Assignment 3

json schema Reference https://json-schema.org/learn/getting-started-step-by-step.html (https://json-schema.org/learn/getting-started-step-by-step.html)

json schema library reference: https://python-jsonschema.readthedocs.io/en/stable/ (https://python-jsonschema.readthedocs.io/en/stable/)

Avro is a language-neutral data serialization system. It can be processed by many languages (currently C, C++, C#, Java, Python, and Ruby). Avro creates binary structured format that is both compressible and splittable. Hence it can be efficiently used as the input to Hadoop MapReduce jobs.

reference: https://www.tutorialspoint.com/avro/avro_overview.htm (https://www.tutorialspoint.com/avro/avro_overview.htm)

FastAvro Schema: https://fastavro.readthedocs.io/en/latest/schema.html (https://fastavro.readthedocs.io/en/latest/schema.html)

FastAvro Home: https://fastavro.readthedocs.io/en/latest/ (https://fastavro.readthedocs.io/en/latest/)

Apache Parquet Gzip Reference: https://docs.python.org/3/library/gzip.html (https://docs.python.org/3/library/gzip.html)

Apache Parquet Load from JSONL files https://arrow.apache.org/docs/python/json.html (https://arrow.apache.org/docs/python/json.html)

Apache Parquet, Read/Write parquet tables https://arrow.apache.org/docs/python/parquet.html (https://arrow.apache.org/docs/python/parquet.html)

Python Check if File Exists: https://www.pythontutorial.net/python-basics/python-check-if-file-exists/ (https://www.pythontutorial.net/python-basics/python-check-if-file-exists/)

Google Protocol Buffers: https://developers.google.com/protocol-buffers/docs/pythontutorial (https://developers.google.com/protocol-buffers/docs/pythontutorial)

note: use .CopyFrom() when assigning a metaclass to a key.

gzip encoding example: https://gist.github.com/LouisAmon/4bd79b8ab80d3851601f3f9016300ac4 (https://gist.github.com/LouisAmon/4bd79b8ab80d3851601f3f9016300ac4)

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
#//*** Note: install with: pip install python-snappy
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)

validation_csv_path = results_dir.joinpath("json_schema_validation.cs

def read_jsonl_data():
    src_data_path = 'routes.jsonl.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 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 [2]:
```
records = read_jsonl_data()
```

Remove Record Keys with None Data types

Some of the SRC and DST airports are Empty/None. This causes issues with the Schema Validation. It's fine to have errors for this assignment, but might as well address the issue before validating the schema. Probably could have set the schema to optional as well.

In [3]:
```
#//*** Remove Records with None Type
for record in records:
    empty_keys = []

    for key,value in record.items():
        if value is None:
            empty_keys.append(key)
    for empty_key in empty_keys:
        del record[empty_key]
```

Programatically Build JSON Schema

Parsing each value of a record to generate a schema probably didn't save me any time. But it might be helpful for future tasks. Rolling through programatically means I won't miss any keys and my typos will be consistent.

In [4]:

```
def type_as_string(val):
    if isinstance(val,int):
        return "integer"

    if isinstance(val,str):
        return "string"

    if isinstance(val,bool):
        return "boolean"

    if isinstance(val,float):
        return "number"

    if isinstance(val,list):
        return "array"

    return type(val)

#//*** Pick an index to build the schema
dex=0

out = ''
out+= '{'
#out+='\n\t'
#out += '"$schema": "https://json-schema.org/draft/2020-12/schema",'
out+='\n\t'
out += '"$id": "http://json-schema.org/draft-07/schema#",'
out+='\n\t'
out += '"$schema": "http://json-schema.org/draft-07/schema#",'
out+='\n\t'
out += '"title": "Product",'
out+='\n\t'
out += '"description": "Some Product",'
out+='\n\t'
out += '"type": "object",'
out+='\n\t'
out += '"properties": {'
#out+='\n\t\t'
for key in records[dex].keys():
    #print(key, isinstance(records[0][key],dict) )

    if isinstance(records[dex][key],dict):
        out+='\n\t\t'
        #print(key)
        out += f'"{key}" : '
        out += "{"
        out+='\n\t\t\t'
        out+= '"type": "object",'
```

```
        out+='\n\t\t\t'
        out += '"properties": {'
        out+='\n\t\t\t\t'
        for key2,value2 in records[dex][key].items():
            out += f'"{key2}" : '
            out += "{"
            out+='\n\t\t\t\t\t'
            if key2 == 'active':
                out += f'"type": "boolean"'
            else:
                out += f'"type": "{type_as_string(value2)}"'
            out+='\n\t\t\t\t'
            out += "},"
            if key == "airline" and key2 == "active":
                out = out[:-1]
                continue
            if key == "src_airport" and key2 == "source":
                out = out[:-1]
                continue

            if key == "dst_airport" and key2 == "source":
                out = out[:-1]
                continue

            out+='\n\t\t\t\t'
        out+='\n\t\t\t'
        out+="}"
        out+='\n\t\t'
        out += "},"
        continue

    out+='\n\t\t'
    out += f'"{key}" : '
    out += "{"
    out+='\n\t\t\t'

    if key == 'codeshare':
        out += f'"type": "boolean"'
    else:
        out += f'"type": "{type_as_string(records[dex][key])}"'
    out+='\n\t\t'
    out+="},"

out = out[:-1]
out+='\n\t'
out+= "}"
out+='\n'
out+= "}"
print(out)

#//*** Write Schema to File
schema_path = schema_dir.joinpath('routes-schema.json')
with open(schema_path, 'w') as f:
```

```
{
        "$id": "http://json-schema.org/draft-07/schema#",
```

```
"$schema": "http://json-schema.org/draft-07/schema#",
"title": "Product",
"description": "Some Product",
"type": "object",
"properties": {
        "airline" : {
                "type": "object",
                "properties": {
                        "airline_id" : {
                                "type": "integer"
                        },
                        "name" : {
                                "type": "string"
                        },
                        "alias" : {
                                "type": "string"
                        },
```

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(validation_csv_path, 'w') as f:
        for i, record in enumerate(records):
            try:
                jsonschema.validate(record,schema)
                f.write(f"{i},valid,,\n")

            except ValidationError as e:
                ## Print message if invalid record
                print("Schema/Record Error Record:",i,"\n\n",e,"\n\n"

#//*** All Data is Schema valid
```

3.1.b Avro

In [8]:

```
#//*** Reload Clean Records
records = read_jsonl_data()


from fastavro import parse_schema
from fastavro import writer

def create_avro_dataset(records):
    schema_path = schema_dir.joinpath('routes.avsc')
    data_path = results_dir.joinpath('routes.avro')

    #//*** Load Avro Schema
    with open(schema_path, 'r') as f:
```

```
        avro_schema = json.loads(f.read())

    #//*** Parse Avro Schema
    parsed_schema = parse_schema(avro_schema)

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

```

3.1.c Parquet

Gzip Reference: https://docs.python.org/3/library/gzip.html (https://docs.python.org/3/library/gzip.html)

Apache Parquet Load from JSONL files https://arrow.apache.org/docs/python/json.html (https://arrow.apache.org/docs/python/json.html)

Apache Parquet, Read/Write parquet tables https://arrow.apache.org/docs/python/parquet.html (https://arrow.apache.org/docs/python/parquet.html)

Python Check if File Exists: https://www.pythontutorial.net/python-basics/python-check-if-file-exists/ (https://www.pythontutorial.net/python-basics/python-check-if-file-exists/)

In [9]:

```
def create_parquet_dataset():
    from pyarrow import json
    import os

    parquet_output_path = results_dir.joinpath('routes.parquet')

    #//*** PyArrow supports native JSONL files
    #//*** Extract the compressed JSONL files to disk
    src_data_path = 'routes.jsonl.gz'
    jsonl_path = 'routes.jsonl'

    #//*** Open the compressed file
    with open(src_data_path, 'rb') as f:

        #//*** Open the extracted file for writing
        with open(jsonl_path,'wb') as writer:

            #//*** Write the decompressed jsonl file
            writer.write(gzip.decompress(f.read()))

    #//*** Load the jsonl file into a parquet table
    parquet_table = json.read_json(jsonl_path)

    #//*** Delete the Extracted File
    if os.path.exists(jsonl_path):
        os.remove(jsonl_path)

    #//*** Print the First 5000 characters of the string output of pa
    print(str(parquet_table)[:5000])

```

```
    #//*** Write Parquet Table to disk
    pq.write_table(parquet_table,parquet_output_path)


```

```
pyarrow.Table
airline: struct<airline_id: int64, name: string, alias: string, iata:
string, icao: string, callsign: string, country: string, active: bool>
  child 0, airline_id: int64
  child 1, name: string
  child 2, alias: string
  child 3, iata: string
  child 4, icao: string
  child 5, callsign: string
  child 6, country: string
  child 7, active: bool
src_airport: struct<airport_id: int64, name: string, city: string, cou
ntry: string, iata: string, icao: string, latitude: double, longitude:
double, altitude: int64, timezone: double, dst: string, tz_id: string,
type: string, source: string>
  child 0, airport_id: int64
  child 1, name: string
  child 2, city: string
  child 3, country: string
```

3.1.d Protocol Buffers

In [10]:
```
#//*** Reload Clean Records Data
```

In [87]:
```
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 key exists, load value into obj
    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')

    if 'active' in airline.keys():
        obj.active = airline['active']

    return obj


def create_protobuf_dataset(records):
    routes = routes_pb2.Routes()


    for record in records:

        #//*** Add a Record to routes
        route = routes.route.add()

```

```
        if 'codeshare' in record.keys():
            route.codeshare = record['codeshare']

        if record.get('stops'):
            route.stops = record.get('stops')

        #//*** Use extend to add lists/arrays.
        if record.get('equipment'):
            route.equipment.extend(record.get('equipment'))

        #//*** generate Airline Object
        if 'src_airport' in record.keys():

            #//*** If src_airport exists Build Object from record
            src_airport = _airport_to_proto_obj(record['src_airport']

            #//*** Skip if None
            if src_airport is not None:
                #//*** Use CopyFrom to assign objects
                route.src_airport.CopyFrom( src_airport )

        if 'dst_airport' in record.keys():


            #//*** If dst_airport exists Build Object from record
            dst_airport = _airport_to_proto_obj(record['dst_airport']

            #//*** Skip if None
            if dst_airport is not None:
                #//*** Use CopyFrom to assign objects
                route.dst_airport.CopyFrom(dst_airport)

        if 'airline' in record.keys():
            #//*** If airline exists Build Object from record
            airline = _airline_to_proto_obj(record['airline'])

            #//*** Skip if None
            if airline is not None:
                #//*** Use CopyFrom to assign objects
                route.airline.CopyFrom(airline)

    #//*** Display the first 10,000 characters of routes
    print(str(routes)[:10000])

    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)
```

```
route {
  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"
  iata: "AER"
  icao: "URSS"
  latitude: 43.449902
  longitude: 39.9566
```

3.1e Size Comparison

In [154]:

```
src_data_path = 'routes.jsonl.gz'

#//*** Get Filesize of Compressed JSON
json_compressed_file_size = os.path.getsize(src_data_path)


jsonl_path = 'routes.jsonl'
#//*** Open the compressed file
with open(src_data_path, 'rb') as f:
    #//*** Open the extracted file for writing
    with open(jsonl_path,'wb') as writer:
        #//*** Write the decompressed jsonl file
        writer.write(gzip.decompress(f.read()))

#//*** Get Filesize of Compressed JSON
json_uncompressed_file_size = os.path.getsize(jsonl_path)

#//*** Delete the Extracted File
if os.path.exists(jsonl_path):
    os.remove(jsonl_path)

#//*** Get Avro File size
avro_path = results_dir.joinpath('routes.avro')
avro_file_size = os.path.getsize(avro_path)

parquet_output_path = results_dir.joinpath('routes.parquet')
parquet_file_size = os.path.getsize(parquet_output_path)

pb_path = results_dir.joinpath('routes.pb')
pb_file_size = os.path.getsize(pb_path)

compressed_pb_path = results_dir.joinpath('routes.pb.snappy')
compressed_pb_file_size = os.path.getsize(compressed_pb_path)

```

```
print("JSON Compressed File Size:                      ",format(json_c
print("JSON Uncompressed File Size:                    ",format(json_u
print("Avro Encoded  File Size:                        ",format(avro_f
print("Parquet Encoded File Size:                      ",format(parque
print("Protocol Buffer File Size:                      ",format(pb_fil
print("Protocol Buffer (Snappy Compressed) File Size: ",format(compre

out = ""
out += f"JSON,compressed,{json_compressed_file_size}\n"
out += f"JSON,uncompressed,{json_uncompressed_file_size}\n"
out += f"Avro,uncompressed,{avro_file_size}\n"
out += f"Parquet,uncompressed,{parquet_file_size}\n"
out += f"Protocol Buffer,uncompressed,{pb_file_size}\n"
out += f"Protocol Buffer,compressed,{compressed_pb_file_size}\n"

comparision_path = results_dir.joinpath('comparison.csv')
print("=================================================================
print("Writing Results to ./results/comparison.csv")

with open(comparision_path,'w') as writer:
    writer.write(out)


```

```
JSON Compressed File Size:                      3,327,145  bytes
JSON Uncompressed File Size:                    59,109,449  bytes
Avro Encoded  File Size:                        19,646,227  bytes
Parquet Encoded File Size:                      1,975,465  bytes
Protocol Buffer File Size:                      22,523,154  bytes
Protocol Buffer (Snappy Compressed) File Size:  3,762,689  bytes
=================================================================
====
Writing Results to ./results/comparison.csv
```

3.2

3.2.a Simple Geohash Index

In [156]:
```
#//*** Load a clean copy of records
```

In [88]:
```
#//*** Reimport json lest we get confused with parquet json loader
import json

#//*****************************************************************
#//*** Crawl each record and generate a dictionary that maps the fold
#//*** Parse the dictionary map to generate the needed files and fold
#//*****************************************************************
def create_hash_dirs(records):
    geoindex_dir = results_dir.joinpath('geoindex')
    geoindex_dir.mkdir(exist_ok=True, parents=True)
    hashes = []

    airport_hash_dict = {}
```

```
    airport_hashmap = {}

    #//*** Generate geohashes for each airport in src_destinations
    #//*** geohashes are stored in hashes for sorting
    #//** and airport_hash_dict to associate the name with the geohas
    for record in records:

        if record['src_airport'] is None:
            continue

        airport_geohash = pygeohash.encode(record['src_airport']['lat
        if airport_geohash not in airport_hash_dict.keys():
            #print(record['src_airport']['name']," - ",airport_geohas

            #//*** Add to hashes if airport is unique.
            #hashes.append(airport_geohash)

            #//*** Add Airport_geohash to dictionary.
            #//*** This assiociates the airport name with the geohash
            airport_hash_dict[airport_geohash] = record['src_airport'

            #//*** add the geohash to the airport_hashmap dictionary
            key1 = airport_geohash[:1]
            key2 = airport_geohash[:2]
            key3 = airport_geohash[:3]

            #//*** Initialize Keys as needed
            if key1 not in airport_hashmap.keys():
                airport_hashmap[key1] = {}

            if key2 not in airport_hashmap[key1].keys():
                airport_hashmap[key1][key2] = {}

            if key3 not in airport_hashmap[key1][key2].keys():
                airport_hashmap[key1][key2][key3] = {}

            #//*** Associate the whole Airport record with the Geohas
            #//*** We'll keep everything together. We could also just
            #//*** the airport info in a separate dictionary/database
            airport_hashmap[key1][key2][key3][airport_geohash] = reco

    #//****************************
    #//*** END record in records
    #//****************************

    #//*****************************************************
    #//*** Parse hashmap, Build directories and json files
    #//*****************************************************
    #//*** Top Level - Level 1
    for key1,values1 in airport_hashmap.items():

        #//*** Build Level1 Folders as needed
        level1_path = geoindex_dir.joinpath(key1)
        level1_path.mkdir(exist_ok=True, parents=True)

        print(key1,level1_path)
```

```
70
71          #//*** Loop through Level2 sub folders
72          for key2,values2 in airport_hashmap[key1].items():
73
74              #//*** Build Level2 Folders as needed
75              level2_path = level1_path.joinpath(key2)
76              level2_path.mkdir(exist_ok=True, parents=True)
77
78              #//*** Only Print Top 2 Levels for display
79              print("--",key2,level2_path)
80
81              #//*** Loop through Level3 - File Level
82              for key3,values3 in airport_hashmap[key1][key2].items():
83
84                  filepath = level2_path.joinpath(f"{key3}.json.gz")
85                  #//*** Generate JSON String
86                  json_data = json.dumps(values3)
87
88                  #//*** Encode JSON data as bytes
89                  encoded = json_data.encode('utf-8')
90
91                  #//*** Compress encoded file and write to disk
92                  with open(filepath,'wb') as f:
93                      f.write(gzip.compress(encoded))
94
95  airport_hashmap = create_hash_dirs(records)
```

```
s C:\Users\family\DSCProjects\DSC\DSC650\assignment03\results\geoinde
x\s
-- sz C:\Users\family\DSCProjects\DSC\DSC650\assignment03\results\geoi
ndex\s\sz
-- s1 C:\Users\family\DSCProjects\DSC\DSC650\assignment03\results\geoi
ndex\s\s1
-- s4 C:\Users\family\DSCProjects\DSC\DSC650\assignment03\results\geoi
ndex\s\s4
-- sr C:\Users\family\DSCProjects\DSC\DSC650\assignment03\results\geoi
ndex\s\sr
-- sw C:\Users\family\DSCProjects\DSC\DSC650\assignment03\results\geoi
ndex\s\sw
-- su C:\Users\family\DSCProjects\DSC\DSC650\assignment03\results\geoi
ndex\s\su
-- sp C:\Users\family\DSCProjects\DSC\DSC650\assignment03\results\geoi
ndex\s\sp
-- s0 C:\Users\family\DSCProjects\DSC\DSC650\assignment03\results\geoi
ndex\s\s0
-- sf C:\Users\family\DSCProjects\DSC\DSC650\assignment03\results\geoi
```

3.2.b Simple Search Feature

In [130]:

```
1  def airport_search(latitude, longitude,distance=150):
2
3      #//*** Distance in kilometers
4
5      geoindex_dir = results_dir.joinpath('geoindex')
6
7      tgt_geohash = pygeohash.encode(latitude, longitude)
```

```
 8
 9     level1 = tgt_geohash[:1]
10     level2 = tgt_geohash[:2]
11
12     folderpath = geoindex_dir.joinpath(level1).joinpath(level2)
13
14     #//*** If Filepath doesn't exist, blame the user!
15     if os.path.exists(folderpath) == False:
16         print("There are no airports near these coordinates")
17         return
18
19     #//*** Get a list of files in the folder path
20     files = os.listdir(folderpath)
21
22     airport_dict = {}
23
24     airports_in_range = []
25
26     #//*** Load all airports into a dictionary
27     for file in files:
28
29         filepath = folderpath.joinpath(file)
30
31         #//*** Decode compressed JSON file.
32         with gzip.open(filepath,'rb') as f:
33
34             #//*** Open each file and add to dictionary
35             for key,value in json.loads(f.read().decode()).items():
36                 airport_dict[key] = value
37
38                 #//*** Find the Distance and add to dist_dict.
39                 #//*** Makes it easier to search by distance
40                 airport_distance = int(pygeohash.geohash_approximate_
41
42                 #//*** If Airport_Distance is less than Target distan
43                 if airport_distance <= distance:
44                     airports_in_range.append(key)
45
46
47
48
49
50     #for key,value in dist_dict.items():
51     #    print(tgt_geohash,key,value,airport_dict[key]['name'],airpor
52
53     print(f"There are {len(airports_in_range)} airports within {dista
54     for index, dst_geohash in enumerate(airports_in_range):
55         print(f"{index+1}.) {int(pygeohash.geohash_approximate_distan
56
57
58     print()
59
60
61
62
63 airport_search(41.1499988, -95.91779)
```

```
64
65 print("Search Near the San Francisco Bay Area")
66 airport_search(37.59592,-122.01375)
67
68 print("Search airports near Berlin, DE")
69 airport_search(52.52246,13.40457)
70
71 print("Search airports near Dubai, UAE")
72 airport_search(25.19721,55.26848)
73
```

```
There are 2 airports within 150km of (41.1499988, -95.91779)
1.) 19km Eppley Airfield, Omaha Region: America/Chicago
2.) 123km Lincoln Airport, Lincoln Region: America/Chicago

Search Near the San Francisco Bay Area
There are 5 airports within 150km of (37.59592, -122.01375)
1.) 123km Monterey Peninsula Airport, Monterey Region: America/Los_Ang
eles
2.) 123km Metropolitan Oakland International Airport, Oakland Region:
America/Los_Angeles
3.) 123km Norman Y. Mineta San Jose International Airport, San Jose Re
gion: America/Los_Angeles
4.) 123km Stockton Metropolitan Airport, Stockton Region: America/Los_
Angeles
5.) 123km Modesto City Co-Harry Sham Field, Modesto Region: America/Lo
s_Angeles

Search airports near Berlin, DE
There are 2 airports within 150km of (52.52246, 13.40457)
1.) 123km Berlin-Tegel Airport, Berlin Region: Europe/Berlin
2.) 123km Berlin-Schönefeld Airport, Berlin Region: Europe/Berlin

Search airports near Dubai, UAE
There are 3 airports within 150km of (25.19721, 55.26848)
1.) 123km Al Maktoum International Airport, Dubai Region: Asia/Dubai
2.) 19km Dubai International Airport, Dubai Region: Asia/Dubai
3.) 123km Al Ain International Airport, Al Ain Region: Asia/Dubai
```