M149/M116: Database Management Systems
University of Athens
Deprt. of Informatics & Telecommunications

*Programming Project II*
Today's Date: January 7th, 2025
Due Date: February 14th, 2025

## PREAMBLE

In this project, you will design, implement and demonstrate a *NoSQL*–database solution to manage *"Crime Data"* openly published by the Los Angeles Police Department. You will also provide access to your database through a *REST API*.

The database you will create, termed `NoSQL-LA-CRIME`, will be populated with data already collected and available at:
`https://data.lacity.org/Public-Safety/Crime-Data-from-2020-to-Present/2nrs-mtv8/about_data`.

Pertinent data of the above data set have to be *stored* in `NoSQL-LA-CRIME` and can be used when the system becomes operational.

## PROBLEM DESCRIPTION:

Your database should include information about the *LAPD* crime reports and should provide various queries and/or aggregations on crimes committed. Moreover, the database should hold data related to report upvotes casted by police officers. Below, you can find a short description of the general guidelines of the project. While you are working on this project, keep in mind that these items make up a minimum set of requirements. Hence, you may extend the scope of your work as you see it appropriate and/or interesting.

### LAPD Crime Reports data:

There are *28* fields available in the single (`.csv` formatted) file used to keep track of all types of crimes reported in Los Angeles.

1. `DR_NO` or Division of Records Number: Official file number made up of a 2 digit year, area ID, and 5 digits. For example: `190326475` where `19` is the year of incident, `03` is the ID of the area, and `6475` is the record number.
2. `Date Rptd`: Date crime reported with format MM/DD/YYYY.
3. `DATE OCC`: Date crime occurred with format MM/DD/YYYY.
4. `TIME OCC`: Time crime occurred expressed in 24 hour military time.
5. `AREA`: The LAPD has 21 *Community Police Stations* referred to as geographic areas within the department. These areas are sequentially numbered from 1–21.
6. `AREA NAME`: The above 21 areas also known as "*patrol divisions*," are also given a name designation that references a landmark or the surrounding community. For example *77th Street Division* is located at the intersection of South Broadway and 77th Street, serving neighborhoods in the broader South Los Angeles.

7. `Rpt Dist No`: This is a 4-digit code that represents a sub-area or *Reporting District (RD)* within a geographic area. Every crime crime record features the "*RD*" number the crime occurred in. This is done for statistical comparisons. You can find the LAPD Reporting Districts (RDs) on the LA City GeoHub at:
   `http://geohub.lacity.org/datasets/c4f83909b81d4786aa8ba8a74a4b4db1_4`

8. `Part 1-2`: not applicable fields.

9. `Crm Cd`: Indicates the crime committed. This is the same as `Crime Code 1` below.

10. `Crm Cd Desc`: Defines the Crime Code provided.

11. `Mocodes`: This is about modus operandi or activities associated with the suspect in commission of the crime.

12. `Vict Age`: Two character numeric.

13. `Vict Sex`: F for Female, M for Male, X for Unknown.

14. `Vict Descent`: A letter indicating the ethnic background of the victim. In particular, the letters used for such abbreviations are:

| | |
|---|---|
| A | Other Asian |
| B | Black |
| C | Chinese |
| D | Cambodian |
| F | Filipino |
| G | Guamanian |
| H | Hispanic/Latin/Mexican |
| I | American Indian/Alaskan Native |
| J | Japanese |
| K | Korean |
| L | Laotian |
| O | Other |
| P | Pacific Islander |
| S | Samoan |
| U | Hawaiian |
| V | Vietnamese |
| W | White |
| X | Unknown |
| Z | Asian Indian |

15. `Premis Cd`: The type of structure, vehicle, or location where the crime took place.

16. `Premis Desc`: Defines the Premise Code provided.

17. `Weapon Used Cd`: The type of weapon used in the crime.

18. `Weapon Desc`: Defines the Weapon Used Code provided.
19. `Status`: Status of the case where the default value is `IC` or investigation continues.
20. `Status Desc`: Defines the status code provided.
21. `Crm Cd 1`: Indicates the crime committed. Crime Code 1 is the primary and most serious one. Crime Code 2, 3, and 4 are respectively less serious offenses. Lower crime class numbers are more serious.
22. `Crm Cd 2`: May contain a code for an additional crime, less serious than Crime Code 1.
23. `Crm Cd 3`: May contain a code for an additional crime, less serious than Crime Code 2.
24. `Crm Cd 4`: May contain a code for an additional crime, less serious than Crime Code 3.
25. `LOCATION`: Street address of crime incident rounded to the nearest hundred block to maintain anonymity. Cross Street: Cross Street of rounded Address
26. `LAT`: Latitude
27. `LON`: Longitude

Using `MongoDB` Community Server,[1] create the *collections* of your database, and insert *all* pieces of information. You have the freedom to define the collections as per your own choice/design.

Upvote Data: Police officers should be able to cast upvotes to each record, to highlight their importance. Upvotes are accompanied with the name, e-mail and badge number of the police officer.

REST API: You should create an API method for each of the following queries, as they are particularly useful to the context of `NoSQL-LA-CRIME`. The responses of your API should use the JSON format:

1. Find the total number of reports per '`Crm Cd`" that occurred within a specified time range and sort them in a descending order.
2. Find the total number of reports per day for a specific "`Crm Cd`" and time range.
3. Find the three most common crimes committed –regardless of code 1, 2, 3, and 4– per area for a specific day.
4. Find the two least common crimes committed with regards to a given time range.
5. Find the types of weapon that have been used for the same crime "`Crm Cd`" in more than one areas
6. Find the fifty most upvoted reports for a specific day.
7. Find the fifty most active police officers, with regard to the total number of upvotes.
8. Find the top fifty police officers, with regard to the total number of areas for which they have upvoted reports.

[1] `https://www.mongodb.com/download-center/community`

9. Find all reports for which the same e-mail has been used for more than one badge numbers when casting an upvote.

10. Find all areas for which a given name has casted a vote for a report involving it.

In addition, you should provide API methods for updating the database. Updates may include *i)* inserting new reports and *ii)* casting of upvotes. In case the same police officer casts a vote for the same report a second time, the vote should be rejected.

It is particularly important that you make sure that the above queries are executed *efficiently* by creating the appropriate *collections* and adding the necessary *indices*.

## IMPLEMENTATION ASPECTS:

You will use `MongoDB` as your database in this project. In addition, you may use any language/framework you want including `Spring-boot`, `Laravel`, `Django`, `Ruby On Rails`, `Flask`, `Express.js Java, Python, PHP`, etc.

You may work in any environment you wish but at the end you should be able to demonstrate your work (with your notebook or via remote access to a machine).

## OVERVIEW OF PROJECT PHASES:

There are two distinct phases in this project that you will need to work on:

### ◇ Database

In this phase, you are required to decide upon the database collections you will use and populate them with data. For the police officer data you are expected to generate the data yourselves. To this end you could employ an open-source library such as `faker`[2][3], `java-faker`[4], and `datafactory`.[5] You are expected to generate enough data so that at least $\frac{1}{3}$ of the reports have at least one upvote, and no police officer has more than 1000 upvotes.

### ◇ REST API

You will use the framework of your choice (such as `Spring-boot`, `Laravel`, `Django`, `RoR`, `Flask`, `Express.js` etc.) to provide an API that offers access to the database and enables users to query and update the database.

[2]`https://github.com/stympy/faker`
[3]`https://github.com/joke2k/faker`
[4]`https://github.com/DiUS/java-faker`
[5]`https://github.com/andygibson/datafactory`

## COOPERATION:

You may either work individually or pick *at most one partner* for this project. If you pick a partner you should let us know who this person is.

## REPORTING:

The final *typed* project report (brief report) must consist of:

1. A description of the schema design of the database used along with justification for your choices.
2. The `MongoDB` queries for the required functionality.
3. The code of your *REST API*, preferably through a link to a `git` repository.
4. Sample responses for each query.

Finally as mentioned earlier, you will have to demonstrate your work.

## SUPPORT:

Panagiotis Liakos (`p.liakos+@-di.`) will be escorting this assignment, fill in questions, and carry out the final interviews.

## References

[1] MongoDB, *MongoDB Architecture Guide*, `https://www.alexdelis.eu/M149/references/MongoDB-Arch.pdf`https://www.alexdelis.eu/M149/references/MongoDB-Arch.pdf, White Paper

[2] Kyle Banker et al., *MongoDB in Action*, `https://www.alexdelis.eu/M149/references/MongoDB-InAction.pdf` 2nd Edition, 2016.

[3] Kristina Chodorow, *MongoDB: The Definitive Guide*, `https://www.alexdelis.eu/M149/references/MongoDB-DefGuid2nd.pdf`, 2nd Edition, 2013.