Pig

Built in Function:
eval funtion:
AVG,COUNT_STAR,DIFF,MAX,MIN,SIZE,SUBTRACT,SUM,TOKENIZE,FLATTEN

Differece between count, count(*)
count() is use for particular coloum
count(*) is use for colouns including null values

Home Assingment
1. Display employee information having second max salary without limit operter?
2.in mapreduce mode: Give 10% increment to emp having sal is less than avg salary and store details to /user/umesh/empsav

DIFF - its used to compare two bags or fileds in a tuple

Syntax: DIFF(expresion1, expresion2)

```
dept1 = LOAD
'/home/umesh/newhadoop/NotesForBatch/OtherFiles&Jars/dept.txt' USING
PigStorage(',') AS (dno:int,dname:chararray,dloc:chararray,dsal:int);
dept2 = LOAD
'/home/umesh/newhadoop/NotesForBatch/OtherFiles&Jars/dept1.txt' USING
PigStorage(',') AS (dno:int,dname:chararray,dloc:chararray,dsal:int);
cogroupdept = COGROUP dept1 by dno,dept2 BY dno;
diff_dept = FOREACH cogroupdept GENERATE DIFF(dept1,dept2);
```

SUBRACT:
subtract two bags .It returns a bag which contain tuples of first bag that are not in the second bag.

```
sub_bag = FOREACH cogroupdept GENERATE SUBTRACT(dept1,dept2);
dump sub_bag;
```

TOKENIZE: Its used to split a string in a single tuple and return a bag.

FLATTEN: Its used for Unbag the tuples from a bag

Counting no. of words from a file.

In the local mode

```
lines = LOAD 'Desktop/SYllabus _of_python.txt' as (line:chararray);
word = FOREACH lines GENERATE FLATTEN(TOKENIZE(line) as word;
grp = GROUP words by word;
wordcount = FOREACH grp GENERATE group, COUNT(words);
dump wordcount;
```

or
In the mapreduce mode

```
lines = LOAD '/inputfile.txt' AS (line:chararray);
words = FOREACH lines GENERATE FLATTEN (TOKENIZE(line)) as word;
groupwords = GROUP words BY word;
word_count = FOREACH groupwords GENERATE COUNT(words);
word_count = FOREACH groupwords GENERATE group, COUNT(words);
dump word_count;
```

Date and Time Function

ToDate,ToString,CurrentTime, GetDay,GetHour,DaysBetween

Get Month from doj of emp

```
emp =LOAD '/home/umesh/newhadoop/NotesForBatch/OtherFiles&Jars/emp.txt'
USING PigStorage(',') AS
(eid:int,ename:chararray,esal:int,dno:int,doj:chararray);
empdate =FOREACH emp GENERATE ToDate(doj,'yyyy/MM/dd HH:mm:ss') as
(dojdate:DateTime);
ddata = FOREACH empdate GENERATE
GetDay(dojdate),GetMonth(dojdate),CurrentTime();
dump ddata

# Employee did maximum service
grp= group emp ALL;

serviceyears = FOREACH empdate GENERATE
YearsBetween(CurrentTime(),dojdate) as service;
```

Assingments

Loal mode:
1.Display employee eid,name who have done maximum serice in my comapany?
2.count the people who joined my company this year?

Mapreduce mode:
3.Give 10% increment to people spend more than 3 years and store this result on hdfs?
4.Give 5% increment to people spend more in between 30 and 35 and store this result on hdfs?

/*
PiggyBank csv, XML json(java standard object notation) handling.

diference between csv and txt file
csv - csv file which proper comma separated value and which proper column contain which comma seperted but in but in the one column like address colunm contain- house no , ap gopalwadi, it will in the one column but in txt file it will be separate field for each comma seperted

name,mobile,address
umesh,862303759, daund */

```
--Online Advertisment --> APT application ---> Spring/Hibernate --> click
--> .log, .csv, .json --> process and store --> table(Final static
table)-->report


--Registar tat jar file to pig

REGISTER /home/umesh/newhadoop/NotesForBatch/OtherFiles&Jars/piggybank-
0.16.0.jar;

--load the csv file using the piggybank jar file


cars = LOAD
'/home/umesh/newhadoop/NotesForBatch/OtherFiles&Jars/cars.csv' USING
org.apache.pig.piggybank.storage.CSVExcelStorage(',') As
      (buying:chararray,main:chararray,doora:chararray,persion:chararray,
lug_boot:chararray,safetly:chararray,remark:chararray);


--To read the XML find using the piggybank

--XML file : XPath way
 Register piggyback.jar
 DEFINE XPath org.apache.pig.piggybank.evaluation.xml.XPath();
a = LOAD
'/home/umesh/newhadoop/NotesForBatch/OtherFiles&Jars/testXML.xml' USING
org.apache.pig.piggybank.storage.XMLLoader('document') as (x:chararray);

b = FOREACH a GENERATE
XPath(x,'document/url'),XPath(x,'document/category'),XPath(x,'document/us
ercount');

c = LOAD
'/home/umesh/newhadoop/NotesForBatch/OtherFiles&Jars/testXML.xml' USING
org.apache.pig.piggybank.storage.XMLLoader('review') as (x:chararray);

d = FOREACH c GENERATE XPath(x,'review');

e = CROSS b,d

dump e;


--json Handling

--Simple json

fistjson = LOAD
'/home/umesh/newhadoop/NotesForBatch/OtherFiles&Jars/first.json' USING
JsonLoader('food:chararray,person:chararray,amount:int');

secondjson = LOAD
'/home/umesh/newhadoop/NotesForBatch/OtherFiles&Jars/second.json/second.j
son' USING
JsonLoader('recipe:chararray,ingredients:{(name:chararray)},inventor:(nam
e:chararray,age:int)');

thirdjson = LOAD
'/home/umesh/newhadoop/NotesForBatch/OtherFiles&Jars/third.json' USING
```

```
JsonLoader('recipe:chararray,ingredients:{(name:chararray)},inventor:(nam
e:chararray,age:int)');

store relation_name  'path of file'


-- in the third file we can create the filed name(field_name) while
definig the schema it will get created automatically



 -- UDF - user defined function using Java:

-- Eclipse -> Create a project --> build pig jars --> extends a class -->
Export as a jar --> Register jar --> use that function as some name

-- 4th power of number FourthPower(3) => 3*3*3*3 = 27*3 = 81
--Assignment: Factorial of number using function -> Fact(0) => 120

--Extend EvalFun in pig..
-- Eclipse -> New Project->Right click on project --> new Folder --> copy
some jars from pig folder

--- Build path => export as jar as Desktop

--Code for the udf jar file
/*
package myudf;

import org.apache.pig.EvalFunc;
import org.apache.pig.data.Tuple;
import java.io.IOException;
public class FourthPower extends EvalFunc<Integer> {

      @Override
      public Integer exec(Tuple arg) throws IOException{
            if(arg == null  || arg.size()==0)
                  return null;
            else {
            int number = (Integer) arg.get(0);
            return number*number*number*number;
         }
      }
}
*/


--- Implementation

REGISTER 'the jar file which you created for udf'
dept = LOAD 'path' USING PigStorage(',') AS
(dno:int,dname:chararray,dloc:chararray,dsal:int);
dnopower = FOREACH dept GENERATE package_name.class_name(column_name);

-- Assignments
dnofacto = FOREACH dept GENERATE myudf.Facto(dno);
--1. Give 30% hike to emp whose salary is maximum and years spend are
more and save this data to HDFS?
--2. NoOfVowels(String) EX: NoOfVowels('abcd') =>1?
--3. Count the people whose day of joining is an odd number .Use own UDF?
```

```
--Date 04/01/2019 10:07:04

-- Word Count By using the Mapreduce

-- Map Reduce Program ..Demo word count
-- Pig in 4 to 5 lines
---Java in 30 to 40 lines
-- Scala 1 to 2 lines
--Python 2 to 3 lines


--Link for jar file --
https://drive.google.com/file/d/1fwdKCLpRu5e42AYhztne7t17Spg6bXL6/view?us
p=sharing

-- Static class - the class which singleton which can not be changed
-- Steps
--1. put the file in hdfs by using the put command
--2. sudo hadoop jar 'path of jarfile which you created'
package_name.Class_name 'location of input file' 'location of
output_location where you want put your output'


--Java programm for word_count

/*


package mrprogram;

import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hsadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.util.GenericOptionsParser;

public class myWordCount {

      public static void main(String[] args) throws Exception
     {
           Configuration c = new Configuration();

           String[] files = new GenericOptionsParser(c,
args).getRemainingArgs();

           Path input = new Path(files[0]);

           Path output = new Path(files[1]);

           Job j = new Job(c, "wordcount");

           j.setJarByClass(myWordCount.class);

           j.setMapperClass(MapForWordCount.class);

           j.setReducerClass(ReduceForWordCount.class);

           j.setOutputKeyClass(Text.class);

           j.setOutputValueClass(IntWritable.class);

           FileInputFormat.addInputPath(j, input);

           FileOutputFormat.setOutputPath(j, output);

           System.exit(j.waitForCompletion(true) ? 0 : 1);

     }

     public static class MapForWordCount extends
                 Mapper<LongWritable, Text, Text, IntWritable> {

           public void map(LongWritable key, Text value, Context con)
                       throws IOException, InterruptedException
           {

                 String line = value.toString();

                 StringTokenizer tokenizer = new StringTokenizer(line);

                 while(tokenizer.hasMoreTokens()) {
                       value.set(tokenizer.nextToken());
                       con.write(value, new IntWritable(1));
                 }
           }

     }

     public static class ReduceForWordCount extends
                 Reducer<Text, IntWritable, Text, IntWritable>
     {

           public void reduce(Text word, Iterable<IntWritable> values,
Context con)
                       throws IOException, InterruptedException
           {
                 int sum = 0;

                 for (IntWritable value : values)
                  {
                        sum += value.get();

                  }
                  con.write(word, new IntWritable(sum));

            }

      }
}

*/

-- hadoop jar Desktop/MrProgramTest.jar mrprogram.myWordCount
/inputfile.txt /mrdir
--hdfs dfs -cat /mrdir/part-r-00000
-- you will see the output like this
/*  are     2
as    8
beautiful   2
care  1
look  1
love  1
not   1
only  1
or    1
people      1
share 1
talk  1
they  7
walk  1

*/




-- Words count in spark-scala


/*
scala> val data = sc.textFile("/home/umesh/inputfile.txt")
data: org.apache.spark.rdd.RDD[String] = /home/umesh/inputfile.txt
MapPartitionsRDD[3] at textFile at <console>:24

scala> data.top(2)
res1: Array[String] = Array("umesh zagade ", they are only as beautiful
as they love)

scala> val step1 = data.flatMap(line => line.split(" "))
step1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[5] at flatMap
at <console>:25

scala> step1.collect()
res2: Array[String] = Array(people, are, not, as, beautiful, as, they,
look, as, they, walk, or, as, they, talk, they, are, only, as, beautiful,
as, they, love, as, they, care, as, they, share, "", umesh, zagade)
                                      ^
scala> val step2 = step1.map(word => (word,1))
```

```
step2: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[6] at
map at <console>:25

scala> step2.collect()
res3: Array[(String, Int)] = Array((people,1), (are,1), (not,1), (as,1),
(beautiful,1), (as,1), (they,1), (look,1), (as,1), (they,1), (walk,1),
(or,1), (as,1), (they,1), (talk,1), (they,1), (are,1), (only,1), (as,1),
(beautiful,1), (as,1), (they,1), (love,1), (as,1), (they,1), (care,1),
(as,1), (they,1), (share,1), ("",1), (umesh,1), (zagade,1))

scala> val step3 = step2.reduceByKey(_ + _)
step3: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[7] at
reduceByKey at <console>:25

scala> step3.collect()
res4: Array[(String, Int)] = Array((are,2), (love,1), (umesh,1),
(only,1), (as,8), ("",1), (talk,1), (they,7), (zagade,1), (not,1),
(people,1), (or,1), (look,1), (care,1), (beautiful,2), (walk,1),
(share,1))

*/
```