Exploratory Data Analysis

- This EDA is performed on a telecom company's dataset provided by Sony Research as part of their data science positions' recruitment process
- More information on this task can be found on https://platform.stratascratch.com/data-projects/customer-churn-prediction

In [2]:
```
# 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 [3]:
```
# Importing necessary packages
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 [4]:
```
#Load the data
df = pd.read_csv("../data/processed/Data_Science_Challenge.csv")
```

In [5]:
```
pd.set_option('display.max_columns', None)
df.head()
```

Out[5]:

	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 cha
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 [6]:
```
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 [7]:
```
df.describe()
```

Out[7]:

	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
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.00
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
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
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
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
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
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
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

In [8]:
```
#Check for missing values
df.isnull().sum()
```

Out[8]:
```
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
```

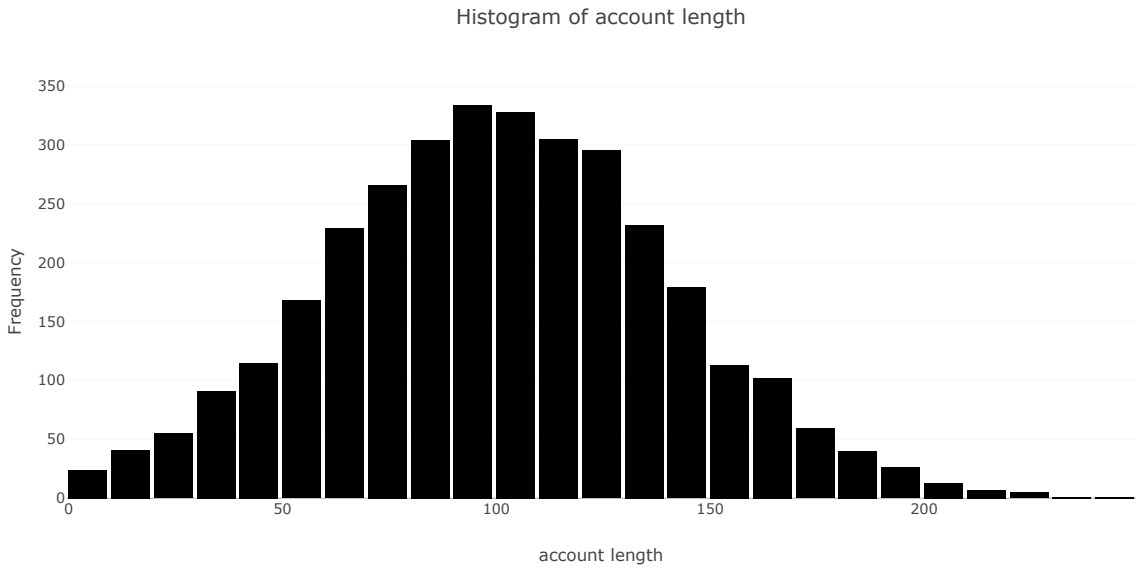
In [9]:
```
df.duplicated().sum()
```

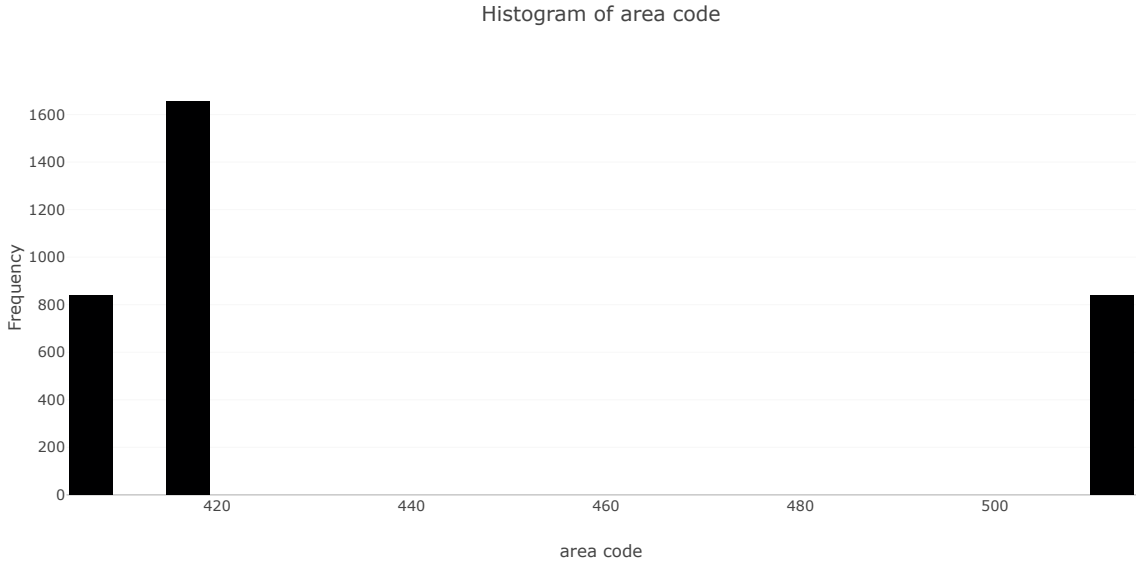
Out[9]:
```
np.int64(0)
```

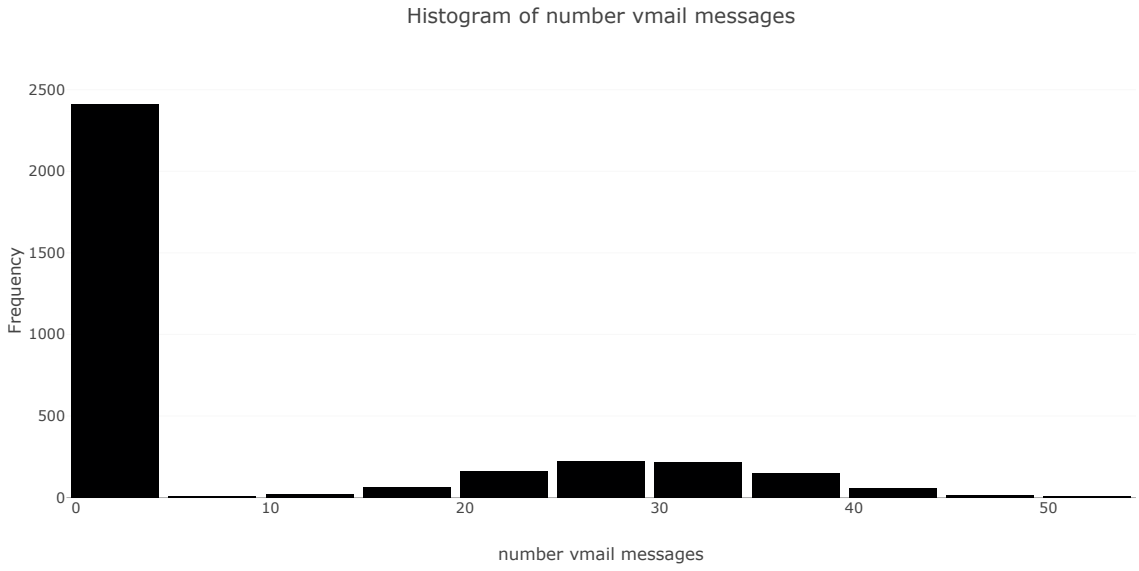
In [10]:
```
df.nunique()
```

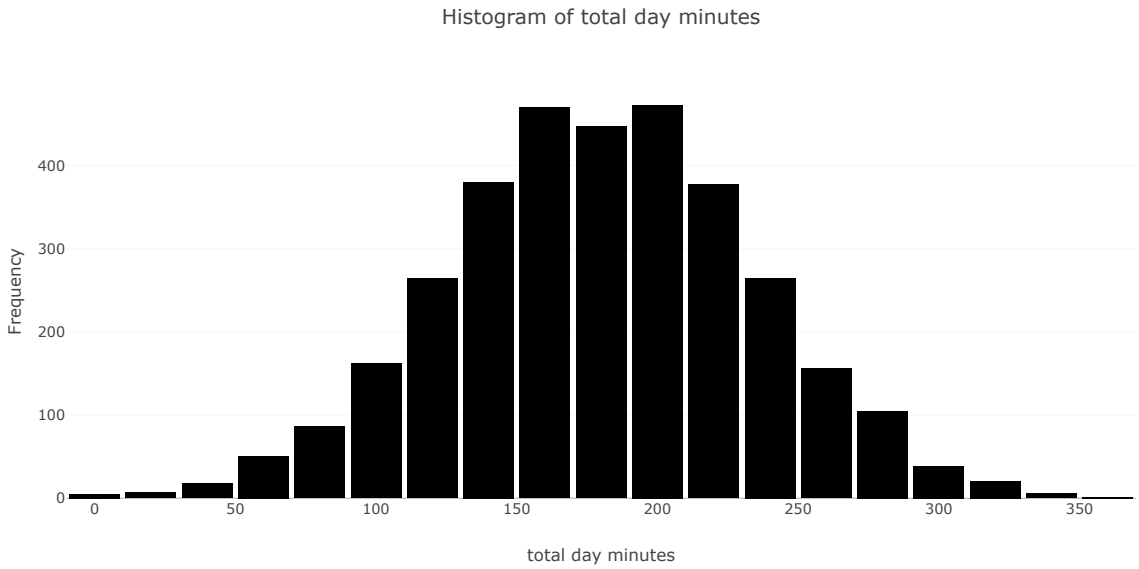
Out[10]:
```
state                     51
account length           212
area code                  3
phone number            3333
international plan         2
voice mail plan            2
number vmail messages     46
total day minutes       1667
total day calls          119
total day charge        1667
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
```

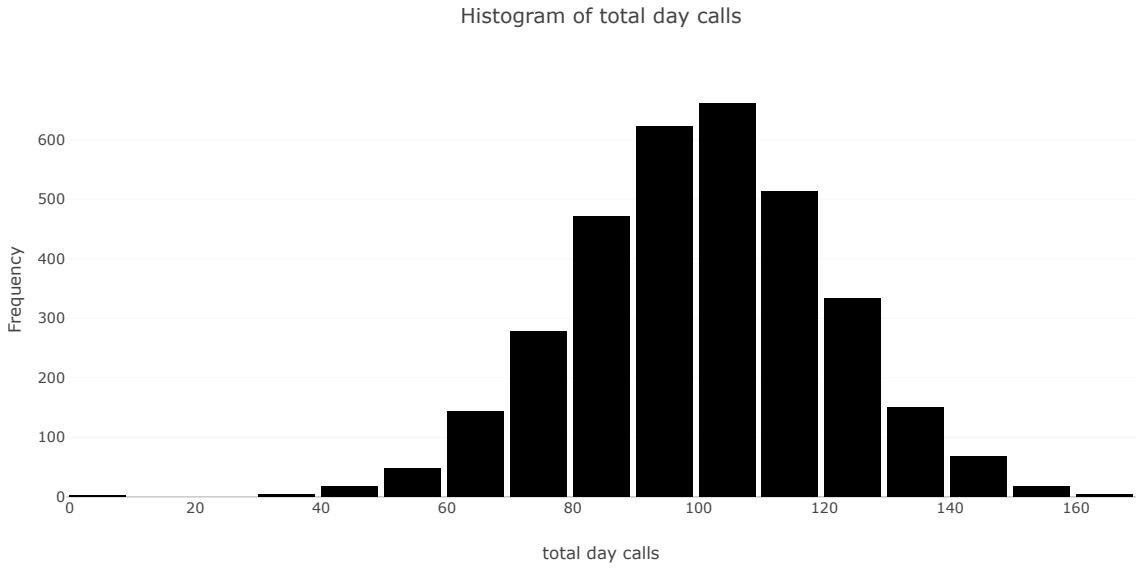
In [11]:
```
# Exploratory Data Analysis

# 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()
```

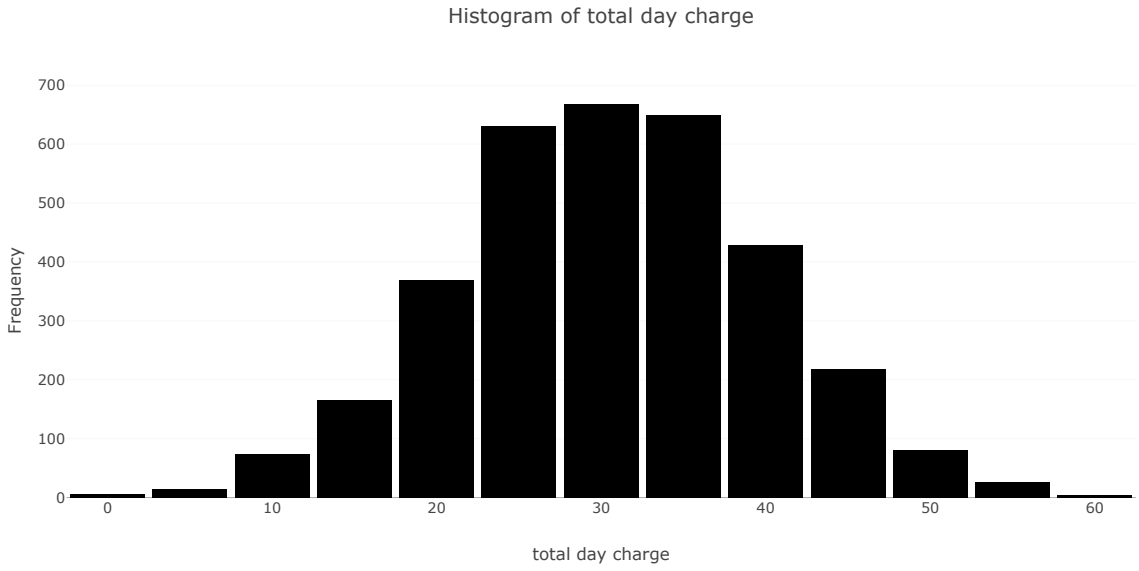

Histogram of account length
Frequency
0
50
100
150
200
250
300
350
0
50
100
150
200
account length

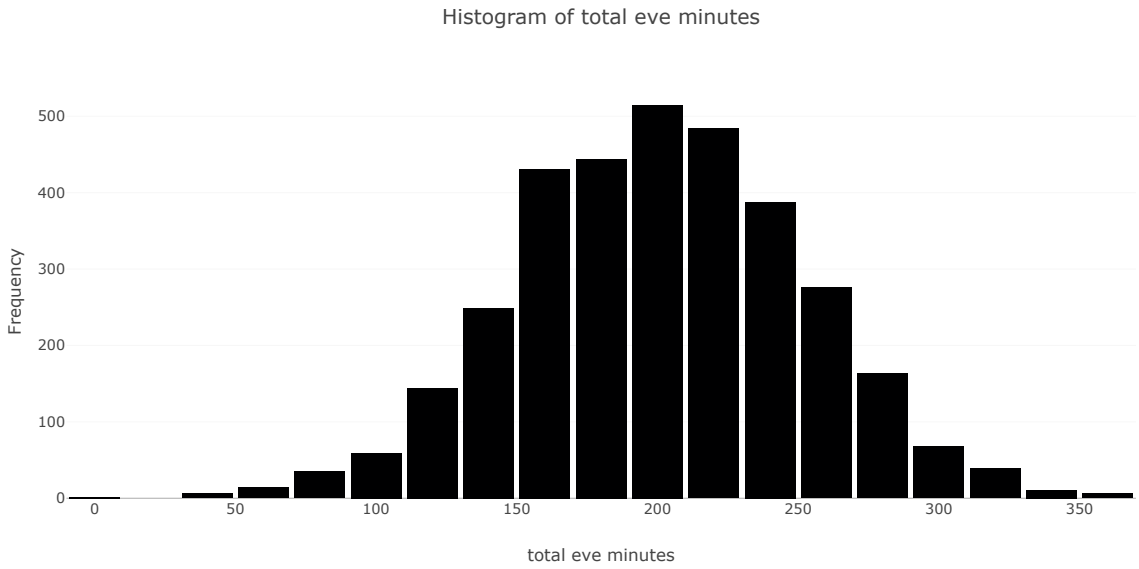

Histogram of area code
Frequency
0
200
400
600
800
1000
1200
1400
1600
420
440
460
480
500
area code

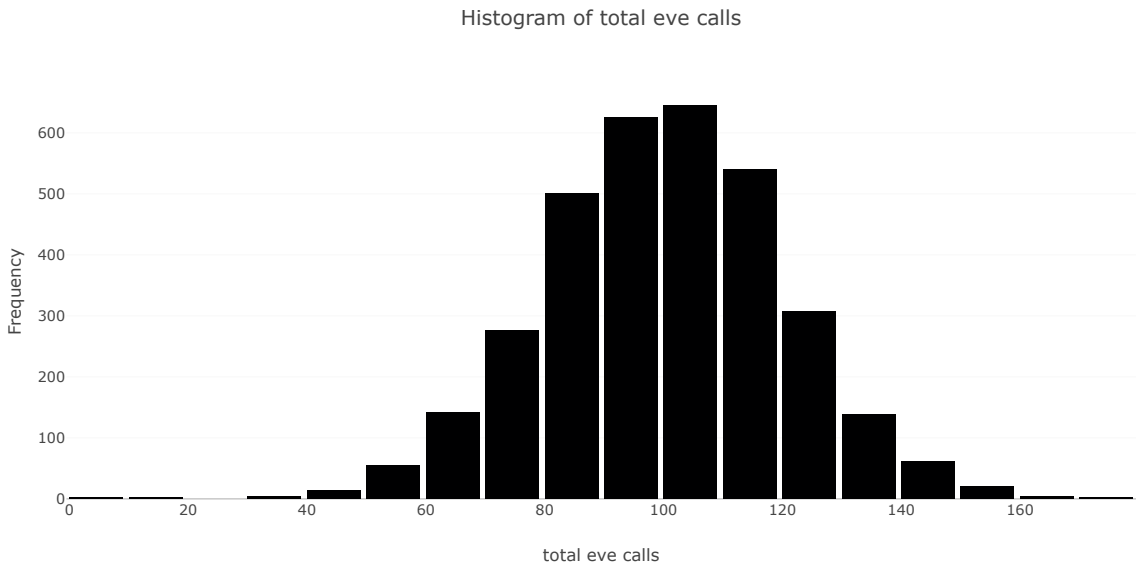

Histogram of number vmail messages
Frequency
0
500
1000
1500
2000
2500
0
10
20
30
40
50
number vmail messages

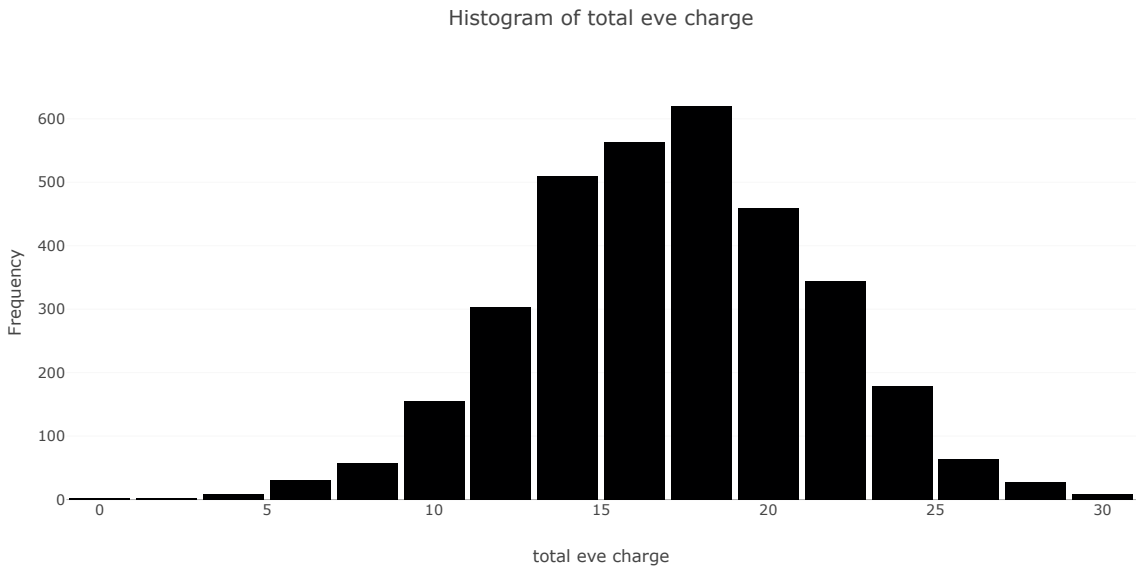

Histogram of total day minutes
Frequency
0
100
200
300
400
0
50
100
150
200
250
300
350
total day minutes

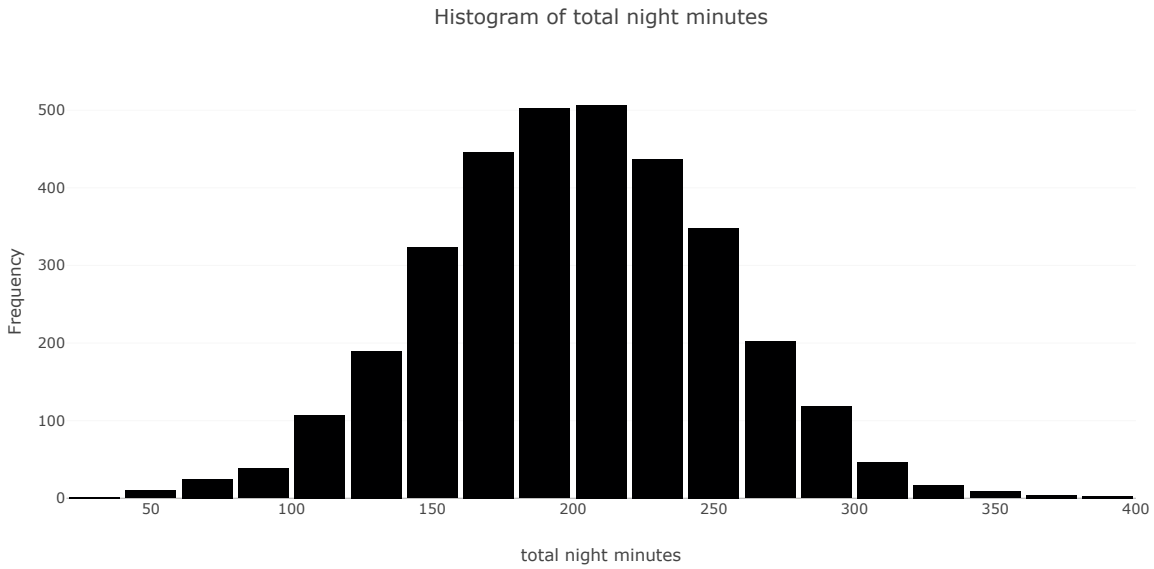

Histogram of total day calls
Frequency
0
100
200
300
400
500
600
0
20
40
60
80
100
120
140
160
total day calls

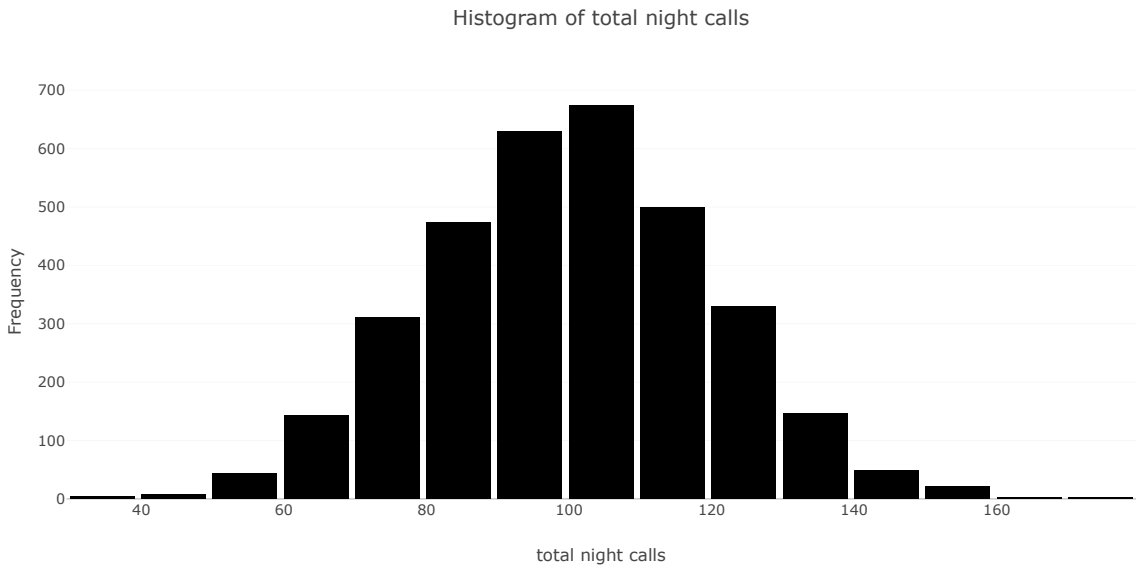

Histogram of total day charge
Frequency
0
100
200
300
400
500
600
700
0
10
20
30
40
50
60
total day charge

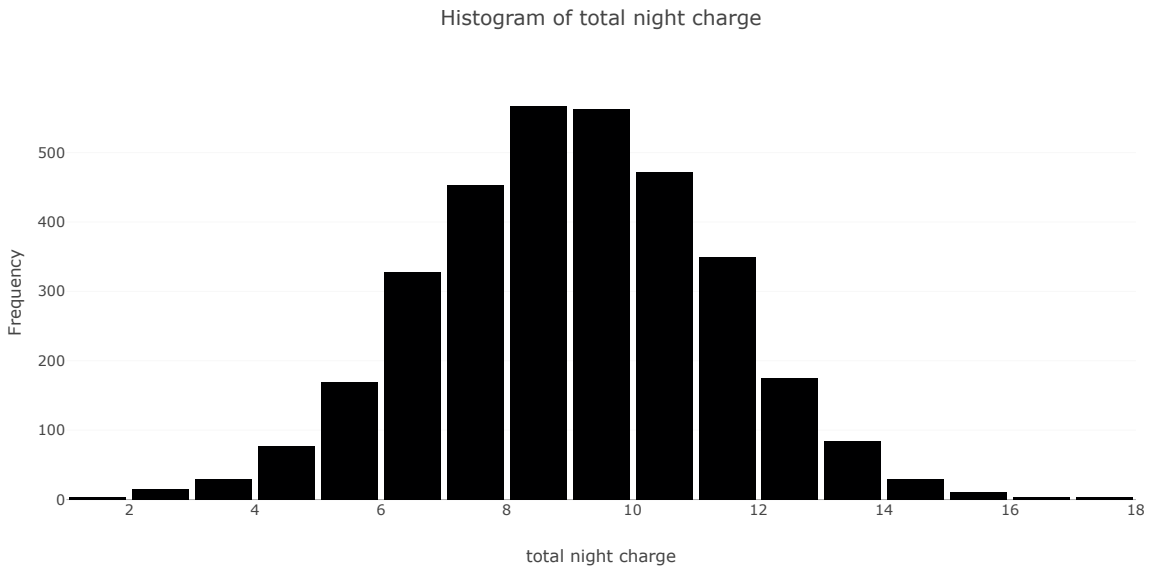

Histogram of total eve minutes
Frequency
0
100
200
300
400
500
0
50
100
150
200
250
300
350
total eve minutes

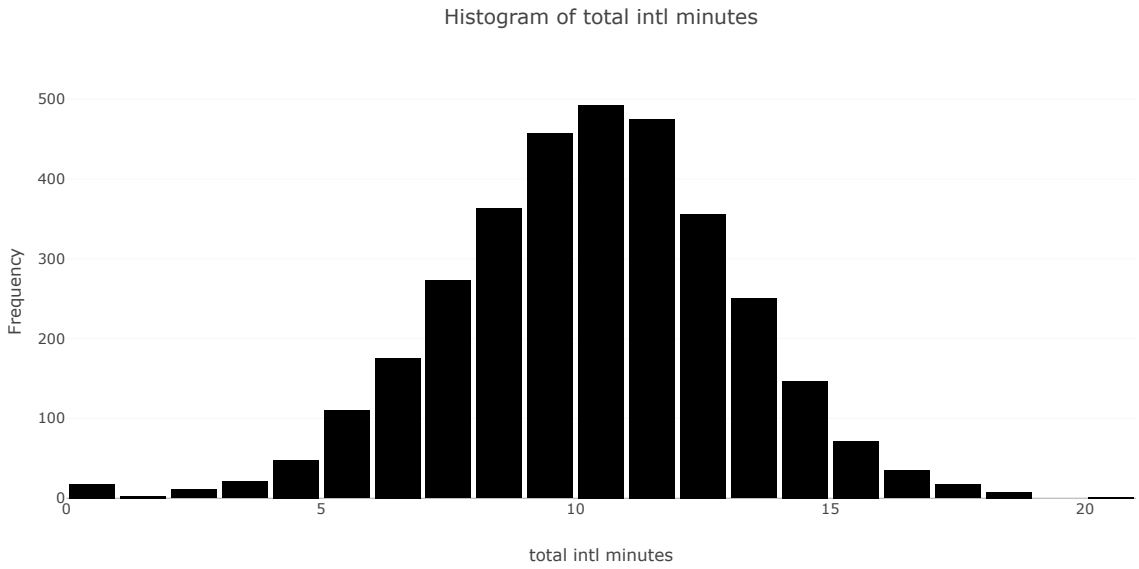

Histogram of total eve calls
Frequency
0
100
200
300
400
500
600
0
20
40
60
80
100
120
140
160
total eve calls

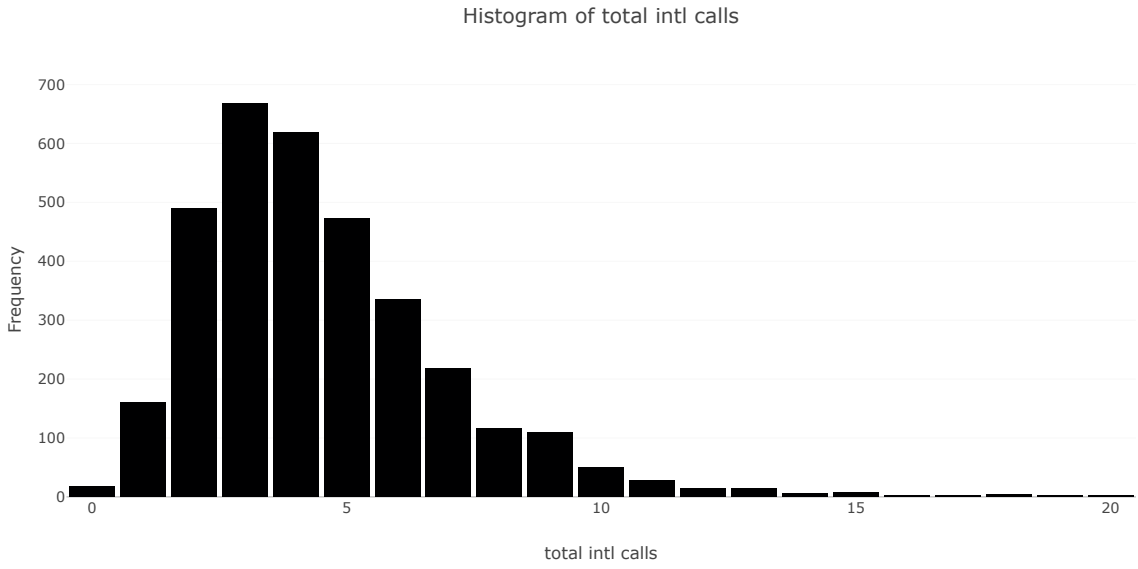

Histogram of total eve charge
Frequency
0
100
200
300
400
500
600
0
5
10
15
20
25
30
total eve charge

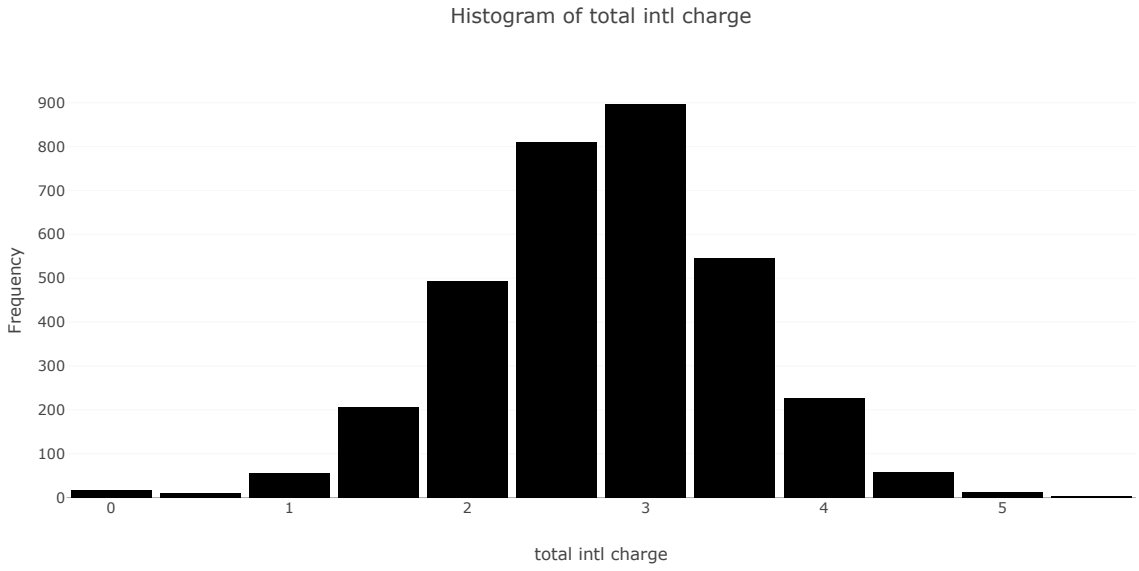

Histogram of total night minutes
Frequency
0
100
200
300
400
500
50
100
150
200
250
300
350
400
total night minutes

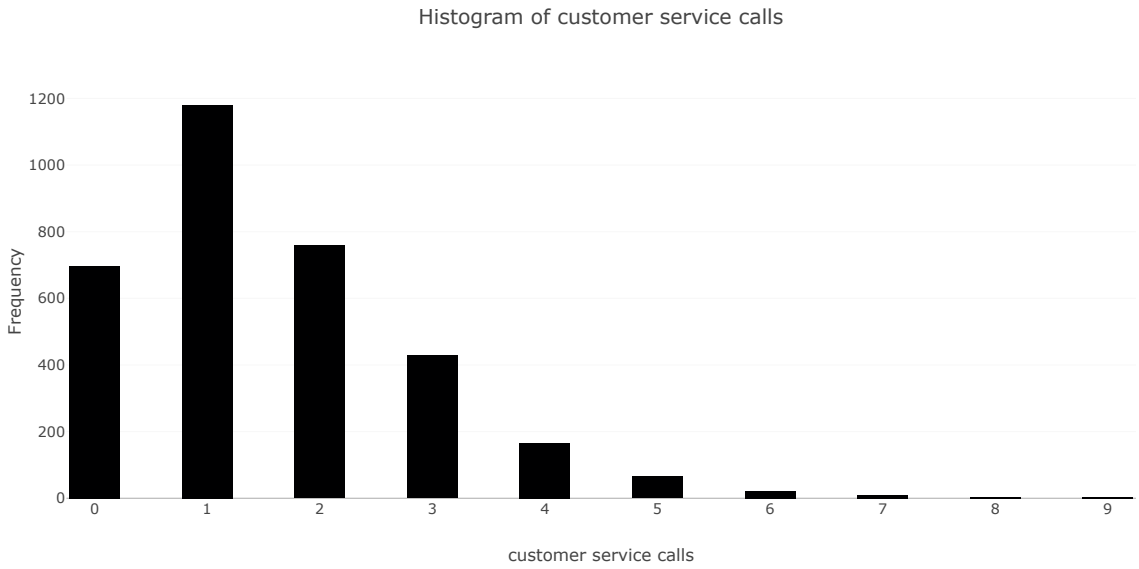

Histogram of total night calls
Frequency
0
100
200
300
400
500
600
700
40
60
80
100
120
140
160
total night calls


Histogram of total night charge
Frequency
0
100
200
300
400
500
2
4
6
8
10
12
14
16
18
total night charge


Histogram of total intl minutes
Frequency
0
100
200
300
400
500
0
5
10
15
20
total intl minutes


Histogram of total intl calls
Frequency
0
100
200
300
400
500
600
700
0
5
10
15
20
total intl calls

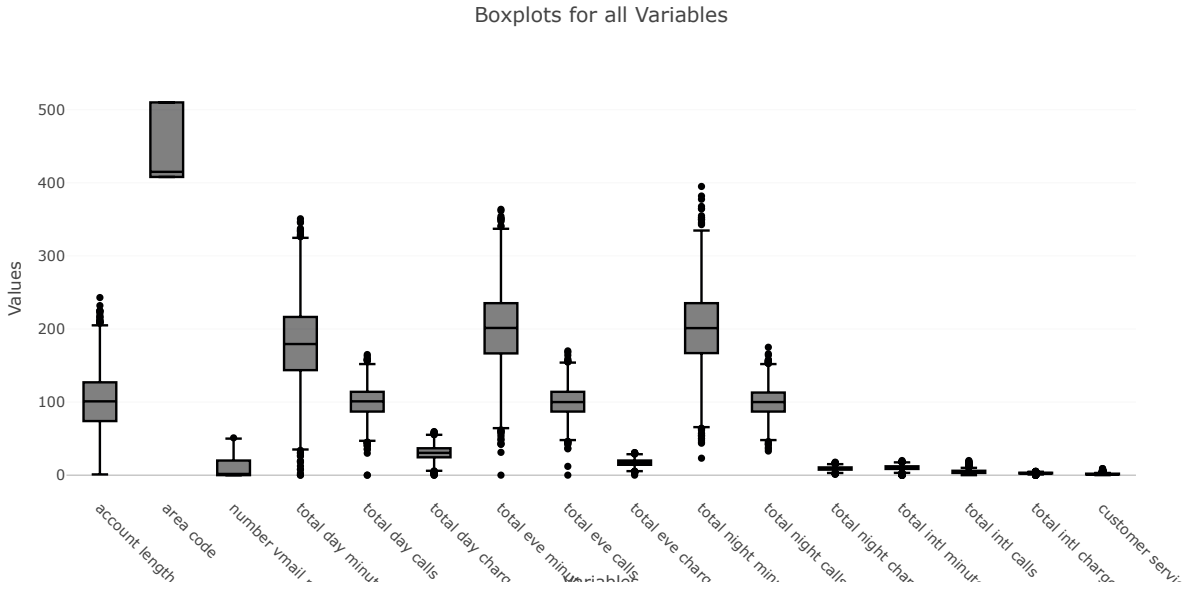

Histogram of total intl charge
Frequency
0
100
200
300
400
500
600
700
800
900
0
1
2
3
4
5
total intl charge



- 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

In [12]:
```
# 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()
```



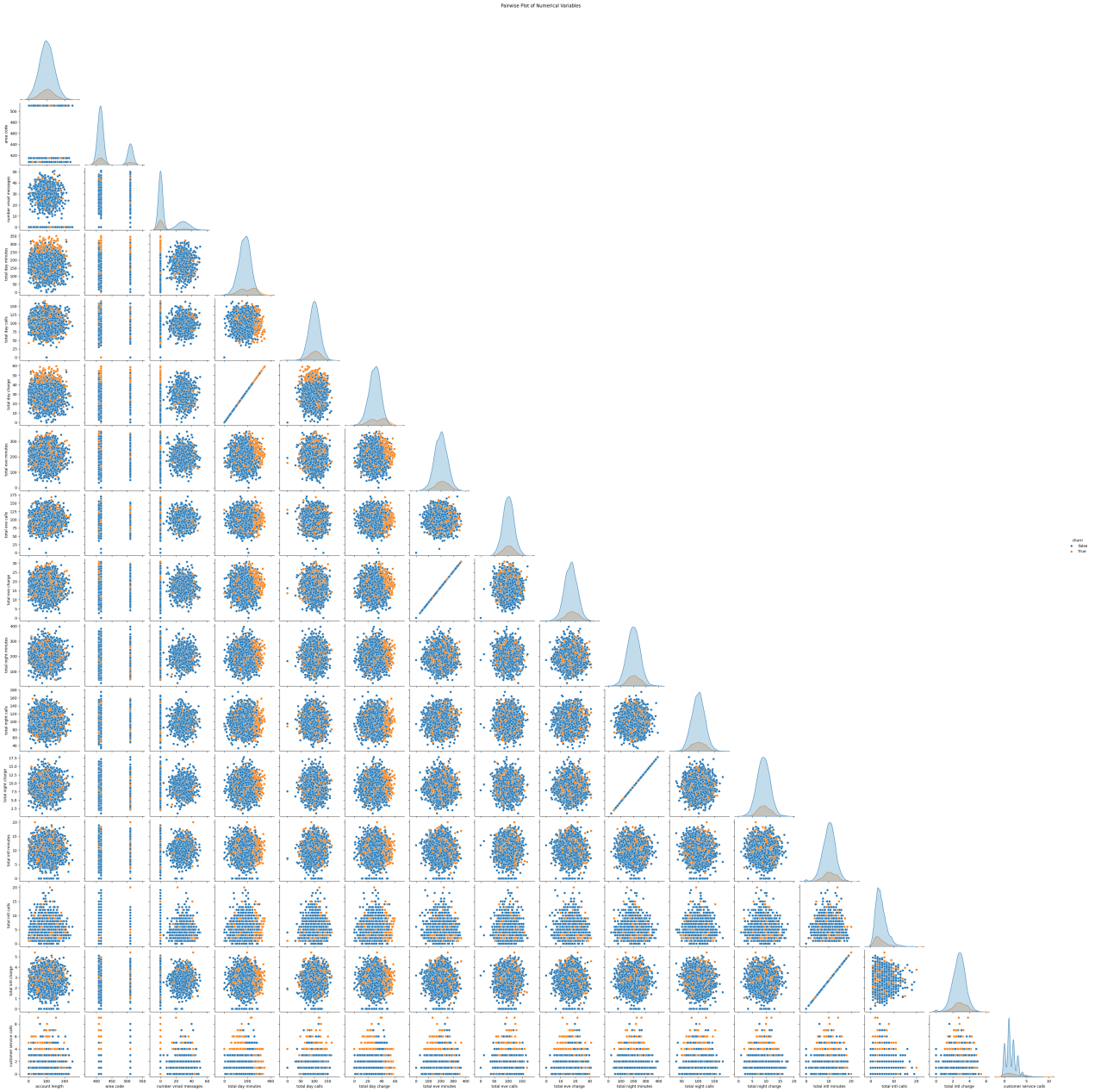
In [13]:
```
# Getting counts of outliers

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
```

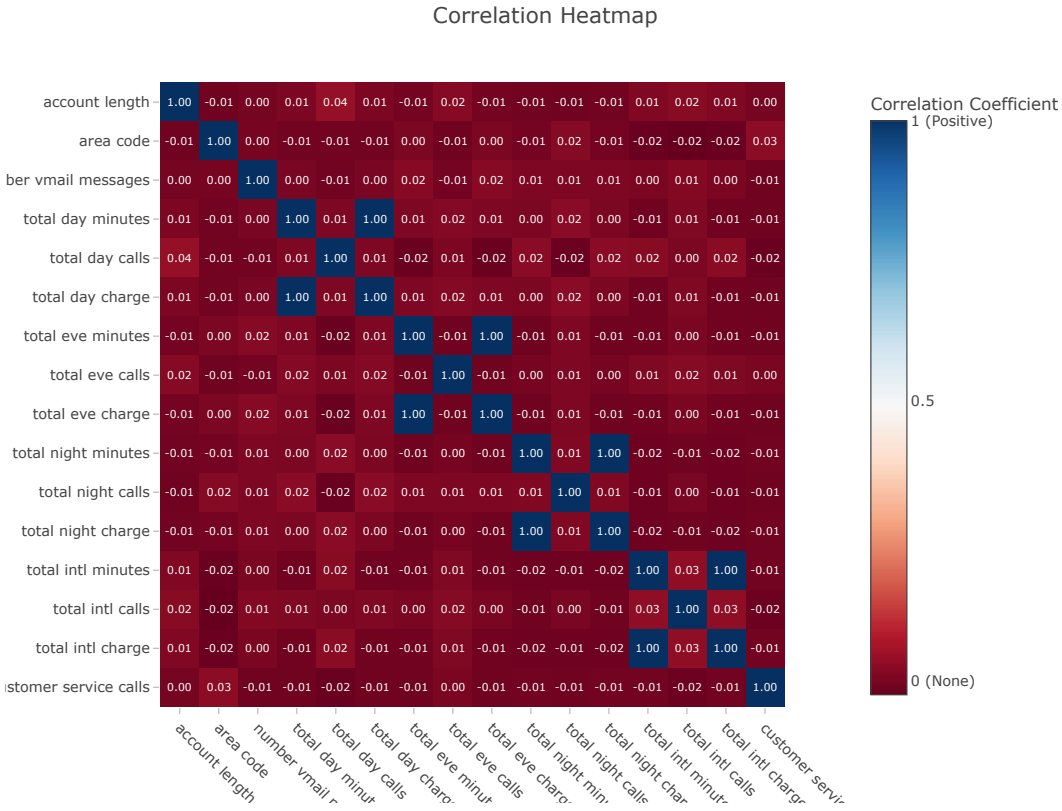
- There are quite a few outliers in the variables however, we will deal with them depending on their correlation so we can accordingly transform or remove them.

In [14]:
```
# Bivariate Analysis and Correlation Matrix
imp_cols = df[numerical_cols].assign(churn=df['churn'])
%matplotlib inline
sns.pairplot(imp_cols, diag_kind='kde', corner=True, hue='churn')
plt.suptitle('Pairwise Plot of Numerical Variables', y=1.02)
plt.show()
```



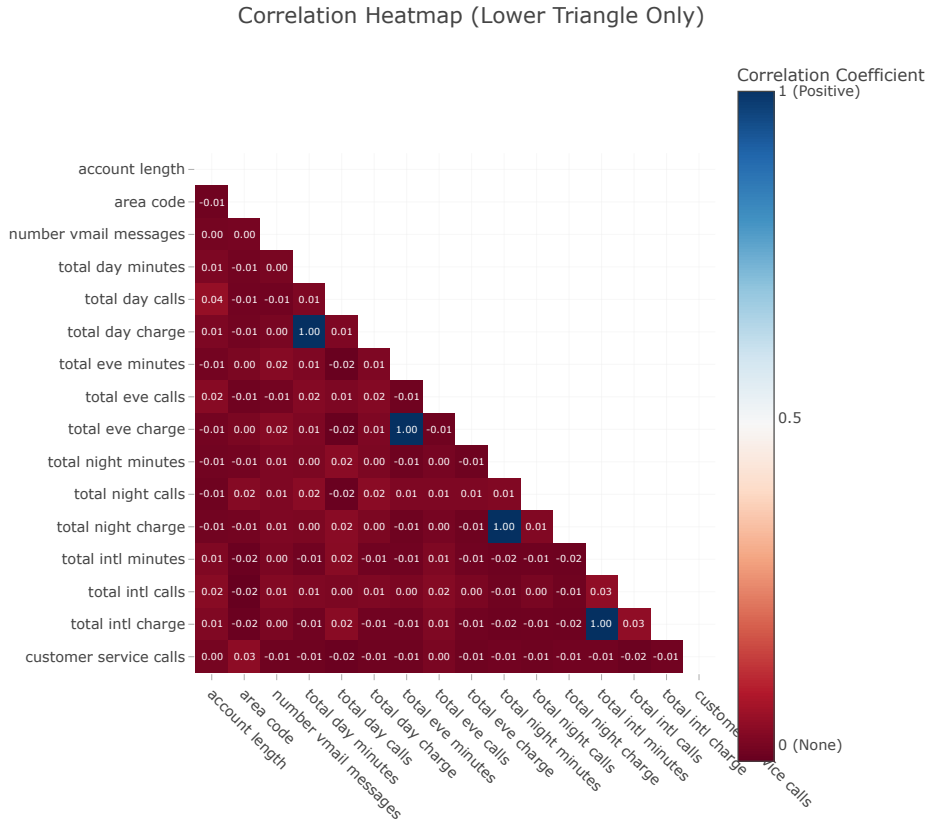
1. We can see that some variables are correlated with each other such as total day charge and total day minutes, total eve charge and total eve minutes, total night charge and total night minutes, total international charge and total international minutes.
2. Some variables are not normal distributed and have multiple peaks possibly suggesting grouping such as number of voicemail messages and customer service calls which could indicate 2 segments of customers

In [15]:
```
correlation_matrix = df[numerical_cols].corr()

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

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 legend title
        tickvals=[-1, -0.5, 0, 0.5, 1],  # Ticks for the color bar
        ticktext=["-1 (Negative)", "-0.5", "0 (None)", "0.5", "1 (Positive)"],  # Label ticks
    ),
    width=900,  # Width of the heatmap
    height=700,  # Adjust the height
)

fig.show()
```



In [16]:
```
# 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",  # 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,
```

```
    margin=dict(
        l=170,  # Add left padding
        r=50,   # Add right padding
        t=50,   # Add top padding
        b=150   # Add bottom padding
    ),
)

fig.show()
```



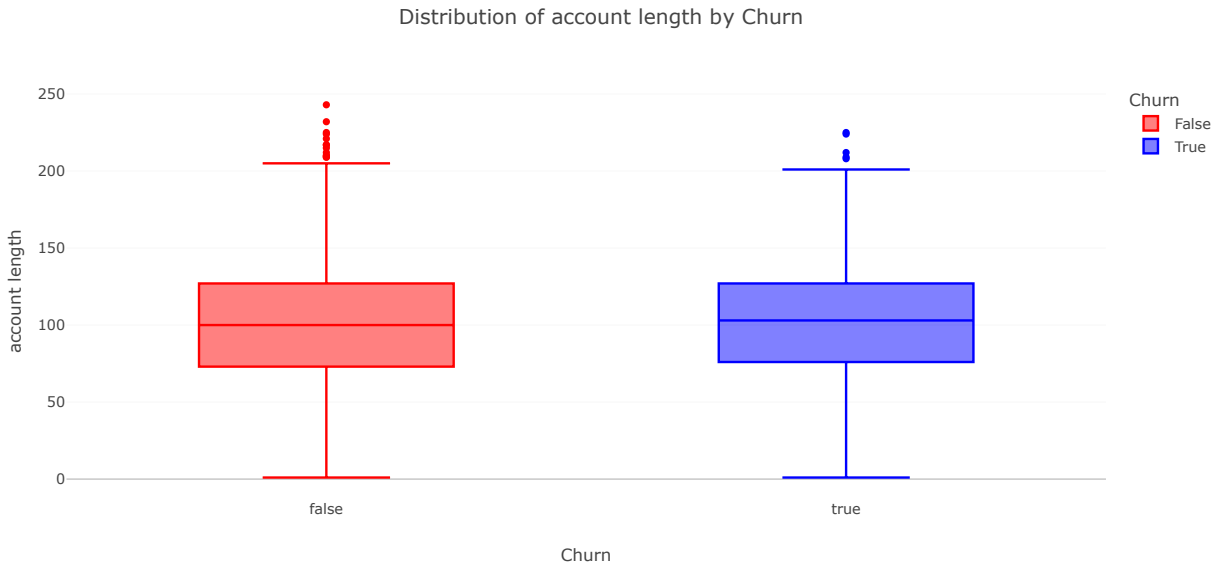
- The relationship between variables observed in the correlation matrix is consistent with the pairplot.
- However, upon looking at the distributions with churn as a feature, we can see that the dataset is imbalanced. To confirm this we look at the distribution of churn.

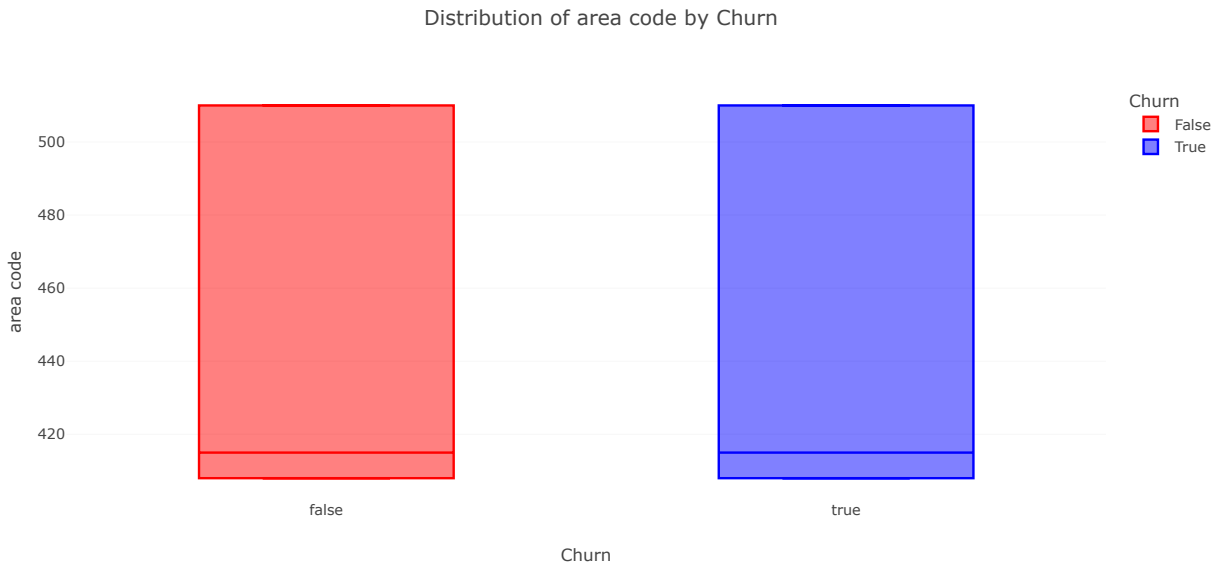
In [17]:
```
churn_distribution = df['churn'].value_counts(normalize=True)
print(churn_distribution)
```

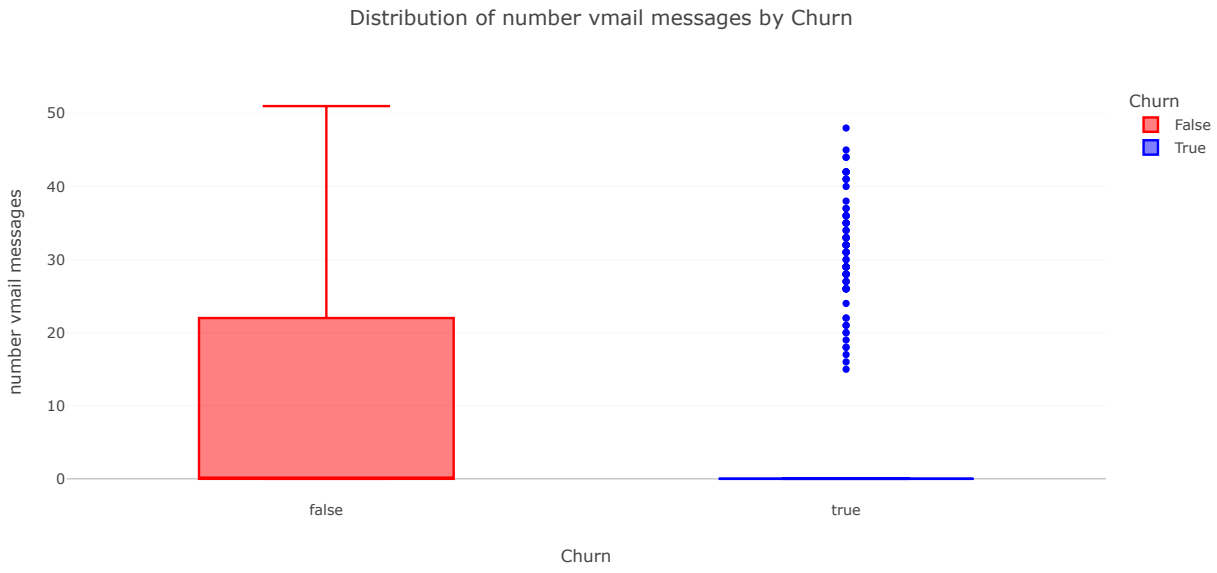
```
churn
False    0.855086
True     0.144914
Name: proportion, dtype: float64
```

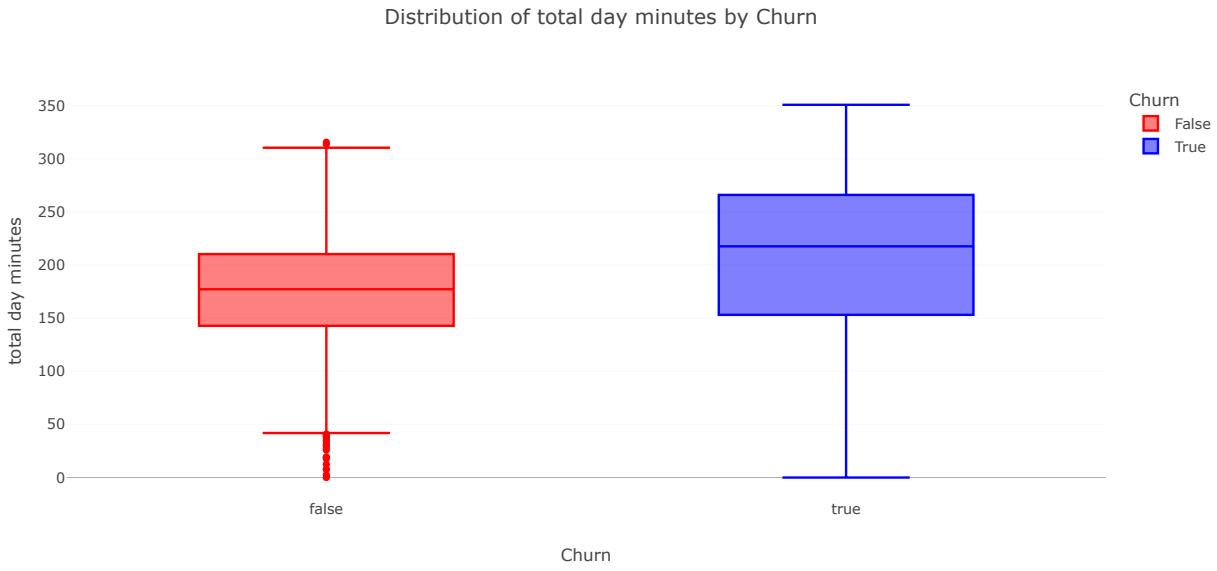
86% of customers did not churn while 14% of customers churned.

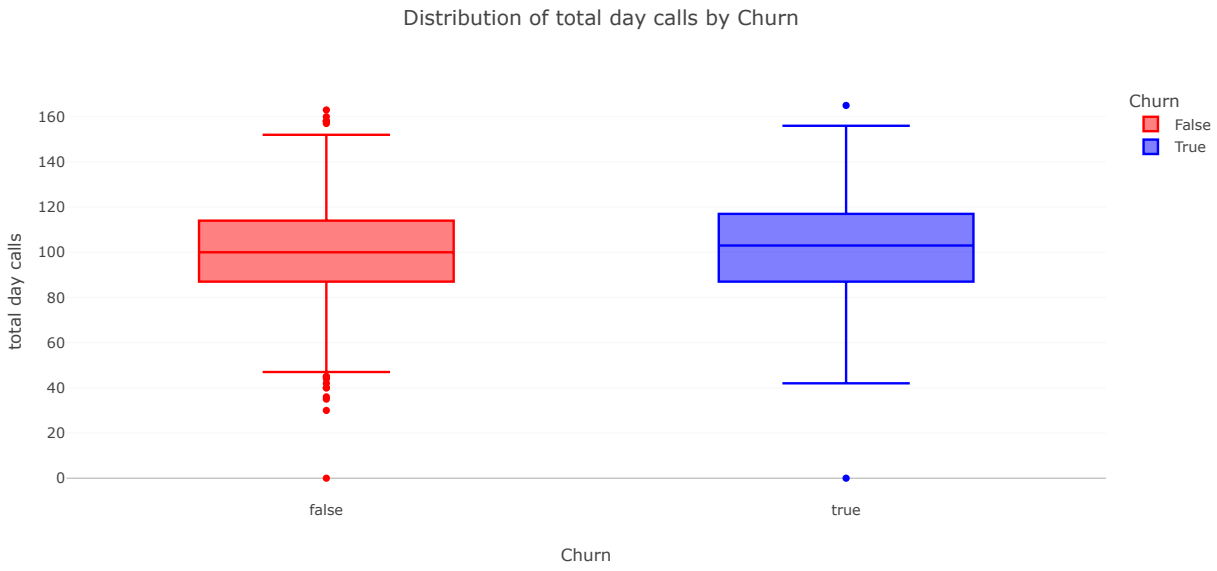
In [18]:
```
# Using boxplots to compare numerical features by churn
for col in numerical_cols:
    fig = px.box(df, x='churn', y=col, color='churn',
                 title=f'Distribution of {col} by Churn',
                 labels={'churn': 'Churn', col: col},
                 color_discrete_map={True: "blue", False: "red"})
    fig.update_traces(quartilemethod="exclusive")
    fig.show()
```

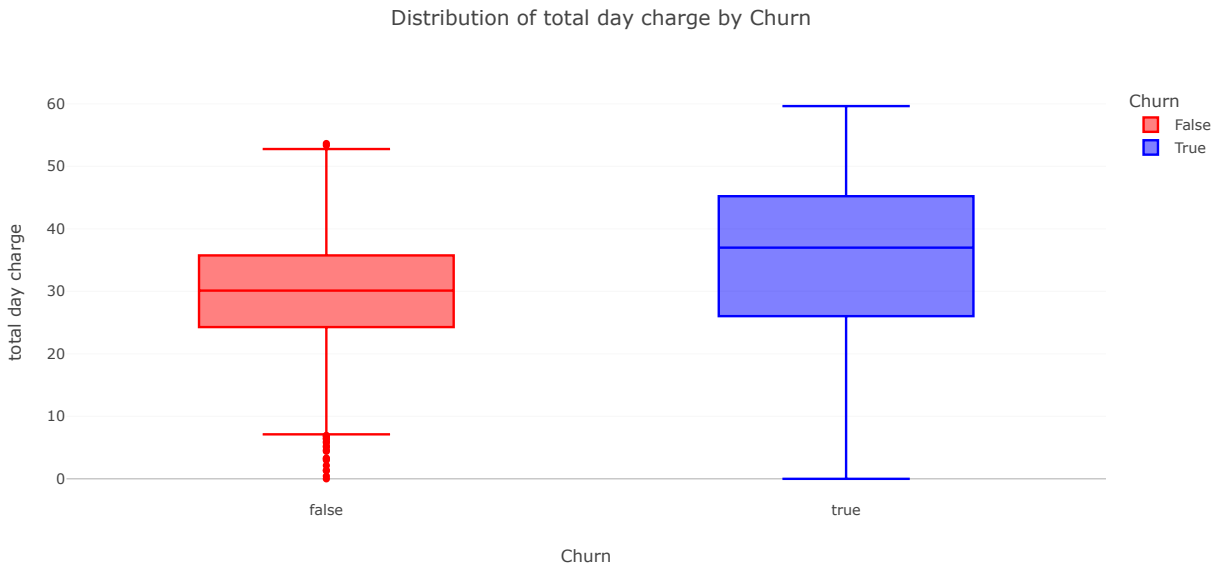

Distribution of account length by Churn
account length
250
200
150
100
50
0
false
true
Churn
Churn
False
True

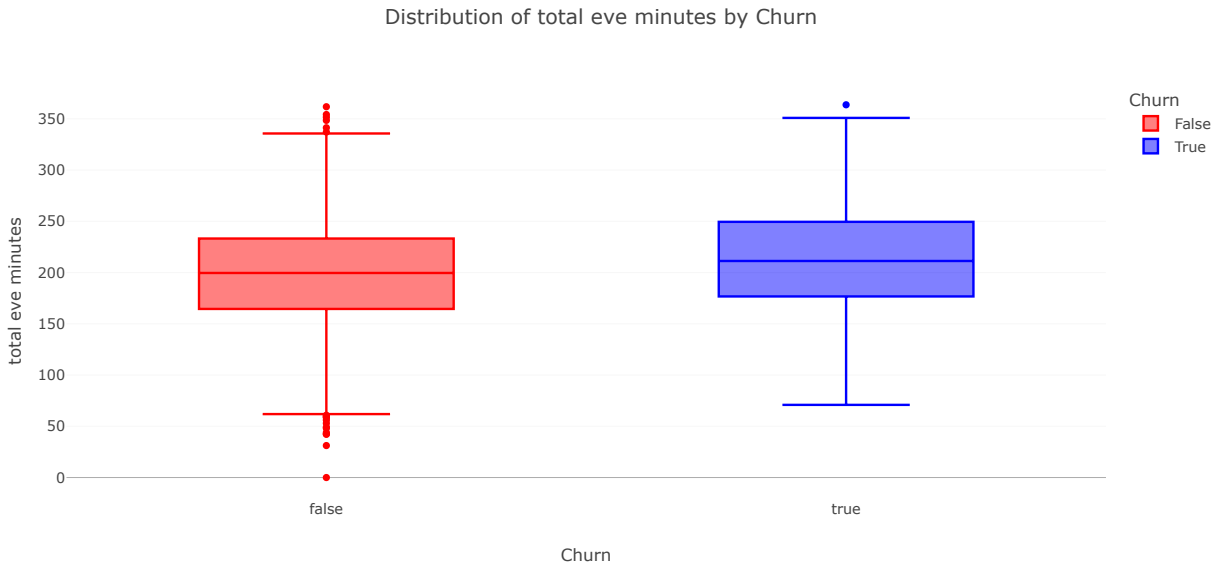

Distribution of area code by Churn
area code
500
480
460
440
420
false
true
Churn
Churn
False
True

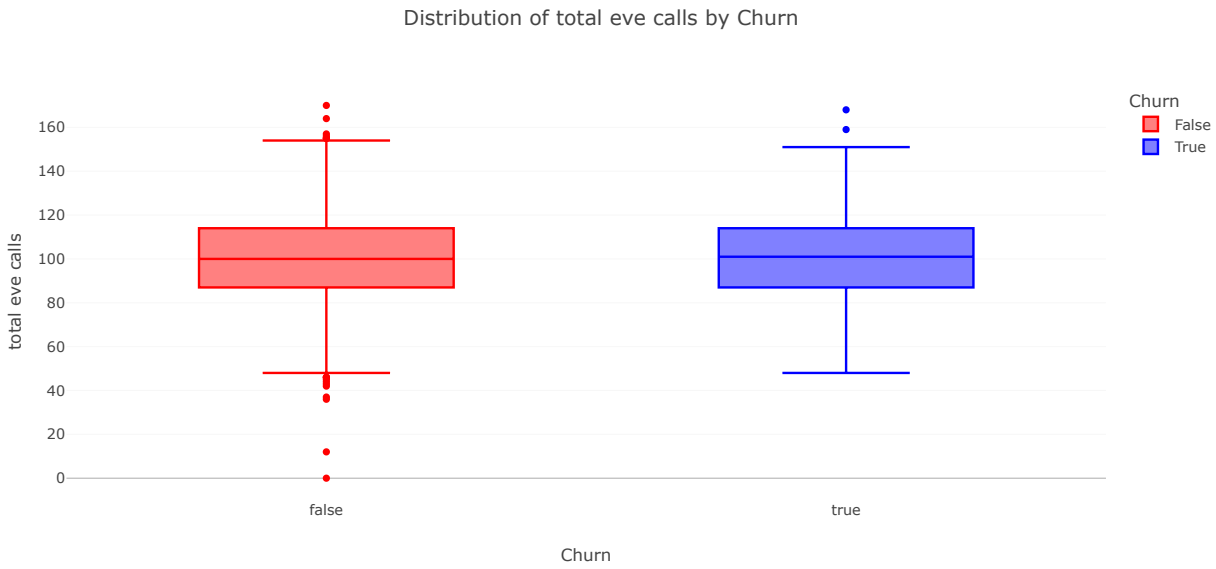

Distribution of number vmail messages by Churn
number vmail messages
50
40
30
20
10
0
false
true
Churn
Churn
False
True

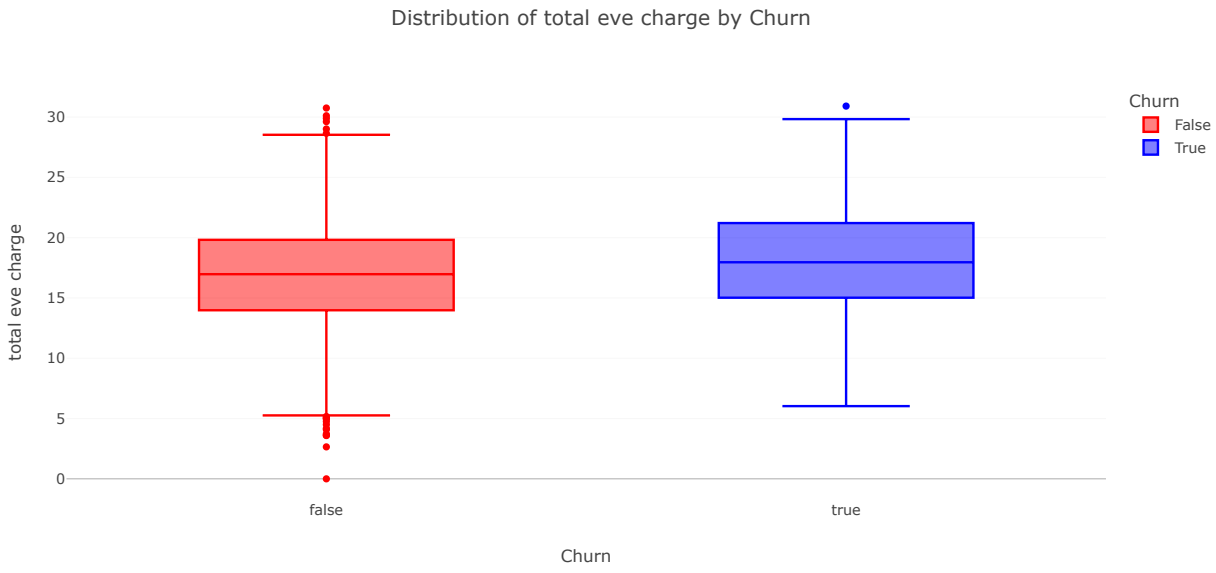

Distribution of total day minutes by Churn
total day minutes
350
300
250
200
150
100
50
0
false
true
Churn
Churn
False
True

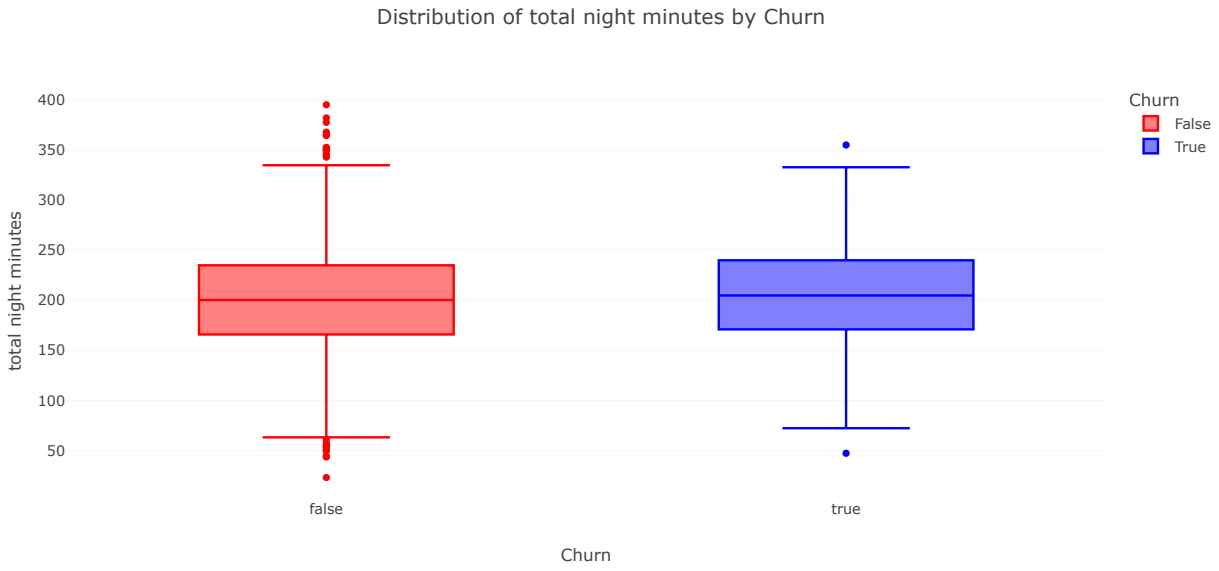

Distribution of total day calls by Churn
total day calls
160
140
120
100
80
60
40
20
0
false
true
Churn
Churn
False
True

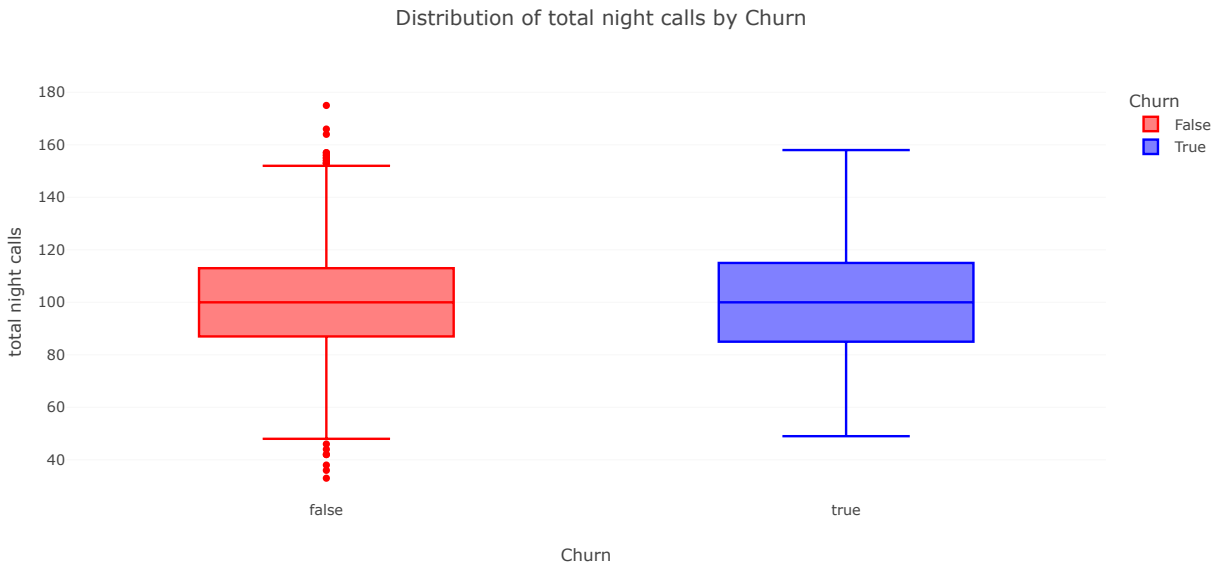

Distribution of total day charge by Churn
total day charge
60
50
40
30
20
10
0
false
true
Churn
Churn
False
True

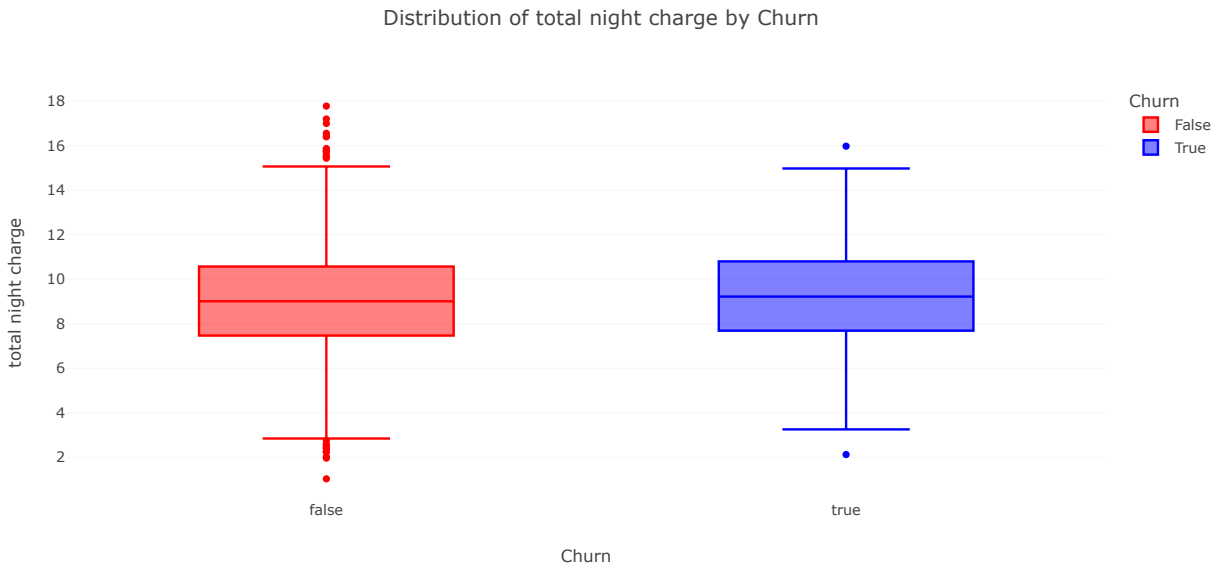

Distribution of total eve minutes by Churn
total eve minutes
0
50
100
150
200
250
300
350
false
true
Churn
Churn
False
True

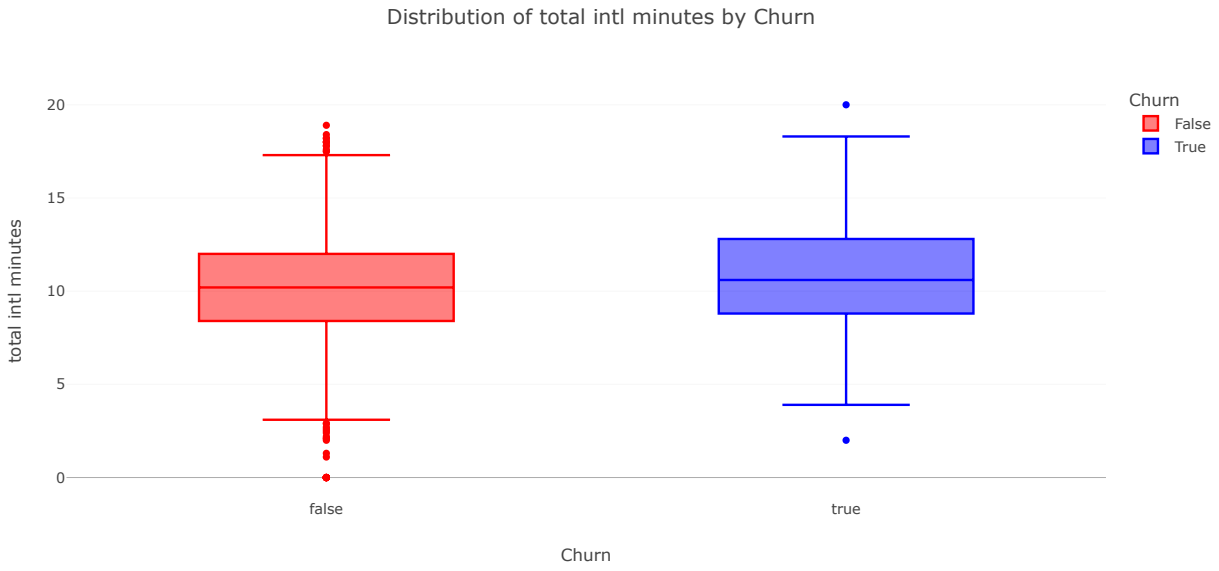

Distribution of total eve calls by Churn
total eve calls
0
20
40
60
80
100
120
140
160
false
true
Churn
Churn
False
True

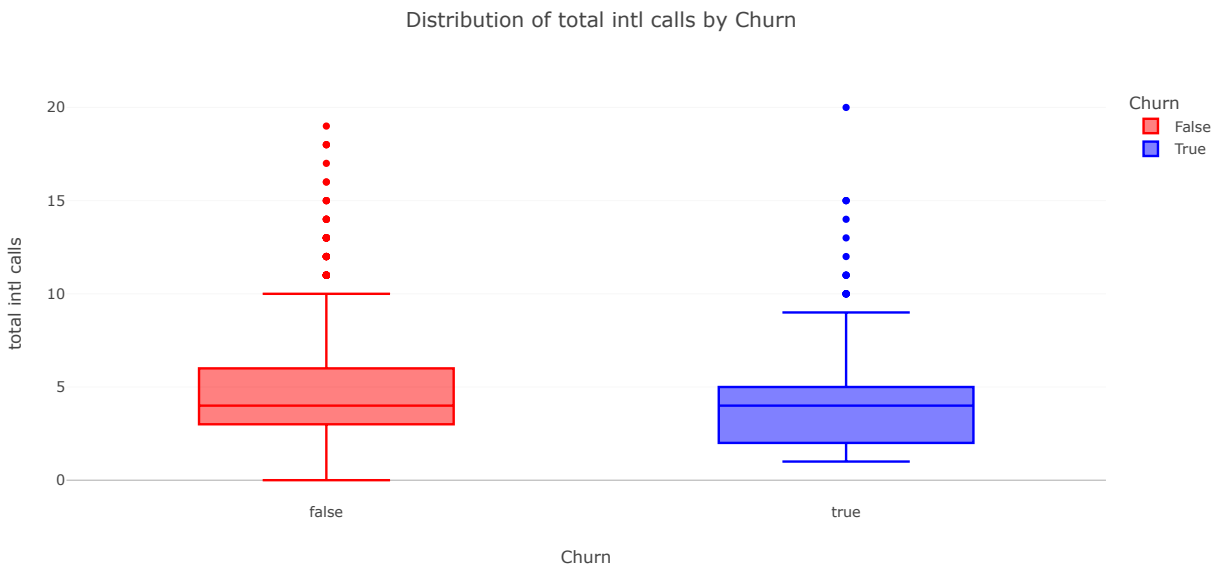

Distribution of total eve charge by Churn
total eve charge
0
5
10
15
20
25
30
false
true
Churn
Churn
False
True

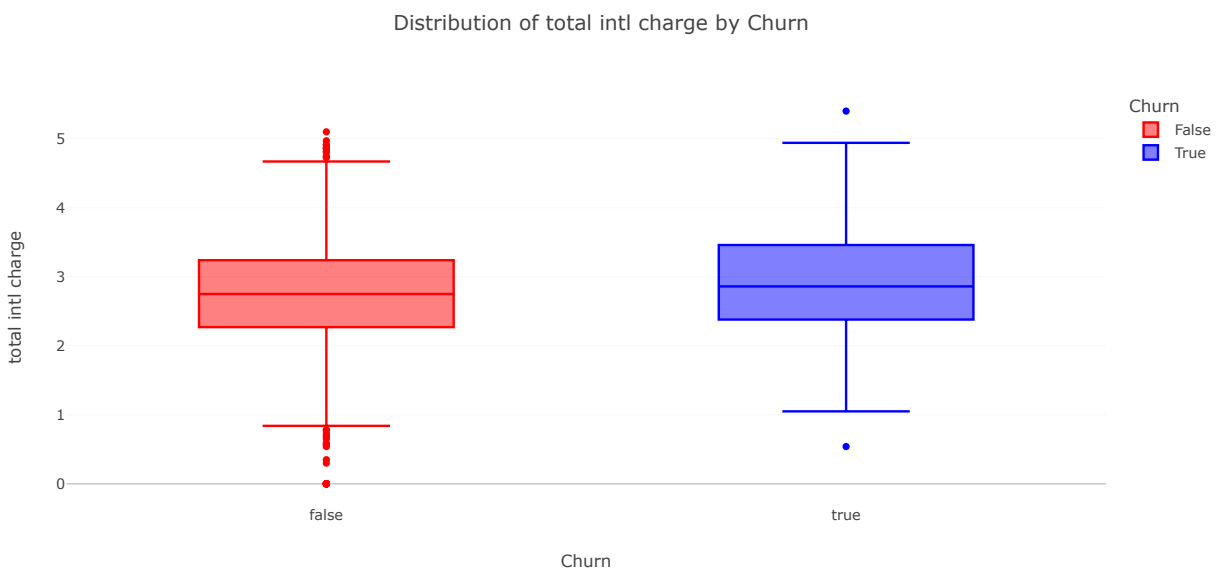

Distribution of total night minutes by Churn
total night minutes
400
350
300
250
200
150
100
50
false
true
Churn
Churn
False
True

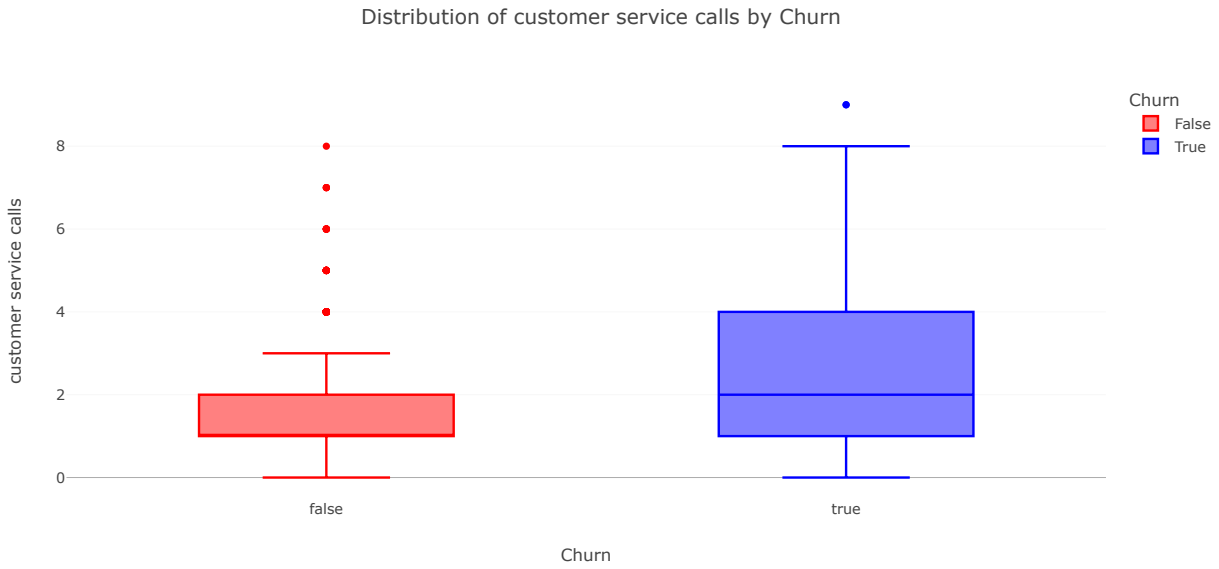

Distribution of total night calls by Churn
total night calls
180
160
140
120
100
80
60
40
false
true
Churn
Churn
False
True


Distribution of total night charge by Churn
total night charge
18
16
14
12
10
8
6
4
2
false
true
Churn
Churn
False
True


Distribution of total intl minutes by Churn
total intl minutes
20
15
10
5
0
false
true
Churn
Churn
False
True


Distribution of total intl calls by Churn
total intl calls
20
15
10
5
0
false
true
Churn
Churn
False
True


Distribution of total intl charge by Churn
total intl charge
5
4
3
2
1
0
false
true
Churn
Churn
False
True



1. 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.
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 voicemail messages. It appears that those who don't use voicemail services would be likely to churn.

In [19]:
```
# 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}

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

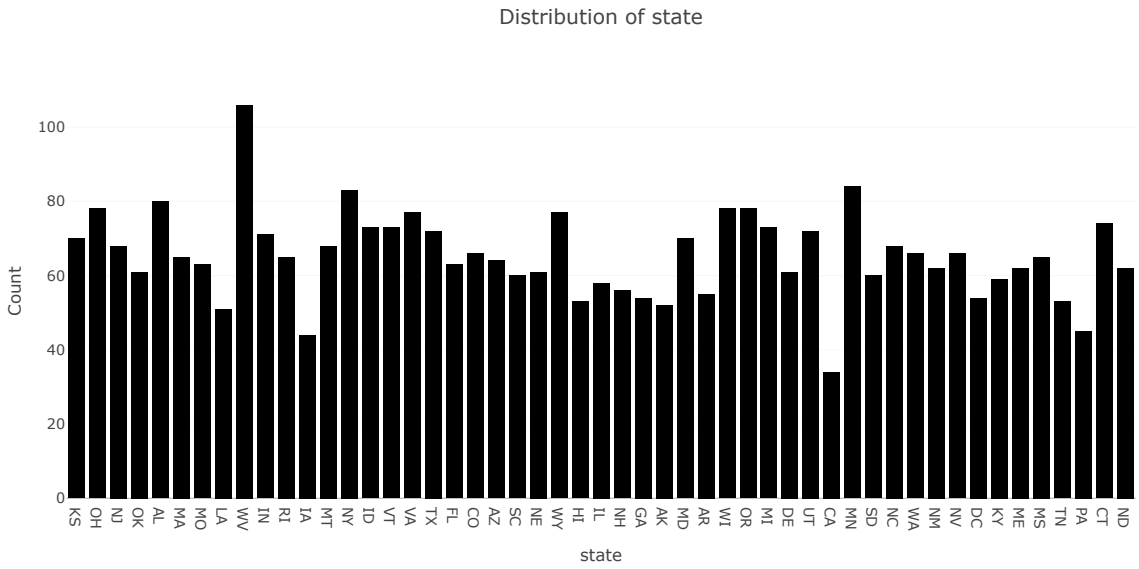
```
account length:
        T-statistic: 0.96, P-value: 0.336

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
```

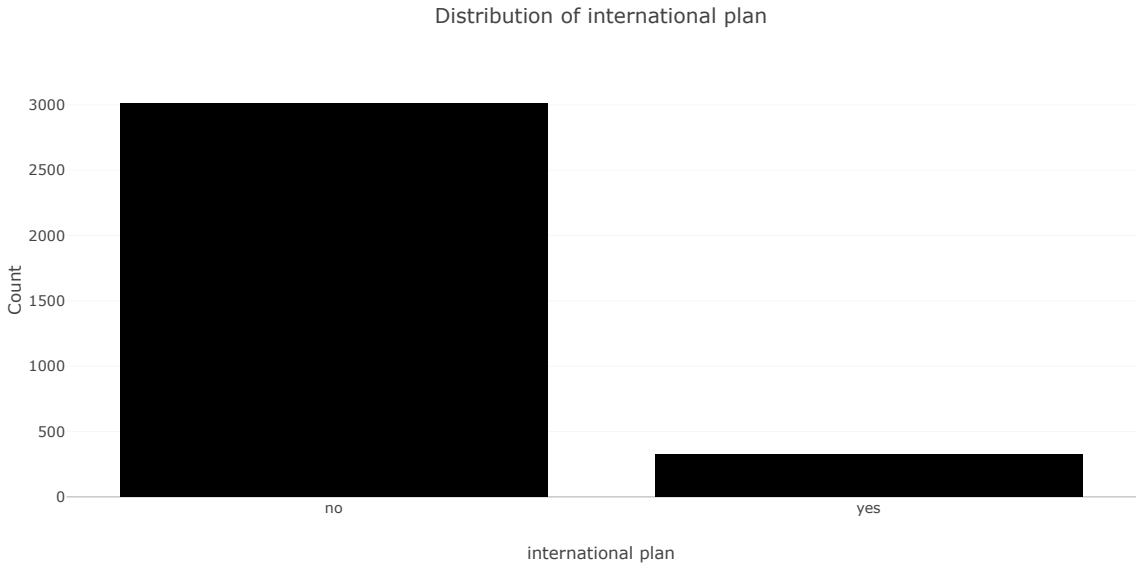
In [20]:
```
# Keeping the p value threshold as 0.05 we can see a significant association with
'''
1. number vmail messages
2. total day minutes
3. total day charge
4. total eve minutes
5. total eve charge
6. total night minutes
7. total night charge
8. total intl minutes
9. total intl charge
10. customer service calls
'''
# this indicates that churned and non-churned customers have a significantly different average 1-10 ^^
```

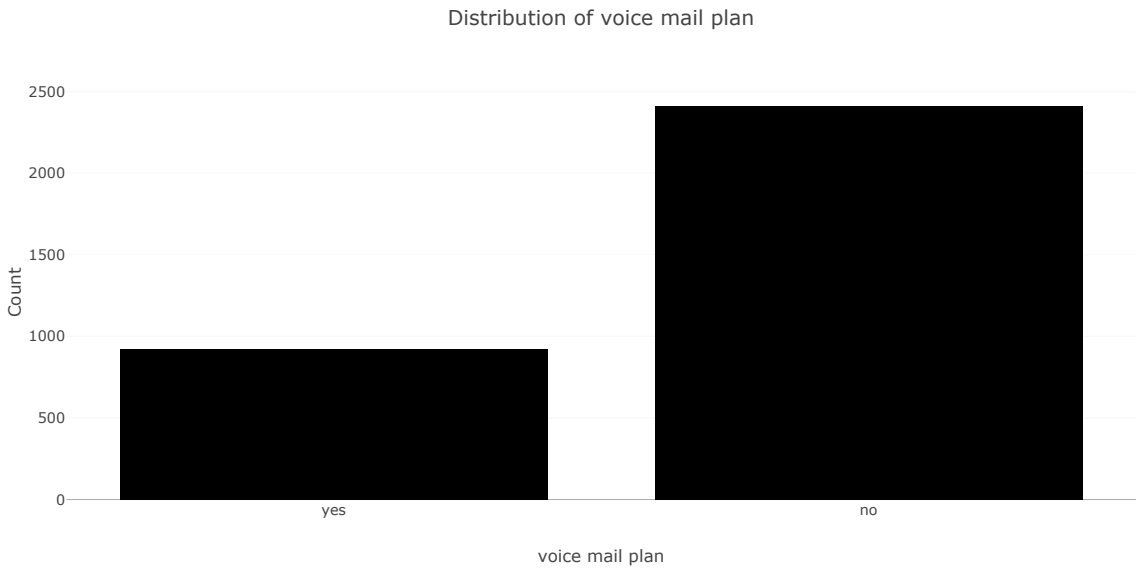
Out[20]:
```
'\n1. number vmail messages\n2. total day minutes\n3. total day charge\n4. total eve minutes\n5. total eve charge\n6. total night minutes\n7. to
tal night charge\n8. total intl minutes\n9. total intl charge\n10. customer service calls\n'
```

In [21]:
```
# We move onto analyse the categorical variables.
```

In [22]:
```
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()
```





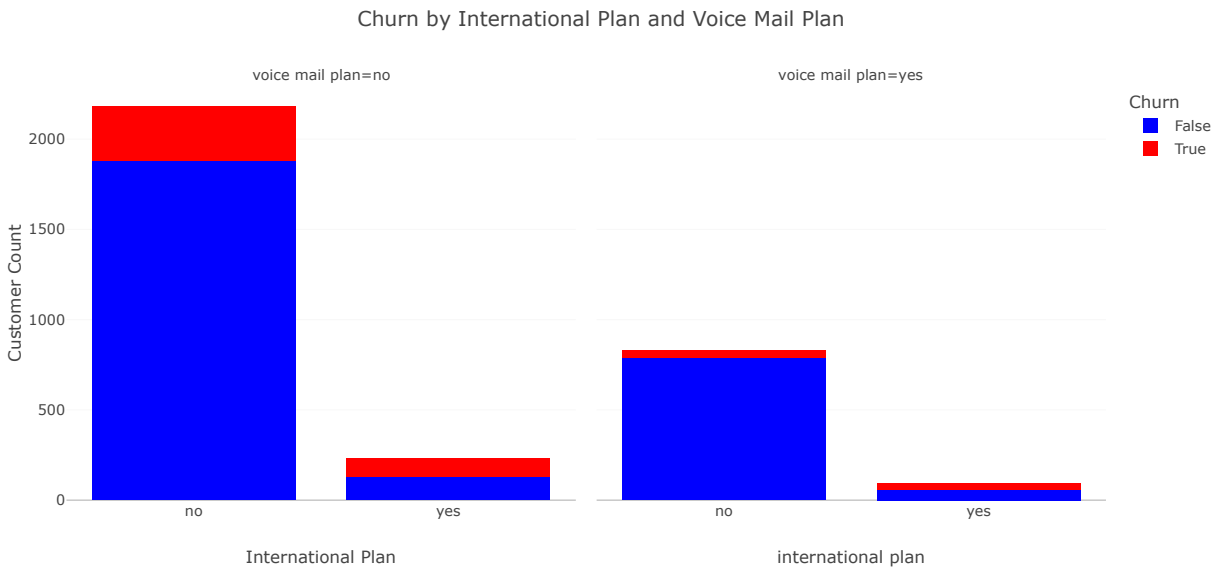


- Majority of the customers did not have an international plan and did not have a voice mail plan.
- Further drilling down using the churned variable.

In [23]:
```
# 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 [24]:
```
# 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 [25]:
```
from scipy.stats import chi2_contingency

contingency_table = pd.crosstab([df['international plan'], df['voice mail plan']], df['churn'])
print(contingency_table)

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 [26]:
```
# Calculating churn rate
contingency_table['Total'] = contingency_table[False] + contingency_table[True]

contingency_table['Churn Rate'] = contingency_table[True] / contingency_table['Total']

print(contingency_table)
```

```
churn                           False  True  Total  Churn Rate
international plan voice mail plan
no                 no            1878   302   2180    0.138532
                   yes            786    44    830    0.053012
yes                no             130   101    231    0.437229
                   yes             56    36     92    0.391304
```

- We can see that customers with an international plan are more likely to churn than the ones without an international plan and the international plan variable seems to have more impact compared to the voice mail plan.

In [27]:
```
'''
From EDA we can confirm that the variables influencing churn are:
1. number vmail messages
2. total day minutes
3. total day charge
4. total eve minutes
5. total eve charge
6. total night minutes
7. total night charge
8. total intl minutes
9. total intl charge
10. customer service calls
11. international plan
12. voice mail plan
'''
```

Out[27]:
```
'\nFrom EDA we can confirm that the variables influencing churn are:\n1. number vmail messages\n2. total day minutes\n3. total day charge\n4. to
tal eve minutes\n5. total eve charge\n6. total night minutes\n7. total night charge\n8. total intl minutes\n9. total intl charge\n10. customer s
ervice calls\n11. international plan\n12. voice mail plan\n'
```

- Proceeding to the modeling.ipynb to further create a model that can predict churn.

Made by Khabeer Ahmed

- GitHub
- LinkedIn