- 이번에는 Statsmodels의 ARIMA Function 을 이용해서 대한 민국의 최대 전력 수요 예측 실시

- 참고 자료 :

1. https://byeongkijeong.github.io/ARIMA-with-Python/
2. https://otexts.com/fppkr/non-seasonal-arima.html
3. https://www.youtube.com/watch?v=rlfMD3t5qiY&list=PL9mhQYIlKEhd60Qq4r2yC7xYKIhs97FfC&index=6&ab_channel=SKplanetTacademy
   4.https://h3imdallr.github.io/2017-08-19/arima/

- 최대 전력 소스 위치

1. http://epsis.kpx.or.kr/epsisnew/selectEkgeEpsMepGrid.do?menuId=040101

About ARIMA :

- ARIMA는 Autoregressive + Intgrate + Moving Average의 통합 모델
- Autoregressive 는 자신의 과거 데이터 기반으로 회귀하고
- Intergrate는 차분을 통해 시계열을 안정화 시킨다.
- Movig Average 를통해 자신의 트랜드를 반영 한다.

시계열 데이터는 일반적으로 불안정 (Non-stationary) 함으로 차분을 통해 안정화 시키고 AR 과 MR을 복합 적으로 적용하여 미래를 예측

- Differencing : 어렵게 생각하지 말자 개별 데이터를 일정한 간격으로 빼서 차이를 통해 Trend 와 Seasnonlity를 삭제 한다.

- 단 differecing 을 많이 하면 1st 면 첫번째 데이터 , 2st 면 두번째 데이터 등을 손실하게 됨으로 일반적으로 3회 이상 진행 하지 않음

ARIMA(p,d,q)(P,D,Q)M : 모수 찾기

- ARIMA 모델을 사용하기 위해서는 p,d,q 의 초이 값을 정해 줘야 한다.
- p : AR(p) 이고 t 시점과 이전 시점과의 회귀 모델 함수
- d : 위에서 말한 differecing 의 d
- q : MA(q) 이고 t 시점과 이전 시점의 moving average 의 잔차의 값의 예측 통상적으로 p + q < 2, p * q = 0 인 값들을 많이 사용한다.

여기서 p * q = 0 이라 하면, 두 값중 하나는 0이라는 이야기이다. ARIMA는 AR모형과 MA모형을 하나로 합쳤다면서 둘 중 하나의 모수가 0인건 또 무슨소리? 라고 할지 모르겠지만, 실제로 대부분의 시계열 자료에서는 하나의 경향만을 강하게 띄기 때문에, 이렇게 사용하는것이 더 잘 맞는다고 한다.

p,d,q 값을 찾는 법 :

- 차분을 통해 I(d)값을 정한다.
- AutoCorrelation 플롯과 Partial Autocorrelation 플롯의 절단점을 통해 p,q 값을 결정 한다.

In [ ]:
```
# 분석에 필요한 라이브러리 호출

import pandas as pd # 판다스 호출
import numpy as np  # 넘파이 호출
import statsmodels.api as sm # statsmodels 호출
import seaborn as sns # 그래프를 그리기위한 Seaborn 호출
from statsmodels.tsa.seasonal import seasonal_decompose # 데이터 필터 라이러리 호출
import matplotlib.pyplot as plt
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use t
he functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

In [ ]:
```
# 구글 드라이브 마운트
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:
```
# 데이터 호출 함수
def readData (path):
  df = pd.read_excel(path)
  return df
```

In [ ]:
```
# 분석에 사용할 시간별 대한민국 전력 수요 데이터 호출
path = './drive/MyDrive/Study/시계열 데이터/일간최대전력실적.xlsx'
need_df = readData(path)
study_df = pd.DataFrame()
study_df['MaxPower(MW)'] = need_df['최대전력(MW)']
study_df['time'] = need_df['최대전력기준일시']
study_df
```

Out[ ]:

	MaxPower(MW)	time
0	91141	2021/07/27(18:00)
1	89426	2021/07/26(18:00)
2	76521	2021/07/25(20:00)
3	78285	2021/07/24(20:00)
4	89812	2021/07/23(17:00)
...	...	...
934	69496	2019/01/05(11:00)
935	84746	2019/01/04(10:00)
936	84196	2019/01/03(10:00)
937	82437	2019/01/02(10:00)
938	65493	2019/01/01(24:00)

939 rows × 2 columns

In [ ]:
```
# 시간 데이터 타입 확인
study_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 939 entries, 0 to 938
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   MaxPower(MW)  939 non-null    int64 
 1   time          939 non-null    object
dtypes: int64(1), object(1)
memory usage: 14.8+ KB
```

In [ ]:
```
# time을 시계열 데이터로 변경
study_df['time'] = study_df['time'].str[0:10]
study_df['time'] = pd.to_datetime(study_df['time'])
study_df.set_index('time',drop=True,inplace=True)
study_df = study_df.sort_index()
study_df
```

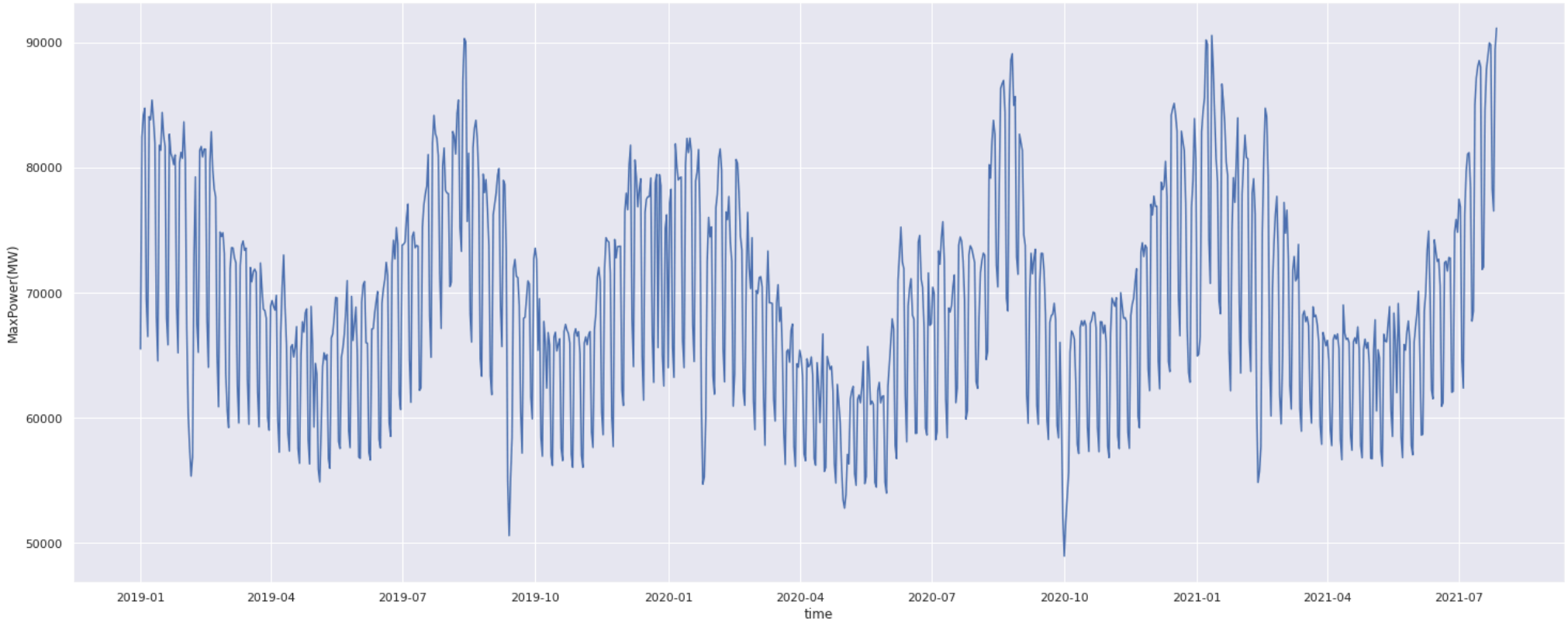
Out[ ]:

time	MaxPower(MW)
2019-01-01	65493
2019-01-02	82437
2019-01-03	84196
2019-01-04	84746
2019-01-05	69496
...	...
2021-07-23	89812
2021-07-24	78285
2021-07-25	76521
2021-07-26	89426
2021-07-27	91141

939 rows × 1 columns

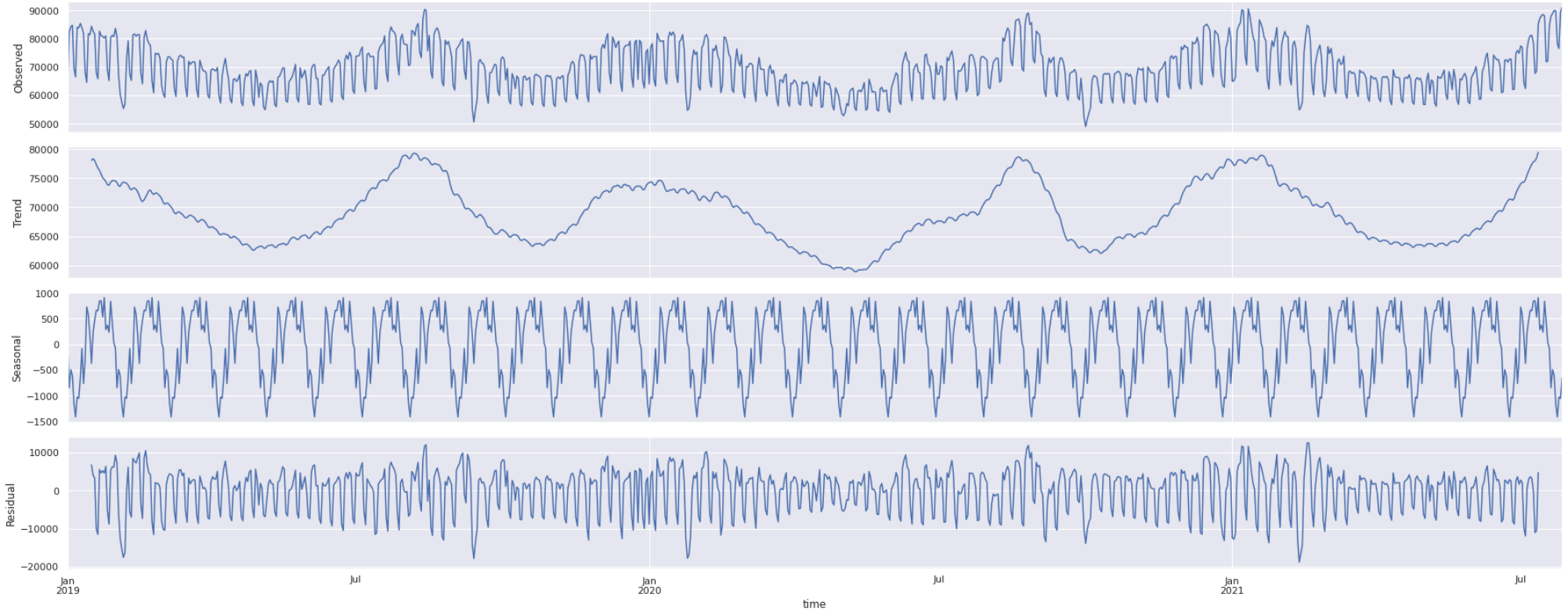
In [ ]:
```
# 최대 전력 수요 가시화
sns.set(rc={'figure.figsize':(25,10)})
sns.lineplot(x=study_df.index , y=study_df['MaxPower(MW)'])
```

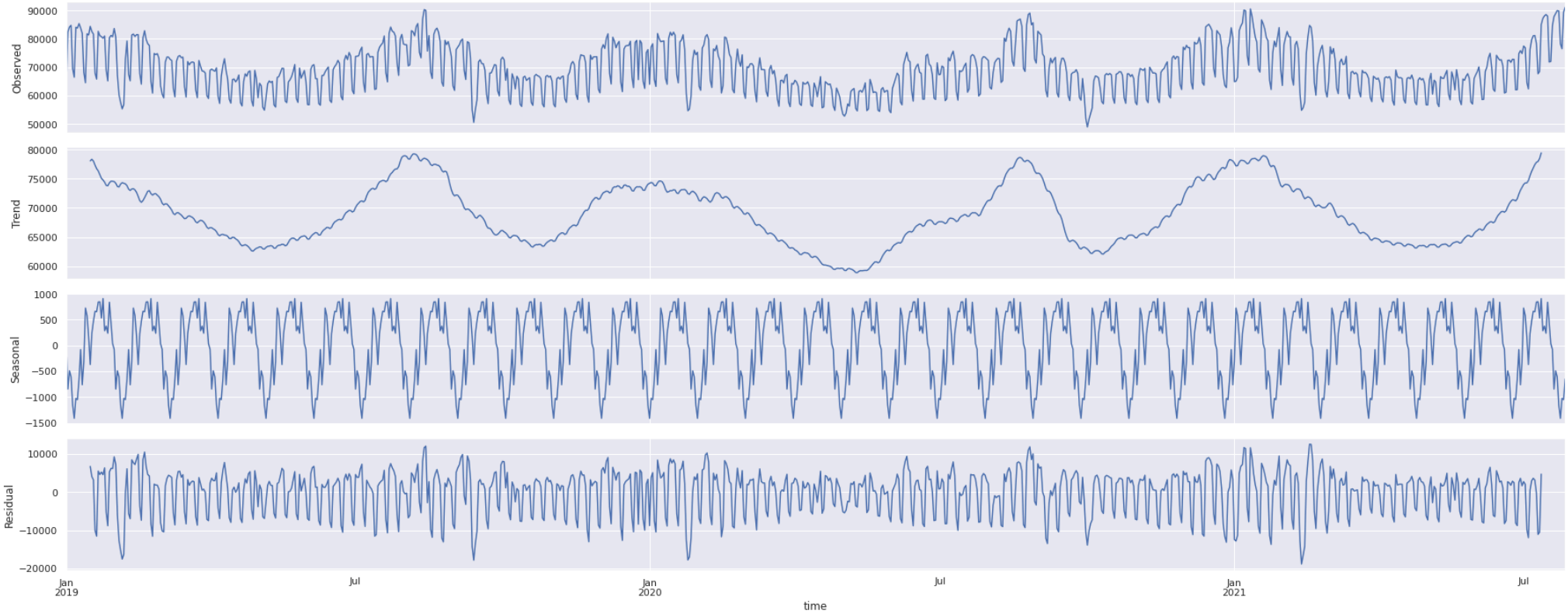
Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff3060b0110>
```



In [ ]:
```
# Seasnonl Decompose 모델로 시계열 육안 확인
result = seasonal_decompose(study_df, model='Mulitiolicative', freq= 30) # 대략 1달치 기준으로 데이터 분해 실시
result.plot()

# 월간 시계열성이 강하다.
```

Out[ ]:


Observed
90000
80000
70000
60000
50000
Trend
80000
75000
70000
65000
60000
Seasonal
1000
500
0
−500
−1000
−1500
Residual
10000
0
−10000
−20000
Jan
2019
Jul
Jan
2020
Jul
Jan
2021
Jul
time



- ARIMA 모델을 사용하기 위해서 시계열 데이터가 정상상태 인지 확인하기 위해 Augmented Dickey Fuller (ADF) Test를 실시 한다.
- Statsmodel 의 adfuler 모듈을 활용하여 쉽게 값을 구한다.

In [ ]:
```
# 데이터 정상 상태 수치 확인
from statsmodels.tsa.stattools import adfuller #ADF Test를 위한 함수 호출

st_result = adfuller(study_df['MaxPower(MW)'])
print(st_result)

study_df['1st diff'] = study_df['MaxPower(MW)']- study_df['MaxPower(MW)'].shift(1)

# Test 결과 이미 P-Velue 가 0.05 이하 임으로 시계열 데이터가 정상 상태로 판다.
# 만약 0.05 이상 이라면 차분을 통해 시계열 안정화 실시
```

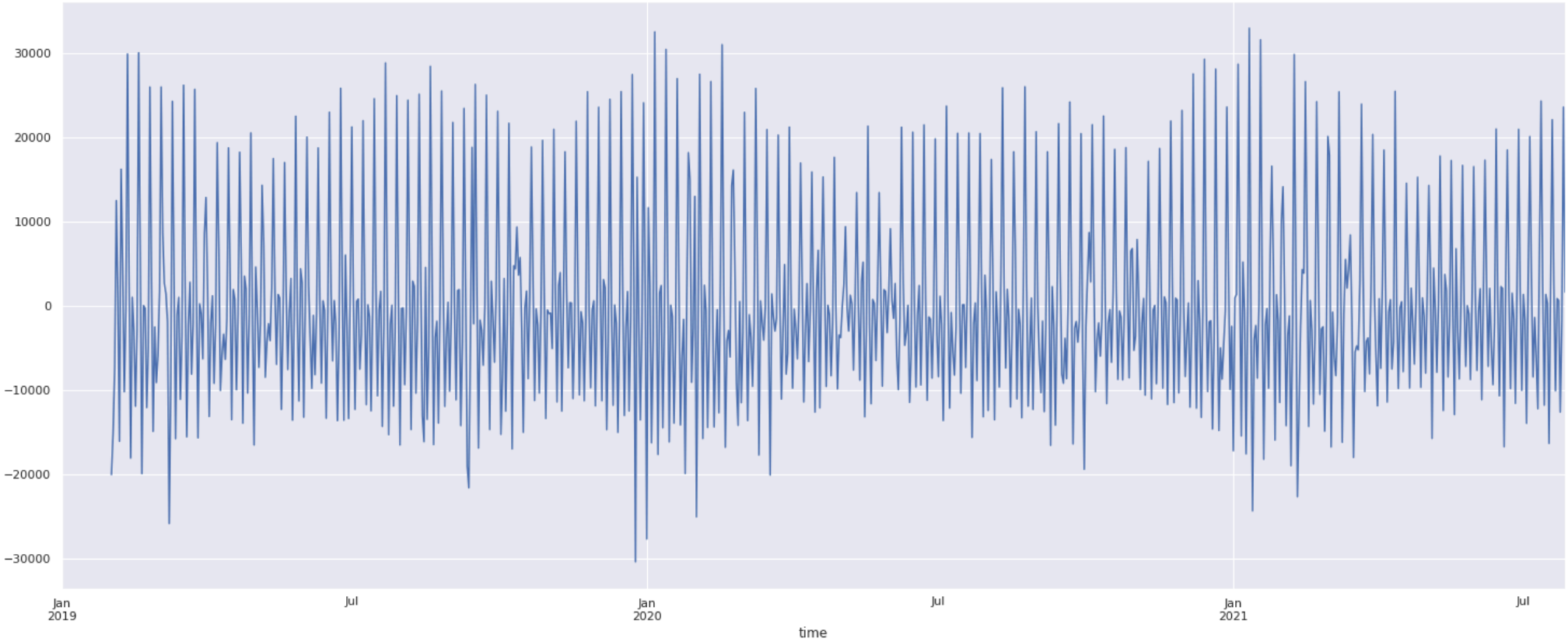
```
(-2.8648766784005697, 0.04958746836469044, 21, 917, {'1%': -3.437501253878553, '5%': -2.864696995470416, '10%': -2.568450985005155}, 17312.592032968387)
```

In [ ]:
```
# diff 값으로 계절성 값 계산
```

```
study_df['seasonal diff'] = study_df['1st diff'] - study_df['1st diff'].shift(30)
st_result = adfuller(study_df['seasonal diff'].dropna()) # 차분에 의해 발생한 Na 값을 제거 하고 진행
print(st_result[1])
study_df['seasonal diff'].plot()

# 결론적으로 ARIMA의 차분 값은
#d = 0, D = 1 사용
```

```
4.549289812479293e-12
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff2fad068d0>



