Report

IT-300
Business Intelligence and Database Management Systems

Business Intelligence Research
Analysis of Collisions in Yorkshire and Calderdale District

Authors:
Mohamed Yassine Mekni
Montaha Ghabri
Samer Laabidi
Zaineb Bouriga

Submitted to:
Dr.Manel Abdelkader
Dr.Ameni Azzouz

Outline

1 Introduction

This business intelligence initiative is dedicated to analyzing road safety dynamics in West Yorkshire. Our primary focus is to unravel the connections between various contributing factors and the observed trends in traffic incidents. By scrutinizing data spanning the pivotal years of 2019 and 2020, we aim to discern patterns and influences contributing to high-severity casualties, ultimately providing a comprehensive understanding of road safety dynamics

Our project aspires to equip Different Businesses from West Yorkshire with valuable Insights that they can utilize to better set their strategies and Enhance their overall performances.

2 Implementation

2.1 Data Gathering

- We extracted the Calderdale 2020 and Accidents 2019 datasets respectively from Kaggle, Calderdale Road Traffic Council.

2.2 Data Preparation

- For the data preparation, we used Talend to manipulate data and configure it for the data warehouse. At first, We established a Talend Job named "Calderdale2020 '' and seamlessly incorporated delimited files through metadata. Uploading the Accidents2019.CSV file involved a straightforward process without additional steps. However, for the CalderdaleCollisions2020.JSON file, we undertook a schema adjustment. This entailed inserting dataset components, including the names of each column, into an XML schema. This modification was crucial to enable Talend to comprehend the JSON data structure, facilitating accurate mapping to the corresponding dataset columns for subsequent processing.
- How we Enhanced Data Types for the Sake of Interpretability:

  The previous data type of the values in the date column was a serial number representation of a date in Microsoft Excel which we will convert into a String data type so we

can filter with tMap in Talend. These are the steps we followed:

1-integrate a tMap component onto the canvas to facilitate data transformation.

2-Establish a connection between the tFileInputDelimited component and the tMap component.

3-Define an output schema within the tMap component, designating the date column with a String data type.

4-Create a secondary connection within the tMap component to propagate the original row post-transformation.

5-Utilize the row.dateColumn variable in the tMap component to convert the serial number into a date. Example code snippet:

```
Date date = new Date((long)row.dateColumn * 24 * 60 * 60 * 1000);
output_row.dateColumn = TalendDate.formatDate("yyyy-MM-dd", date);
```

6-integrate a tFileOutputDelimited component onto the canvas for writing the output file.

7-Establish a connection between the tMap component and the tFileOutputDelimited component.

8-Upon execution, Talend reads the input file, transforms the serial number into a date using the specified logic, and writes the output file.

The column labeled 'Time (24hr)' underwent a conversion process in which its integer format was transformed into a date representation. Within the tMap's output schema, a new column was introduced specifically for storing these time values, with its data type set to "Date" to leverage Talend's internal handling of time representations. The conversion was carried out by creating a dedicated row within the tMap, where the conversion expression was articulated. The expression, "TalendDate.parse("HHmm", row1.Float(Time(24hr)/100)", involved dividing the integer value by 100 to isolate hours and minutes, and then formatting the result as a time string. To capture and store these converted time values, the tMap's output flow was connected to a target component, namely tLogRow.

//On the left is how the data looked before:

E	F
Accident Date	Time (24hr)
43469	701
43469	2308
43472	838
43473	705
43474	1515
43475	1424
43476	550
43476	700
43476	826

//On the right is how the data looked after:

E	F
Accident Date	Time (24hr)
2019-01-04	7:01
2019-01-04	23:08
2019-01-07	8:38
2019-01-08	7:05
2019-01-09	15:15
2019-01-10	14:24
2019-01-11	5:5
2019-01-11	7
2019-01-11	8:26

How we dealt with outliers:

We removed 4 outliers: Number of vehicles= 5/ (Two cases), Number of vehicles =6 (One case), and Number of vehicles=7 (One case). The following are the corresponding accident 'Reference Number' respectively: 6CJ0277, 6CN1273, 6CS0418, 6CS1526. By removing these outliers, a total of 15 columns were empty due to these outliers being the sole contributors to them. Thus we went from 45 columns to 30 columns and therefore we reduced the amount of missing data from 58.28% down to 33.63%.

tJavaRow with a Filtering Condition:

- We placed a tJavaRow component in the job.
- We connected its input flow to the data source.

```
// Total number of rows=272

    for (int i = 0; i < 272; i++)

    if (row1."Number of vehicles" == 5 || row1."Number of
    vehicles" == 6 || row1."Number of vehicles"== 7) {

      output_row.clear();
```

```
} else {

   // Pass the row through

} // Total number of rows=268
```

Such types of data might be confusing to use and they are a minority on the dataset. So they are considered outliers and have been removed to allow for simple and understandable data manipulation

* **How we dealt with duplicate values**

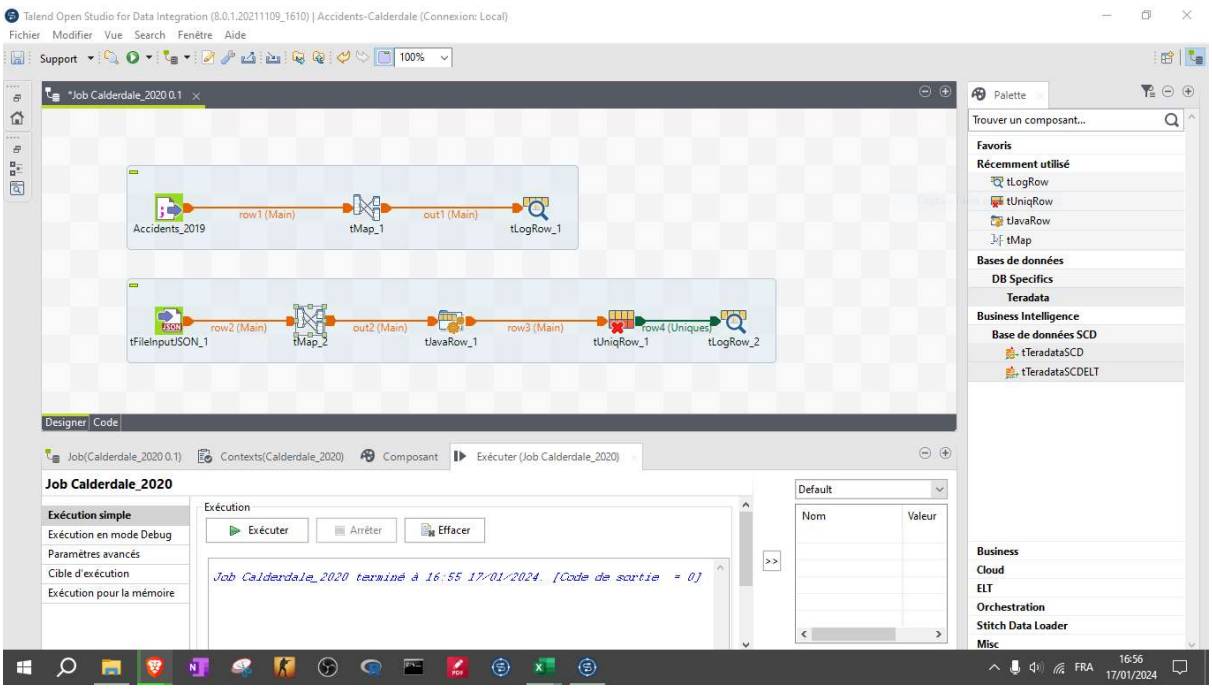
Following the tLogRow component, we incorporated a tUniqRow component into our workflow. This involved establishing a connection between the schema of the tFileInputDelimited component and the tUniqRow component. To identify and eliminate duplicates, we configured the tUniqRow component by selecting all columns as criteria for duplicate detection. Subsequently, we connected the tUniqRow component to a tFileOutputExcel component, configuring it to write to the output file. Upon executing the job, the resulting output file mirrored the data from the input file, but with the removal of any duplicate rows, amounting to a total of 1 row eliminated.

* **How we dealt with null values:**

We used an iterated function since it would not be efficient manually.We went from 58,82% of missing data to 33,63%.

```
// Iterate through all columns using the following Java code

for (int i = 0; i < input_row.length ; i++) {

    // For input_row.length, here it's 268 rows

   // Check for null values and replace with "NO DATA"
```

```
    if (input_row[i] == null) {

        output_row[i] = "NO DATA";

    } else {

        output_row[i] = input_row[i];

    }

}
```

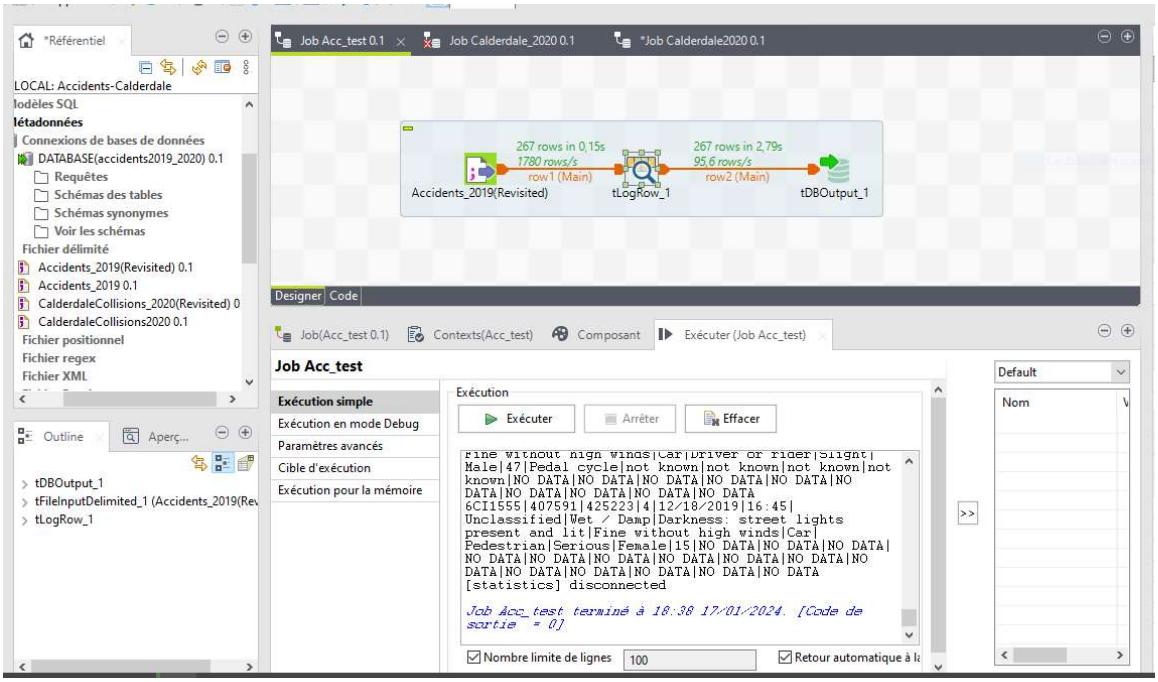


2.3 Data Storage

Using the job output where the delimited transformed file is "Accidents_2019", we proceeded in the following way:

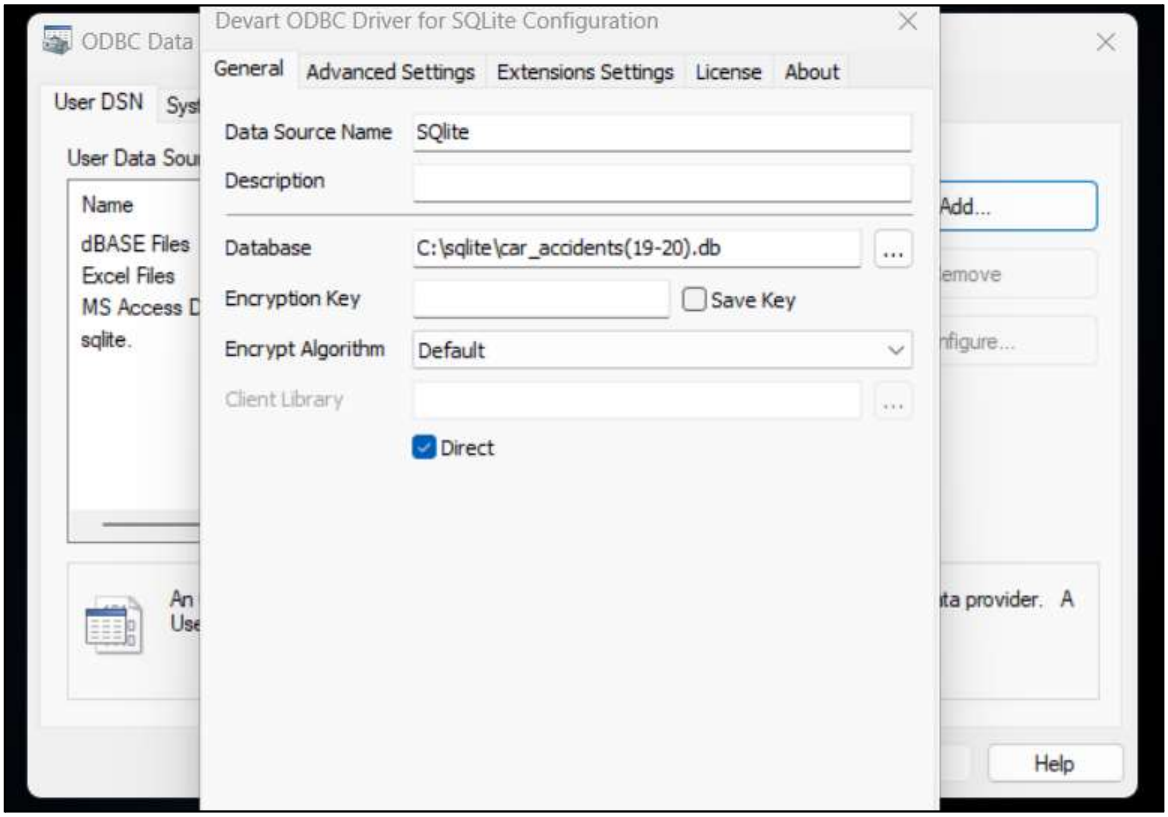
- In SQLite Studio, we created a new Database called "**DATABASE(accidents2019_2020)**", this step was purely to experiment with the SQL server's way of functioning.
- Using Talend, the uploaded version we had, did not allow an instant creation of the tables and insert of data at once, the data couldn't be inserted unless the tables already exist. Talend threw an error when we simply used "**tDBoutput**". Thus, we resorted into "**tSQLiteRow**" which created each table and its columns by synchronizing columns from input to output. To do so we linked "**tSQLiteRow**" to "**tLogRow**" and then we ran the job.
- We then linked "**tLogRow**" to "**tDBoutput**", and by giving the name of the created database and the corresponding table, the data was inserted in the database.

We repeated the last two steps for the "Calderdale Collisions 2020" delimited transformed output.

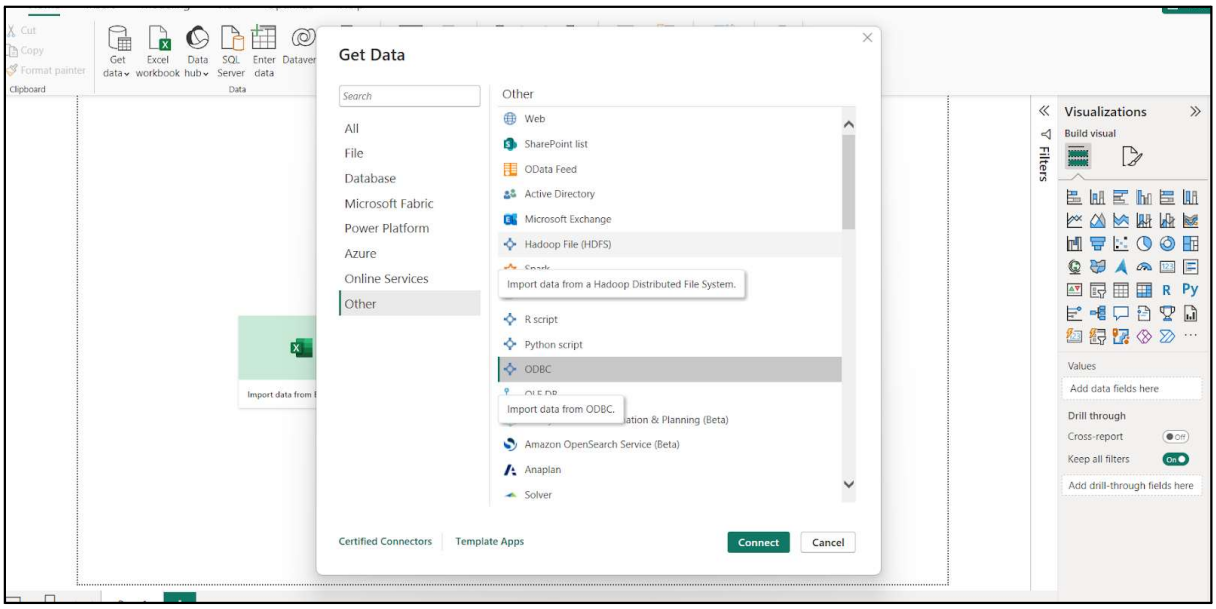


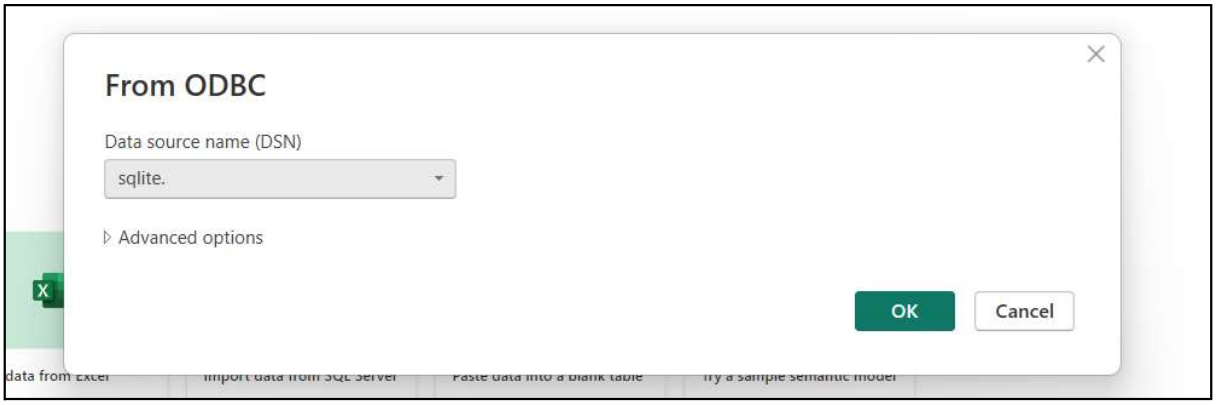
2.4 Modeling and DWH creation

- To model the data warehouse and create the essential fact and dimension tables within Power BI, we had to establish a connection between our SQlite database and Power BI through **ODBC**(Open Database Connectivity) by adding a data source (SQlite).



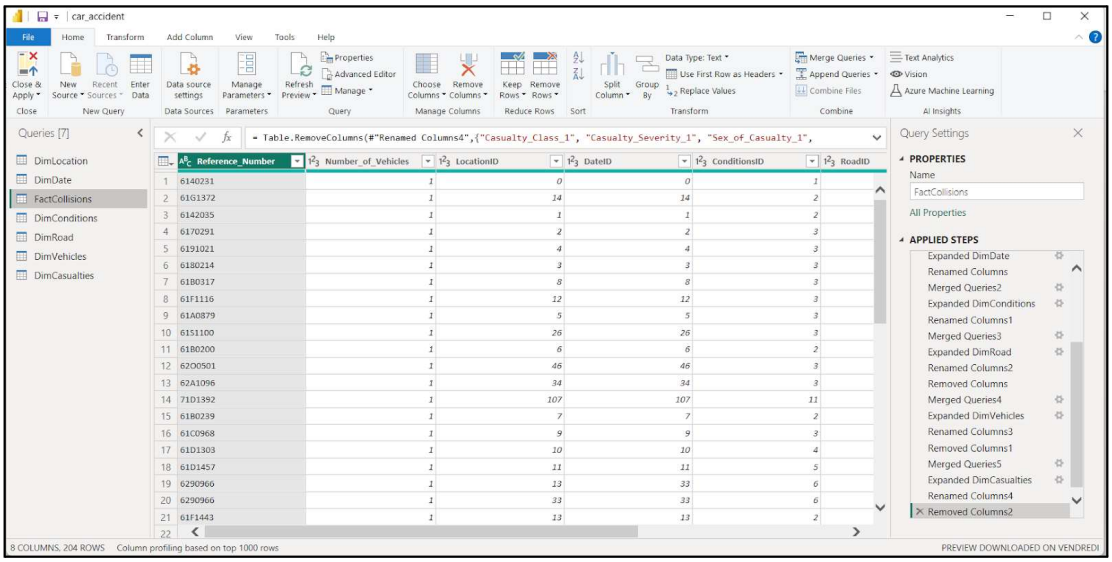
- **Data Loading:**





- **Data Transformation:**

  After loading the data in **Power BI**, we used **Power Query Editor** to perform the necessary data transformations and create the fact and dimension tables while establishing the connections between them to craft a Star Schema. The design and implementation of our data warehouse involves the utilization of ROLAP (Relational OLAP) principles.



  - **Fact table:**
    - Reference_number
    - number_of_vehicles

  - **Dimensions**
    - **DateDim** (DateID, Accident_Date, Time_24hr,MonthName,Year)

- **DimLocation** (LocationID, Grid_Ref_Easting, Grid_Ref_Northing)
- **DimVehicles** (VehicleID, Type_of_Vehicle_1, Type_of_Vehicle_2, Type_of_Vehicle_3, Type_of_Vehicle_4)
- **DimCasualties** (CasualtiesID, Age_of_Casualty_1,Age-Group, Casualty_Class, Casualty_Severity,Casualty_Sex)
- **DimRoad** (RoadID, Road_Surface, _st_Road_Class)
- **DimCondidtions** (ConditionsID, Lighting_Condidtions, Weather_Conditions)

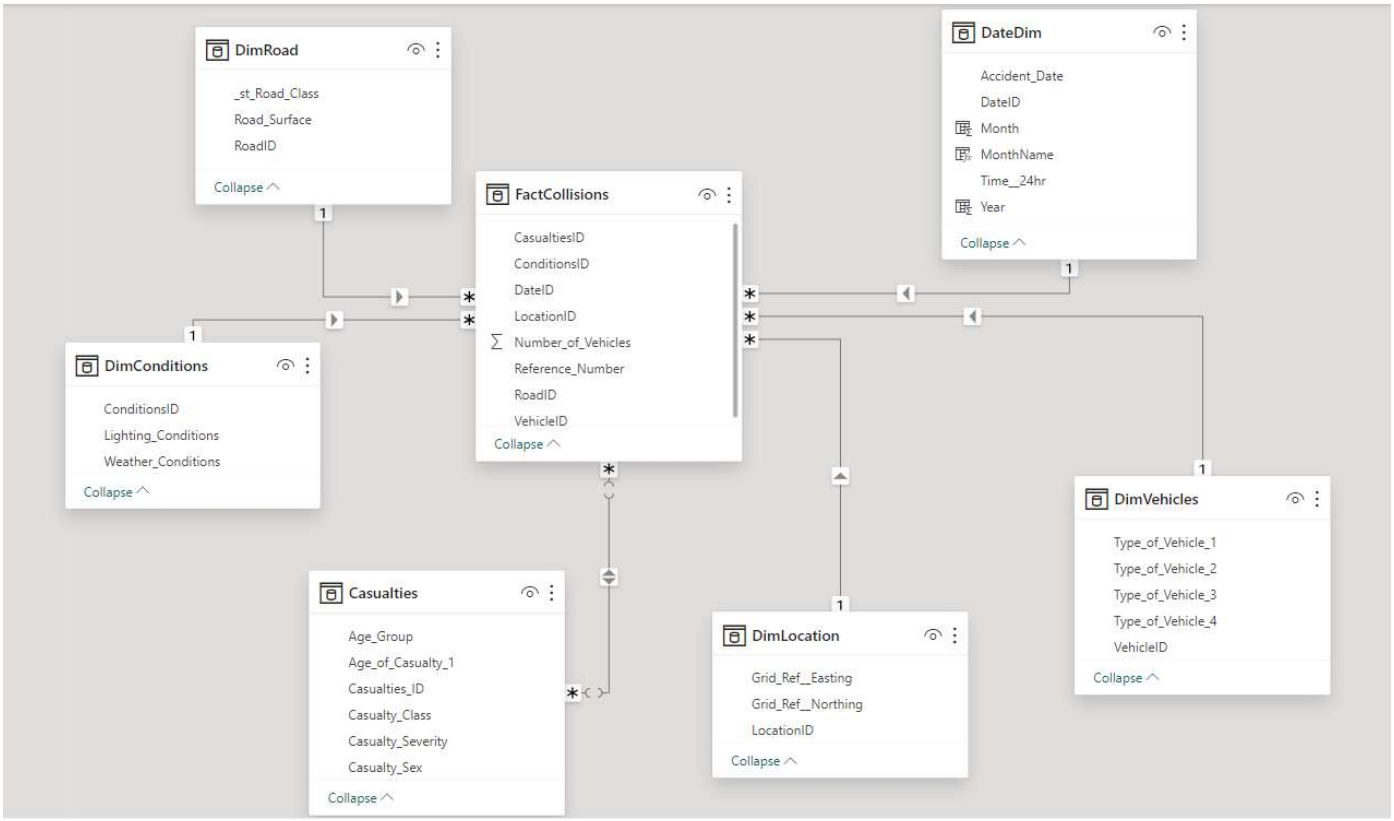
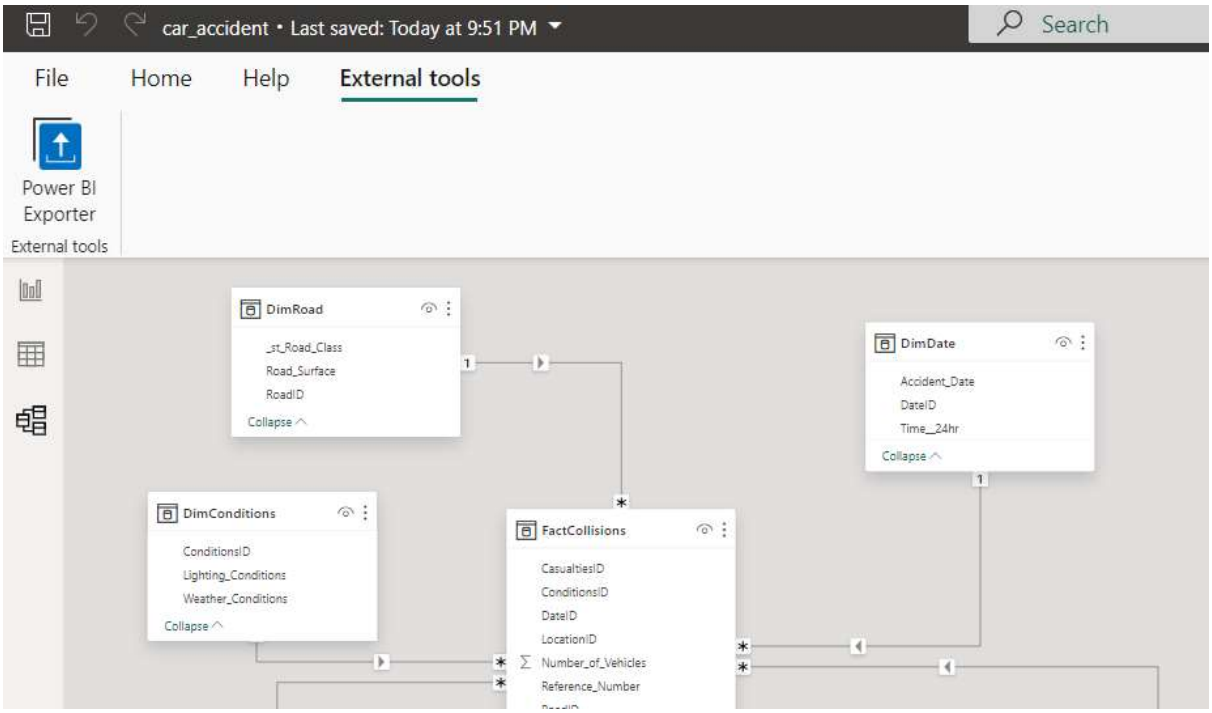


Figure 1: Data Warehouse Schema

- **Data Extraction**

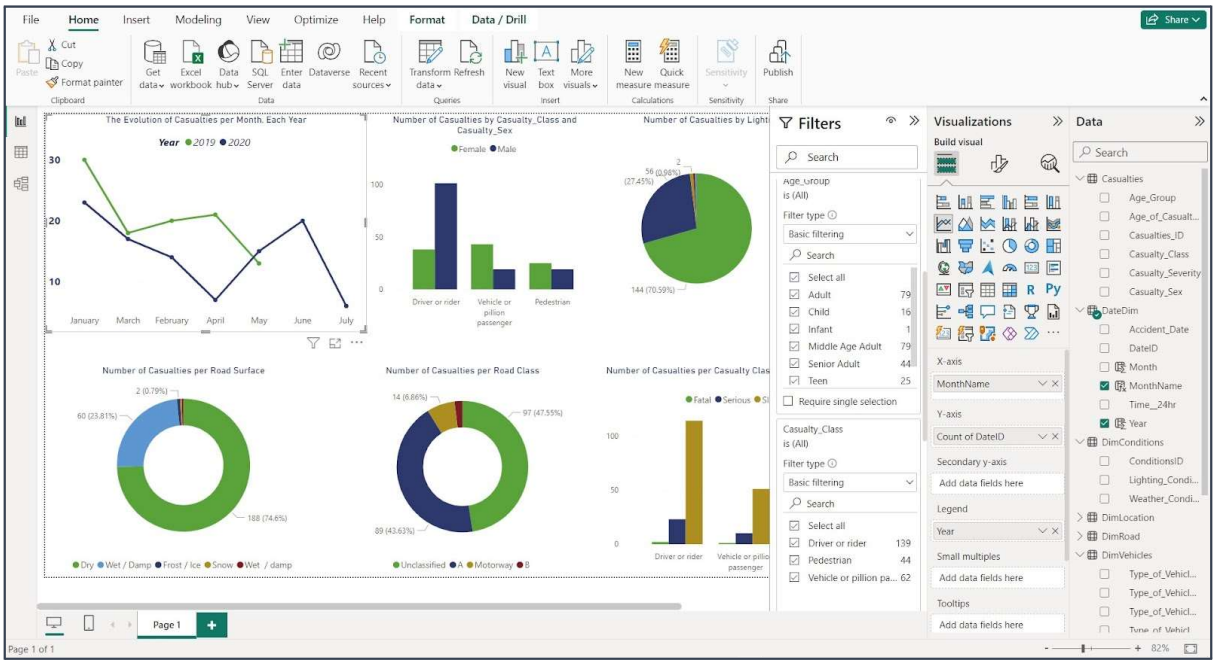
After Transforming the data according to the data model above, we used the external tool “Power Bi Exporter” to export the data.


car_accident • Last saved: Today at 9:51 PM
Search
File
Home
Help
External tools
Power BI Exporter
External tools
DimRoad
_st_Road_Class
Road_Surface
RoadID
Collapse
DimDate
Accident_Date
DateID
Time_24hr
Collapse
DimConditions
ConditionsID
Lighting_Conditions
Weather_Conditions
Collapse
FactCollisions
CasualtiesID
ConditionsID
DateID
LocationID
Number_of_Vehicles
Reference_Number

2.5 Data Visualization

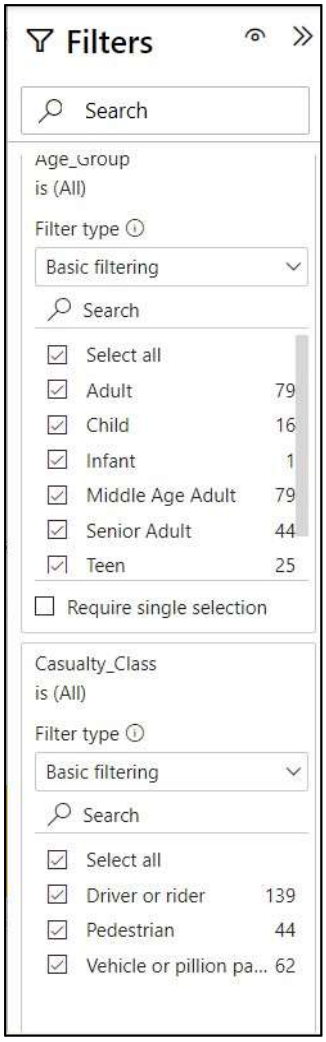
After the data warehouse creation and modeling of the different fact and dimension tables, the goal is to extract meaningful insights and relationships between our different variables that will allow us to infer and draw conclusions about them.

To create the dashboard we utilized **Power BI Imbedded Reports Builder:**



These are **the graphs** included in the Dashboard:

- The Evolution of Casualties per Month Each Year (2019 & 2020)(Line Chart)
- The Number of Casualties by Casualty Class and Casualty Sex (Clustered Column Chart)
- The Number of Casualties per Lighting Conditions(Pie Chart)
- The Number of Casualties per Road Surface (Donut Chart)
- The Number of Casualties per Road Class (Donut Chart)
- The Number of Casualties per Casualty Class and Casualty Severity (Clustered Column Chart)



Through **the use of various filters in our** dashboard we are able to make comparisons and draw more insightful conclusions.

Extracted Insights:

- There is a significant drop in casualties from 2019 to 2020 between the period ranging from January to May.
- The Number of Casualties associated with accidents Tends to Increase In The Summer.
- We can conclude that Male Drivers or Riders are more likely to cause accidents than Female Drivers. However the number of Female Passengers or Pedestrians Involved in Vehicles Accidents is greater than the number of males.
- Most of the Accidents occur in The Daylight causes approximately 71% of all the casualties, followed by accidents that occur through the night when lights are present and lit.
- About 23.81% of The total Casualties occur because of accidents on Wet/Damp Roads, which is approximately 1/3rd

of the casualties that occur because of accidents on dry roads.

- Vehicles Drivers or Riders are more likely to have fatal and Serious Casualties than Vehicle Passengers or Pedestrians.

3 Conclusion:

- The extracted Insights can be utilized by various organizations to enhance and improve their business operations. For Instance:
  - **Insurers** can refine risk assessment models, tailoring premiums based on seasonal and gender-specific driving behaviors.
  - **Transportation agencies** can allocate resources strategically, implementing targeted safety initiatives and enhancing infrastructure in identified accident-prone areas.
  - **Healthcare providers** preparing for potential increases in accidents can establish a more responsive and resilient road safety framework in the region.