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.

```
#!/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:
            try:
                zone = columns[4].strip()
                reg_zone = columns[5].strip()
                prod_wg = columns[-1].strip()
            except:
                continue
            print(f"{zone},{reg_zone},{prod_wg}"
```

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

current_zone = None
current_reg_zone = None
current_wg_sum = 0
current_wg_min = float('inf')
current_wg_max = float('-inf')
product_count = 0

records = []

# Input comes from STDIN
for line in sys.stdin:
    line = line.strip()
    # Parse the input we got from mapper.py
    # Fields are separated by ','
    zone, reg_zone, prod_wg = line.split(",")
    # This IF-switch only works because Hadoop sorts map output
    # by key (here key is word) before it is passed to the reducer
    try:
        prod_wg = int(prod_wg)
    except ValueError:
        continue

    if current_zone == zone and current_reg_zone == reg_zone:
        current_wg_sum += prod_wg
        current_wg_min = min(current_wg_min, prod_wg)
        current_wg_max = max(current_wg_max, prod_wg)
        product_count += 1
    else:
        if current_zone and current_reg_zone:
            avg_wg = current_wg_sum / product_count
```

```
            records.append((current_zone, current_reg_zone, current_wg_sum,
avg_wg, current_wg_min, current_wg_max, product_count))

        current_zone = zone
        current_reg_zone = reg_zone
        current_wg_sum = prod_wg
        current_wg_min = prod_wg
        current_wg_max = prod_wg
        product_count = 1

# Output the last record if needed
if current_zone and current_reg_zone:
    avg_wg = current_wg_sum / product_count
    records.append((current_zone, current_reg_zone, current_wg_sum,
avg_wg, current_wg_min, current_wg_max, product_count))

# Sort the records by zone (or any other metric you want)
records.sort(key=lambda x: x[3], reverse=True)

# Print header labels
print('{:<15} {:<15} {:<15} {:<15} {:<10} {:<10} {:<10}'.format(
    'Zone', 'Regional Zone', 'Total Weight', 'Average Weight', 'Min Weight',
'Max Weight', 'Count'
))

# Print the results
for record in records:
    print('{:<15} {:<15} {:<15} {:<15.2f} {:<10} {:<10} {:<10}'.format(
        record[0],
        record[1],
        record[2],
        record[3],
        record[4],
        record[5],
        record[6]
```

))

```
hadoop@hadoop-VirtualBox:~$ sudo chmod +x mapper1.py
hadoop@hadoop-VirtualBox:~$ sudo chmod +x reducer1.py
hadoop@hadoop-VirtualBox:~$ hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.7.6.jar \
>     -files mapper1.py,reducer1.py \
>     -mapper mapper1.py \
>     -reducer reducer1.py \
>     -input /FMCG \
>     -output /FMCG/output

24/09/06 20:01:24 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using
 builtin-java classes where applicable
24/09/06 20:01:25 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-
id
24/09/06 20:01:25 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
24/09/06 20:01:25 INFO jvm.JvmMetrics: Cannot initialize JVM Metrics with processName=JobTracker, sessionId=
- already initialized
24/09/06 20:01:26 INFO mapred.FileInputFormat: Total input paths to process : 1
```

```
hadoop@hadoop-VirtualBox:~$ hdfs dfs -cat /FMCG/result1/*
24/09/07 06:36:06 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform.
 builtin-java classes where applicable
Zone            Regional Zone   Total Weight    Average Weight  Min Weight Max Weight Count
East            Zone 5          1768074         23892.89        4142       52095      74
East            Zone 4          3306171         23282.89        4086       51105      142
West            Zone 2          15146537        22776.75        2109       55057      665
East            Zone 3          2526684         22559.68        4132       50145      112
South           Zone 2          32467899        22484.69        2065       54150      1444
North           Zone 5          42893115        22375.13        2140       55144      1917
West            Zone 3          20617692        22337.69        3057       54144      923
West            Zone 4          43804669        22281.11        2118       55150      1966
North           Zone 2          18966332        22208.82        3055       54105      854
North           Zone 3          21335735        22201.60        3056       53067      961
North           Zone 6          100249991       22184.11        2093       55095      4519
South           Zone 6          30235650        22166.90        3059       55078      1364
North           Zone 4          26254519        22137.03        3055       54133      1186
East            Zone 6          1274236         21969.59        5104       48084      58
West            Zone 6          52661774        21960.71        3057       55067      2398
North           Zone 1          18466131        21957.35        3064       53133      841
South           Zone 4          19230670        21803.48        3058       55066      882
South           Zone 5          24113697        21782.92        2104       55112      1107
```


Task 2: Warehouse Refill Frequency Correlation

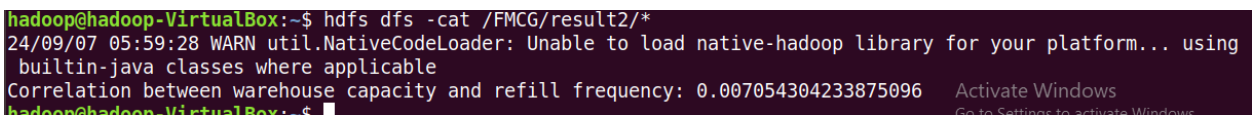
Objective: Determine the correlation between warehouse capacity and refill frequency.
Required Fields: WH_capacity_size, num_refill_req_l3m
Description:
Map: Extract the number of refill requests (num_refill_req_l3m) and warehouse capacity size
(WH_capacity_size) for each warehouse. (For each warehouse, 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
# 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 = columns[3].strip()
            frequency = columns[6].strip()
            print('%s,%s' % (capacity, frequency))

#!/usr/bin/python3
"""reducer2.py"""
import sys
import numpy as np

capacity = 0
refill = 0

warehouse_capacity = []
refill_freq = []
for line in sys.stdin:
    line = line.strip()
    capacity, refill = line.split(",")
    try:
        if capacity == 'Small':
            warehouse_capacity.append(0)
        elif capacity == 'Mid':
```

```
            warehouse_capacity.append(1)
        elif capacity == 'Large':
            warehouse_capacity.append(2)
        refill = int(refill)
        refill_freq.append(refill)
    except ValueError:
        continue
```

```
capacity_array = np.array(warehouse_capacity)
refill_frequency = np.array(refill_freq)
correlation = np.corrcoef(capacity_array,refill_frequency)[0][1]
print(f"Correlation between warehouse capacity and refill frequency: {correlation}")
```



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.

```
#!/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:
            try:
                issue = columns[7].strip()
                prod_wg = columns[-1].strip()
            except:
                continue
            print(f"{issue},{prod_wg}")
```

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

current_issue = None
current_wg_sum = 0
current_wg_min = float('inf')
current_wg_max = float('-inf')
product_count = 0

records = []

# Input comes from STDIN
for line in sys.stdin:
    line = line.strip()
    # Parse the input we got from mapper.py
    # Fields are separated by ','
```

```
    issue, prod_wg = line.split(",")
    # This IF-switch only works because Hadoop sorts map output
    # by key (here key is issue) before it is passed to the reducer
    try:
        prod_wg = int(prod_wg)
    except ValueError:
        continue

    if current_issue == issue:
        current_wg_sum += prod_wg
        current_wg_min = min(current_wg_min, prod_wg)
        current_wg_max = max(current_wg_max, prod_wg)
        product_count += 1
    else:
        if current_issue:
            avg_wg = current_wg_sum / product_count
            records.append((
                current_issue,
                current_wg_sum,
                avg_wg,
                current_wg_min,
                current_wg_max,
                product_count
            ))
        current_issue = issue
        current_wg_sum = prod_wg
        current_wg_min = prod_wg
        current_wg_max = prod_wg
        product_count = 1

# Output the last record if needed
if current_issue:
    avg_wg = current_wg_sum / product_count
    records.append((
        current_issue,
```

```
        current_wg_sum,
        avg_wg,
        current_wg_min,
        current_wg_max,
        product_count
    ))

# Sort the records by average weight (or any other metric you want)
records.sort(key=lambda x: x[1], reverse=True)

# Print header labels
print('{:<15} {:<15} {:<15} {:<15} {:<10} {:<10}'.format(
    'Issue', 'Total Weight', 'Average Weight', 'Min Weight', 'Max Weight',
'Count'
))

# Print the results
for record in records:
    print('{:<15} {:<15} {:<15.2f} {:<15} {:<10} {:<10}'.format(
        record[0],
        record[1],
        record[2],
        record[3],
        record[4],
        record[5]
    ))
```

```
hadoop@hadoop-VirtualBox:~$ hdfs dfs -cat /FMCG/result3/*
24/09/07 06:03:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using
 builtin-java classes where applicable
Issue           Total Weight    Average Weight  Min Weight      Max Weight Count
0               359167349       23606.14        2083            55151      15215
1               99133868        21346.66        2103            52145      4644
2               41450553        18858.30        2106            51094      2198
3               32129593        17673.04        2104            48077      1818
4               14896451        19171.75        2065            48142      777
5               5788009         16632.21        2093            35106      348
```


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.

```
#!/usr/bin/python3
"""mapper4.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:
            try:
                issue = columns[-6].strip()
                prod_wg = columns[-1].strip()
            except:
                continue
            print(f"{issue},{prod_wg}")
```

```
#!/usr/bin/python3
"""reducer4.py"""
import sys
```

```
current_issue = None
current_wg_sum = 0
current_wg_min = float('inf')
current_wg_max = float('-inf')
product_count = 0

records = []

# Input comes from STDIN
for line in sys.stdin:
    line = line.strip()
    # Parse the input we got from mapper.py
    # Fields are separated by ','
    issue, prod_wg = line.split(",")
    # This IF-switch only works because Hadoop sorts map output
    # by key (here key is issue) before it is passed to the reducer
    try:
        prod_wg = int(prod_wg)
    except ValueError:
        continue

    if current_issue == issue:
        current_wg_sum += prod_wg
        current_wg_min = min(current_wg_min, prod_wg)
        current_wg_max = max(current_wg_max, prod_wg)
        product_count += 1
    else:
        if current_issue:
            avg_wg = current_wg_sum / product_count
            records.append((
                current_issue,
                current_wg_sum,
                avg_wg,
                current_wg_min,
```

```
            current_wg_max,
            product_count
        ))
    current_issue = issue
    current_wg_sum = prod_wg
    current_wg_min = prod_wg
    current_wg_max = prod_wg
    product_count = 1

# Output the last record if needed
if current_issue:
    avg_wg = current_wg_sum / product_count
    records.append((
        current_issue,
        current_wg_sum,
        avg_wg,
        current_wg_min,
        current_wg_max,
        product_count
    ))

# Sort the records by average weight (or any other metric you want)
records.sort(key=lambda x: x[2], reverse=True)

# Print header labels
print('{:<15} {:<15} {:<15} {:<15} {:<10} {:<10}'.format(
    'Issue', 'Total Weight', 'Average Weight', 'Min Weight', 'Max Weight',
'Count'
))

# Print the results
for record in records:
    print('{:<15} {:<15} {:<15.2f} {:<15} {:<10} {:<10}'.format(
        record[0],
        record[1],
```

```
        record[2],
        record[3],
        record[4],
        record[5]
    ))
```

```
hadoop@hadoop-VirtualBox:~$ hdfs dfs -cat /FMCG/result4/*
24/09/07 06:28:02 WARN util.NativeCodeLoader: Unable to load native-hadoop library f
 builtin-java classes where applicable
Issue           Total Weight    Average Weight  Min Weight      Max Weight Count
39              8029627         51471.97        48055           55151      156
38              9176172         50697.08        47056           54150      181
37              6921399         49087.94        45126           53150      141
36              7722257         47964.33        44059           51148      161
35              8440349         46631.76        43057           50150      181
34              12750651        44273.09        40071           48148      288
33              12650336        42882.49        39055           47151      295
32              12244881        41367.84        38055           46148      296
31              11698085        40477.80        37055           44143      289
30              13109614        38900.93        35065           43142      337
29              12068423        37596.33        34055           41150      321
28              12281089        36550.86        33081           40150      336
27              19849883        33931.42        31055           39132      585
26              19958755        32773.00        29075           37148      609
25              39461458        31268.98        28055           36149      1262
24              42904667        30129.68        27055           34151      1424
23              26797528        29223.04        26058           33145      917
```