Question 2.Load HollywoodsMostProfitableStories.csv

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# plotly
# import plotly.plotly as py
from plotly.offline import init_notebook_mode, iplot, plot
import plotly as py
init_notebook_mode(connected=True)
import plotly.graph_objs as go


db = pd.read_csv(r'/Users/shilp/Downloads/data_viz_project2/Hollywoods
```

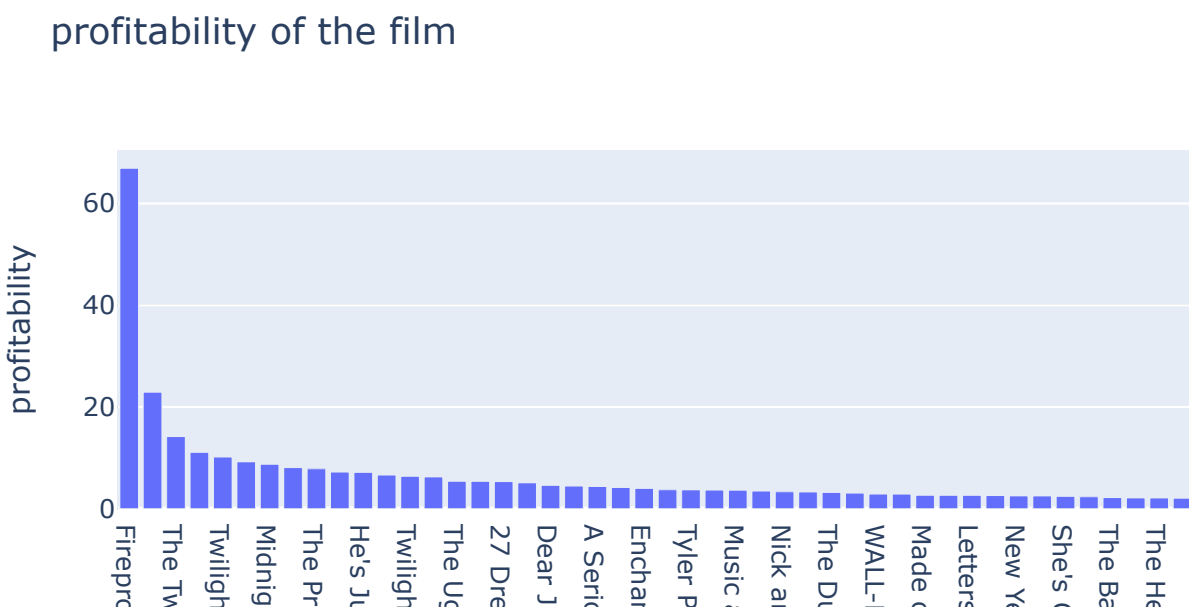
In [2]:
```
db.head(5)
```

Out[2]:

	Film	Genre	Lead Studio	Audience score %	Profitability	Rotten Tomatoes %	Worldwide Gross	Year
0	27 Dresses	Comedy	Fox	71.0	5.343622	40.0	160.308654	2008
1	(500) Days of Summer	Comedy	Fox	81.0	8.096000	87.0	60.720000	2009
2	A Dangerous Method	Drama	Independent	89.0	0.448645	79.0	8.972895	2011
3	A Serious Man	Drama	Universal	64.0	4.382857	89.0	30.680000	2009
4	Across the Universe	Romance	Independent	84.0	0.652603	54.0	29.367143	2007

A. A bar chart showing the profitability of the film. The X-axis is the film. The Y-axis is profitability. The bars should be sorted from the most to the least profitable

In [3]:
```
db.sort_values('Profitability', ascending=False, inplace=True)

trace1 = go.Bar(
                x = db.Film,
                y = db.Profitability,
                name = "A bar chart showing the profitability of the f
)
data = [trace1]
layout = go.Layout(barmode = "stack",
                   title="profitability of the film",
    xaxis_title="film",
    yaxis_title="profitability",)
fig = go.Figure(data = data, layout = layout)
iplot(fig)
```



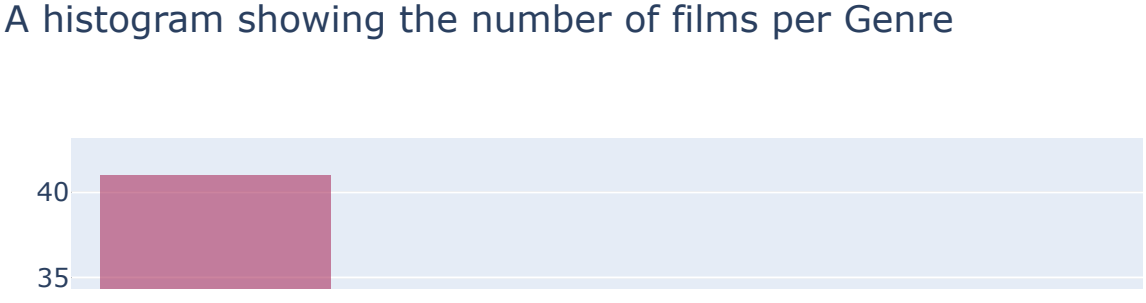
B. A histogram showing the number of films for each Genre. The X-axis is the Genre. The Y-axis is the number of films in the spreadsheet from each Genre.

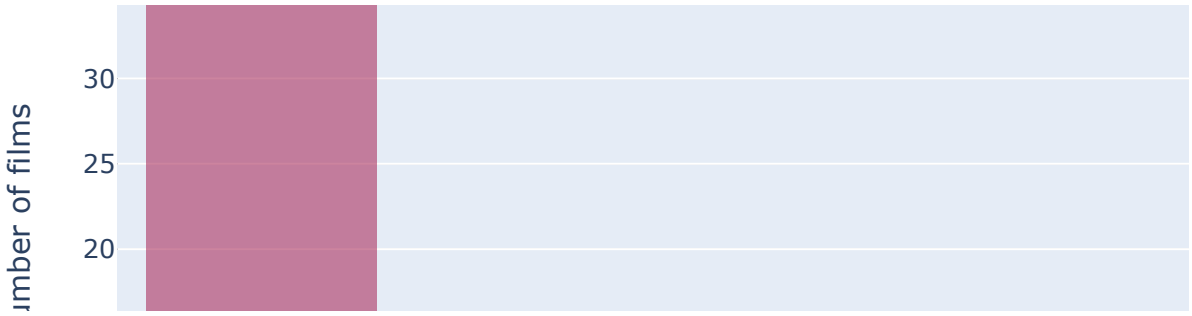
In [4]:
```
df= db = pd.read_csv(r'/Users/shilp/Downloads/data_viz_project2/Hollyv
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_').
df.head(5)
```

Out[4]:

	film	genre	lead_studio	audience__score_%	profitability	rotten_tomatoes_%	worl
0	27 Dresses	Comedy	Fox	71.0	5.343622	40.0	
1	(500) Days of Summer	Comedy	Fox	81.0	8.096000	87.0	
2	A Dangerous Method	Drama	Independent	89.0	0.448645	79.0	
3	A Serious Man	Drama	Universal	64.0	4.382857	89.0	
4	Across the Universe	Romance	Independent	84.0	0.652603	54.0	

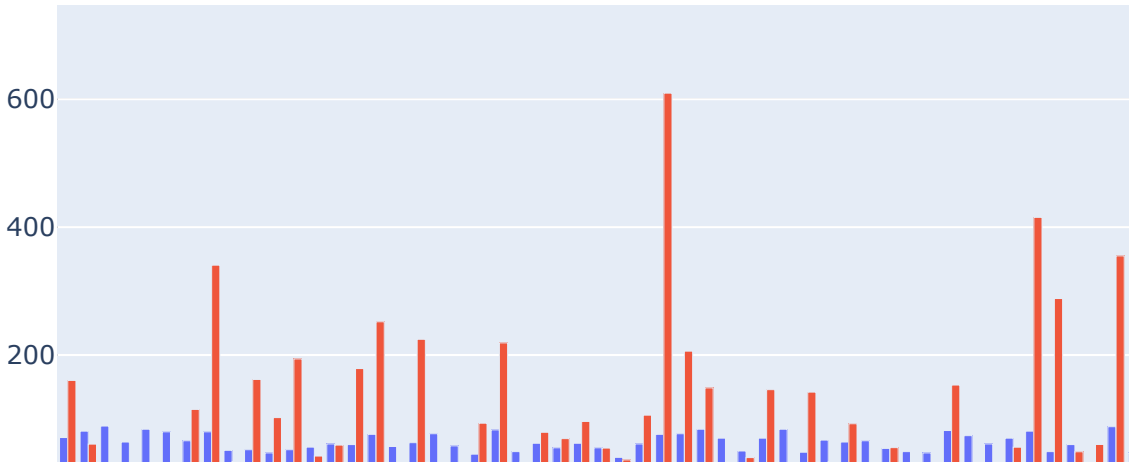
In [5]:
```
trace1 = go.Histogram(
    x=df['genre'],
    opacity=1,
    name = "Genre",
    marker=dict(color='rgba(171, 50, 96, 0.6)')
)


data = [trace1]
layout = go.Layout(barmode='stack',
                   title='A histogram showing the number of films per
                   xaxis=dict(title='Genre'),
                   yaxis=dict( title='number of films'),

)
fig = go.Figure(data=data, layout=layout)
iplot(fig)
```





c. A figure with two bar plots: Worldwide Gross and Audience. The X-axis is the film index. When the mouse cursor hovers over a bar, the film's title should be displayed in the tooltip.

In [6]:
```
trace1 = go.Bar(
                x = db.film,
                y = db['audience__score_%'],
                name = "audience score %",
)
trace2 = go.Bar(
                x = db.film,
                y = db.worldwide_gross,
                name = "worldwide gross ")
fig = go.Figure()
fig.add_trace(trace1)
fig.add_trace(trace2)
fig.show()
```



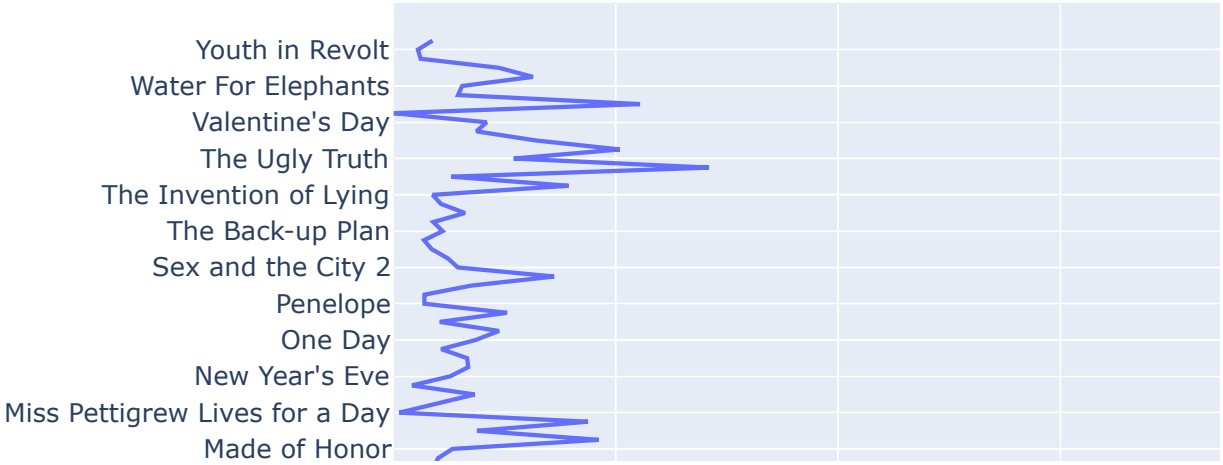
d. A line chart showing the profitability of the films over the years. The X-axis is the Year. The Y-axis is the average profitability.

In [7]:
```
db.isnull().sum()
db['profitability'].mean()
```

Out[7]: 4.741609796577462

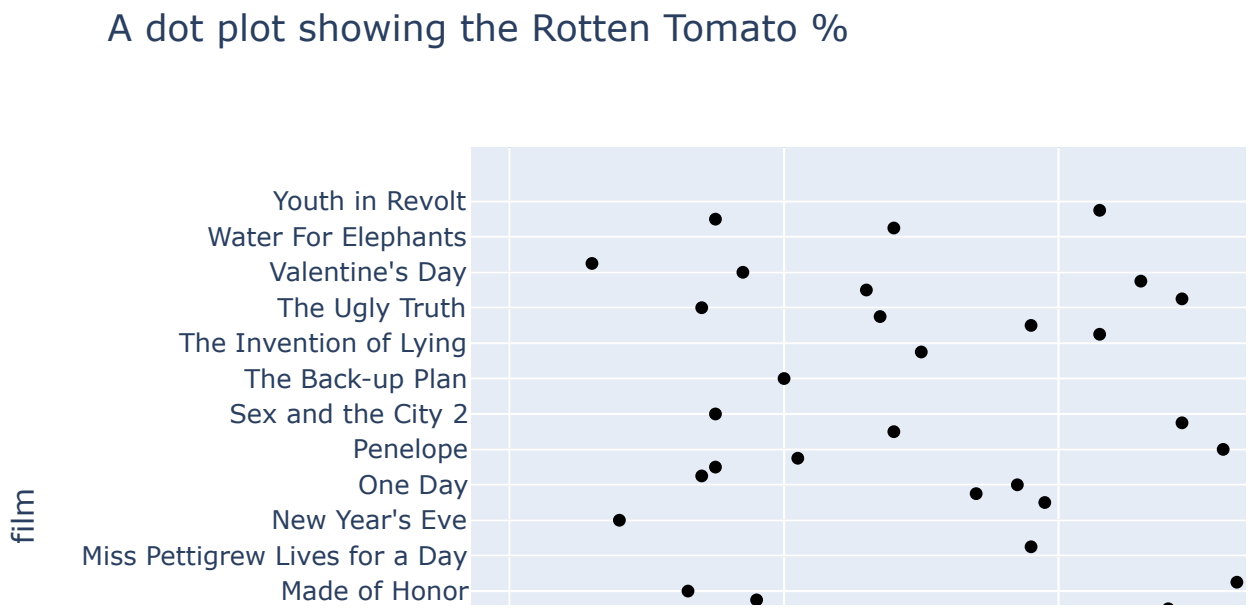
In [8]:
```
df['profitability'] = df['profitability'].fillna(df['profitability'].m
```

In [9]:
```
trace2 = go.Scatter(
                    x = db['profitability'],
                    y = db['film'],
                    mode = "lines",
                    name = "price2007",
                   )
data = [trace2]
layout = dict(title = 'A line chart showing the profitability of the f
              xaxis= dict(title= 'Year',ticklen= 5,zeroline= False)
             )
fig = dict(data = data, layout = layout)
iplot(fig)
```



e. A dot plot showing the Rotten Tomato %. The X-axis is the Rotten Tomato %. The Y-axis is the film title.

In [10]:
```
trace1 =go.Scatter(
                    x = df["rotten_tomatoes_%"],
                    y = df["film"],
                    mode = "markers",
                    marker = dict(color ="black" ),
)
data = [trace1]
layout = dict(title = 'A dot plot showing the Rotten Tomato %',
              xaxis= dict(title= 'World Rank',ticklen= 5,zeroline= Fal
              yaxis= dict(title= 'film',ticklen= 5,zeroline= False)
             )
fig = dict(data = data, layout = layout)
iplot(fig)
```



In [ ]:

3. Load Housing_price.csv and use Plotly to create the following charts. Every figure must include a title. Each axis must be labeled.

In [11]:
```
db = pd.read_csv(r'/Users/shilp/Downloads/data_viz_project2/Housing_pr
```

In [12]:
```
db.head(5)
```

Out[12]:

	housenum	acre	acregroup	adj1998	adj2007	adj2011	bedgroup	bedrooms	bikescore
0	1	0.28	> 1/4 acre	148.3626	233.8419	191.8211	3 beds	3	35
1	2	0.29	> 1/4 acre	135.2073	261.4203	206.9677	3 beds	3	44
2	3	0.36	> 1/4 acre	256.5284	401.0359	347.9472	3 beds	3	66
3	4	0.26	> 1/4 acre	231.6795	305.0861	257.4915	3 beds	3	61
4	5	0.31	> 1/4 acre	271.8762	298.7660	236.6253	4+ beds	4	53

5 rows × 31 columns

a. A figure that contains three boxplots: price2014, squarefeet, and acre. i. The values in “square feet” and “acre” columns are too small compared with the price column. To make the chart look better, you may multiply “squarefeet” by 1000. (The data in the squarefeet column was divided by 1000.) You may convert the values in the “acre” column to squareyard by multiplying 4840.

In [13]:
```
db['squarefeet'] = db['squarefeet']*1000
db['acre']= db['acre']*4840
```

In [14]:
```
db.head(5)
```

Out[14]:

	housenum	acre	acregroup	adj1998	adj2007	adj2011	bedgroup	bedrooms	bikescor
0	1	1355.2	> 1/4 acre	148.3626	233.8419	191.8211	3 beds	3	3
1	2	1403.6	> 1/4 acre	135.2073	261.4203	206.9677	3 beds	3	4
2	3	1742.4	> 1/4 acre	256.5284	401.0359	347.9472	3 beds	3	6
3	4	1258.4	> 1/4 acre	231.6795	305.0861	257.4915	3 beds	3	6
4	5	1500.4	> 1/4 acre	271.8762	298.7660	236.6253	4+ beds	4	5

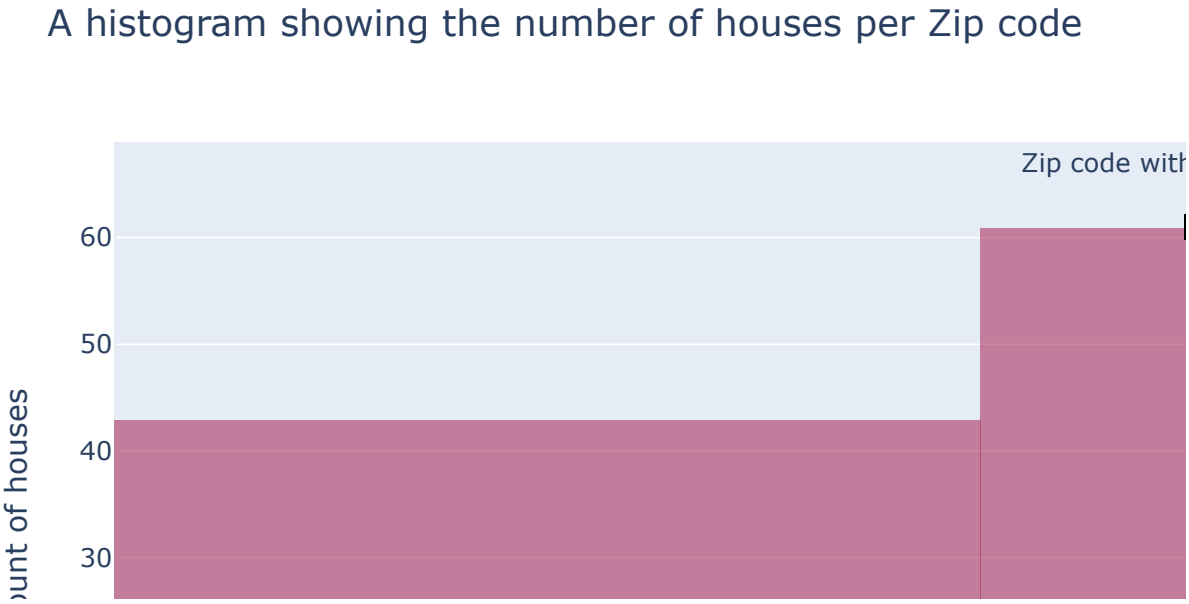
5 rows × 31 columns

In [15]:
```
trace0 = go.Box(
    y= db['acre'],
    name = 'acre',
    marker = dict(
        color = 'rgb(12, 12, 140)',
    )
)
trace1 = go.Box(
    y= db['price2014'],
    name = 'price2014',
    marker = dict(
        color = 'rgb(12, 128, 128)',
    )
)
trace2 = go.Box(
    y= db['squarefeet'],
    name = 'squarefeet',
    marker = dict(

