Assignment 7-1c

DSC 650

Jake Meyer

04/29/2023

Finally, we will simulate multiple geographically distributed data centers. For this example, we will assume we have three data centers located in the western, central, and eastern United States. Google lists the locations of their data centers and we will use the following locations for our three data centers. West The Dalles, Oregon Latitude: 45.5945645 Longitude: -121.1786823 Central Papillion, NE Latitude: 41.1544433 Longitude: -96.0422378 East Loudoun County, Virginia Latitude: 39.08344 Longitude: -77.6497145 Assume that you have an application that provides routes for each of the source airports and you want to store routes in the data center closest to the source airport. The output folders should look as follows. geo ├── location=central ├── location=east └── location=west

In [4]:
```
## Import the necessary packages for the assignment.
import pandas as pd
import pyarrow as pa
import pyarrow.parquet as parq
## import pathlib
from pathlib import Path
import pygeohash as pgh
```

In [8]:
```
## Print versions of essential packages
print("pandas version: {}".format(pd.__version__))
print("pyarrow version: {}".format(pa.__version__))
```

```
pandas version: 1.5.3
pyarrow version: 11.0.0
```

In [9]:
```
## Setup directories
cwd = Path('C:/Users/jkmey/Documents/Github/DSC650_Course_Assignments/dsc650/dsc650
results_dir = cwd.joinpath('results')
pq_file = results_dir.joinpath('routes.parquet')
partitioned_pq_file = results_dir.joinpath('geo')
```

Load the dataset using read_parquet

In [10]:
```
## Use read_parquet() to read routes.parquet
pq = pd.read_parquet(pq_file, engine = 'fastparquet')
```

In [11]:
```
print(list(pq.columns.values))
```

```
['codeshare', 'equipment', 'airline.active', 'airline.airline_id', 'airline.alia
s', 'airline.callsign', 'airline.country', 'airline.iata', 'airline.icao', 'airlin
e.name', 'src_airport.airport_id', 'src_airport.altitude', 'src_airport.city', 'sr
c_airport.country', 'src_airport.dst', 'src_airport.iata', 'src_airport.icao', 'sr
c_airport.latitude', 'src_airport.longitude', 'src_airport.name', 'src_airport.sou
rce', 'src_airport.timezone', 'src_airport.type', 'src_airport.tz_id', 'dst_airpor
t.airport_id', 'dst_airport.altitude', 'dst_airport.city', 'dst_airport.country',
'dst_airport.dst', 'dst_airport.iata', 'dst_airport.icao', 'dst_airport.latitude',
'dst_airport.longitude', 'dst_airport.name', 'dst_airport.source', 'dst_airport.ti
mezone', 'dst_airport.type', 'dst_airport.tz_id']
```

Define the Data Centers

In [16]:
```
## Define the West Data Center as specified in the assignment instructions.
## Latitude: 45.5945645, Longitude: -121.1786823
west_center = pgh.encode(45.5945645, -121.1786823, precision = 15)
print(west_center)
```

```
c21g6s0rs4c7qht
```

In [17]:
```
## Define the Central Data Center as specified in the assignment instructions.
## Latitude: 41.1544433, Longitude:  -96.0422378
central_center = pgh.encode(41.1544433,  -96.0422378, precision = 15)
print(central_center)
```

```
9z7dnebnj8kbwrc
```

In [18]:
```
## Define the East Data Center as specified in the assignment instructions.
## Latitude: 39.08344, Longitude: -77.6497145
east_center = pgh.encode(39.08344,  -77.6497145, precision = 15)
print(east_center)
```

```
dqby34cjw922fem
```

Create Function to Return Data Center

In [24]:
```
## Create a function that will take latitude, longitude, 3 centers as an argument.
## Re-use previous function from 7.1a for returning center or Does Not Exist statem
## Function will return the closest data center.

def get_data_center(latit, longit, west_center, central_center, east_center):

    ## Insert function for returning center or "Does Not Exist"
    def get_center(val):
        for key, value in p_dict.items():
            if val == value:
                return key
        return "Does Not Exist"

    value_list = []
    locations = ['west', 'central', 'east']
    srcgeoval = pygeohash.encode(latit, longit)
    ## Obtain distance from west center
    dist_west_meters = pgh.geohash_approximate_distance(srcgeoval, west_center) / 1
    value_list.append(dist_west_meters)
    ## Obtain distance from central center
    dist_central_meters = pgh.geohash_approximate_distance(srcgeoval, central_cente
    value_list.append(dist_central_meters)
    ## Obtain distance from east center
    dist_east_meters = pgh.geohash_approximate_distance(srcgeoval, east_center) / 1
    value_list.append(dist_east_meters)
    ## Create a dictionary of locations and values.
    p_dict = dict(zip(locations, value_list))
    ## Return the closest center
    shortest_distance = min(value_list, key = float)
    center = get_center(shortest_distance)
    return center
```

Create Columns for key, latit, longit, and location.

In [25]:
```
## Create the concatenated key with src_airport.iata + dst_airport.iata+ airline.ic
pq['key'] = pq['src_airport.iata'] + pq['dst_airport.iata'] + pq['airline.icao']
```

In [26]:
```
## Create latit and longit columns.
pq['latit'] = pq['src_airport.latitude']
pq['longit'] = pq['src_airport.longitude']
```

In [28]:
```
## Create a Column for data center location.
pq['location'] = pq.apply(lambda x: get_data_center(x.latit, x.longit, west_center,
```

Create Table

In [29]:
```
## Create the table with pyarrow.
table = pa.Table.from_pandas(pq)
```

Use Parquet Write_to_Dataset

In [30]:
```
## Use write_to_dataset to generate the directory.
parq.write_to_dataset(table, root_path = partitioned_pq_file, partition_cols = ['lo
```

Show the Table in Notebook

In [31]:
```
## Use read_table() function on the partitioned file
partitioned_table = parq.read_table(partitioned_pq_file)
print(partitioned_table)
```

```
pyarrow.Table
codeshare: bool
equipment: list<item: string>
  child 0, item: string
airline.active: bool
airline.airline_id: int64
airline.alias: string
airline.callsign: string
airline.country: string
airline.iata: string
airline.icao: string
airline.name: string
src_airport.airport_id: double
src_airport.altitude: double
src_airport.city: string
src_airport.country: string
src_airport.dst: string
src_airport.iata: string
src_airport.icao: string
src_airport.latitude: double
src_airport.longitude: double
src_airport.name: string
src_airport.source: string
src_airport.timezone: double
src_airport.type: string
src_airport.tz_id: string
dst_airport.airport_id: double
dst_airport.altitude: double
dst_airport.city: string
dst_airport.country: string
dst_airport.dst: string
dst_airport.iata: string
dst_airport.icao: string
dst_airport.latitude: double
dst_airport.longitude: double
dst_airport.name: string
dst_airport.source: string
dst_airport.timezone: double
dst_airport.type: string
dst_airport.tz_id: string
key: string
latit: double
longit: double
location: dictionary<values=string, indices=int32, ordered=0>
----
codeshare: [[false,false,false,false,false,...,true,true,false,false,true],[true,t
rue,true,true,false,...,false,false,false,false,false],...,[true,true,true,true,tr
ue,...,false,false,false,false,false],[false,false,false,false,false,...,false,fal
se,false,false,false]]
equipment: [[["CNC"],["CNC"],...,["345","346"],["738"]],[["738"],["744"],...,["73
H","73W"],["73W"]],...,[["CR2"],["320"],...,["738"],["738"]],[["738"],["738"],...,
["734"],["734"]]]
airline.active: [[true,true,true,true,true,...,true,true,true,true,true],[true,tru
e,true,true,true,...,true,true,true,true,true],...,[true,true,true,true,true,...,t
rue,true,true,true,true],[true,true,true,true,true,...,true,true,true,true,true]]
airline.airline_id: [[10739,10739,10739,10739,10739,...,2822,2822,2822,2822,2822],
[2822,2822,2822,2822,4867,...,5416,5416,5416,5416,5416],...,[2822,2822,2822,2822,2
822,...,4573,4573,4573,4573,4573],[4573,4573,4573,4573,4573,...,4178,19016,19016,1
9016,19016]]
airline.alias: [["nan","nan","nan","nan","nan",...,"Horizon Airlines","Horizon Air
lines","Horizon Airlines","Horizon Airlines","Horizon Airlines"],["Horizon Airline
s","Horizon Airlines","Horizon Airlines","Horizon Airlines","nan",...,"Varig","Var
ig","Varig","Varig","Varig"],...,["Horizon Airlines","Horizon Airlines","Horizon A
irlines","Horizon Airlines","Horizon Airlines",...,"Swiss European","Swiss Europea
n","Swiss European","Swiss European","Swiss European"],["Swiss European","Swiss Eu
ropean","Swiss European","Swiss European","Swiss European",...,"Qantas Airways","A
pache","Apache","Apache","Apache"]]
airline.callsign: [["nan","nan","nan","nan","nan",...,"IBERIA","IBERIA","IBERI
A","IBERIA","IBERIA"],["IBERIA","IBERIA","IBERIA","IBERIA","TAM",...,"WESTJET","WE
STJET","WESTJET","WESTJET","WESTJET"],...,["IBERIA","IBERIA","IBERIA","IBERIA","IB
ERIA",...,"SUNEXPRESS","SUNEXPRESS","SUNEXPRESS","SUNEXPRESS","SUNEXPRESS"],["SUNE
XPRESS","SUNEXPRESS","SUNEXPRESS","SUNEXPRESS","SUNEXPRESS",...,"REX","APACHE","AP
ACHE","APACHE","APACHE"]]
airline.country: [["United States","United States","United States","United State
s","United States",...,"Spain","Spain","Spain","Spain","Spain"],["Spain","Spai
n","Spain","Spain","Brazil",...,"Canada","Canada","Canada","Canada","Canada"],...,
["Spain","Spain","Spain","Spain","Spain",...,"Turkey","Turkey","Turkey","Turke
y","Turkey"],["Turkey","Turkey","Turkey","Turkey","Turkey",...,"Australia","United
States","United States","United States","United States"]]
airline.iata: [["3E","3E","3E","3E","3E",...,"IB","IB","IB","IB","IB"],["IB","I
B","IB","IB","JJ",...,"WS","WS","WS","WS","WS"],...,["IB","IB","IB","IB","I
B",...,"XQ","XQ","XQ","XQ","XQ"],["XQ","XQ","XQ","XQ","XQ",...,"ZL","ZM","ZM","Z
M","ZM"]]
airline.icao: [["WE1","WE1","WE1","WE1","WE1",...,"IBE","IBE","IBE","IBE","IBE"],
["IBE","IBE","IBE","IBE","TAM",...,"WJA","WJA","WJA","WJA","WJA"],...,["IBE","IB
E","IBE","IBE","IBE",...,"SXS","SXS","SXS","SXS","SXS"],["SXS","SXS","SXS","SX
S","SXS",...,"RXA","IWA","IWA","IWA","IWA"]]
airline.name: [["Air Choice One","Air Choice One","Air Choice One","Air Choice On
e","Air Choice One",...,"Iberia Airlines","Iberia Airlines","Iberia Airlines","Ibe
ria Airlines","Iberia Airlines"],["Iberia Airlines","Iberia Airlines","Iberia Airl
ines","Iberia Airlines","TAM Brazilian Airlines",...,"WestJet","WestJet","WestJe
t","WestJet","WestJet"],...,["Iberia Airlines","Iberia Airlines","Iberia Airline
s","Iberia Airlines","Iberia Airlines",...,"SunExpress","SunExpress","SunExpres
s","SunExpress","SunExpress"],["SunExpress","SunExpress","SunExpress","SunExpres
s","SunExpress",...,"Regional Express","Apache Air","Apache Air","Apache Air","Apa
che Air"]]
...
```