Muhammad Shas K T

UID: 281775

MapReduce Assignment

Task 1: Demand-Supply Mismatch Analysis

mapper.py

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

import sys
import csv

for row in csv.reader(sys.stdin):
    # Emit zone, regional zone, and product weight shipped
    print("%s\t%s\t%s" % (row[4], row[5], row[23]))
```

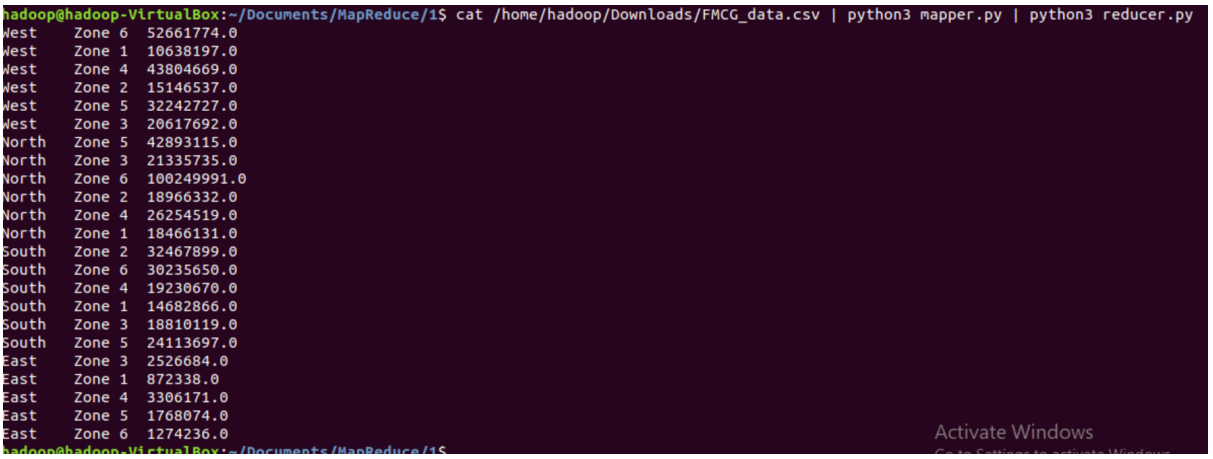
reducer.py

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

import sys

# Dictionary to store aggregated product weights by zone and regional zone
zone_regional_dict = {}

for line in sys.stdin:
    # Split input line into zone, regional zone, and product shipped
```

```
    zone, wh_regional_zone, product_shipped = line.strip().split("\t")
    try:
        product_shipped = float(product_shipped)
    except ValueError:
        continue

    # Update dictionary with aggregated product weights
    if zone in zone_regional_dict:
        if wh_regional_zone in zone_regional_dict[zone]:
            zone_regional_dict[zone][wh_regional_zone] += product_shipped
        else:
            zone_regional_dict[zone][wh_regional_zone] = product_shipped
    else:
        zone_regional_dict[zone] = {wh_regional_zone: product_shipped}

# Output aggregated results
for zone in zone_regional_dict:
    for regional in zone_regional_dict[zone]:
        print("%s\t%s\t%s" % (zone, regional, zone_regional_dict[zone][regional]))
```

```
cat /home/hadoop/Downloads/FMCG_data.csv | python3 mapper.py
```

cat /home/hadoop/Downloads/FMCG_data.csv | python3 mapper.py | python3 reducer.py



Task 2: Warehouse Refill Frequency Correlation

mapper.py

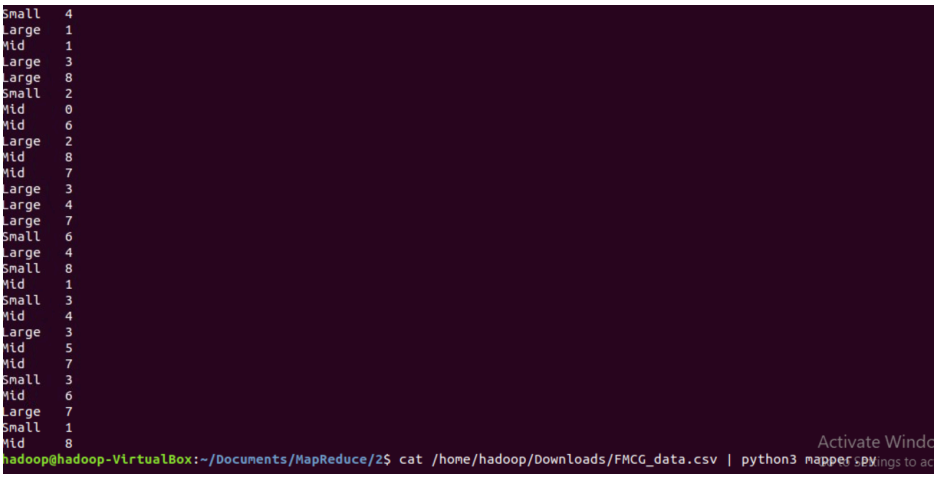
```
#!/usr/bin/python3

"""mapper.py"""


import sys

import csv


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

    # Emit warehouse capacity size and refill requests

    print("%s\t%s" % (row[3], row[6]))
```

reducer.py

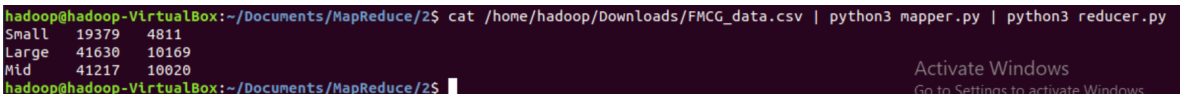
```
#!/usr/bin/python3

"""reducer.py"""


import sys
```

```
# Dictionary to store warehouse capacity and list of refill requests
warehouse_dict = {}

for line in sys.stdin:
    # Split input line into capacity size and refill requests
    capacity, refill = line.strip().split("\t")
    try:
        refill = int(refill)
    except ValueError:
        continue

    # Update dictionary with capacity and refill data
    if capacity in warehouse_dict:
        warehouse_dict[capacity].append(refill)
    else:
        warehouse_dict[capacity] = [refill]

# Output aggregated results (total refill requests and count)
for warehouse in warehouse_dict:
    print("%s\t%s\t%s" % (warehouse, sum(warehouse_dict[warehouse]),
len(warehouse_dict[warehouse])))
```

```
cat /home/hadoop/Downloads/FMCG_data.csv | python3 mapper.py
```



cat /home/hadoop/Downloads/FMCG_data.csv | python3 mapper.py | python3 reducer.py



Task 3. Transport Issue Impact Analysis

mapper.py

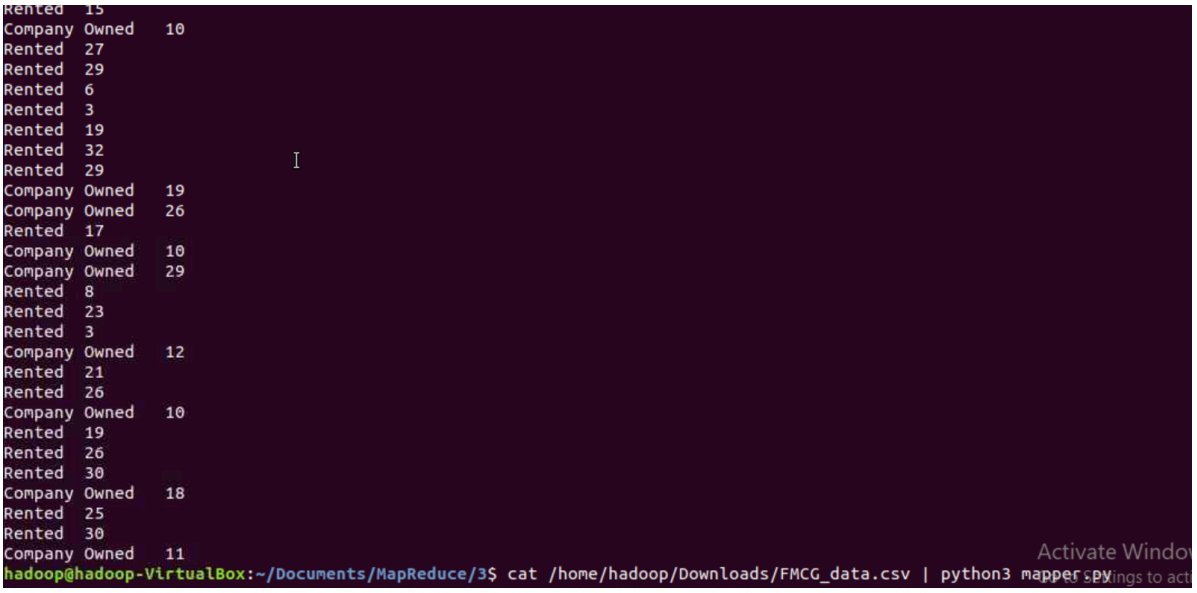
```
#!/usr/bin/python3

"""mapper.py"""


import sys

import csv


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

    # Emit transport issue status and product weight shipped

    print("%s\t%s" % (row[10], row[22]))
```

reducer.py

```
#!/usr/bin/python3

"""reducer.py"""


import sys


# Dictionary to store aggregated product weights by transport issue status

transport_dict = {}


for line in sys.stdin:
```

```
    # Split input line into transport issue status and product weight
    transport, weight = line.strip().split("\t")
    try:
        weight = float(weight)
    except ValueError:
        continue

    # Update dictionary with aggregated product weights
    if transport in transport_dict:
        transport_dict[transport] += weight
    else:
        transport_dict[transport] = weight

# Output aggregated results
for issue_status in transport_dict:
    print("%s\t%s" % (issue_status, transport_dict[issue_status]))
```

```
cat /home/hadoop/Downloads/FMCG_data.csv | python3 mapper.py
```



```
cat /home/hadoop/Downloads/FMCG_data.csv | python3 mapper.py | python3 reducer.py
```

Task 4. Storage Issue Analysis

mapper.py

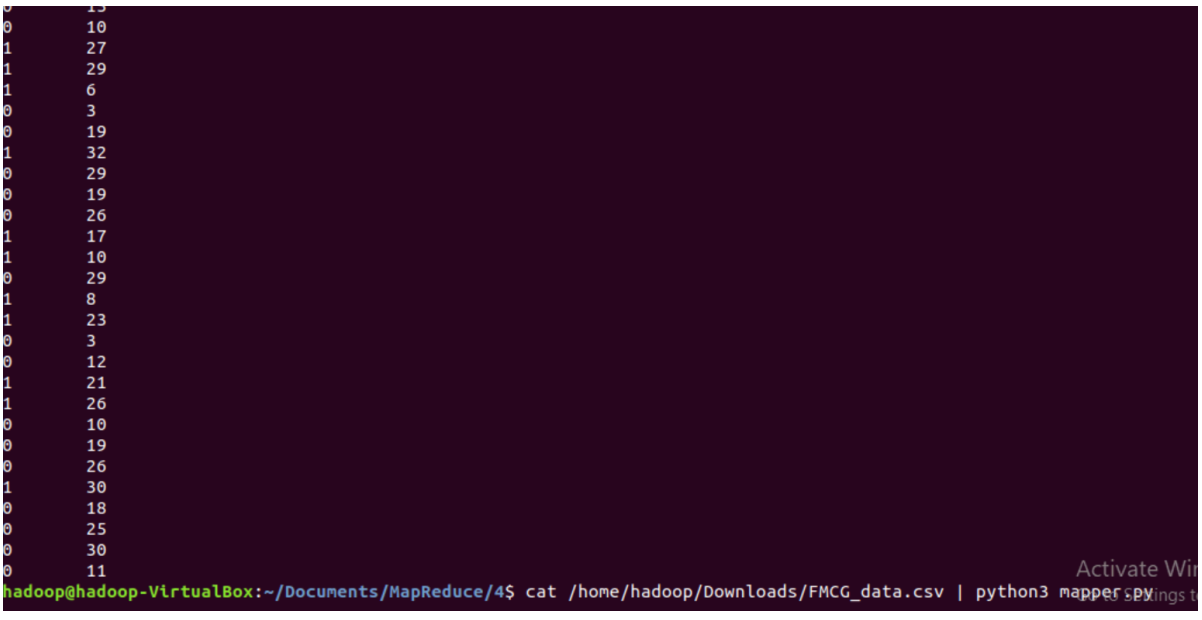
```
#!/usr/bin/python3

"""mapper.py"""


import sys

import csv


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

    # Emit storage issue status and product weight shipped

    print("%s\t%s" % (row[19], row[22]))
```

reducer.py

```
#!/usr/bin/python3

"""reducer.py"""


import sys


# Dictionary to store aggregated product weights by storage issue status

storage_dict = {}


for line in sys.stdin:
```

```
    # Split input line into storage issue status and product weight
    storage, weight = line.strip().split("\t")
    try:
        weight = float(weight)
    except ValueError:
        continue

    # Update dictionary with list of weights
    if storage in storage_dict:
        storage_dict[storage].append(weight)
    else:
        storage_dict[storage] = [weight]

# Output aggregated results (total weight and average weight per storage issue status)
for issue_status in storage_dict:
    total_weight = sum(storage_dict[issue_status])
    average_weight = total_weight / len(storage_dict[issue_status])
    print("%s\t%s\t%s" % (issue_status, total_weight, average_weight))
```

```
cat /home/hadoop/Downloads/FMCG_data.csv | python3 mapper.py
```



cat /home/hadoop/Downloads/FMCG_data.csv | python3 mapper.py | python3 reducer.py

```
hadoop@hadoop-VirtualBox:~/Documents/MapReduce/4$ cat /home/hadoop/Downloads/FMCG_data.csv | python3 mapper.py | python3 reducer.py
0       327281.0        18.789815133769665
1       143026.0        18.863888156159323
hadoop@hadoop-VirtualBox:~/Documents/MapReduce/4$
```