HADOOP Assignment 2

MAP-REDUCE

Business Problem:

A FMCG company has entered into the instant noodles business two years back. Their higher management has notices that there is a miss match 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 the 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 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

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.

CODE

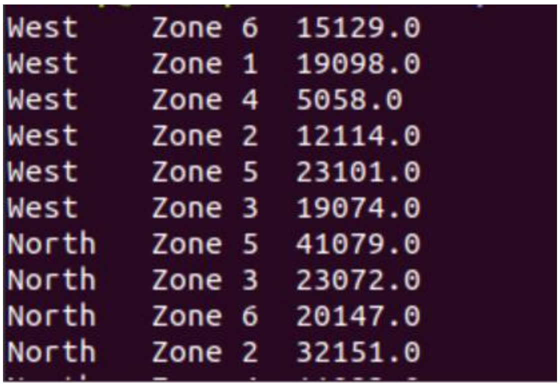
MAPPER

```
#!/usr/bin/python3
"""mapper.py"""

import sys
import csv

for row in csv.reader(sys.stdin):
    print("%s\t%s\t%s"%(row[4],row[5],row[23]))
```

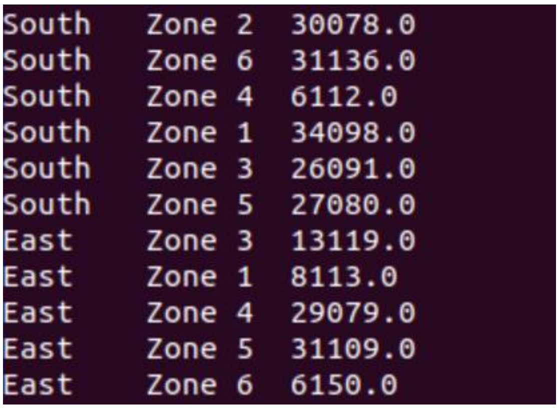
REDUCER

```
#!/usr/bin/python3

import sys
import csv

dict = {}

for line in sys.stdin:
    zone, wh_regional_zone, product_shipped = line.strip().split("\t")
    try:
      product_shipped = float(product_shipped)
    except ValueError:
        continue
    if zone in dict:
        if wh_regional_zone in zone:
            dict[zone][wh_regional_zone]+=product_shipped
        else:
          dict[zone].update({wh_regional_zone:product_shipped})
    else:
      dict[zone]={wh_regional_zone:product_shipped}


for zone in dict:
    for regional in dict[zone]:
        print("%s\t%s\t%s"%(zone,regional, dict[zone][regional]))
```

OUTPUT



```
West    Zone 6  15129.0
West    Zone 1  19098.0
West    Zone 4  5058.0
West    Zone 2  12114.0
West    Zone 5  23101.0
West    Zone 3  19074.0
North   Zone 5  41079.0
North   Zone 3  23072.0
North   Zone 6  20147.0
North   Zone 2  32151.0
```



```
South   Zone 2  30078.0
South   Zone 6  31136.0
South   Zone 4  6112.0
South   Zone 1  34098.0
South   Zone 3  26091.0
South   Zone 5  27080.0
East    Zone 3  13119.0
East    Zone 1  8113.0
East    Zone 4  29079.0
East    Zone 5  31109.0
East    Zone 6  6150.0
```

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.

CODE

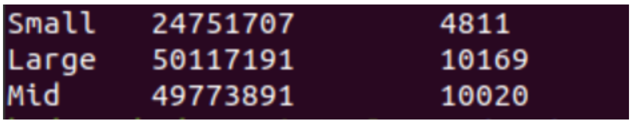
MAPPER

```
#!/usr/bin/env python3

import sys

import csv


for line in csv.reader(sys.stdin):

  if line[0] == 'Ware_house_ID':  # Skip header

    continue

  capacity_size = float(line[3])

  refill_req = int(line[9])

  print(f"{capacity_size}\t{refill_req}")
```

REDUCER

```
#!/usr/bin/env python3
import sys
from collections import defaultdict

capacity_data = defaultdict(list)

for line in sys.stdin:
    capacity_size, refill_req = line.strip().split('\t')
    capacity_size = float(capacity_size)
    refill_req = int(refill_req)
    capacity_data[capacity_size].append(refill_req)

for capacity_size in capacity_data:
    total_refill = sum(capacity_data[capacity_size])
    count = len(capacity_data[capacity_size])
    print(f"{capacity_size}\t{total_refill}\t{count}")
```

OUTPUT



```
Small   24751707        4811
Large   50117191        10169
Mid     49773891        10020
```

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

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.

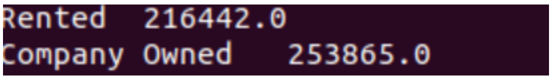
MAPPER

```
#!/usr/bin/env python3
import sys
import csv

for line in csv.reader(sys.stdin):
    if line[0] == 'Ware_house_ID':  # Skip header
        continue
    transport_issue = line[10]
    product_wg_ton = float(line[22])
    print(f"{transport_issue}\t{product_wg_ton}")
```

REDUCER

```
#!/usr/bin/env python3
import sys
from collections import defaultdict

transport_data = defaultdict(float)

for line in sys.stdin:
    transport_issue, product_wg_ton = line.strip().split('\t')
    product_wg_ton = float(product_wg_ton)
    transport_data[transport_issue] += product_wg_ton

for transport_issue in transport_data:
    print(f"{transport_issue}\t{transport_data[transport_issue]}")
```

OUTPUT



```
Rented  216442.0
Company Owned   253865.0
```

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.

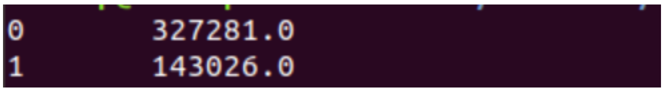
MAPPER

```
#!/usr/bin/env python3

import sys

import csv


for line in csv.reader(sys.stdin):

    if line[0] == 'Ware_house_ID':  # Skip header

        continue

    storage_issue = line[19]

    product_wg_ton = float(line[22])

    print(f"{storage_issue}\t{product_wg_ton}")
```

REDUCER

```
#!/usr/bin/env python3
import sys
from collections import defaultdict

storage_data = defaultdict(float)

for line in sys.stdin:
    storage_issue, product_wg_ton = line.strip().split('\t')
    product_wg_ton = float(product_wg_ton)
    storage_data[storage_issue] += product_wg_ton

for storage_issue in storage_data:
    print(f"{storage_issue}\t{storage_data[storage_issue]}")
```

OUTPUT



```
0       327281.0
1       143026.0
```