CONTENTS

Introduction

The goal of this data warehousing project is to provide a centralized, integrated, and easily accessible repository of data that can be used to support answering business queries.

The data source to design and populate the data warehouse is based on the Crime in Atlanta(2009-2017)

I utilized a range of technologies, including Python as the programing language for data analysis, cleaning, and transformation, SSMS software, and SQL for building and populating the data warehouse with data.

To provide interactive and insightful visualizations, I utilized Power BI to create compelling dashboards and reports. This approach enabled me to effectively analyze complex datasets and provide meaningful insights to my business queries.I have also used Visual Studio for association rule mining.

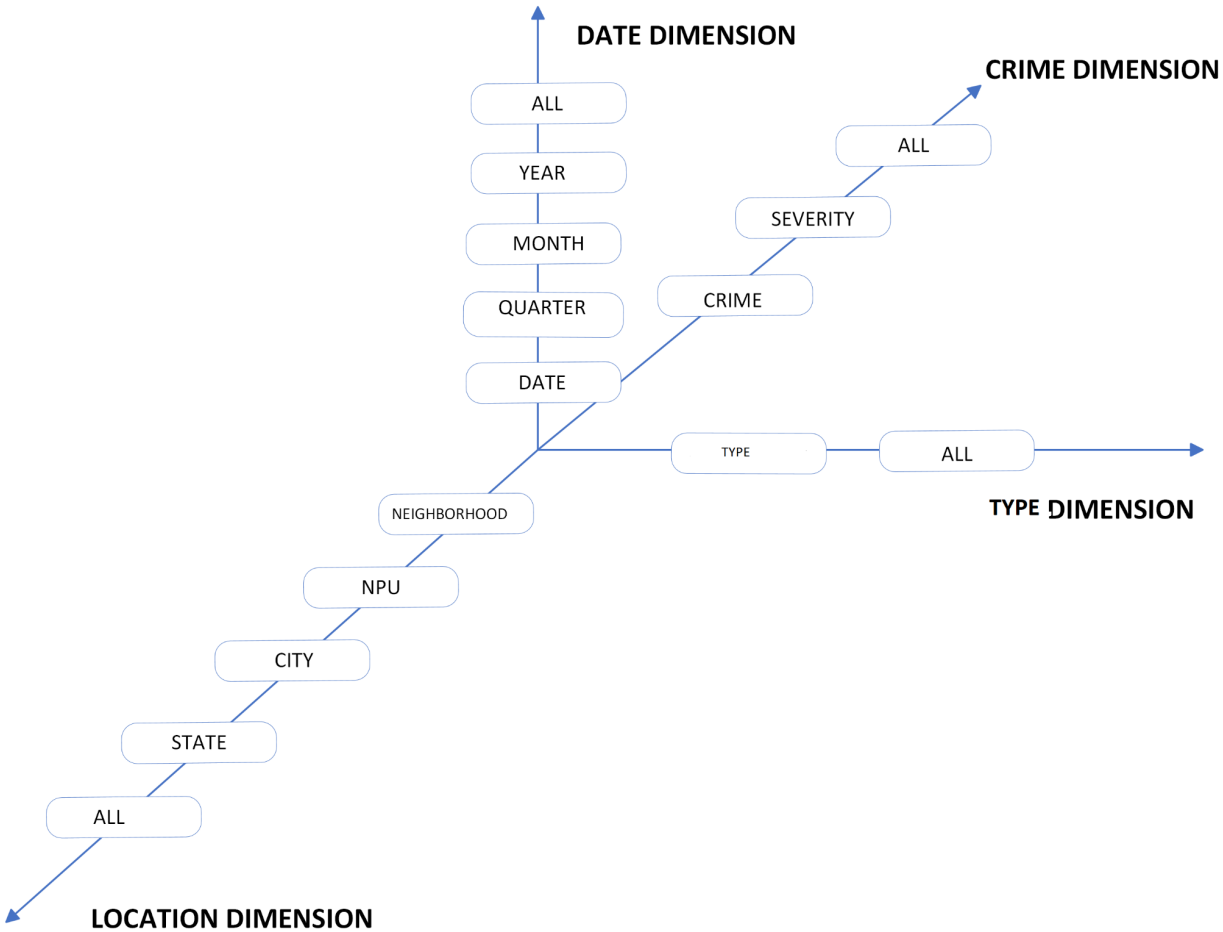
The data warehouse is used to answer these Business queries:

- **Q1.How many crimes were reported in each quarter during Obama's presidential tenure [2009-2016]? How does it compare year-wise? Is there any significant trend?**
- **Q2. What was the most commonly reported crime in the least severe category in each neighborhood?**
- **Q3. Which neighborhood has the highest number of crime reported?**
- **Q4. In 2009, which NPU had the highest number of reported crimes that occurred in offices? Furthermore, within that NPU, which neighborhood had the highest number of reported crimes that occurred in offices?**
- **Q5. Which top three property types had the highest number of crimes reported during president Obama's tenure[2009-2016]**
- **Q6. What is the trend of severity levels of crime annually during Obama's presidential tenure[2009-2016]**

Kimball's Four Steps

- I used Kimball's four steps to design my data warehouse.
    - The process this datawarehouse aims is to analyze the number of crime based on location,date,type and crime dimension.
    - The fact data related to a crime in a particular neighborhood can be stored at a fine-grained level, which includes the type of property where the crime occurred and the date on which it happened.
    - The dimensions:
        1)Date dimension - date, month,quarter, and year
        2)Location dimension - neighborhood,npu,city
        3)Crime - crime, severity
        4)Type

StarNet


DATE DIMENSION
ALL
YEAR
MONTH
QUARTER
DATE
CRIME DIMENSION
ALL
SEVERITY
CRIME
TYPE
ALL
TYPE DIMENSION
NEIGHBORHOOD
NPU
CITY
STATE
ALL
LOCATION DIMENSION

Fact and Dimension Table Design

A.The fact table and dimension tables design explanation:

1. Date dimension -For the date dimension by breaking down the date into components such as months,quarter and year, we can derive valuable insights into trends and patterns across different time frames such as quarterly crime rates,monthly fluctuations, and annual variations. The index column of the date dimension serves as the foreign key and is renamed as dateID

2. Type dimension - The type dimension plays a significant role in providing details about the types of property in which the crimes were reported. It enables us to gain meaningful insights into various aspects such as the types of properties that are more prone to criminal activities, the specific properties that have been targeted, and the nature of crimes associated with them. The index column of the date dimension serves as the foreign key and is renamed as propID

3.Crime Dimension - The crime dimension has been categorized into three distinct levels of severity, namely severe, moderate, and least severe, to create a higher-level concept that facilitates a better understanding of crimes. This categorization enables us to effectively divide crimes into different tiers of severity, which in turn allows for a more nuanced analysis of crime patterns and trends.
The index column is the crime dimension serves as the foreign key and is renamed as crimeID

4.Location Dimension - Lastly, the location neighborhood includes neighborhoods, NPUs, and cities. This allows to get different data about NPU city , neighborhood. However i have decided not to include the followings in the location dimension:

- Location and Road - Both of erroneous data which is further explained in the data observation section of the ETL process.
- County - My business questions did not require county.
- Postcode - My business questions did not require county.
- State - The data set is based in Atlanta.
- Country - The data pertains solely to Atlanta, including the Country field would be redundant, and hence, not required

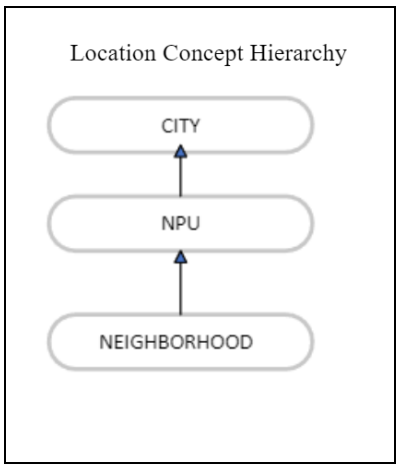
The index column is the location dimension serves as the foreign key and is renamed as locationID

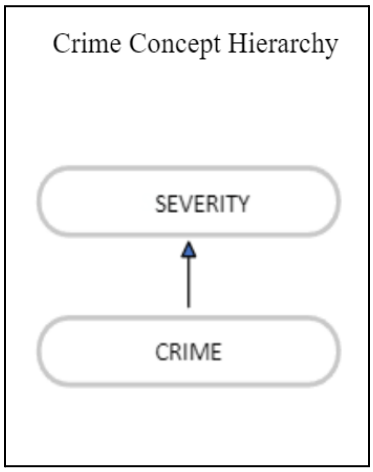
5.Fact table - The Fact table stores locationID,dateID,propID and crimeID as foreign keys in the table.The Fact table contains the unique ID’s of each dimension instead of storing full details of each transaction which help save memory and allows for efficient querying and analysis of large volumes of data. The fact table serves as a lookup table, providing a reference to the dimension table, which stores the descriptive attributes associated with each ID.

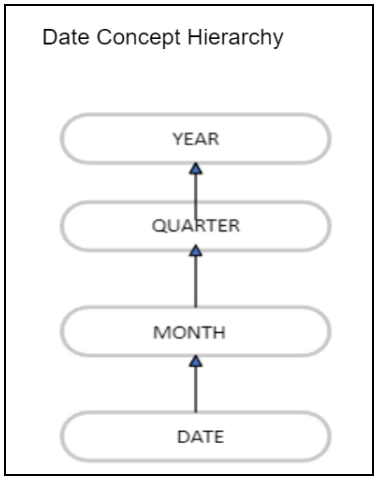
-The index column of the Fact Table was chosen instead of dummy count.It is because the index column already gets created when forming the table and adding another column containing dummy count would be redundant and waste memory space.
-Grain of the fact table: The crime reported for a specific neighborhood, in a specific NPU on a specific day in Atlanta.

The code for how above dimension tables were created and the fact table was connected are descirbed in the ETL process.

B.The Concept Hierarchy and how they support the granularity of the data warehouse:



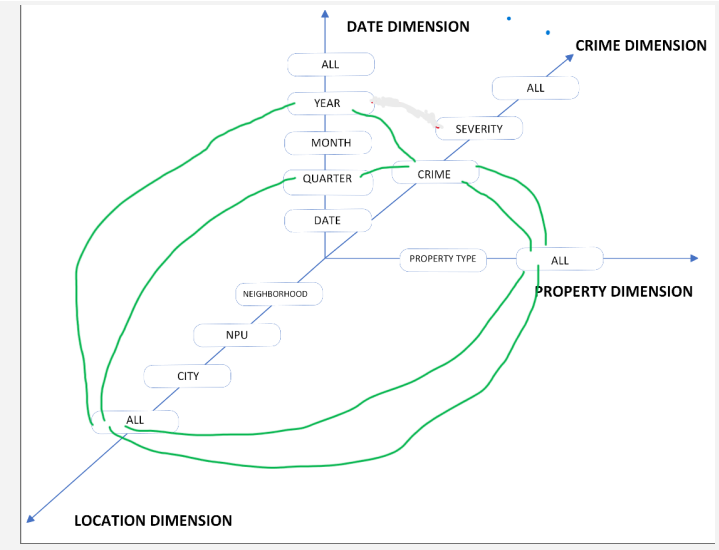




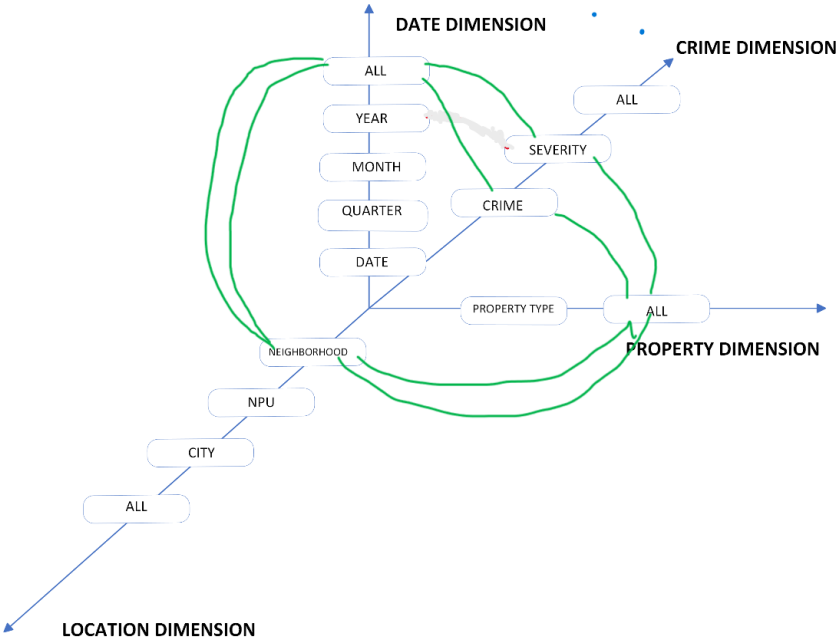
- For Location - the lowest level of granularity is neighborhood.The business queries are designed to focus on overall trends of crimes across different neighborhoods in Atlanta.By using roll-up facilities the concept hierarchy can further help answer questions related to NPU and City.
- For Crime - the lowest level of granularity is Crime which answers questions related to crime types.By rolling up we have a level called severity which is used to encapsulate the different types of crime in the least severe, moderately severe and most severe, which can help answer questions related to severity levels.
- For Date - The lowest of granularity is the date.It helps answer questions related to crime happening on specific dates. Using roll up facilities we can answer questions related to quarterly trends and yearly trends

StarNet Footprints

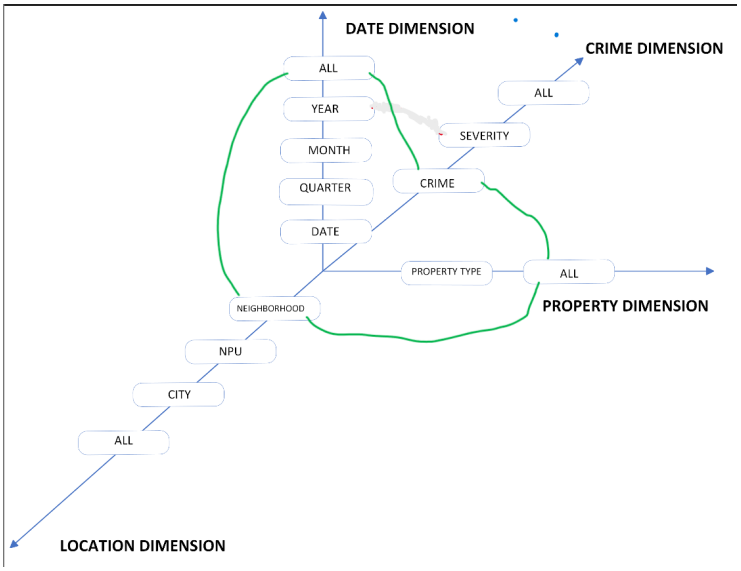
Business query1:How many crimes were reported in each quarter during Obama's presidential tenure [2009-2016]? How does it compare year-wise? Is there any significant trend?



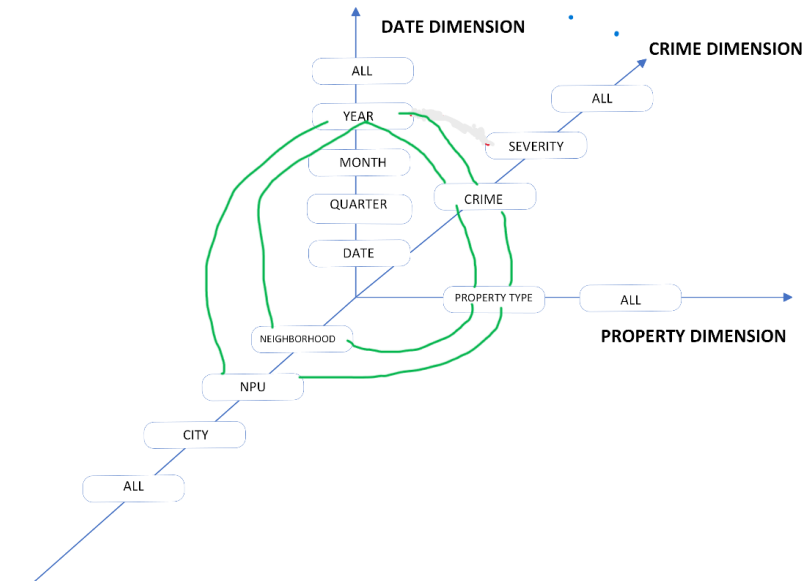
Business query2:What was the most commonly reported crime in the least severe category in each neighborhood?



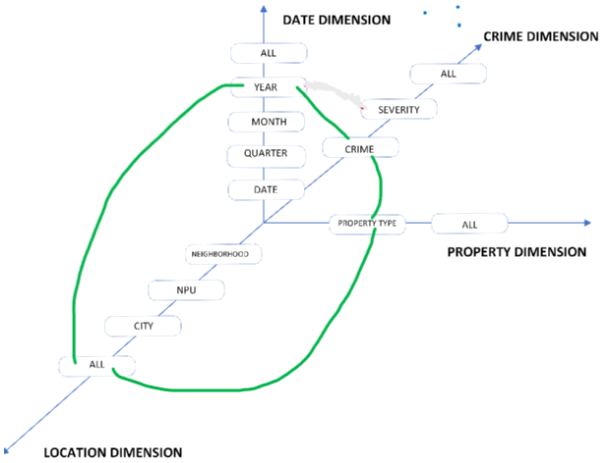
Business query3: Which neighborhood has highest number of crime reported?



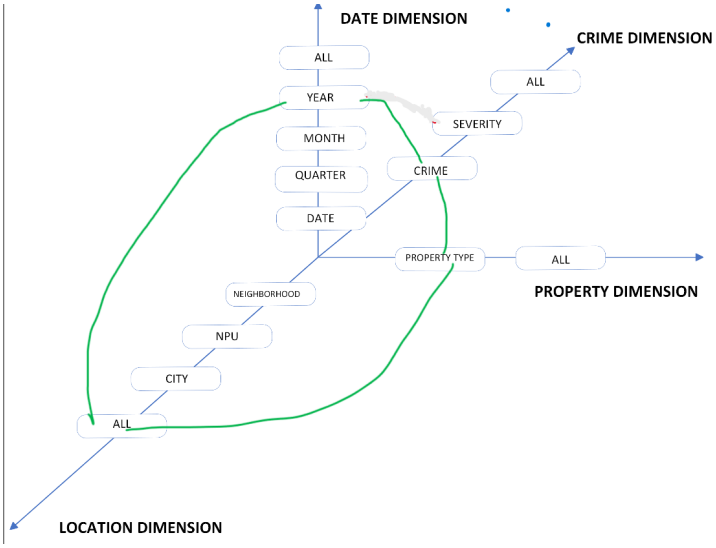
Business query4:In 2009, which NPU had the highest number of reported crimes that occurred in offices? Furthermore, within that NPU, which neighborhood had the highest number of reported crimes that occurred in offices?



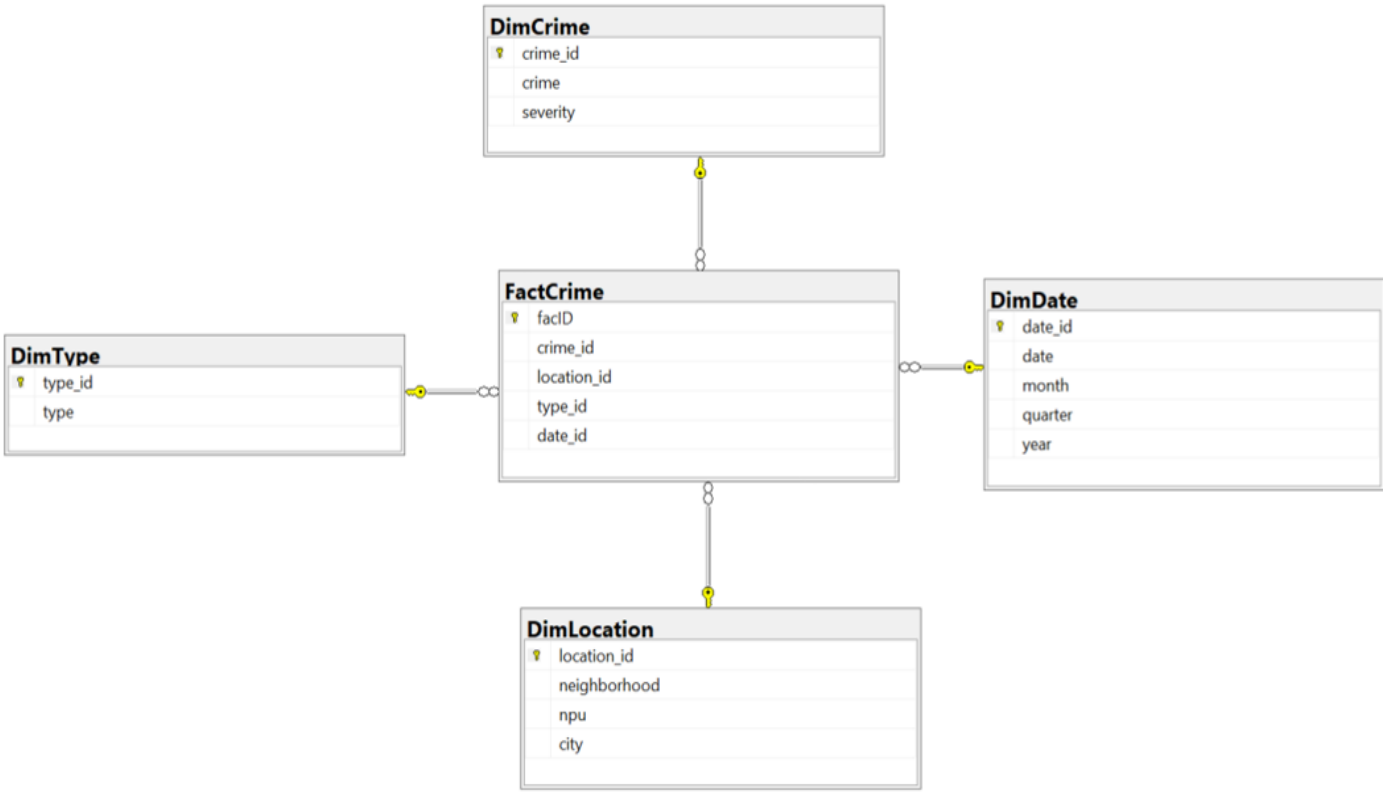
Business query5:Which top three property types had the highest number of crimes reported during president Obama's tenure[2009-2016]



Business query6:What is the trend of severity levels of crime annually during Obama's presidential tenure[2009-2016]



Schema and Concept Hierarchies



A.The reasons for choosing star schema over snowflake and galaxy:

- From the StarNet model I had some sense that has to be a Fact table in the middle connected to four other dimension tables.
- My business queries do not have a further subquery that's why do not need a snowflake.
- dimension tables do not share any columns, the galaxy schema is also not required.
- The schema enables efficient data organization and easy retrieval of information that is required for answering my business queries.It also allows for drill-down and roll up capabilities

ETL Process

❖ Data Source and Extraction

For the data to work with I used the entire dataset by concatenating the data segments given. Data is Crime in Atlanta dataset By utilizing the full dataset, I was able to capture a complete picture of crime activity in Atlanta.

The “crime.csv” had missing values in the type, neighborhood, crime, county, postcode,country after the row n= 25470, Which is why in “crime.csv” I chose rows only until 25470 and used concatenating function to add the other segments.

```
import pandas as pd
```

```
df1= pd.read_csv("crime.csv",nrows=25470)#index=False
df1.to_csv("ds.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")

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

df_concat.to_csv('concat.csv')
```

❖ Data Analysis and Correction

❖ Analysing Location-related variables:Neighborhood, NPU, City

➢ Data Mismatch for NPU and Neighborhood

A)Observations:

1. I Found a discrepancy between the number of unique combinations of NPU, city, and country compared to the number of unique neighborhoods.

```
#data validation for location related variables
df_location = df[["neighborhood","npu","city"]]
#Check the number of unique combinations between neighborhood,npu,city and country
unique_loc = len(df_location[["neighborhood","npu","city"]].drop_duplicates())
#Check the number of unique neighborhood
unique_neighborhood = len(df_location[["neighborhood"]].drop_duplicates())
if(unique_loc != unique_neighborhood):
    print("data mismatch identified")
```

```
data mismatch identified
```

2. I further investigated and found the discrepancy in the number of unique combinations of NPU and neighborhood.

In [18]:
```
#unique combination between npu and neighborhood
unique_loc_nei = len(df_location[["neighborhood","npu"]].drop_duplicates())
if(unique_loc_nei != unique_loc):
    print("mismatch lies in neighborhood and npu")

df_mismatch_npu = df_location[["neighborhood","npu"]].drop_duplicates().groupby("neighborhood").count()
df_mismatch_npu[df_mismatch_npu["npu"]>= 2]
```

```
mismatch lies in neighborhood and npu
```

Out[18]:

neighborhood	npu
Adams Park	2
Almond Park	2
Bankhead	2
Benteen Park	2
Cabbagetown	2
Capitol Gateway	2
Center Hill	2
Chosewood Park	2
Collier Heights	2
Dixie Hills	2
Druid Hills	2
East Chastain Park	2
English Avenue	2
Grant Park	2
Lakewood Heights	2
Midtown	2
Reynoldstown	3
Tuxedo Park	2
Venetian Hills	2
Washington Park	3
Westview	2

fig:*Findings:* **Neighborhoods with multiple NPUs** identified

B)Solution:

- **Approach to fill in Neighborhoods with correct NPUs** -I identified the most frequently assigned Neighborhood Planning Unit (NPU) for each neighborhood and selected the neighborhood-NPU pair with the highest frequency of occurrence. This allows NPUs that are assigned mistakenly due to human error not to be matched as they are lower in number.

```
]:  # Find the number of occurences of an npu being assigned to a neighborhood
    est_loc = pd.DataFrame(df[['neighborhood', 'npu']].groupby(['neighborhood', 'npu'], group_keys = False, as_index = False).siz

    # Choose the neighborhood-npu pair based on the highest number of occurence
    est_loc = test_loc.groupby(['neighborhood'], as_index = False)[['npu', 'size']].max()
    est_loc.drop(columns={'size'}, inplace=True)
    est_loc.to_csv('testDimLoc.csv')

    est_loc
```

:[38]:

	neighborhood	npu
0	Adair Park	V
1	Adams Park	R
2	Adamsville	H
3	Almond Park	J
4	Amal Heights	Y
...	...	...
237	Wilson Mill Meadows	H
238	Wisteria Gardens	H
239	Woodfield	C
240	Woodland Hills	W
241	Wyngate	C

242 rows × 2 columns

In-Text Citation: (*Neighborhood Planning Unit* | *Atlanta City Council, GA*, n.d.)

a)Observation: Missing neighborhood

I have observed that there is a missing data rate of 4% in the neighborhood attribute of the dataset. This poses a significant challenge to the data quality and integrity since the neighborhood serves as the lowest level of granularity for location in the data warehouse design. In light of this issue, I have proposed several data-cleaning options to enhance the quality of the dataset.

b)Solution:

1) **Approach one: Fill using road values:**

Approach one was to fill in missing neighborhood values from road values using mapping and dictionary.However I have noticed some data discrepancy. Road and Neighborhood does not have a one-one relationship.Road can have many neighborhood associated with it,therefore it cannot be the source of truth.We cannot populate the empty neighborhood based on road information because it is not reliable enough.

```
# Gather a unique list of road-neighborhood
nbh_df = df_concat[['location', 'road', 'neighborhood', 'crime']]

# Drop NaN values due to the purpose of this lookup table
# nbh_df.dropna(inplace=True)

nbh_df.reset_index(drop=True, inplace=True)

a_df = nbh_df[nbh_df['road'] == '10th Street Northwest'].sort_values(by = 'location')

#nbh_df.groupby(['road', 'neighborhood'], as_index=False).size()
a_df
```

	location	road	neighborhood	crime
160873	100 -10TH ST NW	10th Street Northwest	Midtown	LARCENY-FROM VEHICLE
953	100 10TH ST NW	10th Street Northwest	Midtown	AUTO THEFT
216553	100 10TH ST NW	10th Street Northwest	Midtown	LARCENY-FROM VEHICLE
151309	100 10TH ST NW	10th Street Northwest	Midtown	LARCENY-FROM VEHICLE
151299	100 10TH ST NW	10th Street Northwest	Midtown	LARCENY-FROM VEHICLE
...	...	...	...	...
62219	I-75-85 NB EXPY NW / 10TH ST NW	10th Street Northwest	Midtown	LARCENY-FROM VEHICLE
21948	I-75-85 NB EXPY NW / 10TH ST NW	10th Street Northwest	Midtown	AGG ASSAULT
198937	IMAGE AVE NW / 10TH ST NW	10th Street Northwest	Home Park	AUTO THEFT
93874	NORTHSIDE DR NW / 10TH ST NW	10th Street Northwest	Home Park	LARCENY-FROM VEHICLE
196663	TECHWOOD DR NW / 10TH ST NW	10th Street Northwest	NaN	LARCENY-NON VEHICLE

fig:Findings:As it can be seen 10th Street Northwest is associated with Midtown and HomePark

2)Approach 2: Fill in using Neighborhood values

Second approach was to fill in with neighborhood values. However it is very prone to human typing errors.For example, with the word road,some uses Rd,RD and St etc. It also does not have a one-one relationship.This will make the cleaning and mapping process very complicated and difficult to validate. Therefore I choose not to go with this method because it may even further compromise the data.

3)Approach3 :Approach three is to drop, missing neighborhood values.

My goal is to ensure the data integrity of the location dimension, which has the lowest granularity level in the neighborhood. However, since there is a 4% missing data rate in the neighborhood attribute of the dataset, simply removing those rows with missing neighborhood data would result in the loss of approximately 10,000 rows. This could potentially impact the

analysis results and other columns associated with those dimensions will also be removed. To determine the extent of data retention, I conduct checks below that will help assess the impact of removing these missing neighborhood rows on the overall dataset.

i)Check of data retention of Crime after dropping missing Neighborhood

In [27]:
```
nbh_crime = df[['neighborhood', 'crime']]

# check % of crime occurence
df_withna = nbh_crime.groupby('crime', as_index=False).size()

# check % of crime occurence after dropping NaN from neighborhood
df_wona = nbh_crime.dropna()
df_wona = df_wona.groupby('crime', as_index = False).size()
df_wona
```

Out[27]:

	crime	size
0	AGG ASSAULT	15020
1	AUTO THEFT	30484
2	BURGLARY-NONRES	6741
3	BURGLARY-RESIDENCE	34396
4	HOMICIDE	590
5	LARCENY-FROM VEHICLE	61682
6	LARCENY-NON VEHICLE	51164
7	RAPE	774
8	ROBBERY-COMMERCIAL	1485
9	ROBBERY-PEDESTRIAN	11432
10	ROBBERY-RESIDENCE	1469

```
]:  # check how much the crime % occurence change after we drop na
    check_crime = pd.merge(df_withna, df_wona, on='crime', how='left', suffixes=['withna', 'withoutna'], validate = 'one_to_one')

    check_crime['change'] = (check_crime['sizewithna'] - check_crime['sizewithoutna'])/check_crime['sizewithna'] *100

    check_crime['retention'] = (check_crime['sizewithoutna']/check_crime['sizewithna'])*100
    check_crime
```

t[28]:

	crime	sizewithna	sizewithoutna	change	retention
0	AGG ASSAULT	15785	15020	4.846373	95.153627
1	AUTO THEFT	31897	30484	4.429884	95.570116
2	BURGLARY-NONRES	7091	6741	4.935834	95.064166
3	BURGLARY-RESIDENCE	35788	34396	3.889572	96.110428
4	HOMICIDE	611	590	3.436989	96.563011
5	LARCENY-FROM VEHICLE	64163	61682	3.866714	96.133286
6	LARCENY-NON VEHICLE	53867	51164	5.017914	94.982086
7	RAPE	805	774	3.850932	96.149068
8	ROBBERY-COMMERCIAL	1534	1485	3.194263	96.805737
9	ROBBERY-PEDESTRIAN	11900	11432	3.932773	96.067227
10	ROBBERY-RESIDENCE	1569	1469	5.772931	94.227069

fig:Findings: Data Retention of Crime ***above 94%***

i)Check of data retention of Type after dropping missing Neighborhood

In [32]:
```
nbh_ppt = df[['neighborhood', 'type']]

# check % of type occurence
df_ppt_wna = nbh_ppt.groupby('type', as_index=False).size()

# check % of type occurence after dropping NaN from neighborhood
df_ppt_wona = nbh_ppt.dropna()
df_ppt_wona = df_ppt_wona.groupby('type', as_index = False).size()
df_ppt_wna
```

Out[32]:

	type	size
0	aeroway	853
1	amenity	22505
2	building	13941
3	city	2873
4	club	50
5	county	2
6	craft	105
7	emergency	1
8	hamlet	57
9	healthcare	1
10	highway	352
11	historic	784
12	house_number	147371
13	landuse	113
14	leisure	1805
15	man_made	131
16	neighbourhood	1704
17	office	1560
18	place	255
19	quarter	69
20	railway	134
21	residential	109
22	retail	191
23	road	10573
24	shop	15339
25	suburb	486
26	tourism	3594
27	town	42

