assignment7

July 23, 2021

[1]:
```
# 7.1 a
```

[3]:
```
import os
import json
from pathlib import Path
import gzip
import hashlib
import shutil
import pandas as pd
import pygeohash
import s3fs
```

[4]:
```
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␣
 ↪records])
df = create_flattened_dataset()
df['key'] = df['src_airport_iata'].astype(str) + df['dst_airport_iata'].
 ↪astype(str) + df['airline_iata'].astype(str)
```

[2]:
```
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')
    )
```

[8]:
```
# create this directory structure is to create a new key called kv_key from the␣
 ↪key column and use the to_parquet method
# with partition_cols=['kv_key'] to save a partitioned dataset

df['kv_key'] = df['key'].apply(lambda x: k for k in x.values if k != None),␣
 ↪axis=1)
```

```
0        AERKZN2B
1        ASFKZN2B
2        ASFMRV2B
3        CEKKZN2B
4        CEKOVB2B
           …
67658    WYAADLZL
67659    DMEFRUZM
67660    FRUDMEZM
67661    FRUOSSZM
67662    OSSFRUZM
Name: key, Length: 67663, dtype: object
```

[8]:
```
table = df.to_parquet()
table(path="/home/jovyan/dsc650-1/dsc650/assignments/assignment07/results/kv",␣
 ↪partition_cols=['kv_key'])
```

```
---------------------------------------------------------------------------
```

```
TypeError                                 Traceback (most recent call last)
<ipython-input-8-0a3d66cd379f> in <module>
      1 table = df.to_parquet()
----> 2 table(path="/home/jovyan/dsc650-1/dsc650/assignments/assignment07/
 ↪results/kv", partition_cols=['kv_key'])

TypeError: 'bytes' object is not callable
```

[3]:
```
# 7.1 b
```

[5]:
```
import hashlib

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

[ ]:
```
# create new hash column, hashed value of key column
```

[9]:
```
# 7.1 c
```

[ ]:
```
df['src_airport_geohash'] = df.apply(
    lambda row: pygeohash.encode(row.src_airport_latitude, row.
 ↪src_airport_longitude), axis=1
)
def determine_location(src_airport_geohash):
    locations = dict(
        central=pygeohash.encode(41.1544433, -96.0422378),
        ## TODO: add west and east
    )

    distances = #TODO: a list of centers and distances using the pygeohash.
 ↪geohash_haversine_distance function

    distances.sort()
    return distances[0][1]
df['location'] = df['src_airport_geohash'].apply(determine_location)
df.to_parquet('results/geo', partition_cols=['location'])
```

[ ]:
```

```

[1]:
```
# 7.1 d
```

[ ]:
```
keys = np.random.randint(low = 1, high = 20, size = 6)

print(f"{len(keys)} keys: {keys}")
```

```python
print("")
for i in range(1, len(keys)+1):
    print(f'{str(i)+" partitions":<15}')
    for j in np.arange(0,i):
        print(f' {j+1:>3} | {balance_partitions(keys,i)[j]}', end = "\n")
        print("")
```