1. Data Cleaning: Handling Missing Values, Outliers, Duplicates, Normalization, and Transformation

Objective: This notebook covers techniques for identifying and handling missing values, removing duplicates, detecting and treating outliers, and normalizing/transformation of data.

Sample Notebook: Data Cleaning in Python
Key Steps:

- **Missing Values**:
    - Checking for missing values using .isnull(), .sum().
    - Imputing missing values using mean, median, or mode.
    - Dropping rows or columns with excessive missing data.
- **Outliers**:
    - Using IQR (Interquartile Range) method or Z-score to detect outliers.
    - Visualizing with boxplots.
- **Duplicates**:
    - Identifying duplicates using .duplicated() and removing them with .drop_duplicates().
- **Normalization and Transformation**:
    - Using MinMaxScaler() or StandardScaler() from sklearn to normalize/standardize data.
    - Transforming non-numeric columns using encoding techniques like One-Hot Encoding.

Key Python Libraries: Pandas, NumPy, Seaborn, Matplotlib, Scikit-learn

---

2. Data Manipulation: Filtering, Grouping, Aggregating, and Feature Engineering

Objective: This notebook covers how to filter, group, and aggregate data, as well as techniques for creating new features to improve analysis.

Sample Notebook: Data Manipulation in Python
Key Steps:

- **Filtering**:
    - Selecting rows based on conditions (df[df['column'] > 50]).
    - Filtering based on multiple conditions using logical operators (&, |).
- **Grouping**:
    - Using .groupby() to group data by categorical variables and aggregate numerical data.
    - Aggregating using .mean(), .sum(), .count(), and custom aggregation functions.
- **Feature Engineering**:
    - Creating new features (e.g., extracting the year, month, day from date columns).
    - Binning continuous data into categories using pd.cut() or pd.qcut().
    - Generating dummy variables (one-hot encoding) for categorical features.

- Handling categorical variables using LabelEncoder or OneHotEncoder for machine learning models.

Key Python Libraries: Pandas, NumPy, Scikit-learn

---

3. Merging Datasets: Join, Merge, Concatenate and Handle Column Discrepancies

Objective: This notebook demonstrates various methods for merging datasets from multiple sources, including joining, merging, and concatenating data, as well as handling discrepancies in column names and data types.

Sample Notebook: Merging Datasets in Python
Key Steps:

- **Concatenation**:
  - Concatenating datasets using pd.concat() along rows or columns.
- **Merging**:
  - Merging datasets on a common column with pd.merge() (inner, outer, left, right joins).
  - Handling column name discrepancies during merging by using the left_on, right_on parameters.
  - Handling differences in data types before merging.
- **Joining**:
  - Using .join() to merge on index or columns in a more straightforward way than .merge().
- **Handling Discrepancies**:
  - Renaming columns using .rename().
  - Ensuring matching data types using .astype() or converting columns to appropriate types.

Key Python Libraries: Pandas

---

4. Deriving Insights: Statistical Methods, Visualizations, and Reporting

Objective: This notebook demonstrates how to apply statistical methods, data visualization, and how to create reports or dashboards to share findings.

Sample Notebook: Deriving Insights in Python
Key Steps:

- **Statistical Summary**:
  - Calculating basic statistics like mean, median, mode, variance, standard deviation.
  - Performing correlation analysis with .corr(), identifying relationships between variables.
- **Data Visualization**:
  - Visualizing distributions using histograms, box plots, and density plots (sns.histplot(), sns.boxplot()).
  - Scatter plots for visualizing relationships between two numerical features (sns.scatterplot()).
  - Heatmaps to visualize correlation matrices (sns.heatmap()).
  - Bar plots and count plots for categorical data (sns.barplot(), sns.countplot()).

- **Advanced Visualization**:
    - Pair plots for multivariate relationships (sns.pairplot()).
    - Violin plots for comparing distributions of a numeric variable across categories.
- **Reporting**:
    - Creating summary tables or text reports using Jupyter Notebook markdown cells.
    - Generating interactive dashboards using Plotly, Dash, or Jupyter widgets (for more advanced reporting).

Key Python Libraries: Pandas, NumPy, Seaborn, Matplotlib, Scikit-learn, Plotly

---