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

# Read line, and split into fields
for line in sys.stdin:
    line = line.strip()
    field = line.split(",")

    try:
        # Capture taxi number (index 1) and distance (index 3), then
print
        if field[1].strip() and field[3].strip():
            taxi_num = int(field[1])
            distance = float(field[3])
            print('%d,%f' % (taxi_num, distance))
    except:
        continue#!/usr/bin/env python3
import sys

# Initiate variables
current_taxi = None
current_distance = 0.0
count = 0
taxi = None

# Read line from mapper
for line in sys.stdin:
    line = line.strip()
    taxi, distance = line.split(',')

    try:
        distance = float(distance)

    # Skip if cannot convert
    except ValueError:
        continue

    if current_taxi == taxi:

        # Count trips and add distance for the same taxi
        count += 1
        current_distance += distance

    else:

        # Print taxi number, trip, and average distance in 2 decimal
format
        if current_taxi:
            avg_distance = current_distance/count
            print('%s,%s,%.2f' % (current_taxi, count, avg_distance))

        current_distance = distance
        current_taxi = taxi
        count = 1

# Print the last line
if current_taxi == taxi:
    avg_distance = current_distance/count
```

```
    print('%s,%s,%.2f' % (current_taxi, count, avg_distance))
#!/bin/bash

hadoop fs -rm -r /input
hadoop fs -rm -r /output/task1

hadoop fs -mkdir /input
hadoop fs -put ./Taxis.txt /input/Taxis.txt
hadoop fs -put ./Trips.txt /input/Trips.txt

hadoop jar ./hadoop-streaming-3.1.4.jar \
-D mapred.reduce.tasks=3 \
-file ./task1-mapper.py \
-mapper ./task1-mapper.py \
-file ./task1-reducer.py \
-reducer ./task1-reducer.py \
-input /input/Trips.txt \
-output /output/task1

hadoop fs -cat /output/task1/*
#!/usr/bin/env python3
import sys

# Read line, and split into fields
for line in sys.stdin:
    line = line.strip()
    field = line.split(',')

    # Check which file the line comes from by the number of columns
    # Then print the result with "-" for missing value

    # This is from Taxis.txt
    if len(field) == 4:
        taxi_num = int(field[0].strip())
        company = int(field[1].strip())
        print('%d,%d,%s' % (taxi_num, company, "-"))

    # This is from Trips.txt
    else:
        taxi_num = int(field[1].strip())
        trip_num = int(field[0].strip())
        print('%d,%s,%d' % (taxi_num, "-", trip_num))
#!/usr/bin/env python3
import sys

# Initiate variables
current_taxi = None
count = 0

# Read line from mapper
for line in sys.stdin:
    line = line.strip()
    taxi_num, company, trip_num = line.split(',')

    # Count lines from Trips.txt
    if current_taxi == taxi_num and trip_num != "-":
        count += 1
```

```
    else:
        # Print company and total trips if from Taxis.txt
        if current_taxi and company != "-":
            print('%s,%s' % (company, count))

        current_taxi = taxi_num
        count = 1
```

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

# Read line, and split into fields
for line in sys.stdin:
    line = line.strip()
    field = line.split(",")
    company = int(field[0].strip())
    count = int(field[1].strip())

    print('%d,%d' % (company, count))
```

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

# Initiate variables
current_company = None
total_count = 0

# Read line from mapper
for line in sys.stdin:
    line = line.strip()
    company, count = line.split(',')

    # If it is the same company, sum up
    if current_company == company:
        total_count = int(total_count)
        count = int(count)
        total_count += count

    else:
        # If it is another company, print the result and move on to set
up a new pair
        if current_company:
            print('%s,%s' % (current_company, total_count))

        # Set both variables to the values in the line
        current_company = company
        total_count = count
```

```
#!/bin/bash

hadoop fs -rm -r /input
hadoop fs -rm -r /task3
hadoop fs -rm -r /output/task3

hadoop fs -mkdir /input
hadoop fs -put ./Taxis.txt /input/Taxis.txt
hadoop fs -put ./Trips.txt /input/Trips.txt
```

```
# Use taxi number and company as keys
# use -k1 as partitioner key which is taxi number
hadoop jar ./hadoop-streaming-3.1.4.jar \
-D stream.num.map.output.key.fields=2 \
-D map.output.key.field.separator=, \
-D mapred.text.key.partitioner.options=-k1,1 \
-D mapred.reduce.tasks=3 \
-file ./task3-1-mapper.py \
-mapper ./task3-1-mapper.py \
-file ./task3-1-reducer.py \
-reducer ./task3-1-reducer.py \
-input /input/Trips.txt \
-input /input/Taxis.txt \
-output /task3 \
-partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner

hadoop fs -getmerge /task3/part-* ./task3-join-output.txt
hadoop fs -put ./task3-join-output.txt /input/task3-join-output.txt
hadoop fs -rm -r /task3

# Use company as a key, and it is also a partition key
# To sort key values in the reducer according to company
hadoop jar ./hadoop-streaming-3.1.4.jar \
-D stream.num.map.output.key.fields=1 \
-D map.output.key.field.separator=, \
-D mapred.text.key.partitioner.options=-k1,1 \
-D mapred.reduce.tasks=3 \
-file ./task3-2-mapper.py \
-mapper ./task3-2-mapper.py \
-file ./task3-2-reducer.py \
-reducer ./task3-2-reducer.py \
-input /input/task3-join-output.txt \
-output /output/task3 \
-partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner

hadoop fs -getmerge /output/task3/part* ./task3-output.txt
hadoop fs -put ./task3-output.txt /output/task3/task3-output.txt
hadoop fs -cat /output/task3/task3-output.txt
```