CS432/532: Final Project Report

Project Title: Flight delay and cancellation data analysis

Team Member(s): 1) Mohiuddin, Mohammed Amer (B00929336)
2) Nageshwarrao Chitturi, Chethan Kumar (B00963787)
3) Nyamathulla, Shaik (B00962641)

I. Problem

According to a 2010 analysis by the US Federal Aviation Administration, the economic cost of domestic flight delays is $32.9 billion per year to passengers, airlines, and other sectors of the economy. More than half of that sum comes from the purses of passengers, who not only waste time waiting for their planes to depart, but also miss connecting flights, spend money on food, and sleep in hotel rooms while stranded.

To provide a solution to this problem, we are going to analyze a data set containing up to 6.2M records of different internal flights in the US and their causes for delay, diversion, and cancellations if any

- Analyzing the flight dataset to know when the best time of day/day of week/ time of year is to fly to minimize delays.
- Based on the age of the planes we can infer what type of planes suffer the most delays.
- To know how often does a delay of plane result in the delay of subsequent connecting planes.
- Analysis of flight delays based on location, day and carriers.
- To co-relate how the frequency of airlines affecting the delays.

II. Software Design and Implementation

A. *Software Design and NoSQL-Databse and Tools Used*

1. MongoDB:
2. PyMongo
3. python

B. *Parts that you have implemented.*

- Best day of the week to travel with minimum arrival delays.
  - Gets day of the week with minimal delays, this is special because of the compound index.
  - Returns first n states with the most incoming/outgoing flights, n is limited to 10 due to the huge data set.
  - Used getCompoundId() to retrieve best day of the week.
- States with most flights leaving/arriving:
  - Returns the first n sates with the most incoming/outgoing flights.
  - Used getSatesByflights() to return destSateId's of incoming/outgoing flights.
- States with most departure/ arrival delays
  - Returns sates with most departure/arrival delays.
  - Returns the first n states grouped by aggregateId.
  - Used getmostfrequentattr() to retrieve origStateId, depDelay, arrDelay and avg.
  - Ordered in descending order of depDelay/arrDelay.
- Age factor in most delayed flights.
  - Considers the age of the carrier and returns the most delayed flights.
  - Returns the first n most delayed flights, n is limited to 10.
  - Sorts first by arrDelay since departure delays are the most common.
  - Returns id, arrDelay, depDelay, carrier, origCity, destiCity and age.
- Delays from subsequent connecting flights.
  - using findNumCascDelays, crsArrTime, crsDepTime and tailNum.
  - crsArrTime - > scheduled arrival time for delayed aircrafts.
  - crsDeptime -> scheduled departure time for delayed flights.
  - tailNum -> tail number of aircraft delayed.
  - Analyzing delays caused in connecting flights due to late arrival of first flight

III. Project Outcome

- Best day of the week to travel with minimum arrival delays.

  - Code:

```
def getCompoundId(aggregateValue="arrDelay", agg="$avg", n=10):
    return flights.aggregate([
        {"$group" : {"_id":
            {"w" : {"$dayOfWeek" : "$date"},
            "h" : {"$hour" : "$date"}},
            "delay" : {agg : "$"+aggregateValue}}}
        , {"$sort": {"delay" : 1}}
        , {"$limit" : n}
        ])
```

  - Output:

```
===TIME OF WEEK WITH LEAST ARRIVAL DELAYS==============
{'_id': {'h': 0, 'w': 7}, 'delay': 1.2793364529422846}
{'_id': {'h': 0, 'w': 3}, 'delay': 3.00425384359354 7}
{'_id': {'h': 0, 'w': 4}, 'delay': 3.0860794338289903}
{'_id': {'h': 0, 'w': 1}, 'delay': 4.091053407677066}
{'_id': {'h': 0, 'w': 2}, 'delay': 4.838740833922768}
{'_id': {'h': 0, 'w': 5}, 'delay': 4.918444394760784}
{'_id': {'h': 0, 'w': 6}, 'delay': 5.526180216845462}
```

- States with most departure/ arrival delays

  - Code:

```
def getmostfrequentattr(aggregateId="origStateId", aggregateValue="depDelay", agg="$avg", n=10):
    return flights.aggregate([
        {"$group" : {"_id": "$"+aggregateId,
                     "delay" : {agg : "$"+aggregateValue}}}
        , { "$sort" : {"delay" : 1} }
        , { "$limit" : n}
    ])
```

  - Output:

```
===MOST DEPARTURE DELAY STATES====================
{'_id': 'MT', 'delay': 0.5942693751116935}
{'_id': 'AK', 'delay': 0.9560849218729345}
{'_id': 'WY', 'delay': 2.1376595948929635}
{'_id': 'HI', 'delay': 2.4874956128904526}
{'_id': 'ID', 'delay': 3.321040610550944}
{'_id': 'UT', 'delay': 3.7036838962127088}
{'_id': 'VI', 'delay': 4.087665869045029}
{'_id': 'WA', 'delay': 4.736326068296655}
{'_id': 'MN', 'delay': 4.901339711372194}
{'_id': 'AZ', 'delay': 4.9772072410279415}
===MOST ARRIVAL DELAY STATES=====================
{'_id': 'MN', 'delay': -0.5293655104208043}
{'_id': 'AK', 'delay': -0.4693459584513196}
{'_id': 'AZ', 'delay': -0.19705455096820265}
{'_id': 'UT', 'delay': -0.044248183943462645}
{'_id': 'WA', 'delay': 0.09875470861954354}
{'_id': 'MT', 'delay': 1.3378813155386082}
{'_id': 'MI', 'delay': 1.5075758727071018}
{'_id': 'NV', 'delay': 1.5475549362607566}
{'_id': 'NC', 'delay': 1.9154802327197633}
{'_id': 'OR', 'delay': 2.224786532559482}
```

- States with most flights leaving/arriving:

  - Code:

```
def getStatesByFlights(aggregateId="origStateId", n=10):
    return flights.aggregate([
        {"$group" : {"_id" : "$"+aggregateId,
                     "numFlights" : {"$sum" : 1}}}
        , {"$sort" : {"numFlights" : -1}}
        , {"$limit" : n}
        ])
```

  - Output:

```
===MOST FLIGHTS LEAVING STATES==================
{'_id': 'CA', 'numFlights': 760113}
{'_id': 'TX', 'numFlights': 725486}
{'_id': 'FL', 'numFlights': 446024}
{'_id': 'GA', 'numFlights': 411166}
{'_id': 'IL', 'numFlights': 410492}
{'_id': 'NY', 'numFlights': 268986}
{'_id': 'CO', 'numFlights': 260408}
{'_id': 'AZ', 'numFlights': 206263}
{'_id': 'NC', 'numFlights': 204447}
{'_id': 'VA', 'numFlights': 188618}
===MOST INCOMING FLIGHTS STATES=================
{'_id': 'CA', 'numFlights': 760142}
{'_id': 'TX', 'numFlights': 725682}
{'_id': 'FL', 'numFlights': 446050}
{'_id': 'GA', 'numFlights': 411184}
{'_id': 'IL', 'numFlights': 410481}
{'_id': 'NY', 'numFlights': 268975}
{'_id': 'CO', 'numFlights': 260300}
{'_id': 'AZ', 'numFlights': 206278}
{'_id': 'NC', 'numFlights': 204439}
{'_id': 'VA', 'numFlights': 188657}
```

- Age factor in most delayed flights.

  - Code:

```
def mostdelayed(n, withages=False):

    q = {"age" : {"$exists" : True}} if withages else {}
    return flights.find(q,
                        {"_id":1
                        ,"arrDelay":1
                        , "depDelay":1
                        , "carrier":1
                        , "origCity":1
                        , "destCity":1
                        , "age": 1}).sort([("arrDelay", -1)]).sort([("depDelay", -1)]).limit(n)
```

  - Output:

```
======AGE FACTOR IN MOST DELAYED FLIGHTS====
{'_id': ObjectId('51bf22d2ca69141e42f66a3c'),
 'age': 26,
 'arrDelay': 1591,
 'carrier': 'AA',
 'depDelay': 1621,
 'destCity': 'Dallas/Fort Worth, TX',
 'origCity': 'Milwaukee, WI'}
{'_id': ObjectId('51bf22fbca69141e42f6d686'),
 'age': 33,
 'arrDelay': 1627,
 'carrier': 'AA',
 'depDelay': 1604,
 'destCity': 'Dallas/Fort Worth, TX',
 'origCity': 'Reno, NV'}
```

```
{'_id': ObjectId('51bf28c0ca69141e420683d9'),
 'age': 25,
 'arrDelay': 1539,
 'carrier': 'AA',
 'depDelay': 1554,
 'destCity': 'Dallas/Fort Worth, TX',
 'origCity': 'Columbus, OH'}
{'_id': ObjectId('51bf3700ca69141e422c650f'),
 'age': 28,
 'arrDelay': 1543,
 'carrier': 'AA',
 'depDelay': 1542,
 'destCity': 'Dallas/Fort Worth, TX',
 'origCity': 'Tucson, AZ'}
{'_id': ObjectId('51bf25b3ca69141e42fe3ebc'),
 'age': 22,
 'arrDelay': 1525,
 'carrier': 'AA',
 'depDelay': 1533,
 'destCity': 'Dallas/Fort Worth, TX',
 'origCity': 'Orlando, FL'}
```


- Delays from subsequent connecting flights.

  - Code:

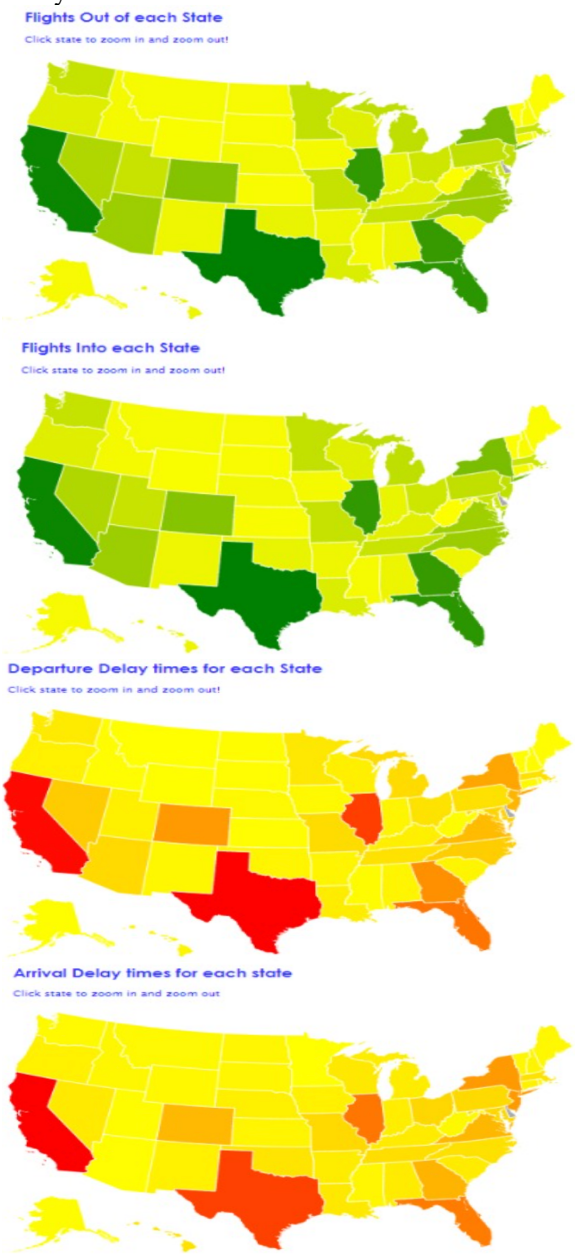
```
def findNumCascDelays(tailNum, crsArrTime):
    after = collection.find({"tailNum" : tailNum,
        "crsDepTime" : {"$gt" : crsArrTime}}).sort([("crsDepTime", 1)])
    cnt = collection.count_documents({"tailNum" : tailNum,
        "crsDepTime" : {"$gt" : crsArrTime}})
    if cnt == 0:
        return 0
    else:
        num = 0

        for trip in after:
            if ("depDelay" in trip and "arrDelay" in trip \
                    and "lateAircraftDelay" in trip ):
                if (trip["depDelay"] > 0 and trip["arrDelay"] > 0 \
                    and trip["lateAircraftDelay"] > 0):
                        num += 1
                        continue
            break
    return num
```

  - Output:

```
===DELAYS FROM SUBSEQUENT CONNECTING FLIGHTS==========
10000 late arrival flights have caused 665 delays
20000 late arrival flights have caused 1541 delays
30000 late arrival flights have caused 2438 delays
40000 late arrival flights have caused 3354 delays
50000 late arrival flights have caused 4505 delays
60000 late arrival flights have caused 5933 delays
70000 late arrival flights have caused 7637 delays
80000 late arrival flights have caused 10138 delays
90000 late arrival flights have caused 13957 delays
100000 late arrival flights have caused 20030 delays
```

- Analysis:



REFERENCES

[1] https://www.mongodb.com/blog/post/aggregation-options-on-big-data-sets-part-1-basic

[2] https://www.kaggle.com/code/adveros/flight-delay-eda-exploratory-data-analysis/notebook

[3] https://www.kaggle.com/datasets/usdot/flight-delays

[4] https://www.kaggle.com/datasets/sherrytp/airline-delay-analysis