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 avro
import fastavro
from fastavro import writer, reader, parse_schema
import pygeohash
import snappy
import jsonschema
from jsonschema.exceptions import ValidationError
from jsonschema import validate


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
```

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

In [74]:
```
with gzip.open('routes.jsonl.gz') as f:
    records = [json.loads(line) for line in f.readlines()]
```

3.1

3.1.a JSON Schema

In [68]:
```
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/validation.csv"
    with open(validation_csv_path, 'w') as f:
        for i, record in enumerate(records):
            record['airline']['active'] == True
            record['codeshare'] == True
            try:
                validate(instance=record, schema=schema)
                pass
            except ValidationError as e:
                print(record)
                print("Record is Not Valid")
                pass


validate_jsonl_data(records)
```

3.1.b Avro

In [4]:
```
def create_avro_dataset(records):
    schema_path = schema_dir.joinpath('routes.avsc')
    data_path = results_dir.joinpath('routes.avro')

    with open(schema_path, 'rb') as schema_file:
        schema = fastavro.schema.load_schema(schema_file.name)

    with open(data_path, 'wb') as out:
        writer(out, schema, records)


create_avro_dataset(records)
```

3.1.c Parquet

In [56]:
```
def create_parquet_dataset():
    src_data_path = 'data/processed/openflights/routes.jsonl.gz'
    parquet_output_path = results_dir.joinpath('routes.parquet')
    s3 = s3fs.S3FileSystem(
        anon=True,
```

```
        client_kwargs={
            'endpoint_url': endpoint_url
        }
    )

    table = read_json('routes.jsonl')
    pq.write_table(table, 'results/routes.parquet')

create_parquet_dataset()
```

3.1.d Protocol Buffers

In [77]:

```
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()
    if airline is None:
        return None
```

```
    if airline.get('airline_id') is None:
        return None
    if airline.get('active'):
        obj.active = airline.get('active')
    else:
        obj.active = False
    if airline.get('airline_id'):
        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')

    return obj


def create_protobuf_dataset(records):
    routes = routes_pb2.Routes()
    for record in records:
        route = routes_pb2.Route()
        route.codeshare = record.get('codeshare')
        airline = _airline_to_proto_obj(record.get('airline', {}))
        if airline:
            route.airline.CopyFrom(airline)
        src_airport = _airport_to_proto_obj(record.get('src_airport', {}))
        if src_airport:
            route.src_airport.CopyFrom(src_airport)
        dst_airport = _airport_to_proto_obj(record.get('dst_airport', {}))
        if dst_airport:
            route.dst_airport.CopyFrom(dst_airport)

        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)
```

3.1.d Results Comparison

In [78]:
```
compression = ['Uncompressed', 'GZip', 'Avro', 'Parquet', 'Snappy']
size = ['56.4MB', '3.2MB', '18.7MB', '1.9MB', '21KB']

comp = pd.DataFrame(list(zip(compression, size)), columns = ['Compression',
comp.to_csv('results/comparison.csv', index=False)
```

3.2

3.2.a Simple Geohash Index

In [6]:
```
def create_hash_dirs(records):
    geoindex_dir = results_dir.joinpath('geoindex')
    geoindex_dir.mkdir(exist_ok=True, parents=True)
    hashes = []

    for record in records:
        src_airport = record.get('src_airport', {})
        if src_airport:
            latitude = src_airport.get('latitude')
            longitude = src_airport.get('longitude')
            if latitude and longitude:
                h = pygeohash.encode(latitude, longitude)
                hashes.append(h)

    hashes.sort()
    three_letter = sorted(list(set([entry[:3] for entry in hashes])))
    hash_index = {value: [] for value in three_letter}
    for record in records:
        geohash = record.get('geohash')
        if geohash:
            hash_index[geohash[:3]].append(record)
    for key, values in hash_index.items():
        output_dir = geoindex_dir.joinpath(str(key[:1])).joinpath(str(key[:2
        output_dir.mkdir(exist_ok=True, parents=True)
        output_path = output_dir.joinpath('{}.jsonl.gz'.format(key))
        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'))

create_hash_dirs(records)
```

3.2.b Simple Search Feature

In [58]:
```
def airport_search(latitude, longitude):
    loc = pygeohash.encode(latitude, longitude)

    geoindex_dir = results_dir.joinpath('geoindex')
    geoindex_dir.mkdir(exist_ok=True, parents=True)

    hashes = []
    airports = []
```

```
    for record in records:
        src_airport = record.get('src_airport', {})
        if src_airport:
            latitude = src_airport.get('latitude')
            longitude = src_airport.get('longitude')
            if latitude and longitude:
                h = pygeohash.encode(latitude, longitude)
                hashes.append(h)
                airports.append(record['src_airport']['name'])

    air = pd.DataFrame(list(zip(airports, hashes)), columns = ['Airport', 'H

    min = ['X', 200000000]

    for index, row in air.iterrows():
        dist = pygeohash.geohash_approximate_distance(row['Hash'], loc)
        if dist < min[1]:
            min[0] = row['Airport']
            min[1] = dist

    miles = min[1]/1609

    print(f"The nearest airport is " + min[0] +
          ", which is located {:.2f} miles away.".format(miles))


airport_search(41.1499988, -95.91779)
```

```
The nearest airport is Eppley Airfield, which is located 12.15 miles away.
```