Aisha Muhammad Nawaz L200921

PySpark Lab 3 8A BSCS MMD

22nd February 2024

Instructions: Solve pyspark excercises done in class and practice questions given in slides

In [1]:
```
# #Running on Colab
!pip install pyspark
!pip install -U -q PyDrive
!apt install openjdk-8-jdk-headless -qq
import os
os.environ['JAVA_HOME'] = '/usr/lib/jvm/java-8-openjdk-amd64'
```

```
Collecting pyspark
  Downloading pyspark-3.5.0.tar.gz (316.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 316.9/316.9 MB 2.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.1
0.9.7)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.5.0-py2.py3-none-any.whl size=317425345 sha256=1202938330e115
91b9ceaebf84747cb537d6aeeeb9242ed915ed1e22e402f43e
  Stored in directory: /root/.cache/pip/wheels/41/4e/10/c2cf2467f71c678cfc8a6b9ac9241e5e44a01940da8fbb17fc
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.0
The following additional packages will be installed:
  libxtst6 openjdk-8-jre-headless
Suggested packages:
  openjdk-8-demo openjdk-8-source libnss-mdns fonts-dejavu-extra fonts-nanum fonts-ipafont-gothic
  fonts-ipafont-mincho fonts-wqy-microhei fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  libxtst6 openjdk-8-jdk-headless openjdk-8-jre-headless
0 upgraded, 3 newly installed, 0 to remove and 35 not upgraded.
Need to get 39.7 MB of archives.
After this operation, 144 MB of additional disk space will be used.
Selecting previously unselected package libxtst6:amd64.
(Reading database ... 121749 files and directories currently installed.)
Preparing to unpack .../libxtst6_2%3a1.2.3-1build4_amd64.deb ...
Unpacking libxtst6:amd64 (2:1.2.3-1build4) ...
Selecting previously unselected package openjdk-8-jre-headless:amd64.
Preparing to unpack .../openjdk-8-jre-headless_8u392-ga-1~22.04_amd64.deb ...
Unpacking openjdk-8-jre-headless:amd64 (8u392-ga-1~22.04) ...
Selecting previously unselected package openjdk-8-jdk-headless:amd64.
Preparing to unpack .../openjdk-8-jdk-headless_8u392-ga-1~22.04_amd64.deb ...
Unpacking openjdk-8-jdk-headless:amd64 (8u392-ga-1~22.04) ...
Setting up libxtst6:amd64 (2:1.2.3-1build4) ...
Setting up openjdk-8-jre-headless:amd64 (8u392-ga-1~22.04) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/orbd to provide /usr/bin/orbd (orbd) in
auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/servertool to provide /usr/bin/servertoo
l (servertool) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/tnameserv to provide /usr/bin/tnameserv
(tnameserv) in auto mode
Setting up openjdk-8-jdk-headless:amd64 (8u392-ga-1~22.04) ...
```

```
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/clhsdb to provide /usr/bin/clhsdb (clhsdb) i
n auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/extcheck to provide /usr/bin/extcheck (extch
eck) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/hsdb to provide /usr/bin/hsdb (hsdb) in auto
mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/idlj to provide /usr/bin/idlj (idlj) in auto
mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/javah to provide /usr/bin/javah (javah) in a
uto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jhat to provide /usr/bin/jhat (jhat) in auto
mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jsadebugd to provide /usr/bin/jsadebugd (jsa
debugd) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/native2ascii to provide /usr/bin/native2asci
i (native2ascii) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/schemagen to provide /usr/bin/schemagen (sch
emagen) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/wsgen to provide /usr/bin/wsgen (wsgen) in a
uto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/wsimport to provide /usr/bin/wsimport (wsimp
ort) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/xjc to provide /usr/bin/xjc (xjc) in auto mo
de
Processing triggers for libc-bin (2.35-0ubuntu3.4) ...
/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc_proxy.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_0.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_5.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbb.so.12 is not a symbolic link
```

In [2]:
```
#  Import the libraries we will need
import pyspark
from pyspark.sql import *
from pyspark.sql.functions import *
from pyspark import SparkContext, SparkConf
```

In [3]:
```
# Create Spark session and ContextRun PySpark.
# create the session
conf = SparkConf().set("spark.ui.port","4050")
# create the context
sc = pyspark.SparkContext(conf=conf)
spark = SparkSession.builder.appName("DataFrame").config('spark.ui.port', '4050').getOrCreate()
```

In [4]:
```
spark
```

Out[4]:

SparkSession - in-memory

SparkContext

Spark UI (http://899e1b092059:4050)

Version

v3.5.0

Master

local[*]

AppName

pyspark-shell

In [ ]:
```
# Q1. Find Students Belonging to Lahore Campus Only
studs = sc.parallelize(['L20-0919 DB A','L20-0921 PPIT B','L20-0914 DB C','I19-0989 DB D','K17-0999 OS C'])
studs.filter(lambda x: x.startswith('L')).collect()
```

Out[ ]:
```
['L20-0919 DB A', 'L20-0921 PPIT B', 'L20-0914 DB C']
```

In [ ]:
```
#Q1. (b) Select the records of students from the Lahore campus. Display a few records and print the count of
the students from Lahore.
studs = sc.parallelize([
'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',])
results=studs.filter(lambda x: x.startswith('L'))
print('Displaying a few students from lahore campus:-')
print(results.take(6))
print('Total Number of students from lahore campus = ',results.count())
```

```
Displaying a few students from lahore campus:-
['L22-2100 DB D', 'L21-1705 OS A', 'L22-2101 DB D', 'L21-1601 SE F', 'L21-1702 SE B', 'L21-1705 DB A']
Total Number of students from lahore campus =  6
```

In [ ]:
```
# Q2. Find Students From Batch 2016-2019 only
studs = sc.parallelize(['L20-0919 DB A','L20-0921 PPIT B','L20-0914 DB C','I19-0989 DB D','K17-0999 OS C'])
studs.map(lambda x: int(x[1:].split('-')[0])).filter(lambda y: (y>=16 and y<=19)).collect()
```

Out[ ]:
```
[19, 17]
```

In [ ]:
```
# Q2. (b) Find records of the students from the year in the range of 1995-2018.
studs = sc.parallelize([
'L22-2100 DB D',
'K01-1601 SE F',
'I21-1601 OS F',
'K96-1702 DS B',
'L21-1705 OS A',
'L22-2101 DB D',
'K98-1601 OS F',
'L99-1601 SE F',
'L11-1702 SE B',
'L17-1705 DB A',])
studs.filter(lambda x: (int(x[1:].split('-')[0])>=95) or (int(x[1:].split('-')[0])<=18)).collect()
```

Out[ ]:
```
['K01-1601 SE F',
 'K96-1702 DS B',
 'K98-1601 OS F',
 'L99-1601 SE F',
 'L11-1702 SE B',
 'L17-1705 DB A']
```

In [ ]:
```
# Q3. Find Count of Students in Each Campus
studs = sc.parallelize(['L20-0919 DB A','L20-0921 PPIT B','L20-0914 DB C','I19-0989 DB D','K17-0999 OS C','K16-1119 DS A'])
studs.map(lambda x: (x[0],x.split(' ')[0])).countByKey()
```

Out[ ]:
```
defaultdict(int, {'L': 3, 'I': 1, 'K': 2})
```

In [ ]:
```
# Q3. (b) Display the count of students on each Campus.
studs = sc.parallelize([
'L22-2100 DB D',
'K01-1601 SE F',
'I21-1601 OS F',
'K96-1702 DS B',
'L21-1705 OS A',
'L22-2101 DB D',
'K98-1601 OS F',
'L99-1601 SE F',
'L11-1702 SE B',
'L17-1705 DB A',])
studs.map(lambda x: (x[0],x.split(' ')[0])).countByKey()
```

Out[ ]:
```
defaultdict(int, {'L': 6, 'K': 3, 'I': 1})
```

In [ ]:
```
# Q4. Remove Duplicate Rows in Input Data
studs = sc.parallelize(['L20-0919 DB A','L20-0921 PPIT B','L20-0914 DB C','I19-0989 DB D','K17-0999 OS C','K16-1119 DS A','L20-0919 DB A'])
print(studs.collect())
print('After Duplicates Removed:-')
print(studs.distinct().collect())
```

```
['L20-0919 DB A', 'L20-0921 PPIT B', 'L20-0914 DB C', 'I19-0989 DB D', 'K17-0999 OS C', 'K16-1119 DS A', 'L20-0919 DB A']
After Duplicates Removed:-
['L20-0919 DB A', 'L20-0921 PPIT B', 'I19-0989 DB D', 'K17-0999 OS C', 'L20-0914 DB C', 'K16-1119 DS A']
```

In [ ]:
```
# Q5. Find MIN MAX Grades In Each Course
studs = sc.parallelize(['L20-0919 DB A','L20-0921 PPIT B','L20-0914 DB C','I19-0989 DB D','K17-0999 OS C','K16-1119 DS A','L20-0919 DB A'])
def findMinMax(s):
  if(not s):
    return (None,None)
  maxGrade=s[0] #65 = A
  minGrade=s[0] #70 = F
  for grades in s:
    if grades<maxGrade:
      maxGrade=grades
    if grades>minGrade:
      minGrade=grades

  return (maxGrade,minGrade)

studs.map(lambda x: (x.split(' ')[1],x.split(' ')[2])).groupByKey().mapValues(lambda x: findMinMax(list(x))).collect()
```

Out[ ]:
```
[('DB', ('A', 'D')),
 ('OS', ('C', 'C')),
 ('PPIT', ('B', 'B')),
 ('DS', ('A', 'A'))]
```

In [ ]:
```
# Q6 Join and It Types Exporation
studsA = sc.parallelize([('0919','A'),('0921','B'),('0919','B')])
studsB = sc.parallelize([('0919','A-'),('0911','B')])
print('Students A Group: ',studsA.collect())
print('Students B Group: ',studsB.collect())
print('----------------------JOIN & ITS TYPES RESULTS--------------------')
print('Simple Join: ',studsA.join(studsB).collect())
print('Left Outer Join: ',studsA.leftOuterJoin(studsB).collect())
print('Right Outer Join: ',studsA.rightOuterJoin(studsB).collect())
res=studsA.cogroup(studsB).collect()
print('Full Outer Join / cogroup: ')
for it,lis in res:
  print(it)
  for it2 in lis:
    print(list(it2))
```

```
Students A Group:  [('0919', 'A'), ('0921', 'B'), ('0919', 'B')]
Students B Group:  [('0919', 'A-'), ('0911', 'B')]
----------------------JOIN & ITS TYPES RESULTS--------------------
Simple Join:  [('0919', ('A', 'A-')), ('0919', ('B', 'A-'))]
Left Outer Join:  [('0921', ('B', None)), ('0919', ('A', 'A-')), ('0919', ('B', 'A-'))]
Right Outer Join:  [('0911', (None, 'B')), ('0919', ('A', 'A-')), ('0919', ('B', 'A-'))]
Full Outer Join / cogroup:
0921
['B']
[]
0911
[]
['B']
0919
['A', 'B']
['A-']
```

Practice Slides Questions [Not Done In Class]

In [ ]:
```
# Q1. 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)
studs = sc.parallelize([
'L22-2100 DB D',
'I21-1601 SE B',
'I21-1601 OS F',
'I21-1601 DS A',
'L22-2100 DS B'])
def convertToPoint(grade):
  if(grade=='A'):
    return 4
  elif(grade=='B'):
    return 3
  elif(grade=='C'):
    return 2
  elif(grade=='D'):
    return 1
  else:
    return 0

valuesWithGradePoints=studs.map(lambda x: (x.split(' ')[0],convertToPoint(x.split(' ')[2])))
fullRecord=valuesWithGradePoints.map(lambda x:(x[0],(x[1],1)))
fullRecordTwo=fullRecord.reduceByKey(lambda x,y:(x[0]+y[0],x[1]+y[1]))
fullRecordTwo.mapValues(lambda x:'GPA: '+str(x[0]/x[1])).collect()
```

Out[ ]:
```
[('L22-2100', 'GPA: 2.0'), ('I21-1601', 'GPA: 2.3333333333333335')]
```

In [ ]:
```
# Q2. Convert grades to GPA as mentioned above and find the average GPA of each Subject
studs = sc.parallelize([
'L22-2100 DB D',
'I21-1601 SE B',
'I21-1601 OS F',
'I21-1601 DS A',
'L22-2100 DS B'])
def convertToPoint(grade):
  if(grade=='A'):
    return 4
  elif(grade=='B'):
    return 3
  elif(grade=='C'):
    return 2
  elif(grade=='D'):
    return 1
  else:
    return 0

valuesWithGradePoints=studs.map(lambda x: (x.split(' ')[1],convertToPoint(x.split(' ')[2])))
fullRecord=valuesWithGradePoints.map(lambda x:(x[0],(x[1],1)))
fullRecordTwo=fullRecord.reduceByKey(lambda x,y:(x[0]+y[0],x[1]+y[1]))
fullRecordTwo.mapValues(lambda x:'GPA: '+str(x[0]/x[1])).collect()
```

Out[ ]:
```
[('DB', 'GPA: 1.0'),
 ('OS', 'GPA: 0.0'),
 ('SE', 'GPA: 3.0'),
 ('DS', 'GPA: 3.5')]
```

In [ ]:
```
# Q3. 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
# b. For year follow the rule of year 16<17 and 99 < 01
# c. For the last part of roll-number, follow int ordering.roll no= L21-1705

studs = sc.parallelize([
'L22-2100 DB D',
'K01-1601 SE F',
'I21-1601 OS F',
'K96-1702 DS B',
'L21-1705 OS A',
'L22-2101 DB D',
'K98-1601 OS F',
'L99-1601 SE F',
'L11-1702 SE B',
'L17-1705 DB A',])

def sortingKey(record):
    campus = record[0]
    year = int(record.split(' ')[0].split('-')[0][1:])
    year = year - 100 if year > 24 else year
    lastPart=record.split(' ')[0].split('-')[1]
    campusOrder={'F':0,'I':1,'K':2 ,'L':3,'P':4}

    return (campusOrder[campus], year,lastPart)


studs.map(lambda x: (sortingKey(x), x)).sortByKey().values().collect()
```

Out[ ]:
```
['I21-1601 OS F',
 'K96-1702 DS B',
 'K98-1601 OS F',
 'K01-1601 SE F',
 'L99-1601 SE F',
 'L11-1702 SE B',
 'L17-1705 DB A',
 'L21-1705 OS A',
 'L22-2100 DB D',
 'L22-2101 DB D']
```

In [5]:
```
# Q4 (Self Made): Find the most commonly occuring grade for each course

studs = sc.parallelize([
'L22-2100 DB D',
'K01-1601 SE F',
'I21-1601 OS F',
'K96-1702 DS B',
'L21-1705 OS A',
'L22-2101 DB D',
'K98-1601 OS F',
'L99-1601 SE F',
'L11-1702 SE B',
'L17-1705 DB A',])

def findMax(countGrade):
  countGrade=list(countGrade)
  maxCount=0
  maxGrade='-'
  for value,grade in countGrade:
    if(value>maxCount):
      maxCount=value
      maxGrade=grade

  return maxGrade

studs.map(lambda x: ((x.split(' ')[1],x.split(' ')[2]), 1)).reduceByKey(lambda x,y:x+y).map(lambda x: (x[0]
[0],(x[1],x[0][1]))).groupByKey().mapValues(lambda x: findMax(x)).collect()
```

Out[5]:
```
[('DB', 'D'), ('OS', 'F'), ('SE', 'F'), ('DS', 'B')]
```

In [6]:
```
# Q4 (Self Made): Find the most commonly occuring grade for each course [VERSION 2]

studs = sc.parallelize([
'L22-2100 DB D',
'K01-1601 SE F',
'I21-1601 OS F',
'K96-1702 DS B',
'L21-1705 OS A',
'L22-2101 DB D',
'K98-1601 OS F',
'L99-1601 SE F',
'L11-1702 SE B',
'L17-1705 DB A',])

def findMax(grades):
  grades=list(grades)
  gradesInfo={}

  for grade in grades:
    gradesInfo.setdefault(grade,0)
    gradesInfo[grade]=gradesInfo[grade]+1

  maxCount=0
  maxGrade='-'
  for grade,value in gradesInfo.items():
    if(value>maxCount):
      maxCount=value
      maxGrade=grade

  return maxGrade

studs.map(lambda x: (x.split(' ')[1],x.split(' ')[2])).groupByKey().mapValues(lambda x: findMax(x)).collect()
```

Out[6]:
```
[('DB', 'D'), ('OS', 'F'), ('SE', 'F'), ('DS', 'B')]
```

Explore the SPARKcluster UI (user-interface)

In [7]:
```
!pip install pyngrok
```

```
Collecting pyngrok
  Downloading pyngrok-7.1.2-py3-none-any.whl (22 kB)
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.10/dist-packages (from pyngrok) (6.0.1)
Installing collected packages: pyngrok
Successfully installed pyngrok-7.1.2
```

In [8]:
```
from pyngrok import ngrok, conf
import getpass

# Set Ngrok authtoken
print("Enter your authtoken, which can be copied from https://dashboard.ngrok.com/auth")
conf.get_default().auth_token = getpass.getpass()

# Define the port
ui_port = 4050

# Connect to Ngrok and get the public URL
try:
    public_url = ngrok.connect(ui_port).public_url
    print(f" * Ngrok tunnel created: {public_url} -> http://127.0.0.1:{ui_port}")
except Exception as e:
    print(f"Error creating Ngrok tunnel: {e}")

# My Authentication Token 2cSK5j3NB6McxNBPb9wFQQfF2MW_6njVGDJ7hkV1W6e9B7v9F
```

```
Enter your authtoken, which can be copied from https://dashboard.ngrok.com/auth
··········
 * Ngrok tunnel created: https://e6b5-34-125-215-18.ngrok-free.app -> http://127.0.0.1:4050
```

In [ ]:
```

```