Social Media Reach Forecasting: Instagram

In [1]:
```
#Let's start this task by importing the necessary Python libraries and the datas
import pandas as pd
import plotly.graph_objs as go
import plotly.express as px
import plotly.io as pio
pio.templates.default = "plotly_white"

data = pd.read_csv("Instagram-Reach.csv", encoding = 'latin-1')
print (data.head())
```

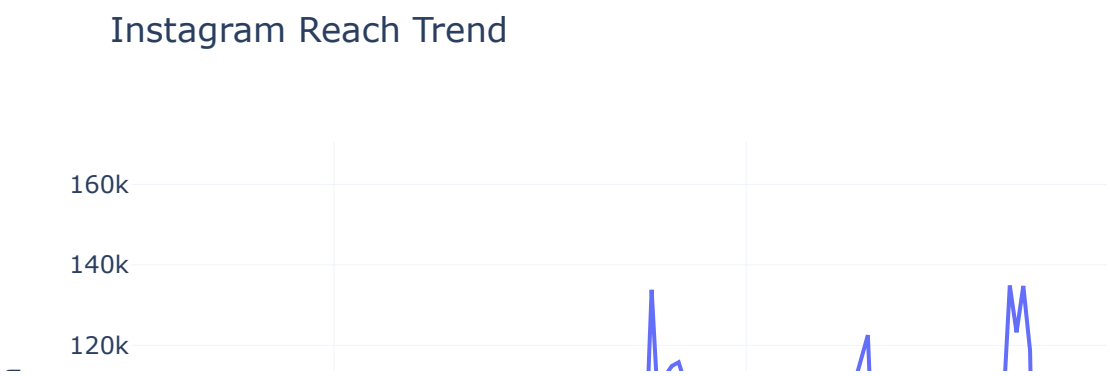
```
                  Date  Instagram reach
0  2022-04-01T00:00:00             7620
1  2022-04-02T00:00:00            12859
2  2022-04-03T00:00:00            16008
3  2022-04-04T00:00:00            24349
4  2022-04-05T00:00:00            20532
```

In [2]:
```
#The Date column into datetime datatype to move forward:
data['Date'] = pd.to_datetime(data['Date'])
print(data.head())
```

```
        Date  Instagram reach
0 2022-04-01             7620
1 2022-04-02            12859
2 2022-04-03            16008
3 2022-04-04            24349
4 2022-04-05            20532
```

Analyzing Reach

In [3]:
```
#Let's analyze the trend of Instagram reach over time using a line chart:
fig = go.Figure()
fig.add_trace(go.Scatter(x=data['Date'],
                         y=data['Instagram reach'],
                         mode='lines', name='Instagram reach'))
fig.update_layout(title='Instagram Reach Trend', xaxis_title='Date',
                                    yaxis_title='Instagram Reach')
fig.show()
```



In [4]:
```
#let's analyze Instagram reach for each day using a bar chart:
fig = go.Figure()
fig.add_trace(go.Bar(x=data['Date'],
                     y=data['Instagram reach'],
                     name='Instagram reach'))
fig.update_layout(title='Instagram Reach by Day',
                  xaxis_title='Date',
                  yaxis_title='Instagram Reach')
fig.show()
```

In [5]:
```
#let's analyze the distribution of Instagram reach using a box plot:
fig = go.Figure()
fig.add_trace(go.Box(y=data['Instagram reach'],
                     name='Instagram reach'))
fig.update_layout(title='Instagram Reach Box Plot',
                  yaxis_title='Instagram Reach')
fig.show()
```

In [6]:
```
#let's create a day column and analyze reach based on the days of the week. To c
data['Day'] = data['Date'].dt.day_name()
print(data.head())
```

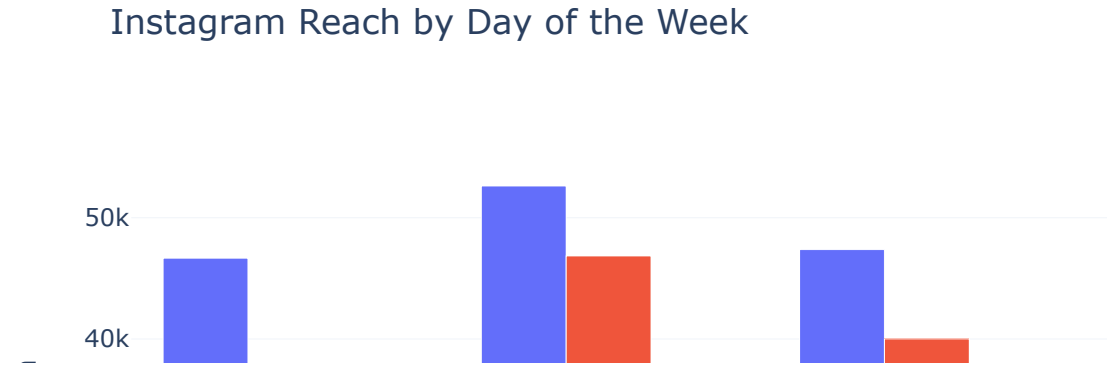
```
        Date  Instagram reach       Day
0 2022-04-01             7620    Friday
1 2022-04-02            12859  Saturday
2 2022-04-03            16008    Sunday
3 2022-04-04            24349    Monday
4 2022-04-05            20532   Tuesday
```

In [7]:
```
#let's analyze the reach based on the days of the week. For this, we can group t
import numpy as np

day_stats = data.groupby('Day')['Instagram reach'].agg(['mean', 'median', 'std']
print(day_stats)
```

```
         Day          mean   median           std
0     Friday  46666.849057  35574.0  29856.943036
1     Monday  52621.692308  46853.0  32296.071347
2   Saturday  47374.750000  40012.0  27667.043634
3     Sunday  53114.173077  47797.0  30906.162384
4   Thursday  48570.923077  39150.0  28623.220625
5    Tuesday  54030.557692  48786.0  32503.726482
6  Wednesday  51017.269231  42320.5  29047.869685
```

In [8]:
```
#let's create a bar chart to visualize the reach for each day of the week:
fig = go.Figure()
```

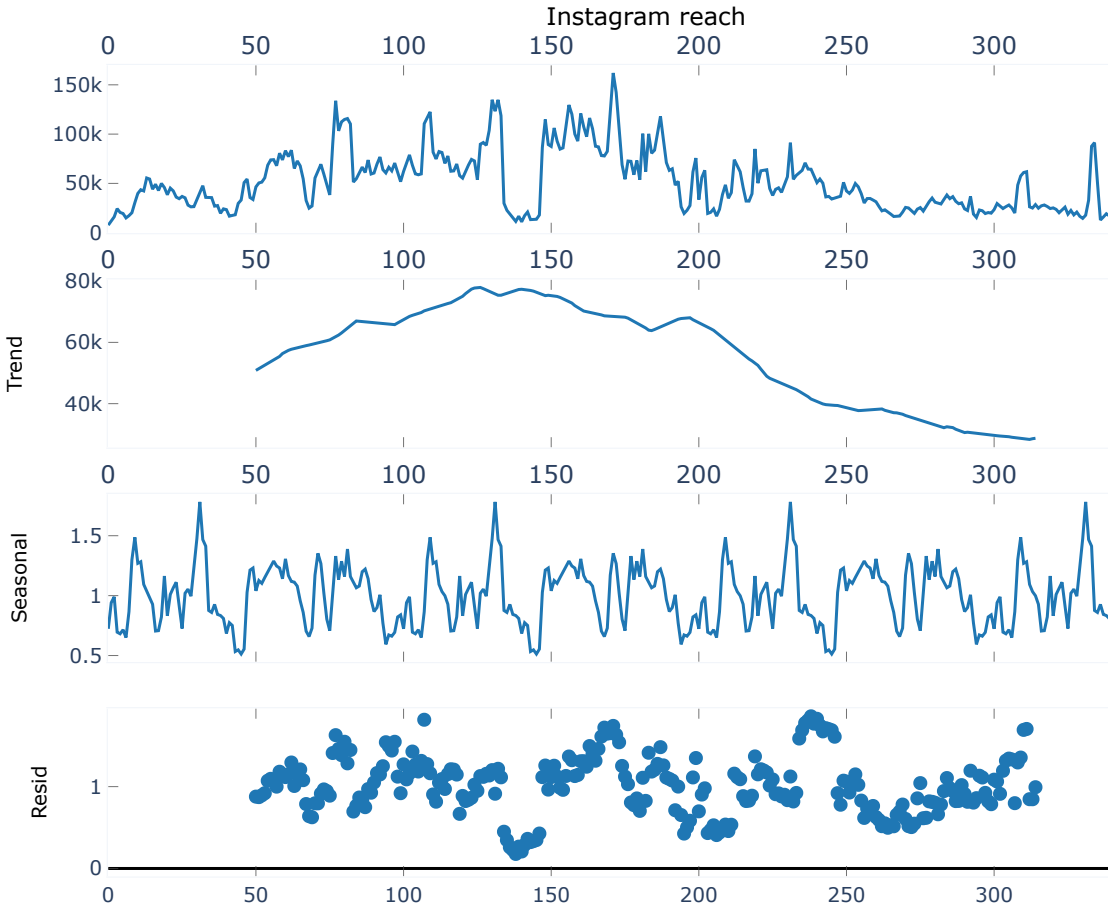
```
fig.add_trace(go.Bar(x=day_stats['Day'],
                     y=day_stats['mean'],
                     name='Mean'))
fig.add_trace(go.Bar(x=day_stats['Day'],
                     y=day_stats['median'],
                     name='Median'))
fig.add_trace(go.Bar(x=day_stats['Day'],
                     y=day_stats['std'],
                     name='Standard Deviation'))
fig.update_layout(title='Instagram Reach by Day of the Week',
                  xaxis_title='Day',
                     yaxis_title='Instagram Reach')
fig.show()
```



Instagram Reach Forecasting using Time Series Forecasting

In [9]:
```
#Let's see how to use Time Series Forecasting to forecast the reach of my Instag
#Let's look at the Trends and Seasonal patterns of Instagram reach:
from plotly.tools import mpl_to_plotly
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose

data = data[["Date", "Instagram reach"]]
```

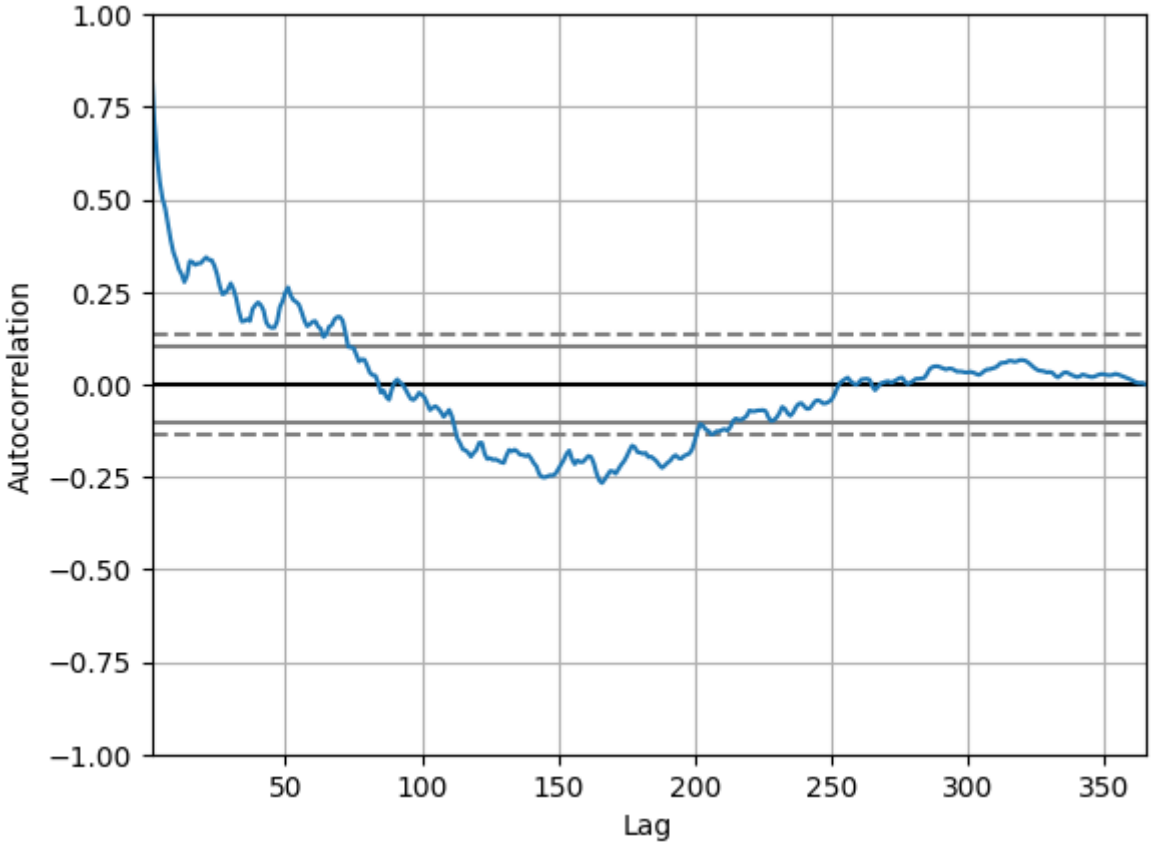
```
result = seasonal_decompose(data['Instagram reach'],
                            model='multiplicative',
                            period=100)

fig = plt.figure()
fig = result.plot()

fig = mpl_to_plotly(fig)
fig.show()
```




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

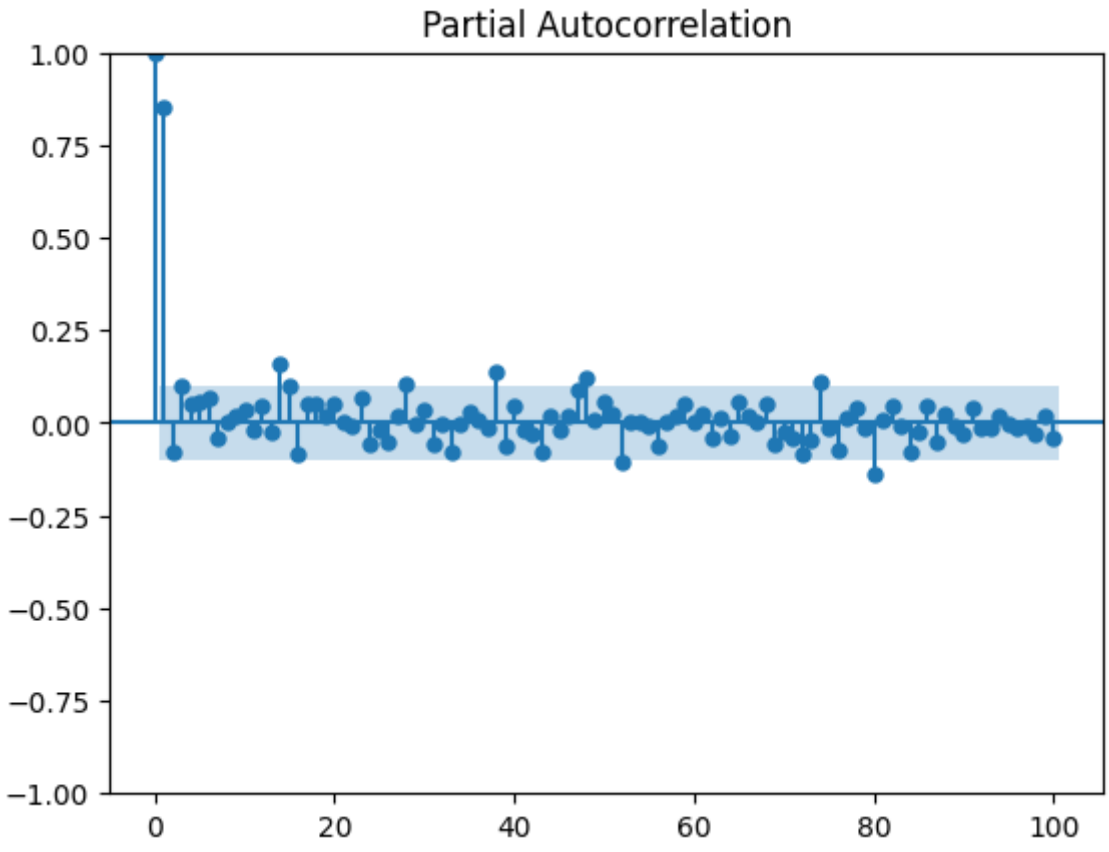
In [10]:
```
#we can use the SARIMA model to forecast the reach of the Instagram account. We
pd.plotting.autocorrelation_plot(data["Instagram reach"])
```

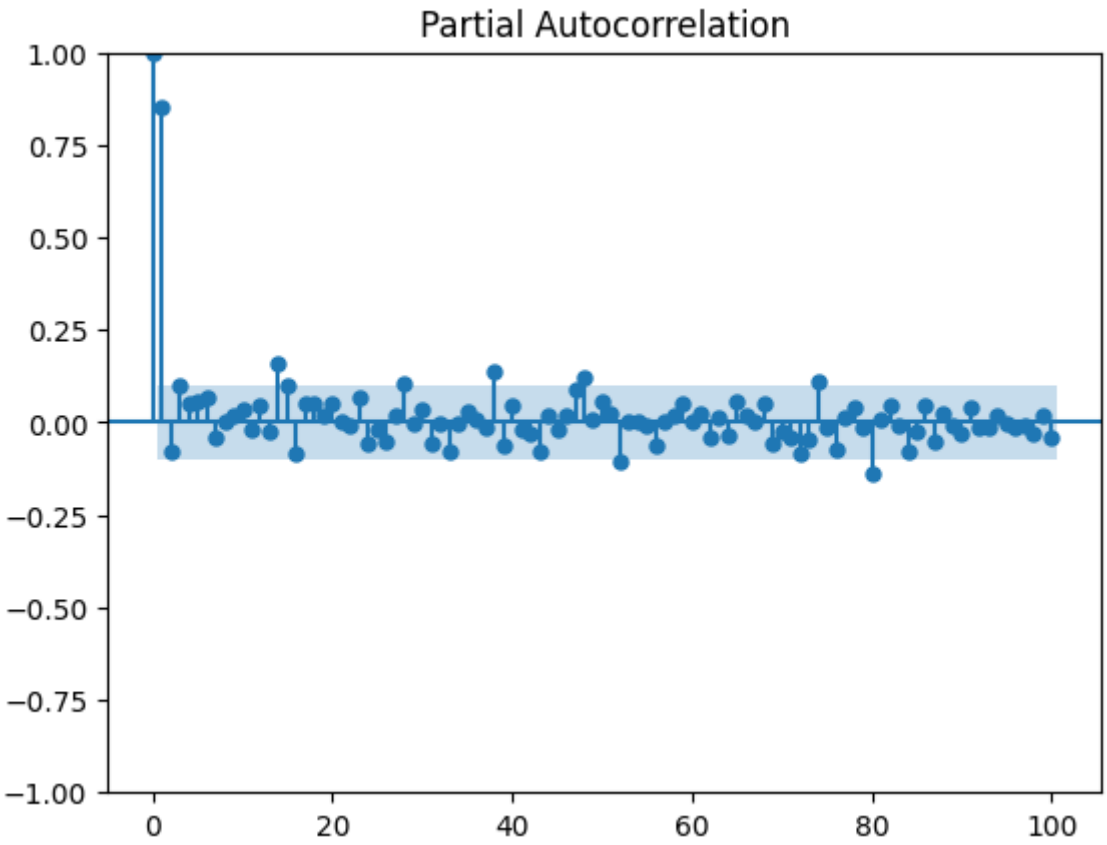
Out[10]:
```
<Axes: xlabel='Lag', ylabel='Autocorrelation'>
```



In [11]:
```
 # visualize a partial autocorrelation plot to find the value of q:
from statsmodels.graphics.tsaplots import plot_pacf
plot_pacf(data["Instagram reach"], lags = 100)
```

Out[11]:





In [12]:
```
p, d, q = 8, 1, 2

import statsmodels.api as sm
import warnings
model=sm.tsa.statespace.SARIMAX(data['Instagram reach'],
                                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\base\model.py:607: ConvergenceWarning:

Maximum Likelihood optimization failed to converge. Check mle_retvals
```

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                    Instagram reach   No. Observations:
365
Model:             SARIMAX(8, 1, 2)x(8, 1, 2, 12)   Log Likelihood
-3938.513
Date:                            Mon, 10 Jul 2023   AIC
7919.025
Time:                                    17:40:07   BIC
8000.161
Sample:                                         0   HQIC
7951.313
                                              - 365
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.1896      6.384      0.030      0.976     -12.323      12.702
ar.L2          0.4749      5.941      0.080      0.936     -11.170      12.119
ar.L3         -0.1189      1.363     -0.087      0.930      -2.790       2.553
ar.L4          0.0429      0.256      0.168      0.867      -0.458       0.544
ar.L5         -0.0202      0.188     -0.108      0.914      -0.389       0.348
ar.L6          0.0315      0.260      0.121      0.904      -0.479       0.542
ar.L7          0.0089      0.410      0.022      0.983      -0.795       0.813
ar.L8         -0.0131      0.231     -0.056      0.955      -0.467       0.441
ma.L1         -0.2235      6.380     -0.035      0.972     -12.728      12.281
ma.L2         -0.7121      6.136     -0.116      0.908     -12.738      11.314
ar.S.L12      -1.0960      1.562     -0.702      0.483      -4.157       1.965
ar.S.L24      -1.7486      2.303     -0.759      0.448      -6.263       2.766
ar.S.L36      -1.4337      1.980     -0.724      0.469      -5.314       2.446
ar.S.L48      -1.0864      1.614     -0.673      0.501      -4.250       2.077
ar.S.L60      -0.7841      1.150     -0.682      0.496      -3.039       1.471
ar.S.L72      -0.4491      0.811     -0.553      0.580      -2.039       1.141
ar.S.L84      -0.2225      0.516     -0.431      0.666      -1.234       0.789
ar.S.L96      -0.0531      0.250     -0.212      0.832      -0.543       0.437
ma.S.L12       0.2338      1.562      0.150      0.881      -2.829       3.296
ma.S.L24       0.8175      1.322      0.618      0.536      -1.774       3.409
sigma2      4.863e+08   1.41e-07   3.45e+15      0.000    4.86e+08    4.86e+08
==========================================================================================
==
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               215.
37
Prob(Q):                              0.93   Prob(JB):                         0.
00
Heteroskedasticity (H):               0.72   Skew:                             0.
29
Prob(H) (two-sided):                  0.07   Kurtosis:                         6.
79
==========================================================================================
==

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 5.43e+3
1. Standard errors may be unstable.
```

In [14]:
```
#let's make predictions using the model and have a look at the forecasted reach
predictions = model.predict(len(data), len(data)+100)
```

```
trace_train = go.Scatter(x=data.index,
                         y=data["Instagram reach"],
                         mode="lines",
                         name="Training Data")
trace_pred = go.Scatter(x=predictions.index,
                        y=predictions,
                        mode="lines",
                        name="Predictions")

layout = go.Layout(title="Instagram Reach Time Series and Predictions",
                   xaxis_title="Date",
                   yaxis_title="Instagram Reach")

fig = go.Figure(data=[trace_train, trace_pred], layout=layout)
fig.show()
```

The forecast reach of an Instagram account using Time Series Forecasting.