

DADT Midterm Report

[Introduction](#)

[Data Source](#)

[Project Motivation](#)

[Interest for the Dataset](#)

[E/R Diagram](#)

[Normalisation](#)

[Database Structure](#)

[Initial Commands \(CREATE, INSERT INTO, LOAD DATA INFILE\)](#)

[Query for Questions](#)

[Web Application](#)

[Conclusion](#)

Introduction

This project focuses on analyzing global mortality data, specifically examining mortality trends by region, age group, gender, and cause-specific deaths. The data is sourced from a health statistics database, which includes mortality counts, death rates, and other related metrics across different countries and regions. The project implements SQL queries to extract and process the data, which is then visualized using Chart.js in a web application.

Data Source

The WHO Mortality Database on noncommunicable diseases (NCDs) provides detailed statistics on deaths caused by NCDs across various regions and countries. The data is extensive, including information on diseases like cardiovascular diseases and cancers, which are the leading causes of mortality. It also covers risk factors such as tobacco use, high blood pressure, and insufficient physical activity.

The database supports interactive tools to explore cause-specific mortality trends, i.e. line and bar charts for trends, bubble plot for area-related comparisons, and a world map to view the overall.

WHO Mortality Database - Noncommunicable Diseases

"The data used in this project is sourced from the World Health Organization (WHO) Mortality Database. The WHO permits the review, reproduction, or translation of the data for research or private study purposes, but not for commercial use. Any use of this data must include appropriate acknowledgment of WHO as the source. For substantial use or

translation, prior written permission is required. The information provided is not warranted to be complete or error-free" ([WHO, 2021](#)).

Project Motivation

This dataset is of particular interest due to its comprehensive collection of global mortality data, particularly in relation to non-communicable diseases. As my project aims to analyze mortality trends across different regions, age groups, and gender, the rich and granular information provided by the WHO's Mortality Database aligns well with my goal of uncovering patterns and insights into health disparities and trends. The dataset's depth, including factors such as cause-specific mortality and death rates across multiple years, enables a thorough analysis, allowing for a data-driven approach to understanding health outcomes on a global scale. Additionally, the dataset's focus on non-communicable diseases aligns with the increasing importance of studying long-term health trends, making it invaluable for both academic research and public health analysis.

Interest for the Dataset

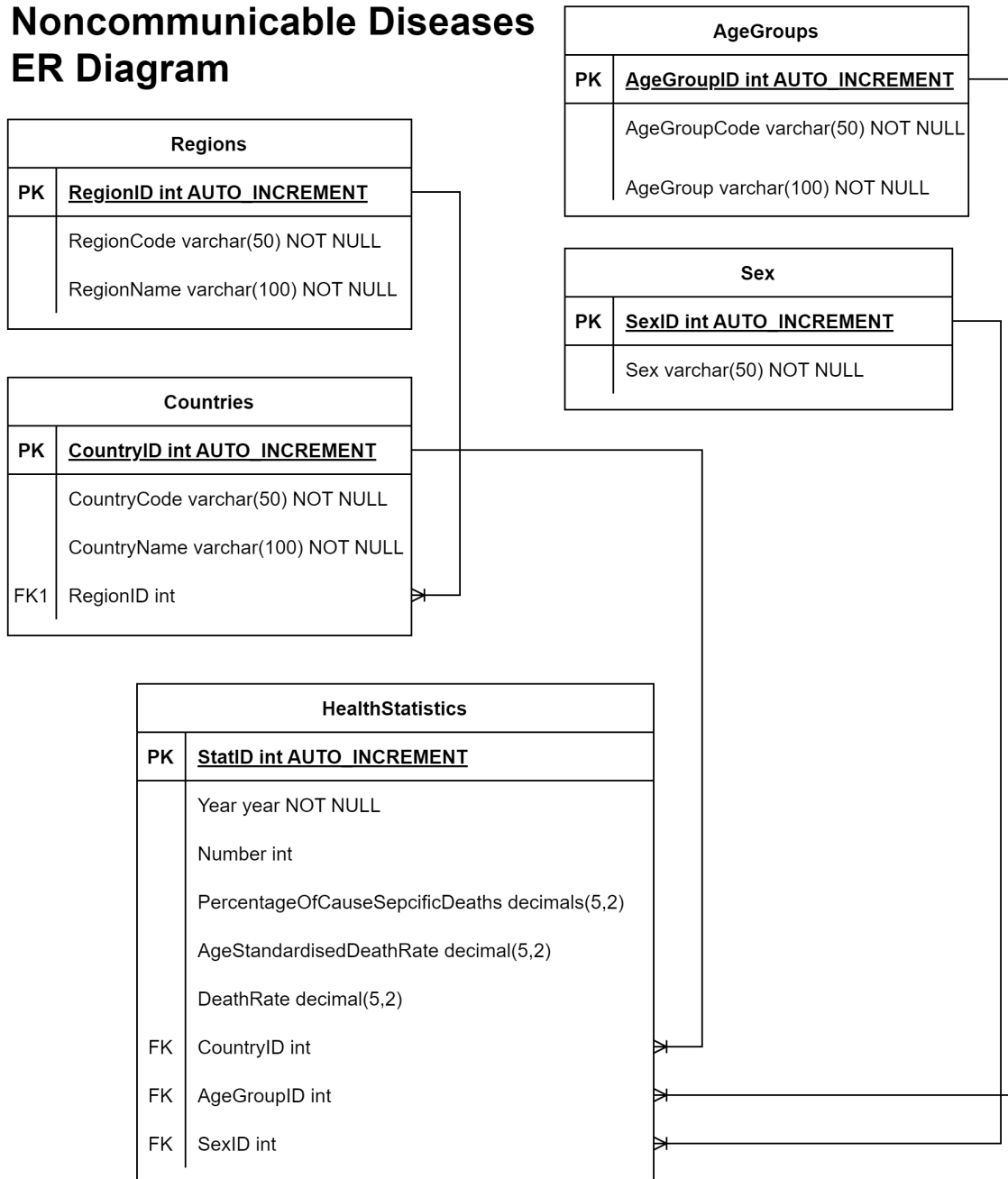
This dataset is of significant interest because it provides global mortality statistics across regions, age groups, and gender, which can help in identifying health trends, disparities, and areas requiring attention. It allows for in-depth analysis of mortality patterns in different populations and over time. Some questions that can be asked from this dataset include:

1. What are the average mortality rates across different regions?
2. Which age groups have the highest death rates in high-data regions?
3. Are there gender-based differences in mortality rates in selected regions?
4. How have mortality rates changed over time in specific countries?

A database application can efficiently query, filter, and aggregate data to provide answers to these questions, assisting in health policy decision-making and identifying regions or demographics that need targeted interventions.

E/R Diagram

WHO Morality Noncommunicable Diseases ER Diagram



Normalisation

Definition used from the lecture 3.104 Normalisation and the normal forms II.

1. 1NF: The table is a relation. All of its attributes are scalar values.
 - The table is already in scalar value.
2. 2NF: Table is in 1NF. Every non-key attribute is irreducibly dependent on the primary key.
 - **Region Name** is dependent on **Region Code**.
 - **Country Name** is dependent on **Country Code**.
 - **Region Code** is indirectly related to **Country Code** through **Country Name**.
3. 3NF: Table is 2NF. Every non-key attribute is non-transitively (directly) dependent on the primary key
 - In the **HealthStatistics** table, all non-key attributes like **Number** , **PercentageOfCauseSpecificDeaths** , **AgeStandardizedDeathRate** , and **DeathRate** depend only on the primary key (**StatID**), not on other non-key attributes.
 - By separating the country, age group, region, and sex information into their own tables, this can avoid redundancy and transitive dependencies.
4. BCNF (Boyce-Codd normal form): Table is in 3NF. All non-trivial functional dependencies depend on a superkey.
 - In the **HealthStatistics** table, attributes like **CountryID** , **RegionID** , and **SexID** are used as foreign keys (FK) to link the data to their respective tables. These FKs uniquely determine the related attributes (i.e. **CountryName** and **RegionName**) without introducing any partial dependency.
 - In the **Countries** table, the **CountryID** is a primary key (PK) that uniquely determines the **CountryName** , ensuring that no non-key attribute depends on something other than a candidate key.
5. 4NF: Table is in 3NF. For every Multi-Valued Dependency $A \twoheadrightarrow B$, is a candidate key.
 - The **HealthStatistics** table stores mortality data and links it to related tables for **Country** , **Region** , **AgeGroup** , and **Sex** . These attributes are stored in separate tables to avoid redundant data and to ensure that multi-valued dependencies, such as having multiple **AgeGroups** for a single **Country** , do not exist in the same record.

Database Structure

```
mysql> show databases;
+-----+
| Database |
+-----+
| WHO      |
| information_schema |
| mysql    |
| performance_schema |
| sys      |
+-----+
5 rows in set (0.00 sec)
```

Initial Commands (CREATE, INSERT INTO, LOAD DATA INFILE)

```
1 CREATE TABLE Temp (
2     RegionCode VARCHAR(10),
3     RegionName VARCHAR(100),
4     CountryCode VARCHAR(10),
5     CountryName VARCHAR(100),
6     Year INT,
7     Sex VARCHAR(50),
8     AgeGroup VARCHAR(50),
9     AgeGroupCode VARCHAR(100),
10    Number FLOAT,
11    PercentageOfCauseSpecificDeaths FLOAT,
12    AgeStandardizedDeathRate FLOAT,
13    DeathRate FLOAT
14 );
```

```

1 LOAD DATA INFILE
2 '/home/coder/project/WHOMortalityDatabase_Map_Noncommunicable_Diseases.csv'
3 INTO TABLE Temp
4 FIELDS TERMINATED BY ','
5 ENCLOSED BY '"'
6 LINES TERMINATED BY '\n'
7 IGNORE 1 ROWS
8 (RegionCode, RegionName, CountryCode, CountryName,
9 Year, Sex, @AgeGroupCode, AgeGroup, @Number,
10 @PercentageOfCauseSpecificDeaths, @AgeStandardizedDeathRate, @DeathRate)
11 SET
12   AgeGroupCode = @AgeGroupCode,
13   Number = NULLIF(TRIM(@Number), ''),
14   PercentageOfCauseSpecificDeaths = NULLIF(TRIM(@PercentageOfCauseSpecificDeaths), ''),
15   AgeStandardizedDeathRate = NULLIF(TRIM(@AgeStandardizedDeathRate), ''),
16   DeathRate = NULLIF(TRIM(@DeathRate), '');

```

While importing the data from csv into the table `Temp`, there was a lot of difficulties, with various error messages, the following are the changes I made before importing the file.

- remove first few rows of description
- remove last empty line
- remove extra comma from each row with regex `,\s*$`

```

1 CREATE TABLE Regions (
2     RegionID INT AUTO_INCREMENT PRIMARY KEY,
3     RegionCode VARCHAR(50) NOT NULL,
4     RegionName VARCHAR(100) NOT NULL
5 );

```



```
1 CREATE TABLE Countries (  
2     CountryID INT AUTO_INCREMENT PRIMARY KEY,  
3     CountryCode VARCHAR(50) NOT NULL,  
4     CountryName VARCHAR(100) NOT NULL,  
5     RegionID INT,  
6     FOREIGN KEY (RegionID) REFERENCES Regions(RegionID)  
7 );
```



```
1 CREATE TABLE AgeGroups (  
2     AgeGroupID INT AUTO_INCREMENT PRIMARY KEY,  
3     AgeGroupCode VARCHAR(50) NOT NULL,  
4     AgeGroup VARCHAR(100) NOT NULL  
5 );
```



```
1 CREATE TABLE Sex (  
2     SexID INT AUTO_INCREMENT PRIMARY KEY,  
3     Sex VARCHAR(50) NOT NULL  
4 );
```

```

1 CREATE TABLE HealthStatistics (
2     StatID INT AUTO_INCREMENT PRIMARY KEY,
3     CountryID INT,
4     Year INT,
5     AgeGroupID INT,
6     SexID INT,
7     Number FLOAT,
8     PercentageOfCauseSpecificDeaths FLOAT,
9     AgeStandardizedDeathRate FLOAT,
10    DeathRate FLOAT,
11    FOREIGN KEY (CountryID) REFERENCES Countries(CountryID),
12    FOREIGN KEY (AgeGroupID) REFERENCES AgeGroups(AgeGroupID),
13    FOREIGN KEY (SexID) REFERENCES Sex(SexID)
14 );

```

```
mysql> describe HealthStatistics;
```

Field	Type	Null	Key	Default	Extra
StatID	int	NO	PRI	NULL	auto_increment
CountryID	int	YES	MUL	NULL	
Year	int	YES		NULL	
AgeGroupID	int	YES	MUL	NULL	
SexID	int	YES	MUL	NULL	
Number	float	YES		NULL	
PercentageOfCauseSpecificDeaths	float	YES		NULL	
AgeStandardizedDeathRate	float	YES		NULL	
DeathRate	float	YES		NULL	

```
9 rows in set (0.00 sec)
```

```
mysql> use WHO;
```

```
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

```
Database changed
```

```
mysql> show tables;
```

Tables_in_WHO
AgeGroups
Countries
HealthStatistics
Regions
Sex
Temp

```
6 rows in set (0.00 sec)
```




```
1 INSERT INTO Sex (Sex)
2 SELECT DISTINCT Sex
3 FROM Temp;
```



```
1 INSERT INTO AgeGroups (AgeGroup, AgeGroupCode)
2 SELECT DISTINCT AgeGroup, AgeGroupCode
3 FROM Temp;
```



```
1 INSERT INTO Countries (CountryCode, CountryName, RegionID)
2 SELECT DISTINCT CountryCode, CountryName,
3     (SELECT RegionID FROM Regions WHERE RegionCode = Temp.RegionCode)
4 FROM Temp;
```



```
1 INSERT INTO Regions (RegionCode, RegionName)
2 SELECT DISTINCT RegionCode, RegionName
3 FROM Temp;
```

```

1 INSERT INTO HealthStatistics (CountryID, Year, AgeGroupID,
2 SexID, Number, PercentageOfCauseSpecificDeaths, AgeStandardizedDeathRate, DeathRate)
3 SELECT
4     (SELECT CountryID FROM Countries WHERE CountryCode = Temp.CountryCode),
5     Year,
6     (SELECT AgeGroupID FROM AgeGroups WHERE AgeGroup = Temp.AgeGroup),
7     (SELECT SexID FROM Sex WHERE Sex = Temp.Sex),
8     Number, PercentageOfCauseSpecificDeaths, AgeStandardizedDeathRate, DeathRate
9 FROM Temp;

```

```
mysql> select * from HealthStatistics limit 10;
```

StatID	CountryID	Year	AgeGroupID	SexID	Number	PercentageOfCauseSpecificDeaths	AgeStandardizedDeathRate	DeathRate
64	1	1987	1	1	1522	68.3738	0	15220
65	1	1987	2	1	1238	75.5339	0	7457.83
66	1	1987	3	1	1664	78.6761	0	5402.6
67	1	1987	4	1	1484	82.1705	0	3637.25
68	1	1987	5	1	1157	83.6587	0	1788.25
69	1	1987	6	1	892	84.2304	0	1152.45
70	1	1987	7	1	696	85.82	0	701.613
71	1	1987	8	1	426	83.0409	0	354.115
72	1	1987	9	1	270	68.5279	0	190.141
73	1	1987	10	1	140	63.0631	0	99.8573

```
10 rows in set (0.00 sec)
```

The database structure generally aligns with the dataset, as it contains distinct tables for countries, regions, and health statistics, allowing for efficient querying and analysis of mortality rates by region, age group, and gender. However, an issue arose when the data was initially imported from the CSV file. Due to faulty data types being set in the `Temp` table — specifically, assigning `INT` instead of `FLOAT` for numeric values and `FLOAT` instead of `VARCHAR` for long floats (wasn't able to import the full dataset if set to `FLOAT`) — fields such as `DeathRate` and `PercentageOfCauseSpecificDeaths` were populated with `NULL`.

The initial insertion of data was a smaller dataset (i.e. 100, 1000), however, whilst the testing data was correctly imported, the entire dataset couldn't, after many tries (10,000 at once, manually, etc), I made the decision to change the two faulty properties' data type from `FLOAT` to `VARCHAR` and convert them in the script instead.

After adjusting the data types correctly, the missing data was resolved, allowing for accurate analysis. This experience highlighted the importance of ensuring correct data types during the import process to maintain the integrity and usability of the dataset. Despite this challenge, the schema remains well-suited for the project's queries, as long as data integrity is preserved.

Query for Questions

```

1  SELECT
2      CASE
3          WHEN AgeGroups.AgeGroup IN ('[0]') THEN '0-9'
4          WHEN AgeGroups.AgeGroup IN ('[1-4]', '[5-9]') THEN '0-9'
5          WHEN AgeGroups.AgeGroup IN ('[10-14]', '[15-19]') THEN '10-19'
6          WHEN AgeGroups.AgeGroup IN ('[20-24]', '[25-29]') THEN '20-29'
7          WHEN AgeGroups.AgeGroup IN ('[30-34]', '[35-39]') THEN '30-39'
8          WHEN AgeGroups.AgeGroup IN ('[40-44]', '[45-49]') THEN '40-49'
9          WHEN AgeGroups.AgeGroup IN ('[50-54]', '[55-59]') THEN '50-59'
10         WHEN AgeGroups.AgeGroup IN ('[60-64]', '[65-69]') THEN '60-69'
11         WHEN AgeGroups.AgeGroup IN ('[70-74]', '[75-79]') THEN '70-79'
12         WHEN AgeGroups.AgeGroup IN ('[80-84]', '[85+]') THEN '80+'
13     END AS AgeRange -- Map specific age groups to broader ranges
14     , SUM(HealthStatistics.Number) AS TotalMortalityCount -- Sum mortality numbers
15 FROM HealthStatistics
16 JOIN Countries ON HealthStatistics.CountryID = Countries.CountryID -- Link countries
17 JOIN Regions ON Countries.RegionID = Regions.RegionID -- Link regions
18 JOIN AgeGroups ON HealthStatistics.AgeGroupID = AgeGroups.AgeGroupID -- Link age groups
19 WHERE Regions.RegionName = '${selectedRegion}' -- Filter by selected region
20     AND AgeGroups.AgeGroup <> '[All]' -- Exclude summary data
21     AND AgeGroups.AgeGroup <> '[Unknown]' -- Exclude unknown groups
22 GROUP BY AgeRange -- Group results by age range
23 ORDER BY
24     CASE
25         WHEN AgeRange = '0-9' THEN 1
26         WHEN AgeRange = '10-19' THEN 2
27         WHEN AgeRange = '20-29' THEN 3
28         WHEN AgeRange = '30-39' THEN 4
29         WHEN AgeRange = '40-49' THEN 5
30         WHEN AgeRange = '50-59' THEN 6
31         WHEN AgeRange = '60-69' THEN 7
32         WHEN AgeRange = '70-79' THEN 8
33         WHEN AgeRange = '80+' THEN 9
34     END; -- Ensure age ranges are ordered logically

```


Question 1

```

1  SELECT
2      CASE
3          WHEN ag.AgeGroup IN ('[0]') THEN '0-9'
4          WHEN ag.AgeGroup IN ('[1-4]', '[5-9]') THEN '0-9'
5          WHEN ag.AgeGroup IN ('[10-14]', '[15-19]') THEN '10-19'
6          WHEN ag.AgeGroup IN ('[20-24]', '[25-29]') THEN '20-29'
7          WHEN ag.AgeGroup IN ('[30-34]', '[35-39]') THEN '30-39'
8          WHEN ag.AgeGroup IN ('[40-44]', '[45-49]') THEN '40-49'
9          WHEN ag.AgeGroup IN ('[50-54]', '[55-59]') THEN '50-59'
10         WHEN ag.AgeGroup IN ('[60-64]', '[65-69]') THEN '60-69'
11         WHEN ag.AgeGroup IN ('[70-74]', '[75-79]') THEN '70-79'
12         WHEN ag.AgeGroup IN ('[80-84]', '[85+]') THEN '80+'
13     END AS AgeRange, -- Map specific age groups to broader ranges
14     r.RegionName, -- Include region names in results
15     SUM(hs.Number) AS TotalMortality -- Sum mortality numbers
16 FROM HealthStatistics hs
17 JOIN AgeGroups ag ON hs.AgeGroupID = ag.AgeGroupID -- Link age groups
18 JOIN Countries c ON hs.CountryID = c.CountryID -- Link countries
19 JOIN Regions r ON c.RegionID = r.RegionID -- Link regions
20 WHERE ag.AgeGroup NOT IN ('[All]', '[Unknown]') -- Exclude summary and unknown data
21 AND (
22     CASE
23         WHEN ag.AgeGroup IN ('[0]') THEN '0-9'
24         WHEN ag.AgeGroup IN ('[1-4]', '[5-9]') THEN '0-9'
25         WHEN ag.AgeGroup IN ('[10-14]', '[15-19]') THEN '10-19'
26         WHEN ag.AgeGroup IN ('[20-24]', '[25-29]') THEN '20-29'
27         WHEN ag.AgeGroup IN ('[30-34]', '[35-39]') THEN '30-39'
28         WHEN ag.AgeGroup IN ('[40-44]', '[45-49]') THEN '40-49'
29         WHEN ag.AgeGroup IN ('[50-54]', '[55-59]') THEN '50-59'
30         WHEN ag.AgeGroup IN ('[60-64]', '[65-69]') THEN '60-69'
31         WHEN ag.AgeGroup IN ('[70-74]', '[75-79]') THEN '70-79'
32         WHEN ag.AgeGroup IN ('[80-84]', '[85+]') THEN '80+'
33     END = '${ageGroup}' -- Filter results by age group
34 )
35 GROUP BY AgeRange, r.RegionName -- Group by age range and region
36 ORDER BY
37     CASE
38         WHEN AgeRange = '0-9' THEN 1
39         WHEN AgeRange = '10-19' THEN 2
40         WHEN AgeRange = '20-29' THEN 3
41         WHEN AgeRange = '30-39' THEN 4
42         WHEN AgeRange = '40-49' THEN 5
43         WHEN AgeRange = '50-59' THEN 6
44         WHEN AgeRange = '60-69' THEN 7
45         WHEN AgeRange = '70-79' THEN 8
46         WHEN AgeRange = '80+' THEN 9
47     END ASC, r.RegionName; -- Order logically by age range and region name

```

Question 2




```

1  SELECT
2      r.RegionName, -- Region name
3      s.Sex, -- Gender
4      SUM(h.Number) AS TotalMortality -- Total mortality for each gender
5  FROM
6      HealthStatistics h
7  JOIN
8      Countries c ON h.CountryID = c.CountryID -- Link countries
9  JOIN
10     Regions r ON c.RegionID = r.RegionID -- Link regions
11  JOIN
12     Sex s ON h.SexID = s.SexID -- Link gender
13  WHERE
14     h.Number IS NOT NULL -- Exclude records with null mortality numbers
15     AND s.Sex IN ('Male', 'Female', 'All') -- Include only specified genders
16     AND r.RegionName = '${region}' -- Filter by user-input region
17  GROUP BY
18     r.RegionName, s.Sex -- Group results by region and gender
19  ORDER BY
20     s.Sex; -- Sort results by gender

```

Question 3

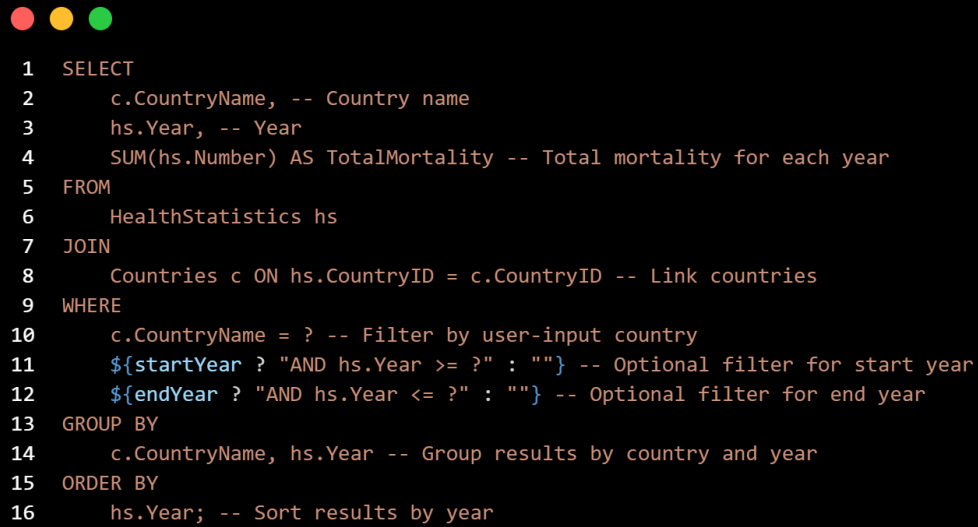


```

1  SELECT
2      c.CountryName,
3      COUNT(DISTINCT hs.Year) AS YearCoverage
4  FROM
5      Countries c
6  JOIN
7      HealthStatistics hs ON c.CountryID = hs.CountryID
8  GROUP BY
9      c.CountryName
10 ORDER BY
11     YearCoverage DESC
12 LIMIT 15; -- Limit to top 15 countries with the most coverage

```

Top 15 most data coverage countries used in the drop-down menu



```
1  SELECT
2      c.CountryName, -- Country name
3      hs.Year, -- Year
4      SUM(hs.Number) AS TotalMortality -- Total mortality for each year
5  FROM
6      HealthStatistics hs
7  JOIN
8      Countries c ON hs.CountryID = c.CountryID -- Link countries
9  WHERE
10     c.CountryName = ? -- Filter by user-input country
11     ${startYear} ? "AND hs.Year >= ?" : "" -- Optional filter for start year
12     ${endYear} ? "AND hs.Year <= ?" : "" -- Optional filter for end year
13  GROUP BY
14     c.CountryName, hs.Year -- Group results by country and year
15  ORDER BY
16     hs.Year; -- Sort results by year
```

Question 4

Web Application

Logo designed in Canva using all copy-right-free materials.

Health Statistics Analysis



This dashboard contains 22 key statistics that provide insights into global mortality data. There are four interactive charts after that allow exploration within the data.

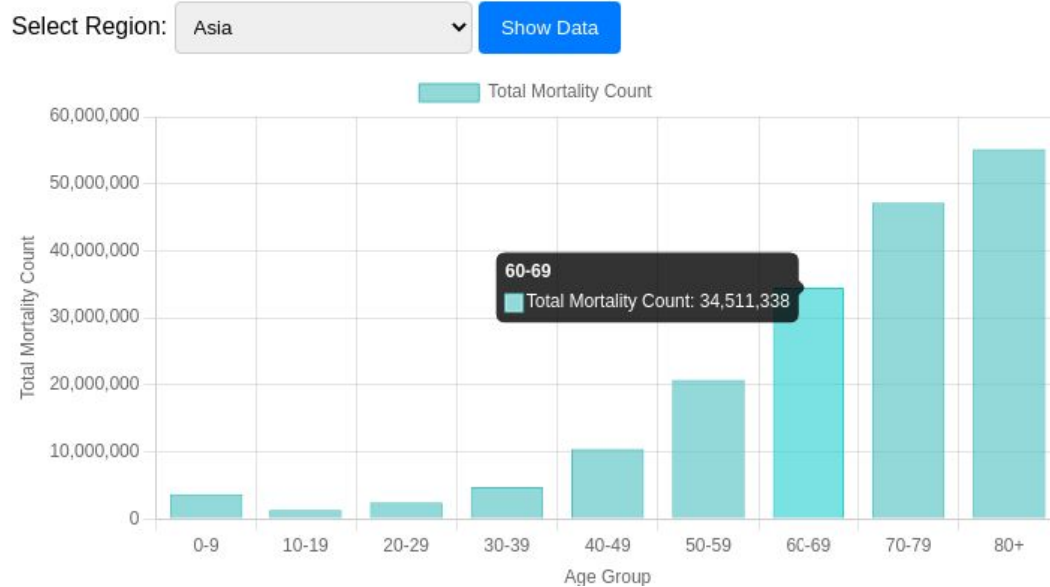
Overview

Statistic	Value
Total Rows in Database	312542
Percentage of Rows with Missing Mortality Data	0.0000%
Percentage of Rows with Missing Death Rate Data	0.0000%
Countries with Complete Data Across All Years	119
Total Years Covered in the Database	73
Earliest Year in the Database	1950
Latest Year in the Database	2022
Average Death Rate Across All Records	1651.5721042881894
Maximum Death Rate Recorded	136842
Minimum Death Rate Recorded	0
Average Percentage of Cause-Specific Deaths	58.54607195277694
Region with the Most Mortality Data	Europe
Country with the Highest Total Mortality	United States of America
Total Number of Countries	119
Average Death Rate Across All Records	1651.5721042881894
Maximum Death Rate Recorded	136842
Minimum Death Rate Recorded	0
Average Percentage of Cause-Specific Deaths	58.54607195277694
Maximum Percentage of Cause-Specific Deaths	100
Total Number of Years in the Database	73
Countries with Missing Mortality Data	0.0000
Countries with Missing Death Rate Data	0.0000
Total Mortality Records Across All Countries	2668257812
Region with the Most Mortality Data	Europe
Country with the Highest Total Mortality	United States of America

Dashboard table view of the 22 statistics of the entire dataset

1. Mortality by Region

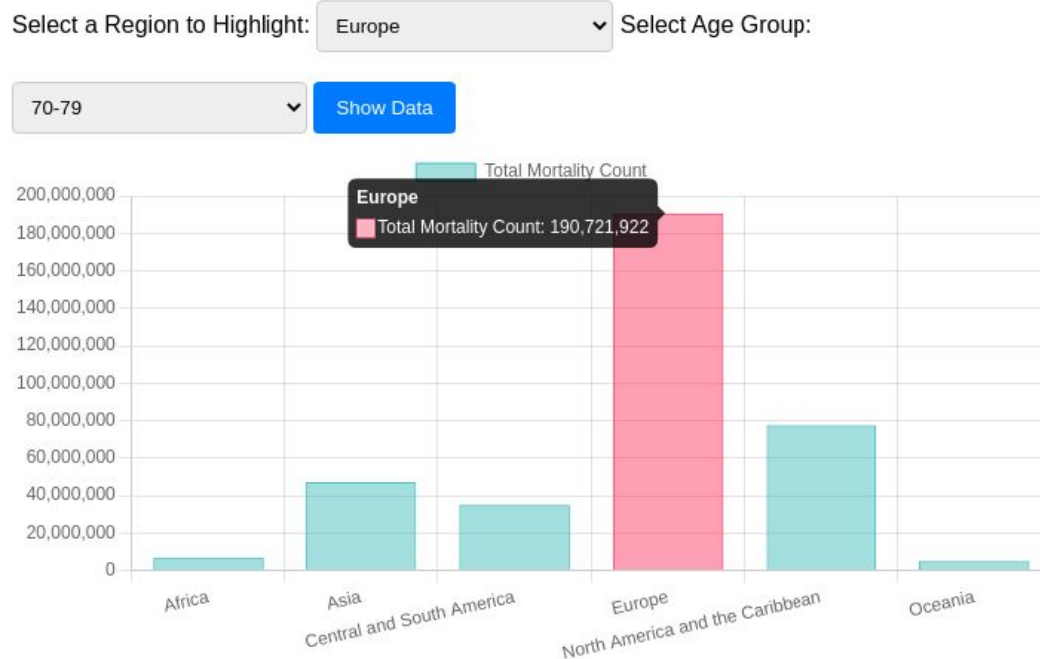
A bar chart displaying mortality data by age group (0-9, 10-19, etc.) for a selected region. Simply choose a region from the dropdown to view the data.



Question 1 answered by bar charts with a drop-down menu selecting region

2. Mortality by Age Group in Regions

This bar chart allows you to explore mortality rates by age group across regions. Select an age group and highlight a region to compare.



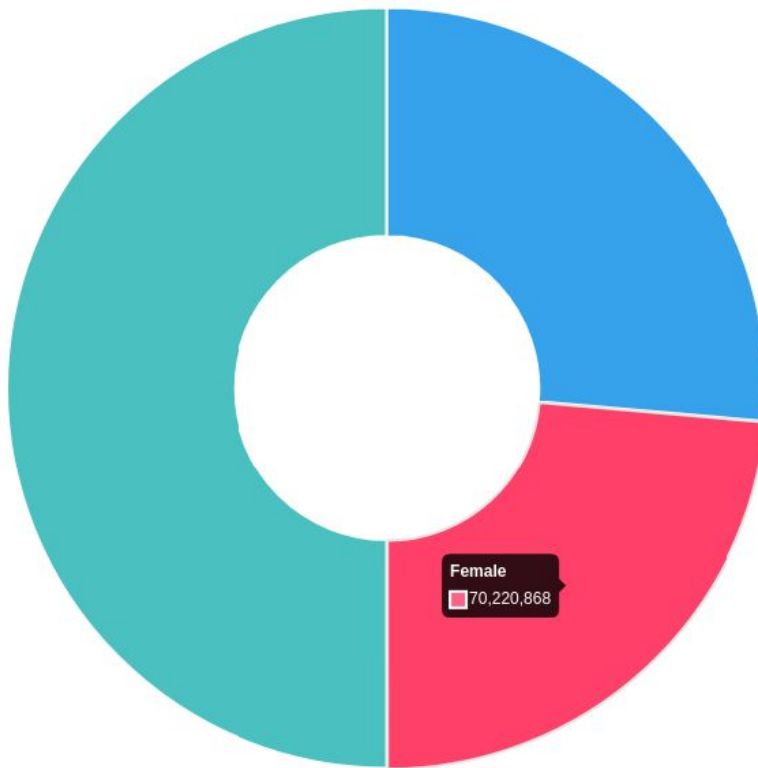
Question 2 answered by bar charts with two drop-downs menu selecting age group and a region to highlight

3. Gender Difference in Mortality by Region

A donut chart showing the difference in mortality between males, females, and all individuals for a selected region.
The data updates based on the region selected.

Select Region:

☐ Male ☐ Female ☐ All

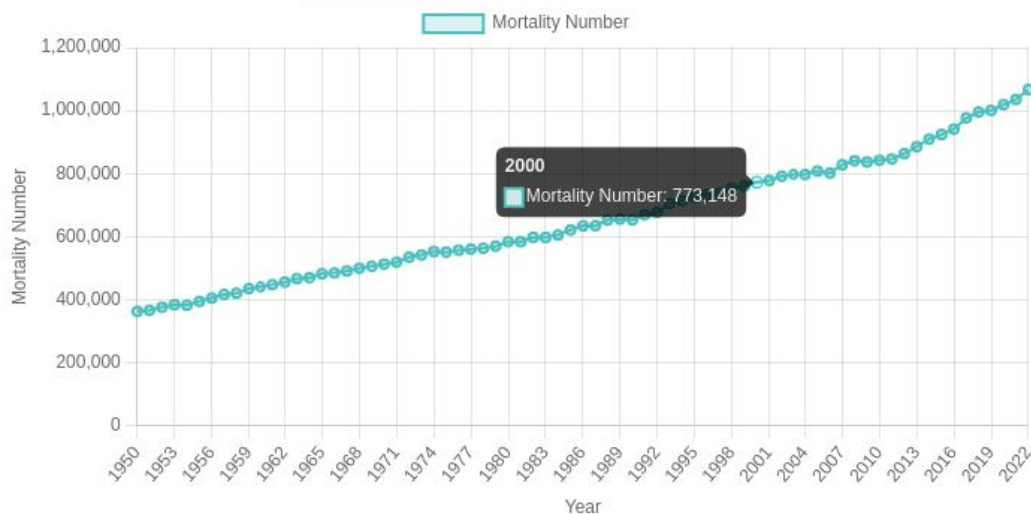


Question 3 answered by a doughnut chart with a drop-down menu selecting the region

4. Yearly Mortality by Country

A line chart showing yearly mortality data for the top 15 countries with the most complete data. Users can select a country and filter the data by a custom range of years. Only the top 15 countries that have the most data coverage across years are available.

Select Country: Start Year: End Year:

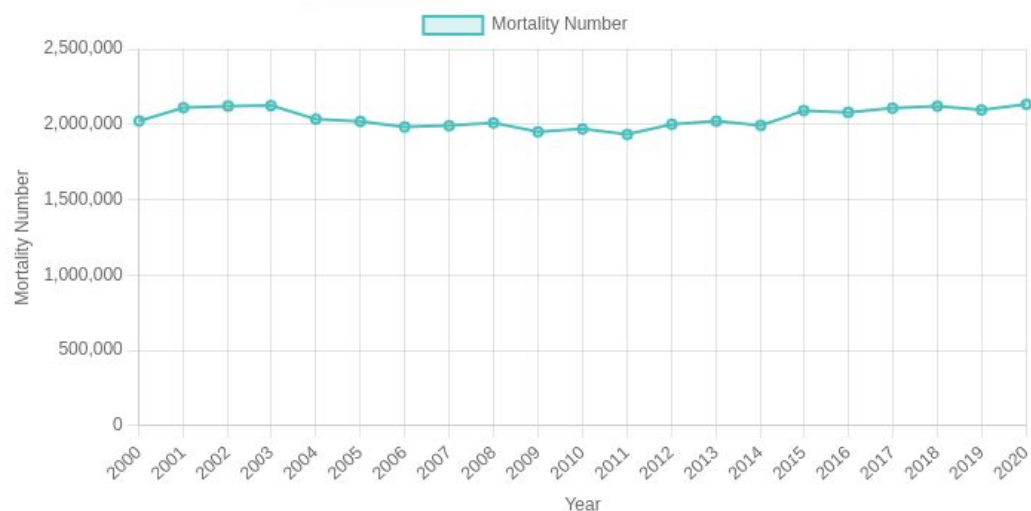


Question 4 answered by a line chart with the top 15 countries available

4. Yearly Mortality by Country

A line chart showing yearly mortality data for the top 15 countries with the most complete data. Users can select a country and filter the data by a custom range of years. Only the top 15 countries that have the most data coverage across years are available.

Select Country: Start Year: End Year:



Question 4 with the optional year range drop-down menu

Conclusion

This project developed a database application to analyze global mortality trends, using WHO data for noncommunicable diseases. It explored mortality by region, age group, gender, and over time in specific countries. The application effectively aggregated data and visualized trends with Chart.js, offering insights into health disparities.

A key strength was the efficient querying and visualization of large datasets. However, data import issues due to incorrect data types in the `Temp` table initially caused missing values. A limitation was the incompleteness of data in certain regions, which affected analysis accuracy. Despite these challenges, the application provided valuable insights, with future improvements focusing on data quality and completeness.