Assignment 7

7.1

In [1]:

```
# Import packages
import os
import json
from pathlib import Path
import gzip
import hashlib
import shutil
import pandas as pd
import pygeohash
import s3fs
```

In [4]:

```
%pip install pyarrow
```

```
Collecting pyarrowNote: you may need to restart the kernel to use updated packages.
  Downloading pyarrow-6.0.1-cp37-cp37m-win_amd64.whl (15.4 MB)
Requirement already satisfied: numpy>=1.16.6 in c:\users\nick\anaconda3\lib\site-packages
(from pyarrow) (1.19.5)
Installing collected packages: pyarrow
Successfully installed pyarrow-6.0.1
```

In [11]:

```
# set directories as usual
current_dir = Path(os.getcwd()).absolute()
results_dir = current_dir.joinpath('results')

df = pd.read_parquet('routes.parquet')

def read_jsonl_data():
    records = [row for index, row in df.iterrows()]
    return records
```

In [12]:

```
# flatten each record
def flatten_record(record):
    flat_record = dict()
    for key, value in record.items():
        if key in ['airline', 'src_airport', 'dst_airport']:
            if isinstance(value, dict):
                for child_key, child_value in value.items():
                    flat_key = '{}_{}'.format(key, child_key)
                    flat_record[flat_key] = child_value
        else:
            flat_record[key] = value

    return flat_record

# flatten the dataset
def create_flattened_dataset():
    records = read_jsonl_data()
    parquet_path = results_dir.joinpath('routes-flattened.parquet')
    return pd.DataFrame.from_records([flatten_record(record) for record in records])

# create the dataframe
df = create_flattened_dataset()
df['key'] = df['src_airport_iata'].astype(str) + df['dst_airport_iata'].astype(str) + df
['airline_iata'].astype(str)
```

a

In [13]:

```
# Make Partitions
partitions = (
        ('A', 'A'), ('B', 'B'), ('C', 'D'), ('E', 'F'),
        ('G', 'H'), ('I', 'J'), ('K', 'L'), ('M', 'M'),
        ('N', 'N'), ('O', 'P'), ('Q', 'R'), ('S', 'T'),
        ('U', 'U'), ('V', 'V'), ('W', 'X'), ('Y', 'Z')
    )
```

In [14]:

```
partition_dict = {}
for key in partitions:
    if key[0] == key[1]:
        kv_key = key[0]
    else:
        kv_key = key[0] + '-' + key[1]
    partition_dict[key] = kv_key

partition_dict
```

Out[14]:

```
{('A', 'A'): 'A',
 ('B', 'B'): 'B',
 ('C', 'D'): 'C-D',
 ('E', 'F'): 'E-F',
 ('G', 'H'): 'G-H',
 ('I', 'J'): 'I-J',
 ('K', 'L'): 'K-L',
 ('M', 'M'): 'M',
 ('N', 'N'): 'N',
 ('O', 'P'): 'O-P',
 ('Q', 'R'): 'Q-R',
 ('S', 'T'): 'S-T',
 ('U', 'U'): 'U',
 ('V', 'V'): 'V',
 ('W', 'X'): 'W-X',
 ('Y', 'Z'): 'Y-Z'}
```

In [15]:

```
def get_key(s_key):
    for key, value in partition_dict.items():
        if s_key[0] == key[0] or s_key[0] == key[1]:
            return value
    return

df['kv_key'] = df['key'].apply(get_key)
```

In [16]:

```
df.to_parquet(os.getcwd() + '/results/kv.parquet', partition_cols = ['kv_key'])
```

b

In [17]:

```
import hashlib
```

In [18]:

```
# make hash of the input
def hash_key(key):
    m = hashlib.sha256()
    m.update(str(key).encode('utf-8'))
    return m.hexdigest()
```

In [19]:

```
# create hashed and hash_key column.
df['key'] = df['src_airport_iata'].astype(str) + df['dst_airport_iata'].astype(str) + df
['airline_iata'].astype(str)
df['hashed'] = df.apply(lambda x: hash_key(x.key), axis=1)
df['hash_key'] = df['hashed'].str[:1]
```

In [21]:

```
df.to_parquet(os.getcwd() + '/results/hash.parquet', partition_cols = ['hash_key'])
```

c.

West

The Dalles, Oregon

Latitude: 45.5945645

Longitude: -121.1786823

Central

Papillion, NE

Latitude: 41.1544433

Longitude: -96.0422378

East

Loudoun County, Virginia

Latitude: 39.08344

Longitude: -77.6497145

In [23]:

```
# Import more libraries needed for this section
import pandas as pd
import numpy as np
import sklearn.neighbors
from geolib import geohash
```

In [24]:

```
df['src_airport_geohash'] = df.apply(lambda row: pygeohash.encode(row.src_airport_latitu
de, row.src_airport_longitude), axis=1)

def determine_location(src_airport_geohash):
    locations = dict(
        central = pygeohash.encode(41.1544433, -96.0422378),
        east = pygeohash.encode(39.08344, -77.6497145),
        west = pygeohash.encode(45.5945645, -121.1786823)
    )

    distances = []
    for location, geohash in locations.items():
        hav = pygeohash.geohash_haversine_distance(src_airport_geohash, geohash)
        distances.append(tuple((hav, location)))


    distances.sort()
    return distances[0][1]
df['location'] = df['src_airport_geohash'].apply(determine_location)
```

In [25]:

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

d

In [27]:

```
def balance_partitions (keys, num_partitions):
    vals = sorted(set(keys))
    num_vals = len(vals)
    partition_counts = (num_vals / num_partitions)+1
    partitions  = []
    x = 1
    y = 1
    for i in range(num_vals):
        key_val ={}
        if x <= partition_counts:
            key_val[vals[i]] = y
            x = x + 1
        else:
            x = 1
            y = y + 1
            key_val[vals[i]] = y
            x = x + 1
        partitions.append(key_val)
    return partitions

# list of keys
keys = ['alpha', 'bravo', 'charlie', 'delta', 'echo',
        'foxtrot', 'golf', 'hotel', 'india', 'juliet',
        'kilo','lima','mike','november']

# number of partitions
num_partitions = 3

# take a look at the partitions
partitions  = balance_partitions(keys, num_partitions)
print(partitions)
```

```
[{'alpha': 1}, {'bravo': 1}, {'charlie': 1}, {'delta': 1}, {'echo': 1}, {'foxtrot': 2}, {
'golf': 2}, {'hotel': 2}, {'india': 2}, {'juliet': 2}, {'kilo': 3}, {'lima': 3}, {'mike':
3}, {'november': 3}]
```

In [28]:

```
# set the number of paritions
num_partitions = 2

# create partitions and then print
partitions = balance_partitions(keys, num_partitions)
print(partitions)
```

```
[{'alpha': 1}, {'bravo': 1}, {'charlie': 1}, {'delta': 1}, {'echo': 1}, {'foxtrot': 1}, {
'golf': 1}, {'hotel': 1}, {'india': 2}, {'juliet': 2}, {'kilo': 2}, {'lima': 2}, {'mike':
2}, {'november': 2}]
```

In [ ]:

```

```