TRENDING BABY NAMES DATA ANALYSIS

In [1]:
```
import math
import collections

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

%matplotlib inline
```

In [2]:
```
pd.options.display.max_rows = 8
```

In [3]:
```
import zipfile
```

In [4]:
```
cd Names_data_Analysis
```

```
C:\Users\kheni\Downloads\Names_data_Analysis
```

In [5]:
```
zipfile.ZipFile('names.zip').extractall('baby_names')
```

In [6]:
```
ls
```

```
 Volume in drive C is OS
 Volume Serial Number is D4B8-A571

 Directory of C:\Users\kheni\Downloads\Names_Data_Analysis

27/09/2023  02:47    <DIR>          .
27/09/2023  02:52    <DIR>          ..
25/09/2023  00:17    <DIR>          .ipynb_checkpoints
25/09/2023  13:42         9,209,242 all_years.csv.gz
26/09/2023  15:05             3,761 app.py
24/09/2023  03:58    <DIR>          baby_names
26/09/2023  04:38         1,315,345 Baby_Names.ipynb
24/09/2023  03:55         7,405,814 names.zip
27/09/2023  02:34            17,374 Trending Baby Names Data Analysis
Report.docx
27/09/2023  02:47           123,403 Trending Baby Names Data Analysis
Report.pdf
               6 File(s)     18,074,939 bytes
               4 Dir(s)  139,576,651,776 bytes free
```

In [7]:
```
ls baby_names
```

```
 Volume in drive C is OS
 Volume Serial Number is D4B8-A571

 Directory of C:\Users\kheni\Downloads\Names_Data_Analysis\baby_nam
es

24/09/2023  03:58    <DIR>          .
27/09/2023  02:47    <DIR>          ..
27/09/2023  02:53            24,933 yob1880.txt
27/09/2023  02:53            24,052 yob1881.txt
27/09/2023  02:53            26,559 yob1882.txt
27/09/2023  02:53            26,002 yob1883.txt
27/09/2023  02:53            28,670 yob1884.txt
27/09/2023  02:53            28,625 yob1885.txt
27/09/2023  02:53            29,822 yob1886.txt
27/09/2023  02:53            29,531 yob1887.txt
27/09/2023  02:53            33,064 yob1888.txt
27/09/2023  02:53            32,297 yob1889.txt
27/09/2023  02:53            33,621 yob1890.txt
```

In [8]:
```
# Opening the file for the year 2022

open('baby_names/yob2022.txt', 'r').readlines()[:10]
```

Out[8]:
```
['Olivia,F,16573\n',
 'Emma,F,14435\n',
 'Charlotte,F,12891\n',
 'Amelia,F,12333\n',
 'Sophia,F,12310\n',
 'Isabella,F,11662\n',
 'Ava,F,11039\n',
 'Mia,F,11018\n',
 'Evelyn,F,9289\n',
 'Luna,F,8922\n']
```

In [9]:
```
# Reading the file as dataframe using pandas

pd.read_csv('baby_names/yob2022.txt', names = ['name', 'sex', 'number']
```

Out[9]:

	name	sex	number
0	Olivia	F	16573
1	Emma	F	14435
2	Charlotte	F	12891
3	Amelia	F	12333
...	...	...	...
31911	Zydn	M	5
31912	Zylon	M	5
31913	Zymeer	M	5
31914	Zymeire	M	5

31915 rows × 3 columns

In [10]:
```
# Creating a year column where all elements are set to 2022

pd.read_csv('baby_names/yob2022.txt', names = ['name', 'sex', 'number']
```

Out[10]:

	name	sex	number	year
0	Olivia	F	16573	2022
1	Emma	F	14435	2022
2	Charlotte	F	12891	2022
3	Amelia	F	12333	2022
...	...	...	...	...
31911	Zydn	M	5	2022
31912	Zylon	M	5	2022
31913	Zymeer	M	5	2022
31914	Zymeire	M	5	2022

31915 rows × 4 columns

In [11]:
```
# Creating a common file for all the years and assigning respective yea

all_years = pd.concat(pd.read_csv(f'baby_names/yob{year}.txt', names =
                      for year in range(1880, 2023))
```

In [12]:
```
# Printing the inforation of the all_years variable

all_years.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2085158 entries, 0 to 31914
Data columns (total 4 columns):
 #   Column  Dtype
---  ------  -----
 0   name    object
 1   sex     object
 2   number  int64
 3   year    int64
dtypes: int64(2), object(2)
memory usage: 79.5+ MB
```

In [13]:
```
# Checking the min and max of the column year

all_years.year.min(), all_years.year.max()
```

Out[13]: (1880, 2022)

In [14]:
```
# Saving the combined file ZIP file and dropping the index

all_years.to_csv('all_years.csv.gz', index = False)
```

In [15]:
```
# Reading all_years.csv file

all_years = pd.read_csv('all_years.csv.gz')
```

