Website Traffic Forecasting

Website Traffic Forecasting means forecasting traffic on a website during a particular period. It is one of the best use cases of Time Series Forecasting.

It contains data about daily traffic data from June 2021 to June 2022.

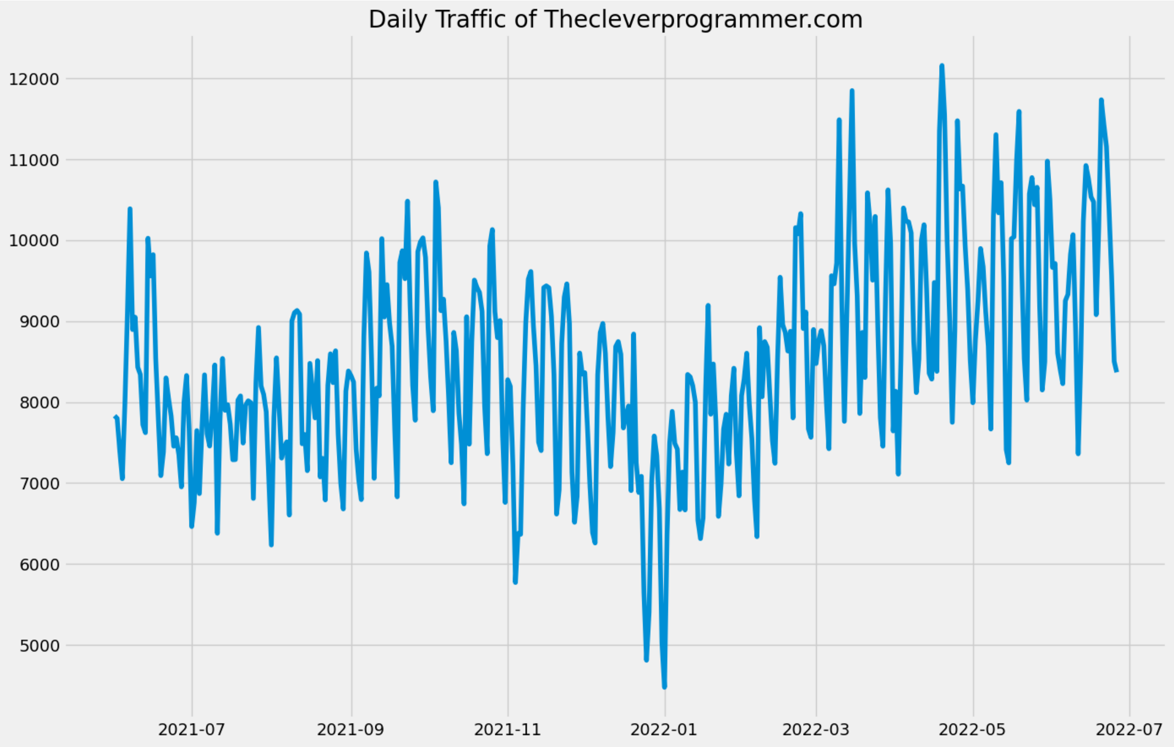
In [1]:
```
#let’s get started with the task of website traffic forecasting by importing the
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.tsa.arima_model import ARIMA
import statsmodels.api as sm

data = pd.read_csv("Thecleverprogrammer.csv")
print(data.head())
```

```
         Date  Views
0  01/06/2021   7831
1  02/06/2021   7798
2  03/06/2021   7401
3  04/06/2021   7054
4  05/06/2021   7973
```

In [2]:
```
#I will convert the Date column into Datetime data type:
data["Date"] = pd.to_datetime(data["Date"],
                              format="%d/%m/%Y")
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 391 entries, 0 to 390
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    391 non-null    datetime64[ns]
 1   Views   391 non-null    int64
dtypes: datetime64[ns](1), int64(1)
memory usage: 6.2 KB
None
```

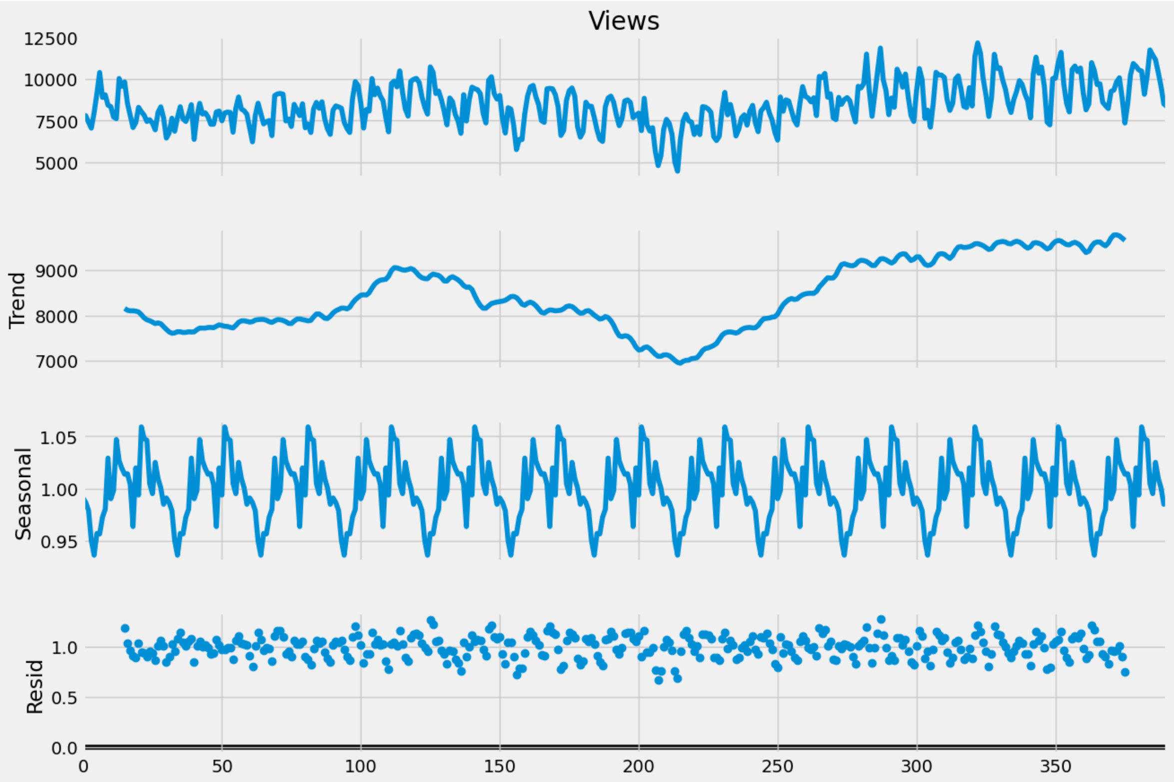
In [3]:
```
#let’s have a look at the daily traffic of the website:
plt.style.use('fivethirtyeight')
plt.figure(figsize=(15, 10))
plt.plot(data["Date"], data["Views"])
plt.title("Daily Traffic of Thecleverprogrammer.com")
plt.show()
```



Data is seasonal because the traffic on the website increases during the weekdays and decreases during the weekends.

In [6]:
```
#look at whether our dataset is stationary or seasonal:
from statsmodels.tsa.seasonal import seasonal_decompose

result = seasonal_decompose(data["Views"], model='multiplicative', period=30)
fig = plt.figure()
fig = result.plot()
fig.set_size_inches(15, 10)
```

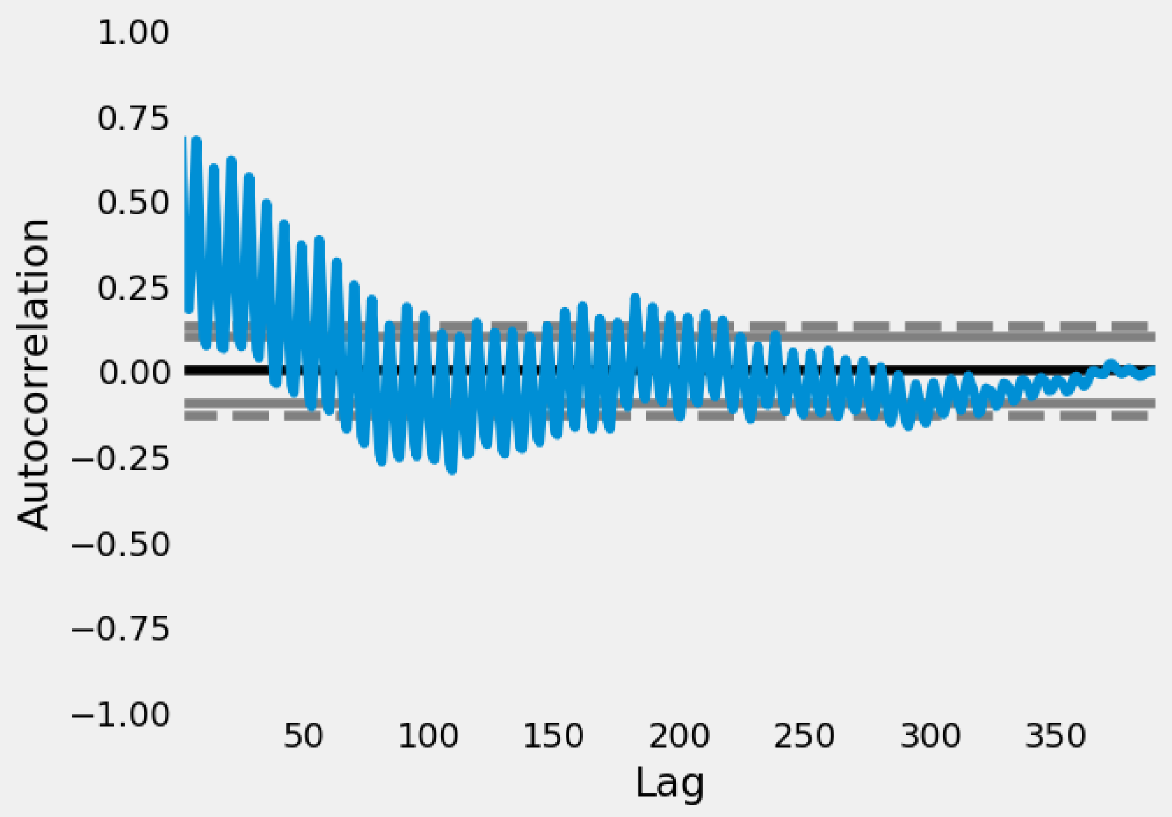
```
<Figure size 640x480 with 0 Axes>
```



I will be using the Seasonal ARIMA (SARIMA) model to forecast traffic on the website. Before using the SARIMA model, it is necessary to find the p, d, and q values.

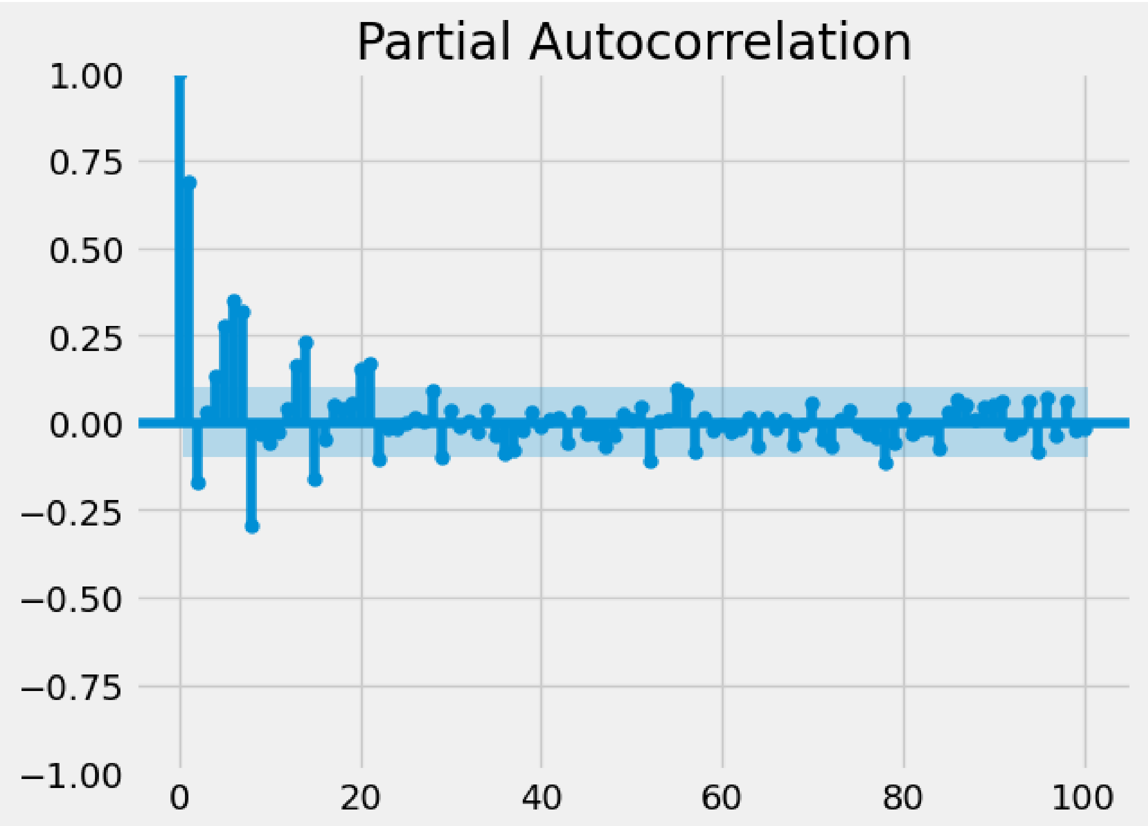
In [7]:
```
#The values of p and q, we can use the autocorrelation and partial autocorrelati
pd.plotting.autocorrelation_plot(data["Views"])
```

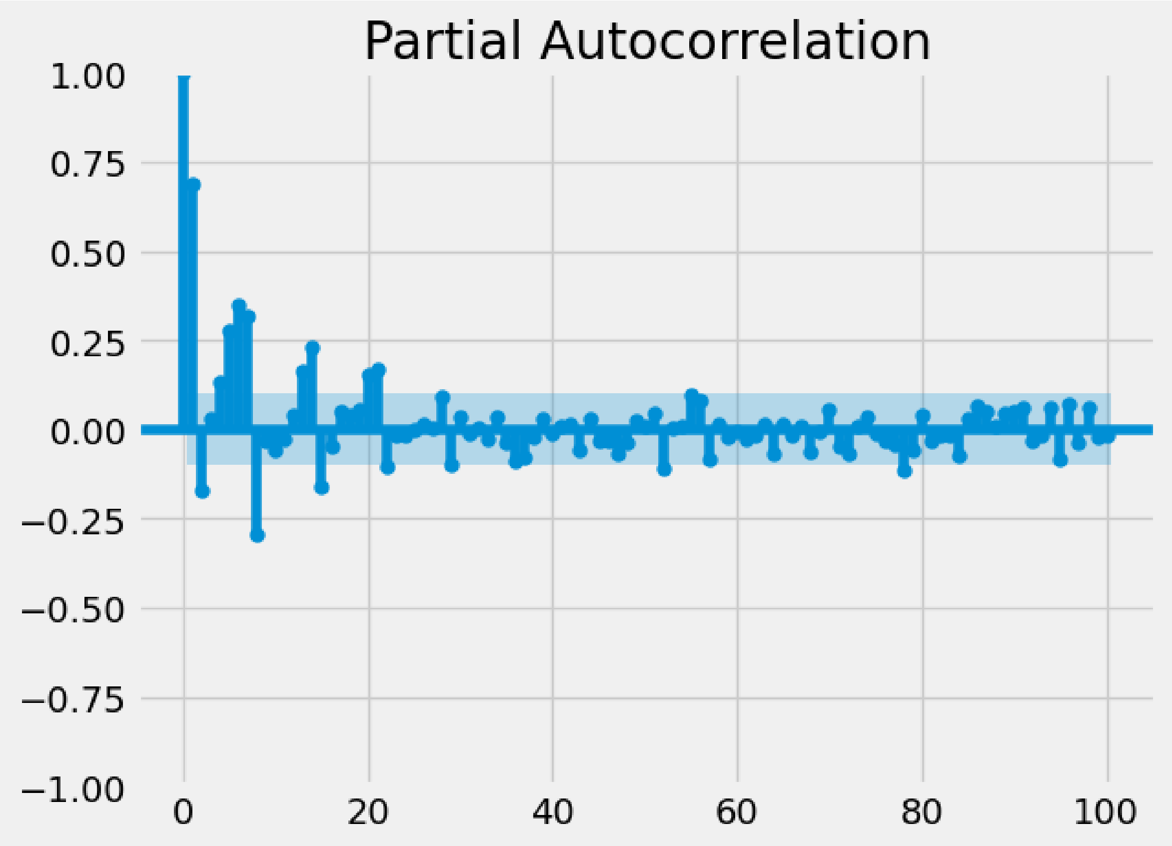
Out[7]: <Axes: xlabel='Lag', ylabel='Autocorrelation'>



In [8]:
```
plot_pacf(data["Views"], lags = 100)
```

Out[8]:





In [12]:

```
p, d, q = 5, 1, 2
model=sm.tsa.statespace.SARIMAX(data['Views'],
                                order=(p, d, q),
                                seasonal_order=(p, d, q, 12))
model=model.fit()
print(model.summary())
```

```
C:\Users\Sethu\AppData\Local\Programs\Python\Python311\Lib\site-packages\statsmod
els\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible starting MA parame
ters found. Using zeros as starting parameters.
  warn('Non-invertible starting MA parameters found.'
C:\Users\Sethu\AppData\Local\Programs\Python\Python311\Lib\site-packages\statsmod
els\base\model.py:607: ConvergenceWarning: Maximum Likelihood optimization failed
to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

```
                                     SARIMAX Results
==========================================================================================
=========
Dep. Variable:                                 Views   No. Observations:
391
Model:             SARIMAX(5, 1, 2)x(5, 1, 2, 12)   Log Likelihood
-3099.430
Date:                             Mon, 10 Jul 2023   AIC
6228.861
Time:                                     19:27:06   BIC
6287.884
Sample:                                          0   HQIC
6252.286
                                             - 391
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7804      0.134      5.822      0.000       0.518       1.043
ar.L2         -0.7983      0.135     -5.931      0.000      -1.062      -0.534
ar.L3         -0.1442      0.169     -0.851      0.395      -0.476       0.188
ar.L4         -0.1829      0.151     -1.209      0.227      -0.480       0.114
ar.L5         -0.1563      0.139     -1.127      0.260      -0.428       0.116
ma.L1         -1.1820      0.095    -12.441      0.000      -1.368      -0.996
ma.L2          0.8847      0.079     11.254      0.000       0.731       1.039
ar.S.L12      -0.2624      4.649     -0.056      0.955      -9.374       8.849
ar.S.L24       0.0417      0.791      0.053      0.958      -1.508       1.591
ar.S.L36      -0.1877      0.245     -0.766      0.444      -0.668       0.292
ar.S.L48      -0.2159      0.965     -0.224      0.823      -2.108       1.676
ar.S.L60       0.0121      0.997      0.012      0.990      -1.942       1.966
ma.S.L12      -0.6886      4.652     -0.148      0.882      -9.805       8.428
ma.S.L24      -0.1000      3.666     -0.027      0.978      -7.286       7.086
sigma2      1.257e+06   1.59e+05      7.915      0.000    9.46e+05    1.57e+06
==========================================================================================
==
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                         1.
32
Prob(Q):                              1.00   Prob(JB):                                 0.
52
Heteroskedasticity (H):               1.03   Skew:                                     0.
14
Prob(H) (two-sided):                  0.86   Kurtosis:                                 3.
01
==========================================================================================
==

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-st
ep).
[2] Covariance matrix is singular or near-singular, with condition number 8.79e+1
4. Standard errors may be unstable.
```

In [13]:

```
#let's forecast traffic on the website for the next 365 days:
predictions = model.predict(len(data), len(data)+365)
print(predictions)
```

```
391     9875.220680
392    10789.302685
393    10758.239498
394     9862.450733
395     8765.353123
           ...
752    11877.805795
753    11850.544081
754    11674.672866
755    11834.487803
756    11973.865502
Name: predicted_mean, Length: 366, dtype: float64
```

In [14]:

```
#plot the predictions:
data["Views"].plot(legend=True, label="Training Data",
                   figsize=(15, 10))
predictions.plot(legend=True, label="Predictions")
```

Out[14]: <Axes: >