시계열 분석

1. 데이터 불러오기

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt

mpl.rc('font',family='Malgun Gothic')
```

In [2]:

```
df1 = pd.read_csv('Medical_Image_Data_01.csv', encoding='cp949')
df2 = pd.read_csv('Patient_Diagnosis_Data.csv')
df3 = pd.read_csv('Patient_Surgery_Data.csv')
df1.isnull().sum()
```

Out[2]:

```
환자ID                  0
전방디스크높이(mm)         0
후방디스크높이(mm)         0
지방축적도                 3
Instability         0
MF + ES             0
Modic change        0
PI                  4
PT                  4
Seg Angle(raw)      1
Vaccum disc         0
골밀도                896
디스크단면적               1
디스크위치                0
척추이동척도               0
척추전방위증               0
dtype: int64
```

In [3]:

```
import matplotlib.pyplot as plt
import matplotlib

plt.rc('font', family='NanumBarunGothic')
matplotlib.rc('axes',unicode_minus=False)
```

In [4]:

```
merge1 = pd.merge(df1, df2, on='환자ID', how='inner')
final = pd.merge(merge1, df3, on=['환자ID','연령', '입원일자', '신장', '체중', '퇴원일자', '헤드
final.columns
```

Out[4]:

```
Index(['환자ID', '전방디스크높이(mm)', '후방디스크높이(mm)', '지방축적도', 'Instability', 'MF +
ES',
       'Modic change', 'PI', 'PT', 'Seg Angle(raw)', 'Vaccum disc', '골밀도',
       '디스크단면적', '디스크위치', '척추이동척도', '척추전방위증', 'Large Lymphocyte',
       'Location of herniation', 'ODI', '가족력', '간질성폐질환', '고혈압여부', '과거수술횟수',
       '당뇨여부', '말초동맥질환여부', '빈혈여부', '성별', '스테로이드치료', '신부전여부', '신장',
'심혈관질환',
       '암발병여부', '연령', '우울증여부', '입원기간', '입원일자', '종양진행여부', '직업', '체중',
'퇴원일자',
       '헤모글로빈수치', '혈전합병증여부', '환자통증정도', '흡연여부', '통증기간(월)', '수술기법',
'수술시간',
       '수술실패여부', '수술일자', '재발여부', '혈액형'],
      dtype='object')
```

수술 수 관련하여 분석하기

In [5]:

```
import statsmodels.tsa.api as tsa
```

In [6]:

```
from pylab import rcParams
```

In [7]:

```
final.head()
```

Out[7]:

	환자ID	전방디스크높이(mm)	후방디스크높이(mm)	지방축적도	Instability	MF + ES	Modic change	PI	PT	Seg Angle(raw)	...	혈전합병증여부	환자통증정도	흡연여부	(
0	1PT	16.1	12.3	282.3	0	1824.6	3	51.6	36.6	14.4	...	0	10	0	1
1	2PT	13.7	6.4	177.3	0	1737.5	0	40.8	7.2	17.8	...	0	10	0	1
2	3PT	13.6	7.4	256.8	0	1188.5	0	67.5	27.3	10.2	...	0	7	0	1
3	4PT	10.6	7.3	250.1	0	2534.5	0	49.2	18.7	19.9	...	0	7	0	2
4	5PT	17.1	8.1	232.2	0	1840.6	0	58.8	14.7	5.2	...	0	7	0	1

5 rows × 51 columns

In [8]:

```
final.columns
```

Out[8]:

```
Index(['환자ID', '전방디스크높이(mm)', '후방디스크높이(mm)', '지방축적도', 'Instability', 'MF +
ES',
       'Modic change', 'PI', 'PT', 'Seg Angle(raw)', 'Vaccum disc', '골밀도',
       '디스크단면적', '디스크위치', '척추이동척도', '척추전방위증', 'Large Lymphocyte',
       'Location of herniation', 'ODI', '가족력', '간질성폐질환', '고혈압여부', '과거수술횟수',
       '당뇨여부', '말초동맥질환여부', '빈혈여부', '성별', '스테로이드치료', '신부전여부', '신장',
'심혈관질환',
       '암발병여부', '연령', '우울증여부', '입원기간', '입원일자', '종양진행여부', '직업', '체중',
'퇴원일자',
       '헤모글로빈수치', '혈전합병증여부', '환자통증정도', '흡연여부', '통증기간(월)', '수술기법',
'수술시간',
       '수술실패여부', '수술일자', '재발여부', '혈액형'],
      dtype='object')
```

In [9]:

```
#수술 일자 관련
final['수술일자(date)'] = pd.to_datetime(final['수술일자'], format='%Y%m%d')
```

In [10]:

```
final['수술일자(date)']
```

Out[10]:

```
0       2019-07-15
1       2019-07-16
2       2019-07-31
3       2019-08-02
4       2019-09-06
           ...    
1889    2017-04-07
1890    2017-04-27
1891    2017-04-11
1892    2017-04-10
1893    2017-04-12
Name: 수술일자(date), Length: 1894, dtype: datetime64[ns]
```

In [11]:

```
final['수술일자(date)'].unique
```

Out[11]:

```
<bound method Series.unique of 0       2019-07-15
1       2019-07-16
2       2019-07-31
3       2019-08-02
4       2019-09-06
           ...    
1889    2017-04-07
1890    2017-04-27
1891    2017-04-11
1892    2017-04-10
1893    2017-04-12
Name: 수술일자(date), Length: 1894, dtype: datetime64[ns]>
```

In [12]:

```
final['수술일자(count)'] = 1
final.head()
```

Out[12]:

	환자ID	전방디스크높이(mm)	후방디스크높이(mm)	지방축적도	Instability	MF + ES	Modic change	PI	PT	Seg Angle(raw)	...	흡연여부	통증기간(월)	수술기법
0	1PT	16.1	12.3	282.3	0	1824.6	3	51.6	36.6	14.4	...	0	1.0	TELI
1	2PT	13.7	6.4	177.3	0	1737.5	0	40.8	7.2	17.8	...	0	1.0	TELI
2	3PT	13.6	7.4	256.8	0	1188.5	0	67.5	27.3	10.2	...	0	1.0	TELI
3	4PT	10.6	7.3	250.1	0	2534.5	0	49.2	18.7	19.9	...	0	2.0	TELI
4	5PT	17.1	8.1	232.2	0	1840.6	0	58.8	14.7	5.2	...	0	1.0	TELI

5 rows × 53 columns

In [13]:

```
# 수술 일자 count
final_series = pd.pivot_table(data=final, index='수술일자(date)', values= '수술일자(count)', a
final_series
```

Out[13]:

	수술일자(count)
수술일자(date)	
2009-01-20	1
2009-01-30	1
2009-03-11	1
2009-03-28	1
2009-04-01	2
...	...
2020-07-29	1
2020-07-30	3
2020-07-31	5
2020-08-04	1
2020-08-06	1

976 rows × 1 columns

1W : 1주 단위 구간

- 특정 주기 단위로 분할 : 1 주일 단위구간

1W : 1 주일 단위 구간

In [14]:

```
w = final_series['수술일자(count)'].resample('1W').sum()
```

In [15]:

```
# w1 = w.fillna(w.mean())
```

In [16]:

```
# w1
```

In [17]:

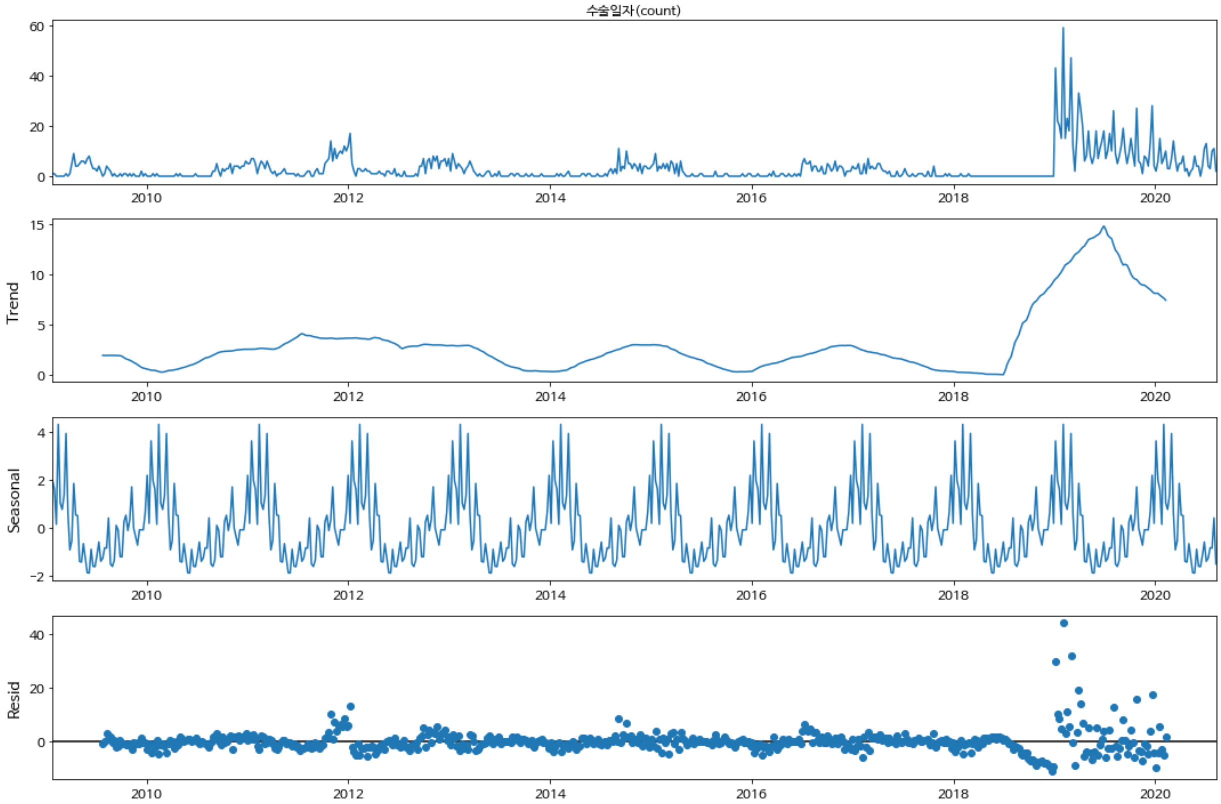
```
rcParams['figure.figsize'] = 15,10

#차트 기본 크기 설정
mpl.rcParams['axes.labelsize'] = 14
mpl.rcParams['xtick.labelsize'] = 12
mpl.rcParams['ytick.labelsize'] = 12
mpl.rcParams['text.color'] = 'k'
```

In [18]:

```
w1 = w.fillna(0)
```

In [19]:

```
model_series = tsa.seasonal_decompose(w1, model='additive')
fig = model_series.plot()
plt.show()
```



resid : 잔차

2. ARIMA

In [31]:

```
import itertools # 반복수를 만드는 라이브러리
```

In [32]:

```
p = d = q = range(0, 2)
pdq = list(itertools.product(p, d, q))
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]
```

In [33]:

```
seasonal_pdq
```

Out[33]:

```
[(0, 0, 0, 12),
 (0, 0, 1, 12),
 (0, 1, 0, 12),
 (0, 1, 1, 12),
 (1, 0, 0, 12),
 (1, 0, 1, 12),
 (1, 1, 0, 12),
 (1, 1, 1, 12)]
```

In [34]:

```
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

In [35]:
```
param_list = []
param_seasonal_list = []
results_AIC_list = []
```

1주일

In [36]:
```
for param in pdq:
    for param_seasonal in seasonal_pdq:
       try:
           mod = tsa.statespace.SARIMAX(w1,order=param,
                                        seasonal_order=param_seasonal,
                                        enforce_stationarity=False,
                                        enforce_invertibility=False)
           results = mod.fit()
           param_list.append(param)
           param_seasonal_list.append(param_seasonal)
           results_AIC_list.append(results.aic)
       except:
            continue
```

In [37]:
```
ARIMA_list = pd.DataFrame({'Parameter':param_list,'Seasonal':param_seasonal_list,'AIC':resul
ARIMA_list.to_excel('arima_model_list.xlsx')
```

In [38]:
```
ARIMA_list.sort_values(by='AIC')
```

Out[38]:

	Parameter	Seasonal	AIC
43	(1, 0, 1)	(0, 1, 1, 12)	3307.531619
47	(1, 0, 1)	(1, 1, 1, 12)	3309.528507
27	(0, 1, 1)	(0, 1, 1, 12)	3311.572864
31	(0, 1, 1)	(1, 1, 1, 12)	3313.571914
59	(1, 1, 1)	(0, 1, 1, 12)	3313.572434
...	...	...	...
10	(0, 0, 1)	(0, 1, 0, 12)	3788.801984
50	(1, 1, 0)	(0, 1, 0, 12)	3803.412889
2	(0, 0, 0)	(0, 1, 0, 12)	3858.731295
0	(0, 0, 0)	(0, 0, 0, 12)	3946.630556
18	(0, 1, 0)	(0, 1, 0, 12)	3955.726645

64 rows × 3 columns

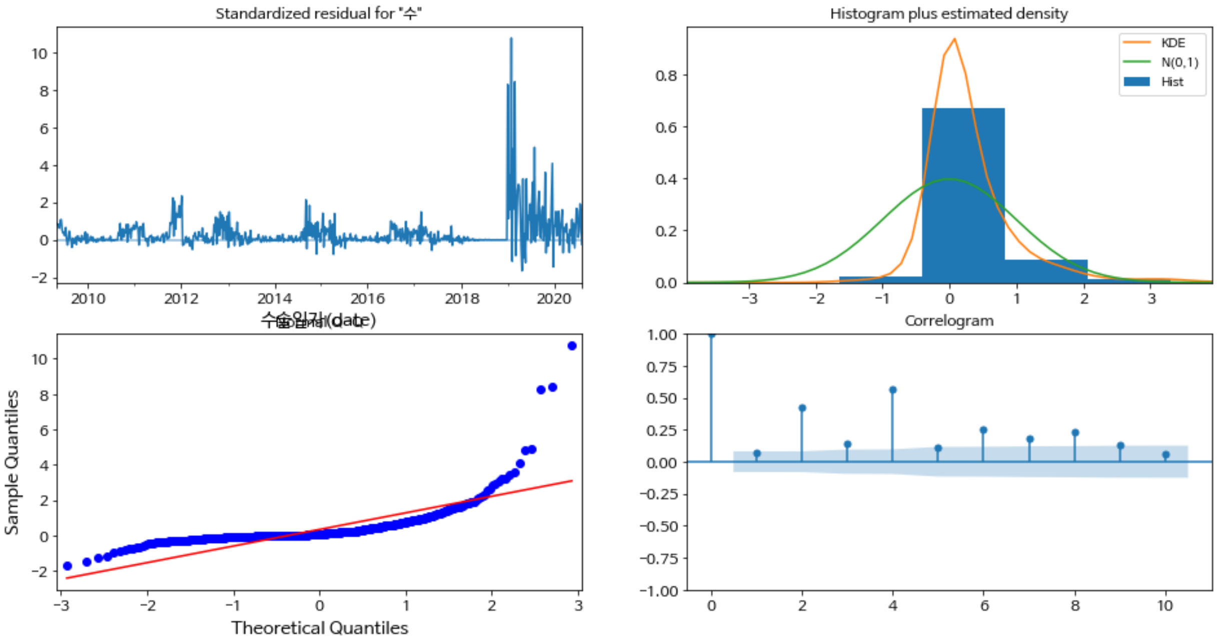
In [39]:
```
mod = tsa.statespace.SARIMAX(w1,order=(0, 0, 1),seasonal_order=(0, 0, 1, 12),
                              enforce_stationarity=False, enforce_invertibility=False)
results = mod.fit()
print(results.summary())
```

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                           수술일자(count)   No. Observations:                  60
3
Model:             SARIMAX(0, 0, 1)x(0, 0, 1, 12)   Log Likelihood                 -1806.901
```

```
Date:                       Sun, 21 Nov 2021   AIC                          3619.803
Time:                               11:49:57   BIC                          3632.938
Sample:                           01-25-2009   HQIC                         3624.920
                                - 08-09-2020
Covariance Type:                         opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1          0.3745      0.025     15.049      0.000       0.326       0.423
ma.S.L12       0.2912      0.022     13.048      0.000       0.247       0.335
sigma2        26.9921      0.343     78.595      0.000      26.319      27.665
===================================================================================
Ljung-Box (L1) (Q):                   2.92   Jarque-Bera (JB):             51921.79
Prob(Q):                              0.09   Prob(JB):                         0.00
Heteroskedasticity (H):               7.52   Skew:                             5.41
Prob(H) (two-sided):                  0.00   Kurtosis:                        47.71
===================================================================================

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

In [40]:

```python
results.plot_diagnostics(figsize=(16, 8))
plt.show()
```



In [41]:

```python
results.get_prediction()
```

Out[41]:

```
<statsmodels.tsa.statespace.mlemodel.PredictionResultsWrapper at 0x7fb375bd90a0>
```

In [42]:

```python
w1.head()
```

Out[42]:

```
수술일자(date)
2009-01-25    1
2009-02-01    1
2009-02-08    0
2009-02-15    0
```

```
2009-02-22    0
Freq: W-SUN, Name: 수술일자(count), dtype: int64
```

In [43]:

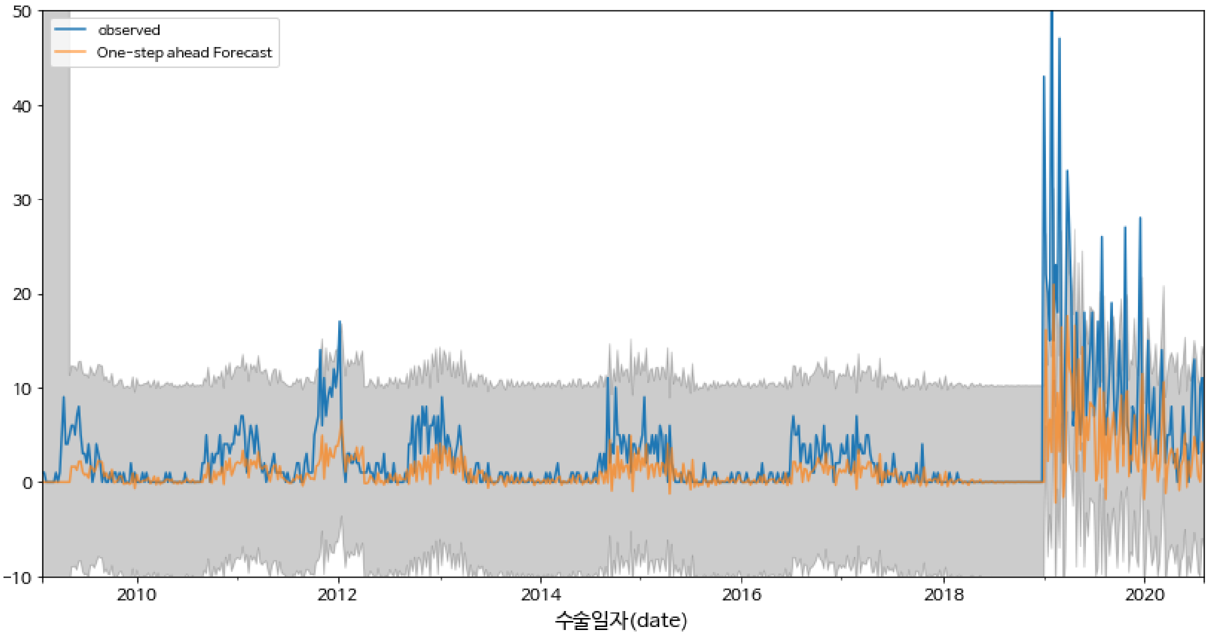
```
w1
```

Out[43]:

```
수술일자(date)
2009-01-25     1
2009-02-01     1
2009-02-08     0
2009-02-15     0
2009-02-22     0
              ..
2020-07-12     4
2020-07-19     3
2020-07-26    10
2020-08-02    11
2020-08-09     2
Freq: W-SUN, Name: 수술일자(count), Length: 603, dtype: int64
```

In [44]:

```
pred = results.get_prediction(start=pd.to_datetime('2009-01-25'), dynamic=False)
pred_ci = pred.conf_int()

plt.ylim([-10,50])
ax = w1.plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 7))
ax.fill_between(pred_ci.index,pred_ci.iloc[:, 0],pred_ci.iloc[:, 1], color='k', alpha=.2)

plt.legend()
plt.show()
```



