Assignment 2 (BDS)

102 Group Members:

* [KRISHNA KUMAR MISHRA](https://bits-pilani.instructure.com/groups/11730/users/3076)
* [MEERA R](https://bits-pilani.instructure.com/groups/11730/users/2816)
* [NAVEENKUMAR T](https://bits-pilani.instructure.com/groups/11730/users/2886)

**Business context**

A large multi-national retail chain has sales orders data across regions and different sales channels for a large variety of item types. The business team wants to use this data to analyze various aspects of sales  - e.g., top selling items in a region, regions with maximum profit in a certain item type, if there is a significant difference in revenue in two item types across regions etc.

**Problem statement:**

 As the data analytics team, use the sales transaction data set with about 100K records to answer these questions below —

1. Average unit\_price by country for a given item type in a certain year
2. Total units\_sold by year for a given country and a given item type
3. Find the max and min units\_sold in any order for each year by country for a given item type. Use a custom partitioner class instead of default hash based.
4. What are the top 10 order id for a given year by the total\_profit

**Infrastructure Setup :**

1. Install docker in the system
2. Install git in the system
3. git clone git@github.com:big-data-europe/docker-hadoop.git
4. cd docker-hadoop
5. docker-compose up -d
6. docker ps
7. now enter containers
8. docker exec -it namenode bash
9. docker exec -it resourcemanager bash
10. docker exec -it nodemanager bash
11. docker exec -it historyserver bash
12. docker exec -it datanode bash
13. install python on all running containers of hadoop
    1. apt-get update
    2. apt-get -y install wget gcc make build-essential zlib1g-dev libncurses5-dev libgdbm-dev libnss3-dev libssl-dev libreadline-dev libffi-dev curl libbz2-dev
    3. cd /tmp/
    4. wget https://www.python.org/ftp/python/3.7.6/Python-3.7.6.tgz
    5. tar xzf Python-3.7.6.tgz
    6. cd Python-3.7.6
    7. ./configure --enable-optimizations
    8. make altinstall
    9. rm -rf /tmp/Python-3.7.6.tgz /tmp/Python-3.7.6
    10. python3.7 -m pip install pandas
14. create workdir inside namenode container
    1. docker exec -it namenode /bin/bash
    2. cd /
    3. mkdir map\_reduce
    4. chmod -R 777 map\_reduce
    5. exit()
15. prepare code directory structure
    1. put mapper.py and reducer.py in q1 directory related to first question
    2. put mapper.py and reducer.py in q2 directory related to second question
    3. put mapper.py and reducer.py in q3 directory related to third question
    4. put mapper.py and reducer.py in q4 directory related to fourth question
    5. keep geosales.csv parallel to questions specific directories
    6. keep hadoop-streaming-3.2.1.jar parallel to questions specific directories
16. put code inside namenode container
    1. docker cp hadoop-streaming-3.2.1.jar namenode:/map\_reduce/
    2. docker cp geosales.csv namenode:/map\_reduce/
    3. docker cp q1 namenode:/map\_reduce/
    4. docker cp q2 namenode:/map\_reduce/
    5. docker cp q3 namenode:/map\_reduce/
    6. docker cp q4 namenode:/map\_reduce/
    7. docker exec -it namenode /bin/bash
    8. hadoop fs -mkdir -p /input
    9. hdfs dfs -put ./input/geosales.csv /input/

**Solution of problem statement:**

1.Average unit\_price by country for a given item type in a certain year

**Mapper.py**

|  |
| --- |
| **import** sys **import** sys **for** line **in** sys.stdin:  *# Setting defaults column names* column\_index\_dict = {**'index'**:0,**'region'**:1,**'country'**:2,**'item\_type'**:3,**'sales\_channel'**:4,**'order\_priority'**:5,  **'order\_date'**:6,**'order\_id'**:7,**'ship\_date'**:8,**'units\_sold'**:9,**'unit\_price'**:10,**'unit\_cost'**:11,  **'total\_revenue'**:12,**'total\_cost'**:13,**'total\_profit'**:14}   line = line.strip()  **if** line.startswith(**'index'**):  **continue  else**:  splits = line.split(**","**)  order\_year = splits[column\_index\_dict[**'order\_date'**]].split(**'-'**)[0]  country = splits[column\_index\_dict[**'country'**]]  item\_type = splits[column\_index\_dict[**'item\_type'**]].replace(**" "**,**"\_"**)  unit\_price = splits[column\_index\_dict[**'unit\_price'**]]  print(**f'{**order\_year**},{**country**},{**item\_type**},{**unit\_price**}'**) |

**Reducer.py**

|  |
| --- |
| **import** sys **import** pandas **as** pd  df = pd.read\_csv(sys.stdin) df.columns = [**'order\_year'**, **'country'**, **'item\_type'**, **'unit\_price'**]  df[**"unit\_price"**] = pd.to\_numeric(df[**"unit\_price"**], downcast=**"float"**) output = df.groupby([**'order\_year'**,**'country'**,**'item\_type'**])[[**'unit\_price'**]].mean().reset\_index()  **for** item **in** [**','**.join(element.split()) **for** element **in** output.to\_string(header=**False**,index=**False**).split(**'\n'**)]:  print(item) |

**command :**

|  |
| --- |
| hadoop jar /map\_reduce/hadoop-streaming-3.2.1.jar \  -input /input/geosales.csv \  -output /input/output/q1/ \  -file /map\_reduce/q1/mapper.py -mapper 'python3.7 mapper.py' \  -file /map\_reduce/q1/reducer.py -reducer 'python3.7 reducer.py' |

**Output Result:**

hdfs dfs -cat /input/output/q1/part-00000

2. Total units\_sold by year for a given country and a given item type

**Mapper.py**

|  |
| --- |
| **import** sys **for** line **in** sys.stdin:  *# Setting defaults column names* column\_index\_dict = {**'index'**:0,**'region'**:1,**'country'**:2,**'item\_type'**:3,**'sales\_channel'**:4,**'order\_priority'**:5,  **'order\_date'**:6,**'order\_id'**:7,**'ship\_date'**:8,**'units\_sold'**:9,**'unit\_price'**:10,**'unit\_cost'**:11,  **'total\_revenue'**:12,**'total\_cost'**:13,**'total\_profit'**:14}   line = line.strip()  **if** line.startswith(**'index'**):  **continue  else**:  splits = line.split(**","**)  order\_year = splits[column\_index\_dict[**'order\_date'**]].split(**'-'**)[0]  country = splits[column\_index\_dict[**'country'**]]  item\_type = splits[column\_index\_dict[**'item\_type'**]].replace(**" "**,**"\_"**)  units\_sold = splits[column\_index\_dict[**'units\_sold'**]]  print(**f'{**order\_year**},{**country**},{**item\_type**},{**units\_sold**}'**) |

**Reducer.py**

|  |
| --- |
| **import** sys **import** pandas **as** pd  df = pd.read\_csv(sys.stdin) df.columns = [**'order\_year'**, **'country'**, **'item\_type'**, **'units\_sold'**] output = df.groupby([**'order\_year'**,**'country'**,**'item\_type'**]).units\_sold.sum().reset\_index()  **for** item **in** [**','**.join(element.split()) **for** element **in** output.to\_string(header=**False**,index=**False**).split(**'\n'**)]:  print(item) |

**command :**

|  |
| --- |
| hadoop jar /map\_reduce/hadoop-streaming-3.2.1.jar \  -input /input/geosales.csv \  -output /input/output/q2/ \  -file /map\_reduce/q2/mapper.py -mapper 'python3.7 mapper.py' \  -file /map\_reduce/q2/reducer.py -reducer 'python3.7 reducer.py' |

**Output Result:**

hdfs dfs -cat /input/output/q2/part-00000

3.Find the max and min units\_sold in any order for each year by country for a given item type

**Mapper.py**

|  |
| --- |
| **import** sys **for** line **in** sys.stdin:  *# Setting defaults column names* column\_index\_dict = {**'index'**:0,**'region'**:1,**'country'**:2,**'item\_type'**:3,**'sales\_channel'**:4,**'order\_priority'**:5,  **'order\_date'**:6,**'order\_id'**:7,**'ship\_date'**:8,**'units\_sold'**:9,**'unit\_price'**:10,**'unit\_cost'**:11,  **'total\_revenue'**:12,**'total\_cost'**:13,**'total\_profit'**:14}   line = line.strip()  **if** line.startswith(**'index'**):  **continue  else**:  splits = line.split(**","**)  order\_year = splits[column\_index\_dict[**'order\_date'**]].split(**'-'**)[0]  country = splits[column\_index\_dict[**'country'**]]  item\_type = splits[column\_index\_dict[**'item\_type'**]].replace(**" "**,**"\_"**)  units\_sold = splits[column\_index\_dict[**'units\_sold'**]]  print(**f'{**order\_year**},{**country**},{**item\_type**},{**units\_sold**}'**) |

**Reducer.py**

|  |
| --- |
| **import** sys **import** pandas **as** pd  df = pd.read\_csv(sys.stdin) df.columns = [**'order\_year'**, **'country'**, **'item\_type'**, **'units\_sold'**]  output = df.groupby([**'order\_year'**,**'country'**,**'item\_type'**])[[**'units\_sold'**]].agg([min, max]).reset\_index()  **for** item **in** [**','**.join(element.split()) **for** element **in** output.to\_string(header=**False**,index=**False**).split(**'\n'**)]:  print(item) |

**command :**

|  |
| --- |
| hadoop jar /map\_reduce/hadoop-streaming-3.2.1.jar \  -input /input/geosales.csv \  -output /input/output/q3/ \  -file /map\_reduce/q3/mapper.py -mapper 'python3.7 mapper.py' \  -file /map\_reduce/q3/reducer.py -reducer 'python3.7 reducer.py' |

**Output Result:**

hdfs dfs -cat /input/output/q3/part-00000

3. (partB)Use a custom partitioner class instead of default hash based.

public static class MyPartitioner extends Partitioner<Text,Text>{

public int getPartition(Text key, Text value, int numReduceTasks){

if(numReduceTasks==0){

return 0;

}

if(key.equals(new Text("2013"))){

return 0;

}

if(key.equals(new Text("2014"))){

return 1;

}

if(key.equals(new Text("2015"))){

return 2;

}

if(key.equals(new Text("2016"))){

return 3;

}

if(key.equals(new Text("2017"))){

return 4;

}

if(key.equals(new Text("2018"))){

return 5;

}

if(key.equals(new Text("2019"))){

return 6;

}

if(key.equals(new Text("2020"))){

return 7;

}

if(key.equals(new Text("2021"))){

return 8;

}

}

}

4.What are the top 10 order id for a given year by the total\_profit

**Mapper.py**

|  |
| --- |
| **import** sys **for** line **in** sys.stdin:  *# Setting defaults column names* column\_index\_dict = {**'index'**:0,**'region'**:1,**'country'**:2,**'item\_type'**:3,**'sales\_channel'**:4,**'order\_priority'**:5,  **'order\_date'**:6,**'order\_id'**:7,**'ship\_date'**:8,**'units\_sold'**:9,**'unit\_price'**:10,**'unit\_cost'**:11,  **'total\_revenue'**:12,**'total\_cost'**:13,**'total\_profit'**:14}   line = line.strip()  **if** line.startswith(**'index'**):  **continue  else**:  splits = line.split(**","**)  order\_year = splits[column\_index\_dict[**'order\_date'**]].split(**'-'**)[0]  order\_id = splits[column\_index\_dict[**'order\_id'**]]  total\_profit = splits[column\_index\_dict[**'total\_profit'**]]  print(**f'{**order\_year**},{**order\_id**},{**total\_profit**}'**) |

**Reducer.py**

|  |
| --- |
| **import** sys **import** pandas **as** pd  df = pd.read\_csv(sys.stdin) df.columns = [**'order\_year'**, **'order\_id'**, **'total\_profit'**]  output = df.set\_index([**'order\_id'**]).groupby(**'order\_year'**)[**'total\_profit'**].nlargest(10).reset\_index()  **for** item **in** [**','**.join(element.split()) **for** element **in** output.to\_string(header=**False**,index=**False**).split(**'\n'**)]:  print(item) |

**command :**

|  |
| --- |
| hadoop jar /map\_reduce/hadoop-streaming-3.2.1.jar \  -input /input/geosales.csv \  -output /input/output/q4/ \  -file /map\_reduce/q4/mapper.py -mapper 'python3.7 mapper.py' \  -file /map\_reduce/q4/reducer.py -reducer 'python3.7 reducer.py' |

**Output Result:**

hdfs dfs -cat /input/output/q4/part-00000