**Assignment 7 Code**

# Assignment 7.1.c

df['src\_airport\_geohash'] = df.apply(

lambda row: pygeohash.encode(row.src\_airport\_latitude, row.src\_airport\_longitude), axis=1

)

def determine\_location(src\_airport\_geohash):

locations = dict(

central=pygeohash.encode(41.1544433, -96.0422378),

## TODO: add west and east

)

distances = #TODO: a list of centers and distances using the pygeohash.geohash\_haversine\_distance function

distances.sort()

return distances[0][1]

df['location'] = df['src\_airport\_geohash'].apply(determine\_location)

df.to\_parquet('results/geo', partition\_cols=['location'])

**Assignment 3 Code**

## assignment 3.1.d

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')

## TODO

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', {}))

## TODO

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()))

# Assignment 3.2.a

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:

## TODO: use pygeohash.encode() to assign geohashes to the records and complete the hashes list

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'))

## assignment 2.2

from pathlib import Path

import json

import os

from tinydb import TinyDB

current\_dir = Path(os.getcwd()).absolute()

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

kv\_data\_dir = results\_dir.joinpath('kvdb')

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

def \_load\_json(json\_path):

with open(json\_path) as f:

return json.load(f)

**Assignment 2 Code**

class DocumentDB(object):

def \_\_init\_\_(self, db\_path):

## You can use the code from the previous exmaple if you would like

people\_json = kv\_data\_dir.joinpath('people.json')

visited\_json = kv\_data\_dir.joinpath('visited.json')

sites\_json = kv\_data\_dir.joinpath('sites.json')

measurements\_json = kv\_data\_dir.joinpath('measurements.json')

self.\_db\_path = Path(db\_path)

self.\_db = None

## TODO

self.\_load\_db()

def \_get\_site(self, site\_id):

return self.\_site\_lookup[str(site\_id)]

def \_get\_measurements(self, person\_id):

measurements = []

for values in self.\_measurements\_lookup.values():

measurements.extend([value for value in values if str(value['person\_id']) == str(person\_id)])

return measurements

def \_get\_visit(self, visit\_id):

visit = self.\_visit\_lookup.get(str(visit\_id))

## TODO: site\_id = ...

## TODO: site = ...

visit['site'] = site

return visit

def \_load\_db(self):

self.\_db = TinyDB(self.\_db\_path)

persons = self.\_person\_lookup.items()

for person\_id, record in persons:

measurements = self.\_get\_measurements(person\_id)

visit\_ids = set([measurement['visit\_id'] for measurement in measurements])

visits = []

for visit\_id in visit\_ids:

visit = self.\_get\_visit(visit\_id)

visit['measurements'] = [

measurement for measurement in measurements

if visit\_id == measurement['visit\_id']

]

visits.append(visit)

record['visits'] = visits

self.\_db.insert(record)

Code for importing from Github

src\_data\_path = '../../../data/processed/openflights/routes.jsonl.gz'   
with gzip.open(src\_data\_path, 'rb') as f:   
records = [json.loads(line) for line in f.readlines()]