Data Warehousing

SEMESTER 1

Project 2
Graph Database

Neha Sharma(23747726)
Rashi Aggarwal (23623739)

Design

Design for our graph database is mainly focusing on the relationship between the crimetype, the location type, neighborhood, beat and zone where crime has occurred , along with that we have a date on which the crime has occurred.

For the database we have 6 Nodes

- Crime - This node represents the specific crime incident. The properties for this crime are “crime_id” and “crime_type”.

- TypeOfLoc - This node represents the type of property that is involved in the crime. This node has property “type”

- Neighborhood - This node represents the neighborhood where the crime has occurred. This node has property “neighborhood”

- Date - This node represents the date when the crime has occurred. This node has properties “date”, “year” and “month”.

- Beat - This node represents which beat covers the crime. This node has the property “beat”.

- Zone - As we have 6 zones in our data, this node represents out of 6 zones, in which zone the crime has taken place. This node has the property “zone”.

We have 6 relationships

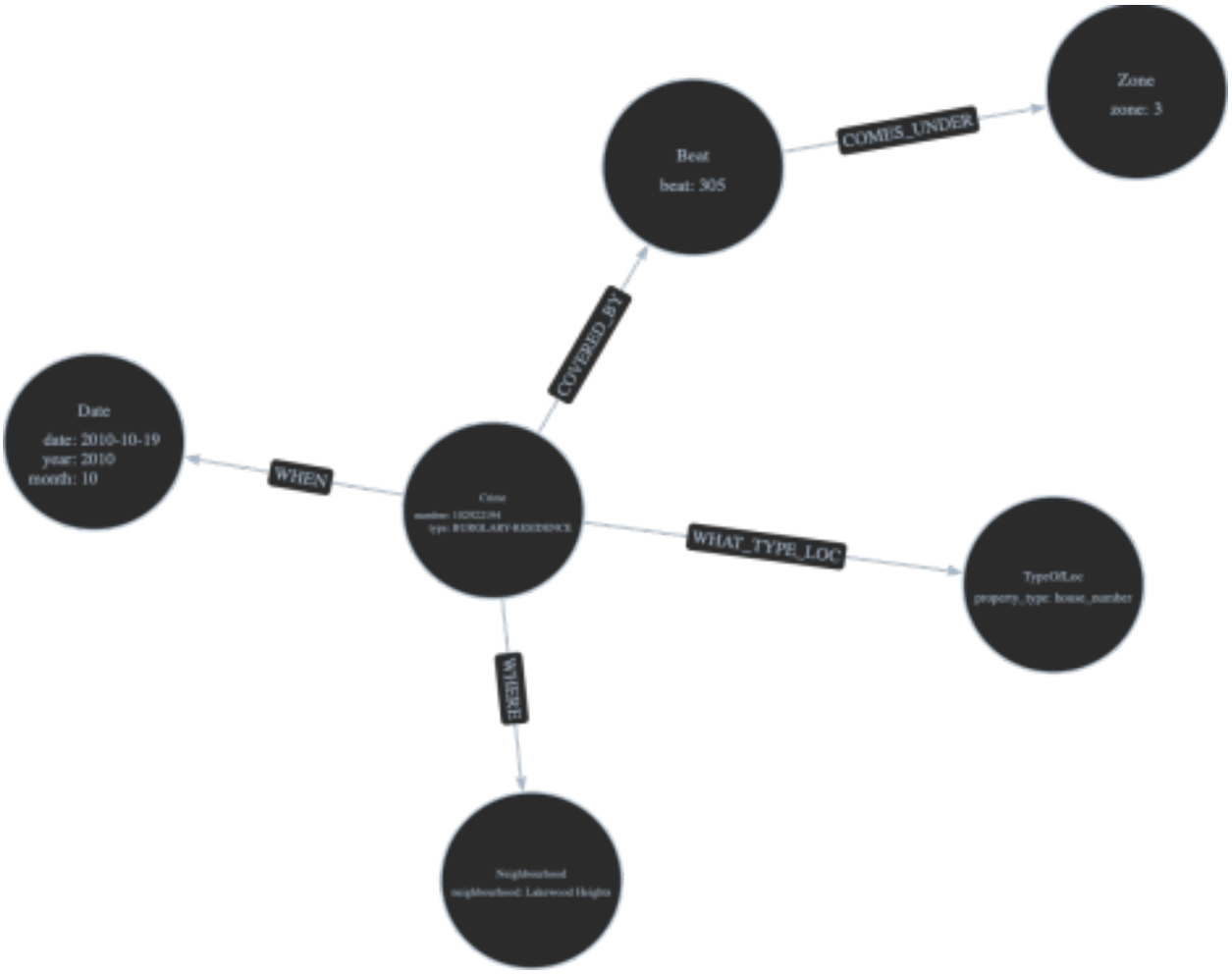
- WHERE - Links a Crime node to a Neighbourhood node to identify the area in which the crime took place.

- WHAT_TYPE_LOC - Indicates the kind of property that was involved in the crime by connecting a Crime node to a Property_type node.

- WHEN - Establishes a connection between a Crime node and a Date node, signifying the precise date of the crime occurred.

- COVERED_BY - The COVERED_BY attribute shows that a criminal incidence is covered by a certain crime beat by joining a crime node to a crime_beat node.

- COMES_UNDER - The crime beat hierarchy that falls under particular crime zones is represented by the COMES_UNDER property, which links a crime beat node to a crime zone node.
- ADJACENT_TO - Connects two Crime_zone nodes to signify that the zones are close to one another.



Implementation

For the implementation we have followed the several steps:

1. Selecting the entities: Choosing the crime data entities that should be represented in the graph database. The entities we selected are crimes, locationtype, neighborhood, date, beat and zone
2. Node labels, properties definition and Relationship: Giving suitable names to nodes and adding properties to each node. Based on the linkages and associations in the crime data, ascertain the links between the entities. Through numerous associations like WHERE, WHAT_TYPE_LOC, WHEN, COVERED_BY, and COMES_UNDER.

3. Cypher queries - using cypher queries to analyze the date for getting the answers for our business queries.

Loading the data to graph database

We have used the following cypher queries to create the node “Neighborhood”

```
LOAD CSV WITH HEADERS FROM 'file:///neighborhood_table.csv' AS row
CREATE (n:Neighborhood {
neighborhoodId: row.neighborhood_id,
neighborhood: row.neighborhood
})
```

We have used the following cypher queries to create the node “TypeOfLoc”

```
LOAD CSV WITH HEADERS FROM 'file:///type_loc_dim_table.csv' AS row
CREATE (l:TypeOfLoc {
typeID: row.type_id,
type: row.type
})
```

We have used the following cypher queries to create the node “Zone”

```
LOAD CSV WITH HEADERS FROM 'file:///zone_dim_table.csv' AS row
CREATE (z:Zone {
zoneID: row.zone_id,
zone: toInteger(row.zone)
})
```

We have used the following cypher queries to create the node “Beat”

```
LOAD CSV WITH HEADERS FROM 'file:///beat_dim_table.csv' AS row
CREATE (b:Beat {
beatID: row.beat_id,
beat: toInteger(row.beat)
})
```

We have used the following cypher queries to create the node “Date”

```
LOAD CSV WITH HEADERS FROM 'file:///date_dim_table.csv' AS row
CREATE (d:Date {
dateID: row.date_id,
date: datetime(row.date),
year: toInteger(row.year),
month: toInteger(row.month)
})
```

Creating the relation

```
LOAD CSV WITH HEADERS FROM 'file:///merged_df_100.csv' AS row
MATCH (l:TypeOfLoc {type: row.type})
```

```
MATCH (n:Neighborhood {neighborhood: row.neighborhood})
MATCH (d:Date {date: dateTime(row.date)})
MATCH (z:Zone {zone: toInteger(row.zone)})
MATCH (b:Beat {beat: toInteger(row.beat)})
MERGE (c:Crime {crime_id: row.number, crime_type: row.crime})
CREATE (c)-[:WHERE]->(n),
(c)-[:WHAT_TYPE_LOC]->(l),
(c)-[:WHEN]->(d),
(c)-[:COVERED_BY]->(b),
(b)-[:COMES_UNDER]->(z)
```

Printing data from 10 rows

```
MATCH p=()-->() RETURN p LIMIT 10
```

Capabilities of graph databases in comparison with relational database

Graph databases differ from their relational counterparts in a number of different ways,

1. Graph databases allow for the representation of intricate connections and relationships between entities and offer flexible data modeling capabilities.whereas the primary focus of relational databases is structured data with predefined schemas and joins are used to create relationships

2. Queries are performed quickly and efficiently by graph databases involving finding connections and patterns but in relational databases as joins and indexing are frequently used to query related data, therefore the performance becomes low and complexity rises.

3. Graph databases are more effective at handling complex relationships than relational

databases. As when we are working with highly connected data, relational databases may encounter difficulties with complex queries and scaling issues.

ETL Process for Graph database

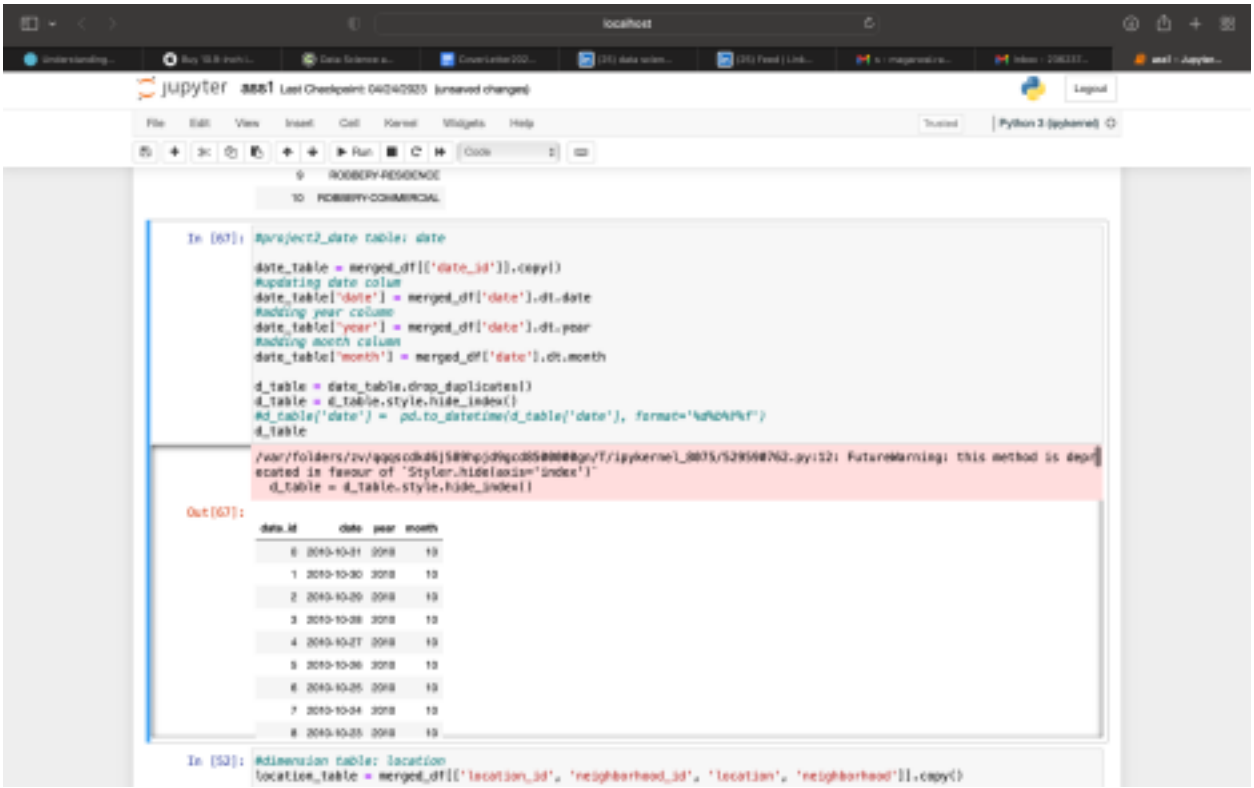
For the data cleaning process we already had dimension tables from Project1.
We just added the Neighborhood table and made a separate csv for that.

```
[]: #project2 location table
    loc_table = merged_df[['neighborhood_id', 'neighborhood']].copy()

    loc_tab = loc_table.drop_duplicates()
    loc_tab = loc_tab.style.hide_index()
    loc_tab

    #location dimension table
    excel_file_path = 'location_2_table.xlsx'
    loc_tab.to_excel(excel_file_path, index=False)
    excelFile = pd.read_excel(excel_file_path)
    # Converting excel file into CSV file
    excelFile.to_csv ("neighborhood_table.csv", index = None, header=True)
```

We also created the Date table with columns with date,year and month



We also modified the zone table for getting the adjacent zones in our query.

```
In [9]: #manually found the adjacent zones
adjacent_zones = [[2,5,6], [2], [1,4], [4], [6], [3,4]]
#adding a new column for the adjacent zones
zone_df_copy = zone_df.assign(new_column=adjacent_zones)
#removing unwanted indexing
zone_table = zone_df_copy.style.hide_index()
zone_table
```

```
/var/folders/zv/qqqscdkd6j5B9hpjd9gcd8500000gn/T/ipykernel_36793/2476385976.py:6: FutureWarning: this method is deprecated in favour of `Styler.hide(axis='index')`
  zone_table = zone_df_copy.style.hide_index()
```

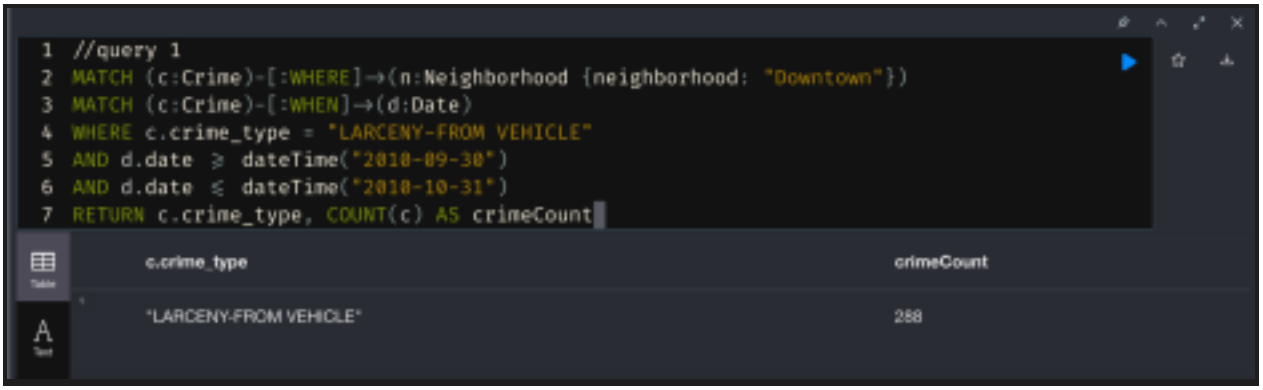
Out[9]:

zone_id	zone	new_column
0	4	[2, 5, 6]
1	3	[2]
2	6	[1, 4]
3	5	[4]
4	1	[6]
5	2	[3, 4]

Queries

1. **How many crimes are recorded for a given crime type in a specified neighborhood for a particular period?**

```
MATCH (c:Crime)-[:WHERE]->(n:Neighborhood {neighborhood: "Downtown"})
MATCH (c:Crime)-[:WHEN]->(d:Date)
WHERE c.crime_type = "LARCENY-FROM VEHICLE"
AND d.date >= dateTime("2010-09-30")
AND d.date <= dateTime("2010-10-31")
RETURN c.crime_type, COUNT(c) AS crimeCount
```



We can see that for the neighborhood “Downtown” and crime type “LARENCY_FROM VEHICLE” we have 288 crimes.

```
1  //query 1
2  MATCH (c:Crime)-[:WHERE]→(n:Neighborhood {neighborhood: "Inman Park"})
3  MATCH (c:Crime)-[:WHEN]→(d:Date)
4  WHERE c.crime_type = "AGG ASSAULT"
5  AND d.date ⩾ dateTime("2010-09-30")
6  AND d.date ⩽ dateTime("2010-10-31")
7  RETURN c.crime_type, COUNT(c) AS crimeCount
```

	c.crime_type	crimeCount
1	"AGG ASSAULT"	4

We can see that for the neighborhood “Inman Park” and crime type “AGG ASSUALT” we have 288 crimes.

2. **Find the neighborhoods that share the same crime types, organize in descending order of the number of common crime types.**

```
MATCH (c1:Crime)-[:WHERE]->(n1:Neighborhood),
(c1:Crime)-[:WHAT_TYPE_LOC]->(l:TypeOfLoc)<-[:WHAT_TYPE_LOC]-(c2:Crime)-[:WHERE]-
>(n2:Neighborhood)
WHERE n1 <> n2
WITH n1, n2, COUNT(DISTINCT l.type) AS commonCrimeTypes
ORDER BY commonCrimeTypes DESC
RETURN n1.neighborhood AS neighborhood1, n2.neighborhood AS neighborhood2,
commonCrimeTypes
```

Query With LIMIT 7

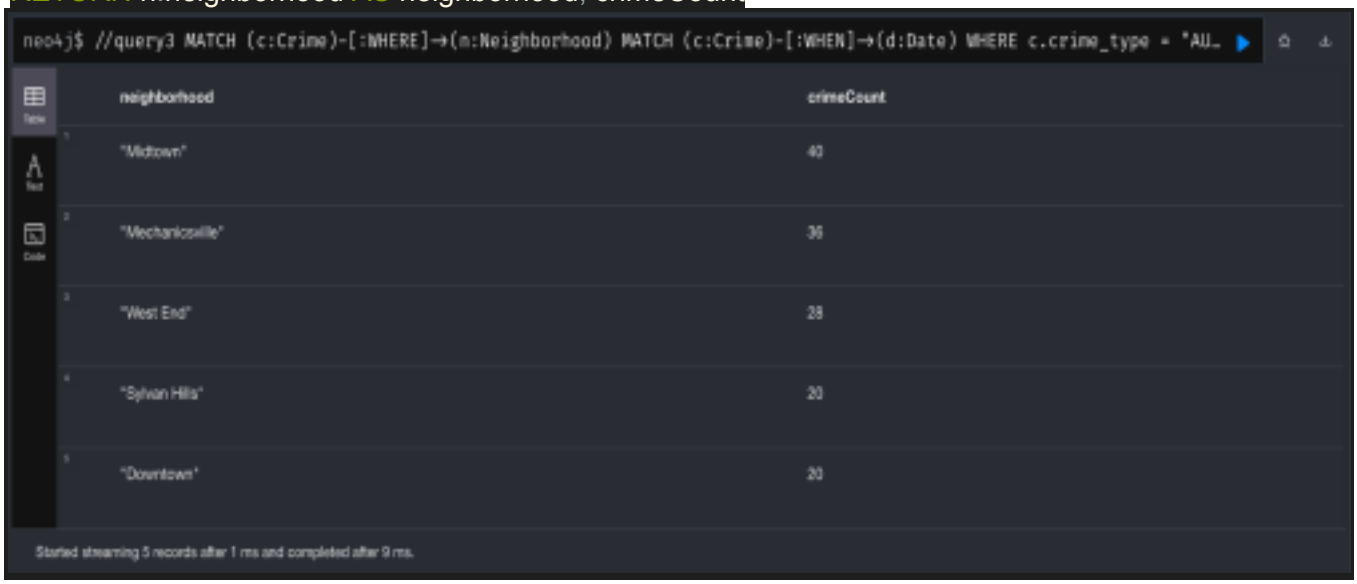
```
1  //query2
2  MATCH (c1:Crime)-[:WHERE]→(n1:Neighborhood), (c1:Crime)-[:WHAT_TYPE_LOC]→(l:TypeOfLoc)←[:WHAT_TYPE_LOC]-
   (c2:Crime)-[:WHERE]→(n2:Neighborhood)
3  WHERE n1 <> n2
4  WITH n1, n2, COUNT(DISTINCT l.type) AS commonCrimeTypes
5  ORDER BY commonCrimeTypes DESC
6  RETURN n1.neighborhood AS neighborhood1, n2.neighborhood AS neighborhood2, commonCrimeTypes
7  LIMIT 7
```

	neighborhood1	neighborhood2	commonCrimeTypes
1	"Downtown"	"Midtown"	6
2	"Midtown"	"Downtown"	6
3	"Midtown"	"West End"	5
4	"West End"	"Downtown"	5
5	"West End"	"Midtown"	6
6	"Downtown"	"West End"	6

Started streaming 7 records after 1 ms and completed after 2481 ms.

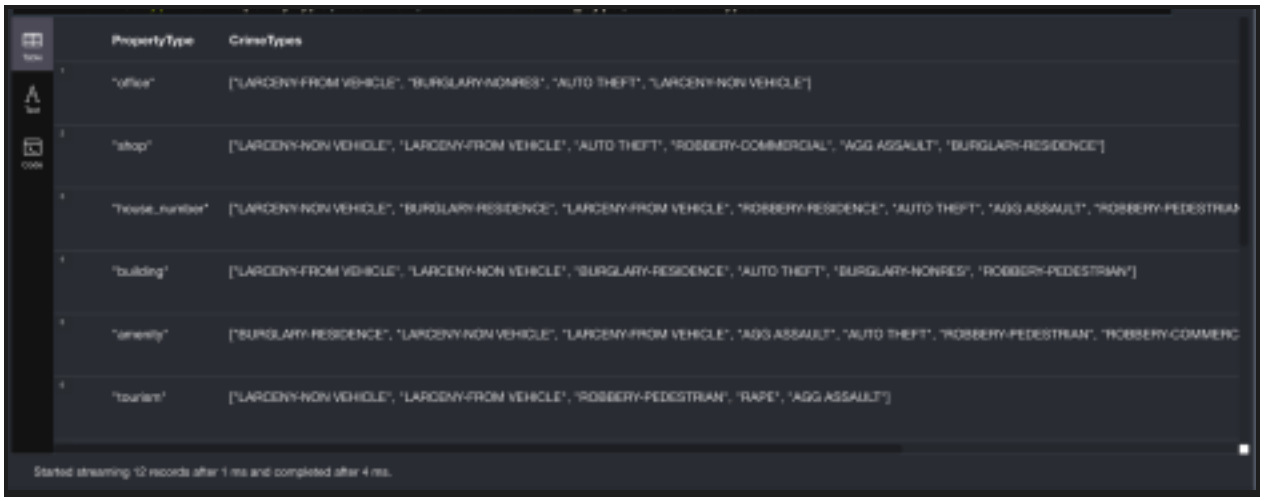
3. Return the top 5 neighborhoods for a specified crime for a specified duration.

```
MATCH (c:Crime)-[:WHERE]->(n:Neighborhood)
MATCH (c:Crime)-[:WHEN]->(d:Date)
WHERE c.crime_type = "AUTO THEFT" AND d.date >= dateTime("2010-09-30")
AND d.date <= dateTime("2010-11-01")
WITH n, COUNT(c) AS crimeCount
ORDER BY crimeCount DESC
LIMIT 5
RETURN n.neighborhood AS neighborhood, crimeCount
```



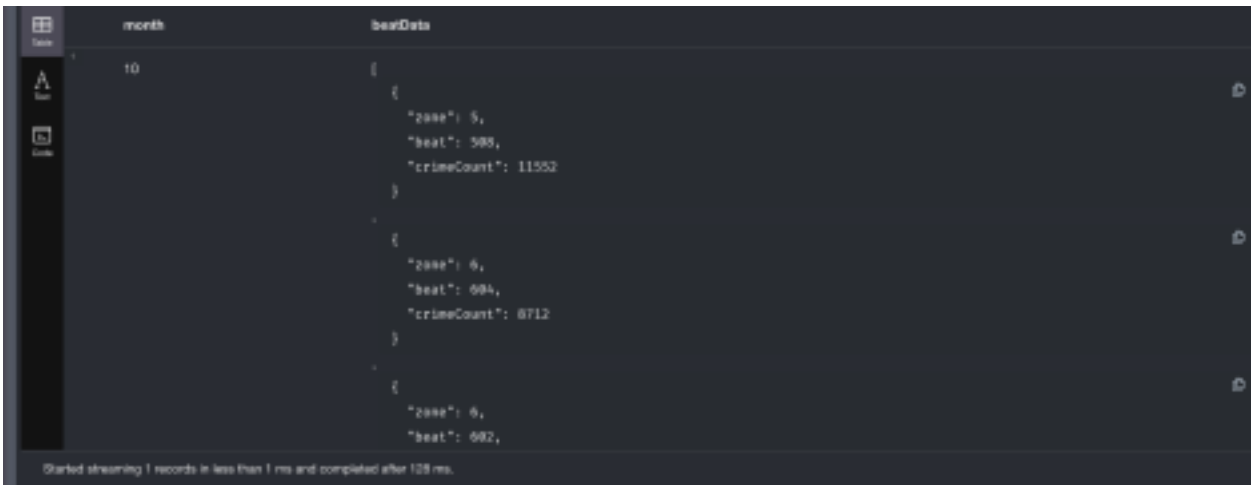
4. Find the types of crimes for each property type.

```
MATCH (l:TypeOfLoc)<-[r:WHAT_TYPE_LOC]-(c:Crime)
RETURN l.type AS PropertyType, COLLECT(DISTINCT c.crime_type) AS CrimeTypes
```



5. Which month of a specified year has the highest crime rate? Return one record each for each beat. Also need to return one record for each zone.

```
//query5
MATCH (c:Crime)-[:WHEN]->(d:Date), (c:Crime)-[:COVERED_BY]->(b:Beat),
(b:Beat)-[:COMES_UNDER]->(z:Zone)
WHERE d.year = 2010
WITH d.month AS month, b.beat AS beat, z.zone AS zone, COUNT(*) AS crimeCount
ORDER BY month, crimeCount DESC
RETURN month, COLLECT({zone: zone, beat: beat, crimeCount: crimeCount}) AS beatData
```



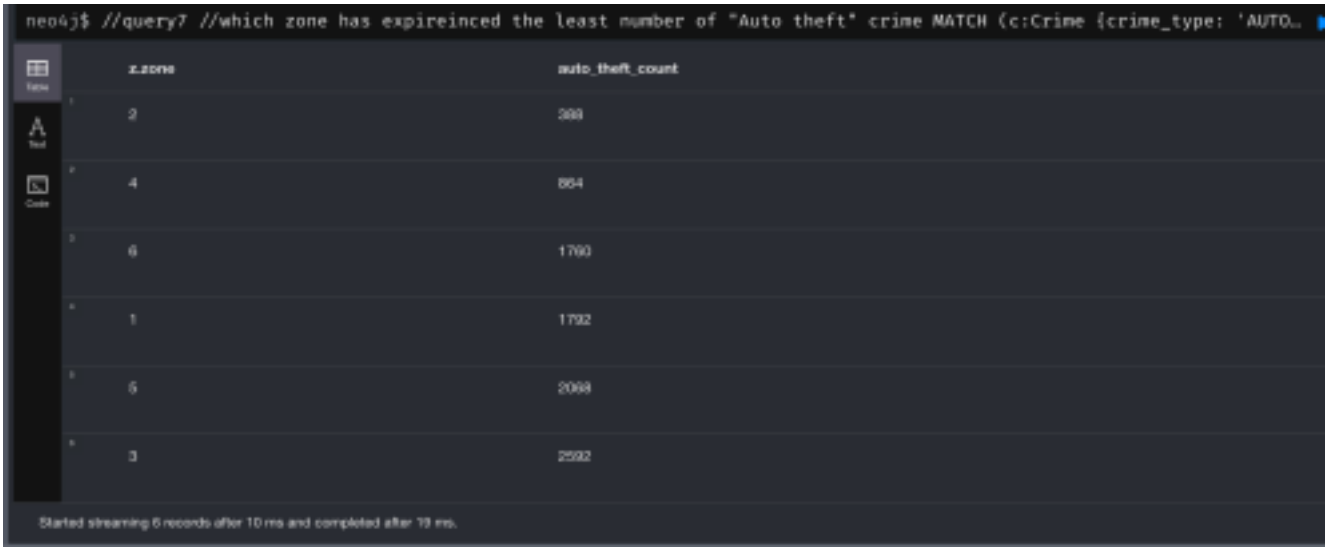
6. Find the zones that are adjacent and share the same high crime months.

```
//query6
MATCH (c:Crime)-[:WHEN]->(d:Date)
MATCH (c:Crime)-[:COVERED_BY]->(b:Beat)-[:COMES_UNDER]->(z:Zone)
// WHERE z1 <> z2 AND z1.zone_adjacent = z2.zone_adjacent
RETURN z.zone_adjacent, COLLECT(DISTINCT z.zone)
```

We tried getting the adjacent zone but didn't get the expected output.

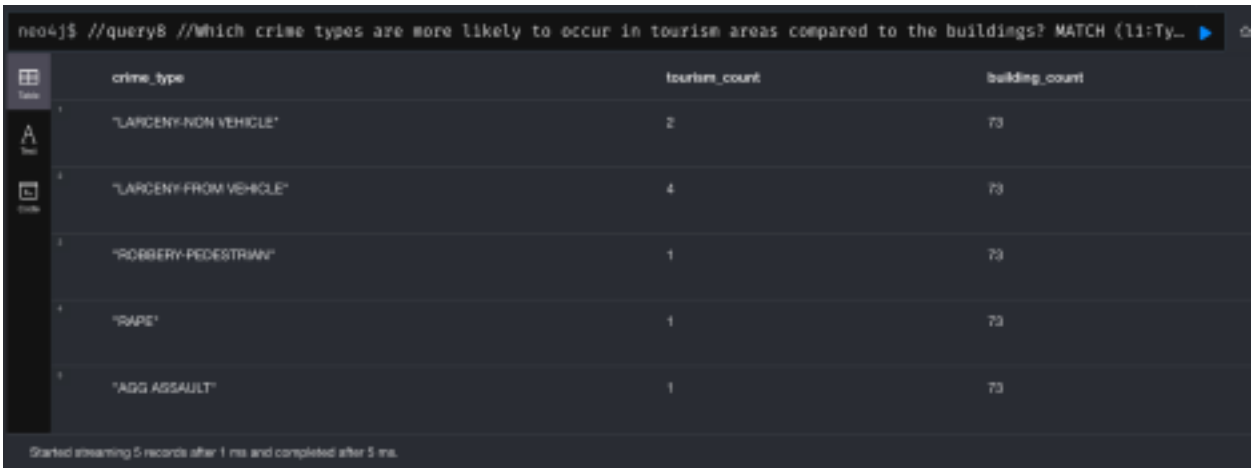
7. Which zone has experienced the least number of “Auto theft” crimes?

```
MATCH (c:Crime {crime_type: 'AUTO
THEFT'})-[:COVERED_BY]->(b:Beat)-[:COMES_UNDER]->(z:Zone)
RETURN z.zone, COUNT(c) AS auto_theft_count
ORDER BY auto_theft_count ASC
```



8. **What crime types are more likely to occur in “tourism” areas compared to the “buildings”?**

```
MATCH (l1:TypeOfLoc {type: 'tourism'})<-[:WHAT_TYPE_LOC]-(c1:Crime) WITH
DISTINCT c1.crime_type AS crime_type, COUNT(DISTINCT c1) AS tourism_count
MATCH (l2:TypeOfLoc {type: 'building'})<-[:WHAT_TYPE_LOC]-(c2:Crime) WITH
crime_type, tourism_count, COUNT(DISTINCT c2) AS building_count RETURN
crime_type, tourism_count, building_count
```



Graph Data Science can be applied in at least one useful application of this graph database

1. Crime Pattern Analysis: We can spot patterns and trends in criminal activity by looking at the connections between crime nodes, date nodes, neighborhood nodes, and other pertinent entities. It is possible to use graph techniques like community discovery, centrality analysis, and clustering to find hidden patterns and crime hotspots in the area.

2. 2. Analysis of criminal links: Graph databases are excellent at displaying intricate linkages. We can discover linkages and ties between various sorts of crimes by looking at the relationships between crimes. This knowledge can be useful for deciphering the methods used by criminals and finding connections between incidents that at first glance appear unconnected.

3. Predictive modeling: To anticipate future instances of crime, graph-based predictive modeling might be used. We can create prediction models using machine learning algorithms by examining previous crime data together with elements like time, location, and other contextual information. These models can aid law enforcement organisations in successfully allocating resources and implementing preventative actions to stop crimes in the community.

YouTube Video Link:

https://youtu.be/ryvfnMwrhog