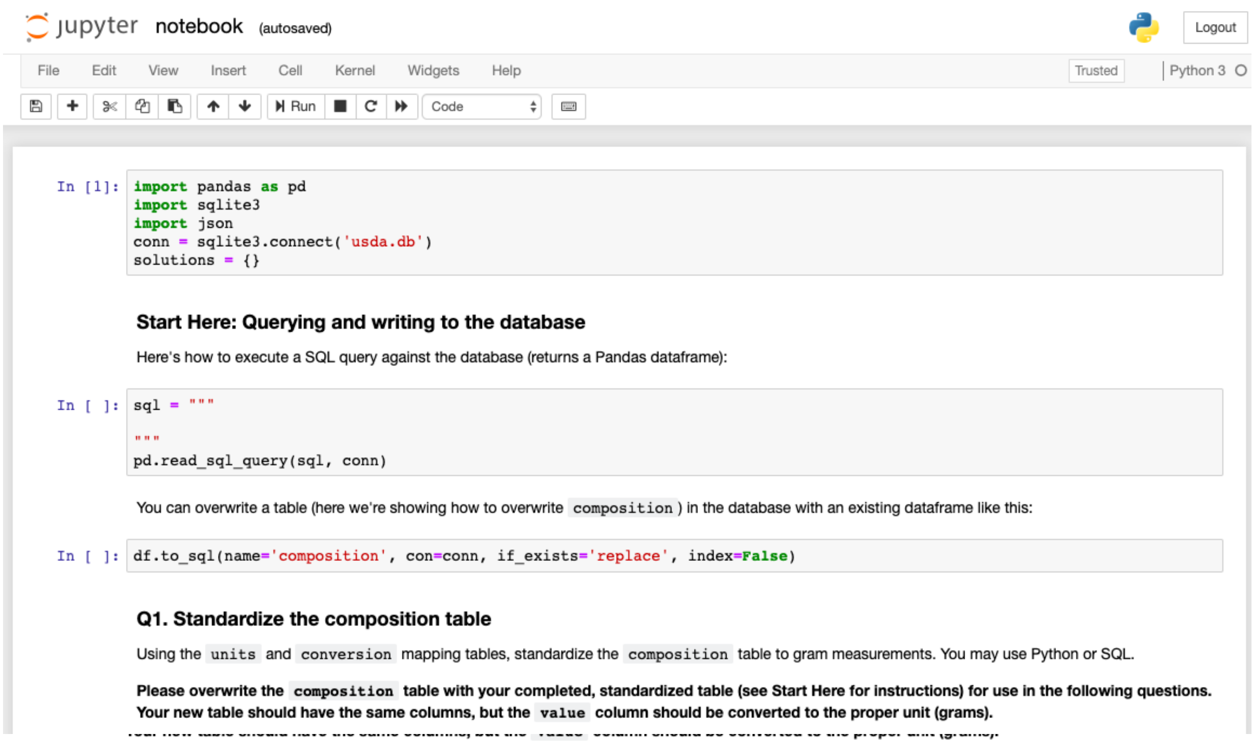

jupyter notebook (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

In [1]:
```
import pandas as pd
import sqlite3
import json
conn = sqlite3.connect('usda.db')
solutions = {}
```

Start Here: Querying and writing to the database

Here's how to execute a SQL query against the database (returns a Pandas dataframe):

In [ ]:
```
sql = """

"""
pd.read_sql_query(sql, conn)
```

You can overwrite a table (here we're showing how to overwrite `composition`) in the database with an existing dataframe like this:

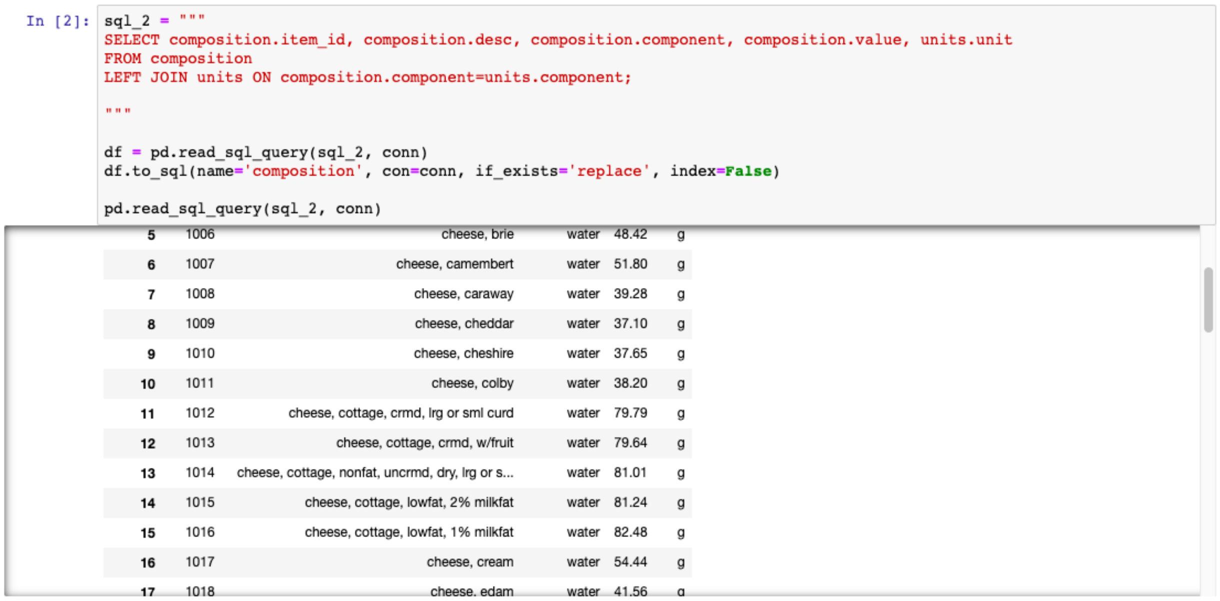
In [ ]:
```
df.to_sql(name='composition', con=conn, if_exists='replace', index=False)
```

Q1. Standardize the composition table

Using the `units` and `conversion` mapping tables, standardize the `composition` table to gram measurements. You may use Python or SQL.

Please overwrite the `composition` table with your completed, standardized table (see Start Here for instructions) for use in the following questions. Your new table should have the same columns, but the `value` column should be converted to the proper unit (grams).

Your new table should have the same columns, but the value column should be converted to the proper unit (grams).



In [2]:
```
sql_2 = """
SELECT composition.item_id, composition.desc, composition.component, composition.value, units.unit
FROM composition
LEFT JOIN units ON composition.component=units.component;

"""

df = pd.read_sql_query(sql_2, conn)
df.to_sql(name='composition', con=conn, if_exists='replace', index=False)

pd.read_sql_query(sql_2, conn)
```

5	1006	cheese, brie	water	48.42	g
6	1007	cheese, camembert	water	51.80	g
7	1008	cheese, caraway	water	39.28	g
8	1009	cheese, cheddar	water	37.10	g
9	1010	cheese, cheshire	water	37.65	g
10	1011	cheese, colby	water	38.20	g
11	1012	cheese, cottage, crmd, lrg or sml curd	water	79.79	g
12	1013	cheese, cottage, crmd, w/fruit	water	79.64	g
13	1014	cheese, cottage, nonfat, uncrmd, dry, lrg or s...	water	81.01	g
14	1015	cheese, cottage, lowfat, 2% milkfat	water	81.24	g
15	1016	cheese, cottage, lowfat, 1% milkfat	water	82.48	g
16	1017	cheese, cream	water	54.44	g
17	1018	cheese, edam	water	41.56	g

```
In [3]: sql_3 = """
SELECT composition.item_id, composition.desc, composition.component, CAST (composition.value * conversion.conversion AS
FROM composition
LEFT JOIN conversion ON composition.unit=conversion.unit;
"""
df = pd.read_sql_query(sql_3, conn)
df.to_sql(name='composition', con=conn, if_exists='replace', index=False)
print(df)
```

```
        item_id                                          desc  component  \
0          1001                               butter, with salt      water
1          1002                      butter, whipped, with salt      water
2          1003                          butter oil, anhydrous      water
3          1004                                   cheese, blue      water
4          1005                                  cheese, brick      water
5          1006                                   cheese, brie      water
6          1007                              cheese, camembert      water
7          1008                                cheese, caraway      water
8          1009                                cheese, cheddar      water
9          1010                               cheese, cheshire      water
10         1011                                  cheese, colby      water
11         1012          cheese, cottage, crmd, lrg or sml curd      water
12         1013                    cheese, cottage, crmd, w/fruit      water
13         1014  cheese, cottage, nonfat, uncrmd, dry, lrg or s...      water
14         1015             cheese, cottage, lowfat, 2% milkfat      water
15         1016             cheese, cottage, lowfat, 1% milkfat      water
16         1017                                  cheese, cream      water
17         1018                                   cheese, edam      water
```

Q2. Using SQL, find the top 5 foods by total vitamin content

All vitamins start with `vit_` (i.e. ignore thiamin and other vitamins that don't have "vitamin" in their name).

```
In [4]: sql_4 = """
SELECT DISTINCT component, desc, max(value)
FROM
 composition
WHERE
 component LIKE "%vit%"
GROUP BY
component
ORDER BY
max(value) desc limit 5;
"""

pd.read_sql_query(sql_4, conn)
```

Out[4]:

	component	desc	max(value)
0	vit_c	beverages, orange-flavor drk, brkfst type, lo ...	2.400000
1	vit_e	oil, wheat germ	0.149400
2	vit_b6	cereals rte, kellogg, kellogg's all-bran original	0.012000
3	vit_k	spices, basil, dried	0.001714
4	vit_d	fish oil, cod liver	0.000250

Replace the `None` with your answer. Your answer should be a list of strings.

```
In [ ]: solutions['q2'] = 'beverages(orange-flavor drink, breakfast type), oil(wheat germ), cereals, spices (dried basil), fish
```

Q3. Using SQL, find the average `sugar_tot` and `lipid_tot` of products containing the words "ice cream"

```
In [5]: sql_5 = """
SELECT DISTINCT
    component, desc, avg(value)
FROM
    composition
WHERE
    component IN ('lipid_tot', 'sugar_tot')
    AND
    desc LIKE "%ice cream%"
GROUP BY component
  ;
"""

pd.read_sql_query(sql_5, conn)
```

Out[5]:

	component	desc	avg(value)
0	lipid_tot	ice creams, choc, rich	9.940667
1	sugar_tot	ice creams, choc, rich	15.250385

Replace each `None` with your answers. Your answers should be floats, rounded to two decimal places.

```
In [ ]: solutions['q3'] = {
            'sugar_tot': 9.94,
            'lipid_tot': 15.25
        }
```

Q4. List all foods with more than 10 g of sodium in descending order by sodium content

You may use Python or SQL, but **bonus points for using Pandas** to solve this question.

```
In [6]: sql_6 = """
        SELECT DISTINCT
            component, desc, value
        FROM
            composition
        WHERE
            component LIKE "sodium" AND
            value >=10
        ORDER BY
        value desc
          ;
        """

        pd.read_sql_query(sql_6, conn)
```

Out[6]:

	component	desc	value
0	sodium	salt, table	38.758
1	sodium	leavening agents, baking soda	27.360
2	sodium	desserts, rennin, tablets, unswtnd	26.050
3	sodium	soup, bf broth or bouillon, pdr, dry	26.000
4	sodium	soup, beef broth, cubed, dry	24.000
5	sodium	soup, chick broth cubes, dry	24.000
6	sodium	soup, chick broth or bouillon, dry	23.875
7	sodium	seasoning mix, dry, sazon, coriander & annatto	17.000
8	sodium	gravy, au jus, dry	11.588
9	sodium	leavening agents, baking pdr, double-acting, n...	10.600

```
In [ ]: solutions['q4'] = 'table salt, baking soda, desserts, soup (chicke or beef broth), seasoning mix (sazon, corriander, ann
```

Q5. Generate a list of numbers by applying the below logic to the integers from 0 to 20

Modify each number based on the following logic:

- If the number is is less than 3, make no change to the number
- Otherwise, if the number is even and not equal to 5 or 6:
    - If the number is greater than or equal to 16, add 1 to the number
    - Otherwise, multiply the number by 2
- Otherwise, subtract 1 from the number

```
In [7]: for i in range(20):
            if i <3:
                print(i)
            elif ((i % 2 == 0) and (i!=5 or i!=6)):
                if (i>=16):
                    print(1+i)
                else:
                    print(i*2)
            else:
                print(i-1)
```

```
0
1
2
2
8
4
12
6
16
8
20
10
24
12
28
14
17
16
19
18
```

Replace the `None` with your answer. Your answer should be a list of integers.

```
In [8]: solutions['q5'] = 0,1,2,2,8,4,12,6,16,8,20,10,24,12,28,14,17,16,19,18
```

Run this cell to export your solutions to the grading file

Please double-check this solutions file before sending to make sure everything exported as expected.

```
In [11]: with open('solutions.json', 'w+') as f:
             json.dump(solutions, f)
```