Assignment 6: Hadoop MapReduce

Sanjose N , Data Engineering Batch

A FMCG company entered into the instant noodles business two years back. Their higher management has noticed that there is a mismatch in the demand and supply. Where the demand is high, supply is pretty low and where the demand is low, supply is pretty high. In both ways it is an inventory cost loss to the company; hence, the higher management wants to optimize the supply quantity in each and every warehouse in the entire country.

Goal & Objective: The objective of this exercise is to build a model, using historical data that will determine an optimum weight of the product to be shipped each time to the warehouse. Also try to analysis the demand pattern in different pockets of the country so management can drive the advertisement campaign particular in those pockets.

File: FMCG_Data.csv

Data Description:

Variable	Business Definition
Ware_house_ID	Product warehouse ID
WH_Manager_ID	Employee ID of warehouse manager
Location_type	Location of warehouse like in city or village
WH_capacity_size	Storage capacity size of the warehouse
zone	Zone of the warehouse
WH_regional_zone	Regional zone of the warehouse under each zone
num_refill_req_l3m	Number of times refilling has been done in last 3 months
transport_issue_l1y	Any transport issue like accident or goods stolen reported in last one year
Competitor_in_mkt	Number of instant noodles competitor in the market
retail_shop_num	Number of retails shop who sell the product under the warehouse area
wh_owner_type	Company is owning the warehouse or they have get the warehouse on rent
distributor_num	Number of distributors works in between warehouse and retail shops
flood_impacted	Warehouse is in the Flood impacted area indicator
flood_proof	Warehouse is flood proof indicators. Like storage is at some height not directly on the ground
electric_supply	Warehouse have electric back up like generator, so they can run the warehouse in load shedding
dist_from_hub	Distance between warehouse to the production hub in Kms
workers_num	Number of workers working in the warehouse
wh_est_year	Warehouse established year
storage_issue_reported_l3m	Warehouse reported storage issue to corporate office in last 3 months. Like rat, fungus because of moisture etc.
temp_reg_mach	Warehouse have temperature regulating machine indicator

approved_wh_govt_certificat e	What kind of standard certificate has been issued to the warehouse from government regulatory body
wh_breakdown_l3m	Number of time warehouses face a breakdown in last 3 months. Like strike from worker, flood, or electrical failure
govt_check_l3m	Number of time government Officers have been visited the warehouse to check the quality and expire of stored food in last 3 months
product_wg_ton	Product has been shipped in last 3 months. Weight is in tons

MapReduce Problem Statements

Here are specific MapReduce problem statements that can be solved using MapReduce streaming and Python programming. Each problem statement includes the objective, the dataset fields required, and a brief description of how to approach the problem using MapReduce.

Task 1: Demand-Supply Mismatch Analysis

Objective: Identify zones and regional zones with the highest mismatch between demand and supply.
Required Fields: zone, WH_regional_zone, product_wg_ton

Description:

Map: For each warehouse, emit the zone and regional zone as the key and the product weight shipped in the last three months as the value.

Reduce: Aggregate the product weight by zone and regional zone to calculate the total supply. Compare this with known demand data to identify mismatches.

```
#!/usr/bin/python3
"""mapper1.py"""
import sys
for line in sys.stdin:
        line = line.strip()
        data = line.split(",")
        zone = data[4]
        regional_zone = data[5]
        product_weight = data[-1]
        print("%s\t%s\t%s" %(zone,regional_zone,product_weight))
```

```python
#!/usr/bin/python3
"""reducer1.py"""
import sys
records = {}

for line in sys.stdin:
    line = line.strip()
    try:
        zone, sub_zone, weight = line.split("\t")
        weight = int(weight)
    except ValueError:
        continue
    key = (zone,sub_zone)
    if key in records:
        records[key] += weight
    else:
        records[key] = weight

for (zone, sub_zone), weight in records.items():
    print(f'{zone}\t{sub_zone}\t{weight}')
```

```
hadoop@hadoop-VirtualBox:~/factory$ hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.7.6.jar -file /home/hadoop/factory/mapper1.py -mapper mapper1.py
 -file /home/hadoop/factory/reducer1.py -reducer reducer1.py -input /factory -output /factoryoutput/output1
packageJobJar: [/home/hadoop/factory/mapper1.py, /home/hadoop/factory/reducer1.py] [] /tmp/streamjob4798398156568169453.jar tmpDir=null
hadoop@hadoop-VirtualBox:~/factory$ hdfs dfs -cat /factoryoutput/output1/part-00000
East    Zone 3  2526684
East    Zone 1  872338
East    Zone 5  1768074
East    Zone 4  3306171
East    Zone 6  1274236
North   Zone 5  42893115
North   Zone 6  100249991
North   Zone 1  18466131
North   Zone 2  18966332
North   Zone 3  21335735
North   Zone 4  26254519
South   Zone 5  24113697
South   Zone 2  32467899
South   Zone 1  14682866
South   Zone 4  19230670
South   Zone 3  18810119
South   Zone 6  30235650
West    Zone 4  43804669
West    Zone 2  15146537
West    Zone 5  32242727
West    Zone 6  52661774
West    Zone 1  10638197
West    Zone 3  20617692
hadoop@hadoop-VirtualBox:~/factory$
```


/factoryoutput/output1 Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	0 B	7/9/2024, 8:48:54 am	1	128 MB	_SUCCESS
-rw-r--r--	hadoop	supergroup	490 B	7/9/2024, 8:48:54 am	1	128 MB	part-00000

Task 2: Warehouse Refill Frequency Correlation

Objective: Determine the correlation between warehouse capacity and refill frequency.
Required Fields: WH_capacity_size, num_refill_req_l3m

Description:
Map: Extract the number of refill requests (num_refill_req_l3m) and warehouse capacity size (WH_capacity_size) for each warehouse. (For each warehouse, emit the capacity size and the number of refill requests as the value)

Reduce: Aggregate the refill requests by capacity size and calculate the correlation.

```
#!/usr/bin/python3
"""mapper2.py"""
import sys
for line in sys.stdin:
        line = line.strip()
        data = line.split(",")
        warehouse_capacity = data[3]
        refill_req = data[6]
        try:
                print("%s\t%s" % (warehouse_capacity, refill_req))
        except BrokenPipeError:
                pass
```

```
#!/usr/bin/python3
"""reducer2.py"""
import sys
import numpy as np
from collections import defaultdict

capacity_sizes = []
refill_requests = []
capacity = defaultdict(list)
capacity_chart = {'Small': 1, 'Mid': 2, 'Large': 3}

for line in sys.stdin:
        capacity_size, refill_request = line.strip().split('\t')

        try:
                refill_request = int(refill_request)
                if capacity_size in capacity_chart:
                        capacity[capacity_chart.get(capacity_size)].append(refill_request)
                        capacity_sizes.append(capacity_chart.get(capacity_size))
                        refill_requests.append(refill_request)
        except ValueError:
                continue


capacity_sizes = np.array(list(capacity.keys()))
refill_requests = np.array([sum(val)/len(val) for val in capacity.values()])
correlation_matrix = np.corrcoef(capacity_sizes, refill_requests)
correlation = correlation_matrix[0, 1]
print(f"Correlation between warehouse capacity and refill requests: {correlation}")
```

```
hadoop@hadoop-VirtualBox:~/factory$ hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.7.6.jar -file /home/hadoop
/factory/mapper2.py -mapper mapper2.py -file /home/hadoop/factory/reducer2.py -reducer reducer2.py -input /factory -output /factoryou
tput/output2
packageJobJar: [/home/hadoop/factory/mapper2.py, /home/hadoop/factory/reducer2.py] [] /tmp/streamjob3375420241503158401.jar tmpDir=nu
ll
hadoop@hadoop-VirtualBox:~/factory$ hdfs dfs -cat /factoryoutput/output2/part-00000
Correlation between warehouse capacity and refill requests: 0.7349881101354251
hadoop@hadoop-VirtualBox:~/factory$
```

/factoryoutput/output4 Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	0 B	7/9/2024, 8:36:43 am	1	128 MB	_SUCCESS
-rw-r--r--	hadoop	supergroup	1019 B	7/9/2024, 8:36:43 am	1	128 MB	part-00000

Task 3. Transport Issue Impact Analysis

Objective: Analyse the impact of transport issues on warehouse supply efficiency.
Required Fields: transport_issue_l1y, product_wg_ton

Description:
Map: For each warehouse, emit whether a transport issue was reported and the product weight shipped.
Reduce: Aggregate the product weight by transport issue status to assess the impact.

```python
#!/usr/bin/python3
"""mapper3.py"""
import sys
for line in sys.stdin:
        line = line.strip()
        data = line.split(",")
        transport_issue = data[7]
        product_weight = data[-1]
        try:
                print("%s\t%s" %(transport_issue,product_weight))
        except BrokenPipeError:
                pass
```

```python
#!/usr/bin/python3
"""reducer3.py"""
import sys
records = {}

for line in sys.stdin:
    line = line.strip()
    try:
        transport_issue, weight = line.split("\t")
        transport_issue = int(transport_issue)
        weight = float(weight)
    except ValueError:
        continue

    if transport_issue in records:
        records[transport_issue][0] += weight
        records[transport_issue][1] += 1
    else:
        records[transport_issue] = [weight, 1]

sorted_transport_issues = sorted(records.items(), key=lambda x: x[1][0] / x[1][1], reverse=True)

for transport_issue, (total_weight, count) in sorted_transport_issues:
    average_weight = total_weight / count
    print(f'{transport_issue}\t{average_weight:.2f}\t{total_weight:.2f}\t{count}')
```

```
hadoop@hadoop-VirtualBox:~/factory$ hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.7.6.jar -file /home/hadoop
/factory/mapper3.py -mapper mapper3.py -file /home/hadoop/factory/reducer3.py -reducer reducer3.py -input /factory -output /factoryou
tput/output3
packageJobJar: [/home/hadoop/factory/mapper3.py, /home/hadoop/factory/reducer3.py] [] /tmp/streamjob5903802586370431170.jar tmpDir=nu
ll
hadoop@hadoop-VirtualBox:~/factory$ hdfs dfs -cat /factoryoutput/output2/part-00000
cat: `/factoryoutput/output2/part-00000': No such file or directory
hadoop@hadoop-VirtualBox:~/factory$ hdfs dfs -cat /factoryoutput/output3/part-00000
0       23606.14        359167349.00    15215
1       21346.66        99133868.00     4644
4       19171.75        14896451.00     777
2       18858.30        41450553.00     2198
3       17673.04        32129593.00     1818
5       16632.21        5788009.00      348
hadoop@hadoop-VirtualBox:~/factory$
```


/factoryoutput/output3 Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	0 B	7/9/2024, 8:31:43 am	1	128 MB	_SUCCESS
-rw-r--r--	hadoop	supergroup	167 B	7/9/2024, 8:31:43 am	1	128 MB	part-00000

Task 4. Storage Issue Analysis

Objective: Evaluate the impact of storage issues on warehouse performance.

Required Fields: storage_issue_reported_l3m, product_wg_ton

Description:
Map: For each warehouse, emit whether a storage issue was reported and the product weight shipped.

Reduce: Aggregate the product weight by storage issue status to assess the impact.

```python
#!/usr/bin/python3
"""mapper4.py"""
import sys
for line in sys.stdin:
        line = line.strip()
        data = line.split(",")
        storage_issue = data[18]
        product_weight = data[-1]
        print("%s\t%s" %(storage_issue,product_weight))
```

```python
#!/usr/bin/python3
"""reducer4.py"""
import sys
records = {}

for line in sys.stdin:
    line = line.strip()
    try:
        storage_issue, weight = line.split("\t")
        storage_issue = int(storage_issue)
        weight = float(weight)
    except ValueError:
        continue

    if storage_issue in records:
        records[storage_issue][0] += weight
        records[storage_issue][1] += 1
    else:
        records[storage_issue] = [weight, 1]

sorted_storage_issues = sorted(records.items(), key=lambda x: x[1][0] / x[1][1], reverse=True)

for storage_issue, (total_weight, count) in sorted_storage_issues:
    average_weight = total_weight / count
    print(f'{storage_issue}\t{average_weight:.2f}\t{total_weight:.2f}\t{count}')
```

/factoryoutput/output4 Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	0 B	7/9/2024, 8:36:43 am	1	128 MB	_SUCCESS
-rw-r--r--	hadoop	supergroup	1019 B	7/9/2024, 8:36:43 am	1	128 MB	part-00000