In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import warnings
import matplotlib.style as style
```

In [2]:
```
exchange_rates = pd.read_csv("euro-daily-hist_1999_2022.csv")
```

In [3]:
```
exchange_rates
```

Out[3]:

	Period\Unit:	[Australian dollar]	[Bulgarian lev]	[Brazilian real]	[Canadian dollar]	[Swiss franc]	[Chinese yuan renminbi]	[Cypriot pound]	[Czech koruna]
0	2023-12-15	1.6324	1.9558	5.4085	1.4653	0.9488	7.7812	NaN	24.477
1	2023-12-14	1.6288	1.9558	5.3349	1.4677	0.949	7.7866	NaN	24.408
2	2023-12-13	1.6452	1.9558	5.3609	1.4644	0.9452	7.7426	NaN	24.476
3	2023-12-12	1.6398	1.9558	5.3327	1.4656	0.9443	7.7447	NaN	24.42
4	2023-12-11	1.642	1.9558	5.3169	1.4609	0.9478	7.7206	NaN	24.367
...	...	...	...	...	...	...	...	...	...
6451	1999-01-08	1.8406	NaN	NaN	1.7643	1.6138	NaN	0.58187	34.938
6452	1999-01-07	1.8474	NaN	NaN	1.7602	1.6165	NaN	0.58187	34.886
6453	1999-01-06	1.8820	NaN	NaN	1.7711	1.6116	NaN	0.58200	34.850
6454	1999-01-05	1.8944	NaN	NaN	1.7965	1.6123	NaN	0.58230	34.917
6455	1999-01-04	1.9100	NaN	NaN	1.8004	1.6168	NaN	0.58231	35.107

6456 rows × 41 columns

In [4]:
```
#As we already know that periods = time but fro individual perspective it is difficult
```

In [5]:
```
exchange_rates.rename(columns = {'[Chinese yuan renminbi ]' : 'Chinese yuan renminbi',
```

In [6]:
```
exchange_rates['Time'] = pd.to_datetime(exchange_rates['Time'])
```

In [7]:
```
exchange_rates
```

Out[7]:

	Time	[Australian dollar]	[Bulgarian lev]	[Brazilian real]	[Canadian dollar]	[Swiss franc]	Chinese yuan renminbi	[Cypriot pound]	[Czech koruna]	[Danis krone
0	2023-12-15	1.6324	1.9558	5.4085	1.4653	0.9488	7.7812	NaN	24.477	7.45
1	2023-12-14	1.6288	1.9558	5.3349	1.4677	0.949	7.7866	NaN	24.408	7.456
2	2023-12-13	1.6452	1.9558	5.3609	1.4644	0.9452	7.7426	NaN	24.476	7.456
3	2023-12-12	1.6398	1.9558	5.3327	1.4656	0.9443	7.7447	NaN	24.42	7.456
4	2023-12-11	1.642	1.9558	5.3169	1.4609	0.9478	7.7206	NaN	24.367	7.456
...	...	...	...	...	...	...	...	...	...	...
6451	1999-01-08	1.8406	NaN	NaN	1.7643	1.6138	NaN	0.58187	34.938	7.443
6452	1999-01-07	1.8474	NaN	NaN	1.7602	1.6165	NaN	0.58187	34.886	7.443
6453	1999-01-06	1.8820	NaN	NaN	1.7711	1.6116	NaN	0.58200	34.850	7.445
6454	1999-01-05	1.8944	NaN	NaN	1.7965	1.6123	NaN	0.58230	34.917	7.449
6455	1999-01-04	1.9100	NaN	NaN	1.8004	1.6168	NaN	0.58231	35.107	7.450

6456 rows × 41 columns

In [8]:
```
exchange_rates = exchange_rates[exchange_rates['Chinese yuan renminbi'] != '-']
exchange_rates['Chinese yuan renminbi'] = exchange_rates['Chinese yuan renminbi'].asty
exchange_rates
```

