EXPLORATORY DATA SCIENCE

Zlatan Firmansyah P. K.

IMPORTING AND LOADING THE DATASET

This code imports the pandas library and loads a CSV file named "student_admission_record_dirty.csv" into a DataFrame called df. The dataset is then displayed to get an overview of the data structure, which consists of 157 rows and 7 columns, including Name, Age, Gender, Admission Test Score, High School Percentage, City, and Admission Status. This step is essential to ensure the data has been successfully imported before proceeding with further analysis.

```
[ ] import pandas as pd

    df = pd.read_csv('/content/student_admission_record_dirty.csv')
```

```
[ ] df
```

	Name	Age	Gender	Admission Test Score	High School Percentage	City	Admission Status
0	Shehroz	24.0	Female	50.0	68.90	Quetta	Rejected
1	Waqar	21.0	Female	99.0	60.73	Karachi	NaN
2	Bushra	17.0	Male	89.0	NaN	Islamabad	Accepted
3	Aliya	17.0	Male	55.0	85.29	Karachi	Rejected
4	Bilal	20.0	Male	65.0	61.13	Lahore	NaN
...	...	...	...	...	...	...	...
152	Ali	19.0	Female	85.0	78.09	Quetta	Accepted
153	Bilal	17.0	Female	81.0	84.40	Islamabad	Rejected
154	Fatima	21.0	Female	98.0	50.86	Multan	Accepted
155	Shoaib	-1.0	Male	91.0	80.12	Quetta	Accepted
156	Maaz	17.0	Male	88.0	86.85	Lahore	Accepted

157 rows × 7 columns

CHECKING DATASET INFORMATION

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 157 entries, 0 to 156
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Name                    147 non-null    object
 1   Age                     147 non-null    float64
 2   Gender                  147 non-null    object
 3   Admission Test Score    146 non-null    float64
 4   High School Percentage  146 non-null    float64
 5   City                    147 non-null    object
 6   Admission Status        147 non-null    object
dtypes: float64(3), object(4)
memory usage: 8.7+ KB
```

The command df.info() is used to display a concise summary of the dataset. It provides key information such as the total number of entries (157), the number of non-null values in each column, the data types of each column, and the overall memory usage. This step helps to quickly identify missing values and understand the structure and types of data available, which is crucial for determining the necessary data cleaning and preprocessing steps.

CHECKING MISSING VALUES

The command df.isna().sum() is used to identify the number of missing (NaN) values in each column of the dataset. The output shows that several columns, such as Name, Age, Gender, City, and Admission Status, each have 10 missing values, while Admission Test Score and High School Percentage have 11 missing values. Detecting these missing entries is essential before proceeding to data cleaning, as they can significantly affect the accuracy of data analysis and model performance.

```
df.isna().sum()
```

	0
Name	10
Age	10
Gender	10
Admission Test Score	11
High School Percentage	11
City	10
Admission Status	10

dtype: int64

DESCRIPTIVE STATISTICS SUMMARY

The df.describe() function summarizes numerical data with metrics like count, mean, min, max, and quartiles. The results reveal invalid values, such as negative ages and scores exceeding realistic limits, indicating data entry errors that need cleaning.

```
[ ] df.describe()
```

	Age	Admission Test Score	High School Percentage
count	147.000000	146.000000	146.000000
mean	19.680272	77.657534	75.684726
std	4.540512	16.855343	17.368014
min	-1.000000	-5.000000	-10.000000
25%	18.000000	68.250000	65.052500
50%	20.000000	79.000000	77.545000
75%	22.000000	89.000000	88.312500
max	24.000000	150.000000	110.500000

ADMISSION STATUS SUMMARY

```
df['Admission Status'].describe()
```

	Admission Status
count	147
unique	2
top	Rejected
freq	76

dtype: object

The df['Admission Status'].describe() function summarizes categorical data. It shows 147 non-null entries, with 2 unique values ("Accepted" and "Rejected"). The most frequent status is "Rejected", appearing 76 times, providing a quick overview of class distribution.

HANDLING MISSING VALUES

```python
# Mengatasi missing value
for column in df.columns:
    if df[column].dtype == 'object':
        # Jika kolom bertipe object, isi dengan mode
        df[column].fillna(df[column].mode()[0], inplace=True)
    else:
        # Jika kolom bertipe numerik, isi dengan mean
        df[column].fillna(df[column].mean(), inplace=True)
```

```
<ipython-input-7-da1a6285f769>:5: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform


  df[column].fillna(df[column].mode()[0], inplace=True)
<ipython-input-7-da1a6285f769>:8: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform


  df[column].fillna(df[column].mean(), inplace=True)
```

This loop iterates through each column to handle missing values. If the column type is object (categorical), it fills missing values with the most frequent value (mode). For numeric columns, it fills missing values with the column's mean. Although it works, this code triggers a FutureWarning in pandas due to chained assignment, which may behave differently in future versions.

CHECKING MISSING VALUES AFTER IMPUTATION

```
df.isna().sum()
```

	0
Name	0
Age	0
Gender	0
Admission Test Score	0
High School Percentage	0
City	0
Admission Status	0

dtype: int64

The command df.isna().sum() is used to verify if there are any remaining missing values after the imputation process. The result shows 0 missing values in all columns, indicating that the missing data has been successfully handled.

DATA SUMMARY AFTER HANDLING MISSING VALUES

The df.info() result shows that all columns now have 157 non-null entries, indicating that missing values have been fully addressed. This ensures the dataset is clean and consistent, making it ready for reliable analysis and modeling without data quality issues.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 157 entries, 0 to 156
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Name                    157 non-null    object
 1   Age                     157 non-null    float64
 2   Gender                  157 non-null    object
 3   Admission Test Score    157 non-null    float64
 4   High School Percentage  157 non-null    float64
 5   City                    157 non-null    object
 6   Admission Status        157 non-null    object
dtypes: float64(3), object(4)
memory usage: 8.7+ KB
```

DUPLICATE DATA CHECK

```
[ ] # Mengecek apakah ada duplicate di seluruh kolom
    check_duplicate = df.duplicated().sum()

    print(f"Jumlah data yang duplikat = {check_duplicate}")
```

```
Jumlah data yang duplikat = 7
```

The result shows there are 7 duplicated records in the dataset, identified using df.duplicated().sum(). These duplicates should be handled appropriately to prevent skewed analysis and ensure data accuracy.

HANDLING DUPLICATES

```python
# Handling duplicate
df = df.drop_duplicates()
```

```python
# Mengecek duplicate setelah di-handle
handle_duplicate = df.duplicated().sum()

print(f"Jumlah data yang duplikat = {handle_duplicate}")
```

```
Jumlah data yang duplikat = 0
```

The duplicates were successfully removed using df.drop_duplicates(). A recheck confirmed that no duplicated records remain (0 duplicates), ensuring the dataset's integrity for further analysis.

THANK YOU FOR YOUR ATTENTION AND TIME.