module 3

July 25, 2021

0.1 Module 3

In this assignment, you will continue working on the movie data from IMDB.

- The data includes movies and ratings from the IMDB website
- Data File(s): imdb.xlsx

Data file contains 3 sheets:

- "imdb": contains records of movies and ratings scraped from IMDB website
- "countries": contains the country (of origin) names
- "directors": contains the director names

We have loaded the data as "df" for you. Follow the instructions and finish the rest.

[1]:
```
#################################################################
### EXECUTE THIS CELL BEFORE YOU TO TEST YOUR SOLUTIONS ###
#################################################################

import imp, os, sys
sol = imp.load_compiled("solutions", "./solutions.py")
sol.get_solutions("imdb.xlsx")
from nose.tools import assert_equal
from pandas.util.testing import assert_frame_equal, assert_series_equal
```

[2]:
```
# Loading the data
import pandas as pd

xls = pd.ExcelFile('imdb.xlsx')
df = xls.parse('imdb')
df_directors = xls.parse('directors')
df_countries = xls.parse('countries')

print("Data Loading Finished.")
```

```
Data Loading Finished.
```

[3]:
```
""" Q1:
Join three Dataframes: df, df_directors, and df_countries with an inner join.
```

```
Store the joined DataFrames in df.

Here are the steps:
1. Merge df with df_countries and assign it df
2. Merge df with df_directors and assign it to df again
There might be errors if the merge is not in this order, so please be careful.

"""


# your code here

df = pd.merge(left = df, right = df_countries, how = 'inner', left_on =␣
↪'country_id', right_on = 'id')
df = pd.merge(left = df,right = df_directors, how = 'inner', left_on =␣
↪'director_id', right_on = 'id')


# After the join, the resulting Dataframe should have 12 columns.
df.shape
df.head()
```

[3]:
```
                movie_title  director_id  country_id content_rating  \
0  The Shawshank RedemptionÊ           34           1              R
1            The Green MileÊ           34           1              R
2             The GodfatherÊ           33           1              R
3    The Godfather: Part IIÊ           33           1              R
4            Apocalypse NowÊ           33           1              R

   title_year  imdb_score      gross  duration  id_x country  id_y  \
0        1994         9.3   28341469       142     1     USA    34
1        1999         8.5  136801374       189     1     USA    34
2        1972         9.2  134821952       175     1     USA    33
3        1974         9.0   57300000       220     1     USA    33
4        1979         8.5   78800000       289     1     USA    33

          director_name
0       Frank Darabont
1       Frank Darabont
2  Francis Ford Coppola
3  Francis Ford Coppola
4  Francis Ford Coppola
```

[4]:
```
assert_equal(df.shape, sol.df.shape)
print("Success!")
```

```
Success!
```

[5]:
```
""" Q2:
Save the first ten rows of movie titles in a variable called first10, then␣
 ↪print it
"""

# your code here

first10 = df['movie_title'][:10]
first10
```

[5]:
```
0    The Shawshank RedemptionÊ
1               The Green MileÊ
2                The GodfatherÊ
3      The Godfather: Part IIÊ
4               Apocalypse NowÊ
5              The Dark KnightÊ
6                    InceptionÊ
7                 InterstellarÊ
8                      MementoÊ
9                 The PrestigeÊ
Name: movie_title, dtype: object
```

[6]:
```
assert_series_equal(first10, sol.first10)
print("Success!")
```

```
Success!
```

[7]:
```
""" Q3:
There's an extra character at the end of each movie title.
Remove it from the data using str.replace.
And print the first ten rows of movie titles again.
"""

# your code here

df['movie_title'] = df['movie_title'].str.replace('Ê','')
df['movie_title'].head(10)
```

[7]:
```
0    The Shawshank Redemption
1               The Green Mile
2                The Godfather
3      The Godfather: Part II
4               Apocalypse Now
5              The Dark Knight
6                    Inception
7                 Interstellar
8                      Memento
```

```
9                The Prestige
Name: movie_title, dtype: object
```

[8]:
```
assert_frame_equal(df, sol.df)
print("Success!")
```

```
Success!
```

[9]:
```
""" Q4:
Who is the director with the most movies? First get the number of movies per␣
 ↪"director_name", then save the director's name
and count as a series of length 1 called "director_with_most"
"""

# your code here
director_with_most =  df["director_name"].value_counts()[:1]
pd.Series(director_with_most)
```

[9]:
```
Christopher Nolan    7
Name: director_name, dtype: int64
```

[10]:
```
assert_series_equal(director_with_most, sol.director_with_most)
print("Success!")
```

```
Success!
```

[11]:
```
"""Q5:
Save all of this director's movies and their ratings in a variable called␣
 ↪all_movies_ratings, then print this variable.
(The director with the most movies you got from the last question.)
"""

# your code here

all_movies = df[df["director_name"] == 'Christopher Nolan']
all_movies_ratings = all_movies[['movie_title','imdb_score']]
all_movies_ratings
```

[11]:

	movie_title	imdb_score
5	The Dark Knight	9.0
6	Inception	8.8
7	Interstellar	8.6
8	Memento	8.5
9	The Prestige	8.5
10	The Dark Knight Rises	8.5
11	Batman Begins	8.3

[12]:
```
assert_frame_equal(all_movies_ratings, sol.all_movies_ratings)
print("Success!")
```

```
Success!
```

[13]:
```
"""Q6:
Recommend a **random** movie that has a rating of over 8.3.
Store the random recommendation in a variable called "rand_goodmovie".
What is the title and imdb_score of your recommendation?

Here are the steps:
1. Query the data ('df' DataFrame) for movies with a rating over 8.3␣
 ↪(imdb_score > 8.3)
2. Create a random integer index location to get a single movie recommendation
3. Save the random movie recommendation in a DataFrame called 'rand_goodmovie'
4. Save the title of the random movie recommendation in a variable␣
 ↪"random_title" and print it
5. Save the imdb_score of the random movie recommendation in a variable␣
 ↪"random_imdb_score" and print it

"""
# Do not modify this part, it's needed for grading
import random
random.seed(0)
grt = df[df["imdb_score"] > 8.3]
good_movie = grt['movie_title']
rand_int = random.randint(0, len(grt) - 1)
rand_goodmovie = grt[rand_int : rand_int + 1]
random_title = rand_goodmovie['movie_title']
random_imdb_score = rand_goodmovie['imdb_score']
# your code here
```

[14]:
```
from nose.tools import assert_in
assert_in(rand_goodmovie[["movie_title", "imdb_score"]].values, sol.
 ↪possible_goodmovies[["movie_title", "imdb_score"]].values)
assert_equal(random_title.iloc[0], rand_goodmovie["movie_title"].iloc[0])
assert_equal(random_imdb_score.iloc[0], rand_goodmovie["imdb_score"].iloc[0])
print("Success!")
```

```
Success!
```