Steps for Initial Setup -

1. Chose Pyspark kernel in Jupyter Notebook.
2. Verify if SparkContext is available
3. Load the dataset in HDFS or check the path of pre-loaded data in HDFS which is /gldata/Churn.csv and /gldata/Churntest.csv If you want to check the data, access it through Hue- HDFS browser.
4. Learn how to work with Spark Dataframe, Pandas Dataframe and Spark ML Pipeline.

Steps for Project

1. Create and Check Spark Context for Pyspark shell.
2. Load necessary libraries
3. Check the informtion provided about data.
4. Import the data files provided from HDFS (Churn.csv and Churntest.csv).
5. Display the data in Spark Dataframe. (Note:: In pyspark, dataframe index the rows from 0 instead of 1)
6. Do data pre-processing required.(Hint - We have some variables which should be of categorical datatype but they are of type integer. Convert them)
7. Do exploratory data analysis.

```
    7.1 - Describe the data using describe function and state your insig
hts.
    7.2 - Create Histogram for Day minutes spent by customers for churn=
0 and 1 values.
    7.3 - Create count plots for Number of customers opt voicemail plan
with Churn values.
    7.4 - Create count plots for International Plan opt by customer with
Churn values.
    7.5 - Plot Areawise churner and non-churner.
    7.6 - Get correlation matrix using corr() function.
```

8. Get correlation between Predicting Variable and independent variable and state your insights. (Now that we want to predict which customer is going to churn, let’s see what columns might be interesting for our prediction.One way is to find the correlation between "Churn" and each of the other columns.This will show us which other columns might predict "Churn" the best.)
9. Applying Machine Learning Model

```
    9.1 - Import necessary libraries
    9.2 - Create vectors of all indepandant variables (Hint - use Vector
Assembler)
    9.3 - Apply Decision Tree Classifier using dependent and independent
variables.
    9.4 - Create pipeline to build the classifier.
    9.5 - Use startified sampling to get a sample of data.
    9.6 - Split the data into train and test dataset.
    9.7 - Make predictions and validate your model by calculating accura
cy score.
    9.8 - Calculate recall and precision score.
    9.9 - Test the model using test data and calculate accuracy, recall
and precision.
    9.10- Repeat steps from 9.3 to 9.9 for Random-forest and Gradient-Bo
ost Classifiers.
```

10. State your insights and conclusions from the above analysis.

About the Data

The dataset is about telecom industry which tells about the number of customers who churned the service. It consists of 3333 observations having 21 variables. We have to predict which customer is going to churn the service.

Account.Length: how long account has been active.

VMail.Message: Number of voice mail messages send by the customer.

Day.Mins: Time spent on day calls.

Eve.Mins: Time spent on evening calls.

Night.Mins: Time spent on night calls.

Intl. Mins: Time spent on international calls.

Day.Calls: Number of day calls by customers.

Eve.Calls: Number of evening calls by customers.

Intl.Calls: Number of international calls.

Night.Calls: Number of night calls by customer.

Day.Charge: Charges of Day Calls.

Night.Charge: Charges of Night Calls.

Eve.Charge: Charges of evening Calls.

Intl.Charge: Charges of international calls.

VMail.Plan: Voice mail plan taken by the customer or not.

State: State in Area of study.

Phone: Phone number of the customer.

Area.Code: Area Code of customer.

Int.l.Plan: Does customer have international plan or not.

CustServ.Calls: Number of customer service calls by customer.

Churn : Customers who churned the telecom service or who doesn’t(0=“Churner”, 1=“Non-Churner”)

Get started

The project can be deivided into 4 blocks. We have added sample solution till step 8 of project.

You need to proceed for step 8 and 9 to complete this project.

Loading the Data and Pre-Processing

Follow steps for Project - 1 to 6.

In [3]:
```
sc
```

Out[3]:
```
''
```

Loading Libraries

In [2]:
```
##We have to load libraries before we start our analysis.

import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
pd.options.display.mpl_style = 'default'
import numpy as np
from pyspark.sql.types import *
from pyspark.sql import Row
import seaborn as sns
from pyspark import SparkContext
from pyspark import SQLContext
```

```
---------------------------------------------------------------------------
-
OptionError                               Traceback (most recent call las
t)
<ipython-input-2-6ac212add3fa> in <module>()
      6 get_ipython().magic(u'matplotlib inline')
      7 import pandas as pd
----> 8 pd.options.display.mpl_style = 'default'
      9 import numpy as np
     10 from pyspark.sql.types import *

/usr/local/anaconda/python2/lib/python2.7/site-packages/pandas/core/confi
g.pyc in __setattr__(self, key, val)
    190             _set_option(prefix, val)
    191         else:
--> 192             raise OptionError("You can only set the value of exist
ing options")
    193
    194     def __getattr__(self, key):

OptionError: 'You can only set the value of existing options'
```

In [ ]:
```
sc = SparkContext()
spark = SQLContext(sc)
```

Import the Data

We have two data files in which training file contains 3333 observations with 21 variables and testing file contains 667 observations.

In [2]:
```
ch = spark.read.csv("C:/churn.csv",header=True,inferSchema=True)
```

Display the dataframe

Note:: In pyspark, dataframe index the rows from 0 instead of 1.

In [3]:
```
ch
```

Out[3]:

	Account Length	VMail Message	Day Mins	Eve Mins	Night Mins	Intl Mins	CustServ Calls	Churn	Intl Plan	VMail Plan	...	D C
0	128	25	265.1	197.4	244.7	10.0	1	0	0	1	...	4
1	107	26	161.6	195.5	254.4	13.7	1	0	0	1	...	2
2	137	0	243.4	121.2	162.6	12.2	0	0	0	0	...	4
3	84	0	299.4	61.9	196.9	6.6	2	0	1	0	...	5
4	75	0	166.7	148.3	186.9	10.1	3	0	1	0	...	2
5	118	0	223.4	220.6	203.9	6.3	0	0	1	0	...	3
6	121	24	218.2	348.5	212.6	7.5	3	0	0	1	...	3
7	147	0	157.0	103.1	211.8	7.1	0	0	1	0	...	2
8	117	0	184.5	351.6	215.8	8.7	1	0	0	0	...	3
9	141	37	258.6	222.0	326.4	11.2	0	0	1	1	...	4
10	65	0	129.1	228.5	208.8	12.7	4	1	0	0	...	2
11	74	0	187.7	163.4	196.0	9.1	0	0	0	0	...	3
12	168	0	128.8	104.9	141.1	11.2	1	0	0	0	...	2
13	95	0	156.6	247.6	192.3	12.3	3	0	0	0	...	2
14	62	0	120.7	307.2	203.0	13.1	4	0	0	0	...	2
15	161	0	332.9	317.8	160.6	5.4	4	1	0	0	...	5
16	85	27	196.4	280.9	89.3	13.8	1	0	0	1	...	3
17	93	0	190.7	218.2	129.6	8.1	3	0	0	0	...	3
18	76	33	189.7	212.8	165.7	10.0	1	0	0	1	...	3
19	73	0	224.4	159.5	192.8	13.0	1	0	0	0	...	3

	Account Length	VMail Message	Day Mins	Eve Mins	Night Mins	Intl Mins	CustServ Calls	Churn	Intl Plan	VMail Plan	...	D C
20	147	0	155.1	239.7	208.8	10.6	0	0	0	0	...	2
21	77	0	62.4	169.9	209.6	5.7	5	1	0	0	...	1
22	130	0	183.0	72.9	181.8	9.5	0	0	0	0	...	3
23	111	0	110.4	137.3	189.6	7.7	2	0	0	0	...	1
24	132	0	81.1	245.2	237.0	10.3	0	0	0	0	...	1
25	174	0	124.3	277.1	250.7	15.5	3	0	0	0	...	2
26	57	39	213.0	191.1	182.7	9.5	0	0	0	1	...	3
27	54	0	134.3	155.5	102.1	14.7	3	0	0	0	...	2
28	20	0	190.0	258.2	181.5	6.3	0	0	0	0	...	3
29	49	0	119.3	215.1	178.7	11.1	1	0	0	0	...	2
...	...	...	...	...	...	...	...	...	...	...	...	.
3303	114	26	137.1	155.7	247.6	11.5	2	0	0	1	...	2
3304	71	0	186.1	198.6	206.5	13.8	4	1	1	0	...	3
3305	58	22	224.1	238.8	174.2	11.5	2	0	0	1	...	3
3306	106	29	83.6	203.9	229.5	8.1	1	0	0	1	...	1
3307	172	0	203.9	234.0	160.7	17.8	4	0	0	0	...	3
3308	45	0	211.3	165.7	265.9	13.3	1	0	0	0	...	3
3309	100	0	219.4	225.7	255.3	12.0	4	0	1	0	...	3
3310	94	0	190.4	92.0	224.8	13.6	2	0	0	0	...	3
3311	128	0	147.7	283.3	188.3	6.9	2	0	0	0	...	2

	Account Length	VMail Message	Day Mins	Eve Mins	Night Mins	Intl Mins	CustServ Calls	Churn	Intl Plan	VMail Plan	...	D C
3312	181	0	229.9	144.4	262.4	14.2	2	0	0	0	...	3
3313	127	0	102.8	143.7	191.4	10.0	1	0	0	0	...	1
3314	89	0	178.7	233.7	131.9	9.1	1	0	0	0	...	3
3315	149	18	148.5	114.5	178.3	6.5	0	0	0	1	...	2
3316	103	29	164.1	219.1	220.3	12.3	0	0	0	1	...	2
3317	163	0	197.2	188.5	211.1	7.8	1	0	1	0	...	3
3318	52	0	124.9	300.5	192.5	11.6	2	0	0	0	...	2
3319	89	0	115.4	209.9	280.9	15.9	3	0	0	0	...	1
3320	122	0	140.0	196.4	120.1	9.7	4	1	1	0	...	2
3321	60	0	193.9	85.0	210.1	13.2	3	0	0	0	...	3
3322	62	0	321.1	265.5	180.5	11.5	4	1	0	0	...	5
3323	117	0	118.4	249.3	227.0	13.6	5	1	0	0	...	2
3324	159	0	169.8	197.7	193.7	11.6	1	0	0	0	...	2
3325	78	0	193.4	116.9	243.3	9.3	2	0	0	0	...	3
3326	96	0	106.6	284.8	178.9	14.9	1	0	0	0	...	1
3327	79	0	134.7	189.7	221.4	11.8	2	0	0	0	...	2
3328	192	36	156.2	215.5	279.1	9.9	2	0	0	1	...	2
3329	68	0	231.1	153.4	191.3	9.6	3	0	0	0	...	3
3330	28	0	180.8	288.8	191.9	14.1	2	0	0	0	...	3
3331	184	0	213.8	159.6	139.2	5.0	2	0	1	0	...	3

	Account Length	VMail Message	Day Mins	Eve Mins	Night Mins	Intl Mins	CustServ Calls	Churn	Intl Plan	VMail Plan	...	D C
3332	74	25	234.4	265.9	241.4	13.7	0	0	0	1	...	3

3333 rows × 21 columns

Data Preprocessing

We have some variables which should be of categorical datatype but they are of type integer. First we have to convert them into as categorical variable.

In [4]:
```
ch['Churn']= ch['Churn'].astype('category')
ch['Intl Plan']= ch['Intl Plan'].astype('category')
ch['VMail Plan']= ch['VMail Plan'].astype('category')
```

Exploratory Data Analysis

Follow steps 7 and 8.

Summary of Dataframe

In pyspark we use 'describe()' to display the summary of variables in dataframe. Also the describe variables doesn't show the summary of categorical variables in dataframe. We have to explore them explicitly.

In [5]:
```
ch.describe()
```

Out[5]:

	Account Length	VMail Message	Day Mins	Eve Mins	Night Mins	Intl Mins
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.00000(
mean	101.064806	8.099010	179.775098	200.980348	200.872037	10.237294
std	39.822106	13.688365	54.467389	50.713844	50.573847	2.791840
min	1.000000	0.000000	0.000000	0.000000	23.200000	0.000000
25%	74.000000	0.000000	143.700000	166.600000	167.000000	8.500000
50%	101.000000	0.000000	179.400000	201.400000	201.200000	10.300000
75%	127.000000	20.000000	216.400000	235.300000	235.300000	12.100000
max	243.000000	51.000000	350.800000	363.700000	395.000000	20.000000

From the obtained table we find that, summary of dataframe includes-

count- displaying the number of observations in each variable.

mean- the mean value around which each observation lies.

std- standard deviation; Its a measure of how much close to the mean value the actual data points are.

min and max- displays the maximum and minimum value of a variable in dataframe. e.g. in variable account length the service used by people for minimum 1 day and maximum for 243 days.

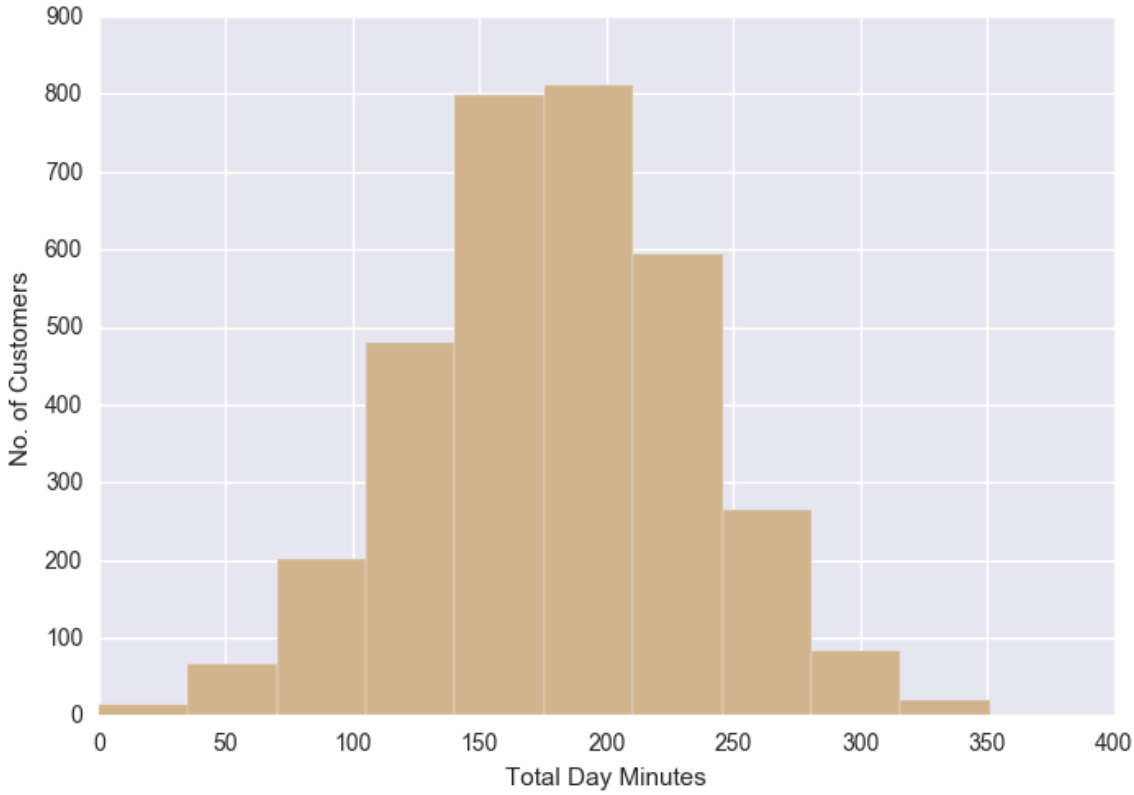
25%- it shows the first quartile of variable e.g. if we look at the Account Length then it means 25% of the people used service for 74 days, similarly for other variables.

50%- it shows second quartile of variable e.g. in the vaiable Account Length then it means 50% of the people used service for 101 days.

75%- shows the third quartile and 75% of people used service for 127 days.

Histogram for Day minutes spent by customers

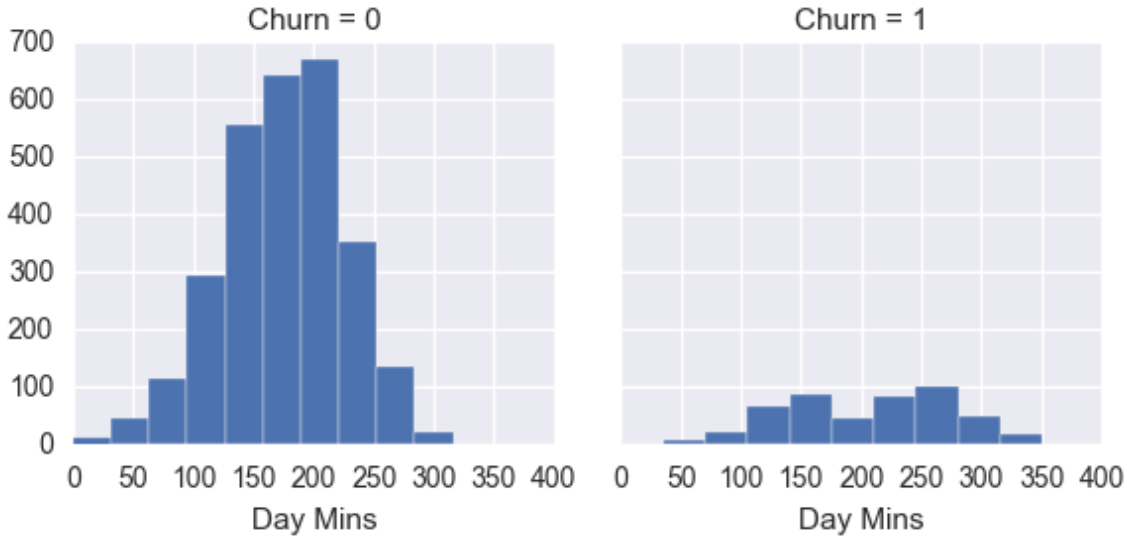
In [6]:
```
plt.hist(ch['Day Mins'], bins= 10, facecolor= 'tan')
plt.xlabel('Total Day Minutes')
plt.ylabel('No. of Customers')
plt.show()
```



The histogram shows that total day minutes spent by customers on day minutes during their service period.

In [8]:
```
import seaborn as sns
g = sns.FacetGrid(ch, col="Churn")
g.map(plt.hist, "Day Mins")
```

Out[8]: <seaborn.axisgrid.FacetGrid at 0xb9da630>



In the above plot **churn=0** shows **non-churner** and **churn=1** shows **churner**.

Number of customers opt voice mail plan

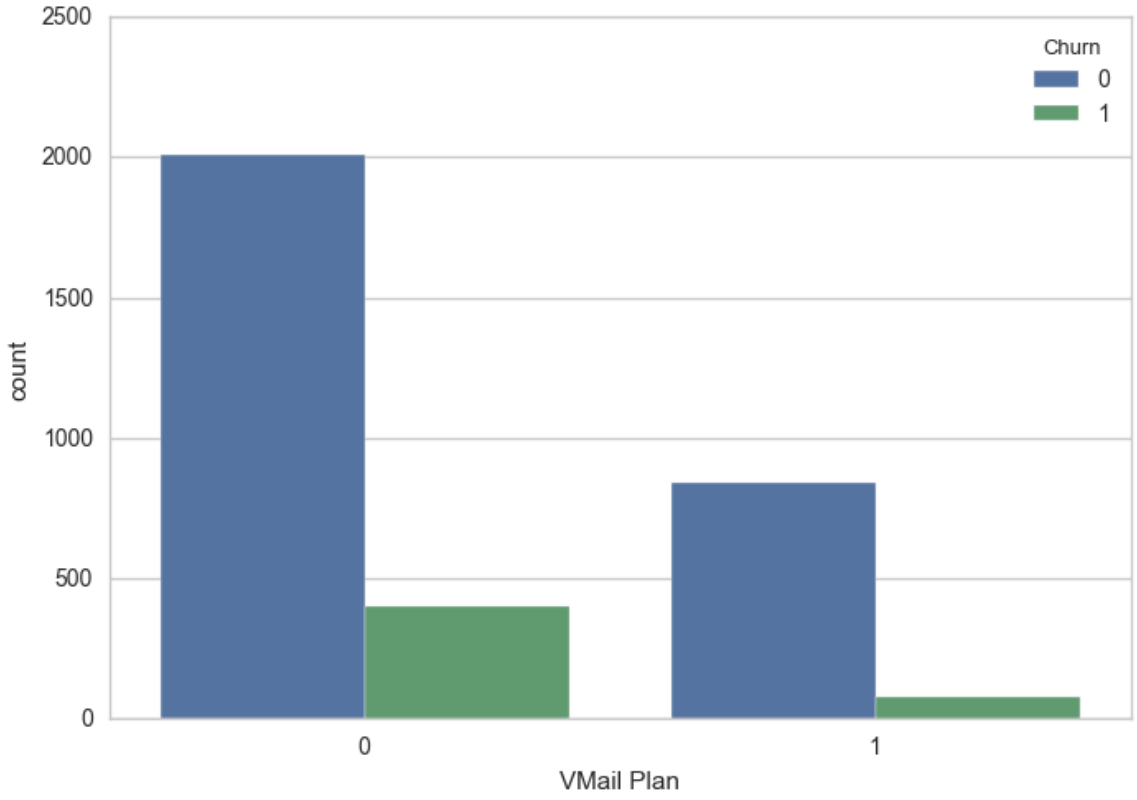
In [8]:
```
ch['VMail Plan'].value_counts()
```

Out[8]:
```
0    2411
1     922
Name: VMail Plan, dtype: int64
```

We find that 2411 customers who didn't opt the voice mail plan and 922 customers who opted the voice mail plan.

In [9]:
```
sns.set(style="whitegrid", color_codes=True)
sns.countplot(x="VMail Plan", hue= "Churn", data=ch)
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x5f33470>
```



International Plan opt by customer

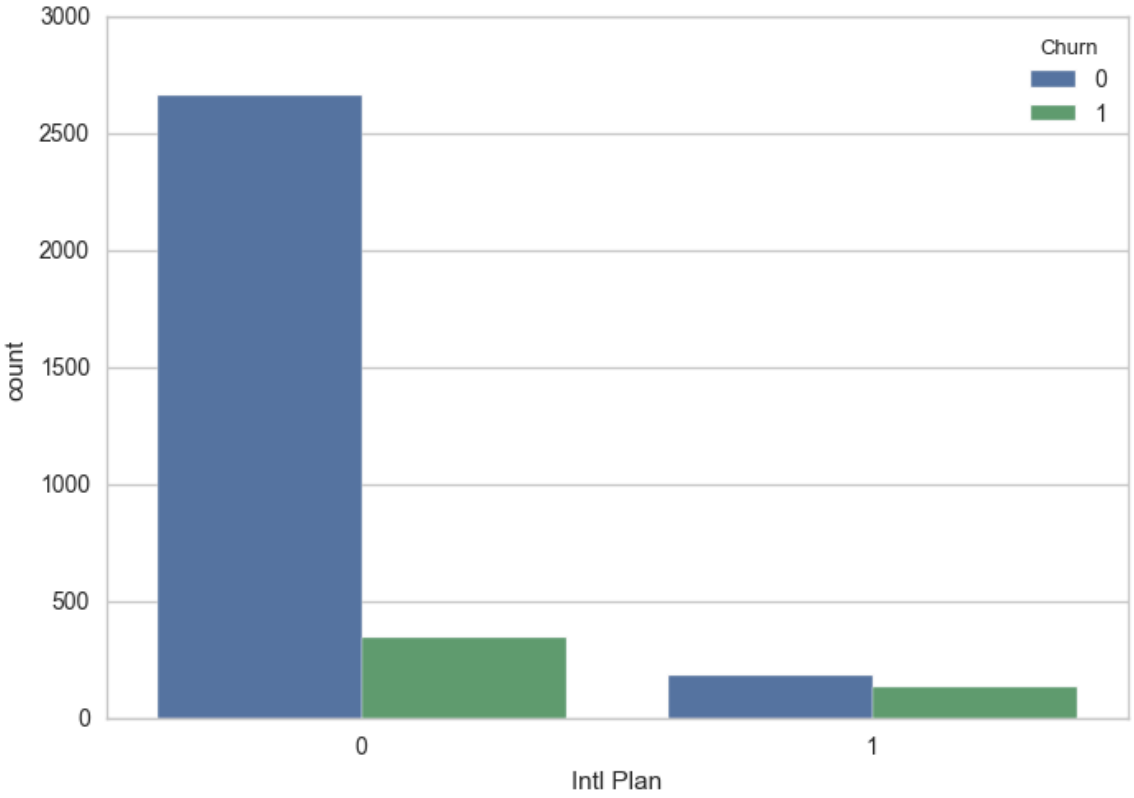
In [10]:
```
ch['Intl Plan'].value_counts()
```

Out[10]:
```
0    3010
1     323
Name: Intl Plan, dtype: int64
```

We find that 3010 customer didn't opt international plan and 323 customers opt the international plan.

In [11]:
```
sns.countplot(x="Intl Plan", hue= "Churn", data=ch)
```

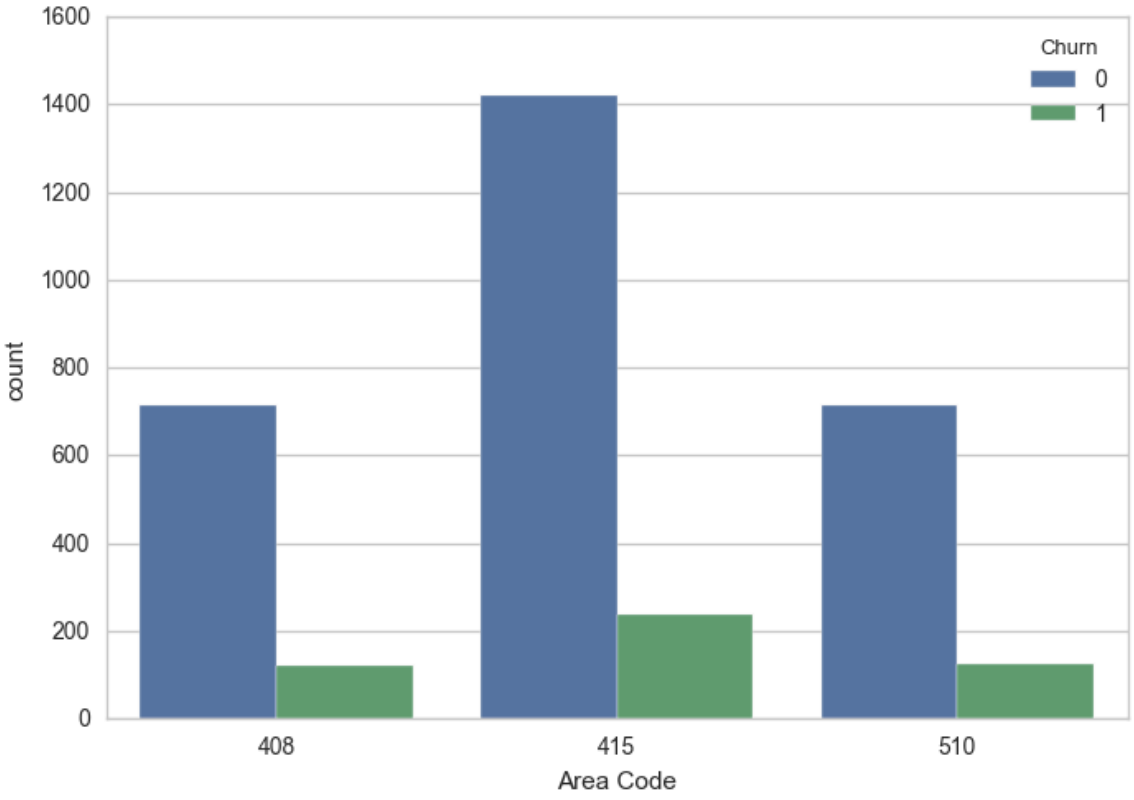
Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xbd21208>
```



Areawise churner and non-churner

In [12]:
```
ch['Area Code']= ch['Area Code'].astype('category')
sns.countplot(x="Area Code", hue= "Churn", data=ch)
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0xbdcaf60>



Correlation Matrix

In [9]:
```
ch.corr('pearson')
```

Out[9]:

	Account Length	VMail Message	Day Mins	Eve Mins	Night Mins	Intl Mins	CustServ Calls	I
Account Length	1.000000	-0.004628	0.006216	-0.006757	-0.008955	0.009514	-0.003796	(
VMail Message	-0.004628	1.000000	0.000778	0.017562	0.007681	0.002856	-0.013263	-
Day Mins	0.006216	0.000778	1.000000	0.007043	0.004323	-0.010155	-0.013423	(
Eve Mins	-0.006757	0.017562	0.007043	1.000000	-0.012584	-0.011035	-0.012985	-
Night Mins	-0.008955	0.007681	0.004323	-0.012584	1.000000	-0.015207	-0.009288	(
Intl Mins	0.009514	0.002856	-0.010155	-0.011035	-0.015207	1.000000	-0.009640	(
CustServ Calls	-0.003796	-0.013263	-0.013423	-0.012985	-0.009288	-0.009640	1.000000	-
Day Calls	0.038470	-0.009548	0.006750	-0.021451	0.022938	0.021565	-0.018942	1
Day Charge	0.006214	0.000776	1.000000	0.007050	0.004324	-0.010157	-0.013427	(
Eve Calls	0.019260	-0.005864	0.015769	-0.011430	-0.002093	0.008703	0.002423	(
Eve Charge	-0.006745	0.017578	0.007029	1.000000	-0.012592	-0.011043	-0.012987	-
Night Calls	-0.013176	0.007123	0.022972	0.007586	0.011204	-0.013605	-0.012802	-
Night Charge	-0.008960	0.007663	0.004300	-0.012593	0.999999	-0.015214	-0.009277	(
Intl Calls	0.020661	0.013957	0.008033	0.002541	-0.012353	0.032304	-0.017561	(
Intl Charge	0.009546	0.002884	-0.010092	-0.011067	-0.015180	0.999993	-0.009675	(
Area Code	-0.012463	-0.001994	-0.008264	0.003580	-0.005825	-0.018288	0.027572	-

Correlation between Predicting Variable and independent variable.

Now that we want to predict which customer is going to churn, let’s see what columns might be interesting for our prediction. One way is to find the correlation between "Churn" and each of the other columns. This will show us which other columns might predict "Churn" the best.

In [99]:
```
ch.corr()["Churn"]
```

Out[99]:
```
Account Length    0.016541
VMail Message    -0.089728
Day Mins          0.205151
Eve Mins          0.092796
Night Mins        0.035493
Intl Mins         0.068239
CustServ Calls    0.208750
Churn             1.000000
Intl Plan         0.259852
VMail Plan       -0.102148
Day Calls         0.018459
Day Charge        0.205151
Eve Calls         0.009233
Eve Charge        0.092786
Night Calls       0.006141
Night Charge      0.035496
Intl Calls       -0.052844
Intl Charge       0.068259
Area Code         0.006174
Name: Churn, dtype: float64
```

From the obtained table, we find that **Day minutes**, **Customer Serv Calls**, **Intl Plan** has weak positive correlation with predicting variable **Churn**, while **VMail Message**, **VMail Plan** and **Intl Calls** has weak negative correlation. While remaining variables has either weak positive or negative correlation.

Strong Correlation lies in the range of ±0.5 to ±1

Weak Correlation lies in the range of ±0.1 to ±0.1

Building Machine Learning Model

Follow project step - 9

Insights and Conclusions

Follow step no 10 of Project.