MCP ANALYTICS

Data Profile — Understand Any Dataset

Statistical analysis report with 8 sections — interactive charts, validated methodology, and AI-generated insights.

Generated **July 18, 2026**

Sections **8**

Platform **mcpanalytics.ai**

Contents

EXECUTIVE SUMMARY

Executive Summary

Key Metrics

Rows	**300**
Columns Profiled	**8**
Numeric Columns	**3**
Categorical Columns	**4**
Date Columns	**1**
Columns Flagged	**4**
Duplicate Rows	**0**
Overall Missing %	**5.5**

Suggested Interpretation

Your dataset spans 300 rows and 8 columns: 3 numeric, 4 categorical, 1 date. Four columns are clean; four were flagged for review. No duplicate rows exist. Overall missingness is 5.5%. The strongest numeric relationship is between Units Sold and Revenue (r = 0.998), indicating they move together closely, likely reflecting a direct business relationship rather than independent variation.

OVERVIEW

Analysis Overview

Column-by-column profile of 8 columns and 300 rows.

Configuration

N Rows	300
N Columns	8
N Numeric	3
N Flagged	4

Suggested Interpretation

Your dataset contains 300 rows and 8 columns with a balanced type mix: 3 numeric, 4 categorical, and 1 date column. Data profiling is the diagnostic foundation before analysis—it detects column types automatically, then applies the right statistics for each (spread and outliers for numbers, distinct counts for categories). Overall, 5.5% of cells are missing, and 4 of 8 columns were flagged for quality issues. Start here to identify problems like constant columns, heavy skew, or missing-data patterns that could bias downstream results.

TABLE 3

Column-by-Column Summary

Detected type, missingness, and a headline note for every column.

Column	Type	Missing PCT	Distinct Or Mean	Note
Revenue	numeric	0	mean 3,604	Clean
Units Sold	numeric	0	mean 123.9	Clean
Session Minutes	numeric	0	mean 4.436	Extreme skew
Region	categorical	0	5 distinct	Clean
Customer ID	id	0	300 distinct	Identifier-like
Signup Date	date	0	203 distinct	Clean
Discount Tier	categorical	44	3 distinct	High missingness
Currency	constant	0	1 distinct	Constant column

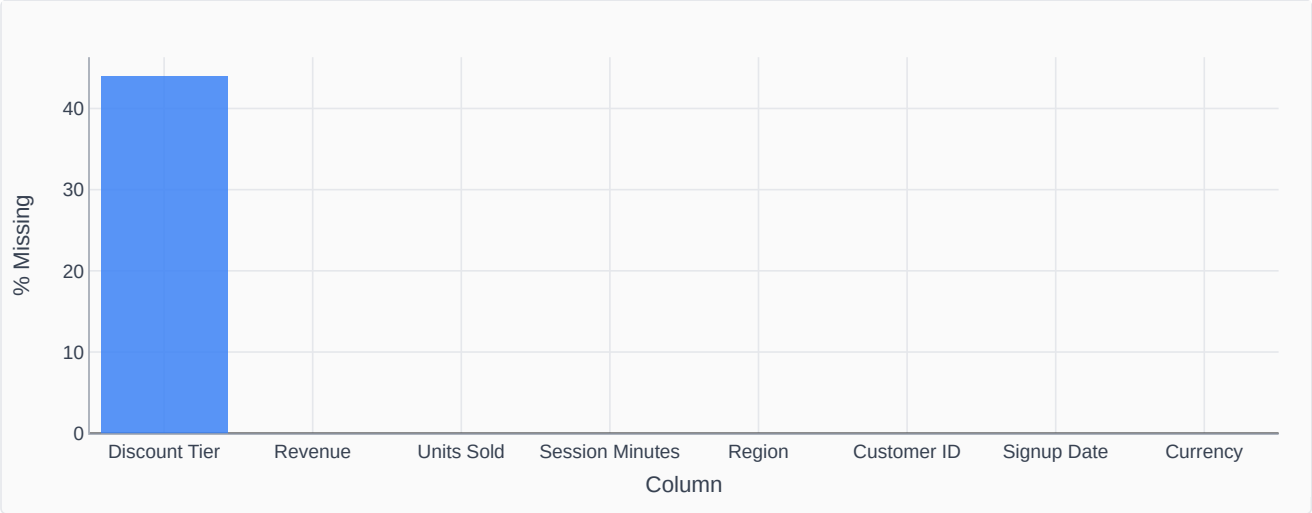
Suggested Interpretation

Seven columns show specific profiles: Revenue (numeric, mean 3,604, clean), Units Sold (numeric, mean 123.9, clean), Session Minutes (numeric, mean 4.436, extreme skew), Region (categorical, 5 distinct values, clean), Customer ID (300 distinct values, identifier-like), Signup Date (203 distinct dates, clean), and Discount Tier (3 distinct values, 44% missing). Currency is constant with only 1 distinct value (USD), adding no analytical information. The flagged columns require attention before use in analysis.

FIGURE 4

Missing Data by Column

Percentage of missing values in each profiled column.



Suggested Interpretation

Only 1 of 8 columns contains missing values. Discount Tier is the sole affected column, with 44% missingness (132 of 300 rows missing). This exceeds the 20% threshold that can bias computed statistics. All other columns—Revenue, Units Sold, Session Minutes, Region, Customer ID, Signup Date, and Currency—are complete with 0% missing. The concentration of missingness in a single column simplifies the decision about handling incomplete data.

TABLE 5

Numeric Distributions

Centre, spread, skew, and outlier counts for numeric columns.

Column	Mean	Median	SD	Min	Max	Skew	Outliers
Revenue	3,604	3,557	1,027	507.6	5,969	-0.07	1
Units Sold	123.9	122.6	41.04	1	215.9	-0.08	1
Session Minutes	4.436	2.46	6.322	0.21	68.54	5.23	33

Suggested Interpretation

The 3 numeric columns show distinct distribution shapes. Revenue (mean 3,604, median 3,557, skew -0.07) and Units Sold (mean 123.9, median 122.6, skew -0.08) are nearly symmetric with minimal skew and only 1 outlier each. Session Minutes is heavily right-skewed (skew 5.23) with mean 4.436 far above median 2.46, indicating a long tail of high values. Session Minutes contains 33 outliers beyond 1.5 times the interquartile range—33 of the dataset's 35 total outliers. This extreme skew warrants transformation or robust methods before modeling.

TABLE 6

Data Quality Flags

Columns with issues that need attention before analysis.

Column	Issue	Detail
Session Minutes	Extreme skew	Strongly skewed to the right (a long high-value tail) (skew 5.233) — the mean is pulled away from typical values.
Customer ID	Identifier-like	300 distinct values across 300 rows — looks like a row identifier, not a variable.
Discount Tier	High missingness	44% of values are missing (132 of 300 rows).
Currency	Constant column	Only one distinct value (USD) — no information for analysis.

Suggested Interpretation

Four columns require attention. Session Minutes exhibits extreme right skew (5.23), pulling the mean away from typical values and creating 33 outliers. Customer ID is identifier-like with 300 distinct values across 300 rows—useful for row reference but not analysis. Discount Tier has 44% missingness, biasing any statistic computed from it. Currency is constant (only USD), providing no variation for analysis. Address these four issues—drop or transform skewed columns, exclude identifiers, handle missing data explicitly, and remove constant columns—before proceeding to modeling or statistical inference.

METHODOLOGY

Methodology

Statistical method, assumptions, and diagnostics

Data Profile — Understand Any Dataset

Standard-library analysis: the "what's in my data" report. Map the columns you want profiled and get a per-column breakdown — detected type (numeric, categorical, date, constant, or identifier), missing-value share, and the right summary statistics for each type — plus a dataset overview, a missing-data map, numeric distribution stats (spread, skew, outliers), a data-quality flag list, and a peek at your strongest numeric correlation. Works on any dataset: map 2 or more columns of any type.

ASSUMPTIONS

- Each column holds one consistent kind of value (numbers, text, or dates)
- Numeric columns are numeric or cleanly convertible to numbers

LIMITATIONS

- Type detection is heuristic — a mostly-numeric column with stray text is still treated as numeric
- Correlation peek uses linear Pearson r and only covers numeric columns
- Profiling summarizes columns independently; it does not model relationships between them

---

SOFTWARE & CITATION

MCP Analytics · mcpanalytics.ai

APPENDIX

Analysis Code

Complete R source code for this analysis

Data Profile — Understand Any Dataset

The "what's in my data" EDA report. The user maps the columns they want profiled; the tool detects each column's type, computes the right summary statistics for that type, flags data-quality problems, and gives a dataset-level overview plus a peek at the strongest numeric correlation.

A.1 Why This Method?

Before any real analysis you need to know your data: how many rows and columns, which columns are numeric vs categorical vs dates, where values are missing, which columns are constant or identifier-like, and whether anything is skewed or full of outliers. This is the single most-demanded first step — it turns a raw upload into an understood dataset.

A.2 What This Analysis Covers

- Per-column type detection and summary statistics
- A missing-data map across every mapped column
- Numeric distribution statistics (spread, skew, outliers)
- A data-quality flag list (missingness, constants, ids, skew)

A.3 Standard Library

Platform standard-library module (LAT-1441): runs on ANY dataset via the semantic mapping {column_1..column_N}. All narrative is derived from the user's own column names and computed values.

```
suppressPackageStartupMessages(library(DT))
suppressPackageStartupMessages(library(htmlwidgets))
suppressPackageStartupMessages(library(arrow))
suppressPackageStartupMessages(library(knitr))
suppressPackageStartupMessages(library(rmarkdown))
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library(tidyr))
suppressPackageStartupMessages(library(ggplot2))
suppressPackageStartupMessages(library(stringr))
suppressPackageStartupMessages(library(lubridate))
suppressPackageStartupMessages(library(broom))
suppressPackageStartupMessages(library(Matrix))
suppressPackageStartupMessages(library(cluster))
suppressPackageStartupMessages(library(data.table))
```

A.4 Helpers

A.5 Core Analysis Pipeline

```
compute_shared <- function(df, params, col_map = list()) {
  # === SHARED EXPORTS ===
  #   initial_rows/final_rows/rows_removed  $ row accounting (no rows removed)
  #   n_rows / k_cols                       $ dataset size
  #   profiles          $ list, one entry per column (name/type/stats/flags)
  #   n_numeric/n_categorical/n_date/n_other $ type counts
  #   overall_miss_pct  $ numeric — missing cells / total cells
  #   dup_rows          $ integer — duplicate rows across mapped columns
  #   n_flagged/n_clean $ integer — columns with/without quality issues
  #   column_summary_df $ data.frame(column, type, missing_pct, distinct_or_mean, note)
  #   missingness_df    $ data.frame(column, missing_pct)
  #   numeric_stats_df  $ data.frame(column, mean, median, sd, min, max, skew, outliers)
  #   quality_flags_df  $ data.frame(column, issue, detail)
  #   corr_peek         $ list(a,b,r) or NULL — strongest numeric pair
  #   metrics / json_output
  # === /SHARED EXPORTS ===
```

Step 1: Discover mapped columns

```
initial_rows <- nrow(df)
  col_cols <- grep("^column_[0-9]+$", names(df), value = TRUE)
  col_cols <- col_cols[order(as.integer(sub("^column_", "", col_cols)))]
  if (length(col_cols) < 1) {
    stop("Map at least one column to profile(column_1, column_2, ...).")
  }
  n_rows <- initial_rows
  if (n_rows < 5) {
    stop(sprintf("Data Profile needs at least 5 rows to profile; only %d supplied.",
                 n_rows))
  }
  disp_names <- setNames(humanize_semantic(col_cols, col_map), col_cols)
```

Step 2: Duplicate-row count across the mapped columns

```
dup_rows <- as.integer(sum(duplicated(df[, col_cols, drop = FALSE])))
```

Step 3: Profile every column

```
ID_FRAC <- 0.9          # unique / n_rows above this ⇒ identifier-like
  MISS_HI <- 20           # missing % above this ⇒ high-missingness flag
  SKEW_HI <- 2            # |skew| above this ⇒ extreme-skew flag
  NZV_CV  <- 0.01         # coefficient of variation below this ⇒ near-zero variance

  profiles <- list()
  numeric_vectors <- list()   # disp_name -> full-length numeric vector (NA where not numeric/missing)
  total_missing_cells <- 0

  for (sc in col_cols) {
    disp <- disp_names[[sc]]
    raw <- df[[sc]]
    s <- as.character(raw)
    miss <- is.na(raw) | is.na(s) | trimws(s) == ""
    n_miss <- sum(miss)
    total_missing_cells <- total_missing_cells + n_miss
    miss_pct <- 100 * n_miss / n_rows
    nonmiss <- s[!miss]
    n_nonmiss <- length(nonmiss)
    n_distinct <- length(unique(nonmiss))

    conv <- suppressWarnings(as.numeric(nonmiss))
    num_ok <- n_nonmiss > 0 && sum(!is.na(conv)) >= 0.95 * n_nonmiss
    is_id_card <- n_nonmiss > 0 && n_distinct > ID_FRAC * n_rows
```

Type detection (order matters): constant → numeric → date → id → categorical

```
ctype <- "categorical"
    if (n_distinct <= 1) {
      ctype <- "constant"
    } else if (num_ok) {
      numv0 <- conv[!is.na(conv)]
      if (length(numv0) > 0 && all(numv0 == round(numv0)) && is_id_card) {
        ctype <- "id"
      } else {
        ctype <- "numeric"
      }
    } else if (.looks_like_date(nonmiss)) {
      ctype <- "date"
    } else if (is_id_card) {
      ctype <- "id"
    } else {
      ctype <- "categorical"
    }
```

Per-type statistics

```
      stat <- list(mean = NA_real_, median = NA_real_, sd = NA_real_,
                   min = NA_real_, max = NA_real_, iqr = NA_real_,
                   skew = NA_real_, outliers = NA_integer_,
                   top_level = NA_character_, top_share = NA_real_)
          if (ctype == "numeric") {
            numv <- conv[is.finite(conv)]
            full <- rep(NA_real_, n_rows)
            full[!miss] <- conv
            numeric_vectors[[disp]] <- full
            if (length(numv) >= 1) {
              qs <- suppressWarnings(stats::quantile(numv, c(0.25, 0.75), names = FALSE))
              iqr <- qs[2] - qs[1]
              stat$mean   <- mean(numv)
              stat$median <- stats::median(numv)
              stat$sd     <- if (length(numv) >= 2) stats::sd(numv) else 0
              stat$min    <- min(numv)
              stat$max    <- max(numv)
              stat$iqr    <- iqr
              stat$skew   <- .skewness(numv)
              lo <- qs[1] - 1.5 * iqr; hi <- qs[2] + 1.5 * iqr
              stat$outliers <- as.integer(sum(numv < lo | numv > hi))
            }
          } else if (n_nonmiss > 0) {
            tab <- sort(table(nonmiss), decreasing = TRUE)
            stat$top_level <- names(tab)[1]
            stat$top_share <- 100 * as.numeric(tab[1]) / n_nonmiss
          }
```

Quality flags for this column

```
    flags <- list()
    if (miss_pct > MISS_HI) {
      flags[[length(flags) + 1]] <- list(
        issue = "High missingness",
        detail = sprintf("%s of values are missing(%s of %s rows).",
                         .fmt_pct(miss_pct), .fmt_num(n_miss), .fmt_num(n_rows)))
    }
    if (ctype == "constant") {
      only_val <- if (n_nonmiss > 0) nonmiss[1] else "(all missing)"
      flags[[length(flags) + 1]] <- list(
        issue = "Constant column",
        detail = sprintf("Only one distinct value(%s) — no information for analysis.",
                         only_val))
    }
    if (ctype == "id") {
      flags[[length(flags) + 1]] <- list(
        issue = "Identifier-like",
        detail = sprintf("%s distinct values across %s rows — looks like a row identifier, not a variable.",
                         .fmt_num(n_distinct), .fmt_num(n_rows)))
    }
    if (ctype == "numeric" && is.finite(stat$sd) && is.finite(stat$mean)) {
      cv <- if (abs(stat$mean) > 1e-9) stat$sd / abs(stat$mean) else if (stat$sd == 0) 0 else Inf
      if (stat$sd == 0 || (is.finite(cv) && cv < NZV_CV)) {
        flags[[length(flags) + 1]] <- list(
          issue = "Near-zero variance",
          detail = sprintf("Values barely vary(standard deviation %s around a mean of %s).",
                           .fmt_num(stat$sd), .fmt_num(stat$mean)))
      }
    }
    if (ctype == "numeric" && is.finite(stat$skew) && abs(stat$skew) > SKEW_HI) {
      dirn <- if (stat$skew > 0) "right(a long high-value tail)" else "left(a long low-value tail)"
      flags[[length(flags) + 1]] <- list(
        issue = "Extreme skew",
        detail = sprintf("Strongly skewed to the %s(skew %s) — the mean is pulled away from typical values.",
                         dirn, .fmt_num(stat$skew)))
    }
```

A single headline note for the summary table (highest-priority issue)

```
    note <- if (length(flags) > 0) flags[[1]]$issue else "Clean"
```

distinct-or-mean cell for the summary table

```
    dm <- if (ctype == "numeric") {
          paste0("mean ", .fmt_num(stat$mean))
        } else {
          paste0(.fmt_num(n_distinct), " distinct")
        }

        profiles[[sc]] <- list(
          sem = sc, disp = disp, type = ctype,
          n_missing = n_miss, miss_pct = miss_pct,
          n_distinct = n_distinct, stat = stat,
          flags = flags, note = note, distinct_or_mean = dm)
      }
```

Step 4: Type counts + overall missingness

```
    types <- vapply(profiles, function(p) p$type, character(1))
      n_numeric     <- sum(types == "numeric")
      n_date        <- sum(types == "date")
      n_categorical <- sum(types %in% c("categorical", "id", "constant"))
      k_cols <- length(profiles)
      overall_miss_pct <- 100 * total_missing_cells / (n_rows * k_cols)
```

Step 6: Numeric correlation peek (strongest |Pearson| pair)

```
    corr_peek <- NULL
      if (length(numeric_vectors) >= 2) {
        M <- do.call(cbind, numeric_vectors)
        colnames(M) <- names(numeric_vectors)
        Cc <- suppressWarnings(stats::cor(M, use = "pairwise.complete.obs"))
        Cc[!is.finite(Cc)] <- NA
        if (nrow(Cc) >= 2) diag(Cc) <- NA
        if (any(!is.na(Cc))) {
          mx <- max(abs(Cc), na.rm = TRUE)
          hit <- which(abs(Cc) == mx, arr.ind = TRUE)
          if (nrow(hit) >= 1) {
            i <- hit[1, 1]; j <- hit[1, 2]
            corr_peek <- list(a = rownames(Cc)[i], b = colnames(Cc)[j],
                              r = round(Cc[i, j], 3))
          }
        }
      }
```

Step 7: Metrics + json_output

```
metrics <- list(
    `Rows`               = n_rows,
    `Columns Profiled`   = k_cols,
    `Numeric Columns`    = as.integer(n_numeric),
    `Categorical Columns`= as.integer(n_categorical),
    `Date Columns`       = as.integer(n_date),
    `Columns Flagged`    = as.integer(n_flagged),
    `Duplicate Rows`     = dup_rows,
    `Overall Missing %`  = round(overall_miss_pct, 1)
  )

  quality_headline <- if (n_flagged == 0) {
    "no columns raised a data-quality flag"
  } else {
    sprintf("%s of %s %s flagged for review",
            .fmt_num(n_flagged), .fmt_num(k_cols),
            .plural(n_flagged, "column was", "columns were"))
  }
  corr_sentence <- if (!is.null(corr_peek)) {
    sprintf(" Strongest numeric relationship: %s and %s(r = %s).",
            corr_peek$a, corr_peek$b, .fmt_num(corr_peek$r))
  } else ""

  json_answer <- paste0(
    "Profiled ", .fmt_num(k_cols), " ", .plural(k_cols, "column", "columns"),
    " across ", .fmt_num(n_rows), " rows: ",
    .fmt_num(n_numeric), " numeric, ", .fmt_num(n_categorical),
    " categorical, ", .fmt_num(n_date), " date. ",
    "Overall ", .fmt_pct(overall_miss_pct), " of cells are missing; ",
    .fmt_num(dup_rows), " duplicate ", .plural(dup_rows, "row", "rows"), ". ",
    toupper(substring(quality_headline, 1, 1)), substring(quality_headline, 2),
    ".", corr_sentence)

  json_output <- list(
    answer = json_answer,
    cards = lapply(
      c("tldr", "overview", "column_summary", "missingness",
        "numeric_distributions", "data_quality_flags"),
      function(cid) list(id = cid, metrics = metrics))
  )

  list(
    initial_rows = initial_rows, final_rows = initial_rows, rows_removed = 0L,
    n_rows = n_rows, k_cols = k_cols,
    profiles = profiles, disp_names = disp_names,
    n_numeric = as.integer(n_numeric), n_categorical = as.integer(n_categorical),
    n_date = as.integer(n_date),
    overall_miss_pct = overall_miss_pct, dup_rows = dup_rows,
    n_flagged = as.integer(n_flagged), n_clean = as.integer(n_clean),
    column_summary_df = column_summary_df, missingness_df = missingness_df,
    numeric_stats_df = numeric_stats_df, quality_flags_df = quality_flags_df,
    corr_peek = corr_peek, metrics = metrics, json_output = json_output
  )
}
```

This is the exact R code executed to produce the analysis above. Run it with the same dataset to reproduce these results.