# Study Points Assignment on Databases SOFT 2024

## 1. Explore the data and formulate considerations about a hosting database.

**Hosting database:**

The considerations for a hosting database would focus on scalability, given the global scope of the data, and flexibility, to accommodate various data types and relationships.

We've explored the dataset and identified the key entities and relationships:

**Entities:**

- Organizations (Cities/Municipalities)
- Countries
- Emission Targets
- Reductions

**Relationships:**

- Organizations to Countries: Many-to-one. Many organizations can be located in one country.
- Emission Targets to Organizations: One-to-many relationship. One organization can have multiple emission targets over time.

## 2. Design and develop proper database structure of the requested type.

Our first draft of the database structure:

**Organizations Table:** Contains details about each city or municipality.
**Columns:** Organisation ID, Organisation Name, Account No, Country ID, City Short Name, C40 Membership, City Location.

**Countries Table:** Contains details about countries.
**Columns:** Country ID, Country Name, Country Location.

**Emission Targets Table:** Contains the emissions reduction targets for each organization.

**Columns:** Target ID, Organisation ID, Reporting Year, Sector, Target Boundary, Baseline Year, Baseline Emissions, Percentage Reduction Target, Target Date, Comment.

The finale database structure illustrated with database schemas:

| Countries | | |
|---|---|---|
| **CountryId** | CountryName | CountryLocation |
| int | varchar | varchar |

| Cities | | | |
|---|---|---|---|
| **CityId** | CountryId | CityName | CityLocation |
| int | int | varchar | varchar |

| Reductions | | | | | | |
|---|---|---|---|---|---|---|
| **ReductionId** | CityId | ReportingYear | BaselineYear | BaselineEmissions | PercentageReductionTarget | TargetDate |
| int | varchar | int | int | float | float | int |

| Emissions | | | |
|---|---|---|---|
| **EmissionId** | CityId | ReportingYear | TotalEmissions |
| int | varchar | int | float |

### 3. Ingest the data into the database, include pre-processing of it, if necessary.

We pre-processed the datasets and ingested the data into our new structured database with MSSQL.

See the repo: https://github.com/Pejomi/jupyter-notebook/tree/main/DB

### 4. Design and develop operations for maintenance of the database.

The following operations have been developed for maintaining the database:

- Regular data validation and cleanup. Removing duplicate or orphaned records.
- Database backups and recovery plans. Utilizing cron jobs for backing up.
- Performance monitoring and tuning. Identifying missing indexes.
- Security audits and updates. Regularly reviewing user permissions, and applying DBMS updates.
- Documentation and change management. Implementing a database versioning tool like Flyway, and maintaining documentation.

5. **Formulate ten relevant questions for extracting information from the database, design and develop database functionality for implementing the information extraction (for the relevance consult the instructor).**

What are the top 5 cities with the highest total emissions reported in a specific year?

Which cities have set the highest percentage reduction targets for their emissions by 2030?

How do C40 cities compare to non-C40 cities in terms of average total emissions reported?

What is the progress of a specific city towards its emissions reduction target compared to its baseline emissions?

Which countries have the most cities reporting emissions data?

What is the average emissions reduction target set by cities with a baseline year before 2010?

How many cities have already met or exceeded their emissions reduction targets?

What is the trend of total emissions reported by cities over the last 5 years?

Which cities have increased their emissions from the baseline year despite having reduction targets?

Are there regional differences in emissions reduction targets?

6. **Design and implement a model for scaling the database, considering ACID and/or CAP theorem rules.**

For scaling, a schema that can handle both structured and semi-structured data is ideal. NoSQL databases e.g. MongoDB offer flexibility and scalability, suitable for varied data types and volumes.

Data can be partitioned based on geographical regions, like country, or reporting years to distribute the load evenly across servers, enhancing query performance as the database grows.

7. **Validate and test all database operations.**

The following aspects have been validated and tested:

- Verification of City and Country Relationships
- Verification of City IDs in Emissions and Reductions
- Validation of Coordinate Formats
- Check for Reporting Year Ranges
- Check for Duplicate Country Names
- Verification that Required Fields Are Not Null

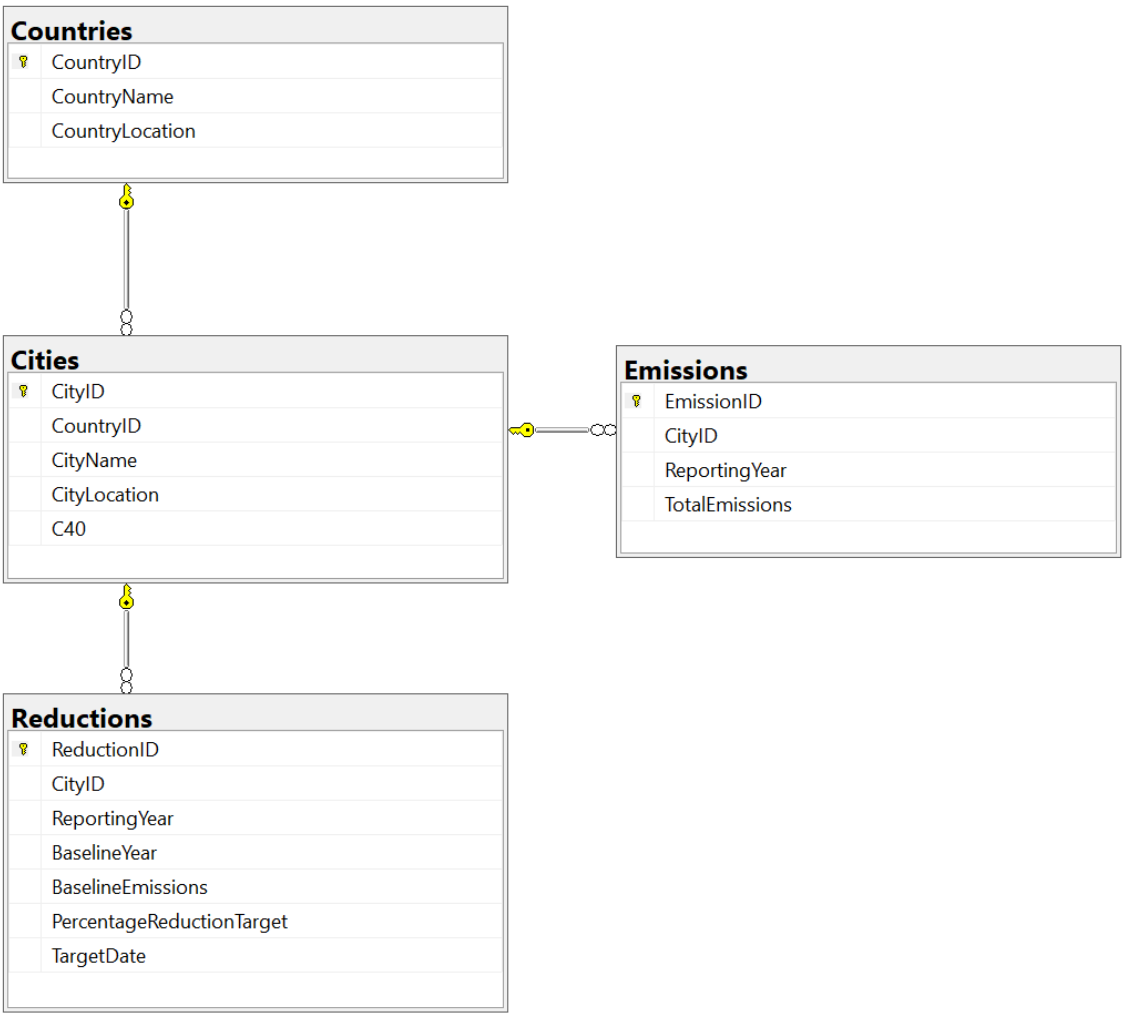
8. **Evaluate the database's performance and suggest measures for improving it.**

We considered the following measures for improving the database in the future:

- Data Normalization and Denormalization
- Indexing
- Partitioning and Sharding
- Optimize Query Statements
- Caching Frequently Accessed Data
- Scale Horizontally
- Use SSDs
- In-memory Databases

9. **Document your work, describing the product and the process. Apply graphical notation (diagrams), when it is possible.**

   We made a ER-diagram to present our work with structuring the new database:




## 10. Formulate conclusions and recommendations.

We spent too much time on preprocessing the data, because it was not compatible to be ingested into the database. We tried some different strategies for handling the logic. We started with Java, but quickly realized that Python was more suitable for handling the CSV files etc. We are using MsSql, but we also considered using MySql as an alternative. We came to the conclusion that MsSql was the better choice. We like the Database management system that Microsoft provides, because of features like fragmentation and performance reports.

We believe that we have a solution that can be used for the next assignment which is about MongoDB.