UNIVERSITÀ DI PISA

LABORATORY OF DATA SCIENCE REPORT
A.Y. 23 / 24

Exploration of Traffic Incidents (2014-2019)

Matilde Polezzi | Gina Santoro | Eyasu Wolde Wedajo

PART 1

Assignment 1 - Data understanding

The analysis highlighted key relationships between the files through the use of RD NO and identified quality issues, such as duplicates and missing values, which will be addressed in subsequent stages of cleaning and preparation to ensure a reliable database for future analyses.

Assignment 2 - Data cleaning

We have managed the missing values and inconsistencies in the files Crashes, Vehicles, and People, with particular attention to maintaining consistency and reliability of the data without introducing excessive assumptions that could influence subsequent analyses.

File Crashes

- REPORT TYPE: Replace missing values with ”Not on Scene” to reflect the majority of existing data.
- STREET DIRECTION: Imputed cross-checked values: ”S” (South) and ”N” (North).
- STREET NAME: The missing value was imputed as ”76TH ST” on the basis of relationships with other
- columns.
- BEAT OF OCCURRENCE: Replacement based on STREET NAME using consistent matches, associating the
- police department number with specific roads.
- MOST SEVERE INJURY: Filled with ”No Indication of Injuries”, consistent with the context.
- LATITUDE, LONGITUDE, LOCATION: Retrieved missing values using Geopy based on complete addresses from other columns in the CSV.

File Vehicles

- UNIT TYPE: Deleted the only record without matches with RD NO in files People or Crashes.
- VEHICLE ID: For Pedestrian, Bicycle, and Non-Motor Vehicle, left empty. For Driver, replaced with 0.0.
- MAKE, MODEL, VEHICLE DEFECT, VEHICLE TYPE, VEHICLE USE, MANEUVER: For Pedestrian, Bicycle, and Non-Motor Vehicle, missing values were left blank. For drivers, missing values were
- replaced with ”Unknown”.
- LIC PLATE STATE: For Pedestrian, Bicycle, and Non-Motor Vehicle, missing values were left blank. For drivers, missing values were replaced with ”XX”.
- VEHICLE YEAR: For Pedestrian, Bicycle, and Non-Motor Vehicle, missing values were left blank. For drivers, missing or values higher than 2019 were replaced with 0.0.
- TRAVEL DIRECTION: For drivers with missing values, replaced with ”Unknown”.

- OCCUPANT CNT: For Pedestrian, Bicycle, and Non-Motor Vehicle, missing values were left blank.
- FIRST CONTACT POINT: For Pedestrian, Bicycle, and Non-Motor Vehicle, missing values were left blank. For drivers, the column was manually integrated for some specific RD NO: (JB376407: Rear); (JB374941: Side Right); (JB374667: Unknown); (JB374571: Front); (JB374559: Unknown).

File People

- VEHICLE ID: Replaced with 0.0 for Pedestrian, Bicycle, and Non-Motor Vehicle.
- CITY, STATE, SEX: Missing values were filled with ”Unknown”, ”XX”, and ”U” respectively.
- AGE: Inconsistencies were corrected for drivers under 14 years of age, assigning values of 0.0 to represent null values. If AGE is not a valid number, the line was ignored.
- SAFETY EQUIPMENT: Missing values were replaced with ”Usage Unknown”.
- AIRBAG DEPLOYED:
    - For Pedestrian, Bicycle, and Non-Motor Vehicle: replaced with ”NOT APPLICABLE”.
    - For Non-Contact Vehicle: replaced with ”DID NOT DEPLOY”.
    - For Drivers: replaced with ”Deployment Unknown”.
- EJECTION:
    - For Pedestrian, Bicycle, and Non-Motor Vehicle: left blank.
    - For others: replaced with ”Unknown”.
- INJURY CLASSIFICATION: Integrated with ”No Indication of Injuries”.
- DRIVER VISION: Missing values estimated using weather and lighting conditions.
- DAMAGE: Awarded the $500 OR LESS category maximum.
- DRIVER ACTION, PHYSICAL CONDITION: Replaced with ”Unknown”.
- BAC RESULT: Left blank as all missing values corresponded to ”Passenger”.

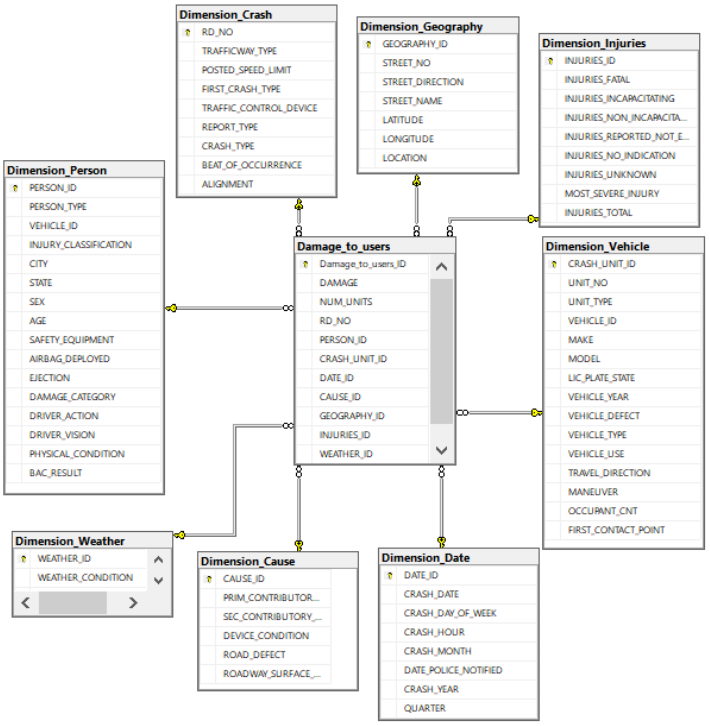
Assignment 3 - Create the DW Schema

The first task involves creating the database schema for the necessary tables using SQL Server Management Studio. The tables were designed, and each attribute was assigned a specific data type.

Assignment 4 - Data Preparation

In Assignment 4, the task consists in the creation of a Python program that splits the content of the different files (’Crashes.csv’, ’People.csv’, ’Vehicle.csv’) and creates other 6 separate tables: Dimension Date, Dimension Geography, Dimension Weather, Dimension Cause, Dimension Injuries, Damage to users.
In the file ***Dimensioni.py***, each dimension is saved in a separate CSV file. The create dimension function was used to create dimensional tables combining data from different sources. The function ensures that combinations of values are unique by using a dictionary, which acts as a cache to verify the existence of a specific combination. When a combination is new, a unique incremental ID is assigned from 1, avoiding ambiguity and making the system more readable.

In the file ***Dimensioni_csv_originali.py***, tables are generated by removing duplicates through the function csv no duplicates, with the output structured in separate CSV files.
In the file ***Damage_to_users.py***, the Damage to users dimension has been created by associating 'RD NO' with the primary keys of the other dimensions. This mapping ensures a clear relationship between the source table data and the dimensions. The code integrates information from pre-generated dimensional tables, matching keys, and adding the corresponding IDs to the fact table. This process is repeated for each relevant dimension, with sequential updates to the final file. We used the same approach for the measurements, matching the correct value based on 'RD NO'.
In the ***Dimension Date:*** The date data type was chosen to conform to standardized date formats, ensuring uniformity and simplifying date-related operations within the database. Employing the date data type guarantees consistent formatting of date entries, enabling precise analysis of temporal data.



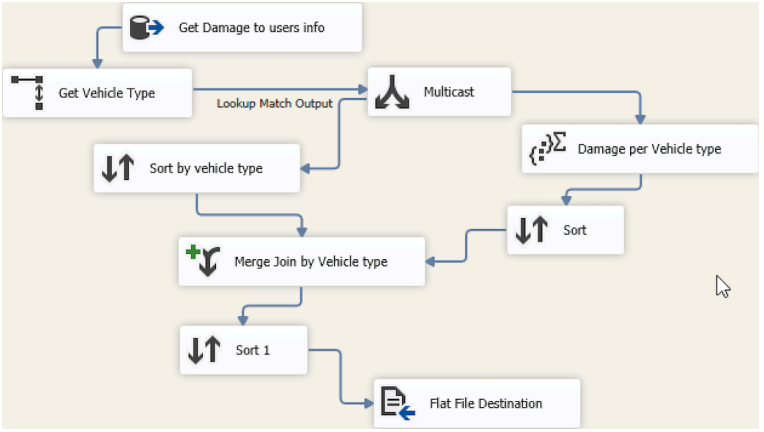
Assignment 5 - Data Uploading with Python

The approach implemented for the database population phase revolves around the use of a single, modular function, populate table from csv batch. This function is designed to dynamically handle batch inserts into SQL Server tables by reading data from CSV files. The code leverages the pyodbc Python library to establish a connection with the SQL Server database and uses a batched execution strategy to optimize performance, thereby significantly improving the overall speed of the data insertion process. Batch Processing: Data is processed in configurable batches, reducing memory usage and improving execution efficiency. By iterating over subsets of the data, the function ensures that large datasets are handled without overwhelming system resources.

Assignment 6 - Data Uploading with SSIS

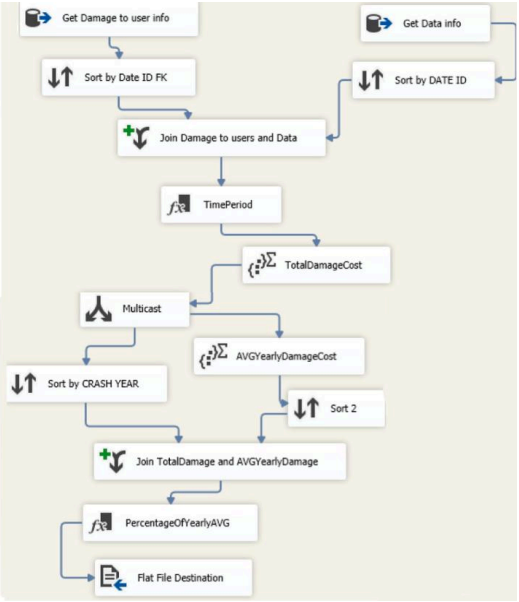
To select the 10% we start from the fact table (Damage_To_Users), once created we use it to create the 8 dimensions using 'lookup' so that only the IDs present in the Damage_To_Users_SSIS are selected.

Assignment 6b



Show all participants ordered by the total damage costs for every vehicle type.

The SSIS workflow aims to display participants sorted by total damage costs grouped by vehicle type. Starting with the fact table, we retrieved Person ID and Crash Unit ID. Using a Lookup, we added the Vehicle Type, sorted it, and simultaneously calculated the total damage. A Merge Join was performed on Vehicle Type to consolidate total damages, followed by final sorting for the flat file output.



Assignment 7b

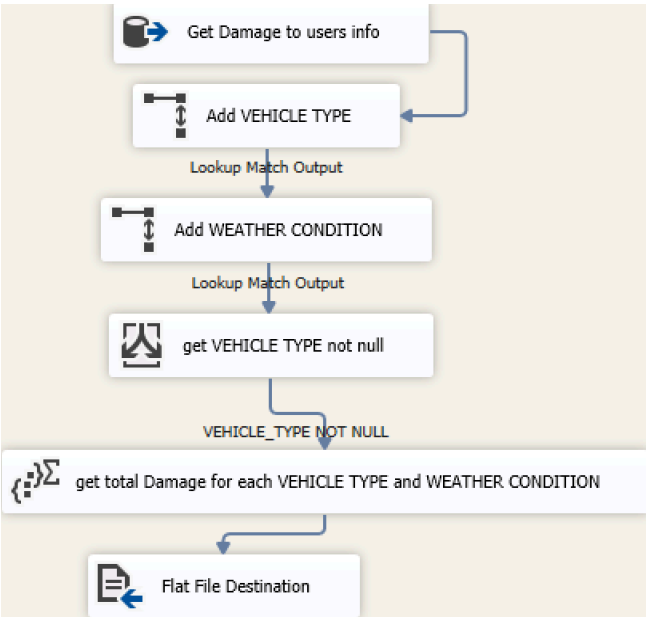
For each month, calculate the percentage of the total damage costs caused by incidents occurring between 9 pm and 8 am and incidents occurring between 8 am and 9 pm, with respect to the average total damage costs for all months within the same year

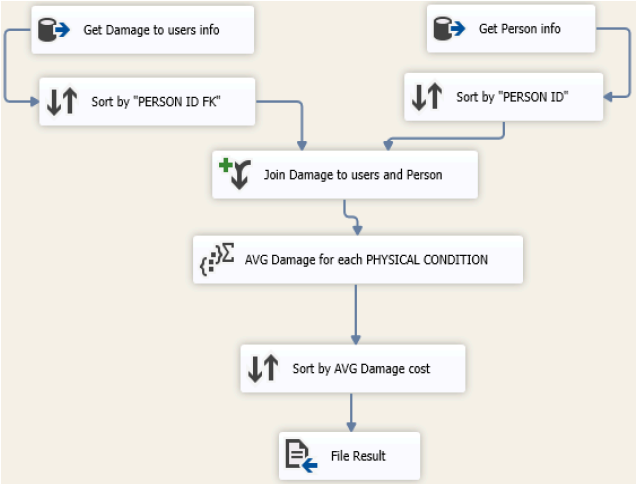
Joined damage and user data to calculate the total damage costs and the yearly average damage costs for each year. Then, you calculated the percentage of total damage costs for incidents occurring between 9pm-8am and 8am-9 pm relative to the yearly average. Finally, the results were sorted and saved to a flat file destination.

Assignment 8b

Show the total crash damage costs for each vehicle type and weather condition.

We created an SSIS flow and imported damage data, added vehicle type information and weather conditions via lookup, and filtered records with vehicle type not null. We then aggregated the total damage by vehicle type and weather conditions. Finally, you saved the results in a file.





Assignment 9b

It calculates the average damage based on the driver's condition, sorting the averages in ascending order.
This SSIS query processes data on damages and driver information. It joins the datasets using the person ID, calculates the average damage for each physical condition of the driver, sorts the results in ascending order by average damage, and exports the output to a file. We chose this query to see which condition causes the most damage

PART 2

Assignment 1 - Build a datacube

To create the data cube, we chose to retain all attributes for each dimension rather than combining only those required to address the specific business question. This approach ensures the system can accommodate potential future business inquiries providing flexibility for unforeseen analytical needs.

- **DATE:** We used DATE_ID as the primary key and created a hierarchy starting from the most general level (DATE_YEAR) to the most specific level (CRASH_DAY_OF_WEEK). All column attributes were organized to allow flexible navigation within the hierarchy, supporting analyses ranging from annual trends to specific daily incidents.
- **CAUSE:** For this dimension, CAUSE_ID was used as the primary key. A hierarchy was established, ranging from the most general level (SEC_CONTRIBUTORY_CAUSE) to the most specific level (PRIM_CONTRIBUTORY_CAUSE).
- **GEOGRAPHY:** We use GEOGRAPHY_ID as the primary key, the most specific level of granularity is defined by LOCATION.
- **PERSON:** We used PERSON_ID as the primary key and created a hierarchy limited to the attributes STATE and CITY, proceeding from the most general to the most specific level.

Measures: The only measures used were those automatically generated by the system during the creation of the data cube. We defined two measures:

- ***DAMAGE:*** Damage costs reimbursement each client incurs for each crash event.
- ***NUM_UNITS:*** Represents the number of units involved in each incident.

Query MDX

Assignment 2

For each month, show the total damage costs for each location and the grand total with respect to the location.

To calculate the total damage costs for each month and each location, two members were defined. The first, *[Measures].[MonthlyDamage]*, contains the total damage for each month. This value was calculated using the IIF function, which checks if the damage value is null. When the value is null, it is replaced with zero; otherwise, the actual value is used. This approach is crucial to avoid calculation errors caused by missing data. The second member, *[Measures].[TotalDamageByLocation]*, represents the overall total damage for each location by summing up the values across all considered months. This sum was obtained using the SUM function applied to the *[Dimension Date].[Hierarchy].[CRASH MONTH*] hierarchy, enabling the aggregation of damages for each location. The *NONEMPTY* function was used to filter the query result, ensuring that only combinations of months and locations with valid data are displayed. This eliminates rows with null values in the output, enhancing readability and clarity.
In cases where data is missing for a specific combination of month and location, the query result shows null values. Handling missing data through the nullity check in *[Measures].[MonthlyDamage]* prevents errors and ensures consistency in the calculations.

