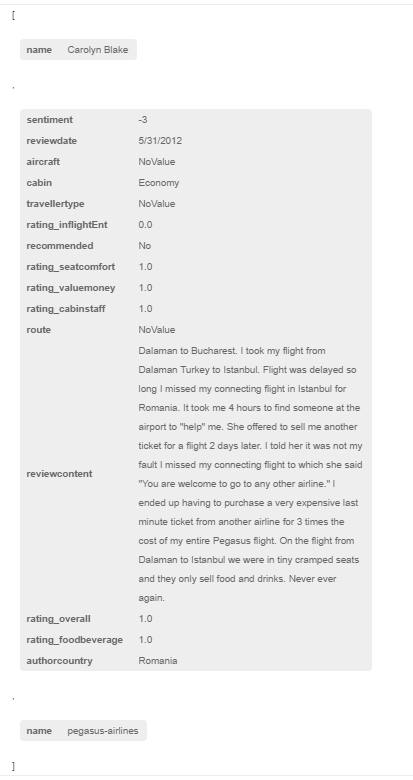
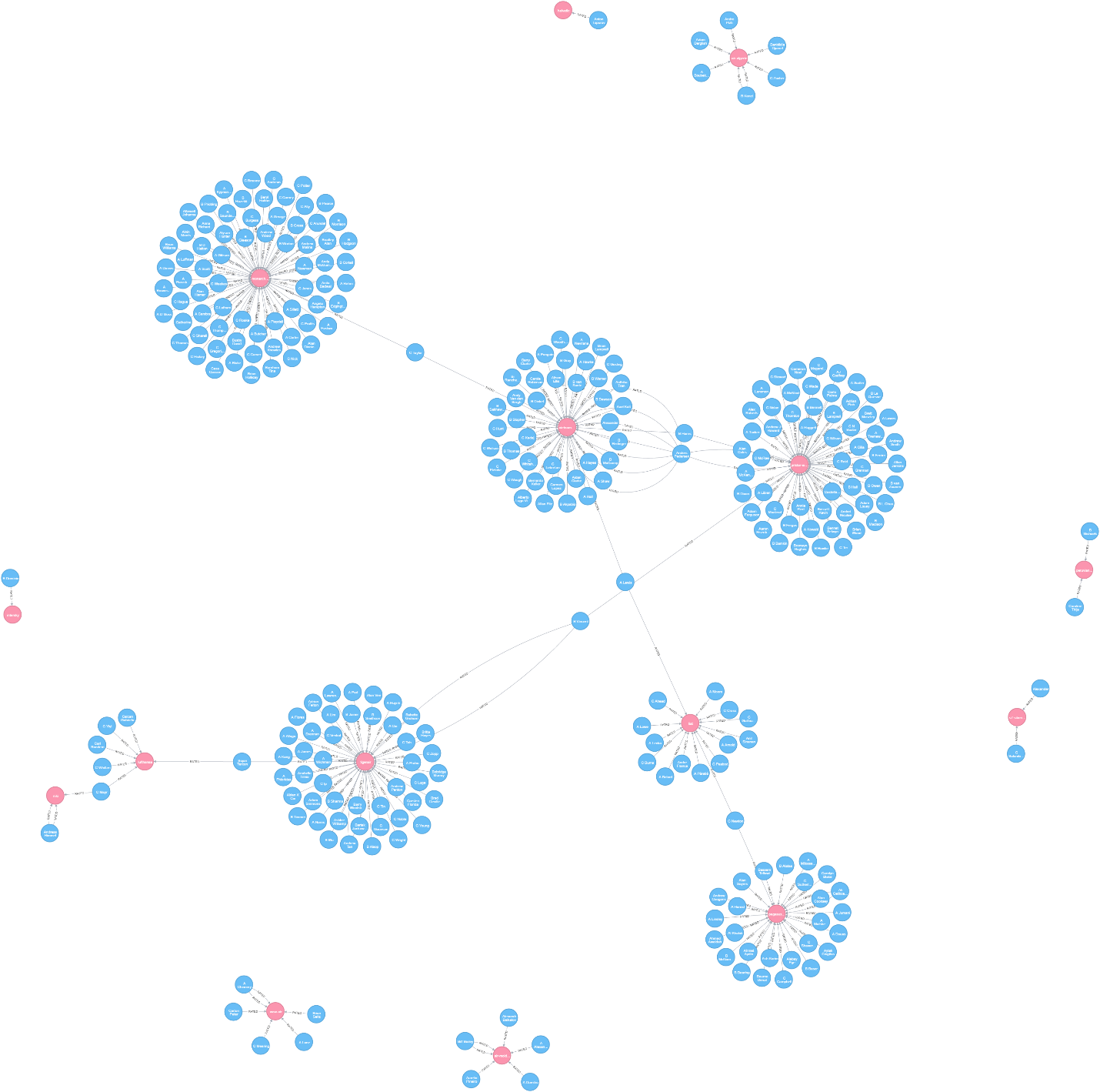
**TEAM FALCONS : TERM PROJECT : NEO4J**

**Part 1: Inserting the data in Neo4j after getting the sentiment for each review from MongoDB**

* Py2neo version 3.1.2 has been used
* Created a python code using py2neo to import data from a csv file and created two nodes: Airline and Author and created a relationship called RATED between these nodes
* As a part of data clean, all null values were substituted with String ‘NoValue’ and used Latin encoding to make sure the parser parses characters such as è, ä, ø, etc.
* To make the data uniform, all number values were converted to String and the Recommend value of 0 and 1 were substituted with No and Yes
* Since the we are sampling 10000 records from the database of 40000 records, the program takes approximately 10 mins to create the nodes and relationships
* A sample relationship is shown below:



* The sample database structure in Neo4j has been shown below:



**Python Code:**

import pandas as pd

import csv

import math

#py2neo 3.1.2

import py2neo

from py2neo import \*

py2neo.authenticate(*"localhost:7474"*, *"neo4j"*, *"termproject"*)

graph = Graph(*"http://localhost:7474/db/data/"*)

#graph.delete\_all()

#1 Creating database in Neo4j from .csv file

# Reading records from the csv file

review = pd.read\_csv(*'AirlineSentimentResults.csv'*, encoding=*'latin-1'*)

#Using only the first 10000 reviews for analysis

review = review[:10000]

# Creating a set of unique author and airline names

setAuthor = set(review[*'authorname'*])

setAirline = set(review[*'airlinename'*])

# Insert Author Nodes

for i in setAuthor:

i = Node(*"Author"*, name=i)

graph.create(i)

# Insert Airline Nodes

for j in setAirline:

j = Node(*"Airline"*, name=j)

graph.create(j)

# Create relationships between Author and Airline and give values to its parameters

statement = *"MATCH (a {name:{A}}), (b {name:{B}}) \*

*CREATE (a)-[:RATED {authorcountry:{C}, \*

*aircraft:{D}, \*

*route:{E}, \*

*travellertype:{F}, \*

*cabin:{G}, \*

*rating\_cabinstaff:{H}, \*

*rating\_foodbeverage:{I}, \*

*rating\_inflightEnt:{J}, \*

*rating\_overall:{K}, \*

*rating\_seatcomfort:{L}, \*

*rating\_valuemoney:{M}, \*

*recommended:{N}, \*

*reviewcontent:{O}, \*

*reviewdate:{P}, \*

*sentiment:{Q}}]->(b)"*

authorcountry = []

aircraft = []

route = []

travellertype = []

cabin = []

rating\_cabinstaff = []

rating\_foodbeverage = []

rating\_inflightEnt = []

rating\_overall = []

rating\_seatcomfort = []

rating\_valuemoney = []

recommended = []

sentiment = []

for k in range(0, len(review)):

if review.loc[k, *'authorcountry'*] != review.loc[k, *'authorcountry'*]:

authorcountry.append(*'NoValue'*)

else:

authorcountry.append(review.loc[k, *'authorcountry'*])

if review.loc[k, *'aircraft'*] != review.loc[k, *'aircraft'*]:

aircraft.append(*'NoValue'*)

else:

aircraft.append(review.loc[k, *'aircraft'*])

if review.loc[k, *'route'*] != review.loc[k, *'route'*]:

route.append(*'NoValue'*)

else:

route.append(review.loc[k, *'route'*])

if review.loc[k, *'travellertype'*] != review.loc[k, *'travellertype'*]:

travellertype.append(*'NoValue'*)

else:

travellertype.append(review.loc[k, *'travellertype'*])

if review.loc[k, *'cabin'*] != review.loc[k, *'cabin'*]:

cabin.append(*'NoValue'*)

else:

cabin.append(review.loc[k, *'cabin'*])

if (math.isnan(review.loc[k, *'rating\_cabinstaff'*])):

rating\_cabinstaff.append(*'NoValue'*)

else:

rating\_cabinstaff.append(str(review.loc[k, *'rating\_cabinstaff'*]))

if (math.isnan(review.loc[k, *'rating\_foodbeverage'*])):

rating\_foodbeverage.append(*'NoValue'*)

else:

rating\_foodbeverage.append(str(review.loc[k, *'rating\_foodbeverage'*]))

if (math.isnan(review.loc[k, *'rating\_inflightEnt'*])):

rating\_inflightEnt.append(*'NoValue'*)

else:

rating\_inflightEnt.append(str(review.loc[k, *'rating\_inflightEnt'*]))

if (math.isnan(review.loc[k, *'rating\_overall'*])):

rating\_overall.append(*'NoValue'*)

else:

rating\_overall.append(str(review.loc[k, *'rating\_overall'*]))

if (math.isnan(review.loc[k, *'rating\_seatcomfort'*])):

rating\_seatcomfort.append(*'NoValue'*)

else:

rating\_seatcomfort.append(str(review.loc[k, *'rating\_seatcomfort'*]))

if (math.isnan(review.loc[k, *'rating\_valuemoney'*])):

rating\_valuemoney.append(*'NoValue'*)

else:

rating\_valuemoney.append(str(review.loc[k, *'rating\_valuemoney'*]))

if (math.isnan(review.loc[k, *'recommended'*])):

recommended.append(*'NoValue'*)

elif review.loc[k, *'recommended'*] == 1:

recommended.append(*'Yes'*)

elif review.loc[k, *'recommended'*] == 0:

recommended.append(*'No'*)

sentiment.append(str(review.loc[k, *'sentiment'*]))

tx = graph.begin()

for k in range(0, len(review)):

tx.run(statement, {*"A"*: review.loc[k,*'authorname'*],

*"B"*: review.loc[k,*'airlinename'*],

*"C"*: authorcountry[k],

*"D"*: aircraft[k],

*"E"*: route[k],

*"F"*: travellertype[k],

*"G"*: cabin[k],

*"H"*: rating\_cabinstaff[k],

*"I"*: rating\_foodbeverage[k],

*"J"*: rating\_inflightEnt[k],

*"K"*: rating\_overall[k],

*"L"*: rating\_seatcomfort[k],

*"M"*: rating\_valuemoney[k],

*"N"*: recommended[k],

*"O"*: review.loc[k,*'reviewcontent'*],

*"P"*: review.loc[k,*'reviewdate'*],

*"Q"*: sentiment[k]})

tx.commit()

**Part 2: Comparison of direct airline competitors as rated and perceived by common raters**

**Python Code:**

import pandas as pd

import csv

import math

#py2neo 3.1.2

import py2neo

from py2neo import \*

py2neo.authenticate(*"localhost:7474"*, *"neo4j"*, *"termproject"*)

graph = Graph(*"http://localhost:7474/db/data/"*)

#2 Comparison between direct competitors

results = graph.run(*"""MATCH (ai1: Airline)<-[r1:RATED]-(au: Author)-[r2:RATED]->(ai2: Airline)*

*WHERE NOT (ai1 = ai2)*

*WITH ai1, ai2, au, r1, r2,*

*CASE toInt(r1.sentiment) > toInt(r2.sentiment)*

*WHEN True THEN ai1.name*

*ELSE ai2.name END as BetterAirline,*

*CASE toInt(r1.rating\_cabinstaff) > toInt(r2.rating\_cabinstaff)*

*WHEN True THEN ai1.name*

*ELSE ai2.name END as BetterCabinStaff,*

*CASE toInt(r1.rating\_foodbeverage) > toInt(r2.rating\_foodbeverage)*

*WHEN True THEN ai1.name*

*ELSE ai2.name END as BetterFoodBeverage,*

*CASE toInt(r1.rating\_inflightEnt) > toInt(r2.rating\_inflightEnt)*

*WHEN True THEN ai1.name*

*ELSE ai2.name END as BetterInFlightEntertainment,*

*CASE toInt(r1.rating\_seatcomfort) > toInt(r2.rating\_seatcomfort)*

*WHEN True THEN ai1.name*

*ELSE ai2.name END as BetterSeatComfort,*

*CASE toInt(r1.rating\_valuemoney) > toInt(r2.rating\_valuemoney)*

*WHEN True THEN ai1.name*

*ELSE ai2.name END as BetterValueForMoney,*

*CASE toInt(r1.recommended) > toInt(r2.recommended)*

*WHEN True THEN ai1.name*

*ELSE ai2.name END as RecommendedAirline*

*Return au.name, ai1.name, r1.sentiment,*

*ai2.name, r2.sentiment, BetterAirline,*

*RecommendedAirline, BetterCabinStaff, BetterFoodBeverage,*

*BetterInFlightEntertainment, BetterSeatComfort, BetterValueForMoney"""*)

for result in results:

print(result)

print()

**Sample Output (Detailed output is mentioned in OutputResults.docx):**

('au.name': 'Anders Pedersen',

'ai1.name': 'vietjetair',

'r1.sentiment': '-5',

'ai2.name': 'turkish-airlines',

'r2.sentiment': '5',

'BetterAirline': 'turkish-airlines',

'RecommendedAirline': 'turkish-airlines',

'BetterCabinStaff': 'vietjetair',

'BetterFoodBeverage': 'turkish-airlines',

'BetterInFlightEntertainment': 'turkish-airlines',

'BetterSeatComfort': 'turkish-airlines',

'BetterValueForMoney': 'turkish-airlines')

**Justification of solution:**

The comparison is done between direct airline competitors i.e. airlines having common raters. We have compared the ratings of these competitor airlines and shown which out of the two airlines wins in different rating categories.

For example in the above sample output, author/customer ‘Anders Pedersen’ has given the rating for Airlines ‘vietjetair’ and ‘turkish-airlines’. While comparing the ratings, the following results come up:

* After doing sentiment analysis on the review content, Turkish airlines has better or more positive sentiment that vietjetair
* Turkish airlines is recommended over vietjetair and has better In-Flight Food and Beverages, In-Flight Entertainment, Seat Comfort and Value for Money than Vietjetair whereas Vietjetair has better Cabin Staff than Turkish airlines

**Part 4.1: Finding the count of common airlines used by two raters**

**Python Code:**

import pandas as pd

import csv

import math

#py2neo 3.1.2

import py2neo

from py2neo import \*

py2neo.authenticate(*"localhost:7474"*, *"neo4j"*, *"termproject"*)

graph = Graph(*"http://localhost:7474/db/data/"*)

results = graph.run(*"""MATCH (au1:Author)-[r1:RATED]->(ai:Airline)<-[r2:RATED]-(au2:Author)*

*WHERE NOT (au1 = au2)*

*WITH count(ai) as SimilarityIndex, au1, au2*

*ORDER BY SimilarityIndex DESC LIMIT 20*

*CREATE UNIQUE (au1)-[d:SimilarityIndex]->(au2)*

*SET d.count=SimilarityIndex*

*RETURN d.count as SimilarityIndex, au1.name, au2.name"""*)

for result in results:

print(result)

**Sample Output (Detailed output is mentioned in OutputResults.docx):**

('(au2.name)': 'Bob Motto', '(au1.name)': 'C Cutts', 'SimilarityIndex': 90)

**Justification of solution:**

The query is written to compare two authors/customers in such a way so as to find the number of common airlines rated/used by these authors. For example: Bob Motto and C Cutts both rated 90 common airlines. After getting this index, we are creating a relationship called SimilarityIndex between these authors to store the number of common airlines rated.

**Part 4.2: Finding the top common flyers for each airline**

**Python Code:**

import pandas as pd

import csv

import math

#py2neo 3.1.2

import py2neo

from py2neo import \*

py2neo.authenticate(*"localhost:7474"*, *"neo4j"*, *"termproject"*)

graph = Graph(*"http://localhost:7474/db/data/"*)

#1 Creating database in Neo4j from .csv file

# Reading records from the csv file

review = pd.read\_csv(*AirlineSentimentResults.csv'*, encoding=*'latin-1'*)

#Using only the first 10000 reviews for analysis

review = review[:10000]

# Creating a set of unique airline names

setAirline = set(review[*'airlinename'*])

statement = *"MATCH (au: Author)-[r:RATED]->(ai: Airline {name:{A}})<-[r2:RATED]-(au2: Author) \*

*WITH count(r) as NumberOfTravels, count(r2) as NumberOfTravels2, ai, au, au2 \*

*WHERE NumberOfTravels > 5 AND NOT (au.name = au2.name)\*

*RETURN DISTINCT(ai.name) as Airline, au.name as Author, \*

*au2.name as Author2, NumberOfTravels, NumberOfTravels2 \*

*ORDER BY NumberOfTravels DESC LIMIT 1"*

tx = graph.begin()

for k in setAirline:

result = tx.run(statement, {*"A"*: k})

print(result.data())

print()

**Sample Output (Detailed output is mentioned in OutputResults.docx):**

[{'Airline': 'british-airways', 'Author': 'Bob Motto', 'Author2': 'C Cutts', 'NumberOfTravels': 90, 'NumberOfTravels2': 90}]

**Justification of solution:**

The query is written to compare two authors/customers in such a way so as to find the top travelers for each airline. This shows the likeness between the authors for each airline.