== Programming Question ==

You are given a CSV file of the form "studentID,courseID,marks" for students in a university. Provide the Spark transformations and actions using RDDs to

- (i) calculate the average marks per course,
- (ii) average marks per student and
- (iii) the number of students in the university.

In [1]:
```
!pip -qq install pyspark
```

```
     |████████████████████████████████| 281.3 MB 36 kB/s
     |████████████████████████████████| 199 kB 37.6 MB/s
  Building wheel for pyspark (setup.py) ... done
```

In [2]:
```
import numpy as np
import pandas as pd
from pyspark.sql import SparkSession
from pyspark.sql.functions import count, avg
from pyspark.sql.functions import *
```

In [3]:
```
# Create a dummy students csv file

df = pd.DataFrame({"studentID": np.random.randint(low = 101, high = 130, size = 50, ),
                   "courseID": np.random.randint(low = 1001, high = 1009, size = 50),
                   "marks": np.random.randint(low = 60, high = 95, size = 50)})
df['studentID'] = df['studentID'].apply(lambda x: "S"+str(x))
df['courseID'] = df['courseID'].apply(lambda x: "C"+str(x))

# Drop combined duplicate entries for 'studentID', 'courseID'
df = df.iloc[df[['studentID', 'courseID']].drop_duplicates().index]

df.head()
```

Out[3]:

	studentID	courseID	marks
0	S113	C1002	89
1	S120	C1004	76
2	S123	C1006	82
3	S129	C1008	61
5	S120	C1008	88

In [4]:
```
df.shape
```

Out[4]:
```
(46, 3)
```

In [5]:
```
# Save to csv file
df.to_csv("students_dataset.csv", index=False)
```

Using RDDs

In [6]:
```
# Create a spark session
spark = (SparkSession.builder.appName("StudentsData").getOrCreate())
spark
```

Out[6]:

SparkSession - in-memory

SparkContext

Spark UI

Version	`v3.3.0`
Master	`local[*]`
AppName	`StudentsData`

In [7]:
```
# Accessing sparkContext from sparkSession instance
sc = spark.sparkContext
```

In [8]:
```
# Read data and create rdd
dataset = sc.textFile("students_dataset.csv")
type(dataset)
```

Out[8]:
```
pyspark.rdd.RDD
```

In [9]:
```
# Filter out header row
header = dataset.first()      # extract header
rdd_data = dataset.filter(lambda x: x != header)
type(rdd_data)
```

Out[9]:
```
pyspark.rdd.PipelinedRDD
```

In [10]:
```
rdd_data.count()
```

Out[10]:
```
46
```

In [11]:
```
# we are creating a new RDD called “rows” by splitting every row in the rdd_data
rows = rdd_data.map(lambda line: line.split(","))
type(rows)
```

Out[11]:
```
pyspark.rdd.PipelinedRDD
```

In [12]:
```
rows.count()
```

Out[12]:
```
46
```

In [13]:
```
# Print Data values
for row in rows.take(rows.count()):
  print(row)
```

```
['S113', 'C1002', '89']
['S120', 'C1004', '76']
['S123', 'C1006', '82']
['S129', 'C1008', '61']
['S120', 'C1008', '88']
['S117', 'C1005', '81']
['S109', 'C1001', '79']
['S108', 'C1007', '62']
['S106', 'C1008', '81']
['S110', 'C1003', '80']
['S115', 'C1007', '61']
['S105', 'C1008', '67']
['S120', 'C1001', '73']
['S101', 'C1007', '62']
['S124', 'C1001', '86']
['S128', 'C1003', '75']
['S123', 'C1008', '64']
['S111', 'C1001', '89']
['S105', 'C1007', '94']
['S111', 'C1003', '63']
['S114', 'C1002', '89']
['S114', 'C1008', '70']
['S117', 'C1008', '83']
['S116', 'C1004', '72']
['S125', 'C1002', '83']
['S126', 'C1006', '73']
['S101', 'C1006', '72']
['S124', 'C1002', '71']
['S127', 'C1005', '69']
['S127', 'C1002', '79']
['S105', 'C1001', '86']
['S122', 'C1003', '67']
['S112', 'C1005', '81']
['S108', 'C1001', '87']
['S128', 'C1001', '61']
['S116', 'C1007', '93']
['S127', 'C1008', '89']
['S111', 'C1008', '64']
['S104', 'C1005', '62']
['S109', 'C1003', '93']
['S110', 'C1006', '75']
['S118', 'C1001', '68']
['S110', 'C1008', '65']
['S106', 'C1007', '84']
['S117', 'C1004', '85']
['S122', 'C1006', '90']
```

(i) calculate the average marks per course

In [14]:
```
# Extract courseID and marks from rdd
course_marks = rows.map(lambda x: (x[1], x[2]))
type(course_marks)
```

Out[14]:
```
pyspark.rdd.PipelinedRDD
```

In [15]:
```
course_marks.take(5)
```

Out[15]:
```
[('C1002', '89'),
 ('C1004', '76'),
 ('C1006', '82'),
 ('C1008', '61'),
 ('C1008', '88')]
```

In [16]:
```
# Store the count of marks associated with each courseID
course_count = course_marks.countByKey()
course_count
```

Out[16]:
```
defaultdict(int,
            {'C1001': 8,
             'C1002': 5,
             'C1003': 5,
             'C1004': 3,
             'C1005': 4,
             'C1006': 5,
             'C1007': 6,
             'C1008': 10})
```

In [17]:
```
# Calculate total marks per course
course_marks_sum = course_marks.reduceByKey(lambda x,y: float(x) + float(y))
course_marks_sum.collect()
```

Out[17]:
```
[('C1002', 411.0),
 ('C1006', 392.0),
 ('C1004', 233.0),
 ('C1008', 732.0),
 ('C1005', 293.0),
 ('C1001', 629.0),
 ('C1007', 456.0),
 ('C1003', 378.0)]
```

In [18]:
```
# Calculate average marks per course
avg_mrks_per_course_rdd = course_marks_sum.map(lambda x: (x[0], float(x[1])/course_cou
sorted(avg_mrks_per_course_rdd.collect())
```

Out[18]:
```
[('C1001', 78.625),
 ('C1002', 82.2),
 ('C1003', 75.6),
 ('C1004', 77.66666666666667),
 ('C1005', 73.25),
 ('C1006', 78.4),
 ('C1007', 76.0),
 ('C1008', 73.2)]
```

(ii) average marks per student

In [19]:
```
# Extract studentID and marks from rdd
stdnt_marks = rows.map(lambda x: (x[0], x[2]))
type(stdnt_marks)
```

Out[19]:
```
pyspark.rdd.PipelinedRDD
```

In [20]:
```
stdnt_marks.take(5)
```

Out[20]:
```
[('S113', '89'),
 ('S120', '76'),
 ('S123', '82'),
 ('S129', '61'),
 ('S120', '88')]
```

In [21]:
```
# Store the count of marks associated with each studentID
stdnt_count = stdnt_marks.countByKey()
stdnt_count
```

Out[21]:
```
defaultdict(int,
            {'S101': 2,
             'S104': 1,
             'S105': 3,
             'S106': 2,
             'S108': 2,
             'S109': 2,
             'S110': 3,
             'S111': 3,
             'S112': 1,
             'S113': 1,
             'S114': 2,
             'S115': 1,
             'S116': 2,
             'S117': 3,
             'S118': 1,
             'S120': 3,
             'S122': 2,
             'S123': 2,
             'S124': 2,
             'S125': 1,
             'S126': 1,
             'S127': 3,
             'S128': 2,
             'S129': 1})
```

In [22]:
```
# Calculate total marks per student
stdnt_marks_sum = stdnt_marks.reduceByKey(lambda x,y: float(x) + float(y))
stdnt_marks_sum.collect()
```

Out[22]:
```
[('S123', 146.0),
 ('S129', '61'),
 ('S108', 149.0),
 ('S106', 165.0),
 ('S110', 220.0),
 ('S105', 247.0),
 ('S101', 134.0),
 ('S124', 157.0),
 ('S128', 136.0),
 ('S111', 216.0),
 ('S125', '83'),
 ('S113', '89'),
 ('S120', 237.0),
 ('S117', 249.0),
 ('S109', 172.0),
 ('S115', '61'),
 ('S114', 159.0),
 ('S116', 165.0),
 ('S126', '73'),
 ('S127', 237.0),
 ('S122', 157.0),
 ('S112', '81'),
 ('S104', '62'),
 ('S118', '68')]
```

In [23]:
```
# Calculate average marks per student
avg_mrks_per_stdnt_rdd = stdnt_marks_sum.map(lambda x: (x[0], float(x[1])/stdnt_count[
sorted(avg_mrks_per_stdnt_rdd.collect())
```

Out[23]:
```
[('S101', 67.0),
 ('S104', 62.0),
 ('S105', 82.33333333333333),
 ('S106', 82.5),
 ('S108', 74.5),
 ('S109', 86.0),
 ('S110', 73.33333333333333),
 ('S111', 72.0),
 ('S112', 81.0),
 ('S113', 89.0),
 ('S114', 79.5),
 ('S115', 61.0),
 ('S116', 82.5),
 ('S117', 83.0),
 ('S118', 68.0),
 ('S120', 79.0),
 ('S122', 78.5),
 ('S123', 73.0),
 ('S124', 78.5),
 ('S125', 83.0),
 ('S126', 73.0),
 ('S127', 79.0),
 ('S128', 68.0),
 ('S129', 61.0)]
```

(iii) the number of students in the university

In [24]:
```
# Store the count of marks associated with each studentID
stdnt_count = stdnt_marks.countByKey()
n_students = len(stdnt_count.keys())
```

```
print("Total number of students in the university = ", n_students)
```

```
Total number of students in the university =  24
```

Using Spark SQL

In [25]:
```
# Read data
st_df = (spark.read.format("csv").option("header", "true").option("inferSchema", "true
```

In [26]:
```
# (i) calculate the average marks per course
avg_mrks_per_course = (st_df.select("courseID", "Marks").groupBy("courseID").agg(avg("
avg_mrks_per_course.show(truncate=False)
```

```
+--------+-----------------+
|courseID|Avg_Marks        |
+--------+-----------------+
|C1002   |82.2             |
|C1001   |78.625           |
|C1006   |78.4             |
|C1004   |77.66666666666667|
|C1007   |76.0             |
|C1003   |75.6             |
|C1005   |73.25            |
|C1008   |73.2             |
+--------+-----------------+
```

In [27]:
```
# (ii) average marks per student
avg_mrks_per_stdnt = (st_df.select("studentID", "Marks").groupBy("studentID").agg(avg(
avg_mrks_per_stdnt.show(truncate=False)
```

```
+---------+-----------------+
|studentID|Avg_Marks        |
+---------+-----------------+
|S113     |89.0             |
|S109     |86.0             |
|S125     |83.0             |
|S117     |83.0             |
|S106     |82.5             |
|S116     |82.5             |
|S105     |82.33333333333333|
|S112     |81.0             |
|S114     |79.5             |
|S120     |79.0             |
|S127     |79.0             |
|S122     |78.5             |
|S124     |78.5             |
|S108     |74.5             |
|S110     |73.33333333333333|
|S126     |73.0             |
|S123     |73.0             |
|S111     |72.0             |
|S118     |68.0             |
|S128     |68.0             |
+---------+-----------------+
only showing top 20 rows
```

In [28]:
```
#  (iii) the number of students in the university
n_stdnts = st_df.select("studentID").distinct().count()
print("Total number of students in the university = ", n_stdnts)
```

```
Total number of students in the university =  24
```

In [ ]:
```

```