System Setup steps:

1. Installed JAVA 8
2. Installed Apache Hadoop 3.2.4
3. Installed Apache Hive 3.1.3
4. Installed Apache Pig 0.17.0

Apache Hadoop:

Configuration Files:

1. Bashrc

```
#HADOOP PATH
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-arm64/jre
export HADOOP_HOME=/home/priya/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

2. core-site.xml

```
<configuration>
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://localhost:9000</value>
</property>
</configuration>
```

3. hdfs-site.xml

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property><property>
    <name>dfs.name.dir</name>
    <value>/home/priya/hadoop/data/namenode</value>
  </property><property>
    <name>dfs.data.dir</name>
    <value>/home/priya/hadoop/data/datanode</value>
  </property>
</configuration>
```

4. mapred-site.xml

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  </configuration>
```

5. yarn-site.xml

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
<property>

<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class<
/name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
</configuration>
```

Commands to run on Terminal:

SSH Key Configure

```
ssh-keygen -t rsa
```

replace id_rsa as authorized keys

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

add read and write access
```
chmod 640 ~/.ssh/authorized_keys
```

To start all the dameons: `start-all.sh`
To stop all the dameons : `stop-all.sh`
To check all the dameons are running or not : `jps`

To check the UI : http://localhost:9870

Apache Hive:

Configuration files:

1. Update bashrc
```
#HIVE PATH
export HIVE_HOME=/home/priya/hive
export HIVE_CONF_DIR=$HIVE_HOME/conf
export PATH=$PATH:$HIVE_HOME/bin
export CLASSPATH=$CLASSPATH:$HADOOP_HOME/lib/*:.
export CLASSPATH=$CLASSPATH:$HIVE_HOME/lib/*:.
```

2. mapred-site.xml (Path: $HADOOP_HOME/etc/hadoop)
```
<property>
  <name>yarn.app.mapreduce.am.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
<property>
  <name>mapreduce.map.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
<property>
  <name>mapreduce.reduce.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
```

3. hive-site.xml (make a copy of hive-default.xml.template and add below properties)
```
<property>
<name>system:java.io.tmpdir</name>
<value>/tmp/hive/java</value>
</property>
<property>
<name>system:user.name</name>
<value>${user.name}</value>
</property>
```

4. hive-env.sh (make a copy of hive-env.sh.template)

```
#Find the HADOOP_HOME and update the path
HADOOP_HOME=/home/priya/hadoop
```

Commands to run on Terminal:

```
# Hdfs folder creation
hdfs dfs -mkdir -p /user/hive/warehouse
hdfs dfs -mkdir /tmp

# Hdfs folder access permison commands
hdfs dfs -chmod g+w /user
hdfs dfs -chmod g+wx /user
hdfs dfs -chmod g+w /tmp
hdfs dfs -chmod g+wx /tmp

# remove and copy guava file command
cd hive/lib
rm guava-19.0.jar
cp  /home/priya/hadoop/share/hadoop/common/lib/guava-27.0-jre.jar
                  /home/priya/hive/lib

#Edit .sql file location
/home/priya/hive/scripts/metastore/upgrade/derby/hive-schema-3.1.0.derby.sql
comment the first to two create statement
--CREATE FUNCTION "APP"."NUCLEUS_ASCII" (C CHAR(1)) RETURNS INTEGER
                  LANGUAGE JAVA PARAMETER STYLE JAVA READS SQL DATA CALLED
                  ON NULL INPUT EXTERNAL NAME
                  'org.datanucleus.store.rdbms.adapter.DerbySQLFunction.asc
                  ii' ;

--CREATE FUNCTION "APP"."NUCLEUS_MATCHES" (TEXT VARCHAR(8000),PATTERN
                  VARCHAR(8000)) RETURNS INTEGER LANGUAGE JAVA PARAMETER
                  STYLE JAVA READS SQL DATA CALLED ON NULL INPUT EXTERNAL
                  NAME
                  'org.datanucleus.store.rdbms.adapter.DerbySQLFunction.mat
                  ches' ;
#  Intializing the script
cd hive/bin
schematool -dbType derby -initSchema

# Launch hive
hive
```

Apache PIG

Configuration file:

Update bashrc:

```
export PIG_HOME=/home/priya/pig
export PATH=$PATH:$PIG_HOME/bin
export PIG_CLASSPATH=$HADOOP_HOME/etc/hadoop
export PIG_CLASSPATH=$PIG_HOME/conf:$HADOOP_HOME/etc/hadoop/bin
export PIG_CONF_DIR=$PIG_HOME/conf
export PIG_CLASSPATH=$PIG_CONF_DIR
```

Commands to run on Terminal:

```
# To start JobHistoryServer
mapred --daemon start historyserver
jps

# Launch pig
pig
```

Problems with Solution:

- **Problems 1,3,4,5,7 done in Apache Hive**

```
# Create a data directory in hdfs
hadoop fs -mkdir /user/data
# Put the dataset in the data directory
hadoop fs -put /home/priya/online_retail_data.csv /user/data
```

Run the hive shell: hive

```
# Create two tables retail_data and retail_data_raw
CREATE TABLE retail_data (
    RecordNo STRING,
    InvoiceNo STRING,
    StockCode STRING,
    Description STRING,
    Quantity INT,
    InvoiceDate TIMESTAMP,
    Price DOUBLE,
    CustomerID STRING,
    Country STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
ESCAPED BY '\\'
LINES TERMINATED BY '\n'
TBLPROPERTIES ("skip.header.line.count"="1");
```

```
CREATE TABLE retail_data_raw (
    RecordNo STRING,
    InvoiceNo STRING,
    StockCode STRING,
    Description STRING,
    Quantity INT,
    InvoiceDate STRING,
    Price DOUBLE,
    CustomerID STRING,
    Country STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
ESCAPED BY '\\'
LINES TERMINATED BY '\n'
TBLPROPERTIES ("skip.header.line.count"="1");
```

Load the data in retail_data_raw table

```
LOAD DATA LOCAL INPATH '/user/data/online_retail_data.csv' INTO TABLE
retail_data_raw;
```

Overwrite the data in retail_data by parsing the InvoiceDate column

```
INSERT OVERWRITE TABLE retail_data
SELECT
    RecordNo,
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    to_date(from_unixtime(unix_timestamp(InvoiceDate, 'dd-MM-yyyy
HH:mm'))) as InvoiceDate,
    Price,
    CustomerID,
    Country
FROM retail_data_raw;
```

#Drop the retail_data_raw table

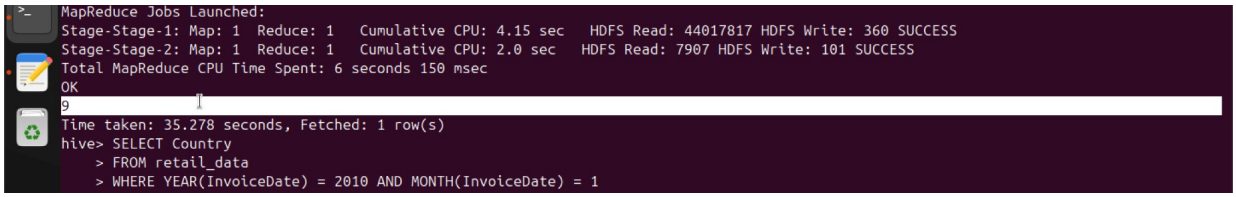
```
Drop table retail_data_raw;
```

1. Total number of unique customers in the "given country".

```
SELECT COUNT(DISTINCT CustomerID)
FROM retail_data
WHERE Country = 'United Kingdom';
```

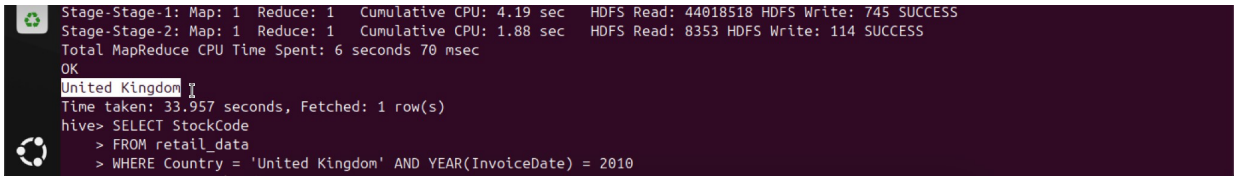
3. Month of 2010 in which maximum number of items were sold.

```
SELECT MONTH(InvoiceDate) AS Month
FROM retail_data
WHERE YEAR(InvoiceDate) = 2010
GROUP BY MONTH(InvoiceDate)
ORDER BY SUM(Quantity) DESC LIMIT 1;
```



4. In the month of January 2010, find the country in which maximum number of items were sold

```
SELECT Country
FROM retail_data
WHERE YEAR(InvoiceDate) = 2010 AND MONTH(InvoiceDate) = 1
GROUP BY Country
ORDER BY SUM(Quantity) DESC
LIMIT 1;
```

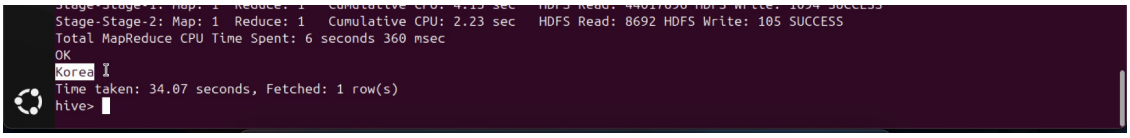


5. The StockCode of the item with the highest number of sales in the "given country" in the year 2010

```
SELECT StockCode
FROM retail_data
WHERE Country = 'United Kingdom' AND YEAR(InvoiceDate) = 2010
GROUP BY StockCode
ORDER BY SUM(Quantity) DESC LIMIT 1;
```

7. The country in which minimum number of sales happened in 2010.

```
SELECT Country
FROM retail_data
WHERE YEAR(InvoiceDate) = 2010
GROUP BY Country
ORDER BY SUM(Quantity) ASC
LIMIT 1;
```



- **<u>Problem 6 in Apache Pig</u>**

6. StockCode of the item for which the maximum revenue was received by sales in the month of December 2010.

Create a directory on hdfs:

```
hadoop fs -mkdir /user/pig_Output
```

Pig Script:

```
REGISTER hdfs:///user/data/online_retail_data.csv;

-- Load the data from HDFS
dataLoad = LOAD 'hdfs:///user/data/online_retail_data.csv' USING
PigStorage(',') AS (Record_No:chararray, Invoice_No:chararray,
StockCode:chararray, Description:chararray, Quantity:chararray,
InvoiceDate:chararray, Price:chararray, Customer_ID:chararray,
Country:chararray);
```

-- Removing the header with the help of RANK

```
dataWithRowNum = RANK dataLoad;
dataNoHeader = FILTER dataWithRowNum BY rank_dataLoad > 1;
```

-- Replacing the '/' to '-' in InvoiceDate

```
normalized_data = FOREACH dataNoHeader GENERATE
    Record_No,
    Invoice_No,
    StockCode,
    Description,
    (float)Quantity AS Quantity,
    REPLACE(InvoiceDate, '/', '-') AS normalized_date,
    (float)Price AS Price,
    Customer_ID,
   Country;
```

-- Filtering the data to get the DEC 2010 data

```
dec_2010_data = FILTER normalized_data BY
    (REGEX_EXTRACT(normalized_date, '^(\\d{2})[/\\-](\\d{2})[/\\-
](\\d{4})\\s', 3) == '2010' AND REGEX_EXTRACT(normalized_date,
'^(\\d{2})[/\\-](\\d{2})[/\\-](\\d{4})\\s', 2) == '12')
    OR
    (REGEX_EXTRACT(normalized_date, '^(\\d{2})[/\\-](\\d{2})[/\\-
](\\d{2})\\s', 3) IS NOT NULL AND CONCAT('20',
REGEX_EXTRACT(normalized_date, '^(\\d{2})[/\\-](\\d{2})[/\\-
](\\d{2})\\s', 3)) == '2010' AND REGEX_EXTRACT(normalized_date,
'^(\\d{2})[/\\-](\\d{2})[/\\-](\\d{2})\\s', 2) == '12');
```

-- Calculate revenue for each StockCode considering absolute values

```
revenue_data = FOREACH dec_2010_data GENERATE StockCode, ABS(Price) *
ABS(Quantity) AS revenue;
```

-- Group by StockCode and calculate total revenue

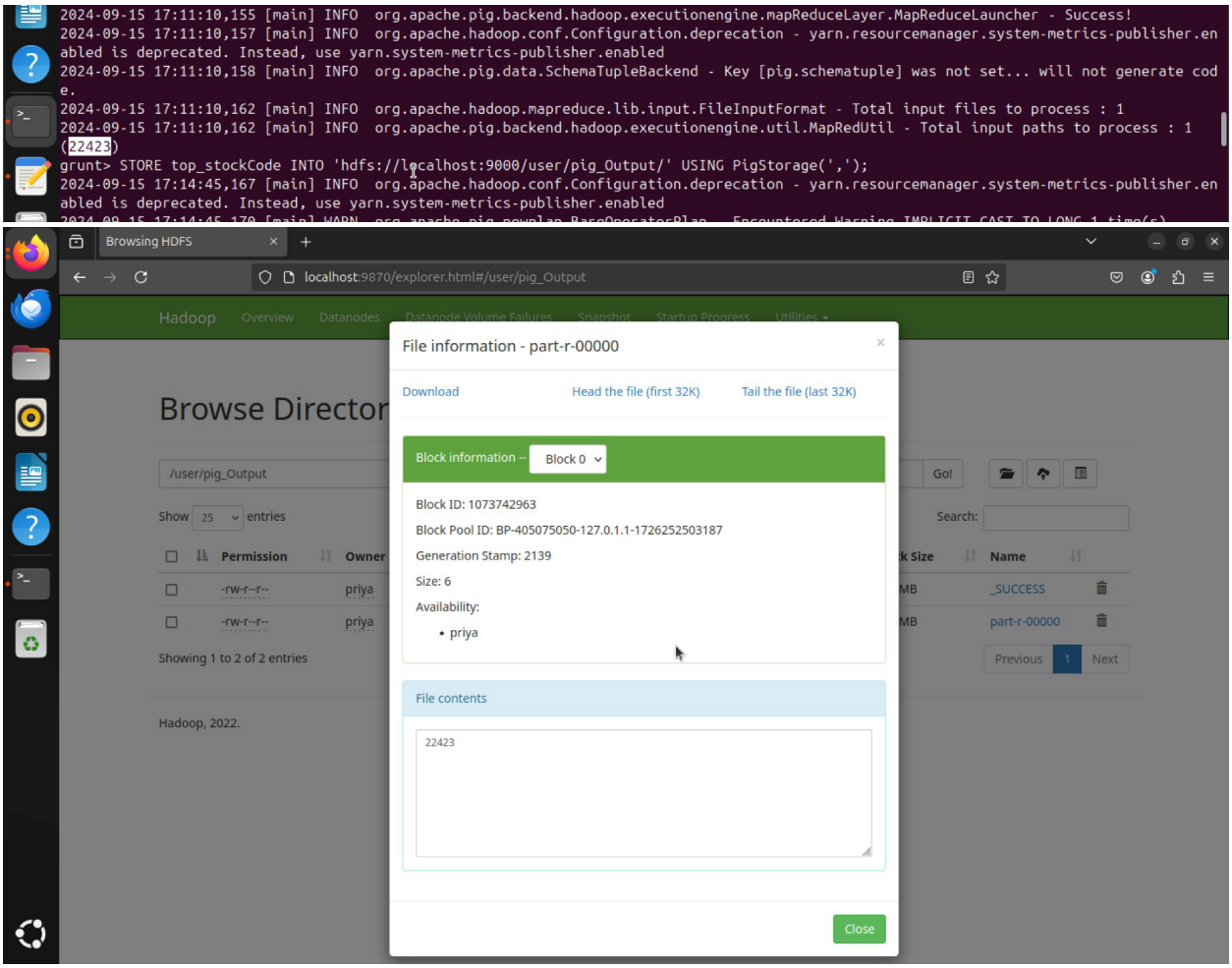
```
grouped_data = GROUP revenue_data BY StockCode;
total_revenue = FOREACH grouped_data GENERATE group,
SUM(revenue_data.revenue) AS total_revenue;
```

-- Find the StockCode with maximum revenue

```
max_revenue = ORDER total_revenue BY total_revenue DESC;
max_revenue_stockcode = LIMIT max_revenue 1;
top_stockCode = FOREACH max_revenue_stockcode GENERATE group;
```

-- Store the result

```
STORE top_stockCode INTO 'hdfs://localhost:9000/user/pig_Output/' USING
PigStorage(',');
```



- **Problem 2 in MapReduce**

2. Country from which the maximum revenue was collected from sales in the month of March 2010.

#Create a directory in local

```
sudo mkdir bds
```

#Give permission to the directory

```
sudo chmod -R 777 bds
```

#Create all 4 files

RetailCountryMapper.java

```
package RetailCountry;

import java.io.IOException;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.util.logging.Logger;
import java.util.logging.Level;
import com.opencsv.CSVReader;
import com.opencsv.CSVParser;
import java.time.LocalDate;
import org.joda.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;

public class RetailCountryMapper extends Mapper<LongWritable, Text, Text,
FloatWritable> {
    private static final Logger LOG =
Logger.getLogger(RetailCountryMapper.class.getName());
    private boolean headerSkipped = false;
    private CSVParser parser = new CSVParser();
    private static final org.joda.time.format.DateTimeFormatter[] formatters
= {
        org.joda.time.format.DateTimeFormat.forPattern("MM-dd-yyyy HH:mm"),
        org.joda.time.format.DateTimeFormat.forPattern("yyyy-MM-dd
HH:mm:ss"),
        org.joda.time.format.DateTimeFormat.forPattern("MM/dd/yyyy
HH:mm:ss"),
        org.joda.time.format.DateTimeFormat.forPattern("MM-dd-yyyy
HH:mm:ss"),
        org.joda.time.format.DateTimeFormat.forPattern("MM/dd/yyyy  HH:mm"),

    };
    @Override
    public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
         if (!headerSkipped) {
        headerSkipped = true;
        return;
      }

      String inputValue = value.toString();
      String[] fields = parser.parseLine(inputValue);
      String quantity = fields[4];
      String invoiceDate = fields[5];
      String price = fields[6];
      String country = fields[8];
```

```
      org.joda.time.LocalDate date = parseDate(invoiceDate);
      if (date != null) {
          int month = date.getMonthOfYear();
          int year = date.getYear();

          if (month == 3 && year == 2010) {
              float revenue = Math.abs(Float.parseFloat(quantity)) *
Math.abs(Float.parseFloat(price));
              context.write(new Text(country), new FloatWritable(revenue));
          }
      }
  }
  private org.joda.time.LocalDate parseDate(String dateStr) {
      for (org.joda.time.format.DateTimeFormatter formatter : formatters) {
          try {
              return org.joda.time.LocalDate.parse(dateStr, formatter);
          } catch (IllegalArgumentException e) {

          }
      }
      LOG.warning("Invalid date format: " + dateStr);
      return null;
  }
}
```

RetailCountryReducer.java

```
package RetailCountry;

import java.io.IOException;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.io.FloatWritable;

import java.util.Map;

import java.util.HashMap;

import java.util.Map.Entry;

public class RetailCountryReducer extends Reducer<Text, FloatWritable, Text,
FloatWritable> {

    private Map<String, Float> revenueMap = new HashMap<>();
```

```
    public void reduce(Text key, Iterable<FloatWritable> values, Context context)
throws IOException, InterruptedException {

        float totalRevenue = 0;

        for (FloatWritable value : values) {

            float revenue = value.get();

            totalRevenue += revenue;

        }

        revenueMap.put(key.toString(), totalRevenue);

    }

     @Override

     public     void     cleanup(Context     context)     throws     IOException,
InterruptedException {

        String maxCountry = null;

        float maxRevenue = 0;

        for (Map.Entry<String, Float> entry : revenueMap.entrySet()) {

            if (entry.getValue() > maxRevenue) {

                maxRevenue = entry.getValue();

                maxCountry = entry.getKey();

            }

        }

        context.write(new Text(maxCountry), new FloatWritable(maxRevenue));

    }

}
```

RetailCountryDriverer.java

```
package RetailCountry;

import org.apache.hadoop.conf.Configured;
```

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
import org.apache.hadoop.io.FloatWritable;
public class RetailCountryDriver extends Configured implements Tool {
    @Override
    public int run(String[] args) throws Exception {
        Job job = Job.getInstance(getConf(), "Max Revenue Country");
        job.setJarByClass(RetailCountryDriver.class);

        TextInputFormat.addInputPath(job, new Path(args[0]));
        TextOutputFormat.setOutputPath(job, new Path(args[1]));

        job.setMapperClass(RetailCountryMapper.class);
        job.setReducerClass(RetailCountryReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(FloatWritable.class);

        return job.waitForCompletion(true) ? 0 : 1;
    }

    public static void main(String[] args) throws Exception {
```

```
        int exitCode = ToolRunner.run(new RetailCountryDriver(), args);

        System.exit(exitCode);

    }

}
```

Manifest.txt

```
Main-Class: RetailCountry.RetailCountryDriver
```

Note: Please note that you have to hit enter key at end of this line.

#Install Opencsv:

```
sudo apt-get install opencsv
```

#Give permission to all files

```
cd bds
sudo chmod +r *.*
```

Export Classpath

```
export CLASSPATH="$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.2.4.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-common-3.2.4.jar:$HADOOP_HOME/share/hadoop/common/hadoop-common-3.2.4.jar:/usr/share/java/opencsv.jar:$HADOOP_HOME/share/hadoop/common/joda-time-2.9.9.jar:~/bds/RetailCountry/*:$HADOOP_HOME/lib/*"
```

#Compile JAVA files

```
javac -d . RetailCountryMapper.java RetailCountryReducer.java RetailCountryDriver.java
```

#Create JAR file

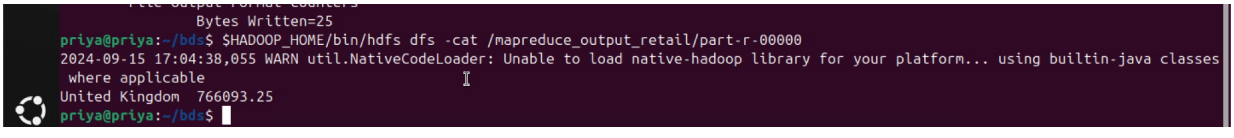
```
jar cfm MaxRevenueCountry.jar Manifest.txt RetailCountry/*.class
```

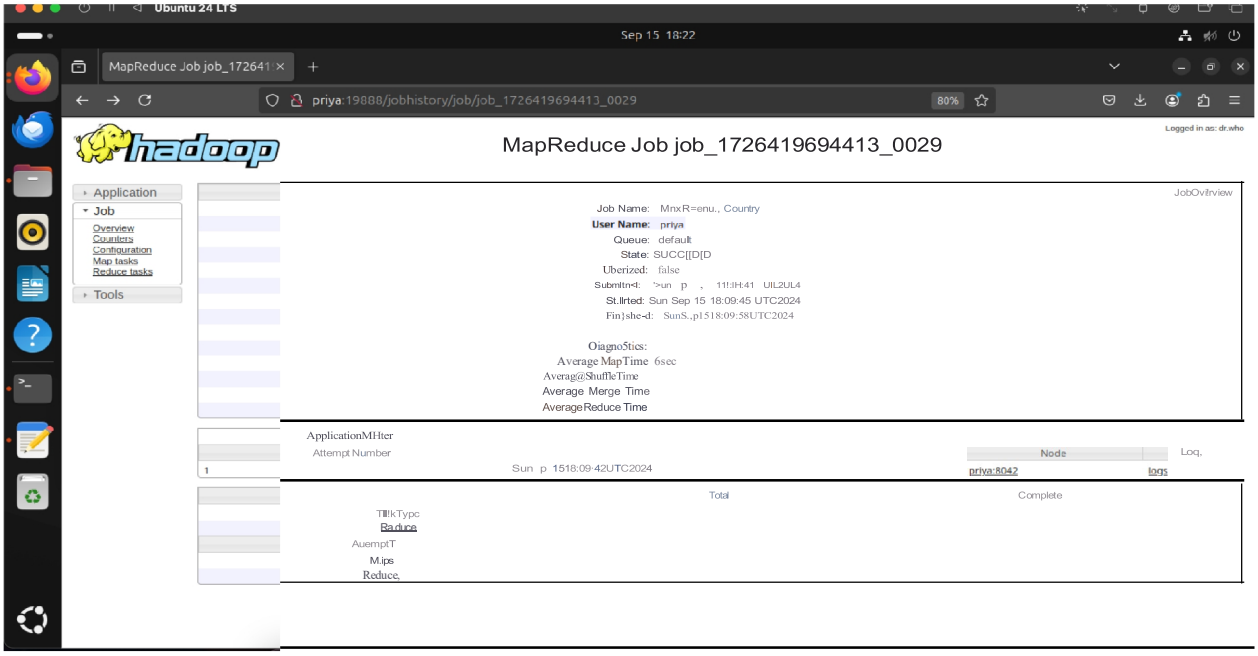
#Run MapReduce

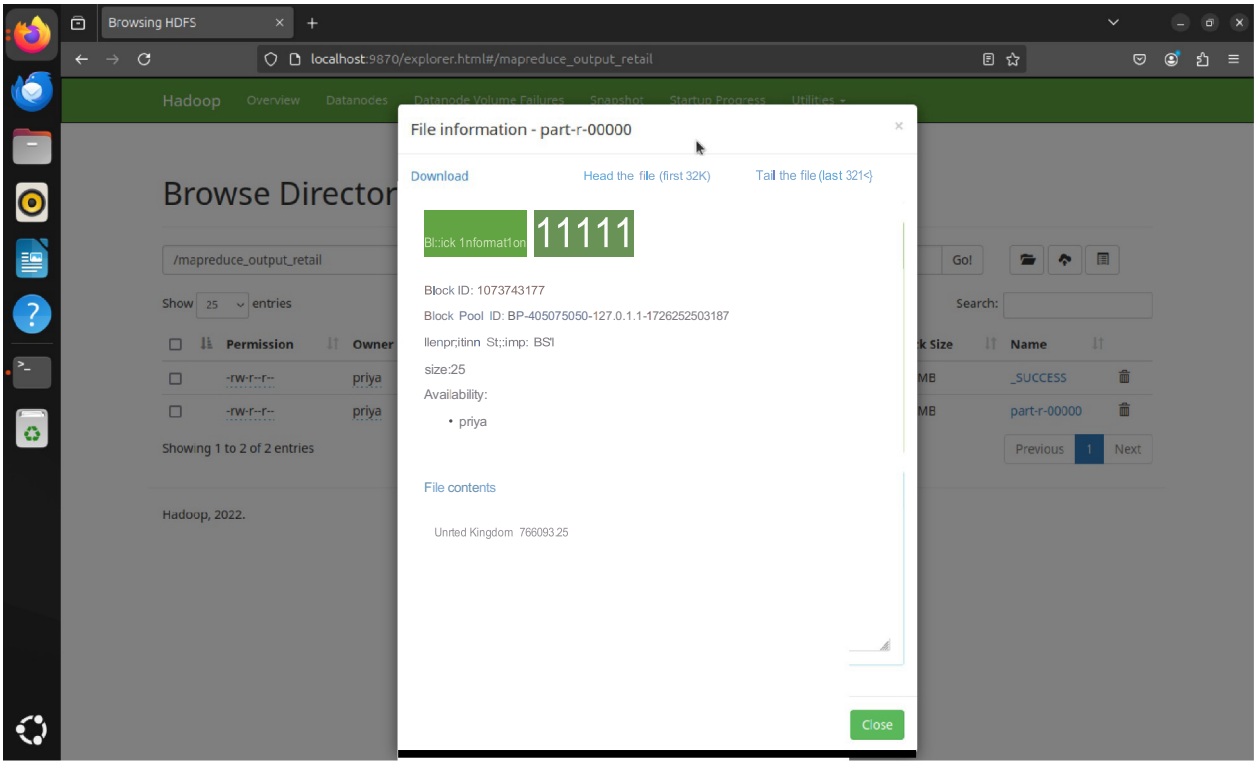
```
hadoop jar MaxRevenueCountry.jar /user/data /mapreduce_output_retail
```

#To see the results

```
$HADOOP_HOME/bin/hdfs dfs -cat /mapreduce_output_retail
/part-r-00000
```




Ubuntu 24 LTS
Sep 15 18:22
MapReduce Job job_172641
priya:19888/jobhistory/job/job_1726419694413_0029
80%
hadoop
MapReduce Job job_1726419694413_0029
Application
Job
Overview
Counters
Configuration
Map tasks
Reduce tasks
Tools
JobOvlrview
Job Name: MnxR=enu., Country
User Name: priya
Queue: default
State: SUCC[[D[D
Uberized: false
St.llrted: Sun Sep 15 18:09:45 UTC2024
Fin}she-d: SunS.,p1518:09:58UTC2024
Oiagno5tics:
Average MapTime 6sec
Averag@ShuffleTime
Average Merge Time
AverageReduce Time
ApplicationMHter
Attempt Number
Node
Log.
1
Sun p 1518:09:42UTC2024
priya:8042
logs
Total
Complete
TIIlkTypc
Reduce
AuemptT
M.ips
Reduce.


Browsing HDFS
localhost:9870/explorer.html#/mapreduce_output_retail
Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Browse Director
/mapreduce_output_retail
Go!
Show 25 entries
Search:
Permission
Owner
Size
Name
-rw-r--r--
priya
_SUCCESS
part-r-00000
Showing 1 to 2 of 2 entries
Previous
1
Next
Hadoop, 2022.
File information - part-r-00000
Download
Head the file (first 32K)
Tail the file (last 321<)
Bl::ick 1nformat1on
11111
Block ID: 1073743177
Block Pool ID: BP-405075050-127.0.1.1-1726252503187
llenpr;itinn St;;imp: BS1
size:25
Availability:
priya
File contents
Unrted Kingdom 766093.25
Close