TradeSphere Analytics Final Report

1. Introduction:

Our project, TradeSphere, is designed to address the critical need for accessible, comprehensive, and comparative analysis of international trade and demographic data. The application offers users the ability to visualize and analyze trade patterns, economic indicators, and health metrics across a multitude of countries. Our team, composed of Lisa Garcia, Zung-Ru Lin, Hanling Su, and Irene Tang, found motivation in the potential of data to inform policies and business strategies that foster global socio-economic development.

TradeSphere's user-friendly platform can serve researchers, social scientists, policymakers, and the general public in understanding and acting on international data. In the pursuit of truth, we leverage the precision of numbers to uncover realities across the globe. We aim to provide a comprehensive picture of country-specific conditions, empowering users to drive data-informed decisions, facilitating the identification of social issues, economic opportunities, and public health conditions. This platform serves as a nexus where demographics, trade data, and health statistics converge to inform policy-making, economic strategy, and social initiatives.

The main goal of the web platform is to highlight key aspects of our database with which the user can dynamically interact, including selecting countries, time intervals, and the type of exchanged goods via sliders, drop-down menus, and pagination. The application stands as a resource for those aiming to instigate social betterment and economic justice through rigorous analysis and strategic planning.

2. Architecture

At the heart of our platform is a robust MySQL database system, chosen for its prowess in handling intricate data structures and providing reliable performance. Our choice of AWS as a hosting service ensures that the platform remains responsive and secure, capable of scaling to meet the demands of our user base.

The technical architecture is designed with precision to ensure that every query yields the most accurate and relevant information. Our backend processes are optimized for efficiency, translating complex data relationships into usable formats. The APIs that connect this data core to the front end are the lifelines of the application, facilitating a seamless and dynamic user experience.

The web application uses the created MySQL database to dynamically change the displayed data based on the user input. In order to facilitate the communication between the user and the database a client and server side is developed using open-source JavaScript-based React, Node and Express.js as frameworks.

The server side implements routers to set the application's interaction with the user input to an endpoint. This includes the definition of the request method, the path on the server, and which function is executed. Every route consists of request and response parameters, which determine the user input and the data output to be expected as well as a SQL query that directly retrieves data from the database. The client side establishes the user interface to implement dynamic views in a web browser, which include requests to the server and the corresponding response which is displayed. Among the multiple features React offers is the integration of HTML, CSS, and JSX to design the web application.

3. Data

Data source #1: International trading reports, hosted by Trading Economics

- https://tradingeconomics.com
- Trading Economics delivers reliable data across 196 nations, encompassing historical metrics for a myriad of economic indicators - from exchange rates and stock market indices to government bonds and commodity valuations. The dataset features a wide range of trade categories, spanning from high-value goods like aircraft and spacecraft to essential materials such as aluminum, copper, and zinc. It also encompasses products like cheese and curd, offering a diverse overview of trade dynamics. Detailed date information accompanies each transaction, facilitating in-depth analysis of international trade patterns between the United States and other countries across various sectors.

Data source #2: Country-level information from various sources, hosted by Our World In Data

- Land area by country: https://ourworldindata.org/grapher/land-area-km
- Countries and their continents: https://statisticstimes.com/geography/countries-by-continents.php
- Population data by country: https://ourworldindata.org/population-growth
- Urbanization development metrics: https://ourworldindata.org/urbanization
- Access to sanitized water and facilities: https://ourworldindata.org/clean-water-sanitation

Each dataset in our system is sourced from reputable databases that provide comprehensive trading data, demographic statistics, and health indices. After downloading the raw data, we cleaned and wrangled it into a set of new datasets optimal for loading into the database. A breakdown of the tables and number of instances kept are noted in Appendix A.

We use this data to present users with insights into global trends and country-specific conditions. Summary statistics like average trade volumes, population growth rates, and health indicators are computed to give users a snapshot of the data. This data forms the backbone of our application, enabling the rich, multifaceted analyses that our users depend on.

The practical utility of our application lies in its capacity to provide users with the data needed to inform policies, guide social programs, and foster economic growth. It is a tool for those who seek to understand the present and shape the future through informed action.

4. Database

The database at the core of our application is architected with a focus on both adherence to normalization standards and practical functionality. Most of the database tables are structured in accordance with the Boyce-Codd Normal Form (BCNF) or the Third Normal Form (3NF). This commitment to normalization ensures that our tables are free from undesirable dependencies, which can lead to anomalies in data storage. By eliminating extraneous columns and keeping only essential superkeys with their related non-key attributes, we've optimized the database structure to prevent data duplication and maintain integrity, with each determinant being a candidate key.

A deliberate choice was made in the design of our UnitConversion table to prioritize user efficiency and operational simplicity. In this table, FrequencyRate and ExchangeRate columns directly determine the ConversionRate. Although this design technically contravenes the strict interpretation of 3NF, it significantly enhances the table's usability. This deviation from normalization is a calculated trade-off that provides immediate, on-the-fly conversions and rate updates without the need for additional computation or table joins, which can be critical for real-time data processing and user experience.

This pragmatic approach to our database architecture balances the theoretical ideals of normalization with the practical demands of a dynamic and responsive application. Our primary objective is to maintain a database that not only upholds data integrity but also supports efficient and user-friendly interactions, ensuring that our application remains robust, agile, and aligned with the needs of our users.

The database consists of over 15 tables most of which have several thousands of instances. The table with the highest number of instances is Prices with around 312,000 and the lowest Country Demographics with approximately 200. The total list of tables and their corresponding instances can be found in Appendix A. The entity-relationship diagram, found in Appendix B, conveys the columns for each table and the connections between the tables. Country-related tables use Country as their primary key while the US Trading Data table uses Symbol.

5. Web App description

The web application consists of three sections: a homepage, a trading page, and a collection of country-info pages.

The Home page introduces the reader to the application and database and conveys the overall purpose and usability. Its main function is serving as a gateway to the other pages, including where to find relevant information using links and showcasing some of the available data using a static table and graph. The homepage uses three available routes to return the authors names, all available export values per trading category as a table, and unimproved drinking water by continent. .

The Trades page focuses on international trade, with the United States. Its main function is to allow a diverse and dynamic user interaction and introduce key aspects of the available trading data. As one of the main audience targets for the web application are researchers and other data literate individuals, the trading page includes a selection of variables allowing the user to change and filter data and display them in the form of tables to be exported for further exploration. A map feature portrays how the available data can be communicated creatively and interactively. The trading page uses four routes, detailed in the API section in this report, which answer questions regarding the largest trading partners with the United States by different trade categories and types and how the trading behavior changes regionally.

The Countries pages examine frequently studied socio-economic questions and user interaction with relevant variables. The idea of the country page is to facilitate said interaction by dedicating separate subpages to different research areas, including regional living and labor condition differences and temperature and wage changes. In total there are five subpages each of which use one route. The pages use a combination of sliders, a dropdown menu, tables, and a graph to diversify and enhance the user experience while maintaining high performance and data usability.

Together, the web application provides a visual and interactive platform for investigating international trade activity—especially that which pertains to the United States economy.

6. API Specification

The web application uses a total of sixteen routes. Described below is a short overview of their respective request and response parameters and functionality.

Route 1 receives the authors names and returns them as a string, which is displayed on our homepage. The remaining routes all incorporate SQL queries that directly communicate with the database

Routes 2 and 3 both require the trading type and category as string-type request parameters and return an array with all country trading data with the United States and the largest trading partner respectively. Both routes are implemented in the trading page and are used to create a map with the top five United States trading partners and a corresponding table with more comprehensive information, including the trade year.

Routes 4 and 5 are also incorporated in the trading page, where the former requires the type and country as string-type user input returning an array with the information on the largest trading category and corresponding value and the latter requiring the type as string-type user input and returning the total trading volume by continent as an array. Both routes display the selected information as tables.

Route 6 receives the page and page size as request parameters and returns an array with information on the United States' total export trading volume for specific categories, which is displayed on the homepage.

Route 7 returns population-relevant information by country and region, including the proportion of the urban population, drinking water, sanitation, and infants, in the form of an array which is displayed as a table. The request parameters include the page, page size and numeric values within a range. For urban, sanitation, and sanitation proportions the user selects a range from 0 to 100 and for infant proportions from 0 to 5 in the form of sliders.

Route 8 returns the maximum number of unimproved drinking water by continent as an array which receives the page size and page as user input and is used to display a barchart on the homepage. The remaining routes are all dedicated to the countries pages.

Route 9 receives two countries as user inputs, which are populated as dropdown menu options using Route 13, and returns an array with the differences of wages between the countries over time.

Routes 10, 11, and 12 are structured similarly and require page and page size as request parameters as well as two additional variables as user inputs implemented as sliders. All of them return the data in the form of an array which is used for a table in the application. Route 10 reports on the average yearly wage growth in percentages using the lowest and highest growth as user input. Route 11 returns the unemployment rate filtering for low and high unemployment and route 12 temperature changes filtering for low and high changes.

Routes 13 and 14, serve as dropdown menus in the trading page. Both routes do not have request parameters and return all distinct countries, the trading partners, and categories in the form of arrays, which are used to populate the country and category-related dropdown menus that serve as user input for the trading-related tables and map.

Routes 15 and 16 serve as helper routes that pull distinct country and year values from our databases. In conjunction with route 9 and route 16, which does not have a request parameter but returns the distinct countries of the labor table, route 15 populates the distinct years and months to populate the x-axis of the plot. Route 9 is used to display both a table and line plot of the data.

All sixteen routes are used together to build the tables and visualizations on the client side.

7. Queries

Several complex queries are utilized within our application, such as:

(1) Show the country that has the highest proportion of population having access only to unimproved drinking water for each continent

```
With MaxUM AS (SELECT C.Continent, MAX(H.Unimproved_Drinking_Water_Access) AS
MaxUnimprovedWater
FROM HealthData H JOIN CountryInfo C on H.Country = C.CountryName
where C.Continent IS NOT NULL
GROUP BY C.Continent)

SELECT  C.Continent, H.Country, MUM.MaxUnimprovedWater
FROM HealthData H JOIN CountryInfo C on H.Country = C.CountryName JOIN MaxUM
MUM on C.Continent = MUM.Continent AND H.Unimproved_Drinking_Water_Access =
MUM.MaxUnimprovedWater
```

(2) Compare two country wages and show them in a time-series plot

```
WITH WageTable AS (
SELECT B.Country, B.Year, B.Month, B.Value*U.ConversionRate, I.Unit,
U.UnitGroup, U.ExchangeRate, U.FrequencyRate, U.ConversionRate
FROM Labour B
JOIN IndexTable I ON B.Category = I.Category AND B.Country = I.Country
JOIN UnitConversion U ON I.Unit = U.Unit
WHERE B.Category = 'Wages'
)

SELECT *
FROM WageTable
WHERE Country = {$country1} OR Country = {$country2}
ORDER BY Year, Month ASC;
```

(3) Rank the highest unemployment rate for all countries and show the year month to provide more insights associated with the ground truth.

```
# year-month with highest unemployment rate by country
WITH RankedLabour AS (
 SELECT Country,Year,Month,Value, RANK() OVER (PARTITION BY Country ORDER BY
Value DESC) AS ValueRank
 FROM Labour
 WHERE Category = 'Unemployment Rate'
)

SELECT Country, Year, Month, Value AS HighestUnemploymentRate
FROM RankedLabour
WHERE ValueRank = 1
ORDER BY HighestUnemploymentRate DESC, Country, Year, Month;
```

(4) Calculate the average yearly growth rate for wages in a comparative time-series format.

```
WITH WageManuTable AS (
SELECT B.Country, B.Year, B.Month, B.Value*U.ConversionRate, I.Unit,
U.UnitGroup, U.ExchangeRate, U.FrequencyRate, U.ConversionRate
FROM Labour B
JOIN IndexTable I ON B.Category = I.Category AND B.Country = I.Country
JOIN UnitConversion U ON I.Unit = U.Unit
WHERE B.Category = 'Wages in Manufacturing'
)
SELECT *
FROM WageManuTable;

# Average wage growth rank (economic growth + currency value fluctuation)
WITH WageTable AS (
SELECT B.Country, B.Year, B.Month, B.Value*U.ConversionRate AS ConvertedValue,
I.Unit, U.UnitGroup, U.ExchangeRate, U.FrequencyRate, U.ConversionRate
FROM Labour B
JOIN IndexTable I ON B.Category = I.Category AND B.Country = I.Country
JOIN UnitConversion U ON I.Unit = U.Unit
WHERE B.Category = 'Wages'
), FirstYear AS (
   SELECT Country, MIN(Year) as FirstYear
   FROM WageTable
   GROUP BY Country
), LastYear AS (
   SELECT Country, MAX(Year) as LastYear
   FROM WageTable
   GROUP BY Country
), FirstValue AS (
   SELECT wt.Country, wt.ConvertedValue as FirstValue
   FROM WageTable wt
   INNER JOIN FirstYear fy ON wt.Country = fy.Country AND wt.Year =
fy.FirstYear
   WHERE Month = (SELECT MIN(Month) FROM WageTable WHERE Country = wt.Country
AND Year = wt.Year)
), LastValue AS (
   SELECT wt.Country, wt.ConvertedValue as LastValue
   FROM WageTable wt
   INNER JOIN LastYear ly ON wt.Country = ly.Country AND wt.Year = ly.LastYear
   WHERE Month = (SELECT MAX(Month) FROM WageTable WHERE Country = wt.Country
AND Year = wt.Year)
), ProportionalIncrease AS (
   SELECT
       fv.Country,
       (lv.LastValue - fv.FirstValue) / fv.FirstValue as PropIncrease,
       ly.LastYear - fy.FirstYear as YearDifference
   FROM FirstValue fv
   INNER JOIN LastValue lv ON fv.Country = lv.Country
   INNER JOIN FirstYear fy ON fv.Country = fy.Country
   INNER JOIN LastYear ly ON fv.Country = ly.Country
)
SELECT
   Country,
      CASE WHEN YearDifference = 0 THEN NULL ELSE PropIncrease /
YearDifference*100 END AS `AvgYearlyIncrease(%)`
FROM ProportionalIncrease
ORDER BY `AvgYearlyIncrease(%)` DESC;
```

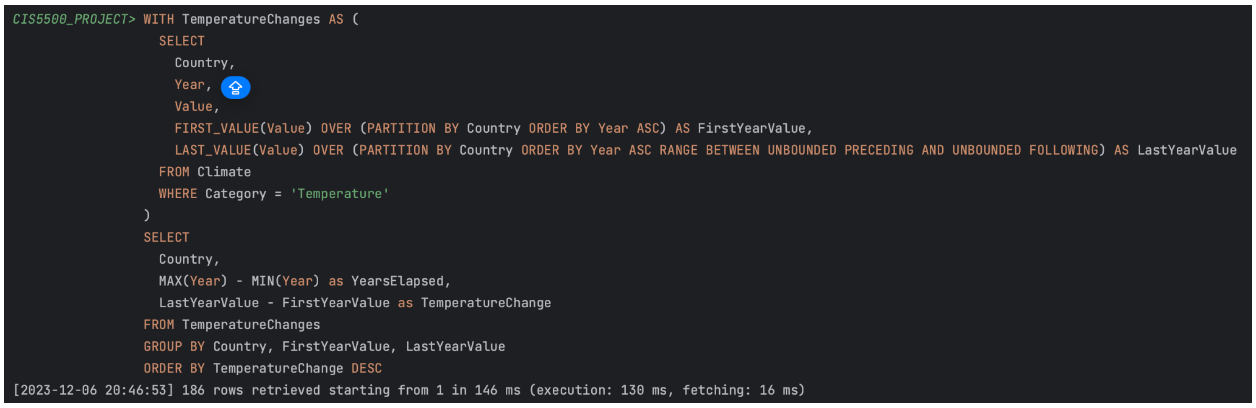
(5) Calculate the average yearly growth rate for temperatures in a comparative time-series format.

```
# Global effect influence
WITH FirstLastYears AS (
   SELECT
       Country,
       MIN(Year) as FirstYear,
       MAX(Year) as LastYear
   FROM Climate
   WHERE Category = 'Temperature'
   GROUP BY Country
), FirstYearValue AS (
   SELECT
       c.Country,
       c.Year as Year,
       c.Value as FirstYearValue
   FROM Climate c
   INNER JOIN FirstLastYears fl ON c.Country = fl.Country AND c.Year =
fl.FirstYear
   WHERE Category = 'Temperature'
), LastYearValue AS (
   SELECT
       c.Country,
       c.Year as Year,
       c.Value as LastYearValue
   FROM Climate c
   INNER JOIN FirstLastYears fl ON c.Country = fl.Country AND c.Year =
fl.LastYear
   WHERE c.Category = 'Temperature'
)
SELECT
   f.Country,
   l.Year - f.Year as YearsElapsed,
   l.LastYearValue - f.FirstYearValue as TemperatureChange
FROM FirstYearValue f
INNER JOIN LastYearValue l ON f.Country = l.Country
ORDER BY TemperatureChange DESC;
```

These queries are fundamental to the application's core functionalities, providing the necessary data to fuel the analyses and visualizations presented to the user.

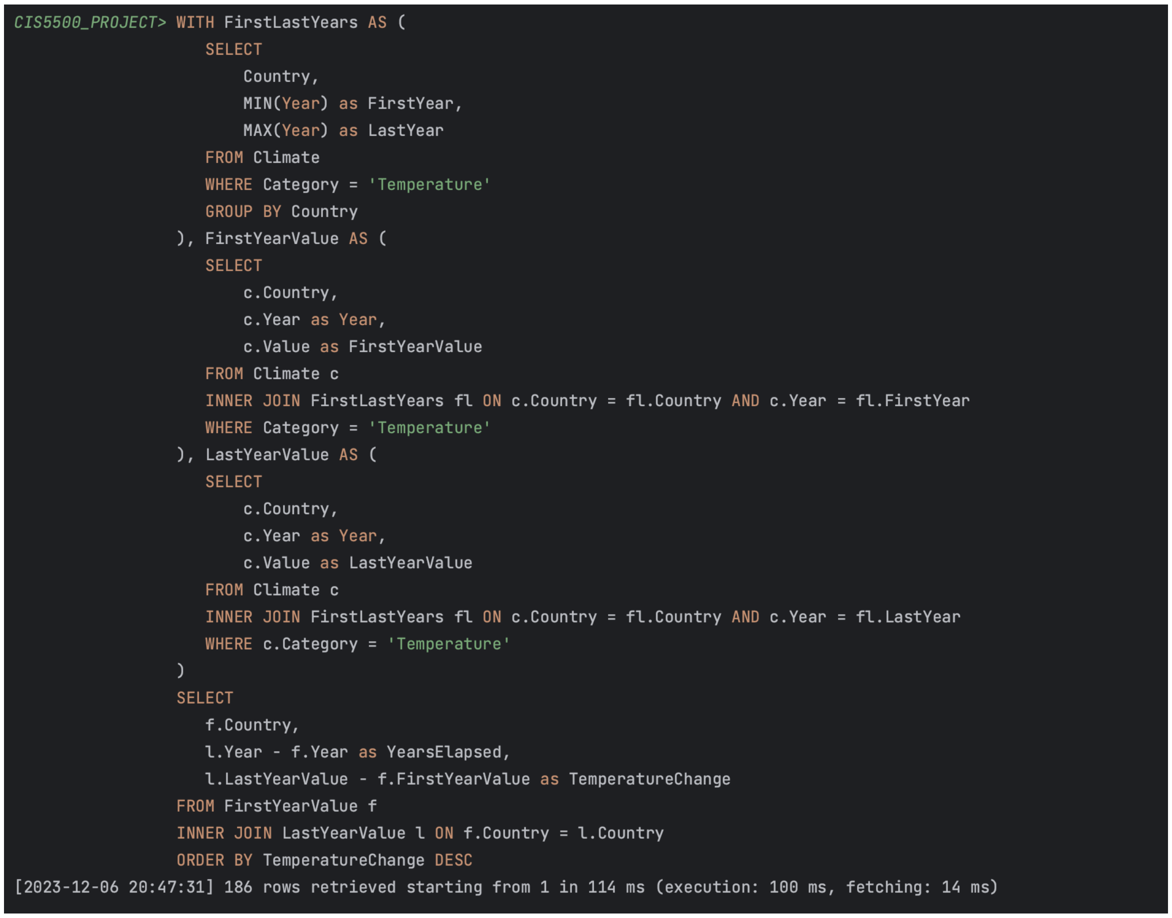
8. Performance evaluation

Query 1-1: Temperature Changes (slower but shorter):



```
CIS5500_PROJECT> WITH TemperatureChanges AS (
                   SELECT
                     Country,
                     Year,
                     Value,
                     FIRST_VALUE(Value) OVER (PARTITION BY Country ORDER BY Year ASC) AS FirstYearValue,
                     LAST_VALUE(Value) OVER (PARTITION BY Country ORDER BY Year ASC RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS LastYearValue
                   FROM Climate
                   WHERE Category = 'Temperature'
                 )
                 SELECT
                   Country,
                   MAX(Year) - MIN(Year) as YearsElapsed,
                   LastYearValue - FirstYearValue as TemperatureChange
                 FROM TemperatureChanges
                 GROUP BY Country, FirstYearValue, LastYearValue
                 ORDER BY TemperatureChange DESC
[2023-12-06 20:46:53] 186 rows retrieved starting from 1 in 146 ms (execution: 130 ms, fetching: 16 ms)
```

Query 1-2: Temperature Changes (faster but lengthy):



```
CIS5500_PROJECT> WITH FirstLastYears AS (
                    SELECT
                        Country,
                        MIN(Year) as FirstYear,
                        MAX(Year) as LastYear
                    FROM Climate
                    WHERE Category = 'Temperature'
                    GROUP BY Country
                 ), FirstYearValue AS (
                    SELECT
                        c.Country,
                        c.Year as Year,
                        c.Value as FirstYearValue
                    FROM Climate c
                    INNER JOIN FirstLastYears fl ON c.Country = fl.Country AND c.Year = fl.FirstYear
                    WHERE Category = 'Temperature'
                 ), LastYearValue AS (
                    SELECT
                        c.Country,
                        c.Year as Year,
                        c.Value as LastYearValue
                    FROM Climate c
                    INNER JOIN FirstLastYears fl ON c.Country = fl.Country AND c.Year = fl.LastYear
                    WHERE c.Category = 'Temperature'
                 )
                 SELECT
                    f.Country,
                    l.Year - f.Year as YearsElapsed,
                    l.LastYearValue - f.FirstYearValue as TemperatureChange
                 FROM FirstYearValue f
                 INNER JOIN LastYearValue l ON f.Country = l.Country
                 ORDER BY TemperatureChange DESC
[2023-12-06 20:47:31] 186 rows retrieved starting from 1 in 114 ms (execution: 100 ms, fetching: 14 ms)
```

Query 1-1 uses window functions (FIRST_VALUE and LAST_VALUE). While concise, it processes every row in each country's partition, which can be computationally heavy, especially with large datasets. In contrast, query 1-2 employs subqueries to first identify specific years and then joins these results with the main dataset. This targeted approach reduces computational load since it avoids scanning through all rows for each country.

Performance Analysis:

- Query 1-1 took 146 milliseconds (130 ms execution, 16 ms fetching).
- Query 1-2 was faster at 114 milliseconds (100 ms execution, 14 ms fetching).

Why the Second Query is More Efficient:

(1) It's more targeted, identifying key years first and then fetching corresponding values, which streamlines processing.
(2) It potentially benefits more from database optimizations like indexing, as it involves focused operations (aggregation and joins).
(3) In contrast, window functions in the first query, although powerful, require processing each row within partitions, leading to longer execution times.

The same strategy, using subqueries to reduce the computational load, was applied to most queries implemented on the country subpages. No latency issues were observed in the web application development once the queries were integrated. Therefore, no additional optimization was considered necessary. For future data integration, such as expanding the size of the tables or adding new tables or variables, the queries should be re-evaluated and adapted where latency is impacted.

According to the available web application performance evaluation using developer tools, the most CPU intensive pages are the trades and countries wages page, attributed to scripting. For both pages rendering the dropdown selection and returning the map and plot have the highest usage and time. Testing the population of wages for different countries took about 1322ms for the animation and 533ms for the function call. The total scripting time for the map feature took about 1670 ms in scripting and 325ms in rendering whereas the country wage comparison plot required about 1644 ms in scripting and 340 in rendering. In comparison, the most slider-heavy page, living conditions, only took about 676ms in scripting and 324ms in rendering. The least CPU intensive page is the homepage due to the use of a static table and graph. Even filtering for specific countries and only selecting some columns only results in 240ms in scripting and 189ms in rendering. Despite evidence that some of our features have a much higher scripting time than others, the user experience is not impacted. We are therefore confident that additional optimization strategies are not necessary at this time.

9. Technical Challenges

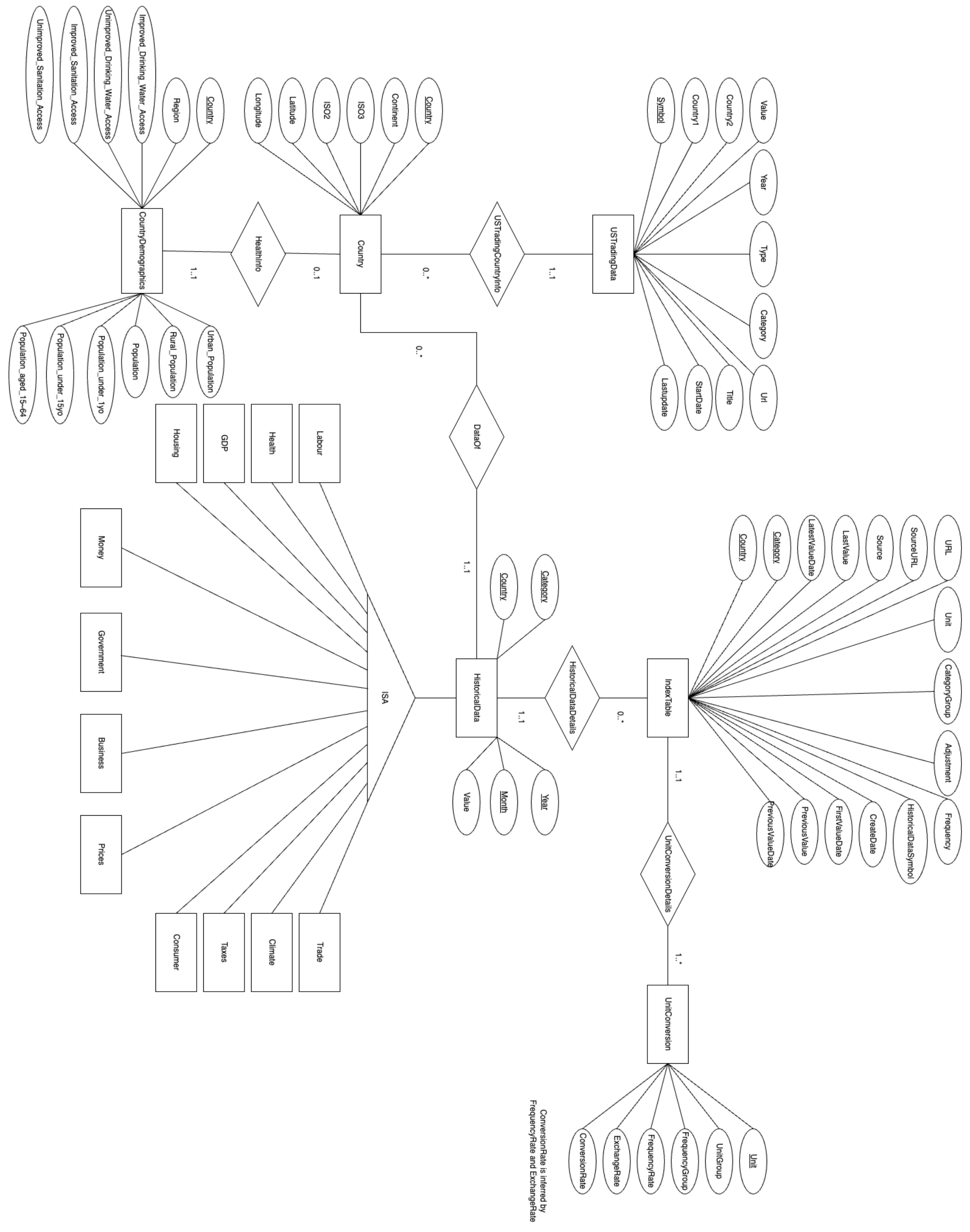
When collecting data through the Trading Economics API using Python, we faced significant technical challenges primarily due to the enormous size of the dataset. Merging and organizing this vast amount of data into a coherent and manageable structure proved to be a complex task, requiring careful consideration of efficient data handling and storage techniques. Throughout our database and server deployment, we encountered technical hurdles in creating and deploying DB tables on AWS. Despite data preprocessing and an organized plan to achieve unit conversions, challenges persisted. Establishing country names as foreign keys revealed discrepancies between tables, disrupting key constraints. Similar issues surfaced with continent data, necessitating manual adjustments to ensure accuracy. Moreover, dependencies between key constraints posed challenges when editing data. This process proved more intricate than anticipated, emphasizing the crucial role of meticulous data editing for maintaining precise key constraints and database integrity. These challenges underscored the significance of accurate data and highlighted the importance of implementing key constraints for seamless database functionality, offering valuable insights into data integrity practices.

Other technical challenges included adequate data integration on the client side. Even though we had a clear understanding of the look and feel of our web application and developed our SQL queries early on, we had to make frequent changes to our request and response parameters when navigating and adjusting web application features and the user interaction with them. This included the type of output to be returned, object or array, and the type of user input necessary, string, number range, or fixed values. We overcame these challenges by maintaining adaptability when debugging error messages and adjusting the client or server side when necessary.

10. Appendix A: Tables and Instances

Table Name	**Number of Instances**
Business	175,086
Climate	6,154
Consumer	87,160
CountryDemographics	198
CountryInfo	257
GDP	95,551
Government	72,477
Health	2,278
Housing	43,172
IndexTable	13,907
Labour	116,887
Money	86,065
Prices	312,300
Taxes	13,315
Trade	205,285
USTradingData	232,412
UnitConversion	396

11. Appendix B: Entity Relationship (ER) Diagram


Unimproved_Sanitation_Access
Improved_Sanitation_Access
Unimproved_Drinking_Water_Access
Improved_Drinking_Water_Access
Region
Country
CountryDemographics
Population_aged_15-64
Population_under_15yo
Population_under_1yo
Population
Rural_Population
Urban_Population
1..1
HealthInfo
0..1
Longitude
Latitude
ISO2
ISO3
Continent
Country
Country
0..*
USTradingCountryInfo
1..1
USTradingData
Symbol
Country1
Country2
Value
Year
Type
Category
Url
Title
StartDate
Lastupdate
0..*
DataOf
1..1
Housing
GDP
Health
Labour
Money
Government
Business
Prices
ISA
Consumer
Taxes
Climate
Trade
HistoricalData
Country
Category
Value
Month
Year
1..1
HistoricalDataDetails
0..*
IndexTable
Country
Category
LatestValueDate
LastValue
Source
SourceURL
URL
Unit
CategoryGroup
Adjustment
Frequency
HistoricalDataSymbol
CreateDate
FirstValueDate
PreviousValue
PreviousValueDate
1..1
UnitConversionDetails
1..*
UnitConversion
ConversionRate
ExchangeRate
FrequencyRate
FrequencyGroup
UnitGroup
Unit
ConversionRate is inferred by FrequencyRate and ExchangeRate