Assignment 7-1a

DSC 650

Jake Meyer

04/28/2023

Start by loading the dataset from the previous assignment using Pandas's read_parquet method. Next, add the concatenated key then using Panda's apply method to create a new column called key. For this part of the example, we will create 16 partitions so that we can compare it to the partitions we create from hashed keys in the next part of the assignment. The partitions are determined by the first letter of the composite key using the following partitions. partitions = (('A', 'A'), ('B', 'B'), ('C', 'D'), ('E', 'F'), ('G', 'H'), ('I', 'J'), ('K', 'L'), ('M', 'M'), ('N', 'N'), ('O', 'P'), ('Q', 'R'), ('S', 'T'), ('U', 'U'), ('V', 'V'), ('W', 'X'), ('Y', 'Z')) In this case ('A', 'A') means the folder should contain all of the routes whose composite key starts with A. Similarly, ('E', 'F') should contain routes whose composite key starts with E or F. The results/kv directory should contain the following folders. kv ├── kv_key=A ├── kv_key=B ├── kv_key=C-D ├── kv_key=E-F ├── kv_key=G-H ├── kv_key=I-J ├── kv_key=K-L ├── kv_key=M ├── kv_key=N ├── kv_key=O-P ├── kv_key=Q-R ├── kv_key=S-T ├── kv_key=U ├── kv_key=V ├── kv_key=W-X └── kv_key=Y-Z An easy way to create this directory structure is to create a new key called kv_key from the key column and use the to_parquet method with partition_cols= ['kv_key'] to save a partitioned dataset.

In [39]:
```
## 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
```

In [40]:
```
## 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 [41]:
```
## 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('kv')
```

Load the dataset using read_parquet

In [42]:
```
## Use read_parquet() to read routes.parquet
pq = pd.read_parquet(pq_file, engine = 'fastparquet')
```

In [43]:
```
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']
```

Add Concatenated Key and Use Apply Method for New Column Key

In [44]:
```
## Code provided in the assignment instructions.
partitions = (
    ('A', 'A'), ('B', 'B'), ('C', 'D'), ('E', 'F'),
    ('G', 'H'), ('I', 'J'), ('K', 'L'), ('M', 'M'),
    ('N', 'N'), ('O', 'P'), ('Q', 'R'), ('S', 'T'),
    ('U', 'U'), ('V', 'V'), ('W', 'X'), ('Y', 'Z'))
```

In [45]:
```
## Create partition value keys as specified for the assignment.
partition_value_keys = ('A', 'B', 'C-D', 'E-F', 'G-H', 'I-J', 'K-L', 'M', 'N', 'O-P
                        'V', 'W-X', 'Y-Z')
```

In [46]:
```
## Create dictionary of partition_value_keys and partitions.
p_dict = dict(zip(partition_value_keys, partitions))
print(p_dict)
```

```
{'A': ('A', 'A'), 'B': ('B', 'B'), 'C-D': ('C', 'D'), 'E-F': ('E', 'F'), 'G-H':
('G', 'H'), 'I-J': ('I', 'J'), 'K-L': ('K', 'L'), 'M': ('M', 'M'), 'N': ('N',
'N'), 'O-P': ('O', 'P'), 'Q-R': ('Q', 'R'), 'S-T': ('S', 'T'), 'U': ('U', 'U'),
'V': ('V', 'V'), 'W-X': ('W', 'X'), 'Y-Z': ('Y', 'Z')}
```

In [47]:
```
## 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 [48]:
```
## Only keep the first two two values as specified in the assignment.
pq['partitioned_value'] = pq['key'].str[:1]
```

In [49]:
```
## Create a function to get key or return non-exist message.
def get_key(val):
    for key, value in p_dict.items():
        if val in value:
            return key
    return "Does Not Exist"
```

In [50]:
```
## Use the apply() method as specified with a lambda function for generating kv_key
pq['kv_key'] = pq.apply(lambda x: get_key(x.partitioned_value), axis = 1)
```

In [51]:
```
## Remove the keys that don't exist.
pq = pq[pq.kv_key != "Does Not Exist"]
```

Create Table

In [52]:
```
## Create the table with pyarrow.
table = pa.Table.from_pandas(pq)
```

Use Parquet Write_to_Dataset

In [53]:
```
## Use write_to_dataset to generate the directory.
parq.write_to_dataset(table, root_path = partitioned_pq_file, partition_cols = ['kv
```

Show the Table in Notebook

In [54]:
```
## 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
partitioned_value: string
__index_level_0__: int64
kv_key: dictionary<values=string, indices=int32, ordered=0>
----
codeshare: [[false,false,false,false,false,...,true,false,false,false,true],[fals
e,false,false,false,false,...,false,false,false,false,false],...,[false,false,fals
e,false,false,...,false,false,false,false,false],[true,true,false,true,true,...,fa
lse,false,false,true,true]]
equipment: [[["CR2"],["CR2"],...,["BNI"],["320"]],[["AT4"],["AN4"],...,["321","75
2"],["321","320"]],...,[["AN4"],["100"],...,["DH1"],["DH1"]],[["737"],["737"],...,
["737","320"],["737"]]]
airline.active: [[true,true,true,true,true,...,true,true,true,true,true],[false,fa
lse,true,true,true,...,true,true,true,true,true],...,[false,true,true,true,tru
e,...,true,true,true,true,true],[true,true,true,true,true,...,true,true,true,true,
true]]
```

airline.airline_id: [[410,410,410,8359,470,...,2822,2822,2896,4797,3463],[10121,29
51,2922,2922,2922,...,220,220,220,220,220],...,[2951,2922,2922,2922,2922,...,-1,-
1,-1,-1,-1],[4611,4611,4611,4611,4611,...,4611,4611,4611,4611,4611]]
airline.alias: [["ANA All Nippon Airways","ANA All Nippon Airways","ANA All Nippon
Airways","nan","ANA All Nippon Airways",...,"Horizon Airlines","Horizon Airline
s","Horizon Airlines","Swiss European","nan"],["nan","Horizon Airlines","Horizon A
irlines","Horizon Airlines","Horizon Airlines",...,"\N","\N","\N","\N","\N"],...,
["Horizon Airlines","Horizon Airlines","Horizon Airlines","Horizon Airlines","Hori
zon Airlines",...,"\N","\N","\N","\N","\N"],["Swiss European","Swiss European","Sw
iss European","Swiss European","Swiss European",...,"Swiss European","Swiss Europe
an","Swiss European","Swiss European","Swiss European"]]
airline.callsign: [["AEROCONDOR","AEROCONDOR","AEROCONDOR","nan","BURKINA",...,"IB
ERIA","IBERIA","INTERLINK","SOLOMON","MERAIR"],["nan","SCILLONIA","IRANAIR","IRANA
IR","IRANAIR",...,"BOURBON","BOURBON","BOURBON","BOURBON","BOURBON"],...,["SCILLON
IA","IRANAIR","IRANAIR","IRANAIR","IRANAIR",...,"\N","\N","\N","\N","\N"],["SHENZH
EN AIR","SHENZHEN AIR","SHENZHEN AIR","SHENZHEN AIR","SHENZHEN AIR",...,"SHENZHEN
AIR","SHENZHEN AIR","SHENZHEN AIR","SHENZHEN AIR","SHENZHEN AIR"]]
airline.country: [["Portugal","Portugal","Portugal","Peru","Burkina Faso",...,"Spa
in","Spain","South Africa","Solomon Islands","Italy"],["United States","United Kin
gdom","Iran","Iran","Iran",...,"Reunion","Reunion","Reunion","Reunion","Reunio
n"],...,["United Kingdom","Iran","Iran","Iran","Ira
n",...,"\N","\N","\N","\N","\N"],["China","China","China","China","China",...,"Chi
na","China","China","China","China"]]
airline.iata: [["2B","2B","2B","2I","2J",...,"IB","IB","ID","IE","IG"],["IL","na
n","IR","IR","IR",...,"ZB","ZB","ZB","ZB","ZB"],...,["nan","IR","IR","IR","I
R",...,"-","-","-","-","-"],["ZH","ZH","ZH","ZH","ZH",...,"ZH","ZH","ZH","ZH","Z
H"]]
airline.icao: [["ARD","ARD","ARD","FOF","VBW",...,"IBE","IBE","ITK","SOL","ISS"],
["ILW","IOS","IRA","IRA","IRA",...,"BUB","BUB","BUB","BUB","BUB"],...,["IOS","IR
A","IRA","IRA","IRA",...,"nan","nan","nan","nan","nan"],["CSZ","CSZ","CSZ","CS
Z","CSZ",...,"CSZ","CSZ","CSZ","CSZ","CSZ"]]
airline.name: [["Aerocondor","Aerocondor","Aerocondor","Star Peru (2I)","Air Burki
na",...,"Iberia Airlines","Iberia Airlines","Interlink Airlines","Solomon Airline
s","Meridiana"],["Illinois Airways","Isles of Scilly Skybus","Iran Air","Iran Ai
r","Iran Air",...,"Air Bourbon","Air Bourbon","Air Bourbon","Air Bourbon","Air Bou
rbon"],...,["Isles of Scilly Skybus","Iran Air","Iran Air","Iran Air","Iran Ai
r",...,"Unknown","Unknown","Unknown","Unknown","Unknown"],["Shenzhen Airlines","Sh
enzhen Airlines","Shenzhen Airlines","Shenzhen Airlines","Shenzhen Airline
s",...,"Shenzhen Airlines","Shenzhen Airlines","Shenzhen Airlines","Shenzhen Airli
nes","Shenzhen Airlines"]]
...