Homework 2 : Data Collection

```
 ## Notebook Presentation [6 points ]
```

```
# For web and html
import requests
from time import sleep
from bs4 import BeautifulSoup

#For working with raw files
import zipfile

#For working with data
import pandas as pd
from datetime import datetime
```

Problem 1 : What's the secret code? [4 points]

Q 1. Use the library we learned in class to get the page's HTML. [1 point]. See 2:36 in the Solutions Video

```
BONUS_URL = "https://csc380.beingenfa.com/Bonus/1.html"
```

```
bonus_req_obj = requests.get(BONUS_URL)
bonus_req_obj
```

```
<Response [200]>
```

```
bonus_page_html = bonus_req_obj.text
bonus_page_html
```

```
'<!DOCTYPE html>\n<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xm
l:lang="en"><head>\n\n<meta charset="utf-8">\n<meta name="generator" conte
nt="quarto-1.3.361">\n\n<meta name="viewport" content="width=device-width,
initial-scale=1.0, user-scalable=yes">\n\n\n<title>CSC 380 – quarto-inputa
5805b45</title>\n<style>\ncode{white-space: pre-wrap;}\nspan.smallcaps{fon
t-variant: small-caps;}\ndiv.columns{display: flex; gap: min(4vw, 1.5em);}
\ndiv.column{flex: auto; overflow-x: auto;}\ndiv.hanging-indent{margin-lef
```

Q 2. Find the section with the secret code by using the Beautiful Soup's find function [2 points]. See 5:53 in the Solutions Video

```
bonus_bs4_obj = BeautifulSoup(bonus_page_html,"html.parser")
bonus_bs4_obj.prettify
```

```
<bound method Tag.prettify of <!DOCTYPE html>

<html lang="en" xml:lang="en" xmlns="http://www.w3.org/1999/xhtml"><head>
<meta charset="utf-8"/>
<meta content="quarto-1.3.361" name="generator"/>
<meta content="width=device-width, initial-scale=1.0, user-scalable=yes"
name="viewport"/>
<title>CSC 380 – quarto-inputa5805b45</title>
<style>
code{white-space: pre-wrap;}
span.smallcaps{font-variant: small-caps;}
div.columns{display: flex; gap: min(4vw, 1.5em);}
div.column{flex: auto; overflow-x: auto;}
div.hanging-indent{margin-left: 1.5em; text-indent: -1.5em;}
ul.task-list{list-style: none;}
ul.task-list li input[type="checkbox"] {
  width: 0.8em;
  margin: 0 0.8em 0.2em -1em; /* quarto-specific, see https://github.com/quarto-
dev/quarto-cli/issues/4556 */
  vertical-align: middle;
}
</style>
<script src="../site_libs/quarto-nav/quarto-nav.js"></script>
<script src="../site_libs/quarto-nav/headroom.min.js"></script>
<script src="../site_libs/clipboard/clipboard.min.js"></script>
<script src="../site_libs/quarto-search/autocomplete.umd.js"></script>
<script src="../site_libs/quarto-search/fuse.min.js"></script>
<script src="../site_libs/quarto-search/quarto-search.js"></script>
<meta content="../" name="quarto:offset"/>
<script src="../site_libs/quarto-html/quarto.js"></script>
<script src="../site_libs/quarto-html/popper.min.js"></script>
<script src="../site_libs/quarto-html/tippy.umd.min.js"></script>
<script src="../site_libs/quarto-html/anchor.min.js"></script>
<link href="../site_libs/quarto-html/tippy.css" rel="stylesheet"/>
<link href="../site_libs/quarto-html/quarto-syntax-highlighting.css" id="quarto-text-
highlighting-styles" rel="stylesheet"/>
<script src="../site_libs/bootstrap/bootstrap.min.js"></script>
<link href="../site_libs/bootstrap/bootstrap-icons.css" rel="stylesheet"/>
<link data-mode="light" href="../site_libs/bootstrap/bootstrap.min.css" id="quarto-
bootstrap" rel="stylesheet"/>
<script id="quarto-search-options" type="application/json">{
  "location": "sidebar",
  "copy-button": false,
  "collapse-after": 3,
  "panel-placement": "start",
  "type": "textbox",
  "limit": 20,
  "language": {
    "search-no-results-text": "No results",
    "search-matching-documents-text": "matching documents",
    "search-copy-link-title": "Copy link to search",
    "search-hide-matches-text": "Hide additional matches",
    "search-more-match-text": "more match in this document",
    "search-more-matches-text": "more matches in this document",
    "search-clear-button-title": "Clear",
    "search-detached-cancel-button-title": "Cancel",
    "search-submit-button-title": "Submit",
    "search-label": "Search"
```

```
secret_section = bonus_bs4_obj.find("div","Secret")
secret_section
```

```
<div class="Secret">
<dr_cynthia_breazeal></dr_cynthia_breazeal>
</div>
```

Q 3. Clean up the secret code and print it as "The Secret Code is: CSC380 " [1 point]. See 10:30 in the Solutions Video

```
list(secret_section.children)
```

```
['\n', <dr_cynthia_breazeal></dr_cynthia_breazeal>, '\n']
```

```
secret_code_tag = list(secret_section.children)[1]
secret_code_tag
```

```
<dr_cynthia_breazeal></dr_cynthia_breazeal>
```

```
secret_code = secret_code_tag.name
secret_code
```

```
'dr_cynthia_breazeal'
```

```
print("The Secret Code is: ",secret_code)
```

```
The Secret Code is:  dr_cynthia_breazeal
```

Problem 2. Random Facts API [11 points].

```
RANDOM_FACT_WEBSITE_URL = "https://uselessfacts.jsph.pl"
RANDOM_FACTS_ENDPOINT = "/api/v2/facts/random"
TODAY_RANDOM_FACT_ENDPOINT = "/api/v2/facts/today"
```

Q1. Find the URL for random facts API. [1 point]. See 13:38 in the Solutions Video

```
random_facts_api = RANDOM_FACT_WEBSITE_URL+RANDOM_FACTS_ENDPOINT
random_facts_api
```

```
'https://uselessfacts.jsph.pl/api/v2/facts/random'
```

Qs. 2,3,4 . Collect 10 Random Facts [3 point]. See 16:48 in the Solutions Video

```
random_facts_list_of_json = []
nos_of_facts = 10

for fact_no in range(nos_of_facts):

    random_fact_req_obj = requests.get(random_facts_api)

    if random_fact_req_obj.status_code == 200:

        random_facts_list_of_json.append(random_fact_req_obj.json())

    sleep(2)


len(random_facts_list_of_json)
```

```
10
```

```
random_facts_list_of_json
```

```
[{'id': 'e5ab712476d43f31e76bb1729af8d864',
  'text': 'The Main Library at Indiana University sinks over an inch every year because
when it was built, engineers failed to take into account the weight of all the books that
would occupy the building.',
  'source': 'djtech.net',
  'source_url': 'http://www.djtech.net/humor/useless_facts.htm',
  'language': 'en',
  'permalink':
'https://uselessfacts.jsph.pl/api/v2/facts/e5ab712476d43f31e76bb1729af8d864'},
 {'id': 'f3b6f48bf9ab74f1937f61e2c00cddfe',
  'text': 'In 1912 a law passed in Nebraska where drivers in the country at night were
required to stop every 150 yards, send up a skyrocket, wait eight minutes for the road to
clear before proceeding cautiously, all the while blowing their horn and shooting off
flares.',
  'source': 'djtech.net',
  'source_url': 'http://www.djtech.net/humor/useless_facts.htm',
  'language': 'en',
  'permalink':
'https://uselessfacts.jsph.pl/api/v2/facts/f3b6f48bf9ab74f1937f61e2c00cddfe'},
 {'id': 'ec0e1f3ff567de996194a62062e5e295',
  'text': 'Tehran is the most expensive city on earth.',
  'source': 'djtech.net',
  'source_url': 'http://www.djtech.net/humor/useless_facts.htm',
  'language': 'en',
  'permalink':
'https://uselessfacts.jsph.pl/api/v2/facts/ec0e1f3ff567de996194a62062e5e295'},
 {'id': 'be3e622e9fee8b4e6e21e1d6d7359edf',
  'text': 'The shape of plant collenchyma’s cells and the shape of the bubbles in beer
foam are the same - they are orthotetrachidecahedrons.',
  'source': 'djtech.net',
  'source_url': 'http://www.djtech.net/humor/useless_facts.htm',
  'language': 'en',
  'permalink':
'https://uselessfacts.jsph.pl/api/v2/facts/be3e622e9fee8b4e6e21e1d6d7359edf'},
 {'id': '1173c33c84146fcc7825c04683ce9496',
  'text': 'One-fourth of the world`s population lives on less than $200 a year.\xa0
Ninety million people survive on less than $75 a year.',
  'source': 'djtech.net',
  'source_url': 'http://www.djtech.net/humor/useless_facts.htm',
  'language': 'en',
  'permalink':
'https://uselessfacts.jsph.pl/api/v2/facts/1173c33c84146fcc7825c04683ce9496'},
```

```
 {'id': '3f4bac8655449adc95b7eb89c129e17e',
  'text': 'Tigers not only have striped fur, they have striped skin!',
  'source': 'djtech.net',
  'source_url': 'http://www.djtech.net/humor/useless_facts.htm',
  'language': 'en',
  'permalink':
'https://uselessfacts.jsph.pl/api/v2/facts/3f4bac8655449adc95b7eb89c129e17e'},
 {'id': '3639a3cf749725c9b903dc0e500be8cc',
  'text': 'One third of all cancers are sun related.',
  'source': 'djtech.net',
  'source_url': 'http://www.djtech.net/humor/useless_facts.htm',
  'language': 'en',
  'permalink':
'https://uselessfacts.jsph.pl/api/v2/facts/3639a3cf749725c9b903dc0e500be8cc'},
 {'id': '3b99d35171dc8fc7abbc1891aab4d6cb',
  'text': 'The dot over the letter `i` is called a tittle. \xa0',
```

Q.5 Creating the Dataframe [1 point]. See 22:26 in the Solutions Video

```
random_facts_df = pd.DataFrame(random_facts_list_of_json)
random_facts_df.sample(2)
```

	id	text	source	
9	ca89075fd3a528c2be2feac0be27383a	A spider has transparent blood.	djtech.net	http://www.djtech.net/humor/us

```
random_facts_df
```

Q 6. Display Full Facts [2 points]. See 24:03 in the Solutions Video

```
random_facts_series = random_facts_df['text'] # Part a
random_facts_series
```

```
0    The Main Library at Indiana University sinks o...
1    In 1912 a law passed in Nebraska where drivers...
2          Tehran is the most expensive city on earth.
3    The shape of plant collenchyma’s cells and the...
4    One-fourth of the world`s population lives on ...
5    Tigers not only have striped fur, they have st...
6            One third of all cancers are sun related.
7    The dot over the letter `i` is called a tittle.
8        The electric chair was invented by a dentist.
9                     A spider has transparent blood.
Name: text, dtype: object
```

```
random_facts_list = random_facts_series.to_list() # Part b
random_facts_list
```

```
['The Main Library at Indiana University sinks over an inch every year because when it was
built, engineers failed to take into account the weight of all the books that would occupy
the building.',
 'In 1912 a law passed in Nebraska where drivers in the country at night were required to
stop every 150 yards, send up a skyrocket, wait eight minutes for the road to clear before
proceeding cautiously, all the while blowing their horn and shooting off flares.',
 'Tehran is the most expensive city on earth.',
 'The shape of plant collenchyma’s cells and the shape of the bubbles in beer foam are the
same - they are orthotetrachidecahedrons.',
 'One-fourth of the world`s population lives on less than $200 a year.\xa0 Ninety million
people survive on less than $75 a year.',
 'Tigers not only have striped fur, they have striped skin!',
 'One third of all cancers are sun related.',
 'The dot over the letter `i` is called a tittle. \xa0',
 'The electric chair was invented by a dentist.',
 'A spider has transparent blood.']
```

Q 7. Show 3 random facts from the data frame [1 point]. See 25:58

```
random_facts_df.sample(3)
```

	id	text	source	
1	f3b6f48bf9ab74f1937f61e2c00cddfe	In 1912 a law passed in Nebraska where drivers...	djtech.net	http://www.djtech.net/humo

Q8. What is today’s random fact? [3 points]. See 26:29 in the Solutions Video

Part a

Double-click (or enter) to edit

```
todays_random_fact_api = RANDOM_FACT_WEBSITE_URL+ TODAY_RANDOM_FACT_ENDPOINT
todays_random_fact_api
```

```
    'https://uselessfacts.jsph.pl/api/v2/facts/today'

today_fact_req_obj = requests.get(todays_random_fact_api)
today_fact_req_obj.status_code
```

```
    200
```

```
random_fact_of_the_day = today_fact_req_obj.json()['text'] #Part a
random_fact_of_the_day
```

```
    'Ants closely resemble human manners:\xa0 When they wake, they stretch & a
    ppear to yawn in a human manner before taking up the tasks of the day.'
```

Part b

```
time_rn = datetime.today()
print('Time right now is :',time_rn.strftime("%Y-%m-%d %I:%M:%S %p"))
```

```
    Time right now is : 2023-07-13 02:52:59 AM
```

Part c

```
print("At", time_rn.strftime("%Y-%m-%d %I:%M:%S %p"), " the random fact of the day is ",random_fa
```

```
    At 2023-07-13 02:52:59 AM  the random fact of the day is  Ants closely resemble human manner
```

Part 3. Movies and Shows [29 points]

Q1. Download the following dataset [2 points]. See 34:18 in the Solutions Video

```
dataset_names = ['hulu','disney','prime','netflix']

for dataset_name in dataset_names:
    with zipfile.ZipFile(dataset_name+".zip","r") as zip_ref:
        zip_ref.extractall(dataset_name)
```

Q2. Create one large dataframe [3 points]. See 39:10 in the Solutions Video

Part a

```
hulu_df = pd.read_csv('hulu/hulu_titles.csv')
hulu_df.sample()
```

	show_id	type	title	director	cast	country	date_added	release_year
1058	s1059	TV Show	WWE Main	NaN	NaN	United States	January 7, 2021	2012

```
netflix_df = pd.read_csv('netflix/netflix_titles.csv')
netflix_df.sample()
```

	show_id	type	title	director	cast	country	date_added	release_year
6487	s6488	TV Show	City in the Sky	NaN	NaN	United Kingdom	October 1, 2017	2016

```
prime_df = pd.read_csv('prime/amazon_prime_titles.csv')
prime_df.sample()
```

	show_id	type	title	director	cast	country	date_added	release_
7912	s7913	Movie	One More Saturday Night	Dennis Klein	Tom Davis, Al Franken, Moira Sinise, Frank	NaN	NaN	

```
disney_df = pd.read_csv('disney/disney_plus_titles.csv')
disney_df.sample()
```

	show_id	type	title	director	cast	country	date_added	releas
1020	s1021	Movie	Lilo & Stitch	Christopher Sanders, Dean DeBlois	Daveigh Chase, Christopher Sanders, Tia Carrer...	United States	November 12, 2019	

Part b

```
hulu_df['Platform'] = "Hulu"
netflix_df['Platform'] = "Netflix"
disney_df['Platform'] = "Disney"
prime_df['Platform'] = "Prime"
```

```
hulu_df.sample()
```

	show_id	type	title	director	cast	country	date_added	release_year
1193	s1194	TV Show	A Sap Story	NaN	NaN	NaN	November 13, 2020	2016

```
netflix_df.sample()
```

	show_id	type	title	director	cast	country	date_added	release
6818	s6819	Movie	G-Force	Hoyt Yeatman	Bill Nighy, Will Arnett, Zach Galifianakis,	United States	March 15, 2019	

```
disney_df.sample()
```

	show_id	type	title	director	cast	country	date_added	release_y
1135	s1136	Movie	Pixel Perfect	Mark Dippé	Ricky Ullman, Leah Pipes, Spencer Redford,	United States	November 12, 2019	2

```
prime_df.sample()
```

	show_id	type	title	director	cast	country	date_added	rele
7077	s7078	Movie	Japanese Story	Sue Brooks	Toni Collette, Otaro Tsunashima, Matthew Dykty...	NaN	NaN	

Part c

```
all_platforms_df = pd.concat([prime_df,netflix_df,disney_df,hulu_df])
all_platforms_df.sample(5)
```

Q3. Longest show and movie [6 points]. See 44:49 for part a, and 55:20 for part b, in the Solutions Video

6181	s6182	Movie	Extraordinary	Desmares,		Paul	Canada	October 1,

Part a

```
shows_df = all_platforms_df[all_platforms_df['type']=='TV Show']
shows_df.sample()
```

	show_id	type	title	director	cast	country	date_added	release_yea
2137	s2138	TV Show	Light as a Feather	NaN	NaN	United States	July 26, 2019	201

```
shows_count_df = shows_df.value_counts("duration").to_frame()
shows_count_df.sample(3)
```

	0
duration	
4 Seasons	280
19 Seasons	3
26 Seasons	1

```
shows_count_df.reset_index(inplace=True)
shows_count_df.sample(3)
```

	duration	0
24	26 Seasons	1
12	13 Seasons	6
15	16 Seasons	4

```
shows_count_df = shows_count_df.rename({
    'duration' : 'Number of seasons',
    0: 'No of shows'
    }, axis = 1)

shows_count_df.sample(3)
```

	Number of seasons	No of shows
22	32 Seasons	1
6	7 Seasons	89
0	1 Season	4183

```
print("Before preprocessing , Longest running season appears to be :",shows_count_df['Number of s
```

```
Before preprocessing , Longest running season appears to be : 9 Seasons
```

Bonus

```
shows_count_df['Number of seasons'] = shows_count_df['Number of seasons'].apply(lambda x : int(x.
shows_count_df.sample()
```

	Number of seasons	No of shows
6	7	89

```
print("After preprocessing , Longest running season appears to be :",shows_count_df['Number of se
```

```
After preprocessing , Longest running season appears to be : 34 Seasons
```

Part b

```
movies_df = all_platforms_df[all_platforms_df['type']=='Movie']
movies_df.sample()
```

	show_id	type	title	director	cast	country	date_added	release_y
2595	s2596	Movie	Dwayne Perkins: Take Note	Ian Harris	Dwayne Perkins	NaN	NaN	

```
movies_count_df = movies_df.value_counts("duration").to_frame()
movies_count_df.sample(3)
```

	0
duration	
25 min	25
69 min	66
229 min	1

```
movies_count_df.reset_index(inplace=True)
movies_count_df.sample(3)
```

	duration	0
184	175 min	5
159	160 min	13
100	145 min	43

```
movies_count_df = movies_count_df.rename({
    'duration' : 'Duration in minutes',
    0: 'No of movies'
    }, axis = 1)

movies_count_df.sample(3)
```

	Duration in minutes	No of movies
88	48 min	54
31	111 min	153
55	125 min	87

Part i

```
movies_count_df_i = movies_count_df.sort_values(by="Duration in minutes", ascending=False)
movies_count_df_i
```

	Duration in minutes	No of movies
12	99 min	288
13	98 min	286
8	97 min	324
10	96 min	303
7	95 min	340
...	...	...
16	101 min	253
14	100 min	260
126	10 min	25
165	1 min	11
166	0 min	10

225 rows × 2 columns

```
longest_movie_duration = movies_count_df_i.head(1)['Duration in minutes'].to_list()[0]
print("Longest Movie Duration is : ",longest_movie_duration)
```

```
Longest Movie Duration is :  99 min
```

```
longest_movies_df = movies_df[movies_df["duration"] == longest_movie_duration ]
longest_movies_df.sample(2)
```

show_id	type	title	director	cast	country	date_added	rele
				Bokeem Woodbine, Darin Snoop			

```
print("Longest Movie Duration is : ",longest_movie_duration)
print("The Movies are : ","\n".join(longest_movies_df['title'].to_list()))
```

```
Longest Movie Duration is :  99 min
The Movies are :  Woman Of Desire
Why We Fight
Valley Uprising
The Zookeeper
The Woman in Black 2: Angel of Death
The Ultimate Legacy
The Italian Job (1969)
The Hungry
The Donkey King
Super Size Me
Sunshine Hotel
Street Dance
Storm Boy
Shottas
Our Town
My Foolish Heart
Line of Duty
Honest Thief
Hiding Out
Hellbound: Hellraiser 2
Hearts in Bondage
Happythankyoumoreplease
Grace of God
Go Fast. Go North.
Doe
Digging to China
Better Luck Tomorrow
Belle and Sebastian
Atlas Shrugged: Part III
Anna
23 Blast
The Adventurer: The Curse of the Midas Box
Hallowed Be Thy Name
Madness in the Method
Hick
Assault on Wall Street
The Baytown Outlaws
Rolling Stone: Life And Death Of Brian Jones
Dark Was the Night
Caught Up
Stinger
Goodbye, Butterfly
Little Kingdom
PERSECUTION
Blame it On Fidel
Reversion
Happy Home
Super Fast
Intersection
Tusks
The Shootist
DeepStar Six
Firefly
```

```
Rock the Casbah
Nightingale: A Melody of Life
Toys Storage 2
```

Part ii

```
movies_df.sample()
```

	show_id	type	title	director	cast	country	date_added	release_y
8078	s8079	Movie	St. Agatha	Darren Lynn Bousman	Sabrina Kern, Carolyn Hennesy, Courtney Halver...	United States	August 8, 2019	

```
movies_count_df_ii = movies_df["duration"].value_counts().to_frame()
movies_count_df_ii.sample()
```

	duration
20 min	25

```
longest_duration_for_movies = movies_count_df_ii.sort_index().tail(1).index[0]
longest_duration_for_movies
```

```
'99 min'
```

```
longest_movies_df = movies_df[movies_df["duration"] == longest_movie_duration ]
longest_movies_df.sample(2)
```

	show_id	type	title	director	cast	country	date_added	release_year
875	s876	Movie	Street Dance	Dania Pasquini, Max Giwa	Nichola Burley, Richard Winsor, Charlotte Ramp...	NaN	NaN	2013
			Dust 2			United	November	

```
print("Longest Movie Duration is : ",longest_movie_duration)
print("The Movies are : ","\n".join(longest_movies_df['title'].to_list()))
```

```
Longest Movie Duration is :  99 min
The Movies are :  Woman Of Desire
Why We Fight
Valley Uprising
The Zookeeper
The Woman in Black 2: Angel of Death
The Ultimate Legacy
The Italian Job (1969)
The Hungry
The Donkey King
Super Size Me
Sunshine Hotel
```

```
Street Dance
Storm Boy
Shottas
Our Town
My Foolish Heart
Line of Duty
Honest Thief
Hiding Out
Hellbound: Hellraiser 2
Hearts in Bondage
Happythankyoumoreplease
Grace of God
Go Fast. Go North.
Doe
Digging to China
Better Luck Tomorrow
Belle and Sebastian
Atlas Shrugged: Part III
Anna
23 Blast
The Adventurer: The Curse of the Midas Box
Hallowed Be Thy Name
Madness in the Method
Hick
Assault on Wall Street
The Baytown Outlaws
Rolling Stone: Life And Death Of Brian Jones
Dark Was the Night
Caught Up
Stinger
Goodbye, Butterfly
Little Kingdom
PERSECUTION
Blame it On Fidel
Reversion
Happy Home
Super Fast
Intersection
Tusks
The Shootist
DeepStar Six
Firefly
Rock the Casbah
Nightingale: A Melody of Life
Toys Storage 2
Hoop Soldiers
```

Part iii

```
movies_df.sample()
```

	show_id	type	title	director	cast	country	date_added	release_year	ra
1999	s2000	Movie	Journey Through The Stars	Mark Knight	NaN	NaN	NaN	2020	

```
movies_df['duration'].max()
```

Show hidden output

```
movies_df.idxmax(axis="columns")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-66-9ffe251f6e56> in <cell line: 1>()
----> 1 movies_df.idxmax(axis="columns")

                                 7 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/nanops.py in _f(*args,
**kwargs)
     86             if any(self.check(obj) for obj in obj_iter):
     87                 f_name = f.__name__.replace("nan", "")
---> 88                 raise TypeError(
     89                     f"reduction operation '{f_name}' not allowed for
this dtype"
     90                 )

TypeError: reduction operation 'argmax' not allowed for this dtype
```

Max function tried to compare values in the column. But the column had two datatypes in it - string and float(for null values). Comparison was not a permitted operation between string and float. Hence the error

Q 4.Shows streaming on multiple platforms [7 points]. See 1:19:12 in the Solutions Video

```
all_platforms_df.sample()
```

	show_id	type	title	director	cast	country	date_added	release_year
2235	s2236	TV Show	Dark Desire	NaN	Maite Perroni, Erik Hayser, Alejandro Speitzer...	Mexico	July 15, 2020	2020

Part a

```
print("The number of rows is :",all_platforms_df.shape[0])
```

```
The number of rows is : 22998
```

Part b

```
unique_titles = all_platforms_df['title'].unique()
unique_titles
```

```
array(['The Grand Seduction', 'Take Care Good Night',
       'Secrets of Deception', ..., 'Star Trek: The Original Series',
       'The Twilight Zone', 'Tokyo Magnitude 8.0'], dtype=object)
```

```
print('Number of unique titles :',len(unique_titles))
```

```
Number of unique titles : 22115
```

```
# alternative solution from student submissions
print(f"Number of unique titles: {all_platforms_df['title'].nunique()}")
```

```
Number of unique titles: 22115
```

Part c

```
titles_count_df = all_platforms_df['title'].value_counts().to_frame()
titles_count_df.sample(2)
```

	title
Once Upon a Time in China III	1
Catch-22	2

Part d

```
titles_count_df.reset_index(inplace=True)
```

```
titles_count_df.rename({
    "index":"name"
}, axis = 1, inplace = True)
```

```
titles_count_df.sample(5)
```

	name	title
6530	America's Future: The Power of the Latino Vote	1
1021	Flavors of Youth: International Version	1
13793	Rock the Casbah	1
7422	Son of Zorn	1
12259	Celebrity Close Calls	1

Part e

```
titles_count_df.rename({
    'name' : "Movie or Show Name",
    "title" : "No of Platforms"
}, axis = 1, inplace = True)
```

```
titles_count_df.sample(5)
```

	Movie or Show Name	No of Platforms
20247	Jeevan Ek Sanghursh	1
21334	Under Crystal Lake	1
4994	Sisters	1
18946	Chhota Bheem & Krishna: Mayanagari	1
17751	Gajendra	1

Part f

```
max_platform_shows = shows_df['title'].value_counts().to_frame()['title'].max()
max_platform_shows
```

```
3
```

```
print("The Maximum number of platforms a show is on is ",max_platform_shows)
```

```
The Maximum number of platforms a show is on is  3
```

Q5. Favorite show or movie [2 points]. See 1:31:20 in the Solutions Video

Part a

```
duplicated_shows = shows_df[shows_df.duplicated(subset=['title'], keep = False)]
duplicated_shows.sample(2)
```

	show_id	type	title	director	cast	country	date_added	release_year
8117	s8118	TV Show	Suits	NaN	Jang Dong-gun, Park Hyung-sik, Jin Hee-kyung, ...	South Korea	December 1, 2019	2018
326	s327	TV Show	Cosmos: Possible	NaN	Neil deGrasse	United States	December 25, 2020	2020

Part b

```
favorite_movie = 'Everything Everywhere All at Once'
```

```
all_platforms_df[all_platforms_df['title'].apply(lambda x : True if x.lower()==favorite_movie.low
```

	show_id	type	title	director	cast	country	date_added	release_year	rating

```
random_row = all_platforms_df.sample(1)
random_title = random_row['title'].values[0]
print(random_title)
random_row
```

```
Fear Street Part 2: 1978
```

	show_id	type	title	director	cast	country	date_added	release_year	r
479	s480	Movie	Fear Street Part 2: 1978	Leigh Janiak	Sadie Sink, Emily Rudd, Ryan Simpkins,	NaN	July 9, 2021	2021	

```
all_platforms_df[all_platforms_df['title'] == random_title]
```

	show_id	type	title	director	cast	country	date_added	release_year	r
479	s480	Movie	Fear Street Part 2: 1978	Leigh Janiak	Sadie Sink, Emily Rudd, Ryan Simpkins,	NaN	July 9, 2021	2021	

Q 6. Save Data [1 points]. See 1:39:11 in the Solutions Video

```
target_file_name = "streaming_titles.csv"
all_platforms_df.to_csv(target_file_name,index=False)
```

Q 7 . Name starts with [8 points]. See 1:39:29 in the Solutions Video

Part a

```
streaming_titles_df = pd.read_csv(target_file_name)
streaming_titles_df.sample(2)
```

Part b

```
first_name = "Enfa"
first_letter_match_movies = movies_df[movies_df['title'].str.startswith(first_name[0])]
first_letter_match_movies.sample(2)
```

	show_id	type	title	director	cast	country	date_added	releas
6721	s6722	Movie	Exit Strategy	Michael Whitton	Jameel Saleem, Kimelia Weathers, Quincy Harris...	United States	March 3, 2019	
				Wilfred	Sathyaraj, Varalakshmi		January 15,	

Part c

```
first_letter_match_shows = shows_df[shows_df['title'].str.startswith(first_name[0])]
first_letter_match_shows.sample(2)
```

	show_id	type	title	director	cast	country	date_added	release_year
2583	s2584	TV Show	Ed Gamble: Blood Sugar	NaN	NaN	NaN	NaN	2019
6690	s6691	TV Show	Emogenius	NaN	Hunter March	United States	December 15, 2018	2017

```
print("Number of unique shows that start with my first name first letter are : ",len(first_letter
```

```
Number of unique shows that start with my first name first letter are :  140
```

Part d

```
TA_first_letter = 'B'

first_letter_match_movies = movies_df[movies_df['title'].str.startswith(TA_first_letter)]
first_letter_match_movies.sample(2)
```

	show_id	type	title	director	cast	country	date_added	re
240	s241	Movie	Baby's Day Out	Patrick Read Johnson	Joe Mantegna, Lara Flynn Boyle, Joe Pantoliano...	United States	April 23, 2021	
4840	s4841	Movie	Bad Genius	Nattawut Poonpiriya	Chutimon Chuengcharoensukying, Chanon Santinat...	Thailand	June 1, 2018	

Part e

```
first_letter_match_shows = shows_df[shows_df['title'].str.startswith(TA_first_letter)]
first_letter_match_shows.sample(2)
```

	show_id	type	title	director	cast	country	date_added	release_ye
9240	s9241	TV Show	Bucket	NaN	Frog Stone, Miriam Margolyes	NaN	NaN	2(
3151	s3152	TV Show	Barbie Dreamtopia	NaN	Erica Lindbeck, Meira Blinkoff,	NaN	NaN	2(

```
first_name_TA_uniq = len(first_letter_match_shows['title'].unique())
print("Number of unique shows that start with our TA's first name first letter are : ",first_name
```

```
Number of unique shows that start with our TA's first name first letter are :  428
```

Part f

```
first_letter_match_shows = shows_df[shows_df['title'].str.startswith(TA_first_letter)]
first_letter_match_shows.sample(2)
```

	show_id	type	title	director	cast	country	date_added	release_year	r
2801	s2802	TV Show	Bakuman	NaN	NaN	Japan	May 1, 2016	2010	
1940	s1941	TV	Black	NaN	NaN	United	December	2014	

```
last_name_TA_uniq = len(first_letter_match_shows['title'].unique())
print("Number of unique shows that start with our TA's Last name first letter are : ",last_name_T
```

```
Number of unique shows that start with our TA's Last name first letter are :  428
```

Part g

```
print("Diff between f and g is", last_name_TA_uniq-first_name_TA_uniq)
```

```
Diff between f and g is 0
```
