**POM.xml**

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.training.section5</groupId>

<artifactId>section5</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.apache.beam</groupId>

<artifactId>beam-sdks-java-core</artifactId>

<version>2.20.0</version>

</dependency>

<dependency>

<groupId>org.apache.beam</groupId>

<artifactId>beam-runners-direct-java</artifactId>

<version>2.20.0</version>

</dependency>

</dependencies>

</project>

InnerJoinExample.java

**package** section5;

**import** org.apache.beam.sdk.Pipeline;

**import** org.apache.beam.sdk.io.TextIO;

**import** org.apache.beam.sdk.transforms.DoFn;

**import** org.apache.beam.sdk.transforms.ParDo;

**import** org.apache.beam.sdk.transforms.join.CoGbkResult;

**import** org.apache.beam.sdk.transforms.join.CoGroupByKey;

**import** org.apache.beam.sdk.transforms.join.KeyedPCollectionTuple;

**import** org.apache.beam.sdk.values.KV;

**import** org.apache.beam.sdk.values.PCollection;

**import** org.apache.beam.sdk.values.TupleTag;

**class** OrderParsing **extends** DoFn<String,KV<String, String>>{

@ProcessElement

**public** **void** processElement(ProcessContext c) {

String arr[] = c.element().split(",");

String strKey = arr[0];

String strVal = arr[1]+","+arr[2]+","+arr[3];

c.output(KV.*of*(strKey, strVal));

}

}

**class** UserParsing **extends** DoFn<String,KV<String, String>>{

@ProcessElement

**public** **void** processElement(ProcessContext c) {

String arr[] = c.element().split(",");

String strKey = arr[0];

String strVal = arr[1];

c.output(KV.*of*(strKey, strVal));

}

}

**public** **class** InnerJoinExample {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Pipeline p = Pipeline.*create*();

// Step 1 : Convert String to KV object.

PCollection<KV<String,String>> pOrderCollection = p.apply(TextIO.*read*().from("C:\\Beam\\user\_order.csv"))

.apply(ParDo.*of*(**new** OrderParsing()));

PCollection<KV<String,String>> pUserCollection = p.apply(TextIO.*read*().from("C:\\Beam\\p\_user.csv"))

.apply(ParDo.*of*(**new** UserParsing()));

// Step 2 create TupleTag object

**final** TupleTag<String> orderTuple = **new** TupleTag<String>();

**final** TupleTag<String> userTuple = **new** TupleTag<String>();

//Step 3 Combine data sets using CoGroupByKey

PCollection<KV<String, CoGbkResult>> result = KeyedPCollectionTuple.*of*(orderTuple, pOrderCollection)

.and(userTuple, pUserCollection)

.apply(CoGroupByKey.<String>*create*());

// Step 4 : iterate CoGbkResult and build String

PCollection<String> output = result.apply(ParDo.*of*(**new** DoFn<KV<String, CoGbkResult>, String>() {

@ProcessElement

**public** **void** processElement(ProcessContext c) {

String strKey = c.element().getKey();

CoGbkResult valObject = c.element().getValue();

Iterable<String> orderTable= valObject.getAll(orderTuple);

Iterable<String> userTable = valObject.getAll(userTuple);

**for** (String order : orderTable) {

**for** (String user : userTable) {

c.output(strKey+","+order+","+user);

}

}

}

}));

// Step 5 : save the result

output.apply(TextIO.*write*().to("C:\\Beam\\cogroup\_by\_key.csv").withNumShards(1).withSuffix(".csv"));

p.run();

}

}

LeftJoinExample.java

**package** section5;

**import** org.apache.beam.sdk.Pipeline;

**import** org.apache.beam.sdk.io.TextIO;

**import** org.apache.beam.sdk.transforms.DoFn;

**import** org.apache.beam.sdk.transforms.ParDo;

**import** org.apache.beam.sdk.transforms.join.CoGbkResult;

**import** org.apache.beam.sdk.transforms.join.CoGroupByKey;

**import** org.apache.beam.sdk.transforms.join.KeyedPCollectionTuple;

**import** org.apache.beam.sdk.values.KV;

**import** org.apache.beam.sdk.values.PCollection;

**import** org.apache.beam.sdk.values.TupleTag;

**class** LeftOrderParsing **extends** DoFn<String,KV<String, String>>{

@ProcessElement

**public** **void** processElement(ProcessContext c) {

String arr[] = c.element().split(",");

String strKey = arr[0];

String strVal = arr[1]+","+arr[2]+","+arr[3];

c.output(KV.*of*(strKey, strVal));

}

}

**class** LeftUserParsing **extends** DoFn<String,KV<String, String>>{

@ProcessElement

**public** **void** processElement(ProcessContext c) {

String arr[] = c.element().split(",");

String strKey = arr[0];

String strVal = arr[1];

c.output(KV.*of*(strKey, strVal));

}

}

**public** **class** LeftJoinExample {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Pipeline p = Pipeline.*create*();

// Step 1 : Convert String to KV object.

PCollection<KV<String,String>> pOrderCollection = p.apply(TextIO.*read*().from("C:\\Beam\\user\_order.csv"))

.apply(ParDo.*of*(**new** LeftOrderParsing()));

PCollection<KV<String,String>> pUserCollection = p.apply(TextIO.*read*().from("C:\\Beam\\p\_user.csv"))

.apply(ParDo.*of*(**new** LeftUserParsing()));

// Step 2 create TupleTag object

**final** TupleTag<String> orderTuple = **new** TupleTag<String>();

**final** TupleTag<String> userTuple = **new** TupleTag<String>();

//Step 3 Combine data sets using CoGroupByKey

PCollection<KV<String, CoGbkResult>> result = KeyedPCollectionTuple.*of*(orderTuple, pOrderCollection)

.and(userTuple, pUserCollection)

.apply(CoGroupByKey.<String>*create*());

// Step 4 : iterate CoGbkResult and build String

PCollection<String> output = result.apply(ParDo.*of*(**new** DoFn<KV<String, CoGbkResult>, String>() {

@ProcessElement

**public** **void** processElement(ProcessContext c) {

String strKey = c.element().getKey();

CoGbkResult valObject = c.element().getValue();

Iterable<String> orderTable= valObject.getAll(orderTuple);

Iterable<String> userTable = valObject.getAll(userTuple);

**for** (String order : orderTable) {

**if**(userTable.iterator().hasNext()) {

**for** (String user : userTable) {

c.output(strKey+","+order+","+user);

}

}

**else** {

c.output(strKey+","+order+","+**null**);

}

}

}

}));

// Step 5 : save the result

output.apply(TextIO.*write*().to("C:\\Beam\\left\_join\_example.csv").withNumShards(1).withSuffix(".csv"));

p.run();

}

}

RightJoinExample.java

**package** section5;

**import** org.apache.beam.sdk.Pipeline;

**import** org.apache.beam.sdk.io.TextIO;

**import** org.apache.beam.sdk.transforms.DoFn;

**import** org.apache.beam.sdk.transforms.ParDo;

**import** org.apache.beam.sdk.transforms.join.CoGbkResult;

**import** org.apache.beam.sdk.transforms.join.CoGroupByKey;

**import** org.apache.beam.sdk.transforms.join.KeyedPCollectionTuple;

**import** org.apache.beam.sdk.values.KV;

**import** org.apache.beam.sdk.values.PCollection;

**import** org.apache.beam.sdk.values.TupleTag;

**class** RightOrderParsing **extends** DoFn<String,KV<String, String>>{

@ProcessElement

**public** **void** processElement(ProcessContext c) {

String arr[] = c.element().split(",");

String strKey = arr[0];

String strVal = arr[1]+","+arr[2]+","+arr[3];

c.output(KV.*of*(strKey, strVal));

}

}

**class** RightUserParsing **extends** DoFn<String,KV<String, String>>{

@ProcessElement

**public** **void** processElement(ProcessContext c) {

String arr[] = c.element().split(",");

String strKey = arr[0];

String strVal = arr[1];

c.output(KV.*of*(strKey, strVal));

}

}

**public** **class** RightJoinExample {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Pipeline p = Pipeline.*create*();

// Step 1 : Convert String to KV object.

PCollection<KV<String,String>> pOrderCollection = p.apply(TextIO.*read*().from("C:\\Beam\\user\_order.csv"))

.apply(ParDo.*of*(**new** RightOrderParsing()));

PCollection<KV<String,String>> pUserCollection = p.apply(TextIO.*read*().from("C:\\Beam\\p\_user.csv"))

.apply(ParDo.*of*(**new** RightUserParsing()));

// Step 2 create TupleTag object

**final** TupleTag<String> orderTuple = **new** TupleTag<String>();

**final** TupleTag<String> userTuple = **new** TupleTag<String>();

//Step 3 Combine data sets using CoGroupByKey

PCollection<KV<String, CoGbkResult>> result = KeyedPCollectionTuple.*of*(orderTuple, pOrderCollection)

.and(userTuple, pUserCollection)

.apply(CoGroupByKey.<String>*create*());

// Step 4 : iterate CoGbkResult and build String

PCollection<String> output = result.apply(ParDo.*of*(**new** DoFn<KV<String, CoGbkResult>, String>() {

@ProcessElement

**public** **void** processElement(ProcessContext c) {

String strKey = c.element().getKey();

CoGbkResult valObject = c.element().getValue();

Iterable<String> orderTable= valObject.getAll(orderTuple);

Iterable<String> userTable = valObject.getAll(userTuple);

**for** (String user : userTable) {

**if**(orderTable.iterator().hasNext()) {

**for** (String order : orderTable) {

c.output(strKey+","+user+","+order);

}

}

**else** {

c.output(strKey+","+user+",null,null,null");

}

}

}

}));

// Step 5 : save the result

output.apply(TextIO.*write*().to("C:\\Beam\\right\_join\_example.csv").withNumShards(1).withSuffix(".csv"));

p.run();

}

}