Assignment 6

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.

Mapper.py

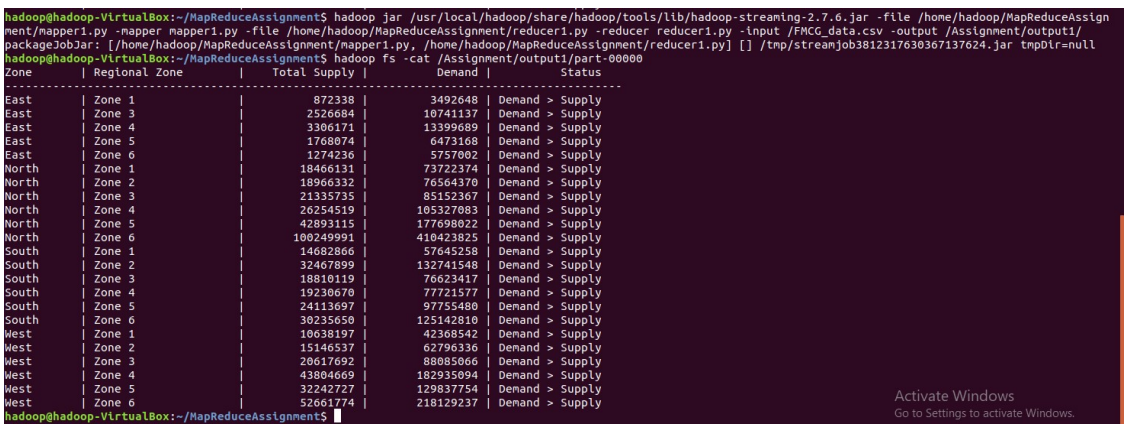
```
#!/usr/bin/python3
"""mapper1.py"""
import sys
# input comes from standard input
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # If the line is not empty
    if line:
        columns = line.split(',')
        if columns:
            zone = columns[4].strip()
            WH_regional_zone = columns[5].strip()
            product_wg_ton = columns[-1].strip()
            refills = columns[6].strip()

            if zone != "zone" and WH_regional_zone != "WH_regional_zone" \
             and product_wg_ton != "product_wg_ton" \
             and refills != "num_refill_req_l3m":
                print('%s,%s,%s,%s' % (zone, WH_regional_zone,
                product_wg_ton, refills))
```

Reducer.py

```
#!/usr/bin/python3
"""reducer1.py"""
```

```
import sys

data = {}

for line in sys.stdin:
    line = line.strip()

    try:
        zone, regional_zone, ton, refill = line.split(",")
        ton = int(ton)
        refill = int(refill)
    except ValueError:
        continue

    key = (zone, regional_zone)

    if key in data:
        data[key][0] += ton
        data[key][1] += refill * ton
    else:
        data[key] = [ton, refill * ton]


print(f"{'Zone':<10} | {'Regional Zone':<20} | {'Total Supply':>15}
 | {'Demand':>15} | {'Status':>15}")
print("-" * 90)

for (zone, regional_zone), values in data.items():
    total_weight, value = values
    print(f"{zone:<10} | {regional_zone:<20} | {total_weight:>15}
    | {value:>15} |
    {'Demand > Supply' if value>total_weight else 'Supply > Demand' }")
```

Output



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.

Mapper.py

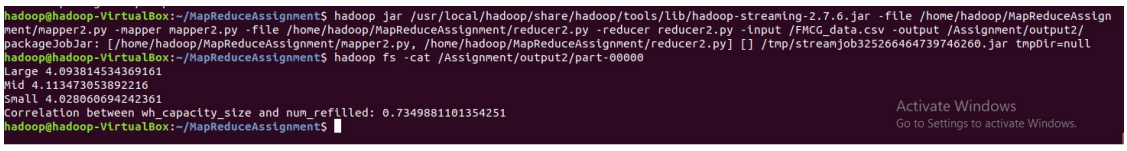
```
#!/usr/bin/python3
"""mapper2.py"""
import sys
# input comes from standard input
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # If the line is not empty
    if line:
        columns = line.split(',')
        if columns:
            capacity_size = columns[3].strip()
            num_refill_req_l3m = columns[6].strip()

            if capacity_size != "capacity_size" and \
            num_refill_req_l3m != "num_refill_req_l3m":
                print('%s,%s' % (capacity_size, num_refill_req_l3m))
```

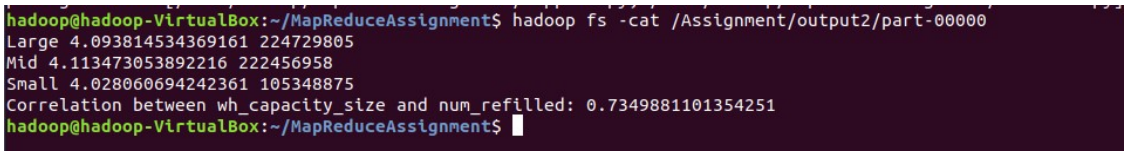
Reducer.py

```
#!/usr/bin/python3
"""reducer2.py"""

import sys
import numpy as np

data={}
encode = {'Large':3,'Mid':2, 'Small':1}

# input comes from STDIN
for line in sys.stdin:
    line = line.strip()
```

```
    capacity, refill = line.split(",")
    try:
        refill = int(refill)
    except:
        continue

    if capacity in data:
        data[capacity][0]+=refill
        data[capacity][1]+=1
    else:
        data[capacity]=[refill,1]

values=[]
sizes=[]

for k, v in data.items():
    avg = v[0]/v[1]
    values.append(avg)
    sizes.append(encode[k])
    print(f"{k} {avg}")


correlation_matrix = np.corrcoef(sizes, values)

correlation_xy = correlation_matrix[0, 1]

print("Correlation between wh_capacity_size and num_refilled:", correlation_xy)
```

Output





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.

Mapper.py

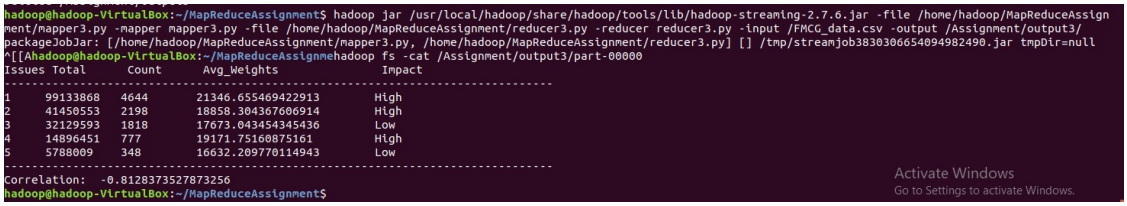
```
#!/usr/bin/python3
"""mapper3.py"""
import sys

# input comes from standard input
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # If the line is not empty
    if line:
        columns = line.split(',')
        if columns:
            trans_issue = columns[7].strip()
            product_wg_ton = columns[-1].strip()

            if trans_issue != "trans_issue" and product_wg_ton != "product_wg_ton":
                print('%s,%s' % (trans_issue, product_wg_ton))
```

Reducer.py

```
#!/usr/bin/python3
"""reducer3.py"""

import sys
import numpy as np

data={}


# input comes from STDIN
for line in sys.stdin:
    line = line.strip()
```

```
    issue, ton = line.split(",")
    try:
        ton = int(ton)
    except:
        continue
    if int(issue) >0:
        if issue in data:
            data[issue][0]+=ton
            data[issue][1]+=1
        else:
            data[issue]=[ton,1]

issues=[]
values=[]
total = []
count = []

for k, v in sorted(data.items()):
    avg = v[0]/v[1]
    values.append(avg)
    total.append(v[0])
    count.append(v[1])
    issues.append(int(k))


sum_avg_total = sum(values)/len(data)

print("Issues | Total | Count | Avg_Weights \t| Impact")
for i in range(len(data)):
    if values[i] > sum_avg_total:
        impact = "High"
    else:
        impact = "Low"
    print(str(issues[i])+"\t"+str(total[i])+" "+
    str(count[i])+"\t"+str(values[i])+" "+impact)


correlation_matrix = np.corrcoef(issues, values)

correlation_xy = correlation_matrix[0, 1]

print("Correlation: ", correlation_xy)
```

Output



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.py

```
#!/usr/bin/python3
"""mapper3.py"""
import sys

# input comes from standard input
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # If the line is not empty
    if line:
        columns = line.split(',')
        if columns:
            storage_issue = columns[-6].strip()
            product_wg_ton = columns[-1].strip()

            if storage_issue != "storage_issue" and \
             product_wg_ton != "product_wg_ton":
                print('%s,%s' % (storage_issue, product_wg_ton))
```

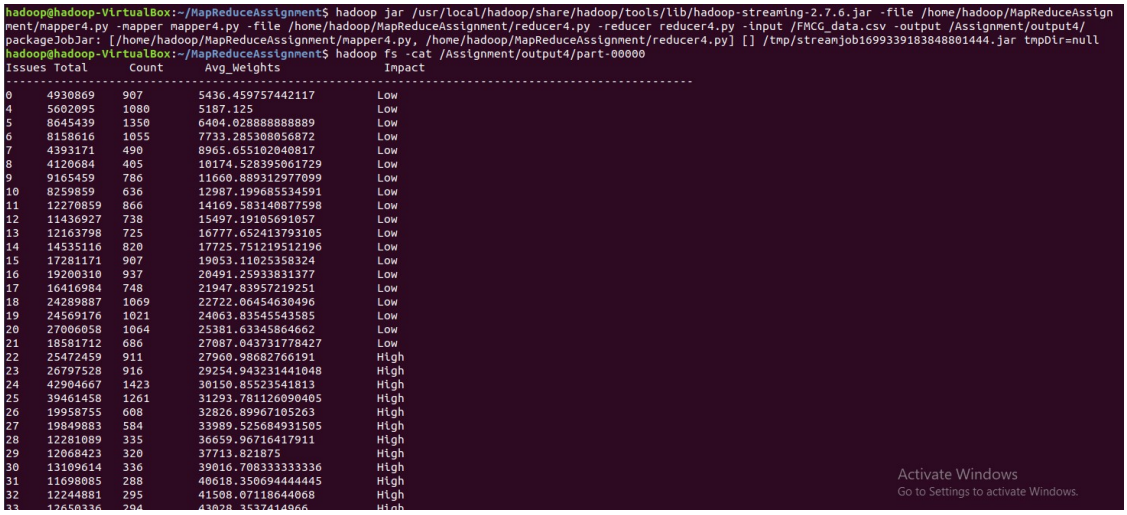
Reducer.py

```
#!/usr/bin/python3
"""reducer3.py"""
```

```
import sys
import numpy as np

data={}


# input comes from STDIN
for line in sys.stdin:
    line = line.strip()

    issue, ton = line.split(",")
    try:
        ton = int(ton)
    except:
        continue
    if int(issue) >0:
        if issue in data:
            data[issue][0]+=ton
            data[issue][1]+=1
        else:
            data[issue]=[ton,1]

issues=[]
values=[]
total = []
count = []

for k, v in sorted(data.items()):
    avg = v[0]/v[1]
    values.append(avg)
    total.append(v[0])
    count.append(v[1])
    issues.append(int(k))


sum_avg_total = sum(values)/len(data)

print("Issues | Total | Count | Avg_Weights \t| Impact")
for i in range(len(data)):
    if values[i] > sum_avg_total:
        impact = "High"
    else:
        impact = "Low"
    print(str(issues[i])+"\t"+str(total[i])+" "+
    str(count[i])+"\t"+str(values[i])+" "+impact)


correlation_matrix = np.corrcoef(issues, values)
```

```
correlation_xy = correlation_matrix[0, 1]

print("Correlation: ", correlation_xy)
```

Output



```
hadoop@hadoop-VirtualBox:~/MapReduceAssignment$ hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.7.6.jar -file /home/hadoop/MapReduceAssign
ment/mapper4.py -mapper mapper4.py -file /home/hadoop/MapReduceAssignment/reducer4.py -reducer reducer4.py -input /FMCG_data.csv -output /Assignment/output4/
packageJobJar: [/home/hadoop/MapReduceAssignment/mapper4.py, /home/hadoop/MapReduceAssignment/reducer4.py] [] /tmp/streamjob1699339183848801444.jar tmpDir=null
hadoop@hadoop-VirtualBox:~/MapReduceAssignment$ hadoop fs -cat /Assignment/output4/part-00000
Issues Total      Count      Avg_Weights              Impact
-----------------------------------------------------------------------------------------------
0     4930869    907        5436.459757442117        Low
4     5602095    1080       5187.125                 Low
5     8645439    1350       6404.028888888889        Low
6     8158616    1055       7733.285308056872        Low
7     4393171    490        8965.655102040817        Low
8     4120684    405        10174.528395061729       Low
9     9165459    786        11660.889312977099       Low
10    8259859    636        12987.199685534591       Low
11    12270859   866        14169.583140877598       Low
12    11436927   738        15497.19105691057        Low
13    12163798   725        16777.652413793105       Low
14    14535116   820        17725.751219512196       Low
15    17281171   907        19053.11025358324        Low
16    19200310   937        20491.25933831377        Low
17    16416984   748        21947.83957219251        Low
18    24289887   1069       22722.06454630496        Low
19    24569176   1021       24063.83545543585        Low
20    27006058   1064       25381.63345864662        Low
21    18581712   686        27087.043731778427       Low
22    25472459   911        27960.98682766191        High
23    26797528   916        29254.943231441048       High
24    42904667   1423       30150.85523541813        High
25    39461458   1261       31293.781126090405       High
26    19958755   608        32826.89967105263        High
27    19849883   584        33989.525684931505       High
28    12281089   335        36659.96716417911        High
29    12068423   320        37713.821875             High
30    13109614   336        39016.708333333336       High
31    11698085   288        40618.350694444445       High
32    12244881   295        41508.07118644068        High
33    12650336   294        43028.353741496966       High
```