Big Data Management

Hadoop Map Reduce | Ubuntu Server | Data Transformations

This project performs map reduce data transformation operations on Irish Traffic Counter data. Irish Road Network has installed sensors on specific location that collect vehicle data including location, type, speed, weight and more. This data is stored in atomic form.

Dataset: [Traffic Dataset Ireland]

Downloading Data Set using 'wget' followed by the download link.

```
ali@bdm:~/assignment$ wget https://data.tii.ie/Datasets/TrafficCountData/2021/01/15/per-vehicle-reco
rds-2021-01-15.csv
--2022-10-24 15:26:49--  https://data.tii.ie/Datasets/TrafficCountData/2021/01/15/per-vehicle-record
s-2021-01-15.csv
Resolving data.tii.ie (data.tii.ie)... 13.224.68.105, 13.224.68.79, 13.224.68.119, ...
Connecting to data.tii.ie (data.tii.ie)|13.224.68.105|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 355884863 (339M) [binary/octet-stream]
Saving to: 'per-vehicle-records-2021-01-15.csv'

-records-2021-01-15.csv    5%[=>                                   ]  20.22M   703KB/s    eta 8m 26s
```

Create directory for project's assignment

```
ali@bdm:~/assignment$ hdfs dfs -mkdir -p /user/ali/assignment
ali@bdm:~/assignment$ hdfs dfs -ls /user/ali
Found 2 items
drwxr-xr-x   - ali supergroup          0 2022-10-24 15:50 /user/ali/assignment
drwxr-xr-x   - ali supergroup          0 2022-10-23 17:54 /user/ali/wordcount
ali@bdm:~/assignment$
```

Create directory for input. This is where the input data comes and gets stored.

```
ali@bdm:~/assignment$ hdfs dfs -mkdir -p /user/ali/assignment/input
ali@bdm:~/assignment$ hdfs dfs -ls /user/ali/assignment
Found 1 items
drwxr-xr-x   - ali supergroup          0 2022-10-24 15:54 /user/ali/assignment/input
ali@bdm:~/assignment$
```

Upload data on Hadoop DFS

```
ali@bdm:~/assignment$ hadoop fs -put -f ~/assignment/* /user/ali/assignment/input
ali@bdm:~/assignment$ hdfs dfs -ls -R /user/ali/assignment
drwxr-xr-x   - ali supergroup          0 2022-10-24 16:03 /user/ali/assignment/input
-rw-r--r--   1 ali supergroup  355884863 2022-10-24 16:03 /user/ali/assignment/input/per-vehicle-rec
ords-2021-01-15.csv
ali@bdm:~/assignment$
```

Query 1: Lets develop custom mapper and reducer to calculate the usage of Irish road network in terms of percentage grouped by vehicle type.

Mapper: Lines are read from standard input and stored in the variable called lines. The class name from line is printed with each iteration through the loop with a '1' delimited by tab.

```
#!/usr/bin/env python3

import sys

# read all lines from input file
lines = sys.stdin.readlines()

# loop through lines and print vehicle type tab delimited with 1
for line in lines[1:]:
    print('%s\t%s' % (line.split(",")[14], 1))
```

Reducer: reduce the key value pairs to single unique key. Instead of counting the occurrences, we divide the occurrences with total rows and multiply with hundred to get the percentage for each individual key. This calculation is performed directly in the print statements.

```
#!/usr/bin/env python3

import sys

lines = sys.stdin.readlines()
total = len(lines)
current_word = ""
current_count = 0
word = ""

for line in lines:
    line = line.strip()
    word, count = line.split('\t', 1)
    try:
        count = int(count)
    except ValueError:
        continue
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print(f"{current_word}\t{round(current_count/total*100, 2)}%")
        current_count = count
        current_word = word
if current_word == word:
    print(f"{current_word}\t{round(current_count/total*100, 2)}%")
```

Output: The output is as follows. We also have 0.01% missing values for class name column that are empty has represented as " " empty string.

```
ali@bdm:~/bdm$ hdfs dfs -ls -R /user/ali/assignment/output_q2
-rw-r--r--   1 ali supergroup          0 2022-11-02 02:45 /user/ali/assignment/output_q2/_SUCCESS
-rw-r--r--   1 ali supergroup        109 2022-11-02 02:45 /user/ali/assignment/output_q2/part-00000
ali@bdm:~/bdm$ hdfs dfs -cat /user/ali/assignment/output_q2/part-00000
""      0.01%
"BUS"   0.78%
"CAR"   70.24%
"CARAVAN"       0.62%
"HGV_ART"       7.57%
"HGV_RIG"       4.37%
"LGV"   15.84%
"MBIKE" 0.56%
ali@bdm:~/bdm$
```

Query 2: Lets calculate the highest and lowest hourly flows for Cars on M50 (use the counters installed between junction 03 - junction 17).

Mapper:

```
#!/usr/bin/env python3

import sys
# read lines from input
lines = sys.stdin.readlines()

for line in lines[1:]:
    line = line.replace('"','').split(',') #remove "" from lines and split

    # we only need data for 'Car' so check for this value on index 14 of line
    if line[14] == 'CAR':
        # we only need trackers for four junctions so check if index 0 contains those tracker values
        if line[0] == '000000020076' or line[0] == '000000020077' or line[0] == '000000020078' or l$
            #print hour which is index 4 of line
            hour = line[4]
            # just to make it look consistant add 0 if hour length is single digit
            # e.g. 0 will become 00 and 1 will become 01 (helps in sorting)
            if len(hour) < 2:
                hour = '0' + hour
            print(f"{hour}\t1")
```

Reducer:

```
#!/usr/bin/env python3

import sys

word = ''
max_count = 0
max_word = ''
min_count = 10 * 100
min_word = ''
hour_dict = {}

#read lines from input
for line in sys.stdin:
    line = line.strip()
    try:
        hour, count = line.split('\t')
    except ValueError:
        continue
    try:
        count = int(count)
    except ValueError:
        continue
    #if key is not in dictionary than add the key in dictionary
    if hour not in hour_dict:
        hour_dict[hour] = 1
    #otherwise increase the count for that key
    else:
        hour_dict[hour] += count
#loop through dictionary to check for minimum and maximum values
for hour in hour_dict:
    if hour_dict[hour] > max_count:
        max_count = hour_dict[hour]
```

```
#loop through dictionary to check for minimum and maximum values
for hour in hour_dict:
    if hour_dict[hour] > max_count:
        max_count = hour_dict[hour]
        max_word = hour
    if hour_dict[hour] < min_count:
        min_count = hour_dict[hour]
        min_word = hour
#print those minimum and maximum values
print(f"Hour with highest flows for Cars: {max_word}:00 - {max_word}:59\t{max_count}")
print(f"Hour with lowest flows for Cars: {min_word}:00 - {min_word}:59\t{min_count}")
```

Output:

```
ali@bdm:~/bdm$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -input /use
r/ali/assignment/input -output /user/ali/assignment/output_q3 -mapper ~/bdm/q3_mapper.py -reducer ~/
bdm/q3_reducer.py
```

```
ali@bdm:~/bdm$ hdfs dfs -ls -R /user/ali/assignment/output_q3
-rw-r--r--   1 ali supergroup          0 2022-11-02 03:09 /user/ali/assignment/output_q3/_SUCCESS
-rw-r--r--   1 ali supergroup        103 2022-11-02 03:09 /user/ali/assignment/output_q3/part-00000
ali@bdm:~/bdm$ hdfs dfs -cat /user/ali/assignment/output_q3/part-00000
Hour with highest flows for Cars: 16:00 - 16:59 2382
Hour with lowest flows for Cars: 03:00 - 03:59  55
ali@bdm:~/bdm$ _
```

Query 3: Lets calculate average speed of Motorbikes.

Mapper:

```python
#!/usr/bin/env python3

import sys
# read lines and save in variable
lines = sys.stdin.readlines()
# loop through line - ignore first line as header
for line in lines[1:]:
    line = line.replace('"','').split(',')
    # we only need data for motorbikes so we look for 'MBIKE' on index 14
    if line[14] == 'MBIKE':
        # we print the 1 \t 'speed' which is at index 18
        print(f"1\t{line[18]}")
```

Reducer:

```python
#!/usr/bin/env python3

import sys

current_count = 0
current_sum = 0

#calculate sum of speed and total count
for line in sys.stdin:
    line = line.strip()
    total, sum = line.split('\t')
    try:
        total = int(total)
        sum = float(sum)
    except ValueError:
        continue

    current_count += total
    current_sum += sum
average = current_sum/current_count #calculate average
print(f"Average speed of motorbikes: {round(average, 2)}") #print results
```

Output:

```
ali@bdm:~/bdm$ hdfs dfs -ls -R /user/ali/assignment/output_q4
-rw-r--r--   1 ali supergroup          0 2022-11-02 03:21 /user/ali/assignment/output_q4/_SUCCESS
-rw-r--r--   1 ali supergroup         35 2022-11-02 03:21 /user/ali/assignment/output_q4/part-00000
ali@bdm:~/bdm$ hdfs dfs -cat /user/ali/assignment/output_q4/part-00000
Average speed of motorbikes: 57.7
ali@bdm:~/bdm$ _
```

Query 4: Lets calculate the top 10 locations with highest number of counts of HGVs (consider both RIGID + ART).

Mapper:

```
#!/usr/bin/env python3

import sys
#read lines and save in variable
lines = sys.stdin.readlines()

# loop through variable - ignoring first line as header
for line in lines[1:]:
    line = line.replace('"','').split(',')
    #we only need data for HGVs so we look HGV in index 14
    if 'HGV' in line[14]:
        #we only need location data so we print the cosit i.e. index 0
        print(f"{line[0]}\t1")
```

Reducer:

```
#!/usr/bin/env python3

import sys

current_word = ''
current_count = 0
word = ''
# create empty dictionary
loc = {}
for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)
    try:
        count = int(count)
    except ValueError:
        continue
    # create a new key if its not in dictionary
    if word not in loc:
        loc[word] = 1
    # otherwise increase count of key
    else:
        loc[word] += count
# sort dictionary based on values of keys in descending order
sorted_dict = sorted(loc.items(), key=lambda x:x[1], reverse=True)
#print only top 10 results using for loop
print('Location\tHGV_Counts')
print('------------\t----------')
for key, value in sorted_dict[:10]:
    print(f"{key}\t{value}")
```

Output:

```
ali@bdm:~/bdm$ hdfs dfs -ls -R /user/ali/assignment/output_q5
-rw-r--r--   1 ali supergroup          0 2022-11-02 03:33 /user/ali/assignment/output_q5/_SUCCESS
-rw-r--r--   1 ali supergroup        226 2022-11-02 03:33 /user/ali/assignment/output_q5/part-00000
ali@bdm:~/bdm$ hdfs dfs -cat /user/ali/assignment/output_q5/part-00000
Location        HGV_Counts
------------    ----------
000000000997    15657
000000000998    14715
000000001508    7200
000000001502    6329
000000001503    6105
000000001501    5771
000000001500    4477
000000001070    4453
000000001072    4413
000000001071    4290
ali@bdm:~/bdm$
```