In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
df=pd.read_csv("C://Users//user//Downloads//churn.csv")
df
```

Out[2]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls
0	LA	117	408	No	No	0	184.5	97	31.37	351.6	80
1	IN	65	415	No	No	0	129.1	137	21.95	228.5	83
2	NY	161	415	No	No	0	332.9	67	56.59	317.8	97
3	SC	111	415	No	No	0	110.4	103	18.77	137.3	102
4	HI	49	510	No	No	0	119.3	117	20.28	215.1	109
...	...	...	...	...	...	...	...	...	...	...	...
662	WI	114	415	No	Yes	26	137.1	88	23.31	155.7	125
663	AL	106	408	No	Yes	29	83.6	131	14.21	203.9	131
664	VT	60	415	No	No	0	193.9	118	32.96	85.0	110
665	WV	159	415	No	No	0	169.8	114	28.87	197.7	105
666	CT	184	510	Yes	No	0	213.8	105	36.35	159.6	84

667 rows × 20 columns

In [3]:
```
df.head()
```

Out[3]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	cl
0	LA	117	408	No	No	0	184.5	97	31.37	351.6	80	
1	IN	65	415	No	No	0	129.1	137	21.95	228.5	83	
2	NY	161	415	No	No	0	332.9	67	56.59	317.8	97	
3	SC	111	415	No	No	0	110.4	103	18.77	137.3	102	
4	HI	49	510	No	No	0	119.3	117	20.28	215.1	109	

In [4]:
```
df.tail()
```

Out[4]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls
662	WI	114	415	No	Yes	26	137.1	88	23.31	155.7	125
663	AL	106	408	No	Yes	29	83.6	131	14.21	203.9	131
664	VT	60	415	No	No	0	193.9	118	32.96	85.0	110
665	WV	159	415	No	No	0	169.8	114	28.87	197.7	105
666	CT	184	510	Yes	No	0	213.8	105	36.35	159.6	84

In [5]:
```
df.shape
```

Out[5]:
```
(667, 20)
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 667 entries, 0 to 666
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   State                   667 non-null    object 
 1   Account length          667 non-null    int64  
 2   Area code               667 non-null    int64  
 3   International plan      667 non-null    object 
 4   Voice mail plan         667 non-null    object 
 5   Number vmail messages   667 non-null    int64  
 6   Total day minutes       667 non-null    float64
 7   Total day calls         667 non-null    int64  
 8   Total day charge        667 non-null    float64
 9   Total eve minutes       667 non-null    float64
 10  Total eve calls         667 non-null    int64  
 11  Total eve charge        667 non-null    float64
 12  Total night minutes     667 non-null    float64
 13  Total night calls       667 non-null    int64  
 14  Total night charge      667 non-null    float64
 15  Total intl minutes      667 non-null    float64
 16  Total intl calls        667 non-null    int64  
 17  Total intl charge       667 non-null    float64
 18  Customer service calls  667 non-null    int64  
 19  Churn                   667 non-null    bool   
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 99.8+ KB
```

In [7]:
```
df.describe().T
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
Account length	667.0	102.841079	40.819480	1.00	76.00	102.00	128.000	232.00
Area code	667.0	436.157421	41.783305	408.00	408.00	415.00	415.000	510.00
Number vmail messages	667.0	8.407796	13.994480	0.00	0.00	0.00	20.000	51.00
Total day minutes	667.0	180.948126	55.508628	25.90	146.25	178.30	220.700	334.30
Total day calls	667.0	100.937031	20.396790	30.00	87.50	101.00	115.000	165.00
Total day charge	667.0	30.761769	9.436463	4.40	24.86	30.31	37.520	56.83
Total eve minutes	667.0	203.355322	49.719268	48.10	171.05	203.70	236.450	361.80
Total eve calls	667.0	100.476762	18.948262	37.00	88.00	101.00	113.000	168.00
Total eve charge	667.0	17.285262	4.226160	4.09	14.54	17.31	20.095	30.75
Total night minutes	667.0	199.685307	49.759931	23.20	167.95	201.60	231.500	367.70
Total night calls	667.0	100.113943	20.172505	42.00	86.00	100.00	113.500	175.00
Total night charge	667.0	8.985907	2.239429	1.04	7.56	9.07	10.420	16.55
Total intl minutes	667.0	10.238381	2.807850	0.00	8.60	10.50	12.050	18.30
Total intl calls	667.0	4.527736	2.482442	0.00	3.00	4.00	6.000	18.00
Total intl charge	667.0	2.764948	0.758167	0.00	2.32	2.84	3.255	4.94
Customer service calls	667.0	1.563718	1.333357	0.00	1.00	1.00	2.000	8.00

In [8]:
```
df["Churn"].value_counts()
```

Out[8]:
```
Churn
False    572
True      95
Name: count, dtype: int64
```

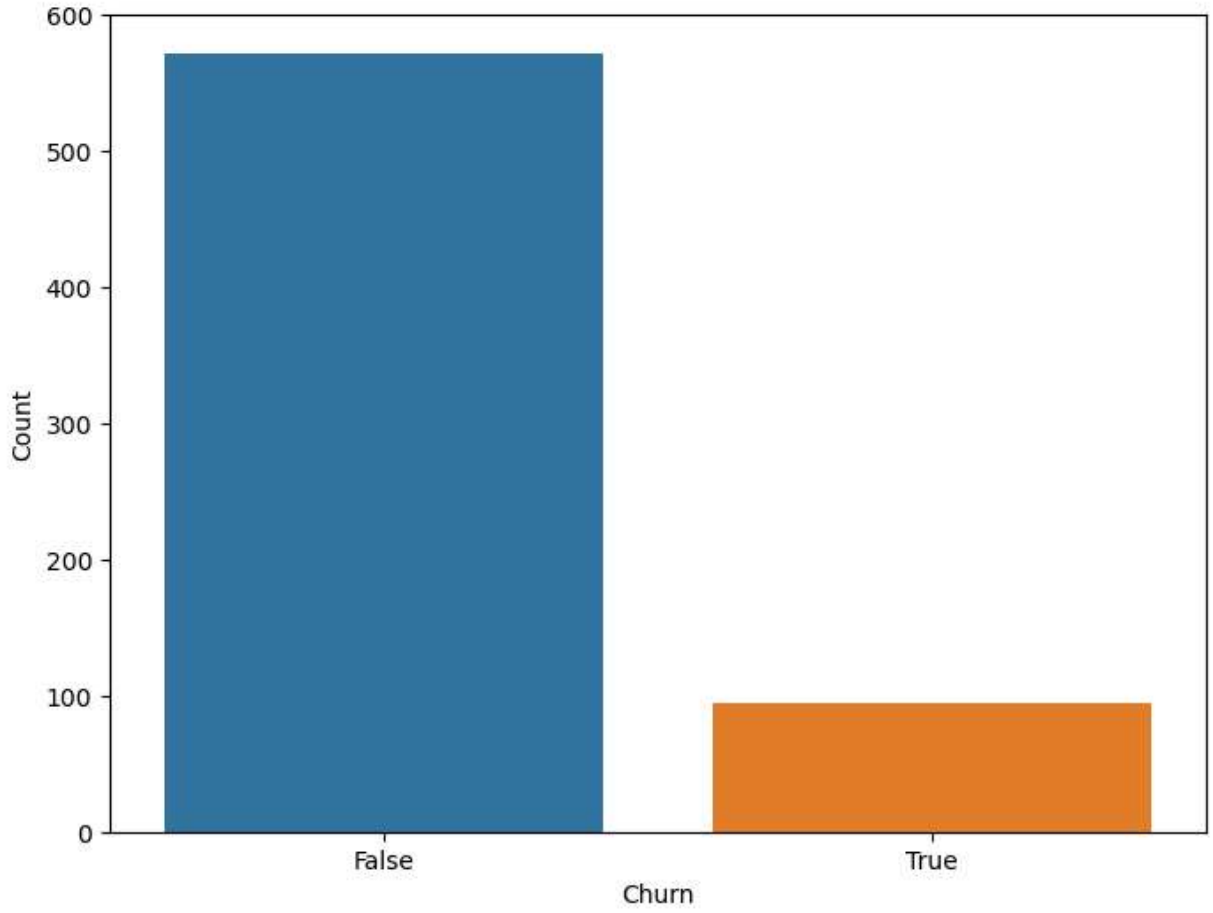
In [9]:
```
df.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
df.isnull().sum()
```

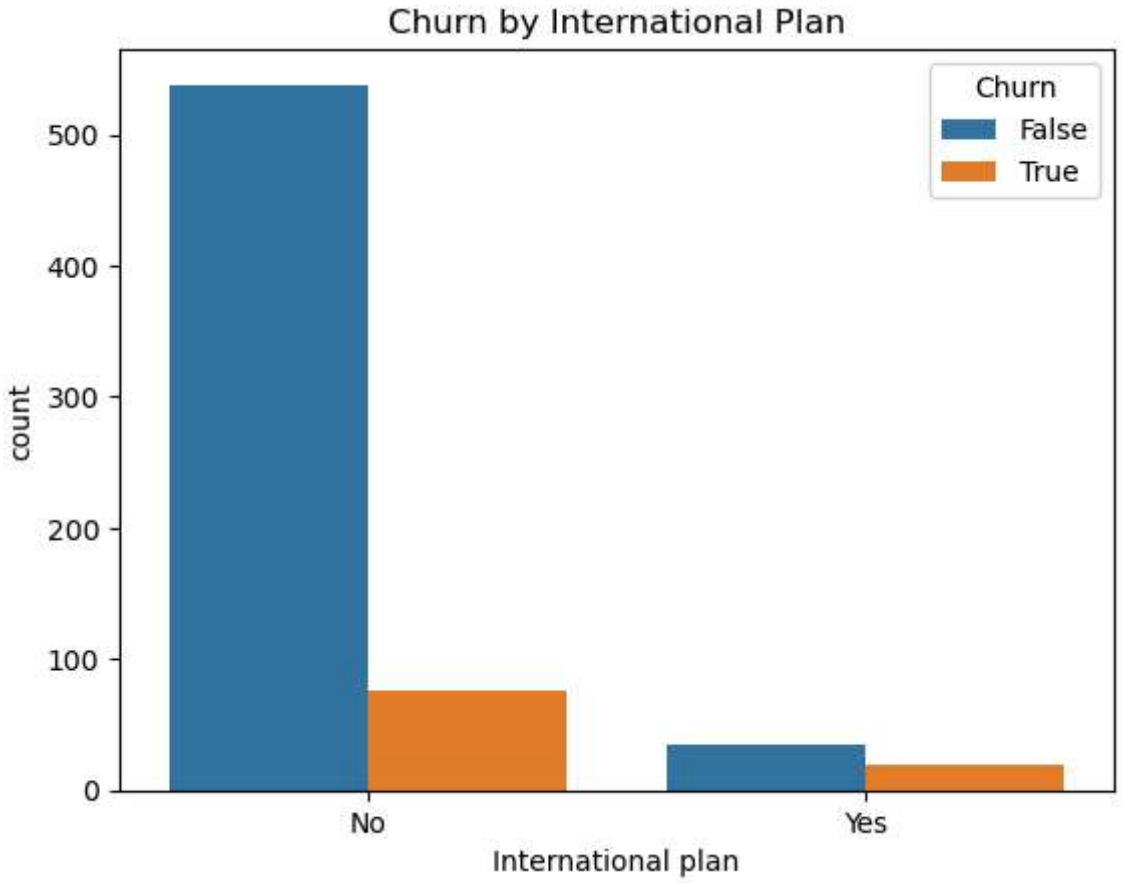
Out[10]:
```
State                     0
Account length            0
Area code                 0
International plan        0
Voice mail plan           0
Number vmail messages     0
Total day minutes         0
Total day calls           0
Total day charge          0
Total eve minutes         0
Total eve 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 [12]:
```
plt.figure(figsize=(8,6))
sns.countplot(x="Churn",data=df)
