Assignment 7.1 a

In [3]:
```
import pandas as pd

# read data from parquet file
df = pd.read_parquet('C:/Users/chris/DSC650-T301/DSC650-T301/Assignment03/Results/rout
df['airline_name'] = df['airline'].apply(lambda x: x['name'])
df['airline_id'] = df['airline'].apply(lambda x: x['airline_id'])
df['airline_name_iata'] = df['airline'].apply(lambda x: x['iata'])
df['src_airport'] = df['src_airport'].fillna({})
df['src_airport_name'] = df['src_airport'].apply(lambda x: x.get('name') if bool(x) ar
df['src_airport_city'] = df['src_airport'].apply(lambda x: x.get('city')if bool(x) and
df['src_airport_iata'] = df['src_airport'].apply(lambda x: x.get('iata') if bool(x) ar
df['dest_airport_name'] = df['dst_airport'].apply(lambda x: x.get('name') if bool(x) a
df['dest_airport_city'] = df['dst_airport'].apply(lambda x: x.get('city')if bool(x) ar
df['dest_airport_iata'] = df['dst_airport'].apply(lambda x: x.get('iata')if bool(x) ar
df['src_airport_latitude'] = df['src_airport'].apply(lambda x: x.get('latitude') if bo
df['src_airport_longitude'] = df['src_airport'].apply(lambda x: x.get('longitude') if
df['key'] = df['src_airport_iata'] +  df['dest_airport_iata'] + df['airline_name_iata'
df = df.drop(['airline', 'src_airport', 'dst_airport','codeshare','stops','equipment']
df.head()
```

Out[3]:

	airline_name	airline_id	airline_name_iata	src_airport_name	src_airport_city	src_airport_iata	dest_a
0	Aerocondor	410	2B	Sochi International Airport	Sochi	AER	Kazan
1	Aerocondor	410	2B	Astrakhan Airport	Astrakhan	ASF	Kazan
2	Aerocondor	410	2B	Astrakhan Airport	Astrakhan	ASF	Mine
3	Aerocondor	410	2B	Chelyabinsk Balandino Airport	Chelyabinsk	CEK	Kazan
4	Aerocondor	410	2B	Chelyabinsk Balandino Airport	Chelyabinsk	CEK	Tolma

In [3]:
```
import os

# Define the 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')
)

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

```
parts_k_v = dict(zip(partitions_keys, partitions))
print(parts_k_v)
```

```
{'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 [4]:
```
df['partition_value'] = df['key'].str[:1]
def get_key(val):
    for key, value in parts_k_v.items():
        if val in value:
            return key
    return "0"
df['kv_key'] = df.apply(lambda x: get_key(x.partition_value), axis=1)
```

In [5]:
```
# Save partitioned dataset to kv directory
df.to_parquet('results/kv', partition_cols=['kv_key'])
```

In [6]:
```
df.head()
```

Out[6]:

	airline_name	airline_id	airline_name_iata	src_airport_name	src_airport_city	src_airport_iata	dest_a
0	Aerocondor	410	2B	Sochi International Airport	Sochi	AER	Kazan
1	Aerocondor	410	2B	Astrakhan Airport	Astrakhan	ASF	Kazan
2	Aerocondor	410	2B	Astrakhan Airport	Astrakhan	ASF	Mine
3	Aerocondor	410	2B	Chelyabinsk Balandino Airport	Chelyabinsk	CEK	Kazan
4	Aerocondor	410	2B	Chelyabinsk Balandino Airport	Chelyabinsk	CEK	Tolma

Assignment 7.1 b

In [7]:
```
import hashlib
import pyarrow as pa
import pyarrow.parquet as pq
import pandas as pd

# Define the hash function
def hash_key(key):
    m = hashlib.sha256()
    m.update(str(key).encode('utf-8'))
    return m.hexdigest()


# Create the key column
df['key'] = df['src_airport_iata'] + df['dest_airport_iata'] + df['airline_name_iata']

# Create the hashed column
```

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

# Create the hash partition column
df['hash_key'] = df['hashed'].str[:1]

# Convert the data frame to a PyArrow table
pa_table = pa.Table.from_pandas(df)


# Write the PyArrow table to a Parquet dataset partitioned by hash_key
pq.write_to_dataset(pa_table, root_path='results/hash', partition_cols=['hash_key'])
```

In [8]:
```
df.head()
```

Out[8]:

	airline_name	airline_id	airline_name_iata	src_airport_name	src_airport_city	src_airport_iata	dest_a
0	Aerocondor	410	2B	Sochi International Airport	Sochi	AER	Kazan
1	Aerocondor	410	2B	Astrakhan Airport	Astrakhan	ASF	Kazan
2	Aerocondor	410	2B	Astrakhan Airport	Astrakhan	ASF	Mine
3	Aerocondor	410	2B	Chelyabinsk Balandino Airport	Chelyabinsk	CEK	Kazan
4	Aerocondor	410	2B	Chelyabinsk Balandino Airport	Chelyabinsk	CEK	Tolma

Assignment 7.1 c

In [9]:
```
from math import radians, cos, sin, asin, sqrt
import pyarrow as pa
import pyarrow.parquet as pq

def haversine(lat1, lon1, lat2, lon2):
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])

    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    r = 6371 # Radius of earth in kilometers. Use 3956 for miles
    return c * r

# Define the locations of the data centers
data_centers = {
    'west': (45.5945645, -121.1786823),
    'central': (41.1544433, -96.0422378),
    'east': (39.08344, -77.6497145)
}

# Calculate the distance between each source airport and the data centers
```

```
for index, row in df.iterrows():
    lat1, lon1 = row['src_airport_latitude'], row['src_airport_longitude']
    distances = {}
    for location, coords in data_centers.items():
        lat2, lon2 = coords
        distance = haversine(lat1, lon1, lat2, lon2)
        distances[location] = distance
    closest_location = min(distances, key=distances.get)
    folder_name = f"location={closest_location}"

    # Select the rows corresponding to the current route
    route_df = df.loc[df['key'] == row['key']]

    # Convert the data frame to a PyArrow table
    route_table = pa.Table.from_pandas(route_df)

    # Write the route to the appropriate folder
    pq.write_table(route_table, "results/geo")
```

Assignment 7.1 d

In [10]:
```
df.head()
```

Out[10]:

	airline_name	airline_id	airline_name_iata	src_airport_name	src_airport_city	src_airport_iata	dest_a
0	Aerocondor	410	2B	Sochi International Airport	Sochi	AER	Kazan
1	Aerocondor	410	2B	Astrakhan Airport	Astrakhan	ASF	Kazan
2	Aerocondor	410	2B	Astrakhan Airport	Astrakhan	ASF	Mine
3	Aerocondor	410	2B	Chelyabinsk Balandino Airport	Chelyabinsk	CEK	Kazan
4	Aerocondor	410	2B	Chelyabinsk Balandino Airport	Chelyabinsk	CEK	Tolma

In [4]:
```
def balance_partitions (keys, num_partitions):
    uniquevals = sorted(set(keys))
    num_uniq_vals = len(uniquevals)
    partition_counts = (num_uniq_vals / num_partitions)+1
    partitions  = []
    currow = 1
    partnum = 1
    for i in range(num_uniq_vals):
        curkeyval ={}
        if currow <= partition_counts:
            curkeyval[uniquevals[i]] = partnum
            currow = currow + 1
        else:
            currow = 1
            partnum = partnum + 1
            curkeyval[uniquevals[i]] = partnum
            currow = currow + 1
```

```
        partitions .append(curkeyval)
    return partitions


keys = df['airline_id']
num_partitions = 3
partitions  = balance_partitions(keys, num_partitions)
print(type(partitions ))
print(partitions )
```

```
<class 'list'>
[{-1: 1}, {10: 1}, {21: 1}, {24: 1}, {28: 1}, {29: 1}, {32: 1}, {35: 1}, {42: 1}, {4
3: 1}, {55: 1}, {68: 1}, {83: 1}, {90: 1}, {96: 1}, {106: 1}, {109: 1}, {116: 1}, {12
5: 1}, {130: 1}, {132: 1}, {137: 1}, {139: 1}, {146: 1}, {153: 1}, {179: 1}, {197:
1}, {214: 1}, {218: 1}, {220: 1}, {221: 1}, {225: 1}, {231: 1}, {240: 1}, {241: 1},
{242: 1}, {246: 1}, {312: 1}, {316: 1}, {319: 1}, {321: 1}, {324: 1}, {328: 1}, {329:
1}, {330: 1}, {333: 1}, {336: 1}, {338: 1}, {341: 1}, {345: 1}, {386: 1}, {397: 1},
{410: 1}, {412: 1}, {426: 1}, {439: 1}, {442: 1}, {462: 1}, {470: 1}, {476: 1}, {477:
1}, {491: 1}, {502: 1}, {503: 1}, {508: 1}, {515: 1}, {524: 1}, {543: 1}, {563: 1},
{567: 1}, {569: 1}, {576: 1}, {595: 1}, {596: 1}, {603: 1}, {608: 1}, {622: 1}, {641:
1}, {683: 1}, {690: 1}, {692: 1}, {751: 1}, {753: 1}, {794: 1}, {807: 1}, {837: 1},
{879: 1}, {881: 1}, {882: 1}, {896: 1}, {897: 1}, {921: 1}, {970: 1}, {995: 1}, {998:
1}, {1006: 1}, {1008: 1}, {1034: 1}, {1048: 1}, {1057: 1}, {1066: 1}, {1073: 1}, {110
9: 1}, {1173: 1}, {1191: 1}, {1203: 1}, {1206: 1}, {1230: 1}, {1266: 1}, {1287: 1},
{1290: 1}, {1299: 1}, {1308: 1}, {1316: 1}, {1317: 1}, {1338: 1}, {1340: 1}, {1355:
1}, {1359: 1}, {1392: 1}, {1401: 1}, {1403: 1}, {1422: 1}, {1434: 1}, {1441: 1}, {146
3: 1}, {1469: 1}, {1472: 1}, {1478: 1}, {1492: 1}, {1500: 1}, {1508: 1}, {1531: 1},
{1539: 1}, {1548: 1}, {1581: 1}, {1611: 1}, {1623: 1}, {1629: 1}, {1654: 1}, {1663:
1}, {1669: 1}, {1680: 1}, {1682: 1}, {1683: 1}, {1729: 1}, {1750: 1}, {1756: 1}, {175
8: 1}, {1767: 1}, {1769: 1}, {1775: 1}, {1790: 1}, {1792: 1}, {1829: 1}, {1844: 1},
{1868: 1}, {1886: 1}, {1889: 1}, {1908: 1}, {1909: 1}, {1925: 1}, {1936: 1}, {1942:
1}, {1943: 1}, {1946: 1}, {1954: 1}, {1966: 1}, {2009: 1}, {2056: 1}, {2058: 1}, {209
1: 1}, {2094: 1}, {2104: 1}, {2117: 1}, {2143: 1}, {2150: 1}, {2183: 1}, {2193: 1},
{2217: 1}, {2218: 1}, {2220: 1}, {2222: 1}, {2226: 2}, {2245: 2}, {2260: 2}, {2264:
2}, {2293: 2}, {2297: 2}, {2324: 2}, {2350: 2}, {2353: 2}, {2354: 2}, {2395: 2}, {240
9: 2}, {2417: 2}, {2418: 2}, {2419: 2}, {2420: 2}, {2421: 2}, {2439: 2}, {2468: 2},
{2520: 2}, {2524: 2}, {2538: 2}, {2541: 2}, {2547: 2}, {2548: 2}, {2575: 2}, {2585:
2}, {2607: 2}, {2622: 2}, {2638: 2}, {2660: 2}, {2681: 2}, {2682: 2}, {2684: 2}, {268
8: 2}, {2692: 2}, {2731: 2}, {2748: 2}, {2750: 2}, {2757: 2}, {2765: 2}, {2773: 2},
{2774: 2}, {2822: 2}, {2825: 2}, {2826: 2}, {2835: 2}, {2850: 2}, {2857: 2}, {2881:
2}, {2896: 2}, {2916: 2}, {2922: 2}, {2923: 2}, {2942: 2}, {2951: 2}, {2954: 2}, {298
7: 2}, {2989: 2}, {2990: 2}, {2993: 2}, {2994: 2}, {3000: 2}, {3021: 2}, {3026: 2},
{3029: 2}, {3052: 2}, {3081: 2}, {3090: 2}, {3097: 2}, {3123: 2}, {3126: 2}, {3148:
2}, {3163: 2}, {3179: 2}, {3197: 2}, {3200: 2}, {3201: 2}, {3210: 2}, {3233: 2}, {325
1: 2}, {3258: 2}, {3287: 2}, {3290: 2}, {3320: 2}, {3329: 2}, {3342: 2}, {3354: 2},
{3370: 2}, {3378: 2}, {3386: 2}, {3391: 2}, {3392: 2}, {3393: 2}, {3432: 2}, {3437:
2}, {3463: 2}, {3490: 2}, {3498: 2}, {3534: 2}, {3539: 2}, {3545: 2}, {3547: 2}, {357
4: 2}, {3589: 2}, {3613: 2}, {3618: 2}, {3637: 2}, {3652: 2}, {3661: 2}, {3674: 2},
{3721: 2}, {3734: 2}, {3737: 2}, {3740: 2}, {3754: 2}, {3764: 2}, {3776: 2}, {3778:
2}, {3781: 2}, {3783: 2}, {3788: 2}, {3805: 2}, {3811: 2}, {3826: 2}, {3834: 2}, {383
5: 2}, {3850: 2}, {3856: 2}, {3857: 2}, {3865: 2}, {3871: 2}, {3926: 2}, {3935: 2},
{3952: 2}, {3969: 2}, {3976: 2}, {4021: 2}, {4026: 2}, {4031: 2}, {4044: 2}, {4066:
2}, {4089: 2}, {4091: 2}, {4165: 2}, {4178: 2}, {4234: 2}, {4248: 2}, {4255: 2}, {425
9: 2}, {4292: 2}, {4296: 2}, {4304: 2}, {4305: 2}, {4311: 2}, {4319: 2}, {4329: 2},
{4335: 2}, {4349: 2}, {4356: 2}, {4375: 2}, {4388: 2}, {4429: 2}, {4435: 2}, {4436:
2}, {4438: 2}, {4454: 2}, {4475: 2}, {4496: 2}, {4513: 2}, {4521: 2}, {4533: 2}, {454
7: 2}, {4550: 2}, {4559: 2}, {4573: 2}, {4599: 2}, {4608: 2}, {4609: 2}, {4611: 2},
{4687: 2}, {4691: 2}, {4735: 2}, {4737: 3}, {4740: 3}, {4750: 3}, {4752: 3}, {4797:
3}, {4805: 3}, {4808: 3}, {4822: 3}, {4840: 3}, {4863: 3}, {4867: 3}, {4869: 3}, {487
0: 3}, {4897: 3}, {4936: 3}, {4937: 3}, {4940: 3}, {4947: 3}, {4951: 3}, {4965: 3},
{5002: 3}, {5013: 3}, {5016: 3}, {5038: 3}, {5039: 3}, {5041: 3}, {5067: 3}, {5083:
3}, {5085: 3}, {5097: 3}, {5133: 3}, {5156: 3}, {5179: 3}, {5188: 3}, {5209: 3}, {523
4: 3}, {5265: 3}, {5281: 3}, {5282: 3}, {5297: 3}, {5309: 3}, {5325: 3}, {5331: 3},
{5333: 3}, {5347: 3}, {5354: 3}, {5360: 3}, {5368: 3}, {5399: 3}, {5416: 3}, {5439:
3}, {5461: 3}, {5479: 3}, {5484: 3}, {5496: 3}, {5521: 3}, {5523: 3}, {5651: 3}, {581
3: 3}, {5982: 3}, {6557: 3}, {8359: 3}, {8463: 3}, {8576: 3}, {8745: 3}, {8809: 3},
{9082: 3}, {9531: 3}, {9541: 3}, {9620: 3}, {9666: 3}, {9764: 3}, {9784: 3}, {9809:
3}, {9810: 3}, {9818: 3}, {9828: 3}, {9829: 3}, {10121: 3}, {10122: 3}, {10128: 3},
{10646: 3}, {10650: 3}, {10675: 3}, {10737: 3}, {10739: 3}, {10741: 3}, {10758: 3},
{10765: 3}, {10776: 3}, {10800: 3}, {10912: 3}, {10955: 3}, {11741: 3}, {11763: 3},
```

```
{11794: 3}, {11806: 3}, {11808: 3}, {11811: 3}, {11814: 3}, {11838: 3}, {11857: 3},
{11948: 3}, {11963: 3}, {12978: 3}, {13088: 3}, {13108: 3}, {13200: 3}, {13335: 3},
{13704: 3}, {13757: 3}, {13899: 3}, {13983: 3}, {14061: 3}, {14118: 3}, {14485: 3},
{14849: 3}, {15814: 3}, {15837: 3}, {15893: 3}, {15999: 3}, {16120: 3}, {16133: 3},
{16136: 3}, {16149: 3}, {16150: 3}, {16262: 3}, {16415: 3}, {16475: 3}, {16508: 3},
{16615: 3}, {16624: 3}, {16625: 3}, {16660: 3}, {16707: 3}, {16725: 3}, {16726: 3},
{16844: 3}, {16882: 3}, {16942: 3}, {16960: 3}, {16963: 3}, {17023: 3}, {17083: 3},
{17094: 3}, {17095: 3}, {17099: 3}, {17408: 3}, {17519: 3}, {17675: 3}, {17885: 3},
{17891: 3}, {18169: 3}, {18232: 3}, {18529: 3}, {18543: 3}, {18553: 3}, {18700: 3},
{18732: 3}, {18825: 3}, {18828: 3}, {18944: 3}, {18946: 3}, {19016: 3}, {19305: 3},
{19582: 3}, {19610: 3}, {19676: 3}, {19804: 3}, {19810: 3}, {19944: 3}, {20004: 3},
{20047: 3}, {20160: 3}, {20270: 3}, {20565: 3}, {20577: 3}, {20686: 3}, {20710: 3},
{20963: 3}, {20976: 3}, {21012: 3}]
```

In [ ]: