Module 6: Data Manipulation

Case Study Solution

Case Study Solution

From the data provided on Hollywood movies:

1. Find the highest rated movie in the “Quest” story type.

Solution

```
import pandas

frame = pandas.read_csv('D:\Datasets\HollywoodMovies.csv', delimiter=',')
selected_columns = frame.loc[:, ["Movie", "RottenTomatoes", "Story"]]

highest_rated = ""
rating = 0

for i in range(0, 970):
   if selected_columns.loc[i]["Story"] == "Quest" and _
columns.loc[i]["RottenTomatoes"] > rating:
      rating = selected_columns.loc[i]["RottenTomatoes"]
      highest_rated = selected_columns.loc[i]["Movie"]

print(highest_rated, rating)
```

2. Find the genre in which there has been the greatest number of movie releases.

Solution

```
import pandas

frame = pandas.read_csv('D:\Datasets\HollywoodMovies.csv', delimiter=',')
genre = frame.loc[:, ["Genre"]]

frequency = dict()
for i in range(0, 970):
   if genre.loc[i][0] in frequency:
      frequency[str(genre.loc[i][0])] += 1
   else:
```

```
    frequency[str(genre.loc[i][0])] = 1
print(frequency)
```

3. Print the names of the top five movies with the costliest budgets.

Solution

```
import pandas

frame = pandas.read_csv('D:\Datasets\HollywoodMovies.csv', delimiter=',')
sorted_frame = frame.sort_values(by='Budget', ascending=False)
print(sorted_frame.loc[:, "Budget"])
```

4. Is there any correspondence between the critics' evaluation of a movie and its acceptance by the public? Find out, by plotting the net profitability of a movie against the ratings it receives on Rotten Tomatoes.

Solution

```
import matplotlib.pyplot as plt, pandas

frame = pandas.read_csv('D:\Datasets\HollywoodMovies.csv', delimiter=',')

selected = frame.loc[:, ["Profitability", "RottenTomatoes"]]

plt.scatter(selected["Profitability"], selected["RottenTomatoes"])
plt.xlim(0, 200)
plt.show()
```

5. Perform Operations on Files

5.1: From the raw data below create a data frame

'first_name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],

'last_name': ['Miller', 'Jacobson', ".", 'Milner', 'Cooze'],

'age': [42, 52, 36, 24, 73],

'preTestScore': [4, 24, 31, ".", "."],

'postTestScore': ["25,000", "94,000", 57, 62, 70]

5.2: Save the dataframe into a csv file as example.csv

5.3: Read the example.csv and print the data frame

5.4: Read the example.csv without column heading

Question 5: Read the example.csv and make the index columns as 'First Name' and 'Last Name'

5.6: Print the data frame in a Boolean form as True or False. True for Null/ NaN values and false for non null values

5.7: Read the dataframe by skipping first 3 rows and print the data frame

5.8: Load a csv file while interpreting "," in strings around numbers as thousands seperators. Check the raw data 'postTestScore' column has , as thousands separator. Comma should be ignored while reading the data. It is default behaviour but you need to give argument to read_csv function which makes sure commas are ignored.

Solution

```
#Answer 5.1:
import pandas as pd
raw_data = {'first_name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
      'last_name': ['Miller', 'Jacobson', ".", 'Milner', 'Cooze'],
      'age': [42, 52, 36, 24, 73],
      'preTestScore': [4, 24, 31, ".", "."],
      'postTestScore': ["25,000", "94,000", 57, 62, 70]}
df = pd.DataFrame(raw_data, columns = ['first_name', 'last_name', 'age',
'preTestScore', 'postTestScore'])
print(df)
```

```
#Answer 5.2:
import pandas as pd
raw_data = {'first_name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
      'last_name': ['Miller', 'Jacobson', ".", 'Milner', 'Cooze'],
      'age': [42, 52, 36, 24, 73],
      'preTestScore': [4, 24, 31, ".", "."],
      'postTestScore': ["25,000", "94,000", 57, 62, 70]}
df = pd.DataFrame(raw_data, columns = ['first_name', 'last_name', 'age',
'preTestScore', 'postTestScore'])
df.to_csv('C:\\Users\\adhyapakss\\Desktop\\example.csv')

#Answer 5.3:
import pandas as pd
df = pd.read_csv('C:\\Users\\adhyapakss\\Desktop\\example.csv')
print(df)

#Answer 5.4:
import pandas as pd
df = pd.read_csv('C:\\Users\\adhyapakss\\Desktop\\example.csv',
header=None)
print(df)

#Answer 5.5:
import pandas as pd
df = pd.read_csv('C:\\Users\\adhyapakss\\Desktop\\example.csv',
index_col=['First Name', 'Last Name'], names=['UID', 'First Name', 'Last Name',
'Age', 'Pre-Test Score', 'Post-Test Score'])
print(df)

#Answer 5.6:
import pandas as pd
df = pd.read_csv('C:\\Users\\adhyapakss\\Desktop\\example.csv',
na_values=['.'])
print(pd.isnull(df))

#Answer 5.7:
import pandas as pd
df = pd.read_csv('C:\\Users\\adhyapakss\\Desktop\\example.csv', skiprows=3)
print(df)

#Answer 5.8:
import pandas as pd
df = pd.read_csv('C:\\Users\\adhyapakss\\Desktop\\example.csv',
thousands=',')
print(df)
```

