Modelling and Forecasting an Industrial Production with Simple Eponential Smoothing,Double Exponential Smoothing and Triple Exponential Smoothing.

In [113]:

```
import pandas as pd
import pandas as pd
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [114]:

```
df=pd.read_csv(r'C:\Users\chumj\Downloads\509982_941525_bundle_archive\INDPRODUCTION.csv',index_col
='DATE',parse_dates=True)
```

In [115]:

```
df.tail(5)
```

Out[115]:

	INDPRO
DATE	
2019-08-01	109.9634
2019-09-01	109.4437
2019-10-01	108.8532
2019-11-01	109.7573
2019-12-01	109.4330

In [116]:

```
df.dropna(inplace=True)
```

In [117]:

```
df.index
```

Out[117]:

```
DatetimeIndex(['1919-01-01', '1919-02-01', '1919-03-01', '1919-04-01',
               '1919-05-01', '1919-06-01', '1919-07-01', '1919-08-01',
               '1919-09-01', '1919-10-01',
               ...
               '2019-03-01', '2019-04-01', '2019-05-01', '2019-06-01',
               '2019-07-01', '2019-08-01', '2019-09-01', '2019-10-01',
               '2019-11-01', '2019-12-01'],
              dtype='datetime64[ns]', name='DATE', length=1212, freq=None)
```

In [119]:

```
df.index.freq='MS'
```

In [120]:

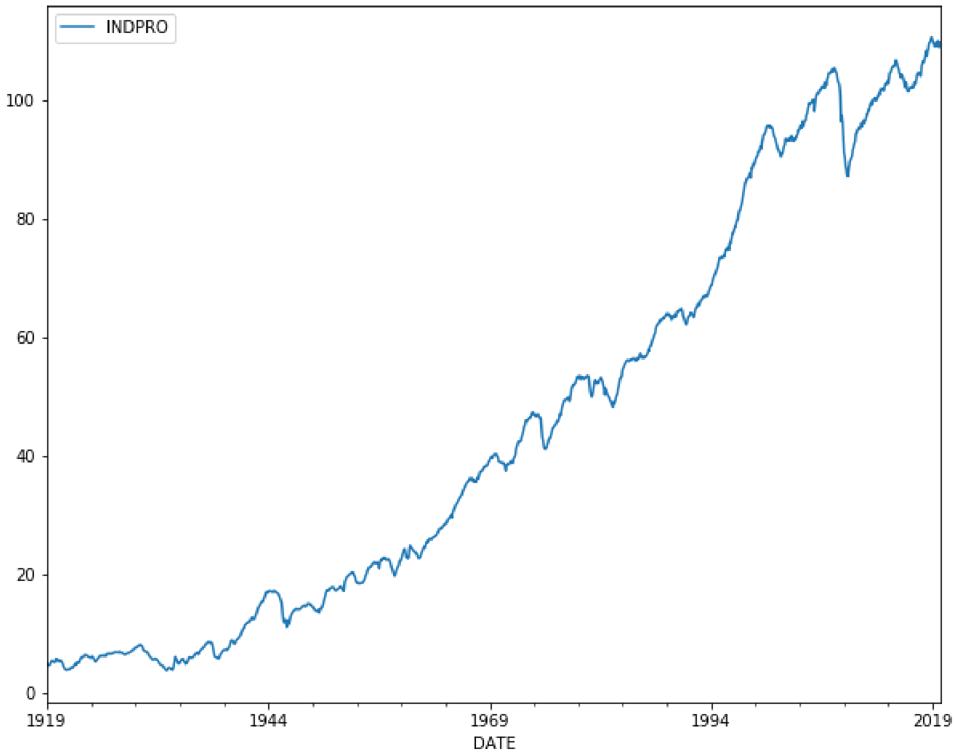
```
df.index
```

Out[120]:

```
DatetimeIndex(['1919-01-01', '1919-02-01', '1919-03-01', '1919-04-01',
               '1919-05-01', '1919-06-01', '1919-07-01', '1919-08-01',
               '1919-09-01', '1919-10-01',
               ...
               '2019-03-01', '2019-04-01', '2019-05-01', '2019-06-01',
               '2019-07-01', '2019-08-01', '2019-09-01', '2019-10-01',
               '2019-11-01', '2019-12-01'],
              dtype='datetime64[ns]', name='DATE', length=1212, freq='MS')
```

In [8]:

```
df.plot(figsize=(10,8));
```



In [9]:

```
#Using Hodrick Prescott filter to get Trend
from statsmodels.tsa.filters.hp_filter import hpfilter
```

In [10]:

```
df_trend,df_cyle=hpfilter(df['INDPRO'],lamb=1600)
```

In [11]:

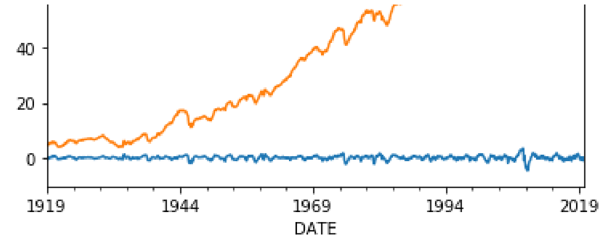
```
df['Trend']=df_trend
```

In [12]:

```
df[['Trend','INDPRO']].plot()
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12e0d562788>
```

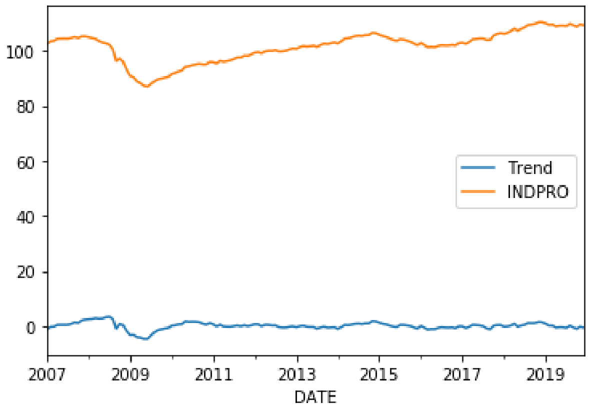


In [13]:

```
df[['Trend','INDPRO']]['2007-01-01':].plot()
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12e10dfcc08>
```



In [14]:

```
#Applying ETS
from statsmodels.tsa.seasonal import seasonal_decompose
```

In [15]:

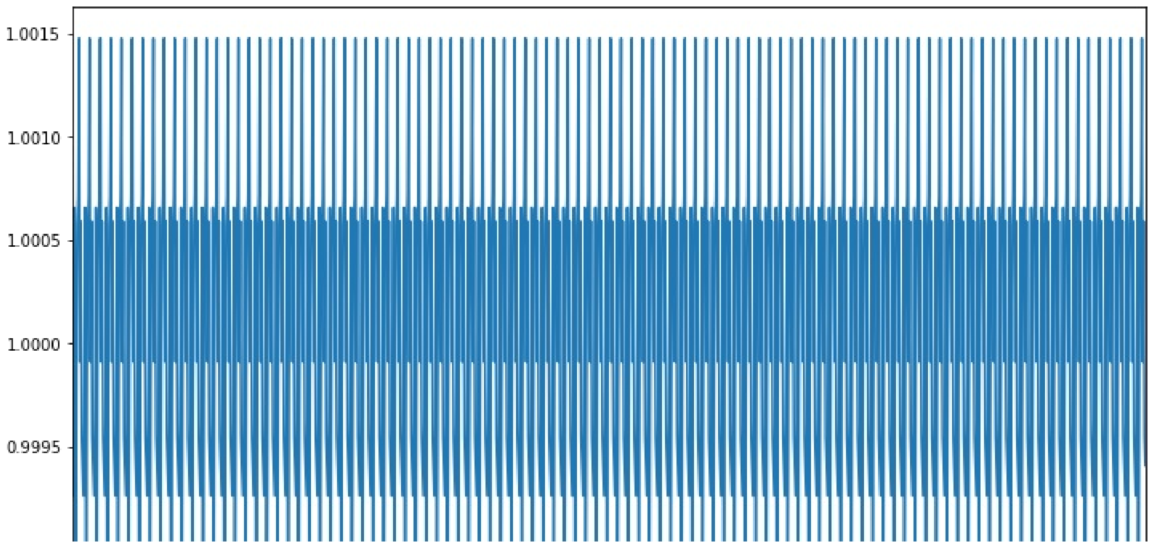
```
result=seasonal_decompose(df['INDPRO'],model='mul')
```

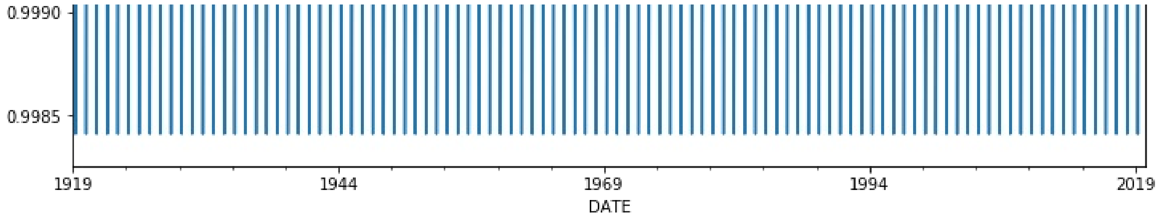
In [16]:

