

```
In [33]: # To create virtual env -python -m venv .venv
# Activate virtual env - ..\venv\Scripts\activate
# Install dependencies using the requirements.txt
# pip install -r requirements.txt
```

```
In [34]: import pandas as pd
import numpy as np
import streamlit as st
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.io as pio
pio.renderers.default = "notebook_connected"
```

```
In [35]: #Load the data
df = pd.read_csv("../data/processed/Data_Science_Challenge.csv")
```

```
In [36]: pd.set_option('display.max_columns', None)
df.head()
```

Out[36]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	t ch
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10.0	3	
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3	
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7.32	12.2	5	
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	61.9	88	5.26	196.9	89	8.86	6.6	7	
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3	

```
In [37]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   state            3333 non-null    object 
 1   account length  3333 non-null    int64  
 2   area code        3333 non-null    int64  
 3   phone number    3333 non-null    object 
 4   international plan 3333 non-null    object 
 5   voice mail plan 3333 non-null    object 
 6   number vmail messages 3333 non-null    int64  
 7   total day minutes 3333 non-null    float64
 8   total day calls  3333 non-null    int64  
 9   total day charge 3333 non-null    float64
 10  total eve minutes 3333 non-null    float64
 11  total eve calls  3333 non-null    int64  
 12  total eve charge 3333 non-null    float64
 13  total night minutes 3333 non-null    float64
 14  total night calls 3333 non-null    int64  
 15  total night charge 3333 non-null    float64
 16  total intl minutes 3333 non-null    float64
 17  total intl calls  3333 non-null    int64  
 18  total intl charge 3333 non-null    float64
 19  customer service calls 3333 non-null    int64  
 20  churn            3333 non-null    bool  
dtypes: bool(1), float64(8), int64(8), object(4)
memory usage: 524.2+ KB
```

```
In [38]: df.describe()
```

Out[38]:

	account length	area code	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	
mean	101.064806	437.182418	8.099010	179.775098	100.435644	30.562307	200.980348	100.114311	17.083540	200.872037	100.107711	9.03	74.000000	408.000000
std	39.822106	42.371290	13.688365	54.467389	20.069084	9.259435	50.713844	19.922625	4.310668	50.573847	19.568609	2.27	1.000000	0.000000
min	1.000000	408.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	23.200000	33.000000	1.04	74.000000	408.000000
25%	74.000000	408.000000	0.000000	143.700000	87.000000	24.430000	166.600000	87.000000	14.160000	167.000000	87.000000	7.52	101.000000	415.000000
50%	101.000000	415.000000	0.000000	179.400000	101.000000	30.500000	201.400000	100.000000	17.120000	201.200000	100.000000	9.05	127.000000	510.000000
75%	127.000000	510.000000	20.000000	216.400000	114.000000	36.790000	235.300000	114.000000	20.000000	235.300000	113.000000	10.59	243.000000	510.000000
max	243.000000	510.000000	51.000000	350.800000	165.000000	59.640000	363.700000	170.000000	30.910000	395.000000	175.000000	17.77	3333.000000	510.000000

```
In [39]: #Check for missing values
df.isnull().sum()
```

```
Out[39]: state          0
account length        0
area code             0
phone number          0
international plan    0
voice mail plan       0
number vmail messages 0
total day minutes     0
total day calls        0
total day charge       0
total eve minutes      0
total eve calls         0
total eve charge        0
total night minutes     0
total night calls        0
total night charge       0
total intl minutes      0
total intl calls         0
total intl charge        0
customer service calls 0
churn                  0
dtype: int64
```

```
In [40]: df.duplicated().sum()
```

```
Out[40]: np.int64(0)
```

```
In [41]: df.nunique()
```

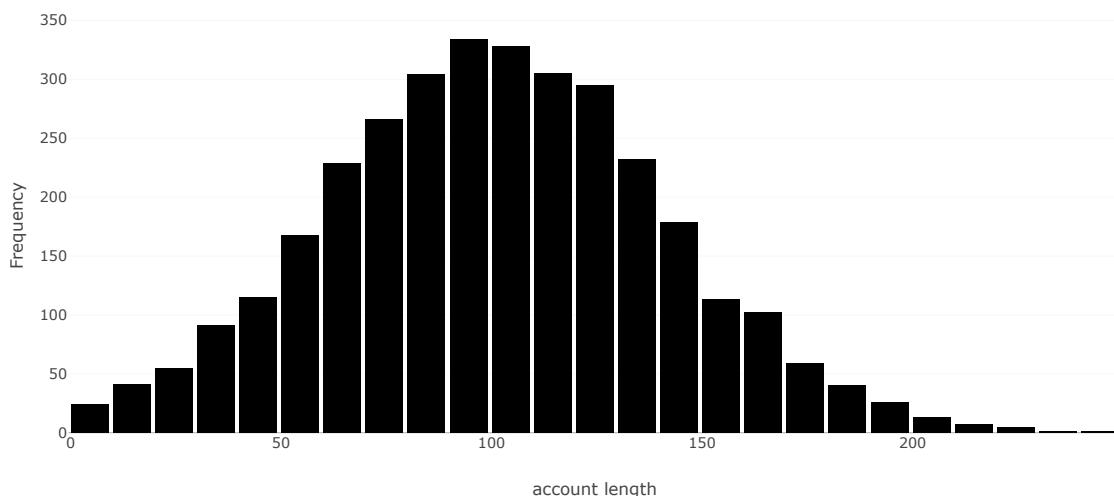
```
Out[41]: state          51
account length        212
area code             3
phone number          3333
international plan    2
voice mail plan       2
number vmail messages 46
total day minutes     1667
total day calls        119
total day charge       1667
total eve minutes      1611
total eve calls         123
total eve charge        1440
total night minutes     1591
total night calls        120
total night charge      933
total intl minutes      162
total intl calls         21
total intl charge        162
customer service calls 10
churn                  2
dtype: int64
```

```
In [42]: # Exploratory Data Analysis
```

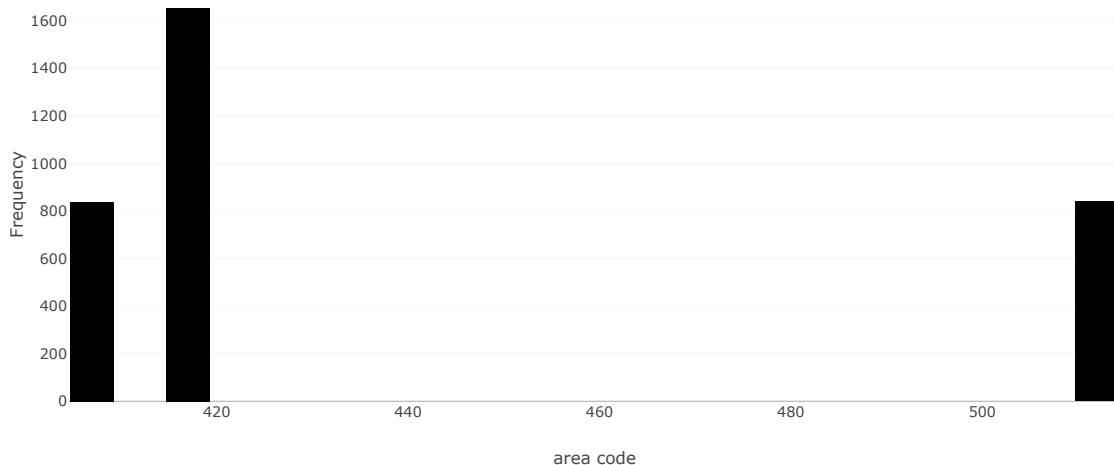
```
#Univariate Analysis with histograms

for col in df.select_dtypes(include=['int64', 'float64']).columns:
    fig = px.histogram(df, x=col, nbins=25, title=f'Histogram of {col}')
    fig.update_layout(xaxis_title=col, yaxis_title='Frequency', bargap=0.1)
    fig.show()
```

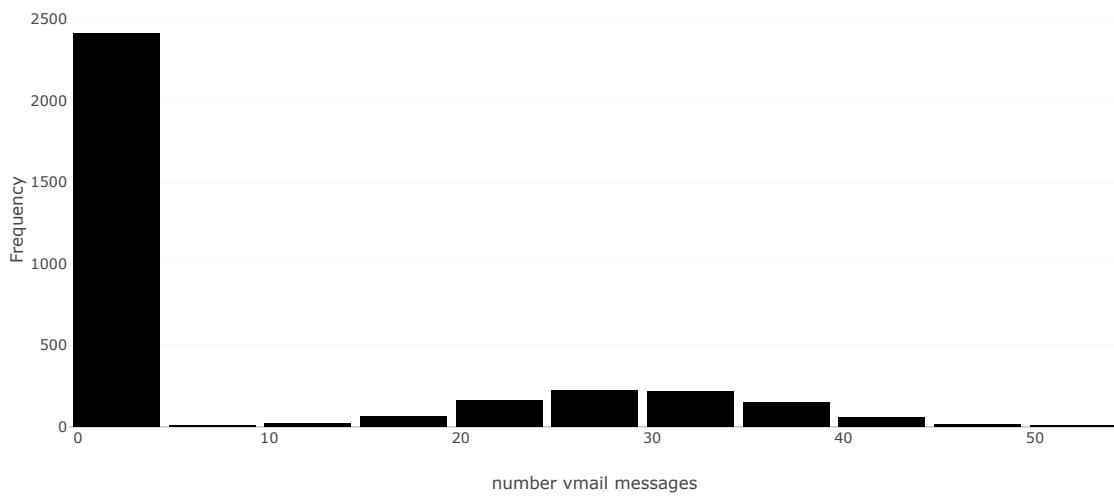
Histogram of account length



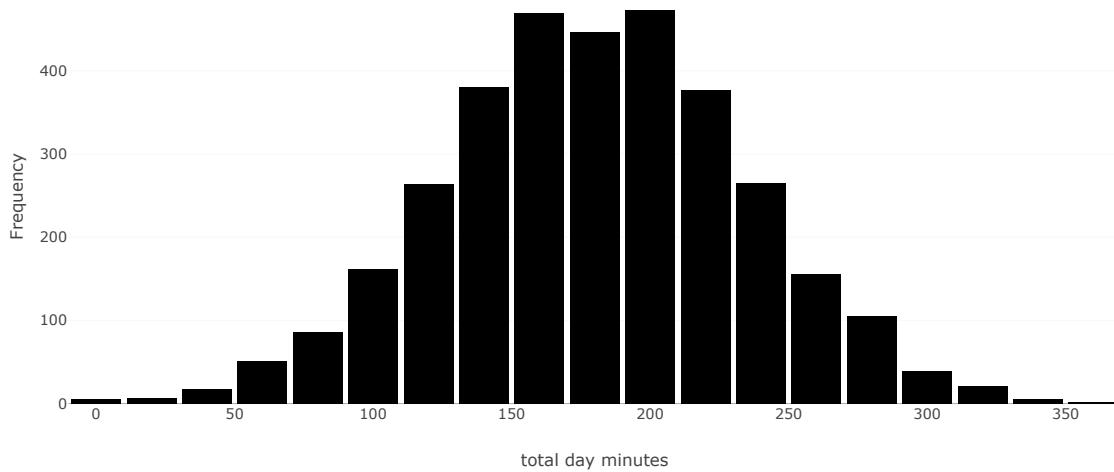
Histogram of area code



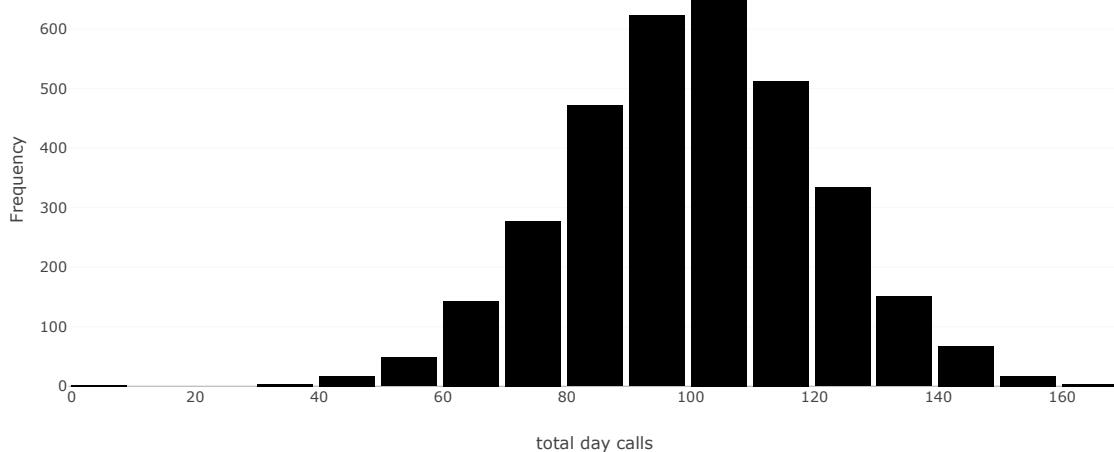
Histogram of number vmail messages



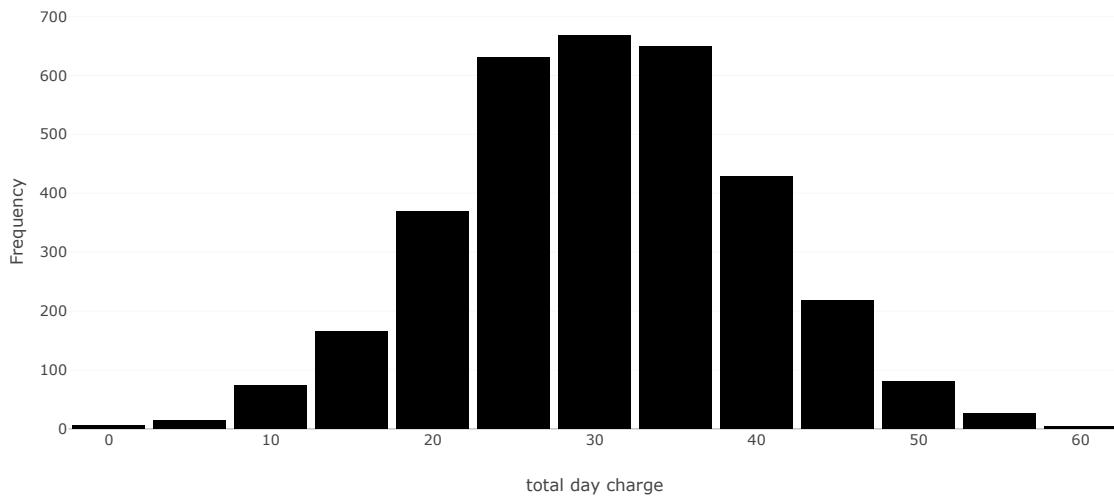
Histogram of total day minutes



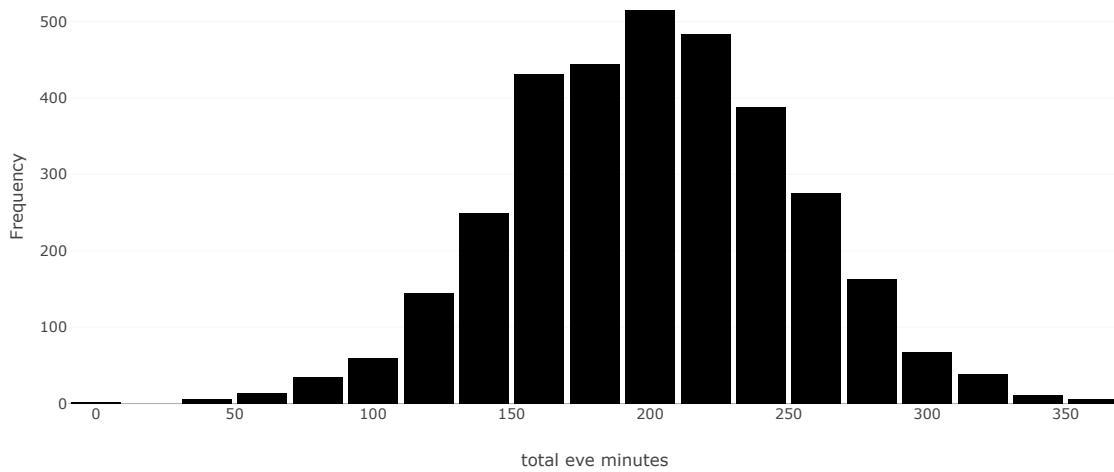
Histogram of total day calls



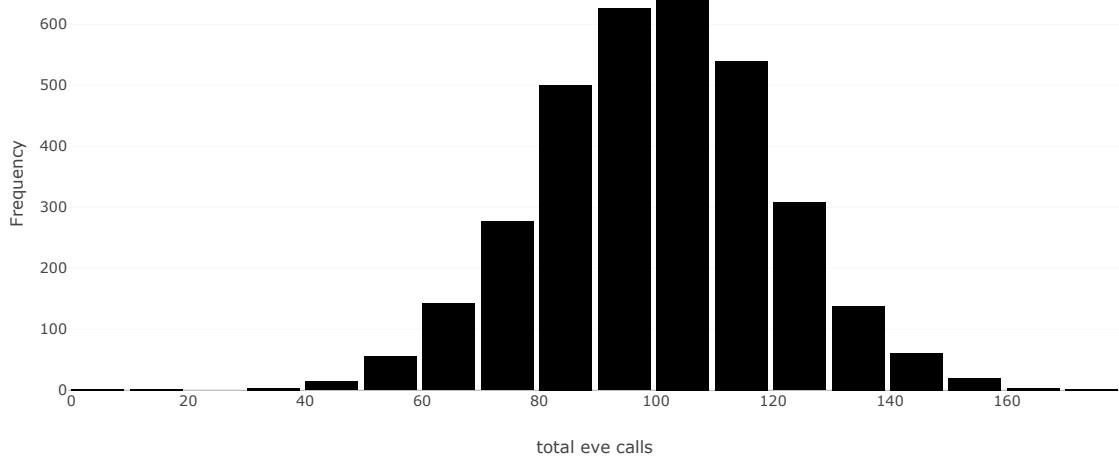
Histogram of total day charge



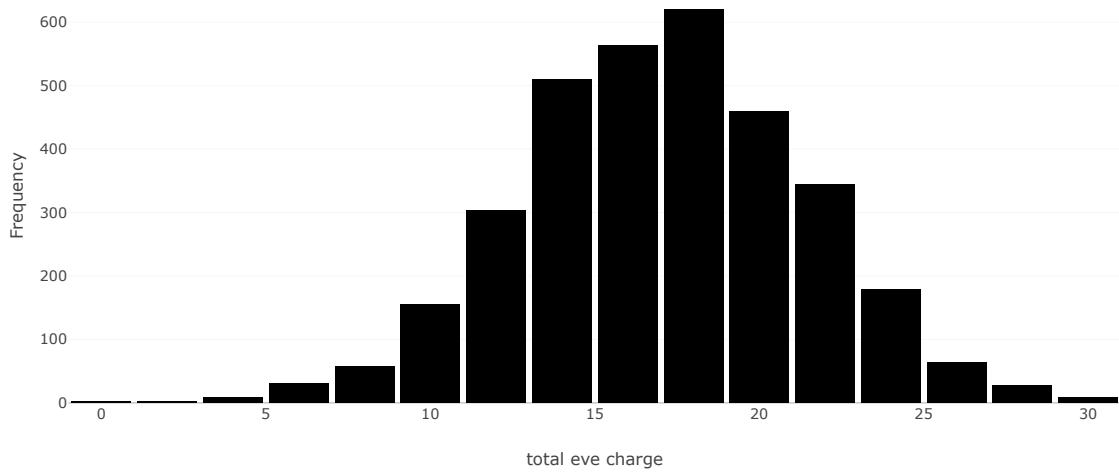
Histogram of total eve minutes



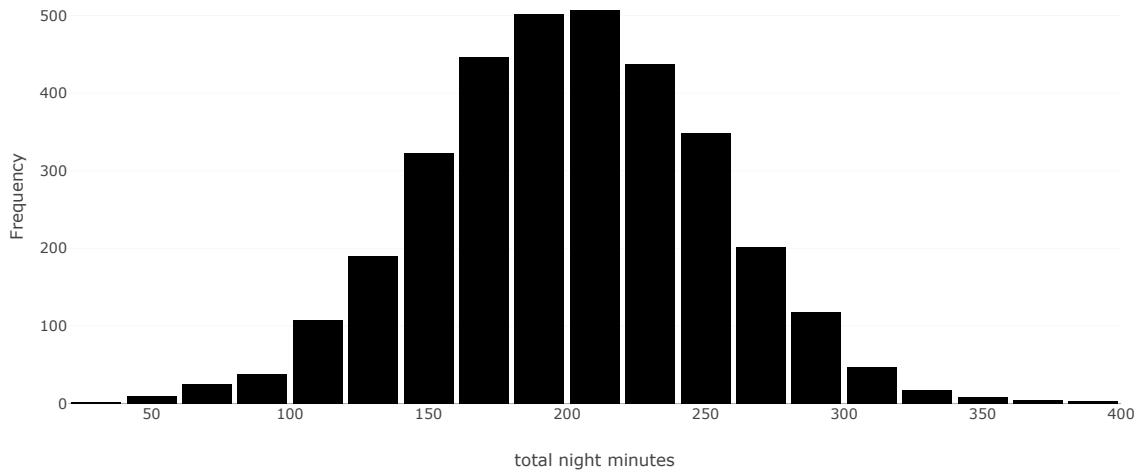
Histogram of total eve calls



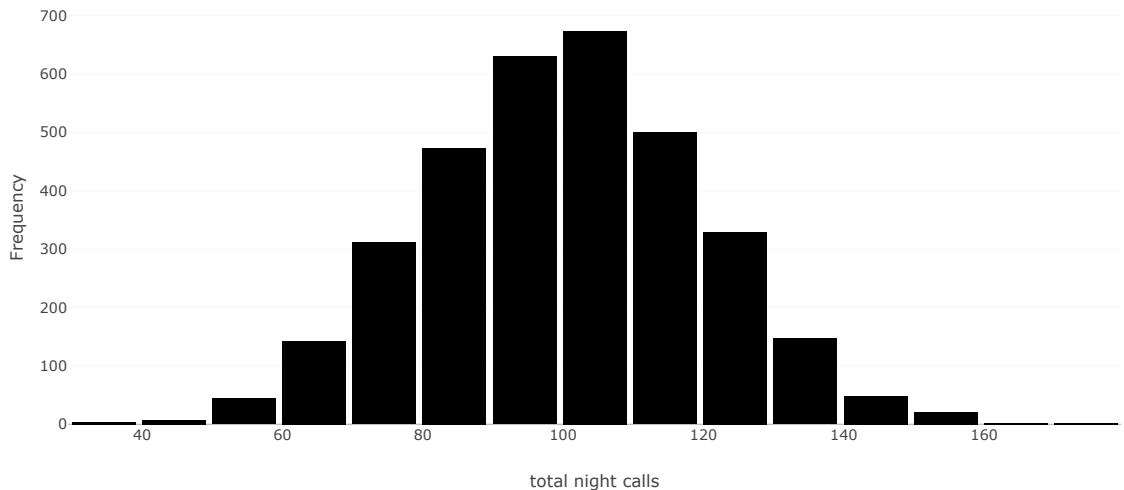
Histogram of total eve charge



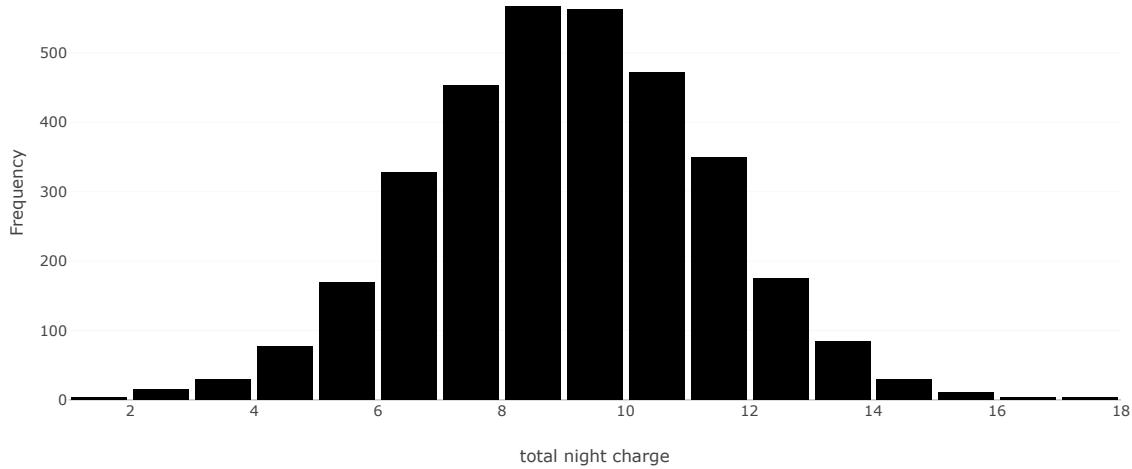
Histogram of total night minutes



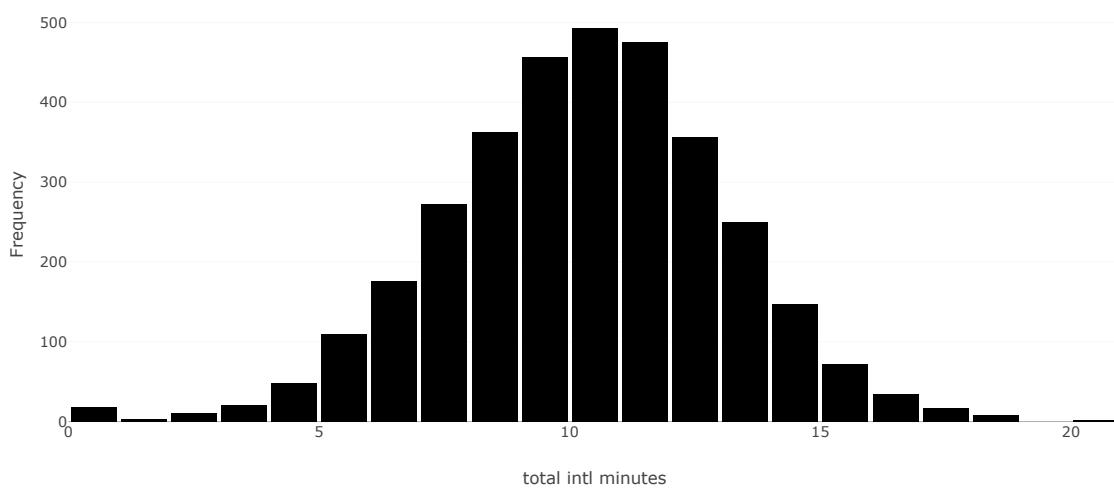
Histogram of total night calls



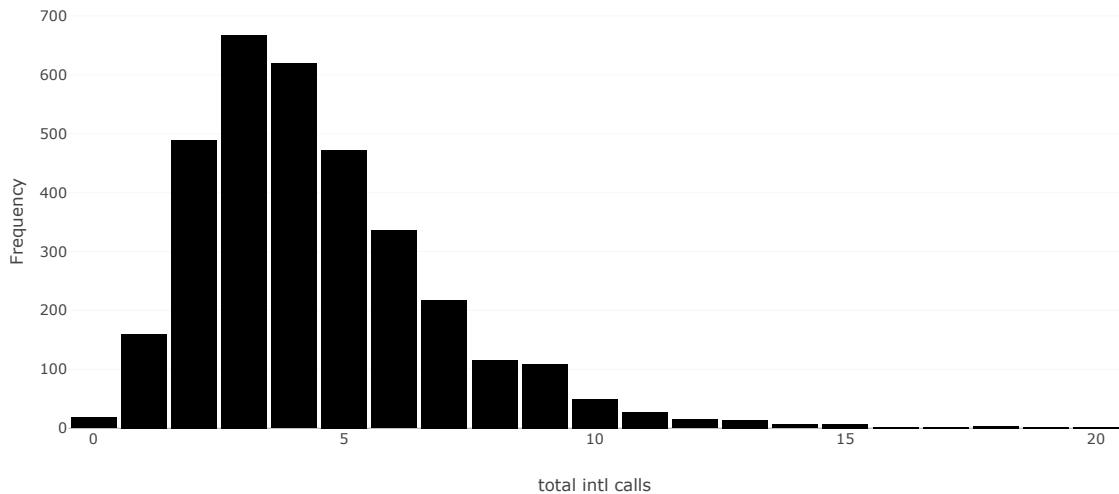
Histogram of total night charge



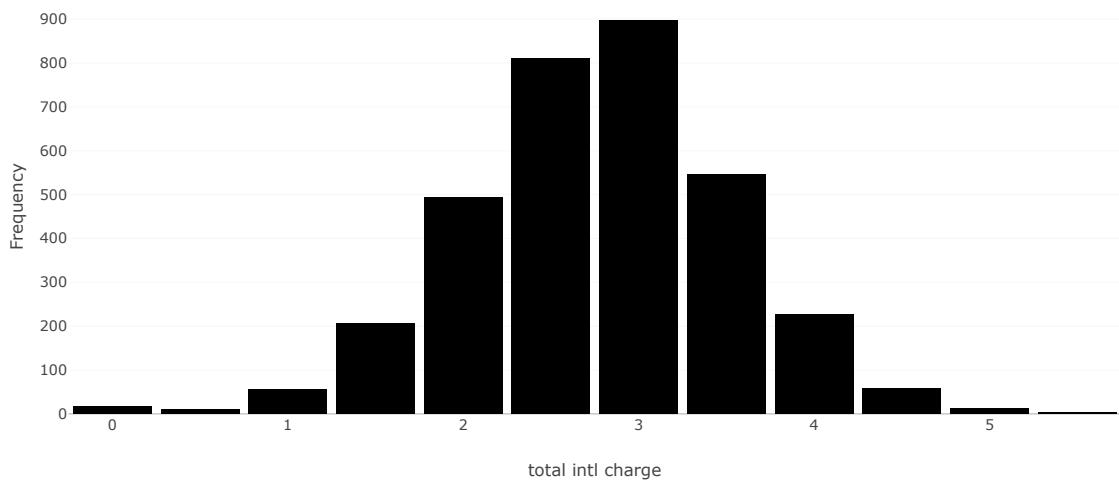
Histogram of total intl minutes



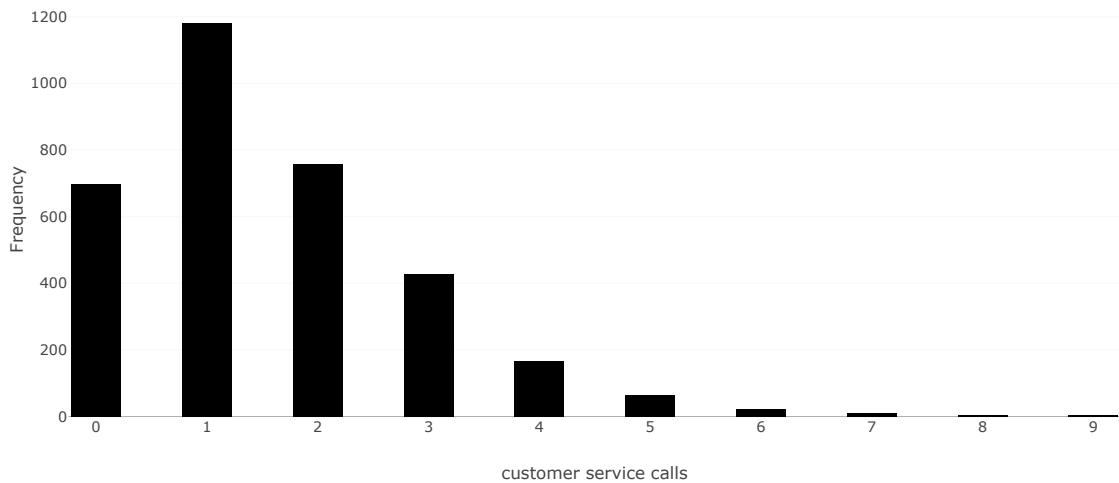
Histogram of total intl calls



Histogram of total intl charge



Histogram of customer service calls



In [43]:
...
Most of the features are normally distributed however there are some features that are skewed.
The features that are right skewed include
1. Total international calls

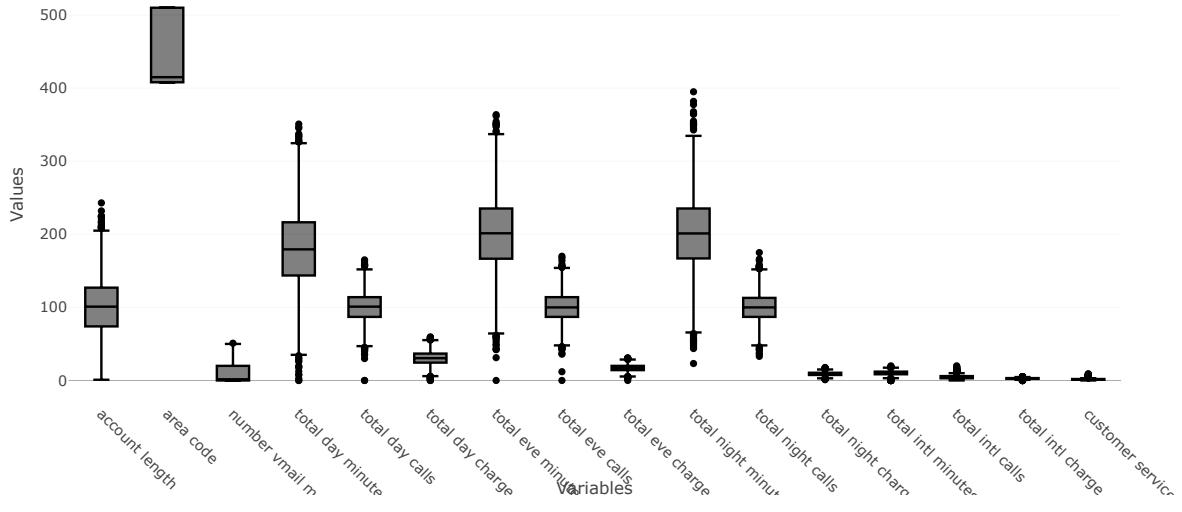
2. Number of customer service calls

...

```
Out[43]: '\nMost of the features are normally distributed however there are some features that are skewed.\nThe features that are right skewed include\n1. Total international calls\n2. Number of customer service calls\n'
```

```
In [44]: # Using boxplots to identify outliers
numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns
df_melted = df.melt(value_vars=numerical_cols, var_name='Variable', value_name='Value')
fig = px.box(df_melted, x='Variable', y='Value', points='outliers', title='Boxplots for all Variables')
fig.update_layout(
    xaxis_title='Variables',
    yaxis_title='Values',
    xaxis={'tickangle': 45},
    boxmode='group'
)
fig.show()
```

Boxplots for all Variables



```
In [45]: # Getting counts of outliers
numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns
outlier_counts = {}

for col in numerical_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

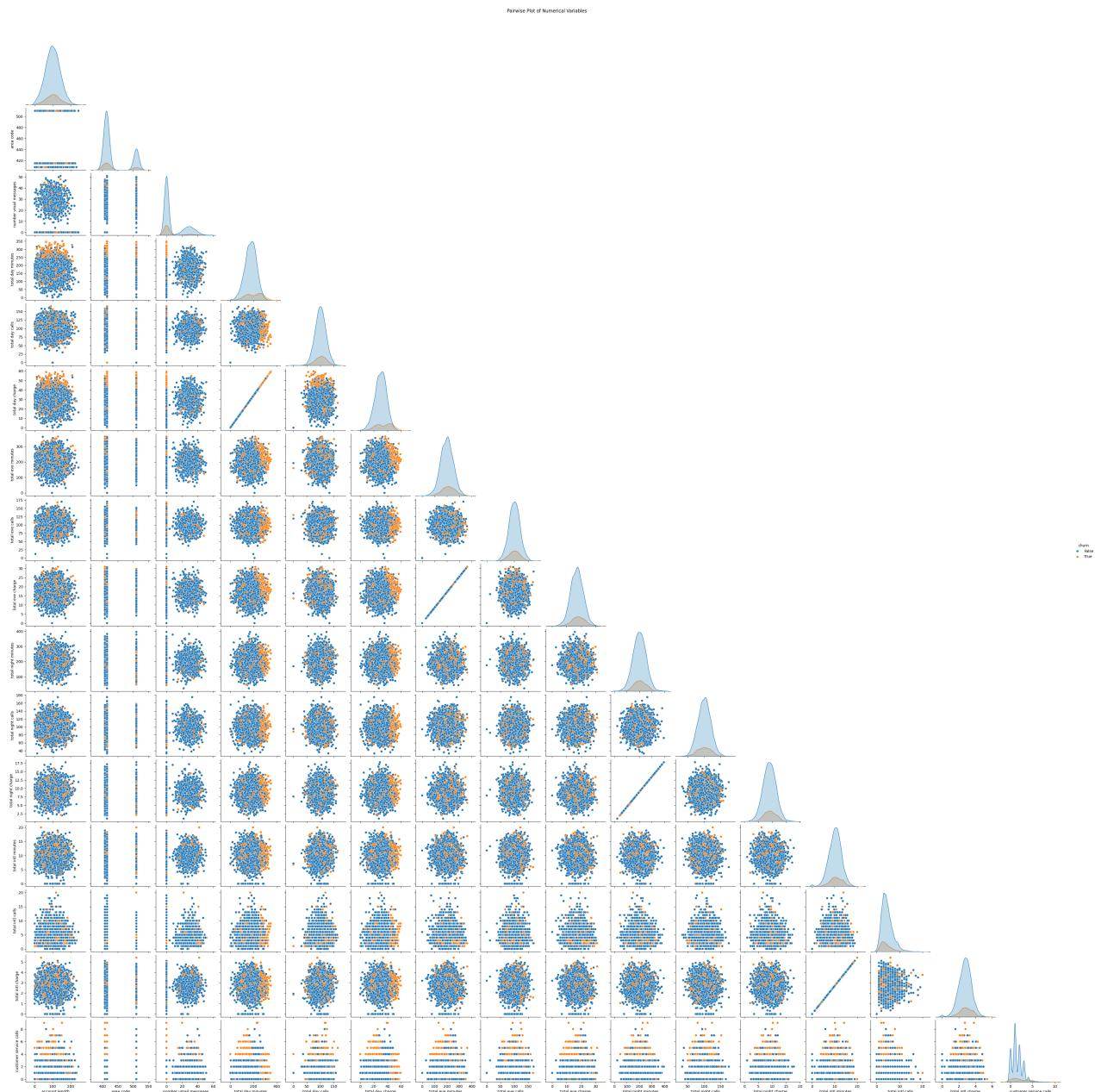
    outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
    outlier_counts[col] = len(outliers)

for col, count in outlier_counts.items():
    print(f'Column: {col}, Number of Outliers: {count}')
```

```
Column: account length, Number of Outliers: 18
Column: area code, Number of Outliers: 0
Column: number vmail messages, Number of Outliers: 1
Column: total day minutes, Number of Outliers: 25
Column: total day calls, Number of Outliers: 23
Column: total day charge, Number of Outliers: 25
Column: total eve minutes, Number of Outliers: 24
Column: total eve calls, Number of Outliers: 20
Column: total eve charge, Number of Outliers: 24
Column: total night minutes, Number of Outliers: 30
Column: total night calls, Number of Outliers: 22
Column: total night charge, Number of Outliers: 30
Column: total intl minutes, Number of Outliers: 46
Column: total intl calls, Number of Outliers: 78
Column: total intl charge, Number of Outliers: 49
Column: customer service calls, Number of Outliers: 267
```

```
In [46]: # There are quite a few outliers in the variables however, we will deal with them depending on their correlation so we can accordingly transform
```

```
In [47]: # Bivariate Analysis and Correlation Matrix
imp_cols = df[numerical_cols].assign(churn=df['churn'])
%matplotlib inline
sns.pairplot(imp_cols, kind='kde', corner=True, hue='churn')
plt.suptitle('Pairwise Plot of Numerical Variables', y=1.02)
plt.show()
```



In [48]:
 ...
 1. We can see that some variables are correlated with each other such as total day charge and total day minutes, total eve charge and total eve minutes.
 2. Some variables are not normal distributed and have multiple peaks possibly suggesting grouping such as number of voicemail messages and customer service calls

Out[48]:
 '\n1. We can see that some variables are correlated with each other such as total day charge and total day minutes, total eve charge and total eve minutes, total night charge and total night minutes, total international charge and total international minutes.\n2. Some variables are not normal distributed and have multiple peaks possibly suggesting grouping such as number of voicemail messages and customer service calls\n'

In [49]:

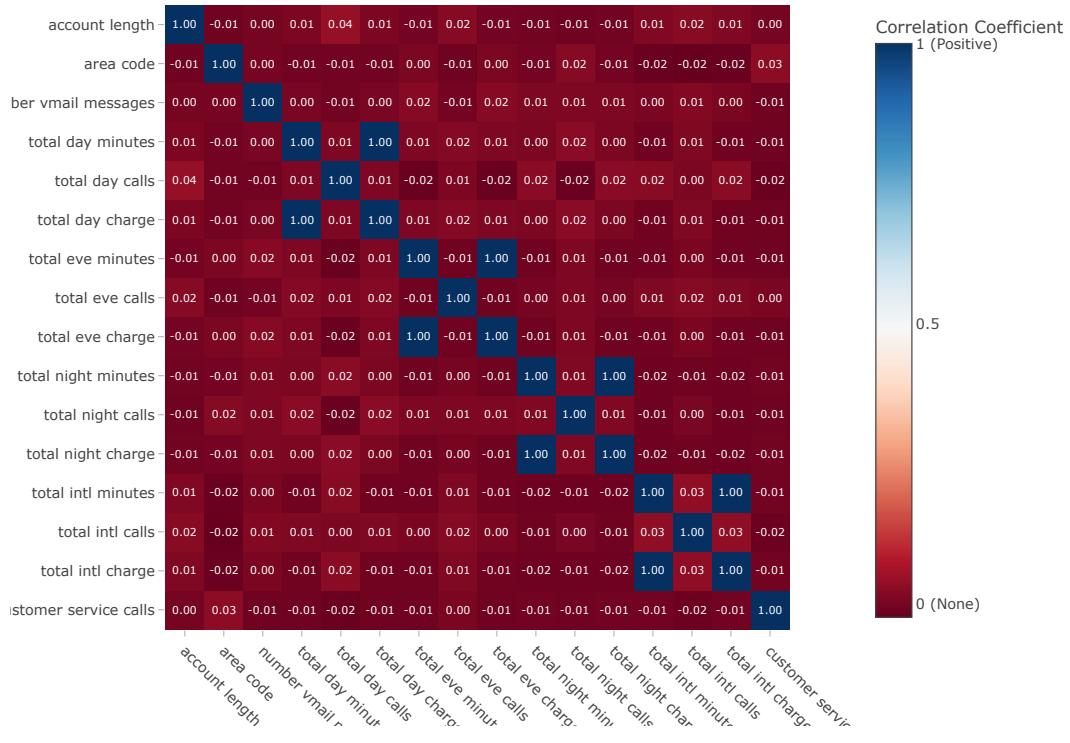
```
correlation_matrix = df[numerical_cols].corr()

fig = px.imshow(
    correlation_matrix,
    text_auto=".2f", # Add annotations with 2 decimal places
    color_continuous_scale="RdBu", # Red-Blue color scale for contrast
    title="Correlation Heatmap",
    labels=dict(color="Correlation"), # Add a color bar Label
)

fig.update_layout(
    title_font_size=18,
    title_x=0.5, # Center the title
    xaxis=dict(tickangle=45), # Rotate x-axis Labels for readability
    coloraxis_colorbar=dict(
        title="Correlation Coefficient", # Color bar title
        tickvals=[-1, -0.5, 0, 0.5, 1], # Define ticks for the color bar
        ticktext=["-1 (Negative)", "-0.5", "0 (None)", "0.5", "1 (Positive)"], # Label ticks
    ),
    width=900, # Adjust the width of the heatmap
    height=700, # Adjust the height
)

fig.show()
```

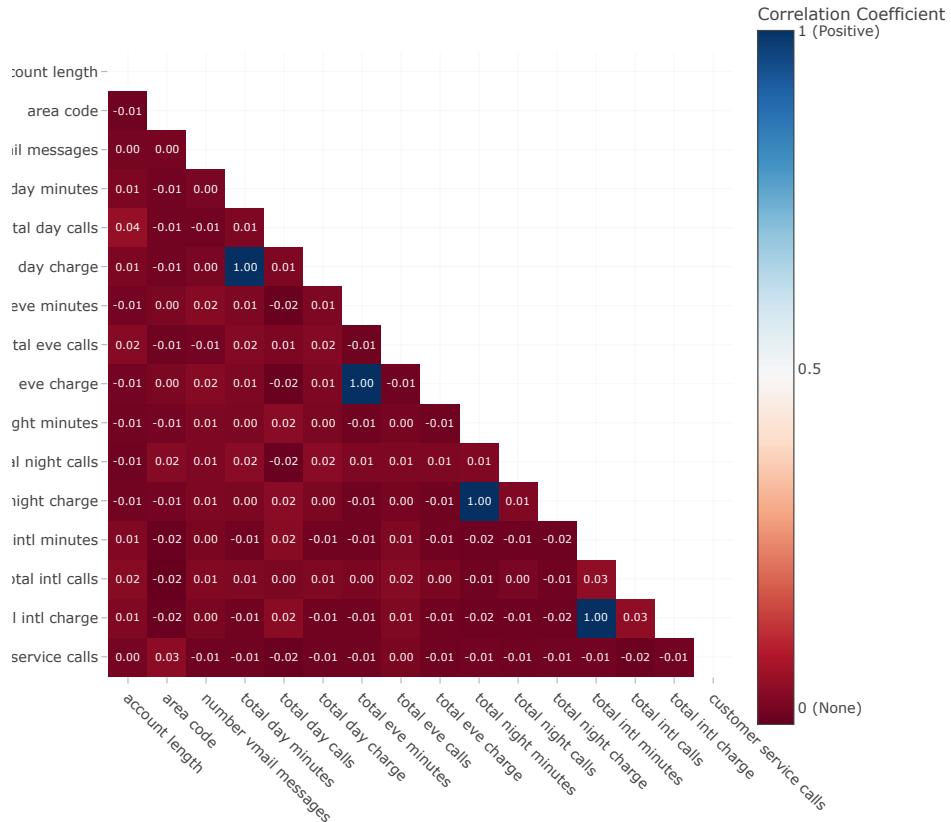
Correlation Heatmap



```
In [50]: # For better readability, we will only show the Lower triangle of the correlation matrix
# Step 1: Mask the upper triangle
mask = np.triu(np.ones_like(correlation_matrix, dtype=bool)) # Upper triangle mask
correlation_matrix_masked = correlation_matrix.mask(mask) # Apply the mask

# Step 2: Plot the heatmap
fig = px.imshow(
    correlation_matrix_masked,
    text_auto=".2f", # Show values with 2 decimal places
    color_continuous_scale="RdBu",
    title="Correlation Heatmap (Lower Triangle Only)",
    labels=dict(color="Correlation"),
)
fig.update_layout(
    title_font_size=18,
    title_x=0.5, # Center the title
    xaxis=dict(tickangle=45), # Rotate x-axis Labels
    coloraxis_colorbar=dict(
        title="Correlation Coefficient",
        tickvals=[-1, -0.5, 0, 0.5, 1],
        ticktext=["-1 (Negative)", "-0.5", "0 (None)", "0.5", "1 (Positive)"],
    ),
    width=800,
    height=800,
)
fig.show()
```

Correlation Heatmap (Lower Triangle Only)



```
In [51]: # The relationship between variables observed in the correlation matrix is consistent with the pairplot.
# However, upon looking at the distributions with churn as a feature, we can see that the dataset is imbalanced. To confirm this we look at the distribution of churn.
```

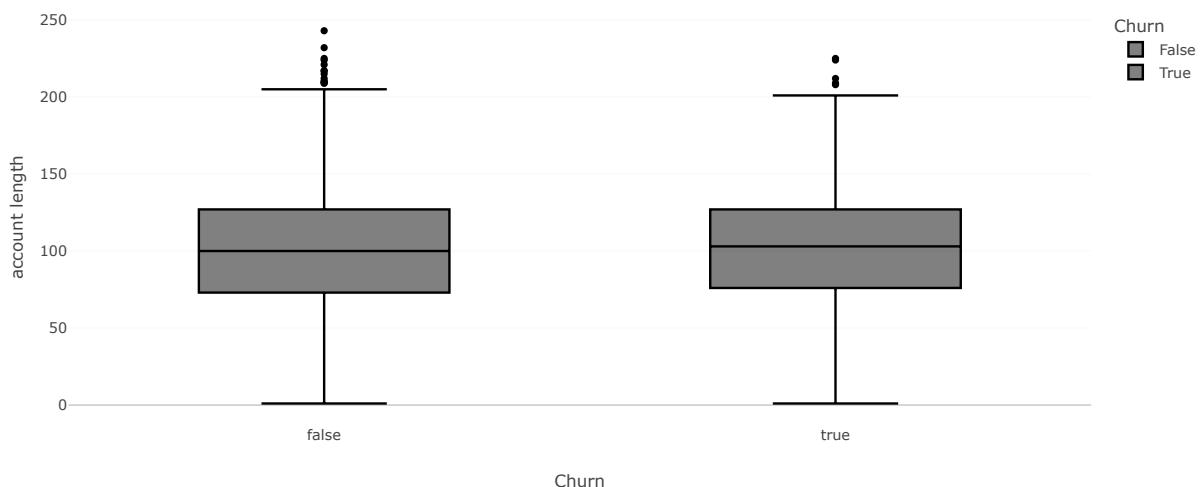
```
In [52]: churn_distribution = df['churn'].value_counts(normalize=True)
print(churn_distribution)

churn
False    0.855086
True     0.144914
Name: proportion, dtype: float64
```

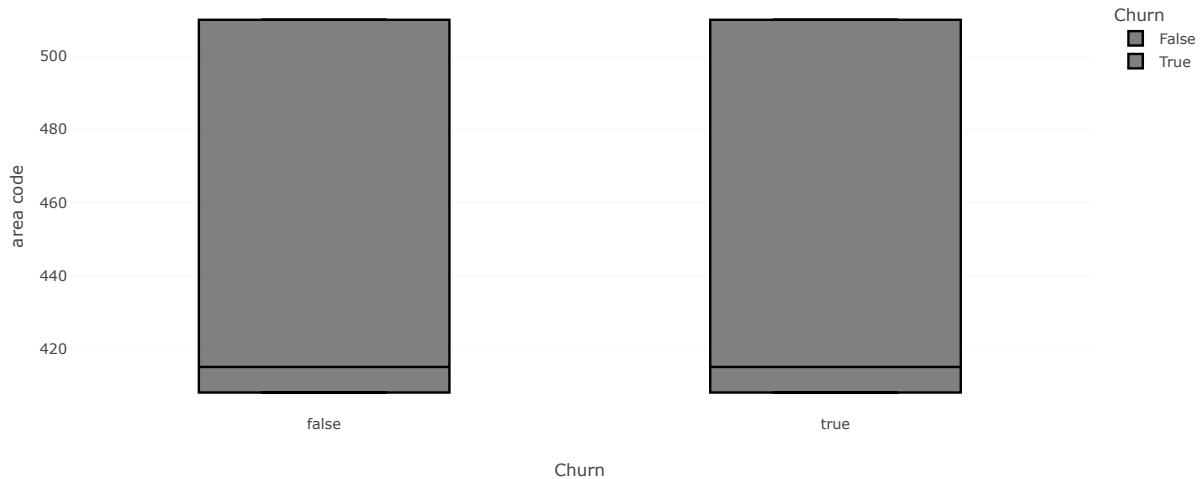
```
In [53]: # 86% of customers did not churn while 14% of customers churned.
```

```
In [54]: # Using boxplots to compare numerical features by churn
for col in numerical_cols:
    fig = px.box(df, x='churn', y=col, color='churn',
                 title=f'Distribution of {col} by Churn',
                 labels={'churn': 'Churn', 'col': col})
    fig.update_traces(quartilemethod="exclusive")
    fig.show()
```

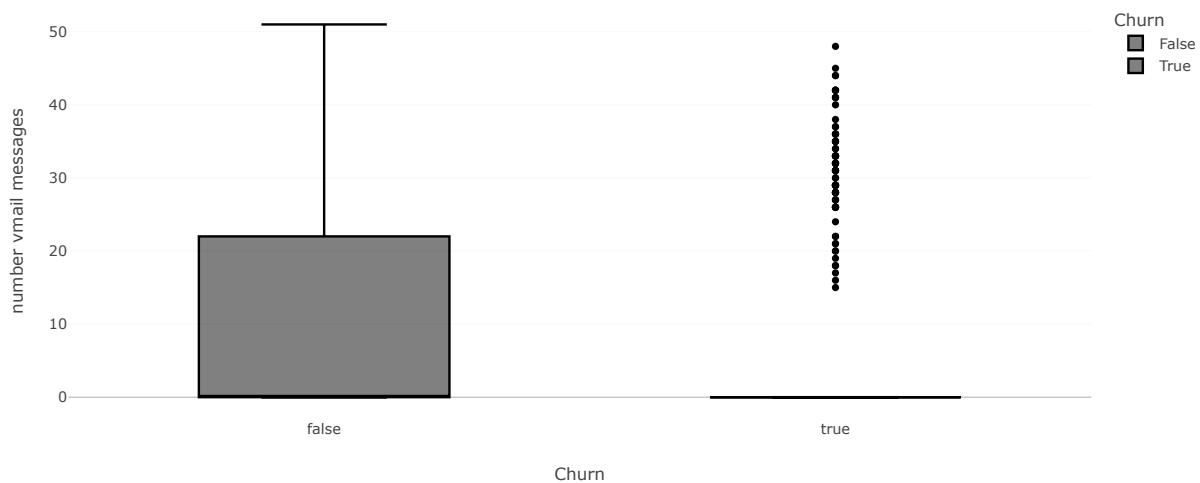
Distribution of account length by Churn



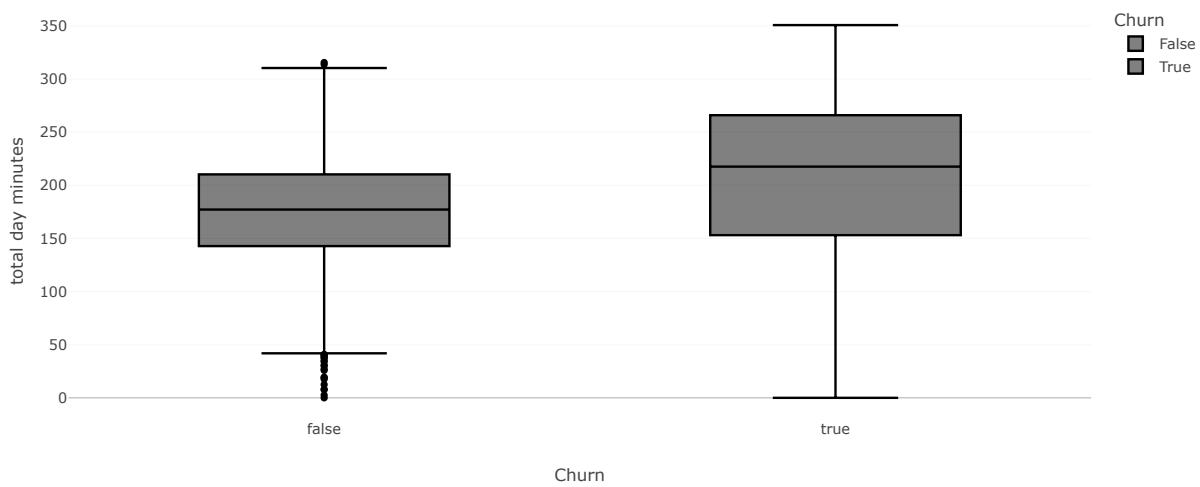
Distribution of area code by Churn



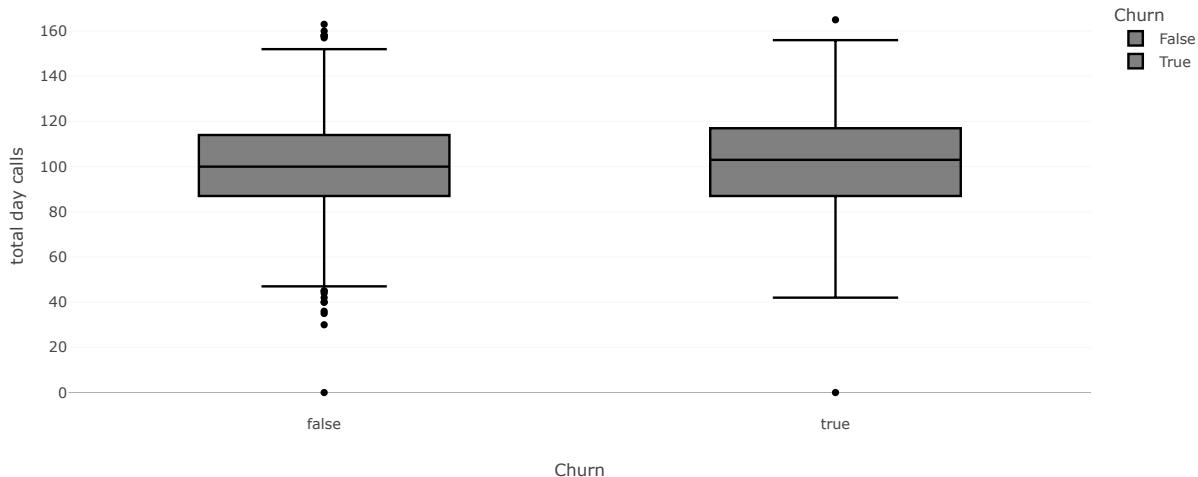
Distribution of number vmail messages by Churn



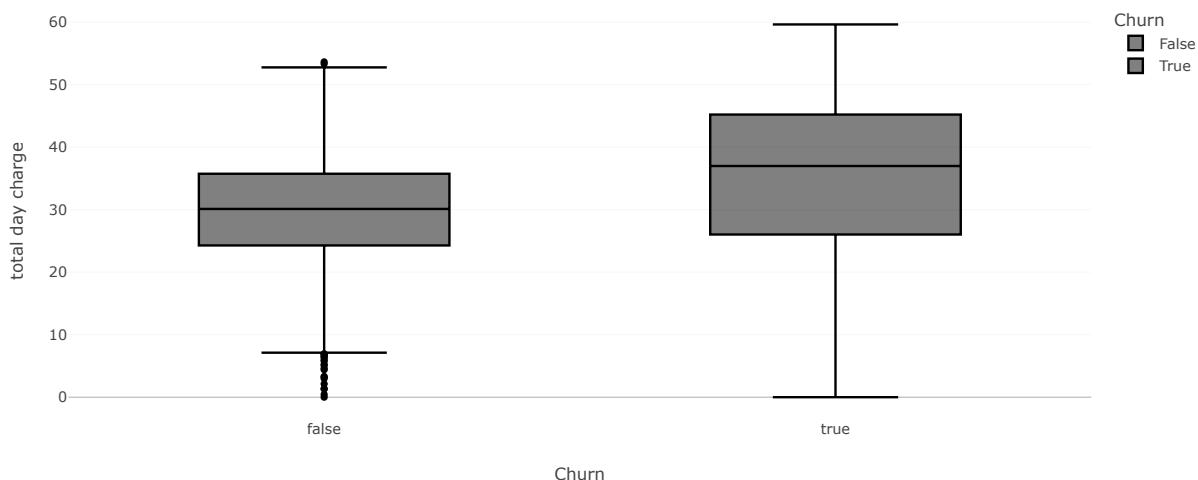
Distribution of total day minutes by Churn



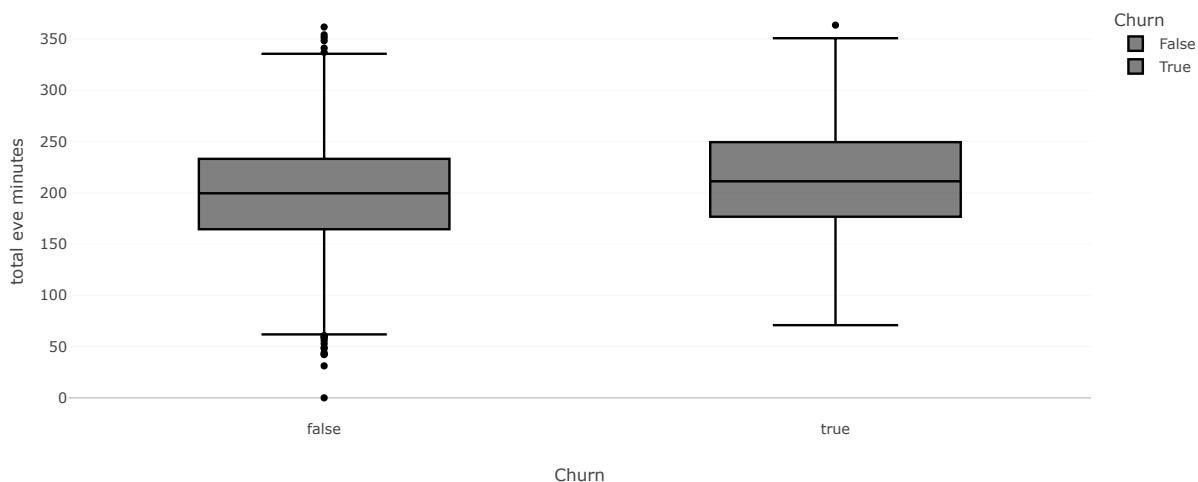
Distribution of total day calls by Churn



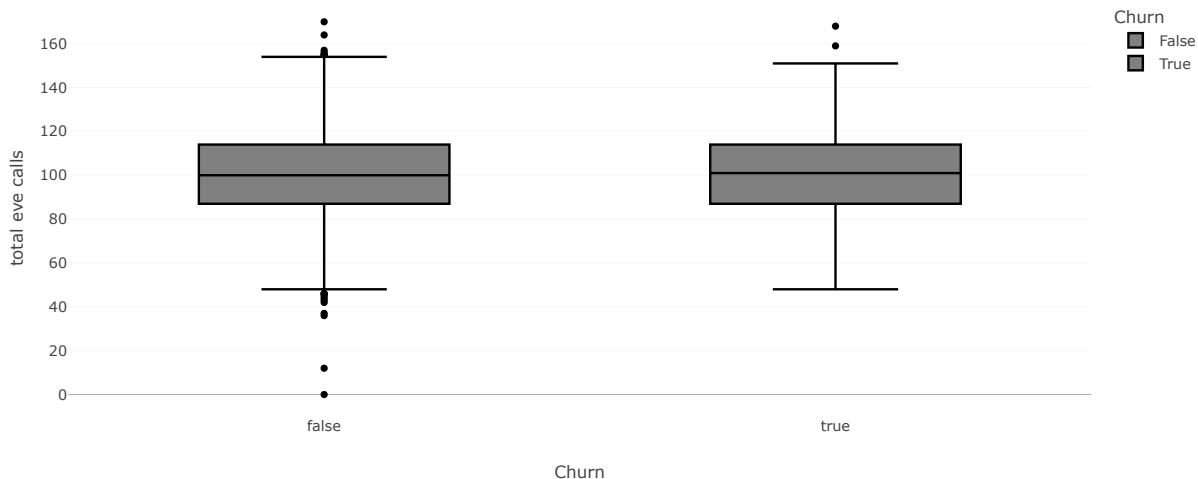
Distribution of total day charge by Churn



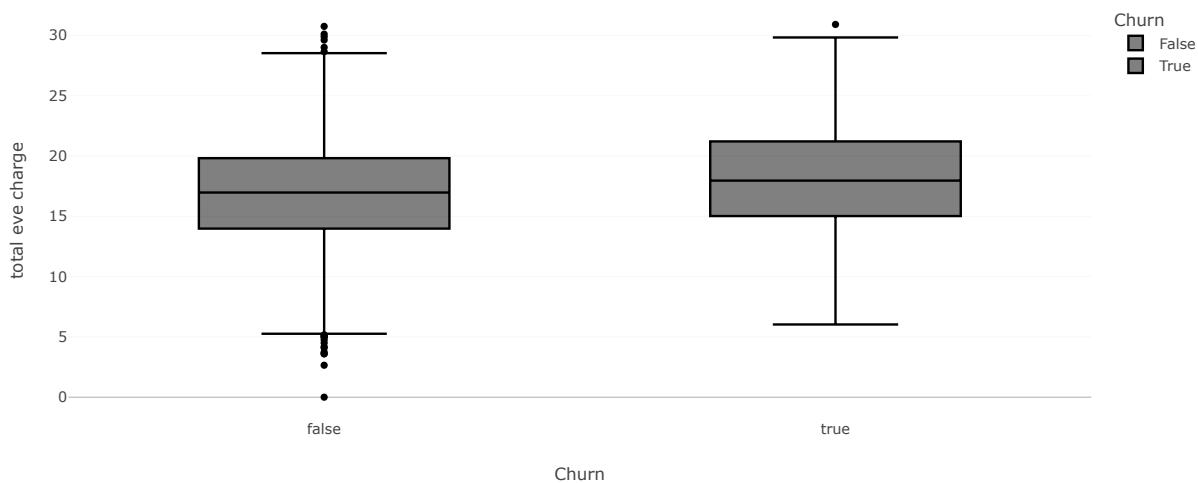
Distribution of total eve minutes by Churn



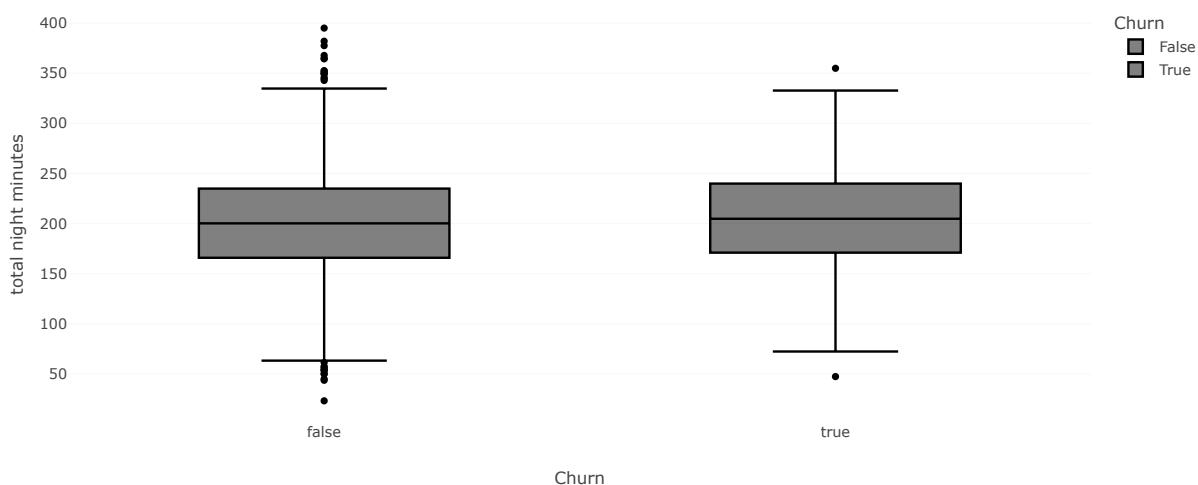
Distribution of total eve calls by Churn



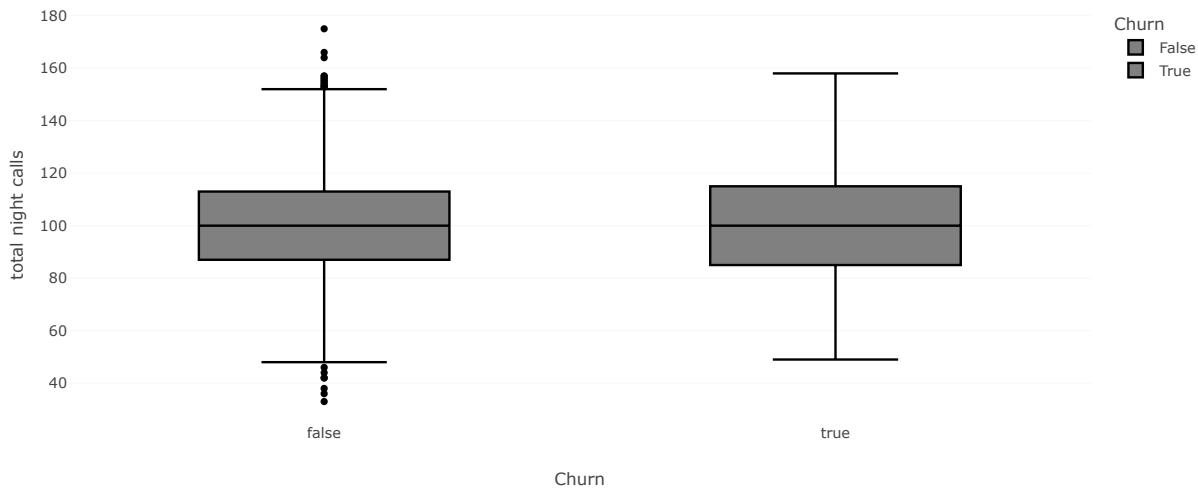
Distribution of total eve charge by Churn



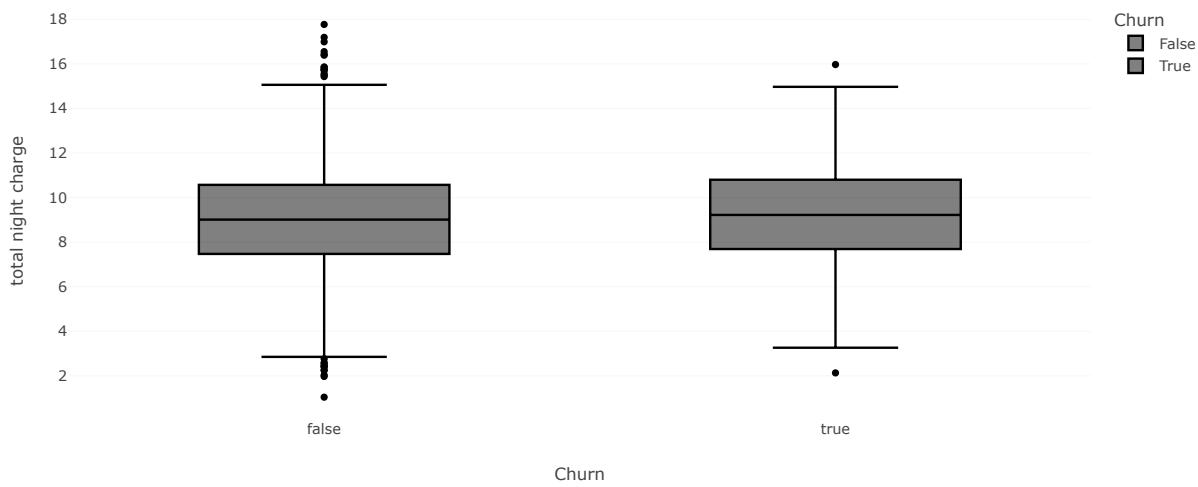
Distribution of total night minutes by Churn



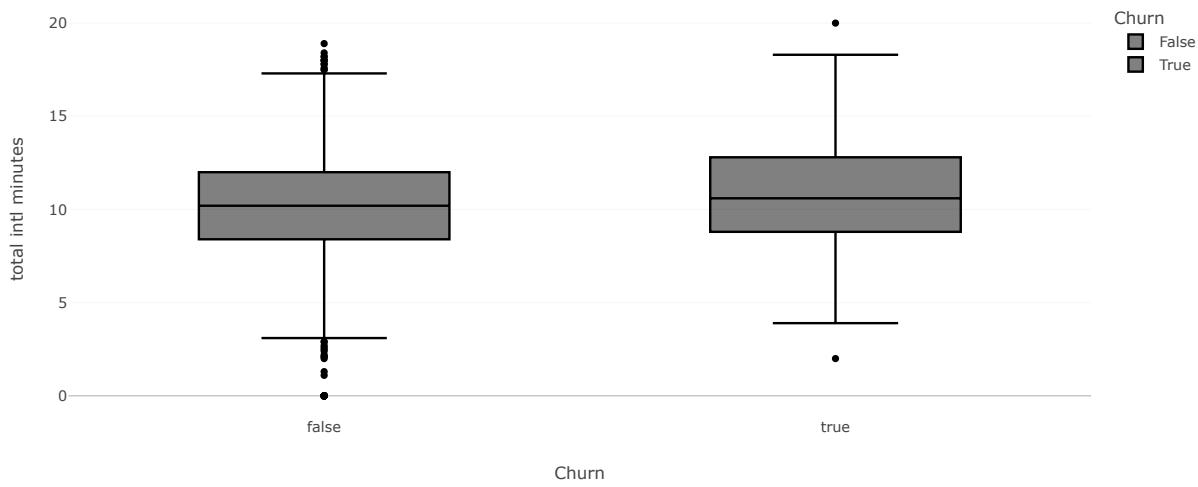
Distribution of total night calls by Churn



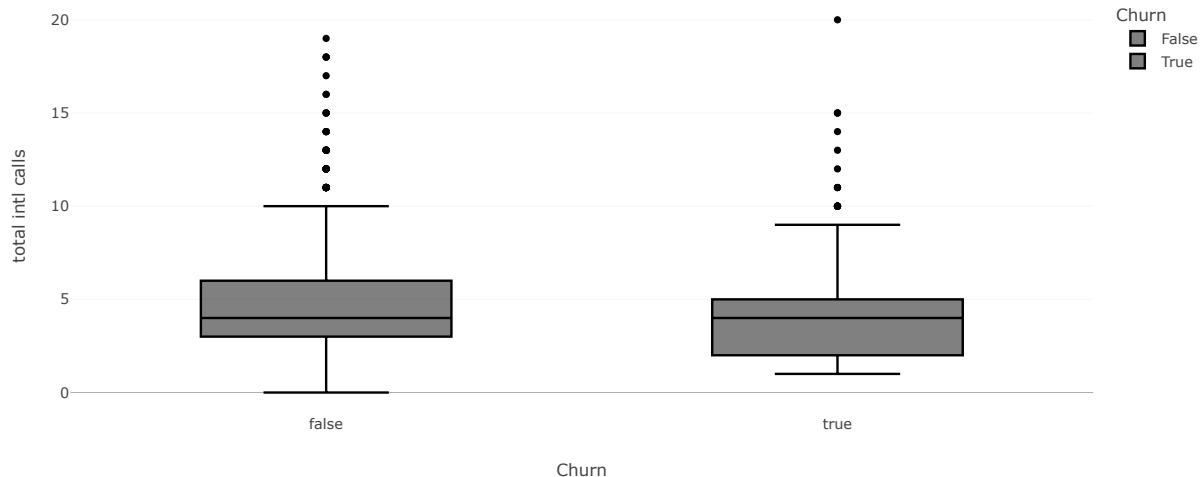
Distribution of total night charge by Churn



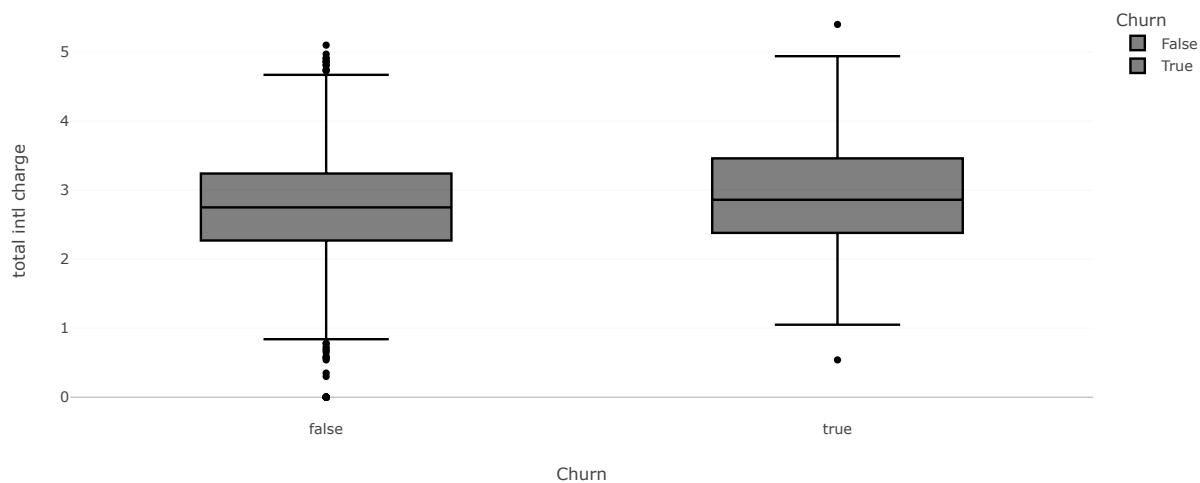
Distribution of total intl minutes by Churn



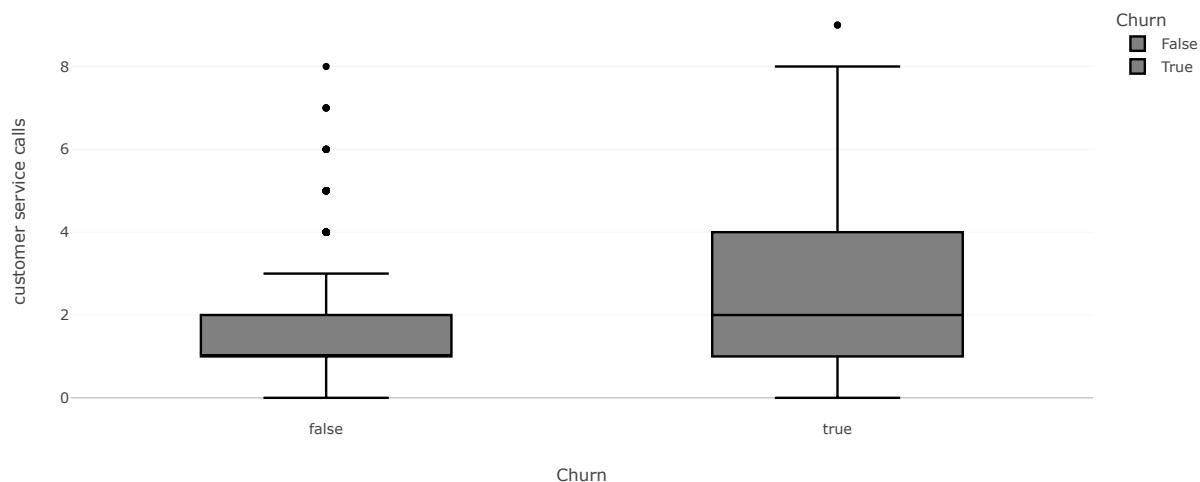
Distribution of total intl calls by Churn



Distribution of total intl charge by Churn



Distribution of customer service calls by Churn



```
In [55]: ...
1. We can observe that the distribution of customer service calls is more dispersed for customers who churned indicating that had more customer s
2. Customers who churned spent more minutes talking during the day
3. The distribution of number of voicemail messages for customers who churned is essentially flat indicating that most churned customers have zero
...
```

Out[55]: "\n1. We can observe that the distribution of customer service calls is more dispersed for customers who churned indicating that had more customer service calls that varied in experience.\n2. Customers who churned spent more minutes talking during the day\n3. The distribution of number of voicemail messages for customers who churned is essentially flat indicating that most churned customers have zero voicemail messages. It appears that those who don't use voicemail services would be likely to churn.\n"

In [56]: # Running statistical tests on numerical variables

```
from scipy.stats import ttest_ind

results = {}
for col in numerical_cols:
    churned = df[df['churn'] == True][col]
    not_churned = df[df['churn'] == False][col]

    # Perform t-test
    t_stat, p_value = ttest_ind(churned, not_churned, equal_var=False)
    results[col] = {'T-statistic': t_stat, 'P-value': p_value}

# Print results
for col, stats in results.items():
    print(f"{col}:\n\tT-statistic: {stats['T-statistic']:.2f}, P-value: {stats['P-value']:.3f}\n")

account length:
    T-statistic: 0.96, P-value: 0.336

area code:
    T-statistic: 0.35, P-value: 0.724

number vmail messages:
    T-statistic: -5.82, P-value: 0.000

total day minutes:
    T-statistic: 9.68, P-value: 0.000

total day calls:
    T-statistic: 1.00, P-value: 0.317

total day charge:
    T-statistic: 9.68, P-value: 0.000

total eve minutes:
    T-statistic: 5.27, P-value: 0.000

total eve calls:
    T-statistic: 0.54, P-value: 0.591

total eve charge:
    T-statistic: 5.27, P-value: 0.000

total night minutes:
    T-statistic: 2.17, P-value: 0.030

total night calls:
    T-statistic: 0.35, P-value: 0.727

total night charge:
    T-statistic: 2.17, P-value: 0.030

total intl minutes:
    T-statistic: 3.94, P-value: 0.000

total intl calls:
    T-statistic: -2.96, P-value: 0.003

total intl charge:
    T-statistic: 3.94, P-value: 0.000

customer service calls:
    T-statistic: 8.96, P-value: 0.000
```

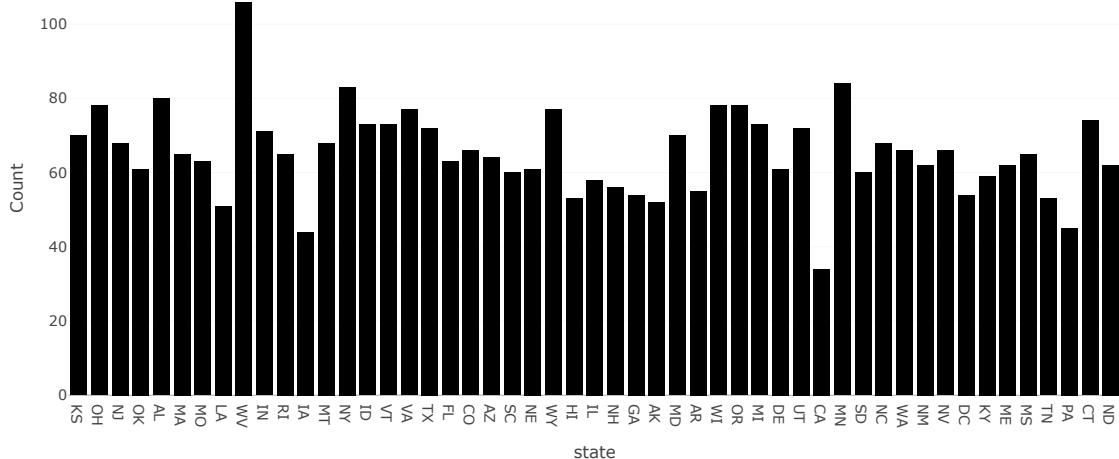
In []:

In [57]: # We move onto analyse the categorical variables.

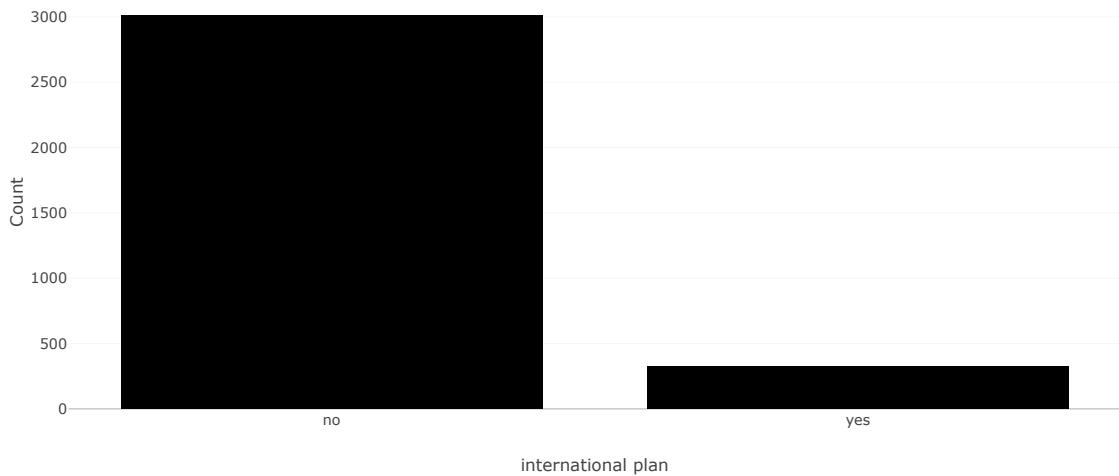
```
categorical_cols = ['state', 'international plan', 'voice mail plan']

for col in categorical_cols:
    fig = px.histogram(df, x=col, title=f'Distribution of {col}', color_discrete_sequence=['black'])
    fig.update_layout(xaxis_title=col, yaxis_title="Count")
    fig.show()
```

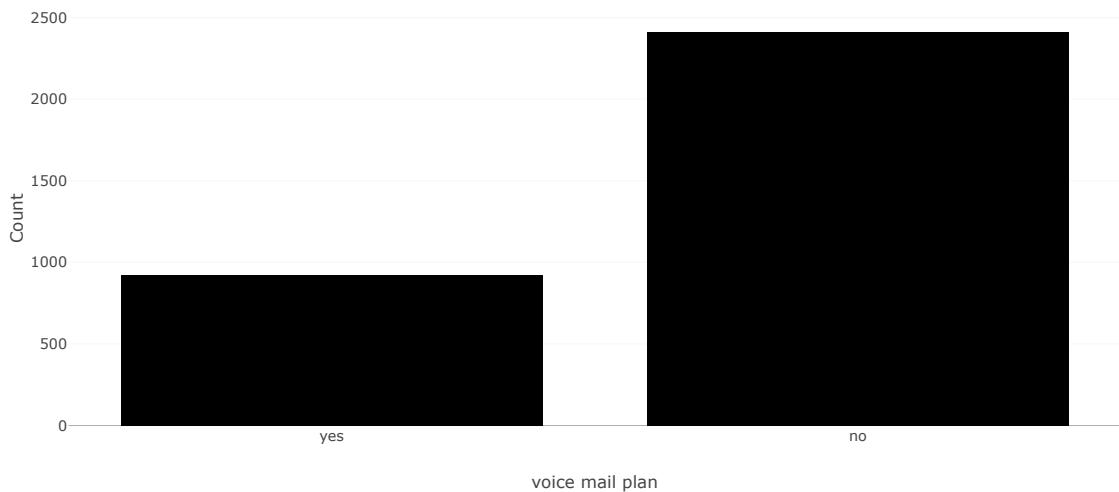
Distribution of state



Distribution of international plan



Distribution of voice mail plan



```
In [59]: # Majority of the customers did not have an international plan and did not have a voice mail plan.
#Further drilling down using the churned variable.
```

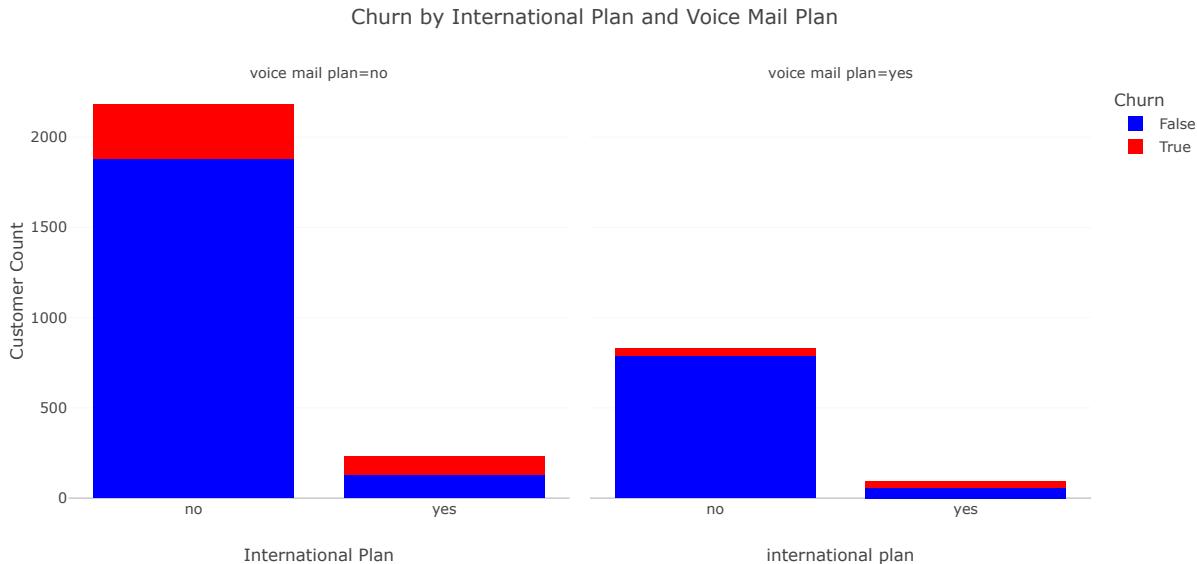
```
In [60]: # Crosstab to analyze churn based on both international plan and voice mail plan
crosstab = pd.crosstab([df['international plan'], df['voice mail plan']], df['churn'])
```

```
print(crosstab)

churn          False  True
international plan voice mail plan
no            no      1878   302
                  yes      786    44
yes           no      130    101
                  yes      56     36

In [61]: # Melt crosstab for visualization
crosstab_reset = crosstab.reset_index()
crosstab_melted = crosstab_reset.melt(id_vars=['international plan', 'voice mail plan'],
                                         var_name='churn', value_name='count')

# Plot
fig = px.bar(
    crosstab_melted,
    x='international plan',
    y='count',
    color='churn',
    facet_col='voice mail plan',
    barmode='stack',
    title='Churn by International Plan and Voice Mail Plan',
    labels={'count': 'Customer Count', 'churn': 'Churn'},
    color_discrete_map={True: 'red', False: 'blue'} # Explicitly map colors
)
fig.update_layout(
    showlegend=True,
    xaxis_title="International Plan",
    yaxis_title="Customer Count"
)
fig.show()
```



```
In [62]: from scipy.stats import chi2_contingency

# Create a contingency table
contingency_table = pd.crosstab([df['international plan'], df['voice mail plan']], df['churn'])
print(contingency_table)
# Perform chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_table)
print(f"Chi-square Statistic: {chi2:.2f}, p-value: {p:.3f}")

churn          False  True
international plan voice mail plan
no            no      1878   302
                  yes      786    44
yes           no      130    101
                  yes      56     36

Chi-square Statistic: 261.65, p-value: 0.000
```

In []: