

In [1]:

```
import importlib

def check_and_install(module_name):
    try:
        importlib.import_module(module_name)
        #print(f"{module_name} is already installed.")
    except ImportError:
        print(f"{module_name} is not installed. Installing...")
        import subprocess
        subprocess.check_call(["pip", "install", module_name])

# List of required modules
required_modules = [
    "seaborn",
    "numpy",
    "sqlalchemy",
    "matplotlib",
    "tabulate",
    "pandas",
    "ipywidgets",
    "plotly"
]

# Check and install each required module
for module in required_modules:
    check_and_install(module)

import plotly.graph_objs as go
from ipywidgets import interact, Dropdown
import ipywidgets as widgets
import sqlalchemy as sql
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tabulate import tabulate as ta
import seaborn as sns
import warnings
# fill in the details for your username and password below
with open("101-4_password.txt") as f:
    passw = f.read()
user_name = '101-4'

engine_string = "mysql+mysqlconnector://{}:{}@datasciencebd.ucalgary.ca/{}"
    user_name=user_name,
    passw=passw,
    db_name=user_name # Assuming database name is the same as user_name
)
engine = sql.create_engine(engine_string)
```

Calgary Community Crime Statistics

Group: Alan Cheun, Jianling Xie, Zane Wu

University Of Calgary DATA 604 Winter 2024: Working with Data at Scale

Abstract

This project focuses into the crime statistics of Calgary's communities from 2018 to 2023, aiming to discover the underlying patterns and trends with spatial and economic factors. We wrangled data from 11 datasets sourced from the Calgary City's Open Data Portal, which included detailed records on crime statistics alongside geo-locations of amenities and economic indicators.

Notable methods included the normalization of datasets, creation of database views for efficient query processing, and the development of a custom function to calculate geospatial distances, enhancing our ability to examine spatial distributions of crime rates effectively.

Our analyses revealed distinct geospatial patterns in community crime statistics, with significant variation in annual crime counts and rates among communities. The Center sector recorded the highest annual crime count, closely followed by the East sector. Within the Center Sector, the Beltline community consistently

reported the highest annual crime counts from 2018 to 2023. Furthermore, the Downtown Commercial Core community, also situated in the Center sector, exhibited the highest annual crime rate, attributed to its relatively high crime count and comparably low population. Theft from a vehicle emerged as the most common type of crime from 2018 to 2023, accounting for 32.8% to 38.6% of the annual total number of crimes. Over this period, the city's annual crime counts and rates experienced slight fluctuations, with the highest counts and rates recorded in 2019, followed by a decreasing trend. In terms of monthly variations, July to September were identified as having higher monthly crime counts compared to other months of the year.

We observe a clear correlation between various economic indicators and the crime rate. Specifically, the unemployment rate, the number of building permits issued, and property assessment values all show a relationship with the rate of crime. Additionally, proximity to CTrain stations emerged as a significant factor, with areas closer to these transit hubs experiencing different crime dynamics, which suggests targeted interventions could be beneficial. Economic trends, such as the relationship between higher property assessment values and lower crime rates in residential areas, further illustrate the complex interplay between urban development and crime patterns.

The results of this study offer insights into the dynamics of urban crime, providing a solid foundation for policymakers and law enforcement agencies to craft targeted interventions aimed at crime prevention and safety enhancement in urban settings. Further research is recommended to explore deeper into the causal relationships and to extend these findings with predictive modeling approaches. While the analysis provides statistical summaries, interpreting these results requires understanding the context of each community and considering additional factors beyond crime statistics.

Introduction

This project was chosen for its potential to shed light on the well-being and safety within Calgary communities, as reflected through crime statistics. Such data is not only indicative of social and economic disparities but also the efficacy of law enforcement strategies. The importance of analyzing crime statistics extends to assessing trends that can inform crime prevention and policy-making decisions.

On a personal note, as a daily user of Calgary Transit, there was a natural curiosity about the perception of crime and safety concerns, often amplified by news and social media surrounding Calgary Transit stations. With an eagerness to learn more about spatial data types and spatial functions, this project provided a means to explore whether economic factors, such as housing values, are negatively related to crime rates. Furthermore, we aimed to investigate if there's a correlation between construction activity and crime, addressing questions of urban development and public safety.

- Figure 1: City to invest \$1.9 million to improve safety in downtown Calgary [Source: Calgary Herald](#)
- Figure 2: Protests continue to put pressure on Calgary police resources; transit safety remains a focus [Source: Calgary Herald](#)
- Figure 3: Calgary police reveal further details about recent undercover CTrain drug trafficking operation [Source: CBC](#)

'Something's got to change': City to invest \$1.9 million to improve safety in downtown Calgary

Bill Kaufmann

The city says it will invest \$1.9 million for measures such as lighting and surveillance cameras to improve safety in Calgary's downtown.

CH calgaryherald

↗ X

<https://calgaryherald.com/news/local-news/protests-pressure-police-resources-transit-safety-focus/wcm/139e6c00-38f7-4903-a2b4-e73d4053b83d/amp/>



Protests continue to put pressure on Calga... X

Police Chief Mark Neufeld says the increasing "complexity and volatility" of protests and demonstrations has put...

calgaryherald.com

Calgary police reveal further details about recent undercover CTrain drug trafficking operation

Karina Zapata

Calgary police have revealed more details about a two-week undercover drug trafficking operation at CTrain stations late last year, as part of an ongoing...

CBC

↗ X

CBC

↗ X

Context

According to the police-reported crime statistics in Canada, as measured by the Crime Severity Index (CSI), crime increased by 5% from 2021 to 2022, reaching 5,668 incidents per 100,000 population. The Violent CSI also rose in 2022, reaching its highest point since 2007. The CSI index for Canada in 2022 was 78.1, with Calgary slightly below the national average at 75.2. However, two Alberta cities ranked high in the CSI, with Edmonton at 100.4 and Lethbridge at 119. Moreover, in a survey conducted in 2023 on perspectives of safety in Calgary, 41% of Calgarians thought crime has increased, and only 33% felt safe riding a C-Train alone after dark, down from 47% in 2022. Studies have shown that, social and economic disadvantage, such as unemployed or employed in low-paying, unskilled jobs were found to be strongly associated with crime. There is also an ecological theory postulates that crime will always display an uneven geographical distribution and that this variation is the result of the interrelationship between humans (or groups of

humans) and their surroundings. Thus, our study objective was to analyze the Calgary crime statistics across the communities to reveal any trends and patterns over the years from 2018 to 2023.

Datasets

We will retrieve the following datasets from the City of Calgary's Open Data Portal.

The crime statistics data from year 2018 to 2023 were stored in the community crime statistics dataset in .csv file format, and contained the cumulative crime counts reported by community and by the date of the crime occurred. The crime count was based on the most serious violation (MSV) per incident and were categorized as assault (non-domestic), break and entry-commercial, break and entry-dwelling, break and entry-other premise, commercial robbery, street robbery, theft from vehicle, theft of vehicle, and violence (non-domestic). There were 6,7262 records of crime in total in the dataset. The other datasets were measured on the community level and by date, therefore they can be combined to the community crime statistics dataset using the unique community identity and by the date of assessment. The datasets from the City of Calgary's Open Data Portal are licensed by Open Government License, the information provider, City of Calgary grants us a worldwide, royalty-free, perpetual, non-exclusive license to use the Information, including for commercial purposes, subject to the terms the license.

Data Accessed during the month of March 2024.

| Dataset | Link | Key Columns | Person Responsible |
|---|----------------------|--|--------------------|
| Unemployment Rates | Link | Date, Unemployment Rate | Zane Wu |
| Community Crime Statistics | Link | Date, Community Code, ID | Jianling Xie |
| Building Permits | Link | Permit Number, Issued Date, Community Code | Zane Wu |
| Historical Property Assessments (Parcel) | Link | Roll Number, Assessed Value, Roll Year | Zane Wu |
| Civic Census by Community | Link | Year, Community Code | Alan Cheun |
| Transit Light-Rail Transit (LRT) Stations | Link | Station Name, LRT Point | Alan Cheun |
| Police Service Locations | Link | Station Name, Location Point | Alan Cheun |
| Community Points | Link | Community Code, Community Point | Alan Cheun |
| Civic Census by Community, Age and Gender | Link | Census Year, Community Code | Alan Cheun |
| Green Line LRT | Link | Station Name, LRT Point | Alan Cheun |
| Community Boundaries | Link | Community Code, Boundary (polygon) | Alan Cheun |

Permission to Use Data

Open Government License: "City of Calgary's Open Data Portal." [Link](#)

Data Limitations and Assumptions

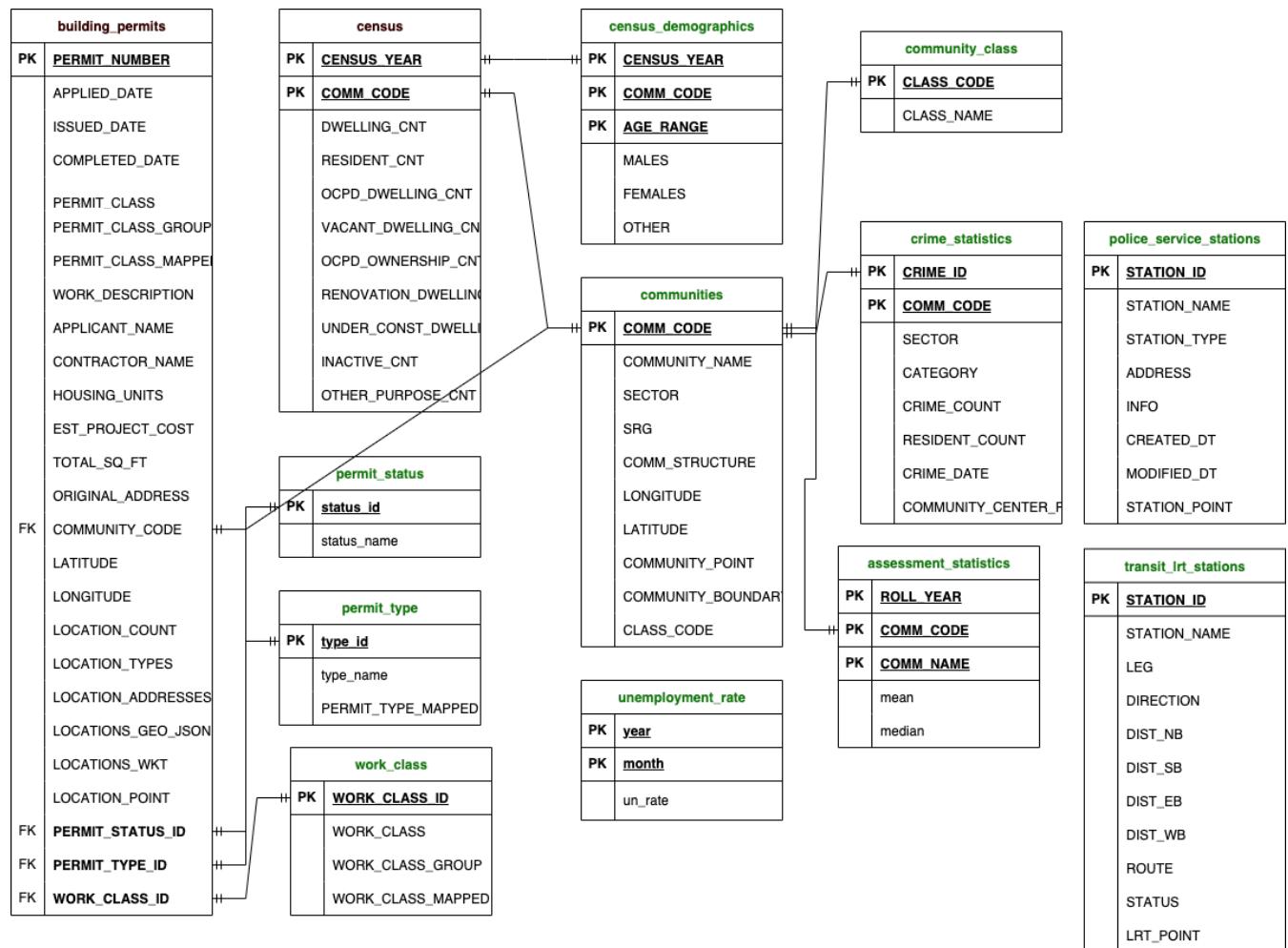
In the analysis of crime data, several limitations were encountered that may have affected the depth and scope of the study. Firstly, the geospatial data provided represented community locations rather than specific crime scenes, which could lead to a generalization of data points rather than precise positioning. Furthermore, crime categories were broadly defined, lacking in more nuanced classifications that might have allowed for a detailed analysis of crime patterns. Notably, demographic data for the perpetrators were not available, which restricted the ability to perform demographic-based analysis to understand correlations between perpetrator characteristics and crime types. Additionally, some of the economic data were not categorized by community, which limited the ability to conduct localized economic impact analyses. These limitations underscore the need for more granular data to enhance the accuracy and relevance of future analyses.

DATA Preparation, Cleaning and Wrangling

What was each individual dataset, why did you choose it, and what did you learn from it?

Include in this section a discussion of any steps you had to take to clean or extract the data

you were working on. Ideally, this will be a summary/compilation of work you have already done, for the individual data exploration



Data Wrangling

The datasets have been cleansed and adjusted to maintain high data quality, ensuring uniformity across datasets and simplifying the analysis process. Detailed descriptions of the data cleaning and adjustment methods used for each dataset are provided below. Additionally, we have normalized the datasets and introduced primary keys to enhance data integrity and optimize query performance.

Limitations: We did not enforce referential integrity with Foreign Keys, there were instances where attempting to apply a Foreign Key would not work due to missing values, and we could not derive a value nor want to drop the row due to the missing value.

Communities

The Communities table is the focal point to our analysis. You will see it be the connective node to the other tables as we bring in different measures all linking to the Community or filtered by the Community's spatial data, either joining on Community Code or by spatial functions with LRT Transit Station, Police Station, Crime Statistics' location data.

The Community Code (COMM_CODE) was designated as our primary key, this is the unique identifier to each row and it will be one of the ways we join to other datasets. The other way will be the Community Point (COMMUNITY_POINT) and Community Boundary (COMMUNITY_BOUNDARY), these are of data type GEOMETRY.

Limitations:

- The Communities Point data export seems to consist of a master data list of Communities that are current, we can see communities and community codes that appear in other datasets either due to it only existing in the past, or has not been updated by Open Data Calgary to be included in their dataset. This may cause us to filter out communities that are not part of the Community Point dataset.

Cleaning:

- Merge the two datasets Community Points and Community Boundaries (this was accomplished via Python)

- Make COMM_CODE a primary key
- Make Point and Polygon geometry data types
- Rename Point and Polygon to COMMUNITY_POINT and COMMUNITY_POLYGON to avoid keywords as column names

Create Community Table

- [D604_CREATE_COMMUNITIES_TABLE.ipynb](#)

Load Community Table

- [D604_LOAD_COMMUNITIES_TABLE.ipynb](#)

```
In [2]: query_table = pd.read_sql_query("""
SELECT ORDINAL_POSITION, COLUMN_NAME, COLUMN_KEY, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'communities';
""", engine)
print("Communities - SQL Table")
query_table
```

Communities - SQL Table

| | ORDINAL_POSITION | COLUMN_NAME | COLUMN_KEY | DATA_TYPE |
|---|------------------|--------------------|------------|-----------|
| 0 | 1 | COMM_CODE | PRI | varchar |
| 1 | 2 | CLASS_CODE | | int |
| 2 | 3 | COMMUNITY_NAME | | varchar |
| 3 | 4 | SECTOR | | varchar |
| 4 | 5 | SRG | | varchar |
| 5 | 6 | COMM_STRUCTURE | | varchar |
| 6 | 7 | LONGITUDE | | double |
| 7 | 8 | LATITUDE | | double |
| 8 | 9 | COMMUNITY_POINT | | geometry |
| 9 | 10 | COMMUNITY_BOUNDARY | | geometry |

Community Class

This is table that is a product of normalization from the Communities export from Open Data Calgary.

```
In [3]: query_table = pd.read_sql_query("""
SELECT ORDINAL_POSITION, COLUMN_NAME, COLUMN_KEY, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'community_class';
""", engine)
print("Community Class - SQL Table")
query_table
```

Community Class - SQL Table

| | ORDINAL_POSITION | COLUMN_NAME | COLUMN_KEY | DATA_TYPE |
|---|------------------|-------------|------------|-----------|
| 0 | 1 | CLASS_CODE | PRI | int |
| 1 | 2 | CLASS_NAME | | varchar |

Crime Statistics

This is our primary table for exploratory data analysis, it is our key measure for crime count and thus deriving crime rate. This table will be joined via values of the Community Code (COMM_CODE) and the Year of the Crime Date fields in addition to spatial functions focused on the Community Center Point (COMMUNITY_CENTER_POINT).

Cleaning:

- Resident Count - strip the comma, so it only contains numeric characters
- Replace Community Name, with Community Code

- Format the date field to datetime format for MariaDB - YYYY-MM
- Convert (Community) Name to Community Code, this is for normalization for better database performance
- Make ID a primary Key and rename to CRIME_ID
- Convert Date to format '%Y/%m' and rename to Crime Date to avoid keyword as column name
- uppercase and _ in column name for consistency
- where we have missing location data such as Sector or Community Point, try to compensate with our Community Point data from a previously loaded dataset

Create Crime Statistics Table

- [D604_CREATE_CRIMES_TABLE.ipynb](#)

Load Crime Statistics Table

- [D604_LOAD_CRIMES_TABLE.ipynb](#)

```
In [4]: query_table = pd.read_sql_query("""
SELECT ORDINAL_POSITION, COLUMN_NAME, COLUMN_KEY, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'crime_statistics';
""", engine)
print("Crime Statistics - SQL Table")
query_table
```

Crime Statistics - SQL Table

| | ORDINAL_POSITION | COLUMN_NAME | COLUMN_KEY | DATA_TYPE |
|---|------------------|------------------------|------------|-----------|
| 0 | 1 | CRIME_ID | PRI | varchar |
| 1 | 2 | SECTOR | | varchar |
| 2 | 3 | COMM_CODE | MUL | varchar |
| 3 | 4 | CATEGORY | | varchar |
| 4 | 5 | CRIME_COUNT | | int |
| 5 | 6 | RESIDENT_COUNT | | int |
| 6 | 7 | CRIME_DATE | | date |
| 7 | 8 | COMMUNITY_CENTER_POINT | | geometry |

Census

This table was to provide us a resident count for calculating our crime rate, in the end we did not use this dataset because not every community was represented in this census that we needed.

Cleaning:

- Each of the Measures need to be converted from text to numeric, stripped of comma
- Set primary key for CENSUS_YEAR, COMM_CODE

Create Census Demographics Table

- [D604_CREATE_CENSUS_TABLE.ipynb](#)

Load Census Demographics Table

- [DD604_LOAD_SQL_CENSUS.ipynb](#)

```
In [5]: query_table = pd.read_sql_query("""
SELECT ORDINAL_POSITION, COLUMN_NAME, COLUMN_KEY, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'census';
""", engine)
print("Census - SQL Table")
query_table
```

Census - SQL Table

| | ORDINAL_POSITION | COLUMN_NAME | COLUMN_KEY | DATA_TYPE |
|----|------------------|--------------------------|------------|-----------|
| 0 | 1 | CENSUS_YEAR | PRI | int |
| 1 | 2 | COMM_CODE | PRI | varchar |
| 2 | 3 | DWELLING_CNT | | int |
| 3 | 4 | RESIDENT_CNT | | int |
| 4 | 5 | OCPD_DWELLING_CNT | | int |
| 5 | 6 | VACANT_DWELLING_CNT | | int |
| 6 | 7 | OCPD_OWNERSHIP_CNT | | int |
| 7 | 8 | RENOVATION_DWELLING_CNT | | int |
| 8 | 9 | UNDER_CONST_DWELLING_CNT | | int |
| 9 | 10 | INACTIVE_CNT | | int |
| 10 | 11 | OTHER_PURPOSE_CNT | | int |

Census Demographics

This table was providing extra insight to each community in terms of gender composition and age brackets. This table will join to other tables by Community Code (COMM_CODE) and by Census Year (CENSUS_YEAR).

Cleaning:

- check all measure columns and ensure they are integer, remove ',' and convert to INTEGER- rename YEAR to CENSUS_YEAR
- set the CENSUS_YEAR, COMM_CODE, AGE_RANGE as primary keys
- this data export had multiple entries per CENSUS_YEAR, COMM_CODE and AGE_RANGE, so before inserting into SQL table, group by primary key and sum the MALES, FEMALES, OTHER counts

Create Census Demographics Table

- [D604_CREATE_CENSUS_DEMOGRAPHICS_TABLE.ipynb](#)

Load Census Demographics Table

- [D604_LOAD_SQL_CENSUS_DEMOGRAPHICS.ipynb](#)

```
In [6]: query_table = pd.read_sql_query("""
SELECT ORDINAL_POSITION, COLUMN_NAME, COLUMN_KEY, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'census_demographics';
""", engine)
print("Census Demographics - SQL Table")
query_table
```

Census Demographics - SQL Table

| | ORDINAL_POSITION | COLUMN_NAME | COLUMN_KEY | DATA_TYPE |
|---|------------------|-------------|------------|-----------|
| 0 | 1 | CENSUS_YEAR | PRI | int |
| 1 | 2 | COMM_CODE | PRI | varchar |
| 2 | 3 | AGE_RANGE | PRI | varchar |
| 3 | 4 | MALES | | int |
| 4 | 5 | FEMALES | | int |
| 5 | 6 | OTHER | | int |

Transit LRT Stations

The LRT transit station was a significant dataset to our analysis as it is a variable we wanted to examine against Calgary Crime Rate. This table will be joined by spatial functions such as CONTAINS or calculating distance on the location point (LRT_POINT).

Cleaning:

- Make the_geom a GEOMETRY data type, rename to LRT_POINT so it is more meaningful
- review names, set to a naming standard - change to uppercase so it is consistent
 - rename STATIONNAM to STATION_NAME
- primary key has to be created as STATION_ID and auto-generated number, Open Data Calgary or Calgary Transit website does not provide a unique identifier for each LRT Station

Create Transit LRT Stations Table

- [D604_CREATE_LRT_TABLE.ipynb](#)

Load Transit LRT Stations Table

- [D604_LOAD_SQL_LRT.ipynb](#)

```
In [7]: query_table = pd.read_sql_query("""
SELECT ORDINAL_POSITION, COLUMN_NAME, COLUMN_KEY, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'transit_lrt_stations';
""", engine)
print("Transit LRT Stations - SQL Table")
query_table
```

Transit LRT Stations - SQL Table

| | ORDINAL_POSITION | COLUMN_NAME | COLUMN_KEY | DATA_TYPE |
|----|------------------|--------------|------------|-----------|
| 0 | 1 | STATION_ID | PRI | int |
| 1 | 2 | STATION_NAME | | text |
| 2 | 3 | LEG | | text |
| 3 | 4 | DIRECTION | | text |
| 4 | 5 | DIST_NB | | text |
| 5 | 6 | DIST_SB | | text |
| 6 | 7 | DIST_EB | | text |
| 7 | 8 | DIST_WB | | text |
| 8 | 9 | ROUTE | | text |
| 9 | 10 | STATUS | | text |
| 10 | 11 | LRT_POINT | | geometry |

Building Permits

This table was joined to other tables by Community Code (COMMUNITY_CODE) and by the Year of the Issued Date.

Cleaning:

- Permit Number is set to Primary Key
- Review names, set to a naming standard - change to uppercase so it is consistent
- Set Location Point to geometry data type, POINT

Load Building Permit table

- [D604_LOAD_BUILDING_PERMITS_TABLE.ipynb](#)

Limitations:

- We could not normalize Permit Class out of this table, there appears to be some data quality issue, where you would expect to normalize Permit Class as a unique entry, then Permit Class Group would be a categorical field, then Permit Class Mapped could be an attribute of Permit Class.
- There were other areas where it looked like Permit Class had leading space or typo, so we determined it was best to just leave it in this Building Permit table and treat as free-text fields given our lack of visibility into the source data.

```
In [8]: query_table = pd.read_sql_query("""
SELECT PERMIT_CLASS, COUNT(DISTINCT PERMIT_CLASS_GROUP) AS PermitClassGroupCount
FROM building_permits
GROUP BY PERMIT_CLASS
HAVING COUNT(DISTINCT PERMIT_CLASS_GROUP) > 1
ORDER BY PERMIT_CLASS;
""", engine)
print("Data Quality Issue with Permit Class from Building Permits export - 1")
query_table
```

Data Quality Issue with Permit Class from Building Permits export - 1

```
Out[8]:
```

| | PERMIT_CLASS | PermitClassGroupCount |
|--|--------------|-----------------------|
|--|--------------|-----------------------|

| | | |
|---|--------------------------------------|---|
| 0 | 1113 - Detached Garage Dwelling Unit | 2 |
|---|--------------------------------------|---|

```
In [9]: query_table = pd.read_sql_query("""
SELECT DISTINCT PERMIT_CLASS, PERMIT_CLASS_GROUP, PERMIT_CLASS_MAPPED FROM BUILDING_PERMITS
WHERE PERMIT_CLASS = '1113 - Detached Garage Dwelling Unit'
ORDER BY 1,2,3
""", engine)
print("Data Quality Issue with Permit Class from Building Permits export - 2")
query_table
```

Data Quality Issue with Permit Class from Building Permits export - 2

```
Out[9]:
```

| | PERMIT_CLASS | PERMIT_CLASS_GROUP | PERMIT_CLASS_MAPPED |
|--|--------------|--------------------|---------------------|
|--|--------------|--------------------|---------------------|

| | | | |
|---|--------------------------------------|---------------------|-------------|
| 0 | 1113 - Detached Garage Dwelling Unit | Additional Dwelling | Residential |
|---|--------------------------------------|---------------------|-------------|

| | | | |
|---|--------------------------------------|---------------|-------------|
| 1 | 1113 - Detached Garage Dwelling Unit | Single Family | Residential |
|---|--------------------------------------|---------------|-------------|

```
In [10]: query_table = pd.read_sql_query("""
SELECT ORDINAL_POSITION, COLUMN_NAME, COLUMN_KEY, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'building_permits';
""", engine)
print("Building Permits - SQL Table")
query_table
```

Building Permits - SQL Table

Out[10]:

| | ORDINAL_POSITION | COLUMN_NAME | COLUMN_KEY | DATA_TYPE |
|----|------------------|---------------------|------------|-----------|
| 0 | 1 | PERMIT_NUMBER | PRI | text |
| 1 | 2 | APPLIED_DATE | | date |
| 2 | 3 | ISSUED_DATE | | date |
| 3 | 4 | COMPLETED_DATE | | date |
| 4 | 5 | PERMIT_CLASS | | text |
| 5 | 6 | PERMIT_CLASS_GROUP | | text |
| 6 | 7 | PERMIT_CLASS_MAPPED | | text |
| 7 | 8 | WORK_DESCRIPTION | | text |
| 8 | 9 | APPLICANT_NAME | | text |
| 9 | 10 | CONTRACTOR_NAME | | text |
| 10 | 11 | HOUSING_UNITS | | bigint |
| 11 | 12 | EST_PROJECT_COST | | double |
| 12 | 13 | TOTAL_SQ_FT | | double |
| 13 | 14 | ORIGINAL_ADDRESS | | text |
| 14 | 15 | COMMUNITY_CODE | | text |
| 15 | 16 | LATITUDE | | double |
| 16 | 17 | LONGITUDE | | double |
| 17 | 18 | LOCATION_COUNT | | double |
| 18 | 19 | LOCATION_TYPES | | text |
| 19 | 20 | LOCATION_ADDRESSES | | text |
| 20 | 21 | LOCATIONS_GEO_JSON | | text |
| 21 | 22 | LOCATIONS_WKT | | text |
| 22 | 23 | LOCATION_POINT | | point |
| 23 | 24 | PERMIT_STATUS_ID | MUL | int |
| 24 | 25 | PERMIT_TYPE_ID | MUL | int |
| 25 | 26 | WORK_CLASS_ID | | int |

Permit Status

This is table that is a product of normalization from the Building Permits export from Open Data Calgary.

In [11]:

```
query_table = pd.read_sql_query("""
SELECT ORDINAL_POSITION, COLUMN_NAME, COLUMN_KEY, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'permit_status';
""", engine)
print("Permit Status - SQL Table")
query_table
```

Permit Status - SQL Table

Out[11]:

| | ORDINAL_POSITION | COLUMN_NAME | COLUMN_KEY | DATA_TYPE |
|---|------------------|-------------|------------|-----------|
| 0 | 1 | STATUS_ID | PRI | int |
| 1 | 2 | STATUS_NAME | | varchar |

Permit Type

This is table that is a product of normalization from the Building Permits export from Open Data Calgary.

In [12]:

```
query_table = pd.read_sql_query("""
SELECT ORDINAL_POSITION, COLUMN_NAME, COLUMN_KEY, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'permit_type';
""", engine)
```

```
print("Permit Type - SQL Table")
query_table
```

Permit Type - SQL Table

| ORDINAL_POSITION | COLUMN_NAME | COLUMN_KEY | DATA_TYPE |
|------------------|-------------|--------------------|-----------|
| 0 | 1 | TYPE_ID | PRI int |
| 1 | 2 | TYPE_NAME | varchar |
| 2 | 3 | PERMIT_TYPE_MAPPED | varchar |

Work Class

This is table that is a product of normalization from the Building Permits export from Open Data Calgary.

```
In [13]: query_table = pd.read_sql_query("""
SELECT ORDINAL_POSITION, COLUMN_NAME, COLUMN_KEY, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'work_class';
""", engine)
print("Permit Type - SQL Table")
query_table
```

Permit Type - SQL Table

| ORDINAL_POSITION | COLUMN_NAME | COLUMN_KEY | DATA_TYPE |
|------------------|-------------|-------------------|-----------|
| 0 | 1 | WORK_CLASS_ID | PRI int |
| 1 | 2 | WORK_CLASS | text |
| 2 | 3 | WORK_CLASS_GROUP | text |
| 3 | 4 | WORK_CLASS_MAPPED | text |

Assessment Statistics

This is a table that we built as an aggregated data table from Python code to store calculations that would be difficult to produce with SQL query and would save on query performance by having statistical measures like Median and Mean already stored for the Year and Community.

Cleaning:

- ROLL_YEAR, ROLL_NUMBER ARE set to Primary Key

Load Assessment Statistics table

- [D604_LOAD_ASSESSMENT_STATISTICS_TABLE.ipynb](#)

```
In [14]: query_table = pd.read_sql_query("""
SELECT ORDINAL_POSITION, COLUMN_NAME, COLUMN_KEY, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'assessment_statistics';
""", engine)
print("Assessment Statistics - SQL Table")
query_table
```

Assessment Statistics - SQL Table

| ORDINAL_POSITION | COLUMN_NAME | COLUMN_KEY | DATA_TYPE |
|------------------|-------------|------------|-------------|
| 0 | 1 | ROLL_YEAR | PRI bigint |
| 1 | 2 | COMM_CODE | PRI varchar |
| 2 | 3 | MEAN | double |
| 3 | 4 | MEDIAN | double |

Unemployment Rate

This table was generated in sql based on an original table which included the unemployment rate of calgary.

Cleaning:

- Year and Month are set to Primary Keys
- change the columns name so it is easier to utilize.

Create and Load unemployment rate table

- [D604_CREATE_LOAD_UNEMPLOYMENT_RATE.ipynb](#)

```
In [15]: query_table = pd.read_sql_query("""
SELECT ORDINAL_POSITION, COLUMN_NAME, COLUMN_KEY, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'unemployment_rate';
""", engine)
print("Unemployment_rate - SQL Table")
query_table
```

Unemployment_rate - SQL Table

| | ORDINAL_POSITION | COLUMN_NAME | COLUMN_KEY | DATA_TYPE |
|---|------------------|-------------|------------|-----------|
| 0 | 1 | year | PRI | int |
| 1 | 2 | month | PRI | int |
| 2 | 3 | un_rate | | decimal |

Exploratory Data Analysis (EDA)

As a team, create a set of queries which draw data from the datasets chosen by multiple team members. The dataset of each team member should be used in conjunction with one or more datasets from more than one team member. The queries should tell you something about your datasets together, which queried alone, would not be obvious or possible to extract from your dataset. What did you learn from the total collection of your chosen datasets? Discuss the steps you took to combine your datasets. Show your results, and the work you did to produce your results . This section is designed to facilitate visual exploration of the data to enhance familiarity with the dataset. Guiding questions for exploratory data analysis (EDA) are provided to direct this exploratio.

For Each Community - How Many LRT stations Are Within Its Community Boundaries?

This query is showcasing aggregation of number of LRT Transit Stations per Community, WHERE clause taking advantage of spatial function CONTAINS where we have GEOMETRY POINT datatype that belong to Transit LRT stations and we have Community GEOMETRY MULTIPOLYGON datatype. This query will play a significant part of telling a story of the association of the location and frequency of LRT Stations to the Community and any pattern with Crime Statistics.

```
In [16]: query_table = pd.read_sql_query("""
SELECT
    community.comm_code AS "Community Code",
    community.COMMUNITY_NAME AS "Community Name",
    COUNT(transit.STATION_NAME) AS "Number of Transit Stations"
FROM
    transit_lrt_stations transit,
    communities community
WHERE
    Contains(community.COMMUNITY_BOUNDARY, transit.LRT_POINT) = 1
GROUP BY
    community.comm_code,
    community.COMMUNITY_NAME
ORDER BY
    COUNT(transit.STATION_NAME) DESC
""", engine)
# Ensure that your plotting library is set to display graphs inline in a Jupyter Notebook
%matplotlib inline

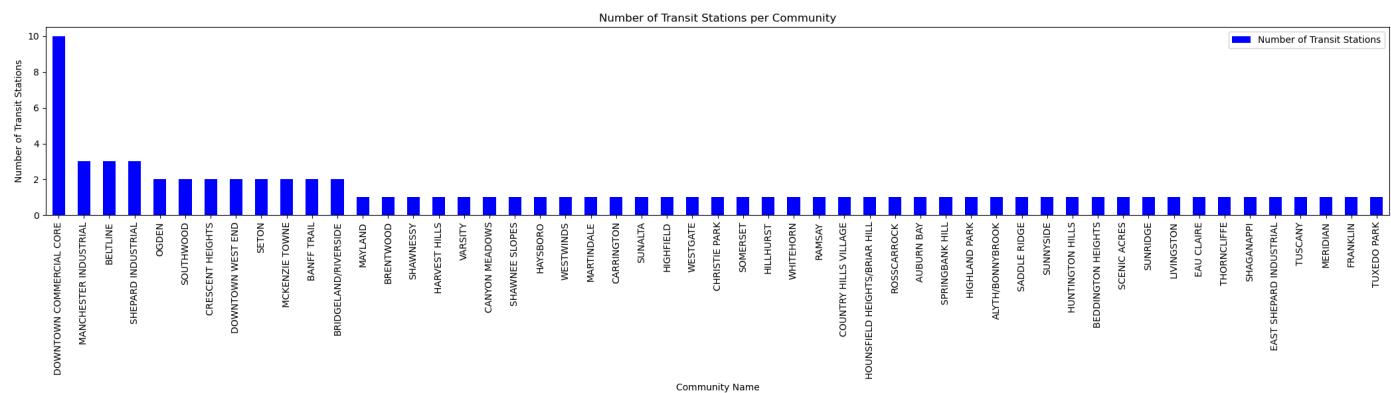
# Plotting the bar graph
query_table.plot(kind='bar', x='Community Name', y='Number of Transit Stations', color='blue', f
```

```

plt.title('Number of Transit Stations per Community')
plt.xlabel('Community Name')
plt.ylabel('Number of Transit Stations')
plt.xticks(rotation=90) # Rotate x-axis Labels for better readability if needed
plt.tight_layout() # Adjusts plot to ensure everything fits without overlap

# Show the plot
plt.show()

```



For Each Community What Are the Closest LRT Station and Closest Police Station?

This query is showcasing creating a VIEW object to query for shortest distances from a community point to an LRT Transit Station and to a Police Station, it is created as a VIEW because we knew we will need this query again in future. As a VIEW it is abstracted and keeps the SELECT query a little cleaner to look at and one team member can update or fix it while other team members can reference it without constantly needing to change their respective SQL query.

This structure for querying data will play a significant part of telling a story of each community and the proximity to an LRT station.

Custom SQL Function - HAVERSINE_DISTANCE_GEO This SQL function provides the distance in meters between two GEOMETRY POINT data types. We needed a tangible measure of distance between a crime and an LRT station which the ST_DISTANCE within our version of MariaDB only provides in degrees of the Earth's radius, we do not have ST_DISTANCE_SPHERE which would return an output in meters. Therfore a custom functio was created to allow us to use the GEOMETRY POINTs in our SQL tables and calculate the distance in meters. This will have great importance when we want to query and visualize distances as spatial data is a focus of our analysis.

```

DELIMITER $$
CREATE DEFINER=`L01-4`@`%` FUNCTION `HAVERSINE_DISTANCE_GEO`(point1 GEOMETRY,
    point2 GEOMETRY
) RETURNS float
    DETERMINISTIC
BEGIN
    DECLARE Lat1 FLOAT;
    DECLARE Lon1 FLOAT;
    DECLARE Lat2 FLOAT;
    DECLARE Lon2 FLOAT;
    DECLARE R INT DEFAULT 6371000; -- Earth radius in meters
    DECLARE phi1 FLOAT;
    DECLARE phi2 FLOAT;
    DECLARE delta_phi FLOAT;
    DECLARE delta_lambda FLOAT;
    DECLARE a FLOAT;
    DECLARE c FLOAT;
    DECLARE d FLOAT;

    -- Extract lat and lon from the points
    SET Lon1 = ST_X(point1);
    SET Lat1 = ST_Y(point1);
    SET Lon2 = ST_X(point2);
    SET Lat2 = ST_Y(point2);

    -- Convert degrees to radians

```

```

SET phi1 = radians(lat1);
SET phi2 = radians(lat2);
SET delta_phi = radians(lat2 - lat1);
SET delta_lambda = radians(lon2 - lon1);

-- Haversine formula
SET a = sin(delta_phi / 2) * sin(delta_phi / 2) +
        cos(phi1) * cos(phi2) *
        sin(delta_lambda / 2) * sin(delta_lambda / 2);
SET c = 2 * atan2(sqrt(a), sqrt(1-a));
SET d = R * c;

RETURN d;
END$$
DELIMITER ;

```

```

In [17]: print("Create VIEW vw_community_distances AS ")
query_table = pd.read_sql_query("""
WITH
    lrt_shortest_distances AS (
        SELECT
            comm.COMM_CODE AS COMM_CODE,
            lrt.STATION_ID AS LRT_STATION_ID,
            HAVERSINE_DISTANCE_GEO(comm.COMMUNITY_POINT, lrt.LRT_POINT) AS DISTANCE_TO_LRT_METER
        ROW_NUMBER() OVER (
            PARTITION BY comm.COMM_CODE
            ORDER BY HAVERSINE_DISTANCE_GEO(comm.COMMUNITY_POINT, lrt.LRT_POINT)
        ) AS rn
    FROM
        communities comm
        JOIN transit_lrt_stations lrt ON (lrt.STATUS = 'Current')
    ),
    police_shortest_distances AS (
        SELECT
            comm.COMM_CODE AS COMM_CODE,
            police.STATION_ID AS POLICE_STATION_ID,
            HAVERSINE_DISTANCE_GEO(comm.COMMUNITY_POINT, police.STATION_POINT) AS DISTANCE_TO_POLICE_METER
        ROW_NUMBER() OVER (
            PARTITION BY comm.COMM_CODE
            ORDER BY HAVERSINE_DISTANCE_GEO(comm.COMMUNITY_POINT, police.STATION_POINT)
        ) AS rn
    FROM
        communities comm
        JOIN police_service_stations police
    )
SELECT
    c.COMM_CODE AS COMM_CODE,
    c.COMMUNITY_NAME AS COMMUNITY_NAME,
    lrt.LRT_STATION_ID AS NEAREST_LRT_STATION_ID,
    lrt.DISTANCE_TO_LRT_METERS AS SHORTEST_DISTANCE_TO_LRT_METERS,
    police.POLICE_STATION_ID AS NEAREST_POLICE_STATION_ID,
    police.DISTANCE_TO_POLICE_METERS AS SHORTEST_DISTANCE_TO_POLICE_METERS
FROM
    communities c
    LEFT JOIN lrt_shortest_distances lrt ON (c.COMM_CODE = lrt.COMM_CODE AND lrt.rn = 1)
    LEFT JOIN police_shortest_distances police ON (c.COMM_CODE = police.COMM_CODE AND police.rn
    """", engine)
query_table.tail(20)

```

Create VIEW vw_community_distances AS

Out[17]:

| | COMM_CODE | COMMUNITY_NAME | NEAREST_LRT_STATION_ID | SHORTEST_DISTANCE_TO_LRT_METERS | NEARE |
|-----|-----------|---------------------------|------------------------|---------------------------------|----------|
| 290 | UNI | UNIVERSITY HEIGHTS | 34 | | 1227.030 |
| 291 | UOC | UNIVERSITY OF CALGARY | 31 | | 740.807 |
| 292 | VAF | VALLEYFIELD | 17 | | 3835.500 |
| 293 | VAL | VALLEY RIDGE | 41 | | 4330.820 |
| 294 | VAR | VARSITY | 5 | | 815.095 |
| 295 | VIS | VISTA HEIGHTS | 36 | | 2551.400 |
| 296 | WAL | WALDEN | 43 | | 4085.290 |
| 297 | WBN | WOODBINE | 25 | | 4120.510 |
| 298 | WES | WESTWINDS | 44 | | 758.207 |
| 299 | WGT | WESTGATE | 1 | | 757.202 |
| 300 | WHI | WHITEHORN | 20 | | 873.487 |
| 301 | WHL | WEST HILLHURST | 14 | | 1371.120 |
| 302 | WIL | WILLOW PARK | 25 | | 1308.580 |
| 303 | WIN | WINSTON HEIGHTS/MOUNTVIEW | 9 | | 2921.000 |
| 304 | WLD | WILDWOOD | 1 | | 2276.290 |
| 305 | WND | WINDSOR PARK | 26 | | 1542.530 |
| 306 | WOO | WOODLANDS | 25 | | 2549.980 |
| 307 | WSP | WEST SPRINGS | 12 | | 2777.680 |
| 308 | WWO | WOLF WILLOW | 43 | | 5280.180 |
| 309 | YKV | YORKVILLE | 43 | | 3234.560 |

What Type of Crimes Occur Within 1km of the University of Calgary?

We have established that the latitude and longitude of the University of Calgary is -114.136719, 51.078621

We wonder what sort of crimes occur within 1km of a specific location.

In [18]:

```
query_table = pd.read_sql_query("""
WITH CrimeData AS (
    SELECT
        YEAR(`101-4`.`crime_statistics`.`CRIME_DATE`) AS `Year`,
        `101-4`.`crime_statistics`.`COMM_CODE` AS `COMM_CODE`,
        `101-4`.`crime_statistics`.`CATEGORY` AS `Crime_Type`,
        SUM(`101-4`.`crime_statistics`.`CRIME_COUNT`) AS `Crime_Count`
    FROM
        `101-4`.`crime_statistics`
    WHERE
        `haversine_distance_geo`(
            ST_GeometryFromText(CONCAT('POINT(', -114.136719, ' ', 51.078621, ')')),
            `101-4`.`crime_statistics`.`COMMUNITY_CENTER_POINT`
        ) <= 1000
    GROUP BY
        YEAR(`101-4`.`crime_statistics`.`CRIME_DATE`),
        `101-4`.`crime_statistics`.`COMM_CODE`,
        `101-4`.`crime_statistics`.`CATEGORY`
)
SELECT
    `cd`.`Year` AS `Year`,
    `cd`.`COMM_CODE` AS `Comm_Code`,
    `c`.`COMMUNITY_NAME` AS `Community_Name`,
    `cd`.`Crime_Type` AS `Crime_Type`,
    `cd`.`Crime_Count` AS `Crime_Count`
FROM
    `CrimeData` `cd`
    LEFT JOIN `101-4`.`communities` `c` ON (`cd`.`COMM_CODE` = `c`.`COMM_CODE`)
```

```

UNION ALL

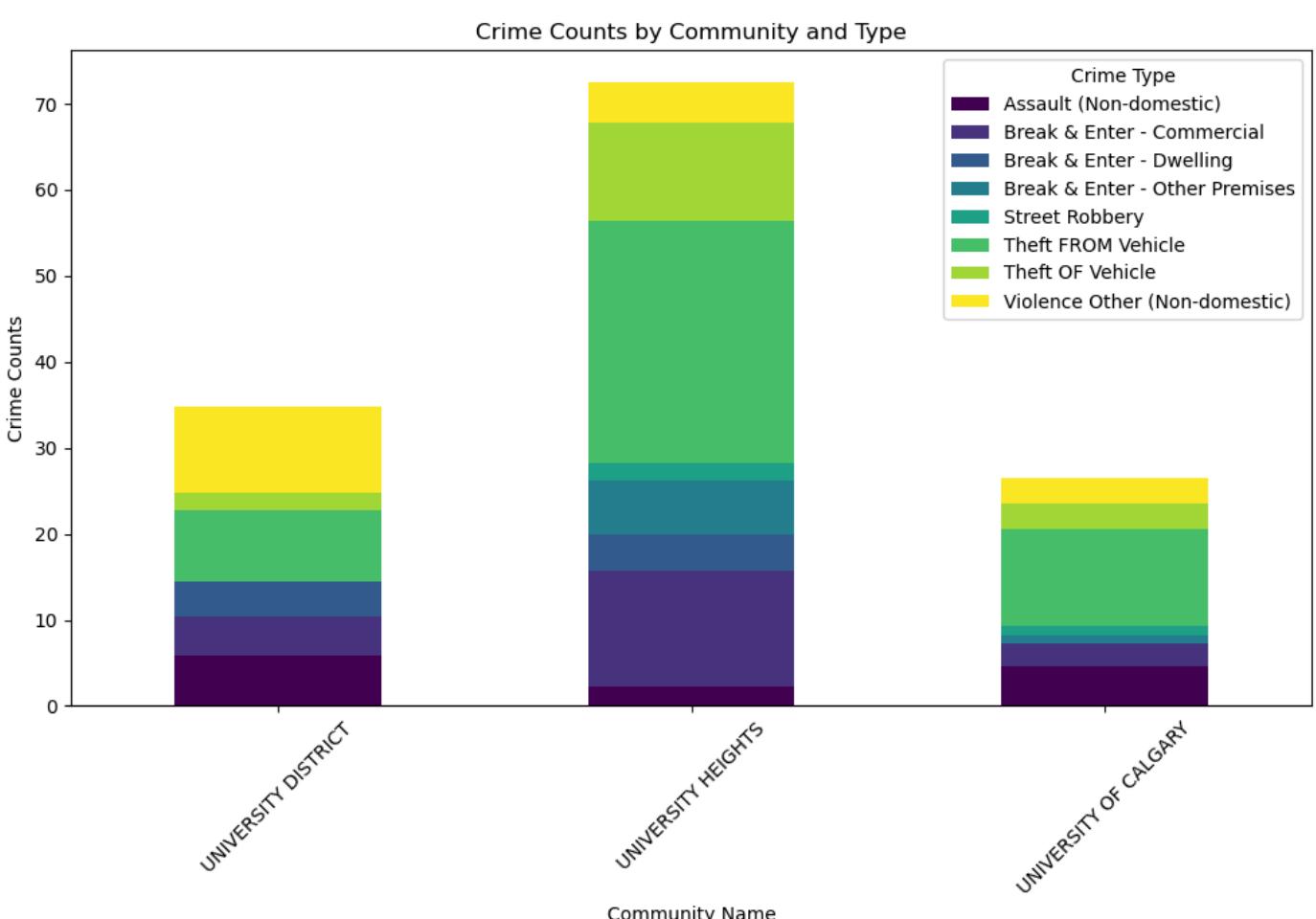
SELECT
    `crimedata`.`Year` AS `Year`,
    '' AS `Comm_code`,
    'Subtotal' AS `Community_Name`,
    NULL AS `Crime_Type`,
    SUM(`crimedata`.`Crime_Count`) AS `Crime_Count`
FROM (
    SELECT
        YEAR(`101-4`.`crime_statistics`.`CRIME_DATE`) AS `Year`,
        `101-4`.`crime_statistics`.`COMM_CODE` AS `COMM_CODE`,
        `101-4`.`crime_statistics`.`CATEGORY` AS `Crime_Type`,
        SUM(`101-4`.`crime_statistics`.`CRIME_COUNT`) AS `Crime_Count`
    FROM
        `101-4`.`crime_statistics`
    WHERE
        `haversine_distance_geo`(
            ST_GeometryFromText(CONCAT('POINT(', -114.136719, ' ', 51.078621, ')')),
            `101-4`.`crime_statistics`.`COMMUNITY_CENTER_POINT`
        ) <= 1000
    GROUP BY
        YEAR(`101-4`.`crime_statistics`.`CRIME_DATE`),
        `101-4`.`crime_statistics`.`COMM_CODE`,
        `101-4`.`crime_statistics`.`CATEGORY`
) `crimedata`
GROUP BY
    `crimedata`.`Year`
ORDER BY
    `Year`, `Community_Name`, `Crime_Type`;
"""

, engine)

# Plotting
pivot_df = query_table.pivot_table(index='Community_Name', columns='Crime_Type', values='Crime_C

# Plotting the stacked bar chart
pivot_df.plot(kind='bar', stacked=True, figsize=(10, 7), colormap='viridis')
plt.title('Crime Counts by Community and Type')
plt.xlabel('Community Name')
plt.ylabel('Crime Counts')
plt.xticks(rotation=45) # Rotate the Labels for better readability
plt.legend(title='Crime Type')
plt.tight_layout() # Adjust Layout to make room for label rotation
plt.show()

```



How Can We Determine the Number of Building Permits Issued in Different Communities Annually from 2018 to 2023

This query is designed to compute the construction time of projects and categorize them accordingly. It aims to explore the relationship between construction time and crime rates at the community level, providing insights that are crucial for exploratory data analysis (EDA). Additionally, analyzing the seasonal distribution of building permits each year may reveal potential connections with seasonal crime trends. The use of DATEDIFF allows for the precise calculation of the difference in days between the issue and completion dates of permits, while CASE WHEN is employed to categorize building times based on duration.

In [19]:

```
query_table = pd.read_sql_query("""
SELECT
    YEAR(`building_permits`.`ISSUED_DATE`) AS `Year`,
    `communities`.`COMMUNITY_NAME` AS `CommunityName`,
    SUM(CASE
        WHEN DATEDIFF(`building_permits`.`COMPLETED_DATE`, `building_permits`.`ISSUED_DATE`) <= 90
        ELSE 0
        END) AS `LessThan3Months`,
    SUM(CASE
        WHEN DATEDIFF(`building_permits`.`COMPLETED_DATE`, `building_permits`.`ISSUED_DATE`) > 90
        ELSE 0
        END) AS `Between3And6Months`,
    SUM(CASE
        WHEN DATEDIFF(`building_permits`.`COMPLETED_DATE`, `building_permits`.`ISSUED_DATE`) > 180
        ELSE 0
        END) AS `MoreThan6Months`
FROM
    `101-4`.`building_permits`
JOIN
    `communities` ON `building_permits`.`COMMUNITY_CODE` = `communities`.`COMM_CODE`
WHERE
    YEAR(`building_permits`.`ISSUED_DATE`) BETWEEN 2018 AND 2023
    AND `building_permits`.`COMPLETED_DATE` IS NOT NULL
GROUP BY
    YEAR(`building_permits`.`ISSUED_DATE`),
    `communities`.`COMMUNITY_NAME`
ORDER BY
    `Year`, `CommunityName`;
""", engine)

display(query_table)
# print("The permit amount of different community by year")
# print(ta(query_table.head(20), headers='keys', tablefmt='psql'))
```

| | Year | CommunityName | LessThan3Months | Between3And6Months | MoreThan6Months |
|-------------|-------------|---------------------------|------------------------|---------------------------|------------------------|
| 0 | 2018 | 01B | 1.0 | 1.0 | 2.0 |
| 1 | 2018 | 01C | 0.0 | 0.0 | 1.0 |
| 2 | 2018 | 01F | 2.0 | 0.0 | 1.0 |
| 3 | 2018 | 01I | 0.0 | 0.0 | 1.0 |
| 4 | 2018 | 01K | 1.0 | 1.0 | 0.0 |
| ... | ... | ... | ... | ... | ... |
| 1586 | 2023 | WINSTON HEIGHTS/MOUNTVIEW | 14.0 | 19.0 | 8.0 |
| 1587 | 2023 | WOLF WILLOW | 27.0 | 50.0 | 21.0 |
| 1588 | 2023 | WOODBINE | 21.0 | 6.0 | 1.0 |
| 1589 | 2023 | WOODLANDS | 15.0 | 2.0 | 3.0 |
| 1590 | 2023 | YORKVILLE | 21.0 | 15.0 | 50.0 |

1591 rows × 5 columns

Guiding Questions

We will conduct the queries based on the three guiding questions. In the first guiding question, we are interested in the overall crime statistics in Calgary from 2018 to 2023, thus we will query if there were any changes in the incident rate in each type of the crime, in the geographic pattern, and in the seasonality pattern over the years. We will also query if the crime statistics changed during the COVID-19 pandemic. In the second guiding question, we will examine the crime statistics of the communities around the LRT stations, thus we will query any patterns or associations with crime and the proximity of a C-train station,

and how do communities with C-train stations versus no C-train stations differed in the crime statistics. In the third guiding question, we will focus on economic factors. we will examine associations between community crime statistics and unemployment rate over the years, and query any patterns with crime and assessments by community, building permit, etc.

City's Overall Crime Statistics and Patterns (Jianling)

To Query Distribution of Crime by Sectors and Communities

This query explored the distribution of crime count and crime rates across different sectors and communities. It provides insights into how crime rates vary geographically, which can help identify high-crime areas and inform resource allocation for crime prevention measures. We will explore the annual crime count (rate /100,000 population) of the city, and by sector and by community level. Also we will query the descriptive statistics of annual crime count (rate/100,000 population) by communities.

In [20]:

```
query = """
SELECT
    r.Year,
    r.Total_Resident_Count,
    c.Total_Crime_Count,
    ROUND((c.Total_Crime_Count / r.Total_Resident_Count) * 100000, 2) AS Crime_Rate_Per_100k
FROM
    (
        SELECT
            Year,
            SUM(Resident_Count) AS Total_Resident_Count
        FROM
            (
                SELECT
                    DISTINCT Year(CRIME_DATE) AS Year,
                    COMM_CODE,
                    Resident_Count
                FROM
                    `101-4`.crime_statistics
            ) AS subquery
        GROUP BY
            Year) AS r
JOIN
    (
        SELECT
            Year(CRIME_DATE) AS Year,
            CAST(SUM(Crime_Count) AS DECIMAL(10,2)) AS Total_Crime_Count
        FROM
            `101-4`.crime_statistics
        GROUP BY
            Year
        ORDER BY
            Year) AS c ON r.Year = c.Year;
"""

query_df = pd.read_sql_query(query, engine)
display(query_df)
```

| | Year | Total_Resident_Count | Total_Crime_Count | Crime_Rate_Per_100k |
|---|------|----------------------|-------------------|---------------------|
| 0 | 2018 | 1280324.0 | 35339.0 | 2760.2 |
| 1 | 2019 | 1281801.0 | 38158.0 | 2976.9 |
| 2 | 2020 | 1283163.0 | 31786.0 | 2477.2 |
| 3 | 2021 | 1283177.0 | 28391.0 | 2212.6 |
| 4 | 2022 | 1283177.0 | 32740.0 | 2551.5 |
| 5 | 2023 | 1283177.0 | 27296.0 | 2127.2 |

Discussion: Based on the above output, we observe fluctuations in the annual crime count (crime rate), with 2019 showing the highest recorded rate. Following 2019, there appears to be a decreasing trend in the crime rate. This pattern suggests that there may have been events or changes in the city's dynamics in 2019 that led to the peak in crime rate. Future analysis could investigate the factors influencing these fluctuations to better understand the underlying trends in crime rates over the years.

To Query Annual Crime Count (Rate /100,000 Population) by Sectors

In [21]:

```
query = """
SELECT
    r.Year,
    r.Sector,
    SUM(r.Resident_Count) AS Total_Resident_Count,
    c.Annual_Crime_Count,
    ROUND((c.Annual_Crime_Count / SUM(r.Resident_Count)) * 100000,2) AS Annual_Crime_Rate_Per_100000_Population
FROM
    (SELECT
        DISTINCT Year(CRIME_DATE) as Year,
        Sector,
        COMM_CODE,
        Resident_Count
    FROM
        `101-4`.crime_statistics
    ) AS r
JOIN
    (SELECT
        Year(CRIME_DATE) as Year,
        Sector,
        CAST(SUM(Crime_Count) AS DECIMAL(10,2)) AS Annual_Crime_Count
    FROM
        `101-4`.crime_statistics
    GROUP BY
        Year, Sector
    ORDER BY
        Year, Sector
    ) AS c ON r.Year = c.Year AND r.Sector = c.Sector
GROUP BY
    r.Year, r.Sector, c.Annual_Crime_Count
ORDER BY
    r.Year, r.Sector;
"""

query_df = pd.read_sql_query(query, engine)
display(query_df)
```

| | Year | Sector | Total_Resident_Count | Annual_Crime_Count | Annual_Crime_Rate_Per_100k |
|----|------|-----------|----------------------|--------------------|----------------------------|
| 0 | 2018 | CENTRE | 202092.0 | 11364.0 | 5623.2 |
| 1 | 2018 | EAST | 58122.0 | 2740.0 | 4714.2 |
| 2 | 2018 | NORTH | 162886.0 | 2546.0 | 1563.1 |
| 3 | 2018 | NORTHEAST | 188567.0 | 6545.0 | 3470.9 |
| 4 | 2018 | NORTHWEST | 174515.0 | 3448.0 | 1975.8 |
| 5 | 2018 | SOUTH | 230015.0 | 4383.0 | 1905.5 |
| 6 | 2018 | SOUTHEAST | 139980.0 | 2123.0 | 1516.6 |
| 7 | 2018 | WEST | 124147.0 | 2183.0 | 1758.4 |
| 8 | 2019 | CENTRE | 202092.0 | 13473.0 | 6666.8 |
| 9 | 2019 | EAST | 58122.0 | 2980.0 | 5127.1 |
| 10 | 2019 | NORTH | 164363.0 | 2488.0 | 1513.7 |
| 11 | 2019 | NORTHEAST | 188567.0 | 6596.0 | 3498.0 |
| 12 | 2019 | NORTHWEST | 174515.0 | 3095.0 | 1773.5 |
| 13 | 2019 | SOUTH | 230015.0 | 4955.0 | 2154.2 |
| 14 | 2019 | SOUTHEAST | 139980.0 | 2282.0 | 1630.2 |
| 15 | 2019 | WEST | 124147.0 | 2286.0 | 1841.4 |
| 16 | 2020 | CENTRE | 202092.0 | 11064.0 | 5474.7 |
| 17 | 2020 | EAST | 58122.0 | 2468.0 | 4246.2 |
| 18 | 2020 | NORTH | 164363.0 | 2067.0 | 1257.6 |
| 19 | 2020 | NORTHEAST | 188567.0 | 5928.0 | 3143.7 |
| 20 | 2020 | NORTHWEST | 174515.0 | 2564.0 | 1469.2 |
| 21 | 2020 | SOUTH | 230115.0 | 3783.0 | 1644.0 |
| 22 | 2020 | SOUTHEAST | 139980.0 | 2025.0 | 1446.6 |
| 23 | 2020 | WEST | 125409.0 | 1882.0 | 1500.7 |
| 24 | 2021 | CENTRE | 202092.0 | 9702.0 | 4800.8 |
| 25 | 2021 | EAST | 58122.0 | 2611.0 | 4492.3 |
| 26 | 2021 | NORTH | 164363.0 | 1783.0 | 1084.8 |
| 27 | 2021 | NORTHEAST | 188567.0 | 5575.0 | 2956.5 |
| 28 | 2021 | NORTHWEST | 174515.0 | 2442.0 | 1399.3 |
| 29 | 2021 | SOUTH | 230129.0 | 3019.0 | 1311.9 |
| 30 | 2021 | SOUTHEAST | 139980.0 | 1534.0 | 1095.9 |
| 31 | 2021 | WEST | 125409.0 | 1714.0 | 1366.7 |
| 32 | 2022 | CENTRE | 202092.0 | 10043.0 | 4969.5 |
| 33 | 2022 | EAST | 58122.0 | 2911.0 | 5008.4 |
| 34 | 2022 | NORTH | 164363.0 | 2292.0 | 1394.5 |
| 35 | 2022 | NORTHEAST | 188567.0 | 6635.0 | 3518.6 |
| 36 | 2022 | NORTHWEST | 174515.0 | 3389.0 | 1942.0 |
| 37 | 2022 | SOUTH | 230129.0 | 3862.0 | 1678.2 |
| 38 | 2022 | SOUTHEAST | 139980.0 | 1929.0 | 1378.1 |
| 39 | 2022 | WEST | 125409.0 | 1677.0 | 1337.2 |
| 40 | 2023 | CENTRE | 202092.0 | 8606.0 | 4258.5 |
| 41 | 2023 | EAST | 58122.0 | 2392.0 | 4115.5 |
| 42 | 2023 | NORTH | 164363.0 | 1908.0 | 1160.8 |
| 43 | 2023 | NORTHEAST | 188567.0 | 5597.0 | 2968.2 |
| 44 | 2023 | NORTHWEST | 174515.0 | 2205.0 | 1263.5 |

| Year | Sector | Total_Resident_Count | Annual_Crime_Count | Annual_Crime_Rate_Per_100k |
|------|--------|----------------------|--------------------|----------------------------|
| 45 | 2023 | SOUTH | 230129.0 | 3447.0 |
| 46 | 2023 | SOUTHEAST | 139980.0 | 1706.0 |
| 47 | 2023 | WEST | 125409.0 | 1429.0 |

In [22]: # visualize the crime statistics by sector and year

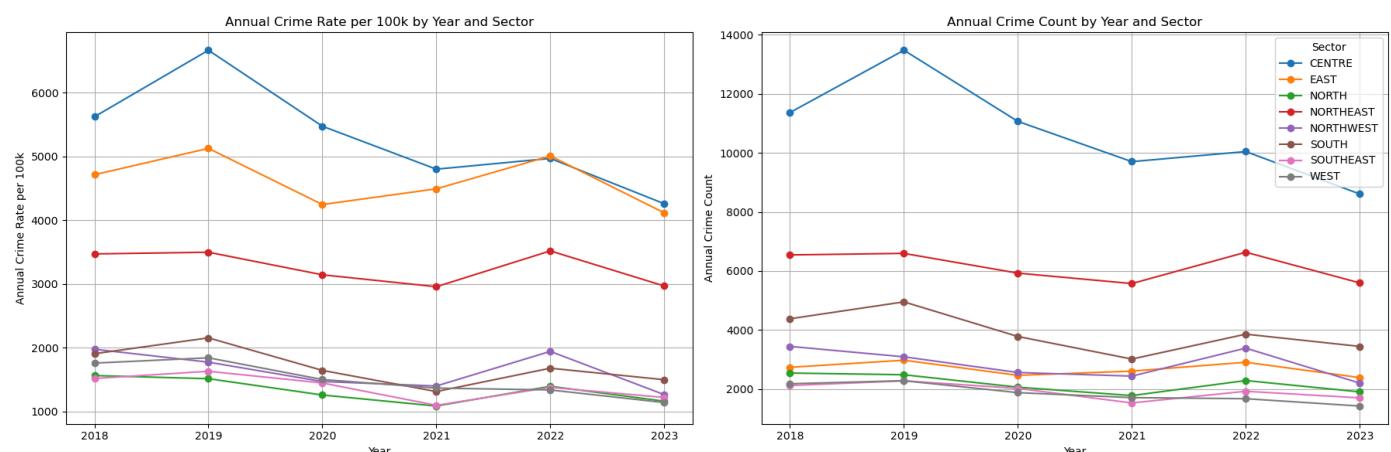
```
pivot_df_rate = query_df.pivot(index='Year', columns='Sector', values='Annual_Crime_Rate_Per_100k')
pivot_df_count = query_df.pivot(index='Year', columns='Sector', values='Annual_Crime_Count')

fig, axs = plt.subplots(1, 2, figsize=(18, 6))

# Plotting each sector's crime rate over the years
pivot_df_rate.plot(ax=axs[0], marker='o', legend=False)
axs[0].set_xlabel('Year')
axs[0].set_ylabel('Annual Crime Rate per 100k')
axs[0].set_title('Annual Crime Rate per 100k by Year and Sector')
axs[0].grid(True)

# Plotting each sector's crime count over the years
pivot_df_count.plot(ax=axs[1], marker='o')
axs[1].set_xlabel('Year')
axs[1].set_ylabel('Annual Crime Count')
axs[1].set_title('Annual Crime Count by Year and Sector')
axs[1].grid(True)

plt.tight_layout()
plt.show()
```



Discussion: Based on the results, the Centre sector exhibited the highest annual crime count, followed by the NorthEast sector. Even after adjusting for population size, the Centre sector maintained the highest crime rate, with the East and NorthEast sectors following closely behind.

To Query Descriptive Statistics of Annual Crime Count (Rate /100,000 Population) Statistics by Communities

In [23]: query_count = """

```
WITH CommunityCrimeSum AS (
    SELECT
        Year(CRIME_DATE) as Year,
        COMM_CODE,
        SUM(Crime_Count) AS Total_Crime_Count
    FROM
        `101-4`.crime_statistics
    GROUP BY
        Year,
        COMM_CODE
),
YearlyCommunityCrime AS (
    SELECT
        Year,
        Total_Crime_Count,
        ROW_NUMBER() OVER (PARTITION BY Year ORDER BY Total_Crime_Count) AS rn,
        COUNT(*) OVER (PARTITION BY Year) AS cnt
    FROM
        CommunityCrimeSum
)
SELECT
```

```

Year,
MAX(Total_Crime_Count) AS Max_Crime_Count,
MIN(Total_Crime_Count) AS Min_Crime_Count,
ROUND(AVG(Total_Crime_Count),2) AS Avg_Crime_Count,
PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Total_Crime_Count) OVER (PARTITION BY Year) AS Q1_Crime_Count,
PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Total_Crime_Count) OVER (PARTITION BY Year) AS Median_Crime_Count,
PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY Total_Crime_Count) OVER (PARTITION BY Year) AS Q3_Crime_Count
FROM
    YearlyCommunityCrime
GROUP BY
    Year
ORDER BY
    Year;
"""

query_count_df = pd.read_sql_query(query_count, engine)
display(query_count_df)

query_rate = """
WITH CommunityCrimeRates AS (
    SELECT
        r.Year,
        r.COMM_CODE,
        SUM(r.Resident_Count) AS Total_Resident_Count,
        c.Total_Crime_Count,
        ROUND((c.Total_Crime_Count / SUM(r.Resident_Count)) * 100000, 2) AS Crime_Rate_Per_100k
    FROM
        (SELECT
            DISTINCT Year(CRIME_DATE) as Year,
            COMM_CODE,
            Resident_Count
        FROM
            `101-4`.crime_statistics
        ) AS r
    JOIN
        (SELECT
            Year(CRIME_DATE) as Year,
            COMM_CODE,
            SUM(Crime_Count) AS Total_Crime_Count
        FROM
            `101-4`.crime_statistics
        GROUP BY
            Year, COMM_CODE
        ORDER BY
            Year, COMM_CODE
        ) AS c ON r.Year = c.Year AND r.COMM_CODE = c.COMM_CODE
    GROUP BY
        r.Year, r.COMM_CODE, c.Total_Crime_Count
)
SELECT
    Year,
    MAX(Crime_Rate_Per_100k) AS Max_Crime_Rate_Per_100k,
    MIN(Crime_Rate_Per_100k) AS Min_Crime_Rate_Per_100k,
    ROUND(AVG(Crime_Rate_Per_100k), 2) AS Avg_Crime_Rate_Per_100k
FROM
    CommunityCrimeRates
GROUP BY
    Year
ORDER BY
    Year;
"""

query_rate_df = pd.read_sql_query(query_rate, engine)
display(query_rate_df)

```

| | Year | Max_Crime_Count | Min_Crime_Count | Avg_Crime_Count | Q1_Crime_Count | Median_Crime_Count | Q3_Crime_Count |
|---|------|-----------------|-----------------|-----------------|----------------|--------------------|----------------|
| 0 | 2018 | 1697.0 | 1.0 | 132.36 | 1.0 | 1.0 | 1.0 |
| 1 | 2019 | 2229.0 | 1.0 | 141.33 | 1.0 | 1.0 | 1.0 |
| 2 | 2020 | 1603.0 | 1.0 | 113.93 | 1.0 | 1.0 | 1.0 |
| 3 | 2021 | 1432.0 | 1.0 | 101.76 | 1.0 | 1.0 | 1.0 |
| 4 | 2022 | 1273.0 | 1.0 | 114.48 | 1.0 | 1.0 | 1.0 |
| 5 | 2023 | 1237.0 | 1.0 | 95.44 | 1.0 | 1.0 | 1.0 |

| | Year | Max_Crime_Rate_Per_100k | Min_Crime_Rate_Per_100k | Avg_Crime_Rate_Per_100k |
|---|------|-------------------------|-------------------------|-------------------------|
| 0 | 2018 | 3372730.0 | 350.0 | 21332.28 |
| 1 | 2019 | 3227270.0 | 70.0 | 21194.69 |
| 2 | 2020 | 2936360.0 | 340.0 | 18824.38 |
| 3 | 2021 | 2463640.0 | 400.0 | 16883.08 |
| 4 | 2022 | 3427270.0 | 320.0 | 21949.24 |
| 5 | 2023 | 3609090.0 | 280.0 | 22833.22 |

Discussion: From the descriptive statistics, we can see that the number of crimes varied tremendously across communities, with annual counts ranging from 1 to more than one thousand. The average annual crime count per community varied from 95 to 141. This finding supports our hypothesis that there are significant variations in crime rates among different communities. Such variations can be attributed to a multitude of factors, including but not limited to socioeconomic conditions, population density, and the presence of community facilities.

To Query Annual Crime Count and Crime Rate of Each Community

In this query, we generate the annual crime count for each community. To visualize the distribution of crime counts across communities in year 2023, you will use the data generated by the query.

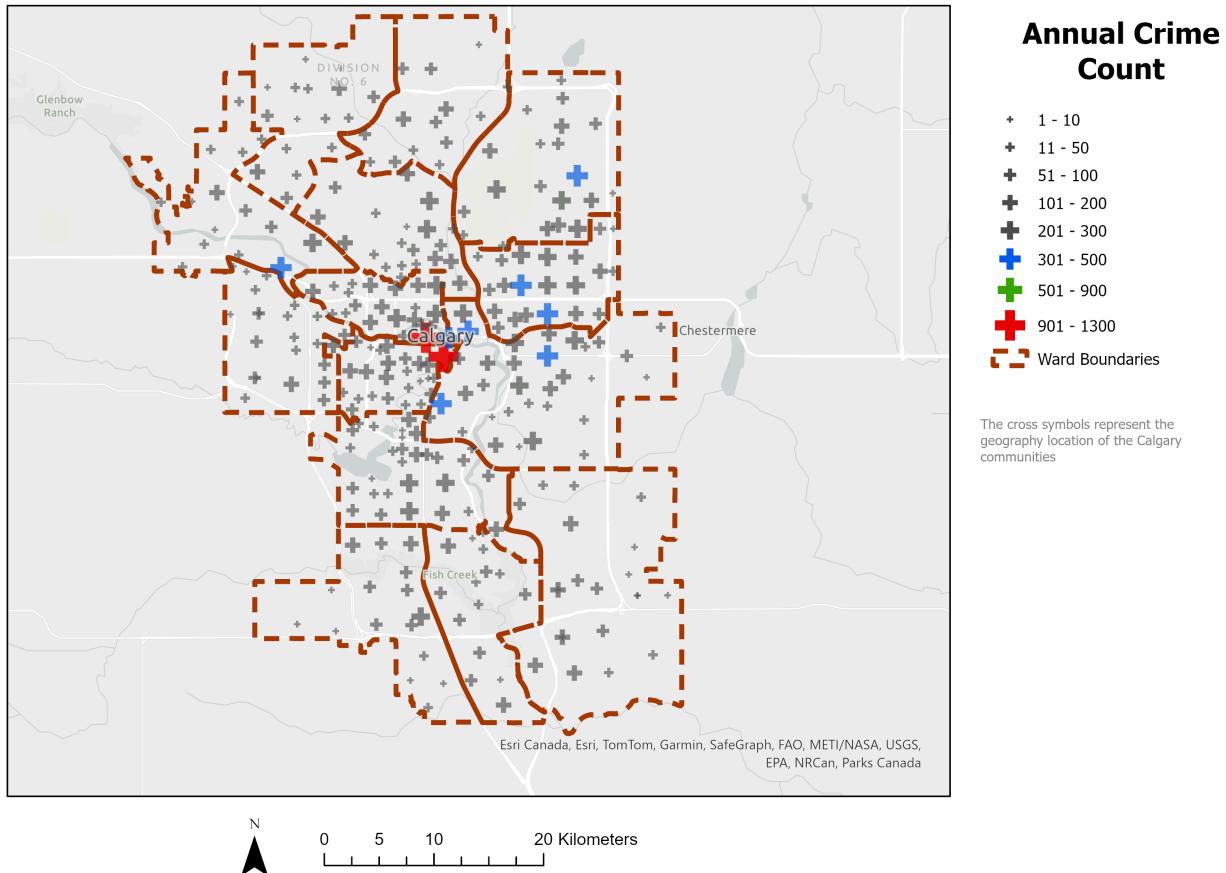
```
In [24]: query = """SELECT
    r.Year,
    r.COMM_CODE,
    SUM(r.Resident_Count) AS Total_Resident_Count,
    c.Total_Crime_Count,
    ROUND((c.Total_Crime_Count / SUM(r.Resident_Count)) * 100000, 2) AS Crime_Rate_Per_100k
FROM
    (SELECT
        DISTINCT Year(CRIME_DATE) as Year,
        COMM_CODE,
        Resident_Count
    FROM
        `101-4`.crime_statistics
    ) AS r
JOIN
    (SELECT
        Year(CRIME_DATE) as Year,
        COMM_CODE,
        CAST(SUM(Crime_Count) AS DECIMAL(10,2)) AS Total_Crime_Count
    FROM
        `101-4`.crime_statistics
    GROUP BY
        Year, COMM_CODE
    ORDER BY
        Year, COMM_CODE
    ) AS c ON r.Year = c.Year AND r.COMM_CODE = c.COMM_CODE
GROUP BY
    r.Year, r.COMM_CODE, c.Total_Crime_Count
ORDER BY
    r.Year, CASE WHEN r.Year = 2018 THEN c.Total_Crime_Count END DESC,
    CASE WHEN r.Year = 2023 THEN c.Total_Crime_Count END ASC;
"""
query_df = pd.read_sql_query(query, engine)
display(query_df)
```

| | Year | COMM_CODE | Total_Resident_Count | Total_Crime_Count | Crime_Rate_Per_100k |
|------|------|-----------|----------------------|-------------------|---------------------|
| 0 | 2018 | BLN | 25129.0 | 1697.0 | 6753.2 |
| 1 | 2018 | DNC | 8683.0 | 1148.0 | 13221.2 |
| 2 | 2018 | FLN | 7814.0 | 528.0 | 6757.1 |
| 3 | 2018 | BOW | 11150.0 | 494.0 | 4430.5 |
| 4 | 2018 | MRL | 9162.0 | 430.0 | 4693.3 |
| ... | ... | ... | ... | ... | ... |
| 1662 | 2023 | MRL | 9162.0 | 402.0 | 4387.7 |
| 1663 | 2023 | FLN | 7814.0 | 420.0 | 5375.0 |
| 1664 | 2023 | DNE | 3893.0 | 472.0 | 12124.3 |
| 1665 | 2023 | DNC | 8683.0 | 1028.0 | 11839.2 |
| 1666 | 2023 | BLN | 25129.0 | 1237.0 | 4922.6 |

1667 rows × 5 columns

To visualize the crime counts across communities in 2023, we extracted the data from this query and plot on a map using ArcGIS Pro

2023 Calgary Community Annual Crime Count



To Query Top Five Communities for Highest Annual Crime Incidence

```
In [25]: query = """
WITH CommunityCrimeCounts AS (
    SELECT
        COMM_CODE,
        Year(CRIME_DATE) as Year,
        Sector,
        SUM(Crime_Count) AS Total_Crime_Count
    FROM
        `101-4`.crime_statistics
    GROUP BY
        COMM_CODE, Year, Sector
),
RankedCommunities AS (
    SELECT
        c.COMM_CODE,
        c.Year,
```

```

        c.Sector,
        c.Total_Crime_Count
    ,ROW_NUMBER() OVER (PARTITION BY c.Year ORDER BY c.Total_Crime_Count DESC) AS `Rank`
FROM
    CommunityCrimeCounts c
),
CommunityRowCount AS (
SELECT
    COMM_CODE,
    Year(CRIME_DATE) as Year,
    COUNT(*) AS Row_Count
FROM
    `101-4`.crime_statistics
GROUP BY
    COMM_CODE, Year
),
AverageResidentCount AS (
SELECT
    c.COMM_CODE,
    Year(c.CRIME_DATE) as Year,
    SUM(c.Resident_Count) / r.Row_Count AS Avg_Resident_Count
FROM
    `101-4`.crime_statistics c
    JOIN CommunityRowCount r ON c.COMM_CODE = r.COMM_CODE AND Year(c.CRIME_DATE) = r.Year
GROUP BY
    c.COMM_CODE, Year(c.CRIME_DATE)
)
SELECT
    rc.Sector,
    rc.COMM_CODE,
    c.community_name,
    rc.Year,
    rc.Total_Crime_Count,
    ar.Avg_Resident_Count
    ,ROUND(CAST(rc.Total_Crime_Count AS DECIMAL (10,5))/ CAST(ar.Avg_Resident_Count AS DECIMAL (
FROM RankedCommunities rc
    JOIN AverageResidentCount ar ON rc.COMM_CODE = ar.COMM_CODE AND rc.Year = ar.Year
    LEFT JOIN communities c on rc.comm_code = c.comm_code
WHERE
    `Rank` <= 5
ORDER BY
    rc.Year,
    rc.Total_Crime_Count DESC;
"""
query_df = pd.read_sql_query(query, engine)
display(query_df)

```

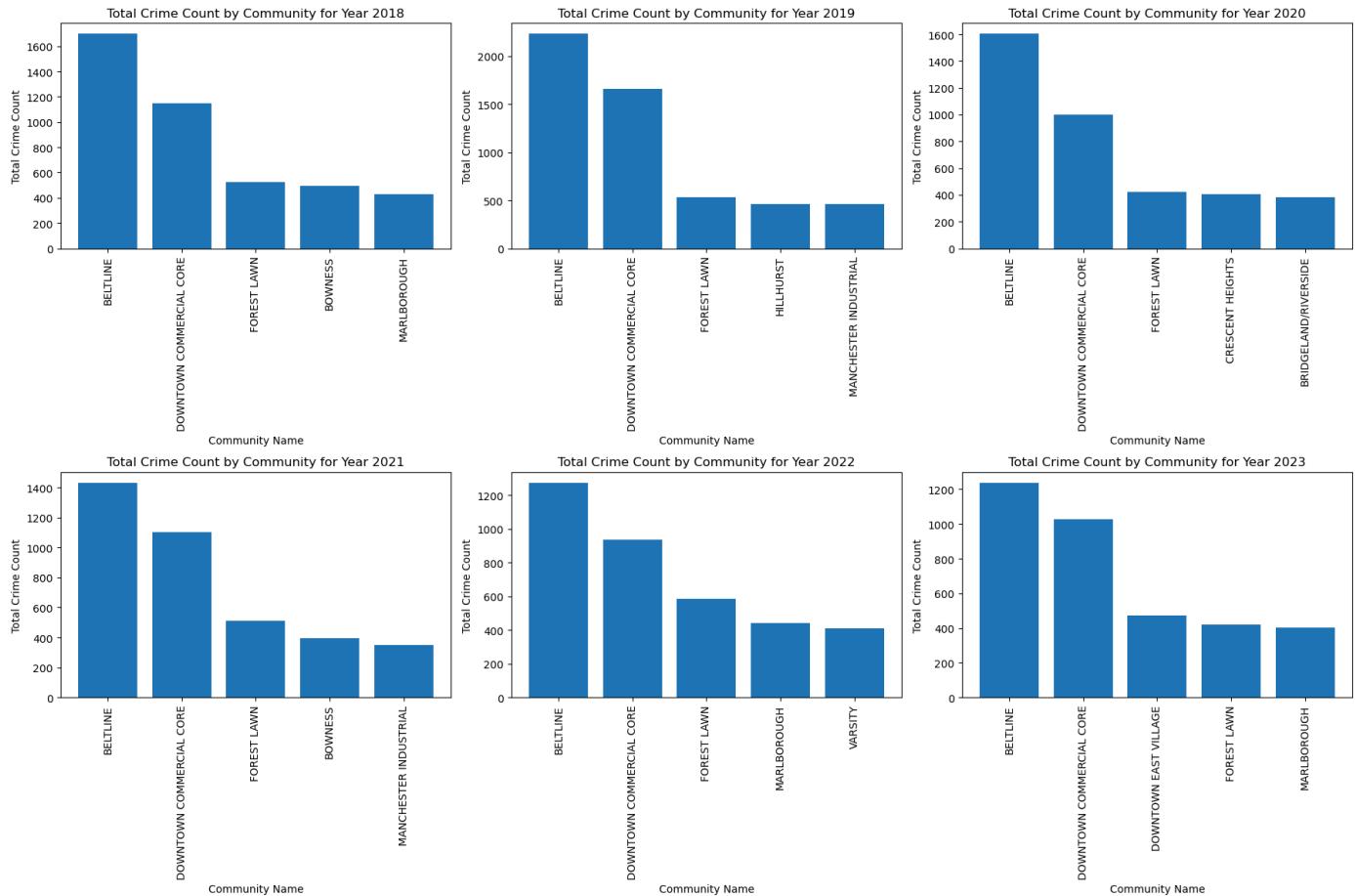
| | Sector | COMM_CODE | community_name | Year | Total_Crime_Count | Avg_Resident_Count | Crime_Rate |
|----|-----------|-----------|--------------------------|------|-------------------|--------------------|------------|
| 0 | CENTRE | BLN | BELTLINE | 2018 | 1697.0 | 25129.0 | |
| 1 | CENTRE | DNC | DOWNTOWN COMMERCIAL CORE | 2018 | 1148.0 | 8683.0 | |
| 2 | EAST | FLN | FOREST LAWN | 2018 | 528.0 | 7814.0 | |
| 3 | NORTHWEST | BOW | BOWNESS | 2018 | 494.0 | 11150.0 | |
| 4 | NORTHEAST | MRL | MARLBOROUGH | 2018 | 430.0 | 9162.0 | |
| 5 | CENTRE | BLN | BELTLINE | 2019 | 2229.0 | 25129.0 | |
| 6 | CENTRE | DNC | DOWNTOWN COMMERCIAL CORE | 2019 | 1661.0 | 8683.0 | |
| 7 | EAST | FLN | FOREST LAWN | 2019 | 532.0 | 7814.0 | |
| 8 | CENTRE | HIL | HILLHURST | 2019 | 461.0 | 6558.0 | |
| 9 | CENTRE | MNI | MANCHESTER INDUSTRIAL | 2019 | 460.0 | 0.0 | |
| 10 | CENTRE | BLN | BELTLINE | 2020 | 1603.0 | 25129.0 | |
| 11 | CENTRE | DNC | DOWNTOWN COMMERCIAL CORE | 2020 | 1001.0 | 8683.0 | |
| 12 | EAST | FLN | FOREST LAWN | 2020 | 421.0 | 7814.0 | |
| 13 | CENTRE | CRE | CRESCENT HEIGHTS | 2020 | 407.0 | 6620.0 | |
| 14 | CENTRE | BRD | BRIDGELAND/RIVERSIDE | 2020 | 384.0 | 6835.0 | |
| 15 | CENTRE | BLN | BELTLINE | 2021 | 1432.0 | 25129.0 | |
| 16 | CENTRE | DNC | DOWNTOWN COMMERCIAL CORE | 2021 | 1102.0 | 8683.0 | |
| 17 | EAST | FLN | FOREST LAWN | 2021 | 512.0 | 7814.0 | |
| 18 | NORTHWEST | BOW | BOWNESS | 2021 | 393.0 | 11150.0 | |
| 19 | CENTRE | MNI | MANCHESTER INDUSTRIAL | 2021 | 352.0 | 0.0 | |
| 20 | CENTRE | BLN | BELTLINE | 2022 | 1273.0 | 25129.0 | |
| 21 | CENTRE | DNC | DOWNTOWN COMMERCIAL CORE | 2022 | 936.0 | 8683.0 | |
| 22 | EAST | FLN | FOREST LAWN | 2022 | 583.0 | 7814.0 | |
| 23 | NORTHEAST | MRL | MARLBOROUGH | 2022 | 442.0 | 9162.0 | |
| 24 | NORTHWEST | VAR | VARSITY | 2022 | 409.0 | 12874.0 | |
| 25 | CENTRE | BLN | BELTLINE | 2023 | 1237.0 | 25129.0 | |
| 26 | CENTRE | DNC | DOWNTOWN COMMERCIAL CORE | 2023 | 1028.0 | 8683.0 | |
| 27 | CENTRE | DNE | DOWNTOWN EAST VILLAGE | 2023 | 472.0 | 3893.0 | |
| 28 | EAST | FLN | FOREST LAWN | 2023 | 420.0 | 7814.0 | |
| 29 | NORTHEAST | MRL | MARLBOROUGH | 2023 | 402.0 | 9162.0 | |

In [26]: `## To visualize the five communities that had the highest crime count`

```
query_df = pd.read_sql_query(query, engine)

fig, axs = plt.subplots(2, 3, figsize=(18, 12))
for i, year in enumerate(sorted(query_df['Year'].unique())):
    year_data = query_df[query_df['Year'] == year]
    ax = axs[i // 3, i % 3]
    ax.bar(year_data['community_name'], year_data['Total_Crime_Count'])
    ax.set_xlabel('Community Name')
    ax.set_ylabel('Total Crime Count')
    ax.set_title(f'Total Crime Count by Community for Year {year}')
    ax.tick_params(axis='x', rotation=90)
```

```
plt.tight_layout()
plt.show()
```



Discussion: The Beltline community in the Center Sector was consistently found to have the highest annual crime counts from 2018 to 2023. After accounting for the community population size, the Downtown Commercial Core community in the Center sector had the highest annual crime rate. This suggested that Beltline and Downtown Commercial Core communities may face specific challenges or factors that contribute to a higher crime rate per capita compared to other communities. Understanding these factors could help in implementing targeted interventions to address crime and improve community safety in these areas.

To Query the Distribution of Types of Crimes

In this query, we tried to get insights into the categories of crimes and explore which community had a specific type of crime occur most frequently:

In [27]:

```
query = """
WITH CrimeByYear AS (
    SELECT
        Year(CRIME_DATE) as Year,
        Category AS Crime_Type,
        SUM(Crime_Count) AS Total_Crime_Count
    FROM
        `101-4`.crime_statistics
    GROUP BY
        Year,
        Category
),
TotalCrimeByYear AS (
    SELECT
        Year(CRIME_DATE) as Year,
        SUM(Crime_Count) AS Total_Crime_Count
    FROM
        `101-4`.crime_statistics
    GROUP BY
        Year
)
SELECT
    c.Year,
    c.Crime_Type,
    c.Total_Crime_Count,
    ROUND((c.Total_Crime_Count / t.Total_Crime_Count) * 100,2) AS Percentage_of_Total_Crime
FROM
    CrimeByYear c
JOIN
    TotalCrimeByYear t
ON
    c.Year = t.Year
ORDER BY
    c.Year, c.Crime_Type, Percentage_of_Total_Crime DESC;"""

# Execute the query
pd.read_sql(query, engine)
```

```
TotalCrimeByYear t ON c.Year = t.Year
ORDER BY
    c.Year,
    Percentage_Of_Total_Crime DESC;
"""
query_df = pd.read_sql_query(query, engine)
display(query_df)
```

| Year | Crime_Type | Total_Crime_Count | Percentage_Of_Total_Crime |
|------|--------------------------------|-------------------|---------------------------|
| 0 | Theft FROM Vehicle | 12624.0 | 35.72 |
| 1 | Theft OF Vehicle | 6059.0 | 17.15 |
| 2 | Break & Enter - Commercial | 5057.0 | 14.31 |
| 3 | Assault (Non-domestic) | 3828.0 | 10.83 |
| 4 | Break & Enter - Dwelling | 2693.0 | 7.62 |
| 5 | Break & Enter - Other Premises | 2104.0 | 5.95 |
| 6 | Violence Other (Non-domestic) | 2011.0 | 5.69 |
| 7 | Street Robbery | 658.0 | 1.86 |
| 8 | Commercial Robbery | 305.0 | 0.86 |
| 9 | Theft FROM Vehicle | 14723.0 | 38.58 |
| 10 | Theft OF Vehicle | 6028.0 | 15.80 |
| 11 | Break & Enter - Commercial | 5785.0 | 15.16 |
| 12 | Assault (Non-domestic) | 3913.0 | 10.25 |
| 13 | Break & Enter - Dwelling | 2401.0 | 6.29 |
| 14 | Break & Enter - Other Premises | 2303.0 | 6.04 |
| 15 | Violence Other (Non-domestic) | 1974.0 | 5.17 |
| 16 | Street Robbery | 649.0 | 1.70 |
| 17 | Commercial Robbery | 382.0 | 1.00 |
| 18 | Theft FROM Vehicle | 12388.0 | 38.97 |
| 19 | Theft OF Vehicle | 4654.0 | 14.64 |
| 20 | Break & Enter - Commercial | 4209.0 | 13.24 |
| 21 | Assault (Non-domestic) | 3529.0 | 11.10 |
| 22 | Break & Enter - Other Premises | 2500.0 | 7.87 |
| 23 | Violence Other (Non-domestic) | 2014.0 | 6.34 |
| 24 | Break & Enter - Dwelling | 1686.0 | 5.30 |
| 25 | Street Robbery | 525.0 | 1.65 |
| 26 | Commercial Robbery | 281.0 | 0.88 |
| 27 | Theft FROM Vehicle | 10159.0 | 35.78 |
| 28 | Theft OF Vehicle | 4453.0 | 15.68 |
| 29 | Assault (Non-domestic) | 4036.0 | 14.22 |
| 30 | Break & Enter - Commercial | 3218.0 | 11.33 |
| 31 | Violence Other (Non-domestic) | 2179.0 | 7.67 |
| 32 | Break & Enter - Other Premises | 1835.0 | 6.46 |
| 33 | Break & Enter - Dwelling | 1577.0 | 5.55 |
| 34 | Street Robbery | 575.0 | 2.03 |
| 35 | Commercial Robbery | 359.0 | 1.26 |
| 36 | Theft FROM Vehicle | 12352.0 | 37.73 |
| 37 | Theft OF Vehicle | 5427.0 | 16.58 |
| 38 | Break & Enter - Commercial | 4277.0 | 13.06 |
| 39 | Assault (Non-domestic) | 4139.0 | 12.64 |
| 40 | Violence Other (Non-domestic) | 2084.0 | 6.37 |
| 41 | Break & Enter - Other Premises | 1948.0 | 5.95 |
| 42 | Break & Enter - Dwelling | 1586.0 | 4.84 |
| 43 | Street Robbery | 536.0 | 1.64 |
| 44 | Commercial Robbery | 391.0 | 1.19 |

| Year | Crime_Type | Total_Crime_Count | Percentage_Of_Total_Crime |
|------|--------------------------------|-------------------|---------------------------|
| 45 | Theft FROM Vehicle | 8957.0 | 32.81 |
| 46 | Assault (Non-domestic) | 4577.0 | 16.77 |
| 47 | Theft OF Vehicle | 4448.0 | 16.30 |
| 48 | Break & Enter - Commercial | 3352.0 | 12.28 |
| 49 | Violence Other (Non-domestic) | 2133.0 | 7.81 |
| 50 | Break & Enter - Dwelling | 1525.0 | 5.59 |
| 51 | Break & Enter - Other Premises | 1300.0 | 4.76 |
| 52 | Street Robbery | 693.0 | 2.54 |
| 53 | Commercial Robbery | 311.0 | 1.14 |

```
In [28]: # To visualize the distribution of categories of crime
```

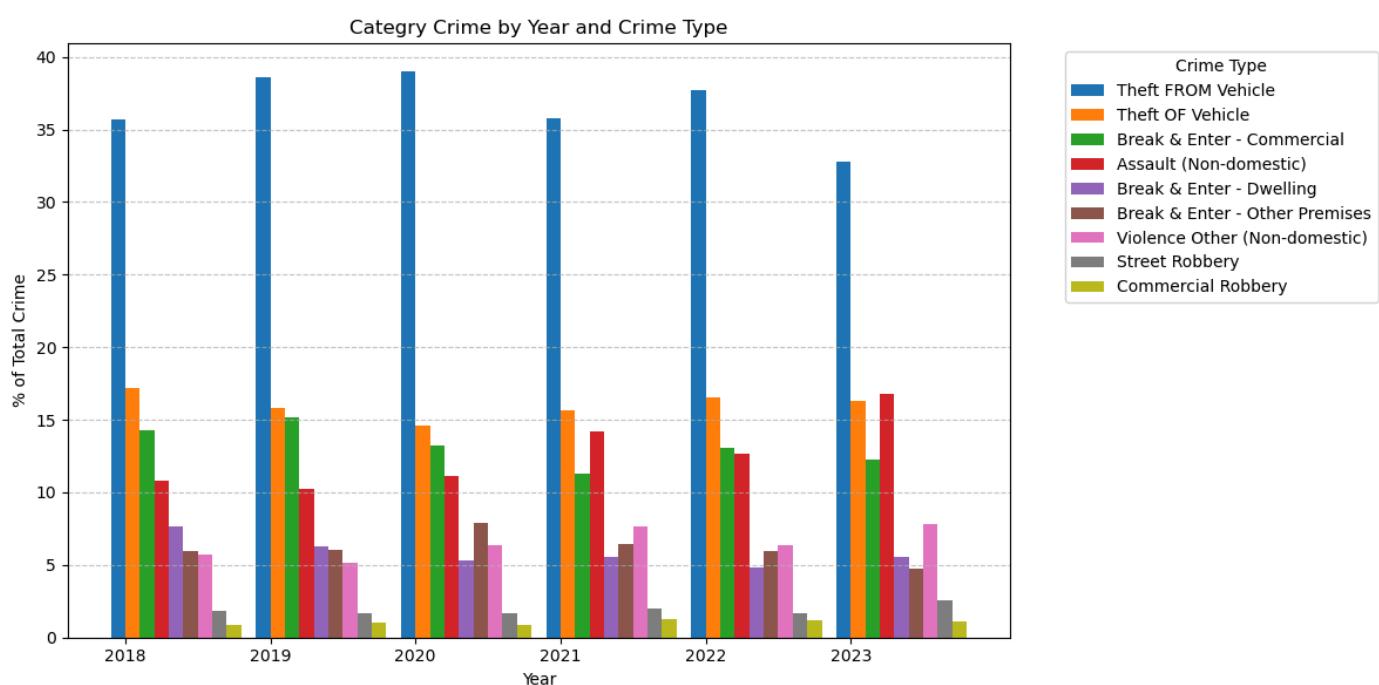
```
query_df = pd.read_sql_query(query, engine)

crime_types = query_df['Crime_Type'].unique()

fig, ax = plt.subplots(figsize=(12, 6))

bar_width = 0.1
years = query_df['Year'].unique()
for i, crime_type in enumerate(crime_types):
    crime_data = query_df[query_df['Crime_Type'] == crime_type]
    bar_positions = np.arange(len(years)) + i * bar_width
    ax.bar(bar_positions, crime_data['Percentage_Of_Total_Crime'], bar_width, label=crime_type)

ax.set_xlabel('Year')
ax.set_ylabel('% of Total Crime')
ax.set_title('Category Crime by Year and Crime Type')
ax.set_xticks(np.arange(len(years)) + bar_width / 2)
ax.set_xticklabels(years)
ax.legend(title='Crime Type', bbox_to_anchor=(1.05, 1), loc='upper left')
ax.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



Discussion: From the output, theft from a vehicle was the most common type of crime from 2018 to 2023, accounting for 32.8% to 38.6% of the total annual count. And commercial robbery was the least frequent. The pattern was consistency through the years.

To Query the Community That Had a Specific Type of Crime Occurred Most Frequently

```
In [29]: query = """
WITH CrimeByCommunity AS (
    SELECT
        COMM_CODE,
```

```

        Year(CRIME_DATE) as Year,
        Category AS Crime_Type,
        Sector,
        CAST( SUM(Crime_Count) AS DECIMAL(10,2)) AS Total_Crime_Count
    FROM
        `101-4`.crime_statistics
    GROUP BY
        Year,
        COMM_CODE,
        Crime_Type,
        Sector
),
ResidentCountByCommunity AS (
    SELECT
        COMM_CODE,
        Year(CRIME_DATE) as Year,
        SUM(Resident_Count) / COUNT(*) AS Total_Resident_Count
    FROM
        `101-4`.crime_statistics
    GROUP BY
        Year,
        COMM_CODE
),
RankedCommunities AS (
    SELECT
        cc.COMM_CODE,
        cc.Year,
        cc.Crime_Type,
        cc.Sector,
        c.community_name,
        cc.Total_Crime_Count,
        rc.Total_Resident_Count,
        ROUND((cc.Total_Crime_Count / rc.Total_Resident_Count) * 100000,2) AS Crime_Rate_Per_100000,
        ROW_NUMBER() OVER (PARTITION BY cc.Year, cc.Crime_Type ORDER BY cc.Total_Crime_Count DESC) AS Rank
    FROM
        CrimeByCommunity cc
    JOIN ResidentCountByCommunity rc ON cc.COMM_CODE = rc.COMM_CODE AND cc.Year = rc.Year
    LEFT JOIN communities c ON c.comm_code = cc.comm_code
)
SELECT
    Year,
    Crime_Type,
    community_name AS Community_Name,
    Sector,
    Total_Resident_Count,
    CONCAT(Total_Crime_Count, ' (' , Crime_Rate_Per_100000, ')') AS Crime_Count_And_Rate
FROM
    RankedCommunities
WHERE
    Rank <= 1
ORDER BY
    Year,
    Crime_Type,
    Total_Crime_Count DESC;
"""
query_df = pd.read_sql_query(query, engine)
display(query_df)

```

| | Year | Crime_Type | Community_Name | Sector | Total_Resident_Count | Crime_Count_And_Rate |
|----|------|--------------------------------|--------------------------|-----------|----------------------|----------------------|
| 0 | 2018 | Assault (Non-domestic) | DOWNTOWN COMMERCIAL CORE | CENTRE | 8683.0 | 359.00 (4134.50) |
| 1 | 2018 | Break & Enter - Commercial | BELTLINE | CENTRE | 25129.0 | 459.00 (1826.60) |
| 2 | 2018 | Break & Enter - Dwelling | PANORAMA HILLS | NORTH | 25710.0 | 57.00 (221.70) |
| 3 | 2018 | Break & Enter - Other Premises | WEST HILLHURST | CENTRE | 6447.0 | 63.00 (977.20) |
| 4 | 2018 | Commercial Robbery | SUNRIDGE | NORTHEAST | 11.0 | 18.00 (163636.40) |
| 5 | 2018 | Street Robbery | DOWNTOWN COMMERCIAL CORE | CENTRE | 8683.0 | 49.00 (564.30) |
| 6 | 2018 | Theft FROM Vehicle | BELTLINE | CENTRE | 25129.0 | 556.00 (2212.60) |
| 7 | 2018 | Theft OF Vehicle | BELTLINE | CENTRE | 25129.0 | 132.00 (525.30) |
| 8 | 2018 | Violence Other (Non-domestic) | DOWNTOWN COMMERCIAL CORE | CENTRE | 8683.0 | 119.00 (1370.50) |
| 9 | 2019 | Assault (Non-domestic) | DOWNTOWN COMMERCIAL CORE | CENTRE | 8683.0 | 362.00 (4169.10) |
| 10 | 2019 | Break & Enter - Commercial | BELTLINE | CENTRE | 25129.0 | 607.00 (2415.50) |
| 11 | 2019 | Break & Enter - Dwelling | EDGEMONT | NORTHWEST | 15395.0 | 51.00 (331.30) |
| 12 | 2019 | Break & Enter - Other Premises | MOUNT PLEASANT | CENTRE | 5889.0 | 72.00 (1222.60) |
| 13 | 2019 | Commercial Robbery | BELTLINE | CENTRE | 25129.0 | 21.00 (83.60) |
| 14 | 2019 | Street Robbery | BELTLINE | CENTRE | 25129.0 | 45.00 (179.10) |
| 15 | 2019 | Theft FROM Vehicle | BELTLINE | CENTRE | 25129.0 | 892.00 (3549.70) |
| 16 | 2019 | Theft OF Vehicle | BELTLINE | CENTRE | 25129.0 | 147.00 (585.00) |
| 17 | 2019 | Violence Other (Non-domestic) | DOWNTOWN COMMERCIAL CORE | CENTRE | 8683.0 | 113.00 (1301.40) |
| 18 | 2020 | Assault (Non-domestic) | BELTLINE | CENTRE | 25129.0 | 349.00 (1388.80) |
| 19 | 2020 | Break & Enter - Commercial | BELTLINE | CENTRE | 25129.0 | 312.00 (1241.60) |
| 20 | 2020 | Break & Enter - Dwelling | EDGEMONT | NORTHWEST | 15395.0 | 35.00 (227.30) |
| 21 | 2020 | Break & Enter - Other Premises | WEST HILLHURST | CENTRE | 6447.0 | 82.00 (1271.90) |
| 22 | 2020 | Commercial Robbery | BELTLINE | CENTRE | 25129.0 | 17.00 (67.70) |
| 23 | 2020 | Street Robbery | BELTLINE | CENTRE | 25129.0 | 37.00 (147.20) |
| 24 | 2020 | Theft FROM Vehicle | BELTLINE | CENTRE | 25129.0 | 634.00 (2523.00) |
| 25 | 2020 | Theft OF Vehicle | BELTLINE | CENTRE | 25129.0 | 119.00 (473.60) |
| 26 | 2020 | Violence Other (Non-domestic) | BELTLINE | CENTRE | 25129.0 | 94.00 (374.10) |
| 27 | 2021 | Assault (Non-domestic) | DOWNTOWN COMMERCIAL CORE | CENTRE | 8683.0 | 395.00 (4549.10) |
| 28 | 2021 | Break & Enter - Commercial | BELTLINE | CENTRE | 25129.0 | 262.00 (1042.60) |
| 29 | 2021 | Break & Enter - Dwelling | PANORAMA HILLS | NORTH | 25710.0 | 28.00 (108.90) |
| 30 | 2021 | Break & Enter - | KILLARNEY/GLENGARRY | CENTRE | 7685.0 | 54.00 (702.70) |

| Year | Crime_Type | Community_Name | Sector | Total_Resident_Count | Crime_Count_And_Rate |
|----------------|--------------------------------|--------------------------|-----------|----------------------|----------------------|
| Other Premises | | | | | |
| 31 | Commercial Robbery | BELTLINE | CENTRE | 25129.0 | 17.00 (67.70) |
| 32 | Street Robbery | DOWNTOWN COMMERCIAL CORE | CENTRE | 8683.0 | 66.00 (760.10) |
| 33 | Theft FROM Vehicle | BELTLINE | CENTRE | 25129.0 | 517.00 (2057.40) |
| 34 | Theft OF Vehicle | BELTLINE | CENTRE | 25129.0 | 108.00 (429.80) |
| 35 | Violence Other (Non-domestic) | BELTLINE | CENTRE | 25129.0 | 103.00 (409.90) |
| 36 | Assault (Non-domestic) | BELTLINE | CENTRE | 25129.0 | 348.00 (1384.90) |
| 37 | Break & Enter - Commercial | BELTLINE | CENTRE | 25129.0 | 218.00 (867.50) |
| 38 | Break & Enter - Dwelling | VARSITY | NORTHWEST | 12874.0 | 41.00 (318.50) |
| 39 | Break & Enter - Other Premises | KILLARNEY/GLENGARRY | CENTRE | 7685.0 | 51.00 (663.60) |
| 40 | Commercial Robbery | BELTLINE | CENTRE | 25129.0 | 15.00 (59.70) |
| 41 | Street Robbery | DOWNTOWN COMMERCIAL CORE | CENTRE | 8683.0 | 64.00 (737.10) |
| 42 | Theft FROM Vehicle | BELTLINE | CENTRE | 25129.0 | 419.00 (1667.40) |
| 43 | Theft OF Vehicle | FOREST LAWN | EAST | 7814.0 | 139.00 (1778.90) |
| 44 | Violence Other (Non-domestic) | BELTLINE | CENTRE | 25129.0 | 102.00 (405.90) |
| 45 | Assault (Non-domestic) | DOWNTOWN COMMERCIAL CORE | CENTRE | 8683.0 | 376.00 (4330.30) |
| 46 | Break & Enter - Commercial | BELTLINE | CENTRE | 25129.0 | 167.00 (664.60) |
| 47 | Break & Enter - Dwelling | BELTLINE | CENTRE | 25129.0 | 34.00 (135.30) |
| 48 | Break & Enter - Other Premises | DOVER | EAST | 10351.0 | 35.00 (338.10) |
| 49 | Commercial Robbery | BELTLINE | CENTRE | 25129.0 | 28.00 (111.40) |
| 50 | Street Robbery | DOWNTOWN COMMERCIAL CORE | CENTRE | 8683.0 | 65.00 (748.60) |
| 51 | Theft FROM Vehicle | BELTLINE | CENTRE | 25129.0 | 402.00 (1599.70) |
| 52 | Theft OF Vehicle | BELTLINE | CENTRE | 25129.0 | 102.00 (405.90) |
| 53 | Violence Other (Non-domestic) | BELTLINE | CENTRE | 25129.0 | 103.00 (409.90) |

In [30]:

```
# Pivot the DataFrame to create a table with rows for each crime category and columns for the year
pivot_table = query_df.pivot(index='Crime_Type', columns='Year', values='Community_Name')

display(pivot_table)
```

| Year | 2018 | 2019 | 2020 | 2021 | 2022 | 2023 |
|--------------------------------|--------------------------|--------------------------|----------------|--------------------------|--------------------------|--------------------------|
| Crime_Type | | | | | | |
| Assault (Non-domestic) | DOWNTOWN COMMERCIAL CORE | DOWNTOWN COMMERCIAL CORE | BELTLINE | DOWNTOWN COMMERCIAL CORE | BELTLINE | DOWNTOWN COMMERCIAL CORE |
| Break & Enter - Commercial | BELTLINE | BELTLINE | BELTLINE | BELTLINE | BELTLINE | BELTLINE |
| Break & Enter - Dwelling | PANORAMA HILLS | EDGEMONT | EDGEMONT | PANORAMA HILLS | VARSITY | BELTLINE |
| Break & Enter - Other Premises | WEST HILLHURST | MOUNT PLEASANT | WEST HILLHURST | KILLARNEY/GLENGARRY | KILLARNEY/GLENGARRY | DOVER |
| Commercial Robbery | SUNRIDGE | BELTLINE | BELTLINE | BELTLINE | BELTLINE | BELTLINE |
| Street Robbery | DOWNTOWN COMMERCIAL CORE | BELTLINE | BELTLINE | DOWNTOWN COMMERCIAL CORE | DOWNTOWN COMMERCIAL CORE | DOWNTOWN COMMERCIAL CORE |
| Theft FROM Vehicle | BELTLINE | BELTLINE | BELTLINE | BELTLINE | BELTLINE | BELTLINE |
| Theft OF Vehicle | BELTLINE | BELTLINE | BELTLINE | BELTLINE | FOREST LAWN | BELTLINE |
| Violence Other (Non-domestic) | DOWNTOWN COMMERCIAL CORE | DOWNTOWN COMMERCIAL CORE | BELTLINE | BELTLINE | BELTLINE | BELTLINE |

Discussion: The findings suggest that certain types of crimes, such as Assault (Non-domestic), Break & Enter-Commercial, Commercial Robbery, Street Robbery, Theft FROM Vehicle, Theft OF Vehicle, and Violence Other (Non-domestic), are more prevalent in the Beltline and downtown commercial core communities at the Center sector. This information can be used by law enforcement agencies and community organizations to allocate resources more effectively and implement targeted crime prevention strategies in these areas. Additionally, the observation that the communities where Break & Enter-Dwelling and Break & Enter-Other-Premises occurred most frequently differed each year indicates that these types of crimes may be influenced by different factors or may occur in different areas over time. Understanding these patterns can help authorities better understand the dynamics of these crimes and tailor interventions to address specific community needs.

To Query the Crime Count Over Time

In this query, we investigated whether there are discernible patterns in the time trends of annual crime count within a year. We hypothesized that these patterns might reveal seasonality, such as increased crime rates during holiday seasons or periods of heightened mobility and activity among people.

In [31]:

```
query = """
WITH CrimeCounts AS (
    SELECT
        Year(CRIME_DATE) as Year,
        Month(CRIME_DATE) as Month,
        CAST(SUM(Crime_Count) AS DECIMAL(10,2) ) AS Total_Crime_Count
    FROM
        `101-4`.crime_statistics
    GROUP BY
        Year,
        Month
),
TotalCounts AS (
    SELECT
        Year(CRIME_DATE) as Year,
        SUM(Crime_Count) AS Total_Crime_Count
    FROM
        `101-4`.crime_statistics
    GROUP BY
        Year
)
SELECT
    T1.Year,
    T1.Month,
    T1.Total_Crime_Count,
    T2.Total_Crime_Count
FROM
    CrimeCounts T1
JOIN
    TotalCounts T2
    ON T1.Year = T2.Year
    AND T1.Month = T2.Month
ORDER BY
    T1.Year,
    T1.Month
    
```

```

`101-4`.crime_statistics
GROUP BY
    Year
),
ResidentCounts AS (
    SELECT
        Year,
        SUM(Resident_Count) AS Total_Resident_Count
FROM
(
    SELECT
        DISTINCT Year(CRIME_DATE) as Year,
        COMM_CODE,
        Resident_Count
    FROM
        `101-4`.crime_statistics
    ) AS subquery
GROUP BY
    Year
)
SELECT
    c.Year,
    c.Month,
    c.Total_Crime_Count,
    tc.Total_Crime_Count AS Total_Count_Per_Year,
    ROUND((c.Total_Crime_Count / tc.Total_Crime_Count) * 100, 2) AS Proportion,
    rc.Total_Resident_Count,
    ROUND((c.Total_Crime_Count / rc.Total_Resident_Count) * 100000, 2) AS Crime_Rate_Per_100000
FROM
    CrimeCounts c
JOIN
    TotalCounts tc ON c.Year = tc.Year
JOIN
    ResidentCounts rc ON c.Year = rc.Year
ORDER BY
    c.Year,
    FIELD(c.Month, 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov',
"""
query_df = pd.read_sql_query(query, engine)
display(query_df)

```

| | Year | Month | Total_Crime_Count | Total_Count_Per_Year | Proportion | Total_Resident_Count | Crime_Rate_Per_1000 |
|-----|------|-------|-------------------|----------------------|------------|----------------------|---------------------|
| 0 | 2018 | 3 | 2571.0 | 35339.0 | 7.28 | 1280324.0 | 200 |
| 1 | 2018 | 4 | 2641.0 | 35339.0 | 7.47 | 1280324.0 | 200 |
| 2 | 2018 | 5 | 3060.0 | 35339.0 | 8.66 | 1280324.0 | 230 |
| 3 | 2018 | 6 | 2935.0 | 35339.0 | 8.31 | 1280324.0 | 220 |
| 4 | 2018 | 7 | 3451.0 | 35339.0 | 9.77 | 1280324.0 | 260 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 67 | 2023 | 2 | 2058.0 | 27296.0 | 7.54 | 1283177.0 | 160 |
| 68 | 2023 | 3 | 2547.0 | 27296.0 | 9.33 | 1283177.0 | 190 |
| 69 | 2023 | 4 | 2294.0 | 27296.0 | 8.40 | 1283177.0 | 170 |
| 70 | 2023 | 5 | 2523.0 | 27296.0 | 9.24 | 1283177.0 | 190 |
| 71 | 2023 | 6 | 2574.0 | 27296.0 | 9.43 | 1283177.0 | 200 |

72 rows × 7 columns

In [32]: query_df = pd.read_sql_query(query, engine)

```

crime_monthly = query_df.groupby([query_df['Year'], query_df['Month']])['Total_Crime_Count'].sum

# Plot the data
plt.figure(figsize=(12, 6))
for year, query_df in crime_monthly.groupby('Year'):
    plt.plot(query_df['Month'], query_df['Total_Crime_Count'], label=year)

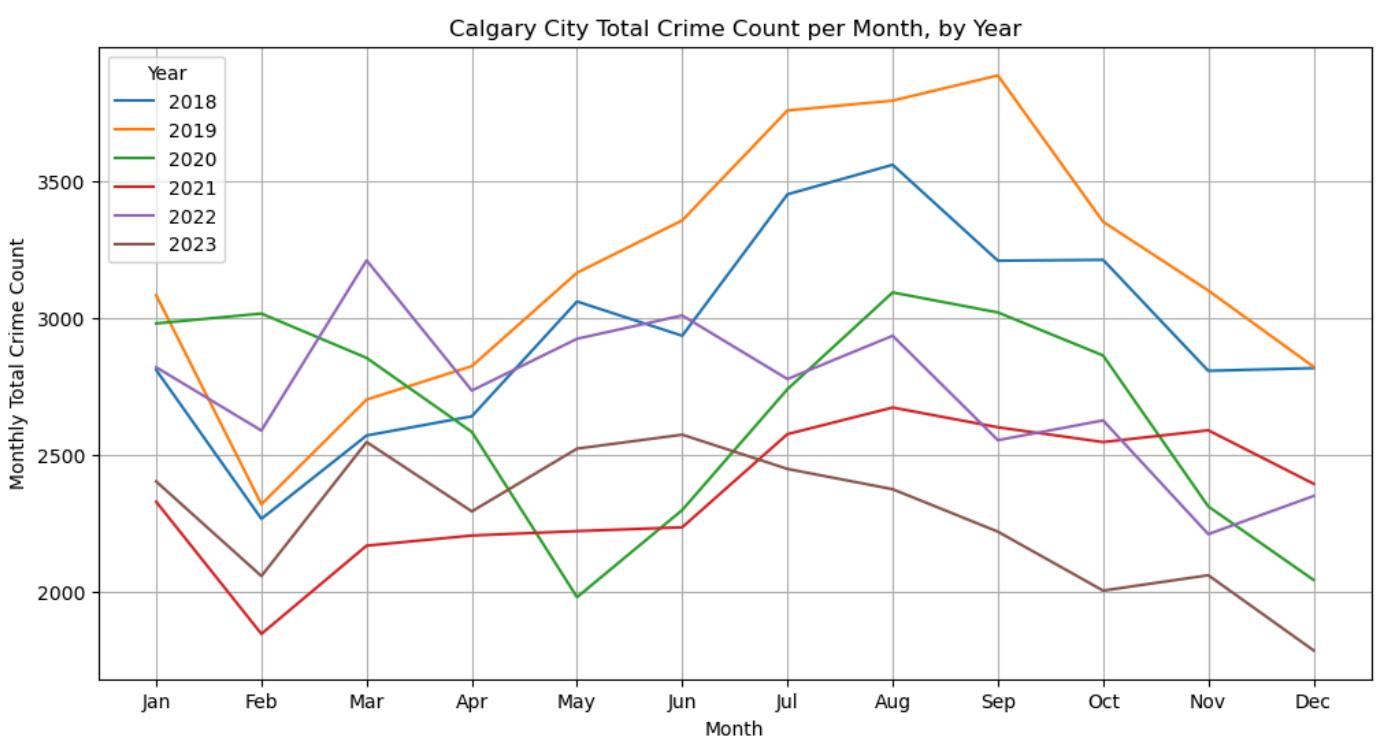
plt.xlabel('Month')
plt.ylabel('Monthly Total Crime Count')
plt.title('Calgary City Total Crime Count per Month, by Year')

```

```

plt.legend(title='Year', loc='upper left')
plt.xticks(range(1, 13), [ 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct',
plt.grid(True)
plt.show()

```



Discussion: Compared to the other months of the year, July to September were found to have higher monthly crime counts. This finding has some implications. There may be seasonal trends in crime, with more incidents occurring during the summer months. This could be due to various factors such as warmer weather, more outdoor activities, and increased social interactions. Law enforcement agencies may need to allocate more resources, such as personnel and patrols, during the summer months to address the higher crime rates effectively. Communities and law enforcement agencies could implement targeted preventive measures and campaigns during the summer months to reduce crime, such as increased community policing and public awareness campaigns. The findings could inform policy decisions related to crime prevention and public safety, such as the timing of resource allocation and the implementation of specific programs and initiatives. Future studies for crime patterns may focus more on understanding the underlying reasons for the seasonal variation in crime rates and how to address them effectively.

A Focus on LRT Transit Station Proximities with Crime Statistics (Alan)

To Query the Frequency of Crime by Shortest Distance to the LRT

A SQL View, `vw_community_distances`, was used in this analysis, this is the same SQL view explored in the EDA section and now used in our Guiding Questions section.

```

In [33]: crime_dist_lrt_table = pd.read_sql_query("""
SELECT crime_id, dist.shortest_distance_to_lrt_meters FROM
crime_statistics cs
LEFT JOIN vw_community_distances dist ON cs.COMM_CODE = dist.comm_code
WHERE 1=1
AND YEAR(CRIME_DATE) BETWEEN 2018 and 2023
""", engine)

# Function to round to the nearest 500 - we want nice bin labels to look at
def round_to_nearest_500(number):
    return round(number / 500) * 500

# Determine min and max, then round them to the nearest 500
min_distance = round_to_nearest_500(crime_dist_lrt_table['SHORTEST_DISTANCE_TO_LRT_METERS'].min())
max_distance = round_to_nearest_500(crime_dist_lrt_table['SHORTEST_DISTANCE_TO_LRT_METERS'].max())

# If min_distance is equal to max_distance after rounding, adjust max_distance, nudging the bins
if min_distance == max_distance:
    max_distance += 500

# Number of bins - digestible number of bins on a frequency chart
num_bins = 6

```

```

# Calculate equally spaced bins between min and max, then round
raw_bins = np.linspace(min_distance, max_distance, num_bins+1)
bins = np.array([round_to_nearest_500(edge) for edge in raw_bins])

# Adjust bins to ensure unique bin edges
bins = np.unique(bins)
# If bins got reduced due to rounding, extend it to ensure 6 unique bins
while len(bins) < num_bins + 1:
    max_distance += 500
    raw_bins = np.linspace(min_distance, max_distance, num_bins+1)
    bins = np.array([round_to_nearest_500(edge) for edge in raw_bins])
    bins = np.unique(bins)

# Generate bin labels based on the bins
bin_labels = [f"{int(bins[i])}-{int(bins[i+1])}" for i in range(len(bins)-1)]

# Assign data to bins
crime_dist_lrt_table['distance_bin'] = pd.cut(crime_dist_lrt_table['SHORTEST_DISTANCE_TO_LRT_MET']

# Create the frequency distribution
frequency_distribution = crime_dist_lrt_table['distance_bin'].value_counts().sort_index()
# Calculate percentages
frequency_distribution_percent = frequency_distribution / frequency_distribution.sum() * 100

```

In [34]:

```

# Set up a figure with two subplots side by side
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 8)) # figsize might need adjustment

# Plot the bar plot on the first subplot
frequency_distribution.plot(kind='bar', ax=ax1)
ax1.set_title('Frequency Distribution of Crime by Shortest Distance to LRT Between 2018 and 2023')
ax1.set_xlabel('Distance to LRT (meters)')
ax1.set_ylabel('Frequency')
ax1.tick_params(axis='x', rotation=45)

# Plot the pie chart on the second subplot
wedges, texts, autotexts = ax2.pie(frequency_distribution_percent, autopct='%.1f%%', startangle=90)

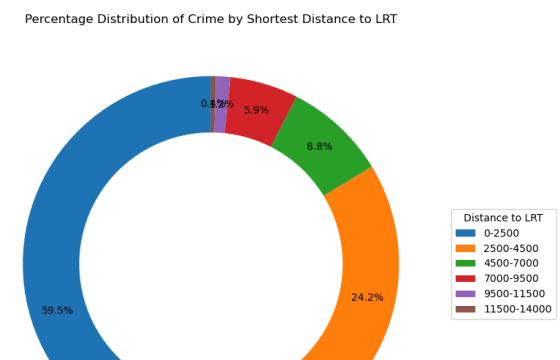
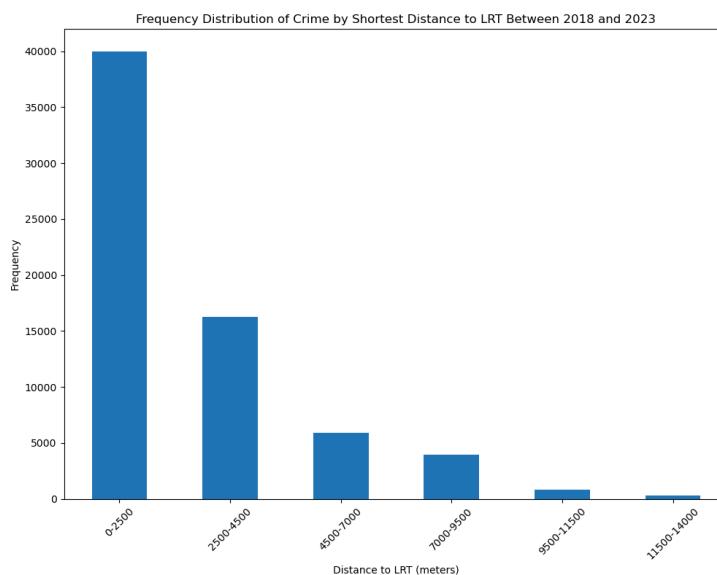
# Draw a circle at the center of pie to make it look like a donut
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig.gca().add_artist(centre_circle)

# Hide the texts on the pie chart and set up the legend
for text in texts:
    text.set_visible(False)
ax2.legend(wedges, frequency_distribution_percent.index, title="Distance to LRT", loc="center left")
ax2.set_title('Percentage Distribution of Crime by Shortest Distance to LRT')

# Adjust layout to make room for the legend
plt.tight_layout()

# Show the figure with both subplots
plt.show()

```



Discussion: In our analysis, we focused on visualizing the geographical relationship between crime and their proximity to the LRT (Light Rail Transit) stations. We began by utilizing the central points of Calgary communities where crimes were reported. We then calculated the distances from these central points to the nearest LRT stations. Our goal was to categorize these distances into six distinct bins, which allowed us to

distribute the crime incidents based on their proximity to LRT infrastructure. This approach initially leads us to conclude that LRT proximity has a strong influence on crime incidence.

To Query Crime Count by Resident Count and by Number of C-Trains

We utilize a bubble chart to analyze three different data points in the Crime Count, the Resident Count and Number of C-Trains within the Community's boundary. Following feedback from colleagues during the presentation, we have applied a logarithmic transformation to the Crime Count to optimize chart space and enhance the clarity of the data points for viewers.

```
In [35]: crime_num_lrt_table = pd.read_sql_query("""
WITH TransitStations AS (
    SELECT
        community.comm_code AS "CommunityCode",
        COUNT(transit.STATION_NAME) AS "NumberOfTransitStations"
    FROM
        transit_lrt_stations transit
        JOIN communities community ON Contains(community.COMMUNITY_BOUNDARY, transit.LRT_POINT)
    GROUP BY
        community.comm_code
),
Top10Communities AS (
    SELECT
        top10.*
        ,dist.SHORTEST_DISTANCE_TO_LRT_METERS as "Distance to LRT (m)"
    FROM
        vw_annual_top10_communities_by_crime_count top10
        LEFT JOIN vw_community_distances dist ON top10.comm_code = dist.comm_code
)
SELECT
    t10.Year,
    t10.Rank,
    t10.Comm_Code,
    t10.Community_name as "Community_Name",
    t10.Total_Crime_Count,
    t10.Resident_Count as "Resident_Count",
    t10.Crime_Rate_Per_100k,
    t10.Crime_Count,
    t10.`Distance to LRT (m)` ,
    ts.NumberOfTransitStations as "# of LRTs"
    FROM
        Top10Communities t10
        LEFT JOIN TransitStations ts ON t10.COMM_CODE = ts.CommunityCode
"""
, engine)
```

```
In [54]: # Replace NaN values in NumberOfTransitStations for visualization purposes
crime_num_lrt_table['# of LRTs'].fillna(0, inplace=True)
crime_num_lrt_table['Crime_Rate_Per_100k'].fillna(0, inplace=True)
# Replace 0 with a small value for log transformation, if necessary
crime_num_lrt_table['Crime_Count'] = crime_num_lrt_table['Crime_Count'].replace(0, np.nextafter(0, 1))

# Apply log transformation to the 'Crime_Count'
crime_num_lrt_table['Log_Crime_Count'] = np.log(crime_num_lrt_table['Crime_Count'])

# Get the unique years and sort them if necessary
years = sorted(crime_num_lrt_table['Year'].unique())

# Setup the matplotlib figure - adjust figsize as needed, now with rows for each year
fig_height = 5 * len(years) # Increasing height for each subplot
fig, axes = plt.subplots(len(years), 1, figsize=(8, fig_height), sharex=True)

# Ensure axes is iterable for a single year case
if len(years) == 1:
    axes = [axes]

for ax, year in zip(axes, years):
    data_for_year = crime_num_lrt_table[crime_num_lrt_table['Year'] == year]

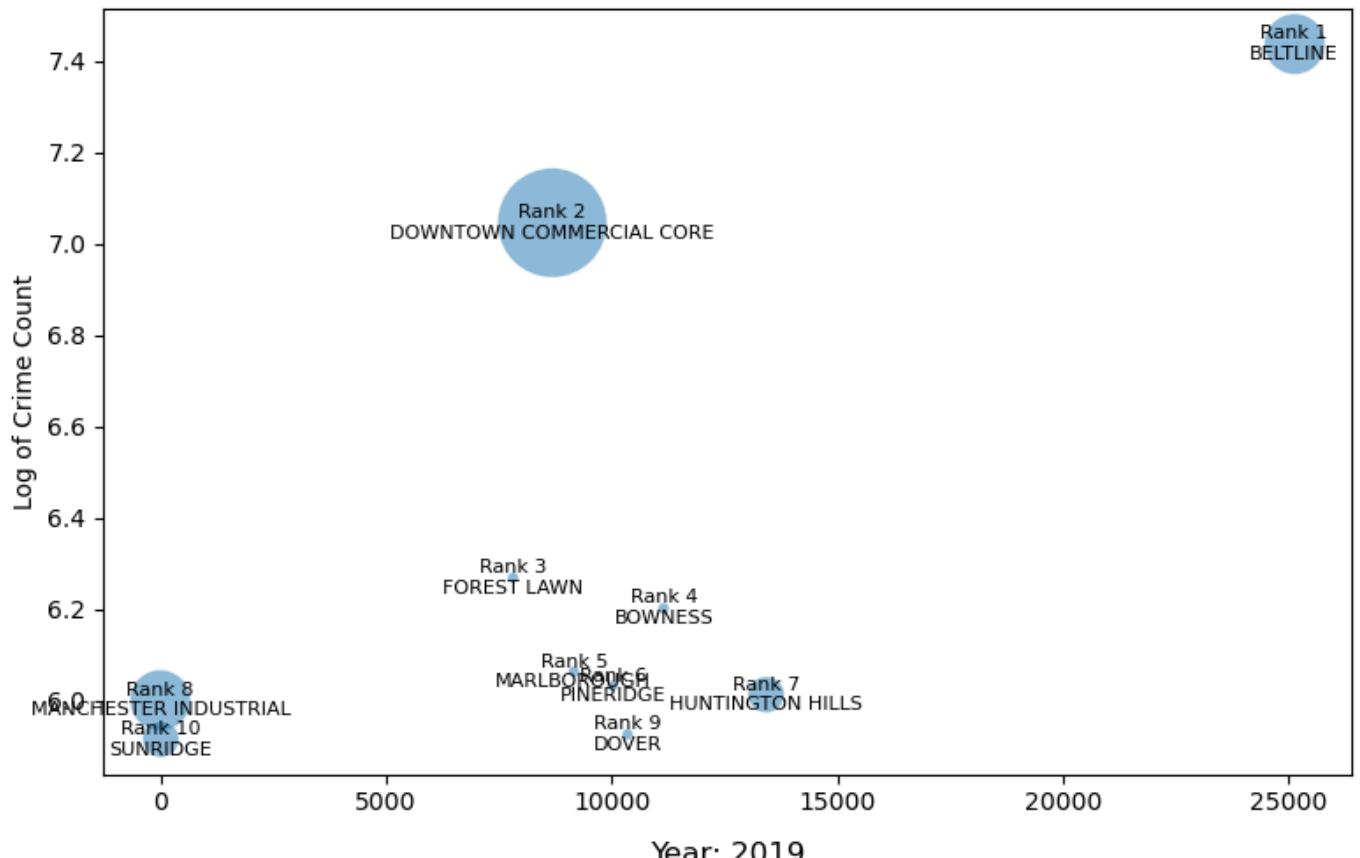
    # Create the scatter plot for this year
    scatter = sns.scatterplot(ax=ax, data=data_for_year, x='Resident_Count', y='Log_Crime_Count',
                             size='# of LRTs', sizes=(20, 2000), alpha=0.5, legend=False)

    # Annotate each bubble with the Rank and Community Name
    for _, row in data_for_year.iterrows():
        label = f"Rank {row['Rank']}\n{row['Community_Name']}"
        ax.text(row['Resident_Count'], row['Log_Crime_Count'], label)
```

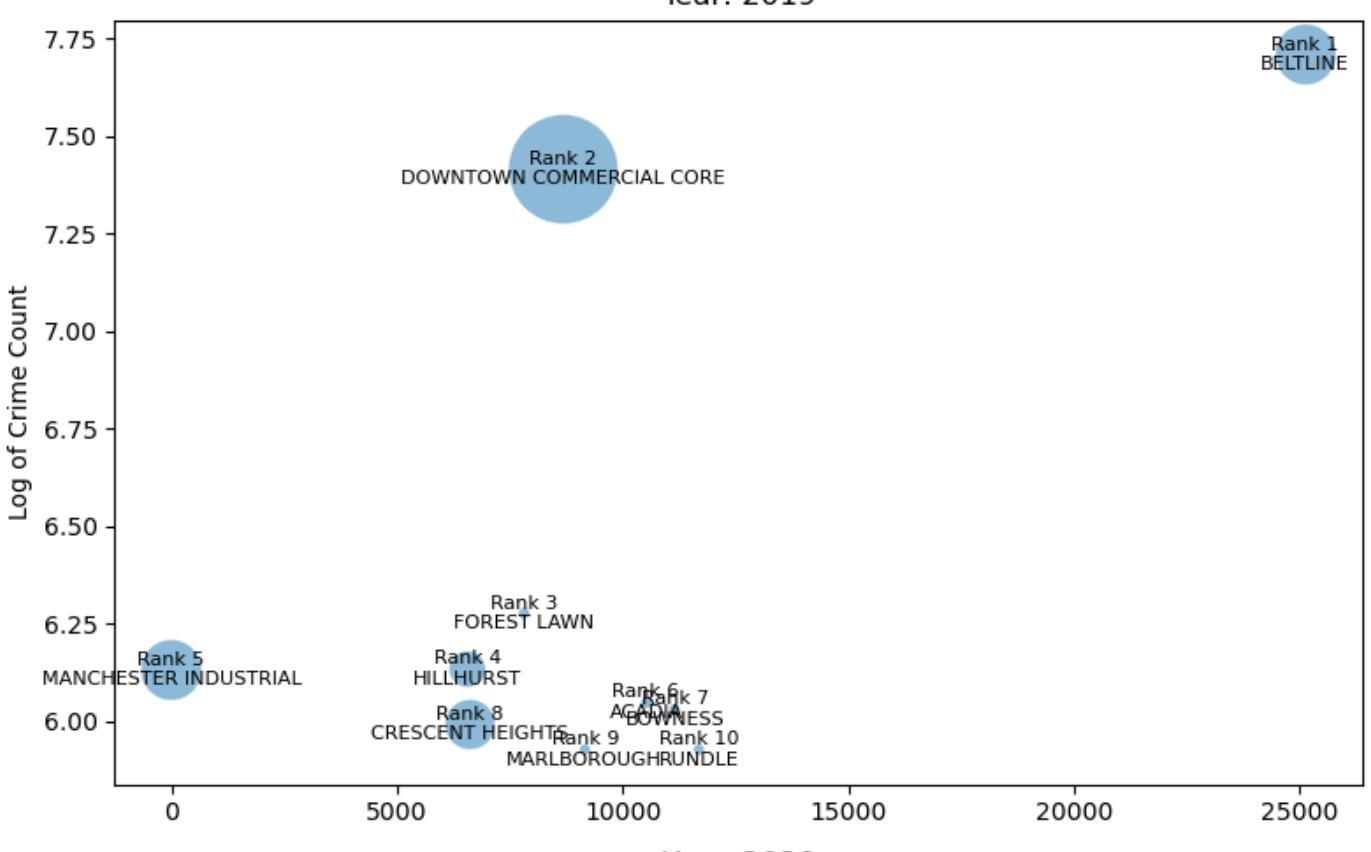
```
    ax.text(row['Resident_Count'], row['Log_Crime_Count'], label,
            horizontalalignment='center', verticalalignment='center',
            color='black', fontsize=8, ha='center')
    # Make x-axis labels visible for each subplot
    ax.tick_params(axis='x', which='both', bottom=True, top=False, labelbottom=True)
    ax.set_title(f'Year: {year}')
    ax.set_xlabel('Resident Count')
    ax.set_ylabel('Log of Crime Count')

# Adjust layout to not overlap
plt.tight_layout()
plt.show()
```

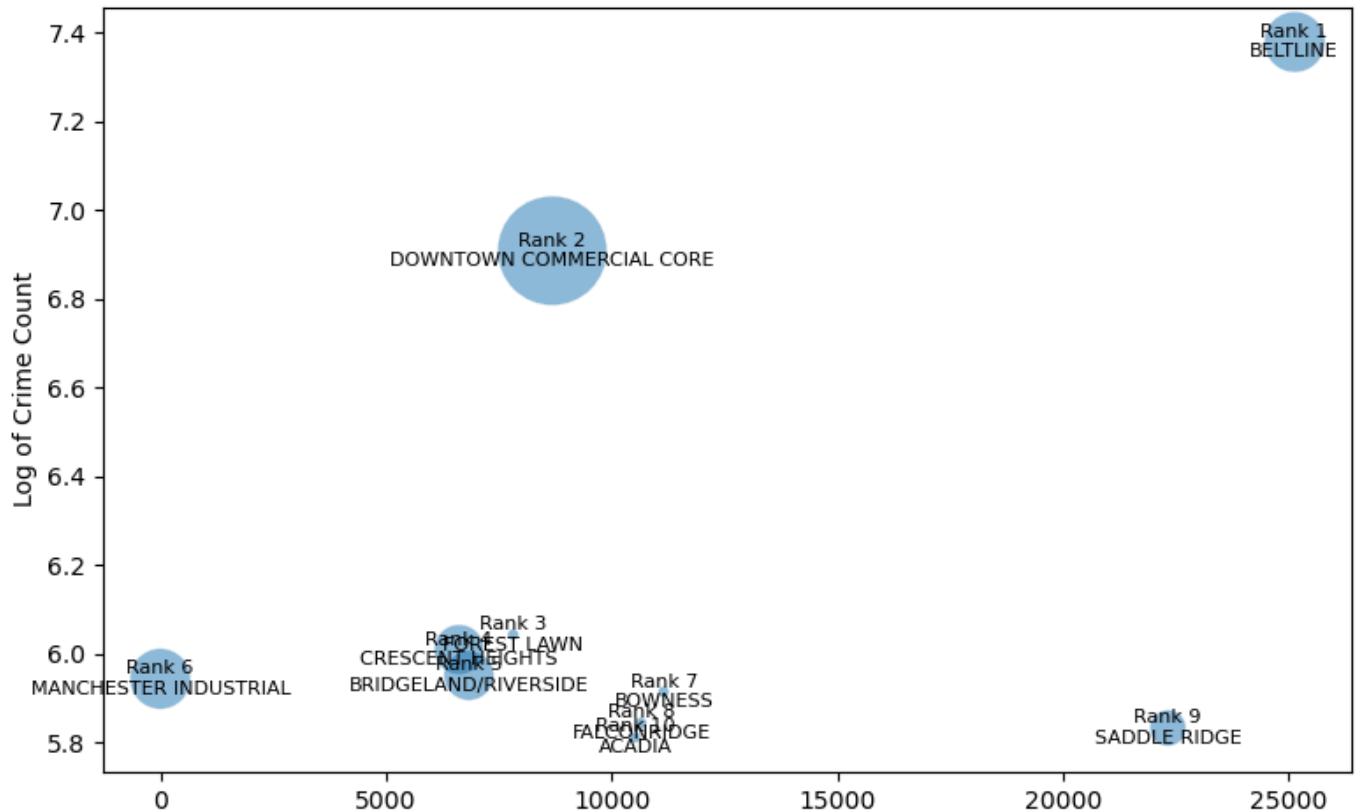
Year: 2018



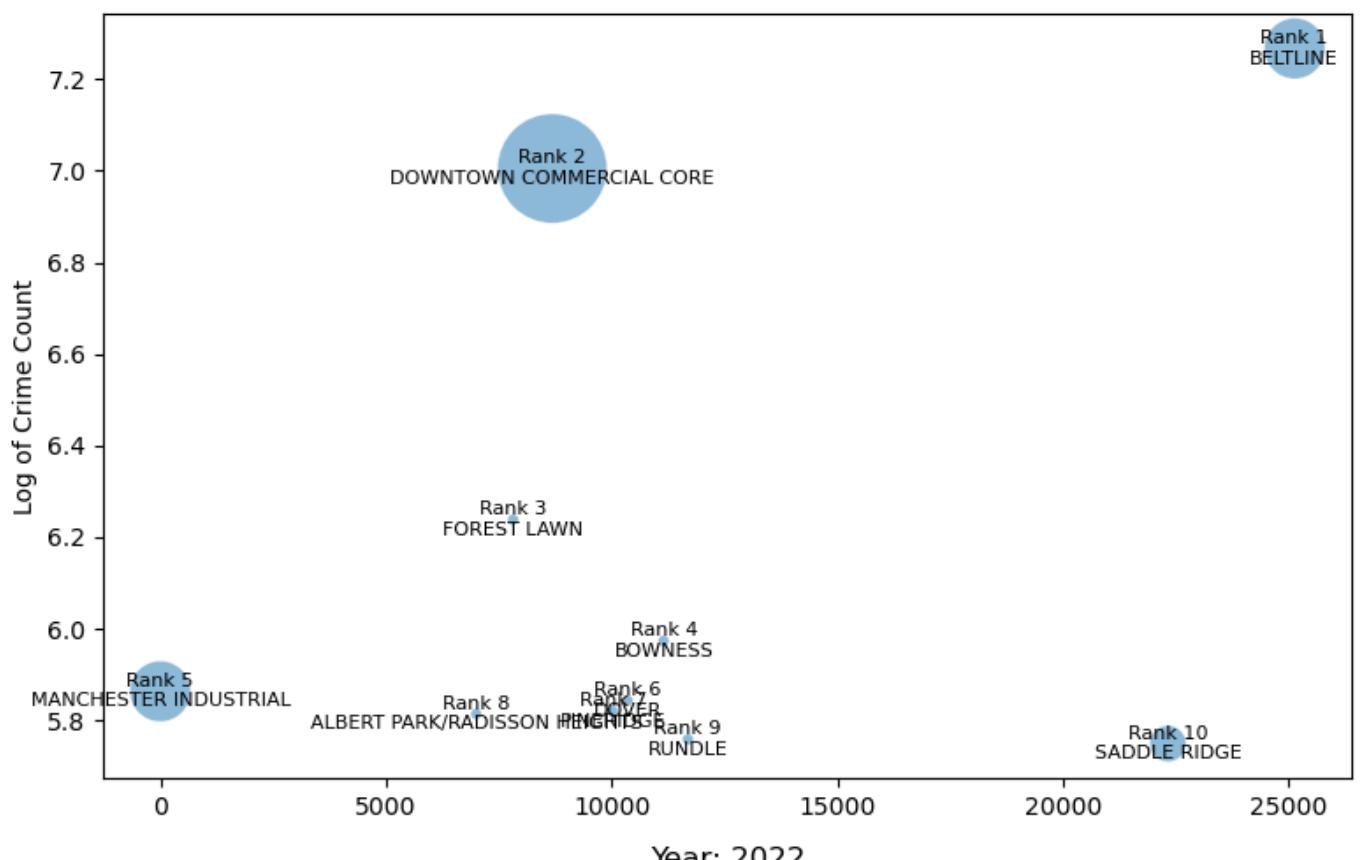
Year: 2019



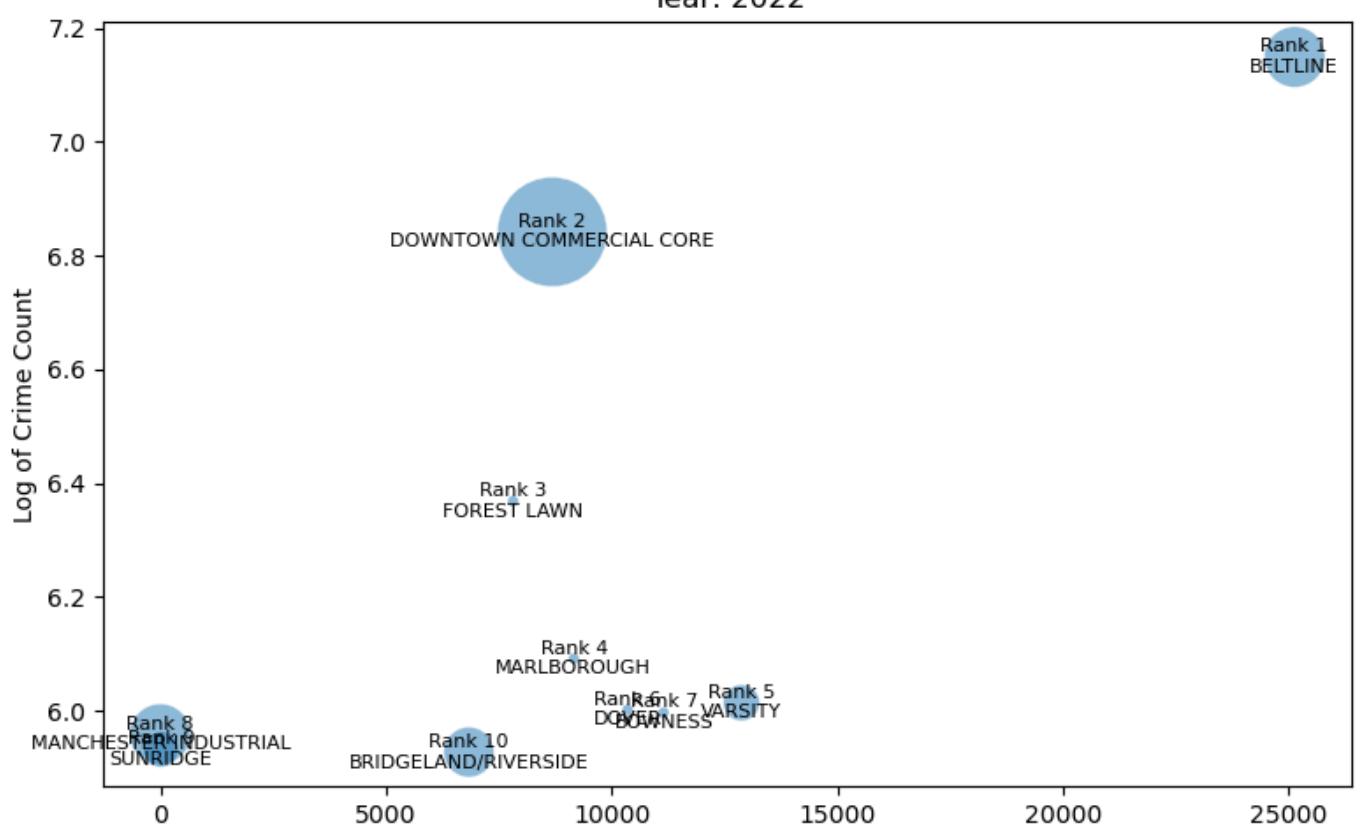
Year: 2020



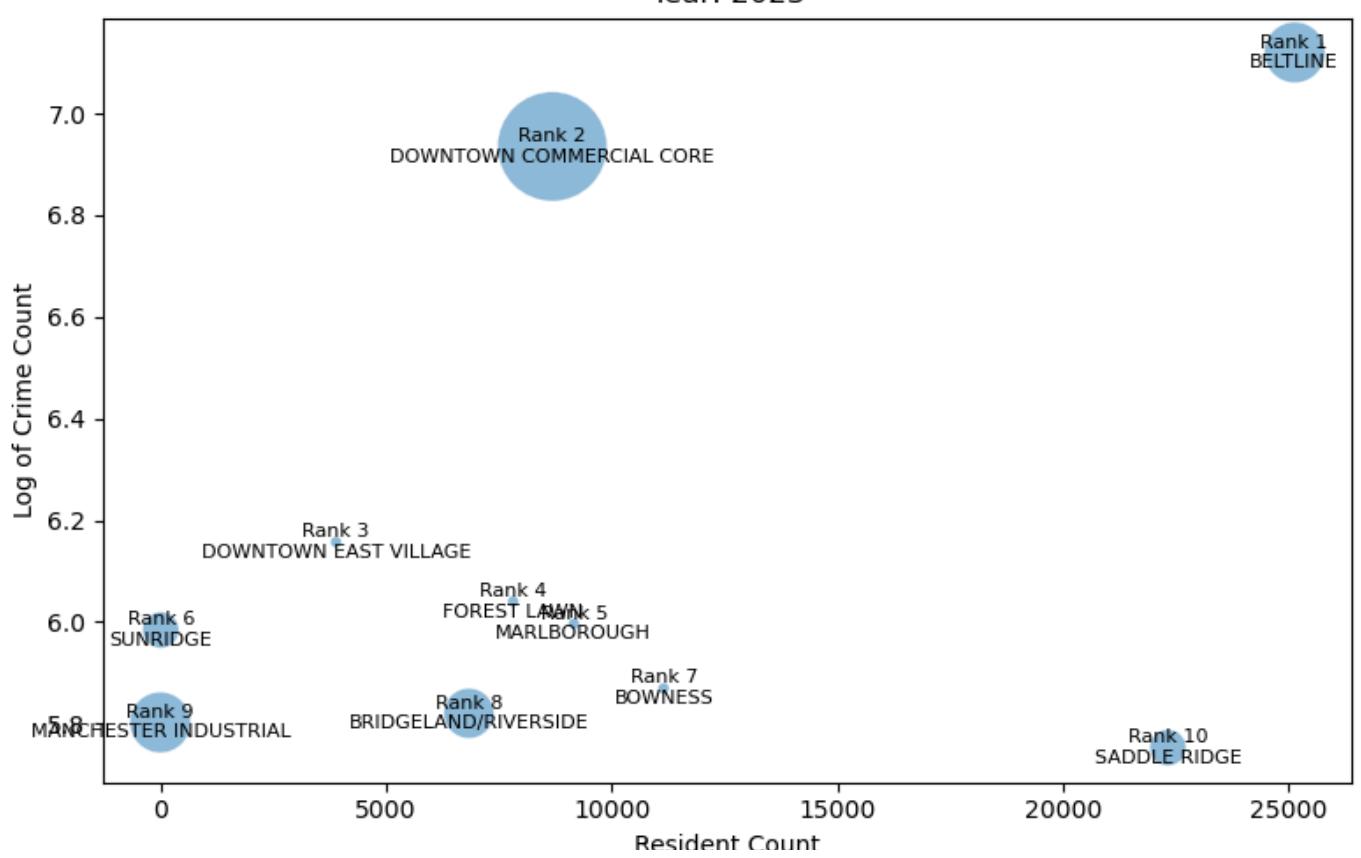
Year: 2021



Year: 2022



Year: 2023



In []:

Discussion: Upon examining the bubble charts, it becomes evident that the Resident Count within a community significantly impacts the Crime Count. This suggests that factors beyond mere proximity to LRT stations contribute to the observed crime rates. A limitation to this visualization is we are purposefully querying for top 10 communities to not clutter and obfuscate the bubble chart.

To Query Crime Rate per 100,000 Residents, by Shortest Distance to LRT and by Number of C-Trains

We utilize a bubble chart to analyze three different data points in the Crime Rate per 100k, this should allow us to indirectly measure against crime count and resident count, Shortest Distance to LRT, and the Number of C-Trains within the Community's boundary. We have applied a logarithmic transformation to the Crime Rate per 100k to optimize chart space and enhance the clarity of the data points for viewers.

In [37]:

```
# Replace NaN values in NumberOfTransitStations for visualization purposes
crime_num_lrt_table['# of LRTs'].fillna(0, inplace=True)
crime_num_lrt_table['Crime_Rate_Per_100k'].fillna(0, inplace=True)
# Apply Log transformation to the 'Crime Rate'
# Replace 0 with a small value for log transformation, if necessary
#crime_num_lrt_table['Crime_Rate_Per_100k'] = crime_num_lrt_table['Crime_Rate_Per_100k'].replace(0, 1)
crime_num_lrt_table['Log_Crime_Rate_Per_100k'] = np.log(crime_num_lrt_table['Crime_Rate_Per_100k'])
crime_num_lrt_table.replace(-np.inf, 0, inplace=True)
```

C:\Users\notso\anaconda3\Lib\site-packages\pandas\core\arraylike.py:396: RuntimeWarning: divide by zero encountered in log
result = getattr(ufunc, method)(*inputs, **kwargs)

In [55]:

```
# Get the unique years and sort them if necessary
years = sorted(crime_num_lrt_table['Year'].unique())

# Setup the matplotlib figure - adjust figsize as needed, now with rows for each year
# Setup the matplotlib figure - adjust figsize as needed, now with rows for each year
fig_height = 5 * len(years) # Increasing height for each subplot
fig, axes = plt.subplots(len(years), 1, figsize=(8, fig_height), sharex=True)

# Ensure axes is iterable for a single year case
if len(years) == 1:
    axes = [axes]

for ax, year in zip(axes, years):
    data_for_year = crime_num_lrt_table[crime_num_lrt_table['Year'] == year]

    # Create the scatter plot for this year
    sns.scatterplot(ax=ax, data=data_for_year, x='Distance to LRT (m)', y='Log_Crime_Rate_Per_100k',
                    size='# of LRTs', sizes=(20, 2000), alpha=0.5, legend=False)

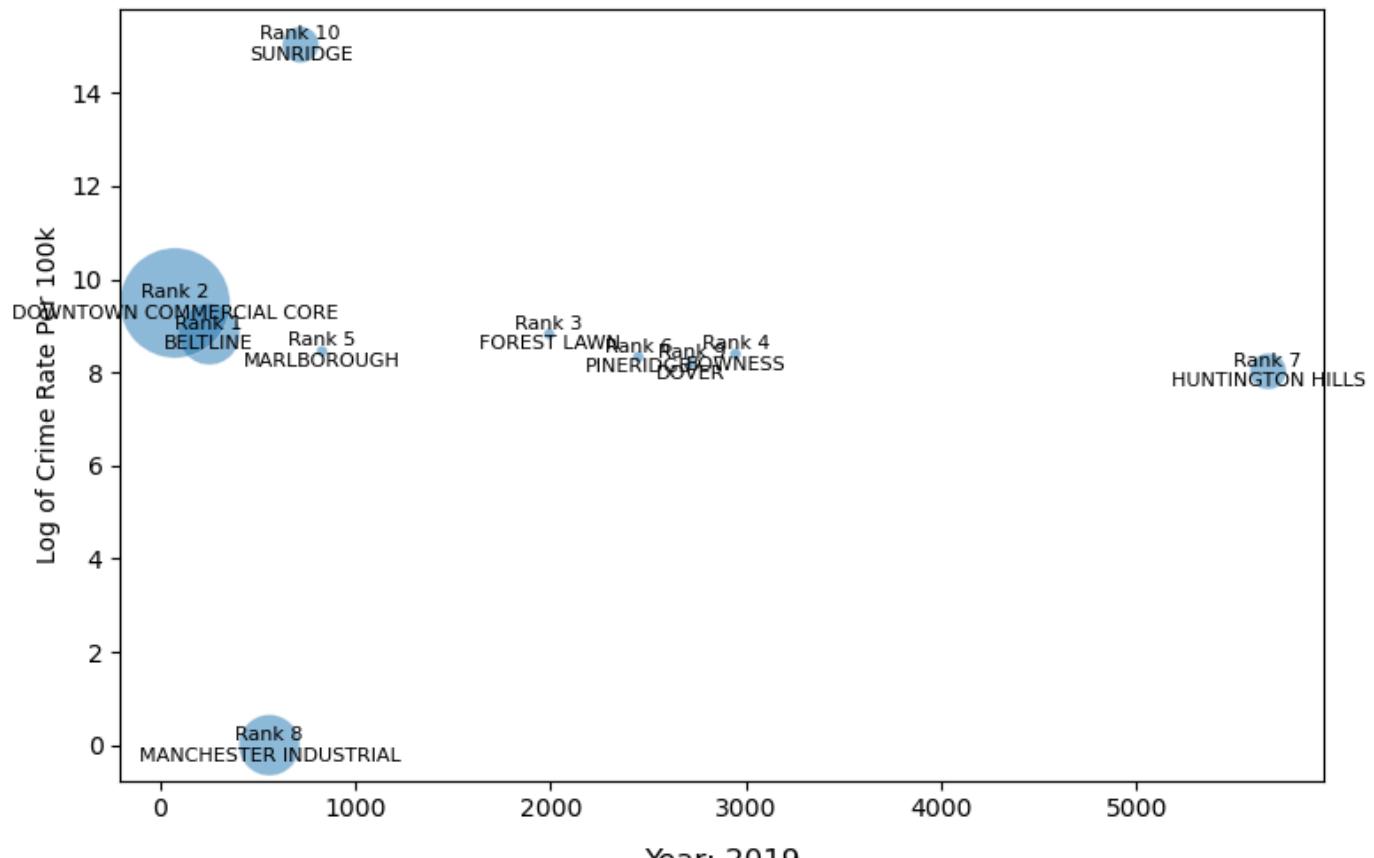
    # Annotate each bubble with the Rank and Community Name
    for _, row in data_for_year.iterrows():
        label = f"Rank {row['Rank']}\n{row['Community_Name']}"
        ax.text(row['Distance to LRT (m)'], row['Log_Crime_Rate_Per_100k'], label,
                horizontalalignment='center', verticalalignment='center',
                color='black', fontsize=8, ha='center')

    ax.set_title(f'Year: {year}')
    ax.set_xlabel('Distance to LRT (m)')
    ax.set_ylabel('Log of Crime Rate Per 100k')

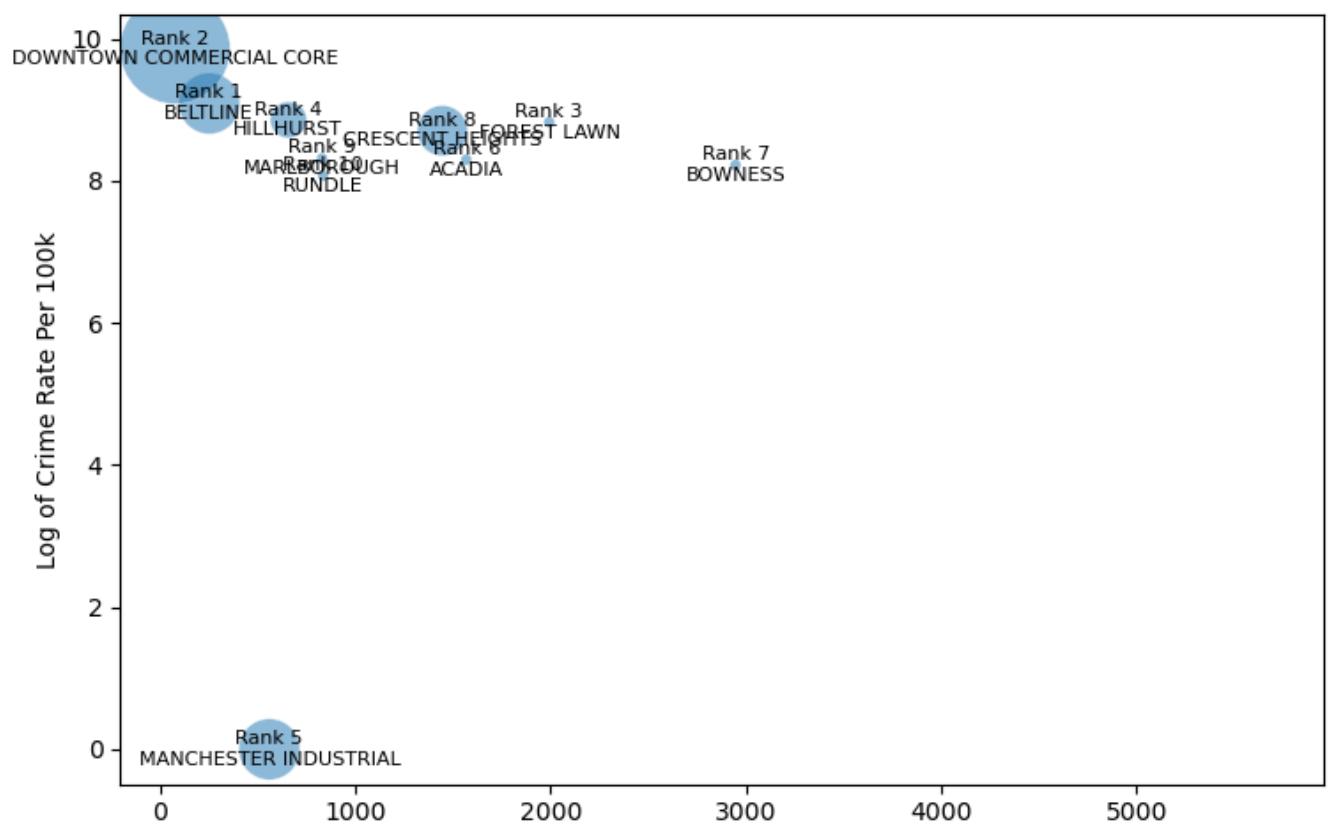
    # Make x-axis labels visible for each subplot
    ax.tick_params(axis='x', which='both', bottom=True, top=False, labelbottom=True)

# Adjust Layout to not overlap
plt.tight_layout()
plt.show()
```

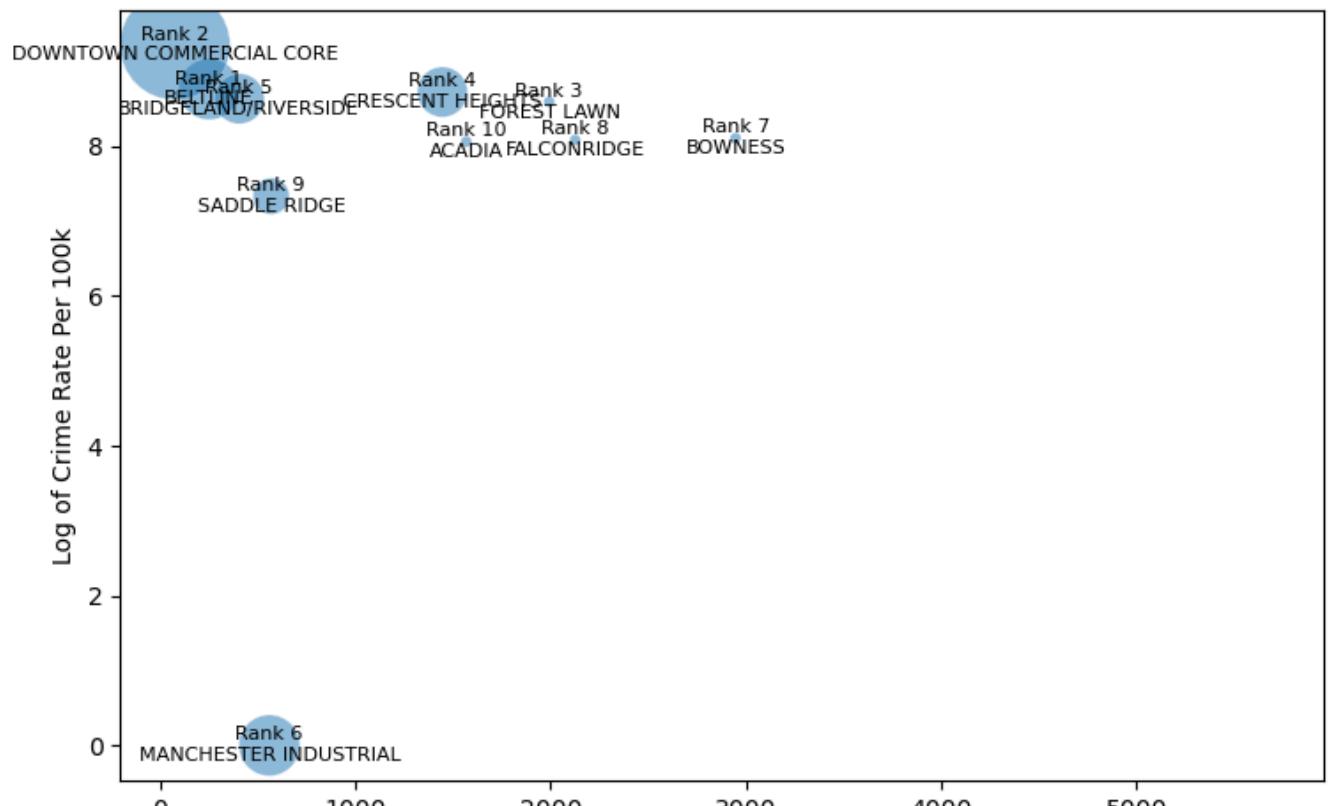
Year: 2018



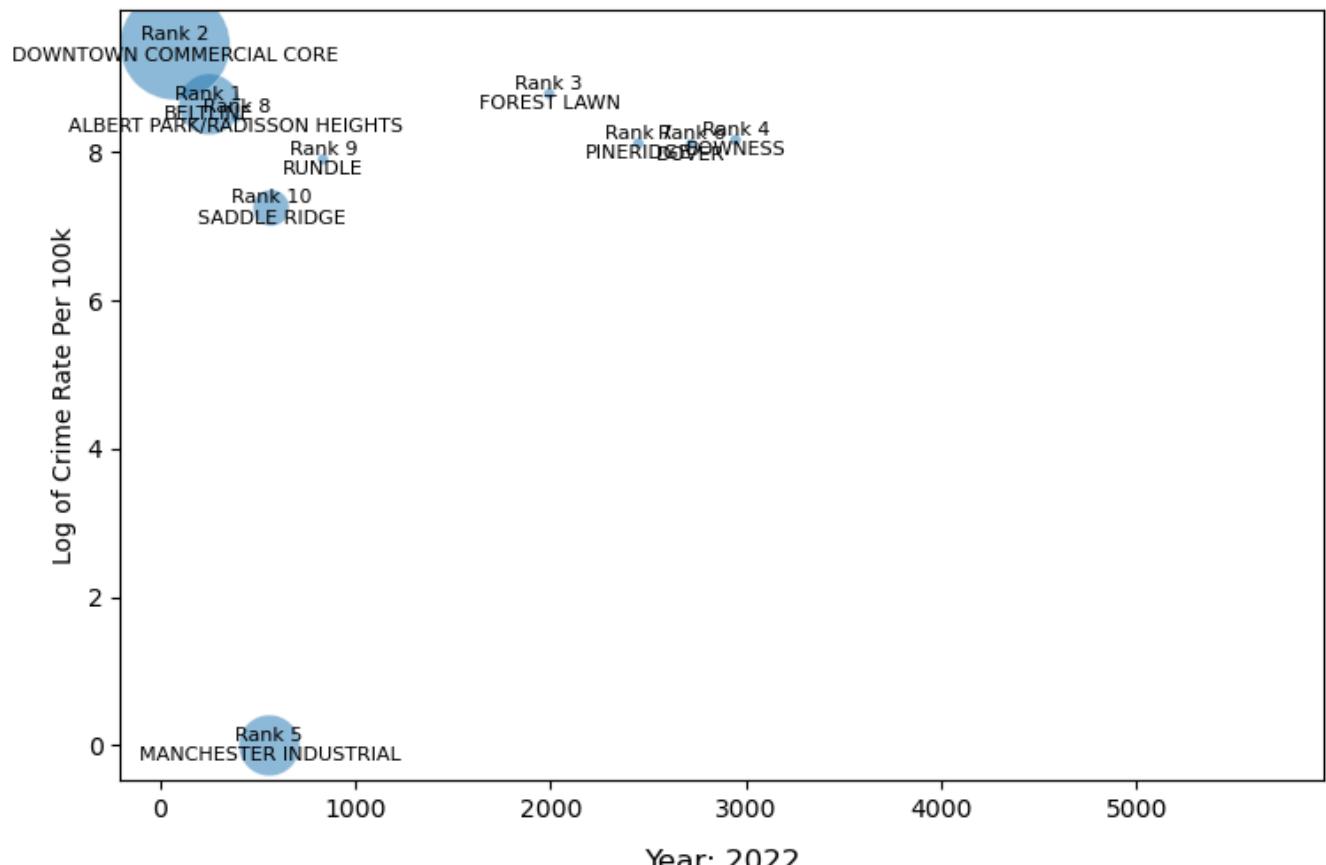
Year: 2019



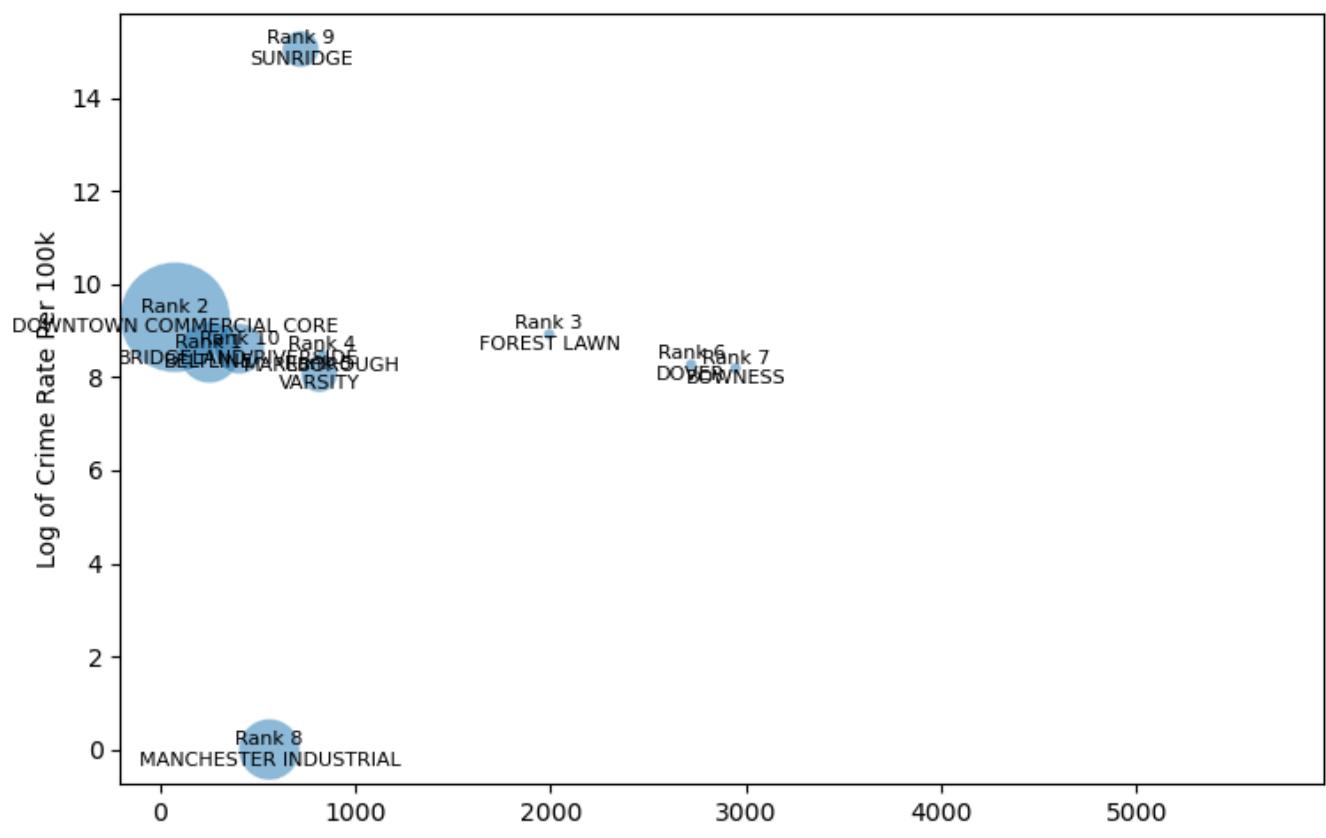
Year: 2020



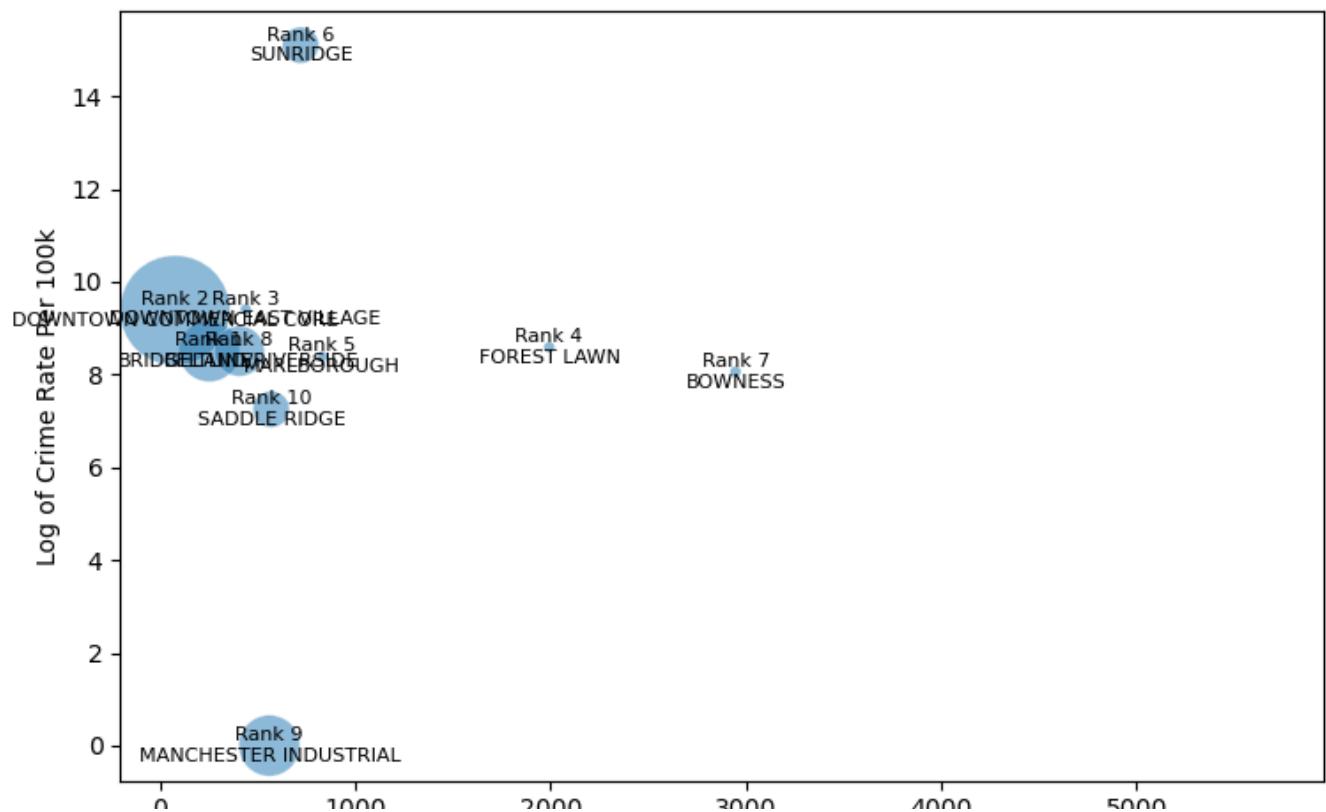
Year: 2021



Year: 2022



Year: 2023



In []:

Discussion: If proximity to LRT stations was the predominant factor affecting crime rates, we would expect to see the majority of the bubbles clustered to the left of the chart, signifying shorter distances to LRT stations. Instead, the distribution of the top 10 crime-leading communities is dispersed across the chart, with several positioned towards the far right, indicating a greater distance to LRT stations.

This observation leads to the conclusion that while LRT station proximity may influence crime statistics, it is not the sole factor. The Resident Count also appears to play a significant role, suggesting the presence of additional underlying factors. Therefore, for a more comprehensive understanding of the factors influencing crime rates, further investigation into other factors is warranted such as economic indicators.

A Focus on Economic Indicators with Crime Statistics (Zane)

To Query the Unemployment rate and Crime rate

In this query, I will explore the relationship between unemployment rate and crime rate in calgary. I will use monthly data to compare the trends of unemployment rate and crime rate. The time scale is from 2018 to 2023.

In [39]:

```
query1 = '''SELECT un.year, un.month, un.un_rate AS unemployment_rate, cr.Crime_Rate_Per_100000/
FROM unemployment_rate un JOIN vw_city_monthly_crime_count_rate cr ON un.year = cr.Year AND un.m
ORDER BY un.year, un.month ASC'''
query1_df = pd.read_sql_query(query1, engine)
print("The monthly unemployment rate and crime rate in calgary from 2018 to 2023")
print(ta(query1_df, headers='keys', tablefmt='psql'))
```

The monthly unemployment rate and crime rate in calgary from 2018 to 2023

| | year | month | unemployment_rate | crime_rate |
|----|------|-------|-------------------|------------|
| 0 | 2018 | 1 | 0.077 | 0.22 |
| 1 | 2018 | 2 | 0.078 | 0.18 |
| 2 | 2018 | 3 | 0.079 | 0.2 |
| 3 | 2018 | 4 | 0.074 | 0.21 |
| 4 | 2018 | 5 | 0.07 | 0.24 |
| 5 | 2018 | 6 | 0.07 | 0.23 |
| 6 | 2018 | 7 | 0.075 | 0.27 |
| 7 | 2018 | 8 | 0.082 | 0.28 |
| 8 | 2018 | 9 | 0.083 | 0.25 |
| 9 | 2018 | 10 | 0.08 | 0.25 |
| 10 | 2018 | 11 | 0.074 | 0.22 |
| 11 | 2018 | 12 | 0.07 | 0.22 |
| 12 | 2019 | 1 | 0.072 | 0.24 |
| 13 | 2019 | 2 | 0.076 | 0.18 |
| 14 | 2019 | 3 | 0.08 | 0.21 |
| 15 | 2019 | 4 | 0.075 | 0.22 |
| 16 | 2019 | 5 | 0.067 | 0.25 |
| 17 | 2019 | 6 | 0.065 | 0.26 |
| 18 | 2019 | 7 | 0.066 | 0.29 |
| 19 | 2019 | 8 | 0.076 | 0.3 |
| 20 | 2019 | 9 | 0.075 | 0.3 |
| 21 | 2019 | 10 | 0.073 | 0.26 |
| 22 | 2019 | 11 | 0.065 | 0.24 |
| 23 | 2019 | 12 | 0.065 | 0.22 |
| 24 | 2020 | 1 | 0.069 | 0.23 |
| 25 | 2020 | 2 | 0.073 | 0.24 |
| 26 | 2020 | 3 | 0.087 | 0.22 |
| 27 | 2020 | 4 | 0.106 | 0.2 |
| 28 | 2020 | 5 | 0.131 | 0.15 |
| 29 | 2020 | 6 | 0.149 | 0.18 |
| 30 | 2020 | 7 | 0.149 | 0.21 |
| 31 | 2020 | 8 | 0.142 | 0.24 |
| 32 | 2020 | 9 | 0.128 | 0.24 |
| 33 | 2020 | 10 | 0.117 | 0.22 |
| 34 | 2020 | 11 | 0.108 | 0.18 |
| 35 | 2020 | 12 | 0.101 | 0.16 |
| 36 | 2021 | 1 | 0.104 | 0.18 |
| 37 | 2021 | 2 | 0.107 | 0.14 |
| 38 | 2021 | 3 | 0.108 | 0.17 |
| 39 | 2021 | 4 | 0.1 | 0.17 |
| 40 | 2021 | 5 | 0.091 | 0.17 |
| 41 | 2021 | 6 | 0.091 | 0.17 |
| 42 | 2021 | 7 | 0.094 | 0.2 |
| 43 | 2021 | 8 | 0.096 | 0.21 |
| 44 | 2021 | 9 | 0.087 | 0.2 |
| 45 | 2021 | 10 | 0.079 | 0.2 |
| 46 | 2021 | 11 | 0.075 | 0.2 |
| 47 | 2021 | 12 | 0.074 | 0.19 |
| 48 | 2022 | 1 | 0.076 | 0.22 |
| 49 | 2022 | 2 | 0.074 | 0.2 |
| 50 | 2022 | 3 | 0.075 | 0.25 |
| 51 | 2022 | 4 | 0.07 | 0.21 |
| 52 | 2022 | 5 | 0.064 | 0.23 |
| 53 | 2022 | 6 | 0.054 | 0.23 |
| 54 | 2022 | 7 | 0.051 | 0.22 |
| 55 | 2022 | 8 | 0.054 | 0.23 |
| 56 | 2022 | 9 | 0.056 | 0.2 |
| 57 | 2022 | 10 | 0.057 | 0.2 |
| 58 | 2022 | 11 | 0.055 | 0.17 |
| 59 | 2022 | 12 | 0.057 | 0.18 |
| 60 | 2023 | 1 | 0.064 | 0.19 |
| 61 | 2023 | 2 | 0.067 | 0.16 |
| 62 | 2023 | 3 | 0.07 | 0.2 |
| 63 | 2023 | 4 | 0.064 | 0.18 |
| 64 | 2023 | 5 | 0.06 | 0.2 |
| 65 | 2023 | 6 | 0.058 | 0.2 |
| 66 | 2023 | 7 | 0.061 | 0.19 |
| 67 | 2023 | 8 | 0.06 | 0.19 |
| 68 | 2023 | 9 | 0.06 | 0.17 |
| 69 | 2023 | 10 | 0.055 | 0.16 |
| 70 | 2023 | 11 | 0.054 | 0.16 |
| 71 | 2023 | 12 | 0.052 | 0.14 |

In [40]:

```
#visualization
import matplotlib.pyplot as plt
```

```

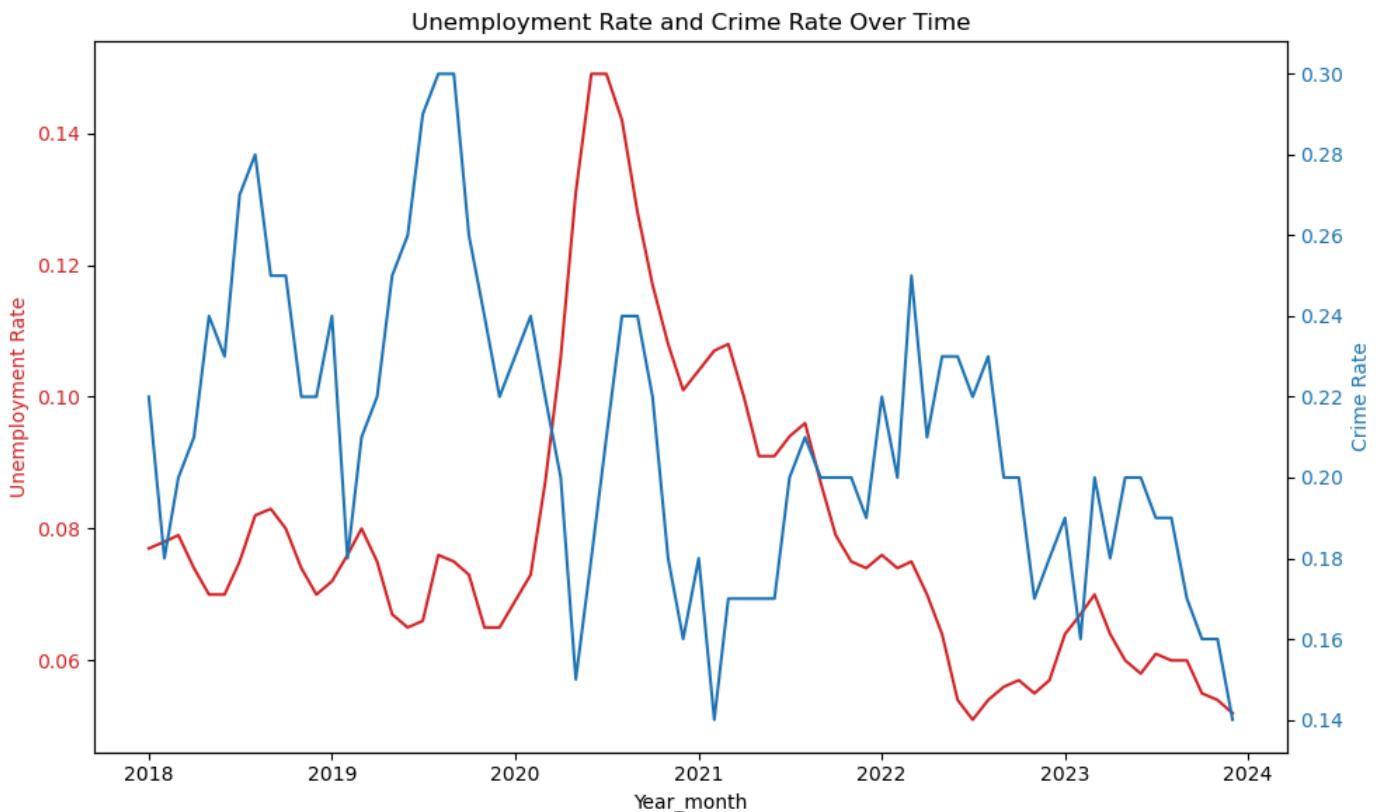
query1_df['Date'] = pd.to_datetime(query1_df['year'].astype(str) + '-' + query1_df['month'].astype(str))

fig, ax1 = plt.subplots(figsize=(10, 6))

color = 'tab:red'
ax1.set_xlabel('Year_month')
ax1.set_ylabel('Unemployment Rate', color=color)
ax1.plot(query1_df['Date'], query1_df['unemployment_rate'], color=color)
ax1.tick_params(axis='y', labelcolor=color)

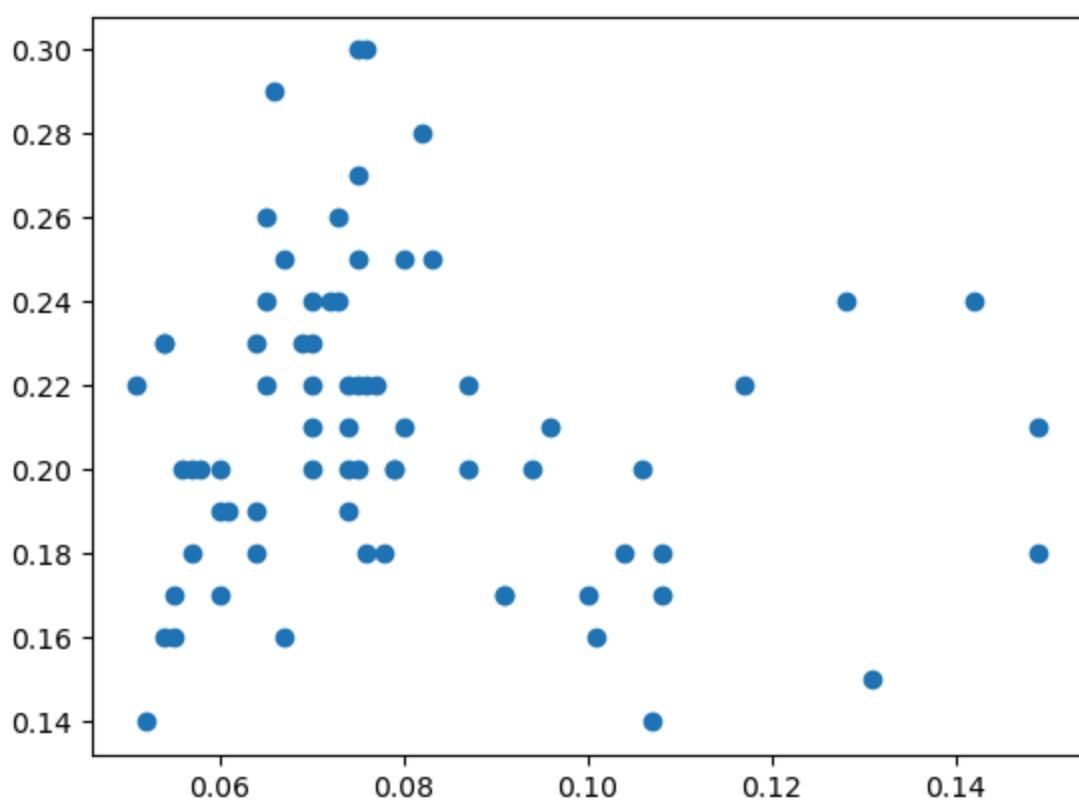
ax2 = ax1.twinx()
color = 'tab:blue'
ax2.set_ylabel('Crime Rate', color=color)
ax2.plot(query1_df['Date'], query1_df['crime_rate'], color=color)
ax2.tick_params(axis='y', labelcolor=color)
plt.title('Unemployment Rate and Crime Rate Over Time')
fig.tight_layout()
plt.show()

```



In [41]: `plt.scatter(query1_df['unemployment_rate'], query1_df['crime_rate'])`

Out[41]: `<matplotlib.collections.PathCollection at 0x21c01f2f890>`



The relationship between the unemployment rate and the crime rate is notably significant, acting as a strong influence on crime trends. Analysis of a line plot reveals periods during which the unemployment rate and crime rate exhibit parallel trends, indicating a correlation between these two factors during specific times.

Additionally, a scatter plot analysis suggests a linear relationship between the unemployment and crime rates when certain outlier values are disregarded. This implies that, aside from a few extreme cases, higher unemployment rates tend to coincide with higher crime rates, suggesting a predictive or causal relationship under normal circumstances.

To Query the Building Permits and Crime Rate

In this part, I will demonstrate the queries to explore the relationship between Building Permits and Crime Rate. The datasets/view about building permits I use below are created in my individual milestone. The datasets/view about crime rate are created by my teammates.

To Query the Amount of Bulding Permits and the Amount of Crime

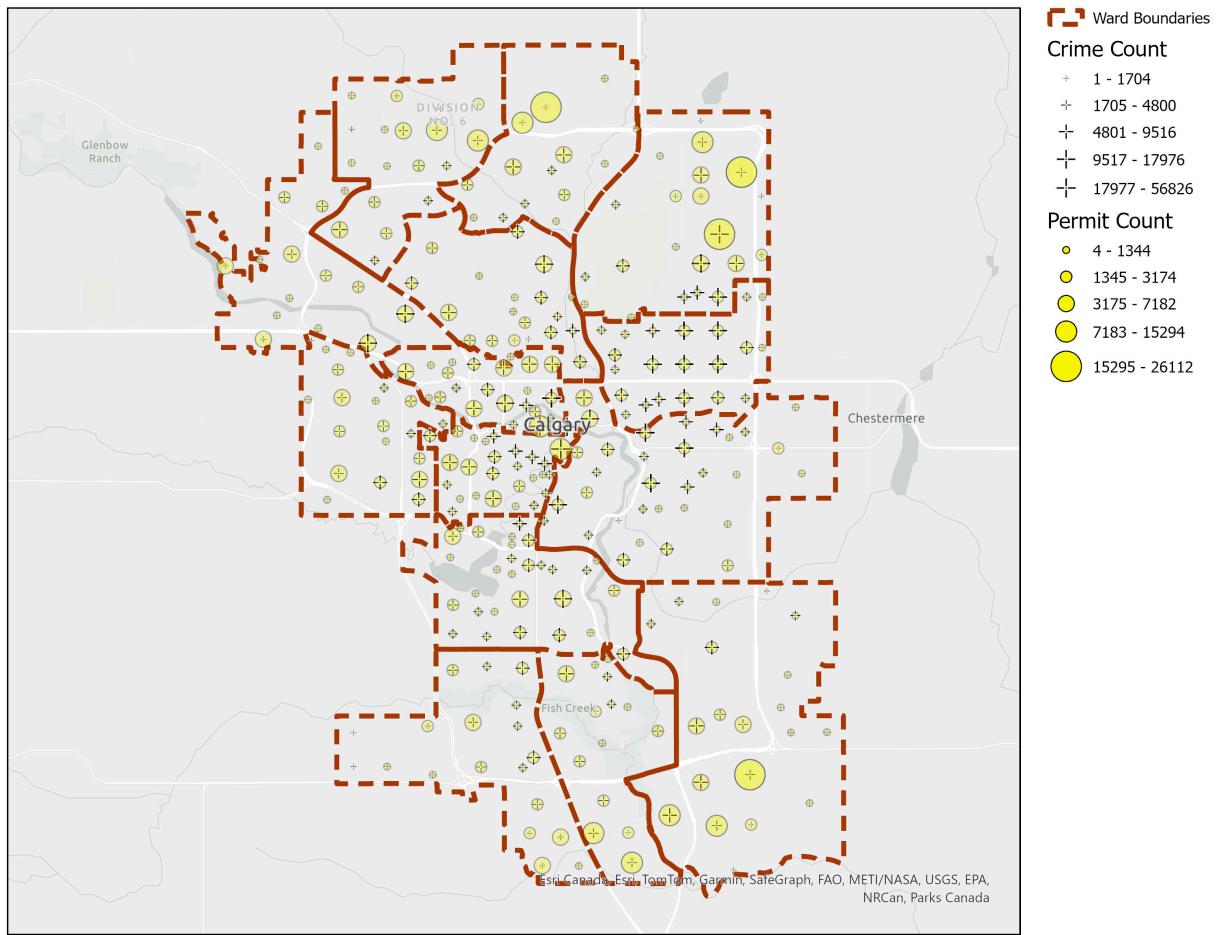
In [42]:

```
query2_1 = '''SELECT c.COMM_CODE,
    SUM(p.PermitCount) AS permitcount,
    SUM(c.Total_Crime_Count) AS crimecount,
    cc.LONGITUDE AS LONGITUDE,cc.LATITUDE AS LATITUDE
FROM vw_annual_permit_counts_by_community1 p
RIGHT JOIN vw_community_annual_crime c
ON p.COMM_CODE = c.COMM_CODE
LEFT JOIN communities cc ON c.COMM_CODE = cc.COMM_CODE
GROUP BY c.COMM_CODE;'''
query2_1_df = pd.read_sql_query(query2_1, engine)
display(query2_1_df)
# print("The amount of building permits and crime by community from 2018 to 2023")
# print(ta(query2_1_df, headers='keys', tablefmt='psql'))
```

| | COMM_CODE | permitcount | crimecount | LONGITUDE | LATITUDE |
|-----|-----------|-------------|------------|-------------|-----------|
| 0 | 01B | 75.0 | 125.0 | -114.242455 | 51.102838 |
| 1 | 01F | 80.0 | 66.0 | -114.263366 | 51.119619 |
| 2 | 01K | 60.0 | 365.0 | -114.222759 | 51.168724 |
| 3 | 02B | Nan | 3.0 | -114.199400 | 51.176032 |
| 4 | 02C | 96.0 | 57.0 | -114.176675 | 51.175902 |
| ... | ... | ... | ... | ... | ... |
| 288 | WND | 1230.0 | 6312.0 | -114.083549 | 51.005043 |
| 289 | WOO | 1128.0 | 3498.0 | -114.106169 | 50.942993 |
| 290 | WSP | 4848.0 | 3480.0 | -114.206168 | 51.059732 |
| 291 | WWO | 2532.0 | 105.0 | -114.008637 | 50.870724 |
| 292 | YKV | 2376.0 | 150.0 | -114.076648 | 50.870403 |

293 rows × 5 columns

The Crime counts and the amount of building permits across communities in 2023 and plotted on a map using ArcGIS Pro



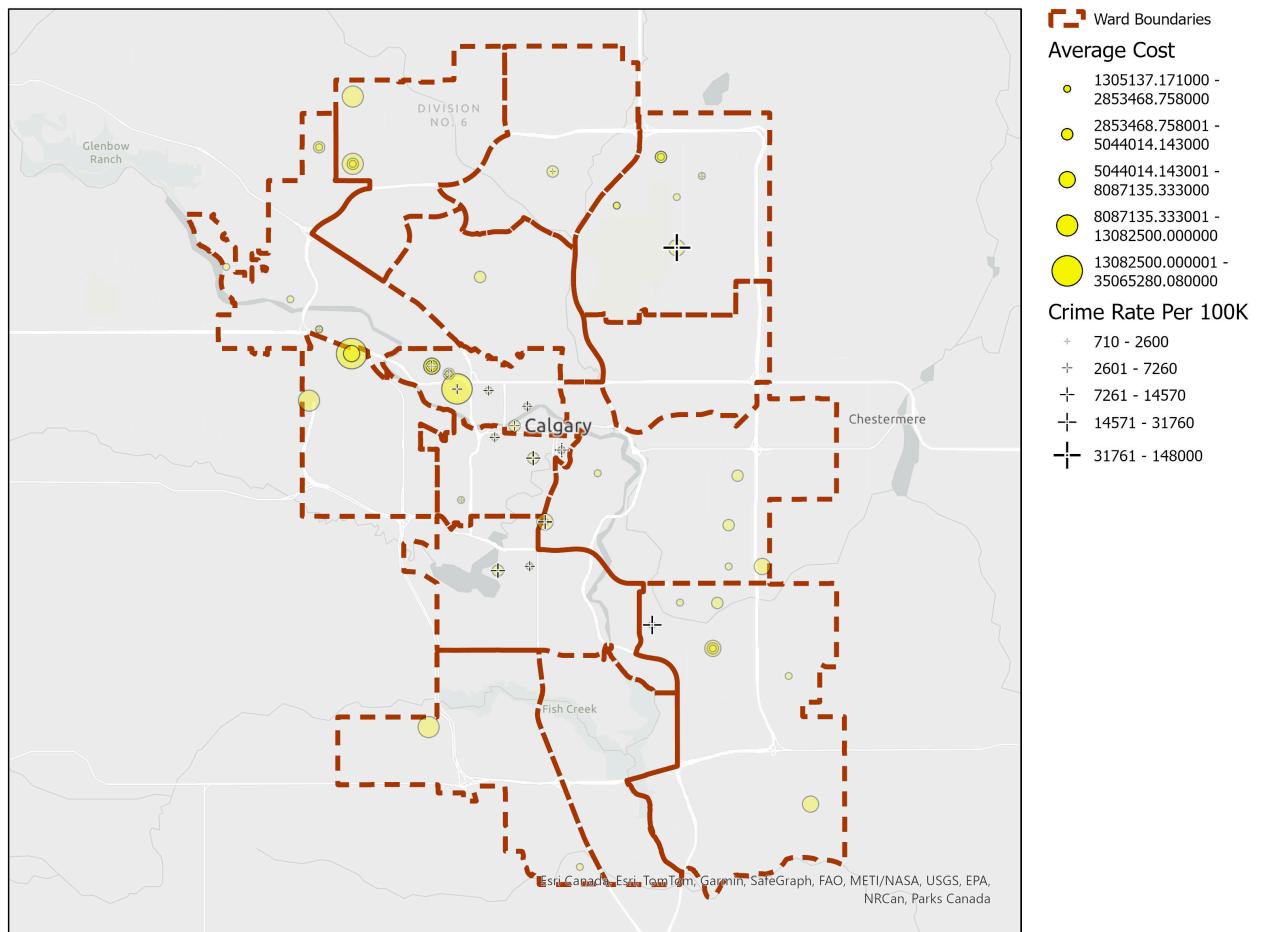
To Query the Top 10 Communities That Have the Highest Estimated Building Cost by Year and Corresponding Crime Rate

```
In [43]: query2_2 = '''SELECT p.Year, p.CommunityName, p.avgcost,
c.Crime_Rate_Per_100k, cc.LONGITUDE AS LONGITUDE,cc.LATITUDE AS LATITUDE
FROM vw_annual_top10_communities_avg_est_proj_cost P
LEFT JOIN communities cc ON p.CommunityName = cc.COMMUNITY_NAME
LEFT JOIN vw_community_annual_crime c ON cc.COMM_CODE = c.COMM_CODE AND p.Year = c.Year
ORDER BY p.Year ASC, c.Crime_Rate_Per_100k DESC;'''
query2_2_df = pd.read_sql_query(query2_2, engine)
display(query2_2_df)
```

| Year | CommunityName | avgcost | Crime_Rate_Per_100k | LONGITUDE | LATITUDE |
|------|-------------------------------|--------------|---------------------|-------------|-----------|
| 0 | SADDLE RIDGE INDUSTRIAL | 6.696995e+06 | 148000.0 | -113.975729 | 51.125190 |
| 1 | MANCHESTER | 5.578000e+06 | 10150.0 | -114.066650 | 51.006239 |
| 2 | UNIVERSITY DISTRICT | 6.970241e+06 | 3400.0 | -114.144946 | 51.073855 |
| 3 | SKYVIEW RANCH | 1.783155e+06 | 1800.0 | -113.958375 | 51.156126 |
| 4 | MEDICINE HILL | 2.797502e+07 | NaN | -114.200146 | 51.079185 |
| 5 | STONEY 2 | 1.395849e+06 | NaN | -114.017218 | 51.143387 |
| 6 | HASKAYNE | 1.750000e+06 | NaN | -114.286632 | 51.116943 |
| 7 | STONEGATE LANDING | 4.579302e+06 | NaN | -113.986682 | 51.164470 |
| 8 | 02E | 1.054850e+07 | NaN | -114.199437 | 51.161434 |
| 9 | STONEY 3 | 2.053620e+06 | NaN | -113.975839 | 51.147033 |
| 10 | BELTLINE | 1.761614e+06 | 8870.0 | -114.055183 | 51.037450 |
| 11 | ST. ANDREWS HEIGHTS | 3.506528e+07 | 4850.0 | -114.127435 | 51.063919 |
| 12 | UNIVERSITY HEIGHTS | 2.509568e+06 | 3460.0 | -114.133091 | 51.070406 |
| 13 | UNIVERSITY DISTRICT | 7.824080e+06 | 2410.0 | -114.144946 | 51.073855 |
| 14 | GREENWOOD/GREENBRIAR | 1.701182e+06 | 1000.0 | -114.222710 | 51.089861 |
| 15 | 12B | 2.079000e+06 | NaN | -113.898687 | 50.939231 |
| 16 | 12J | 7.500000e+06 | NaN | -113.883447 | 50.883580 |
| 17 | 01K | 2.502000e+06 | NaN | -114.222759 | 51.168724 |
| 18 | GREAT PLAINS | 1.305137e+06 | NaN | -113.939936 | 50.986877 |
| 19 | EAST SHEPARD INDUSTRIAL | 2.008910e+06 | NaN | -113.950957 | 50.951252 |
| 20 | SHEPARD INDUSTRIAL | 1.707914e+06 | 31760.0 | -113.992775 | 50.961480 |
| 21 | MANCHESTER | 1.962846e+06 | 12200.0 | -114.066650 | 51.006239 |
| 22 | CLIFF BUNGALOW | 3.339948e+06 | 9020.0 | -114.074901 | 51.033933 |
| 23 | HOUNSFIELD HEIGHTS/BRIAR HILL | 1.423883e+06 | 6000.0 | -114.105731 | 51.063185 |
| 24 | UNIVERSITY DISTRICT | 1.667514e+06 | 4680.0 | -114.144946 | 51.073855 |
| 25 | DOWNTOWN WEST END | 4.459239e+06 | 3590.0 | -114.087788 | 51.047966 |
| 26 | UNIVERSITY HEIGHTS | 3.095342e+06 | 2600.0 | -114.133091 | 51.070406 |
| 27 | NOSE HILL PARK | 3.609053e+06 | NaN | -114.111587 | 51.112489 |
| 28 | MEDICINE HILL | 5.498031e+06 | NaN | -114.200146 | 51.079185 |
| 29 | ALPINE PARK | 8.900000e+06 | NaN | -114.146987 | 50.917036 |
| 30 | SUNALTA | 2.754712e+06 | 7260.0 | -114.101474 | 51.042958 |
| 31 | BELTLINE | 2.471313e+06 | 5700.0 | -114.055183 | 51.037450 |
| 32 | KINGSLAND | 1.913290e+06 | 3730.0 | -114.077347 | 50.986977 |
| 33 | UNIVERSITY DISTRICT | 3.423446e+06 | 3260.0 | -114.144946 | 51.073855 |
| 34 | COUNTRY HILLS VILLAGE | 3.175047e+06 | 2310.0 | -114.061395 | 51.158204 |
| 35 | GREENWOOD/GREENBRIAR | 1.740846e+06 | 890.0 | -114.222710 | 51.089861 |
| 36 | ALYTH/BONNYBROOK | 1.964742e+06 | NaN | -114.030490 | 51.027363 |
| 37 | 02E | 4.300000e+06 | NaN | -114.199437 | 51.161434 |
| 38 | STONEY 2 | 1.470118e+06 | NaN | -114.017218 | 51.143387 |
| 39 | EAST SHEPARD INDUSTRIAL | 5.563515e+06 | NaN | -113.950957 | 50.951252 |
| 40 | SUNNYSIDE | 2.195444e+06 | 4370.0 | -114.079019 | 51.056420 |
| 41 | UNIVERSITY HEIGHTS | 2.853469e+06 | 1780.0 | -114.133091 | 51.070406 |
| 42 | CURRIE BARRACKS | 2.315759e+06 | 710.0 | -114.124595 | 51.015717 |
| 43 | EAST SHEPARD INDUSTRIAL | 3.014762e+06 | NaN | -113.950957 | 50.951252 |
| 44 | 02E | 1.937500e+06 | NaN | -114.199437 | 51.161434 |

| Year | CommunityName | avgcost | Crime_Rate_Per_100k | LONGITUDE | LATITUDE |
|------|---------------|---------------------|---------------------|-----------|-----------------------|
| 45 | 2022 | SECTION 23 | 3.562426e+06 | NaN | -113.947803 50.970942 |
| 46 | 2022 | GREAT PLAINS EAST | 6.957600e+06 | NaN | -113.916662 50.986955 |
| 47 | 2022 | SOUTH FOOTHILLS | 2.430702e+06 | NaN | -113.973625 50.971229 |
| 48 | 2022 | 06A | 1.308250e+07 | NaN | -114.229539 51.058873 |
| 49 | 2022 | STONEGATE LANDING | 5.044014e+06 | NaN | -113.986682 51.164470 |
| 50 | 2023 | EAGLE RIDGE | 4.051008e+06 | 14570.0 | -114.099300 50.985081 |
| 51 | 2023 | UNIVERSITY DISTRICT | 3.471663e+06 | 5110.0 | -114.144946 51.073855 |
| 52 | 2023 | STONEGATE LANDING | 2.036674e+06 | NaN | -113.986682 51.164470 |
| 53 | 2023 | 01B | 2.436552e+06 | NaN | -114.242455 51.102838 |
| 54 | 2023 | STARFIELD | 3.436105e+06 | NaN | -113.940036 51.004848 |
| 55 | 2023 | 02K | 1.000000e+07 | NaN | -114.199453 51.190553 |
| 56 | 2023 | 13M | 2.352000e+06 | NaN | -114.042754 50.856239 |
| 57 | 2023 | 01K | 4.775000e+06 | NaN | -114.222759 51.168724 |
| 58 | 2023 | MEDICINE HILL | 8.087135e+06 | NaN | -114.200146 51.079185 |
| 59 | 2023 | 09Q | 3.251684e+06 | NaN | -113.933951 51.026263 |

To visualize the average cost and crime rate across communities, we extracted the data from this query and plotted on a map using ArcGIS Pro



From query *To Query the Amount of Bulding Permits and the Amount of Crime*, we observe a positive correlation between building permits and crime rates in central zones, contrasting with adjacent areas where high crime occurs despite fewer permits, indicating a complex relationship. Notably, areas on the city's edge with many permits show low crime, aligning with expectations due to differing socio-economic conditions. Query *To Query the Top 10 Communities That Have the Highest Estimated Building Cost by Year and Corresponding Crime Rate* reveals that in communities with high building costs, there is a correlation with higher crime rates in central areas. In contrast, peripheral areas display mixed patterns, with some high-crime areas despite lower construction costs. A significant limitation is the presence of many null values in crime data, possibly because high-cost areas are often non-residential, affecting crime rate reporting.

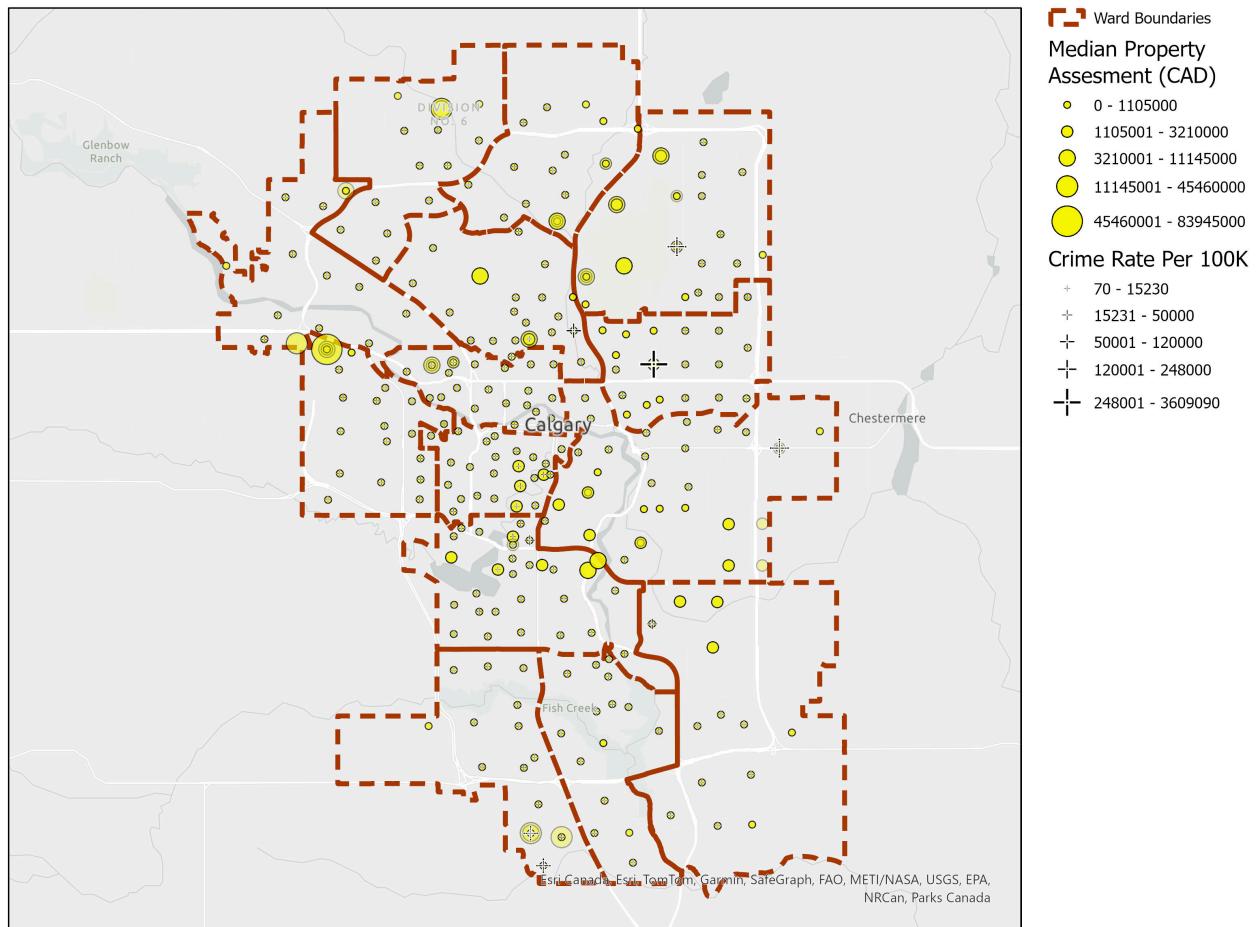
To Query the Assessment Values and Crime rate

In this section, I will query the assessment values and crime rate at the community level and try to get insights into the categories of crime occurring in those communities with a high assessment values.

```
In [44]: query3_1 = '''SELECT p.ROLL_YEAR,p.COMM_CODE, p.median, c.Crime_Rate_Per_100k,
cc.LONGITUDE AS LONGITUDE,cc.LATITUDE AS LATITUDE
FROM assessment_statistics p LEFT JOIN vw_community_annual_crime c
ON p.COMM_CODE = c.COMM_CODE AND p.ROLL_YEAR = c.Year
LEFT JOIN communities cc ON p.COMM_CODE = cc.COMM_CODE
GROUP BY p.ROLL_YEAR, p.COMM_CODE
ORDER BY p.ROLL_YEAR;'''
query3_1_df = pd.read_sql_query(query3_1, engine)
display(query3_1_df)
# print("The median assessments value and crime rate by community")
# print(ta(query3_1_df, headers='keys', tablefmt='psql'))
```

| | ROLL_YEAR | COMM_CODE | median | Crime_Rate_Per_100k | LONGITUDE | LATITUDE |
|------|-----------|-----------|-----------|---------------------|-----------|-------------|
| 0 | 2018 | 01B | 349250.0 | | NaN | -114.242455 |
| 1 | 2018 | 01C | 393500.0 | | NaN | -114.237134 |
| 2 | 2018 | 01D | 40750.0 | | NaN | NaN |
| 3 | 2018 | 01F | 1010000.0 | | NaN | -114.263366 |
| 4 | 2018 | 01G | 433750.0 | | NaN | NaN |
| ... | ... | ... | ... | ... | ... | ... |
| 1865 | 2023 | WND | 185000.0 | | 2620.0 | -114.083549 |
| 1866 | 2023 | WOO | 504250.0 | | 1350.0 | -114.106169 |
| 1867 | 2023 | WSP | 675500.0 | | 640.0 | -114.206168 |
| 1868 | 2023 | WWO | 488500.0 | | NaN | -114.008637 |
| 1869 | 2023 | YKV | 456000.0 | | 50000.0 | -114.076648 |

1870 rows × 6 columns



To Query the Type of Crime in the Most Expensive Communities

```
In [45]: query3_2 = '''WITH Rankdedcomm AS (SELECT ROLL_YEAR, COMM_CODE, `mean`,
ROW_NUMBER() OVER (PARTITION BY ROLL_YEAR ORDER BY `mean` DESC) AS Rankedmean
```

```

FROM assessment_statistics,
Rankedtype AS (SELECT YEAR(CRIME_DATE) AS `Year`, COMM_CODE, CATEGORY, CRIME_COUNT,
ROW_NUMBER() OVER(PARTITION BY YEAR(CRIME_DATE), COMM_CODE ORDER BY CRIME_COUNT DESC) AS Rankedc
FROM crime_statistics)
SELECT r.ROLL_YEAR, r.COMM_CODE, r.`mean`, c.CRIME_COUNT,c.CATEGORY
FROM Rankdedcomm r LEFT JOIN
Rankedtype c ON
r.ROLL_YEAR = c.Year AND r.COMM_CODE = c.COMM_CODE
WHERE r.Rankedmean <=10 AND c.Rankedcrime =1
ORDER BY r.ROLL_YEAR, r.`mean` DESC;'''

query3_2_df = pd.read_sql_query(query3_2, engine)
print("The median assessments value and crime rate by community")
print(ta(query3_2_df, headers='keys', tablefmt='psql'))

```

The median assessments value and crime rate by community

| | ROLL_YEAR | COMM_CODE | mean | CRIME_COUNT | CATEGORY |
|----|-----------|-----------|-------------|-------------|-------------------------------|
| 0 | 2018 | UOC | 5.26448e+07 | 2 | Theft OF Vehicle |
| 1 | 2018 | NPK | 2.94978e+07 | 5 | Theft FROM Vehicle |
| 2 | 2018 | UND | 2.40275e+07 | 3 | Violence Other (Non-domestic) |
| 3 | 2018 | 02E | 2.35722e+07 | 12 | Assault (Non-domestic) |
| 4 | 2018 | 01K | 1.14293e+07 | 2 | Theft FROM Vehicle |
| 5 | 2018 | EFV | 1.06662e+07 | 5 | Theft FROM Vehicle |
| 6 | 2019 | UOC | 4.85727e+07 | 3 | Theft OF Vehicle |
| 7 | 2019 | NPK | 3.82784e+07 | 4 | Theft FROM Vehicle |
| 8 | 2019 | 02E | 2.39709e+07 | 11 | Assault (Non-domestic) |
| 9 | 2019 | UND | 1.66584e+07 | 2 | Break & Enter - Commercial |
| 10 | 2019 | 01K | 1.4148e+07 | 4 | Theft OF Vehicle |
| 11 | 2019 | 09Q | 1.11632e+07 | 2 | Theft FROM Vehicle |
| 12 | 2019 | EFV | 9.87538e+06 | 7 | Theft FROM Vehicle |
| 13 | 2019 | QPK | 9.80792e+06 | 1 | Theft FROM Vehicle |
| 14 | 2020 | NPK | 3.82844e+07 | 3 | Theft FROM Vehicle |
| 15 | 2020 | 02E | 2.49184e+07 | 8 | Assault (Non-domestic) |
| 16 | 2020 | HSN | 1.85375e+07 | 1 | Break & Enter - Commercial |
| 17 | 2020 | UND | 1.80816e+07 | 5 | Violence Other (Non-domestic) |
| 18 | 2020 | UOC | 1.1723e+07 | 1 | Theft FROM Vehicle |
| 19 | 2020 | ABP | 1.01661e+07 | 2 | Theft FROM Vehicle |
| 20 | 2020 | 01K | 1.01384e+07 | 1 | Theft FROM Vehicle |
| 21 | 2020 | ST2 | 9.61357e+06 | 12 | Theft FROM Vehicle |
| 22 | 2020 | EFV | 9.07065e+06 | 8 | Theft FROM Vehicle |
| 23 | 2020 | CIA | 8.41836e+06 | 10 | Theft FROM Vehicle |
| 24 | 2021 | UOC | 4.79065e+07 | 2 | Assault (Non-domestic) |
| 25 | 2021 | NPK | 3.44449e+07 | 3 | Theft FROM Vehicle |
| 26 | 2021 | 02E | 2.72909e+07 | 8 | Assault (Non-domestic) |
| 27 | 2021 | SVR | 1.982e+07 | 1 | Theft OF Vehicle |
| 28 | 2021 | HSN | 1.89587e+07 | 1 | Break & Enter - Commercial |
| 29 | 2021 | 01K | 1.09765e+07 | 1 | Break & Enter - Commercial |
| 30 | 2021 | ABP | 9.74143e+06 | 2 | Commercial Robbery |
| 31 | 2021 | QPK | 9.70917e+06 | 2 | Theft FROM Vehicle |
| 32 | 2021 | ST2 | 9.43942e+06 | 10 | Theft FROM Vehicle |
| 33 | 2021 | EFV | 9.0696e+06 | 6 | Theft FROM Vehicle |
| 34 | 2022 | COP | 7.34e+07 | 1 | Assault (Non-domestic) |
| 35 | 2022 | UOC | 4.82149e+07 | 4 | Theft FROM Vehicle |
| 36 | 2022 | NPK | 3.47217e+07 | 7 | Theft FROM Vehicle |
| 37 | 2022 | 02E | 3.15793e+07 | 11 | Assault (Non-domestic) |
| 38 | 2022 | SVR | 1.991e+07 | 3 | Theft FROM Vehicle |
| 39 | 2022 | 01K | 1.17934e+07 | 2 | Theft FROM Vehicle |
| 40 | 2022 | QPK | 1.05008e+07 | 1 | Theft OF Vehicle |
| 41 | 2022 | EFV | 1.01536e+07 | 12 | Theft FROM Vehicle |
| 42 | 2022 | ST2 | 9.58797e+06 | 15 | Theft FROM Vehicle |
| 43 | 2023 | COP | 8.3945e+07 | 1 | Assault (Non-domestic) |
| 44 | 2023 | NPK | 3.53492e+07 | 3 | Violence Other (Non-domestic) |
| 45 | 2023 | 02E | 3.18677e+07 | 8 | Assault (Non-domestic) |
| 46 | 2023 | SVR | 2.118e+07 | 2 | Theft FROM Vehicle |
| 47 | 2023 | 02F | 1.50654e+07 | 1 | Theft FROM Vehicle |
| 48 | 2023 | 01K | 1.29101e+07 | 2 | Theft FROM Vehicle |
| 49 | 2023 | UOC | 1.1403e+07 | 5 | Theft FROM Vehicle |
| 50 | 2023 | QPK | 1.10442e+07 | 2 | Theft FROM Vehicle |
| 51 | 2023 | EFV | 1.05849e+07 | 5 | Theft FROM Vehicle |

In [46]:

```

query3_2_df.fillna(0, inplace=True)
def plot_table(ROLL_YEAR):
    filtered_data = query3_2_df[query3_2_df['ROLL_YEAR'] == ROLL_YEAR]
    header_values = ["ROLL_YEAR", "COMM_NAME", "mean", "CRIME_COUNT", "CATEGORY"]
    cell_values = [
        filtered_data["ROLL_YEAR"],
        filtered_data["COMM_CODE"],

```

```

        filtered_data["mean"],
        filtered_data["CRIME_COUNT"],
        filtered_data["CATEGORY"]
    ]
    fig = go.Figure(data=[go.Table(
        header=dict(values=header_values, fill_color='paleturquoise', align='left'),
        cells=dict(values=cell_values, fill_color='lavender', align='left'))
])
    fig.show()
year_dropdown = Dropdown(options=query3_2_df['ROLL_YEAR'].unique(), description='Year: ')
interact(plot_table, ROLL_YEAR=year_dropdown)

interactive(children=(Dropdown(description='Year:', options=(2018, 2019, 2020, 2021, 2022, 2023), value=2018),...
<function __main__.plot_table(ROLL_YEAR)>

```

Out[46]:

The query *To Query the Assessment Values and Crime rate* analysis indicates a general trend where higher property assessment values correlate with lower crime rates, suggesting that crime is more prevalent in communities with lower to moderate property values. However, there are exceptions; some communities with moderate property values experience disproportionately high crime rates. It's also noted that instances of null crime rates in areas with high property values often correspond to industrial or commercial communities, not residential. Despite this, it's important to acknowledge that high-value residential communities are not immune to crime.

From query *To Query the Type of Crime in the Most Expensive Communities*, an examination of crime types within the top 10 highest-valued communities reveals that "Theft from vehicle," "Assault," and "Other violence" are among the most common offenses. This highlights the specific nature of crimes affecting even the most affluent areas.

Discussion

Learnings

Jianling

"I am a new learner to Structured Query Language (SQL). This project provided me with the opportunity to research a topic outside my expertise, and to practice my SQL skills in a real-world dataset. Upon completion of this project, I have gained valuable insights into the SQL features. I am now not only able to execute basic queries, but also understand more complex elements such as joining tables in multiple ways (left join, inner join), subqueries, and understand the concept of normalization process to eliminate data redundancy and enhance data integrity in the table. Our project explored the pattern of Calgary community crime statistics and association with spatial and economic factors. During our rigours queries, we were able to refine our study objectives and extended our project to another course, DATA 603. We cleaned, wrangled and generated a master dataset from SQL containing six years data of annual community crime statistics, spatial and economic factors. With these data we have built a regression model to quantify the associations between community crime rate and the spatial and economic factors, and made predictions for the crime rates in the communities that will have the green LRT stations in 2030. It has been a great learning experience to seamlessly use different analytic tools to accomplish our research work. I believe that incorporating SQL into my daily work can not only streamline my workflow but also significantly improve my efficiency. Finally, effective team collaboration has been the key to accomplish the group projects in such a short period of time. It has been my honor to work with, Alan (our team's SQL powerhouse), and Zane."

Alan

"I'm experienced with SQL and relational databases, however spatial data was unfamiliar territory. Now, I've improved my skills with GEOMETRY data types significantly. It involved a learning curve, especially with importing location data into SQL tables and transforming text into WKT (Well-Known Text) and then into GEOMETRY. I've also come to understand spatial

standards, like SRID 4326 corresponding to the WGS84 coordinate system which essentially means the Earth's longitude and latitude coordinates. The WITH common table expression, which I had seldom encountered in my career, has now become a function I certainly will use going-forward in crafting complex queries—I can appreciate its advantages in readability and debugging queries."

Zane

"Through this project, I gained invaluable knowledge and skills. Firstly, I mastered downloading data from open data websites and importing it into SQL, utilizing various Python libraries to handle large datasets—an essential skill for data analysis. Secondly, configuring a database became clearer to me as I navigated through challenges associated with importing sizable datasets into SQL, enhancing my understanding of database setup and optimization. Thirdly, I learned to create ER diagrams for database schemas and to apply data normalization techniques, which are crucial for efficient database design. Lastly, my proficiency in SQL advanced significantly; I became adept at using complex query functions like "CASE" and "CIE", and learned to create views and functions, further broadening my data manipulation capabilities. I've developed a keen interest in NoSQL databases. Although our project didn't incorporate NoSQL tools, the fact that our data was available in JSON format opens up intriguing possibilities for exploration with tools like MongoDB or Neo4j for querying. I would also like to express my gratitude to my teammates. Their assistance and support were invaluable, and it was an honor to collaborate with such dedicated individuals."

Team

It took some time for us to understand this, as one of us was wondering why our Crime Rate (Crime Count (integer at scale 0) / Resident Count (integer at scale 0)) * 100,000 had such neat numbers, usually with four significant digits. At first, we chalked it up to the INTEGER data types, thinking we were losing decimal places along the way. However another member knew the principles of database design and storage, if you have a Count, it should be stored as an INTEGER. A bit of research and a conversation with ChatGPT led us to the "real culprit": our database had a "div_precision_increment" setting fixed at 4. This meant that dividing two scale 0 numbers defaulted to a scale 4 result thus when we multiplied by 100,000 the numbers were not what we were expecting. To rectify this we can perform a CAST to a data type at a higher scale or we can set the variable to a different value which only lasts for the connection session.

```
In [47]: query_table = pd.read_sql_query("""
SHOW VARIABLES LIKE 'div_precision_increment';
""", engine)
print("Database Setting - DIV_PRECISION_INCREMENT")
print(ta(query_table, headers='keys', tablefmt='psql'))
```

| Variable_name | Value |
|-------------------------|-------|
| div_precision_increment | 4 |

```
In [48]: query_table = pd.read_sql_query("""
SELECT YEAR, community_name, (Total_Crime_Count/Resident_Count)*100000 AS 'No CAST',
CAST(Total_Crime_Count AS DECIMAL(20,10)) / CAST(Resident_Count AS DECIMAL(20,10))*100000 AS 'Wi-
FROM `vw_annual_top10_communities_by_crime_count`
GROUP BY YEAR, community_name
LIMIT 5
""", engine)

print("The Difference Between Cast to a data type with increased Scale and No Cast at default sc
query_table
```

The Difference Between Cast to a data type with increased Scale and No Cast at default scale

Out[48]:

| | Year | community_name | No CAST | With CAST |
|---|------|--------------------------|---------|--------------|
| 0 | 2018 | BELTLINE | 6750.0 | 6753.153727 |
| 1 | 2018 | DOWNTOWN COMMERCIAL CORE | 13220.0 | 13221.236900 |
| 2 | 2018 | FOREST LAWN | 6760.0 | 6757.102636 |
| 3 | 2018 | BOWNESS | 4430.0 | 4430.493274 |
| 4 | 2018 | MARLBOROUGH | 4690.0 | 4693.298406 |

Conclusion

Our investigation into Calgary's crime statistics from 2018 to 2023 has yielded significant insights, enabled by our use of SQL and analytical visualizations. By employing SQL views, we streamlined the querying process, allowing for more efficient team analysis of complex data sets. This method proved essential in managing and interpreting the extensive crime data collected in tandem, which included using GEOMETRY data types to analyze spatial patterns effectively. Adjustments to the database's div_precision_increment setting enhanced the accuracy of our crime rate computations, ensuring more reliable data for our analyses.

In the discussions throughout our report, we contextualized the crime trends within Calgary's broader socio-economic landscape, including the impact of proximity to LRT stations on crime rates. This aspect was particularly insightful, as it highlighted potential risk factors and areas requiring focused security enhancements.

Further analyses revealed a general trend where higher property assessment values correlate with lower crime rates, with exceptions in some moderate-value communities that experience disproportionately high crime rates. Additionally, crime patterns in high-value residential and commercial communities underscore that no area is immune to crime, with "Theft from vehicle," "Assault," and "Other violence" being prevalent even in the wealthiest sectors.

The exploration of the relationship between building permits and crime rates across different zones highlighted a complex interplay where increased construction activity in central areas correlates with higher crime rates, contrary to the lower crime observed in peripheral high-permit zones. These findings suggest fluctuations in crime incidence related to urban development and socio-economic conditions.

Moreover, our analysis of unemployment trends alongside crime rates provided compelling evidence of a strong correlation, particularly during certain periods where these rates moved in tandem. This relationship suggests a potential predictive or causal linkage between higher unemployment and increased crime under typical conditions.

This project not only highlights the dynamics of urban crime but also underscores the importance of sophisticated data management. Improvements would be to improve the data quality of the Building Permits dataset, potential data errors and to increase the accuracy of the crime location to a perhaps the central point of postal code or forward sortation area. measures..

Additional References

- Police-reported crime statistics in Canada, 2022. "Statistics Canada." Retrieved from <https://www150.statcan.gc.ca/n1/daily-quotidien/230727/dq230727b-eng.htm?HPA=1>. Date accessed: 2024, March 3.
- Crime severity index and weighted clearance rates, Canada, provinces, territories and Census Metropolitan Areas. "Statistics Canada." Retrieved from <https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=3510002601>. Date accessed: 2024, March 3.
- The Corporate Research Team, Customer Service and Communications, The City of Calgary. 2023 Perspectives on Calgary Safety Perceptions, Final Report. June 2023.
- Wilkinson, D. L., and Fagan, J. A theory of violent events. In R. F. Meier, L. W. Kennedy, & V. F. Sacco (Eds.), *The process and structure of crime: Criminal events and crime analysis* (pp. 169–195). New York: Transaction Publishers; 2001.

- The Chicago School And Cultural/Subcultural Theories of Crime. Retrieved from https://www.sagepub.com/sites/default/files/upm-binaries/29411_6.pdf. Date accessed: 2024, March 3.2009.
- Calculating distance between two points in SQL using the Haversine formula. Retrieved from <https://www.aiops.com/blog/calculating-distance-between-two-points-in-sql#:~:text=The%20Haversine%20formula%20allows%20you,using%20latitudinal%20and%20longitudinal%20coordinates%20to%20calculate%20the%20distance%20between%20two%20points%20in%20SQL.>
- Haversine calculation in user-defined function. "SQL Server Central." Retrieved from <https://www.sqlservercentral.com/scripts/haversine-calculation-in-user-defined-function>.

SQL Views Used to Support Team Members

SQL views served as an abstraction layer for our team, providing a consistent interface for members to access and utilize data. This approach allowed for a reliable structure of expected columns, while enabling individual team members to refine the underlying queries without necessitating changes to subsequent queries built upon that foundation by others. Not all SQL views made it to the final report for answering our Guiding Questions or showcased in the Exploratory Data Analysis section.

```
In [49]: query_table = pd.read_sql_query("""
    SELECT TABLE_NAME AS VIEW_NAME, VIEW_DEFINITION FROM
        INFORMATION_SCHEMA.VIEWS AS v
    """
    , engine)

print("SQL Views")

query_table
```

SQL Views

Out[49]:

| | VIEW_NAME | VIEW_DEFINITION |
|----|---|--|
| 0 | average_value | select `I01-4`.`assessments_by_community`.`ROL... |
| 1 | vw_annual_avg_est_cost_by_community | select year(`bp`.`ISSUED_DATE`) AS `year` ,`c`.... |
| 2 | vw_annual_city_crime_rate | with CityCrimeSum as (select year(`I01-4`.`cri... |
| 3 | vw_annual_communities_crime_statistics | with CommunityCrimeCounts as (select `cs`.`COM... |
| 4 | vw_annual_monthly_permit_counts | select year(`I01-4`.`building_permits`.`ISSUED... |
| 5 | vw_annual_permit_counts_by_community | select year(`bp`.`ISSUED_DATE`) AS `Year(Issue... |
| 6 | vw_annual_permit_counts_by_community1 | select `p`.`Year(Issued_Date)` AS `Year(Issued... |
| 7 | vw_community_lrt_distances | with lrt_distances as (select `comm`.`COMM_COD... |
| 8 | vw_known_unknown_areas | select count(distinct case when `cs`.`SECTOR` ... |
| 9 | vw_annual_community_census_demographics | with crime_years as (select distinct `I01-4`.`... |
| 10 | vw_annual_community_crime_type_specific | with CrimeByCommunity as (select `I01-4`.`crim... |
| 11 | vw_annual_construction_times_by_community | select year(`bp`.`ISSUED_DATE`) AS `Year` ,`c`.... |
| 12 | vw_number_future_lrt_by_community | select `community`.`COMM_CODE` AS `Community C... |
| 13 | vw_number_of_years_crime_statistics | select count(distinct year(`I01-4`.`crime_stat... |
| 14 | vw_annual_crime_by_type | with CrimeByYear as (select year(`I01-4`.`crim... |
| 15 | vw_annual_permit_counts_by_community_by_type | with PermitCounts as (select year(`I01-4`.`bui... |
| 16 | vw_annual_top10_communities_avg_est_proj_cost | with RankedCommunity as (select year(`bp`.`ISS... |
| 17 | vw_annual_top10_communities_by_crime_count | with CommunityCrimeCounts as (select `I01-4`.`... |
| 18 | vw_annual_uoc_crime_corrected | with CrimeData as (select year(`I01-4`.`crime_... |
| 19 | vw_city_annual_crime | select `r`.`Year` AS `Year` ,`r`.`Total_Residen... |
| 20 | vw_city_monthly_crime_count_rate | with CrimeCounts as (select year(`I01-4`.`crim... |
| 21 | vw_community_annual_crime | select `r`.`YEAR` AS `Year` ,`r`.`COMM_CODE` AS... |
| 22 | vw_community_annual_max_min_crime_count | with CommunityCrimeSum as (select year(`I01-4`... |
| 23 | vw_community_annual_max_min_crime_rate | with CommunityCrimeRates as (select `r`.`YEAR`... |
| 24 | vw_community_distances | with lrt_shortest_distances as (select `comm`.... |
| 25 | vw_community_distances_all | with lrt_shortest_distances as (select `comm`.... |
| 26 | vw_crime_stats_previous_year | with CRIME_STATS_PREVIOUS_YEAR as (select `I01... |
| 27 | vw_data_quality_issue_communities | select 'Issue-Missing' AS `ISSUE` ,`Community C... |
| 28 | vw_monthly_crime_count_by_year | with CrimeCounts as (select year(`I01-4`.`crim... |
| 29 | vw_sector_annual_crime | select `r`.`YEAR` AS `Year` ,`r`.`Sector` AS `S... |

In [50]:

```
# to disconnect from the database
engine.dispose()
```

In [51]:

```
#THANK YOU
```