***Scenario**: Metro Council has engaged your analytics consulting company to help the council understand if citizens are less satisfied in areas where adverse events are more likely to occur.*

You have been asked to explore both quantitative and qualitative datasets:

1. ***Police_calls_2018.csv**, a Nashville police department calls for service, dataset from 2018 that details where calls were made from (emergency and non-emergency calls)*
2. ***hubNashville_2018.csv**, hubNashville 311 service requests, data from 2018 that shows all requests for service made to hubNashville*
3. ***metro_survey.db**, a database containing the results of a 2018 resident satisfaction survey*

*Metro Council has asked your company to **prepare a brief (5-7 minutes) presentation of your findings to deliver to the city council and representatives from the mayor's office**. This guide will walk you through the initial data exploration and the steps needed to answer these specific questions:*

- *What kinds of police calls occurred most often in 2018?*
- *Where did these calls originate (which zipcodes)?*
- *In which months do calls occur most frequently?*
- *What kinds of requests for service are made to hubNashville?*
- *Are they being handled promptly?*

- *Are there particular kinds of requests in particular areas (zipcodes) that are especially problematic?*
- *How do the results of the 2018 survey align with what you observe in the police calls and hubNashville data? Are there any actionable insights from comparing the data? Are there any surprises?*
- *Are there any other findings you would like to share?*

Week One

- *Unzip the shared file to create a directory for your Analytics Jumpstart work. It should be called '**Analytics Jumpstart'**.*
- *Inside that directory, be sure there is a folder called '**data**'.*
- *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 '**jumpstart_analysis'**.*

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 the 2018 police calls data (police_calls_2018.csv to a dataframe called **police_calls**.
    a. Look at the first 5 rows.
    b. Look at the last 3 rows.

c. How many rows and columns does **police_calls** contain?

3. Keep just these columns:
    a. 'Call Received'
    b. 'Shift'
    c. 'Tencode'
    d. ‘'Tencode Description'
    e. 'Disposition Code'
    f. 'Disposition Description'
    g. 'Unit Dispatched'
    h. 'Sector'
    i. 'Zone'
    j. 'Latitude'
    k. 'Longitude'
    l. 'Zipcode'
    m. 'PO'

4. Rename the columns above:
    a. ‘call_time'
    b. 'shift'
    c. 'tencode'
    d. ‘'tencode_desc'
    e. 'disposition'
    f. 'disposition_desc'
    g. 'unit_dispatched'
    h. 'sector'
    i. 'zone'
    j. 'lat'
    k. 'lng'
    l. 'zipcode'
    m. 'po'

5. What are the unique disposition descriptions? How many are there?

6. Remove all rows from police_calls where the disposition description is missing (nan) or one of these two: DISREGARD / SIGNAL 9, NO RESPONSE. Check to see that you have 624841 rows remaining.

7. Create a dataframe from the **tencode_desc** value counts called **tencode_counts**. It should have two columns called **tencode** and **tencode_count**.

8. Create a seaborn barplot to show the **2018 Calls for Police Service by Tencode**. Set the figsize to 10 x 10 so you can see all of the data.

9. Find the counts of calls by **zipcode** and plot that. Zip codes look like numeric data, but they are actually descriptive data. Be sure to convert the **zipcode** column to a string before plotting. This will avoid having big gaps where there are numbers but no zip codes. Give the plot a meaningful title.

10. Convert the **call_time** column in **police_calls** to a pandas datetime. The code may take a few minutes to run. Create a new column in **police_calls** to show the month that a call for service occurred. In which month(s) did most calls occur? What do you notice about the months for which data is provided?

Week Two

11. Take a look at the 2018 hubNashville data by reading it into a DataFrame called **hub**.

12. Clean the **hub** column names to make everything lowercase and eliminate spaces so you can use dot-notation. Make the new names **'request_id'**, **'status'**, **'request_type'**, **'subrequest_type'**, **'add_subreqest_type'**, **'opened'**, **'closed'**, **'origin'**, **'zipcode'**, **'lat'**, **'lng'.**

13. Drop the rows from **hub** where the **closed_datetime** is missing. Then create a new column, **resolution_time** that calculates how long the request was open. You'll need to convert **opened** and **closed** to pandas datetimes before calculating the time delta.

14. Were any requests open for longer than a year? How many? Save these in a DataFrame called **slow_to_resolve**.

15. Aggregate the **slow_to_resolve** hubNashville requests by **zipcode** and **request_type**. You'll learn about the pandas groupby on Thursday of this week. For now, copy and paste this code into your notebook:

    ```
    grouped_slow_requests = slow_to_resolve.groupby(['zipcode', 'request_type'])['request_id'].agg('count').reset_index()
    ```

    What do you see?

16. Create a connection to the survey data **(metro_survey.db**) and then create a cursor in order to find all the available tables in the database. They should match the tables shown on the **metro_survey_ERD** diagram.

17. Let's check whether the people most satisfied with 'Police Overall' live in zip codes with a lot of calls for service or very few calls for service. Which do you think is true. The safety table has survey results that pertain to fire and police service and the info table has zip code and other information for survey respondents. Load each table to separate DataFrames (**safety_df** and **respondent_info**) and then merge them on Id to get the zip codes for respondents with their safety-related survey responses into a single DataFrame (**safety_exp1**). Check the shape of the DataFrame.

18. Next write a SQL SELECT statement to join the two tables and load them to a single pandas DataFrame (**safety_exp2**). Verify that **safety_exp1** is the same size as **safety_exp2.** If the DataFrames are different sizes, try to figure out why.

19. Use **safety_exp1**. Slice this DataFrame to get the **Id**, police related data ('**Crime Prevention**', '**Police - Overall**', '**Police Visibility**', '**Police Professionalism**', '**Police Response Time**') and **ZIP Code** columns. It's fine to save it back to the **safety_exp1** variable.

20. Group **safety_exp1** by the **Police - Overall** and **ZIP Code** columns and count the Id column. Create a DataFrame (**safety_grouped**) from this aggregation.

21. Compare this to your findings in #9 above. One approach you can take is to filter **safety_grouped** to look at just the 5 zipcodes where the most calls for police service were made.

22. Install the folium package. Type **!pip install folium** in a cell by itself (notice the exclamation point).

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

24. Construct 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.

25. Teams! You will be assigned to a team. Prepare for week 3 by brainstorming the things your team wants to explore, analyze, and present to the Metro Council. Choose a name for your consulting firm and create a Slack channel to collaborate over.

Week Three

26. Assign the folium map you created in step 24 to a variable **nash_map**. Write a for loop that makes use of the iterrows() method to:
    a. Create a location for every hubNashville request in the **slow_to_resolve** DataFrame.
    b. Create a popup that gives information about the request_type and resolution_time for each request.
    c. Create a marker using the folium Marker() constructor.
    d. Add the marker to **nash_map.**

    After you exit the for loop, you can simply call nash_map to display your map with the markers you created.

You may want to construct an icon using one of the Font Awesome icons (https://fontawesome.com/v4.7.0/icons/). You can pass that icon to the **icon** argument in the **folium.Marker()** function.

The syntax for creating an icon is:

icon = folium.Icon(color = <pick a color>, icon = <pick and icon>, prefix = 'fa')

Spend the rest of the week working with your team to complete your analysis and create your presentation!