
UNIVERSITÀ DI PISA

**Laboratory of Data Science**

Project Report

# Gun Violence Data Warehouse

**Group ID - 200:**

Mohamed Arafaath - 659588

Mohan Upadhyay - 667179

---

**Academic Year 2023/2024**

# Contents


**1 Part 1 - Datawarehouse building** **1**
- 1.1 Data Understanding and Pre-processing . . . 1
- 1.2 Implentation . . . 1
  - 1.2.1 Assignment - 0: Creating Database Schema . . . 1
  - 1.2.2 Assignment - 1: Splitting and Integration of Data . . . 1
  - 1.2.3 Assignment - 2: Populating the Database . . . 2
  - 1.2.4 Conclusion: . . . 3

**2 Part 2 - SSIS Solution** **3**
- 2.1 SSIS Data Flows . . . 3
  - 2.1.1 Assignment - 0 . . . 4
  - 2.1.2 Assignment - 1 . . . 4
  - 2.1.3 Assignment - 2 . . . 6

**3 Part 3 - Multidimensional Data Analysis** **7**
- 3.1 Creation of the OLAP Cube . . . 7
  - 3.1.1 Connecting with Database and creation of View . . . 7
  - 3.1.2 Creation of Dimensions . . . 7
- 3.2 MDX Query . . . 7
  - 3.2.1 Assignment - 0 . . . 7
  - 3.2.2 Assignment - 1 . . . 8
  - 3.2.3 Assignment - 2 . . . 8
  - 3.2.4 Assignment - 3 . . . 9
- 3.3 Dashboard - Power BI Solution . . . 9
  - 3.3.1 Assignment - 4 . . . 9
  - 3.3.2 Assignment - 5 . . . 10

# 1 Part 1 - Datawarehouse building

In the first part of the project, the objective was to build a data warehouse using SQL Server Management Studio on the server lds.di.unipi.it. The resulting schema, as depicted in Figure-1, was created under the database named GroupID_200_DB. This schema serves as the foundation for organizing and storing data related to gun violence incidents in the United States between January 2013 and March 2018.

## 1.1 Data Understanding and Pre-processing

We conducted initial data exploration and preprocessing activities to identify any instances of missing values or redundant data as follows:

- The `police.csv` dataframe has 11 columns and no missing value was found
- The same dataframe has a total of 170928 rows:
  - `custody_id` has 170928 unique values
  - `incident_id` has 105168 unique values
  - `date_fk` has 1634 unique values
  - `latitude & longitude` has 65553 & 68607 unique values respectively
  - Rest were categorical attributes with finite unique values.
- Dictionaries for `Partecipant age, status` and `type` with numerical values to be mapped to actual values in police.csv to caluclate crime_gravity value for each record
- `dates.xml` file to map dates to the `date_fk` in the `police.csv` file

## 1.2 Implentation

### 1.2.1 Assignment - 0: Creating Database Schema

The project commenced with Assignment 0, where the task was to create the database schema outlined in Figure-1 using SQL Server Management Studio on the server lds.di.unipi.it. The designated name for the database was GroupID_200_DB.

#### 1.2.1.1 Schema Creation:

Utilizing SQL Server Management Studio, the schema was meticulously designed and implemented to accommodate the data related to gun violence incidents between January 2013 and March 2018. Six tables were created, namely Custody, Gun, Partecipant, Date, Incident, and Geography, in accordance with the specified schema.

#### 1.2.1.2 Database Naming Convention:

The naming convention adhered to the guidelines, with the database named GroupID_200_DB. The successful completion of Assignment - 0 laid the groundwork for subsequent tasks, providing a well-defined database structure that would serve as the repository for the project's data. This schema was pivotal in organizing and managing the diverse information extracted from the `Police.csv` and `dates.xml` files during the subsequent assignments.

### 1.2.2 Assignment - 1: Splitting and Integration of Data

To achieve the schema structure outlined in Figure-1 , a Python program was developed without utilizing the pandas library. The primary focus was on processing the Police.csv and dates.xml files, as well as leveraging dictionaries (partecipant_age.json, partecipant_status.json, partecipant_type.json) to calculate the crime gravity attribute.

#### 1.2.2.1 Parsing Dates and Creating Date Mapping:

A function was implemented to compute additional date-related data using the datetime module. Another function parsed dates.xml to create a mapping of date_fk to real date.

#### 1.2.2.2 Computing Crime Gravity:

A function, compute_crime_gravity, was defined to calculate the crime gravity using provided dictionaries.

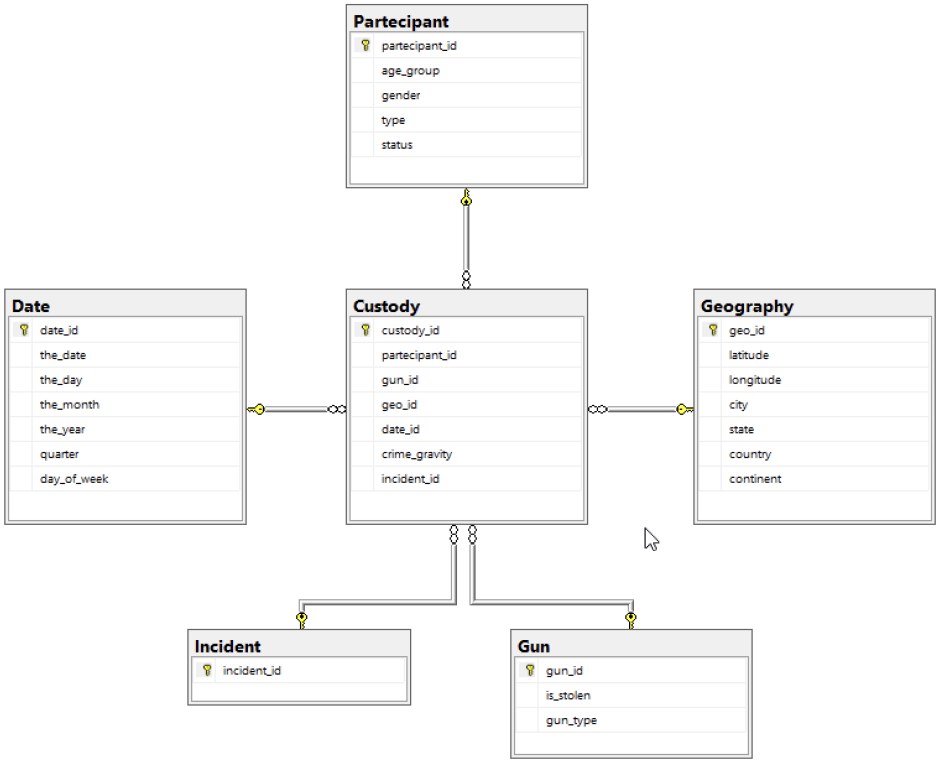


Figure 1: DB Schema obtained in Microsoft SQL Server Management Studio

#### 1.2.2.3 Geocoding and Location Information:

A function, get_location_info_with_retry, was created to retrieve location information using the Google Maps Geocoding API. Retry logic was implemented to handle potential errors.

#### 1.2.2.4 Loading Data into Database Tables:

The main function, split_and_integrate, processed the Police.csv file, split it into six separate tables (Custody, Gun, Partecipant, Date, Incident, Geography), and inserted data into the database. Special attention was given to generating missing IDs (partecipant_id, gun_id and geo_id) and computing the crime gravity attribute. Batch processing was utilized for efficiency, committing records in batches to avoid performance issues.

#### 1.2.2.5 ID Handling Functions:

Two functions, insert_data_with_ID and insert_data_without_ID, played a crucial role in managing IDs during data insertion. insert_data_with_ID was employed for tables with predefined IDs (Date, Custody, Incident). It ensured that duplicate entries were not inserted, maintaining data integrity. insert_data_without_ID was utilized for tables with auto-incremental IDs (partecipant, Gun, Geography). This function facilitated the insertion of data, generating and fetching the last inserted row's ID. The detailed implementation of these functions contributed to the seamless integration of data into the designated database tables, aligning with the project requirements.

### 1.2.3 Assignment - 2: Populating the Database

With the data prepared in Assignment 1, the process of populating the GroupID_200_DB database on the server lds.di.unipi.it was undertaken. This involved establishing schema relations as appropriate, connecting to the database, and executing SQL queries to insert the prepared data into the respective tables.Several methods were explored and documented in the project report, each representing a distinct attempt at optimizing the data population process.

#### 1.2.3.1 Initial Attempt: Simultaneous Upload with Python-Initiated IDs

In the first attempt, six different CSV files corresponding to the six tables were created. For three tables with auto-incremental IDs (gun_id, partecipant_id, and geo_id), a counter was initiated from 0 in the Python code. Initially, an effort was made to upload all 170,000 records simultaneously to the database.

**Challenges:**

Storing records in CSV files and then reading them into the database proved time-consuming. This approach faced significant challenges in terms of upload times, exceeding three days.Interruptions during the process led to data loss and inconsistency.

#### 1.2.3.2 Second Attempt: SSMS Auto-Incremental ID Integration

In the second attempt, a crucial modification was made directly in SQL Server Management Studio (SSMS). Instead of relying on auto-incremental IDs initiated from Python, the decision was made to utilize the inherent auto-incremental functionality of the SQL Server database for three tables (gun_id, partecipant_id, and geo_id).

**Key Modifications:**

In SSMS, the ID columns for gun_id, partecipant_id, and geo_id were configured with the "IDENTITY(1,1)" property, enabling SQL Server to automatically manage and increment these IDs. This modification eliminated the need for Python to initiate IDs and relied on the database's native auto-incremental capabilities.

**Outcome:**

This adjustment significantly improved efficiency, and records were inserted into the three tables seamlessly, But still it was very time consuming because of the Geopy library we used.

#### 1.2.3.3 Third Attempt: Batched Upload with Attribute Skip

Acknowledging the challenges faced in the second attempt, a third strategy was employed. In this attempt, the decision was made to skip the upload of city and state details for all latitude and longitude records. Moreover, the approach was shifted from attempting to upload all 170,000 records simultaneously to batching records and committing in batches of 10k.

**Key Modifications:**

The omission of city and state details significantly improved the upload speed, completing the process in less than 2 hours. Batching records in groups of 10k prevented data loss and allowed easy continuation from the last successfully uploaded records.

#### 1.2.3.4 Fourth Attempt: Optimizing Location Upload with Google Maps API

Recognizing the time-consuming nature of location data retrieval, a fourth attempt involved optimizing the upload process.

**Key Modifications:**

Experimenting with various libraries (geopy, reverse-geocoder) for geolocation data. Ultimately choosing Google Maps API for its superior speed and accuracy in retrieving city and state information. Manually adding country (USA) and continent (North America) for each record where states were already in the USA.

**Outcome:**

This final modification streamlined the location upload process, ensuring both speed and accuracy.

### 1.2.4 Conclusion:

Assignment 2 involved iterative exploration of methods to optimize data population. The refined third attempt, combining batch processing and attribute skipping, significantly enhanced efficiency. The final modification, using the Google Maps API, struck a balance between speed and accuracy, successfully populating the GroupID_200_DB database.

# 2 Part 2 - SSIS Solution

## 2.1 SSIS Data Flows

In order to build the data flows required for the seccond assignment, we first opened Visual Studio 2019 and created an Integration Service Project named 2nd_Project. Afterwards, we've initiated three packages: Assignment_0.dtsx, Assignment_1.dtsx and Assignment_2.dtsx.

### 2.1.1 Assignment - 0

**Problem : For every year, compute the number of total custodies.**

Calculate the annual total custodies, providing a comprehensive overview of custody trends over the years. The following steps were taken to achieve this, formatted as per the report structure:

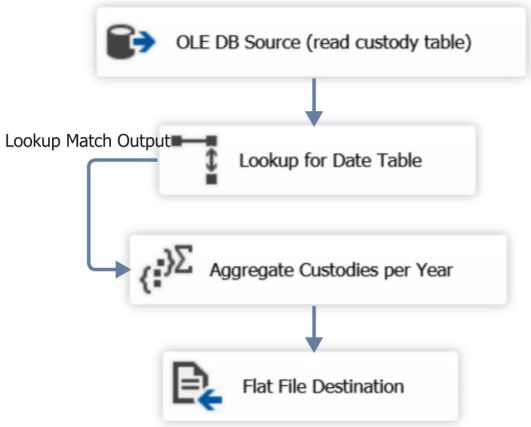



Figure 2: Assignment 0 - Data Flow Diagram

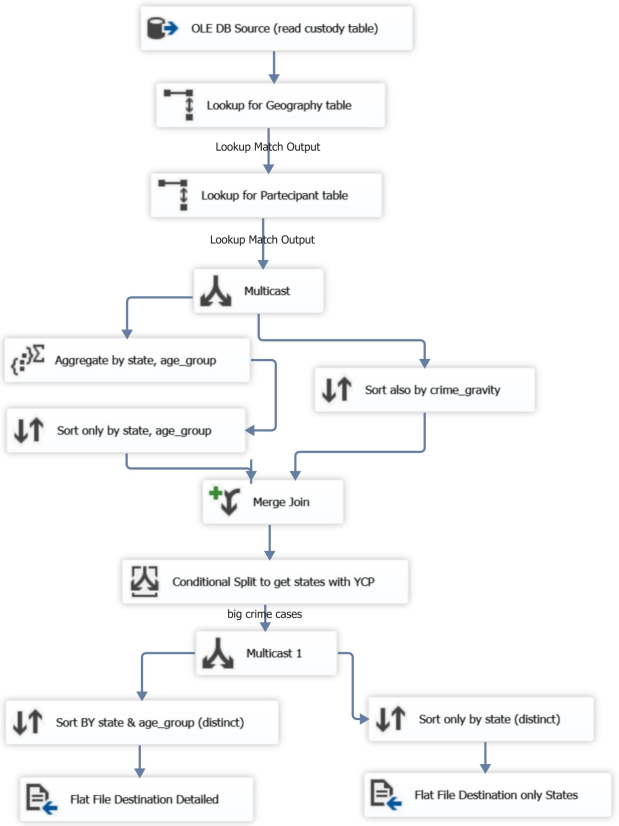
- **Data Flow Task:** Created a data flow task within the control flow of the `Assignment_0` package.
- **OLE DB Source:** Connected to the Custody table to retrieve information on `custody_id`, and `date_id`, which are essential for this task.
- **Lookup Transformation:** Connected to the Date table to retrieve the year corresponding to each custody record.
- **Aggregation Transformation:** Utilized the Aggregate Transformation Editor to group custodies by year and calculate the count of custodies for each year.
- **Flat File Destination:** Employed the Flat File Destination Editor to write the aggregated results to a CSV file named `Assignment_0.csv`, which includes columns for the year and the corresponding custody count.
- **Execute Process Task:** In the control flow, connect the Data Flow Task to an Execute Process Task. This task opens the result file automatically after package execution. The file path and executable file (Excel, in this case) were specified in the task configuration.

### 2.1.2 Assignment - 1

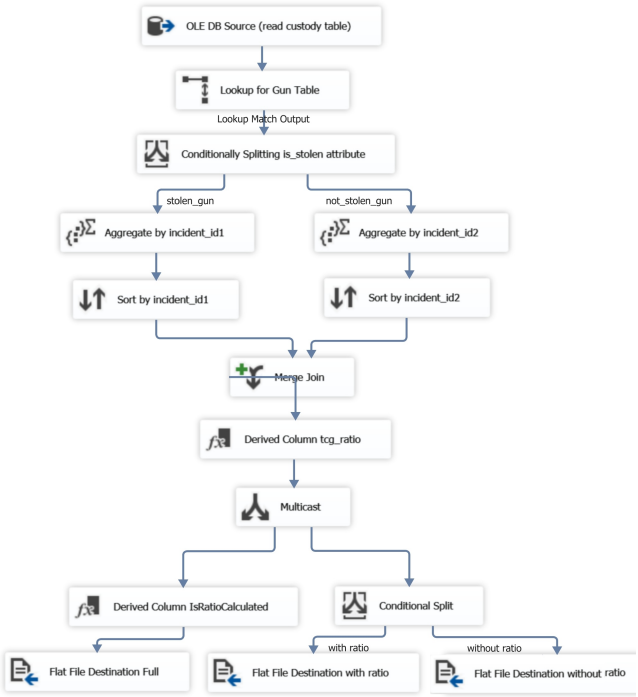
**Problem: A state is considered to have a youth criminal problem if the age group with the highest overall gravity consists of individuals under 18. List every state with a youth criminal problem.**

For this task, the objective was to determine states with a youth criminal problem, defined as having the highest overall gravity within the age group under 18. The following steps were taken to achieve this, formatted as per the report structure:

- **Data Flow Task:** Created a data flow task within the control flow of the `Assignment_1` package.
- **OLE DB Source:** Connected to the Custody table to retrieve information on `partecipant_id`, `geo_id`, and `crime_gravity`, which are essential for this task.
- **Lookup Transformation:** Looked up the geography table to retrieve the attribute `state`. Additionally, looked up the partecipant table to retrieve the attribute `age_group`.
- **Multicast Transformation:** The records undergo multicast into two distinct paths:
  1. **Left Path:**
     - **Aggregation:** Conduct aggregation on `crime_gravity` to determine the maximum gravity.
     - **Grouping:** Group the records by `state` and `age_group`in ascending order.
  2. **Right Path:**
     - **Sorting:** Sort the records by `state` and `age_group` in ascending order.
     - **Secondary Sorting:** Within the right path, perform a secondary sorting by `crime_gravity` in descending order.
- **Merge Join Transformation:** Merged the left and right paths using the Merge Join Transformation Editor with an inner join on common attributes (`state`, `age_group`, and `crime_gravity`). This ensures only matching records are included.




(a) Assignment 1 - Data Flow Diagram




(b) Assignment 2 - Data Flow Diagram

Figure 3: Data Flow Diagrams

- **Conditional Split Transformation:** Filtered out records where `crime_gravity` is maximum for the state and age group under 18, identifying states with youth criminal problems.
- **Multicast Transformation:** The records undergo multicast into two paths:
  1. **Left Path:**
     - **Sorting:** Sort records by `state` and `age_group` in ascending order, maintaining distinct rows.
  2. **Right Path:**
     - **Sorting:** Sort records by `state` in ascending order, maintaining distinct rows.
- **Flat File Destinations:** Created two flat file destinations to write results:
  1. `Assignment_1_full.csv` - Includes columns `state`, `age_group`, and `crime_gravity`, providing detailed information on states with youth criminal problems.
  2. `Assignment_1_states.csv` - Includes only the `state` column, offering a concise list of states with youth criminal problems.
- **Execute Process Task:** Connected the Data Flow Task to an Execute Process Task in the control flow. This task opens the result files automatically after package execution. The file paths and executable file (Excel, in this case) were specified in the task configuration, allowing the client to view the files separately and choose based on their preference.

This sequence of steps ensures a comprehensive analysis of states with youth criminal problems, providing flexibility for users to explore detailed or summarized results.

### 2.1.3 Assignment - 2

**Problem: For each incident, compute the ratio between the total gravity of crimes with a stolen gun and the total gravity of crimes with a not-stolen gun.**

Objective: Determine the ratio between the gravity of crimes involving stolen guns and not-stolen guns for each incident, offering insights into the comparative severity of these incidents. The following steps were taken to achieve this, formatted as per the report structure:

**Note:** In this report, the attributes prefixed with "tcg_" represent values related to Total Crime Gravity. Throughout the report, you will encounter attributes such as `tcg_stolen_gun`, `tcg_not_stolen_gun`, and `tcg_ratio`.

- **Data Flow Task:** Created a data flow task within the control flow of the `Assignment_2` package.
- **OLE DB Source:** Connected to the Custody table to retrieve information on `gun_id`, `crime_gravity`, and `incident_id`, which are essential for this task.
- **Lookup Transformation:** Looked up the Gun table to retrieve the boolean attribute `is_stolen`.
- **Conditional Split Transformation:** Split the data into two paths based on the condition of `is_stolen` (true for stolen_gun, false for not_stolen_gun).
- **Aggregation Transformation:** Calculated the sum of crime gravity on both paths to get `tcg_stolen_gun` and `tcg_not_stolen_gun`, grouped by incident_id.
- **Sort Transformation:** Sorted data on incident_id on both paths in ascending order.
- **Merge Join Transformation:** Merged the two paths using the Merge Join Transformation Editor with an inner join on common incident_ids. Included `tcg_stolen_gun` and `tcg_not_stolen_gun` for further processing.
- **Derived Column Transformation:** Created a derived column `tcg_ratio` representing the ratio between stolen and non-stolen guns. The calculation is expressed by the formula:

$$\text{tcg\_ratio} = \text{ROUND}\left(\frac{\text{tcg\_stolen\_gun}}{\text{tcg\_not\_stolen\_gun}}, 1\right)$$

  The ROUND function is applied to round the result to one decimal place. In cases where tcg_not_stolen_gun is zero, the result is set to NULL to avoid division by zero errors.
- **Multicast Transformation:** Created two nodes. The left node contains a derived column `IsRatioCalculated`.The right node involves a Conditional Split Transformation to split records based on the presence of the ratio (`IsRatioCalculated`).
- **Derived Column Transformation:** Added a Derived Column Transformation Editor, adding a derived column `IsRatioCalculated`. This flag is set to 1 if the ratio exists and 0 if the ratio is null
- **Conditional Split Transformation:** Split records into two paths based on the presence of the ratio (`IsRatioCalculated`).
- **Flat File Destinations:** Created three flat file destinations to write results:
  1. `Assignment_2_full.csv` - All records, irrespective of the ratio's existence. Includes columns: `incident_id`, `tcg_stolen_gun`, `tcg_not_stolen_gun`, `tcg_ratio`, and `IsRatioCalculated`.
  2. `Assignment_2_with_ratio.csv` - Records with a calculated ratio.
  3. `Assignment_2_without_ratio.csv` - Records without a calculated ratio.

     Both `Assignment_2_with_ratio.csv` and `Assignment_2_without_ratio.csv` includes `incident_id`, `tcg_stolen_gun` `tcg_not_stolen_gun`, and `IsRatioCalculated` columns.
- **Execute Process Task:** Connected the Data Flow Task to an Execute Process Task in the control flow. This task opens the result files automatically after package execution. The file paths and executable file (Excel, in this case) were specified in the task configuration, allowing the client to view the files separately and choose based on their preference.

This sequence of steps ensures a comprehensive analysis of states with youth criminal problems, providing flexibility for users to explore data with or without calculated ratios.

# 3 Part 3 - Multidimensional Data Analysis

## 3.1 Creation of the OLAP Cube

### 3.1.1 Connecting with Database and creation of View

In our quest for an effective OLAP Cube, we connected to GroupID_200_DB, establishing a comprehensive database view. Within the date table, strategic incorporation of `month_of_year` and `the_day_week` named calculation took place. The former captures month names, while the latter denotes the numerical day value. These calculated members were intentionally integrated into the date dimension, ensuring seamless and efficient data querying for future analyses.

### 3.1.2 Creation of Dimensions

During this phase, we established dimensions for Date, Geography, Gun, Incident, and partecipant. Each dimension encapsulates flat hierarchies of the attributes originally present in their respective tables. Additionally, structured hierarchies were crafted for Date, Geography, and Gun.

- **Date Dimension:** The `date_month_year` structured hierarchy was implemented, featuring relationships: The Year → Quarter → The Month → Month of Year → Day of Week → The Day Week.
- **Geography Dimension:** The `Geography` structured hierarchy was implemented, featuring relationships: Continent → Country → State → City.
- **Gun Dimension:** The `Gun` structured hierarchy was incorporated, establishing relationships: Is Stolen → Gun Type.

Following this stage, we created the OLAP cube, incorporating essential measures for analysis, namely, Crime_Gravity and Custody_Count, which were automatically identified. We configured the Aggregate Function for Crime_Gravity as `Sum` and for `Custody_Count` as `Count`. Additionally, both measures were formatted with the `Standard` Formating String.

## 3.2 MDX Query

The MDX queries are stored in a file named "Project-3.mdx," located within the "MDX_Assignment" subfolder.

### 3.2.1 Assignment - 0

**Problem: 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.**

We've built an OLAP cube for in-depth analysis of our database. The cube features key hierarchies and four crucial measures, focusing on crime gravity dynamics, to facilitate comprehensive insights and informed decision-making.

- **[prev_year_crime_gravity]:**
  1. **Expression:**

```
([Date].[day_month_year].currentmember.lag(1), [Measures].[Crime Gravity])
```

  2. **Description:** Captures the crime gravity for the previous year, providing a temporal dimension to our analysis.

- **[percent_diff]:**
  1. **Expression:**

```
IIF(
IsEmpty([Date].[day_month_year].currentmember.lag(1))
OR IsEmpty([Measures].[prev_year_crime_gravity])
OR [Measures].[prev_year_crime_gravity] = 0,
0,
([Measures].[Crime Gravity] - [Measures][prev_year_crime_gravity])/
[Measures].[prev_year_crime_gravity])
```

2. **Description:** Calculates the percentage change in crime gravity compared to the previous year, offering insights into trends and variations.

- **[total_crime_gravity]:**

1. **Expression:**

```
[Measures].[Crime Gravity] / sum([Gun].[Gun Id].members, [Measures].[Crime Gravity])
```

2. **Description:** Normalizes the crime gravity by dividing it by the total crime gravity across different gun IDs, providing a standardized view of crime severity.

- **[Average_Gravity_Score]:**

1. **Expression:**

```
avg([Geography].[Geography].[State].members, [Measures].[Crime Gravity])
```

2. **Description:** Calculates the average crime gravity score across different states, enabling a regional perspective on crime severity.

These computed measures streamline queries alongside the existing cube measures, enhancing our OLAP cube for in-depth analysis, aligning with project objectives, and focusing on crime gravity dynamics.

### 3.2.2 Assignment - 1

**Problem: Show the percentage increase or decrease in total crime gravity with respect to the previous year for each state.**

```
SELECT {
    [Measures].[percent_diff],
    [Measures].[Crime Gravity],
    [Measures].[prev_year_crime_gravity]
} ON COLUMNS,
NONEMPTY(([Date].[day_month_year].[The Year], [Geography].[State].[State])) ON ROWS
FROM [Group_ID_200_DB]
```

**Analysis:**

- Utilizes pre-computed measures, [Measures].[percent_diff] and [Measures].[prev_year_crime_gravity], as described in Assignment-0, optimizing the calculation of percentage change in crime gravity.
- Organizes results by state, offering insights into the annual variations in total crime gravity and enabling quick identification of states experiencing notable shifts in crime gravity trends.

The MDX query efficiently extracts valuable insights into the dynamics of total crime gravity at the state level, contributing to a nuanced understanding of regional crime patterns. The approach aligns with our project objectives of leveraging the OLAP cube for comprehensive analysis.

### 3.2.3 Assignment - 2

**Problem: For each gun, show the total crime gravity in percentage with respect to the total crime gravity of all the guns.**

```
SELECT {
    [Measures].[Crime Gravity],
    [Measures].[total_crime_gravity]
} ON COLUMNS,
NONEMPTY([Gun].[Gun Id].[Gun Id]) ON ROWS
FROM [Group_ID_200_DB]
```

**Analysis:**

- Leverages pre-computed measure [Measures].[total_crime_gravity], as described in Assignment-0, for efficient calculation of the percentage contribution of each gun to the overall crime gravity.
- Provides a clear view of the proportional impact of individual guns on the total crime gravity and allows for the identification of guns with a significant influence on the overall crime gravity.

The MDX query efficiently extracts insights into the distribution of crime gravity across different guns, providing a valuable perspective on the contribution of each gun to the overall crime landscape. This aligns with our project's objective of utilizing the OLAP cube for in-depth analysis and understanding of crime patterns.

### 3.2.4 Assignment - 3

**Problem: Show the incidents having a total gravity score greater or equal to the average gravity score in each state.**

```
WITH SET incidents AS
FILTER([Incident].[Incident Id].[Incident Id],
    [Measures].[Crime Gravity] >= [Measures].[Average_Gravity_Score])
SELECT {
    [Measures].[Crime Gravity],
    [Measures].[Average_Gravity_Score]
} ON COLUMNS,
NONEMPTY(([Geography].[Geography].[State], incidents)) ON ROWS
FROM [Group_ID_200_DB]
```

**Analysis:**

- Utilizes the pre-computed measure [Measures].[Average_Gravity_Score] established in Assignment-0 for efficient comparison.
- The MDX query extracts incidents where the total gravity score is greater than or equal to the average gravity score in each state.
- Provides a focused view of incidents that stand out in terms of gravity, helping identify states with above-average severity in reported incidents.

The MDX query efficiently filters incidents based on their gravity scores, highlighting areas where the severity of incidents surpasses the average within each state. This approach enhances our ability to pinpoint regions with notable crime severity, aiding law enforcement and policymakers in targeted interventions.

## 3.3 Dashboard - Power BI Solution

We first established a connection to the server and choosing Analysis Services option as data source. After selecting our Group_ID_200_DB, Power BI showed us the Cube's structure on the right side. This allowed us to draw the plots of the following sections.

### 3.3.1 Assignment - 4

In response to Assignment 4, wherein we were tasked with creating a dashboard illustrating the geographical distribution of total crime gravity within each age group, we executed the following steps. Initially, we established a connection to our database to extract real-time data, leveraging the Analysis Services tool. Subsequently, within the visualization pane, we incorporated an arcGIS map.

In the map configuration, we specified three key elements: state, crime gravity, and age group. These elements were assigned to the fields of location, size, and tooltip, respectively. This approach resulted in a visually striking representation of crime gravity across states, presented in circular spot form. Hovering over these spots provided detailed information, including the state name, crime gravity corresponding to each age group, enhancing the interactive nature of the visualization.

To offer a more nuanced perspective on crime gravity concerning age groups, we introduced a donut chart. This chart visually depicted the distribution of crime gravity within each state when clicking on any specific state on the map. Additionally, to convey an overall perspective, we incorporated a card displaying the aggregate crime gravity.

