# Project: E-commerce Sales Analytics Dashboard

The UCI Online Retail dataset contains transactional data from a UK-based online gift retailer (Dec 2010–Dec 2011), with 541,909 rows and 8 columns: InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country. Download from https://archive.ics.uci.edu/dataset/352/online+retail (Excel format, ~25MB); split into 4 logical tables (<30k rows each after cleaning): invoices (~400k rows), customers (~4k unique), products (~4k unique), countries (37). Contains 30% missing CustomerID, negative quantities for returns.

### Problem Statement

Analyze UK online retail transactions to identify customer purchasing patterns, product performance, sales trends, and profitability. Compute RFM segmentation, detect seasonal peaks, quantify top products/customers, and handle data issues like missing values/duplicates while validating SQL vs. pandas results.

### Dataset Details

**Source:** UCI ML Repository (Online Retail) – real transactional data, no PII.

- **Format**: Excel (.xlsx) – one sheet, denormalized.
- **Size:** 541,909 transactions; clean to ~400k valid rows.
- **Key Stats:** £8.19M total revenue, 4,332 customers, 4,068 products, 97% UK sales.
- **Columns & Types:**

| Column | Type | Description | Sample |
|---|---|---|---|
| InvoiceNo | Varchar | Transaction ID (‘C’ prefix = cancel) | 536365 |
| StockCode | Varchar | Product code | 85123A |
| Description | Varchar | Product name | WHITE HANGING HEART T-LIGHT HOLDER |
| Quantity | Integer | Items per transaction (± for returns) | 6 |
| InvoiceDate | Datetime | Transaction timestamp | 12/1/2010 8:26 |
| UnitPrice | Float | £ per unit | 2.55 |
| CustomerID | Integer | Unique customer (30% null) | 17850 |
| Country | Varchar | Billing country | United Kingdom |

## Technology/Tools

- Microsoft SQL Server + Python 3.x (pandas, SQLAlchemy/pyodbc).
- Jupyter Notebook (matplotlib/seaborn for viz).
- GitHub for version control.

## High-Level Instructions

1) **Data Prep (Pandas):** Load Excel → drop null CustomerID → filter Quantity>0 & UnitPrice>0 → remove 'C' invoices → engineer Revenue=Quantity×UnitPrice.
2) **SQL Load:** Create 4 tables (invoices, customers, products, countries) with PK/FK → bulk insert via `df.to_sql()`.
3) **SQL Analysis:** Write 8+ queries for KPIs below.
4) **Pandas Validation:** Replicate SQL results → compare (assert <1% diff).
5) **Viz & Report:** Dashboard with 6 charts → business insights.

## KPIs & Calculations

**RFM (Recency, Frequency, Monetary) – Reference date: 2011-12-10**

- **Recency (R):** Days since last purchase = `DATEDIFF(day, MAX(InvoiceDate), '2011-12-10')` → score 1-5 (1=recent).
- **Frequency (F):** Invoice count = `COUNT(DISTINCT InvoiceNo)`.
- **Monetary (M):** Total revenue = `SUM(Quantity * UnitPrice)`.
- **Scoring:** Quintiles (SQL `NTILE(5)` or pandas `qcut`) → Champions (555), At Risk (311), etc.

## Expected Results/Evaluation Criteria

- **Data Quality:** Pre: 541k rows → Post: 397k valid (73%) with schema diagram.
- **SQL:** 8 queries using JOINs, CTEs, window functions (LAG for MoM growth).
- **Pandas:** ETL <5min, RFM table (4k rows), correlation matrix.
- **Accuracy:** SQL=pandas results (±0.5%); RFM segments match (e.g., 200 Champions).
- **Insights:** 5 actionable findings like "France RFM 555s = 20% revenue despite 10% customers."
- **Deliverables:** GitHub repo (Jupyter/SQL notebook, ERD, dashboard PNGs, README).
- **Bonus:** Indexes on SQL tables → query time <1s; Streamlit app.

**Success:** Reproducible analysis matching industry benchmarks (total revenue £8.19M).