



## Laboratory of Data Science Project Report AY 2023/2024

### Studenti:

Graziano Amodio

Marco Vasta

# Contents

<b>1</b>	<b>Costruzione del Data-warehouse</b>	<b>1</b>
1.1	Introduzione . . . . .	1
1.2	Assignment 0 - Database schema creation . . . . .	1
1.3	Assignment 1 - Creazione delle tabelle . . . . .	1
1.4	Assignment 2 . . . . .	3
<b>2</b>	<b>SISS Solution</b>	<b>4</b>
2.1	Assignment 0 . . . . .	4
2.2	Assignment 1 . . . . .	5
2.3	Assignment 2 . . . . .	6
<b>3</b>	<b>Multidimensional Data Analysis</b>	<b>8</b>
3.1	Assignment 0 . . . . .	8
3.2	Assignment 1 . . . . .	9
3.3	Assignment 2 . . . . .	10
3.4	Assignment 3 . . . . .	11
3.5	Assignment 4 . . . . .	12
3.6	Assignment 5 . . . . .	12

# 1. Costruzione del Data-warehouse

## 1.1 Introduzione

La prima parte del progetto consiste nel creare e popolare un database a partire da vari file che ci sono stati consegnati. In particolar modo il database conterrà dati legati a crimini commessi negli Usa nell'arco temporale di alcuni anni. Il database conterrà informazioni di vario genere: temporali, spaziali ed altre riferite ai soggetti interessati ed alle modalità con la quale sono stati commessi i reati in questione.

## 1.2 Assignment 0 - Database schema creation

La prima task che ci viene richiesta riguarda la creazione dello schema del data-warehouse. Per riprodurlo e per caricarlo sul server dell'università utilizziamo **SQL Management Studio**. Quindi creiamo ciascuna *dimension table* e la *fact table* assegnando per ciascuna chiavi primarie ed esterne corrispondenti. Infine assegnamo le relazioni corrette tra le tabelle.

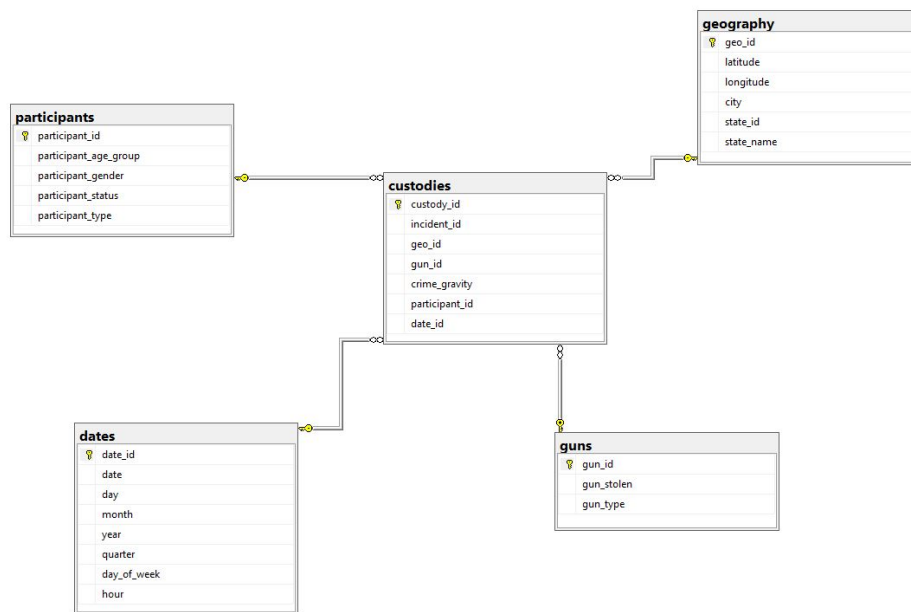


Figure 1: Database schema

## 1.3 Assignment 1 - Creazione delle tabelle

A partire da due file che ci sono stati consegnati ci viene richiesto di creare sei tabelle diverse. I file in questione sono: **Police.csv** e **Date.xml**. Per prima cosa abbiamo dovuto unire i dati contenuti in essi e per fare ciò abbiamo scritto un codice che ci ha permesso di analizzare il file *date.xml* andando a raccogliere in un dizionario le informazioni contenute in esso (la funzione definita è stata chiamata *parse\_xml*). Le chiavi del dizionario che abbiamo popolato sono *date* e *date\_pk* e tramite di ciò è stato possibile eseguire un *merge* su *date.date\_pk = police.date\_fk* associando i dati contenuti

nei due diversi file. Per leggere e scrivere sui file invece abbiamo definito una funzione *open* ed aggiunto le modifiche necessarie per la creazione di un nuovo file unito che abbiamo denominato **police\_date.csv**.

Successivamente abbiamo proseguito alla creazione delle sei differenti tabelle e per fare ciò abbiamo usato una funzione che abbiamo chiamato *estrai\_crea\_tabella* che ci permette di estrarre solo le colonne degli attributi che ci interessano dal file *police\_date.csv* precedentemente creato aggiungendo una colonna *"id"* (chiave artificiale) utile alla identificazione univoca di ogni riga da usare come chiave esterna per alcune tabelle.

Nel dettaglio:

- **guns**: Abbiamo estratto da *police\_date.csv* le colonne **gun\_stolen**, **gun\_type** contenenti informazioni sulla modalità di acquisizione dell'arma e del tipo di arma usata per ogni record collezionato. Inoltre abbiamo aggiunto una colonna chiamata **gun\_id** partendo da 0 fino alla lunghezza delle colonne ed abbiamo aggiunto una colonna **Is\_stolen** che assegna 1 al record con valore *"Stolen"* nella colonna *gun\_stolen* e 0 altrimenti.
- **participants**: Esattamente come svolto precedentemente per questa tabella abbiamo aggiunto una colonna chiamata **participant\_id** ed abbiamo estratto dal file principale le colonne **participant\_age\_group**, **participant\_gender**, **participant\_status**, **participant\_type**. Le tabelle in questione contengono informazioni in merito ai soggetti coinvolti.
- **geography**: Per la creazione di questa tabella abbiamo prima di tutto estratto le colonne **latitude**, **longitude** assegnandole ad una colonna creata **geo.id**. Successivamente ci viene richiesto di associare ad ogni record appena estratto altre informazioni legate all'effettiva posizione del crimine commesso. Per fare ciò, considerando i numeri elevati di record, invece che utilizzare il pacchetto *geo.py* abbiamo cercato un file che ci restituisse le coordinate geografiche delle città degli Stati Uniti ed abbiamo calcolato la differenza tra le coordinate del nostro dataset con quelle del dataset comprendente il nome delle città. Abbiamo calcolato quindi ogni possibile distanza sia sulla latitudine che sulla longitudine ed aggiornato i valori alla minima distanza trovata. Ad essa si è poi associato il nome della città più vicina. Le colonne aggiunte nella tabella finale sono **city**, **state\_id**, **state\_name**.
- **dates**: Per la creazione di questa tabella abbiamo estratto la colonna **date** e attraverso l'utilizzo del pacchetto *datetime* abbiamo creato altre colonne quali: **day**, **month**, **year**, **quarter**, **day of the week**, **hour**, in aggiunta delle colonne **date** e **date\_id**. Pur avendo tutti i campi uguali, abbiamo pensato di conservare le informazioni legate all'ora per eventuali ulteriori applicazioni successive da parte del committente.
- **incidents**: La tabella in questione, precedentemente, è stata creata associando alla colonna **incident** una colonna *"incident\_id"* come richiesto nello schema della traccia. Successivamente per motivazioni legate alla semantica dei dati, abbiamo deciso di eliminarla dallo schema ed aggiungere solamente la colonna **incident\_id**, già presente nel file principale sulla quale stiamo lavorando. In questo modo abbiamo evitato la creazione di una tabella avente un solo attributo, associando ogni record di *incident\_id* alla chiave della tabella *custodies*.

- **custodies:** La fact table è l'ultima da creare e in questa fase del lavoro ci viene richiesto di aggiungere una colonna chiamata **crime\_gravity**. Essa dipende dalle colonne *participant age group*, *participant status* e *participant type* che attraverso 3 diversi dizionari in formato .json, ci permettono di associare alle diverse modalità delle colonne in questione un valore numerico. Infine il risultato di *crime\_gravity* è dato dalla moltiplicazione dai valori associati di ogni colonna. Per la sua realizzazione abbiamo definito una funzione chiamata *aggiungi\_colonna* che legge i dati contenuti nelle colonne di *participants* e dei dizionari e li associa. Infine viene creata una vera e propria tabella provvisoria che associa ad ogni riga il valore della moltiplicazione richiesta. Una volta fatto ciò estratto e creato le colonne della tabella *custodies* utilizzando i percorsi delle tabelle precedentemente create. Le colonne di *custodies* sono **custody\_id**, **incident\_id**, **gun\_id**, **participant\_id**, **geo\_id**, **date\_id**, **crime\_gravity**.

In conclusione dell'assignment 1 ci viene richiesto di generare dei valori mancanti su una delle colonne di ids e noi per ciò abbiamo generato il 10% di valori mancanti sulla colonna **participant\_id**. Per il replace dei valori mancanti, considerando che la colonna è *participant\_id*, vuol dire che contiene valori numerici ordinati ed unici. Considerando ciò abbiamo sostituito i valori *None* con 0 e impostato un'iterazione che appena incontra un valore corrispondente a 0, lo sostituisce con il valore precedente non nullo, aggiungendo 1. Questo risolve due problemi: il primo problema riguarda il primo valore presente che dev'essere 0 e che resta tale in quanto non ha nessun valore precedente; il secondo problema riguarda la possibilità di avere più valori mancanti in sequenza che viene risolto considerando che l'iterazione è ordinata e che quindi di volta in volta il replace avviene incontrando sempre un valore 0 preceduto da un valore che non è nullo.

## 1.4 Assignment 2

Infine abbiamo popolato il database, attraverso un codice scritto con python, caricando le tabelle precedentemente create specificando nel codice le proprie caratteristiche.

## 2. SISS Solution

In questa parte del progetto ci viene richiesta la risoluzione di alcuni quesiti attraverso l'utilizzo di SQL server integration services lavorando sul database appena creato.

### 2.1 Assignment 0

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

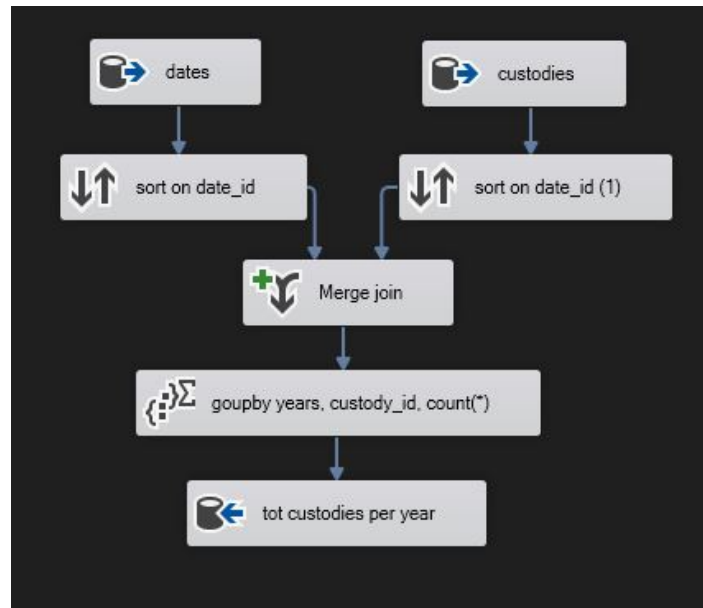


Figure 2: SSIS solution Assignment 0

Per prima cosa selezioniamo le tabelle di nostro interesse tramite *Origin OLE DB* e sia per la tabella **dates** che per la tabella *custodies* eseguiamo un ordinamento sulla chiave *date\_id* per eseguire un *inner join*. Successivamente, per completare la richiesta raggruppiamo per *years* e contiamo il numero di records. Infine carichiamo il risultato che riportiamo in un estratto di seguito:

SQLQuery3.sql - Ids...(Group\_ID\_888 (63))\*

```
SELECT *
FROM [Group_ID_888].[tot_custodies_per_year]
```

100 %

Risultati

	year	total_custodies
1	2014	8864
2	2016	45256
3	2018	14803
4	2013	470
5	2015	27515
6	2017	74020

Messaggi

Figure 3: Risultato

## 2.2 Assignment 1

*"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."*

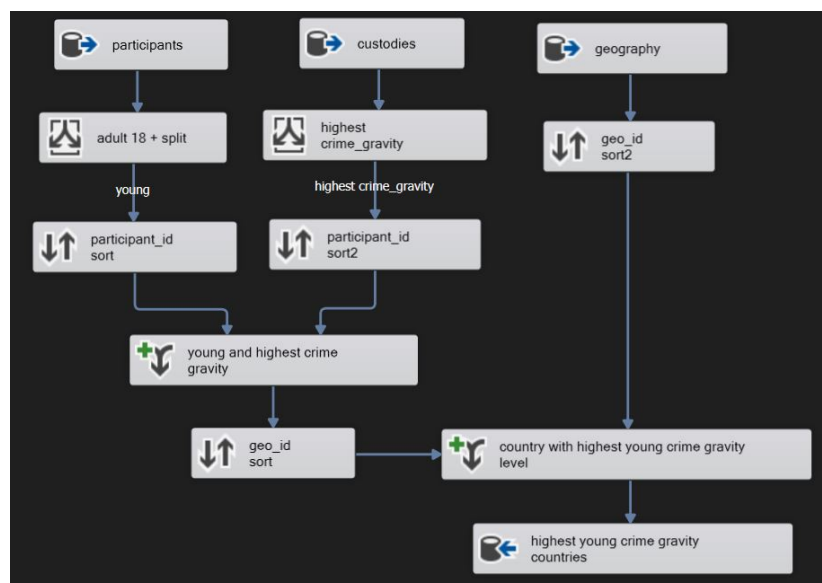


Figure 4: SSIS solution Assignment 1

Per la risoluzione di questo quesito abbiamo eseguito una restrizione per la tabella **participants** selezionando solamente i record che riguardassero i minori di 18 anni. In modo analogo abbiamo eseguito una restrizione per la tabella **custodies** selezionando solamente i records aventi il valore di *crime\_gravity* massimo che risulta essere uguale a 24. Dopo di ciò abbiamo eseguito un *inner join* per unire le tabelle. Per ottenere anche la parte di informazione geografica di interesse, abbiamo eseguito un *merge* con la

tabella **geography**. Prima dei merge abbiamo eseguito degli ordinamenti sulle chiavi per eseguirli e abbiamo proiettato solamente le colonne di nostro interesse. In seguito il risultato ottenuto ci mostra che sono 4 i paesi degli Stati Uniti ad avere un *youth criminal problem*.

	state_id	state_name
1	CO	Colorado
2	LA	Louisiana
3	MO	Missouri
4	MS	Mississippi

Figure 5: risultato

## 2.3 Assignment 2

*"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."*

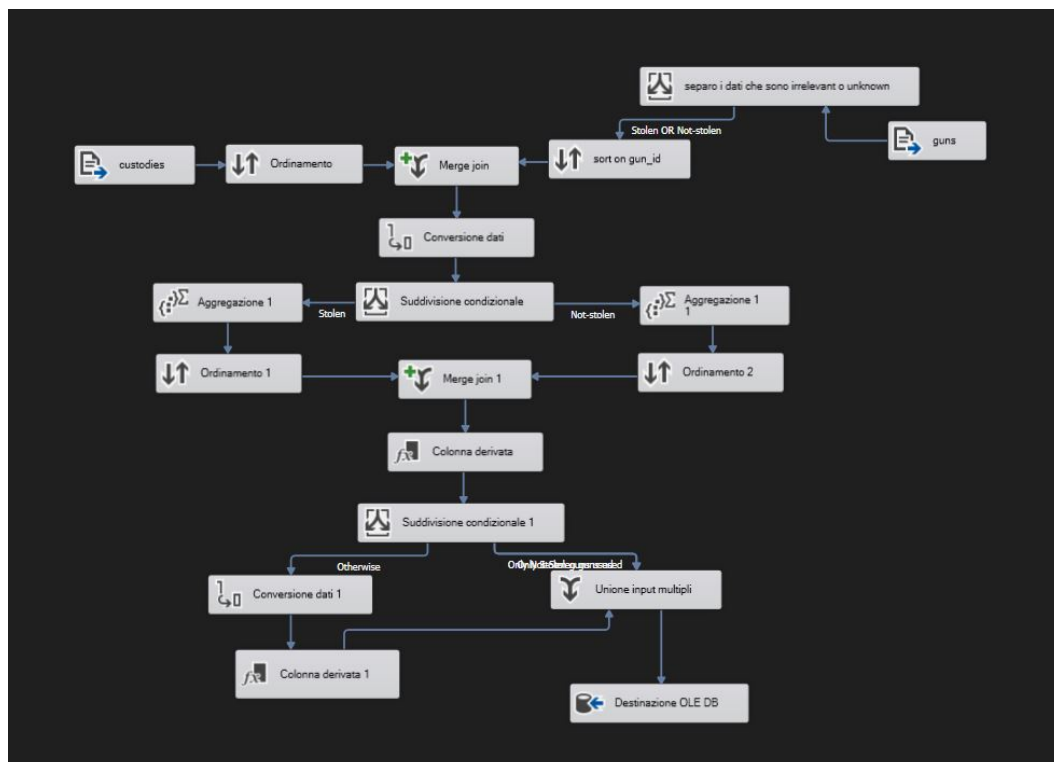


Figure 6: SSIS solution Assignment 2



Per la risoluzione di questo quesito è stato necessario restringere la tabella **guns** riducendola ai soli crimini commessi con armi rubate o con armi non rubate. Il resto del dataset scartato comprende crimini commessi con armi definite irrilevanti o dalla provenienza sconosciuta. Essendoci richiesto il calcolo di un indice, abbiamo ritenuto giusto quindi restringere il campo ai soli certamente riguardati armi rubate o armi non rubate. Dopo di ciò abbiamo eseguito un merge ottenendo un *inner join* con la tabella **custodies**. Dopo aver convertito la colonna *crime gravity* in formato numerico, abbiamo eseguito una suddivisione per separare i record con modalità *Stolen* e *Not-stolen*. Infatti a noi interesserà associare ad ogni *incident\_id* il rapporto tra i due. Per entrambi gli split prodotti dalla restrizione, utilizziamo una aggregazione su *incident\_id*, poi otteniamo una colonna chiamata *count of Stolen/Not-stolen guns* ottenuta con il count e infine aggiungiamo un'altra colonna ottenuta dalla somma di *crime gravity*. In questo modo **per ogni incident\_id** ottengo il conteggio delle armi usate (quindi delle righe) e soprattutto la somma dei *crime gravity*, **distinti** per *Stolen* e *Not-Stolen*. A questo punto del problema è necessario porre l'attenzione sul fatto che in caso in cui, considerando un *incident\_id*, esso comprendesse solamente una tipologia di acquisizione dell'arma (quindi *Stolen* o *Not-stolen*), il rapporto restituito sarebbe impossibile da visualizzare. Infatti SSIS ci restituirebbe un errore per le divisioni con il valore 0. In merito, quindi, abbiamo optato per una soluzione che restituisse anche questa informazione al committente. Il passaggio successivo è stato creare un *full outer join* sull'attributo *incident\_id* allo scopo di ottenere quattro diverse colonne:

- incident id (stolen guns)
- incident id (not stolen guns)
- total crime gravity per stolen guns
- total crime gravity per not stolen guns

La strategia per ottenere il massimo dell'informazione richiesta, consiste adesso di trasformare i numerosi valori nulli ottenuti in formato booleano. In questo modo al risultato aggiungeremo due colonne che avranno come modalità TRUE o FALSE. Le colonne in questione, per ogni *incident id*, ci informeranno se è vero che ad esso corrisponde un solo tipo di armi usate. Utilizziamo le modalità TRUE e FALSE per applicare una restrizione ai record che presentano, per *incident id*, entrambe le modalità (consone per calcolarne il rapporto richiesto) e ne calcolo il rapporto dopo aver convertito i valori in float. Come ultimo passaggio, posso unire il dataset.

Abbiamo deciso di conservare il numero di colonne per avere una maggiore chiarezza cercando di interpretare la richiesta del committente.

Il risultato ottenuto segue estraendo le tre diverse tipologie di rappresentazione.

	incident_id (Stolen)	incident_id (Not-Stolen)	Only Stolen guns	Only Not-stolen guns	Stolen Not-stolen sum of c g
5...	69687		True	False	
5...	69689		True	False	
5...	69698		True	False	
5...	69700		True	False	
5...	6971		True	False	
5...	69711		True	False	
5...	69714		True	False	
5...	69723		True	False	
5...	69725		True	False	
5...	69734		True	False	
5...	69737		True	False	
5...	69739		True	False	
5...	69743		True	False	
5...	69745		True	False	
5...	69746		True	False	
5...	69750		True	False	
5...	69762		True	False	
5...	69764		True	False	
5...	69769		True	False	

	incident_id (Stolen)	incident_id (Not-Stolen)	Only Stolen guns	Only Not-stolen guns	Stolen Not-stolen sum of c g
6...	98467	98467	False	False	2
6...	98487	98487	False	False	2
6...	98699	98699	False	False	1,5
6...	98805	98805	False	False	3
6...	98874	98874	False	False	0,66666669
6...	98894	98894	False	False	2,3333333
6...	99088	99088	False	False	1
6...	99325	99325	False	False	3
6...	99381	99381	False	False	1
6...	99387	99387	False	False	2
6...	99437	99437	False	False	1
6...	99462	99462	False	False	1
6...	99550	99550	False	False	2
6...	99579	99579	False	False	2
6...	99596	99596	False	False	7
6...	99614	99614	False	False	5
6...	9979	9979	False	False	5
6...	99839	99839	False	False	1
6...	99940	99940	False	False	1

Figure 7: risultato

## 3. Multidimensional Data Analysis

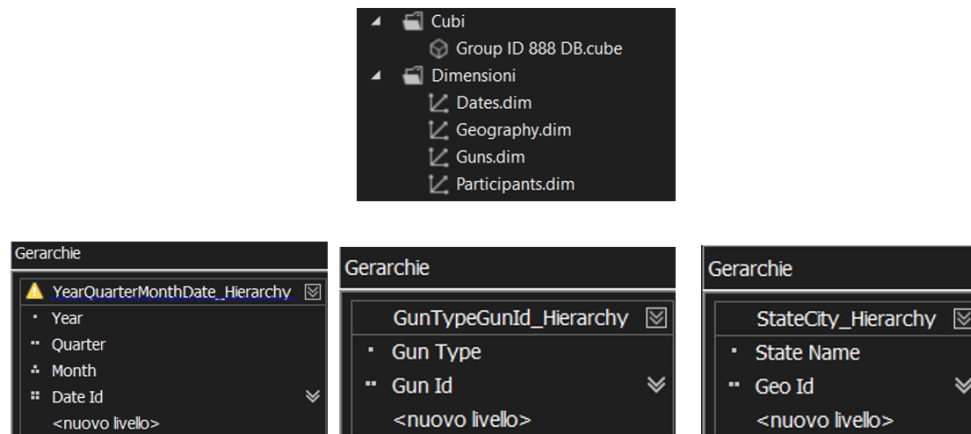
In questa parte spiegheremo velocemente i processi per la crezione del nostro datacube e come abbiamo ottenuto i differenti tipi di visualizzazione attraverso le query in MDX. Nella parte finale analizzeremo in dettaglio attraverso il software Power BI il alcune caratteristiche del dataset.

### 3.1 Assignment 0

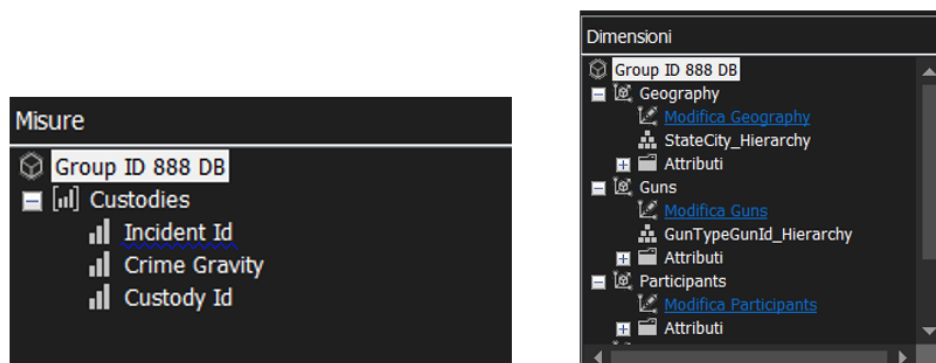
*Build a datacube from the data of the tables in your database, defining the appropriate hierarchies. Create the needed measures based on the queries you need to answer.*

Il processo inizia collegando un nuovo progetto al server SQL principale in modalità Analysis Service per accedere al database precedentemente creato sul server in modalità Database Engine. Seguendo le configurazioni necessarie per essere sicuri di distribuire il lavoro lato client al server, si è acceduto all'origine dei dati, si è creata una vista e si sono aggiunte le dimensioni creando le necessarie gerarchie al loro interno. Infine si è creato il nostro cubo selezionando le misure necessarie. In particolare: le tabelle del database Dates, guns, Participants e Geography sono state scelte per rappresentare le quattro dimensioni del cubo; le chiavi primarie corrispondenti sono nell'ordine date\_id, guns\_id, participants\_id e geo\_id. All'interno di ogni dimensione sono state create delle

gerarchie per consentire un'esplorazione più pertinente del cubo. Le gerarchie sono: "YearQuarterMonthDate\_Hierarchy" in Date, "StateCity\_Hierarchy" in Geography e "GunTypeGunId\_Hierarchy" in Guns.



Si noti che all'interno delle gerarchie delle dimensioni Dates e Geography, i livelli più dettagliati della gerarchia (cioè date\_id e geo\_id) non corrispondono realmente ai valori id della chiave, ma ai valori associati più dettagliati (cioè "date" e "city"). Per quanto riguarda la creazione di misure, invece, sono stati selezionati gli attributi della tabella Custodies: 'Crime Gravity', 'Custody\_id' and 'Incident\_id'. Le funzioni di aggregazione sono in ordine: sum, count, and distinct count.



A questo punto abbiamo terminato le configurazioni del cubo.

### 3.2 Assignment 1

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

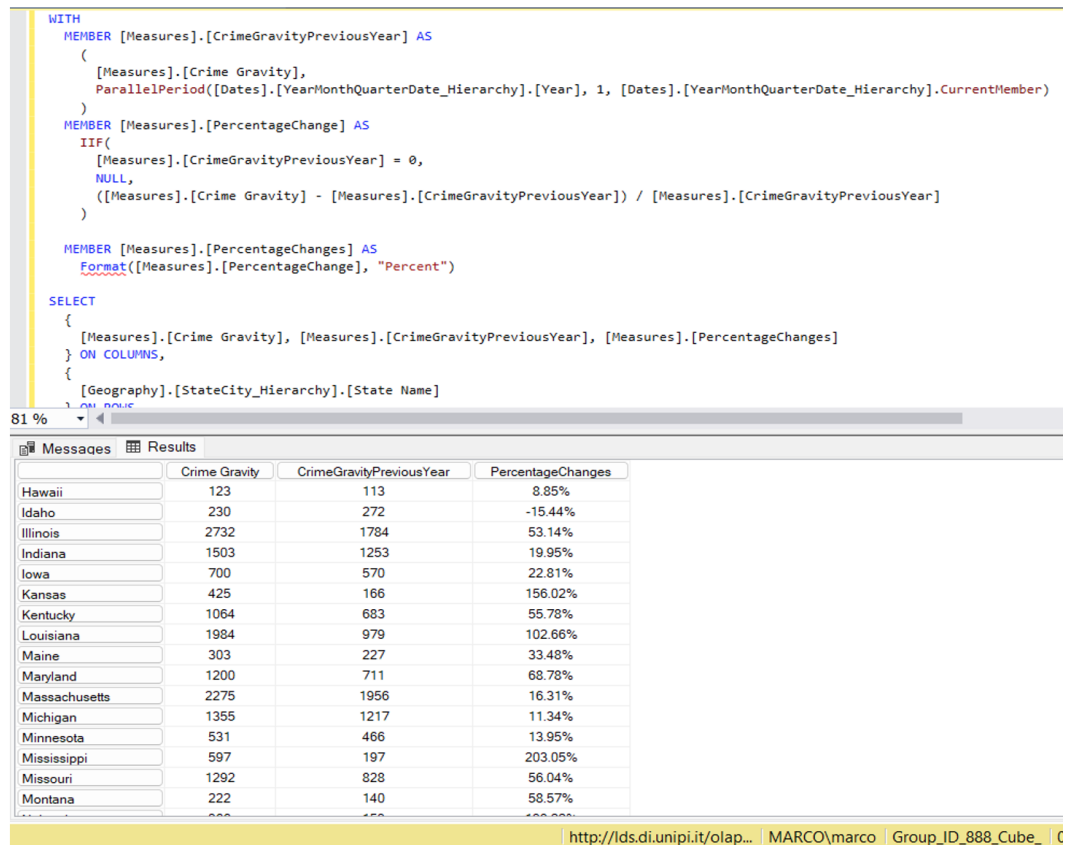
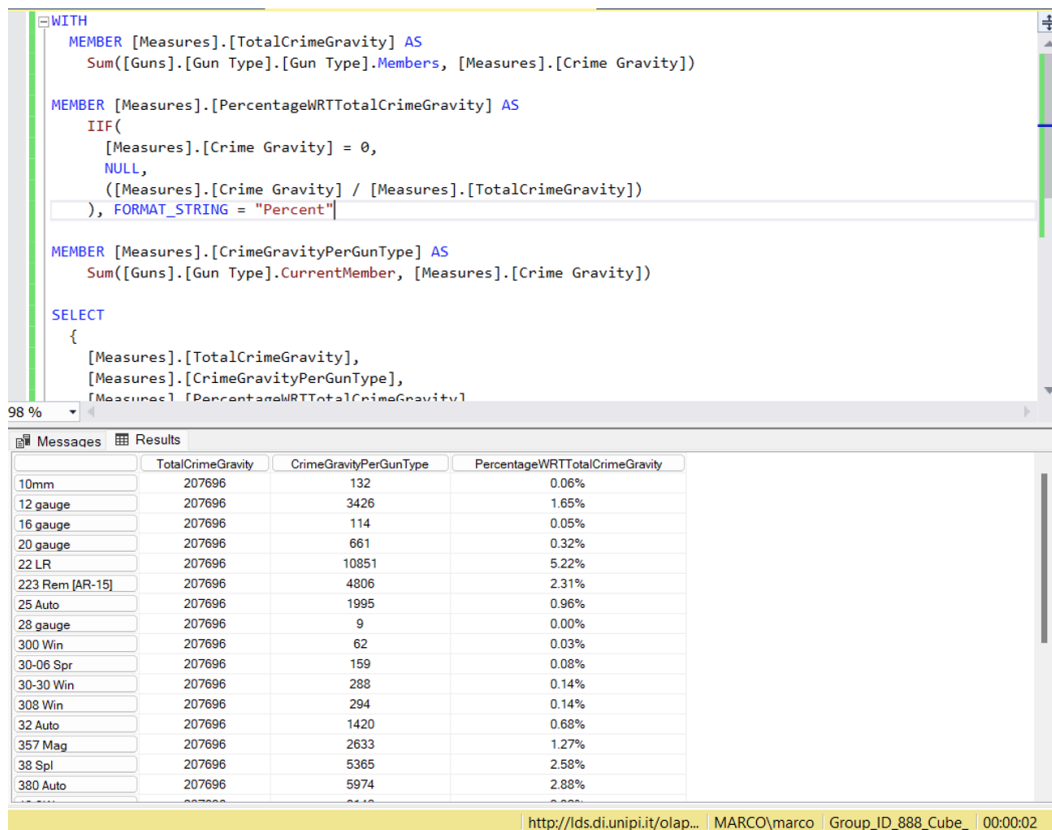


Figure 8: Assignment 1

### 3.3 Assignment 2

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



```

WITH
    MEMBER [Measures].[TotalCrimeGravity] AS
        Sum([Guns].[Gun Type].[Gun Type].Members, [Measures].[Crime Gravity])

    MEMBER [Measures].[PercentageWRTTotalCrimeGravity] AS
        IIF(
            [Measures].[Crime Gravity] = 0,
            NULL,
            ([Measures].[Crime Gravity] / [Measures].[TotalCrimeGravity])
        ), FORMAT_STRING = "Percent"

    MEMBER [Measures].[CrimeGravityPerGunType] AS
        Sum([Guns].[Gun Type].CurrentMember, [Measures].[Crime Gravity])

SELECT
    {
        [Measures].[TotalCrimeGravity],
        [Measures].[CrimeGravityPerGunType],
        [Measures].[PercentageWRTTotalCrimeGravity]
    }

```

	TotalCrimeGravity	CrimeGravityPerGunType	PercentageWRTTotalCrimeGravity
10mm	207696	132	0.06%
12 gauge	207696	3426	1.65%
16 gauge	207696	114	0.05%
20 gauge	207696	661	0.32%
22 LR	207696	10851	5.22%
223 Rem [AR-15]	207696	4806	2.31%
25 Auto	207696	1995	0.96%
28 gauge	207696	9	0.00%
300 Win	207696	62	0.03%
30-06 Spr	207696	159	0.08%
30-30 Win	207696	288	0.14%
308 Win	207696	294	0.14%
32 Auto	207696	1420	0.68%
357 Mag	207696	2633	1.27%
38 Spl	207696	5365	2.58%
380 Auto	207696	5974	2.88%

Figure 9: Assignment 2

### 3.4 Assignment 3

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

```

WITH
    MEMBER [Measures].[AvgGravityPerState] AS
        AVG(
            [Geography].[State Name].MEMBERS,
            [Measures].[Crime Gravity]
        ),
        FORMAT_STRING = "#,###,00"

SELECT
    {[Measures].[Crime Gravity]} ON COLUMNS,
    NON EMPTY
    [Geography].[State Name].MEMBERS *,
    [Custodies].[Incident Id].MEMBERS
    FILTER(
        [Measures].[Crime Gravity] >= [Measures].[AvgGravityPerState]
    ) ON ROWS
FROM [Group ID 888 DB]

```

Figure 10: Assignment 3

Sfortunatamente, non siamo riusciti ad eseguire la query. Ci siamo infatti resi conto della necessità di avere una dimensione tramite la quale accedere e sfruttare

“Incident id” non come misura ma come membro da posizionare sull’asse delle rows al fine di mostrare esattamente gli incidents la cui gravità è maggiore o uguale rispetto alla media della gravità in ogni stato. Tale problema è dovuto ad una nostra iniziale errata interpretazione della costruzione del cubo ed inoltre alla ripetuta impossibilità di effettuare modifiche per via di malfunzionamenti al server.

### 3.5 Assignment 4

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

Per mostrare in maniera quanto più leggibile possibile la distribuzione geografica del *total crime gravity* abbiamo optato per una rappresentazione per i singoli stati. La grandezza delle bolle ci informa efficacemente sulla distribuzione evidenziando una maggiore concentrazione di crimini “gravi” negli stati del sud. Inoltre è possibile, come richiesto, avere informazioni in merito alla distribuzione per ogni gruppo di età, mostrando una prevalenza netta di crimini compiuti da adulti. La Florida, ad esempio, in percentuale ha una parte consistente di crimini commessi da minorenni.

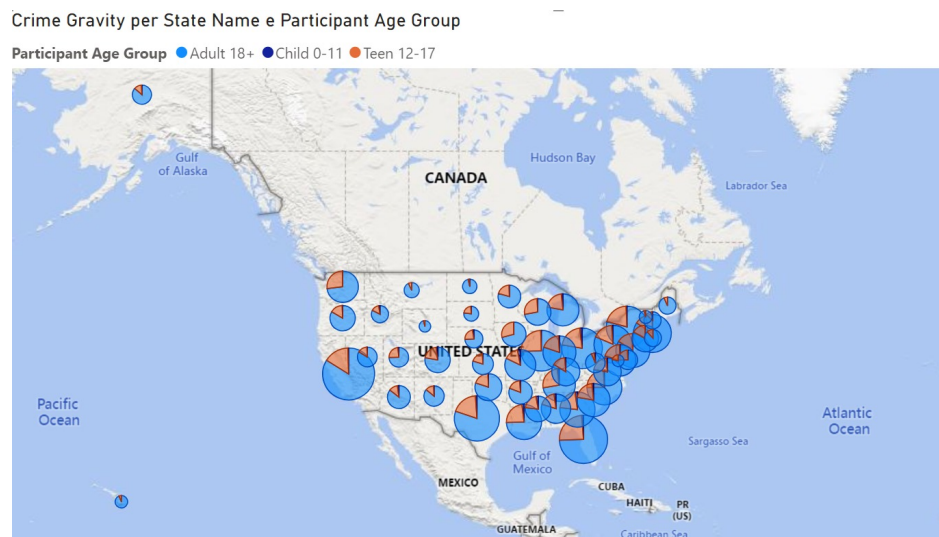


Figure 11: Geographical dashboard

### 3.6 Assignment 5

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

Per ottenere una dashbord informativa del dataset abbiamo optato per 4 diverse soluzioni di visualizzazione:

Nel primo grafico abbiamo voluto analizzare l’andamento dei crimini commessi nello spazio temporale a nostra disposizione. Infatti è possibile notare che negli anni i crimini commessi (secondo il nostro dataset) sono aumentati negli Stati Uniti. Nel 2017 infatti sono stati coinvolti quasi 80.000 soggetti per un numero di incidenti registrato pari a circa 45.000. Dal 2017 al 2018 è registrato un calo perché le nostre informazioni

si fermano al primo trimestre del 2018.

Il secondo grafico ci informa in merito al tipo di armi utilizzate nei crimini dai partecipanti. Questo grafico l'abbiamo trovato importante a conclusione delle nostre analisi in quanto ci permette di avere una visione di insieme sulle modalità di perpetuazione dei crimini. Infatti l'Illinois, inaspettatamente, si mostra come lo Stato con il maggior numero di persone coinvolte in crimini registrati, ma per una grande parte le armi usate sono state "irrilevanti". Probabilmente, quindi, sono diffusi molti episodi di risse senza l'utilizzo di armi da fuoco. Dallo stesso grafico è possibile notare come i crimini commessi con armi rubate superano gli altri soprattutto in regioni ricche come la California, il Massachusetts e lo stato di Washington.

Proseguendo l'analisi con quanto detto rispetto allo stato dell'Illinois, abbiamo voluto visualizzare informazioni in merito alla distribuzione della gravità dei crimini con la California al primo posto.

Infine come ultimo grafico abbiamo voluto analizzare la composizione dei soggetti coinvolti nei crimini. Possiamo notare la grandissima prevalenza di soggetti maschili e che circa la metà dei partecipanti sono stati o uccisi oppure feriti, confermando la prevalenza di crimini molto violenti con l'utilizzo ampio di armi da fuoco nel nostro dataset.

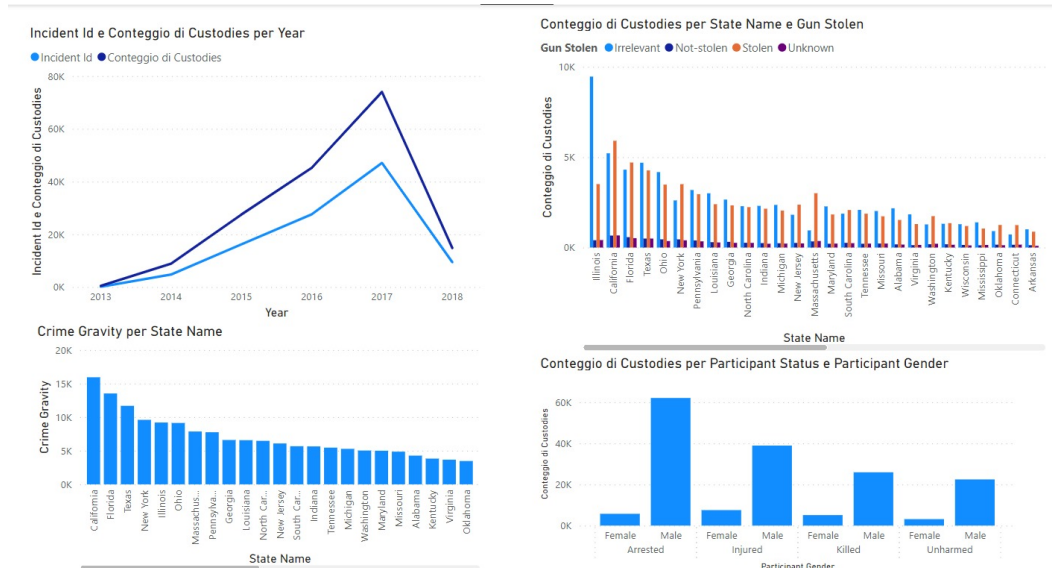


Figure 12: Dashboard