Assignment#7

Ganesh Kale

Assignment 7.1.a

In [135…

```python
# import required packages


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

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
```

In [2]:

```python
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)
```

In [3]:

```python
# function to read and load jsonl zip file


def read_jsonl_data_s3():
    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


# read file from local directory

def read_jsonl_data():

    file_path = '/Users/ganeshkale/work/virtual_envs/DSC650_Big_Data/dsc650/data/processed/openflights/routes.j

    with gzip.open(file_path,'rb') as f:
        records = [json.loads(line) for line in f.readlines()]
        return records
```

In [4]:

```python
# 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
```

In [5]:

```python
# function to load records and create dataframe

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])
```

In [6]:

```python
# create df by calling above functions

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

In [7]:

```python
# display head and shape of df

df.shape
df.head()
```

Out[7]:

```
(67663, 39)
```

Out[7]:

	airline_airline_id	airline_name	airline_alias	airline_iata	airline_icao	airline_callsign	airline_country	airline_active	src_airport_airp
0	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portugal	True	
1	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portugal	True	
2	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portugal	True	
3	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portugal	True	
4	410	Aerocondor	ANA All Nippon Airways	2B	ARD	AEROCONDOR	Portugal	True	

5 rows × 39 columns

In [8]:

```python
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 [35]:

```python
#  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
```

In [32]:

```python
# create new column kv_key based on partitions

df['kv_key'] = df['key'].apply(lambda x: key_value_partitions(partitions,x))
```

In [33]:

```python
# display new column

df.sample(2)
```

Out[33]:

	airline_airline_id	airline_name	airline_alias	airline_iata	airline_icao	airline_callsign	airline_country	airline_active	src_airpor
54511	2297	easyJet	EasyJet Airline	U2	EZY	EASY	United Kingdom	True	
39655	4559	Swiss International Air Lines	Swiss Airlines	LX	SWR	SWISS	Switzerland	True	

2 rows × 40 columns

In [34]:

```python
# create df parttitions based on kv_key

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

END

Assignment 7.1.b

In [36]:

```python
# function to create hashvalue

import hashlib

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

In [38]:

```python
# new column hashed using hash value functon

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

In [41]:

```python
# create new column for partitioning based on hashed column

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

In [42]:

```python
# display new column

df[['hashed','hash_key']][:4]
```

Out[42]:

	hashed	hash_key
0	652cdec02010381f175efe499e070c8cbaac1522bac59a...	6
1	9eea5dd88177f8d835b2bb9cb27fb01268122b635b241a...	9
2	161143856af25bd4475f62c80c19f68936a139f653c1d3...	1
3	39aa99e6ae2757341bede9584473906ef1089e30820c90...	3

In [43]:

```python
# create df parttitions based on hash_key

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

END

Assignment 7.1.c

In [47]:

```python
# 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_lo
```

In [48]:

```python
# display new column value to make sure geohash values are correct

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

Out[48]:

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

In [73]:

```python
#  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]
```

In [75]:

```python
# create new columns location to store closest data center

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

In [76]:

```python
# display new column value to make sure location values are correct

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

Out[76]:

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

In [77]:

```python
# create df parttitions based on hash_key

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

END

Assignment 7.1.d

In [125…

```python
#  function to create partiions


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
```

In [130…

```python
df.sample(3)
```

Out[130…

	airline_airline_id	airline_name	airline_alias	airline_iata	airline_icao	airline_callsign	airline_country	airline_active	src_airpor
63125	4292	Rwandair Express	Qantas Airways	WB	RWD	RWANDAIR	Rwanda	True	
50758	130	Aeroflot Russian Airlines	\N	SU	AFL	AEROFLOT	Russia	True	
992	4608	Sichuan Airlines	Swiss European	3U	CSC	SI CHUAN	China	True	

3 rows × 44 columns

In [134…

```python
# validate the function - balance_partitions

airline_names = df.airline_iata.sample(50).to_list()

partitions = balance_partitions(airline_names,5)
partitions
```

Out[134…

```
[['AZ', 'CA', 'FL', 'FL', 'HU', 'SQ', 'UN', 'UN', 'W6', 'ZH'],
 ['AF', 'AF', 'AS', 'AZ', 'CZ', 'IG', 'OS', 'OZ', 'US', 'Y4'],
 ['AA', 'AB', 'AF', 'AZ', 'BC', 'CX', 'EP', 'HU', 'U2', 'VB'],
 ['G4', 'GR', 'ME', 'MF', 'NK', 'SV', 'UA', 'US', 'V7', 'VJ'],
 ['AM', 'CZ', 'CZ', 'HU', 'JD', 'JD', 'KL', 'TO', 'US', 'WY']]
```

END
