Assignment 7

October 21, 2022

[1]:
```
import os
import json
from pathlib import Path
import gzip
import hashlib
import shutil
import pandas as pd
import pygeohash
import s3fs
import uuid
import math
import itertools
```

[2]:
```
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)
```

[3]:
```
current_dir = Path(os.getcwd()).absolute()

# read file from local directory

def read_jsonl_data():

    file_path = '/home/jovyan/dsc650/data/processed/openflights/routes.jsonl.gz'
    with gzip.open(file_path,'rb') as f:
        records = [json.loads(line) for line in f.readlines()]
    return records
```

[4]:
```
# function to flatten 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
```

[5]:
```
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])
```

[6]:
```
df = create_flattened_dataset()
df['key'] = df['src_airport_iata'].astype(str) + df['dst_airport_iata'].
 ↪astype(str) + df['airline_iata'].astype(str)
```

[7]:
```
# display head and shape of df
df.shape
df.head()
```

[7]:

	airline_airline_id	airline_name	airline_alias	airline_iata	\
0	410	Aerocondor	ANA All Nippon Airways	2B	
1	410	Aerocondor	ANA All Nippon Airways	2B	
2	410	Aerocondor	ANA All Nippon Airways	2B	
3	410	Aerocondor	ANA All Nippon Airways	2B	
4	410	Aerocondor	ANA All Nippon Airways	2B	

	airline_icao	airline_callsign	airline_country	airline_active	\
0	ARD	AEROCONDOR	Portugal	True	
1	ARD	AEROCONDOR	Portugal	True	
2	ARD	AEROCONDOR	Portugal	True	
3	ARD	AEROCONDOR	Portugal	True	
4	ARD	AEROCONDOR	Portugal	True	

	src_airport_airport_id	src_airport_name	…	\
0	2965.0	Sochi International Airport	…	
1	2966.0	Astrakhan Airport	…	
2	2966.0	Astrakhan Airport	…	
3	2968.0	Chelyabinsk Balandino Airport	…	
4	2968.0	Chelyabinsk Balandino Airport	…	

	dst_airport_longitude	dst_airport_altitude	dst_airport_timezone	\
0	49.278702	411.0	3.0	
1	49.278702	411.0	3.0	
2	43.081902	1054.0	3.0	

```
3              49.278702                411.0                   3.0
4              82.650703                365.0                   7.0

  dst_airport_dst  dst_airport_tz_id  dst_airport_type  dst_airport_source  \
0               N      Europe/Moscow           airport         OurAirports
1               N      Europe/Moscow           airport         OurAirports
2               N      Europe/Moscow           airport         OurAirports
3               N      Europe/Moscow           airport         OurAirports
4               N   Asia/Krasnoyarsk           airport         OurAirports

   codeshare equipment       key
0      False     [CR2]  AERKZN2B
1      False     [CR2]  ASFKZN2B
2      False     [CR2]  ASFMRV2B
3      False     [CR2]  CEKKZN2B
4      False     [CR2]  CEKOVB2B

[5 rows x 39 columns]
```

[8]:
```
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')
)
```

[9]:
```
#function to create key_value column based on partitions

def key_value_partitions(partitions,strng):

    first_char = strng[0]
    for tup in partitions:

        if first_char in tup:
            if tup.count(tup[0])==len(tup):
                return tup[0]
            else:
                return tup[0]+''+tup[1]
    return 'None' #added None for nan values for source airport code
```

[10]:
```
# create new column kv_key based on partitions
df['kv_key'] = df['key'].apply(lambda x: key_value_partitions(partitions,x))
```

[11]:
```
# display new column
df.sample(2)
```

[11]:
```
       airline_airline_id        airline_name airline_alias airline_iata  \
7657                  214          Air Berlin            \N           AB
63590                4547  Southwest Airlines       SkyWork           WN

      airline_icao airline_callsign airline_country  airline_active  \
7657           BER       AIR BERLIN         Germany            True
63590          SWA        SOUTHWEST   United States            True

       src_airport_airport_id  \
7657                   1056.0
63590                  3849.0

                                     src_airport_name  …  \
7657                             Tenerife South Airport  …
63590  Baltimore/Washington International Thurgood Ma…  …

      dst_airport_altitude dst_airport_timezone dst_airport_dst  \
7657                1411.0                 1.0               E
63590                599.0                -6.0               A

      dst_airport_tz_id  dst_airport_type  dst_airport_source  codeshare  \
7657      Europe/Vienna           airport         OurAirports      False
63590   America/Chicago           airport         OurAirports      False

             equipment       key kv_key
7657             [320]  TFSSZGAB    ST
63590  [733, 73C, 73W]  BWIBNAWN      B

[2 rows x 40 columns]
```

[12]:
```
# create df parttitions based on kv_key
df.to_parquet('./results/kv',partition_cols=['kv_key'])
```

1 Assignment 7.1b

[13]:
```
# function to create hashvalue

import hashlib

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

[14]:
```
# new column hashed using hash value functon

```

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

[15]:
```
# create new column for partitioning based on hashed column

df['hash_key'] = df['hashed'].apply(lambda x:x[0])
```

[16]:
```
# display new column

df[['hashed','hash_key']][:5]
```

[16]:

	hashed	hash_key
0	652cdec02010381f175efe499e070c8cbaac1522bac59a…	6
1	9eea5dd88177f8d835b2bb9cb27fb01268122b635b241a…	9
2	161143856af25bd4475f62c80c19f68936a139f653c1d3…	1
3	39aa99e6ae2757341bede9584473906ef1089e30820c90…	3
4	143b3389bce68eea3a13ac26a9c76c1fa583ec2bd26ea8…	1

[17]:
```
# create df parttitions based on hash_key

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

2 Assignment 7.1.c

[18]:
```
# create new column to calculate source airportt geohash value

df['source_airport_geohash'] = df.apply(lambda x : pygeohash.
 ↪encode(x['src_airport_latitude'],x['src_airport_longitude']),axis=1)
```

[19]:
```
# display new column value to make sure geohash values are correct

df['source_airport_geohash'][:4]
```

[19]:
```
0    szsrjjzd02b3
1    v04pk3t5gbjj
2    v04pk3t5gbjj
3    v3gdxs17du83
Name: source_airport_geohash, dtype: object
```

[20]:
```
#  create a function to get closest datacenter from source airport

def get_datacenter_location(geohash):

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

```
    # calculate the distance and store center and disatnce from airport

    distance_dict = {}

    for key in data_centers.keys():
        distance_dict[key] = pygeohash.geohash_haversine_distance(data_centers.
↪get(key),geohash)
    return sorted(distance_dict.items(), key=lambda x: x[1])[0][0]
```

[21]:
```
# create new columns location to store closest data center

df['location'] = df['source_airport_geohash'].apply(lambda x:␣
↪get_datacenter_location(x))
```

[22]:
```
# display new column value to make sure location values are correct

df['location'][:4]
```

[22]:
```
0    east
1    east
2    east
3    west
Name: location, dtype: object
```

[23]:
```
# create df parttitions based on hash_key

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

3 Assignment 7.1.d

[24]:
```
#  function to create partitions


def balance_partitions(keys, num_partitions):
    part_size = round(len(keys)/num_partitions)
    iters = iter(keys)
    partitions_iters = iter(lambda: tuple(itertools.islice(iters, part_size)),␣
↪())
    partitions = [sorted(part) for part in partitions_iters]
    return partitions
```

[25]:
```
df.sample(5)
```

[25]:

	airline_airline_id	airline_name	airline_alias	\
43007	324	All Nippon Airways	ANA All Nippon Airways	
53022	8745	Transavia France	nan	
46736	4089	Qantas	Qantas Airways	
46050	12978	West Air China	nan	
44618	491	Austrian Airlines	ANA All Nippon Airways	

	airline_iata	airline_icao	airline_callsign	airline_country	\
43007	NH	ANA	ALL NIPPON	Japan	
53022	TO	TVF	FRENCH SUN	France	
46736	QF	QFA	QANTAS	Australia	
46050	PN	CHB	WEST CHINA	China	
44618	OS	AUA	AUSTRIAN	Austria	

	airline_active	src_airport_airport_id	\
43007	True	3992.0	
53022	True	469.0	
46736	True	3341.0	
46050	True	3393.0	
44618	True	348.0	

	src_airport_name	…	dst_airport_type	\
43007	Kansai International Airport	…	airport	
53022	Birmingham International Airport	…	airport	
46736	Adelaide International Airport	…	airport	
46050	Chongqing Jiangbei International Airport	…	airport	
44618	Leipzig/Halle Airport	…	airport	

	dst_airport_source	codeshare	equipment	key	kv_key	\
43007	OurAirports	True	[320]	KIXFOCNH	KL	
53022	OurAirports	False	[73H, 75W]	BHXACETO	B	
46736	OurAirports	False	[73H]	ADLASPQF	A	
46050	OurAirports	False	[320]	CKGXIYPN	CD	
44618	OurAirports	True	[DH4]	LEJVIEOS	KL	

	hashed	hash_key	\
43007	ba99d82f226121e4fe9031716c630a805ccac0ee29385e…	b	
53022	9a725852611b7acee65afe2c75bf84e35cb740e622325e…	9	
46736	9660c3f80dc7dc4a2d8ad47ac9165efcc14b05cb6be967…	9	
46050	83afea3d1085fa3d6269a2d3ba9083b285313e081b9126…	8	
44618	c2021b414df8c1d5a0b6124f8d5c44a7ee79dfab971a3a…	c	

	source_airport_geohash	location
43007	xn05ve39t9y0	west
53022	gcqf2hzn5gw3	east
46736	r1f90q7nw7ug	west
46050	wm7c4dnqfsmq	west

```
44618           u30sw3qmbe8y     east

[5 rows x 44 columns]
```

[26]:
```
# validate the function - balance_partitions

airline_names = df.airline_iata.sample(50).to_list()
partitions = balance_partitions(airline_names,5)
partitions
```

[26]:
```
[['AC', 'CX', 'CZ', 'G3', 'LO', 'SQ', 'UA', 'UA', 'UA', 'nan'],
 ['3U', '4U', 'AM', 'AZ', 'JL', 'LS', 'UA', 'US', 'XQ', 'YI'],
 ['A5', 'AH', 'FL', 'IR', 'MU', 'MU', 'PX', 'UG', 'VA', 'XK'],
 ['AA', 'CI', 'CZ', 'IB', 'MI', 'R3', 'UA', 'UE', 'UG', 'W6'],
 ['5T', 'BE', 'FR', 'GA', 'KE', 'MH', 'ST', 'US', 'W6', 'WF']]
```

[ ]: