Bu bölümde RDD Dönüşüm(Transformations) fonksiyonları hakkında genel bilgiler vereceğiz

Örnekler için

http://grouplens.org/datasets/movielens/ adresinden veri indirebiliriz

Bu metodların en önemli özellikleri, mevcut RDD üzerinden yeni bir RDD dönüşümü yapmasıdır

map

Yeni bir RDD oluşturmak için kullanılır.Örnek verirsek film(movie) verileri alttaki formatta indirilir

movieId,title,genres
1,Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy
2,Jumanji (1995),Adventure|Children|Fantasy
3,Grumpier Old Men (1995),Comedy|Romance
4,Waiting to Exhale (1995),Comedy|Drama|Romance
5,Father of the Bride Part II (1995),Comedy

Spark ile veri üzerinde işlemler yapabilmek için map metodu ile satır satır okuyarak , okuduğumuz verileri Movie objesine dönüştürebiliriz

Örnekte map metodu verileri satır satır okuyarak yeni bir RDD<Movie> objesi hazırlıyor.Burada dikkat etmemiz gereken konu ise her bir input değerinden bir output değeri oluşturulur

```
class Movie(var movieId: String, var title: String, var genres: String) extends
Serializable{
 // Overriding tostring method
 override def toString(): String = {
   return "[Id : " + movieId + ", Title = " + title + " Genre = " + genres + " ]"
 }
}
```

Spark RDD

```
import org.apache.spark.sql.SparkSession

object MovieTransformation {
 def main(args: Array[String]): Unit = {

   val spark =
SparkSession.builder.master("local").appName("SparkByExample").getOrCreate()
   val
rdd=spark.sparkContext.textFile("/Users/serkan/Desktop/Training/ml-latest-small/mov
ies.csv");
   println("Count : " + rdd.count())
   println("First : " + rdd.first())

   val movieRdd = rdd.map(row => {
     val arr = row.split(",")
     (new Movie(arr(0),arr(1),arr(2)))
   })

   println("Movie RDD Count : " + movieRdd.count())
   println("Movie RDD First : " + movieRdd.first())

 }
}
```

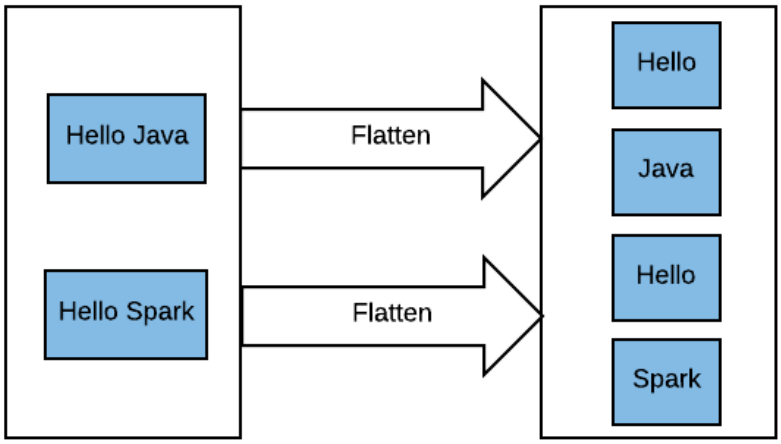
filter

Verileri filtreleyerek yeni bir RDD oluşturur.Alttaki örnekte movieId değeri 2 ile başlayan kayıtlardan yeni bir RDD oluşturuluyor

```
import org.apache.spark.sql.SparkSession

object MovieTransformationFilter {
 def main(args: Array[String]): Unit = {

   val spark =
SparkSession.builder.master("local").appName("SparkByExample").getOrCreate()
   val
rdd=spark.sparkContext.textFile("/Users/serkan/Desktop/Training/ml-latest-small/mov
ies.csv");
   println("Count : " + rdd.count())
   println("First : " + rdd.first())

   val movieRdd = rdd.filter(row => {
     row.startsWith("2")
   })

   println("Filtered Movie RDD Count : " + movieRdd.count())
   println("Filtered RDD First : " + movieRdd.first())

 }
}
```

flatmap

map metodu ile benzerdir fakat her input değerinden birden fazla output değeri oluşturulabilir




```
import org.apache.spark.sql.SparkSession

object MovieTransformationFlatMap {
 def main(args: Array[String]): Unit = {

   val spark =
SparkSession.builder.master("local").appName("SparkByExample").getOrCreate()
   val
rdd=spark.sparkContext.textFile("/Users/serkan/Desktop/Training/ml-latest-small/mov
ies.csv");
   println("Count : " + rdd.count())
   println("First : " + rdd.first())

   val movieRdd = rdd.flatMap(row => {
     row.split(",")
   })

   println("Flatmap RDD Count : " + movieRdd.count())
   println("Flatmap RDD First : " + movieRdd.first())

 }
}
```

Sample
Bir RDD içerisinden ornek kayitlar verir

```
import org.apache.spark.sql.SparkSession

object MovieTransformationSample {
 def main(args: Array[String]): Unit = {

   val spark = SparkSession.builder.master("local").
     appName("SparkByExample").
     getOrCreate()

   val
rdd=spark.sparkContext.textFile("/Users/serkan/Desktop/Training/ml-latest-small/mov
ies.csv");
   println("Count : " + rdd.count())
   println("First : " + rdd.first())

   //1.parametre  > Sampling with Replacement(false ise aynı kayıtlar gelebilir
   //2.parametre  > yüzde olarak kaç tane seçilecek
   val movieRdd = rdd.sample(false,0.9)

   println("Sample Movie RDD Count : " + movieRdd.count())
   println("Sample RDD First : " + movieRdd.first())

 }
}
```

Union
Iki RDD birlestirilir.

```
import org.apache.spark.sql.SparkSession

object MovieTransformationUnion {
 def main(args: Array[String]): Unit = {

   val spark = SparkSession.builder.master("local").
     appName("SparkByExample").
     getOrCreate()

   val
rdd=spark.sparkContext.textFile("/Users/serkan/Desktop/Training/ml-latest-small/mov
ies.csv");
   println("Count : " + rdd.count())
   println("First : " + rdd.first())

   //1.parametre  > Sampling with Replacement(false ise aynı kayıtlar gelebilir
   //2.parametre  > yüzde olarak kaç tane seçilecek
   val sampleRdd = rdd.sample(false,0.1)
   val unionRdd = rdd.union(sampleRdd)

   println("Union Movie RDD Count : " + unionRdd.count())
   println("Union RDD First : " + unionRdd.first())

 }
}
```

Intersection

Intersection metodu ile iki RDD kümesinin kesişimi alınır

```
import org.apache.spark.sql.SparkSession

object MovieTransformationIntersection {
 def main(args: Array[String]): Unit = {

   val spark = SparkSession.builder.master("local").
     appName("SparkByExample").
     getOrCreate()

   val
rdd=spark.sparkContext.textFile("/Users/serkan/Desktop/Training/ml-latest-small/mov
ies.csv");
   println("Count : " + rdd.count())
   println("First : " + rdd.first())

   //1.parametre  > Sampling with Replacement(false ise aynı kayıtlar gelebilir
   //2.parametre  > yüzde olarak kaç tane seçilecek
   val sampleRdd = rdd.sample(false,0.1)
   val unionRdd = rdd.intersection(sampleRdd)

   println("Intersection Movie RDD Count : " + unionRdd.count())
   println("Intersection RDD First : " + unionRdd.first())

 }
}
```

subtract

işlem yapılan RDD kümelerinden yalnızca ilk RDD kümesinde bulunan elemanlar alınır

```
import org.apache.spark.sql.SparkSession

object MovieTransformationSubstract {
 def main(args: Array[String]): Unit = {

   val spark = SparkSession.builder.master("local").
     appName("SparkByExample").
     getOrCreate()

   val
rdd=spark.sparkContext.textFile("/Users/serkan/Desktop/Training/ml-latest-small/mov
ies.csv");
   println("Count : " + rdd.count())
   println("First : " + rdd.first())

   //1.parametre  > Sampling with Replacement(false ise aynı kayıtlar gelebilir
   //2.parametre  > yüzde olarak kaç tane seçilecek
   val sampleRdd = rdd.sample(false,0.1)
   val unionRdd = rdd.subtract(sampleRdd)

   println("Substract Movie RDD Count : " + unionRdd.count())
   println("Substract RDD First : " + unionRdd.first())

 }
}
```

Cartesian
Cartesian metoduile kartezyen eşlemeler oluşturulur

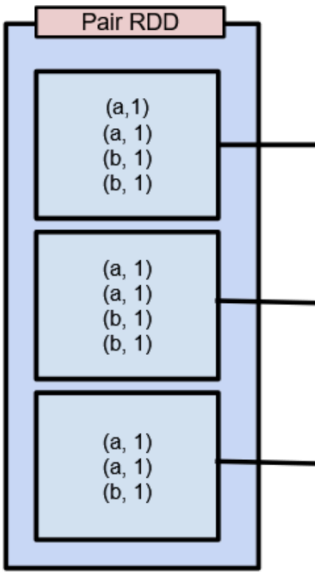
Örnek : RDDs {1, 2, 3} ve {3, 4, 5}

Oluşturulan yeni küme : {(1, 3), (1,4), ... (3,5)}

```
import org.apache.spark.sql.SparkSession

object MovieTransformationCartesien {
 def main(args: Array[String]): Unit = {

   val spark = SparkSession.builder.master("local").
     appName("SparkByExample").
     getOrCreate()

   val
rdd=spark.sparkContext.textFile("/Users/serkan/Desktop/Training/ml-latest-small/mov
ies.csv");
   println("Count : " + rdd.count())
   println("First : " + rdd.first())

   //1.parametre  > Sampling with Replacement(false ise aynı kayıtlar gelebilir
   //2.parametre  > yüzde olarak kaç tane seçilecek
   val sampleRdd = rdd.sample(false,0.1)
   val cartesienRdd = rdd.cartesian(sampleRdd)

   println("cartesian Movie RDD Count : " + cartesienRdd.count())
   println("cartesian RDD First : " + cartesienRdd.first())

 }
}
```

Pair RDD

<key,value> yapılarındaki RDD formatları pair RDD olarak kullanılabilir




```
import org.apache.spark.sql.SparkSession

object MovieTransformationPairRdd {
 def main(args: Array[String]): Unit = {

   val spark =
SparkSession.builder.master("local").appName("SparkByExample").getOrCreate()
   val
rdd=spark.sparkContext.textFile("/Users/serkan/Desktop/Training/ml-latest-small/mov
ies.csv");
   println("---------------->Count : " + rdd.count())
   println("---------------->First : " + rdd.first())

   val pairRdd = rdd.map(row => {
       val arr = row.split(",")
       (arr(0),arr(1))
   })
   println("---------------->PairRDD Movie RDD Count : " + pairRdd.count())
   println("---------------->PairRDD RDD First : " + pairRdd.first())

 }
}
```

Join

aynı key değerine sahip kayıtları birleştirir

```
import org.apache.spark.sql.SparkSession

object MovieTransformationJoin {
 def main(args: Array[String]): Unit = {

   val spark =
SparkSession.builder.master("local").appName("SparkByExample").getOrC
reate()
   val
rdd=spark.sparkContext.textFile("/Users/serkan/Desktop/Training/ml-la
test-small/movies.csv");
   println("----------------->Count : " + rdd.count())
   println("----------------->First : " + rdd.first())

   val pairRdd = rdd.map(row => {
       val arr = row.split(",")
       (arr(0),arr(1))
   })
   println("----------------->PairRDD Movie RDD Count : " +
pairRdd.count())
   println("----------------->PairRDD RDD First : " + pairRdd.first())

   val pairRdd1 = pairRdd.sample(false,0.1)
   val joinRdd = pairRdd.join(pairRdd1);

   println("----------------->joinRdd Movie RDD Count : " +
joinRdd.count())
   println("----------------->joinRdd RDD First : " + joinRdd.first())

 }
}
```