# Assignment07_Muley_Tushar

January 21, 2022

Name: Muley, Tushar

Assignment: Assignment 7

Date: January 30, 2022

**Assignment 7a**

[1]:
```
# import libraries
import os
from pathlib import Path
import shutil
import pandas as pd
import hashlib
import pygeohash
```

[2]:
```
# create results folder

current_dir = Path(os.getcwd()).absolute()
results_dir = current_dir.joinpath('results')

if results_dir.exists():
    shutil.rmtree(results_dir)

results_dir.mkdir(parents=True, exist_ok=True)
```

[4]:
```
df = pd.read_parquet('routes.parquet')
```

[5]:
```
# check
df.head()
```

[5]:
```
                                             airline  \
0  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
1  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
2  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
3  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
4  {'airline_id': 410, 'name': 'Aerocondor', 'ali…

                                         src_airport  \
0  {'airport_id': 2965.0, 'name': 'Sochi Internat…
1  {'airport_id': 2966.0, 'name': 'Astrakhan Airp…
2  {'airport_id': 2966.0, 'name': 'Astrakhan Airp…
3  {'airport_id': 2968.0, 'name': 'Chelyabinsk Ba…
4  {'airport_id': 2968.0, 'name': 'Chelyabinsk Ba…

                                         dst_airport  codeshare equipment
0  {'airport_id': 2990.0, 'name': 'Kazan Internat…      False     [CR2]
1  {'airport_id': 2990.0, 'name': 'Kazan Internat…      False     [CR2]
2  {'airport_id': 2962.0, 'name': 'Mineralnyye Vo…      False     [CR2]
3  {'airport_id': 2990.0, 'name': 'Kazan Internat…      False     [CR2]
4  {'airport_id': 4078.0, 'name': 'Tolmachevo Air…      False     [CR2]
```

[6]:

```
# check starting size of df

df.shape
```

[6]:

```
(67663, 5)
```

[7]:

```
# drop rows with empty source airport, destination airport, and airline

df = df.dropna(subset = ['src_airport', 'dst_airport', 'airline'])
```

[8]:

```
# check ending size of df

df.shape
```

[8]:

```
(66771, 5)
```

[9]:

```
# define function for key

def generate_key(df):
    src = df['src_airport'].get('iata')
    dst = df['dst_airport'].get('iata')
    airline = df['airline'].get('iata')
    key = str('{}{}{}'.format(src, dst, airline))
    return key
```

[10]:

```
# cenerate key column

df['key'] = df.apply(generate_key, axis=1)
```

[11]:

```
# check data

df.head()
```

[11]:

```
                                             airline  \
0  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
1  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
2  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
3  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
4  {'airline_id': 410, 'name': 'Aerocondor', 'ali…

                                         src_airport  \
0  {'airport_id': 2965.0, 'name': 'Sochi Internat…
1  {'airport_id': 2966.0, 'name': 'Astrakhan Airp…
2  {'airport_id': 2966.0, 'name': 'Astrakhan Airp…
3  {'airport_id': 2968.0, 'name': 'Chelyabinsk Ba…
4  {'airport_id': 2968.0, 'name': 'Chelyabinsk Ba…

                                         dst_airport  codeshare equipment  \
0  {'airport_id': 2990.0, 'name': 'Kazan Internat…      False     [CR2]
1  {'airport_id': 2990.0, 'name': 'Kazan Internat…      False     [CR2]
2  {'airport_id': 2962.0, 'name': 'Mineralnyye Vo…      False     [CR2]
3  {'airport_id': 2990.0, 'name': 'Kazan Internat…      False     [CR2]
4  {'airport_id': 4078.0, 'name': 'Tolmachevo Air…      False     [CR2]

        key
0  AERKZN2B
1  ASFKZN2B
2  ASFMRV2B
3  CEKKZN2B
4  CEKOVB2B
```

[13]:

```
# create kv_key

df['kv_key'] = df['key'].str[0]

# replace the kv_key based on partition

df['kv_key'] = df['kv_key'].replace({'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', 'O':␣
 ↪'O-P', 'P': 'O-P','Q': 'Q-R', 'R': 'Q-R',
                                      'S': 'S-T', 'T':'S-T','W': 'W-X', 'X':␣
 ↪'W-X', 'Y': 'Y-Z', 'Z':'Y-Z'})
```

[14]:

```
# check first few rows are correct
df.head()
```

[14]:

```
                                             airline  \
0  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
1  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
```

```
2  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
3  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
4  {'airline_id': 410, 'name': 'Aerocondor', 'ali…

                                         src_airport  \
0  {'airport_id': 2965.0, 'name': 'Sochi Internat…
1  {'airport_id': 2966.0, 'name': 'Astrakhan Airp…
2  {'airport_id': 2966.0, 'name': 'Astrakhan Airp…
3  {'airport_id': 2968.0, 'name': 'Chelyabinsk Ba…
4  {'airport_id': 2968.0, 'name': 'Chelyabinsk Ba…

                                         dst_airport  codeshare equipment  \
0  {'airport_id': 2990.0, 'name': 'Kazan Internat…      False     [CR2]
1  {'airport_id': 2990.0, 'name': 'Kazan Internat…      False     [CR2]
2  {'airport_id': 2962.0, 'name': 'Mineralnyye Vo…      False     [CR2]
3  {'airport_id': 2990.0, 'name': 'Kazan Internat…      False     [CR2]
4  {'airport_id': 4078.0, 'name': 'Tolmachevo Air…      False     [CR2]

        key kv_key
0  AERKZN2B      A
1  ASFKZN2B      A
2  ASFMRV2B      A
3  CEKKZN2B    C-D
4  CEKOVB2B    C-D
```

[15]:

```
# create directory structure

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

### Assginemt 7b

[16]:

```
# define fuction to create hash key

def hash_key(key):
    m = hashlib.sha256()
    m.update(str(key).encode('utf-8'))
    return m.hexdigest().upper()
```

[17]:

```
# generate hashed column and populate values

df['hashed'] = df.apply(hash_key, axis=1)
```

[18]:

```
# create hash_key

df['hash_key'] = df['hashed'].str[0]
```

```
[19]: # check data

      df.head()
```

```
[19]:                                              airline  \
      0  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
      1  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
      2  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
      3  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
      4  {'airline_id': 410, 'name': 'Aerocondor', 'ali…

                                             src_airport  \
      0  {'airport_id': 2965.0, 'name': 'Sochi Internat…
      1  {'airport_id': 2966.0, 'name': 'Astrakhan Airp…
      2  {'airport_id': 2966.0, 'name': 'Astrakhan Airp…
      3  {'airport_id': 2968.0, 'name': 'Chelyabinsk Ba…
      4  {'airport_id': 2968.0, 'name': 'Chelyabinsk Ba…

                                             dst_airport  codeshare equipment  \
      0  {'airport_id': 2990.0, 'name': 'Kazan Internat…      False     [CR2]
      1  {'airport_id': 2990.0, 'name': 'Kazan Internat…      False     [CR2]
      2  {'airport_id': 2962.0, 'name': 'Mineralnyye Vo…      False     [CR2]
      3  {'airport_id': 2990.0, 'name': 'Kazan Internat…      False     [CR2]
      4  {'airport_id': 4078.0, 'name': 'Tolmachevo Air…      False     [CR2]

              key kv_key                                           hashed hash_key
      0  AERKZN2B      A  6BE72CE1DF4C9891AA30336AF9AF50AEB2B6ADAFF48180…        6
      1  ASFKZN2B      A  E250BB3A1FDBA40235E3C7529A9924AD777631603448CD…        E
      2  ASFMRV2B      A  611CBF68C32694D98BF1A469FFAC950F15A5AA608C444D…        6
      3  CEKKZN2B    C-D  BB1FA222B179AA3E535ABEEECB8B692CECBF86C4EEBABC…        B
      4  CEKOVB2B    C-D  00E77E6BBE4310E0E29F3B9C7B02B43292C5EF78FD7D82…        0
```

```
[20]: # create directory structure

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

### Assignment 7c

```
[21]: # get geohash for datacenters

      datacenters = {}

      datacenters['west'] = pygeohash.encode(45.5945645, -121.1786823)
      datacenters['central'] = pygeohash.encode(41.1544433, -96.0422378)
      datacenters['east'] = pygeohash.encode(39.08344, -77.6497145)

      # print
```

```
print(datacenters)
```

```
{'west': 'c21g6s0rs4c7', 'central': '9z7dnebnj8kb', 'east': 'dqby34cjw922'}
```

[22]:

```
# define function to find closest data center
def closest_datacenter(df):
    latitude = df['src_airport'].get('latitude')
    longitude = df['src_airport'].get('longitude')
    geohash = pygeohash.encode(latitude, longitude)
    west_dist = pygeohash.geohash_approximate_distance(geohash,␣
 ↪datacenters['west'])
    east_dist = pygeohash.
 ↪geohash_approximate_distance(geohash,datacenters['east'])
    central_dist = pygeohash.
 ↪geohash_approximate_distance(geohash,datacenters['central'])
    min_dist = min(west_dist, east_dist, central_dist)

    if west_dist == min_dist:
        return 'west'
    if east_dist == min_dist:
        return 'east'
    return 'central'
```

[23]:

```
# generate location column

df['location'] = df.apply(closest_datacenter, axis=1)
```

[24]:

```
# check the counts of unique locations
df['location'].value_counts()
```

[24]:

```
west       51311
east        9980
central     5480
Name: location, dtype: int64
```

[25]:

```
# create directory structure

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

### Assginment 7d

[26]:

```
def balance_partitions(keys, num_partitions):
    keys.sort()
    partitions = []

    # get approximate number of keys per partition
    partition_size = int(len(keys) / num_partitions)
    for i in range(num_partitions):
        if i == (num_partitions-1): # remaining keys are added to the␣
 ↪last␣partition
            partitions.append(keys[i*partition_size:])
        else: partitions.append(keys[i*partition_size:(i+1)*partition_size])

    return partitions
```

[27]:

```
# define a list
keys = [1, 7, 2, 8, 3, 3, 8, 4, 4, 5, 5, 6, 6, 7, 1, 8, 3, 4, 1, 9, 6, 5, 1, 7,␣
 ↪7]

# define number of partitions
num_partitions = 5

# call function
partitions = balance_partitions(keys, num_partitions)

# print it
print(partitions)
```

```
[[1, 1, 1, 1, 2], [3, 3, 3, 4, 4], [4, 5, 5, 5, 6], [6, 6, 7, 7, 7], [7, 8, 8,
8, 9]]
```

[ ]:

```

```