```
result.seasonal.plot(figsize=(12,8))
```

Out[16]:

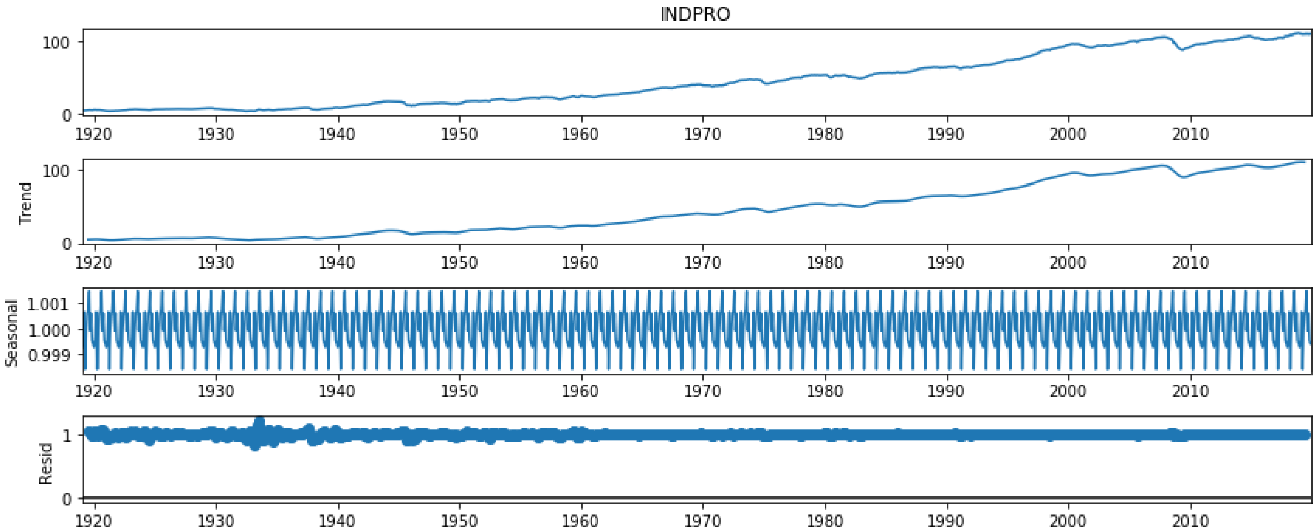
```
<matplotlib.axes._subplots.AxesSubplot at 0x12e12badb08>
```





In [17]:

```python
from pylab import rcParams
rcParams['figure.figsize']=12,5
result.plot();
```



In [18]:

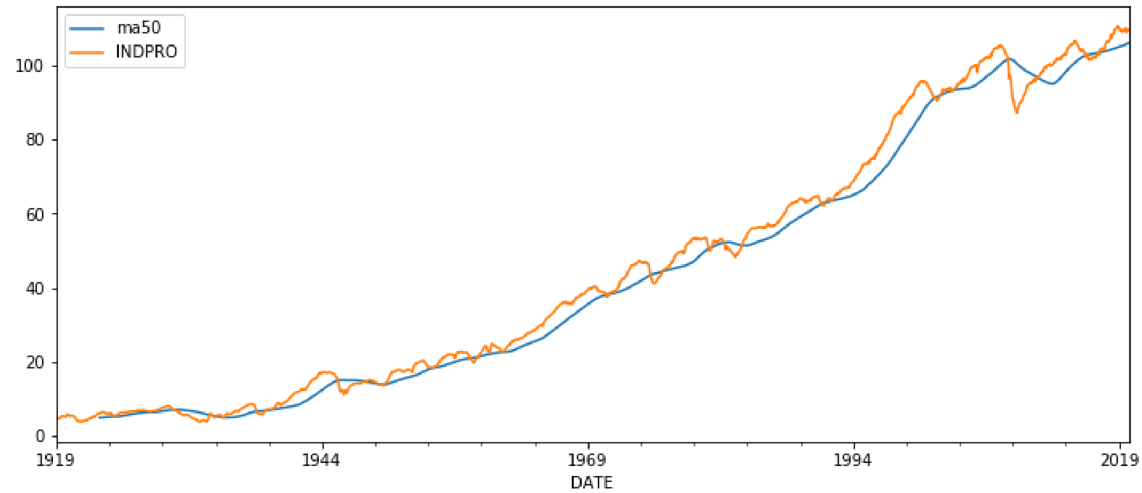
```python
#SMA
df['ma50']=df['INDPRO'].rolling(50).mean()
```

In [19]:

```python
df[['ma50','INDPRO']].plot()
```

Out[19]:

<matplotlib.axes._subplots.AxesSubplot at 0x12e12dd3388>



In [20]:

```python
#Exponential moving average or Simple Exponential Smoothing
```

```
span=24
alpha=2/(span+1)
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
df['EWA24']=df['INDPRO'].ewm(alpha=alpha,adjust=False).mean()
df['SES24']=SimpleExpSmoothing(df['INDPRO']).fit(smoothing_level=alpha,optimized=False).fittedvalues.shift(-1)
```

In [21]:

```
df
```

Out[21]:

	INDPRO	Trend	ma50	EWA24	SES24
DATE					
1919-01-01	5.0124	-0.040006	NaN	5.012400	5.012400
1919-02-01	4.7908	-0.286230	NaN	4.994672	4.994672
1919-03-01	4.6524	-0.449230	NaN	4.967290	4.967290
1919-04-01	4.7355	-0.390501	NaN	4.948747	4.948747
1919-05-01	4.7632	-0.386458	NaN	4.933903	4.933903
...	...	...	...	...	...
2019-08-01	109.9634	0.139045	105.545012	108.269967	108.269967
2019-09-01	109.4437	-0.428749	105.647400	108.363866	108.363866
2019-10-01	108.8532	-1.065339	105.741222	108.403013	108.403013
2019-11-01	109.7573	-0.206481	105.860832	108.511356	108.511356
2019-12-01	109.4330	-0.575664	105.981540	108.585087	NaN

1212 rows × 5 columns

In [22]:

```
df.columns
```

Out[22]:

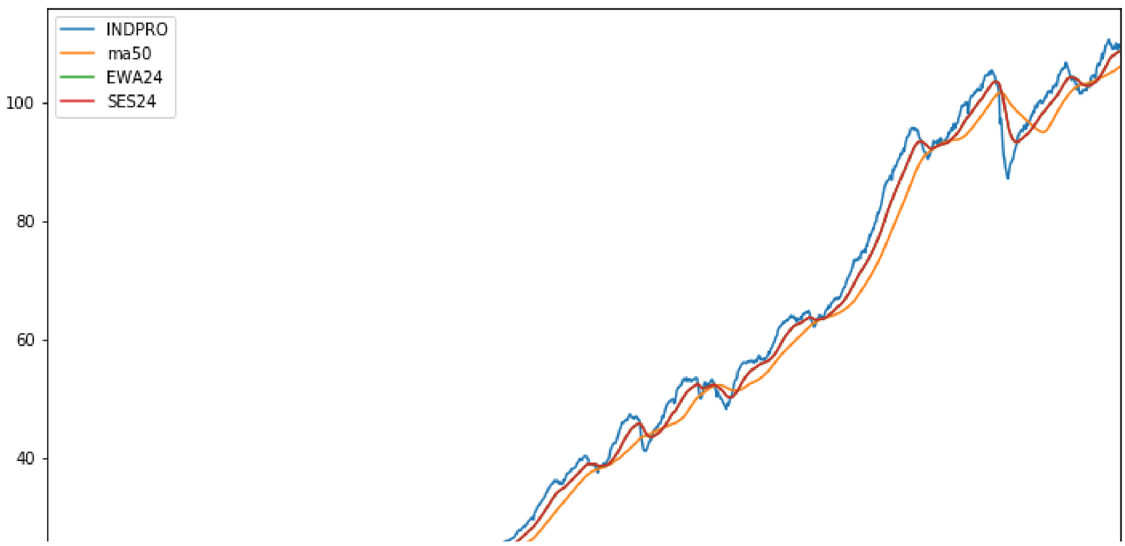
```
Index(['INDPRO', 'Trend', 'ma50', 'EWA24', 'SES24'], dtype='object')
```

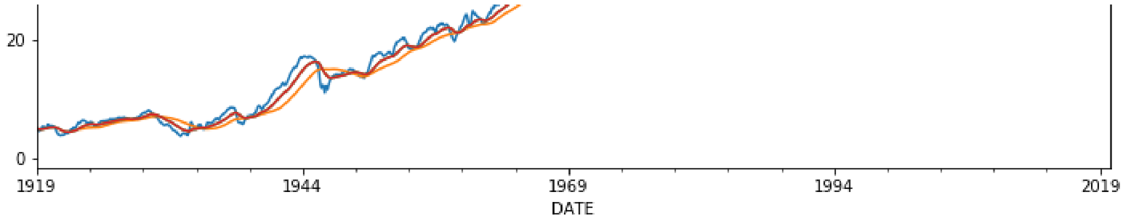
In [23]:

```
df[['INDPRO','ma50','EWA24','SES24']].plot(figsize=(12,8))
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12e12fff148>
```



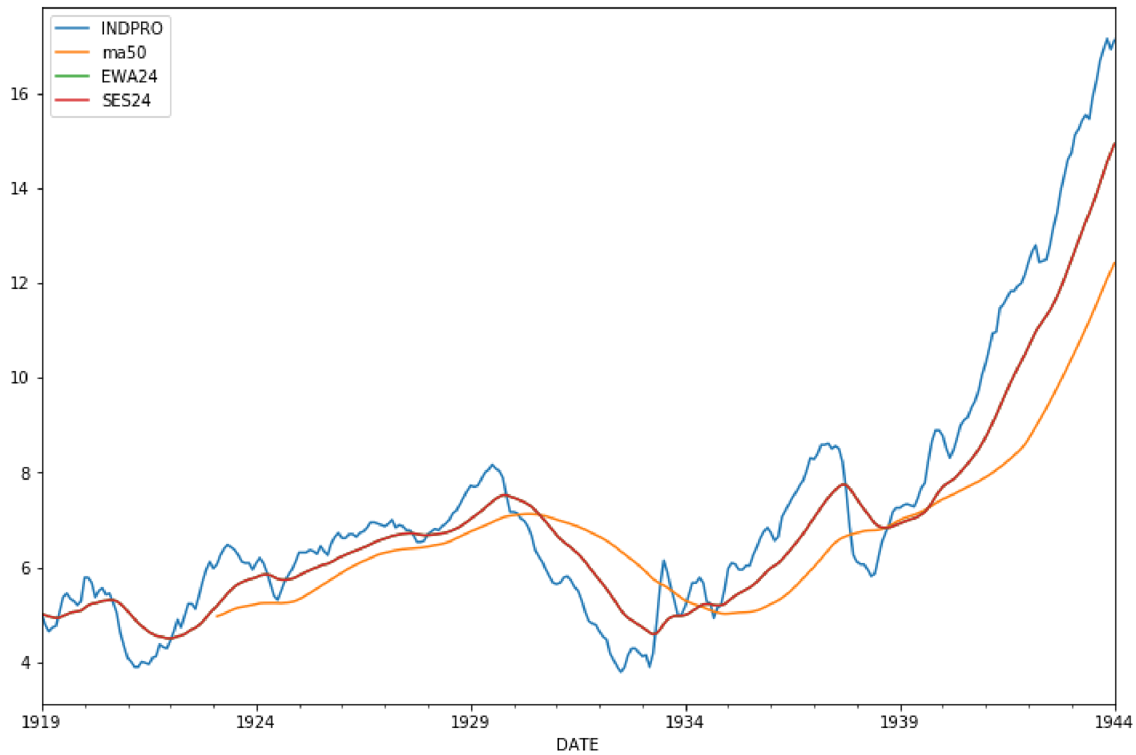


In [24]:

```
df[['INDPRO','ma50','EWA24','SES24']][:'1944-01-01'].plot(figsize=(12,8))
```

Out[24]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12e12d3d1c8>
```

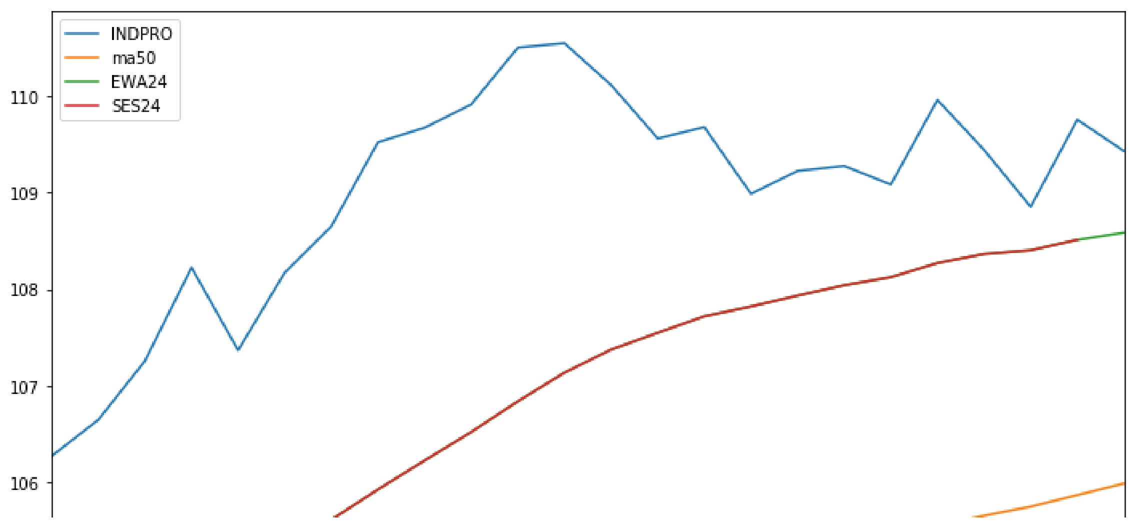


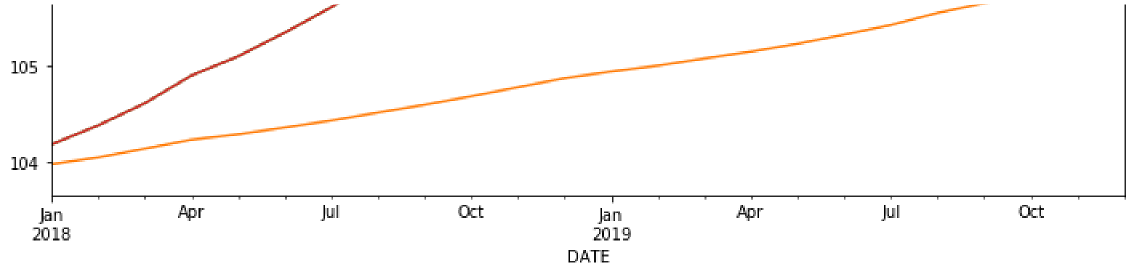
In [25]:

```
df[['INDPRO','ma50','EWA24','SES24']]['2018-01-01':].plot(figsize=(12,8)).plot(figsize=(12,8))
```

Out[25]:

```
[]
```





In [26]:

```
#Double Exponential Smoothing with alpha  and beta, that addresses the trend of the data using the
multiplicative approach
from statsmodels.tsa.holtwinters import ExponentialSmoothing
df['DES_MUL24']=ExponentialSmoothing(df['INDPRO'],trend='mul').fit().fittedvalues.shift(-1)
```

```
C:\Users\chumj\Anaconda3\Ben\lib\site-packages\statsmodels\tsa\holtwinters.py:731: RuntimeWarning:
invalid value encountered in greater_equal
  loc = initial_p >= ub
```

In [27]:

```
df
```

Out[27]:

DATE	INDPRO	Trend	ma50	EWA24	SES24	DES_MUL24
1919-01-01	5.0124	-0.040006	NaN	5.012400	5.012400	4.979524
1919-02-01	4.7908	-0.286230	NaN	4.994672	4.994672	4.716834
1919-03-01	4.6524	-0.449230	NaN	4.967290	4.967290	4.565812
1919-04-01	4.7355	-0.390501	NaN	4.948747	4.948747	4.688103
1919-05-01	4.7632	-0.386458	NaN	4.933903	4.933903	4.733341
...	...	...	...	...	...	...
2019-08-01	109.9634	0.139045	105.545012	108.269967	108.269967	110.118578
2019-09-01	109.4437	-0.428749	105.647400	108.363866	108.363866	109.439723
2019-10-01	108.8532	-1.065339	105.741222	108.403013	108.403013	108.711656
2019-11-01	109.7573	-0.206481	105.860832	108.511356	108.511356	109.863250
2019-12-01	109.4330	-0.575664	105.981540	108.585087	NaN	NaN

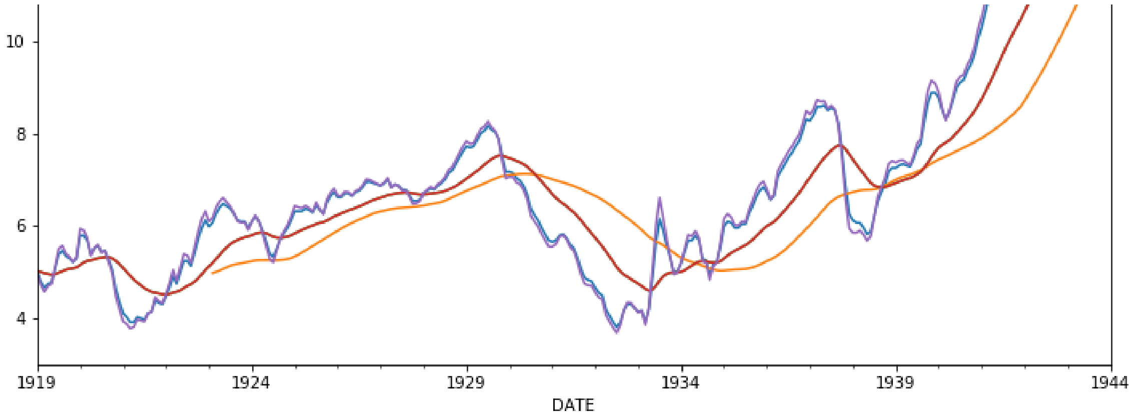
1212 rows × 6 columns

In [28]:

```
df[['INDPRO','ma50','EWA24','SES24','DES_MUL24']][:'1944-01-01'].plot(figsize=(12,8))
```

Out[28]:

<matplotlib.axes._subplots.AxesSubplot at 0x12e134b0b08>

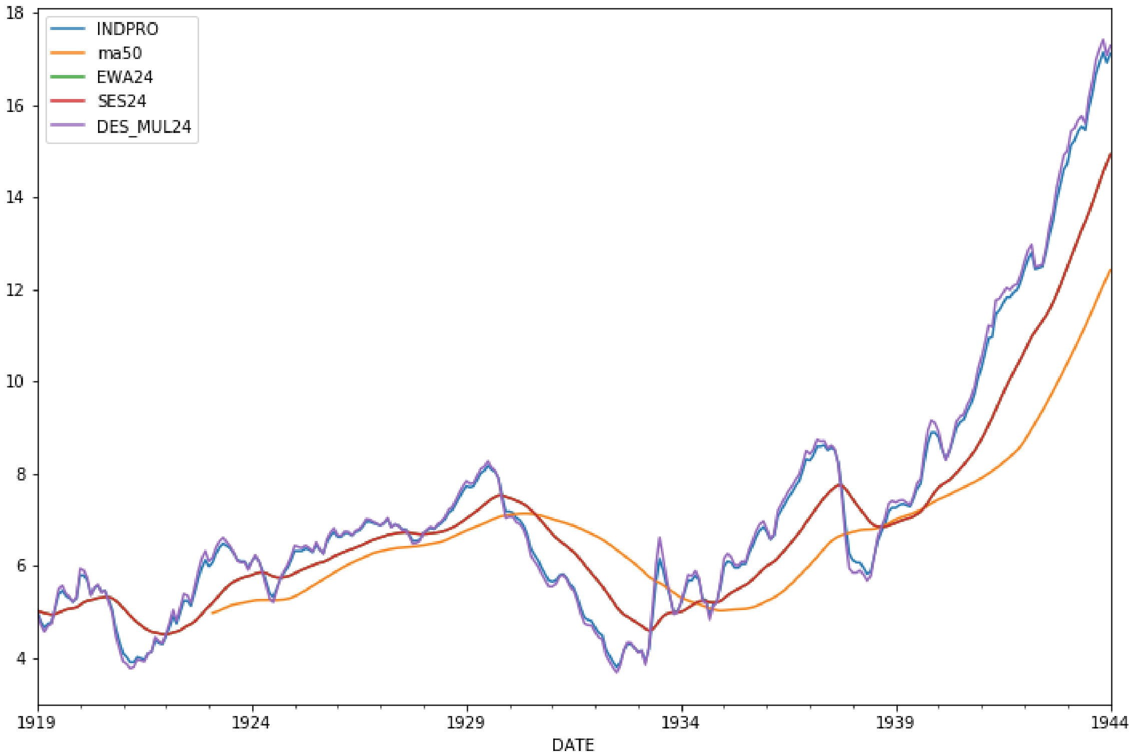


In [29]:

```
df[['INDPRO','ma50','EWA24','SES24','DES_MUL24']][:'1944-01-01'].plot(figsize=(12,8))
```

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12e135dc6c8>
```

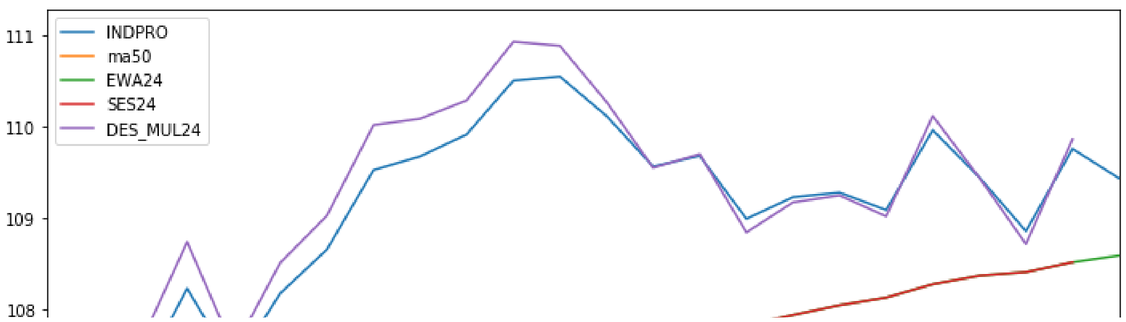


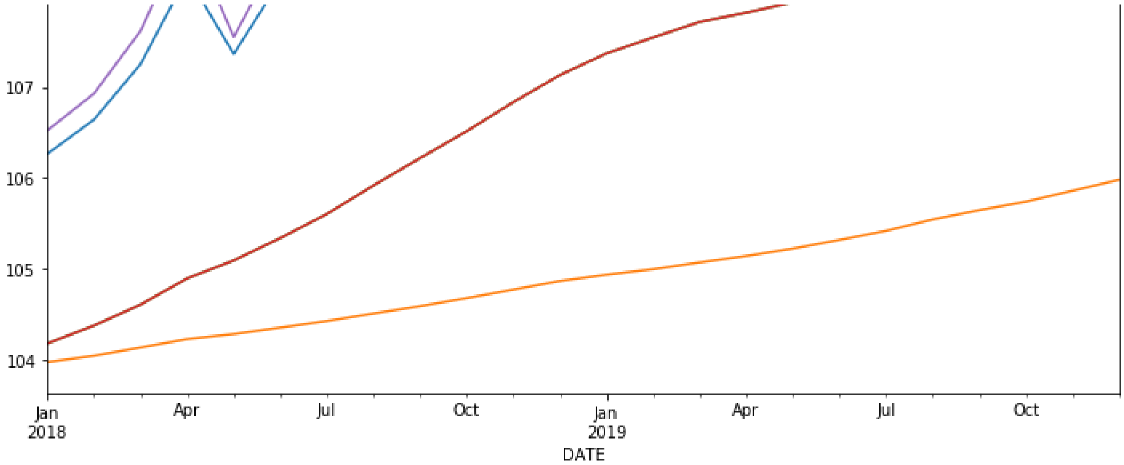
In [30]:

```
df[['INDPRO','ma50','EWA24','SES24','DES_MUL24']]['2018-01-01':].plot(figsize=(12,8))
```

Out[30]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12e135dcb88>
```





In [31]:

```
#Double Exponential Smoothing with alpha  and beta, that addresses the trend of the data using the addiction approach
from statsmodels.tsa.holtwinters import ExponentialSmoothing
df['DES_ADD24']=ExponentialSmoothing(df['INDPRO'],trend='add').fit().fittedvalues.shift(-1)
```

```
C:\Users\chumj\Anaconda3\Ben\lib\site-packages\statsmodels\tsa\holtwinters.py:731: RuntimeWarning: invalid value encountered in greater_equal
  loc = initial_p >= ub
```

In [34]:

```
df
```

Out[34]:

DATE	INDPRO	Trend	ma50	EWA24	SES24	DES_MUL24	DES_ADD24
1919-01-01	5.0124	-0.040006	NaN	5.012400	5.012400	4.979524	5.010177
1919-02-01	4.7908	-0.286230	NaN	4.994672	4.994672	4.716834	4.736807
1919-03-01	4.6524	-0.449230	NaN	4.967290	4.967290	4.565812	4.578488
1919-04-01	4.7355	-0.390501	NaN	4.948747	4.948747	4.688103	4.698641
1919-05-01	4.7632	-0.386458	NaN	4.933903	4.933903	4.733341	4.741576
...	...	...	...	...	...	...	...
2019-08-01	109.9634	0.139045	105.545012	108.269967	108.269967	110.118578	110.116475
2019-09-01	109.4437	-0.428749	105.647400	108.363866	108.363866	109.439723	109.438010
2019-10-01	108.8532	-1.065339	105.741222	108.403013	108.403013	108.711656	108.709504
2019-11-01	109.7573	-0.206481	105.860832	108.511356	108.511356	109.863250	109.860868
2019-12-01	109.4330	-0.575664	105.981540	108.585087	NaN	NaN	NaN

1212 rows × 7 columns

In [35]:

```
df.columns
```

Out[35]:

```
Index(['INDPRO', 'Trend', 'ma50', 'EWA24', 'SES24', 'DES_MUL24', 'DES_ADD24'], dtype='object')
```

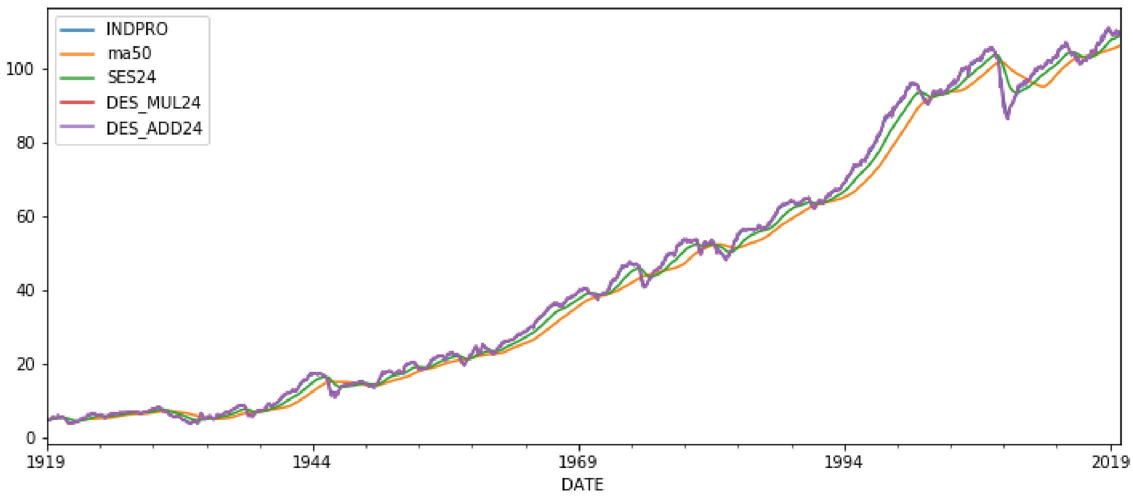
In [36]:

```
df[['INDPRO', 'ma50', 'SES24','DES_MUL24', 'DES_ADD24']].plot()
```

Out[36]:

Out[36]:

<matplotlib.axes._subplots.AxesSubplot at 0x12e13a3b308>

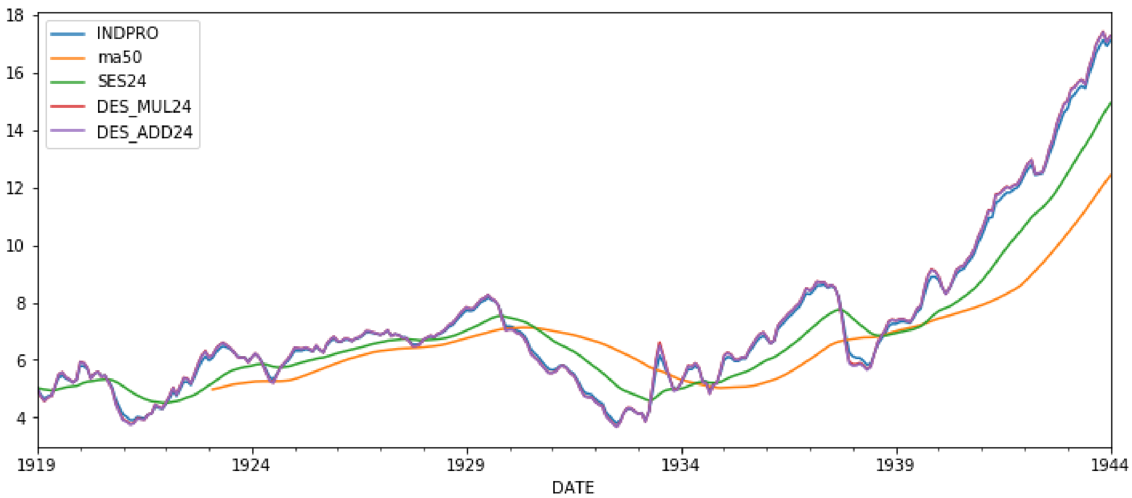


In [37]:

```
df[['INDPRO', 'ma50', 'SES24','DES_MUL24', 'DES_ADD24']][:'1944-01-01'].plot()
```

Out[37]:

<matplotlib.axes._subplots.AxesSubplot at 0x12e152c3cc8>

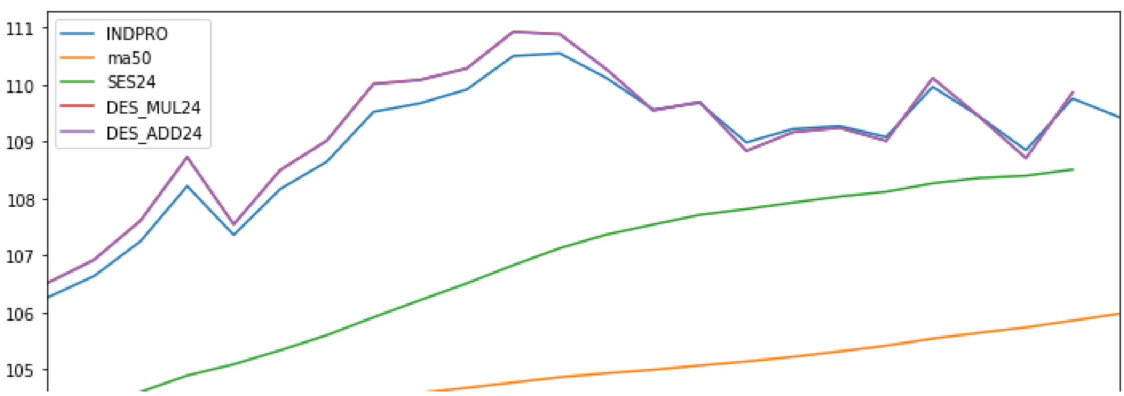


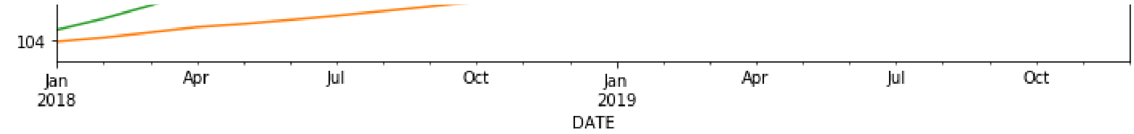
In [38]:

```
df[['INDPRO', 'ma50', 'SES24','DES_MUL24', 'DES_ADD24']]['2018-01-01':].plot()
```

Out[38]:

<matplotlib.axes._subplots.AxesSubplot at 0x12e153877c8>



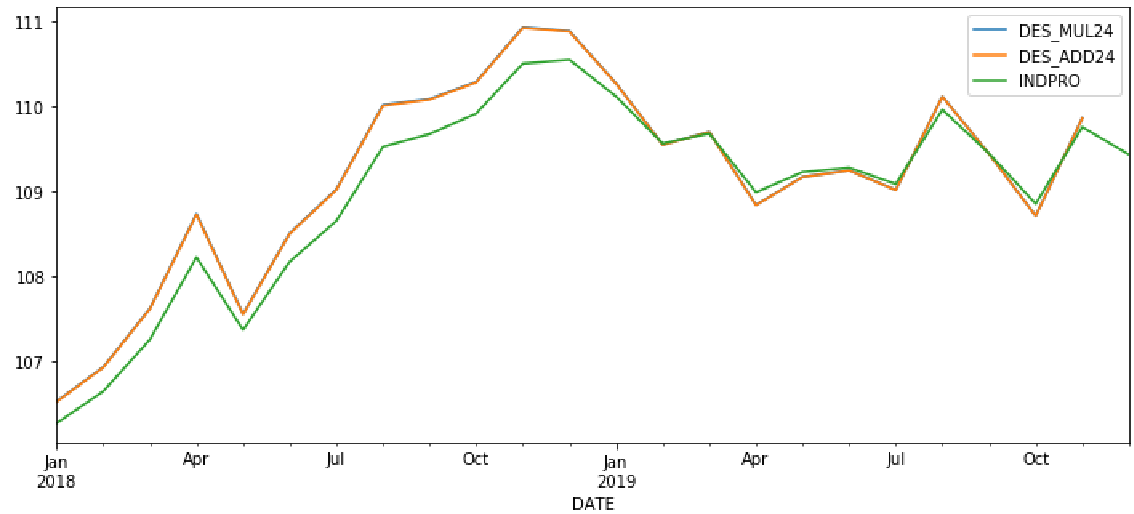


In [39]:

```
df[['DES_MUL24', 'DES_ADD24','INDPRO']]['2018-01-01':].plot()
```

Out[39]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12e1536a108>
```



In [42]:

```
#Implementing Triple Exponential Smoothing,supporting trend and seasonality that is alpha,beta and gamma.
df['TES_MUL24']=ExponentialSmoothing(df['INDPRO'],trend='mul',seasonal='mul',seasonal_periods=24).fit().fittedvalues
df['TES_ADD24']=ExponentialSmoothing(df['INDPRO'],trend='add',seasonal='add',seasonal_periods=24).fit().fittedvalues
```

```
C:\Users\chumj\Anaconda3\Ben\lib\site-packages\statsmodels\tsa\holtwinters.py:725: RuntimeWarning: invalid value encountered in less_equal
  loc = initial_p <= lb
C:\Users\chumj\Anaconda3\Ben\lib\site-packages\statsmodels\tsa\holtwinters.py:731: RuntimeWarning: invalid value encountered in greater_equal
  loc = initial_p >= ub
```

In [43]:

```
df
```

Out[43]:

	INDPRO	Trend	ma50	EWA24	SES24	DES_MUL24	DES_ADD24	TES_MUL24	TES_ADD24
DATE									
1919-01-01	5.0124	-0.040006	NaN	5.012400	5.012400	4.979524	5.010177	4.989045	5.277892
1919-02-01	4.7908	-0.286230	NaN	4.994672	4.994672	4.716834	4.736807	4.754877	4.936286
1919-03-01	4.6524	-0.449230	NaN	4.967290	4.967290	4.565812	4.578488	4.609886	4.680236
1919-04-01	4.7355	-0.390501	NaN	4.948747	4.948747	4.688103	4.698641	4.688182	4.473652
1919-05-01	4.7632	-0.386458	NaN	4.933903	4.933903	4.733341	4.741576	4.713969	4.695956
...	...	...	...	...	...	...	...	...	...
2019-08-01	109.9634	0.139045	105.545012	108.269967	108.269967	110.118578	110.116475	109.292725	109.054745
2019-09-01	109.4437	-0.428749	105.647400	108.363866	108.363866	109.439723	109.438010	109.845289	110.073773

DATE	INDPRO	Trend	ma50	EWA24	SES24	DES_MUL24	DES_ADD24	TES_MUL24	TES_ADD24
2019-10-01	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2019-11-01	109.7573	-0.206481	105.860832	108.511356	108.511356	109.863250	109.860868	110.086742	108.747264
2019-12-01	109.4330	-0.575664	105.981540	108.585087	NaN	NaN	NaN	109.990222	109.888814

1212 rows × 9 columns

In [44]:

```
df.columns
```

Out[44]:

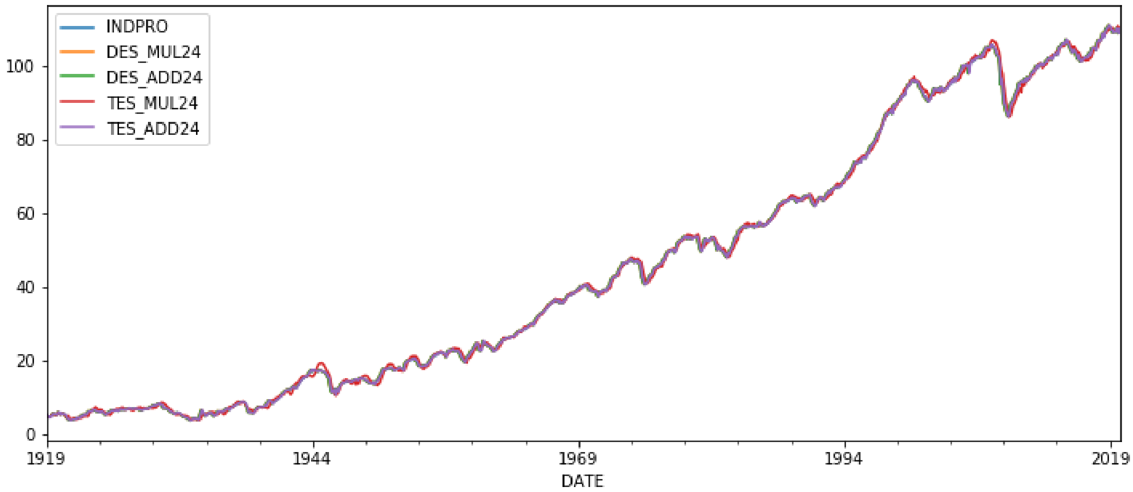
```
Index(['INDPRO', 'Trend', 'ma50', 'EWA24', 'SES24', 'DES_MUL24', 'DES_ADD24',
       'TES_MUL24', 'TES_ADD24'],
      dtype='object')
```

In [45]:

```
df[['INDPRO','DES_MUL24', 'DES_ADD24','TES_MUL24', 'TES_ADD24']].plot()
```

Out[45]:

<matplotlib.axes._subplots.AxesSubplot at 0x12e173fe788>

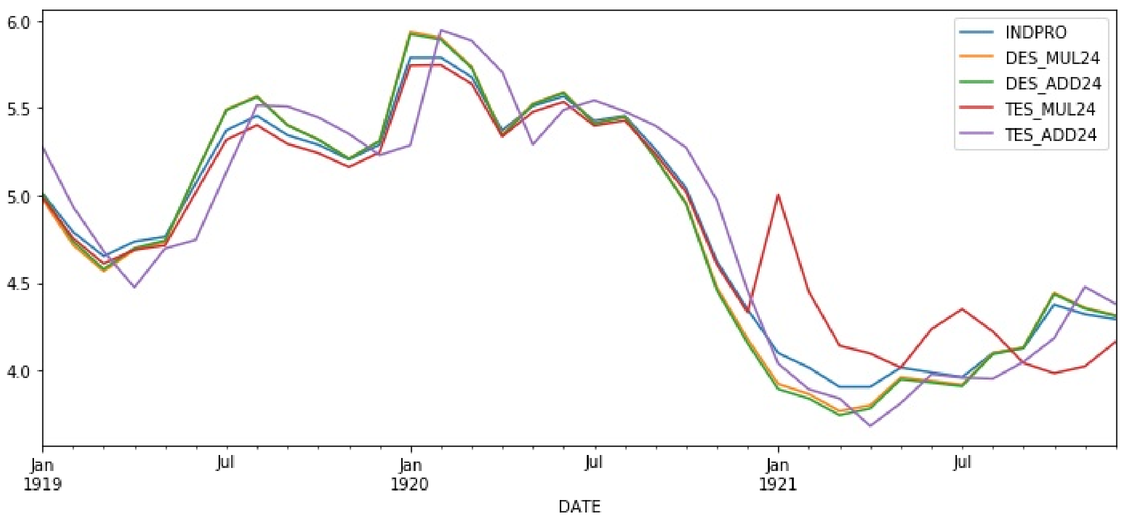


In [50]:

```
df[['INDPRO','DES_MUL24', 'DES_ADD24','TES_MUL24', 'TES_ADD24']].iloc[:36].plot()
```

Out[50]:

<matplotlib.axes._subplots.AxesSubplot at 0x12e16db5fc8>

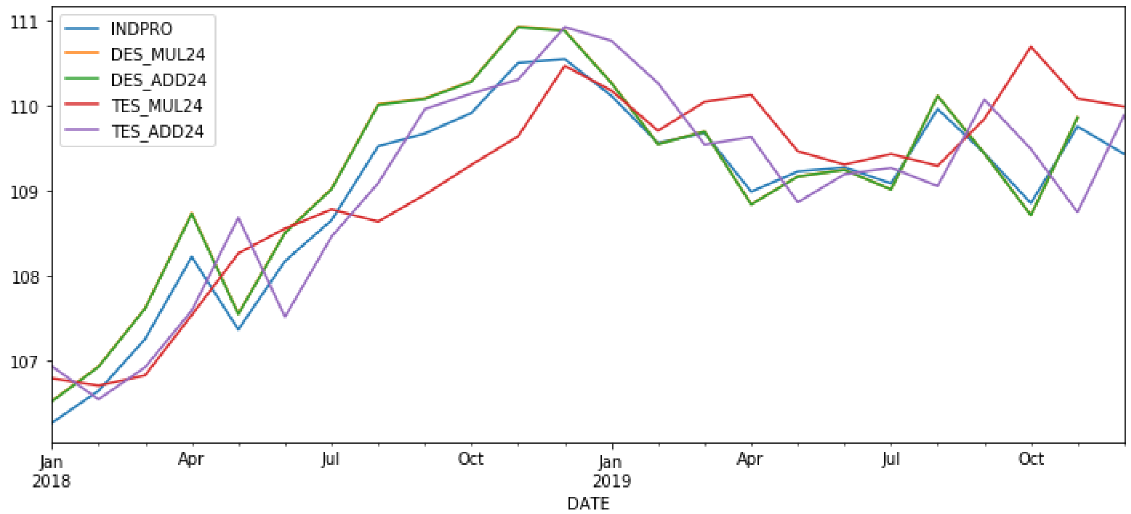


In [49]:

```
df[['INDPRO','DES_MUL24', 'DES_ADD24','TES_MUL24', 'TES_ADD24']].iloc[-24:].plot()
```

Out[49]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12e164c8848>
```

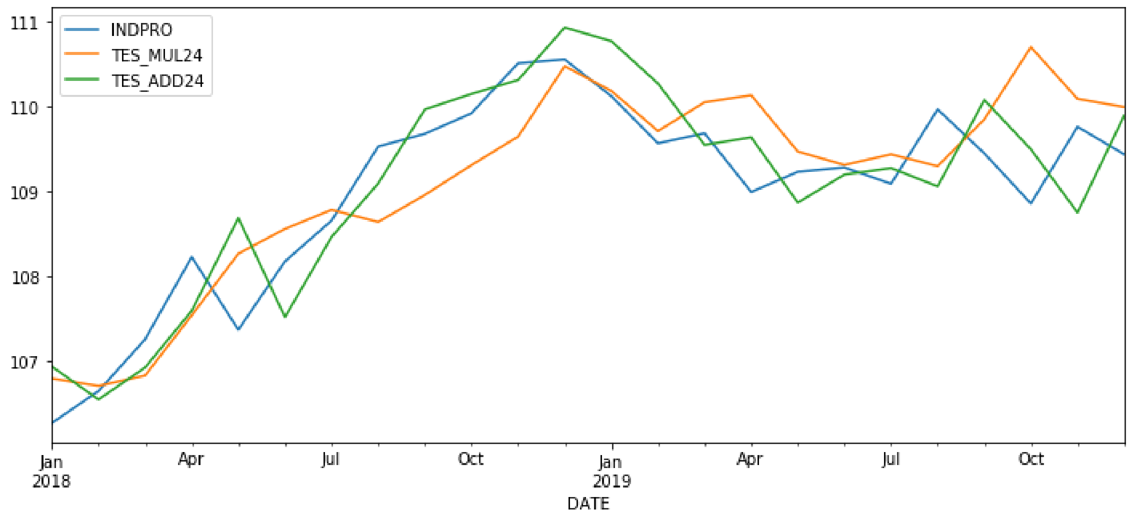


In [51]:

```
df[['INDPRO','TES_MUL24', 'TES_ADD24']].iloc[-24:].plot()
```

Out[51]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12e1723c348>
```



Base on our analysis, it seems DES does well or fits well more than TES,but the main aspect that comes in to clarify us is the forecasting evaluation,from there we can make our conclusions.

In [54]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1212 entries, 1919-01-01 to 2019-12-01
```

```
DatetimeIndex: 1212 entries, 1919-01-01 to 2019-12-01
Freq: MS
Data columns (total 9 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   INDPRO     1212 non-null   float64
 1   Trend      1212 non-null   float64
 2   ma50       1163 non-null   float64
 3   EWA24      1212 non-null   float64
 4   SES24      1211 non-null   float64
 5   DES_MUL24  1211 non-null   float64
 6   DES_ADD24  1211 non-null   float64
 7   TES_MUL24  1212 non-null   float64
 8   TES_ADD24  1212 non-null   float64
dtypes: float64(9)
memory usage: 134.7 KB
```

In [55]:

```
#Lets split our data into train and test set and make 3years forecast.That is train=1212-36 and te
st=36
#train=1176
#test=36
```

In [84]:

```
train=df.iloc[:973]
test=df.iloc[972:]
```

In [85]:

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

In [94]:

```
TES_model=ExponentialSmoothing(train['INDPRO'],trend='mul',seasonal='mul',seasonal_periods=12).fit
()
```

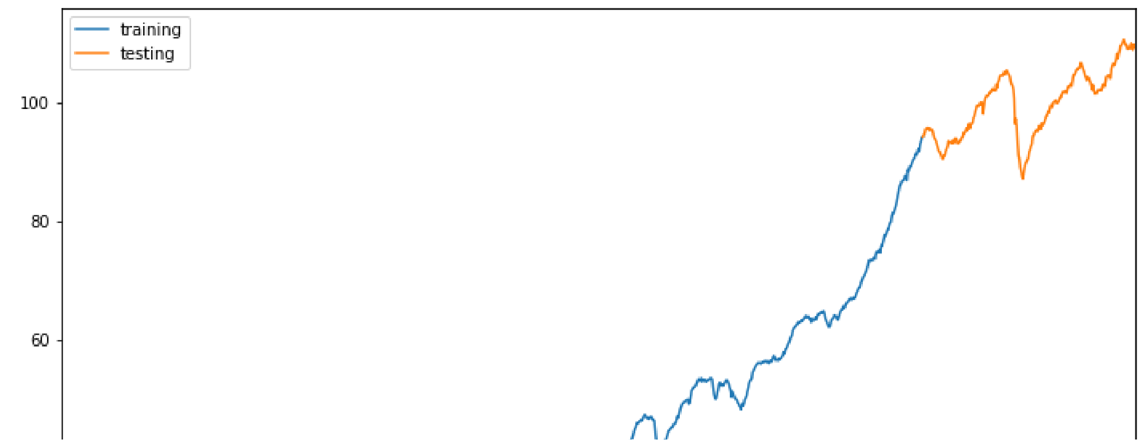
```
C:\Users\chumj\Anaconda3\Ben\lib\site-packages\statsmodels\tsa\holtwinters.py:725: RuntimeWarning:
invalid value encountered in less_equal
  loc = initial_p <= lb
C:\Users\chumj\Anaconda3\Ben\lib\site-packages\statsmodels\tsa\holtwinters.py:731: RuntimeWarning:
invalid value encountered in greater_equal
  loc = initial_p >= ub
```

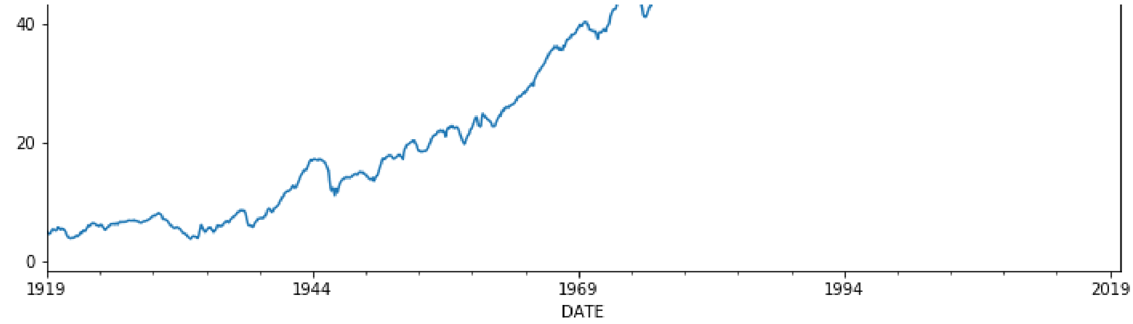
In [109]:

```
test_prediction=TES_model.forecast(240)
```

In [110]:

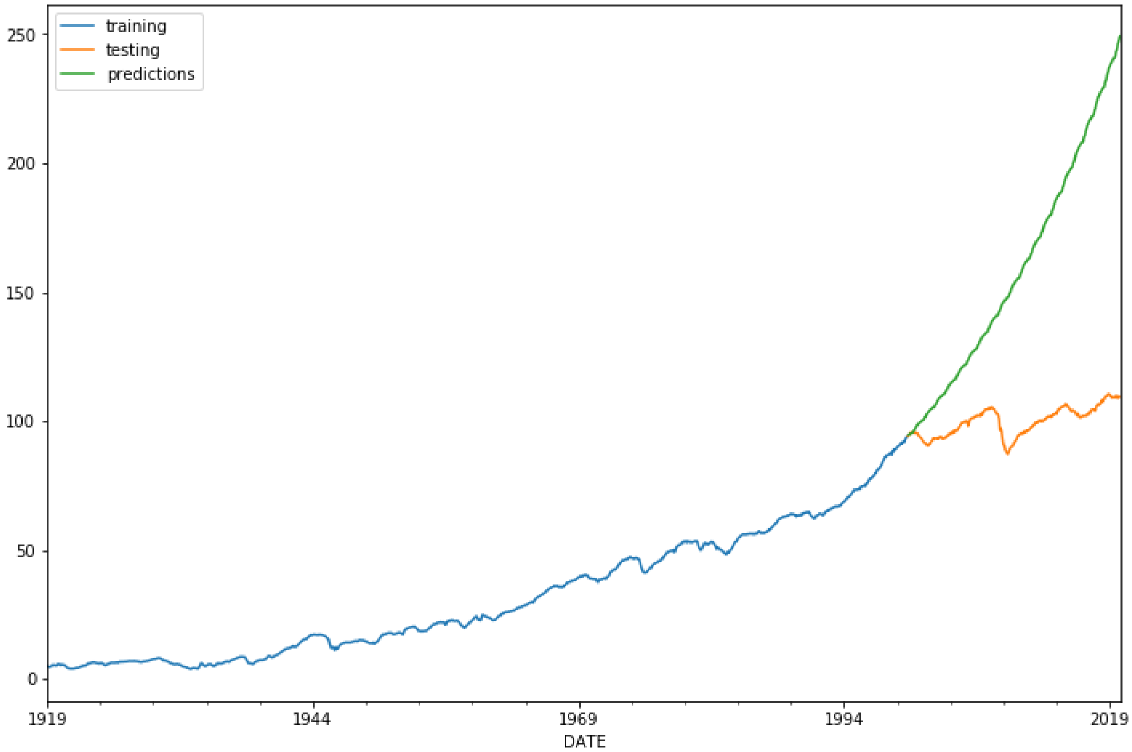
```
train['INDPRO'].plot(legend=True,label='training',figsize=(12,8))
test['INDPRO'].plot(legend=True,label='testing');
```





In [111]:

```
train['INDPRO'].plot(legend=True,label='training',figsize=(12,8))
test['INDPRO'].plot(legend=True,label='testing')
test_prediction.plot(legend=True,label='predictions');
```



In [101]:

```
model=ExponentialSmoothing(df['INDPRO'],trend='mul',seasonal='mul',seasonal_periods=12).fit()
```

```
C:\Users\chumj\Anaconda3\Ben\lib\site-packages\statsmodels\tsa\holtwinters.py:725: RuntimeWarning:
invalid value encountered in less_equal
  loc = initial_p <= lb
C:\Users\chumj\Anaconda3\Ben\lib\site-packages\statsmodels\tsa\holtwinters.py:731: RuntimeWarning:
invalid value encountered in greater_equal
  loc = initial_p >= ub
```

In [102]:

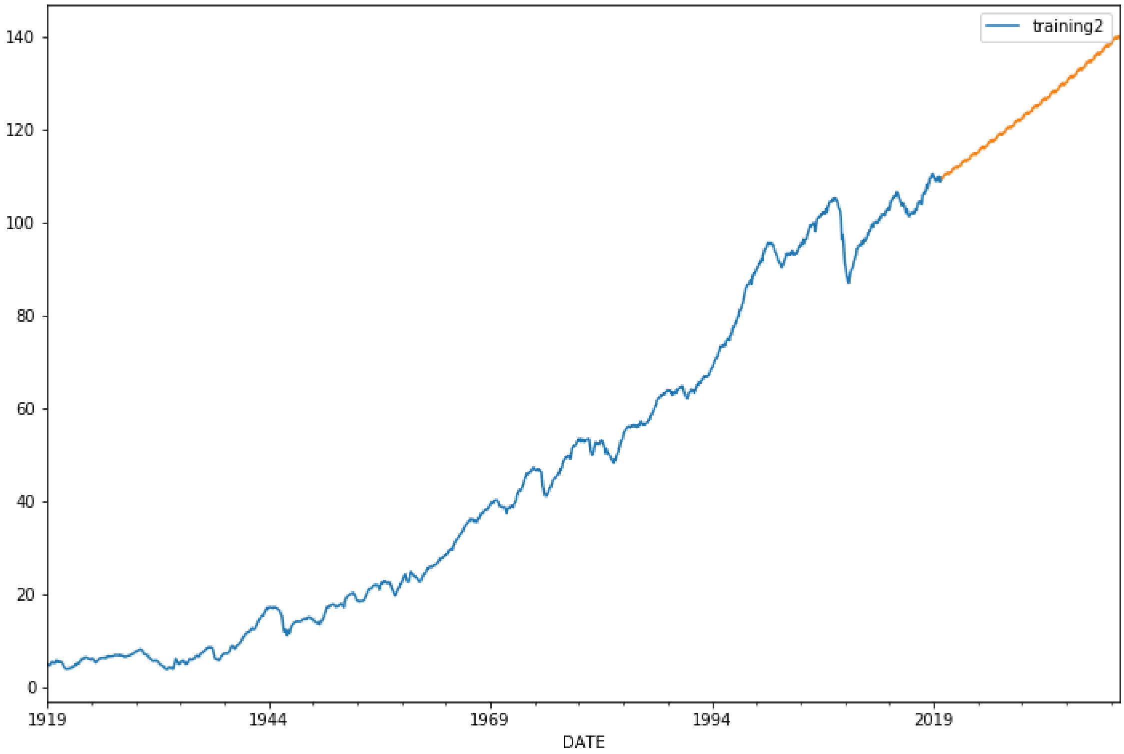
```
ft=model.forecast(240)
```

In [104]:

```
df['INDPRO'].plot(legend=True,label='training2',figsize=(12,8))
ft.plot()
```

Out[104]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12e1aee4088>
```



In [105]:

```
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score,
```

In [125]:

```
mean_absolute_error(test['INDPRO'],test_prediction)
```

Out[125]:

```
59.626285583773026
```

In [126]:

```
r2_score(test['INDPRO'],test_prediction)
```

Out[126]:

```
-162.82233127970912
```

In [127]:

```
mean_squared_error(test['INDPRO'],test_prediction)
```

Out[127]:

```
5186.955060712123
```

In [128]:

```
np.sqrt(mean_squared_error(test['INDPRO'],test_prediction))
```

Out[128]:

```
72.02051833132086
```

In [129]:

```
test['INDPRO'].describe()
```

Out[129]:

```
count    240.000000
mean      99.665681
std        5.638666
min       87.074200
25%       95.110300
50%      100.328500
75%      104.056200
max      110.551600
Name: INDPRO, dtype: float64
```

In [130]:

```
test_prediction.describe()
```

Out[130]:

```
count    240.000000
mean     159.290858
std       44.482849
min       94.693537
25%      120.612138
50%      153.624691
75%      195.672755
max      249.229257
dtype: float64
```

In [ ]:

```

```