**SESSION 9 – ADVANCED HIVE ASSIGNMENT.**

**Associated Data Files**

This Data set is about Olympics. You can download the data set from the below link:

<https://drive.google.com/open?id=0ByJLBTmJojjzV1czX3Nha0R3bTQ>

**DATE SET DESCRIPTION**

The data set consists of the following fields.

Athlete: This field consists of the athlete name

Age: This field consists of athlete ages

Country: This fields consists of the country names which participated in Olympics

Year: This field consists of the year

Closing Date: This field consists of the closing date of ceremony

Sport: Consists of the sports name

Gold Medals: No. of Gold medals

Silver Medals: No. of Silver medals

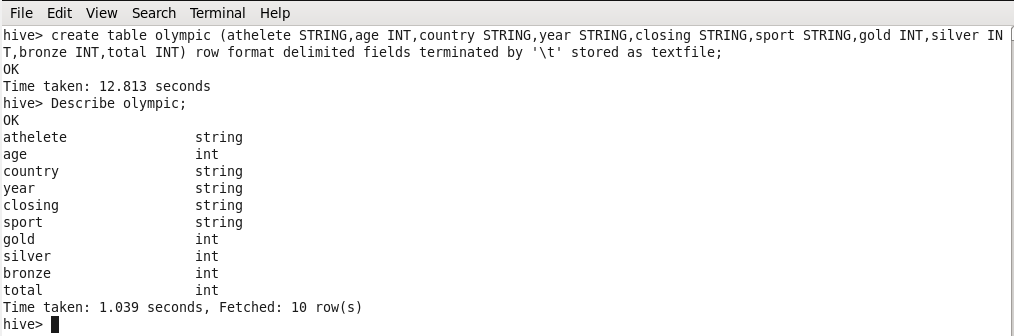
Bronze Medals: No. of Bronze medals

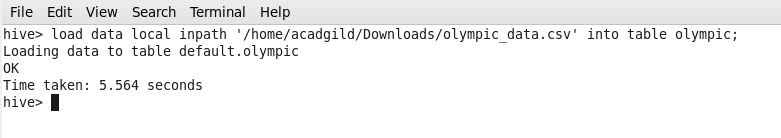
Total Medals: Consists of total no. of medals

**Solution** :

First Create a table Olympic in Hive and Loaa data to it from the locally saved Olympic\_data.csv provided as shown below :

create table olympic (athelete STRING,age INT,country STRING,year STRING,closing STRING,sport STRING,gold INT,silver INT,bronze INT,total INT) row format delimited fields terminated by ‘\t’ stored as textfile;

  
  
load data local inpath ‘/home/acadgild/Downloads/olympic\_data.csv’ into table olympic;



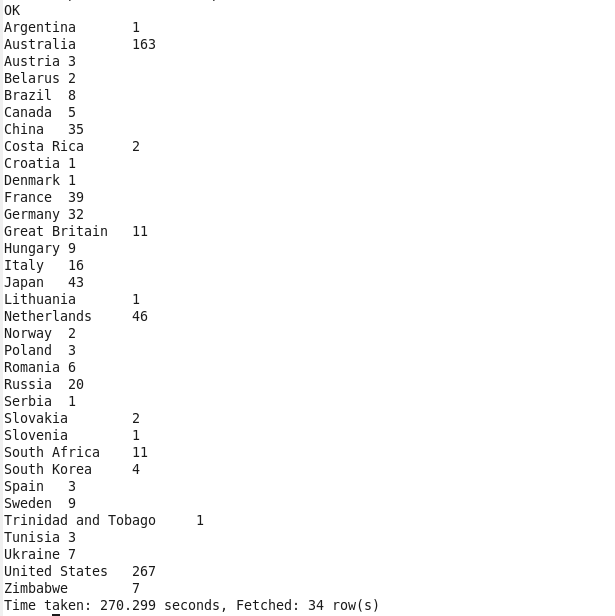
**Problem Statement**

**Task 1**

1. **Write a Hive program to find the number of medals won by each country in swimming.**

**Solution :**

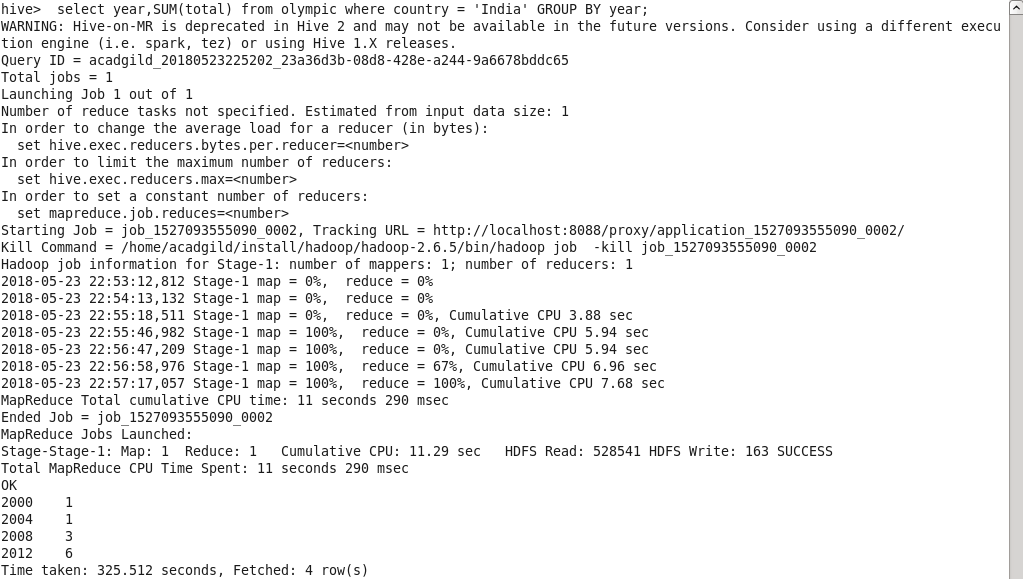
select country,SUM(total) from olympic where sport = ‘Swimming’ GROUP BY country;



1. **Write a Hive program to find the number of medals that India won year wise**.

**Solution :**

**S**elect year,SUM(total) from olympic where country = ‘India’ GROUP BY year;



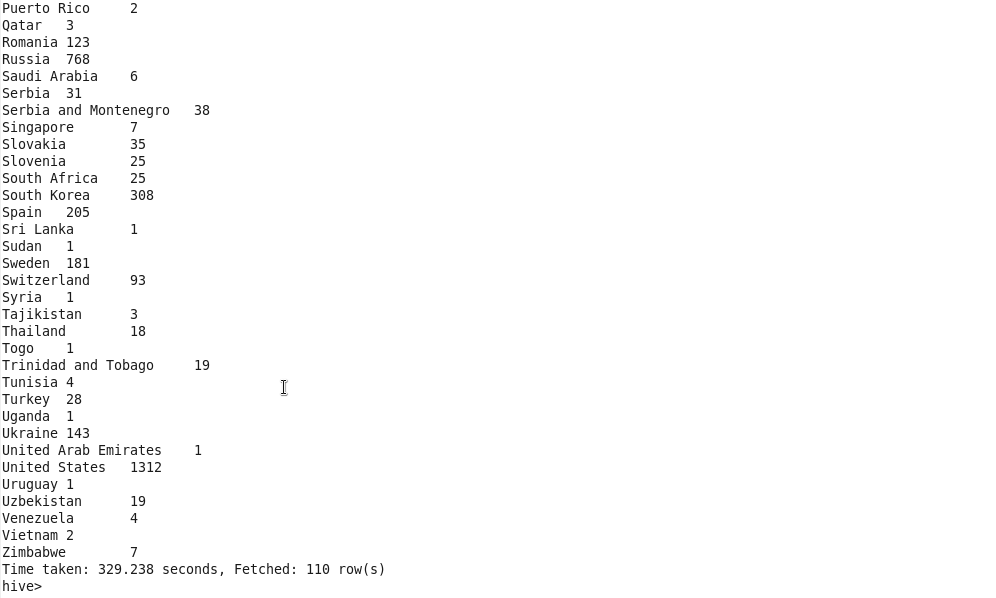
1. **Write a Hive Program to find the total number of medals each country won.**

**Solution :**

select country,SUM(total) from olympic GROUP BY country**;**

****

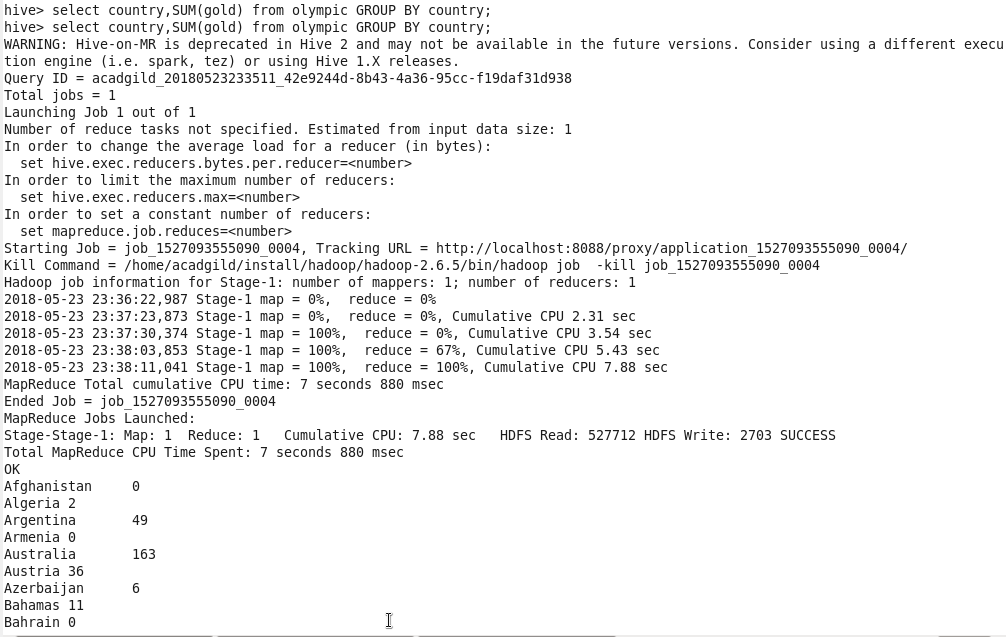
****

****

1. **Write a Hive program to find the number of gold medals each country won.**

**Solution:**

select country,SUM(gold) from olympic GROUP BY country;





**Task 2**

**Write a hive UDF that implements functionality of string concat\_ws(string SEP, array<string>).**

**This UDF will accept two arguments, one string and one array of string.**

**It will return a single string where all the elements of the array are separated by the SEP.**

**Solution :**

|  |
| --- |
| /\* |
|  |

|  |
| --- |
| \* Licensed to the Apache Software Foundation (ASF) under one |
|  |

|  |
| --- |
| \* or more contributor license agreements. See the NOTICE file |
|  |

|  |
| --- |
| \* distributed with this work for additional information |
|  |

|  |
| --- |
| \* regarding copyright ownership. The ASF licenses this file |
|  |

|  |
| --- |
| \* to you under the Apache License, Version 2.0 (the |
|  |

|  |
| --- |
| \* "License"); you may not use this file except in compliance |
|  |

|  |
| --- |
| \* with the License. You may obtain a copy of the License at |
|  |

|  |
| --- |
| \* |
|  |

|  |
| --- |
| \* http://www.apache.org/licenses/LICENSE-2.0 |
|  |

|  |
| --- |
| \* |
|  |

|  |
| --- |
| \* Unless required by applicable law or agreed to in writing, software |
|  |

|  |
| --- |
| \* distributed under the License is distributed on an "AS IS" BASIS, |
|  |

|  |
| --- |
| \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. |
|  |

|  |
| --- |
| \* See the License for the specific language governing permissions and |
|  |

|  |
| --- |
| \* limitations under the License. |
|  |

|  |
| --- |
| \*/ |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| package org.apache.hadoop.hive.ql.udf.generic; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import org.apache.hadoop.hive.ql.exec.Description; |
|  |

|  |
| --- |
| import org.apache.hadoop.hive.ql.exec.UDFArgumentException; |
|  |

|  |
| --- |
| import org.apache.hadoop.hive.ql.exec.UDFArgumentLengthException; |
|  |

|  |
| --- |
| import org.apache.hadoop.hive.ql.exec.UDFArgumentTypeException; |
|  |

|  |
| --- |
| import org.apache.hadoop.hive.ql.metadata.HiveException; |
|  |

|  |
| --- |
| import org.apache.hadoop.hive.serde.serdeConstants; |
|  |

|  |
| --- |
| import org.apache.hadoop.hive.serde2.objectinspector.ListObjectInspector; |
|  |

|  |
| --- |
| import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector; |
|  |

|  |
| --- |
| import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector.Category; |
|  |

|  |
| --- |
| import org.apache.hadoop.hive.serde2.objectinspector.PrimitiveObjectInspector; |
|  |

|  |
| --- |
| import org.apache.hadoop.hive.serde2.objectinspector.PrimitiveObjectInspector.PrimitiveCategory; |
|  |

|  |
| --- |
| import org.apache.hadoop.hive.serde2.objectinspector.primitive.PrimitiveObjectInspectorFactory; |
|  |

|  |
| --- |
| import org.apache.hadoop.hive.serde2.objectinspector.primitive.PrimitiveObjectInspectorUtils; |
|  |

|  |
| --- |
| import org.apache.hadoop.hive.serde2.objectinspector.primitive.PrimitiveObjectInspectorUtils.PrimitiveGrouping; |
|  |

|  |
| --- |
| import org.apache.hadoop.hive.serde2.objectinspector.primitive.StringObjectInspector; |
|  |

|  |
| --- |
| import org.apache.hadoop.io.Text; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| /\*\* |
|  |

|  |
| --- |
| \* Generic UDF for string function |
|  |

|  |
| --- |
| \* <code>CONCAT\_WS(sep, [string | array(string)]+)<code>. |
|  |

|  |
| --- |
| \* This mimics the function from |
|  |

|  |
| --- |
| \* MySQL http://dev.mysql.com/doc/refman/5.0/en/string-functions.html# |
|  |

|  |
| --- |
| \* function\_concat-ws |
|  |

|  |
| --- |
| \* |
|  |

|  |
| --- |
| \* @see org.apache.hadoop.hive.ql.udf.generic.GenericUDF |
|  |

|  |
| --- |
| \*/ |
|  |

|  |
| --- |
| @Description(name = "concat\_ws", |
|  |

|  |
| --- |
| value = "\_FUNC\_(separator, [string | array(string)]+) - " |
|  |

|  |
| --- |
| + "returns the concatenation of the strings separated by the separator.", |
|  |

|  |
| --- |
| extended = "Example:\n" |
|  |

|  |
| --- |
| + " > SELECT \_FUNC\_('.', 'www', array('facebook', 'com')) FROM src LIMIT 1;\n" |
|  |

|  |
| --- |
| + " 'www.facebook.com'") |
|  |

|  |
| --- |
| public class GenericUDFConcatWS extends GenericUDF { |
|  |

|  |
| --- |
| private transient ObjectInspector[] argumentOIs; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public ObjectInspector initialize(ObjectInspector[] arguments) throws UDFArgumentException { |
|  |

|  |
| --- |
| if (arguments.length < 2) { |
|  |

|  |
| --- |
| throw new UDFArgumentLengthException( |
|  |

|  |
| --- |
| "The function CONCAT\_WS(separator,[string | array(string)]+) " |
|  |

|  |
| --- |
| + "needs at least two arguments."); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // check if argument is a string or an array of strings |
|  |

|  |
| --- |
| for (int i = 0; i < arguments.length; i++) { |
|  |

|  |
| --- |
| switch(arguments[i].getCategory()) { |
|  |

|  |
| --- |
| case LIST: |
|  |

|  |
| --- |
| if (isStringOrVoidType( |
|  |

|  |
| --- |
| ((ListObjectInspector) arguments[i]).getListElementObjectInspector())) { |
|  |

|  |
| --- |
| break; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| case PRIMITIVE: |
|  |

|  |
| --- |
| if (isStringOrVoidType(arguments[i])) { |
|  |

|  |
| --- |
| break; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| default: |
|  |

|  |
| --- |
| throw new UDFArgumentTypeException(i, "Argument " + (i + 1) |
|  |

|  |
| --- |
| + " of function CONCAT\_WS must be \"" + serdeConstants.STRING\_TYPE\_NAME |
|  |

|  |
| --- |
| + " or " + serdeConstants.LIST\_TYPE\_NAME + "<" + serdeConstants.STRING\_TYPE\_NAME |
|  |

|  |
| --- |
| + ">\", but \"" + arguments[i].getTypeName() + "\" was found."); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| argumentOIs = arguments; |
|  |

|  |
| --- |
| return PrimitiveObjectInspectorFactory.writableStringObjectInspector; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| protected boolean isStringOrVoidType(ObjectInspector oi) { |
|  |

|  |
| --- |
| if (oi.getCategory() == Category.PRIMITIVE) { |
|  |

|  |
| --- |
| if (PrimitiveGrouping.STRING\_GROUP |
|  |

|  |
| --- |
| == PrimitiveObjectInspectorUtils.getPrimitiveGrouping( |
|  |

|  |
| --- |
| ((PrimitiveObjectInspector) oi).getPrimitiveCategory()) |
|  |

|  |
| --- |
| || ((PrimitiveObjectInspector) oi).getPrimitiveCategory() == PrimitiveCategory.VOID) { |
|  |

|  |
| --- |
| return true; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| return false; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private final Text resultText = new Text(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public Object evaluate(DeferredObject[] arguments) throws HiveException { |
|  |

|  |
| --- |
| if (arguments[0].get() == null) { |
|  |

|  |
| --- |
| return null; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| String separator = PrimitiveObjectInspectorUtils.getString( |
|  |

|  |
| --- |
| arguments[0].get(), (PrimitiveObjectInspector)argumentOIs[0]); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| StringBuilder sb = new StringBuilder(); |
|  |

|  |
| --- |
| boolean first = true; |
|  |

|  |
| --- |
| for (int i = 1; i < arguments.length; i++) { |
|  |

|  |
| --- |
| if (arguments[i].get() != null) { |
|  |

|  |
| --- |
| if (first) { |
|  |

|  |
| --- |
| first = false; |
|  |

|  |
| --- |
| } else { |
|  |

|  |
| --- |
| sb.append(separator); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| if (argumentOIs[i].getCategory().equals(Category.LIST)) { |
|  |

|  |
| --- |
| Object strArray = arguments[i].get(); |
|  |

|  |
| --- |
| ListObjectInspector strArrayOI = (ListObjectInspector) argumentOIs[i]; |
|  |

|  |
| --- |
| boolean strArrayFirst = true; |
|  |

|  |
| --- |
| for (int j = 0; j < strArrayOI.getListLength(strArray); j++) { |
|  |

|  |
| --- |
| if (strArrayFirst) { |
|  |

|  |
| --- |
| strArrayFirst = false; |
|  |

|  |
| --- |
| } else { |
|  |

|  |
| --- |
| sb.append(separator); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| sb.append(strArrayOI.getListElement(strArray, j)); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } else { |
|  |

|  |
| --- |
| sb.append(PrimitiveObjectInspectorUtils.getString( |
|  |

|  |
| --- |
| arguments[i].get(), (PrimitiveObjectInspector)argumentOIs[i])); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| resultText.set(sb.toString()); |
|  |

|  |
| --- |
| return resultText; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public String getDisplayString(String[] children) { |
|  |

|  |
| --- |
| assert (children.length >= 2); |
|  |

|  |
| --- |
| return getStandardDisplayString("concat\_ws", children); |
|  |

|  |
| --- |
| } |
|  |

}

**Task 3**

[**Link: https://acadgild.com/blog/transactions-in-hive/**](Link:%20https://acadgild.com/blog/transactions-in-hive/)

**Refer the above given link for transactions in Hive and implement the operations given in the**

**blog using your own sample data set and send us the screenshot.**