CSC 365
Spring 2023
Alex Dekhtyar

Lab 8

Due: Thursday, June 15, 8:00pm

Note: This is a hard deadline, there will be no extensions.

Overview

This is an individual lab. For this lab you will study a new database made available to you (see below), decide what questions you want to attempt to answer, and create a Python Jupyter notebook that connects to the database, and uses SQL queries to extract the data for your subsequent analysis and visualization.

The key purpose of this lab is to demonstrate the use of relational databases in a somewhat realistic context. Traditionally, databases are used for two core types of services: (i) maintenance of organization-wide operations (e.g., product inventory and sales, HR management, or course enrollments), and (ii) data analytics. The realities of this course make it impossible to build an "operational" database-supported application of any reasonable depth. However, data analytics can be performed at a relatively small scale, hence the subject matter of this lab.

The URBANUNREST Database

You will be working with a database I call URBANUNREST. This database represents a significant portion of the Urban Social Disorder dataset available on Kaggle:

https://www.kaggle.com/datasets/waltervanhuissteden/urban-social-disorder

The Urban Social Disorder dataset (and thus, the URBANUNREST database) contains a fairly comprehensive list[1] of urban civil unrest events around the world that took place since 1967 and ending in 2014.

The Kaggle dataset consists of two "core" files, ***events.csv*** and ***reports.csv,*** and three files containing summary statistics: ***cities.csv, citymonth.csv,*** and ***cityyear.csv.***

The ***events.csv*** file contains a list of urban social disorder events. For each event, the CSV file contains information about the country and the city where it took place (including geographic location), the dates

[1] There is a certain bias in this dataset as it selects primarily the countries from Latin and South America, Africa, and Asia, and does not contain information about similar events in North America, Europe, or Australia/New Zealand. For the remaining countries, the dataset codebook (available at https://www.kaggle.com/datasets/waltervanhuissteden/urban-social-disorder?select=USD+Codebook.pdf) documents the rules for inclusion of the events in the dataset.

(start and end) of the event, the event type, number of participants, number of resulting deaths (if any), the actors (people/groups participating in the unrest) and targets (people/groups the actors were against) of the unrest, and a brief summary of the event.

The ***reports.csv*** file contains a list of full-text reports about individual urban social unrest events found in the ***events.csv*** table.

The ***cities.csv, citymonth.csv,*** and ***cityyear.csv*** files contain overall (***cities.csv***), month-by-month (***citymonth.csv*****)** and annual (***cityyear.csv*****)** statistical summaries of the protests, documenting total number of events, number of events that resulted in participant/target/bystander deaths, and number of events that did not result in deaths.

The **URBANUNREST** database is designed to store the data from the ***events.csv*** file in a relational database, and organizes this storage by breaking the data into several separate tables. The URBANUNREST database omits certain available in the Kaggle dataset (both for convenience, and pedagogical reasons). Specifically:

- the contents of the ***reports.csv*** file are not stored in the URBANUNREST database. This is done primarily to streamline the data. Later versions of the URBANUNREST database may include this information.
- All summary statistics from the ***citymonth.csv, cityyear.csv,*** and ***cities.csv*** are omitted from the URBANUNREST database. This information is derivative from the data stored in the ***events.csv*** file, and can be obtained via grouping and aggregation from the URBANUNREST database.
- The Kaggle dataset stores for each unrest event both the current country where the event took place, and the historic country - i.e. the country as it existed at the time when the event took place. To simplify the URBANUNREST database, we only store the current country information.
- Some of the text in the Kaggle dataset is in UTF-8 encoding and contains Unicode characters. To simplify parsing, URBANUNREST dataset is in ASCII, and therefore, the contents of some fields of some records have missing, or mangled characters. While this will be fixed in the future versions of the URBANUNREST database, this issue should not prevent you from completing your Lab 8 tasks successfully.

The CREATE TABLE statements for the URBANUNREST database is made available to you. In addition, please read the database description below. Please note, that while this description is consistent with the Kaggle dataset Codebook[2], the database looks different and its description contains additional information regarding the coding of the data (I made some changes to the way in which the data is represented in order to save you some parsing trouble).

The URBANUNREST database consist of six tables briefly described in Table 1.

---

2 https://www.kaggle.com/datasets/waltervanhuissteden/urban-social-disorder?select=USD+Codebook.pdf

Table Name	Descripion
ProblemTypes	List of types of social unrest events found in the **Events** table
Countries	List of countries in which urban social disorder events took place
Cities	List of cities in which urban social disorder events took place
Events	List of urban social disorder events
EventActors	List of event actors for each urban social disorder event
EventTargets	List of event targets for each urban social disorder event

Table 1. URBANUNRESET database tables.

We describe each table in turn.

Table ProblemTypes

The **ProblemTypes** table contains a comprehensive list of different types of urban unrest events. This table contains the following attributes:

Attribute Name	Type	Description
Ptype	INT	Unique id of each type
ProblemType	VARCHAR(64)	Name of the type of unrest event
description	TEXT	Definition/Explanation of the type

Table 2. ProblemTypes table.

The codes for the problem types, the names and the descriptions are taken directly from the Urban Social Disorder dataset Codebook (pages 3-4). One additional record (for problem type 99 - 'Unknown") has been added to the **ProblemTypes** above and beyond the codebook information to ensure referential integrity of the data in the database. The list of different event types can be found in Table 3.

Ptype	ProblemType
10	General Warfare
20	Inter-communal Warfare
30	Armed Battle/Clash
31	Armed Attack
40	Pro-Government Terrorism (Repression)
41	Anti-Government Terrorism
42	Communal Terrorism
50	Organized Violent Riot
51	Spontaneous Violent Riot
60	Organized Demonstration
61	Pro-Government Demonstration
62	Spontaneous Demonstration
70	Other
99	Unknown

Table 3. List of social disorder event types stored in the ProblemTypes table.

Table `Countries`

The **`Countries`** table contains a list of countries in which the events described in the **`Events`** table of the database took place. This table contains the following attributes.

Attribute Name	Type	Description
Country	VARCHAR(64)	Name of the country
ISO3	CHAR(3)	ISO 3-letter International Country Code
GWNO	INT	Gleditsch and Ward[3] Independent State Country Code
Region	VARCHAR(32)	Name of the geographic region the country is in

Table 4. Countries Table.

There are three candidate keys in this table: `(Country)`, `(ISO3)`, and (`GWNO`), of which we selected `Country` as the primary key (notice that we actually use - see below - a different candidate key as a foreign key).

Table `Cities`

The `Cities` table contains information about the cities in which the social disorder events described in the `Events` table took place. This table contains the following attributes:

Attribute Name	Type	Description
CityId	INT	Unique Id of the city in the database
City	VARCHAR(64)	Name of the city
Country	CHAR(3)	ISO3 country code of the country the city is in
isCapital	INT	1: capital of its country; 0: not a capital of its country
Longitude	DOUBLE	Longitude of the city
Latitude	DOUBLE	Latitude of the city

Table 4. Cities Table.

Table `Events`

The central table of the URBANUNREST database, the **`Events`** tables stores the list of social disorder events. This table contains the following attributes:

[3] http://ksgleditsch.com/data-4.html

Attribute Name	Type	Description
eventID	VARCHAR(10)	Unique Id of the civil disorder event
City	INT	Id of the city in which the event took place
Ptype	INT	Type of the event (see **ProblemTypes.Ptype**)
StartDate	DATE	Starting date of the event
EndDate	DATE	Ending date of the event
nPart	INT	Number of participants (coded - see below)
minDeaths	INT	Minimal assessed number of deaths at the event
maxDeaths	INT	Maximal assessed number of deaths at the event
deathFlag	INT	1: event resulted in at least one death; 0: no deaths, -1: unknown
eLocal	VARCHAR(128)	Specific location of the event (if known) in the city
summary	TEXT	Summary of the event

Table 5. Events Table.

Please read the Kaggle Dataset Codebook for more information about the way the events are coded. In converting the data from the Kaggle Dataset into the **Events** table in the URBANUNREST database we made the following coding decisions:

Start and end dates of the events. The ***events.csv*** file in the Kaggle dataset stores separately Month, Day, and Year of the start and end dates of the event. I elected to replace them with a single DATE attribute for each of the two dates. In doing so, the following decisions were made:

- In the ***events.csv*** file, the Day, and the Month of the date (and - on some rare occasions, also the Year) may occasionally be unspecified, meaning that there is uncertainty about the exact event start/end dates. To code this in the `Events` table I represent missing data with a "00" (or "0000") value for the missing component of the date.
    - for example, a missing day in June of 2000 is coded as **"2000-06-00"** while a missing month and day in the year 1975 is coded as **"1975-00-00".**

  The decision to code this way was made because (a) MySQL admits these kinds of DATE values, and (b) **correctly identifies the known parts of these dates.** For example,

  DATE_FORMAT("2000-06-00", "%M")

  properly identifies the month (June) in this date despite missing day of month information.

This allows you to work with the known components of each date, while using only one attribute to store the date information.

Note: this leads to some (very few) dates being rendered as "0000-00-00" - when no information about the time of the event exists in the Kaggle dataset.

Number of participants in the event. `Events.nPart` attribute stores information about the number of participants in each event. This information is coded to represent orders of magnitude of the participation. The codes can be found in the dataset Codebook. For the sake of completeness, they are also included in Table 6.

