DATA 311 - Fall 2020

Assignment 3 Name: J.Mo Yang Discussed with Sejin Park and Joanne Lee

We'll be using the same NYC_Health.db we've been looking at in class for this assignment

In [1]:
```
# Import any needed libraries
import sqlite3
import pandas as pd
!rm -f Test.db
```

In [2]:
```
# Connect to the database
conn=sqlite3.connect('./data/NYC_Health.db')
curs = conn.cursor()
```

In [3]:
```
# Helpful reminder - you can list out all of the tables and column names with this code:
x = pd.read_sql("""SELECT name
                FROM sqlite_master
                WHERE type = 'table'
                AND name LIKE 't%';""",conn)
for table in x.values:
    sql = "PRAGMA table_info(" + table[0] + ");"
    print(table)
    print(pd.read_sql(sql,conn))
    print('\n')
```

```
['tRestaurant']
   cid      name     type  notnull dflt_value  pk
0    0     camis  INTEGER        0       None   1
1    1       dba     TEXT        1       None   0
2    2  building     TEXT        1       None   0
3    3    street     TEXT        1       None   0
4    4       zip     TEXT        1       None   0
5    5     phone     TEXT        1       None   0
6    6   cuisine     TEXT        1       None   0
7    7      boro     TEXT        1       None   0


['tViolation']
   cid       name     type  notnull dflt_value  pk
0    0      camis  INTEGER        1       None   1
1    1       date     TEXT        1       None   2
2    2    viol_id     TEXT        1       None   3
3    3  action_id  INTEGER        1       None   4


['tViolationDetail']
   cid       name  type  notnull dflt_value  pk
0    0    viol_id  TEXT        0       None   1
1    1  viol_desc  TEXT        1       None   0


['tAction']
   cid         name  type  notnull dflt_value  pk
0    0    action_id  TEXT        0       None   1
1    1  action_desc  TEXT        1       None   0
```

---

1) For restaurants with cuisine type containing "Soups" (there might be a few different cuisine types containing that word), how many times has each of the five actions been taken?

Have your query return 3 columns: the cuisine type, the action description, and a count of the number of occurences. Make sure to return all 5 actions, even if their count is 0.

Hint: Since there are 5 actions, make sure the number of rows in your results are a multiple of five!

In [17]:
```
curs.execute("DROP VIEW IF EXISTS vSoups;")
curs.execute("""CREATE VIEW vSoups AS
                SELECT cuisine, action_desc
```

```
                FROM tAction
                JOIN tRestaurant
                WHERE cuisine like '%Soups%'
                GROUP BY cuisine, action_desc;""")
pd.read_sql("""SELECT *
                FROM vSoups;""", conn)
```

Out[17]:

	cuisine	action_desc
0	Soups	Establishment Closed by DOHMH. Violations wer...
1	Soups	Establishment re-closed by DOHMH
2	Soups	Establishment re-opened by DOHMH
3	Soups	No violations were recorded at the time of thi...
4	Soups	Violations were cited in the following area(s).
5	Soups & Sandwiches	Establishment Closed by DOHMH. Violations wer...
6	Soups & Sandwiches	Establishment re-closed by DOHMH
7	Soups & Sandwiches	Establishment re-opened by DOHMH
8	Soups & Sandwiches	No violations were recorded at the time of thi...
9	Soups & Sandwiches	Violations were cited in the following area(s).

In [31]:

```
curs.execute("DROP ViEW IF EXISTS vSoups2;")
curs.execute("""CREATE VIEW vSoups2 AS
                SELECT cuisine AS cuisine_s, action_desc AS action_desc_s, count(*) as Action_Occur
                FROM tViolation
                JOIN tRestaurant USING (camis)
                JOIN tAction USING (action_id)
                WHERE cuisine LIKE '%Soups%'
                GROUP BY cuisine, action_id""")
pd.read_sql("""SELECT *
                FROM vSoups2;""",conn)
```

Out[31]:

	cuisine_s	action_desc_s	Action_Occur
0	Soups	Violations were cited in the following area(s).	27
1	Soups	No violations were recorded at the time of thi...	1
2	Soups & Sandwiches	Violations were cited in the following area(s).	553
3	Soups & Sandwiches	Establishment Closed by DOHMH. Violations wer...	6
4	Soups & Sandwiches	Establishment re-opened by DOHMH	3

In [32]:

```
#Final
pd.read_sql("""SELECT cuisine, action_desc, IFNULL(Action_occur, 0) AS Action_occur
                FROM vSoups as A
                    LEFT JOIN vSoups2 as B
                    ON A.cuisine=B.cuisine_s
                    AND A.action_desc=B.action_desc_s;""", conn)
```

Out[32]:

	cuisine	action_desc	Action_occur
0	Soups	Establishment Closed by DOHMH. Violations wer...	0
1	Soups	Establishment re-closed by DOHMH	0
2	Soups	Establishment re-opened by DOHMH	0
3	Soups	No violations were recorded at the time of thi...	1
4	Soups	Violations were cited in the following area(s).	27
5	Soups & Sandwiches	Establishment Closed by DOHMH. Violations wer...	6
6	Soups & Sandwiches	Establishment re-closed by DOHMH	0
7	Soups & Sandwiches	Establishment re-opened by DOHMH	3
8	Soups & Sandwiches	No violations were recorded at the time of thi...	0

	cuisine	action_desc	Action_occur
9	Soups & Sandwiches	Violations were cited in the following area(s).	553

---

2) What is the most common violation for places that sell donuts? (That is, the cuisine type contains the word "donut")

Have your query return a single row with two columns: The description of the violation, and the number of occurences.

In [11]:
```
pd.read_sql("""SELECT viol_desc, count(viol_id) as NumViolations
                FROM tViolation
                    JOIN tViolationDetail USING(viol_id)
                    JOIN tRestaurant USING (camis)
                WHERE cuisine LIKE "%donut%"
                GROUP BY viol_id
                ORDER BY NumViolations DESC
                LIMIT 1;""", conn)
```

Out[11]:

	viol_desc	NumViolations
0	Non-food contact surface improperly constructe...	1236

In [76]:
```
```

Out[76]:

	viol_desc	NumViolations
0	Raw, cooked or prepared food is adulterated, c...	205
1	Evidence of mice or live mice present in facil...	238
2	Sanitized equipment or utensil, including in-u...	281
3	Cold food item held above 41Â° F (smoked fish ...	323
4	Plumbing not properly installed or maintained;...	327
5	Filth flies or food/refuse/sewage-associated (...	333
6	Food not protected from potential source of co...	437
7	Food contact surface not properly washed, rins...	455
8	Facility not vermin proof. Harborage or condit...	460
9	Non-food contact surface improperly constructe...	1236

---

3) For places that sell donuts, how many times have each of them committed the violation above? You do not need to return any restaurants that have not committed that violation.

Have your query return: dba, violation code, and number of occurences.

Group the results by DBA, and sort them with the biggest offender at the top.

Note: It may look like some DBA names are repeated even if you GROUP BY properly - some of them actually have extra spaces, etc.

In [47]:
```
curs.execute("DROP VIEW IF EXISTS vDonut_viol")
curs.execute("""CREATE VIEW vDonut_viol AS
                SELECT *
                FROM tRestaurant
                    JOIN tViolation USING (camis)
                    JOIN tViolationDetail USING (viol_id)
                    JOIN tAction USING (action_id)
                    JOIN vDonut_seller USING (viol_id)
                WHERE cuisine LIKE '%donut%'
                AND viol_id = '10F';""")
```

Out[47]: <sqlite3.Cursor at 0x7fcdf0bd45e0>

In [52]:
```
pd.read_sql("""SELECT DISTINCT dba, viol_id, count(viol_id) AS Viol_occurance
                FROM vDonut_viol
                GROUP BY (dba)
                ORDER BY Viol_occurance DESC;""",conn)
```

Out[52]:

	dba	viol_id	Viol_occurance
0	DUNKIN' DONUTS	10F	859
1	DUNKIN' DONUTS, BASKIN ROBBINS	10F	216
2	DUNKIN DONUTS	10F	35
3	DUNKIN DONUTS & BASKIN ROBBINS	10F	11
4	DUNKIN' DONUTS/ BASKIN ROBBINS	10F	10
5	TWIN DONUT	10F	9
6	DUNKIN' DONUTS/BASKIN ROBBINS	10F	8
7	DONUT SHOPPE	10F	7
8	TWIN DONUT PLUS	10F	6
9	NOSTRAND DONUT SHOP	10F	6
10	COUNTRY DONUTS	10F	6
11	DUNKIN' DONUTS, BASKIN ROBBINS, SUBWAY	10F	4
12	DUNKIN' DONUTS	10F	4
13	DONUT PUB	10F	4
14	KRISPY KREME	10F	3
15	IDEAL DONUT	10F	3
16	DUNKIN' DONUTS/SUBWAY	10F	3
17	DUNKIN' DONUTS/Baskin Robbins	10F	3
18	DUNKIN' DONUTS/BR EXPRESS	10F	3
19	DUNKIN' DONUTS, BASKINS ROBBINS	10F	3
20	DUNKIN' DONUTS BASKIN ROBBINS	10F	3
21	DUNKIN DONUTS BASKIN ROBBINS	10F	3
22	DUN-WELL DOUGHNUTS	10F	3
23	7TH AVENUE DONUT SHOP	10F	3
24	HILLSIDE GOURMET	10F	2
25	Dunkin→ Donuts	10F	2
26	DUNKIN' DONUTS/HUDSON NEWS	10F	2
27	DUNKIN' DONUTS - BASKIN ROBBINS	10F	2
28	DONUT CONNECTIONS	10F	2
29	CRUSTY & TASTY BAGEL	10F	2
30	TAJ DONUT SHOP	10F	1
31	MIKE'S DONUTS	10F	1
32	MARKET FOODS	10F	1
33	Dunkin Donuts	10F	1
34	DUNKIN' DONUTS-BASKIN ROBBINS	10F	1
35	DUNKIN DONUTS/POPEYES	10F	1
36	DUNKIN DONUTS,BASKIN ROBBINS	10F	1
37	DU'S DONUTS AND COFFEE	10F	1

	dba	viol_id	Viol_occurance
38	DOUGHNUT PLANT	10F	1

---

4) How many violations per DBA per Boro?

Return four columns: The Boro, the number of distinct DBA in the Boro, the number of violations in the Boro, and the number of violations per distinct DBA per Boro.

Hint: Watch out for integer division!

In [55]:
```
#DBA per boro
pd.read_sql("""SELECT BORO, count(DISTINCT dba) as NumLocations
                FROM tRestaurant
                GROUP BY BORO;""", conn)
```

Out[55]:

	boro	NumLocations
0	BRONX	1870
1	BROOKLYN	5397
2	MANHATTAN	8190
3	QUEENS	4869
4	STATEN ISLAND	783

In [61]:
```
#Violations per boro
pd.read_sql("""SELECT BORO, count(viol_id) as NumViolations
                FROM tViolation as A
                    JOIN tRestaurant as B
                    ON A.camis = B.camis
                GROUP BY BORO;""",conn)
```

Out[61]:

	boro	NumViolations
0	BRONX	34311
1	BROOKLYN	97628
2	MANHATTAN	156845
3	QUEENS	90817
4	STATEN ISLAND	13232

In [62]:
```
#Views, then join
curs.execute("""DROP VIEW IF EXISTS vNumDBAPerBoro;""")
curs.execute("""CREATE VIEW vNumDBAPerBoro AS
                SELECT BORO, count(DISTINCT DBA) as NumDBA
                FROM tRestaurant
                GROUP BY BORO;""")
```

Out[62]: `<sqlite3.Cursor at 0x7f1d8bc3aea0>`

In [63]:
```
curs.execute("DROP VIEW IF EXISTS vNumViolPerBoro;")
curs.execute("""CREATE VIEW vNumViolPerBoro AS
                SELECT BORO, count(viol_id) as NumViolations
                FROM tViolation as A
                    JOIN tRestaurant as B
                    ON A.camis = B.camis
                GROUP BY BORO;""")
```

Out[63]: `<sqlite3.Cursor at 0x7f1d8bc3aea0>`

In [12]:
```
#Final
pd.read_sql("""SELECT BORO, NumDBA, NumViolations,
                            1.0*NumViolations/NumDBA AS ViolPerDBA
```

```
                FROM vNumDBAPerBoro
                JOIN vNumViolPerBoro USING(BORO);""",conn)
```

Out[12]:

	boro	NumDBA	NumViolations	ViolPerDBA
0	BRONX	1870	34311	18.348128
1	BROOKLYN	5397	97628	18.089309
2	MANHATTAN	8190	156845	19.150794
3	QUEENS	4869	90817	18.652085
4	STATEN ISLAND	783	13232	16.899106

---

5) In Brooklyn, how many of each type of violation have been committed, grouped by cuisine type?

Return three columns: The cuisine type, the violation description, and the number of occurences

It will be a long list - don't do any ordering, I'll just look at the first and last few results that come out.

In [5]:
```
pd.read_sql("""SELECT cuisine, viol_desc, count(viol_desc) AS viol_occurences
                FROM tViolation
                JOIN tViolationDetail USING (viol_id)
                JOIN tRestaurant USING (camis)
                WHERE boro LIKE 'BROOKLYN'
                GROUP BY cuisine;""",conn)
```

Out[5]:

	cuisine	viol_desc	viol_occurences
0	Afghan	Evidence of mice or live mice present in facil...	40
1	African	Evidence of rats or live rats present in facil...	366
2	American	Live roaches present in facility's food and/or...	19314
3	Armenian	Plumbing not properly installed or maintained;...	82
4	Asian	Plumbing not properly installed or maintained;...	1034
...	...	...	...
74	Tex-Mex	Food not protected from potential source of co...	847
75	Thai	Cold food item held above 41Âº F (smoked fish ...	1179
76	Turkish	Facility not vermin proof. Harborage or condit...	413
77	Vegetarian	Facility not vermin proof. Harborage or condit...	535
78	Vietnamese/Cambodian/Malaysia	Non-food contact surface improperly constructe...	467

79 rows × 3 columns

In [ ]:

In [ ]: