Assignment 7

April 30, 2023

[1]:
```
import os
from pathlib import Path
import pandas as pd
import geopy.distance
import math

# Setting dataframe path
df_path = os.path.abspath('../../assignments/assignment03/results/routes.
 ↪parquet')
current_dir = Path(os.getcwd()).absolute()
```

1 7.a

[2]:
```
# Reading Parquet dataframe
df = pd.read_parquet(df_path)
df.head(5)
```

[2]:
```
                                              airline  \
0  {'active': True, 'airline_id': 410, 'alias': '…
1  {'active': True, 'airline_id': 410, 'alias': '…
2  {'active': True, 'airline_id': 410, 'alias': '…
3  {'active': True, 'airline_id': 410, 'alias': '…
4  {'active': True, 'airline_id': 410, 'alias': '…

                                         src_airport  \
0  {'airport_id': 2965.0, 'altitude': 89.0, 'city…
1  {'airport_id': 2966.0, 'altitude': -65.0, 'cit…
2  {'airport_id': 2966.0, 'altitude': -65.0, 'cit…
3  {'airport_id': 2968.0, 'altitude': 769.0, 'cit…
4  {'airport_id': 2968.0, 'altitude': 769.0, 'cit…

                                         dst_airport  codeshare equipment
0  {'airport_id': 2990.0, 'altitude': 411.0, 'cit…      False     [CR2]
1  {'airport_id': 2990.0, 'altitude': 411.0, 'cit…      False     [CR2]
2  {'airport_id': 2962.0, 'altitude': 1054.0, 'ci…      False     [CR2]
3  {'airport_id': 2990.0, 'altitude': 411.0, 'cit…      False     [CR2]
```

```
4  {'airport_id': 4078.0, 'altitude': 365.0, 'cit…      False     [CR2]
```

[3]:
```
# Reading each column to read in a formatted table
airline_df = pd.json_normalize(df['airline'])
airline_df = airline_df.add_prefix('airline.') # Adding prefix to avoid␣
 ↪confusing with same column names

src_df = pd.json_normalize(df['src_airport'])
src_df = src_df.add_prefix('src.')

dst_df = pd.json_normalize(df['dst_airport'])
dst_df = dst_df.add_prefix('dst.')

# Merging dataframes
df = pd.concat([airline_df, src_df, dst_df, df], axis=1)

# Dropping unnecessary columns
df = df.drop(columns=['airline', 'src_airport', 'dst_airport'])

df.sample(5)
```

[3]:
```
       airline.active  airline.airline_id         airline.alias  \
5707             True                  24                    \N
57683            True                5209                   TWA
19855            True                1868  SN Brussels Airlines
12972            True                 596              Air Asia
20160            True                2009    CSA Czech Airlines

      airline.callsign airline.country airline.iata airline.icao  \
5707          AMERICAN   United States           AA          AAL
57683           UNITED   United States           UA          UAL
19855           CONDOR         Germany           DE          CFG
12972         ALITALIA           Italy           AZ          AZA
20160            DELTA   United States           DL          DAL

            airline.name  src.airport_id  src.altitude  … dst.icao  \
5707   American Airlines          3484.0         125.0  …     SBGR
57683    United Airlines          3469.0          13.0  …     KPIT
19855  Condor Flugdienst           478.0         257.0  …     GCTS
12972           Alitalia          3682.0        1026.0  …     KBOS
20160    Delta Air Lines          3682.0        1026.0  …     MKJP

      dst.latitude dst.longitude  \
5707    -23.435556    -46.473057
57683    40.491501    -80.232903
19855    28.044500    -16.572500
12972    42.364300    -71.005203
```

```
20160    17.935699    -76.787498

                                                dst.name   dst.source  \
5707   Guarulhos - Governador André Franco Montoro In…  OurAirports
57683                    Pittsburgh International Airport  OurAirports
19855                              Tenerife South Airport  OurAirports
12972  General Edward Lawrence Logan International Ai…  OurAirports
20160                 Norman Manley International Airport  OurAirports

       dst.timezone  dst.type          dst.tz_id codeshare        equipment
5707           -3.0   airport  America/Sao_Paulo     False            [777]
57683          -5.0   airport   America/New_York     False       [319, 738]
19855           0.0   airport    Atlantic/Canary     False            [757]
12972          -5.0   airport   America/New_York      True  [M90, 757, 319]
20160          -5.0   airport    America/Jamaica     False            [319]

[5 rows x 38 columns]
```

[4]:
```
# Adding key to the dataframe
df['key'] = df.apply(lambda row: str(row['src.iata']) + str(row['dst.iata']) +␣
 ↪str(row['airline.iata']), axis=1)
df[['src.iata','dst.iata','airline.iata','key']].sample(5)
```

[4]:
```
      src.iata dst.iata airline.iata       key
12738      ICN      HEL           AY  ICNHELAY
34662      KIX      HAN           JL  KIXHANJL
40363      TAO      SHE           MF  TAOSHEMF
47044      PER      ADL           QF  PERADLQF
233        GYN      CGB           2Z  GYNCGB2Z
```

[5]:
```
# Defining partition
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')
    )

# Creating partition
def get_partition(key):
    key_letter = str(key[0])

    for first,last in partitions:
        if str(first) <= key_letter <= str(last):
            if first == last:
                return str(first)
            else:
```

```
            return str(''.join((first,'-',last)))

# Creating a new column to save partition
df['kv_key'] = df['key'].apply(get_partition)
df[['key', 'kv_key']].sample(5)
```

[5]:

	key	kv_key
66466	GIBLTNZB	G-H
13949	BOSSAVB6	B
19440	XNNKWECZ	W-X
58891	CMHPHXUS	C-D
13970	CLTJFKB6	C-D

[39]:
```
# Saving directory structure
df.to_parquet('results/kv/', partition_cols=['kv_key'])
```

[ ]:

2 7.b

[7]:
```
import hashlib

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

# Creating a hashed column
df['hashed'] = df['key'].apply(hash_key)
df[['key','hashed']].head(5)
```

[7]:

	key	hashed
0	AERKZN2B	652cdec02010381f175efe499e070c8cbaac1522bac59a…
1	ASFKZN2B	9eea5dd88177f8d835b2bb9cb27fb01268122b635b241a…
2	ASFMRV2B	161143856af25bd4475f62c80c19f68936a139f653c1d3…
3	CEKKZN2B	39aa99e6ae2757341bede9584473906ef1089e30820c90…
4	CEKOVB2B	143b3389bce68eea3a13ac26a9c76c1fa583ec2bd26ea8…

[8]:
```
# creating a partitioned dataset based on the first character of the hashed key
df['hash_key'] = df['hashed'].apply(lambda x: x[0].upper())
df[['key','hashed','hash_key']].sample(5)
```

[8]:

	key	hashed	hash_key
48361	MIACCSS3	8328bf93c9cf4fa080150132bad57a6666450148ca684f…	8
58125	BHKDMEUN	b92bce08c63dcfe4358c26d3610339b97cc71d163a64a8…	B
38788	TLSMUCLH	7402381ced4d6db0fd35c41df546cc03fbef421a920bd3…	7

```
55363   SENAMSU2   1c37c5adeb36c4207d55b7d1a49b0e64143a193c27d416…        1
26465   CIAWMIFR   670c307ae6306c01ef29e4230066ceac29741828f0b398…        6
```

[56]:
```
# Saving directory structure
df.to_parquet('results/hash/', partition_cols=['hash_key'])
```

[ ]:
```

```

3 7.c

[9]:
```
data_center_dict = {
    'west': (45.5945645,-121.1786823),
    'central': (41.1544433,-96.0422378),
    'east': (39.08344,-77.6497145)
}

# Function to find closest center based on Haversine formula
def closest_center(lat,lon, center_dict = data_center_dict):
    # Initializing variables
    closest_dist = 1e100
    closest_center = ''

    if isinstance(lat, (int, float)) and isinstance(lon, (int, float)) and math.
 ↪isfinite(lat) and math.isfinite(lon):
        for center, coor in center_dict.items():
            # Finding distance between two points
            src_distance = geopy.distance.geodesic((float(lat), float(lon)),␣
 ↪coor).km

            # Assiging closest center
            if src_distance < closest_dist:
                closest_dist = src_distance
                closest_center = center

        return closest_center

    else: None

df['location'] = df.apply(lambda x: closest_center(x['src.latitude'], x['src.
 ↪longitude']), axis=1)
df[['src.latitude','src.longitude','location']].sample(5)
```

[9]:
```
       src.latitude  src.longitude location
19997     12.533500      -7.949940     east
57364     41.978600     -87.904800  central
34385     25.793200     -80.290604     east
```

```
48824    -24.368601      31.048700     east
25716     31.197901     121.335999     west
```

[121]:
```
# Saving directory structure
df.to_parquet('results/geo/', partition_cols=['location'])
```

[ ]:
```

```

4 7.d

[6]:
```
def balance_partitions(keys, num_partitions):

    # Sorting the input keys
    sort_key = sorted(keys)

    # Computing the partition size
    partition_size = len(sort_key) // num_partitions

    partitions = []

    # Ensuring  each partition contains all the keys between the least key in␣
 ↪the partition and the greatest key in the partition
    for i in range(num_partitions):
        start_index = i * partition_size
        end_index = start_index + partition_size
        partitions.append(sort_key[start_index:end_index])

    # Ensuring that each partition is ordered
    partitions = [sorted(par) for par in partitions]

    return partitions

# printing first partition keys
print(balance_partitions(list(df.key), 1000)[0])
```

```
['AAEALGAH', 'AAECDGAH', 'AAEISLAH', 'AAELYSAH', 'AAEMRSAH', 'AAEMRSZI',
'AAEORNAH', 'AAEORYAH', 'AAEORYZI', 'AALAARBA', 'AALAGPDY', 'AALALCDY',
'AALAMSAZ', 'AALAMSKL', 'AALARNSK', 'AALBCNIB', 'AALBCNVY', 'AALBLLDX',
'AALBLLSK', 'AALBLLTK', 'AALCPHDY', 'AALCPHSK', 'AALISLTK', 'AALLGWDY',
'AALOSLBA', 'AALOSLM3', 'AALOSLSK', 'AALPMIDY', 'AALSVGDX', 'AANCCJIX',
'AANPEWNL', 'AAQDMES7', 'AAQLEDSU', 'AAQSVOSU', 'AARAALBA', 'AARAGPFR',
'AARBMABA', 'AARCPHSK', 'AARGOTBA', 'AAROSLBA', 'AARPMIFR', 'AARSTNFR',
'AATURCCZ', 'AATURCGS', 'AAXPOJAD', 'AAYSAHFO', 'ABADMES7', 'ABAIKTnan',
'ABANSKY7', 'ABASVOSU', 'ABDMHDB9', 'ABDMHDIR', 'ABDSYZEP', 'ABDTHRB9',
'ABDTHREP', 'ABDTHRIR', 'ABEATLAF', 'ABEATLDL', 'ABEATLKL', 'ABECLTAA',
'ABECLTUS', 'ABEDTWDL', 'ABEMYRG4', 'ABEORDUA', 'ABEPGDG4', 'ABEPHLAA',
```

```
'ABEPHLUS']
```

[ ]: