Problem_Set_3

Name : Ayush Patel

NUID: 002765119

Q1

In [1]:
```
import pandas as pd
from lxml import BeautifulSoup
import requests
import re
import json
import requests as r
```

In [2]:
```
url = 'https://www.imdb.com/chart/top'
page = requests.get(url)
soup = BeautifulSoup(page.text, 'html.parser')
```

In [3]:
```
table = soup.find_all('table',{"data-caller-name":"chart-top250movie"})
```

In [4]:
```
movie_dictionary = {'Title':[], 'Director':[], 'Actors':[], 'Release_Year':[], 'IMDB_Rating':[]}
```

In [5]:
```
for text in table[0].find_all("tr"):

    try:
        movie_name = " ".join(x for x in text.find('td', class_ = 'titleColumn').text.split("\n")[2].split(" "))
        movie_dictionary["Title"].append(movie_name)

        director = text.find('td', class_ = 'titleColumn').find('a').attrs['title'].split(" (dir.), ")[0]
        movie_dictionary["Director"].append(director)

        actors = (text.find('td', class_ = 'titleColumn').find('a').attrs['title'].split(" (dir.), ")[1])
        movie_dictionary["Actors"].append(actors)

        release_year = text.find('td', class_ = 'titleColumn').span.text.replace('(','').replace(')','')
        movie_dictionary["Release_Year"].append(release_year)


        rating = text.find('td', class_ = 'ratingColumn imdbRating').strong.text
        movie_dictionary['IMDB_Rating'].append(rating)
    except:
        pass
```

In [6]:
```
movie_dictionary = pd.DataFrame(movie_dictionary)
display(movie_dictionary)
```

	Title	Director	Actors	Release_Year	IMDB_Rating
0	The Shawshank Redemption	Frank Darabont	[Tim Robbins, Morgan Freeman]	1994	9.2
1	The Godfather	Francis Ford Coppola	[Marlon Brando, Al Pacino]	1972	9.2
2	The Dark Knight	Christopher Nolan	[Christian Bale, Heath Ledger]	2008	9.0
3	The Godfather Part II	Francis Ford Coppola	[Al Pacino, Robert De Niro]	1974	9.0
4	12 Angry Men	Sidney Lumet	[Henry Fonda, Lee J. Cobb]	1957	9.0
...	...	...	...	...	...
245	Dersu Uzala	Akira Kurosawa	[Maksim Munzuk, Yuriy Solomin]	1975	8.0
246	The Help	Tate Taylor	[Viola Davis, Emma Stone]	2011	8.0
247	Aladdin	Ron Clements	[Scott Weinger, Robin Williams]	1992	8.0
248	Gandhi	Richard Attenborough	[Ben Kingsley, John Gielgud]	1982	8.0
249	The Iron Giant	Brad Bird	[Eli Marienthal, Harry Connick Jr.]	1999	8.0

250 rows × 5 columns

Q2

1)

In [7]:
```
from xml.etree import ElementTree as ET
```

In [8]:
```
file_name = 'recipes.xml'

dom = ET.parse(file_name)
```

In [9]:
```
recipe = dom.findall('recipe')
```

In [10]:
```
recipe_dictionary = {'Title': [], 'Ingredients':[], 'Calories': []}
```

In [11]:
```
for r in recipe:
    try:
        title = r.find('title').text
        recipe_dictionary['Title'].append(title)

        ing = [x.get('name') for x in r.findall('ingredient')]
        recipe_dictionary['Ingredients'].append(ing)

        cal = r.find('nutrition').get('calories')
        recipe_dictionary['Calories'].append(cal)
    except:
        pass
```

In [12]:
```
recipe_dictionary = pd.DataFrame(recipe_dictionary)
recipe_dictionary
```

Out[12]:

	Title	Ingredients	Calories
0	Beef Parmesan with Garlic Angel Hair Pasta	[beef cube steak, onion, sliced into thin ring...	1167
1	Ricotta Pie	[filling, dough, milk]	349
2	Linguine Pescadoro	[linguini pasta, sauce]	532
3	Zuppa Inglese	[egg yolks, milk, Savoiardi biscuits, sugar, A...	612
4	Cailles en Sarcophages	[pastry, filling, package phyllo dough, egg wh...	8892

2)

a)

In [13]:
```
from lxml import etree
tree = etree.parse('recipes.xml')
```

In [14]:
```
results = tree.xpath("/collection/recipe/title")

for result in results:
    print(result.text)
```

```
Beef Parmesan with Garlic Angel Hair Pasta
Ricotta Pie
Linguine Pescadoro
Zuppa Inglese
Cailles en Sarcophages
```

b)

In [15]:
```
results = tree.xpath("/collection/recipe[ingredient[@name = 'olive oil']]/title")

for result in results:
    print(result.text)
```

```
Beef Parmesan with Garlic Angel Hair Pasta
```

c)

In [16]:
```
results = tree.xpath("/collection/recipe[nutrition[@calories < 500 ]]/title")

for result in results:
    print(result.text)
```

```
Ricotta Pie
```

d)

In [17]:
```
results = tree.xpath("/collection/recipe[title = 'Zuppa Inglese' ]/ingredient[@name = 'sugar"]/@amount")

print(results[0])
```

```
0.75
```

e)

In [18]:
```
results = tree.xpath("/collection/recipe[count(preparation/step)<4]/title")

for result in results:
    print(result.text)
```

```
Beef Parmesan with Garlic Angel Hair Pasta
Ricotta Pie
```

f)

In [19]:
```
results = tree.xpath("/collection/recipe/ingredient[count(ingredient) > 0]//ingredient/@name")

for result in results:
    print(result)
```

```
ricotta cheese
eggs
white sugar
vanilla extract
semisweet chocolate chips
flour
baking powder
white sugar
shortening
eggs, lightly beaten
vanilla extract
olive oil
minced cloves of garlic
Italian seasoning
dried thyme
crushed red pepper flakes
crushed tomatoes
black olives, drained
whole baby clams
minced clams, with juice
small salad shrimp
scallops
lemon zest
salt
ground black pepper
chilled unsalted butter
flour
salt
ice water
baked chicken
marinated chicken
small chickens, cut up
Herbes de Provence
dry white wine
orange juice
minced garlic
truffle oil
stock
chicken wings, giblets, and kidney
onions, peeled
carrots, peeled and cut lengthwise
celery, cut lengthwise
bay leaf
small bunch parsley
whole peppercorns
salt
sauteed mushrooms
white button mushrooms
butter
dry white wine
minced garlic
minced shallots
sauce
chicken juices
mushroom juices
sherry
flour
butter
```

g)

In [20]:
```
results = tree.xpath("/collection/recipe//ingredient[count(ingredient)>1]/@name")
for result in results:
    print(result)
```

```
filling
dough
sauce
pastry
filling
baked chicken
marinated chicken
stock
sauteed mushrooms
sauce
```

h)

In [21]:
```
results = tree.xpath("/collection/recipe/ingredient[position() <= 3]/@name")

for result in results:
    print(result)
```

```
beef cube steak
onion, sliced into thin rings
green bell pepper, sliced in rings
filling
dough
milk
linguini pasta
sauce
egg yolks
milk
Savoiardi biscuits
pastry
filling
package phyllo dough
```

Q3

In [22]:
```
key = "JmUxupsYub3kqBPOjBTtNwQQMJacGGdC"
```

In [23]:
```
source = input("Enter Origin : ")
destination = input("Enter Final Destination : ")
```

```
Enter Origin : Northeastern University, Boston
Enter Final Destination : Faneuil Hall Marketplace
```

In [24]:
```
url = "http://www.mapquestapi.com/directions/v2/route?key={}&from={}&to={}".format(key, source, destination)
```

In [25]:
```
result = requests.get(url)
```

In [26]:
```
json_response = result.json()
```

In [27]:
```
dictionary = { "Instruction" : [] , "Distance(mi)" : [] , "Time(s)" : []}
```

In [28]:
```
for leg_information in json_response['route']['legs']:
        for maneuver in leg_information['maneuvers']:
            dictionary['Instruction'].append(maneuver["narrative"])
            dictionary['Distance(mi)'].append(maneuver["distance"])
            dictionary['Time(s)'].append(maneuver["time"])
```

In [29]:
```
dictionary = pd.DataFrame(dictionary)
```

In [30]:
```
dictionary
```

Out[30]:

	Instruction	Distance(mi)	Time(s)
0	Head toward Gainsborough St on Huntington Ave ...	0.1839	37
1	Keep right onto Huntington Ave (RT-9). Go for ...	0.0926	43
2	Turn left onto Massachusetts Ave toward Cambri...	0.0628	25
3	Turn slightly left onto Westland Ave, Go for 0...	0.2746	71
4	Continue on Fenway. Go for 0.1 mi.	0.1404	29
5	Take ramp onto Storrow Dr toward Newton/Downto...	1.2005	154
6	Keep left onto Storrow Dr. Go for 0.4 mi.	0.4374	57
7	Keep left onto Storrow Dr (RT-28) toward I-93/...	0.3461	42
8	Keep right onto Storrow Dr (RT-28 N) toward Lo...	0.1603	36
9	Turn right and take ramp onto I-93 S toward Qu...	0.7357	94
10	Take exit 17A-B toward Gov't Ctr. Go for 0.3 mi.	0.3026	50
11	Turn left onto John F Fitzgerald Surface Rd. G...	0.1106	28
12	Turn right onto State St. Go for 0.2 mi.	0.1560	44
13	Turn right onto Merchants Row. Go for 187 ft.	0.0354	8
14	Arrive at Merchants Row. Your destination is o...	0.0000	0