codecademy

Aggregates in Pandas

This Is Jeopardy, Tasks

Data Analysis with Pandas

Alex Ricciardi

Date: 05/24/2020

This Is Jeopardy!

You will work to write several functions that investigate a dataset of Jeopardy! questions and answers. Filter the dataset for topics that you're interested in, compute the average difficulty of those questions, and train to become the next Jeopardy champion!

Project Tasks:

1. In order to complete this project, you should have completed the Pandas lessons in the Analyze Data with Python Skill Path.
You can also find those lessons in the Data Analysis with Pandas course.
Finally, the Practical Data Cleaning course may also be helpful.

2. We've provided a csv file containing data about the game show Jeopardy! in a file named jeopardy.csv.
Load the data into a DataFrame and investigate its contents. Try to print out specific columns. pages are on the CoolTShirts website?

3. Write a function that filters the dataset for questions that contains all of the words in a list of words.

4. Test your original function with a few different sets of words to try to find some ways your function breaks.
Edit your function so it is more robust.

5. We may want to eventually compute aggregate statistics, like .mean() on the " Value" column.
But right now, the values in that column are strings.
Convert the " Value" column to floats. If you'd like to, you can create a new column with the float values.

6. Write a function that returns the count of the unique answers to all of the questions in a dataset. many last touches on the purchase page is each campaign responsible for?

7. Explore from here! This is an incredibly rich dataset, and there are so many interesting things to discover.

1. Project Task:

Display on the console.

1.1 Description of the task

In order to display the full contents of a column, we've added this line of code to the top of your file:

```
pd.set_option('display.max_colwidth', -1)
```

1.2 Displaying the full contents of a column in the console.

Using the code

```
pd.set_option('display.max_colwidth', -1)
```

Resulted on a future warning on my PC's console:

```
FutureWarning: Passing a negative integer is deprecated in version 1.0 and will not be supported in future version.
Instead, use None to not limit the column width.
  pd.set_option('display.max_colwidth', -1)
```

I replaced the previous line code with the following code:

```
# display on the console
pd.set_option('display.width', 400)
pd.options.display.max_colwidth = 120 # Allows to fully read most jeopardy.Question columns values
pd.set_option('display.max_columns', 20)
pd.set_option('display.max_rows', 50)
```

2. Project Task:

Load the data into a DataFrame

2.1 Description of the task

We've provided a csv file containing data about the game show Jeopardy! in a file named jeopardy.csv. Load the data into a `DataFrame` and investigate its contents. Try to print out specific columns.

Note that in order to make this project as "real-world" as possible, we haven't modified the data at all—we're giving it to you exactly how we found it. As a result, this data isn't as "clean" as the datasets you normally find on Codecademy. More specifically, there's something odd about the column names. After you figure out the problem with the column names, you may want to rename them to make your life easier the rest of the project..

2.2 Import inspect with .head()

code

```
# Importing jeopardy data files
jeopardy = pd.read_csv("data/jeopardy.csv")
# Inspecting the first 5 row of the DataFrame jeopardy
print(jeopardy.head())
```

Console output

```
---------------------------------- Table -------------------------------------------------------------------------

   Show Number    Air Date      Round                       Category  Value                                                                                  Question      Answer
0         4680  2004-12-31  Jeopardy!                        HISTORY   $200            For the last 8 years of his life, Galileo was under house arrest for espousing this man's theory  Copernicus
1         4680  2004-12-31  Jeopardy!  ESPN's TOP 10 ALL-TIME ATHLETES   $200  No. 2: 1912 Olympian; football star at Carlisle Indian School; 6 MLB seasons with the Reds, Giants & Braves  Jim Thorpe
2         4680  2004-12-31  Jeopardy!      EVERYBODY TALKS ABOUT IT...   $200                     The city of Yuma in this state has a record average of 4,055 hours of sunshine each year      Arizona
3         4680  2004-12-31  Jeopardy!                THE COMPANY LINE   $200                         In 1963, live on "The Art Linkletter Show", this company served its billionth burger  McDonald's
4         4680  2004-12-31  Jeopardy!             EPITAPHS & TRIBUTES   $200    Signer of the Dec. of Indep., framer of the Constitution of Mass., second President of the United States  John Adams
```

2.3 Formatting columns name and inspecting data further

The columns names have a whitespace at the beginning of their string values.

The lambda function remove the whitespace removes the unwanted whitespace at the beginning of the column names and keep the column name ‘Show Number’ unchanged.

code

```
print('---------------------------------- Data Type -------------------------------')
print()
# Finding what is wrong with the columns names
print(jeopardy.dtypes)
print()
print('---------------------------------- Data type Formatted Columns Names --------')
print()
# Renaming the mis-formatted columns names
jeopardy = jeopardy.rename(columns=lambda column_name: column_name[1:] if column_name !=
'Show Number' else column_name)
print(jeopardy.dtypes)
print()
print('---------------------------------- Table Info -------------------------------')
print()
# Finding more (info) from the database jeopardy
print(jeopardy.info())
```

Console output

```
---------------------------------- Data Type -------------------------------

Show Number     int64
 Air Date      object
 Round         object
 Category      object
 Value         object
 Question      object
 Answer        object
dtype: object

---------------------------------- Formatted Columns Names ------------

Show Number     int64
Air Date       object
Round          object
Category       object
Value          object
Question       object
Answer         object
dtype: object

---------------------------------- Table Info -------------------------

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 216930 entries, 0 to 216929
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Show Number  216930 non-null  int64
 1   Air Date     216930 non-null  object
 2   Round        216930 non-null  object
 3   Category     216930 non-null  object
 4   Value        216930 non-null  object
 5   Question     216930 non-null  object
 6   Answer       216928 non-null  object
dtypes: int64(1), object(6)
memory usage: 6.6+ MB
None
```

2.4 Formatting Answer column values NaN

Two Answer column values are NaN data type, but they were meat to be string data type, with a string = ‘Null’. The pandas .fillna() function replaces the NaN data type with the string ‘Null’.

code

```
print('----------------------------------- Reformatted NaN rows ')
print()
# Reformmatting rows with a NaN Answer value
jeopardy = jeopardy.fillna(value={'Answer' : 'Null'})
# Checking the Reformmatting rows with a NaN Answer value
print(jeopardy.loc[[143297]])
print()
print('------------------------------------ Value Column --------------------------')
print()
# Inspecting the (value) column
print(pd.Series(jeopardy['Value'].unique()))
```

Console output

```
---------------------------------- NaN Answers meat to be the Null answer ----------------------------------

        Show Number    Air Date             Round       Category Value                                                                  Question Answer
94817          4346  2003-06-23         Jeopardy!  GOING "N"SANE  $200                                        It often precedes "and void"    NaN
143297         6177  2011-06-21  Double Jeopardy!        NOTHING  $400  This word for "nothing" precedes "and void" to mean "not valid"    NaN

---------------------------------- Reformatted NaN rows

        Show Number    Air Date             Round Category Value                                                                  Question Answer
143297         6177  2011-06-21  Double Jeopardy!  NOTHING  $400  This word for "nothing" precedes "and void" to mean "not valid"   Null
```

2.5 Removing the '$' sign character from Value

The Value column has '$' sign character in the front of its values, we need to tidy the data to covert the column values into integer data types.

The pandas function .replace() removed the '$' sign character.

code

```
print('------------------------------------ Value Column ------------------')
print()
# Inspecting the (value) column
print(pd.Series(jeopardy['Value'].unique()))
print()
print('------------------------------------ Reformatted Value Column values ---')
print()
# Reformatting the column (Value) by removing ($) and (,) turning the data value into integers
jeopardy.Value = jeopardy['Value'].replace('[\$,]', '', regex=True)
# Checking the reformatting the column (Value) results
print(pd.Series(jeopardy['Value'].unique()))
```

Console output

```
------------------------------------ Value Column ----------

0         $200
1         $400
2         $600
3         $800
4       $2,000

---------------------------------- Reformmatted Value Column values --------

0       200
1       400
2       600
3       800
4      2000
```

2.6 Formatting the Value column values to integer

code

```
print('---------------------------------- Reformatted Value Column values (none) -------')
print()
# Replacing the (None) values with (0), the None values were inputted when players placed wagers during a Tiebreaker a Final Jeopardy round
jeopardy.Value = jeopardy['Value'].replace('None', '0')
print(jeopardy.Value.loc[[55]])
print()
print()
print('---------------------------------- Converted Value Column data type to int64 ------')
print()
jeopardy.Value = pd.to_numeric(jeopardy.Value)
print(jeopardy.Value.dtypes)
```

Console output

```
---------------------------------- Reformatted Value Column values (none) -------------

55    0
Name: Value, dtype: object


---------------------------------- Converted Value Column data type to int64 ----------

int64
```

Some Value column valuesare equal to ‘None’,the pandas .replace()function replaces it with the string data type ‘0’.

And the pandas to_numeric() function converts the values from string data type to integer data type

2.7 Formatting the Air Date column values to datetime data type

The [‘Air Time’] column values are (object) string data type, we need to tidy the data, the values need to be datetime data type values.
The pandas to_datetime() function converts the values from string data type to datetime data type.

code

```
print('------------------------------------ Air Date Column --------------------')
print()
# Inspecting the (value) column
print(pd.Series(jeopardy['Air Date'].unique()))
print()
print('---------------------------------- Reformatted Air Date data type  ------')
print()
print()
jeopardy['Air Date'] = pd.to_datetime(jeopardy['Air Date'])
print(jeopardy['Air Date'].head())
```

Console output

```
------------------------------------ Air Date Column --------------------

0    2004-12-31
1    2004-12-31
2    2004-12-31
3    2004-12-31
4    2004-12-31
Name: Air Date, dtype: object

---------------------------------- Reformatted Air Date data type  ---

0   2004-12-31
1   2004-12-31
2   2004-12-31
3   2004-12-31
4   2004-12-31
Name: Air Date, dtype: datetime64[ns]
```

2.8 Number of unique categories and questions

code

```
print('---------------------------------- Numbers of categories ----------------------')
print()
print(len(jeopardy['Category'].unique()))
print()
print('---------------------------------- Numbers of questions ----------------------')
print()
print(len(jeopardy['Question'].unique()))
```

Console output

```
---------------------------------- Numbers of categories -------

27995

---------------------------------- Numbers of questions --------

216124
```

The data base has an impressive amount of categories and questions

2.9 Formatting Question column values with hyperlinks

Some Question column values have hyperlinks attached to it.

The pandas .replace()function with regex code removes the hyperlinks strings.

code

```
print('----------------------------------- Reformatted question column with <a > values ----------------------')
print()
print(jeopardy.loc[[51115]])
# Removing Hyperlinks
jeopardy.Question = jeopardy['Question'].replace('(\(?<.*>\.?\)?)', '', regex=True)
print()
print(jeopardy.loc[[51115]])
print
```

Console output

```
---------------------------------- Reformatted question column with <a > values ----------------------------

      Show Number   Air Date      Round           Category  Value                                                                                                Question       Answer
51115        3574 2000-03-02  Jeopardy!  ALL THINGS BRITISH    200  (<a href="http://www.j-archive.com/media/2000-03-02_J_22.jpg" target="_blank">Alex Trebek reads from England.</a>)  ...  Magna Carta

      Show Number   Air Date      Round           Category  Value                                                                                                Question       Answer
51115        3574 2000-03-02  Jeopardy!  ALL THINGS BRITISH    200    In 1214 barons fed up with King John met at the abbey whose ruins we see here; the result a year later was this ch...  Magna Carta
```

2.10 List of rounds

What are the name of rounds in the Jeopardy game?

code

```
print('------------------------------------ list of Rounds ------------------------------------')
print()
print(jeopardy['Round'].unique())
```

Console output

```
------------------------------------ list of Rounds   ---------------------------

['Jeopardy!' 'Double Jeopardy!' 'Final Jeopardy!' 'Tiebreaker']
```

3. Project Task:

Question Filtering function

3.1 Description of the task

Write a function that filters the dataset for questions that contains all the words in a list of words.

For example,
when the list ["King", "England"] was passed to our function, the function returned a DataFrame of 152 rows.
Every row had the strings "King" and "England" somewhere in its " Question"

3.2.0 The filter function code

code

```
print('-------------------------------------------------------------------------------------------------------------------------------')
print("                                                ------ Task.3 -------")
print('-------------------------------------------------------------------------------------------------------------------------------')
print()
# Function that filters the data set for questions containing all the words in a list
def filter_questions_data(df, words):
    # Function that filters the word list for a question
    def filter_word_list(question, words):
        words_in_question = True # Boolean to check if all the words are in the question
        question = question.lower()
        for word in words: # loops through the word list
            if (word.lower() in question) == False: # Check if the word is in the question
                words_in_question = False
        return words_in_question
    # The lambda function filters the dataset using the filter_word_list function
    filter_datset = lambda x: filter_word_list(x, words)
    # df.loc returns the dataframe rows where the questions are located
    return df.loc[df['Question'].apply(filter_datset)]

filter_questions_df = filter_questions_data(jeopardy, ["kinG", "England"])
print(filter_questions_df)
print()
print(filter_questions_df.Question.loc[[86353]])
print()
```

3.2.1 The filter function console output

Note the ‘king’ string in the word ‘taking’.

Console output

```
----------------------------------------------------------------------------------------------------
                                    ------ Task.3 -------
----------------------------------------------------------------------------------------------------

       Show Number   Air Date            Round              Category  Value                                                                                Question                      Answer
4953          3003 1997-09-24  Double Jeopardy!          "PH"UN WORDS    200                Both England's King George V & FDR put their stamp of approval on this "King of Hobbies"  Philately (stamp collecting)
6337          3517 1999-12-14  Double Jeopardy!                   Y1K    800  In retaliation for Viking raids, this "Unready" king of England attacks Norse areas of the Isle of Man                     Ethelred
9191          3907 2001-09-04  Double Jeopardy!        WON THE BATTLE    800                This king of England beat the odds to trounce the French in the 1415 Battle of Agincourt                     Henry V
11710         2903 1997-03-26  Double Jeopardy!      BRITISH MONARCHS    600             This Scotsman, the first Stuart king of England, was called "The Wisest Fool in Christendom"                     James I
13454         4726 2005-03-07         Jeopardy!  A NUMBER FROM 1 TO 10   1000                                    It's the number that followed the last king of England named William                           4
...            ...        ...               ...                   ...    ...                                                                                   ...                         ...
208295        4621 2004-10-11         Jeopardy!           THE VIKINGS    600      In 1066 this great-great grandson of Rollo made what some call the last Viking invasion of England          William the Conqueror
208742        4863 2005-11-02  Double Jeopardy!        BEFORE & AFTER   3000                   Dutch-born king who ruled England jointly with Mary II & is a tasty New Zealand fish       William of Orange roughy
213870        5856 2010-02-15  Double Jeopardy!               URANUS   1600              In 1781 William Herschel discovered Uranus & initially named it after this king of England                  George III
216021        1881 1992-11-09  Double Jeopardy!        HISTORIC NAMES   1000         His nickname was "Bertie", but he used this name & number when he became king of England in 1901                 Edward VII
216789        5070 2006-09-29  Double Jeopardy!       ANCIENT HISTORY   1200      This kingdom of England grew from 2 settlements, one founded around 495 by Cerdic & his son Cynric                      Wessex

[148 rows x 7 columns]

86353    The taking of England by William in 1066 is known as this "Conquest"
Name: Question, dtype: object

129106    King Edward I of England, who fought William Wallace, had this nickname relating to his height
Name: Question, dtype: object
```

4. Project Task:

Edit your function so it is more robust.

4.1 Description of the task

Test your original function with a few different sets of words to try to find some ways your function breaks. Edit your function so it is more robust.

For example, think about capitalization. We probably want to find questions that contain the word "King" or "king".

You may also want to check to make sure you don’t find rows that contain substrings of your given words. For example, our function found a question that didn’t contain the word "king", however it did contain the word "viking"—it found the "king" inside "viking". Note that this also comes with some drawbacks—you would no longer find questions that contained words like "England's".

4.2 The filter function with a more robust code

To make the function more robust I used the functions compile() and search() from the regex library.
The pandas .apply() function combined with the lambda function filter_datset loops though the values of the Question column.

code

```
# Function that filters the dataset for questions containing all the words in a list
def regex_filter_questions_data(df, words):
    # Function that filters the word list for a question
    def filter_word_list(question, words):
        words_in_question = True # Boolean to check if all the words are in the question
        question = question.lower()
        for word in words: # loops through the word list
            word = word.lower()
            word = re.compile(r'(\A|\s|\'|\"|\( re.IGNORECASE)' + word + r"([,]|\.|\'|\"|\)|s|\s|\Z|\W, re.IGNORECASE)")
            if re.search(word, question) == None: # Check if the word is in the question
                words_in_question = False # Set the boolean to False if word is not in question
        return words_in_question
    # The lambda function filters the dataset using the filter_word_list function
    filter_datset = lambda question: filter_word_list(question, words)
    # df.loc returns the dataframe rows where the questions are located
    return df[df['Question'].apply(filter_datset)]
```

4.3 The function check for the words and not the strings

I use pandas the .loc method with the row index to target specifics Question column values.

code

```
print('---------------------------- Checking if the Function check for the words and not the strings --------------------')
print()
print(filter_questions_df.Question.loc[[86353]]) # Has the word england and taking
regex_filter_questions_df = regex_filter_questions_data(jeopardy, ["King", "England"])
print(filter_questions_df.Question.loc[[86353]].isin(regex_filter_questions_df['Question'])) # output false
```

Console output

```
---------------------------- Checking if the Function check for the words and not the strings --------------------

86353    The taking of England by William in 1066 is known as this "Conquest"
Name: Question, dtype: object
86353    False
Name: Question, dtype: bool
```

4.2 Checking the function with different size of words list

code

```
print()
print('---------------------------- Checking function with length = 1 word list ---------------------------------------')
print()
print(regex_filter_questions_data(jeopardy, ["weed"]))
print()
print('---------------------------- Checking function with length = 3 words list ---------------------------------------')
print()
print(regex_filter_questions_data(jeopardy, ["weed", "CrEw", "great"]))
```

Console output

---------------------------- Checking function with length = 1 word list ---------------------------------------

	Show Number	Air Date	Round	Category	Value	Question	Answer
956	3619	2000-05-04	Jeopardy!	WEEDS	200	This common weed seen here has a beverage in its name	Milkweed
968	3619	2000-05-04	Jeopardy!	WEEDS	400	This fabric follows Queen Anne's in the name of the weed seen here	Lace
974	3619	2000-05-04	Jeopardy!	WEEDS	500	In the names of weeds, this old word for a plant follows soap- & St. John's	Wort
5840	3470	1999-10-08	Double Jeopardy!	BOTANY	200	To the horror of homeowners, this lawn weed, taraxacum officinale, can grow 1 1/2' high	Dandelion
6314	3517	1999-12-14	Jeopardy!	GOOSE...MOTHER GOOSE	500	"A man of words and not of" these "is like a garden full of weeds"	Deeds
7108	5171	2007-02-19	Jeopardy!	BOTANY	200	The flowers of this lawn weed, Taraxacum oficinale, are sometimes used to make wine	dandelions
12556	4798	2005-06-15	Jeopardy!	DO US A FLAVOR	400	Its seeds are used in curing gherkins; the leaves, or weeds, are used on salads & meats	dill
35544	4631	2004-10-25	Double Jeopardy!	CROSSWORD CLUES "C"	1600	Cranky lawn weed (9)	crabgrass
36640	734	1987-11-12	Jeopardy!	BOTANY	300	The floss of this weed, named for the white liquid in its stems, was used in lifebelts in World War II	milkweed
44711	4141	2002-09-09	Double Jeopardy!	BOTANY	400	The flowers of this yellow lawn weed are sometimes used to make wine	dandelion

---------------------------- Checking function with length = 3 words list ---------------------------------------

	Show Number	Air Date	Round	Category	Value	Question	Answer
115533	5711	2009-06-08	Double Jeopardy!	POETRY IN EARLY AMERICA	4000	An anonymous poet addressed this man as "great patron of the sailing crew / Who gav'st us weed to smoke and chew"	Sir Walter Raleigh

5. Project Task:

Edit your function so it is more robust.

5.1 Description of the task

We may want to eventually compute aggregate statistics, like .mean() on the " Value" column. But right now, the values in that column are strings. Convert the " Value" column to floats. If you'd like to, you can create a new column with the float values.

Now that you can filter the dataset of question, use your new column that contains the float values of each question to find the "difficulty" of certain topics. For example, what is the average value of questions that contain the word "King"?

5.2 Value column values to floats, mean its values

code

```
print('-------------------------------------------------------------------------------------------------------------------------------')
print("                                                ------ Task.5 -------")
print('-------------------------------------------------------------------------------------------------------------------------------')
print()
jeopardy.Value = pd.to_numeric(jeopardy.Value, downcast='float')
print()
print('---------- Value column data type----------------------')
print()
print(jeopardy.Value.dtypes)
print()
print('------------------------------------ average value of questions that contain the word "King" --------------------')
print()
filter_questions_king = regex_filter_questions_data(jeopardy, ["King"])
print(filter_questions_king.Value.mean())
```

Console output

```
-------------------------------------------------------------------------------------------------------------------------------
                                          ------ Task.5 -------
-------------------------------------------------------------------------------------------------------------------------------


--------- Value column data type----------------------

float32

----------------------------------- average value of questions that contain the word "King" -------------------

800.64435
```

6. Project Task:

Write a function that returns the count of the unique answers.

6.1 Description of the task

Write a function that returns the count of the unique answers to all of the questions in a dataset. For example, after filtering the entire dataset to only questions containing the word "King", we could then find all of the unique answers to those questions. The answer “Henry VIII” appeared 3 times and was the most common answer.

6.2 Function returns the count of the unique answers

The function returns the count of the unique answers to all of the questions in a dataset

code

```
print('-------------------------------------------------------------------------------------------------------------------------')
print("                                                ------ Task.6 -------")
print('-------------------------------------------------------------------------------------------------------------------------')
print()
def get_answers(word_list):
    df = regex_filter_questions_data(jeopardy, word_list)
    return df.Answer.value_counts()
#
print(get_answers(['King']))
```

Console output

```
-------------------------------------------------------------------------------------------------------------------------
                                                ------ Task.6 -------
-------------------------------------------------------------------------------------------------------------------------

Henry VIII                  53
Solomon                     34
Louis XIV                   31
David                       29
Richard III                 25
```

7. Project Task:

Explore from here!

7.1 Description of the task

Explore from here! This is an incredibly rich dataset, and there are so many interesting things to discover. There are a few columns that we haven’t even started looking at yet. Here are some ideas on ways to continue working with this data:

- Investigate the ways in which questions change over time by filtering by the date. How many questions from the 90s use the word "Computer" compared to questions from the 2000s?

- Is there a connection between the round and the category? Are you more likely to find certain categories, like "Literature" in Single Jeopardy or Double Jeopardy?

- Build a system to quiz yourself. Grab random questions, and use the input function to get a response from the user. Check to see if that response was right or wrong. Note that you can’t do this on the Codecademy platform —to do this, download the data, and write and run the code on your own computer!

7.2 Compare the numbers of questions that use the word "Computer“ from the 90s to the ones from the 2000s

Note that use the regex_filter_questions_data() function from Task -5, and I created a newDataFrame to store the numbers of questions per decade that the word “Computer” in it

code

```
# How many questions from the 90s use the word "Computer" compared to questions from the 2000s?
filter_questions_pc = regex_filter_questions_data(jeopardy, ["Computer"])
# Number of questions per decade
# 1990s
pc_1990_questions = filter_questions_pc.\
    loc[(filter_questions_pc['Air Date'] >= '01-01-1990') & (filter_questions_pc['Air Date'] <= '12-31-1999')]

num_pc_1990_unique_questions = len(pc_1990_questions.Question.unique())
# 2000s
pc_2000_questions = filter_questions_pc.\
  loc[(filter_questions_pc['Air Date'] >= '01-01-2000') & (filter_questions_pc['Air Date'] <= '12-31-2009')]

num_pc_2000_unique_questions = len(pc_2000_questions.Question.unique())
# Data Frame storing the number of question with the word (computer) in it
num_pc_questions = pd.DataFrame(
    { 'Decade': ['1990s', '2000s'],
      'Numbers of Question':[num_pc_1990_unique_questions, num_pc_2000_unique_questions]
     })
print(num_pc_questions
```

Console output

```
------------------------------------------------------------------------------------------------------------------------
                                                ------ Task.7 -------
------------------------------------------------------------------------------------------------------------------------

----------------------------------- How many questions from the 90s -----------------------------------

  Decade  Numbers of Question
0  1990s                  83
1  2000s                 238
```

7.3.0 Is there a connection between the Round and the Category?

code

```
print('--------------- Is there a connection between the Round and the Category -----')
print()
# Creating a Data Frame with Round and Category
round_category_value = jeopardy.groupby(['Round', 'Category']).Value.mean().reset_index()
# Category as rows and Round as columns number of questions per Round/Category as values
round_category = round_category_value.pivot(
    index='Category',
    columns='Round',
    values='Value'
).reset_index()
# Reordering the columns
round_category = round_category[['Category', 'Tiebreaker', 'Final Jeopardy!', 'Double Jeopardy!', 'Jeopardy!']]
# Output
print(round_category)
```

Console output

```
---------------------------------- Is there a connection between the Round and the Category -------

Round                      Category  Tiebreaker  Final Jeopardy!  Double Jeopardy!  Jeopardy!
0       A JIM CARREY FILM FESTIVAL         NaN              NaN               NaN      600.0
1                              "!"         NaN              NaN               NaN      300.0
2                         "-ARES"          NaN              NaN            1200.0        NaN
3               "-ICIAN" EXPEDITION         NaN              NaN               NaN      600.0
4                    "...OD" WORDS         NaN              NaN             600.0        NaN
...                            ...         ...              ...               ...        ...
27990                   “R” MOVIES         NaN              NaN             600.0        NaN
27991                    “SAINTS”          NaN              NaN             650.0        NaN
27992                    “SOUTH”           NaN              NaN             600.0        NaN
27993                   “STREETS”          NaN              NaN               NaN      340.0
27994                 “WH”AT IS IT?        NaN              NaN             520.0        NaN

[27995 rows x 5 columns]
```

I use the pandas.mean() function to compute the average value of the question per rounds. Saved the results in a pivoted DataFrame with the column names:
['Category', 'Tiebreaker', 'Final Jeopardy!', 'Double Jeopardy!', 'Jeopardy!']]

Rounds with a category having an average question value result of NaN, are rounds where that particular category.

Note that the number of rows in the DataFrame is equal to 27,995, meaning that is a total of 27,995 ‘unique’ categories in the data.

7.3.0 Is there a connection between the Round and the Category?

code

```
print('--------------- Is there a connection between the Round and the Category -----')
print()
# Creating a Data Frame with Round and Category
round_category_value = jeopardy.groupby(['Round', 'Category']).Value.mean().reset_index()
# Category as rows and Round as columns number of questions per Round/Category as values
round_category = round_category_value.pivot(
    index='Category',
    columns='Round',
    values='Value'
).reset_index()
# Reordering the columns
round_category = round_category[['Category', 'Tiebreaker', 'Final Jeopardy!', 'Double Jeopardy!', 'Jeopardy!']]
# Output
print(round_category)
```

Console output

```
---------------------------------- Is there a connection between the Round and the Category -------

Round                      Category  Tiebreaker  Final Jeopardy!  Double Jeopardy!  Jeopardy!
0       A JIM CARREY FILM FESTIVAL         NaN              NaN               NaN      600.0
1                              "!"         NaN              NaN               NaN      300.0
2                         "-ARES"          NaN              NaN            1200.0        NaN
3               "-ICIAN" EXPEDITION         NaN              NaN               NaN      600.0
4                    "...OD" WORDS         NaN              NaN             600.0        NaN
...                            ...         ...              ...               ...        ...
27990                   “R” MOVIES         NaN              NaN             600.0        NaN
27991                    “SAINTS”          NaN              NaN             650.0        NaN
27992                    “SOUTH”           NaN              NaN             600.0        NaN
27993                   “STREETS”          NaN              NaN               NaN      340.0
27994                 “WH”AT IS IT?        NaN              NaN             520.0        NaN

[27995 rows x 5 columns]
```

I use the pandas.mean() function to compute the average value of the question per rounds. Saved the results in a pivoted DataFrame with the column names:
['Category', 'Tiebreaker', 'Final Jeopardy!', 'Double Jeopardy!', 'Jeopardy!']]

Rounds with a category having an average question value result of NaN, are rounds where that particular category.

Note that the number of rows in the DataFrame is equal to 27,995, meaning that is a total of 27,995 ‘unique’ categories in the data.

7.3.1 Looking if same categories are used is all 4 rounds

The Tiebreaker and Final Jeopardy! rounds are wager rounds, the player need to place a wager instead of using the set value for the question. In Task-2.6, I reformatted the string 'None' value to a numeric value equal to 0.
The console results output shows that the category 'LITERARY CHARACTERS' was used in all 4 rounds, it is safe to assume that is not a real connection between categories and rounds.
The console results output shows the average question value in the Double Jeopardy! round is double than the Jeopardy! round average question value, it is safe to assume that is connection between the question values and the rounds.
In the televised Jeopardy game show during the Double Jeopardy! round the question values are double from the Jeopardy! Round.

code

```
# removing the the DatFrame name (Round)
round_category.columns.name = ""
# Looking if a Category is used is all 4 rounds
print()
print(round_category.loc[(round_category['Final Jeopardy!'].notna()) & \
                         (round_category['Double Jeopardy!']).notna() & \
                         (round_category['Jeopardy!']).notna() & \
                         (round_category['Tiebreaker']).notna()])
```

Console output

```
                Category  Tiebreaker  Final Jeopardy!  Double Jeopardy!  Jeopardy!
14366  LITERARY CHARACTERS         0.0              0.0       1032.727295      548.0
```

7.4 Build a system to quiz yourself.

I made a Pandas library base one playerJeopardy console game featuring user input with getch(), error handling, tidying data and data manipulation.

Game description:
The Gameplay consists of a questions/clues quiz comprising of 3 round.

- The clues in the quiz are presented as "answers" and responses must be phrased in the form of a question..
- Round-1 Jeopardy: 2 categories 2 question.
- Round-2 Double Jeopardy: 2 categories 2 question, the question values are double.
- Round-3 Final Jeopardy: 1 question wager.
- To move from round-1 to round-2 and from round- to round-3, all the clues in all the categories in the round, have to be answered, and your winnings can not be $0 or less.
- The game feature a settings options where the number of categories and the number of clues per category can be change up to 4 categories and 4 clues per category.
- The game also feature a cheat mode when activated will display the response to the clues.

For more Information:
Power point presentation: https://1drv.ms/p/s!AsKPX_vZuHCqg6ZXq7mExCxvvdfMog?e=8V3jj1
GitHub: https://github.com/ARiccGitHub/jeoprady_game_console