Row-Level Cleaning Pipeline – Comprehensive Documentation

*(Online Retail II dataset – pandas implementation)*

**Version**: 2025-06-02  **Script**: row\_cleaning.py  **Python** ≥ 3.9

📑 Table of Contents

1. [High-Level Overview](#1-high-level-overview)
2. [Prerequisites & Installation](#2-prerequisites--installation)
3. [Configuration Reference](#3-configuration-reference)
4. [Expected Raw Data Schema](#4-expected-raw-data-schema)
5. [Execution Flow](#5-execution-flow)
6. [Functional Breakdown (Step 0 → 12)](#6-functional-breakdown-step-012)
7. [Row Transformation Logic](#7-row-transformation-logic)
8. [Artefacts & Logs](#8-artefacts--logs)
9. [Troubleshooting & FAQ](#9-troubleshooting--faq)
10. [Extensibility Notes](#10-extensibility-notes)

# **1 High-Level Overview**

|  |  |
| --- | --- |
| **Goal** | Convert raw Online Retail II transactions into an analytically safe table: duplicates removed, placeholder rows dropped, credit-note semantics fixed, monetary outliers capped, derived features added, integrity validated. |
| **Input** | Excel **online\_retail\_II.xlsx** (sheets *Year 2009-2010*, *Year 2010-2011*) – or cached df\_raw\_combined.parquet. |
| **Output** | **orders\_clean.parquet** (fallback **orders\_clean.csv**). <br/>Auxiliary: row\_loss\_waterfall.csv, cap\_thresholds.yml, cleaning\_log.md. |
| **Common use cases** | RFM / cohort analytics, customer-journey modelling, inventory KPI dashboards, ML feature engineering, academic reproducibility. |

# **2 Prerequisites & Installation**

bash

CopyEdit

# Core libraries

pip install pandas numpy scikit-learn pyyaml pyarrow # pyarrow ≈ parquet engine

# Optional: generate diagrams

pip install diagrams mermaid-cli

Python ≥ 3.9 recommended for type-hint support and performance.

# **3 Configuration Reference (config/cleaning.yaml)**

| **Key** | **Default** | **Description** |
| --- | --- | --- |
| DROP\_MISSING\_ID | true | true → drop rows with missing Customer\_ID; false → impute anon\_<country> + add missing\_id flag. |
| PRICE\_CAP\_Q | 0.98 | Upper quantile used to cap **price** on non-credit rows. |
| BASKET\_CAP\_Q | 0.99 | Upper quantile used to cap **basket\_value** on non-credit rows. |
| ISOLATION\_CONTAM | 0.0 | Proportion of multivariate outliers to remove via Isolation Forest. 0 disables Step 11. |

**Tip:** duplicate the YAML under a new name (e.g., fast\_debug.yaml) to experiment safely.

# **4 Expected Raw Data Schema**

| **Column** | **Example** | **Dtype** | **Notes** |
| --- | --- | --- | --- |
| Invoice | 536365 | object | Mixed string / int ➜ coerced to str. |
| StockCode | 85123A | object | Product identifier. |
| Description | WHITE HANGING HEART T-LIGHT HOLDER | object | Free-text; used for placeholder detection. |
| Quantity | 6 | int64 | Negatives = returns. |
| InvoiceDate | 2010-12-01 08:26:00 | datetime64[ns] | UTC assumed. |
| Price | 2.55 | float64 | Unit price. |
| Customer\_ID | 17850 | float64 (nullable) | Customer identifier. |
| Country | United Kingdom | object | Free-text. |

Additional columns may exist; unused ones are preserved unless filtered by steps.

# **5 Execution Flow**

text

CopyEdit

main()

├─ load\_config()

├─ load\_and\_combine\_data() ──┐ (or read cached parquet)

│ │

│ Sheet 2009-2010 ──┐ │

│ Sheet 2010-2011 ──┴─ concat ▼

├─ Step-0 normalise column names

├─ Step-1 collapse duplicates

├─ Step-2 drop placeholders

├─ Step-3 drop extreme prices

├─ Step-4 drop zero-price non-credit

├─ Step-5 fix credit-note prefix

├─ Step-6 drop negative non-credit

├─ Step-7 handle missing customer\_id

├─ Step-8 compute basket value

├─ Step-9 apply caps

├─ Step-10 country bucket

├─ Step-10b cancel flag

├─ Step-11 isolation forest (optional)

├─ Step-12 assertions

├─ save cleaned data (parquet or CSV)

├─ write waterfall, caps, cleaning log

└─ print CLI summary

*Run*:

bash

CopyEdit

python row\_cleaning.py # uses default YAML

python row\_cleaning.py --config config/fast\_debug.yaml # (if CLI wrapper added)

# **6 Functional Breakdown (Step 0 → 12)**

| **Step** | **Function** | **Purpose** | **Input → Output** | **Key Log Metrics** |
| --- | --- | --- | --- | --- |
| **0** | step\_0\_normalize\_columns | Snake-case column headers. | df → df | new column list |
| **1** | step\_1\_collapse\_duplicates | Drop exact duplicates; group by (invoice, stockcode) and **sum quantity**, take **first** of other cols. | → df, rows removed | duplicates removed |
| **2** | step\_2\_drop\_technical\_placeholders | Remove rows with ` | quantity | == 1000\*\*or\*\* description containingTEST |
| **3** | step\_3\_drop\_extreme\_prices | Filter rows with price > 10 000. | → df, rows removed | rows removed |
| **4** | step\_4\_drop\_zero\_price\_non\_credit | Remove zero-price rows unless invoice is credit note (C…). | … | rows removed |
| **5** | step\_5\_fix\_credit\_note\_prefix | If *all* quantities in an invoice are negative and lacks 'C', prepend 'C'. | df → df | invoices fixed |
| **6** | step\_6\_drop\_negative\_non\_credit | Remove negative quantities on non-credit invoices. | … | rows removed |
| **7** | step\_7\_handle\_missing\_customer\_id | *Drop* or *impute* missing IDs per YAML; optional missing\_id flag. | … | rows affected |
| **8** | step\_8\_compute\_basket\_value | basket\_value = quantity × price for positive quantities. | df → df | — |
| **9** | step\_9\_apply\_caps | Cap price and basket\_value at quantile thresholds; return cap dict. | → df, caps | rows capped |
| **10** | step\_10\_create\_country\_bucket | Derive country\_bucket ∈ {UK, EU, NonEU, Unknown}. | df → df | distribution |
| **10b** | step\_10b\_create\_is\_cancelled | Flag credit notes (is\_cancelled). | df → df | cancelled % |
| **11** | step\_11\_isolation\_forest | Optional multivariate outlier removal on *non-credit* rows. | → df, rows removed | outliers removed |
| **12** | step\_12\_assertions | Enforce invariants: no duplicate (invoice, stockcode), positive prices, date range. | df → none | assertion pass |

All step functions are pure (no I/O), easing unit testing and orchestration.

# **7 Row Transformation Logic**

**Ordered Rule Set**

text

CopyEdit

For each row r in df:

0. rename cols → snake\_case

1. if exact duplicate removed earlier (frame-level)

2. if |qty| = 1000 or desc matches /TEST|MANUAL|ADJUST/ ⇒ DROP

3. if price > 10 000 ⇒ DROP

4. if price == 0 and not credit note ⇒ DROP

5. if invoice all neg qty and no 'C' ⇒ prepend 'C' (row value updated)

6. if qty < 0 and not credit note ⇒ DROP

7. if customer\_id missing:

DROP if config.DROP\_MISSING\_ID

else IMPUTE 'anon\_<country>' + missing\_id=1

8. basket\_value = qty\*price if qty>0 else 0

9. if price > price\_cap ⇒ price = cap

if basket\_value > basket\_cap ⇒ basket\_value = cap

10. derive country\_bucket

10b. is\_cancelled = invoice.startswith('C')

11. if multivariate outlier (iso-forest) ⇒ DROP

12. final dataframe must satisfy assertions

**Dual Worked Examples**

|  | **Raw (sale)** | **Cleaned** |  | **Raw (credit note)** | **Cleaned** |
| --- | --- | --- | --- | --- | --- |
| Invoice | 536365 | 536365 |  | 536540 | C536540 *(prefix added)* |
| Quantity | 6 | 6 |  | -2 | -2 *(kept; credit note)* |
| Price | 12 000 | *row dropped* *(Step 3)* |  | -1.95 | -1.95 |
| Customer\_ID | NaN | *dropped* or anon\_united\_kingdom |  | 15311 | 15311 |
| basket\_value | — | — |  | — | -3.90 |
| country\_bucket | — | UK |  | — | UK |
| is\_cancelled | — | 0 |  | — | 1 |

# **8 Artefacts & Logs**

| **File** | **Location** | **What It Contains** |
| --- | --- | --- |
| orders\_clean.parquet | data/processed/ | Final cleaned dataset. |
| row\_loss\_waterfall.csv | docs/cleaning/ | Row count after each step (easy plotting). |
| cap\_thresholds.yml | docs/cleaning/ | Computed price\_cap, basket\_cap, row counts capped. |
| cleaning\_log.md | docs/cleaning/ | Human-readable summary (suitable for audit). |
| Standard log stream | console + logger | INFO-level messages for each step. Use export LOGLEVEL=DEBUG for more detail. |

# **9 Troubleshooting & FAQ**

| **Symptom** | **Likely Cause** | **Fix** |
| --- | --- | --- |
| **AssertionError: duplicate (invoice, stockcode)** | Upstream data changed; Step 1 aggregation keys altered. | Inspect duplicates via df[df.duplicated(['invoice','stockcode'], keep=False)]. |
| **Parquet engine not available** | pyarrow or fastparquet missing. | pip install pyarrow. |
| **MemoryError on load** | Large dataset + 32-bit Python or low RAM. | Load via chunks (pd.read\_csv(..., chunksize=...)) or run on 64-bit Python with sufficient memory. |
| **Isolation Forest removal too aggressive** | ISOLATION\_CONTAM too high. | Lower contamination (e.g., 0.005) or set to 0. |

# **10 Extensibility Notes**

* **Orchestration** – Each step\_n\_\* is stateless; wrap them in Airflow/Dagster nodes.
* **Spark adaptation** – Replace pandas ops with PySpark equivalents; group-by semantics remain.
* **Unit tests** – Provide a 5-row fixture per edge case and assert post-conditions for each step.
* **CLI flags** – Consider adding argparse to override YAML at runtime (--price-cap-q 0.97).
* **Docs automation** – Generate this file from docstrings via MkDocs + mkdocstrings during CI.

**You’re set!** Run the script, inspect the generated artefacts, and plug the cleaned parquet into your analytics or ML pipeline. For questions or contributions, open an issue in the repo