

## Exploratory Data Analysis (EDA) — Complete & Detailed Cheat sheet

“By Akshen Dhami”

---

### What is EDA?

**Exploratory Data Analysis (EDA)** is the process of exploring, summarizing, and visualizing data to understand its underlying structure, detect patterns, and find anomalies before applying modelling or prediction.

### When to Use:

👉 Always perform EDA before any modelling or machine learning — it forms your understanding of the dataset and reveals necessary preprocessing steps.

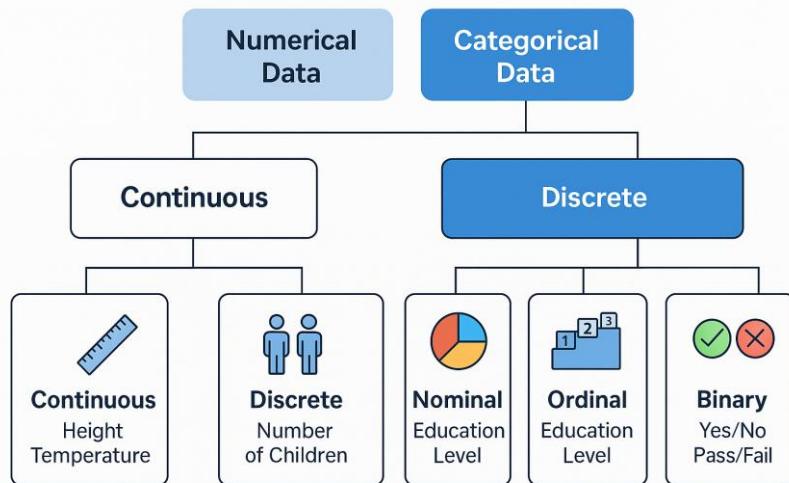
---

## Understanding Data

### Data

Units of information — structured (like CSV files or databases) or unstructured (like images, audio, or text).

## A) Types of Variables



### 1. Numerical Variables (Quantitative)

| Type       | Description                             | Example                             | Python Example     |
|------------|---|-------------------------------------|--------------------|
| Continuous | Infinite possible values within a range | Temperature, height, weight         | tips['total_bill'] |
| Discrete   | Countable finite values                 | Number of siblings, products bought | titanic['sibsp']   |

```

import seaborn as sns
tips = sns.load_dataset('tips')
titanic = sns.load_dataset('titanic')
print(tips['total_bill'].head()) # Continuous
print(titanic['sibsp'].head()) # Discrete

```

#### When to Use:

- Continuous → For statistical and numerical modeling (regression, distribution analysis).
- Discrete → When counting items, frequencies, or categorical occurrences numerically.

---

### 2. Categorical Variables (Qualitative)

| Type    | Description               | Example                | Python Code             |
|---------|---------------------------|------------------------|-------------------------|
| Nominal | No order among categories | Gender (Male/Female)   | titanic['sex'].head()   |
| Binary  | Exactly two categories    | Smoker: Yes/No         | tips['smoker'].head()   |
| Ordinal | Ordered categories        | Class: 1st > 2nd > 3rd | titanic['class'].head() |

#### When to Use:

- Nominal → When categories have **no ranking** (e.g., colors, city names).
  - Ordinal → When categories have **meaningful order** (e.g., education level).
  - Binary → When working with **two possible states** (True/False, 1/0).
-

## B) Data Types (Check & Change)

### Check Data Types

```
titanic.dtypes
```

### Change Data Type

```
titanic['age'] = titanic['age'].astype('float')
```

### When to Use:

- Always check data types before computation or visualization.
  - Convert incorrect data types (like strings in numeric columns) to appropriate formats to avoid calculation errors.
- 

## C) Accessing Data with loc and iloc

| Method            | Description                          | Example                          | Output                            |
|-------------------|--------------------------------------|----------------------------------|-----------------------------------|
| <b>loc</b>        | Access rows/columns by <b>labels</b> | titanic.loc[0, 'sex']            | Access "sex" of first passenger   |
| <b>iloc</b>       | Access by <b>index positions</b>     | titanic.iloc[0, 2]               | Access first row, third column    |
| <b>loc slice</b>  | Select multiple rows by label        | titanic.loc[0:5, ['sex', 'age']] | Returns multiple rows and columns |
| <b>iloc slice</b> | Select by range                      | titanic.iloc[0:5, 0:3]           | First 5 rows, 3 columns           |

### When to Use:

- loc → When you know the **column names** or **row labels**.
  - iloc → When you want to slice by **numerical index position**.
  - Always use iloc in loops or when column names are dynamic.
- 

## D) Descriptive Statistics

| Measure       | Description        | Example                 | Python Code                 |
|---------------|--------------------|-------------------------|-----------------------------|
| <b>Mean</b>   | Arithmetic average | Average restaurant bill | tips['total_bill'].mean()   |
| <b>Median</b> | Middle value       | Middle bill value       | tips['total_bill'].median() |
| <b>Mode</b>   | Most frequent      | Common tip              | tips['tip'].mode()[0]       |

```
tips[['total_bill', 'tip']].describe()
```

### When to Use:

- Mean → Data is **normally distributed** and without outliers.
  - Median → Data is **skewed** or has **outliers** (robust to extreme values).
  - Mode → Data is **categorical** or you want the most common occurrence.
-

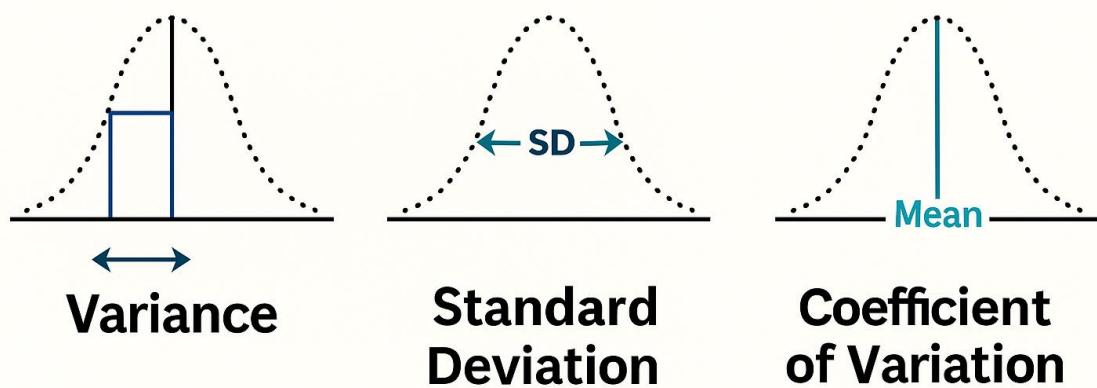
## E) Distribution of Data

```
sns.histplot(tips['total_bill'], kde=True)
```

### When to Use:

- Always visualize distribution before deciding on transformations or model types.
  - If highly skewed → apply **log** or **Yeo-Johnson** transformations.
- 

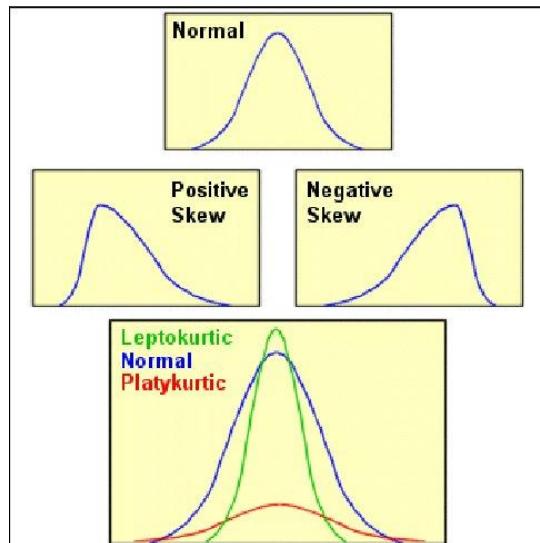
## F) Measures of Dispersion



| Measure                         | Description                   | Code  | When to Use   |
|---------------------------------|-------------------------------|---|---|
| <b>Variance</b>                 | Spread of data around mean    | <code>tips['total_bill'].var()</code>                           | To measure variability in continuous data                     |
| <b>Standard Deviation</b>       | Square root of variance       | <code>tips['total_bill'].std()</code>                           | To compare spread in similar scale                            |
| <b>Coefficient of Variation</b> | $\text{Std} \div \text{Mean}$ | <code>tips['total_bill'].std()/tips['total_bill'].mean()</code> | When comparing relative variability across different datasets |

---

## G) Skewness & Kurtosis

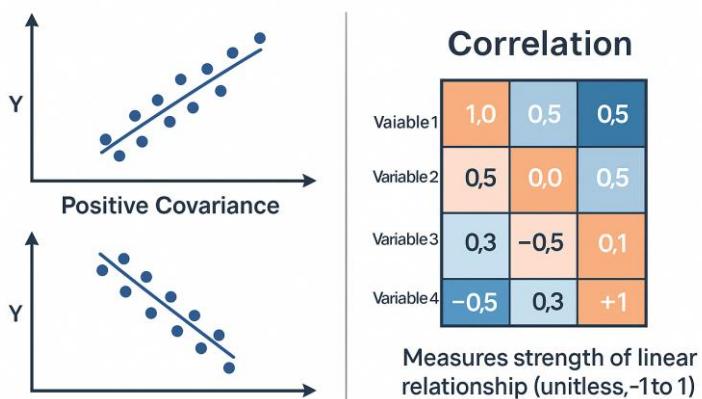


| Concept         | Meaning           | Types   | Example                | Code                                   |
|-----------------|-------------------|---|------------------------|--|
| <b>Skewness</b> | Asymmetry of data | Right (+ve), Left (-ve), Zero                                 | Income (right-skewed)  | <code>tips['total_bill'].skew()</code> |
| <b>Kurtosis</b> | Peakedness        | Leptokurtic (peaked), Platykurtic (flat), Mesokurtic (normal) | Exam scores clustering | <code>tips['total_bill'].kurt()</code> |

### When to Use:

- Use skewness & kurtosis to understand **shape of data distribution** before applying parametric statistical tests.
  - If skewness  $> \pm 1$  → data is **heavily skewed** → consider transformation.
- 

## H) Covariance & Correlation



Covariance: Measures how variables move together  
(units dependent)

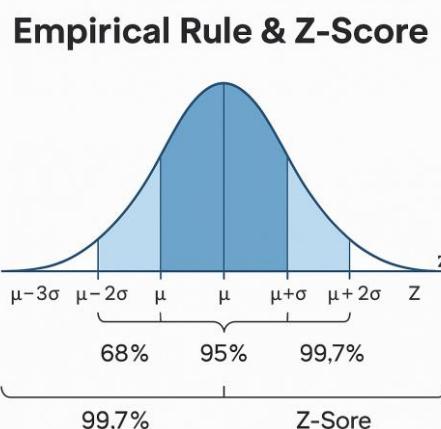
| Measure     | Description                         | Types                    | Example                                    | Code        |
|-------------|-------------------------------------|--------------------------|--|-------------|
| Covariance  | How two variables vary together     | +ve, -ve, 0              | Bill $\uparrow \rightarrow$ Tip $\uparrow$ | tips.cov()  |
| Correlation | Strength of relationship (-1 to +1) | Positive, Negative, None | Bill vs Tip                                | tips.corr() |

`sns.heatmap(tips.corr(), annot=True, cmap='coolwarm')`

#### When to Use:

- Correlation → for **linear relationships** between variables.
  - Covariance → for **magnitude and direction** of co-movement.
  - Use **correlation** more often — easier to interpret and scale-independent.
- 

## I) Empirical Rule (68–95–99.7 Rule) & Z-Score



#### Empirical Rule

- 68% of values within **1 std**
- 95% within **2 std**
- 99.7% within **3 std**

#### Z-Score

Represents how many std deviations a value is from mean.

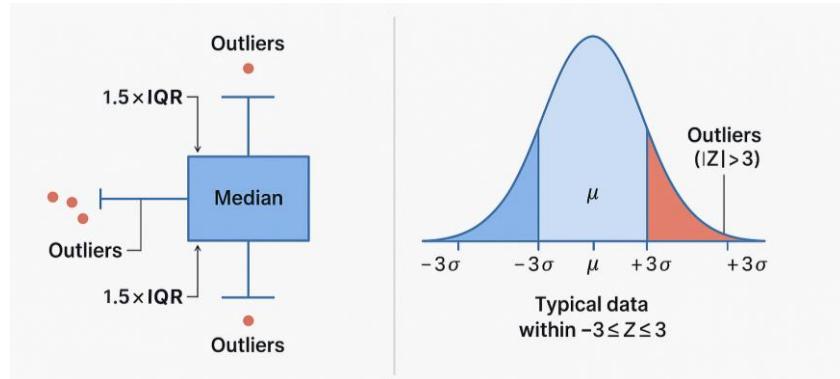
from `scipy import stats`

`tips['z_score'] = stats.zscore(tips['total_bill'])`

#### When to Use:

- Use Z-scores for **outlier detection** and **standardization**.
  - Great for normally distributed variables to find extremes.
-

## J) Outliers



**What:** Data points far from the central trend.

### 1 Z-Score Method

```
outliers_z = tips[(tips['z_score'] > 3) | (tips['z_score'] < -3)]
```

### 2 IQR Method

```
Q1, Q3 = tips['tip'].quantile([0.25, 0.75])
```

IQR = Q3 - Q1

```
outliers_iqr = tips[(tips['tip'] < (Q1 - 1.5 * IQR)) | (tips['tip'] > (Q3 + 1.5 * IQR))]
```

**When to Use:**

- Use **Z-score** for **normal data**;
- Use **IQR** for **non-normal or skewed data**.

## K) Missing Values

```
titanic.isnull().sum()
```

| Method               | Description                   | Code   | When to Use                              |
|----------------------|-------------------------------|--|--|
| <b>Imputation</b>    | Replace with mean/median/mode | titanic['age'].fillna(titanic['age'].median(), inplace=True) | If missingness is small and predictable  |
| <b>Deletion</b>      | Drop rows/columns             | titanic.dropna(subset=['embarked'], inplace=True)            | If missing data is small fraction (<5%)  |
| <b>Visualization</b> | Detect pattern                | sns.heatmap(titanic.isnull(), cbar=False)                    | Before imputing, check patterns visually |

## L) Encoding Categorical Data

| Type               | Description                     | Code  | When to Use                             |
|--------------------|---------------------------------|---|---|
| <b>Dummy (k-1)</b> | Creates k-1 binary columns      | pd.get_dummies(titanic['sex'], drop_first=True)   | Use when avoiding multicollinearity     |
| <b>One-Hot</b>     | Creates separate binary columns | pd.get_dummies(titanic['sex'])  | For non-ordinal, independent categories |
| <b>Label</b>       | Converts to integer codes       | from sklearn.preprocessing import LabelEncoder;<br>titanic['sex']=LabelEncoder().fit_transform(titanic['sex'])  | For ordinal categories or tree models   |
| <b>Frequency</b>   | Uses occurrence frequency       | freq =<br>titanic['embarked'].value_counts()/len(titanic);<br>titanic['embarked']=titanic['embarked'].map(freq) | When number of categories is large      |

---

## M) Feature Scaling

| Method                 | Description       | Code  | When to Use   |
|------------------------|-------------------|---|---|
| <b>Standardization</b> | Mean=0, Std=1     | from sklearn.preprocessing import StandardScaler; tips[['total_bill','tip']] = StandardScaler().fit_transform(tips[['total_bill','tip']]) | For algorithms like SVM, PCA, Logistic Regression     |
| <b>Min-Max</b>         | Scale between 0-1 | from sklearn.preprocessing import MinMaxScaler; tips[['total_bill','tip']] = MinMaxScaler().fit_transform(tips[['total_bill','tip']])     | For bounded feature importance (Neural Networks, KNN) |

---

## N) Data Transformation

| Transformation               | Purpose                      | Clarification           | Code   | When to Use  |
|------------------------------|------------------------------|-------------------------|--|--|
| <b>Log Transform</b>         | Reduce right skewness        | Makes data symmetric    | import numpy as np;<br>np.log(tips['total_bill'])                | For exponential distributions (e.g. income, sales) |
| <b>Exponential Transform</b> | Inverse of log               | Revert to original      | np.exp(tips['total_bill'])                                       | To reverse log scaling                             |
| <b>Box-Cox</b>               | Normalize positive-only data | Requires >0             | from scipy import stats;<br>stats.boxcox(tips['total_bill'])     | When data is strictly positive                     |
| <b>Yeo-Johnson</b>           | Works with negatives         | Handles zeros/negatives | from scipy import stats;<br>stats.yeojohnson(tips['total_bill']) | When data includes negative values                 |

## O) Types of EDA Analysis

| Type                | Focus        | Numerical Methods           | Categorical Methods    | When to Use  |
|---------------------|--------------|-----------------------------|------------------------|--|
| <b>Univariate</b>   | 1 variable   | Histogram, Boxplot, Summary | Count plot, Bar chart  | To understand distribution of individual variables |
| <b>Bivariate</b>    | 2 variables  | Scatterplot, Correlation    | Crosstab, Stacked bar  | To explore relationships                           |
| <b>Multivariate</b> | 3+ variables | Pairplot, Heatmap           | Grouped bar, Hue plots | To observe combined effects and interactions       |

## P) Train-Test Split

```
from sklearn.model_selection import train_test_split
X = tips[['total_bill', 'size']]
y = tips['tip']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

### When to Use:

- Before model training to evaluate generalization.
- 70–30 or 80–20 split is typical depending on data size.

## **Q) Summary—Everything Covered**

- Data types & variable types
- Data type checks & conversions
- Accessing data (iloc, loc)
- Descriptive stats
- Dispersion & shape measures
- Covariance & correlation
- Empirical Rule & Z-scores
- Outlier detection (Z & IQR)
- Missing value handling
- Encoding & scaling
- Data transformation
- EDA types (univariate/bivariate/multivariate)
- Train-test split with use cases