Assignment 4

For each location, show the damage costs increase or decrease, in percentage, with respect to the previous year

This query calculates the percentage change in damage costs for each location compared to the previous year. The cumulative damage for the current year, *[Measures].[YearDamage]*, is computed using the SUM function over the range of dates provided by *PERIODSTODATE*. Similarly, the cumulative damage for the previous year, *[Measures].[YearDamagePrev]*, uses the same logic but applies it to the PrevMember of the year hierarchy. Both calculations include checks for null or zero values to ensure robust results.
The absolute difference between the two years is represented by *[Measures].[DiffDamage], while [Measures].[DiffPercDamage]* calculates the percentage change, handling cases where the previous year's damage is null or zero by returning *NULL*.
The *NONEMPTY* function filters the results to include only locations and years with valid data, ensuring a clean and concise output. This structure highlights year-over-year trends in damage costs for each location, with a focus on percentage variations, providing a comprehensive view of increases or decreases.

Assignment 5

For each quarter, show all the locations where the number of vehicles involved exceeds the average number of vehicles involved in the corresponding quarter of the previous year. Also, report the increase in both percentages.

We use [*Measures].[AvgVehiclesPrevYear]* computes the average number of vehicles involved in crashes during the same quarter of the previous year using the ParallelPeriod function. Any missing data is handled using the *CoalesceEmpty* function, which replaces NULL values with 0, ensuring robust and accurate calculations.
The percentage increase is calculated as *[Measures].[PercIncreaseVehicles]*, which divides the difference between the current number of vehicles *([Measures].[NUM UNITS])* and the average from the previous year ([Measures].[AvgVehiclesPrevYear]) by the average, and multiplies the result by 100. If the average from the previous year is 0, the measure returns NULL to avoid division errors.

The query filters results using *NONEMPTY* combined with a FILTER condition. It includes only those rows where both *[Measures].[NUM UNITS]* and *[Measures].[AvgVehiclesPrevYear]* are greater than 0 and ensures that the current number of vehicles is greater than the calculated average. This filtering ensures that only meaningful and relevant data is presented.

Assignment 7

For each location, it shows the percentage of damage associated with bad weather (as primary cause) compared to total damage, and calculates the average damage from bad weather-related accidents.

To achieve this, *[Measures].[TotalDamageByLocation]* calculates the aggregated total damage for each location, iterating over all location members and handling missing values by treating them as zero. *[Measures].[WeatherDamage]* isolates and sums the damages specifically caused by incidents where weather is identified as the primary contributory cause, ensuring accurate aggregation by filtering for the relevant cause.

The percentage of weather-related damage is computed as *[Measures].[WeatherDamagePercentage]*, which divides the total weather-related damage by the total damage for each location and multiplies the result by 100. If a location has no recorded total damage, the percentage is set to NULL to avoid misleading results.

The query uses *NONEMPTY* to filter out locations with no weather-related damage, ensuring that only relevant data is included in the results. The output displays locations in rows alongside the calculated measures for total weather-related damage and its percentage impact on the overall damage.

This analysis provides insights into the extent of damage caused by weather across different locations, helping identify areas most affected and informing strategies for targeted interventions.

Assignment 8

For each year, show the most frequent cause of crashes and the corresponding total damage costs. The primary crash contributing factor is given twice the weight of the secondary factor in the analysis. Additionally, show the overall most frequent crash cause across all years.

This query calculates the most frequent cause of crashes for each year by assigning a higher weight to primary contributory causes compared to secondary contributory causes. Specifically, primary causes are given a weight of 2, while secondary causes are given a weight of 1. The measure [Measures].[PRIM CAUSE Frequency] computes the frequency of primary causes by counting all rows associated with a given primary cause for each year. Similarly, [Measures].[SEC CAUSE Frequency] calculates the frequency of secondary causes. These two measures are combined in [Measures].[WEIGHTED Freq], which represents the weighted frequency by applying the respective weights to the primary and secondary causes.

To identify the most frequent cause for each year, the set [Most Frequent Cause Per Year Set] is defined. This set uses the TopCount function to select the cause with the highest weighted frequency for a given year, excluding causes labeled as "All". The name of this most frequent cause is then retrieved by the measure [Measures].[Most Frequent Cause Per Year], which generates the corresponding cause name for each year.

The query also calculates the total damage costs associated with the most frequent cause for each year. This is accomplished using [Measures].[TOT Damage for Most Freq per Year], which applies the Sum function to aggregate the damage values for the cause identified within the year, the sum ensures that each line contributes correctly to the total, considering each individual occurrence.

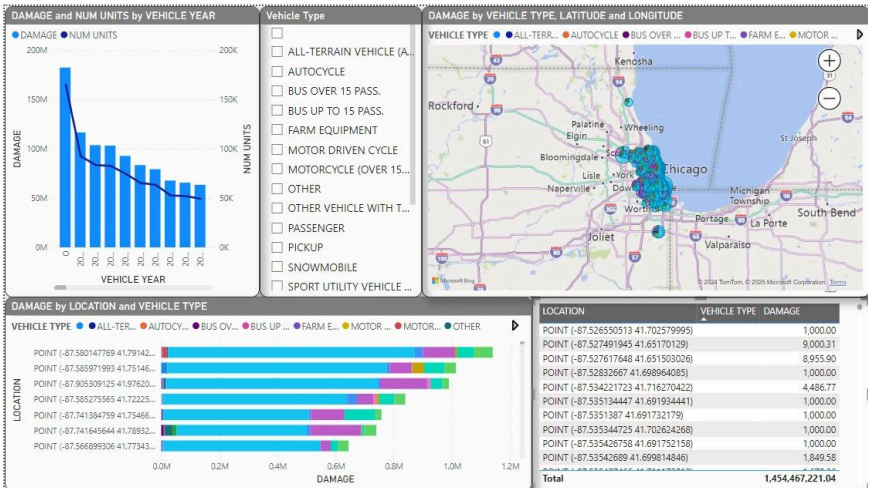
For the analysis across all years, the query determines the overall most frequent cause using [Measures].[Most Frequent Cause Overall]. This measure sums the weighted frequencies of each cause across all years and identifies the top cause, again excluding categories labeled as "All", this sum ensures that the primary cause contributes correctly with a weight of 2 while the secondary one contributes correctly with a weight of 1 obtaining a weighted calculation precise. Additionally, the total damage costs for this overall most frequent cause are calculated.
Throughout the query, NONEMPTY filtering is employed to ensure that only relevant data is included in the calculations, improving precision by excluding rows with null or non-pertinent values. Filters are applied to refine the results further, ensuring accurate identification of the most frequent causes and their associated damage costs.

Power BI

Regarding the reporting phase, the Power BI tool has enabled, through its visual component, the representation of a key aspect that concludes this exploration.

Assignment 9

Create a dashboard that shows the geographical distribution of the total damage costs for each vehicle category.

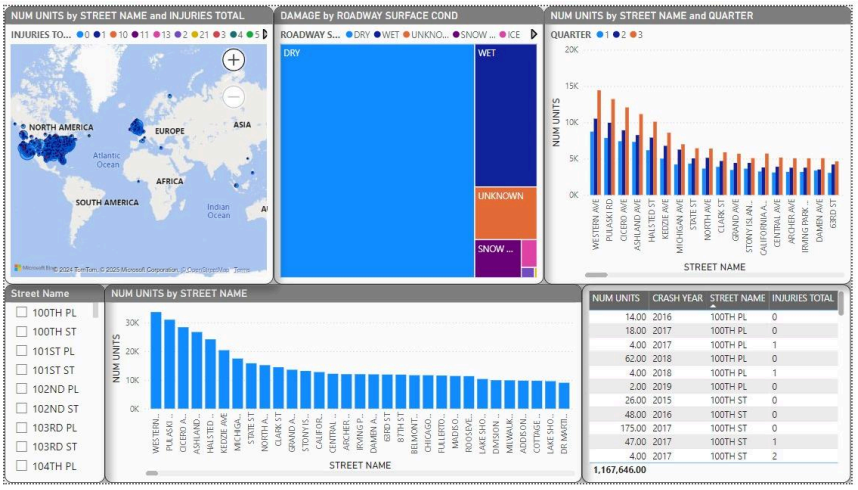


The first graph demonstrates a clear correlation between vehicle age and damage costs, with older vehicles being consistently associated with significantly higher damage expenses. This trend is evident across all categories and is paired with a steady decrease in the number of units involved in accidents as vehicles age, suggesting reduced frequency but greater severity of damages for older vehicles.
To enable more specific analysis, a central filtering panel has been incorporated, allowing users to select particular vehicle types. This feature makes it possible to focus on the impact of individual categories, offering deeper insights into how each contributes to the overall damage landscape.
The map visualizes the geographical distribution of damage costs by vehicle category, effectively highlighting the localized nature of most accidents. The distribution aligns closely with population density and traffic patterns, pinpointing the exact locations where accidents occur and identifying high-risk areas.
An analysis of damage by vehicle type and location reveals that passenger vehicles dominate the data. This dominance is consistent regardless of geographic location, further emphasizing their disproportionate contribution to total damage costs. This insight underscores the critical role of passenger vehicles in shaping the financial impact of accidents.
The accompanying data table provides a detailed numerical breakdown of damage costs, vehicle types, and precise geographic coordinates. This enhances the accuracy and traceability of information presented in the dashboard visuals, enabling users to explore the data in greater detail. The total

damage cost, shown at the bottom, offers a sense of the scale of the issue, amounting to an astounding $1.454 billion. This figure highlights the immense financial burden associated with vehicle damage in the Chicago area and underscores the importance of addressing these challenges through data-driven policy and planning.

Assignment 10

Create a plot/dashboard that you deem interesting w.r.t. the data available in your cube, focussing on data about the street.



The analysis begins by mapping the global distribution of accidents, focusing on individual routes to identify high-risk areas and the total injuries they cause. This spatial representation allows for the identification of accident-prone zones and provides a basis for targeted interventions.

The treemap visualizes damage based on road surface conditions, revealing that the majority of accidents occur on dry surfaces, followed by wet conditions. Accidents on snow or ice, as well as those on roads with unknown surface conditions, make up a much smaller proportion. These insights highlight the critical role environmental factors play in influencing the frequency and severity of accidents.

The bar chart categorizes the number of units involved in accidents by street name and quarter. Prominent streets such as Western Ave, Pulaski Rd, and Cicero Ave stand out with the highest number of units involved, demonstrating notable fluctuations across various neighborhoods. This data underscores specific urban hotspots that may warrant prioritized safety interventions.

A secondary bar chart provides a consolidated view of the total number of units involved by street name, reinforcing the observations from the previous chart. Once again, streets like Western Ave and Pulaski Rd dominate the dataset, with a noticeable decline in accident counts for other streets.

Finally, the data table in the bottom right corner offers a detailed breakdown of the dataset, listing the number of units, crash year, street names, and total injuries for specific accidents. This tabular representation ensures traceability and offers numerical evidence to complement the visual insights. The total number of units involved, an impressive 1,167,646, reflects the scale of the dataset and emphasizes the high urban density of incidents captured.

Assignment 11

Create a plot/dashboard that you deem interesting w.r.t. the data available in your cube, focussing on data about the people involved in a crash.

The treemap categorizes damages by driver actions, revealing that the majority of accidents fall under the classifications of "Unknown" or "None," which highlights significant gaps in reporting. Among recorded actions, behaviors such as "Failure to Yield" and "Improper Turns" stand out as major

contributors to damage, pointing to areas where improved driver education and stricter enforcement could reduce accidents.

The combined bar and line chart examines crash trends across months, segmented by gender. A clear seasonal pattern emerges, with accidents and associated damages peaking in the latter half of the year, particularly among males. This trend likely correlates with external factors such as seasonal weather variations, changing road conditions, or heightened activity during the holiday period.

The horizontal bar graph delves into the number of units involved by person type and injury classification. Drivers dominate this data, with a notable presence in both fatal and non-fatal accident categories. Passengers follow as the second-largest group, while pedestrians and cyclists, despite lower numbers, represent critical cases often linked to urban accidents and increased vulnerability.

Further analysis of age groups, person types, and gender highlights that adult drivers, both male and female, overwhelmingly dominate the dataset. Pedestrians and cyclists, though fewer in number, maintain a significant presence, underscoring their susceptibility in urban mobility contexts.

A breakdown of damage costs by gender shows that accidents involving males account for the highest total damages ($765.34M), followed by females ($554.50M). A smaller portion of the total is attributed to individuals of unknown or mixed gender, further reflecting the need for improved demographic accuracy in reporting.

The line graph addressing safety equipment and airbag deployment illustrates a striking peak in damages associated with accidents where airbags were deployed. This trend reinforces the critical role of safety measures in mitigating accident severity, even in cases of significant impact.

The total damages presented in the dashboard emphasize the extensive financial and societal cost of road accidents involving individuals. Together, these visualizations provide a foundation for analyzing behavioral and situational factors that influence accident outcomes. Key patterns in driver actions, injury classifications, and safety equipment usage are particularly noteworthy, highlighting the role of demographic and behavioral insights in addressing road safety challenges.