UNIVERSITÀ DI PISA

Department of Informatics

LABORATORY OF DATA SCIENCE

GUN VIOLENCE DATA WAREHOUSE

Group_ID_3: Xinyi Gu, Yian Li

Academic Year: 2023/2024

Part 1 Data Warehouse Building

1 Project Organization Snapshot

Outlined here is the project's organizational structure, detailing the arrangement of folders and the specifics of the file contents:

- /dataset/: this directory contains the raw data CSV file provided by the professor, some additional datasets for integrate some detailed information and tables derived from raw data exported into CSV prepared for loading onto the SQL Server Management Studio.
  - Raw data: *Police.csv*
  - Split tables: *custody.csv, participant.csv, dates.csv geography.csv, gun.csv*
  - External data: *city_country_continent.csv, uscities.csv*
- /code/: this section of the project structure is dedicated to the Python scripts that facilitate the construction and population of the data warehouse:
  - *data_prep.py*: utilized for data understanding and preparation tasks.
  - *load.py*: executed to load the processed data into the server.
  - *myfunctions.py*: self-defined some reusable functions.

2 Data Understanding and Preparation

In the initial phase, basic data understanding and preparation tasks were conducted on the dataset *Police.csv* to identify the data type of each column and possible missing or redundant data. The key observations performed are as follows:

- 11 columns and 170,928 entries.
- No missing values across all columns.
- Unique Values and data type in Columns:
  - *custody_id*: int, 170,928 unique values.
  - *participant_age_group*: non-numeric, 3 unique values.
  - *participant_gender*: non-numeric, 2 unique values.
  - *participant_status*: non-numeric, 4 unique values.
  - *participant_type*: non-numeric, 2 unique values.
  - *latitude and longitude*: float, 65,553 and 68,607 unique values respectively
  - *gun_stolen*: non-numeric, 4 unique values.
  - *gun_type*: non-numeric, 28 unique values.
  - *incident_id*: int, 105,168 unique values.
  - *date_id*: int, 1,634 unique values.

3 Table Description

Based on our understanding of the original dataset *Police.csv* and the Data Warehouse schema given as reference, it was split into 5 tables, including 1 fact table and 4 dimension tables.

participant
(<u>participant_id</u>, participant_age_group, participant_gender, participant_status, participant_type)

We first generated the *participant_id* and added it into the *Police* table. Then we splitted away the *participant* table selecting all attributes needed and eliminating duplicates.
The ids were generated based on the attributes *participant_age_group, participant _gender, participant _status* and *participant_ type* (e.g.: 1121 represent participant in *age_group* Adult 18+ with *gender* Male, *status* Arrested and *type* Suspect). The *participant _gender* was assigned with value 1 for Female and 1 for Male. The remaining categorical attributes were mapped using the JSON files provided by the professor with the only change of the value {*injured: 2}* with 3 (to ensure the uniqueness of the IDs).

gun
(<u>gun_id,</u> gun_stolen, gun_type)

The *gun* table was created in a similar way as the *participant* table. The *gun_id* is the combination of values corresponding to *gun_type* and *gun_stolen* (e.g.: 181 represent gun with *gun_type* Shotgun and *gun_stolen* Stolen). The *gun_stolen* was represented with values 0 for Not-stolen and 1 for Stolen, and the gun_type was mapped with unique numbers generated in a sequential way from 1 to n.

geography
(<u>geo_id,</u> city, state, latitude, longitude, country, continent)

Similarly, we generated a col *geo_id* adding it to the *Police* table, while creating a separate *geography* table and removing duplicates. Ids were generated from 0 to n according to the unique combination of lat&lng. We additionally integrated the table with some location information using external dataset[1].

dates
(<u>date_id,</u> date, day, month, year, quarter, day_of_the_week)

The dates table was converted from an XML file. We eliminate the time contained in the date column as it does not provide any additional information. For integration, we completed the table with some detailed date information derived from the *date* column.

custody
(<u>custody_id,</u> incident_id <<DD>>, participant_id*, gun_id*, date_id*, geo_id*, crime_gravity)

This is the fact table for our data warehouse, with one measure *crime_graivty*[2]. The primary key *custody_id* is provided by the original table *Police*. And we decided to store the dimension table *incident* into the fact table as degenerate dimension *incident_id* as it does not have its own attributes. In this case, we can reduce the complexity the data warehouse schema, reduces the database's space usage, and simplify the querying process and ETL processes when dealing with degenerate dimension.

[1] https://simplemaps.com/data/us-cities, https://gitcode.com/mirrors/wizardcode/world-area/tree/master/children
[2] crime gravity(x) = F1(x.partecipant age) * F2(x.partecipant type) * F3(x.partecipant status)

4 Database Design and Load

We first created the necessary table structures, designed the data structure in *SQL Server Management Studio* and uploaded the prepared data. The *load.py* includes SQL statements for creating tables, loading, and integrating data from various source files.

1-1

Fig. 1: Data Warehouse Schema

Part 2 SSIS

5 SSIS Solution

We used *Visual Studio* to establish an Integration Services Project for executing the ETL process within our data warehouse.

Assignment 0

For every year, the participant ordered by the total number of custodies.

Data Extraction: used **OLE DB Source**

- **Table**: *custody*
- **Columns**: *custody_id, participant_id, date_id*

Data Transformation:

1. **Data Joining**: applied a **Lookup** transformation to join the "custody" table with the column year in the "dates" table based on the foreign key date_id.
2. **Data Aggregation**: used an **Aggregate** transformation to group data by *year* and *participant_id* and counted *custody_id* stored as *custody_count*.

Output of Results:

- Exported the results to a csv file named *assignment_0.csv* using a **Flat File Destination**.

Assignment 1

For each state, compute the *stolen gravity index* defined as the ratio between the total gravity of custodies involving stolen guna divided by the overall gravity of custodies.

Data Extraction: used **OLE DB Source**

- **Table**: *custody*
- **Columns**: *custody_id*, *geo_id*, *crime_gravity*

Data Transformation:

1 **Data Joining and Preprocessing**: used **lookup** transformation, to join the data with column *state* in the *geography* table based on the foreign key *geo_id.*

- Applied **MultiCast** transformation for reusing the joined tables.
  - One branch used a **Conditional Split** to separate the data according to whether the gun is stolen based on the last digit of the *gun_id*.

2. **Data Aggregation**: Both branches are connected to the **Aggregate** transformation which grouped the data by *state*and computed the two required totals (total gravity of custodies involving stolen guns named *tot_stolen_gravity_state* and the overall gravity of custodies named *tot_crime_gravity*).
3. **Data Merging**: - After **sorting** by *state*, merged the two data streams using the **MergeJoin** transformation.

4. **Calculation**: using the **Derived Column** tool to comput the stolen gravity index by dividing the *tot_stolen_gravity_state* by the *tot_crime_gravity* named *stolen_gravity_index* and to convert the result to floating-point.

Output of Results:

- Exported the results to a csv file named *assignment_1.csv* selecting the columns *state* and *stolen_gravity_index* using a **Flat File Destination**.

Assignment 2

For each month, compute the total gravity in percentage with respect to the annual total.

Data Extraction: used **OLE DB Source**

- **Table**: *custody*
- **Columns**: *date_id, crime_gravity*

Data Transformation:

1. **Data Joining and Preprocessing**: used **Lookup** transformation to join the data with column *year* and *month* in the *date* table based on the foreign key *date_id*.
   - Used **MultiCast** transformation for reusing the joint table.
     - Two branches were used, each applying an **Aggregate** transformation: one grouped by *year* and *month* and calculated total gravity stored as *tot_gravity_month*, and the other grouped only by *year* and calculated total gravity stored as *tot_gravity_year*.
2. **Data Merging**: after aggregation, **sorted** the two data by *year*, then merged them using **MergeJoin** transformation.
3. **Calculation:** connected the merged data to **Derived Column** block. Used the expression '*(DT_DECIMAL,4)((DT_R4)100 * tot_gravity_month / tot_gravity_year)*' to calculate the percentage of monthly gravity with respect to annual total gravity named *month_pct_gravity(%)*.

Output of Results:

- Exported the results to a csv file named *assignment_2.csv* selecting *year, month* and *month_pct_gravity(%)*using a **Flat File Destination**.

Fig.2: Data Flow - assigment_0, 1, 2 (from left to right)

Part 3 Multidimensional Data Analysis

6 Data Cube

To build a OLAP cube, we used *Visual Studio* and created an Analysis Service Multidimensional and Data Mining Project.

6.1 Data Source and View

We first established a data source connection to our data, Group_ID_3_DB. Afterwards, we created a view of our data warehouse. As a part of our preparation for the next step, we decided to add a new calculated column into the *geography* table called *coordinates*, concatenating the *latitude* and *longitude* fields.

6.2 Dimensions

In this step, we defined all the hierarchies needed to build our cube and complete our tasks.

Geography Dimension: created based on the *geography* table with the key column *geo_id* setting stored values of *coordinates* as the minimum granularity.

- flat hierarchies: continent, country, state and city
- geo_Hierarchy : continent → country → state → city → geo_id

Date Dimension: created based on the *dates* table with the key column *date_id* representing values of *date* column as the minimum granularity.

- flat hierarchies: year, quarter, month and day
- time_Hierarchy : year →quarter → month → day → date_id

Participant Dimension: created based on the *participant* table containing all the flat hierarchies derived from the attributes present in the table.

6.3 Cube

In the final step, we created the OLAP cube including the measures needed for MDX queries and visualization tasks:

- *Crime Gravity* calculating the sum of *crime_gravity*
- *Custody Count* as the count of *custody_id*.
- *Incident Count* as the count of distinct *incident_id*.

7 MDX query

Assignment 1

Show the total crime gravity for each city and the grand total with respect to the state.

As required, two columns are needed:

- **Crime Gravity**: total crime gravity of each combination of *state* and *city*

- **grand_tot_state**: calculated member as the total *Crime Gravity* for each state obtained by retrieving the parent of the current member in the *geo_Hierarchy* level of the *geography* dimension.

For the rows, a cross-join operation is used on the states and cities to avoid possible duplicate city names.

		Crime Gravity	grand_tot_state
Alabama	Abbeville	4	4261
Alabama	Adamsville	1	4261
Alabama	Alabaster	4	4261
Alabama	Albertville	18	4261
Alabama	Alexander City	32	4261
Alabama	Andalusia	0	4261
Alabama	Anniston	13	4261
Alabama	Arab	12	4261
Alabama	Ardmore	1	4261

Fig. 3: Output of assignment 1

Assignment 2

Show the percentage increase or decrease in total crime gravity answers with respect to the previous year for each age group.

For this task, we need to retrieve the crime gravity for each year of each age group alongside the crime gravity of the previous year and to calculate the YOY percentage change. To achieve this, 3 columns are shown:

- **Crime Gravity**: total crime gravity for each year.
- **prev**: previous value of *Crime Gravity* for the specified time hierarchy.
- **pct**: the ratio is calculated as *([Crime Gravity] - prev)/ prev*. For special cases, if current year is the first year, we set the ratio as 0%, if both the values of previous year and current year is 0, we set the ratio as 0% and if the value of *prev* is 0, the ratio is set to 100%.

For the rows, we selected the set of non-empty combinations of years and participant age groups.

And for a better view, the Percentage format is used, and all the null values are shown as '-'.

		Crime Gravity	prev	pct
2013	Adult 18+	225	-	0.00%
2013	Child 0-11	0	-	0.00%
2013	Teen 12-17	30	-	0.00%
2014	Adult 18+	7491	225	3229.33%
2014	Child 0-11	192	0	100.00%
2014	Teen 12-17	2412	30	7940.00%
2015	Adult 18+	30027	7491	300.84%
2015	Child 0-11	726	192	278.13%
2015	Teen 12-17	7929	2412	228.73%
2016	Adult 18+	44478	30027	48.13%
2016	Child 0-11	528	726	-27.27%
2016	Teen 12-17	10107	7929	27.47%
2017	Adult 18+	67820	44478	52.48%
2017	Child 0-11	690	528	30.68%
2017	Teen 12-17	16362	10107	61.89%
2018	Adult 18+	14635	67820	-78.42%
2018	Child 0-11	186	690	-73.04%
2018	Teen 12-17	3858	16362	-76.42%

Fig. 4: Output of assignment 2

Assignment 3

Show the ratio between the total crime gravity of each year w.r.t the previous year.

This MDX query aims to provide insights into the year-wise evolution of *Crime Gravity*, with columns:

- **Crime Gravity**: total crime gravity for each year.
- **prev**: previous value of *Crime Gravity* for the specified time hierarchy.
- **ratio**: calculated as *[Crime Gravity]/prev*, special cases are handled similarly to the previous query.

	Crime Gravity	prev	ratio
2013	255	-	0.00%
2014	10095	255	3958.82%
2015	38682	10095	383.18%
2016	55113	38682	142.48%
2017	84872	55113	154.00%
2018	18679	84872	22.01%

Fig. 5: Output of assignment 3

On the row axis, it displays the data for each year in the *time_Hierarchy*.

8 Data Visualization

In this section, we used the *Power BI* to draw some plots choosing Analysis Services to connect the OLAP cube created in section 7.

Assignment 4

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

Two charts were created to present data from different perspectives using a Visualization tool Map. First, we focused on crime distribution for each state and use age group as slicer. In Fig. 6, we chose the state level to show the geographical distribution of the crime gravity specifically focusing on 'Adult 18+' age group. Notably, severe gun violence is evident in the East America region. The states with highest levels of crime gravity include California, Florida, and Texas.

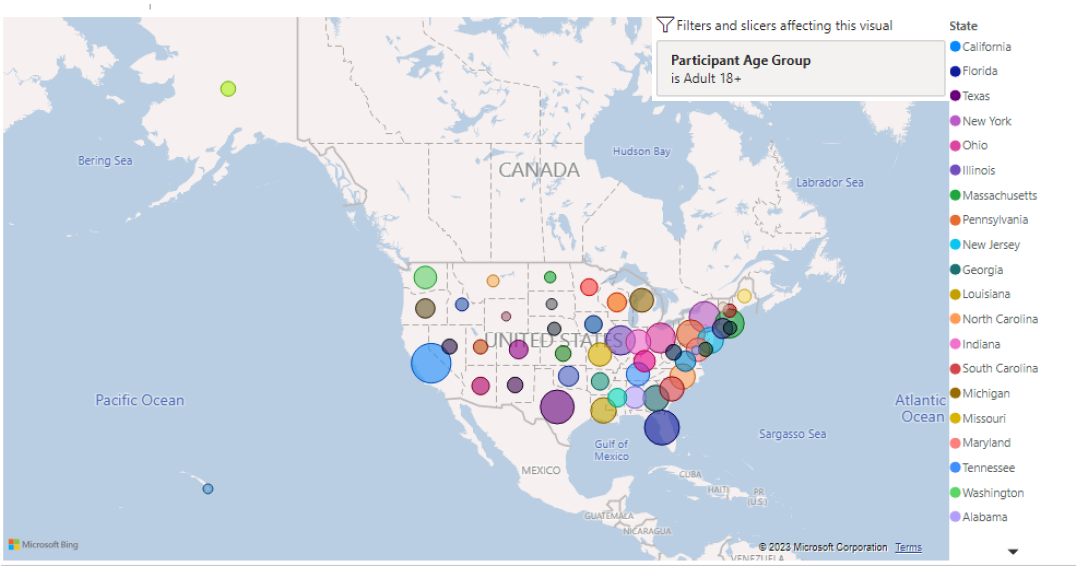

Fig. 6: Crime Gravity Distribution among Adults 18+ for Each State

While in Fig. 7, we focused on crime age trends by state to show the distribution of age group in each state. In most of the states, the crime gravity among children and teenagers accounts for less than a quarter of total crime gravity while it is a significant proportion. Areas that tend to have relatively lower crime gravity also have a smaller proportion of juvenile delinquency.

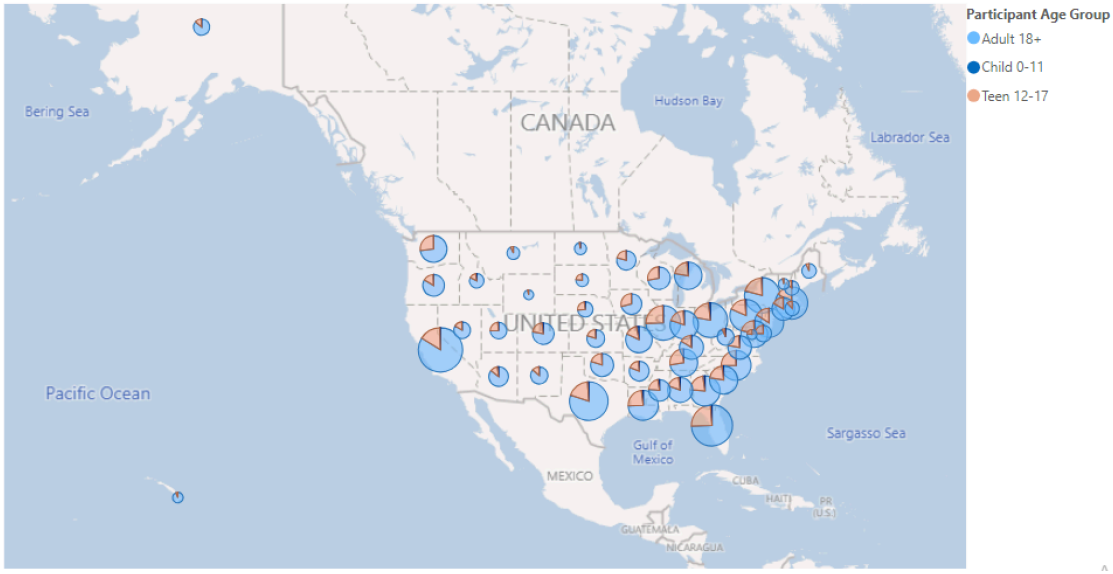

Fig. 7: Crime Gravity by State and Age Group

Assignment 5

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

We then created a dashboard that displays the number of incidents each month across different years. The cart identifies seasonal and monthly crime trends, and this could help public safety organizations to prepare and allocate resources effectively. There was an upward trend in gun crime after 2013 while after the second half of 2017, the trend has been downward. The months with the highest number of gun violence incidents were August and December.

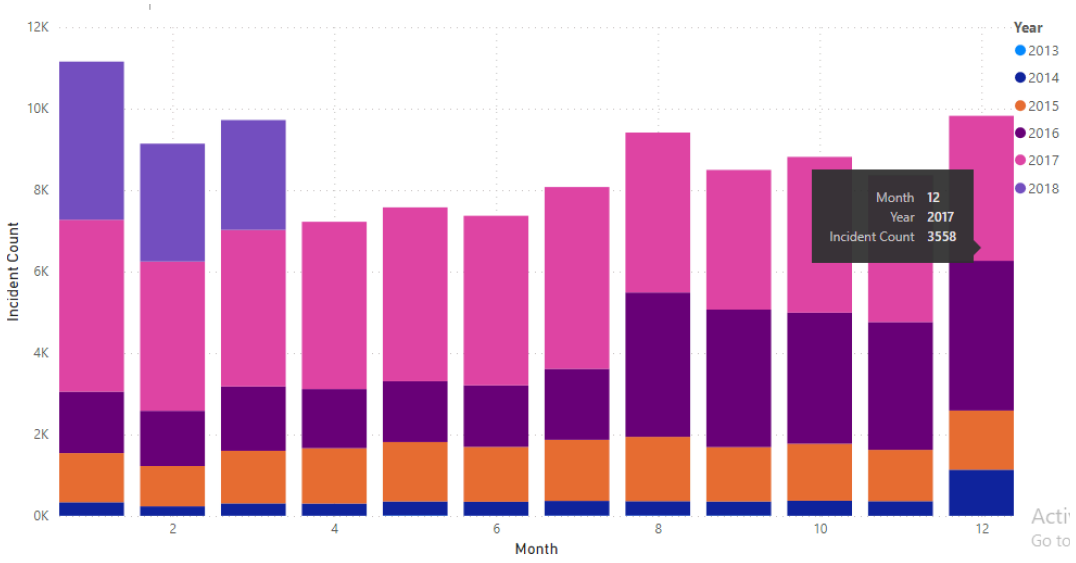

Fig. 8: Incidents Distribution by Month and Year