Analysing Online Retail Transactions using Big Data Framework

Research · April 2019

DOI: 10.13140/RG.2.2.22682.70081

1 author:

Aashish Prasad
National College of Ireland
8 PUBLICATIONS **6** CITATIONS

Analysing Online Retail Transactions using Big Data Framework

Aashish Prasad
X17170826
MSc in Data Analytics
National College of Ireland
Dublin, Ireland

28th April 2019

***Abstract*—Easy access to the internet has broadened the use of online retail and shopping websites. An analysis of the enormous amount of data generated from such transactions is important as part of business growth and development. In this research, we used a distributed computing approach to analyse big data using Hadoop. The dataset was processed and loaded into MySQL as an input for other data processing frameworks. We used sqoop for data transfer from MySQL into HDFS. As per the business objectives, the tasks were distributed among Java, Pig, Hive and Spark frameworks for which the results were generated and stored into HBase database. Further, the outputs were visualized to provide graphical business insights.**

***Keywords*—** Hadoop, MapReduce, Sqoop, Pig, Hive, Spark, HBase, MySQL, HDFS

I. Introduction

In the era of internet, data growth is a constant process. This has opened better opportunities for consumer market analysis in terms of quality with larger samples. Many organizations involved in direct and indirect interaction with the consumer market have realized the potential of data analysis of such large datasets for business growth and expansion. With the growing number of internet access to people, the consumer demand for products and services have also increased over the online retail and shopping websites.[1]

To meet such business requirements, many approaches have been used proposed using traditional data management techniques, such as relational database management system, which often result in low system performance due to architectural incompetence. Here, comes Hadoop, a distributed computing framework that follows MapReduce programming model for handling large dataset and well known for its fault tolerance feature [2]. It provides the advantage of high-speed processing with its data splitting technique.

This research aims to analyse online retail transaction dataset using MapReduce implementation of Java, Pig, Hive and Spark. Also, technologies such as Hadoop Distributed File System (HDFS), Sqoop, MySQL and HBase has been used for data storage and processing. The business objectives and research questions are elaborated in the next section.

A. Objective and Research Questions

The objective of the research is to find meaningful insights that can be useful in the development of an online retail store. This objective is listed in the following Business queries as mentioned below.

1) What are the top 20 most popular products among customers from the United Kingdom, sold in the first quarter of the year?
2) What is the average number of online orders received on an hourly basis in a week?
3) How much was Net Profit on a Quarterly basis for the year 2011?
4) What is the average order amount for each country?
5) In which month the sales revenue reached a maximum?
6) What is the total number of customers from each country?
7) Which top 10 customer ID purchased the most?

B. Motivation for Research

According to [3], the online retail market is price sensitive. Promotional offers and discounts offered to the customers have a high impact on the revenue in this sector. Also, it is necessary to understand the consumer demands for products to maintain stock availability or avoid the unnecessary stockpile up. Finding the correct answer to such critical business queries is only possible through data analytic approach. Hence, a system to analyse big data is an essential step towards business goals. This system will be useful in making correct business decision and strategies.

Section II discusses, some of the previous work and research accomplished using similar technologies that we have used in this research. While Section III, details of dataset and methodology used for prepossessing and overall system development process has been explained. Results for the study using data visualisation techniques has been presented in Section IV.

II. Related Work

To analyse large data on internet traffic, research was conducted by [4], using MapReduce based frameworks. According to the author, MapReduce processes data efficiently with its distributed parallel processing mechanism that includes a map and reduce functions. To analyse ever growing internet traffic data, a traffic analysis system using Hadoop and hive is proposed. The results showed that this Hadoop based system is more efficient and faster when compared with previously proposed solutions.

Research by, [5] discusses a methodology to migrate data relational data storage system like MySQL to from NoSQL databases, such as MongoDB and HBase using Hadoop MapReduce framework on a Hadoop Distributed File System (HDFS). The paper introduces Sqoop, a big data tool, that is used to achieve this migration. The migrated data was further analysed and moved into MongoDB using Hive.

[6] proposed implementation of the Spark framework for analyzing Twitter data. Spark was able to process the tasks within less time which shows the high efficiency of the framework. Spark uses storage such as HDFS for input-output operations. The high-speed performance of spark is contributed by the high rate of access of disk in a second and storing of current data in memory. These features encourage the use of Spark for data processing.

A research survey by, [7] on parallel processing systems explains the working of MapReduce architecture and its efficiency for data processing. It introduces Spark as a clusters computing framework, which is easy to use. It provides parallel operation features such as reduce, foreach and collect that facilities users to use functions to perform filter, map and reduce operations. Further, the author highlights Hive, a warehouse, which provides SQL like processing also known as Hive Query Language(HQL). Hive uses the directory in HDFS to store tables.

Similarly, [2] discusses the use of Apache Pig for data processing which is a high-level scripting language. The pig can access data using HDFS and perform execution parallelly. In contrast to SQL, there is no requirement for the schema in Pig and it can process both semi-structured and unstructured data.

III. Methodology

A. Dataset

The dataset used is downloaded from https://data.world/aprasla0922/online-retail#. It contains transaction details of an Online Retail store with over 541,909 rows. The columns for the dataset are incoiceNo, stockCode, description, quantity, invoiceDate, unitPrice, customerId and country.

B. Data Pre-Processing

In this stage, cleansing of the dataset was performed using the R programming language in RStudio. The process involved the following steps.

1) Generating ID: To identify each row uniquely for analysis ID column was dynamically generated for each record in the table.
2) Splitting column: The invoice date column was split to form year, month, day and time columns.
3) Calculating Amount: A new column, ’amount’ was created in which the total cost of each transaction was calculated stored using unitPrice and quantity columns.
4) Calculating Quarter: A new column, ’quarter’ was created which describes the Quarter of Year in which a specific transaction was done.

After the completing of the above pre-processing tasks, the dataset is stored in a specified location in txt format for further analysis which is discussed in the next section.

C. Process Flow Approach

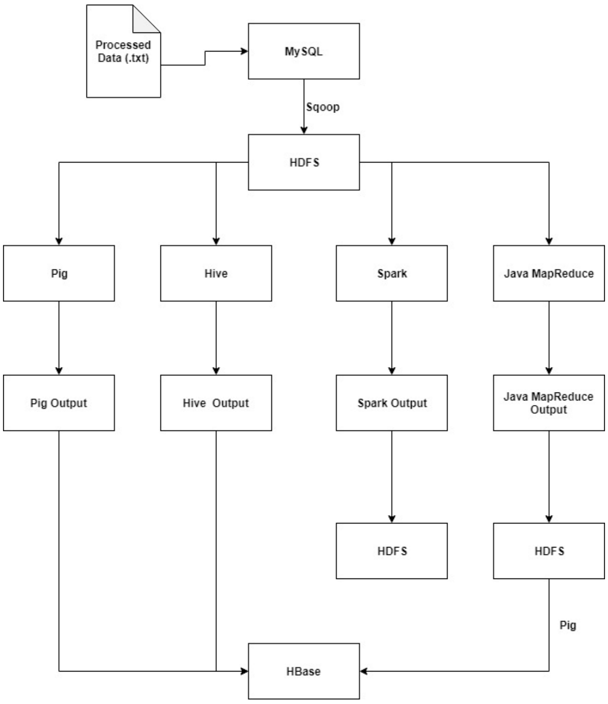


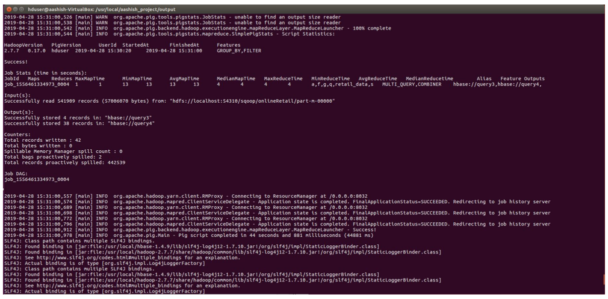
Fig. 1. Process Flow

As shown in 1, the processed data is stored in a MySQL database. Further, the data is moved to HDFS environment using Sqoop, in-order to load into Pig, Hive, Spark and Java MapReduce Frameworks. Each of the four frameworks, execute assigned MapReduce tasks (queries). The output generated by Pig script and Hive is directly stored into the HBase database. Whereas, outputs of Java MapReduce is first stored in HDFS before moving to HBase. To store Java MapReduce outputs into HBase, pig script has been used. The output of Spark SQL is stored in HDFS.

All the above tasks including cleaning of data using R script is automated except for Spark SQL, which is discussed in Section V. To automate the process, a shell script(.sh) has been used.

1) Pig MapReduce: :

The data for Pig queries were fetched from HDFS and the output were stored in HBase table. Figure 2

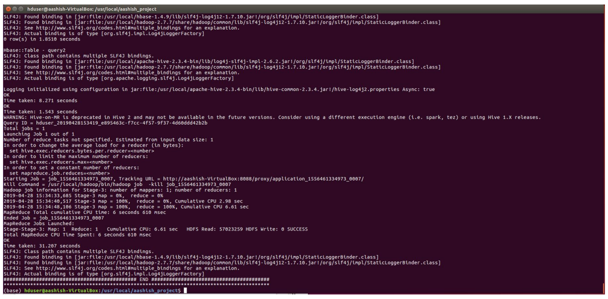


```
hduser@aashish-VirtualBox: /usr/local/aashish_project/output
2019-04-28 15:31:00,526 [main] WARN  org.apache.pig.tools.pigstats.JobStats - unable to find an output size reader
2019-04-28 15:31:00,538 [main] WARN  org.apache.pig.tools.pigstats.JobStats - unable to find an output size reader
2019-04-28 15:31:00,542 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 100% complete
2019-04-28 15:31:00,544 [main] INFO  org.apache.pig.tools.pigstats.mapreduce.SimplePigStats - Script Statistics:

HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.7.7   0.17.0  hduser  2019-04-28 15:30:20     2019-04-28 15:31:00     GROUP_BY,FILTER

Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTime      AvgMapTime      MedianMapTime   MaxReduceTime   MinReduceTime   AvgReduceTime   MedianReducetime        Alias   Feature Outputs
job_1556461334973_0004  1       1       13      13      13      13      4       4       4       4       a,f,g,q,retail_data,s   MULTI_QUERY,COMBINER    hbase://query3,hbase://query4,

Input(s):
Successfully read 541909 records (57006070 bytes) from: "hdfs://localhost:54310/sqoop/onlineRetail/part-m-00000"

Output(s):
Successfully stored 4 records in: "hbase://query3"
Successfully stored 38 records in: "hbase://query4"

Counters:
Total records written : 42
Total bytes written : 0
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 2
Total records proactively spilled: 442539

Job DAG:
job_1556461334973_0004


2019-04-28 15:31:00,557 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2019-04-28 15:31:00,574 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2019-04-28 15:31:00,689 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2019-04-28 15:31:00,698 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2019-04-28 15:31:00,772 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2019-04-28 15:31:00,796 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2019-04-28 15:31:00,912 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2019-04-28 15:31:00,978 [main] INFO  org.apache.pig.Main - Pig script completed in 44 seconds and 881 milliseconds (44881 ms)
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase-1.4.9/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.7/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase-1.4.9/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.7/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
```

Fig. 2. Pig Script Completed

shows successfully executed pig script with outputs stored in HBase.

2) *Hive:* :

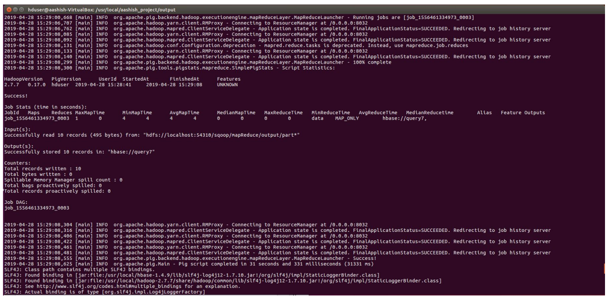


```
hduser@aashish-VirtualBox: /usr/local/aashish_project
SLF4J: Found binding in [jar:file:/usr/local/hbase-1.4.9/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.7/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
0 row(s) in 1.8510 seconds

Hbase::Table - query2
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/apache-hive-2.3.4-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.7/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/usr/local/apache-hive-2.3.4-bin/lib/hive-common-2.3.4.jar!/hive-log4j2.properties Async: true
OK
Time taken: 8.271 seconds
OK
Time taken: 1.543 seconds
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = hduser_20190428153419_e895463c-f7cc-4f57-9f37-4d60ddd42b2b
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1556461334973_0007, Tracking URL = http://aashish-VirtualBox:8088/proxy/application_1556461334973_0007/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1556461334973_0007
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 1
2019-04-28 15:34:33,685 Stage-3 map = 0%,  reduce = 0%
2019-04-28 15:34:40,517 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 2.98 sec
2019-04-28 15:34:48,106 Stage-3 map = 100%,  reduce = 100%, Cumulative CPU 6.61 sec
MapReduce Total cumulative CPU time: 6 seconds 610 msec
Ended Job = job_1556461334973_0007
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 6.61 sec   HDFS Read: 57023259 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 610 msec
OK
Time taken: 31.207 seconds
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase-1.4.9/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.7/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
########################################## END ##########################################
*****************************************************************************************
(base) hduser@aashish-VirtualBox:/usr/local/aashish_project$ 
```

Fig. 3. Hive Script Completed

Similar to Pig, data for Hive was fetched from HDFS and inserted into Hive table. Figure 5 is a snapshot from the automated script showing executed Hive scripts.

3) *Java MapReduce:* :



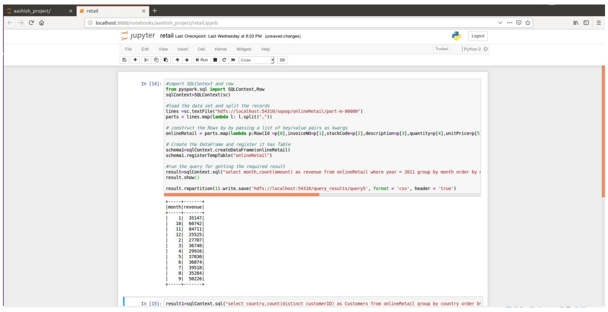
```
hduser@aashish-VirtualBox: /usr/local/aashish_project/output
2019-04-28 15:29:00,668 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Running jobs are [job_1556461334973_0003]
2019-04-28 15:29:06,707 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2019-04-28 15:29:06,762 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2019-04-28 15:29:08,085 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2019-04-28 15:29:08,092 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2019-04-28 15:29:08,131 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.reduces
2019-04-28 15:29:08,133 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2019-04-28 15:29:08,140 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2019-04-28 15:29:08,299 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 100% complete
2019-04-28 15:29:08,300 [main] INFO  org.apache.pig.tools.pigstats.mapreduce.SimplePigStats - Script Statistics:

HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.7.7   0.17.0  hduser  2019-04-28 15:28:41     2019-04-28 15:29:08     UNKNOWN

Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTime      AvgMapTime      MedianMapTime   MaxReduceTime   MinReduceTime   AvgReduceTime   MedianReducetime        Alias   Feature Outputs
job_1556461334973_0003  1       0       4       4       4       4       0       0       0       0       data    MAP_ONLY        hbase://query7,

Input(s):
Successfully read 10 records (495 bytes) from: "hdfs://localhost:54310/sqoop/mapReduce/output/part*"

Output(s):
Successfully stored 10 records in: "hbase://query7"

Counters:
Total records written : 10
Total bytes written : 0
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1556461334973_0003


2019-04-28 15:29:08,304 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2019-04-28 15:29:08,316 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2019-04-28 15:29:08,406 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2019-04-28 15:29:08,422 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2019-04-28 15:29:08,461 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2019-04-28 15:29:08,481 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2019-04-28 15:29:08,555 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2019-04-28 15:29:08,625 [main] INFO  org.apache.pig.Main - Pig script completed in 31 seconds and 331 milliseconds (31331 ms)
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase-1.4.9/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.7/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
```

Fig. 4. Java MapReduce Completed

For Java MapReduce, Java code was complied into JAR file which was integrated into shell script for execution. Figure 5 shows completed Hive script.

4) *Spark:* :

For Spark SQL, Python code was used and the output was stored into HDFS.



aashish_project/
retail
localhost:8888/notebooks/aashish_project/retail.ipynb
jupyter retail Last Checkpoint: Last Wednesday at 8:23 PM (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 2
Run
Code

```
In [14]: #import SQLContext and row
from pyspark.sql import SQLContext,Row
sqlContext=SQLContext(sc)

#load the data set and split the records
lines =sc.textFile("hdfs://localhost:54310/sqoop/onlineRetail/part-m-00000")
parts = lines.map(lambda l: l.split(","))

# construct the Rows by by passing a list of key/value pairs as kwargs
onlineRetail = parts.map(lambda p:Row(Id =p[0],invoiceNO=p[1],stockCode=p[2],description=p[3],quantity=p[4],unitPrice=p[5

# Create the DataFrame and register it has Table
schema1=sqlContext.createDataFrame(onlineRetail)
schema1.registerTempTable("onlineRetail")

#run the query for getting the required result
result=sqlContext.sql("select month,count(amount) as revenue from onlineRetail where year = 2011 group by month order by r
result.show()

result.repartition(1).write.save('hdfs://localhost:54310/query_results/query5', format = 'csv', header = 'true')

+-----+-------+
|month|revenue|
+-----+-------+
|    1|  35147|
|   10|  60742|
|   11|  84711|
|   12|  25525|
|    2|  27707|
|    3|  36748|
|    4|  29916|
|    5|  37030|
|    6|  36874|
|    7|  39518|
|    8|  35284|
|    9|  50226|
+-----+-------+

In [15]: result1=sqlContext.sql("select country,count(distinct customerID) as Customers from onlineRetail group by country order by
```

Fig. 5. Spark SQL in Jupyuter Notebook

D. Technologies Used and Justification for choosing them

The technologies used for the research are as mentioned below.

Apache Hadoop: Hadoop is an open-source framework that supports distributed computing. It is a Java implementation which runs on the MapReduce programming model and uses the Hadoop Distributed File System (HDFS)for data storage [4]. It ensures fault tolerance and eliminates data loss due to replication on server [2]. These features enable Hadoop to process data efficiently in less time and so it has been considered for this research.

Hadoop MapReduce: A programming model based on Java implementation for distributed computing. MapReduce algorithm involves map and reduces tasks. The map function takes data as input and split it before passing its output to the reducer. The research involves implementation of the MapReduce framework along with Pig, Hive and Spark.

Apache Pig: It is an open source framework that provides Pig Latin, a high-level programming language. It supports parallel processing of MapReduce tasks, hence reducing complexity [2]. The research uses pig script for MapReduce operation.

Apache Hive: A data warehouse that simplifies the use of Hadoop. It stores data in a structured format which is easy to understand from the user perspective. It provides SQL like query processing known as Hive query language (HQL) which provides accessibility and manipulation of stored data in HDFS or HBase. This makes Hive suitable for business applications [2].

Apache Spark: It is a distributed processing framework, that works on the in-memory system for fast processing. It is known for its high performance, easy to use and flexibility with efficiency in handling large datasets. Also, it supports application development in languages like python and java using Hadoop based storage system. One important feature of Spark is that it provides Spark SQL, which allows users to write SQL commands to derive information from external data [2]. In this research, we use Spark SQL for query processing using the python programming language.

Sqoop: Sqoop is a big data tool that is used to transfer mass data from MySQL to HDFS. It uses MapReduce for import na dexport of data [5]. Being a fast and reliable tool for data transfer, we have considered the use of Scoop for data movement in our research.

MySQL: A Relational Database Management System to store structured data, known for its reliability, easy to use features, flexibility and high performance. We have used this database as initial storage, which acts as an input for other databases.

Apache HBase: HBase is an open source non-relational database which uses a distributed storage system for storing structured data. On top of Hadoop and HDFS, HBase provides capabilities of BigTable [1]. We have used HBase to store outputs generated by Pig, Hive, Spark and Java MapReduce.

Java: Java is a high-level object-oriented programming language. The MapReduce implementation is developed in java for its Map and reduce functionality. So, the research has used Java for MapReduce.

R: It is a programming language for statistical analysis and supports various operating system environments including UNIX [2]. We have used R data prepossessing and cleansing with RStudio software.

Python: Python is another powerful programming language which we have used with the Spark framework. PySpark is used to implement the Spark programming model using python code[3].

PowerBI: Power BI is a business analytics tool for data visualisation developed by Microsoft. It has been used in our research to represent various data outputs in graphical form.

Tableau: It is a data visualisation tool that is used for dynamic data representation. We have used this tool to represent data in Heat Map.

IV. Results

This section describes the results achieved for the objectives mentioned in Section I-A

What are the top 20 most popular products among customers from the United Kingdom, sold in the first quarter of the year?

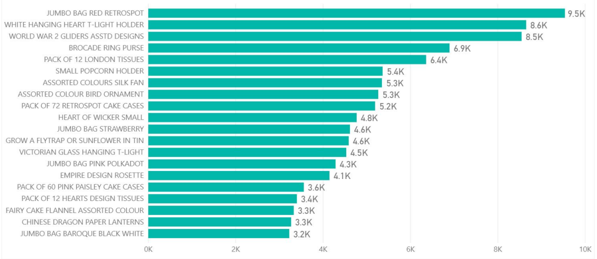


Fig. 6. Most popular Products

Figure 6 shows the top 20 most popular products purchased by customers from the United Kingdom. The units sold for these popular products range from 3.2K to 9.5K of which 'Jumbo Bag Red Retrospot' is the maximum.

What is the average number of daily online orders received on an hourly basis?

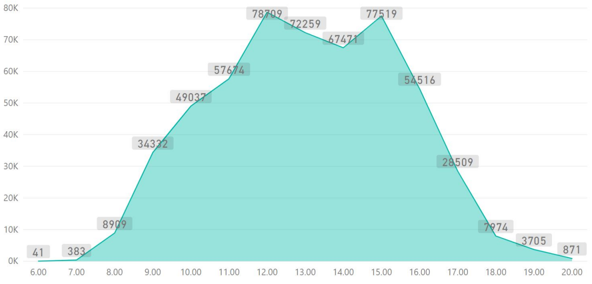


Fig. 7. Average number of orders per hour

As shown in figure 7, most of the orders are received between 12.00 to 15.00 daily. As users during these hours are most active, this duration can be used for promotional offers, special discounts and advertisement which shall reach mass users, improving the chances of business growth.

How much was Net Profit on a Quarterly basis for the year 2011?

Figure 8 describes the annual profit for the year on a quarterly basis. Quarter 4 accounted for the maximum profit with 32.96 percent of total net profit, followed by Quarter 3. This information gives an insight of increasing revenue with respect to time quarterly

[1] https://hbase.apache.org/

[2] https://www.r-project.org/

[3] https://spark.apache.org/docs/0.9.1/python-programming-guide.html

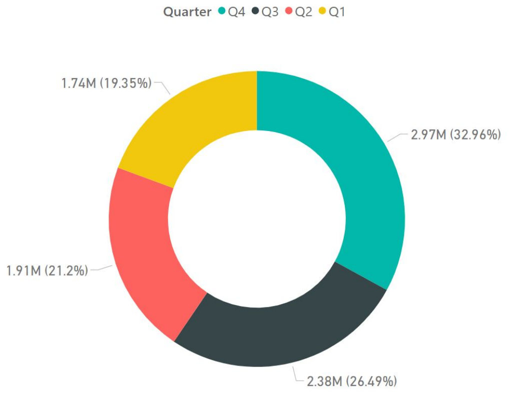


Fig. 8. Quarterly Net Profit

What is the average order amount for each country?

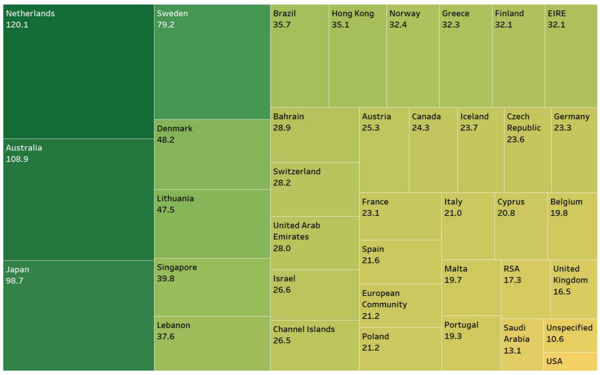


Fig. 9. Order Amount by Country

Figure 9, represents average order value for various countries. It can be seen that order value from the Netherlands and Australia is highest with 120 and 108 Euros. This shows high purchasing power of Customers from these countries, hence, products recommendation can be developed for such customers based on their purchasing habits which can increase the number of orders from these countries.

In which month the sales revenue reached a maximum?

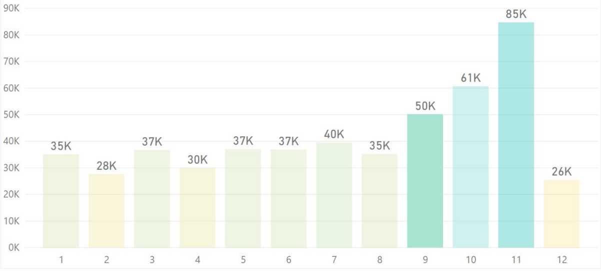
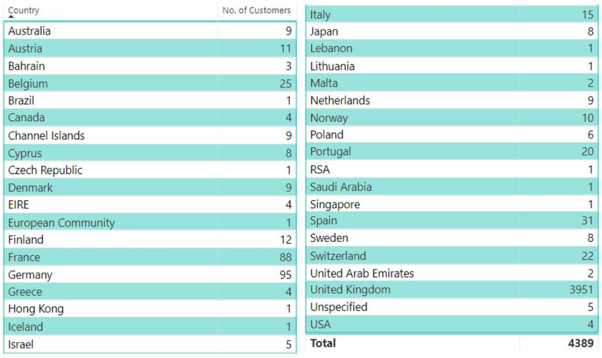


Fig. 10. Monthly Sales Revenue

According to 10, sales revenue in the first half of the year was mostly levelled in the first half of the year. Whereas, in the second half revenue increased in the month of September, October and November. This shows that people tend to buy more in the second half of the year. This can be used as an opportunity for special offers and discounts.

What is the total number of customers from each country?



Country	No. of Customers
Australia	9
Austria	11
Bahrain	3
Belgium	25
Brazil	1
Canada	4
Channel Islands	9
Cyprus	8
Czech Republic	1
Denmark	9
EIRE	4
European Community	1
Finland	12
France	88
Germany	95
Greece	4
Hong Kong	1
Iceland	1
Israel	5
Italy	15
Japan	8
Lebanon	1
Lithuania	1
Malta	2
Netherlands	9
Norway	10
Poland	6
Portugal	20
RSA	1
Saudi Arabia	1
Singapore	1
Spain	31
Sweden	8
Switzerland	22
United Arab Emirates	2
United Kingdom	3951
Unspecified	5
USA	4
Total	**4389**

Fig. 11. Number of Customer by Country

Figure 11 shows the number of customers from each county. Even though the order value for the United Kingdom is very less in comparison with other countries as seen in Figure 9, the number of customers is highest.

Which top 10 customer ID purchased the most?

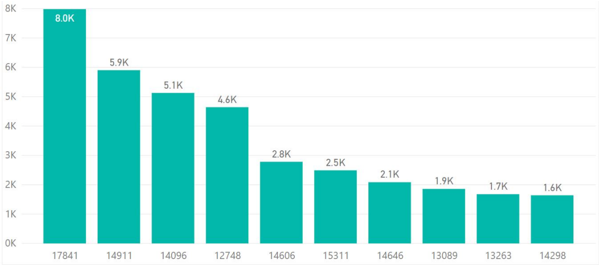


Fig. 12. Top ten Customers with Maximum number of Purchases.

Figure 12 shows the top 10 customers who purchased maximum products. Hence, these customers tend to be high valued for the organisation.

V. Challenges And Limitations

The development of the system involved many challenges. The most challenging part was integrating data cleaning with R script within automation process and saving Hive outputs into HBase within a single script.

Few limitations involve, compatibility issues, the Spark script could not be automated within the automation script. Unlike other outputs, the Spark outputs were stored into HDFS from where they were moved to local directory. The data storage from Spark into HBase was not successful.

VI. Conclusion and Future Work

For analysing Big Data of online retail transactions, we developed a Hadoop MapReduce based system. The objective of the study was achieved as mentioned in Section I-A, to answer business queries. The system uses different MapReduce implementations using Pig, Hive, Java and Spark SQL with python which uses HDFS storage. The data was first imported into R for data cleansing before storing into MySQL database. Further, MySQL database was used as an input for other data processing frameworks. This demonstrate the capability of these platforms as a data processing frameworks.

The future work for this research should focus on including other useful parameters in dataset such as 'customer feedback' and 'product category' which can provide more meaningful insights. Also, the limitations in this research should be overcome by developing possible solutions for the issue mentioned in Section V.

References

[1] C. Jones and N. Livingstone, "Emerging implications of online retailing for real estate: Twenty-first century clicks and bricks," *Journal of Corporate Real Estate*, vol. 17, no. 3, pp. 226–239, 2015.

[2] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih, "Big data technologies: A survey," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 4, pp. 431–448, 2018.

[3] D. Gao, N. Wang, Z. He, and T. Jia, "The bullwhip effect in an online retail supply chain: a perspective of price-sensitive demand based on the price discount in e-commerce," *IEEE Transactions on Engineering Management*, vol. 64, no. 2, pp. 134–148, 2017.

[4] A. M. Hendawi, F. Alali, X. Wang, Y. Guan, T. Zhou, X. Liu, N. Basit, and J. A. Stankovic, "Hobbits: Hadoop and hive based internet traffic analysis," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 2590–2599.

[5] P. M. Bante and K. Rajeswari, "Big data analytics using hadoop map reduce framework and data migration process," in *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*. IEEE, 2017, pp. 1–5.

[6] K. Aziz, D. Zaidouni, and M. Bellafkih, "Real-time data analysis using spark and hadoop," in *2018 4th International Conference on Optimization and Applications (ICOA)*. IEEE, 2018, pp. 1–6.

[7] Y. Zhang, T. Cao, S. Li, X. Tian, L. Yuan, H. Jia, and A. V. Vasilakos, "Parallel processing systems for big data: a survey," *Proceedings of the IEEE*, vol. 104, no. 11, pp. 2114–2136, 2016.