In [1]:
```python
import pandas as pd
import numpy as np
from prophet import Prophet
import statsmodels.api as sm
import matplotlib.pyplot as plt
from statsmodels.tsa.tsatools import freq_to_period
from statsmodels.tsa.tsatools import freq_to_period
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.graphics.tsaplots import plot_pacf
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
import seaborn as sns
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [2]:
```python
data = pd.read_csv("daily-website-visitors.csv")
data.head()
```

Out[2]:

	Row	Day	Day.Of.Week	Date	Page.Loads	Unique.Visits	First.Time.Visits	Returning.Vi
0	1	Sunday	1	9/14/2014	2,146	1,582	1,430	
1	2	Monday	2	9/15/2014	3,621	2,528	2,297	
2	3	Tuesday	3	9/16/2014	3,698	2,630	2,352	
3	4	Wednesday	4	9/17/2014	3,667	2,614	2,327	
4	5	Thursday	5	9/18/2014	3,316	2,366	2,130	

In [3]:
```python
data = data.rename(columns={"Date": "ds", "Page.Loads": "y"})
```

In [4]:
```python
data.head()
```

Out[4]:

	Row	Day	Day.Of.Week	ds	y	Unique.Visits	First.Time.Visits	Returning.Visits
0	1	Sunday	1	9/14/2014	2,146	1,582	1,430	152
1	2	Monday	2	9/15/2014	3,621	2,528	2,297	231
2	3	Tuesday	3	9/16/2014	3,698	2,630	2,352	278
3	4	Wednesday	4	9/17/2014	3,667	2,614	2,327	287
4	5	Thursday	5	9/18/2014	3,316	2,366	2,130	236

In [5]:

```
data.describe()
```

Out[5]:

	Row	Day.Of.Week
count	2167.000000	2167.000000
mean	1084.000000	3.997231
std	625.703338	2.000229
min	1.000000	1.000000
25%	542.500000	2.000000
50%	1084.000000	4.000000
75%	1625.500000	6.000000
max	2167.000000	7.000000

In [6]:

```
data.columns
```

Out[6]:

```
Index(['Row', 'Day', 'Day.Of.Week', 'ds', 'y', 'Unique.Visits',
       'First.Time.Visits', 'Returning.Visits'],
      dtype='object')
```

In [7]:

```
data.isnull ().sum ()
```

Out[7]:

```
Row                  0
Day                  0
Day.Of.Week          0
ds                   0
y                    0
Unique.Visits        0
First.Time.Visits    0
Returning.Visits     0
dtype: int64
```

In [8]:

```
data['ds'] = pd.to_datetime(data['ds'])
```

In [9]:

```
data['y'] = data['y'].str.replace(',', '').astype(float)
```

In [10]:

```
missing_values = data.isnull().sum()
```

In [11]:

```
data.fillna(method='ffill', inplace=True)
```

In [12]:
```
model = Prophet()
model.fit(data)
```

```
18:47:53 - cmdstanpy - INFO - Chain [1] start processing
18:47:54 - cmdstanpy - INFO - Chain [1] done processing
```

Out[12]:
```
<prophet.forecaster.Prophet at 0x2116eac8790>
```

In [13]:
```
future = model.make_future_dataframe(periods=365)
```

In [14]:
```
forecast = model.predict(future)
```

In [15]:
```
print(data.head())
```

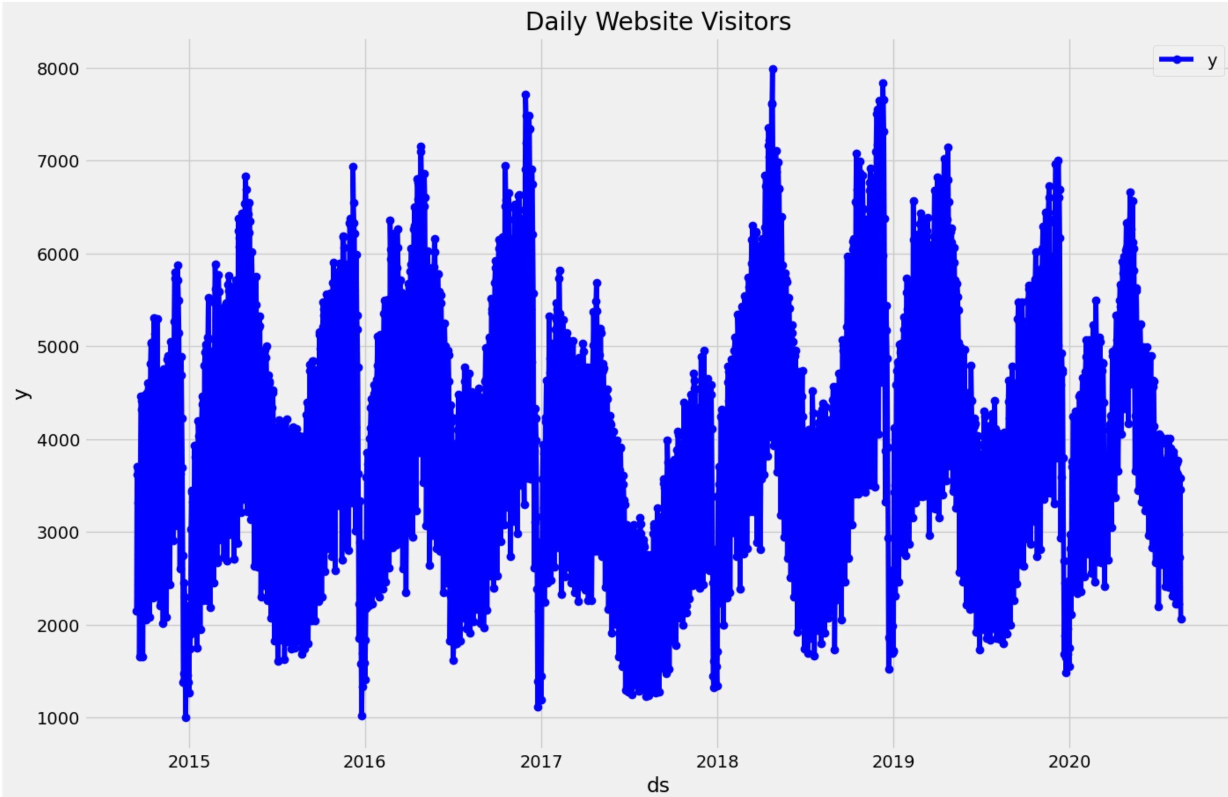
```
   Row        Day  Day.Of.Week         ds       y Unique.Visits  \
0    1     Sunday            1 2014-09-14  2146.0         1,582
1    2     Monday            2 2014-09-15  3621.0         2,528
2    3    Tuesday            3 2014-09-16  3698.0         2,630
3    4  Wednesday            4 2014-09-17  3667.0         2,614
4    5   Thursday            5 2014-09-18  3316.0         2,366

  First.Time.Visits Returning.Visits
0             1,430              152
1             2,297              231
2             2,352              278
3             2,327              287
4             2,130              236
```

In [16]:
```
data["ds"] = pd.to_datetime(data["ds"], format="%y/%m/%d")
print(data.info())
```

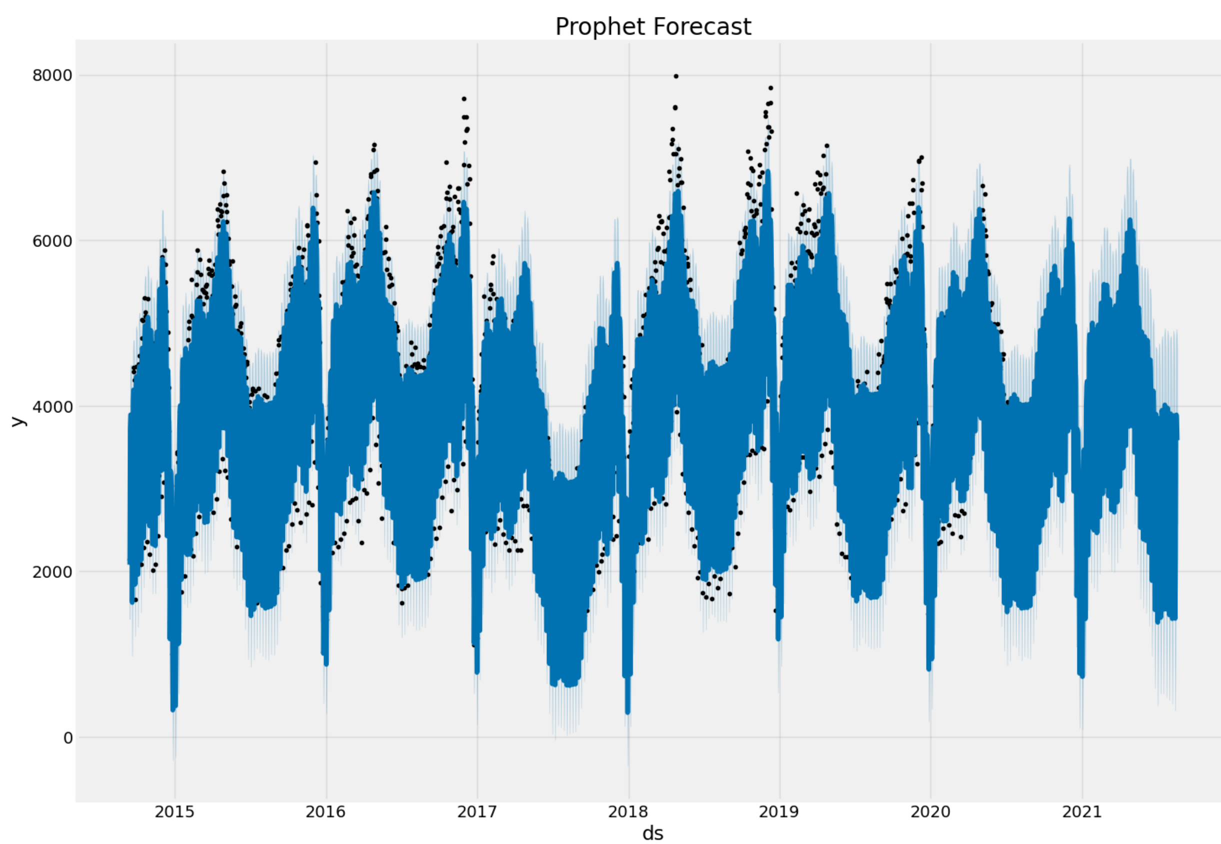
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2167 entries, 0 to 2166
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Row                2167 non-null   int64
 1   Day                2167 non-null   object
 2   Day.Of.Week        2167 non-null   int64
 3   ds                 2167 non-null   datetime64[ns]
 4   y                  2167 non-null   float64
 5   Unique.Visits      2167 non-null   object
 6   First.Time.Visits  2167 non-null   object
 7   Returning.Visits   2167 non-null   object
dtypes: datetime64[ns](1), float64(1), int64(2), object(4)
memory usage: 135.6+ KB
None
```

In [17]:
```
plt.style.use('fivethirtyeight')
plt.figure(figsize=(15, 10))
plt.plot(data['ds'], data['y'], marker='o', linestyle='-', color='b', label='y')
plt.title("Daily Website Visitors")
plt.xlabel("ds")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.show()
```



In [18]:

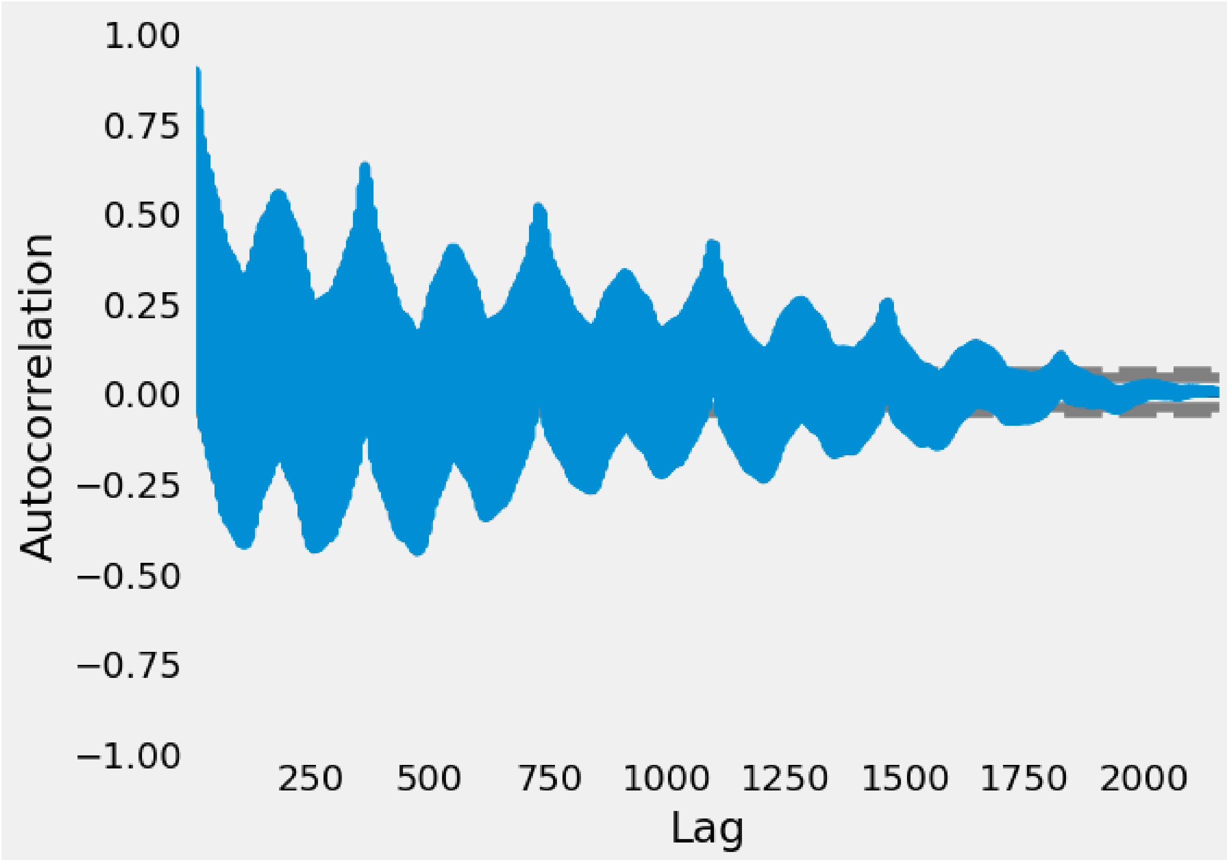
```
future = model.make_future_dataframe(periods=365)
forecast = model.predict(future)
fig = model.plot(forecast, figsize=(15, 10))
plt.title("Prophet Forecast")
plt.show()
```



In [19]:

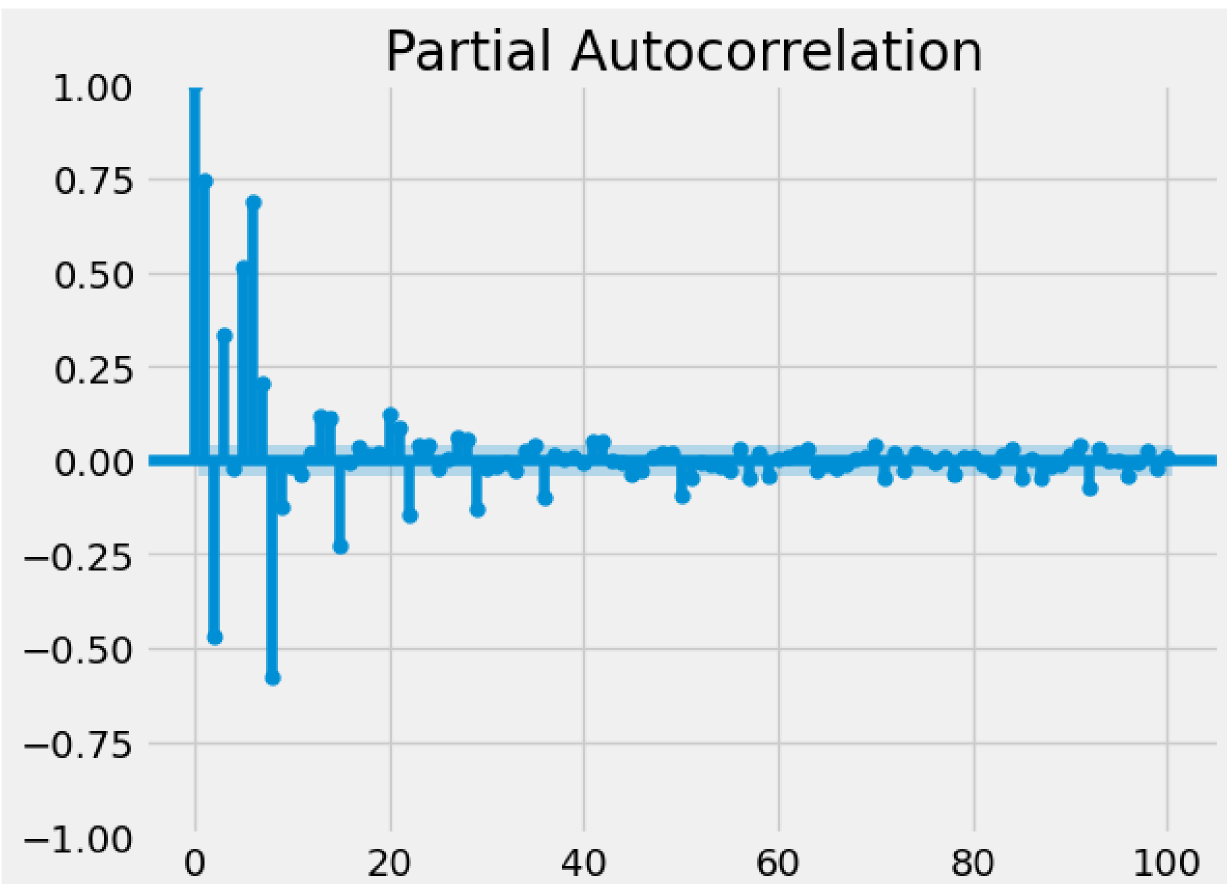
```
pd.plotting.autocorrelation_plot(data["y"])
```

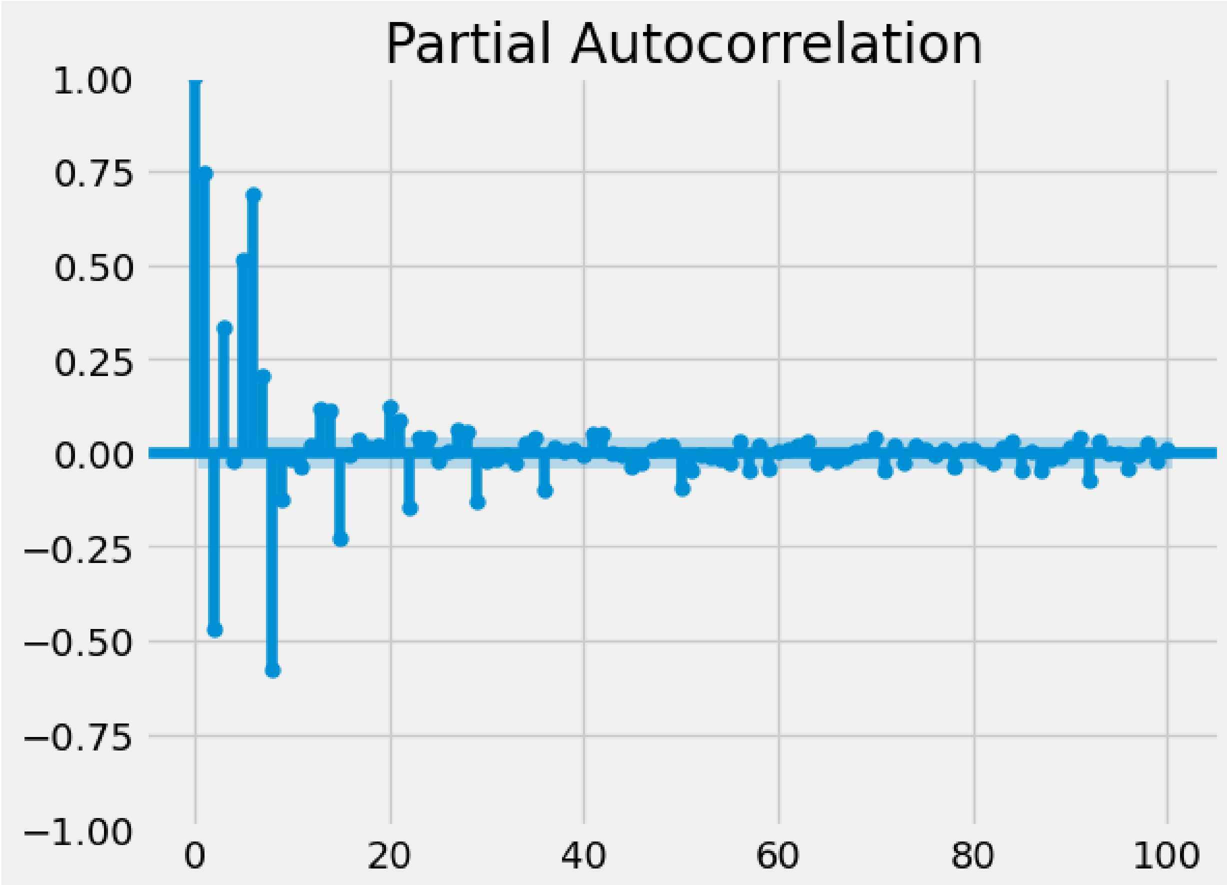
Out[19]: <Axes: xlabel='Lag', ylabel='Autocorrelation'>



In [20]:
```python
plot_pacf(data["y"], lags = 100)
```

Out[20]:





In [21]:

```
p, d, q = 5, 1, 2
P, D, Q, s = 0, 0, 0, 12
model = sm.tsa.SARIMAX(data['y'], order=(p, d, q), seasonal_order=(P, D, Q, s))
results = model.fit()
print(results.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 2167
Model:               SARIMAX(5, 1, 2)   Log Likelihood              -16087.041
Date:                Wed, 01 Nov 2023   AIC                          32190.082
Time:                        18:48:04   BIC                          32235.527
Sample:                             0   HQIC                         32206.701
                               - 2167
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.0950      0.024      3.963      0.000       0.048       0.142
ar.L2         -0.8710      0.018    -47.730      0.000      -0.907      -0.835
ar.L3         -0.2317      0.031     -7.394      0.000      -0.293      -0.170
ar.L4         -0.4125      0.020    -20.832      0.000      -0.451      -0.374
ar.L5         -0.6798      0.024    -28.156      0.000      -0.727      -0.632
ma.L1         -0.2394      0.025     -9.567      0.000      -0.288      -0.190
ma.L2          0.4957      0.023     21.328      0.000       0.450       0.541
sigma2      1.834e+05   5277.134     34.754      0.000    1.73e+05    1.94e+05
===================================================================================
Ljung-Box (L1) (Q):                   4.69   Jarque-Bera (JB):               232.93
Prob(Q):                              0.03   Prob(JB):                         0.00
Heteroskedasticity (H):               1.08   Skew:                            -0.05
Prob(H) (two-sided):                  0.32   Kurtosis:                         4.60
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

```
D:\AnacondaNavigator\Lib\site-packages\statsmodels\base\model.py:607: ConvergenceWarn
ing: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

In [22]:

```
forecast = results.get_forecast(steps=50)
forecast_mean = forecast.predicted_mean
print(forecast_mean)
```

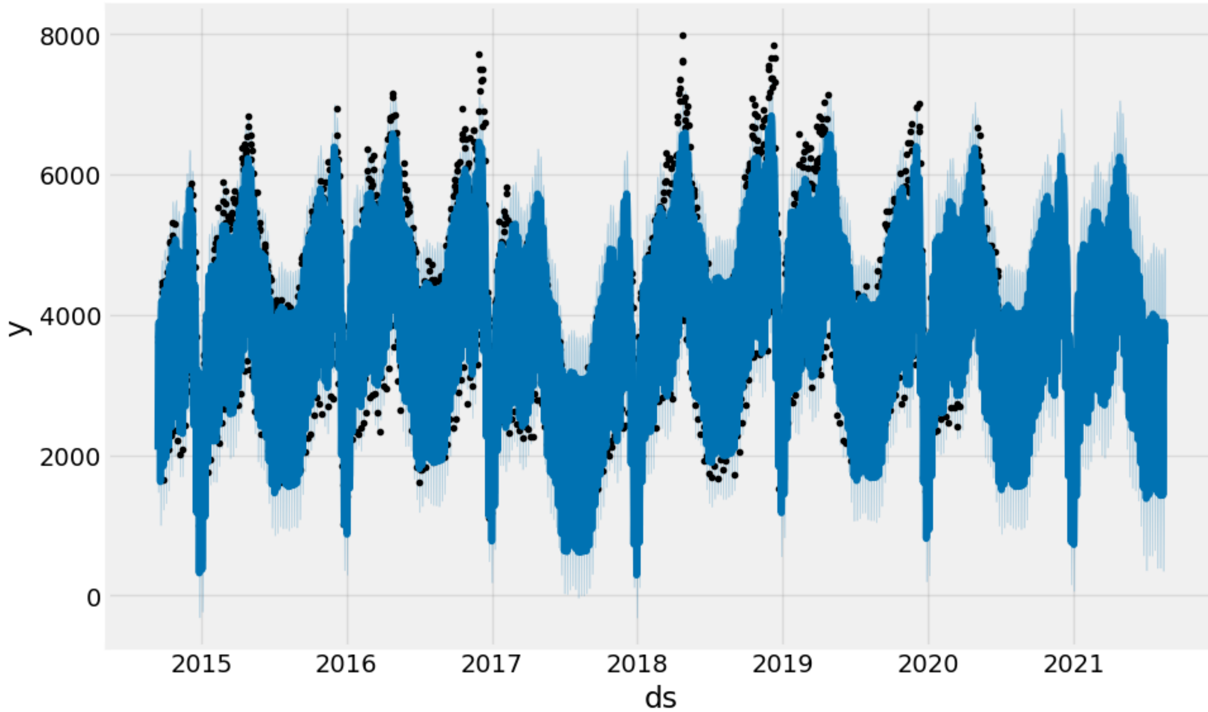
```
2167    2235.851049
2168    2164.469489
2169    1810.307491
2170    2339.773713
2171    3675.419869
2172    3335.827578
2173    2212.169428
2174    2114.116272
2175    2251.332728
2176    1842.233998
2177    2400.898753
2178    3582.785901
2179    3313.314039
2180    2204.336182
2181    2107.526928
2182    2259.403331
2183    1922.811378
2184    2421.586393
2185    3520.742148
2186    3271.883130
2187    2210.924261
2188    2095.305298
2189    2273.631321
2190    1992.546813
2191    2444.096936
2192    3459.408217
2193    3232.732676
2194    2216.974827
2195    2087.500074
2196    2286.686781
2197    2057.025577
2198    2464.777661
2199    3401.294334
2200    3194.181232
2201    2223.666909
2202    2082.816163
2203    2299.256923
2204    2116.154985
2205    2483.977936
2206    3346.087925
2207    3156.513192
2208    2230.789910
2209    2080.977166
2210    2311.324462
2211    2170.318100
2212    2501.706783
2213    3293.717375
2214    3119.820164
2215    2238.266568
2216    2081.648427
Name: predicted_mean, dtype: float64
```

In [23]:

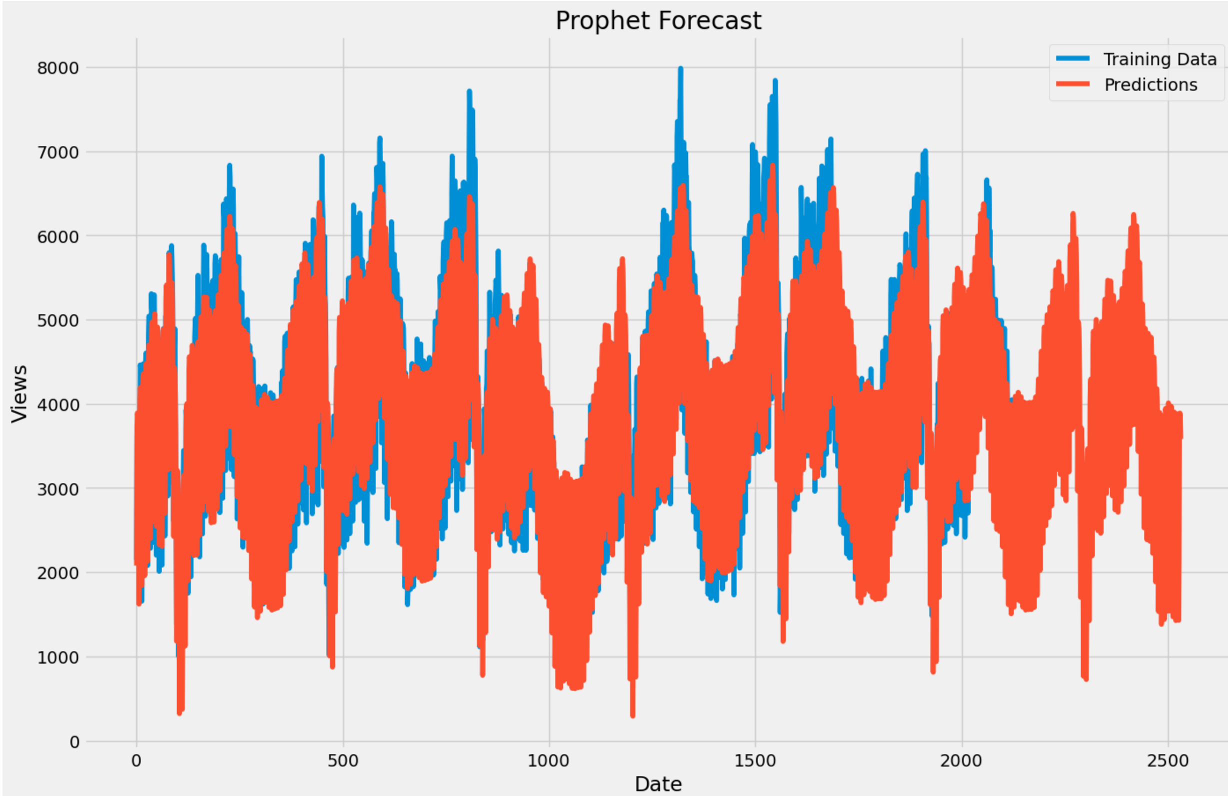
```
data = data.rename(columns={"Date": "ds", "Page.Loads": "y"})
data['ds'] = pd.to_datetime(data['ds'])
if data['y'].dtype == object:
    data['y'] = data['y'].str.replace(',', '').astype(float)
data = data.dropna(subset=['y'])
if len(data) < 2:
    print("")
else:
```

```
    model = Prophet()
    model.fit(data)
    future = model.make_future_dataframe(periods=365)
    forecast = model.predict(future)
    fig = model.plot(forecast)
```

```
18:48:05 - cmdstanpy - INFO - Chain [1] start processing
18:48:05 - cmdstanpy - INFO - Chain [1] done processing
```



In [24]:
```
data["y"].plot(legend=True, label="Training Data", figsize=(15, 10))
forecast["yhat"].plot(legend=True, label="Predictions")
plt.title("Prophet Forecast")
plt.xlabel("Date")
plt.ylabel("Views")
plt.show()
```



In [25]:
```
def time_series_analysis(data):
    result = seasonal_decompose(data['Page.Loads'], model='additive')
    result.plot()
    plt.show()
    p, d, q, P, D, Q, s = 5, 1, 2, 0, 0, 0, 12
    model = SARIMAX(data['Page.Loads'], order=(p, d, q), seasonal_order=(P, D, Q, s))
    results = model.fit()
    forecast = results.get_forecast(steps=365)
    forecast_mean = forecast.predicted_mean
    return forecast_mean
```

In [26]:
```
def user_segmentation(data):
    # Select relevant features for segmentation
    user_data = data[['feature1', 'feature2', 'feature3']]

    # Normalize data if needed
    user_data_normalized = (user_data - user_data.mean()) / user_data.std()

    # Apply K-Means clustering
    num_clusters = 3  # Choose the number of clusters
    kmeans = KMeans(n_clusters=num_clusters)
    data['Cluster'] = kmeans.fit_predict(user_data_normalized)

    return data
```

In [27]:
```
def machine_learning_predictions(data):
    X = data[['feature1', 'feature2', 'feature3']]
    y = data['Page.Loads']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st
    model = RandomForestRegressor()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
```

```
    mse = mean_squared_error(y_test, y_pred)
    return mse
```

In [28]:
```
def data_visualization(data):
 def ab_testing(data):
  def custom_analysis(data):
    forecast = time_series_analysis(data)
    segmented_data = user_segmentation(data)
    mse = machine_learning_predictions(data)
    data_visualization(data)
    ab_testing(data)
    custom_analysis(data)
```

In [29]:
```
data.columns
```

Out[29]:
```
Index(['Row', 'Day', 'Day.Of.Week', 'ds', 'y', 'Unique.Visits',
       'First.Time.Visits', 'Returning.Visits'],
      dtype='object')
```

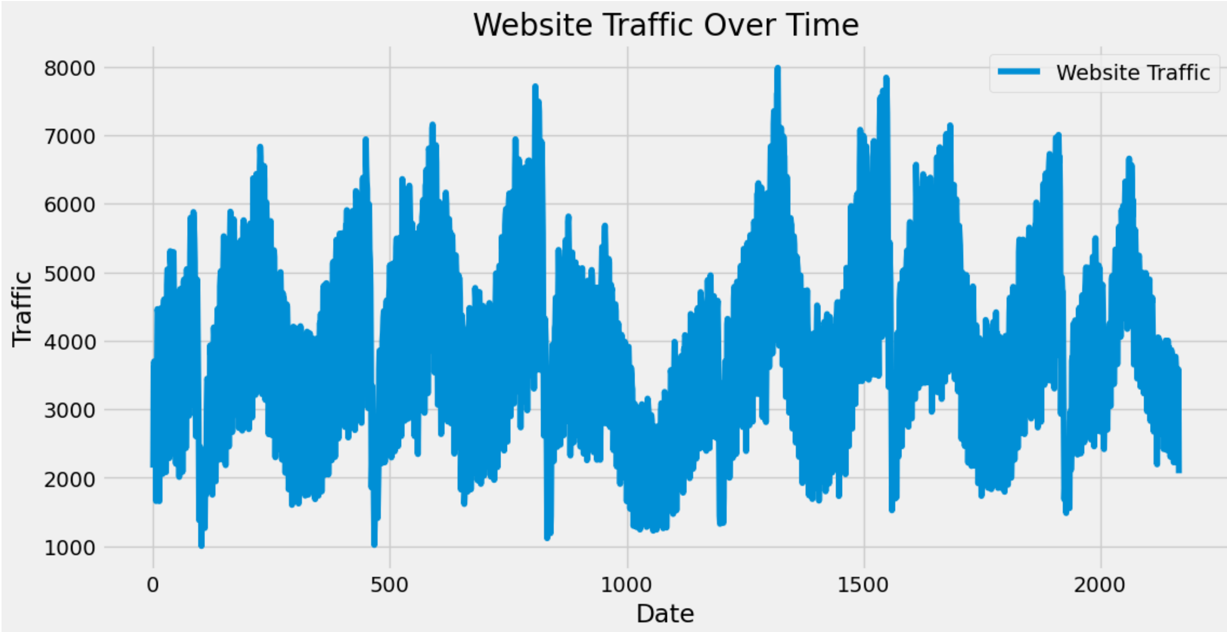
In [30]:
```
print(data.head())
```

```
   Row        Day  Day.Of.Week          ds       y Unique.Visits  \
0    1     Sunday            1 2014-09-14  2146.0         1,582   
1    2     Monday            2 2014-09-15  3621.0         2,528   
2    3    Tuesday            3 2014-09-16  3698.0         2,630   
3    4  Wednesday            4 2014-09-17  3667.0         2,614   
4    5   Thursday            5 2014-09-18  3316.0         2,366   

  First.Time.Visits Returning.Visits  
0             1,430              152  
1             2,297              231  
2             2,352              278  
3             2,327              287  
4             2,130              236  
```

