Week 2 Course 2 Assignment: Preprocessing Nigerian Business Funding Data

Azeez Azeez Ajibola

```
#Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#Load the dataset
df = pd.read_csv('Business Funding Data.csv', encoding='latin-1')
```

```
display(df.head())
```

	Website Domain	Effective date	Found At	Financing Type	Financing Type Normalized	Categories	Investo
0	trafigura.com	NaN	2024-03-14T01:00:00+01:00	NaN	NaN	[]	N
1	zenobe.com	NaN	2024-05-31T02:00:00+02:00	NaN	NaN	[]	avivainvestors.c lloydsbankinggroup.c s
2	zenobe.com	NaN	2024-07-24T02:00:00+02:00	NaN	NaN	["private_equity"]	N
3	canva.com	NaN	2024-05-01T02:00:00+02:00	NaN	NaN	[]	stackcapitalgroup.c
4	fidelity.com	NaN	2024-04-11T02:00:00+02:00	NaN	NaN	[]	chevychasetrust.c

Double-click (or enter) to edit

Summary of Data Exploration

Based on the initial exploration of the dataset:

- The dataset contains **26 entries** and **11 columns**.
- Several columns have **missing values**:
  - `Effective date`: 20 missing values
  - `Financing Type`: 18 missing values
  - `Financing Type Normalized`: 18 missing values
  - `Investors`: 13 missing values

- `Investors Count`: 13 missing values
- The data types are mostly `object` (strings), with `Amount Normalized` as `int64` and `Investors Count` as `float64`.
- The `Amount` column contains values with different currencies and formats, while `Amount Normalized` provides a standardized numerical representation of the funding amount.
- The numerical columns (`Investors Count` and `Amount Normalized`) have a wide range of values, indicated by the standard deviation and the difference between minimum and maximum values in the statistical summary. The median `Amount Normalized` is significantly lower than the mean, suggesting the presence of some large outliers.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 11 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Website Domain             26 non-null     object 
 1   Effective date             6 non-null      object 
 2   Found At                   26 non-null     object 
 3   Financing Type             8 non-null      object 
 4   Financing Type Normalized  8 non-null      object 
 5   Categories                 26 non-null     object 
 6   Investors                  13 non-null     object 
 7   Investors Count            13 non-null     float64
 8   Amount                     26 non-null     object 
 9   Amount Normalized          26 non-null     int64  
 10  Source Urls                26 non-null     object 
dtypes: float64(1), int64(1), object(9)
memory usage: 2.4+ KB
```

```
# Exploring the data to understand it
print("First 5 rows of the data:")
print(df.head())

print("\n\nInformation about the dataset (data types, null values):")
df.info()

print("\n\nStatistical summary of numerical columns:")
print(df.describe())

print("\n\nNumber of missing values in each column:")
print(df.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 11 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   Website Domain              26 non-null     object 
 1   Effective date              6 non-null      object 
 2   Found At                    26 non-null     object 
 3   Financing Type              8 non-null      object 
 4   Financing Type Normalized  8 non-null      object 
 5   Categories                  26 non-null     object 
 6   Investors                   13 non-null     object 
 7   Investors Count             13 non-null     float64
 8   Amount                      26 non-null     object 
 9   Amount Normalized           26 non-null     int64  
 10  Source Urls                 26 non-null     object 
dtypes: float64(1), int64(1), object(9)
memory usage: 2.4+ KB


Statistical summary of numerical columns:
       Investors Count  Amount Normalized
count        13.000000       2.600000e+01
mean          1.846154       2.264687e+08
std           2.230327       5.383239e+08
min           1.000000       1.600000e+06
25%           1.000000       4.685750e+06
50%           1.000000       1.160000e+07
75%           1.000000       4.750000e+07
max           9.000000       2.000000e+09


Number of missing values in each column:
Website Domain                0
Effective date               20
Found At                      0
Financing Type               18
Financing Type Normalized    18
Categories                    0
Investors                    13
Investors Count              13
Amount                        0
Amount Normalized             0
Source Urls                   0
dtype: int64
```

Steps and Justifications for Cleaning, Preprocessing, and Transformation

Below are the steps I took to clean the data and my reasons for each action.

Handling Missing Values

In this step, I adressed the missing information in the dataset. I filled in the gaps in columns that had missing entries to make the data complete for analysis.

For columns with text or categories (like 'Effective date', 'Financing Type', 'Financing Type Normalized', and 'Investors'), we filled the missing spots with the most frequent value found in that column. This is a common approach for categorical data.

For the column with numerical data ('Investors Count'), we filled the missing values with the median (the middle value) of that column. Using the median is helpful because it's not heavily affected by very large or very small numbers (outliers), giving a more typical value for the missing entries.

After filling in the missing data, we confirmed that there are no more missing values in the dataset.

```
# Handle missing values
# Fill missing values in object columns with mode
for col in ['Effective date', 'Financing Type', 'Financing Type Normalized', 'Investors']:
    if df[col].isnull().any():
        mode_value = df[col].mode()[0]
        df[col] = df[col].fillna(mode_value)

# Fill missing values in numerical columns with median
if df['Investors Count'].isnull().any():
    median_value = df['Investors Count'].median()
    df['Investors Count'] = df['Investors Count'].fillna(median_value)

display(df.isnull().sum())
```

	0
Website Domain	0
Effective date	0
Found At	0
Financing Type	0
Financing Type Normalized	0
Categories	0
Investors	0
Investors Count	0
Amount	0
Amount Normalized	0
Source Urls	0

dtype: int64

Correcting Data Types

```
# Convert date columns to datetime objects
df['Effective date'] = pd.to_datetime(df['Effective date'], errors='coerce')
df['Found At'] = pd.to_datetime(df['Found At'], errors='coerce', utc=True)

display(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 11 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Website Domain             26 non-null     object
 1   Effective date             26 non-null     datetime64[ns, UTC+02:00]
 2   Found At                   26 non-null     datetime64[ns, UTC]
 3   Financing Type             26 non-null     object
 4   Financing Type Normalized  26 non-null     object
 5   Categories                 26 non-null     object
 6   Investors                  26 non-null     object
 7   Investors Count            26 non-null     float64
 8   Amount                     26 non-null     object
 9   Amount Normalized          26 non-null     int64
 10  Source Urls                26 non-null     object
dtypes: datetime64[ns, UTC+02:00](1), datetime64[ns, UTC](1), float64(1), int64(1), object(7)
memory usage: 2.4+ KB
None
```

```
display(df.head())
```

	Website Domain	Effective date	Found At	Financing Type	Financing Type Normalized	Categories	Investo
0	trafigura.com	2024-04-16 02:00:00+02:00	2024-03-14 00:00:00+00:00	Seed	seed	[]	accelia
1	zenobe.com	2024-04-16 02:00:00+02:00	2024-05-31 00:00:00+00:00	Seed	seed	[]	avivainvestors.co lloydsbankinggroup.co s
2	zenobe.com	2024-04-16 02:00:00+02:00	2024-07-24 00:00:00+00:00	Seed	seed	["private_equity"]	accelia
3	canva.com	2024-04-16 02:00:00+02:00	2024-05-01 00:00:00+00:00	Seed	seed	[]	stackcapitalgroup.co
4	fidelity.com	2024-04-16 02:00:00+02:00	2024-04-11 00:00:00+00:00	Seed	seed	[]	chevychasetrust.co

Double-click (or enter) to edit

Handling Missing Values

Missing values in the dataset were addressed as follows:

- For categorical columns (`Effective date`, `Financing Type`, `Financing Type Normalized`, and `Investors`), missing values were imputed with the mode (most frequent value) of each respective column.
- For the numerical column (`Investors Count`), missing values were imputed with the median value to mitigate the influence of potential outliers.

After these steps, all missing values in the dataset were successfully handled.

Handling Duplicates

```
# Check for duplicates
print(f"Number of duplicate rows: {df.duplicated().sum()}")
```

```
Number of duplicate rows: 0
```

I checked for any fully duplicate rows to ensure that each record is unique. Duplicate data can skew analysis and lead to incorrect conclusions.

Reflections on the Importance of Preprocessing

Data preprocessing is arguably the most critical stage in any data analysis or machine learning project. The principle of "Garbage In, Garbage Out" (GIGO) perfectly applies here. Without rigorous preprocessing: Analysis would be inaccurate: Calculating the average funding amount would fail or give wrong results if the data isn't a clean numeric type. Models would be unreliable: A machine learning model trained on messy data with missing values and incorrect types will produce poor and untrustworthy predictions. Insights would be flawed: We might draw incorrect conclusions, for instance, by thinking a category is unpopular when in reality its name is just misspelled in various ways (e.g., 'Fintech', 'fintech', 'Fin-tech').

In a real-world scenario, clean and well-prepared data is the foundation upon which all trustworthy business decisions are built. This exercise shows that raw data is rarely perfect and that a data scientist's first duty is to bring order and structure to it.