```
    inputs.map(f = file => spark.read
      .option("...", "...")
      .csv(file)
      // emits (attribute, attribute name) as tuple
      .flatMap(row => row.schema.fields.zipWithIndex.map(tup => (row.getString(tup._2), tup._1.name ))))
      // union all CSVs
      .reduce { (a, b) => a.union(b) }
      .dropDuplicates
      // group by attribute
      .groupByKey(_._1)
      // emit attribute names as set
      .mapGroups { case (_, rows) => rows.map(_._2).toSet }
      .dropDuplicates
      // builds inclusion lists
      .flatMap(set => set.map(e => (e, set - e)))
      .groupByKey(_._1)
      // build intersection of all related inclusion lists
      .mapGroups { case (k, it) => (k, it.foldLeft(it.next()._2) { (a, b) => a.intersect(b._2) }) }
      // filter all attributes that have no foreign key candidate
      .filter(_._2.nonEmpty)
      // sort by dependent
      .sort("_1")
      .collect()
      // generate output
      .foreach { case (dep, ref) => println(s"$dep < ${ref.toList.sorted.mkString(", ")}") }
```