In [31]:
```
data.reset_index(inplace=True)
data['ds'] = pd.to_datetime(data['ds'])
```

In [32]:
```
data['ds'] = pd.to_datetime(data['ds'])
daily_data = data.resample('D', on='ds').sum()
plt.figure(figsize=(12, 6))
plt.plot(daily_data.iloc[:, 0], daily_data['y'], label='Website Traffic')  # Use iloc
plt.title('Website Traffic Over Time')
plt.xlabel('Date')
plt.ylabel('Traffic')
plt.legend()
plt.show()
```



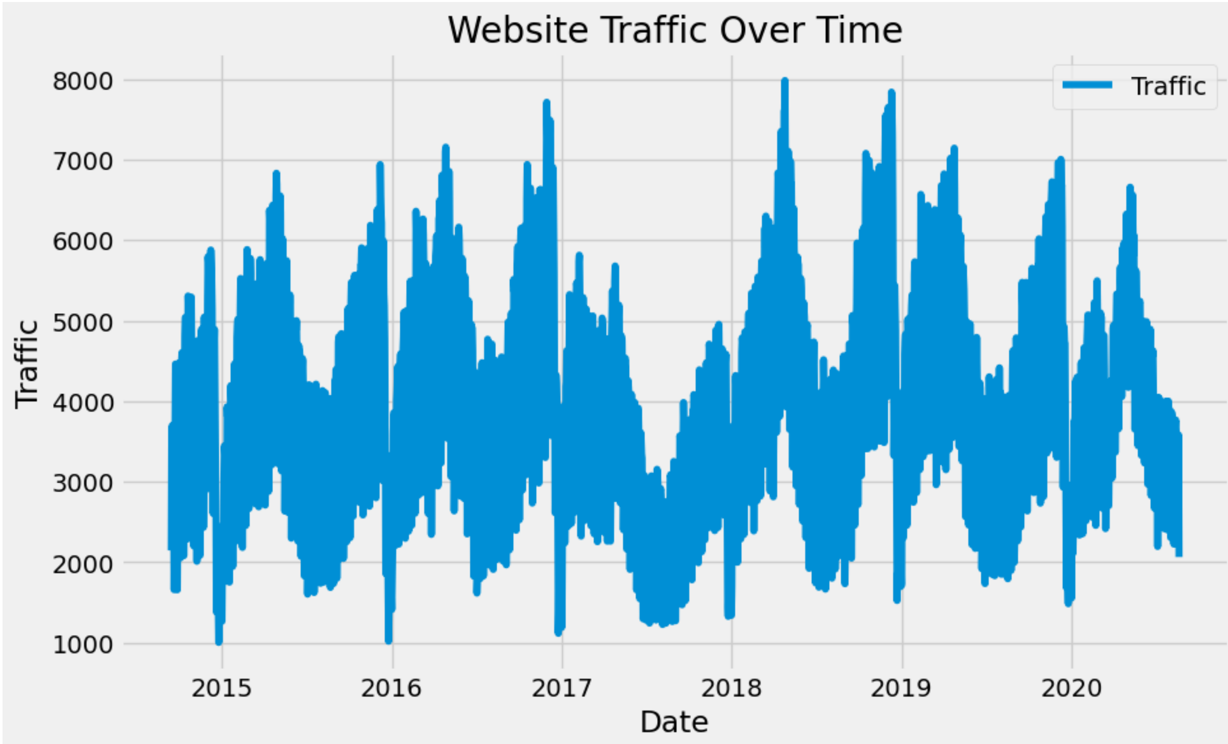
In [33]:
```
user_data = pd.read_csv("daily-website-visitors.csv")
numeric_columns = user_data.select_dtypes(include=[float, int])
scaler = StandardScaler()
user_data_scaled = scaler.fit_transform(numeric_columns)
kmeans = KMeans(n_clusters=3)
user_data['Cluster'] = kmeans.fit_predict(user_data_scaled)
```

In [34]:
```
print(data.columns)
```

```
Index(['index', 'Row', 'Day', 'Day.Of.Week', 'ds', 'y', 'Unique.Visits',
       'First.Time.Visits', 'Returning.Visits'],
      dtype='object')
```

In [35]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
X = data[['index', 'Row', 'Day', 'Day.Of.Week', 'ds', 'Unique.Visits', 'First.Time.Vis
y = data['y']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [36]:
```
plt.figure(figsize=(10, 6))
plt.plot(data['ds'], data['y'], label='Traffic')
plt.xlabel('Date')
plt.ylabel('Traffic')
plt.legend()
plt.title('Website Traffic Over Time')
plt.show()
```



In [37]:
```
group_a_data = data[data['Day.Of.Week'] == 1]['y']
group_b_data = data[data['Day.Of.Week'] == 1]['y']
result = stats.ttest_ind(group_a_data, group_b_data)
if result.pvalue < 0.05:
    print("Statistically significant difference")
else:
    print("No statistically significant difference")
```

```
No statistically significant difference
```

In [ ]: