In [1]:
```
import json
from pathlib import Path
import os
import pandas as pd


def read_cluster_csv(file_path):
    #file_path ='/VZW Twinsburg/Tuition assistance/Bellevue University_MSDS/D
    return pd.read_csv(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')
```

In [3]:
```
os.getcwd()
```

Out[3]:
```
'D:\\VZW Twinsburg\\Tuition assistance\\Bellevue University_MSDS\\DSC 650
\\dsc650\\dsc650\\assignments\\assignment02'
```

In [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)
```

In [7]:
```
def create_sites_kvdb():
    db = KVDB(sites_json)
    df = read_cluster_csv('D:/VZW Twinsburg/Tuition assistance/Bellevue Unive
    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)
    df = read_cluster_csv('D:/VZW Twinsburg/Tuition assistance/Bellevue Unive
    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)
    df = read_cluster_csv('D:/VZW Twinsburg/Tuition assistance/Bellevue Unive
    for key, group_df in df.groupby(['visit_id', 'site_id']):
        db.set_value(str(key), group_df.to_dict(orient='records')[0])
    db.save()


def create_measurements_kvdb():
    db = KVDB(measurements_json)
    df = read_cluster_csv('D:/VZW Twinsburg/Tuition assistance/Bellevue Unive
    for key, group_df in df.groupby(['person_id', 'visit_id', 'quantity']):
        db.set_value(str(key), group_df.to_dict(orient='records')[0])
    db.save()
```

In [8]:
```
create_sites_kvdb()
create_people_kvdb()
create_visits_kvdb()
create_measurements_kvdb()
```

In [10]:
```
kvdb_path = 'visits.json'
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)
```

In [2]:

```
from pathlib import Path
import json
import os

! pip install tinydb
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)
```

```
Collecting tinydb
  Downloading tinydb-4.7.0-py3-none-any.whl (24 kB)
Installing collected packages: tinydb
Successfully installed tinydb-4.7.0
```

In [3]:
```
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
        # Load in the jsons as dicts based on Teams thread feedback
        with open('results/kvdb/people.json', 'r') as file:
            people_dict = json.load(file)
        with open('results/kvdb/visited.json', 'r') as file:
            visited_dict = json.load(file)
        with open('results/kvdb/sites.json', 'r') as file:
            sites_dict = json.load(file)
        with open('results/kvdb/measurements.json', 'r') as file:
            measurements_dict = json.load(file)
        self._load_db()

        # Create records by people dictionary
        for people_k, people_v in people_dict.items():

            #Create a list of visits
            people_v['visits'] = []
            for visited_k, visited_v in visited_dict.items():
                for sites_k, sites_v in sites_dict.items():
                    # Matching up site_id by sites and visits. Then storing i
                    if sites_v['site_id'] == visited_v['site_id']:
                        visited_v['site'] = sites_v

                        # Create measurements list
                        visited_v['measurements'] = []
                        for measurements_k, measurements_v in measurements_di
                            # Matching visit_id and person_id with measuremen
                            if measurements_v['visit_id'] == visited_v['visit
                                visited_v['measurements'].append(measurements
                        # If there are measurements, add it to visit using pe
                        if len(visited_v['measurements']) != 0:
                            people_v['visits'].append(visited_v)

            # Once person record is complete add it to the db
            self._db.insert(people_v)

    def _load_db(self):
        self._db = TinyDB(self._db_path, indent=4, separators=(',', ': '))
```

In [4]:
```
db_path = results_dir.joinpath('patient-info.json')
if db_path.exists():
    os.remove(db_path)

db = DocumentDB(db_path)
```

In [1]:
```
from pathlib import Path
import os
import sqlite3
import pandas as pd
```

In [2]:
```
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 read_cluster_csv(file_path):
    #file_path ='D:/VZW Twinsburg/Tuition assistance/Bellevue University_MSDS
    return pd.read_csv(open(file_path, mode='rb'))
```

## Create and Load Measurements Table

In [3]:
```
def create_measurements_table(conn):
    sql = """
    CREATE TABLE IF NOT EXISTS measurements (
        visit_id integer NOT NULL,
        person_id text NOT NULL,
        quantity text,
        reading real,
        FOREIGN KEY (visit_id) REFERENCES visits (visit_id),
        FOREIGN KEY (person_id) REFERENCES people (people_id)
        );
    """

    c = conn.cursor()
    c.execute(sql)

def load_measurements_table(conn):
    create_measurements_table(conn)
    df = read_cluster_csv('D:/VZW Twinsburg/Tuition assistance/Bellevue Unive
    measurements = df.values
    c = conn.cursor()
    c.execute('DELETE FROM measurements;') # Delete data if exists
    c.executemany('INSERT INTO measurements VALUES (?,?,?,?)', measurements)
```

## Create and Load People Table

In [4]:
```
def create_people_table(conn):
    ## TODO: Complete SQ
    sql = """ CREATE TABLE IF NOT EXISTS people (
        person_id text PRIMARY KEY,
        personal text,
        family_name text
        );
    """
    c = conn.cursor()
    c.execute(sql)

def load_people_table(conn):
    create_people_table(conn)
    ## TODO: Complete code
    df = read_cluster_csv('D:/VZW Twinsburg/Tuition assistance/Bellevue Unive
    people = df.values
    c = conn.cursor()
    c.execute('DELETE FROM people;') # Delete data if exists
    c.executemany('INSERT INTO people VALUES (?,?,?)', people)
```

## Create and Load Sites Table

In [5]:
```
def create_sites_table(conn):
    sql = """
    CREATE TABLE IF NOT EXISTS sites (
        site_id text PRIMARY KEY,
        latitude double NOT NULL,
        longitude double NOT NULL
        );
    """

    c = conn.cursor()
    c.execute(sql)

def load_sites_table(conn):
    create_sites_table(conn)
    ## TODO: Complete code
    df = read_cluster_csv('D:/VZW Twinsburg/Tuition assistance/Bellevue Unive
    sites = df.values
    c = conn.cursor()
    c.execute('DELETE FROM sites;') # Delete data if exists
    c.executemany('INSERT INTO sites VALUES (?,?,?)', sites)
```

## Create and Load Visits Table

In [6]:
```
def create_visits_table(conn):
    sql = """
    CREATE TABLE IF NOT EXISTS visits (
        visit_id integer PRIMARY KEY,
        site_id text NOT NULL,
        visit_date text,
        FOREIGN KEY (site_id) REFERENCES sites (site_id)
        );
    """

    c = conn.cursor()
    c.execute(sql)

def load_visits_table(conn):
    create_visits_table(conn)
    ## TODO: Complete code
    df = read_cluster_csv('D:/VZW Twinsburg/Tuition assistance/Bellevue Unive
    visits = df.values
    c = conn.cursor()
    c.execute('DELETE FROM visits;') # Delete data if exists
    c.executemany('INSERT INTO visits VALUES (?,?,?)', visits)
```

## Create DB and Load Tables

In [7]:
```
db_path = results_dir.joinpath('patient-info.db')
conn = sqlite3.connect(str(db_path))
# TODO: Uncomment once functions completed
load_people_table(conn)
load_sites_table(conn)
load_visits_table(conn)
load_measurements_table(conn)

conn.commit()
conn.close()
```

In [ ]:
```

```

Do you need help creating a query? You can build queries without having to write SPARQL in the new Query Builder (https://query.wikidata.org/querybuilder/?uselang=en).

```
#Recent Events
 SELECT ?date ?event ?eventLabel
 WHERE
 {
    # find events
    ?event wdt:P31/wdt:P279* wd:Q1190554.
    # with a point in time or start date
    OPTIONAL { ?event wdt:P585 ?date. }
    OPTIONAL { ?event wdt:P580 ?date. }
    # but at least one of those
    FILTER(BOUND(?date) && DATATYPE(?date) = xsd:dateTime).
    # not in the future, and not more than 31 days ago
    BIND(NOW() - ?date AS ?distance).
    FILTER(0 <= ?distance && ?distance < 31).
    # and get a label as well
    OPTIONAL {
        ?event rdfs:label ?eventLabel.
        FILTER(LANG(?eventLabel) = "en").
    }
 }
# limit to 10 results so we don't timeout
 LIMIT 10
```

(https://www.wikidata.org/wiki/Special:MyLanguage/Wikidata:SPARQL_query_service/Wikidata_Query_Help/Result_Views)

10 results in 41531 ms  Code  Download  Link

| date | event | eventLabel |
|---|---|---|
| 11 August 2022 | (http://www.wikidata.org/entity/Q97000628) wd:Q97000628 (http://www.wikidata.org/entity/Q97000628) | Wikimania 2022 |
| 28 August 2022 | (http://www.wikidata.org/entity/Q108330783) wd:Q108330783 (http://www.wikidata.org/entity/Q108330783) | Another World: The Transcendental Painting Group |
| 15 August 2022 | (http://www.wikidata.org/entity/Q108749396) wd:Q108749396 (http://www.wikidata.org/entity/Q108749396) | 75th Year of Independence Day of India |
| 25 August 2022 | (http://www.wikidata.org/entity/Q108822097) wd:Q108822097 (http://www.wikidata.org/entity/Q108822097) | 2022 IIHF Women's World Championship |
| 19 August 2022 | (http://www.wikidata.org/entity/Q111527387) wd:Q111527387 (http://www.wikidata.org/entity/Q111527387) | State of the Map 2022 |
| 15 August 2022 | (http://www.wikidata.org/entity/Q111739923) wd:Q111739923 (http://www.wikidata.org/entity/Q111739923) | GAMM 2022 |
| 7 September 2022 | (http://www.wikidata.org/entity/Q111751391) wd:Q111751391 (http://www.wikidata.org/entity/Q111751391) | The 18th International Conference on Open Source Systems and The 18th International Symposium on Open Collaboration |
| 1 September 2022 | (http://www.wikidata.org/entity/Q111806007) wd:Q111806007 (http://www.wikidata.org/entity/Q111806007) | Wiki Loves Monuments 2022 in Italy |
| 9 September 2022 | (http://www.wikidata.org/entity/Q112034807) wd:Q112034807 (http://www.wikidata.org/entity/Q112034807) | André Devambez (1867-1944) Vertiges de l'imagination |
| 21 August 2022 | (http://www.wikidata.org/entity/Q112228075) wd:Q112228075 (http://www.wikidata.org/entity/Q112228075) | 2022 Asian Women's Volleyball Cup |