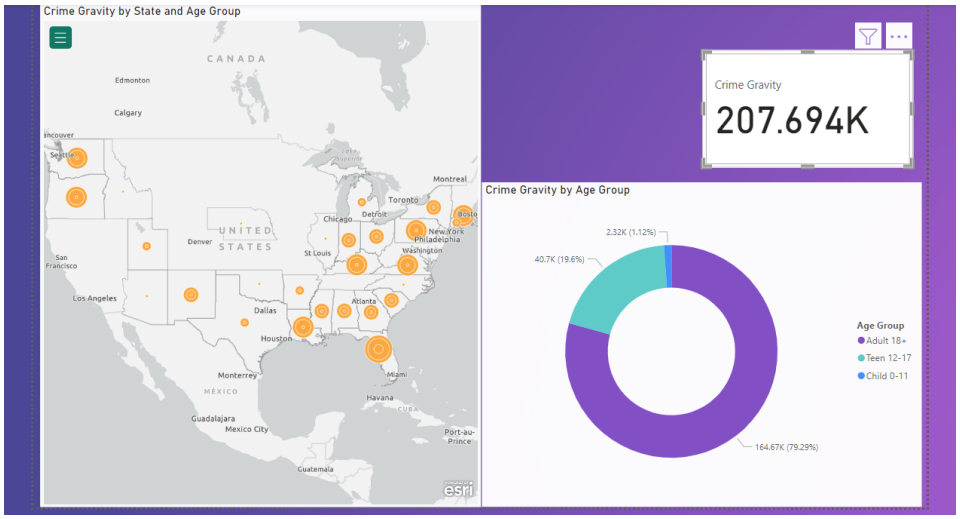



Figure 4: Geospatial Insight: Crime Gravity Across Age Groups

Through these thoughtful additions and configurations, we aimed to address all the requirements outlined in the assignment, providing a comprehensive and visually appealing representation of the geographical distribution of crime gravity across different age groups.

### 3.3.2 Assignment - 5

For the fifth assignment, we were presented with the opportunity to craft a dashboard of our choosing, focusing on elements that intrigued us. Opting for a selection of attributes—gun type, gun status (stolen or not), age group, and year—we initiated the process by introducing two slicers centered around age group and year. This allowed users the flexibility to designate their preferred age group and year for data exploration.

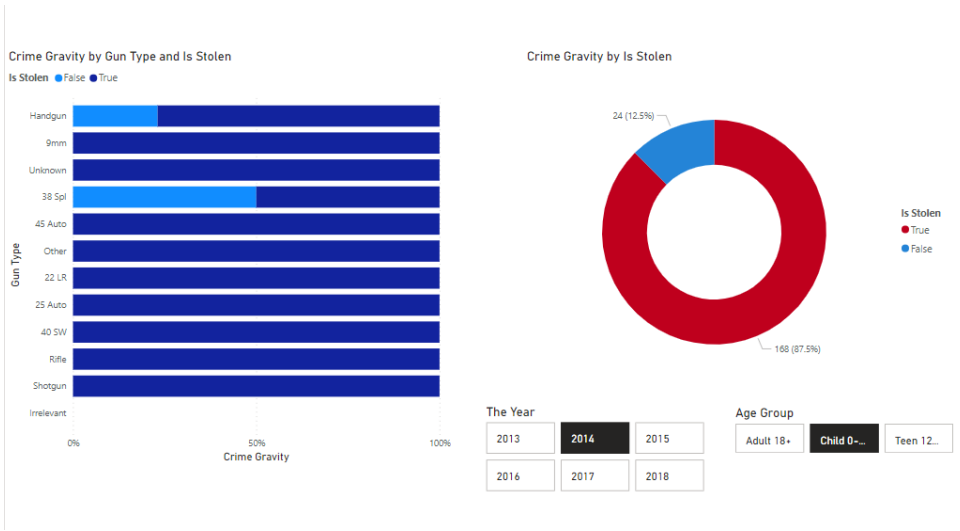



Figure 5: Dynamic Crime Analysis Dashboard: Exploring Gun Type, Status, Age Groups, and Years

Subsequently, we incorporated a stacked bar chart, extracted from the visualization pane, where crime gravity was positioned on the x-axis, gun type on the y-axis, and gun status (stolen or not) serving as the legend. This configuration facilitated the generation of a comprehensive stacked bar chart illustrating crime gravity for both stolen and non-stolen guns. The integration of slicers enabled dynamic adjustments to age group and year.

To augment our visual representation, we introduced a donut chart that showcased the distribution of gun status (stolen or not) along with their corresponding crime gravity. This addition allowed us to discern whether stolen or non-stolen guns contributed more to crime gravity. The slicers, once again, enabled users to refine the data analysis by modifying the selected year and age group.

Furthermore, the dashboard was designed to offer an interactive experience. Clicking on a specific bar within the stacked bar chart prompted the donut chart to dynamically adjust, displaying data exclusively for that particular gun type. This feature provided a focused insight into the relationship between gun status, crime gravity, and other selected attributes.