Quiz 05: Spark APIs [100 points]

Accumulators [10 points]

1. [10 points].The title of this Q&A is wrong. It's really about global variables (aka accumulators). The question shows code that is *incorrect*.

```
val data = Array(1,2,3,4,5)
var counter = 0
var rdd = sc.parallelize(data)

// Wrong: Don't do this!!
rdd.foreach(x => counter += x)

println("Counter value: " + counter)
```

   Write a corrected version of the code and demonstrate its intended operation.

Airline Traffic [45 points]

Ontime statistics for domestic airlines are published by the Bureau of Transportation Statistics[1][2]. The schema is here[3], but the actual data has 4 additional columns (between B. and C.) which are not documented and may be safely deleted for the purpose of this exercise.

Based on the statistics for June 2024 and July 2024, please report on

1. [15 points] Describe in words and in code (where applicable) the steps you took to set up the environment for gathering the statistical data in the below questions.
2. [6 points] Which US Airline Has the Least Delays[4]? Report by full names, (e.g., Delta Airlines, not DL)
3. [6 points] What Departure Time of Day Is Best to Avoid Flight Delays[5], segmented into 5 time blocks [night (10 pm - 6 am), morning (6 am to 10 am), mid-day (10 am to 2 pm), afternoon (2 pm - 6 pm), evening (6 pm - 10 pm)]
4. [5 points] Which Airports Have The Most Flight Delays[6]? Report by full name, (e.g., “Newark Liberty International,” not “EWR,” when the airport code EWR is provided).
5. [5 points] What Are the Top 5 Busiest Airports in the US[7]. Report by full name, (e.g., “Newark Liberty International,” not “EWR”).

[1] BTS Data Schema
[2] BTS: Ontime stats
[3] The list of airport codes in the document is incomplete. You are not responsible for filling in the table by looking up the remaining airport codes.
[4] Mean delay of all flights for that airline.
[5] Mean delay for all flights in that time-block.
[6] As measured by arrival delays (column R) plus departure delays (column Q).
[7] In terms of total arrivals and departures, not total passengers.

ShortStoryJam [45 pts]

ShortStoryJam is a proposed new business for users to upload their short stories. We wish to set up a framework for analyzing an arbitrarily large number of stories. We would like to be able to deploy hundreds of servers to analyze different stories in parallel.

1. [3 points] To seed the effort, the text of about 22 short stories by Edgar Allan Poe[8], he of the "quoth the raven" fame, are available in my github repository. Clean the text and remove stopwords[9],
2. [8 points] Use NLTK to decompose the first story (A_DESCENT_INTO…) into sentences & sentences into tokens. Here is the code for doing that, after you set the variable `paragraph` to hold the text of the story.

```
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
from nltk.tokenize import sent_tokenize, word_tokenize
sent_text = nltk.sent_tokenize(paragraph) # this gives us a list of sentences
# now loop over each sentence and tokenize it separately
all_tagged = [nltk.pos_tag(nltk.word_tokenize(sent)) for sent in sent_text]
```

3. [11 points] Tag all remaining words in the story as parts of speech using the Penn POS Tags. This SO answer shows how to obtain the POS tag values. Create and print a dictionary with the Penn POS Tags as keys and a list of words as the values[10].

[8] Edgar Allan Poe was a prolific 19th century writer/poet.
[9] The code for cleaning and removing stopwords is provided below:

```
import requests
import re
import string
stopwords_list =
requests.get("https://gist.githubusercontent.com/rg089/35e00abf8941d72d419224cfd5b5925d
/raw/12d899b70156fd0041fa9778d657330b024b959c/stopwords.txt").content
stopwords = list(set(stopwords_list.decode().splitlines()))

def remove_stopwords(words):
    list_ = re.sub(r"[^a-zA-Z0-9]", " ", words.lower()).split()
    return [itm for itm in list_ if itm not in stopwords]

def clean_text(text):
    text = text.lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), ' ', text)
    text = re.sub('[\d\n]', ' ', text)
    return ' '.join(remove_stopwords(text))
```

[10] The tag dictionary is structured sort of like:

```
{'DT': ['the', 'any', …, 'a'],
 'VB': ['have', 'speak', …,],
 etc.}
```

4. [11 points] In this framework, each row will represent a story. The columns will be as follows:
    a. The text of the story,
    b. Two-letter prefixes of each tag[11], for example NN, VB, RB, JJ etc.and the words belonging to that tag in the story.
    Show your code and the tag columns, at least for the one story.

5. [12 points] The conjecture of many linguists is that the number of different parts of speech per thousand words, (nouns, verbs, adjectives, adverbs, …). is pretty much the same for all stories in a given language. In this case, with all stories in English, and all from the same author, we expect it to be true. Is the conjecture consistent with your findings?[12]

[11] In other words, the column NN will have all nouns in the story (including tags NN, NNP, NNPS, NNS). Similarly, another column will have all verbs. Nouns, Verbs, Adjectives and Adverbs are the only columns that are required. All others are optional.
[12] This question is intentionally open ended. “Conjecture consistent with findings” invites you to define how it should be expressed mathematically and evaluate the mathematics.