# Data Engineering Take-Home Challenge

## Overview

In this exercise, you’ll work with two CSV files representing user data and user events. The goal is to clean and join the data to produce a usable dataset for downstream analytics.

This task is intentionally lightweight and open-ended. It’s designed to reflect the kinds of small-but-real cleanup and data integration tasks that come up in everyday data engineering work.

We’re not looking for over-engineering or boilerplate—just solid, readable code and attention to detail.

## The Data

You’ll be given two CSV files:

### users.csv

Contains information about registered users.

* user\_id: Unique ID (integer)
* name: Full name (may include prefixes, suffixes, compound names, etc.)
* email: User’s email address (clean, canonical form)
* signup\_date: Date of signup (YYYY-MM-DD)

### events.csv

Contains user activity logs.

* event\_id: Unique event ID
* user\_email: Email address of the user who triggered the event  
  *(Note: these may have inconsistent formatting, casing, and occasional quirks like invisible or special characters)*
* event\_type: Type of event (e.g. login, purchase, etc.)
* timestamp: Timestamp of the event

## Your Task

Produce a cleaned output CSV file (e.g. joined\_events.csv) that joins user events with user data using the email address. The final output should contain:

* user\_id
* first\_name
* last\_name
* event\_type
* timestamp
* signup\_date

### Also:

* Normalize emails as needed to get a successful join.
* Log or output a list of any event\_ids that could not be matched to a user.
* Detect and report any duplicate emails in users.csv (flag these, do not force-fix).
* Extract first\_name and last\_name from the full name field in users.csv.
  + **Discard any prefixes or suffixes** (e.g., “Dr.”, “Jr.”).
  + If the name can’t be cleanly split, use your best judgment and note any assumptions.

## Stretch Goals (Optional, Not Required)

If you have time or want to go above and beyond:

* Add a summary table showing number of events per user.
* Allow file paths for input/output to be passed via command-line arguments.
* Handle unusual input cases such as:
  + Non-breaking spaces (\u00A0)
  + Zero-width characters (\u200B)
  + Unicode homoglyphs
* Add basic unit tests or data validations.
* Use logging and comments effectively.

## Submission Instructions

Please submit the following:

* Your code (in a .py or .ipynb file)
* The final output file (joined\_events.csv)
* Any notes you want us to read (assumptions, challenges, comments)

This should take around **1–2 hours**. If you run into issues or want to explain decisions you made, that’s totally fine—just include a short note.

## What We’re Evaluating

* Correctness and completeness
* Code readability and structure
* Attention to edge cases and data quality
* Practical decision-making — good enough vs overkill
* Thoughtfulness around messy data

Use whatever libraries you’re comfortable with (e.g. polars or pandas). Focus on clarity, professionalism, and real-world instincts.