# Assignment 03 Outputs and Code

## Assignment 3.1a Code Output (JSON Schema):

Graphical user interface, application

Description automatically generated

Graphical user interface

Description automatically generated

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

If validation-results has an error, then the message will be displayed in the csv file. Otherwise, it is empty.

## Assignment 3.1b Code Output (Avro):

Graphical user interface, application

Description automatically generated

## Assignment 3.1c Code Output (Parquet):

Graphical user interface

Description automatically generated

## Graphical user interface, application, Word Description automatically generated

## Assignment 3.1d Code Output (Protocol Buffers):

Graphical user interface, application

Description automatically generated

## Assignment 3.1e Code Output (Output Sizes):

## 

Graphical user interface, text, application, table

Description automatically generated

## Assignment 3.2a Code Output (Create a Simple Geohash Index):

Graphical user interface, application

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Graphical user interface, application

Description automatically generated

## Assignment 3.2b Code Output (Implement a Simple Search Feature):

Text

Description automatically generated

Example Code Output shown below:

Airports within 10000 meters from the latitude and longitude coordinates.

Latitude: 41.1499988

Longitude:-95.91779

{'Airport': 'Southeast Iowa Regional Airport', 'Geoval': '9zr0n2k7mg', 'Latitude': 40.783199310302734, 'Longitude': -91.12550354003906, 'Distance(meters)': 625.441}

{'Airport': 'Jonesboro Municipal Airport', 'Geoval': '9yrec6cqcb', 'Latitude': 35.83169937133789, 'Longitude': -90.64640045166016, 'Distance(meters)': 5003.53}

{'Airport': 'St Louis Lambert International Airport', 'Geoval': '9yzsrtfxd2', 'Latitude': 38.748697, 'Longitude': -90.370003, 'Distance(meters)': 5003.53}

{'Airport': 'Mid Delta Regional Airport', 'Geoval': '9vzq3x6k1b', 'Latitude': 33.482898712158196, 'Longitude': -90.98560333251952, 'Distance(meters)': 5003.53}

## Assignment 3.1a Code (JSON Schema):

def validate\_jsonl\_data(records):

schema\_path = schema\_dir.joinpath('routes-schema.json')

with open(schema\_path) as f:

schema = json.load(f)

print(schema)

## Error surfaced stating validation\_csv\_path not found. Adding path.

validation\_csv\_path = results\_dir.joinpath('validation-results.csv')

with open(validation\_csv\_path, 'w') as f:

for i, record in enumerate(records):

try:

## Validate record

jsonschema.validate(record, schema)

## No message passed to validation\_csv\_path if validation successful.

## pass

except ValidationError as e:

## Print message if invalid record

message\_detail = e.message

print(message\_detail)

f.write(str(e.path))

f.write(str(e.instance))

f.write(str(message\_detail))

return message\_detail

## pass

validate\_jsonl\_data(records)

## Assignment 3.1b Code (Avro):

## Import additional modules from fastavro

from fastavro.schema import load\_schema

from fastavro import writer, reader, parse\_schema

def create\_avro\_dataset(records):

schema\_path = schema\_dir.joinpath('routes.avsc')

data\_path = results\_dir.joinpath('routes.avro')

## Use fastavro to create Avro dataset

## Load the schema from the specified path. See Note at bottom of this cell for replacement of schema file.

avro\_schema = load\_schema(schema\_path)

## Write the records to routes.avro as specified.

with open(data\_path, 'wb') as output:

writer(output, avro\_schema, records)

create\_avro\_dataset(records)

## Note: Default was modified from None to null in schema file to eliminate errors. Replaced schema file with alternative file.

## Assignment 3.1c Code (Parquet):

def create\_parquet\_dataset():

## Replaced src\_data\_path with code Professor provided.

src\_data\_path = '../../../data/processed/openflights/routes.jsonl.gz'

parquet\_output\_path = results\_dir.joinpath('routes.parquet')

'''

Commenting out S3 portion of this function and replacing with code Professor provided.

s3 = s3fs.S3FileSystem(

anon=True,

client\_kwargs={

'endpoint\_url': endpoint\_url

}

)

with s3.open(src\_data\_path, 'rb') as f\_gz:

with gzip.open(f\_gz, 'rb') as f:

pass

'''

with gzip.open(src\_data\_path, 'rb') as f:

records = [json.loads(line) for line in f.readlines()]

## Use Apache Arrow to create Parquet table and save the dataset.

## Start by creating a dataframe.

df\_apache = pd.DataFrame(records)

## Create a table using pyarrow.

table\_apache = pa.Table.from\_pandas(df\_apache)

## Print out the results of the table.

print(table\_apache)

## Write to the routes.parquet file.

pq.write\_table(table\_apache, parquet\_output\_path, compression = 'none')

create\_parquet\_dataset()

## Assignment 3.1d Code (Protocol Buffers):

## Needed to install python-snappy due to error with snappy.compress() showing up.

## Uninstalled snappy and python-snappy. Reinstalled python-snappy.

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()

## Create an Airline obj using Protocol Buffers API.

## Follow the similar code as specified in \_airport\_to\_proto\_obj function, except for ailrine.

if airline is None:

return None

if airline.get('airline\_id') is None:

return None

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')

if airline.get('active'):

obj.active = airline.get('active')

return obj

def create\_protobuf\_dataset(records):

routes = routes\_pb2.Routes()

for record in records:

route = routes\_pb2.Route()

## Implement the code to create the Protocol Buffers Dataset

## Utilize for loop to obtain key, value pairs for airline, source airport, and distance airport.

for key, value in record.items():

if key== 'airline':

airline = \_airline\_to\_proto\_obj(value)

air\_input = route.airline

air\_input.name = airline.name

air\_input.airline\_id = airline.airline\_id

air\_input.active = airline.active

if key == 'src\_airport' and value is not None:

src\_airport = \_airport\_to\_proto\_obj(value)

src\_airport\_input = route.src\_airport

src\_airport\_input.name = src\_airport.name

src\_airport\_input.airport\_id = src\_airport.airport\_id

src\_airport\_input.latitude = src\_airport.latitude

src\_airport\_input.longitude = src\_airport.longitude

if key == 'dst\_airport' and value is not None:

dst\_airport = \_airport\_to\_proto\_obj(value)

dst\_airport\_input = route.dst\_airport

dst\_airport\_input.name = dst\_airport.name

dst\_airport\_input.airport\_id = dst\_airport.airport\_id

dst\_airport\_input.latitude = dst\_airport.latitude

dst\_airport\_input.longitude = dst\_airport.longitude

if key =='codeshare':

route.codeshare = value

routes.route.append(route)

data\_path = results\_dir.joinpath('routes.pb')

with open(data\_path, 'wb') as f:

bytes\_to\_string = routes.SerializeToString()

f.write(bytes\_to\_string)

'''

Resolved an issue with snappy.compress.

Reinstalled with python-snappy to resolve.

'''

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)

## Assignment 3.1e Code (Output Sizes):

'''

Working through Output Sizes of the files in compressed and uncompressed formats.

Used example code provided by classmate in Teams channel as starting point for this code.

Replaced the file paths with my local locations since I wasn't able to finish the assignment within the VM.

'''

import os

import csv

## Function get\_file\_size to return the file size in bytes.

def get\_file\_size(file\_path):

"""Get the size of a file in bytes"""

return os.stat(file\_path).st\_size

## Define the File paths

avro\_file = r"C:\Users\jkmey\Documents\Github\DSC650\_Course\_Assignments\dsc650\dsc650\assignments\assignment03\results\routes.avro"

pb\_file = r"C:\Users\jkmey\Documents\Github\DSC650\_Course\_Assignments\dsc650\dsc650\assignments\assignment03\results\routes.pb"

pb\_snappy\_file = r"C:\Users\jkmey\Documents\Github\DSC650\_Course\_Assignments\dsc650\dsc650\assignments\assignment03\results\routes.pb.snappy"

output\_file = r"C:\Users\jkmey\Documents\Github\DSC650\_Course\_Assignments\dsc650\dsc650\assignments\assignment03\results\comparison.csv"

## Get file sizes

avro\_size = get\_file\_size(avro\_file)

pb\_size = get\_file\_size(pb\_file)

pb\_snappy\_size = get\_file\_size(pb\_snappy\_file)

## Write results to CSV file

with open(output\_file, mode='w', newline='') as csv\_file:

fieldnames = ['File Format', 'Uncompressed Size (bytes)', 'Compressed Size (bytes)']

writer = csv.DictWriter(csv\_file, fieldnames=fieldnames)

writer.writeheader()

writer.writerow({'File Format': 'Avro', 'Uncompressed Size (bytes)': avro\_size, 'Compressed Size (bytes)': 'N/A'})

writer.writerow({'File Format': 'Protocol Buffers', 'Uncompressed Size (bytes)': pb\_size, 'Compressed Size (bytes)': 'N/A'})

writer.writerow({'File Format': 'Protocol Buffers (with Snappy compression)', 'Uncompressed Size (bytes)': pb\_size, 'Compressed Size (bytes)': pb\_snappy\_size})

## Print the results for confirmation.

print("Comparison results saved to:", output\_file)

## Assignment 3.2a Code (Create a Simple Geohash Index):

def create\_hash\_dirs(records):

geoindex\_dir = results\_dir.joinpath('geoindex')

geoindex\_dir.mkdir(exist\_ok=True, parents=True)

hashes = []

## Create hash index

## Iterate through the records with a for loop.

## Specify the geohash values to setup the index as specified for the assignment.

for record in records:

for key, value in record.items():

if key == 'src\_airport' and value is not None:

geohash\_value = pygeohash.encode(value['latitude'], value['longitude'])

geohash\_value\_1 = str(geohash\_value)[0]

geohash\_value\_2 = str(geohash\_value)[0:2]

geohash\_value\_3 = str(geohash\_value)[0:3]+".jsonl.gz"

geoindex\_dir = results\_dir.joinpath('geoindex')

geoindex\_dir\_1 = geoindex\_dir.joinpath(geohash\_value\_1)

geoindex\_dir\_1.mkdir(parents=True, exist\_ok = True)

geoindex\_dir\_2 = geoindex\_dir\_1.joinpath(geohash\_value\_2)

geoindex\_dir\_2.mkdir(parents=True, exist\_ok = True)

jsonfilename = geoindex\_dir\_2.joinpath(geohash\_value\_3)

with gzip.GzipFile(jsonfilename, 'w') as fout:

fout.write(json.dumps(value).encode('utf-8'))

create\_hash\_dirs(records)

## Assignment 3.2b Code (Implement a Simple Search Feature):

## Import unique\_everseen, iteration\_utilities.

from iteration\_utilities import unique\_everseen

## Added the distance\_km as an argument in the function airport\_search.

def airport\_search(latitude, longitude, distance\_km):

## Create simple search to return nearest airport

## Modify the input distance in Kilometers to meters (1km = 1000m)

distance\_m = distance\_km \* 1000

## Utilize pygeohash.encode().

source\_geoval = pygeohash.encode(latitude, longitude, precision = 10)

## Setup empty lists for the airport\_distances and rec\_out.

airport\_distances = []

rec\_out = []

## Setup for loop for unique source airports lookup

for record in records:

for key, value in record.items():

if key == 'src\_airport' and value is not None:

if value not in rec\_out:

rec\_out.append(value)

## Iterate through elmements in rec\_out to append to airport\_distances list.

## Get the name, latitude, and longitude, geohash value, and distance (meters)

for record in rec\_out:

distance\_name = record['name']

distance\_latitude = record['latitude']

distance\_longitude = record['longitude']

geohash\_value = pygeohash.encode(distance\_latitude, distance\_longitude, precision = 10)

distancem\_distancegeo\_1 = pygeohash.geohash\_approximate\_distance(source\_geoval, geohash\_value) / 1000

airport\_dist = {

"Airport" : distance\_name,

"Geoval" : geohash\_value,

"Latitude" : distance\_latitude,

"Longitude" : distance\_longitude,

"Distance(meters)" : distancem\_distancegeo\_1

}

airport\_distances.append(airport\_dist)

airport\_distance\_output = list(unique\_everseen(airport\_distances))

print("Airports within "+str(distance\_m)+" meters from the latitude and longitude coordinates.")

print("Latitude: "+str(latitude))

print("Longitude:"+str(longitude))

for i in range(len(airport\_distance\_output)):

for k, v in airport\_distance\_output[i].items():

if k == 'Distance(meters)':

if v <= distance\_m:

print(airport\_distance\_output[i])

## pass

## Try the search with a distance of 10km distance and previously specified latitude and longitude.

distance\_km = 10

airport\_search(41.1499988, -95.91779, distance\_km)