National University of Computer and Emerging Sciences

Laboratory Manual-10

for

Fundamentals of Big Data Lab

Course Instructor: Dr. Iqra Safdar
Lab Instructors: Mr. Muhammad Mazarib; Mr Muhammad Aiss Shahid
Section: BDS-4B
Date: 19-Apr-2023
Semester: Spring 2023

Department of Computer Science

FAST-NU, Lahore, Pakistan

Table · *Common supported file formats*

Format name	Structured	Comments
Text files	No	Plain old text files. Records are assumed to be one per line.
JSON	Semi	Common text-based format, semistructured; most libraries require one record per line.
CSV	Yes	Very common text-based format, often used with spreadsheet applications.
SequenceFiles	Yes	A common Hadoop file format used for key/value data.
Protocol buffers	Yes	A fast, space-efficient multilanguage format.
Object files	Yes	Useful for saving data from a Spark job to be consumed by shared code. Breaks if you change your classes, as it relies on Java Serialization.

Example · *Loading CSV in full in Python*

```python
def loadRecords(fileNameContents):
    """Load all the records in a given file"""
    input = StringIO.StringIO(fileNameContents[1])
    reader = csv.DictReader(input, fieldnames=["name", "favoriteAnimal"])
    return reader
fullFileData = sc.wholeTextFiles(inputFile).flatMap(loadRecords)
```

Example ·· *Loading CSV with textFile() in Python*

```python
import csv
import StringIO
...
def loadRecord(line):
    """Parse a CSV line"""
    input = StringIO.StringIO(line)
    reader = csv.DictReader(input, fieldnames=["name", "favouriteAnimal"])
    return reader.next()
input = sc.textFile(inputFile).map(loadRecord)
```

Table   Basic RDD transformations on an RDD containing {1, 2, 3, 3}

Function name	Purpose	Example	Result
map()	Apply a function to each element in the RDD and return an RDD of the result.	rdd.map(x => x + 1)	{2, 3, 4, 4}
flatMap()	Apply a function to each element in the RDD and return an RDD of the contents of the iterators returned. Often used to extract words.	rdd.flatMap(x => x.to(3))	{1, 2, 3, 2, 3, 3, 3}
filter()	Return an RDD consisting of only elements that pass the condition passed to filter().	rdd.filter(x => x != 1)	{2, 3, 3}
distinct()	Remove duplicates.	rdd.distinct()	{1, 2, 3}
sample(withReplacement, fraction, [seed])	Sample an RDD, with or without replacement.	rdd.sample(false, 0.5)	Nondeterministic

Table   Two-RDD transformations on RDDs containing {1, 2, 3} and {3, 4, 5}

| Function name	Purpose	Example	Result
union()	Produce an RDD containing elements from both RDDs.	rdd.union(other)	{1, 2, 3, 3, 4, 5}
intersection()	RDD containing only elements found in both RDDs.	rdd.intersection(other)	{3}
subtract()	Remove the contents of one RDD (e.g., remove training data).	rdd.subtract(other)	{1, 2}
cartesian()	Cartesian product with the other RDD.	rdd.cartesian(other)	{(1, 3), (1, 4), … (3,5)}
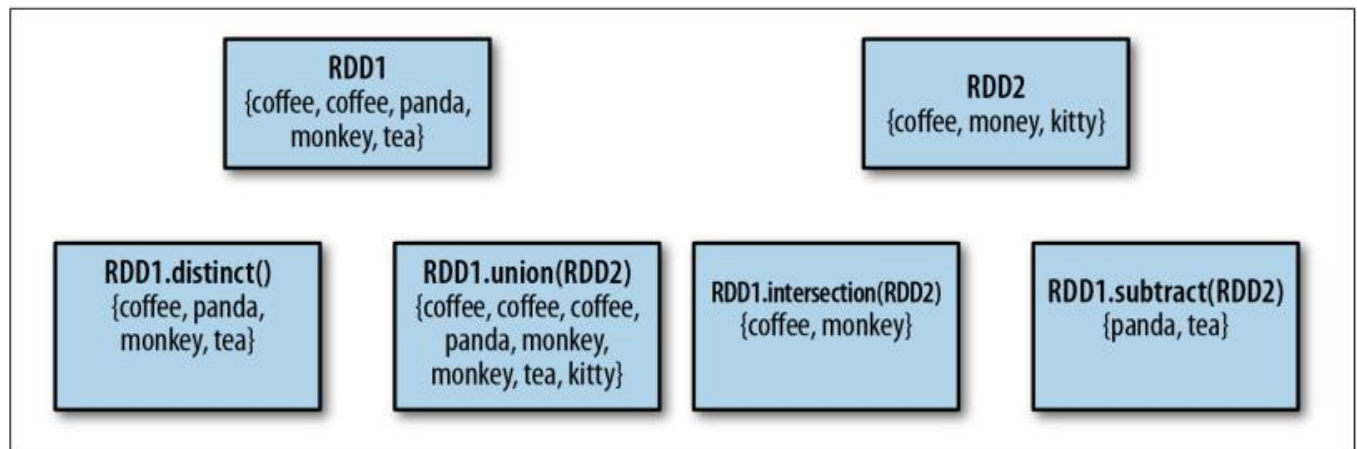
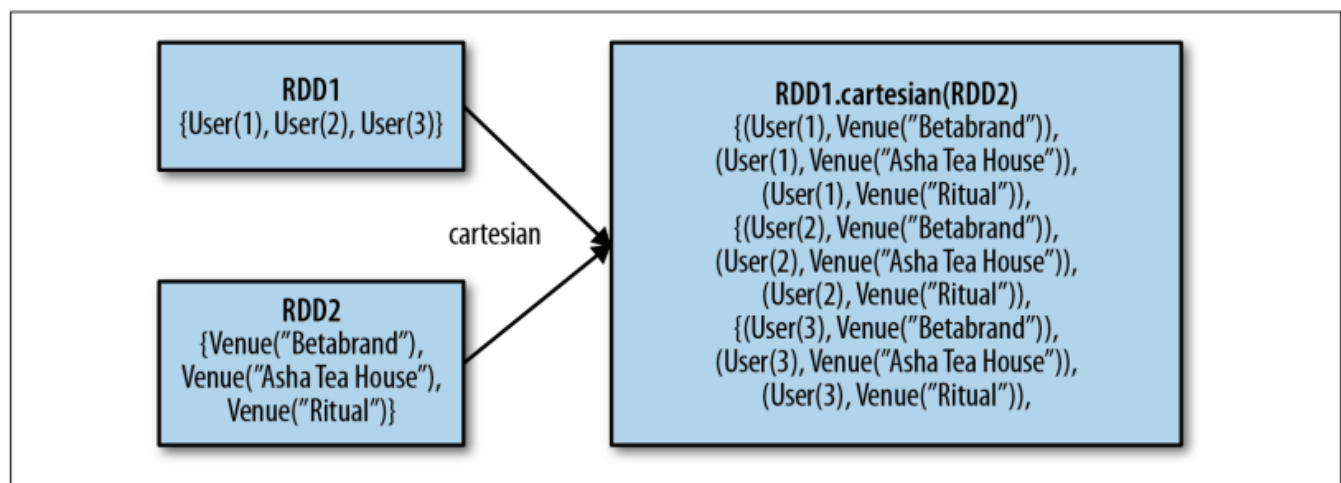*Figure* .   *Some simple set operations*

Figure :   *Cartesian product between two RDDs*

Lab Task

Write spark code for the following. Consider each part as separate. Input the data from the text file in an RDD.

1. Select the records of students from the Lahore campus.
2. Display a few records and print the count of the students from Lahore.
3. Filter the records of the students from the year in the range of 1995- 2023.
4. Display the count of students on each Campus. Partition the input data on the base of Campus. (override Spark Partitioner).
5. For each course, print the number of failures on each Campus.
6. Remove all the duplicate rows from input data.

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