    )
)
data = [trace0, trace1, trace2]
layout = dict(title = 'three boxplots: price2014, squarefeet, and acre

              )
fig = dict(data = data, layout = layout)
iplot(data)
```

b. A histogram showing the number of houses per Zip code. The X-axis is the Zip code. Create an annotation pointing to the Zip code with the most houses.

In [16]:
```
trace1 = go.Histogram(
    x=db['zip'],
    opacity=1,
    name = "Zip Code",
    marker=dict(color='rgba(171, 50, 96, 0.6)')
)


data = [trace1]
layout = go.Layout(barmode='overlay',
                   title='A histogram showing the number of houses per
                   xaxis=dict(title='Zip codes'),
                   yaxis=dict( title='Count of houses'),
                       annotations=[
        dict(
            x=1062,
```

```
            y=61,
            xref="x",
            yref="y",
            text="Zip code with the most houses",
            showarrow=True,
            arrowhead=7,
            ax=10,
            ay=-30,
            arrowcolor="black",
            arrowsize=3,
            arrowwidth=1,

        )
    ]
)
fig = go.Figure(data=data, layout=layout)
iplot(fig)
```



c. A line chart with four lines: price1998, price2007, price2011, and price2014. The X-axis is the house num. The Y-axis is the value from the four columns listed above. When the mouse cursor hovers over a marker, the street address of the house should be displayed in the tooltip.

In [17]:
```
trace1 = go.Scatter(
                    x = db['housenum'],
                    y = db['price1998'],
                    mode = "lines",
                    name = "price1998",

    text= db.streetname,

)
trace2 = go.Scatter(
                    x = db['housenum'],
                    y = db['price2007'],
                    mode = "lines",
                    name = "price2007",

                    text= db.streetname,
                   )

trace3 = go.Scatter(
                    x = db['housenum'],
                    y = db['price2011'],
                    mode = "lines",
                    name = "price2011",

                    text= db.streetname,
                   )

trace4 = go.Scatter(
                    x = db['housenum'],
                    y = db['price2014'],
                    mode = "lines",
                    name = "price2014",

                    text= db.streetname,
                   )


data = [trace1, trace2, trace3, trace4]
layout = dict(title = 'A line chart with four lines',
              xaxis= dict(title= 'House num',ticklen= 5,zeroline= Fals
             )
fig = dict(data = data, layout = layout)
iplot(fig)
```

4. Load wimbledons_champions.csv and use Seaborn to create the following charts.

In [18]:
```
df = pd.read_csv(r'/Users/shilp/Downloads/data_viz_project2/wimbledons
```

In [19]:
```
df.head()
```

Out[19]:

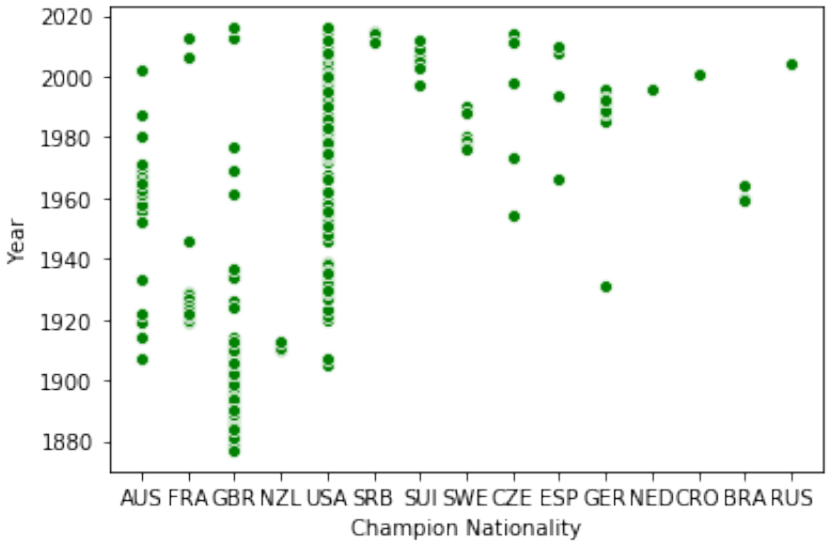
	Gender	Champion	Mins	Runner-up Nationality	Champion Nationality	Runner-up	Score	Runner-up Seed	Champion Seed	Yea
0	Men's	G.L. Patterson	NaN	AUS	AUS	N.E. Brookes	6-3, 7-5, 6-2	NaN	NaN	191
1	Men's	G.L. Patterson	NaN	GBR	AUS	R. Lycett	6-3, 6-4, 6-2	NaN	NaN	192
2	Men's	N.E. Brookes	NaN	GBR	AUS	A.W. Gore	6-4, 6-2, 6-2	NaN	NaN	190
3	Men's	N.E. Brookes	NaN	NZL	AUS	A.F. Wilding	6-4, 6-4, 7-5	NaN	NaN	191
4	Men's	J.R. Borotra	80.0	FRA	FRA	J.R. Lacoste	6-1, 3-6, 6-1, 3-6, 6-4	NaN	NaN	192

In [20]:
```
import seaborn as sns
```

a. A scatterplot showing when a player from different countries won the championship. The X-axis is the country. The Y-axis is the year. Each circle/dot indicates a player from a certain country won the championship in a certain year. The circles should be filled with green color.

In [21]:
```
sns.scatterplot(
    x=df['Champion Nationality'],
    y=df['Year'],
    color='green',
    data=df)
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x1a224f7c50>



b. Create a grid with four cells. In the first row, show two charts: a histogram showing the number of men's champions for different countries and a histogram showing the number of women's champions for different countries. In the second row, show two charts: a histogram showing the number of men's runners-up for different countries and a histogram showing the number of women's runners-up for different countries.