In [16]:
```
all_years
```

Out[16]:

	name	sex	number	year
0	Mary	F	7065	1880
1	Anna	F	2604	1880
2	Emma	F	2003	1880
3	Elizabeth	F	1939	1880
...	...	...	...	...
2085154	Zydn	M	5	2022
2085155	Zylon	M	5	2022
2085156	Zymeer	M	5	2022
2085157	Zymeire	M	5	2022

2085158 rows × 4 columns

In [17]:
```
# Setting the 'sex', 'name', and 'year' column as index

all_years_indexed = all_years.set_index(['sex', 'name', 'year']).sort_i
```

In [18]:
```
# Count of number by year where name = 'Emma', and sex = 'F'

all_years_indexed.loc[('F', 'Emma')]
```

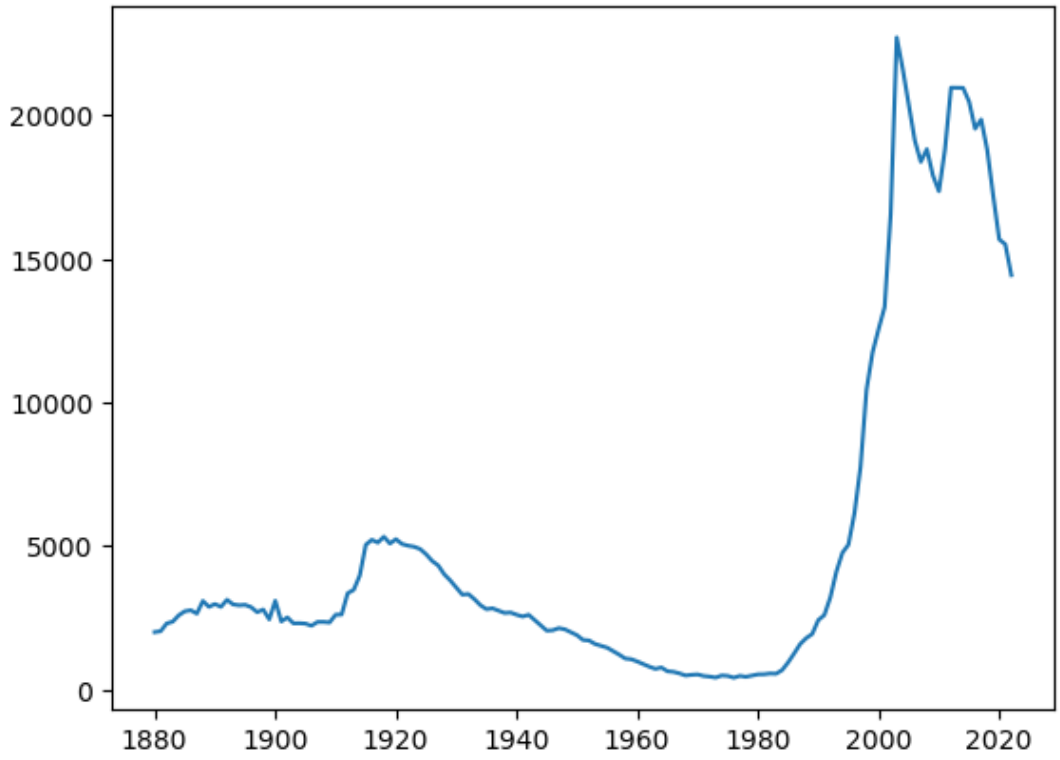
Out[18]:

	number
year	
1880	2003
1881	2034
1882	2303
1883	2367
...	...
2019	17194
2020	15680
2021	15510
2022	14435

143 rows × 1 columns

In [19]:
```
# Plotting it using matplotlib

plt.plot(all_years_indexed.loc[('F', 'Emma')])
```

