Assignment 3

Name: Anjani Bonda
Course: DSC650 - Big Data
Date: 4/1/2023

Import libraries and define common helper functions

In [1]:
```
import os
import sys
import gzip
import json
from pathlib import Path
import csv

import pandas as pd
import s3fs
import pyarrow as pa
from pyarrow.json import read_json
import pyarrow.parquet as pq
from fastavro import parse_schema, writer, reader
import pygeohash
import snappy
import jsonschema
from jsonschema.exceptions import ValidationError
```

In [2]:
```
endpoint_url='https://storage.budsc.midwest-datascience.com'

current_dir = Path(os.getcwd()).absolute()
schema_dir = current_dir.joinpath('schemas')
results_dir = current_dir.joinpath('results')
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 read_jsonl_data_local():
    '''Creating a function to read the file from local'''
    src_data_path = r'/Users/anjanibonda/Bellevue_Git_Repos/dsc650/data/pro
    with 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
```

Load the records from https://storage.budsc.midwest-datascience.com/data/processed/openflights/routes.jsonl.gz (https://storage.budsc.midwest-datascience.com/data/processed/openflights/routes.jsonl.gz)

In [3]:
```
records = read_jsonl_data_local()
```

3.1.a JSON Schema

In [5]:
```
def validate_jsonl_data(records):
    schema_path = schema_dir.joinpath('routes-schema.json')
    with open(schema_path) as f:
        schema = json.load(f)

    validation_csv_path = results_dir.joinpath('validation-results.csv')
    with open(validation_csv_path, 'w') as f:
        for i, record in enumerate(records):
            try:
                ## TODO: Validate record
                jsonschema.validate(record, schema)
                pass
            except ValidationError as e:
                ## Print message if invalid record
                f.write(f"Error: {e.message}; failed validating {e.validato
                print(e)
                pass

validate_jsonl_data(records)
```

In [6]:
```
## Printing the json schema validation result
validation_csv_path = results_dir.joinpath('validation-results.csv')

if os.path.getsize(validation_csv_path) > 0:
    print("Json schema file has been validated with errors;")
    print("Please check {} file for more details".format(validation_csv_pat
else:
    print("Json schema file has been validated with no error")
```

```
Json schema file has been validated with no error
```

3.1.b Avro

In [43]:
```
def create_avro_dataset(records):
    schema_path = schema_dir.joinpath('routes.avsc')
    data_path = results_dir.joinpath('routes.avro')
    ## TODO: Use fastavro to create Avro dataset
    ## load schema .avro file
    with open(schema_path,'r') as f:
        schema = json.load(f)
    # parse schema
    parsed_schema = parse_schema(schema)
    # write record according to schema
    with open(data_path, 'wb') as out:
        writer(out, parsed_schema, records)
try:
    create_avro_dataset(records)
except Exception as e:
    print("Avro file creation has been failed with below error")
    print(e.message)
else:
    print("Avro file creation is successful")
```

```
Avro file creation has been failed with below error
```

```
---------------------------------------------------------------------------
--
SchemaParseException                      Traceback (most recent call las
t)
Input In [43], in <cell line: 13>()
     13 try:
---> 14     create_avro_dataset(records)
     15 except Exception as e:

Input In [43], in create_avro_dataset(records)
      8 # parse schema
----> 9 parsed_schema = parse_schema(schema)
     10 # write record according to schema

File fastavro/_schema.pyx:146, in fastavro._schema.parse_schema()

File fastavro/_schema.pyx:381, in fastavro._schema._parse_schema()

File fastavro/_schema.pyx:449, in fastavro._schema.parse_field()

File fastavro/_schema.pyx:372, in fastavro._schema._parse_schema()

File fastavro/_schema.pyx:168, in fastavro._schema._raise_default_value_e
rror()

SchemaParseException: Default value <NONE> must match schema type: record

During handling of the above exception, another exception occurred:

AttributeError                            Traceback (most recent call las
t)
Input In [43], in <cell line: 13>()
     15 except Exception as e:
     16     print("Avro file creation has been failed with below error")
---> 17     print(e.message)
     18 else:
     19     print("Avro file creation is successful")

AttributeError: 'SchemaParseException' object has no attribute 'message'
```

3.1.c Parquet

```
In [9]: def create_parquet_dataset():
            #src_data_path = 'data/processed/openflights/routes.jsonl.gz'
            src_data_path = r'/Users/anjanibonda/Bellevue_Git_Repos/dsc650/data/pro
            parquet_output_path = results_dir.joinpath('routes.parquet')
            s3 = s3fs.S3FileSystem(
                anon=True,
                client_kwargs={
                    'endpoint_url': endpoint_url
                }
            )

            with open(src_data_path, 'rb') as f_gz:
                with gzip.open(f_gz, 'rb') as f:
                # read json into Parquet table:
                    table = read_json(f)

            ## TODO: Use Apache Arrow to create Parquet table and save the dataset
            pq.write_table(table, parquet_output_path)

        try:
            create_parquet_dataset()
        except Exception as e:
            print("Parquet dataset creation has been failed with below error")
            print(e.message)
        else:
            print("Parquet dataset creation is successful")
```

```
Parquet dataset creation is successful
```

3.1.d Protocol Buffers

In [10]:
```
sys.path.insert(0, os.path.abspath('routes_pb2'))

import routes_pb2

def _airport_to_proto_obj(airport):
    obj = routes_pb2.Airport()
    if airport is None:
        return None
    if airport.get('airport_id') is None:
        return None

    obj.airport_id = airport.get('airport_id')
    if airport.get('name'):
        obj.name = airport.get('name')
    if airport.get('city'):
        obj.city = airport.get('city')
    if airport.get('iata'):
        obj.iata = airport.get('iata')
    if airport.get('icao'):
        obj.icao = airport.get('icao')
    if airport.get('altitude'):
        obj.altitude = airport.get('altitude')
    if airport.get('timezone'):
        obj.timezone = airport.get('timezone')
    if airport.get('dst'):
        obj.dst = airport.get('dst')
    if airport.get('tz_id'):
        obj.tz_id = airport.get('tz_id')
    if airport.get('type'):
        obj.type = airport.get('type')
    if airport.get('source'):
        obj.source = airport.get('source')

    obj.latitude = airport.get('latitude')
    obj.longitude = airport.get('longitude')

    return obj


def _airline_to_proto_obj(airline):
    obj = routes_pb2.Airline()
    ## TODO: Create an Airline obj using Protocol Buffers API
    # Check for airline id
    if airline is None:
        return None
    if airline.get('airline_id') is None:
        return None

    # Get airline info
    obj.airline_id = airline.get('airline_id')
    if airline.get('name'):
        obj.name = airline.get('name')
    if airline.get('alias'):
        obj.alias = airline.get('alias')
    if airline.get('iata'):
        obj.iata = airline.get('iata')
    if airline.get('icao'):
        obj.icao = airline.get('icao')
    if airline.get('callsign'):
        obj.callsign = airline.get('callsign')
    if airline.get('country'):
        obj.country = airline.get('country')
    obj.active = airline.get('active') # boolean

    return obj


def create_protobuf_dataset(records):
    routes = routes_pb2.Routes()
    for record in records:
        route = routes_pb2.Route()
        ## TODO: Implement the code to create the Protocol Buffers Dataset
         # Copy 'airline' data
        airline = _airline_to_proto_obj(record.get('airline'))
        if airline:
            route.airline.CopyFrom(airline)

        # Copy 'src_airport' data
        src_airport = _airport_to_proto_obj(record.get('src_airport'))
        if src_airport:
            route.src_airport.CopyFrom(src_airport)

        # Copy 'dst_airport' data
        dst_airport = _airport_to_proto_obj(record.get('dst_airport'))
        if dst_airport:
            route.dst_airport.CopyFrom(dst_airport)

        # Get 'codeshare' boolean
        route.codeshare = record.get('codeshare')

        # Get 'equipment' and iterate through for multiple
        equipment = record.get('equipment')
        for equip in equipment:
            route.equipment.append(equip)

        routes.route.append(route)

    data_path = results_dir.joinpath('routes.pb')

    with open(data_path, 'wb') as f:
        f.write(routes.SerializeToString())

    compressed_path = results_dir.joinpath('routes.pb.snappy')

    with open(compressed_path, 'wb') as f:
        f.write(snappy.compress(routes.SerializeToString()))

try:
    create_protobuf_dataset(records)
except Exception as e:
    print("Route database creation is failed with below reason")
    print(e)
else:
    print("Route database creation is successful")
```

```
Route database creation is successful
```

3.2.a Simple Geohash Index

```
In [11]: import collections
```

```
In [12]: def create_hash_dirs(records):
             geoindex_dir = results_dir.joinpath('geoindex')
             geoindex_dir.mkdir(exist_ok=True, parents=True)
             hashes = []
             hashes_dict = {}
             ## TODO: Create hash index

             for record in records: # iterate records
                 origin_data = record.get('src_airport') # get source airport info
                 if origin_data: # if source airport available, get lat/lon
                     lat, lon = origin_data.get('latitude'), origin_data.get('longit
                     record['src_airport']['geohash'] = pygeohash.encode(lat, lon) #
                     key = pygeohash.encode(lat, lon, precision=3)

                     ## Add first three digit of hash values to hashes list
                     if key not in hashes:
                         hashes.append(key)

                     if key in hashes_dict.keys():
                         hashes_dict[key].append(pygeohash.encode(lat, lon))
                     else:
                         hashes_dict[key] = [pygeohash.encode(lat, lon)]

             hashes = sorted(hashes) ## Sort the hash values
             hashes_od = collections.OrderedDict(sorted(hashes_dict.items())) ##Sort

             for key, values in hashes_od.items():
                 # create folder / subfolder directories by short hash key
                 output_dir = geoindex_dir.joinpath(str(key[:1])).joinpath(str(key[:
                 output_dir.mkdir(exist_ok=True, parents=True)
                 output_path = output_dir.joinpath(f'{key}.jsonl.gz')
                 # save record to appropriate subfolder/file
                 with gzip.open(output_path, 'w') as f:
                     json_output = '\n'.join([json.dumps(value) for value in values]
                     f.write(json_output.encode('utf-8'))

         try:
             create_hash_dirs(records)
         except Exception as e:
             print("The hash index creation process is failed")
         else:
             print("The has index creation process is completed successfully")
```

```
The has index creation process is completed successfully
```

3.2.b Simple Search Feature

In [13]:
```
def airport_search(records, latitude, longitude, distance):
    ## TODO: Create simple search to return nearest airport
    ## Calculate hashvalue for source latitude and longitude
    srcHash = pygeohash.encode(latitude, longitude)
    airports = []

    ## Iterate through records and get source airport
    for record in records:
        src_airport = record['src_airport']
        if src_airport:
            src_airportHash = pygeohash.encode(src_airport['latitude'], src
            ## Calulcate the distance in KM
            distToLockm = pygeohash.geohash_approximate_distance(srcHash, s
            ## If the distance is within given distance and airport is not
            if distToLockm <= distance and src_airport['name'] not in airpo
                airports.append(src_airport['name'])

    ## Sort the airports
    airports = sorted(airports)

    print(f'The following airports are within {distance} km of ({latitude},
    for airport in airports:
        print(airport)
```

In [14]:
```
## Getting input from user for latitude, longitude and distance in km
while True:
    try:
        lat = float(input("Enter the latitude: "))
    except:
        print("Enter correct value for latitude")
    else:
        while True:
            try:
                lon = float(input("Enter the longitude: "))
            except:
                print("Enter correct value for longitude")
            else:
                while True:
                    try:
                        dist = float(input("Enter the search radius(km): ")
                    except:
                        print("Enter correct value for radius(km)")
                    else:
                        break
                break
        break
```

```
Enter the latitude: 41.1499988
Enter the longitude: -95.91779
Enter the search radius(km): 1000
```

In [15]:
```
## Calling airport search function to calculate the nearby airports
airport_search(records, lat, lon, dist)
```

```
The following airports are within 1000.0 km of (41.1499988, -95.91779):
Central Nebraska Regional Airport
Chippewa Valley Regional Airport
Des Moines International Airport
Dubuque Regional Airport
Eppley Airfield
Huron Regional Airport
Joe Foss Field Airport
Kirksville Regional Airport
La Crosse Municipal Airport
Lincoln Airport
Mc Cook Ben Nelson Regional Airport
Minneapolis-St Paul International/Wold-Chamberlain Airport
North Platte Regional Airport Lee Bird Field
Pierre Regional Airport
Quad City International Airport
Quincy Regional Baldwin Field
Rochester International Airport
Sioux Gateway Col. Bud Day Field
Southeast Iowa Regional Airport
The Eastern Iowa Airport
Waterloo Regional Airport
Watertown Regional Airport
```

In [ ]: