UNIVERSITÀ DI PISA

LABORATORY OF DATA SCIENCE REPORT

A.Y. 23/24

Exploration of Gun Violence Incidents (2013-2018)

Camilla Chiruzzi[1] | **Diletta Ferri**[1]

[1]Department of Computer Science, MSc in Data Science and Business Informatics, University of Pisa

Project will be divided in three parts:

Part 1: create and populate a database starting from different files and perform some operations on it.

Part 2: solve the assigned problems using SQL Server Integration Services (SSIS) with computation on the client side.

Part 3: answer some business questions on a datacube created on the database.

Contents

1 Part 1 **2**
1.1 Assignement 0 - Create the database schema . . . 2
1.2 Assignment 1 - Data splitting and integration . . . 2
1.2.1 Geography information . . . 3
1.2.2 Date . . . 3
1.2.3 Other tables . . . 4
1.3 Assignment 2 - Database population . . . 4

2 Part 2 **5**
2.1 Assignment 0 . . . 5
2.2 Assignment 1 . . . 6
2.3 Assignment 2 . . . 7

3 Part 3 **7**
3.1 Assignment 0 . . . 7
3.2 Assignment 1 . . . 8
3.3 Assignment 2 . . . 9
3.4 Assignment 3 . . . 9
3.5 Assignment 4 . . . 10
3.6 Assignment 5 . . . 10

1 | PART 1

1.1 | Assignement 0 - Create the database 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 (See Fig. 1).

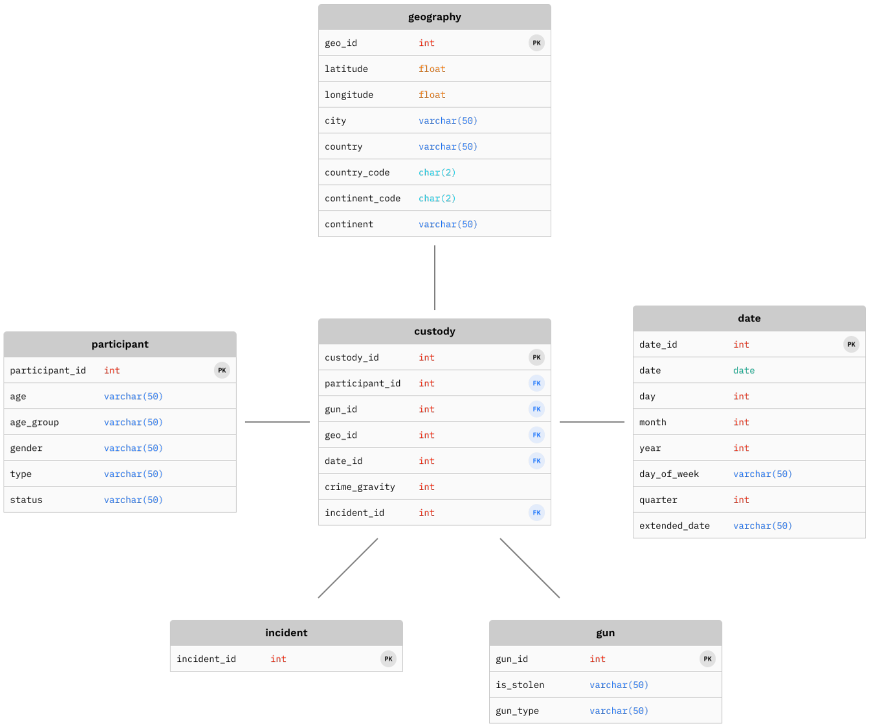


FIGURE 1 Datawarehouse schema

In particular, for certain attributes, special attention was given, specifically:

- **Geography:** the `country_code` and `continent_code` were set to `char(2)`. This decision was made to optimize storage efficiency by conforming to the standard that mandates a two-character code representation for countries and continents codes. This choice not only minimizes storage space but also aligns with the specified standard, ensuring consistency in data representation.
- **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.

1.2 | Assignment 1 - Data splitting and integration

In Assignment 1, the task consists in the creation of a Python program that splits the content of the different files (`'police.csv'`; `'dates.xml'` and three dictionaries: `'partecipant_age.json'`, `'partecipant_type.json'` and `'partecipant_status.json'`) and computes the six separate tables: custody, gun, participant, date, incident, and geography. The first step of the approach was the handling of geography information and date-related data, as they required specific methods for extracting necessary attributes in these dimensions. Then, a more generalized program was developed to simultaneously generate the remaining tables.

1.2.1 | Geography information

In terms of geographical granularity, the decision was made to maintain the highest level of detail, aware that it would compromise spatial aspects but enable data navigation from the continent down to specific coordinates. The approach followed is the creation of functions that incrementally help to achieve the goal, in particular the choice of these specific libraries is driven by their ability to make the code highly efficient, bringing down the total execution time.

1. Extraction of coordinates from the `'police.csv'` file. To handle duplicates, coordinate pairs were placed in a set. This function allows us to have all unique coordinate pairs.
2. Retrieval of geographical information using the `reverse_geocode` library (`rg`)[1]. Specifically, the use of `rg.search` allows obtaining country, city, and crucially, the `country_code` given the latitude/-longitude coordinates. It's important to note that before applying `reverse_geocode`, the set was converted into a list since the function does not accept sets as input.
3. Mapping of countries to their corresponding continents: using the `country_code`, the `country_alpha2_to_continent_code` function using the `pycountry_convert` library[2] was applied. Alpha 2 refers to the number of letters in the continent code. Specifically, SO 3166-1 alpha-2 codes are two-letter country codes defined in ISO 3166-1, part of the ISO 3166 standard. For example, `NA` stands for North America. Considering the continent code, the continent name was derived to have an attribute even clearer than the simple code. For example, given the continent code 'NA', the corresponding continent name could be derived as 'North America'.
4. The primary objective of the final step is to generate a CSV file encapsulating the geospatial data. Essentially, the `'geography.csv'` file will contain all the derived geographical information, including latitude, longitude, city, country, country code, continent code, and continent name. Additionally, an incremental `geo_id` has been generated to ensure a unique identifier for each location.

1.2.2 | Date

Concerning temporal aspects, the decision made was to retain as much information as possible. The two key steps in this process are represented by:

1. The use of the `xml.etree.ElementTree` (ET) module[3] to extract information from the `dates.xml` file, including dates and unique identifiers for each date (`date_pk`). Regarding the date format, which was in the form of '2013-01-01 00:00:00', the time component (given that it was '00' for all entries) was not considered.
2. The use of the `datetime` module[4] to extract the day, month, year, day of the week, and the quarter of the year (calculated with the formula `quarter_of_year` $= \frac{\text{month}-1}{3} + 1$). Additionally, information about the extended date format (e.g., '1 January of 2013') was included to improve the interpretability and human readability of the temporal data, so to foster a more intuitive understanding of the dates.

[1]Visit `reverse-geocode 1.4.1` for more details
[2]Visit `pycountry-convert 0.7.2` for more details
[3]Visit `xml.etree.ElementTree` for more details
[4]Visit `datetime` for more details

Finally, as for all other tables, a CSV file was generated.

1.2.3 | Other tables

Regarding the tables Participant, Gun, Incident and Custody, the majority of the information were readiliy available in the 'police.csv' file. Almost all the operations were executed while reading the 'police.csv' file, in order to reduce the computation power needed to open numerous times this file. The main steps needed were the computation of the crime_gravity index, the generation of incremental IDs for Participant and Gun and the creation of the CSV files for all tables.

Generation of incremental IDs

It was decided to generate incremental IDs for Participant and Gun, while for Incident and Custody the IDs already present in the original file were used (incident_id and custody_id).
In order to do so, a function was defined that was able to create a dictionary mapping the ID to the corresponding values of the different attributes. The function works by checking if the list of values read from the 'police.csv' and associated with that particular dimension is already present in the dictionary: if it is, it returns the corresponding ID, while if it's not it generates a new incremental one, exploiting the length of the dictionary. To avoid any possible ambiguity the decision was made to start these IDs from 1 rather than from 0.

Computation of crime_gravity

In order to compute the crime_gravity index the three available dictionaries were imported using the library json[5], opening and saving them as dictionary structures. Then, while reading the rows of the 'police.csv' file, the values of the participant_age_group, participant_type and participant_status were used to get the scores in the dictionaries and compute the index.

Creation of the CSV files

For the purpose of creating a CSV file for each table, different data structures were created, in particular dictionaries for participant, guns and custody. These dictionaries have as keys the IDS and as value a list of all the values of the attributes of that dimension. Reading the 'police.csv' file, the dictionaries and the index are computed and at the end, using a defined function create_csv, the CSVs are created. Regarding the incident table, even if it could be considered as a degenerate dimension, it was preferred to create a dimension table, so to guarantee that if additional information becomes available in the future, it can be easily added. In order to save the different values of incident_id, it was decided to place them in a set, and create the CSV separately.

1.3 | Assignment 2 - Database population

The approach followed for the final phase, concerning the database population, relies on creating a single function, populate_table. The function is designed to dynamically construct an SQL query based on the given table name and its associated fields. The code employs the pyodbc Python module[6] to establish a connection to the SQL Server database and utilizes the fast_executemany method[7] for optimized performance enhancing the overall speed of the process.

This modular and dynamic approach not only ensures code reusability but also facilitates the integration of additional tables in the future with minimal modification. The flexibility in constructing SQL queries based on varying table structures enhances the adaptability of the code to diverse datasets.

[5]Visit json for more details

[6]Visit pyodbc 5.0.1 for more details

[7]Visit cursor.executemany for more details

2 | PART 2

2.1 | Assignment 0

| For every type of gun, determine the number of custodies of young participants (<18)

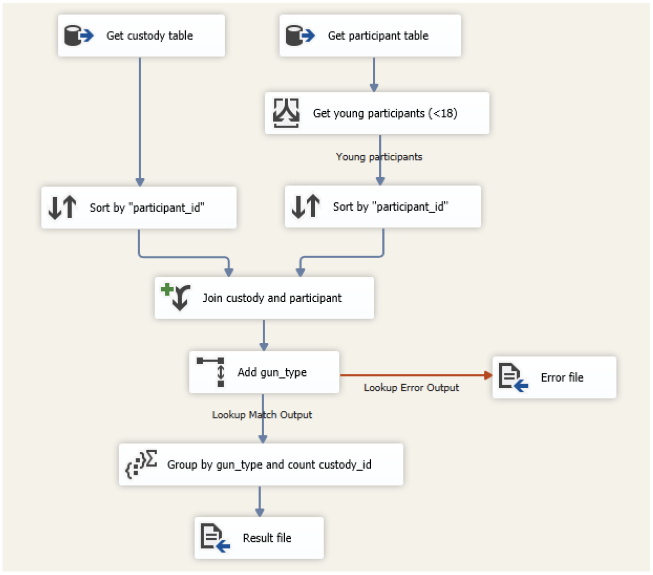


FIGURE 2 Assignment 0 - SISS schema

From the participant table get all the +18 participants by applying a conditional split checking the `age` attribute. This choice allows to better manage the following phases due to the reduced number of rows. Since the threshold between Adult and Young participants is 18 years of age, it is noteworthy the decision to consider the `age` column instead of the `age_group` one to better generalize the distinction between young and adult among different countries (different regulations set the adult age differently).

After sorting both custody and young participants tables by `participant_id`, the merge join operation was computed taking as join key `participant_id`, retaining as output only `custody_id`, `gun_id` and `participant_id`.

A lookup node was connected to add the `gun_type` information and, after that, an aggregation function was applied to group by `gun_type` and count the distinct `custody_id`. The use of lookup instead of a merge join, which would require another connection to the database and a sorting, is to reduce the number of steps needed to get the `gun_type` information.

It is important to deal with the possibility of missing `gun_type`. This issue is managed by redirecting the rows with no matching entries to an error output that automatically populates a flat file.

2.2 | Assignment 1

| A city is classified as dangerous if the total gravity of custodies within that city is higher than the average gravity for that city's state. List all the dangerous cities.

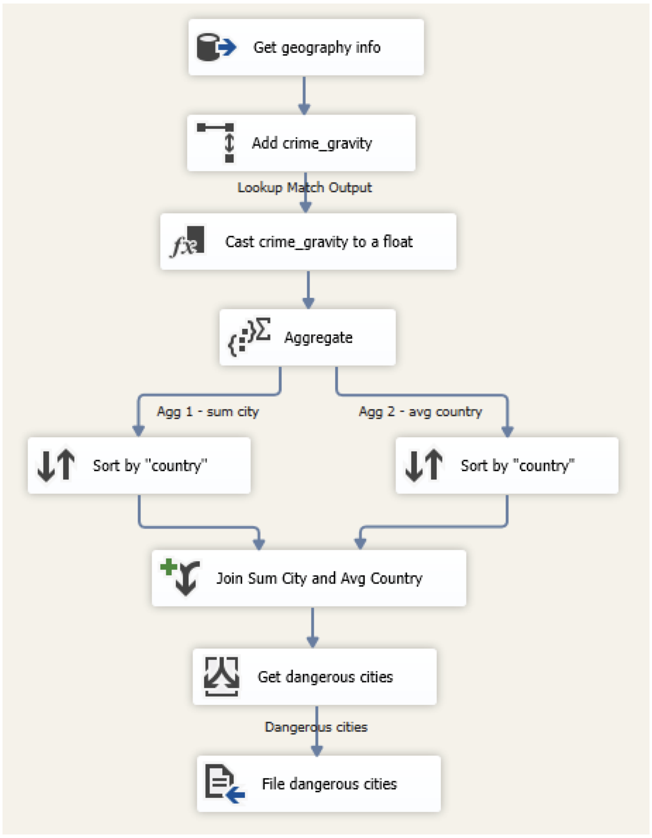


FIGURE 3 Assignment 1 - SISS schema

From the geography table get `geo_id`, `city` and `country`. Additionally, adding a lookup node, the `crime_gravity` information was taken from the custody table. Using a node calculating a derived column, the `crime_gravity` index was casted, converting the attribute from an integer to a float. This allows the mean operation, useful in the next phases, to have also the decimal part. This guarantees a non excessive approximation of the results. Two aggregate operations were created through a unique aggregate node:

1. Agg1: sum `crime_gravity` grouping by `city` and `country`;
2. Agg2: average `crime_gravity` grouping by `country`.

Before merge joining the results, using `country` as key, the aggregations results were necessary both sorted. The join operation allows to pair the total gravity of each city (Agg1 results) with the average crime_gravity (Agg2 results) of the corresponding state.

After merging, a conditional split was applied to divide between dangerous and not dangerous cities, by checking if the total gravity of that city is higher than the average gravity for that city's state. The dangerous cities were then saved in a flat file.

2.3 | Assignment 2

| For each state, determine the gun type with the highest total gravity

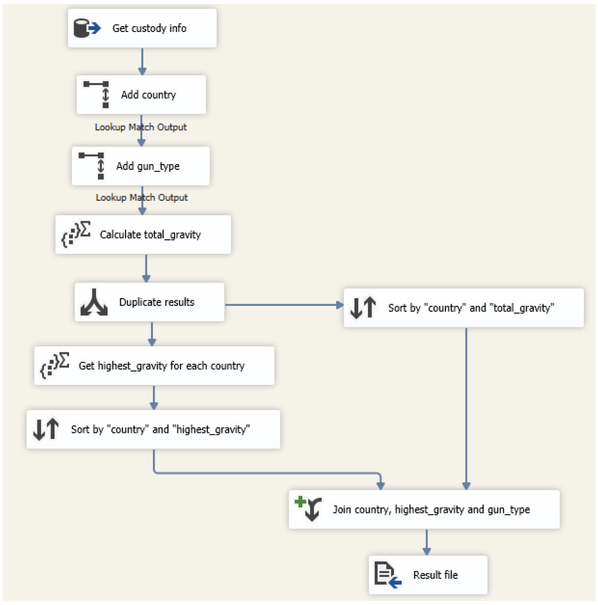

Get custody info
Add country
Lookup Match Output
Add gun_type
Lookup Match Output
Calculate total_gravity
Duplicate results
Sort by "country" and "total_gravity"
Get highest_gravity for each country
Sort by "country" and "highest_gravity"
Join country, highest_gravity and gun_type
Result file

FIGURE 4 Assignment 2 - SISS schema

From the custody table get `gun_id`, `geo_id` and `crime_gravity`. With two consecutive lookup nodes, `country` and `gun_type` were added, then an aggregate node is used to calculate the total crime_gravity grouping by `country` and `gun_type`.

Data was duplicated through a multicast node to separate in two branches the workflow, in order to:

1. Take the highest crime gravity grouping by `country`;
2. Maintain the original information to subsequently merge these two branches not losing the `gun_type` information.

In order to merge the two branches, they were both sorted by country and `total_gravity`, which are then used as keys in the merge join. As a result we get all the states with the associated `gun_type` with the highest total gravity.

3 | PART 3

3.1 | Assignment 0

| Build a datacube from the data of the tables in your database, defining the appropriate hierarchies. Create the needed measures based on the queries you need to answer

To create the datacube the choice was to keep all the attributes of every dimension instead of considering only the ones needed to answer the given business questions. This approach enables the system to answer potential future business inquiries.

The dimensions were set up as follows:

- **Date**: `date_id` was used as key, and `date` as the minimum detail element. The choice was to set up two distinct hierarchies:
  - The Date Hierarchy: the granularity spans from year to the date;
  - The Day Week: considers the day and the resultant day of week.

  For the second one, the column `day_of_week`, with values ranging from 'Monday' to 'Sunday', was utilized to generate an attribute designed to facilitate the sorting of days (using 'new calculation'). While this attribute is typically sorted alphabetically by default, the new attribute was created to adapt to potential future MDX queries searching for example the "value of the preceding one". This attribute was generated using an expression that assigns an ascending number from one to seven to each day of the week (e.g., 'Monday' = 1, 'Sunday' = 7). The ordering of days (Day of week) was then set using the numeric attribute in the advanced properties. Almost the same methodology was applied to the "Date Hierarchy" for the months. In this instance, months were represented as numbers from 1 to 12. An attribute extraction was performed through an expression associating the number with each month's name (1 = 'January', 12 = 'December'). Subsequently, the ordering was set based on the numbered months to preserve the temporal dimension
- **Geography**: `geo_id` was used as key and `city` as the minimum detail element
- **Incident**, **Participant**, **Gun**: these dimensions only required to set up the dimensions without the need to define any hierarchies, so just the keys and the attributes were specified

Measures

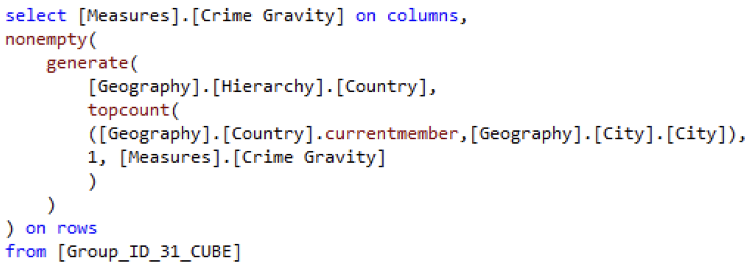
The only measures used were the ones automatically created by the system when the datacube was created. The two measures are:

- `Crime Gravity`: it's already present in the fact table, so it's translated into a measure
- `Custody Count`: contains the number of rows of the table `Custody`

3.2 | Assignment 1

| Query 1 - Get the city with the highest crime gravity for each country

In order to get the city with the highest crime gravity for each country, the `topcount` function was used, in combination with the `generate` function. This allows to consider for each country all the different cities, each with its total `Crime Gravity`, and returns the Top 1, so the one with the highest value of the measure.



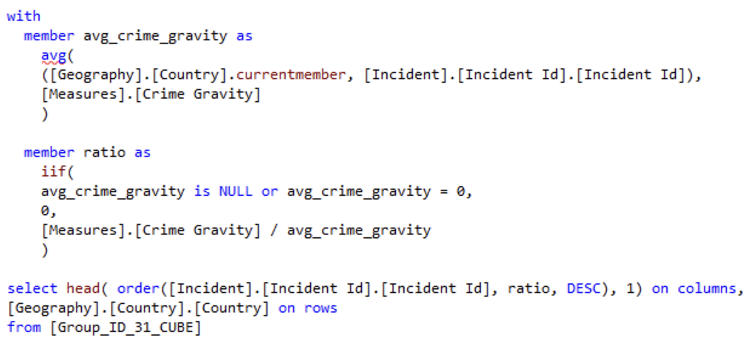
```
select [Measures].[Crime Gravity] on columns,
nonempty(
    generate(
        [Geography].[Hierarchy].[Country],
        topcount(
        ([Geography].[Country].currentmember,[Geography].[City].[City]),
        1, [Measures].[Crime Gravity]
        )
    )
) on rows
from [Group_ID_31_CUBE]
```

FIGURE 5 Assignment 1 - MDX Query

3.3 | Assignment 2

| Query 2 - For each country, get the incident with the highest ratio between his total crime gravity and the average crime gravity of that country

In order to answer this business question, two members were created: `avg_crime_gravity` calculates the average Crime Gravity for each country, `ratio` computes the ratio between each incident's Crime Gravity and the average crime gravity of the corresponding country.
Using the function `Order`, the `Incident Id` were sorted in descending order with respect to the measure `ratio`, and then, using the function `head`, the first result, so the one with the highest ratio, was showed. It is important to note that in the calculation of the `ratio`, a check on the value of `avg_crime_gravity` was added: if it is null or equal to 0 then the `ratio` is also set to 0.



```
with
  member avg_crime_gravity as
    avg(
    ([Geography].[Country].currentmember, [Incident].[Incident Id].[Incident Id]),
    [Measures].[Crime Gravity]
    )

  member ratio as
    iif(
    avg_crime_gravity is NULL or avg_crime_gravity = 0,
    0,
    [Measures].[Crime Gravity] / avg_crime_gravity
    )

select head( order([Incident].[Incident Id].[Incident Id], ratio, DESC), 1) on columns,
[Geography].[Country].[Country] on rows
from [Group_ID_31_CUBE]
```

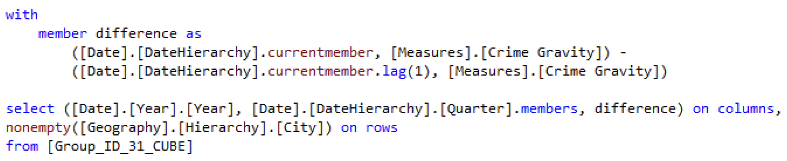
FIGURE 6 Assignment 2 - MDX Query

3.4 | Assignment 3

| Query 3 - For each city, get the difference between each quarter's total crime gravity and the previous quarter's total crime gravity

To obtain the difference between each quarter's total crime gravity and value of the same measure in the previous quarter, the member `difference` was created: it contains the difference of the measure `Crime Gravity` calculated between the date considered at each iteration and the previous one. In order to get the value of the measure for the previous quarter, the function `lag` with parameter 1 was used. This is equivalent to the `prevmember` function, but it was chosen because it is more general, and it can be used for future further comparisons: by changing the value of the parameter, we can get the difference with other quarters (for example with parameter = 4 we can get the difference with the same quarter of the previous year) without changing anything else.

In the result of the query, there are many `(null)` values, this happens when the data for one of the considered quarters is not available.



```
with
    member difference as
        ([Date].[DateHierarchy].currentmember, [Measures].[Crime Gravity]) -
        ([Date].[DateHierarchy].currentmember.lag(1), [Measures].[Crime Gravity])

select ([Date].[Year].[Year], [Date].[DateHierarchy].[Quarter].members, difference) on columns,
nonempty([Geography].[Hierarchy].[City]) on rows
from [Group_ID_31_CUBE]
```

FIGURE 7 Assignment 3 - MDX Query

3.5 | Assignment 4

| Create a dashboard that shows the geographical distribution of the total crime gravity in each age group

Regarding the reporting phase, the Power BI tool has allowed, through its visual component, to represent an important aspect concluding this exploration.
In particular, the underlying dashboard focuses on the geographical distribution of crime gravity.
Regarding the consideration in the analysis of the age groups of participants (Adult, Teen and Child), it can be immediately noticed that the vast majority (consistently exceeding ~75%) of crimes are committed by adults, followed by teens and a negligible portion by children, and this is consistent for all countries. It can be observed that the discussion pertains to North America, and by looking at the map in the bottom right corner, the overwhelming majority of crimes are committed in the United States or its vicinity. For Canada and Mexico, the distribution of crimes also concentrates along their borders with the United States, revealing a significant imbalance in terms of both quantity and spatial distribution among the three involved countries.
Although not strictly required in assignment four, this element of the dashboard has been highlighted for completeness. As will be evident in the following dashboard (see Fig. 9 and 10), it is a factor that significantly influences the overall analysis, particularly emphasizing the weight that the US carries.

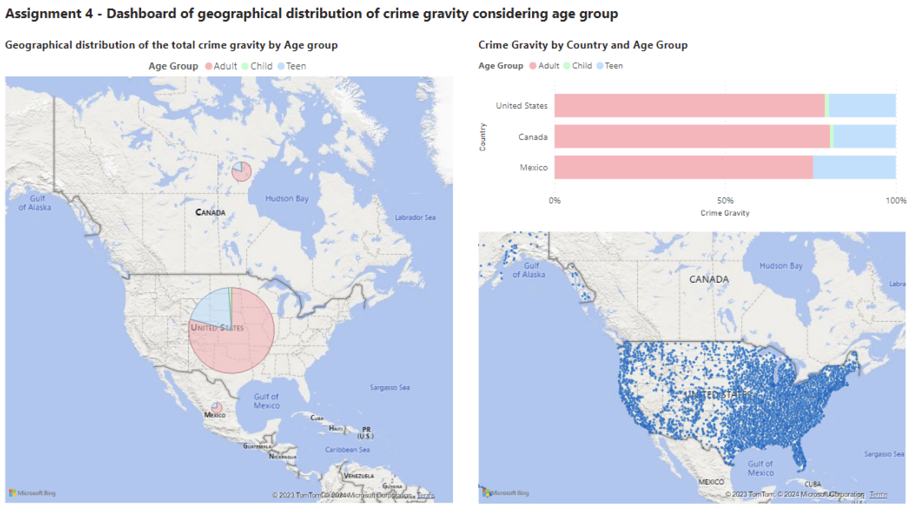

Assignment 4 - Dashboard of geographical distribution of crime gravity considering age group
Geographical distribution of the total crime gravity by Age group
Age Group ● Adult ● Child ● Teen
Crime Gravity by Country and Age Group
United States
Canada
Mexico
Country
0%
50%
100%
Crime Gravity
Canada
United States
Mexico
Hudson Bay
Gulf of Alaska
Gulf of Mexico

FIGURE 8 Dashboard assignment 4

3.6 | Assignment 5

| Create a plot/dashboard of your choice that you deem interesting w.r.t. the data available in your cube

Geographic dimension has been addressed, and the focus has shifted to the temporal dimension. The following dashboard primarily considers the evolution of crime gravity over various days of the week and across quarters of the year, with detailed insights into the weapons in relation with the severity of the crimes.
Regarding the first two graphs at the top, on the left, a stacked column chart, and on the right, an area

chart, this choice was made to consider the gender of the participants. It can be observed that crime gravity tends to remain stable for women on a weekly basis. However, variations are noted for men, who are disproportionately involved in crimes, particularly with an increase in the middle of the week, peaking on Wednesdays. This trend in crime gravity does not follow the registered custodies, which increase over the weekend, without a significantly higher crime gravity. So, on weekends, there is a decrease in crime gravity and an increase in custodies, which implies more criminal events, but of lower severity.
Concerning the different quarters of the year represented in the area chart, criminal events involving men show a decrease in the second quarter while remaining stable for women. In relation to custodies, it was decided to analyze its typology through a clustered column chart. It is interesting to note an increase in injured subjects over the weekend, and in this graph, there is also a peak on Wednesdays, particularly in arrests. Regarding the weapons involved in crimes represented in the treemap, Handgun has the highest crime gravity, followed by 9mm, unknown guns, and Rifle.

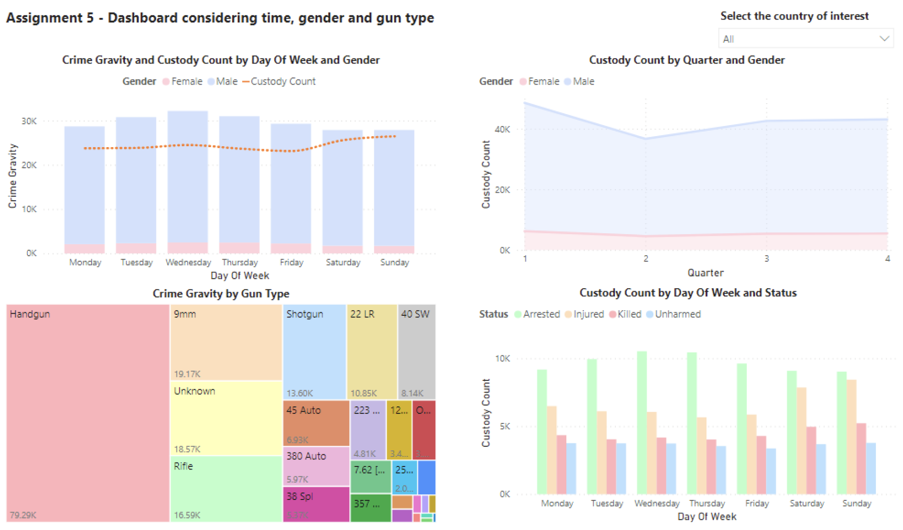


FIGURE 9 Dashboard assignment 5

To conclude this analysis, it is relevant to revisit the aspect mentioned in the previous section regarding the differences among the different countries and the prevalence of data on the United States. Using a slicer, it was possible to filter the data by countries (see Fig. 10), and as evident, the United States aligns with the aggregated data for all countries, dominating data almost entirely. As for Mexico (in the center) and Canada (on the right), the graphs exhibit changes. The decrease in criminal events for men in Mexico is observed in the third quarter instead of the second, increasing for women. It is also interesting to note the absence of injuries on Mondays in Mexico and a significant peak in arrests on Saturdays.

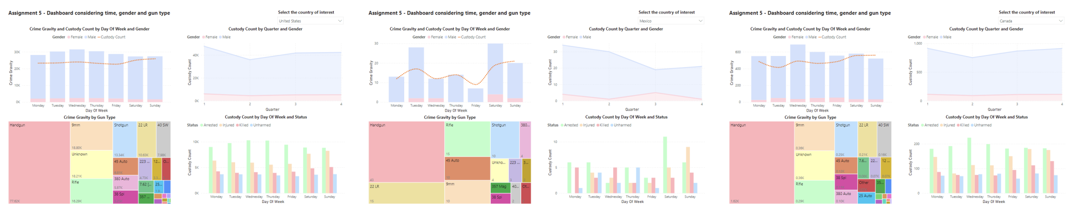


FIGURE 10 Differences among different countries