3]:
```
# check how much the type % occurence change after we drop na
check_ppt = pd.merge(df_ppt_wna, df_ppt_wona, on='type', how='left', suffixes=['withna', 'withoutna'], validate = 'one_to_one

check_ppt['change'] = (check_ppt['sizewithna'] - check_ppt['sizewithoutna'])/check_ppt['sizewithna'] *100

check_ppt['retention'] = check_ppt['sizewithoutna']/check_ppt['sizewithna']*100
check_ppt['representation_original'] = (check_ppt['sizewithna'] / (sum(check_ppt['sizewithna']))) * 100
check_ppt['repre_new'] = (check_ppt['sizewithoutna'] / check_ppt['sizewithoutna'].sum(skipna=True)) * 100
check_ppt
```

ut[33]:

	type	sizewithna	sizewithoutna	change	retention	representation_original	repre_new
0	aeroway	853	15.0	98.241501	1.758499	0.379111	0.006969
1	amenity	22505	21213.0	5.740946	94.259054	10.002222	9.855647
2	building	13941	13468.0	3.392870	96.607130	6.196000	6.257288
3	city	2873	2586.0	9.989558	90.010442	1.276889	1.201466
4	club	50	50.0	0.000000	100.000000	0.022222	0.023230
5	county	2	NaN	NaN	NaN	0.000889	NaN
6	craft	105	102.0	2.857143	97.142857	0.046667	0.047390
7	emergency	1	1.0	0.000000	100.000000	0.000444	0.000465
8	hamlet	57	57.0	0.000000	100.000000	0.025333	0.026482
9	healthcare	1	1.0	0.000000	100.000000	0.000444	0.000465
10	highway	352	352.0	0.000000	100.000000	0.156444	0.163541
11	historic	784	783.0	0.127551	99.872449	0.348444	0.363785
12	house_number	147371	142063.0	3.601794	96.398206	65.498222	66.003057
13	landuse	113	113.0	0.000000	100.000000	0.050222	0.052500
14	leisure	1805	1700.0	5.817175	94.182825	0.802222	0.789827
15	man_made	131	129.0	1.526718	98.473282	0.058222	0.059934
16	neighbourhood	1704	1445.0	15.199531	84.800469	0.757333	0.671353
17	office	1560	1550.0	0.641026	99.358974	0.693333	0.720136
18	place	255	255.0	0.000000	100.000000	0.113333	0.118474
19	quarter	69	69.0	0.000000	100.000000	0.030667	0.032058
20	railway	134	134.0	0.000000	100.000000	0.059556	0.062257
21	residential	109	109.0	0.000000	100.000000	0.048444	0.050642
22	retail	191	191.0	0.000000	100.000000	0.084889	0.088739
23	road	10573	9735.0	7.925849	92.074151	4.699111	4.522921
24	shop	15339	15259.0	0.521546	99.478454	6.817333	7.089394
25	suburb	486	355.0	26.954733	73.045267	0.216000	0.164934
26	tourism	3504	3484.0	3.060657	96.939343	1.557333	1.618681
27	town	42	18.0	57.142857	42.857143	0.018667	0.008363

fig:Findings:After conducting checks on the impact of removing rows of type column with missing neighborhood data, I have determined that the data retention rate is high, except for the aeroway, county, and town. However, upon analyzing the representation_original column, it was evident that these attributes have low representation in the original dataset. Therefore, removing these rows with missing neighborhood data ***would not significantly compromise the data integrity.***

iii)Data retention check for date

In [29]:
```
nbh_date = df[['neighborhood', 'date']]

# check % of date occurence
df_date_wna = nbh_date.groupby('date', as_index=False).size()

# check % of date occurence after dropping NaN from neighborhood
df_date_wona = nbh_date.dropna()
df_date_wona = df_date_wona.groupby('date', as_index = False).size()
df_date_wona
```

Out[29]:

	date	size
0	2009-01-01	113
1	2009-01-02	137
2	2009-01-03	110
3	2009-01-04	106
4	2009-01-05	129
...	...	...
2502	2017-02-24	44
2503	2017-02-25	55
2504	2017-02-26	44
2505	2017-02-27	64
2506	2017-02-28	59

2507 rows × 2 columns

In [30]:
```
# check how much the date % occurence change after we drop na
check_date = pd.merge(df_date_wna, df_date_wona, on='date', how='left', suffixes=['withna', 'withoutna'], validate = 'one_to_

check_date['change'] = (check_date['sizewithna'] - check_date['sizewithoutna'])/check_date['sizewithna'] *100

check_date['retention'] = check_date['sizewithoutna']/check_date['sizewithna']*100
```

In [31]:
```
len(check_date[check_date['retention'] > 90]) / len(check_date) * 100
```

Out[31]: 98.60390905464699

fig:Findings: The data retention for date in the case of removing neighborhood is above 98%

Conclusion: After a thorough evaluation of the advantages and disadvantages of removing the rows with missing neighborhood data, I have concluded that it is the optimal solution. Despite the loss of data in type attribute, such as aeroway, county, and town, the **high retention rate of the overall dataset remains intact** even after removing the missing neighborhood values. Attempting to fill in the missing data could lead to inaccuracies and **compromise the integrit**y of the data. Hence, the decision to drop the missing neighborhood values is the most viable and reliable approach.

3) Data check for City

a)Observation:

1. In order to verify the accuracy of the city data, I conducted a check on a neighborhood level to identify any conflicting records of the city.

```
# Check whether one neighborhood has conflicting records of cities
city_check = df_concat[['neighborhood', 'city']].drop_duplicates().groupby(['neighborhood', 'city'], as_index=False).size()

if len(city_check[city_check['size'] != 1]) == 0:
    print("No duplicated records for neighborhood and city. Each neighborhood has 1 corresponding city")
```

```
No duplicated records for neighborhood and city. Each neighborhood has 1 corresponding city
```

fig:Findings:No discrepancies were found in the city data within that particular neighborhood.

2. Subsequently, I detected the presence of empty rows under the 'city' column once the 'neighborhood' data was removed.

b)Solution:

Approach taken to fix: empty rows under the 'city' column once the 'neighborhood' data was removed.
I created a dictionary that maps a neighborhood with its corresponding cities.

]:
```
# Check whether there are empty city rows when neighborhood is removed
nbh_city = df[['neighborhood', 'city']]
nbh_full = nbh_city[nbh_city['neighborhood'].isna()]
if len(nbh_full[nbh_full['city'].isna()]) != 0:
    print('Empty rows detected under "City" column after "Neighborhood" is removed ')

    # Create a dictionary which maps a neighborhood with their corresponding city
    city_mapper = df[['neighborhood', 'city']].groupby(['neighborhood', 'city'], as_index=False).size()
    city_mapper.set_index(city_mapper['neighborhood'], inplace=True)
    city_mapper.drop(columns={'size'}, inplace=True)
    city_dict = dict(city_mapper.to_dict(orient='split')['data'])
```

```
Empty rows detected under "City" column after "Neighborhood" is removed
```

]:
```
city_mapper
```

:[36]:

neighborhood	neighborhood	city
Adair Park	Adair Park	Atlanta
Adams Park	Adams Park	Atlanta
Adamsville	Adamsville	Atlanta
Almond Park	Almond Park	Atlanta
Amal Heights	Amal Heights	Atlanta
...	...	...
Wilson Mill Meadows	Wilson Mill Meadows	Atlanta
Wisteria Gardens	Wisteria Gardens	Atlanta
Woodfield	Woodfield	Atlanta
Woodland Hills	Woodland Hills	Atlanta
Wyngate	Wyngate	Atlanta

243 rows × 2 columns

]:
```
df['city'] = df['city'].fillna(df["neighborhood"].map(city_dict))
```

4) Data check for Date
a) Observation:

1)I conducted the check to see if the dates are valid.
I conducted a check to identify whether there were any instances of dates in February with the value '29', which would be invalid for non-leap years.

```
#Checking if there are dates in february that are 29 but not leap years
non_leap_years = df.loc[(df['date'].dt.month == 2) & (df['date'].dt.day == 29) & (~df['date'].dt.is_leap_year)]

# print the result
if len(non_leap_years) > 0:
    print("There are some dates in February with day 29 that are not leap years:")
    print(non_leap_years)
else:
    print("There are no dates in February with day 29 that are not leap years.")
```

```
There are no dates in February with day 29 that are not leap years.
```

ii)Next, I performed a validation check to identify any dates that exceed the maximum limit of 31 days in a month.

```
# check if there are any dates with day greater than 31
invalid_dates = df.loc[df['date'].dt.day > 31]

# print the result
if len(invalid_dates) > 0:
    print("There are some invalid dates:")
    print(invalid_dates)
else:
    print("All dates are valid.")
```

```
All dates are valid.
```

iii)Check for missing values in date column

```
# check for missing values
if df["date"].isnull().any().any():
    print("There are missing values in the DataFrame.")
else:
    print("There are no missing values in the DataFrame.")
```

```
There are no missing values in the DataFrame.
```

b)Solution:
No unusual data dound in date column.

4) Data check for a crime:
Observation:
check to identify any misspelled words that could potentially result in redundant or erroneous data.

```
54]:  correct_spells = df["crime"].unique()
      # check if all the values have the same spelling
      for crime in df["crime"]:
          if crime not in correct_spells:
              print(f'The word "{crime}" is misspelled.')
```

COnculsion:

(a) No mispelled data

5) Data check for type:
Observation:

Similarly as before, check to identify any misspelled words that could potentially result in redundant or erroneous data.

```
correct_spells = df["crime"].unique()
# check if all the values have the same spelling
for crime in df["crime"]:
    if crime not in correct_spells:
        print(f'The word "{crime}" is misspelled.')
```

Solution:
(a)No misspelled data

DIMENSION TABLE CREATION

i)DATE DIMENSION

```
#Date dimension
date_df = df[["date"]].drop_duplicates()
date_df = date_df.reset_index(drop=True)
date_df = date_df.rename_axis("date_id")

date_df['month'] = date_df['date'].dt.month
date_df['quarter'] = date_df['date'].dt.quarter#q1
date_df['year'] = date_df['date'].dt.year
```

ii) CRIME DIMENSION

```
#crime dimension
crime_df = df[["crime"]].drop_duplicates()
crime_df = crime_df.reset_index(drop=True)
crime_df=crime_df.rename_axis("crime_id")

def assign_severity(crime):
    if crime in ['HOMICIDE', 'RAPE', 'AGG ASSAULT']:
        return 'Most severe'
    elif crime in ['BURGLARY-RESIDENCE', 'ROBBERY-RESIDENCE', 'ROBBERY-PEDESTRIAN',
                   'ROBBERY-COMMERCIAL', 'AUTO THEFT', 'LARCENY-FROM VEHICLE']:
        return 'Moderately severe'
    else:
        return 'Least severe'#LIST FOR LEAST SEVERE , MENTION EXHAUSTED

# apply the function to the 'crime' column to create a new 'severity' column
crime_df['severity'] = crime_df['crime'].apply(assign_severity)
```

iii) TYPE DIMENSION

```
#type dimension
type_df = df[['type']].drop_duplicates()
type_df= type_df.reset_index(drop=True)

type_df = type_df.rename_axis("type_id")
```

iv) LOCATION DIMENSION

```
#Location dimension

location = df[["neighborhood"]].drop_duplicates()

location_df = pd.merge(location,df[["npu","city"]],left_index=True,right_index=True,how= "left")
loc2 = location_df.copy()
loc2 = loc2.reset_index(drop = True)
location_df = location_df.reset_index(drop=True)
location_df = location_df.rename_axis("location_id")
location_df
```

