In [50]:
```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

In [51]:
```
exchange_rates = pd.read_csv('euro-daily-hist_1999_2022.csv')
exchange_rates.shape
```

Out[51]: (6456, 41)

In [52]:
```
exchange_rates.head()
```

Out[52]:

	Period\Unit:	[Australian dollar]	[Bulgarian lev]	[Brazilian real]	[Canadian dollar]	[Swiss franc]	[Chinese yuan renminbi]	[Cypriot pound]	[Czech koruna]	[Danish krone]	...	[Romanian leu]	[R r
0	2023-12-15	1.6324	1.9558	5.4085	1.4653	0.9488	7.7812	NaN	24.477	7.4556	...	4.9710	
1	2023-12-14	1.6288	1.9558	5.3349	1.4677	0.949	7.7866	NaN	24.408	7.4566	...	4.9712	
2	2023-12-13	1.6452	1.9558	5.3609	1.4644	0.9452	7.7426	NaN	24.476	7.4566	...	4.9738	
3	2023-12-12	1.6398	1.9558	5.3327	1.4656	0.9443	7.7447	NaN	24.42	7.4569	...	4.9732	
4	2023-12-11	1.642	1.9558	5.3169	1.4609	0.9478	7.7206	NaN	24.367	7.4563	...	4.9707	

5 rows × 41 columns

In [53]:
```
1 exchange_rates.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6456 entries, 0 to 6455
Data columns (total 41 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Period\Unit:              6456 non-null   object
 1   [Australian dollar ]      6456 non-null   object
 2   [Bulgarian lev ]          6054 non-null   object
 3   [Brazilian real ]         6188 non-null   object
 4   [Canadian dollar ]        6456 non-null   object
 5   [Swiss franc ]            6456 non-null   object
 6   [Chinese yuan renminbi ]  6188 non-null   object
 7   [Cypriot pound ]          2346 non-null   object
 8   [Czech koruna ]           6456 non-null   object
 9   [Danish krone ]           6456 non-null   object
 10  [Estonian kroon ]         3130 non-null   object
 11  [UK pound sterling ]      6456 non-null   object
 12  [Greek drachma ]          520 non-null    object
 13  [Hong Kong dollar ]       6456 non-null   object
 14  [Croatian kuna ]          5941 non-null   object
 15  [Hungarian forint ]       6456 non-null   object
 16  [Indonesian rupiah ]      6456 non-null   object
 17  [Israeli shekel ]         6188 non-null   object
 18  [Indian rupee ]           6188 non-null   object
 19  [Iceland krona ]          4049 non-null   float64
 20  [Japanese yen ]           6456 non-null   object
 21  [Korean won ]             6456 non-null   object
 22  [Lithuanian litas ]       4159 non-null   object
 23  [Latvian lats ]           3904 non-null   object
 24  [Maltese lira ]           2346 non-null   object
 25  [Mexican peso ]           6456 non-null   object
 26  [Malaysian ringgit ]      6456 non-null   object
 27  [Norwegian krone ]        6456 non-null   object
 28  [New Zealand dollar ]     6456 non-null   object
 29  [Philippine peso ]        6456 non-null   object
 30  [Polish zloty ]           6456 non-null   object
 31  [Romanian leu ]           6394 non-null   float64
 32  [Russian rouble ]         5994 non-null   object
 33  [Swedish krona ]          6456 non-null   object
 34  [Singapore dollar ]       6456 non-null   object
 35  [Slovenian tolar ]        2085 non-null   object
 36  [Slovak koruna ]          2608 non-null   object
 37  [Thai baht ]              6456 non-null   object
 38  [Turkish lira ]           6394 non-null   float64
 39  [US dollar ]              6456 non-null   object
 40  [South African rand ]     6456 non-null   object
dtypes: float64(3), object(38)
memory usage: 2.0+ MB
```

In [54]:
```
1 exchange_rates.rename(columns={'[US dollar ]':'US_dollar','Period\\Unit:':'Time'}, inplace=True)
```

In [55]:
```
1 exchange_rates['Time'] = pd.to_datetime(exchange_rates['Time'])
```

In [56]:
```
1 exchange_rates.sort_values('Time', inplace=True)
```

In [57]:
```
euro_to_dollar = exchange_rates[['Time','US_dollar']].copy()
euro_to_dollar['US_dollar'].value_counts()
```

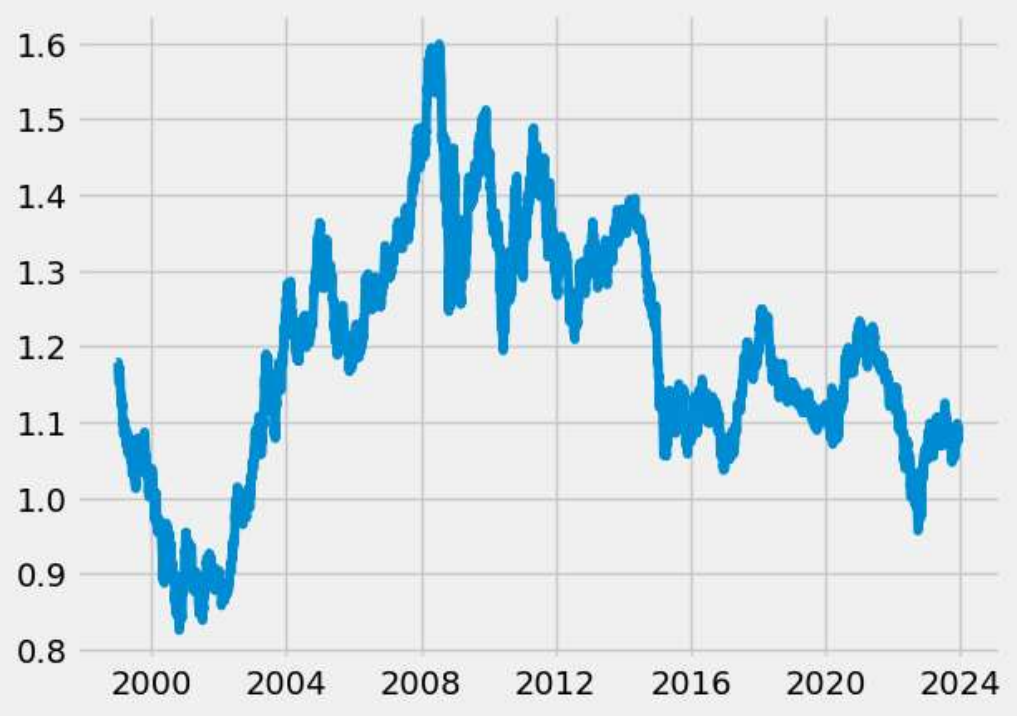
Out[57]:
```
-         62
1.2276     9
1.1215     8
1.0888     7
1.0868     7
          ..
1.4304     1
1.4350     1
1.4442     1
1.4389     1
1.0804     1
Name: US_dollar, Length: 3769, dtype: int64
```

In [58]:
```
euro_to_dollar = euro_to_dollar[euro_to_dollar['US_dollar'] != '-']
euro_to_dollar['US_dollar'] = euro_to_dollar['US_dollar'].astype(float)
euro_to_dollar['US_dollar'].info()
```

```
<class 'pandas.core.series.Series'>
Int64Index: 6394 entries, 6455 to 0
Series name: US_dollar
Non-Null Count  Dtype
--------------  -----
6394 non-null   float64
dtypes: float64(1)
memory usage: 99.9 KB
```

In [59]:
```
import matplotlib.pyplot as plt
```

In [60]:
```
plt.plot(euro_to_dollar['Time'], euro_to_dollar['US_dollar'])
plt.show()
```



In [61]:
```
values = pd.DataFrame()
values['daily_values'] = pd.Series(range(1,20,2))
values
```

Out[61]:

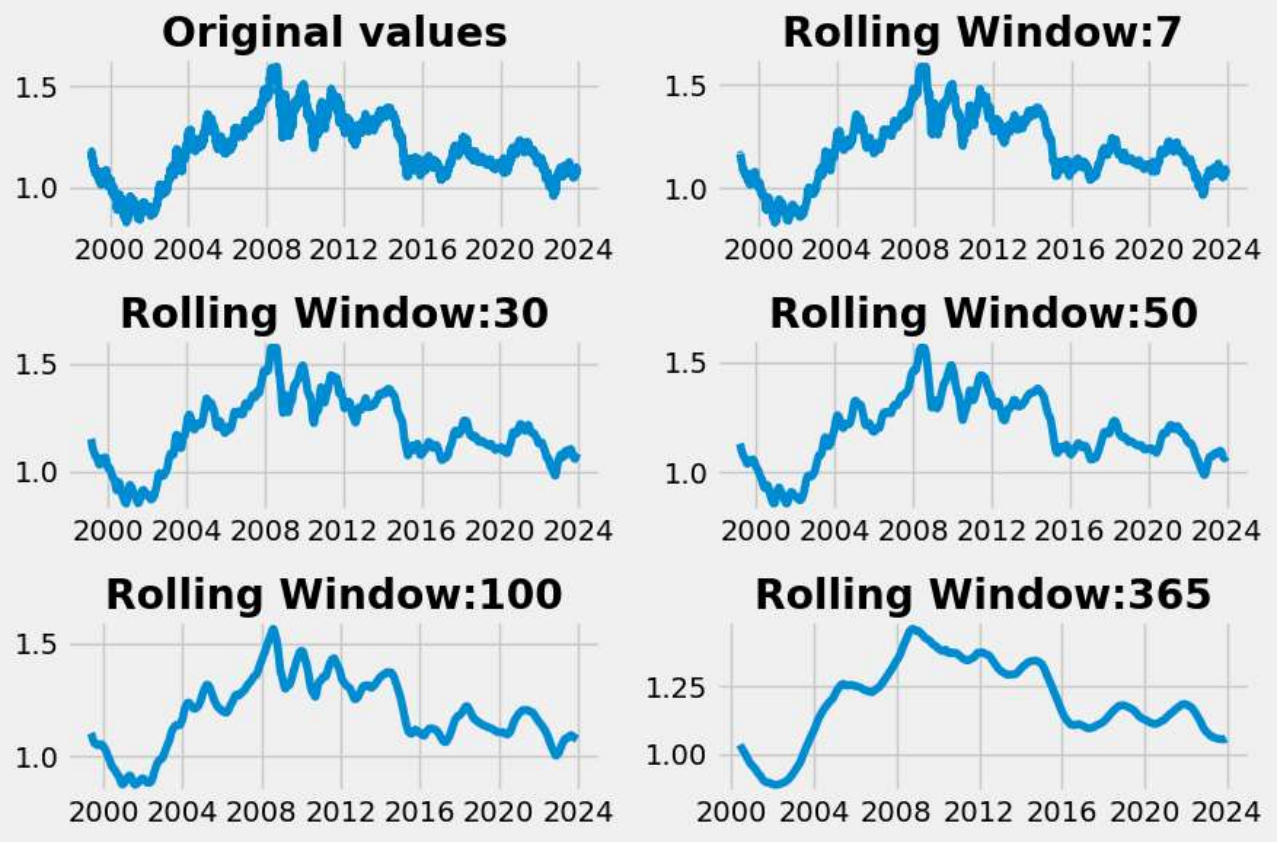
	daily_values
0	1
1	3
2	5
3	7
4	9
5	11
6	13
7	15
8	17
9	19

In [62]:
```
values['Rolling_mean_2'] = values['daily_values'].rolling(2).mean()
```

In [63]:
```
values['Rolling_mean_3'] = values['daily_values'].rolling(3).mean()
values['Rolling_mean_5'] = values['daily_values'].rolling(5).mean()
```

In [64]:

```
plt.figure(figsize=(9,6))

plt.subplot(3,2,1)
plt.plot(euro_to_dollar['Time'], euro_to_dollar['US_dollar'])
plt.title('Original values', weight = 'bold')

for i, rolling_mean in zip([2, 3, 4, 5, 6],
                           [7, 30, 50, 100, 365]):
    plt.subplot(3,2,i)
    plt.plot(euro_to_dollar['Time'],
             euro_to_dollar['US_dollar'].rolling(rolling_mean).mean())
    plt.title('Rolling Window:' + str(rolling_mean), weight='bold')

plt.tight_layout()
plt.show()
```



In [65]:
```
euro_to_dollar['Rolling_mean'] = euro_to_dollar['US_dollar'].rolling(30).mean()
euro_to_dollar
```

Out[65]:

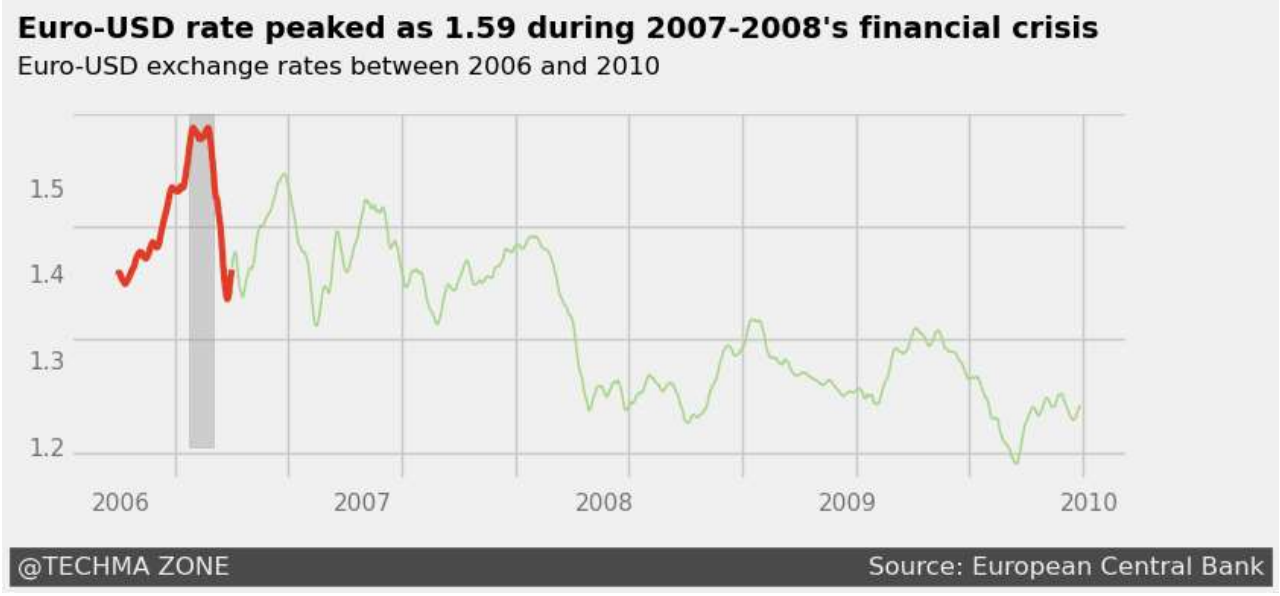
	Time	US_dollar	Rolling_mean
6455	1999-01-04	1.1789	NaN
6454	1999-01-05	1.1790	NaN
6453	1999-01-06	1.1743	NaN
6452	1999-01-07	1.1632	NaN
6451	1999-01-08	1.1659	NaN
...	...	...	...
4	2023-12-11	1.0757	1.080143
3	2023-12-12	1.0804	1.080760
2	2023-12-13	1.0787	1.081593
1	2023-12-14	1.0919	1.082453
0	2023-12-15	1.0946	1.083267

6394 rows × 3 columns

In [66]:
```
financial_crisis = euro_to_dollar.copy(
                   )[(euro_to_dollar['Time'].dt.year >= 2006
                   ) & (euro_to_dollar['Time'].dt.year >= 2009)]
financial_crisis_7_8 = euro_to_dollar.copy(
                   )[(euro_to_dollar.Time.dt.year >= 2007
                   ) & (euro_to_dollar.Time.dt.year <= 2008)]
```

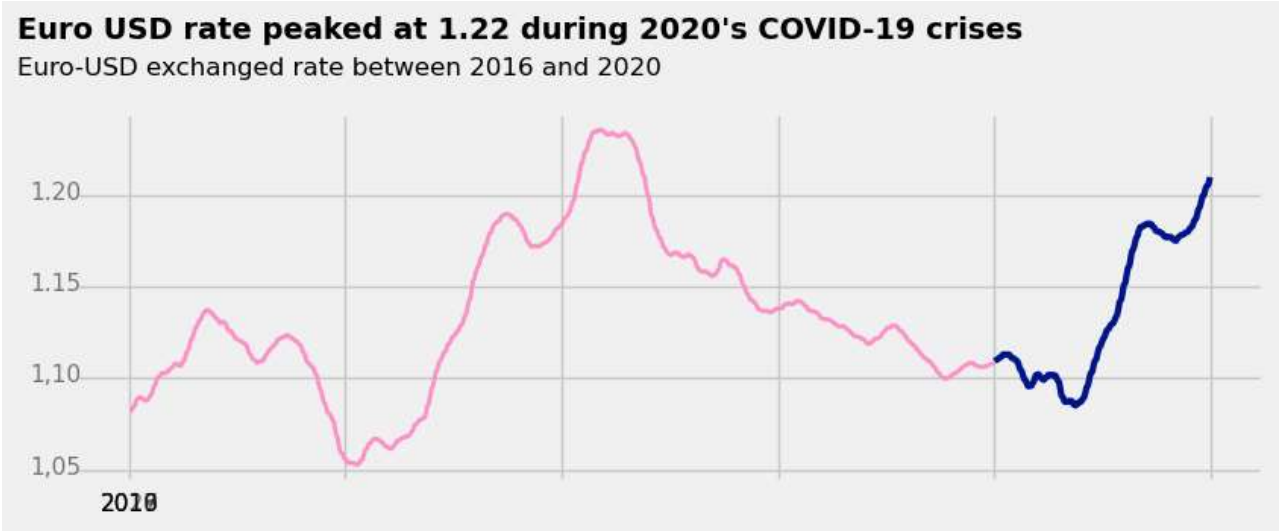
In [69]:

```
import matplotlib.style as style
style.use('fivethirtyeight')

fig,ax = plt.subplots(figsize=(8,3))
ax.plot(financial_crisis['Time'],
        financial_crisis['Rolling_mean'],
        linewidth=1, color='#A6d785')

ax.plot(financial_crisis_7_8['Time'],
        financial_crisis_7_8['Rolling_mean'],
        linewidth=3, color='#e23d28')

ax.set_xticklabels([])

x = 0.02
for year in ['2006', '2007', '2008', '2009', '2010']:
    ax.text(x, -0.08, year, alpha=0.5, fontsize=11, transform = plt.gca(). transAxes)
    x += 0.22888

ax.set_yticklabels([])
y = 0.07
for rate in ['1.2', '1.3', '1.4', '1.5']:
    ax.text(-.04, y, rate, alpha=0.5, fontsize=11, transform = plt.gca().transAxes)
    y += 0.2333

ax.text(-.05, 1.2, "Euro-USD rate peaked as 1.59 during 2007-2008's financial crisis",
        weight='bold', transform = plt.gca().transAxes)
ax.text(-.05, 1.1, 'Euro-USD exchange rates between 2006 and 2010',
        size=12, transform = plt.gca().transAxes)

ax.text(-.05, -0.25, '@TECHMA ZONE' + ' '*80 + 'Source: European Central Bank',
        color = '#f0f0f0', backgroundcolor = '#4d4d4d',
        size=12, transform = plt.gca().transAxes)

ax.axvspan(xmin=pd.to_datetime("2008-04-1"), xmax=pd.to_datetime("2008-09-1"), ymin=0.09,
           alpha=0.3, color='grey')

plt.show()
```



In [70]:
```
corona_crisis_20 = euro_to_dollar.loc[(euro_to_dollar['Time']>='2020-01-1')&
                                      (euro_to_dollar['Time']<='2020-12-31')]
corona_crisis = euro_to_dollar.loc[(euro_to_dollar['Time']>='2016-01-01')&
                                   (euro_to_dollar['Time']<='2019-12-31')]
```

In [97]:
```
import matplotlib.style as style
style.use('fivethirtyeight')

fig,ax=plt.subplots(figsize=(9,3))
ax.plot(corona_crisis['Time'],corona_crisis['Rolling_mean'], linewidth=2,color='#FF91CB')

ax.plot(corona_crisis_20['Time'],corona_crisis_20['Rolling_mean'], linewidth=3,color='#00148C')

ax.set_xticklabels([])
x = 0.02
for year in ["2016", "2017", "2018", "2019", "2020"]:
    ax.text(x,-0.08,year,alpha=0.5,fontsize=11,transform=plt.gca().transAxes)
    y +=0.22888

ax.set_yticklabels([])

y= 0.02
for rate in['1,05', '1,10', '1.15', '1.20']:
    ax.text(-.04,y,rate,alpha=0.5,fontsize=11,transform=plt.gca().transAxes)
    y += 0.248

ax.text(-0.05,1.2,"Euro USD rate peaked at 1.22 during 2020's COVID-19 crises", weight='bold',
        transform=plt.gca().transAxes)
ax.text(-.05,1.1,"Euro-USD exchanged rate between 2016 and 2020", size=12,
        transform=plt.gca().transAxes)
ax4.text(-0.05, -0.15, '@IMMAZ AHMED' + ' '*133 + 'Source: European Central Bank',
        color = '#f0f0f0', backgroundcolor = '#4d4d4d',
        size=14, transform=plt.gca().transAxes)
plt.tight_layout
plt.show()
```



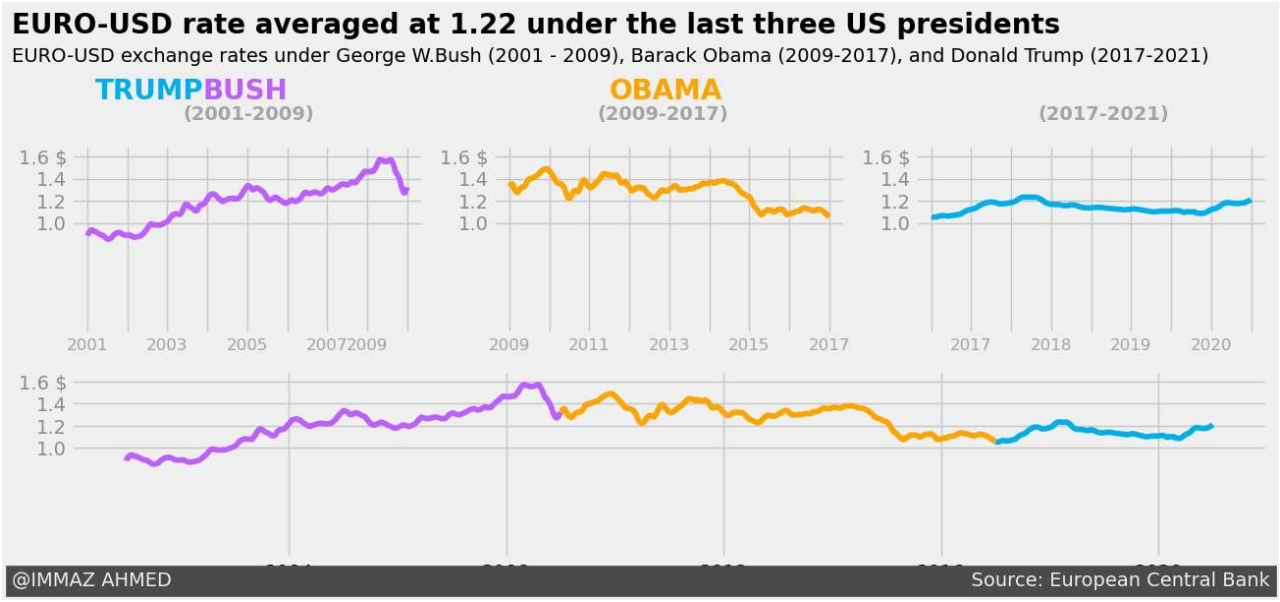
The Three US Presidencies Example

In [80]:

```
bush_obama_trump = euro_to_dollar.copy(
                   )[(euro_to_dollar['Time'].dt.year >= 2001) & (euro_to_dollar['Time'].dt.year
bush = bush_obama_trump.copy(
       )[bush_obama_trump['Time'].dt.year < 2009]
obama = bush_obama_trump.copy(
        )[(bush_obama_trump['Time'].dt.year >= 2009) & (bush_obama_trump['Time'].dt.year < 2017)
trump = bush_obama_trump.copy(
        )[(bush_obama_trump['Time'].dt.year >= 2017) & (bush_obama_trump['Time'].dt.year < 2021)
```

In [94]:

```
style.use("fivethirtyeight")
plt.figure(figsize=(14,8))

ax1 = plt.subplot(3,3,1)
ax2 = plt.subplot(3,3,2)
ax3 = plt.subplot(3,3,3)

ax4 = plt.subplot(3,1,2)
axes = [ax1, ax2, ax3, ax4]

for ax in axes:
    ax.set_ylim(0.0, 1.7)
    ax.set_yticks([1.0, 1.2, 1.4, 1.6])
    ax.set_yticklabels(['1.0', '1.2', '1.4', '1.6 $'],
                       alpha= 0.4)

ax1.plot(bush['Time'], bush['Rolling_mean'],
        color = '#BF5FFF')
ax1.set_xticklabels(['', '2001', '', '2003', '', '2005', '', '2007', '2009'],
                    alpha= 0.3, size=12)
ax1.text(0.11, 2.45, "BUSH", fontsize=20, weight = 'bold',
        color = '#BF5FFF', transform=plt.gca().transAxes)
ax1.text(0.093, 2.34, '(2001-2009)', weight= 'bold',
        alpha=0.3, transform=plt.gca().transAxes)
ax2.plot(obama['Time'], obama['Rolling_mean'],
        color= '#ffa500')
ax2.set_xticklabels(['', '2009', '', '2011', '', '2013', '', '2015', '', '2017'],
                    alpha=0.3, size=12)
ax2.text(0.45, 2.45, "OBAMA", fontsize=20, weight= 'bold',
        color= '#ffa500', transform=plt.gca().transAxes)
ax2.text(0.44, 2.34, '(2009-2017)', weight='bold',
        alpha=0.3, transform=plt.gca().transAxes)
ax3.plot(trump['Time'], trump['Rolling_mean'],
     color= '#00B2EE')
ax3.set_xticklabels(['', '2017', '', '2018', '', '2019', '', '2020', '', '2021'],
                    alpha=0.3, size=12)
ax3.text(0.02, 2.45, "TRUMP", fontsize=20, weight= 'bold',
        color= '#00B2EE', transform=plt.gca().transAxes)
ax3.text(0.808, 2.34, '(2017-2021)', weight='bold',
        alpha=0.3, transform=plt.gca().transAxes)
ax4.plot(bush['Time'], bush['Rolling_mean'],
        color= '#BF5FFF')
ax4.plot(obama['Time'], obama['Rolling_mean'],
        color= '#FFa500')
ax4.plot(trump['Time'], trump['Rolling_mean'],
        color= '#00B2EE')
ax1.text(-0.05, 2.8, 'EURO-USD rate averaged at 1.22 under the last three US presidents',
        fontsize=20, weight='bold', transform = plt.gca().transAxes)
ax1.text(-0.05,2.65, "EURO-USD exchange rates under George W.Bush (2001 - 2009), Barack Obama (20
        fontsize=14, transform = plt.gca().transAxes)
ax4.text(-0.05, -0.15, '@IMMAZ AHMED' + ' '*133 + 'Source: European Central Bank',
        color = '#f0f0f0', backgroundcolor = '#4d4d4d',
        size=14, transform=plt.gca().transAxes)
plt.tight_layout()
plt.show()
```


EURO-USD rate averaged at 1.22 under the last three US presidents
EURO-USD exchange rates under George W.Bush (2001 - 2009), Barack Obama (2009-2017), and Donald Trump (2017-2021)
TRUMPBUSH
(2001-2009)
OBAMA
(2009-2017)
(2017-2021)
1.6 $
1.4
1.2
1.0
2001
2003
2005
20072009
2009
2011
2013
2015
2017
2017
2018
2019
2020
@IMMAZ AHMED
Source: European Central Bank

1	