In [6]:
```
# import sqlite3 package
import sqlite3
import pandas as pd
```

In [36]:
```
# Connect to our database
connection = sqlite3.connect('dogs.db')
```

In [37]:
```
#cursor.execute('DROP TABLE parents')
```

In [38]:
```
sql_command1 = """
CREATE TABLE parents (
    parent VARCHAR(20),
    child VARCHAR(20));"""

sql_command2 = """
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 = connection.cursor()
cursor.execute(sql_command1)
cursor.execute(sql_command2)
```

Out[38]:
```
<sqlite3.Cursor at 0x1f1c28ee880>
```

In [39]:
```
connection.commit()
```

In [40]:
```
# Q1 - 1
# Select everythin in the table parents

sql_command = '''SELECT * FROM parents'''

pd.read_sql_query(sql_command, connection)
```

Out[40]:

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

In [41]:
```
# Q1 - 2
# SELECT child and parent, where abraham is the parent

sql_command = '''SELECT child, parent FROM parents WHERE parent = "abraham"'''

pd.read_sql_query(sql_command, connection)
```

Out[41]:

	child	parent
0	barack	abraham
1	clinton	abraham

In [42]:
```
# Q1 - 3
#SELECT all children that have an 'e' in their name (hint: use LIKE and '%e%').

sql_command = "SELECT child FROM parents WHERE child LIKE '%e%'"

pd.read_sql_query(sql_command, connection)
```

Out[42]:

	child
0	herbert
1	fillmore
2	delano
3	grover

In [43]:
```
# Q1 - 4
#SELECT all unique parents (use SELECT DISTINCT) and order them by name, descendi

sql_command = """SELECT DISTINCT parent FROM parents ORDER BY parent"""

pd.read_sql_query(sql_command, connection)
```

Out[43]:

	parent
0	abraham
1	delano
2	eisenhower
3	fillmore

In [44]:
```
# Q1 - 5
# SELECT all dogs that are siblings (one-to-one relations). Only show a sibling p
# To do this you need to select two times from the parents table.

sql_command = """
SELECT DISTINCT A.PARENT AS Parent,
CASE WHEN A.CHILD < B.CHILD THEN A.CHILD ELSE B.CHILD END,
CASE WHEN A.CHILD > B.CHILD THEN A.CHILD ELSE B.CHILD END
FROM PARENTS A INNER JOIN PARENTS B ON A.PARENT = B.PARENT
WHERE A.CHILD != B.CHILD"""

pd.read_sql_query(sql_command, connection)
```

Out[44]:

	Parent	CASE WHEN A.CHILD < B.CHILD THEN A.CHILD ELSE B.CHILD END	CASE WHEN A.CHILD > B.CHILD THEN A.CHILD ELSE B.CHILD END
0	abraham	barack	clinton
1	fillmore	abraham	delano
2	fillmore	abraham	grover
3	fillmore	delano	grover

In [45]:
```
# Create dogs table including the fur characteristic

sql_command = """
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(sql_command)
connection.commit()
```

In [46]:
```
# Q2 - 1
# COUNT the number of short haired dogs

sql_command = """SELECT COUNT(name) AS ShortHairDogs FROM dogs WHERE fur LIKE 'sh

pd.read_sql_query(sql_command, connection)
```

Out[46]:

	ShortHairDogs
0	3

In [47]:
```
#Q2 - 2
#JOIN tables parents and dogs and SELECT the parents of curly dogs

sql_command = """
SELECT parents.parent
FROM parents
INNER JOIN dogs
ON parents.child = dogs.name
WHERE dogs.fur LIKE 'curly'"""

pd.read_sql_query(sql_command, connection)
```

Out[47]:

	parent
0	eisenhower
1	delano

In [48]:
```
#Q2 - 3
#JOIN tables parents and dogs, and SELECT the parents and children that have the s

sql_command = """
SELECT P.name AS ParentName, C.name AS ChildName, P.fur AS Fur
FROM dogs P, dogs C
WHERE P.name <> C.name
AND P.fur = C.fur
AND EXISTS (SELECT parent, child FROM parents WHERE child = C.name AND parent= P.
ORDER BY P.fur
"""

pd.read_sql_query(sql_command, connection)
```

Out[48]:

	ParentName	ChildName	Fur
0	abraham	clinton	long

In [11]:
```
#Creat table with many different animals

sql_command = """
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(sql_command)
```

Out[11]:
```
<sqlite3.Cursor at 0x201ef3895e0>
```

In [50]:
```
# Q3 - 1
# SELECT the animal with the minimum weight. Display kind and min_weight.

sql_command = """
SELECT kind, MIN(weight)
FROM animals
"""

pd.read_sql_query(sql_command, connection)
```

Out[50]:

	kind	MIN(weight)
0	parrot	6

In [51]:
```
# Q3 - 2
# Use the aggregate function AVG to display a table with the average number of leg

sql_command = """
SELECT AVG(legs), AVG(weight)
FROM animals
"""

pd.read_sql_query(sql_command, connection)
```

Out[51]:

	AVG(legs)	AVG(weight)
0	3.0	2009.333333

In [52]:
```
# Q3 - 3
# SELECT the animal kind(s) that have more than two legs, but weighs less than 20

sql_command = """
SELECT kind, weight, legs
FROM animals
WHERE weight < 20 AND legs > 2
"""
pd.read_sql_query(sql_command, connection)
```

Out[52]:

	kind	weight	legs
0	cat	10	4
1	ferret	10	4

In [53]:
```
# Q3 - 4
# SELECT the average weight for all the animals with 2 legs and the animals with 4

sql_command = """
SELECT AVG(weight)
FROM animals
GROUP BY legs
"""

pd.read_sql_query(sql_command, connection)
```

Out[53]:

	AVG(weight)
0	4005.333333
1	13.333333