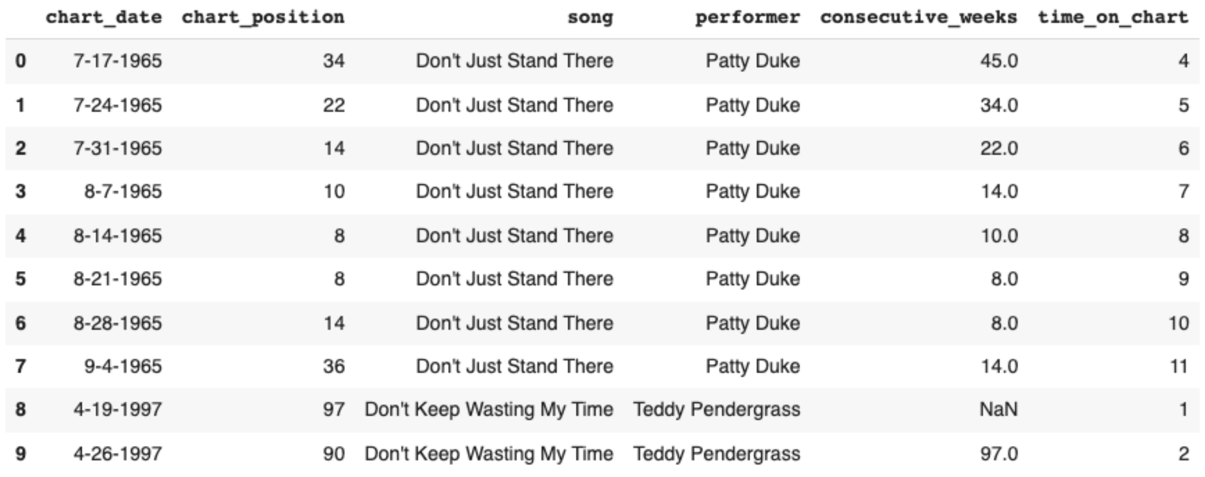
Data 100, Spring 2023

Exam Prep Section #2 Solutions

Cleaning Up the Billboard Hot 100

1. Onika Tanya Maraj (better known by her stage name Nicki Minaj) knows she's the baddest alive, but nonetheless, she wants to learn more about the popularity of her greatest hits. She obtains a dataset of entries from the Billboard Hot 100, an American music chart updated weekly containing the 100 most popular songs of that week. The dataset has 100 entries per week (corresponding to each song on the chart for that particular week), for every week from 1958 until today in 2022. As of 2018, rank is determined by a specific combination of physical/digital sales, radio play, and online streaming. However, the dataset is large and messy, so she enlists you, a Data 100 student, as a consultant to help her clean and understand the data. You load the data into a Pandas DataFrame called `billboard`. A call to `billboard.head(10)` is shown below:



	chart_date	chart_position	song	performer	consecutive_weeks	time_on_chart
0	7-17-1965	34	Don't Just Stand There	Patty Duke	45.0	4
1	7-24-1965	22	Don't Just Stand There	Patty Duke	34.0	5
2	7-31-1965	14	Don't Just Stand There	Patty Duke	22.0	6
3	8-7-1965	10	Don't Just Stand There	Patty Duke	14.0	7
4	8-14-1965	8	Don't Just Stand There	Patty Duke	10.0	8
5	8-21-1965	8	Don't Just Stand There	Patty Duke	8.0	9
6	8-28-1965	14	Don't Just Stand There	Patty Duke	8.0	10
7	9-4-1965	36	Don't Just Stand There	Patty Duke	14.0	11
8	4-19-1997	97	Don't Keep Wasting My Time	Teddy Pendergrass	NaN	1
9	4-26-1997	90	Don't Keep Wasting My Time	Teddy Pendergrass	97.0	2

(a) As with any good EDA, you try to understand the variables included. Of the variables `song`, `performer`, `time_on_chart`, and `consecutive_weeks`, which variables are quantitative and which are categorical/qualitative? Of the categorical/qualitative variable(s), which are nominal and which are ordinal (if any)? Of the quantitative variable(s), which are continuous and which are discrete (if any)?

Solution: `song` and `performer` are both qualitative/categorical nominal variables. `time_on_chart` and `consecutive_weeks` are discrete quantitative variables.

(b) `chart_position` and `chart_date` are a little bit harder to categorize, and largely depend on what we are doing with the dataset. How would you categorize them, according to the last part? Why?

Solution: I would categorize `chart_position` as an ordinal categorical variable, since there are a finite number of categories (100), and rank is not really a quantitative measure. I would categorize `chart_date` as a discrete quantitative variable, since the chart date is essentially measuring the week number from the start of the charts, and assuming the chart continues to collect data, are not finitely-valued. Additionally, the meaning of a distance between adjacent values of week number is constant, since Week 1 is always 1 week apart from Week 2, and Week 100 is always 1 week apart from Week 1, etc. However, there is no set answer for this.

(c) Since Onika only wants an assessment of her music, you need to obtain a DataFrame of only her Billboard Hot 100 entries. Write a line of code to accomplish this (be sure to use her stage name "Nicki Minaj"). Store your result in a variable called `nicki`.

Solution:

```
nicki = billboard[billboard['performer'] == 'Nicki Minaj']
```

(d) In this new `nicki` DataFrame, are there any missing values in any columns? Write a line of Pandas code to return a Series whose index is the column name and whose values are a boolean indicating if that column has any null values (the `DataFrame.any()` function may be useful).

Solution:

```
nicki.isnull().any()
```

(e) Assume all columns do not have missing values, except `consecutive_weeks`, which we can see in the above image. Why do you think that value is marked missing? Would you impute anything for it? Give justification for your answer.

Solution: The value is marked missing because when a song first enters the chart on a particular week, asking how many consecutive weeks it has been on the chart is nonsensical. It is reasonable to impute `0` for these entries, since if it's your first week on the chart, you have no consecutive weeks on the chart.

Additionally, when looking at this `consecutive_weeks` column, we are probably never going to aggregate over the values in a way that would bias a calculation. For instance, taking the mean of this column has no real interpretation. We may eventually want the the max number, but imputing 0s will not affect this.

(f) Onika wants to know what her most successful year was, where success is measured by the number of Billboard Hot 100 entries she had across all weeks of a given year. Write two lines of `Pandas` code to return a Series whose index is the year, and whose values are the number of entries. *(Hint: you might want to add a column corresponding to the year of each observation, and `Series.str.split('-')` is your friend!)*

Solution:

```
nicki.loc[:, 'year']= nicki['chart_date'].str.split('-').str[2]
nicki.groupby('year')['song'].count()
```

(g) You find her most successful year was 2012. Should you conclude that her music was most globally popular in 2012, but that she has since fallen off? Present two reasons in favor or against this conclusion. (*Hint: refer to the intro*).

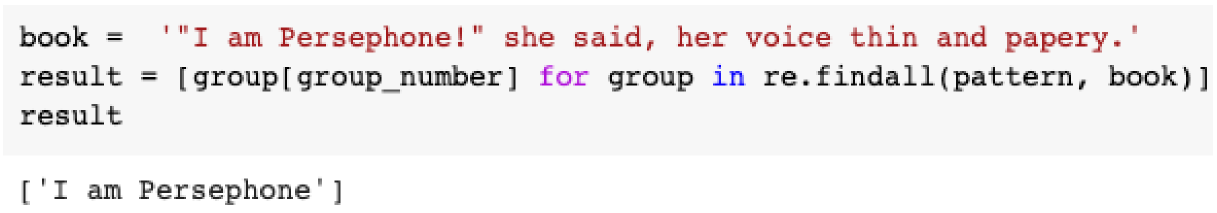
Solution: This is not a reasonable conclusion. First, this is data only based on listeners in the US, so it should say nothing about global popularity (the data here is not scoped properly for inferences about the world's music taste). Additionally, as mentioned in the intro, Billboard regularly changes the way they compile information to obtain popularity ranks. So, a rank in one year may not be computed using the same standards as another year, challenging our ability to fairly and reasonably compare ranks across years. Another reasonable answer is that being on the Billboard Hot 100 does not necessarily mean you are less popular in an absolute sense.

Percy Jackson

2. Morgan is reading all the Percy Jackson books in one setting, and she wants to use REGEX to find out some interesting facts about the books! Assume that a newline denotes the start of a new paragraph and that any standard punctuation such as a period, question mark or exclamation denotes the end of a sentence.

   Assume the entirety of the book, including newlines, are stored in a string `book` - this would imply the string begins in chapter 1 and ends at the end of the book.

   (a) Define an exclamation as any quoted sequence of three words or fewer that ends with an exclamation point. In other words, an exclamation should begin with a quotation mark, include 0-3 words followed by an exclamation point, then end with a quotation mark. Using REGEX, write Python code to capture all exclamations (without the exclamation mark itself) and assign the result to the variable `result`. See the example below:



```
book =  '"I am Persephone!" she said, her voice thin and papery.'
result = [group[group_number] for group in re.findall(pattern, book)]
result
```

```
['I am Persephone']
```

```
pattern = r‘____________________’
group_number = ________
result = [group[group_number] for group in re.findall(pattern, book)]
```

Solution:

```
pattern = r‘"(([A-Za-z]+\s*){0,3})!"’
group_number = 0
result = [group[group_number] for group in \
            re.findall(pattern, book)]
```

`[A-Za-z]+` matches words, `[A-Za-z]+\s*` matches words and the spaces that follow them, if any (note, if a word ends a sentence, it is not followed by a space but nonetheless should be included in our exclamation, so we use `*` to have the option of being followed by a space or not). Note that we want spaces because we eventually want to capture the entire exclamation. `([A-Za-z]+\s*){0, 3}` matches at most 3 of these words consecutively, the capture `(([A-Za-z]+\s*){0, 3})` will capture this group of at most 3 consecutive words, and `(([A-Za-z]+\s*){0, 3})!` will capture this group only if it's followed by an exclamation point, as desired. Note that a call to `re.findall()`

will capture the outer group and the last instance of the inner group, and put them in a tuple. The outer group capture always comes first, so we index by 0.

(b) Using REGEX, write Python code to capture all sentences that constitute an entire paragraph and assign the result to the variable `result`.

```
pattern = r '___________________'
result = _______________________
```

Solution:

```
pattern = r'([^\n]+)\n*'
result = re.findall(pattern, book)
```

`([ ^\n]+)` captures a sequence of any character that's not a newline character (one or more). `([^\n]+)\n*` captures these sequences until they end in one or more new lines. This ensures the last sentence is also captured, since it may not be followed by a new line.

(c) Using REGEX, write Python code to return a list of the first word of each sentence that contains the word "prophecy" at least once. You can ignore sentences which start with the word "prophecy."

Solution:

```
pattern = r'([A-Za-z]+)[^.!?]+prophecy[^.!?]*[.!?]'
re.findall(pattern, book)
```

`([A-Za-z]+)` captures the first word of a sentence by matching any sequence of letters that extends for one character or longer. `[^.!?] (read: not a ., !, or ?)`defines a character class that will match any character other than endofsentence punctuation, which ensures that the pattern will not extend past a single sentence. `[^.!?]+` matches any words or whitespace that follow the first word of the sentence and come before the "prophecy." prophecy matches the word "prophecy" directly, after which `[^.!?]*` will match any remaining words or whitespace before the end of the sentence. `*` is used here to account for the case where "prophecy" is the final word in the sentence and is immediately followed by end-of-sentence punctuation. Finally, `[.!?]` defines a character class for any punctuation that indicates the end of the sentence.

(d) Suppose we wish to extract the most common name(s) preceding a question mark. We do this first by extracting all words that begin with capital letters that precede a question mark. Which of the following REGEX patterns will accomplish this first goal? Select all that apply.

◯ A. `r'([A-Z]+\w+)?'`

◯ B. `r'([A-Z]\w)+?'`

◯ C. `r‘([A-Z]\w+)\?’`

◯ D. `r‘([A-Z]\w)+\?’`