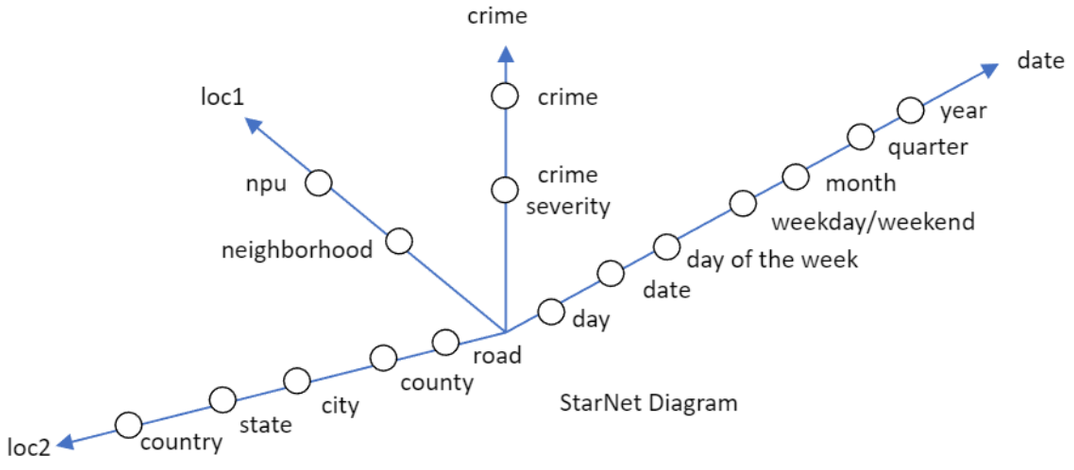
The purpose of this report is to present the design and implementation of a data warehouse from a dataset containing information about different types of crimes having occurred in different locations. To effectively design a data warehouse, it involves identifying the process being modeled, determining the grain at which facts can be stored, selecting the appropriate dimensions, and identifying the relevant numeric measures for the facts. Using Kimball's four steps to dimensional modeling, we were able to design a data warehouse that incorporates four dimensions:

- crime, which includes the type of crimes committed, and it's severity from 0-11, 0 being the worst e.g. homicide.
- date, which includes the year, quarter, month, whether it was a weekday/weekend, day of the week, full date and numerical day.
- loc1, which is short for location1, and loc2 which is short for location2. the location dimension was split into two,
  loc1 being based on the government's classification of locations. it consists of the npu (neighborhood planning units) and the neighborhoods.
  loc2 on the other hand is based on the general way to classify locations and consists of the country, state, city, county and road of where the crime was committed.

To demonstrate the query capabilities of the data warehouse, five business questions were formed that could be answered using the data. These questions are:

1. Is there a seasonal pattern to certain crime occurrences? How or when, and what crimes?
2. Do crimes happen in certain areas more often depending on the time of month?
3. Which neighborhoods are highest in each crime and are overall most dangerous? Which roads should be avoided?
4. Is there a difference in crime severity between weekdays and weekends?
5. Do the crimes rates change overtime in heavy crime locations?

To help identify the dimensions and concept hierarchies for each dimension, a StarNet diagram was created. This diagram showcases the various hierarchies and levels within each dimension, providing a clear overview of the data model and its structure.



StarNet Diagram

To the left is the StarNet diagram. By using the footprints, we can identify how the business queries can be answered.

Questions 1 uses the crime and date dimensions, specifically with year quarter.

Question 2 also uses the crime and date dimensions, day

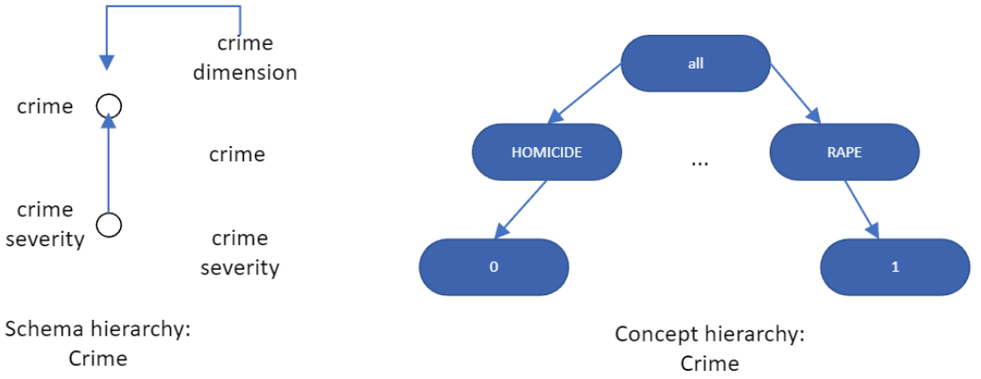
Question 3 makes use of the crime, loc1 and loc2 dimensions. We will be checking road in conjunction to neighborhood.

Question 4 makes use of crime severity from the crime dimension as well as weekday/weekend from the date dimension.

Question 5 makes use of crime, date and loc1. We will be referring to q3 for the neighborhood findings.

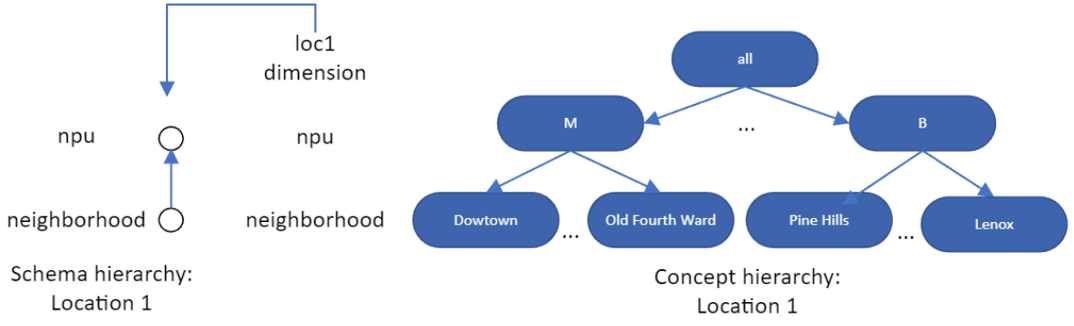
To better understand the structure and relationships within our data warehouse, we will now present the schema for each dimension, along with their corresponding concept hierarchies; a concept hierarchy is a hierarchical arrangement of concepts or categories that allows for drill-down analysis and aggregation of data. By examining these hierarchies, we can gain some insight on how the data is organized and how it can be queried at different levels of granularity.

Diagram 1: Crime schema hierarchy and concept hierarchy



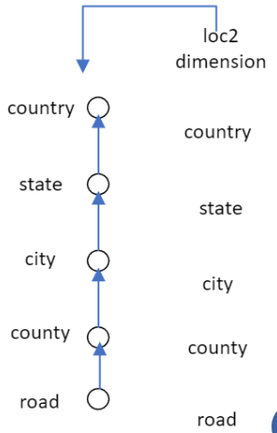
Let's talk about the crime schema hierarchy first. It only contains 2 footprints because there are only 2 columns in the dimension table, and in this case, since all the crimes are in each own's category, we cannot put crime severity over crime since all the crimes are of different footing, meaning that there could only be one crime severity for each crime. In the concept hierarchy, we can see that there are many crimes, but only of one crime severity each.

Diagram 2: Loc1 (location1) schema hierarchy and concept hierarchy

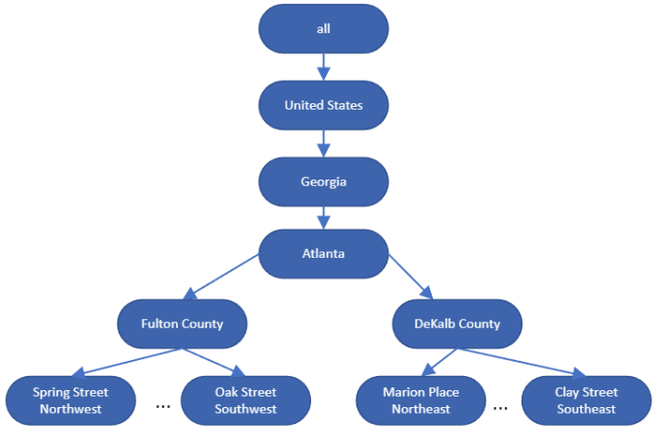


In the loc1 schema hierarchy, it also only contains 2 footprints. Upon inspection, it was revealed that multiple neighborhoods can be in one npu, therefore making npu a more 'general' term than neighborhood. As can be seen in the concept hierarchy, there are multiple arrows pointing from one npu to different neighborhoods.

Diagram 3: Loc2 (location 2) schema and concept hierarchy



Schema Hierarchy:
Location 2

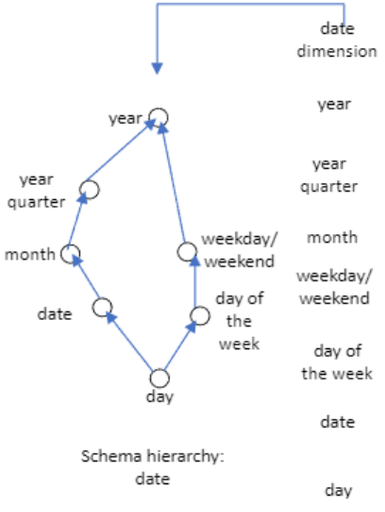


Concept Hierarchy:
Location 2

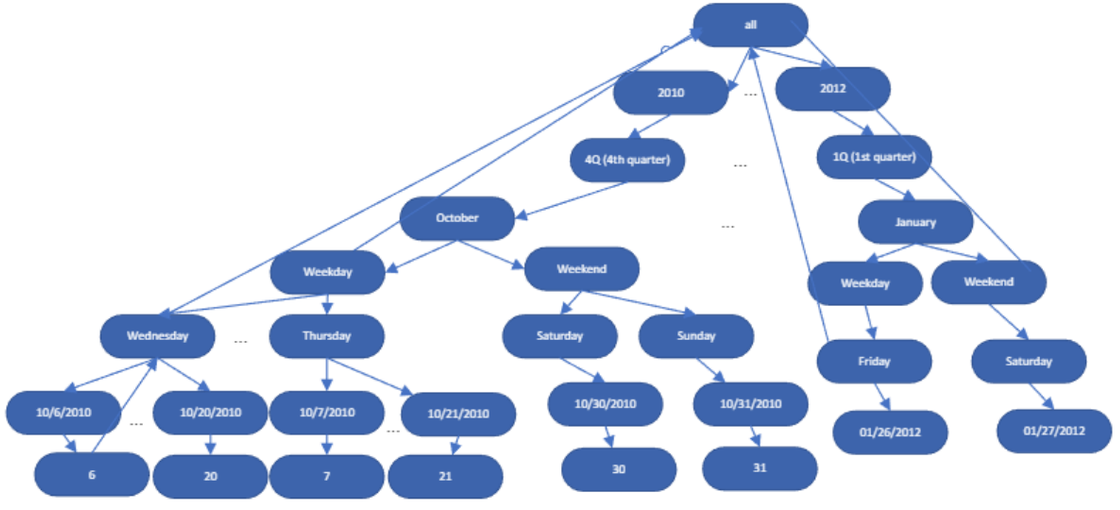
To the left is the schema and concept hierarchy for loc2 (location2). In the schema hierarchy, it goes from country > state > city > county > road. Since we don't have counties in Australia, I wasn't entirely sure what they were and asked ChatGPT if county was larger than city which they said yes. That being said, upon closer inspection, the city Atlanta had multiple counties, so I put city above county since it was only one single value that I couldn't have broken down properly had county been above city in the concept hierarchy. Even though the country, state, and city are all pointing in one straight line in the concept hierarchy, this is because we only have those values in the dataset. In a bigger more wide dataset, we would still implement this hierarchy since countries are bigger that states, and states are bigger than cities.

Diagram 4: Date schema and concept hierarchy

To the right is the schema hierarchy and concept hierarchy for date. This is so far the biggest dimension as it was broken down to multiple footprints, them being year, year quarter, month, weekday/weekend, day of the week, day, date. The reason why there is a partial order in the schema hierarchy, is because weekdays and weekends, as well as days of a week don't exactly add up to a year. There is still a total order on the left hand side of the schema hierarchy, since year will fall into a year quarter, then to months, etc.



Schema hierarchy:
date



Concept hierarchy:
date

Having understood the granularity of the data warehouse queries, we can move on to the ETL process. ETL stands for Extract, Transform, Load and it is a data integration process that involves extracting data from various sources, transforming the data to fit a particular structure or format, and then loading the transformed data into a target database or data warehouse. In this project, I made use of Python pandas and Excel for the data cleaning process, SQL Server Management Studio 2019 to populate the database and create our Star Schema, and Microsoft Visual Studio 2019 to create a multidimensional analysis project, create a cube and extract, load, and deploy it

Step 1: we modify crime.csv by removing the unenriched rows and concatenating the other files to make up 22500 rows of data.

```python
def allcsv():
    #removes unenriched rows from og crime file, then combines the rest of the file
    df = pd.read_csv('crime.csv', header=None)
    #rows up to 25470
    index_to_remove = df[df[0] == 25470].index[0]
    df = df.drop(df.index[index_to_remove+1:])
    df.to_csv('crime.csv', index=False, header=None)

    df1 = pd.read_csv('crime.csv')
    df2 = pd.read_csv('crime_25471_50000.csv')
    df3 = pd.read_csv('crime_50001_75000.csv')
    df4 = pd.read_csv('crime_75001_100000.csv')
    df5 = pd.read_csv('crime_100001_125000.csv')
    df6 = pd.read_csv('crime_125001_150000.csv')
    df7 = pd.read_csv('crime_150001_175000.csv')
    df8 = pd.read_csv('crime_175001_200000.csv')
    df9 = pd.read_csv('crime_200001_225000.csv')

    combined_df = pd.concat([df1, df2, df3, df4, df5, df6, df7, df8, df9], axis=0)

    #drop the last two columns, bc it generated some extra stuff
    combined_df = combined_df.iloc[:, :-1]
    #renamed header for convenience
    combined_df = combined_df.rename(columns={'Unnamed: 0': 'id'})

    combined_df.to_csv('crime.csv', index=False)
```

Step 2: any empty neighborhood values were replaced by neighbourhood_lookup values to preserve as much data as we can.

```python
def replace_neighborhood():
    df = pd.read_csv("crime.csv", dtype=str)

    #replace empty neighborhood values with values from neighborhood_lookup
    for i in range(len(df)):
        if pd.isna(df.at[i, 'neighborhood']):
            df.at[i, 'neighborhood'] = df.at[i, 'neighbourhood_lookup']

    df.to_csv("crime.csv", index=False)
```

Step 3: we filter out only the columns we want and keep them. this is so that, instead of just dropping any row in the full csv file, we are able to only check for values that matter to us.

```python
def removecols():
    df = pd.read_csv("crime.csv")

    #fill missing values in 'country' column with 'United States'
    df['country'].fillna('United States', inplace=True)

    #drop rows with empty cells in said columns
    columns_to_check = ['crime','date','neighborhood','npu','road','county','city','state','country']
    df.dropna(subset=columns_to_check, inplace=True)

    #select the columns to save
    cols_to_save = ['id'] + columns_to_check
    df.loc[:, cols_to_save].to_csv("crime.csv", index=False)
```

Step 4: we remove duplicates from the file with just the columns we want

```python
def removeduplicates():
    df = pd.read_csv("crime.csv")
    df.drop_duplicates(subset=df.columns.difference(['id']), inplace=True)
    df.to_csv("crime.csv", index=False)
```

Step 5: Use the filter function on excel to check for any invalid rows, or typos. For example in the screenshot below, we have a typo that says NW instead of Northwest, but we know it's a typo because the npu's and neighborhoods make sense. Another example is there only being one Sandy Springs (city) row, and since it would hold no significance in our analysis it was removed.

Midtown	E	17th Street Northwest
Loring Hei	E	17th Street Northwest
Atlantic St	E	17th Street Northwest
Atlantic St	E	17th Street Northwest
Atlantic St	E	17th Street Northwest
Atlantic St	E	17th Street Northwest
Atlantic St	E	17th Street Northwest
Atlantic St	E	17th Street Northwest
Atlantic St	E	17th Street Northwest
Midtown	E	17th Street NW

F	G	H
road	neighbour	city
East Palisades		Sandy Springs

Step 6: before running this, run the removeduplicate function once more to check for any duplicates. and then, this no_duplicates function is made to be put into the functions for the dim and fact table creations

```python
def no_duplicates(input_file, output_file):
    #removes all duplicates, with parameters
    df = pd.read_csv(input_file)
    df.drop_duplicates(subset=df.columns.difference(['id']), inplace=True)
    #reset index
    df.reset_index(drop=True, inplace=True)
    df.reset_index(inplace=True)
    df.rename(columns={'index': 'key'}, inplace=True)
    df.drop('id', axis=1, inplace=True)

    df.to_csv(output_file, index=False)
```

Step 7: we start with the dim table making, from crime. what this function does is it maps the crimes according how severe they are from 1-11, then removes duplicates to create the dim file

```python
def crimedim():
    df = pd.read_csv("crime.csv")

    crime_severity = {
        'HOMICIDE': 0,
        'RAPE': 1,
        'AGG ASSAULT': 2,
        'ROBBERY-PEDESTRIAN': 3,
        'ROBBERY-RESIDENCE': 4,
        'ROBBERY-COMMERCIAL': 6,
        'BURGLARY-RESIDENCE': 7,
        'BURGLARY-NONRES': 8,
        'AUTO THEFT': 9,
        'LARCENY-FROM VEHICLE': 10,
        'LARCENY-NON VEHICLE': 11
    }

    df_severity = df[['id', 'crime']].copy()
    #map the crime severity values to the 'crime' column
    df_severity['crime_severity'] = df_severity['crime'].map(crime_severity)
    #drop invalid rows
    df_severity.dropna(subset=['crime_severity'], inplace=True)
    df_severity.to_csv("crimeIDS.csv", index=False)
    #drops duplicates
    no_duplicates('crimeIDS.csv', 'dimcrime.csv')
```

Step 7, continued: this function works in conjunction with the previous function in step 7. So we are creating 2 csv files, one for the dim table (dimcrime) and the other (crimeIDS) for the fact table, to get the id that we'll later put inside the fact table. In this function, we're modifying the crimeIDS file to change the index of those rows that are the same in the dimcrime file. So say LARCENY-NON VEHICLE, 11 is originally index 7, but after executing the code, all the LARCENY-NON VEHICLE rows should say index 10 since that's what it says in the dimcrime csv file.

```python
def crimeid():
    #for fact table, crime_key
    dimcrime1 = pd.read_csv('dimcrime.csv')
    crimedim = pd.read_csv('crimeIDS.csv')

    key_map = dict(zip(dimcrime1[['crime', 'crime_severity']].apply(tuple, axis=1), dimcrime1['key']))
    crimedim['id'] = crimedim[['crime', 'crime_severity']].apply(tuple, axis=1).map(key_map)

    crimedim.to_csv('crimeIDS.csv', index=False)
```

Example: this is what's in dimcrime

6	BURGLARY-RESIDENCE	7
7	BURGLARY-NONRES	8
8	AUTO THEFT	9
9	LARCENY-FROM VEHICLE	10
10	LARCENY-NON VEHICLE	11

This is what the crimeIDS file should look like after executing both codes. Any rows with LARCENY-NON VEHICLE will have 10 as it's id

id	crime	crime_seve
10	LARCENY-NON VEHICLE	11
8	AUTO THEFT	9
9	LARCENY-FROM VEHICLE	10
8	AUTO THEFT	9
10	LARCENY-NON VEHICLE	11

Step 8: the logic is the same for date dimension, and the two other dimensions. But for this code, what we're doing is we're separating the date (m/d/y format) into multiple columns: year, quarter, month, weekday/weekend, day of the week, date and day.

```python
def datedim():
    df = pd.read_csv('crime.csv', usecols=['id', 'date'])

    #extract year, month, and day from the date column
    df['year'] = pd.DatetimeIndex(df['date']).year

    #extract year quarter
    df['year quarter'] = pd.PeriodIndex(pd.to_datetime(df['date']), freq='Q').strftime('%qQ%y')
    df['year quarter'] = df['year quarter'].str[:-3] + 'Q'

    df['month'] = pd.DatetimeIndex(df['date']).month

    #map the month numbers to month names
    month_names = {1:'January', 2:'February', 3:'March', 4:'April', 5:'May', 6:'June', 7:'July', 8:'August', 9:'September',
    df['month'] = df['month'].map(month_names)

    df['day'] = pd.DatetimeIndex(df['date']).day
    #extract day of the week
    df['day of the week'] = pd.DatetimeIndex(df['date']).day_name()

    #map day of the week to weekday/weekend
    weekend_days = ['Saturday', 'Sunday']
    df['weekday/weekend'] = df['day of the week'].apply(lambda x: 'weekend' if x in weekend_days else 'weekday')

    df.to_csv('dateIDS.csv', index=False)
    no_duplicates('dateIDS.csv', 'dimdate.csv')
```

Step 8, continued: this is the function for the dateIDS. same as the previous step, it is just matching the index with what's in the dim table.

```python
def dateid():
    #for fact table, date_key
    dimdate1 = pd.read_csv('dimdate.csv')
    datedim = pd.read_csv('dateIDS.csv')

    key_map = dict(zip(dimdate1[['date', 'year','year quarter','month','day','day of the week','weekday/weekend']].apply(tuple, axis=1), dimdate1['k
    datedim['id'] = datedim[['date', 'year','year quarter','month','day','day of the week','weekday/weekend']].apply(tuple, axis=1).map(key_map)

    datedim.to_csv('dateIDS.csv', index=False)
```

Step 9: same as the two previous steps, this one's for location1. we are just taking the columns we want for this one, removing duplicates and matching the index

```python
def loc1dim():
    df = pd.read_csv('crime.csv', usecols=['id', 'neighborhood', 'npu'])
    df.to_csv('loc1IDS.csv', index=False)
    no_duplicates('loc1IDS.csv', 'dimloc1.csv')

def loc1id():
    #for fact table, loc1_key
    dimloc11 = pd.read_csv('dimloc1.csv')
    loc1dim = pd.read_csv('loc1IDS.csv')

    key_map = dict(zip(dimloc11[['neighborhood','npu']].apply(tuple, axis=1), dimloc11['key']))
    loc1dim['id'] = loc1dim[['neighborhood','npu']].apply(tuple, axis=1).map(key_map)

    loc1dim.to_csv('loc1IDS.csv', index=False)
```

Step 10: this is for location2. they are functions loc2id and loc2dim on the python script.

```python
def loc2dim():
    df = pd.read_csv('crime.csv', usecols=['id', 'country','state','city','county','road'])
    df.to_csv('loc2IDS.csv', index=False)
    no_duplicates('loc2IDS.csv', 'dimloc2.csv')

def loc2id():
    #for fact table, loc2_key
    dimloc21 = pd.read_csv('dimloc2.csv')
    loc2dim = pd.read_csv('loc2IDS.csv')

    key_map = dict(zip(dimloc21[['country','state','city','county','road']].apply(tuple, axis=1), dimloc21['key']))
    loc2dim['id'] = loc2dim[['country','state','city','county','road']].apply(tuple, axis=1).map(key_map)

    loc2dim.to_csv('loc2IDS.csv', index=False)
```

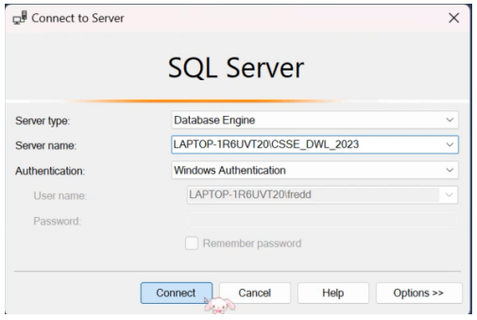
Step 11: this function is for the fact table. from the files we modified earlier (crimeIDS, dateIDS, loc1IDS, loc2IDS), we take the id column for each of those files along with the crime.csv id, and a count column for the powerbi visualisation

```python
def fact():
    df = pd.read_csv('crime.csv', usecols=['id'])
    df = pd.concat([df, pd.read_csv('crimeIDS.csv',
    usecols=['id']).rename(columns={'id': 'crimekey'})], axis=1)
    df = pd.concat([df, pd.read_csv('dateIDS.csv',
    usecols=['id']).rename(columns={'id': 'datekey'})], axis=1)
    df = pd.concat([df, pd.read_csv('loc1IDS.csv',
    usecols=['id']).rename(columns={'id': 'loc1key'})], axis=1)
    df = pd.concat([df, pd.read_csv('loc2IDS.csv',
    usecols=['id']).rename(columns={'id': 'loc2key'})], axis=1)

    #add a count column with 1's
    count = [1]*len(df)
    df['countn'] = count
    df.to_csv('fact.csv', index=False, float_format='%.0f', sep=',')
```

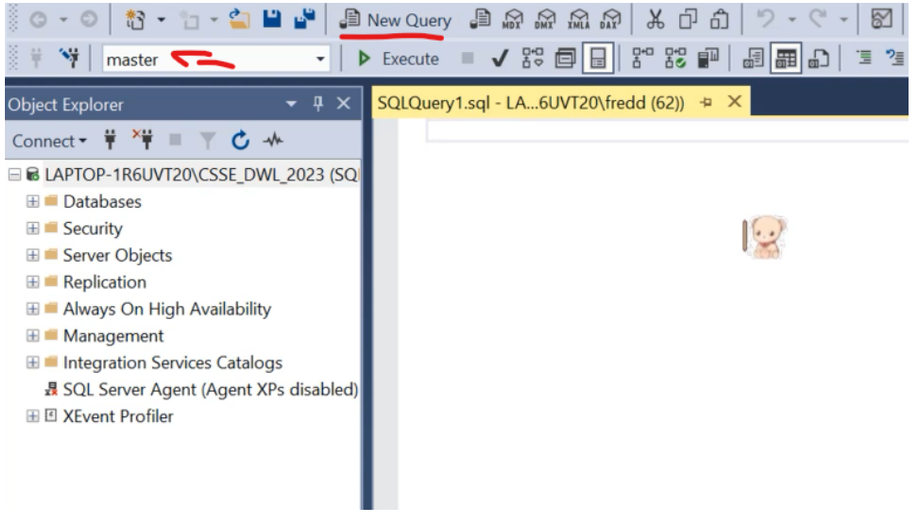
This one is entirely optional, but it just re arranges the columns in the dimension csv files. I just have it there for my convenience, but you could always re-arrange the hierarchies in vscode.

```python
def colshift():
    #reordering columns
    df = pd.read_csv('dimloc1.csv')
    df1 = pd.read_csv('dimloc2.csv')
    df2 = pd.read_csv('dimdate.csv')

    df = df[['key','npu','neighborhood']]
    df1 = df1[['key','country','state','city','county','road']]
    df2 = df2[['key','year','year quarter','month','weekday/weekend','day of the week','date','day']]

    # Save the updated dataframe to a new CSV file
    df.to_csv('dimloc1.csv', index=False)
    df1.to_csv('dimloc2.csv', index=False)
    df2.to_csv('dimdate.csv', index=False)
```

Next up, we will use an SQL query on SSMS to load the warehouse. It is important to check on the SQL server configuration services if your SQL server is running. If it can't be found through the windows search bar, it should be in your Windows>System32 folder.



When loading up SSMS, it will ask you to connect to a server. Enter in your server from the drop down menu and make sure that the server type is on 'Database Engine', since it will not work otherwise. Once you're in, on the left top hand side of the window, you will see a New Query option. Also make sure you are loading the New Query option from the 'master' tab.



We will be talking about the SQL query here onwards, from left to right. First, let's make sure we set our directory/path to where all our csv files are

```
:setvar ProjectSQL "C:\Users\fredd\Documents\Uni\CITS3401 Data Warehousing\Project\"
:setvar DatabaseName "CrimeIncidents"
```

In my query, I have a drop database section of code which basically checks to see if the database named 'CrimeIncidents' already exists, and if it does to delete it and allow us to make a new one. The next code to the top right is one of the ddl statements used to make the database. This only initializes the rows and headers, where crimekey is the primary key of the dimension table. Crimekey will be used as a foreign key in the fact table.

```
CREATE TABLE DimCrime
(
    crimekey INT PRIMARY KEY IDENTITY,
    crime VARCHAR(50),
    crime_severity INT
)
GO
```

After making ddl statements for the rest of the dimensions, we bulk insert the data from our csv files, the one with unique values. The fieldterminator and row terminator is just to separate each value into different columns and rows, while firstrow removes the headers and inputs purely just the values. SQL automatically resets the index from 1 when you populate a table, so KEEPIDENTITY stops that from happening and keeps whatever index you had in your csv file the same when you populate it to the database.

```
BULK INSERT [dbo].[DimCrime]
FROM '$(ProjectSQL)dimcrime.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2,
    KEEPIDENTITY
);
```

For the fact table, putting KEEPIDENTITY won't work alone because of the alter table statements (that will be explained shortly). So we have to put an IDENTITY(0,1) to make sure indexes stay from 0, instead of changing to 1

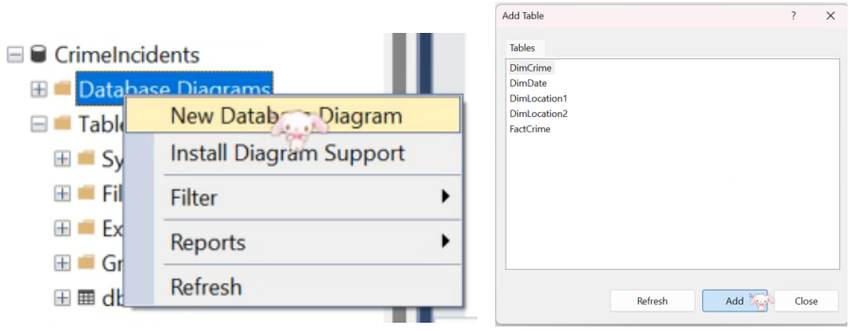
```
Create Table FactCrime
(
id bigint primary key IDENTITY(0,1) not null,
crimekey int not null,
datekey int not null,
loc1key int not null,
loc2key int not null,
countn int not null
)
```

These statements set up foreign keys to the fact table we just made. They connect the tables together, and so we can have the full number of data rows up without actually needing to upload the entire single csv file.

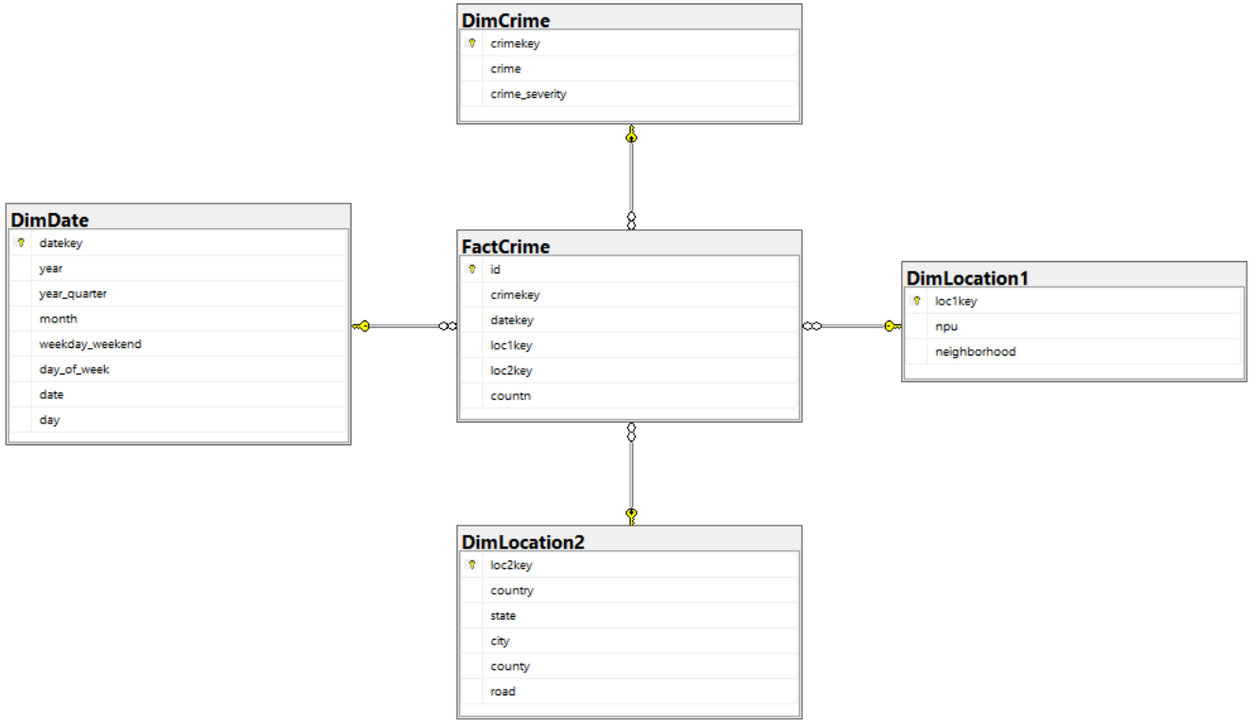
```
ALTER TABLE FactCrime ADD CONSTRAINT
FK_crimekey FOREIGN KEY (crimekey) REFERENCES DimCrime(crimekey);

ALTER TABLE FactCrime ADD CONSTRAINT
FK_datekey FOREIGN KEY (datekey) REFERENCES DimDate(datekey);

ALTER TABLE FactCrime ADD CONSTRAINT
FK_loc1key FOREIGN KEY (loc1key) REFERENCES DimLocation1(loc1key);

ALTER TABLE FactCrime ADD CONSTRAINT
FK_loc2key FOREIGN KEY (loc2key) REFERENCES DimLocation2(loc2key);
```

Then populate the fact table just like how the other dimension tables were done, including KEEPIDENTITY. the IDENTITY(0,1) just sets it so that it starts from 0.

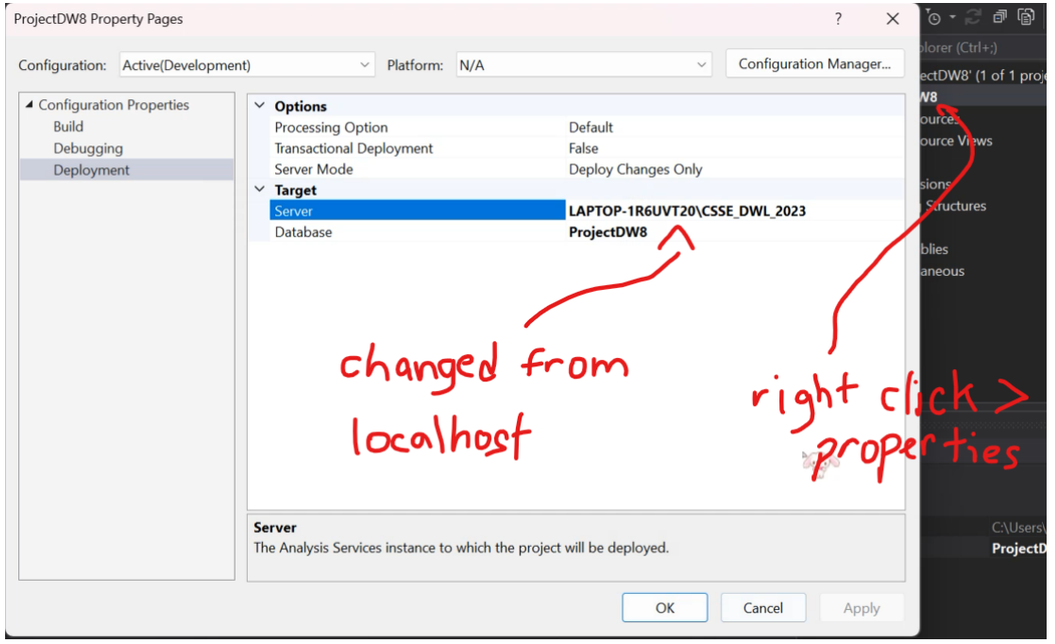
The query should execute with no errors, and upon checking each table the indexes should all start from 0 except for count (because all the values for count is 1). Then, we want to create a database diagram which will be our Star Schema of the fact table, along with the dimensions connected to it.



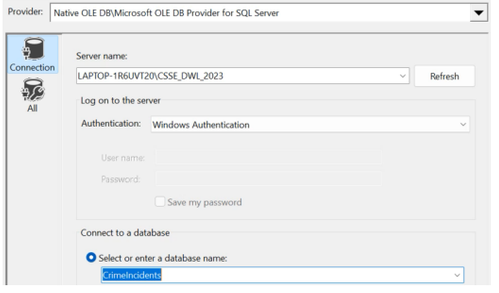
Below is the fact table Star Schema for the DW design. The id represents the id of the crime.csv index, so it is basically the dataset id. The crimekey is the crime list, datekey is the dates, loc1key is the government classified locations, loc2key is the generally classified locations, while countn is the numeric measure for the fact table.

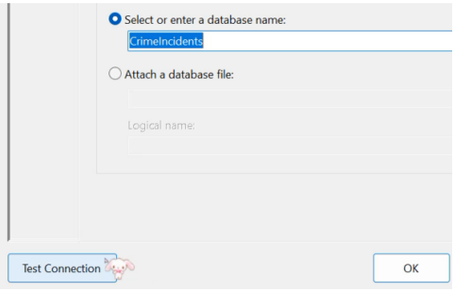


Moving on, we'll be using the Analysis Services Multidimensional Mining Project on Visual Studio 2019 to extract, load and deploy the cube onto the database. When you open it up, click on create a new project and search for the Analysis services project template. On the right hand side you'll see your project name, right click it and select properties. In properties, select deployment and change the server from localhost to your server, the one you used to connect on SSMS. Click apply.



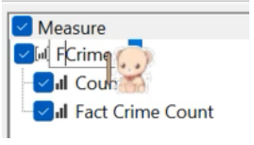
Then, below your project name, you'll see New Data Source. Right click on it and click new data source. When it opens, click 'create data source based on an existing connection', then click new. From the drop down menu at the top, change it to SQL server if it hasn't already, and then enter in the server and database name. Click test connection and click finish.

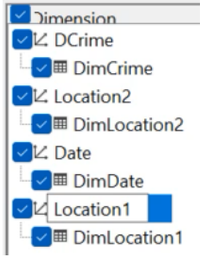


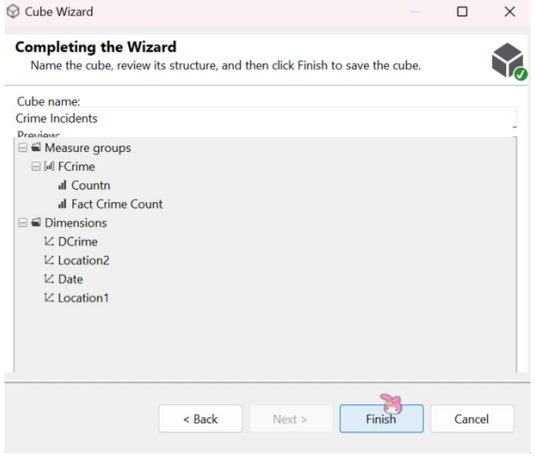


Next, we're going to build a data source view. Right click on data source views and then create new data source. For this one, just click next until you see a table with >>. Click that, then finish.

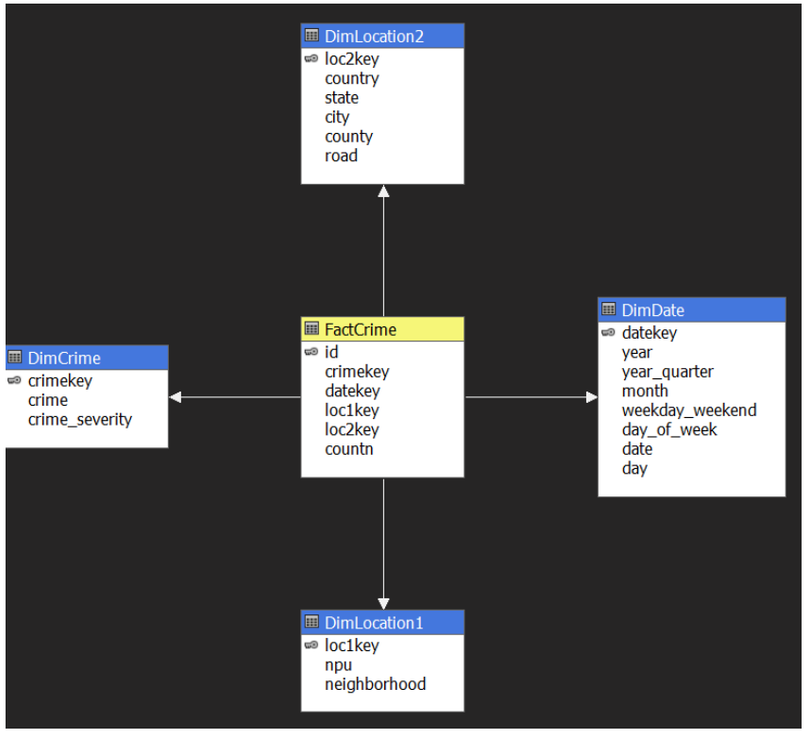
Then, you're going to create a new cube. Right click the cubes button and when it runs, keep it on use existing tables. In the next window, select the Fact table and rename it Crime, or what I did was FCrime incase any errors happened so I'd know where it happened. Do it for DimCrime etc as well, not the tables, just the one with the starnet symbol. Then just click next and then click finish





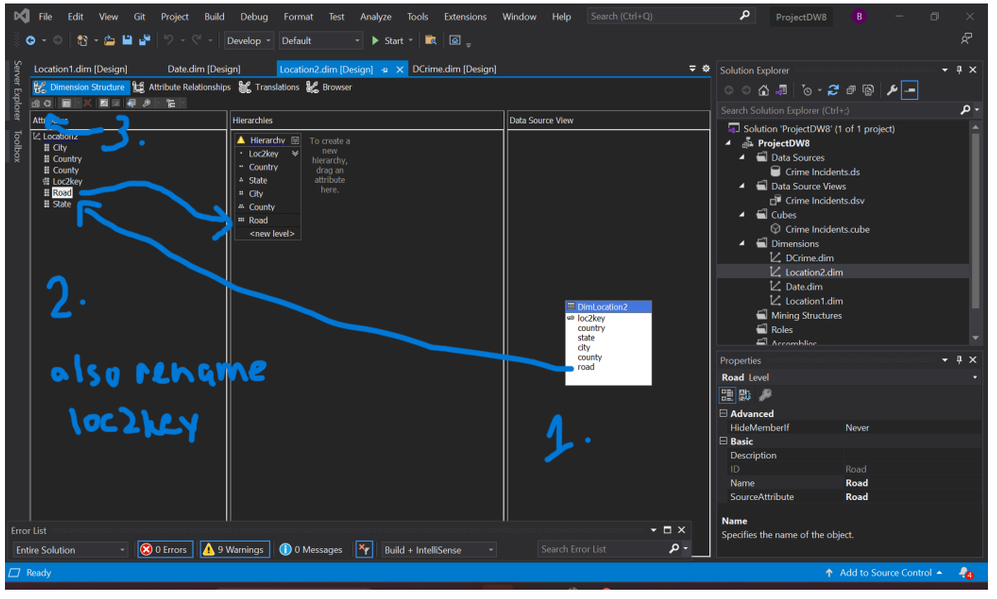


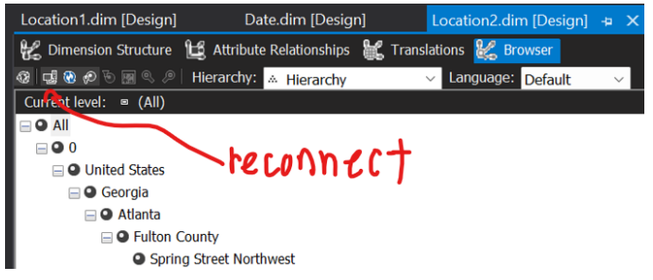
A cube diagram should appear and it should look like this



Multi-dimensional cubes are used to facilitate roll-up and drill-down analysis by providing a way to aggregate and disaggregate data along multiple dimensions. Roll-up involves summarizing data at a higher level of aggregation, while drill-down involves exploring data at a lower level of detail. With a multi-dimensional cube, users can easily move between different levels of aggregation and drill down into the details of specific data points. This allows for more in-depth analysis of the data and helps users to identify trends and patterns that might not be immediately apparent at a higher level of aggregation.

Next, we generate dimensions and fine tune the hierarchies. Click on one of the dims, and when you click browser you'll see that only the index is there. Going back to dimension structure, you'll see that there are 3 columns: attributes, hierarchies and views. In views, you'll see a table of your columns from the dim table, and when you click on one of the columns it should highlight. Move all the columns, except for index, to the attributes column of the window. Once they're all there, rename your id column to something more meaningful, something that will assist you in your analysis. Then, with all those columns in the attributes column, move them to the hierarchy column of the window, and on the left hand side of the page you'll see a little process button. Click on that and it should run, and when you re connect to the browser, it will show the hierarchies.





Now we visualize our analysis on PowerBi. Open up Powerbi and click get data, then choose 'SQL Server Analysis Services Database'. Live connect didn't work for me so I had to import the data, but even though the interface looks weird we are still able to use it to assist with query result visualisation based on our business questions.

Let's start with question 1: Is there a seasonal pattern to certain crime occurrences? How or when, and what crimes?

Diagram 1: Crime count in the 4 different quarters of 2009

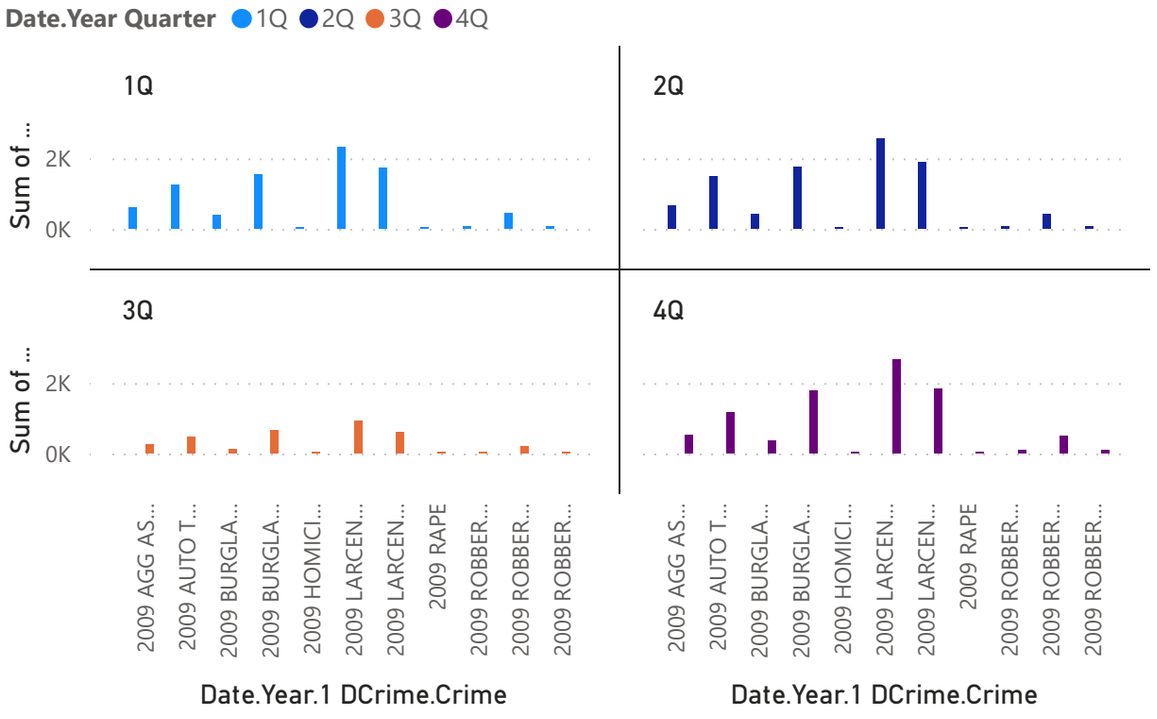
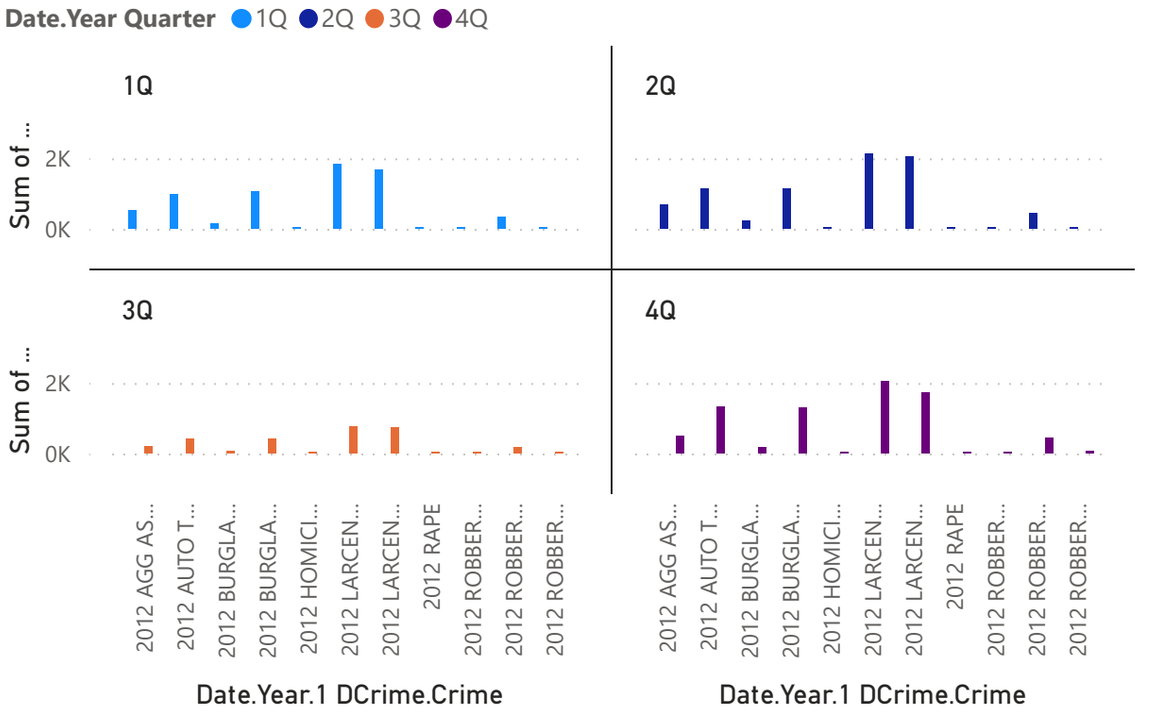


Diagram 2: Crime count in 4 different quarters of 2012



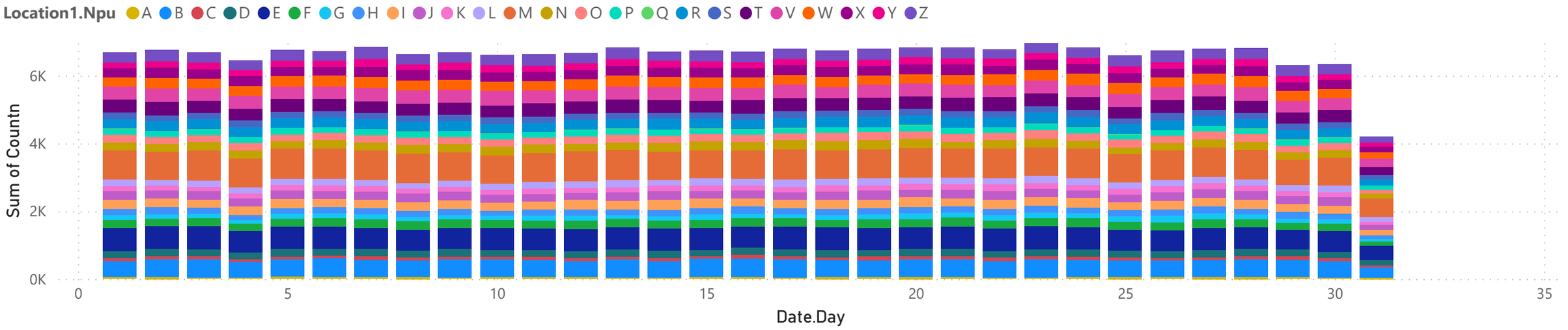
In the diagrams above, there is actually a recurring pattern for Larceny, where it drops in the 3rd quarter and goes up again in the 4th quarter.

For example, in diagram 1, LARCENY - FROM VEHICLE goes up from 2309 to 2568 in the 1st and 2nd quarter, and in the 3rd quarter it goes all the way down to 912 and in the 4th quarter shoots back up to 2666.
As with LARCENY - NON VEHICLE, there is a similar pattern per quarter, but instead of it shooting back up the highest in quarter 4, it is mainly highest in quarter two, at 1882 counts in diagram 1.
In both diagrams, AUTO THEFT and BURGLARY - RESIDENCE are also highest in both quarters 2 and 4.

Overall, in general, there seems to be a trend in more vehicle and place related crimes in the 2nd and 4th quarters, while there is a general trend of there being less crimes in the 3rd quarter which consists of months July-September.

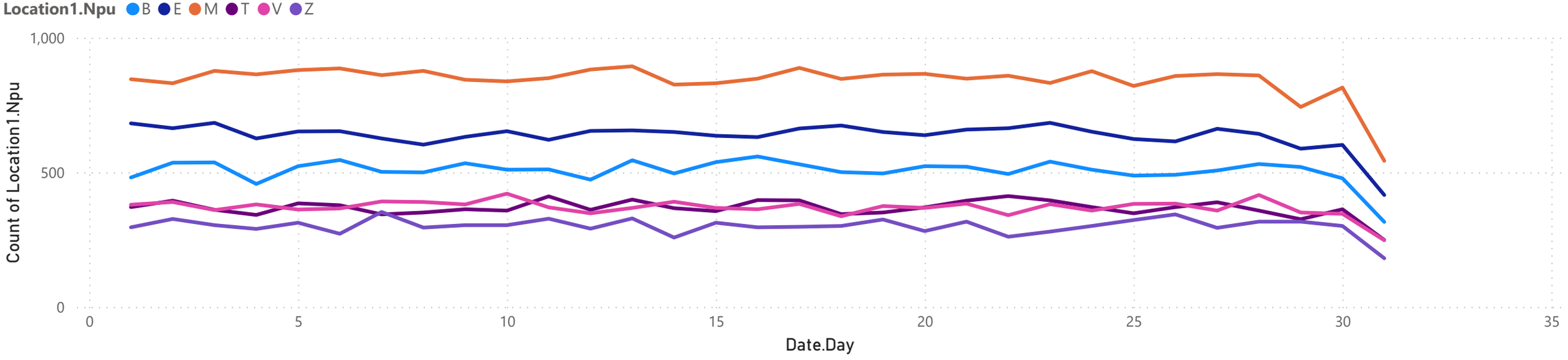
Let's move on to question 2: Do crimes happen in certain areas more often depending on the time of month?

Bargraph of crime count against days of the month over all the years



The graph above makes use of the date index on the fact table so it shows all 205k data. From this graph, there is not really a clear pattern to if crimes rate varies between certain times of the month but we can see that NPU's M and E have the highest crime rates. Let's take a look at those NPU's, and some other NPUs that have a moderate crime and lower crime rate as well. Taking a closer look at those the NPU's.

Line graph of color coded NPU's against the days.

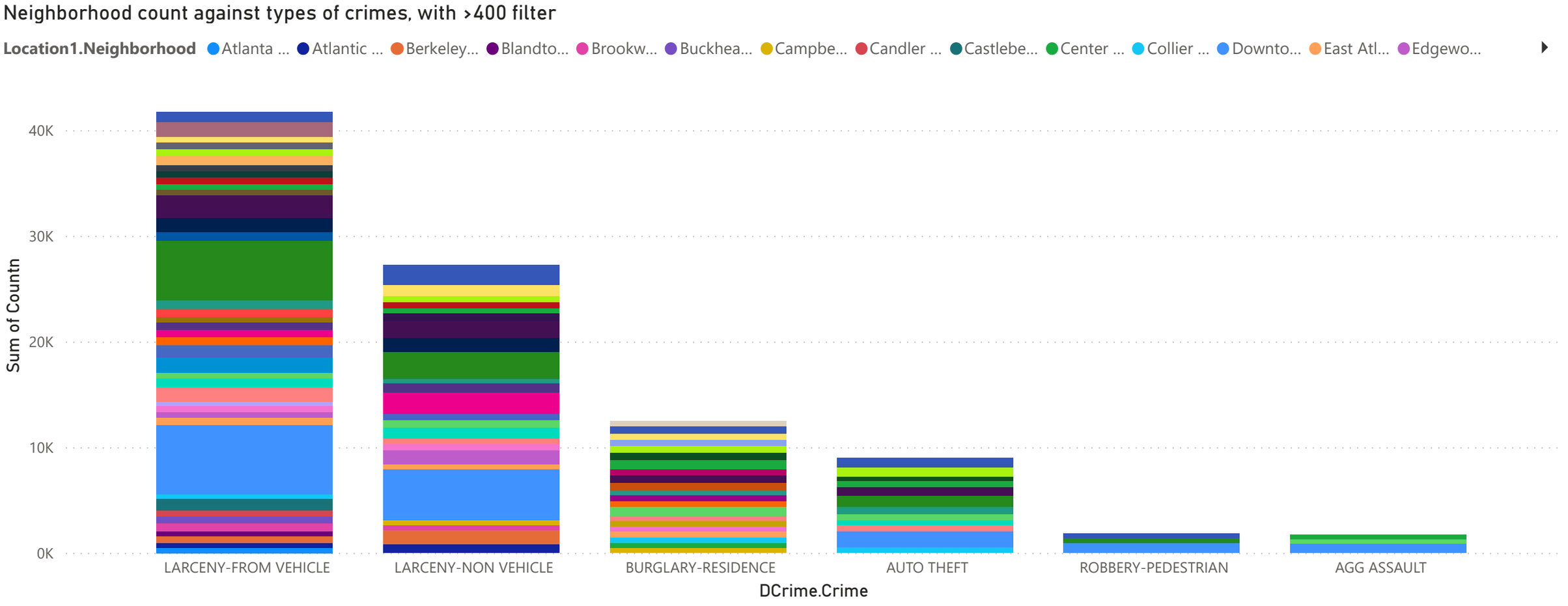


From the 2nd graph in the previous page, we can see that there is a general trend in a spike on day 30 of months. We're going to ignore the fact that it immediately goes down on the 31st, because only some months have a 31st, so it only takes into account half the months of the year; the reason why data rows with 31st as a day was not removed was because if we removed it, it would impact the rest of the analysis that do not ask for the date dimension so it was kept to preserve as much data as possible.

There were also some spikes in all the NPU's except for E on days 13 and then going down on the 14th then coming back up on the 17th, as well as a spike down on all NPU's from the 28th to 29th except for Z.

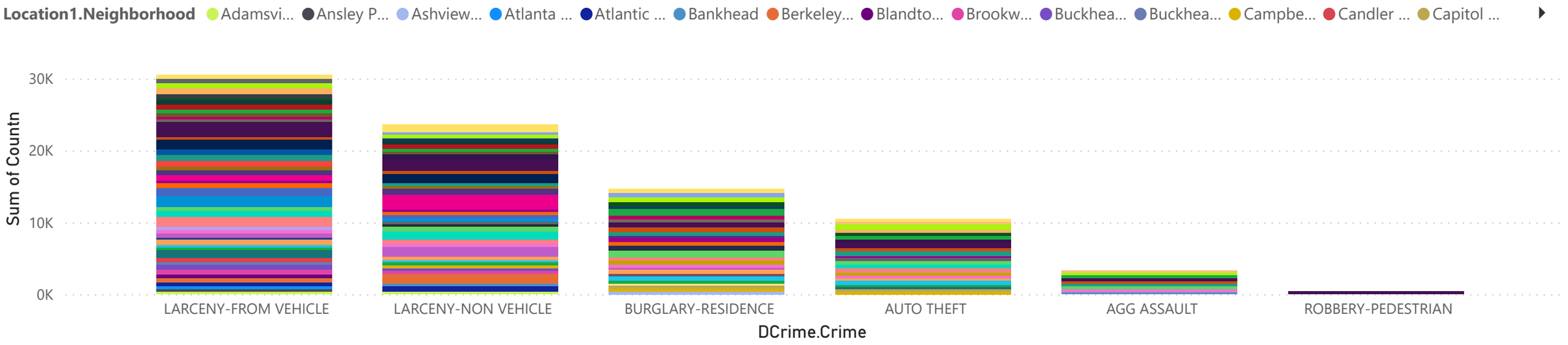
Overall, to answer question 2 crime rates fluctuate depending on the time of month and while there is no recurring pattern, except for the fact that most crime fluctuations happen mid month and towards the end of the month.

1. **Moving on, question 3 is:** Which neighborhoods are highest in each crime and are overall most dangerous? Which roads should be avoided?



The graph in the previous page was filtered to include only neighborhoods with over 400 crimes reported. All of the crimes have Downtown as the biggest crime area reported, and upon closer inspection, we see that it's because Downtown has over 15k reports, so we will remove it in the meantime for our analysis. Let's also change the filter from 300, and remove Midtown since it has 10k and affects the bargraphs when i

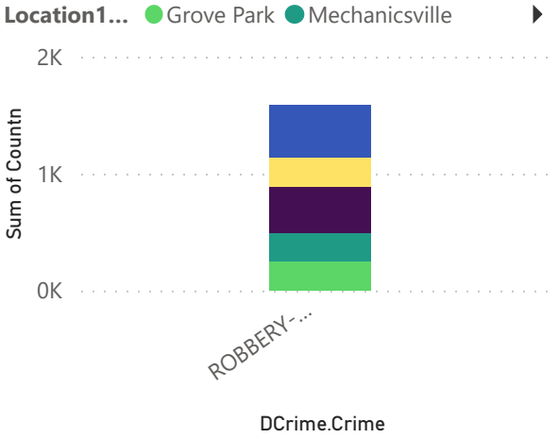
Neighborhood count against crime types with >300 filter and no Downtown & Midtown



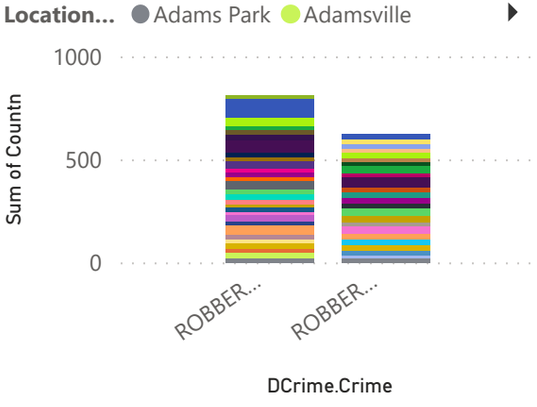
In the graph above, we've excluded Downtown and Midtown and changed the filter to more than 300.

The filter affects the ROBBERY-PEDESTRIAN column, so we will include it with the rest of the crimes that were also affected by the filter, with different filters applied for each.

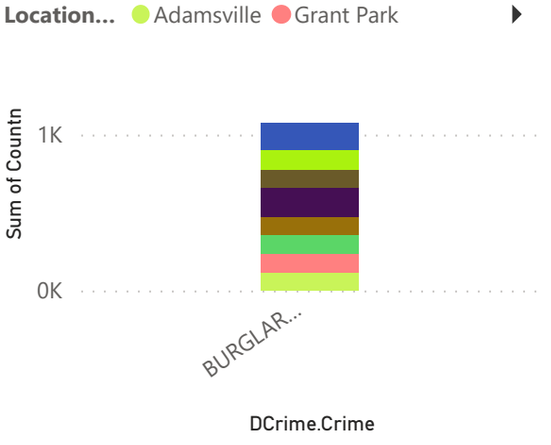
Robbery-Pedestrian, filter >230



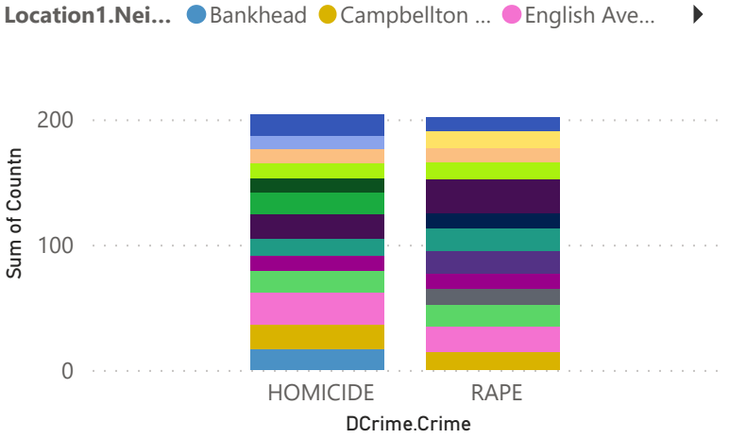
Robbery-Commercial and Robbery Residence, filter >15



Burglary-Non Residence, filter >110



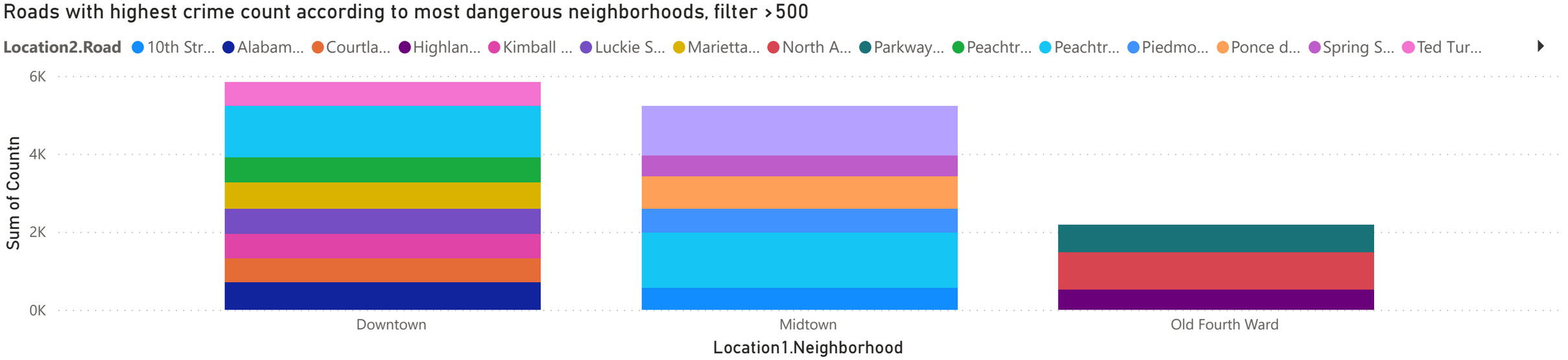
Homicide and rape, filter >10



From the graphs in the previous page, we can infer that the following top 5 neighborhoods have the highest crime rates for each crime type: (all except Downtown and Midtown)

- For HOMICIDE -> English Avenue (26), Old Fourth Ward and Campbellton Road (19), Pittsburgh (18), West End (17)
- For RAPE -> Old Fourth Ward (27), English Avenue (21), Mechanicsville and Landridge/Morosgo (18), Grove Park (17)
- For AGG ASSAULT -> Pittsburgh at 452, Grove Park (418), Old Fourth Ward (397). English Avenue (392), Mechanicsville (355)
- For ROBBERY-PEDESTRIAN -> West End (451), Old Fourth Ward (394), Vine City (254), Grove Park (252), Mechanicsville (239)
- For ROBBERY-COMMERCIAL -> Old Fourth Ward (59), Sylvan Hills and East Atlanta (43), Harland Terrace (40), Landridge/Morosgo (34)
- For ROBBERY-RESIDENCE -> Old Fourth Ward (49), Grove Park (37), English Avenue (36), Pittsburgh and Glenrose Heights (34)
- For AUTO THEFT -> Sylvan Hills at 831, Old Fourth Ward (801), Mechanicsville (693), Grant Park (612), Collier Heights (564)
- For BURGLARY-RESIDENCE -> Grove Park at 984, Pittsburgh (905), Oakland City (726), Old Fourth Ward (709), Southwest and Sylvan Hills (655)
- For BURGLARY-NON RES -> Old Fourth Ward (183), West End (170), Sylvan Hills (131), Grove and Grant Park (121)
- For LARCENY-FROM VEHICLE -> Old Fourth Ward (2191), Home Park (1482), Grant Park (1361), North Buckhead (1297), Inman Park (1107)
- For LARCENY-NON VEHICLE -> Lenox (1997), Old Fourth Ward (1581), North Buckhead (1339), Edgeward (1389), Berkeley Park (1360)

From our gatherings, these are the most dangerous neighborhoods: Downtown, Midtown, Old Fourth Ward, English Avenue, Grove Park and Mechanicsville. Let's dive further deep down to the top 3 neighborhoods to see exactly which roads we should avoid.



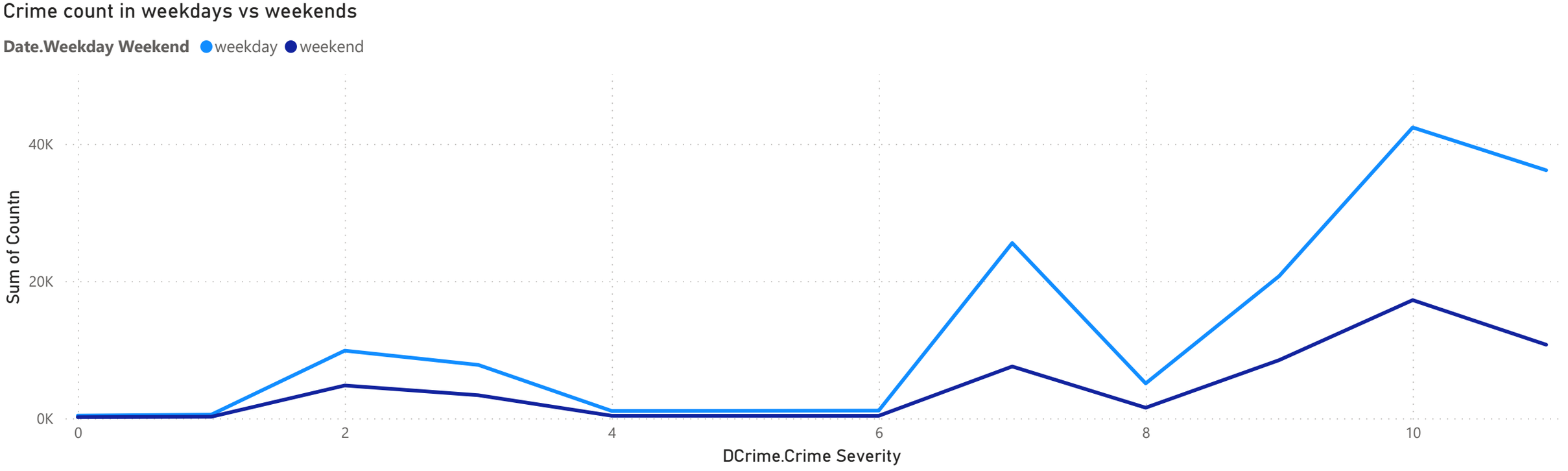
From the graph above, we can see the count of number of crimes in each neighborhood. From each neighborhood:

Downtown: Peachtree Street Northeast (1310), Alabama Street Southwest (706)
Midtown: Peachtree Street Northeast (1428), West Peachtree Street Northwest (1259)
Old Fourth Ward: North Avenue Northeast (965), Parkway Drive Northeast (691)

Therefore the top 3 roads that should be avoided are Peachtree Street Northeast, West Peachtree Street Northwest and North Avenue Northeast.

Question 4: Is there a difference in crime severity between weekdays and weekends?

I realized I accidentally forgot to use the number 5, so instead of it being 1,2,3,4,5,6 in the crime severity table, it is 1,2,3,4,6,7. Either way, 6 is bigger than 4, so we will just use it and pretend 6 is 5, and 7 is 6 so on.



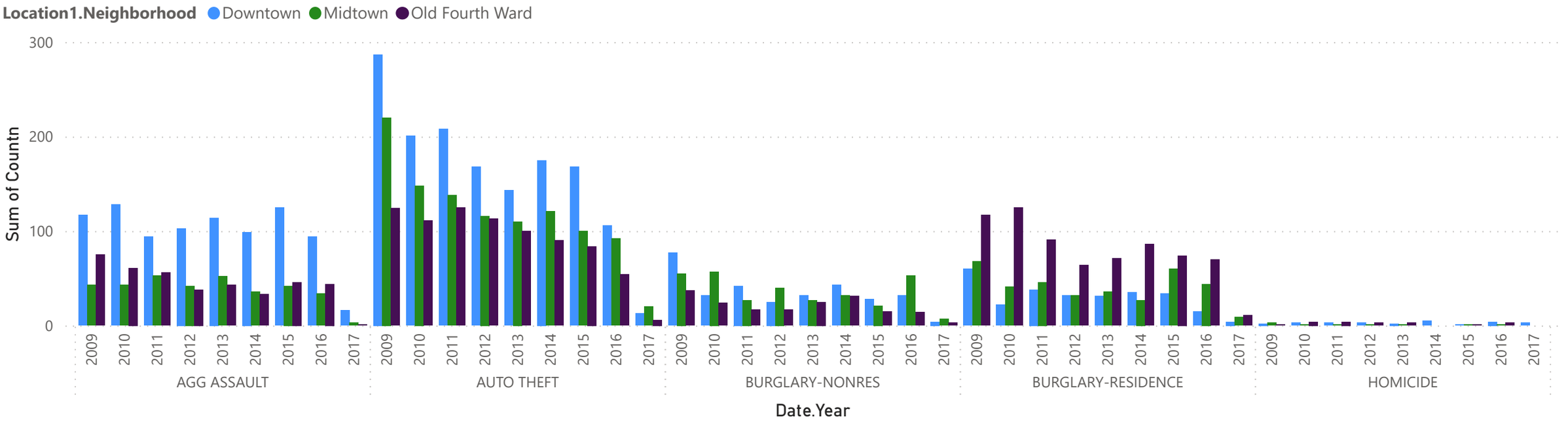
Considering the fact that there are more weekdays than weekends, we'll ignore the fact that the weekday line is alot higher than the weekend line. Looking at the graph, we can see that both weekdays and weekends have the same general trend. So this suggests that no, crime rates are more or less the same on weekends and weekdays since the overall pattern is the same.

Let's move on to our last question, question 5: Do the crime rates change overtime in heavy crime locations?

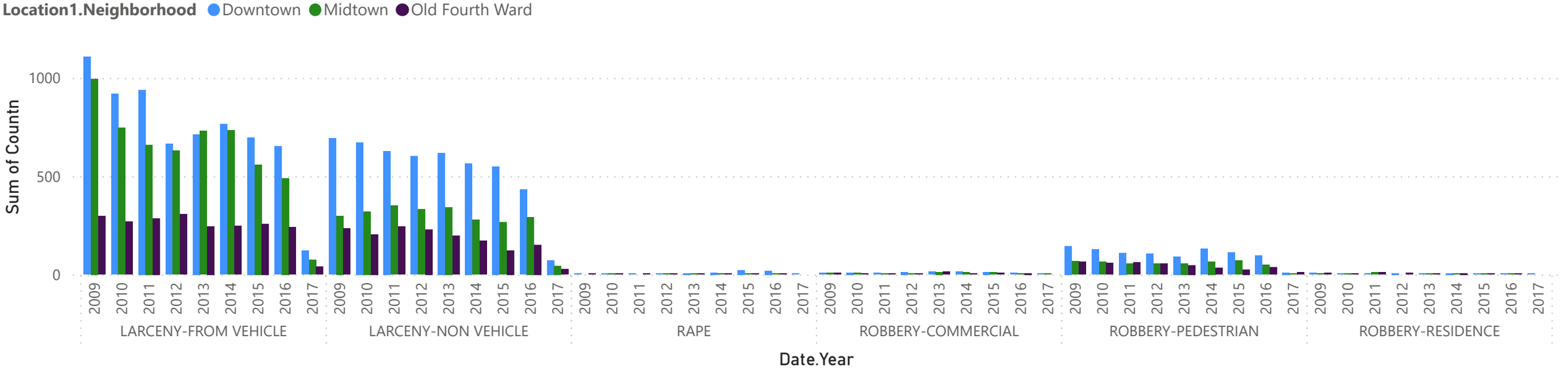
We can use the data from question 3, where we explored the different neighborhoods to determine which were most dangerous. In question 2, we also looked into the different NPU's, and the heavy crime NPU's have the dangerous neighborhoods in it; for instance, Downtown is in NPU M. So we're just going to use the neighborhoods from question 3.

Graphs will be in the next page. For the analysis, we will ignore 2017 since it only records 1 quarter.

Graph for AGG ASSAULT, AUTO THEFT, BURGLARY-NONRES, BURGLARY-RESIDENCE, HOMICIDE



Graph for LARCENY-FROM VEHICLE, LARCENY-NON VEHICLE, RAPE, ROBBERY-PEDESTRIAN, ROBBERY-COMMERCIAL AND ROBBERY-RESIDENCE



From the graphs above, the patterns that can be seen are:

AGG ASSAULT:
All 3 neighborhoods fluctuate, there is no clear pattern

AUTO THEFT:
There is generally a going down trend for all of them, except for year 2014 where the values for Downtown and Midtown spike up before going down again. 2011 for Downtown and Old Fourth Ward too.

LARCENY-FROM VEHICLE:
Only Downtown has a slight downwards trend, but the values fluctuate in-between the years.

LARCENY-NON VEHICLE:
Slightly increases while fluctuating a bit for Midtown, while there is a small downwards trend for Downtown.

RAPE:
Very little crimes committed, but it seems to increase specifically for Midtown towards the end

ROBBERY-COMMERCIAL:
A small spike for all in the middle years, but nothing too much.

ROBBERY-PEDESTRIAN:
It is kind of constant for Midtown except for in 2014 and 2015 when it spikes up, and for Old Fourth Ward there is a downwards trend with only one spike in 2011.

ROBBERY-RESIDENCE:
Small spike in 2011 for Midtown and Old Fourth Ward.

BURGLARY-NONRES:
Not much of a trend, just fluctuates between the years.

BURGLARY-RESIDENCE:
There is a general downwards trend for Old Fourth Ward while it fluctuates for the others.

HOMICIDE:
Fluctuates, although from 2014 there were no murders committed in Midtown and Old Fourth Ward.

So, were we able to properly analyse the dataset? It can be confidently said that the fact and dimension tables can answer the business queries.