Code	Range of values
1	less than 10 participants
2	10 - 100 participants
3	101 - 1,000 participants
4	1,001 - 10,000 participants
5	10,000 - 100,000 participants
6	100,000 - 1,000,000 participants
7	over 1,000,000 participants
11	Unknown, but probably relatively small number (less than 1,000) of participants
12	Unknown, but probably large number (1,000 - 100,000) of participants
13	Unknown, but probably very large number (over 100,000) of participants
99	Unknown (and no estimates exist)

Table 6. Values of Events.nPart attribute (number of participants in an event).

Number of deaths estimates. The Kaggle dataset has a variety of ways in which the number of deaths are recorded for different events. This primarily depends on how precise the available information is about the event. In some cases, the precise number of deaths is known. In some cases, there is a range. In some cases it is known that the number of deaths was greater than (or equal to) some number, or less than (or equal to) some number. Finally, in some cases, there is no information about whether any deaths took place, but the possibility of deaths cannot be ruled out. The treatment of the death estimates in the Kaggle dataset lacks consistency and presents several parsing challenges.

To that end, I elected a different (from the Kaggle dataset) approach for recording the possible number of deaths resulting in a given social disorder event. This information is stored in three attributes: `minDeaths`, `maxDeaths`, and `deathFlag`.

- **`minDeaths` and `maxDeaths`.** These two attributes contain the best estimates of the minimum and the maximum number of deaths at a social disorder event, based on information provided in the ***events.csv*** file of the Kaggle dataset. The information is organized as follows:
    - If the event resulted in NO deaths, both `minDeaths` and `maxDeaths` are set to 0.
    - If the event resulted in a known number of deaths, we set `minDeaths = maxDeaths` with the given number of deaths.
    - When the upper bound on the number of deaths is known, but lower bound is not, `minDeaths` is set of 0, and `maxDeaths` is set of the upper bound (or upper bound - 1, depending on whether the upper bound is inclusive or exclusive).
    - When the lower bound on the number of deaths is known, but the upper bound is not, `minDeaths` is set to the lower bound (or lower bound +1 depending on whether the lower bound is inclusive or exclusive), while `maxDeaths` is set to the value of **-1.**
    - If both the lower and the upper bounds on the number of deaths are known, then both minDeaths and maxDeaths are set to the appropriate lower (`minDeaths`) and upper (`maxDeaths`) bounds taking into account whether those boundsa are inclusive or exclusive.
    - When the number of deaths is not known, ***both*** `minDeaths` and `maxDeaths` are set to -1.
- **`deathFlag`.** Because figuring out whether an event resulted in deaths based on the logic of assigning values to `minDeaths` and `maxDeaths` is more complex than the simple **`minDeaths > 0`** check, we include a definitive flag indicating whether an event results in any deaths.
    - This flag is set to 1 for ALL events where `minDeaths > 0`
    - In the case when `minDeaths = 0` and `maxDeaths=0` deathFlag is set to 0.
    - It is also set to 1 if the number of deaths is in the range of `[ 0, maxDeaths]`, where `maxDeath > 0`. This is done on the assumption that if a non-zero upper estimate on the number of deaths is given, in all likelihood it means that at least one person died.
    - `deathFlag` is set to **`-1`** ("Unknown"), if both `minDeaths` and `maxDeaths` are set to -1.

Specific Event Location. Some events have a specific location associated in which they took place. This information is stored in the `ELOCAL` column of the ***events.csv*** Kaggle dataset file, and it is copied into the `Events.eLocal` column in the `Events` table. This information however, is not universally available. Whenever this information is not available in the Kaggle dataset for a given event, we set the value of the `eLocal` attribute to `'N/A'`.

Table `EventActors`

Event actors are people/forces/organizations/entities that took an active part in the social disorder event. The Kaggle dataset identifies up to three different actors per each event. Some events have missing event actor information. The `EventActors` table documents the actors for each civil disorder event in the Events table (when this information is available). This table contains the following attributes:

Attribute Name	Type	Description
`eventID`	VARCHAR(10)	Unique Id of the civil disorder event (see `Events.eventId`)
`actor`	VARCHAR(200)	Description of an event actor.

Table 7. EventActors Table.

Table `EventTarget`

Event targets are people/forces/organizations/entities (and sometimes issues) that serve as a focus of the protset/activities during a social disorder event. The Kaggle dataset identifies up to two different targets per each event. Most events have just one target, some events have no specified target. The `EventTargets` table documents the actors for each civil disorder event in the Events table (when this information is available). This table contains the following attributes:

Attribute Name	Type	Description
`eventID`	VARCHAR(10)	Unique Id of the civil disorder event (see `Events.eventId`)
`target`	VARCHAR(200)	Description of an event target

Table 5. EventTargets Table.

Final note about this dataset. While this dataset appears to be somewhat synthetic, it, and similar datasets are an important tool used by political and social scientists in the studies of political and societal changes. I have first-hand experience of working with a ***very similar***[4] dataset as part of an industry project, where our team was trying to predict if a given protest would turn violent, and understand what protest characteristics (features) correlated with violence. The dataset we used had more features, but significantly fewer data points. So, while by modern standards a dataset consisting of under 10,000

[4] In addition to information about deaths, the dataset we worked with also included explicit information about whether an event results in any violence (non-lethal violence included), and broke the victims of the violence into protesters (actors) and targets.

records is not considered large, this dataset is large for the application domain it comes from, and (especially when enhanced with the information from the ***reports.csv*** file) can be used for some in-depth analysis of trends in civil disorders across (significant part of) the globe.

Database Access

The **`mysql.labthreesixfive.com`** MySQL server we use in our class has a new database accessible to you in a read-only format: LAB8. This database is not accessible via the Lab365 lab portal, but you can freely access it via the mysql client, any DBMS IDE (such as DataGrip), or programmatically, via SQL connectivity functionality in a host programming language.

Your Task

The URBANUNREST dataset can be used in a variety of ways to study the trends associated with civil disorder. For this assignment, you will study the dataset, do the following.

- **Ask five analytical questions for which you would need to extract the data from the URBANUNREST database.** I am giving each of you the leeway in deciding what questions you want to ask. To me "analytical" means that the answer to the question is a result of some manipulations of the "raw" data from the database, and that this result can provide some level of insight the trends in social disorder.

  I am going to **leave it up to you to choose your five questions.** However, I reserve the right to decrease your grade for this lab, if you select trivial questions that (a) are easy to answer (and don't require much in terms of SQL to find the answers), and/or (b) not interesting/insightful.

- Create **one** Jupyter notebook (you can use the course Jupyter) that for each analytical question:
  - extracts the data necessary to answer the question from the URBANUNREST database using SQL connectivity and SQL queries
  - visualizes (using programmatically constructed tables, graphs, charts, figures) the answer to the question
  - contains Markdown text explaining the question you are asking, your approach to collecting the results, and visualizing it, and (at the end of each question's narrative) - your observations/conclusions.

Your code for the assignment shall meet the following expectations:

- **Most** of the data processing necessary to answer the questions you are asking **must be done in SQL.** Your analytical questions may require retrieval of information using multiple SQL queries (unlike the problems in Labs 4-7), **which is fine.** But your queries must return the exact (as much as possible) data needed for the question.
  - for example, simply running a

```
SELECT * FROM Events;
```

query in SQL, and then extracting all information about peaceful demonstrations in African countries from it is NOT ACCEPTABLE and will result in penalties.

- Python postprocessing of the data returned by your SQL queries shall be reserved only for activities that are either impossible or **very hard/inefficient** to complete in SQL.
- You can use ANY SQL syntax for writing your queries, but be aware that using the LIMIT clause may be considered incorrect if there are possible instances of the database on which it returns incorrect results (even if it returns correct results on the current instance of the data).

Grading. Unlike most other labs, where I care about the correctness of your queries, here, I will also evaluate the sophistication of your solution. You should be asking fairly complex questions, and SQL queries matching those questions must be sufficiently complex. As such, for each of the five questions I adopt the following grading rubric:

1. **15% Quality of the question.** Is your question interesting and will the answer to it be insightful?
2. **30% Correctness of implementation.** Did your code deliver the correct answer to your question?
3. **10 % Quality of implementation.** How sophisticated are your SQL queries, how well-designed is your solution (I primarily consider the SQL parts, as this is a database course, but I will look at the overall readability of your code).
4. **20% Results and visualizations.** The quality and appropriateness of the output/visualizations.
5. **15% Insight and explanation.** This measures how well you are commenting on the results obtained and how sound your conclusions are.
6. **10%. Compliance and ease of running.** This part measures how easy it was for me to run your code, observe results and score your work.

Submission.

Do all your analysis in a single Jupyter notebook. Call your notebook **Lab8-<username>.ipynb** where **<username>** is your Cal Poly username (e.g., my submission would be called **Lab8-dekhtyar.ipynb**).

Submit using the `handin` utility:

```
$ handin dekhtyar 365-lab08 <filename>
```

PS. I know that this assignment stretches somewhat beyond the scope of this class. However, it is meant to be a fun exercise which is relatively easy to accomplish for anyone approaching this assignment **in good faith** and with some basic command of `pandas` and `matplotlib`.

GOOD LUCK!