module 2

July 29, 2021

0.1 Module 2

In this assignment, you will work on 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

[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]:
```
""" Q1:
Load and read the 'imdb.xlsx' file. Read the 'imdb' sheet into a DataFrame, df.
"""

import pandas as pd

# your code here
xls = pd.ExcelFile('imdb.xlsx')
df = xls.parse('imdb')
print(type(df))
```

```
<class 'pandas.core.frame.DataFrame'>
```

[3]:
```
############################
### TEST YOUR SOLUTION ###
############################
```

```
assert_frame_equal(df, sol.df)
print("Success!")
```

```
Success!
```

[4]:
```
""" Q2:
Store the dimensions of the DataFrame as a tuple in a variable called 'shape' and print it.

Hint: A tuple is made up of comma separated values inside parenthesis.  e.g. (1, 2)
"""

# your code here
shape = ()
shape = df.shape
print(shape)
```

```
(178, 8)
```

[5]:
```
############################
### TEST YOUR SOLUTION ###
############################

assert_equal(shape, sol.shape)
print("Success!")
```

```
Success!
```

[6]:
```
""" Q3:
Store the column titles and the types of data in variables named 'columns' and 'dtypes', then print them.
"""

# your code here
columns = df.columns
print(columns)
dtypes = df.dtypes
print(dtypes)
```

```
Index(['movie_title', 'director_id', 'country_id', 'content_rating',
       'title_year', 'imdb_score', 'gross', 'duration'],
      dtype='object')
movie_title        object
director_id         int64
country_id          int64
content_rating     object
```

```
title_year          int64
imdb_score        float64
gross               int64
duration            int64
dtype: object
```

[7]:
```
############################
### TEST YOUR SOLUTION ###
############################

assert_equal(columns.all(), sol.columns.all())
assert_series_equal(dtypes, sol.dtypes)
print("Success!")
```

```
Success!
```

[19]:
```
""" Q4:
Examine the first 10 rows of data; store them in a variable called first10
"""

# your code here
first10 = df.head(10)
print(first10)
```

```
                                    movie_title  director_id  country_id  \
0                       The Shawshank RedemptionÊ           34           1
1                                  The GodfatherÊ           33           1
2                                The Dark KnightÊ           16           1
3                          The Godfather: Part IIÊ           33           1
4  The Lord of the Rings: The Return of the KingÊ           83           1
5                                    Pulp FictionÊ           85           1
6                  The Good, the Bad and the UglyÊ           98           2
7                                Schindler's ListÊ          103           1
8                                       InceptionÊ           16           1
9                                    Fight ClubÊ           22           1

  content_rating  title_year  imdb_score      gross  duration
0              R        1994         9.3   28341469       142
1              R        1972         9.2  134821952       175
2          PG-13        2008         9.0  533316061       152
3              R        1974         9.0   57300000       220
4          PG-13        2003         8.9  377019252       192
5              R        1994         8.9  107930000       178
6       Approved        1966         8.9    6100000       142
7              R        1993         8.9   96067179       185
8          PG-13        2010         8.8  292568851       148
9              R        1999         8.8   37023395       151
```

[20]:
```
############################
### TEST YOUR SOLUTION ###
############################

assert_frame_equal(first10, sol.first10)
print("Success!")
```

```
Success!
```

[11]:
```
""" Q5:
Examine the first 5 rows of data; store them in a variable called first5
"""

# your code here
first5 = df.head()
print(first5)
```

```
                                    movie_title  director_id  country_id  \
0                       The Shawshank RedemptionÊ           34           1
1                                  The GodfatherÊ           33           1
2                                The Dark KnightÊ           16           1
3                          The Godfather: Part IIÊ           33           1
4  The Lord of the Rings: The Return of the KingÊ           83           1

  content_rating  title_year  imdb_score      gross  duration
0              R        1994         9.3   28341469       142
1              R        1972         9.2  134821952       175
2          PG-13        2008         9.0  533316061       152
3              R        1974         9.0   57300000       220
4          PG-13        2003         8.9  377019252       192
```

[12]:
```
############################
### TEST YOUR SOLUTION ###
############################

assert_frame_equal(first5, sol.first5)
print("Success!")
```

```
Success!
```

[15]:
```
""" Q6:
Import the "directors" and "countries" sheets into their own DataFrames, df_directors and df_countries.
"""

# your code here
xls = pd.ExcelFile('imdb.xlsx')
```

```
df_directors = xls.parse('directors')
print(type(df_directors))
xls = pd.ExcelFile('imdb.xlsx')
df_countries = xls.parse('countries')
print(type(df_countries))
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
```

[16]:
```
############################
### TEST YOUR SOLUTION ###
############################

assert_frame_equal(df_directors, sol.df_directors)
assert_frame_equal(df_countries, sol.df_countries)
print("Success!")
```

```
Success!
```

[17]:
```
""" Q7:
Check the "directors" sheet
1. Count how many records there are based on the "id" column. (To get the␣
 ↪number of records per "id",
   use the value_counts method.) Store the result in a variable named count.
2. Remove the duplicates from the directors dataframe and store the result in a␣
 ↪variable called df_directors_clean.
"""

# your code here
count = df_directors["id"].value_counts()
df_directors_clean = df_directors.drop_duplicates()
```

[18]:
```
############################
### TEST YOUR SOLUTION ###
############################

assert_series_equal(count, sol.count)
assert_frame_equal(df_directors_clean, sol.df_directors_clean)
print("Success!")
```

```
Success!
```

[ ]:
```

```