Chapter 4: Data Wrangling

Repeated Questions:

1. Data Wrangling and Missing Values:

- **Q.** Explain Data wrangling / Define Data wrangling and explain steps needed to perform data wrangling. **(SUMMER 2023, 3 Marks; WINTER 2024, 7 Marks OR)**

Answer:

- Data wrangling is the process of cleaning, transforming, and organizing raw data into a usable format for analysis.
- **Steps for 3 marks:** (1) Collect raw data, (2) Clean missing/duplicate/outlier values, (3) Transform/format data, (4) Validate and store.

Detailed steps for 7 marks:

- Data discovery/collection: Gather data from CSV, databases, APIs, web scraping.
- Data cleaning: Handle missing values, duplicates, outliers, inconsistent formats, and type errors.
- Data transformation: Normalize/scale, encode categorical variables, create derived features, parse dates.
- Data structuring: Reshape, merge/join, pivot, split columns, aggregate.
- Data enrichment: Add external data or computed fields.
- Validation: Check completeness, accuracy, consistency, and schema.
- Storage/export: Save cleaned data to CSV, database, or DataFrame for modeling.

- **Q.** Why is it important to find/handle missing values? Discuss different approaches to handle missing values with examples. **(SUMMER 2022, 4 Marks OR; SUMMER 2024, 7 Marks; WINTER 2024, 7 Marks OR; SUMMER 2025, 4 Marks OR)**

Answer:

- **Importance:** Missing values bias statistics, reduce model accuracy, break algorithms, and cause errors/exceptions.

Approaches (for 4 marks):

- **Remove rows/columns:** df.dropna() removes incomplete records.
- **Impute mean/median/mode:** df.fillna(df.mean()) replaces missing numeric values.
- **Forward/backward fill:** df.fillna(method='ffill') propagates previous value.
- **Predictive imputation:** Use regression/KNN to estimate missing values.

For 7 marks, add:

- **Constant fill:** df.fillna(0) or "Unknown" for categorical.
- **Indicator variable:** Add flag column marking missingness.
- **Interpolation:** df.interpolate() for time-series.
- **Domain-specific rules:** Replace with known defaults or business logic.
- **Evaluate impact:** Compare model performance before/after imputation.

---

2. Web Scraping and Parsing:

- **Q.** Explain HTML/XML parsing / Web Scraping using Beautiful Soup with example/program. **(SUMMER 2022, 7 Marks OR; WINTER 2022, 4 Marks; SUMMER 2023, 4 Marks; SUMMER 2025, 7 Marks OR)**

Answer:

- HTML/XML parsing converts markup text into a tree structure so tags, attributes, and text can be searched/extracted.
- Beautiful Soup is a Python library for parsing HTML/XML and navigating the parse tree.
- Steps: Request page → parse with BeautifulSoup → find elements (find, find_all) → extract text/attributes.

Example program:

```
import requests
from bs4 import BeautifulSoup

url = "https://example.com"
html = requests.get(url).text
soup = BeautifulSoup(html, "html.parser")

print(soup.title.text)
for link in soup.find_all("a"):
    print(link.get("href"), link.text)
```

- **Explanation:** requests.get() fetches HTML; BeautifulSoup(..., "html.parser") parses; find_all("a") collects all anchor tags; get("href") extracts link.
- **For 7 marks:** Mention parsers (html.parser, lxml, html5lib), navigation (parent, children, siblings), CSS selectors (select()), and handling malformed HTML.

---

3. Text Processing:

- **Q.** Explain stemming / stemming and stop words removal / Word Tokenization, Stemming and Stop word removal using nltk. **(WINTER 2022, 4 Marks; SUMMER 2023, 7 Marks; SUMMER 2024, 7 Marks OR; WINTER 2024, 4 Marks)**

Answer:

- **Tokenization:** Splitting text into words/sentences. nltk.word_tokenize(text).
- **Stemming:** Reducing words to root/stem form (e.g., “running” → “run”) using PorterStemmer.
- **Stop word removal:** Removing common words (“the”, “is”, “and”) that add little meaning.

Example code:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

text = "Running dogs are quickly jumping"
tokens = word_tokenize(text.lower())
stop_words = set(stopwords.words("english"))
filtered = [w for w in tokens if w not in stop_words]
stemmer = PorterStemmer()
stems = [stemmer.stem(w) for w in filtered]
print(stems)
```

- **Explanation:** Tokenize → lowercase → remove stop words → stem each remaining word.
- **For 7 marks:** Add difference between stemming vs lemmatization; mention SnowballStemmer, WordNetLemmatizer, and use in text classification/search.

- **Q.** Explain TF-IDF transformations / Define n-gram and explain TF-IDF techniques. **(SUMMER 2024, 4 Marks; WINTER 2024, 7 Marks; SUMMER 2025, 7 Marks OR)**

Answer:

- TF-IDF (Term Frequency–Inverse Document Frequency) weights words by importance in a document relative to corpus.
- TF = frequency of term in a document; IDF = log(N / df) where df = documents containing term; TF-IDF = TF × IDF.
- **n-gram:** Contiguous sequence of n tokens; unigram (1), bigram (2), trigram (3). Captures word order/phrases.

Example:

```
from sklearn.feature_extraction.text import TfidfVectorizer
docs = ["data science is fun", "python for data science"]
vec = TfidfVectorizer(ngram_range=(1,2))
X = vec.fit_transform(docs)
print(vec.get_feature_names_out())
print(X.toarray())
```

- **Explanation:** TfidfVectorizer converts text to TF-IDF matrix; ngram_range=(1,2) includes unigrams and bigrams.
- **For 7 marks:** Mention high TF-IDF = term frequent in doc but rare in corpus; used in search, classification, clustering; normalize vectors.

- **Q.** Explain Bag of Words model / What kind of data is analyzed with Bag of Words model? **(SUMMER 2022, 3 Marks; WINTER 2022, 4 Marks OR; WINTER 2024, 7 Marks; SUMMER 2025, 7 Marks)**

Answer:

- Bag of Words (BoW) represents text as a vector of word counts, ignoring grammar and word order.
- Steps: Tokenize → build vocabulary → count occurrences per document → form matrix.
- **Data analyzed:** Text documents such as emails, reviews, tweets, news articles, product descriptions.

Example:

```
from sklearn.feature_extraction.text import CountVectorizer
docs = ["I love Python", "Python is great"]
vec = CountVectorizer()
X = vec.fit_transform(docs)
print(vec.get_feature_names_out())
print(X.toarray())
```

- **For 7 marks:** Mention limitations (sparse, no semantics/order), variants (binary BoW, n-grams), and use in sentiment analysis, spam detection, topic classification.

- **Q.** Explain Hashing Trick / hashing trick and hash function in Scikit-learn with example. **(SUMMER 2022, 3 Marks OR; WINTER 2022, 7 Marks; SUMMER 2023, 3 Marks OR; WINTER 2023, 7 Marks; WINTER 2024, 7 Marks; SUMMER 2025, 7 Marks)**

Answer:

- Hashing Trick maps features (words) to fixed-size integer indices using a hash function, avoiding a vocabulary dictionary.
- **Advantages:** Low memory, fast, handles large/streaming text, no need to store vocabulary.
- **Disadvantage:** Hash collisions; features not interpretable.

Scikit-learn example:

```
from sklearn.feature_extraction.text import HashingVectorizer
docs = ["data science", "python data"]
vec = HashingVectorizer(n_features=8, alternate_sign=False)
X = vec.transform(docs)
print(X.toarray())
```

- **Explanation:** n_features sets hash space size; each token hashed to an index; counts accumulated.
- **For 7 marks:** Mention alternate_sign=True reduces collision bias; compare with CountVectorizer/TfidfVectorizer; use in large-scale text classification.

---

4. Scikit-learn and Machine Learning:

- **Q.** Explain any three functions/interfaces from Scikit-learn library. **(WINTER 2022, 3 Marks OR; SUMMER 2024, 3 Marks OR; WINTER 2024, 3 Marks OR)**

Answer:

- **fit(X, y):** Trains model on data X and labels y.
- **predict(X):** Returns predicted labels/values for new data.
- **transform(X):** Applies preprocessing/feature transformation (e.g., scaling, PCA).
- **Also:** fit_transform(X) combines fit and transform; score(X, y) returns accuracy/R^2.

- **Q.** Explain Regression and Classification problems / Regression class / Classification class of Scikit-learn with example. **(SUMMER 2022, 7 Marks; SUMMER 2022, 7 Marks OR; WINTER 2023, 3 Marks; WINTER 2023, 4 Marks; SUMMER 2024, 7 Marks; WINTER 2024, 4 Marks OR; WINTER 2024, 4 Marks)**

Answer:

- **Regression:** Predicts continuous numeric output; e.g., house price, temperature.
- **Example:** from sklearn.linear_model import LinearRegression; model = LinearRegression(); model.fit(X, y); model.predict(X_new)
- **Classification:** Predicts discrete class/category; e.g., spam/not spam, disease/no disease.
- **Example:** from sklearn.linear_model import LogisticRegression; model = LogisticRegression(); model.fit(X, y); model.predict(X_new)
- **For 7 marks:** Mention common algorithms: Regression — Linear, Ridge, Lasso, DecisionTreeRegressor; Classification — LogisticRegression, KNeighborsClassifier, RandomForestClassifier, SVC. Evaluation: regression uses MSE/RMSE/R^2; classification uses accuracy, precision, recall, F1, confusion matrix. Use train_test_split and cross_val_score.

- **Q.** Differentiate Supervised and Unsupervised learning. **(SUMMER 2022, 4 Marks; WINTER 2024, 3 Marks OR)**

Answer:

Feature	Supervised Learning	Unsupervised Learning
Data	Labeled data (X, y)	Unlabeled data (X only)
Goal	Predict output/class	Find patterns/structure
Examples	Regression, classification	Clustering, dimensionality reduction
Algorithms	Linear regression, SVM, Random Forest	K-Means, PCA, DBSCAN

5. Magic Functions:

- **Q.** What is the use of %timeit and %%timeit magic functions? **(SUMMER 2024, 3 Marks; SUMMER 2025, 3 Marks OR)**

Answer:

- %timeit measures execution time of a single line/statement; runs multiple loops and reports best average.
- %%timeit measures execution time of an entire cell in Jupyter.
- **Use:** Compare code performance, identify bottlenecks, benchmark algorithms.
- **Example:** %timeit sum(range(1000)); %%timeit at top of cell.

Other Important Questions:

1. Web Data Access:

- **Q.** Elaborate XPath in detail with relatable example. **(SUMMER 2023, 7 Marks)**

Answer:

- XPath (XML Path Language) is a query language to select nodes from HTML/XML documents using path expressions.
- **Syntax:** / root, // descendant, @ attribute, [] predicate, * wildcard.

Examples:

- //div[@class="product"] selects all div with class product.
- //a/@href selects all link URLs.
- //table/tr[2]/td[1] selects first cell of second row.
- **Relatable example:** On an e-commerce page, //span[@class="price"]/text() extracts all product prices.
- **Python use:** lxml.html or BeautifulSoup.select() with CSS/XPath-like selectors; Scrapy supports XPath natively.
- **Advantages:** Precise navigation, handles nested structures, supports conditions and functions (contains(), text()).
- **Limitations:** Brittle if page structure changes; not all parsers support full XPath.

- **Q.** Write a Python code to access data from web. **(WINTER 2022, 4 Marks)**

Answer:

```
import requests

url = "https://api.example.com/data"
response = requests.get(url)
print("Status:", response.status_code)
data = response.json()   # if JSON API
print(data)
```

- **Explanation:** requests.get() sends HTTP GET; status_code checks success (200); .text gives HTML/text; .json() parses JSON response.
- For HTML pages, use response.text and parse with BeautifulSoup.

- **Q.** How to obtain online graphics and multimedia? Explain with example. **(WINTER 2022, 4 Marks OR)**

Answer:

- Use requests to download binary content; save with open(..., "wb").

Example image download:

```
import requests
url = "https://example.com/image.jpg"
r = requests.get(url)
with open("image.jpg", "wb") as f:
    f.write(r.content)
```

- For video/audio, same binary download; or use APIs (YouTube API, etc.).
- In notebooks, display with IPython.display.Image, Audio, Video, or matplotlib.imread for images.
- Ensure permissions and respect robots.txt/terms.

---

2. Data Manipulation Programs:

- **Q.** Write a small code to perform following operations on data: Slicing, Dicing, Concatenation, Transformation. **(WINTER 2022, 7 Marks)**

Answer:

```
import pandas as pd
import numpy as np

# Sample data
df1 = pd.DataFrame({"A": [1,2,3], "B": [4,5,6]})
df2 = pd.DataFrame({"A": [7,8], "B": [9,10]})

# Slicing: select rows/columns
slice_df = df1.iloc[0:2, 0:1]        # first 2 rows, first column
print("Slicing:\n", slice_df)

# Dicing: select arbitrary subsets
dice_df = df1.loc[[0,2], ["B"]]
print("Dicing:\n", dice_df)

# Concatenation: combine DataFrames
concat_df = pd.concat([df1, df2], ignore_index=True)
print("Concatenation:\n", concat_df)

# Transformation: apply function / create new column
df1["C"] = df1["A"] * 10
print("Transformation:\n", df1)
```

- **Explanation:** iloc slices by position; loc selects by labels; pd.concat stacks rows; transformation uses vectorized operations or apply().

- **Q.** Explain time transformation with suitable program. **(SUMMER 2025, 7 Marks OR)**

Answer:

- Time transformation converts date/time strings into datetime objects and extracts components (year, month, day, weekday, hour).

Program:

```
import pandas as pd

df = pd.DataFrame({"date": ["2026-01-15", "2026-02-20"]})
df["date"] = pd.to_datetime(df["date"])
df["year"] = df["date"].dt.year
df["month"] = df["date"].dt.month
df["day"] = df["date"].dt.day
df["weekday"] = df["date"].dt.day_name()
print(df)
```

- **Explanation:** pd.to_datetime() parses strings; .dt accessor extracts parts; day_name() gives weekday name.
- **Use:** Time-series analysis, grouping by month/year, feature engineering for ML.
- **Other transformations:** timezone conversion, resampling (resample('M').mean()), lag/diff, date arithmetic.

Chapter 5: Exploring Data Analysis (EDA)

Repeated Questions:

1. EDA Concepts:

- **Q.** Explain EDA / EDA approach in detail. **(SUMMER 2022, 7 Marks; WINTER 2022, 3 Marks; WINTER 2023, 7 Marks OR; SUMMER 2025, 7 Marks OR)**

Answer:

For 3 Marks:

- EDA (Exploratory Data Analysis) is an iterative process of summarizing and visualizing data to understand patterns, anomalies, and relationships before modeling.
- **Key steps:** Summarize data (mean, median, spread), visualize distributions (histogram, boxplot), check missing values/outliers, and examine relationships between variables.
- **Goal:** Form hypotheses, detect data quality issues, and guide feature selection/model choice.

For 7 Marks:

- **Data collection & inspection:** Load data, check shape, data types, sample rows (df.head(), df.info()).
- **Summary statistics:** Compute mean, median, mode, standard deviation, quartiles, min/max for numeric data.
- **Univariate analysis:** Plot histograms, boxplots, bar charts, count plots to understand each variable's distribution.
- **Missing values & outliers:** Identify nulls (df.isnull().sum()), visualize missingness, detect outliers using IQR/z-score.
- **Bivariate/multivariate analysis:** Use scatter plots, correlation matrix, grouped summaries, pivot tables to find relationships.
- **Categorical analysis:** Frequency tables, cross-tabulation, chi-square tests for association.
- **Feature engineering:** Create derived variables, transform skewed data, encode categorical variables.
- **Iteration & documentation:** Repeat as needed; record insights that guide modeling and business decisions.

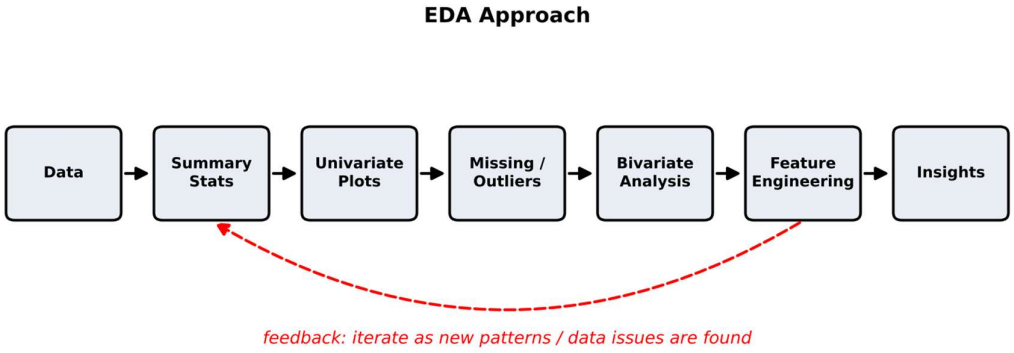


Fig. 8: EDA approach flowchart.

- **Q.** Define/Explain covariance and correlation and its importance in EDA with example. **(SUMMER 2023, 4 Marks; SUMMER 2024, 4 Marks; WINTER 2024, 3 Marks; SUMMER 2025, 3 Marks)**

Answer:

- **Covariance:** measures the joint variability of two variables: $Cov(X,Y) = \Sigma(xi - \bar{x})(yi - \bar{y}) / (n - 1)$. Positive means variables move together; negative means opposite; zero means no linear relationship. Units depend on variables.

- **Correlation:** standardizes covariance to [−1, 1]: r = Cov(X,Y) / (σX σY). It shows strength and direction of linear relationship, independent of units.
- **Importance in EDA:** Identifies redundant features, guides feature selection, detects multicollinearity, and suggests potential causal/associative patterns.
- **Example:** In a dataset of study hours and marks, positive correlation (r ≈ 0.8) suggests more study hours are associated with higher marks; near-zero correlation suggests no linear relationship.
- **Note:** Correlation does not imply causation.

---

2. Statistical Measures:

- **Q.** Explain skewness and kurtosis / use of skew() and kurtosis() function / program to demonstrate skewness and kurtosis. **(WINTER 2022, 7 Marks OR; WINTER 2023, 4 Marks OR; WINTER 2024, 7 Marks; SUMMER 2025, 4 Marks)**

Answer:

- **Skewness:** measures asymmetry of a distribution. Skew = 0: symmetric; > 0: right-skewed (tail right); < 0: left-skewed (tail left).
- **Kurtosis:** measures tailedness/peakness relative to normal distribution. Excess kurtosis = 0 for normal; > 0: heavy tails/peaked; < 0: light tails/flat.
- **Why in EDA:** Skewness indicates need for transformation (log/sqrt); kurtosis flags outliers/heavy tails affecting models.
- **Pandas functions:** df['col'].skew() and df['col'].kurt() (excess kurtosis).
- **SciPy functions:** scipy.stats.skew() and scipy.stats.kurtosis().

Program:

```
import pandas as pd
import numpy as np
from scipy.stats import skew, kurtosis


data = [10, 12, 13, 15, 18, 20, 22, 25, 30, 100]  # right-skewed with outlier
s = pd.Series(data)
print("Pandas skew:", s.skew())
print("Pandas kurtosis:", s.kurt())
print("SciPy skew:", skew(data))
print("SciPy kurtosis:", kurtosis(data))
```

- **Interpretation:** Positive skew indicates right tail; high kurtosis indicates outlier/heavy tail. Consider log transform or outlier treatment.

- **Q.** Explain Z-score standardization / Why we need to perform Z-score standardization in EDA? **(WINTER 2022, 3 Marks OR; WINTER 2024, 3 Marks OR; SUMMER 2025, 4 Marks OR)**

Answer:

- Z-score standardization transforms a variable to mean 0 and standard deviation 1: z = (x − μ) / σ.
- **Why needed:** Puts variables on the same scale, so distance-based algorithms (KNN, K-Means, SVM) and regularized models are not dominated by large-scale features.
- **In EDA:** Helps compare distributions across variables with different units, detect outliers (|z| > 3), and prepare data for modeling.
- **Example:** If marks range 0–100 and income ranges 0–1,00,000, z-scoring makes both comparable.

```
from sklearn.preprocessing import StandardScaler
X = [[10], [20], [30], [40]]
scaled = StandardScaler().fit_transform(X)
print(scaled)
```

---

3. Categorical Data:

- **Q.** What do you mean by categorical data/variables? Explain with example/program. **(WINTER 2023, 3 Marks OR; SUMMER 2024, 3 Marks OR; SUMMER 2025, 7 Marks)**

Answer:

- Categorical data represents labels/groups rather than numeric quantities. Two types: Nominal (no order, e.g., color, city) and Ordinal (ordered, e.g., low/medium/high).
- **Examples:** Gender, department, product category, satisfaction level.
- **In EDA:** Analyze using frequency counts, bar charts, cross-tabulation, chi-square tests; encode for modeling (one-hot, label encoding).

Program:

```
import pandas as pd
df = pd.DataFrame({"Dept": ["HR", "IT", "HR", "Sales"], "Rating": ["High", "Low", "Medium",
"High"]})
print(df["Dept"].value_counts())
print(pd.crosstab(df["Dept"], df["Rating"]))
```

- **For 7 marks, add:** Ordinal encoding preserves order; one-hot encoding creates dummy variables; avoid using numeric codes as continuous values; categorical data often needs handling of rare categories and missing labels.

Other Important Questions:

1. Sampling and Statistics:

- **Q.** Describe sampling along with its types in detail with suitable example. **(SUMMER 2023, 7 Marks OR)**

Answer:

- Sampling is selecting a subset (sample) from a population to estimate characteristics without examining every unit.
- **Probability sampling:** Every unit has a known non-zero chance of selection.
- **Simple random sampling:** Randomly pick units; e.g., draw 100 students from a college list using random numbers.
- **Stratified sampling:** Divide population into homogeneous strata, sample from each; e.g., sample proportionally from each department.
- **Systematic sampling:** Select every k-th unit after random start; e.g., every 10th product from assembly line.
- **Cluster sampling:** Randomly select whole clusters/groups; e.g., select 5 schools and survey all students in them.
- **Non-probability sampling:** Selection not random; useful in exploratory/qualitative work.
- **Convenience sampling:** Available/easy units; e.g., survey people near campus.
- **Purposive/judgment sampling:** Expert selects representative units.
- **Quota sampling:** Fixed quotas per category.
- **Example:** To estimate average income of a city, use stratified sampling by income brackets to ensure representation.
- **Importance:** Reduces cost/time, enables inference, but sampling bias must be minimized.

- **Q.** Explain the importance of chi-square test in EDA. **(SUMMER 2024, 4 Marks OR)**

Answer:

- Chi-square test checks association between two categorical variables (independence) or goodness-of-fit to an expected distribution.

- **Formula:** $\chi^2 = \Sigma (Oi - Ei)^2 / Ei$, where O = observed, E = expected frequency.
- **In EDA:** Determines whether categorical features like gender and product preference are related; guides feature selection and business hypotheses.
- **Example:** Test if department and attrition are independent; low p-value (< 0.05) suggests significant association.
- **Assumptions:** Expected frequencies sufficiently large (typically ≥5); observations independent.

- **Q.** Write a note on different descriptive statistics measures for numeric data. **(SUMMER 2024, 7 Marks OR)**

Answer:

Central tendency:

- **Mean:** Average; sensitive to outliers.
- **Median:** Middle value; robust to outliers.
- **Mode:** Most frequent value; useful for discrete data.

Dispersion/spread:

- **Range:** Max − Min; simple but outlier-sensitive.
- **Variance:** Average squared deviation from mean.
- **Standard deviation:** Square root of variance; same units as data.
- **Interquartile range (IQR):** Q3 − Q1; robust spread measure.

Shape:

- **Skewness:** Asymmetry; positive = right tail, negative = left tail.
- **Kurtosis:** Tail heaviness/peakedness relative to normal.
- **Position/percentiles:** Quartiles (Q1, Q2, Q3), percentiles; used in boxplots and outlier detection.

Program example:

```
import pandas as pd
s = pd.Series([10, 20, 30, 40, 50])
print("Mean:", s.mean(), "Median:", s.median(), "Mode:", s.mode()[0])
print("Std:", s.std(), "Var:", s.var(), "IQR:", s.quantile(0.75)-s.quantile(0.25))
print("Skew:", s.skew(), "Kurtosis:", s.kurt())
```

- **Use in EDA:** Summarize distribution, compare groups, detect outliers, decide transformations, and communicate data characteristics.

****