Final Project Submission

Please fill out:

- Student name: Tia Plagata
- Student pace: full time
- Scheduled project review date/time:
- Instructor name: Rafael Carrasco
- Blog post URL:

Exploratory Data Analysis (Explore, and maybe more Scrubbing)

EDA Outline

- Is calling customer service a sign of customer unhappiness/potential churn?
- How much are people using their plan? What can this tell us about churn?
- Are customers in certain areas more likely to churn?
- Other feature engineering and exploration

In [1]:

```
# Import Statements
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')

import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set_style(style="darkgrid")
import plotly.express as px

from sklearn.model_selection import train_test_split
```

In [2]:

```
# Function for churn rate
def get_churn_rate(array, include_retention=False):
    """
    returns the percentage of customers who churned out of an array of churn
    if include_retention == True, also returns the customer retention rate
    """
    churns = sum(array)
    churn_rate = churns / len(array)
    if include_retention:
        return churn_rate, 1 - churn_rate
    else:
        return churn_rate
```

Split DF to Create Validation Set

In [3]:

```
df = pd.read_csv('/Users/jordanrjohnson/DataScienceCourseMaterial/phase_3/ds
df.head()
```

Out[3]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	tc e ca
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	

5 rows × 21 columns

In [60]:

```
df_train, df_validation = train_test_split(df, test_size=.1, random_state=1)
```

In [61]:

```
print(df_train.shape, df_validation.shape)
```

```
(2999, 21) (334, 21)
```

In [62]:

```
df_validation.to_csv('/Users/jordanrjohnson/DataScienceCourseMaterial/phase_
```

In [63]:

```
df_train.to_csv('/Users/jordanrjohnson/DataScienceCourseMaterial/phase_3/dsc
```

Validation set with 10% of our data is blind from everything we will do to model, but will be used to che quality of our model at the end. I split it and saved to a csv file for later.

Question 1: Is calling customer service a sign of customer unhappiness/potential churn?

In [15]:

```
# What is our current churn rate? Retention rate?
get_churn_rate(df_train['churn'], include_retention=True)
```

Out[15]:

```
(0.14338112704234746, 0.8566188729576525)
```

In [16]:

```
df_train.loc[(df_train['customer service calls'] > 1) & (df_train['churn'] =
```

Out[16]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...
3304	IL	71	510	330-7137	yes	no	0	186.1	114	31.64	...
2402	NY	77	415	388-9285	no	yes	33	143.0	101	24.31	...
1919	WA	100	408	382-4932	no	no	0	70.8	94	12.04	...
2980	KS	84	415	335-7144	no	no	0	225.9	86	38.40	...
3255	RI	138	510	411-6823	yes	no	0	286.2	61	48.65	...
...	...	...	...	...	...	...	...	...	...	...	...
319	SD	128	510	413-9269	yes	yes	32	223.5	81	38.00	...
3287	KS	170	415	404-5840	no	yes	42	199.5	119	33.92	...
144	VT	117	408	390-2390	yes	no	0	167.1	86	28.41	...
905	WV	161	415	418-9036	no	no	0	191.9	113	32.62	...
235	MN	139	510	374-9107	no	no	0	134.4	106	22.85	...

238 rows × 21 columns

In [17]:

```
df_train.loc[df_train['customer service calls'] > 1]
```

Out[17]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...
3304	IL	71	510	330-7137	yes	no	0	186.1	114	31.64	...
2402	NY	77	415	388-9285	no	yes	33	143.0	101	24.31	...
756	WY	33	415	331-3202	no	no	0	213.9	88	36.36	...
133	TX	82	408	400-3446	no	no	0	200.3	96	34.05	...
366	NC	112	415	334-1872	no	no	0	193.3	96	32.86	...
...	...	...	...	...	...	...	...	...	...	...	...
144	VT	117	408	390-2390	yes	no	0	167.1	86	28.41	...
960	AR	5	415	380-2758	no	no	0	199.2	106	33.86	...
2763	NC	116	408	338-7527	no	yes	19	155.7	104	26.47	...
905	WV	161	415	418-9036	no	no	0	191.9	113	32.62	...
235	MN	139	510	374-9107	no	no	0	134.4	106	22.85	...

1303 rows × 21 columns

In [18]:

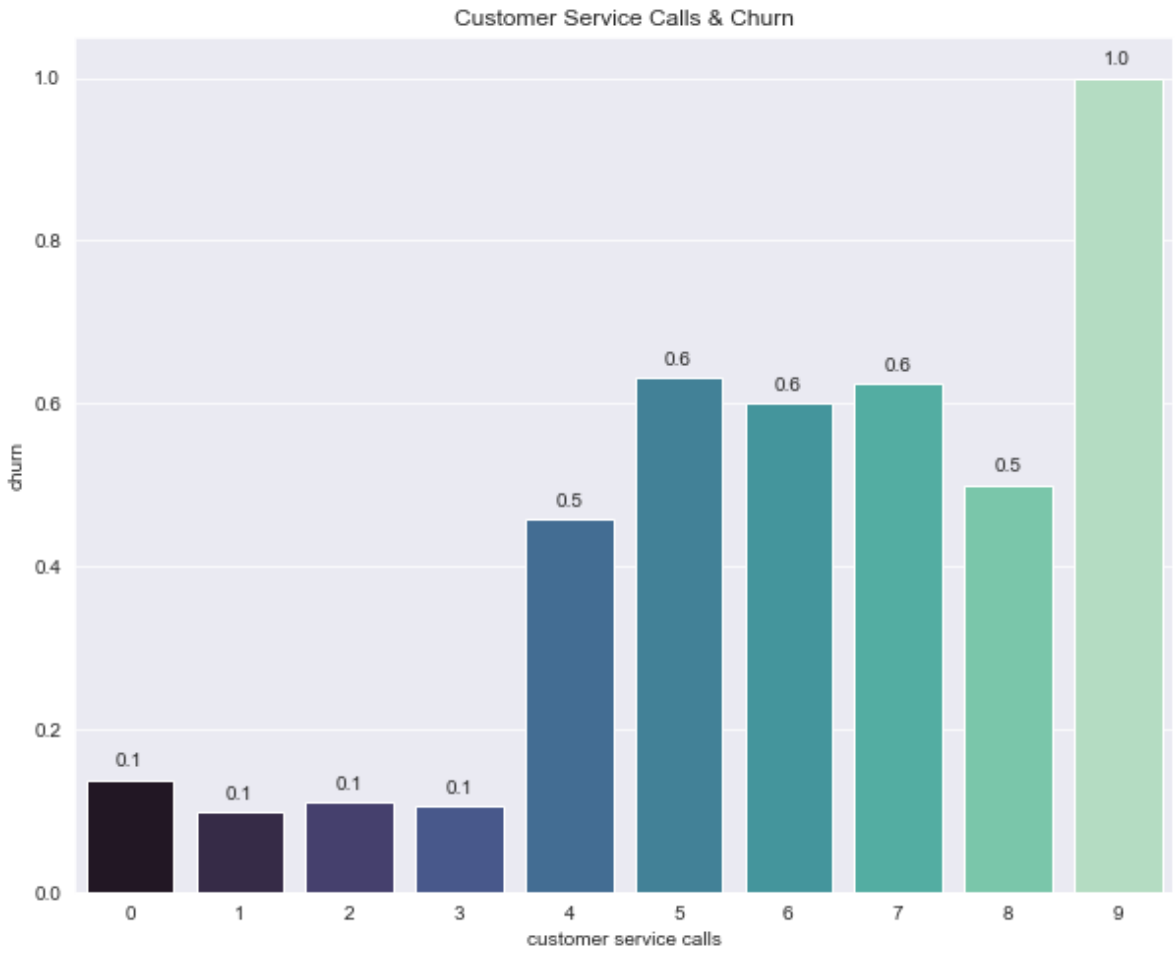
```
customer_service = df_train.groupby('customer service calls')['churn'].agg([
customer_service
```

Out[18]:

customer service calls	count
0	630
1	1066
2	676
3	392
4	146
5	57
6	20
7	8
8	2
9	2

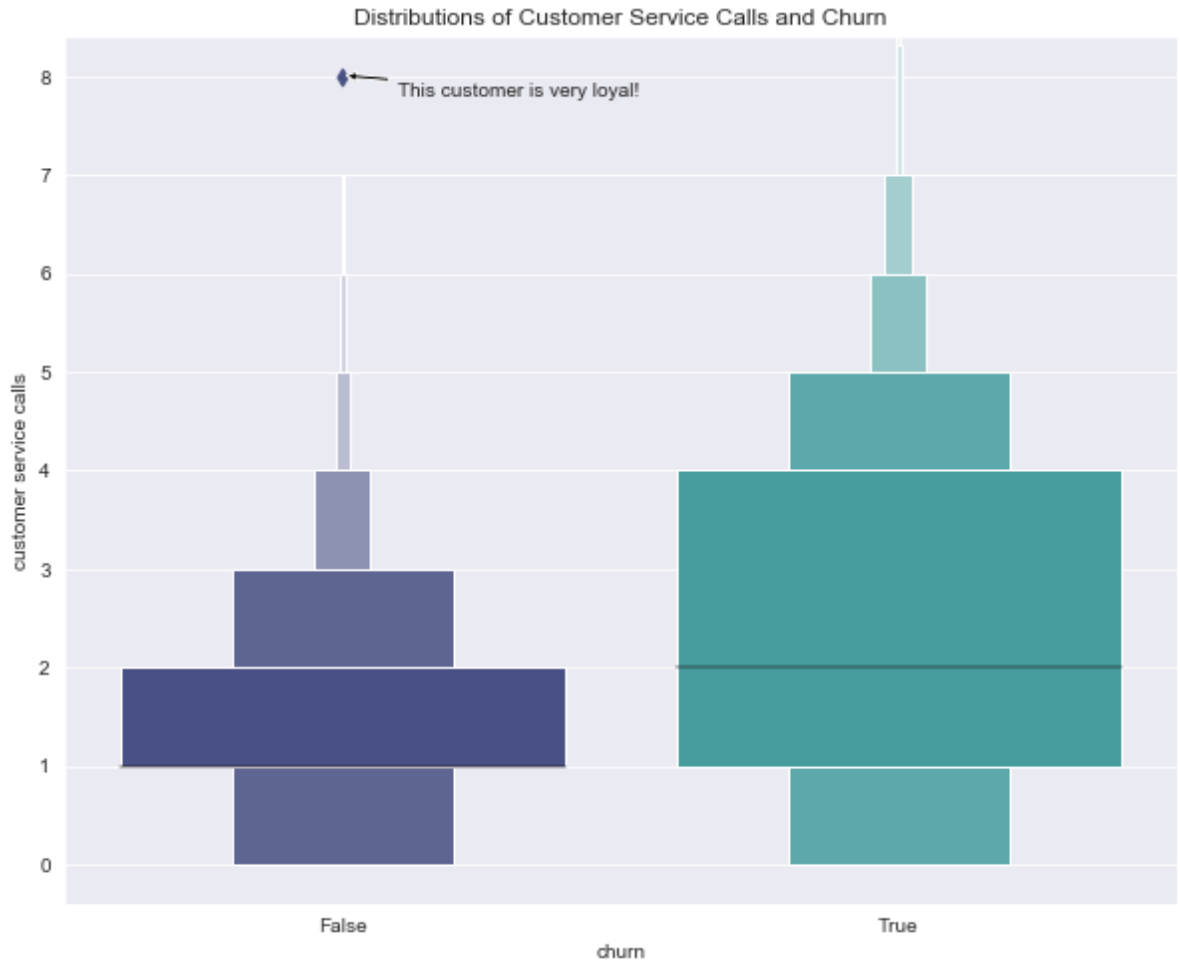
In [19]:

```
plt.figure(figsize=(10, 8))
splot = sns.barplot(x='customer service calls', y='churn',
                    data=df_train, palette='mako', ci=None)
# Add annotations to bars
for p in splot.patches:
    splot.annotate(format(p.get_height(), '.1f'),
                   (p.get_x() + p.get_width() / 2., p.get_height()),
                   ha = 'center', va = 'center',
                   xytext = (0, 9),
                   textcoords = 'offset points')
plt.title('Customer Service Calls & Churn')
plt.show()
```



In [24]:

```
plt.figure(figsize=(10, 8))
sns.boxenplot(x='churn', y='customer service calls',
              data=df_train, palette='mako')

plt.annotate('This customer is very loyal!', xy=(0, 8.01), xytext=(0.1, 7.8)
             arrowprops=dict(facecolor='black', arrowstyle='simple'))

plt.title('Distributions of Customer Service Calls and Churn')
plt.show()
# That loyal customer is probably an outlier so I will need to deal with him
```



Findings & Recommendations

Our current churn rate for the training data set is about 14.5%. When we look at customer service calls see that as the number of customer service calls increases, the *likelihood* of churning increases as wel Specifically, with at least 4 customer service calls, the likelihood of a customer churning increases from 10% to 50%.

Customer service calls alone cannot guarantee that a customer will churn. In fact, the majority of custo who DID NOT churn made 1-2 customer service calls. However, it is important to note that the majority who DID churn made 1-4 calls to customer service. Therefore, more than 3 calls to customer service sh a red flag that a customer is more likely to churn.

Recommendation:

Based on these findings, I would recommend revisiting our customer service protocol. It may be useful a larger incentive/discount to customers making more than 3 calls to customer service.

Question 2: How much are people using their plan? What can tell us about churn?

In [25]:

```
# First look at how much people use their regular plans
df_calls = df_train[['total day calls', 'total eve calls', 'total night call
df_calls.head()
```

Out[25]:

	total day calls	total eve calls	total night calls	churn
2682	77	100	92	False
3304	114	140	80	True
757	108	111	78	False
2402	101	102	120	True
792	98	102	85	True

In [26]:

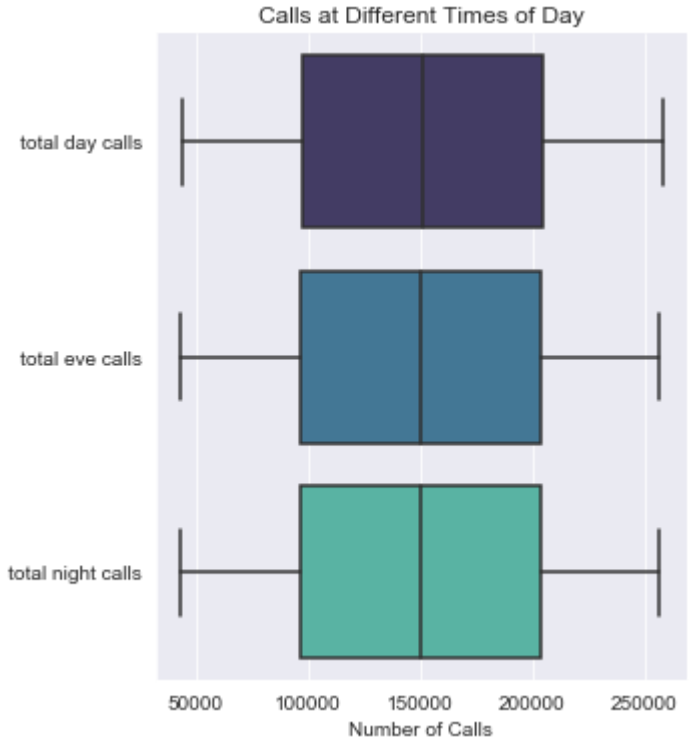
```
calls = df_calls.groupby('churn').sum()
calls.reset_index()
```

Out[26]:

	churn	total day calls	total eve calls	total night calls
0	False	257671	256525	256621
1	True	43432	43113	43039

In [27]:

```
sns.catplot(data=calls, orient="h", kind="box", palette='mako')
plt.title('Calls at Different Times of Day')
plt.xlabel('Number of Calls')
plt.show()
```



In [28]:

```
# What are the calling rates? (Checked mean and median and random ones... th
day_rate = (df_train['total day charge'] / df_train['total day minutes']).me
eve_rate = (df_train['total eve charge'] / df_train['total eve minutes']).me
night_rate = (df_train['total night charge'] / df_train['total night minutes
intl_rate = (df_train['total intl charge'] / df_train['total intl minutes'])

names = ['Day Rate', 'Eve Rate', 'Night Rate', 'Intl Rate']

for name, rate in zip(names, [day_rate, eve_rate, night_rate, intl_rate]):
    print(f'{name}:', round(rate, 2))
```

```
Day Rate: 0.17
Eve Rate: 0.08
Night Rate: 0.04
Intl Rate: 0.27
```

In [29]:

```
df_intl = df_train.loc[df_train['international plan'] == 'yes']
df_non_intl = df_train.loc[df_train['international plan'] == 'no']
```

In [30]:

```
intl_plan_rate = (df_intl['total intl charge'] / df_intl['total intl minutes
non_intl_plan_rate = (df_non_intl['total intl charge'] / df_non_intl['total
print(intl_plan_rate, non_intl_plan_rate)
# They are the same!
```

```
0.27 0.270063694267516
```

In [31]:

```
df_intl.head()
```

Out[31]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...
2682	DC	55	510	354-5058	yes	no	0	106.1	77	18.04	...
3304	IL	71	510	330-7137	yes	no	0	186.1	114	31.64	...
792	NV	69	510	397-6789	yes	yes	33	271.5	98	46.16	...
933	KY	74	510	368-7555	yes	no	0	125.8	103	21.39	...
1804	CT	125	415	409-7523	yes	no	0	187.3	118	31.84	...

5 rows × 21 columns

In [32]:

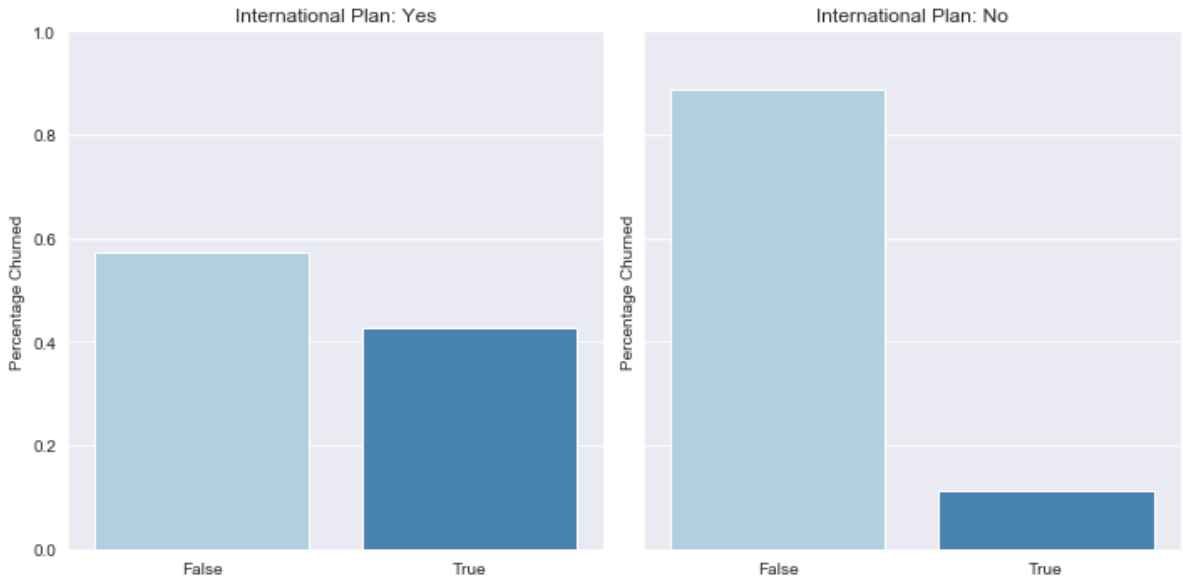
```
intl_churn = pd.DataFrame(df_intl['churn'].value_counts(normalize=True), col
non_intl_churn = pd.DataFrame(df_non_intl['churn'].value_counts(normalize=Tr

display(intl_churn)
display(non_intl_churn)
```

	churn
False	0.571918
True	0.428082

	churn
False	0.887329
True	0.112671

In [33]:

```
f, axes = plt.subplots(1, 2, figsize=(10, 5), sharey=True)
sns.barplot(x=['False', 'True'], y='churn', data=intl_churn, ax=axes[0], pal
sns.barplot(x=['False', 'True'], y='churn', data=non_intl_churn, ax=axes[1],
axes[0].set_title('International Plan: Yes')
axes[1].set_title('International Plan: No')
axes[0].set_ylabel('Percentage Churned')
axes[1].set_ylabel('Percentage Churned')
plt.ylim(0,1)
plt.tight_layout()
plt.show()
```



In [34]:

```
df_train.groupby('international plan')['total intl calls'].mean()
```

Out[34]:

```
international plan
no     4.482453
yes    4.623288
Name: total intl calls, dtype: float64
```

In [35]:

```
df_train.groupby('international plan')['total intl charge'].mean()
# Whether or not people have an international plan, they still average the s
```

Out[35]:

```
international plan
no     2.753170
yes    2.878459
Name: total intl charge, dtype: float64
```

In [36]:

```
# Create functions to create new features so we can use them later in a pipe

def create_total_calls_column(df):

    """
    creates a column of the total customer calls, excluding customer service
    returns the df with new column
    """

    df['total calls'] = df['total day calls'] + df['total eve calls'] + df['t
    return df

def create_minutes_per_intl_call(df):

    """
    creates a column of the average length (in minutes) per international ca
    returns the df with new column
    """

    df['avg minutes per intl call'] = df['total intl minutes'] / df['total in
    return df
```

In [37]:

```
# Create 2 new features on the training data set
df_train = create_minutes_per_intl_call(df_train)
df_train = create_total_calls_column(df_train)
```

In [38]:

```
df_train.head()
```

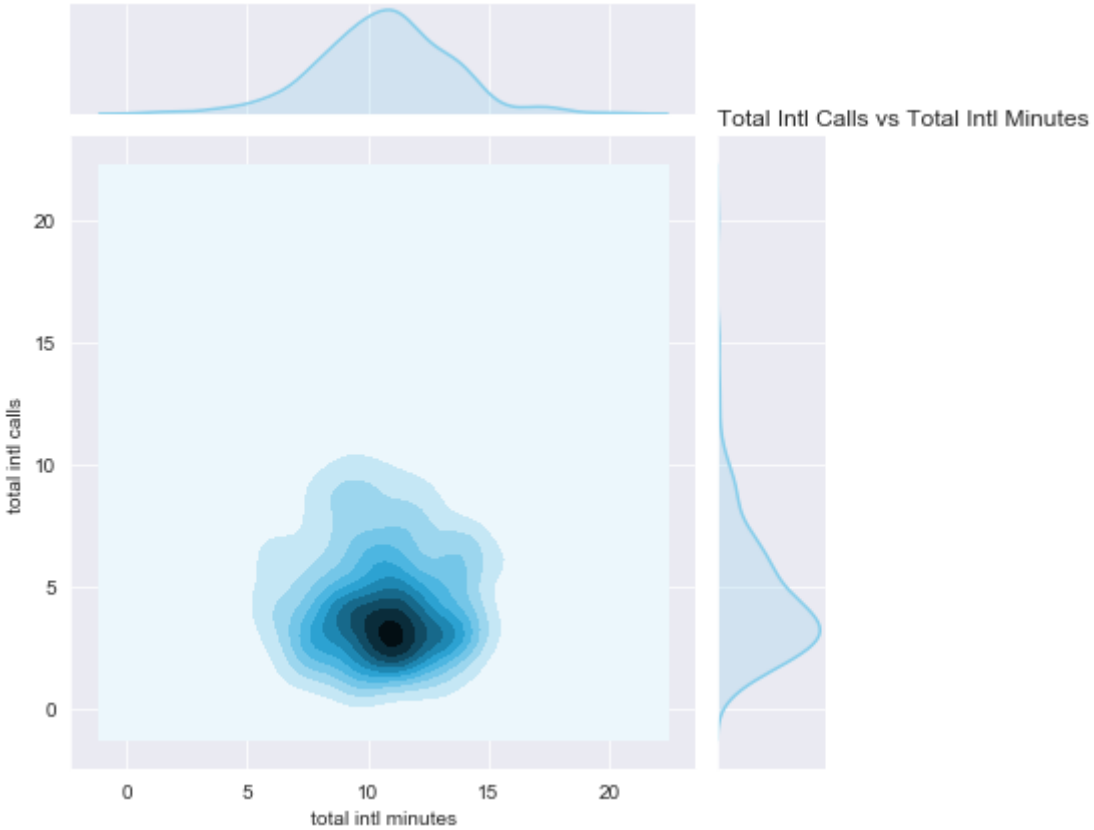
Out[38]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...
2682	DC	55	510	354-5058	yes	no	0	106.1	77	18.04	...
3304	IL	71	510	330-7137	yes	no	0	186.1	114	31.64	...
757	UT	112	415	358-5953	no	no	0	115.8	108	19.69	...
2402	NY	77	415	388-9285	no	yes	33	143.0	101	24.31	...
792	NV	69	510	397-6789	yes	yes	33	271.5	98	46.16	...

5 rows × 23 columns

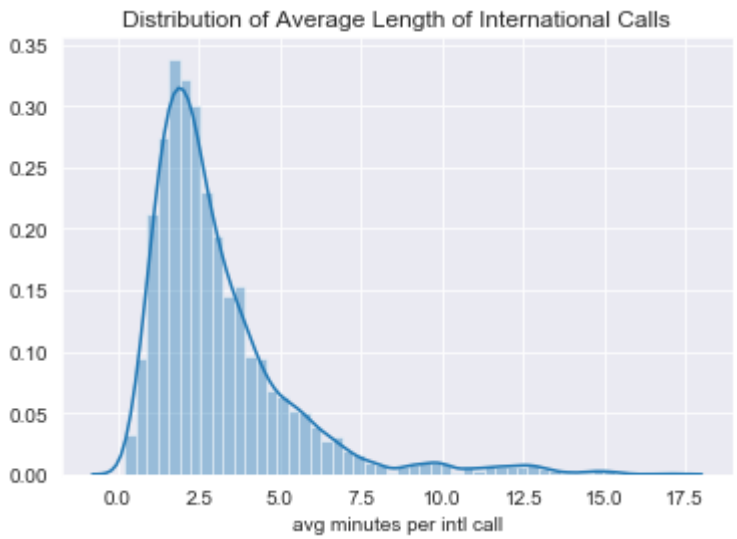
In [39]:

```
sns.jointplot(x=df_intl["total intl minutes"], y=df_intl["total intl calls"]
plt.title('Total Intl Calls vs Total Intl Minutes', loc='left')
plt.show()
# This tells us that most of the intl calls are relatively short (2-4 mins l
```



In [40]:

```
sns.distplot(df_train['avg minutes per intl call'])
plt.title('Distribution of Average Length of International Calls')
plt.show()
```



In [41]:

```
stacked_bar_data = df_train.groupby('churn').sum().reset_index()
```

In [42]:

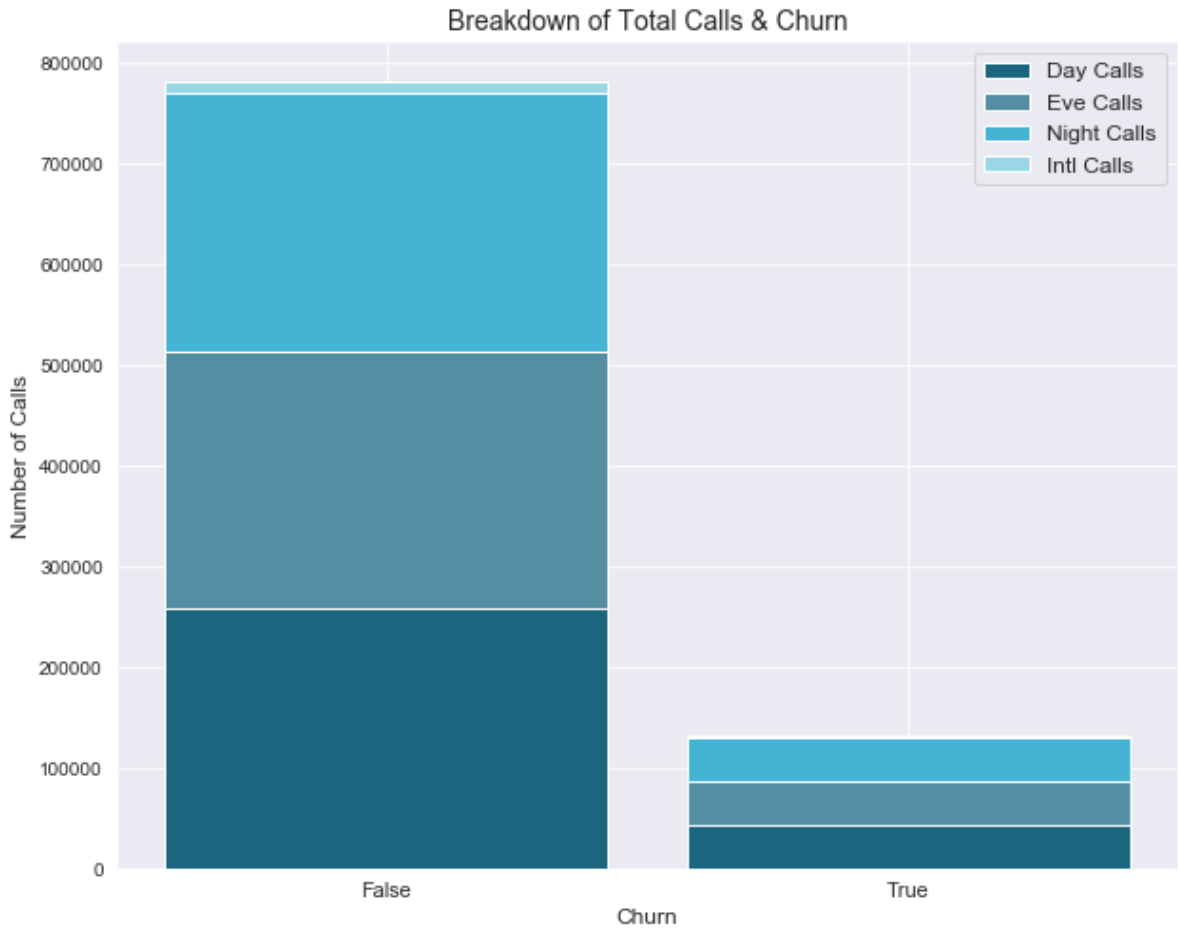
```
stacked_bar_data
```

Out[42]:

	churn	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
0	False	258560	1122642	22015	449548.1	257671	76424.54	512365.8	256525	43551.72
1	True	43692	188415	2194	88889.3	43432	15111.40	91290.1	43113	7759.73

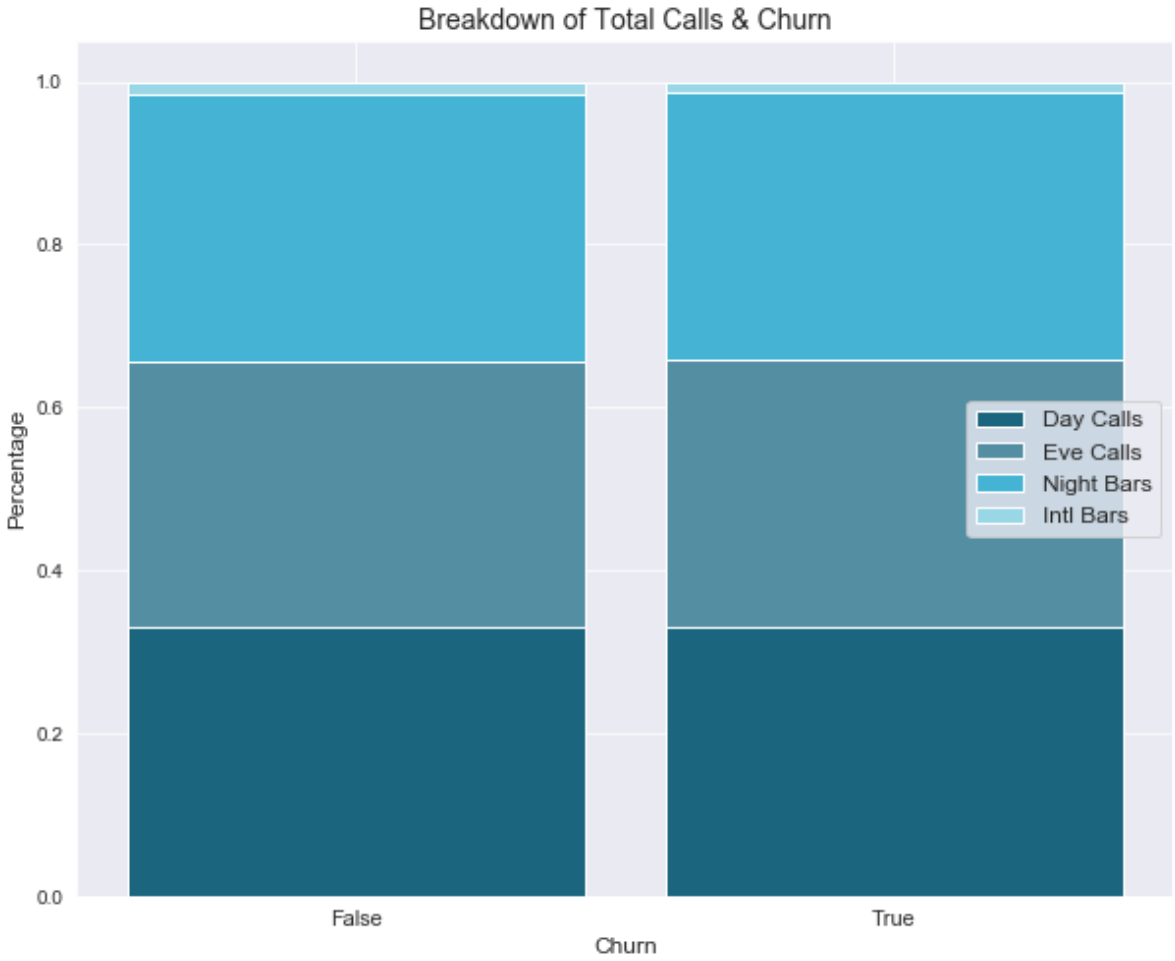
In [43]:

```
# Create an awesome stacked bar graph of all calls
r = stacked_bar_data['churn']

# Values
totals = stacked_bar_data['total calls']
DayBars = stacked_bar_data['total day calls']
EveBars = stacked_bar_data['total eve calls']
NightBars = stacked_bar_data['total night calls']
IntlBars = stacked_bar_data['total intl calls']

# Plot
plt.figure(figsize=(10,8))
names = ('False', 'True')
barWidth = 0.85

# Create Day Bars
plt.bar(r, DayBars, color='#1b667e', edgecolor='white',
        width=barWidth, label='Day Calls')

# Create Eve Bars
plt.bar(r, EveBars, bottom=DayBars, color='#548ea3',
        edgecolor='white', width=barWidth, label='Eve Calls')

# Create Night Bars
plt.bar(r, NightBars, bottom=[i+j for i,j in zip(DayBars, EveBars)],
        color='#45b4d4', edgecolor='white', width=barWidth, label='Night Cal

# Create Intl Bars
plt.bar(r, IntlBars, bottom=[i+j+k for i,j,k in zip(DayBars, EveBars, NightB
        color='#99d6e6', edgecolor='white', width=barWidth, label='Intl Call

# Graph details
plt.xticks(r, names, fontsize=11)
plt.xlabel('Churn', fontsize=12)
plt.ylabel('Number of Calls', fontsize=12)
plt.title('Breakdown of Total Calls & Churn', fontsize=14)
plt.legend(loc=1, fontsize='large')

# Show graph
plt.show()
```


Breakdown of Total Calls & Churn
Day Calls
Eve Calls
Night Calls
Intl Calls
Number of Calls
800000
700000
600000
500000
400000
300000
200000
100000
0
False
True
Churn

In [44]:

```
# Create an awesome stacked bar graph of all calls
r = stacked_bar_data['churn']

# Turn values to percentage
totals = stacked_bar_data['total calls']
DayBars = stacked_bar_data['total day calls'] / totals
EveBars = stacked_bar_data['total eve calls'] / totals
NightBars = stacked_bar_data['total night calls'] / totals
IntlBars = stacked_bar_data['total intl calls'] / totals

# Plot
plt.figure(figsize=(10,8))
names = ('False', 'True')
barWidth = 0.85

# Create Day Bars
plt.bar(r, DayBars, color='#1b667e', edgecolor='white',
        width=barWidth, label='Day Calls')

# Create Eve Bars
plt.bar(r, EveBars, bottom=DayBars, color='#548ea3',
        edgecolor='white', width=barWidth, label='Eve Calls')

# Create Night Bars
plt.bar(r, NightBars, bottom=[i+j for i,j in zip(DayBars, EveBars)],
        color='#45b4d4', edgecolor='white', width=barWidth, label='Night Bar

# Create Intl Bars
plt.bar(r, IntlBars, bottom=[i+j+k for i,j,k in zip(DayBars, EveBars, NightB
        color='#99d6e6', edgecolor='white', width=barWidth, label='Intl Bars

# Graph details
plt.xticks(r, names, fontsize=11)
plt.xlabel('Churn', fontsize=12)
plt.ylabel('Percentage', fontsize=12)
plt.title('Breakdown of Total Calls & Churn', fontsize=14)
plt.legend(loc=7, fontsize='large')

# Show graph
plt.show()
```



Findings & Recommendations

It is clear that the customers who churned and those that did not churn had almost exactly the same u across day, eve, night and international calls. The rates for international minutes are the same regardles whether the customer has an international plan or not (27 cents per minute). It is also interesting to not the percentage of customers who churned was higher for customers with international plans than for customers without international plans. Because of this similar charge for international calls, it is possib the customers who had an international plan and churned did not feel that paying for the international worth it.

Recommendations:

Based on these findings, I recommend changing the rates for international minutes. If a customer has a international plan, they should have cheaper rates for international calls than a customer without an international plan.

Question 3: Are customers in certain areas more likely to chu

In [46]:

```
# Obviously this doesn't tell us much if there are only 3 area codes across
# We are better off using states
display(df_train['state'].unique())
display(df_train['area code'].unique())
```

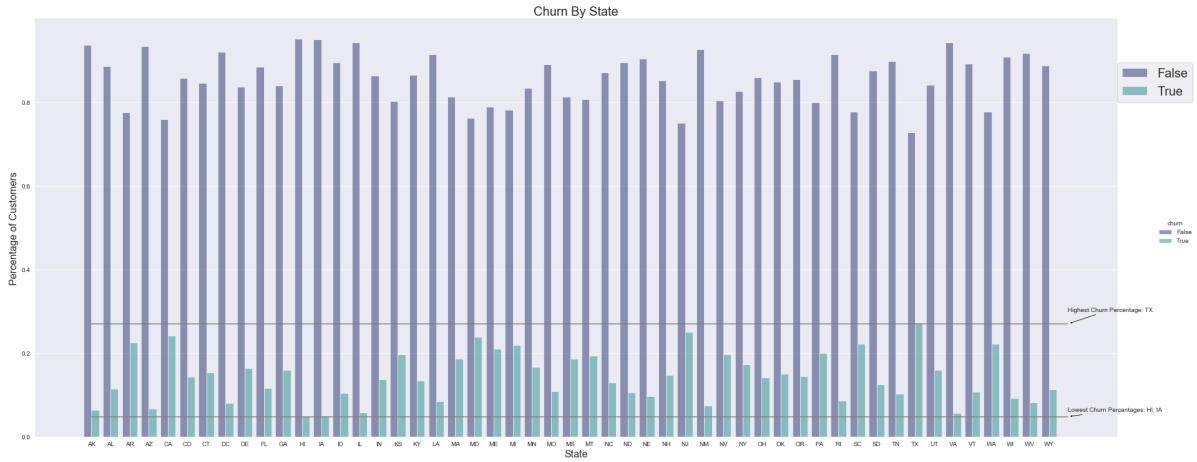
```
array(['DC', 'IL', 'UT', 'NY', 'NV', 'KY', 'WY', 'MT', 'NE', 'CT', 'M
O',
       'SC', 'DE', 'CO', 'IN', 'NM', 'TX', 'FL', 'ND', 'AL', 'OK', 'N
C',
       'MA', 'VA', 'VT', 'MD', 'KS', 'MN', 'WA', 'AZ', 'IA', 'AK', 'M
I',
       'OR', 'WV', 'GA', 'MS', 'OH', 'ID', 'WI', 'SD', 'HI', 'LA', 'M
E',
       'RI', 'NJ', 'AR', 'NH', 'CA', 'TN', 'PA'], dtype=object)


array([510, 415, 408])
```

In [47]:

```
churn_by_state = df_train.groupby('state')['churn'].value_counts(normalize=T
churn_by_state = pd.DataFrame(churn_by_state)
churn_by_state.columns = ['value']
churn_by_state = churn_by_state.reset_index()
```

In [54]:

```
# Plot
sns.catplot(data=churn_by_state, kind='bar',
            x='state', y='value', hue='churn',
            palette='mako', alpha=.6, height=10, aspect=2.5)

# Graph details
plt.title('Churn By State', fontsize=20)
plt.ylabel('Percentage of Customers', fontsize=16)
plt.xlabel('State', fontsize=16)
plt.legend(loc=(1, .8), fontsize=20)

# hlines on highest and lowest churns
plt.hlines(y=.27, xmin=0, xmax=51, color='gray')
plt.hlines(y=.048, xmin=0, xmax=51, color='gray')

# annotate highest and lowest churns
plt.annotate('Highest Churn Percentage: TX', xy=(51, .27), xytext=(51, .3),
            arrowprops=dict(facecolor='black', arrowstyle='simple'))
plt.annotate('Lowest Churn Percentages: HI, IA', xy=(51, .048), xytext=(51,
            arrowprops=dict(facecolor='black', arrowstyle='simple'))

plt.show()
```



In [51]:

```
churn_by_state.loc[(churn_by_state['value'] > .23) & (churn_by_state['churn'
```

Out[51]:

	state	churn	value
9	CA	True	0.241379
41	MD	True	0.238095
63	NJ	True	0.250000
87	TX	True	0.272727

In [52]:

```
churn_by_state.loc[churn_by_state['value'] <= .05]
```

Out[52]:

	state	churn	value
23	HI	True	0.04878
25	IA	True	0.05000

In [56]:

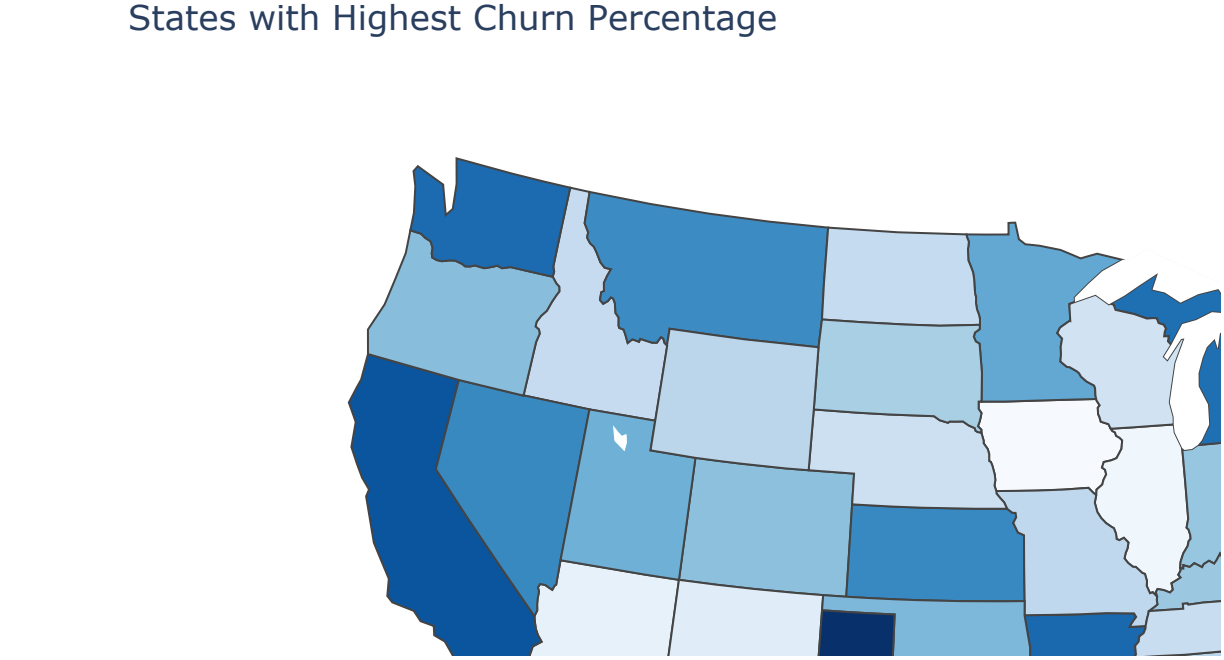
```
lowest_churn_percent = churn_by_state.loc[churn_by_state['value'] <= .05]
highest_churn_percent = churn_by_state.loc[(churn_by_state['value'] > .23) &
```

In [57]:

```
churn_choropleth = churn_by_state.loc[churn_by_state['churn']==True]
```

In [58]:

```
fig = px.choropleth(data_frame=churn_choropleth, locations='state', location
                    color='value', scope="usa", title='States with Highest C
                    color_continuous_scale='Blues')
fig.show()
```



In [94]:

```
high_competition = churn_by_state.loc[(churn_by_state['value'] >= .2) & (chu
high_competition['state'].unique()
```

Out[94]:

```
array(['AR', 'CA', 'MD', 'ME', 'MI', 'NJ', 'PA', 'SC', 'TX', 'WA'],
      dtype=object)
```

In [91]:

```
med_high_competition = churn_by_state.loc[(churn_by_state['value'] >= .15) &
med_high_competition['state'].unique()
```

Out[91]:

```
array(['CT', 'DE', 'GA', 'KS', 'MA', 'MN', 'MS', 'MT', 'NV', 'NY', 'O
K',
       'UT'], dtype=object)
```

In [93]:

```
medium_competition = churn_by_state.loc[(churn_by_state['value'] < .15) & (c
medium_competition['state'].unique()
```

Out[93]:

```
array(['AL', 'CO', 'FL', 'ID', 'IN', 'KY', 'MO', 'NC', 'ND', 'NH', 'O
H',
       'OR', 'SD', 'TN', 'VT', 'WY'], dtype=object)
```

In [82]:

```
low_competition = churn_by_state.loc[(churn_by_state['value'] < .1) & (churn
low_competition['state'].unique()
```

Out[82]:

```
array(['AK', 'AZ', 'DC', 'HI', 'IA', 'IL', 'LA', 'NE', 'NM', 'RI', 'V
A',
       'WI', 'WV'], dtype=object)
```

In [98]:

```
def categorize_state(state):

    """
    Encodes states in terms of how much competition is present in that marke
    Returns each encoded state.
    """

    if state in ['AK', 'AZ', 'DC', 'HI', 'IA', 'IL', 'LA', 'NE', 'NM', 'RI',
        state = 1
    elif state in ['AL', 'CO', 'FL', 'ID', 'IN', 'KY', 'MO', 'NC', 'ND', 'NH
        state = 2
    elif state in ['CT', 'DE', 'GA', 'KS', 'MA', 'MN', 'MS', 'MT', 'NV', 'NY
        state = 3
    else:
        state = 4
    return state


def create_competition_feat(df):

    """
    Creates the competition feature, which encodes each state in regards to
    Returns entire df with new feature.
    """

    df['competiton'] = df['state'].apply(categorize_state)
    return df
```

In [97]:

```
create_competition_feat(df_train)

df_train.head(20)
```

Out[97]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	to e char
2682	DC	55	510	354-5058	yes	no	0	106.1	77	18.04	...	10.
3304	IL	71	510	330-7137	yes	no	0	186.1	114	31.64	...	16.
757	UT	112	415	358-5953	no	no	0	115.8	108	19.69	...	20.
2402	NY	77	415	388-9285	no	yes	33	143.0	101	24.31	...	18.
792	NV	69	510	397-6789	yes	yes	33	271.5	98	46.16	...	21.
933	KY	74	510	368-7555	yes	no	0	125.8	103	21.39	...	17.

Findings & Recommendations

It is clear that there are certain states with much higher churn. When grouped by state, Texas has a mu higher churn than any other state (27%). New Jersey, Maryland and California also have higher churn (23%). States with the least churn include Hawaii and Iowa (under .05%).

There could be a few reasons for this difference in churn in different states. One reason could be the la competitors in places like Hawaii and Iowa, which are more remote. States such as California, New Jer Texas could have many other big players in the market, which causes our customers to have other opt when they feel inclined to leave. Another reason could be the lack of good service in certain areas in st with high churn.

Recommendations

Based on these findings, I would recommend looking into competitors in Texas, California, New Jersey other states with high churn to see if they are offering introductory offers that might compel some of ou customers to churn. I also recommend looking into the cell signal in these states with higher churn to s there are any deadzones contributing to the higher rates.

Conclusion and Future Work

In conclusion, calls to customer service seems to be one of the biggest indicators of customer churn. \ also see higher churn in certain states, although the reasons why specific states are more likely to chur unclear based on this data. We can also see that customers may not be happy with their international p which is why customers with an international plan are more likely to churn than customers without an international plan.

Future Work I would like to do:

- Get more data regarding competitors in states with higher churn
- Get more data on cell signal across the US to look for patterns in states with higher churn
- Look into voicemail data to see if that may be a good indicator of churn

In [ ]: