Report on Database Project

Introduction

Purpose: This report details the creation and management of a database designed to store and analyze data on city-wide greenhouse gas emissions, population, GDP, and related factors.

Data Source: The database was built using a merged dataset from multiple sources, from 2016-2023, providing comprehensive insights into emissions, population, GDP, and other urban metrics across various cities and countries.

Questions

1. **What is the purpose of the database created in this project?**
   The database stores and analyzes city-wide greenhouse gas emissions, population, GDP, and related factors.

2. **What data sources were used for the database?**
   Data was compiled from multiple sources spanning from 2016 to 2023, offering comprehensive insights into various urban metrics.

3. **How were significant variations in emissions and GDP across different cities identified?**
   Initial data exploration was conducted using SQL queries, revealing variations in emissions and GDP.

4. **What hosting considerations were made for the database?**
   A standard Supabase hosting plan was chosen, considering the dataset's size and complexity, with a provision for future scalability.

5. **How was data from different years merged and standardized for analysis?**
   A Python script was used to filter and merge data from five datasets, focusing on emissions, targets, and factors considered in assessments.

6. **What is the structure of the database schema, and how was it designed for efficiency?**
   A single table schema, merged_data, was created with fields like city, country, emission, sector, and more, with indexes on country and city.

7. **What were the top 5 cities in terms of emissions according to the database?**
   A specific SQL query (SELECT city, emission FROM city_emissions ORDER BY emission DESC LIMIT 5;) was used to identify these cities.

8. **What scaling strategy was employed for the database?**

Vertical scaling was used, with the consideration for partitioning the table by country if necessary.

9. **How was the functionality and integrity of the database confirmed?**
   Through testing all CRUD operations and performing data integrity checks post-ingestion.
   Examples of this are:

   Create (Inserting Data into the Database)

```
INSERT INTO city_emissions (city, country, emission, year)
VALUES ('New City', 'New Country', 10000, 2024);
```

   Read (Querying Data from the Database)

```
SELECT city, emission FROM city_emissions WHERE year = 2017;
```

   Update (Modifying Existing Data)

```
UPDATE city_emissions
SET emission = 11000
WHERE city = 'New City' AND year = 2024;
```

   Delete (Removing Data from the Database)

```
DELETE FROM city_emissions
WHERE city = 'New City' AND year = 2024;
```

   Aggregate Query (Summarizing Data)

```
SELECT country, SUM(emission) as total_emission
FROM city_emissions
WHERE year = 2023
GROUP BY country;
```

10. **What are the future recommendations for improving the database?**
    Future improvements could include adding more granular data, such as sector-wise emissions, and expanding geographic coverage.

Data Exploration and Hosting Considerations

Exploration Findings: Initial data exploration revealed significant variations in emissions and GDP across different cities. Certain countries showed higher average emissions, correlating with their population sizes.

SQL Query Example: SELECT country, AVG(city_population) as avg_population, AVG(emission) as avg_emission FROM all_dataset GROUP BY country;

Hosting Considerations: Given the dataset's moderate size and complexity, a standard Supabase hosting plan was deemed sufficient. Future scalability is an option, however the structure is easily copied over to other databases, like MongoDB.

```
import pandas as pd

# Load datasets
dataset_2016_targets =
pd.read_csv('2016_-_Cities_Emissions_Reduction_Targets_20240207.cs
v')
dataset_2016_ghg =
pd.read_csv('2016_-_Citywide_GHG_Emissions_20240207.csv')
dataset_2017_community =
pd.read_csv('2017_-_Cities_Community_Wide_Emissions.csv')
dataset_2017_targets =
pd.read_csv('2017_-_Cities_Emissions_Reduction_Targets_20240207.cs
v')
dataset_2023_risk =
pd.read_csv('2023_Cities_Climate_Risk_and_Vulnerability_Assessment
s_20240207.csv')

# Function to prepare and rename columns
def prepare_dataset(dataset, year, col_mappings):
    dataset = dataset.rename(columns=col_mappings)
    dataset['year'] = year
    return dataset

# Define column mappings for each dataset
col_mappings_2016_targets = {
    'City Name': 'city',
    'Country': 'country',
    'Baseline emissions (metric tonnes CO2e)': 'emission',
    'Target date': 'emission_target',
    'Baseline year':'baseline_year',
    'Percentage reduction target': 'target',
    'Sector':'sector'
}
col_mappings_2016_ghg = {
    'City Name': 'city',
```

```
    'Country': 'country',
    'Total CO2 emissions (metric tonnes CO2e)': 'emission',
    'City GDP': 'city_gdp',
    'Current Population': 'city_population',
    'Increase/Decrease from last year':'year_status',
    'Reporting Year':'year'
}


col_mappings_2017_community = {
    'City': 'city',
    'Country': 'country',
    'Total emissions (metric tonnes CO2e)': 'emission',
    'GDP': 'city_gdp',
    'Population': 'city_population',
    'Increase/Decrease from last year':'year_status',
    'Reporting Year':'year'
}

col_mappings_2017_targets = {
    'City': 'city',
    'Country': 'country',
    'Baseline emissions (metric tonnes CO2e)': 'emission',
    'Target date': 'emission_target',
    'Baseline year':'baseline_year',
    'Percentage reduction target': 'target',
    'Sector':'sector'
}

col_mappings_2023_risk = {
    'City': 'city',
    'Country/Area': 'country',
    'Year of publication or approval': 'year',
    'Factors considered in assessment': 'factors',
    'Population': 'city_population',
}

# Prepare each dataset
dataset_2016_targets_prepared =
prepare_dataset(dataset_2016_targets, 2016,
col_mappings_2016_targets)
dataset_2016_ghg_prepared = prepare_dataset(dataset_2016_ghg,
2016, col_mappings_2016_ghg)
dataset_2017_community_prepared =
prepare_dataset(dataset_2017_community, 2017,
col_mappings_2017_community)
```

```
dataset_2017_targets_prepared =
prepare_dataset(dataset_2017_targets, 2017,
col_mappings_2017_targets)
dataset_2023_risk_prepared = prepare_dataset(dataset_2023_risk,
2023, col_mappings_2023_risk)

# Merge all datasets
merged_dataset = pd.concat([
   dataset_2016_targets_prepared,
   dataset_2016_ghg_prepared,
   dataset_2017_community_prepared,
   dataset_2017_targets_prepared,
   dataset_2023_risk_prepared
])

# Drop duplicates and keep the latest record for each city per year
merged_dataset = merged_dataset.drop_duplicates(subset=['city',
'year'], keep='last')


final_selected_columns_dataset = merged_dataset[[
   'city', 'city_gdp', 'city_population', 'country', 'emission',
'emission_target', 'target', 'baseline_year', 'year',
'year_status', 'factors', 'sector'
]]

# Save the merged dataset
final_selected_columns_dataset.to_csv('merged_dataset.csv',
index=False)
```

We made a Python script to filter through the 5 datasets. Our main focus was to look at the emission of 2016-2017, what was their baseline from that year and what are the target year and in which sector. By doing this we can see what the current situation is for each city and country and see if there are any changes positively or negatively. Lastly we included a 2023 report to see what factors were considered in assessments with the current baseline co2 emission.

Database Structure Design

Schema Design: The database schema was designed for efficiency and scalability, with a single table encompassing all necessary fields.

SQL for Table Creation:

```
CREATE TABLE merged_data (
   city TEXT,
   city_gdp NUMERIC,
```

```
    city_population INTEGER,
    country TEXT,
    emission NUMERIC,
    emission_target NUMERIC,
    sector TEXT
);
```

Indexing: Indexes were created on the country and city columns to optimize query performance.

Image Insertion: [ER Diagram or Schema Diagram]

Data Ingestion and Pre-processing

Pre-processing Steps: The dataset underwent cleaning for missing values and standardizing data formats, particularly for numeric fields like GDP and population.

Ingestion Process: Data was ingested into Supabase via the CSV upload feature, ensuring alignment with the defined schema.

Test

With python we have tested our database, to make sure the correct columns have been created and that the data we want has been successful imported.

```
import unittest
import pandas as pd

class TestData set(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        # Load the dataset only once for all tests
        cls.dataset = pd.read_csv('/path/to/your/merged_dataset.csv')

    def test_column_presence(self):
        """Test if all required columns are present in the dataset."""
        required_columns = ['city', 'city_gdp', 'city_population', 'country',
                            'emission', 'emission_target', 'sector', 'year']
        for column in required_columns:
            self.assertIn(column, self.dataset.columns)

    def test_no_null_values(self):
        """Test if there are any null values in critical columns."""
```

```
        critical_columns = ['city', 'country', 'emission', 'year']
        for column in critical_columns:
            self.assertFalse(self.dataset[column].isnull().any())

    def test_emission_values(self):
        """Test if emission values are within an expected range."""
        self.assertTrue((self.dataset['emission'] >= 0).all())

    def test_year_values(self):
        """Test if year values are within an expected range."""
        self.assertTrue(self.dataset['year'].between(2016, 2023).all())

    # More testing will be added later

if __name__ == '__main__':
    unittest.main()
```

Information Extraction Queries

Query Examples and Results:

Query for Top 5 Emission Cities:

```
select
 city,
 country,
 baseline_year,
 emission
from
 city_emissions
order by
 emission desc
limit
 5;
```

Result: [List top 5 cities and their emissions]

Results (5) Chart

city	country	baseline_year	emission
"Kaohsiung"	"Taiwan"	"2005"	"67326998.0"
"Tokyo"	"Japan"	"2000"	"62100000.0"
"New York City"	"USA"	"2005"	"61060000.0"
"London "	"United Kingdom"	"1990"	"45050000.0"
"Glasgow"	"United Kingdom"	"2006"	"39873000.0"

Scaling Model

Scaling Strategy: The current setup employs vertical scaling, with considerations for partitioning the table by country if the dataset grows significantly.

Validation and Testing

Testing Procedures: All CRUD operations were tested for functionality. Data integrity checks were performed post-ingestion.

Validation Results: The tests confirmed that all database functionalities were operating as expected.

Performance Evaluation

Performance Metrics: Initial metrics indicated efficient query processing. No significant delays or resource bottlenecks were observed.

Optimization Steps: N/A

Image Insertion: [Screenshots of performance metrics]

Documentation

Overview: Comprehensive documentation was created, detailing the database schema, ETL processes, query structures, and maintenance procedures.

Conclusions and Recommendations

Conclusions: The database effectively consolidates and manages the emissions data, allowing for insightful analyses across different geographic and economic dimensions.

Recommendations: Future improvements could include integrating more granular data, such as sector-wise emissions, and expanding the dataset to cover more geographical regions.