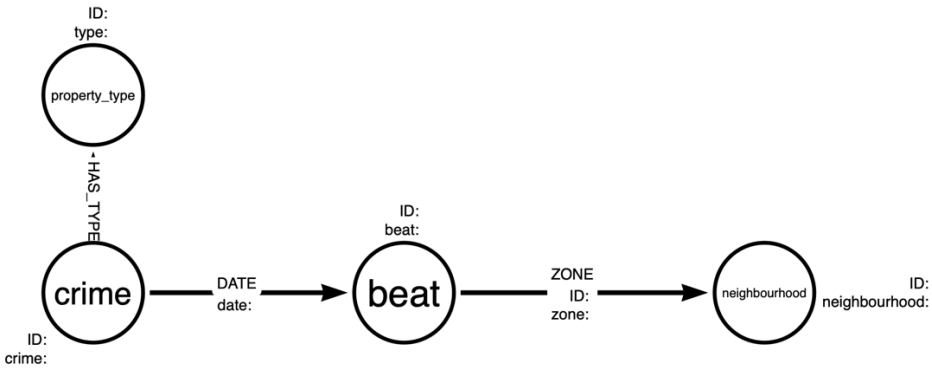
Author: Rongchuan Sun 23715251

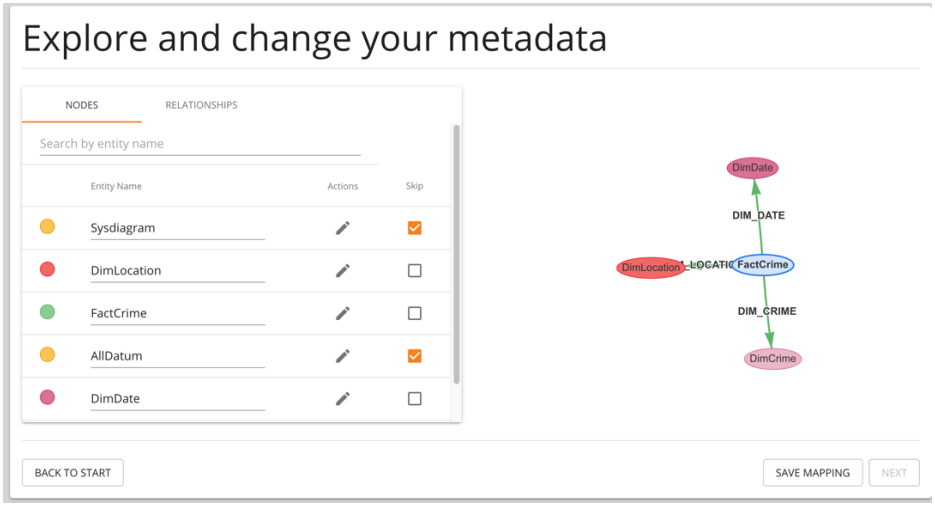
Ziqi Wang 236650644

This link is video link. https://youtu.be/XDSjIqzrRO4

I. Graph Database Design using the Arrow tool in comparison with direct ETL import.

1. The following two screenshots are graph database design by arrow tool and ETL import respectively.





2. According to the screenshot from arrow tool, we can see that there are five nodes and four relationships. Also, the property of beat, date and location is zone, year, month and neighbourhood respectively.
   According to the screenshot from ETL import, we can see that there are four nodes and three relationships. These four nodes are from fact table and dimension tables in SSMS.

II. Describes the design and implementation.

1. Design

Our design and implementation for the Neo4j project involved several steps, starting from importing the data files to creating the nodes and establishing relationships in the graph database.

The first step was to import the data files into Neo4j using the CSV import functionality. We had separate CSV files for each node: crimetype, date, neighbourhood, beat, propertytype, and zone. These CSV files contained the relevant information for each node, such as crime types, dates, neighbourhood names, beat IDs, property types, and zone IDs.

Using the LOAD CSV command in Neo4j's Cypher query language, we loaded the data from the CSV files and created the corresponding nodes

Once the nodes were created, we moved on to establishing the relationships between them. We had a separate CSV file, named relationship.csv, which contained the necessary information to establish the relationships. The relationship.csv file included identifiers or names for the crime type, date, neighbourhood, beat, property type, and zone associated with each crime incident.

To create the relationships, we used the MATCH clause in combination with the identifiers or names to find the corresponding nodes, and then used the CREATE clause to establish the relationships.

Throughout the process, we ensured that the node labels, properties, and relationship types were defined accurately to maintain consistency in the graph database.

By following this step-by-step process, we successfully imported the data, created the nodes for crimetype, date, neighbourhood, beat, propertytype, and zone, and established the relationships between them. This allowed us to build a comprehensive graph database representing the crime data in Atlanta.

With the nodes and relationships in place, we were able to leverage the power of graph data science techniques to analyze the crime data. We could perform queries, traversals, and graph algorithms to gain insights into crime patterns, spatial correlations, temporal trends, and more. The design and implementation of our Neo4j project provided a solid foundation for exploring and understanding crime in Atlanta using graph data science methodologies.

2. Pros and cons:

1) The design choices made for the Neo4j project involve nodes, relationships, data integration, and visualization. These choices have both advantages and disadvantages that impact the overall effectiveness and efficiency of the project.
2) Regarding nodes, the decision was made to include six nodes: crimetype, date, neighbourhood, beat, propertytype, and zone. This choice offers several benefits. Firstly, it allows for clear separation and organization of different entity types, ensuring better data management. Secondly, it enables focused analysis and querying on specific aspects of crime data. Lastly, it provides a structured representation of the attributes associated with each node. However, maintaining and updating these nodes when new data is added or modified requires additional effort, and it may introduce complexity into the queries and data modeling.
3) The establishment of relationships between nodes is crucial for capturing connections and interactions within the crime data. Relationships such as OCCURRED_ON_DATE, OCCURRED_IN_NEIGHBOURHOOD, OCCURRED_IN_BEAT, OCCURRED_IN_ZONE, and HAS_PROPERTY_TYPE were defined. This choice brings several advantages. It enables the exploration of various crime-related connections and associations, supporting traversals and graph algorithms to uncover patterns and insights. Additionally, it provides a flexible framework to capture dynamic relationships. However, careful planning and design are

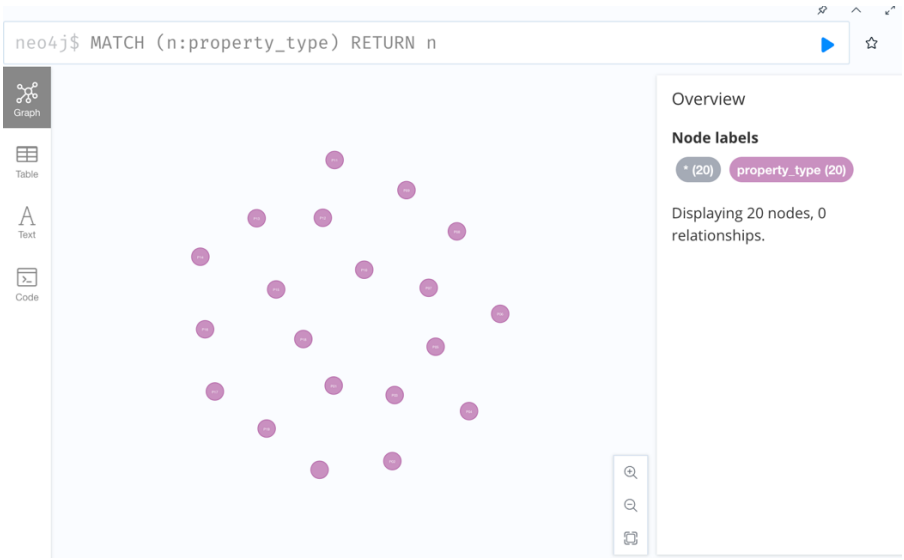
required to ensure accurate and meaningful relationships, and the complexity of query construction and optimization may increase.

4) Data integration is a crucial step in building a comprehensive crime graph database. In this design, data from various sources were imported through CSV files for each node, and relationships were established based on identifiers or names. This approach offers several benefits. It allows for easy integration of data from diverse sources into a unified graph structure. It provides flexibility in handling different data formats and schemas, and it enables the combination of structured and unstructured data for a holistic analysis. However, it requires data preprocessing and transformation to ensure consistency and accuracy, and challenges may arise when handling large volumes of data during the import process.
5) Visualization plays a vital role in understanding and communicating the crime data in the graph database. By utilizing Neo4j's built-in visualization tools or third-party libraries, informative visual representations can be created. Visualization brings several advantages to the project. It enables intuitive exploration and understanding of complex crime patterns and relationships. It facilitates the communication of insights to stakeholders through interactive and visually appealing visualizations. Additionally, it supports the identification of trends, clusters, and outliers. However, appropriate data aggregation and filtering are required for effective visual representation, and complex visualizations may impact performance and scalability.
6) In conclusion, the design choices made for the Neo4j project encompass nodes, relationships, data integration, and visualization. Despite certain challenges and trade-offs associated with each choice, the benefits outweigh the drawbacks. By carefully considering the pros and cons, a well-structured crime graph database has been created, facilitating comprehensive analysis, efficient querying, and meaningful visualization of crime data in Atlanta**.**

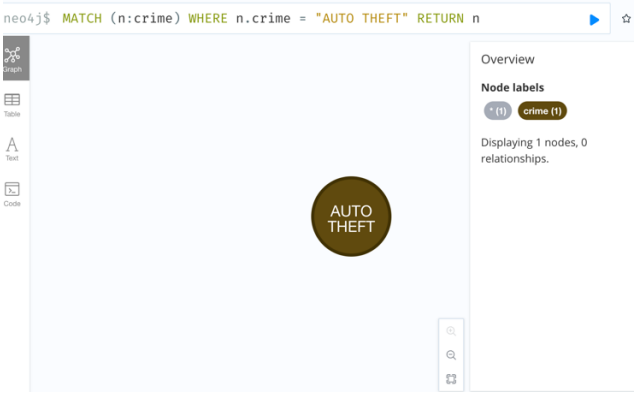
III. Meaningful Graph Database navigation discussed using the Neo4j browser.

1. Find all nodes of type crime
   From the picture below, we can see that there are 11 crimes in total.

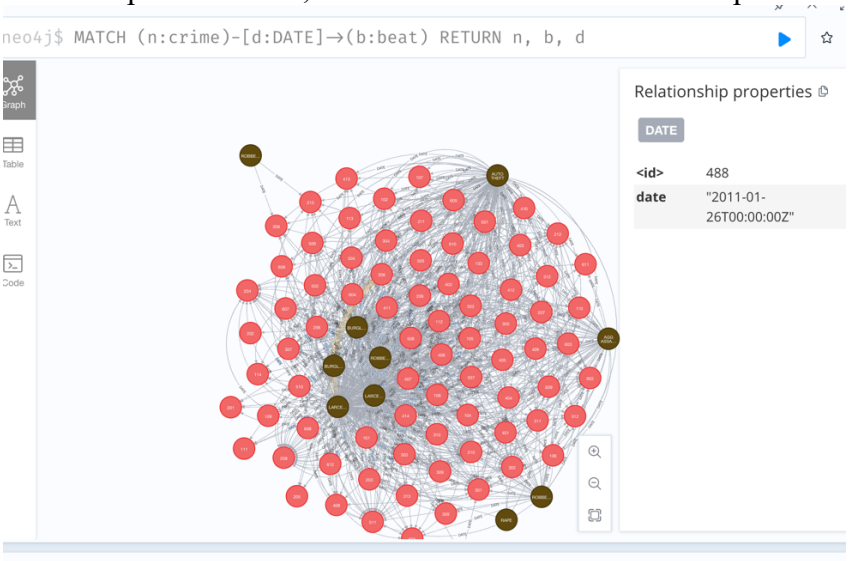
2. Find all nodes of type property_type
   From the picture below, we can see that there are 20 property-type in total.



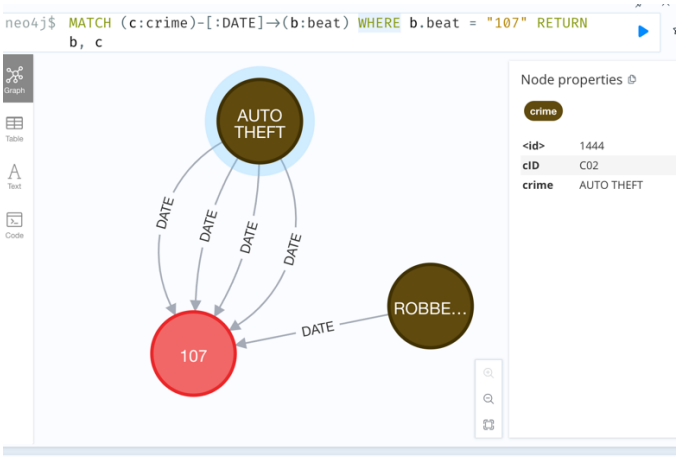
3. Find nodes with a special property value
   This query is to return the node of “AUTO THEFT”.



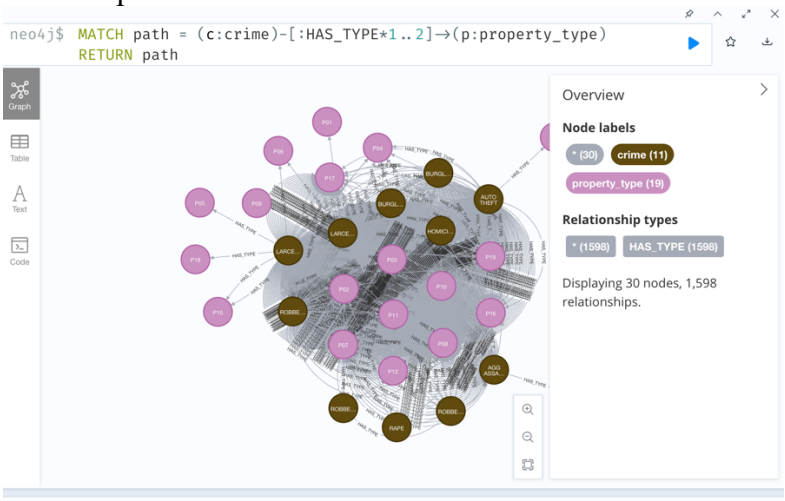
4. Find relationships between nodes
   From the picture below, we can know that the relationship between crime and beat.



5. Find the neighbour nodes of a node
   From the picture below, we can see that the crimes related to beat 107 are “AUTO THEFT” and “ROBBERY-RESIDENCE”

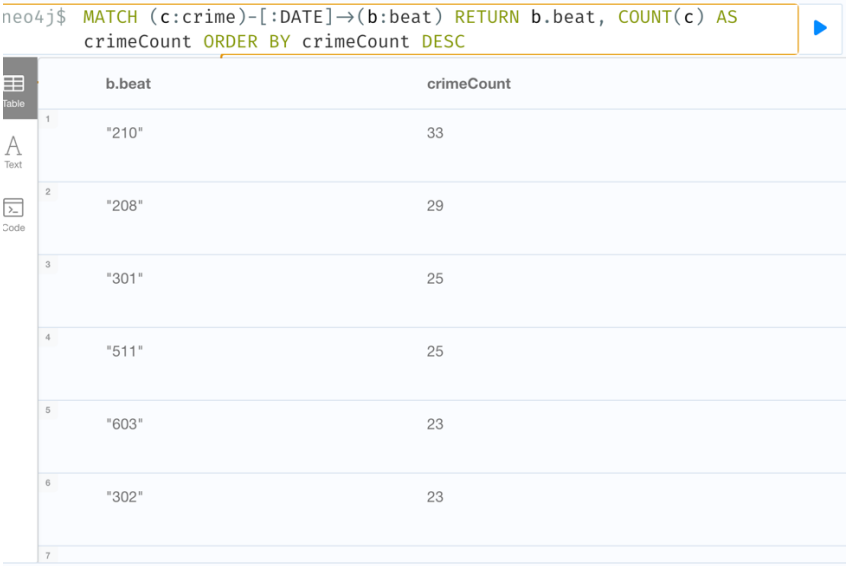


6. Find the path of a node



7. Aggregation

From the picture below, we can see that the number of crimes per beat and sort them in descending order. For example, the number of beat 210 is 33.



	b.beat	crimeCount
1	"210"	33
2	"208"	29
3	"301"	25
4	"511"	25
5	"603"	23
6	"302"	23
7		

IV. Queries

Question 1. How many crimes are recorded for auto theft in Buckhead Village in 2010?

Answer:36

```
MATCH (c:crime{crime: "AUTO THEFT"})-[r:DATE]→(b:beat)-[z:ZONE]→(n:neighbourhood {neighbourhood: "Buckhead Village"})
WHERE r.date ⩾ datetime('2010-01-01') AND r.date ⩽ datetime('2010-12-31')
RETURN COUNT(c.crime) as crimeCount
```

crimeCount
36

Question 2. Find the neighbourhoods that share the same crime types, organise in decending order of the number of common crime types.

```
MATCH (c:crime)-[:DATE]→(b:beat)-[:ZONE]→(n:neighbourhood)
WITH n, COLLECT(DISTINCT c.crime) AS commonCrimeTypes, COUNT(DISTINCT c) AS numCommonCrimeTypes
WHERE numCommonCrimeTypes > 1
RETURN numCommonCrimeTypes AS crimeCount, commonCrimeTypes, COLLECT(n.neighbourhood) AS neighborhoodNames
ORDER BY crimeCount DESC
```

crimeCount	commonCrimeTypes
7	["AUTO THEFT", "BURGLARY-RESIDENCE", "HOMICIDE", "LARCENY-FROM VEHICLE", "LARCENY-NON VEHICLE", "ROBBERY-PEDESTRIAN", "ROBBERY-RESIDENCE"]
7	["AGG ASSAULT", "AUTO THEFT", "BURGLARY-RESIDENCE", "LARCENY-NON VEHICLE", "RAPE", "LARCENY-FROM VEHICLE", "ROBBERY-RESIDENCE"]
7	["AGG ASSAULT", "AUTO THEFT", "BURGLARY-NONRES", "BURGLARY-RESIDENCE", "LARCENY-FROM VEHICLE", "ROBBERY-PEDESTRIAN", "LARCENY-NON VEHICLE"]
7	["AUTO THEFT", "BURGLARY-RESIDENCE", "LARCENY-FROM VEHICLE", "LARCENY-NON VEHICLE", "AGG ASSAULT", "BURGLARY-NONRES", "ROBBERY-PEDESTRIAN"]
6	["AGG ASSAULT", "AUTO THEFT", "BURGLARY-RESIDENCE", "LARCENY-FROM VEHICLE", "LARCENY-NON VEHICLE", "ROBBERY-PEDESTRIAN"]
6	["BURGLARY-NONRES", "LARCENY-FROM VEHICLE", "LARCENY-NON VEHICLE", "AGG ASSAULT", "AUTO THEFT", "ROBBERY-RESIDENCE"]

```
MATCH (c:crime)-[:DATE]→(b:beat)-[:ZONE]→(n:neighbourhood)
WITH n, COLLECT(DISTINCT c.crime) AS commonCrimeTypes, COUNT(DISTINCT c) AS numCommonCrimeTypes
WHERE numCommonCrimeTypes > 1
RETURN numCommonCrimeTypes AS crimeCount, commonCrimeTypes, COLLECT(n.neighbourhood) AS neighborhoodNames
ORDER BY crimeCount DESC
```

neighborhoodNames
["Mechanicsville"]
["Glenrose Heights"]
["Campbellton Road"]
["Downtown"]
["Washington Park", "Hunter Hills", "West Lake", "Carver Hills", "Almond Park", "Rockdale", "West Highlands", "Lenox", "Buckhead Heights", "Pine Hills", "Old Fourth Ward", "Edgewood"]
["Harland Terrace"]

Question 3. Return the top 5 neighbourhoods for auto theft in 2010

```
MATCH (c:crime {crime: "AUTO THEFT"})-[d:DATE]→(b:beat)-[:ZONE]→(n:neighbourhood)
WHERE d.date ⩾ datetime('2010-01-01') AND d.date ⩽ datetime('2010-12-31')
WITH n, COUNT(c) AS crimeCount
ORDER BY crimeCount DESC
LIMIT 5
RETURN n.neighbourhood, crimeCount
```

Table | Text | Code

	n.neighbourhood	crimeCount
1	"Lindbergh/Morosgo"	42
2	"Midtown"	36
3	"Buckhead Village"	36
4	"Lakewood Heights"	36
5	"Grant Park"	33

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

```
MATCH (c:crime)-[:HAS_TYPE]→(p:property_type)
WITH p.type AS propertyType, COLLECT(DISTINCT c.crime) AS crimes
RETURN propertyType, crimes
```

Table | Text | Code

	propertyType	crimes
1	"amenity"	["AGG ASSAULT", "AUTO THEFT", "BURGLARY-NONRES", "BURGLARY-RESIDENCE", "LARCENY-FROM VEHICLE", "LARCENY-NON VEHICLE", "
2	"building"	["AGG ASSAULT", "AUTO THEFT", "BURGLARY-NONRES", "BURGLARY-RESIDENCE", "LARCENY-FROM VEHICLE", "LARCENY-NON VEHICLE", "
3	"house_number"	["AGG ASSAULT", "AUTO THEFT", "BURGLARY-NONRES", "BURGLARY-RESIDENCE", "HOMICIDE", "LARCENY-FROM VEHICLE", "LARCENY-NOI
4	"leisure"	["AGG ASSAULT", "AUTO THEFT", "BURGLARY-NONRES", "BURGLARY-RESIDENCE", "LARCENY-FROM VEHICLE", "LARCENY-NON VEHICLE"]
5	"office"	["AGG ASSAULT", "LARCENY-FROM VEHICLE", "ROBBERY-PEDESTRIAN"]
6	"place"	["AGG ASSAULT"]

Question 5. Which month in 2010 has the highest crime rate? Return one record each for each beat.

```
MATCH (c:crime)-[d:DATE]→(b:beat)
WHERE d.date ⩾ datetime('2010-01-01') AND d.date ⩽ datetime('2010-12-31')
WITH b, datetime(d.date).month AS month, COUNT(c) AS crimeCount
ORDER BY b.beat, crimeCount DESC
WITH b.beat AS beat, COLLECT({month: month, crimeCount: crimeCount})[0] AS highestCrimeMonth
RETURN beat, highestCrimeMonth.month AS month, highestCrimeMonth.crimeCount AS crimeCount
ORDER BY beat
```

Table | Text | Code

	beat	month	crimeCount
1	"101"	1	2
2	"102"	1	5
3	"103"	1	2
4	"104"	1	2
5	"105"	1	6
6	"106"	1	7

```
MATCH (c:crime)-[d:DATE]→(b:beat)-[z:ZONE]→(n:neighbourhood)
WHERE d.date ⩾ datetime('2010-01-01') AND d.date ⩽ datetime('2010-12-31')
WITH z, datetime(d.date).month AS month, COUNT(c) AS crimeCount
ORDER BY z.zone, crimeCount DESC
WITH z.zone AS zone, COLLECT({month: month, crimeCount: crimeCount})[0] AS highestCrimeMonth
RETURN zone, highestCrimeMonth.month AS month, highestCrimeMonth.crimeCount AS crimeCount
ORDER BY zone
```

Table | Text | Code

	zone	month	crimeCount
1	"1"	1	7
2	"2"	1	22
3	"3"	1	11
4	"4"	1	9
5	"5"	1	10
6	"6"	1	10

Our own query:

1. How many auto theft has been reported in beat 107 in 2010

```
MATCH (c:crime{crime: "AUTO THEFT"})-[r:DATE]→(b:beat {beat: "107"})
WHERE r.date ⩾ datetime('2010-01-01') AND r.date ⩽ datetime('2010-01-31')
RETURN COUNT(c.crime) as crimeCount
```

Table | Text

	crimeCount
1	1

2. Query the type of crime with the highest number of crimes in each beat, and display the number of occurrences

```
MATCH (c:crime)-[:DATE]→(b:beat)
WITH b.beat AS beat, c.crime AS crimeType, COUNT(*) AS crimeCount
ORDER BY beat, crimeCount DESC
WITH beat, COLLECT({crimeType: crimeType, crimeCount: crimeCount})[0] AS
mostFrequentCrime
RETURN beat, mostFrequentCrime.crimeType AS mostFrequentCrimeType,
mostFrequentCrime.crimeCount AS mostFrequentCrimeCount
```

Table | Text | Code

	beat	mostFrequentCrimeType	mostFrequentCrimeCount
1	"101"	"LARCENY-FROM VEHICLE"	7
2	"102"	"AUTO THEFT"	4
3	"103"	"AUTO THEFT"	3
4	"104"	"AGG ASSAULT"	2
5	"105"	"LARCENY-NON VEHICLE"	4
6	"106"	"AGG ASSAULT"	3

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

1. **Crime Pattern Analysis**: By utilizing Graph Data Science techniques such as Breadth First Search (BFS) and Depth First Search (DFS), investigators can analyze the patterns and relationships among different crime incidents in Atlanta. They can start with a specific crime incident and traverse the graph to explore its connections with other related crimes, suspects, locations, or entities. This analysis can help identify crime hotspots, recurring patterns, and potential criminal networks.

2. **Shortest Path Analysis**: Techniques like Dijkstra's algorithm and A* algorithm can be applied to find the shortest path between two specific locations or entities in the crime graph. For example, investigators may want to determine the shortest path between a crime scene and a suspect's residence, or between two high-crime areas. This information can assist in

resource allocation, response planning, and strategic decision-making for law enforcement agencies.

3. **Crime Network Analysis**: Graph Data Science techniques can help in analyzing the relationships and connections among different entities involved in criminal activities, such as suspects, victims, and accomplices. By applying algorithms like BFS and DFS, investigators can uncover hidden relationships and identify key nodes within criminal networks. This analysis can aid in identifying the most influential individuals, tracking criminal operations, and disrupting organized crime networks.

4. **Crime Prediction and Prevention**: Graph Data Science techniques can be utilized to build predictive models for crime in Atlanta. By leveraging historical crime data, relationships, and contextual information, machine learning algorithms can be trained to predict the likelihood of future crimes in specific locations or neighborhoods. This information can enable law enforcement agencies to allocate resources proactively, implement targeted crime prevention strategies, and enhance public safety.

5. **Community Policing and Engagement**: The crime graph can also be used to facilitate community policing efforts and enhance community engagement. By visualizing crime data and its relationships, law enforcement agencies can communicate crime trends and patterns effectively to the public. This can promote community awareness, encourage citizen involvement, and facilitate collaborative efforts between law enforcement and community members to address crime-related issues.