Project ADA

Airline Delay Analysis

INFO 7250 Final Project

INFO 7250 Engineering Big-Data Systems
Jialiang Xiao
NUID 001569847
Northeastern University

Contents

1 OVERVIEW OF THE DATASET

1.1 Dataset Source and Basic Description

The dataset I chose for this project is Airline Delay Analysis. It is a Kaggle dataset, and its link is https://www.kaggle.com/sherrytp/airline-delay-analysis.
The datasets contain daily airline information covering flight information, carrier company, taxing-in, taxing-out time, and generalized delay reason of precisely eleven years, from 2009 to 2020. The data for 2020 is not complete. The DOT's database is renewed from 2018, so there might be a minor change in the column names.

1.2 Dataset Fields Explanation

The project mainly uses four dataset fields, namely **FL_DATE**, **OP_CARRIER**, **ORIGIN**, **DEST**, **DEP_DELAY,** and **ARR_DELAY**. Below are the simple explanations of all the dataset fields.

FL_DATE:
date of the flight
OP_CARRIER / OP_UNIQUE_CARRIER:
flight operation company
OP_CARRIER_FL_NUM:
flight number, maybe remain the same if it's a round trip on the same day
ORIGIN:
origin of the flight
DEST:
destination of the flight
CRS_DEP_TIME:
computer reservation system departure time
only 2019 doesn't have this column
DEP_TIME:
actual departure time
different meaning for 2020 (?)
DEP_DELAY:
DEP_TIME – CRS_DEP_TIME
Actual departure time – computed departure time
TAXI_OUT:
The Taxi Out phase includes many things happening in the cockpit and the cabin. The flight attendants make sure the cabin is ready for takeoff and conduct the emergency brief. The pilots are starting the engines and requesting permission to taxi.
Boarding to WHEELS_OFF

WHEELS_OFF:

To depart or travel away (from someone or something) on wheels or a wheeled vehicle or apparatus.

The time that wheel off the runaway (departure)

WHEELS_ON:

The time that wheel on the runaway (arrival)

TAXI_IN:

WHEELS_ON to getting to the gate

CRS_ARR_TIME:

computer reservation system arrival time

ARR_TIME:

Actual arrival time

ARR_DELAY:

Arrival delay

CANCELLED:

Whether this flight was canceled or not

CANCELLATION_CODE:

The cancellation code of the flight

DIVERTED:

Whether a flight was diverted or not

CRS_ELAPSED_TIME:

Computer reserve system estimated flight elapsing time

ACTUAL_ELAPSED_TIME:

Actual flight elapsing time

AIR_TIME:

Actual arrival time

DISTANCE:

The flight distance

CARRIER_DELAY:

The delay caused by the carrier

WEATHER_DELAY:

The delay caused by the weather

NAS_DELAY:

The delay caused by National Airspace System

SECURITY_DELAY:

The delay caused by the security issues

LATE_AIRCRAFT_DELAY:

Unknown

Unnamed: 27:

Unknown

2 Data Preprocessing

2.1 Null and Missing Values

There are rare null and missing values of the dataset fields DEP_DELAY and ARR_DELAY. For these values, I use value 0.0 as the default value in the analysis. Using 0.0 would not affect the whole analysis process and output for the rare occurrence of null and missing values.

2.2 Value Format Differentiation between tables

In table 2019.csv and 2020.csv, the two tables contain flight data for the years 2019 and 2020, and the data formats are different from other tables. I discard the 2019.csv and 2020.csv from the whole analysis process.

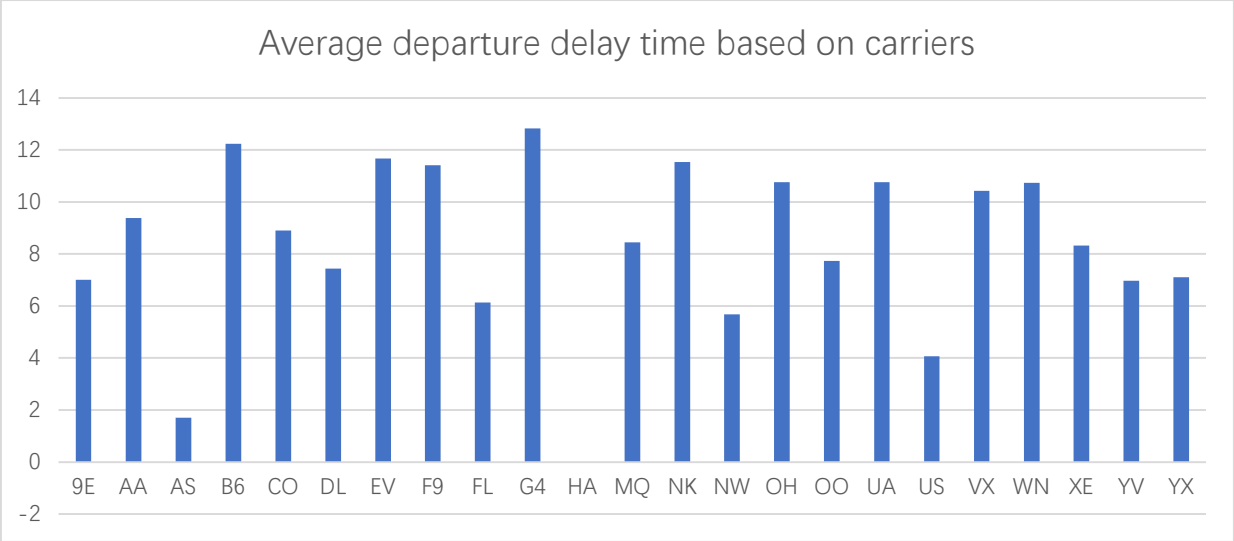
3 ANALYSIS OF FLIGHT DATA USING MAPREDUCE ON HADOOP

3.1 Analysis Basic Explanation

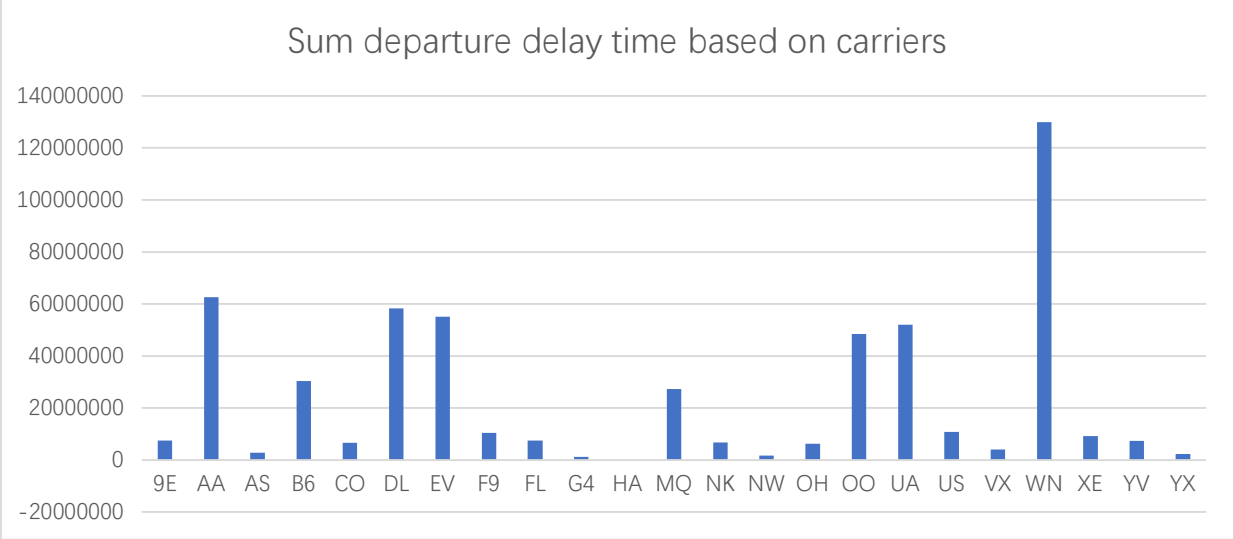
To find the most valuable information for users to determine which carrier to choose when flying from a certain origin to a certain destination, I first make several explorative analyses. Then I perform several analyses with carrier, origin, destination, and date combined to dig out more detailed information about flight delay patterns.
I only use Hadoop MapReduce with Java 8 without Apache Pig or Apache Hive for the whole task. The reason is that I believe the versatility of MapReduce would give me more options in dealing with data, especially when using custom WritableComparables, GroupComparators, and WritableComparators.
ADA is short for Airline Delay Analysis.
The detailed output files can be found in the output folder under the project folder.
Time unit for delay is minute.
Below are my analyses and the output.

3.2 Departure Delay Based on Carriers

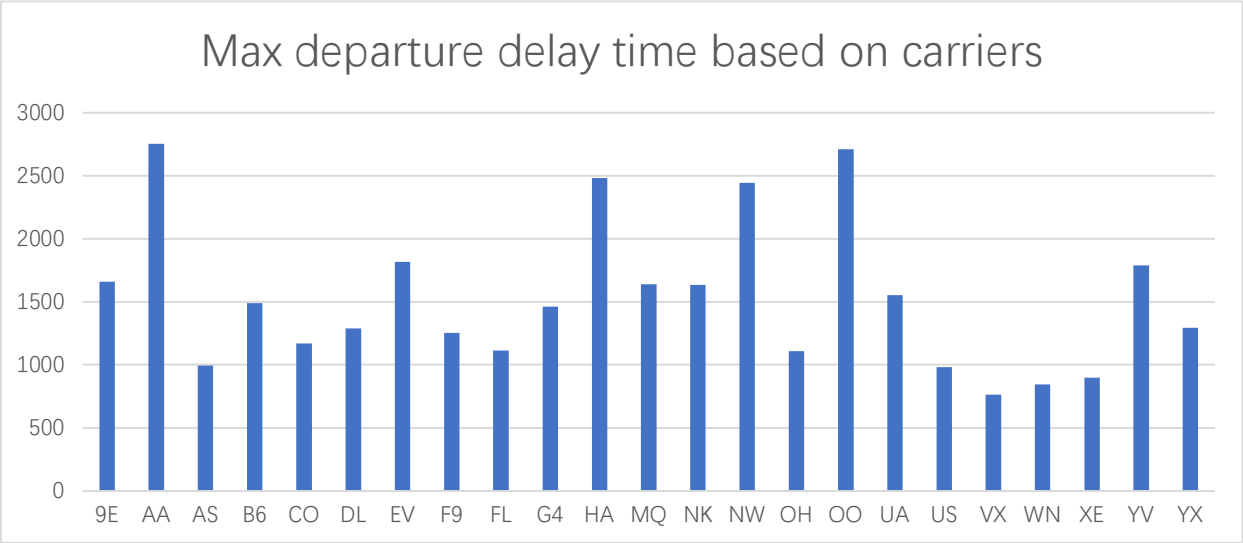
ADA1: Average departure delay time based on carriers



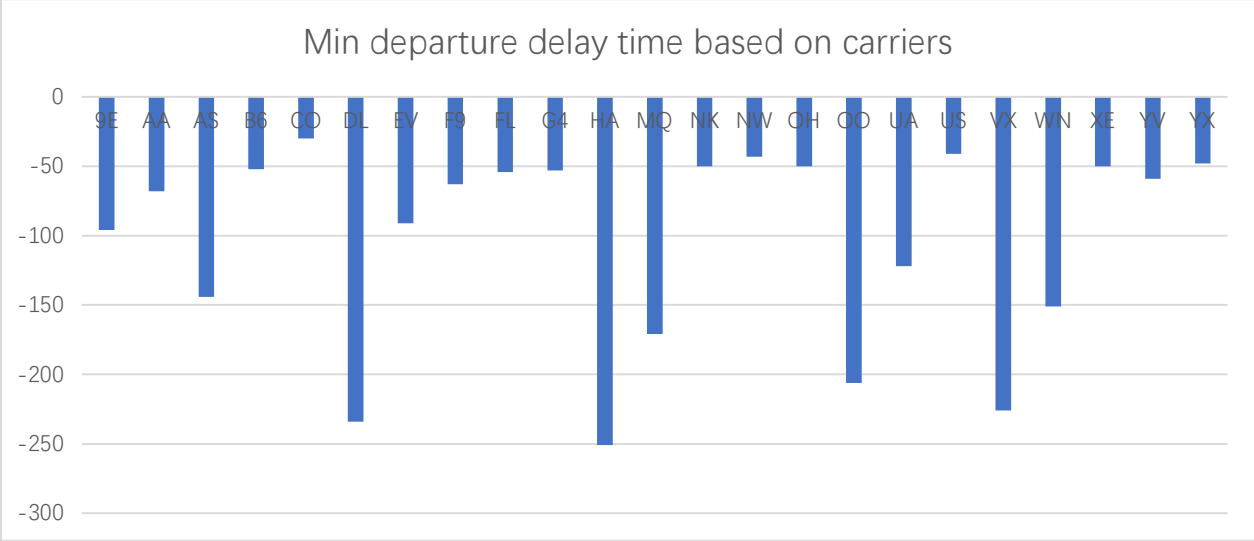
ADA2: Sum departure delay time based on carriers



ADA3: Max departure delay time based on carriers

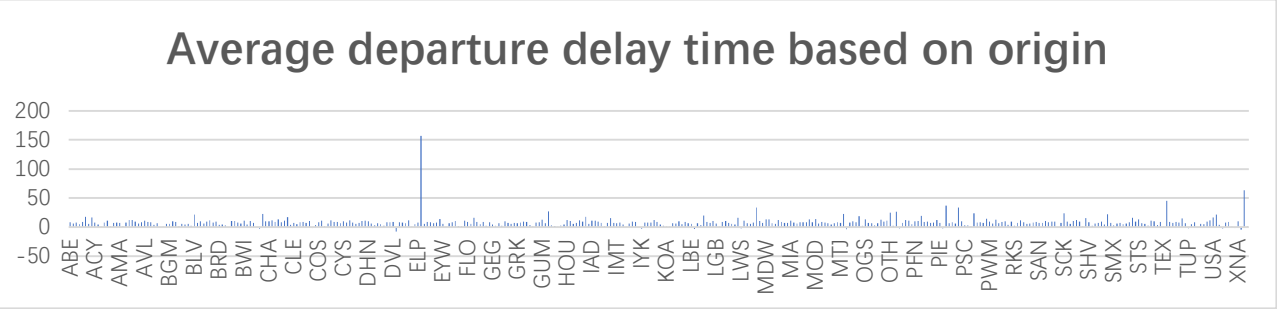


ADA4: Min departure delay time based on carriers

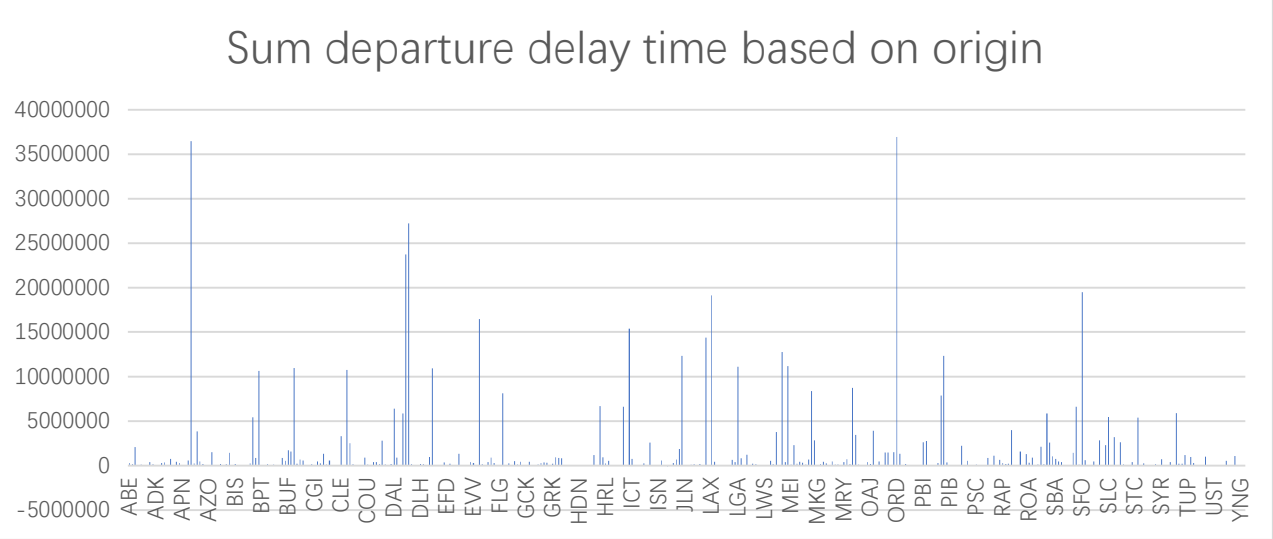


3.2 Departure Delay Based on Origins

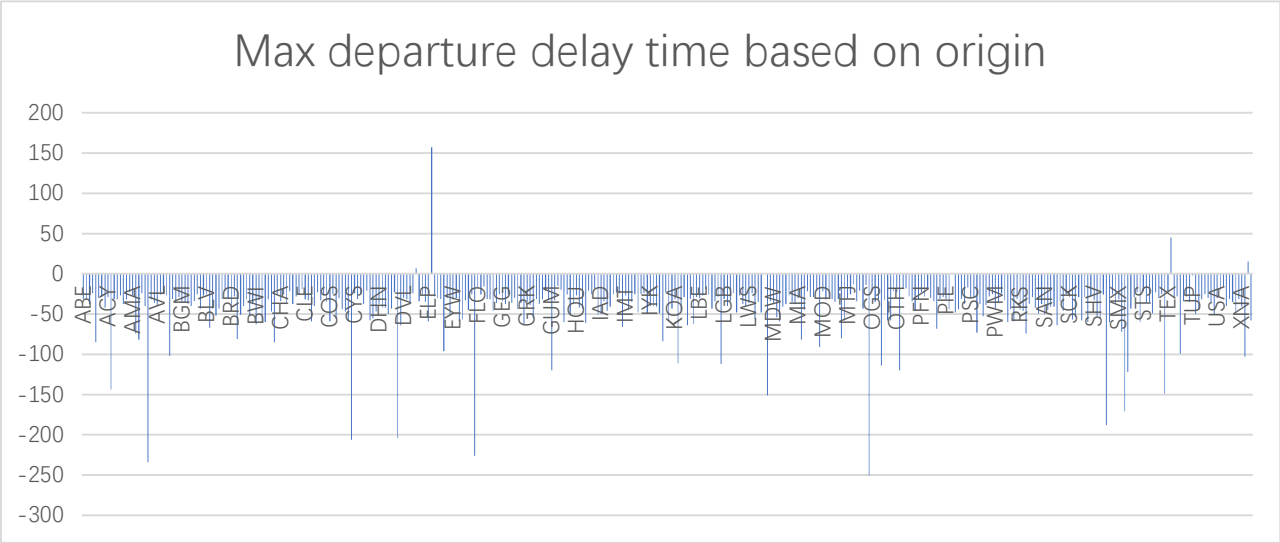
ADA5: Average departure delay time based on origin



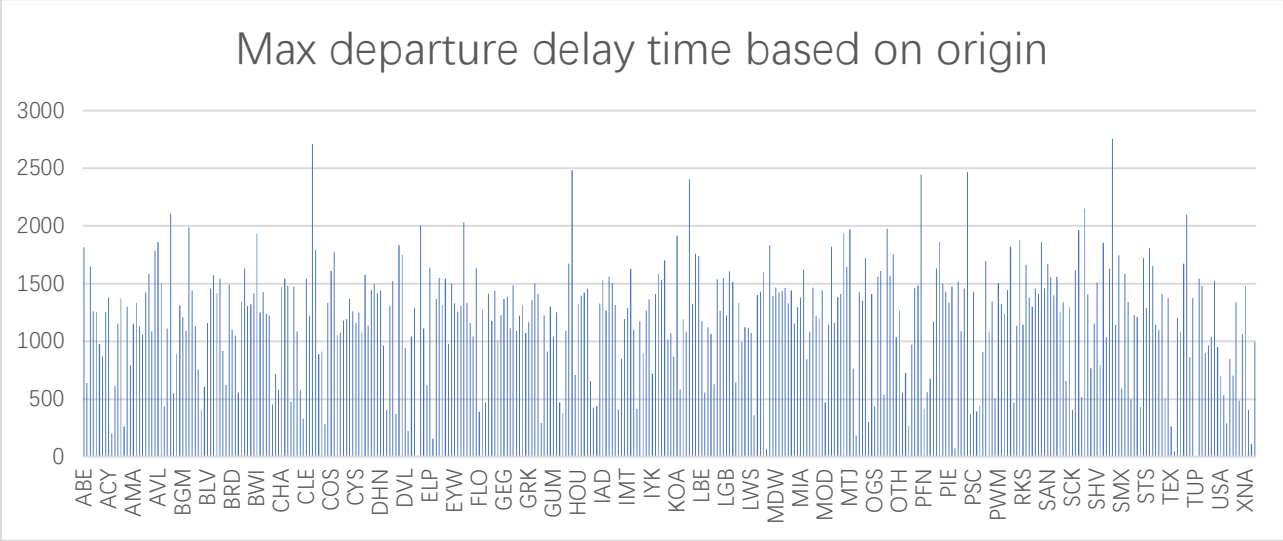
ADA6: Sum departure delay time based on origin



ADA7: Min departure delay time based on origin

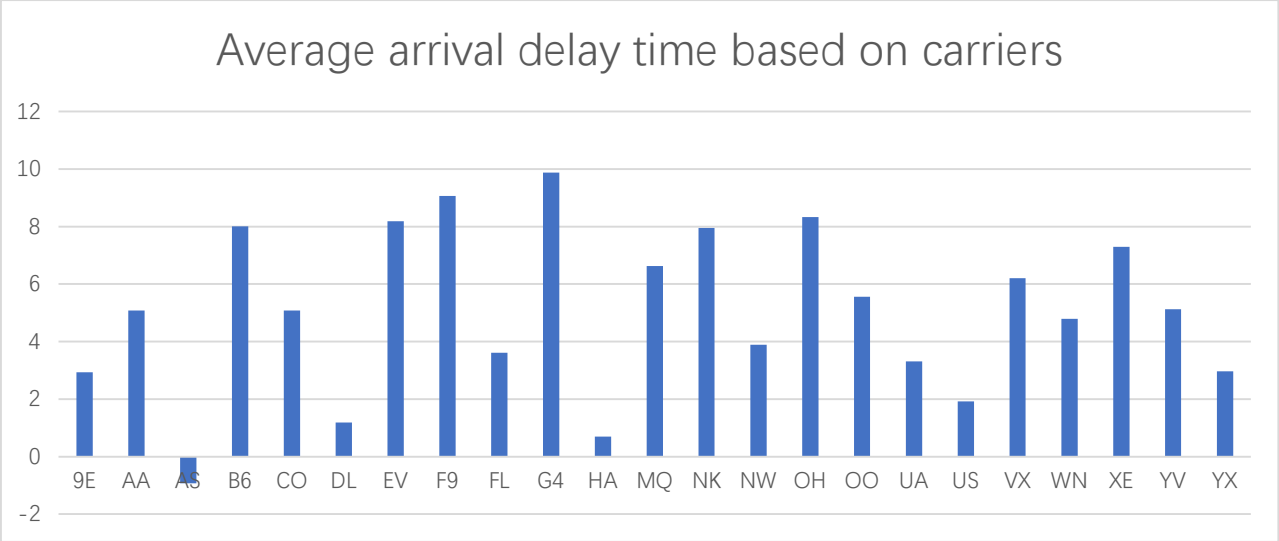


ADA8: Max departure delay time based on origin

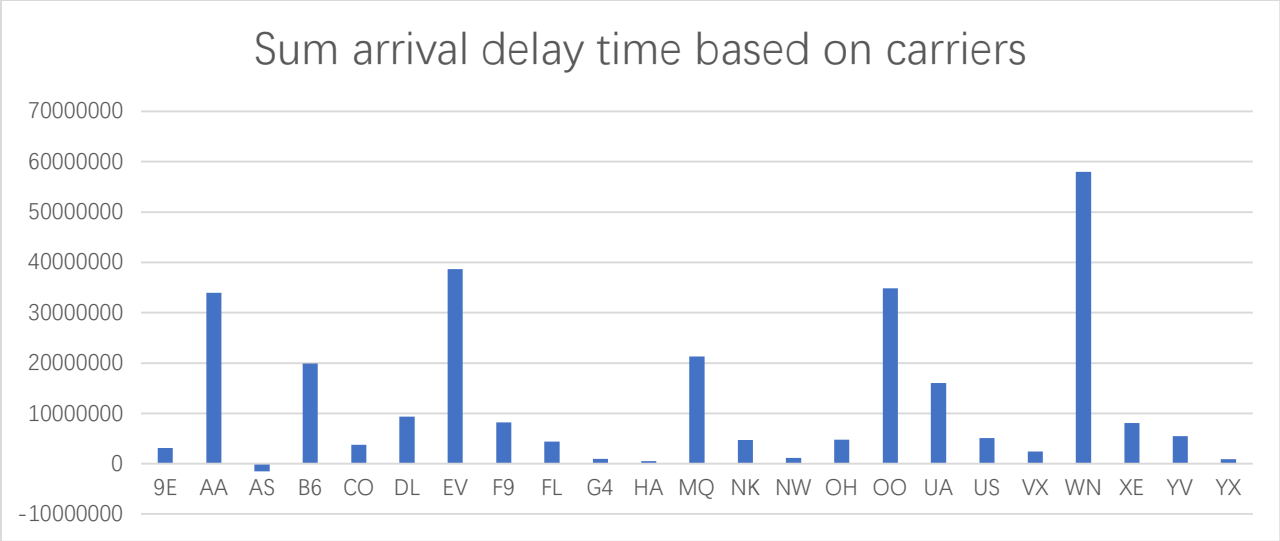


3.3 Arrival Delay Based on Carriers

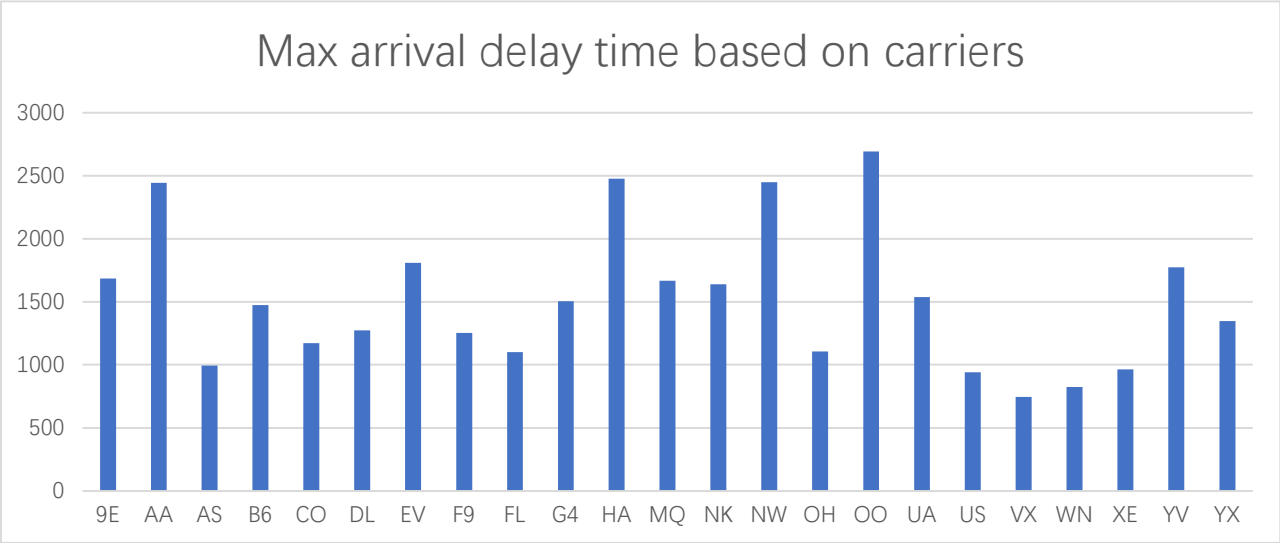
ADA9: Average arrival delay time based on carriers



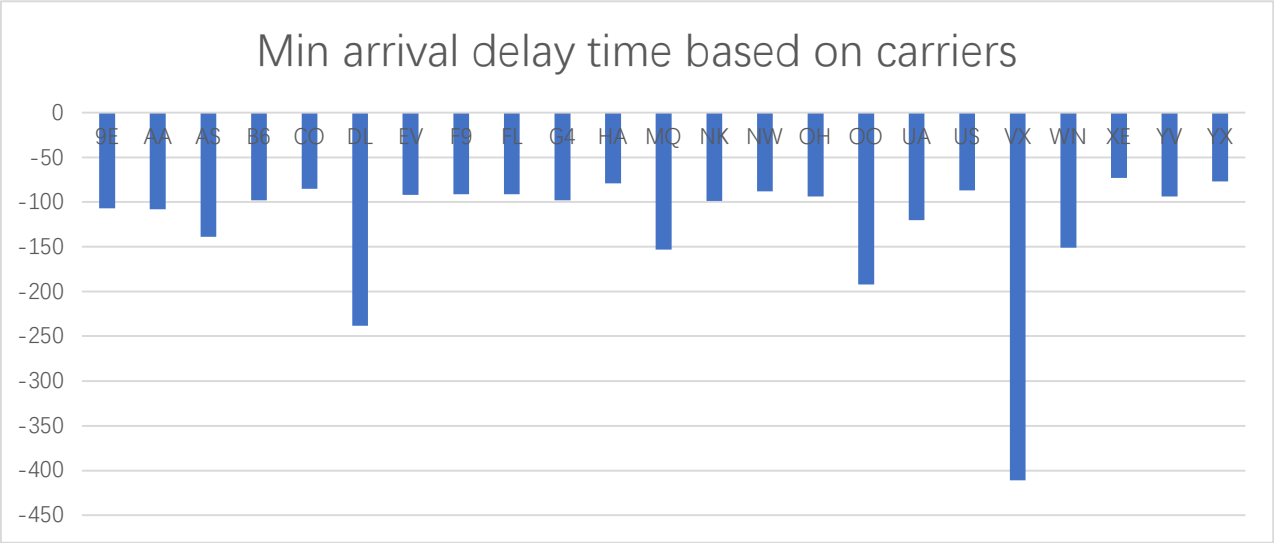
ADA10: Sum arrival delay time based on carriers



ADA11: Max arrival delay time based on carriers

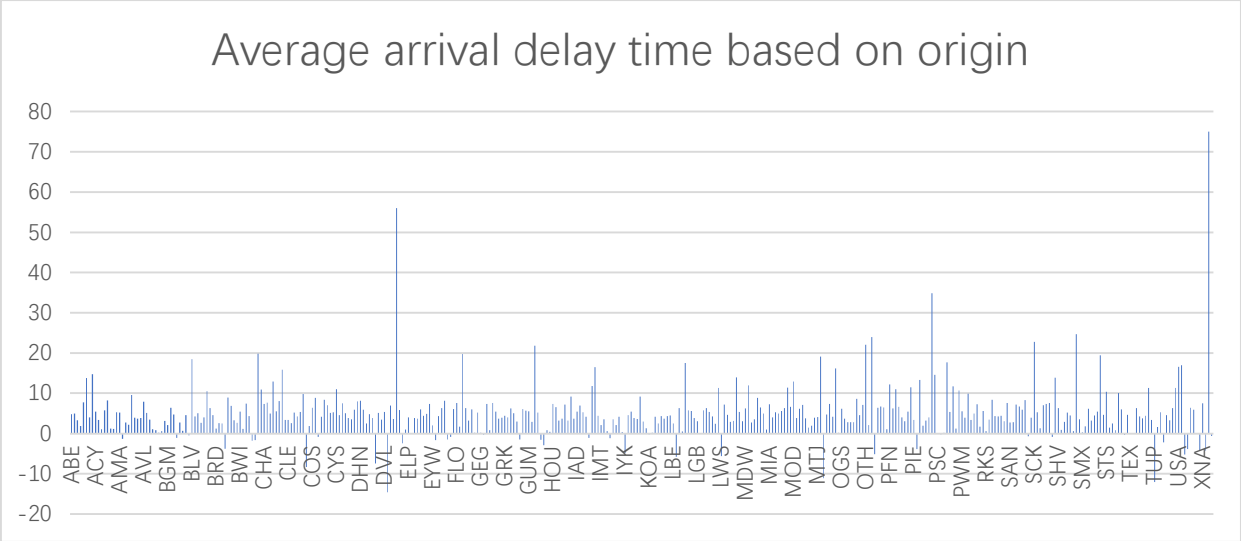


ADA12: Min arrival delay time based on carriers

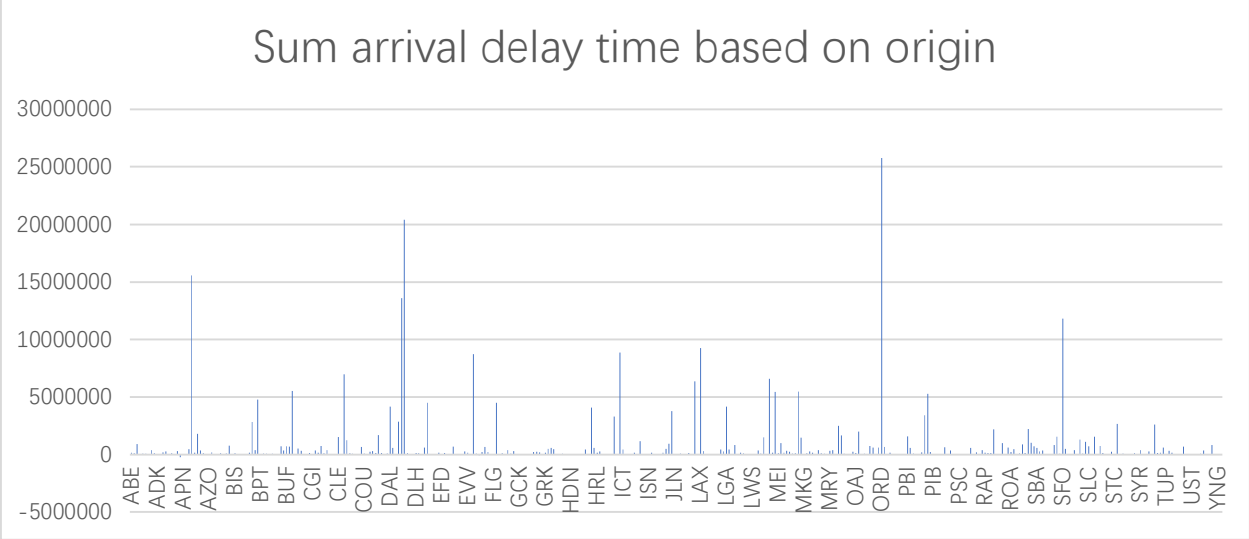


3.3 Arrival Delay Based on Origins

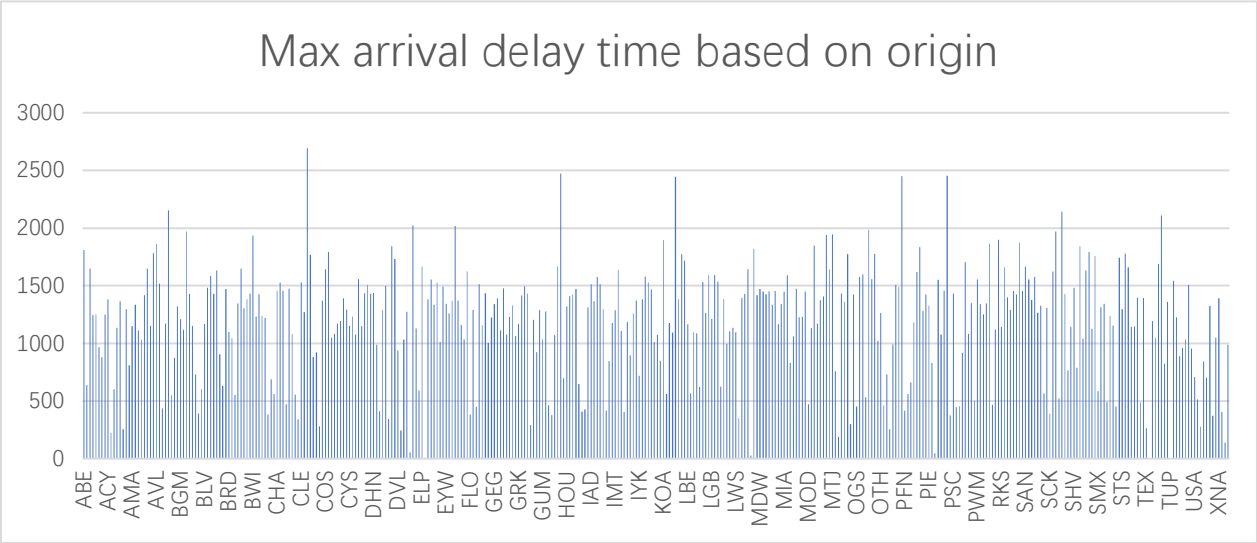
ADA13: Average arrival delay time based on origin



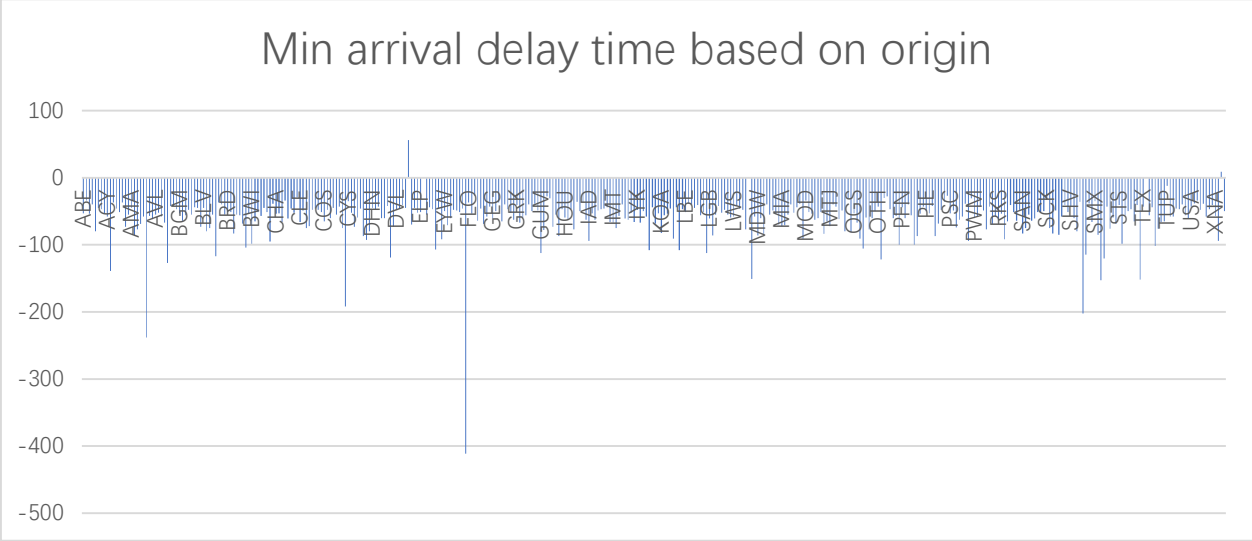
ADA14: Sum arrival delay time based on origin



ADA15: Max arrival delay time based on origin

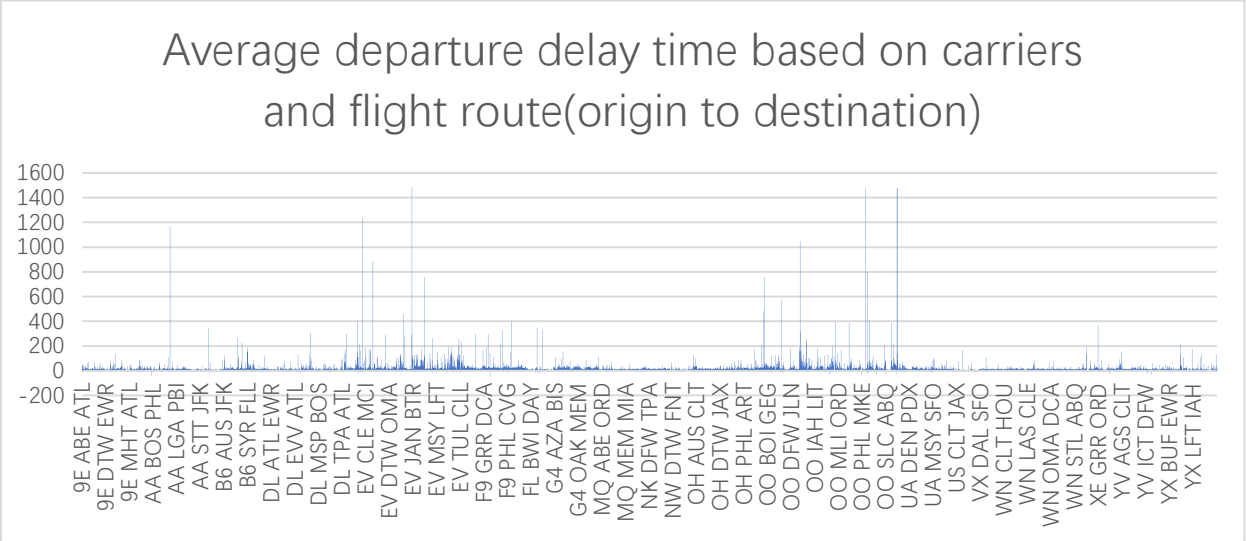


ADA16: Min arrival delay time based on origin

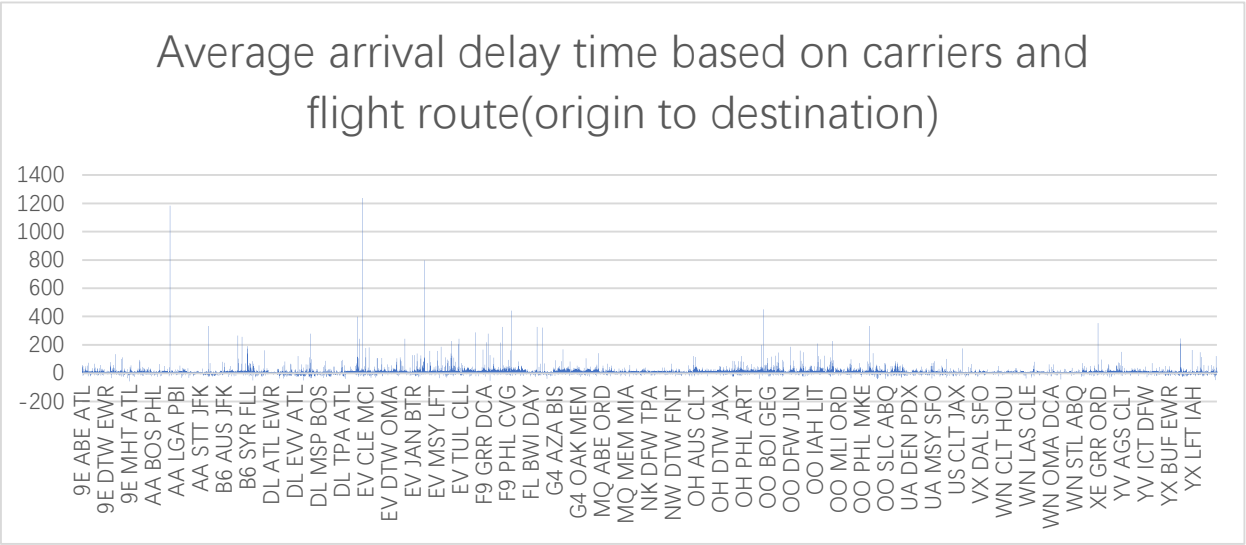


3.4 Departure and Arrival Delay Based on Carriers and Flight Routes

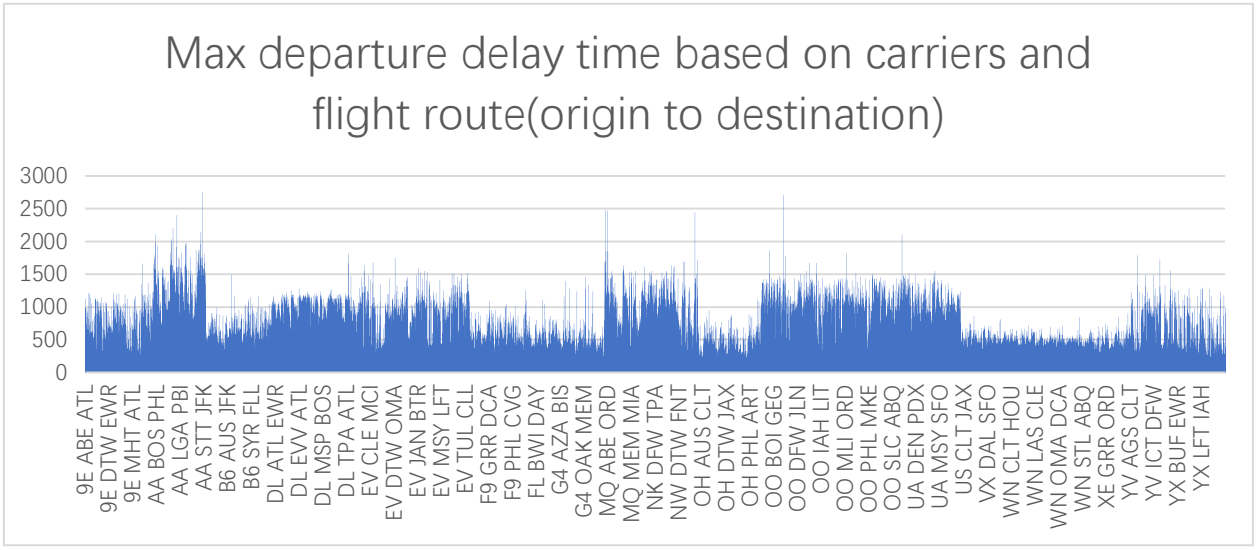
ADA17: Average departure delay time based on carriers and flight route(origin to destination)



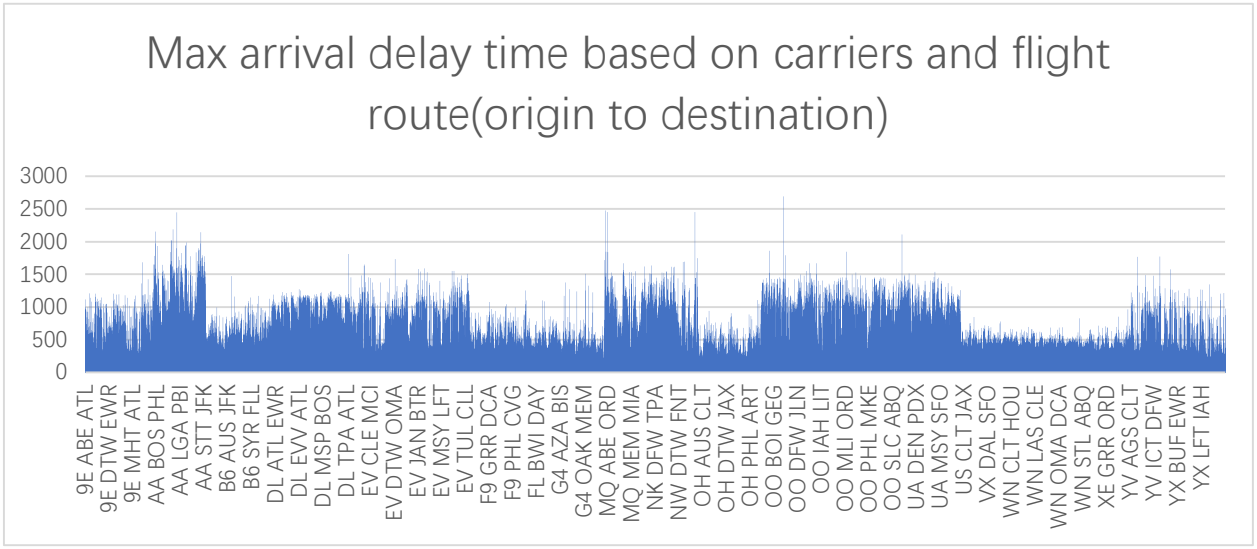
ADA18: Average arrival delay time based on carriers and flight route(origin to destination)



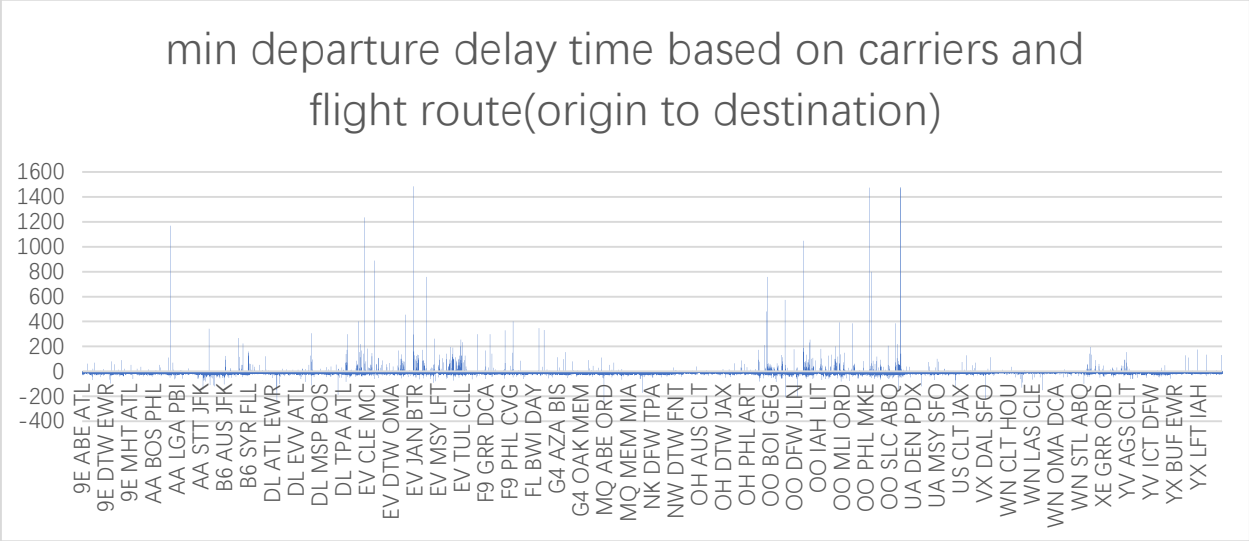
ADA19: Max departure delay time based on carriers and flight route(origin to destination)



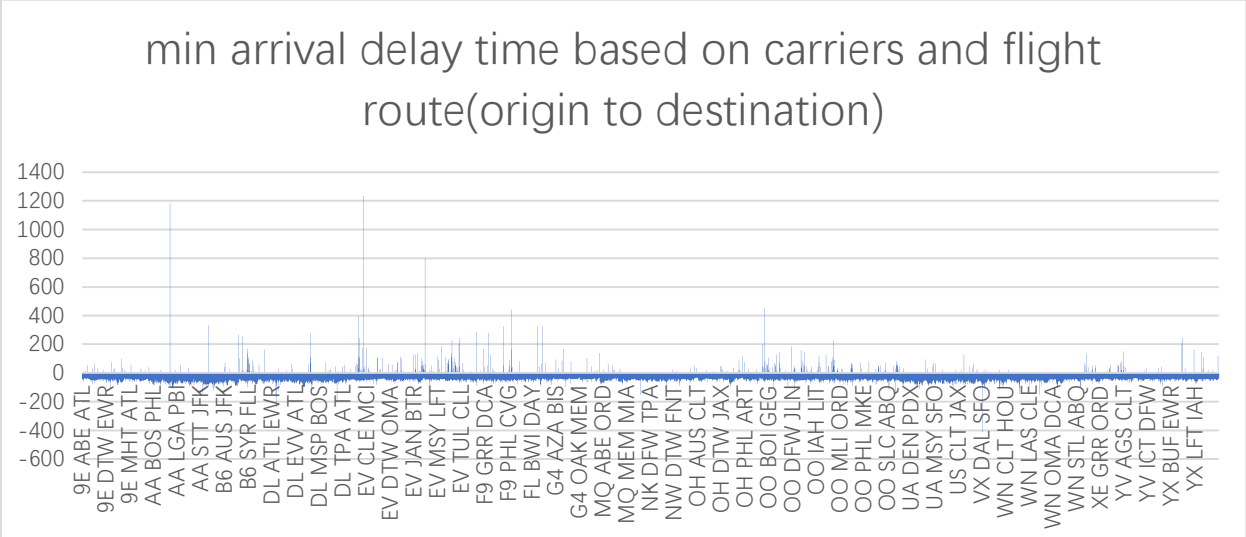
ADA20: Max arrival delay time based on carriers and flight route(origin to destination)



ADA21: min departure delay time based on carriers and flight route(origin to destination)

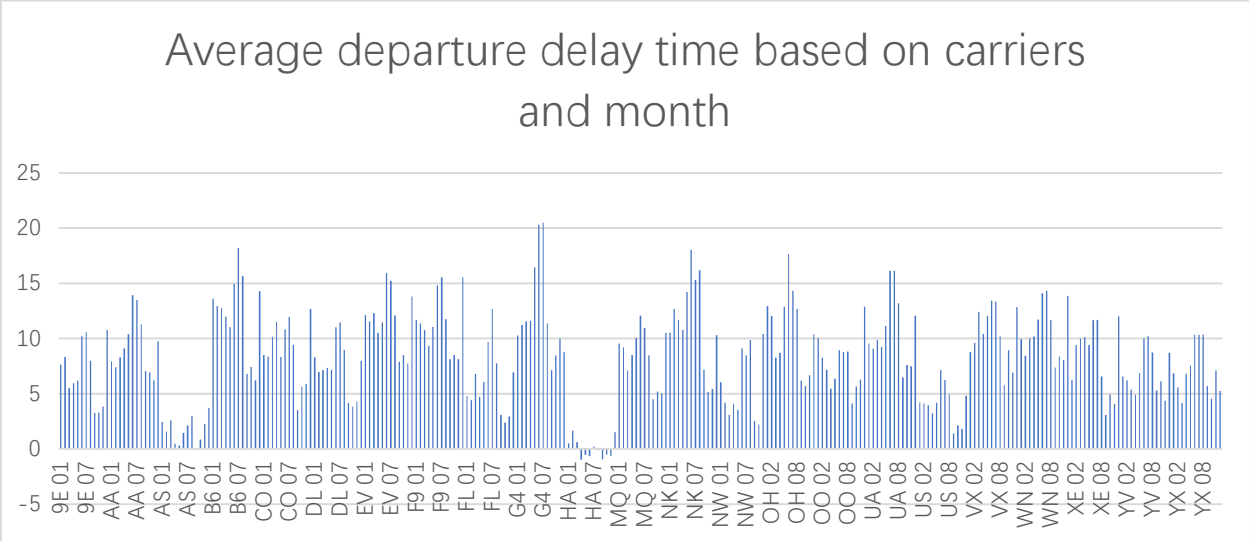


ADA22: min arrival delay time based on carriers and flight route(origin to destination)

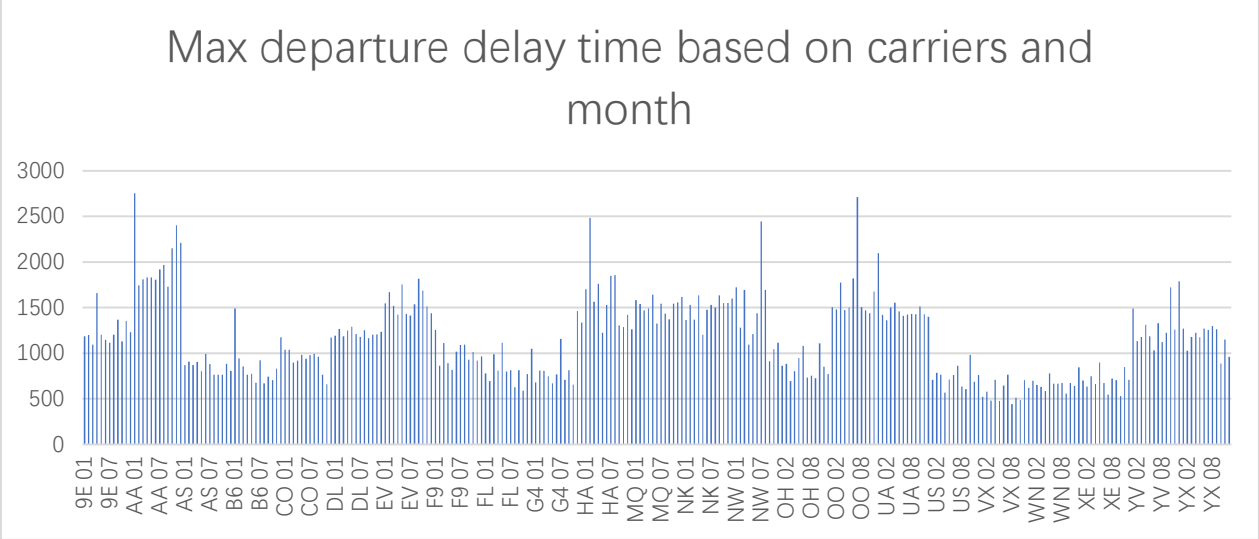


3.5 Departure Arrival Delay Based on Carriers and Month

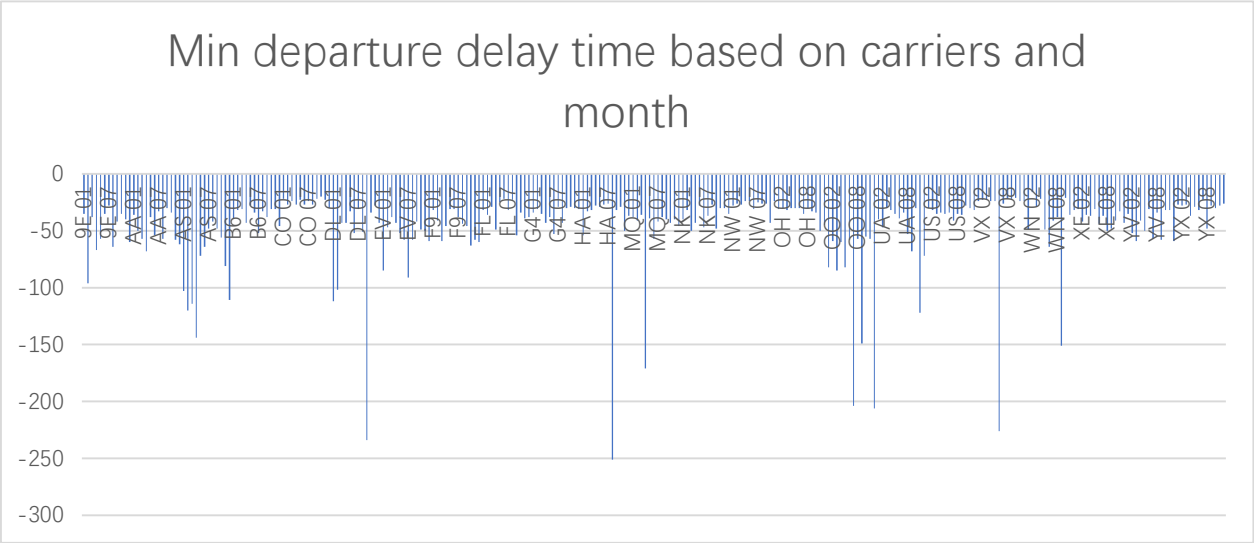
ADA23: Average departure delay time based on carriers and month



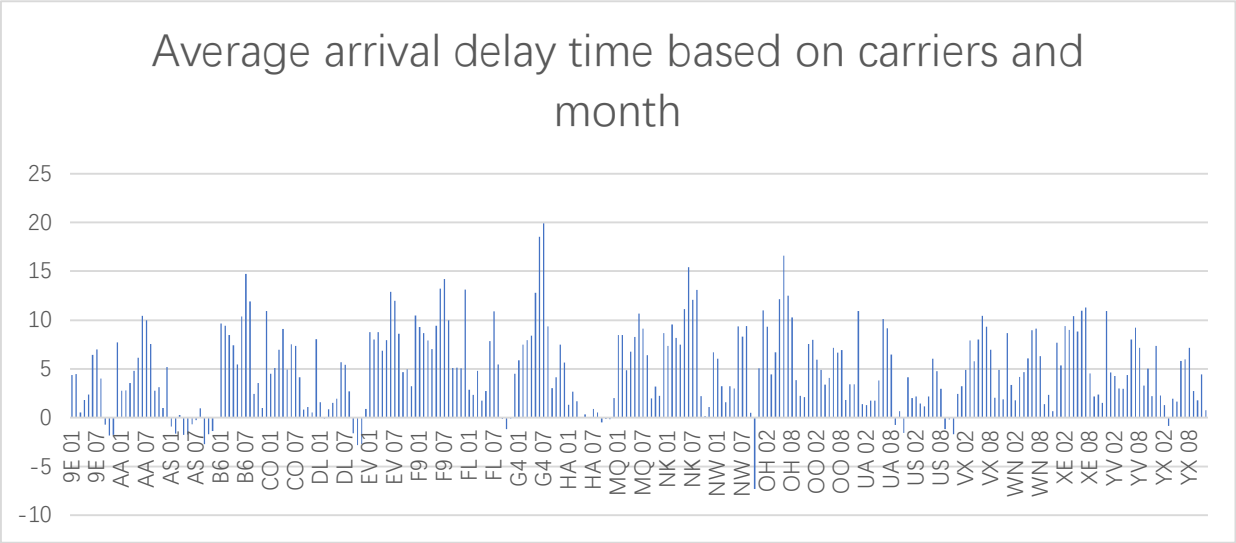
ADA24: Max departure delay time based on carriers and month



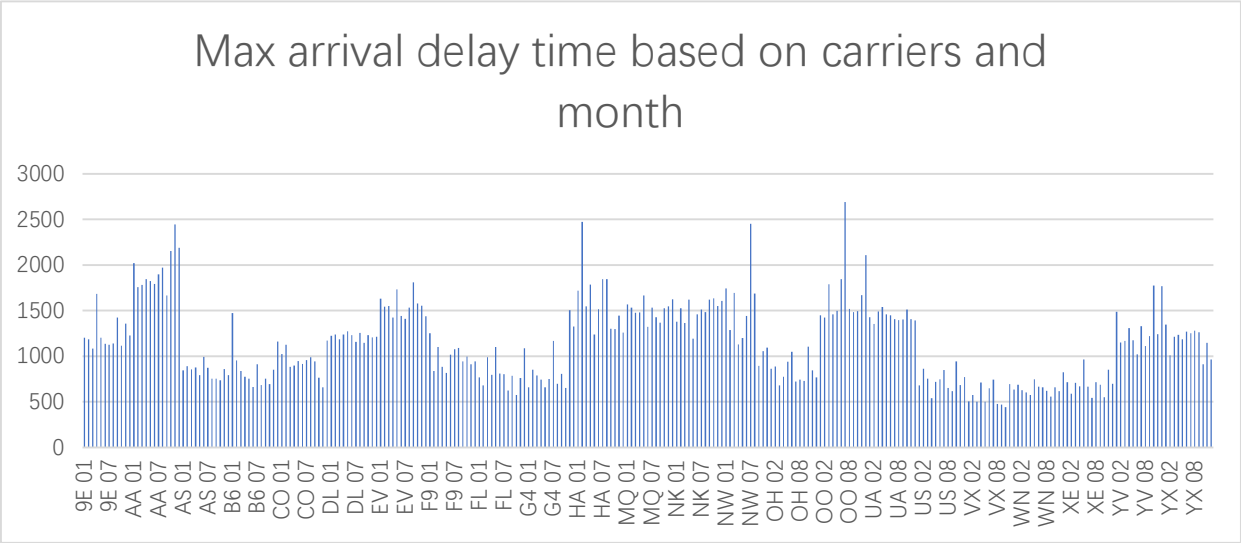
ADA25: Min departure delay time based on carriers and month



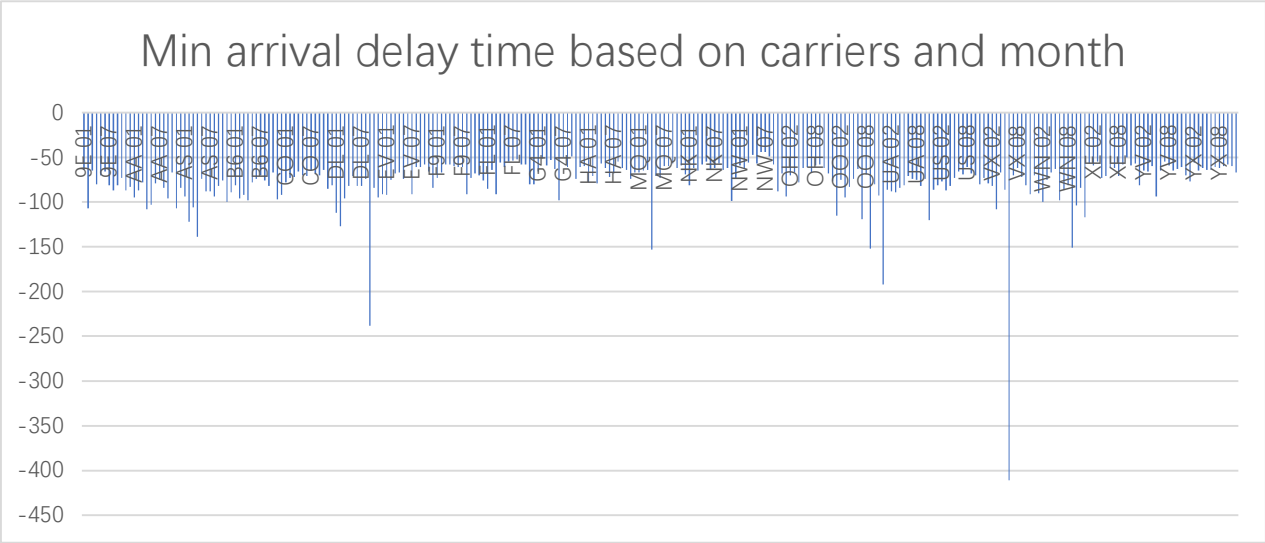
ADA26: Average arrival delay time based on carriers and month



ADA27: Max arrival delay time based on carriers and month

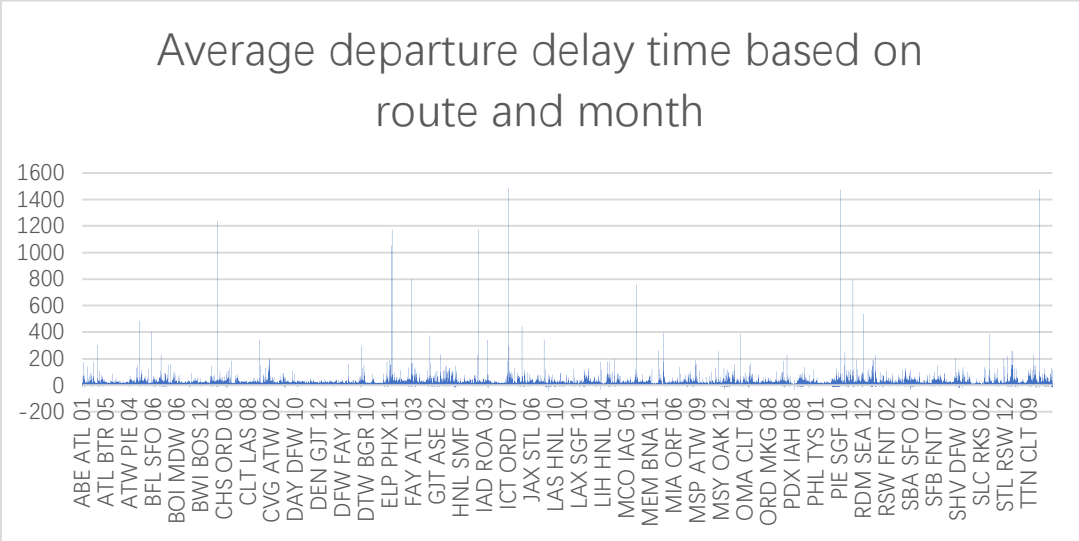


ADA28: Min arrival delay time based on carriers and month

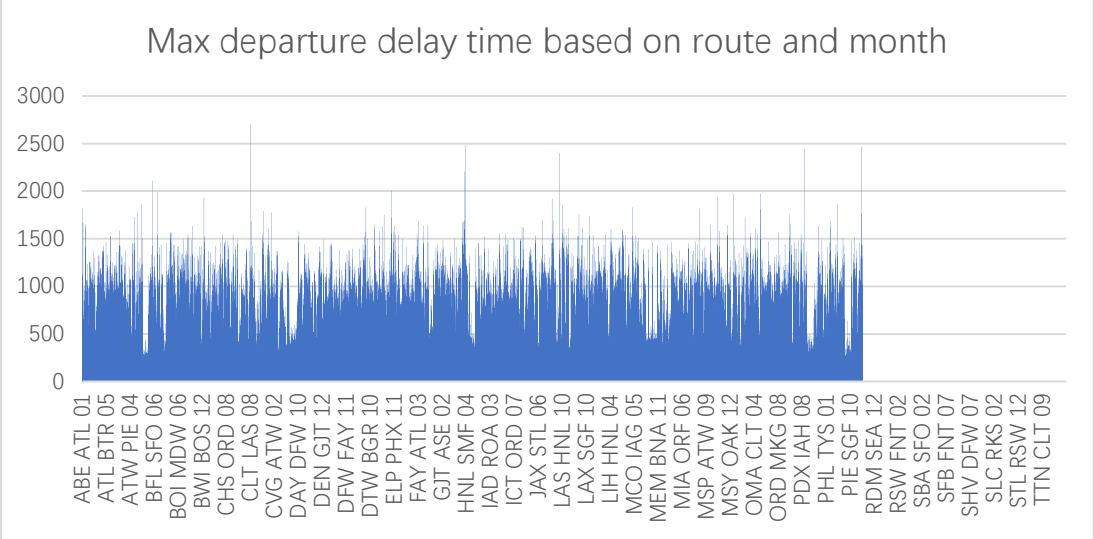


3.6 Departure Arrival Delay Based on Routes and Month

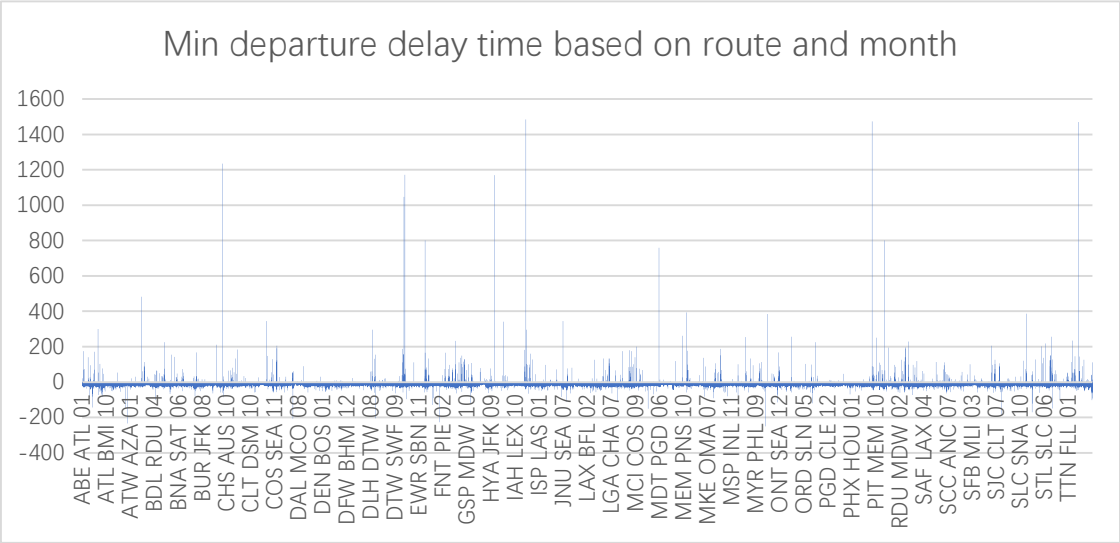
ADA29: Average departure delay time based on route and month



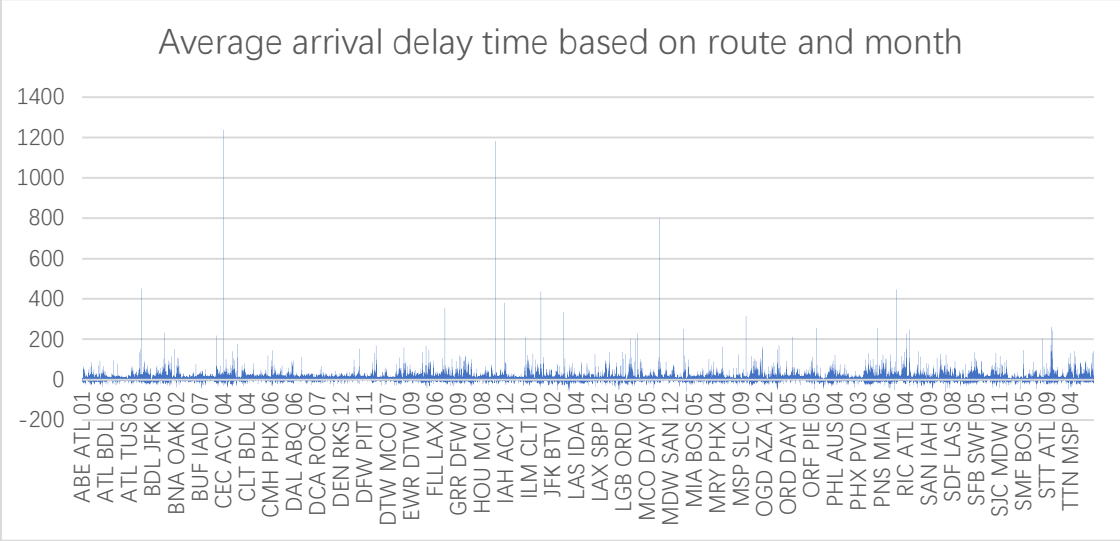
ADA30: Max departure delay time based on route and month



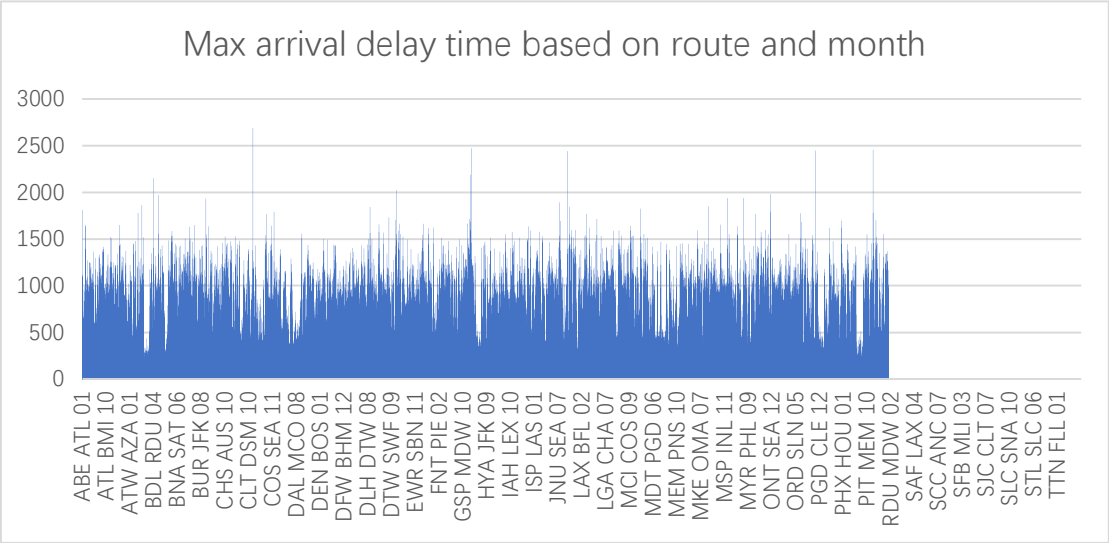
ADA31: Min departure delay time based on route and month



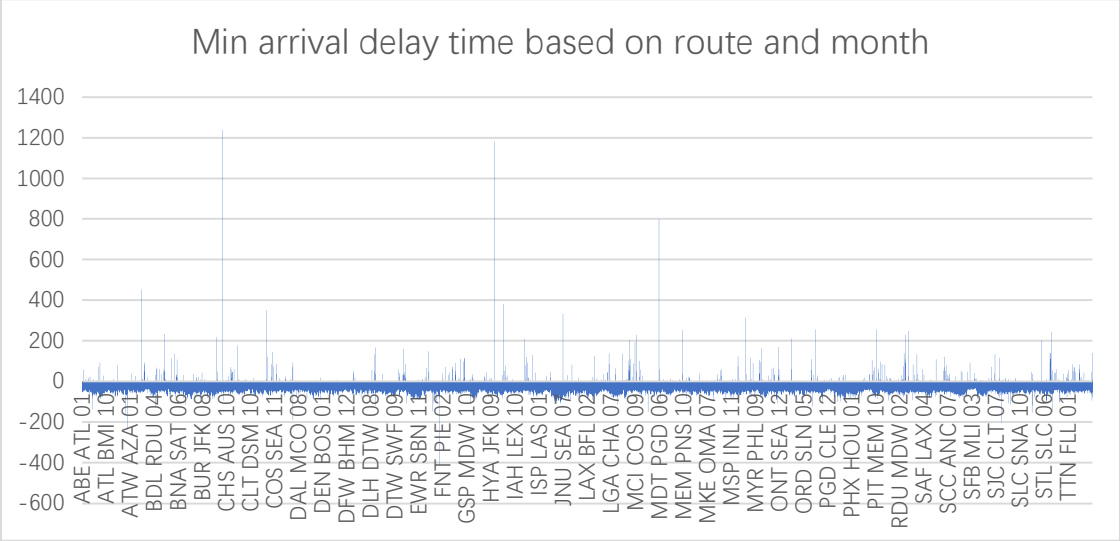
ADA32: Average arrival delay time based on route and month



ADA33: Max arrival delay time based on route and month



ADA34: Min arrival delay time based on route and month



4 REFERENCES

https://learning.oreilly.com/library/view/mapreduce-design-patterns/9781449341954/
https://learning.oreilly.com/library/view/data-algorithms/9781491906170/ch01.html

5 APPENDICES

The code at this project can be found at GitHub repository at:
https://github.com/XiaoQ162/INFO7250FINAL

5.1 ADAMain

```
6 import Comparators.CarrierMonthComparator;
  import Comparators.CarrierOriginDestComparator;
  import Comparators.OriginDestMonthComparator;
  import
  GroupComparators.CarrierMonthGroupComparator;
  import
  GroupComparators.CarrierOriginDestGroupComparato
  r;
  import
  GroupComparators.OriginDestMonthGroupComparator;
  import Mappers.*;
  import Partitioners.CarrierMonthPartitioner;
  import
  Partitioners.CarrierOriginDestPartitioner;
```

```
import Partitioners.OriginDestMonthPartitioner;
import Reducers.*;
import WritableComparables.CarrierMonth;
import WritableComparables.CarrierOriginDest;
import WritableComparables.OriginDestMonth;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import
org.apache.hadoop.mapreduce.lib.input.FileInputF
ormat;
import
org.apache.hadoop.mapreduce.lib.input.TextInputF
ormat;
import
org.apache.hadoop.mapreduce.lib.output.FileOutpu
tFormat;
import
org.apache.hadoop.mapreduce.lib.output.TextOutpu
```

```
tFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class ADAMain extends Configured
implements Tool {
    @Override
    public int run(String[] strings) throws
Exception {
        Configuration conf = new Configuration();
        Job job = new Job(conf,"play with
secondary sort");

        job.setJarByClass(ADAMain.class);


job.setGroupingComparatorClass(OriginDestMonthGr
oupComparator.class);

job.setSortComparatorClass(OriginDestMonthCompar
ator.class);
```

```
job.setPartitionerClass(OriginDestMonthPartition
er.class);


job.setInputFormatClass(TextInputFormat.class);
//
job.setOutputFormatClass(TextOutputFormat.class)
;


job.setOutputKeyClass(OriginDestMonth.class);

job.setOutputValueClass(DoubleWritable.class);


job.setMapperClass(OriginDestMonthArrDelayMapper
.class);
//       job.setCombinerClass(.class);

job.setReducerClass(OriginDestMonthArrDelayMinRe
ducer.class);
```

```
        FileInputFormat.addInputPath(job,new
Path(strings[0]));
        FileOutputFormat.setOutputPath(job,new
Path(strings[1]));



        boolean b = job.waitForCompletion(false);


        return b?0:1;
    }


    public static void main(String[] args)
throws Exception {
        int exit = ToolRunner.run(new
ADAMain(),args);
        System.exit(exit);


    }
}
```

5.2 WritableComparables.CarrierMonth

```
package WritableComparables;

import org.apache.hadoop.io.WritableComparable;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

public class CarrierMonth implements
WritableComparable {
    private String carrier;
    private String month;

    public CarrierMonth() { super(); }
    public CarrierMonth(String carrier, String
month) {
        super();
        this.carrier = carrier;
        this.month = month;
    }
```

```
    @Override
    public String toString() {
        return carrier + " " + month;
    }

    @Override
    public int compareTo(Object o) {
        if (o.getClass() != this.getClass()) {
            return 0;
        } else {
            int result =
this.carrier.compareTo(((CarrierMonth)o).getCarrie
r());
            if (result != 0) {
                return result;
            }
            return
this.month.compareTo(((CarrierMonth)o).getMonth())
;
        }
    }
```

```
    @Override
    public void write(DataOutput dataOutput)
throws IOException {
        dataOutput.writeUTF(carrier);
        dataOutput.writeUTF(month);
    }

    @Override
    public void readFields(DataInput dataInput)
throws IOException {
        carrier = dataInput.readUTF();
        month = dataInput.readUTF();
    }

    public String getCarrier() {
        return carrier;
    }

    public void setCarrier(String carrier) {
        this.carrier = carrier;
    }
```

```
    public String getMonth() {
        return month;
    }

    public void setMonth(String month) {
        this.month = month;
    }
}
```

5.3 WritableComparables.CarrierOriginDest

```
package WritableComparables;

import org.apache.hadoop.io.WritableComparable;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

public class CarrierOriginDest implements
WritableComparable {
    private String carrier;
    private String origin;
```

```
    private String destination;

    public CarrierOriginDest() { super(); }
    public CarrierOriginDest(String carrier,
String origin, String destination)
    {
        super();
        this.carrier = carrier;
        this.origin = origin;
        this.destination = destination;
    }

    @Override
    public String toString() {
        return carrier + " " + origin + " " +
destination;
    }

    @Override
    public int compareTo(Object o) {
        if (o.getClass() != this.getClass()) {
            return 0;
```

```
        } else {
            int result =
this.carrier.compareTo(((CarrierOriginDest)o).getC
arrier());
            if (result != 0) {
                return result;
            }
            result =
this.origin.compareTo(((CarrierOriginDest)o).getOr
igin());
            if (result != 0) {
                return result;
            }
            result =
this.destination.compareTo(((CarrierOriginDest)o).
getDestination());
            if (result != 0) {
                return result;
            }
            return  0;
        }
```

```
    }

    @Override
    public void write(DataOutput dataOutput)
throws IOException {
        dataOutput.writeUTF(carrier);
        dataOutput.writeUTF(origin);
        dataOutput.writeUTF(destination);
    }

    @Override
    public void readFields(DataInput dataInput)
throws IOException {
        carrier = dataInput.readUTF();
        origin = dataInput.readUTF();
        destination = dataInput.readUTF();
    }

    public String getCarrier() {
        return carrier;
    }
```

```
    public void setCarrier(String carrier) {
        this.carrier = carrier;
    }

    public String getOrigin() {
        return origin;
    }

    public void setOrigin(String origin) {
        this.origin = origin;
    }

    public String getDestination() {
        return destination;
    }

    public void setDestination(String destination)
{
        this.destination = destination;
    }
}
```

5.4 WritableComparables.OriginDestMonth

```
package WritableComparables;

import org.apache.hadoop.io.WritableComparable;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

public class OriginDestMonth implements
WritableComparable {
    private String origin;
    private String destination;
    private String month;

    public OriginDestMonth() {
        super();
    }

    public OriginDestMonth(String origin, String
destination, String month) {
        super();
```

```
        this.origin = origin;
        this.destination = destination;
        this.month = month;
    }

    @Override
    public String toString() {
        return origin + " " + destination + " " +
month;
    }

    @Override
    public int compareTo(Object o) {
        if (o.getClass() != this.getClass()) {
            return 0;
        } else {
            int result =
this.origin.compareTo(((OriginDestMonth)o).getOrig
in());
            if (result != 0) {
                return result;
            }
```

```
            result =
this.destination.compareTo(((OriginDestMonth)o).ge
tDestination());
            if (result != 0) {
                return result;
            }
            result =
this.month.compareTo(((OriginDestMonth)o).getMonth
());
            if (result != 0) {
                return result;
            }
            return  0;
        }
    }

    @Override
    public void write(DataOutput dataOutput)
throws IOException {
        dataOutput.writeUTF(origin);
        dataOutput.writeUTF(destination);
        dataOutput.writeUTF(month);
```

```
    }

    @Override
    public void readFields(DataInput dataInput)
throws IOException {
        origin =dataInput.readUTF();
        destination = dataInput.readUTF();
        month = dataInput.readUTF();
    }

    public String getOrigin() {
        return origin;
    }

    public void setOrigin(String origin) {
        this.origin = origin;
    }

    public String getDestination() {
        return destination;
    }
```

```
    public void setDestination(String destination)
{
        this.destination = destination;
    }

    public String getMonth() {
        return month;
    }

    public void setMonth(String month) {
        this.month = month;
    }
}
```

5.5 Comparators.CarrierMonthComparator

```
package Comparators;

import WritableComparables.CarrierMonth;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;

public class CarrierMonthComparator extends
```

```
WritableComparator {
    public CarrierMonthComparator() {
        super(CarrierMonth.class, true);
    }

    @Override
    public int compare(WritableComparable a,
WritableComparable b) {
        CarrierMonth cm1 = (CarrierMonth) a;
        CarrierMonth cm2 = (CarrierMonth) b;

        return cm1.compareTo(cm2);
    }
}
```

5.6 Comparators.CarrierOriginDestComparator

```
package Comparators;

import WritableComparables.CarrierOriginDest;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;
```

```
public class CarrierOriginDestComparator extends
WritableComparator {
    public CarrierOriginDestComparator() {
        super(CarrierOriginDest.class, true);
    }

    @Override
    public int compare(WritableComparable a,
WritableComparable b) {
        CarrierOriginDest c1 = (CarrierOriginDest)
a;
        CarrierOriginDest c2 = (CarrierOriginDest)
b;

        return c1.compareTo(c2);
    }
}
```

5.7 Comparators.OriginDestMonthComparator

```
package Comparators;

import WritableComparables.OriginDestMonth;
```

```
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;

public class OriginDestMonthComparator extends
WritableComparator {
    public OriginDestMonthComparator() {
        super(OriginDestMonth.class, true);
    }

    @Override
    public int compare(WritableComparable a,
WritableComparable b) {
        OriginDestMonth o1 = (OriginDestMonth) a;
        OriginDestMonth o2 = (OriginDestMonth) b;

        return o1.compareTo(o2);
    }
}
```

5.8 GroupComparators.CarrierMonthGroupComparator

```
package GroupComparators;
```

```
import WritableComparables.CarrierMonth;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;

public class CarrierMonthGroupComparator extends
WritableComparator {
    public CarrierMonthGroupComparator() {
        super(CarrierMonth.class, true);
    }

    @Override
    public int compare(WritableComparable a,
WritableComparable b) {
        CarrierMonth c1 = (CarrierMonth) a;
        CarrierMonth c2 = (CarrierMonth) b;

        return c1.compareTo(c2);
    }

    @Override
    public int compare(Object a, Object b) {
        CarrierMonth c1 = (CarrierMonth) a;
```

```
        CarrierMonth c2 = (CarrierMonth) b;

        return c1.compareTo(c2);
    }
}
```

5.9 GroupComparators.CarrierOriginDestGroupComparator

```
package GroupComparators;

import WritableComparables.CarrierOriginDest;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;

public class CarrierOriginDestGroupComparator
extends WritableComparator {
    public CarrierOriginDestGroupComparator() {
        super(CarrierOriginDest.class, true);
    }

    @Override
    public int compare(WritableComparable a,
WritableComparable b) {
```

```
        CarrierOriginDest c1 = (CarrierOriginDest)
a;
        CarrierOriginDest c2 = (CarrierOriginDest)
b;

        return c1.compareTo(c2);
    }

    @Override
    public int compare(Object a, Object b) {
        CarrierOriginDest c1 = (CarrierOriginDest)
a;
        CarrierOriginDest c2 = (CarrierOriginDest)
b;

        return c1.compareTo(c2);
    }
}
```

5.10 GroupComparators.OriginDestMonthGroupComparator

```
package GroupComparators;
```

```
import WritableComparables.OriginDestMonth;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;

public class OriginDestMonthGroupComparator
extends WritableComparator {
    public OriginDestMonthGroupComparator() {
        super(OriginDestMonth.class, true);
    }

    @Override
    public int compare(WritableComparable a,
WritableComparable b) {
        OriginDestMonth o1 = (OriginDestMonth) a;
        OriginDestMonth o2 = (OriginDestMonth) b;

        return o1.compareTo(o2);
    }

    @Override
    public int compare(Object a, Object b) {
        OriginDestMonth o1 = (OriginDestMonth) a;
```

```
        OriginDestMonth o2 = (OriginDestMonth) b;

        return o1.compareTo(o2);
    }
}
```

5.11 Partitioners.CarrierMonthPartitioner

```
package Partitioners;

import WritableComparables.CarrierMonth;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Partitioner;

public class CarrierMonthPartitioner extends
Partitioner<CarrierMonth, LongWritable> {
    @Override
    public int getPartition(CarrierMonth
carrierMonth, LongWritable longWritable, int i) {
        return
carrierMonth.getCarrier().hashCode() % i;
    }
}
```

5.12 Partitioners.CarrierOriginDestPartitioner

```
package Partitioners;

import WritableComparables.CarrierOriginDest;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Partitioner;

public class CarrierOriginDestPartitioner extends
Partitioner<CarrierOriginDest, LongWritable> {
    @Override
    public int getPartition(CarrierOriginDest
carrierOriginDest, LongWritable longWritable, int
i) {
        return
carrierOriginDest.getCarrier().hashCode() % i;
    }
}
```

5.13 Partitioners.OriginDestMonthPartitioner

```
package Partitioners;

import WritableComparables.OriginDestMonth;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Partitioner;

public class OriginDestMonthPartitioner extends
Partitioner<OriginDestMonth, LongWritable> {
    @Override
    public int getPartition(OriginDestMonth
originDestMonth, LongWritable longWritable, int i)
{
        return
originDestMonth.getOrigin().hashCode() % i;
    }
}
```

5.14 Mappers.CarrierArrDelayMapper

```
package Mappers;
```

```
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class CarrierArrDelayMapper extends
Mapper<LongWritable, Text, Text, DoubleWritable> {
    private Text carrierText = new Text();
    private DoubleWritable arrDelayDouble = new
DoubleWritable();

    @Override
    protected void map(LongWritable key, Text
value, Mapper<LongWritable, Text, Text,
DoubleWritable>.Context context) throws
IOException, InterruptedException {
        String[] flight_data =
value.toString().split(",");
        String carrier = flight_data[1];
```

```
        if (carrier.equals("")) {
            carrier = "unknown";
        }
        String arrDelayS = flight_data[14];
        if (arrDelayS.equals("")) {
            arrDelayS = "0.0";
        }
        double arrDelay =
Double.parseDouble(arrDelayS);
        carrierText.set(carrier);
        arrDelayDouble.set(arrDelay);

        context.write(carrierText, arrDelayDouble);
    }
}
```

5.15 Mappers.CarrierDepaDelayMapper

```
package Mappers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
```

```
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class CarrierDepaDelayMapper extends
Mapper<LongWritable, Text, Text, DoubleWritable> {
    private Text carrierText = new Text();
    private DoubleWritable depaDelayDouble = new
DoubleWritable();

    @Override
    protected void map(LongWritable key, Text
value, Mapper<LongWritable, Text, Text,
DoubleWritable>.Context context) throws
IOException, InterruptedException {
        String[] flight_data =
value.toString().split(",");
        String carrier = flight_data[1];
        if (carrier.equals("")) {
            carrier = "unknown";
        }
```

```
        String depaDelayS = flight_data[7];
        if (depaDelayS.equals("")) {
            depaDelayS = "0.0";
        }
        double depaDelay =
Double.parseDouble(depaDelayS);
        carrierText.set(carrier);
        depaDelayDouble.set(depaDelay);

        context.write(carrierText,
depaDelayDouble);
    }
}
```

5.16 Mappers.CarrierMonthArrDelayMapper

```
package Mappers;

import WritableComparables.CarrierMonth;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
```

```
import java.io.IOException;

public class CarrierMonthArrDelayMapper extends
Mapper<LongWritable, Text, CarrierMonth,
DoubleWritable> {
    private CarrierMonth cm = new CarrierMonth();
    private DoubleWritable arrDelayDouble = new
DoubleWritable();

    @Override
    protected void map(LongWritable key, Text
value, Mapper<LongWritable, Text, CarrierMonth,
DoubleWritable>.Context context) throws
IOException, InterruptedException {
        String[] flightRecord =
value.toString().split(",");
        String carrier = flightRecord[1];
        if (carrier.equals("")) {
            carrier = "unknown";
        }
        String date = flightRecord[0];
```

```
        if (date.equals("")) {
            date = "0000-00-00";
        }
        String[] dateArr = date.split("-");
        String monthS = dateArr[1];
        if (monthS.equals("")) {
            monthS = "0";
        }

        String arrDelayS = flightRecord[14];
        if (arrDelayS.equals("")) {
            arrDelayS = "0.0";
        }

        cm.setCarrier(carrier);
        cm.setMonth(monthS);


arrDelayDouble.set(Double.parseDouble(arrDelayS));
        context.write(cm, arrDelayDouble);
    }
}
```

5.17 Mappers.CarrierMonthDepaDelayMapper

```
package Mappers;

import WritableComparables.CarrierMonth;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class CarrierMonthDepaDelayMapper extends
Mapper<LongWritable, Text, CarrierMonth,
DoubleWritable> {
    private CarrierMonth cm = new CarrierMonth();
    private DoubleWritable depaDelayDouble = new
DoubleWritable();

    @Override
    protected void map(LongWritable key, Text
value, Mapper<LongWritable, Text, CarrierMonth,
```

```
DoubleWritable>.Context context) throws
IOException, InterruptedException {
        String[] flightRecord =
value.toString().split(",");
        String carrier = flightRecord[1];
        if (carrier.equals("")) {
            carrier = "unknown";
        }
        String date = flightRecord[0];
        if (date.equals("")) {
            date = "0000-00-00";
        }
        String[] dateArr = date.split("-");
        String monthS = dateArr[1];
        if (monthS.equals("")) {
            monthS = "0";
        }

        String depaDelayS = flightRecord[7];
        if (depaDelayS.equals("")) {
            depaDelayS = "0.0";
        }
```

```
        cm.setCarrier(carrier);
        cm.setMonth(monthS);


depaDelayDouble.set(Double.parseDouble(depaDelayS)
);
        context.write(cm, depaDelayDouble);
    }
}
```

5.18 Mappers.CarrierOriginDestArrDelayMapper

```
package Mappers;

import WritableComparables.CarrierOriginDest;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;
```

```
public class CarrierOriginDestArrDelayMapper
extends Mapper<LongWritable, Text,
CarrierOriginDest, DoubleWritable> {
    private CarrierOriginDest c = new
CarrierOriginDest();
    private DoubleWritable arrDelayDouble = new
DoubleWritable();

    @Override
    protected void map(LongWritable key, Text
value, Mapper<LongWritable, Text,
CarrierOriginDest, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        String[] flightRecord =
value.toString().split(",");
        String carrier = flightRecord[1];
        if (carrier.equals("")) {
            carrier = "unknown";
        }
        String origin = flightRecord[3];
        if (origin.equals("")) {
```

```
            origin = "unknown";
        }
        String destination = flightRecord[4];
        if (destination.equals("")) {
            destination = "unknown";
        }
        String arrDelayS = flightRecord[14];
        if (arrDelayS.equals("")) {
            arrDelayS = "0.0";
        }

        c.setCarrier(carrier);
        c.setOrigin(origin);
        c.setDestination(destination);


arrDelayDouble.set(Double.parseDouble(arrDelayS));
        context.write(c, arrDelayDouble);
    }
}
```

5.19 Mappers.CarrierOriginDestDepaDelayMapper

```
package Mappers;

import WritableComparables.CarrierOriginDest;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class CarrierOriginDestDepaDelayMapper
extends Mapper<LongWritable, Text,
CarrierOriginDest, DoubleWritable> {
    private CarrierOriginDest c = new
CarrierOriginDest();
    private DoubleWritable depaDelayDouble = new
DoubleWritable();

    @Override
    protected void map(LongWritable key, Text
value, Mapper<LongWritable, Text,
```

```
CarrierOriginDest, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        String[] flightRecord =
value.toString().split(",");
        String carrier = flightRecord[1];
        if (carrier.equals("")) {
            carrier = "unknown";
        }
        String origin = flightRecord[3];
        if (origin.equals("")) {
            origin = "unknown";
        }
        String destination = flightRecord[4];
        if (destination.equals("")) {
            destination = "unknown";
        }
        String depaDelayS = flightRecord[7];
        if (depaDelayS.equals("")) {
            depaDelayS = "0.0";
        }
```

```
        c.setCarrier(carrier);
        c.setOrigin(origin);
        c.setDestination(destination);


depaDelayDouble.set(Double.parseDouble(depaDelayS)
);
        context.write(c, depaDelayDouble);
    }
}
```

5.20 Mappers.OriginArrDelayMapper

```
package Mappers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;
```

```
public class OriginArrDelayMapper extends
Mapper<LongWritable, Text, Text, DoubleWritable> {
    private Text originText = new Text();
    private DoubleWritable arrDelayDouble = new
DoubleWritable();

    @Override
    protected void map(LongWritable key, Text
value, Mapper<LongWritable, Text, Text,
DoubleWritable>.Context context) throws
IOException, InterruptedException {
        String[] flight_data =
value.toString().split(",");
        String origin = flight_data[3];
        if (origin.equals("")) {
            origin = "unknown";
        }
        String arrDelayS = flight_data[14];
        if (arrDelayS.equals("")) {
            arrDelayS = "0.0";
        }
        double arrDelay =
```

```
Double.parseDouble(arrDelayS);
        originText.set(origin);
        arrDelayDouble.set(arrDelay);

        context.write(originText, arrDelayDouble);
    }
}
```

5.21 Mappers.OriginDepaDelayMapper

```
package Mappers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class OriginDepaDelayMapper extends
Mapper<LongWritable, Text, Text, DoubleWritable> {
    private Text originText = new Text();
```

```
    private DoubleWritable depaDelayDouble = new
DoubleWritable();

    @Override
    protected void map(LongWritable key, Text
value, Mapper<LongWritable, Text, Text,
DoubleWritable>.Context context) throws
IOException, InterruptedException {
        String[] flight_data =
value.toString().split(",");
        String origin = flight_data[3];
        if (origin.equals("")) {
            origin = "unknown";
        }
        String depaDelayS = flight_data[7];
        if (depaDelayS.equals("")) {
            depaDelayS = "0.0";
        }
        double depaDelay =
Double.parseDouble(depaDelayS);
        originText.set(origin);
        depaDelayDouble.set(depaDelay);
```

```
        context.write(originText, depaDelayDouble);
    }
}
```

5.22 Mappers.OriginDestMonthArrDelayMapper

```
package Mappers;

import WritableComparables.OriginDestMonth;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class OriginDestMonthArrDelayMapper extends
Mapper<LongWritable, Text, OriginDestMonth,
DoubleWritable> {
    private OriginDestMonth o = new
OriginDestMonth();
    private DoubleWritable arrDelayDouble = new
```

```
DoubleWritable();


    @Override
    protected void map(LongWritable key, Text
value, Mapper<LongWritable, Text, OriginDestMonth,
DoubleWritable>.Context context) throws
IOException, InterruptedException {
        String[] flightRecord =
value.toString().split(",");
        String origin = flightRecord[3];
        if (origin.equals("")) {
            origin = "unknown";
        }
        String destination = flightRecord[4];
        if (destination.equals("")) {
            destination = "unknown";
        }
        String arrDelayS = flightRecord[14];
        if (arrDelayS.equals("")) {
            arrDelayS = "0.0";
        }
        String date = flightRecord[0];
```

```
        if (date.equals("")) {
            date = "0000-00-00";
        }
        String[] dateArr = date.split("-");
        String monthS = dateArr[1];
        if (monthS.equals("")) {
            monthS = "0";
        }

        o.setOrigin(origin);
        o.setDestination(destination);
        o.setMonth(monthS);


arrDelayDouble.set(Double.parseDouble(arrDelayS));

        context.write(o, arrDelayDouble);

    }
}
```

5.23 Mappers.OriginDestMonthDepaDelayMapper

```
package Mappers;

import WritableComparables.OriginDestMonth;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class OriginDestMonthDepaDelayMapper
extends Mapper<LongWritable, Text,
OriginDestMonth, DoubleWritable> {
    private OriginDestMonth o = new
OriginDestMonth();
    private DoubleWritable depaDelayDouble = new
DoubleWritable();

    @Override
    protected void map(LongWritable key, Text
value, Mapper<LongWritable, Text, OriginDestMonth,
```

```
DoubleWritable>.Context context) throws
IOException, InterruptedException {
        String[] flightRecord =
value.toString().split(",");
        String origin = flightRecord[3];
        if (origin.equals("")) {
            origin = "unknown";
        }
        String destination = flightRecord[4];
        if (destination.equals("")) {
            destination = "unknown";
        }
        String depaDelayS = flightRecord[7];
        if (depaDelayS.equals("")) {
            depaDelayS = "0.0";
        }
        String date = flightRecord[0];
        if (date.equals("")) {
            date = "0000-00-00";
        }
        String[] dateArr = date.split("-");
        String monthS = dateArr[1];
```

```
        if (monthS.equals("")) {
            monthS = "0";
        }

        o.setOrigin(origin);
        o.setDestination(destination);
        o.setMonth(monthS);


depaDelayDouble.set(Double.parseDouble(depaDelayS)
);

        context.write(o, depaDelayDouble);

    }
}
```

5.24 Reducers.CarrierArrDelayAvgReducer

```
package Reducers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
```

```
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierArrDelayAvgReducer extends
Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private DoubleWritable avgArrDeplayDouble =
new DoubleWritable();
    @Override
    protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text,
DoubleWritable, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double avgArrDelay = 0.0;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            avgArrDelay += doubleWritable.get();
            count++;
        }
```

```
        avgArrDelay = avgArrDelay / count;
        avgArrDeplayDouble.set(avgArrDelay);
        context.write(key,avgArrDeplayDouble);
    }
}
```

5.25 Reducers.CarrierArrDelayMaxReducer

```
package Reducers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierArrDelayMaxReducer extends
Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private DoubleWritable maxArrDeplayDouble =
new DoubleWritable();
    @Override
    protected void reduce(Text key,
```

```
Iterable<DoubleWritable> values, Reducer<Text,
DoubleWritable, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double maxArrDelay = Double.MIN_VALUE;
//        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double curVal = doubleWritable.get();
            if (curVal > maxArrDelay) {
                maxArrDelay = curVal;
            }
        }
//        avgDepaDelay = avgDepaDelay / count;
        maxArrDeplayDouble.set(maxArrDelay);
        context.write(key,maxArrDeplayDouble);
    }
}
```

5.26 Reducers.CarrierArrDelayMinReducer

```
package Reducers;
```

```
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierArrDelayMinReducer extends
Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private DoubleWritable minArrDeplayDouble =
new DoubleWritable();
    @Override
    protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text,
DoubleWritable, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double minArrDelay = Double.MAX_VALUE;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double curVal = doubleWritable.get();
```

```
            if (curVal < minArrDelay) {
                minArrDelay = curVal;
            }
        }
        minArrDeplayDouble.set(minArrDelay);
        context.write(key,minArrDeplayDouble);
    }
}
```

5.27 Reducers.CarrierArrDelaySumReducer

```
package Reducers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierArrDelaySumReducer extends
Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private DoubleWritable sumArrDeplayDouble =
```

```
new DoubleWritable();
    @Override
    protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text,
DoubleWritable, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double sumArrDelay = 0.0;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            sumArrDelay += doubleWritable.get();
//            count++;
        }
//        avgDepaDelay = avgDepaDelay / count;
        sumArrDeplayDouble.set(sumArrDelay);
        context.write(key,sumArrDeplayDouble);
    }
}
```

5.28 Reducers.CarrierDepaDelayAvgReducer

```
package Reducers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierDepaDelayAvgReducer extends
Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private DoubleWritable avgDepaDeplayDouble =
new DoubleWritable();
    @Override
    protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text,
DoubleWritable, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double avgDepaDelay = 0.0;
        int count = 0;
```

```
        for (DoubleWritable doubleWritable :
values) {
            avgDepaDelay += doubleWritable.get();
            count++;
        }
        avgDepaDelay = avgDepaDelay / count;
        avgDepaDeplayDouble.set(avgDepaDelay);
        context.write(key,avgDepaDeplayDouble);
    }
}
```

5.29 Reducers.CarrierDepaDelayMaxReducer

```
package Reducers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierDepaDelayMaxReducer extends
Reducer<Text, DoubleWritable, Text,
```

```
DoubleWritable> {
    private DoubleWritable maxDepaDeplayDouble =
new DoubleWritable();
    @Override
    protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text,
DoubleWritable, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double maxDepaDelay = Double.MIN_VALUE;
//        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double curVal = doubleWritable.get();
            if (curVal > maxDepaDelay) {
                maxDepaDelay = curVal;
            }
        }
//        avgDepaDelay = avgDepaDelay / count;
        maxDepaDeplayDouble.set(maxDepaDelay);
        context.write(key,maxDepaDeplayDouble);
```

```
    }
}
```

5.30 Reducers.CarrierDepaDelayMinReducer

```
package Reducers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierDepaDelayMinReducer extends
Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private DoubleWritable minDepaDeplayDouble =
new DoubleWritable();
    @Override
    protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text,
DoubleWritable, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException
```

```
{
        double minDepaDelay = Double.MAX_VALUE;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double curVal = doubleWritable.get();
            if (curVal < minDepaDelay) {
                minDepaDelay = curVal;
            }
        }
        minDepaDeplayDouble.set(minDepaDelay);
        context.write(key,minDepaDeplayDouble);
    }
}
```

5.31 Reducers.CarrierDepaDelaySumReducer

```
package Reducers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
```

```
import java.io.IOException;

public class CarrierDepaDelaySumReducer extends
Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private DoubleWritable sumDepaDeplayDouble =
new DoubleWritable();
    @Override
    protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text,
DoubleWritable, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double sumDepaDelay = 0.0;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            sumDepaDelay += doubleWritable.get();
//            count++;
        }
//        avgDepaDelay = avgDepaDelay / count;
        sumDepaDeplayDouble.set(sumDepaDelay);
```

```
        context.write(key,sumDepaDeplayDouble);
    }
}
```

5.32 Reducers.CarrierMonthArrDelayAvgReducer

```
package Reducers;

import WritableComparables.CarrierMonth;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierMonthArrDelayAvgReducer
extends Reducer<CarrierMonth, DoubleWritable,
CarrierMonth, DoubleWritable> {
    private DoubleWritable avgArrDelayDouble = new
DoubleWritable();

    @Override
    protected void reduce(CarrierMonth key,
Iterable<DoubleWritable> values,
```

```
Reducer<CarrierMonth, DoubleWritable,
CarrierMonth, DoubleWritable>.Context context)
throws IOException, InterruptedException {
        double avgArrDelay = 0.0;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            avgArrDelay += doubleWritable.get();
            count++;
        }
        avgArrDelay = avgArrDelay / count;
        avgArrDelayDouble.set(avgArrDelay);
        context.write(key,avgArrDelayDouble);
    }
}
```

5.33 Reducers.CarrierMonthArrDelayMaxReducer

```
package Reducers;

import WritableComparables.CarrierMonth;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;
```

```
import java.io.IOException;

public class CarrierMonthArrDelayMaxReducer
extends Reducer<CarrierMonth, DoubleWritable,
CarrierMonth, DoubleWritable> {
    private DoubleWritable maxArrDelayDouble = new
DoubleWritable();

    @Override
    protected void reduce(CarrierMonth key,
Iterable<DoubleWritable> values,
Reducer<CarrierMonth, DoubleWritable,
CarrierMonth, DoubleWritable>.Context context)
throws IOException, InterruptedException {
        double maxArrDelay = Double.MIN_VALUE;
//        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double max = doubleWritable.get();
            if (max > maxArrDelay) {
                maxArrDelay = max;
```

```
            }
        }
//        avgDepaDelay = avgDepaDelay / count;
        maxArrDelayDouble.set(maxArrDelay);
        context.write(key,maxArrDelayDouble);
    }
}
```

5.34 Reducers.CarrierMonthArrDelayMinReducer

```
package Reducers;

import WritableComparables.CarrierMonth;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierMonthArrDelayMinReducer
extends Reducer<CarrierMonth, DoubleWritable,
CarrierMonth, DoubleWritable> {
    private DoubleWritable minArrDelayDouble = new
DoubleWritable();
```

```
    @Override
    protected void reduce(CarrierMonth key,
Iterable<DoubleWritable> values,
Reducer<CarrierMonth, DoubleWritable,
CarrierMonth, DoubleWritable>.Context context)
throws IOException, InterruptedException {
        double minArrDelay = Double.MAX_VALUE;
//        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double min = doubleWritable.get();
            if (min < minArrDelay) {
                minArrDelay = min;
            }
        }
//        avgDepaDelay = avgDepaDelay / count;
        minArrDelayDouble.set(minArrDelay);
        context.write(key,minArrDelayDouble);
    }
}
```

5.35 Reducers.CarrierMonthDepaDelayAvgReducer

```
package Reducers;

import WritableComparables.CarrierMonth;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierMonthDepaDelayAvgReducer
extends Reducer<CarrierMonth, DoubleWritable,
CarrierMonth, DoubleWritable> {
    private DoubleWritable avgDepaDelayDouble =
new DoubleWritable();

    @Override
    protected void reduce(CarrierMonth key,
Iterable<DoubleWritable> values,
Reducer<CarrierMonth, DoubleWritable,
CarrierMonth, DoubleWritable>.Context context)
throws IOException, InterruptedException {
        double avgDepaDelay = 0.0;
```

```
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            avgDepaDelay += doubleWritable.get();
            count++;
        }
        avgDepaDelay = avgDepaDelay / count;
        avgDepaDelayDouble.set(avgDepaDelay);
        context.write(key,avgDepaDelayDouble);
    }
}
```

5.36 Reducers.CarrierMonthDepaDelayMaxReducer

```
package Reducers;

import WritableComparables.CarrierMonth;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierMonthDepaDelayMaxReducer
```

```
extends Reducer<CarrierMonth, DoubleWritable,
CarrierMonth, DoubleWritable> {
    private DoubleWritable maxDepaDelayDouble =
new DoubleWritable();

    @Override
    protected void reduce(CarrierMonth key,
Iterable<DoubleWritable> values,
Reducer<CarrierMonth, DoubleWritable,
CarrierMonth, DoubleWritable>.Context context)
throws IOException, InterruptedException {
        double maxDepaDelay = Double.MIN_VALUE;
//        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double max = doubleWritable.get();
            if (max > maxDepaDelay) {
                maxDepaDelay = max;
            }
        }
//        avgDepaDelay = avgDepaDelay / count;
        maxDepaDelayDouble.set(maxDepaDelay);
```

```
        context.write(key,maxDepaDelayDouble);
    }
}
```

5.37 Reducers.CarrierMonthDepaDelayMinReducer

```
package Reducers;

import WritableComparables.CarrierMonth;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierMonthDepaDelayMinReducer
extends Reducer<CarrierMonth, DoubleWritable,
CarrierMonth, DoubleWritable> {
    private DoubleWritable minDepaDelayDouble =
new DoubleWritable();

    @Override
    protected void reduce(CarrierMonth key,
Iterable<DoubleWritable> values,
```

```
Reducer<CarrierMonth, DoubleWritable,
CarrierMonth, DoubleWritable>.Context context)
throws IOException, InterruptedException {
        double minDepaDelay = Double.MAX_VALUE;
//        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double min = doubleWritable.get();
            if (min < minDepaDelay) {
                minDepaDelay = min;
            }
        }
//        avgDepaDelay = avgDepaDelay / count;
        minDepaDelayDouble.set(minDepaDelay);
        context.write(key,minDepaDelayDouble);
    }
}
```

5.38 Reducers.CarrierOriginDestArrDelayAvgReducer

```
package Reducers;

import WritableComparables.CarrierOriginDest;
```

```
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierOriginDestArrDelayAvgReducer
extends Reducer<CarrierOriginDest, DoubleWritable,
CarrierOriginDest, DoubleWritable> {
    private DoubleWritable avgArrDelayDouble = new
DoubleWritable();

    @Override
    protected void reduce(CarrierOriginDest key,
Iterable<DoubleWritable> values,
Reducer<CarrierOriginDest, DoubleWritable,
CarrierOriginDest, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double avgArrDelay = 0.0;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
```

```
            avgArrDelay += doubleWritable.get();
            count++;
        }
        avgArrDelay = avgArrDelay / count;
        avgArrDelayDouble.set(avgArrDelay);
        context.write(key, avgArrDelayDouble);
    }
}
```

5.39 Reducers.CarrierOriginDestArrDelayMaxReducer

```
package Reducers;

import WritableComparables.CarrierOriginDest;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierOriginDestArrDelayMaxReducer
extends Reducer<CarrierOriginDest, DoubleWritable,
CarrierOriginDest, DoubleWritable> {
    private DoubleWritable maxArrDelayDouble = new
```

```
DoubleWritable();


    @Override
    protected void reduce(CarrierOriginDest key,
Iterable<DoubleWritable> values,
Reducer<CarrierOriginDest, DoubleWritable,
CarrierOriginDest, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double maxArrDelay = Double.MIN_VALUE;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double max = doubleWritable.get();
            if (max > maxArrDelay) {
                maxArrDelay = max;
            }
        }
//        avgDepaDelay = avgDepaDelay / count;
        maxArrDelayDouble.set(maxArrDelay);
        context.write(key, maxArrDelayDouble);
```

```
    }
}
```

5.40 Reducers.CarrierOriginDestArrDelayMinReducer

```
package Reducers;

import WritableComparables.CarrierOriginDest;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierOriginDestArrDelayMinReducer
extends Reducer<CarrierOriginDest, DoubleWritable,
CarrierOriginDest, DoubleWritable> {
    private DoubleWritable minArrDelayDouble = new
DoubleWritable();

    @Override
    protected void reduce(CarrierOriginDest key,
Iterable<DoubleWritable> values,
Reducer<CarrierOriginDest, DoubleWritable,
```

```
CarrierOriginDest, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double minArrDelay = Double.MAX_VALUE;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double min = doubleWritable.get();
            if (min < minArrDelay) {
                minArrDelay = min;
            }
        }
//        avgDepaDelay = avgDepaDelay / count;
        minArrDelayDouble.set(minArrDelay);
        context.write(key, minArrDelayDouble);
    }
}
```

5.41 Reducers.CarrierOriginDestDepaDelayAvgReducer

```
package Reducers;

import WritableComparables.CarrierOriginDest;
```

```
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierOriginDestDepaDelayAvgReducer
extends Reducer<CarrierOriginDest, DoubleWritable,
CarrierOriginDest, DoubleWritable> {
    private DoubleWritable avgDepaDelayDouble =
new DoubleWritable();

    @Override
    protected void reduce(CarrierOriginDest key,
Iterable<DoubleWritable> values,
Reducer<CarrierOriginDest, DoubleWritable,
CarrierOriginDest, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double avgDepaDelay = 0.0;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
```

```
            avgDepaDelay += doubleWritable.get();
            count++;
        }
        avgDepaDelay = avgDepaDelay / count;
        avgDepaDelayDouble.set(avgDepaDelay);
        context.write(key, avgDepaDelayDouble);
    }
}
```

5.42 Reducers.CarrierOriginDestDepaDelayMaxReducer

```
package Reducers;

import WritableComparables.CarrierOriginDest;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierOriginDestDepaDelayMaxReducer
extends Reducer<CarrierOriginDest, DoubleWritable,
CarrierOriginDest, DoubleWritable> {
    private DoubleWritable maxDepaDelayDouble =
```

```
new DoubleWritable();


    @Override
    protected void reduce(CarrierOriginDest key,
Iterable<DoubleWritable> values,
Reducer<CarrierOriginDest, DoubleWritable,
CarrierOriginDest, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double maxDepaDelay = Double.MIN_VALUE;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double max = doubleWritable.get();
            if (max > maxDepaDelay) {
                maxDepaDelay = max;
            }
        }
//        avgDepaDelay = avgDepaDelay / count;
        maxDepaDelayDouble.set(maxDepaDelay);
        context.write(key, maxDepaDelayDouble);
```

```
    }
}
```

5.43 Reducers.CarrierOriginDestDepaDelayMinReducer

```
package Reducers;

import WritableComparables.CarrierOriginDest;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CarrierOriginDestDepaDelayMinReducer
extends Reducer<CarrierOriginDest, DoubleWritable,
CarrierOriginDest, DoubleWritable> {
    private DoubleWritable minDepaDelayDouble =
new DoubleWritable();

    @Override
    protected void reduce(CarrierOriginDest key,
Iterable<DoubleWritable> values,
Reducer<CarrierOriginDest, DoubleWritable,
```

```
CarrierOriginDest, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double minDepaDelay = Double.MAX_VALUE;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double min = doubleWritable.get();
            if (min < minDepaDelay) {
                minDepaDelay = min;
            }
        }
//        avgDepaDelay = avgDepaDelay / count;
        minDepaDelayDouble.set(minDepaDelay);
        context.write(key, minDepaDelayDouble);
    }
}
```

5.44 Reducers.OriginArrDelayAvgReducer

```
package Reducers;

import org.apache.hadoop.io.DoubleWritable;
```

```
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class OriginArrDelayAvgReducer extends
Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private DoubleWritable avgArrDeplayDouble =
new DoubleWritable();
    @Override
    protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text,
DoubleWritable, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double avgArrDelay = 0.0;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            avgArrDelay += doubleWritable.get();
            count++;
```

```
        }
        avgArrDelay = avgArrDelay / count;
        avgArrDeplayDouble.set(avgArrDelay);
        context.write(key,avgArrDeplayDouble);
    }
}
```

5.45 Reducers.OriginArrDelayMaxReducer

```
package Reducers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class OriginArrDelayMaxReducer extends
Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private DoubleWritable maxArrDeplayDouble =
new DoubleWritable();
    @Override
```

```
    protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text,
DoubleWritable, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double maxArrDelay = Double.MIN_VALUE;
//        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double curVal = doubleWritable.get();
            if (curVal > maxArrDelay) {
                maxArrDelay = curVal;
            }
        }
//        avgDepaDelay = avgDepaDelay / count;
        maxArrDeplayDouble.set(maxArrDelay);
        context.write(key,maxArrDeplayDouble);
    }
}
```

5.46 Reducers.OriginArrDelayMinReducer

```
package Reducers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class OriginArrDelayMinReducer extends
Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private DoubleWritable minArrDeplayDouble =
new DoubleWritable();
    @Override
    protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text,
DoubleWritable, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double minArrDelay = Double.MAX_VALUE;
        int count = 0;
```

```
        for (DoubleWritable doubleWritable :
values) {
            double curVal = doubleWritable.get();
            if (curVal < minArrDelay) {
                minArrDelay = curVal;
            }
        }
        minArrDeplayDouble.set(minArrDelay);
        context.write(key,minArrDeplayDouble);
    }
}
```

5.47 Reducers.OriginArrDelaySumReducer

```
package Reducers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class OriginArrDelaySumReducer extends
```

```
Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private DoubleWritable sumArrDeplayDouble =
new DoubleWritable();
    @Override
    protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text,
DoubleWritable, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double sumArrDelay = 0.0;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            sumArrDelay += doubleWritable.get();
//            count++;
        }
//        avgDepaDelay = avgDepaDelay / count;
        sumArrDeplayDouble.set(sumArrDelay);
        context.write(key,sumArrDeplayDouble);
    }
}
```

5.48 Reducers.OriginDepaDelayAvgReducer

```
package Reducers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class OriginDepaDelayAvgReducer extends
Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private DoubleWritable avgDepaDeplayDouble =
new DoubleWritable();
    @Override
    protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text,
DoubleWritable, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double avgDepaDelay = 0.0;
```

```
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            avgDepaDelay += doubleWritable.get();
            count++;
        }
        avgDepaDelay = avgDepaDelay / count;
        avgDepaDeplayDouble.set(avgDepaDelay);
        context.write(key,avgDepaDeplayDouble);
    }
}
```

5.49 Reducers.OriginDepaDelayMaxReducer

```
package Reducers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class OriginDepaDelayMaxReducer extends
```

```
Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private DoubleWritable maxDepaDeplayDouble =
new DoubleWritable();
    @Override
    protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text,
DoubleWritable, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double maxDepaDelay = Double.MIN_VALUE;
//        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double curVal = doubleWritable.get();
            if (curVal > maxDepaDelay) {
                maxDepaDelay = curVal;
            }
        }
//        avgDepaDelay = avgDepaDelay / count;
        maxDepaDeplayDouble.set(maxDepaDelay);
        context.write(key,maxDepaDeplayDouble);
```

```
    }
}
```

5.50 Reducers.OriginDepaDelayMinReducer

```
package Reducers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class OriginDepaDelayMinReducer extends
Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private DoubleWritable minDepaDeplayDouble =
new DoubleWritable();
    @Override
    protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text,
DoubleWritable, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException
```

```
{
        double minDepaDelay = Double.MAX_VALUE;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double curVal = doubleWritable.get();
            if (curVal < minDepaDelay) {
                minDepaDelay = curVal;
            }
        }
        minDepaDeplayDouble.set(minDepaDelay);
        context.write(key,minDepaDeplayDouble);
    }
}
```

5.51 Reducers.OriginDepaDelaySumReducer

```
package Reducers;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
```

```
import java.io.IOException;

public class OriginDepaDelaySumReducer extends
Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private DoubleWritable sumDepaDeplayDouble =
new DoubleWritable();
    @Override
    protected void reduce(Text key,
Iterable<DoubleWritable> values, Reducer<Text,
DoubleWritable, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException
{
        double sumDepaDelay = 0.0;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            sumDepaDelay += doubleWritable.get();
//            count++;
        }
//        avgDepaDelay = avgDepaDelay / count;
        sumDepaDeplayDouble.set(sumDepaDelay);
```

```
        context.write(key,sumDepaDeplayDouble);
    }
}
```

5.52 Reducers.OriginDestMonthArrDelayAvgReducer

```
package Reducers;

import WritableComparables.OriginDestMonth;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class OriginDestMonthArrDelayAvgReducer
extends Reducer<OriginDestMonth, DoubleWritable,
OriginDestMonth, DoubleWritable> {
    private DoubleWritable avgArrDeplayDouble =
new DoubleWritable();

    @Override
    protected void reduce(OriginDestMonth key,
Iterable<DoubleWritable> values,
```

```
Reducer<OriginDestMonth, DoubleWritable,
OriginDestMonth, DoubleWritable>.Context context)
throws IOException, InterruptedException {
        double avgArrDelay = 0.0;
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            avgArrDelay += doubleWritable.get();
            count++;
        }
        avgArrDelay = avgArrDelay / count;
        avgArrDeplayDouble.set(avgArrDelay);
        context.write(key,avgArrDeplayDouble);
    }
}
```

5.53 Reducers.OriginDestMonthArrDelayMaxReducer

```
package Reducers;

import WritableComparables.OriginDestMonth;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;
```

```
import java.io.IOException;

public class OriginDestMonthArrDelayMaxReducer
extends Reducer<OriginDestMonth, DoubleWritable,
OriginDestMonth, DoubleWritable> {
    private DoubleWritable maxArrDeplayDouble =
new DoubleWritable();

    @Override
    protected void reduce(OriginDestMonth key,
Iterable<DoubleWritable> values,
Reducer<OriginDestMonth, DoubleWritable,
OriginDestMonth, DoubleWritable>.Context context)
throws IOException, InterruptedException {
        double maxArrDelay = Double.MIN_VALUE;
//        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double max = doubleWritable.get();
            if (max > maxArrDelay) {
                maxArrDelay = max;
```

```
            }
        }
//        avgDepaDelay = avgDepaDelay / count;
        maxArrDeplayDouble.set(maxArrDelay);
        context.write(key,maxArrDeplayDouble);
    }
}
```

5.54 Reducers.OriginDestMonthArrDelayMinReducer

```
package Reducers;

import WritableComparables.OriginDestMonth;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class OriginDestMonthArrDelayMinReducer
extends Reducer<OriginDestMonth, DoubleWritable,
OriginDestMonth, DoubleWritable> {
    private DoubleWritable minArrDeplayDouble =
new DoubleWritable();
```

```
    @Override
    protected void reduce(OriginDestMonth key,
Iterable<DoubleWritable> values,
Reducer<OriginDestMonth, DoubleWritable,
OriginDestMonth, DoubleWritable>.Context context)
throws IOException, InterruptedException {
        double minArrDelay = Double.MAX_VALUE;
//        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double min = doubleWritable.get();
            if (min < minArrDelay) {
                minArrDelay = min;
            }
        }
//        avgDepaDelay = avgDepaDelay / count;
        minArrDeplayDouble.set(minArrDelay);
        context.write(key,minArrDeplayDouble);
    }
}
```

5.55 Reducers.OriginDestMonthDepaDelayAvgReducer

```
package Reducers;

import WritableComparables.OriginDestMonth;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class OriginDestMonthDepaDelayAvgReducer
extends Reducer<OriginDestMonth, DoubleWritable,
OriginDestMonth, DoubleWritable> {
    private DoubleWritable avgDepaDeplayDouble =
new DoubleWritable();

    @Override
    protected void reduce(OriginDestMonth key,
Iterable<DoubleWritable> values,
Reducer<OriginDestMonth, DoubleWritable,
OriginDestMonth, DoubleWritable>.Context context)
throws IOException, InterruptedException {
        double avgDepaDelay = 0.0;
```

```
        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            avgDepaDelay += doubleWritable.get();
            count++;
        }
        avgDepaDelay = avgDepaDelay / count;
        avgDepaDeplayDouble.set(avgDepaDelay);
        context.write(key,avgDepaDeplayDouble);
    }
}
```

5.56 Reducers.OriginDestMonthDepaDelayMaxReducer

```
package Reducers;

import WritableComparables.OriginDestMonth;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class OriginDestMonthDepaDelayMaxReducer
```

```
extends Reducer<OriginDestMonth, DoubleWritable,
OriginDestMonth, DoubleWritable> {
    private DoubleWritable maxDepaDeplayDouble =
new DoubleWritable();

    @Override
    protected void reduce(OriginDestMonth key,
Iterable<DoubleWritable> values,
Reducer<OriginDestMonth, DoubleWritable,
OriginDestMonth, DoubleWritable>.Context context)
throws IOException, InterruptedException {
        double maxDepaDelay = Double.MIN_VALUE;
//        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double max = doubleWritable.get();
            if (max > maxDepaDelay) {
                maxDepaDelay = max;
            }
        }
//        avgDepaDelay = avgDepaDelay / count;
        maxDepaDeplayDouble.set(maxDepaDelay);
```

```
        context.write(key,maxDepaDeplayDouble);
    }
}
```

5.57 Reducers.OriginDestMonthDepaDelayMinReducer

```
package Reducers;

import WritableComparables.OriginDestMonth;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class OriginDestMonthDepaDelayMinReducer
extends Reducer<OriginDestMonth, DoubleWritable,
OriginDestMonth, DoubleWritable> {
    private DoubleWritable minDepaDeplayDouble =
new DoubleWritable();

    @Override
    protected void reduce(OriginDestMonth key,
Iterable<DoubleWritable> values,
```

```
Reducer<OriginDestMonth, DoubleWritable,
OriginDestMonth, DoubleWritable>.Context context)
throws IOException, InterruptedException {
        double minDepaDelay = Double.MAX_VALUE;
//        int count = 0;
        for (DoubleWritable doubleWritable :
values) {
            double min = doubleWritable.get();
            if (min < minDepaDelay) {
                minDepaDelay = min;
            }
        }
//        avgDepaDelay = avgDepaDelay / count;
        minDepaDeplayDouble.set(minDepaDelay);
        context.write(key,minDepaDeplayDouble);
    }
}
```