Network Analysis Project

Research Topic

The research topic I chose to analyze for this project was the flow of airport traffic in the United States. Airport traffic is a highly interconnected and complex system in the United States. This makes it a good candidate for analysis via a network graph to search for novel patterns. The data source chosen was from the Bureau of Transportation Statistics, which keeps statistics on all forms of transportation in the United States. The plan is for two network graphs to be constructed in Python, one for passenger traffic between airports, and one for freight transported between airports. For the first network graph, node size will be determined by total amount of incoming and outgoing passengers for an airport, while edge attributes will be the number of passengers transferred between both airports. For the second network graph, node size will be determined by total amount of freight received and transported for an airport, while edge attributes will be determined by amount of freight transported between airports. The particular dataset utilized was exact flight data in a csv file from the Bureau of Transportation Statistics from January 2022 to July 2022, which showed exact number of passengers and freight per flight. The csv file was then loaded into excel, where preliminary analysis was done for aggregation of flight data by airport. Since there are thousands of airports in the United States, only the top 150 airports in both network graphs were chosen to make sure visuals were legible.

Python Analysis

First step below was to load the appropriate libraries in python and read in the data into 3 data frames:

In [2]:
```
import pandas as pd
from matplotlib.pyplot import figure
import matplotlib.pyplot as plt
import networkx as nx
```

In [3]:
```
node_df=pd.read_excel("us_top_airports_passengers.xlsx") #data frame for top a
irports by passenger traffic.
node_df_2=pd.read_excel("us_top_airports_freights.xlsx") #data frame for top a
irports by freight traffic.
edge_df=pd.read_excel("us_airplane_edge_attributes.xlsx") #data frame for pass
enger and freight flow between airports.
```

Below is the data structure of each data frame, the first data frame shows number of total transported passengers by airport ticker. The second data frame shows amount of total freight transported by airport, and the third data frame shows total amount of freight and passengers transported between origin and destination airport tickers.

In [40]:
```
node_df #first data frame showing total number of transported passenger by air
port ticker from January 2022 to July 2022.
```

Out[40]:

	ticker	sum_pass
0	ATL	46045579
1	DEN	35788898
2	DFW	34864798
3	ORD	31017768
4	LAX	27869239
...	...	...
1249	GMT	0
1250	FWL	0
1251	SYA	0
1252	HLI	0
1253	BYH	0

1254 rows × 2 columns

In [4]:
```
node_df_2 #second data frame showing total amount of transported freight in po
unds
#by airport ticker from January 2022 to July 2022.
```

Out[4]:

	ticker	sum_freight
0	MEM	4486933287
1	SDF	3272560317
2	ANC	2157600512
3	CVG	1390067806
4	IND	1236530393
...	...	...
1249	NKX	0
1250	ULS	0
1251	CWF	0
1252	TIK	0
1253	BYH	0

1254 rows × 2 columns

In [12]:
```
edge_df #third data frame showing total amount of transported freight and passengers between origin and destination airports.
```

Out[12]:

	org_tik	des_tik	sum_pass	sum_freight
0	ANC	ANC	3196	5814689
1	GEG	SLC	91547	5597
2	PDX	ATL	119949	623495
3	MKE	ONT	85	3
4	IND	LAX	52221	39709463
...	...	...	...	...
10346	DFW	TUS	191575	41531
10347	TUS	DFW	198245	18013
10348	DCA	DFW	243722	76090
10349	DFW	CLT	330090	515752
10350	CLT	DFW	328414	267306

10351 rows × 4 columns

Now the data was loaded in Python, the next step is to create unique lists which can be used to create the network graphs. Lists were capped at top 150 entries to represent the top 150 airports:

In [56]:
```
node_list=node_df['ticker'].values.tolist() #create a node list for the top 150 tickers for the first network graph.
node_list=node_list[0:150]  #cap first node list to top 150 entries.
node_list_2=node_df_2['ticker'].values.tolist() #create a node list for the top 150 tickers for the second network graph.
node_list_2=node_list_2[0:150] #cap second node list to top 150 entries.
pass_list=node_df['sum_pass'].values.tolist() #create a passanger list for the top 150 tickers for the first network graph.
pass_list=pass_list[0:150]
freight_list=node_df_2['sum_freight'].values.tolist() #create a freight list for the top 150 tickers for the first network graph.
freight_list=freight_list[0:150]
org_list=edge_df['org_tik'].values.tolist() #create a unique list for 'origin' airports.
des_list=edge_df['des_tik'].values.tolist() #create a unique list for 'destination' airports.
edge_pass_list=edge_df['sum_pass'].values.tolist() #create edge list for first network graph
edge_freight_list=edge_df['sum_freight'].values.tolist() #create edge list for second network graph
```

The next challenge was to find the total amount of passengers and freight transported between two airports. The way the Bureau of Transportation Statistics had formatted the data was for airport traffic between original and destination airports. This means that the combination of two of the same airports can appear multiple times since origin and destination airports can be swapped. For the purposes of this network analysis, it does not matter which airport was the origin or destination, we are just looking for total freight and passenger traffic between two airports for our edge attributes. A python dictionary was used for this:

In [57]:
```
dict_edge={} #dictionary for first network graph
dict_edge_2={} #dictionary for second network graph
for i in range(0,len(org_list)): #analysis for first dictionary to get total p
assenger flow between two airports
    temp_list=[org_list[i],des_list[i]]
    temp_list.sort()
    temp_str=str(temp_list)
    temp_val=temp_str[1:len(temp_str)-1]
    if temp_val in dict_edge and org_list[i] in node_list and des_list[i] in n
ode_list :
        dict_edge[temp_val]+=edge_pass_list[i]
    elif org_list[i] in node_list and des_list[i] in node_list:
        dict_edge[temp_val]=edge_pass_list[i]

for i in range(0,len(org_list)): #analysis for second dictionary to get total
 freight flow between two airports
    temp_list=[org_list[i],des_list[i]]
    temp_list.sort()
    temp_str=str(temp_list)
    temp_val=temp_str[1:len(temp_str)-1]
    if temp_val in dict_edge_2 and org_list[i] in node_list_2 and des_list[i]
in node_list_2 :
        dict_edge_2[temp_val]+=edge_freight_list[i]
    elif org_list[i] in node_list_2 and des_list[i] in node_list_2:
        dict_edge_2[temp_val]=edge_freight_list[i]
```

Now that the data was formatted correctly for our nodes and edges, the network graphs can be created. There will be two network graphs, one titled "pass_network" and another titled "freight_network." Code to create the network graphs are below:

In [22]:

```
pass_network = nx.Graph() # initialize network graph for total passangers tran
sported at the top 150 U.S. airports
for i in range(0,len(node_list)): #top 150 airports from node list to be added
as nodes to our network graph
    temp_node=node_list[i] #get temporary node
    temp_pass=pass_list[i]/1000000 #node size, divided by 1 million to scale t
he data for the network graph.
    pass_network.add_node(temp_node) #add unique airport as node to network gr
aph
    pass_network.nodes[temp_node]['total_pass'] = temp_pass #add total amount
 of transported passangers as node size.

for i in dict_edge: #edge attribute list to be added to the network graph
    temp_list_2=i.split(",") #split the dictionary into a list with both airpo
rts
    temp_val_1=temp_list_2[0].replace(' ','') #first airport placeholder
    temp_val_2=temp_list_2[1].replace(' ','') #second airport placeholder
    temp_val_3=temp_val_1[1:len(temp_val_1)-1] #string manipulation to extract
first airport
    temp_val_4=temp_val_2[1:len(temp_val_2)-1] #string manipulation to extract
second airport
    temp_val_5=dict_edge[i]/1000000 #edge attribute size, divided by 1 million
to scale the data for the network graph.
    if temp_val_4 != temp_val_3: #ensure airports don't equal each other befor
e adding as edge attribute
        pass_network.add_edge(temp_val_3,temp_val_4, shared_pass = temp_val_5)
#add the edge to the network
```

In [68]:

```
freight_network = nx.Graph() # initialize network graph for total freight tran
sported at the top 150 U.S. airports
for i in range(0,len(node_list_2)): #top 150 airports from node list to be add
ed as nodes to our network graph
    temp_node=node_list_2[i] #get temporary node
    temp_freight=freight_list[i]/100000000 #node size, divided by 100 million
 to scale the data for the network graph.
    freight_network.add_node(temp_node) #add unique airport as node to network
graph
    freight_network.nodes[temp_node]['total_freight'] = temp_freight #add tota
l amount of transported freught as node size.

for i in dict_edge_2: #edge attribute list to be added to the network graph
    temp_list_2=i.split(",") #split the dictionary into a list with both airpo
rts
    temp_val_1=temp_list_2[0].replace(' ','') #first airport placeholder
    temp_val_2=temp_list_2[1].replace(' ','') #second airport placeholder
    temp_val_3=temp_val_1[1:len(temp_val_1)-1] #string manipulation to extract
first airport
    temp_val_4=temp_val_2[1:len(temp_val_2)-1] #string manipulation to extract
second airport
    temp_val_5=dict_edge[i]/1000000 #edge attribute size, divided by 1 million
to scale the data for the network graph.
    if temp_val_4 != temp_val_3: #ensure airports don't equal each other befor
e adding as edge attribute
        freight_network.add_edge(temp_val_3,temp_val_4, shared_freight = temp_
val_5) #add the edge to the network
```

Below are the largest 150 nodes in order for the pass_network and freight_network:

In [53]:
```
pass_network.nodes
```

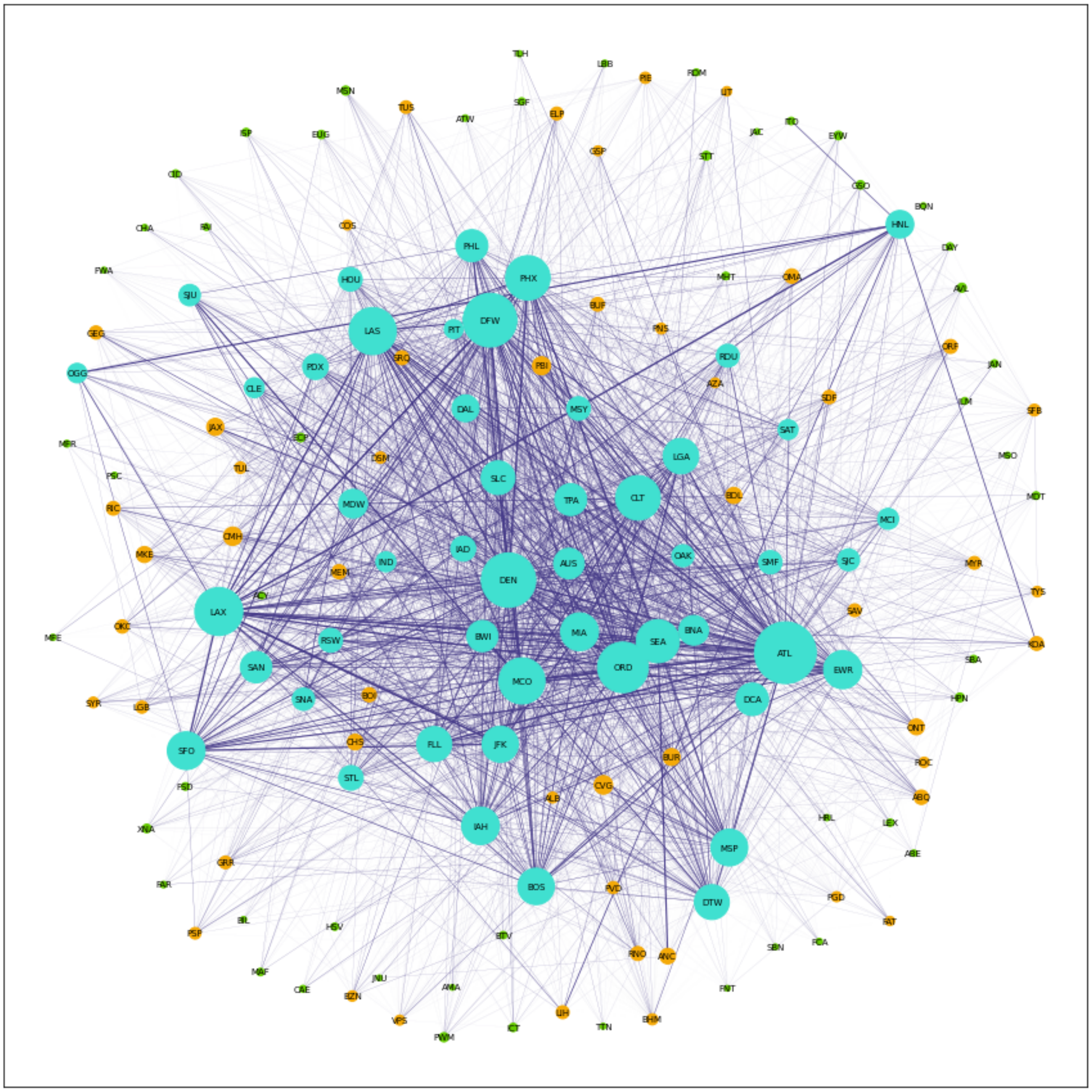
Out[53]:
```
NodeView(('ATL', 'DEN', 'DFW', 'ORD', 'LAX', 'LAS', 'MCO', 'CLT', 'PHX', 'SE
A', 'EWR', 'MIA', 'SFO', 'IAH', 'MSP', 'BOS', 'JFK', 'LGA', 'FLL', 'DTW', 'SL
C', 'DCA', 'PHL', 'TPA', 'SAN', 'BWI', 'AUS', 'BNA', 'MDW', 'HNL', 'DAL', 'PD
X', 'IAD', 'STL', 'RSW', 'HOU', 'MSY', 'SMF', 'RDU', 'SNA', 'OAK', 'SJC', 'SJ
U', 'MCI', 'IND', 'SAT', 'CLE', 'OGG', 'PIT', 'CVG', 'CMH', 'PBI', 'JAX', 'BU
R', 'BDL', 'MKE', 'ONT', 'CHS', 'ANC', 'OMA', 'ABQ', 'MEM', 'BOI', 'RNO', 'SR
Q', 'ORF', 'KOA', 'BUF', 'RIC', 'SDF', 'OKC', 'GEG', 'ELP', 'LIH', 'SAV', 'MY
R', 'GRR', 'TUS', 'LGB', 'PVD', 'SFB', 'PSP', 'TUL', 'DSM', 'PIE', 'BHM', 'AL
B', 'PNS', 'SYR', 'TYS', 'BZN', 'ROC', 'PGD', 'VPS', 'COS', 'GSP', 'AZA', 'LI
T', 'FAT', 'PWM', 'MSN', 'AVL', 'HPN', 'STT', 'XNA', 'EYW', 'ECP', 'GSO', 'EU
G', 'ICT', 'ISP', 'MHT', 'FSD', 'MDT', 'CID', 'MAF', 'ITO', 'LEX', 'JAN', 'BT
V', 'DAY', 'HSV', 'SBA', 'SGF', 'FAI', 'CAE', 'ILM', 'RDM', 'MFR', 'ACY', 'LB
B', 'FAR', 'ABE', 'ATW', 'CHA', 'HRL', 'FCA', 'MFE', 'MSO', 'TLH', 'PSC', 'TT
N', 'FWA', 'BIL', 'JAC', 'JNU', 'SBN', 'FNT', 'AMA', 'BQN'))
```

In [59]:
```
freight_network.nodes
```

Out[59]:
```
NodeView(('MEM', 'SDF', 'ANC', 'CVG', 'IND', 'LAX', 'ONT', 'OAK', 'MIA', 'OR
D', 'HNL', 'DFW', 'EWR', 'JFK', 'PHL', 'RFD', 'ATL', 'AFW', 'SEA', 'PHX', 'PD
X', 'IAH', 'DEN', 'BWI', 'ILN', 'SBD', 'TPA', 'BOS', 'SJU', 'SFO', 'MSP', 'SL
C', 'BDL', 'MCO', 'GSO', 'CLT', 'LAL', 'AUS', 'SAT', 'DTW', 'MCI', 'SMF', 'SA
N', 'RDU', 'ELP', 'ABE', 'PIT', 'IAD', 'FLL', 'LAS', 'STL', 'BNA', 'MHT', 'BF
I', 'JAX', 'CLE', 'OMA', 'GEG', 'ABQ', 'MKE', 'RIC', 'MSY', 'RNO', 'MHR', 'CA
E', 'LCK', 'TUL', 'KOA', 'MDT', 'ROC', 'BOI', 'LBB', 'SYR', 'BIL', 'OGG', 'FA
R', 'SCK', 'FSD', 'CID', 'BUR', 'RIV', 'GRR', 'TYS', 'BUF', 'DSM', 'BQN', 'HS
V', 'OKC', 'HRL', 'SJC', 'GSP', 'PBI', 'TUS', 'LIH', 'ICT', 'ORF', 'LRD', 'SH
V', 'FWA', 'LAN', 'ABY', 'MSN', 'BHM', 'ALB', 'BFM', 'SWF', 'GTF', 'MDW', 'RS
W', 'LIT', 'SGF', 'LFT', 'ITO', 'TOL', 'SNA', 'DAL', 'LGB', 'FAT', 'PVD', 'GU
M', 'PIA', 'FAI', 'FNT', 'JAN', 'ROA', 'PNS', 'HOU', 'MFE', 'CPR', 'AKN', 'CH
S', 'BET', 'HTS', 'CHA', 'RST', 'SBN', 'COS', 'TLH', 'ATW', 'JNU', 'GYY', 'PW
M', 'BMI', 'SKF', 'SAV', 'OME', 'DAY', 'BRW', 'LGA', 'OTZ'))
```

The first network graph for passenger flow at U.S airports can now be created with the code below:

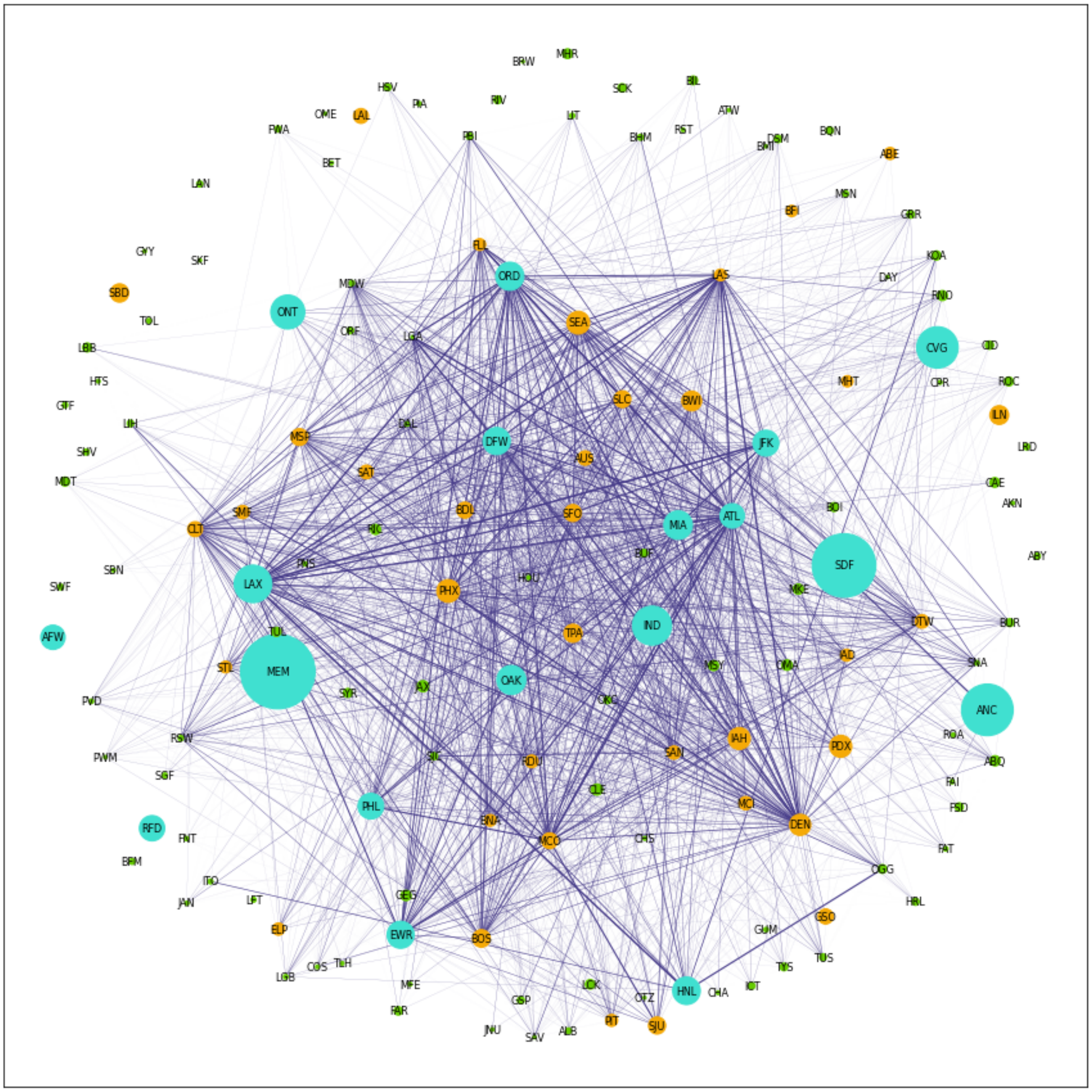
In [48]:
```
figure(figsize=(12,12)) #figure size for first network graph of passenger traf
fic flow
air_pos = nx.spring_layout(pass_network, k=2.8) #specify distance parameter be
tween nodes for first network graph
node_size=[50*pass_network.nodes[v]["total_pass"] for v in pass_network] #crea
te node size based on total passenger flow for airport
shared_pass = [pass_network.edges[e]['shared_pass'] for e in pass_network.edge
s] #create edge size based on passenger flow between airports
node_color=[]
for v in pass_network: #use different node colors for different airport sizes
    if pass_network.nodes[v]["total_pass"] > 4.3:
        node_color.append('#40E0D0')
    elif pass_network.nodes[v]["total_pass"] > 1.04:
        node_color.append('#f5aa07')
    else:
        node_color.append('#66cc00')
nx.draw_networkx_nodes(pass_network, air_pos, node_color=node_color, node_size
=node_size)
nx.draw_networkx_edges(pass_network, air_pos, width = shared_pass*10, edge_col
or ='#483D8B')
nx.draw_networkx_labels(pass_network, air_pos, font_size = 8)
plt.tight_layout()
plt.show() #plot network graph
```



In [ ]:
```
The second network graph for freight flow at U.S airports can now be created w
ith the code below:
```

In [70]:

```
figure(figsize=(12,12)) #figure size for first freight traffic flow
air_pos_2 = nx.spring_layout(freight_network, k=2.8)
node_size_2=[75*freight_network.nodes[v]['total_freight'] for v in freight_net
work]
shared_freight = [freight_network.edges[e]['shared_freight'] for e in freight_
network.edges]
node_color_2=[]
for v in freight_network:
    if freight_network.nodes[v]["total_freight"] > 4.3:
        node_color_2.append('#40E0D0')
    elif freight_network.nodes[v]["total_freight"] > 1.04:
        node_color_2.append('#f5aa07')
    else:
        node_color_2.append('#66cc00')
nx.draw_networkx_nodes(freight_network, air_pos_2, node_color=node_color_2, no
de_size=node_size_2)
nx.draw_networkx_edges(freight_network, air_pos_2, width = shared_freight*10,
edge_color ='#483D8B')
nx.draw_networkx_labels(freight_network, air_pos_2, font_size = 8)
plt.tight_layout()
plt.show()
```



It also is important to measure which nodes are the closet to all other nodes on average, the following code was used for this:

```
In [72]: closeness = nx.closeness_centrality(pass_network)
         sorted(closeness.items(), key=lambda x:x[1], reverse=True)[0:5] #top 5 closest
         nodes for first network graph

Out[72]: [('ORD', 0.907448835860469),
          ('LAS', 0.9018816650883188),
          ('ATL', 0.8909497661175514),
          ('DEN', 0.8855825988517829),
          ('DFW', 0.8855825988517829)]
```

In [73]:
```
closeness_2 = nx.closeness_centrality(freight_network)
sorted(closeness_2.items(), key=lambda x:x[1], reverse=True)[0:5] #top 5 close
st nodes for second network graph
```

Out[73]:
```
[('ORD', 0.7335404071218593),
 ('MEM', 0.7216129208271949),
 ('ATL', 0.7216129208271949),
 ('LAS', 0.7216129208271949),
 ('PHX', 0.7157934617882659)]
```

The resulting network graphs are very complex as anticipated since U.S. airport traffic is a highly interlinked and sophisticated system. However, the network graphs and node closeness measure clearly indicate what the major airports are in the United States. For passenger traffic, ATL, ORD, LAS, DEN, DFW, LAX, MCO, and CLT are all major airports. For freight traffic, MEM, ORD, SDF, ANC, and IND are all major airports. It is also clear that passenger traffic is a more complex network than freight traffic. This is not entirely surprising given the flexibility of travel for passengers, verse likely more defined routes for freight traffic.

Conclusion

In conclusion, Python was utilized to analyze passenger traffic and freight traffic for the top 150 U.S. airports from January to July 2022. The dataset was pulled from the Bureau of Transportation Statistics which maintains public data for all flights in the United States. The resulting network graphs revealed a highly complex and interlinked system for traffic between U.S. airports. However, it was noted the passenger network was more sophisticated and interlinked than the freight network, which is not surprising. The network graphs can potentially be utilized to optimize traffic for U.S. airports and future research is warranted on the topic.