Mini_Project_Data_Wrangling_Pandas

November 1, 2020

1 Mini-Project: Data Wrangling and Transformation with Pandas

Working with tabular data is a necessity for anyone with enterprises having a majority of their data in relational databases and flat files. This mini-project is adopted from the excellent tutorial on pandas by Brandon Rhodes which you have watched earlier in the Data Wrangling Unit. In this mini-project, we will be looking at some interesting data based on movie data from the IMDB.

This assignment should help you reinforce the concepts you learnt in the curriculum for Data Wrangling and sharpen your skills in using Pandas. Good Luck!

1.0.1 Please make sure you have one of the more recent versions of Pandas

[4]:
```
import pandas as pd
import matplotlib.pyplot as plt

%matplotlib inline
```

[5]:
```
pd.__version__
```

[5]:
```
'1.0.1'
```

1.1 Taking a look at the Movies dataset

This data shows the movies based on their title and the year of release

[6]:
```
movies = pd.read_csv('titles.csv')
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244914 entries, 0 to 244913
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   title   244914 non-null  object
 1   year    244914 non-null  int64 
```

```
dtypes: int64(1), object(1)
memory usage: 3.7+ MB
```

[7]:
```
movies.head()
```

[7]:
```
                                        title  year
0                       The Ticket to the Life  2009
1  Parallel Worlds: A New Rock Music Experience  2016
2                     Morita - La hija de Jesus  2008
3                                         Gun  2017
4                        Love or Nothing at All  2014
```

1.2 Taking a look at the Cast dataset

This data shows the cast (actors, actresses, supporting roles) for each movie

- The attribute `n` basically tells the importance of the cast role, lower the number, more important the role.
- Supporting cast usually don't have any value for `n`

[8]:
```
cast = pd.read_csv('cast.csv')
cast.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3786176 entries, 0 to 3786175
Data columns (total 6 columns):
 #   Column     Dtype
---  ------     -----
 0   title      object
 1   year       int64
 2   name       object
 3   type       object
 4   character  object
 5   n          float64
dtypes: float64(1), int64(1), object(4)
memory usage: 173.3+ MB
```

[9]:
```
cast.head(10)
```

[9]:
```
                                    title  year  \
0                           Closet Monster  2015
1                           Suuri illusioni  1985
2                        Battle of the Sexes  2017
3                      Secret in Their Eyes  2015
4                                Steve Jobs  2015
5                    Straight Outta Compton  2015
6                    Straight Outta Compton  2015
```

```
7                                    For Thy Love 2  2009
8  Lapis, Ballpen at Diploma, a True to Life Journey  2014
9                                      Desire (III)  2014

                        name   type                                  character  \
0                   Buffy #1  actor                                    Buffy 4
1                    Homo $   actor                                     Guests
2                    $hutter  actor                           Bobby Riggs Fan
3                    $hutter  actor                           2002 Dodger Fan
4                    $hutter  actor                  1988 Opera House Patron
5                    $hutter  actor                               Club Patron
6                    $hutter  actor                                   Dopeman
7                Bee Moe $lim  actor                                    Thug 1
8  Jori ' Danilo' Jurado Jr.  actor                             Jaime (young)
9            Syaiful 'Ariffin  actor  Actor Playing Eteocles from 'Antigone'

      n
0  31.0
1  22.0
2  10.0
3   NaN
4   NaN
5   NaN
6   NaN
7   NaN
8   9.0
9   NaN
```

1.3 Taking a look at the Release dataset

This data shows details of when each movie was release in each country with the release date

[10]:
```
release_dates = pd.read_csv('release_dates.csv', parse_dates=['date'],␣
↪infer_datetime_format=True)
release_dates.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 479488 entries, 0 to 479487
Data columns (total 4 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   title    479488 non-null  object
 1   year     479488 non-null  int64
 2   country  479488 non-null  object
 3   date     479488 non-null  datetime64[ns]
dtypes: datetime64[ns](1), int64(1), object(2)
```

```
memory usage: 14.6+ MB
```

[11]:
```
release_dates.head()
```

[11]:

	title	year	country	date
0	#73, Shaanthi Nivaasa	2007	India	2007-06-15
1	#BKKY	2016	Cambodia	2017-10-12
2	#Beings	2015	Romania	2015-01-29
3	#Captured	2017	USA	2017-09-05
4	#Ewankosau saranghaeyo	2015	Philippines	2015-01-21

2 Section I - Basic Querying, Filtering and Transformations

2.0.1 What is the total number of movies?

[9]:
```
len(movies)
```

[9]: 244914

2.0.2 List all Batman movies ever made

[10]:
```
batman_df = movies[movies.title == 'Batman']
print('Total Batman Movies:', len(batman_df))
batman_df
```

```
Total Batman Movies: 2
```

[10]:

	title	year
52734	Batman	1943
150621	Batman	1989

2.0.3 List all Batman movies ever made - the right approach

[11]:
```
batman_df = movies[movies.title.str.contains('Batman', case=False)]
print('Total Batman Movies:', len(batman_df))
batman_df.head(10)
```

```
Total Batman Movies: 35
```

[11]:

	title	year
16813	Batman: Anarchy	2016
30236	Batman Forever	1995
31674	Batman Untold	2010

```
31711  Scooby-Doo & Batman: the Brave and the Bold  2018
41881                   Batman the Rise of Red Hood  2018
43484        Batman: Return of the Caped Crusaders  2016
46333                                 Batman & Robin  1997
51811                               Batman Revealed  2012
52734                                        Batman  1943
56029                 Batman Beyond: Rising Knight  2014
```

2.0.4 Display the top 15 Batman movies in the order they were released

[12]:
```
batman_df.sort_values(by=['year'], ascending=True).iloc[:15]
```

[12]:
```
                             title  year
52734                       Batman  1943
100056             Batman and Robin  1949
161439               Batman Dracula  1964
84327          Alyas Batman at Robin  1965
68364                  James Batman  1966
161527             Batman: The Movie  1966
56159          Batman Fights Dracula  1967
168504         Fight! Batman, Fight!  1973
150621                      Batman  1989
156239         Alyas Batman en Robin  1991
156755                Batman Returns  1992
63366   Batman: Mask of the Phantasm  1993
30236                 Batman Forever  1995
46333                 Batman & Robin  1997
208220                 Batman Begins  2005
```

2.0.5 Section I - Q1 : List all the 'Harry Potter' movies from the most recent to the earliest

[13]:
```
harrypotter_df = movies[movies.title.str.contains('harry potter', case = False)]
harrypotter_sorted_df = harrypotter_df.sort_values(by=['year'], ascending=False)
harrypotter_sorted_df
```

[13]:
```
                                          title  year
143147  Harry Potter and the Deathly Hallows: Part 2  2011
152831  Harry Potter and the Deathly Hallows: Part 1  2010
109213       Harry Potter and the Half-Blood Prince  2009
50581      Harry Potter and the Order of the Phoenix  2007
187926           Harry Potter and the Goblet of Fire  2005
61957       Harry Potter and the Prisoner of Azkaban  2004
82791        Harry Potter and the Chamber of Secrets  2002
223087         Harry Potter and the Sorcerer's Stone  2001
```

2.0.6 How many movies were made in the year 2017?

[14]:
```
len(movies[movies.year == 2017])
```

[14]: 11474

2.0.7 Section I - Q2 : How many movies were made in the year 2015?

[15]:
```
len(movies[movies.year == 2015])
```

[15]: 8702

2.0.8 Section I - Q3 : How many movies were made from 2000 till 2018?

- You can chain multiple conditions using OR (|) as well as AND (&) depending on the condition

[16]:
```
len(movies[(movies.year >= 2000) & (movies.year < 2018)])
```

[16]: 106029

2.0.9 Section I - Q4: How many movies are titled "Hamlet"?

[17]:
```
len(movies[movies.title == 'Hamlet'])
```

[17]: 20

2.0.10 Section I - Q5: List all movies titled "Hamlet"

- The movies should only have been released on or after the year 2000
- Display the movies based on the year they were released (earliest to most recent)

[18]:
```
hamlet_df = movies[(movies.title == 'Hamlet') & (movies.year >= 2000)]
hamlet_sorted_df = hamlet_df.sort_values(by = ['year'], ascending = True)
hamlet_sorted_df
```

[18]:

	title	year
55639	Hamlet	2000
1931	Hamlet	2009
227953	Hamlet	2011
178290	Hamlet	2014
186137	Hamlet	2015
191940	Hamlet	2016
244747	Hamlet	2017

2.0.11 Section I - Q6: How many roles in the movie "Inception" are of the supporting cast (extra credits)

- supporting cast are NOT ranked by an "n" value (NaN)
- check for how to filter based on nulls

[19]:
```
len(cast[(cast.title == 'Inception') & (cast['n'].isnull() == True)])
```

[19]: 27

2.0.12 Section I - Q7: How many roles in the movie "Inception" are of the main cast

- main cast always have an 'n' value

[20]:
```
len(cast[(cast.title == 'Inception') & (cast['n'].isnull() == False)])
```

[20]: 51

2.0.13 Section I - Q8: Show the top ten cast (actors or actresses) in the movie "Inception"

- main cast always have an 'n' value
- remember to sort!

[21]:
```
cast_top = cast[(cast.title == 'Inception') & (cast['n'].isnull() == False)]
cast_top_sorted = cast_top.sort_values(by = ['n'], ascending = True)
cast_top_sorted[:10]
```

[21]:

	title	year	name	type	character	n
590576	Inception	2010	Leonardo DiCaprio	actor	Cobb	1.0
859993	Inception	2010	Joseph Gordon-Levitt	actor	Arthur	2.0
3387147	Inception	2010	Ellen Page	actress	Ariadne	3.0
940923	Inception	2010	Tom Hardy	actor	Eames	4.0
2406531	Inception	2010	Ken Watanabe	actor	Saito	5.0
1876301	Inception	2010	Dileep Rao	actor	Yusuf	6.0
1615709	Inception	2010	Cillian Murphy	actor	Robert Fischer	7.0
183937	Inception	2010	Tom Berenger	actor	Browning	8.0
2765969	Inception	2010	Marion Cotillard	actress	Mal	9.0
1826027	Inception	2010	Pete Postlethwaite	actor	Maurice Fischer	10.0

2.0.14 Section I - Q9:

(A) List all movies where there was a character 'Albus Dumbledore'

(B) Now modify the above to show only the actors who played the character 'Albus Dumbledore'

- For Part (B) remember the same actor might play the same role in multiple movies

[22]:
```
movies_with_albus = cast[cast.character == 'Albus Dumbledore']
movies_with_albus.title
```

[22]:
```
704984                                    Epic Movie
792421           Harry Potter and the Goblet of Fire
792423     Harry Potter and the Order of the Phoenix
792424      Harry Potter and the Prisoner of Azkaban
947789       Harry Potter and the Chamber of Secrets
947790         Harry Potter and the Sorcerer's Stone
1685537                       Ultimate Hero Project
2248085                                      Potter
Name: title, dtype: object
```

[23]:
```
movies_with_albus.name
```

[23]:
```
704984            Dane Farwell
792421          Michael Gambon
792423          Michael Gambon
792424          Michael Gambon
947789          Richard Harris
947790          Richard Harris
1685537    George (X) O'Connor
2248085      Timothy Tedmanson
Name: name, dtype: object
```

2.0.15 Section I - Q10:

(A) How many roles has 'Keanu Reeves' played throughout his career?

(B) List the leading roles that 'Keanu Reeves' played on or after 1999 in order by year.

[24]:
```
movies_with_keanu = cast[(cast.name == 'Keanu Reeves')]
len(movies_with_keanu)
```

[24]:
```
62
```

[25]:
```
leading_movies_after_1999 = movies_with_keanu[(movies_with_keanu.n == 1) &␣
↪(movies_with_keanu.year >= 1999)]
leading_movies_after_1999_sorted = leading_movies_after_1999.sort_values(by =␣
↪['year'])
leading_movies_after_1999_sorted
```

[25]:
```
                            title  year          name   type  \
1892390                 The Matrix  1999  Keanu Reeves  actor
1892397           The Replacements  2000  Keanu Reeves  actor
1892358                  Hard Ball  2001  Keanu Reeves  actor
1892383              Sweet November  2001  Keanu Reeves  actor
```

```
1892348                    Constantine  2005  Keanu Reeves  actor
1892388                The Lake House  2006  Keanu Reeves  actor
1892382                   Street Kings  2008  Keanu Reeves  actor
1892385  The Day the Earth Stood Still  2008  Keanu Reeves  actor
1892359                   Henry's Crime  2010  Keanu Reeves  actor
1892342                        47 Ronin  2013  Keanu Reeves  actor
1892361                      John Wick  2014  Keanu Reeves  actor
1892366                    Knock Knock  2015  Keanu Reeves  actor
1892399                The Whole Truth  2016  Keanu Reeves  actor
1892362           John Wick: Chapter 2  2017  Keanu Reeves  actor
1892378                         Siberia  2018  Keanu Reeves  actor

                  character    n
1892390                 Neo  1.0
1892397           Shane Falco  1.0
1892358         Conor O'Neill  1.0
1892383           Nelson Moss  1.0
1892348      John Constantine  1.0
1892388            Alex Wyler  1.0
1892382  Detective Tom Ludlow  1.0
1892385               Klaatu  1.0
1892359           Henry Torne  1.0
1892342                   Kai  1.0
1892361             John Wick  1.0
1892366                  Evan  1.0
1892399               Ramsey  1.0
1892362             John Wick  1.0
1892378             Lucas Hill  1.0
```

2.0.16 Section I - Q11:

(A) List the total number of actor and actress roles available from 1950 - 1960

(B) List the total number of actor and actress roles available from 2007 - 2017

[26]:
```
number_of_actor_actress_1950_1960 = cast[((cast.type == 'actor') | (cast.type␣
↪== 'actress')) & ((cast.year >= 1950) & (cast.year <= 1960))]
len(number_of_actor_actress_1950_1960)
```

[26]: 234635

[27]:
```
number_of_actor_actress_2007_2017 = cast[((cast.type == 'actor') | (cast.type␣
↪== 'actress')) & ((cast.year >= 2007) & (cast.year <= 2017))]
len(number_of_actor_actress_2007_2017)
```

[27]: 1452413

2.0.17 Section I - Q12:

(A) List the total number of leading roles available from 2000 to present

(B) List the total number of non-leading roles available from 2000 - present (exclude support cast)

(C) List the total number of support extra-credit roles available from 2000 - present

[28]:
```
lead_roles = cast[(cast.n == 1) & (cast.year >= 2000)]
len(lead_roles)
```

[28]:
```
60568
```

[29]:
```
non_lead_roles = cast[(cast.n > 1) & (cast.n.isnull() == False) & (cast.year >=␣
↪2000)]
len(non_lead_roles)
```

[29]:
```
1001710
```

[30]:
```
support_roles = cast[(cast.n.isnull() == True) & (cast.year >= 2000)]
len(support_roles)
```

[30]:
```
887484
```

3 Section II - Aggregations, Transformations and Visualizations

3.1 What are the top ten most common movie names of all time?

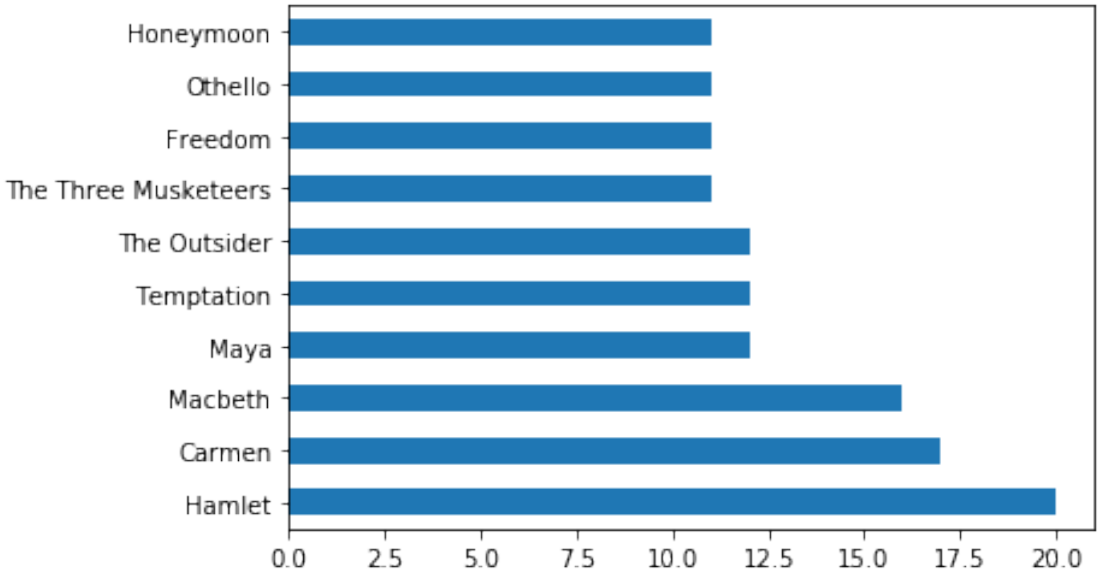
[31]:
```
top_ten = movies.title.value_counts()[:10]
top_ten
```

