# Solving an SDG Problem with Data

https://eu.docworkspace.com/d/sIDKrwqyNAomzjrYG

### https://gamma.app/docs/Improving-Maternal-Health-Outcomes-through-Data-Analysis-f41fw4u8duq0r11

### SDG Selection and Problem Definition

#### ****SDG Selection:****

**SDG 3: Good Health and Well-being**

#### ****Problem Definition:****

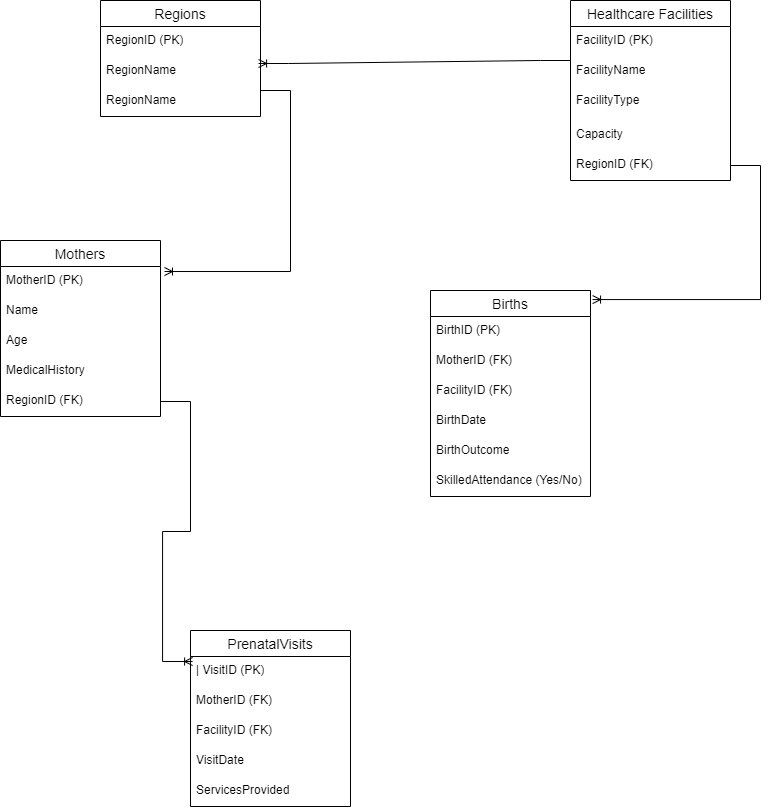
**Problem:** High maternal mortality rates in developing countries.

**Specific Issue:** One of the major contributors to high maternal mortality rates in developing countries is the lack of access to adequate prenatal care and the availability of skilled health professionals during childbirth. This project will focus on analyzing the availability and impact of prenatal care services and skilled birth attendance in reducing maternal mortality rates.

**Objective:** The goal is to identify patterns and correlations between maternal mortality rates and factors like prenatal care visits, availability of skilled birth attendants, and healthcare facility access. The insights gained will guide interventions to improve maternal health outcomes in high-risk regions.

### Database Design

#### ****ERD Design:****



Entities:

1. **Mothers**: Tracks personal details and medical history.
2. **Prenatal Visits**: Records each visit, including the date, healthcare provider, and services provided.
3. **Healthcare Facilities**: Details of available facilities.
4. **Births**: Records childbirths, including attendance by skilled professionals and outcomes.
5. **Regions**: Geographical data to link mothers and facilities to specific regions.

Relationships:

* **Mothers** to **Prenatal Visits** (1)
* **Mothers** to **Births** (1)
* **Healthcare Facilities** to **Prenatal Visits** (1)
* **Healthcare Facilities** to **Births** (1)
* **Regions** to **Mothers**, **Healthcare Facilities**, **Births** (1)

Schema:

CREATE TABLE Regions (

RegionID INT PRIMARY KEY,

RegionName VARCHAR(100),

Population INT,

HealthcareAccessScore DECIMAL(5, 2)

);

CREATE TABLE Mothers (

MotherID INT PRIMARY KEY,

Name VARCHAR(100),

Age INT,

RegionID INT,

MedicalHistory TEXT,

FOREIGN KEY (RegionID) REFERENCES Regions(RegionID)

);

CREATE TABLE HealthcareFacilities (

FacilityID INT PRIMARY KEY,

FacilityName VARCHAR(100),

RegionID INT,

FacilityType VARCHAR(50),

Capacity INT,

FOREIGN KEY (RegionID) REFERENCES Regions(RegionID)

);

CREATE TABLE PrenatalVisits (

VisitID INT PRIMARY KEY,

MotherID INT,

FacilityID INT,

VisitDate DATE,

ServicesProvided TEXT,

FOREIGN KEY (MotherID) REFERENCES Mothers(MotherID),

FOREIGN KEY (FacilityID) REFERENCES HealthcareFacilities(FacilityID)

);

CREATE TABLE Births (

BirthID INT PRIMARY KEY,

MotherID INT,

FacilityID INT,

BirthDate DATE,

BirthOutcome VARCHAR(50),

SkilledAttendance BOOLEAN,

FOREIGN KEY (MotherID) REFERENCES Mothers(MotherID),

FOREIGN KEY (FacilityID) REFERENCES HealthcareFacilities(FacilityID)

);

### ****Data Retrieval:****

#### ****Query 1: List all mothers who had fewer than 4 prenatal visits.****

SELECT Mothers.MotherID, Mothers.Name, COUNT(PrenatalVisits.VisitID) AS NumberOfVisits

FROM Mothers

LEFT JOIN PrenatalVisits ON Mothers.MotherID = PrenatalVisits.MotherID

GROUP BY Mothers.MotherID, Mothers.Name

HAVING COUNT(PrenatalVisits.VisitID) < 4;

**Explanation:** This query joins the Mothers table with the PrenatalVisits table to count the number of prenatal visits for each mother. It then filters out those who had fewer than 4 visits.

**Query 2: Count the number of births attended by skilled professionals per region.**

SELECT Regions.RegionName, COUNT(Births.BirthID) AS SkilledAttendanceCount

FROM Births

INNER JOIN HealthcareFacilities ON Births.FacilityID = HealthcareFacilities.FacilityID

INNER JOIN Regions ON HealthcareFacilities.RegionID = Regions.RegionID

WHERE Births.SkilledAttendance = 'Yes'

GROUP BY Regions.RegionName;

**Explanation:** This query counts the number of births where skilled attendance was marked as "Yes," grouping the results by region.

### ****Data Analysis:****

#### ****Query 3: Correlate the number of prenatal visits with birth outcomes to determine the impact on maternal mortality.****

SELECT Mothers.MotherID, Mothers.Name, COUNT(PrenatalVisits.VisitID) AS NumberOfVisits, Births.BirthOutcome

FROM Mothers

LEFT JOIN PrenatalVisits ON Mothers.MotherID = PrenatalVisits.MotherID

LEFT JOIN Births ON Mothers.MotherID = Births.MotherID

GROUP BY Mothers.MotherID, Mothers.Name, Births.BirthOutcome

ORDER BY NumberOfVisits DESC;

**Explanation:** This query correlates the number of prenatal visits with birth outcomes, such as mortality. It joins Mothers, PrenatalVisits, and Births and groups the results by each mother and their birth outcome.

**Query 4: Analyze regions with the highest maternal mortality rates and compare them with the availability of healthcare facilities.**

SELECT Regions.RegionName, COUNT(Births.BirthID) AS MaternalMortalityCount, COUNT(HealthcareFacilities.FacilityID) AS FacilityCount

FROM Regions

LEFT JOIN HealthcareFacilities ON Regions.RegionID = HealthcareFacilities.RegionID

LEFT JOIN Births ON Regions.RegionID = HealthcareFacilities.RegionID AND Births.BirthOutcome = 'Maternal Mortality'

GROUP BY Regions.RegionName

ORDER BY MaternalMortalityCount DESC;

**Explanation:** This query finds the number of maternal mortality cases by region and compares it with the number of healthcare facilities in those regions. It provides insight into whether regions with higher maternal mortality have fewer healthcare facilities.

* **Query 1** helps identify mothers at risk by focusing on those who have fewer than the recommended number of prenatal visits.
* **Query 2** evaluates how effectively healthcare facilities are ensuring skilled professionals attend births across regions.
* **Query 3** provides a critical analysis of how prenatal visits impact birth outcomes, particularly maternal mortality.
* **Query 4** investigates potential correlations between maternal mortality rates and healthcare facility availability.

These queries are designed to provide valuable insights for addressing the Sustainable Development Goal (SDG) related to maternal and child health, ensuring that interventions can be more targeted and effective.

### ****Integration and Testing****

#### ****Integration Documentation****

**Exporting SQL Query Results:**

* 1. **Step 1:** Run the SQL queries in your database management system (e.g., MySQL, SQL Server, PostgreSQL).
  2. **Step 2:** Export the query results to a CSV file:
     1. In MySQL Workbench: After running the query, right-click on the result grid, choose "Export," and select "CSV."
     2. In SQL Server Management Studio: After running the query, right-click on the result grid, choose "Save Results As," and save as a CSV file.
  3. **Step 3:** Save the CSV file in a designated folder where it can be easily accessed by Excel.

**Importing Data into Excel:**

* 1. **Step 1:** Open Microsoft Excel and navigate to the "Data" tab.
  2. **Step 2:** Click on "Get Data" and select "From Text/CSV."
  3. **Step 3:** Locate and select the CSV file that contains your SQL query results.
  4. **Step 4:** Follow the import wizard to load the data into Excel. Ensure that the data types are correctly identified.
  5. **Step 5:** Once the data is imported, review it to confirm that all columns and rows are correctly populated and consistent with the data in the database.

**Data Consistency Check:**

* 1. **Step 1:** Compare the data in Excel with the data in the original SQL query results.
  2. **Step 2:** Check for any discrepancies in row counts, missing data, or incorrectly formatted values.
  3. **Step 3:** If any inconsistencies are found, re-export the data and re-import it into Excel, making necessary adjustments to ensure accuracy.

**Setting Up the Excel Dashboard:**

* 1. **Step 1:** Organize the imported data into separate sheets if necessary (e.g., one sheet for mothers, one for prenatal visits, etc.).
  2. **Step 2:** Use Excel’s pivot tables and charts to create the visualizations needed for your dashboard.
  3. **Step 3:** Ensure that the pivot tables are connected to the data and are refreshed automatically when new data is added or updated.

#### ****Testing****

**Functionality Testing:**

* 1. **Step 1:** Test each pivot table to ensure it accurately reflects the data from the SQL queries.
  2. **Step 2:** Verify that each chart in the dashboard correctly represents the data and matches the expected outcomes.
  3. **Step 3:** Check for any broken links or errors in the Excel formulas, pivot tables, and charts.

**Dynamic Updates:**

* 1. **Step 1:** Simulate an update by modifying a portion of the original SQL data and re-exporting it to a CSV file.
  2. **Step 2:** Re-import the updated CSV file into Excel and refresh all data connections, pivot tables, and charts.
  3. **Step 3:** Confirm that the dashboard dynamically reflects the changes and updates without errors.

**User Experience Testing:**

* 1. **Step 1:** Navigate through the dashboard to ensure all interactive elements like slicers and filters work as intended.
  2. **Step 2:** Test the filtering options to see if they correctly update all relevant charts and tables.
  3. **Step 3:** Verify that the dashboard is user-friendly, with clear labels, appropriate chart types, and intuitive navigation.

### ****Conclusion****

This integration and testing phase ensures that your data flows seamlessly from your database to Excel, maintaining consistency and accuracy. Testing confirms that your dashboard is reliable, user-friendly, and dynamically updates based on new data, making it a robust tool for addressing the SDG problem you’ve chosen.