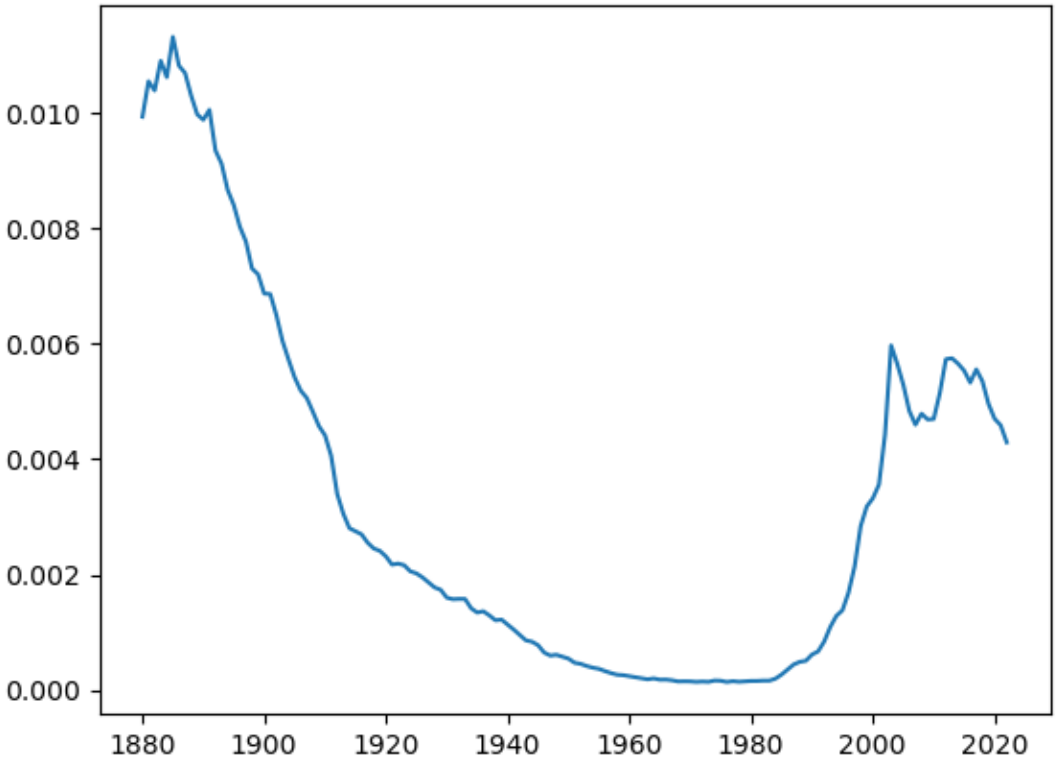
Out[19]:
```
[<matplotlib.lines.Line2D at 0x1df38c083a0>]
```



In [20]:
```
# Normalize F/'Emma' time series by the total number of births each yea

plt.plot(all_years_indexed.loc[('F', 'Emma')] / all_years.groupby('year
```

```
C:\Users\kheni\AppData\Local\Temp\ipykernel_20076\3431854176.py:3: Fut
ureWarning: The default value of numeric_only in DataFrameGroupBy.sum
is deprecated. In a future version, numeric_only will default to Fals
e. Either specify numeric_only or select only columns which should be
valid for the function.
  plt.plot(all_years_indexed.loc[('F', 'Emma')] / all_years.groupby('y
ear').sum())
```

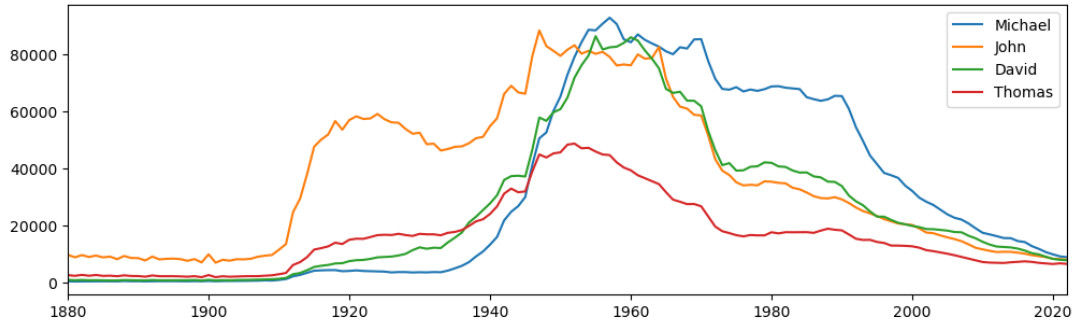
Out[20]:
```
[<matplotlib.lines.Line2D at 0x1df4e9590f0>]
```



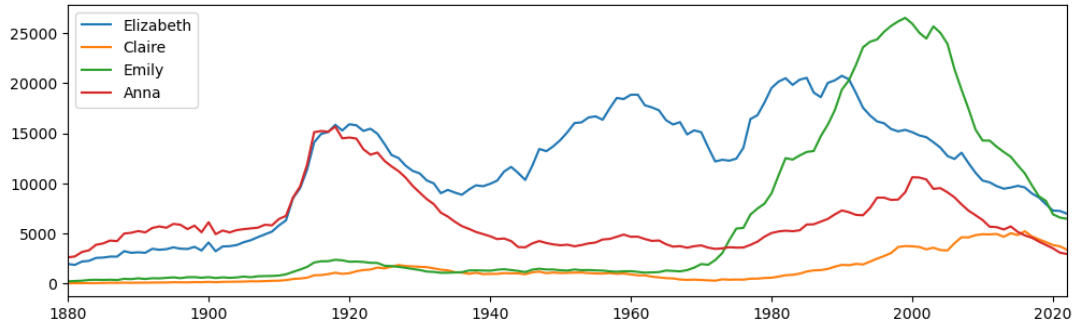
In [21]:
```
# Plotting number of sex/name babies as a function of year

def plotname(sex, name):
    data = all_years_indexed.loc[(sex, name)]

    plt.plot(data.index, data.values, label = name)
    plt.axis(xmin = 1880, xmax = 2022)
```

In [22]:
```
# Combining several "plotname()" plots for given sex and list of names

def comparenames(sex, names):
    plt.figure(figsize = (12, 3.5))

    for name in names:
        plotname(sex, name)

    plt.legend()
```

In [23]:
```
comparenames('M', ['Michael', 'John', 'David', 'Thomas'])
```

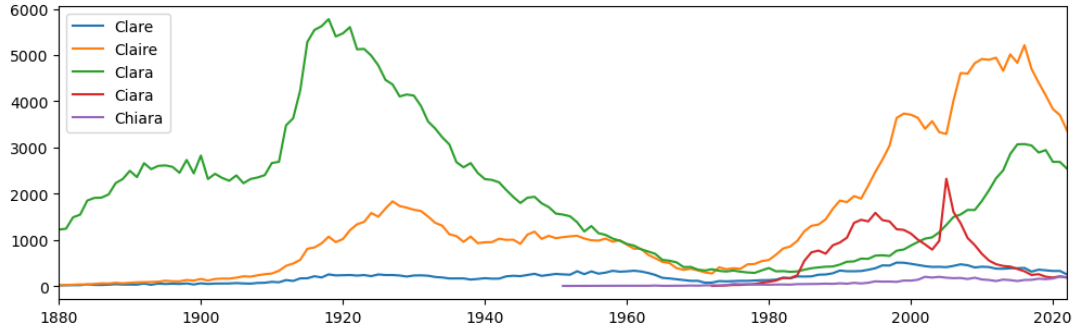


In [24]:
```
comparenames('F', ['Elizabeth', 'Claire', 'Emily', 'Anna'])
```



In [25]:
```
# There are different spellings or pronounciations for the name 'Claire

claires = ['Clare', 'Claire', 'Clara', 'Ciara', 'Chiara']
```

In [26]:
```
comparenames('F', claires)
```



In [27]:
```
all_years_indexed.loc[('F', claires),:]
```

Out[27]:

			number
sex	**name**	**year**	
F	**Clare**	**1880**	15
		1881	20
		1882	21
		1883	22
	...	**...**	...
	Chiara	**2019**	152
		2020	166
		2021	206
		2022	182

542 rows × 1 columns

In [28]:
```
# 'Pivoting' the third level of the multiindex (years) to create a row

all_years_indexed.loc[('F', claires),:].unstack(level = 2)
```

Out[28]:

		number									
	year	**1880**	**1881**	**1882**	**1883**	**1884**	**1885**	**1886**	**1887**	**1888**	**188**
sex	**name**										
F	**Clare**	15.0	20.0	21.0	22.0	38.0	27.0	26.0	31.0	43.0	3
	Claire	21.0	23.0	30.0	38.0	33.0	49.0	61.0	58.0	73.0	6
	Clara	1226.0	1242.0	1490.0	1548.0	1852.0	1910.0	1916.0	1984.0	2230.0	231
	Ciara	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
	Chiara	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N

5 rows × 143 columns

In [29]:
```
# 'Pivoting' the second level of the multiindex (names) to create a row

all_years_indexed.loc[('F', claires), :].unstack(level = 1)
```

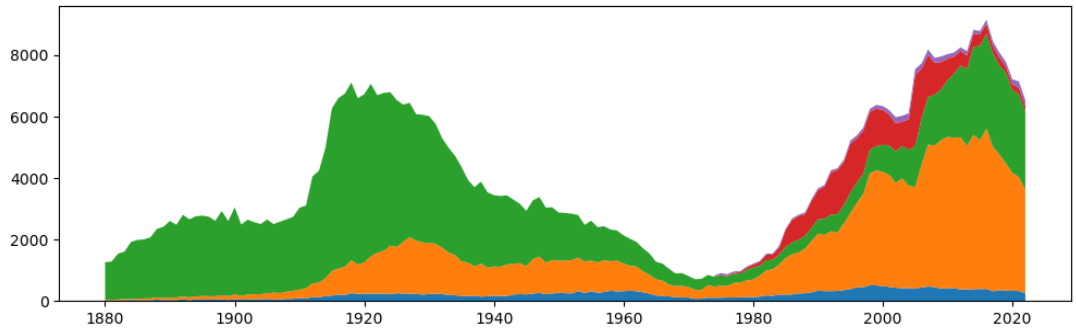
Out[29]:

		number				
	name	**Clare**	**Claire**	**Clara**	**Ciara**	**Chiara**
sex	**year**					
F	**1880**	15.0	21.0	1226.0	NaN	NaN
	1881	20.0	23.0	1242.0	NaN	NaN
	1882	21.0	30.0	1490.0	NaN	NaN
	1883	22.0	38.0	1548.0	NaN	NaN
	...	...	...	...	...	...
	2019	344.0	4128.0	2945.0	204.0	152.0
	2020	329.0	3834.0	2691.0	187.0	166.0
	2021	329.0	3700.0	2688.0	213.0	206.0
	2022	254.0	3365.0	2541.0	192.0	182.0

143 rows × 5 columns

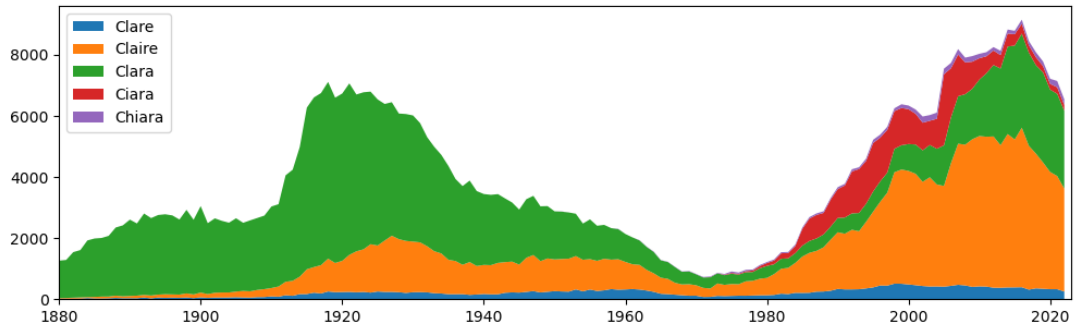
In [30]:
```
# Making an area plot using names and years table

plt.figure(figsize = (12, 3.5))
plt.stackplot(range(1880,2023), all_years_indexed.loc[('F', claires), :
```

Out[30]:
```
[<matplotlib.collections.PolyCollection at 0x1df359224d0>,
 <matplotlib.collections.PolyCollection at 0x1df35922830>,
 <matplotlib.collections.PolyCollection at 0x1df35922b90>,
 <matplotlib.collections.PolyCollection at 0x1df35922ef0>,
 <matplotlib.collections.PolyCollection at 0x1df35923280>]
```



In [31]:
```
# Filling the null values and adding the labels to plot

plt.figure(figsize = (12, 3.5))
plt.stackplot(range(1880,2023), all_years_indexed.loc[('F', claires), :

plt.legend(loc = 'upper left')
plt.axis(xmin = 1880, xmax = 2023)
```

Out[31]: (1880.0, 2023.0, 0.0, 9598.05)



In [32]:
```
# Setting year as index

all_years_byyears = all_years.set_index(['sex', 'year']).sort_index()
```

In [33]:
```
all_years_byyears
```

Out[33]:

		name	number
sex	**year**		
F	**1880**	Mary	7065
	1880	Anna	2604
	1880	Emma	2003
	1880	Elizabeth	1939
...	**...**	...	...
M	**2022**	Zydn	5
	2022	Zylon	5
	2022	Zymeer	5
	2022	Zymeire	5

2085158 rows × 2 columns

In [34]:
```
# Sorting the values by number in descending order for the year 2022 of
all_years_byyears.loc[('M', 2022)].sort_values('number', ascending = Fa
```

Out[34]:

		name	number
sex	**year**		
M	**2022**	Liam	20456
	2022	Noah	18621
	2022	Oliver	15076
	2022	James	12028
	...	...	...
	2022	Gianluka	5
	2022	Gerrell	5
	2022	Germani	5
	2022	Zymeire	5

14255 rows × 2 columns

In [35]:
```
# Sorting the values by number in descending order for the year 2022 of
all_years_byyears.loc[('F', 2022)].sort_values('number', ascending = Fa
```

Out[35]:

		name	number
sex	**year**		
F	**2022**	Olivia	16573
	2022	Emma	14435
	2022	Charlotte	12891
	2022	Amelia	12333
	...	...	...
	2022	Galadriel	5
	2022	Galya	5
	2022	Ganeev	5
	2022	Zymeria	5

17660 rows × 2 columns

In [36]:
```
all_years_byyears.loc[('F', 2022)].sort_values('number', ascending = Fa
```

Out[36]:
```
0       Olivia
1         Emma
2    Charlotte
3       Amelia
       ...
6          Ava
7          Mia
8       Evelyn
9         Luna
Name: name, Length: 10, dtype: object
```

In [37]:
```
# Making a function to get top ten names for sex and year

def getyear(sex, year):
    return (all_years_byyears.loc[(sex, year)]
            .sort_values('number', ascending = False)
            .head(10)
            .reset_index()
            .name)
```

In [38]:
```
# Use of the function to fetch top 10 names of year 1880 of gender 'M'

getyear('M', 1880)
```

Out[38]:
```
0       John
1    William
2      James
3    Charles
      ...
6     Joseph
7     Thomas
8      Henry
9     Robert
Name: name, Length: 10, dtype: object
```

In [39]:
```
# Printing the table of the names for each year of gender 'M'

pd.DataFrame({year : getyear('M', year) for year in range(1880, 2023)})
```

