Assignment 3

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
import fastavro
import pygeohash
import snappy
import jsonschema
from jsonschema.exceptions import ValidationError


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)
```

In [2]:
```


def read_jsonl_data():
    #s3 = s3fs.S3FileSystem(
    #    anon=True,
    #    client_kwargs={
    #        'endpoint_url': endpoint_url
    #    }
    #)
    #updated this to new path since not working
    src_data_path = '/home/jovyan/data/processed/openflights/routes.jsonl.gz'
    with gzip.open(src_data_path, 'rb') as f:
        records = [json.loads(line) for line in f.readlines()]
    return records
```

In [3]:
```
#src_data_path = 'routes.jsonl.gz'
#with gzip.open(src_data_path, 'rb') as f:
#    records = [json.loads(line) for line in f.readlines()]
```

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 [4]:
```
records = read_jsonl_data()
```

3.1

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)

    with open(schema_path, 'w') as f:
        for i, record in enumerate(records):
            try:
                ## TODO: Validate record
                pass
            except ValidationError as e:
                ## Print message if invalid record
                pass


validate_jsonl_data(records)
```

```
 not kw):
--> 357         return _default_decoder.decode(s)
    358     if cls is None:
    359         cls = JSONDecoder

/opt/conda/lib/python3.8/json/decoder.py in decode(self, s, _w)
    335
    336         """
--> 337         obj, end = self.raw_decode(s, idx=_w(s, 0).end())
    338         end = _w(s, end).end()
    339         if end != len(s):

/opt/conda/lib/python3.8/json/decoder.py in raw_decode(self, s, idx)
    353             obj, end = self.scan_once(s, idx)
    354         except StopIteration as err:
--> 355             raise JSONDecodeError("Expecting value", s, err.value) fr
om None
    356         return obj, end
```

3.1.b Avro

In [6]:
```
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


create_avro_dataset(records)
```

3.1.c Parquet

In [7]:
```
def create_parquet_dataset():
    src_data_path = 'routes.jsonl.gz'
    parquet_output_path = results_dir.joinpath('routes.parquet')

    s3 = s3fs.S3FileSystem(
        anon=True,
        client_kwargs={
            'endpoint_url': endpoint_url
        }
    )
    # updated this to reflect new path since s3 not working
    with gzip.open(src_data_path, 'rb') as f:
        pass
            ## TODO: Use Apache Arrow to create Parquet table and save the datase

create_parquet_dataset()
```

3.1.d Protocol Buffers

In [8]:
```
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
    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

        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()))

create_protobuf_dataset(records)
```

```
---------------------------------------------------------------------------
EncodeError                                Traceback (most recent call last)
<ipython-input-8-4c1c7200e19f> in <module>
     62         f.write(snappy.compress(routes.SerializeToString()))
     63
---> 64 create_protobuf_dataset(records)

<ipython-input-8-4c1c7200e19f> in create_protobuf_dataset(records)
     55
     56     with open(data_path, 'wb') as f:
---> 57         f.write(routes.SerializeToString())
     58
     59     compressed_path = results_dir.joinpath('routes.pb.snappy')

EncodeError: Message dsc650.assignment03.Routes is missing required fields: r
oute[0].codeshare,route[1].codeshare,route[2].codeshare,route[3].codeshare,ro
ute[4].codeshare,route[5].codeshare,route[6].codeshare,route[7].codeshare,rou
te[8].codeshare,route[9].codeshare,route[10].codeshare,route[11].codeshare,ro
ute[12].codeshare,route[13].codeshare,route[14].codeshare,route[15].codeshar
```

3.2

3.2.a Simple Geohash Index

In [9]:
```
def create_hash_dirs(records):
    geoindex_dir = results_dir.joinpath('geoindex')
    geoindex_dir.mkdir(exist_ok=True, parents=True)
    hashes = []
    ## TODO: Create hash index

create_hash_dirs(records)
```

3.2.b Simple Search Feature

In [10]:
```
def airport_search(latitude, longitude):
    ## TODO: Create simple search to return nearest airport
    pass

airport_search(41.1499988, -95.91779)
```