Assignment 1

Some advice (I guess!)

Setting THINGS UP (Postgres)

Download:
https://www.postgresql.org/download/

Create a database:

```
CREATE DATABASE db_name;
\c db_name
```

Create table and copy csv file to table:
https://www.postgresqltutorial.com/import-csv-file-into-posgresql-table/

PANDAS AND POSTGRES

https://blog.panoply.io/connecting-jupyter-notebook-with-postgresql-for-python-data-analysis

In [1]:
```
import pandas as pd
import psycopg2
import sqlalchemy

from sqlalchemy import create_engine
```

In [2]:
```
POSTGRES_ADDRESS = 'localhost'
POSTGRES_PORT = '5439'
POSTGRES_USERNAME = 'postgres'
POSTGRES_PASSWORD = 'postgres'
POSTGRES_DBNAME = 'postgres'

postgres_str = ('postgresql://{username}:{password}@{ipaddress}:{port}/{dbname}'
    .format(
        username=POSTGRES_USERNAME,
        password=POSTGRES_PASSWORD,
        ipaddress=POSTGRES_ADDRESS,
        port=POSTGRES_PORT,
        dbname=POSTGRES_DBNAME
    )
)

cnx = create_engine(postgres_str)
```

In [3]:
```
pd.read_sql_query('''SELECT COUNT(*) FROM ingredients;''', cnx)
```

Out[3]:

	count
0	189059

MORE OPTIONS FOR SQL IN PYTHON

- psycopg2: https://pynative.com/python-postgresql-tutorial/
  Another way to execute SQL commands through Python

- https://towardsdatascience.com/jupyter-magics-with-sql-921370099589

PART A

- Select 20 random ingredients using SQL.
- For each element, find recipe ids whose corresponding ingredients match with the element.
- A match is determined using similarity measures: https://sites.google.com/site/anhaidgroup/projects/magellan/py_stringmatching/supported-features

PART B

- Traverse the rows (after retrieving from SQL)
- For each row, compute a similarity metric between ingredient and recipe name.
- If the metric indicates a match, include in output.

PART C

- Similar to A.
- Find rows whose ingredients match with the given query.
- Return the unique recipe titles.

General advice

- Traversing 1.9 million rows might take some time: I was able to do part A with a run time of around 15 seconds (Intel i7 @ 1.3 GHz).
- Add analysis - why did you end up preferring certain similarity metrics over others?