Assignment 3

In [8]:
```
! pip install pyarrow
! pip install fastavro
! pip install pygeohash
! pip install snappy
! pip install jsonschema
! pip install google
! pip install protobuf
```

```
Requirement already satisfied: pyarrow in c:\users\spashtunyar\anaconda3\lib\site-pac
kages (8.0.0)
Requirement already satisfied: numpy>=1.16.6 in c:\users\spashtunyar\anaconda3\lib\si
te-packages (from pyarrow) (1.23.5)
Requirement already satisfied: fastavro in c:\users\spashtunyar\anaconda3\lib\site-pa
ckages (1.7.3)
Requirement already satisfied: pygeohash in c:\users\spashtunyar\anaconda3\lib\site-p
ackages (1.2.0)
Collecting snappy
  Downloading snappy-3.0.3-cp39-cp39-win_amd64.whl (11.6 MB)
     ---------------------------------------- 11.6/11.6 MB 10.1 MB/s eta 0:00:00
Collecting plink>=2.4.1
  Downloading plink-2.4.1-py3-none-any.whl (339 kB)
     ---------------------------------------- 339.5/339.5 kB 10.6 MB/s eta 0:00:00
Requirement already satisfied: decorator in c:\users\spashtunyar\anaconda3\lib\site-p
ackages (from snappy) (5.1.1)
Collecting FXrays>=1.3
  Downloading FXrays-1.3.5-cp39-cp39-win_amd64.whl (25 kB)
Collecting snappy-manifolds>=1.1.2
  Downloading snappy_manifolds-1.1.2-py3-none-any.whl (45.0 MB)
     ---------------------------------------- 45.0/45.0 MB 18.7 MB/s eta 0:00:00
Collecting pypng
  Downloading pypng-0.20220715.0-py3-none-any.whl (58 kB)
     ---------------------------------------- 58.1/58.1 kB 3.0 MB/s eta 0:00:00
Collecting spherogram>=2.1
  Downloading spherogram-2.1-cp39-cp39-win_amd64.whl (319 kB)
     ---------------------------------------- 319.5/319.5 kB 19.3 MB/s eta 0:00:00
Requirement already satisfied: ipython>=5.0 in c:\users\spashtunyar\anaconda3\lib\sit
e-packages (from snappy) (8.10.0)
Collecting cypari>=2.3
  Downloading cypari-2.4.1-cp39-cp39-win_amd64.whl (5.2 MB)
     ---------------------------------------- 5.2/5.2 MB 19.5 MB/s eta 0:00:00
Requirement already satisfied: future in c:\users\spashtunyar\anaconda3\lib\site-pack
ages (from cypari>=2.3->snappy) (0.18.2)
Requirement already satisfied: six in c:\users\spashtunyar\anaconda3\lib\site-package
s (from cypari>=2.3->snappy) (1.16.0)
Requirement already satisfied: pickleshare in c:\users\spashtunyar\anaconda3\lib\site
-packages (from ipython>=5.0->snappy) (0.7.5)
Requirement already satisfied: traitlets>=5 in c:\users\spashtunyar\anaconda3\lib\sit
e-packages (from ipython>=5.0->snappy) (5.7.1)
Requirement already satisfied: colorama in c:\users\spashtunyar\anaconda3\lib\site-pa
ckages (from ipython>=5.0->snappy) (0.4.6)
Requirement already satisfied: backcall in c:\users\spashtunyar\anaconda3\lib\site-pa
ckages (from ipython>=5.0->snappy) (0.2.0)
Requirement already satisfied: prompt-toolkit<3.1.0,>=3.0.30 in c:\users\spashtunyar
\anaconda3\lib\site-packages (from ipython>=5.0->snappy) (3.0.36)
Requirement already satisfied: pygments>=2.4.0 in c:\users\spashtunyar\anaconda3\lib
\site-packages (from ipython>=5.0->snappy) (2.11.2)
Requirement already satisfied: jedi>=0.16 in c:\users\spashtunyar\anaconda3\lib\site-
packages (from ipython>=5.0->snappy) (0.18.1)
Requirement already satisfied: matplotlib-inline in c:\users\spashtunyar\anaconda3\li
b\site-packages (from ipython>=5.0->snappy) (0.1.6)
Requirement already satisfied: stack-data in c:\users\spashtunyar\anaconda3\lib\site-
packages (from ipython>=5.0->snappy) (0.2.0)
Requirement already satisfied: networkx in c:\users\spashtunyar\anaconda3\lib\site-pa
ckages (from spherogram>=2.1->snappy) (2.8.4)
Collecting knot-floer-homology>=1.1
  Downloading knot_floer_homology-1.2-cp39-cp39-win_amd64.whl (67 kB)
     ---------------------------------------- 67.9/67.9 kB ? eta 0:00:00
Requirement already satisfied: parso<0.9.0,>=0.8.0 in c:\users\spashtunyar\anaconda3
```

```
\lib\site-packages (from jedi>=0.16->ipython>=5.0->snappy) (0.8.3)
Requirement already satisfied: wcwidth in c:\users\spashtunyar\anaconda3\lib\site-pac
kages (from prompt-toolkit<3.1.0,>=3.0.30->ipython>=5.0->snappy) (0.2.5)
Requirement already satisfied: pure-eval in c:\users\spashtunyar\anaconda3\lib\site-p
ackages (from stack-data->ipython>=5.0->snappy) (0.2.2)
Requirement already satisfied: asttokens in c:\users\spashtunyar\anaconda3\lib\site-p
ackages (from stack-data->ipython>=5.0->snappy) (2.0.5)
Requirement already satisfied: executing in c:\users\spashtunyar\anaconda3\lib\site-p
ackages (from stack-data->ipython>=5.0->snappy) (0.8.3)
Installing collected packages: snappy-manifolds, pypng, knot-floer-homology, spherogr
am, plink, FXrays, cypari, snappy
Successfully installed FXrays-1.3.5 cypari-2.4.1 knot-floer-homology-1.2 plink-2.4.1
pypng-0.20220715.0 snappy-3.0.3 snappy-manifolds-1.1.2 spherogram-2.1
Requirement already satisfied: jsonschema in c:\users\spashtunyar\anaconda3\lib\site-
packages (4.17.3)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in c:\us
ers\spashtunyar\anaconda3\lib\site-packages (from jsonschema) (0.18.0)
Requirement already satisfied: attrs>=17.4.0 in c:\users\spashtunyar\anaconda3\lib\si
te-packages (from jsonschema) (22.1.0)
Collecting google
  Downloading google-3.0.0-py2.py3-none-any.whl (45 kB)
     ---------------------------------------- 45.3/45.3 kB 1.1 MB/s eta 0:00:00
Requirement already satisfied: beautifulsoup4 in c:\users\spashtunyar\anaconda3\lib\s
ite-packages (from google) (4.11.1)
Requirement already satisfied: soupsieve>1.2 in c:\users\spashtunyar\anaconda3\lib\si
te-packages (from beautifulsoup4->google) (2.3.2.post1)
Installing collected packages: google
Successfully installed google-3.0.0
Requirement already satisfied: protobuf in c:\users\spashtunyar\anaconda3\lib\site-pa
ckages (3.20.3)
```

In [1]:
```python
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
```

In [2]:
```python
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)
```

Import libraries and define common helper functions

In [40]:
```python
#rewriting to pull from the path in my own directory
def read_jsonl_data_sp():
    src_data_path = r'C:\Users\spashtunyar\Documents\School\dsc650\data\processed\oper
```

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


    return records
```

Load the records from local directory

In [41]:
```
records = read_jsonl_data_sp()
```

In [42]:
```
#val
records[0:1]
```

Out[42]:
```
[{'airline': {'airline_id': 410,
   'name': 'Aerocondor',
   'alias': 'ANA All Nippon Airways',
   'iata': '2B',
   'icao': 'ARD',
   'callsign': 'AEROCONDOR',
   'country': 'Portugal',
   'active': True},
  'src_airport': {'airport_id': 2965,
   'name': 'Sochi International Airport',
   'city': 'Sochi',
   'country': 'Russia',
   'iata': 'AER',
   'icao': 'URSS',
   'latitude': 43.449902,
   'longitude': 39.9566,
   'altitude': 89,
   'timezone': 3.0,
   'dst': 'N',
   'tz_id': 'Europe/Moscow',
   'type': 'airport',
   'source': 'OurAirports'},
  'dst_airport': {'airport_id': 2990,
   'name': 'Kazan International Airport',
   'city': 'Kazan',
   'country': 'Russia',
   'iata': 'KZN',
   'icao': 'UWKD',
   'latitude': 55.606201171875,
   'longitude': 49.278701782227,
   'altitude': 411,
   'timezone': 3.0,
   'dst': 'N',
   'tz_id': 'Europe/Moscow',
   'type': 'airport',
   'source': 'OurAirports'},
  'codeshare': False,
  'equipment': ['CR2']}]
```

3.1

3.1.a JSON Schema

In [6]:
```
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('validation_csv_path', 'w', encoding='utf-8') 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.validator} in schem
                print(e)
                pass


validate_jsonl_data(records)
```

3.1.b Avro

In [64]:
```
from fastavro.schema import load_schema
```

In [67]:
```
schema
```

Out[67]:
```
{'type': 'record',
 'name': 'Route',
 'namespace': 'edu.bellevue.dsc650',
 'fields': [{'name': 'airline',
   'type': {'type': 'record',
    'name': 'Airline',
    'fields': [{'name': 'airline_id', 'type': 'int', 'default': -1},
     {'name': 'name', 'type': 'string', 'default': 'NONE'},
     {'name': 'alias', 'type': 'string', 'default': 'NONE'},
     {'name': 'iata', 'type': 'string', 'default': 'NONE'},
     {'name': 'icao', 'type': 'string', 'default': 'NONE'},
     {'name': 'callsign', 'type': 'string', 'default': 'NONE'},
     {'name': 'country', 'type': 'string', 'default': 'NONE'},
     {'name': 'active', 'type': 'boolean', 'default': False}]},
   'default': 'NONE'},
  {'name': 'src_airport',
   'type': [{'type': 'record',
     'name': 'Airport',
     'fields': [{'name': 'airport_id', 'type': 'int', 'default': -1},
      {'name': 'name', 'type': 'string', 'default': 'NONE'},
      {'name': 'city', 'type': 'string', 'default': 'NONE'},
      {'name': 'iata', 'type': 'string', 'default': 'NONE'},
      {'name': 'icao', 'type': 'string', 'default': 'NONE'},
      {'name': 'latitude', 'type': 'double'},
      {'name': 'longitude', 'type': 'double'},
      {'name': 'timezone', 'type': 'double'},
      {'name': 'dst', 'type': 'string', 'default': 'NONE'},
      {'name': 'tz_id', 'type': 'string', 'default': 'NONE'},
      {'name': 'type', 'type': 'string', 'default': 'NONE'},
      {'name': 'source', 'type': 'string', 'default': 'NONE'}]},
    'null'],
   'default': 'NONE'},
  {'name': 'dst_airport', 'type': ['Airport', 'null'], 'default': 'NONE'},
  {'name': 'codeshare', 'type': 'boolean', 'default': False},
  {'name': 'stops', 'type': 'int', 'default': 0},
  {'name': 'equipment', 'type': {'type': 'array', 'items': 'string'}}]}
```

In [57]:
```
from fastavro import writer, reader, parse_schema
```

In [71]:
```
pip install fastavro==1.5.1
```

```
Collecting fastavro==1.5.1
  Downloading fastavro-1.5.1-cp39-cp39-win_amd64.whl (435 kB)
     ---------------------------------------- 435.4/435.4 kB 3.0 MB/s eta 0:00:00
Installing collected packages: fastavro
  Attempting uninstall: fastavro
    Found existing installation: fastavro 1.7.3
    Uninstalling fastavro-1.7.3:
      Successfully uninstalled fastavro-1.7.3
Note: you may need to restart the kernel to use updated packages.
ERROR: Could not install packages due to an OSError: [WinError 5] Access is denied:
'C:\\Users\\spashtunyar\\Anaconda3\\Lib\\site-packages\\~astavro\\_logical_readers.cp
39-win_amd64.pyd'
Consider using the `--user` option or check the permissions.
```

In [74]:
```
#got it to work FINALLY!!! Abed in the teams chat provided a fixed acsc schema file th
```

In [75]:
```
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
    with open(schema_path, 'r') as f1:
        schema = json.loads(f1.read())
    parsed_schema = fastavro.parse_schema(schema)
    ## create dataset
    with open(data_path, 'wb') as out:
        fastavro.writer(out, parsed_schema, records)

create_avro_dataset(records)
```

In [76]:
```
# validation of what I created
data_path = results_dir.joinpath('routes.avro')
with open(data_path, mode = 'rb') as f:
    reader = fastavro.reader(f)
    records = [r for r in reader]
    df = pd.DataFrame.from_records(records)
    print(df.head())
```

```
                                             airline  \
0  {'airline_id': 410, 'name': 'Aerocondor', 'ali...
1  {'airline_id': 410, 'name': 'Aerocondor', 'ali...
2  {'airline_id': 410, 'name': 'Aerocondor', 'ali...
3  {'airline_id': 410, 'name': 'Aerocondor', 'ali...
4  {'airline_id': 410, 'name': 'Aerocondor', 'ali...

                                         src_airport  \
0  {'airport_id': 2965, 'name': 'Sochi Internatio...
1  {'airport_id': 2966, 'name': 'Astrakhan Airpor...
2  {'airport_id': 2966, 'name': 'Astrakhan Airpor...
3  {'airport_id': 2968, 'name': 'Chelyabinsk Bala...
4  {'airport_id': 2968, 'name': 'Chelyabinsk Bala...

                                         dst_airport  codeshare  stops  \
0  {'airport_id': 2990, 'name': 'Kazan Internatio...      False      0
1  {'airport_id': 2990, 'name': 'Kazan Internatio...      False      0
2  {'airport_id': 2962, 'name': 'Mineralnyye Vody...      False      0
3  {'airport_id': 2990, 'name': 'Kazan Internatio...      False      0
4  {'airport_id': 4078, 'name': 'Tolmachevo Airpo...      False      0

  equipment
0     [CR2]
1     [CR2]
2     [CR2]
3     [CR2]
4     [CR2]
```

3.1.c Parquet

In [10]:
```
def create_parquet_dataset():
    src_data_path = r'C:\Users\spashtunyar\Documents\School\dsc650\data\processed\oper
    parquet_output_path = results_dir.joinpath('routes.parquet')
    with gzip.open(src_data_path, 'rb') as f:
        table = read_json(f)
    pq.write_table(table, parquet_output_path)
```

```
create_parquet_dataset()
```

In [12]:
```
# Parquet validation
parquet_output_path = results_dir.joinpath('routes.parquet')
pqFile = pq.ParquetFile(parquet_output_path)
pqFile.metadata
```

Out[12]:
```
<pyarrow._parquet.FileMetaData object at 0x000001E574929400>
  created_by: parquet-cpp-arrow version 8.0.0
  num_columns: 38
  num_rows: 67663
  num_row_groups: 1
  format_version: 1.0
  serialized_size: 7567
```

3.1.d Protocol Buffers

In [20]:
```
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 not airline.get('name'):
        return None
    if not airline.get('airline_id'):
        return None

    obj.airline_id = airline.get('airline_id')
    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')
    if airline.get('active') is not None:
        obj.active = airline.get('active')

    return obj


def create_protobuf_dataset(records):
    routes = routes_pb2.Routes()
    for record in records:
        route = routes_pb2.Route()
        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)
        if record.get('codeshare'):
            route.codeshare = record.get('codeshare')
        else:
            route.codeshare = False
        if record.get('stops') is not None:
            route.stops = record.get('stops')
        if record.get('equipment'):
            route.equipment.extend(record.get('equipment'))

        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)
```

3.2

3.2.a Simple Geohash Index

In [21]:
```
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:
                hashes.append(pygeohash.encode(latitude, longitude))
    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 [22]:
```
def airport_search(latitude, longitude):
    ## TODO: Create simple search to return nearest airport
    a = pygeohash.encode(latitude, longitude)
    dist = 0
    name = ''

    for i, record in enumerate(records):
        src_airport = record.get('src_airport', {})
        if src_airport:
            lat = src_airport.get('latitude')
            long = src_airport.get('longitude')
            airport_name = src_airport.get('name')
            if lat and long:
                a1 = pygeohash.encode(lat, long)

                dist_n = pygeohash.geohash_approximate_distance(a, a1)
                if i==0:
```

```
                    dist = dist_n
                    name = airport_name
                else:
                    if dist > dist_n:
                        dist = dist_n
                        name = airport_name
    print(name)
```

In [30]:
```
#Validation searches, used google to pull examples
```

In [23]:
```
airport_search(41.1499988, -95.91779)
```

```
Eppley Airfield
```

In [26]:
```
airport_search(54.8028, 23.9172)
```

```
Vilnius International Airport
```

In [27]:
```
airport_search(37.61636, -122.391027)
```

```
San Francisco International Airport
```

In [28]:
```
airport_search(41.9803, -87.9090)
```

```
Chicago O'Hare International Airport
```

In [ ]:
```

```