```
import pandas as pd
import numpy as np


ts = pd.Series([17, 21, 19, 23, 18, 16, 20, 18, 22, 20, 15, 22])
l = pd.Series([np.nan]*len(ts))
lookback = 3
pred_ma = l.copy()
for i in range(lookback, len(ts)):
    pred_ma[i] = np.mean(ts[i - lookback:i])

pred_ma
```

```
     0      NaN
     1      NaN
     2      NaN
     3     19.0
     4     21.0
     5     20.0
     6     19.0
     7     18.0
     8     18.0
     9     20.0
     10    20.0
     11    19.0
     dtype: float64
```

```
lookback = 3
pred_map = l.copy()
for i in range(lookback, len(ts)):
    pred_map[i] = np.dot(ts[i - lookback:i], [1/6, 2/6, 3/6])

pred_map
```

```
     0           NaN
     1           NaN
     2           NaN
     3     19.333333
     4     21.333333
     5     19.833333
     6     17.833333
     7     18.333333
     8     18.333333
     9     20.333333
     10    20.333333
     11    17.833333
     dtype: float64
```

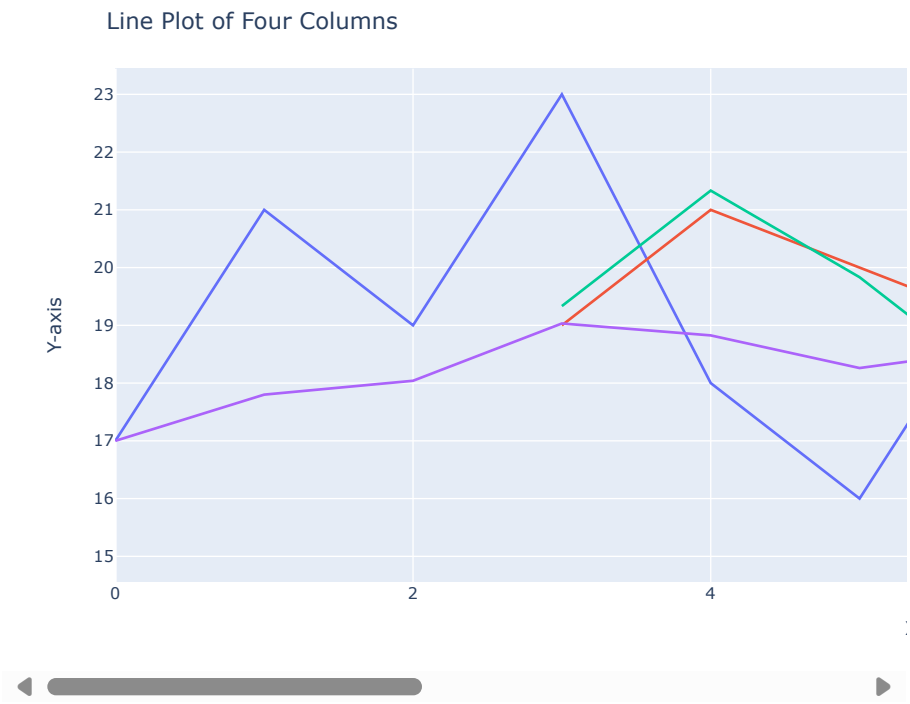
```
sm=l.copy()
alpha = 0.2
sm[0] = ts[0]
for i in range(1, len(ts)):
    sm[i] = alpha * ts[i] + (1-alpha)*sm[i-1]
sm
```

```
     0     17.000000
     1     17.800000
     2     18.040000
     3     19.032000
     4     18.825600
     5     18.260480
     6     18.608384
     7     18.486707
     8     19.189366
     9     19.351493
     10    18.481194
     11    19.184955
     dtype: float64
```

```
data = pd.DataFrame({'Real':ts, 'Moving Avg':pred_ma, 'Weighted MA':pred_map, 'Exp Smooth':sm})
data
```

	Real	Moving Avg	Weighted MA	Exp Smooth
0	17	NaN	NaN	17.000000
1	21	NaN	NaN	17.800000
2	19	NaN	NaN	18.040000
3	23	19.0	19.333333	19.032000
4	18	21.0	21.333333	18.825600
5	16	20.0	19.833333	18.260480
6	20	19.0	17.833333	18.608384
7	18	18.0	18.333333	18.486707
8	22	18.0	18.333333	19.189366

```
import plotly.express as px

fig = px.line(data, x=data.index, y=data.columns, labels={'index': 'X-axis'})

# Customize the layout if needed
fig.update_layout(
    title='Line Plot of Four Columns',
    xaxis_title='X-axis',
    yaxis_title='Y-axis',
)

# Show the plot
fig.show()
```



Problema 2

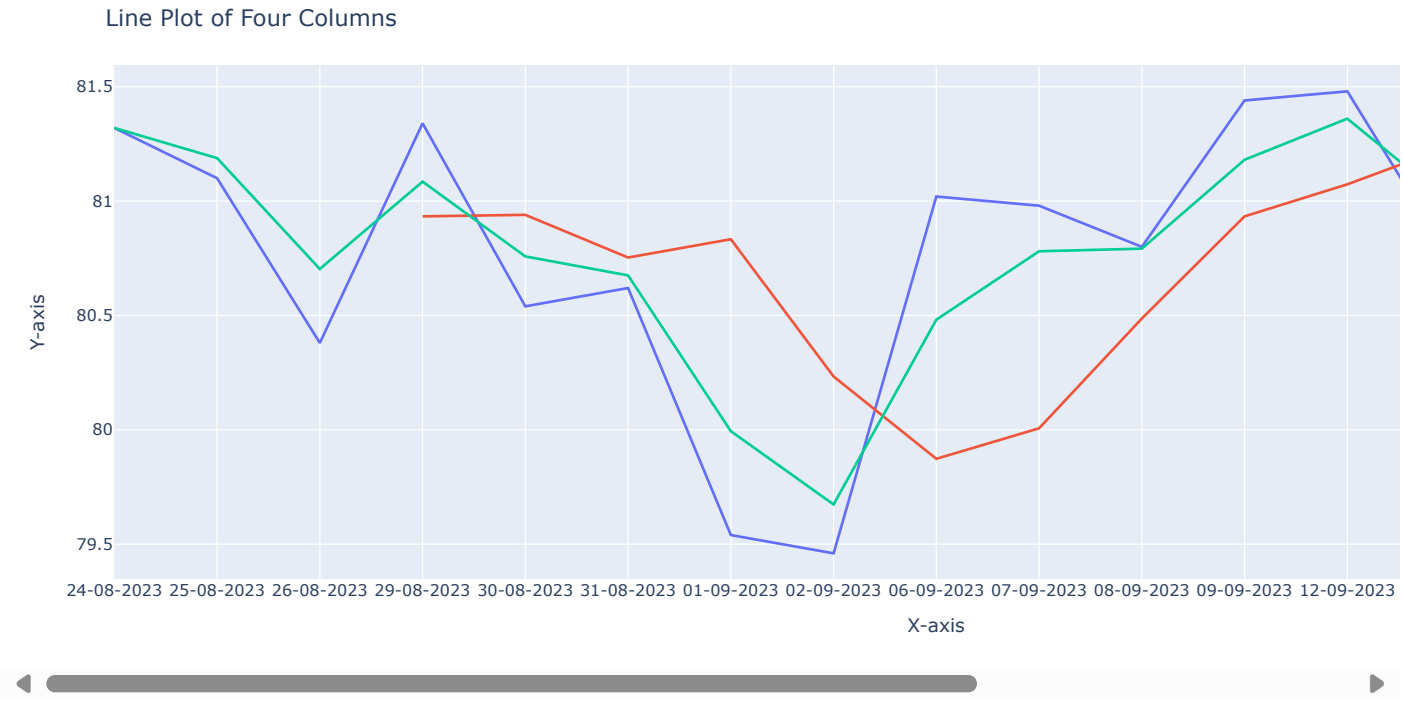
Generando la información

```
index = ['24-08-2023',#
         '25-08-2023',#
         '26-08-2023',#
         '29-08-2023',#
         '30-08-2023',#
         '31-08-2023',#
         '01-09-2023',#
         '02-09-2023',#
         '06-09-2023',#
         '07-09-2023',#
         '08-09-2023',#
```

```
         '09-09-2023',#
         '12-09-2023',#
         '13-09-2023',#
         '14-09-2023',#
         '15-09-2023',#
         '16-09-2023']#
vals = [81.32, 81.10, 80.38, 81.34, 80.54, 80.62, 79.54, 79.46, 81.02,
        80.98, 80.80, 81.44, 81.48, 80.75, 80.48, 80.01, 80.33]

df = pd.DataFrame(index=index, data={'Value':vals})
```

Creando las series de tiempo y realizando la suavización

```
lookback = 3
ts = df['Value']
pred_ma = ts.copy()
for i in range(lookback, len(ts)):
    pred_ma[i] = np.mean(ts[i - lookback:i])

sm=ts.copy()
alpha = 0.6
sm[0] = ts[0]
for i in range(1, len(ts)):
    sm[i] = alpha * ts[i] + (1-alpha)*sm[i-1]

data = pd.DataFrame(index = ts.index, data={'Real':ts, 'Moving Avg':[np.nan, np.nan, np.nan] + list(pred_ma)[3:], 'Exp Smooth':list(sm)})

fig = px.line(data, x=data.index, y=data.columns, labels={'index': 'X-axis'})

# Customize the layout if needed
fig.update_layout(
    title='Line Plot of Four Columns',
    xaxis_title='X-axis',
    yaxis_title='Y-axis',
)

# Show the plot
fig.show()
```



Generando los residuos de cada serie y obteniendo la suma de todos los residuos.

```
residuals = data.apply(lambda col: col - data['Real'])
residuals.sum()
```

```
Real          0.000000
Moving Avg    1.056667
Exp Smooth    0.673263
dtype: float64
```

Tomaria el suavizamiento exponencial porque a pesar de tener 3 registros menos la suma del error es todavía menor.