Out[39]:

	1880	1881	1882	1883	1884	1885	1886	1887	1888	18
0	John	John	John	John	John	John	John	John	John	Jo
1	William	William	William	William	William	William	William	William	William	Willi
2	James	James	James	James	James	James	James	James	James	Jam
3	Charles	George	George	Charles	George	George	George	George	George	Geo
...	...	...	...	...	...	...	...	...	...	
6	Joseph	Joseph	Joseph	Joseph	Joseph	Joseph	Joseph	Joseph	Joseph	Jose
7	Thomas	Henry	Thomas	Henry	Thomas	Henry	Robert	Henry	Robert	Ha
8	Henry	Thomas	Henry	Robert	Henry	Robert	Henry	Thomas	Harry	Rob
9	Robert	Edward	Robert	Thomas	Robert	Thomas	Thomas	Edward	Henry	Edw

10 rows × 143 columns

In [40]:
```
# Printing the table of the names for each year of gender 'F'

pd.DataFrame({year : getyear('F', year) for year in range(1880, 2023)})
```

Out[40]:

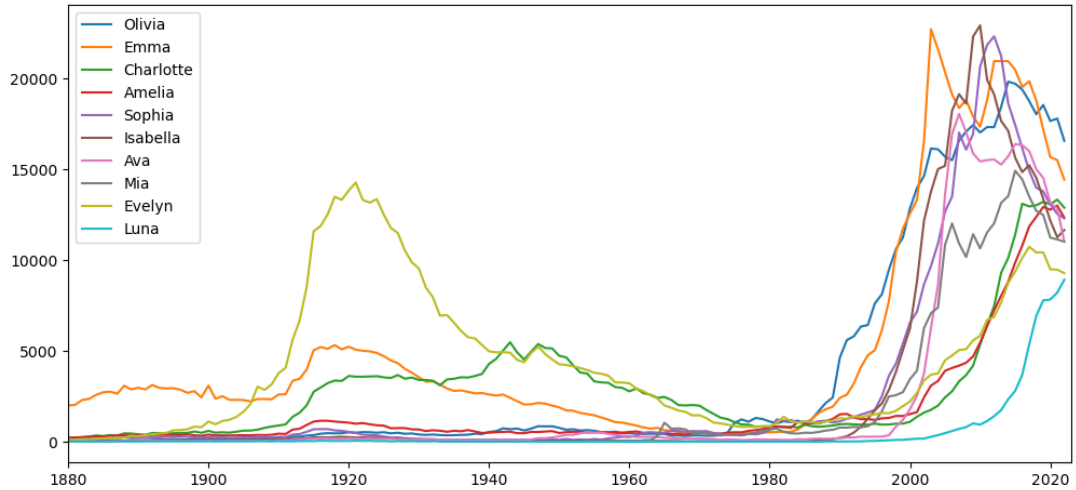
	1880	1881	1882	1883	1884	1885	1886	1887	1
0	Mary	Mary	Mary	Mary	Mary	Mary	Mary	Mary	M
1	Anna	Anna	Anna	Anna	Anna	Anna	Anna	Anna	A
2	Emma	Emma	Emma	Emma	Emma	Emma	Emma	Elizabeth	Elizab
3	Elizabeth	Elizabeth	Elizabeth	Elizabeth	Elizabeth	Elizabeth	Elizabeth	Emma	Em
...	...	...	...	...	...	...	...	...	
6	Ida	Ida	Ida	Bertha	Ida	Clara	Ida	Bertha	Be
7	Alice	Annie	Alice	Ida	Clara	Bertha	Bertha	Clara	Flore
8	Bertha	Bertha	Bertha	Annie	Bertha	Ida	Clara	Florence	E
9	Sarah	Alice	Annie	Clara	Annie	Annie	Florence	Ida	Be

10 rows × 143 columns

In [41]:
```
# Function to create plot of all top 10 names by sex

def plotname(sex, name):
    data = all_years.query('sex == @sex and name == @name')

    plt.plot(data.year, data.number, label = name)
    plt.axis(xmin = 1880, xmax = 2023)
```

In [42]:
```
plt.figure(figsize=(12,5.5))

for name in getyear('F', 2022):
    plotname('F', name)

plt.legend()
```