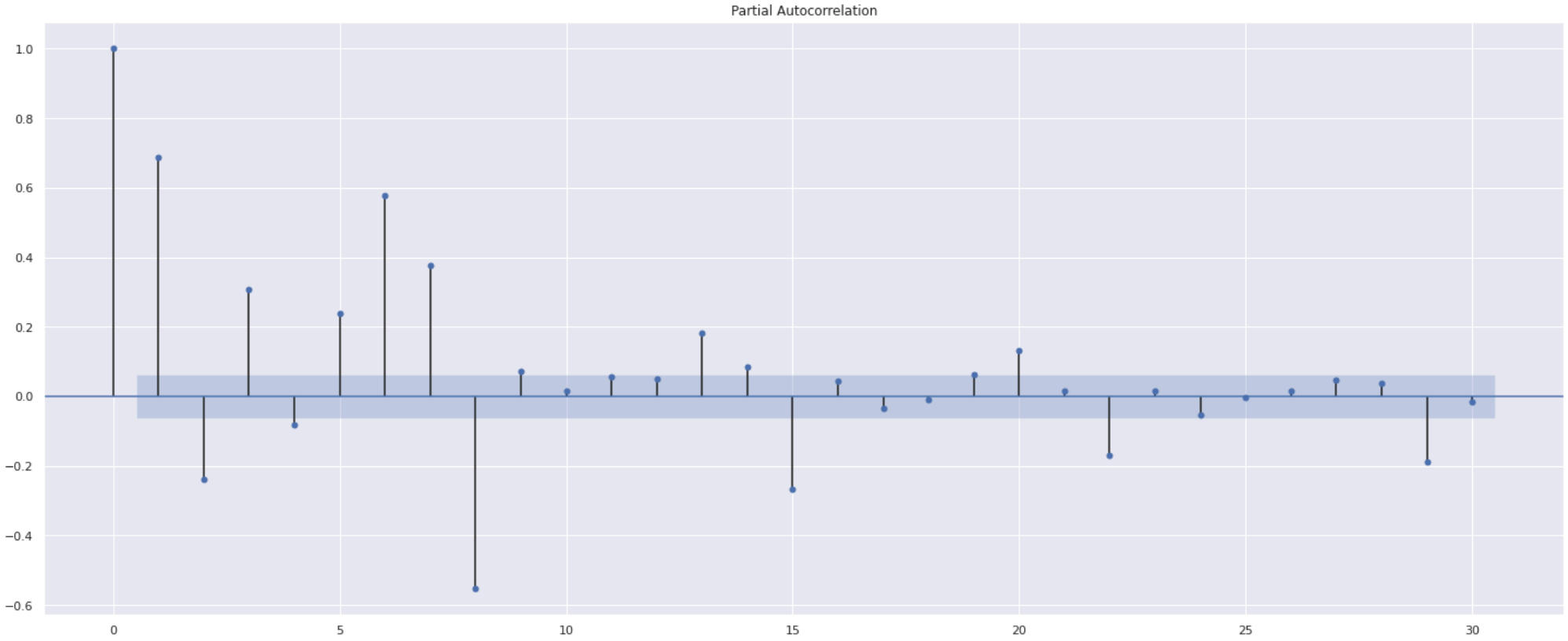
Auto Correlation Plot : 시계열 데이터에서 특정 시간 만큼 지연되 데이터와의 상관 성을 그래프트 보여줌

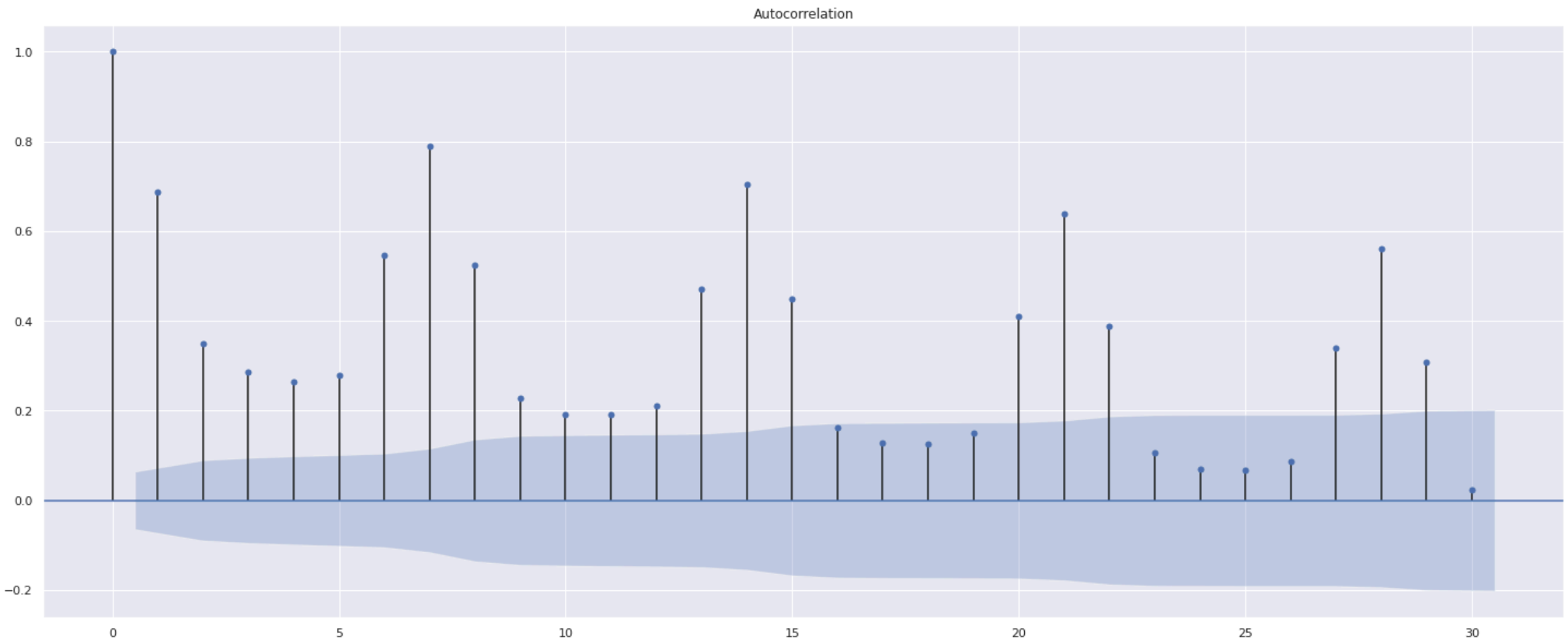
ACF 와 PACF 그래프를 통해 적절한 ARIMA의 모수를 결정해 보자

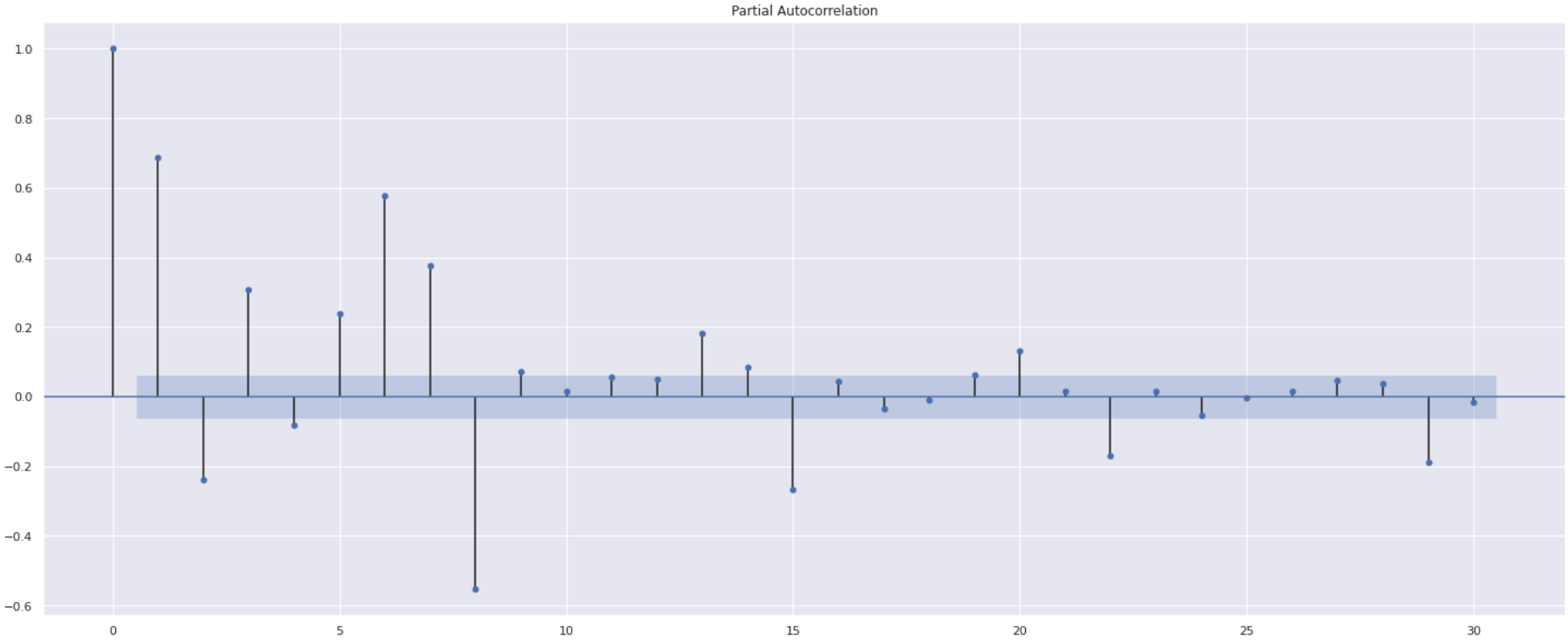
Model	ACF	PACF
AR(p)	점차 감소하여 0에 접근	시차 p 이후에 0
MA(q)	시차 q 이후에 0	점차 감소하여 0에 접근
ARMA(p,q)	점차 감소하여 0에 접근 (시차 q 이후에 0)	점차 감소하여 0에 접근 (시차 p 이후에 0)

In [ ]:
```
# ACF 그려 보기 / PACF
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf # acf 와 pacf 시각화를 위한 라이브러리 호출

plot_acf(study_df['MaxPower(MW)'])
plot_pacf(study_df['MaxPower(MW)'])
```

Out[ ]:

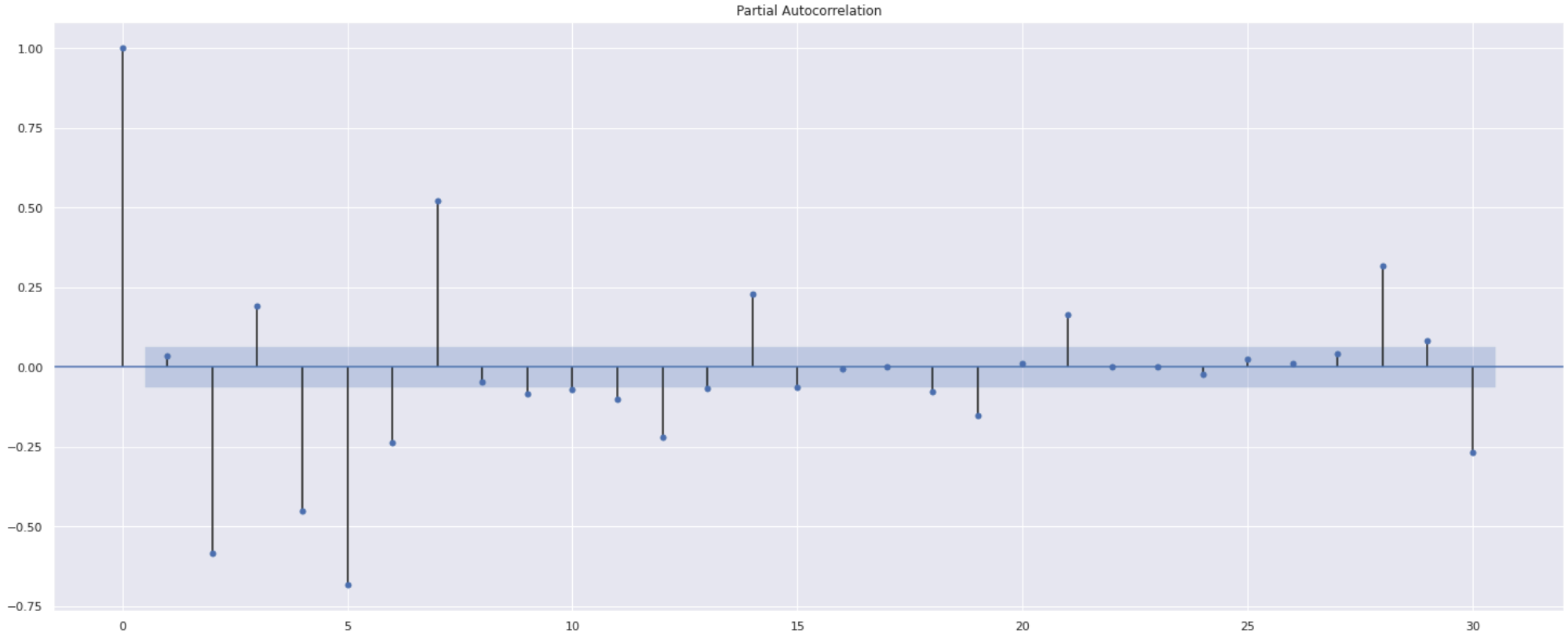

Partial Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
5
10
15
20
25
30

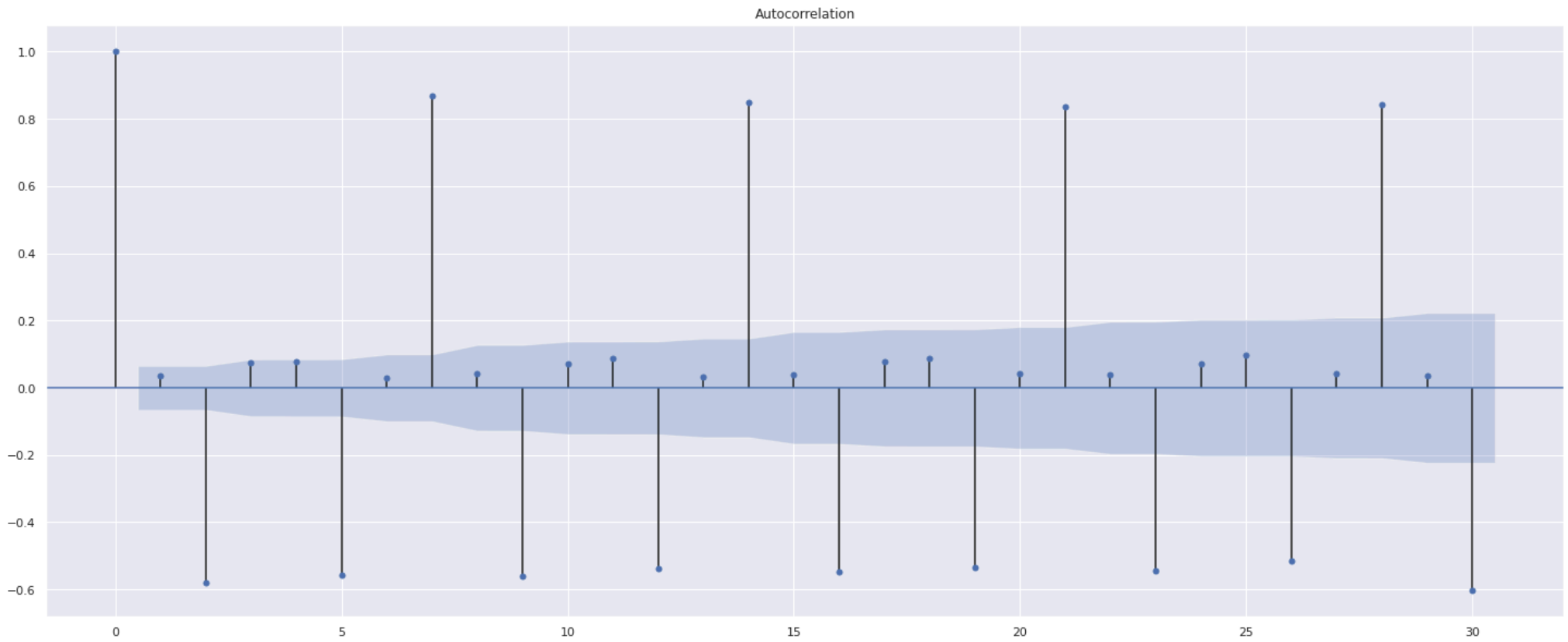

Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
5
10
15
20
25
30

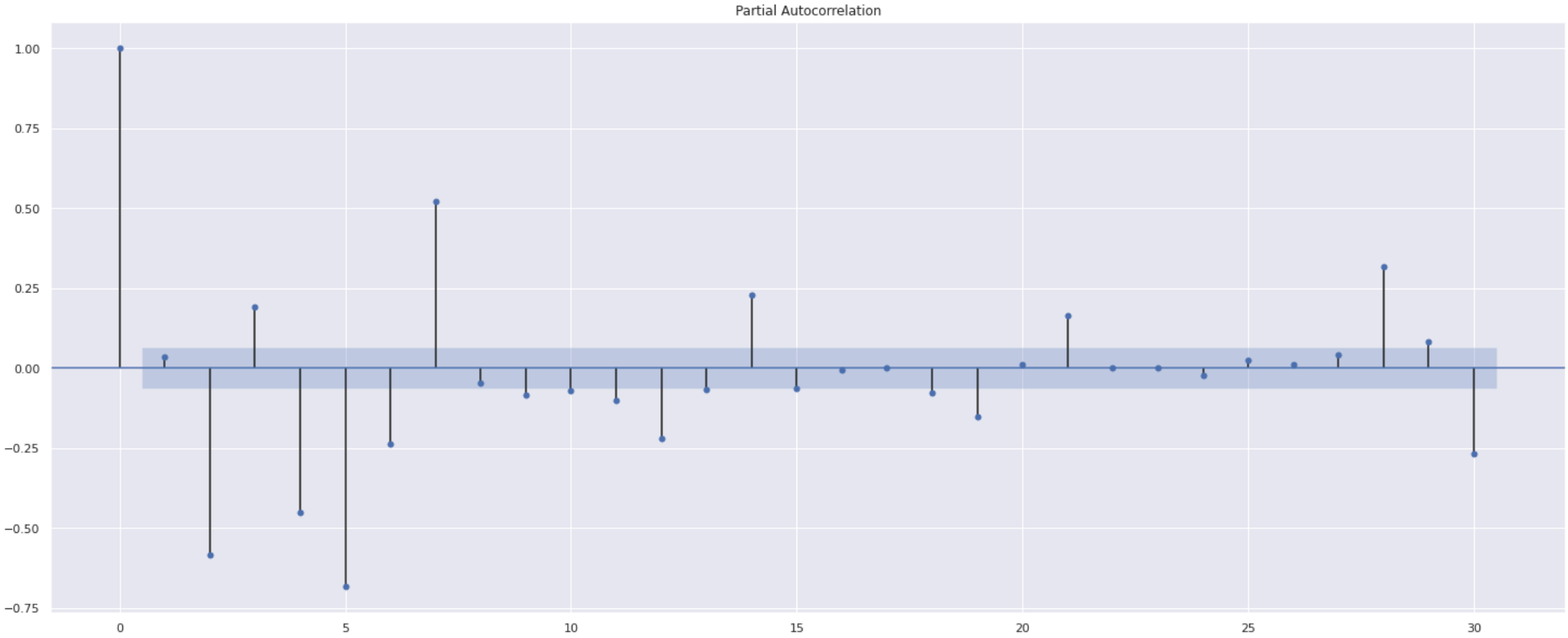


In [ ]:
```
# ACF 그려 보기 / PACF (계절성)

plot_acf(study_df['seasonal diff'].dropna())
plot_pacf(study_df['seasonal diff'].dropna())
```

Out[ ]:


Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0
5
10
15
20
25
30


Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
5
10
15
20
25
30



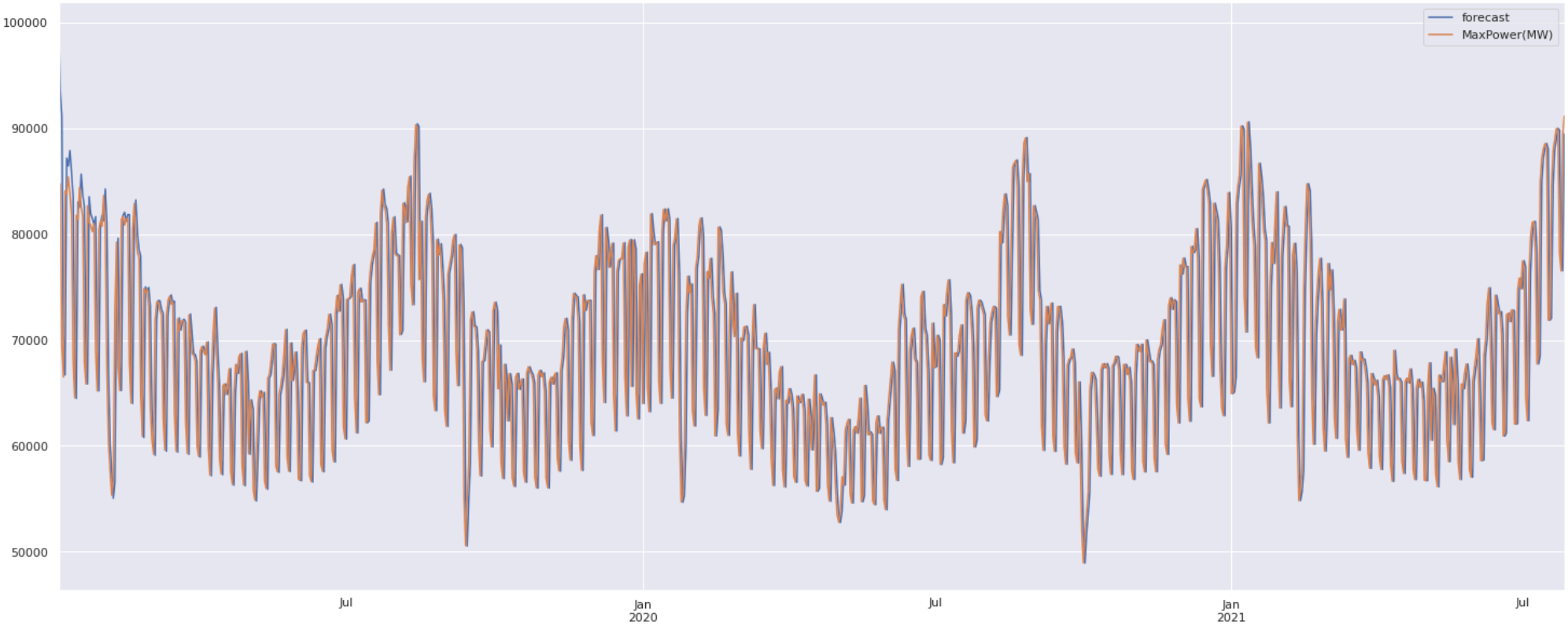
In [ ]:
```
# Arima 예측
from statsmodels.tsa.arima_model import ARIMA

model = ARIMA(study_df['MaxPower(MW)'], order=(0,2,1))
model_fit = model.fit(trend='nc',full_output=True, disp=1)
print(model_fit.summary())
```

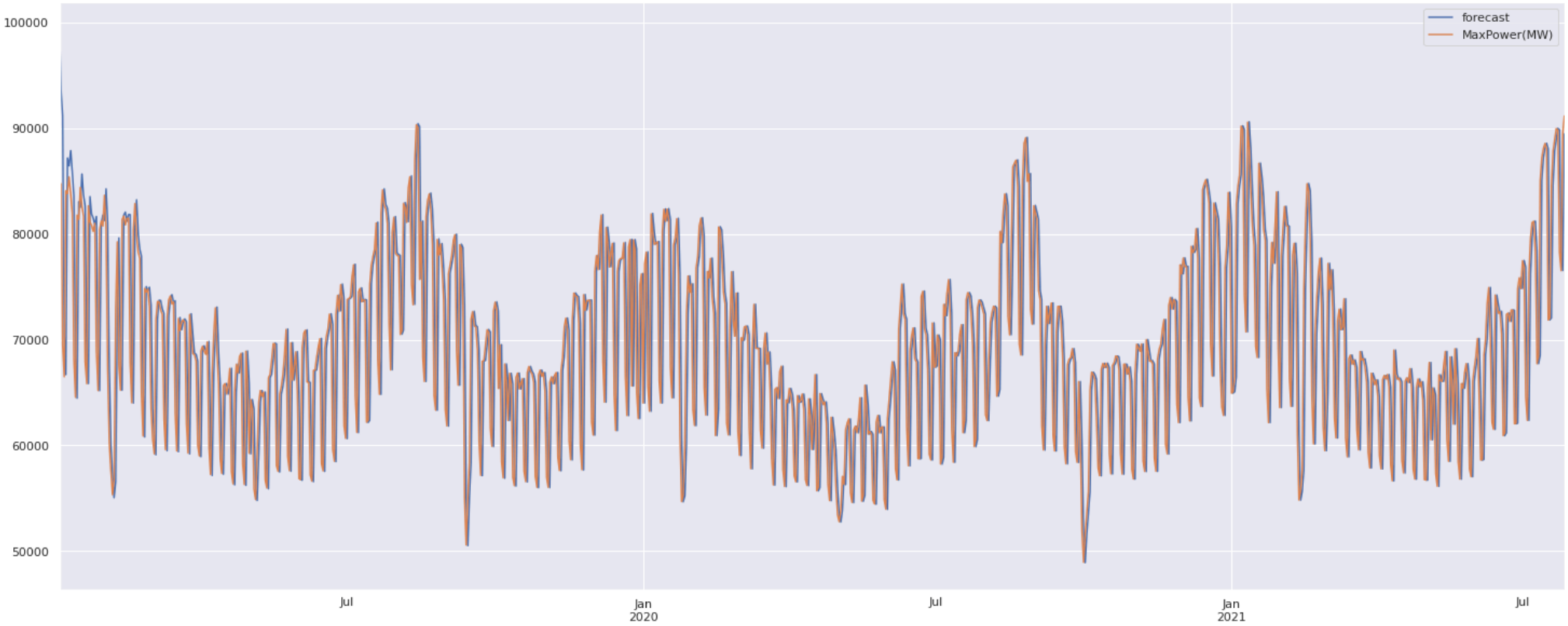
```
                             ARIMA Model Results
==============================================================================
Dep. Variable:        D2.MaxPower(MW)   No. Observations:                  937
Model:                 ARIMA(0, 2, 1)   Log Likelihood               -9578.656
Method:                       css-mle   S.D. of innovations           6635.044
Date:                Mon, 02 Aug 2021   AIC                          19161.312
Time:                        14:17:21   BIC                          19170.997
Sample:                  01-03-2019   HQIC                         19165.005
                       - 07-27-2021
================================================================================================
                             coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
ma.L1.D2.MaxPower(MW)    -0.9999      0.003   -377.007      0.000      -1.005      -0.995
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
MA.1            1.0001           +0.0000j            1.0001            0.0000
-----------------------------------------------------------------------------
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/base/tsa_model.py:165: ValueWarning: No frequency information was provide
d, so inferred frequency D will be used.
  % freq, ValueWarning)
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/base/tsa_model.py:165: ValueWarning: No frequency information was provide
d, so inferred frequency D will be used.
  % freq, ValueWarning)
```

In [ ]:
```
# 시각화 하기
model_fit.plot_predict()
```

Out[ ]:


forecast
MaxPower(MW)
100000
90000
80000
70000
60000
50000
Jul
Jan
2020
Jul
Jan
2021
Jul



In [ ]:

```
# 미래 예측 및 종료
# 19~20년 까지 데이터를 활용하여 ARIMA로 예측하고 21년도 데이터와 오차 계산해 보자

df_1920 = study_df['2019-01':'2020-12']

model = ARIMA(df_1920['MaxPower(MW)'], order=(0,1,1))
model_fit = model.fit(trend='c',full_output=True, disp=1)
fore = model_fit.forecast(steps=27,alpha=0.05)
fore_df = pd.DataFrame()
result_df = study_df['2021-07':'2021-08']
fore_df['Real'] = result_df['MaxPower(MW)']
fore_df.reset_index(drop=False)
fore_df['ARIMA'] = fore[0]
temp_df = study_df['2020-07-01':'2020-07-27'].reset_index()
fore_df.reset_index(inplace=True)
fore_df['20year'] = temp_df['MaxPower(MW)']*1.05
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/base/tsa_model.py:165: ValueWarning: No frequency information was provide
d, so inferred frequency D will be used.
  % freq, ValueWarning)
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/base/tsa_model.py:165: ValueWarning: No frequency information was provide
d, so inferred frequency D will be used.
  % freq, ValueWarning)
```

Partial AutoCorrelation Plot

t 시점과 특정 시간만큼 지연된 (lag) 시점 t-p의 연관성을 그사이의 영향성을 배제 하고 보여준다. (Auto와 다르게 중간데 영향을 주는 연관 시점을 모두 삭제)

In [ ]:
```
fore_df
```

Out[ ]:

	time	Real	ARIMA	20year
0	2021-07-01	77476	79626.348223	70885.50
1	2021-07-02	76865	79647.086603	73956.75
2	2021-07-03	64612	79667.824983	73453.80
3	2021-07-04	62368	79688.563363	61153.05
4	2021-07-05	76256	79709.301743	61822.95
5	2021-07-06	79663	79730.040122	76997.55
6	2021-07-07	81086	79750.778502	75878.25
7	2021-07-08	81187	79771.516882	78089.55
8	2021-07-09	78397	79792.255262	79458.75
9	2021-07-10	67726	79812.993642	76128.15
10	2021-07-11	68518	79833.732022	64576.05
11	2021-07-12	85011	79854.470402	61331.55
12	2021-07-13	87172	79875.208782	72209.55
13	2021-07-14	88087	79895.947162	71879.85
14	2021-07-15	88551	79916.685542	72399.60
15	2021-07-16	88015	79937.423922	74034.45
16	2021-07-17	71849	79958.162302	74988.90
17	2021-07-18	72054	79978.900682	64264.20
18	2021-07-19	84586	79999.639062	65495.85
19	2021-07-20	87841	80020.377442	77490.00
20	2021-07-21	88937	80041.115822	78172.50
21	2021-07-22	89958	80061.854202	77816.55
22	2021-07-23	89812	80082.592581	76170.15
23	2021-07-24	78285	80103.330961	72871.05
24	2021-07-25	76521	80124.069341	62884.50
25	2021-07-26	89426	80144.807721	63606.90

	time	Real	ARIMA	20year
26	2021-07-27	91141	80165.546101	76656.30

ACF와 PACF를 활용하여 P,Q 값 일반 적인 결정 방법

- 하지만 아래의 일반적인 상황과 달리 이상한 경우가 많기 때문에 Try Error를 통해 최적의 값을 찾아야 한다.

In [ ]:
```
mod = sm.tsa.statespace.SARIMAX(df_1920['MaxPower(MW)'].shift(1), order=(2,1,2),seasonal_order=(0,1,0,30)).fit()
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/base/tsa_model.py:165: ValueWarning: No frequency information was provide
d, so inferred frequency D will be used.
  % freq, ValueWarning)
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/statespace/sarimax.py:961: UserWarning: Non-invertible starting MA parame
ters found. Using zeros as starting parameters.
  warn('Non-invertible starting MA parameters found.'
```

In [ ]:
```
print(mod.summary())
```

```
                                Statespace Model Results
==========================================================================================
Dep. Variable:                        MaxPower(MW)   No. Observations:                  731
Model:             SARIMAX(2, 1, 2)x(0, 1, 0, 30)   Log Likelihood               -7326.933
Date:                            Mon, 02 Aug 2021   AIC                          14663.866
Time:                                    14:17:28   BIC                          14686.621
Sample:                                01-01-2019   HQIC                         14672.662
                                     - 12-31-2020
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.1191      0.055      2.146      0.032       0.010       0.228
ar.L2         -0.2523      0.050     -5.017      0.000      -0.351      -0.154
ma.L1         -0.0200      0.038     -0.533      0.594      -0.094       0.054
ma.L2         -0.5687      0.042    -13.612      0.000      -0.651      -0.487
sigma2       7.12e+07   3.12e-10   2.28e+17      0.000    7.12e+07    7.12e+07
==========================================================================================
Ljung-Box (Q):                     1727.97   Jarque-Bera (JB):               124.05
Prob(Q):                              0.00   Prob(JB):                         0.00
Heteroskedasticity (H):               0.58   Skew:                             0.00
Prob(H) (two-sided):                  0.00   Kurtosis:                         5.06
==========================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number  5e+32. Standard errors may be unstable.
```

In [ ]:
```
df = pd.DataFrame()
df['ARIMA'] = mod.predict(start = 365*2+1, end= 365*2+30, dynamic= True)
```

In [ ]:
```
real = study_df['2021-01':'2021-01']
df['Real'] = real['MaxPower(MW)']
```

In [ ]:
```
df['erro'] = (df['ARIMA'] - df['Real']) / df['ARIMA'] * 100
```

In [ ]:
```
erro = abs(df['erro']).mean()
```

In [ ]:
```
erro
```

Out[ ]:
```
14.422144862924338
```

In [ ]: