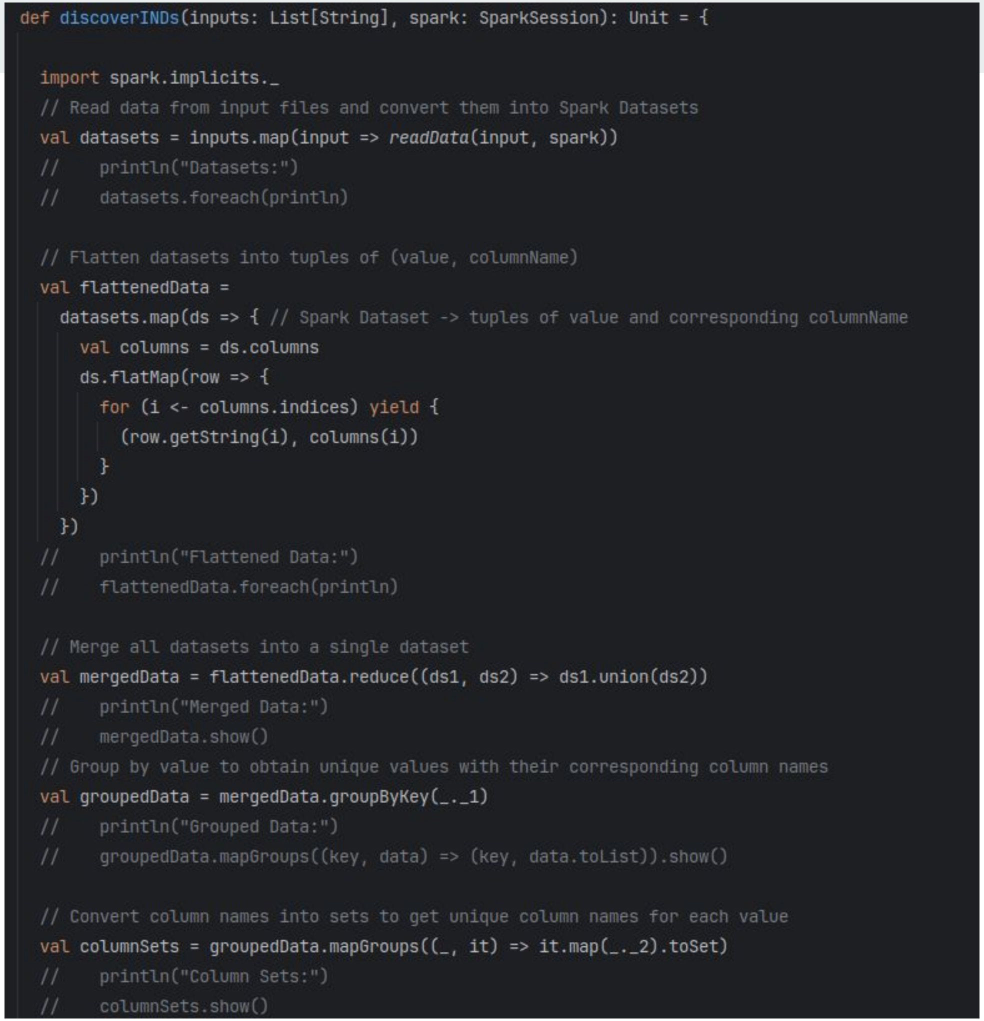
Big Data Systems WS 23/24

IT-Boys:
Sina Fadavi
Mohsen Saleki
Sayedfarhad Emami Dehcheshmeh



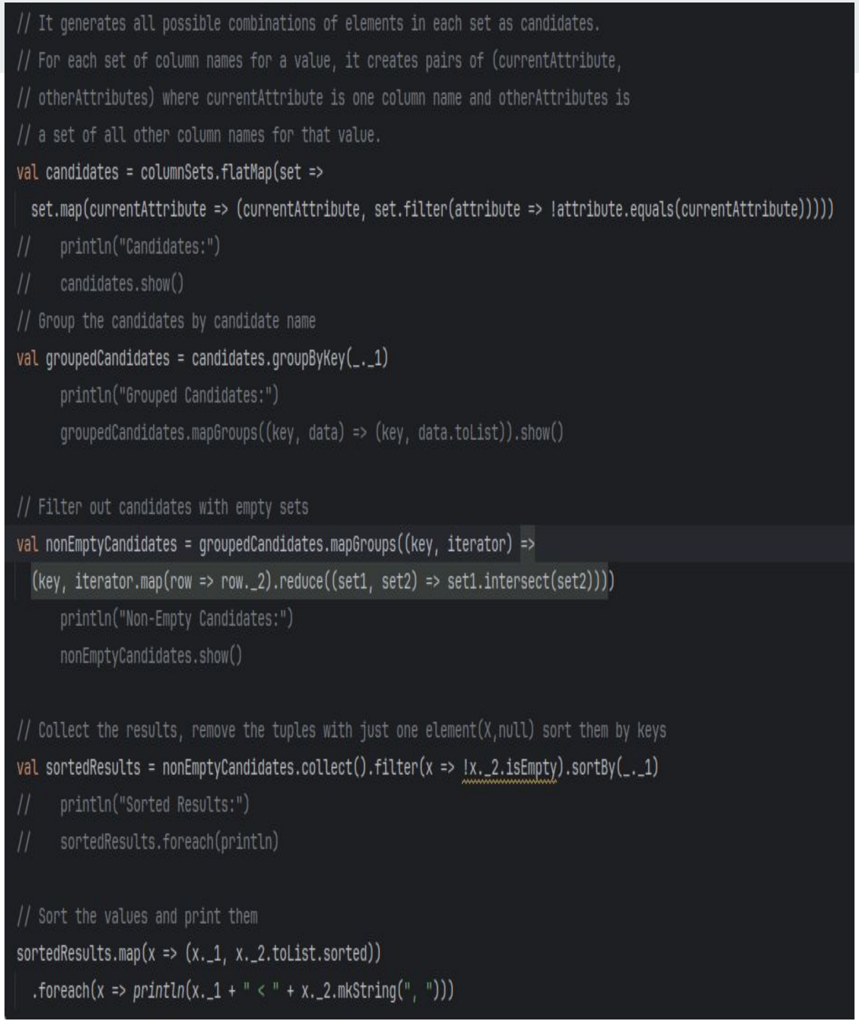
```scala
def discoverINDs(inputs: List[String], spark: SparkSession): Unit = {

  import spark.implicits._
  // Read data from input files and convert them into Spark Datasets
  val datasets = inputs.map(input => readData(input, spark))
  //    println("Datasets:")
  //    datasets.foreach(println)

  // Flatten datasets into tuples of (value, columnName)
  val flattenedData =
    datasets.map(ds => { // Spark Dataset -> tuples of value and corresponding columnName
      val columns = ds.columns
      ds.flatMap(row => {
        for (i <- columns.indices) yield {
          (row.getString(i), columns(i))
        }
      })
    })
  //    println("Flattened Data:")
  //    flattenedData.foreach(println)

  // Merge all datasets into a single dataset
  val mergedData = flattenedData.reduce((ds1, ds2) => ds1.union(ds2))
  //    println("Merged Data:")
  //    mergedData.show()
  // Group by value to obtain unique values with their corresponding column names
  val groupedData = mergedData.groupByKey(_._1)
  //    println("Grouped Data:")
  //    groupedData.mapGroups((key, data) => (key, data.toList)).show()

  // Convert column names into sets to get unique column names for each value
  val columnSets = groupedData.mapGroups((_, it) => it.map(_._2).toSet)
  //    println("Column Sets:")
  //    columnSets.show()
```



```scala
// It generates all possible combinations of elements in each set as candidates.
// For each set of column names for a value, it creates pairs of (currentAttribute,
// otherAttributes) where currentAttribute is one column name and otherAttributes is
// a set of all other column names for that value.
val candidates = columnSets.flatMap(set =>
  set.map(currentAttribute => (currentAttribute, set.filter(attribute => !attribute.equals(currentAttribute)))))
//    println("Candidates:")
//    candidates.show()
// Group the candidates by candidate name
val groupedCandidates = candidates.groupByKey(_._1)
      println("Grouped Candidates:")
      groupedCandidates.mapGroups((key, data) => (key, data.toList)).show()

// Filter out candidates with empty sets
val nonEmptyCandidates = groupedCandidates.mapGroups((key, iterator) =>
  (key, iterator.map(row => row._2).reduce((set1, set2) => set1.intersect(set2))))
      println("Non-Empty Candidates:")
      nonEmptyCandidates.show()

// Collect the results, remove the tuples with just one element(X,null) sort them by keys
val sortedResults = nonEmptyCandidates.collect().filter(x => !x._2.isEmpty).sortBy(_._1)
//    println("Sorted Results:")
//    sortedResults.foreach(println)

// Sort the values and print them
sortedResults.map(x => (x._1, x._2.toList.sorted))
  .foreach(x => println(x._1 + " < " + x._2.mkString(", ")))
```

So We thought the Comments on the code for each part explain enough and we didn’t know what to add more