```
#Import the necessary files
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
# !pip install -q openpyxl  # Uncomment this line if openpyxl is not installed
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go

# Load the excel file
file_path = "customers.xlsx"
# Load all sheets
sheets = pd.read_excel(file_path, sheet_name=None)
# Assign sheets to variables (update sheet names correctly)
df_demo = sheets['Customer_Demographics']
df_trans = sheets['Transaction_History']
df_inter = sheets['Customer_Service']
df_login = sheets['Online_Activity']
df_churn = sheets['Churn_Status']
```

```
# Step 1: Standardize CustomerId across all DataFrames
for data in [df_demo, df_trans, df_inter, df_login, df_churn]:
    data["CustomerID"] = data["CustomerID"].astype(str).str.strip().str.upper()

# Step 2: Merge them one by one
customers = df_demo.merge(df_trans, on="CustomerID", how="left") \
                .merge(df_inter, on="CustomerID", how="left") \
                .merge(df_login, on="CustomerID", how="left") \
                .merge(df_churn, on="CustomerID", how="left")

# Step 3: Save result
customers.to_csv("customers.csv", index=False)
customers.head(5)
```

	C…			Marita…	Inc…	Transa…	TransactionDate	Am…
0	1	62	M	Single	Low	7194	2022-03-27T00:00:00.000	416.5
1	2	65	M	Married	Low	7250	2022-08-08T00:00:00.000	54.96
2	2	65	M	Married	Low	9660	2022-07-25T00:00:00.000	197.5
3	2	65	M	Married	Low	2998	2022-01-25T00:00:00.000	101.31
4	2	65	M	Married	Low	1228	2022-07-24T00:00:00.000	397.37

Rows: 5 Expand

```
# Check for missing values in each column of the 'customers' dataframe
customers.isnull().sum()
```

index
CustomerID
Age
Gender
MaritalStatus
IncomeLevel
TransactionID
TransactionDate
AmountSpent
ProductCategory
InteractionID
InteractionDate
InteractionType
ResolutionStatus
LastLoginDate
LoginFrequency
ServiceUsage

Rows: 17 Expand

Exploratory Data Analysis (EDA) on `customers` Dataset

This section provides a comprehensive exploratory data analysis (EDA) of the `customers` DataFrame. We will examine the structure, missing values, summary statistics, distributions of the data.

```
# Basic info and structure
display(customers.info())
#display(customers.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6812 entries, 0 to 6811
Data columns (total 17 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   CustomerID        6812 non-null   object
 1   Age               6812 non-null   int64
 2   Gender            6812 non-null   object
 3   MaritalStatus     6812 non-null   object
 4   IncomeLevel       6812 non-null   object
 5   TransactionID     6812 non-null   int64
 6   TransactionDate   6812 non-null   datetime64[ns]
 7   AmountSpent       6812 non-null   float64
 8   ProductCategory   6812 non-null   object
 9   InteractionID     5204 non-null   float64
 10  InteractionDate   5204 non-null   datetime64[ns]
 11  InteractionType   5204 non-null   object
 12  ResolutionStatus  5204 non-null   object
 13  LastLoginDate     6812 non-null   datetime64[ns]
 14  LoginFrequency    6812 non-null   int64
 15  ServiceUsage      6812 non-null   object
 16  ChurnStatus       6812 non-null   int64
dtypes: datetime64[ns](3), float64(2), int64(4), object(8)
memory usage: 904.8+ KB
```

```
None
```

```
#Featur engineering. Convert the null into Flag
customers["HasInteraction"] = customers["InteractionID"].notnull().astype(int)
customers.head()
```

	C...			Marita...	Inc...	Transa...	TransactionDate	Am...
0	1	62	M	Single	Low	7194	2022-03-27T00:00:00.000	416.5
1	2	65	M	Married	Low	7250	2022-08-08T00:00:00.000	54.96
2	2	65	M	Married	Low	9660	2022-07-25T00:00:00.000	197.5
3	2	65	M	Married	Low	2998	2022-01-25T00:00:00.000	101.31
4	2	65	M	Married	Low	1228	2022-07-24T00:00:00.000	397.37

Rows: 5 Expand

Investigating Customer Churn

The goal of this EDA is to identify potential factors that contribute to customer churn. We will analyze the relationships between `ChurnStatus` and various customer attributes, including demographics, transaction behavior, and service usage. This will help us uncover patterns and insights that may explain why customers churn.

```
# Customer interact more than once. So it will be good to group them by CustomerID
customer_level = customers.groupby("CustomerID").agg({
    "Gender": "first",
    "MaritalStatus": "first",
    "IncomeLevel": "first",
    "Age": "first",
    "AmountSpent": "sum",
    "LoginFrequency": "sum",
    "HasInteraction": "max",
    "ProductCategory": "first",
    # Clean ServiceUsage: join unique values as comma-separated string
    "ServiceUsage": lambda x: ', '.join(sorted(set(x.dropna()))),
    "InteractionType": lambda x: ', '.join(sorted(set(x.dropna()))),
    "ResolutionStatus": "max",
    "ChurnStatus": "max"
}).reset_index()

# Map churn label
churn_map = {0: "Active", 1: "Churned"}
customer_level["ChurnStatusLabel"] = customer_level["ChurnStatus"].map(churn_map)

customer_level
```

	C...		Marita...	Inc...		Am...	LoginFre...	HasInter...	ProductC...
0	1	M	Single	Low	62	416.5	34	1	Electronics
1	10	M	Married	High	68	1397.36	203	0	Furniture
2	100	F	Married	Low	41	2217.12	360	1	Clothing
3	1000	M	Widowed	Low	34	1670.79	132	0	Furniture
4	101	F	Single	High	33	984.7	70	1	Groceries
5	102	M	Married	High	31	4059.92	384	1	Clothing
6	103	M	Widowed	Low	39	4595.46	828	1	Clothing
7	104	F	Widowed	Medium	66	5333.68	54	1	Electronics
8	105	F	Widowed	Low	67	3727.2	480	1	Furniture
9	106	M	Married	High	23	1750.16	294	1	Furniture
10	107	F	Married	High	59	1542.76	42	0	Clothing
11	108	M	Divorced	Low	53	988.62	4	1	Books
12	109	M	Married	Medium	18	2549.4	240	1	Furniture
13	11	M	Divorced	Medium	54	3467.66	300	1	Books
14	110	M	Married	Medium	49	1446	150	1	Clothing
15	111	F	Widowed	High	23	613.18	66	1	Furniture

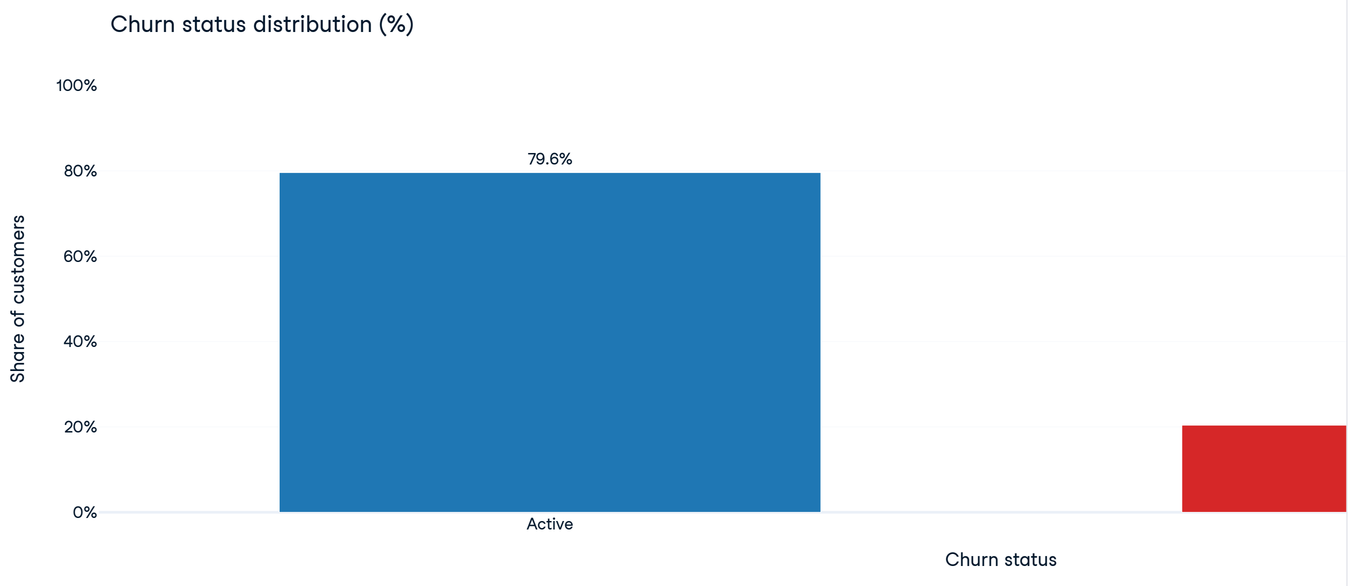
Rows: 1,000 Expand

```
#Replace the missing value and ampty column
def fix_resolution(row):
    if (pd.isnull(row["ResolutionStatus"]) or str(row["ResolutionStatus"]).strip() == "") and row["HasInteraction"] == 0:
        return "No Interaction"
    return row["ResolutionStatus"]

def fix_interaction(row):
    if (pd.isnull(row["InteractionType"]) or str(row["InteractionType"]).strip() == "") and row["HasInteraction"] == 0:
        return "No Interaction"
    return row["InteractionType"]

customer_level["ResolutionStatus"] = customer_level.apply(fix_resolution, axis=1)
customer_level["InteractionType"] = customer_level.apply(fix_interaction, axis=1)
customer_level.to_csv("customer_level.csv",index=False)

customer_level
```

	C...		Marita...	Inc...		Am...	LoginFre...	HasInter...	ProductC...
0	1	M	Single	Low	62	416.5	34	1	Electronics
1	10	M	Married	High	68	1397.36	203	0	Furniture
2	100	F	Married	Low	41	2217.12	360	1	Clothing
2	100	F	Married	Low	41	2217.12	360	1	Clothing
3	1000	M	Widowed	Low	34	1670.79	132	0	Furniture
4	101	F	Single	High	33	984.7	70	1	Groceries
4	101	F	Single	High	33	984.7	70	1	Groceries
5	102	M	Married	High	31	4059.92	384	1	Clothing
5	102	M	Married	High	31	4059.92	384	1	Clothing
6	103	M	Widowed	Low	39	4595.46	828	1	Clothing
7	104	F	Widowed	Medium	66	5333.68	54	1	Electronics
8	105	F	Widowed	Low	67	3727.2	480	1	Furniture
9	106	M	Married	High	23	1750.16	294	1	Furniture
9	106	M	Married	High	23	1750.16	294	1	Furniture
10	107	F	Married	High	59	1542.76	42	0	Clothing
11	108	M	Divorced	Low	53	988.62	4	1	Books

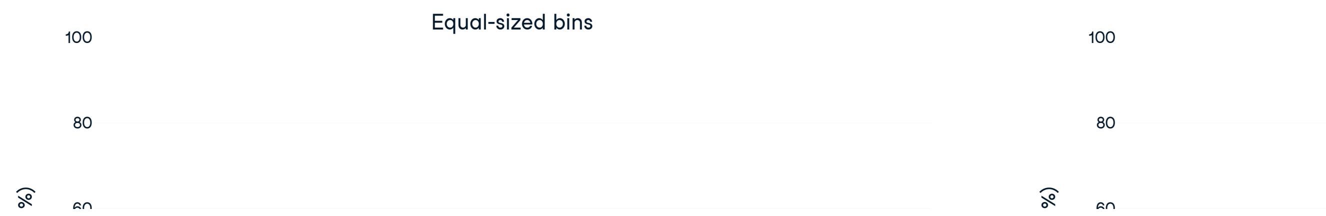
Rows: 1,215 Expand

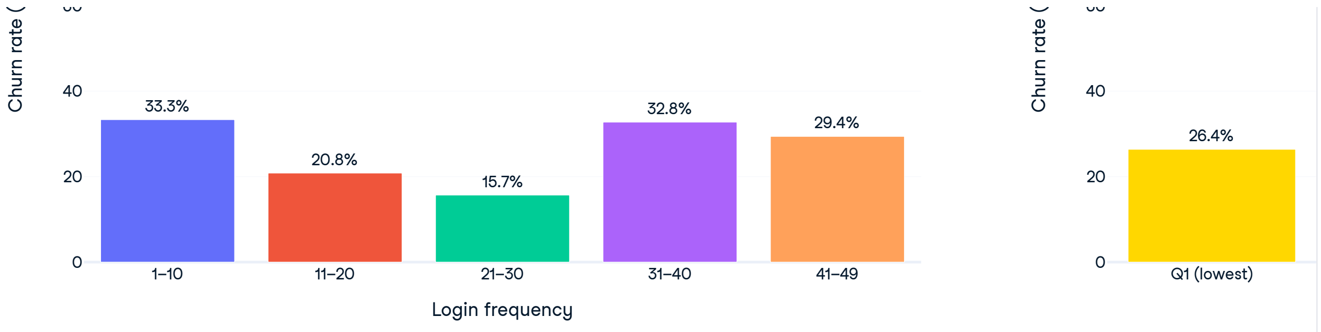
```python
#Calculate churn distribution
churn_pct = customer_level['ChurnStatus'].value_counts(normalize=True).sort_index().reset_index()
churn_pct.columns = ['ChurnStatus', 'Rate']

# Map to readable labels
churn_pct['ChurnLabel'] = churn_pct['ChurnStatus'].map({0: 'Active', 1: 'Churned'})

# Bar chart
fig = px.bar(
    churn_pct,
    x='ChurnLabel',
    y='Rate',               # keep as fraction
    color='ChurnLabel',     # color by label
    color_discrete_map={'Active': '#1f77b4', 'Churned': '#d62728'},  # blue vs red
    text='Rate',
    labels={'ChurnLabel': 'Churn status', 'Rate': 'Share of customers'},
    title='Churn status distribution (%)'
)

# Format labels as percentages and ensure visibility
fig.update_traces(texttemplate='%{text:.1%}', textposition='outside', cliponaxis=False)

# Format y-axis as percent and set sensible range
fig.update_yaxes(tickformat='.0%', range=[0, 1])  # 0-100%

# Increase figure size and margins to avoid clipping
fig.update_layout(
    autosize=True,
    height=500,
    margin=dict(l=60, r=60, t=80, b=100),
    bargap=0.4,
    uniformtext_minsize=12,
    uniformtext_mode='hide'
)

fig.show()
```



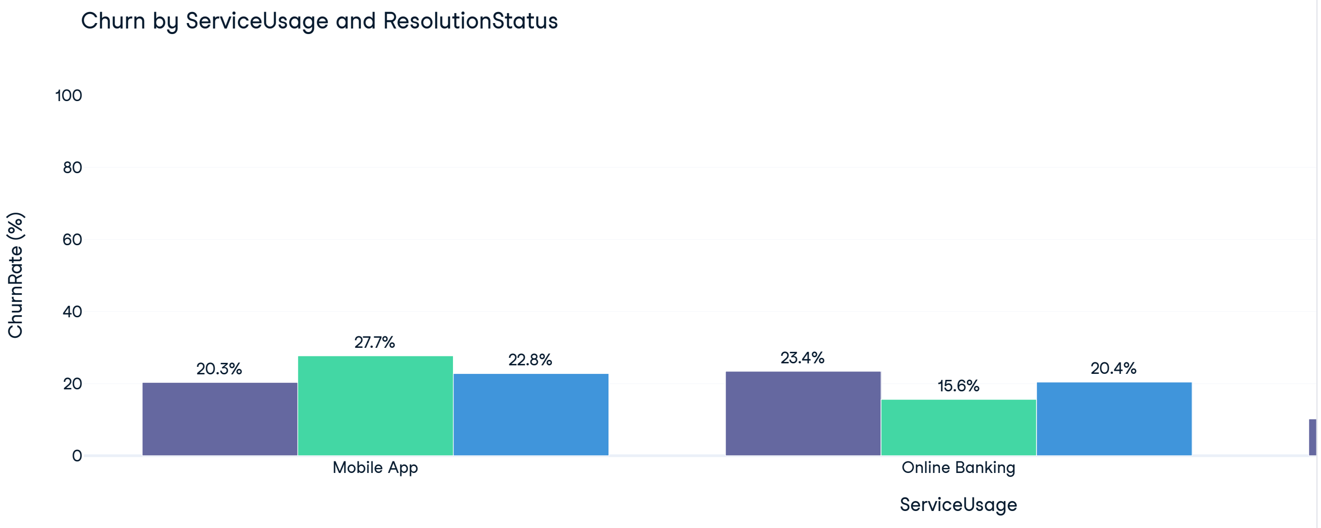
```
#Visualise how the login frequency impact the churn status. 2 ways will used . Equal bins and Quartile
# Define distinct colors for each bin/quartile
equal_bin_colors = ["#636EFA", "#EF553B", "#00CC96", "#AB63FA", "#FFA15A"]  # 5 colors for equal bins
quartile_colors = ["#FFD700", "#FF7F0E", "#2CA02C", "#1F77B4"]  # 4 colors for quartiles

# Equal-sized bins (based on min=1, max=49)
bins_equal = [1, 10, 20, 30, 40, 49] #Bins: equal length of time spend.  Use min and max to find interval
labels_equal = ["1-10", "11-20", "21-30", "31-40", "41-49"]
customer_level["LoginBin_equal"] = pd.cut(customer_level["LoginFrequency"], bins=bins_equal, labels=labels_equal,
include_lowest=True)

churn_equal = customer_level.groupby("LoginBin_equal")["ChurnStatus"].mean().reset_index()
churn_equal["ChurnRate (%)"] = churn_equal["ChurnStatus"] * 100

# Quartiles
#Quartiles: divide a set of numbers into four equal parts (25%)
customer_level["LoginBin_quartile"] = pd.qcut(customer_level["LoginFrequency"], q=4, labels=["Q1 (lowest)", "Q2", "Q3", "Q4
(highest)"])
churn_quartile = customer_level.groupby("LoginBin_quartile")["ChurnStatus"].mean().reset_index()
churn_quartile["ChurnRate (%)"] = churn_quartile["ChurnStatus"] * 100

# Create subplots
fig = make_subplots(rows=1, cols=2, subplot_titles=("Equal-sized bins", "Quartiles"))

# Equal bins bar chart (each bar a different color)
for i, row in churn_equal.iterrows():
    fig.add_trace(
        go.Bar(
            x=[row["LoginBin_equal"]],
            y=[row["ChurnRate (%)"]],
            text=[f"{row['ChurnRate (%)']:.1f}%"],
            textposition="outside",
            marker_color=equal_bin_colors[i % len(equal_bin_colors)],
            name=str(row["LoginBin_equal"]),
            showlegend=False
        ),
        row=1, col=1
    )

# Quartiles bar chart (each bar a different color)
for i, row in churn_quartile.iterrows():
    fig.add_trace(
        go.Bar(
            x=[row["LoginBin_quartile"]],
            y=[row["ChurnRate (%)"]],
            text=[f"{row['ChurnRate (%)']:.1f}%"],
            textposition="outside",
            marker_color=quartile_colors[i % len(quartile_colors)],
            name=str(row["LoginBin_quartile"]),
            showlegend=False
        ),
        row=1, col=2
    )

# Layout
fig.update_yaxes(title_text="Churn rate (%)", range=[0, 100], row=1, col=1)
fig.update_yaxes(title_text="Churn rate (%)", range=[0, 100], row=1, col=2)
fig.update_xaxes(title_text="Login frequency", row=1, col=1)
fig.update_xaxes(title_text="Login frequency ", row=1, col=2)
fig.update_layout(
    title_text="Churn by Login Frequency: Equal Bins vs Quartiles (Customer Level)",
    height=500
)

fig.show()
```

Churn by Login Frequency: Equal Bins vs Quartiles (Customer Level)

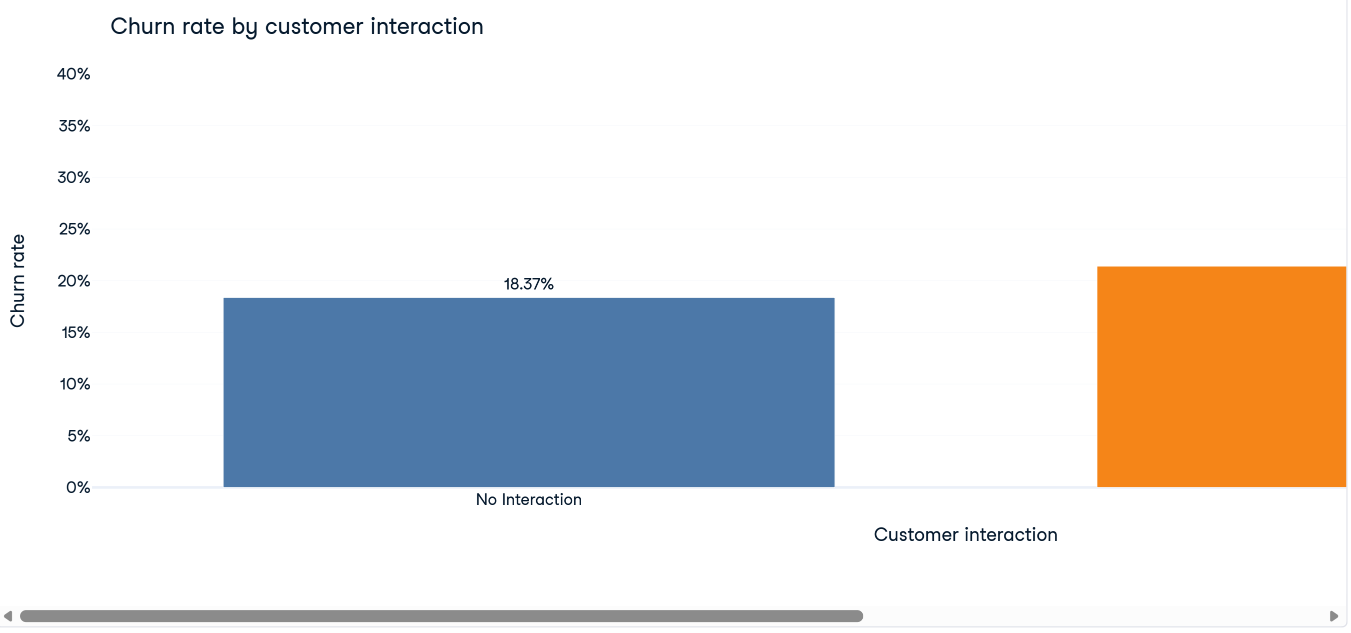




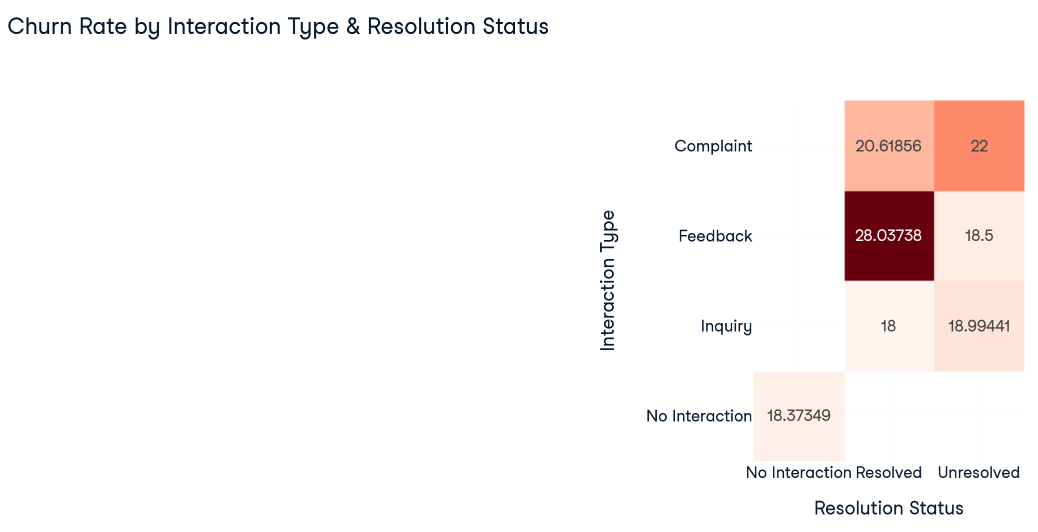
```
#Churn by ServiceUsage and ResolutionStatus
churn_combo = (
    customer_level
    .groupby(["ServiceUsage", "ResolutionStatus"])["ChurnStatus"]
    .mean()
    .reset_index()
)
churn_combo["ChurnRate (%)"] = churn_combo["ChurnStatus"] * 100

fig = px.bar(
    churn_combo,
    x="ServiceUsage",
    y="ChurnRate (%)",
    color="ResolutionStatus",
    barmode="group",
    text="ChurnRate (%)",
    title="Churn by ServiceUsage and ResolutionStatus"
)
fig.update_traces(texttemplate='%{text:.1f}%', textposition="outside", cliponaxis=False)
fig.update_yaxes(range=[0, 100])
fig.show()
```



```
# churn by interaction
churn_by_interaction = customer_level.groupby("HasInteraction")["ChurnStatus"].mean().reset_index()

# Map labels for clarity
interaction_map = {0: "No Interaction", 1: "Had Interaction"}
churn_by_interaction["InteractionLabel"] = churn_by_interaction["HasInteraction"].map(interaction_map)

# Bar chart
fig = px.bar(
    churn_by_interaction,
    x="InteractionLabel",
    y="ChurnStatus",  # keep as fraction
    color="InteractionLabel",
    color_discrete_map={"No Interaction": "#4C78A8", "Had Interaction": "#F58518"},
    text="ChurnStatus",
    labels={"InteractionLabel": "Customer interaction", "ChurnStatus": "Churn rate"},
    title="Churn rate by customer interaction"
)

# Show labels as percentages and avoid clipping
fig.update_traces(texttemplate='%{text:.2%}', textposition='outside', cliponaxis=False)

# Format axis as percent and set a sensible range (adjust upper bound if needed)
fig.update_yaxes(tickformat='.0%', range=[0, 0.4])

# Improve layout to prevent cut-off
fig.update_layout(
    margin=dict(l=40, r=40, t=60, b=80),
    bargap=0.3,
    uniformtext_minsize=12,
    uniformtext_mode='hide'
)

fig.show()
```



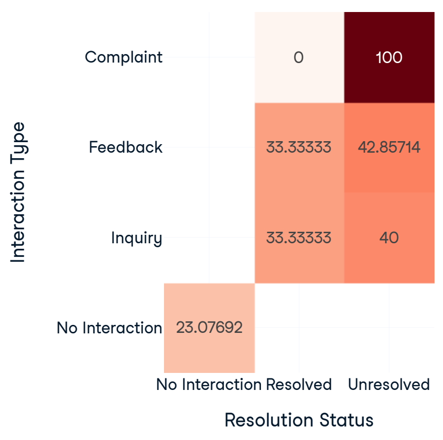
```
#Show the Churn Rate by Interaction Type & Resolution Status"
# --- Step 1: Normalize InteractionType ---
# Split multi-valued entries like "Complaint, Feedback" into separate rows
customer_level["InteractionType"] = customer_level["InteractionType"].str.split(",")
customer_level = customer_level.explode("InteractionType")
customer_level["InteractionType"] = customer_level["InteractionType"].str.strip()

# --- Step 2: Aggregate churn rates ---
churn_spend = (
    customer_level
    .groupby(["InteractionType", "ResolutionStatus"])
    .agg(ChurnRate=("ChurnStatus", "mean"))
    .reset_index()
)
churn_spend["ChurnRate (%)"] = churn_spend["ChurnRate"] * 100

# --- Step 3: Pivot into matrix for heatmap ---
heatmap_data = churn_spend.pivot(
    index="InteractionType",
    columns="ResolutionStatus",
    values="ChurnRate (%)"
)

# --- Step 4: Plot heatmap inline ---
fig = px.imshow(
    heatmap_data,
    text_auto=True,
    color_continuous_scale="Reds",
    title="Churn Rate by Interaction Type & Resolution Status",
    labels=dict(x="Resolution Status", y="Interaction Type", color="Churn Rate (%)")
)
fig.show()
```



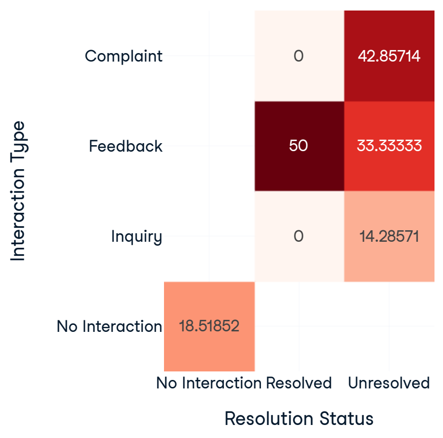
```
# --- Step 1: Create LoginFrequency bins ---
bins_equal = [1, 10, 20, 30, 40, 49]
labels_equal = ["1-10", "11-20", "21-30", "31-40", "41-49"]

customer_level["LoginBin_equal"] = pd.cut(
    customer_level["LoginFrequency"],
    bins=bins_equal,
    labels=labels_equal,
    include_lowest=True
)

# --- Step 2: Normalize InteractionType ---
customer_level["InteractionType"] = customer_level["InteractionType"].str.split(",")
customer_level = customer_level.explode("InteractionType")
customer_level["InteractionType"] = customer_level["InteractionType"].str.strip()

# --- Step 3: Aggregate churn rates ---
churn_bins = (
    customer_level
    .groupby(["InteractionType", "ResolutionStatus", "LoginBin_equal"])
    .agg(ChurnRate=("ChurnStatus", "mean"))
    .reset_index()
)
churn_bins["ChurnRate (%)"] = churn_bins["ChurnRate"] * 100
```

```
for bin_label in churn_bins["LoginBin_equal"].unique():
    heatmap_data = churn_bins[churn_bins["LoginBin_equal"] == bin_label].pivot(
        index="InteractionType",
        columns="ResolutionStatus",
        values="ChurnRate (%)"
    )

    fig = px.imshow(
        heatmap_data,
        text_auto=True,
        color_continuous_scale="Reds",
        title = f"Churn Rate by Interaction Type and Resolution Status - Login Frequency between {bin_label} hours",
        labels=dict(x="Resolution Status", y="Interaction Type", color="Churn Rate (%)")
    )
    fig.show()
```

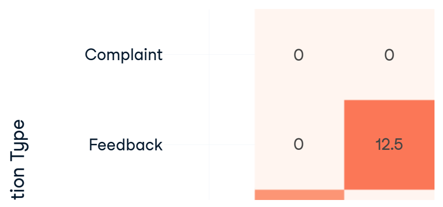
Churn Rate by Interaction Type and Resolution Status - Login Frequency between 1–10 hours

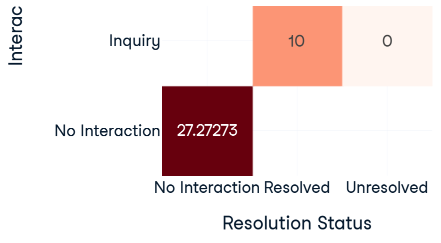


Churn Rate by Interaction Type and Resolution Status - Login Frequency between 11–20 hours

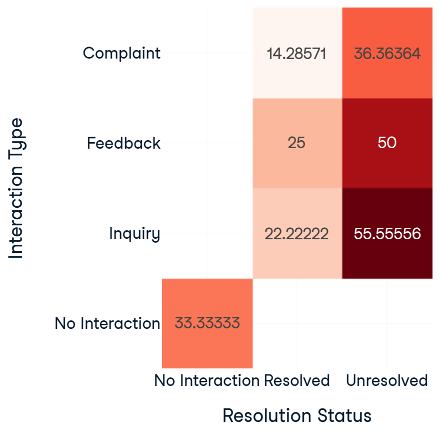


Churn Rate by Interaction Type and Resolution Status - Login Frequency between 21–30 hours

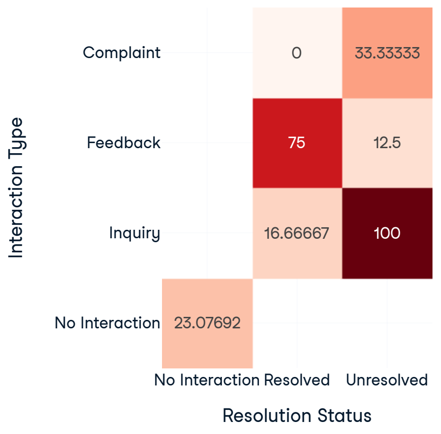




Churn Rate by Interaction Type and Resolution Status - Login Frequency between 31–40 hours



Churn Rate by Interaction Type and Resolution Status - Login Frequency between 41–49 hours



Data Gathering, EDA, and Data Cleaning

This project uses a customer dataset containing five interconnected tables: Customer Demographics, Transaction History, Customer Service Interactions, Online Activity, and Churn Status. These datasets were selected because together they provide a complete view of customer behaviour, demographics, engagement, and service experience—factors that are all relevant to understanding and predicting churn.

Exploratory Data Analysis (EDA) Initial exploration included reviewing the structure of each dataset, checking for missing values, and examining summary statistics. Visualisations were used to analyse the distribution of churn and its relationship with key customer attributes. The results showed that customers with low login frequency tend to churn more, unresolved customer service issues correspond to higher churn, and customers with no interactions also show elevated churn, suggesting a link between low engagement and churn risk. Heatmaps further revealed that unresolved complaint interactions are strongly associated with higher churn rates.

Data Cleaning and Preprocessing CustomerID fields were standardised to ensure accurate merging across all tables. The five datasets were then combined into a single unified table. Missing values related to service interactions were replaced with “No Interaction” for customers who never contacted customer service. Multi-value fields such as interaction type were cleaned and separated so they could be analysed correctly. A new feature, “HasInteraction”, was created to flag whether a customer had ever interacted with customer service.

Because some customers had multiple transactions or service events, the dataset was aggregated to a customer-level view. This included summing their total spending, total login frequency, capturing their service experience, and retaining a single churn label. Categorical values were cleaned and simplified, and churn status was mapped into both numerical and readable labels.

Final Output A fully cleaned, consolidated customer-level dataset was produced. It includes demographic details, behavioural metrics, service usage indicators, and churn status. This dataset is now ready for model building and further analysis.

Churn Insights by Interaction Type and Login Frequency

Complaints • Low login (1–10 hrs) + Unresolved → churn ~22% • High login (41–49 hrs) + Unresolved → churn ~19% ***High login customers churn slightly less, but unresolved complaints remain a concern.***

Feedback • Low login (1–10 hrs) + Resolved → churn ~28% • High login (41–49 hrs) + Unresolved → churn ~15% ***Low login feedback is a warning sign: they churn more. High login feedback customers are more resilient.***

Inquiries • Low login (1–10 hrs) + Unresolved → churn ~19% • High login (41–49 hrs) + Unresolved → churn ~9% ***Unresolved inquiries hurt low login customers more, while high login customers stay loyal.***

No Interaction • Across all logins → churn ~18–33% ***Lack of interaction is a risk factor but consistent across login frequencies.***

** Key Takeaways**

• Engagement reduces churn: High login customers churn less even when issues are unresolved. • Critical risk group: Low login + unresolved interactions have the highest churn. • Focus area: Complaints and feedback require attention, especially for low login customers.