A yellow triangle with black background

Description automatically generatedA red and black rectangle

Description automatically generated

HDPSA Milestone 2

Bin 381 Group A

9/19/2025

**Members:**

Llewellyn Jakobus Fourie – 601314

Juan Oosthuizen – 600161

Erin David Cullen – 600531

Qhamaninande Ndlume – 601436

Table of Contents

[Introduction 3](#_Toc209192716)

[Step 1: Data Selection 3](#_Toc209192717)

[Step 2: Verifying Data Quality 3](#_Toc209192718)

[Step 3: Data Cleaning 4](#_Toc209192719)

[Step 4: Encoding and Scaling 4](#_Toc209192720)

[Step 5: Train-Test Split 4](#_Toc209192721)

[Verify Data Quality - Statistical Validation and Field Selection 6](#_Toc209192722)

[1 Data Preparation & Standardization 6](#_Toc209192723)

[1.1 Setup Global Variables 6](#_Toc209192724)

[1.2 Load All Datasets 7](#_Toc209192725)

[1.3 Standardize Field Structure 8](#_Toc209192726)

[2 Field Quality Assessment 11](#_Toc209192727)

[2.1 Comprehensive Field Quality Analysis 11](#_Toc209192728)

[2.2 Quality Issues Identification 15](#_Toc209192729)

[3 Cross-Dataset Correlation Analysis 16](#_Toc209192730)

[3.1 Calculate Average Correlation Matrix 16](#_Toc209192731)

[4 Feature Importance & Variance Analysis 20](#_Toc209192732)

[4.1 Variance and Information Content Analysis 20](#_Toc209192733)

[5 Field Importance Weighting 22](#_Toc209192734)

[5.1 Field Scoring and Recommendations 22](#_Toc209192735)

[6 Dataset Quality Comparison 26](#_Toc209192736)

[6.1 Basic Dataset Rankings 26](#_Toc209192737)

[7 Summary and Recommendations 27](#_Toc209192738)

[7.1 Key Findings Summary 27](#_Toc209192739)

[7.2 Final Dataset and Field Selection Decisions 29](#_Toc209192740)

[7.2.1 DATASETS TO KEEP (7 datasets - 609 total records) 29](#_Toc209192741)

[7.2.2 DATASETS TO DROP (6 datasets - 298 total records) 30](#_Toc209192742)

[7.2.3 FIELDS TO KEEP (11 fields) 30](#_Toc209192743)

[7.2.4 FIELDS TO DROP (24 fields) 30](#_Toc209192744)

[7.2.5 Summary Statistics 31](#_Toc209192745)

[7.2.6 Implementation for Task 3 31](#_Toc209192746)

[Data Cleaning Implementation 32](#_Toc209192747)

# Introduction

This report aims to document the data-preparation and standardization of this milestone. We will be using the 13 data sets that are related to the health and demographics in South Africa. These include access to health care, child mortality and others.

The main objective of this milestone is to transform raw data into a clean, consistent, and analysis-ready format. This is critical because the quality of insights produced by any data science pipeline depends heavily on how well the input data has been prepared.

The goal is to:

* select relevant data sets
* verify data quality
* clean and transform data sets
* selecting attributes
* preparing the data for modelling

## 1.1 Data Selection

The datasets were chosen because of their:

* Relevance: They provide key insights into South Africa’s health and demographic patterns.
* Accessibility: They are publicly available, which ensures transparency and reproducibility.
* Manageability: The dataset sizes are reasonable, making them practical to clean and process.

Findings so far:

We deduced that the datasets have inconsistencies, missing values, and duplicates. These issues highlight the importance of robust data and how cleaning workflow is as equally important.

## 1.2 Verifying Data Quality

Prior cleaning, it is important to analyse the raw data to make note on where the issues lie.

Checking for Missing Values

**Guidelines for handling missing values:**

0% missing → Safe, no action needed.

1–10% missing → Impute or drop depending on importance.

20–30% missing → Requires a careful decision.

40% missing → Likely better to drop the column

**Checking for Duplicates**

Duplicates can skew results and inflate sample sizes artificially.

**Checking Numeric Correlations**

Correlations allow us to see how variables are related. This step is particularly useful for feature selection and detecting multicollinearity (a problem for regression models).

## 1.3 Data Cleaning

Once issues are identified, we move on to cleaning the datasets.

1. Removing Duplicates
2. Handling Missing Values

Different strategies are applied based on data type:

Numeric → Imputed using the median (robust against outliers).

Categorical → Replaced with "Unknown".

Boolean → Replaced with the most frequent (modal) value.

1. Diagnosing Cleaned Data

To verify that cleaning worked, we run a diagnostic function. This provides structure, previews, and summaries.

## 1.4 Encoding and Scaling

Normalizing Numeric Variables

Scaling ensures that all numeric fields are on the same scale. This is especially important for distance-based models (e.g., clustering, k-NN) and algorithms sensitive to magnitude (e.g., gradient descent in regression/ML).

## 1.5 Train-Test Split

Finally, we prepare the datasets for modelling by splitting them into training and testing subsets. The training set is used to build models, while the testing set evaluates performance on unseen data.

We use a 70/30 split with createDataPartition() from the caret package.

In this milestone, we successfully:

Imported and explored three datasets related to health and demographics in South Africa.

Assessed their quality by checking for missing values, duplicates, and correlations.

Cleaned the data systematically through duplicate removal, missing value imputation, and column pruning.

Encoded categorical variables and normalized numeric fields to ensure model compatibility.

Prepared the data for modelling by splitting into training and testing sets.

This establishes a strong foundation for the next milestone, where we will begin applying statistical models and machine learning algorithms to uncover patterns and generate insights.

# Verify Data Quality - Statistical Validation and Field Selections

# 1 Data Preparation & Standardization

## 1.1 Setup Global Variables

# Auto-detect environment and set paths  
current\_dir <- basename(getwd())  
if (current\_dir == "Task\_02") {  
 # Running in RStudio (current directory is Task\_02)  
 base\_path <- "../../Data/01\_Raw"  
 outputs\_path <- "outputs"  
 cat("Environment: RStudio\n")  
} else {  
 # Running in VS Code (from project root)  
 base\_path <- "02\_Project/Data/01\_Raw"  
 outputs\_path <- "02\_Project/Milestone\_2/Task\_02/outputs"  
 cat("Environment: VS Code\n")  
}

## Environment: RStudio

# Create outputs directory  
if (!dir.exists(outputs\_path)) {  
 dir.create(outputs\_path, recursive = TRUE)  
 cat("Created outputs directory:", outputs\_path, "\n")  
}  
  
# Quality thresholds  
MISSING\_THRESHOLD <- 0.5 # 50% missing data  
OUTLIER\_THRESHOLD <- 0.05 # 5% outliers  
LOW\_VARIANCE\_THRESHOLD <- 0.01 # CV < 0.01  
  
cat("Global variables set successfully\n")

## Global variables set successfully

cat("Base path:", base\_path, "\n")

## Base path: ../../Data/01\_Raw

cat("Outputs path:", outputs\_path, "\n")

## Outputs path: outputs

# Check if directory exists and show contents  
if (dir.exists(base\_path)) {  
 cat("Directory exists:", base\_path, "\n\n")  
  
 # List all files and folders in the directory  
 all\_items <- list.files(base\_path, full.names = FALSE)  
 cat("Contents of", base\_path, ":\n")  
  
 if (length(all\_items) > 0) {  
 for (item in all\_items) {  
 item\_path <- file.path(base\_path, item)  
 if (dir.exists(item\_path)) {  
 cat(" [DIR] ", item, "\n")  
 } else {  
 cat(" [FILE] ", item, "\n")  
 }  
 }  
 } else {  
 cat(" Directory is empty\n")  
 }  
} else {  
 cat("Directory does not exist:", base\_path, "\n")  
}

## Directory exists: ../../Data/01\_Raw   
##   
## Contents of ../../Data/01\_Raw :  
## [FILE] access-to-health-care\_national\_zaf.csv   
## [FILE] anthropometry\_national\_zaf.csv   
## [FILE] child-mortality-rates\_national\_zaf.csv   
## [FILE] covid-19-prevention\_national\_zaf.csv   
## [FILE] dhs-quickstats\_national\_zaf.csv   
## [FILE] hiv-behavior\_national\_zaf.csv   
## [FILE] immunization\_national\_zaf.csv   
## [FILE] iycf\_national\_zaf.csv   
## [FILE] literacy\_national\_zaf.csv   
## [FILE] maternal-mortality\_national\_zaf.csv   
## [FILE] symptoms-of-acute-respiratory-infection-ari\_national\_zaf.csv   
## [FILE] toilet-facilities\_national\_zaf.csv   
## [FILE] water\_national\_zaf.csv

## 1.2 Load All Datasets

# List all CSV files  
csv\_files <- list.files(base\_path, pattern = "\\.csv$", full.names = TRUE)  
cat("Found", length(csv\_files), "CSV files:\n")

## Found 13 CSV files:

for(file in csv\_files) {  
 cat("- ", basename(file), "\n")  
}

## - access-to-health-care\_national\_zaf.csv   
## - anthropometry\_national\_zaf.csv   
## - child-mortality-rates\_national\_zaf.csv   
## - covid-19-prevention\_national\_zaf.csv   
## - dhs-quickstats\_national\_zaf.csv   
## - hiv-behavior\_national\_zaf.csv   
## - immunization\_national\_zaf.csv   
## - iycf\_national\_zaf.csv   
## - literacy\_national\_zaf.csv   
## - maternal-mortality\_national\_zaf.csv   
## - symptoms-of-acute-respiratory-infection-ari\_national\_zaf.csv   
## - toilet-facilities\_national\_zaf.csv   
## - water\_national\_zaf.csv

# Function to read CSV with proper header handling  
read\_csv\_clean <- function(path) {  
 # Read the first line to get proper column names  
 headers <- readr::read\_lines(path, n\_max = 1)  
 col\_names <- unlist(strsplit(headers, ","))  
  
 # Skip the metadata line (line 2) and read with proper headers  
 readr::read\_csv(path, show\_col\_types = FALSE, skip = 2, col\_names = col\_names) %>%  
 janitor::clean\_names()  
}  
  
# Load all datasets  
datasets <- map(csv\_files, read\_csv\_clean)  
names(datasets) <- tools::file\_path\_sans\_ext(basename(csv\_files))  
  
# Remove any datasets with no rows  
datasets <- datasets[map\_int(datasets, nrow) > 0]  
  
cat("\nLoaded", length(datasets), "datasets with data\n")

##   
## Loaded 13 datasets with data

## 1.3 Standardize Field Structure

# Get all unique column names across datasets  
all\_columns <- unique(unlist(map(datasets, names)))  
cat("Total unique columns across all datasets:", length(all\_columns), "\n")

## Total unique columns across all datasets: 29

# Identify core fields present in all datasets  
common\_fields <- Reduce(intersect, map(datasets, names))  
cat("Common fields in all datasets:", length(common\_fields), "\n")

## Common fields in all datasets: 29

print(common\_fields)

## [1] "iso3" "data\_id"   
## [3] "indicator" "value"   
## [5] "precision" "dhs\_country\_code"   
## [7] "country\_name" "survey\_year"   
## [9] "survey\_id" "indicator\_id"   
## [11] "indicator\_order" "indicator\_type"   
## [13] "characteristic\_id" "characteristic\_order"   
## [15] "characteristic\_category" "characteristic\_label"   
## [17] "by\_variable\_id" "by\_variable\_label"   
## [19] "is\_total" "is\_preferred"   
## [21] "sdrid" "region\_id"   
## [23] "survey\_year\_label" "survey\_type"   
## [25] "denominator\_weighted" "denominator\_unweighted"   
## [27] "ci\_low" "ci\_high"   
## [29] "level\_rank"

# Dynamically categorize fields by analyzing their actual data types and content  
sample\_dataset <- datasets[[1]] # Use first dataset as reference  
  
# Define expected numeric fields based on data dictionary  
expected\_numeric\_fields <- c("value", "precision", "survey\_year", "indicator\_order",  
 "characteristic\_order", "denominator\_weighted",  
 "denominator\_unweighted", "ci\_low", "ci\_high", "level\_rank")  
  
# Detect numeric fields - combine expected fields with actual numeric detection  
numeric\_fields <- names(sample\_dataset)[sapply(sample\_dataset, function(x) {  
 # Check if it's already numeric  
 if (is.numeric(x)) return(TRUE)  
  
 # Check if it's character but contains only numeric values (including decimals and negatives)  
 if (is.character(x)) {  
 non\_na\_values <- x[!is.na(x) & x != ""]  
 if (length(non\_na\_values) == 0) return(FALSE)  
 return(all(grepl("^-?[0-9]\*\\.?[0-9]+([eE][+-]?[0-9]+)?$", non\_na\_values, perl = TRUE)))  
 }  
  
 return(FALSE)  
})]  
  
# Also include any expected numeric fields that might be in the data  
numeric\_fields <- unique(c(numeric\_fields,  
 intersect(tolower(names(sample\_dataset)), expected\_numeric\_fields),  
 intersect(names(sample\_dataset), expected\_numeric\_fields)))  
  
# Detect logical/boolean fields  
expected\_logical\_fields <- c("is\_total", "is\_preferred")  
logical\_fields <- names(sample\_dataset)[sapply(sample\_dataset, function(x) {  
 if (is.logical(x)) return(TRUE)  
 if (is.character(x)) {  
 non\_na\_values <- tolower(x[!is.na(x) & x != ""])  
 if (length(non\_na\_values) == 0) return(FALSE)  
 return(all(non\_na\_values %in% c("true", "false", "t", "f", "yes", "no", "1", "0")))  
 }  
 return(FALSE)  
})]  
  
# Also include expected logical fields  
logical\_fields <- unique(c(logical\_fields,  
 intersect(tolower(names(sample\_dataset)), expected\_logical\_fields),  
 intersect(names(sample\_dataset), expected\_logical\_fields)))  
  
# All remaining fields are categorical  
categorical\_fields <- setdiff(names(sample\_dataset), c(numeric\_fields, logical\_fields))  
  
cat("\nDynamic field categorization:\n")

##   
## Dynamic field categorization:

cat("Numeric fields (", length(numeric\_fields), "):", paste(numeric\_fields, collapse = ", "), "\n")

## Numeric fields ( 16 ): data\_id, value, precision, survey\_year, indicator\_order, characteristic\_id, characteristic\_order, by\_variable\_id, is\_total, is\_preferred, survey\_year\_label, denominator\_weighted, denominator\_unweighted, ci\_low, ci\_high, level\_rank

cat("Logical fields (", length(logical\_fields), "):", paste(logical\_fields, collapse = ", "), "\n")

## Logical fields ( 6 ): region\_id, ci\_low, ci\_high, level\_rank, is\_total, is\_preferred

cat("Categorical fields (", length(categorical\_fields), "):", paste(categorical\_fields, collapse = ", "), "\n")

## Categorical fields ( 12 ): iso3, indicator, dhs\_country\_code, country\_name, survey\_id, indicator\_id, indicator\_type, characteristic\_category, characteristic\_label, by\_variable\_label, sdrid, survey\_type

# Create standardized dataset summary  
dataset\_summary <- map\_dfr(datasets, function(df) {  
 tibble(  
 total\_rows = nrow(df),  
 total\_cols = ncol(df),  
 numeric\_cols = sum(names(df) %in% numeric\_fields),  
 categorical\_cols = sum(names(df) %in% categorical\_fields),  
 logical\_cols = sum(names(df) %in% logical\_fields),  
 missing\_cells = sum(is.na(df)),  
 missing\_pct = round(100 \* sum(is.na(df)) / (nrow(df) \* ncol(df)), 2)  
 )  
}, .id = "dataset")  
  
# Display summary  
gt(dataset\_summary) %>%  
 tab\_header(title = "Standardized Dataset Summary")

Table 1: Standardized Dataset Summary

| dataset | total\_rows | total\_cols | numeric\_cols | categorical\_cols | logical\_cols | missing\_cells | missing\_pct |
| --- | --- | --- | --- | --- | --- | --- | --- |
| access-to-health-care\_national\_zaf | 275 | 29 | 16 | 12 | 6 | 1181 | 14.81 |
| anthropometry\_national\_zaf | 37 | 29 | 16 | 12 | 6 | 193 | 17.99 |
| child-mortality-rates\_national\_zaf | 40 | 29 | 16 | 12 | 6 | 192 | 16.55 |
| covid-19-prevention\_national\_zaf | 34 | 29 | 16 | 12 | 6 | 174 | 17.65 |
| dhs-quickstats\_national\_zaf | 52 | 29 | 16 | 12 | 6 | 249 | 16.51 |
| hiv-behavior\_national\_zaf | 118 | 29 | 16 | 12 | 6 | 667 | 19.49 |
| immunization\_national\_zaf | 116 | 29 | 16 | 12 | 6 | 536 | 15.93 |
| iycf\_national\_zaf | 22 | 29 | 16 | 12 | 6 | 114 | 17.87 |
| literacy\_national\_zaf | 20 | 29 | 16 | 12 | 6 | 104 | 17.93 |
| maternal-mortality\_national\_zaf | 21 | 29 | 16 | 12 | 6 | 133 | 21.84 |
| symptoms-of-acute-respiratory-infection-ari\_national\_zaf | 26 | 29 | 16 | 12 | 6 | 120 | 15.92 |
| toilet-facilities\_national\_zaf | 46 | 29 | 16 | 12 | 6 | 238 | 17.84 |
| water\_national\_zaf | 100 | 29 | 16 | 12 | 6 | 508 | 17.52 |

# Export summary  
write\_csv(dataset\_summary, file.path(outputs\_path, "standardized\_datasets\_summary.csv"))  
cat("Exported:", file.path(outputs\_path, "standardized\_datasets\_summary.csv"), "\n")

## Exported: outputs/standardized\_datasets\_summary.csv

# 2 Field Quality Assessment

## 2.1 Comprehensive Field Quality Analysis

# Function to assess field quality across multiple dimensions  
assess\_field\_quality <- function(df, dataset\_name) {  
 all\_cols <- names(df)  
  
 map\_dfr(all\_cols, function(col) {  
 values <- df[[col]]  
 non\_missing <- values[!is.na(values)]  
  
 # Basic metrics  
 total\_count <- length(values)  
 missing\_count <- sum(is.na(values))  
 missing\_rate <- missing\_count / total\_count  
 unique\_count <- length(unique(non\_missing))  
 unique\_rate <- unique\_count / length(non\_missing)  
  
 # Initialize result  
 result <- tibble(  
 dataset = dataset\_name,  
 field = col,  
 total\_count = total\_count,  
 missing\_count = missing\_count,  
 missing\_rate = missing\_rate,  
 unique\_count = unique\_count,  
 unique\_rate = unique\_rate  
 )  
  
 # Field type classification  
 is\_numeric <- is.numeric(values)  
 is\_categorical <- !is\_numeric  
  
 if (is\_numeric && length(non\_missing) > 0) {  
 # Numeric field quality metrics  
 q25 <- quantile(non\_missing, 0.25, na.rm = TRUE)  
 q75 <- quantile(non\_missing, 0.75, na.rm = TRUE)  
 iqr <- q75 - q25  
 outlier\_threshold\_low <- q25 - 1.5 \* iqr  
 outlier\_threshold\_high <- q75 + 1.5 \* iqr  
 outliers <- sum(non\_missing < outlier\_threshold\_low | non\_missing > outlier\_threshold\_high)  
 outlier\_rate <- outliers / length(non\_missing)  
  
 result <- result %>%  
 mutate(  
 field\_type = "numeric",  
 mean\_value = mean(non\_missing, na.rm = TRUE),  
 std\_dev = sd(non\_missing, na.rm = TRUE),  
 coefficient\_of\_variation = std\_dev / abs(mean\_value),  
 skewness = moments::skewness(non\_missing),  
 kurtosis = moments::kurtosis(non\_missing),  
 outlier\_count = outliers,  
 outlier\_rate = outlier\_rate,  
 min\_value = min(non\_missing, na.rm = TRUE),  
 max\_value = max(non\_missing, na.rm = TRUE)  
 )  
 } else if (is\_categorical && length(non\_missing) > 0) {  
 # Categorical field quality metrics  
 value\_counts <- table(non\_missing)  
 max\_frequency <- max(value\_counts)  
 mode\_frequency\_rate <- max\_frequency / length(non\_missing)  
 rare\_categories <- sum(value\_counts <= 5)  
 cardinality\_ratio <- unique\_count / length(non\_missing)  
  
 result <- result %>%  
 mutate(  
 field\_type = "categorical",  
 mean\_value = NA\_real\_,  
 std\_dev = NA\_real\_,  
 coefficient\_of\_variation = NA\_real\_,  
 skewness = NA\_real\_,  
 kurtosis = NA\_real\_,  
 outlier\_count = NA\_integer\_,  
 outlier\_rate = NA\_real\_,  
 max\_frequency = max\_frequency,  
 mode\_frequency\_rate = mode\_frequency\_rate,  
 rare\_categories = rare\_categories,  
 cardinality\_ratio = cardinality\_ratio  
 )  
 } else {  
 # Empty or all-missing field  
 result <- result %>%  
 mutate(  
 field\_type = ifelse(is\_numeric, "numeric", "categorical"),  
 mean\_value = NA\_real\_,  
 std\_dev = NA\_real\_,  
 coefficient\_of\_variation = NA\_real\_,  
 skewness = NA\_real\_,  
 kurtosis = NA\_real\_,  
 outlier\_count = NA\_integer\_,  
 outlier\_rate = NA\_real\_  
 )  
 }  
  
 return(result)  
 })  
}  
  
# Run field quality assessment on all datasets  
field\_quality\_results <- map2\_dfr(datasets, names(datasets), assess\_field\_quality)  
  
# Summarize quality by field across datasets  
field\_quality\_summary <- field\_quality\_results %>%  
 group\_by(field, field\_type) %>%  
 summarise(  
 datasets\_present = n(),  
 avg\_missing\_rate = mean(missing\_rate, na.rm = TRUE),  
 avg\_unique\_rate = mean(unique\_rate, na.rm = TRUE),  
 avg\_outlier\_rate = mean(outlier\_rate, na.rm = TRUE),  
 avg\_cv = mean(coefficient\_of\_variation, na.rm = TRUE),  
 avg\_skewness = mean(abs(skewness), na.rm = TRUE),  
 high\_cardinality\_issues = sum(cardinality\_ratio > 0.8, na.rm = TRUE),  
 .groups = "drop"  
 ) %>%  
 arrange(field\_type, desc(avg\_missing\_rate))  
  
gt(field\_quality\_summary %>% head(20)) %>%  
 tab\_header(title = "Field Quality Summary (Top 20 by Missing Rate)") %>%  
 fmt\_number(columns = c(avg\_missing\_rate, avg\_unique\_rate, avg\_outlier\_rate,  
 avg\_cv, avg\_skewness), decimals = 3)

Table 2: Field Quality Summary (Top 20 by Missing Rate)

| field | field\_type | datasets\_present | avg\_missing\_rate | avg\_unique\_rate | avg\_outlier\_rate | avg\_cv | avg\_skewness | high\_cardinality\_issues |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ci\_high | categorical | 10 | 1.000 | NaN | NaN | NaN | NaN | 0 |
| ci\_low | categorical | 10 | 1.000 | NaN | NaN | NaN | NaN | 0 |
| level\_rank | categorical | 13 | 1.000 | NaN | NaN | NaN | NaN | 0 |
| region\_id | categorical | 13 | 1.000 | NaN | NaN | NaN | NaN | 0 |
| by\_variable\_label | categorical | 13 | 0.743 | 0.097 | NaN | NaN | NaN | 0 |
| characteristic\_category | categorical | 13 | 0.000 | 0.042 | NaN | NaN | NaN | 0 |
| characteristic\_label | categorical | 13 | 0.000 | 0.042 | NaN | NaN | NaN | 0 |
| country\_name | categorical | 13 | 0.000 | 0.026 | NaN | NaN | NaN | 0 |
| dhs\_country\_code | categorical | 13 | 0.000 | 0.026 | NaN | NaN | NaN | 0 |
| indicator | categorical | 13 | 0.000 | 0.561 | NaN | NaN | NaN | 2 |
| indicator\_id | categorical | 13 | 0.000 | 0.601 | NaN | NaN | NaN | 3 |
| indicator\_type | categorical | 13 | 0.000 | 0.079 | NaN | NaN | NaN | 0 |
| iso3 | categorical | 13 | 0.000 | 0.026 | NaN | NaN | NaN | 0 |
| sdrid | categorical | 13 | 0.000 | 0.601 | NaN | NaN | NaN | 3 |
| survey\_id | categorical | 13 | 0.000 | 0.046 | NaN | NaN | NaN | 0 |
| survey\_type | categorical | 13 | 0.000 | 0.026 | NaN | NaN | NaN | 0 |
| ci\_high | numeric | 3 | 0.613 | 0.810 | 0.071 | 0.883 | 0.983 | 0 |
| ci\_low | numeric | 3 | 0.613 | 0.843 | 0.048 | 0.781 | 1.032 | 0 |
| denominator\_weighted | numeric | 13 | 0.267 | 0.345 | 0.040 | 0.619 | 0.687 | 0 |
| denominator\_unweighted | numeric | 13 | 0.251 | 0.296 | 0.045 | 0.619 | 0.697 | 0 |

# Export field quality results  
write\_csv(field\_quality\_results, file.path(outputs\_path, "field\_quality\_assessment.csv"))  
write\_csv(field\_quality\_summary, file.path(outputs\_path, "field\_quality\_summary.csv"))  
cat("Exported: field quality assessment results\n")

## Exported: field quality assessment results

## 2.2 Quality Issues Identification

# Function to flag data quality issues  
flag\_quality\_issues <- function(field\_quality\_data) {  
 field\_quality\_data %>%  
 mutate(  
 # Flag high missing data  
 high\_missing\_flag = missing\_rate > MISSING\_THRESHOLD,  
  
 # Flag excessive outliers (numeric fields only)  
 excessive\_outliers\_flag = !is.na(outlier\_rate) & outlier\_rate > OUTLIER\_THRESHOLD,  
  
 # Flag low variance (numeric fields only)  
 low\_variance\_flag = !is.na(coefficient\_of\_variation) &  
 coefficient\_of\_variation < LOW\_VARIANCE\_THRESHOLD,  
  
 # Calculate total issue count per field  
 total\_issues = as.numeric(high\_missing\_flag) +  
 as.numeric(excessive\_outliers\_flag) +  
 as.numeric(low\_variance\_flag)  
 ) %>%  
 # Create overall quality rating  
 mutate(  
 quality\_rating = case\_when(  
 total\_issues == 0 ~ "Good Quality",  
 total\_issues == 1 ~ "Moderate Issues",  
 total\_issues >= 2 ~ "Significant Issues"  
 )  
 )  
}  
  
# Apply quality flagging  
flagged\_quality <- flag\_quality\_issues(field\_quality\_results)  
  
# Summarize issues by field across all datasets  
issue\_summary\_by\_field <- flagged\_quality %>%  
 group\_by(field, field\_type) %>%  
 summarise(  
 datasets\_present = n(),  
 high\_missing\_datasets = sum(high\_missing\_flag),  
 excessive\_outliers\_datasets = sum(excessive\_outliers\_flag, na.rm = TRUE),  
 low\_variance\_datasets = sum(low\_variance\_flag, na.rm = TRUE),  
 avg\_issues\_per\_dataset = mean(total\_issues),  
 .groups = "drop"  
 ) %>%  
 arrange(desc(avg\_issues\_per\_dataset), desc(high\_missing\_datasets))  
  
gt(issue\_summary\_by\_field %>% head(15)) %>%  
 tab\_header(title = "Data Quality Issues Summary by Field (Top 15 Problematic)") %>%  
 fmt\_number(columns = avg\_issues\_per\_dataset, decimals = 2)

Table 3: Data Quality Issues Summary by Field (Top 15 Problematic)

| field | field\_type | datasets\_present | high\_missing\_datasets | excessive\_outliers\_datasets | low\_variance\_datasets | avg\_issues\_per\_dataset |
| --- | --- | --- | --- | --- | --- | --- |
| survey\_year | numeric | 13 | 0 | 1 | 13 | 1.08 |
| survey\_year\_label | numeric | 13 | 0 | 1 | 13 | 1.08 |
| level\_rank | categorical | 13 | 13 | 0 | 0 | 1.00 |
| region\_id | categorical | 13 | 13 | 0 | 0 | 1.00 |
| ci\_high | categorical | 10 | 10 | 0 | 0 | 1.00 |
| ci\_low | categorical | 10 | 10 | 0 | 0 | 1.00 |
| ci\_high | numeric | 3 | 2 | 1 | 0 | 1.00 |
| ci\_low | numeric | 3 | 2 | 1 | 0 | 1.00 |
| is\_total | numeric | 13 | 0 | 0 | 13 | 1.00 |
| indicator\_order | numeric | 13 | 0 | 3 | 9 | 0.92 |
| is\_preferred | numeric | 13 | 0 | 2 | 8 | 0.77 |
| by\_variable\_label | categorical | 13 | 9 | 0 | 0 | 0.69 |
| precision | numeric | 13 | 0 | 8 | 1 | 0.69 |
| value | numeric | 13 | 0 | 9 | 0 | 0.69 |
| denominator\_unweighted | numeric | 13 | 2 | 5 | 0 | 0.54 |

# Export quality issues data  
write\_csv(flagged\_quality, file.path(outputs\_path, "flagged\_quality\_issues.csv"))  
write\_csv(issue\_summary\_by\_field, file.path(outputs\_path, "quality\_issues\_by\_field.csv"))  
cat("Exported: data quality issues flagging results\n")

## Exported: data quality issues flagging results

# 3 Cross-Dataset Correlation Analysis

## 3.1 Calculate Average Correlation Matrix

# Calculate correlation matrices for each dataset  
all\_correlations <- map2\_dfr(datasets, names(datasets), function(df, dataset\_name) {  
 # Select numeric fields  
 numeric\_cols <- intersect(names(df), numeric\_fields)  
 numeric\_data <- df %>% select(all\_of(numeric\_cols))  
  
 # Convert to numeric and remove columns with no variance  
 numeric\_data <- numeric\_data %>%  
 mutate(across(everything(), as.numeric)) %>%  
 select(where(function(x) {  
 variance <- var(x, na.rm = TRUE)  
 !is.na(variance) && variance > 0  
 }))  
  
 if (ncol(numeric\_data) < 2) return(tibble())  
  
 # Calculate correlation matrix  
 cor\_matrix <- cor(numeric\_data, use = "pairwise.complete.obs")  
  
 # Convert to long format  
 as\_tibble(cor\_matrix, rownames = "field1") %>%  
 pivot\_longer(-field1, names\_to = "field2", values\_to = "correlation") %>%  
 filter(field1 != field2) %>%  
 mutate(dataset = dataset\_name)  
})  
  
cat("Calculated correlations for", length(unique(all\_correlations$dataset)), "datasets\n")

## Calculated correlations for 13 datasets

# Export all correlations  
write\_csv(all\_correlations, file.path(outputs\_path, "all\_dataset\_correlations.csv"))  
  
# Create summary statistics across datasets  
correlation\_summary <- all\_correlations %>%  
 group\_by(field1, field2) %>%  
 summarise(  
 datasets\_present = n(),  
 avg\_correlation = mean(correlation, na.rm = TRUE),  
 abs\_avg\_correlation = mean(abs(correlation), na.rm = TRUE),  
 .groups = "drop"  
 ) %>%  
 filter(datasets\_present >= 2) %>%  
 arrange(desc(abs\_avg\_correlation))  
  
gt(correlation\_summary %>% head(15)) %>%  
 tab\_header(title = "Top Field Correlations Across Datasets") %>%  
 fmt\_number(columns = c(avg\_correlation, abs\_avg\_correlation), decimals = 3)

Table 4: Top Field Correlations Across Datasets

| field1 | field2 | datasets\_present | avg\_correlation | abs\_avg\_correlation |
| --- | --- | --- | --- | --- |
| survey\_year | survey\_year\_label | 11 | 1.000 | 1.000 |
| survey\_year\_label | survey\_year | 11 | 1.000 | 1.000 |
| ci\_high | value | 3 | 0.999 | 0.999 |
| value | ci\_high | 3 | 0.999 | 0.999 |
| denominator\_unweighted | denominator\_weighted | 13 | 0.998 | 0.998 |
| denominator\_weighted | denominator\_unweighted | 13 | 0.998 | 0.998 |
| ci\_low | value | 3 | 0.997 | 0.997 |
| value | ci\_low | 3 | 0.997 | 0.997 |
| ci\_high | ci\_low | 3 | 0.993 | 0.993 |
| ci\_low | ci\_high | 3 | 0.993 | 0.993 |
| characteristic\_id | characteristic\_order | 7 | 0.972 | 0.972 |
| characteristic\_order | characteristic\_id | 7 | 0.972 | 0.972 |
| by\_variable\_id | characteristic\_id | 3 | 0.101 | 0.754 |
| characteristic\_id | by\_variable\_id | 3 | 0.101 | 0.754 |
| characteristic\_order | indicator\_order | 7 | -0.426 | 0.738 |

write\_csv(correlation\_summary, file.path(outputs\_path, "correlation\_summary.csv"))  
  
# Field-level correlation rankings  
field\_rankings <- all\_correlations %>%  
 group\_by(field1) %>%  
 summarise(  
 avg\_abs\_correlation = mean(abs(correlation), na.rm = TRUE),  
 max\_abs\_correlation = max(abs(correlation), na.rm = TRUE),  
 high\_correlations\_count = sum(abs(correlation) > 0.7, na.rm = TRUE),  
 .groups = "drop"  
 ) %>%  
 arrange(desc(avg\_abs\_correlation))  
  
gt(field\_rankings %>% head(10)) %>%  
 tab\_header(title = "Field Correlation Rankings") %>%  
 fmt\_number(columns = c(avg\_abs\_correlation, max\_abs\_correlation), decimals = 3)

Table 5: Field Correlation Rankings

| field1 | avg\_abs\_correlation | max\_abs\_correlation | high\_correlations\_count |
| --- | --- | --- | --- |
| denominator\_weighted | 0.452 | 1.000 | 25 |
| denominator\_unweighted | 0.447 | 1.000 | 26 |
| ci\_low | 0.445 | 1.000 | 8 |
| ci\_high | 0.421 | 1.000 | 8 |
| characteristic\_id | 0.379 | 1.000 | 12 |
| characteristic\_order | 0.378 | 1.000 | 12 |
| indicator\_order | 0.345 | 1.000 | 20 |
| survey\_year | 0.339 | 1.000 | 21 |
| survey\_year\_label | 0.339 | 1.000 | 21 |
| data\_id | 0.317 | 1.000 | 12 |

write\_csv(field\_rankings, file.path(outputs\_path, "field\_correlation\_rankings.csv"))  
  
# High correlation pairs  
high\_corr\_pairs <- all\_correlations %>%  
 filter(abs(correlation) > 0.8) %>%  
 arrange(desc(abs(correlation)))  
  
if (nrow(high\_corr\_pairs) > 0) {  
 gt(high\_corr\_pairs %>% head(15)) %>%  
 tab\_header(title = "Highly Correlated Field Pairs (|r| > 0.8)") %>%  
 fmt\_number(columns = correlation, decimals = 3)  
  
 write\_csv(high\_corr\_pairs, file.path(outputs\_path, "high\_correlation\_pairs.csv"))  
}  
  
cat("Exported correlation analysis results\n")

## Exported correlation analysis results

# 4 Feature Importance & Variance Analysis

## 4.1 Variance and Information Content Analysis

# Function to calculate variance metrics for each dataset  
calc\_variance\_metrics <- function(df, dataset\_name) {  
 numeric\_cols <- intersect(names(df), numeric\_fields)  
  
 map\_dfr(numeric\_cols, function(col) {  
 values <- df[[col]][!is.na(df[[col]])]  
  
 if (length(values) < 2) {  
 return(tibble(  
 dataset = dataset\_name,  
 field = col,  
 variance = NA,  
 coefficient\_of\_variation = NA,  
 range\_normalized = NA,  
 unique\_values = length(unique(values)),  
 information\_content = NA  
 ))  
 }  
  
 # Calculate various variance metrics  
 var\_val <- var(values)  
 mean\_val <- mean(values)  
 cv <- if (mean\_val != 0) sd(values) / abs(mean\_val) else NA  
 range\_norm <- (max(values) - min(values)) / (abs(max(values)) + abs(min(values)) + 1e-10)  
 unique\_vals <- length(unique(values))  
  
 # Information content (entropy-like measure)  
 if (unique\_vals > 1) {  
 value\_counts <- table(cut(values, breaks = min(unique\_vals, 20)))  
 proportions <- value\_counts / sum(value\_counts)  
 proportions <- proportions[proportions > 0]  
 info\_content <- -sum(proportions \* log2(proportions))  
 } else {  
 info\_content <- 0  
 }  
  
 tibble(  
 dataset = dataset\_name,  
 field = col,  
 variance = var\_val,  
 coefficient\_of\_variation = cv,  
 range\_normalized = range\_norm,  
 unique\_values = unique\_vals,  
 information\_content = info\_content  
 )  
 })  
}  
  
# Calculate variance metrics for all datasets  
variance\_results <- map2\_dfr(datasets, names(datasets), calc\_variance\_metrics)  
  
# Summarize variance by field  
variance\_summary <- variance\_results %>%  
 group\_by(field) %>%  
 summarise(  
 datasets\_analyzed = n(),  
 avg\_variance = mean(variance, na.rm = TRUE),  
 avg\_cv = mean(coefficient\_of\_variation, na.rm = TRUE),  
 avg\_range\_norm = mean(range\_normalized, na.rm = TRUE),  
 avg\_unique\_values = mean(unique\_values, na.rm = TRUE),  
 avg\_information\_content = mean(information\_content, na.rm = TRUE),  
 .groups = "drop"  
 ) %>%  
 arrange(desc(avg\_information\_content))  
  
gt(variance\_summary) %>%  
 tab\_header(title = "Feature Variance and Information Content Summary") %>%  
 fmt\_number(columns = c(avg\_variance, avg\_cv, avg\_range\_norm,  
 avg\_unique\_values, avg\_information\_content), decimals = 3)

Table 6: Feature Variance and Information Content Summary

| field | datasets\_analyzed | avg\_variance | avg\_cv | avg\_range\_norm | avg\_unique\_values | avg\_information\_content |
| --- | --- | --- | --- | --- | --- | --- |
| data\_id | 13 | 62,405,229,842.875 | 0.557 | 0.837 | 69.769 | 2.587 |
| ci\_low | 13 | 5,711.677 | 0.781 | 0.775 | 2.615 | 2.083 |
| denominator\_unweighted | 13 | 143,468,320.370 | 0.619 | 0.672 | 10.615 | 1.864 |
| denominator\_weighted | 13 | 195,225,185.470 | 0.619 | 0.663 | 10.769 | 1.850 |
| ci\_high | 13 | 54,888.993 | 0.883 | 0.755 | 2.385 | 1.809 |
| indicator\_order | 13 | 292,508,459,485,773.688 | 0.060 | 0.080 | 36.308 | 1.806 |
| value | 13 | 142,970,169.023 | 1.997 | 0.987 | 54.231 | 1.690 |
| survey\_year | 13 | 65.300 | 0.004 | 0.004 | 1.846 | 0.800 |
| survey\_year\_label | 13 | 65.300 | 0.004 | 0.004 | 1.846 | 0.800 |
| precision | 13 | 0.200 | 0.660 | 0.846 | 2.000 | 0.622 |
| characteristic\_id | 13 | 3,116,050,688.867 | 0.548 | 0.470 | 1.615 | 0.506 |
| characteristic\_order | 13 | 1,495,524,749.110 | 1.176 | 0.538 | 1.615 | 0.491 |
| is\_preferred | 13 | 0.082 | 0.300 | 0.385 | 1.385 | 0.336 |
| by\_variable\_id | 13 | 1,877,488,258.196 | 1.467 | 0.308 | 2.000 | 0.262 |
| is\_total | 13 | 0.000 | 0.000 | 0.000 | 1.000 | 0.000 |
| level\_rank | 13 | NaN | NaN | NaN | 0.000 | NaN |

# Export results  
write\_csv(variance\_results, file.path(outputs\_path, "variance\_analysis\_results.csv"))  
write\_csv(variance\_summary, file.path(outputs\_path, "feature\_importance\_rankings.csv"))  
cat("Exported: variance analysis and feature importance rankings\n")

## Exported: variance analysis and feature importance rankings

# 5 Field Importance Weighting

## 5.1 Field Scoring and Recommendations

# Combine all analysis results for scoring  
field\_scores <- field\_quality\_summary %>%  
 select(field, field\_type, avg\_missing\_rate, avg\_cv) %>%  
 # Add correlation metrics  
 left\_join(  
 field\_rankings %>%  
 select(field1, avg\_abs\_correlation, high\_correlations\_count) %>%  
 rename(field = field1, high\_correlations = high\_correlations\_count),  
 by = "field"  
 ) %>%  
 # Add variance metrics (including information content)  
 left\_join(  
 variance\_summary %>%  
 select(field, avg\_information\_content),  
 by = "field"  
 ) %>%  
 # Add quality issue counts  
 left\_join(  
 issue\_summary\_by\_field %>%  
 select(field, avg\_issues\_per\_dataset),  
 by = "field"  
 ) %>%  
 # Replace NAs with appropriate values  
 mutate(  
 across(c(avg\_missing\_rate, avg\_cv, avg\_information\_content,  
 avg\_abs\_correlation, high\_correlations,  
 avg\_issues\_per\_dataset), ~ coalesce(.x, 0))  
 )  
  
# Normalize scores to 0-1 scale  
normalize\_score <- function(x) {  
 if (all(is.na(x)) || max(x, na.rm = TRUE) == min(x, na.rm = TRUE)) return(rep(0, length(x)))  
 (x - min(x, na.rm = TRUE)) / (max(x, na.rm = TRUE) - min(x, na.rm = TRUE))  
}  
  
field\_scores <- field\_scores %>%  
 mutate(  
 # Data Quality Scores (higher is better)  
 completeness\_score = normalize\_score(1 - avg\_missing\_rate),  
 variance\_score = normalize\_score(avg\_cv),  
 information\_score = normalize\_score(avg\_information\_content),  
  
 # Correlation (lower correlation is better)  
 uniqueness\_score = normalize\_score(-avg\_abs\_correlation),  
  
 # Quality Issues (fewer issues is better)  
 issue\_penalty = normalize\_score(-avg\_issues\_per\_dataset),  
  
 # Composite score (rebalanced without PCA)  
 composite\_score = (  
 0.35 \* completeness\_score + # Data completeness is crucial  
 0.25 \* information\_score + # Information content matters  
 0.20 \* variance\_score + # Variance indicates signal  
 0.15 \* uniqueness\_score + # Avoid redundant features  
 0.05 \* issue\_penalty # Penalize problematic fields  
 )  
 ) %>%  
 arrange(desc(composite\_score))  
  
# Create recommendation categories  
field\_scores <- field\_scores %>%  
 mutate(  
 recommendation = case\_when(  
 composite\_score >= 0.7 ~ "High Priority - Include",  
 composite\_score >= 0.5 ~ "Medium Priority - Consider",  
 composite\_score >= 0.3 ~ "Low Priority - Evaluate",  
 TRUE ~ "Consider Exclusion"  
 )  
 )  
  
# Display scoring results  
gt(field\_scores %>% head(20) %>%  
 select(field, field\_type, composite\_score, recommendation,  
 completeness\_score, information\_score, variance\_score)) %>%  
 tab\_header(title = "Field Importance Rankings (Top 20)") %>%  
 fmt\_number(columns = c(composite\_score, completeness\_score, information\_score, variance\_score),  
 decimals = 3)

Table 7: Field Importance Rankings (Top 20)

| field | field\_type | composite\_score | recommendation | completeness\_score | information\_score | variance\_score |
| --- | --- | --- | --- | --- | --- | --- |
| value | numeric | 0.785 | High Priority - Include | 1.000 | 0.653 | 1.000 |
| data\_id | numeric | 0.747 | High Priority - Include | 1.000 | 1.000 | 0.279 |
| by\_variable\_id | numeric | 0.634 | Medium Priority - Consider | 1.000 | 0.101 | 0.735 |
| precision | numeric | 0.591 | Medium Priority - Consider | 1.000 | 0.240 | 0.331 |
| characteristic\_order | numeric | 0.579 | Medium Priority - Consider | 1.000 | 0.190 | 0.589 |
| indicator\_order | numeric | 0.573 | Medium Priority - Consider | 1.000 | 0.698 | 0.030 |
| characteristic\_category | categorical | 0.550 | Medium Priority - Consider | 1.000 | 0.000 | 0.000 |
| characteristic\_label | categorical | 0.550 | Medium Priority - Consider | 1.000 | 0.000 | 0.000 |
| country\_name | categorical | 0.550 | Medium Priority - Consider | 1.000 | 0.000 | 0.000 |
| dhs\_country\_code | categorical | 0.550 | Medium Priority - Consider | 1.000 | 0.000 | 0.000 |
| indicator | categorical | 0.550 | Medium Priority - Consider | 1.000 | 0.000 | 0.000 |
| indicator\_id | categorical | 0.550 | Medium Priority - Consider | 1.000 | 0.000 | 0.000 |
| indicator\_type | categorical | 0.550 | Medium Priority - Consider | 1.000 | 0.000 | 0.000 |
| iso3 | categorical | 0.550 | Medium Priority - Consider | 1.000 | 0.000 | 0.000 |
| sdrid | categorical | 0.550 | Medium Priority - Consider | 1.000 | 0.000 | 0.000 |
| survey\_id | categorical | 0.550 | Medium Priority - Consider | 1.000 | 0.000 | 0.000 |
| survey\_type | categorical | 0.550 | Medium Priority - Consider | 1.000 | 0.000 | 0.000 |
| denominator\_unweighted | numeric | 0.531 | Medium Priority - Consider | 0.749 | 0.721 | 0.310 |
| denominator\_weighted | numeric | 0.522 | Medium Priority - Consider | 0.733 | 0.715 | 0.310 |
| is\_preferred | numeric | 0.509 | Medium Priority - Consider | 1.000 | 0.130 | 0.150 |

# Summary by recommendation category  
recommendation\_summary <- field\_scores %>%  
 count(recommendation, name = "field\_count") %>%  
 arrange(desc(field\_count))  
  
gt(recommendation\_summary) %>%  
 tab\_header(title = "Field Recommendation Summary")

Table 8: Field Recommendation Summary

| recommendation | field\_count |
| --- | --- |
| Medium Priority - Consider | 20 |
| Consider Exclusion | 7 |
| Low Priority - Evaluate | 6 |
| High Priority - Include | 2 |

# Export scoring results  
write\_csv(field\_scores, file.path(outputs\_path, "field\_importance\_scores.csv"))  
write\_csv(recommendation\_summary, file.path(outputs\_path, "field\_recommendation\_summary.csv"))  
cat("Exported: field importance scores and recommendations\n")

## Exported: field importance scores and recommendations

# 6 Dataset Quality Comparison

## 6.1 Basic Dataset Rankings

# Calculate basic dataset quality metrics  
dataset\_quality <- map\_dfr(names(datasets), function(dataset\_name) {  
 df <- datasets[[dataset\_name]]  
  
 tibble(  
 dataset = dataset\_name,  
 total\_fields = ncol(df),  
 total\_rows = nrow(df),  
 missing\_rate = sum(is.na(df)) / (nrow(df) \* ncol(df)),  
 quality\_score = (1 - missing\_rate) \* log10(total\_rows) / 6 # Simple quality metric  
 )  
}) %>%  
 arrange(desc(quality\_score))  
  
gt(dataset\_quality) %>%  
 tab\_header(title = "Dataset Quality Rankings") %>%  
 fmt\_number(columns = c(missing\_rate, quality\_score), decimals = 3)

Table 9: Dataset Quality Rankings

| dataset | total\_fields | total\_rows | missing\_rate | quality\_score |
| --- | --- | --- | --- | --- |
| access-to-health-care\_national\_zaf | 29 | 275 | 0.148 | 0.346 |
| immunization\_national\_zaf | 29 | 116 | 0.159 | 0.289 |
| hiv-behavior\_national\_zaf | 29 | 118 | 0.195 | 0.278 |
| water\_national\_zaf | 29 | 100 | 0.175 | 0.275 |
| dhs-quickstats\_national\_zaf | 29 | 52 | 0.165 | 0.239 |
| toilet-facilities\_national\_zaf | 29 | 46 | 0.178 | 0.228 |
| child-mortality-rates\_national\_zaf | 29 | 40 | 0.166 | 0.223 |
| anthropometry\_national\_zaf | 29 | 37 | 0.180 | 0.214 |
| covid-19-prevention\_national\_zaf | 29 | 34 | 0.176 | 0.210 |
| symptoms-of-acute-respiratory-infection-ari\_national\_zaf | 29 | 26 | 0.159 | 0.198 |
| iycf\_national\_zaf | 29 | 22 | 0.179 | 0.184 |
| literacy\_national\_zaf | 29 | 20 | 0.179 | 0.178 |
| maternal-mortality\_national\_zaf | 29 | 21 | 0.218 | 0.172 |

# Export dataset quality rankings  
write\_csv(dataset\_quality, file.path(outputs\_path, "dataset\_quality\_rankings.csv"))  
cat("Exported: dataset quality rankings\n")

## Exported: dataset quality rankings

# 7 Summary and Recommendations

## 7.1 Key Findings Summary

cat("=== TASK 2: DATA QUALITY VERIFICATION SUMMARY ===\n\n")

## === TASK 2: DATA QUALITY VERIFICATION SUMMARY ===

cat("DATASETS ANALYZED:", length(datasets), "\n")

## DATASETS ANALYZED: 13

cat("TOTAL UNIQUE FIELDS:", nrow(field\_quality\_summary), "\n")

## TOTAL UNIQUE FIELDS: 31

cat("COMMON FIELDS ACROSS ALL DATASETS:", length(common\_fields), "\n\n")

## COMMON FIELDS ACROSS ALL DATASETS: 29

cat("FIELD RECOMMENDATIONS:\n")

## FIELD RECOMMENDATIONS:

if (exists("field\_scores")) {  
 rec\_counts <- field\_scores %>% count(recommendation)  
 for (i in 1:nrow(rec\_counts)) {  
 cat("-", rec\_counts$recommendation[i], ":", rec\_counts$n[i], "fields\n")  
 }  
}

## - Consider Exclusion : 7 fields  
## - High Priority - Include : 2 fields  
## - Low Priority - Evaluate : 6 fields  
## - Medium Priority - Consider : 20 fields

cat("\nTOP 5 RECOMMENDED FIELDS FOR MODELING:\n")

##   
## TOP 5 RECOMMENDED FIELDS FOR MODELING:

if (exists("field\_scores")) {  
 top\_fields <- field\_scores %>% head(5)  
 for (i in 1:nrow(top\_fields)) {  
 cat(i, ".", top\_fields$field[i],  
 "(Score:", round(top\_fields$composite\_score[i], 3),  
 "- Type:", top\_fields$field\_type[i], ")\n")  
 }  
}

## 1 . value (Score: 0.785 - Type: numeric )  
## 2 . data\_id (Score: 0.747 - Type: numeric )  
## 3 . by\_variable\_id (Score: 0.634 - Type: numeric )  
## 4 . precision (Score: 0.591 - Type: numeric )  
## 5 . characteristic\_order (Score: 0.579 - Type: numeric )

cat("\nTOP 3 DATASETS FOR QUALITY:\n")

##   
## TOP 3 DATASETS FOR QUALITY:

if (exists("dataset\_quality")) {  
 top\_datasets <- dataset\_quality %>% head(3)  
 for (i in 1:nrow(top\_datasets)) {  
 cat(i, ".", top\_datasets$dataset[i],  
 "(Quality Score:", round(top\_datasets$quality\_score[i], 3), ")\n")  
 }  
}

## 1 . access-to-health-care\_national\_zaf (Quality Score: 0.346 )  
## 2 . immunization\_national\_zaf (Quality Score: 0.289 )  
## 3 . hiv-behavior\_national\_zaf (Quality Score: 0.278 )

cat("\nDATA QUALITY ISSUES SUMMARY:\n")

##   
## DATA QUALITY ISSUES SUMMARY:

if (exists("issue\_summary\_by\_field")) {  
 total\_issues <- sum(issue\_summary\_by\_field$high\_missing\_datasets > 0)  
 cat("- Fields with missing data issues:", total\_issues, "\n")  
  
 outlier\_issues <- sum(issue\_summary\_by\_field$excessive\_outliers\_datasets > 0)  
 cat("- Fields with outlier issues:", outlier\_issues, "\n")  
  
 low\_var\_issues <- sum(issue\_summary\_by\_field$low\_variance\_datasets > 0)  
 cat("- Fields with low variance issues:", low\_var\_issues, "\n")  
}

## - Fields with missing data issues: 9   
## - Fields with outlier issues: 13   
## - Fields with low variance issues: 9

cat("\nEXPORTED ANALYSIS FILES:\n")

##   
## EXPORTED ANALYSIS FILES:

output\_files <- list.files("outputs", pattern = "\\.csv$", full.names = FALSE)  
for (file in output\_files) {  
 cat("- outputs/", file, "\n")  
}

## - outputs/ all\_dataset\_correlations.csv   
## - outputs/ correlation\_summary.csv   
## - outputs/ dataset\_quality\_rankings.csv   
## - outputs/ feature\_importance\_rankings.csv   
## - outputs/ field\_correlation\_rankings.csv   
## - outputs/ field\_importance\_scores.csv   
## - outputs/ field\_quality\_assessment.csv   
## - outputs/ field\_quality\_summary.csv   
## - outputs/ field\_recommendation\_summary.csv   
## - outputs/ flagged\_quality\_issues.csv   
## - outputs/ high\_correlation\_pairs.csv   
## - outputs/ quality\_issues\_by\_field.csv   
## - outputs/ standardized\_datasets\_summary.csv   
## - outputs/ variance\_analysis\_results.csv

## 7.2 Final Dataset and Field Selection Decisions

Based on comprehensive analysis results, the following specific datasets and fields will be retained for modeling:

### 7.2.1 DATASETS TO KEEP (7 datasets - 609 total records)

**Tier 1 - Primary Datasets (3 datasets - 509 records):** 1. **access-to-health-care\_national\_zaf** - 275 records (Quality Score: 0.346) 2. **immunization\_national\_zaf** - 116 records (Quality Score: 0.289) 3. **hiv-behavior\_national\_zaf** - 118 records (Quality Score: 0.278)

**Tier 2 - Secondary Datasets (4 datasets - 238 records):** 4. **water\_national\_zaf** - 100 records (Quality Score: 0.275) 5. **dhs-quickstats\_national\_zaf** - 52 records (Quality Score: 0.239) 6. **toilet-facilities\_national\_zaf** - 46 records (Quality Score: 0.228) 7. **child-mortality-rates\_national\_zaf** - 40 records (Quality Score: 0.223)

### 7.2.2 DATASETS TO DROP (6 datasets - 298 total records)

**Rationale: Quality Score <0.20 or insufficient sample size:** 1. **maternal-mortality\_national\_zaf** - 21 records (Quality Score: 0.172) - Poorest quality, 21.8% missing rate 2. **anthropometry\_national\_zaf** - 37 records (Quality Score: 0.214) - Small sample, 18.0% missing rate 3. **covid-19-prevention\_national\_zaf** - 34 records (Quality Score: 0.210) - Small sample 4. **symptoms-of-acute-respiratory-infection-ari\_national\_zaf** - 26 records (Quality Score: 0.198) - Very small sample 5. **iycf\_national\_zaf** - 22 records (Quality Score: 0.184) - Very small sample 6. **literacy\_national\_zaf** - 20 records (Quality Score: 0.178) - Very small sample

### 7.2.3 FIELDS TO KEEP (11 fields)

**Essential Fields (2 fields):** - **value** (Score: 0.785) - Primary measurement values - **data\_id** (Score: 0.747) - Unique record identifier

**Core Analytical Fields (4 fields):** - **by\_variable\_id** (Score: 0.634) - Important grouping variable - **precision** (Score: 0.591) - Measurement precision indicator - **characteristic\_order** (Score: 0.579) - Demographic ordering - **indicator\_order** (Score: 0.573) - Indicator hierarchy

**Descriptive Fields (3 fields):** - **indicator** (Score: 0.55) - Health indicator description - **indicator\_type** (Score: 0.55) - Indicator categorization - **characteristic\_category** (Score: 0.55) - Demographic categories

**Sample Size Field (1 field):** - **denominator\_unweighted** (Score: 0.531) - Unweighted sample sizes (categorical pattern)

**Quality Flag (1 field):** - **is\_preferred** (Score: 0.509) - Preferred estimate indicator

### 7.2.4 FIELDS TO DROP (24 fields)

**Complete Exclusion (7 fields - Fatal quality issues):** - **region\_id** - 100% missing across all datasets - **level\_rank** - 100% missing across all datasets - **ci\_low** (categorical) - 100% missing in 10/13 datasets - **ci\_high** (categorical) - 100% missing in 10/13 datasets - **by\_variable\_label** - 74% missing data - **ci\_low** (numeric) - 61% missing, limited availability - **ci\_high** (numeric) - 61% missing, limited availability

**Redundancy Removal (6 fields):** - **survey\_year\_label** - Perfect duplicate of survey\_year (r=1.0) - **denominator\_weighted** - Near-perfect correlation with denominator\_unweighted (r=0.998) - **characteristic\_id** - High correlation with characteristic\_order - **characteristic\_label** - Redundant with characteristic\_category - **country\_name** - Constant value (South Africa) - **iso3** - Constant value (ZAF)

**Low Priority Exclusion (11 fields):** - **survey\_year** - Low variance (same values across datasets) - **is\_total** - Low variance, minimal analytical value - **dhs\_country\_code**, **sdrid**, **survey\_id**, **survey\_type** - Administrative fields - **indicator\_id** - Redundant with indicator

### 7.2.5 Summary Statistics

**Final Dataset Composition:** - **Total Records**: 609 (67% of original 907 records) - **Total Fields**: 11 (35% of original 31 fields) - **Average Missing Rate**: 16.8% (improvement from 18.1% overall) - **Data Quality Score**: Weighted average of 0.271 (vs 0.229 for all datasets)

**Quality Improvements:** - Eliminated 100% missing fields (4 fields) - Removed high-missing datasets (6 datasets with <40 records or >20% missing) - Eliminated perfect redundancies (6 field pairs) - Retained high-information content fields (Score >0.5)

### 7.2.6 Implementation for Task 3

**Data Cleaning Priorities:** 1. Load only the 7 selected datasets 2. Select only the 11 specified fields 3. Address outliers in precision, value, and indicator\_order fields 4. Handle remaining missing data in denominator\_weighted 5. Validate data types and ranges for all retained fields

# Data Cleaning Implementation

The following unified R pipeline was executed to operationalize the field and dataset selection decisions. It integrates data cleanup, with an additional step to drop the first row of each dataset. Key steps included:

* Standardization of column names
* Dropping redundant and low-quality fields
* Retaining only the 11 curated fields
* Removing duplicates and completely empty columns
* Coercing numeric-looking character fields
* Outlier capping at the 1st and 99th percentiles
* Imputation strategies:
  + Mode for categorical fields
  + Median for numeric fields
  + Advanced imputation with MICE (PMM method) for residual missing values

Export of cleaned datasets

**First 9 rows of cleaned access-to-health-care\_national\_zaf:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **value** | **data\_id** | **by\_variable\_id** | **precision** | **characteristic\_order** | **indicator\_order** | **indicator** | **indicator\_type** | **characteristic\_category** | **denominator\_unweighted** | **is\_preferred** |
| 28.5 | 751751 | 14000 | 1 | 0 | 83363010 | Antenatal care provider: Doctor | I | Total | 2903 | 0 |
| 30 | 567476 | 14001 | 1 | 0 | 83363010 | Antenatal care provider: Doctor | I | Total | 4148 | 0 |
| 27.3 | 205488 | 14002 | 1 | 0 | 83363010 | Antenatal care provider: Doctor | I | Total | 2041 | 1 |
| 66.6 | 751748 | 14000 | 1 | 0 | 83363020 | Antenatal care provider: Nurse/midwife | I | Total | 2903 | 0 |
| 65 | 567472 | 14001 | 1 | 0 | 83363020 | Antenatal care provider: Nurse/midwife | I | Total | 4148 | 0 |
| 68.4 | 205485 | 14002 | 1 | 0 | 83363020 | Antenatal care provider: Nurse/midwife | I | Total | 2041 | 1 |
| 0.1 | 751753.26 | 14000 | 1 | 0 | 83363070 | Antenatal care provider: Traditional attendant | I | Total | 2903 | 0 |
| 0.1 | 567471 | 14001 | 1 | 0 | 83363070 | Antenatal care provider: Traditional attendant | I | Total | 4148 | 0 |
| 0.1 | 205487 | 14002 | 1 | 0 | 83363070 | Antenatal care provider: Traditional attendant | I | Total | 2041 | 1 |

# Feature Engineering & Data Preparation

# 1 Load and Combine Datasets

# Load the 7 selected cleaned datasets  
selected\_datasets <- c(  
 "access-to-health-care\_national\_zaf", "immunization\_national\_zaf", "hiv-behavior\_national\_zaf",  
 "water\_national\_zaf", "dhs-quickstats\_national\_zaf", "toilet-facilities\_national\_zaf", "child-mortality-rates\_national\_zaf"  
)  
  
cleaned\_csv\_files <- paste0(file.path(cleaned\_data\_path, selected\_datasets), "\_final.csv")  
if (length(cleaned\_csv\_files[file.exists(cleaned\_csv\_files)]) == 0) {  
 cleaned\_csv\_files <- paste0(file.path(cleaned\_data\_path, selected\_datasets), ".csv")  
}  
  
# Load and combine datasets  
datasets <- lapply(cleaned\_csv\_files[file.exists(cleaned\_csv\_files)], function(file\_path) {  
 df <- read\_csv(file\_path, show\_col\_types = FALSE)  
 df$dataset\_source <- tools::file\_path\_sans\_ext(basename(file\_path))  
 return(df)  
})  
  
combined\_df <- bind\_rows(datasets)  
cat("Combined", length(datasets), "datasets:", nrow(combined\_df), "records,", ncol(combined\_df), "fields\n")

## Combined 7 datasets: 747 records, 12 fields

# 2 Feature Engineering

df\_features <- combined\_df  
  
# Categorical encoding  
df\_features$indicator\_encoded <- as.numeric(as.factor(df\_features$indicator))  
df\_features$survey\_cohort <- as.numeric(as.factor(df\_features$denominator\_unweighted))  
df\_features$dataset\_source\_encoded <- as.numeric(as.factor(df\_features$dataset\_source))  
  
# Group rare categories in by\_variable\_id  
by\_var\_counts <- table(df\_features$by\_variable\_id)  
rare\_threshold <- RARE\_CATEGORY\_THRESHOLD \* nrow(df\_features)  
rare\_categories <- names(by\_var\_counts[by\_var\_counts < rare\_threshold])  
df\_features$by\_variable\_id\_grouped <- ifelse(  
 df\_features$by\_variable\_id %in% rare\_categories, "Other", df\_features$by\_variable\_id  
)  
  
# Create dummy variables  
by\_var\_dummies <- model.matrix(~ by\_variable\_id\_grouped - 1, data = df\_features)  
type\_dummies <- model.matrix(~ indicator\_type - 1, data = df\_features)  
char\_dummies <- model.matrix(~ characteristic\_category - 1, data = df\_features)  
colnames(by\_var\_dummies) <- paste0("by\_var\_", gsub("by\_variable\_id\_grouped", "", colnames(by\_var\_dummies)))  
colnames(type\_dummies) <- paste0("type\_", gsub("indicator\_type", "", colnames(type\_dummies)))  
colnames(char\_dummies) <- paste0("char\_", gsub("characteristic\_category", "", colnames(char\_dummies)))  
  
cat("Encoded categorical variables:", ncol(by\_var\_dummies) + ncol(type\_dummies) + ncol(char\_dummies), "dummy variables created\n")

## Encoded categorical variables: 14 dummy variables created

# 3 Numeric Feature Engineering and Scaling

# Create engineered numeric features  
df\_features$high\_precision <- as.numeric(df\_features$precision <= 1)  
df\_features$char\_order\_quintile <- if(max(df\_features$characteristic\_order, na.rm = TRUE) > 10) {  
 ntile(df\_features$characteristic\_order, 5)  
} else {  
 df\_features$characteristic\_order  
}  
  
df\_features$indicator\_importance <- case\_when(  
 df\_features$indicator\_order <= 3 ~ "High",  
 df\_features$indicator\_order <= 6 ~ "Medium",  
 TRUE ~ "Low"  
)  
  
# Target variable processing  
value\_skewness <- skewness(df\_features$value, na.rm = TRUE)  
df\_features$value\_log <- if(abs(value\_skewness) > 2) log1p(abs(df\_features$value)) else df\_features$value  
df\_features$value\_category <- cut(df\_features$value, breaks = quantile(df\_features$value, c(0, 0.33, 0.67, 1), na.rm = TRUE),  
 labels = c("Low", "Medium", "High"), include.lowest = TRUE)  
  
# Additional engineered features  
df\_features$sample\_size\_tier <- case\_when(  
 df\_features$denominator\_unweighted < SAMPLE\_SIZE\_SMALL ~ "Small",  
 df\_features$denominator\_unweighted < SAMPLE\_SIZE\_LARGE ~ "Medium",  
 TRUE ~ "Large"  
)  
df\_features$data\_quality\_score <- (df\_features$high\_precision \* 0.6) + (df\_features$is\_preferred \* 0.4)  
  
# Scale numeric variables  
numeric\_vars <- c("value\_log", "precision", "characteristic\_order", "indicator\_order", "data\_quality\_score")  
scaled\_data <- df\_features[numeric\_vars] %>% mutate(across(everything(), ~ as.numeric(scale(.))))  
names(scaled\_data) <- paste0(names(scaled\_data), "\_scaled")  
  
cat("Created", ncol(scaled\_data), "scaled numeric features\n")

## Created 5 scaled numeric features

cat("Target variable skewness:", round(value\_skewness, 3),  
 if(abs(value\_skewness) > 2) " (log transformed)" else " (no transform)", "\n")

## Target variable skewness: 8.313 (log transformed)

# 4 Create Final Dataset

# Combine all features into final dataset  
all\_dummies <- cbind(by\_var\_dummies, type\_dummies, char\_dummies)  
  
final\_features <- bind\_cols(  
 df\_features %>% select(data\_id, value, value\_log, value\_category),  
 scaled\_data,  
 df\_features %>% select(is\_preferred, high\_precision, indicator\_encoded, survey\_cohort, dataset\_source\_encoded),  
 df\_features %>% select(char\_order\_quintile, indicator\_importance, sample\_size\_tier),  
 as.data.frame(all\_dummies)  
)  
  
# Create modeling-ready dataset (features only)  
modeling\_features <- final\_features %>% select(-data\_id, -value, -value\_category)  
  
# Export datasets  
write\_csv(final\_features, file.path(outputs\_path, "final\_features\_comprehensive.csv"))  
write\_csv(modeling\_features, file.path(outputs\_path, "modeling\_features.csv"))  
  
# Summary  
cat("Final dataset:", nrow(final\_features), "records,", ncol(modeling\_features), "features\n")

## Final dataset: 747 records, 28 features

cat("- Scaled numeric:", ncol(scaled\_data), "\n")

## - Scaled numeric: 5

cat("- Categorical encoded:", ncol(all\_dummies) + 6, "\n")

## - Categorical encoded: 20

cat("- Exported: modeling\_features.csv (ready for ML)\n")

## - Exported: modeling\_features.csv (ready for ML)

# 5 Dataset Preview

cat("Final training file: modeling\_features.csv\n")

## Final training file: modeling\_features.csv

cat("Records:", nrow(modeling\_features), "| Features:", ncol(modeling\_features), "\n\n")

## Records: 747 | Features: 28

cat("Sample data (first 5 rows, first 8 columns):\n")

## Sample data (first 5 rows, first 8 columns):

head(modeling\_features[1:8], 5)

## # A tibble: 5 × 8  
## value\_log value\_log\_scaled precision\_scaled characteristic\_order\_scaled  
## <dbl> <dbl> <dbl> <dbl>  
## 1 3.38 -0.303 0.660 -0.323  
## 2 3.43 -0.282 0.660 -0.323  
## 3 3.34 -0.320 0.660 -0.323  
## 4 4.21 0.0386 0.660 -0.323  
## 5 4.19 0.0288 0.660 -0.323  
## # ℹ 4 more variables: indicator\_order\_scaled <dbl>,  
## # data\_quality\_score\_scaled <dbl>, is\_preferred <dbl>, high\_precision <dbl>