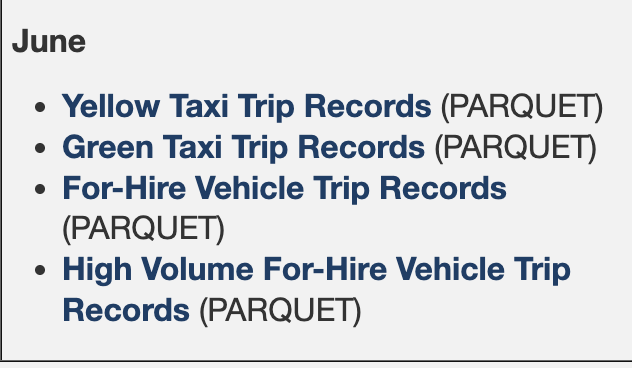
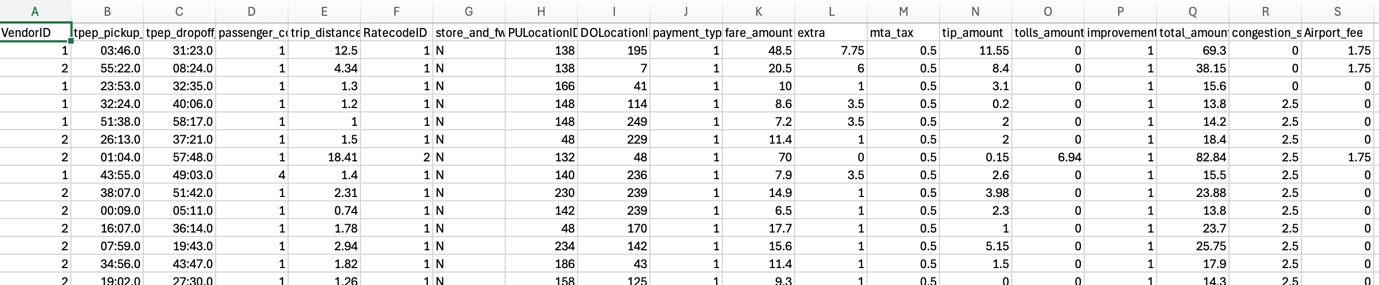
**Taxi Average Fare Calculator**

**SAP ID**: 70180418

**Name:** Muhammad Asghar

**Steps**

1. Download dataset from <https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page>
2. As per the instructions I downloaded June 2024 dataset of Yellow Taxi Trip Records in PARQUET format which I later converted to CSV format.



1. Our record values are in Colum ***trip\_distance (Cloumn Index 4)*** and ***fare\_amount (Column index 10)***
2. Now my Mapper code is below

package com.hadoop;

import java.io.IOException;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.Mapper;

public class TaxiFareMapper extends Mapper<LongWritable, Text, FloatWritable, FloatWritable> {

//my csv file data is in index 4 and 10

// 4th index is distance and 10th index is fare

// 4: trip\_distance,

// 10: fare\_amount,

private static final int DISTANCE\_INDEX = 4; //

private static final int FARE\_INDEX = 10;

private static final int MIN\_EXPECTED\_FIELDS = 11; // to check total number of fields.

@Override

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

String line = value.toString().trim();

if (line.isEmpty()) {

return; // Skip blank lines

}

String[] fields = line.split(",");

// Validate field length

if (fields.length < MIN\_EXPECTED\_FIELDS) {

System.err.println("row is invalid " + line);

return;

}

try {

float tripDistance = Float.parseFloat(fields[DISTANCE\_INDEX]);

float tripFare = Float.parseFloat(fields[FARE\_INDEX]);

// Write only valid (non-zero) distances

if (tripDistance > 0.0f) {

context.write(new FloatWritable(tripDistance), new FloatWritable(tripFare));

}

} catch (NumberFormatException nfe) {

// Log malformed numeric data

System.err.println("number foramt issue " + line);

} catch (Exception e) {

// Catch-all for unexpected errors (e.g., array issues)

System.err.println("all other errors: " + line);

}

}

}

Which simple reads the CSV file and fetches tripDistance and tripFare and emits as key value pair in context.

1. Below is my reducer code.

package com.hadoop;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.Reducer;

public class TaxiFareReducer extends Reducer<FloatWritable, FloatWritable, FloatWritable, FloatWritable> {

public void reducer(FloatWritable key, Iterable<FloatWritable> values, Context context) throws IOException, InterruptedException {

float sum = 0; //sum of fare

int count = 0;

List<FloatWritable> valueList = new ArrayList<>();

for (FloatWritable val : values) {

valueList.add(val);

}

for (int i = 0; i < valueList.size(); i++) {

FloatWritable val = valueList.get(i);

sum += val.get();

count++;

}

if (count > 0) {

context.write(key, new FloatWritable(sum / count));

}

}

}

Reducer code get values from map function as a key value format for each key which is distance it calculates sum of values which are fares.

1. Driver class
2. package com.hadoop;
3. /\*Driver Class fro MapReduce \*/
4. import org.apache.hadoop.conf.Configuration;
5. import org.apache.hadoop.fs.Path;
6. import org.apache.hadoop.io.FloatWritable;
7. import org.apache.hadoop.mapreduce.Job;
8. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
9. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
10. public class TaxiFareDriver {
11. public static void main(String[] args) throws Exception {
12. Configuration conf = new Configuration();
13. Job job = Job.getInstance(conf, "Average Fare per Trip Distance");
14. job.setJarByClass(TaxiFareDriver.class);
15. job.setMapperClass(TaxiFareMapper.class);
16. job.setReducerClass(TaxiFareReducer.class);
17. job.setOutputKeyClass(FloatWritable.class);
18. job.setOutputValueClass(FloatWritable.class);
19. FileInputFormat.addInputPath(job, new Path(args[0])); // Input CSV file
20. FileOutputFormat.setOutputPath(job, new Path(args[1])); // Output path
21. System.exit(job.waitForCompletion(true) ? 0 : 1);
22. }
23. }
24. Now Steps include

|  |  |
| --- | --- |
| Command | Usage |
| javac -classpath `hadoop classpath` -d classes TaxiFareMapper.java TaxiFareReducer.java TaxiFareDriver.java | Compiles classes mapper, reducer and Driver in classes folder |
| jar -cvf TaxiFareJob.jar -C classes/ . | Generates jar file for all the classes |
|  |  |

Now everything is done and now we need to start Hadoop yar and hdfs

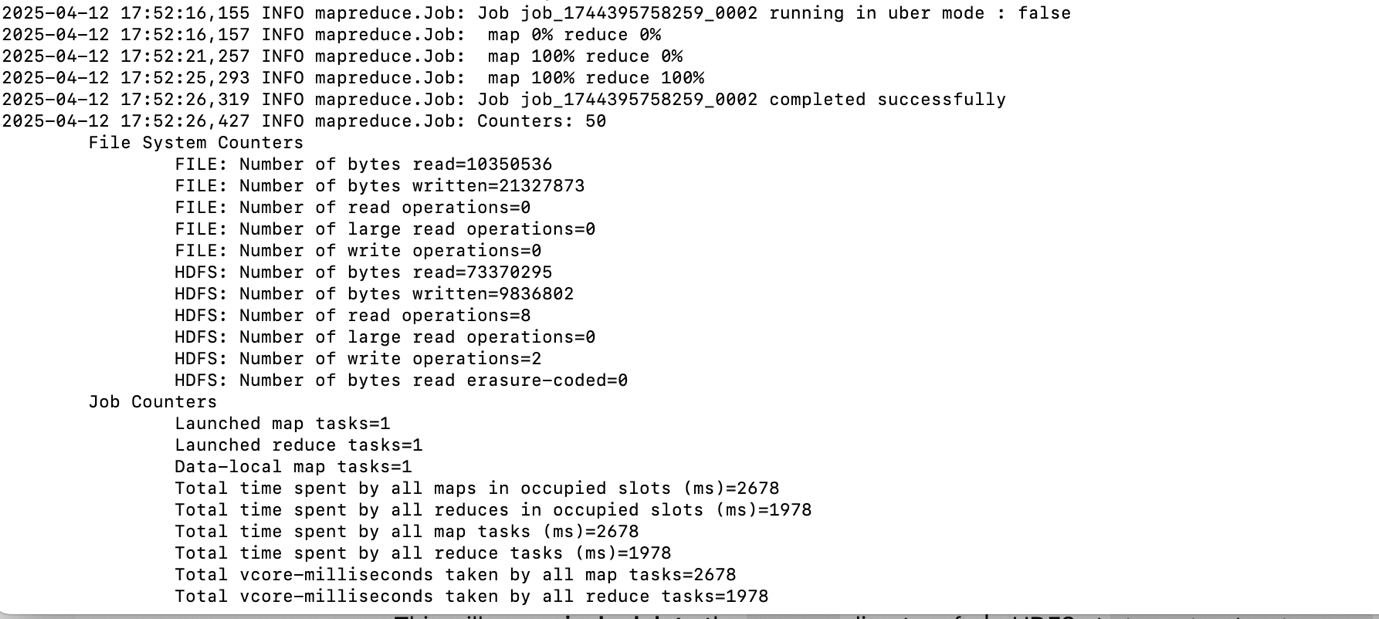
1. start-yarn.sh
2. start-dfs.sh

Create folder as HDFS and copy dataset to HDFS

hdfs dfs -put input/taxi\_data.csv /taxi-input

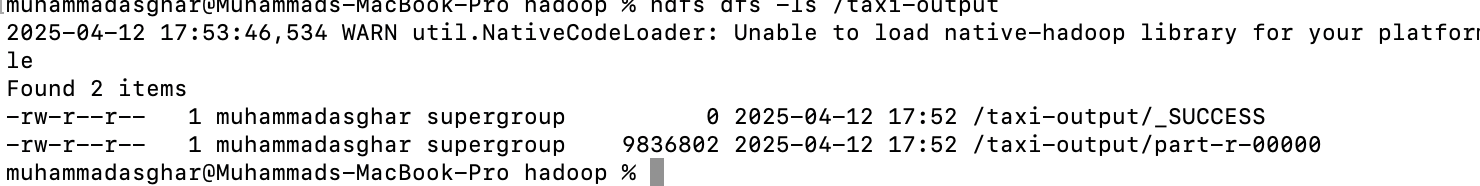
now check if the dataset file exists at hdfs and then run the jar

hadoop jar TaxiFareJob.jar com.hadoop.TaxiFareDriver /taxi-input /taxi-output



Job has completed successfully.

Viewing the output

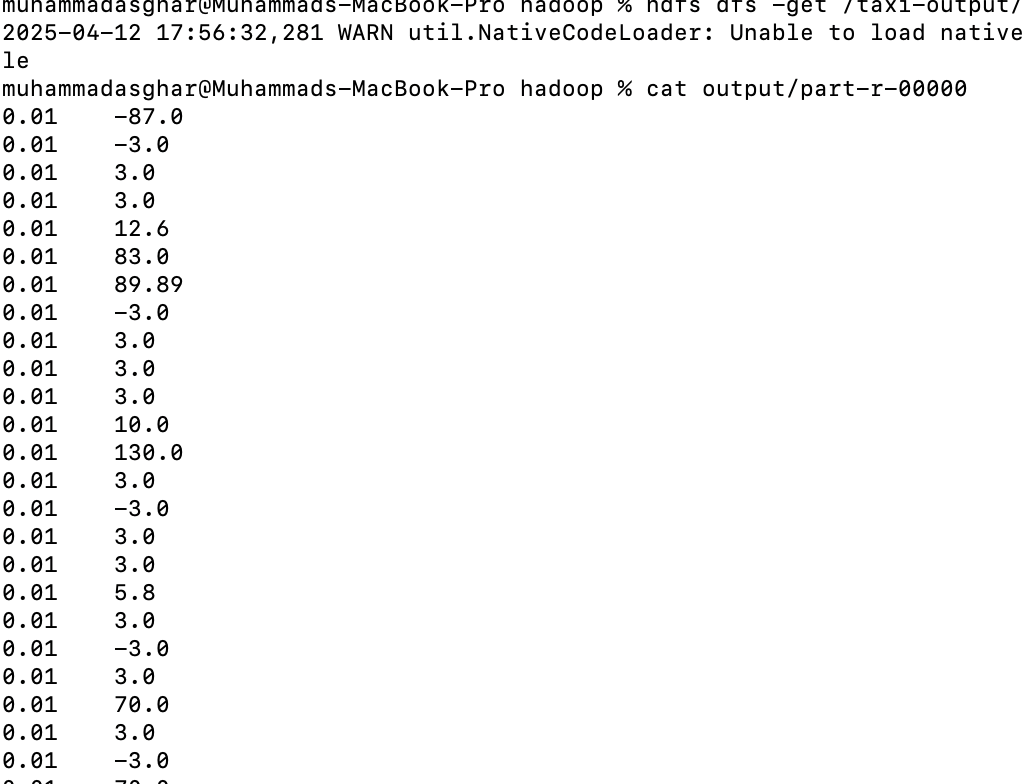


/taxi-output/part-r-00000 is the output file.

Can view this file on hdfs or can copy to local drive

hdfs dfs -get /taxi-output/part-r-00000 output

now view file



In CSV first is distance and the other value is average fare for that distance.