FACT TABLE CREATION

Approach Merge vs Mapping: Merge is more efficient when theres multiple columns that needs to be included in the data. However, in this case, I only want to populate the fact table with the index values from each of the dimension table. Therefore, mapping will be more efficient & storage-wise.

```
# Create a crime dictionary for mapping
factTable = df.copy()
crime = crime_df[["crime"]]
crimeDict = crime_df.to_dict()['crime']

# Reverse the dictionary to map the crime type to crimeID
crimeDict= {v:k for k, v in crimeDict.items()}

# Map crime type to crimeID
factTable['crimeID'] = factTable.crime.map(crimeDict)
```

```
# Create a dictionary with locationID & neighborhood
location = location_df[["neighborhood"]]
locationDict = location.to_dict()["neighborhood"]

locationDict2 = {v:k for k,v in locationDict.items()}

factTable['locationID'] = factTable["neighborhood"].map(locationDict2)
```

```
prop = type_df[["type"]]

propDict = prop.to_dict()['type']
propDict = {v:k for k,v in propDict.items()}

factTable['propID'] = factTable['type'].map(propDict)
factTable.to_csv('testFact')
```

```
date_df['date'] = date_df['date'].astype("datetime64[ns]")
dateDict = date_df.to_dict()['date']

dateDict2 = {v:k for k,v in dateDict.items()}

factTable['dateID'] = factTable['date'].map(dateDict2)

fac = factTable[["crimeID","locationID","propID","dateID"]]

fac = fac.rename_axis("facID")
```

Out[65]:

facID	crimeID	locationID	propID	dateID
1	1	1	1	1
2	2	2	2	1
3	3	3	3	1
4	2	4	1	1
5	1	5	1	1
...	...	...	...	...
224996	4	110	1	2506
224997	3	12	4	2507
224998	1	86	1	2507
224999	4	117	4	2507
225000	5	114	1	2507

215237 rows × 4 columns

1.Index change : I changed the index because in SSMS index starts from one however Pandas index starts from zero.

```
location_df.index = location_df.index+1
type_df.index = type_df.index+1
date_df.index = date_df.index+1
crime_df.index = crime_df.index+1
```

```
fac.index = fac.index+1
```

6) SSMS to plug in data

1) Converted the dimension and fact table into csv files.

```
location_df.to_csv("locationDim.csv")
type_df.to_csv("TypeDim.csv")
date_df.to_csv("DateDim.csv")
crime_df.to_csv("crimeDim.csv")
fac.to_csv("factt.csv")
```

2) Steps in SSMS:

i) Creating and dropping the database

```
PRINT '';
PRINT '*** Dropping Database';
GO
IF EXISTS (SELECT [name] FROM [master].[sys].[databases] WHERE [name] = N'CrimeDW')
DROP DATABASE CrimeDW;
GO
PRINT '';
PRINT '*** Creating Database';
GO
Create database CrimeDW
Go
Use CrimeDW
Go
```

ii) Creating the dimension tables

```
--Create Crime Dimension
Create table DimCrime
(
crime_id int primary key identity,
crime varchar(30),
severity varchar(30)


)
Go

--Create Location Dimention
Create table DimLocation
(
location_id int primary key identity,
neighborhood varchar(100),
npu varchar(1),
city varchar(10)

)
Go

--Create Type Dimension
Create table DimType
(
type_id int primary key identity,
type varchar(50)
)
Go



--Create Date Dimension
Create table DimDate
(
date_id int primary key identity,
date DATE,
month int,
quarter int,
year int
)
Go
```

iii) Creating Fact table

```
--Create Fact table
PRINT '';
PRINT '*** Creating Table FactCrime';
GO
Create Table FactCrime
(facID bigint primary key identity,
crime_id int not null,
location_id int not null,
type_id  int not null,
date_id int not null,
```

iv)Adding Relations

```
--Add Relations
PRINT '';
PRINT '*** Add relation between fact table foreign keys to Primary keys of Dimensions';
GO
ALTER TABLE FactCrime ADD CONSTRAINT
FK_CrimeID FOREIGN KEY (crime_id)REFERENCES DimCrime(crime_id);
PRINT '1';
ALTER TABLE FactCrime ADD CONSTRAINT
FK_LocationID FOREIGN KEY (location_id)REFERENCES DimLocation(location_id);
PRINT '2';
ALTER TABLE FactCrime ADD CONSTRAINT
FK_TypeID FOREIGN KEY (type_id)REFERENCES DimType(type_id);
PRINT '3';
ALTER TABLE FactCrime ADD CONSTRAINT
FK_DateID FOREIGN KEY (date_id)REFERENCES DimDate(date_id);
Go
```

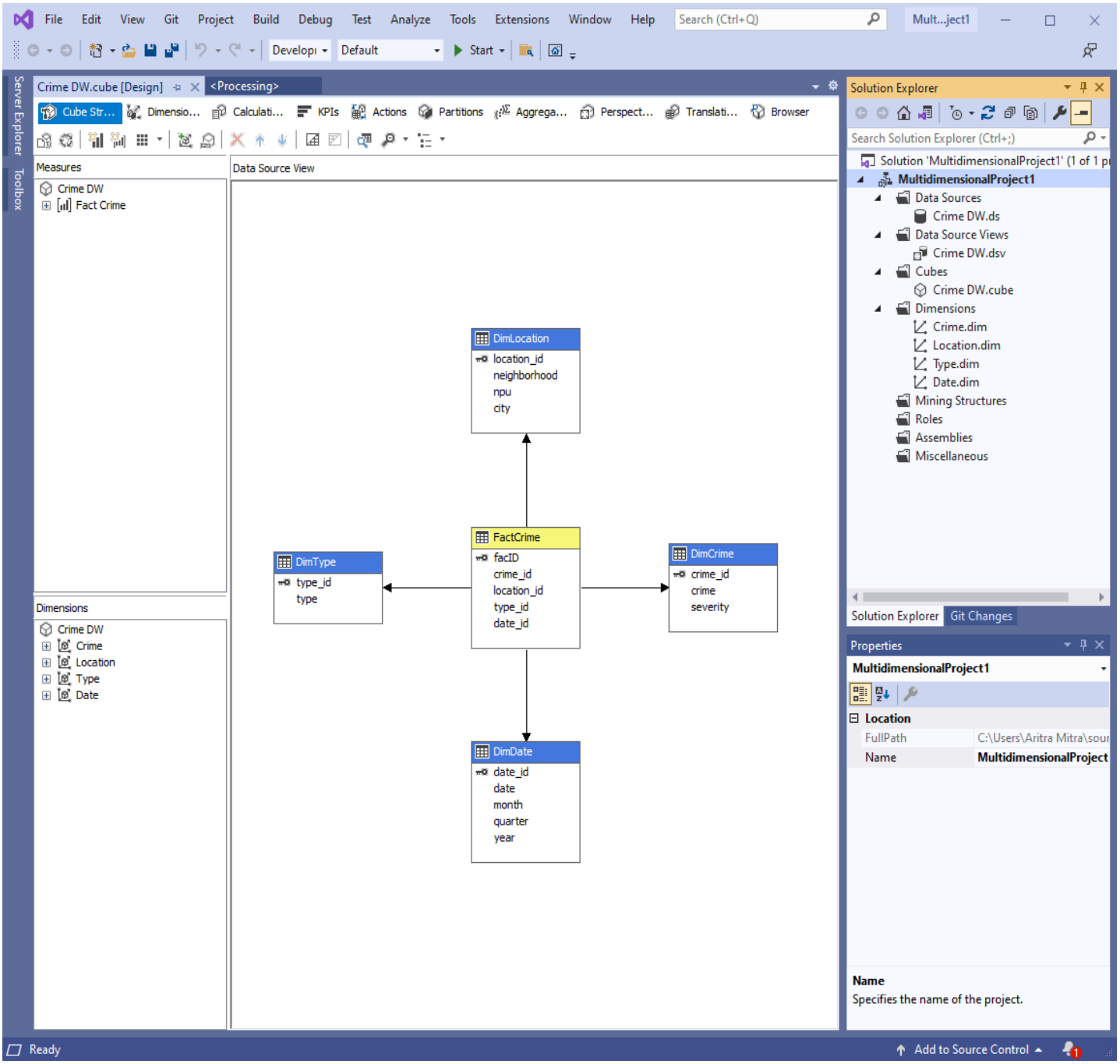
v)Populating the Dimension table and Fact table

```
BULK INSERT [dbo].[DimCrime] FROM 'C:\Users\Aritra Mitra\Documents\CITS3401\Project1\crimeDim.csv'
WITH (
    CHECK_CONSTRAINTS,
    --CODEPAGE='ACP',
    DATAFILETYPE='char',
    FIRSTROW=2,
    FIELDTERMINATOR=',',
    ROWTERMINATOR='0X0A',
    --KEEPIDENTITY,
    TABLOCK
);

BULK INSERT [dbo].[DimLocation] FROM 'C:\Users\Aritra Mitra\Documents\CITS3401\Project1\locationDim.csv'
WITH (
    CHECK_CONSTRAINTS,
    --CODEPAGE='ACP',
    DATAFILETYPE='char',
    FIRSTROW=2,
    FIELDTERMINATOR=',',
    ROWTERMINATOR='\n',
    --KEEPIDENTITY,
    TABLOCK
);

BULK INSERT [dbo].[DimType] FROM 'C:\Users\Aritra Mitra\Documents\CITS3401\Project1\TypeDim.csv'
WITH (
    CHECK_CONSTRAINTS,
    --CODEPAGE='ACP',
    DATAFILETYPE='char',
    FIRSTROW=2,
    FIELDTERMINATOR=',',
    ROWTERMINATOR='\n',
    --KEEPIDENTITY,
    TABLOCK
);
```

```
BULK INSERT [dbo].[DimDate] FROM 'C:\Users\Aritra Mitra\Documents\CITS3401\Project1\DateDim.csv'
WITH (
    CHECK_CONSTRAINTS,
    --CODEPAGE='ACP',
    DATAFILETYPE='char',
    FIRSTROW=2,
    FIELDTERMINATOR=',',
    ROWTERMINATOR='\n',
    --KEEPIDENTITY,
    TABLOCK
);
```

```
PRINT '4';
BULK INSERT [dbo].[FactCrime] FROM 'C:\Users\Aritra Mitra\Documents\CITS3401\Project1\factt.csv'
WITH (
    CHECK_CONSTRAINTS,
    --CODEPAGE='ACP',
    DATAFILETYPE='char',
    FIRSTROW=2,
    FIELDTERMINATOR=',',
    ROWTERMINATOR='\n',
    --KEEPIDENTITY,
    TABLOCK
);
PRINT '5';
```

Cube


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Crime DW.cube [Design]
<Processing>
Cube Str... Dimensio... Calculati... KPIs Actions Partitions Aggrega... Perspect... Translati... Browser
Measures
Crime DW
Fact Crime
Dimensions
Crime DW
Crime
Location
Type
Date
Data Source View
DimLocation
location_id
neighborhood
npu
city
FactCrime
facID
crime_id
location_id
type_id
date_id
DimType
type_id
type
DimCrime
crime_id
crime
severity
DimDate
date_id
date
month
quarter
year
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MultidimensionalProject1' (1 of 1 pr
MultidimensionalProject1
Data Sources
Crime DW.ds
Data Source Views
Crime DW.dsv
Cubes
Crime DW.cube
Dimensions
Crime.dim
Location.dim
Type.dim
Date.dim
Mining Structures
Roles
Assemblies
Miscellaneous
Solution Explorer
Git Changes
Properties
MultidimensionalProject1
Location
FullPath
C:\Users\Aritra Mitra\sour
Name
MultidimensionalProject
Name
Specifies the name of the project.
Ready
Add to Source Control

The cube allows for roll-up and drill-down analysis of the data.
For example, if we have crime data for a specific date we can roll up the data to the quarterly level to get a better understanding of which quarter of the year crimes are likely to happen most. This is achieved by moving up the hierarchy levels in the cube.

Drill-down analysis, on the other hand, involves analyzing data in greater detail. For example, if we have data on a city then we can drill down to the neighborhood to find the trends related to specific neighborhoods. This is achieved by moving down the hierarchy levels.

The dimensions and operations used:
For creating the cubes the following steps were taken:

- The cube was created using SSDT
- • Connect to a data source
- • Create the data source view
- • Build a cube
    - I used the Fact Crime fact table as Measure Group table
    - Then I ticked Fact Crime as measures
    - Rename each dimension name by removing the word Dim.

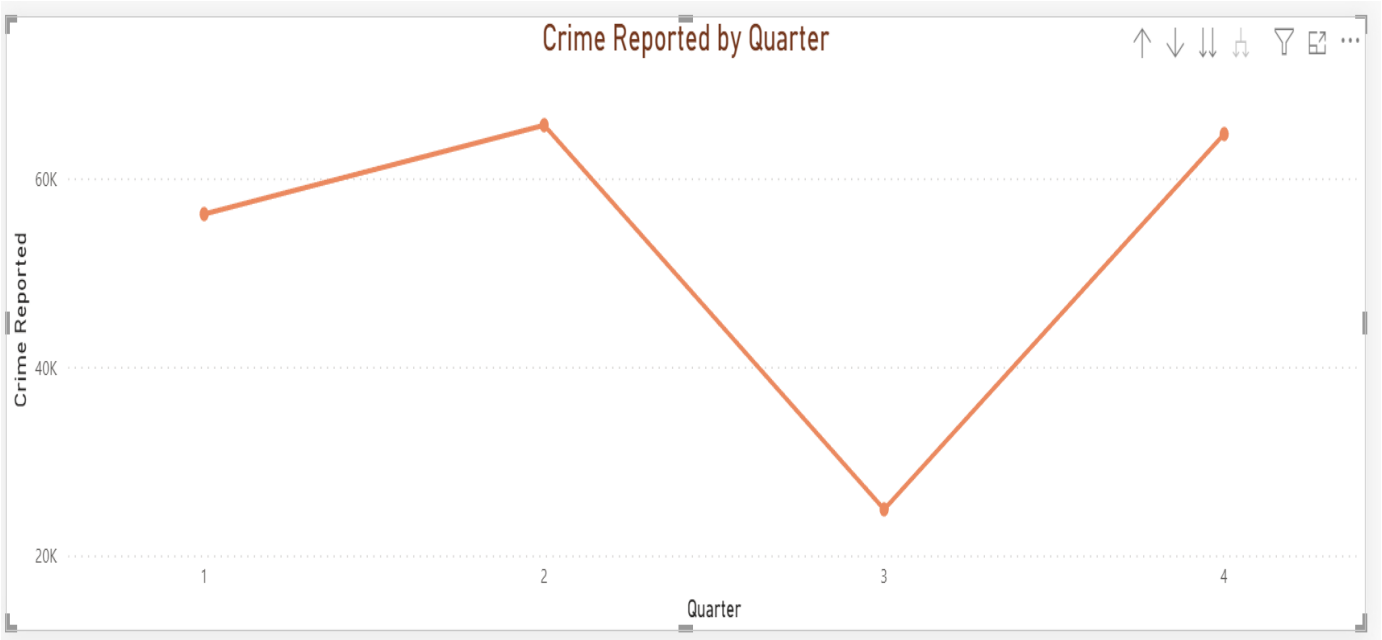
- • Generate and fine tune the dimension hierarchies(Using location dimension as example)
  -To build the concept Hierarchy i dragged attributes of the dimensions from the Data Source View pane to the Attribute pane in the dimension design window.
  - Arange them by order : City>NPU>Neighborhood.
  date>month>quarter>year
  crime>severity
  type

  -Clicked Process button
  -This extracts data from the original relational database.

PowerBI

Aim:The aim of this study is to analyze the trends of crime in America following the Great Recession in 2009 and to examine how the crime was reported during the tenure of the first President elected after the recession, from 2009 to 2016..
(Presidency of Barack Obama - Wikipedia, 2009)

Business query1:How many crimes were reported in each quarter during Obama's presidential tenure [2009-2016]?How does it compare year wise? Is there any significant trend?

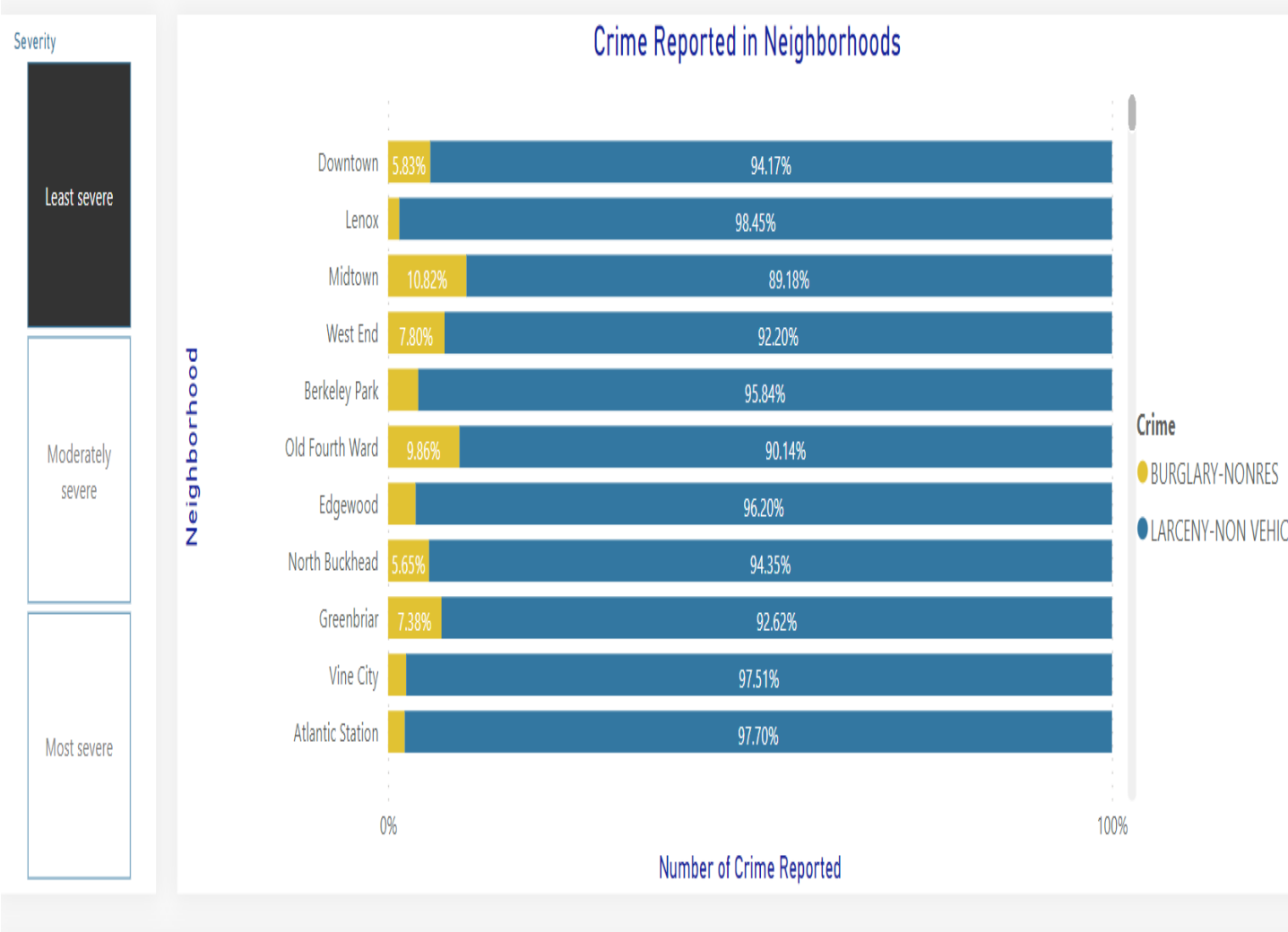


Over the years we can see that there is a sharp increase in quarter 2 then a a sharp decrease in quarter 3 before rising again in quarter 4. This graph can tell us the second quarter of the year has the highest number of crimes reported and quarter 3 has the lowest number of crimes reported.
According to the data, in quarter 1 of the year, there were 56,207 reported crimes, while in quarter 2, the number increased to 65,631. However, there was a sharp decrease in quarter 3 with only 24,892 reported crimes before rising again to 64,693 in quarter 4.
Furthermore, by rolling up and examining the data for each year, it can be seen that there has been a decrease in the number of reported crimes from 31,743 in 2009 to 22,669 in 2016. This trend is reflected in all quarters.

The query below was executed in SSMS to validate the results:

```
--check for question 1
SELECT YEAR(d.date) AS year, COUNT(f.crime_id) AS crime_count
FROM FactCrime f
JOIN DimDate d ON f.date_id = d.date_id
JOIN DimCrime c ON f.crime_id = c.crime_id

GROUP BY YEAR(d.date)
ORDER BY YEAR(d.date) ASC;
```

Business query2:What was the most commonly reported crime in the least severe category in each neighborhood?

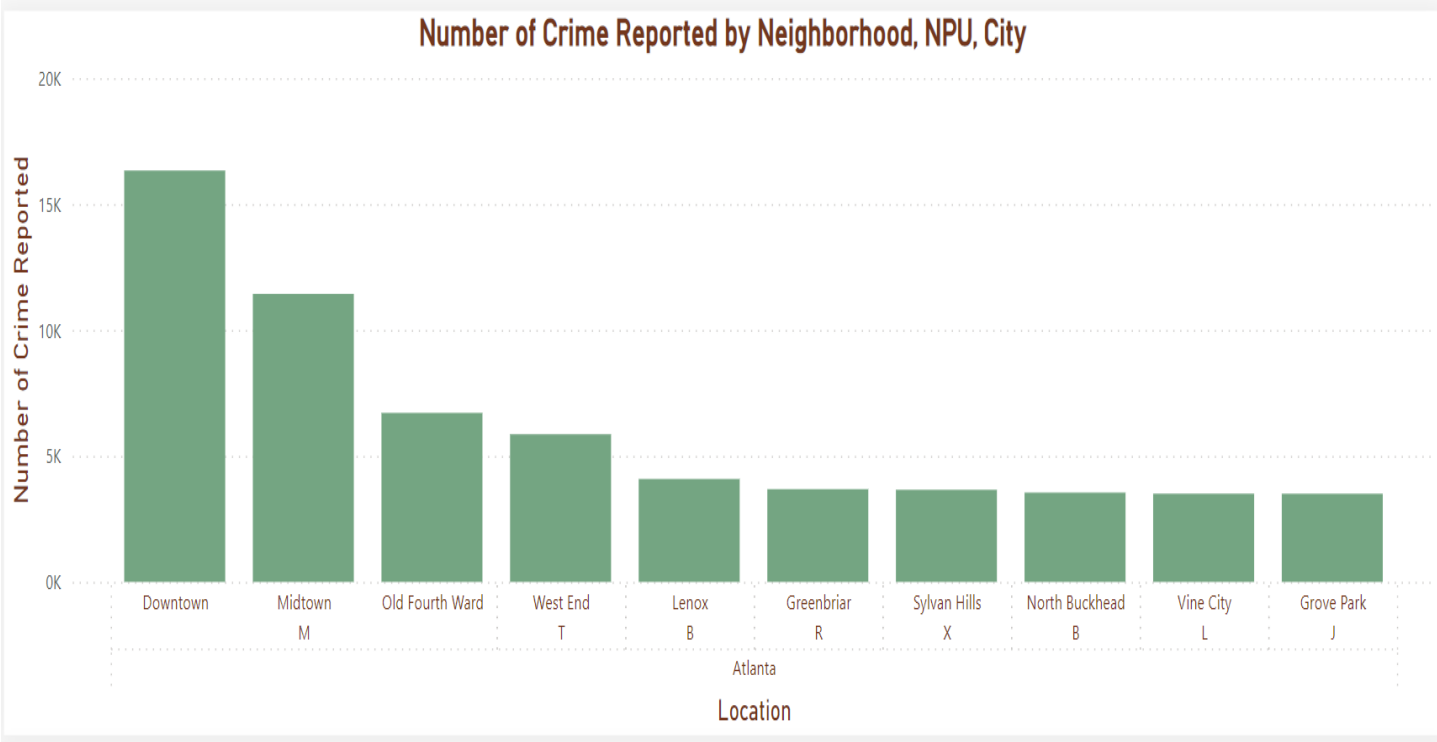


In the least severe category Larceny-non vehicle crime is most commonly reported. The percentages Larceny- non vehicle is reported is mostly above 95% with percentages sometimes falling in between 90% to 80% and 80% and 70%

The query below was executed in SSMS to validate the results:

```sql
WITH joinedTable AS
    (SELECT f.crime_id, l.neighborhood, c.severity
    FROM FactCrime f
    LEFT JOIN DimCrime c ON c.crime_id = f.crime_id
    LEFT JOIN DimLocation l ON l.location_id = f.location_id
    WHERE severity LIKE '%Least%'
    )
SELECT COUNT(crime_id) CrimeNum, neighborhood
FROM joinedTable
GROUP BY neighborhood
ORDER BY neighborhood
```

Business query3: Which neighborhood has highest number of crime reported?

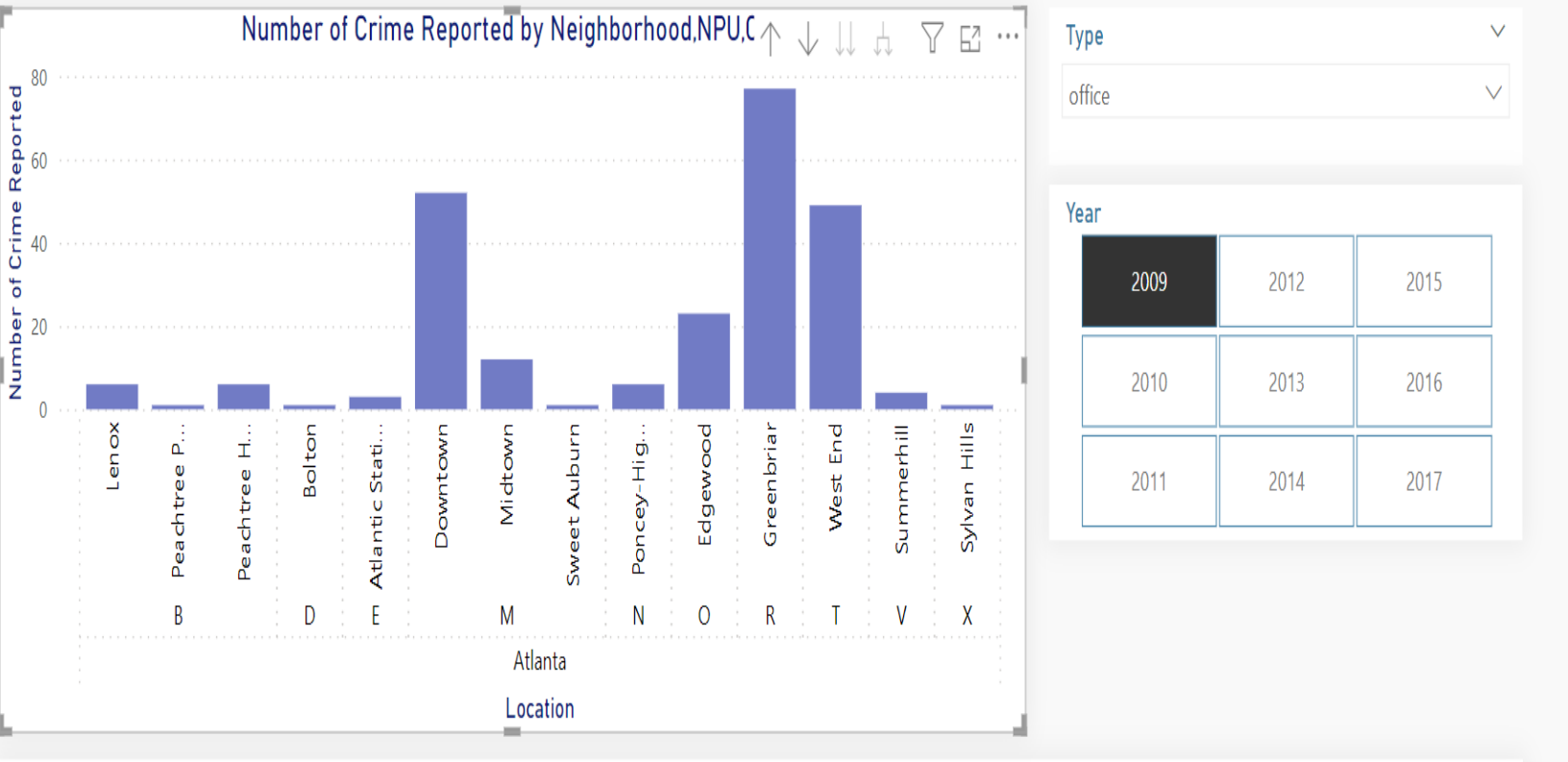


Downtown is the town with the highest number, with value reported as 16334 of crimes happening reported. Midtown being the second highest with value reported as 11435.

The query below was executed to validate the results in SSMS:

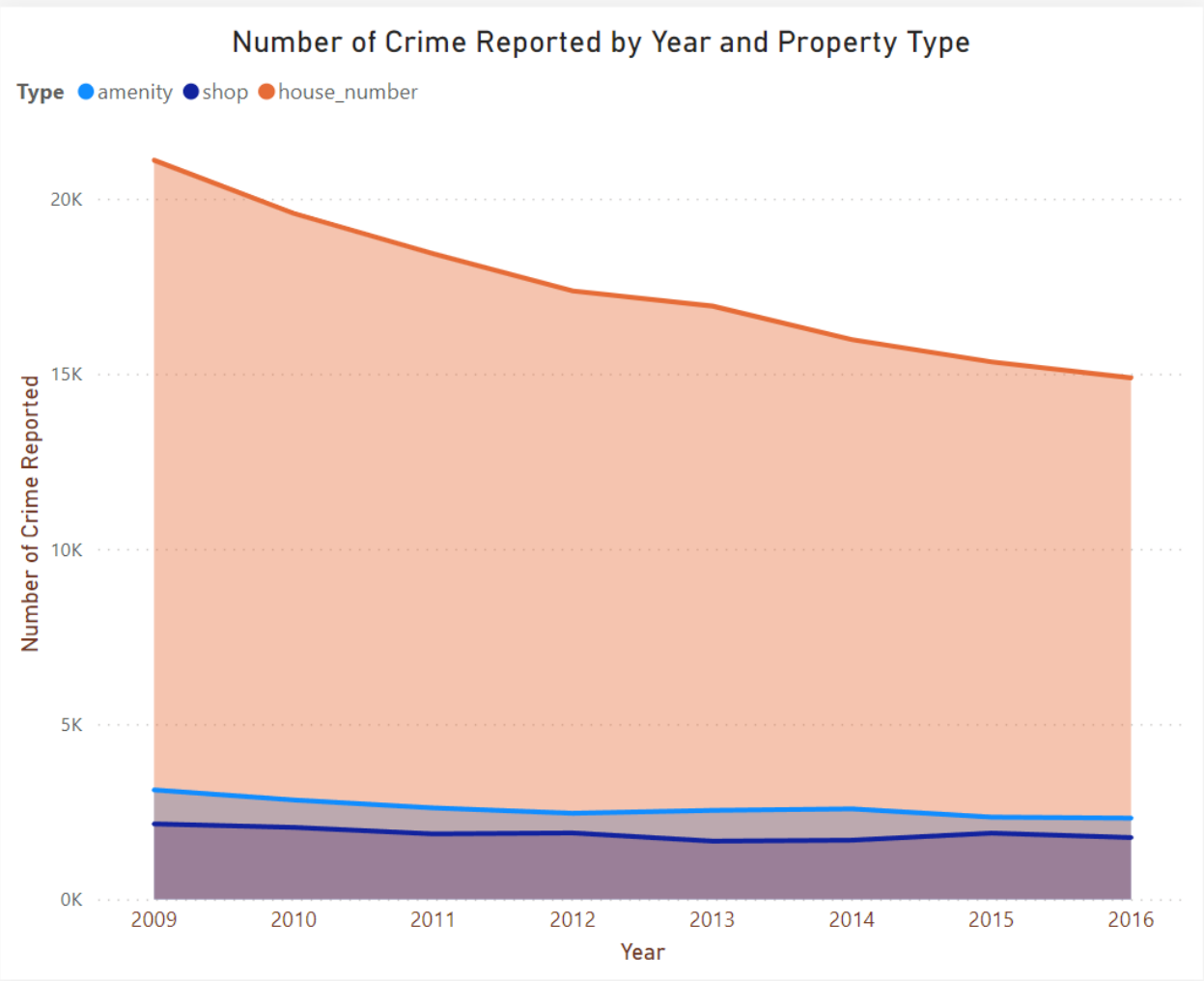
```
--check for question 3 Q. What is the neighborhood with highest number of crime?
SELECT n.neighborhood, COUNT(*) AS crime_count
FROM FactCrime f
JOIN DimLocation n ON f.location_id = n.location_id
GROUP BY n.neighborhood
ORDER BY COUNT(*) DESC;
```

Business query4:In 2009, which NPU had the highest number of reported crimes that occurred in offices? Furthermore, within that NPU, which neighborhood had the highest number of reported crimes that occurred in offices?



NPU R had the highest number of reported cases of 77 crimes in offices. By using drill down facilities with it can be seen that the neighborhood under NPU R with highest number of cases reported in office is Greenbriar.

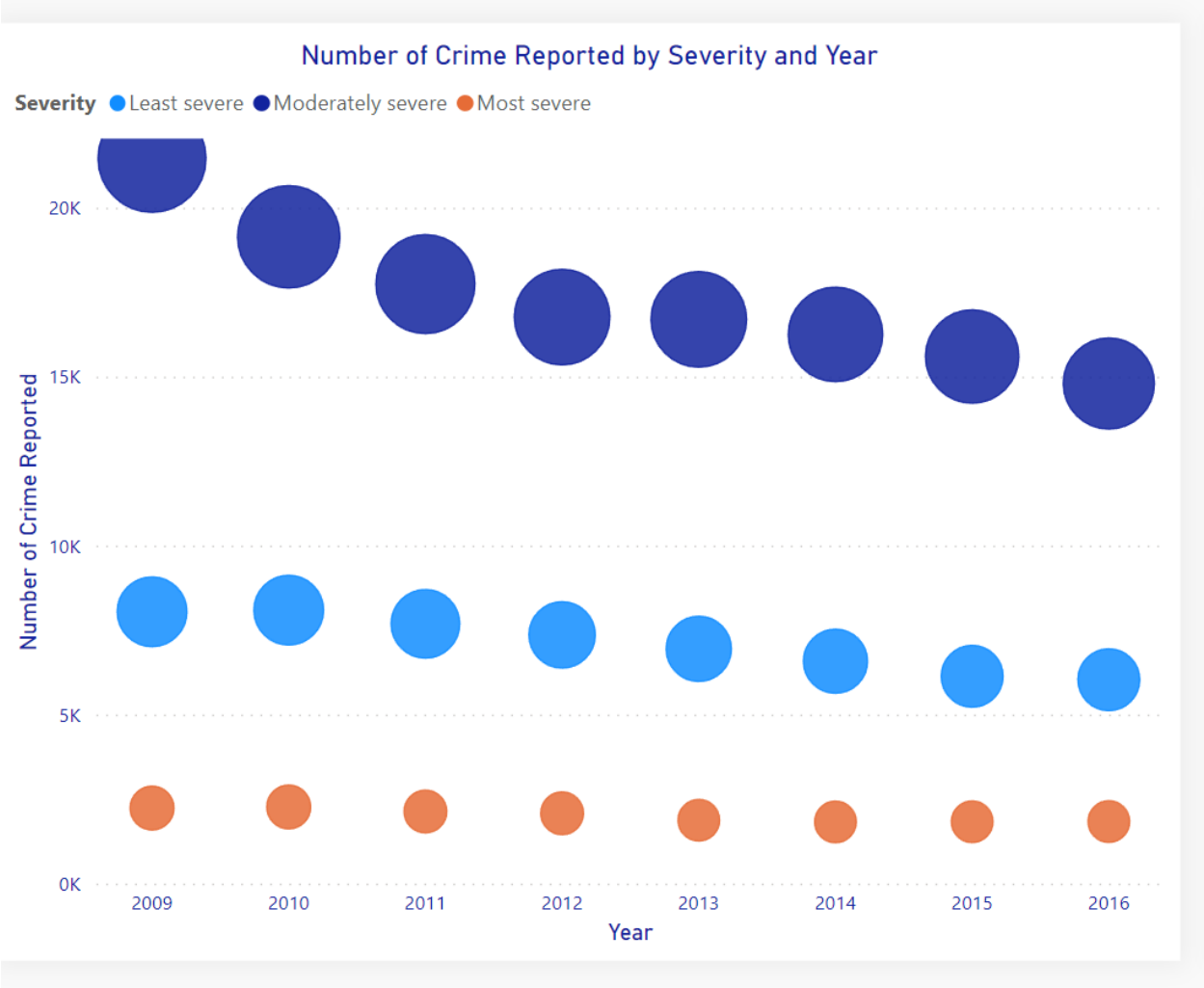
Business query5:Which top three property types had the highest number of crimes reported during president Obama's tenure[2009-2016]



House, shop and amenity are the top most targeted property types.

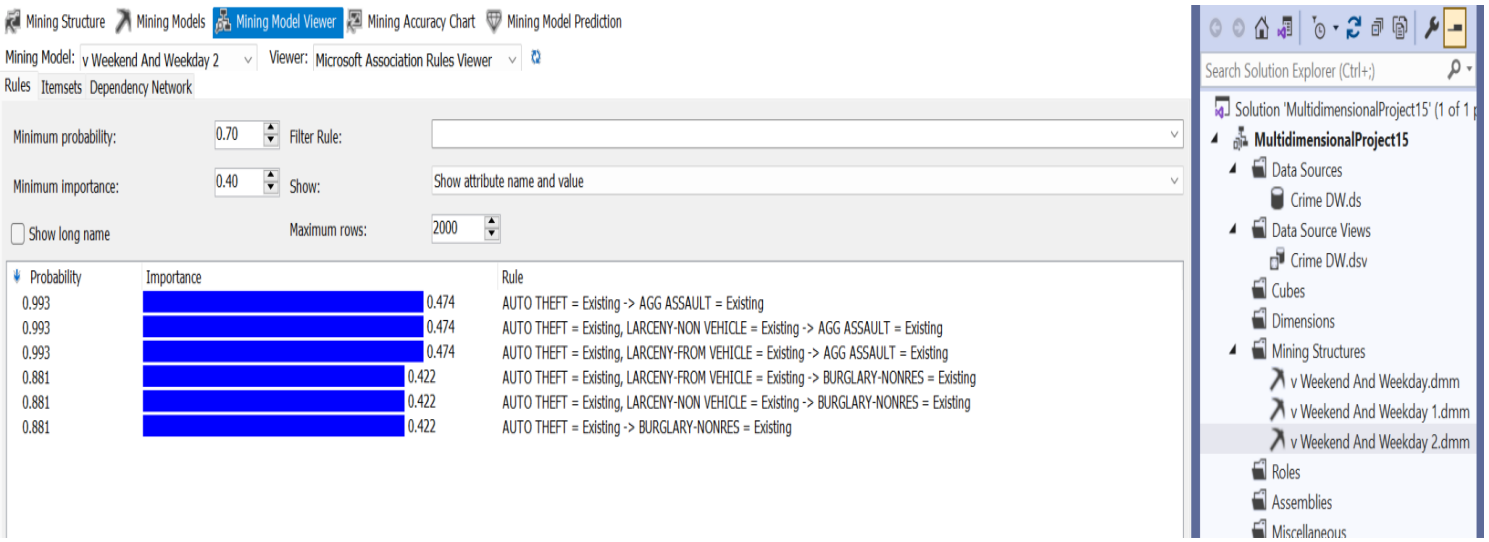
```sql
--Q.Which three property types had the highest number of crimes during presidenytObama's tenure[2009-2016]
SELECT d.year,t.type, COUNT(*) AS crime_count
FROM FactCrime f
JOIN DimType t ON f.type_id = t.type_id
JOIN DimDate d ON f.date_id = d.date_id
WHERE d.year >=2009  AND d.year <= 2016
GROUP BY t.type, d.year
ORDER BY COUNT(*) DESC
```

Business query6:What is the trend of severity levels of crime annually during Obama's presidential tenure[2009-2016]



Least severe trend - In terms of the least severe crimes, while there has been a consistent decrease overall, there was a spike in 2010 with 8,092 cases reported before dropping down to 6,033 in 2016.
Moderately severe trend - The crime that is most frequently reported has seen a steady decrease over the years, going from 21,464 cases to 14,800.
Most severe - The most severe crimes, on the other hand, are the least reported, and there was a sharp decline in reported cases between 2010-2011 and 2012-2013. The highest number of most severe crime is 2267.In recent years, the number of most severe crimes has been consistently low, with an average of about 1,800 cases compared to over 2,000 cases reported five years ago.

Association Rule Minning



Question Targeted: To find out the Crimes that are reported together on the same day.

A)Top 5 Rules

B)Top 5 Rules explanation in English

A)

Rule1 : On the same day that an AUTO THEFT is reported there is a probability of 0.993 of AGG ASSUALT happening on the same day.

Rule 2: On the same day that an AUTO THEFT and LARCENY-NON VEHICLE is reported there's a probability of 0.993 AGG ASSAULT happening on the same day

Rule 3: On a day that AUTO THEFT and LARCENY-FROM VEHICLE is reported there's a probability of 0.993 of AGG ASSAULT happening on the same day

Rule 4: On a day that AUTO THEFT and LARCENY-FROM VEHICLE is reported there's a probability of 0.442 of BURGLARY-NONRES happening on the same day.

Rule 5: On a day that AUTO THEFT and LARCENY-NON VEHICLE is reported there's a probability of 0.442 of BURGLARY-NONRES happening on the same day.

B)
Rule1:If an auto theft is reported on a certain day, it is very likely 99 percent that an aggravated assault incident will also occur on the same day (as an auto theft.)))

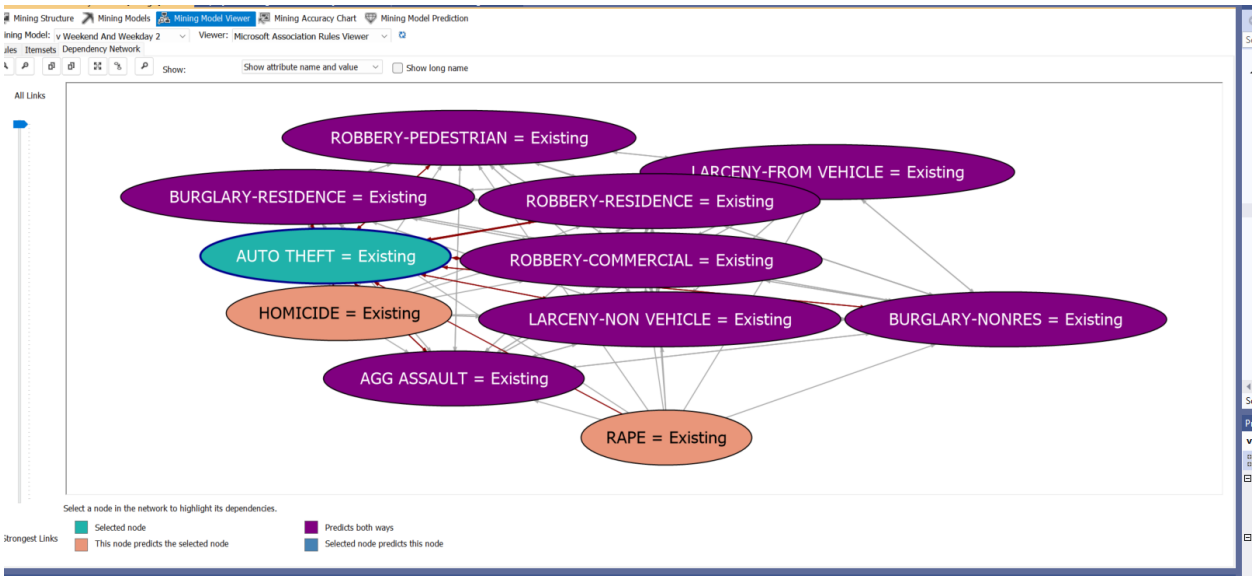
Rule 2: If an auto theft and a larceny-non-vehicle incident are reported on a certain day, it is very likely, about 99 percent, that an aggravated assault incident will occur on the same day.

Rule 3: If an auto theft and larceny-from-vehicle incident is reported on a certain day, it is very likely,about 99 percent that an aggravated assault will occur on the same day as an auto theft.

Rule 4:If an auto theft and larceny-from-vehicle incident is reported on a certain day, it is moderately likely,about 88 percent burgualry -non- res will occur on the same day.

Rule 5:If an auto theft and larceny-non-vehicle incident is reported on a certain day, it is moderately likely about 88 percent,a burgualry -non- res will occur on the same day.

The Dependency Network:



References

1.Neighborhood Planning Unit | Atlanta City Council, GA. (n.d.). Neighborhood Planning Unit | Atlanta City Council, GA. https://citycouncil.atlantaga.gov/other/npu-by-neighborhood/neighborhood-planning-unit

2.Presidency of Barack Obama - Wikipedia. (2009, June 15). Presidency of Barack Obama - Wikipedia. https://en.wikipedia.org/wiki/Presidency_of_Barack_Obama

3.https://chat.openai.com/

4..Starnet footprint:https://people.cmix.louisiana.edu/vijay/CMPS566/class_notes/data%20warehouse%20part2.pdf