**🧭 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).

1. **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).

1. **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.

1. **Accessing Data with loc and iloc**

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

1. **Descriptive Statistics**

| **Measure** | **Description** | **Example** | **Python Code** |
| --- | --- | --- | --- |
| **Mean** | Arithmetic average | Average restaurant bill | tips['total\_bill'].mean() |
| **Median** | Middle value | Middle bill value | tips['total\_bill'].median() |
| **Mode** | 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.

1. **Distribution of Data**

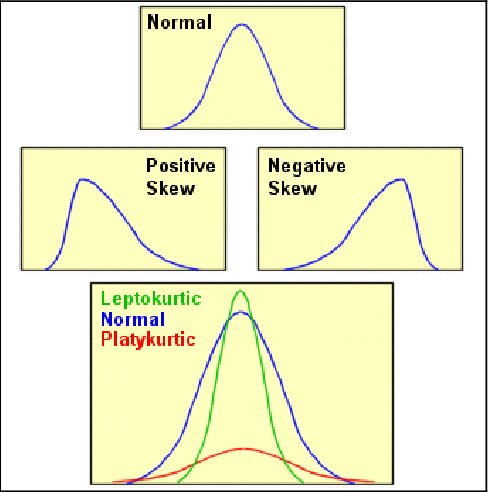
sns.histplot(tips['total\_bill'], kde=True)

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

1. A diagram of normal distribution

   AI-generated content may be incorrect.**Measures of Dispersion**

| **Measure** | **Description** | **Code** | **When to Use** |
| --- | --- | --- | --- |
| **Variance** | Spread of data around mean | tips['total\_bill'].var() | To measure variability in continuous data |
| **Standard Deviation** | Square root of variance | tips['total\_bill'].std() | To compare spread in similar scale |
| **Coefficient of Variation** | Std ÷ Mean | tips['total\_bill'].std()/tips['total\_bill'].mean() | When comparing relative variability across different datasets |

1. **Skewness & Kurtosis**

| **Concept** | **Meaning** | **Types** | **Example** | **Code** |
| --- | --- | --- | --- | --- |
| **Skewness** | Asymmetry of data | Right (+ve), Left (−ve), Zero | Income (right-skewed) | tips['total\_bill'].skew() |
| **Kurtosis** | Peakedness | Leptokurtic (peaked), Platykurtic (flat), Mesokurtic (normal) | Exam scores clustering | tips['total\_bill'].kurt() |

**When to Use:**  
✅ Use skewness & kurtosis to understand **shape of data distribution** before applying parametric statistical tests.  
✅ If skewness > ±1 → data is **heavily skewed** → consider transformation.

1. **Covariance & Correlation**

**A diagram of covarian and covarian relationships

AI-generated content may be incorrect.**

| **Measure** | **Description** | **Types** | **Example** | **Code** |
| --- | --- | --- | --- | --- |
| **Covariance** | How two variables vary together | +ve, −ve, 0 | Bill ↑ → Tip ↑ | 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.

1. **Empirical Rule (68–95–99.7 Rule) & Z-Score**

**A diagram of a normal distribution

AI-generated content may be incorrect.**

**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.

1. **Outliers**

**A diagram of a normal distribution

AI-generated content may be incorrect.**

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

**1️⃣ Z-Score Method**

outliers\_z = tips[(tips['z\_score'] > 3) | (tips['z\_score'] < -3)]

**2️⃣ 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**.

1. **Missing Values**

titanic.isnull().sum()

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

1. **Encoding Categorical Data**

| **Type** | **Description** | **Code** | **When to Use** |
| --- | --- | --- | --- |
| **Dummy (k−1)** | Creates k−1 binary columns | pd.get\_dummies(titanic['sex'], drop\_first=True) | Use when avoiding multicollinearity |
| **One-Hot** | Creates separate binary columns | pd.get\_dummies(titanic['sex']) | For non-ordinal, independent categories |
| **Label** | Converts to integer codes | from sklearn.preprocessing import LabelEncoder; titanic['sex']=LabelEncoder().fit\_transform(titanic['sex']) | For ordinal categories or tree models |
| **Frequency** | Uses occurrence frequency | freq = titanic['embarked'].value\_counts()/len(titanic); titanic['embarked']=titanic['embarked'].map(freq) | When number of categories is large |

1. **Feature Scaling**

| **Method** | **Description** | **Code** | **When to Use** |
| --- | --- | --- | --- |
| **Standardization** | 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 |
| **Min–Max** | 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) |

1. **Data Transformation**

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

1. **Types of EDA Analysis**

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

1. **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.

1. **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