Laboratory of Data Science - Group 26

Pietro Argento, Emad Chelhi

05/12/2024

Intro

In the following report, we are going to present the steps to design and implement the decision support system for an insurance company.

1 Data Understanding

Dataset description As suggested, we started understanding and exploring the data at hand. The crash data show information on each traffic crash on city streets within the City of Chicago limits and under the jurisdiction of the Chicago Police Department (CPD). Along with crashes, there are data about people involved in a crash and if any injuries were sustained. Each record corresponds to an occupant in a vehicle listed in the Crash dataset. Some people involved in a crash may not have been an occupant in a motor vehicle, but may have been a pedestrian, bicyclist, or using another non-motor vehicle mode of transportation.

The last set of information regards vehicles involved in a traffic crash. This includes motor vehicles and non-motor vehicle modes of transportation, such as bicycles and pedestrians. Each mode of transportation involved in a crash is a "unit" and gets one entry here. This type of identification of "units" is needed to determine how each movement affected the crash. Many of the fields are coded to denote the type and location of damage on the vehicle. Since this last dataset is a combination of vehicles, pedestrians, and pedal cyclists, not all columns are applicable to each record, and many "null" values are present.

Approaches We considered different approaches to deal with the many missing values present in the 3 datasets. These include: leaving the attribute "null" as a blank space (given how we read the files), recovering them from other columns and external sources, or adding some meaningful labels like "unknown" or "n/a". We addressed categorical columns with the latter where the original data types match the label (e.g., "string"). We chose the right label based on the meaning of the columns.

While for the numerical ones, we left a blank space. This decision is motivated by the purpose of this project: to offer ad hoc analysis and custom reporting to analysts and decision-makers of an assurance company. Considering this, we avoided imputation strategies common in data mining/machine learning or deleting the affected rows. It is of interest to keep all information at our disposal and to avoid skewing it (substituting a "zero" to missing values in numerical features, e.g., metrics, would improperly skew all aggregated calculations, often subject of the business questions).

2 Data Cleaning

Having terminated this first exploratory data analysis, we started designing the "extract and transform" part of the ETL pipeline. We designed the code, free of pandas and pandas-like packages, focusing on readability, maintainability, and reusability. It is composed of small functions (easier to understand and to test) with meaningful names, avoiding duplication of code (ensuring the "DRY principle": don't repeat yourself).

We decided to approach the **data cleaning be-**

fore merging the three different tables, step that we delayed as it would create a much bigger (and memory intensive) table. We are aware that, given our goals, the best would be to recover all the missing values, but this was possible only for some columns.

Missing Data Given the assignments on beats, we decided to concentrate our effort on retrieving the **missing beats**, which was possible for all the rows. The process, however, took differente steps. First, we obtained the shapely files of the beats from the Chicago Data Portal, basically the polygons with the borders. Second, we got the missing coordinates querying Nominatim through geopy, using the address available, ensuring that the coordinates were within the bounding box of Chicago. Third, we performed a spatial join between the polygons of the beats and the coordinates with no beat. This steps allowed to fill the missing beats.

Another problematic column was ***damage_amount***, since there were many missing values and it was supposed to be the main measure of the fact table. Since we observed that all the missing amounts corresponded with the category *$500 or less*, we decided to fill them with zero.

Regarding the vehicle table, we encountered value in ***vehicle_year*** that were implausible because after 2019. We tried to retrieve a plausible year, but it was not always deterministic, so we decided to replace all the implausible values with null.

Additional Data During the cleaning of the crash table, we added two columns in the crash table. The first one is the *is_holiday* column, that, using the *holidays* python package, included a boolean variable depending on the day of the year. The second column is *avg_beat_crimes*, which is obtained from data available on the City of Chicago Data Portal. We decide to integrate the information calculating the average number of crimes in each beat in the same time interval of the crash data, then we performed an inner join on the beats.

3 DW Schema

DW Design The next task was to design the data warehouse schema. After gathering business requirements and looking at the business questions we were asked to answer, we agreed on the "damage to user" the granularity of the fact. As measure, we ended up choosing only the column people["DAMAGE"], which represents the damaga amount, because it was the only field whose semantics was closely related to the fact table. This additive metric had some missing values.

The design of the data mart continued with the dimensions and related attributes.

DW Alternative We also considered the use of a snowflake schema, a normalized variation of the star schema. Logically, on the one side, we could agree that some dimensions, like "PLACE" and "DATE", closely relate to the "CRASH" dimension and it would make sense to combine them in hierarchical levels, potentially normalizing the whole schema by further dividing them. On the other hand, searching for damages filtered by specific time periods or geographic locations is common, and a star schema ensures that we can quickly aggregate data without the additional complexity of joins. So, the risk of degrading performance in simple queries, having many smaller tables, was more important for us than the advantages of a normalized schema.

Tables Creation In order to facilitate changes in the structure of the DW, we decided to create the tables on the Sql Server in Python. A global dictionary contained all the dimensions, attributes and types. It was used to create the queries that were then sent to the server with pyodbc.

4 Data Preparation

The goal of data preparation was to obtain the different dimensions tables and the fact table, ready to be uploaded.

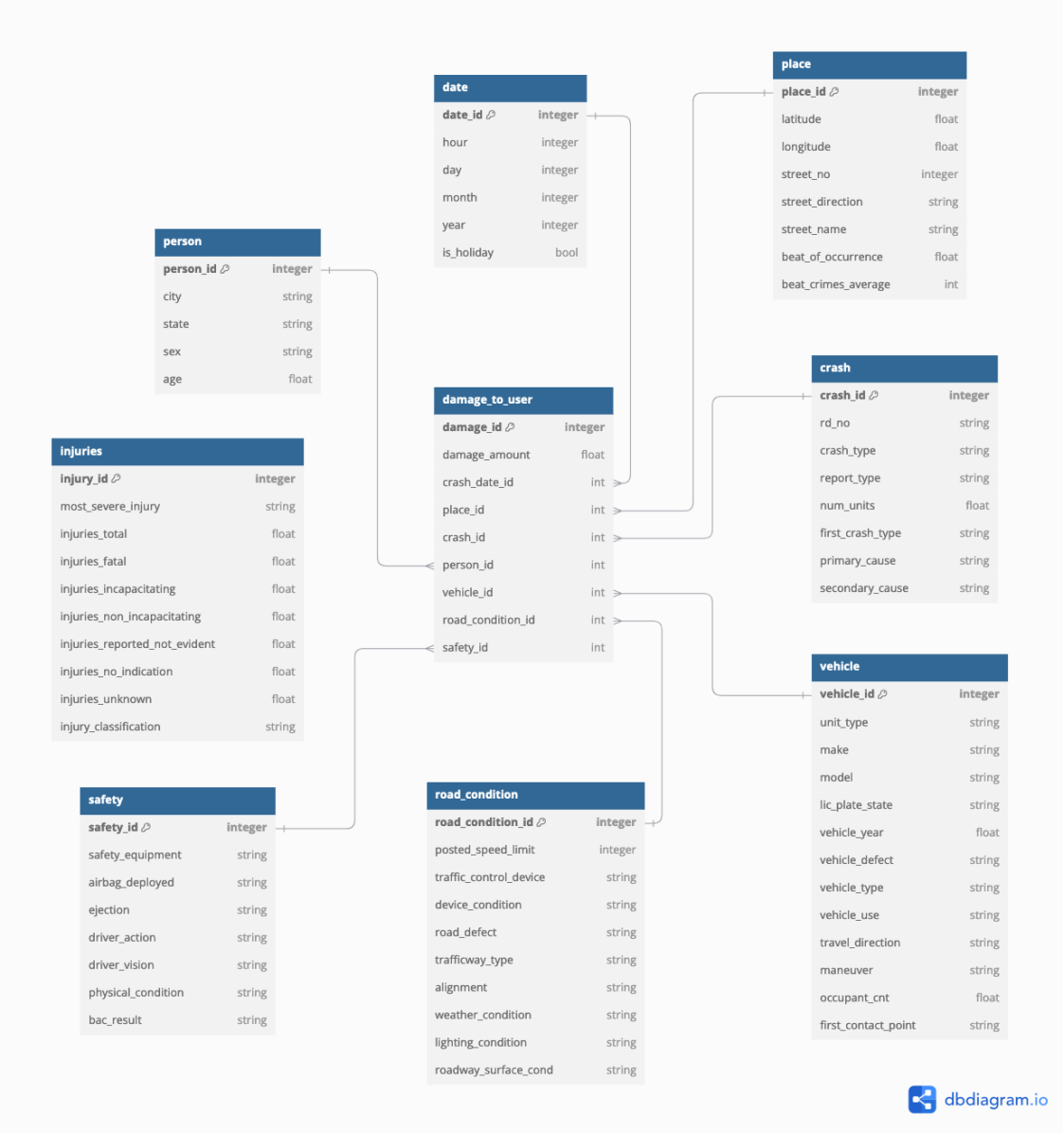


Figure 1: Data Warehouse Schema

Join all tables The first step was to finally merge the different tables, now cleaned. This step was crucial because, distributing the single columns in the various dimensions, we could assign the correct id in the fact table to link the fact with the correct rows of the dimensions. There were two steps to obtain the merged table, the first was a left join of people (the one with the chosen granularity) with crashes on `RD_NO`, the second was a left join also with vehicles on `VEHICLE_IS`. The keys have been chosen during the data understanding phase and reading the documentation of the data available online.

Split the data As mentioned, a fundamental part for creating the dimension tables was a dictionary mapping all the rows to fill the columns of the fact table with the correct ids. Additionally, we decided to fill the incremental id of the fact table with a simple line in python, and not server-side, since this is supposed the first upload of data from a single computer. However, it is important to notice that for further uploads, especially from different clients, it is important to update the surrogate key server-side.

Deal with data types An important function that is worth mentioning, is the one that addressed the problem of data types. Since Python, Sql Server and SSIS work with different data types, we made a great effort in finding the correct types that worked with all the tools without losing information.

5 Data uploading with python

Upload of batches Exploiting the global dictionary mentioned before, we created a single function that was able to populate all the dimensions tables plus one for the fact table. We found that the fastest way to upload the data was using a recent update of the pyodbc function to execute the queries and the use of batches. In particular, the interesting snipped is the following: `cursor.fast_executemany = True; cursor.executemany(insert_statement, batch); cursor.connection.commit()`

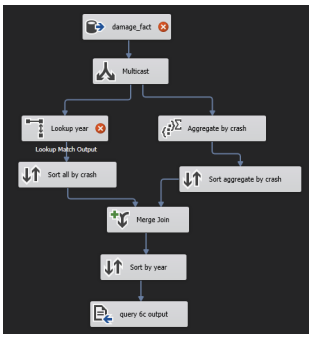


Figure 2: SSIS 6C

6 Data uploading with SSIS

Populate the tables After creating the tables with Python as done previously, we created a single control flow with one data flow for each table. We tried to further automate this process, but each table required special attention to handling columns and data types. We manually check every columns and every type, heuristically choosing a more efficient type in terms of space.

Sample the data To approach the 10% sampling of the data, as required, we decided to start from the fact table. As it is clear, sampling each table would have been wrong. So, we sampled the fact table and then, in the data flow, obtained only the rows with the ids that were present in the fact table.

7 Queries with SISS

Requirements gathering The queries assigned were a constant reference in the design of the DW. To be sure that the proposed design was suitable to answer the queries, we performed all the steps suggested by the theory, from requirement gathering to the identification of the dimensional attribute hierarchies.

Results of the queries The results of the queries can be found in the folder "ssis". They are saved as csv files.

Figure 3: SSIS 9C

As interesting query (Figure 3), we proposed the following: `"A beat is classified as criminal if the number of crimes within that beat exceeds the average number of crashes across all other beats by more than 10%. List all the beats that meet this criterion and, for these beats, show the primary contributory cause to the crash, ordered by the total crash damage costs in percentage, considering only the crashes happened during the holidays."`

8 Multidimensional Data Analysis

8.1 OLAP Cube

The second part of the project involved the creation of a multidimensional cube to answer additional business questions using the MDX language and Power BI.

Steps Followed

The following steps were taken to create and deploy the OLAP cube:

1. Creation of an SQL Server Analysis Services project;
2. Connection to our data warehouse (`Group_26_cube`);
3. Definition of a source view to work independently of the underlying data sources (e.g., renaming columns without affecting the original data warehouse);
4. Definition of the cube, measures, dimensions, attributes, and hierarchies;
5. Deployment of the cube.

Dimensions and Attributes

All dimensions and attributes were retained for completeness. In the *Time* dimension, a calculated field, `Quarter`, was added using the following expression: `"add expression"`.

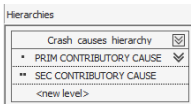
We curated in first place the data source so any change of datatype was necessary. No issues with ordering have been encountered as well.

Hierarchies

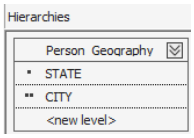
In addition to flat hierarchies, the following multidimensional hierarchies were created:

- **Time_hierarchy**

- **Crash_causes_hierarchy**



- **Person_geography**



- **Crash_geography**

These have been created with a view to potential future utility. But not all of them have been used.

Measures

Initially, only the `Damage_amount` measure was included, with plans to add additional measures as needed. However, subsequent tasks did not require any new ones.

Final Cube

The final deployed cube is composed by:

- **2 measures:** `Damage_amount` and `Damage_fact_count`;
- **8 dimensions:** All attributes present in the data warehouse, along with the multidimensional hierarchies listed above.

8.2 MDX queries

Query of Assignment 2

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

```
select
NONEMPTY({[Date].[MONTH].[MONTH], [Date].[MONTH].[All]})
on columns,
NONEMPTY([Place].[BEAT OF OCCURRENCE].[BEAT OF OCCURRENCE])
on rows
from [Damages]
where [Measures].[DAMAGE AMOUNT]
```

Figure 4: Query assignment 2

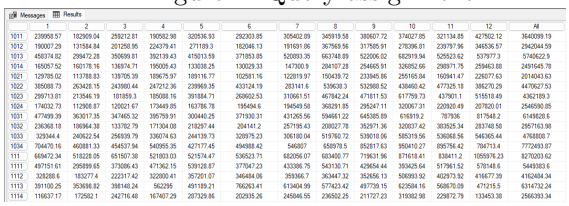


Figure 5: Result assignment 2

As it can be seen from the report of the query, for each beat, listed on the rows, it is shown the damage cost for each month and for all months summed up.

Query of Assignment 4

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

The damage cost variation has been computed as:

$$\frac{Damagecost(currentyear) - Damagecost(previousyear)}{Damagecost(previousyear)}.$$

The amount of the previous year is obtained through the MDX function `ParallelPeriod`, that returns a member from a previous (or next) period with the same position in the level as the specified member.

For each beat, on the rows, it is shown the increase or decrease of the damage cost with respect to the previous year. Year "2014" does not present any value because we don't have any data about 2013. While year "2015" present only few values (not shown in the photo) because of the limited statistics about 2014 present in our data warehouse.

```
with
    member prev as
     (PARALLELPERIOD([Date].[YEAR].[YEAR], 1,[Date].[YEAR].currentmember), [Measures].[DAMAGE AMO
    member perc as
        IIF(
            NOT ISEMPTY(prev),
            ([Measures].[DAMAGE AMOUNT] - prev) / prev,
            NULL
            ),
    format_string ="percent"
select
[Date].[YEAR].[YEAR] on columns,
[Place].[BEAT OF OCCURRENCE].[BEAT OF OCCURRENCE] on rows
from [Damages]
where perc
```

Figure 6: Query assignment 4

Messages | Results

	2014	2015	2016	2017	2018	2019
1011	(null)	(null)	550.16%	82.48%	46.58%	-99.88%
1012	(null)	(null)	975.67%	45.77%	77.85%	-99.46%
1013	(null)	(null)	516.56%	55.43%	74.36%	-99.14%
1014	(null)	(null)	192.39%	77.80%	67.20%	-95.53%
1021	(null)	(null)	393.26%	84.92%	30.04%	-97.84%
1022	(null)	(null)	716.16%	73.21%	74.20%	-97.76%
1023	(null)	(null)	375.85%	57.60%	69.48%	-98.91%
1024	(null)	(null)	547.50%	13.33%	140.76%	-97.62%
1031	(null)	(null)	219.14%	86.97%	42.24%	-97.17%
1032	(null)	(null)	260.07%	33.86%	76.47%	-97.84%
1033	(null)	(null)	661.03%	52.91%	88.11%	-98.72%
1034	(null)	(null)	327.08%	89.67%	64.74%	-97.92%
111	(null)	(null)	345.77%	69.80%	38.84%	-95.35%
1111	(null)	(null)	412.56%	184.09%	59.96%	-97.80%
1112	(null)	(null)	218.70%	289.89%	66.46%	-99.51%

Figure 7: Result assignment 4

```
WITH MEMBER [Measures].[Max Damage Person] AS
TOPCOUNT(NONEMPTY(
        ([Person].[PERSON ID].[PERSON ID].MEMBERS), [Measures].[DAMAGE AMOUNT]),
    1,
    [Measures].[DAMAGE AMOUNT]).Item(0).Member_Key

MEMBER [Measures].[Max Damage] AS
MAX(NONEMPTY(
            ([Person].[PERSON ID].[PERSON ID].MEMBERS),
            [Measures].[DAMAGE AMOUNT]),
    [Measures].[DAMAGE AMOUNT])

SELECT
    {[Measures].[Max Damage Person], [Measures].[Max Damage]} ON COLUMNS,
    ([Date].[YEAR].[YEAR].MEMBERS,
    [Vehicle].[VEHICLE TYPE].[VEHICLE TYPE].MEMBERS) ON ROWS
FROM Damages
```

Figure 8: Query assignment 6

Messages | Results

		Max Damage Person	Max Damage
2019	BUS OVER 15 PASS.	203	69159.95
2019	BUS UP TO 15 PASS.	88	7477.55
2019	FARM EQUIPMENT	(null)	(null)
2019	MOTOR DRIVEN CYCLE	602	2691.14
2019	MOTORCYCLE (OVER 150CC)	(null)	(null)
2019	OTHER	1	15719.02
2019	OTHER VEHICLE WITH TRAILER	1001	4420.87
2019	PASSENGER	7	533478.69
2019	PICKUP	7	36105.14
2019	SNOWMOBILE	(null)	(null)
2019	SPORT UTILITY VEHICLE (SUV)	7	143746.6
2019	TRACTOR W/ SEMI-TRAILER	7	26853.26
2019	TRACTOR W/O SEMI-TRAILER	387	5369.26

Figure 9: Result assignment 6

Query of 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.

To answer this business question two new calculated members have been defined:

- `Max damage Person` as the first element of the column `person id` sorted on `damage amount`. The function `TopCount` and the methods `.item()`, `.Member_Key` have been used for this purpose.
- `Max damage` as the total amount of damage the previous found person incurred on.

As the result shows, years and vehicle types are cross joined and, for each combination, the id of the person and the total damage amount are shown.

Query of Assignment 7

Rank the weather conditions by their contribution to total damage costs (as a percentage). Additionally display the number of crashes occurring during daytime (8 AM to 9 PM) and nighttime (9 PM to 8 AM) for each weather condition.

As the description of the query suggests, the aim here is to investigate the role of weather and of daytime/ nighttime in the number and entity of vehicle crashes.

The contribution of each weather condition to the total damage costs has been obtained dividing the damage amount for the current member (each specific weather condition) by the damage amount for its parent member (total damage costs). Daytimes and Nighttimes have been obtained through the function `Aggregate`, that aggregates values for specified

```
WITH
    MEMBER perc AS
        [Measures].[Damage Amount] /
        ([Road Condition].[WEATHER CONDITION].currentmember.parent, [Measures].[Damage Amount]),
        FORMAT_STRING = 'Percent'
    MEMBER Daytime AS
        AGGREGATE({[Date].[HOUR].[8] : [Date].[HOUR].[21]},  [Measures].[Damage Fact Count])
    MEMBER Nighttime AS
        AGGREGATE({[Date].[HOUR].[22] : [Date].[HOUR].[7]},  [Measures].[Damage Fact Count])

    SELECT
    {perc, Daytime, Nighttime} ON COLUMNS,
    order(NONEMPTY([Road Condition].[WEATHER CONDITION].[WEATHER CONDITION]),
        perc, bDESC) ON ROWS
FROM [Damages]
```

Figure 10: Query assignment 7

Messages | Results

	perc	Daytime	Nighttime
CLEAR	79.65%	102600	64027
RAIN	9.93%	15010	9824
UNKNOWN	3.32%	4740	3231
CLOUDY/OVERCAST	3.19%	4337	2408
SNOW	3.19%	5385	3344
OTHER	0.31%	562	389
FOG/SMOKE/HAZE	0.24%	539	418
SLEET/HAIL	0.15%	291	206
SEVERE CROSS WIND GATE	0.02%	26	12

Figure 11: Result assignment 7

members over numeric expression.

The report shows that crashes occurring on sunny days account for approximately 80% of the total damage costs. This suggests that sunny days may present specific conditions, such as higher traffic volumes or increased vehicle speeds, that contribute to the greater severity and frequency of crashes. Another interesting finding is that crashes during nighttime are approximately 30–40% lower than those during daytime. This difference may be attributed to the reduced traffic volume at night, although factors such as limited visibility and driver fatigue still keep these numbers quite high.

Query of Assignment 8.b

For each year, show the most risky crash type and its total damage costs. To measure how risky a crash type is, you should assign a weight to each type of injury you encounter in the data (for example, a fatal injury weighs 5 times an incapacitating one, which weighs twice a non-incapacitating injury).

```
WITH
    MEMBER [Measures].[Risk] AS
        50*SUM(
            [Injuries Dim].[INJURIES FATAL].members,
            VAL([Injuries Dim].[INJURIES FATAL].CURRENTMEMBER.NAME) * [Measures].[Damage Fact Count]
        ) +
         10* SUM(
            [Injuries Dim].[INJURIES INCAPACITATING].members,
            VAL([Injuries Dim].[INJURIES INCAPACITATING].CURRENTMEMBER.NAME) * [Measures].[Damage Fact Count]
        ) +
             5*  SUM(
            [Injuries Dim].[INJURIES NON INCAPACITATING].members,
            VAL([Injuries Dim].[INJURIES NON INCAPACITATING].CURRENTMEMBER.NAME) * [Measures].[Damage Fact Count]
        ) +
                   1*   SUM(
            [Injuries Dim].[INJURIES NO INDICATION].members,
            VAL([Injuries Dim].[INJURIES NO INDICATION].CURRENTMEMBER.NAME) * [Measures].[Damage Fact Count]
        ) +
               1*      SUM(
            [Injuries Dim].[INJURIES REPORTED NOT EVIDENT].members,
            VAL([Injuries Dim].[INJURIES REPORTED NOT EVIDENT].CURRENTMEMBER.NAME) * [Measures].[Damage Fact Count]
        )

    MEMBER [Measures].[Most Risky Crash Type] AS
        TOPCOUNT(
            [Crash].[FIRST CRASH TYPE].[FIRST CRASH TYPE],
            1,
            [Measures].[Risk]
        ).ITEM(0).NAME

    MEMBER [Measures].[Most Risky Crash Type Damage] AS
        (TOPCOUNT(
            [Crash].[FIRST CRASH TYPE].[FIRST CRASH TYPE],
            1,
            [Measures].[Risk]
        ).ITEM(0), [Measures].[DAMAGE AMOUNT])

SELECT
    {[Measures].[Most Risky Crash Type], [Measures].[Risk], [Measures].[Most Risky Crash Type Damage]} ON COLUMNS,
    [Date].[YEAR].[YEAR] ON ROWS
FROM [Damages]
```

Figure 12: Query assignment 8.b

Messages | Results

	Most Risky Crash Type	Risk	Most Risky Crash Type Damage
2014	ANGLE	25	10961.95
2015	REAR END	58822	16090775.8
2016	REAR END	293557	76016976.5899999
2017	REAR END	650026	136204283.72
2018	REAR END	994152	181114876.99
2019	REAR END	22060	3939890.15
Unknown	ANGLE	(null)	(null)

Figure 13: Result assignment 8.b

We answered this request using the function "Sum", useful to count and assign different weights, and "TopCount", to pick the most risky crash type.

9 Dashboards

The software used to create the following dashboards is Power BI.

9.1 Dashboard of assignment 9

Create a dashboard that shows the geographical distribution of the total damage costs for each vehicle category. To visualize the geographical distribution of total damage costs across vehicle categories, we

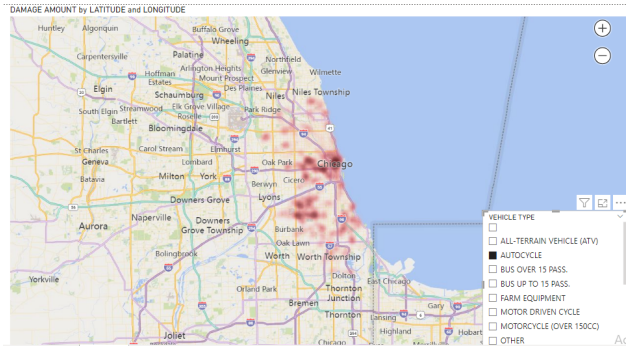


Figure 14: Report assign. 9, "autocycle"

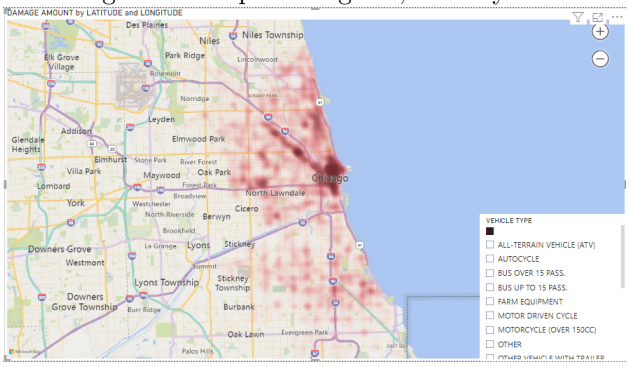


Figure 15: Report assign. 9, all vehicle types

thought of a heat map overlaying the map of Chicago. To enhance interactivity, we incorporated a slicer object, allowing users to select a specific vehicle type and explore areas with the highest verified damage amounts.

The 2 reports show this for all vehicle types (1st image) and for only autocycles.

9.2 Dashboard of assignment 10

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

Road defects play a significant role in causing traffic accidents. Potholes, road debris, and overall poor maintenance of roadways can lead to a temporary loss of vehicle control, increasing the risk of an accident.

With this report our aim is to show the geographical distribution, the number of crashes caused over the years and the damage costs per time, with hours as finest detail. We focused on the defect "debris on road" that, despite being frequent on the roads of Chicago, it's not cause of many crashes over the 5 years if we consider the total amount collected in the database. If we want to locate geographically the crash/es, caused by the analysed road defect, subject of a high damage amount , it is sufficient to click on the bar of interest, in the bar chart, and the location/ of the defect cause of the accidents will be shown in the map of Chicago.

With this report, we aim to present the geographical distribution, the number of crashes over the years, and the associated damage costs, shown with hourly detail. Our focus is on the road defect "debris on road," which, while frequently observed on Chicago's roads, has not been a significant cause of crashes over the five years covered in the database.

To identify the geographical location of crashes caused by this specific road defect, like those associated with high damage costs, users can simply click on the relevant bar in the bar chart. The corresponding crash locations will then be displayed on the map of Chicago.

9.3 Dashboard of assignment 11

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

With the reports presented above, we aimed to provide an overview of the causes and consequences of traffic accidents by leveraging the attributes: "physical conditions," "driver actions," and "injury classification." For each physical condition, it is possible to explore the most common driver actions, the gender distribution, and the classification of injuries.

Noteworthy examples include the conditions "emotional" and "impaired - alcohol." It is interesting to observe the contrasting gender distributions and how, under the influence of alcohol, the severity of injuries tends to increase.

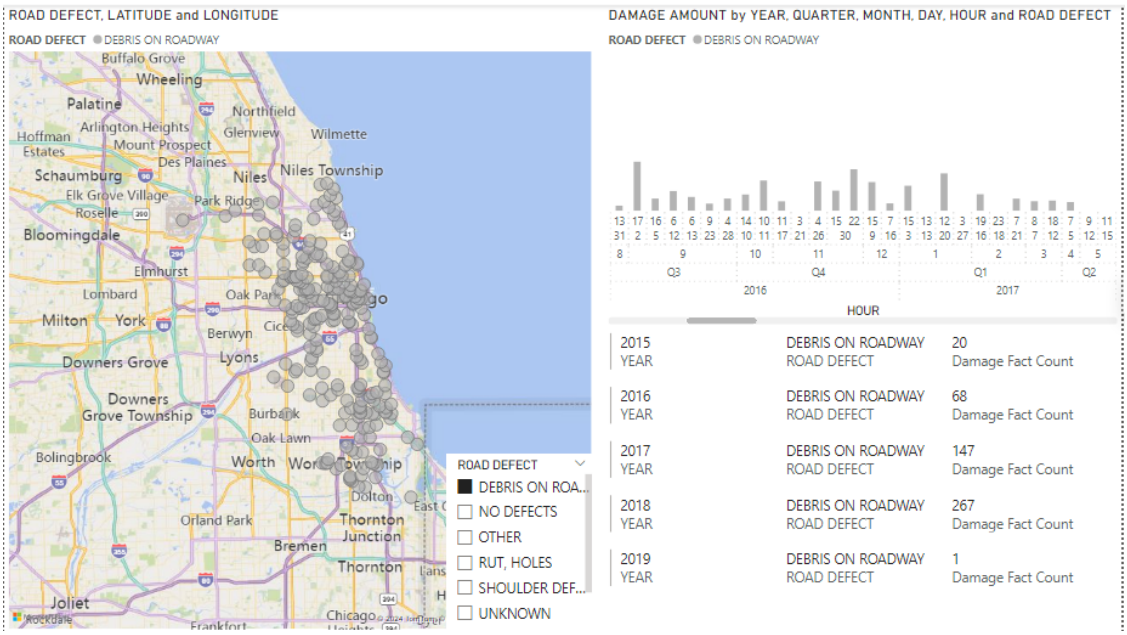


Figure 16: Report assign. 10

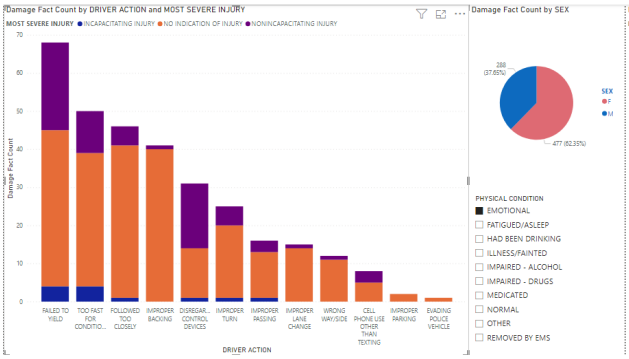


Figure 17: Report assign. 11.1

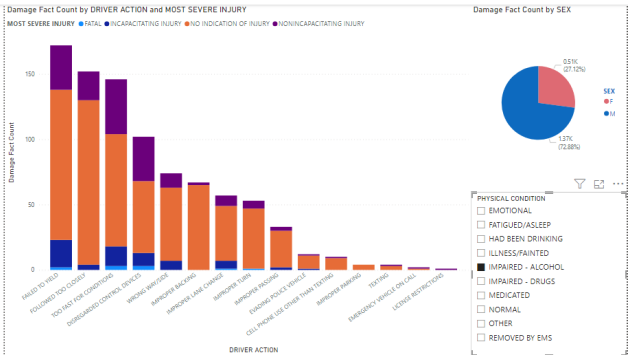


Figure 18: Report assign. 11.2