HDPSA Milestone 2

Bin 381 Group A

9/19/2025

<u>Members:</u>

Llewellyn Jakobus Fourie – 601314

Juan Oosthuizen – 600161

Erin David Cullen – 600531

Qhamaninande Ndlume – 601436

Note Karl Christiaan Schmutz – 577511 did not contribute to this milestone. We will include evidence in a separate file.

Table of Contents

1. 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.

3. 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).

2. Verify Data Quality - Statistical Validation and Field Selections

2.1 Data Preparation & Standardization

2.1.1 Setup Global Variables

```r
# 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
```

```r
# 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
```

```r
cat("Base path:", base_path, "\n")
```

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

```r
cat("Outputs path:", outputs_path, "\n")
```

```
## Outputs path: outputs
```

```r
# 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
```

2.1.2 Load All Datasets

```r
# 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
```

2.1.3 Standardize Field Structure

```r
# 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, charact
eristic_order, by_variable_id, is_total, is_preferred, survey_year_label, denominator_weighted, denominator_u
nweighted, 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, indic
ator_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

```r
# 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
```

3. Field Quality Assessment

3.1 Comprehensive Field Quality Analysis

```r
# 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
```

3.2 Quality Issues Identification

```r
# 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
```

4. Cross-Dataset Correlation Analysis

4.1 Calculate Average Correlation Matrix

```r
# 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

```r
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

```r
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
```

5. Feature Importance & Variance Analysis

5.1 Variance and Information Content Analysis

```r
# 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
```

```r
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

```r
# 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
```

6. Field Importance Weighting

6.1 Field Scoring and Recommendations

```r
# 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

```r
# 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.c
sv"))
cat("Exported: field importance scores and recommendations\n")

## Exported: field importance scores and recommendations
```

7. Dataset Quality Comparison

7.1 Basic Dataset Rankings

```r
# 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

```r
# 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
```

8. Summary and Recommendations

8.1 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
```

8.2 Final Dataset and Field Selection Decisions

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

8.3 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)

8.4 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

8.5 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

8.6 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`*

8.7 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)

8.8 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

9. 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

10. Feature Engineering & Data Preparation

10.1 Load and Combine Datasets

```
# Load the 7 selected cleaned datasets
selected_datasets <- c(
  "access-to-health-care_national_zaf", "immunization_national_zaf", "hiv-behavior_nation
al_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(combin
ed_df), "fields\n")

## Combined 7 datasets: 747 records, 12 fields
```

10.2 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_dummie
s)))
colnames(char_dummies) <- paste0("char_", gsub("characteristic_category", "", colnames(ch
ar_dummies)))

cat("Encoded categorical variables:", ncol(by_var_dummies) + ncol(type_dummies) + ncol(ch
ar_dummies), "dummy variables created\n")

## Encoded categorical variables: 14 dummy variables created
```

Numeric Feature Engineering and Scaling

```r
# 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$value)) else d
f_features$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 = T
RUE)

# 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_pr
eferred * 0.4)

# Scale numeric variables
numeric_vars <- c("value_log", "precision", "characteristic_order", "indicator_order", "d
ata_quality_score")
scaled_data <- df_features[numeric_vars] %>% mutate(across(everything(), ~ as.numeric(sca
le(.))))
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)
```

10.3 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), "feature
s\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)
```

10.4 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>
```

Modelling_features.csv will be the final dataset for modelling, it will be split in the next milestone for training and test sets.

This is the cleaned, processed and scaled data.