In [325]:
```python
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
data = pd.read_csv('Airlines_graph.csv')
```

In [326]:
```python
import matplotlib.pyplot as plt
import networkx as nx
data.shape
data.dtypes
```

Out[326]:
```
year                int64
month               int64
day                 int64
dep_time          float64
sched_dep_time      int64
dep_delay         float64
arr_time          float64
sched_arr_time      int64
arr_delay         float64
carrier            object
flight              int64
tailnum            object
origin             object
dest               object
air_time          float64
distance            int64
dtype: object
```

In [327]:
```python
nx.__version__
```

Out[327]:
```
'2.5'
```

In [328]:
```python
# converting sched_dep_time to 'std' - Scheduled time of departure
data['std'] = data.sched_dep_time.astype(str).str.replace('(\d{2}$)', '') + ':' + data.sched_dep_time.astype(str).str
```

In [329]:
```python
# converting sched_arr_time to 'sta' - Scheduled time of arrival
data['sta'] = data.sched_arr_time.astype(str).str.replace('(\d{2}$)', '') + ':' + data.sched_arr_time.astype(str).str

# converting dep_time to 'atd' - Actual time of departure
data['atd'] = data.dep_time.fillna(0).astype(np.int64).astype(str).str.replace('(\d{2}$)', '') + ':' + data.dep_time.
```

In [330]:
```python
# converting arr_time to 'ata' - Actual time of arrival
data['ata'] = data.arr_time.fillna(0).astype(np.int64).astype(str).str.replace('(\d{2}$)', '') + ':' + data.arr_time.
```

In [331]:
```python
data['date'] = pd.to_datetime(data[['year', 'month', 'day']])
```

In [332]:
```python
# finally we drop the columns we don't need
data = data.drop(columns = ['year', 'month', 'day'])
```

In [333]:
```python
data["origin"].value_counts()
```

Out[333]:
```
LGA    42
EWR    31
JFK    27
Name: origin, dtype: int64
```

In [334]:
```python
len(data["dest"].value_counts())
```

Out[334]:
```
33
```

In [335]:
```python
FG = nx.from_pandas_edgelist(data, source='origin', target='dest', edge_attr=True,)
```

In [336]:
```python
len(FG.nodes())
```

Out[336]: 36

In [337]:
```python
len(FG.edges())
```

Out[337]: 57

In [338]:
```python
FG.nodes()
```

Out[338]:
```
NodeView(('EWR', 'MEM', 'LGA', 'FLL', 'SEA', 'JFK', 'DEN', 'ORD', 'MIA', 'PBI', 'MCO', 'CMH', 'MSP', 'IAD', 'CLT',
'TPA', 'DCA', 'SJU', 'ATL', 'BHM', 'SRQ', 'MSY', 'DTW', 'LAX', 'JAX', 'RDU', 'MDW', 'DFW', 'IAH', 'SFO', 'STL', 'CV
G', 'IND', 'RSW', 'BOS', 'CLE'))
```

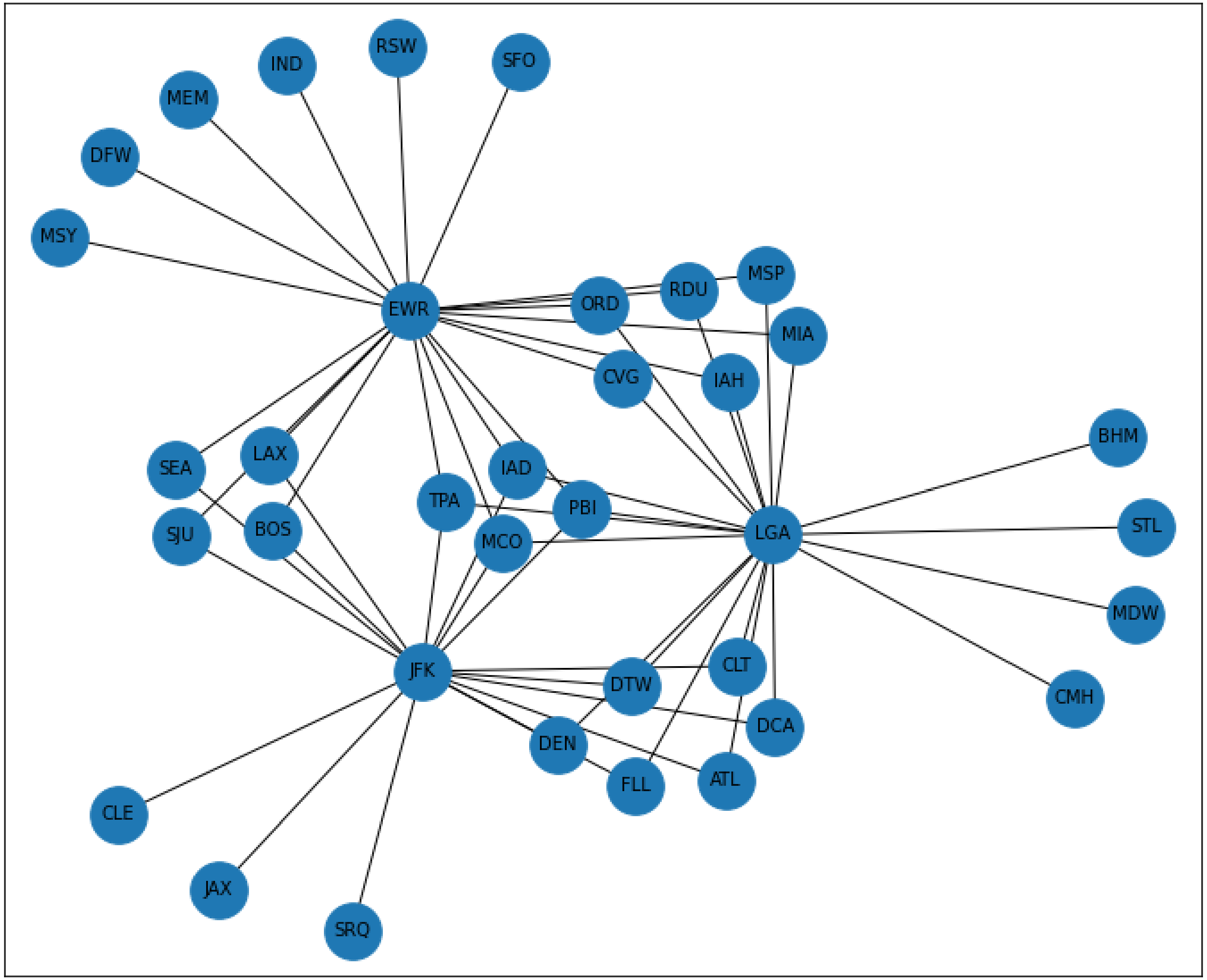
In [339]:
```python
# show how 36 connected by 57 edges
FG.edges()
```

Out[339]:
```
EdgeView([('EWR', 'MEM'), ('EWR', 'SEA'), ('EWR', 'MIA'), ('EWR', 'ORD'), ('EWR', 'MSP'), ('EWR', 'TPA'), ('EWR',
'MSY'), ('EWR', 'DFW'), ('EWR', 'IAH'), ('EWR', 'SFO'), ('EWR', 'CVG'), ('EWR', 'IND'), ('EWR', 'RDU'), ('EWR', 'IA
D'), ('EWR', 'RSW'), ('EWR', 'BOS'), ('EWR', 'PBI'), ('EWR', 'LAX'), ('EWR', 'MCO'), ('EWR', 'SJU'), ('LGA', 'FL
L'), ('LGA', 'ORD'), ('LGA', 'PBI'), ('LGA', 'CMH'), ('LGA', 'IAD'), ('LGA', 'CLT'), ('LGA', 'MIA'), ('LGA', 'DC
A'), ('LGA', 'BHM'), ('LGA', 'RDU'), ('LGA', 'ATL'), ('LGA', 'TPA'), ('LGA', 'MDW'), ('LGA', 'DEN'), ('LGA', 'MS
P'), ('LGA', 'DTW'), ('LGA', 'STL'), ('LGA', 'MCO'), ('LGA', 'CVG'), ('LGA', 'IAH'), ('FLL', 'JFK'), ('SEA', 'JF
K'), ('JFK', 'DEN'), ('JFK', 'MCO'), ('JFK', 'TPA'), ('JFK', 'SJU'), ('JFK', 'ATL'), ('JFK', 'SRQ'), ('JFK', 'DC
A'), ('JFK', 'DTW'), ('JFK', 'LAX'), ('JFK', 'JAX'), ('JFK', 'CLT'), ('JFK', 'PBI'), ('JFK', 'CLE'), ('JFK', 'IA
D'), ('JFK', 'BOS')])
```

In [340]:

```
# your code is here (Quick view of the Graph.)
for i in list(FG.edges):
    FG.add_edge(i[0],i[1])

plt.figure(figsize=(12,10))
nx.draw_networkx(FG, node_size = 1000, font_size=10)
plt.show()
```



In [341]:
```python
max(dict(nx.eccentricity(FG)).values())
```
Out[341]: 4

In [342]:
```python
nx.algorithms.degree_centrality(FG) # Notice the 3 airports from which all of our 100 rows of data originates
# Calculate average edge density of the Graph

np.mean(list(nx.algorithms.degree_centrality(FG).values()))
# your code is here
```
Out[342]: 0.09047619047619045

In [343]:
```python
nx.average_shortest_path_length(FG) # Average shortest path length for ALL paths in the Graph
```
Out[343]: 2.36984126984127

In [344]:
```python
nx.average_degree_connectivity(FG) # For a node of degree k - What is the average of its neighbours' degree?
```
Out[344]: {20: 1.95, 1: 19.307692307692307, 2: 19.0625, 17: 2.0588235294117645, 3: 19.0}

In [345]:
```python
# Let us find the dijkstra path from JAX to DFW.
# You can read more in-depth on how dijkstra works from this resource - https://courses.csail.mit.edu/6.006/fall11/le
dijpath = nx.dijkstra_path(FG, source='JAX', target='DFW')
dijpath
```
Out[345]: ['JAX', 'JFK', 'SEA', 'EWR', 'DFW']

In [346]:
```python
# Let us try to find the dijkstra path weighted by airtime (approximate case)
shortpath = nx.dijkstra_path(FG, source='JAX', target='DFW', weight='air_time')
shortpath
```

Out[346]: ['JAX', 'JFK', 'BOS', 'EWR', 'DFW']

In [347]:
```python
# if we don't consider airtime
nx.shortest_path(FG, 'JAX', 'DFW')
```

Out[347]: ['JAX', 'JFK', 'SEA', 'EWR', 'DFW']

In [348]:

```python
for path in nx.all_simple_paths(FG, source='JAX', target='DFW'):
    print(path)
```

```
['JAX', 'JFK', 'DEN', 'LGA', 'ORD', 'EWR', 'DFW']
['JAX', 'JFK', 'DEN', 'LGA', 'PBI', 'EWR', 'DFW']
['JAX', 'JFK', 'DEN', 'LGA', 'IAD', 'EWR', 'DFW']
['JAX', 'JFK', 'DEN', 'LGA', 'MIA', 'EWR', 'DFW']
['JAX', 'JFK', 'DEN', 'LGA', 'RDU', 'EWR', 'DFW']
['JAX', 'JFK', 'DEN', 'LGA', 'TPA', 'EWR', 'DFW']
['JAX', 'JFK', 'DEN', 'LGA', 'MSP', 'EWR', 'DFW']
['JAX', 'JFK', 'DEN', 'LGA', 'MCO', 'EWR', 'DFW']
['JAX', 'JFK', 'DEN', 'LGA', 'CVG', 'EWR', 'DFW']
['JAX', 'JFK', 'DEN', 'LGA', 'IAH', 'EWR', 'DFW']
['JAX', 'JFK', 'SEA', 'EWR', 'DFW']
['JAX', 'JFK', 'MCO', 'LGA', 'ORD', 'EWR', 'DFW']
['JAX', 'JFK', 'MCO', 'LGA', 'PBI', 'EWR', 'DFW']
['JAX', 'JFK', 'MCO', 'LGA', 'IAD', 'EWR', 'DFW']
['JAX', 'JFK', 'MCO', 'LGA', 'MIA', 'EWR', 'DFW']
['JAX', 'JFK', 'MCO', 'LGA', 'RDU', 'EWR', 'DFW']
['JAX', 'JFK', 'MCO', 'LGA', 'TPA', 'EWR', 'DFW']
['JAX', 'JFK', 'MCO', 'LGA', 'MSP', 'EWR', 'DFW']
['JAX', 'JFK', 'MCO', 'LGA', 'CVG', 'EWR', 'DFW']
['JAX', 'JFK', 'MCO', 'LGA', 'IAH', 'EWR', 'DFW']
['JAX', 'JFK', 'MCO', 'EWR', 'DFW']
['JAX', 'JFK', 'TPA', 'EWR', 'DFW']
['JAX', 'JFK', 'TPA', 'LGA', 'ORD', 'EWR', 'DFW']
['JAX', 'JFK', 'TPA', 'LGA', 'PBI', 'EWR', 'DFW']
['JAX', 'JFK', 'TPA', 'LGA', 'IAD', 'EWR', 'DFW']
['JAX', 'JFK', 'TPA', 'LGA', 'MIA', 'EWR', 'DFW']
['JAX', 'JFK', 'TPA', 'LGA', 'RDU', 'EWR', 'DFW']
['JAX', 'JFK', 'TPA', 'LGA', 'MSP', 'EWR', 'DFW']
['JAX', 'JFK', 'TPA', 'LGA', 'MCO', 'EWR', 'DFW']
['JAX', 'JFK', 'TPA', 'LGA', 'CVG', 'EWR', 'DFW']
['JAX', 'JFK', 'TPA', 'LGA', 'IAH', 'EWR', 'DFW']
['JAX', 'JFK', 'SJU', 'EWR', 'DFW']
['JAX', 'JFK', 'ATL', 'LGA', 'ORD', 'EWR', 'DFW']
['JAX', 'JFK', 'ATL', 'LGA', 'PBI', 'EWR', 'DFW']
['JAX', 'JFK', 'ATL', 'LGA', 'IAD', 'EWR', 'DFW']
['JAX', 'JFK', 'ATL', 'LGA', 'MIA', 'EWR', 'DFW']
['JAX', 'JFK', 'ATL', 'LGA', 'RDU', 'EWR', 'DFW']
```

```
['JAX', 'JFK', 'ATL', 'LGA', 'TPA', 'EWR', 'DFW']
['JAX', 'JFK', 'ATL', 'LGA', 'MSP', 'EWR', 'DFW']
['JAX', 'JFK', 'ATL', 'LGA', 'MCO', 'EWR', 'DFW']
['JAX', 'JFK', 'ATL', 'LGA', 'CVG', 'EWR', 'DFW']
['JAX', 'JFK', 'ATL', 'LGA', 'IAH', 'EWR', 'DFW']
['JAX', 'JFK', 'DCA', 'LGA', 'ORD', 'EWR', 'DFW']
['JAX', 'JFK', 'DCA', 'LGA', 'PBI', 'EWR', 'DFW']
['JAX', 'JFK', 'DCA', 'LGA', 'IAD', 'EWR', 'DFW']
['JAX', 'JFK', 'DCA', 'LGA', 'MIA', 'EWR', 'DFW']
['JAX', 'JFK', 'DCA', 'LGA', 'RDU', 'EWR', 'DFW']
['JAX', 'JFK', 'DCA', 'LGA', 'TPA', 'EWR', 'DFW']
['JAX', 'JFK', 'DCA', 'LGA', 'MSP', 'EWR', 'DFW']
['JAX', 'JFK', 'DCA', 'LGA', 'MCO', 'EWR', 'DFW']
['JAX', 'JFK', 'DCA', 'LGA', 'CVG', 'EWR', 'DFW']
['JAX', 'JFK', 'DCA', 'LGA', 'IAH', 'EWR', 'DFW']
['JAX', 'JFK', 'DTW', 'LGA', 'ORD', 'EWR', 'DFW']
['JAX', 'JFK', 'DTW', 'LGA', 'PBI', 'EWR', 'DFW']
['JAX', 'JFK', 'DTW', 'LGA', 'IAD', 'EWR', 'DFW']
['JAX', 'JFK', 'DTW', 'LGA', 'MIA', 'EWR', 'DFW']
['JAX', 'JFK', 'DTW', 'LGA', 'RDU', 'EWR', 'DFW']
['JAX', 'JFK', 'DTW', 'LGA', 'TPA', 'EWR', 'DFW']
['JAX', 'JFK', 'DTW', 'LGA', 'MSP', 'EWR', 'DFW']
['JAX', 'JFK', 'DTW', 'LGA', 'MCO', 'EWR', 'DFW']
['JAX', 'JFK', 'DTW', 'LGA', 'CVG', 'EWR', 'DFW']
['JAX', 'JFK', 'DTW', 'LGA', 'IAH', 'EWR', 'DFW']
['JAX', 'JFK', 'LAX', 'EWR', 'DFW']
['JAX', 'JFK', 'FLL', 'LGA', 'ORD', 'EWR', 'DFW']
['JAX', 'JFK', 'FLL', 'LGA', 'PBI', 'EWR', 'DFW']
['JAX', 'JFK', 'FLL', 'LGA', 'IAD', 'EWR', 'DFW']
['JAX', 'JFK', 'FLL', 'LGA', 'MIA', 'EWR', 'DFW']
['JAX', 'JFK', 'FLL', 'LGA', 'RDU', 'EWR', 'DFW']
['JAX', 'JFK', 'FLL', 'LGA', 'TPA', 'EWR', 'DFW']
['JAX', 'JFK', 'FLL', 'LGA', 'MSP', 'EWR', 'DFW']
['JAX', 'JFK', 'FLL', 'LGA', 'MCO', 'EWR', 'DFW']
['JAX', 'JFK', 'FLL', 'LGA', 'CVG', 'EWR', 'DFW']
['JAX', 'JFK', 'FLL', 'LGA', 'IAH', 'EWR', 'DFW']
['JAX', 'JFK', 'CLT', 'LGA', 'ORD', 'EWR', 'DFW']
['JAX', 'JFK', 'CLT', 'LGA', 'PBI', 'EWR', 'DFW']
['JAX', 'JFK', 'CLT', 'LGA', 'IAD', 'EWR', 'DFW']
['JAX', 'JFK', 'CLT', 'LGA', 'MIA', 'EWR', 'DFW']
['JAX', 'JFK', 'CLT', 'LGA', 'RDU', 'EWR', 'DFW']
['JAX', 'JFK', 'CLT', 'LGA', 'TPA', 'EWR', 'DFW']
```

```
['JAX', 'JFK', 'CLT', 'LGA', 'MSP', 'EWR', 'DFW']
['JAX', 'JFK', 'CLT', 'LGA', 'MCO', 'EWR', 'DFW']
['JAX', 'JFK', 'CLT', 'LGA', 'CVG', 'EWR', 'DFW']
['JAX', 'JFK', 'CLT', 'LGA', 'IAH', 'EWR', 'DFW']
['JAX', 'JFK', 'PBI', 'LGA', 'ORD', 'EWR', 'DFW']
['JAX', 'JFK', 'PBI', 'LGA', 'IAD', 'EWR', 'DFW']
['JAX', 'JFK', 'PBI', 'LGA', 'MIA', 'EWR', 'DFW']
['JAX', 'JFK', 'PBI', 'LGA', 'RDU', 'EWR', 'DFW']
['JAX', 'JFK', 'PBI', 'LGA', 'TPA', 'EWR', 'DFW']
['JAX', 'JFK', 'PBI', 'LGA', 'MSP', 'EWR', 'DFW']
['JAX', 'JFK', 'PBI', 'LGA', 'MCO', 'EWR', 'DFW']
['JAX', 'JFK', 'PBI', 'LGA', 'CVG', 'EWR', 'DFW']
['JAX', 'JFK', 'PBI', 'LGA', 'IAH', 'EWR', 'DFW']
['JAX', 'JFK', 'PBI', 'EWR', 'DFW']
['JAX', 'JFK', 'IAD', 'LGA', 'ORD', 'EWR', 'DFW']
['JAX', 'JFK', 'IAD', 'LGA', 'PBI', 'EWR', 'DFW']
['JAX', 'JFK', 'IAD', 'LGA', 'MIA', 'EWR', 'DFW']
['JAX', 'JFK', 'IAD', 'LGA', 'RDU', 'EWR', 'DFW']
['JAX', 'JFK', 'IAD', 'LGA', 'TPA', 'EWR', 'DFW']
['JAX', 'JFK', 'IAD', 'LGA', 'MSP', 'EWR', 'DFW']
['JAX', 'JFK', 'IAD', 'LGA', 'MCO', 'EWR', 'DFW']
['JAX', 'JFK', 'IAD', 'LGA', 'CVG', 'EWR', 'DFW']
['JAX', 'JFK', 'IAD', 'LGA', 'IAH', 'EWR', 'DFW']
['JAX', 'JFK', 'IAD', 'EWR', 'DFW']
['JAX', 'JFK', 'BOS', 'EWR', 'DFW']
```

ASSIGNMENT-4 (100 Points)

Please use the Airlines_graph.csv for the following questions.

1. Please fill "your code here" sections on above cells (10 Points).
2. How many maximal cliques we can spot in this airline network? (20 Points)
3. List the most busiest/popular airport. (20 Points)
4. As a thought leader, identify 6 new routes to recommend. Hint: Think if the pairs are symmetric or not and make your assumption/observation accordingly i.e. whether ORD-LAX and LAX-ORD two separate routes? (50 Points)

2.How many maximal cliques we can spot in this airline network? (20 Points)

In [349]:

```
# Define maximal_cliques()
def maximal_cliques(G, size):
    """
    Finds all maximal cliques in graph `G` that are of size `size`.
    """
    mcs = []
    for clique in nx.find_cliques(G):
        if len(clique) == size:
            mcs.append(clique)
    return mcs
```

In [363]:

```python
for i in range(10,0,-1):
    if list(maximal_cliques(FG,i)) != []:
        print("max_cliques:",len(list(maximal_cliques(FG,2))))

list(maximal_cliques(FG,2))
```

```
max_cliques: 57
```

Out[363]:

```
[['SFO', 'EWR'],
 ['LGA', 'MIA'],
 ['LGA', 'ORD'],
 ['LGA', 'DCA'],
 ['LGA', 'DTW'],
 ['LGA', 'IAH'],
 ['LGA', 'CVG'],
 ['LGA', 'CLT'],
 ['LGA', 'DEN'],
 ['LGA', 'STL'],
 ['LGA', 'PBI'],
 ['LGA', 'BHM'],
 ['LGA', 'ATL'],
 ['LGA', 'MCO'],
 ['LGA', 'FLL'],
 ['LGA', 'MSP'],
 ['LGA', 'IAD'],
 ['LGA', 'TPA'],
 ['LGA', 'MDW'],
 ['LGA', 'RDU'],
 ['LGA', 'CMH'],
 ['JAX', 'JFK'],
 ['LAX', 'JFK'],
 ['LAX', 'EWR'],
 ['BOS', 'JFK'],
 ['BOS', 'EWR'],
 ['SEA', 'JFK'],
 ['SEA', 'EWR'],
 ['MEM', 'EWR'],
 ['MSY', 'EWR'],
 ['SRQ', 'JFK'],
 ['DFW', 'EWR'],
 ['RSW', 'EWR'],
```

```
['CLE', 'JFK'],
['IND', 'EWR'],
['JFK', 'CLT'],
['JFK', 'DEN'],
['JFK', 'DCA'],
['JFK', 'DTW'],
['JFK', 'ATL'],
['JFK', 'FLL'],
['JFK', 'SJU'],
['JFK', 'MCO'],
['JFK', 'IAD'],
['JFK', 'TPA'],
['JFK', 'PBI'],
['SJU', 'EWR'],
['EWR', 'MIA'],
['EWR', 'ORD'],
['EWR', 'MSP'],
['EWR', 'IAH'],
['EWR', 'CVG'],
['EWR', 'MCO'],
['EWR', 'IAD'],
['EWR', 'TPA'],
['EWR', 'RDU'],
['EWR', 'PBI']]
```

3. List the most busiest/popular airport. (20 Points)

In [352]:

```python
# EWR and LGA are two busiest/popular airport due to the most air route.
degree_dict = dict(nx.degree(FG))
dict(sorted(degree_dict.items(), key=lambda item: item[1],reverse=True))
```

Out[352]:

```
{'EWR': 20,
 'LGA': 20,
 'JFK': 17,
 'PBI': 3,
 'MCO': 3,
 'IAD': 3,
 'TPA': 3,
 'FLL': 2,
 'SEA': 2,
 'DEN': 2,
 'ORD': 2,
 'MIA': 2,
 'MSP': 2,
 'CLT': 2,
 'DCA': 2,
 'SJU': 2,
 'ATL': 2,
 'DTW': 2,
 'LAX': 2,
 'RDU': 2,
 'IAH': 2,
 'CVG': 2,
 'BOS': 2,
 'MEM': 1,
 'CMH': 1,
 'BHM': 1,
 'SRQ': 1,
 'MSY': 1,
 'JAX': 1,
 'MDW': 1,
 'DFW': 1,
 'SFO': 1,
 'STL': 1,
 'IND': 1,
 'RSW': 1,
 'CLE': 1}
```

In [353]:
```python
# EWR and LGA are two busiest/popular airport as they are most important node.
degree_cent_dict = nx.degree_centrality(FG)
most_popular_airport = dict(sorted(degree_cent_dict.items(),
                                    key=lambda item: item[1],reverse=True))
most_popular_airport
```

Out[353]:
```
{'EWR': 0.5714285714285714,
 'LGA': 0.5714285714285714,
 'JFK': 0.4857142857142857,
 'PBI': 0.08571428571428572,
 'MCO': 0.08571428571428572,
 'IAD': 0.08571428571428572,
 'TPA': 0.08571428571428572,
 'FLL': 0.05714285714285714,
 'SEA': 0.05714285714285714,
 'DEN': 0.05714285714285714,
 'ORD': 0.05714285714285714,
 'MIA': 0.05714285714285714,
 'MSP': 0.05714285714285714,
 'CLT': 0.05714285714285714,
 'DCA': 0.05714285714285714,
 'SJU': 0.05714285714285714,
 'ATL': 0.05714285714285714,
 'DTW': 0.05714285714285714,
 'LAX': 0.05714285714285714,
```

4.As a thought leader, identify 6 new routes to recommend. Hint: Think if the pairs are symmetric or not and make your assumption/observation accordingly i.e. whether ORD-LAX and LAX-ORD two separate routes? (50 Points)

In [354]:
```python
# I assume routes are undirected. ie. ORD-LAX & LAX-ORD are the same
# so use combination rather than permutation here
from itertools import combinations
comb = []

for i in combinations(list(FG.nodes()),2):
    comb.append(i)
```

In [355]:

```
# find any undeveloped routes
undeveloped_route = list(set(comb) - set(FG.edges()))
```

In [357]:

```python
# -------------------------strategy---------------------------

# recommend the highest 6th degree_centrality undeveloped routes.
# since developing these routes will help others most.
most_popular_airport
```

Out[357]:

```
{'EWR': 0.5714285714285714,
 'LGA': 0.5714285714285714,
 'JFK': 0.4857142857142857,
 'PBI': 0.08571428571428572,
 'MCO': 0.08571428571428572,
 'IAD': 0.08571428571428572,
 'TPA': 0.08571428571428572,
 'FLL': 0.05714285714285714,
 'SEA': 0.05714285714285714,
 'DEN': 0.05714285714285714,
 'ORD': 0.05714285714285714,
 'MIA': 0.05714285714285714,
 'MSP': 0.05714285714285714,
 'CLT': 0.05714285714285714,
 'DCA': 0.05714285714285714,
 'SJU': 0.05714285714285714,
 'ATL': 0.05714285714285714,
 'DTW': 0.05714285714285714,
 'LAX': 0.05714285714285714,
 'RDU': 0.05714285714285714,
 'IAH': 0.05714285714285714,
 'CVG': 0.05714285714285714,
 'BOS': 0.05714285714285714,
 'MEM': 0.02857142857142857,
 'CMH': 0.02857142857142857,
 'BHM': 0.02857142857142857,
 'SRQ': 0.02857142857142857,
 'MSY': 0.02857142857142857,
 'JAX': 0.02857142857142857,
 'MDW': 0.02857142857142857,
 'DFW': 0.02857142857142857,
 'SFO': 0.02857142857142857,
 'STL': 0.02857142857142857,
 'IND': 0.02857142857142857,
```

```
 'RSW': 0.028571428571428857,
 'CLE': 0.028571428571428857}
```

In [358]:

```
# convert to list types of for easier handling
most_popular_airport = [list(x) for x in list(most_popular_airport.items())]
undeveloped_route = [list(x) for x in undeveloped_route]
```

In [359]:

```
a = pd.DataFrame(undeveloped_route)
```

In [360]:

```
# calculate sum of degree centrality of each pair

for i in undeveloped_route:
    for j in most_popular_airport:
        if i[0] == j[0]:
            i[0] = j[1]
        if i[1] == j[0]:
            i[1] = j[1]

undeveloped_degree_centrality = [i[0]+i[1] for i in undeveloped_route]
```

In [361]:

```python
b = pd.DataFrame(undeveloped_degree_centrality)
undeveloped_degree_centrality = pd.concat([a,b],axis=1)
undeveloped_degree_centrality.columns = ["airport1","airport2","degree_centrality"]
undeveloped_degree_centrality.sort_values("degree_centrality",ascending=False).head(6)
```

Out[361]:

	airport1	airport2	degree_centrality
159	EWR	LGA	1.142857
320	EWR	JFK	1.057143
27	LGA	JFK	1.057143
240	EWR	DCA	0.628571
252	EWR	DTW	0.628571
462	LGA	SJU	0.628571