LONDON'S GLOBAL UNIVERSITY

UCL

INST0001 Individual Assignment

Anon.

INST0001 Database Systems 2024-25

Module Tutors: Karen Stepanyan and Foteini Valeonti

Student Number: 24119250

Word Count: 1068

April 2025

Table of Contents

1 | Context and Relevance of SDG and Indicators

To track progress toward SDG 2 (Zero Hunger), the charity uses two key indicators:

- Indicator 8: Proportion of population below minimum dietary energy consumption.
- Indicator 10: Prevalence of stunting and wasting in children under 5.

Concern's projects relevant to SDG 2 fall into three categories:

- Emergency response nutrition projects (ERNE)
- Long-term food production efforts (e.g., in DPRK)
- Malnutrition awareness initiatives (e.g., in Kenya)

ERNE projects, while essential, are harder to evaluate as they are short-term. Indicators focused on specific demographics do not encapsulate the project. Indicator 10, however, is general and easier to measure in emergency contexts, especially where data collection is limited in a disaster.

For awareness projects, indicator 10 is a good proxy as it reflects outcomes among the target demographic, malnourished children under-5 through clinics for mothers and nutrient powders.

Food production projects are directly related to Indicator 8, which reflects access to sufficient food. Although Concern supports agricultural development (Etherington and Anema, 2014), many agricultural indicators (e.g. Indicator 15, nitrogen efficiency) are not applicable to the small-scale facilities they implement, and agriculture metrics do not cover all overall food projects (Concern Worldwide, 2019). Thus, Indicator 8 is the most practical as a metric which encapsulates the entire project archetype.

2 | Database Design Decisions

The database includes two main groups of entity types: those used for mapping and those tracking SDG indicator data. The classification of these is shown in Figure 2.1 (the *italicised* entity types are those I worked on and justify below).

Mapping Entity Types	**Indicator Entity Types**
Country	*Malnourishment*
Project	*Stunting and Wasting*
	Safe Water Services
	Wastewater Flows
	Under-5 Mortality
	Vaccinations

Table 2.1: Mapping between Entity Types and Indicator Entity Types

In the tables below, I define the following reasons:

- Mapping attribute: attributes used for relations between tables
- Filtering attribute: attributes which can be used to filter for specific data in queries
- Disaggregated data: data columns for disaggregated data

The `Project` and `Country` entities serve as reference points and enable more complex queries. The `Project` entity is intended to be used to implement restrictions and as a starting point for queries that aim to track progress for specific countries, with justifications for its attributes shown in Figure 2.2.

In the `Country` entity, both `country_name` and `country_code` are candidate keys. While `country_name` is used as the primary key for clarity, `country_code` aligns better with most datasets (which use tricodes) and is used as the foreign key in related indicator tables. Justifications for all attributes are given in Figure 2.3.

All indicator entity types have a similar structure: a composite key composed of `year`

Project Attribute	**Justification for Existence**
`project_id`	Primary key
`start_date`	Determines which years to look at data for
`end_date`	Determines which years to look at data for
`country_code`	Mapping attribute*
`SDG`	Filtering attribute*
`indicator`	Mapping attribute*

Table 2.2: Justification for Project Entity Attributes

Country Attribute	**Justification for Existence**
`country_name`	Primary key
`country_code`	Mapping attribute
`HDI`	Filtering attribute
`continent`	Filtering attribute

Table 2.3: Justification for Country Entity Attributes

and `country_code`, and a few specific attributes which store the data. A wide-table format (with disaggregated values as separate columns rather than using an attribute to distinguish between disaggregation types) was used for two main reasons:

1. Given the size of the UN/WHO databases from which most indicators are derived, the database model is relatively static; further disaggregation is unlikely to be available for 3–8 years (UN-GGIM, 2016, 2018, 2020, 2022), meaning scalability matters less.

2. The wide-table model is simpler; one row provides all the information for a given year and country, and SQL queries can be less complex to achieve the same result (Reynolds, 2018).

Figure 2.4 provides justifications for all indicator attributes.

Selected Indicator Attributes	**Justification for Existence**
`country_code`	Primary key and mapping attribute
`year`	Primary key and mapping attribute
`undnr_rate_male`	Disaggregated data
`undnr_rate_female`	Disaggregated data
`stunt_rate`	Disaggregated data
`waste_rate`	Disaggregated data
`stunt_waste`	Disaggregated data

Table 2.4: Justification for Selected Indicator Entity Attributes

3 | SQL Implementation

For the UNDERNOURISHMENT and STUNTING_AND_WASTING tables, aggregate data fields are defined as NOT NULL, since overall values must always be recorded. Disaggregated fields can be NULL if detailed data is unavailable. Additional constraints ensure data validity, for example: undnr_rate_female between 0 and 100

A trigger constraint was also written to prevent indicator data unless a country has at least one linked project of the same indicator type. This constraint could not be implemented due to phpMyAdmin restrictions. A similar constraint could apply to time (e.g., limiting data to a window around project dates), but such an interval would likely be different for different projects.

Several design decisions were made regarding data types to improve efficiency and consistency:

- **Minimising VARCHAR() length:** For example, VARCHAR(60) is used for country names, as the longest known name is 56 characters.
- **Using VARCHAR() instead of TINYTEXT**: VARCHAR allows for length control and improves sorting performance (Stack Exchange, 2018).
- **Using CHAR(3) for country_code:** This enforces that all country codes are exactly three characters long.
- **Using DECIMAL over DOUBLE:** As the values are moderate in size, DECIMAL ensures precision and avoids the rounding issues associated with floating-point storage (MySQL, 2025).

4 | SQL SDG Monitoring

The database can be used to monitor the charity's progress towards Zero Hunger (SDG 2) in the following key ways:

1. Yearly trends for malnourishment and stunting/wasting.
2. Percentage change from the previous year in malnourishment and stunting/wasting rates.
3. Average change across continents in malnourishment and stunting/wasting.

Customizable Filters

We can further refine the analysis by applying filters such as:

- A specified interval of Human Development Index (HDI) values
- A particular year range
- Sex-specific analysis for malnourishment (male/female)
- Comparison between:
  - Stunt rate
  - Wasting rate
  - Proportion both stunted and wasted

Best Practices

- Use window functions for calculating year-over-year trends (e.g, LAG()).
- Clear and simple output columns (Lemahieu et Al, 2018)

5 | Lessons Learned and Reflection

This project was a stressful but valuable learning experience. I misinterpreted the brief and initially focused on building a general-purpose charity database rather than one tailored to SDG tracking. While this led to unnecessary work and several lengthy group meetings, it proved useful for the individual component; I found that I could write about technical topics such as indicator selection and database design efficiently and with genuine enjoyment.

The pressure tested my resilience and proved that even if things go wrong, I am capable of recovering and producing work I am proud of. This experience has shown me that setbacks do not define outcomes, and persistence matters more.

From a technical perspective, I developed a deeper understanding of SQL, particularly constraints, data types (e.g., VARCHAR vs TEXT, DECIMAL vs DOUBLE), and advanced query logic such as wildcards and the CASE statement. These concepts fundamentally changed my view of SQL, making it feel less like a static language and more like logical problem-solving. I now approach SQL and coding as logical skills rather than foreign languages.

As part of my original work, I reviewed the charity's existing database and supporting documentation to extract useful reports and evaluations. This process sharpened my ability to quickly identify relevant information, an essential skill I expect to draw on in future academic and professional projects.

Bibliography

[1] Concern Worldwide, European Commission and Etherington, A. (2014). *Multi Sector Nutrition & Food Security (MSNFS) Project Evaluation.* [online] Available at: https://admin.concern.org.uk/sites/default/files/media/migrated/final_evaluation_of_the_multi_sector_nutrition_and_food_security_project.pdf (Accessed 15 March 2025).

[2] Concern Worldwide (2019). *Emergency Nutrition Response Project for Drought Affected Communities in Marsabit County, Kenya - Final Evaluation 2019*. Available at: https://www.concern.org.uk/knowledge-hub/emergency-nutrition-response-project-drought-affected-communities-marsabit-county (Accessed: 13 March 2025).

[3] Reynolds, D. (2018). *Data Tables: Long or Wide.* Available at: https://community.ibm.com/community/user/ai-datascience/blogs/diane-reynolds/2018/07/26/data-tables-long-or-wide (Accessed: 1 April 2025).

[4] Stack Exchange (2018). *Performance Difference between TEXT and VARCHAR in MySQL.* Available at: https://dba.stackexchange.com/questions/222176/performance-difference-between-text-and-varchar-in-mysql (Accessed: 29 March 2025).

[5] MySQL Reference Manual (2025). *Numeric Data Types.* Available at: https://dev.mysql.com/doc/refman/8.0/en/numeric-types.html (Accessed: 10 April 2025).

[6] Lemahieu, W., vanden Broucke, S., and Baesens, B. (2018). *Principles of Database Management: The Practical Guide to Storing, Managing and Analyzing Big and Small Data.* Cambridge: Cambridge University Press.

A | ER Diagram

Figure B.3 shows the ER diagram representing the system architecture.

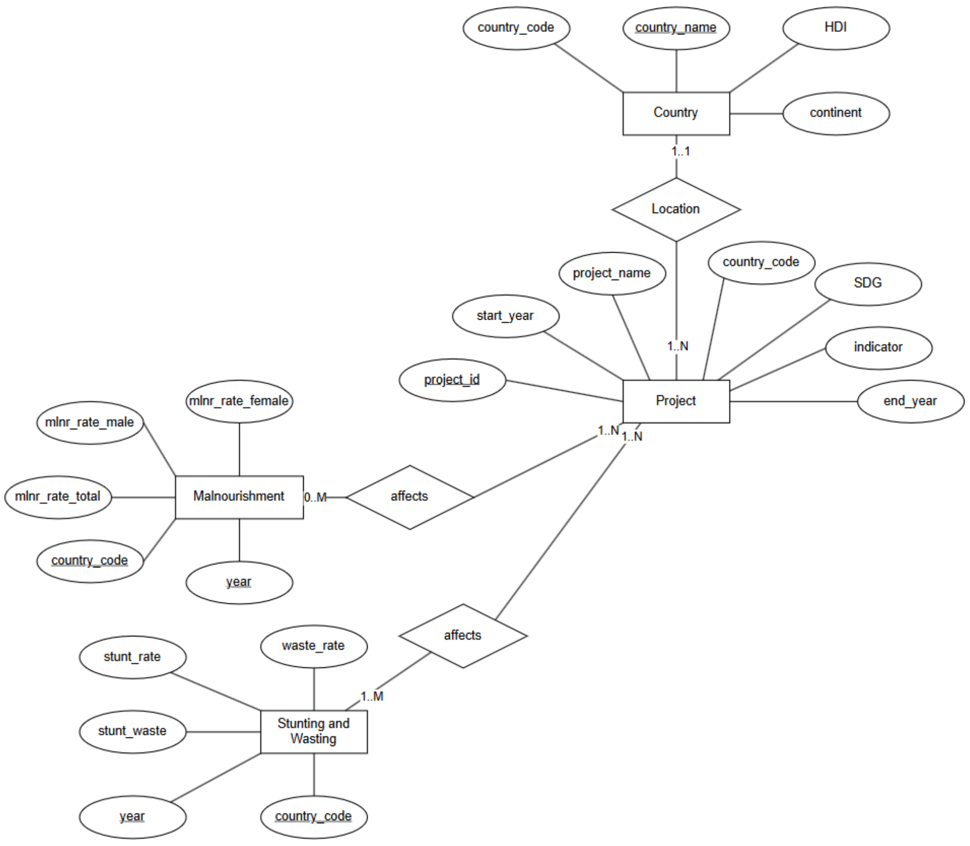
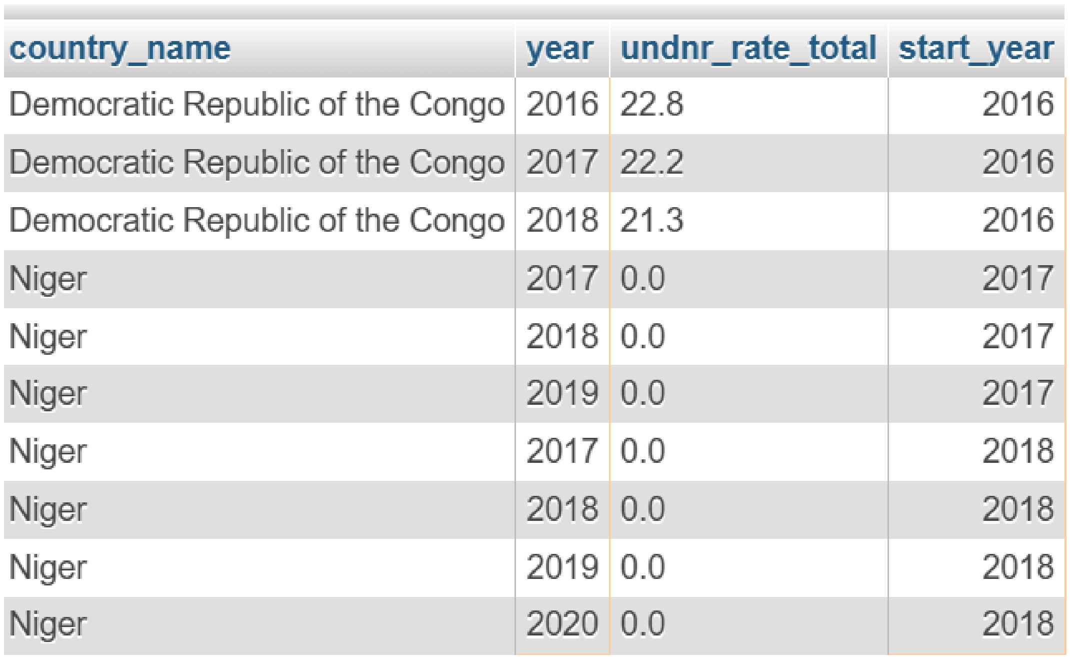


Figure A.1: Entity-Relationship Diagram

B | Example SQL Queries and Outputs

1. Yearly Trends for Overall Malnourishment (HDI < 0.5)

```sql
SELECT
    cn.country_name,
    m.year,
    m.undnr_rate_total,
    p.start_year
FROM
    MALNOURISHMENT m
JOIN
    COUNTRY_CODE cn ON m.country_code = cn.country_code
JOIN
    COUNTRY c ON cn.country_name = c.country_name
JOIN
    PROJECT p ON m.country_code = p.country_code
WHERE
    c.HDI < 0.5
    AND m.year BETWEEN (p.start_year - 2) AND (p.start_year + 2)
ORDER BY
    cn.country_name,
    p.start_year,
    m.year;
```



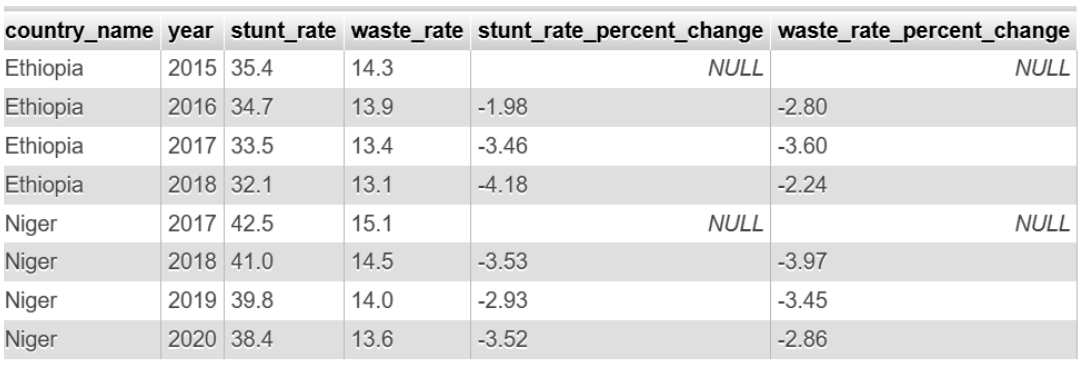
country_name	year	undnr_rate_total	start_year
Democratic Republic of the Congo	2016	22.8	2016
Democratic Republic of the Congo	2017	22.2	2016
Democratic Republic of the Congo	2018	21.3	2016
Niger	2017	0.0	2017
Niger	2018	0.0	2017
Niger	2019	0.0	2017
Niger	2017	0.0	2018
Niger	2018	0.0	2018
Niger	2019	0.0	2018
Niger	2020	0.0	2018

Figure B.1: Query 1 output

2. Percentage Change in Stunting and Wasting Rates

```sql
SELECT
  c.country_name,
  sw.year,
  sw.stunt_rate,
  sw.waste_rate,
  -- Percent change in stunt_rate
  CASE
    WHEN LAG(sw.stunt_rate) OVER (PARTITION BY c.country_name ORDER BY
 sw.year) IS NOT NULL
      AND LAG(sw.stunt_rate) OVER (PARTITION BY c.country_name ORDER BY
 sw.year) != 0
    THEN ROUND(
      100 * (sw.stunt_rate - LAG(sw.stunt_rate) OVER (PARTITION BY c.
 country_name ORDER BY sw.year))
```

```sql
      / LAG(sw.stunt_rate) OVER (PARTITION BY c.country_name ORDER BY sw.year), 2
    )
    ELSE NULL
  END AS stunt_rate_percent_change,
  -- Percent change in waste_rate
  CASE
    WHEN LAG(sw.waste_rate) OVER (PARTITION BY c.country_name ORDER BY sw.year) IS NOT NULL
      AND LAG(sw.waste_rate) OVER (PARTITION BY c.country_name ORDER BY sw.year) != 0
    THEN ROUND(
      100 * (sw.waste_rate - LAG(sw.waste_rate) OVER (PARTITION BY c.country_name ORDER BY sw.year))
      / LAG(sw.waste_rate) OVER (PARTITION BY c.country_name ORDER BY sw.year), 2
    )
    ELSE NULL
  END AS waste_rate_percent_change
FROM STUNTING_WASTING sw
JOIN COUNTRY_CODE cc ON sw.country_code = cc.country_code
JOIN COUNTRY c ON cc.country_name = c.country_name
ORDER BY c.country_name, sw.year;
```



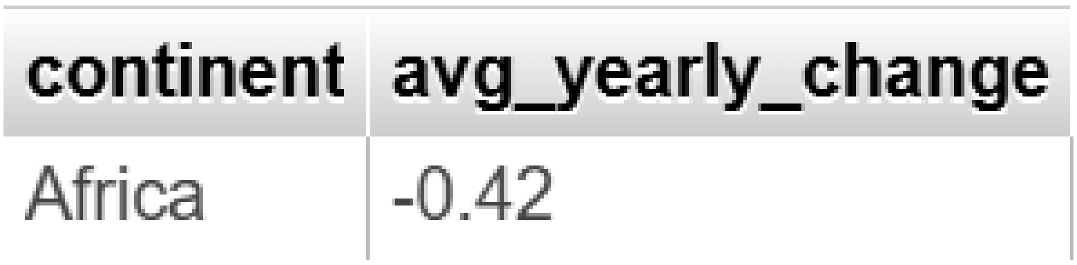
country_name	year	stunt_rate	waste_rate	stunt_rate_percent_change	waste_rate_percent_change
Ethiopia	2015	35.4	14.3	*NULL*	*NULL*
Ethiopia	2016	34.7	13.9	-1.98	-2.80
Ethiopia	2017	33.5	13.4	-3.46	-3.60
Ethiopia	2018	32.1	13.1	-4.18	-2.24
Niger	2017	42.5	15.1	*NULL*	*NULL*
Niger	2018	41.0	14.5	-3.53	-3.97
Niger	2019	39.8	14.0	-2.93	-3.45
Niger	2020	38.4	13.6	-3.52	-2.86

Figure B.2: Query 2 output

3. Average Change in Total Malnutrition Rates by Continent

```sql
WITH malnutrition_with_change AS (
  SELECT
    c.continent,
    cc.country_name,
    m.year,
    m.undnr_rate_total,
    LAG(m.undnr_rate_total) OVER (PARTITION BY cc.country_name ORDER BY
    m.year) AS previous_rate
  FROM MALNOURISHMENT m
  JOIN COUNTRY_CODE cc ON m.country_code = cc.country_code
  JOIN COUNTRY c ON cc.country_name = c.country_name
)

SELECT
  continent,
  ROUND(AVG(mw.undnr_rate_total - mw.previous_rate), 2) AS
   avg_yearly_change
FROM malnutrition_with_change mw
WHERE mw.previous_rate IS NOT NULL
GROUP BY continent
ORDER BY avg_yearly_change DESC;
```



continent	avg_yearly_change
Africa	-0.42

Figure B.3: Query 3 output