[31]:
```
Hamlet                  20
Carmen                  17
Macbeth                 16
Maya                    12
Temptation              12
The Outsider            12
The Three Musketeers    11
Freedom                 11
Othello                 11
Honeymoon               11
Name: title, dtype: int64
```

3.1.1 Plot the top ten common movie names of all time

[32]:
```
top_ten.plot(kind='barh')
```

[32]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd1e08d0150>
```



3.1.2 Section II - Q1: Which years in the 2000s saw the most movies released? (Show top 3)

[33]:
```
movies_after_2000 = movies[(movies['year'] >= 2000)]
top_three = movies_after_2000.year.value_counts()[:3]
top_three
```

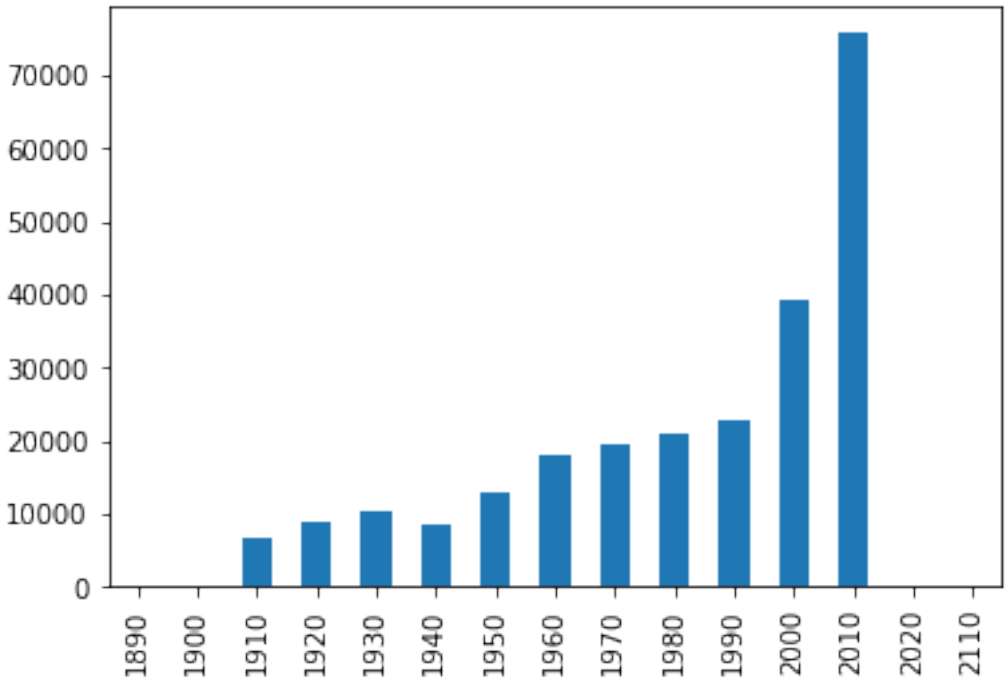
[33]:
```
2017    11474
2016     9440
2015     8702
Name: year, dtype: int64
```

3.1.3 Section II - Q2: # Plot the total number of films released per-decade (1890, 1900, 1910,....)

- Hint: Dividing the year and multiplying with a number might give you the decade the year falls into!
- You might need to sort before plotting

[34]:
```
movies_bucket = movies['year'].astype('int') / 10
movies['decade'] = movies_bucket.astype(int) * 10
movies_bucket_counts = movies.sort_values(by = ['decade'], ascending = True)
movies_bucket_counts = movies_bucket_counts.decade.sort_values()
to_plot = movies_bucket_counts.value_counts().sort_index()
to_plot.plot(kind = 'bar')
```

[34]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd1e2c2db10>
```



3.1.4 Section II - Q3:

(A) What are the top 10 most common character names in movie history?

(B) Who are the top 10 people most often credited as "Herself" in movie history?

(C) Who are the top 10 people most often credited as "Himself" in movie history?

[35]:
```
common_char = cast.character.value_counts()
common_char[:10]
```

[35]:
```
Himself        20746
Dancer         12477
Extra          11948
Reporter        8434
Student         7773
```

```
Doctor          7669
Party Guest     7245
Policeman       7029
Nurse           6999
Bartender       6802
Name: character, dtype: int64
```

[36]:
```
most_herself_cast = cast[(cast.character == 'Herself')]
most_herself_cast.name.value_counts()[:10]
```

[36]:
```
Queen Elizabeth II        12
Joyce Brothers             9
Mar?a Luisa (V) Mart?n     9
Luisa Horga                9
Hillary Clinton            8
Margaret Thatcher          8
Joan Rivers                6
Rekha                      6
Marilyn Monroe             6
Sumie Sakai                6
Name: name, dtype: int64
```

[37]:
```
most_himself_cast = cast[(cast.character == 'Himself')]
most_himself_cast.name.value_counts()[:10]
```

[37]:
```
Adolf Hitler             99
Richard Nixon            44
Ronald Reagan            41
John F. Kennedy          37
George W. Bush           25
Winston Churchill        24
Martin Luther King       23
Bill Clinton             22
Ron Jeremy               22
Franklin D. Roosevelt    21
Name: name, dtype: int64
```

3.1.5 Section II - Q4:

(A) What are the top 10 most frequent roles that start with the word "Zombie"?

(B) What are the top 10 most frequent roles that start with the word "Police"?

- Hint: The `startswith()` function might be useful

[38]:
```
freq_roles_zombie = cast[(cast.character.str.startswith('Zombie'))]
freq_roles_zombie.character.value_counts()[:10]
```

[38]:
```
Zombie                          6264
Zombie Horde                     206
Zombie - Protestor - Victim       78
Zombie Extra                      70
Zombie Dancer                     43
Zombie Girl                       36
Zombie #1                         36
Zombie #2                         31
Zombie Vampire                    25
Zombie Victim                     22
Name: character, dtype: int64
```

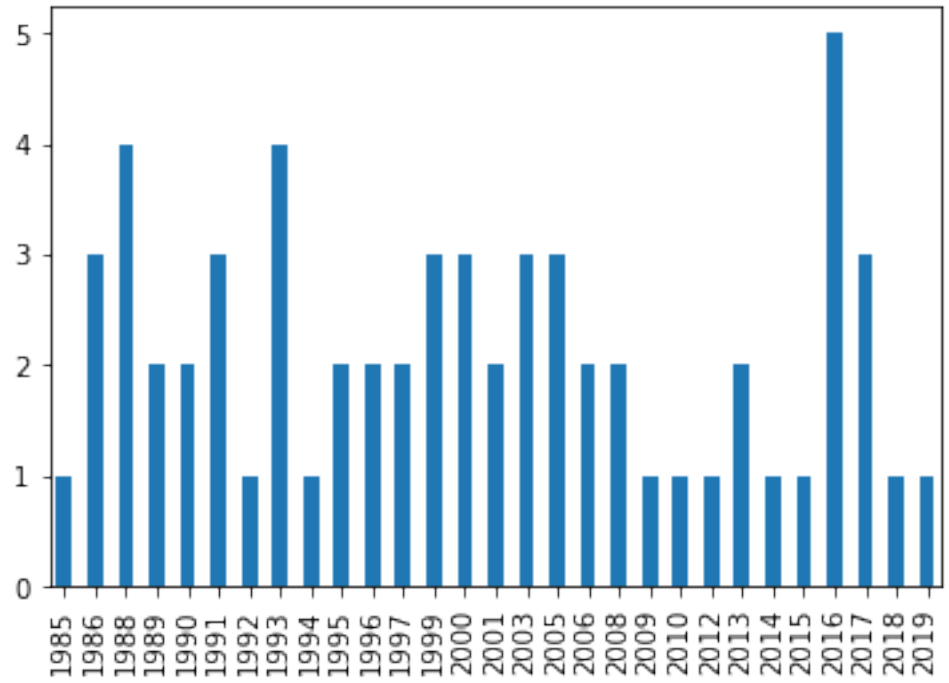
[39]:
```
freq_roles_police = cast[(cast.character.str.startswith('Police'))]
freq_roles_police.character.value_counts()[:10]
```

[39]:
```
Policeman              7029
Police Officer         4808
Police Inspector        742
Police Sergeant         674
Police officer          539
Police                  456
Policewoman             415
Police Chief            410
Police Captain          387
Police Commissioner     337
Name: character, dtype: int64
```

3.1.6 Section II - Q5: Plot how many roles 'Keanu Reeves' has played in each year of his career.

[40]:
```
movies_by_keanu = cast[(cast.name == 'Keanu Reeves')]
movies_by_keanu_per_year = movies_by_keanu.year.value_counts()
movies_by_keanu_per_year_sorted = movies_by_keanu_per_year.sort_index()
movies_by_keanu_per_year_sorted.plot(kind = 'bar')
```

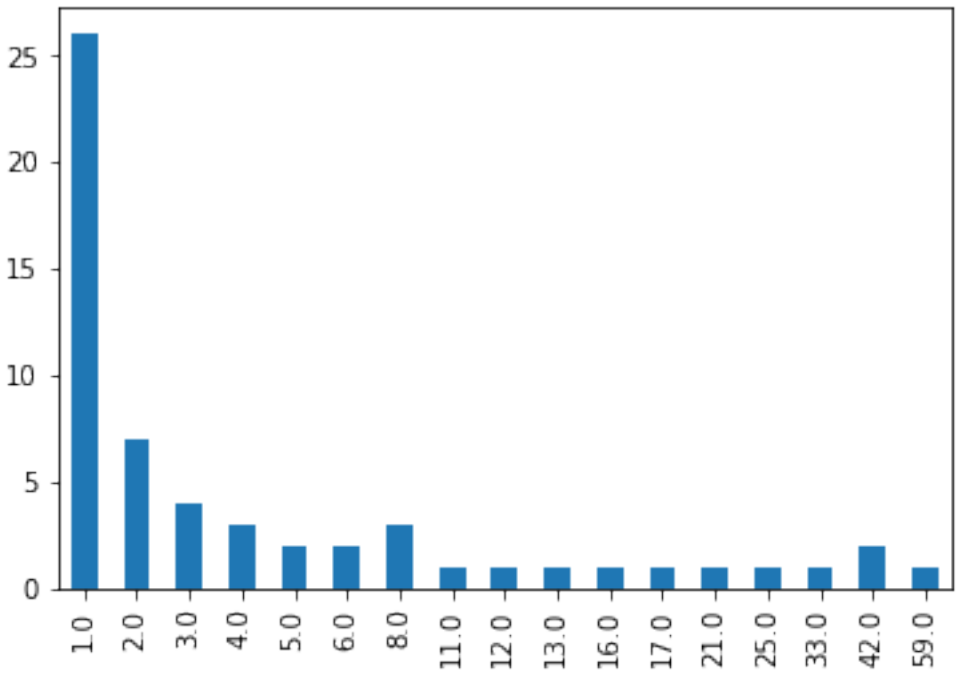
[40]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd1e2be2090>
```



3.1.7 Section II - Q6: Plot the cast positions (n-values) of Keanu Reeve's roles through his career over the years.

[41]:
```
movies_by_keanu_per_n = movies_by_keanu.n.value_counts()
movies_by_keanu_per_n_sorted = movies_by_keanu_per_n.sort_index()
movies_by_keanu_per_n_sorted.plot(kind = 'bar')
```

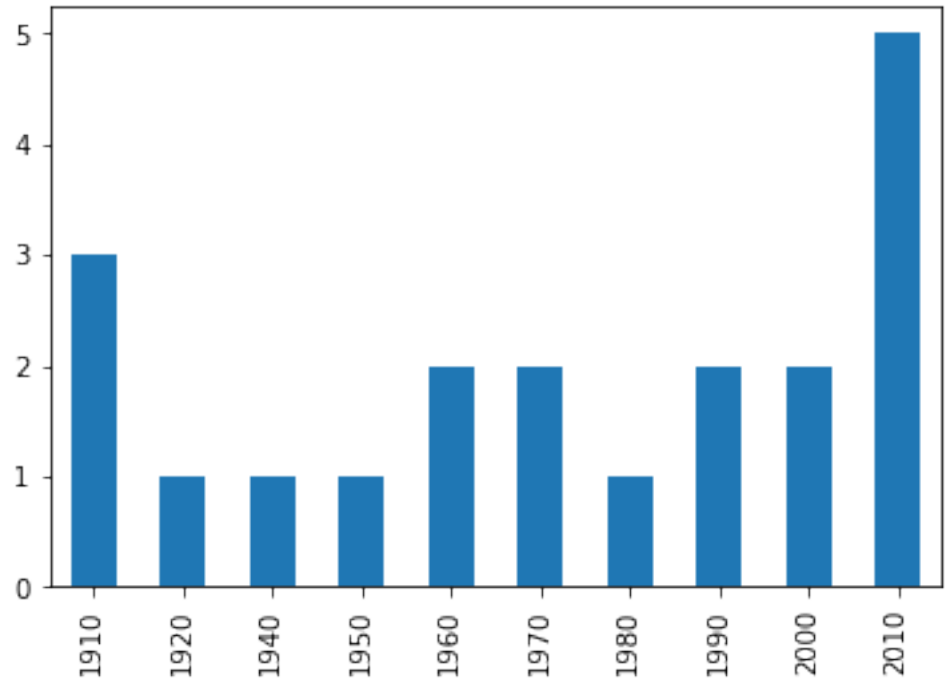
[41]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd1e276c450>



3.1.8 Section II - Q7: Plot the number of "Hamlet" films made by each decade

[42]:
```
movies_bucket = movies['year'].astype('int') / 10
movies['decade'] = movies_bucket.astype(int) * 10
movies_hamlet = movies[movies.title == 'Hamlet']
movies_bucket_counts = movies_hamlet.sort_values(by = ['decade'], ascending =␣
↪True)
movies_bucket_counts = movies_bucket_counts.decade.sort_values()
to_plot = movies_bucket_counts.value_counts().sort_index()
to_plot.plot(kind = 'bar')
```

[42]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd1e0ed60d0>
```



3.1.9 Section II - Q8:

(A) How many leading roles were available to both actors and actresses, in the 1960s (1960-1969)?

(B) How many leading roles were available to both actors and actresses, in the 2000s (2000-2009)?

- Hint: A specific value of n might indicate a leading role

[43]:
```
leading_roles = cast[(cast.n == 1) & ((cast.type == 'actor') | (cast.type ==␣
↪'actress')) & ((cast.year >= 1960) & (cast.year < 1970))]
len(leading_roles)
```

[43]: 11823

[44]:
```
leading_roles = cast[(cast.n == 1) & ((cast.type == 'actor') | (cast.type ==␣
↪'actress')) & ((cast.year >= 2000) & (cast.year < 2010))]
len(leading_roles)
```

[44]: 26344

3.1.10 Section II - Q9: List, in order by year, each of the films in which Frank Oz has played more than 1 role.

[45]:
```
movies_by_frank = cast[(cast.name == 'Frank Oz')]
movies_by_frank_groupby = movies_by_frank.groupby(by = ['title', 'year']).
↪size().to_frame('count').reset_index()
movies_more_than_one = movies_by_frank_groupby[movies_by_frank_groupby['count']␣
↪> 1]
movies_more_than_one = movies_more_than_one.sort_values(by = ['year'])
movies_more_than_one
```

[45]:

	title	year	count
24	The Muppet Movie	1979	8
0	An American Werewolf in London	1981	2
22	The Great Muppet Caper	1981	6
20	The Dark Crystal	1982	2
25	The Muppets Take Manhattan	1984	7
2	Follow That Bird	1985	3
23	The Muppet Christmas Carol	1992	7
7	Muppet Treasure Island	1996	4
8	Muppets from Space	1999	4
18	The Adventures of Elmo in Grouchland	1999	3

3.1.11 Section II - Q10: List each of the characters that Frank Oz has portrayed at least twice

[46]:
```
movies_by_frank = cast[(cast.name == 'Frank Oz')]
char_by_frank_groupby = movies_by_frank.groupby(by = ['character']).size().
↪to_frame('count').reset_index()
char_more_than_one = char_by_frank_groupby[char_by_frank_groupby['count'] > 1]
char_more_than_one
```

[46]:

	character	count
0	Animal	6
2	Bert	3
5	Cookie Monster	5
10	Fozzie Bear	4
15	Grover	2
18	Miss Piggy	6
25	Sam the Eagle	5
34	Yoda	6

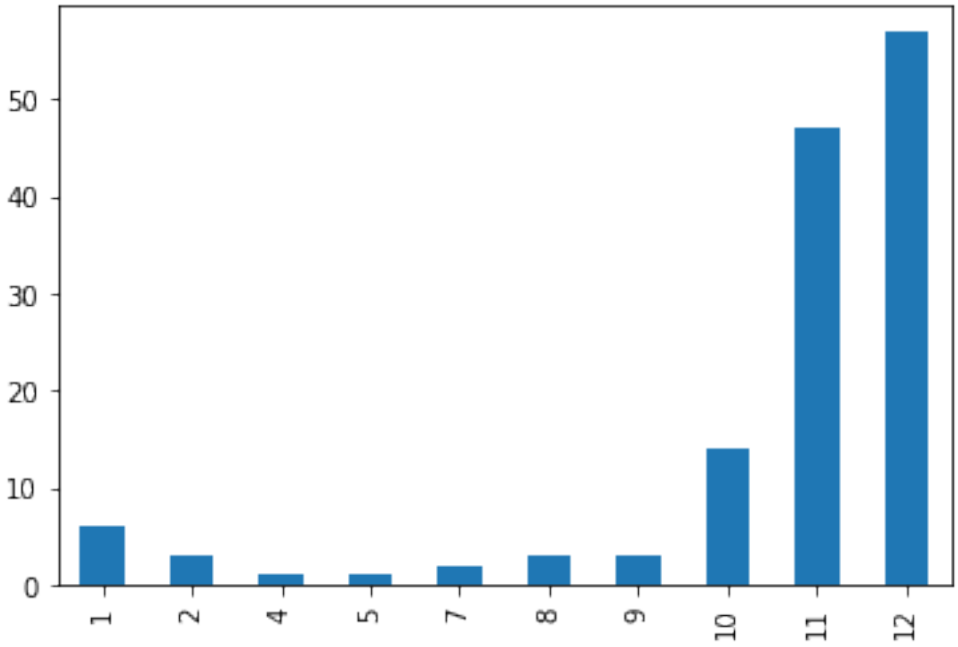
4 Section III - Advanced Merging, Querying and Visualizations

4.1 Make a bar plot with the following conditions

- Frequency of the number of movies with "Christmas" in their title
- Movies should be such that they are released in the USA.
- Show the frequency plot by month

[47]:
```
christmas = release_dates[(release_dates.title.str.contains('Christmas')) &␣
↪(release_dates.country == 'USA')]
christmas.date.dt.month.value_counts().sort_index().plot(kind='bar')
```

[47]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd1e0b37b50>
```

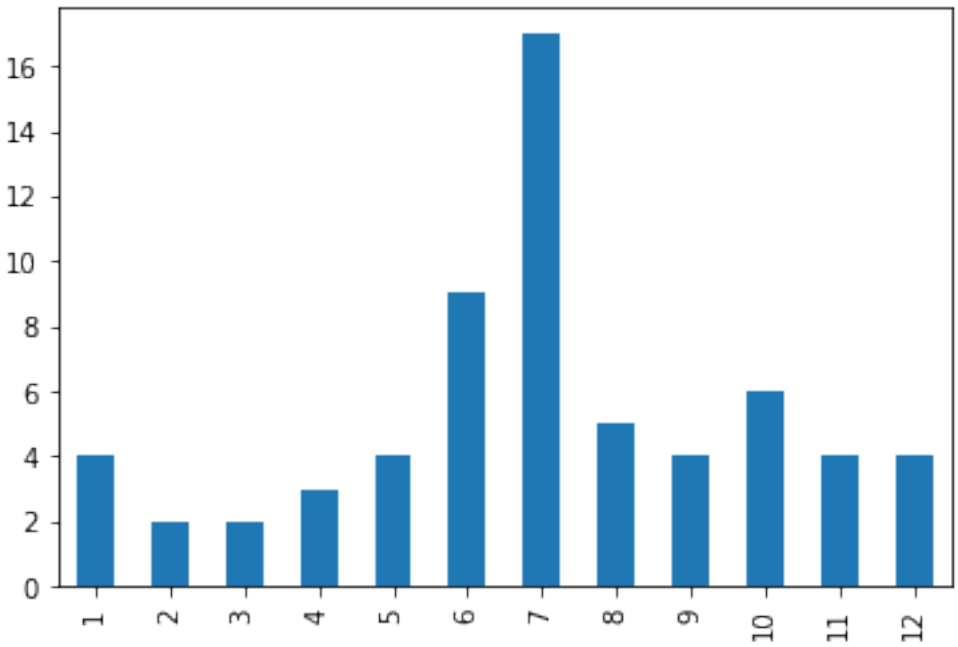


4.1.1 Section III - Q1: Make a bar plot with the following conditions

- Frequency of the number of movies with "Summer" in their title
- Movies should be such that they are released in the USA.
- Show the frequency plot by month

[48]:
```
summer = release_dates[(release_dates.title.str.contains('Summer')) &␣
↪(release_dates.country == 'USA')]
summer.date.dt.month.value_counts().sort_index().plot(kind='bar')
```

[48]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd1e0ab9c10>
```

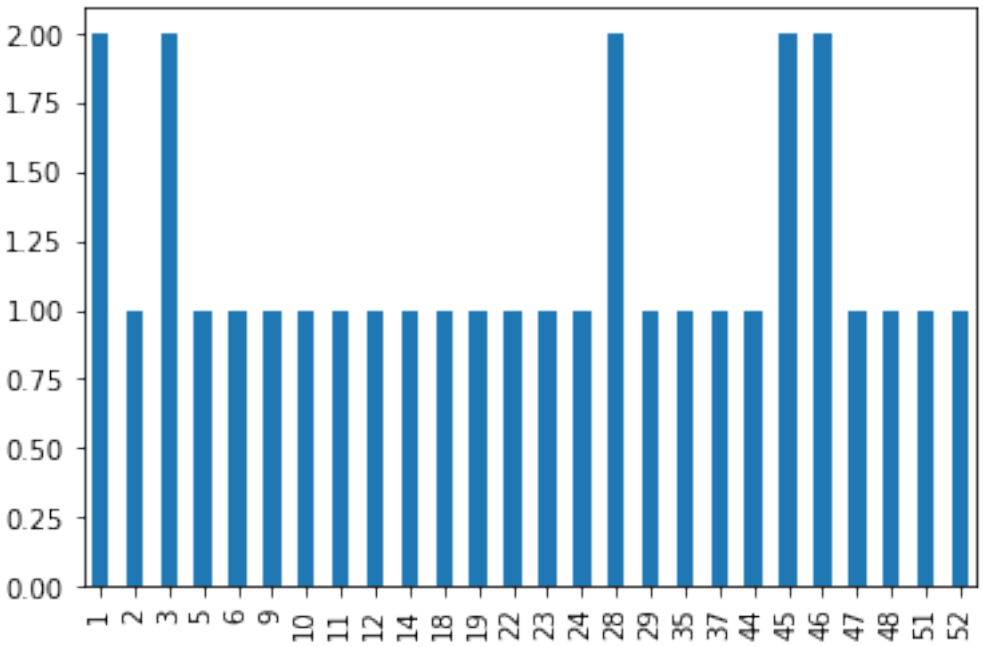


4.1.2 Section III - Q2: Make a bar plot with the following conditions

- Frequency of the number of movies with "Action" in their title
- Movies should be such that they are released in the USA.
- Show the frequency plot by week

[51]:
```
action = release_dates[(release_dates.title.str.contains('Action')) &␣
↪(release_dates.country == 'USA')]
action.date.dt.week.value_counts().sort_index().plot(kind='bar')
```

[51]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd1df8788d0>
```



4.1.3 Section III - Q3: Show all the movies in which Keanu Reeves has played the lead role along with their release date in the USA sorted by the date of release

- Hint: You might need to join or merge two datasets!

[81]:
```
lead_keanu = cast[(cast['name'] == 'Keanu Reeves') & (cast['n'] == 1)]
usa_release = release_dates[(release_dates.country == 'USA')]
#lead_keanu
cast_and_release = pd.merge(lead_keanu, usa_release, on='title')
required_df = cast_and_release.groupby(by = ['title']).count().reset_index()
#required_df.title

lst = []
for movie in required_df.title:
    lst.append(cast_and_release.iloc[cast_and_release[cast_and_release.title ==␣
 ↪movie].index]['date'].to_string().split(' ')[-1])
required_df['date'] = lst
required_df.iloc[:, [0,-1]]
```

[81]:

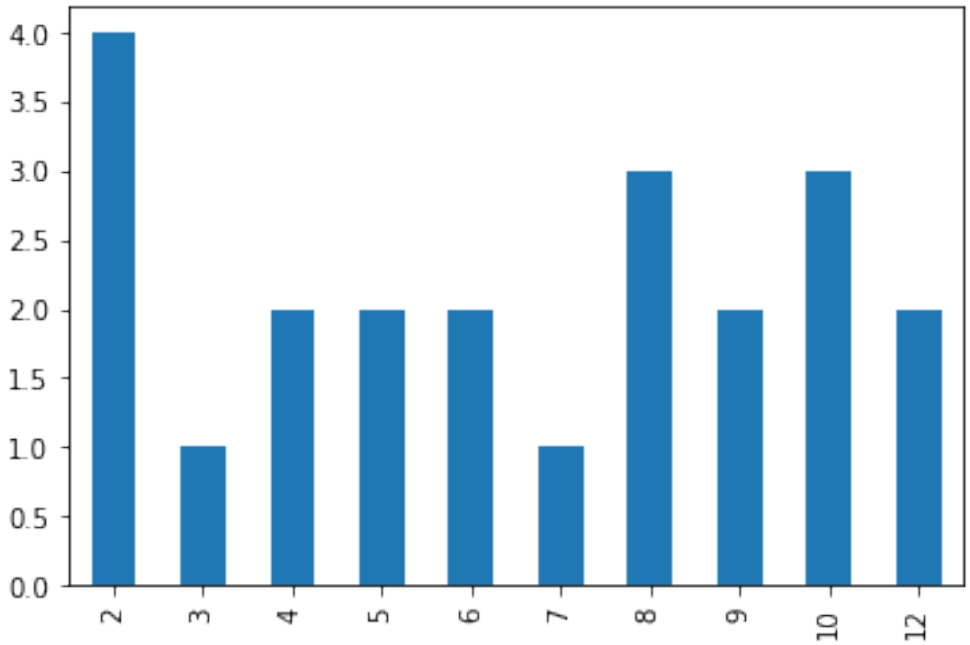
	title	date
0	47 Ronin	2013-12-25
1	A Walk in the Clouds	1995-08-11
2	Bill & Ted's Bogus Journey	1991-07-19
3	Bill & Ted's Excellent Adventure	1989-02-17
4	Chain Reaction	1996-08-02

5	Constantine	2005-02-18
6	Feeling Minnesota	1996-09-13
7	Hard Ball	2001-09-14
8	John Wick	2014-10-24
9	John Wick: Chapter 2	2017-02-10
10	Johnny Mnemonic	1995-05-26
11	Knock Knock	2017-10-06
12	Little Buddha	1994-05-25
13	Speed	1994-06-10
14	Street Kings	2008-04-11
15	Sweet November	2001-02-16
16	The Day the Earth Stood Still	2008-12-12
17	The Devil's Advocate	1997-10-17
18	The Lake House	2006-06-16
19	The Matrix	1999-03-31
20	The Night Before	1988-04-15
21	The Replacements	2000-08-11

4.1.4 Section III - Q4: Make a bar plot showing the months in which movies with Keanu Reeves tend to be released in the USA?

[87]:
```
required_df['date'] = pd.to_datetime(required_df['date'])
required_df.date.dt.month.value_counts().sort_index().plot(kind='bar')
```

[87]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc7c178f610>



4.1.5 Section III - Q5: Make a bar plot showing the years in which movies with Ian McKellen tend to be released in the USA?

[90]:
```
movie_ian = cast[(cast['name'] == 'Ian McKellen')]
usa_release = release_dates[(release_dates.country == 'USA')]
cast_and_release = pd.merge(movie_ian, usa_release, on='title')
required_df = cast_and_release.groupby(by = ['title']).count().reset_index()

lst = []
for movie in required_df.title:
    lst.append(cast_and_release.iloc[cast_and_release[cast_and_release.title ==␣
 ↪movie].index]['date'].to_string().split(' ')[-1])
required_df['date'] = lst
required_df.iloc[:, [0,-1]]
required_df['date'] = pd.to_datetime(required_df['date'])
required_df.date.dt.year.value_counts().sort_index().plot(kind='bar')
```

[90]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc7beb36210>