6. Perform Operations on Files

6.1: From the raw data below create a Pandas Series

'Amit', **'Bob'**, **'Kate'**, **'A'**, **'b'**, np.nan, **'Car'**, **'dog'**, **'cat'**

a) Print all elements in lower case

b) Print all the elements in upper case

c) Print the length of all the elements

6.2: From the raw data below create a Pandas Series

' Atul', **'John '**, **' jack '**, **'Sam'**

a) Print all elements after stripping spaces from the left and right

b) Print all the elements after removing spaces from the left only

c) Print all the elements after removing spaces from the right only

6.3: - Create a series from the raw data below
'India_is_big', **'Population_is_huge'**, np.nan, **'Has_diverse_culture'**

a) split the individual strings wherever '_' comes and create a list out of it.
b) Access the individual element of a list
c) Expand the elements so that all individual elements get splitted by '_' and insted of list returns individual elements

6.4: Create a series and replace either X or dog with XX-XX
'A', **'B'**, **'C'**, **'AabX'**, **'BacX'**,**''**, np.nan, **'CABA'**, **'dog'**, **'cat'**

6.5: Create a series and remove dollar from the numeric values

'12', **'-$10'**, **'$10,000'**

6.6:- Create a series and reverse all lower case words

'india 1998', **'big country'**, np.nan

6.7: Create pandas series and print true if value is alphanumeric in series or false if value is not alpha numeric in series.

'1', **'2'**, **'1a'**, **'2b'**, **'2003c'**

6.8: Create pandas series and print true if value is containing 'A'

'1', **'2'**, **'1a'**, **'2b'**, **'America'**, **'VietnAm'**,**'vietnam'**, **'2003c'**

6.9: Create pandas series and print in three columns value 0 or 1 is a or b or c exists in values

'a', **'a|b'**, np.nan, **'a|c'**

6.10: Create pandas dataframe having keys and ltable and rtable as below -

'key': [**'One'**, **'Two'**], **'ltable'**: [1, 2]
'key': [**'One'**, **'Two'**], **'rtable'**: [4, 5]

Merge both the tables based of key

Solution

#Answer 6.1 (a)

```
import pandas as pd
import numpy as np
s = pd.Series(['Amit', 'Bob', 'Kate', 'A', 'b', np.nan, 'Car', 'dog', 'cat'])
print(s.str.lower())
```

#Answer 6.1(b)

```
import pandas as pd
import numpy as np
s = pd.Series(['Amit', 'Bob', 'Kate', 'A', 'b', np.nan, 'Car', 'dog', 'cat'])
print(s.str.upper())
```

#Answer 6.1(c):

```
import pandas as pd
import numpy as np
s = pd.Series(['Amit', 'Bob', 'Kate', 'A', 'b', np.nan, 'Car', 'dog', 'cat'])
print(s.str.len())
```

#Answer 6.2(a):

```
import pandas as pd
import numpy as np
```

```
s = pd.Index([' Atul', 'John ', ' jack ', 'Sam'])
print(s.str.strip())

#Answer 6.2(b):
import pandas as pd
import numpy as np
s = pd.Index([' Atul', 'John ', ' jack ', 'Sam'])
print(s.str.lstrip())


#Answer 6.2(c):
import pandas as pd
import numpy as np
s = pd.Index([' Atul', 'John ', ' jack ', 'Sam'])
print(s.str.rstrip())

#Answer 6.3(a):

import pandas as pd
import numpy as np
s = pd.Series(['India_is_big', 'Population_is_huge', np.nan, 'Has_diverse_culture'])
print(s.str.split('_'))

#Answer 6.3(b):

import pandas as pd
import numpy as np
s = pd.Series(['India_is_big', 'Population_is_huge', np.nan, 'Has_diverse_culture'])
print(s.str.split('_').str.get(1))


#Answer 6.3(c):
import pandas as pd
import numpy as np
s = pd.Series(['India_is_big', 'Population_is_huge', np.nan, 'Has_diverse_culture'])
print(s.str.split('_', expand=True))


#Answer 6.4

import pandas as pd
import numpy as np
s = pd.Series(['A', 'B', 'C', 'AabX', 'BacX','', np.nan, 'CABA', 'dog', 'cat'])
print(s.str.replace('^.a|dog', 'XX-XX ', case=False))
```

```
#Answer 6.5:
import pandas as pd
import numpy as np
d = pd.Series(['12', '-$10', '$10,000'])
print(d.str.replace('$', ''))

#Answer 6.6
import pandas as pd
import numpy as np
pattern = r'[a-z]+'
replacement = lambda m: m.group(0)[::-1]
s=pd.Series(['india 1998', 'big country', np.nan]).str.replace(pattern, replacement)
print(s)

#Answer 6.7:

import pandas as pd
pattern = r'[0-9][a-z]'
print(pd.Series(['1', '2', '1a', '2b', '2003c']).str.contains(pattern))

#Answer 6.8:
import pandas as pd
pattern = r'[0-9][a-z]'
print(pd.Series(['1', '2', '1a', '2b', 'America', 'VietnAm','vietnam',
'2003c']).str.contains('A', na=False))

#Answer 6.9:
import pandas as pd
import numpy as np
s = pd.Series(['a', 'a|b', np.nan, 'a|c'])
print(s.str.get_dummies(sep='|'))


#Answer 6.10:
import pandas as pd
left = pd.DataFrame({'key': ['One', 'Two'], 'ltable': [1, 2]})
right = pd.DataFrame({'key': ['One', 'Two'], 'rtable': [4, 5]})
new=pd.merge(left, right, on='key')
print(new)
```