ASSIGNMENT 6

Task 1: Demand-Supply Mismatch Analysis

Objective: Identify zones and regional zones with the highest mismatch between demand and supply.

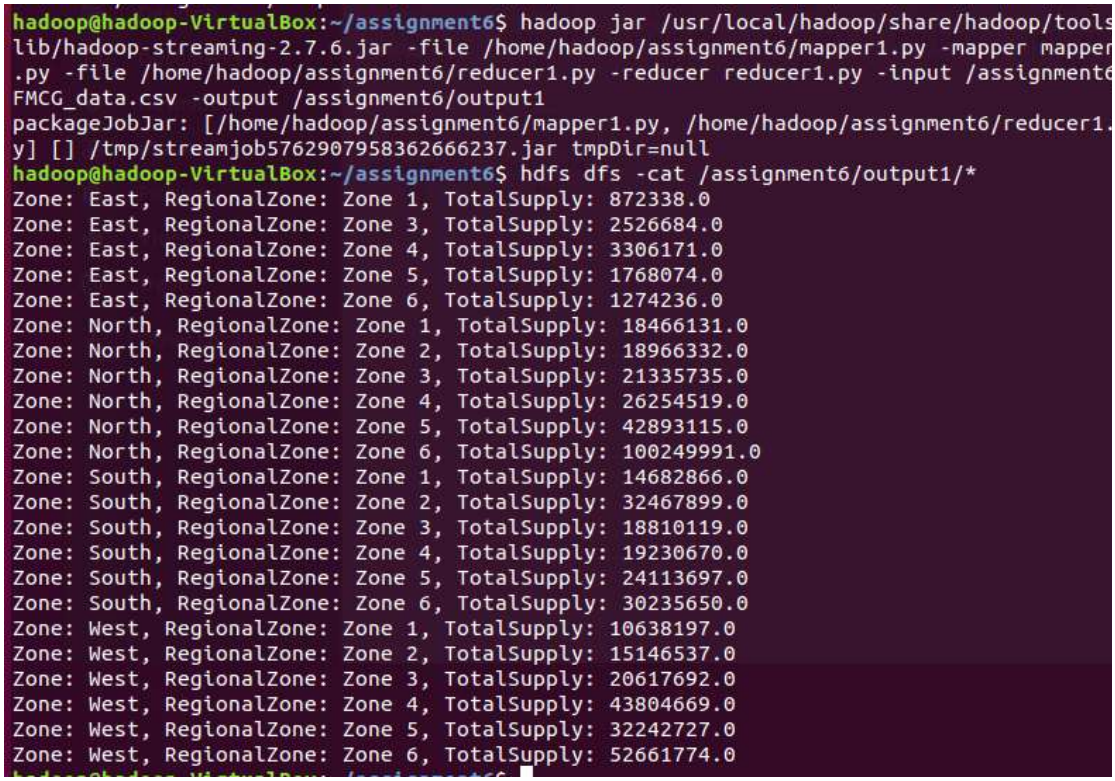
mapper1.py:

```
#!/usr/bin/python3
"""mapper1.py"""

import sys

for line in sys.stdin:
    line = line.strip()
    if line:
        columns = line.split(',')
        if columns:
            zone = columns[4].strip()
            wh_regional_zone = columns[5].strip()
            product_wg_ton = columns[-1].strip()
            print('%s,%s,%s' % (zone, wh_regional_zone, product_wg_ton))
```

reducer1.py:

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

import sys

data_dict = {}
#count_dict = {}
for line in sys.stdin:
    line = line.strip()
    zone, WH_regional_zone, product_wg_ton = line.split(',', 2)

    try:
        product_wg_ton = float(product_wg_ton)
    except ValueError:
        continue

    key = (zone, WH_regional_zone)
    if key in data_dict:
        data_dict[key] += product_wg_ton
#       count_dict[key] += 1
    else:
        data_dict[key] = product_wg_ton
#       count_dict[key] = 1
```

```
#for key in data_dict.keys():
#    data_dict[key] = data_dict[key]/count_dict[key]

for key, value in data_dict.items():
   print(f'Zone: {key[0]}, RegionalZone: {key[1]}, TotalSupply: {value}')
#    print("%s,%s" % (key, value))
```



Task 2: Warehouse Refill Frequency Correlation

Objective: Determine the correlation between warehouse capacity and refill frequency.

mapper2.py:

```
#!/usr/bin/python3
"""mapper2.py"""

import sys

for line in sys.stdin:
   line = line.strip()
   if line:
      columns = line.split(',')
      if columns:
         wh_capacity_size = columns[3].strip()
```

```
        num_refill_req_l3m = columns[6].strip()
        print('%s,%s' % (wh_capacity_size, num_refill_req_l3m))

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

import sys
import numpy as np

data_dict = {}
count_dict = {}
for line in sys.stdin:
   line = line.strip()
   wh_capacity_size, num_refill_req = line.split(',', 1)

   try:
      num_refill_req = int(num_refill_req)
   except ValueError:
      continue

   wh_capacity_value = 1 if wh_capacity_size=='Small' else 2 if wh_capacity_size=='Mid' else
3 if wh_capacity_size=='Largel' else 4
#    wh_capacity_value_list.append(wh_capacity_value)
#    num_refill_req_list.append(num_refill_req)
   if wh_capacity_value in data_dict:
      data_dict[wh_capacity_value] += num_refill_req
      count_dict[wh_capacity_value] += 1
   else:
      data_dict[wh_capacity_value] = num_refill_req
      count_dict[wh_capacity_value] = 1


for key in data_dict.keys():
   data_dict[key] = data_dict[key]/count_dict[key]

wh_capacity_value_list = []
num_refill_req_list = []

correlation_value = np.corrcoef(list(data_dict.keys()), list(data_dict.values()))[0,1]
print(f"Correelation between warehouse capacity and number of refills is:
{round(correlation_value, 5)}")
```

```
hadoop@hadoop-VirtualBox:~/assignment6$ hadoop jar /usr/local/hadoop/share/hadoop/too
lib/hadoop-streaming-2.7.6.jar -file /home/hadoop/assignment6/mapper2.py -mapper mapp
.py -file /home/hadoop/assignment6/reducer2.py -reducer reducer2.py -input /assignmen
FMCG_data.csv -output /assignment6/output2
packageJobJar: [/home/hadoop/assignment6/mapper2.py, /home/hadoop/assignment6/reducer
y] [] /tmp/streamjob7512408383400070 30.jar tmpDir=null
hadoop@hadoop-VirtualBox:~/assignment6$ hdfs dfs -cat /assignment6/output2/*
Correelation between warehouse capacity and number of refills is: -0.73499
hadoop@hadoop-VirtualBox:~/assignment6$
```

Insight: Correlation coefficient between warehouse capacity and average number of refill requests is *0.73499*, thus there is a positive correlation

Task 3. Transport Issue Impact Analysis

Objective: Analyse the impact of transport issues on warehouse supply efficiency.

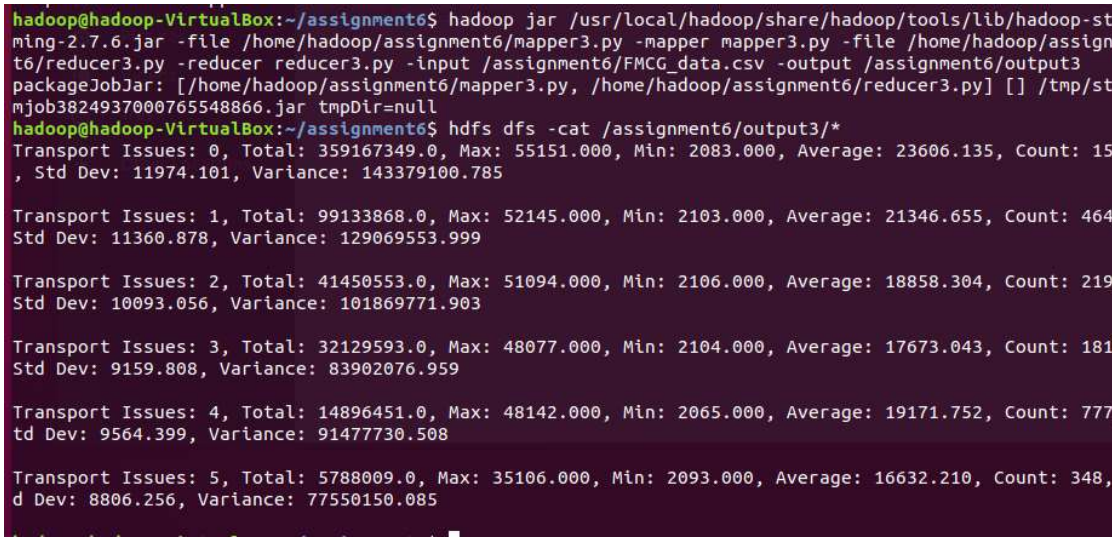
mapper3.py:

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

import sys

for line in sys.stdin:
    line = line.strip()
    if line:
        columns = line.split(',')
        if columns:
            transport_issue_l1y = columns[7].strip()
            product_wg_ton = columns[23].strip()
            print('%s,%s' % (transport_issue_l1y, product_wg_ton))
```

recucer3.py:

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

import sys
import numpy as np

data_dict = {}
##count_dict = {}
for line in sys.stdin:
    line = line.strip()
    transport_issues, product_wg_ton = line.split(',', 1)

    try:
        transport_issues = int(transport_issues)
        product_wg_ton = float(product_wg_ton)
    except ValueError:
        continue

    if transport_issues in data_dict:
        data_dict[transport_issues] += [product_wg_ton]
```

```
##       count_dict[transport_issues] += 1
   else:
      data_dict[transport_issues] = [product_wg_ton]
##       count_dict[transport_issues] = 1

#for key in data_dict.keys():
#    data_dict[key] = round(data_dict[key]/count_dict[key], 3)

data_dict = sorted(data_dict.items(), key=lambda x:x[0])
for (key, value) in data_dict:
   value = np.array(value)
   total = np.sum(value)
   max_value = np.max(value)
   min_value = np.min(value)
   average = np.mean(value)
   count = len(value)
   std_dev = np.std(value)
   variance = np.var(value)
   print(f"Transport Issues: {key}, Total: {total}, Max: {max_value:.3f}, Min: {min_value:.3f},
Average: {average:.3f}, Count: {count}, Std Dev: {std_dev:.3f}, Variance: {variance:.3f}\n")
```



Insight: Comparing Transport Issues with average product weight supplied, we see there is an *inverse relation* - ie, as the *number of transport issues increase*, *average weight of products supplied decreases*.

Task 4. Storage Issue Analysis

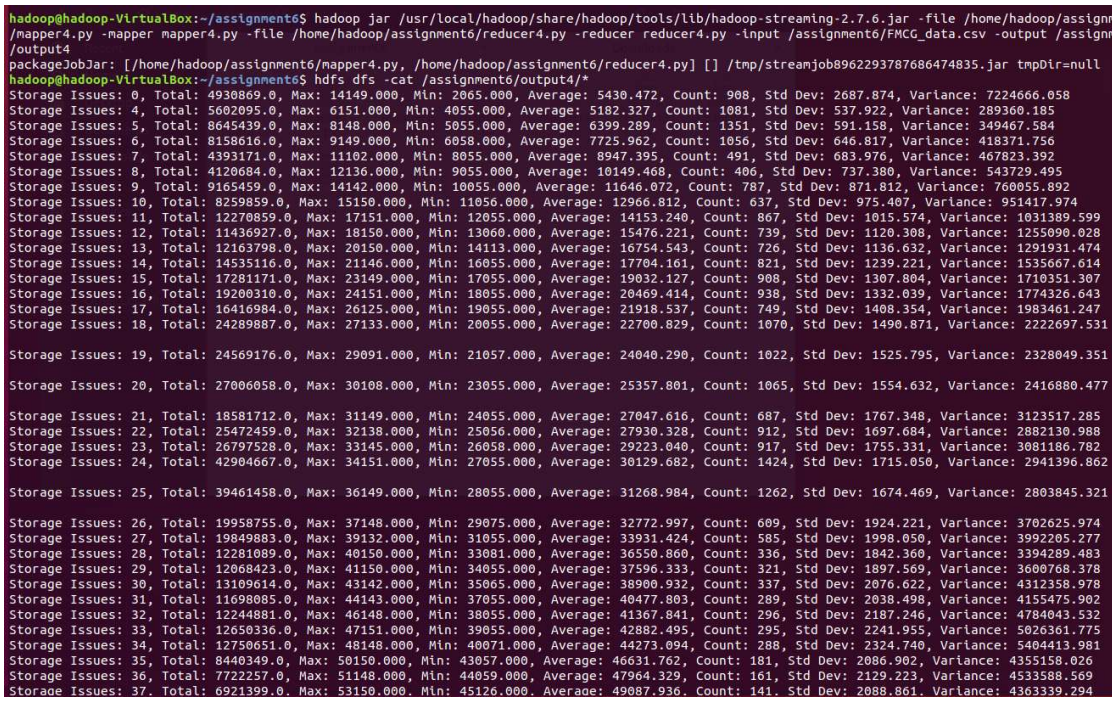
Objective: Evaluate the impact of storage issues on warehouse performance.

mapper4.py:

```
#!/usr/bin/python3
"""mapper4.py"""

import sys
```

```
for line in sys.stdin:
   line = line.strip()
   if line:
      columns = line.split(',')
      if columns:
         storage_issues_reported = columns[18].strip()
         product_wg_ton = columns[23].strip()
         print('%s,%s' % (storage_issues_reported, product_wg_ton))


reducer4.py:
#!/usr/bin/python3
"""reducer4.py"""

import sys
import numpy as np

data_dict = {}
count_dict = {}
for line in sys.stdin:
   line = line.strip()
   storage_issues_reported, product_wg_ton = line.split(',', 1)

   try:
      storage_issues_reported = int(storage_issues_reported)
      product_wg_ton = float(product_wg_ton)
   except ValueError:
      continue

   if storage_issues_reported in data_dict:
      data_dict[storage_issues_reported] += [product_wg_ton]
      count_dict[storage_issues_reported] += 1
   else:
      data_dict[storage_issues_reported] = [product_wg_ton]
      count_dict[storage_issues_reported] = 1

#for key in data_dict.keys():
#    data_dict[key] = round(data_dict[key]/count_dict[key], 3)
#
#for key, value in data_dict.items():
#    print("%s,%s" % (key, value))
data_dict = sorted(data_dict.items(), key=lambda x:x[0])
for (key, value) in data_dict:
   value = np.array(value)
   total = np.sum(value)
   max_value = np.max(value)
   min_value = np.min(value)
   average = np.mean(value)
   count = len(value)
   std_dev = np.std(value)
```

```
    variance = np.var(value)
    print(f"Storage Issues: {key}, Total: {total}, Max: {max_value:.3f}, Min: {min_value:.3f},
Average: {average:.3f}, Count: {count}, Std Dev: {std_dev:.3f}, Variance: {variance:.3f}")
```



Insight: Comparing Storage Issues with product weight supplied, we see that even when there is an increase in storage issues, the warehouses have higher average and total product weight supplied. Thus, there is *a positive coreelation between Storage Issue and Total Product Weight* [also with Average Product Weight].
Also, maximum and minimum product weight supplied is directly proportional to storage issues.
This suggests that warehouses that ships more products will have more storage issues compared to warehouses that ships less products.