Introduction

The Philippines, a nation comprised of diverse islands and cultures, stands at a crucial juncture in its pursuit of Sustainable Development Goals (SDGs). As the country strives to address various socio-economic and environmental challenges, the need for a comprehensive system to monitor and analyze SDG indicators becomes paramount. In this context, the decision to embark on the creation of a relational database dedicated to Philippine SDG indicators emerges as a strategic response to the existing gaps in data management.

The archipelagic nature of the Philippines presents unique challenges in aggregating and interpreting data across regions, making it imperative to streamline information flow for effective decision-making. The absence of a centralized and user-friendly platform hinders not only the monitoring of progress but also the formulation of targeted policies that can drive positive change.

This paper delves into the rationale behind choosing this specific problem—highlighting the intricate web of challenges that hinder the nation's ability to holistically address SDGs. By developing a relational database tailored to Philippine regions, we aim to provide a tailored solution that aligns with the nation's specific needs and accelerates progress towards a sustainable and equitable future.

Statement of the Problem/Objectives

Problem: In the current landscape of SDG monitoring in the Philippines, a significant gap exists—while there is a national monitoring system available on the SDG website, the granularity required for effective decision-making is lacking. The absence of detailed regional data hinders the ability to pinpoint specific challenges, understand localized trends, and formulate targeted interventions. As such, there is a critical need to enhance the existing monitoring infrastructure by delving into the intricacies of SDG indicators at the regional level.

Objectives: The primary objective of this project is to address the deficiency in regional data representation by creating a comprehensive and user-friendly relational database. This database will adhere to both normalized and denormalized models to accommodate varied analytical needs.

1. **Normalized Database Model:**
    - Develop a normalized database model to ensure efficient data storage, minimize redundancy, and maintain data integrity.
    - Establish relationships between entities, reflecting the intricate connections within the SDG indicators.

2. **Denormalized Database Model:**
    - Create a denormalized database model to facilitate quick and simplified data retrieval for dashboard visualization.
    - Optimize the structure for faster query performance, catering to the specific requirements of regional data analysis.
3. **Regional Dashboard Recreation: (Only For Discussion)**
    - Utilize the relational database to recreate a static dashboard specifically tailored to the diverse regions of the Philippines.
    - Implement visualization tools that allow for easy interpretation of regional SDG data, enabling stakeholders to identify areas that require focused attention.

By achieving these objectives, this project aims to empower decision-makers with a robust tool that not only visualizes the current state of SDG indicators in each Philippine region but also guides targeted interventions for sustainable development.

Result And Discussion

On this section we tackle on creating two database model(normalized model and Dimensional Model) and demonstrate if we possible visualize the data inside the serverless database namely SQLite and plotly module using Jupyter Notebook . In addition is it possible to migrates geospatial data in SQLite even the data type has a heavy load lead to 1gb plus data and compressed to 100 times smaller to its original size. But before that I explained why “SQLite” used on this capstone .

SQLite has very little overhead to set up and maintain as long as you don't have to share data on a network. The fact that it has less bells and whistles is also a plus for learning in my opinion. It lets you focus on relational database fundamentals.

Many projects and apps will never need more than SQLite provides, so I think it's honestly a great technology to have in your back pocket for bootstrapping applications. If you get to the point where SQLite can no longer handle your demands, break out the champagne because you've got a popular app on your hands.

Creation of Normalized Model

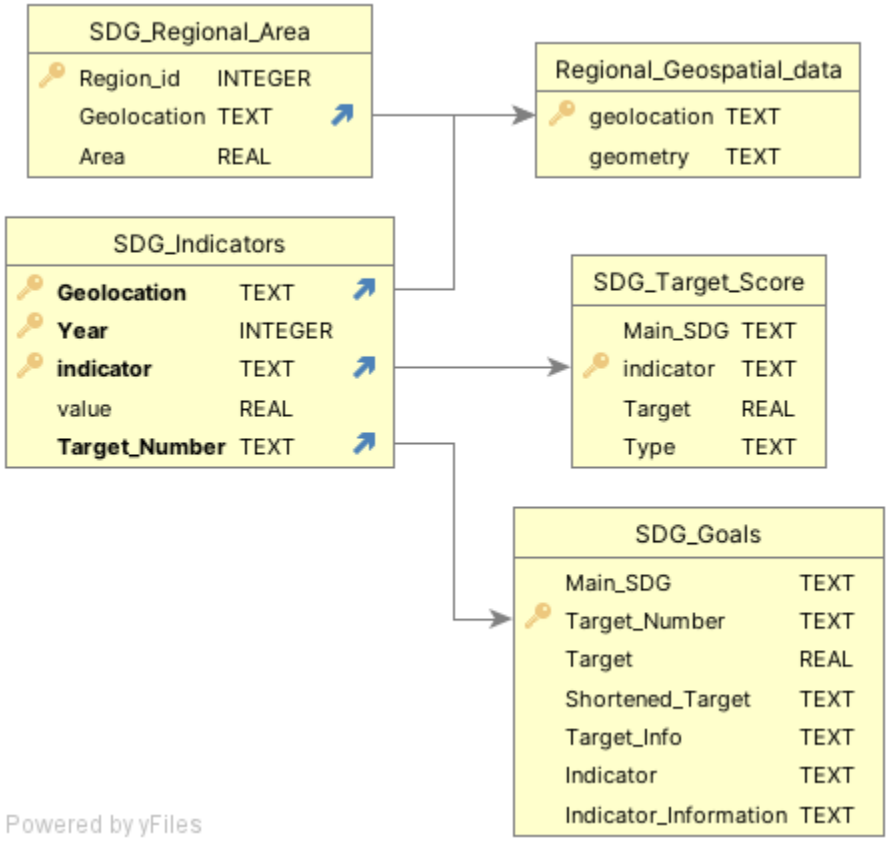


Fig. 1.1 ERD of Normalized Model

Normalized model deal on relationship of every entity for one of them and depend on what attributes link to their entities . Logically inherit and we need to rely on each step Normalization Form 1NF,2NF,3NF,4NF,and so on to ensure the data integrity of your database if it worked perfectly. I will write the bullets of every normalization .

First Normal Form (1NF)

- Using row order to convey information is not permitted
- Mixing data types within the same column is not permitted
- Having a table without primary key is not permitted
- Repeating Groups are not permitted

Second Normal Form(2NF)

- Each non-key attribute in the table must depend on the entire primary key.

Third Normal Form(3NF)

- Each non-key attributes in the table must depend on the key ,the whole key, and nothing but the key.

Fourth Normal Form(4NF)

- The only kinds of multivalued dependency allowed in a table are multivalued dependencies on the key .

Fifth Normal Form(5NF)

- It must not be possible to describe the table as being the logical result of joining some other tables together.

I don't need to paste here all script on how I create the normalized model and migrate the data on the database please view ipynb file related to creating on said model. However, I show the output using DbBrowser proof that it has been created.

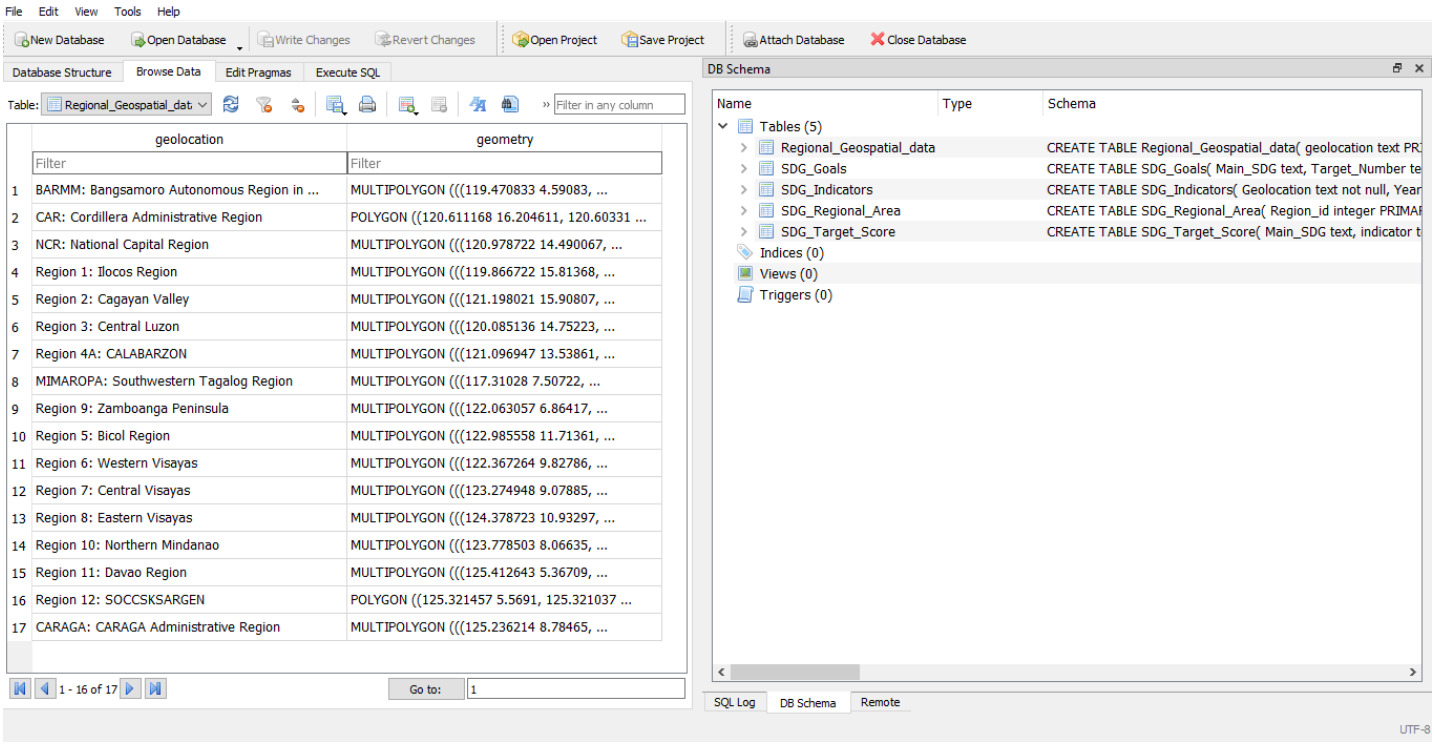


Fig. 1.2 Normalized Database

Creation Denormalized Model (Dimensional Model)

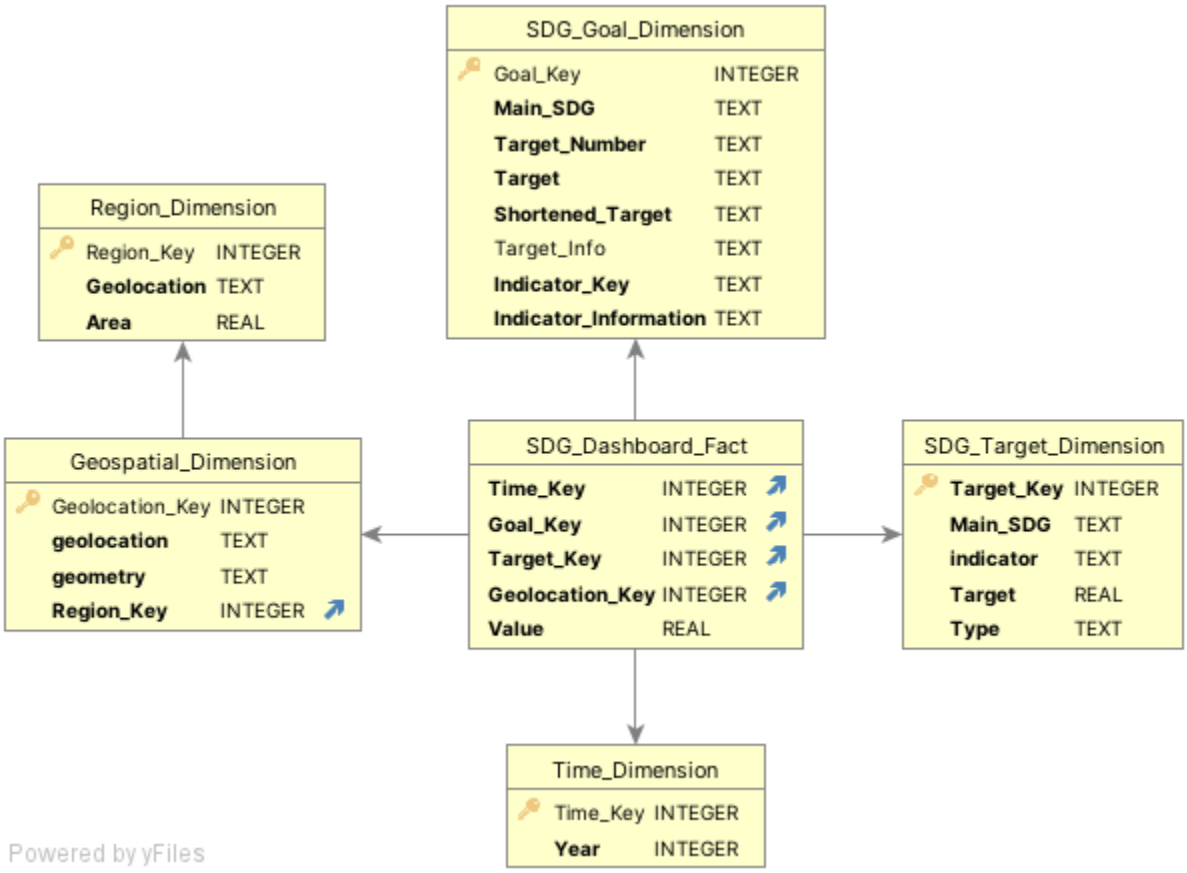


Fig.2.1 ERD of Dimensional Model (SnowFlakes Schema)

In creating denormalized database we need always to consider the part of this model . In a denormalized model, several elements play crucial roles in structuring and organizing data efficiently for reporting and analysis purposes. Here are some important components:

Surrogate Keys: These are unique identifiers for records within a table, typically used as primary keys. In a denormalized model, surrogate keys are essential for maintaining uniqueness and facilitating data retrieval. They're especially important when merging data from different sources or when there isn't a natural primary key available.

Dimensional Tables: In a denormalized model, dimensional tables hold descriptive attributes related to business entities, such as customers, products, time, or locations.

These tables store the details that provide context to the measurements in the fact table. Ensuring these tables are well-designed and denormalized helps in simplifying queries and data retrieval.

Fact Tables: Fact tables contain the numerical or quantitative measurements or metrics that a business wants to analyze. These tables usually store transactional or event-based data and link to the dimensional tables via foreign keys. In a denormalized model, ensuring that the fact table incorporates necessary dimensions and maintains a denormalized structure helps in simplifying complex queries and analysis.

Data Integrity Constraints: In denormalized models, maintaining data integrity is crucial. This involves enforcing constraints such as foreign key relationships between dimensional and fact tables, ensuring referential integrity, and the use of constraints to maintain data accuracy.

Indexes and Optimizations: Denormalized models often benefit from well-placed indexes. Indexes improve query performance, especially when dealing with larger datasets. Carefully selecting the fields for indexing within the denormalized structure is important for efficient data retrieval.

Data Redundancy and Performance: Denormalization often involves duplicating some data for performance reasons. Ensuring that this redundancy doesn't compromise data consistency and managing it effectively is crucial. Redundancy should be strategically employed to enhance query performance without sacrificing data accuracy.

Aggregations and Pre-calculations: In a denormalized model, aggregations and pre-calculations might be stored within the fact table to improve query performance. These can include summary statistics or calculated fields that enable faster analysis.

Balancing denormalization with maintaining data consistency, integrity, and performance is key in a denormalized model. It aims to simplify querying and analysis by reducing the need for joins and complex relationships while ensuring that data remains accurate and meaningful.

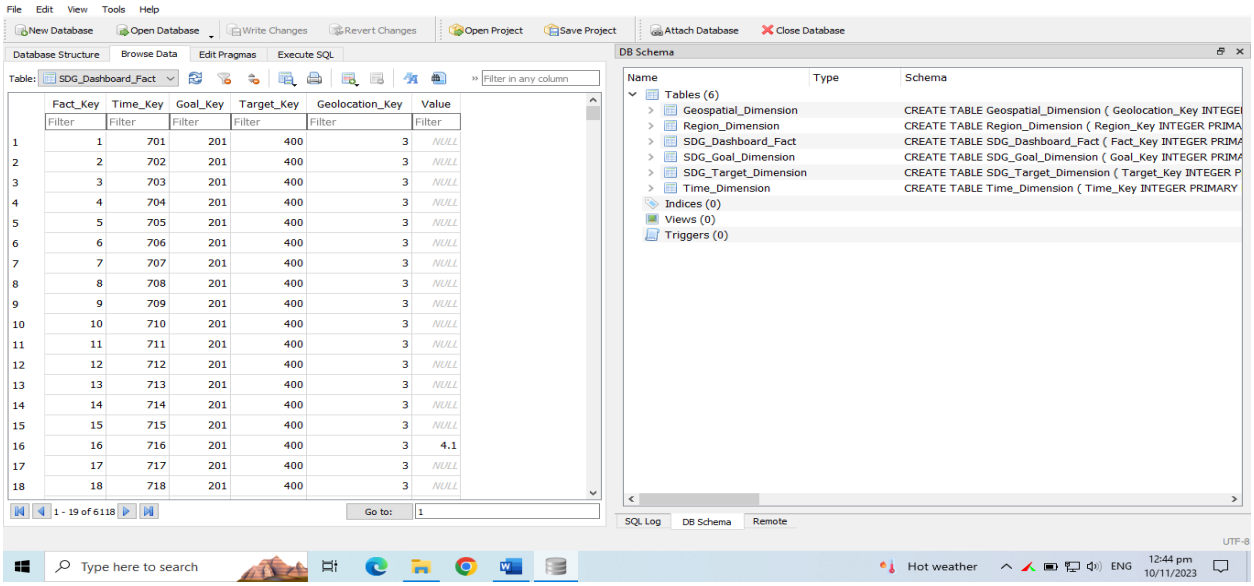


Fig. 2.2 Denormalized Database

We know that denormalization will tend to worsen the performance of a database that's subject to frequent updates.However many data engineers often used this model.One reason is we all know that data in real time is always been dirty,redundant and not organize well . Second,business organization policy and rules changed through time pass by. Lastly it can minimize the time to normalize all data we need . It can be easy now to updates dimensional model if organization changes the criteria ,don't mind the data redundancy , what about the database that predominantly read only.

Data Visualization Query on Database

1.On visualizing the data inside we need to query the data we need.

2. Then convert to dataframes and geodataframes to read it and visualized.

3. use Appropriate visual module on python like plotly ,mathplotlib and etc.

Please open Ipynb file on data visualization using jupyter notebook I'll put all output chart here as proof.

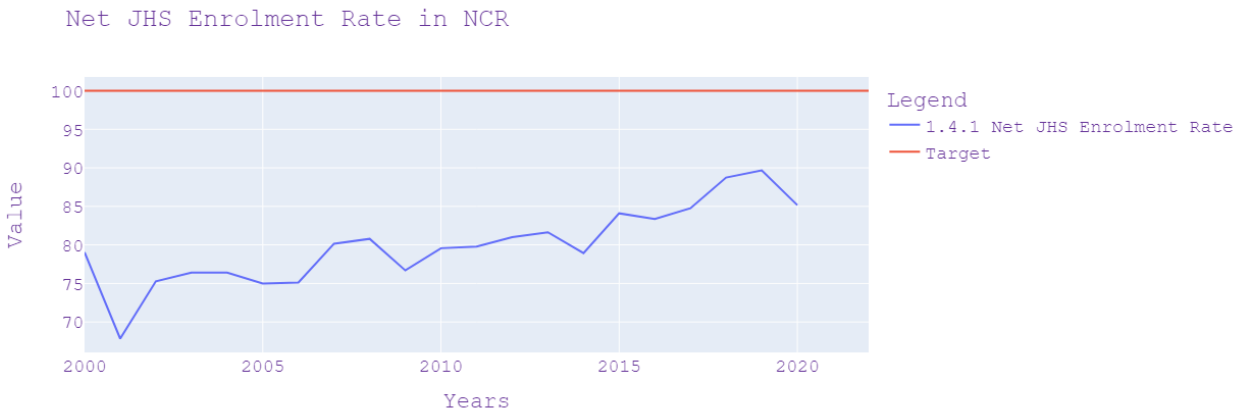


Fig. 3.1 Line Chart of Selected Indicators and Region

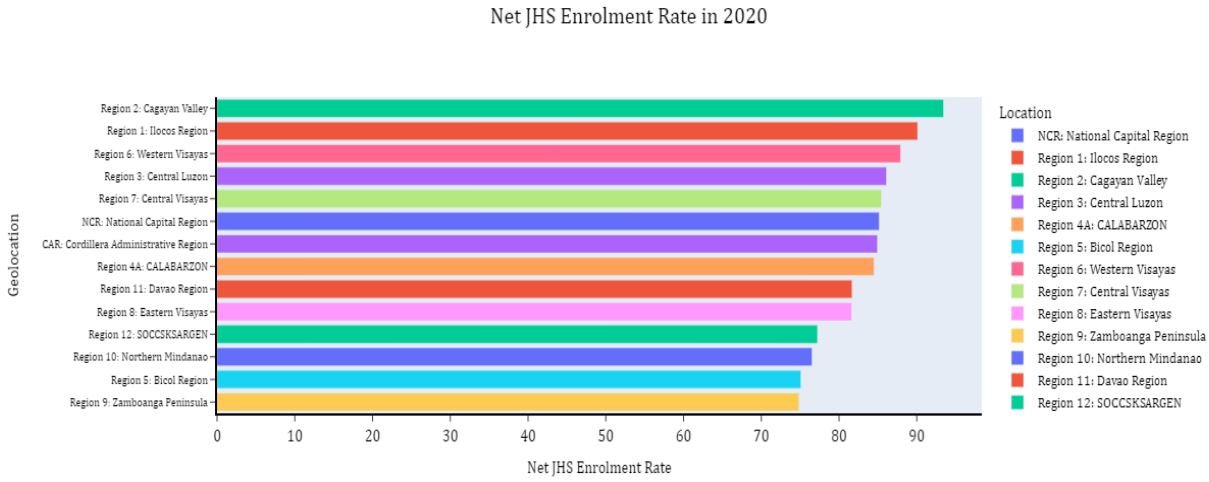


Fig.3.2 Bar Chart of Selected Indicators

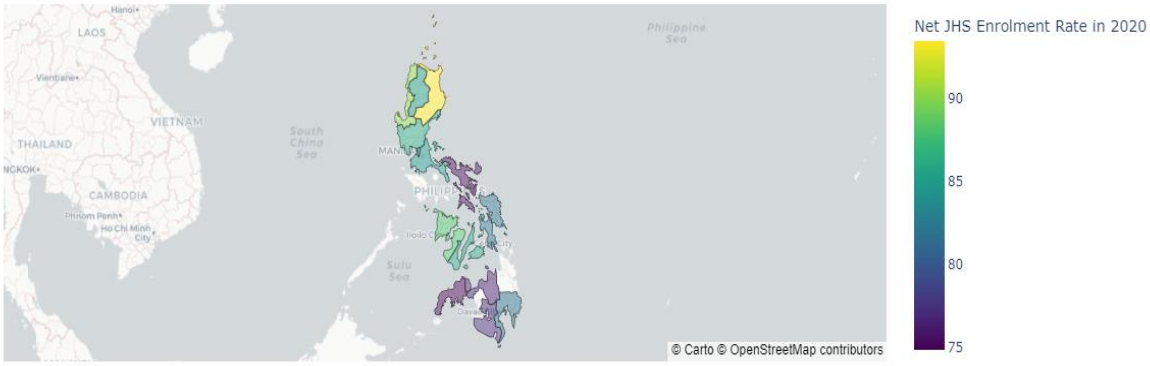


Fig. 3.4 Choropleth Map of Selected Indicators

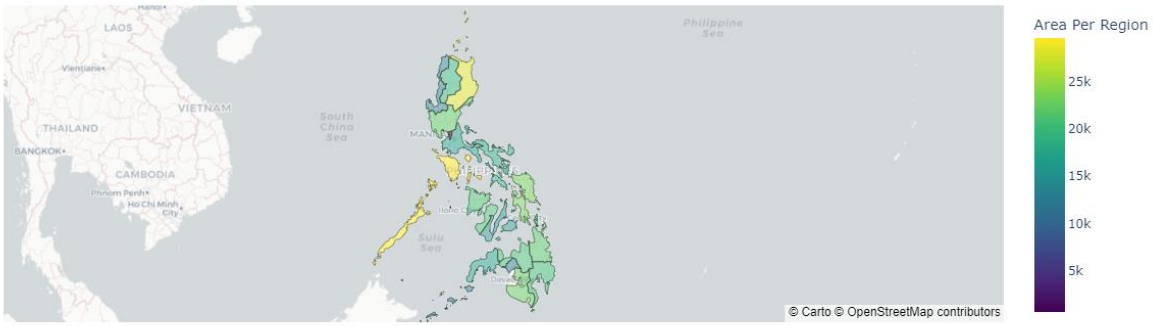
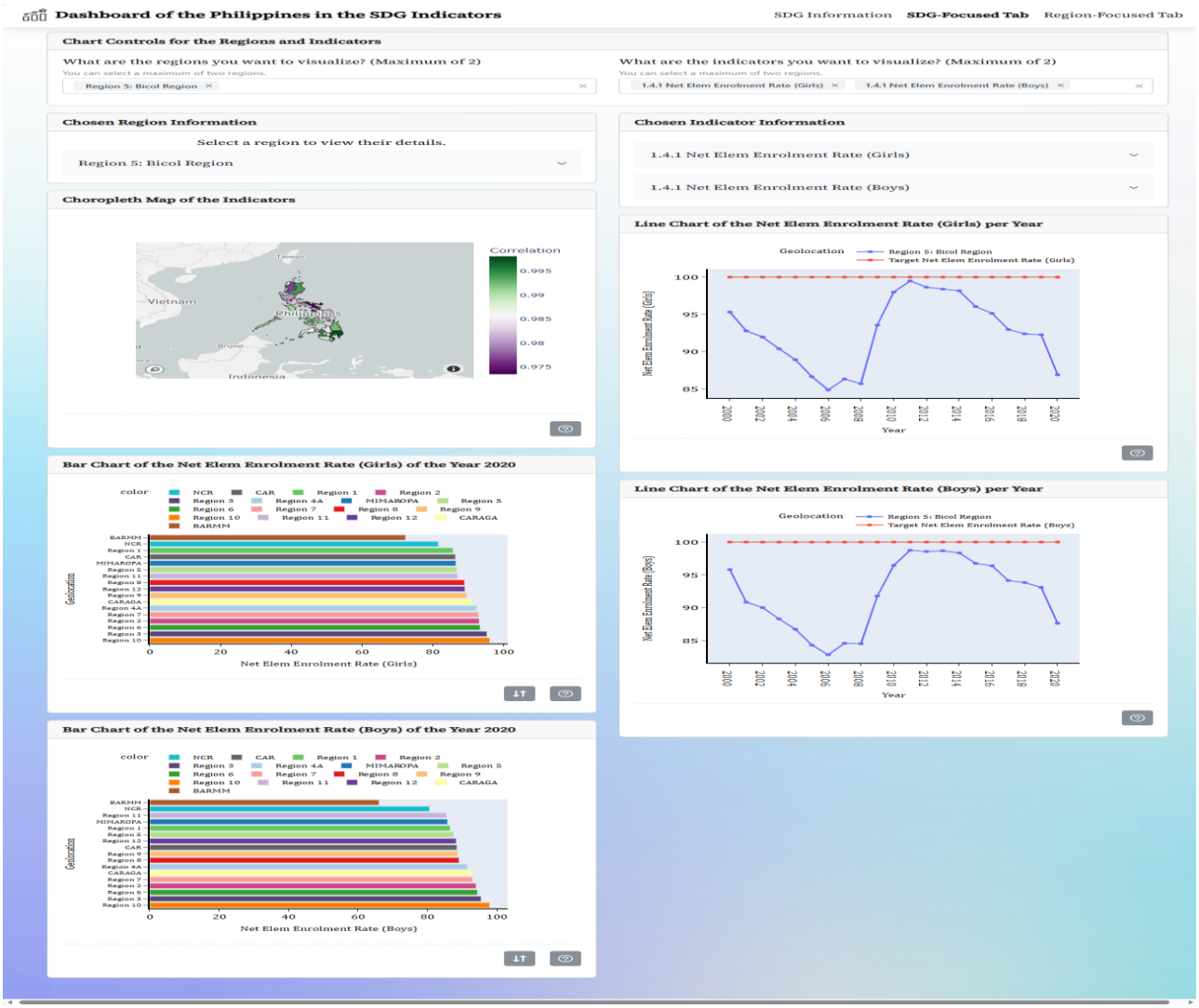


Fig. 3.5 Choropleth Map Area per Region

Final Dashboard

We can use Dash for creating interactive user interfaces in python to run our project perfectly.In Addition we need also other modules furthermore I don't want to explain here because I only make a simple documentation for this capstone As part of the requirement .I provide recorded video for running the interface because I don't want to share my mapbox token account for this event ("pricey kasi ng usage ng api Nila"😿)



Conclusion

Therefore ,we conclude that sqlite can provide data to make a dashboard work well .However it cannot be use this on corporate world on solving business problem of every organization and have more limitation . But there is a goodside to learner like me to execute all learnings using this serverless module it can easy to use hold heavy data like Geospatial . On Data model ,dimensional model is fit to modern data technology for maximizing the business or organization needs . We can Easy delete or update the dimensional table as a description information for a fact table anytime we want .But normalize model is also important for ensuring the data integrity and logically related for every tables link on it .

References

1. *ArcGIS Hub. (n.d.). https://hub.arcgis.com/*

2. *Decomplexify. (2021, November 21). Learn Database Normalization - 1NF, 2NF,3NF,4NF,5NF[Video].YouTube. https://www.youtube.com/watch?v=GFQaEYEc8_8*

3. *Francheska-Vicente. (n.d.). GitHub - francheska-vicente/data101-ph-un-sgd: This project creates a Plotly dashboard of the Philippines' data on the Sustainable Development Goals (SDGs) of the United Nations. GitHub. https://github.com/francheska-vicente/data101-ph-un-sgd*

4. *PSA OpenSTAT > Home. (n.d.). https://openstat.psa.gov.ph/*

5. *ONSdigital. (n.d.). GitHub - ONSdigital/sdg-indicators: Development website for collecting and disseminating UK data for the Sustainable Development Goal global indicators. GitHub. https://github.com/ONSdigital/sdg-indicators*

6. *SQLite Python: Creating new Tables example. (2022, May 13). SQLite Tutorial. https://www.sqlitetutorial.net/sqlite-python/creating-tables/*