Building a Data Warehouse for Road Safety: The Case of Chicago

Arcangelo Franco, *Data Science & Business Informatics*, *584174*, *University of Pisa*;
Nicola Pastorelli, *Digital Humanities*, *656431*, *University of Pisa*

December 31, 2024

Abstract. *This report was prepared as part of the "Decision Support Systems: Laboratory of Data Science" course for the academic year 2024/2025, and documents a comprehensive analysis of traffic incidents data in Chicago, using data cleaning techniques, data warehouse modelling and multidimensional queries. The analysed datasets include information on incidents, people involved and vehicles, with challenges related to missing values and anomalies. Conceptual and logical schemas were implemented for a data warehouse, which was subsequently populated using Python and SSIS. The construction of a datacube enabled complex analytical questions to be answered, such as the analysis of incidents causes, associated damage costs and seasonal trends. Finally, interactive dashboards were developed to visualise the geographical distribution of costs, the impact of road and weather conditions, and the demographic profiles of the individuals involved. These tools provide useful insights to improve road safety and optimise traffic management policies.*

1. Part 1

1.1. Assignment 1: data understanding

This section explores a collection of data on traffic accidents in Chicago, with the aim of analysing the variables involved and understanding their interrelationships. The study focused on three main datasets. The first, called 'Crashes', provides details on accidents, including date, geographical location and weather conditions, with a total of 257,925 rows and 36 columns. The second dataset, 'People', collects information on the people involved, such as physical condition, sobriety status and nationality, and has 564,565 rows and 19 columns. Finally, the 'Vehicles' dataset contains details on vehicles, such as make, type and defects, with 460,437 rows and 17 columns.

In the 'Crashes' dataset, correlation analysis revealed an interesting relationship between the variables 'beat_of_occurrence', 'latitude' and 'longitude'. In particular, a positive correlation of 0.59 was observed between 'beat_of_occurrence' and 'latitude', and a negative correlation of -0.49 between 'beat_of_occurrence' and 'longitude'. These relationships are represented in the correlation matrix in Figure 1.

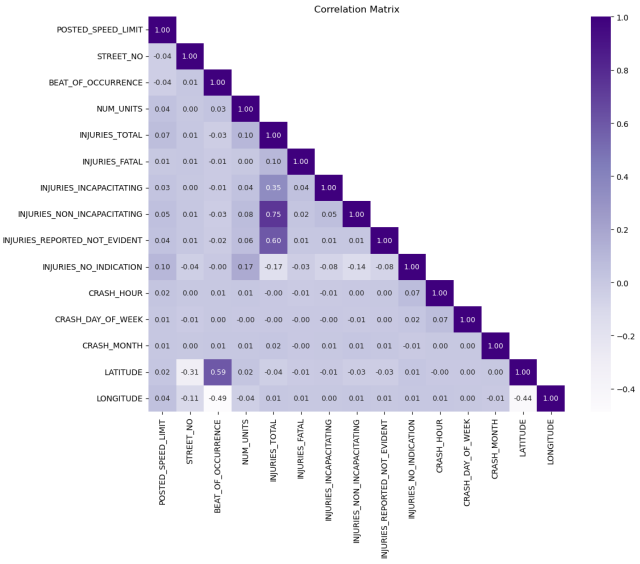

Figure 1. Correlation Matrix of dataset "Crashes"

Upon further investigation, it was identified that 'beat_of_occurrence' corresponds to the districts of Chicago. With this information, missing 'latitude' and 'longitude' values were imputed using the centroids of the multipolygon coordinates associated with the districts, derived from the official dataset *'PoliceBeatDec2012_20241126.csv'* obtained from the City of Chicago Data Portal[1].

[1] `https://data.cityofchicago.org/Public-Safety/Boundaries-Police-Beats-current-/aerh-rz74`

The 'People' dataset has a significant amount of missing values in crucial variables. Amongst these are: 'city' (25.49%), 'state' (25.14%), 'age' (28.51%), 'driver_action' (19.44%), 'driver_vision' (19.46%), 'physical_condition' (19.39%), 'bac_result' (19.27%) and 'damage' (13.16%). In addition, inconsistencies were found in the variable 'city', which indicates the city of birth of the persons involved. Incorrect values such as 'Chyicago,' 'Chicago+,' 'Chgo,' 'Chucago,' as well as entries such as 'Unknow,' 'Unk,' 'Un' and 'Un Known' were found.

The 'Vehicles' dataset is also critical, with significant data gaps: 10.77 per cent of the values in the variable 'lic_plate_state' and 19.25 per cent in 'vehicle_year' are missing. The latter has particular anomalies: some entries refer to years before the invention of the first car, while others refer to dates well beyond the period covered by the dataset.

Furthermore, inconsistencies were found in the variables relating to vehicle 'make' and 'model', with identical values in many rows, especially in the variable 'make' (vehicle manufacturer). This suggests that, at the time of the accident, the vehicle 'model' may not have been recorded or that the information provided was incomplete.

The correlation matrix analysis of the 'Vehicles' dataset revealed a correlation of 1.0 between 'vehicle_id' and 'crash_unit_id', suggesting a direct and unique relationship between vehicle identifiers and accident units.

1.2. Assignment 2: data cleaning

The data cleaning process aimed to improve the quality of the data through various operations: handling missing values, correcting anomalies and inconsistencies, enriching with new calculated variables and normalising certain fields.

The first dataset processed was 'Crashes'. The first step was to replace some outliers, such as in the 'posted_speed_limit' column, where values above a certain threshold were reduced to 70, while those below another threshold were increased to 20[2]. Subsequently, the missing values in the 'report_type' column were replaced with 'UNKNOWN', while in the 'street_direction' column they were set to 'U', trying to maintain consistency with the format of the string. Missing values in the 'most_severe_injury' column were set to 'NO INDICATION OF INJURY'. In addition, the numeric values in the column 'crash_day_of_week', which represented the days of the week, were mapped to corresponding strings, e.g. '1: Monday'.

As described in the previous chapter, 'latitude' and 'longitude' geographic coordinates were imputed by calculating the centroids of the multi-polygon coordinates from the official City of Chicago Data Portal dataset. In order to avoid over-simplification, missing values were imputed only when the 'beat_of_occurrence matched' that in the external dataset, and a new 'location_point' column was generated in which the latitude and longitude variables were used to generate a GEOHASH code referring to them.

In terms of structural changes, columns such as 'injuries_unknown' (removed due to zero variance) were removed. In addition, the column 'crash_date' was split into several components, creating six new columns: 'crash_month', 'crash_day', 'crash_year', 'crash_time', 'crash_period' (indicating AM or PM) and 'crash_season', which indicates the season of the accident.

Columns such as 'trafficway_type', 'prim_contributory_cause' and 'road_defect' were normalised, removing unnecessary inverted commas, parentheses and symbols. Four new unique identifier columns were also added: 'crash_id', 'date_id', 'location_id' and 'injury_id'.

The 'People' dataset was processed by replacing missing values in columns such as 'city', 'safety_equipment', 'airbag_deployed' and 'driver_action' with 'UNKNOWN'. Missing values in 'state' were replaced with 'XX' and in sex with 'X'. Missing values in the 'age' column have been replaced with the mean of the distribution and in 'bac_result' have been filled with 'TEST NOT OFFERED'.

For the 'injury_classification' column, values were calculated by analysing the data in the 'Crashes' dataset and identifying the highest numerical value corresponding to the most severe injury.

In relation to inconsistencies concerning the 'city', the column was standardised using an external dataset (UScities.csv)[3]. The **Levenshtein Distance** was used to identify the city most similar to the value in the dataset, also correcting some states that did not match the city.

Finally, the fields 'airbag_deployed', 'driver_vision' and 'damage_category' were normalised by removing punctuation and brackets. A new column for identifiers has been added: 'person_id'.

[2] https://www.isp.illinois.gov/TrafficSafety/SpeedLimitEnforcement

[3] https://simplemaps.com/data/us-cities

In the 'Vehicles' dataset, missing values in the variables 'make' and 'model' were replaced with 'UNKNOWN' and in 'lic_plate_state' with 'XX'. Outliers in 'vehicle_year' were normalised to 1886 and 2018, while in 'occupant_cnt' the missing values were replaced with the mean of the distribution.

Again, the variables 'make' and 'model' were cleaned of parentheses, symbols and irrelevant content. The values 'UNKNOWN/NA' were replaced with 'UNKNOWN' to maintain consistency with the data. Finally, a new column was added for the identifier: 'vehicle_id'.

1.3. Assignment 3: DW schema

In order to create the data warehouse, first the conceptual schema was designed, i.e. the dimensional fact model. In this schema, the fact is the damage related to a person involved in a crash; the measures are the damage cost and the number of vehicles involved in the crash ('num_units'); the dimensions are crash, date, location, injury, person and vehicle, each with their own dimensional attributes. The fact, measures and dimensions were obtained by analysing the business questions of successive assignments, while the dimensional attributes were obtained from the 3 csv files extracted from the operational database.

Once the conceptual schema of the data warehouse was designed, it was transformed into the logical schema, i.e. the star schema (Figure 2). This schema is made up of 7 tables: 6 dimensional tables, corresponding to the dimensions of the conceptual schema, each with a surrogate primary key and its own attributes, and a fact table ('damage'), containing the measures of the conceptual schema and the foreign keys that relate to the primary keys of the dimensional tables. These tables were created on the server by calling the function 'create_schema' in the file 'main.py'.

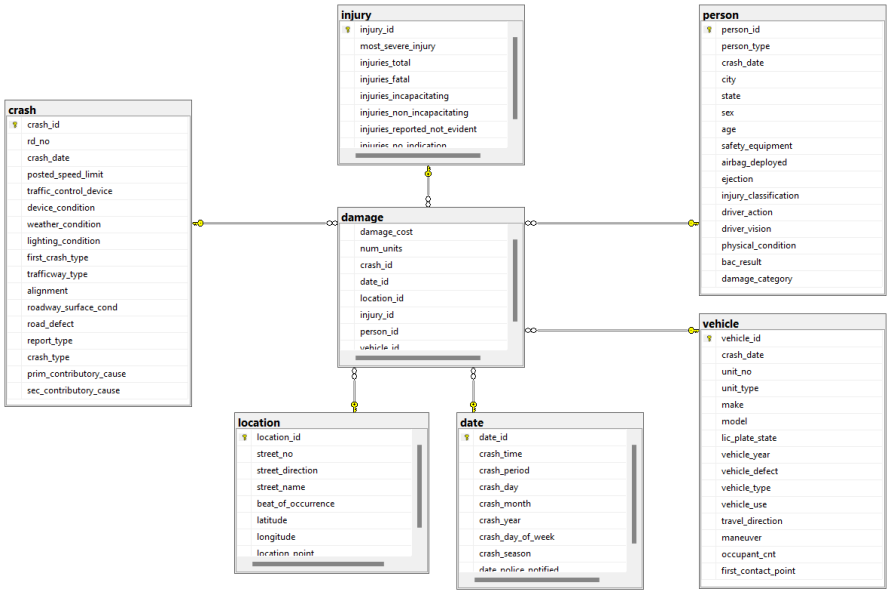

Figure 2. Star Schema

1.4. Assignment 4: data preparation

After creating the data warehouse, 7 csv files were created, one for each table in the data warehouse, by calling the function 'generate_starschema_files' in the file 'main.py'.

To create the 6 csv files corresponding to the dimension tables, the 3 csv files processed in assignment 2 were used: the csv file 'crashes_cleaned' (containing the surrogate keys 'CRASH_ID', 'DATE_ID', 'LOCATION_ID' and 'INJURY_ID') was split into the csv files 'crash', 'date', 'injury' and 'location', while the csv files 'people_cleaned' and 'vehicles_cleaned' (containing the surrogate keys 'PERSON_ID' and 'VEHICLE_ID' respectively) were derived into the csv files 'person' and 'vehicle' respectively.

Finally, in order to create the csv file corresponding to the fact table, the two csv files 'crashes_cleaned' and 'people_cleaned' were merged using 'RD_NO' as a common column; the resulting dataset was in turn merged with the csv file 'vehicles_cleaned', again using 'RD_NO' as a common column; the final merged dataset was derived in the csv file 'damage', containing (in addition to the measures) all surrogate keys used as foreign keys.

1.5. Assignment 5: data uploading with python

At this point, each table in the data warehouse was populated with the data contained in the csv file corresponding to that table. Specifically, first the dimensional tables were populated, then the fact table was populated. The populating process was performed by calling the function 'populate_database' in the file 'main.py'.

1.6. Assignment 6: data uploading with SSIS

This assignment was carried out entirely in the SSIS package 'assignment_6'. In the control flow of the package (Figure 3), first an SQL query was defined that duplicates each table in the data warehouse by replicating its attributes and keys, but without the records. Then, in the control flow of the same package, 7 data flows were created that populate the duplicate tables with 10% of the data of the original tables.

The first data flow is the one which populates the 'crash_ssis' table (Figure 4): the source of the flow is the table 'damage', of which only the column 'crash_id' was selected; a percentage sampling of 10% was performed on the records of the table 'damage' by specifying a fixed random seed; a sorting was then performed on the column 'crash_id', in order to remove the rows with duplicate values; subsequently, a join was performed with the table 'crash' through a lookup, using the common column 'crash_id'; finally, the table 'crash_ssis' was selected as the destination of the flow.

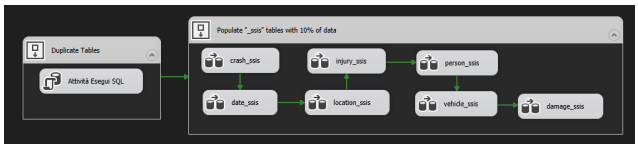

Figure 3. Control flow

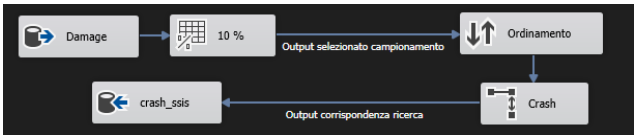

Figure 4. Data flow

The other data flows follow the same logic, with the exception of the last flow, i.e. the one populating the table 'damage_ssis': the origin of the flow is the table 'damage', of which all the columns were selected this time; on the records of the table 'damage' a percentage sampling of 10% was performed by specifying the same random seed used in the previous flows; finally, the table 'damage_ssis' was selected as the destination of the flow.

1.7. Assignment 6b

"Show all participants ordered by total damage costs for each vehicle type."

To answer this question, the data flow in Figure 5 was implemented: the source of the flow is the table 'damage', of which the columns 'damage_cost', 'person_id' and 'vehicle_id' were selected; through a lookup, a join was performed with the table 'vehicle', in order to retrieve the column 'vehicle_type'; then, through a second look-up, a join was performed with the table 'person', in order to retrieve the columns 'city', 'state', 'sex' and 'age'; a sorting was then performed on the column 'damage_cost' in descending order; finally, a csv file containing the results of the query was selected as the destination of the flow.

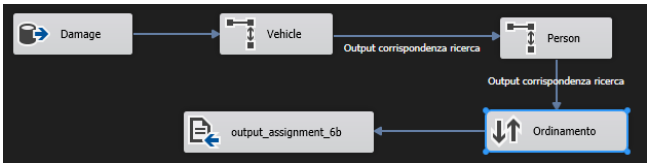

Figure 5. Data flow 6b

1.8. 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."

To answer this question, two data flows were implemented. The first data flow calculates, for each month, the total damage costs caused by incidents occurring between 9 pm and 8 am and incidents occurring between 8 am and 9

pm. The second data flow calculates, for each month, the percentage of the total damage costs with respect to the average total damage costs for all months within the same year.

In the first data flow (Figure 6) the source is the table 'damage', of which the columns 'damage_cost' and 'date_id' were selected; through a look-up, a join was performed with the table 'date' in order to retrieve the columns 'crash_time', 'crash_period', 'crash_month' and 'crash_year'; then a conditional split was performed which, based on a condition on the 'crash_time' and 'crash_period' columns, divided the data into two branches ('Day' and 'Night'); in each branch, the sum on the column 'damage_cost' was calculated by grouping on the basis of the columns 'crash_month' and 'crash_year'; furthermore, in each branch, a derived column called 'crash_period' was added whose values are 'DAY' or 'NIGHT' depending on the branch; after merging the two branches, a sorting was performed on the columns 'crash_year', 'crash_month' and 'crash_period' in ascending order; finally, a csv file containing the results was selected as the destination of the flow.

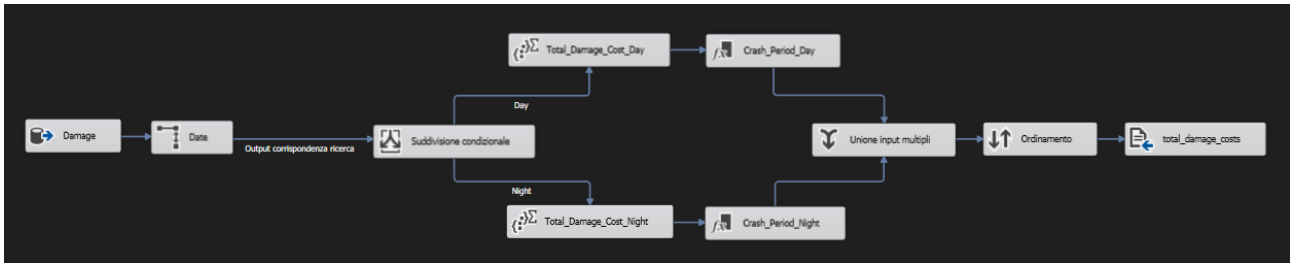

Figure 6. First data flow

This csv file was selected twice as the origin of the second flow (Figure 7); on the first origin the sum on the column 'total_damage_cost' and a count distinct on the column 'crash_month' were calculated by grouping according to the column 'crash_year'; a derived column called 'average_damage_cost' was then added by dividing the year damage cost by the number of months; then, a sorting was performed on the column 'crash_year' in ascending order; on the second source, a sorting was performed on the column 'crash_year' in ascending order, also selecting the columns 'crash_month', 'crash_period' and 'total_damage_cost'; at this point, a merge join was performed between the two sources using 'crash_year' as the key; a derived column called 'percentage' was then added by dividing the total damage cost by the average damage cost and multiplying by 100; then, the columns 'crash_year', 'crash_month' and 'crash_period' were sorted in ascending order; finally, a csv file containing the final results was selected as the destination of the flow.

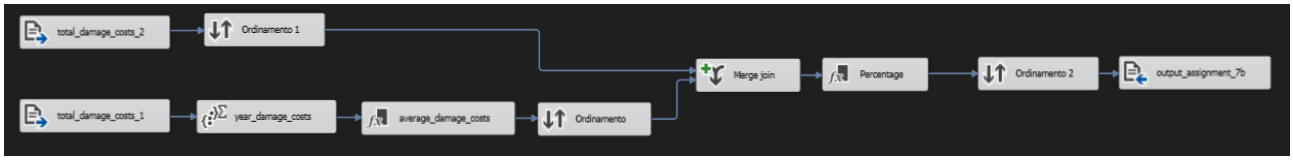

Figure 7. Second data flow

1.9. Assignment 8b

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

To answer this question, the data flow in Figure 8 was implemented: the origin of the flow is the table 'damage', of which the columns 'damage_cost', 'crash_id' and 'vehicle_id' were selected; through a lookup, a join was performed with the table 'vehicle', in order to retrieve the column 'vehicle_type'; then, through a second look-up, a join was performed with the table 'crash', in order to retrieve the column 'weather_condition'; the sum was then calculated on the column 'damage_cost' by grouping according to the columns 'vehicle_type' and 'weather_condition'; finally, a csv file containing the results of the query was selected as the destination of the flow.

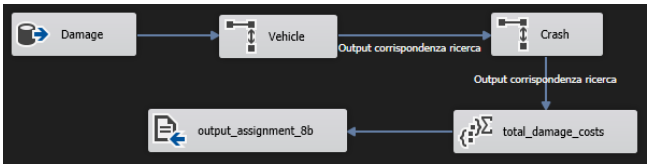

Figure 8. Data flow 8b

1.10. Assignment 9b

> "For every police beat, compute the total number of vehicles involved in an incident occurring in summer or spring, and the total number of vehicles involved in an incident occurring in winter or autumn."

This query was defined taking into account the previous queries and the data understanding phase. In fact, it was decided to use the police beat as it is part of the table 'location' which was not used in the previous queries. Similarly, it was decided to use the number of vehicles, i.e. the measure 'num_units', because only the measure 'damage_cost' was used in the previous queries. Furthermore, it was chosen to use the attribute 'crash_season' created in assignment 2 after realising in the data understanding phase that incidents can be grouped into seasons.

To answer this query, the data flow in Figure 9 was implemented: the origin of the flow is the table 'damage', of which the columns 'num_units', 'date_id' and 'location_id' were selected; through a lookup, a join was performed with the table 'location', in order to retrieve the column 'beat_of_occurrence'; then, through a second lookup, a join was performed with the table 'date', in order to retrieve the column 'crash_season'; then a conditional split was performed, which, based on a condition on the column 'crash_season', divided the data into two branches ('Summer_Spring' and 'Winter_Autumn'); in each branch, the sum on the column 'num_units' was calculated by grouping according to the column 'beat_of_occurrence'; furthermore, in each branch, a derived column called 'crash_seasons' was added whose values are 'SUMMER_SPRING' or 'WINTER_AUTUMN' depending on the branch; after merging the two branches, a sorting was performed on the columns 'beat_of_occurrence' and 'crash_seasons' in ascending order; finally, a csv file containing the results of the query was selected as the destination of the flow.

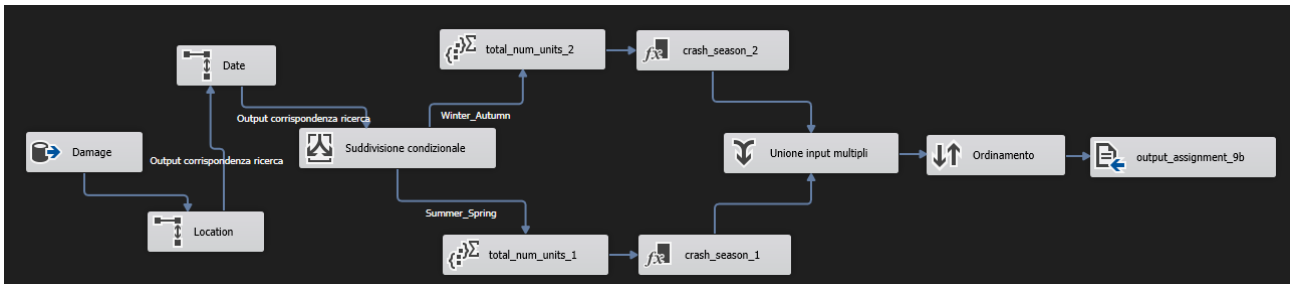

Figure 9. Data flow 9b

2. Part 2

2.1. Assignment 1

To build the datacube, first a new data source was created, i.e. a connection to the data warehouse designed and populated in part 1. After also creating a data source view, the dimensions of the cube were created, which correspond to the dimensional tables of the data warehouse. In fact, each cube dimension was created from a dimensional table and contains the same attributes as that table. The exception is the Date dimension, in which the new attribute 'crash_month_string' was created, which is in a functional dependency with the original attribute 'crash_month'. The following hierarchies have been defined in the dimensions:

- Crash: prim_contributory_cause → sec_contributory_cause
- Date: crash_year → crash_month_string → crash_day → crash_period → crash_time

- Location: beat_of_occurrence → location_point
- Injury: injuries_total → most_severe_injury
- Person: state → city
- Vehicle: make → model

After creating the dimensions, the cube was created by including the measures from the fact table of the data warehouse, i.e. 'damage_cost' and 'num_units'. In fact, 'damage_cost' is needed to answer the queries of assignments 2, 4 and 6, while 'num_units' is needed to answer the query of assignment 7. In addition to these two measures, the measure 'num_crashes' was created to answer the first query of assignment 8. On the measures 'damage_cost' and 'num_units', sum was specified as the aggregation function, while on the measure 'num_crashes', a distinct count on the 'crash_id' column of the fact table was specified as the aggregation function. Once the cube with the measures was created, it was deployed in order to make it visible on the server.

2.2. Assignment 2

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

The presented MDX [4] code is designed to perform a detailed damage cost analysis by exploiting the multidimensional data structure called 'Damage Cube'.

Initially, a member called [Measures].[Grand Total] was defined, representing the grand total of damage costs aggregated at two hierarchical levels above the current member of the [Date].[Hierarchy] hierarchy. The [Measures].[Damage Cost] measure, on the other hand, provides the value of the damage cost associated with each member.

In the SELECT clause, two measures were selected for the columns: [Measures].[Damage Cost] and [Measures].[Grand Total]. For the rows, a cross-join operation was applied on the following sets: [Date].[Hierarchy].[Crash Year].MEMBERS: the set of years in which the incidents occurred; [Date].[Crash Month String].[Crash Month String].MEMBERS: the set of months, represented as strings; [Location].[Hierarchy].[Location Point].MEMBERS: the set of previously calculated geohashes, representing geographical points.

The NONEMPTY operation ensures that empty values are excluded for the measurement [Measures].[Damage Cost]. This approach reduces noise in the data, improving the readability and quality of the results produced.

2.3. Assignment 4

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

Our MDX query focuses on analysing annual percentage changes in damage costs in order to identify areas where significant changes have occurred.

Initially, a member called [Measures].[Year Per Damage Cost] is defined, which represents the damage cost for the current year, using the current member of the time hierarchy [Date].[Hierarchy]. This member provides the damage costs for the selected year, serving as a basis of comparison for the following year. Another member is then defined, called [Measures].[Previous Year Per Damage Cost], which calculates the damage cost of the previous year. Using the function .LAG(1), the member moves the calculation to the previous member in the time hierarchy, thus obtaining the value for the previous year.

Next, the member [Measures].[Damage Cost Change] is defined, representing the percentage change in damage cost compared to the previous year. If no data exists for the previous year, as in the case of empty values, the change is set to zero. Otherwise, the percentage change is calculated as the difference between the current and previous year's cost, divided by the previous year's cost, and the result is multiplied by 100 to obtain a percentage. The result is then formatted as a percentage using the FORMAT_STRING attribute.

[4] MultiDimensional Expressions

In the body of the query, the measure [Measures].[Damage Cost Change], which represents the percentage change in damage costs, is selected as the main column. For the rows, members of the [Location].[Hierarchy].[Location Point] (geographical points) and [Date].[Hierarchy].[Crash Year] (years) hierarchies are selected. The NONEMPTY function is used to include only members that contain non-empty values for the measure [Measures].[Damage Cost Change]. In this way, combinations of data with no significant change are eliminated, improving the readability of the results.

Finally, the FILTER function is used to select only combinations in which the percentage change in damage costs is non-zero. This filter allows one to focus on cases where there are significant changes in costs, while excluding combinations of data that show no significant change.

2.4. Assignment 6

"For each vehicle type and each year, show the information and the (total) damage costs of the person with the highest reported damage."

The main objective of this query is to select the damage cost in relation to vehicle type and accident year, focusing on the combination with the highest damage value for each pair of dimensions.

Initially, the measure [Measures].[Damage Cost] is selected as the analysis measure, and is arranged in the query columns. This value represents the damage cost associated with each combination of dimensions that will be explored in the query.

For the rows, a combination of the dimensions [Vehicle].[Vehicle Type] and [Date].[Hierarchy].[Crash Year] is used. The GENERATE function creates a series of member combinations, where each combination represents a particular vehicle type and crash year. Then, for each combination, the most relevant member is selected, sorted by damage cost, using the TOPCOUNT function. This yields the combination with the highest damage cost from a number of additional dimensions: city, state, gender and age group of the person involved in the accident.

The NONEMPTY operation is applied to ensure that only member combinations with a non-zero value for the measure [Measures].[Damage Cost] are included. This step is essential to eliminate combinations that have no data, improving the quality of results and reducing noise in the data.

Finally, the query analyses data from the 'Damage Cube', which contains all the information needed to calculate and return the required values.

2.5. Assignment 7

"Extend the Assignment 9b solved in the first part of the project by calculating for every police beat the seasonality crash index, i.e. the ratio between the total number of vehicles involved in an incident occurring in summer or spring, and the total number of vehicles involved in an incident occurring in winter or autumn."

This query was proposed to show a fact that was discovered by solving assignment 9b in part 1. In particular, it was discovered that, in most police beats, the total number of vehicles involved in incidents occurring in summer or spring is lower than the total number of vehicles involved in incidents occurring in winter or autumn. Thus, it was decided to investigate this fact by calculating a seasonality index for each police beat, i.e., a ratio between the two numbers.

To begin with, two members are defined: [Measures].[Summer And Spring Number] and [Measures].[Winter And Autumn Number]. The first member calculates the total number of units (vehicles) involved in accidents that occurred during the summer and spring seasons. This is achieved by summing the values for the [Measures].[Num Units] across the seasons labeled as SUMMER and SPRING in the [Date].[Crash Season] dimension. Similarly, the second member, [Measures].[Winter And Autumn Number], calculates the total number of units involved in accidents during the winter and autumn seasons, summing the values for the WINTER and AUTUMN seasons.

Next, the seasonality index is calculated using the member [Measures].[Seasonality Crash Index]. This index is derived by dividing the total number of units involved in summer and spring accidents by the total number of units involved in winter and autumn accidents. The formula for this index is [Measures].[Summer And Spring Number] / [Measures].[Winter And Autumn Number]. An index value greater than 1 would indicate a higher incidence of accidents during the warmer seasons, while an index less than 1 suggests that the colder seasons (winter and autumn) are associated with more accidents.

In the SELECT clause, the [Measures].[Seasonality Crash Index] is selected for the columns, allowing us to analyze the seasonality index for each police beat. For the rows, the query uses [Location].[Gerarchia].[Beat Of Occurrence], which

represents the different geographical beats where incidents occur. This enables the investigation of seasonal patterns across various locations.

In summary, this MDX query investigates the seasonality of traffic accidents by calculating an index that compares the frequency of accidents in warm and cold seasons. The results suggest that in most police beats, accidents in colder seasons tend to involve more vehicles, as reflected by an index that is consistently less than 1. This analysis can be valuable for understanding seasonal trends and informing traffic management strategies.

2.6. 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."

The MDX code presented is designed to analyse the causes of accidents and their impact, in particular the cost of damage.

Initially, the weights for primary and secondary causes are calculated. The weight for primary causes is determined by the maximum number of accidents associated with each primary cause, multiplied by a factor of 2, thereby emphasising the causes with the greatest impact. Similarly, the weight for secondary causes is calculated as the maximum number of accidents associated with each secondary cause, but without applying any multiplier. These weights are then used to compare the relative importance of primary causes versus secondary causes.

Next, the most frequent type of cause (primary or secondary) is identified by comparing the calculated weights. The cause with the highest weight is defined as the predominant cause, whether primary or secondary. The name of the most frequent primary and secondary cause is then identified, using the TOPCOUNT function to select the cause with the highest number of incidents.

In addition, the code calculates the total damage cost associated with the most frequent cause, both primary and secondary. Using a filtering operation, the damage cost is summed for all occurrences of the most frequent cause, allowing the economic impact associated with each cause to be understood.

Finally, the code extends the calculation of the weights to a higher level of the time hierarchy by aggregating the data at year level. In this way, it is possible to compare primary and secondary causes not only at the accident level, but also in an aggregated view at the temporal level, providing an overall perspective on the factors influencing the number of accidents and their associated costs.

In the SELECT clause, several calculated measures are selected, including the weight of primary and secondary causes, the type of most frequent cause, the name of the most frequent cause and the damage associated with these causes. The rows of the query are organised according to the years in which the accidents occurred, making it possible to observe the development of the causes and associated costs over time.

2.7. Assignment 9

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

Our first dashboard focuses on analysing the distribution of total costs per vehicle, examining correlations with geographical distributions in our data cube.

A pie chart gives a clear view of the types of injuries resulting from accidents. The most represented category is 'NO INDICATION OF INJURY', which covers 59.19% of the total distribution, followed by 'NON-INCAPACITATING INJURY' with 12.46%. The other categories, such as 'FATAL', have significantly lower percentages. This gives us an immediate picture of the general severity of accidents.

A horizontal bar graph illustrates the annual change in costs associated with accidents over the period 2014-2018. A significant peak is observed in 2018, with costs reaching approximately USD 637.16 million, showing a significant increase over the years analysed.

On the right, an interactive map represents the spatial distribution of accidents, categorised by vehicle type (e.g. buses, mopeds). The size of the bubbles is proportional to the damage costs, while the areas of highest accident density coincide with strategic areas, probably characterised by critical infrastructure or high traffic.

The total costs associated with accidents exceed USD 1.4 billion, with 1,169,014 units involved. A grouped bar graph shows how these costs are distributed over the different seasons, showing significant peaks in summer and autumn.

Figure 10. Geographical Dashboard

2.8. 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 second dashboard focuses on road analysis, examining road conditions, seasons and weather.

A bubble graph shows the number of units involved in relation to weather conditions. Most accidents occur under clear sky conditions, but events such as rain and snow, although less frequent, account for a significant percentage, signalling a greater risk under adverse weather conditions.

A radar diagram shows the monthly distribution of accidents, with noticeable peaks in the summer months, particularly July and August. This could be attributable to increased road traffic and outdoor activities during this season.

A comparative bar chart shows that most accidents occur on roads with no apparent defects or unknown conditions. However, specific defects such as potholes or worn surfaces are associated with a considerable number of events, highlighting the importance of road maintenance for accident prevention.

Finally, a heat map visually represents the roads with the highest concentration of accidents, identifying critical areas requiring priority infrastructure interventions.

Figure 11. Streets Dashboard

2.9. 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 third dashboard focuses on people involved in incidents, analysing demographic details and the severity of the events.

A pie chart shows that almost 50% of the reported injuries are classified as ’NON-DISABLING’, followed by ’NO INDICATION OF INJURY’, which accounts for 39.64%. A very small percentage of events, 0.1%, is associated with the category ’FATAL’, underlining the seriousness of such incidents.

A line graph shows that men are more frequently involved in accidents than women. The number of cases recorded for gender ’X’ or undeclared is very small, probably due to gaps or limitations in the data collected.

A combined bar and line graph illustrates the development of costs and the number of accidents on different days of the week. Saturday emerges as the day with the highest costs, probably due to the higher traffic volume typical of weekends.

Finally, an interactive map represents the spatial distribution of incidents, categorised according to the variable ’DAMAGE_CATEGORY’. The areas with the highest costs are clearly visible, facilitating the identification of critical areas for targeted interventions.

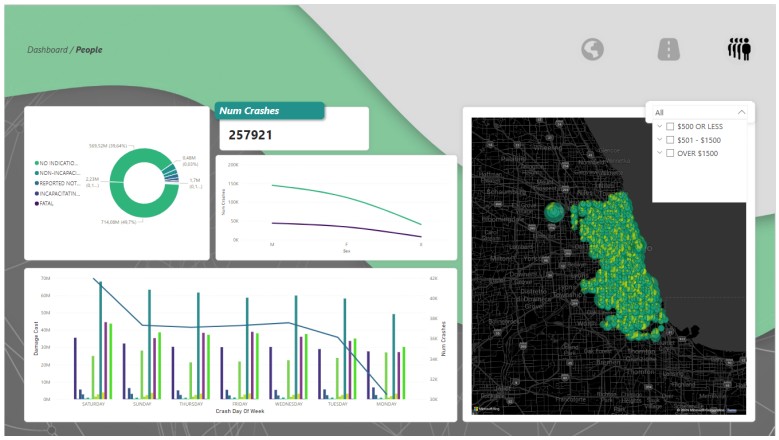

Figure 12. People Dashboard