Out[42]:
```
<matplotlib.legend.Legend at 0x1df35e7c4c0>
```

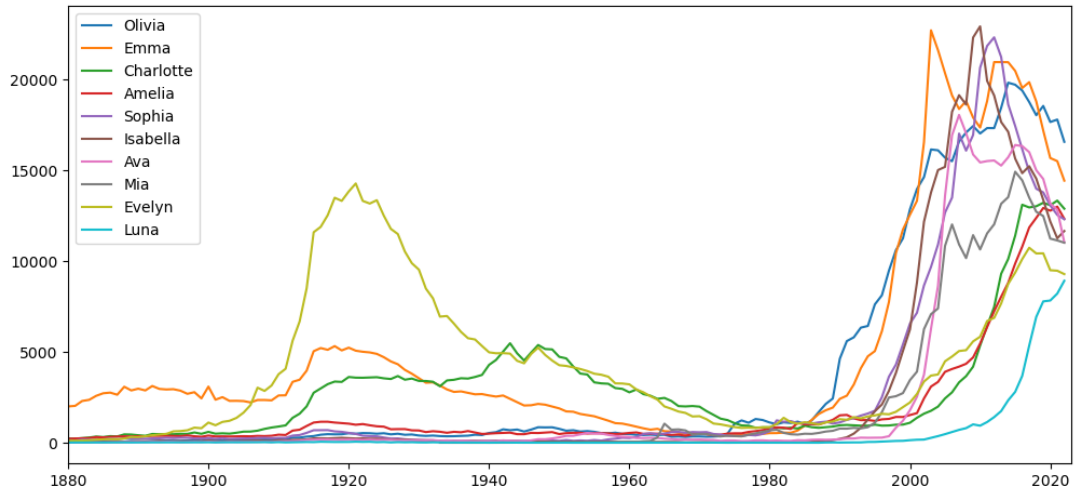


In [43]:
```
# Converting the object into list

list(getyear('F', 2022))
```

Out[43]:
```
['Olivia',
 'Emma',
 'Charlotte',
 'Amelia',
 'Sophia',
 'Isabella',
 'Ava',
 'Mia',
 'Evelyn',
 'Luna']
```

In [44]:
```
# Plotting the same graph by passing the list of top 10 names

plt.figure(figsize = (12,5.5))

for name in ['Olivia', 'Emma', 'Charlotte', 'Amelia', 'Sophia', 'Isabel
    plotname('F', name)

plt.legend()
```

Out[44]:
```
<matplotlib.legend.Legend at 0x1df35e7d600>
```



In [45]:
```
# Get all time favourite names of gender 'F'

alltime_fav_f = all_years_byyears.loc[('F')].groupby('name').sum().sort
```

In [46]:
```
alltime_fav_f
```

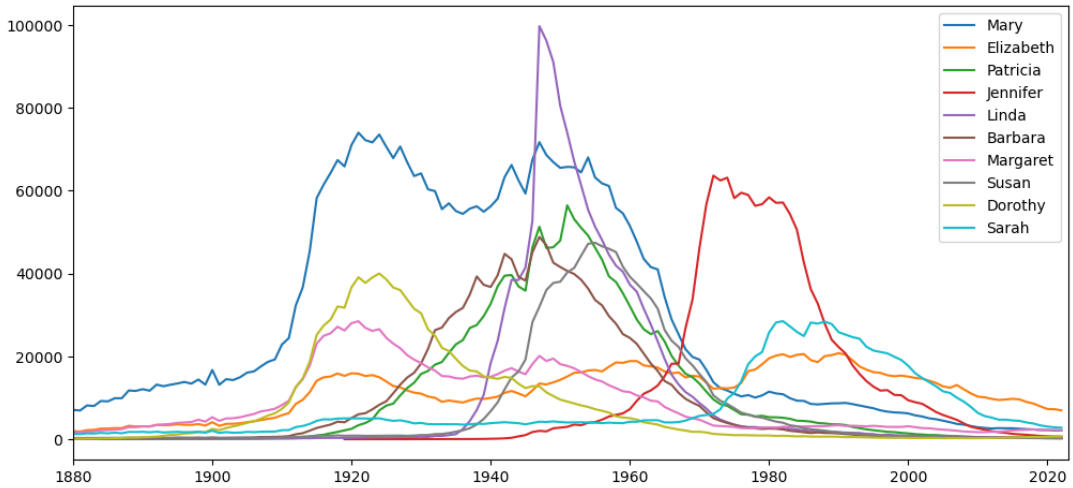
Out[46]:

name	number
Mary	4134713
Elizabeth	1668146
Patricia	1573024
Jennifer	1470012
...	...
Margaret	1257878
Susan	1122752
Dorothy	1110081
Sarah	1090100

10 rows × 1 columns

In [47]:
```
# Plotting alltime_fav top 10 for gender 'F'

plt.figure(figsize = (12,5.5))

for name in alltime_fav_f.index:
    plotname('F', name)

plt.legend()
```

Out[47]: <matplotlib.legend.Legend at 0x1df367d06d0>



In [48]:
```
# Get all time favourite names of gender 'M'

alltime_fav_m = all_years_byyears.loc[('M')].groupby('name').sum().sort
```

In [49]:
```
alltime_fav_m
```

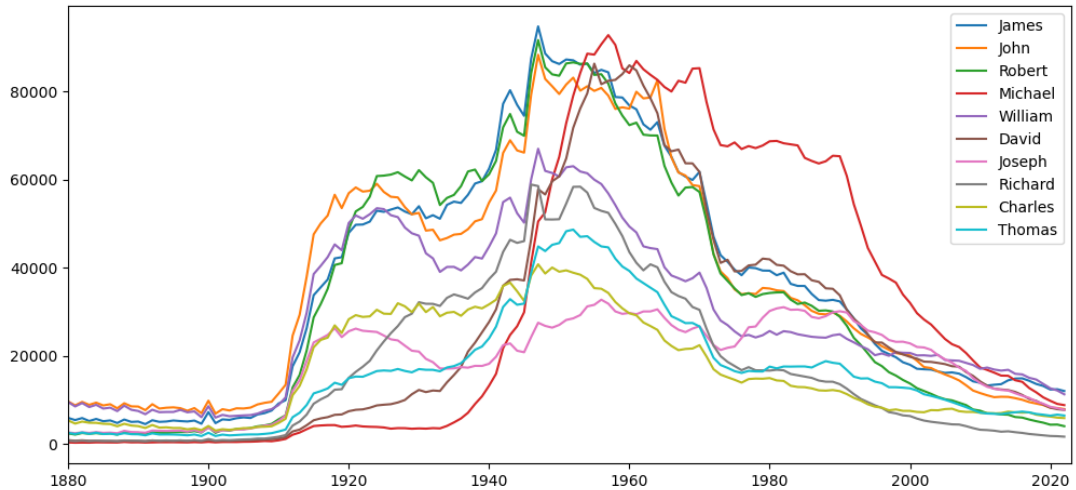
Out[49]:

name	number
James	5214844
John	5158428
Robert	4838129
Michael	4401604
...	...
Joseph	2647283
Richard	2572740
Charles	2417569
Thomas	2338310

10 rows × 1 columns

In [50]:
```
# Plotting alltime_fav top 10 for gender 'M'

plt.figure(figsize = (12,5.5))

for name in alltime_fav_m.index:
    plotname('M', name)

plt.legend()
```

Out[50]:
```
<matplotlib.legend.Legend at 0x1df3595abf0>
```



In [51]:
```
# Total number across all years, group by sex and name

totals = all_years.groupby(['sex', 'name']).number.sum()
```

In [52]:
```
totals
```

Out[52]:
```
sex  name
F    Aabha         56
     Aabidah        5
     Aabriella     51
     Aada          13
                   ..
M    Zyvion         5
     Zyvon          7
     Zyyon          6
     Zzyzx         10
Name: number, Length: 113882, dtype: int64
```

In [53]:
```
# Split into male and female totals

male, female = totals.loc['M'], totals.loc['F']
```

In [54]:
```
ratios = (totals.loc['M'] / totals.loc['F']).dropna()
```

In [55]:
```
ratios
```

Out[55]:
```
name
Aaden      1008.000000
Aadi         64.937500
Aadyn        36.437500
Aalijah       1.838926
                ...
Zyon          5.145138
Zyonn        16.400000
Zyree         8.125000
Zyrie         1.923077
Name: number, Length: 11433, dtype: float64
```

In [56]:
```
# Names that fit unisex ratio constraint

unisex = ratios[(ratios > 0.5) & (ratios < 2)].index
```

In [57]:
```
unisex
```

Out[57]:
```
Index(['Aalijah', 'Aamari', 'Aari', 'Aaris', 'Aaryn', 'Aavyn', 'Abey',
'Abrar',
       'Abriel', 'Adair',
       ...
       'Zi', 'Zihan', 'Zikora', 'Zixuan', 'Ziyan', 'Zohar', 'Zyarie',
'Zyian',
       'Zyn', 'Zyrie'],
      dtype='object', name='name', length=1731)
```

In [58]:
```
common = (male.loc[unisex] + female.loc[unisex]).sort_values(ascending
```

In [59]:
```
common
```

Out[59]:
```
name
Jessie     279884
Riley      228226
Casey      190815
Jackie     169743
            ...
Kendall     98466
Kerry       98455
Jody        87210
Quinn       79293
Name: number, Length: 10, dtype: int64
```

In [60]:
```
all_years_indexed = all_years.set_index(['sex', 'name', 'year']).sort_i
```

In [61]:
```
all_years_indexed
```

Out[61]:

			number
sex	**name**	**year**	
F	**Aabha**	**2011**	7
		2012	5
		2014	9
		2015	7
...	**...**	**...**	...
M	**Zyvon**	**2015**	7
	Zyyon	**2014**	6
	Zzyzx	**2010**	5
		2018	5

2085158 rows × 1 columns

In [62]:

```
# Plotting the top 10 unisex names

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

for i, name in enumerate(common.index):
    plt.subplot(5, 2, i+1)

    plt.plot(all_years_indexed.loc['M', name], label = 'M')
    plt.plot(all_years_indexed.loc['F', name], label = 'F')

    plt.legend()
    plt.title(name)

plt.tight_layout()
```