Data 607 Project 3

Joao De Oliveira

2025-10-15

Overview

This notebook reads Google Trends CSV exported from the website (from a local path or GitHub RAW URL), cleans and reshapes it to a simple long format, saves the clean CSV (Beginner), and optionally loads the data into a small SQLite database (Intermediate). The database step can be turned on/off with a single flag.

Parameters

```
library(fs)

# GitHub raw csv file
csv_urls <- c(
  "https://raw.githubusercontent.com/JDO-MSDS/Data-607-Project-3/refs/heads/main/data/data_science_skil
)

# Skills to keep - we used r programming because the results for r are not very helpful
skills <- c("python", "r programming", "sql", "tableau")

time_window  <- "2004-01-01 2025-10-01"
granularity  <- "weekly"
retrieved_on <- Sys.Date()

# Outputs
dir_create("data")
clean_csv_out <- "data/trends_long.csv"
db_path       <- "data/warehouse.db"

# database load (TRUE = write to SQLite - FALSE = CSV only)
write_to_db <- TRUE
```

Load Packages

```
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyr)
library(stringr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(DBI)
library(RSQLite)
library(purrr)
```

Load and Clean Google Trends Data

```r
detect_header_row <- function(path_or_url) {
  lines <- readr::read_lines(path_or_url, n_max = 200)
  if (length(lines) && startsWith(lines[1], "\ufeff")) {
    lines[1] <- sub("^\ufeff", "", lines[1])
  }
  for (i in seq_along(lines)) {
    parts <- strsplit(lines[i], ",", fixed = TRUE)[[1]]
    if (length(parts) > 1) {
      first <- trimws(parts[1])
      if (grepl("^Week$|^Date$", first, ignore.case = TRUE)) return(i - 1L)
      if (!is.na(suppressWarnings(lubridate::ymd(first)))) return(i - 1L)
      if (!is.na(suppressWarnings(lubridate::mdy(first)))) return(i - 1L)
    }
  }
  stop("Could not find a header with a date or Week/Date column in: ", path_or_url)
}

read_trends_csv <- function(path_or_url, default_region = "US") {
  skip_n <- detect_header_row(path_or_url)
  df <- readr::read_csv(path_or_url, skip = skip_n, show_col_types = FALSE)

  # Normalizing first column to 'date'
  names(df)[1] <- "date"
  df <- df %>% mutate(
    date = as.character(date),
    date = sub("^\ufeff", "", date),
    date = sub("\\s+-\\s+.*$", "", date) # keep start of a range like "2020-01-01 - 2020-01-07"
  )
  parsed <- suppressWarnings(lubridate::ymd(df$date))
  parsed2 <- suppressWarnings(lubridate::mdy(df$date))
  df$date <- ifelse(is.na(parsed), parsed2, parsed)
  df <- df %>% filter(!is.na(date))

  # only date + columns that mention any of our skills
  nm <- names(df)
  skill_pattern <- paste0("(", paste(stringr::str_to_lower(skills), collapse = "|"), ")")
  lower_nm <- stringr::str_to_lower(nm)
  skill_match <- stringr::str_detect(lower_nm, skill_pattern)
  keep_cols <- unique(c("date", nm[skill_match]))
  df <- dplyr::select(df, dplyr::all_of(keep_cols))

  # make all non-date columns character
  df <- df %>% mutate(across(-date, as.character))

  if (ncol(df) <= 1) {
    stop(
      "No series columns matched your skills in CSV: ", path_or_url,
      "\nAvailable columns:\n", paste(nm, collapse = " | "),
      "\nTip: confirm headers include your terms (e.g., 'python', 'r programming', 'sql', 'tableau')."
    )
  }

  # Long format
  long <- df %>%
    tidyr::pivot_longer(cols = -date, names_to = "raw_col", values_to = "interest") %>%
    dplyr::mutate(
      interest = ifelse(interest == "<1", "0.5", as.character(interest)),
      interest = suppressWarnings(as.numeric(interest)),
      term = stringr::str_trim(
        stringr::str_to_lower(stringr::str_remove(raw_col, "\\s*\\(.*\\)$"))
      ),
      region = dplyr::case_when(
        stringr::str_detect(raw_col, fixed("(United States)")) ~ "US",
        stringr::str_detect(raw_col, fixed("(Worldwide)"))     ~ "WORLD",
        TRUE                                                    ~ default_region
      )
    ) %>%
    dplyr::filter(!is.na(interest))

  canon <- function(x) {
    x %>%
      stringr::str_replace(stringr::fixed("(search term)"), "") %>%
      stringr::str_replace(stringr::fixed("(programming language)"), "") %>%
      stringr::str_replace_all("[^A-Za-z0-9[:space:]]", " ") %>%
      stringr::str_squish()
  }

  long %>%
    dplyr::mutate(skill_name = canon(term)) %>%
    dplyr::select(date, skill_name, region, interest) %>%
    dplyr::arrange(skill_name, date)
}

load_all_from_sources <- function(url_vec) {
  if (length(url_vec) == 0) stop("No CSV sources provided.")
  purrr::map_df(url_vec, read_trends_csv)
}
```

```
iot_clean <- load_all_from_sources(csv_urls) %>%
  dplyr::mutate(
    skill_name = dplyr::case_when(
      skill_name %in% c("r", "r programming", "r programming language", "r  programming") ~ "r programmi
      TRUE ~ skill_name
    )
  ) %>%
  dplyr::filter(stringr::str_to_lower(skill_name) %in% stringr::str_to_lower(skills))

# Simple check
dup <- iot_clean %>% dplyr::count(date, skill_name, region) %>% dplyr::filter(n > 1)
if (nrow(dup) > 0) message("Warning: duplicate (date, skill, region) rows: ", nrow(dup))

# Basic summary per skill
iot_clean %>%
  dplyr::group_by(skill_name) %>%
  dplyr::summarise(
    min_date = min(date),
    max_date = max(date),
    rows = dplyr::n(),
    mean_interest = mean(interest, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  print()
```

```
## # A tibble: 3 x 5
##   skill_name min_date max_date  rows mean_interest
##   <chr>         <dbl>    <dbl> <int>         <dbl>
## 1 python        18546    20373   262         65.2
## 2 sql           18546    20373   262         16.5
## 3 tableau       18546    20373   262          1.20
```

```
# Write clean CSV
readr::write_csv(iot_clean, clean_csv_out)
cat("\nWrote clean long-format CSV to:", clean_csv_out, "\n")
```

```
##
## Wrote clean long-format CSV to: data/trends_long.csv
```

Load into SQLite

```r
# Ensure db directory exists
fs::dir_create(dirname(db_path))

con <- dbConnect(RSQLite::SQLite(), db_path)
if (!DBI::dbIsValid(con)) stop("Failed to open SQLite connection: ", db_path)

# tables
dbExecute(con, "CREATE TABLE IF NOT EXISTS Skill (
skill_id INTEGER PRIMARY KEY,
skill_name TEXT UNIQUE NOT NULL
);")
```

```
## [1] 0
```

```r
dbExecute(con, "CREATE TABLE IF NOT EXISTS TrendQuery (
query_id INTEGER PRIMARY KEY,
skill_id INTEGER NOT NULL,
region TEXT NOT NULL,
time_window TEXT NOT NULL,
granularity TEXT NOT NULL,
retrieved_on DATE NOT NULL,
FOREIGN KEY(skill_id) REFERENCES Skill(skill_id)
);")
```

```
## [1] 0
```

```r
dbExecute(con, "CREATE TABLE IF NOT EXISTS TrendPoint (
point_id INTEGER PRIMARY KEY,
query_id INTEGER NOT NULL,
date DATE NOT NULL,
interest_score REAL NOT NULL,
FOREIGN KEY(query_id) REFERENCES TrendQuery(query_id)
);")
```

```
## [1] 0
```

```r
skills_df <- iot_clean %>% distinct(skill_name) %>% arrange(skill_name)
existing_skills <- dbGetQuery(con, "SELECT skill_id, skill_name FROM Skill;")
to_insert <- dplyr::anti_join(skills_df, existing_skills, by = "skill_name")
if (nrow(to_insert) > 0) dbWriteTable(con, "Skill", to_insert, append = TRUE)
skill_dim <- dbGetQuery(con, "SELECT skill_id, skill_name FROM Skill;")

# New rows
query_df <- iot_clean %>%
  distinct(skill_name, region) %>%
  inner_join(skill_dim, by = "skill_name") %>%
  transmute(
    skill_id,
    region,
    time_window = time_window,
    granularity = granularity,
    retrieved_on = as.character(retrieved_on)
  )

existing_query <- dbGetQuery(con, "SELECT skill_id, region, time_window, granularity, retrieved_on FROM
existing_query <- existing_query %>% mutate(retrieved_on = as.character(retrieved_on))

query_new <- dplyr::anti_join(query_df, existing_query,
  by = c("skill_id","region","time_window","granularity","retrieved_on")
)
if (nrow(query_new) > 0) dbWriteTable(con, "TrendQuery", query_new, append = TRUE)
query_dim <- dbGetQuery(con, "SELECT query_id, skill_id, region FROM TrendQuery;")

# Insert tendpoint rows
points_df <- iot_clean %>%
  inner_join(skill_dim, by = "skill_name") %>%
  inner_join(query_dim, by = c("skill_id","region")) %>%
  transmute(query_id, date = as.Date(date), interest_score = interest)

existing_points <- dbGetQuery(con, "SELECT query_id, date FROM TrendPoint;") %>%
  mutate(date = as.Date(date))
points_to_insert <- dplyr::anti_join(points_df, existing_points, by = c("query_id","date"))
if (nrow(points_to_insert) > 0) dbWriteTable(con, "TrendPoint", points_to_insert, append = TRUE)

cat("Loaded", nrow(points_to_insert), "TrendPoint rows into", db_path, "\n")
```

```
## Loaded 786 TrendPoint rows into data/warehouse.db
```

```
# Clean disconnect
try(DBI::dbDisconnect(con), silent = TRUE)
```