***Scenario**: You have been hired as an analyst by Nashvillians Encouraging Responsible Growth (NERG), a community organization that follows change in the city and the resulting impact to quality of services and quality of life around Nashville. You have been asked to explore both quantitative and qualitative datasets: Nashville building permits, short term rental permits, fire stations, and a summary of resident satisfaction, **and prepare a brief (5-7 minutes) presentation on your findings to deliver to the city council**. This guide will walk you through the initial data exploration and the steps needed to answer these specific questions:*

- *What are the most common types of building permits, and who is requesting them?*
- *What time of year are permits usually submitted?*
- *How long do permits take to be issued?*
- *How do the number of building permits differ across council districts?*
- *How do short term rental (Airbnb) permits compare to building permits across council districts?*
- *Are fire stations sufficient to handle the addition of new buildings?*
- *How do constituents feel about short term rental enforcement, fire services, and Nashville as a place to live?*

**Week One**

- *Unzip the shared file to create a directory for your Analytics Jumpstart work. Call it '**Analytics Jumpstart**'.*
- *Inside that directory, be sure there is a directory called '**data**'.*
- *Verify the **stations.db, multi_table.db, and entity_relationship_diagram.png** files are in the data folder.*
- *Over the next 3 weeks, you will be exploring the building permits, short-term rental permits, and fire stations in Nashville Davidson County. Download the CSV files for Building Permits Issued and Residential Short Term Rental Permits from https://data.nashville.gov and place them in the data folder.*
- *Launch Jupyter notebook from Anaconda Navigator. In your Jupyter server browser window, navigate to your Analytics Jumpstart directory and create a new Python 3 notebook. Call your notebook '**building_permits**'.*

1. Import the packages by running the code below in a Jupyter notebook cell. If you decide to import additional packages over the next few weeks, be sure to add them to this cell. Also add the command that starts with **%** to the bottom of this cell. **T**his is called a magic command and displays your plots in the notebook without having to also call a separate command.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sqlite3 as sql

%matplotlib inline
```

2. Read in building permits data to a dataframe called **building_permits**.
    a. Look at the first 5 rows.
    b. Look at the last 7 rows.
    c. How many rows and columns does **building_permits** contain?

3. Keep just these 15 columns:
    a. 'Permit #'
    b. 'Permit Type Description'
    c. 'Permit Subtype Description'

d. 'Parcel'
e. 'Date Entered'
f. 'Date Issued'
g. 'Construction Cost'
h. 'Address'
i. 'City'
j. 'State'
k. 'ZIP'
l. 'Contact'
m. 'Purpose'
n. 'Council District'
o. 'Mapped Location'

4. Rename the columns above:
    a. 'permit_num'
    b. 'type_desc'
    c. 'subtype_desc'
    d. 'parcel'
    e. 'date_entered'
    f. 'date_issued'
    g. 'cost'
    h. 'address'
    i. 'city'
    j. 'state'
    k. 'zipcode'
    l. 'contact'
    m. 'purpose'
    n. 'council_district'
    o. 'mapped_location'

5. Which **contact** is listed the most (ignore entries with **SELF CONTRACTOR**)? How many permits were issued to this contact?

6. Create a dataframe from the **type_desc** value counts called **description_counts**. It should have two columns called '**type_desc**' and '**type_desc_count**'.

7. Were there more **Residential** than C**ommercial** permits? How many of each? Were there any that were not **Residential** or **Commercial**? If so, how many?

8. Store the month from the **'date_entered'** column in the **building_permits dataframe** to a new column called **'month_entered'**. Create a dataframe from the value counts of this column and call it **'month_count'**.

9. Make a bar plot from **'month_count'** using matplotlib that shows the number of permits entered per month. What does the bar plot tell you?

10. What is the longest time a permit took to be issued? What is the shortest?

11. How many took more than 100 days?

**Week Two**

12. What years are covered in **building_permits**? Subset **building_permits** to just contain permits issued in **2019**, call it **building_permits_2019**.

13. Use the pandas groupby method to count the number of permits issued in each council district for 2019. Save this to a new dataframe, **building_permits_2019_council.** It should only contain two columns, call them **'district'** and **'count'**. Make a barplot showing the number of permits issued for each council district in 2019.

14. Import the short term rental permits dataset that you downloaded from data.nashville.gov into a data frame called **strp**.

15. Keep columns:
    a. 'Permit #'
    b. 'Applicant'
    c. 'Contact'
    d. 'Permit Subtype Description'
    e. 'Permit Status'
    f. 'Parcel'
    g. 'Date Entered'
    h. 'Date Issued'
    i. 'Expiration Date'
    j. 'Address'
    k. 'City'
    l. 'State'
    m. 'ZIP'
    n. 'Purpose'
    o. 'Permit Owner Name'
    p. 'Permit Type'
    q. 'Council District'
    r. 'Mapped Location'

16. Rename to:
    a. 'id'
    b. 'applicant'
    c. 'contact'
    d. 'subtype_desc'
    e. 'status'
    f. 'parcel'
    g. 'entered'
    h. 'issued'
    i. 'expiration'
    j. 'address'
    k. 'city'
    l. 'state'
    m. 'zip_code'
    n. 'purpose'
    o. 'permit_owner_name'
    p. 'permit_type'
    q. 'council_district'
    r. 'mapped_location'

17. Select the short term rental permits issued in 2019 and save them to a data frame called **strp_2019.**

18. Count the number of short term rental permits issued per council district, similar to what you did in question 13. Call the new data frame **strp_2019_council.**

19. Add a new column to **building_permits_2019_council** and **strp_2019_council** called **'permit_type'**.
    a. Fill **'permit_type'** in **building_permits_2019_council** with **'building'**
    b. Fill **'permit_type'** in and **strp_2019_council** with **'short_term_rental'**

20. Concatenate **'building_permits_2019'** and **'strp_2019_council'**.

21. Use seaborn to make a grouped barplot to compare the number of building permits and short term rental permits for each council district in 2019. Are the number of building and short term rental permits related? Are there council districts that seem to have a particularly different ratio of building to short term rental permits? What other conclusions can you draw from the visualization?

22. The city council is concerned that with all the new construction projects and short term rental properties, fire stations may not be properly positioned to best handle fires. They give you access to a database with fire station information called **stations.db**.
    a. First execute the following code to see what tables are in the database:

```
db = <path to stations.db>
con = sql.connect(db)
mycursor = con.cursor()
mycursor.execute("SELECT name FROM sqlite_master WHERE
type='table' ORDER BY name;")
tables=(mycursor.fetchall())
print(tables)
```

    a. Load the fire stations data to a pandas data frame called **'fire_stations'**.

b. How many rows and columns are in this data frame?

23. Install the folium package. You can do this from a terminal or Anaconda prompt or by typing in your notebook. If you run it in your notebook you can run **!pip install folium** in a cell by itself (notice the exclamation point). If you run it from a terminal or Anaconda prompt run **pip install folium** (no exclamation point).

24. Delete the cell where you installed folium and add '**import folium**' to the top cell where your packages are imported.

25. Create a folium map of Nashville using [36.1612, -86.7775] as the location to center the map on. Experiment with different values for the **zoom_start** argument.

**Week Three**

26. Copy the function below into a cell and run that cell to make the function available for use. Next pass in the **building_permits_2019** and **strp_2019** dataframes *one at a time* to add latitude and longitude columns to each dataframe.

```
def add_lat_lng(df):
    lat_lng = pd.DataFrame(df['mapped_location'].apply(lambda s:
    s[s.find("(")+1:s.find(")")].split(', ')))
        lat_lng_df = lat_lng['mapped_location'].apply(pd.Series)
        lat_lng_df.columns = ['lat', 'lng']
        df = pd.concat([df, lat_lng_df], axis = 1)
        return df
```

27. In order to identify areas of greatest increased burden for the fire department, the city council asks you to identify high risk locations and plot the points on a map. To identify regions that represent the greatest increased burden on the fire department in 2020:

a. Subset **building_permits_2019** to only contain rows where the **'type_desc'** contains **'Residential - New'** and the **'subytpe_desc'** contains **'Multifamily'**.
b. Subset **strp_2019** to only contain rows where the **'status'** is **'ISSUED'**, **'subtype_desc'** contains **'Multifamily'**, and **'purpose'** contains **'Property is not Owner occupied'**.
c. It may be useful to drop duplicate entries in the **'lat'** and **'lng'** columns for both subsets.

28. Add points to the folium map by building a for-loop and using the iterrows() function to create point locations, and map each point as you loop through a dataframe. First add points from the subsetted **strp_2019**, then the subsetted **building_permits**, and lastly **fire_stations**. It may be useful to use the **icon** argument in the **folium.Marker()** function.

29. Identify potential areas of concern for the fire department in 2020 based on high concentrations of new building projects and/or short term rental properties.

30. The Nashville Fire Department heard about your analysis and wanted you to look at some data from 2018 and do some analysis on how satisfied residents were with the fire department in each council district based on a summary of survey responses. They are worried that council districts with fewer fire stations aren't as happy. They gave you access to another database with some information that you might find useful in your analysis called **multi_table.db**, as well as an Entity Relationship Diagram called **entity_relationship_diagram.png**.
    a. See what tables are in the database by modifying the code from **22.a** above to reference **multi_table.db**
    b. Look at **entity_relationship_diagram.png** in your **data** folder. Open it up to see how the different tables in the database are related and what steps you need to take to combine the information you want from the survey data with the fire station data.

31. Once you understand how the tables are related, use SQL (and python) to help the fire department understand how the residents of each council district felt about their performance in 2018. First you will need to figure out the stations associated with each zip code. Conceptually, the steps for that are:

a. Look at the ERD and decide which columns you ultimately want to get from the tables in the database. For example, you will want the zip code and the station number, but you may also want other columns as well. The columns will likely be from different tables, which is ok, but don't choose columns from more than 2 tables.
b. Once you know what columns you want, that will tell you which tables you will need to use.
c. To combine data across multiple tables you will need to perform a **JOIN**. There are multiple kinds of joins (see the Joining Dataframes slides). Decide which **JOIN** you want to do and the columns you will need from each table to combine the data. These columns should have the same information (both have council district data, for example). These columns don't need to be in the list you made in part **a**.
d. Now you have the information you will need to construct the query. First, you can use the list of columns you want (part **a**) to create the **SELECT** statement. The syntax for referencing a column is **<table_name>.<column_name>**. For example, you can indicate **table_1.zip_code, table_2.station**. Following this syntax, the **SELECT** statement could look something like:

    **SELECT table_1.zip_code, table_2.station**

e. Next, you will use one of the tables from part **b** to make the **FROM** statement. Since we are only performing 1 join, either table can go here. The **FROM** statement may look something like:

    **FROM table_1**

f. Lastly, you will make the **JOIN** statement. You will write out the kind of join you want to do, then the second table name, then **ON**, and lastly the columns you want to join on from the two tables, separated by an **=**. This statement may look like:

    **INNER JOIN table_2 ON table_1.district = table_2.council_district**

g. It would be good to save the completed query to its own variable as a string. The completed query may look like:

    query = '''
    **SELECT table_1.zip_code, table_2.station**
    **FROM table_1**
    **INNER JOIN table_2 ON table_1.district = table_2.council_district**
    '''

e. Execute this query using the pandas **pd.read_sql** function. You can verify the results are correct by querying each table individually and performing the merge in pandas and comparing the results.

32. You can expand on this query to include another **JOIN** or import any remaining table(s) into python and do the final merge there. If you add to your previous query, you would specify an additional **JOIN** as another statement in the query. For example:

```
SELECT table_1.council_district, table_2.station, table_3.fire_satisfaction
FROM table_1
INNER JOIN table_2 ON table_1.district = table_2.council_district
LEFT JOIN table_3 ON table_1.zip_code = table_3.zip
```

In the above example, two tables will be joined with **table_1** (since it is in the **FROM** statement). Then the columns specified in the **SELECT** statement will be returned from the resulting multi-merge table. Consider also including other keyword arguments such as **DISTINCT** and **ORDER BY** in your query to refine your results.

33. Execute this query and look at the resulting dataframe. Are residents generally happy or unhappy about the fire department? What are some areas where the fire department was successful? What are some areas where the fire department could improve?