**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.section9</groupId>

<artifactId>section9</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

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

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

<version>2.16.0</version>

</dependency>

<dependency>

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

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

<version>2.16.0</version>

</dependency>

<dependency>

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

<artifactId>beam-sdks-java-extensions-sql</artifactId>

<version>2.16.0</version>

</dependency>

</dependencies>

</project>

SQLCountExample.java

package section9;

import java.io.Serializable;

import java.util.Date;

import java.util.Iterator;

import java.util.List;

import java.util.stream.Collectors;

import java.util.stream.Stream;

import org.apache.beam.sdk.Pipeline;

import org.apache.beam.sdk.coders.RowCoder;

import org.apache.beam.sdk.transforms.Create;

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

import org.apache.beam.sdk.transforms.MapElements;

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

import org.apache.beam.sdk.transforms.SerializableFunction;

import org.apache.beam.sdk.transforms.SerializableFunctions;

import org.apache.beam.sdk.transforms.SimpleFunction;

import org.apache.beam.sdk.values.PBegin;

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

import org.apache.beam.sdk.values.Row;

import org.apache.beam.sdk.extensions.sql.SqlTransform;

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

import org.apache.beam.sdk.schemas.Schema;

public class SQLCountExample {

final static String HEADER = "userId,orderId,productId,Amount";

final static Schema schema = Schema.builder().addStringField("userId")

.addStringField("orderId")

.addStringField("productId")

.addDoubleField("Amount").build();

public static void main(String[] args) {

// TODO Auto-generated method stub

Pipeline pipeline = Pipeline.create();

// Step 1 : Read csv file.

PCollection<String> fileInput= pipeline.apply(TextIO.read().from("C:\\Beam\\user\_order.csv"));

// Step 2 : Convert PCollection<String> to PCollection<Row>

PCollection<Row> rowInput = fileInput.apply(ParDo.of(new StringToRow())).setRowSchema(schema);

// Step 3 : Apply SqlTramsform.query

PCollection<Row> sqlInput = rowInput.apply(SqlTransform.query("select userId,Count(userId) from PCOLLECTION group by userId "));

// Step 4 : Convert PCollection<Row> to PCollection<String>

PCollection<String> pOutput = sqlInput.apply(ParDo.of(new RowToString()));

pOutput.apply(TextIO.write().to("C:\\Beam\\sql\_count\_output.csv").withNumShards(1).withSuffix(".csv"));

pipeline.run();

}

//ParDo for String -> Row (SQL)

public static class StringToRow extends DoFn<String, Row> {

@ProcessElement

public void processElement(ProcessContext c) {

if(!c.element().equalsIgnoreCase(HEADER)) {

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

Row record=Row.withSchema(schema).addValues(arr[0],arr[1],arr[2],Double.valueOf(arr[3])).build();

c.output(record);

}

}

}

//ParDo for Row (SQL) -> String

public static class RowToString extends DoFn<Row, String> {

@ProcessElement

public void processElement(ProcessContext c) {

String outString=c.element().getValues().stream().

map(Object::toString).collect(Collectors.joining(","));

c.output(outString);

}

}

}

SQLExample.java

package section9;

import java.io.Serializable;

import java.util.Date;

import java.util.Iterator;

import java.util.List;

import java.util.stream.Collectors;

import java.util.stream.Stream;

import org.apache.beam.sdk.Pipeline;

import org.apache.beam.sdk.coders.RowCoder;

import org.apache.beam.sdk.transforms.Create;

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

import org.apache.beam.sdk.transforms.MapElements;

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

import org.apache.beam.sdk.transforms.SerializableFunction;

import org.apache.beam.sdk.transforms.SerializableFunctions;

import org.apache.beam.sdk.transforms.SimpleFunction;

import org.apache.beam.sdk.values.PBegin;

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

import org.apache.beam.sdk.values.Row;

import org.apache.beam.sdk.extensions.sql.SqlTransform;

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

import org.apache.beam.sdk.schemas.Schema;

public class SQLExample {

final static String HEADER = "userId,orderId,productId,Amount";

final static Schema schema = Schema.builder().addStringField("userId")

.addStringField("orderId")

.addStringField("productId")

.addDoubleField("Amount").build();

public static void main(String[] args) {

// TODO Auto-generated method stub

Pipeline pipeline = Pipeline.create();

// Step 1 : Read csv file.

PCollection<String> fileInput= pipeline.apply(TextIO.read().from("C:\\Beam\\user\_order.csv"));

// Step 2 : Convert PCollection<String> to PCollection<Row>

PCollection<Row> rowInput = fileInput.apply(ParDo.of(new StringToRow())).setRowSchema(schema);

// Step 3 : Apply SqlTramsform.query

PCollection<Row> sqlInput = rowInput.apply(SqlTransform.query("select \* from PCOLLECTION "));

// Step 4 : Convert PCollection<Row> to PCollection<String>

PCollection<String> pOutput = sqlInput.apply(ParDo.of(new RowToString()));

pOutput.apply(TextIO.write().to("C:\\Beam\\sql\_output.csv").withNumShards(1).withSuffix(".csv"));

pipeline.run();

}

//ParDo for String -> Row (SQL)

public static class StringToRow extends DoFn<String, Row> {

@ProcessElement

public void processElement(ProcessContext c) {

if(!c.element().equalsIgnoreCase(HEADER)) {

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

Row record=Row.withSchema(schema).addValues(arr[0],arr[1],arr[2],Double.valueOf(arr[3])).build();

c.output(record);

}

}

}

//ParDo for Row (SQL) -> String

public static class RowToString extends DoFn<Row, String> {

@ProcessElement

public void processElement(ProcessContext c) {

String outString=c.element().getValues().stream().

map(Object::toString).collect(Collectors.joining(","));

c.output(outString);

}

}

}

SQLJoinExample.java

package section9;

import java.io.Serializable;

import java.util.Date;

import java.util.Iterator;

import java.util.List;

import java.util.stream.Collectors;

import java.util.stream.Stream;

import org.apache.beam.sdk.Pipeline;

import org.apache.beam.sdk.coders.RowCoder;

import org.apache.beam.sdk.transforms.Create;

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

import org.apache.beam.sdk.transforms.MapElements;

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

import org.apache.beam.sdk.transforms.SerializableFunction;

import org.apache.beam.sdk.transforms.SerializableFunctions;

import org.apache.beam.sdk.transforms.SimpleFunction;

import org.apache.beam.sdk.values.PBegin;

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

import org.apache.beam.sdk.values.PCollectionTuple;

import org.apache.beam.sdk.values.Row;

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

import org.apache.beam.sdk.extensions.sql.SqlTransform;

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

import org.apache.beam.sdk.schemas.Schema;

public class SQLJoinExample {

final static String order\_header="userId,orderId,productId,Amount";

final static Schema order\_schema=Schema.builder()

.addStringField("userId")

.addStringField("orderId")

.addStringField("productId")

.addDoubleField("Amount").build();

final static String user\_header="userId,name";

final static Schema user\_schema=Schema.builder()

.addStringField("userId")

.addStringField("name").build();

final static Schema order\_user\_schema=Schema.builder()

.addStringField("userId")

.addStringField("orderId")

.addStringField("productId")

.addDoubleField("Amount").addStringField("name").build();

public static void main(String[] args) {

// TODO Auto-generated method stub

Pipeline pipeline = Pipeline.create();

// Step 1 : Read csv file.

PCollection<String> order= pipeline.apply(TextIO.read().from("C:\\Beam\\user\_order.csv"));

PCollection<String> user= pipeline.apply(TextIO.read().from("C:\\Beam\\p\_user.csv"));

// Step 2 : Convert PCollection<String> to PCollection<Row>

PCollection<Row> rowUserOrder = order.apply(ParDo.of(new StringToOrderRow()))

.setRowSchema(order\_schema);

PCollection<Row> rowUser = user.apply(ParDo.of(new StringToUserRow()))

.setRowSchema(user\_schema);

// Step 3 : Apply SqlTramsform.query

PCollection<Row> sqlInput = PCollectionTuple.of(new TupleTag<>("orders"), rowUserOrder)

.and(new TupleTag<>("users"), rowUser)

.apply(SqlTransform.query("select o.\*,u.name from orders o inner join users u on o.userId=u.userId"));

// Step 4 : Convert PCollection<Row> to PCollection<String>

PCollection<String> pOutput = sqlInput.apply(ParDo.of(new RowToString()));

pOutput.apply(TextIO.write().to("C:\\Beam\\sql\_join\_output.csv").withNumShards(1).withSuffix(".csv"));

pipeline.run();

}

//ParDo for String -> Row (SQL)

public static class StringToOrderRow extends DoFn<String, Row> {

@ProcessElement

public void processElement(ProcessContext c) {

if(!c.element().equalsIgnoreCase(order\_header)) {

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

Row record=Row.withSchema(order\_schema)

.addValues(arr[0],arr[1],arr[2],Double.valueOf(arr[3])).build();

c.output(record);

}

}

}

//ParDo for String -> Row (SQL)

public static class StringToUserRow extends DoFn<String, Row> {

@ProcessElement

public void processElement(ProcessContext c) {

if(!c.element().equalsIgnoreCase(user\_header)) {

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

Row record=Row.withSchema(user\_schema)

.addValues(arr[0],arr[1]).build();

c.output(record);

}

}

}

//ParDo for Row (SQL) -> String

public static class RowToString extends DoFn<Row, String> {

@ProcessElement

public void processElement(ProcessContext c) {

String outString=c.element().getValues().stream()

// For Left Join

// .filter(entity -> entity != null)

.map(Object::toString).collect(Collectors.joining(","));

c.output(outString);

}

}

}