Create SQLite DB from Wikipedia

Create an SQLite DB based on the British Monarch Family Tree.

https://en.wikipedia.org/w/index.php?title=Family_tree_of_British_monarchs&oldid=1043575587

```
## set up tables: british_monarch_family_tree
import sqlite3
conn = sqlite3.connect('british_monarch_family_tree.db')

cur = conn.cursor()

cur.execute('DROP TABLE IF EXISTS british_monarch_family_tree ')
cur.execute('CREATE TABLE british_monarch_family_tree ('\
            'id INTEGER PRIMARY KEY AUTOINCREMENT, '\
            'name TEXT, '\
            'wiki_url TEXT)')

import requests
from bs4 import BeautifulSoup

# Fetch the page content from the url
user_agent = {'User-agent': 'Mozilla/5.0'}
url = 'https://en.wikipedia.org/w/index.php?title=Family_tree_of_British_monarchs&oldid=1043575587'
page = requests.get(url, headers = user_agent)
soup = BeautifulSoup(page.content)

conn = sqlite3.connect('british_monarch_family_tree.db')
cur = conn.cursor()

for tr in soup.find('table').find_all('td'):
    link = tr.find('a')
    if link and link.text != '' and 'House' not in link.text and 'House' not in link.attrs['href'] and 'History' not in
link.attrs['href']:
        cur.execute('INSERT INTO british_monarch_family_tree (name, wiki_url) VALUES (?, ?)',
```

```
                (link.get_text(separator=" ").strip(), link.attrs['href']))
conn.commit()
conn.close()
```

Adding foreign key, Mother and Father.

```
# Alter table to add columns with foreign key
conn = sqlite3.connect('british_monarch_family_tree.db')
cur = conn.cursor()
cur.execute('ALTER TABLE british_monarch_family_tree '\
        'ADD COLUMN father_id INTEGER DEFAULT NULL '\
        'REFERENCES british_monarch_family_tree(id)')
cur.execute('ALTER TABLE british_monarch_family_tree '\
        'ADD COLUMN mother_id INTEGER DEFAULT NULL '\
        'REFERENCES british_monarch_family_tree(id)')
conn.close()

# Fetch all from table
user_agent = {'User-agent': 'Mozilla/5.0'}

conn = sqlite3.connect('british_monarch_family_tree.db')
cur = conn.cursor()
cur.execute('SELECT COUNT(*) from british_monarch_family_tree')
table_size = cur.fetchone()[0]

#for c in range(1, table_size+1):
c = 1
while(c <= table_size):
    cur.execute("SELECT id, wiki_url from british_monarch_family_tree WHERE id = ?", (c,))
    r = cur.fetchone()

    # Navigate to each wiki_url https://en.wikipedia.org+wiki_url
    if r[1] == '':
        pass

    url = 'https://en.wikipedia.org' + r[1]
    page = requests.get(url, headers = user_agent)
    soup = BeautifulSoup(page.content)

    # find the father mother
```

```
    th_content = soup.find_all(class_="infobox-label")
    td_content = soup.find_all(class_="infobox-data")
    for i in range(0,len(th_content)):
        if th_content[i].text == 'Father' or th_content[i].text == 'Mother':
            link_tag = td_content[i].find('a')
            if link_tag is None:
                link = ''
            else:
                link = link_tag.attrs['href']
            name = td_content[i].get_text(separator=" ").strip()
            # Check db for exisiting record of parent and get id
            cur.execute("SELECT id from british_monarch_family_tree WHERE wiki_url LIKE ? OR name LIKE ?",
                        (link,name))
            # Add to db if missing
            if cur.fetchone() is None:
                cur.execute('INSERT INTO british_monarch_family_tree (name, wiki_url) VALUES (?, ?)',
                            (name, link))
                conn.commit()

                # Question did not state to find the parents of the parents
                # If true,row count will be 1000 == DatabaseError: database disk image is malformed
                # table_size += 1

            # Get the father id
            cur.execute("SELECT id from british_monarch_family_tree WHERE wiki_url LIKE ? OR name LIKE ?",
                        (link,name))

            # Update record with parents
            parent_id = cur.fetchone()
            if th_content[i].text == 'Father':
                cur.execute("UPDATE british_monarch_family_tree SET father_id = ? where id=?", (parent_id[0],r[0]))
            elif th_content[i].text == 'Mother':
                cur.execute("UPDATE british_monarch_family_tree SET mother_id = ? where id=?", (parent_id[0],r[0]))
    c+=1
conn.commit()
conn.close()
```

Find all children of King "George III":

```
conn = sqlite3.connect('british_monarch_family_tree.db')
cur = conn.cursor()
cur.execute('SELECT * from british_monarch_family_tree b1 '\
        'WHERE father_id = (SELECT id from british_monarch_family_tree '\
        'WHERE name = "George III")')
result = cur.fetchall()
conn.close()
for r in result:
    print (r)
```

Output:

```
(85, 'George IV', '/wiki/George_IV_of_the_United_Kingdom', 84, 176)
(86, 'William IV', '/wiki/William_IV_of_the_United_Kingdom', 84, 176)
(87, 'Edward Duke of Kent and Strathearn', '/wiki/Prince_Edward,_Duke_of_Kent_and_Strathearn', 84, 176)
```

Find the father and mother of King "George III":

```
conn = sqlite3.connect('british_monarch_family_tree.db')
cur = conn.cursor()
cur.execute('Select * from british_monarch_family_tree '\
        'WHERE id = (SELECT father_id from british_monarch_family_tree WHERE name = "George III") '\
        'OR id = (SELECT mother_id from british_monarch_family_tree WHERE name = "George III")')
result = cur.fetchall()
conn.close()
for r in result:
    print (r)
```

Output:

```
(83, 'Frederick Prince of Wales', '/wiki/Frederick,_Prince_of_Wales', 82, 174)
(175, 'Princess Augusta of Saxe-Gotha', '/wiki/Princess_Augusta_of_Saxe-Gotha', None, None)
```

Find all descendants of "Queen Victoria":

```
# Load db table into df
conn = sqlite3.connect('british_monarch_family_tree.db')
cur = conn.cursor()
cur.execute('SELECT * from british_monarch_family_tree')
```

```
result = cur.fetchall()
conn.close()

df_db = pd.DataFrame(result,
                     columns=['id', 'name', 'wiki_url', 'father_id', 'mother_id'])
df_db = df_db.set_index('id')

# find queen victoria id
root_id = df_db.loc[df_db['name'] == 'Victoria'].index[0]

# make a temp list of descendants to hold index
descendants = []

# insert victoria id into the list index 0
descendants.append(root_id)

# while loop to keep looping till no more descendant
# for loop to fetch new descendants
index = 0
while(index < len(descendants)):
    for id in df_db.loc[(df_db['father_id'] == descendants[index]) |
                        (df_db['mother_id'] == descendants[index])].index:
        if id not in descendants:
            descendants.append(id)
    index+=1

# print df and skip first index 0 victoria
df_db.iloc[[i-1 for i in descendants[1::]]]
```

Output in Dataframe:

id	name	wiki_url	father_id	mother_id
89	Edward VII	/wiki/Edward_VII	178.0	88.0
90	George V	/wiki/George_V	89.0	179.0
91	Edward VIII	/wiki/Edward_VIII	90.0	180.0
92	George VI	/wiki/George_VI	90.0	180.0
93	Elizabeth II	/wiki/Elizabeth_II	92.0	181.0

Revision #1
Created 1 January 2024 07:06:53 by aki
Updated 1 January 2024 07:15:07 by aki