<u>CSCI 620 Project Phase 3</u>

Group 33

Flight Delay

Shlok Gupta, Suraj Sureshkumar, Vidit Ketan Kothari

Table of Contents

Cleaning and Integration

For cleaning the dataset, we used sql and mongo queries to remove duplicate values. On checking for missing values, we found that there are no missing values in our dataset.

SQL

SQL cleaning was a simple drop duplicates query.

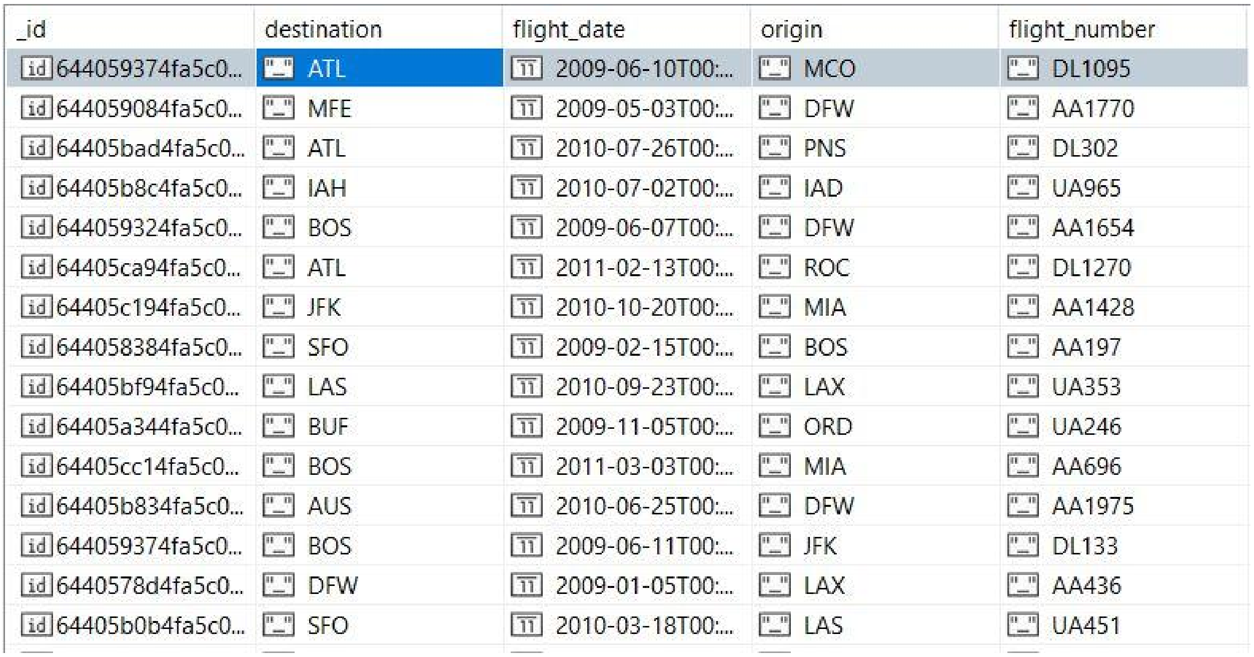
Mongo

We concatenated the carrier name and the flight number to create a unique identifier for each flight.

```
db.delays.aggregate([

  {

    $match:

      {

        $and: [

          { departure_delay: { $gt: 30 } },

          { arrival_delay: { $gt: 30 } },

          { OP_CARRIER: { $in: ['AA', 'DL', 'UA'] } }

        ]

      }

  },

  {

    $project: {

      OP_CARRIER: 1,
      flight_date: 1,
      origin: 1,
      destination: 1,
      OP_CARRIER_FL_NUM: { $substr: ["$OP_CARRIER_FL_NUM", 0, -1] }
    }
  },
  {
    $project: {
      flight_number: {
        $concat: ["$OP_CARRIER", "$OP_CARRIER_FL_NUM"]
      },
      flight_date: 1,
      origin: 1,
      destination: 1
    }
  },
  { $out: "major_airlines" }
])
```

```
db.major_airlines.aggregate([
    {
        $group: {
            _id: { destination: "$destination", flight_date: "$flight_date", origin: "$origin", flight_number: "$flight_number" },
            uniqueIds: { $addToSet: "$_id" },
            doc: { $first: "$$ROOT" }
        }
    },
    {
        $replaceRoot: {
            newRoot: "$doc"
        }
    }
], { allowDiskUse: true }).forEach(function (doc) {
    db.temp.insertOne(doc);
});
```

Output:



_id	destination	flight_date	origin	flight_number
644059374fa5c0...	ATL	2009-06-10T00:...	MCO	DL1095
644059084fa5c0...	MFE	2009-05-03T00:...	DFW	AA1770
64405bad4fa5c0...	ATL	2010-07-26T00:...	PNS	DL302
64405b8c4fa5c0...	IAH	2010-07-02T00:...	IAD	UA965
644059324fa5c0...	BOS	2009-06-07T00:...	DFW	AA1654
64405ca94fa5c0...	ATL	2011-02-13T00:...	ROC	DL1270
64405c194fa5c0...	JFK	2010-10-20T00:...	MIA	AA1428
644058384fa5c0...	SFO	2009-02-15T00:...	BOS	AA197
64405bf94fa5c0...	LAS	2010-09-23T00:...	LAX	UA353
64405a344fa5c0...	BUF	2009-11-05T00:...	ORD	UA246
64405cc14fa5c0...	BOS	2011-03-03T00:...	MIA	AA696
64405b834fa5c0...	AUS	2010-06-25T00:...	DFW	AA1975
644059374fa5c0...	BOS	2009-06-11T00:...	JFK	DL133
6440578d4fa5c0...	DFW	2009-01-05T00:...	LAX	AA436
64405b0b4fa5c0...	SFO	2010-03-18T00:...	LAS	UA451

Itemset Mining

In our dataset, we are using itemset mining to find the largest number of flights delayed on the same day.

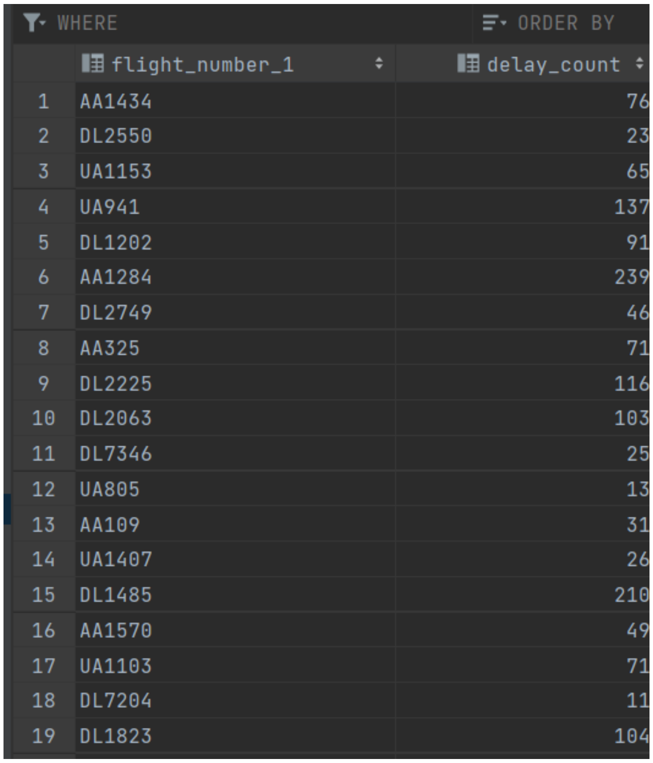
USING SQL

Generating Lattice 1

Below is the query for generating the L1(level 1 lattice) and the output is as shown below where the minimum support is 10.

```
SELECT flight_number as flight_number_1, COUNT(*) as delay_count

INTO flights.L1

FROM flights.major_airlines

GROUP BY  flight_number

HAVING COUNT(*) > 9;
```

Output:



	flight_number_1	delay_count
1	AA1434	76
2	DL2550	23
3	UA1153	65
4	UA941	137
5	DL1202	91
6	AA1284	239
7	DL2749	46
8	AA325	71
9	DL2225	116
10	DL2063	103
11	DL7346	25
12	UA805	13
13	AA109	31
14	UA1407	26
15	DL1485	210
16	AA1570	49
17	UA1103	71
18	DL7204	11
19	DL1823	104

Itemset Mining for remaining Lattices

We used pyscopg2 in python to perform the itemset generation for each level of lattice.

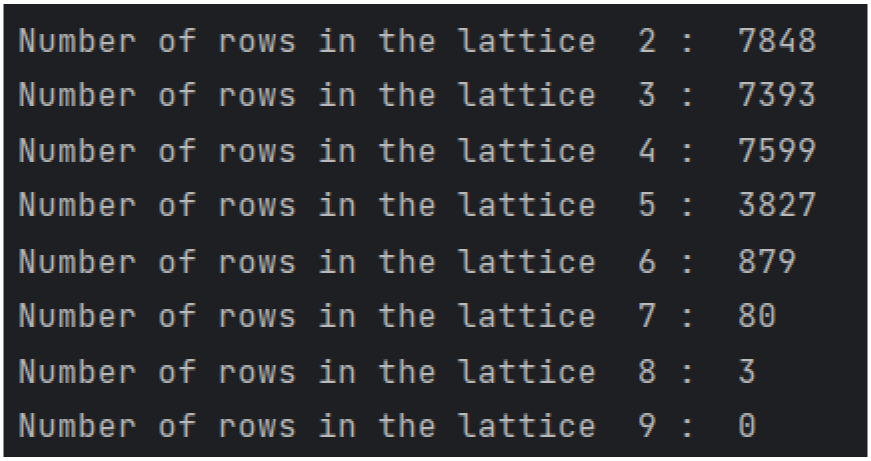
```
import psycopg2 as pg

db = pg.connect(database='Project',

            user='postgres',

            password='your_password',

            host='localhost',

            port=5432)

cursor = db.cursor()

query = ''
```

```
cursor.execute('select * from flights.l1')

rows = cursor.fetchall()

lattice_level = 2

while rows and lattice_level < 11:

    query = "select "

    for i in range(1, lattice_level):

        query += "p.flight_number_" + str(i) + " as flight_number_" + str(i) + ','

    query += "q.flight_number_" + str(lattice_level - 1) + " as flight_number_" + str(

        lattice_level) + ", count(*) into flights.l" + str(lattice_level) + " "

    query += "from flights.l" + str(lattice_level - 1) + " p, flights.l" + str(lattice_level - 1) +
" q, "

    for i in range(1, lattice_level + 1):

        query += "flights.major_airlines ma" + str(i) + ","

    query = query[:-1] + " where "

    for i in range(1, lattice_level - 1):

        query += "p.flight_number_" + str(i) + " = q.flight_number_" + str(i) + " and "

    query += "p.flight_number_" + str(lattice_level - 1) + "< q.flight_number_" +
str(lattice_level - 1) + " and "

    for i in range(1, lattice_level):

        query += 'p.flight_number_' + str(i) + "= ma" + str(i) + ".flight_number and "

    query += 'q.flight_number_' + str(lattice_level - 1) + ' = ma' + str(lattice_level) +
'.flight_number and '

    for i in range(1, lattice_level + 1):
```

```
    for j in range(i + 1, lattice_level + 1):
        query += "ma" + str(i) + ".flight_date = ma" + str(j) + ".flight_date and "
for i in range(1, lattice_level + 1):
    for j in range(i + 1, lattice_level + 1):
        query += "ma" + str(i) + ".source = ma" + str(j) + ".source and "
for i in range(1, lattice_level + 1):
    for j in range(i + 1, lattice_level + 1):
        query += "ma" + str(i) + ".destination = ma" + str(j) + ".destination and "

query = query[:-4] + 'group by ('
for i in range(1, lattice_level):
    query += 'p.flight_number_' + str(i) + ','
query += 'q.flight_number_' + str(lattice_level - 1) + ") having count(*)>9"
print('query', query)
cursor.execute(query)
db.commit()
cursor.execute('select * from flights.l' + str(lattice_level))
rows = cursor.fetchall()
print('Number of rows in the lattice ', lattice_level, ": ", len(rows))
lattice_level += 1
```

Output:

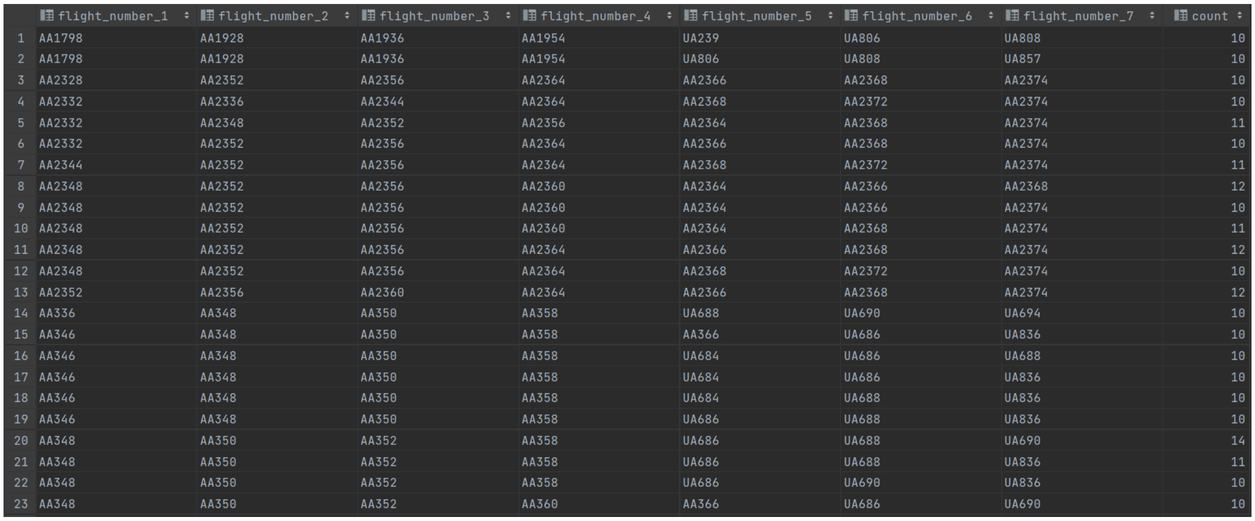
These are the number of records in each lattice



```
Number of rows in the lattice  2 :  7848
Number of rows in the lattice  3 :  7393
Number of rows in the lattice  4 :  7599
Number of rows in the lattice  5 :  3827
Number of rows in the lattice  6 :  879
Number of rows in the lattice  7 :  80
Number of rows in the lattice  8 :  3
Number of rows in the lattice  9 :  0
```

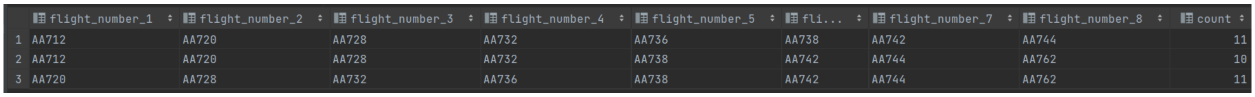
Here are the last two lattices generated

Lattice 7



	flight_number_1	flight_number_2	flight_number_3	flight_number_4	flight_number_5	flight_number_6	flight_number_7	count
1	AA1798	AA1928	AA1936	AA1954	UA239	UA806	UA808	10
2	AA1798	AA1928	AA1936	AA1954	UA806	UA808	UA857	10
3	AA2328	AA2352	AA2356	AA2364	AA2366	AA2368	AA2374	10
4	AA2332	AA2336	AA2344	AA2364	AA2368	AA2372	AA2374	10
5	AA2332	AA2348	AA2352	AA2356	AA2364	AA2368	AA2374	11
6	AA2332	AA2352	AA2356	AA2364	AA2366	AA2368	AA2374	10
7	AA2344	AA2352	AA2356	AA2364	AA2368	AA2372	AA2374	11
8	AA2348	AA2352	AA2356	AA2360	AA2364	AA2366	AA2368	12
9	AA2348	AA2352	AA2356	AA2360	AA2364	AA2366	AA2374	10
10	AA2348	AA2352	AA2356	AA2360	AA2364	AA2368	AA2374	11
11	AA2348	AA2352	AA2356	AA2364	AA2366	AA2368	AA2374	12
12	AA2348	AA2352	AA2356	AA2364	AA2368	AA2372	AA2374	10
13	AA2352	AA2356	AA2360	AA2364	AA2366	AA2368	AA2374	12
14	AA336	AA348	AA350	AA358	UA688	UA690	UA694	10
15	AA346	AA348	AA350	AA358	AA366	UA686	UA836	10
16	AA346	AA348	AA350	AA358	UA684	UA686	UA688	10
17	AA346	AA348	AA350	AA358	UA684	UA686	UA836	10
18	AA346	AA348	AA350	AA358	UA684	UA688	UA836	10
19	AA346	AA348	AA350	AA358	UA686	UA688	UA836	10
20	AA348	AA350	AA352	AA358	UA686	UA688	UA690	14
21	AA348	AA350	AA352	AA358	UA686	UA688	UA836	11
22	AA348	AA350	AA352	AA358	UA686	UA690	UA836	10
23	AA348	AA350	AA352	AA360	AA366	UA686	UA690	10

Lattice 8



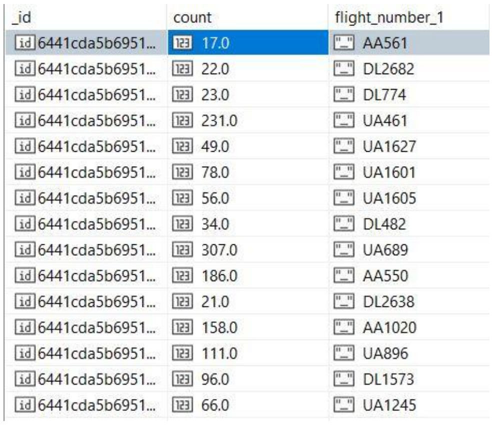
	flight_number_1	flight_number_2	flight_number_3	flight_number_4	flight_number_5	fli...	flight_number_7	flight_number_8	count
1	AA712	AA720	AA728	AA732	AA736	AA738	AA742	AA744	11
2	AA712	AA720	AA728	AA732	AA738	AA742	AA744	AA762	10
3	AA720	AA728	AA732	AA736	AA738	AA742	AA744	AA762	11

USING MONGO

Generating Lattice 1

```
db.major_airlines.aggregate([
   { "$group": { _id: "$flight_number", count: { $sum: 1 } } },
   { $match: { count: { $gt: 9 } } },
   {
      $project: {
         flight_number_1: "$_id",
         count: 1,
         _id: 0
      }
   },
   { $out: "l1" }
])
```

Output:

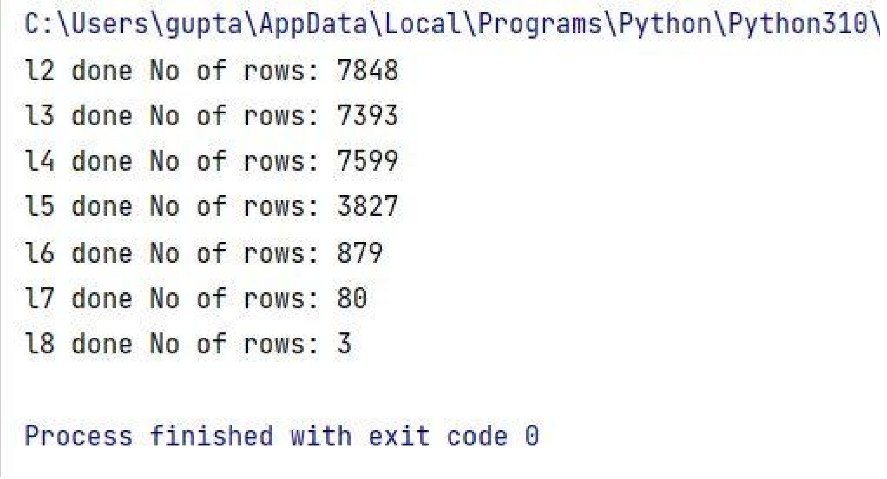


_id	count	flight_number_1
6441cda5b6951...	17.0	AA561
6441cda5b6951...	22.0	DL2682
6441cda5b6951...	23.0	DL774
6441cda5b6951...	231.0	UA461
6441cda5b6951...	49.0	UA1627
6441cda5b6951...	78.0	UA1601
6441cda5b6951...	56.0	UA1605
6441cda5b6951...	34.0	DL482
6441cda5b6951...	307.0	UA689
6441cda5b6951...	186.0	AA550
6441cda5b6951...	21.0	DL2638
6441cda5b6951...	158.0	AA1020
6441cda5b6951...	111.0	UA896
6441cda5b6951...	96.0	DL1573
6441cda5b6951...	66.0	UA1245

Itemset mining for remaining lattices

Since writing mongo queries for each lattice was difficult, we used python to create each lattice and perform itemset mining.

```
import pandas as pd

from pymongo import MongoClient


client = MongoClient('localhost', 27017)

db = client['project']

ma = pd.DataFrame(list(db["major_airlines"].find({})))

ma.drop(columns=["_id"], inplace=True)

ma.rename(columns={'flight_number': 'flight_number_1'}, inplace=True)

lattice_count = 2

while True:

    temp = []

    l1 = pd.DataFrame(list(db["l1"].find({})))

    l1.drop(columns=['count', '_id'], inplace=True)

    for i in range(0, lattice_count):

        temp.append(pd.merge(l1, ma, how='inner', on='flight_number_1'))

    for i in range(1, lattice_count):

        new = pd.merge(temp[i - 1], temp[i], how='inner',

                       on=['flight_date', 'origin', 'destination'])

        new.rename(columns={'flight_number_1_x': 'flight_number_1',

                            'flight_number_1_y': f'flight_number_{i + 1}'},
                inplace=True)
        new = new[new[f'flight_number_{i}'] < new[f'flight_number_{i + 1}']]
        temp[i] = new
    new_lattice = temp[-1]
    columns = [f'flight_number_{i}' for i in range(1, lattice_count + 1)]
    final_lattice = new_lattice.groupby(columns)['flight_date'].aggregate(
        'count').reset_index()
    final_lattice.rename(columns={'flight_date': 'count'}, inplace=True)
    final_lattice = final_lattice[final_lattice['count'] > 9]
    if len(final_lattice) == 0:
        break
    db.create_collection(f'l{lattice_count}')
    db[f'l{lattice_count}'].insert_many(final_lattice.to_dict('records'))
    print(f'l{lattice_count}', 'done', 'No of rows:', len(final_lattice))
    lattice_count += 1
```



```
C:\Users\gupta\AppData\Local\Programs\Python\Python310\
l2 done No of rows: 7848
l3 done No of rows: 7393
l4 done No of rows: 7599
l5 done No of rows: 3827
l6 done No of rows: 879
l7 done No of rows: 80
l8 done No of rows: 3

Process finished with exit code 0
```

What is better for itemset mining - Mongo or SQL

In our analysis, we found that SQL is better at performing itemset mining. This is because joining tables in a relational database is much easier and faster as compared to a document database. Secondly, the structure for the data in relational databases is more feasible to perform itemset mining as all data is organized in separate tables. However, in a document database, there can be sub-documents embedded in each entry, making it difficult to fetch the data and join the records based on those values.
This was experienced by us when writing the code for mongo and sql to perform itemset mining.