plt.xlabel("Churn")
plt.ylabel("Count")
plt.show()
```

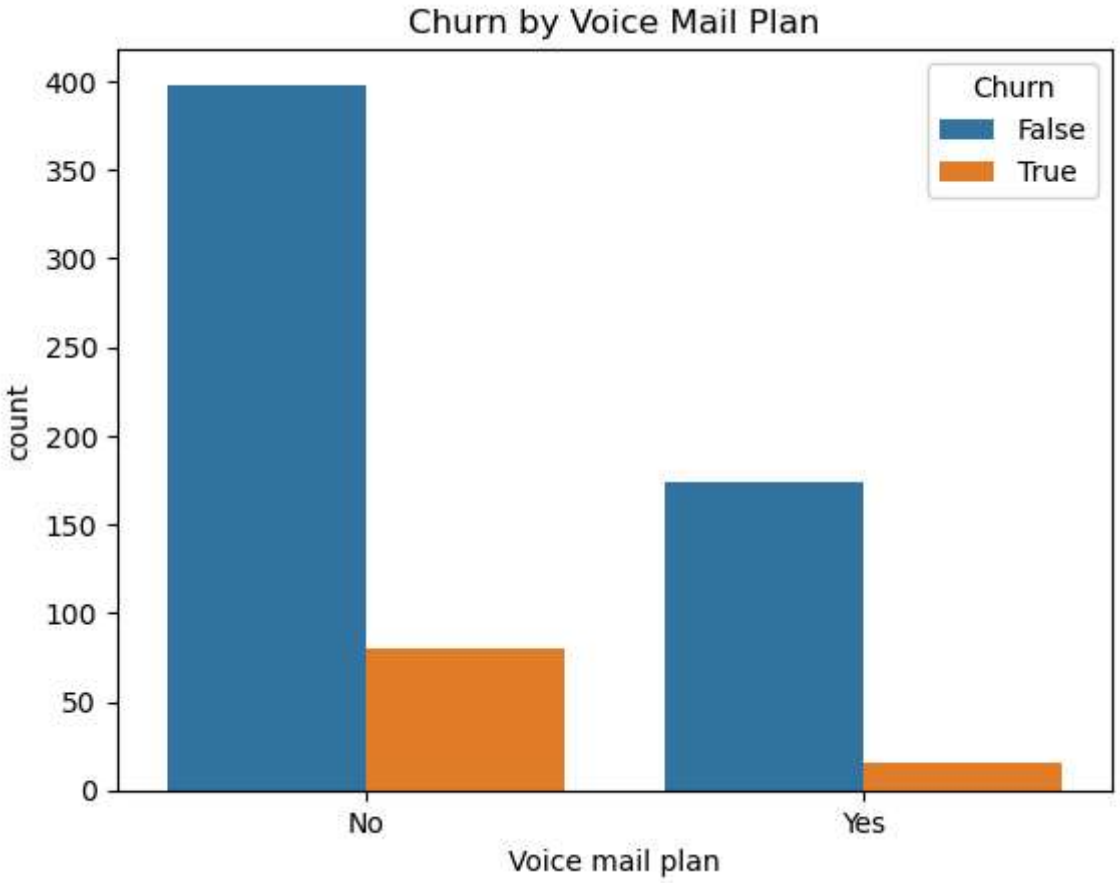


In [ ]:
```
df.columns
```

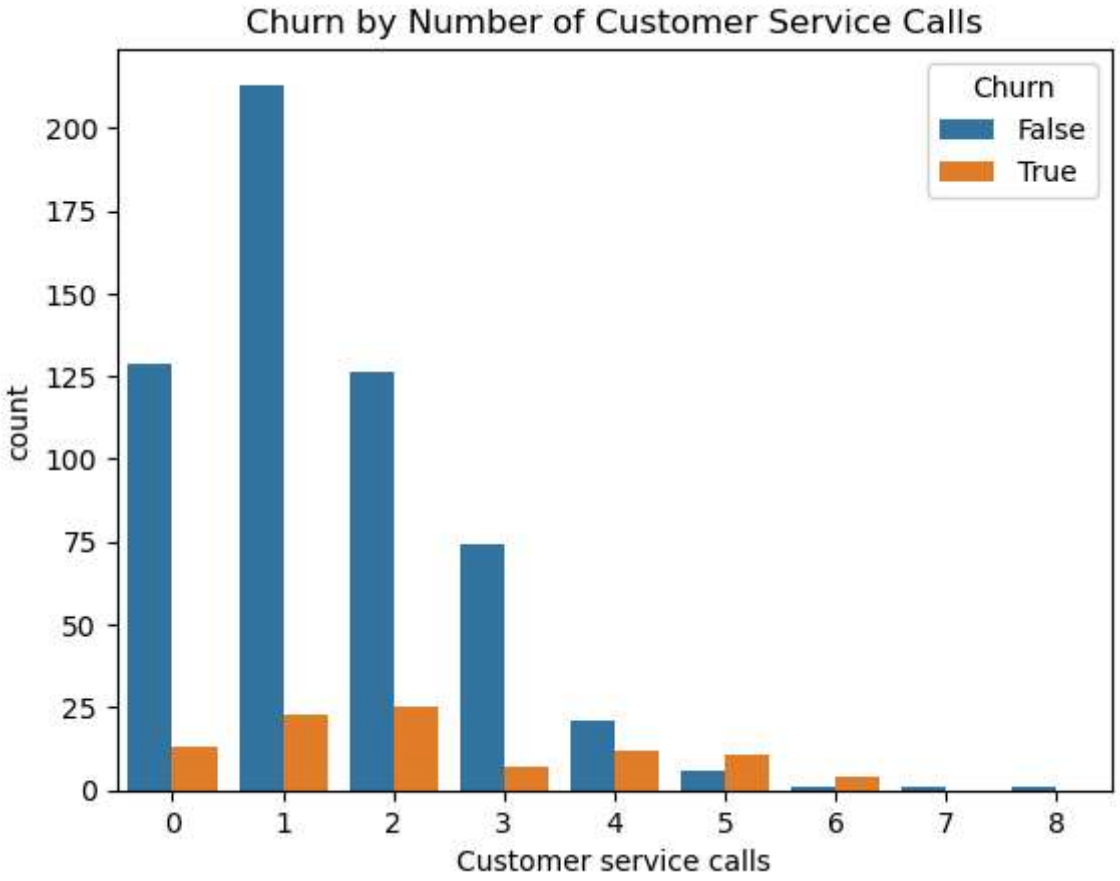
In [13]:
```
sns.countplot(x='International plan', hue='Churn', data=df)
plt.title("Churn by International Plan")
plt.show()
```



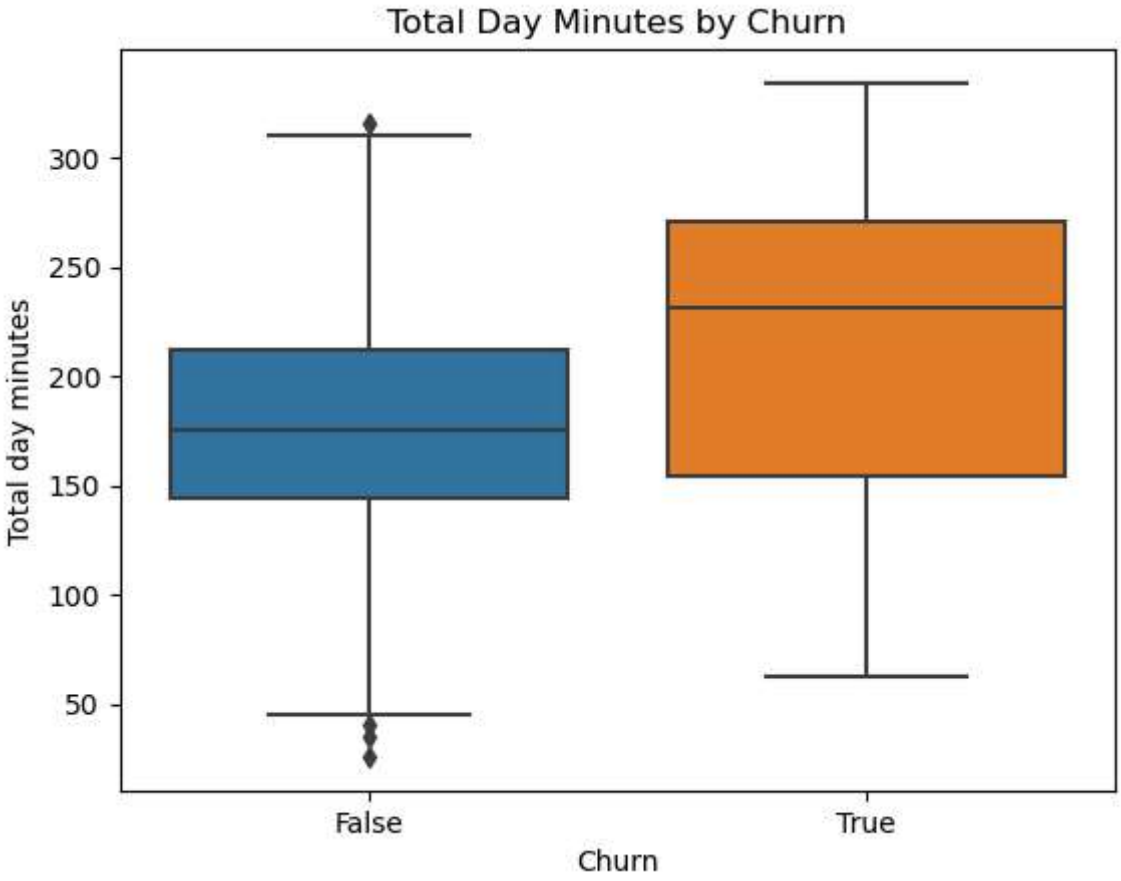
In [14]:
```
sns.countplot(x='Voice mail plan', hue='Churn', data=df)
plt.title("Churn by Voice Mail Plan")
plt.show()
```



In [16]:
```
sns.countplot(x='Customer service calls', hue='Churn', data=df)
plt.title("Churn by Number of Customer Service Calls")
plt.show()
```

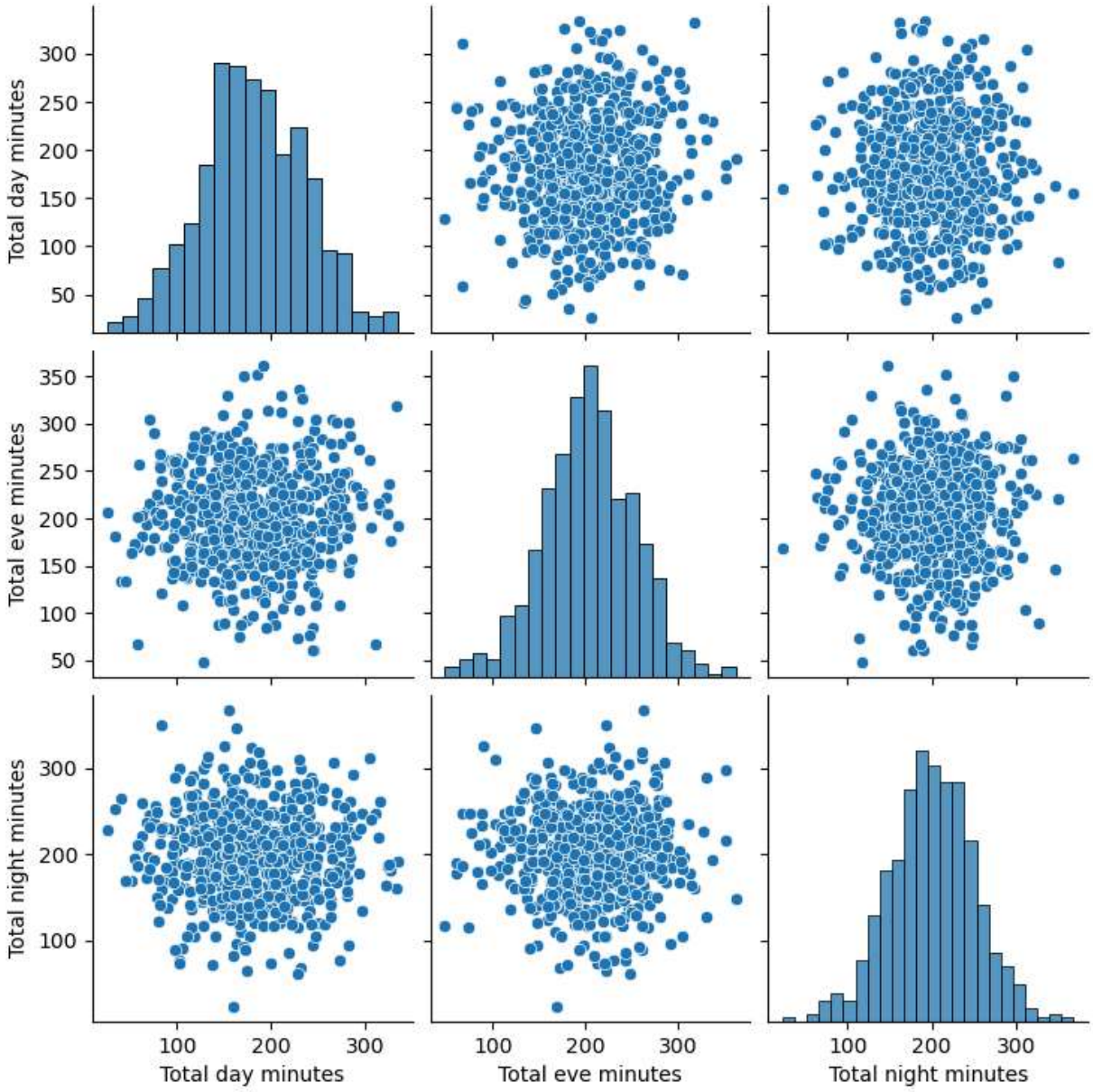


In [18]:
```
sns.boxplot(x='Churn', y='Total day minutes', data=df)
plt.title("Total Day Minutes by Churn")
plt.show()
```



In [19]:
```
sns.pairplot(df[['Total day minutes', 'Total eve minutes', 'Total night minutes']
plt.show()
```

```
C:\Users\user\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



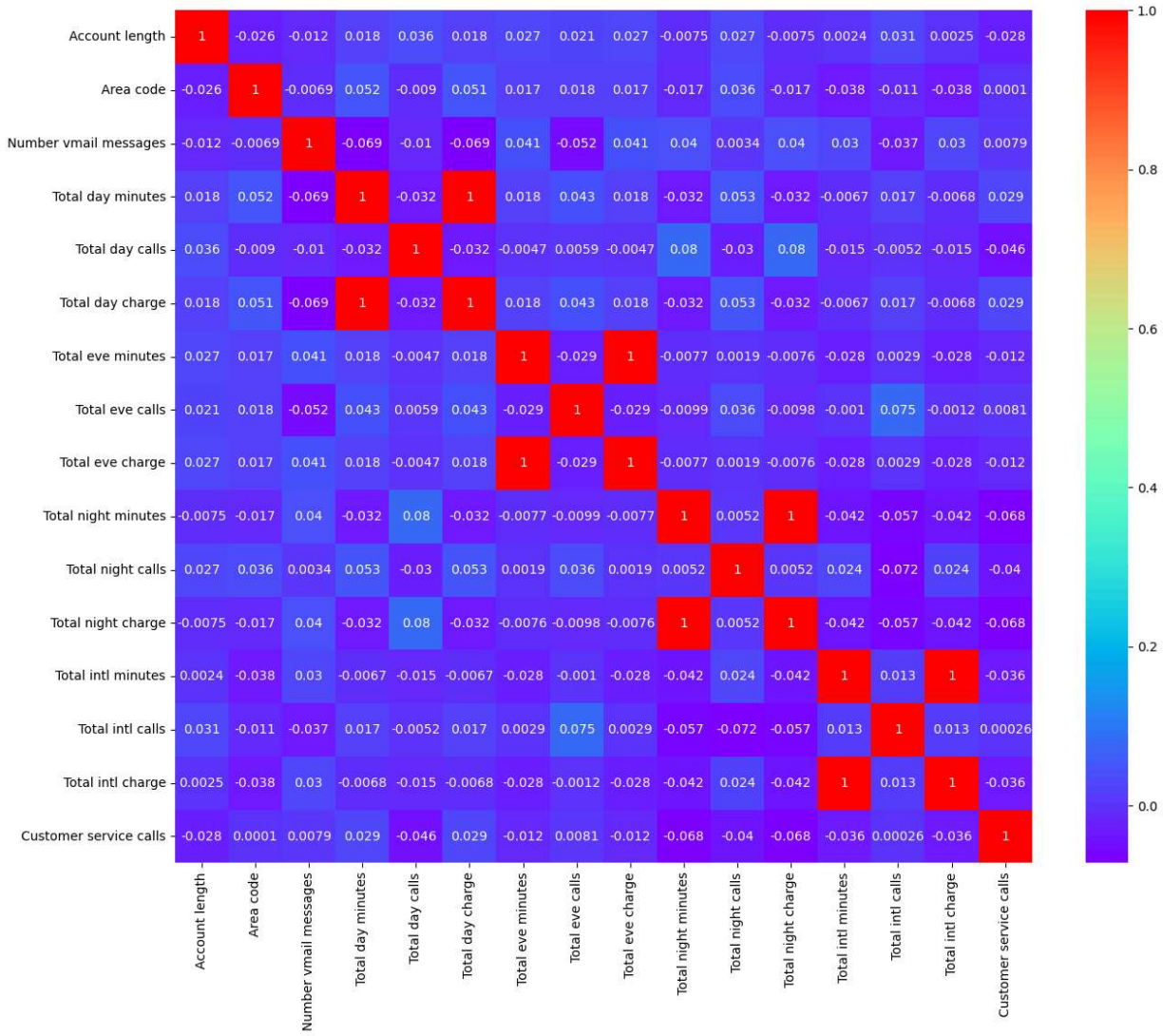
In [20]:
```
num_data=df.select_dtypes(include=["int64","float64"])
corr_matrix=num_data.corr()
corr_matrix
```

