Statistics and Data Analysis
Unit 01 – Lecture 04 Notes
In-class Activity: Cleaning + Summary Tables + Plots

Tofik Ali

February 9, 2026

Purpose of This Activity

In real projects, you rarely receive a perfectly clean dataset. Before you can compute statistics or build models, you must first:

- understand what each column means,
- detect problems (missing values, invalid entries, outliers),
- apply clear cleaning rules,
- and then produce summary tables and plots that answer simple questions.

Repository. `https://github.com/tali7c/Statistics-and-Data-Analysis`

Learning Outcomes (What You Should Be Able To Do)

After completing this activity, you should be able to:

1. load a CSV in Python and inspect shape, head, and data types,
2. compute missingness and spot invalid values,
3. write and justify basic cleaning rules,
4. engineer simple features from existing columns,
5. compute group summaries using `groupby`,
6. generate and interpret three basic plots.

1. Dataset Description

File: `data/campus_cafe_transactions.csv`

1.1 What one row represents

Each row is one transaction at a campus cafe/store.

1.2 Column meanings

- `txn_id`: transaction id (identifier).
- `date`: transaction date (should become a datetime).
- `category`: product category (Snacks/Drinks/Stationery).
- `payment_mode`: Cash/UPI/Card.
- `units`: number of items purchased (should be non-negative).
- `unit_price`: price per unit in INR (should be positive).
- `discount_pct`: discount percentage (0 means no discount).

Important. This dataset intentionally contains:

- missing values (blank cells),
- invalid values (e.g., negative units),
- and outliers (unrealistic price/discount).

Your job is to clean it in a transparent, reproducible way.

2. Step-by-Step Instructions

2.1 Task A: Load and inspect (first 5 minutes)

Run these steps in a Python script or notebook:

```
import pandas as pd

df = pd.read_csv("data/campus_cafe_transactions.csv")
print(df.shape)
print(df.head())
print(df.dtypes)
```

What to check:

- Are numbers stored as numbers (or as strings)?
- Are there blanks in numeric columns?
- Does `date` look like a date?

2.2 Task B: Missingness (quick diagnostic)

Compute missing percentage per column:

```
missing_pct = df.isna().mean().sort_values(ascending=False) * 100
print(missing_pct)
```

Why we do this. If a column is mostly missing, dropping it may be reasonable. If only a few values are missing, imputation is often better than dropping rows.

2.3 Task C: Cleaning rules (the most important part)

Cleaning is not "one correct answer". The key is to apply **defensible rules**. For this activity, use rules like the following:

Rule 1: Convert types.

- parse `date` as datetime,
- convert `units`, `unit_price`, `discount_pct` to numeric,
- trim whitespace in categorical columns.

Rule 2: Handle missing discounts. If `discount_pct` is missing, fill it with 0 because in most sales systems, a missing discount usually means "no discount recorded".

Rule 3: Handle missing units (imputation). If only a few `units` values are missing, impute them with the median units. Median is used because it is robust (a few large purchases do not strongly affect the median).

Rule 4: Handle negative units (invalid). If `units` is negative, drop the row. Negative units would mean "selling -1 items", which is physically impossible for a purchase transaction.

Rule 5: Cap discount outliers. Discounts like 150% do not make sense for normal retail transactions. Cap `discount_pct` to a reasonable range, e.g., 0–50. This prevents absurd discounts from creating negative or near-zero net amounts.

Rule 6: Handle price outliers. If `unit_price` is unreasonably high (e.g., greater than 200 INR for this small campus dataset), replace it with the median price of its `category`. The reasoning: within a category, typical prices are similar, so category median is a stable fallback.

2.4 Task D: Feature engineering

Create three new columns:

- `gross_amount` = `units` × `unit_price`
- `net_amount` = `gross_amount` $\times(1 -$ `discount_pct`$/100)$
- `is_weekend` = 1 if Saturday/Sunday else 0

Why this matters. Raw columns are often not directly useful. The engineered `net_amount` is the key value that we will summarize and plot.

2.5 Task E: Summary tables

Compute:

- total net revenue by `category`
- total net revenue by `payment_mode`
- top 5 transactions by `net_amount`
- daily net revenue (sum of net amounts per day)

Typical patterns:

```
rev_by_cat = df.groupby("category")["net_amount"].sum()
rev_by_pay = df.groupby("payment_mode")["net_amount"].sum()
top5 = df.sort_values("net_amount", ascending=False).head(5)
daily = df.groupby(df["date"].dt.date)["net_amount"].sum()
```

2.6 Task F: Plots (and what to look for)

1. **Bar chart (revenue by category):** which category dominates?
2. **Histogram (net amount):** do we have many small transactions and a few big ones?
3. **Line chart (daily revenue):** is revenue stable day-to-day, or spiky?

3. Expected Results (After Applying the Example Cleaning Rules)

If you run the provided solution script (see next section), you should get results close to these:

3.1 Net revenue by category

Category	Count	Net Revenue (INR)
Snacks	9	1129.25
Drinks	8	368.00
Stationery	7	283.90

Interpretation. Snacks dominate revenue in this sample because there are several snack transactions and also one very large snack purchase. Even after cleaning, a few large transactions can strongly influence totals.

3.2 Net revenue by payment mode

Mode	Count	Net Revenue (INR)
UPI	12	1124.15
Cash	6	304.00
Card	6	353.00

Interpretation. UPI has the highest count and highest total revenue, which suggests it is the preferred payment mode in this small dataset.

3.3 Top transactions (by net amount)

The largest transaction is:

- `txn_id=5009` with `net_amount=365.0`.

This row originally had an unrealistic `unit_price=500`. After cleaning, the unit price is replaced with the category median (36.5 for Snacks), producing a sensible net amount.

4. Provided Solution Script (Mini Demo)

After you attempt the activity yourself, run:

```
python demo/activity_solution.py
```

It will generate:

- `data/campus_cafe_clean.csv`
- `data/revenue_by_category.csv`
- `data/revenue_by_payment.csv`
- `data/daily_revenue.csv`
- `data/top_transactions.csv`
- plots in `images/`: `revenue_by_category.png`, `net_amount_hist.png`, `daily_revenue.png`

5. Common Mistakes (And How To Avoid Them)

- **Forgetting type conversion:** always run `pd.to_numeric` and `pd.to_datetime`.
- **Dropping too many rows:** if missingness is small, prefer imputation.
- **Blindly deleting outliers:** first decide if it is an error or a true extreme.
- **Not writing rules:** if rules are not documented, results cannot be trusted or reproduced.
- **Not saving outputs:** always save cleaned data and plots.

6. Extension Questions (Optional)

If you finish early, try any two:

1. Compare weekend vs weekday revenue using `is_weekend`.
2. Compute average discount by category.
3. Count transactions per day and compare with daily revenue.
4. Identify whether "Snacks" dominate because of quantity or price.

References

- McKinney, W. *Python for Data Analysis.* O'Reilly, 2022.
- Montgomery, D. C., & Runger, G. C. *Applied Statistics and Probability for Engineers.* Wiley, 7th ed., 2020.