Assignment 2: Map Reduce Assignment

Extract Data:

Step-1:

Based on the given question, I have extracted the specific book from the given PDF using below code.

BirthMonth: March (03)

According to the books mentioned in the pdf, considering the page numbers of start and end of a specific book (**Harry Potter and the Prisoner of Azkaban**) I have created a new pdf with a specific book as per my birthday month. Below is the attached code.

Code:

```
import math
import requests
from PyPDF2 import PdfReader, PdfWriter

birth_date = 24
birth_month = 3
birth_year = 2001

if birth_month <= 7:
   book_number = birth_month
else:
   book_number = math.ceil(birth_month / 2)

print(f"Selected Book Number: {book_number}")

pdf_path = "/content/harrypotter.pdf"
reader = PdfReader(pdf_path)
total_pages = len(reader.pages)

print(f"Total pages in combined PDF: {total_pages}")

# ---------------- BOOK TITLES ----------------
book_titles = [
   "Harry Potter and the Sorcerer's Stone",
   "Harry Potter and the Chamber of Secrets",
   "Harry Potter and the Prisoner of Azkaban",
   "Harry Potter and the Goblet of Fire",
   "Harry Potter and the Order of the Phoenix",
   "Harry Potter and the Half-Blood Prince",
   "Harry Potter and the Deathly Hallows"
]

# book title according to birth month
title = book_titles[book_number - 1]
print(f"Selected Book Title: {title}")
```

Output:

```
Selected Book Number: 3
Total pages in combined PDF: 3623
Selected Book Title: Harry Potter and the Prisoner of Azkaban
```

Code:

```
reader = PdfReader(pdf_path)
writer = PdfWriter()

total_pages = len(reader.pages)
print(f"Total pages in PDF: {total_pages}")

# ---------------- GIVEN PAGE RANGE ----------------
start_page = 568
end_page = 940

# Convert to 0-based indexing
start_index = start_page - 1
end_index = end_page - 1

print(f"Extracting pages {start_page} to {end_page}...")

# ---------------- EXTRACT PAGES ----------------
for page_num in range(start_index, end_index + 1):
   writer.add_page(reader.pages[page_num])

# ---------------- SAVE NEW PDF ----------------
output_file = "Prisoner_of_Azkaban.pdf"

with open(output_file, "wb") as f:
   writer.write(f)

print(f"\nBook extracted successfully as: {output_file}")
```

Output:

```
Total pages in PDF: 3623
Extracting pages 568 to 940...

Book extracted successfully as: Prisoner_of_Azkaban.pdf
```

Step-2:
Once the specific book pdf is extracted, based on my birthdate (24), I have extracted the next 10 pages of text information and put it into a file named **file1.txt**. Below is the code and output.

Input:

```
book_pdf = "/content/Prisoner_of_Azkaban.pdf"
reader = PdfReader(book_pdf)
total_pages = len(reader.pages)

print(f"Total pages in selected book: {total_pages}")

start_page1 = birth_date
end_page1 = start_page1 + 9   # next 10 pages

# Convert to 0-based indexing
start_index1 = start_page1 - 1
end_index1 = end_page1 - 1

print(f"Extracting file1.txt from pages {start_page1} to {end_page1}")

text1 = ""

for page_num in range(start_index1, min(end_index1 + 1, total_pages)):
   text1 += reader.pages[page_num].extract_text()

with open("file1.txt", "w", encoding="utf-8") as f:
   f.write(text1)

print("file1.txt created successfully.")
```

Output:

```
Total pages in selected book: 373
Extracting file1.txt from pages 24 to 33
file1.txt created successfully.
```

With the given conditions specified in question, go to page number that corresponds to your birth year (last 2 digits). For year 2000 onwards, use 1 infront of the year number to find the page number (so year 2000 becomes

100,2001 - 101 and so on). Extract the next 10 pages into another text file (file2.txt). Below is the code and output of above logic.

Input:

```
year_last2 = birth_year % 100

if birth_year >= 2000:
   page_number = int("1" + f"{year_last2:02d}")
else:
   page_number = year_last2

start_page2 = page_number
end_page2 = start_page2 + 9

start_index2 = start_page2 - 1
end_index2 = end_page2 - 1

print(f"Extracting file2.txt from pages {start_page2} to {end_page2}")
text2 = ""

for page_num in range(start_index2, min(end_index2 + 1, total_pages)):
   text2 += reader.pages[page_num].extract_text()

with open("file2.txt", "w", encoding="utf-8") as f:
   f.write(text2)

print("file2.txt created successfully.")
```

Output:

```
Extracting file2.txt from pages 101 to 110
file2.txt created successfully.
```

Write Code to Analyze Data :

Question-1:

Write Python code and use MapReduct to count occurrences of each word in the first text file (file.txt). How many times each word is repeated?

Below is the map reduce code logic to get the word count of each word in file1.txt using **mrjobs** python package. MRJob allows the mechanism from **Map to Shuffle to Reduce pipeline** locally while preserving the logical structure of distributed MapReduce processing.

The code is initially written to a python file (wordcount_mrjob.py). There are two separate functions mapper and reduce defined in a class **WordCount**. **Map Function** tokenizes clean text into (word,1) pairs, **MRJob** performs shuffle and sort whereas **reduce function** aggregates the identical keys to get word count or frequency.

Input:

```
%%writefile wordcount_mrjob.py
from mrjob.job import MRJob
from mrjob.step import MRStep
import string

class WordCount(MRJob):
   def steps(self):
       return [
           MRStep(mapper=self.map_function,
                  reducer=self.reduce_function)
       ]
   def map_function(self, _, line):
       line = line.translate(str.maketrans('', '', string.punctuation))
       words = line.lower().split()

       for word in words:
           if word.isalpha():
               yield word, 1

   def reduce_function(self, word, counts):
       yield word, sum(counts)

if __name__ == '__main__':
   WordCount.run()
```

```
!python wordcount_mrjob.py file1.txt > wordcount_output.txt
```

The above line is executed to get the wordcount of each words in file1.txt and save it to text file named **wordcount_output.txt**

Output:
Below is the sample snippet of output from the python code after execution.

wordcount_output.txt

```
"a" 62
"able"  1
"about" 8
"abroad"    1
"abruptly"  2
"absolutely"    1
"across"    1
"added" 1
"afford"    1
"after" 3
"again" 7
"against"   3
"air"   2
"all"   9
"alleyway"  2
"almost"    5
"alone" 3
"already"   2
"also"  1
"an"    9
"and"   72
"anger" 3
"angry" 2
"animals"   1
"annoyed"   1
"any"   5
"anyone"    2
"anything"  1
"anywhere"  1
"apoplectic"    1
"appeared"  3
"approve"   1
"are"   4
```

Question-2:
From the second text file (file2.txt), write Python code and use MapReduct to count how many times non-English words (names, places, spells etc.) were used. List those words and how many times each was repeated.

Below is the python code to count the non-English words using MR Jobs logic.
It has a map function and reduce function. The **mapper function** reads the input text line by line. Each word is checked against an English dictionary using the **`SpellChecker`** library.If the word is not found in the English dictionary, it is considered a non-English word. The mapper emits a key-value pair in the form of (word, 1)
MRJob automatically groups all identical keys (words) together before passing them to the reducer.
The **reducer function** receives each word and its list of counts.
It sums the values to calculate total occurrences:

Input:

```
%%writefile nonenglish_mrjob.py
from mrjob.job import MRJob
from mrjob.step import MRStep
from spellchecker import SpellChecker
import string


class NonEnglish(MRJob):
   def steps(self):
       return [
           MRStep(mapper_init=self.mapper_init,
                  mapper=self.map_function,
                  reducer=self.reduce_function)
       ]
   def mapper_init(self):
       self.spell = SpellChecker()

   def map_function(self, _, line):
       line = line.translate(str.maketrans('', '', string.punctuation))
       words = line.lower().split()

       for word in words:
           if word.isalpha() and word not in self.spell:
               yield word, 1
```

```
    def reduce_function(self, word, counts):
        yield word, sum(counts)

if __name__ == '__main__':
    NonEnglish.run()
```

```
!python nonenglish_mrjob.py file2.txt > nonenglish_output.txt
```

Output:
The Sample output snippet taken from python code after executing above code.

nonenglish_output.txt

```
 1 "alfoy" 1
 2 "beetleblack"   1
 3 "bonebreaking"  1
 4 "buckbeak"  6
 5 "dementors" 1
 6 "flobberworms"  1
 7 "goyle" 7
 8 "gryffindor"    2
 9 "gryffindors"   1
10 "hagrid"    36
11 "mcgonagall"    1
12 "mindyeh"   1
13 "onwith"    1
14 "pomfrey"   3
15 "shoulda"   1
16 "shrivelfig"    3
17 "slytherin" 2
18 "slytherins"    3
19 "snape" 12
20 "steakandkidney"    1
21 "ter"   8
22 "twelvefoot"    1
23 "waterout"  1
24 "weasley"   1
25 "yeh"   13
26 "hermione"  16
27 "highpitched"   1
28 "hippogriffs"   7
29 "illassorted"   1
30 "malfoy"    29
31
```