NATIONAL UNIVERSITY – LAGUNA
Km. 53, National Highway, Brgy. Milagrosa, Calamba City, Laguna

SCHOOL OF ENGINEERING AND ARCHITECTURE
COMPUTER ENGINEERING DEPARTMENT
1st Trimester – 2025-2026
CPELX230 – ELECTIVE 2

Panda-Based Analysis of Bakery Sales: Evidence from a One-Week Dataset

Submitted by:
Jandicala, Elroy A.
Pore, Christine Gee C.

BS CpE221-A

Oct 6, 2025

Objectives

- Load a self-created, Excel-style bakery dataset into Pandas and perform initial EDA (head, tail, describe, info).
- Automatically compute Total Sales**,** Total Cost, and Profit from the raw fields.
- Display week-level totals for sales, cost, and profit.
- Rank products by quantity and by profit and visualize “most/least” results with bar charts.

Content

INTRODUCTION

This study analyzes a one-week point-of-sale dataset from a small bakery to surface operational insights about product demand and profitability. The input is a comma-separated values (CSV) file, bread_sales_week.csv, containing the fields Date, Product, Price, Cost, and Quantity. The analysis system is a lightweight Python program built with Pandas and Matplotlib that (i) ingests the CSV, (ii) derives business-ready features, and (iii) exposes a simple, reproducible command-line interface for exploration and reporting.

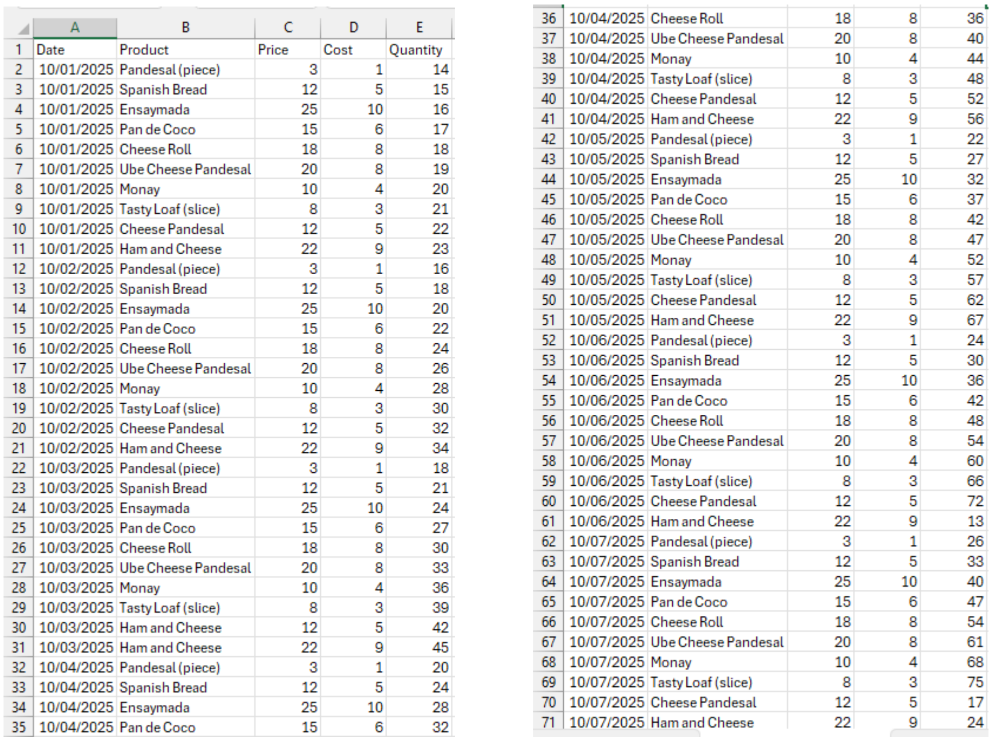
At startup, the program reads the dataset via a single configuration constant (CSV_FILE) and pd.read_csv, ensuring that the analysis remains source-controlled and easily swappable across weeks or branches of data. To support revenue and margin calculations, the code materializes three engineered columns—Total Sales (= Price×Quantity), Total Cost (= Cost×Quantity), and Profit (= Total Sales−Total Cost)—through an idempotent helper function ensure_totals(). This guarantees that higher-level views never fail if a user jumps directly to totals or profitability outputs.

The user interface is intentionally minimal: a 12-option text menu provides access to basic EDA (head, tail, describe, info), one-click weekly KPIs (Sales, Cost, Profit), and product-level rankings with bar charts for "most/least sold" and "most/least profitable," plus a clean Exit. This design keeps the workflow consistent with a short, narrated demonstration while producing screenshot-ready outputs aligned with the assignment's format. The choice of a plain CSV and a small Python script is deliberate: it satisfies the brief's requirement for an Excel-style example presented by the student, while emphasizing originality, transparency of calculations, and ease of replication for future weeks of bakery operations.

DISCUSSION OF EXAMPLES

CVS Dataset

This project analyzes a one-week transaction snapshot from a small bakery using a compact, Excel-style table. The dataset is stored in **bread_sales_week.csv** and contains five fields: **Date, Product, Price, Cost, and Quantity**. This schema is intentionally minimal: each row represents one product's sales entry on a specific date, with unit-level economics captured by **Price** and **Cost**, and demand captured by **Quantity**.



	A	B	C	D	E
1	Date	Product	Price	Cost	Quantity
2	10/01/2025	Pandesal (piece)	3	1	14
3	10/01/2025	Spanish Bread	12	5	15
4	10/01/2025	Ensaymada	25	10	16
5	10/01/2025	Pan de Coco	15	6	17
6	10/01/2025	Cheese Roll	18	8	18
7	10/01/2025	Ube Cheese Pandesal	20	8	19
8	10/01/2025	Monay	10	4	20
9	10/01/2025	Tasty Loaf (slice)	8	3	21
10	10/01/2025	Cheese Pandesal	12	5	22
11	10/01/2025	Ham and Cheese	22	9	23
12	10/02/2025	Pandesal (piece)	3	1	16
13	10/02/2025	Spanish Bread	12	5	18
14	10/02/2025	Ensaymada	25	10	20
15	10/02/2025	Pan de Coco	15	6	22
16	10/02/2025	Cheese Roll	18	8	24
17	10/02/2025	Ube Cheese Pandesal	20	8	26
18	10/02/2025	Monay	10	4	28
19	10/02/2025	Tasty Loaf (slice)	8	3	30
20	10/02/2025	Cheese Pandesal	12	5	32
21	10/02/2025	Ham and Cheese	22	9	34
22	10/03/2025	Pandesal (piece)	3	1	18
23	10/03/2025	Spanish Bread	12	5	21
24	10/03/2025	Ensaymada	25	10	24
25	10/03/2025	Pan de Coco	15	6	27
26	10/03/2025	Cheese Roll	18	8	30
27	10/03/2025	Ube Cheese Pandesal	20	8	33
28	10/03/2025	Monay	10	4	36
29	10/03/2025	Tasty Loaf (slice)	8	3	39
30	10/03/2025	Ham and Cheese	12	5	42
31	10/03/2025	Ham and Cheese	22	9	45
32	10/04/2025	Pandesal (piece)	3	1	20
33	10/04/2025	Spanish Bread	12	5	24
34	10/04/2025	Ensaymada	25	10	28
35	10/04/2025	Pan de Coco	15	6	32
36	10/04/2025	Cheese Roll	18	8	36
37	10/04/2025	Ube Cheese Pandesal	20	8	40
38	10/04/2025	Monay	10	4	44
39	10/04/2025	Tasty Loaf (slice)	8	3	48
40	10/04/2025	Cheese Pandesal	12	5	52
41	10/04/2025	Ham and Cheese	22	9	56
42	10/05/2025	Pandesal (piece)	3	1	22
43	10/05/2025	Spanish Bread	12	5	27
44	10/05/2025	Ensaymada	25	10	32
45	10/05/2025	Pan de Coco	15	6	37
46	10/05/2025	Cheese Roll	18	8	42
47	10/05/2025	Ube Cheese Pandesal	20	8	47
48	10/05/2025	Monay	10	4	52
49	10/05/2025	Tasty Loaf (slice)	8	3	57
50	10/05/2025	Cheese Pandesal	12	5	62
51	10/05/2025	Ham and Cheese	22	9	67
52	10/06/2025	Pandesal (piece)	3	1	24
53	10/06/2025	Spanish Bread	12	5	30
54	10/06/2025	Ensaymada	25	10	36
55	10/06/2025	Pan de Coco	15	6	42
56	10/06/2025	Cheese Roll	18	8	48
57	10/06/2025	Ube Cheese Pandesal	20	8	54
58	10/06/2025	Monay	10	4	60
59	10/06/2025	Tasty Loaf (slice)	8	3	66
60	10/06/2025	Cheese Pandesal	12	5	72
61	10/06/2025	Ham and Cheese	22	9	13
62	10/07/2025	Pandesal (piece)	3	1	26
63	10/07/2025	Spanish Bread	12	5	33
64	10/07/2025	Ensaymada	25	10	40
65	10/07/2025	Pan de Coco	15	6	47
66	10/07/2025	Cheese Roll	18	8	54
67	10/07/2025	Ube Cheese Pandesal	20	8	61
68	10/07/2025	Monay	10	4	68
69	10/07/2025	Tasty Loaf (slice)	8	3	75
70	10/07/2025	Cheese Pandesal	12	5	17
71	10/07/2025	Ham and Cheese	22	9	24

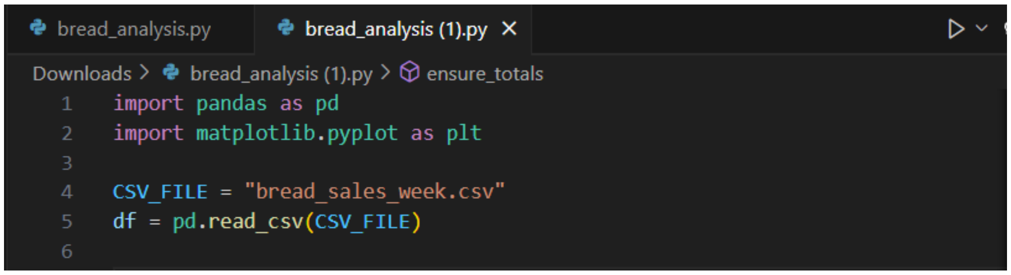
Figure 1 - CVS Content

Basic exploratory checks confirm that the table is analysis ready. Using the program's menu options, the first rows (**df.head()**) and the last rows (**df.tail()**) provide a quick visual sanity check on dates, product names, and numeric ranges; summary statistics via **df.describe()** show reasonable central tendencies (e.g., a mid-teens price point), and **df.info()** reports a complete, 70-row table with the expected dtypes (object for labels, integers for numeric columns). These four views are exposed directly in the CLI to make data validation part of the workflow rather than a hidden step.

EXPLANATION OF EXAMPLES

Inputs & setup

At startup, the Python program reads the CSV designated by the CSV_FILE constant and binds it to a Pandas DataFrame (df) using pd.read_csv. Keeping the source path as a single constant makes the pipeline reproducible and easy to switch to a new week's file without code changes.

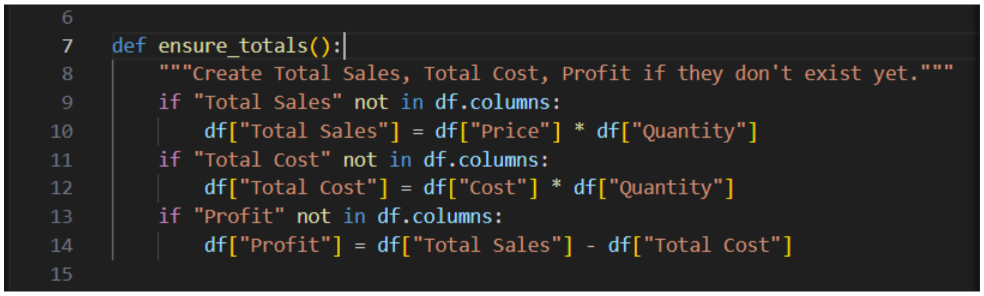


```python
import pandas as pd
import matplotlib.pyplot as plt

CSV_FILE = "bread_sales_week.csv"
df = pd.read_csv(CSV_FILE)

```

Figure 2 – Input and Setup Function

Feature Engineering

Feature engineering is centralized in an idempotent helper, **ensure_totals()**, which creates the three derived columns only if they are missing: **Total Sales (= Price×Quantity)**, **Total Cost (= Cost×Quantity)**, and **Profit (= Total Sales−Total Cost)**. Calling this function before totals or profit-based rankings guarantees the program still works if a user jumps straight to those outputs.

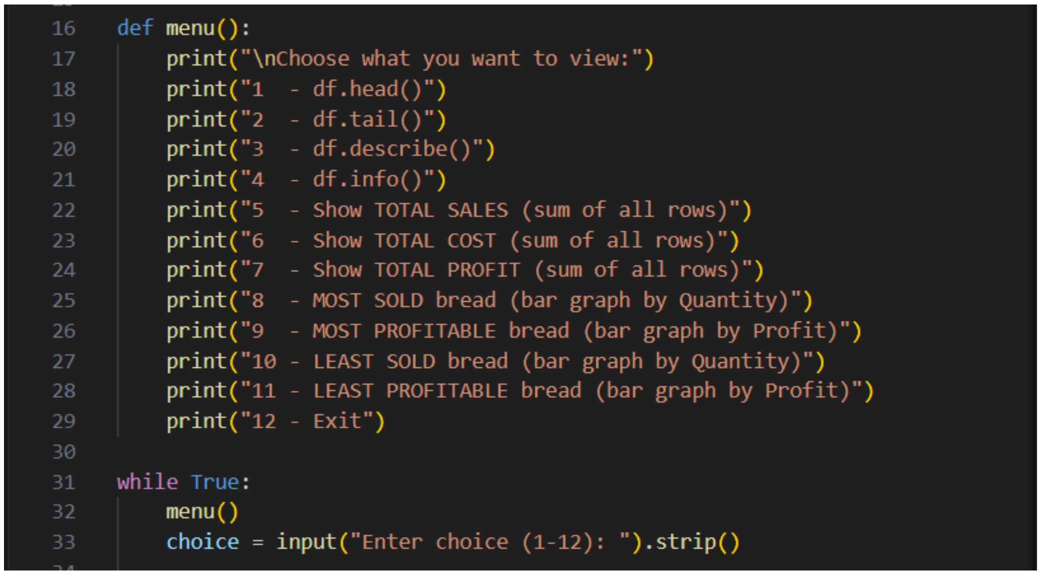


```python

def ensure_totals():
    """Create Total Sales, Total Cost, Profit if they don't exist yet."""
    if "Total Sales" not in df.columns:
        df["Total Sales"] = df["Price"] * df["Quantity"]
    if "Total Cost" not in df.columns:
        df["Total Cost"] = df["Cost"] * df["Quantity"]
    if "Profit" not in df.columns:
        df["Profit"] = df["Total Sales"] - df["Total Cost"]

```

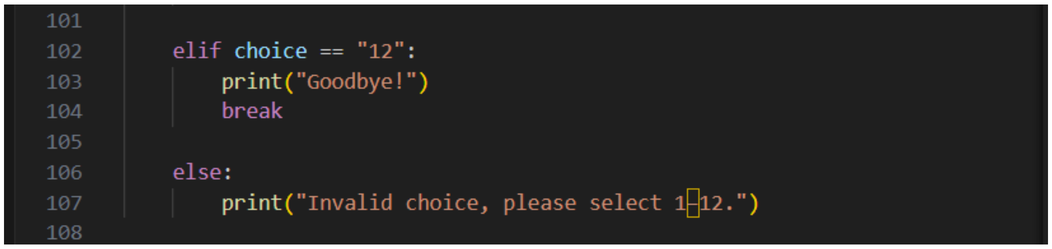
Figure 2.1 – Idempotent Helper Function

User interaction

User interaction is provided through a 12-option command-line menu that loops until exit. Options 1–4 expose the EDA views; 5–7 print week-level Sales, Cost, and Profit after ensuring engineered columns exist; 8–11 perform product-level groupby/sort operations and render labeled bar charts for "Most/Least Sold" (Quantity) and "Most/Least Profitable" (Profit); 12 exits. This visible sequence mirrors the "Demonstration of the Code in PANDA" step in the required content flow.



```python
def menu():
    print("\nChoose what you want to view:")
    print("1  - df.head()")
    print("2  - df.tail()")
    print("3  - df.describe()")
    print("4  - df.info()")
    print("5  - Show TOTAL SALES (sum of all rows)")
    print("6  - Show TOTAL COST (sum of all rows)")
    print("7  - Show TOTAL PROFIT (sum of all rows)")
    print("8  - MOST SOLD bread (bar graph by Quantity)")
    print("9  - MOST PROFITABLE bread (bar graph by Profit)")
    print("10 - LEAST SOLD bread (bar graph by Quantity)")
    print("11 - LEAST PROFITABLE bread (bar graph by Profit)")
    print("12 - Exit")

while True:
    menu()
    choice = input("Enter choice (1-12): ").strip()
```

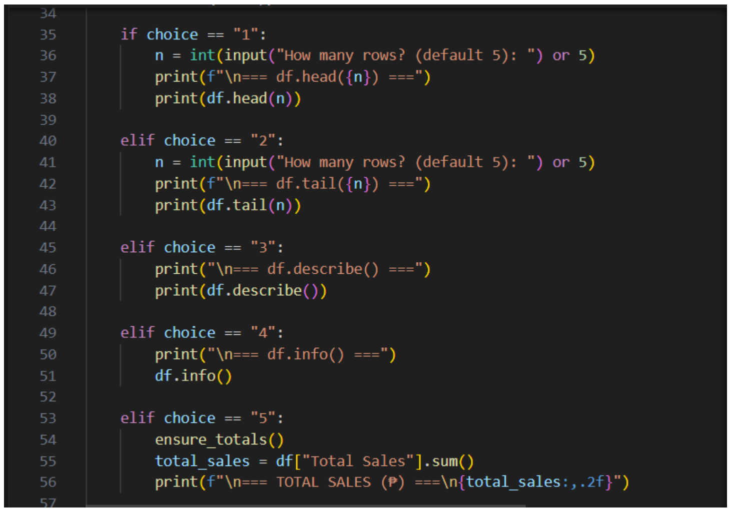


```python

    elif choice == "12":
        print("Goodbye!")
        break

    else:
        print("Invalid choice, please select 1–12.")

```

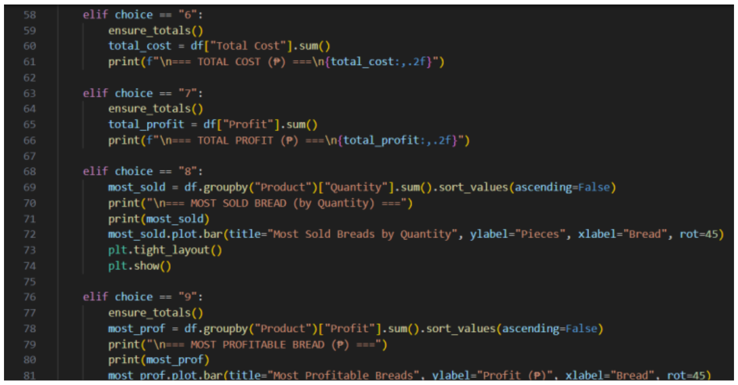
Figure 2.3 – User Interface Function

Algorithmic pattern

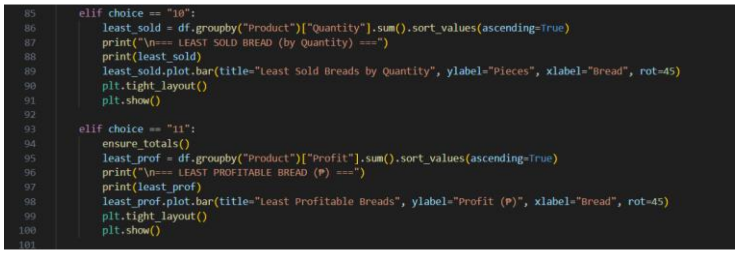
Algorithmically, week-level KPIs are simple sums on engineered fields (df["Total Sales"].sum(), etc.), and rankings come from df.groupby("Product")[metric].sum().sort_values(ascending=...), which feed directly into .plot.bar(...) to produce the figures used in your results. Titles, axis labels, and rotated x-ticks are set for readability. This compact pattern keeps the implementation transparent and ensures every printed table and chart in the paper can be traced back to a short, declarative block of code.



```python
    if choice == "1":
        n = int(input("How many rows? (default 5): ") or 5)
        print(f"\n=== df.head({n}) ===")
        print(df.head(n))

    elif choice == "2":
        n = int(input("How many rows? (default 5): ") or 5)
        print(f"\n=== df.tail({n}) ===")
        print(df.tail(n))

    elif choice == "3":
        print("\n=== df.describe() ===")
        print(df.describe())

    elif choice == "4":
        print("\n=== df.info() ===")
        df.info()

    elif choice == "5":
        ensure_totals()
        total_sales = df["Total Sales"].sum()
        print(f"\n=== TOTAL SALES (₱) ===\n{total_sales:,.2f}")
```



```python
    elif choice == "6":
        ensure_totals()
        total_cost = df["Total Cost"].sum()
        print(f"\n=== TOTAL COST (₱) ===\n{total_cost:,.2f}")

    elif choice == "7":
        ensure_totals()
        total_profit = df["Profit"].sum()
        print(f"\n=== TOTAL PROFIT (₱) ===\n{total_profit:,.2f}")

    elif choice == "8":
        most_sold = df.groupby("Product")["Quantity"].sum().sort_values(ascending=False)
        print("\n=== MOST SOLD BREAD (by Quantity) ===")
        print(most_sold)
        most_sold.plot.bar(title="Most Sold Breads by Quantity", ylabel="Pieces", xlabel="Bread", rot=45)
        plt.tight_layout()
        plt.show()

    elif choice == "9":
        ensure_totals()
        most_prof = df.groupby("Product")["Profit"].sum().sort_values(ascending=False)
        print("\n=== MOST PROFITABLE BREAD (₱) ===")
        print(most_prof)
        most_prof.plot.bar(title="Most Profitable Breads", ylabel="Profit (₱)", xlabel="Bread", rot=45)
```



```python
    elif choice == "10":
        least_sold = df.groupby("Product")["Quantity"].sum().sort_values(ascending=True)
        print("\n=== LEAST SOLD BREAD (by Quantity) ===")
        print(least_sold)
        least_sold.plot.bar(title="Least Sold Breads by Quantity", ylabel="Pieces", xlabel="Bread", rot=45)
        plt.tight_layout()
        plt.show()

    elif choice == "11":
        ensure_totals()
        least_prof = df.groupby("Product")["Profit"].sum().sort_values(ascending=True)
        print("\n=== LEAST PROFITABLE BREAD (₱) ===")
        print(least_prof)
        least_prof.plot.bar(title="Least Profitable Breads", ylabel="Profit (₱)", xlabel="Bread", rot=45)
        plt.tight_layout()
        plt.show()
```

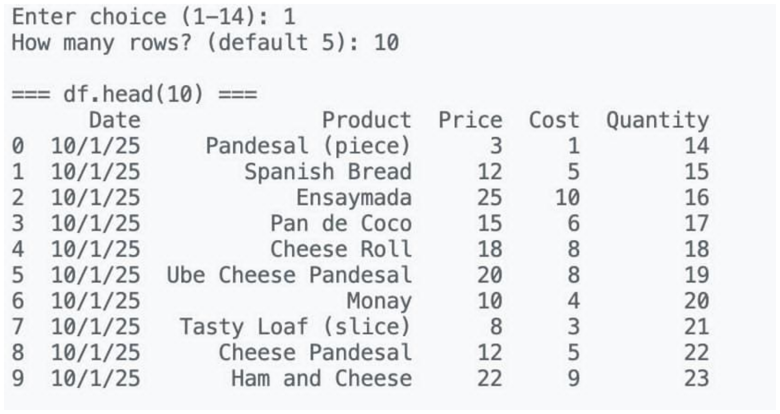
Figure 2.4 – Algorithm Pattern

DEMONSTRATION OF THE CODE IN PANDA

Quick EDA (Menu 1–4)

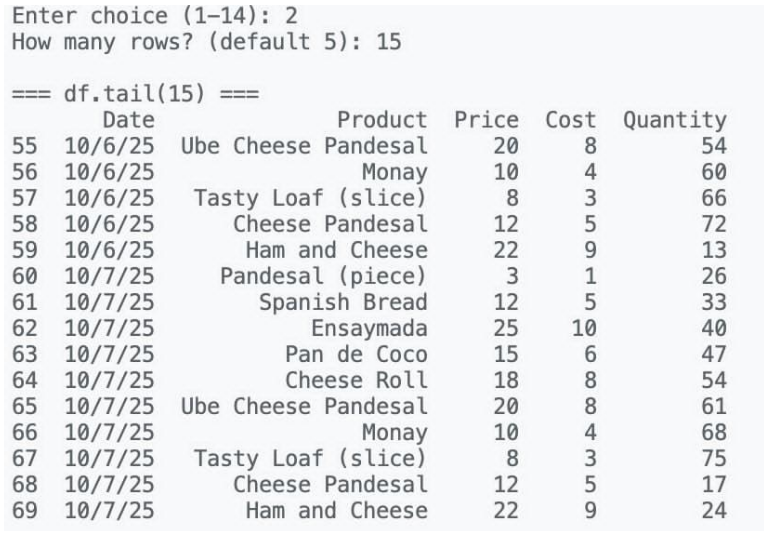
Goal: prove the data is complete and reasonable before calculations (you are the one explaining—keep narration clear).

1. **df.head():** "Here are the first rows for 10/1/25; columns are Date, Product, Price, Cost, Quantity."



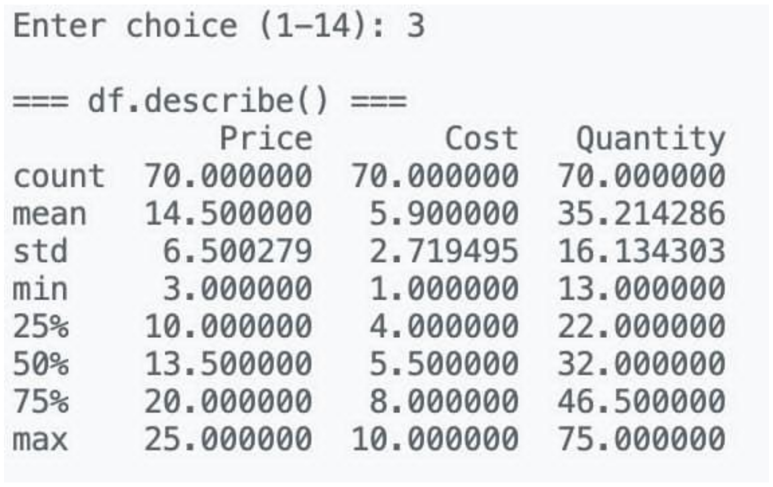
```
Enter choice (1-14): 1
How many rows? (default 5): 10

=== df.head(10) ===
      Date              Product  Price  Cost  Quantity
0  10/1/25     Pandesal (piece)      3     1        14
1  10/1/25        Spanish Bread     12     5        15
2  10/1/25            Ensaymada     25    10        16
3  10/1/25          Pan de Coco     15     6        17
4  10/1/25          Cheese Roll     18     8        18
5  10/1/25  Ube Cheese Pandesal     20     8        19
6  10/1/25                Monay     10     4        20
7  10/1/25   Tasty Loaf (slice)      8     3        21
8  10/1/25      Cheese Pandesal     12     5        22
9  10/1/25       Ham and Cheese     22     9        23
```

2. **df.tail():** "Here are the last rows (10/6–10/7/25); later-week volumes look higher for some items."



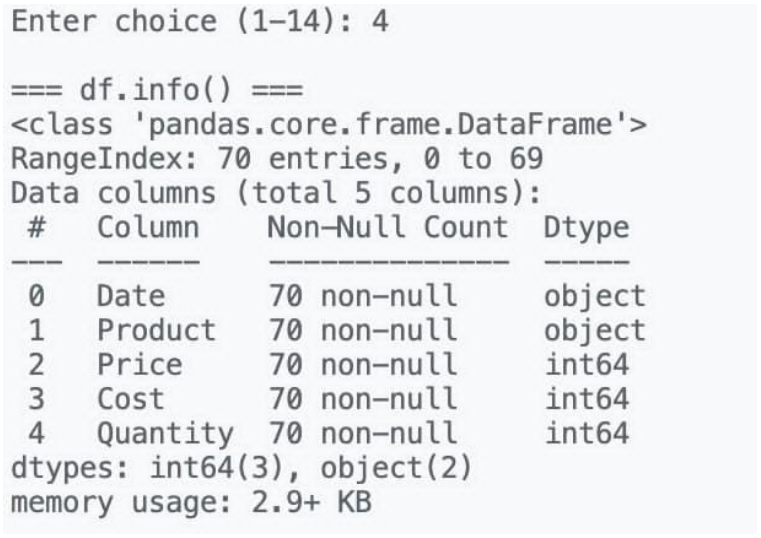
```
Enter choice (1-14): 2
How many rows? (default 5): 15

=== df.tail(15) ===
       Date              Product  Price  Cost  Quantity
55  10/6/25  Ube Cheese Pandesal     20     8        54
56  10/6/25                Monay     10     4        60
57  10/6/25   Tasty Loaf (slice)      8     3        66
58  10/6/25      Cheese Pandesal     12     5        72
59  10/6/25       Ham and Cheese     22     9        13
60  10/7/25     Pandesal (piece)      3     1        26
61  10/7/25        Spanish Bread     12     5        33
62  10/7/25            Ensaymada     25    10        40
63  10/7/25          Pan de Coco     15     6        47
64  10/7/25          Cheese Roll     18     8        54
65  10/7/25  Ube Cheese Pandesal     20     8        61
66  10/7/25                Monay     10     4        68
67  10/7/25   Tasty Loaf (slice)      8     3        75
68  10/7/25      Cheese Pandesal     12     5        17
69  10/7/25       Ham and Cheese     22     9        24
```

3. **df.describe():** “Typical price is mid-teens; quantities range ~13–75.”



```
Enter choice (1-14): 3

=== df.describe() ===
            Price       Cost   Quantity
count  70.000000  70.000000  70.000000
mean   14.500000   5.900000  35.214286
std     6.500279   2.719495  16.134303
min     3.000000   1.000000  13.000000
25%    10.000000   4.000000  22.000000
50%    13.500000   5.500000  32.000000
75%    20.000000   8.000000  46.500000
max    25.000000  10.000000  75.000000
```

4. **df.info():** “70 non-null rows; expected dtypes.”

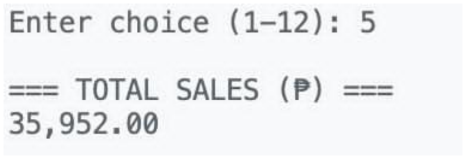


```
Enter choice (1-14): 4

=== df.info() ===
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70 entries, 0 to 69
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Date      70 non-null     object
 1   Product   70 non-null     object
 2   Price     70 non-null     int64
 3   Cost      70 non-null     int64
 4   Quantity  70 non-null     int64
dtypes: int64(3), object(2)
memory usage: 2.9+ KB
```

Week KPIs (Menu 5–7)

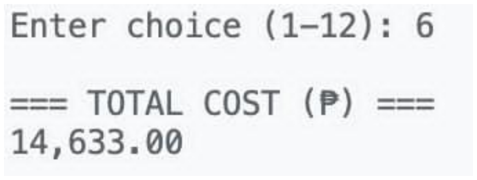
Before each total the code calls ensure_totals() to create **Total Sales**, **Total Cost**, **Profit** if needed.

5. TOTAL SALES: say the number and define it (Price×Quantity), e.g., ₱35,952.



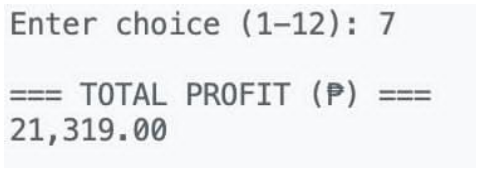
```
Enter choice (1–12): 5

=== TOTAL SALES (₱) ===
35,952.00
```

6. TOTAL COST: say the number, e.g., ₱14,633.



```
Enter choice (1–12): 6

=== TOTAL COST (₱) ===
14,633.00
```

7. TOTAL PROFIT: say the number, e.g., ₱21,319 (Sales − Cost).

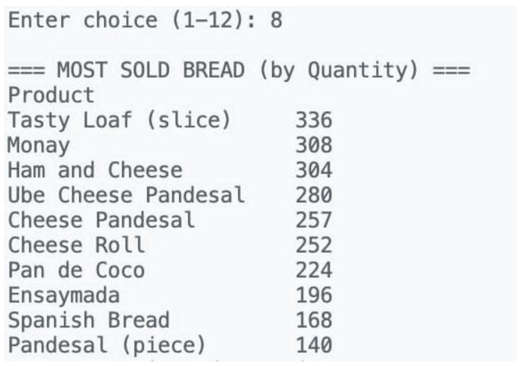


```
Enter choice (1–12): 7

=== TOTAL PROFIT (₱) ===
21,319.00
```

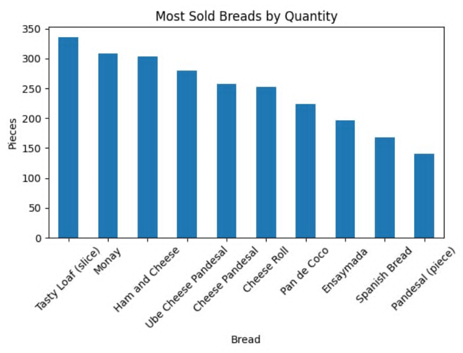
Product Rankings + Charts (Menu 8–11)

Comparison of **demand** (Quantity) vs **margin** (Profit).

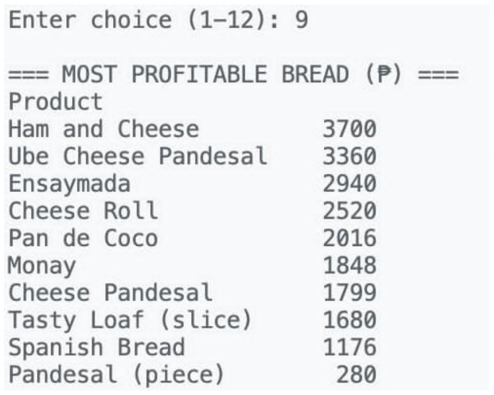
- **8 – Most Sold (Quantity)**: "Most Sold" ranks products by pieces sold (descending): Tasty Loaf (slice) leads with 336, followed by Monay (308) and Ham and Cheese (304), with Pandesal (piece) lowest at 140. This highlights demand leaders for production planning and slower items for attention. The accompanying bar chart makes the drop-off across items obvious at a glance.



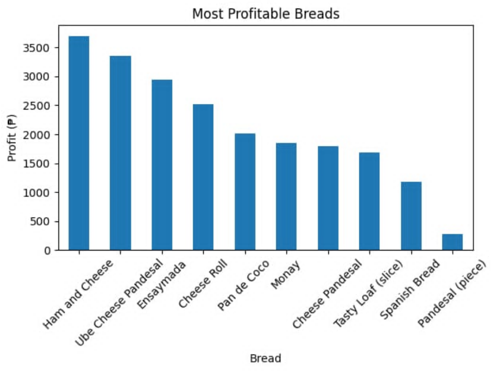
```
Enter choice (1–12): 8

=== MOST SOLD BREAD (by Quantity) ===
Product
Tasty Loaf (slice)     336
Monay                  308
Ham and Cheese         304
Ube Cheese Pandesal    280
Cheese Pandesal        257
Cheese Roll            252
Pan de Coco            224
Ensaymada              196
Spanish Bread          168
Pandesal (piece)       140
```



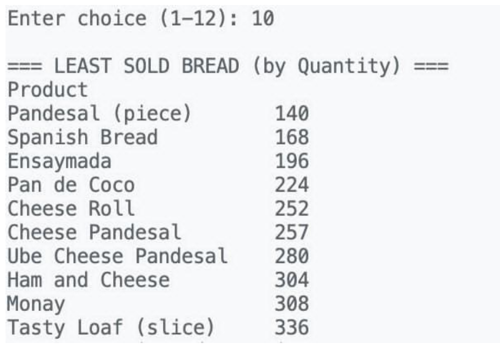
- **9 – Most Profitable (Profit)**: “Most Profitable” sorts by total Profit (descending): Ham and Cheese tops at ₱3,700, then Ube Cheese Pandesal (₱3,360) and Ensaymada (₱2,940), while Pandesal (piece) trails at ₱280. It shows that volume ≠ profit (e.g., Tasty Loaf is high volume but mid-profit). The bar chart clearly contrasts margin leaders vs. laggards.



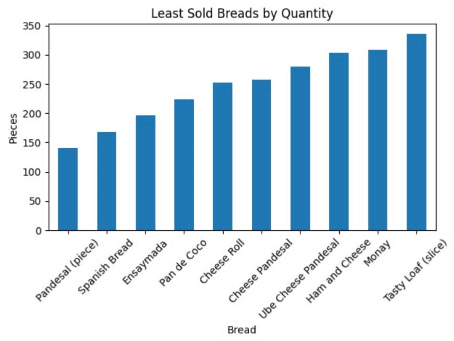
```
Enter choice (1–12): 9

=== MOST PROFITABLE BREAD (₱) ===
Product
Ham and Cheese          3700
Ube Cheese Pandesal     3360
Ensaymada               2940
Cheese Roll             2520
Pan de Coco             2016
Monay                   1848
Cheese Pandesal         1799
Tasty Loaf (slice)      1680
Spanish Bread           1176
Pandesal (piece)         280
```



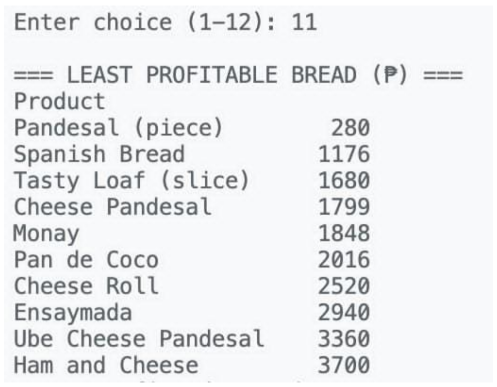
- **10 – Least Sold (Quantity):** “Least Sold” flips to ascending quantity, spotlighting slow movers: Pandesal (piece) 140 and Spanish Bread 168 are lowest, with demand stepping up through Ensaymada and Pan de Coco to Tasty Loaf (slice) 336. These are targets for pricing, bundling, or promotion—or intentional low-stock items to keep niche. The bar chart emphasizes which items sit far below the rest.



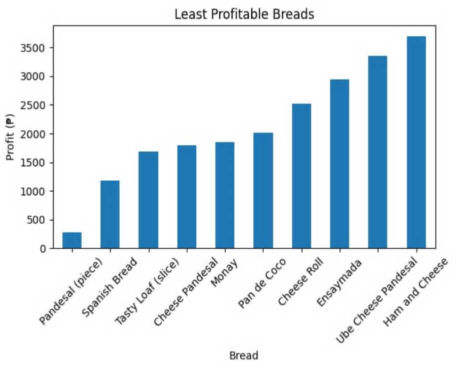
```
Enter choice (1–12): 10

=== LEAST SOLD BREAD (by Quantity) ===
Product
Pandesal (piece)       140
Spanish Bread          168
Ensaymada              196
Pan de Coco            224
Cheese Roll            252
Cheese Pandesal        257
Ube Cheese Pandesal    280
Ham and Cheese         304
Monay                  308
Tasty Loaf (slice)     336
```



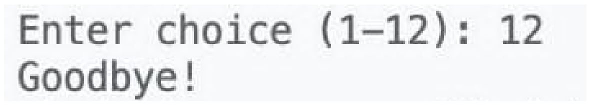
- **11 – Least Profitable (Profit):** “Least Profitable” orders products by ascending profit: Pandesal (piece ₱280) and Spanish Bread (₱1,176) are weakest, rising through Tasty Loaf and Cheese Pandesal to top earners like Ube Cheese Pandesal and Ham and Cheese. This view surfaces margin issues for cost, portion, or pricing adjustments. The bar chart makes the profit gap visually clear for quick decisions.



```
Enter choice (1-12): 11

=== LEAST PROFITABLE BREAD (₱) ===
Product
Pandesal (piece)        280
Spanish Bread          1176
Tasty Loaf (slice)     1680
Cheese Pandesal        1799
Monay                  1848
Pan de Coco            2016
Cheese Roll            2520
Ensaymada              2940
Ube Cheese Pandesal    3360
Ham and Cheese         3700
```



- **Exit (Menu 12):** show the clean termination ("Goodbye!").



```
Enter choice (1-12): 12
Goodbye!
```

SUMMARY OF CONTENT

This paper demonstrated a lightweight Python system that ingests a one-week bakery CSV, engineers' business features (Total Sales, Total Cost, Profit), and exposes a 12-option CLI for EDA, weekly KPIs, and product rankings with charts. Initial checks (head, tail, describe, info) confirmed a clean, 70-row dataset with sensible ranges, enabling reliable downstream summaries and visuals. Across the week, the shop recorded ₱35,952 in Sales, ₱14,633 in Cost, and ₱21,319 in Profit (Menus 5–7). Product rankings revealed that demand leaders (Menu 8)—notably Tasty Loaf (slice), Monay, and Ham and Cheese—are not identical to margin leaders (Menu 9), where Ham and Cheese, Ube Cheese Pandesal, and Ensaymada dominate.

The Least Sold and Least Profitable views (Menus 10–11) consistently surfaced Pandesal (piece) and Spanish Bread as laggards, suggesting targeted actions (price/portion tweaks, promotions, bundling, or reduced production). The paired bar charts (Menus 8–11) made these contrasts immediately visible—volume versus profit—supporting clear decisions in production planning and pricing. Overall, the workflow satisfies the assignment's Title→End structure, keeps originality (self-created dataset and narration), and delivers screenshot-ready outputs suitable for a concise, ~15-minute presentation.

CREDITS

Industry collaborator:

- Gil & Lily Bakery, 646G+CQG, St Christopher 2 Rd, Calamba, 4027 Laguna, Philippines.

References:

- Hayes, A. (2025, June 10). Gross profit: What it is and how to calculate it. *Investopedia*. https://www.investopedia.com/terms/g/grossprofit.asp
- Library of Congress. (2024, May 9). CSV, comma separated values (RFC 4180). *Sustainability of Digital Formats*. https://www.loc.gov/preservation/digital/formats/fdd/fdd000323.shtml
- Matplotlib Development Team. (2025). matplotlib.pyplot.bar. *Matplotlib 3.10.6 Documentation*. Retrieved October 5, 2025, from https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.bar.html

ACKNOWLEDGEMENT

We gratefully acknowledge **Gil & Lily Bakery** (646G+CQG, St Christopher 2 Rd, Calamba, 4027 Laguna, Philippines) for collaborating on operational context and dataset framing for this analysis and for helping us validate the relevance of our key performance indicators (Sales, Cost, Profit) and the usefulness of product-level rankings for decision-making in production and merchandising.