**ARUNIMA SINGH THAKUR**

**SECTION C**

**ROLL NO. 31**

**180905218**

**DS LAB 5**

**25TH MAY 2021**

**QUESTION 1:**

### col\_extractor.py

#!/usr/bin/env python

# Python program to illustrate

# creating a data frame using CSV files

# import pandas module

import pandas as pd

import sys

file\_name = sys.argv[1]

col = int(sys.argv[2]) - 1

# creating a data frame

if file\_name.endswith('.csv'):

df = pd.read\_csv(file\_name)

print(df.iloc[:,col].to\_string(index=False))

elif file\_name.endswith('.txt'):

with open(file\_name, 'r') as f:

for row in f.readlines():

count = 0

for cell in row.split('\t'):

if count == col:

print(cell)

break

count += 1

### mapper.py

#!/usr/bin/env python

import sys

# input comes from STDIN (standard input)

for line in sys.stdin:

# remove leading and trailing whitespace

line = line.strip()

# split the line into words

words = line.split()

# increase counters

for word in words:

# write the results to STDOUT (standard output);

# what we output here will be the input for the

# Reduce step, i.e. the input for reducer.py

#

# tab-delimited; the trivial word count is 1

print('%s\t%s' % (word, 1))

### reducer.py

#!/usr/bin/env python

from operator import itemgetter

import sys

current\_word = None

current\_count = 0

word = None

# input comes from STDIN

for line in sys.stdin:

# remove leading and trailing whitespace

line = line.strip()

# parse the input we got from mapper.py

word, count = line.split('\t', 1)

# convert count (currently a string) to int

try:

count = int(count)

except ValueError:

# count was not a number, so silently

# ignore/discard this line

continue

# this IF-switch only works because Hadoop sorts map output

# by key (here: word) before it is passed to the reducer

if current\_word == word:

current\_count += count

else:

if current\_word:

# write result to STDOUT

print ('%s\t%s' % (current\_word, current\_count))

current\_count = count

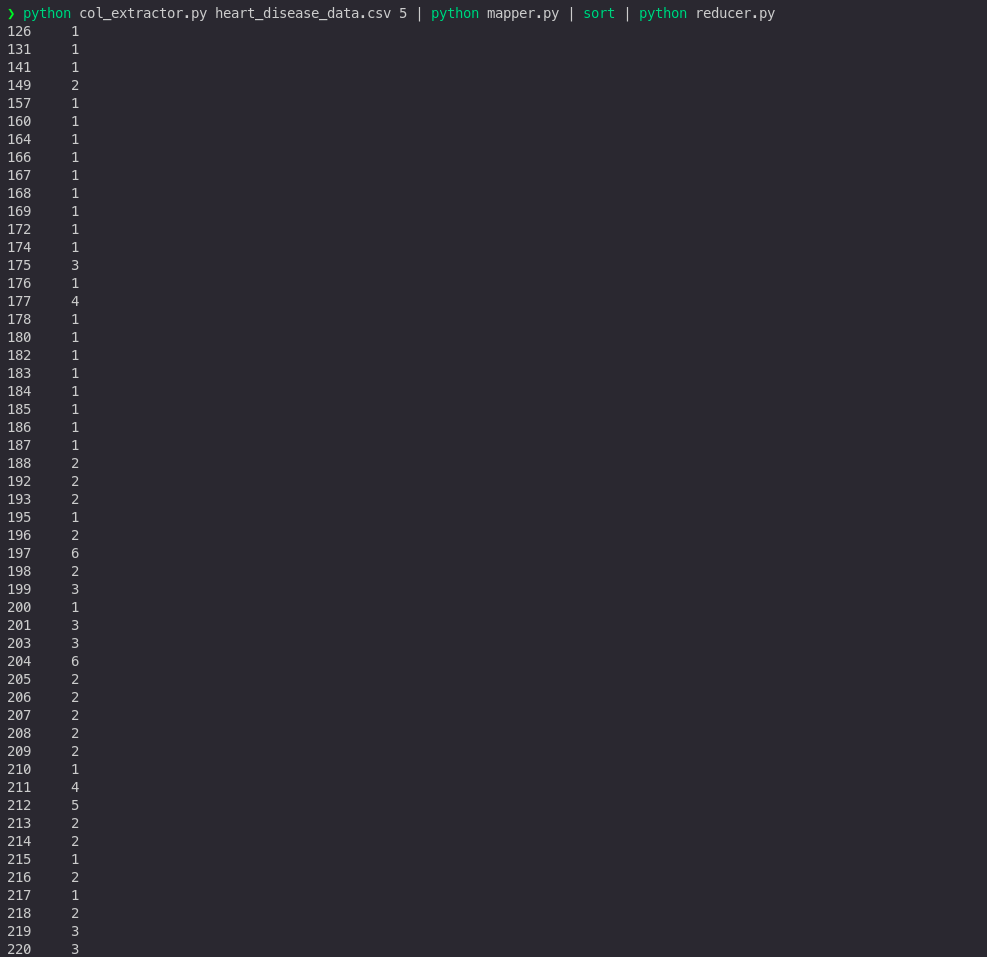
current\_word = word

# do not forget to output the last word if needed!

if current\_word == word:

print ('%s\t%s' % (current\_word, current\_count))

## Output:



**QUESTION 2:**

### freqmap1.py

#!/usr/bin/env python

# A basic mapper function/program that

# takes whatever is passed on the input and

# outputs tuples of all the words formatted

# as (word, 1)

from \_\_future\_\_ import print\_function

import sys

# input comes from STDIN (standard input)

for line in sys.stdin:

# create tuples of all words in line

L = [(word.strip().lower(), 1) for word in line.strip().split()]

# increase counters

for word, n in L:

# write the results to STDOUT (standard output);

# what we output here will be the input for the

# Reduce step, i.e. the input for reducer.py

#

# tab-delimited; the trivial word count is 1

print('%s\t%d' % (word, n))

### freqred1.py

#!/usr/bin/env python

# reducer.py

from \_\_future\_\_ import print\_function

import sys

lastWord = None

sum = 0

for line in sys.stdin:

word, count = line.strip().split('\t', 1)

count = int(count)

if lastWord == None:

lastWord = word

sum = count

continue

if word == lastWord:

sum += count

else:

print("%s\t%d" % (lastWord, sum))

sum = count

lastWord = word

# output last word

if lastWord == word:

print('%s\t%s' % (lastWord, sum))

### freqmap2.py

#!/usr/bin/env python

# A basic mapper function/program that

# takes whatever is passed on the input and

# outputs tuples of all the words formatted

# as (word, 1)

from \_\_future\_\_ import print\_function

import sys

# input comes from STDIN (standard input)

for line in sys.stdin:

word, count = line.strip().split('\t', 1)

count = int(count)

print('%d\t%s' % (count, word))

### freqred2.py

#!/usr/bin/env python

# reducer.py

from \_\_future\_\_ import print\_function

import sys

mostFreq = []

currentMax = -1

for line in sys.stdin:

count, word = line.strip().split('\t', 1)

count = int(count)

if count > currentMax:

currentMax = count

mostFreq = [ word ]

elif count == currentMax:

mostFreq.append( word )

# output mostFreq word(s)

for word in mostFreq:

print( '%s\t%s' % ( word, currentMax ) )

## Output:



**QUESTION 3:**

### itemmap.py

#!/usr/bin/python

"""

INPUT: Transactions of products in multiple stores and location; these can also be passed to STDIN

Format of each line is: date\ttime\tstore location\titem description\tcost\tmethod of payment

OUTPUT: E.g.

Las Vegas 208.97

Miami 84.11

Tucson 489.93

San Francisco 388.3

Dallas 145.63

Tampa 353.23

Washington 481.31

San Jose 492.8

Newark 410.37

Memphis 354.44

Jersey City 369.07

Plano 4.65

Buffalo 337.35

Louisville 213.64

Miami 154.64

...

"""

#import string

import fileinput

for line in fileinput.input():

data = line.strip().split("\t")

if len(data) == 6:

date, time, location, item, cost, payment = data

print ("{0}\t{1}".format(location, cost))

#can try with different instances.....

#print ("{0}\t{1}".format(payment, cost))

#print ("{0}\t{1}".format(item, cost))

### itempred.py

#!/usr/bin/python

"""

INPUT: Output from mapper.py

Format of each line is: location\tcost

OUTPUT: E.g.

50 12268.16

"""

import fileinput

transactions\_count = 0

sales\_total = 0

for line in fileinput.input():

data = line.strip().split("\t")

if len(data) != 2:

# Something has gone wrong. Skip this line.

continue

current\_key, current\_value = data

transactions\_count += 1

sales\_total += float(current\_value)

print (str(transactions\_count) + "\t" + str(sales\_total))

## Output:



**QUESTION 4:**

### sepmap.py

#!/usr/bin/env python

"""A more advanced Mapper, using Python iterators and generators."""

import sys

def read\_input(file):

for line in file:

# split the line into words

yield line.split()

def main(separator='\t'):

# input comes from STDIN (standard input)

data = read\_input(sys.stdin)

for words in data:

# write the results to STDOUT (standard output);

# what we output here will be the input for the

# Reduce step, i.e. the input for reducer.py

# tab-delimited; the trivial word count is 1

for word in words:

print ('%s%s%d' % (word, separator, 1))

if \_\_name\_\_ == "\_\_main\_\_":

main()

### sepred.py

#!/usr/bin/env python

"""A more advanced Reducer, using Python iterators and generators."""

from itertools import groupby

from operator import itemgetter

import sys

def read\_mapper\_output(file, separator='\t'):

for line in file:

yield line.rstrip().split(separator, 1)

def main(separator='\t'):

# input comes from STDIN (standard input)

data = read\_mapper\_output(sys.stdin, separator=separator)

# groupby groups multiple word-count pairs by word,

# and creates an iterator that returns consecutive keys and their group:

# current\_word - string containing a word (the key)

# group - iterator yielding all ["&lt;current\_word&gt;", "&lt;count&gt;"] items

for current\_word, group in groupby(data, itemgetter(0)):

try:

total\_count = sum(int(count) for current\_word, count in group)

print ("%s%s%d" % (current\_word, separator, total\_count))

except ValueError:

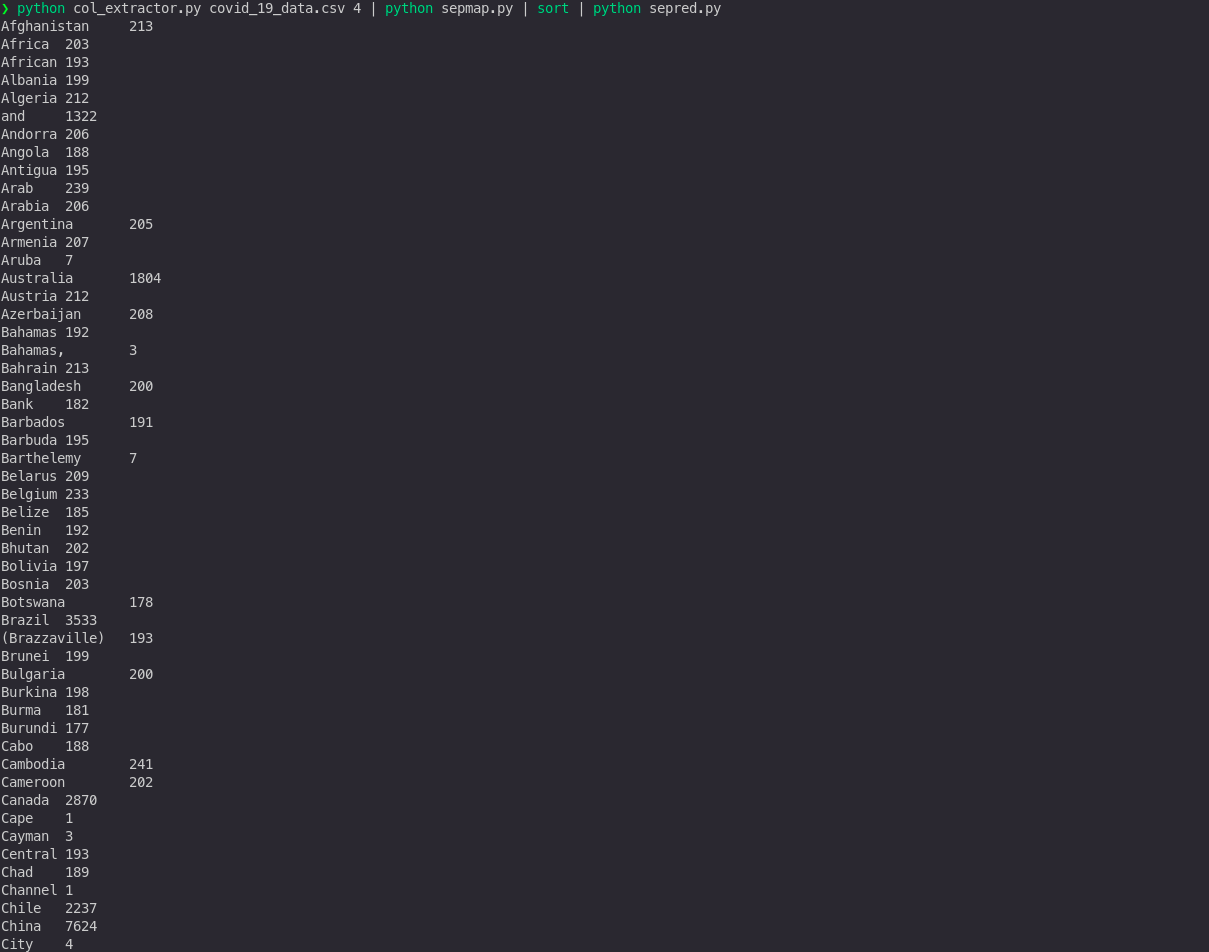
# count was not a number, so silently discard this item

pass

if \_\_name\_\_ == "\_\_main\_\_":

main()

## Output:



**QUESTION 5:**

### itemmap\_expensive.py

#!/usr/bin/python

"""

INPUT: Transactions of products in multiple stores and location; these can also be passed to STDIN

Format of each line is: date\ttime\tstore location\titem description\tcost\tmethod of payment

OUTPUT: E.g.

Las Vegas 208.97

Miami 84.11

Tucson 489.93

San Francisco 388.3

Dallas 145.63

Tampa 353.23

Washington 481.31

San Jose 492.8

Newark 410.37

Memphis 354.44

Jersey City 369.07

Plano 4.65

Buffalo 337.35

Louisville 213.64

Miami 154.64

...

"""

#import string

import fileinput

for line in fileinput.input():

data = line.strip().split("\t")

if len(data) == 6:

date, time, location, item, cost, payment = data

print ("{0}\t{1}".format(location, cost))

#can try with different instances.....

#print ("{0}\t{1}".format(payment, cost))

#print ("{0}\t{1}".format(item, cost))

### itemred\_expensive.py

#!/usr/bin/python

"""

INPUT: Output from mapper.py

Format of each line is: location\tcost

OUTPUT: E.g.

Atlanta 189.22

Aurora 82.38

Austin 48.09

Birmingham 1.64

Boston 397.21

Buffalo 386.56

"""

import fileinput

max\_value = 0

old\_key = None

for line in fileinput.input():

data = line.strip().split("\t")

if len(data) != 2:

# Something has gone wrong. Skip this line.

continue

current\_key, current\_value = data

# Refresh for new keys (i.e. locations in the example context)

if old\_key and old\_key != current\_key:

print (old\_key + "\t" + str(max\_value))

old\_key = current\_key

max\_value = 0

old\_key = current\_key

if float(current\_value) > float(max\_value):

max\_value = float(current\_value)

if old\_key != None:

print (old\_key + "\t" + str(max\_value))

## Output:

