ASS_01_21L-7289

March 25, 2022

BIG DATA ANALYTICS

ASSIGNMENT 1

Name: SAAD ATHER Roll No. 21L-7289

Question 1: (10 marks) We have a huge data file of FAST students. The file consists of the roll number of the FAST students, followed by the subject code and the student grade in that subject. Consider an input file in the following format:

L22-2100 DB D K21-1601 SE F I21-1601 OS F K21-1702 DS B L21-1705 OS A L22-2101 DB D K21-1601 OS F L21-1601 SE F L21-1702 SE B L21-1705 DB A

---

[1]:
```
import findspark
```

[2]:
```
findspark.init()
```

[3]:
```
import pyspark
```

[4]:
```
from pyspark.sql import SparkSession as ss1
```

[5]:
```
spark = ss1.builder.appName('SESSION1').getOrCreate()
```

[6]:
```
sc = spark.sparkContext
```

i. Select the records of students from the Lahore campus. Display a few records and print the count of the students from Lahore.

[7]:
```
file = sc.textFile('FAST.txt')
```

[8]:
```
op1 = file.map(lambda x:x).collect()

op1 ## Open the file and show all the record of students of FAST
```

[8]:
```
['L22-2100 DB D',
 'K21-1601 SE F',
 'I21-1601 OS F',
```

```
 'K21-1702 DS B',
 'L21-1705 OS A',
 'L22-2101 DB D',
 'L21-1601 SE F',
 'L21-1702 SE B',
 'L01-2002 DB D',
 'K18-1891 SE F',
 'I95-1223 OS F',
 'L09-7245 DB D',
 'K10-1983 SE F',
 'I15-1245 OS F',
 'P10-1991 SE F',
 'F19-1111 OS F',
 'P21-7252 DB D',
 'F17-1963 SE D',
 'P12-2425 OS F',
 'F11-1235 OS F']
```

[9]:
```
## Shows the record of students from Lahore

op2 = file.filter(lambda x: x[0]=='L').collect()
op2
```

[9]:
```
['L22-2100 DB D',
 'L21-1705 OS A',
 'L22-2101 DB D',
 'L21-1601 SE F',
 'L21-1702 SE B',
 'L01-2002 DB D',
 'L09-7245 DB D']
```

[10]:
```
## COUNT OF STUDENT OF LAHORE

op3 = file.filter(lambda x: x[0]=='L').count()
```

[11]:
```
print('TOTAL OF {} STUDENTS ARE FROM LAHORE CAMPUS'.format(op3))
```

```
TOTAL OF 7 STUDENTS ARE FROM LAHORE CAMPUS
```

ii. Filter the records of the students from the year in the range of 1995- 2018.

[12]:
```
## creating a list of years from 1995 to 2018

range_ = [str(x) for x in range(0,19)]
for x in range(95,100): range_.append(str(x))
```

[13]:
```
op4 = file.filter(lambda x : x[1:3] in range_).collect()
```

[14]:
```
print(op4)
```

```
['K18-1891 SE F', 'I95-1223 OS F', 'K10-1983 SE F', 'I15-1245 OS F', 'P10-1991
SE F', 'F17-1963 SE D', 'P12-2425 OS F', 'F11-1235 OS F']
```

iii. Display the count of students on each Campus.

[15]:
```
op5 = file.groupBy(lambda x: x[0]).mapValues(len).collect()
op5
```

[15]:
```
[('L', 7), ('K', 4), ('I', 3), ('P', 3), ('F', 3)]
```

[16]:
```
for x,y in op5:
    if (x=='L'): print('Students in {} campus are {} '.format('Lahore',y))
    if (x=='I'): print('Students in {} campus are {} '.format('Islamabad',y))
    if (x=='K'): print('Students in {} campus are {} '.format('Karachi',y))
    if (x=='P'): print('Students in {} campus are {} '.format('Peshaware',y))
    if (x=='F'): print('Students in {} campus are {} '.format('Faisalabad',y))
```

```
Students in Lahore campus are 7
Students in Karachi campus are 4
Students in Islamabad campus are 3
Students in Peshaware campus are 3
Students in Faisalabad campus are 3
```

iv. Partition the input data on the base of Campus. (override Spark Partitioner).

[17]:
```
op6 = file.map(lambda x: x.split('-'))
```

[18]:
```
op7 = op6.groupByKey().mapValues(len)
```

[19]:
```
def partfun(l):
    for x in l:

        #print(x[0][0])

        if (x[0][0]== 'L'): return 0
        if (x[0][0]== 'I'): return 1
        if (x[0][0]== 'K'): return 2
        if (x[0][0]== 'P'): return 3
        if (x[0][0]== 'F'): return 4
```

[20]:
```
op = file.map(lambda x: x).collect()
op
```

[20]:
```
['L22-2100 DB D',
 'K21-1601 SE F',
```

```
 'I21-1601 OS F',
 'K21-1702 DS B',
 'L21-1705 OS A',
 'L22-2101 DB D',
 'L21-1601 SE F',
 'L21-1702 SE B',
 'L01-2002 DB D',
 'K18-1891 SE F',
 'I95-1223 OS F',
 'L09-7245 DB D',
 'K10-1983 SE F',
 'I15-1245 OS F',
 'P10-1991 SE F',
 'F19-1111 OS F',
 'P21-7252 DB D',
 'F17-1963 SE D',
 'P12-2425 OS F',
 'F11-1235 OS F']
```

[21]:
```
op[0][0]
```

[21]:
```
'L'
```

[22]:
```
opx = file.groupBy(lambda x : x[0]).mapValues(list).partitionBy(5,lambda x :␣
 ↪partfun(x))
opx
```

[22]:
```
MapPartitionsRDD[23] at mapPartitions at PythonRDD.scala:145
```

[23]:
```
## Partition based on Campuses

print('Number of partitions {} : '.format(opx.getNumPartitions()),'\n')
print('Partitioner {} : '.format(opx.partitioner),'\n')
print('Partitions structure {} : '.format(opx.glom().collect()),'\n')
```

```
Number of partitions 5 :

Partitioner <pyspark.rdd.Partitioner object at 0x00000265337853D0> :

Partitions structure [[('L', ['L22-2100 DB D', 'L21-1705 OS A', 'L22-2101 DB D',
'L21-1601 SE F', 'L21-1702 SE B', 'L01-2002 DB D', 'L09-7245 DB D'])], [('I',
['I21-1601 OS F', 'I95-1223 OS F', 'I15-1245 OS F'])], [('K', ['K21-1601 SE F',
'K21-1702 DS B', 'K18-1891 SE F', 'K10-1983 SE F'])], [('P', ['P10-1991 SE F',
'P21-7252 DB D', 'P12-2425 OS F'])], [('F', ['F19-1111 OS F', 'F17-1963 SE D',
'F11-1235 OS F'])]] :
```

v. For each course, print the number of failures on each Campus.

[24]:
```
opv = file.filter(lambda x: x[-1]=='F')
```

[25]:
```
### Total Record of the failure students

opv.collect()
```

[25]:
```
['K21-1601 SE F',
 'I21-1601 OS F',
 'L21-1601 SE F',
 'K18-1891 SE F',
 'I95-1223 OS F',
 'K10-1983 SE F',
 'I15-1245 OS F',
 'P10-1991 SE F',
 'F19-1111 OS F',
 'P12-2425 OS F',
 'F11-1235 OS F']
```

[26]:
```
opv2 = opv.groupBy(lambda x : x[0]).mapValues(list).collect()
```

[27]:
```
opv2
```

[27]:
```
[('K', ['K21-1601 SE F', 'K18-1891 SE F', 'K10-1983 SE F']),
 ('L', ['L21-1601 SE F']),
 ('I', ['I21-1601 OS F', 'I95-1223 OS F', 'I15-1245 OS F']),
 ('P', ['P10-1991 SE F', 'P12-2425 OS F']),
 ('F', ['F19-1111 OS F', 'F11-1235 OS F'])]
```

[28]:
```
print('\n')

for x,y in opv2:
    if (x=='L'): print('Students in {} campus are {} '.format('Lahore',y))
    if (x=='I'): print('Students in {} campus are {} '.format('Islamabad',y))
    if (x=='K'): print('Students in {} campus are {} '.format('Karachi',y))
    if (x=='P'): print('Students in {} campus are {} '.format('Peshaware',y))
    if (x=='F'): print('Students in {} campus are {} '.format('Faisalabad',y))
print('\n')
```

```
Students in Karachi campus are ['K21-1601 SE F', 'K18-1891 SE F', 'K10-1983 SE
F']
Students in Lahore campus are ['L21-1601 SE F']
Students in Islamabad campus are ['I21-1601 OS F', 'I95-1223 OS F', 'I15-1245 OS
F']
Students in Peshaware campus are ['P10-1991 SE F', 'P12-2425 OS F']
```

```
Students in Faisalabad campus are ['F19-1111 OS F', 'F11-1235 OS F']
```

vi. Remove all the duplicate rows from input data.

[29]:
```
## Shows file including the duplicates

doc =sc.textFile('FAST-Copy.txt')
doc.take(50)
```

[29]:
```
['L22-2100 DB D',
 'K21-1601 SE F',
 'I21-1601 OS F',
 'K21-1702 DS B',
 'L21-1705 OS A',
 'L22-2101 DB D',
 'K21-1601 OS F',
 'L21-1601 SE F',
 'L21-1702 SE B',
 'L21-1705 DB A',
 'L01-2002 DB D',
 'K18-1891 SE F',
 'I95-1223 OS F',
 'L09-7245 DB D',
 'K10-1983 SE F',
 'I15-1245 OS F',
 '#added duplictae below for part vi',
 'L22-2100 DB D',
 'K21-1601 SE F',
 'I21-1601 OS F',
 'K21-1702 DS B']
```

[30]:
```
print('Total size ',len(doc.take(30)))
```

```
Total size  21
```

[31]:
```
## Removing duplicates

opvi = doc.filter(lambda x: x != '#added duplictae below for part vi').
 ↪distinct().collect()

## Applied filter here coz with just the .distinct() it was just removing the␣
 ↪duplicates but showing the string as well
```

[32]:
```
opvi
```

[32]:
```
['K21-1601 SE F',
 'K21-1702 DS B',
 'L22-2101 DB D',
 'L21-1601 SE F',
 'L21-1702 SE B',
 'L21-1705 DB A',
 'L01-2002 DB D',
 'L09-7245 DB D',
 'K10-1983 SE F',
 'L22-2100 DB D',
 'I21-1601 OS F',
 'L21-1705 OS A',
 'K21-1601 OS F',
 'K18-1891 SE F',
 'I95-1223 OS F',
 'I15-1245 OS F']
```

[33]:
```
print('After removing duplicates, size of file is',len(opvi))
```

```
After removing duplicates, size of file is 16
```

vii. Find the minimum and maximum grades in each subject. The ordering of grades is as follows A >B >C >D >F

[34]:
```
opvii = file.groupBy(lambda x : x[-4:]).mapValues(list)
```

[35]:
```
opvii.collect()
```

[35]:
```
[('DB D',
  ['L22-2100 DB D',
   'L22-2101 DB D',
   'L01-2002 DB D',
   'L09-7245 DB D',
   'P21-7252 DB D']),
 ('SE F',
  ['K21-1601 SE F',
   'L21-1601 SE F',
   'K18-1891 SE F',
   'K10-1983 SE F',
   'P10-1991 SE F']),
 ('OS A', ['L21-1705 OS A']),
 ('SE B', ['L21-1702 SE B']),
 ('OS F',
  ['I21-1601 OS F',
   'I95-1223 OS F',
   'I15-1245 OS F',
   'F19-1111 OS F',
   'P12-2425 OS F',
```

```
  'F11-1235 OS F']),
 ('DS B', ['K21-1702 DS B']),
 ('SE D', ['F17-1963 SE D'])]
```

[36]:
```
sub_grade = opvii.keys().sortBy(lambda x: x[-1]).map(lambda x : x.split(' ')).
↪groupByKey().mapValues(list).collect()
```

[37]:
```
### Final answer will be

sub_grade
```

[37]:
```
[('OS', ['A', 'F']), ('DB', ['D']), ('SE', ['B', 'D', 'F']), ('DS', ['B'])]
```

viii. We wish to sort the file based on the roll number (hint work with sortByKey). The two roll-numbers are compared using the following rule

a. For Campus use lexicographic ordering that is F < I < k <L< P

[38]:
```
opviii = file.groupBy(lambda x: x[0]).mapValues(list)
```

[39]:
```
opviiia = opviii.sortByKey().flatMap(lambda x: x[1])
```

[40]:
```
### a. For rollno based sorting Campus lexicographic ordering that is F < I < k␣
↪<L< P is given below

opviiia.collect()
```

[40]:
```
['F19-1111 OS F',
 'F17-1963 SE D',
 'F11-1235 OS F',
 'I21-1601 OS F',
 'I95-1223 OS F',
 'I15-1245 OS F',
 'K21-1601 SE F',
 'K21-1702 DS B',
 'K18-1891 SE F',
 'K10-1983 SE F',
 'L22-2100 DB D',
 'L21-1705 OS A',
 'L22-2101 DB D',
 'L21-1601 SE F',
 'L21-1702 SE B',
 'L01-2002 DB D',
 'L09-7245 DB D',
 'P10-1991 SE F',
 'P21-7252 DB D',
 'P12-2425 OS F']
```

b. For year follow the rule of year 16<17 and 99 < 01

[41]:
```
## For this part sorting based on student year has been followed

opviii = file.groupBy(lambda x: x[1:3]).mapValues(list)
```

[42]:
```
opviiib = opviii.sortByKey().flatMap(lambda x: x[1])
```

[43]:
```
opviiib.collect()
```

[43]:
```
['L01-2002 DB D',
 'L09-7245 DB D',
 'K10-1983 SE F',
 'P10-1991 SE F',
 'F11-1235 OS F',
 'P12-2425 OS F',
 'I15-1245 OS F',
 'F17-1963 SE D',
 'K18-1891 SE F',
 'F19-1111 OS F',
 'K21-1601 SE F',
 'I21-1601 OS F',
 'K21-1702 DS B',
 'L21-1705 OS A',
 'L21-1601 SE F',
 'L21-1702 SE B',
 'P21-7252 DB D',
 'L22-2100 DB D',
 'L22-2101 DB D',
 'I95-1223 OS F']
```

c. For the last part of roll-number, follow int ordering.

[44]:
```
## For this part, sorting is based on last four digits of student Roll# has␣
 ↪been followed

opviii = file.groupBy(lambda x: x[4:8]).mapValues(list)
```

[45]:
```
opviiic = opviii.sortByKey().flatMap(lambda x: x[1])
```

[46]:
```
opviiic.collect()
```

[46]:
```
['F19-1111 OS F',
 'I95-1223 OS F',
 'F11-1235 OS F',
 'I15-1245 OS F',
 'K21-1601 SE F',
 'I21-1601 OS F',
```

```
 'L21-1601 SE F',
 'K21-1702 DS B',
 'L21-1702 SE B',
 'L21-1705 OS A',
 'K18-1891 SE F',
 'F17-1963 SE D',
 'K10-1983 SE F',
 'P10-1991 SE F',
 'L01-2002 DB D',
 'L22-2100 DB D',
 'L22-2101 DB D',
 'P12-2425 OS F',
 'L09-7245 DB D',
 'P21-7252 DB D']
```

ix. For each student, compute the GPA. Assume only five grades (Grade A GPA=4, Grade B GPA=3, Grade C GPA 2, Grade D GPA 1, and Grade F GPA=0)

[47]:
```
opix = file.groupBy(lambda x: x[0:8]).distinct().mapValues(list)
```

[48]:
```
opix1 = opix.mapValues(lambda x : x[0][-1])
```

[49]:
```
gpa = opix1.collect()
gpa
```

[49]:
```
[('L09-7245', 'D'),
 ('I15-1245', 'F'),
 ('P10-1991', 'F'),
 ('F19-1111', 'F'),
 ('L22-2100', 'D'),
 ('I21-1601', 'F'),
 ('K21-1702', 'B'),
 ('L21-1705', 'A'),
 ('P21-7252', 'D'),
 ('P12-2425', 'F'),
 ('F11-1235', 'F'),
 ('K21-1601', 'F'),
 ('L21-1702', 'B'),
 ('L01-2002', 'D'),
 ('I95-1223', 'F'),
 ('F17-1963', 'D'),
 ('L22-2101', 'D'),
 ('L21-1601', 'F'),
 ('K18-1891', 'F'),
 ('K10-1983', 'F')]
```

[50]:
```
for x,y in gpa:
    if (y=='A'):
        print('GPA of student {} is {} '.format(x,4.0))
    if (y=='B'):
        print('GPA of student {} is {} '.format(x,3.0))
    if (y=='C'):
        print('GPA of student {} is {} '.format(x,2.0))
    if (y=='D'):
        print('GPA of student {} is {} '.format(x,1.0))
    if (y=='F'):
        print('GPA of student {} is {} '.format(x,0.0))
```

```
GPA of student L09-7245 is 1.0
GPA of student I15-1245 is 0.0
GPA of student P10-1991 is 0.0
GPA of student F19-1111 is 0.0
GPA of student L22-2100 is 1.0
GPA of student I21-1601 is 0.0
GPA of student K21-1702 is 3.0
GPA of student L21-1705 is 4.0
GPA of student P21-7252 is 1.0
GPA of student P12-2425 is 0.0
GPA of student F11-1235 is 0.0
GPA of student K21-1601 is 0.0
GPA of student L21-1702 is 3.0
GPA of student L01-2002 is 1.0
GPA of student I95-1223 is 0.0
GPA of student F17-1963 is 1.0
GPA of student L22-2101 is 1.0
GPA of student L21-1601 is 0.0
GPA of student K18-1891 is 0.0
GPA of student K10-1983 is 0.0
```

x. Convert grades to GPA as mentioned in part viii and find the average GPA of each Subject

[51]:
```
### THis will generate the grades per subject so further we will use these␣
 ↪grade to calculate the subject averages

opx = file.sortBy(lambda x : x[-4:]).map(lambda x : x[-4:]).map(lambda x: x.
 ↪split(' ')).groupByKey().mapValues(list)
opx.collect()
```

[51]:
```
[('DB', ['D', 'D', 'D', 'D', 'D']),
 ('OS', ['A', 'F', 'F', 'F', 'F', 'F', 'F']),
 ('DS', ['B']),
```

```
('SE', ['B', 'D', 'F', 'F', 'F', 'F', 'F'])]
```

[52]:
```
### utility function to convert the GPA of a subject

avg_list = list()
def avg_cal(data):
    for y in data:
        if (y=='A'):
            avg_list.append(4)
        if (y=='B'):
            avg_list.append(3)
        if (y=='C'):
            avg_list.append(2)
        if (y=='D'):
            avg_list.append(1)
        if (y=='F'):
            avg_list.append(0)
    size = len(avg_list)
    lsum = sum(avg_list)
    avg = lsum/size
    return avg

def avg():
    for i in range(0,len(opx1)):
        print('AVERAGE GPA OF {} is {} '.format(opx1[i][0],avg_cal(opx1[i][1])))
```

[53]:
```
opx1 = opx.collect()
opx2 = opx.map(avg())
```

```
AVERAGE GPA OF DB is 1.0
AVERAGE GPA OF OS is 0.75
AVERAGE GPA OF DS is 0.9230769230769231
AVERAGE GPA OF SE is 0.8
```

[ ]:
```

```