In [1]:
```
from pyspark.sql import SparkSession
import pyspark.sql.functions as F
from pyspark.sql.types import *

spark = SparkSession\
    .builder\
    .appName("chapter-12-RDD-basic")\
    .getOrCreate()
```

In [2]:
```
df1 = spark.range(10).rdd
```

In [3]:
```
type(df1)
```

Out[3]:
```
pyspark.rdd.RDD
```

In [4]:
```
df1.collect()
```

Out[4]:
```
[Row(id=0),
 Row(id=1),
 Row(id=2),
 Row(id=3),
 Row(id=4),
 Row(id=5),
 Row(id=6),
 Row(id=7),
 Row(id=8),
 Row(id=9)]
```

In [5]:
```
# COMMAND ----------

df2 = spark.range(10).toDF("id").rdd.map(lambda row: row[0])
```

In [6]:
```
type(df2)
```

Out[6]:
```
pyspark.rdd.PipelinedRDD
```

In [7]:
```
df2.collect()
```

Out[7]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [8]:
```
# COMMAND ----------

df3 = spark.range(10).rdd.toDF()
```

In [9]:
```
type(df3)
```

Out[9]:
```
pyspark.sql.dataframe.DataFrame
```

In [10]:
```
df3.collect()
```

Out[10]:
```
[Row(id=0),
 Row(id=1),
 Row(id=2),
 Row(id=3),
 Row(id=4),
 Row(id=5),
 Row(id=6),
 Row(id=7),
 Row(id=8),
 Row(id=9)]
```

In [11]:
```
# COMMAND ----------

myCollection = "Spark The Definitive Guide : Big Data Processing Made S
  .split(" ")
words = spark.sparkContext.parallelize(myCollection, 2)
```

In [12]:
```
type(words)
```

Out[12]:
```
pyspark.rdd.RDD
```

In [13]:
```
# COMMAND ----------

words.setName("myWords")
words.name() # myWords
```

Out[13]:
```
'myWords'
```

In [14]:
```
# COMMAND ----------

def startsWithS(individual):
  return individual.startswith("S")


# COMMAND ----------

words.filter(lambda word: startsWithS(word)).collect()
```

Out[14]:
```
['Spark', 'Simple']
```

In [15]:
```
# COMMAND ----------

words2 = words.map(lambda word: (word, word[0], word.startswith("S")))
```

In [16]:
```
words2.collect()
```

Out[16]:
```
[('Spark', 'S', True),
 ('The', 'T', False),
 ('Definitive', 'D', False),
 ('Guide', 'G', False),
 (':', ':', False),
 ('Big', 'B', False),
 ('Data', 'D', False),
 ('Processing', 'P', False),
 ('Made', 'M', False),
 ('Simple', 'S', True)]
```

In [18]:
```
# COMMAND ----------

words2.filter(lambda record: record[2]).collect()
```

Out[18]:
```
[('Spark', 'S', True), ('Simple', 'S', True)]
```

In [19]:
```
# COMMAND ----------

words.flatMap(lambda word: list(word)).take(5)
```

Out[19]:
```
['S', 'p', 'a', 'r', 'k']
```

In [20]:
```
# COMMAND ----------

words.sortBy(lambda word: len(word) * -1).take(2)
```

Out[20]:
```
['Definitive', 'Processing']
```

In [21]:
```
# COMMAND ----------

fiftyFiftySplit = words.randomSplit([0.5, 0.5])
```

In [23]:
```
type(fiftyFiftySplit)
```

Out[23]:
```
list
```

In [24]:
```
fiftyFiftySplit
```

Out[24]:
```
[PythonRDD[37] at RDD at PythonRDD.scala:53,
 PythonRDD[38] at RDD at PythonRDD.scala:53]
```

In [25]:
```
fiftyFiftySplit[0].collect()
```

Out[25]:
```
['The', 'Guide', 'Big', 'Data', 'Processing', 'Made']
```

In [17]:
```
# COMMAND ----------

spark.sparkContext.parallelize(range(1, 21)).reduce(lambda x, y: x + y)
```

Out[17]:
```
210
```

In [27]:
```
# COMMAND ----------

def wordLengthReducer(leftWord, rightWord):
  if len(leftWord) > len(rightWord):
    return leftWord
  else:
    return rightWord

words.reduce(wordLengthReducer)
```

Out[27]:
```
'Processing'
```

In [28]:
```
# COMMAND ----------

words.getStorageLevel()
```

Out[28]:
```
StorageLevel(False, False, False, False, 1)
```

In [29]:
```
# COMMAND ----------

words.mapPartitions(lambda part: [1]).sum() # 2
```

Out[29]:
```
2
```

In [30]:
```
# COMMAND ----------

def indexedFunc(partitionIndex, withinPartIterator):
  return ["partition: {} => {}".format(partitionIndex,
    x) for x in withinPartIterator]
words.mapPartitionsWithIndex(indexedFunc).collect()
```

Out[30]:
```
['partition: 0 => Spark',
 'partition: 0 => The',
 'partition: 0 => Definitive',
 'partition: 0 => Guide',
 'partition: 0 => :',
 'partition: 1 => Big',
 'partition: 1 => Data',
 'partition: 1 => Processing',
 'partition: 1 => Made',
 'partition: 1 => Simple']
```

In [31]:
```
# COMMAND ----------

spark.sparkContext.parallelize(["Hello", "World"], 2).glom().collect()
# [['Hello'], ['World']]


# COMMAND ----------
```

Out[31]:
```
[['Hello'], ['World']]
```

In [ ]:
```

```