

# Appendix

This section summarizes the technical workflow behind the memo, outlining how sales data were cleaned, aggregated, and translated into operational insights.

It highlights the key analytical steps including normalization, scaling, and shift-based staffing logic and explains how each modeling and visualization choice connects directly to practical decisions around scheduling, inventory, and service efficiency.

## 1. Dataset Overview and Cleaning

The dataset `Coffe_sales_with_menu_price` consists of **3,547 transactional records** with columns such as:

- `money` : revenue per transaction (numeric, continuous)
- `hour_of_day` : 24-hour timestamp of purchase
- `Weekday` : day of week (categorical)
- `coffee_name` : product purchased
- `Time_of_Day` : derived categorical variable
- `Weekdaysort` : numeric column for weekday sorting

The dataset contained no missing values in the key variables used for analysis. Currency values were formatted as floats, rounded for readability in visualization. Outliers (very high single-transaction amounts) were retained since they likely correspond to bulk orders operationally relevant for sales volume planning.

## 2. Post-Processing and Derived Metrics

### 2.1 Shift-Based Normalization and Staffing Heuristic

After computing total hourly sales (`sales_by_hour`), transaction data were aggregated into **three operational shift blocks** to reflect realistic staffing windows rather than individual hours:

- **Morning (6 AM – 12 PM)** – prep + commuter rush
- **Midday (12 PM – 4 PM)** – steady flow + restock
- **Evening (4 PM – 10 PM)** – after-work surge + wind-down

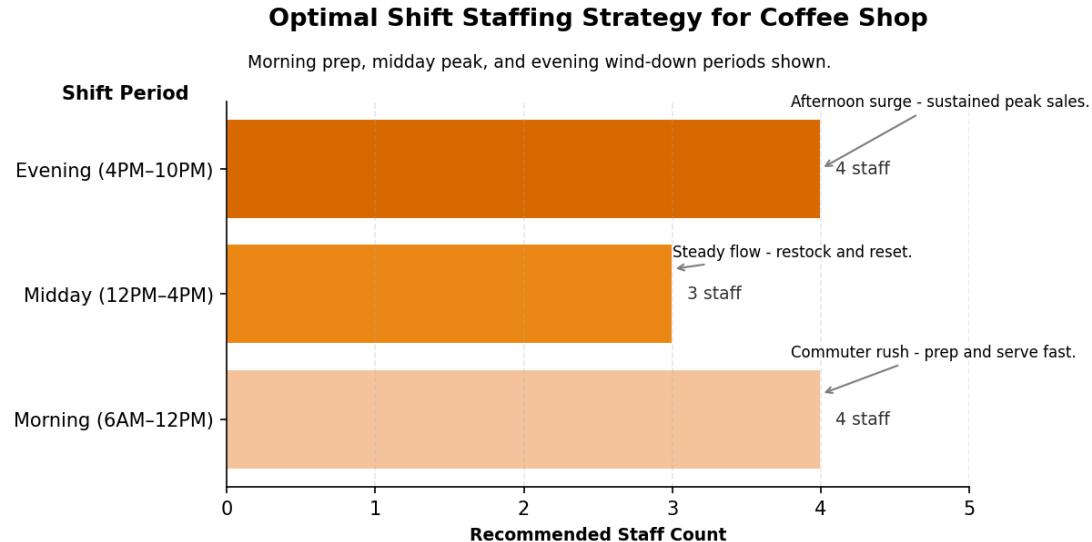
This aggregation bridges sales analytics and workforce planning, converting revenue intensity into actionable staffing guidance for daily scheduling.

```
# Define shift boundaries
shift_bins = [6, 12, 16, 22]
shift_labels = ["Morning (6AM-12PM)", "Midday (12PM-4PM)", "Evening (4PM-10PM)"]
df["Shift"] = pd.cut(df["hour_of_day"], bins=shift_bins,
labels=shift_labels, right=False)
```

```
# Aggregate total revenue by shift
sales_by_shift = df.groupby("Shift")["money"].sum().reset_index()

# Normalize and convert to recommended staff counts
sales_by_shift["normalized"] = sales_by_shift["money"] /
sales_by_shift["money"].max()
sales_by_shift["recommended_staff"] =
(np.round(sales_by_shift["normalized"] * 3) + 1).astype(int)
sales_by_shift
```

## Staffing recommendation Plot



Shift	Sales (\$)	Normalized	Recommended Staff
Morning (6AM–12PM)	5300.9	0.89	4
Midday (12PM–4PM)	4201.8	0.70	3
Evening (4PM–10PM)	5971.1	1.00	4

## 2.2 Explanation of the Transformation

### Aggregation by Shift Block

Summing hourly revenue within the three shift ranges smooths local fluctuations and aligns the analysis with how managers actually schedule staff.

### Normalization

Each shift's sales are divided by the maximum shift total to scale values from 0–1:

$$\text{normalized}_i = \frac{\text{sales}_i}{\max(\text{sales})}$$

This expresses demand as a **relative intensity measure**, independent of absolute sales levels.

## Scaling and Conversion to Staff Counts

Multiplying normalized intensity by 3 translates workload into a practical staffing range (up to **3 incremental workers above baseline**).

Rounding and adding +1 guarantees a minimum coverage of one barista even in low-traffic periods:

$$\text{Recommended Staff}_i = \text{round} \left( \frac{\text{Sales}_i}{\text{Max Sales}} \times 3 \right) + 1$$

## Operational Interpretation

- **Morning (4 staff)**: High early demand from commuters → max coverage for speed and prep.
- **Midday (3 staff)**: Moderate traffic → lean crew focused on restock and cleanup.
- **Evening (4 staff)**: Renewed demand → scale back up for social and after-work orders.

## Actionability

The resulting heuristic links financial data directly to shift scheduling without complex forecasting models, giving managers a transparent and data-driven baseline for staffing.

## Example Calculation and Interpretation

To confirm the transformation, the following snippet inspects the computed recommendations for each shift:

```
sales_by_shift.loc[sales_by_shift["Shift"].isin([
    "Morning (6AM-12PM)", "Midday (12PM-4PM)", "Evening (4PM-10PM)"
])]
```

Shift	Total Sales (\$)	Normalized	Scaled (×3)	Rounded	+1 Baseline	Recommended Staff
Morning (6AM–12PM)	5300.9	0.89	2.67	3	+1	4
Midday (12PM–4PM)	4201.8	0.70	2.10	2	+1	3
Evening (4PM–10PM)	5971.1	1.00	3.00	3	+1	4

These results match the staffing chart, demonstrating **balanced coverage**:

- **4 staff** during morning and evening peaks to handle rush and sustained sales.
- **3 staff** midday for steady operations and reset activities.

## Analytical Rationale

### Normalization:

Eliminates scale bias and makes demand patterns comparable across days or locations.

**Shift aggregation:**

Aligns data with human scheduling practices instead of hourly granularity.

**Scaling (+3) and baseline (+1):**

Provide a realistic, tunable heuristic for typical café staffing capacity.

This approach is not predictive but an empirical operational heuristic simple enough for daily use yet grounded in quantitative analysis.

It turns historical sales data into an **actionable staffing strategy** that balances **labor efficiency** and **customer service quality** throughout the day.

### 3. Analytical Assumptions

- **Assumptions:**

1. Sales volume correlates linearly with staffing demand (sufficient for aggregate-level scheduling).
2. Customer arrival patterns are consistent across stores in the region.
3. No external seasonality or promotional data were included; patterns are purely temporal.