01-data-cleaning

January 16, 2026

IMPORT LIBRARIES

[3]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

pd.set_option('display.max_columns', None)
```

LOAD DATASET

[5]:
```
df = pd.read_csv(
    "D:\\decision-intelligence-project\\Data\\online_retail.csv",
    encoding="ISO-8859-1"
)
print("Loaded")
```

```
Loaded
```

BASIC DATA

[7]:
```
df.head()
df.info()
df.shape
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    541909 non-null  object 
 1   StockCode    541909 non-null  object 
 2   Description  540455 non-null  object 
 3   Quantity     541909 non-null  int64  
 4   InvoiceDate  541909 non-null  object 
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object 
```

```
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

[7]:

	Quantity	UnitPrice	CustomerID
count	541909.000000	541909.000000	406829.000000
mean	9.552250	4.611114	15287.690570
std	218.081158	96.759853	1713.600303
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13953.000000
50%	3.000000	2.080000	15152.000000
75%	10.000000	4.130000	16791.000000
max	80995.000000	38970.000000	18287.000000

DATA CLEANING — INVALID TRANSACTIONS

[8]:
```
df.dropna(subset=['CustomerID'])
df=df[df["Quantity"]>0]
df=df[df["UnitPrice"]>0]
```

DATA HANDLING AND TIME FEATURES

[9]:
```
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
df['Year'] = df['InvoiceDate'].dt.year
df['Month'] = df['InvoiceDate'].dt.month
df['Quarter'] = df['InvoiceDate'].dt.to_period('Q').astype(str)
```

REVENUE CALCULATION

[10]:
```
df['Revenue'] = df['Quantity'] * df['UnitPrice']
df[['Quantity', 'UnitPrice', 'Revenue']].head()
```

[10]:

	Quantity	UnitPrice	Revenue
0	6	2.55	15.30
1	6	3.39	20.34
2	8	2.75	22.00
3	6	3.39	20.34
4	6	3.39	20.34

BUSINESS ASSUMPTION 1 — CHANNEL SIMULATION ###### (assume 70% online, 30% offline)

[11]:
```
np.random.seed(42)
df['Channel'] = np.random.choice(
    ['Online', 'Offline'],
    size=len(df),
    p=[0.7, 0.3]
)
```

#BUSINESS ASSUMPTION 2 — PROMOTION FLAG ###### (Promotion active 25% time)

[12]:
```
df['Promotion_Flag'] = np.random.choice(
    [0, 1],
    size=len(df),
    p=[0.75, 0.25]
)
```

BUSINESS ASSUMPTION 3 — PROFIT MARGIN ###### (Margin between 15% and 40%)

[13]:
```
df['Profit_Margin'] = np.random.uniform(0.15, 0.40, size=len(df))
df['Profit'] = df['Revenue'] * df['Profit_Margin']
df[['Revenue', 'Profit_Margin', 'Profit']].head()
```

[13]:

	Revenue	Profit_Margin	Profit
0	15.30	0.235824	3.608111
1	20.34	0.355912	7.239256
2	22.00	0.166214	3.656700
3	20.34	0.218856	4.451527
4	20.34	0.310762	6.320903

CUSTOMER-LEVEL DATASET (OUTPUT 1)

[14]:
```
customer_df = (
    df.groupby('CustomerID')
    .agg(
        Total_Revenue=('Revenue', 'sum'),
        Total_Profit=('Profit', 'sum'),
        Avg_Profit_Margin=('Profit_Margin', 'mean'),
        Num_Transactions=('InvoiceNo', 'nunique'),
        First_Transaction_Date=('InvoiceDate', 'min'),
        Last_Transaction_Date=('InvoiceDate', 'max')
    )
    .reset_index()
)

customer_df.head()
```

[14]:

	CustomerID	Total_Revenue	Total_Profit	Avg_Profit_Margin	Num_Transactions	First_Transaction_Date	Last_Transaction_Date
0	12346.0	77183.60	11666.373986	0.151151	1	2011-01-18 10:01:00	2011-01-18 10:01:00
1	12347.0	4310.00	1170.503745	0.276163	7	2010-12-07 14:57:00	2011-12-07 15:52:00
2	12348.0	1797.24	505.253665	0.273754	4	2010-12-16 19:09:00	2011-09-25 13:13:00
3	12349.0	1757.55	526.463936	0.284261	1	2011-11-21 09:51:00	2011-11-21 09:51:00
4	12350.0	334.40	91.762098	0.274248			

```
4                  1    2011-02-02 16:01:00   2011-02-02 16:01:00
```

PRODUCT-LEVEL DATASET (OUTPUT 2)

[15]:
```
product_df = (
    df.groupby('StockCode')
    .agg(
        Total_Revenue=('Revenue', 'sum'),
        Total_Profit=('Profit', 'sum'),
        Avg_Profit_Margin=('Profit_Margin', 'mean'),
        Total_Quantity_Sold=('Quantity', 'sum'),
        Num_Transactions=('InvoiceNo', 'nunique')
    )
    .reset_index()
)

product_df.head()
```

[15]:

	StockCode	Total_Revenue	Total_Profit	Avg_Profit_Margin	\
0	10002	759.89	203.059914	0.268673	
1	10080	119.09	33.786145	0.279529	
2	10120	40.53	11.031979	0.261472	
3	10123C	3.25	0.987529	0.266197	
4	10124A	6.72	1.867102	0.266238	

	Total_Quantity_Sold	Num_Transactions
0	860	71
1	303	22
2	193	29
3	5	3
4	16	5

TIME-SERIES DATASET (OUTPUT 3)

[16]:
```
time_series_df = (
    df.groupby(['Year', 'Month'])
    .agg(
        Monthly_Revenue=('Revenue', 'sum'),
        Monthly_Profit=('Profit', 'sum'),
        Avg_Profit_Margin=('Profit_Margin', 'mean'),
        Num_Transactions=('InvoiceNo', 'nunique')
    )
    .reset_index()
)
```

SAVE CLEAN DATA

[18]:
```
df.to_csv("D:/decision-intelligence-project/Data/Processed_Data/
  ↪clean_transactions.csv", index=False)
customer_df.to_csv("D:/decision-intelligence-project/Data/Processed_Data/
  ↪customer_level_data.csv", index=False)
product_df.to_csv("D:/decision-intelligence-project/Data/Processed_Data/
  ↪product_level_data.csv", index=False)
time_series_df.to_csv("D:/decision-intelligence-project/Data/Processed_Data/
  ↪time_series_data.csv", index=False)
```