**# Mapper code for maximum departure delay for each originating airport**

#!/usr/bin/env python

import sys

import csv

SEP = "," #easy to change separator

class Mapper(object):

def \_\_init\_\_(self, stream, sep=SEP):

self.stream = stream

self.sep = SEP

def emit(self, key, value):

sys.stdout.write("{0}{1}{2}\n".format(key, self.sep, value))

def map(self):

for row in self:

self.emit(row[3], row[6])

def \_\_iter\_\_(self):

reader = csv.reader(self.stream)

for row in reader:

yield row

if \_\_name\_\_ == "\_\_main\_\_":

mapper = Mapper(sys.stdin)

mapper.map()

**# Reducer code for maximum departure delay for each originating airport**

#!/usr/bin/env python

import sys

from itertools import groupby

from operator import itemgetter

SEP = "," #easy to change separator

class Reducer(object):

def \_\_init\_\_(self, stream, sep=SEP):

self.stream = stream

self.sep = sep

def emit(self, key, value):

sys.stdout.write("{0}{1}{2}\n".format(key, self.sep, value))

def reduce(self):

for current, group in groupby(self, itemgetter(0)):

total = 0

count = 0

lst=[]

for item in group:

lst.append(item[1])

#total += item[1]

#count += 1

self.emit(current, max(lst))

#self.emit(current, float(total)/float(count))

def \_\_iter\_\_(self):

for line in self.stream:

try: #this is important

parts = line.split(self.sep)

yield parts[0], float(parts[1])

except:

continue

if \_\_name\_\_ == "\_\_main\_\_":

reducer = Reducer(sys.stdin)

reducer.reduce()

**# output for maximum departure delay for each originating airport**

**ABE,374.0**

**ABQ,1274.0**

**ABR,507.0**

**ABY,182.0**

**ACT,211.0**

**ACV,121.0**

**ACY,210.0**

**ADK,59.0**

**ADQ,40.0**

**AEX,243.0**

**# Mapper code for average arrival delay by flight#**

#!/usr/bin/env python

import sys

import csv

SEP = "," #easy to change separator

class Mapper(object):

def \_\_init\_\_(self, stream, sep=SEP):

self.stream = stream

self.sep = SEP

def emit(self, key, value):

sys.stdout.write("{0}{1}{2}\n".format(key, self.sep, value))

def map(self):

for row in self:

self.emit(row[2], row[8])

def \_\_iter\_\_(self):

reader = csv.reader(self.stream)

for row in reader:

yield row

if \_\_name\_\_ == "\_\_main\_\_":

mapper = Mapper(sys.stdin)

mapper.map()

**# Reducer code for average arrival delay by flight#**

#!/usr/bin/env python

import sys

from itertools import groupby

from operator import itemgetter

SEP = "," #easy to change separator

class Reducer(object):

def \_\_init\_\_(self, stream, sep=SEP):

self.stream = stream

self.sep = sep

def emit(self, key, value):

sys.stdout.write("{0}{1}{2}\n".format(key, self.sep, value))

def reduce(self):

for current, group in groupby(self, itemgetter(0)):

total = 0

count = 0

for item in group:

total += item[1]

count += 1

self.emit(current, float(total)/float(count))

def \_\_iter\_\_(self):

for line in self.stream:

try: #this is important

parts = line.split(self.sep)

yield parts[0], float(parts[1])

except:

continue

if \_\_name\_\_ == "\_\_main\_\_":

reducer = Reducer(sys.stdin)

reducer.reduce()

**# Output for average arrival delay by flight#.**

1,-7.81318681319

2,1.95

3,-5.48275862069

4,-0.441964285714

5,5.51315789474

6,5.0350877193

7,14.817679558

8,-0.383838383838

9,0.5390625

10,-2.17142857143

**# Mapper code for minimum arrival delay for all origin-destination airport combinations**

#!/usr/bin/env python

import sys

import csv

SEP = "," #easy to change separator

class Mapper(object):

def \_\_init\_\_(self, stream, sep=SEP):

self.stream = stream

self.sep = SEP

def emit(self, key, value):

sys.stdout.write("{0}{1}{2}\n".format(key, self.sep, value))

def map(self):

for row in self:

self.emit(row[3]+"-"+row[4], row[8])

def \_\_iter\_\_(self):

reader = csv.reader(self.stream)

for row in reader:

yield row

if \_\_name\_\_ == "\_\_main\_\_":

mapper = Mapper(sys.stdin)

mapper.map()

**# Reducer code for minimum arrival delay for all origin-destination airport combinations**

#!/usr/bin/env python

import sys

from itertools import groupby

from operator import itemgetter

SEP = "," #easy to change separator

class Reducer(object):

def \_\_init\_\_(self, stream, sep=SEP):

self.stream = stream

self.sep = sep

def emit(self, key, value):

sys.stdout.write("{0}{1}{2}\n".format(key, self.sep, value))

def reduce(self):

for current, group in groupby(self, itemgetter(0)):

total = 0

count = 0

lst=[]

for item in group:

lst.append(item[1])

self.emit(current, min(lst))

def \_\_iter\_\_(self):

for line in self.stream:

try: #this is important

parts = line.split(self.sep)

yield parts[0], float(parts[1])

except:

continue

if \_\_name\_\_ == "\_\_main\_\_":

reducer = Reducer(sys.stdin)

reducer.reduce()

**# Output for minimum arrival delay for all origin-destination airport combinations**

ABE-ATL,-33.0

ABE-DTW,-37.0

ABE-ORD,-25.0

ABQ-ATL,-29.0

ABQ-BWI,-19.0

ABQ-DAL,-21.0

ABQ-DEN,-31.0

ABQ-DFW,-35.0

ABQ-HOU,-17.0

ABQ-IAH,-38.0