2-1

September 16, 2020

0.0.1 Week 2 Assignments

The dsc650/assignments/assignment02a folder contains skeleton code for this assignment. Provide the code to implement the functions in the run_assignment.py file. For this assignment, we will be working with the CSV data found in the data/external/tidynomicon folder. Specifically, we will be using with the measurements.csv, person.csv, site.csv, and visited.csv files. 2.1 Assignment

Complete the code in kvdb.py to implement a basic key-value database that saves its state to a pickle file. Use that code to create databases that store each of CSV files by key. The pickle files should be stored in the dsc650/assignments/assignment02b/results/kvdb/ folder.

Input File Output File Key measurements.csv measurements.pickle Composite key person.csv people.pickle person_id site.csv sites.pickle site_id visited.csv visits.pickle Composite key

The measurements.csv and visited.csv have composite keys that use multiple columns. For measurements.csv those fields are visit_id, person_id, and quantity. For visited.csv those fields are visit_id and site_id. The following is an example of code that sets and gets the value using a composite key.

kvdb_path = 'visits.pickle'

kvdb = KVDB(kvdb_path)

key = (619, 'DR-1')

value = dict(

```
visit_id=619,

site_id='DR-1',

visit_date='1927-02-08'
```

)

kvdb.set_value(key, value)

retrieved_value kvdb.get_value(key)# Retrieved should be the same as value

[3]:
```
import json
from pathlib import Path
import os

import pandas as pd
```

```
import s3fs


def read_cluster_csv(file_path, endpoint_url='https://storage.budsc.
↪midwest-datascience.com'):
    s3 = s3fs.S3FileSystem(
        anon=True,
        client_kwargs={
            'endpoint_url': endpoint_url
        }
    )
    return pd.read_csv(s3.open(file_path, mode='rb'))

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)

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')
```

[4]:
```
class KVDB(object):
    def __init__(self, db_path):
        self._db_path = Path(db_path)
        self._db = {}
        self._load_db()

    def _load_db(self):
        if self._db_path.exists():
            with open(self._db_path) as f:
                self._db = json.load(f)

    def get_value(self, key):
        return self._db.get(key)

    def set_value(self, key, value):
        self._db[key] = value

    def save(self):
        with open(self._db_path, 'w') as f:
            json.dump(self._db, f, indent=2)
```

[5]:
```
def create_sites_kvdb():
    db = KVDB(sites_json)
    df = read_cluster_csv('data/external/tidynomicon/site.csv')
```

```
    for site_id, group_df in df.groupby('site_id'):
        db.set_value(site_id, group_df.to_dict(orient='records')[0])
    db.save()

def create_people_kvdb():
    db = KVDB(people_json)
    ## TODO: Implement code
    df = read_cluster_csv('data/external/tidynomicon/person.csv')
    for person_id, group_df in df.groupby('person_id'):
        db.set_value(person_id, group_df.to_dict(orient='records')[0])
    db.save()


def create_visits_kvdb():
    db = KVDB(visited_json)
    ## TODO: Implement code
    df = read_cluster_csv('data/external/tidynomicon/visited.csv')
    for visit_id, group_df in df.groupby('visit_id'):
        db.set_value(visit_id, group_df.to_dict(orient='records')[0])
    db.save()


def create_measurements_kvdb():
    db = KVDB(measurements_json)
    df = read_cluster_csv('data/external/tidynomicon/measurements.csv')
    group_columns = ['visit_id', 'person_id', 'quantity']
    for group, group_df in df.groupby(group_columns):
        key = str(group)
        db.set_value(key, group_df.to_dict(orient='records'))
    db.save()
```

[6]:
```
create_sites_kvdb()
create_people_kvdb()
create_visits_kvdb()
create_measurements_kvdb()
```

[ ]:
```

```