Cleaning Messy Data Without Stress

Processing Real-World Datasets Using Automated Jupyter Pipelines

Part 1: Sourcing & Ingesting

Accessing public business payloads directly in Python

The UCI Online Retail Dataset

Target Data Payload

This dataset contains 12 months of actual transactional history from a UK-based e-commerce storefront, providing a massive volume of raw operational records.

Resource URL:
`https://archive.ics.uci.edu/ml/machine-learning-databases/00352/`

Legacy Layout Flaws

Because it comes straight from an uncurated corporate server export, the dataset contains real errors: null tracking IDs, empty item categories, negative purchases, and erratic text characters.

Ingesting the Dataset via Code

Streamlined Ingestion

Because the source file is packaged inside an Excel workbook container, we pull it from the web and read it using standard Pandas wrappers.

By chaining read_excel(), Pandas instantly interprets tabular boundaries and columns in your notebook.

```
import pandas as pd

# Sourcing live Excel dataset from URL
url = "https://archive.ics.uci.edu/ml/machine-learning-
      databases/00352/Online%20Retail.xlsx"
# Pulling raw transactional records
df_raw = pd.read_excel(url)


# Verifying dimensions
print(df_raw.shape)
```

Before Cleansing: Inspecting Raw Rows

InvoiceNo	StockCode	Description	Quantity	UnitPrice	CustomerID
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2.55	17850.0
C536379	D	Discount	**-1**	27.50	14527.0
536414	22139	*NaN* (Missing Text)	1	0.00	**NaN (Guest Checkout)**
536544	21773	DECORATIVE ROSE	1	1.25	**NaN (Guest Checkout)**

**The raw checkout data shows negative quantities (cancellations), null customer IDs, and trailing spaces.*

Part 2: Cleansing Pipelines

Using vector routines to align profiles and isolate values

Standardizing Critical Records

1. Drop Null Profiles

Filter records where CustomerID is blank, removing guest sessions that lack customer matching IDs.

2. Filter Negative Rows

Isolate cancels by setting strict numeric boundaries: Quantity > 0 and UnitPrice > 0.

3. Strip Text Spacing

Clean categorical names via string vectors using str.strip() and str.upper() wrappers.

The Complete Pipeline Code

```
# 1. Standardize string spacing
df_raw['Description'] = df_raw['Description'].str.strip().s


# 2. Exclude customerless transaction rows
df_clean = df_raw.dropna(subset=['CustomerID'])


# 3. Limit dataset to actual product sales
df_clean = df_clean[
    (df_clean['Quantity'] > 0) &
    (df_clean['UnitPrice'] > 0)
]


# Verify final rows count
print(df_clean.shape)
```

Why Chaining Works

Because Pandas processes operations natively in memory, these vectorized statements bypass traditional slow loops.

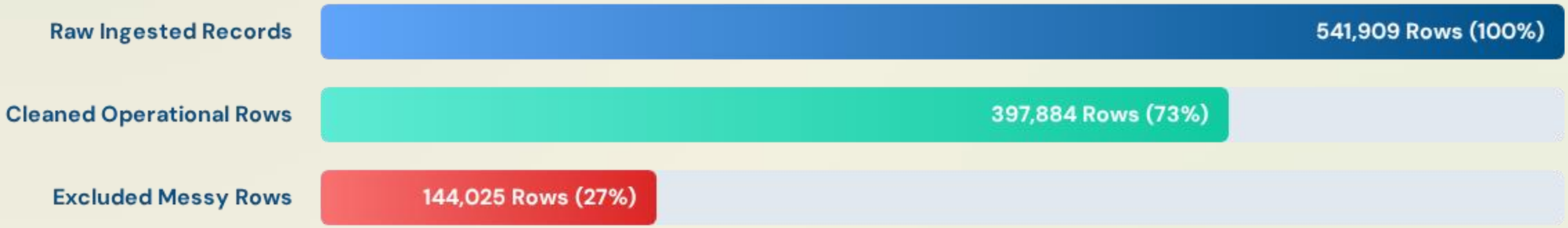
We instantly isolate legitimate customer purchases and standardize category names in under a second.

Post-Cleansing: Normal Layout Schema

InvoiceNo	StockCode	Cleaned Description	Quantity	UnitPrice	CustomerID
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2.55	17850.0
536544	21773	DECORATIVE ROSE	1	1.25	13421.0
536592	22960	JAM MAKING SET WITH JARS	12	3.75	12433.0
536601	22139	RED RETROSPOT CUP	8	4.25	15688.0

✓All negative values, missing customer indices, and chaotic spacing variants have been resolved.

Data Ingestion Effectiveness

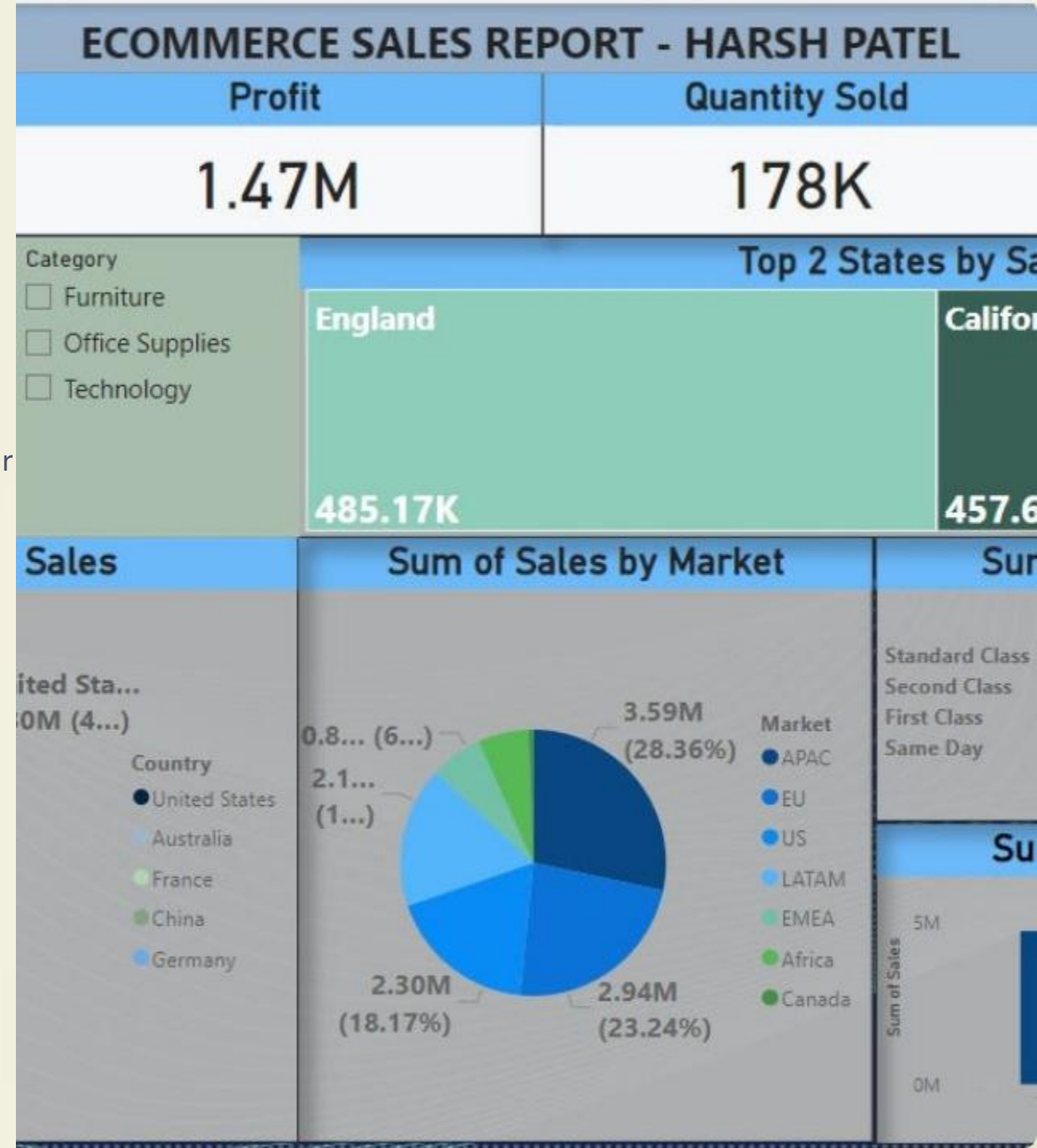


Our pipeline extracted 397,884 validated rows for analytics, eliminating 144,025 invalid or incomplete database entries.

Ready for Strategy

By routing uncurated online datasets through clean, automated pipelines, you immediately secure your downstream business intelligence models.

With correct customer identifiers and positive sales records, your cohort models, market basket algorithms, and customer lifetime value forecasts are highly reliable.

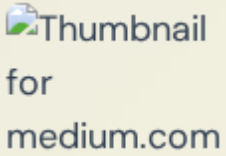


Questions?

Module 3: Data Cleansing Best Practices

www.anaconda.com | support@anaconda.com

Image Sources



https://miro.medium.com/1*nbq-5l_QW_-pL9JuHvPLow.jpeg

Source: medium.com