

# RDD

## Filter() and Map()

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Airport id | Name airport | City name | Country of airport | IATA/FAA code | ICAO code | Latitude | Longitude | Altitude | Timezome DST | Timezone in Olson |

import re

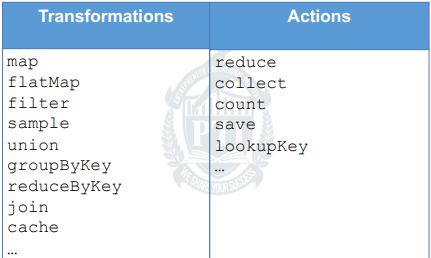
class Utils():

COMMA\_DELIMITER = re.compile(''',(?=(?:[^"]\*"[^"]\*")\*[^"]\*$)''')

|  |  |
| --- | --- |
| Bài 1: find all the airports which are located in United States and output the airport's name and the city's name to out/airports\_in\_usa.text.  import sys  sys.path.insert(0, '.')  from pyspark import SparkContext, SparkConf  from commons.Utils import Utils  def splitComma(line: str):  splits = Utils.COMMA\_DELIMITER.split(line)  return "{}, {}".format(splits[1], splits[2])  if \_\_name\_\_ == "\_\_main\_\_":  conf = SparkConf().setAppName("airports").setMaster("local[\*]")  sc = SparkContext(conf = conf)  airports = sc.textFile("in/airports.text")  airportsInUSA = airports.filter(lambda line : Utils.COMMA\_DELIMITER.split(line)[3] == "\"United States\"")  airportsNameAndCityNames = airportsInUSA.map(splitComma)  '''  map airport's name and the city's name  '''  airportsNameAndCityNames.saveAsTextFile("out/airports\_in\_usa.text") | Bài 2: tìm tất cả các sân bay có vĩ độ lớn hơn 40, sau đó xuất tên sân bay và vĩ độ của sân bay đó ra / airport\_by\_latitude.text.  import sys  sys.path.insert(0, '.')  from pyspark import SparkContext, SparkConf  from commons.Utils import Utils  def splitComma(line: str):  splits = Utils.COMMA\_DELIMITER.split(line)  return "{}, {}".format(splits[1], splits[6])  if \_\_name\_\_ == "\_\_main\_\_":  conf = SparkConf().setAppName("airports").setMaster("local[\*]")  sc = SparkContext(conf = conf)    airports = sc.textFile("in/airports.text")  airportsInUSA = airports.filter(lambda line: float(Utils.COMMA\_DELIMITER.split(line)[6]) > 40)    airportsNameAndCityNames = airportsInUSA.map(splitComma)  airportsNameAndCityNames.saveAsTextFile("out/airports\_by\_latitude.text") |

## flatMap()

//WordCount

from pyspark import SparkContext, SparkConf

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

conf = SparkConf().setAppName("word count").setMaster("local[3]")

sc = SparkContext(conf = conf)

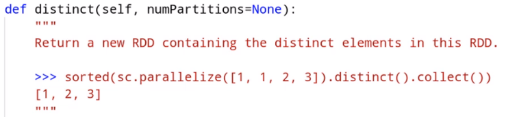
lines = sc.textFile("in/word\_count.text")

words = lines.flatMap(lambda line: line.split(" "))

wordCounts = words.countByValue()

for word, count in wordCounts.items():

print("{} : {}".format(word, count))



# Set Operations

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Host | Logname time | Method | url | Response | bytes |

|  |  |
| --- | --- |
| Bài 1: **new RDD which contains the log lines from both July 1st and August 1st**  **from pyspark import SparkContext, SparkConf**  def isNotHeader(line: str):  return not (line.startswith("host") and "bytes" in line)  if \_\_name\_\_ == "\_\_main\_\_":  conf = SparkConf().setAppName("unionLogs").setMaster("local[\*]")  sc = SparkContext(conf = conf)  julyFirstLogs = sc.textFile("in/nasa\_19950701.tsv")  augustFirstLogs = sc.textFile("in/nasa\_19950801.tsv")  aggregatedLogLines = julyFirstLogs.**union**(augustFirstLogs)  cleanLogLines = aggregatedLogLines.filter(isNotHeader)  sample = cleanLogLines.sample(withReplacement = True, fraction = 0.1)  sample.saveAsTextFile("out/sample\_nasa\_logs.csv") | **Bài 2: new RDD which contains the hosts which are accessed on BOTH days**  **from pyspark import SparkContext, SparkConf**  if \_\_name\_\_ == "\_\_main\_\_":  conf = SparkConf().setAppName("sameHosts").setMaster("local[1]")  sc = SparkContext(conf = conf)  julyFirstLogs = sc.textFile("in/nasa\_19950701.tsv")  augustFirstLogs = sc.textFile("in/nasa\_19950801.tsv")    julyFirstHosts = julyFirstLogs.map(lambda line: line.split("\t")[0])  augustFirstHosts = augustFirstLogs.map(lambda line: line.split("\t")[0])    intersection = julyFirstHosts.**intersection**(augustFirstHosts)    cleanedHostIntersection = intersection.filter(lambda host: host != "host")  cleanedHostIntersection.saveAsTextFile("out/nasa\_logs\_same\_hosts.csv") |

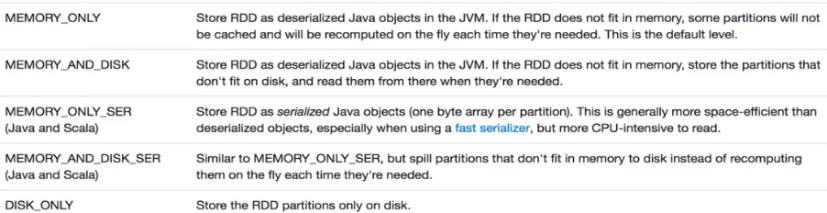
# Action

Bài 1: **Mỗi hàng 10 số. Viết hàm đọc và tính tổng các số**

from pyspark import SparkContext, SparkConf

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

conf = SparkConf().setAppName("primeNumbers").setMaster("local[\*]")

 sc = SparkContext(conf = conf)

lines = sc.textFile("in/prime\_nums.text")

numbers = lines.flatMap(lambda line: line.split("\t"))

validNumbers = numbers.filter(lambda number: number)

intNumbers = validNumbers.map(lambda number: int(number))

print("Sum is: {}".format(intNumbers.reduce(lambda x, y: x + y)))

integerRdd.persist(StorageLevel.MEMORY\_ONLY)

# Pair RDD

# Create pair RDD

|  |  |
| --- | --- |
| C1:pairRDD = sc.parallelize([("Lily", 23), ("Jack", 29), ("Mary", 29), ("James", 8)]) | C2: inputStrings = ["Lily 23", "Jack 29", "Mary 29", "James 8"]  regularRDDs = sc.parallelize(inputStrings)  pairRDD = regularRDDs.map(lambda s: (s.split(" ")[0], s.split(" ")[1])) |

# Filter() and mapValue()

Bài airport

|  |  |
| --- | --- |
| Bài 1: tạo **một cặp RDD với tên sân bay là khóa và tên quốc gia là giá trị. Sau đó, xóa tất cả các sân bay nằm ở Hoa Kỳ và xuất cặp RDD ra**  import sys  sys.path.insert(0, '.')  from pyspark import SparkContext, SparkConf  from commons.Utils import Utils  if \_\_name\_\_ == "\_\_main\_\_":  conf = SparkConf().setAppName("airports").setMaster("local[\*]")  sc = SparkContext(conf = conf)  airportsRDD = sc.textFile("in/airports.text")  airportPairRDD = airportsRDD.**map**(lambda line: \  (Utils.COMMA\_DELIMITER.split(line)[1],  Utils.COMMA\_DELIMITER.split(line)[3]))  airportsNotInUSA = airportPairRDD.**filter**(lambda keyValue: keyValue[1] != "\"United States\"")  airportsNotInUSA.saveAsTextFile("out/airports\_not\_in\_usa\_pair\_rdd.text") | Bài 2**: tạo một cặp RDD với tên sân bay là khóa và tên quốc gia là giá trị. Sau đó chuyển đổi tên quốc gia thành chữ hoa và xuất cặp RDD ra**  import sys  sys.path.insert(0, '.')  from pyspark import SparkContext, SparkConf  from commons.Utils import Utils  if \_\_name\_\_ == "\_\_main\_\_":  conf = SparkConf().setAppName("airports").setMaster("local[\*]")  sc = SparkContext(conf = conf)  airportsRDD = sc.textFile("in/airports.text")  airportPairRDD = airportsRDD.**map**(lambda line: \  (Utils.COMMA\_DELIMITER.split(line)[1], \  Utils.COMMA\_DELIMITER.split(line)[3]))  upperCase = airportPairRDD.mapValues(lambda countryName: countryName.**upper**())  upperCase.saveAsTextFile("out/airports\_uppercase.text") |

# ReduceByKey()

|  |  |
| --- | --- |
| Bài 1: **WordCount**  from pyspark import SparkContext, SparkConf  if \_\_name\_\_ == "\_\_main\_\_":  conf = SparkConf().setAppName("wordCounts").setMaster("local[3]")  sc = SparkContext(conf = conf)  lines = sc.textFile("in/word\_count.text")  wordRdd = lines.flatMap(lambda line: line.split(" "))  wordPairRdd = wordRdd.map(lambda word: (word, 1))  wordCounts = wordPairRdd.reduceByKey(lambda x, y: x + y)  for word, count in wordCounts.collect():  print("{} : {}".format(word, count))  **1. MLS: Nhiều số dịch vụ niêm yết cho ngôi nhà (ID duy nhất).**  **2. Vị trí: thành phố / thị trấn nơi ngôi nhà tọa lạc. Hầu hết các địa điểm đều ở Hạt San Luis Obispo và hạt phía bắc Santa Barbara (Santa MariaOrcutt, Lompoc,Guadelupe, Los Alamos), nhưng cũng có một số địa điểm ngoài khu vực.**  **3. Giá: giá niêm yết gần đây nhất của ngôi nhà (tính bằng đô la).**  **4. Phòng ngủ: số phòng ngủ.**  **5. Phòng tắm: số lượng phòng tắm.**  **6. Kích thước: kích thước của ngôi nhà tính bằng feet vuông.**  **7. Price / SQ.ft: giá của ngôi nhà trên mỗi foot vuông.**  //của bài 2  class AvgCount():  def \_\_init\_\_(self, count: int, total: float):  self.count = count  self.total = total | Bài 2: **đưa ra mức giá trung bình cho những ngôi nhà có số lượng phòng ngủ khác nhau.**  import sys  sys.path.insert(0, '.')  from pyspark import SparkContext, SparkConf  from pairRdd.aggregation.reducebykey.housePrice.AvgCount import AvgCount  if \_\_name\_\_ == "\_\_main\_\_":  conf = SparkConf().setAppName("avgHousePrice").setMaster("local[3]")  sc = SparkContext(conf = conf)  lines = sc.textFile("in/RealEstate.csv")  cleanedLines = lines.filter(lambda line: "Bedrooms" not in line)  housePricePairRdd = cleanedLines.map(lambda line: \  (line.split(",")[3], AvgCount(1, float(line.split(",")[2]))))  housePriceTotal = housePricePairRdd \  .reduceByKey(lambda x, y: AvgCount(x.count + y.count, x.total + y.total))  print("housePriceTotal: ")  for bedroom, avgCount in housePriceTotal.collect():  print("{} : ({}, {})".format(bedroom, avgCount.count, avgCount.total))  housePriceAvg = housePriceTotal.mapValues(lambda avgCount: avgCount.total / avgCount.count)  print("\nhousePriceAvg: ")  for bedroom, avg in housePriceAvg.collect():  print("{} : {}".format(bedroom, avg)) |

|  |  |
| --- | --- |
| groupByKey()  Bài 1: **danh sách tên của các sân bay nằm ở mỗi quốc gia.**  import sys  sys.path.insert(0, '.')  from pyspark import SparkContext, SparkConf  from commons.Utils import Utils  if \_\_name\_\_ == "\_\_main\_\_":  conf = SparkConf().setAppName("airports").setMaster("local[\*]")  sc = SparkContext(conf = conf)  lines = sc.textFile("in/airports.text")  countryAndAirportNameAndPair = lines.map(lambda airport:\  (Utils.COMMA\_DELIMITER.split(airport)[3],  Utils.COMMA\_DELIMITER.split(airport)[1]))  airportsByCountry = countryAndAirportNameAndPair.**groupByKey**()  for country, airportName in airportsByCountry.collectAsMap().items():  print("{}: {}".format(country, list(airportName))) | SortByKey()  **Bài 2:Bài sắp xếp word count**  lines = sc.textFile("in/word\_count.text")  wordRdd = lines.flatMap(lambda line: line.split(" "))  wordPairRdd = wordRdd.map(lambda word: (word, 1))  wordToCountPairs = wordPairRdd.reduceByKey(lambda x, y: x + y)  sortedWordCountPairs = wordToCountPairs \  .sortBy(lambda wordCount: wordCount[1], ascending=False)  **Bai3: WordCount kí tự đầu**  val words = input.flatMap(word => word.split(","))  // "Liverpool","100","Red","Leads United", etc. -- includes numbers  val wCount = words.filter(word => Character.isLetter(word.head)) // ignores numbers  .map(word => (word.head, 1)) // gets the first letter of each word  val result = wCount.reduceByKey((x, y) => x + y)  result.collect().foreach(println) |

|  |  |
| --- | --- |
| **Thống kê các giá trị**  import sys  sys.path.insert(0, '.')  from pyspark import SparkContext, SparkConf  from commons.Utils import Utils  if \_\_name\_\_ == "\_\_main\_\_":  conf = SparkConf().setAppName('StackOverFlowSurvey').setMaster("local[\*]")  sc = SparkContext(conf = conf)  total = sc.accumulator(0)  missingSalaryMidPoint = sc.accumulator(0)  responseRDD = sc.textFile("in/2016-stack-overflow-survey-responses.csv")  def filterResponseFromCanada(response):  splits = Utils.COMMA\_DELIMITER.split(response)  total.add(1)  if not splits[14]:  missingSalaryMidPoint.add(1)  return splits[2] == "Canada"  responseFromCanada = responseRDD.filter(filterResponseFromCanada)  print("Bao cao o Canada: {}".format(responseFromCanada.count()))  print("Tong bao cao: {}".format(total.value))  print("Thieu gia tri: {}" .format(missingSalaryMidPoint.value)) | **Join bảng**  from pyspark.sql import SparkSession, functions as fs  if \_\_name\_\_ == "\_\_main\_\_":  session = SparkSession.builder.appName("UkMakerSpaces").master("local[\*]").getOrCreate()  makerSpace = session.read.option("header", "true") \  .csv("in/uk-makerspaces-identifiable-data.csv")  postCode = session.read.option("header", "true").csv("in/uk-postcode.csv") \  .withColumn("PostCode", fs.concat\_ws("", fs.col("PostCode"), fs.lit(" ")))  print("=== Print 20 records of makerspace table ===")  makerSpace.select("Name of makerspace", "Postcode").show()  print("=== Print 20 records of postcode table ===")  postCode.select("PostCode", "Region").show()  joined = makerSpace \  .join(postCode, makerSpace["Postcode"].startswith(postCode["Postcode"]), "left\_outer")  print("=== Group by Region ===")  joined.groupBy("Region").count().show(200) |

1. phát biểu nào sau đây sai về kafka

b, message sau khi được tiêu thụ (consume) thì không bị xóa.

1. Phát biểu nào đúng về Quorum trong Amazon DynamoDB

b, với N là tổng số nhân bản, R là số nhân bản cần đọc trong 1 thao tác đọc. W là số nhân bản cần ghi trong 1 thao tác ghi. N < R + W

1. Phát biểu nào sau đây sai về Kafka

d, 1 message chỉ có thể được đọc bởi 1 consumer trong 1 consumer group

1. phát biểu nào sai về Presto

d, presto được quản lý bởi apache software foundation

1. Phát biểu nào đúng về Presto

a, các stage được thực thi theo cơ chế pipeline, không có thời gian chờ giữa các stage như Map Reduce

1. Giữa Pig và Hive, công cụ nào có giao diện truy vấn gắn với ANSI SQL hơn
2. Pig
3. không phải 2 đáp án trên
4. Hive
5. Công ty nào đã phát triển Apache Cassandra giai đoạn đầu tiên

d, facebook

# SparkSQL

|  |  |
| --- | --- |
| from pyspark.sql import SparkSession  AGE\_MIDPOINT = "age\_midpoint"  SALARY\_MIDPOINT = "salary\_midpoint"  SALARY\_MIDPOINT\_BUCKET = "salary\_midpoint\_bucket"  if \_\_name\_\_ == "\_\_main\_\_":  session = SparkSession.builder.appName("StackOverFlowSurvey").getOrCreate()  dataFrameReader = session.read  responses = dataFrameReader \  .option("header", "true") \  .option("inferSchema", value = True) \  .csv("in/2016-stack-overflow-survey-responses.csv")  print("=== Print out schema ===")  responses.printSchema()    responseWithSelectedColumns = responses.select("country", "occupation",  AGE\_MIDPOINT, SALARY\_MIDPOINT)  print("=== 20 row cua Dataframe ===")  responseWithSelectedColumns.show()  print("=== Response o Afghanistan ===")  responseWithSelectedColumns.filter(responseWithSelectedColumns["country"] == "Afghanistan").show()  print("=== Tong so nguoi o moi nghe ===")  groupedData = responseWithSelectedColumns.groupBy("occupation")  groupedData.count().show()  print("=== Print records with average mid age less than 20 ===")  responseWithSelectedColumns.filter(responseWithSelectedColumns[AGE\_MIDPOINT] < 20).show()  print("=== Giam dan SALARY ===")  responseWithSelectedColumns.orderBy(responseWithSelectedColumns[SALARY\_MIDPOINT], ascending = False).show()  print("===Luong trung binh o moi noi ===")  dataGroupByCountry = responseWithSelectedColumns.groupBy("country")  dataGroupByCountry.avg(SALARY\_MIDPOINT).show()  responseWithSalaryBucket = responses.withColumn(SALARY\_MIDPOINT\_BUCKET,  ((responses[SALARY\_MIDPOINT]/20000).cast("integer")\*20000))  print("=== With salary bucket column ===")  responseWithSalaryBucket.select(SALARY\_MIDPOINT, SALARY\_MIDPOINT\_BUCKET).show()  print("===Bao nhieu nguoi co luong duoi x ===")  responseWithSalaryBucket \  .groupBy(SALARY\_MIDPOINT\_BUCKET) \  .count() \  .orderBy(SALARY\_MIDPOINT\_BUCKET) \  .show()  session.stop() | Bài 2: **Bài house. Nhóm theo địa điểm, tính giá trị TB của (tiền/m2) và sắp xếp**  1. MLS: Multiple listing service number for the house (unique ID).  2. Location: city/town where the house is located. Most locations are in  San Luis Obispo county and northern Santa Barbara county (Santa Maria­Orcutt, Lompoc,  Guadelupe, Los Alamos), but there some out of area locations as well.  3. Price: the most recent listing price of the house (in dollars).  4. Bedrooms: number of bedrooms.  5. Bathrooms: number of bathrooms.  6. Size: size of the house in square feet.  7. Price/SQ.ft: price of the house per square foot.  8. Status: type of sale. Thee types are represented in the dataset: Short Sale,  Foreclosure and Regular.  Each field is comma separated.  Sample output:  +----------------+-----------------+  | Location| avg(Price SQ Ft)|  +----------------+-----------------+  | Oceano| 95.0|  | Bradley| 206.0|  | San Luis Obispo| 359.0|  | Santa Ynez| 491.4|  | Cayucos| 887.0|  |................|.................|  |................|.................|  |................|.................|  '''  from pyspark.sql import SparkSession  PRICE\_SQ\_FT = "Price SQ Ft"  if \_\_name\_\_ == "\_\_main\_\_":  session = SparkSession.builder.appName("HousePriceSolution").master("local[\*]").getOrCreate()    realEstate = session.read \  .option("header","true") \  .option("inferSchema", value=True) \  .csv("in/RealEstate.csv")  realEstate.groupBy("Location") \  .avg(PRICE\_SQ\_FT) \  .orderBy("avg(Price SQ FT)") \  .show() |

1. Giả sử một textfile kích thước lớn được đặt ở đường dẫn hdfs://user/bigfile.txt. Viết chương trình Spark **đếm số lần xuất hiện của chuỗi “big data processing”** trong file text này.

linesRDD = sc.textFile("hdfs://user/bigfile.txt")

