Assignment 7.1

a.

In [1]:
```
import os
import sys
import gzip
import json
from pathlib import Path
import numpy as np
import pandas as pd
import pygeohash
import math
```

In [2]:
```
path = 'C:/Users/bibek/Documents/GitHub/dsc650/dsc650/assignments/assignment03/results'
df_file = path +"/routes.parquet"
df_file
```

Out[2]:
```
'C:/Users/bibek/Documents/GitHub/dsc650/dsc650/assignments/assignment03/results/routes.parquet'
```

In [3]:
```
current_dir = Path(os.getcwd()).absolute()
current_dir
```

Out[3]:
```
WindowsPath('C:/Users/bibek/Documents/GitHub/dsc650/dsc650/assignments/assignment07')
```

In [4]:
```
df = pd.read_parquet(df_file)
```

In [6]:
```
df
```

Out[6]:

	airline	src_airport	dst_airport	codeshare	equipment
0	{'airline_id': 410, 'name': 'Aerocondor', 'ali...	{'airport_id': 2965.0, 'name': 'Sochi Internat...	{'airport_id': 2990.0, 'name': 'Kazan Internat...	False	[CR2]
1	{'airline_id': 410, 'name': 'Aerocondor', 'ali...	{'airport_id': 2966.0, 'name': 'Astrakhan Airp...	{'airport_id': 2990.0, 'name': 'Kazan Internat...	False	[CR2]
2	{'airline_id': 410, 'name': 'Aerocondor', 'ali...	{'airport_id': 2966.0, 'name': 'Astrakhan Airp...	{'airport_id': 2962.0, 'name': 'Mineralnyye Vo...	False	[CR2]
3	{'airline_id': 410, 'name': 'Aerocondor', 'ali...	{'airport_id': 2968.0, 'name': 'Chelyabinsk Ba...	{'airport_id': 2990.0, 'name': 'Kazan Internat...	False	[CR2]
4	{'airline_id': 410, 'name': 'Aerocondor', 'ali...	{'airport_id': 2968.0, 'name': 'Chelyabinsk Ba...	{'airport_id': 4078.0, 'name': 'Tolmachevo Air...	False	[CR2]
...	...	...	...	...	...
67658	{'airline_id': 4178, 'name': 'Regional Express...	{'airport_id': 6334.0, 'name': 'Whyalla Airpor...	{'airport_id': 3341.0, 'name': 'Adelaide Inter...	False	[SF3]
67659	{'airline_id': 19016, 'name': 'Apache Air', 'a...	{'airport_id': 4029.0, 'name': 'Domodedovo Int...	{'airport_id': 2912.0, 'name': 'Manas Internat...	False	[734]
67660	{'airline_id': 19016, 'name': 'Apache Air', 'a...	{'airport_id': 2912.0, 'name': 'Manas Internat...	{'airport_id': 4029.0, 'name': 'Domodedovo Int...	False	[734]
67661	{'airline_id': 19016, 'name': 'Apache Air', 'a...	{'airport_id': 2912.0, 'name': 'Manas Internat...	{'airport_id': 2913.0, 'name': 'Osh Airport', ...	False	[734]
67662	{'airline_id': 19016, 'name': 'Apache Air', 'a...	{'airport_id': 2913.0, 'name': 'Osh Airport', ...	{'airport_id': 2912.0, 'name': 'Manas Internat...	False	[734]

67663 rows × 5 columns

In [7]:
```
nan_value = float("NaN")
df.replace("", nan_value, inplace=True)
df.dropna(inplace=True)
```

```
C:\Users\bibek\anaconda3\lib\site-packages\pandas\core\missing.py:49: FutureW
arning: elementwise comparison failed; returning scalar instead, but in the f
uture will perform elementwise comparison
  mask = arr == x
```

In [8]:
```
df
```

Out[8]:

	airline	src_airport	dst_airport	codeshare	equipment
0	{'airline_id': 410, 'name': 'Aerocondor', 'ali...	{'airport_id': 2965.0, 'name': 'Sochi Internat...	{'airport_id': 2990.0, 'name': 'Kazan Internat...	False	[CR2]
1	{'airline_id': 410, 'name': 'Aerocondor', 'ali...	{'airport_id': 2966.0, 'name': 'Astrakhan Airp...	{'airport_id': 2990.0, 'name': 'Kazan Internat...	False	[CR2]
2	{'airline_id': 410, 'name': 'Aerocondor', 'ali...	{'airport_id': 2966.0, 'name': 'Astrakhan Airp...	{'airport_id': 2962.0, 'name': 'Mineralnyye Vo...	False	[CR2]
3	{'airline_id': 410, 'name': 'Aerocondor', 'ali...	{'airport_id': 2968.0, 'name': 'Chelyabinsk Ba...	{'airport_id': 2990.0, 'name': 'Kazan Internat...	False	[CR2]
4	{'airline_id': 410, 'name': 'Aerocondor', 'ali...	{'airport_id': 2968.0, 'name': 'Chelyabinsk Ba...	{'airport_id': 4078.0, 'name': 'Tolmachevo Air...	False	[CR2]
...	...	...	...	...	...
67658	{'airline_id': 4178, 'name': 'Regional Express...	{'airport_id': 6334.0, 'name': 'Whyalla Airpor...	{'airport_id': 3341.0, 'name': 'Adelaide Inter...	False	[SF3]
67659	{'airline_id': 19016, 'name': 'Apache Air', 'a...	{'airport_id': 4029.0, 'name': 'Domodedovo Int...	{'airport_id': 2912.0, 'name': 'Manas Internat...	False	[734]
67660	{'airline_id': 19016, 'name': 'Apache Air', 'a...	{'airport_id': 2912.0, 'name': 'Manas Internat...	{'airport_id': 4029.0, 'name': 'Domodedovo Int...	False	[734]
67661	{'airline_id': 19016, 'name': 'Apache Air', 'a...	{'airport_id': 2912.0, 'name': 'Manas Internat...	{'airport_id': 2913.0, 'name': 'Osh Airport', ...	False	[734]
67662	{'airline_id': 19016, 'name': 'Apache Air', 'a...	{'airport_id': 2913.0, 'name': 'Osh Airport', ...	{'airport_id': 2912.0, 'name': 'Manas Internat...	False	[734]

66771 rows × 5 columns

In [12]:
```
def keys(src_airport, dst_airport, airline):
    src = src_airport.get('iata')
    dst = dst_airport.get('iata')
    airline = airline.get('iata')
    if src == "":
        return None
    elif dst =="":
        return None
    else:
        key = '{}{}{}'.format(src, dst, airline)

        return key
```

In [13]:
```
df['key'] = df.apply(lambda row : keys(row['src_airport'],row['dst_airport'], row['airline']), axis = 1)
```

In [23]:
```
df.dropna(inplace=True)
df
```

Out[23]:

	airline	src_airport	dst_airport	codeshare	equipment	key
0	{'airline_id': 410, 'name': 'Aerocondor', 'ali...	{'airport_id': 2965.0, 'name': 'Sochi Internat...	{'airport_id': 2990.0, 'name': 'Kazan Internat...	False	[CR2]	AERKZN2B
1	{'airline_id': 410, 'name': 'Aerocondor', 'ali...	{'airport_id': 2966.0, 'name': 'Astrakhan Airp...	{'airport_id': 2990.0, 'name': 'Kazan Internat...	False	[CR2]	ASFKZN2B
2	{'airline_id': 410, 'name': 'Aerocondor', 'ali...	{'airport_id': 2966.0, 'name': 'Astrakhan Airp...	{'airport_id': 2962.0, 'name': 'Mineralnyye Vo...	False	[CR2]	ASFMRV2B
3	{'airline_id': 410, 'name': 'Aerocondor', 'ali...	{'airport_id': 2968.0, 'name': 'Chelyabinsk Ba...	{'airport_id': 2990.0, 'name': 'Kazan Internat...	False	[CR2]	CEKKZN2B
4	{'airline_id': 410, 'name': 'Aerocondor', 'ali...	{'airport_id': 2968.0, 'name': 'Chelyabinsk Ba...	{'airport_id': 4078.0, 'name': 'Tolmachevo Air...	False	[CR2]	CEKOVB2B
...	...	...	...	...	...	...
67658	{'airline_id': 4178, 'name': 'Regional Express...	{'airport_id': 6334.0, 'name': 'Whyalla Airpor...	{'airport_id': 3341.0, 'name': 'Adelaide Inter...	False	[SF3]	WYAADLZL
67659	{'airline_id': 19016, 'name': 'Apache Air', 'a...	{'airport_id': 4029.0, 'name': 'Domodedovo Int...	{'airport_id': 2912.0, 'name': 'Manas Internat...	False	[734]	DMEFRUZM
67660	{'airline_id': 19016, 'name': 'Apache Air', 'a...	{'airport_id': 2912.0, 'name': 'Manas Internat...	{'airport_id': 4029.0, 'name': 'Domodedovo Int...	False	[734]	FRUDMEZM
67661	{'airline_id': 19016, 'name': 'Apache Air', 'a...	{'airport_id': 2912.0, 'name': 'Manas Internat...	{'airport_id': 2913.0, 'name': 'Osh Airport', ...	False	[734]	FRUOSSZM
67662	{'airline_id': 19016, 'name': 'Apache Air', 'a...	{'airport_id': 2913.0, 'name': 'Osh Airport', ...	{'airport_id': 2912.0, 'name': 'Manas Internat...	False	[734]	OSSFRUZM

66565 rows × 6 columns

In [24]:
```
def partition_keys(key):
    part_dict = {"A":"A","B":"B","C":"C-D","D":"C-D",
                 "E":"E-F","F":"E-F","G":"G-H","H":"G-H",
                 "I":"I-J","J":"I-J","K":"K-L","L":"K-L",
                 "M":"M","N":"N","O":"O-P","P":"O-P",
                 "Q":"Q-R","R":"Q-R","S":"S-T","T":"S-T",
                 "U":"U","V":"V","W":"W-X","X":"W-X",
                 "Y":"Y-Z","Z":"Y-Z"}

    key = str(key)
    k = key[0]
    kv_key = part_dict[k]
    return kv_key
```

In [26]:
```
df['kv_key'] = df.apply(lambda row : partition_keys(row['key']), axis = 1)
```

In [27]:
```
df.to_parquet('results/kv/',partition_cols=['kv_key'])
```

b.

In [34]:
```
import hashlib

def partition_hash_key(key):
    m = hashlib.sha256()
    m.update(str(key).encode('utf-8'))
    hash_key = m.hexdigest()
    hv_key = hash_key[0]
    return hv_key
```

In [35]:
```
df['hash_key'] = df.apply(lambda row : partition_hash_key(row['key']), axis =
1)
```

In [37]:
```
df.to_parquet('results/hash/',partition_cols=['hash_key'])
```

c.

In [40]:
```
# create data center dictionary
datacenters = [
    {
        "location": "west",
        "city": "The Dalles, Oregon",
        "latitude": 45.5945645,
        "longitude": -121.1786823
    },
    {
        "location": "central",
        "city": "Papillion, NE",
        "latitude": 41.1544433,
        "longitude": -96.0422378
    },
    {
        "location": "east",
        "city": "Loudoun County, Virginia",
        "latitude": 39.08344,
        "longitude": -77.6497145
    }
]
```

In [41]:
```
# adding geohash to all datacenter
for datacenter in datacenters:
    datacenter['geohash'] = pygeohash.encode(datacenter['latitude'], datacente
r['longitude'])
```

In [42]:
```
datacenters
```

Out[42]:
```
[{'location': 'west',
  'city': 'The Dalles, Oregon',
  'latitude': 45.5945645,
  'longitude': -121.1786823,
  'geohash': 'c21g6s0rs4c7'},
 {'location': 'central',
  'city': 'Papillion, NE',
  'latitude': 41.1544433,
  'longitude': -96.0422378,
  'geohash': '9z7dnebnj8kb'},
 {'location': 'east',
  'city': 'Loudoun County, Virginia',
  'latitude': 39.08344,
  'longitude': -77.6497145,
  'geohash': 'dqby34cjw922'}]
```

In [80]:
```
def datacenter_search(geohash, datacenters):
    Distance = []
    location_dist = {} # dictionary for geohash and distance of airport
    for datacenter in datacenters:
        center_geohash = datacenter['geohash']
        d = pygeohash.geohash_haversine_distance(geohash, center_geohash)/1000
        Distance.append(d)
        location_dist[datacenter['location']] = d

    for k, v in location_dist.items():
        if v == min(Distance): #finds minimum distance
            location = k
    return location
```

In [81]:
```
def datacenter_key(src_airport):
    src_lat = src_airport.get('latitude')
    src_long = src_airport.get('longitude')
    geohash = pygeohash.encode(src_lat, src_long)
    location = datacenter_search(geohash, datacenters)
    return location
```

In [82]:
```
df['location'] = df.apply(lambda row : datacenter_key(row['src_airport']), axi
s = 1)
```

In [84]:
```
df.to_parquet('results/geo/',partition_cols=['location'])
```

d.

In [85]:
```
def balance_partitions(keys, partition_count):
    keys = sorted(keys)
    # Calculate the size of the parititions with the remainders
    partition_size, remainder = np.divmod(len(keys), partition_count)

    # list to add the partitions
    partitions = []

    for i in np.arange(partition_count):

        # Find the starting place to index the partition
        start = i * partition_size + min(i, remainder)

        # Find the ending spot for the partition
        finish = (i + 1) * partition_size + min(i + 1, remainder)

        # Add the sorted partition to the list
        partitions.append(sorted(keys[start:finish]))

    # Return the partitions
    return(partitions)
```

In [86]:
```
# Generate a list of keys
keys = ['9','5','a','b','c','d','e','f','g','1','2','3']

for i in range(1,len(keys)+1):
    print(f'{str(i)+" partitions":<15}')
    for j in np.arange(0,i):
        print(f' {j+1:>3} : {balance_partitions(keys,i)[j]}', end = "\n")
    print("")
```

```
12 keys : ['9', '5', 'a', 'b', 'c', 'd', 'e', 'f', 'g', '1', '2', '3']

1 partitions
   1 : ['1', '2', '3', '5', '9', 'a', 'b', 'c', 'd', 'e', 'f', 'g']

2 partitions
   1 : ['1', '2', '3', '5', '9', 'a']
   2 : ['b', 'c', 'd', 'e', 'f', 'g']

3 partitions
   1 : ['1', '2', '3', '5']
   2 : ['9', 'a', 'b', 'c']
   3 : ['d', 'e', 'f', 'g']

4 partitions
   1 : ['1', '2', '3']
   2 : ['5', '9', 'a']
   3 : ['b', 'c', 'd']
   4 : ['e', 'f', 'g']

5 partitions
   1 : ['1', '2', '3']
   2 : ['5', '9', 'a']
   3 : ['b', 'c']
   4 : ['d', 'e']
   5 : ['f', 'g']

6 partitions
   1 : ['1', '2']
   2 : ['3', '5']
   3 : ['9', 'a']
   4 : ['b', 'c']
   5 : ['d', 'e']
   6 : ['f', 'g']

7 partitions
   1 : ['1', '2']
   2 : ['3', '5']
   3 : ['9', 'a']
   4 : ['b', 'c']
   5 : ['d', 'e']
   6 : ['f']
   7 : ['g']

8 partitions
   1 : ['1', '2']
   2 : ['3', '5']
   3 : ['9', 'a']
   4 : ['b', 'c']
   5 : ['d']
   6 : ['e']
   7 : ['f']
   8 : ['g']

9 partitions
   1 : ['1', '2']
   2 : ['3', '5']
```

```
   3 : ['9', 'a']
   4 : ['b']
   5 : ['c']
   6 : ['d']
   7 : ['e']
   8 : ['f']
   9 : ['g']

10 partitions
   1 : ['1', '2']
   2 : ['3', '5']
   3 : ['9']
   4 : ['a']
   5 : ['b']
   6 : ['c']
   7 : ['d']
   8 : ['e']
   9 : ['f']
  10 : ['g']

11 partitions
   1 : ['1', '2']
   2 : ['3']
   3 : ['5']
   4 : ['9']
   5 : ['a']
   6 : ['b']
   7 : ['c']
   8 : ['d']
   9 : ['e']
  10 : ['f']
  11 : ['g']

12 partitions
   1 : ['1']
   2 : ['2']
   3 : ['3']
   4 : ['5']
   5 : ['9']
   6 : ['a']
   7 : ['b']
   8 : ['c']
   9 : ['d']
  10 : ['e']
  11 : ['f']
  12 : ['g']
```

In [ ]: