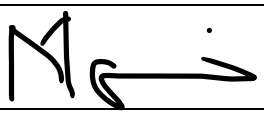
SCHOOL OF INFORMATION TECHNOLOGY

MASTER OF INFORMATION TECHNOLOGY

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Faculty of Engineering, Built Environment and Information Technology

INDIVIDUAL ASSIGNMENT COVER PAGE

Name of Student	Mabel Mapuwei
Student Number	U18362983
Name of Module	Big data management
Module Code	MIT 806
Name of Lecturer	Prof Hanlie Smuts
Date of Submission	31 October 2022
Contact telephone number	0730470122
E-mail address	U18362983@tuks.co.za
Declaration:	*I declare that this assignment, submitted by me, is my own work and that I have referenced all the sources that I have used.*
Signature of Student	
Date received	
Signature of Administrator	
Mark	
Date	
Signature of Lecturer	

Contents

Part 1 MapReduce

The Dataset

The data was collected to study the eCommerce behaviour from multi category store by the OpenCDP project. It was downloaded from Kaggle[3]. It contains 285 million user events or user actions on an eCommerce website selling different categories of items. It was collected for a period of seven months (from October 2019 to April 2020) from the store website. Each row in the file represents an event. All events are related to products and users. Each event is like many-to-many relation between products and users Kechinov M[4]. The fields in the data set are event_time, event_type, product_id, category_id, category_code, brand, price, user_id, and user_session. I was looking for transactions or events data with the intention to analyse people behaviours with regards to window shopping and making a decision to buy an item. For this specific exercise I would like to see which customers are always on our website viewing, carting as well as purchasing our products. In an ideal situation the data would be streamed hourly at a frequency chosen by each business units depending on the insights they want to draw from it.

Only the data from the month of October 2019 was used for the project due to the technical resources constraints.

Map Reduce Algorithm

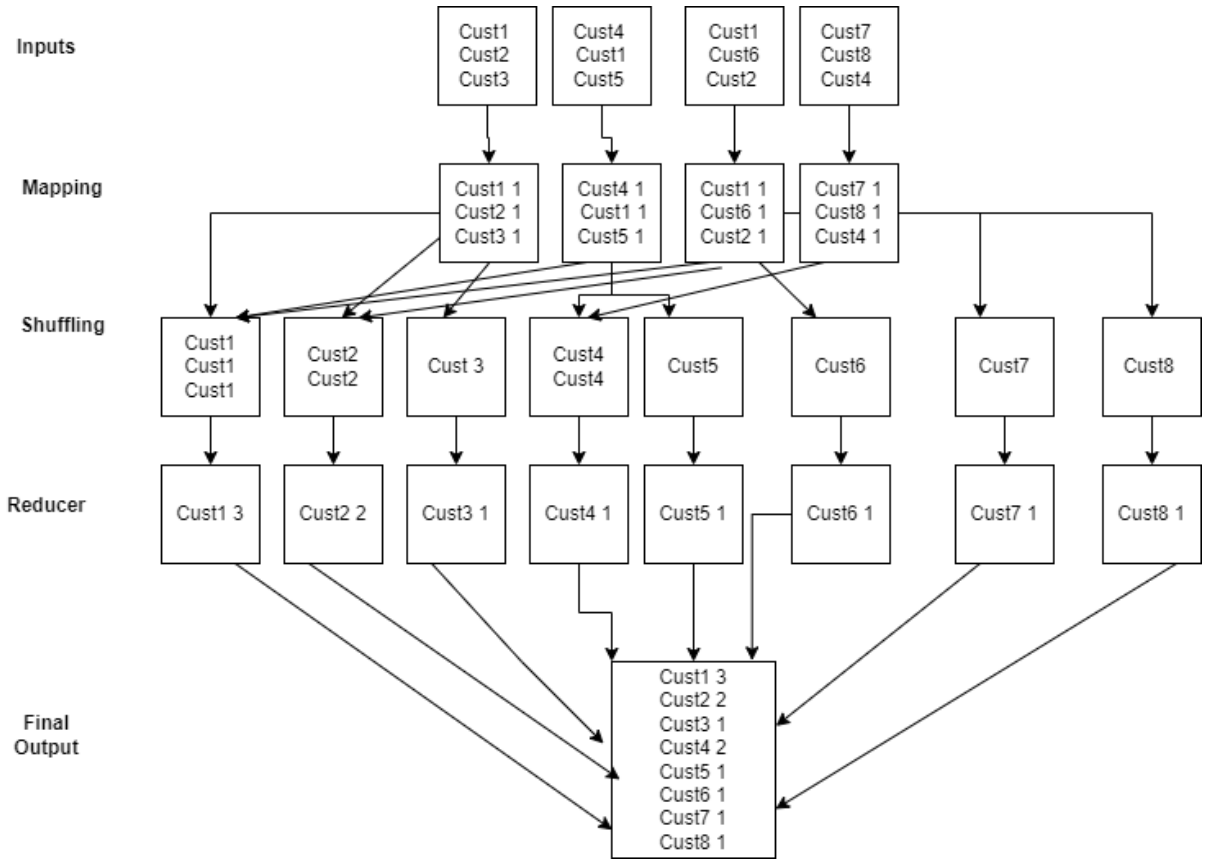
The mapReduce algorithm has been written in java. It consists of three classes, ViewsMapper1 which is the mapper class, ViewsReducer1 which is the reducer class and ViewsDriver1 which is the driver class containing the main class from which we call the mapper and reducer. The code was extracted from a tutorial on guru99[5]. Modifications were done to align with the environment as well as the data being used.

The mapper does the splitting of the data and maps it according to the same values.

The reducer will shuffle the data then reduces the attributes combining the ones with the same values.

Running the code with a small file with two hundred records, there were only 2 splits however when a bigger file with over six hundred and seventy five million records was used there were forty three splits.

The diagram below illustrates the algorithm.[5]



The attribute chosen

The attribute chosen is the user_id which is the customer with the intention to see how many shoppers check the website as well as who the frequent customers are. Further processing will be to see what these customers are looking for, how often they visit the website that is to view items and after how many views do they buy. This data will help the organisation to service their top clients perfectly offering them promotions as well as some personalised services to reward them for their loyalty. Such actions will lead to more customers being drawn to buy from the shop and it will give the organisation a competitive advantage, leading to more profits.

Results

The result from the MapReduce is a list of all customers and how many times they did an action on the websites. Action being view, cart or purchase. A further analysis will be to pull out the top customers, drill down their actions and classify them to determine the suitable value add services to offer to them.

The data has reduced significantly which makes it easier to process on a normal database or using the available visualisation tools.

Part 2 Visualisation

Among the information and insights that can be drawn from the data about the customer activity, the customer activity will be analysed. Having been pre-processed in Hadoop the records are in a format that can be easily sorted and analysed.

Imported the output from Hadoop into python and below is the top 5 records. The actions are the number of times each customer made an action on the website.

```
In [68]: #show first five rows of record
         custlist.head(5)
```

Out[68]:

	cust_id	actions
0	83503497	1
1	184265397	6
2	195082191	1
3	200673532	4
4	205053188	2

Summary of the data

```
custlist.describe()
```

Out[71]:

	cust_id	actions
count	3022290.000	3022290.000
mean	540467341.968	14.045
std	19472125.924	32.774
min	33869381.000	1.000
25%	520576182.000	2.000
50%	544165817.500	4.000
75%	558285614.750	13.000
max	566280860.000	7436.000

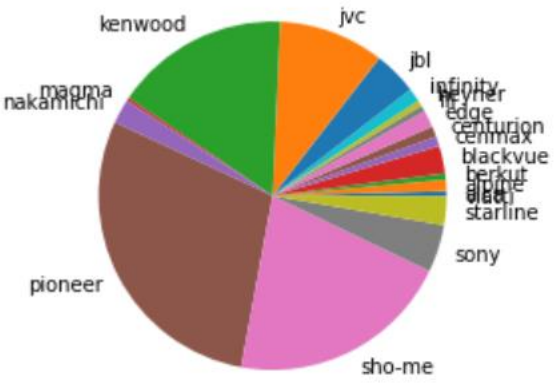
Data is sorted

```
In [72]: #Sorting the values and listing
         df = custlist.sort_values(by=['actions'], ascending=False)
         df.head

In [73]:

Out[73]: <bound method NDFrame.head of           cust_id  actions
         40078    512475445     7436
         21429    512365995     4013
         976095   526731152     2912
         44955    512505687     2894
         123299   513021392     2862
         ...            ...      ...
         1918090  554259735        1
         1918092  554259739        1
         1918093  554259804        1
         1918097  554259898        1
         3022289   64078358        1

         [3022290 rows x 2 columns]>
```

Result

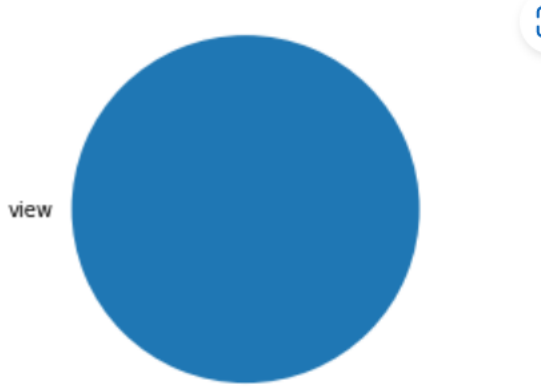
Customer 512475445 has the highest number of events 7436. It is unusual that one person views products for that count and the customer has been viewing all products. This customer needs to be checked and investigated as it looks like it is a robot.

```
Name: user_id, dtype: int64
```



The same customer is only viewing and for the whole month has not bought anything.

```
Name: user_id, dtype: int64
```



References

1. F.J. Ohlhorst (2012. Nov 28) Big Data Analytics: Turning Big Data into Big Money, John Wiley & Sons
2. J. Brownlee, (2020). Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python. Machine Learning Mastery
3. Kaggle (2019) eCommerce behaviour data from multi category store [online] Available: https://www.kaggle.com/datasets/mkechinov/ecommerce-behavior-data-from-multicategory-store
4. M. Kechinov(2019) Available: https://www.kaggle.com/datasets/mkechinov/ecommercebehavior-data-from-multi-category-store
5. Taylor, D. (2022, October 29). *Hadoop & Mapreduce Examples: Create first program in Java*. Guru99. Retrieved October 30, 2022, from https://www.guru99.com/create-your-first-hadoop-program.html

APPENDIX

Code

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;
import java.util.*;
```

```
public class ViewsDriver1 {
   public static void main(String[] args) {
      JobClient my_client = new JobClient();
      // Create a configuration object for the job
      JobConf job_conf = new JobConf(ViewsDriver1.class);

      // Set a name of the Job
      job_conf.setJobName("ViewsPerProduct");

      // Specify data type of output key and value
      job_conf.setOutputKeyClass(Text.class);
      job_conf.setOutputValueClass(IntWritable.class);

      // Specify names of Mapper and Reducer Class
      job_conf.setMapperClass(ViewsMapper1.class);
      job_conf.setReducerClass(ViewsReducer1.class);

      // Specify formats of the data type of Input and output
      job_conf.setInputFormat(TextInputFormat.class);
      job_conf.setOutputFormat(TextOutputFormat.class);

      // Set input and output directories using command line arguments,
      //arg[0] = name of input directory on HDFS, and arg[1] =  name of output directory to be created
to store the output file.

      FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
      FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));

      my_client.setConf(job_conf);
      try {
         // Run the job
         JobClient.runJob(job_conf);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
```

```
 static class ViewsMapper1 extends MapReduceBase implements Mapper <LongWritable, Text, Text,
IntWritable> {

            IntWritable one = new IntWritable(1);

           public void map(LongWritable key, Text value, OutputCollector <Text, IntWritable> output,
Reporter reporter) throws IOException {

                        String valueString = value.toString();
```

```
                String[] SalesData = valueString.split(",");
                output.collect(new Text(SalesData[7]), one);
        }
}



static class ViewsReducer1 extends MapReduceBase implements Reducer<Text, IntWritable, Text,
IntWritable> {

        public void reduce(Text t_key, Iterator<IntWritable> values,
OutputCollector<Text,IntWritable> output, Reporter reporter) throws IOException {
                Text key = t_key;
                int frequencyForViews = 0;
                while (values.hasNext()) {
                        // replace type of value with the actual type of our value
                        IntWritable value = (IntWritable) values.next();
                        frequencyForViews += value.get();

                }
                output.collect(key, new IntWritable(frequencyForViews));
        }
}

}

PYTHON
!/usr/bin/env python
# coding: utf-8

# In[1]:


import csv
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn
import seaborn as sns


custlist = pd.read_csv("c:/AWorkfile/archive/part-00000.csv", header='infer', sep=",")
custlist .columns


# In[2]:


#show first five rows of record
custlist.head(5)
```

In[3]:

```
#The last rows of the dataset
custlist .tail()
```

In[4]:

```
#Dataset shape
custlist.shape
```

In[5]:

```
#Summary of the dataset
pd.set_option('display.float_format', lambda x: '%.3f' % x)
custlist .describe()
```

In[6]:

```
#Sorting the values and listing and checking customers with highest actions
df = custlist.sort_values(by=['actions'], ascending=False)
df.head
```

In[8]:

```
#Checking the customers with the most views and writting to a file
df_first_10 = df.head(10)
df_first_10
```

In[ ]:

In[9]:

```
top100 = custlist.nlargest(100, 'actions')
top100.to_csv('c:/AWorkfile/archive/top100.csv')
custlist.head()
```

In[10]:

```
#Counting customers number of actions
count1 = custlist.groupby('actions')['cust_id'].count()
count1
```

In[11]:

```
#Plotting the how many customers had an action in a count of events
plt.plot(count1)
plt.show()
```

In[12]:

```
import pandas as pd
from matplotlib.pyplot import pie, axis, show
get_ipython().run_line_magic('matplotlib', 'inline')

df =  pd.read_csv('c:/AWorkfile/archive/Topcust.csv')
df.head()

sums = df.groupby('brand')["user_id"].count()
print(sums)
axis('equal');
pie(sums, labels=sums.index);
show()
```

In[13]:

```
df =  pd.read_csv('c:/AWorkfile/archive/Topcust.csv')
sums = df.groupby('event_type')["user_id"].count()
print(sums)
axis('equal');
pie(sums, labels=sums.index);
show()
```

In[ ]: