A diagram of a neighborhood

Description automatically generated with low confidence

Figure : Graph from Arrow tool

A screenshot of a computer

Description automatically generated with medium confidence

Figure : Graph from Neo4j

Graph database creation:

|  |
| --- |
| // Load Crime Nodes  LOAD CSV WITH HEADERS FROM 'file:///crime\_in.csv' AS row  CREATE (c:Crime {crimeID: toInteger(row.number),          crimeType: row.crime,          locationID:toInteger(row.location\_id),          dateID:toInteger(row.date\_id),          typeID:toInteger(row.type\_id)          }) |
| A screenshot of a computer code  Description automatically generated with medium confidence |
| // Load Date Nodes  LOAD CSV WITH HEADERS FROM 'file:///date.csv' AS row  CREATE (d:Date {dateID: toInteger(row.date\_id),          day:toInteger(row.day),          month:toInteger(row.month),          year:toInteger(row.year)          }) |
| A screenshot of a computer code  Description automatically generated with low confidence |

|  |
| --- |
| // Load Location Nodes  LOAD CSV WITH HEADERS FROM 'file:///location.csv' AS row  CREATE (l:Location {locationID: toInteger(row.location\_id),          propertyID: toInteger(row.type\_id),          neighborhoodID:toInteger(row.neighborhood\_id),          property: row.type,          beat: toInteger(row.beat)          }) |
| A screenshot of a computer  Description automatically generated with medium confidence |
| // Load Beat Nodes  LOAD CSV WITH HEADERS FROM 'file:///beat.csv' AS row  CREATE (b:Beat {beat: toInteger(row.beat),          zone:toInteger(row.ZONE)          }) |
| A screenshot of a computer code  Description automatically generated with low confidence |

|  |
| --- |
| // Load Neighborhood nodes  LOAD CSV WITH HEADERS FROM 'file:///neighborhood.csv' AS row  CREATE (n:Neighborhood {neighborhoodID: toInteger(row.neighborhood\_id),          neighborhood:row.neighborhood          }) |
| A screen shot of a computer  Description automatically generated with medium confidence |
| // Create relationships between Crime and Date  MATCH (c:Crime), (d:Date)  WHERE c.dateID = d.dateID  CREATE (c)-[:OCCURRED\_ON]->(d) |
| A screenshot of a computer  Description automatically generated with medium confidence |
| // Create relationships between Crime and Location  MATCH (c:Crime), (l:Location)  WHERE c.locationID = l.locationID  CREATE (c)-[:OCCURRED\_AT]->(l) |
| A screenshot of a computer program  Description automatically generated with low confidence |
| // Create relationships between Location and Neighborhood  MATCH (l:Location), (n:Neighborhood)  WHERE l.neighborhoodID = n.neighborhoodID  CREATE (l)-[:LOCATED\_IN]->(n) |
| A screenshot of a computer  Description automatically generated with medium confidence |
| // Create relationships between Location and Beat  MATCH (l:Location), (b:Beat)  WHERE l.beat = b.beat  CREATE (l)-[:UNDER]->(b) |
|  |

Querries:

|  |
| --- |
| // How many crimes are recorded for a given crime type in a specified neighbourhood for a particular period?  MATCH (c:Crime)-[:OCCURRED\_AT]->(l:Location)-[:LOCATED\_IN]->(n:Neighborhood),         (c)-[:OCCURRED\_ON]->(d:Date)  WHERE c.crimeType = 'LARCENY-NON VEHICLE'    AND n.neighborhood='Downtown'    AND d.day = 31    AND d.month = 10  RETURN count(c) AS crimeCount |
|  |

|  |
| --- |
| //Find the neighbourhoods that share the same crime types, organise in decending order of the number of common crime types.  MATCH (n1:Neighborhood)<-[:LOCATED\_IN]-(:Location)<-[:OCCURRED\_AT]-(c1:Crime)  WITH n1, COLLECT(DISTINCT c1.crimeType) AS crimeTypes  MATCH (n2:Neighborhood)<-[:LOCATED\_IN]-(:Location)<-[:OCCURRED\_AT]-(c2:Crime)  WHERE n2 <> n1  WITH n1, n2, COLLECT(DISTINCT c2.crimeType) AS commonCrimeTypes, crimeTypes  RETURN n1.neighborhood AS neighborhood1, n2.neighborhood AS neighborhood2, SIZE([x IN commonCrimeTypes WHERE x IN crimeTypes]) AS numCommonCrimeTypes  ORDER BY numCommonCrimeTypes DESC |
|  |

|  |
| --- |
| // Return the top 5 neighbourhoods for a specified crime for a specified duration.  MATCH (n:Neighborhood)<-[:LOCATED\_IN]-(:Location)<-[:OCCURRED\_AT]-(c:Crime)-[:OCCURRED\_ON]->(d:Date)  WHERE c.crimeType = 'AGG ASSAULT'    AND d.day >=10    AND d.day <=28  RETURN n.neighborhood AS neighborhood, COUNT(c) AS numCrimes  ORDER BY numCrimes DESC  LIMIT 5 |
|  |

|  |
| --- |
| // Find the types of crimes for each property type.  MATCH (l:Location)<-[:OCCURRED\_AT]-(c:Crime)  RETURN l.property AS propertyType, COLLECT(DISTINCT c.crimeType) AS crimeTypes |
|  |

|  |
| --- |
| // 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.)  MATCH (c:Crime)-[:OCCURRED\_ON]->(d:Date), (l:Location)-[:UNDER]->(b:Beat)  WHERE d.year = 2010  WITH d.month AS month, b.beat AS beat, b.zone AS zone, COUNT(c) AS numCrimes  ORDER BY beat, zone, numCrimes DESC  WITH beat, zone, COLLECT(month)[0] AS highestCrimeMonth  RETURN beat, zone, highestCrimeMonth |
|  |

|  |
| --- |
| //  Find the zones that have are adjacent and sharing the same high crime months.  MATCH (c1:Crime)-[:OCCURRED\_AT]->(l1:Location)-[:UNDER]->(b1:Beat),        (c2:Crime)-[:OCCURRED\_AT]->(l2:Location)-[:UNDER]->(b2:Beat),        (c1)-[:OCCURRED\_ON]->(d:Date)  WHERE c1 <> c2    AND l1 <> l2    AND b1 <> b2    AND d.year = 2010    AND d.month = 10  WITH b1.zone AS zone1, b2.zone AS zone2, d.month AS highCrimeMonth  RETURN DISTINCT zone1, zone2, highCrimeMonth |
|  |

|  |
| --- |
| // Write cypher code to find which property type has the lowest count of any type of crime  MATCH (l:Location)<-[:OCCURRED\_AT]-(c:Crime)  WITH l.property AS propertyType, COUNT(c) AS numCrimes  RETURN propertyType  ORDER BY numCrimes ASC  LIMIT 1 |
|  |
| // Write cypher code to find which neighbourhood has the lowest crime count of all crime type  MATCH (n:Neighborhood)<-[:LOCATED\_IN]-(l:Location)<-[:OCCURRED\_AT]-(c:Crime)  WITH n, COUNT(c) AS numCrimes  RETURN n.neighborhood AS neighborhood  ORDER BY numCrimes ASC  LIMIT 1 |
|  |

The Atlanta dataset's crime data lends itself well to analysis using the graphic database.

1. Complex linkages between crimes, suspects, victims, and locations can be handled effectively using graph databases.

2. Investigating relationships and locating crime trends, hotspots, or repeated behaviors are made easier with the help of graph database traversal and pattern matching. 3. Scalability and good performance are features of graph databases. This is crucial for handling big quantities of criminal data and running intricate searches across various related entities.

4. Because graph databases are schema less, it is simple to adapt them to new attributes and growing criminal data models.

5. Context-rich investigations in graph databases offer more information on gang affiliations, social ties, and networks between suspects and victims.

6. Logical reasoning and semantic reasoning assist reveal hidden relationships and foresee probable links to criminal activity.

7. A chart database combines information from various sources and offers a consolidated view of crime data from various domains, including case reports, arrests, and court records. 8. Multiple agencies or departments can provide and analyze criminal data together thanks to the chart database's collaboration and data sharing features.

9. Chart databases can be used to determine resource allocations, criminal tendencies, and preemptive law enforcement tactics.

10. The chart database makes use of the chart structure to visualize crime networks and enable geographical analysis to comprehend crime trends in Atlanta.

YouTube Video link:

https://youtu.be/peEenVdcj7g