WEEK 6

Exploratory Data Analysis I

Understanding data before modeling

Day 26	**Data Distributions**	Histograms, KDE, skewness, modality, central tendency
Day 27	**Histograms & Boxplots**	Bin size tuning, boxplot anatomy, outliers, group comparisons
Day 28	**Grouping & Aggregation**	groupby mechanics, multi-level, Simpson's paradox
Day 29	**Correlations**	Pearson coefficient, heatmaps, causation, multicollinearity
Day 30	**EDA Exercise**	Full workflow integration, insights, modeling recommendations

EDA WORKFLOW PATTERN

```
df.describe()  # Summary stats → df["col"].hist()  # Distribution shape
sns.boxplot(x="group", y="value", data=df)  # Group comparisons
df.groupby("key").agg(["mean", "std"])  # Segment analysis
```

DAY 26

Data Distributions

Univariate Analysis: Histograms, KDE, Skewness, Modality

OBJECTIVES

- Understand how values spread for a single variable
- Construct and read histograms and KDE plots
- Distinguish normal vs skewed vs multimodal
- Relate mean, median, mode to distribution shape
- Connect distribution shape to ML preprocessing

ACTIVITY

Plot distributions of several numeric features.

Compare histograms and KDE side by side.

Identify skewness and modality from plots.

ASSESSMENT

Short-answer questions on plot interpretation.

Peer review of plots and written interpretations.

DAY 26 · DISTRIBUTIONS

Univariate Analysis — Concept and Purpose

Examining one variable at a time

Why Before Modeling

- Reveals skewness, outliers, heavy tails
- Highlights measurement errors (negative ages)
- Suggests transformations (log, sqrt)
- Informs scaling and normalization choices

What It Measures

- Frequency of values across bins (histograms)
- Smoothed probability density (KDE)
- Central tendency and spread
- Distribution shape (symmetric, skewed, multimodal)

SUMMARY STATISTICS

```
summary_stats = df[["income", "age", "transactions"]].describe()
print(summary_stats)  # count, mean, std, min, 25%, 50%, 75%, max
# Large std relative to mean → high variability; check for outliers
```

DAY 26 · DISTRIBUTIONS

Histograms — Construction and Interpretation

Count observations falling into value intervals (bins)

What Histograms Show

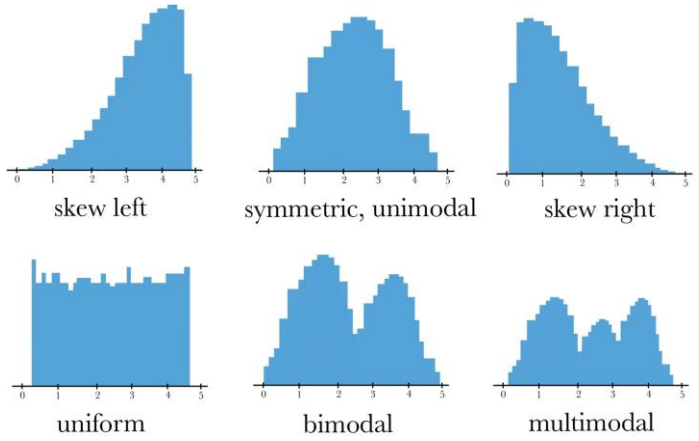
- Empirical approximation of underlying distribution
- Quick visual check for skewness, modality, outliers
- Helps detect data entry errors (spikes at 0 or max)

Reading Histograms

- Tall bars at low values + long right tail → right-skewed
- Gaps or isolated bars → potential outliers
- Single peak → unimodal; multiple → subpopulations

BASIC HISTOGRAM WITH PANDAS

```
plt.figure(figsize=(6, 4))
df["income"].hist(bins=30, edgecolor="black")
plt.title("Income Distribution — Histogram")
plt.xlabel("Income"); plt.ylabel("Count")
plt.tight_layout(); plt.show()
```



DAY 26 · DISTRIBUTIONS

Histogram Bin Size Effects

Too few hides structure; too many creates noise

Bin Size Impact

- Few bins → hides multimodality
- Many bins → noisy, hard to see shape
- Misleading preprocessing decisions if wrong

Guidelines

- Coarse (10): overall range, rough symmetry
- Medium (30): good default, balance noise/visibility
- Fine (60): reveals sampling noise, may be unstable

COMPARING BIN SIZES

```
fig, axes = plt.subplots(1, 3, figsize=(15, 4), sharey=True)
df["age"].hist(bins=10, ax=axes[0]); axes[0].set_title("10 bins")
df["age"].hist(bins=30, ax=axes[1]); axes[1].set_title("30 bins")
df["age"].hist(bins=60, ax=axes[2]); axes[2].set_title("60 bins")
```

DAY 26 · DISTRIBUTIONS

Density Plots (KDE) — Concept and Usage

Smooth estimate of probability density function

What KDE Shows

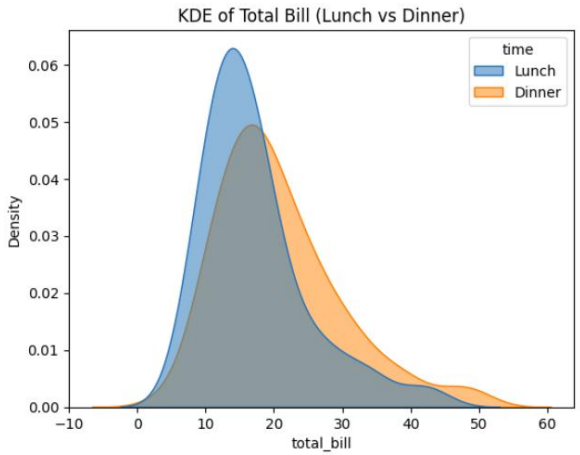
- Highlights shape without binning artifacts
- Uses kernel functions and bandwidth to smooth
- Easier to compare multiple distributions

Reading KDE

- Peak location ≈ mode of distribution
- Long tail → skew; values far from peak less probable
- Small bumps → potential subgroups or mixtures

KDE PLOT WITH SEABORN

```
plt.figure(figsize=(6, 4))
sns.kdeplot(df["income"], fill=True)
plt.title("Income Distribution — KDE")
plt.xlabel("Income"); plt.ylabel("Density")
plt.tight_layout(); plt.show()
```



DAY 26 · DISTRIBUTIONS

Comparing Histograms and KDE

Combined view for robust shape understanding

Why Compare Both

- Histogram shows actual counts
- KDE shows smoothed density
- Differences highlight bin vs bandwidth sensitivity

Discrete Data Caution

- KDE may smooth away discrete nature
- Peaks between integer values are artifacts
- For counts: consider pure histogram or different treatment

HISTOGRAM + KDE OVERLAY

```
plt.figure(figsize=(6, 4))
df["transactions"].hist(bins=10, alpha=0.5, density=True, label="Histogram")
sns.kdeplot(df["transactions"], color="red", label="KDE")
plt.legend(); plt.title("Transactions — Histogram + KDE")
```

DAY 26 · DISTRIBUTIONS

Normal vs Skewed Distributions

Identifying distribution type for preprocessing

Normal Characteristics

- Symmetric bell shape around mean
- Mean ≈ median ≈ mode
- Tails decay relatively quickly
- Good candidate for z-score standardization

Skewed Characteristics

- Right skew: long right tail, mean > median
- Left skew: long left tail, mean < median
- May benefit from log or Box-Cox transforms
- Robust scaling (median, IQR) often better

VISUAL COMPARISON

```
sns.kdeplot(df["age"], label="Age (approx normal)")
sns.kdeplot(df["income"], label="Income (right-skewed)")
plt.legend(); plt.title("Normal vs Skewed Example")
```

DAY 27

Histograms & Boxplots

Detailed Tuning and Spread/Outlier Visualization

OBJECTIVES

- Tune histograms to reveal meaningful structure
- Read boxplots accurately (quartiles, whiskers)
- Use boxplots to compare distributions across categories
- Understand boxplot limitations and misuses
- Connect spread/outliers to ML model behavior

ACTIVITY

Create boxplots comparing numeric variables across categories.

Explore bin size effects on histogram interpretation.

ASSESSMENT

Submit notebook with histograms, boxplots, and interpretation bullets.

DAY 27 · HISTOGRAMS

Bin Size, Scale, and Range Decisions

Affect perception of skewness and outliers

Key Decisions

- Number of bins (resolution of detail)
- Range displayed (clip extreme tails?)
- Density vs raw counts

Interpretation Impact

- Can hide or exaggerate differences
- Fewer bins: coarse overview
- Density normalization: compare different sample sizes

COUNTS VS DENSITY

```
fig, axes = plt.subplots(1, 2, figsize=(10, 4))
df2["score"].hist(bins=10, ax=axes[0]); axes[0].set_title("10 Bins (Counts)")
df2["score"].hist(bins=40, ax=axes[1], density=True); axes[1].set_title("40 Bins (Density)")
```

DAY 27 · BOXPLOTS

Boxplot Anatomy — Components and Definitions

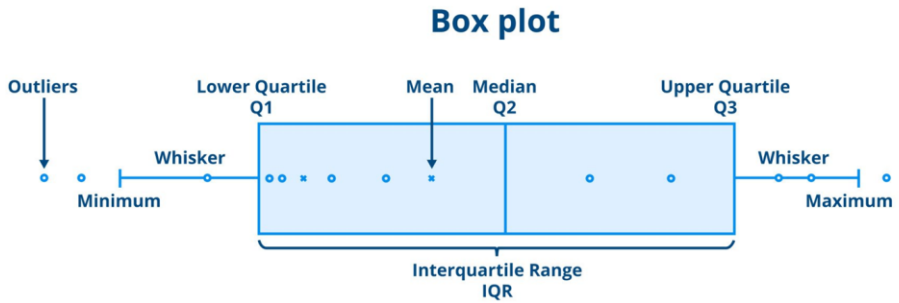
Compact summary of distribution

Components

- Median: central line in box
- Q1, Q3: bottom and top of box
- IQR = Q3 − Q1
- Whiskers: data within 1.5 × IQR
- Points beyond: potential outliers

What It Shows

- Central tendency and spread
- Compare many groups in single figure
- Box height = spread of middle 50%
- Median position inside box shows skewness



BASIC BOXPLOT

```
plt.figure(figsize=(6, 4))
plt.boxplot(df2["score"], vert=True, showfliers=True)
plt.title("Score — Boxplot"); plt.ylabel("Score")
```

Implementation with pandas and seaborn

Quick inspection and flexible aesthetics

pandas boxplot

- df.boxplot(column='score') for quick inspection
- Works for single or multiple numeric columns

seaborn boxplot

- sns.boxplot(x='group', y='score') for grouped
- Flexible aesthetics and customization
- Highlights group differences clearly

PANDAS VS SEABORN BOXPLOT

```
df2.boxplot(column="score"); plt.title("pandas boxplot")
plt.show()
sns.boxplot(x="group", y="score", data=df2); plt.title("seaborn grouped")
```

DAY 27 · BOXPLOTS

Visualizing Outliers with Boxplots

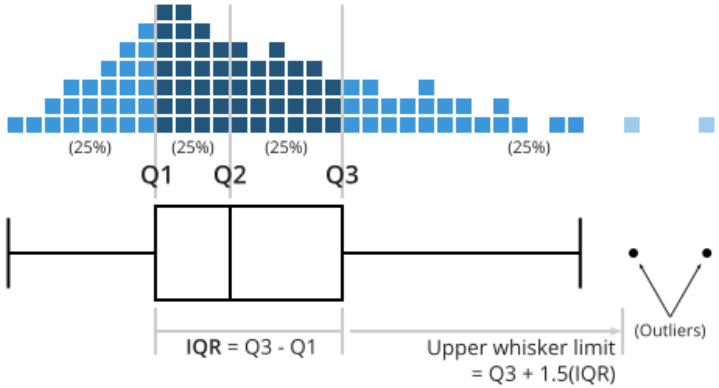
Identify points far from bulk of data

Role in Outlier Detection

- Points beyond whiskers flagged as potential outliers
- Based on IQR rule (convention, not absolute)
- Visual cue for anomalies or errors

Limitations

- High-variance distributions may show many 'outliers'
- Outliers depend on convention (1.5 × IQR)
- Check data quality before automatic removal



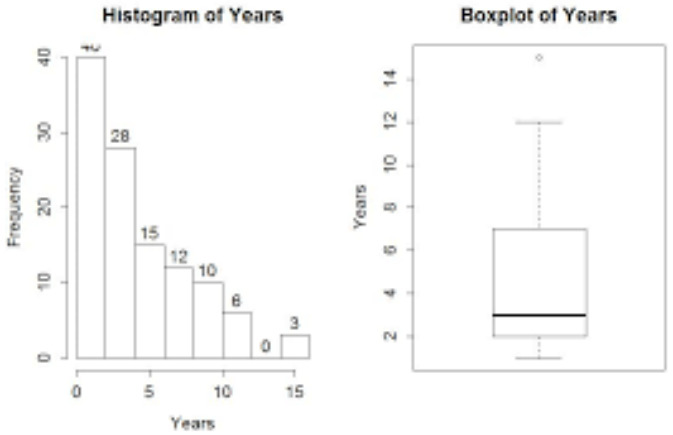
BOXPLOT WITH EXTREME VALUES

```
scores = np.concatenate([np.random.normal(70, 10, 300), np.array([20, 30, 120, 130])])
sns.boxplot(y=scores); plt.title("Scores with Outliers")
# Points far from whiskers are flagged
```

DAY 27 · BOXPLOTS

Boxplots vs Histograms — When to Use Which

Different tools for different purposes



Histograms

- Detailed shape and modality
- Frequency distribution with bin control
- Best for understanding single distribution shape

Boxplots

- Summary using quartiles and outliers
- Best for comparing many groups side by side
- Use histogram for transforms; boxplot for comparisons

SIDE-BY-SIDE COMPARISON

```
fig, axes = plt.subplots(1, 2, figsize=(10, 4))
df2["score"].hist(bins=20, ax=axes[0]); axes[0].set_title("Histogram")
sns.boxplot(y="score", data=df2, ax=axes[1]); axes[1].set_title("Boxplot")
```

DAY 27 · BOXPLOTS

Limitations and Common Misuses

Boxplots can hide important structure

Limitations

- Hide multimodality; multiple peaks not visible
- Sensitive to skew and heavy tails
- Outlier rule is arbitrary (not inherently 'wrong' points)

Common Misuses

- Declaring outliers as errors solely from boxplot
- Comparing vastly different sample sizes
- Always pair with histograms/KDE when modality matters

BIMODAL HIDDEN BY BOXPLOT

```
# Boxplot suggests single distribution; histogram reveals bimodality
sns.boxplot(y=bimodal_data); plt.title("Boxplot hides bimodality")
```

DAY 27 · BOXPLOTS

ML Impact — Outliers and Spread on Models

How extremes affect different model families

Linear Models

Coefficients heavily influenced by extremes

Outliers can dominate loss function

Tree-Based

More robust; splits may isolate outliers

Still affected by spread

Distance-Based

Outliers distort distance metrics

High variance features dominate

OUTLIER IMPACT ON LINEAR REGRESSION

```
y_with_outliers = y.copy(); y_with_outliers[[5, 20]] += 10  # Add extremes
model_clean = LinearRegression().fit(X, y)
model_outliers = LinearRegression().fit(X, y_with_outliers)
print("Slope clean:", model_clean.coef_[0], "Slope with outliers:", model_outliers.coef_[0])
```

DAY 28

Grouping & Aggregation

Using groupby() for EDA

OBJECTIVES

- Use groupby().agg() for summary statistics
- Interpret grouped summaries and variability
- Detect potential data leakage and aggregation bias
- Use grouped EDA to inform feature engineering
- Understand multi-level grouping and Simpson's paradox

ACTIVITY

Compute and analyze summary statistics by group.

Visualize aggregated metrics with bar plots.

ASSESSMENT

Code-based challenge using multiple groupby() operations.

DAY 28 · AGGREGATION

Purpose of Aggregation in EDA

Condense rows into summary statistics per group

What Aggregation Does

- Condenses many rows into summary per group
- Computes mean, median, count, sum, min, max
- Reveals differences between segments

Why Before Modeling

- Identifies groups with systematically different outcomes
- Highlights where separate models may be needed
- Informs group-level feature engineering

BASIC GROUPBY AGGREGATION

```
region_summary = df3.groupby("region")["sales"].agg(["count", "mean", "median", "std"])
print(region_summary)
# count: sample size; high std: high variability
```

region	count	mean	median	std
North	1200	55.3	52.0	10.1
South	200	60.1	58.0	25.4
West	50	58.0	57.0	5.2

DAY 28 · AGGREGATION

groupby() Mechanics — Single Key

Groups by unique values in column

Basic Usage

- df.groupby('col') creates groups
- Aggregation functions applied per group
- .agg(['mean', 'median', 'std']) for numeric

Interpretation

- Compare average profit across segments
- High discount with low profit → bad strategy
- Different std across groups → varying variability

SEGMENT SUMMARY

```
segment_summary = df3.groupby("segment")[["sales", "discount", "profit"]].agg(
    ["mean", "median", "std"]
)
print(segment_summary)
```

	sales			discount			profit		
	mean	median	std	mean	median	std	mean	median	std
segment									
Consumer	250.4	180.0	310.2	0.16	0.15	0.09	28.5	22.0	45.7
Corporate	310.8	230.0	420.5	0.13	0.10	0.08	35.2	30.0	60.1
Home Office	190.6	140.0	260.7	0.18	0.20	0.10	18.9	15.0	35.4

DAY 28 · AGGREGATION

groupby().agg() — Multiple Aggregations

Powerful named aggregations in one call

Why Powerful

- Multiple metrics per column in one call
- Named aggregations for clarity
- Accepts custom functions

Reading Results

- Spot regions with high sales but low profit
- profit_min and profit_max show outcome range
- Combine metrics for richer understanding

NAMED AGGREGATIONS

```
agg_summary = df3.groupby("region").agg(
    sales_mean=("sales", "mean"), sales_median=("sales", "median"),
    discount_mean=("discount", "mean"), profit_min=("profit", "min"), profit_max=("profit", "max")
)
print(agg_summary)
```

region	sales_mean	sales_median	discount_mean	profit_min	profit_max
East	245.3	180.0	0.15	-820.0	4500.0
West	265.8	190.0	0.12	-1200.0	5200.0
Central	210.4	160.0	0.18	-1500.0	3100.0
South	230.1	170.0	0.16	-600.0	2800.0

DAY 28 · AGGREGATION

Multi-level Grouping — region × segment

Capture interaction between categorical features

What It Captures

- Interaction between multiple categorical features
- Differences within segments across regions
- Expose where one segment behaves differently

Interpretation Tips

- Compare across region-segment pairs
- Very low count → interpret cautiously
- May reveal need for interaction features

MULTI-LEVEL GROUPBY

```
multi_group = df3.groupby(["region", "segment"])[["sales", "discount", "profit"]].agg(
    ["mean", "median", "std", "count"]
)
print(multi_group.head(10))
```

		sales				discount				profit			
		mean	median	std	count	mean	median	std	count	mean	median	std	count
region	segment												
East	Consumer	250.4	180.0	310.2	520	0.16	0.15	0.09	520	28.5	22.0	45.7	520
East	Corporate	310.8	230.0	420.5	310	0.13	0.10	0.08	310	35.2	30.0	60.1	310
East	Home Office	190.6	140.0	260.7	180	0.18	0.20	0.10	180	18.9	15.0	35.4	180
West	Consumer	270.1	195.0	330.8	480	0.12	0.10	0.07	480	32.0	26.0	50.3	480

DAY 28 · AGGREGATION

ML Impact — Using Aggregation for Features

Create group-level features from EDA insights

Feature Engineering Uses

- Create group-level features (e.g., region mean)
- Use segment statistics as baselines
- Detect groups needing separate models

Leakage Warning

- Group features must use only training data
- Calculate on train, apply to all
- Careful with time-based aggregations

GROUP-LEVEL FEATURE

```
region_mean = df3.groupby("region")["sales"].mean().rename("region_sales_mean")
df3_with_feature = df3.join(region_mean, on="region")
print(df3_with_feature[["region", "sales", "region_sales_mean"]].head())
```

DAY 28 · AGGREGATION

groupby() and Missing Data

Handle NAs in group keys explicitly

Issues

- Rows with NA in group key excluded from groups
- Aggregation functions may ignore NA by default
- Missing rows disappear from output

Solutions

- Add 'Unknown' category for missing group keys
- Check for excluded rows after groupby
- Explicit handling before aggregation

HANDLING MISSING GROUP KEYS

```
df_missing = df3.copy(); df_missing.loc[::50, "region"] = np.nan
summary = df_missing.groupby("region")["sales"].agg(["count", "mean"])
print(summary)  # NA region rows are excluded
```

DAY 29

Correlations

Linear Relationships, Heatmaps, and Multicollinearity

OBJECTIVES

- Compute and interpret correlation matrices
- Use scatter plots to inspect relationships
- Understand correlation vs causation
- Identify multicollinearity in predictors
- Connect correlations to feature selection

ACTIVITY

Compute and interpret correlations in numeric dataset.

Create correlation heatmap and scatter plots.

ASSESSMENT

Interpretation-focused quiz on correlation results.

DAY 29 · CORRELATIONS

Pearson Correlation — Definition and Computation

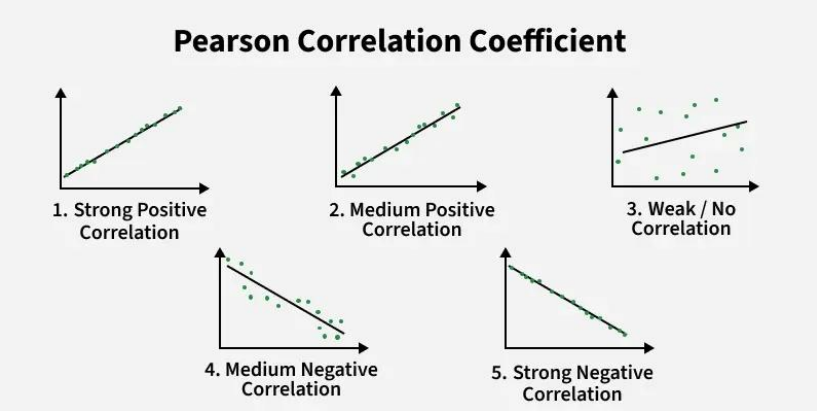
Measures linear relationship between continuous variables

Definition

- Range: −1 (perfect negative) to +1 (perfect positive)
- 0 indicates no linear correlation (not independence)
- Measures strength and direction of linear relationship

Why It Matters

- Identifies redundant features (high pairwise corr)
- Detects linear relationships with target
- Guides feature selection decisions



CORRELATION MATRIX

```
corr_matrix = df4[["feature_x", "feature_y", "feature_z", "feature_w", "target"]].corr()
print(corr_matrix)
# Values near +1 or -1: strong linear relationships
```

DAY 29 · CORRELATIONS

Scatter Plots — Visualizing Pairwise Relationships

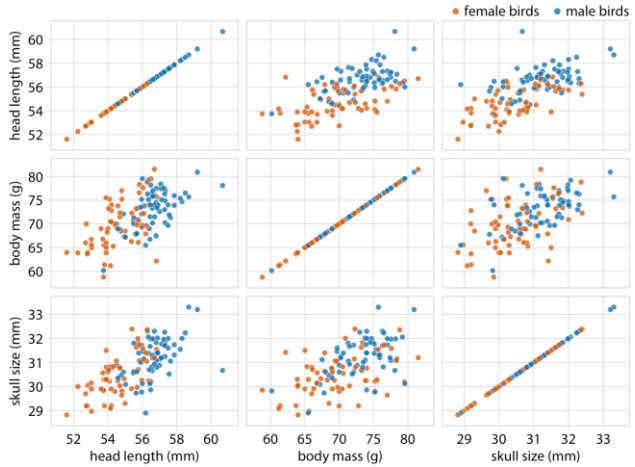
Point-by-point relationship visualization

What Scatter Plots Show

- Relationship between two numeric variables
- Patterns: linear, non-linear, clusters, outliers
- Reveal heteroscedasticity and outliers

Complement to Correlation

- Non-linear relationships may have low Pearson
- Tight upward band → strong positive linear
- Always visualize, don't rely on numbers alone



SCATTER PLOT

```
plt.figure(figsize=(6, 4))
plt.scatter(df4["feature_x"], df4["feature_y"], alpha=0.5)
plt.title("feature_x vs feature_y"); plt.xlabel("feature_x"); plt.ylabel("feature_y")
```

DAY 29 · CORRELATIONS

Correlation Heatmaps — Visual Overview

Color-coded correlation matrix

Purpose

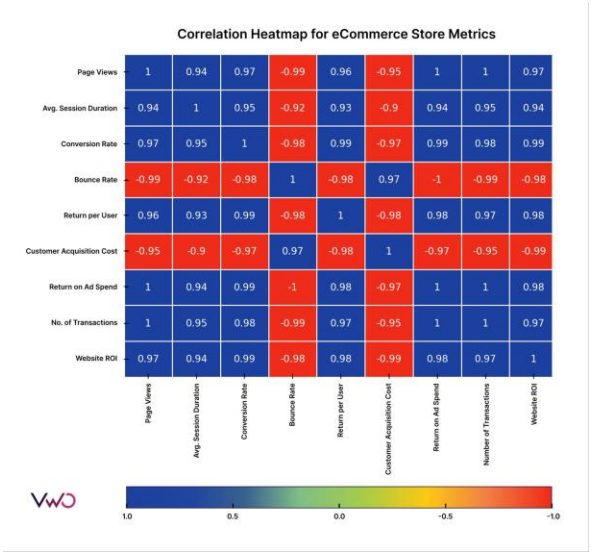
- Visualize entire correlation matrix at once
- Quickly spot strong positive/negative correlations
- Identify clusters of related features

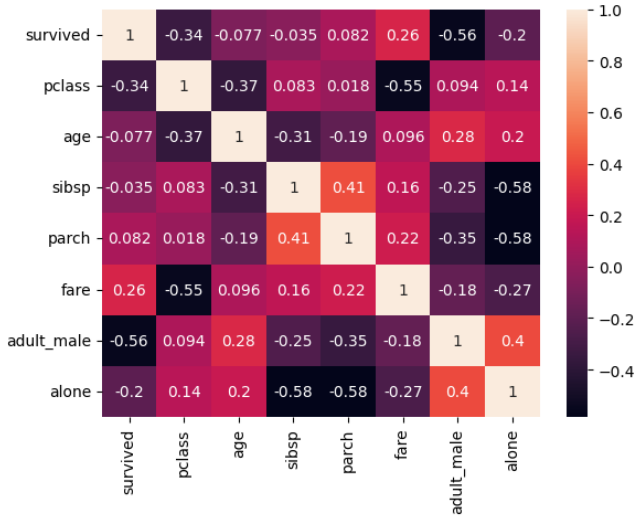
Reading Heatmaps

- Warm (red) → positive; Cool (blue) → negative
- Darker = stronger magnitude
- High magnitude pairs → redundancy candidates

CORRELATION HEATMAP

```
plt.figure(figsize=(6, 5))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", vmin=-1, vmax=1)
plt.title("Correlation Heatmap")
```





DAY 30

EDA Exercise

Putting It All Together

OBJECTIVES

- Structure a complete EDA workflow
- Choose appropriate plots and tables
- Extract actionable insights
- Avoid over-interpretation
- Guide feature engineering and modeling decisions

ACTIVITY

Perform full EDA on realistic dataset.

Extract 3+ concrete insights with supporting plots.

ASSESSMENT

Submit EDA findings with modeling recommendations.

DAY 30 · EDA WORKFLOW

Structuring an EDA Workflow — Stepwise Plan

Systematic approach to data understanding

Typical EDA Steps

1. Understand shape, types, missingness
2. Univariate analysis (histograms, KDE)
3. Boxplots (overall and group-wise)
4. Groupby aggregation for segments
5. Correlation analysis for numeric features
6. Summarize insights and next steps

Initial Inspection

- Check dataset shape and dtypes
- Identify heavy missingness early
- Decide numeric vs categorical candidates

INITIAL DATA INSPECTION

```
print("Shape:", df_eda.shape, "\nDtypes:\n", df_eda.dtypes)
missing = df_eda.isna().sum().sort_values(ascending=False)
print("Missing values:\n", missing)
```

DAY 30 · EDA WORKFLOW

Choosing Plots for Numeric Features

Systematic visualization for each variable

Plot Types

- Histograms with .hist() for each numeric
- KDE plots with sns.kdeplot() for continuous
- Boxplots for outlier visualization

What to Note

- For each: skewness, modality, outliers
- Flag variables needing transformation
- Document findings as you go

LOOP THROUGH NUMERIC FEATURES

```
for col in df_eda.select_dtypes(include=["number"]).columns:
    plt.subplot(1, 2, 1); df_eda[col].hist(bins=30); plt.title(f"{col} — Histogram")
    plt.subplot(1, 2, 2); sns.kdeplot(df_eda[col].dropna()); plt.title(f"{col} — KDE")
    plt.tight_layout(); plt.show()
```

DAY 30 · EDA WORKFLOW

Boxplots for Group Comparisons in Exercise

Compare target across categories

Steps

- Identify key numeric outcome (target)
- Identify categorical variables
- Create group-wise boxplots

Interpretation Focus

- Categories with higher median target
- Categories with large spread or many outliers
- Consider interactions or segment-specific models

GROUP-WISE BOXPLOTS IN EXERCISE

```
for cat in cat_cols[:3]:  # First few categorical columns
    if df_eda[cat].nunique() <= 10:
        sns.boxplot(x=cat, y=target_col, data=df_eda)
        plt.title(f"{target_col} by {cat}"); plt.show()
```

DAY 30 · EDA WORKFLOW

Groupby Summaries in Exercise

Aggregate metrics by segments

Steps

- Choose grouping keys (region, category)
- Compute summary stats for key metrics
- Identify groups with high/low values

Focus Areas

- Groups with high target mean and sufficient count
- Groups with unusually high variability
- Potential group-level features

GROUPBY IN EXERCISE

```
group_key = cat_cols[0] if len(cat_cols) > 0 else None
if group_key:
    group_summary = df_eda.groupby(group_key)[numeric_cols].agg(["mean", "median", "std", "count"])
    print(group_summary.head())
```

DAY 30 · EDA WORKFLOW

Correlation Analysis in Exercise

Identify relationships and redundancies

Steps

- Select numeric features plus target
- Compute correlation matrix and heatmap
- Follow up with scatter plots for key pairs

What to Find

- Features strongly correlated with target
- Clusters of highly correlated predictors
- Multicollinearity risk areas

CORRELATION IN EXERCISE

```
corr_ex = df_eda[numeric_cols].corr()
sns.heatmap(corr_ex, annot=False, cmap="coolwarm", vmin=-1, vmax=1)
plt.title("EDA Exercise — Correlation Heatmap")
```

DAY 30 · EDA WORKFLOW

Extracting Insights — From Plots to Statements

Turn visualizations into actionable recommendations

Guidelines

- Refer explicitly to distribution shape
- Mention group differences with magnitudes
- Link patterns to hypotheses about behavior

Good Insights

- Each backed by at least one plot/table
- Should suggest actions (transform, create feature)
- Specific, not vague ('somewhat skewed')

EXAMPLE INSIGHTS

```
insights = ["Feature A is right-skewed; consider log transform before modeling.",
    "Category X has higher median target; may warrant separate modeling.",
    "Two predictors are strongly correlated; one may be redundant."]
for i, text in enumerate(insights, 1): print(f"Insight {i}: {text}")
```

DAY 30 · EDA WORKFLOW

Avoiding Over-Interpretation in EDA

Not every pattern is meaningful

Common Pitfalls

- Treating every pattern as meaningful
- Making causal claims from correlations
- Ignoring small sample sizes in subgroups

Best Practice

- Consider randomness and sample size
- Verify patterns with domain knowledge
- Statistical significance is a guide, not proof

SMALL-SAMPLE CORRELATION CAUTION

```
from scipy.stats import pearsonr
r, p_value = pearsonr(feature_a[:50], feature_b[:50])  # Small sample
print("r:", r, "p-value:", p_value)  # May fluctuate; interpret cautiously
```

DAY 30 · EDA WORKFLOW

From EDA to Modeling Decisions

Translate findings into preprocessing and model choices

How EDA Guides

- Identify transformations (log, scaling) needed
- Highlight important segments for interactions
- Suggest features to exclude (redundant, noisy)

Concrete Outputs

- List of features requiring transformation
- List of redundant features to drop
- Segment-specific modeling recommendations

RECOMMENDED NEXT STEPS

```
recommended = ["Apply log1p to heavily right-skewed variables.",
    "Use robust scaling for heavy-tailed features.",
    "Drop one of each highly correlated pair.",
    "Engineer group-level features where differences are strong."]
for step in recommended: print("-", step)
```

DAY 30 · ACTIVITY

Full EDA Workflow — 3+ Insights

Complete EDA exercise on real dataset

Deliverables

1. Choose real dataset relevant to your work
2. Univariate analysis for numeric features (Histogram)
3. Boxplots for target
4. groupby().agg() for at least one grouping key
5. Correlation heatmap for numeric features
6. Write concrete insights with supporting plots

Assessment

Q1: Which features require transformation and why?

Q2: Which groups show notably different behavior?

Q3: Which features are likely redundant?

Q4: Top 3 EDA-driven modeling recommendations?

Submit notebook with code, plots, and interpretations.

Week 6 Complete!

Exploratory Data Analysis I — Understanding Before Modeling

Distributions	Histograms, KDE, skewness, modality; guides transformation and scaling choices
Boxplots	Quartiles, outliers, group comparisons; pair with histograms for complete picture
Aggregation	groupby().agg() for segment analysis; watch for Simpson's paradox
Correlations	Linear relationships, heatmaps; correlation ≠ causation; multicollinearity
EDA Workflow	Systematic approach; concrete insights; modeling recommendations

Key Takeaway

EDA is not optional. Understanding your data through visualization and summary statistics prevents costly mistakes in feature engineering and modeling.

EDA CHECKLIST

```
df.describe() → histograms/KDE → boxplots → groupby().agg() → correlations → insights → recommendations
```