Part 1: Identify all inconsistencies in the dataset.

In [1]:
```python
# Importing Libraries
import pandas as pd
```

In [2]:
```python
# Reading the datasets:
df = pd.read_csv(r"A:\Data Mining\Dataset\dataset.txt", sep = ",")
df
```

Out[2]:

	custID	custName	Age	Product	DatePurchased	Price	RatingOfProduct	AdvertisingAgency
0	1	John Doe	30.0	Shirt	2015-01-10	25.99	4.5	Social Media
1	2	Jane Smith	28.0	Shoes	2015-02-15	59.99	3.8	TV
2	3	Robert Johnson	NaN	Hat	2015-03-20	12.99	4.2	Newspapers
3	4	Sarah Williams	35.0	Jeans	2015-04-05	39.99	4.0	NaN
4	5	Michael Brown	32.0	Shirt	2015-05-12	NaN	3.5	Social Media
...	...	...	...	...	...	...	...	...
95	96	Isabella Lee	35.0	Shirt	2014-10-02	18.99	4.4	Newspapers
96	97	Noah Turner	27.0	Shoes	NaN	99.99	4.9	Social Media
97	98	Mia Miller	38.0	Jeans	2014-11-16	44.99	4.2	TV
98	99	Charlotte Wilson	36.0	Shirt	2014-12-23	26.99	NaN	Newspapers
99	100	Mason Davis	27.0	Shoes	2013-01-30	79.99	4.8	Social Media

100 rows × 8 columns

In [3]:
```python
df.describe()
```

Out[3]:

	custID	Age	Price	RatingOfProduct
count	100.000000	93.000000	96.000000	93.000000
mean	50.500000	32.688172	51.844167	4.466667
std	29.011492	4.522837	28.160435	0.366930
min	1.000000	25.000000	12.990000	3.500000
25%	25.750000	29.000000	25.490000	4.100000
50%	50.500000	33.000000	47.490000	4.500000
75%	75.250000	36.000000	79.990000	4.800000
max	100.000000	45.000000	99.990000	4.900000

In [4]:
```python
# Checking for null Values
df.isnull().sum()
```

Out[4]:
```
custID               0
custName             1
Age                  7
Product              1
DatePurchased        7
Price                4
RatingOfProduct      7
AdvertisingAgency    6
dtype: int64
```

From the above dataset, there are columns with Null values which gives the first inconsisten data

In [5]:
```python
# Check for Duplicates
df.duplicated().sum()
```

Out[5]:
```
0
```

There are no duplicate values in the dataset

In [6]:
```python
# Checking the date Values if they are valid
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   custID             100 non-null    int64  
 1   custName           99 non-null     object 
 2   Age                93 non-null     float64
 3   Product            99 non-null     object 
 4   DatePurchased      93 non-null     object 
 5   Price              96 non-null     float64
 6   RatingOfProduct    93 non-null     float64
 7   AdvertisingAgency  94 non-null     object 
dtypes: float64(3), int64(1), object(4)
memory usage: 6.4+ KB
```

The Date column is not consistent, it should be in the correct date format that is "datetime64"

Part 2: Discuss in detail the FIVE techniques/methods you can use to solve the inconsistencies identified in Part 1. How can you ensure that data is correctly captured during data collection?

Techniques used to solve inconsistencies.

```
i) Using Imputation - This involves filling the missing values with estimated or calculated values. This includes mean median or mode. for our scenario, we
will fill in the missing values in the Price column with the mean and use the median to fill the missing values for the rating column.
ii) Deleting Rows with Missing Values. This involves removing rows with missing values in the dataset. for our scenario, missing values will be removed in
custName column.
iii) Date format conversion - The datePurchased column is in the format of the object and we will need to convert it to a proper format in the format of
"datetime64"
iv) Pursing dates when loading date: Another approach to applying the correct date format is use of the pursing method correctly when loading the dataset
v) Binning can also be used to handle missing values in numerical data such as age column, where the dataset in the column can be divided into intervals based
on meaningful intervals
```

Ensuring Data Correctness During Collection:

```
i)  Define and enforce data validation rules during data collection to ensure that only valid and consistent data is captured. This involves setting criteria
for acceptable data values, formats, and ranges, for example during data collection using a specific date format or enforcing a date range
ii) Provide structured data entry forms with predefined fields and formats to reduce the likelihood of data entry errors. Include validation checks within the
forms, for example, Create user-friendly interfaces or forms that guide data entry and minimize the chances of inconsistencies.
```

Part 3: Clean the data and save it in an Excel format explaining in detail all the steps taken.

In [7]:
```python
# Change the date format:
df['DatePurchased'] = pd.to_datetime(df['DatePurchased'])
```

In [8]:
```python
# Checking the data type of the date column:
df['DatePurchased'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 100 entries, 0 to 99
Series name: DatePurchased
Non-Null Count  Dtype         
--------------  -----         
93 non-null     datetime64[ns]
dtypes: datetime64[ns](1)
memory usage: 932.0 bytes
```

The date type have been converted to the format datetime64 which is the correct date format.

In [9]:
```python
#Removing Rows with null values in the "custName" column
df.dropna(subset=['custName'], inplace=True)
```

In [10]:
```python
# Remove Rows with missing values in "DatePurchased" column:
df.dropna(subset=['DatePurchased'], inplace=True)
```

In [11]:
```python
# Filling missing values with average in "Price" column
average_price = df['Price'].mean()
df['Price'].fillna(average_price, inplace=True)
```

In [12]:
```python
# Filling null values in the "RatingofProduct" column
median_rating = df['RatingOfProduct'].median()
df['RatingOfProduct'].fillna(median_rating, inplace=True)
```

In [13]:
```python
#Removing Rows with null values in the 'age', 'Product', 'AdvertisingAgency' columns
df.dropna(subset=['Age', 'Product', 'AdvertisingAgency'], inplace=True)
```

In [14]:
```python
# Convert price to 2 decimal places
df['Price'] = df['Price'].round(2)
```

In [15]:
```python
# Convert rating to one decimal point
df['RatingOfProduct'] = df['RatingOfProduct'].round(1)
```

In [16]:
```python
# Checking if the dataset still have null Values:
df.isnull().sum()
```

Out[16]:
```
custID               0
custName             0
Age                  0
Product              0
DatePurchased        0
Price                0
RatingOfProduct      0
AdvertisingAgency    0
dtype: int64
```

In [17]:
```python
# Exporting the cleaned dataset to Excel
df.to_csv('A:\\Data Mining\\Dataset\\dataset.csv', index=False)
print("Export was successful")
```

```
Export was successful
```

In [18]:
```python
# Checking on the final cleaned dataframe
df
```

Out[18]:

	custID	custName	Age	Product	DatePurchased	Price	RatingOfProduct	AdvertisingAgency
0	1	John Doe	30.0	Shirt	2015-01-10	25.99	4.5	Social Media
1	2	Jane Smith	28.0	Shoes	2015-02-15	59.99	3.8	TV
4	5	Michael Brown	32.0	Shirt	2015-05-12	51.40	3.5	Social Media
5	6	Lisa Davis	45.0	Shoes	2015-06-18	89.99	4.8	TV
9	10	Amy Thompson	29.0	Jeans	2015-10-21	37.99	3.9	TV
...	...	...	...	...	...	...	...	...
94	95	Liam Anderson	33.0	Jeans	2014-08-25	49.99	4.0	TV
95	96	Isabella Lee	35.0	Shirt	2014-10-02	18.99	4.4	Newspapers
97	98	Mia Miller	38.0	Jeans	2014-11-16	44.99	4.2	TV
98	99	Charlotte Wilson	36.0	Shirt	2014-12-23	26.99	4.4	Newspapers
99	100	Mason Davis	27.0	Shoes	2013-01-30	79.99	4.8	Social Media

78 rows × 8 columns

In [ ]:
```python
# End
```