In [22]:
```
import matplotlib.pyplot as plt
df
```

Out[22]:

	Gender	Champion	Mins	Runner-up Nationality	Champion Nationality	Runner-up	Score	Runner-up Seed	Champion Seed
0	Men's	G.L. Patterson	NaN	AUS	AUS	N.E. Brookes	6-3, 7-5, 6-2	NaN	NaN
1	Men's	G.L. Patterson	NaN	GBR	AUS	R. Lycett	6-3, 6-4, 6-2	NaN	NaN
2	Men's	N.E. Brookes	NaN	GBR	AUS	A.W. Gore	6-4, 6-2, 6-2	NaN	NaN
3	Men's	N.E. Brookes	NaN	NZL	AUS	A.F. Wilding	6-4, 6-4, 7-5	NaN	NaN
4	Men's	J.R. Borotra	80.0	FRA	FRA	J.R. Lacoste	6-1, 3-6, 6-1, 3-6, 6-4	NaN	NaN
...	...	...	...	...	...	...	...	...	...
248	Women's	V.E.S. Williams	84.0	NaN	USA	NaN	6-3, 7-6 (7-3)	2	5.0
249	Women's	S.J. Fry	50.0	NaN	USA	NaN	6-3, 6-1,	6	5.0
250	Women's	S. Williams	122.0	NaN	USA	NaN	6-1, 5-7, 6-2	3	6.0
251	Women's	V.E.S. Williams	111.0	NaN	USA	NaN	7-5, 6-4	6	7.0
252	Women's	J.R. Susman	57.0	NaN	USA	NaN	6-4, 6-4,	NaN	8.0

253 rows × 12 columns

In [23]:
```
df['Runner-up Nationality'].fillna(df["Runner-up Nationality (Men's)"]
```

Out[23]:
```
0      AUS
1      GBR
2      GBR
3      NZL
4      FRA
      ...
248    USA
249    GBR
250    POL
251    USA
252    TCH
Name: Runner-up Nationality, Length: 253, dtype: object
```

In [24]:
```
df
```

Out[24]:

	Gender	Champion	Mins	Runner-up Nationality	Champion Nationality	Runner-up	Score	Runner-up Seed	Champion Seed
0	Men's	G.L. Patterson	NaN	AUS	AUS	N.E. Brookes	6-3, 7-5, 6-2	NaN	NaN
1	Men's	G.L. Patterson	NaN	GBR	AUS	R. Lycett	6-3, 6-4, 6-2	NaN	NaN
2	Men's	N.E. Brookes	NaN	GBR	AUS	A.W. Gore	6-4, 6-2, 6-2	NaN	NaN
3	Men's	N.E. Brookes	NaN	NZL	AUS	A.F. Wilding	6-4, 6-4, 7-5	NaN	NaN
4	Men's	J.R. Borotra	80.0	FRA	FRA	J.R. Lacoste	6-1, 3-6, 6-1, 3-6, 6-4	NaN	NaN
...	...	...	...	...	...	...	...	...	...
248	Women's	V.E.S. Williams	84.0	NaN	USA	NaN	6-3, 7-6 (7-3)	2	5.0
249	Women's	S.J. Fry	50.0	NaN	USA	NaN	6-3, 6-1,	6	5.0
250	Women's	S. Williams	122.0	NaN	USA	NaN	6-1, 5-7, 6-2	3	6.0
251	Women's	V.E.S. Williams	111.0	NaN	USA	NaN	7-5, 6-4	6	7.0
252	Women's	J.R. Susman	57.0	NaN	USA	NaN	6-4, 6-4,	NaN	8.0

253 rows × 12 columns

In [25]:
```
df['Runner-up Nationality'].fillna(df["Runner-up Nationality (Men's)"]
```

Out[25]:
```
0      AUS
1      GBR
2      GBR
3      NZL
4      FRA
      ...
248    USA
249    GBR
250    POL
251    USA
252    TCH
Name: Runner-up Nationality, Length: 253, dtype: object
```

In [26]:
```
df['Final Runner-up Nationality']=df['Runner-up Nationality'].mask(pd.
df
```

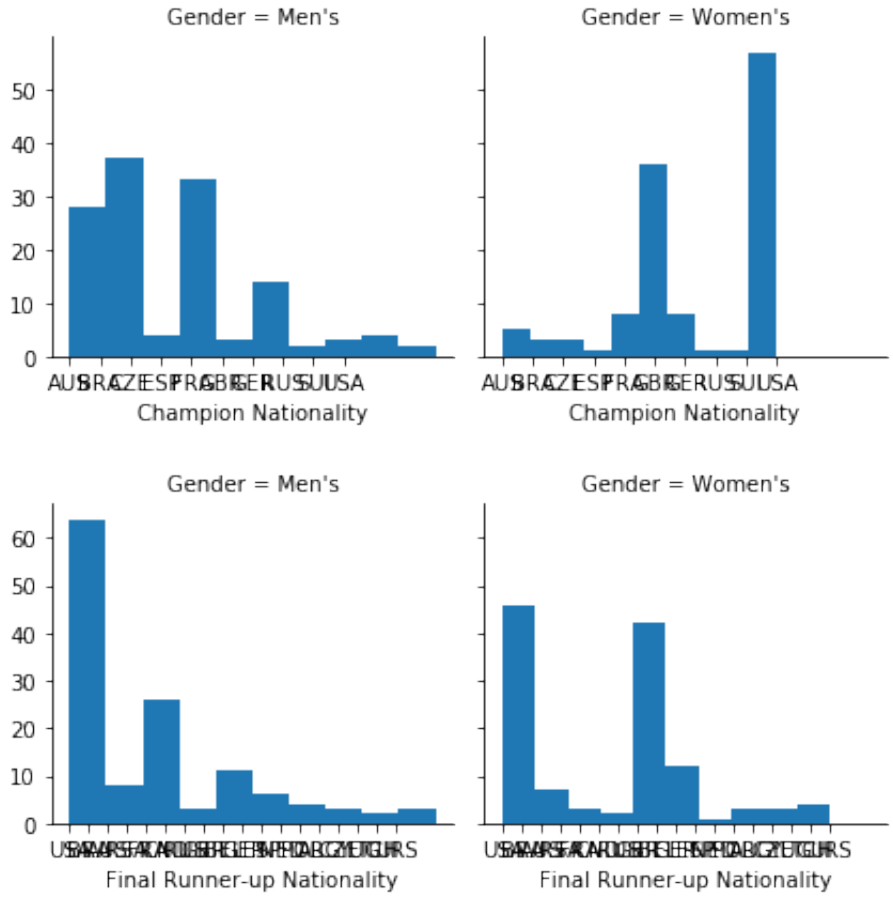
Out[26]:

	Gender	Champion	Mins	Runner-up Nationality	Champion Nationality	Runner-up	Score	Runner-up Seed	Champion Seed
0	Men's	G.L. Patterson	NaN	AUS	AUS	N.E. Brookes	6-3, 7-5, 6-2	NaN	NaN
1	Men's	G.L. Patterson	NaN	GBR	AUS	R. Lycett	6-3, 6-4, 6-2	NaN	NaN
2	Men's	N.E. Brookes	NaN	GBR	AUS	A.W. Gore	6-4, 6-2, 6-2	NaN	NaN
3	Men's	N.E. Brookes	NaN	NZL	AUS	A.F. Wilding	6-4, 6-4, 7-5	NaN	NaN
4	Men's	J.R. Borotra	80.0	FRA	FRA	J.R. Lacoste	6-1, 3-6, 6-1, 3-6, 6-4	NaN	NaN
...	...	...	...	...	...	...	...	...	...
248	Women's	V.E.S. Williams	84.0	NaN	USA	NaN	6-3, 7-6 (7-3)	2	5.0
249	Women's	S.J. Fry	50.0	NaN	USA	NaN	6-3, 6-1,	6	5.0
250	Women's	S. Williams	122.0	NaN	USA	NaN	6-1, 5-7, 6-2	3	6.0
251	Women's	V.E.S. Williams	111.0	NaN	USA	NaN	7-5, 6-4	6	7.0
252	Women's	J.R. Susman	57.0	NaN	USA	NaN	6-4, 6-4,	NaN	8.0

253 rows × 13 columns

In [27]:
```
fig = plt.gcf()
fig.set_size_inches(140, 120)
g = sns.FacetGrid(df, col="Gender", margin_titles=True)
g.map(plt.hist, "Champion Nationality");
g = sns.FacetGrid(df, col="Gender",  margin_titles=True)
g.map(plt.hist, "Final Runner-up Nationality",);
```

```
<Figure size 10080x8640 with 0 Axes>
```



In [28]:
```
conda install -c conda-forge pandoc
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

In [ ]:
```

```