**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.section7</groupId>

<artifactId>section7</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

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

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

<version>2.19.0</version>

</dependency>

<dependency>

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

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

<version>2.19.0</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.46</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.beam/beam-sdks-java-io-jdbc -->

<dependency>

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

<artifactId>beam-sdks-java-io-jdbc</artifactId>

<version>2.4.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.beam/beam-sdks-java-io-mongodb -->

<dependency>

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

<artifactId>beam-sdks-java-io-mongodb</artifactId>

<version>2.1.0</version>

</dependency>

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-common</artifactId>

<version>3.1.2</version>

</dependency>

<dependency>

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

<artifactId>beam-sdks-java-io-hadoop-file-system</artifactId>

<version>2.8.0</version>

</dependency>

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-hdfs-client</artifactId>

<version>3.1.2</version>

</dependency>

</dependencies>

<build>

<finalName>MavenShadeExample-uber</finalName>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-shade-plugin</artifactId>

<version>2.3</version>

<executions>

<execution>

<phase>package</phase>

<goals>

<goal>shade</goal>

</goals>

<configuration>

<transformers>

<transformer

implementation=*"org.apache.maven.plugins.shade.resource.ServicesResourceTransformer"* />

</transformers>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

HDFSExample.java

package section7;

import java.util.Collections;

import java.util.HashMap;

import java.util.Map;

import org.apache.beam.sdk.Pipeline;

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

import org.apache.beam.sdk.io.hdfs.HadoopFileSystemOptions;

import org.apache.beam.sdk.options.PipelineOptionsFactory;

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

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

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

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

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

import org.apache.hadoop.conf.Configuration;

import org.bson.Document;

public class HDFSExample {

public static void main(String[] args) {

//

Configuration conf = new Configuration();

conf.set("fs.defaultFS", "hdfs://172.31.26.230:8020");

conf.set("fs.hdfs.impl", org.apache.hadoop.hdfs.DistributedFileSystem.class.getName());

conf.set("fs.file.impl", org.apache.hadoop.fs.LocalFileSystem.class.getName());

String[] args1 = new String[] { "--hdfsConfiguration=[{\"fs.default.name\" : \"hdfs://172.31.26.230:8020\"}]",

"--runner=DirectRunner" };

HadoopFileSystemOptions hfdsOptions = PipelineOptionsFactory.fromArgs(args1).withValidation().as(HadoopFileSystemOptions.class);

hfdsOptions.setHdfsConfiguration(Collections.singletonList(conf));

Pipeline p = Pipeline.create(hfdsOptions);

PCollection<String> pHdfs = p.apply(TextIO.read().from("hdfs://172.31.26.230:8020/user/user.csv"));

pHdfs.apply(ParDo.of(new DoFn<String, Void>() {

@ProcessElement

public void processElement(ProcessContext c) {

System.out.println(c.element());

}

}));

p.run();

}

}

JDBCIOExample.java

package section7;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import org.apache.beam.sdk.Pipeline;

import org.apache.beam.sdk.coders.StringUtf8Coder;

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

import org.apache.beam.sdk.io.jdbc.JdbcIO;

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

public class JDBCIOExample {

public static void main(String[] args) {

Pipeline p = Pipeline.create();

PCollection<String> poutput = p.apply(JdbcIO.<String>read().

withDataSourceConfiguration(JdbcIO.DataSourceConfiguration

.create("com.mysql.jdbc.Driver","jdbc:mysql://127.0.0.1:3306/products?useSSL=false")

.withUsername("root")

.withPassword("root"))

.withQuery("SELECT name, city, currency from product\_info WHERE name = ? ")

.withCoder(StringUtf8Coder.of())

.withStatementPreparator(new JdbcIO.StatementPreparator() {

public void setParameters(PreparedStatement preparedStatement) throws Exception {

// TODO Auto-generated method stub

preparedStatement.setString(1, "iphone");

}

})

.withRowMapper(new JdbcIO.RowMapper<String>() {

public String mapRow(ResultSet resultSet) throws Exception {

return resultSet.getString(1)+","+resultSet.getString(2)+","+resultSet.getString(3);

}

})

);

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

p.run();

}

}

MongoDBExample.java

package section7;

import java.util.HashMap;

import java.util.Map;

import org.apache.beam.sdk.Pipeline;

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

import org.apache.beam.sdk.io.mongodb.MongoDbIO;

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

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

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

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

import org.bson.Document;

public class MongoDBExample {

public static void main(String[] args) {

Pipeline p = Pipeline.create();

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

PCollection<Document> pDocument = pInput.apply(ParDo.of(new DoFn<String, Document>() {

@ProcessElement

public void processElement(ProcessContext c) {

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

Map<String, Object> mapDocuments = new HashMap<String, Object>();

mapDocuments.put("userId", arr[0]);

mapDocuments.put("OrderId", arr[1]);

mapDocuments.put("Name", arr[2]);

mapDocuments.put("ProductId", arr[3]);

mapDocuments.put("Amount", arr[4]);

mapDocuments.put("Order\_Date", arr[5]);

mapDocuments.put("Country", arr[6]);

Document d1 = new Document(mapDocuments);

c.output(d1);

}

}));

pDocument.apply(MongoDbIO.write().withUri("mongodb://localhost:27017")

.withDatabase("training").withCollection("user"));

p.run();

}

}