In [45]:

```
pred = results.get_prediction(start=pd.to_datetime('2009-01-25'), dynamic=False)
pd.DataFrame(pred.predicted_mean).reset_index()
```

Out[45]:

	수술일자(date)	predicted_mean
0	2009-01-25	0.000000e+00
1	2009-02-01	0.000000e+00

	수술일자(date)	predicted_mean
2	2009-02-08	1.010906e-05
3	2009-02-15	-1.021927e-10
4	2009-02-22	1.033069e-15
...	...	...
598	2020-07-12	3.313627e+00
599	2020-07-19	6.126668e-01
600	2020-07-26	4.346580e-02
601	2020-08-02	4.181768e+00
602	2020-08-09	3.223006e+00

603 rows × 2 columns

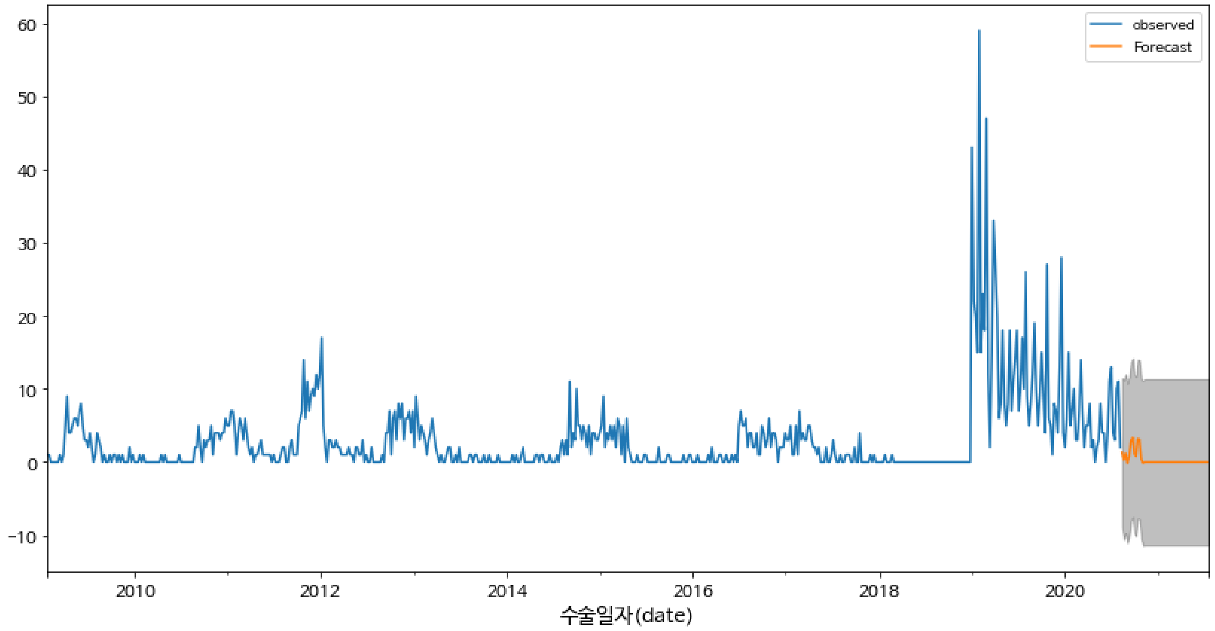
In [46]:

```
w_forecasted = pred.predicted_mean
w_truth = w['2009-01-01':]
mse = ((w_forecasted - w_truth) ** 2).sum()
print('MSE {}'.format(round(mse, 2)))
```

```
MSE 16057.25
```

In [47]:

```
pred_uc = results.get_forecast(steps=50)
pred_ci = pred_uc.conf_int() #추정된 계수의 신뢰구간 계산

ax = w1.plot(label='observed', figsize=(14, 7))
pred_uc.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.25)

plt.legend()
plt.show()
```



3. 구간 조정

In [48]:
```
final2 = final[final['수술일자(date)'] > '2017-01-01']
```

In [49]:
```
final2.head()
```

Out[49]:

	환자 ID	전방 디스크높이 (mm)	후방 디스크높이 (mm)	지방축적도	Instability	MF + ES	Modic change	PI	PT	Seg Angle(raw)	...	흡연여부	통증기간 (월)	수술기법
0	1PT	16.1	12.3	282.3	0	1824.6	3	51.6	36.6	14.4	...	0	1.0	TELI
1	2PT	13.7	6.4	177.3	0	1737.5	0	40.8	7.2	17.8	...	0	1.0	TELI
2	3PT	13.6	7.4	256.8	0	1188.5	0	67.5	27.3	10.2	...	0	1.0	TELI
3	4PT	10.6	7.3	250.1	0	2534.5	0	49.2	18.7	19.9	...	0	2.0	TELI
4	5PT	17.1	8.1	232.2	0	1840.6	0	58.8	14.7	5.2	...	0	1.0	TELI

5 rows × 53 columns

1W : 1주 단위 구간

- 특정 주기 단위로 분할 : 1주 단위구간

In [50]:
```
#수술 일자 관련
final2['수술일자(date)'] = pd.to_datetime(final2['수술일자'], format='%Y%m%d')
```

```
<ipython-input-50-6b359c67bb25>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
  final2['수술일자(date)'] = pd.to_datetime(final2['수술일자'], format='%Y%m%d')
```

In [51]:
```
final2['수술일자(date)']
```

Out[51]:
```
0       2019-07-15
1       2019-07-16
2       2019-07-31
3       2019-08-02
4       2019-09-06
           ...
1889    2017-04-07
1890    2017-04-27
1891    2017-04-11
1892    2017-04-10
1893    2017-04-12
Name: 수술일자(date), Length: 1033, dtype: datetime64[ns]
```

In [52]:

```
final2['수술일자(date)'].unique
```

Out[52]:

```
<bound method Series.unique of 0       2019-07-15
1       2019-07-16
2       2019-07-31
3       2019-08-02
4       2019-09-06
           ...
1889    2017-04-07
1890    2017-04-27
1891    2017-04-11
1892    2017-04-10
1893    2017-04-12
Name: 수술일자(date), Length: 1033, dtype: datetime64[ns]>
```

In [53]:

```
final2['수술일자(count)'] = 1
final2.head()
```

```
<ipython-input-53-3bfdebe40e19>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
  final2['수술일자(count)'] = 1
```

Out[53]:

	환자 ID	전방 디스크높이 (mm)	후방 디스크높이 (mm)	지방축적도	Instability	MF + ES	Modic change	PI	PT	Seg Angle(raw)	...	흡연여부	통증기간 (월)	수술기
0	1PT	16.1	12.3	282.3	0	1824.6	3	51.6	36.6	14.4	...	0	1.0	TELI
1	2PT	13.7	6.4	177.3	0	1737.5	0	40.8	7.2	17.8	...	0	1.0	TELI
2	3PT	13.6	7.4	256.8	0	1188.5	0	67.5	27.3	10.2	...	0	1.0	TELI
3	4PT	10.6	7.3	250.1	0	2534.5	0	49.2	18.7	19.9	...	0	2.0	TELI
4	5PT	17.1	8.1	232.2	0	1840.6	0	58.8	14.7	5.2	...	0	1.0	TELI

5 rows × 53 columns

In [54]:

```
# 수술 일자 count
final2_series = pd.pivot_table(data=final2, index='수술일자(date)', values= '수술일자(count)'
final2_series
```

Out[54]:

	수술일자(count)
수술일자(date)	

	수술일자(count)
수술일자(date)	
2017-01-02	1
2017-01-05	2
2017-01-11	1
2017-01-13	2
2017-01-16	2
...	...
2020-07-29	1
2020-07-30	3
2020-07-31	5
2020-08-04	1
2020-08-06	1

403 rows × 1 columns

In [55]:

```
w = final2_series['수술일자(count)'].resample('1W').sum()
```

In [56]:

```
predicted_sumw1 = w.fillna(0)
```

In [57]:

```
w1
```

Out[57]:

```
수술일자(date)
2009-01-25     1
2009-02-01     1
2009-02-08     0
2009-02-15     0
2009-02-22     0
              ..
2020-07-12     4
2020-07-19     3
2020-07-26    10
2020-08-02    11
2020-08-09     2
Freq: W-SUN, Name: 수술일자(count), Length: 603, dtype: int64
```

In [58]:

```
w1.unique()
```

Out[58]:

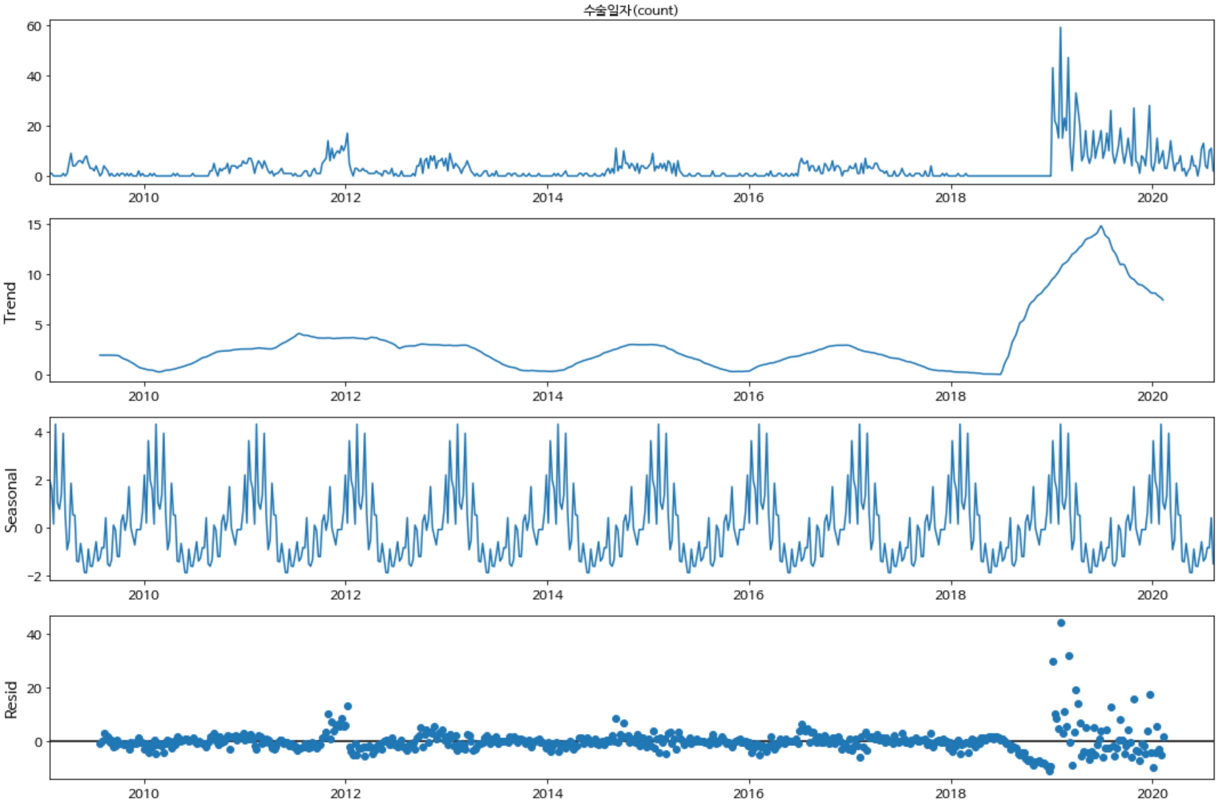
```
array([ 1,  0,  5,  9,  4,  6,  7,  8,  3,  2, 14, 11, 10, 12, 17, 43, 22,
       20, 15, 59, 23, 18, 47, 33, 27, 26, 19, 13, 28])
```

In [59]:

```
rcParams['figure.figsize'] = 15,10

#차트 기본 크기 설정
mpl.rcParams['axes.labelsize'] = 14
mpl.rcParams['xtick.labelsize'] = 12
mpl.rcParams['ytick.labelsize'] = 12
mpl.rcParams['text.color'] = 'k'
```

In [60]:

```
model_series = tsa.seasonal_decompose(w1, model='predicted_sumpredicted_sumpredicted_sumpred
fig = model_series.plot()
plt.show()
```



In [61]:

```
import itertools # 반복수를 만드는 라이브러리
```

In [62]:

```
p = d = q = range(0, 2)
pdq = list(itertools.product(p, d, q))
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]
```

In [63]:

```
seasonal_pdq
```

Out[63]:

```
[(0, 0, 0, 12),
 (0, 0, 1, 12),
 (0, 1, 0, 12),
 (0, 1, 1, 12),
 (1, 0, 0, 12),
 (1, 0, 1, 12),
 (1, 1, 0, 12),
 (1, 1, 1, 12)]
```

In [64]:

```
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
```

```
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

In [65]:

```
param_list = []
param_seasonal_list = []
results_AIC_list = []
```

1주일

In [66]:

```
import statsmodels.tsa.api as tsa
for param in pdq:
     for param_seasonal in seasonal_pdq:
        try:
            mod = tsa.statespace.SARIMAX(w1,order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)
            results = mod.fit()
            param_list.append(param)
            param_seasonal_list.append(param_seasonal)
            results_AIC_list.append(results.aic)
        except:
             continue
```

In [67]:

```
ARIMA_list = pd.DataFrame({'Parameter':param_list,'Seasonal':param_seasonal_list,'AIC':resul
ARIMA_list.to_excel('arima_model_list.xlsx')
```

In [68]:

```
ARIMA_list.sort_values(by='AIC')
```

Out[68]:

	Parameter	Seasonal	AIC
43	(1, 0, 1)	(0, 1, 1, 12)	3307.531619
47	(1, 0, 1)	(1, 1, 1, 12)	3309.528507
27	(0, 1, 1)	(0, 1, 1, 12)	3311.572864
31	(0, 1, 1)	(1, 1, 1, 12)	3313.571914
59	(1, 1, 1)	(0, 1, 1, 12)	3313.572434
...	...	...	...
10	(0, 0, 1)	(0, 1, 0, 12)	3788.801984
50	(1, 1, 0)	(0, 1, 0, 12)	3803.412889
2	(0, 0, 0)	(0, 1, 0, 12)	3858.731295
0	(0, 0, 0)	(0, 0, 0, 12)	3946.630556
18	(0, 1, 0)	(0, 1, 0, 12)	3955.726645

64 rows × 3 columns

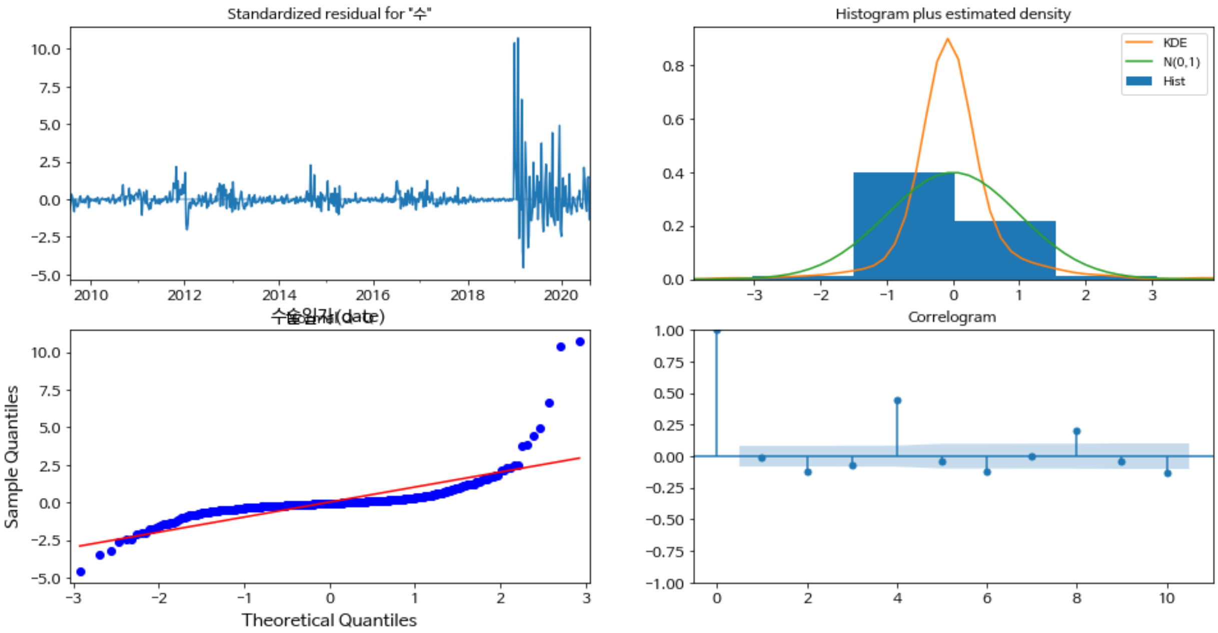
In [69]:

```
mod = tsa.statespace.SARIMAX(w1,order=(1, 0, 1),seasonal_order=(0, 1, 1, 12),
                              enforce_stationarity=False, enforce_invertibility=False)
results = mod.fit()
print(results.summary())
```

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                           수술일자(count)   No. Observations:                60
3
Model:             SARIMAX(1, 0, 1)x(0, 1, 1, 12)   Log Likelihood               -1649.766
Date:                            Sun, 21 Nov 2021   AIC                           3307.532
Time:                                    11:50:26   BIC                           3324.963
Sample:                                01-25-2009   HQIC                          3314.329
                                     - 08-09-2020
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.9551      0.007    127.623      0.000       0.940       0.970
ma.L1         -0.6857      0.020    -34.145      0.000      -0.725      -0.646
ma.S.L12      -0.9280      0.017    -53.720      0.000      -0.962      -0.894
sigma2        17.3256      0.230     75.297      0.000      16.875      17.777
===================================================================================
Ljung-Box (L1) (Q):                   0.04   Jarque-Bera (JB):             56387.83
Prob(Q):                              0.84   Prob(JB):                         0.00
Heteroskedasticity (H):              10.58   Skew:                             4.79
Prob(H) (two-sided):                  0.00   Kurtosis:                        50.47
===================================================================================

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

In [70]:

```
results.plot_diagnostics(figsize=(16, 8))
plt.show()
```



In [71]:

```
results.get_prediction()
```

Out[71]:

```
<statsmodels.tsa.statespace.mlemodel.PredictionResultsWrapper at 0x7fb375f28f70>
```

In [72]:

```
w1.head()
```

Out[72]:

```
수술일자(date)
2009-01-25    1
2009-02-01    1
2009-02-08    0
2009-02-15    0
2009-02-22    0
Freq: W-SUN, Name: 수술일자(count), dtype: int64
```

In [73]:

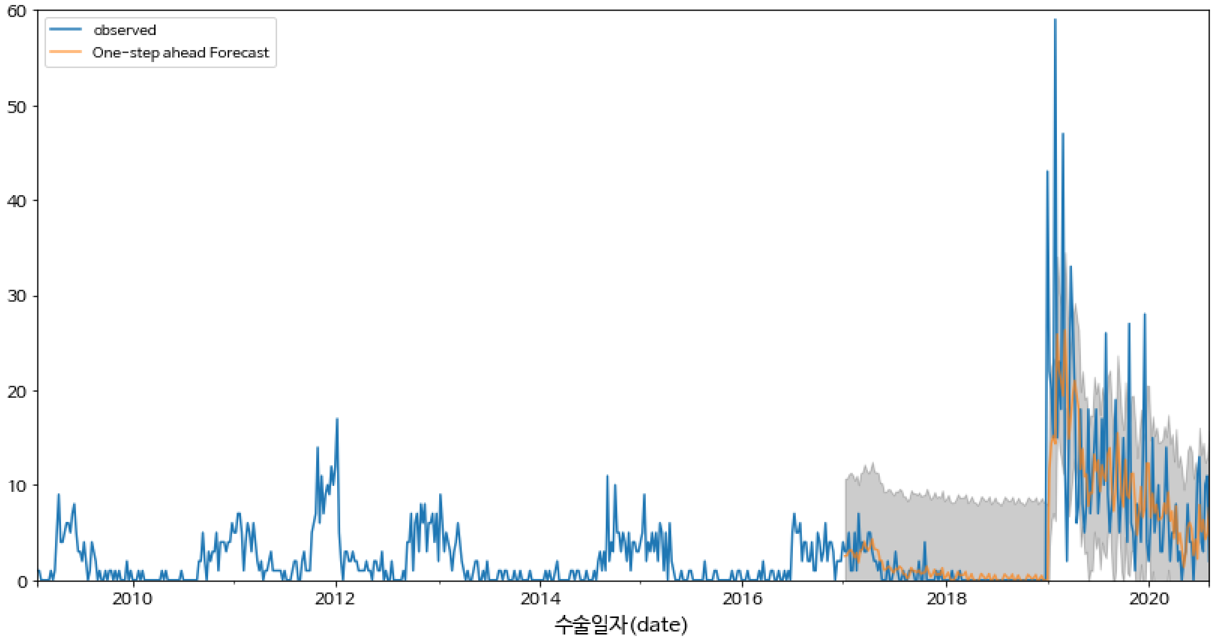
```
w1
```

Out[73]:

```
수술일자(date)
2009-01-25     1
2009-02-01     1
2009-02-08     0
2009-02-15     0
2009-02-22     0
              ..
2020-07-12     4
2020-07-19     3
2020-07-26    10
2020-08-02    11
2020-08-09     2
Freq: W-SUN, Name: 수술일자(count), Length: 603, dtype: int64
```

In [74]:

```
pred = results.get_prediction(start=pd.to_datetime('2017-01-08'), dynamic=False)
pred_ci = pred.conf_int()

plt.ylim([0,60])
ax = w1.plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 7))
ax.fill_between(pred_ci.index,pred_ci.iloc[:, 0],pred_ci.iloc[:, 1], color='k', alpha=.2)

plt.legend()
plt.show()
```



In [75]:

```
pred = results.get_prediction(start=pd.to_datetime('2017-01-08'), dynamic=False)
pd.DataFrame(pred.predicted_mean).reset_index()
```

Out[75]:

	수술일자(date)	predicted_mean
0	2017-01-08	2.545497
1	2017-01-15	2.579354
2	2017-01-22	3.026723
3	2017-01-29	3.191478
4	2017-02-05	2.923000
...	...	...
183	2020-07-12	5.106641
184	2020-07-19	6.333528
185	2020-07-26	4.271733
186	2020-08-02	4.817843
187	2020-08-09	7.609526

188 rows × 2 columns

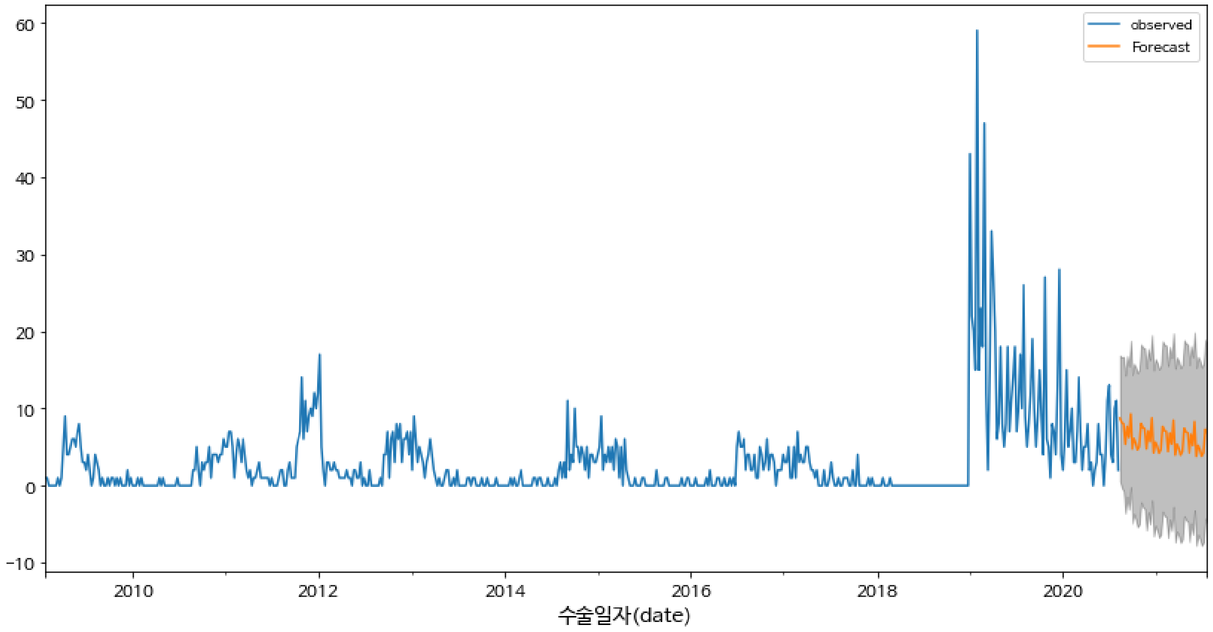
In [76]:

```
w_forecasted = pred.predicted_mean
w_truth = w['2009-01-01':]
mse = ((w_forecasted - w_truth) ** 2).sum()
print('MSE {}'.format(round(mse, 2)))
```

```
MSE 8678.31
```

In [77]:

```
pred_uc = results.get_forecast(steps=50)
pred_ci = pred_uc.conf_int() #추정된 계수의 신뢰구간 계산

ax = w1.plot(label='observed', figsize=(14, 7))
pred_uc.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.25)

plt.legend()
plt.show()
```



신규 수술 건수 유치 필요

2018년 초부터 수술 건수가 급등하는데, 2020년 이후부터 감소

의사스케줄 조절 및 급등하는 수술 건수 대응 했어야 하는데 못해서 지금 일정하게 유지하는 거 같다.

그러니까 사전 대비를 해서

의사 피로도 /

- 실제 왜 줄었는지 찾아보기 :2018년 6~7월 급증 // 2019년 6~7월 감소 (1년 동안) : 우리 병원 문제인지 / 사회적으로 뭐가 있었는지..?

2017년도 이후 부터의 요일별 분석

In [78]:

```
final3 = final[final['수술일자(date)'] > '2017-01-01']
```

In [79]:

```
#수술 일자 관련
final3['수술일자(date)'] = pd.to_datetime(final3['수술일자'], format='%Y%m%d')
```

```
<ipython-input-79-935124a90e03>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
  final3['수술일자(date)'] = pd.to_datetime(final3['수술일자'], format='%Y%m%d')
```

In [80]:

```
from datetime import datetime, timedelta
def 요일(date_time):
    s = str(date_time)
    days = ['월','화','수','목','금','토','일']
    date = datetime(year=int(s[0:4]), month=int(s[4:6]), day=int(s[6:8]))
    return days[date.weekday()]
```

In [81]:

```
final3['수술일자(weekday)'] = final['수술일자'].apply(요일)
```

```
<ipython-input-81-fdc1cdf2488c>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
  final3['수술일자(weekday)'] = final['수술일자'].apply(요일)
```

In [82]:

```
final3
```

Out[82]:

	환자ID	전방 디스크높이 (mm)	후방 디스크높이 (mm)	지방축적도	Instability	MF + ES	Modic change	PI	PT	Seg Angle(raw)	...	통증기간 (월)

	환자ID	전방디스크높이(mm)	후방디스크높이(mm)	지방축적도	Instability	MF + ES	Modic change	PI	PT	Seg Angle(raw)	...	통증기간(월)	
0	1PT	16.1	12.3	282.3	0	1824.6	3	51.6	36.6	14.4	...	1.0	T
1	2PT	13.7	6.4	177.3	0	1737.5	0	40.8	7.2	17.8	...	1.0	T
2	3PT	13.6	7.4	256.8	0	1188.5	0	67.5	27.3	10.2	...	1.0	T
3	4PT	10.6	7.3	250.1	0	2534.5	0	49.2	18.7	19.9	...	2.0	T
4	5PT	17.1	8.1	232.2	0	1840.6	0	58.8	14.7	5.2	...	1.0	T
...	...	...	...	...	...	...	...	...	...	...	...	...	
1889	1890PT	17.0	10.7	237.5	0	2795.7	2	59.5	23.0	21.8	...	12.0	
1890	1891PT	9.4	8.2	288.0	0	1473.0	0	47.7	20.2	5.0	...	6.0	
1891	1892PT	13.5	5.5	148.5	0	3864.1	0	44.6	15.0	17.4	...	1.0	
1892	1893PT	14.0	10.0	89.0	0	2481.8	2	32.2	11.1	17.7	...	24.0	
1893	1894PT	16.1	9.5	251.4	0	1796.1	0	38.9	6.8	27.8	...	6.0	

1033 rows × 54 columns

In [83]:

```
data_weekdays = final3.groupby(by=['수술일자(weekday)']).sum()
print(data_weekdays)
```

```
                 전방디스크높이(mm)  후방디스크높이(mm)     지방축적도  Instability     MF + ES  \
수술일자(weekday)
금                   1739.41       1196.54  36412.25            6  309312.21
목                   2313.05       1669.85  44377.94           10  429639.93
수                   1920.19       1415.98  33489.85            6  340388.55
월                   1669.92       1285.18  30675.68            5  308651.21
일                    981.89        672.08  28037.91            4  183389.39
토                   1397.95       1002.35  32460.54            5  254776.22
화                   1902.07       1409.04  39003.04            7  348227.17

                Modic change      PI      PT  Seg Angle(raw)  Vaccum disc  ...  \
수술일자(weekday)                                                          ...
금                         30  7540.5  2382.4         2336.43            6  ...
목                         57  9301.5  3243.7         3041.13           16  ...
수                         49  7365.3  2677.0         2518.30            5  ...
월                         46  6657.9  2351.9         2183.11            7  ...
일                         23  4468.0  1424.0         1378.80            9  ...
토                         28  5382.8  1937.2         1910.53            5  ...
화                         46  7358.3  2560.5         2667.25           12  ...
```

```
               헤모글로빈수치  혈전합병증여부  환자통증정도  흡연여부  통증기간(월)      수술시간  
수술실패여부  \
수술일자(weekday)
금              2196.81         0    1046    28   266.00  11184.0        7
목              2990.09         0    1396    46   487.75  14219.0       12
수              2320.77         0    1128    40   277.25  10822.0       10
월              2130.38         0    1030    31   321.75   9778.0        7
일              1229.19         0     585    24   110.00   5614.0        8
토              1803.41         0     894    20   179.00   8345.0       10
화              2401.23         2    1112    39   365.00  10705.0        5

                     수술일자  재발여부  수술일자(count)
수술일자(weekday)
금              3068747683    23          152
목              4139051544    22          205
수              3230367851    25          160
월              2927634201    18          145
일              1716376598     9           85
토              2463482302    15          122
화              3311372246    22          164

[7 rows x 47 columns]
```

In [84]:

```
# 수술 일자 count
count_1= pd.pivot_table(data=final3, index='수술일자(weekday)', values= '수술일자(count)', agg
count_1
```

Out[84]:

	수술일자(count)
수술일자(weekday)	
금	152
목	205
수	160
월	145
일	85
토	122
화	164

In [85]:

```
# data_weekdays_time = data_weekdays['수술건수']
```

In [86]:

```
final3['수술월(month)'] = final3['수술일자(date)'].dt.month
final3['수술연도(year)']= final3['수술일자(date)'].dt.year
```

```
<ipython-input-86-4a6d468a510b>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
```

```
e/indexing.html#returning-a-view-versus-a-copy
  final3['수술월(month)'] = final3['수술일자(date)'].dt.month
<ipython-input-86-4a6d468a510b>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
  final3['수술연도(year)']= final3['수술일자(date)'].dt.year
```

In [87]:

```
final3['수술연도(year)'].unique()
```

Out[87]:

```
array([2019, 2020, 2017, 2018])
```

2019,2020년도 한정 월별, 요일별 분석

In [88]:

```
cond1 = (final3['수술연도(year)']==2019)
cond2 = (final3['수술연도(year)']==2020)

final_year = final3.loc[cond1|cond2]
count_2= pd.pivot_table(data=final_year, index=['수술월(month)','수술일자(weekday)'], values=
count_3= count_2.reset_index()
```

In [89]:

```
count_3
```

Out[89]:

	수술월(month)	수술일자(weekday)	수술일자(count)
0	1	금	20
1	1	목	54
2	1	수	28
3	1	월	24
4	1	일	12
...	...	...	...
77	12	수	15
78	12	월	5
79	12	일	3
80	12	토	9
81	12	화	4

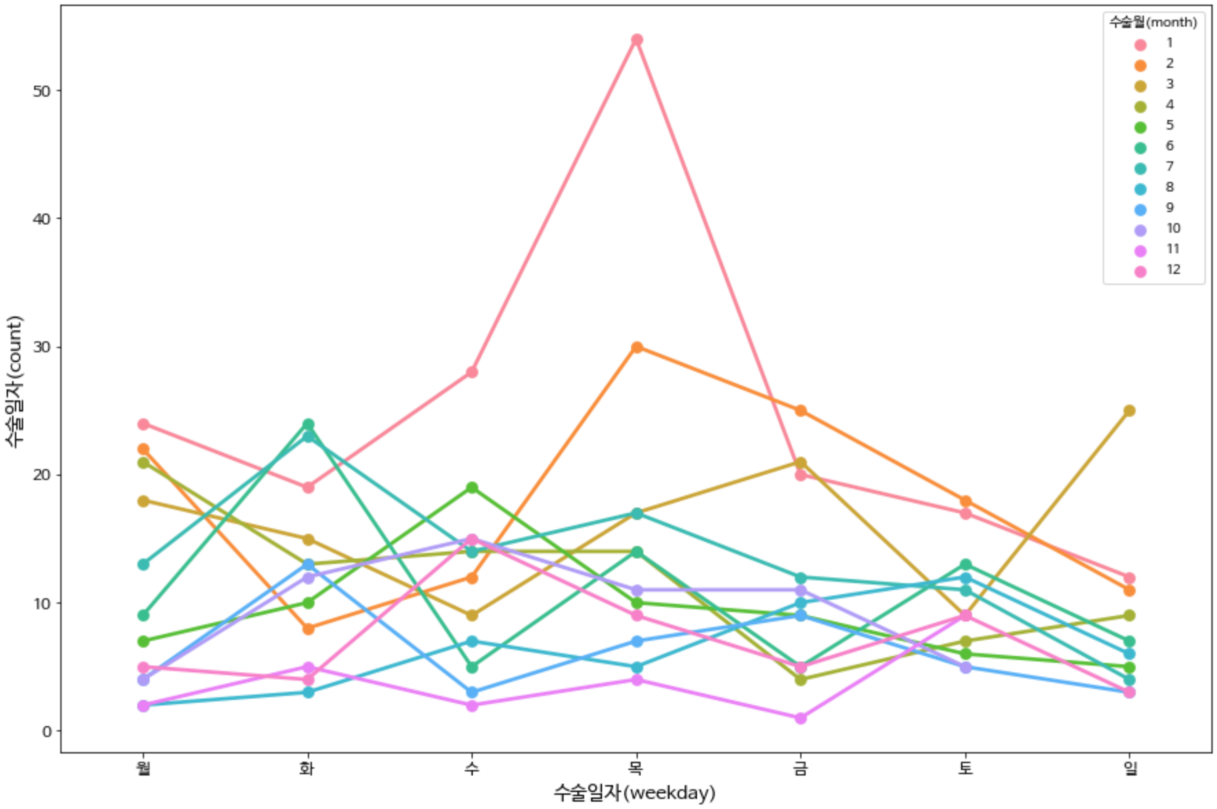
82 rows × 3 columns

In [90]:

```
sns.pointplot(data=count_3, x='수술일자(weekday)', y='수술일자(count)',hue='수술월(month)',
             order=['월','화','수','목','금','토','일'])
```

Out[90]:

```
<AxesSubplot:xlabel='수술일자(weekday)', ylabel='수술일자(count)'>
```



In [91]:

```
final3.columns
```

Out[91]:

```
Index(['환자ID', '전방디스크높이(mm)', '후방디스크높이(mm)', '지방축적도', 'Instability', 'MF +
ES',
       'Modic change', 'PI', 'PT', 'Seg Angle(raw)', 'Vaccum disc', '골밀도',
       '디스크단면적', '디스크위치', '척추이동척도', '척추전방위증', 'Large Lymphocyte',
       'Location of herniation', 'ODI', '가족력', '간질성폐질환', '고혈압여부', '과거수술횟수',
       '당뇨여부', '말초동맥질환여부', '빈혈여부', '성별', '스테로이드치료', '신부전여부', '신장',
'심혈관질환',
       '암발병여부', '연령', '우울증여부', '입원기간', '입원일자', '종양진행여부', '직업', '체중',
'퇴원일자',
       '헤모글로빈수치', '혈전합병증여부', '환자통증정도', '흡연여부', '통증기간(월)', '수술기법',
'수술시간',
       '수술실패여부', '수술일자', '재발여부', '혈액형', '수술일자(date)', '수술일자(count)',
       '수술일자(weekday)', '수술월(month)', '수술연도(year)'],
      dtype='object')
```

In [92]:

```
final3['직업'].unique()
```

Out[92]:

```
array(['자영업', '운동선수', '특수전문직', '주부', '사업가', nan, '건설업', '운수업', '사무직',
       '공무원', '농업', '의료직', '학생', '군인', '노동직', '교사', '예술가', '무직'],
      dtype=object)
```

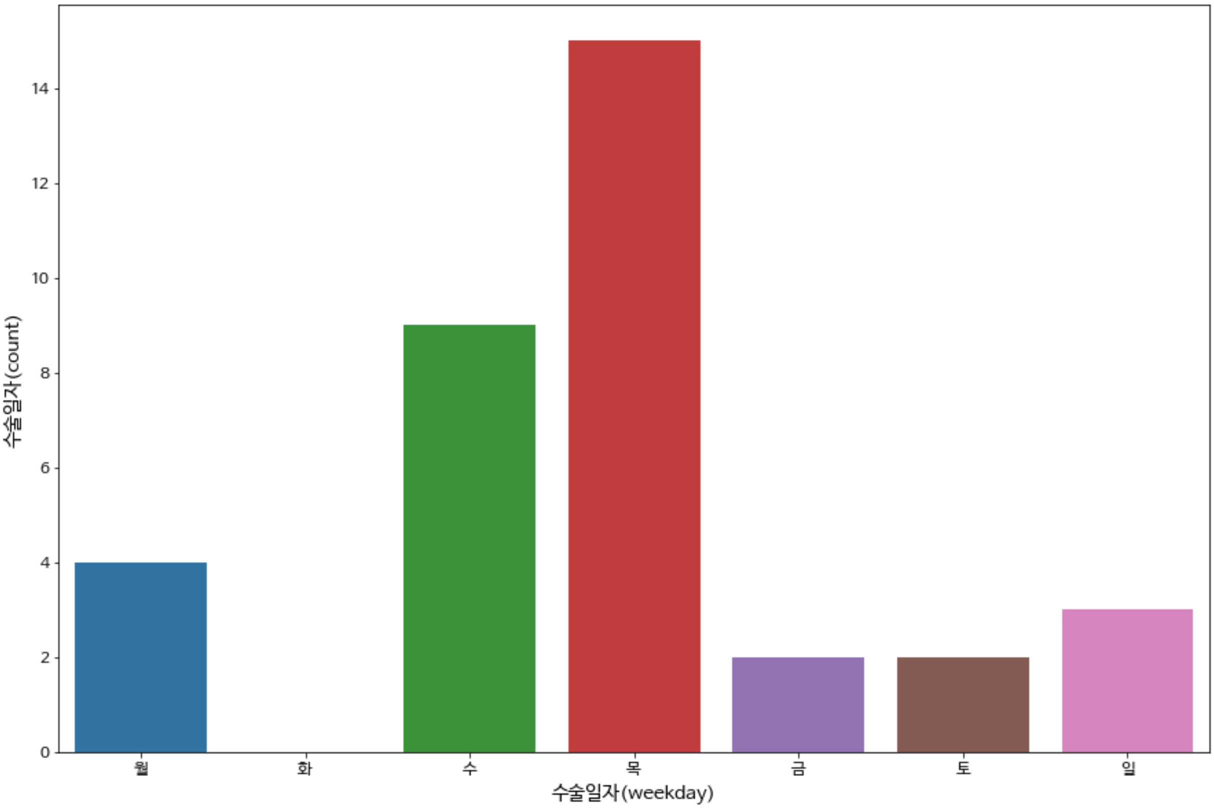
In [93]:

```
cond1 = (final3['수술연도(year)']==2019)
cond2 = (final3['수술연도(year)']==2020)
cond3 = (final3['직업']=='사무직')

final_year = final3.loc[(cond1|cond2)&cond3]

count_2= pd.pivot_table(data=final_year, index=['수술월(month)','수술일자(weekday)'], values=
count_3= count_2.reset_index()
```

```
cond4 = (count_3['수술월(month)']==1)
count_4 = count_3.loc[cond4]
sns.barplot(data=count_4, x='수술일자(weekday)', y='수술일자(count)',order=['월','화','수','목
```

Out[93]:

```
<AxesSubplot:xlabel='수술일자(weekday)', ylabel='수술일자(count)'>
```

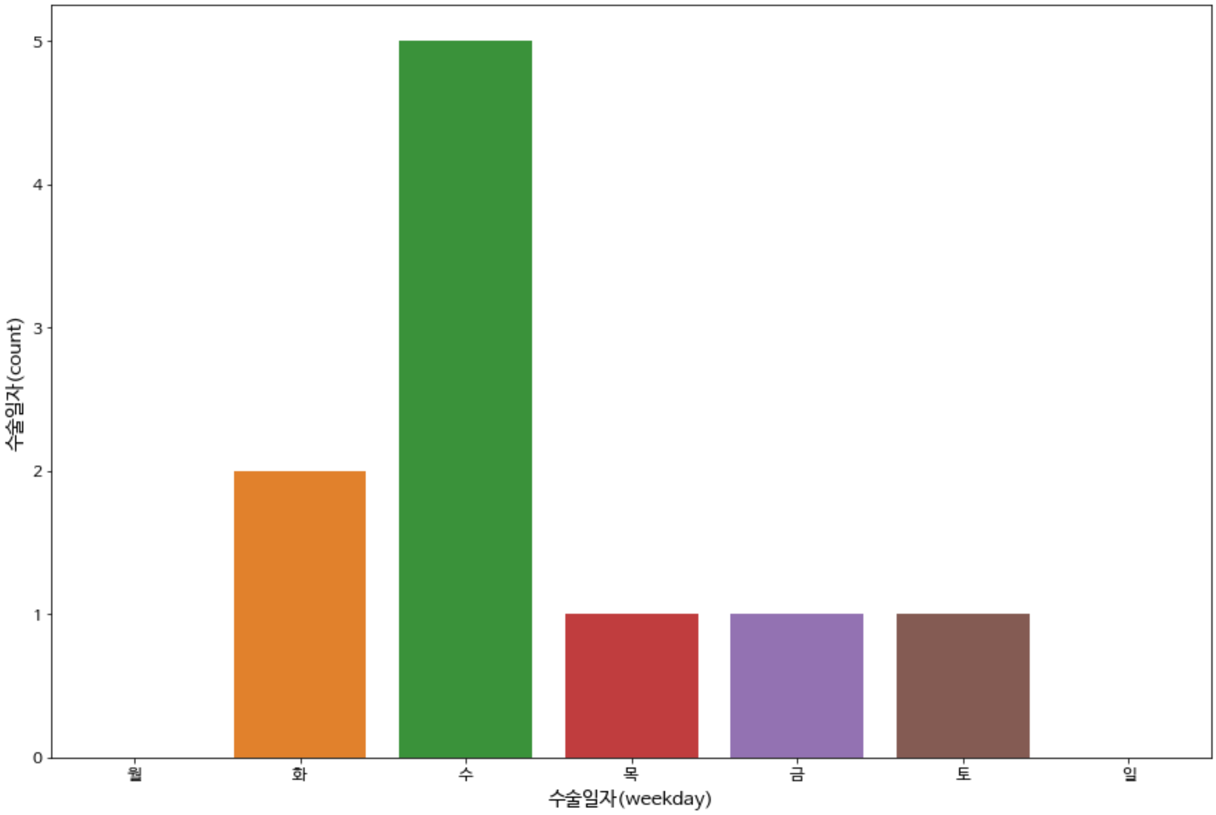


In [94]:

```
cond1 = (final3['수술연도(year)']==2019)
cond2 = (final3['수술연도(year)']==2020)
cond3 = (final3['직업']=='사무직')

final_year = final3.loc[(cond1|cond2)&cond3]

count_2= pd.pivot_table(data=final_year, index=['수술월(month)','수술일자(weekday)'], values=
count_3= count_2.reset_index()

cond4 = (count_3['수술월(month)']==10)
count_4 = count_3.loc[cond4]
sns.barplot(data=count_4, x='수술일자(weekday)', y='수술일자(count)',order=['월','화','수','목
```

Out[94]:

```
<AxesSubplot:xlabel='수술일자(weekday)', ylabel='수술일자(count)'>
```

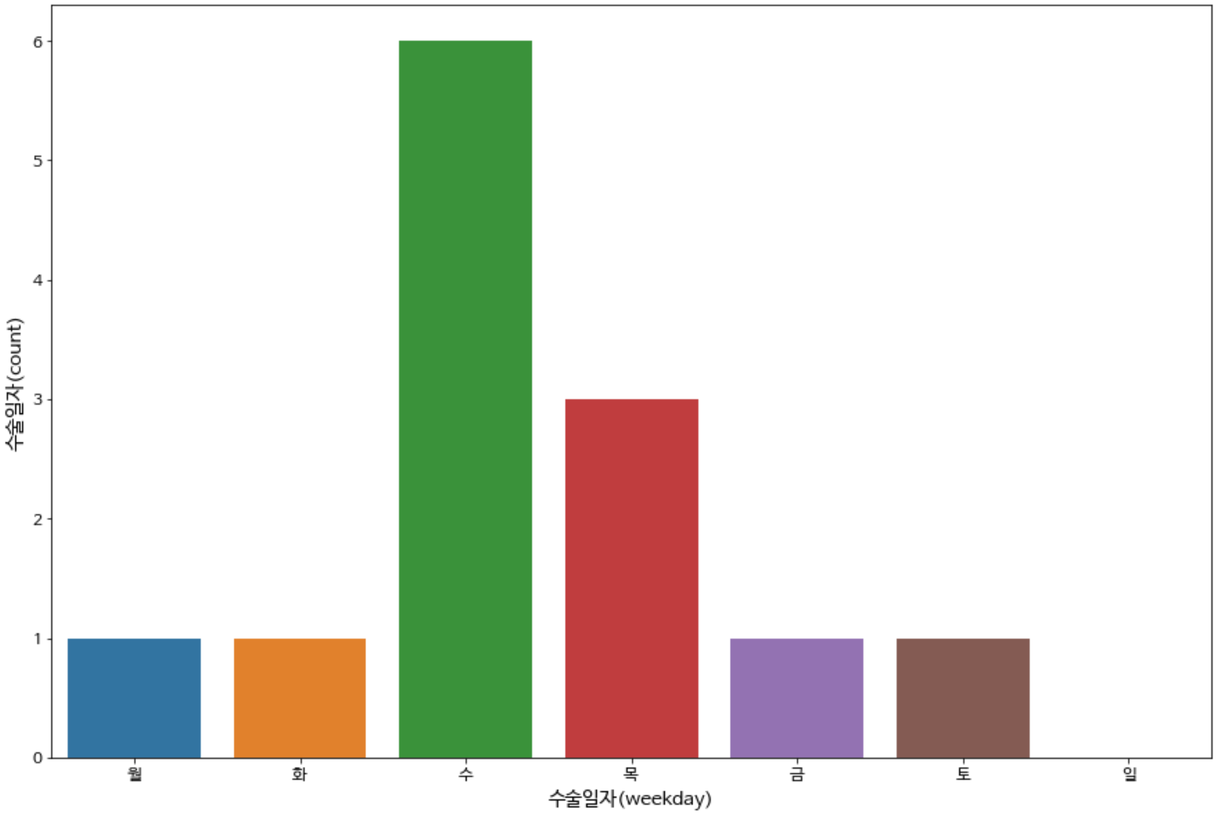


In [95]:

```
수술실패여부cond1 = (final3['수술연도(year)']==2019)
cond2 = (final3['수술연도(year)']==2020)
cond3 = (final3['직업']=='사무직')

final_year = final3.loc[(cond1|cond2)&cond3]

count_2= pd.pivot_table(data=final_year, index=['수술월(month)','수술일자(weekday)'], values=
count_3= count_2.reset_index()

cond4 = (count_3['수술월(month)']==12)
count_4 = count_3.loc[cond4]
sns.barplot(data=count_4, x='수술일자(weekday)', y='수술일자(count)',order=['월','화','수','목
```

Out[95]:

```
<AxesSubplot:xlabel='수술일자(weekday)', ylabel='수술일자(count)'>
```



전체 연도 - 요일별 수술 건수

전체 연도

In [96]:

```
final['수술월(month)'] = final['수술일자(date)'].dt.month
final['수술연도(year)']= final['수술일자(date)'].dt.year
```

In [97]:

```
final
```

Out[97]:

	환자ID	전방 디스크높이 (mm)	후방 디스크높이 (mm)	지방 축적도	Instability	MF + ES	Modic change	PI	PT	Seg Angle(raw)	...	수술 기법
0	1PT	16.1	12.3	282.3	0	1824.6	3	51.6	36.6	14.4	...	TELD
1	2PT	13.7	6.4	177.3	0	1737.5	0	40.8	7.2	17.8	...	TELD
2	3PT	13.6	7.4	256.8	0	1188.5	0	67.5	27.3	10.2	...	TELD
3	4PT	10.6	7.3	250.1	0	2534.5	0	49.2	18.7	19.9	...	TELD
4	5PT	17.1	8.1	232.2	0	1840.6	0	58.8	14.7	5.2	...	TELD
...	...	...	...	...	...	...	...	...	...	...	...	...
1889	1890PT	17.0	10.7	237.5	0	2795.7	2	59.5	23.0	21.8	...	NaN
1890	1891PT	9.4	8.2	288.0	0	1473.0	0	47.7	20.2	5.0	...	NaN

	환자ID	전방 디스크높이 (mm)	후방 디스크높이 (mm)	지방축적도	Instability	MF + ES	Modic change	PI	PT	Seg Angle(raw)	...	수술기법
1891	1892PT	13.5	5.5	148.5	0	3864.1	0	44.6	15.0	17.4	...	IELD
1892	1893PT	14.0	10.0	89.0	0	2481.8	2	32.2	11.1	17.7	...	NaN
1893	1894PT	16.1	9.5	251.4	0	1796.1	0	38.9	6.8	27.8	...	NaN

1894 rows × 55 columns

전체 년도

In [98]:

```
#수술 일자 관련
final['수술일자(date)'] = pd.to_datetime(final['수술일자'], format='%Y%m%d')
```

In [99]:

```
from datetime import datetime, timedelta
def 요일(date_time):
    s = str(date_time)
    days = ['월','화','수','목','금','토','일']
    date = datetime(year=int(s[0:4]), month=int(s[4:6]), day=int(s[6:8]))
    return days[date.weekday()]
```

In [100]:

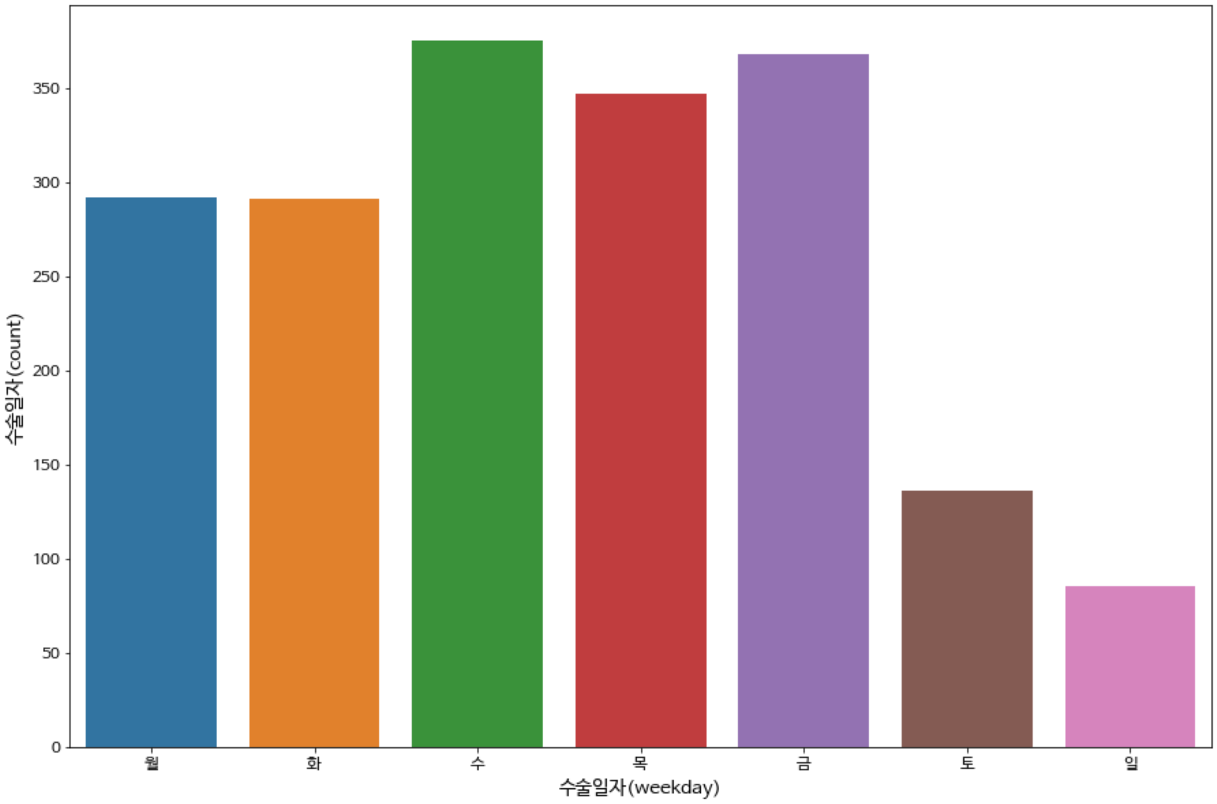
```
final['수술일자(weekday)'] = final['수술일자'].apply(요일)
```

In [101]:

```
count_2= pd.pivot_table(data=final, index=['수술일자(weekday)'], values= '수술일자(count)', ag
count_3= count_2.reset_index()

# count_4 = count_3.loc[cond4]
sns.barplot(data=count_3, x='수술일자(weekday)', y='수술일자(count)',order=['월','화','수','목
```

Out[101]:

```
<AxesSubplot:xlabel='수술일자(weekday)', ylabel='수술일자(count)'>
```



In [102]:

```
count_2
```

Out[102]:

수술일자(weekday)	수술일자(count)
금	368
목	347
수	375
월	292
일	85
토	136
화	291

년도별 꺾은선 그래프

In [103]:

```
count_2= pd.pivot_table(data=final, index=['수술연도(year)','수술일자(weekday)'], values= '수
count_3= count_2.reset_index()
```

In [104]:

```
count_3
```

Out[104]:

	수술연도(year)	수술일자(weekday)	수술일자(count)
0	2009	금	28
1	2009	목	15
2	2009	수	30

	수술연도(year)	수술일자(weekday)	수술일자(count)
3	2009	월	16
4	2009	토	3
...	...	...	...
63	2020	수	21
64	2020	월	30
65	2020	일	19
66	2020	토	25
67	2020	화	41

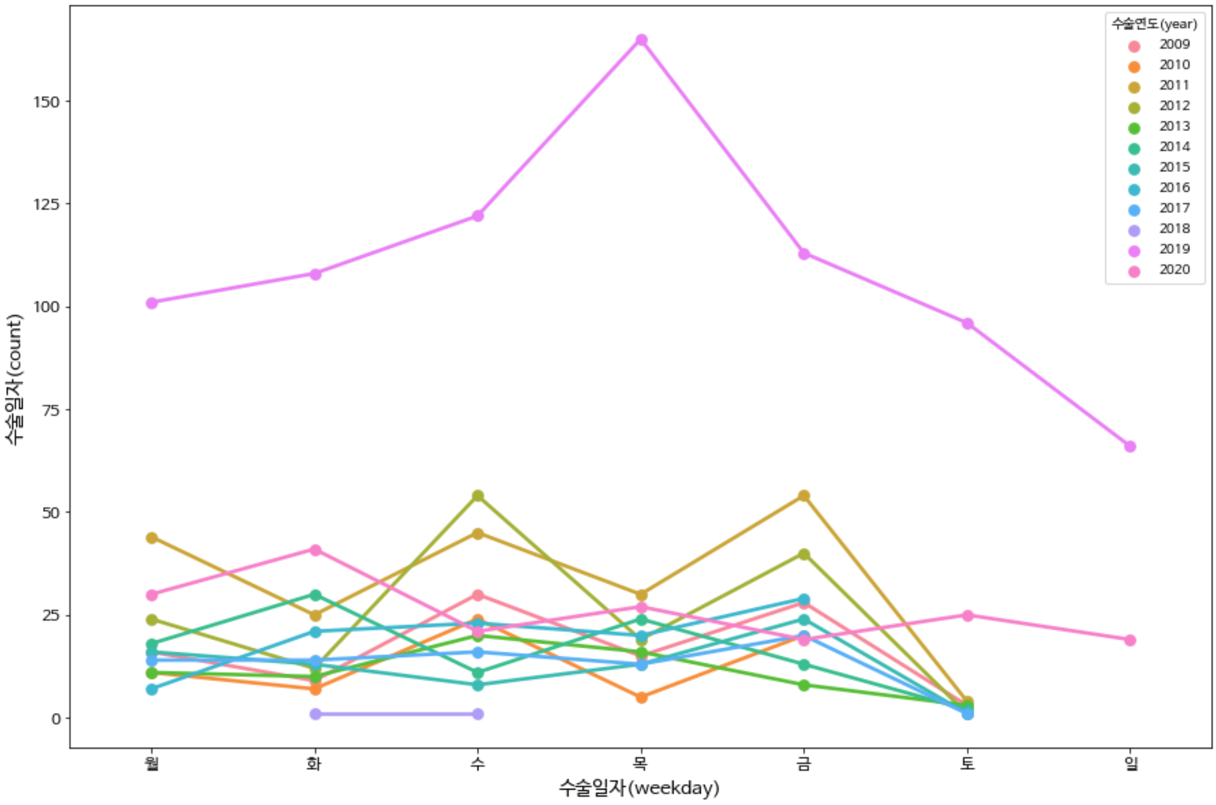
68 rows × 3 columns

In [105]:

```
sns.pointplot(data=count_3, x='수술일자(weekday)', y='수술일자(count)',hue='수술연도(year)',
              order=['월','화','수','목','금','토','일'])
```

Out[105]:

```
<AxesSubplot:xlabel='수술일자(weekday)', ylabel='수술일자(count)'>
```



In [106]:

```
import seaborn as sns

# 연령대 통증기간 긴지? -> 참아온것 이를 개선할 수 있는가?
def 연령(age):
  return age//10

final['연령대'] = final['연령'].apply(연령)
```

In [107]:

```
final
```

Out[107]:

	환자ID	전방디스크높이(mm)	후방디스크높이(mm)	지방축적도	Instability	MF + ES	Modic change	PI	PT	Seg Angle(raw)	...	수술실패여부	수
0	1PT	16.1	12.3	282.3	0	1824.6	3	51.6	36.6	14.4	...	0	201
1	2PT	13.7	6.4	177.3	0	1737.5	0	40.8	7.2	17.8	...	0	201
2	3PT	13.6	7.4	256.8	0	1188.5	0	67.5	27.3	10.2	...	0	201
3	4PT	10.6	7.3	250.1	0	2534.5	0	49.2	18.7	19.9	...	0	201
4	5PT	17.1	8.1	232.2	0	1840.6	0	58.8	14.7	5.2	...	0	201
...	...	...	...	...	...	...	...	...	...	...	...	...	
1889	1890PT	17.0	10.7	237.5	0	2795.7	2	59.5	23.0	21.8	...	0	201
1890	1891PT	9.4	8.2	288.0	0	1473.0	0	47.7	20.2	5.0	...	0	201
1891	1892PT	13.5	5.5	148.5	0	3864.1	0	44.6	15.0	17.4	...	0	201
1892	1893PT	14.0	10.0	89.0	0	2481.8	2	32.2	11.1	17.7	...	0	201
1893	1894PT	16.1	9.5	251.4	0	1796.1	0	38.9	6.8	27.8	...	0	201

1894 rows × 57 columns

In [109]:

```
pd.pivot_table(data=final, index=['연령대'], values= '수술일자(count)', aggfunc='sum')
```

Out[109]:

	수술일자(count)
연령대	
1	68
2	324
3	469
4	581
5	272
6	115
7	60
8	5