```
C:\Users\iqra com\AppData\Local\Temp\ipykernel_22452\1107561610.py:2: SettingWithCopy
Warning:
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/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  exchange_rates['Chinese yuan renminbi'] = exchange_rates['Chinese yuan renminbi'].a
stype(float)
```

Out[8]:

	Time	[Australian dollar]	[Bulgarian lev]	[Brazilian real]	[Canadian dollar]	[Swiss franc]	Chinese yuan renminbi	[Cypriot pound]	[Czech koruna]	[Danis krone
0	2023-12-15	1.6324	1.9558	5.4085	1.4653	0.9488	7.7812	NaN	24.477	7.45
1	2023-12-14	1.6288	1.9558	5.3349	1.4677	0.949	7.7866	NaN	24.408	7.45
2	2023-12-13	1.6452	1.9558	5.3609	1.4644	0.9452	7.7426	NaN	24.476	7.45
3	2023-12-12	1.6398	1.9558	5.3327	1.4656	0.9443	7.7447	NaN	24.42	7.45
4	2023-12-11	1.642	1.9558	5.3169	1.4609	0.9478	7.7206	NaN	24.367	7.45
...	...	...	...	...	...	...	...	...	...	
6451	1999-01-08	1.8406	NaN	NaN	1.7643	1.6138	NaN	0.58187	34.938	7.44
6452	1999-01-07	1.8474	NaN	NaN	1.7602	1.6165	NaN	0.58187	34.886	7.44
6453	1999-01-06	1.8820	NaN	NaN	1.7711	1.6116	NaN	0.58200	34.850	7.44
6454	1999-01-05	1.8944	NaN	NaN	1.7965	1.6123	NaN	0.58230	34.917	7.44
6455	1999-01-04	1.9100	NaN	NaN	1.8004	1.6168	NaN	0.58231	35.107	7.45

6395 rows × 41 columns

In [ ]:

In [9]:
```
exchange_rates.reset_index(drop=True, inplace=True)
```

In [10]:
```
chinese_to_dollar = exchange_rates[['Time' , 'Chinese yuan renminbi']].copy()
```

In [11]:
```
exchange_rates.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6395 entries, 0 to 6394
Data columns (total 41 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Time                   6395 non-null   datetime64[ns]
 1   [Australian dollar ]   6395 non-null   object
 2   [Bulgarian lev ]       5996 non-null   object
 3   [Brazilian real ]      6127 non-null   object
 4   [Canadian dollar ]     6395 non-null   object
 5   [Swiss franc ]         6395 non-null   object
 6   Chinese yuan renminbi  6127 non-null   float64
 7   [Cypriot pound ]       2305 non-null   object
 8   [Czech koruna ]        6395 non-null   object
 9   [Danish krone ]        6395 non-null   object
 10  [Estonian kroon ]      3075 non-null   object
 11  [UK pound sterling ]   6395 non-null   object
 12  [Greek drachma ]       515 non-null    object
 13  [Hong Kong dollar ]    6395 non-null   object
 14  [Croatian kuna ]       5880 non-null   object
 15  [Hungarian forint ]    6395 non-null   object
 16  [Indonesian rupiah ]   6395 non-null   object
 17  [Israeli shekel ]      6127 non-null   object
 18  [Indian rupee ]        6127 non-null   object
 19  [Iceland krona ]       4049 non-null   float64
 20  [Japanese yen ]        6395 non-null   object
 21  [Korean won ]          6395 non-null   object
 22  [Lithuanian litas ]    4098 non-null   object
 23  [Latvian lats ]        3843 non-null   object
 24  [Maltese lira ]        2305 non-null   object
 25  [Mexican peso ]        6395 non-null   object
 26  [Malaysian ringgit ]   6395 non-null   object
 27  [Norwegian krone ]     6395 non-null   object
 28  [New Zealand dollar ]  6395 non-null   object
 29  [Philippine peso ]     6395 non-null   object
 30  [Polish zloty ]        6395 non-null   object
 31  [Romanian leu ]        6394 non-null   float64
 32  [Russian rouble ]      5933 non-null   object
 33  [Swedish krona ]       6395 non-null   object
 34  [Singapore dollar ]    6395 non-null   object
 35  [Slovenian tolar ]     2050 non-null   object
 36  [Slovak koruna ]       2561 non-null   object
 37  [Thai baht ]           6395 non-null   object
 38  [Turkish lira ]        6394 non-null   float64
 39  [US dollar ]           6395 non-null   object
 40  [South African rand ]  6395 non-null   object
dtypes: datetime64[ns](1), float64(4), object(36)
memory usage: 2.0+ MB
```

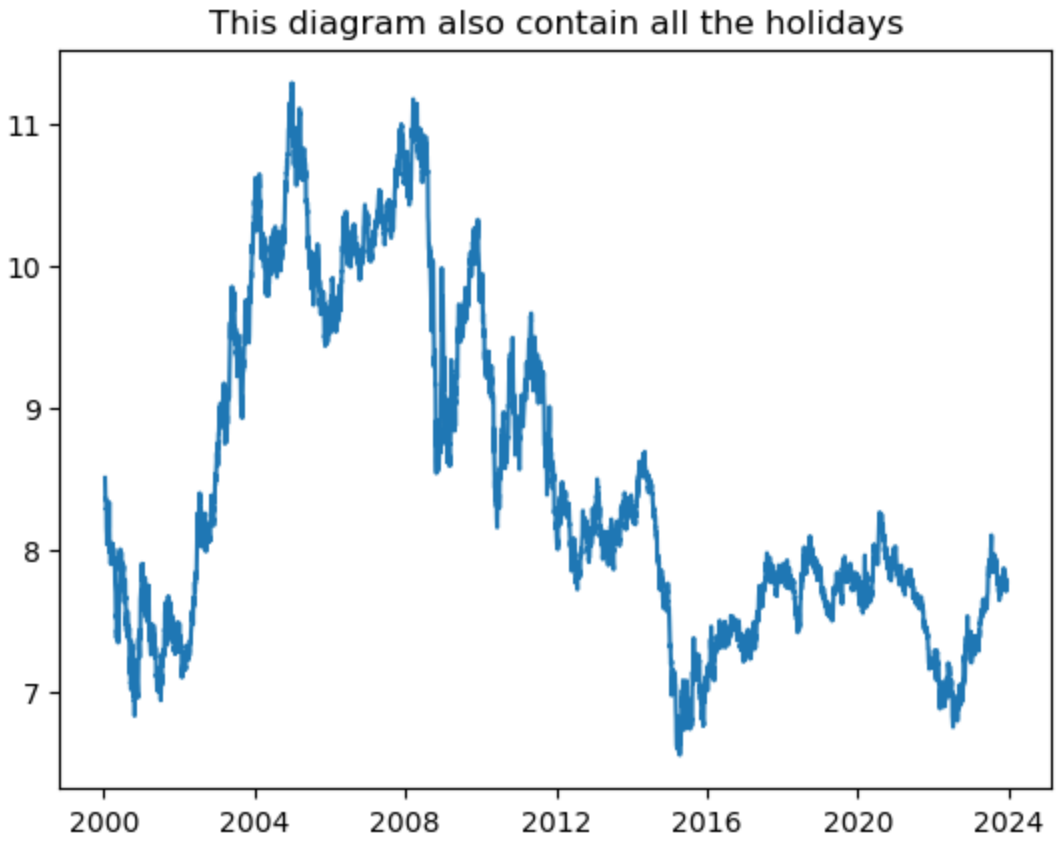
In [ ]:

In [12]:
```
#chinese_to_dollar = chinese_to_dollar[chinese_to_dollar['Chinese yuan renminbi'] != '
#chinese_to_dollar['Chinese yuan renminbi'] = chinese_to_dollar['Chinese yuan renminbi
#chinese_to_dollar
```

In [13]:
```
chinese_to_dollar.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6395 entries, 0 to 6394
Data columns (total 2 columns):
 #   Column                 Non-Null Count  Dtype         
---  ------                 --------------  -----         
 0   Time                   6395 non-null   datetime64[ns]
 1   Chinese yuan renminbi  6127 non-null   float64       
dtypes: datetime64[ns](1), float64(1)
memory usage: 100.1 KB
```

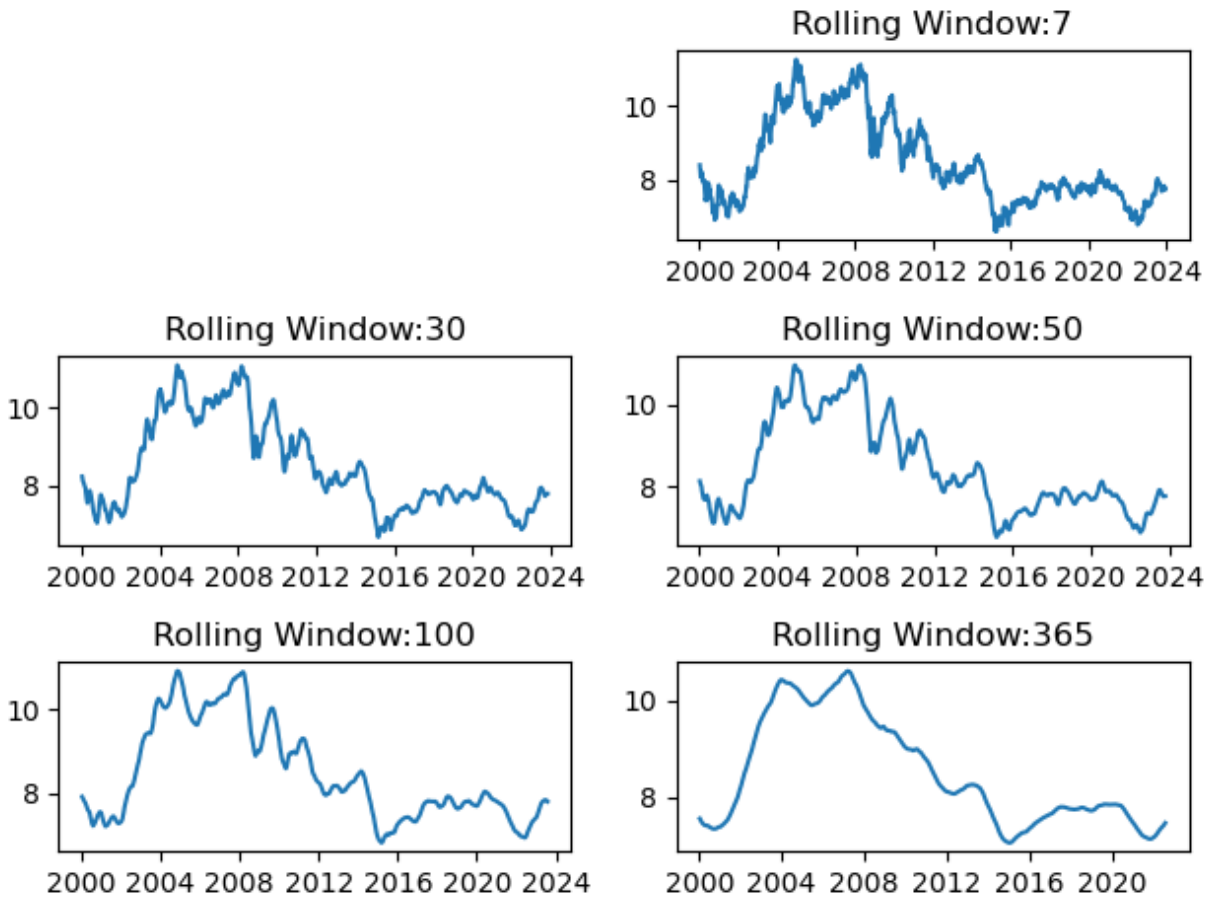
In [14]:
```
plt.title('This diagram also contain all the holidays')
plt.plot(chinese_to_dollar['Time'],chinese_to_dollar['Chinese yuan renminbi'])
plt.show()
```



Now we make analysis on rolling meaning we find what insight we can get if we convert on week, days, month and year how much fluctuation are there

In [15]:
```
plt.title('Graph based on rolling')
plt.plot(chinese_to_dollar["Time"],chinese_to_dollar["Chinese yuan renminbi"])

for i, rolling_mean in zip ([2,3,4,5,6,7],
                            [7,30,50,100,365]) :
    plt.subplot(3,2,i)
    plt.plot(chinese_to_dollar["Time"],
             chinese_to_dollar["Chinese yuan renminbi"].rolling(rolling_mean).mean())
    plt.title("Rolling Window:" + str(rolling_mean))
plt.tight_layout()
plt.show()
```

```
C:\Users\iqra com\AppData\Local\Temp\ipykernel_22452\1713599069.py:6: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and will be removed two minor releases later; explicitly call ax.remove() as needed.
  plt.subplot(3,2,i)
```



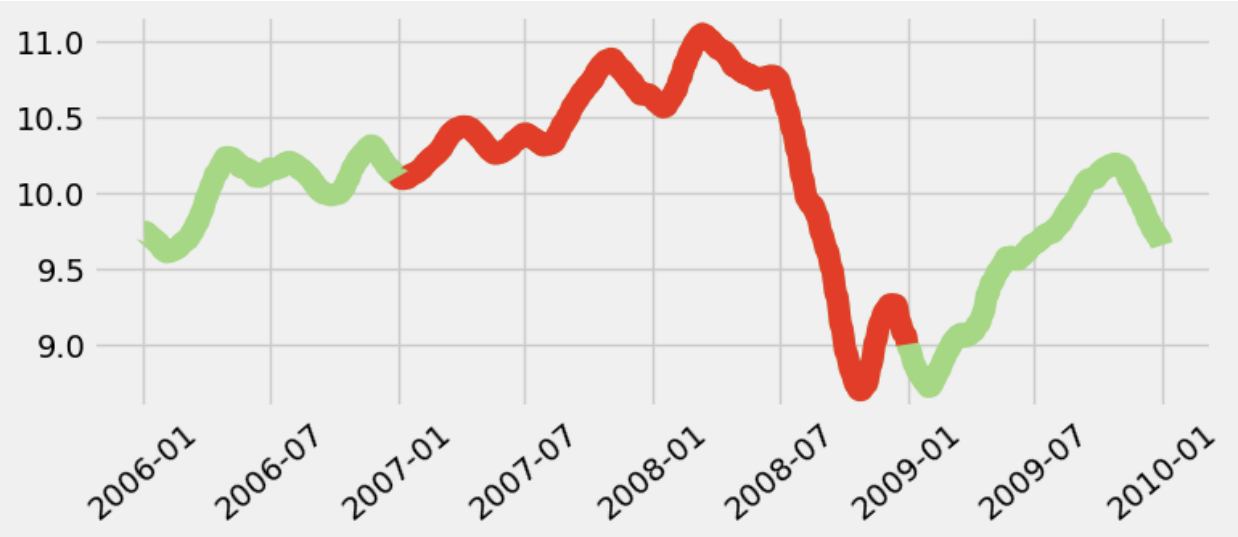
In [16]:
```
chinese_to_dollar["rolling_mean"] = chinese_to_dollar["Chinese yuan renminbi"].rolling
```

In [17]:
```
financial_rises_china = chinese_to_dollar.copy(
)[(chinese_to_dollar['Time'].dt.year >=2006
  )&(chinese_to_dollar['Time'].dt.year <=2009)]
financial_rises_china_7_8 = chinese_to_dollar.copy(
)[(chinese_to_dollar['Time'].dt.year >=2007
  )&(chinese_to_dollar['Time'].dt.year <=2008)]
```

In [18]:
```
style.use('fivethirtyeight')

fig, ax = plt.subplots(figsize = (8,3))
ax.plot(financial_rises_china["Time"],
       financial_rises_china["rolling_mean"],
       linewidth = 10 , color = "#A6D785")

ax.plot(financial_rises_china_7_8["Time"],
       financial_rises_china_7_8["rolling_mean"],
       linewidth = 10 , color = "#e23d28")
plt.xticks(rotation=40)

#ax.set_xticklabels([])
```

```
#x =0.02
#for year in ["2006",'2007',"2008","2009","2010"]:
#    ax.text(x , -0.02, aplha= 0.5, fontsize = 30, transform = plt.gca().transAxes)
#    x += 0.22888
#ax.set_xticklabels([])
```

Out[18]:
```
(array([13149., 13330., 13514., 13695., 13879., 14061., 14245., 14426.,
        14610.]),
 [Text(13149.0, 0, '2006-01'),
  Text(13330.0, 0, '2006-07'),
  Text(13514.0, 0, '2007-01'),
  Text(13695.0, 0, '2007-07'),
  Text(13879.0, 0, '2008-01'),
  Text(14061.0, 0, '2008-07'),
  Text(14245.0, 0, '2009-01'),
  Text(14426.0, 0, '2009-07'),
  Text(14610.0, 0, '2010-01')])
```

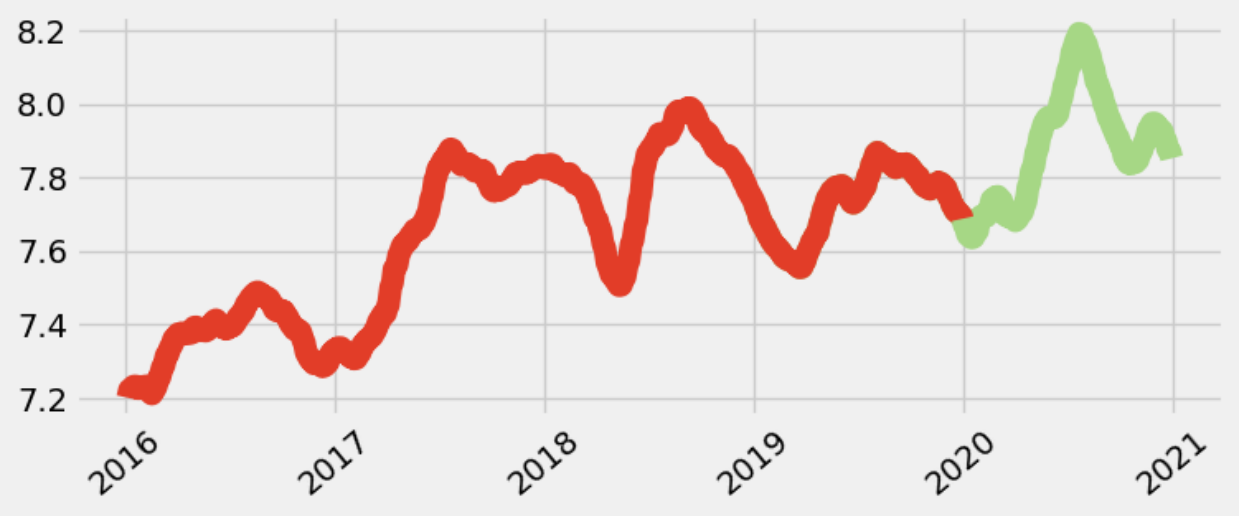


EFFECT OF COVID 19 ONTO CHINA ECONOMY

In [19]:
```
corona_rises_china = chinese_to_dollar.loc(
)[(chinese_to_dollar['Time']>='2019-01-01'
  )&(chinese_to_dollar['Time']<='2020-12-31')]
corona_rises_china_7_8 = chinese_to_dollar.loc(
)[(chinese_to_dollar['Time']>='2016-01-01'
  )&(chinese_to_dollar['Time']<='2019-12-31')]
```

In [20]:
```
style.use('fivethirtyeight')

fig, ax = plt.subplots(figsize = (8,3))
ax.plot(corona_rises_china["Time"],
       corona_rises_china["rolling_mean"],
       linewidth = 10 , color = "#A6D785")

ax.plot(corona_rises_china_7_8["Time"],
       corona_rises_china_7_8["rolling_mean"],
       linewidth = 10 , color = "#e23d28")
plt.xticks(rotation=40)
```

Out[20]:
```
(array([16436., 16801., 17167., 17532., 17897., 18262., 18628.]),
 [Text(16436.0, 0, '2015'),
  Text(16801.0, 0, '2016'),
  Text(17167.0, 0, '2017'),
  Text(17532.0, 0, '2018'),
  Text(17897.0, 0, '2019'),
  Text(18262.0, 0, '2020'),
  Text(18628.0, 0, '2021')])
```



As you can see from above grapgh covid have siginifican effect ON chinese monetary and economy.

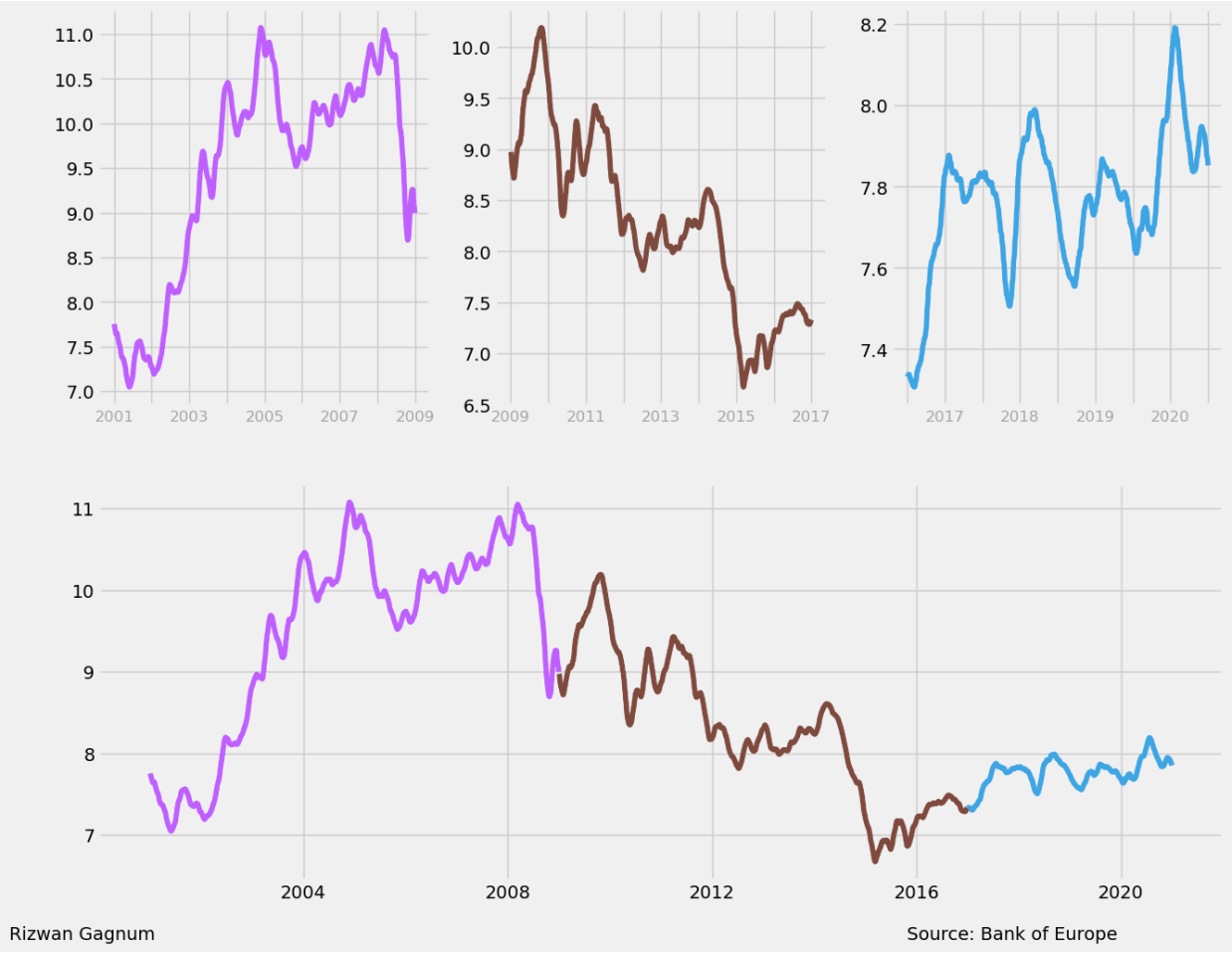
Major event that happen in china that effect china economy as you can see in the graph

In [21]:
```
bush_obama_trump = chinese_to_dollar.copy(
)[(chinese_to_dollar['Time'].dt.year >=2001
  )&(chinese_to_dollar['Time'].dt.year <2021)]
china1 = bush_obama_trump.copy(
)[bush_obama_trump['Time'].dt.year < 2009]
china2 = bush_obama_trump.copy(
)[(bush_obama_trump['Time'].dt.year >=2009
  )&(bush_obama_trump['Time'].dt.year <2017)]
china3 = bush_obama_trump.copy(
)[(bush_obama_trump['Time'].dt.year >=2017
  )&(bush_obama_trump['Time'].dt.year < 2021)]
```

In [24]:
```
style.use('fivethirtyeight')
plt.figure(figsize = (14,18))
#x = plt.subplot(3,3,1)
ax1 = plt.subplot(3,3,1)
ax2 = plt.subplot(3,3,2)
ax3 = plt.subplot(3,3,3)
ax4 = plt.subplot(3,1,2)

#AX1
ax1.plot(china1['Time'] , china1['rolling_mean'],
        color = '#BF5FFF')
```

```
ax1.set_xticklabels(['' , '2001' , '' , '2003' , '' , '2005' , '' , '2007' , '' ,'2009
                  alpha = 0.3 , size = 12)
ax1.text(0.11 , 0.25 , 'Exchange rate' , fontsize = 12 , weight = 'bold',
        color = '#BF5FFF' , transform = plt.gca().transAxes)
ax1.text(0.093 , 0.25, '(2001-2009)', weight = 'bold',
         alpha = 0.3, transform = plt.gca().transAxes)

##AX2
ax2.plot(china2['Time'] , china2['rolling_mean'],
        color = '#7F4A3E')
ax2.set_xticklabels(['' , '2009' , '' , '2011' , '' , '2013' , '' , '2015' , '' ,'2017
                  alpha = 0.3 , size = 12)
ax2.text(0.11 , 0.25 , 'Exchange rate' , fontsize = 12 , weight = 'bold',
        color = '#7F4A3E', transform = plt.gca().transAxes)
ax2.text(0.093 , 0.25, '(2009-2017)', weight = 'bold',
        alpha = 0.3, transform = plt.gca().transAxes)

##AX3
ax3.plot(china3['Time'] , china3['rolling_mean'],
        color = '#3DA5E4')
ax3.set_xticklabels(['' , '2017','', '2018','' , '2019' , '' , '2020','' , '2021',''],
                  alpha = 0.3 , size = 12)
ax3.text(0.11 , 0.25 , 'Exchange rate' , fontsize = 12 , weight = 'bold',
        color = '#3DA5E4' , transform = plt.gca().transAxes)
ax3.text(0.093 , 0.25, '(2017-2021)', weight = 'bold',
         alpha = 0.3, transform = plt.gca().transAxes)

##AX4
ax4.plot(china1['Time'] , china1['rolling_mean'],
        color = '#BF5FFF')
ax4.plot(china2['Time'] , china2['rolling_mean'],
        color = '#7F4A3E')
ax4.plot(china3['Time'] , china3['rolling_mean'],
        color = '#3DA5E4')
### Signature
ax4.text(-0.08 , -0.15, 'Rizwan Gagnum' + ' '*133 + 'Source: Bank of Europe',
         size =14 , transform = plt.gca().transAxes)
plt.tight_layout()
plt.show()
```

```
C:\Users\iqra com\AppData\Local\Temp\ipykernel_22452\392685953.py:12: UserWarning: Fi
xedFormatter should only be used together with FixedLocator
  ax1.set_xticklabels(['' , '2001' , '' , '2003' , '' , '2005' , '' , '2007' , '' ,'2
009'],
C:\Users\iqra com\AppData\Local\Temp\ipykernel_22452\392685953.py:22: UserWarning: Fi
xedFormatter should only be used together with FixedLocator
  ax2.set_xticklabels(['' , '2009' , '' , '2011' , '' , '2013' , '' , '2015' , '' ,'2
017'],
C:\Users\iqra com\AppData\Local\Temp\ipykernel_22452\392685953.py:32: UserWarning: Fi
xedFormatter should only be used together with FixedLocator
  ax3.set_xticklabels(['' , '2017','', '2018','' , '2019' , '' , '2020','' , '202
1',''],
C:\Users\iqra com\AppData\Local\Temp\ipykernel_22452\392685953.py:49: UserWarning: Ti
ght layout not applied. tight_layout cannot make axes width small enough to accommoda
te all axes decorations
  plt.tight_layout()
```



Conclusion

In conclusion, the exchange rates of China's currency, the Yuan Renminbi, have been significantly influenced by major events over the years. For instance, the global financial crisis of 2008 had a profound impact, leading to a decrease in the value of the Chinese Yuan as economic uncertainties prevailed.

On the contrary, strategic economic initiatives like the Belt and Road Initiative and periods of robust economic growth have contributed to an increase in the value of the Yuan. Additionally, shifts in exchange rate policies by the People's Bank of China, trade tensions with other nations, and geopolitical events have played crucial roles in shaping the fluctuations in China's currency exchange rates.

These major events highlight the sensitivity and responsiveness of the Chinese Yuan to changes in the economic and geopolitical landscape. A nuanced understanding of these events is essential for comprehending the dynamics behind the fluctuations, as they underscore the intricate relationship between global events and the valuation of China's currency in the international market.

In [ ]: