Assignment 7.1

a.

In [3]:

```
import os
import json
from pathlib import Path
import gzip
import hashlib
import shutil
import pandas as pd
import pygeohash
import s3fs
endpoint_url='https://storage.budsc.midwest-datascience.com'
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)
def read_jsonl_data():
    s3 = s3fs.S3FileSystem(
        anon=True,
        client_kwargs={
            'endpoint_url': endpoint_url
        }
    )
    src_data_path = 'data/processed/openflights/routes.jsonl.gz'
    with s3.open(src_data_path, 'rb') as f_gz:
        with gzip.open(f_gz, 'rb') as f:
            records = [json.loads(line) for line in f.readlines()]
    return records
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
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 recor
df = create_flattened_dataset()
df['key'] = df['src_airport_iata'].astype(str) + df['dst_airport_iata'].astype(s
```

In [14]:

```
df.head()
```

Out[14]:

	airline_airline_id	airline_name	airline_alias	airline_iata	airline_icao	airline_callsign	airline_cour
0	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu

	airline_airline_id	airline_name	airline_alias	airline_iata	airline_icao	airline_callsign	airline_cour
1	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
2	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
3	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
4	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu

5 rows × 39 columns

In [33]:
```
# df['kv_key'] = df['key'].str[0]
```

In [111…
```
# df.head()
```

In [36]:
```
# df.to_parquet("./results/kv/",partition_cols=['kv_key'])
```

In [38]:
```
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 [109…
```
def def_part(z):
    for x in partitions:
        if z in x:
            if x[0] == x[1]:
                return(x[0])
            else:
                return(x[0]+"-"+x[1])
```

In [110…
```
print(def_part('K'))
```

```
K-L
```

In [114…
```
df = df.drop(['kv_key'], axis = 1)
```

In [118…
```
df['kv_key'] = df['key'].str[0].apply(def_part)
```

In [119…
```
df.head()
```

Out[119…

	airline_airline_id	airline_name	airline_alias	airline_iata	airline_icao	airline_callsign	airline_cour
0	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu

	airline_airline_id	airline_name	airline_alias	airline_iata	airline_icao	airline_callsign	airline_cour
1	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
2	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
3	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
4	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu

5 rows × 40 columns

In [120…

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

b.

In [121…

```
import hashlib

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

In [126…

```
df['hashed'] = df['key'].apply(hash_key)
```

In [131…

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

In [132…

```
df.head()
```

Out[132…

	airline_airline_id	airline_name	airline_alias	airline_iata	airline_icao	airline_callsign	airline_cour
0	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
1	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
2	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
3	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
4	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu

5 rows × 42 columns

In [133…

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

C.

In [134…

```
df['src_airport_geohash'] = df.apply(
    lambda row: pygeohash.encode(row.src_airport_latitude, row.src_airport_longi
)
```

In [135…

```
df.head()
```

Out[135…

	airline_airline_id	airline_name	airline_alias	airline_iata	airline_icao	airline_callsign	airline_cour
0	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
1	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
2	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
3	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
4	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu

5 rows × 43 columns

In [145…

```
# df.count()
```

In [137…

```
df.dropna(subset=['src_airport_iata', 'dst_airport_iata'], inplace=True)
```

In [146…

```
#df.count()
```

In [147…

```
#  locations = dict(
#         central=pygeohash.encode(41.1544433, -96.0422378),
#         ## TODO: add west and east
#         west=pygeohash.encode(45.5945645, -121.1786823),
#         east=pygeohash.encode(39.08344,  -77.6497145)
#    )
```

In [252…

```
# print(locations)
```

In [253…

```
# distance = list()
# for key, value in locations.items():
#     distance.append((pygeohash.geohash_haversine_distance(value,'szsrjjzd02b3'
# #     .add(key,value)

# print(distance)
```

In [254…

```
# distance.sort()
# print(distance)
```

In [255…

```
# distance = (
#     ('central' , pygeohash.geohash_haversine_distance('9z7dnebnj8kb','szsrjjzd
#     ('west' , pygeohash.geohash_haversine_distance('c21g6s0rs4c7','szsrjjzd02b
# )
# type(distance)
# # distance[0][1]
```

In [245…

```
def determine_location(src_airport_geohash):
    locations = dict(
        central=pygeohash.encode(41.1544433, -96.0422378),
        ## TODO: add west and east
        west=pygeohash.encode(45.5945645, -121.1786823),
        east=pygeohash.encode(39.08344,  -77.6497145)
      )
    #TODO: a list of centers and distances using the pygeohash.geohash_haversine
    distances = list()
    for key,value in locations.items():
        distances.append((pygeohash.geohash_haversine_distance(value,src_airport

    distances.sort()
    return distances[0][1]
#     return distances
```

In [246…

```
determine_location('szsrjjzd02b3')
```

Out[246…

```
'east'
```

In [247…

```
df['location'] = df['src_airport_geohash'].apply(determine_location)
```

In [248…

```
df.head()
```

Out[248…

	airline_airline_id	airline_name	airline_alias	airline_iata	airline_icao	airline_callsign	airline_cour
0	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
1	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
2	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
3	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu
4	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portu

5 rows × 44 columns

In [251…

```
df['location'].value_counts()
```

Out[251…

```
east       39107
west       23033
central     4631
Name: location, dtype: int64
```

In [249…

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

d.

In [1]:

```
import numpy as np
lowv = 1
highv = 50
size = 25
num_partitions = 7
keys = np.random.randint(low = lowv, high =highv, size =size)
keys.sort()
print(keys)
# num_in_bkt = math.ceil((len(keys) - (len(keys)%num_partitions))/num_partitions
num_in_bkt = len(keys)%num_partitions


def create_bins(lower_bound,num_partitions,width,size):
  bins = []
  low =lower_bound
  high = width
  while num_partitions > 0 and high < size:
    bins.append((keys[low], keys[high]))
    low = high+1
    high =low + width
    num_partitions = num_partitions -1
  return bins


# create_bins(lowv,num_partitions,num_in_bkt,size)



def find_bin(value, bins):
    for i in range(0, len(bins)):
        if bins[i][0] <= value < bins[i][1]:
            return i
    return -1
```

```
[ 1  3  6  7  7  9 13 19 21 23 23 25 26 27 29 30 30 31 35 38 42 43 46 48
 49]
```

In [3]:

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

In [4]:

```
from collections import Counter
binned_keys= []
bins = create_bins(lowv,num_partitions,num_in_bkt,size)
print(bins)
for value in keys:
    bin_index = find_bin(value, bins)
    print(value, bin_index, bins[bin_index])
    binned_keys.append(bin_index)
```

```
frequencies = Counter(binned_keys)
print(frequencies)
```

```
[(3, 7), (9, 23), (23, 29), (30, 38), (42, 49)]
1 -1 (42, 49)
3 0 (3, 7)
6 0 (3, 7)
7 -1 (42, 49)
7 -1 (42, 49)
9 1 (9, 23)
13 1 (9, 23)
19 1 (9, 23)
21 1 (9, 23)
23 2 (23, 29)
23 2 (23, 29)
25 2 (23, 29)
26 2 (23, 29)
27 2 (23, 29)
29 -1 (42, 49)
30 3 (30, 38)
30 3 (30, 38)
31 3 (30, 38)
35 3 (30, 38)
38 -1 (42, 49)
42 4 (42, 49)
43 4 (42, 49)
46 4 (42, 49)
48 4 (42, 49)
49 -1 (42, 49)
Counter({-1: 6, 2: 5, 1: 4, 3: 4, 4: 4, 0: 2})
```

```
In [ ]:
```