Client Data Onboarding Cleaner

Fernando Djingga
24 July 2025

Introduction

I created a lightweight client data onboarding cleaner that processes messy client spreadsheets containing inconsistent column names, invalid or missing emails, improperly formatted phone numbers, duplicates, and dates.

My project generates two outputs per run:

1. clean_clients.csv produces a CSV file with standardized, deduplicated, and validated records suitable for analytics dashboards.

2. validation_report.html generates a HTML report that highlights what rows were cleaned, flagged, or dropped, enabling quick triage and documentation.

With simple Python and Pandas, my project automates spreadsheet cleaning tasks that reduce processing time and ensuring consistent quality and validation; this provides real value to SaaS onboarding and workflow efficiency.

High-Level Implementation

My cleaner is designed to handle typical onboarding spreadsheet issues systematically. Firstly, I apply column normalization because raw CSV exports often contain inconsistent headings, such as for the Client ID, name, and email. My cleaner ensures that the names are properly standardized, ensuring uniformity across datasets. I also apply string clean up that trims whitespace, remove empty strings, and converts text into consistent casing. If only a ‘full_name’ column is present, it is automatically split into ‘first_name’ and ‘last_name’ for better structure.

Email should all be lowercased, and invalid formats and syntax are flagged in the HTML report. Email domains are extracted into a new ‘email_domain’ column for segmentation and analysis. Phone numbers should be in digits and keep the ‘+’ for international country code. This ensures comparability while discarding inconsistent spacing or formatting. Dates are converted into ISO format (YYYY-MM-DD), and invalid dates that does not follow this format will be flagged. Rows that are missing both the email and phone are considered unreachable and must be dropped. Any duplicated data must be dropped and removed. Output columns must also be reordered into a canonical order for consistency. Finally, my program generates the two reports.

Coding Implementation

My implementation is structured into a single and readable module. ‘cleaner.py’ contains the full cleaning pipeline with modular helper functions (_normalize_columns, _normalize_email, _parse_dates, etc.). It also includes a CLI interface that supports custom input/output/report paths.

I also created a test code that verifies core functionality, where emails are lowercased and invalid ones are flagged, full names are correctly split, dates are standardized to ISO format, and duplicate entries are removed as expected.

In terms of my input, ‘raw_clients.csv’ is the messy input with sample of typos, duplicates, and invalid emails or dates. My program produces the output as ‘clean_clients.csv’ and ‘validation_report.html’.

Instructions of how to run my program is found on README.md.

Project Results

The following is the initial ‘raw_clients.csv’.

data > raw_clients.csv

```
1  Client ID,Full Name,Email,Phone,Signup Date,Plan,Country,Company,Notes
2  1,   Alice   Smith  ,ALICE.SMITH@Example.COM, +852  6123-4567 ,2024-01-02,Pro,HK,Acme  , VIP
3  2,Bob Jones,bob.jones[at]mail.com,  (021) 812-3344  ,01/15/2024,Basic,ID,Fuga Ltd,"typo email"
4  3,Charlie,,+62 812-0000-000,2024/02/30,Pro,ID,,no email
5  4,,dora@sample.io, ,2024-03-05,Pro,HK,Innotech,"missing name"
6  5,Edward Nigma,edward.nigma@sample.io,+1 (415) 555-1212,03-05-2024,Enterprise,US,Riddler Inc.,"ok"
7  5,Edward Nigma,EDWARD.NIGMA@sample.io,+1-415-555-1212,2024-03-05,Enterprise,US,Riddler Inc.,"dup id & email"
8  ,No Phone Or Email,, ,2024-04-01,Free,HK,,"drop me"
```

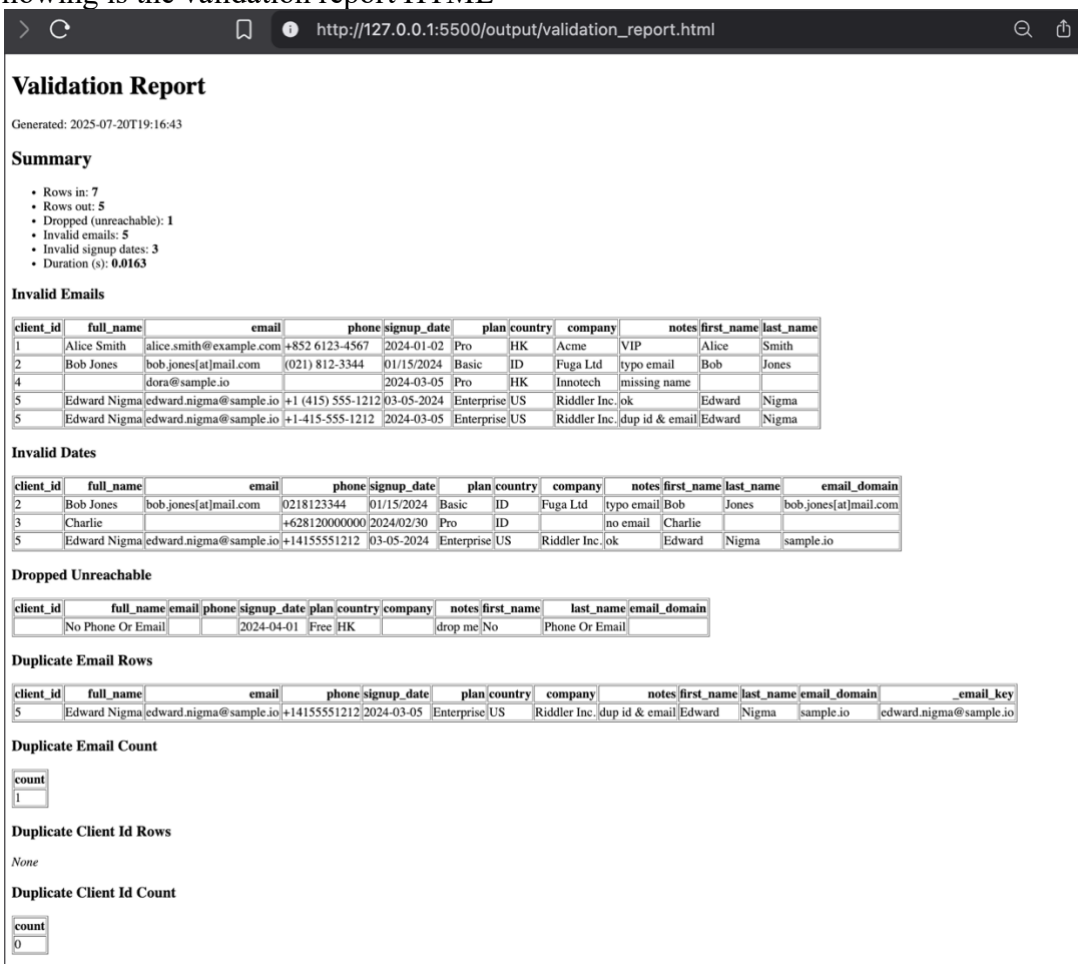
The following is the cleaned ‘clean_clients.csv’.

data > clean_clients.csv

```
1  client_id,first_name,last_name,full_name,email,email_domain,phone,signup_date,plan,country,company,notes
2  1,Alice,Smith,Alice   Smith,alice.smith@example.com,example.com,+85261234567,2024-01-02,Pro,HK,Acme,VIP
3  2,Bob,Jones,Bob Jones,bob.jones[at]mail.com,bob.jones[at]mail.com,0218123344,,Basic,ID,Fuga Ltd,typo email
4  3,Charlie,,Charlie,,,+628120000000,,Pro,ID,,no email
5  4,,,,dora@sample.io,sample.io,,2024-03-05,Pro,HK,Innotech,missing name
6  5,Edward,Nigma,Edward Nigma,edward.nigma@sample.io,sample.io,+14155551212,,Enterprise,US,Riddler Inc.,ok
7
```

The fixes showed 5 duplicated client ID entries merged, invalid email flagged, invalid date flagged, and discarded those with missing email and phone number. Improper phone numbers are normalized with ‘+’ and all emails are lowercased with the right format.

The following is the validation report HTML



http://127.0.0.1:5500/output/validation_report.html

Validation Report

Generated: 2025-07-20T19:16:43

Summary

- Rows in: **7**
- Rows out: **5**
- Dropped (unreachable): **1**
- Invalid emails: **5**
- Invalid signup dates: **3**
- Duration (s): **0.0163**

Invalid Emails

client_id	full_name	email	phone	signup_date	plan	country	company	notes	first_name	last_name
1	Alice Smith	alice.smith@example.com	+852 6123-4567	2024-01-02	Pro	HK	Acme	VIP	Alice	Smith
2	Bob Jones	bob.jones[at]mail.com	(021) 812-3344	01/15/2024	Basic	ID	Fuga Ltd	typo email	Bob	Jones
4		dora@sample.io		2024-03-05	Pro	HK	Innotech	missing name		
5	Edward Nigma	edward.nigma@sample.io	+1 (415) 555-1212	03-05-2024	Enterprise	US	Riddler Inc.	ok	Edward	Nigma
5	Edward Nigma	edward.nigma@sample.io	+1-415-555-1212	2024-03-05	Enterprise	US	Riddler Inc.	dup id & email	Edward	Nigma

Invalid Dates

client_id	full_name	email	phone	signup_date	plan	country	company	notes	first_name	last_name	email_domain
2	Bob Jones	bob.jones[at]mail.com	0218123344	01/15/2024	Basic	ID	Fuga Ltd	typo email	Bob	Jones	bob.jones[at]mail.com
3	Charlie		+628120000000	2024/02/30	Pro	ID		no email	Charlie		
5	Edward Nigma	edward.nigma@sample.io	+14155551212	03-05-2024	Enterprise	US	Riddler Inc.	ok	Edward	Nigma	sample.io

Dropped Unreachable

client_id	full_name	email	phone	signup_date	plan	country	company	notes	first_name	last_name	email_domain
	No Phone Or Email			2024-04-01	Free	HK		drop me	No	Phone Or Email	

Duplicate Email Rows

client_id	full_name	email	phone	signup_date	plan	country	company	notes	first_name	last_name	email_domain	_email_key
5	Edward Nigma	edward.nigma@sample.io	+14155551212	2024-03-05	Enterprise	US	Riddler Inc.	dup id & email	Edward	Nigma	sample.io	edward.nigma@sample.io

Duplicate Email Count

count
1

Duplicate Client Id Rows

None

Duplicate Client Id Count

count
0

The following is the terminal console for running the program.

```
(.venv) fernando@foda Client Data Onboarding Cleaner % time python -m src.cleaner
/Users/fernando/Desktop/Projects/Client Data Onboarding Cleaner/src/cleaner.py:125: UserWarning: The argument 'infer_datetime_format' is deprecated and will be re
moved in a future version. A strict version of it is now the default, see https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You can safely
 remove this argument.
  parsed = pd.to_datetime(df["signup_date"], errors="coerce", dayfirst=False, infer_datetime_format=True)
Cleaning complete.
Rows in: 7 -> Rows out: 5
Wrote: data/clean_clients.csv
Report: output/validation_report.html
python -m src.cleaner  0.28s user 0.09s system 65% cpu 0.567 total
```

It took 0.567 seconds to run the program processing 7 rows. If we extrapolate the performance to about 1000 rows, then it would take 81 seconds.

$$0.567 \div 7 \times 1000 = 81$$

Manual cleaning 1000 rows manually in Excel would typically take 10 minutes at its fastest. This gives an efficiency gain of about 86%.

$$\frac{[(10 \times 60) - 81]}{10 \times 60} \approx 86\%$$

On a larger scale, my project offers a time speedup of 86%.

Conclusion and Project Significance

My Client Data Onboarding Cleaner significantly accelerates client data preparation for SaaS onboarding where time is reduced by 86%, ensures accuracy by automatic flagging and removing of duplicates, maintains consistency in names and formats, presentability in the output being both a CSV file and a HTML report, as well as scalability where the efficiency is maintained for higher number of rows.

For IT/Application support, customer success, and QA/UAT roles, my project demonstrates my ability to automate repetitive data hygiene tasks, ensure clean and reliable datasets for downstream systems, as well as save hours of manual effort while reducing human error.

As a fresh graduate, my project replicates industry-relevant automation in a practical way. Modern SaaS and fintech workflows rely heavily on clean client data for onboarding, compliance, and reporting. By building this cleaner, I showcase my ability to identify real business pain points and deliver a reliable and efficient Python solution that can be directly applied to professional environments.