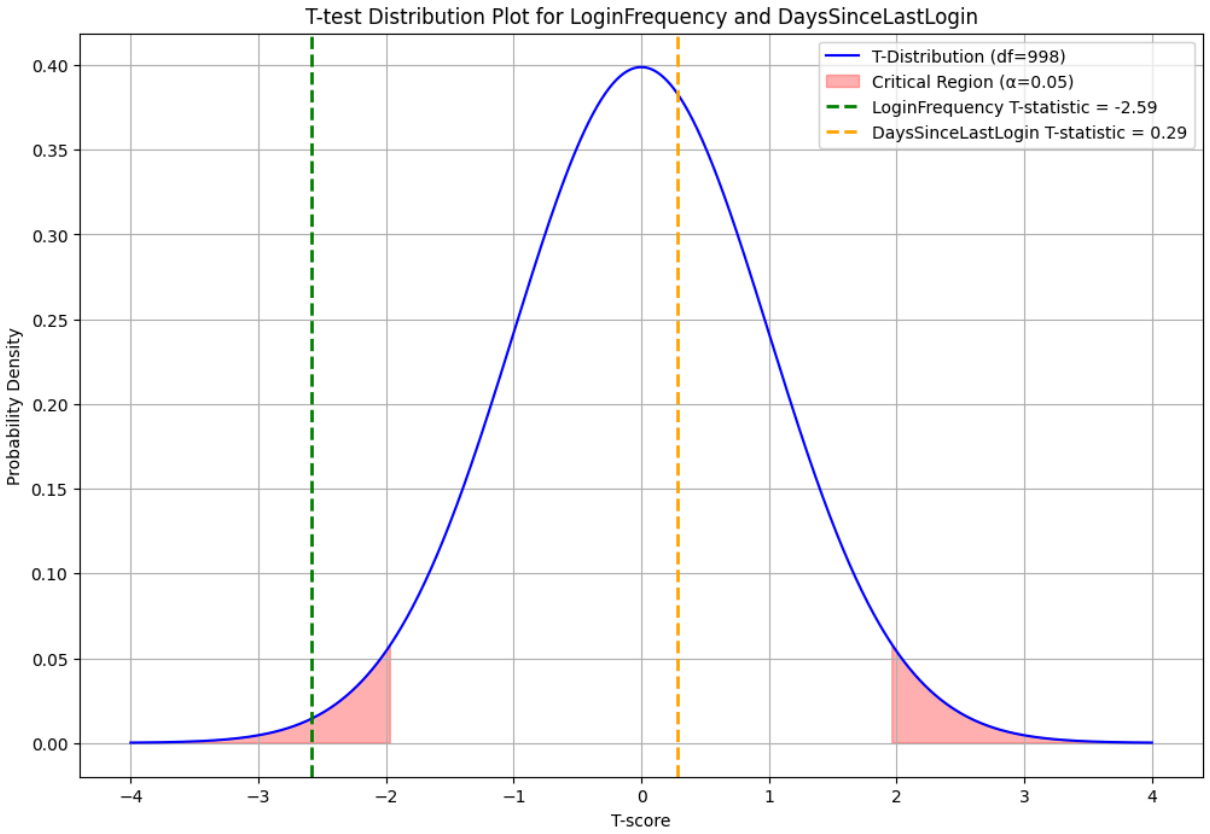
```
print(f"Kai squared statistics: {chi2:.4f}")
print(f"P value: {p:.4f}")


alpha = 0.05
if p < alpha:
    print("ServiceUsage and ChurnStatus have obvious correlation")
else:
    print("ServiceUsage and ChurnStatus have no obvious correlation")
```

```
Kai squared statistics: 2.8469
P value: 0.2409
ServiceUsage and ChurnStatus have no obvious correlation
```

In [160…

```
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt


churned = Online_Activity.loc[Online_Activity["ChurnStatus"] == 1, "LoginFre
retained = Online_Activity.loc[Online_Activity["ChurnStatus"] == 0, "LoginFr

churned1 = Online_Activity.loc[Online_Activity["ChurnStatus"] == 1, "DaysSir
retained1 = Online_Activity.loc[Online_Activity["ChurnStatus"] == 0, "DaysSi


n1 = len(churned)
n2 = len(retained)
df = n1 + n2 - 2


x = np.linspace(-4, 4, 1000)
y = stats.t.pdf(x, df)


t_stat, p_value = stats.ttest_ind(churned, retained)
t_stat1, p_value1 = stats.ttest_ind(churned1, retained1)


plt.figure(figsize=(12, 8))
plt.plot(x, y, label=f"T-Distribution (df={df})", color='blue')


alpha = 0.05
t_critical = stats.t.ppf(1 - alpha / 2, df)
plt.fill_between(x, y, where=(x < -t_critical) | (x > t_critical), color='re


plt.axvline(t_stat, color='green', linestyle='dashed', linewidth=2, label=f"
plt.axvline(t_stat1, color='orange', linestyle='dashed', linewidth=2, label=


plt.title("T-test Distribution Plot for LoginFrequency and DaysSinceLastLogi
plt.xlabel("T-score")
plt.ylabel("Probability Density")
```

```
plt.legend()
plt.grid()
plt.savefig("Plots/T_test_Distribution.png", dpi=300, bbox_inches='tight')

plt.show()

print(f"T-statistic: {t_stat:.4f}, p-value: {p_value:.4f}")
```

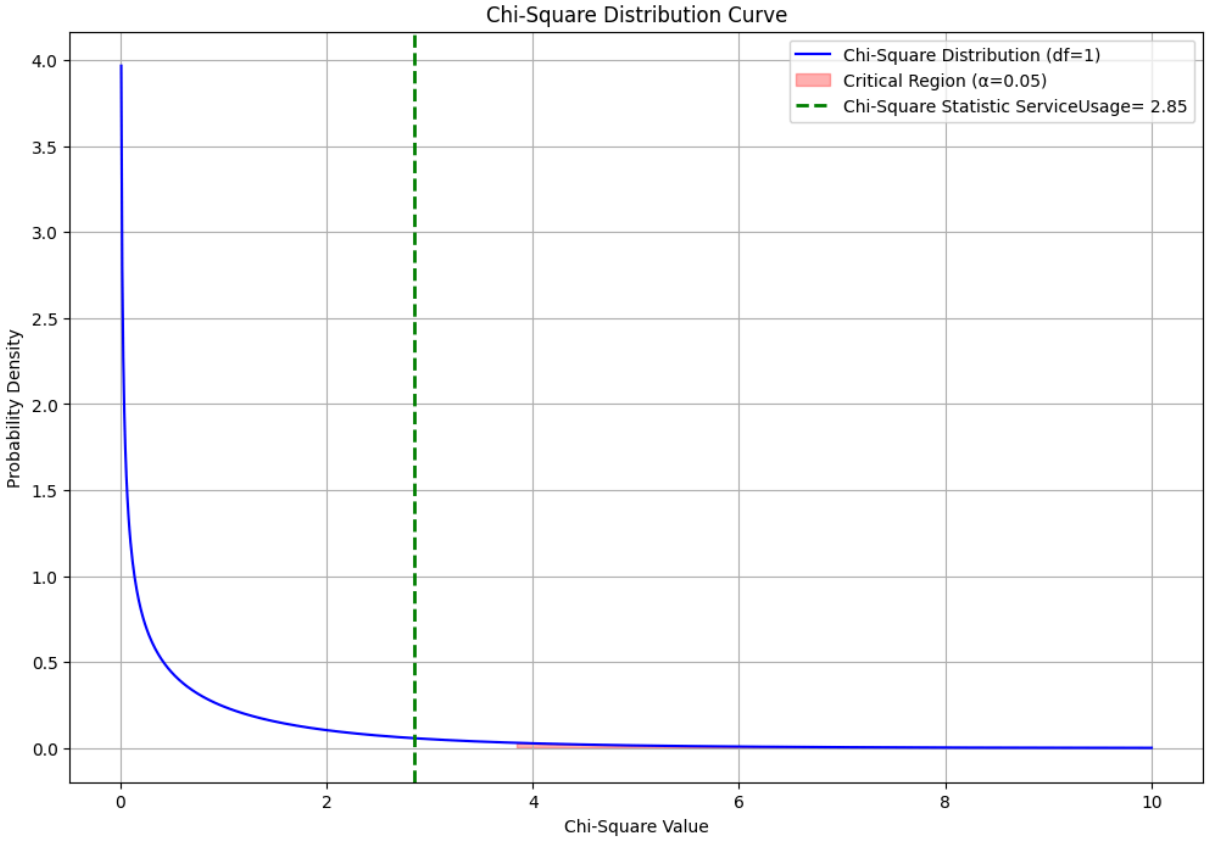



```
T-statistic: -2.5870, p-value: 0.0098
```

In [159…

```
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt


df = 1


x = np.linspace(0, 10, 1000)
y = stats.chi2.pdf(x, df)


contingency_table = pd.crosstab(Online_Activity["ServiceUsage"], Online_Acti


chi2_stat, p, dof, expected = stats.chi2_contingency(contingency_table)



plt.figure(figsize=(12, 8))
plt.plot(x, y, label=f"Chi-Square Distribution (df={df})", color='blue')
```

```
alpha = 0.05
chi2_critical = stats.chi2.ppf(1 - alpha, df)
plt.fill_between(x, y, where=(x > chi2_critical), color='red', alpha=0.3, la


plt.axvline(chi2_stat, color='green', linestyle='dashed', linewidth=2, label


plt.title("Chi-Square Distribution Curve")
plt.xlabel("Chi-Square Value")
plt.ylabel("Probability Density")
plt.legend()
plt.grid()
plt.savefig("Plots/Chi-Square_ServiceUsage.png", dpi=300, bbox_inches='tight

plt.show()
```



In [170…
```
from scipy.stats import spearmanr
corr, p_value = spearmanr(enhanced_Interaction_Customer_Transaction_Login["A
print(f"Spearman Correlation: {corr:.4f}, p-value: {p_value:.4f}")
```

```
Spearman Correlation: 0.6096, p-value: 0.0000
```

In [162…
```
Online_Activity["LoginFrequency_Ratio"] = Online_Activity["LoginFrequency"]

Online_Activity = pd.get_dummies(Online_Activity,columns=["ServiceUsage"], c
```

In [164…
```
Online_Activity = Online_Activity.drop(columns=["LastLoginDate"])
```

In [165…
```
Online_Activity["LoginCategory"] = pd.cut(Online_Activity["LoginFrequency"],
```

In [166…
```
Online_Activity = pd.get_dummies(Online_Activity, columns=["LoginCategory"],
```

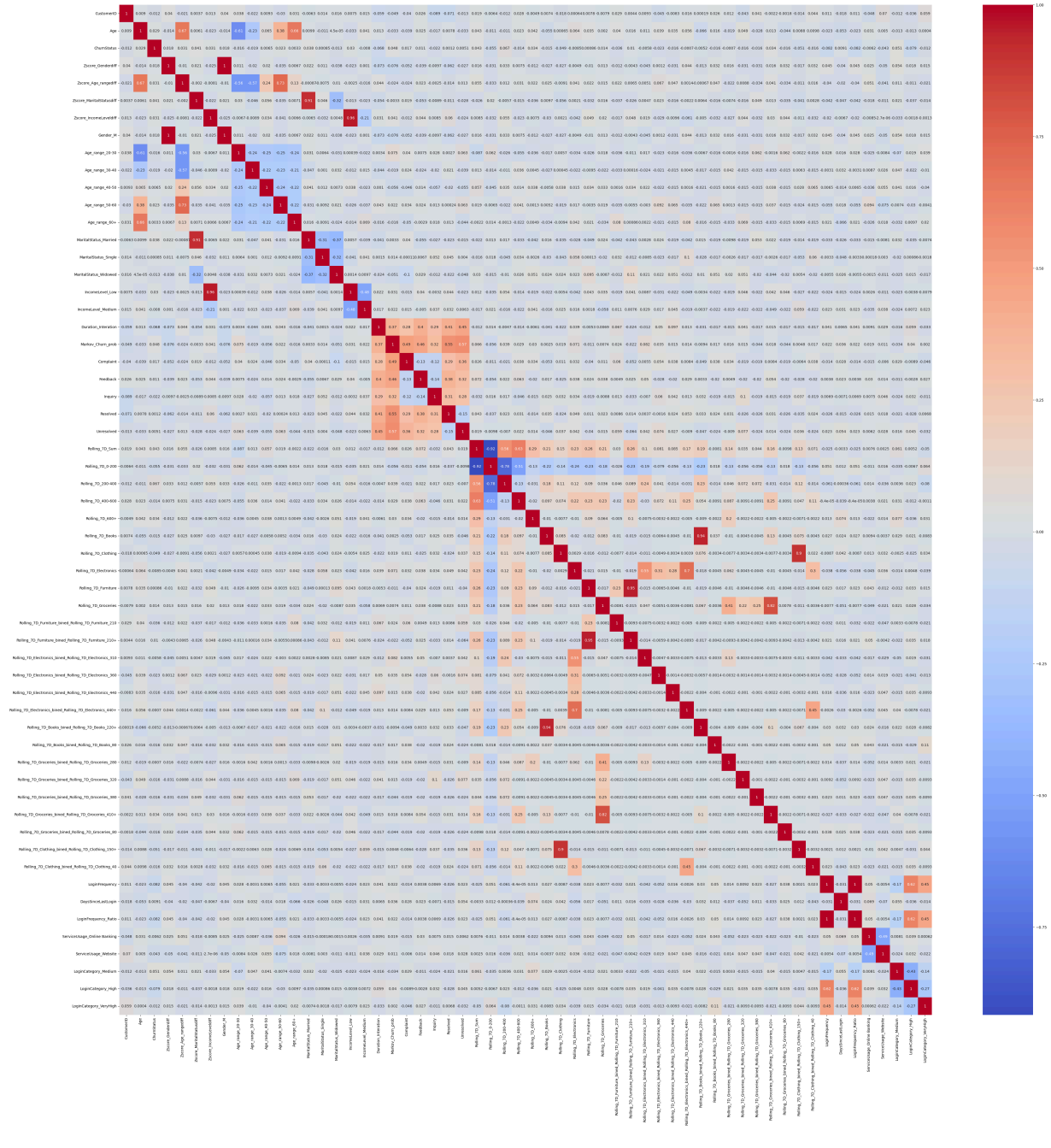
In [167…
```
enhanced_Interaction_Customer_Transaction_Login =  enhanced_Interaction_Cust
```

In [172…
```
for col in enhanced_Interaction_Customer_Transaction_Login.columns:
    try:
        enhanced_Interaction_Customer_Transaction_Login[col] = pd.to_numeric
    except:
        print(f"failed at {col}")
```

```
/var/folders/pt/0nf2m3pj1f3b373wsyfc6glh0000gn/T/ipykernel_52421/2591289443.
py:3: FutureWarning:

errors='ignore' is deprecated and will raise in a future version. Use to_num
eric without passing `errors` and catch exceptions explicitly instead
```

In [173…
```
enhanced_Interaction_Customer_Transaction_Login["Duration_Interation"] = enh
```

In [178…
```
corr = enhanced_Interaction_Customer_Transaction_Login.corr()

plt.figure(figsize=(50, 50))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.savefig("Plots/Correlation.png", dpi=300, bbox_inches='tight')
plt.show()
```



In [177…

```
enhanced_Interaction_Customer_Transaction_Login.to_csv("enhanced_customer_da
```