Sales and Transaction Analytics Dashboard

Fernando Djingga
18 August 2025

Introduction

I created a lightweight automated sales and transaction analytics dashboard that transforms raw data into decision-ready insights. My program ingests two CSV files – one containing customer records and another containing transaction records – then processes, aggregates, and exports professional outputs in two forms:

1. dashboard.xlsx – an Excel workbook containing key performance index (KPI) cards, pivot-style summaries, and three interactive charts (Revenue by Month, Top Products by Revenue, and Revenue by Segment/Channel). This is suitable for analysts, managers, or executives accustomed to Excel reporting.

2. summary.html – a clean, browser-ready report containing the same KPI cards and summary tables, designed for easy stakeholder sharing without requiring Excel.

By automating KPI computation, pivot creation, and chart generation, my project should improve the efficiency of the reports for business analysts, finance teams, and product operations.

High-Level Implementation

My design of the pipeline follows the natural workflow of how transaction analytics are typically carried out in business environments. For data inputs, 2 datasets are needed. The customer dataset in 'customers.csv' includes the 'customer_id', 'signup_date', 'segment', and 'channel'. The transaction dataset in 'transactions.csv' includes the 'txn_id', 'txn_date', 'customer_id', 'product', 'quantity', 'unit_price', and 'currency'. During input, my script should parse the dates, computes amount as the product between 'quantity' and 'unit_price', and derives 'month' and 'cohort_month'.

My program will calculate 5 essential business KPIs: total revenue (overall sales value), orders (number of distinct transaction IDs), customers (number of unique customer IDs), average order value (average transaction revenue), and the repeat order rate (percentage or orders placed by returning customers. To support deeper insights, my program also builds grouped summaries of the revenue by month, top products by revenue, and revenue by segment and channel, as well as details regarding new and returning customers (based on first purchase cohort analysis).

As my output, I designed an Excel file to replicate a dashboard sheet with KPI cards at the top and three charts placed clearly below. Additional sheets ('MonthlyRevenue', 'TopProducts', 'SegmentChannel', 'Transactions') that hold raw tables for pivot or chart references are also set up where charts are dynamically linked to data sheets to ensure that updates are automatic. I also included a HTML summary file that includes the styled KPI cards using CSS and tabular breakdowns of monthly revenue, top products, and revenue by segment/channel.

Coding Implementation

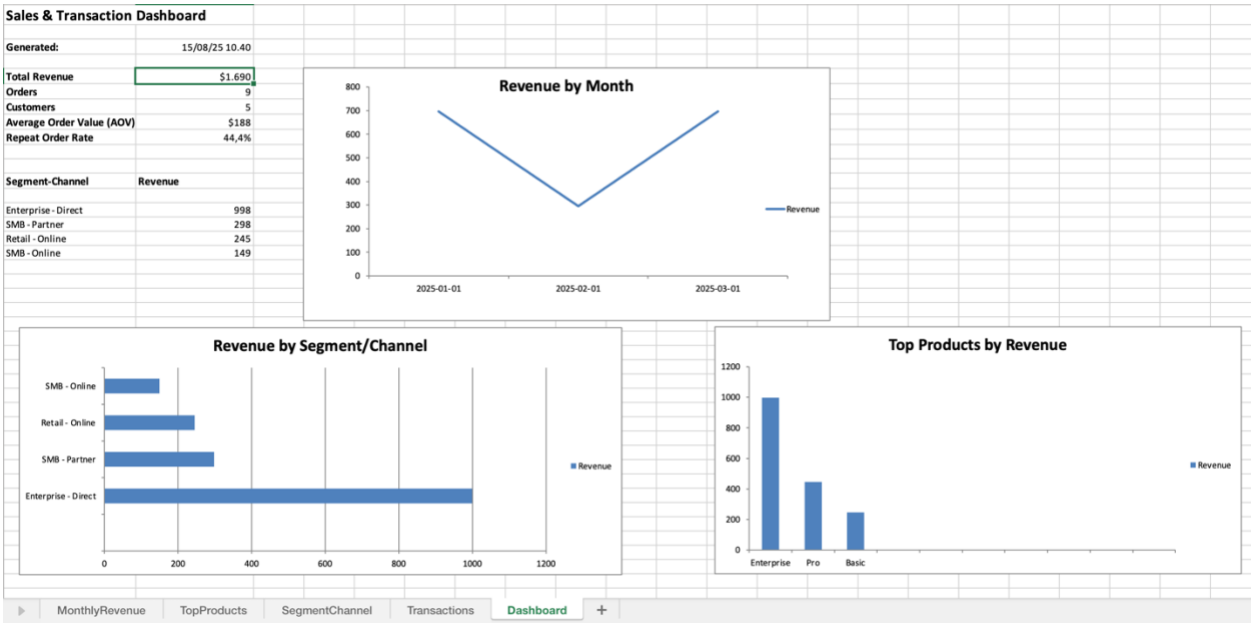
My project code is structured as a simple and readable way. ‘analyze.py’ is the main pipeline script. ‘ensure_dirs()’ ensures that the ‘data’ and ‘output’ folders exists. ‘seed_sample_data()’ generates small demo datasets if none are found. ‘load_data()’ reads the CSVs using pandas, parses, and enriches data. ‘compute_kpis()’ calculates KPIs and grouped aggregates. ‘write_excel()’ builds the Excel dashboard using ‘xlsxwriter’ with KPI cards and charts, while ‘write_html()’ generates the HTML summary with CSS styling.

‘query.sql’ contains the example SQL queries for databases, which reports the monthly revenue trend, average order value by month, top products by revenue, new and returning customer analysis, and cohort revenue by signup month. I included the sample data files ‘transactions.csv’ and ‘customers.csv’. The output generated are ‘dashboard.xlsx’ and ‘summary.html’.

Instructions to run my program is found in README.md.

Project Results

The following is the Excel dashboard generated by my program.



The dashboard sheet has KPI cards and three charts, data sheets for transparency and reproducibility and consistent formatting across the runs.

The following is the HTML summary of my program, which is browser-viewable and shared easily.

Sales & Transaction Summary

Generated on 2025-09-30 10:40

Total Revenue	Orders	Customers	AOV	Repeat Order Rate
$1,690	9	5	$188	44.4%

Revenue by Month

month	revenue
2025-01-01	697.0
2025-02-01	296.0
2025-03-01	697.0

Top Products

product	units_sold	revenue
Enterprise	2	998.0
Pro	3	447.0
Basic	5	245.0

Revenue by Segment & Channel

segment	channel	revenue	orders
Enterprise	Direct	998.0	2
SMB	Partner	298.0	2
Retail	Online	245.0	4
SMB	Online	149.0	1

The following is the terminal console output after running my program.

```
(.venv) fernando@foda Sales and Transaction Analytics Dashboard % python src/analyze.py
Done.
- Excel:  /Users/fernando/Desktop/Projects/Sales and Transaction Analytics Dashboard/output/dashboard.xlsx
- HTML:   /Users/fernando/Desktop/Projects/Sales and Transaction Analytics Dashboard/output/summary.html
(.venv) fernando@foda Sales and Transaction Analytics Dashboard % /usr/bin/time -p python src/analyze.py
Done.
- Excel:  /Users/fernando/Desktop/Projects/Sales and Transaction Analytics Dashboard/output/dashboard.xlsx
- HTML:   /Users/fernando/Desktop/Projects/Sales and Transaction Analytics Dashboard/output/summary.html
real 0.94
user 0.62
sys 0.10
```

The script runtime is about 0.94 seconds. It takes about 5 minutes to do a final practical review before it can be shared. When done manually, the process would take about 60 minutes to collect the data in Excel, create the charts, and format the results. This meant that 55 minutes would be saved per cycle, and that there is an efficiency gain of about 91.7% faster.

If considering script runtime alone, the improvement is about 99.99% or 3800 times faster, but the reported improvement includes realistic human review for better credibility of results.

Conclusion and Project Significance

In conclusion, my Sales and Transaction Analytics Dashboard successfully reduced reporting effort from about 60 minutes of manual Excel work to only 5 minutes end-to-end with automation. My tool provides time savings of 55 minutes per reporting cycle (91.7% faster), maintains accuracy and reproducibility, standardized KPI definitions and the chart layouts, as well as high scalability, portability, usability, and shareability.

For finance, SaaS, or operations teams, my project demonstrates how automation can eliminate repetitive reporting tasks, reduce the risk of manual errors, and free analysts to focus on insights rather than mechanics.

As a fresh graduate, I designed this project with industry workflows in mind. Monthly sales reporting is a universal need across sectors. Therefore, by automating the repetitive portions of this process, my dashboard bridges the gap between raw transactional data and actionable business intelligence. By delivering both machine-readable outputs (CSV, JSON) and human-friendly deliverables (Excel, HTML), my project shows how a single pipeline can support multiple purposes or stakeholders. The result is faster reporting cycles, more accurate KPIs, and improved stakeholder confidence.