MapReduce Tasks

1) **Which vendors have the most trips, and what is the total revenue generated by that vendor?**

Execution and output -

```
[hadoop@ip-172-31-76-131 files]$ cat mrtask_a.py
from mrjob.job import MRJob
import csv

class TripRevenue(MRJob):

    def mapper(self, _, line):
        row = list(csv.reader([line]))[0]
        try:
            vendor_id = int(row[1])
            total_amount = float(row[17])
            yield vendor_id, total_amount
        except:
            pass

    def reducer(self, key, values):
        total_revenue = sum(values)
        yield key, total_revenue

    def combiner(self, key, values):
        total_revenue = sum(values)
        yield key, total_revenue

if __name__ == '__main__':
    TripRevenue.run()

[hadoop@ip-172-31-76-131 files]$ python mrtask_a.py yellow_tripdata_2017-05.csv
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/mrtask_a.hadoop.20230507.075028.198057
Running step 1 of 1...
job output is in /tmp/mrtask_a.hadoop.20230507.075028.198057/output
Streaming final output from /tmp/mrtask_a.hadoop.20230507.075028.198057/output...
1       1143.18
2       747.49
Removing temp directory /tmp/mrtask_a.hadoop.20230507.075028.198057...
[hadoop@ip-172-31-76-131 files]$
```

Code - mrtask_a.py

```
from mrjob.job import MRJob
import csv

class TripRevenue(MRJob):
# The mapper function reads each line of the input file, extracts the vendor ID and total amount for each trip.
    def mapper(self, _, line):
        row = list(csv.reader([line]))[0]
        try:
            vendor_id = int(row[1])
            total_amount = float(row[17])
            yield vendor_id, total_amount
        except:
            pass
# The reducer function receives the key-value pairs from the mapper and calculates the total revenue generated by each vendor by summing up the total amounts for all their trips.

    def reducer(self, key, values):
        total_revenue = sum(values)
        yield key, total_revenue
# The combiner function is used to aggregate the intermediate outputs from the mapper

    def combiner(self, key, values):
        total_revenue = sum(values)
        yield key, total_revenue

if __name__ == '__main__':
    TripRevenue.run()
```

2) Which pickup location generates the most revenue?

Execution and output -

```
[hadoop@ip-172-31-73-144 ~]$ cat mrtask_b.py
from functools import reduce
import sys
import csv

def map_reduce(input_file):
    # Read the CSV file
    with open(input_file) as f:
        reader = csv.reader(f)

        # Skip the header row
        next(reader)

        # Map the pickup location ID to the total amount charged
        mapped_values = map(lambda row: (row[8], float(row[17])), reader)

        # Group the mapped values by pickup location ID
        grouped_values = {}
        for pickup_loc, total_amount in mapped_values:
            if pickup_loc in grouped_values:
                grouped_values[pickup_loc].append(total_amount)
            else:
                grouped_values[pickup_loc] = [total_amount]

        # Reduce the grouped values to find the pickup location that generated the most revenue
        reduced_values = reduce(lambda x, y: (x[0], x[1]) if x[1] > y[1] else (y[0], y[1]), grouped_values.items())
        # Print the result
        print(f"Pickup location {reduced_values[0]} generated the most revenue: {reduced_values[1]}")

if __name__ == '__main__':
    # Get the input file name from the command line arguments
    input_file = sys.argv[1]

    # Call the map_reduce function with the input file name
    map_reduce(input_file)

[hadoop@ip-172-31-73-144 ~]$
[hadoop@ip-172-31-73-144 ~]$ python mrtask_b.py yellow_tripdata_2017-06.csv
Pickup location 132 generated the most revenue: [66.06, 70.27, 70.27, 56.56, 35.76]
[hadoop@ip-172-31-73-144 ~]$
```

Code - mrtask_b.py

```
from functools import reduce
import sys
import csv

def map_reduce(input_file):
    # Read the CSV file
    with open(input_file) as f:
        reader = csv.reader(f)

        # Skip the header row
        next(reader)

        # Map the pickup location ID to the total amount charged
        mapped_values = map(lambda row: (row[8], float(row[17])), reader)

        # Group the mapped values by pickup location ID
        grouped_values = {}
        for pickup_loc, total_amount in mapped_values:
            if pickup_loc in grouped_values:
                grouped_values[pickup_loc].append(total_amount)
            else:
                grouped_values[pickup_loc] = [total_amount]
```

```
    # Reduce the grouped values to find the pickup location that generated the most revenue
    reduced_values = reduce(lambda x, y: (x[0], x[1]) if x[1] > y[1] else (y[0], y[1]),
grouped_values.items())
    # Print the result
    print(f"Pickup location {reduced_values[0]} generated the most revenue: {reduced_values[1]}")

if __name__ == '__main__':
  # Get the input file name from the command line arguments
  input_file = sys.argv[1]

  # Call the map_reduce function with the input file name
  map_reduce(input_file)
```

3) **What are the different payment types used by customers and their count? The final results should be in a sorted format.**

Execution and Output –

```
[hadoop@ip-172-31-73-144 ~]$ cat mrtask_c.py
from mrjob.job import MRJob
import csv

class PaymentTypeCount(MRJob):

    def mapper(self, _, line):
        # Skip the header line
        if line.startswith('ID'):
            return

        # Parse the CSV line
        fields = list(csv.reader([line]))[0]

        # Extract the payment type and emit a count of 1
        payment_type = fields[10]
        yield payment_type, 1

    def combiner(self, payment_type, counts):
        # Sum up the counts for each payment type
        yield payment_type, sum(counts)

    def reducer(self, payment_type, counts):
        # Sum up the counts for each payment type
        yield payment_type, sum(counts)

if __name__ == '__main__':
    PaymentTypeCount.run()

[hadoop@ip-172-31-73-144 ~]$ python mrtask_c.py yellow_tripdata_2017-06.csv
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/mrtask_c.hadoop.20230507.164621.142151
Running step 1 of 1...
job output is in /tmp/mrtask_c.hadoop.20230507.164621.142151/output
Streaming final output from /tmp/mrtask_c.hadoop.20230507.164621.142151/output...
"1"     64
"2"     32
"3"     3
Removing temp directory /tmp/mrtask_c.hadoop.20230507.164621.142151...
```

Code - mrtask_c.py

```
from mrjob.job import MRJob
import csv

class PaymentTypeCount(MRJob):

    def mapper(self, _, line):
        # Skip the header line
        if line.startswith('ID'):
            return

        # Parse the CSV line
        fields = list(csv.reader([line]))[0]

        # Extract the payment type and emit a count of 1
        payment_type = fields[10]
        yield payment_type, 1

    def combiner(self, payment_type, counts):
        # Sum up the counts for each payment type
        yield payment_type, sum(counts)

    def reducer(self, payment_type, counts):
        # Sum up the counts for each payment type
        yield payment_type, sum(counts)

if __name__ == '__main__':
    PaymentTypeCount.run()
```

4) **What is the average trip time for different pickup locations**?

```
[hadoop@ip-172-31-70-201 ~]$ cat mrtask_d.py
from mrjob.job import MRJob
from mrjob.step import MRStep
from datetime import datetime
from itertools import groupby
import csv

class TripTime(MRJob):

    def mapper(self, _, line):
        if line.startswith('ID'):
            return
    # Parse the CSV line
        data = next(csv.reader([line]))
        try:
            pickup_loc = int(data[8])
            dropoff_loc = int(data[9])
            pickup_time = datetime.strptime(data[2], '%Y-%m-%d %H:%M:%S')
            dropoff_time = datetime.strptime(data[3], '%Y-%m-%d %H:%M:%S')
            yield pickup_loc, (dropoff_time - pickup_time).seconds / 60
        except:
            pass

    def reducer(self, pickup_loc, trip_times):
        total_trip_time = 0
        count = 0
        for trip_time in trip_times:
            total_trip_time += trip_time
            count += 1
        yield pickup_loc, round(total_trip_time / count, 2)

    def steps(self):
        return [
            MRStep(mapper=self.mapper,
                   reducer=self.reducer)
        ]
if __name__ == '__main__':
    TripTime.run()
```

Output –

```
[hadoop@ip-172-31-70-201 ~]$ python mrtask_d.py yellow_tripdata_2017-06.csv
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/mrtask_d.hadoop.20230508.162507.005622
Running step 1 of 1...
job output is in /tmp/mrtask_d.hadoop.20230508.162507.005622/output
Streaming final output from /tmp/mrtask_d.hadoop.20230508.162507.005622/output...
107     12.33
113     17.8
114     11.75
13      19.0
132     29.6
137     13.0
138     25.5
141     10.33
142     10.67
143     9.0
144     5.0
145     1.33
146     12.0
148     11.75
151     17.0
158     12.0
161     11.57
162     5.0
163     11.0
164     9.0
166     54.0
170     8.0
181     12.0
193     0.0
211     30.0
229     8.0
230     7.0
231     5.0
232     16.0
234     13.0
236     5.0
237     9.5
238     7.0
239     13.0
246     15.5
249     8.33
25      4.0
255     39.0
263     10.0
264     2.0
4       6.0
48      10.5
50      8.0
68      10.0
79      10.75
87      20.0
90      3.0
Removing temp directory /tmp/mrtask_d.hadoop.20230508.162507.005622...
[hadoop@ip-172-31-70-201 ~]$
```

Code - mrtask_d.py

```
from mrjob.job import MRJob
from mrjob.step import MRStep
from datetime import datetime
from itertools import groupby
import csv

class TripTime(MRJob):

    def mapper(self, _, line):
        if line.startswith('ID'):
            return
    # Parse the CSV line
        data = next(csv.reader([line]))
# each line of input data is parsed and the pickup location, dropoff location, pickup time, and dropoff time are extracted
        try:
            pickup_loc = int(data[8])
            dropoff_loc = int(data[9])
            pickup_time = datetime.strptime(data[2], '%Y-%m-%d %H:%M:%S')
            dropoff_time = datetime.strptime(data[3], '%Y-%m-%d %H:%M:%S')
            yield pickup_loc, (dropoff_time - pickup_time).seconds / 60
        except:
            pass
# trip time is calculated by subtracting the pickup time from the dropoff time, converting the result to minutes
    def reducer(self, pickup_loc, trip_times):
        total_trip_time = 0
        count = 0
# trip times for each pickup location are aggregated by calculating the total trip time and the count of trips for that pickup location
        for trip_time in trip_times:
            total_trip_time += trip_time
            count += 1
        yield pickup_loc, round(total_trip_time / count, 2)
# Return a list of steps that define the job
    def steps(self):
        return [
            MRStep(mapper=self.mapper,
                   reducer=self.reducer)
        ]
if __name__ == '__main__':
    TripTime.run()
```

5) **Calculate the average tips to revenue ratio of the drivers for different pickup locations in sorted format**.

```
[hadoop@ip-172-31-70-201 ~]$ cat mrtask_e.py
from mrjob.job import MRJob
import csv
from mrjob.step import MRStep

class TipsToRevenueRatio(MRJob):
    def mapper(self, _, line):
        # Skip the header row
        if line.startswith('ID'):
            return

        # Parse the CSV line
        row = next(csv.reader([line]))

        # Extract the relevant columns
        pickup_location = int(row[8])
        tip_amount = float(row[14])
        total_amount = float(row[17])

        # Emit the pickup location and the tip to revenue ratio
        if total_amount > 0:
            yield pickup_location, tip_amount / total_amount

    def reducer(self, key, values):
        # Calculate the average tip to revenue ratio for each pickup location
        tip_to_revenue_ratios = list(values)
        avg_tip_to_revenue_ratio = round(sum(tip_to_revenue_ratios) / len(tip_to_revenue_ratios), 2)
        yield key, avg_tip_to_revenue_ratio

    def steps(self):
        return [
            MRStep(mapper=self.mapper,
                   reducer=self.reducer),
            MRStep(mapper=None,
                   reducer=self.reducer_sort)
        ]

    def reducer_sort(self, key, values):
        for value in sorted(values):
            yield key, value


if __name__ == '__main__':
    TipsToRevenueRatio.run()
```

Output -

```
[hadoop@ip-172-31-70-201 ~]$ python mrtask_e.py yellow_tripdata_2017-06.csv
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/mrtask_e.hadoop.20230508.163535.307045
Running step 1 of 2...
Running step 2 of 2...
job output is in /tmp/mrtask_e.hadoop.20230508.163535.307045/output
Streaming final output from /tmp/mrtask_e.hadoop.20230508.163535.307045/output...
107     0.12
113     0.02
114     0.13
13      0.17
132     0.12
137     0.0
138     0.09
141     0.14
142     0.06
143     0.0
144     0.07
145     0.07
146     0.14
148     0.04
151     0.0
158     0.17
161     0.09
162     0.0
163     0.0
164     0.07
166     0.0
170     0.19
181     0.0
193     0.0
211     0.03
229     0.08
230     0.15
231     0.2
232     0.24
234     0.17
236     0.17
237     0.06
238     0.0
239     0.09
246     0.09
249     0.13
25      0.15
255     0.03
263     0.0
264     0.0
4       0.17
48      0.15
50      0.17
68      0.0
79      0.03
87      0.0
90      0.23
Removing temp directory /tmp/mrtask_e.hadoop.20230508.163535.307045...
[hadoop@ip-172-31-70-201 ~]$
```

Code - mrtask_e.py

```
from mrjob.job import MRJob
import csv
from mrjob.step import MRStep

class TipsToRevenueRatio(MRJob):

    def mapper(self, _, line):
        # Skip the header row
        if line.startswith('ID'):
            return

        # Parse the CSV line
        row = next(csv.reader([line]))

        # Extract the relevant columns
        pickup_location = int(row[8])
        tip_amount = float(row[14])
        total_amount = float(row[17])

        # Emit the pickup location and the tip to revenue ratio
        if total_amount > 0:
            yield pickup_location, tip_amount / total_amount

    def reducer(self, key, values):
        # Calculate the average tip to revenue ratio for each pickup location
        tip_to_revenue_ratios = list(values)
        avg_tip_to_revenue_ratio = round(sum(tip_to_revenue_ratios) / len(tip_to_revenue_ratios), 2)
        yield key, avg_tip_to_revenue_ratio
# defines the two steps of the MapReduce job in the First steps and reducer_sort in the second steps
    def steps(self):
        return [
            MRStep(mapper=self.mapper,
                   reducer=self.reducer),
            MRStep(mapper=None,
                   reducer=self.reducer_sort)
        ]
# sorts the output by pickup location and average ratio.
    def reducer_sort(self, key, values):
        for value in sorted(values):
            yield key, value

if __name__ == '__main__':
    TipsToRevenueRatio.run()
```

6) **How does revenue vary over time? Calculate the average trip revenue per month - analysing it by hour of the day (day vs night) and the day of the week (weekday vs weekend).**

implementation to calculate the average revenue per trip by different time dimensions, including year, month, hour, and weekday

```
[hadoop@ip-172-31-69-50 ~]$ cat mrtask_f.py
from mrjob.job import MRJob
from datetime import datetime

class MRRevenueByTime(MRJob):

    def mapper(self, _, line):
        fields = line.split(',')
        if fields[0] == 'ID':
            return
        pickup_time = datetime.strptime(fields[2], '%Y-%m-%d %H:%M:%S')
        revenue = float(fields[17])

        ## extracts the year, month, hour, and weekday

        yield (pickup_time.year, pickup_time.month, pickup_time.hour, pickup_time.weekday()), (revenue, 1)

    def reducer(self, key, values):
        total_revenue = 0
        total_trips = 0
        ##calculates the total revenue and total number of trips for each group
        for revenue, trips in values:
            total_revenue += revenue
            total_trips += trips
        yield key, round(total_revenue / total_trips,2 )

if __name__ == '__main__':
    MRRevenueByTime.run()
```

Output –

```
[hadoop@ip-172-31-69-50 ~]$ python mrtask_f.py yellow_tripdata_2017-06.csv
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/mrtask_f.hadoop.20230508.193132.872876
Running step 1 of 1...
job output is in /tmp/mrtask_f.hadoop.20230508.193132.872876/output
Streaming final output from /tmp/mrtask_f.hadoop.20230508.193132.872876/output...
[2017, 6, 0, 3] 16.62
Removing temp directory /tmp/mrtask_f.hadoop.20230508.193132.872876...
[hadoop@ip-172-31-69-50 ~]$
```

Code - mrtask_f.py

```
from mrjob.job import MRJob
from datetime import datetime

class MRRevenueByTime(MRJob):

    def mapper(self, _, line):
        fields = line.split(',')
        if fields[0] == 'ID':
            return
```

```
        pickup_time = datetime.strptime(fields[2], '%Y-%m-%d %H:%M:%S')
        revenue = float(fields[17])

        ## extracts the year, month, hour, and weekday

        yield (pickup_time.year, pickup_time.month, pickup_time.hour,
pickup_time.weekday()), (revenue, 1)

    def reducer(self, key, values):
        total_revenue = 0
        total_trips = 0
        ##calculates the total revenue and total number of trips for each group
        for revenue, trips in values:
            total_revenue += revenue
            total_trips += trips
        yield key, round(total_revenue / total_trips,2 )

if __name__ == '__main__':
    MRRevenueByTime.run()
```