Garretthall27 changed name to data_cleaning notebook and continued modeling no... History

1 contributor

2.71 MB

Data Cleaning

In [1]:

```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
```

In [2]:

```
# Load dataset
df = pd.read_csv('./Data/syriatel_data.csv')
```

In [3]:

```
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	...
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 [4]:

```
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 [5]:

```
df.iloc[:, 0:10]
```

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
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
...	...	...	...	...	...	...	...	...	...	...
3328	AZ	192	415	414-4276	no	yes	36	156.2	77	26.55
3329	WV	68	415	370-3271	no	no	0	231.1	57	39.29
3330	RI	28	510	328-8230	no	no	0	180.8	109	30.74
3331	CT	184	510	364-6381	yes	no	0	213.8	105	36.35
3332	TN	74	415	400-4344	no	yes	25	234.4	113	39.85

3333 rows × 10 columns

In [6]:

```
df.iloc[:, 10:]
```

Out[6]:

	total eve minutes	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	customer service calls	chu

0	197.4	99	16.78	244.7	91	11.01	10.0	3	2.70	1	Fa
1	195.5	103	16.62	254.4	103	11.45	13.7	3	3.70	1	Fa
2	121.2	110	10.30	162.6	104	7.32	12.2	5	3.29	0	Fa
3	61.9	88	5.26	196.9	89	8.86	6.6	7	1.78	2	Fa
4	148.3	122	12.61	186.9	121	8.41	10.1	3	2.73	3	Fa
...	...	...	...	...	...	...	...	...	...	...	
3328	215.5	126	18.32	279.1	83	12.56	9.9	6	2.67	2	Fa
3329	153.4	55	13.04	191.3	123	8.61	9.6	4	2.59	3	Fa
3330	288.8	58	24.55	191.9	91	8.64	14.1	6	3.81	2	Fa
3331	159.6	84	13.57	139.2	137	6.26	5.0	10	1.35	2	Fa
3332	265.9	82	22.60	241.4	77	10.86	13.7	4	3.70	0	Fa

3333 rows × 11 columns

In [7]:

```
sorted(df['state'].unique())
```

Out[7]:

```
['AK',
 'AL',
 'AR',
 'AZ',
 'CA',
 'CO',
 'CT',
 'DC',
 'DE',
 'FL',
 'GA',
 'HI',
 'IA',
 'ID',
 'IL',
 'IN',
 'KS',
 'KY',
 'LA',
 'MA',
 'MD',
 'ME',
 'MI',
 'MN',
 'MO',
 'MS',
 'MT',
 'NC',
 'ND',
 'NE',
 'NH',
 'NJ',
 'NM',
```

```
'NV',
'NY',
'OH',
'OK',
'OR',
'PA',
'RI',
'SC',
'SD',
'TN',
'TX',
'UT',
'VA',
'VT',
'WA',
'WI',
'WV',
'WY']
```

51 states uncluding DC

In [8]:

```
df['phone number'].value_counts()
```

Out[8]:

```
361-5936    1
346-7302    1
370-9533    1
345-2448    1
382-4872    1
           ..
401-5915    1
379-5933    1
403-6225    1
331-7425    1
334-6142    1
Name: phone number, Length: 3333, dtype: int64
```

Each phone number is unique in the dataset. Used as identifier.

In [9]:

```
df['international plan'].value_counts()
```

Out[9]:

```
no     3010
yes     323
Name: international plan, dtype: int64
```

In [10]:

```
df['voice mail plan'].value_counts()
```

Out[10]:

```
no     2411
yes     922
Name: voice mail plan, dtype: int64
```

Mapping 'international plan' and 'voice mail plan' as 1 or 0

In [11]:

```
# dictionary for mapping
yes_no_dict = {
    'yes': 1,
    'no': 0
```

```
}
```

In [12]:

```
# changing international plan to 0 and 1
df['international plan'] = df['international plan'].map(yes_no_dict)
```

In [13]:

```
# changing voice mail plan to 0 and 1
df['voice mail plan'] = df['voice mail plan'].map(yes_no_dict)
```

In [14]:

```
df
```

Out[14]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge
0	KS	128	415	382-4657	0	1	25	265.1	110	45.07
1	OH	107	415	371-7191	0	1	26	161.6	123	27.47
2	NJ	137	415	358-1921	0	0	0	243.4	114	41.38
3	OH	84	408	375-9999	1	0	0	299.4	71	50.90
4	OK	75	415	330-6626	1	0	0	166.7	113	28.34
...	...	...	...	...	...	...	...	...	...	...
3328	AZ	192	415	414-4276	0	1	36	156.2	77	26.55
3329	WV	68	415	370-3271	0	0	0	231.1	57	39.29
3330	RI	28	510	328-8230	0	0	0	180.8	109	30.74
3331	CT	184	510	364-6381	1	0	0	213.8	105	36.35
3332	TN	74	415	400-4344	0	1	25	234.4	113	39.85

3333 rows × 21 columns

Looking at churn column

In [15]:

```
df['churn'].value_counts(normalize=True)
```

Out[15]:

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

```
Name: churn, dtype: float64
```

True column is underrepresented I will have to use a method to increase the minority class when modeling.

Changing churn column to int type

In [16]:
```python
df['churn'] = df['churn'].astype(int)
```

In [17]:
```python
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   int64  
 5   voice mail plan         3333 non-null   int64  
 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   int32  
dtypes: float64(8), int32(1), int64(10), object(2)
memory usage: 533.9+ KB
```

In [18]:
```python
df.duplicated().sum()
```

Out[18]:
```
0
```

No duplicate rows in the data frame

Data Analysis

In [77]:
```python
df.head()
```

Out[77]:

	state	account	international	voice mail	number vmail	total day	total day	total day	total eve	total eve	...

		length	plan	plan	messages	minutes	calls	charge	minutes	calls	
0	KS	128	0	1	25	265.1	110	45.07	197.4	99	...
1	OH	107	0	1	26	161.6	123	27.47	195.5	103	...
2	NJ	137	0	0	0	243.4	114	41.38	121.2	110	...
3	OH	84	1	0	0	299.4	71	50.90	61.9	88	...
4	OK	75	1	0	0	166.7	113	28.34	148.3	122	...

5 rows × 22 columns

In [20]:

```
df.describe()
```

Out[20]:

	account length	area code	international plan	voice mail plan	number vmail messages	total day minutes	tota
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.00
mean	101.064806	437.182418	0.096910	0.276628	8.099010	179.775098	100.43
std	39.822106	42.371290	0.295879	0.447398	13.688365	54.467389	20.06
min	1.000000	408.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	74.000000	408.000000	0.000000	0.000000	0.000000	143.700000	87.00
50%	101.000000	415.000000	0.000000	0.000000	0.000000	179.400000	101.00
75%	127.000000	510.000000	0.000000	1.000000	20.000000	216.400000	114.00
max	243.000000	510.000000	1.000000	1.000000	51.000000	350.800000	165.00

Looking at the table above I can see the following information:

- Account length is how long the customer has been with them. I assuming it is in days.
- Total charge seems to be how much the customer was charged for those particular minutes whether it was day, evening, night, or international.
- Average international minutes is far lower than evening, night, or day which makes sense.
- Min customer service calls and max customer services calls are 0 and 9 respectively. Average being about 1.5.
- Average amount of calls is about the same between day, evening, and night. The average minutes for night and evening are about the same but both higher than during the day. People talk on the phone longer in the evening and night than during the day.

Now I want to see the distributions of the continuous variable columns and relations to churn

churn.

In [21]:

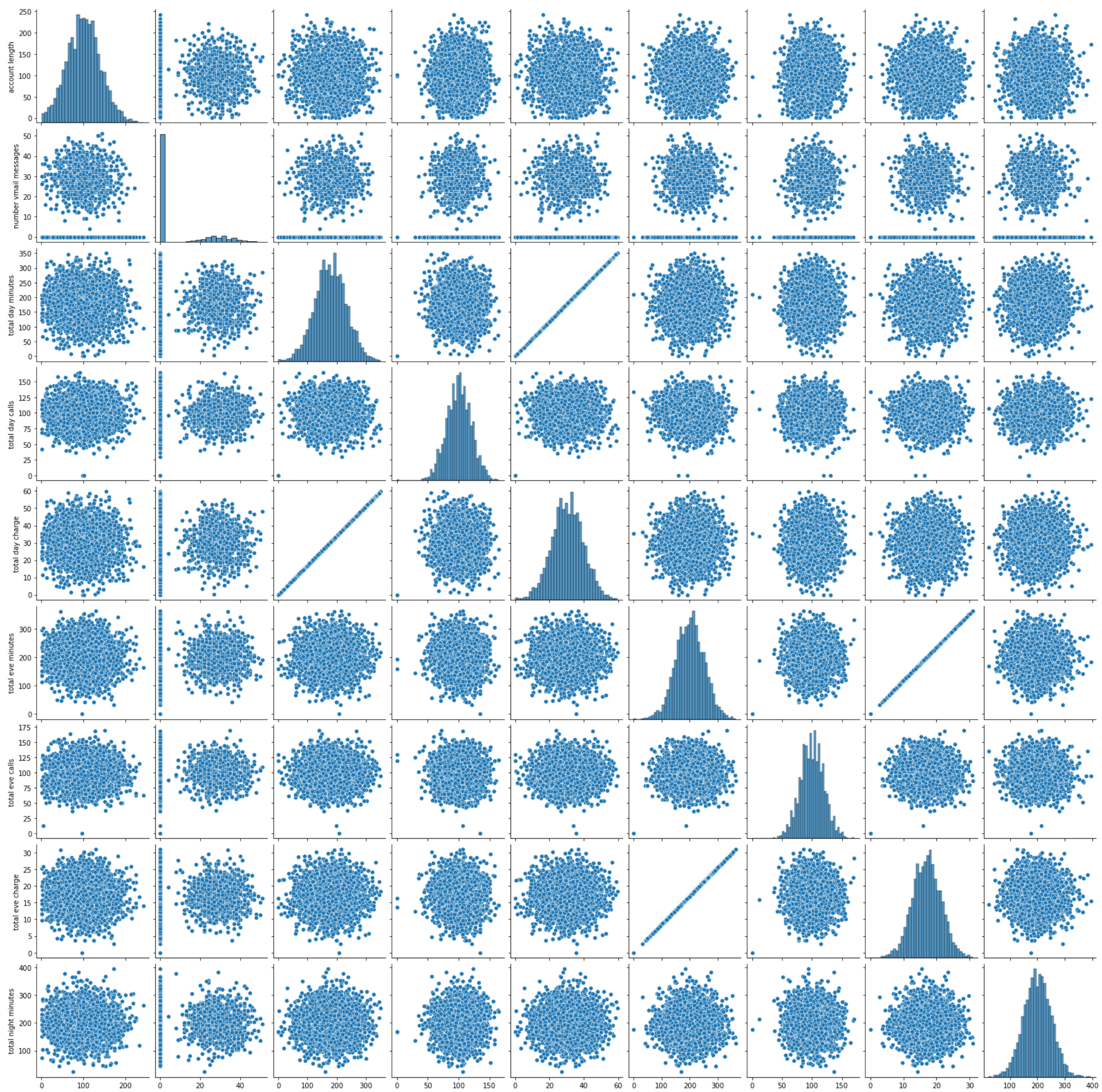
```
cont_cols = ['account length', 'number vmail messages', 'total day minutes',
             'total day calls', 'total day charge', 'total eve minutes',
             'total eve calls', 'total eve charge', 'total night minutes',
             'total night calls', 'total night charge', 'total intl minutes',
             'total intl calls', 'total intl charge', 'customer service calls',
             'international plan', 'voice mail plan', 'churn']
```

In [22]:

```
# pair plot for first 9 of cont_cols
sns.pairplot(data=df[cont_cols[0:9]])
```

Out[22]:

```
<seaborn.axisgrid.PairGrid at 0x1b2e2f30190>
```



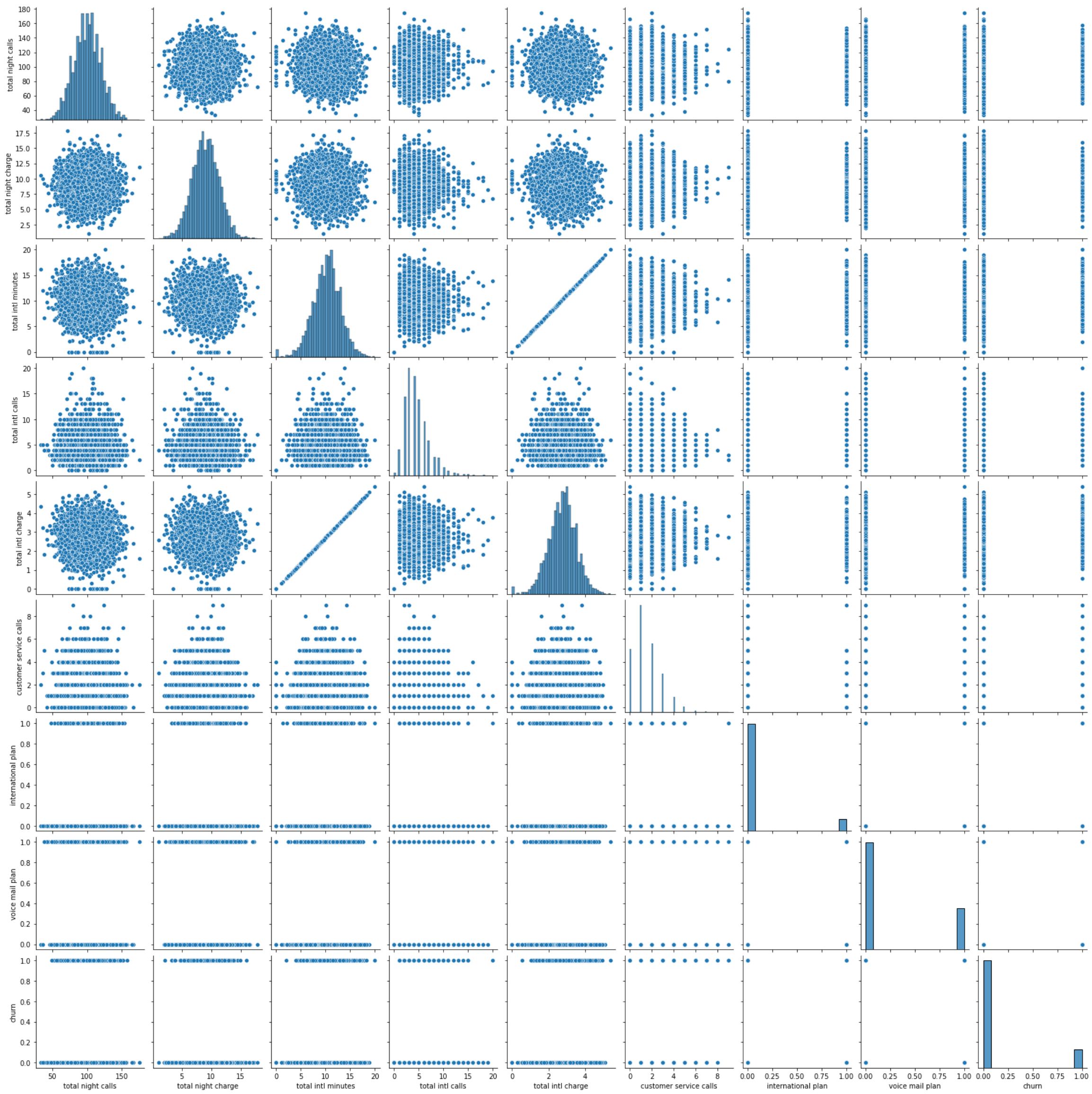
From this pair plot above I can see the following info:

- All the continous variables are normally distributed except for number of vmail messages which has a lot of 0 values. During modeling I may be able to create a new column that will have a boolean value of whether the customer left a voicemail message.

- There seems to be no clear linear relationships between variables except for minutes and charges but that is to be expected because customers are charged a rate by the minute.

In [23]:
```python
# pair plot for last 9 of cont_cols
sns.pairplot(data=df[cont_cols[9:]])
```

Out[23]:
```
<seaborn.axisgrid.PairGrid at 0x1b2c82221f0>
```

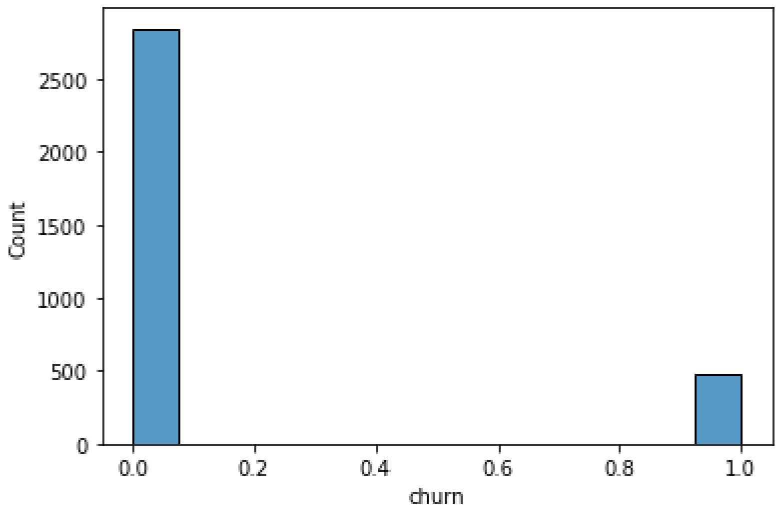


From this pair plot above I can see the following info:

- All the continous variables are normally distributed except for total intl calls and customer service calls which both look to be skewed right..
- There seems to be no clear linear relationships between variables except for minutes and charges but that is to be expected because customers are charged a rate by the minute.

In [24]:
```python
sns.histplot(data=df['churn'])
```

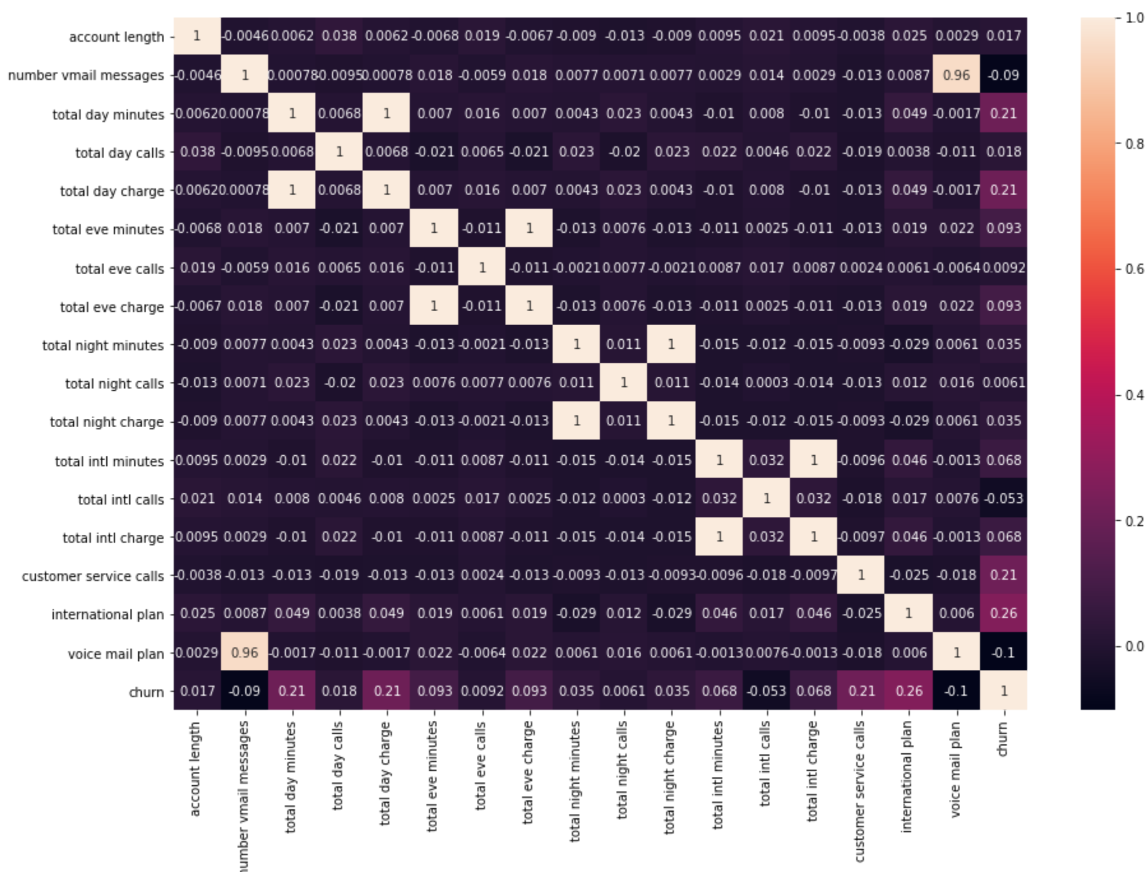
Out[24]:
```
<AxesSubplot:xlabel='churn', ylabel='Count'>
```



Now I want to take a look at the correlations of the columns

In [25]:
```python
# correlation heatmap
fig, ax = plt.subplots(figsize=(15,10))
sns.heatmap(df[cont_cols].corr(), ax=ax, annot=True)
```

Out[25]:
```
<AxesSubplot:>
```



As you can see here, there is not much of any strong correlations between the variables. Highest correlations are between minutes and charges and that is because charges is calculated from minutes.

There are some positive correlations between churn and total day minutes, total day charge, international plan, and customer service calls. However, these correlations are rather weak.

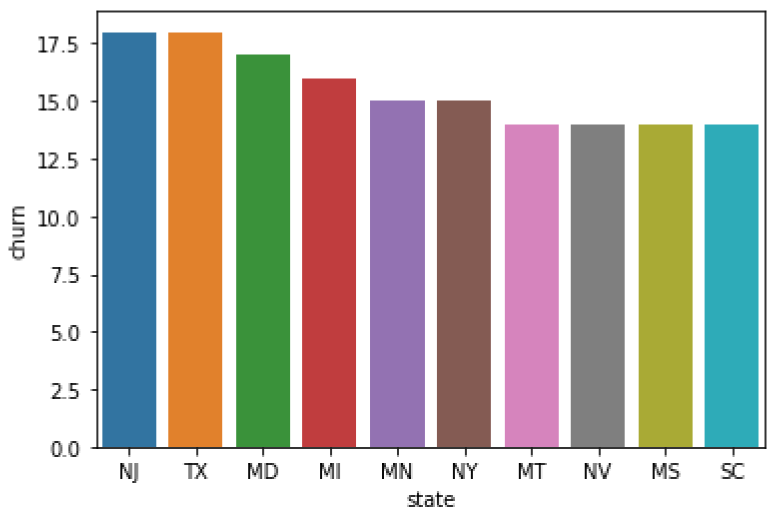
It is possible people with international plans churn more because they are unhappy with the international service provided.

Now I'll take a look at the states with the most churns.

In [26]:

```
# creating new dataframe for the visual
df_state = df[['state', 'churn']].groupby('state').sum()\
            .sort_values('churn', ascending=False).reset_index()

# bar chart of top 10 states by total churn
sns.barplot(data=df_state.iloc[0:10], x='state',y='churn')
```

Out[26]: <AxesSubplot:xlabel='state', ylabel='churn'>



The top 10 states range from 14 churns to 18 churns.

Now I want to take a look at the average of the continuous variables by churn.

In [27]:

```
# group by churn and average
df[cont_cols].groupby('churn').mean()
```

Out[27]:

	account length	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	tota ch
churn								
0	100.793684	8.604561	175.175754	100.283158	29.780421	199.043298	100.038596	16.91
1	102.664596	5.115942	206.914079	101.335404	35.175921	212.410145	100.561077	18.05

Key takeaways from this:

- Customers who have churned on average have fewer voicemail messages, more total day minutes, more total eve minutes, and more customer service calls.

In [28]:
```
df.churn.value_counts()
```

Out[28]:
```
0    2850
1     483
Name: churn, dtype: int64
```

Hypothesis Testing

I want to perform hypothesis testing to see if there is any significant difference between churn and non-churned customers when it comes to average voicemail messages, total day minutes, total eve minutes, and customer services calls. This will help me determine if they are significant patterns between customers who have churned or not.

In [29]:
```
# setting up hypothesis test using t test
# voice mail messages
filter0 = df['churn'] == 0
filter1 = df['churn'] == 1

# filter dataframes
vm_churn_0 = df.loc[filter0]['number vmail messages']
vm_churn_1 = df.loc[filter1]['number vmail messages']

# t test
alpha = 0.05
print('Alpha:', alpha)

p_value = stats.ttest_ind(vm_churn_0, vm_churn_1).pvalue / 2
print('P-value:', p_value)
if p_value < alpha:
    print('Reject null hypothesis')
else:
    print('Failed to reject null hypothesis')
```

```
Alpha: 0.05
P-value: 1.0587609201356013e-07
Reject null hypothesis
```

For average voicemails, I can reject the null hypothesis that average voicemails for churned customers is not lower than nonchurned customers.

In [30]:
```
# create list for remaining columns for hypothesis test
test_cols = ['total day minutes', 'total eve minutes',
             'customer service calls']

for column in test_cols:
    #filters for filtering dataframe
    filter0 = df['churn'] == 0
    filter1 = df['churn'] == 1

    # filtering dataframes for hypothesis test
```

```
    df_0 = df.loc[filter0][column]
    df_1 = df.loc[filter1][column]

    # t test
    print('Null: Average', column, 'for churned customers is not greater', \
            'than non-churned customers')
    print('Alt: Average', column, 'for churned customers is greater', \
            'than non-churned customers', '\n')

    alpha = 0.05
    print('Alpha:', alpha)

    p_value = stats.ttest_ind(df_0, df_1).pvalue / 2
    print('P-value:', p_value)
    if p_value < alpha:
        print('Reject null hypothesis')
    else:
        print('Failed to reject null hypothesis')

    # space out the outputs
    print('\n','='*40)
```

```
Null: Average total day minutes for churned customers is not greater than non-ch
urned customers
Alt: Average total day minutes for churned customers is greater than non-churned
customers

Alpha: 0.05
P-value: 2.650139113746147e-33
Reject null hypothesis

 ========================================
Null: Average total eve minutes for churned customers is not greater than non-ch
urned customers
Alt: Average total eve minutes for churned customers is greater than non-churned
customers

Alpha: 0.05
P-value: 4.005669280643118e-08
Reject null hypothesis

 ========================================
Null: Average customer service calls for churned customers is not greater than n
on-churned customers
Alt: Average customer service calls for churned customers is greater than non-ch
urned customers

Alpha: 0.05
P-value: 1.9501801200945587e-34
Reject null hypothesis

 ========================================
```

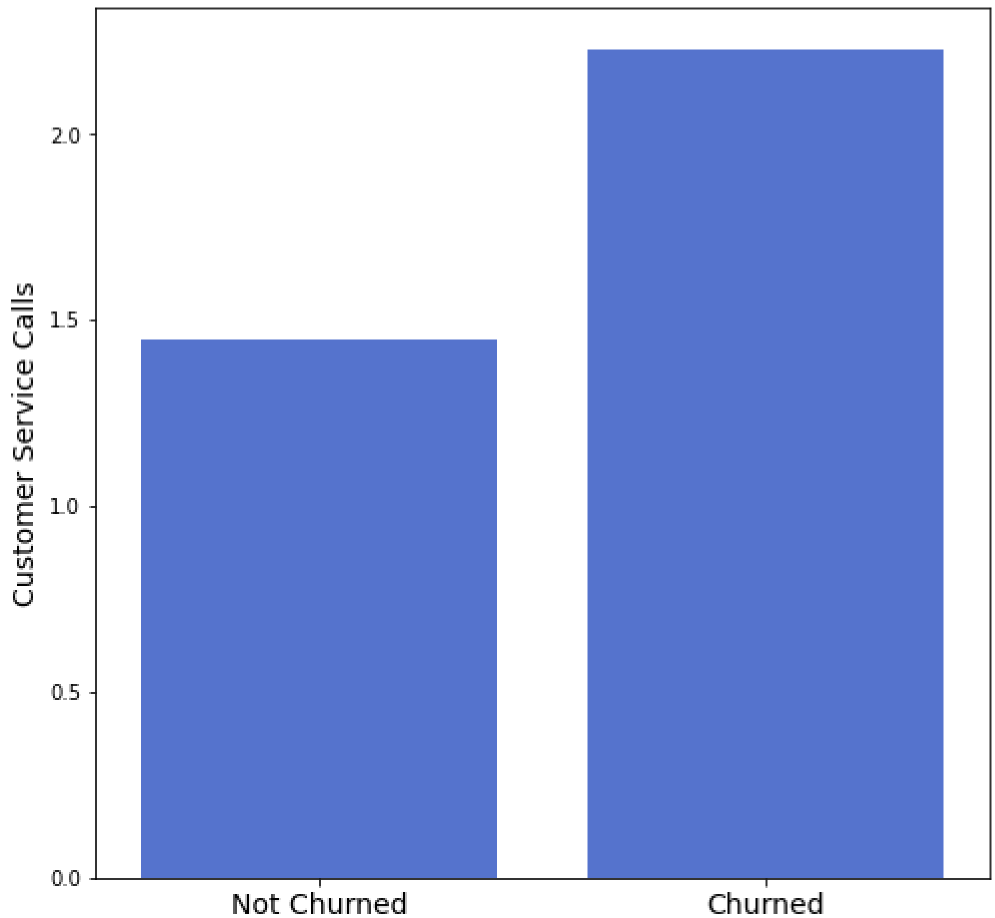
Visualizations

In [75]:

```
# average customer service calls
cs_calls_mean = df.groupby('churn').mean().reset_index()[['customer service cal
cs_calls_mean
```

```python
fig, ax = plt.subplots(figsize=(8,8))

# bar plot
sns.barplot(data=cs_calls_mean, x='churn', y='customer service calls',
            ax=ax, color='royalblue')

# set labels
ax.set_ylabel('Customer Service Calls', fontsize=14)
ax.set_xlabel('', fontsize=14)
ax.set_xticklabels(['Not Churned', 'Churned'], fontsize=14)
```

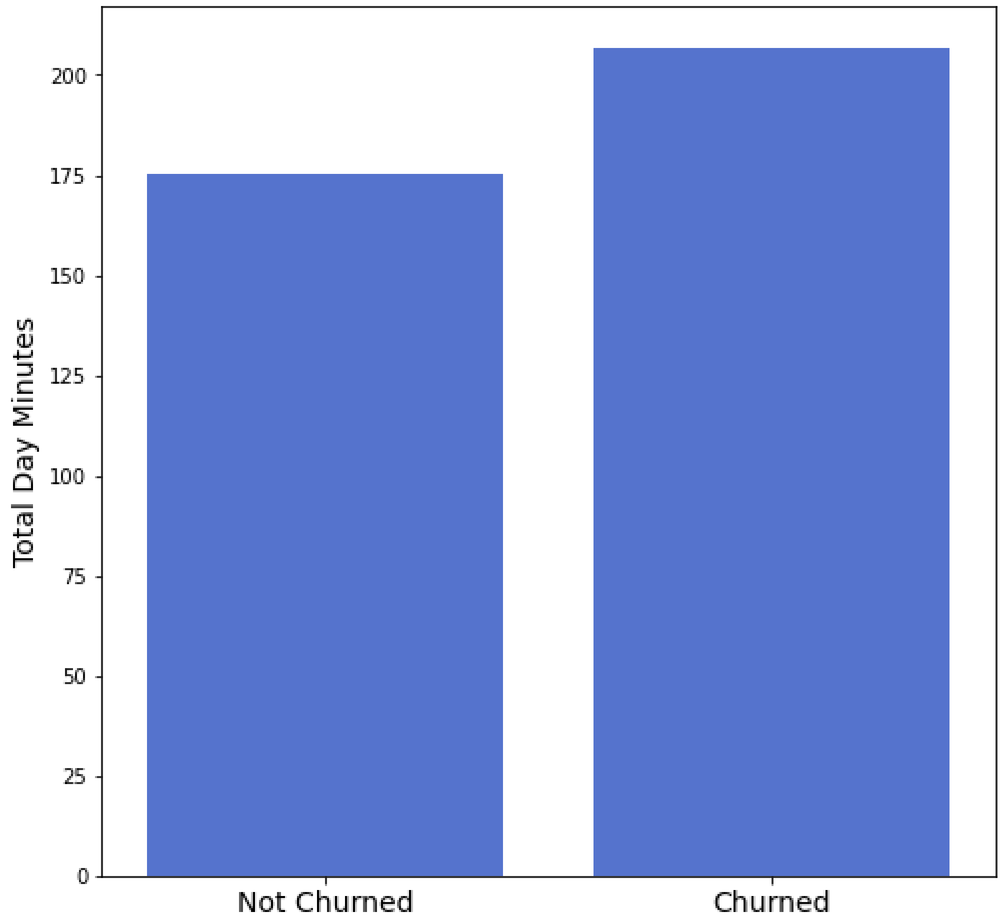
Out[75]: [Text(0, 0, 'Not Churned'), Text(1, 0, 'Churned')]



In [74]:

```python
# average customer service calls
td_mins_mean = df.groupby('churn').mean().reset_index()[['total day minutes', '


fig, ax = plt.subplots(figsize=(8,8))

# bar plot
sns.barplot(data=td_mins_mean, x='churn', y='total day minutes',
            ax=ax,  color='royalblue')

# set labels
ax.set_ylabel('Total Day Minutes', fontsize=14)
ax.set_xlabel('', fontsize=14)
ax.set_xticklabels(['Not Churned', 'Churned'], fontsize=14)
```

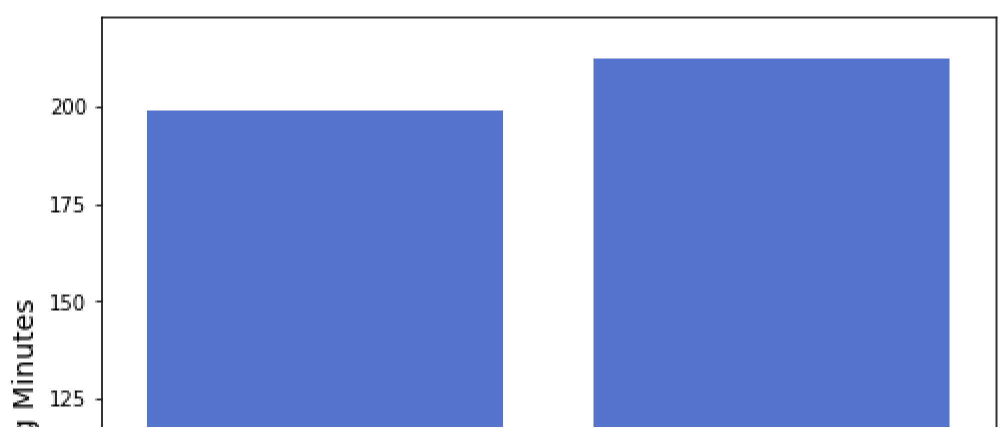
Out[74]: [Text(0, 0, 'Not Churned'), Text(1, 0, 'Churned')]

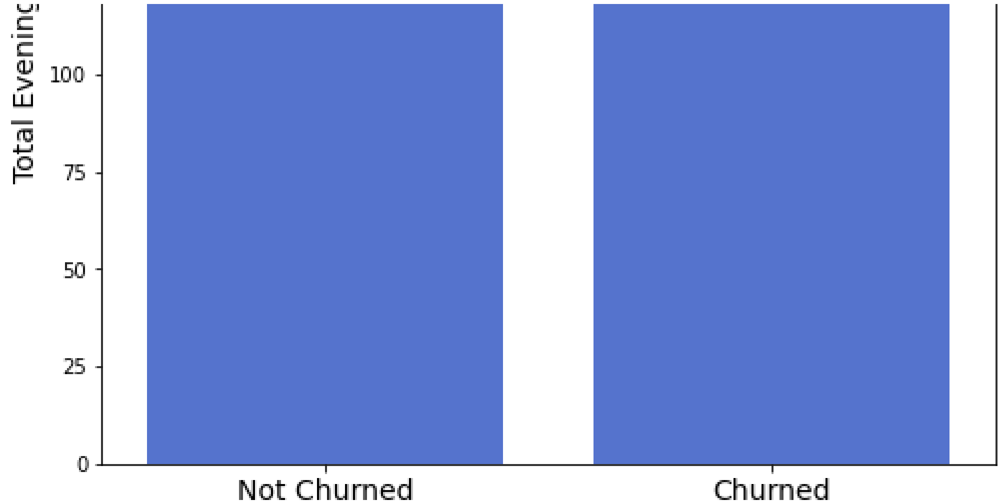


In [76]:

```
# average customer service calls
te_mins_mean = df.groupby('churn').mean().reset_index()[['total eve minutes', '

fig, ax = plt.subplots(figsize=(8,8))

# bar plot
sns.barplot(data=te_mins_mean, x='churn', y='total eve minutes',
            ax=ax, color='royalblue')

# set labels
ax.set_ylabel('Total Evening Minutes', fontsize=14)
ax.set_xlabel('', fontsize=14)
ax.set_xticklabels(['Not Churned', 'Churned'], fontsize=14)
```

Out[76]: [Text(0, 0, 'Not Churned'), Text(1, 0, 'Churned')]





Adding total columns

I want to add a total columns for minutes, calls, and charges for day, evening, night, and international.

In [32]:

```
# Total columns

# total minutes
df['total_minutes'] = df['total day minutes'] + df['total eve minutes'] \
                        + df['total night minutes'] + df['total intl minutes']

# total calls
df['total_calls'] = df['total day calls'] + df['total eve calls'] \
                        + df['total night calls'] + df['total intl calls']

# total charge
df['total_charge'] = df['total day charge'] + df['total eve charge'] \
                        + df['total night charge'] + df['total intl charge']
```

In [33]:

```
total_and_churn = ['total_minutes', 'total_calls',
                   'total_charge', 'churn']

# correlation between totals and churn
df[total_and_churn].corr()
```

Out[33]:

	total_minutes	total_calls	total_charge	churn
total_minutes	1.000000	0.018204	0.890804	0.198607
total_calls	0.018204	1.000000	0.022225	0.015807
total_charge	0.890804	0.022225	1.000000	0.231549
churn	0.198607	0.015807	0.231549	1.000000

In [34]:

```
# combining cont_cols and total lists
cont_cols.pop()
cont_cols.extend(total_and_churn)
```

Dropping unnecessary columns

Don't need area code or phone number for modeling. State I'll leave in for now. Maybe it will be useful when it comes to modeling.

In [37]:

```
df.drop(columns=['area code', 'phone number'], inplace=True)
```

In [38]:

```
# save to csv
df.to_csv('./Data/syriatel_clean.csv')
```