Out[20]:

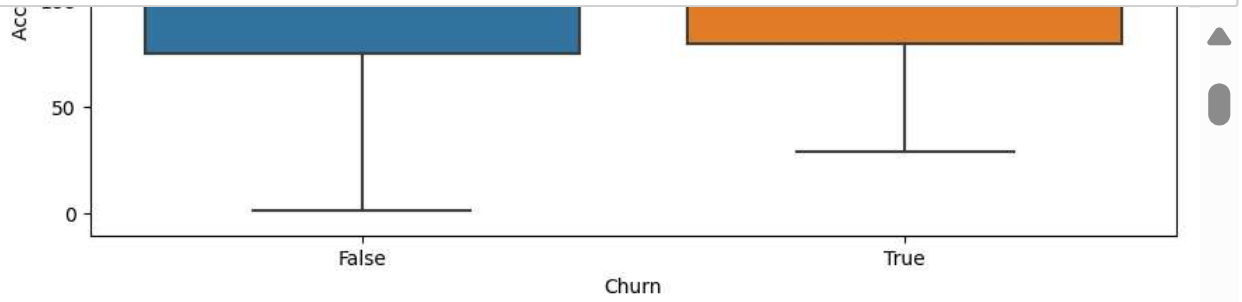
	Account length	Area code	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	T
Account length	1.000000	-0.026327	-0.011993	0.017833	0.035703	0.017839	0.027043	0.021237	0
Area code	-0.026327	1.000000	-0.006907	0.051507	-0.008972	0.051492	0.017160	0.017783	0
Number vmail messages	-0.011993	-0.006907	1.000000	-0.069172	-0.009952	-0.069187	0.040865	-0.051951	0
Total day minutes	0.017833	0.051507	-0.069172	1.000000	-0.032306	1.000000	0.017987	0.043219	0
Total day calls	0.035703	-0.008972	-0.009952	-0.032306	1.000000	-0.032319	-0.004688	0.005851	-0
Total day charge	0.017839	0.051492	-0.069187	1.000000	-0.032319	1.000000	0.017983	0.043231	0
Total eve minutes	0.027043	0.017160	0.040865	0.017987	-0.004688	0.017983	1.000000	-0.029077	1
Total eve calls	0.021237	0.017783	-0.051951	0.043219	0.005851	0.043231	-0.029077	1.000000	-0
Total eve charge	0.027051	0.017182	0.040876	0.017945	-0.004664	0.017941	1.000000	-0.029089	1
Total night minutes	-0.007527	-0.016832	0.039751	-0.031600	0.079536	-0.031613	-0.007705	-0.009856	-0
Total night calls	0.027228	0.036421	0.003367	0.052761	-0.030074	0.052748	0.001938	0.036068	0
Total night charge	-0.007528	-0.016818	0.039680	-0.031603	0.079529	-0.031616	-0.007603	-0.009833	-0
Total intl minutes	0.002362	-0.037980	0.029949	-0.006725	-0.015319	-0.006720	-0.027855	-0.001050	-0
Total intl calls	0.031279	-0.010530	-0.036847	0.016597	-0.005155	0.016582	0.002929	0.074829	0
Total intl charge	0.002456	-0.038044	0.029999	-0.006841	-0.015201	-0.006836	-0.027887	-0.001152	-0
Customer service calls	-0.027677	0.000103	0.007859	0.029291	-0.045953	0.029290	-0.012213	0.008126	-0

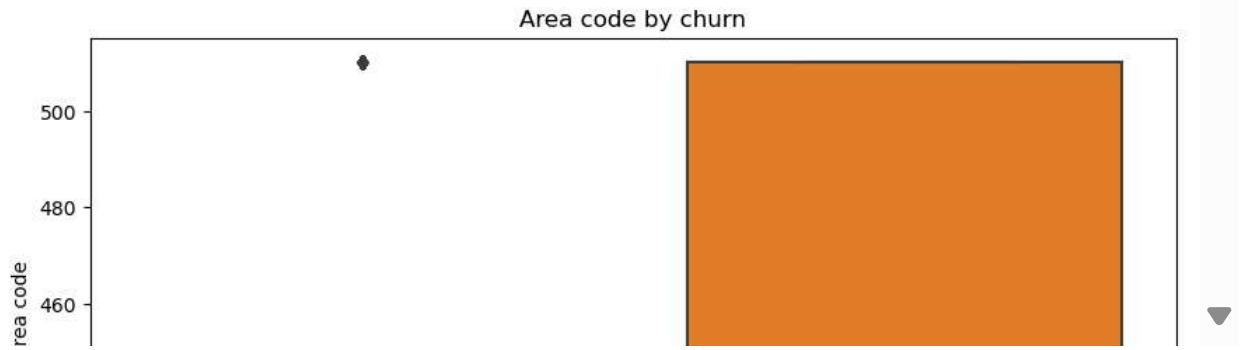
In [21]:
```
plt.figure(figsize=(15,12))
sns.heatmap(corr_matrix,annot=True,cmap="rainbow")
```

Out[21]: <Axes: >



In [23]:
```
for col in num_data:
    plt.figure(figsize=(10,5))
    sns.boxplot(x="Churn",y=col,data=df)
    plt.title(f'{col} by churn')
```





In [24]:
```
obj_col=df.select_dtypes(include=["object","bool"])
```

In [25]:
```
df['International plan'] = df['International plan'].astype(str).str.lower().str.s
df['Voice mail plan']    = df['Voice mail plan'].astype(str).str.lower().str.str
df['Churn']              = df['Churn'].astype(str).str.lower().str.strip().map({
df
```

Out[25]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls
0	LA	117	408	0	0	0	184.5	97	31.37	351.6	80
1	IN	65	415	0	0	0	129.1	137	21.95	228.5	83
2	NY	161	415	0	0	0	332.9	67	56.59	317.8	97
3	SC	111	415	0	0	0	110.4	103	18.77	137.3	102
4	HI	49	510	0	0	0	119.3	117	20.28	215.1	109
...	...	...	...	...	...	...	...	...	...	...	...
662	WI	114	415	0	1	26	137.1	88	23.31	155.7	125
663	AL	106	408	0	1	29	83.6	131	14.21	203.9	131
664	VT	60	415	0	0	0	193.9	118	32.96	85.0	110
665	WV	159	415	0	0	0	169.8	114	28.87	197.7	105
666	CT	184	510	1	0	0	213.8	105	36.35	159.6	84

667 rows × 20 columns

In [40]:
```
x = df.drop(['Churn',"State"], axis=1)
y = df['Churn']
```

In [41]:
```
encoder=LabelEncoder()
df["State"]=encoder.fit_transform(df["State"])
```

In [42]:
```
from sklearn.preprocessing import StandardScaler,LabelEncoder
```

In [44]:
```
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
```

In [45]:
```
from imblearn.over_sampling import SMOTE
```

In [47]:
```
sm=SMOTE()
x_resampled,y_resampled=sm.fit_resample(x_scaled,y)
```

In [54]:
```
from sklearn.feature_selection import SelectKBest
selector = SelectKBest( k=10)
x_selected = selector.fit_transform(x_resampled, y_resampled)
```

In [55]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(
    x_selected, y_resampled, test_size=0.2, random_state=42, stratify=y_resampled
)
```

In [57]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
```

In [59]:
```
rf = RandomForestClassifier(random_state=42)
rf.fit(x_train, y_train)
y_pred = rf.predict(x_test)
```

In [60]:
```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9432314410480349
Confusion Matrix:
 [[106   9]
 [  4 110]]
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.92      0.94       115
           1       0.92      0.96      0.94       114

    accuracy                           0.94       229
   macro avg       0.94      0.94      0.94       229
weighted avg       0.94      0.94      0.94       229
```

In [66]:
```
lr=LogisticRegression()
lr.fit(x_train,y_train)
y_pred_lr=lr.predict(x_test)
```

In [67]:
```
print("Accuracy:", accuracy_score(y_test, y_pred_lr))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_lr))
print("Classification Report:\n", classification_report(y_test, y_pred_lr))
```

```
Accuracy: 0.851528384279476
Confusion Matrix:
 [[ 95  20]
 [ 14 100]]
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.83      0.85       115
           1       0.83      0.88      0.85       114

    accuracy                           0.85       229
   macro avg       0.85      0.85      0.85       229
weighted avg       0.85      0.85      0.85       229
```

In [68]:
```
svc=SVC()
svc.fit(x_train,y_train)
y_pred_svc=svc.predict(x_test)
```

In [69]:
```
print("Accuracy:", accuracy_score(y_test, y_pred_svc))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_svc))
print("Classification Report:\n", classification_report(y_test, y_pred_svc))
```

```
Accuracy: 0.9519650655021834
Confusion Matrix:
 [[106   9]
 [  2 112]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.92      0.95       115
           1       0.93      0.98      0.95       114

    accuracy                           0.95       229
   macro avg       0.95      0.95      0.95       229
weighted avg       0.95      0.95      0.95       229
```

In [ ]:
```

```