SQlite3

December 20, 2021

[27]:
```
import sqlite3
import pandas as pd
connection=sqlite3.connect('database.db')
cursor=connection.cursor()
cursor.execute('DROP TABLE IF EXISTS parents')

table='''CREATE TABLE parents (
          parent VARCHAR(20),
          child VARCHAR(20));'''
cursor.execute(table)
insert='''INSERT INTO parents (parent, child)
          VALUES ("abraham", "barack") UNION
          VALUES ("abraham", "clinton") UNION
          VALUES ("delano", "herbert") UNION
          VALUES ("fillmore", "abraham") UNION
          VALUES ("fillmore", "delano") UNION
          VALUES ("fillmore", "grover") UNION
          VALUES ("eisenhower", "fillmore");
      '''
cursor.execute(insert)
connection.commit()
connection.close()
```

[28]:
```
connection=sqlite3.connect('database.db')
cursor=connection.cursor()
A=cursor.execute('SELECT * FROM parents')

for row in A.fetchall():
  print(row)
```

```
('abraham', 'barack')
('abraham', 'clinton')
('delano', 'herbert')
('eisenhower', 'fillmore')
('fillmore', 'abraham')
('fillmore', 'delano')
('fillmore', 'grover')
```

1 Q1 Simple SELECTS

Select all records in table

[29]:
```
connection=sqlite3.connect('database.db')
cursor=connection.cursor()
pd.read_sql("SELECT * FROM parents",con = connection)
```

[29]:

	parent	child
0	abraham	barack
1	abraham	clinton
2	delano	herbert
3	eisenhower	fillmore
4	fillmore	abraham
5	fillmore	delano
6	fillmore	grover

Select Child and parent where abraham is present

[30]:
```
pd.read_sql('SELECT * FROM parents WHERE [parent]="abraham"',con=connection)
```

[30]:

	parent	child
0	abraham	barack
1	abraham	clinton

SELECT all children that have an 'e' in their name

[31]:
```
pd.read_sql('SELECT * FROM parents WHERE [child] LIKE "%e%"',con=connection)
```

[31]:

	parent	child
0	delano	herbert
1	eisenhower	fillmore
2	fillmore	delano
3	fillmore	grover

SELECT all unique parents (use SELECT DISTINCT) and order them by name, descending order (i.e. fillmore first

[32]:
```
pd.read_sql('SELECT DISTINCT [parent] FROM parents ORDER BY parent␣
↪DESC',con=connection)
```

[32]:

	parent
0	fillmore
1	eisenhower
2	delano
3	abraham

SELECT all dogs that are siblings (one-to-one relations). Only show a sibling pair once. To do this you need to select two times from the parents table.

[33]:
```
# Creating pairs of siblings(
pd.read_sql('SELECT A.child AS "Child1", B.child AS "Child2"\
            FROM (SELECT [child],[parent] \
                  FROM parents \
                  WHERE [parent] \
                  IN (SELECT [parent] from parents GROUP BY [parent] \
                  HAVING COUNT([parent])>1)) A \
            cross JOIN parents B \
            ON A.parent=B.parent\
            WHERE ((A.child!=B.child) AND (A.child>B.child))'\
                     ,con=connection)
```

[33]:

	Child1	Child2
0	clinton	barack
1	delano	abraham
2	grover	abraham
3	grover	delano

[34]:
```
connection.close()
```

2 Q2. Create a new table Dogs

[35]:
```
connection=sqlite3.connect('database.db')
cursor=connection.cursor()
cursor.execute('DROP TABLE IF EXISTS dogs')

query='''CREATE TABLE dogs AS
SELECT "abraham" AS name, "long" AS fur UNION
SELECT "barack", "short" UNION
SELECT "clinton", "long" UNION
SELECT "delano", "long" UNION
SELECT "eisenhower", "short" UNION
SELECT "fillmore", "curly" UNION
SELECT "grover", "short" UNION
SELECT "herbert", "curly";'''
cursor.execute(query)
connection.commit()
connection.close()
```

[36]:
```
connection=sqlite3.connect('database.db')
cursor=connection.cursor()
pd.read_sql('SELECT * FROM dogs',con=connection)
```

[36]:

	name	fur
0	abraham	long
1	barack	short

```
2     clinton   long
3      delano   long
4  eisenhower  short
5    fillmore  curly
6      grover  short
7     herbert  curly
```

COUNT the number of long haired dogs

[37]:
```
pd.read_sql('SELECT COUNT(*)AS "Dogs with long hair" FROM dogs WHERE␣
↪[fur]="long"',con=connection)
```

[37]:
```
   Dogs with long hair
0                    3
```

JOIN tables parents and dogs and SELECT the parents of curly dogs.

[38]:
```
query='''SELECT parents.parent,dogs.fur\
      FROM parents\
      INNER JOIN dogs ON parents.child=dogs.name
      WHERE fur="curly"'''

pd.read_sql(query,con=connection)
```

[38]:
```
       parent    fur
0  eisenhower  curly
1      delano  curly
```

JOIN tables parents and dogs, and SELECT the parents and children that have the same fur type. Only show them once.

[39]:
```
pd.read_sql('SELECT [parent],[child] FROM (SELECT [parent],[child],(SELECT␣
↪[fur] FROM dogs where name==[parent]) AS "parent fur" ,\
            (SELECT [fur] FROM dogs where name==[child]) AS "child fur"\
            FROM parents A \
            INNER JOIN dogs B\
            ON A.parent=B.name) \
            WHERE [parent fur]=[child fur]\
            ',con=connection)
```

[39]:
```
    parent    child
0  abraham  clinton
```

[19]:
```
#Alternate approach
pd.read_sql('SELECT [parent],[child] \
            FROM parents A \
            INNER JOIN dogs B\
            ON A.parent=B.name\
```

```
            WHERE (SELECT [fur] FROM dogs where name==[parent])\
            =(SELECT [fur] FROM dogs where name==[child]) ',con=connection)
```

[19]:

	parent	child
0	abraham	clinton

[41]:
```
#Alternate approach without joins


pd.read_sql('SELECT [parent],[child]\
            FROM parents\
            WHERE (SELECT [fur] FROM dogs where name==[parent])\
            =(SELECT [fur] FROM dogs where name==[child])\
            ',con = connection)
```

[41]:

	parent	child
0	abraham	clinton

[ ]:
```
connection.close()
```

3 Q3. Aggregate functions, numerical logic and grouping

[ ]:
```
connection = sqlite3.connect('database.db')
cursor=connection.cursor()
cursor.execute('DROP TABLE IF EXISTS animals')
query='''create table animals as
select "dog" as kind, 4 as legs, 20 as weight union
select "cat" , 4 , 10 union
select "ferret" , 4 , 10 union
select "parrot" , 2 , 6 union
select "penguin" , 2 , 10 union
select "t-rex" , 2 , 12000;'''

cursor.execute(query)
connection.commit()
connection.close()
```

[ ]:
```
# Printing the data
connection=sqlite3.connect('database.db')
cursor=connection.cursor()

pd.read_sql('SELECT * from animals ',con=connection)
```

[ ]:

	kind	legs	weight
0	cat	4	10
1	dog	4	20

```
2   ferret     4      10
3   parrot     2       6
4  penguin     2      10
5    t-rex     2   12000
```

SELECT the animal with the minimum weight. Display kind and min_weight.

[ ]:
```
pd.read_sql('SELECT [kind],MIN([weight]) FROM animals',con=connection)
```

[ ]:
```
     kind  MIN([weight])
0  parrot              6
```

Use the aggregate function AVG to display a table with the average number of legs and the average

[ ]:
```
pd.read_sql('SELECT AVG([legs]) AS "Average Legs" ,AVG([weight]) AS "Average␣
 ↪Weight" FROM animals',con = connection)
```

[ ]:
```
   Average Legs  Average Weight
0           3.0     2009.333333
```

SELECT the animal kind(s) that have more than two legs, but weighs less than 20. Display kind, weight,legs.

[ ]:
```
pd.read_sql('SELECT [kind] FROM animals WHERE legs=2 AND weight>20',con =␣
 ↪connection)
```

[ ]:
```
    kind
0  t-rex
```

SELECT the average weight for all the animals with 2 legs and the animals with 4 legs (by using GROUPBY)

[ ]:
```
pd.read_sql('SELECT [legs],AVG([weight]) AS "Average Weights"\
             FROM animals \
             GROUP BY [legs]',con = connection)
```

[ ]:
```
   legs  Average Weights
0     2      4005.333333
1     4        13.333333
```

[ ]: