**import pandas as pd**

**import numpy as np**

**\_**

**np.random.seed(50) #for consistency**

**data = {**

**'Student\_id': range(1, 51),**

**'Name': ['Student\_' + str(i) for i in range(1, 51)],**

**'Age': np.random.randint(18, 25, size=50),**

**'Gender': np.random.choice(['Male', 'Female'], size=50),**

**'Scores': [np.random.randint(50, 100, size=3).tolist() for \_ in range(50)],**

**'Attendance': np.random.randint(20,100,size=50),**

**'Grade': np.random.choice(['A', 'B', 'C', 'D', 'F'], size=50)**

**}**

**\_**

**df = pd.DataFrame(data)**

**def assign\_grade(scores):**

**avg\_score = np.mean(scores)**

**if avg\_score > 90:**

**return 'A'**

**elif avg\_score > 80:**

**return 'B'**

**elif avg\_score > 70:**

**return 'C'**

**elif avg\_score > 60:**

**return 'D'**

**else:**

**return 'F'**

**df['Grade'] = df['Scores'].apply(assign\_grade)**

**\_**

**#Introduce missing values and inconsistencies**

**df = pd.DataFrame(data)**

**df.loc[8, 'Age'] = np.nan**

**df.loc[29, 'Age'] = np.nan**

**df.loc[35, 'Age'] = np.nan**

**df.loc[11, 'Scores'] = None**

**df.loc[19, 'Scores'] = None**

**df.loc[9, 'Attendance'] = 105 #invalid percentage**

**df.loc[15, 'Grade'] = 'Z' #invalid grade**

**df.head(20)**

**\_**

**missing\_values=df.isnull().sum()**

**invalid\_attendence=df[(df['Attendance']<0)|(df['Attendance']>100)]**

**invalid\_grades=df[~df['Grade'].isin(['A','B','C','D','F'])]**

**print("Missing Values are :\n",missing\_values)**

**print("Invalid attendence :\n",invalid\_attendence)**

**print("invalid\_grades :\n",invalid\_grades)**

**\_  
df.head(20)  
\_**

**df['Age'] = df['Age'].fillna(df['Age'].median()) #fill by median**

**df['Attendance'] = df['Attendance'].apply(lambda x: 100 if x > 100 else (0 if x < 0 else x))**

**def handle\_invalid\_scores(scores):**

**if scores is None:**

**return [0, 0, 0]**

**return [max(0, min(100, score)) for score in scores]**

**df['Scores'] = df['Scores'].apply(handle\_invalid\_scores)**

**df['Grade'] = df['Scores'].apply(assign\_grade)**

**df['Grade'] = df['Grade'].apply(lambda x: x if x in ['A', 'B', 'C', 'D', 'F'] else 'F')**

**df.head(20) # Print first 20 rows**

**\_**

**df.loc[5, 'Age'] = 35**

**df.loc[5, 'Age'] = 50**

**df.loc[5, 'Age'] = 65**

**df.loc[10, 'Attendance'] = 200**

**df.loc[12, 'Attendance'] = 175**

**df.loc[12, 'Attendance'] = 166**

**print("DataFrame with Outliers:")**

**print(df.iloc[5:20])**

**\_**

**def handle\_outliers\_iqr(df, column):**

**Q1 = df[column].quantile(0.25)**

**Q3 = df[column].quantile(0.75)**

**IQR = Q3 - Q1**

**lower\_bound = Q1 - 1.5 \* IQR**

**upper\_bound = Q3 + 1.5 \* IQR**

**df[column] = df[column].apply(lambda x: upper\_bound if x > upper\_bound else (lower\_bound if x < lower\_bound else x))**

**handle\_outliers\_iqr(df, 'Age')**

**handle\_outliers\_iqr(df, 'Attendance')**

**print(df.iloc[5:20])**

**\_**

**df['Scaled\_Attendance'] = (df['Attendance'] - df['Attendance'].min()) / (df['Attendance'].max() - df['Attendance'].min())**

**print("DataFrame with Min-Max Scaling on 'Attendance':")**

**print(df[['Attendance', 'Scaled\_Attendance']].head(20))**

**\_**

**🔍 What is .loc[] in Pandas?**

.loc[] is used to **access or modify specific rows and columns** in a Pandas DataFrame **by label/index**.

Think of it like:

“Hey DataFrame, go to this row and this column, and either get or set the value.”

### What is an ****Outlier****?

An **outlier** is a value that lies far outside the range of the rest of the data. It's an observation that is significantly different from other data points. Outliers can often skew statistical analysis and affect the performance of machine learning models.

In this specific case, you're using the row with Age to create **outliers**:

#### Normal Age Range:

* Ages are typically between 18 and 25 for students (based on the previous data generation).

#### Outliers Introduced:

* **35**, **50**, and **65** are clearly **outliers** because they are much higher than the typical age range in this dataset (18–25). These extreme values are much higher than most of the other ages in the dataset, so they would be considered outliers.

Sure! Let's break down the **IQR method** and other common methods for handling **outliers** in easy language.

### 1. ****What is IQR (Interquartile Range)?****

Imagine you have a bunch of numbers (data). The **IQR method** is a way to **detect outliers** by looking at the spread of the middle values in your data.

Here’s how it works:

* **Step 1: Split the data into quartiles.**
  + **Q1 (First Quartile)**: This is the point where 25% of the data is below it. In other words, it’s the value at the 25th percentile of your data.
  + **Q3 (Third Quartile)**: This is the point where 75% of the data is below it. It’s the value at the 75th percentile.
* **Step 2: Calculate the IQR (Interquartile Range).**
  + **IQR = Q3 - Q1**. It tells you how spread out the middle 50% of the data is.
* **Step 3: Define the "outlier" range.**
  + Outliers are values that fall too far from the middle 50% of the data. To find these, you calculate:
    - **Lower bound**: Q1 - 1.5 \* IQR
    - **Upper bound**: Q3 + 1.5 \* IQR
* **Step 4: Identify outliers.**
  + If any data points are **below the lower bound** or **above the upper bound**, they are considered outliers.
* **Step 5: Handle the outliers.**
  + Once identified, outliers can be **removed**, **replaced**, or **capped**.

### Example of IQR Method:

Imagine this data (ages of students):  
[18, 22, 24, 25, 30, 35, 40, 45, 50, 100]

1. **Step 1: Find Q1 and Q3.**
   * Q1 = 22 (25th percentile)
   * Q3 = 45 (75th percentile)
2. **Step 2: Calculate IQR.**
   * IQR = 45 - 22 = 23
3. **Step 3: Calculate lower and upper bounds.**
   * Lower bound = Q1 - 1.5 \* IQR = 22 - 1.5 \* 23 = -5.5
   * Upper bound = Q3 + 1.5 \* IQR = 45 + 1.5 \* 23 = 72.5
4. **Step 4: Identify outliers.**
   * Any value **below -5.5** or **above 72.5** is an outlier.
   * In this case, **100** is an outlier because it's greater than 72.5.

### 2. ****Other Methods to Handle Outliers:****

#### ****Method 1: Z-Score Method****

* **What is Z-Score?**  
  A **Z-score** tells you how many standard deviations a data point is from the mean (average).
  + Formula: **Z = (X - Mean) / Standard Deviation**
  + If a Z-score is greater than **+3** or less than **-3**, the data point is considered an outlier.
* **How to use it?**
  + Calculate the Z-score for each data point.
  + If the Z-score is above 3 or below -3, that data point is an outlier.

#### ****Example:****

* Suppose the mean age is 30, and the standard deviation is 10.
* For age 50: **Z = (50 - 30) / 10 = 2**.  
  This is **not an outlier** because it’s less than 3.
* For age 100: **Z = (100 - 30) / 10 = 7**.  
  This **is an outlier** because it’s greater than 3.

#### ****Method 2: Visual Methods (Boxplot and Histogram)****

* **Boxplot**: A **boxplot** shows the distribution of the data and helps you see the spread of values. Outliers appear as points that are far away from the box (the "whiskers").
* **Histogram**: A **histogram** shows how often each range of values occurs. Outliers are values that are far away from the rest.

#### ****Method 3: Trimming (Removing Outliers)****

* **What is it?**  
  You **remove the outliers** by deleting them from your dataset. This method is useful when the outliers are mistakes or don’t make sense in the context of the data.
* **When to use it?**  
  If outliers are clearly errors (e.g., negative ages or impossible values), you might remove them.

#### ****Method 4: Capping (Winsorization)****

* **What is it?**  
  Instead of removing outliers, you can **cap** them to a reasonable value. For example, if the highest value is 1000, but you decide to cap all values above 500, you would replace values greater than 500 with 500.
* **How to do it?**
  + If a value exceeds the upper limit, replace it with the upper limit.
  + If a value is below the lower limit, replace it with the lower limit.

### Summary of Methods:

| **Method** | **What It Does** | **How It Works** |
| --- | --- | --- |
| **IQR (Interquartile Range)** | Detects outliers based on spread of data | Uses Q1, Q3, and IQR to find outliers outside the range |
| **Z-Score** | Detects outliers based on how far a value is from the mean | Uses the standard deviation to see how far a value is from the average |
| **Boxplot/Histogram** | Visual methods to detect outliers | Outliers are visible as distant points or bars in the graph |
| **Trimming (Removing)** | Removes outliers entirely from the data | Simply deletes outliers from the dataset |
| **Capping (Winsorization)** | Replaces outliers with a cap value | Replaces extreme values with a reasonable upper or lower limit |

### When to Use Each Method:

* **IQR**: Best when you want a method that considers the spread of data.
* **Z-Score**: Works well when data is **normally distributed** and you want a statistical approach.
* **Boxplot/Histogram**: Best for **visual inspection** of data.
* **Trimming**: Use when outliers are clearly mistakes.
* **Capping**: Use when you want to keep extreme values but make them more reasonable.

I hope this explanation helps you understand **outlier detection** in a simpler way! Let me know if you need any more details.

**🔢 What is Min-Max Scaling?**

It is a technique to **rescale values** between **0 and 1**.

The formula is:

Scaled Value=Original Value−MinimumMaximum−Minimum\text{Scaled Value} = \frac{\text{Original Value} - \text{Minimum}}{\text{Maximum} - \text{Minimum}}Scaled Value=Maximum−MinimumOriginal Value−Minimum​

**🧠 Why do we use it?**

* To bring all values in a column to the **same scale** (0 to 1).
* This is especially useful in machine learning or plotting where large value ranges can cause problems.

Sure Vishwas! Here's the short and clear version:

**1. Methods to detect outliers:**

* **IQR Method**: Values outside [Q1 - 1.5×IQR, Q3 + 1.5×IQR] are outliers.
* **Z-Score Method**: Values with Z-score > 3 or < -3 are outliers.
* **Boxplot**: Visual method to detect outliers.
* **Scatter Plot**: Points that are far away from the cluster.

**2. Data transformation methods:**

* **Normalization**: Scale data between 0 and 1 (Min-Max Scaling).
* **Standardization**: Transform data to have mean 0 and std deviation 1.
* **Log Transformation**: Reduce skewness by applying log(x).
* **Box-Cox / Yeo-Johnson**: Advanced transformations to make data normal.

**3. Algorithm to display statistics of Null values:**

# Algorithm:

- For each column in the dataset:

- Count number of missing (null) values.

- Print column name and null count.

# Code Example:

null\_values = df.isnull().sum()

print(null\_values)

**4. Algorithm to replace outlier value with mean:**

# Algorithm:

- Calculate mean of the column.

- Detect outliers (using IQR or Z-Score method).

- Replace outlier values with mean.

# Code Example using IQR:

Q1 = df['column'].quantile(0.25)

Q3 = df['column'].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

mean\_value = df['column'].mean()

df['column'] = df['column'].apply(lambda x: mean\_value if x < lower\_bound or x > upper\_bound else x)

Would you also like a **super short handwritten-style summary** if you want to revise faster? 📒✨

Sure! Here's a simple **example of the IQR method** to detect outliers **without code**:

**🧠 Example:**

Suppose we have these **ages** of students:  
**[18, 19, 20, 21, 22, 23, 24, 25, 60]**

**🔹 Step 1: Find Q1 and Q3**

* **Q1 (25th percentile)** = 20
* **Q3 (75th percentile)** = 24

**🔹 Step 2: Calculate IQR**

* **IQR = Q3 - Q1 = 24 - 20 = 4**

**🔹 Step 3: Calculate Bounds**

* **Lower Bound = Q1 - 1.5 × IQR = 20 - 6 = 14**
* **Upper Bound = Q3 + 1.5 × IQR = 24 + 6 = 30**

**🔹 Step 4: Detect Outliers**

Any value **below 14** or **above 30** is an **outlier**.

In our data:  
**[60] is greater than 30**, so **60 is an outlier** ✅

Let me know if you want one more example!

Of course Vishwas! 🔥  
I'll now give you **Viva Questions with Detailed Answers** — exactly how you should reply in your viva.  
I'll also make sure the answers are simple, confident, and easy to remember. 🎯

# 🎯 ****Viva Questions with Detailed Answers (Based on Your Code)****

### 1. ****What is Data Wrangling? Why is it important?****

**Answer:**  
Data wrangling is the process of cleaning, transforming, and organizing raw data into a usable format.  
It is important because real-world data is often messy — it can have missing values, wrong entries, or outliers.  
Without wrangling, the data would lead to wrong analysis or poor machine learning models.

### 2. ****What types of missing values did you find in your dataset? How did you handle them?****

**Answer:**  
I found missing values in the Age column (NaN values) and missing entries in the Scores column (None values).

* I handled missing Age values by filling them with the **median** age, because median is less sensitive to outliers.
* I handled missing Scores by replacing them with [0, 0, 0], meaning the student scored zero in all subjects.

### 3. ****What inconsistencies did you find? How did you correct them?****

**Answer:**  
I found two types of inconsistencies:

* **Attendance** greater than 100% (e.g., 105%).
  + I corrected it by capping attendance to 100%.
* **Invalid Grades** like 'Z'.
  + I reassigned grades based on the average of the scores using a function.

### 4. ****Why did you choose median to fill missing values for Age instead of mean?****

**Answer:**  
I chose the **median** because it is more robust to outliers.  
If there are very high or low extreme values, the mean can get pulled away, but the median remains stable.  
Since Age can have outliers, median was a safer choice.

### 5. ****How did you detect outliers in Age and Attendance?****

**Answer:**  
I used the **IQR (Interquartile Range)** method.

* I calculated Q1 (25th percentile) and Q3 (75th percentile).
* IQR = Q3 - Q1.
* Any value outside [Q1 - 1.5×IQR, Q3 + 1.5×IQR] was considered an outlier.

### 6. ****How did you treat the outliers?****

**Answer:**  
I used **capping** to treat the outliers.

* If a value was greater than the upper bound, I replaced it with the upper bound.
* If a value was lower than the lower bound, I replaced it with the lower bound.

This way, I kept all rows without removing any important data.

### 7. ****Why did you apply Min-Max Scaling to Attendance?****

**Answer:**  
I applied Min-Max Scaling to **normalize** the Attendance between 0 and 1.

* It makes the scale easier to understand and compare.
* It prepares the data for machine learning models which work better when features are on the same scale.
* It reduces the effect of high numeric ranges.

### 8. ****What is the formula for Min-Max Scaling?****

**Answer:**  
The formula is:

Scaled Value=Value−Min(Value)Max(Value)−Min(Value)\text{Scaled Value} = \frac{\text{Value} - \text{Min(Value)}}{\text{Max(Value)} - \text{Min(Value)}}

It compresses all values between 0 and 1.

### 9. ****Explain the**** assign\_grade ****function.****

**Answer:**  
The assign\_grade function takes the average of the three scores of a student.

Based on the average:

* 90 ⇒ Grade 'A'
* 80 ⇒ Grade 'B'
* 70 ⇒ Grade 'C'
* 60 ⇒ Grade 'D'
* Otherwise ⇒ Grade 'F'

### 10. ****What is the purpose of**** np.random.seed(50) ****in your code?****

**Answer:**  
np.random.seed(50) is used to make the random numbers predictable and reproducible.

Every time I run the code, I will get the **same random dataset**.

### 11. ****Why did you use a lambda function for Attendance correction?****

**Answer:**  
I used a lambda function for quick, one-line checking.

* If Attendance > 100, set it to 100.
* If Attendance < 0, set it to 0.
* Otherwise, keep the original value.

### 12. ****What would happen if you don't handle outliers?****

**Answer:**  
Outliers can negatively impact data analysis and machine learning models.

* They can shift mean values, affect standard deviation, and cause wrong model predictions.
* Some models like Linear Regression are highly sensitive to outliers.

### 13. ****What is the purpose of**** handle\_invalid\_scores ****function?****

**Answer:**  
The handle\_invalid\_scores function:

* Checks if Scores are missing (None) and replaces them with [0,0,0].
* Ensures all individual scores are between 0 and 100 using max-min boundary checks.

### 14. ****Suppose you have categorical missing values (like Gender missing). How would you handle it?****

**Answer:**

* If Gender is missing, I can fill it with the **mode** (most frequent value) of the Gender column.
* Another option is to create a separate category called "Unknown".

### 15. ****Can Min-Max scaling be harmful sometimes?****

**Answer:**  
Yes, Min-Max scaling can be harmful if there are **new extreme values** in future data.

* New values outside the original range will give scaled values <0 or >1.
* In such cases, **Standardization** (Z-score) is preferred.

Perfect, Vishwas! 🌟  
Here’s the full **basic viva questions with short, clear answers** — like a **"last-minute cheatsheet"** you can revise quickly. 📚⚡

# 📋 ****Basic Viva Questions with Answers (for Data Wrangling Practical)****

### 1. ****What is data wrangling?****

👉 Data wrangling is the process of cleaning, transforming, and organizing raw data into a usable format for analysis.

### 2. ****Why is data cleaning important before analysis?****

👉 Because dirty data (missing, inconsistent, or wrong data) can mislead the analysis and produce wrong conclusions.

### 3. ****What methods are used to handle missing values?****

👉 We can handle missing values by removing them, filling them with mean/median/mode, or predicting them using models.

### 4. ****What are outliers? Why should we handle them?****

👉 Outliers are extreme values that differ a lot from other observations. They can skew results and affect model performance.

### 5. ****How can you detect outliers in a dataset?****

👉 Using methods like the Interquartile Range (IQR), boxplots, or Z-score methods.

### 6. ****What is the IQR method?****

👉 IQR is the range between the 25th percentile (Q1) and 75th percentile (Q3). Values outside [Q1–1.5IQR, Q3+1.5IQR] are outliers.

### 7. ****What is scaling? Why is it necessary?****

👉 Scaling changes the range of variables. It makes sure features are comparable and improves the performance of ML models.

### 8. ****What is the difference between Min-Max Scaling and Standardization?****

👉

* **Min-Max Scaling**: Rescales data between 0 and 1.
* **Standardization**: Centers the data around 0 mean with a standard deviation of 1.

### 9. ****What did you do when you found invalid data in your dataset?****

👉 I corrected invalid Attendance values to within 0-100 and assigned correct Grades where wrong grades were present.

### 10. ****How did you handle missing values in your project?****

👉 I filled missing Age values with the median and replaced missing Scores with [0, 0, 0].

### 11. ****What is the purpose of capping outliers instead of removing them?****

👉 Capping adjusts extreme values without losing records, thus preserving important data.

### 12. ****Explain any one function you wrote in your project.****

👉 I wrote assign\_grade(scores) to calculate the average score and assign grades A, B, C, D, or F based on rules.

### 13. ****What happens if we don't handle missing values?****

👉 It can cause errors in calculations, wrong analysis, or failure of machine learning models.

### 14. ****What happens if we don't handle outliers?****

👉 Outliers can distort mean, standard deviation, and can reduce the accuracy of models.

### 15. ****What libraries did you use in your project?****

👉 I used **Pandas** and **NumPy** for data handling and transformation.

### 16. ****What is the use of**** np.random.seed()****?****

👉 It ensures reproducibility — random numbers stay the same each time the code is run.

### 17. ****What is the benefit of using median instead of mean for missing values?****

👉 Median is not affected by extreme outliers, making it more robust than mean.

### 18. ****Can you explain the Min-Max scaling formula?****

👉

Xscaled=X−XminXmax−XminX\_{\text{scaled}} = \frac{X - X\_{\text{min}}}{X\_{\text{max}} - X\_{\text{min}}}

It rescales feature values between 0 and 1.

### 19. ****What are common techniques for handling missing categorical data?****

👉 Filling with the mode (most frequent value), creating a new 'Unknown' category, or predicting the missing value.

### 20. ****Why is it important to preprocess data before feeding it into machine learning models?****

👉 Clean and formatted data ensures better model accuracy, faster training, and more reliable results.

# 1. ****Explain the methods to detect outliers.****

👉 **Outliers** are extreme values that differ significantly from other data points.  
**Methods to detect outliers are:**

* **Box Plot Method:**
  + Use a boxplot to visualize data.
  + Points outside the whiskers (1.5 \* IQR above Q3 or below Q1) are outliers.
* **Interquartile Range (IQR) Method:**
  + Calculate Q1 (25th percentile) and Q3 (75th percentile).
  + IQR = Q3 - Q1.
  + Outlier if:

value<Q1−1.5×IQRorvalue>Q3+1.5×IQR\text{value} < Q1 - 1.5 \times \text{IQR} \quad \text{or} \quad \text{value} > Q3 + 1.5 \times \text{IQR}

* **Z-Score Method:**
  + Calculate Z-score = (value - mean) / standard deviation.
  + If |Z| > 3, the value is considered an outlier.
* **Visualization Methods:**
  + Scatter plots and histograms can also help detect outliers visually.

# 2. ****Explain data transformation methods.****

👉 **Data transformation** is the process of converting data into a suitable format for analysis.  
**Common methods:**

* **Normalization (Min-Max Scaling):**
  + Rescales the data between 0 and 1.
  + Formula:

Xscaled=X−XminXmax−XminX\_{\text{scaled}} = \frac{X - X\_{\text{min}}}{X\_{\text{max}} - X\_{\text{min}}}

* **Standardization (Z-Score Scaling):**
  + Centers the data around 0 mean and standard deviation 1.
  + Formula:

Z=X−μσZ = \frac{X - \mu}{\sigma}

* **Log Transformation:**
  + Used to reduce skewness and handle exponential growth.
  + Apply log(x+1) to data.
* **Square Root Transformation:**
  + Reduces the effect of large values and stabilizes variance.
* **Box-Cox Transformation:**
  + Makes data more normally distributed using a mathematical transformation.

# 3. ****Write the algorithm to display the statistics of Null values present in the dataset.****

✅ **Algorithm Steps:**

1. **Start.**
2. **Load** the dataset using pandas.
3. **Use** isnull() function to find missing values.
4. **Sum** the missing values for each column using sum().
5. **Display** the count of missing values.
6. **End.**

✅ **Python Code:**

import pandas as pd

# Load dataset

df = pd.read\_csv('your\_dataset.csv')

# Find and display missing values statistics

missing\_values = df.isnull().sum()

print("Missing Values Statistics:\n", missing\_values)

# 4. ****Write an algorithm to replace the outlier value with the mean of the variable.****

✅ **Algorithm Steps:**

1. **Start.**
2. **Load** the dataset.
3. **Select** the numeric column.
4. **Calculate** mean of the column.
5. **Detect** outliers using IQR method.
6. **Replace** outliers with the mean value.
7. **End.**

✅ **Python Code:**

import pandas as pd

# Load dataset

df = pd.read\_csv('your\_dataset.csv')

# Select column

col = 'YourColumnName'

# Calculate mean

mean\_val = df[col].mean()

# Calculate IQR

Q1 = df[col].quantile(0.25)

Q3 = df[col].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

# Replace outliers with mean

df[col] = df[col].apply(lambda x: mean\_val if x < lower\_bound or x > upper\_bound else x)

print(df.head())

Of course Vishwas! 🔥  
I’ll now explain everything **even more deeply** — so that you understand it **completely** and can **easily explain in exams or viva**.  
Let’s go **line-by-line, concept-by-concept**, and I’ll cover even the **background theory** where needed.

# 📚 Full Detailed Explanation of the Code:

## 1. ****Import Libraries****

import pandas as pd

import numpy as np

* **pandas (pd)** → Used for **creating, modifying, analyzing** data tables (called **DataFrames**).
* **numpy (np)** → Used for **numerical operations** like random number generation, statistical calculations (mean, median, etc.).

✅ Without pandas and numpy, data handling would become manual and very hard.

## 2. ****Create Dataset****

np.random.seed(50)

* **np.random.seed(50)** → ensures that **random values** are **same every time** you run the code.
* It is useful when you want **reproducible results**.

data = {

'Student\_id': range(1, 51),

'Name': ['Student\_' + str(i) for i in range(1, 51)],

'Age': np.random.randint(18, 25, size=50),

'Gender': np.random.choice(['Male', 'Female'], size=50),

'Scores': [np.random.randint(50, 100, size=3).tolist() for \_ in range(50)],

'Attendance': np.random.randint(20, 100, size=50),

'Grade': np.random.choice(['A', 'B', 'C', 'D', 'F'], size=50)

}

* **Student\_id** → numbers 1 to 50.
* **Name** → Student\_1, Student\_2, … Student\_50.
* **Age** → random number between 18 and 24.
* **Gender** → random Male or Female.
* **Scores** → a list of 3 random numbers between 50 and 100 (marks in 3 subjects).
* **Attendance** → random number between 20% and 99%.
* **Grade** → Random letter among A, B, C, D, F.

✅ This makes the data look like **real college students’ records**.

## 3. ****Create DataFrame****

df = pd.DataFrame(data)

* **DataFrame** is like an **Excel sheet**: rows and columns.
* Now df holds all the generated student records.

## 4. ****Assign Grades Based on Average Score****

def assign\_grade(scores):

avg\_score = np.mean(scores)

if avg\_score > 90:

return 'A'

elif avg\_score > 80:

return 'B'

elif avg\_score > 70:

return 'C'

elif avg\_score > 60:

return 'D'

else:

return 'F'

df['Grade'] = df['Scores'].apply(assign\_grade)

* Function calculates the **average of 3 scores**.
* Depending on average:
  + 90 → A
  + 80 → B
  + 70 → C
  + 60 → D
  + Otherwise → F

✅ Assigns **realistic grades** instead of random ones.

## 5. ****Introduce Missing Values and Errors****

df.loc[8, 'Age'] = np.nan

df.loc[29, 'Age'] = np.nan

df.loc[35, 'Age'] = np.nan

df.loc[11, 'Scores'] = None

df.loc[19, 'Scores'] = None

df.loc[9, 'Attendance'] = 105

df.loc[15, 'Grade'] = 'Z'

* **Missing values** (Age and Scores) inserted manually.
* **Invalid Attendance** → 105% (above 100%, which is wrong).
* **Invalid Grade** → 'Z' (grades should only be A–F).

✅ This step is done **to simulate real-world dirty data** where mistakes happen during data collection.

Perfect! Vishwas, let’s again go very **slowly and clearly** 🧠.  
I’ll explain this line-by-line with examples and a simple mindset!

# 📚 Full Code:

missing\_values = df.isnull().sum() # check missing values

invalid\_attendance = df[(df['Attendance'] < 0) | (df['Attendance'] > 100)]

invalid\_grades = df[~df['Grade'].isin(['A', 'B', 'C', 'D', 'F'])]

print("Missing values:\n", missing\_values)

print("Invalid attendance:\n", invalid\_attendance)

print("Invalid grades:\n", invalid\_grades)

# 🛠 Step-by-Step Explanation:

## 1. Check Missing Values

missing\_values = df.isnull().sum()

👉 What's happening here?

* df.isnull() → checks each cell in the DataFrame:
  + If the value is **missing** (NaN or None), it returns **True**.
  + Otherwise, it returns **False**.

Example:

| **Name** | **Age** | **Scores** | **Attendance** | **Grade** |
| --- | --- | --- | --- | --- |
| A | 20 | 80 | 90 | B |
| B | NaN | None | 85 | A |

df.isnull() → will become:

| **Name** | **Age** | **Scores** | **Attendance** | **Grade** |
| --- | --- | --- | --- | --- |
| False | False | False | False | False |
| False | True | True | False | False |

* .sum() → **adds up** the number of True (missing) values **column-wise**.

Thus, missing\_values will tell you:

| **Column** | **Missing Values** |
| --- | --- |
| Name | 0 |
| Age | 1 |
| Scores | 1 |
| Attendance | 0 |
| Grade | 0 |

**Meaning:** 1 missing value in Age, 1 missing value in Scores.

✅ So, **missing\_values** gives a **summary of missing data** column-by-column.

## 2. Find Invalid Attendance

invalid\_attendance = df[(df['Attendance'] < 0) | (df['Attendance'] > 100)]

👉 What's happening here?

* df['Attendance'] → selects the Attendance column.
* Checking **two conditions**:
  + Attendance is **less than 0** (impossible!).
  + Attendance is **greater than 100** (also impossible!).
* | → means **OR**.

Thus:

* If **any student** has attendance **< 0** or **> 100**, they are **picked**.

🔵 Example:

| **Name** | **Attendance** |
| --- | --- |
| A | 90 |
| B | 105 |
| C | -5 |

→ B and C are invalid.

✅ **invalid\_attendance** will contain all students whose attendance is **wrong**.

## 3. Find Invalid Grades

invalid\_grades = df[~df['Grade'].isin(['A', 'B', 'C', 'D', 'F'])]

👉 What's happening here?

* df['Grade'].isin(['A', 'B', 'C', 'D', 'F'])
  + Checks if each student's grade is **in** the list of valid grades.
  + Returns True if valid, False if invalid.

🔵 Example:

| **Name** | **Grade** |
| --- | --- |
| A | A |
| B | C |
| C | Z |

isin(['A', 'B', 'C', 'D', 'F']) → returns:

| **Name** | **Valid** |
| --- | --- |
| A | True |
| B | True |
| C | False |

* ~ → means **NOT**.
  + Picks all the **False** (i.e., invalid grades).

Thus:

* C is picked because 'Z' is **not** in A, B, C, D, F.

✅ **invalid\_grades** will have all rows where grade is **invalid**.

## 4. Print Statements

print("Missing values:\n", missing\_values)

print("Invalid attendance:\n", invalid\_attendance)

print("Invalid grades:\n", invalid\_grades)

* Shows missing values,
* Shows which students have wrong attendance,
* Shows which students have invalid grades.

Certainly! Here is the explanation without examples:

### Step-by-Step Explanation of the Code

#### 1. ****Handling Missing Age Values (Filling with Median)****

df['Age'] = df['Age'].fillna(df['Age'].median()) # fill by median

* **df['Age']**: This selects the **Age** column in the DataFrame.
* **fillna(df['Age'].median())**: This method **fills missing values (NaN)** in the **Age** column with the **median value** of the **Age** column.
  + **Why the median?** The median is less affected by outliers compared to the mean, making it a more robust measure for filling missing values.

#### 2. ****Correcting Invalid Attendance Values****

df['Attendance'] = df['Attendance'].apply(lambda x: 100 if x > 100 else (0 if x < 0 else x))

* **.apply()**: This is used to apply a function (in this case, a lambda function) on each value of the **Attendance** column.
* **lambda x: 100 if x > 100 else (0 if x < 0 else x)**: This function ensures that:
  + If the attendance is greater than 100, it is set to **100**.
  + If the attendance is less than 0, it is set to **0**.
  + If the attendance is between 0 and 100, it remains unchanged.

#### 3. ****Handling Invalid Scores****

def handle\_invalid\_scores(scores):

if scores is None:

return [0, 0, 0]

return [max(0, min(100, score)) for score in scores]

df['Scores'] = df['Scores'].apply(handle\_invalid\_scores)

* **handle\_invalid\_scores()** is a function that:
  + Checks if the **scores** are **None**. If they are, it replaces the scores with [0, 0, 0].
  + If the **scores** are not None, it ensures that each score is within the range 0 to 100 using the max(0, min(100, score)) function.

#### 4. ****Assigning Grades Based on Scores****

df['Grade'] = df['Scores'].apply(assign\_grade)

* **assign\_grade()** is a function that calculates the grade based on the average score.
* **.apply(assign\_grade)** applies this function to the **Scores** column to calculate the corresponding grade for each student.

#### 5. ****Ensuring Valid Grades****

df['Grade'] = df['Grade'].apply(lambda x: x if x in ['A', 'B', 'C', 'D', 'F'] else 'F')

* **.apply()** is used here with a **lambda function** to check if the grade is valid.
  + If the grade is in the list ['A', 'B', 'C', 'D', 'F'], it remains unchanged.
  + If the grade is **invalid**, it is replaced with a default grade of **'F'**.

#### 6. ****Printing the First 20 Rows****

df.head(20) # Print first 20 rows

* This function displays the first 20 rows of the cleaned DataFrame.

handling the outliers:

Let's break down the code step-by-step:

### ****Code Explanation****

#### 1. handle\_outliers\_iqr(df, column) ****Function****

def handle\_outliers\_iqr(df, column):

Q1 = df[column].quantile(0.25)

Q3 = df[column].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

df[column] = df[column].apply(lambda x: upper\_bound if x > upper\_bound else (lower\_bound if x < lower\_bound else x))

* **df[column].quantile(0.25)**: This calculates the **25th percentile** (Q1) of the data in the specified **column**. It is also known as the **lower quartile**.
* **df[column].quantile(0.75)**: This calculates the **75th percentile** (Q3) of the data in the specified **column**. It is also known as the **upper quartile**.
* **IQR (Interquartile Range)**: It is calculated as:
  + **IQR = Q3 - Q1**.
  + The **IQR** represents the spread of the middle 50% of the data.
* **Lower and Upper Bounds**: These are used to identify the outliers in the data.
  + **Lower Bound** = **Q1 - 1.5 \* IQR**.
  + **Upper Bound** = **Q3 + 1.5 \* IQR**.
  + Any value below the lower bound or above the upper bound is considered an outlier.
* **Handling Outliers**: The **apply()** method is used to apply a lambda function on each value in the **column**.
  + If a value is greater than the **upper bound**, it is set to the **upper bound**.
  + If a value is less than the **lower bound**, it is set to the **lower bound**.
  + If a value is between the **lower bound** and **upper bound**, it remains unchanged.

This technique is known as **Outlier Treatment using IQR**.

#### 2. ****Applying the**** handle\_outliers\_iqr() ****Function****

handle\_outliers\_iqr(df, 'Age')

handle\_outliers\_iqr(df, 'Attendance')

* This applies the **outlier handling** function to the **Age** and **Attendance** columns of the DataFrame (df).
  + For the **Age** column, the function identifies and treats outliers based on the IQR.
  + For the **Attendance** column, the same function is applied to handle any outliers.

#### 3. ****Printing the Modified Data****

print(df.iloc[5:20])

* **df.iloc[5:20]**: This selects and prints the rows from index 5 to index 19 (20th row is not included).
* This allows you to observe the rows after the outliers have been handled and replaced with the bounds.

Min max Scaling:

### ****Code Explanation****

#### 1. ****Min-Max Scaling****

df['Scaled\_Attendance'] = (df['Attendance'] - df['Attendance'].min()) / (df['Attendance'].max() - df['Attendance'].min())

* **Min-Max Scaling** is a technique used to normalize the data in a range, typically between 0 and 1. This is done to bring all the features to the same scale, which is particularly useful in machine learning algorithms that are sensitive to the magnitude of data (e.g., distance-based algorithms like KNN, clustering algorithms).

Here’s how the code works:

* **df['Attendance'].min()**: This calculates the minimum value in the **Attendance** column.
* **df['Attendance'].max()**: This calculates the maximum value in the **Attendance** column.
* **The formula for Min-Max scaling** is:

Scaled Value=(Value−Min)(Max−Min)\text{Scaled Value} = \frac{(\text{Value} - \text{Min})}{(\text{Max} - \text{Min})}

* + This formula transforms all values to a scale between 0 and 1, where the minimum value in the column becomes 0, and the maximum value becomes 1.
* **df['Scaled\_Attendance']**: The result of applying the scaling formula to each value in the **Attendance** column is stored in a new column **Scaled\_Attendance**.

#### 2. ****Displaying the Result****

print("DataFrame with Min-Max Scaling on 'Attendance':")

print(df[['Attendance', 'Scaled\_Attendance']].head(20))

* **df[['Attendance', 'Scaled\_Attendance']]**: This selects only the **Attendance** and **Scaled\_Attendance** columns from the DataFrame for display.
* **.head(20)**: This will show the first 20 rows of the **Attendance** and **Scaled\_Attendance** columns.

### ****Purpose of the Code****

* This code applies **Min-Max Scaling** to the **Attendance** column in the dataset and creates a new column **Scaled\_Attendance** to hold the normalized values.
* It then prints out the first 20 rows showing both the original and scaled **Attendance** values.