**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)}")



**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.