2020

Big Data Architecture & Governance

Northeastern University

Assignment Name:

US Accidents (2016 - 2021)

Student Names:
Vraj Mashruwala
Sahitya Kothapalli
Zarana Bhadricha

1. Contents

2. Assignment

2.1. Case

Each team should select a dataset to analyze and build an analytical dashboard as a Proof-of-concept to illustrate the value of data driven analytics. You need to present your dataset.

.

2.2. Assignment Goals

To work with datasets, Perform/Create:

- Create you group assignment project in Velero:
  - Project
  - Project Plan
  - Resource Allocation
  - Timesheet
  - Issues & Risks.
  - You are required to report on your team progress every week
- **Data Profiling** – Using Python profiling library, describe your understanding of the data.
- **Data Wrangling and Cleansing** - Pandas/Alteryx/XSV
  - Filtering and Aggregating if needed.
  - Missing value handling.
  - Deriving additional columns from existing datasets if needed.
  - Cleaning (removing blank spaces, formatting dates, Capitalizing etc.) .
- Database Installation: Install NEO4J database .
- Data Mapping and Integration to your Database for the Entire Dataset.
- **Business and Technical Metadata** – develop business term list describing all the data elements available in the file.
- **Data Validation** – Validate the data using python data libraries.
- **Data Visualization** – Create a presentation dashboard to reflect your understanding of the data, you may use python visualization libraries or Power BI/Tableau
- **System Integration and User Acceptance Testing** - Test Cases – describe your validation & testing process.
- **Risks/Issues** – identify risks and issues related to your project.
- Describe challenges encountered and how you resolved them.
- **End User Instructions (Steps to run your Dashboard)** – provide a full description how to run your process:

- Database Creation and load.
- Visualization interpretation - describe information regarding your findings.

2.2.1. Visualization Deliverables

Once you wrangle/clean/join/integrate the data, import the data into **NEO4J** and illustrate how to use the appropriate graph to illustrate various aspects of analysis.

Questions to consider:

- Columns used for dimensions, and columns that are used for measurement.
- How would you generate new dimensions if needed
- Who would use this dashboard and how they benefit from your dashboard
- What value would be generated using this dashboard

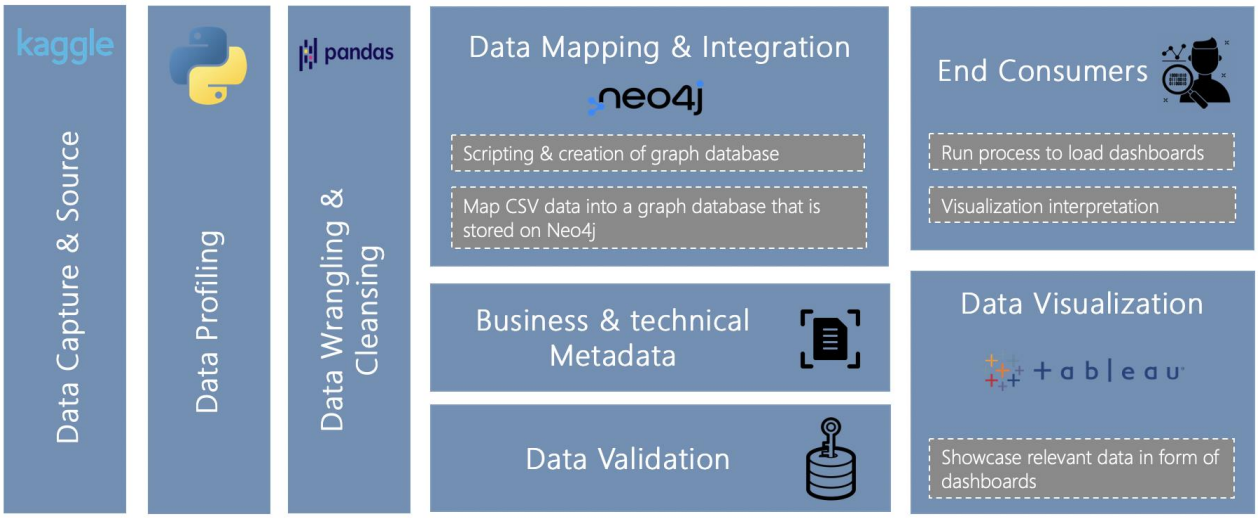
2.2.2. Other deliverables

- Presentation of the entire work from the first step till the dashboards including the Velero screenshots.
- Business and technical metadata presentation – Identifying all available business terms and extracting related technical metadata.
- Complete explanation of the dashboard and usability.
- Complete instruction as how to implement and run the database load, technical meta data extraction, and dashboard.

3. Documentation

3.1. Vision Diagram

The vision diagram gives a great brief up of what steps are going to be taken during the course of this project along with the technologies used. Please find below the vision diagram.

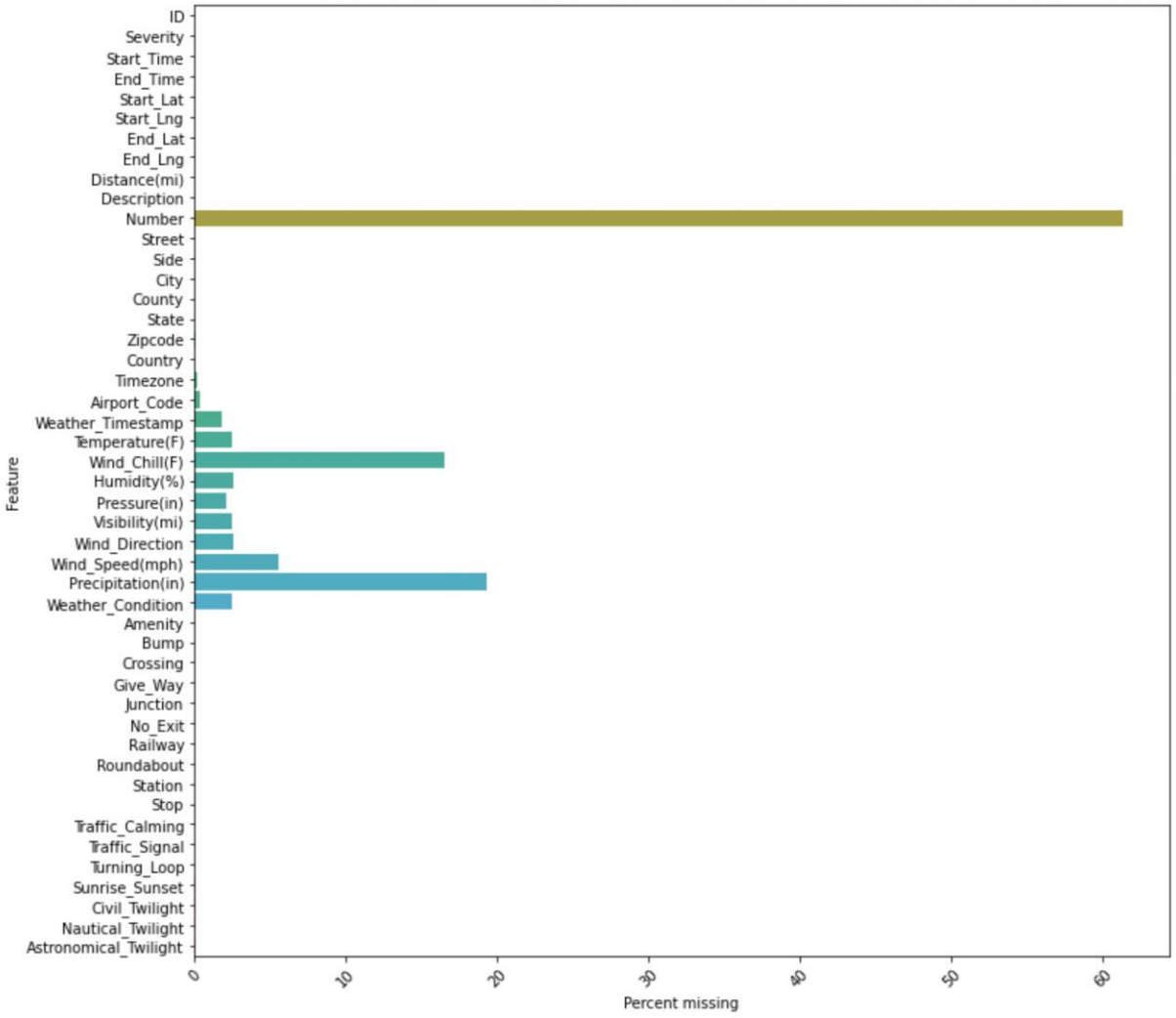


3.2. Data Profiling

The first step to getting familiar with the dataset assigned (US Accidents 2016 - 2021) is to do data profiling. Without having a knowledge of the data, it is impossible to carry out data cleansing or even design the graph database. Data profiling was done using the **pandas, matplotlib & seaborn** libraries in Python before generating the entire profiling report using the **pandas_profiling** library. Some in-line data profiling done in the Jupyter notebook include:

- Viewing all column names & data types
- Finding the amount of null values within each column & plotting a bar graph for this as a percentage of null values in the column
- Plotting the correlation matrix between all columns
- Finally, the profiling report was generated as HTML document using the **pandas_profiling** library

Plot showing the percentage of missing values for each column:



Following snapshots are from the data profiling report that is generated using the **pandas_profiling** library:

Overview of the data:

Dataset statistics

Number of variables	47
Number of observations	2845342
Missing cells	3414349
Missing cells (%)	2.6%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	773.4 MiB
Average record size in memory	285.0 B

Variable types

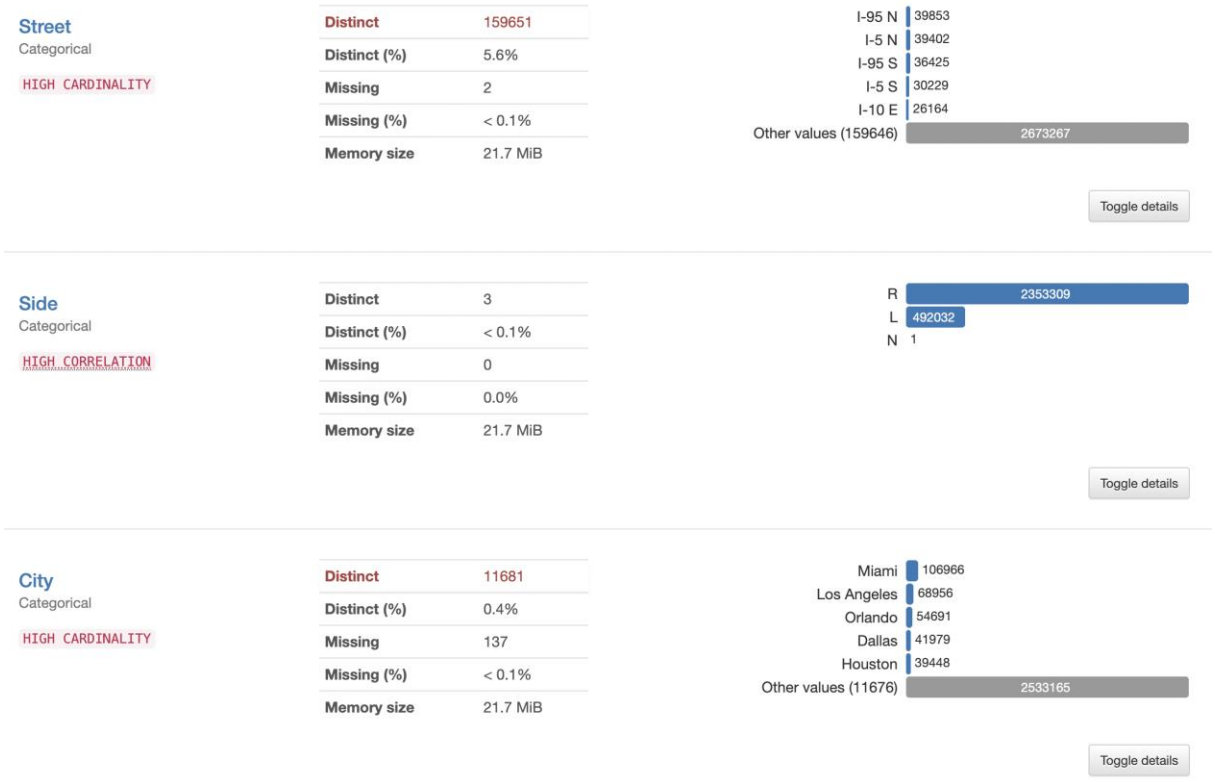
Categorical	21
Numeric	13
Boolean	13

Viewing first few rows of the data:

First rows

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Description
0	A-1	3	2016-02-08 00:37:08	2016-02-08 06:37:08	40.108910	-83.092860	40.112060	-83.031870	3.230	Between Sawmill Rd/Exit 20 and OH
1	A-2	2	2016-02-08 05:56:20	2016-02-08 11:56:20	39.865420	-84.062800	39.865010	-84.048730	0.747	At OH-4/OH-235/Exit 41 - Accident.
2	A-3	2	2016-02-08 06:15:39	2016-02-08 12:15:39	39.102660	-84.524680	39.102090	-84.523960	0.055	At I-71/US-50/Exit 1 - Accident.
3	A-4	2	2016-02-08 06:51:45	2016-02-08 12:51:45	41.062130	-81.537840	41.062170	-81.535470	0.123	At Dart Ave/Exit 21 - Accident.
4	A-5	3	2016-02-08 07:53:43	2016-02-08 13:53:43	39.172393	-84.492792	39.170476	-84.501798	0.500	At Mitchell Ave/Exit 6 - Accident.
5	A-6	2	2016-02-08 08:16:57	2016-02-08 14:16:57	39.063240	-84.032430	39.067310	-84.058510	1.427	At Dela Palma Rd - Accident.
6	A-7	2	2016-02-08 08:15:41	2016-02-08 14:15:41	39.775650	-84.186030	39.772750	-84.188050	0.227	At OH-4/Exit 54 - Accident.
7	A-8	2	2016-02-08 11:51:46	2016-02-08 17:51:46	41.375310	-81.820170	41.367860	-81.821740	0.521	At Bagley Rd/Exit 235 - Accident.
8	A-9	2	2016-02-08 14:19:57	2016-02-08 20:19:57	40.702247	-84.075887	40.699110	-84.084293	0.491	At OH-65/Exit 122 - Accident.
9	A-10	2	2016-02-08 15:16:43	2016-02-08 21:16:43	40.109310	-82.968490	40.110780	-82.984000	0.826	At I-71/Exit 26 - Accident.

Detailed overview of the data in *Street, Side, City* columns; such overview is provided for every single of 47 columns in the dataset. This overview shows the total distinct values, missing values and size of a particular column



For each column, you can view the statistics, common values and extreme values. Here, as an example, we show these details for the *Temperature* column:

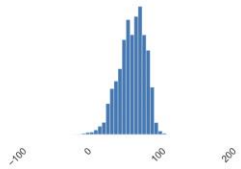
Temperature(F)

Real number (ℝ)

HIGH CORRELATION
MISSING

Distinct	788
Distinct (%)	< 0.1%
Missing	69274
Missing (%)	2.4%
Infinite	0
Infinite (%)	0.0%
Mean	61.79355592

Minimum	-89
Maximum	196
Zeros	984
Zeros (%)	< 0.1%
Negative	6444
Negative (%)	0.2%
Memory size	21.7 MiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	-89
5-th percentile	29
Q1	50
median	64
Q3	76
95-th percentile	88.9
Maximum	196
Range	285
Interquartile range (IQR)	26

Descriptive statistics

Standard deviation	18.62262938
Coefficient of variation (CV)	0.3013684697
Kurtosis	0.03002155249
Mean	61.79355592
Median Absolute Deviation (MAD)	13
Skewness	-0.4910794711
Sum	171543113.2
Variance	346.8023252
Monotonicity	Not monotonic

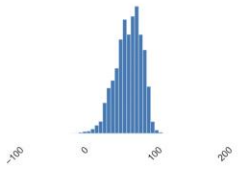
Temperature(F)

Real number (ℝ)

HIGH CORRELATION
MISSING

Distinct	788
Distinct (%)	< 0.1%
Missing	69274
Missing (%)	2.4%
Infinite	0
Infinite (%)	0.0%
Mean	61.79355592

Minimum	-89
Maximum	196
Zeros	984
Zeros (%)	< 0.1%
Negative	6444
Negative (%)	0.2%
Memory size	21.7 MiB



Toggle details

Statistics | Histogram | Common values | Extreme values

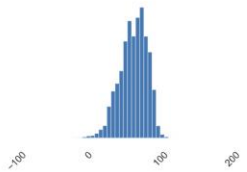
Value	Count	Frequency (%)
73	64505	2.3%
77	63575	2.2%
75	60534	2.1%
72	59681	2.1%
68	58557	2.1%
63	58259	2.0%
64	57937	2.0%
70	57760	2.0%
66	56336	2.0%
59	56025	2.0%

Temperature(F)

Real number (ℝ)

HIGH CORRELATION
MISSING

Distinct	788	**Minimum**	-89
Distinct (%)	< 0.1%	**Maximum**	196
Missing	69274	**Zeros**	984
Missing (%)	2.4%	**Zeros (%)**	< 0.1%
Infinite	0	**Negative**	6444
Infinite (%)	0.0%	**Negative (%)**	0.2%
Mean	61.79355592	**Memory size**	21.7 MiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Minimum 10 values | Maximum 10 values

Value	Count	Frequency (%)
-89	2	< 0.1%
-77.8	1	< 0.1%
-58	1	< 0.1%
-50	1	< 0.1%
-40	1	< 0.1%
-33	1	< 0.1%
-30	1	< 0.1%
-29	4	< 0.1%

3.3. Data Wrangling and Cleansing

This is a crucial step in our project. In order to rightly visualize the data, it is important to strategize the data cleaning. Since we do not want to have any NULLs or NaNs being displayed or being left out in our final dashboards, we decided to replace these null values with placeholder values. All the steps are undertaken using the **Pandas** library in a Jupyter Notebook

1. For all columns which are of string/object type, the placeholder value assigned for all nulls is '**Not listed**'. Similarly, for columns having numeric values/data type, the placeholder value for nulls is **–99**. For instance, in the ***Street*** column, all nulls have been replaced by 'Not listed'. Similarly, for the ***Temperature_F*** column which captures the temperature in Fahrenheit, all nulls are placed by –99. This helps in differentiating the absent values and at the same time also ensuring no blank/null values remain

```
string_type_columns = ['Street', 'City', 'Zipcode', 'Timezone', 'AirportCode', 'WindDirection', 'WeatherCondition',
                       'SunriseSunset', 'CivilTwilight', 'NauticalTwilight', 'AstronomicalTwilight']
#first making a list of column names which are of string type and have NULL values
```

```
df[string_type_columns] = df[string_type_columns].fillna('Not listed')
```

For all int/float columns, fill NULLs with -99

```
integer_type_columns = ['Number', 'Temperature_F', 'WindChill_F', 'Humidity_Perc', 'Pressure_in', 'Visibility_mi',
                        'WindSpeed_mph', 'Precipitation_in']
```

```
df[integer_type_columns] = df[integer_type_columns].fillna(-99)
```

2. The *Side* column primarily has '**R**' (right) and '**L**' (left) as values, however there is one sole row where the value for this column is '**N**'. It is best to remove this outlier to avoid confusion. But remove does not mean deleting the entire row as data loss is not something we want. So, the better strategy is to replace this '**N**' with the most frequent value in this column, which happens to be '**R**' in this case

```
df['Side'] = df['Side'].replace(['N'], 'R')
```

```
df['Side'].value_counts() #just to verify that 'N' is replaced
```

```
R    2353310
L     492032
Name: Side, dtype: int64
```

3. **Rename the columns** to make them CamelCase. In the case of columns that also mention the measurement units (eg: F, %, in etc), add these units as a suffix along with an underscore. For eg: ***Temperature_F***, ***Humidity_%*** etc.

```
df.columns = df.columns.str.replace("[_,)]", "", regex = True)
df.columns = df.columns.str.replace("(", "_", regex = False)
df.columns = df.columns.str.replace('Humidity_%', 'Humidity_Perc')
```

4. **Type conversions** are done for the columns that have time stamps. They were of string type initially, but in order to have some useful visualizations pertaining to time/date, these columns need to be in the Timestamp data type. This data type conversion is done

5. **Creating new column**: ***AccidentYear*** column has been created to capture the year the accident occurred in. This is done keeping in mind the visualizations we will do

After performing these cleansing & wrangling steps, the final cleansed data set was stored as a CSV. This is the CSV that will be used for the remaining steps. Here is a photo showing the final columns in the CSV:

```
ID                  Pressure_in
Severity            Visibility_mi
StartTime           WindDirection
EndTime             WindSpeed_mph
StartLat            Precipitation_in
StartLng            WeatherCondition
EndLat              Amenity
EndLng              Bump
Distance_mi         Crossing
Description         GiveWay
Number              Junction
Street              NoExit
Side                Railway
City                Roundabout
County              Station
State               Stop
Zipcode             TrafficCalming
Country             TrafficSignal
Timezone            TurningLoop
AirportCode         SunriseSunset
WeatherTimestamp    CivilTwilight
Temperature_F       NauticalTwilight
WindChill_F         AstronomicalTwilight
Humidity_Perc       AccidentYear
```

3.4. Database Installation

Database installation was rather straight forward. At first, Neo4j (Desktop) was installed. Tableau was the other tool needed. Python was pre-installed in all our systems. Versions used:

Neo4j Desktop Version 1.4.15 (1.4.15.85)

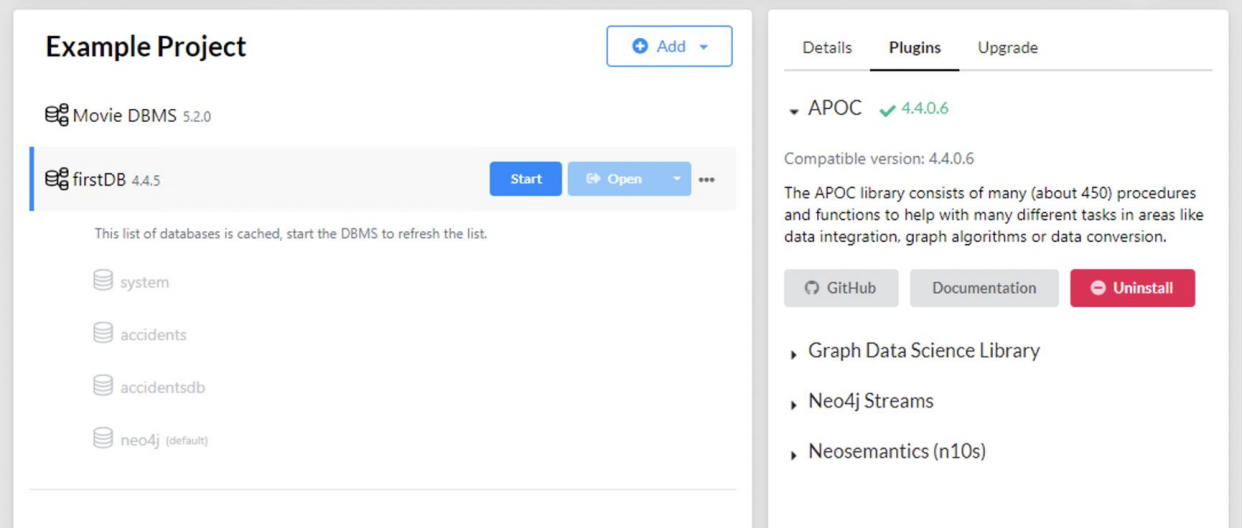
Tableau Desktop 2022.3.1(20223.22.1108.0821) 64-bit
Python 3.8.12

In order to connect Neo4j's graph database with Tableau for the visualization, we also need to setup the connection capability between these two. This needs to be done using Tableau and allowing to setup up a connection with our graph database present on Neo4j from there. To achieve this, we use the JDBC driver & do the following:

1) Download the Neo4j - BI Connector JDBC file (.jar)

2) Move this .jar file in the Drivers folder of Tableau (your Tableau installation directory/Tableau/Drivers/)

3) Make sure you install the APOC plugin on the Neo4j database that is in use. Do this by clicking on the database which opens a pop up on the right of the screen and under plugins install APOC. Restart your database after installing the plugin



4) On Tableau select ‘Other JDBC connection’ option for the data source & enter the JDBC connection URL in the following format:

jdbc:neo4j://localhost:7687/dbname?&UID=neo4j&PWD=yourdbpassword

Note: dbname here is the name of the databse that is inside your holder databse on Neo4j. From the picture given above, my dbname could be accidents or accidentsdb

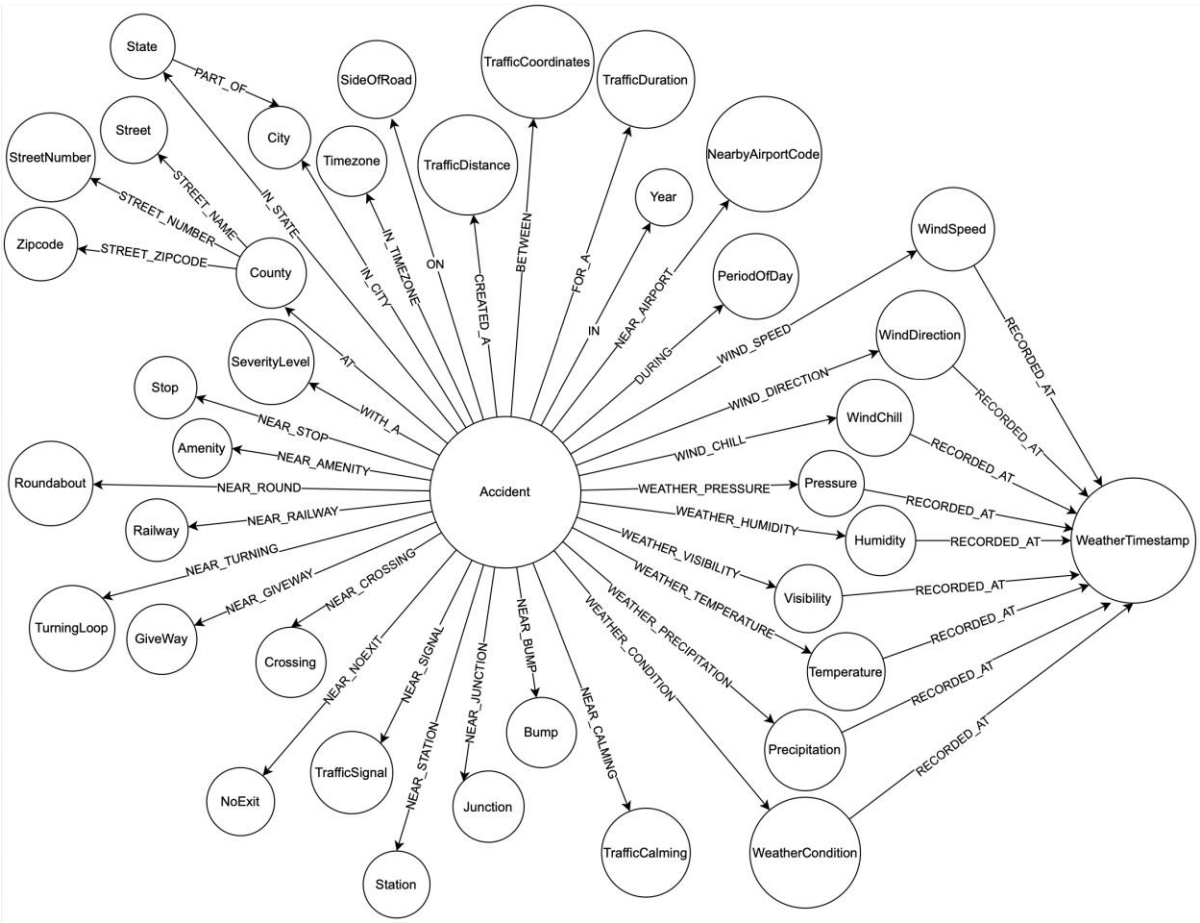
5) Also enter the username and password in the dialog boxes below (these are same as the ones you mention in your connection string; the username in Neo4j is always defaulted to neo4j whereas password is set when you create a new database on Neo4j)

6) Once you click on sign in, you will be successfully taken to the data source page to then create your model within Tableau before using it on sheets and dashboards (brief up about this process is further mentioned in the 3.6 Data Validation and Data Visualization topic below)

3.5. Data Mapping and Integration

To get familiar with the working and the features of Neo4j, the sample employee data and the corresponding script that has been provided to the class was run. This gives a thorough understanding of not only graph databases but also how to map a normal CSV/excel file to a graph database.

Extracting knowledge from the sample, the data model for our data is created. This is our graph DB model. The graph database consists of **Nodes, Properties and Relationships**. The end goal of our project is to present the important findings as visualizations, as a result of this we decided our model keeping this in mind. The model has maximum of the nodes in a direct relationship with the main accident node. This helps us to easily query any of these details rather than needing to traverse multiple nodes to get to a node from accident node. After going over a couple of iterations, here is the final graph model:

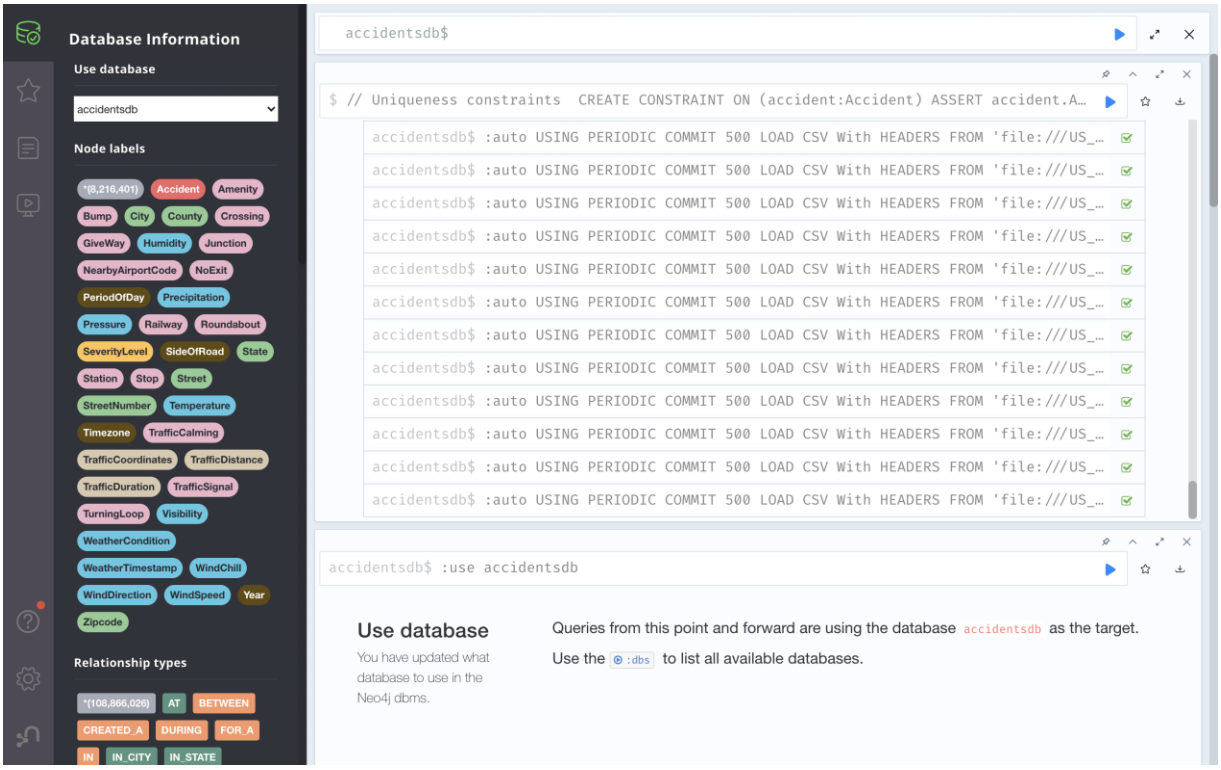


The ‘mapping’ of CSV columns to the Node label are as follows*:

Properties	Node
ID	Accident
Description	
Severity	SeverityLevel
Start Time	TrafficDuration
End Time	
TrafficDuration	
Start Lat	TrafficCoordinates
Start Long	
End Lat	
End Long	
TrafficCoordinates	
Distance	TrafficDistance
Number	StreetNumber
Street	Street
Side	SideofRoad
City	City
County	County
State	State
Zipcode	Zipcode
Country	
Timezone	Timezone
Airport Code	NearbyAirportCode
Weather Timestamp	WeatherTimestamp
Temperature	Temperature
Wind Chill	WindChill
Humidity	Humidity
Pressure	Pressure
Visibility	Visibility
Wind Direction	WindDirection
Wind Speed	WindSpeed
Precipitation	Precipitation
Weather Condition	WeatherCondition
Amenity	Amenity
Bump	Bump
Crossing	Crossing
Give Way	GiveWay
Junction	Junction
No Exit	NoExit
Railway	Railway
Roundabout	Roundabout
Station	Station
Stop	Stop
Traffic Calming	TrafficCalming
Traffic signal	TrafficSignal
Turning loop	TurningLoop
Sunrise Sunset	PeriodOfDay
Civil Twilight	
Nautical Twilight	
Astronomical Twilight	
AccidentYear	Year

*Note: Column ***Country*** has not been mapped to a node since it has a constant value of USA, it is not needed in the visualization. Similarly, the columns ***CivilTiwlight, NauticalTwilight, AstronomicalTwilight*** have the same value for each record as the value of column ***SunriseSunset***. Therefore, these 3 columns have also not been mapped to any node and can be ignored for our purposes.

With the model as reference, the cypher script for our data model was created. The script was made in steps: first the constraints, then the nodes, then the relationships. To check the functioning of the script, it was run on a subset consisting 10,000 records from the cleaned dataset which actually consists of about 2.9 million records. Creating this subset file and using that is a good way to quickly run the cypher script multiple times to troubleshoot any errors in the script. Once the script was finalized and ran without any errors, with adequate results, it is then run on the whole cleaned dataset. Here is the screenshot of a successful run of the cypher script:

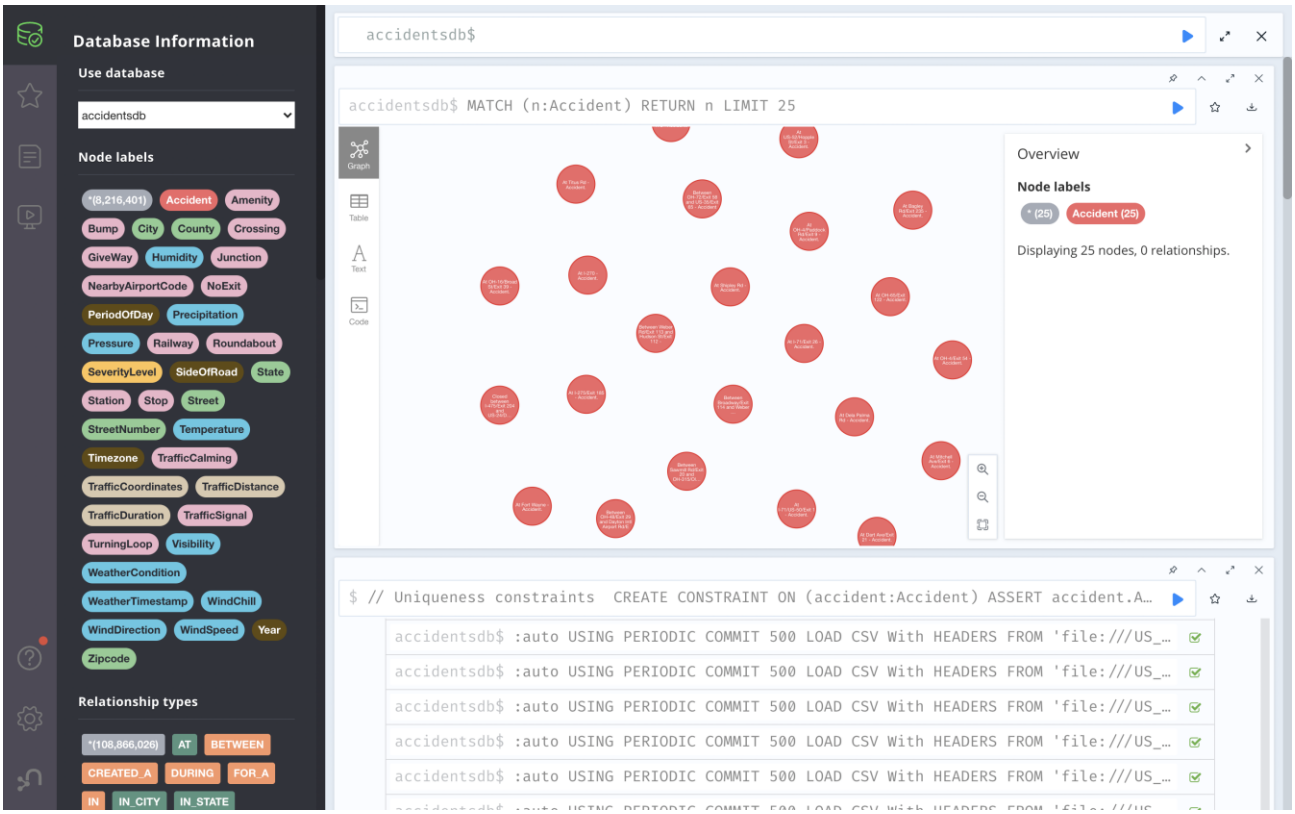

Database Information
Use database
accidentsdb
Node labels
*(8,216,401)
Accident
Amenity
Bump
City
County
Crossing
GiveWay
Humidity
Junction
NearbyAirportCode
NoExit
PeriodOfDay
Precipitation
Pressure
Railway
Roundabout
SeverityLevel
SideOfRoad
State
Station
Stop
Street
StreetNumber
Temperature
Timezone
TrafficCalming
TrafficCoordinates
TrafficDistance
TrafficDuration
TrafficSignal
TurningLoop
Visibility
WeatherCondition
WeatherTimestamp
WindChill
WindDirection
WindSpeed
Year
Zipcode
Relationship types
*(108,866,026)
AT
BETWEEN
CREATED_A
DURING
FOR_A
IN
IN_CITY
IN_STATE
accidentsdb$
$ // Uniqueness constraints CREATE CONSTRAINT ON (accident:Accident) ASSERT accident.A…
accidentsdb$:auto USING PERIODIC COMMIT 500 LOAD CSV With HEADERS FROM 'file:///US_…
accidentsdb$:use accidentsdb
Use database
You have updated what database to use in the Neo4j dbms.
Queries from this point and forward are using the database accidentsdb as the target.
Use the :dbs to list all available databases.

3.6. Data Validation and Data Visualization

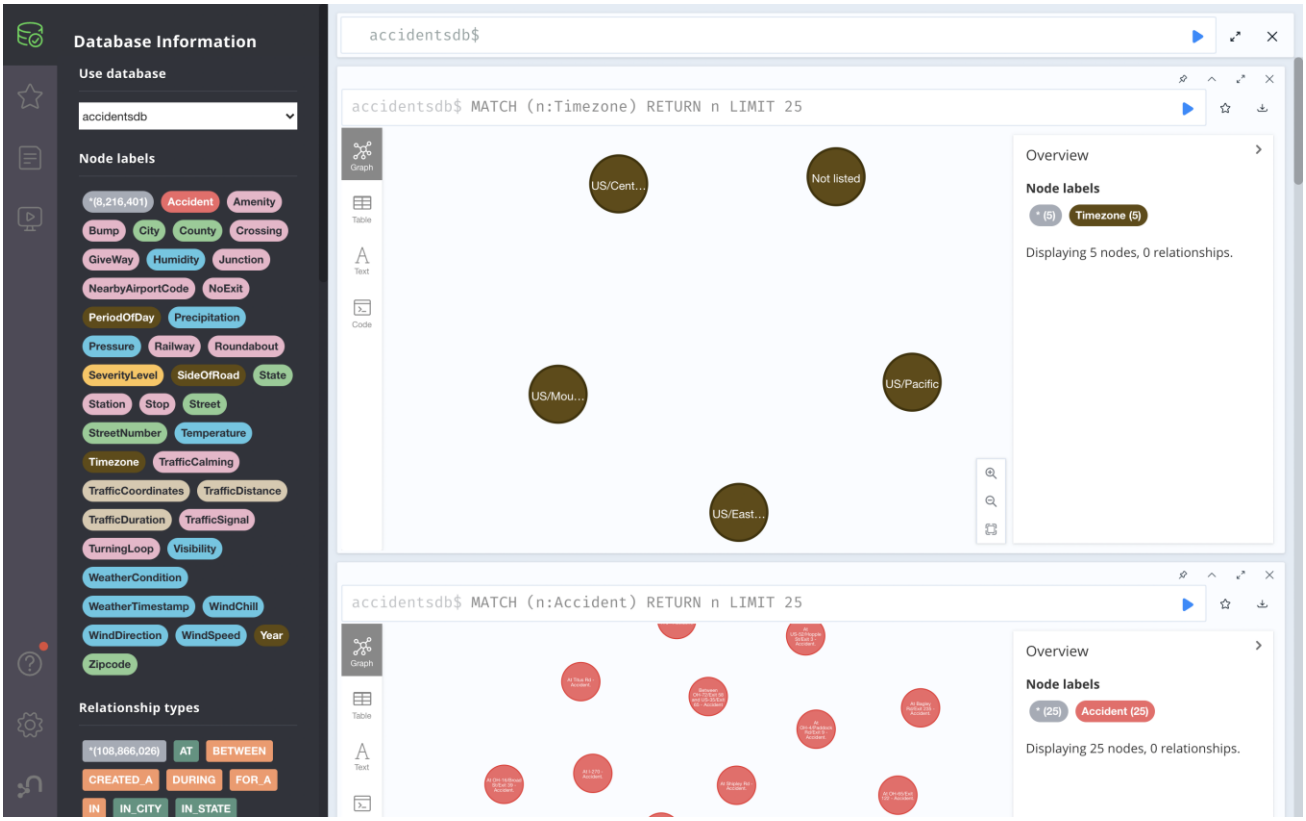
Before moving onto the Data Visualization, it is important to verify that the data loaded onto Neo4j is what we expected it to be. We need to check that everything was loaded correctly. This is pretty naive, however, we also need to do some User Acceptance Testing to thoroughly verify all data before moving to the visualization. The details of UAT carried out as mentioned in the next section. Over here, we talk about the basic data validation we perform before moving to visualization.

First, upon succesful execution of the cypher script we have a look at few of the nodes and relationships, to see if these graph DB components have been loaded properly or not.

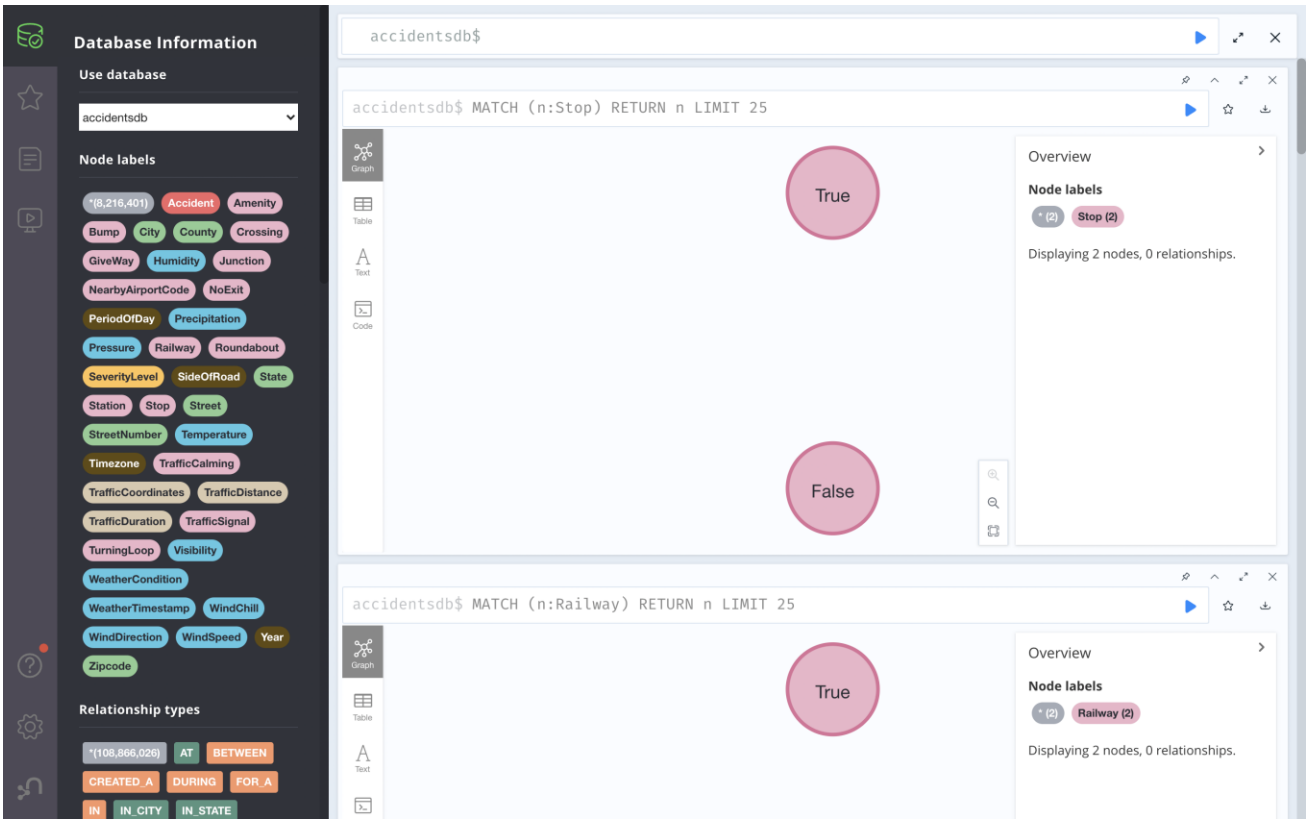
Viewing the ***Accident*** node:



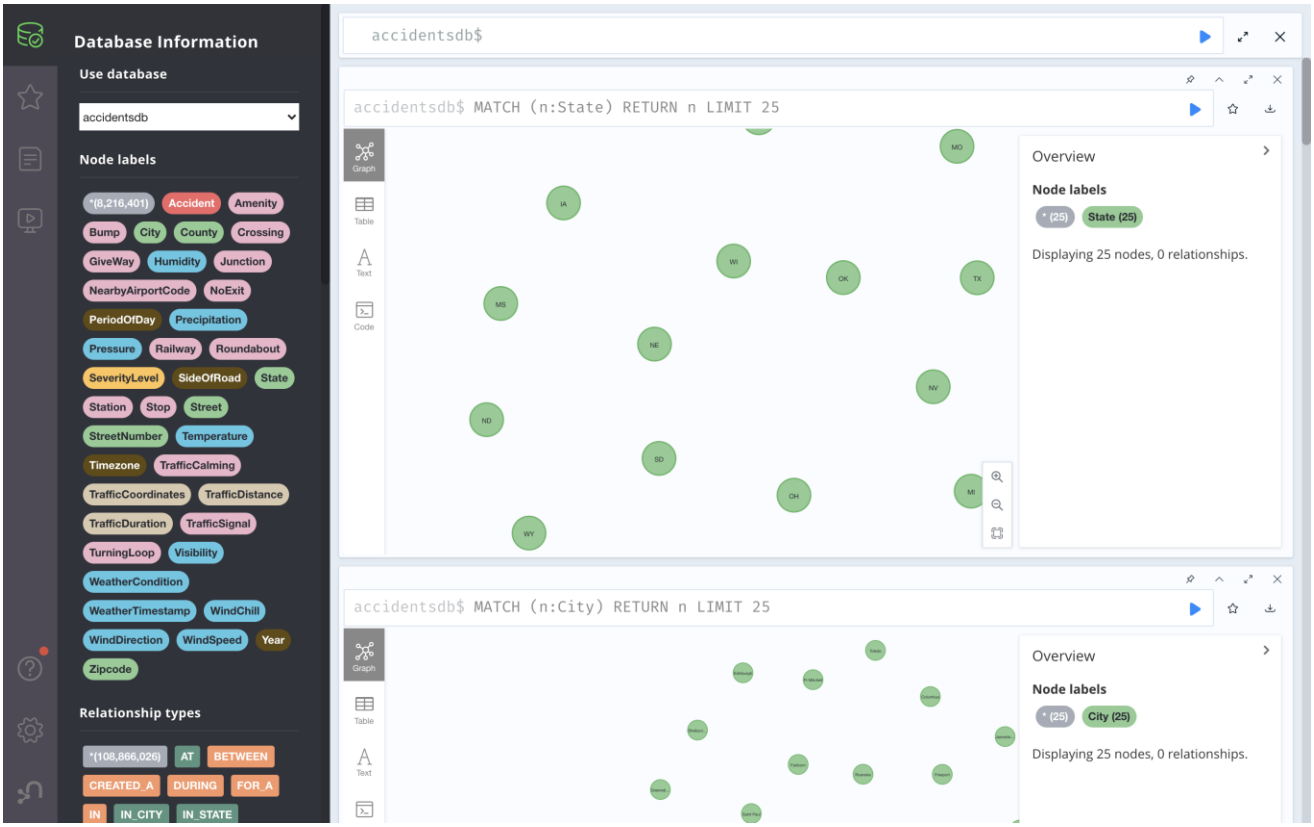
Viewing the ***Timezone*** node:



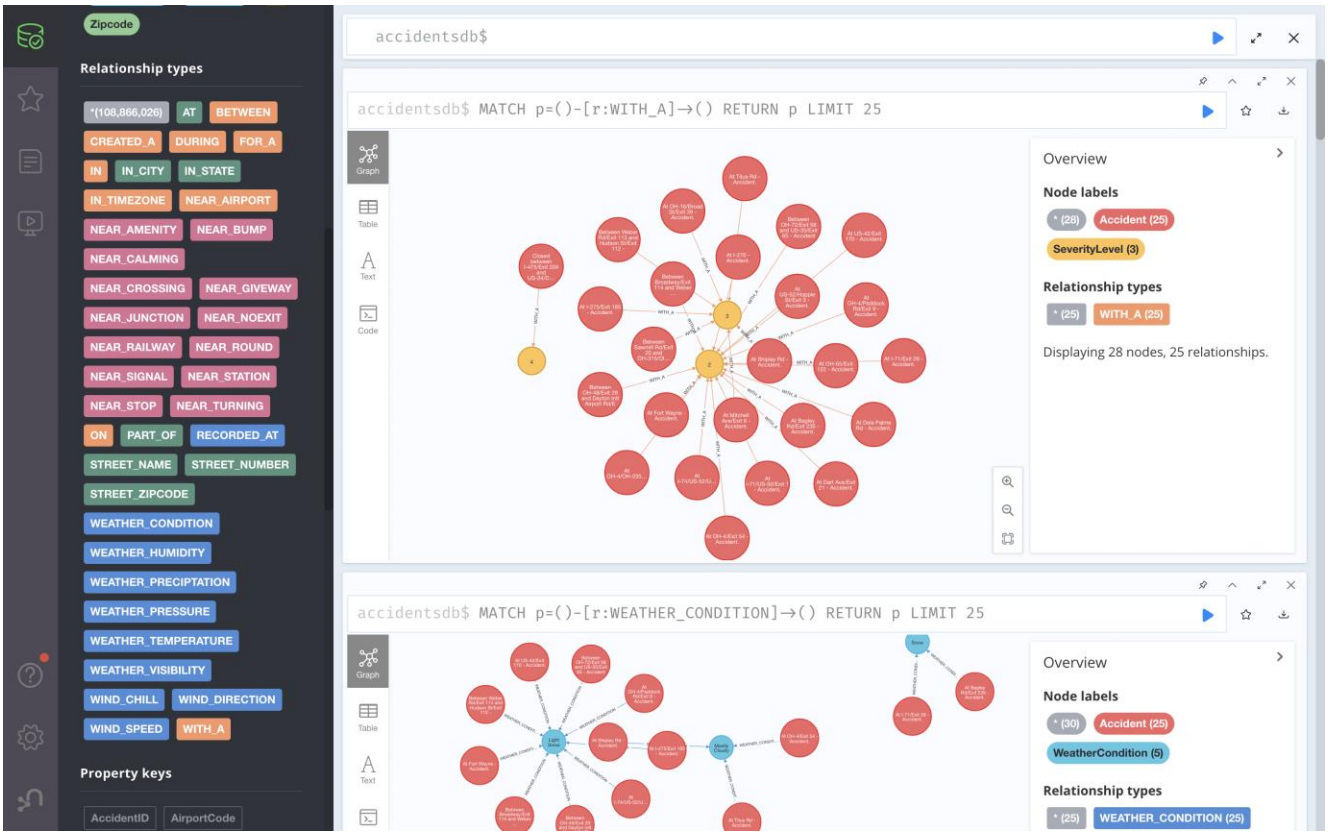
Viewing the ***Stop*** node:



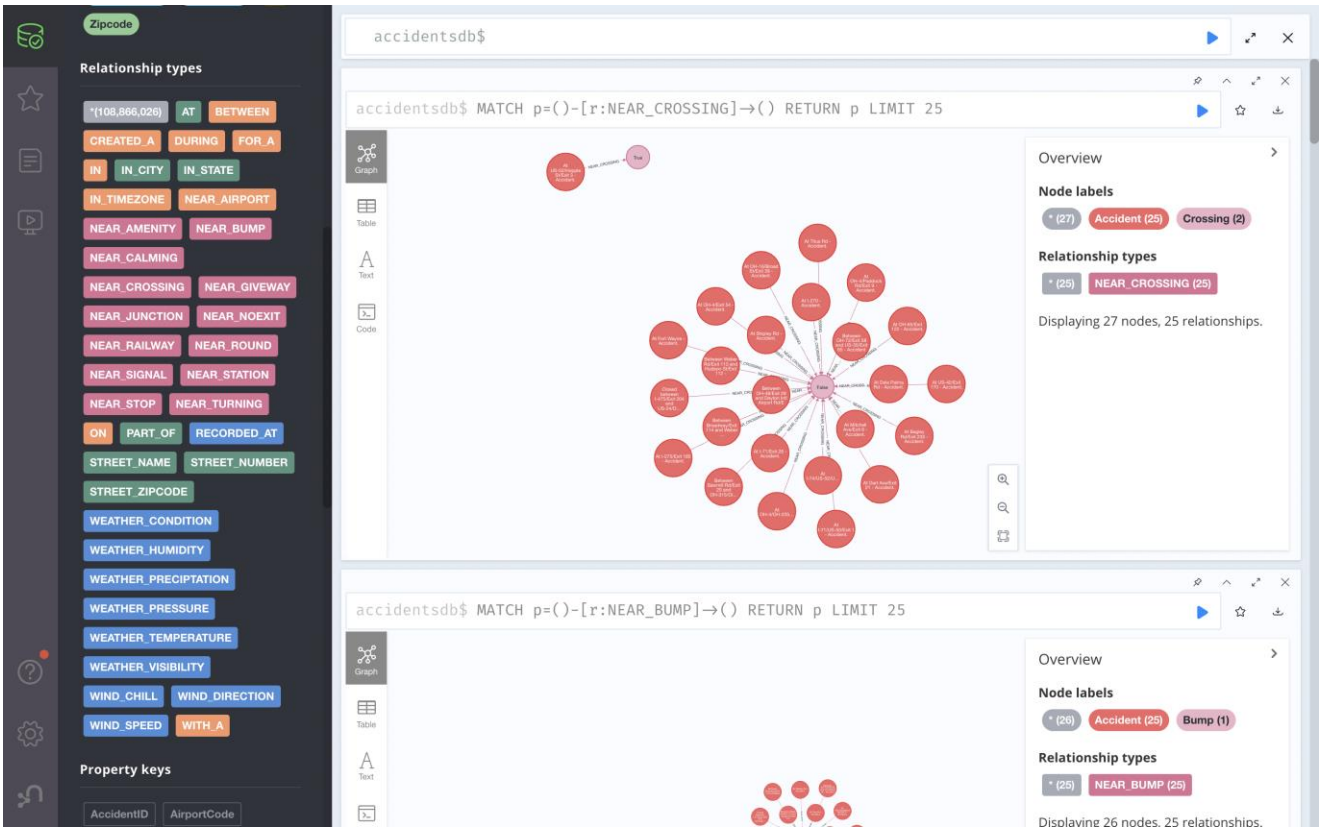
Viewing the ***State*** node:



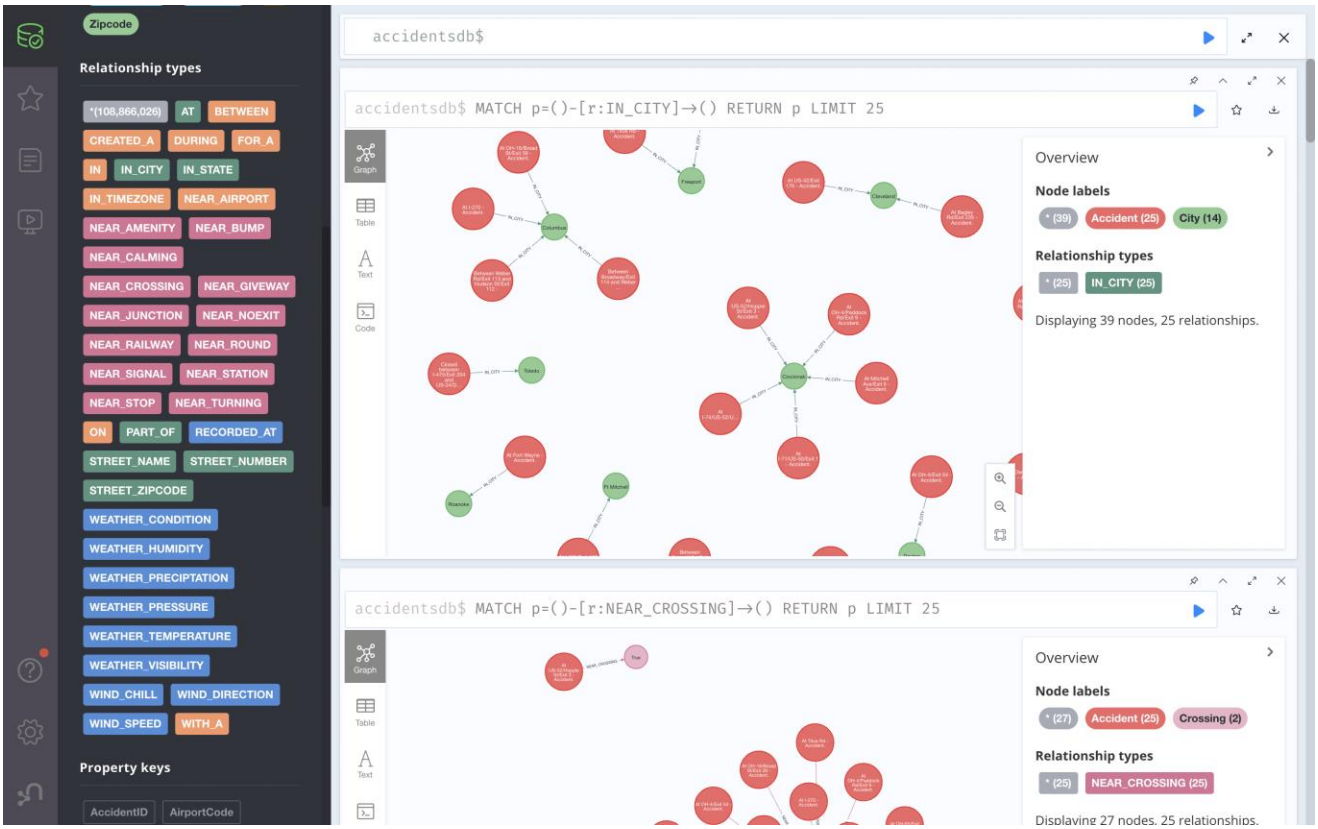
Viewing the ***WITH_A*** relationship:



Viewing the ***NEAR_CROSSING*** relationship:



Viewing the ***IN_CITY*** relationship:

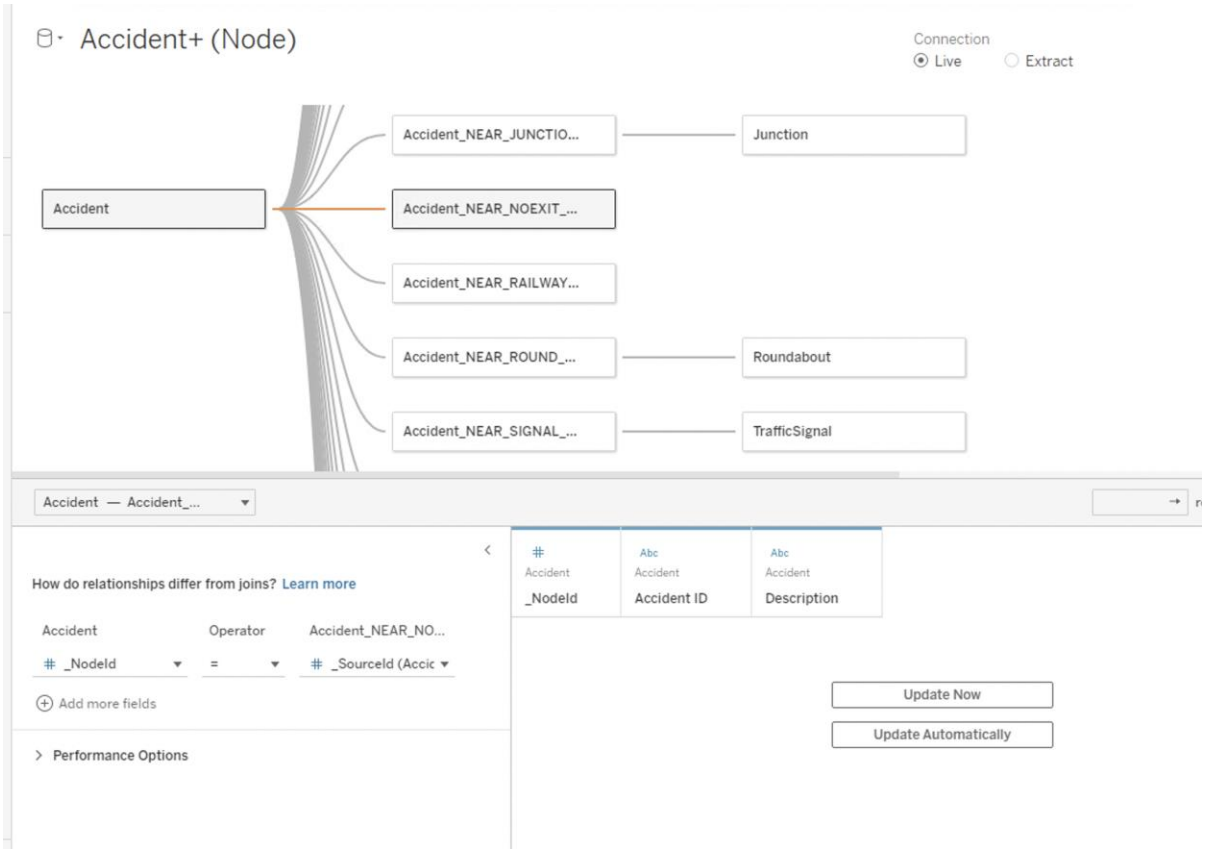


Data Visualization

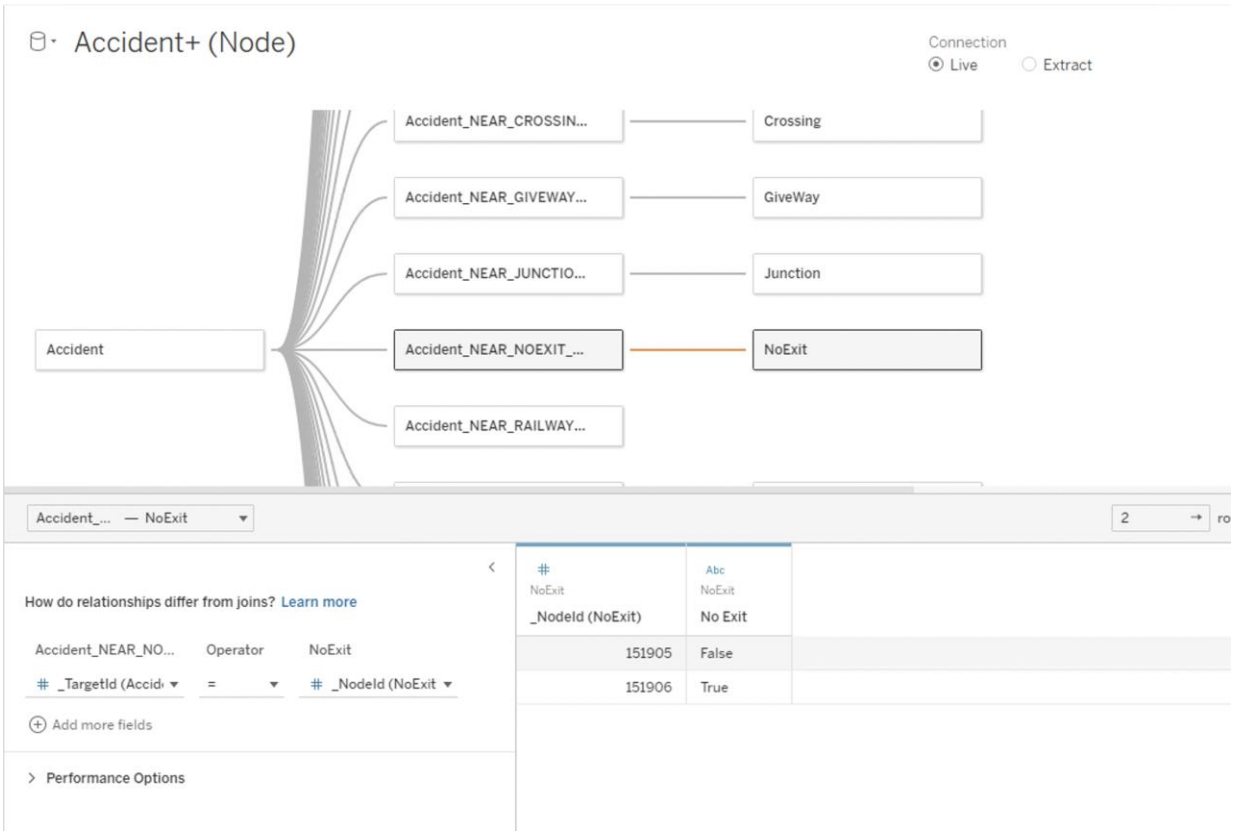
After following the steps mentioned under Database Installation (topic 3.4 above) to connect Tableau with your database that is on Neo4j, you need to first setup your data model. We setup our data model by following the databse design that we created (mentioned in 3.5 Data Mapping and Integration). To set up the model within Tableau, we do the following:

- Drag and drop the parent node (Accidents) into the space

- Next, drag and drop a realtionship. Once you do this, we also need to set up the relation between these two components (the node and the relationship). To do this, we select the ID properties to create this mapping between the two. The parent node's ID is equated to the Soucre ID of the relationship



- Next, drop in the target node of the relationship. And to make the join this time select the relationship's Target ID and equate it to the destination node's ID. This way a single logic of Node – Relationship -> Node is created. For example Accident -> NEAR_NOEXIT -> NoExit



- We repeat this process to create the entire model consisting of all the relationships & nodes

Once the model is defined, we are ready to create Dashboards. The US Accidents dataset contains a lot of informative columns. Designing just a single dashboard would not do justice to the amount of details that we can capture from such an informative dataset. We decide to create a number of dashboards to caputre various details. But, one question that is crucial here is what value will these dashboards provide? Will it only be a visualization of what the data is? This was already captured in our thorough data profiling report. So what will these dahsboards be? We need to look at these dashboards from a business perspective. These dahsboards will be used by the stakeholders to understand several insights regarding US accidents. These insights can help stakeholders to take business decisions that helps the business overall. A stakeholder of our US accidents data could be someone who wishes to know where major accidents occur, at what time do accidents usually occur, near what vicinity do accidents occurs and many such business questions. Answers to these questions can help the stakeholders to better come up with strategies that lead to reduce in the number of accidents or can make take measures to reduce the severity of accidents in case of situations where highly severe accidents occur. All this carries a very crucial role and our dashboards need to be able to translate to these business requirements and deliver business insights.

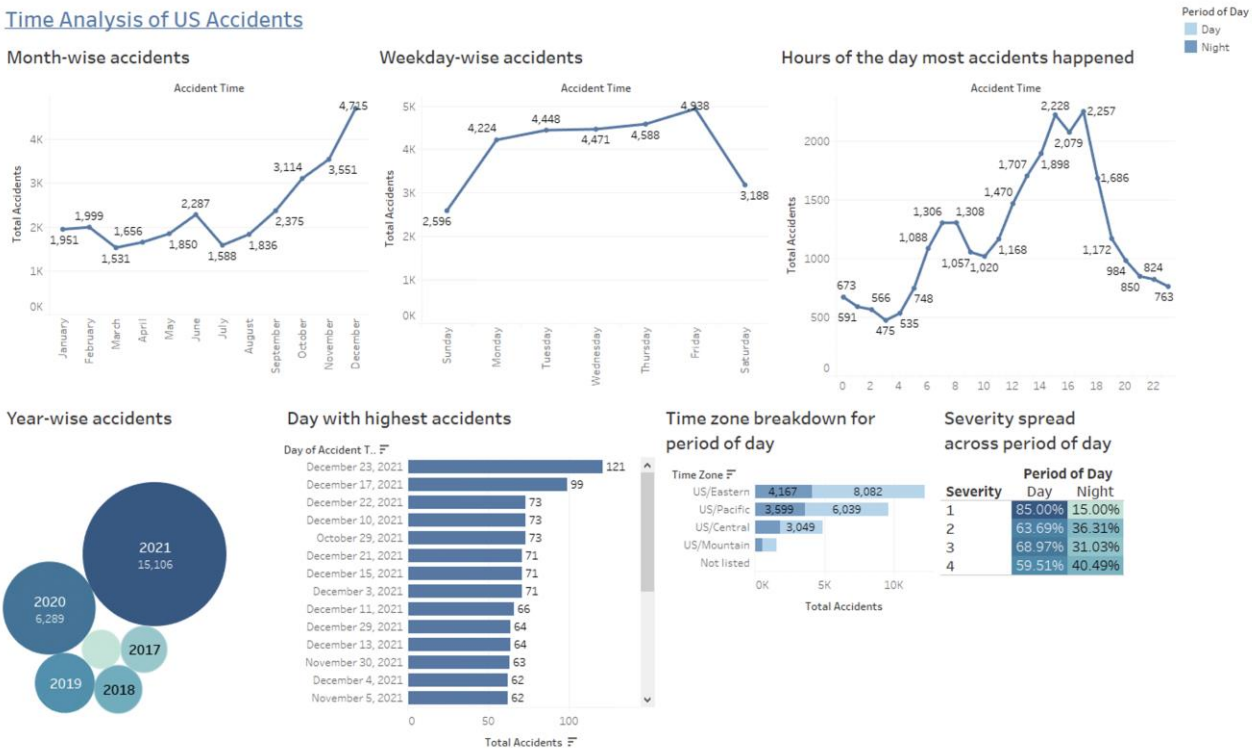
We divide the data present into 4 different domains and get to answering some business questions and delivering insights on these. The domains are:

1. **Time Analysis** (year, month, hour etc)
2. **Location Analysis** (State, City etc)
3. **Weather Analysis** (weather conditions, visibility, humidity etc)
4. **Vicinity Analysis** (amenities, traffic signal, crossing etc)

1. Time Analysis of US Accidents

Few questions that this analysis/dashboard answers:

- Year-wise count of accidents
- Which months typically have higher accidents?
- How are accidents spread across during weekdays?
- Which hours of a day have the most frequency of accidents?
- How are accidents divided between different time zones and periods of day?
- What is the spread of severity levels of accidents that happen during Day vs Night?



Business insights gathered from this Time Analysis dashboard:

- There has been an increase in the number of accidents every year since 2016
- During a year, the most number of accidents occurs during the months of December & November
- **Friday has the most number of accidents generally during the week**
- **During a day, 3pm, 4pm & 5pm are peak accident hours**
- 85% of severity level 1 accidents duirng the day, and only 15% during the night. However, percentage of accidents during the night is high for severity accidents- Severity level 4 accidents have 40% chance of being occured in the night (compared to 15% severity level 1 accidents) This means: **there are higher chances of an accident occured at night to be of high severity level**

2. Location Analysis for US Accidents

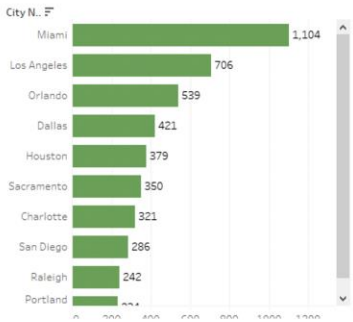
Few questions that this analysis/dashboard answers:

- What are the number of accidents breakdown by State?
- Top cities in terms of number of accidents?
- What are the percentages of accidents during the day and night for different States?
- Top traffic distances caused by accidents

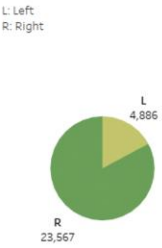
Location Analysis of US Accidents

Period of Day
Day
Night
Not listed

Top 10 Cities for Accidents



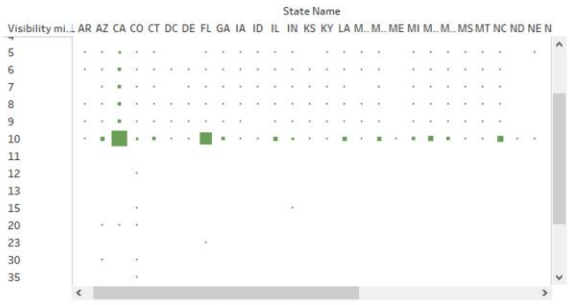
Accident Prone Side of Road



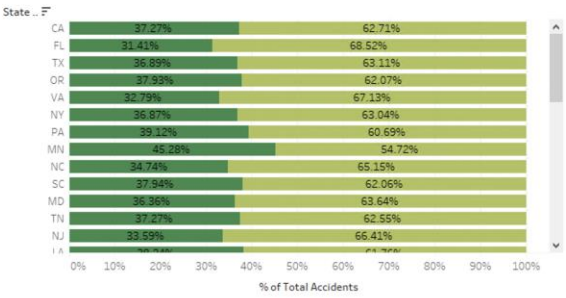
Top Traffic Distances w/ severity

Description	Distance	Severity
At NE-2 (North) - Accident.	21.313	2
Between CR-62/Geneva Rd/Guanella Pass Rd and CO-9/Main St - Accid..	22.299	3
Between I-83/Harrisburg West Intr/Exit 242 and Lebanon-Lancaster I..	18.512	2
Between RT-28/Exit 19 and RT-299/Exit 18 - Accident.	15.276	2
Closed between 7th St and 1st St - Road closed due to accident.	16.664	4
Closed between Antone Rd and NF-2/Fields Creek Rd - Road closed du..	19.236	4
Closed between BIA-6486 and Holbrook - Road closed due to accident.	45.83	4
Closed between CR-E18/Parrotts Ferry Rd and CA-89 - Road closed du..	56.171	4
Closed between Government Rd/Snake Rd/Exit 49 and US-27/SR-25/E..	26.686	4
Closed between I-70-BR/US-5/US-6/Exit 26 and US-6-BR/FAS-347/Exit..	30.608	4
Closed between Railroad St/Exit 70 and Asahel Curtis Rd/Exit 47 - Roa..	16.66	4
Closed between TX-494-loop and N Jackson St/N Chenevert St - Road ..	19.175	4
Closed between US-101/Highway Ave/Oregon Coast Hwy and Weathe..	18.559	4
Crash on PA 120 both directions between 1 miles WEST of SULLIVAN ..	15.387	2
Hazard @ National Forest Development Road 46 -- Impact: Closure	15.17	4
Incident on FLORIDA'S TPKE SB near MM 133 Drive with caution.	17.378	2
Incident on FLORIDA'S TPKE SB near MM 155 Drive with caution.	36.165	2

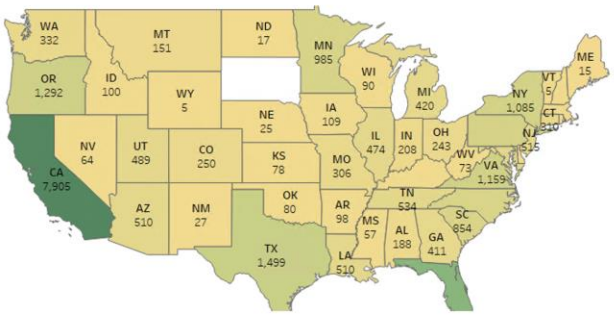
Visibility



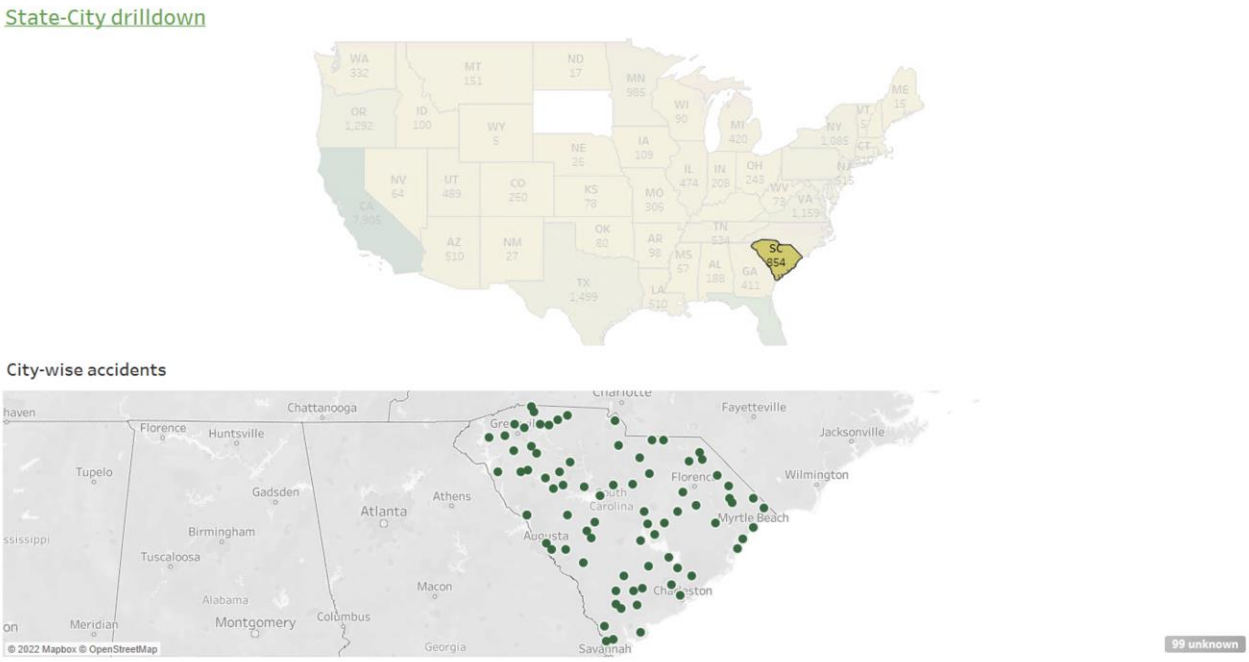
State wise day & night accidents breakdown



State-City drilldown



City-wise accidents



Business insights gathered from this Location Analysis dashboard:

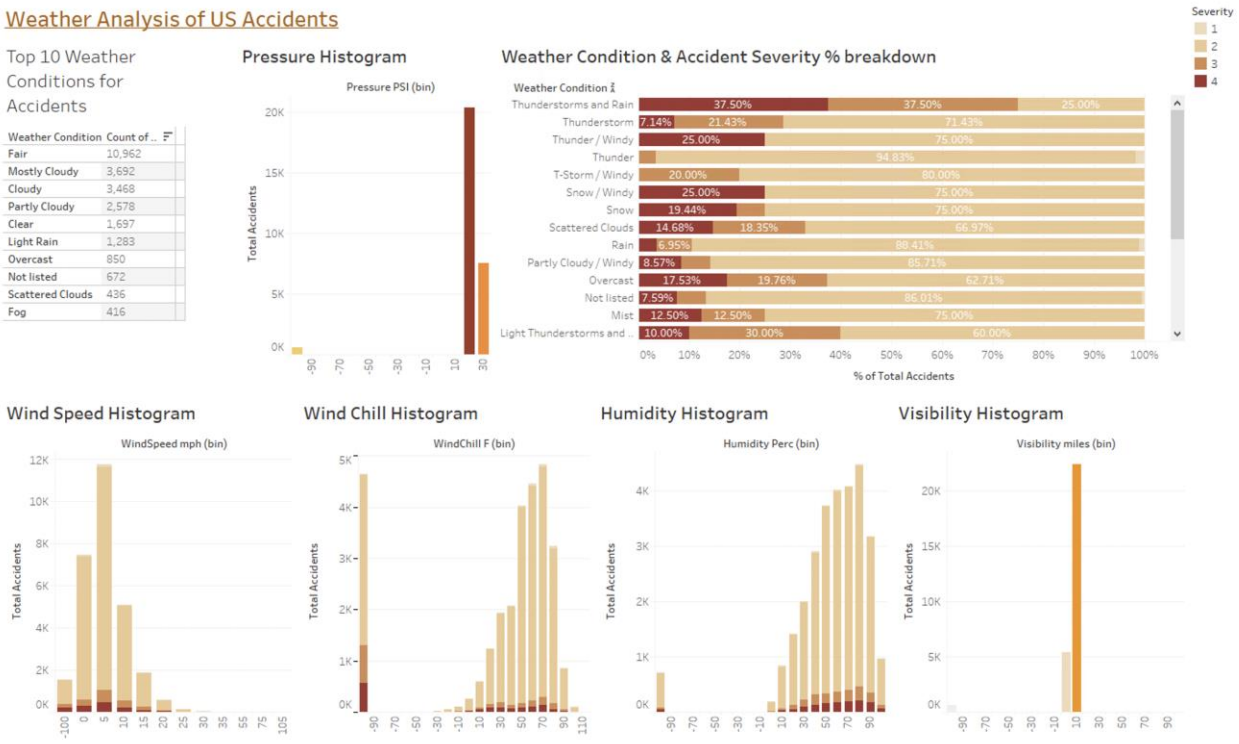
- **Cities that have most number of accidents: Miami, Los Angeles, Orlando**
- A drill-down for State and City: click on a State to have a view of accidents that occured in different City within the State
- **Almost 85% of accidents happen on the right side of the road**
- The percentage of accidents that occur in day and night for various states do not vary much and are more or less in the same range for all states
- The box plot for Visibility vs State tells us that there is barely any variation in visibility across States

3. Weather Analysis for US Accidents

Few questions that this analysis/dashboard answers:

- How is the humidity/wind chill/wind speed/temperature/pressure spread for various accidents?

- What are the top weather conditions in which accidents occur?
- What impact does weather condition have on severity of the accident?



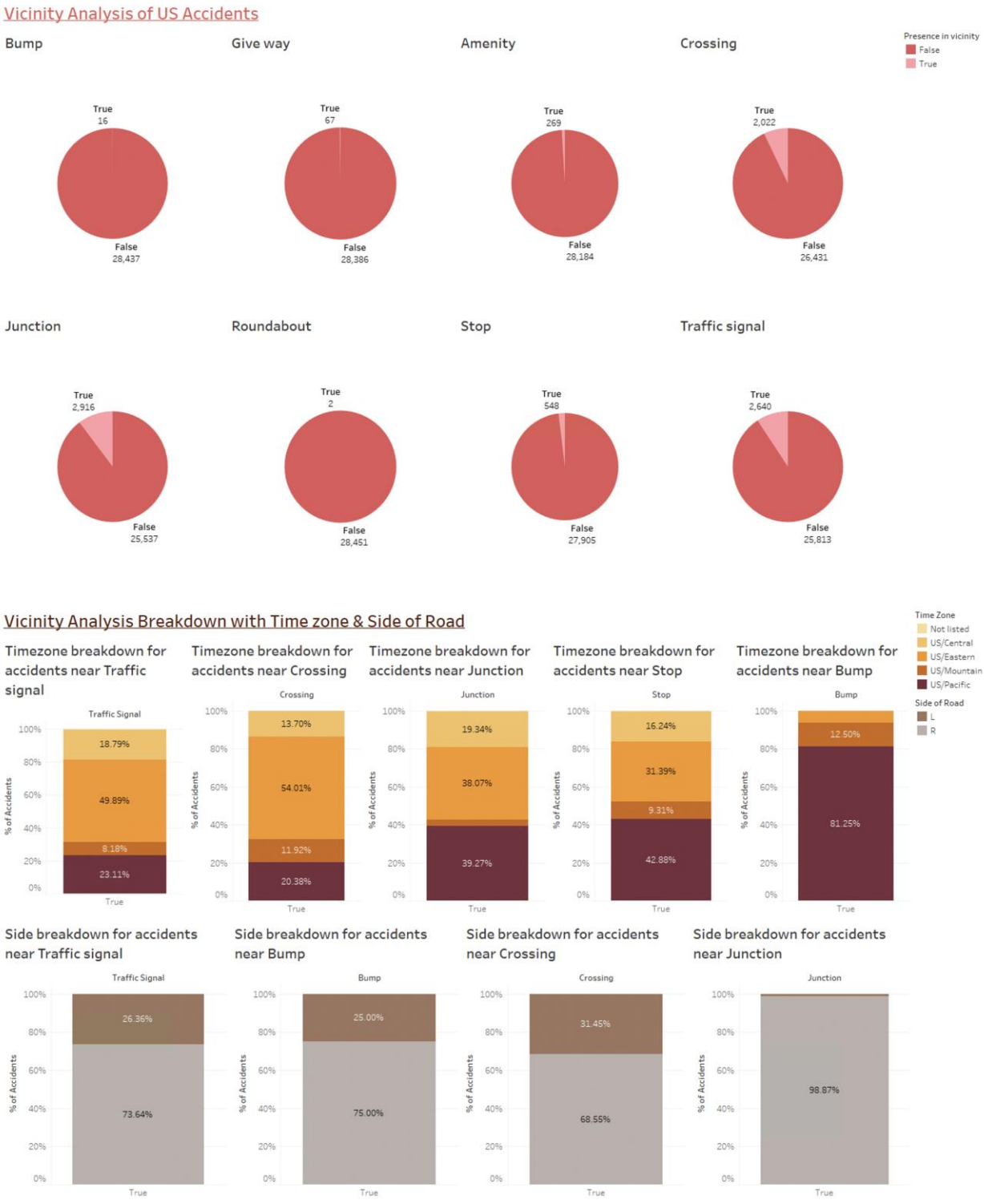
Business insights gathered from this Weather Analysis dashboard:

- Pressure and Visibility do not vary across accidents and severity level
- Most common weather conditions during accidents are ‘Fair’, ‘Mostly Cloudy’ and ‘Cloudy’
- **Windy conditions lead to more severe accidents: Snow/Windy, Thunder/Windy, Haze/Windy and a few other conditions are an example when the percentage of severity level 4 accidents increase compared to other conditions**
- **‘Blowing Snow’ is the most severe weather condition with 100% accidents during this weather being of severity level 4**

4. Vicinity Analysis for US Accidents

Few questions that this analysis/dashboard answers:

- How often do accidents happen near a roundabout? Near a traffic signal? What about a crossing?
- Does the vicinity of a accident impact the severity of the accident?
- Do certain states/time zones have more accidents near a bump? Near a crossing? Traffic signal?
- How are accidents near certain vicinities spread across in terms of period of day (day/night)?



Business insights gathered from this Vicinity Analysis dashboard:

- **There are hardly 1% of accidents that occur near Bump, Give way, Roundabout, Stop or Amenity**

- Around 10% of accidents (each) take place near a Junction, Crossing and Traffic signal
- Of all the accidents that happen near Crossing, 54% occur in the US Eastern time zone. Similarly of all the accidents that happen near a traffic signal, 50% occur in the US Eastern time zone:
    - **US Eastern zone has higher number of accidents occur near Crossing & Traffic Signal as compared to other zones in the US**
- Of all the accidents that happen near Bump, 81% occur in the US Pacific time zone. Similarly of all the accidents that happen near a Stop, 43% occur in the US Pacific time zone:
    - **US Pacific zone has higher number of accidents occur near Bump & Stop as compared to other zone in the US**
- **99% of accidents had occured near a Junction happened on the Right (R) side of the road**

3.7. System Integration and User Acceptance Testing

User Acceptance Testing primarly involved asking a bunch of questions to ouselves:

Is the data present? Does the data that is loaded look correct? Will this fulfill our requirements?

To answer these we check metrics in steps. This involves verifying the total number of nodes, properties & relationships as well as the total number of records with each node. The total number of records within each node in Neo4j will be equal to the number of distinct values present in that particular column in the CSV (found during data profiling) plus any new values that you create (which in our case is –99 or 'Not listed'). So, the number of records should be the number of distinct values plus 1. However, for the central node Accident, the number of records in Neo4j will be equal to the number of total rows in the CSV file. We verified these numbers viewing the data profiling report and the running a cypher query to catch the number of rows in various nodes. The results came out as expected and attached below are a few screenshots.

Total number of observations from data profiling report: 2,845,342

Count of records in the ***Accident*** node from Neo4j: 2,845,342

Total number of distinct values in the ***Distance*** column from data profiling report: 14165

Count of records in the ***TrafficDistance*** node from Neo4j: 14165

Total number of distinct values in the ***Street*** column from data profiling report: 159651

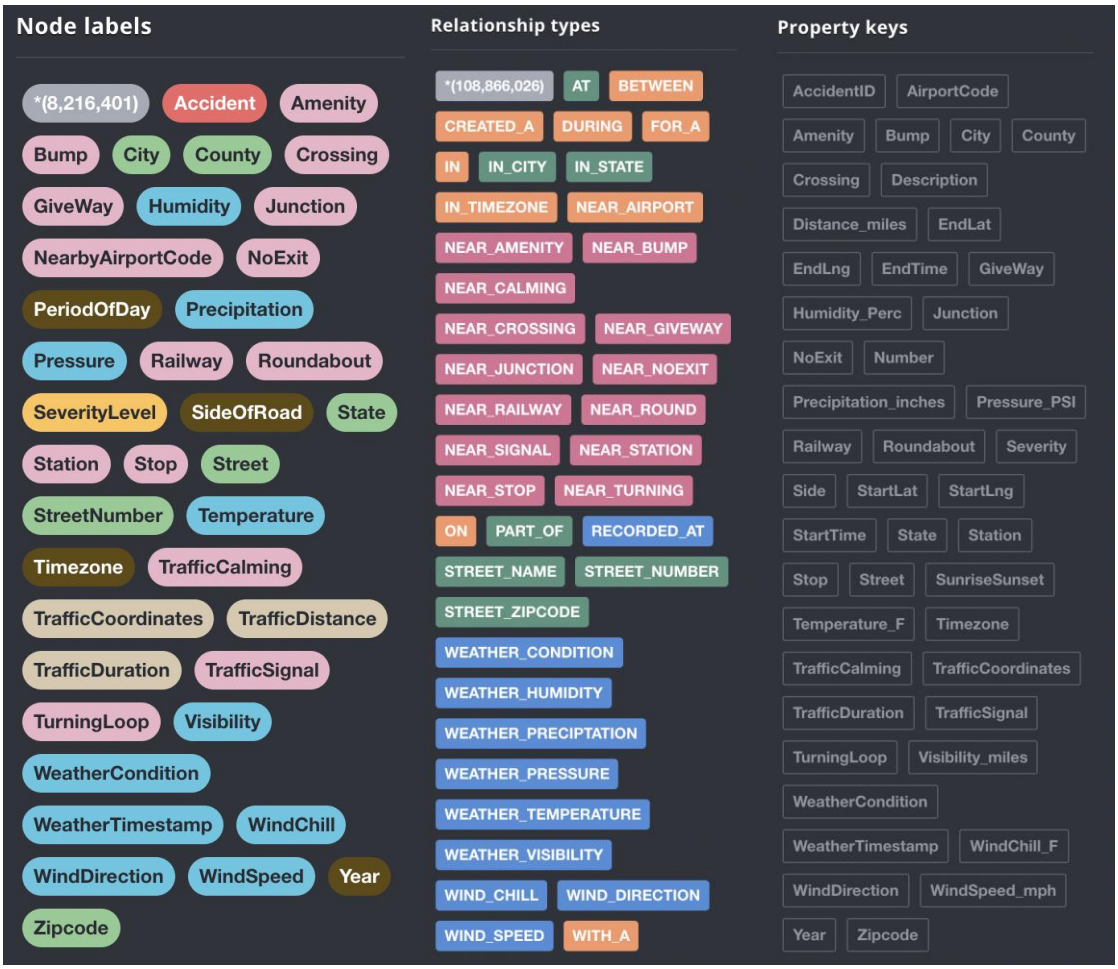
Count of records in the ***Street*** node from Neo4j: 159652 (+1 for the null values)

Overall statistics of the Neo4j graph database:

accidentsdb

Nodes	8216401
Relationships	108866026
Labels	39
Relationship Types	39
Property Keys	46

Viewing the Nodel labels, Relationship types & Property keys as loaded on Neo4j:


Node labels
*(8,216,401)
Accident
Amenity
Bump
City
County
Crossing
GiveWay
Humidity
Junction
NearbyAirportCode
NoExit
PeriodOfDay
Precipitation
Pressure
Railway
Roundabout
SeverityLevel
SideOfRoad
State
Station
Stop
Street
StreetNumber
Temperature
Timezone
TrafficCalming
TrafficCoordinates
TrafficDistance
TrafficDuration
TrafficSignal
TurningLoop
Visibility
WeatherCondition
WeatherTimestamp
WindChill
WindDirection
WindSpeed
Year
Zipcode
Relationship types
*(108,866,026)
AT
BETWEEN
CREATED_A
DURING
FOR_A
IN
IN_CITY
IN_STATE
IN_TIMEZONE
NEAR_AIRPORT
NEAR_AMENITY
NEAR_BUMP
NEAR_CALMING
NEAR_CROSSING
NEAR_GIVEWAY
NEAR_JUNCTION
NEAR_NOEXIT
NEAR_RAILWAY
NEAR_ROUND
NEAR_SIGNAL
NEAR_STATION
NEAR_STOP
NEAR_TURNING
ON
PART_OF
RECORDED_AT
STREET_NAME
STREET_NUMBER
STREET_ZIPCODE
WEATHER_CONDITION
WEATHER_HUMIDITY
WEATHER_PRECIPTATION
WEATHER_PRESSURE
WEATHER_TEMPERATURE
WEATHER_VISIBILITY
WIND_CHILL
WIND_DIRECTION
WIND_SPEED
WITH_A
Property keys
AccidentID
AirportCode
Amenity
Bump
City
County
Crossing
Description
Distance_miles
EndLat
EndLng
EndTime
GiveWay
Humidity_Perc
Junction
NoExit
Number
Precipitation_inches
Pressure_PSI
Railway
Roundabout
Severity
Side
StartLat
StartLng
StartTime
State
Station
Stop
Street
SunriseSunset
Temperature_F
Timezone
TrafficCalming
TrafficCoordinates
TrafficDuration
TrafficSignal
TurningLoop
Visibility_miles
WeatherCondition
WeatherTimestamp
WindChill_F
WindDirection
WindSpeed_mph
Year
Zipcode

3.8. Challenges Encountered

- Deciding on data types was a tricky part because you need to keep in mind the data type mapping from the CSV into Neo4j and then also when you move from Neo4j to Tableau. You need to do the data type conversion initially while keeping in mind all the technologies/tools that this data is going to get into

- Designing the data model. This again needs to be done in multiple iterations because you need to make sure to have a relevant model which can capture all information from your initial data set. Another small challenge that was encountered on this stage is that of graph database. Since the model needs to be a graph database and not a traditional RDBMS model, a sample database was needed to get familiar with this before going ahead with the model designing

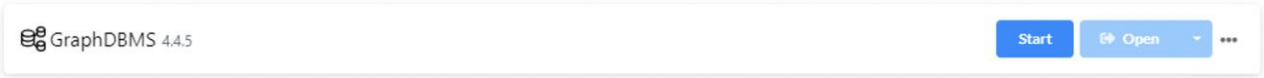
- Data mapping & integration was very challenging due to the volume of the data at hand. The dataset provided has almost 3 million records and running the script of this CSV data made us run into multiple out of memory on our system. Necessary memory accommodations were made before successfully loading the transformed data into Neo4j using the cypher script

- Handling/loading the Neo4j graph data into Tableau in another challenge given the RAM and memory restrictions of our devices. The Neo4j data that we load using the cypher script creates as many as 100 million relationships between the nodes. Connecting to and then loading such a vast amount of data in Tableau (or even Power BI) possesses a deep challenge and causes the software to crash making it impossible to create charts. As a result, we tried to reduce the data that is loaded into Neo4j to make sure it creates less number of relationships in total. Despite numerous efforts at this, the system/software would always hang and crash. To be able to perform the visualization task with the great insights that we wanted to, we had to privision a Virtual Machine (VM) using Azure which allows us to scale the system as much as we want. We used a 32gb RAM instance with 124gb of storage to perform all our tasks. This machine was able to handle all the workload allowing us to do the project with the perfection we wished in the visualization

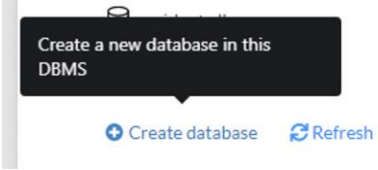
3.9. End User Instructions

The steps followed are as described in the Vision Diagram and as stated thoroughly above in this document. In order for the end user to perform these tasks completely on their own, we describe step-wise procedures in this section.

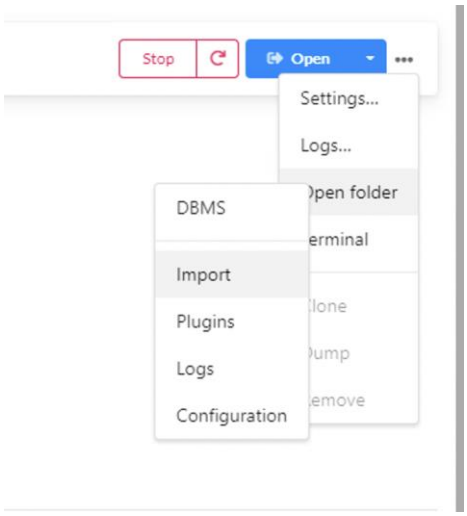
1) Make sure you have downloaded the dataset from here
2) The first thing to do is run the Jupyter notebook that contains Python code for data profiling and cleaning. Running this notebook on the raw CSV data will generate 2 CSVs – the cleaned data CSV and a sampled version of the same cleaned data CSV
3) The notebook mentioned in the above step is named ***"GRP#5 - Data Profiling & Cleansing.ipynb".*** Run this notebook – it reads the dataset you downloaded in step 1) and saves 2 CSVs named - ***"US_Accidents_cleaned.csv"*** and ***"US_Accidents_sampled.csv"***
4) Next, open your Neo4j instance and create a new project by clicking on the "+Add" button. Lets say you name this project as "GraphDBMS". While adding this new database Neo4j will also ask you to set a password for it, lets say that you set the password as "mypass". Click on done/create after this and you will see the project created as shown:



5) Once you have successfully created this project, start it by clicking on the blue start button (as visible in above picture)
6) After it has started, create a new database by clicking the button on the bottom. Lets say you name the database "accidentsdb"

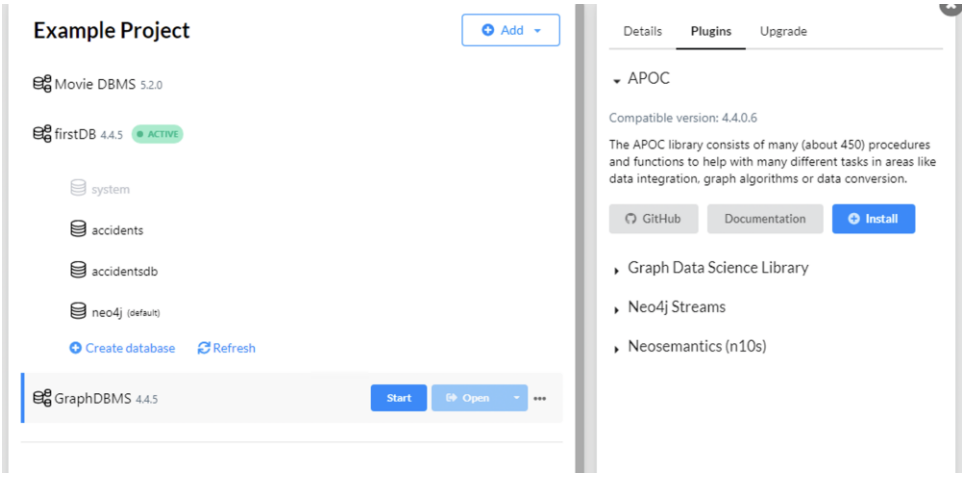


7) Before you move any further, please click on the three dots just beside the blue open button and then select Open folder and within that select Import

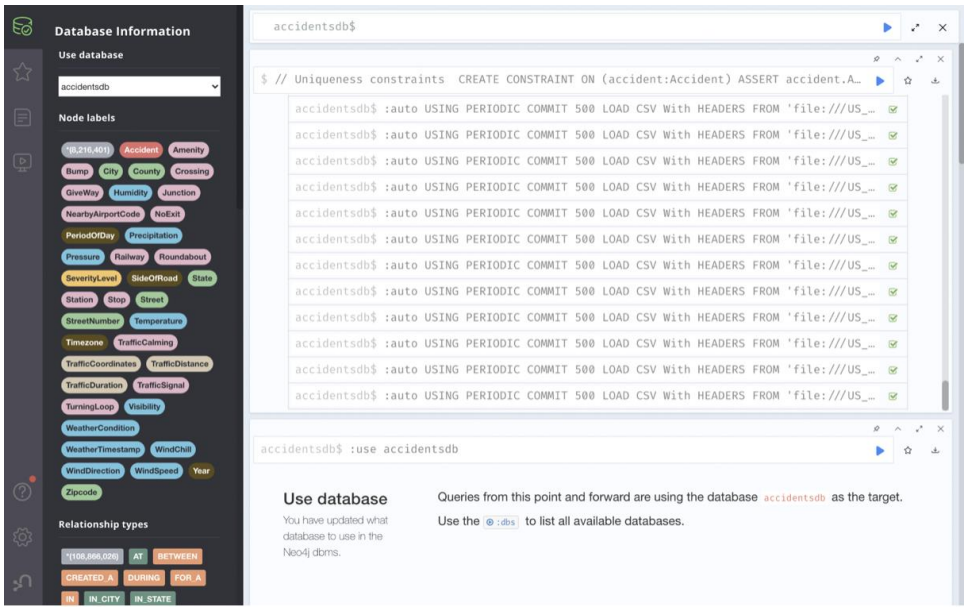


8) This opens a folder, in this folder paste the ***"US_Accidents_sampled.csv"*** filed that was created in step 3). This is the import folder for your Neo4j project and you just added the CSV into this. Now, this CSV will be used by the cypher script that we run

9) One last thing to do before you go ahead to load the graph database is to install the APOC plugin (this is needed to establish the connection with Tableau later on). To do this just click on your project name "GraphDBMS" and this opens a dialog box on the right. Within this dialog box click on Plugins and then install the APOC plugin



10) Finally click on blue Open button to open the Neo4j browser. Make sure to select the "accidentsdb". Next, copy the whole script from the ***"GRP#5 - Cypher script_sampled.txt"*** file and paste it into the query prompt in the Neo4j browser. Then, click on Run and wait for the query to execute



11) Upon successful execution of the script you will see all the nodes lables, relationship types and property keys on the left of the screen as visible in the above picture

12) With this you have loaded the graph DB

13) Next step is to move onto the visualization part. For this, you need to setup the Tableau and Neo4j connection. Please follow the detailed steps mentioned under topic 3.4 Database installation to install and move the JDBC in the mentioned folder; you do not need to make the JDBC URL connection through Tableau yet

14) You could go ahead and open the file ***"GRP#5 - Tableau Workbook (Extracted).twb***" in your Tableau. This should directly open the workbook file. But, if it asks you to connect/select the data extract file then select the ***"GRP#5 - Tableau Data Extract.hyper"*** file

15) You can directly scroll to the end of the list of chart/dashboards now and have a look at the final 6 dashboards for the project which are named as: "Time Analysis of US Accidents", "Location Analysis of US Accidents", "State-City drilldown", "Weather Analysis of US Accidents", "Vicinity Analysis of US Accidents", "Vicinity Analysis Breakdown with Time zone & Side of Road"