Identifying Inconsistencies and Cleaning Dataset

Loading the Customer Dataset

In [5]:
```
import pandas as pd

# Read the dataset (delimited text file)
data = pd.read_csv("C:\dataset\customer_data.txt", delimiter=',')  # Adjust delimiter i
```

Part 1: Identify all inconsistencies in the dataset.

> By running the below checks, one can identify inconsistencies like

- Missing values
- Invalid data types
- Outliers
- Duplicate rows
- nconsistencies in text
- logical relationships across columns.

Check for Missing Values

In [6]:
```
# Identifying for missing values
missing_data = data.isnull().sum()
print(missing_data)
```

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

- After identifying the missing values then dropping and filling missing values
- Dropping rows with missing values.
- Filling missing values with mean/median/mode for numerical columns or a placeholder for categorical columns.

In [7]:
```
# Example: Fill missing values in the 'RatingOfProduct' column with the median
data['RatingOfProduct'].fillna(data['RatingOfProduct'].median(), inplace=True)
```

```
# Example: Fill missing values in 'AdvertisingAgency' with 'Unknown'
data['AdvertisingAgency'].fillna('Unknown', inplace=True)
```

Check for Duplicates

In [8]:
```
# Checking for duplicate rows
duplicates = data.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")

# Removing duplicate rows
data.drop_duplicates(inplace=True)
```

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

Ensure Data Types are Correct

In [9]:
```
# Convert 'DatePurchased' to datetime format
data['DatePurchased'] = pd.to_datetime(data['DatePurchased'], errors='coerce')

# Ensure 'Price' and 'RatingOfProduct' are numerical
data['Price'] = pd.to_numeric(data['Price'], errors='coerce')
data['RatingOfProduct'] = pd.to_numeric(data['RatingOfProduct'], errors='coerce')

# Ensure 'Age' is integer
data['Age'] = pd.to_numeric(data['Age'], downcast='integer', errors='coerce')
```

Handle Outliers

In [10]:
```
# Identifying outliers in 'Price'
import numpy as np

price_outliers = data[(data['Price'] > data['Price'].mean() + 3 * data['Price'].std()) |
                      (data['Price'] < data['Price'].mean() - 3 * data['Price'].std())]

print(price_outliers)
```

```
Empty DataFrame
Columns: [custID, custName, Age, Product, DatePurchased, Price, RatingOfProduct, Adverti
singAgency]
Index: []
```

Standardize the Format of Text Columns

In [11]:
```
# Remove leading/trailing spaces and standardize text columns to lower case
data['custName'] = data['custName'].str.strip().str.title()
data['Product'] = data['Product'].str.strip().str.title()
data['AdvertisingAgency'] = data['AdvertisingAgency'].str.strip().str.title()
```

Handle Invalid Values

In [12]:
```
# Remove leading/trailing spaces and standardize text columns to lower case
data['custName'] = data['custName'].str.strip().str.title()
data['Product'] = data['Product'].str.strip().str.title()
data['AdvertisingAgency'] = data['AdvertisingAgency'].str.strip().str.title()
```

Final Data Check

In [13]:
```
# Check for any remaining issues
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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     datetime64[ns]
 5   Price              96 non-null     float64
 6   RatingOfProduct    100 non-null    float64
 7   AdvertisingAgency  100 non-null    object
dtypes: datetime64[ns](1), float64(3), int64(1), object(3)
memory usage: 7.0+ KB
None
```

Export the Cleaned Dataset

In [14]:
```
# Save cleaned data to a new CSV file
data.to_csv('cleaned_customer_data.csv', index=False)
```

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 to Solve Data Inconsistencies

Handling Missing Values

- Imputation: Replace missing values with calculated estimates like the mean, median, or mode, or by using advanced methods like machine learning.
- Removal: In cases where too many values are missing, remove the rows or columns altogether.

Data Collection Measures

- Use real-time validation to ensure mandatory fields are filled, and provide sensible default values where applicable.

Correcting Inconsistent Data Types

- Convert columns to appropriate data types (e.g., numeric or date) to ensure proper analysis.
- Handle errors by flagging or removing invalid data entries.

Data Collection Measures

- Implement formatting restrictions for input fields, and validate data types as part of the collection process.

Removing Duplicates

- Identify and remove duplicate entries to ensure accuracy in data analysis.
- Use unique identifiers to prevent duplicate data from being recorded.

Data Collection Measures

- Assign unique IDs to records and implement real-time duplicate detection to avoid redundant data entry.

Correcting Invalid or Out-of-Range Values

- Filter out or flag values that fall outside acceptable ranges, such as outliers in numerical fields.
- Use range checks to maintain consistency.

Data Collection Measures

- Set constraints for data entry (e.g., limiting numeric values to valid ranges) and provide real-time alerts for invalid inputs.

Standardizing Inconsistent Text Entries

- Normalize text by ensuring consistent formats (e.g., case consistency) and remove extra spaces.
- Use dictionary mapping to standardize common variations in text entries.

Data Collection Measures

- Use dropdowns or selection lists instead of free-form text fields to avoid variations, and apply automatic formatting for consistent data entry.
- Ensuring Correct Data Capture

To ensure data is accurately captured during collection

- Automated Data Validation: Ensure all fields follow the correct format and rules during data entry.
- Pre-Defined Input Formats: Use tools like dropdowns, date pickers, and masked fields to limit errors.
- Unique Identifiers: Assign unique IDs for each entry to prevent duplicates.
- Training: Provide proper training for data entry personnel to follow standard protocols.
- Auditing: Regularly audit and monitor the data collection process to catch and correct errors early.

These techniques and safeguards will address inconsistencies and help ensure reliable, high-quality data.

Part 4: Discuss any FIVE potential challenges and FIVE opportunities that a company using this dataset will potentially experience.

Challenges a Company Using This Dataset May Experience

Data Quality Issues: Poor data quality, such as missing or incorrect values, duplicates, and inconsistencies, can limit the accuracy of insights drawn from the dataset. For instance, if customer names or product details are incomplete or duplicated, it could lead to flawed analyses and decisions.

Data Integration Difficulties: Integrating this dataset with other systems (e.g., customer relationship management (CRM) or enterprise resource planning (ERP)) might be challenging, especially if the formats are incompatible. Data inconsistencies such as varying formats for dates or product names can make it difficult to merge datasets effectively.

Privacy and Compliance Risks: Storing and using customer data, particularly sensitive information like purchasing history and demographic details, introduces privacy concerns. If not properly managed, it could lead to violations of regulations like GDPR or CCPA, resulting in fines or reputational damage.

Data Overload and Complexity: The volume of data generated can be overwhelming, particularly for a company that is not equipped with the right tools for data processing and analysis. Without proper filtering and data governance, it becomes challenging to extract actionable insights from the data.

Maintaining Data Freshness: If the dataset is not regularly updated, the company could be working with outdated information. Customer behaviors and product performance can change quickly, so old data may lead to poor decisions if not refreshed.

Opportunities a Company Using This Dataset May Experience

Improved Customer Segmentation: With detailed purchasing history, age, and product ratings, the company can segment its customer base more accurately. This allows for targeted marketing campaigns, personalized offers, and better understanding of customer preferences.

Enhanced Product Development and Feedback: The product ratings in the dataset provide direct customer feedback, which can be used to improve existing products or guide the development of new ones. High-rated products can be promoted, while low-rated ones may signal the need for improvements.

Optimized Marketing Strategies: The presence of advertising agency information along with product purchases provides an opportunity to evaluate the effectiveness of different advertising campaigns. The company can track which agencies or campaigns lead to better sales and focus resources accordingly.

Data-Driven Decision Making: By analyzing the dataset, the company can identify trends and patterns that can inform decision-making. For example, they may find that certain products sell better to specific age groups, enabling more precise inventory management and marketing efforts.

Personalization and Customer Retention: With customer history available, the company can offer personalized recommendations based on past purchases. This enhances customer satisfaction and

increases loyalty, as personalized experiences are often linked to better customer retention.

In [ ]: