Cafe Sales Data Cleaning

Cleaning the `dirty_cafe_sales.csv` Dataset

This notebook cleans the `dirty_cafe_sales.csv` dataset, which contains **10,000 rows** of sales transactions from a café.

Dataset Overview

The dataset includes the following **eight columns**:

1. `Transaction ID`
2. `Item`
3. `Quantity`
4. `Price Per Unit`
5. `Total Spent`
6. `Payment Method`
7. `Location`
8. `Transaction Date`

Column Details

🧾 `Transaction ID`

- No missing values.
- Contains 10,000 unique identifiers.
- Stored as `object` datatype.
- No consistent pattern observed (e.g., no embedded date/location info).

🗓️ `Transaction Date`

- Converted to `datetime` datatype.
- 159 missing entries.
- No consistent interval or seasonal pattern, so missing values **not imputed**.

🏷️ Categorical Columns

- `Item`, `Payment Method`, and `Location`
- All stored as `object` datatype.
- Each contains missing values.
- Due to lack of reliable imputation logic, **missing values replaced with `null`**.

🔢 Numeric Columns

- `Quantity`, `Price Per Unit`, and `Total Spent`
- All contain missing values.
- Imputation performed using the following formulas:
  - `Total Spent = Quantity × Price Per Unit`
  - `Quantity = Total Spent ÷ Price Per Unit`
  - `Price Per Unit = Total Spent ÷ Quantity`
- After imputation: **9,942 rows retained** with complete numeric data.

Exported Cleaned Files

Two cleaned versions of the dataset are saved:

- `cafe_sales_no_nulls.csv`
  - Drops **all** rows with any null values.
  - Retains only **36%** of the original data.
- `cafe_sales_no_numeric_nulls.csv`
  - Drops rows with **numeric** null values only.
  - Retains **99.42%** of the original data.

Load Libraries, Dataset, and Do Initial Assessment

In [109…
```
# Import libraries
import pandas as pd
import numpy as np
```

In [110…
```
# Load the datset
df = pd.read_csv(r'C:\Users\Joseph Finch\OneDrive - Tuyen Data\Portfolio Projects\d

df.head()
```

Out[110…

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transaction Date
0	TXN_1961373	Coffee	2	2.0	4.0	Credit Card	Takeaway	2023-09-08
1	TXN_4977031	Cake	4	3.0	12.0	Cash	In-store	2023-05-16
2	TXN_4271903	Cookie	4	1.0	ERROR	Credit Card	In-store	2023-07-19
3	TXN_7034554	Salad	2	5.0	10.0	UNKNOWN	UNKNOWN	2023-04-27
4	TXN_3160411	Coffee	2	2.0	4.0	Digital Wallet	In-store	2023-06-11

In [111…
```
df.tail()
```

Out[111…

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transaction Date
9995	TXN_7672686	Coffee	2	2.0	4.0	NaN	UNKNOWN	2023-08-30
9996	TXN_9659401	NaN	3	NaN	3.0	Digital Wallet	NaN	2023-06-02
9997	TXN_5255387	Coffee	4	2.0	8.0	Digital Wallet	NaN	2023-03-02
9998	TXN_7695629	Cookie	3	NaN	3.0	Digital Wallet	NaN	2023-12-02
9999	TXN_6170729	Sandwich	3	4.0	12.0	Cash	In-store	2023-11-07

In [112…
```
df.shape
```

Out[112…
```
(10000, 8)
```

In [113…
```
df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Transaction ID    10000 non-null  object
 1   Item              9667 non-null   object
 2   Quantity          9862 non-null   object
 3   Price Per Unit    9821 non-null   object
 4   Total Spent       9827 non-null   object
 5   Payment Method    7421 non-null   object
 6   Location          6735 non-null   object
 7   Transaction Date  9841 non-null   object
dtypes: object(8)
memory usage: 625.1+ KB
```

Out[113...

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transaction Date
count	10000	9667	9862	9821	9827	7421	6735	9841
unique	10000	10	7	8	19	5	4	367
top	TXN_9226047	Juice	5	3.0	6.0	Digital Wallet	Takeaway	UNKNOWN
freq	1	1171	2013	2429	979	2291	3022	159

In [114...

```
df.columns
```

Out[114...

```
Index(['Transaction ID', 'Item', 'Quantity', 'Price Per Unit', 'Total Spent',
       'Payment Method', 'Location', 'Transaction Date'],
      dtype='object')
```

Analysis of Categorical, Numeric, Identification, and Date Columns

The columns Item, Payment Method, and Location are categorical columns. Quantity, Price Per Unit, and Total Spent are numeric columns. Quantity should be an integer, while Price Per Unit and Total Spent can be decimal numbers with two decimal places. The Transaction ID column should remain as an object and be used for identifying records. The Transaction Date column will be converted to a datetime datatype.

In [115...

```
columns_to_use = df.columns[1:7]

for x in columns_to_use:
    print("="*40)
    print(f"Column: {x}")
    print(f"Number of unique values (incl. NaN): {len(df[x].unique())}")
    print(f"Number of distinct values (excl. NaN): {df[x].nunique()}")
    print("List of values:")
    print(df[x].unique())
    print()
```

```
========================================
Column: Item
Number of unique values (incl. NaN): 11
Number of distinct values (excl. NaN): 10
List of values:
['Coffee' 'Cake' 'Cookie' 'Salad' 'Smoothie' 'UNKNOWN' 'Sandwich' nan
 'ERROR' 'Juice' 'Tea']

========================================
Column: Quantity
Number of unique values (incl. NaN): 8
Number of distinct values (excl. NaN): 7
List of values:
['2' '4' '5' '3' '1' 'ERROR' 'UNKNOWN' nan]

========================================
Column: Price Per Unit
Number of unique values (incl. NaN): 9
Number of distinct values (excl. NaN): 8
List of values:
['2.0' '3.0' '1.0' '5.0' '4.0' '1.5' nan 'ERROR' 'UNKNOWN']

========================================
Column: Total Spent
Number of unique values (incl. NaN): 20
Number of distinct values (excl. NaN): 19
List of values:
['4.0' '12.0' 'ERROR' '10.0' '20.0' '9.0' '16.0' '15.0' '25.0' '8.0' '5.0'
 '3.0' '6.0' nan 'UNKNOWN' '2.0' '1.0' '7.5' '4.5' '1.5']

========================================
Column: Payment Method
Number of unique values (incl. NaN): 6
Number of distinct values (excl. NaN): 5
List of values:
['Credit Card' 'Cash' 'UNKNOWN' 'Digital Wallet' 'ERROR' nan]

========================================
Column: Location
Number of unique values (incl. NaN): 5
Number of distinct values (excl. NaN): 4
List of values:
['Takeaway' 'In-store' 'UNKNOWN' nan 'ERROR']
```

In [116…

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

Out[116…

```
Transaction ID         0
Item                 333
Quantity             138
Price Per Unit       179
Total Spent          173
Payment Method      2579
Location            3265
Transaction Date     159
dtype: int64
```

Checking Imputing Options in Categorical Data

In order to determine if missing transaction IDs or dates can be imputed, we need to check for patterns in the data. Unfortunately, there appears to be no pattern. Transaction IDs and dates can not be imputed.

In [117…
```
is_sorted = df["Transaction ID"].is_monotonic_increasing
print("Is it lexicographically increasing? ", is_sorted)
```

```
Is it lexicographically increasing?  False
```

In [118…
```
df["Transaction ID"] = (
    df["Transaction ID"]
    .str.replace("TXN_", "", regex=False)
    .astype(int)
)
```

In [119…
```
print(df["Transaction ID"].is_monotonic_increasing)
```

```
False
```

In [120…
```
df["Transaction ID"].head()
```

Out[120…
```
0    1961373
1    4977031
2    4271903
3    7034554
4    3160411
Name: Transaction ID, dtype: int64
```

In [121…
```
dftemp = pd.read_csv(r'C:\Users\Joseph Finch\OneDrive - Tuyen Data\Portfolio Projec
df["Transaction ID"] = dftemp["Transaction ID"]

df["Transaction ID"].head()
```

Out[121…
```
0    TXN_1961373
1    TXN_4977031
2    TXN_4271903
3    TXN_7034554
4    TXN_3160411
Name: Transaction ID, dtype: object
```

In [122…
```
df["Transaction Date"] = pd.to_datetime(df["Transaction Date"], errors='coerce')
df["Transaction Date"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 10000 entries, 0 to 9999
Series name: Transaction Date
Non-Null Count  Dtype
--------------  -----
9540 non-null   datetime64[ns]
dtypes: datetime64[ns](1)
memory usage: 78.3 KB
```

In [123…
```
is_sorted = df["Transaction Date"].is_monotonic_increasing
print("Is it lexicographically increasing? ", is_sorted)
```

```
Is it lexicographically increasing?  False
```

In [124…
```
is_sorted = df["Transaction Date"].is_monotonic_decreasing
print("Is it lexicographically decreasing? ", is_sorted)
```

```
Is it lexicographically decreasing?  False
```

Imputing Data In Numeric Columns

The numeric columns need to be converted to numbers. They have a relatinship, so they can be used to impute values. Assuming that Total spent = Quantity * Price Per Unit we can calculate the missing values. From those 3 columns there are only 58 missing 58 which is a loss ration of 0.58%.

In [125…
```
df["Price Per Unit"] = pd.to_numeric(df["Price Per Unit"], errors='coerce')
df["Total Spent"] = pd.to_numeric(df["Total Spent"], errors='coerce')
df["Quantity"] = pd.to_numeric(df["Quantity"], errors='coerce')
```

In [126…
```
# Avoid division by zero
df.replace({0: np.nan}, inplace=True)

# Fill Quantity
df.loc[
    df["Quantity"].isnull() & df["Total Spent"].notnull() & df["Price Per Unit"].no
    "Quantity"
] = df["Total Spent"] / df["Price Per Unit"]

# Fill Price per Unit
df.loc[
    df["Price Per Unit"].isnull() & df["Total Spent"].notnull() & df["Quantity"].no
    "Price Per Unit"
] = df["Total Spent"] / df["Quantity"]

# Fill Total Spent
df.loc[
    df["Total Spent"].isnull() & df["Price Per Unit"].notnull() & df["Quantity"].no
    "Total Spent"
] = df["Price Per Unit"] * df["Quantity"]
```

In [127…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Transaction ID    10000 non-null  object
 1   Item              9667 non-null   object
 2   Quantity          9962 non-null   float64
 3   Price Per Unit    9962 non-null   float64
 4   Total Spent       9960 non-null   float64
 5   Payment Method    7421 non-null   object
 6   Location          6735 non-null   object
 7   Transaction Date  9540 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(3), object(4)
memory usage: 625.1+ KB
```

In [128…

```
rows_missing_numbers = df[df[["Quantity", "Price Per Unit", "Total Spent"]].isnull(
print (rows_missing_numbers[0])
print("Loss Ratio: " + str(round(rows_missing_numbers[0] / df.shape[0] * 100, 2)) +
```

```
58
Loss Ratio: 0.58%
```

In [129…

```
# Define pattern: case-insensitive match for unknown, error, blank
bad_pattern = r'(?i)^(unknown|error|\s*)$'

# Clean multiple columns
for col in ["Item", "Payment Method"]:
    df[col] = df[col].str.strip().replace(bad_pattern, np.nan, regex=True)
```

In [130…

```
# 1. Strip spaces
df["Payment Method"] = df["Payment Method"].str.strip()

# 2. Replace bad values with NaN
df["Payment Method"] = df["Payment Method"].str.strip().replace(
    r'(?i)^(unknown|error|n/a|missing|\s*)$',
    np.nan,
    regex=True
)
```

In [131…

```
# 1. Strip spaces
df["Location"] = df["Location"].str.strip()

# 2. Replace bad values with NaN
df["Location"] = df["Location"].str.strip().replace(
    r'(?i)^(unknown|error|n/a|missing|\s*)$',
    np.nan,
    regex=True
)
```

In [132…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype         
---  ------            --------------  -----         
 0   Transaction ID    10000 non-null  object        
 1   Item              9031 non-null   object        
 2   Quantity          9962 non-null   float64       
 3   Price Per Unit    9962 non-null   float64       
 4   Total Spent       9960 non-null   float64       
 5   Payment Method    6822 non-null   object        
 6   Location          6039 non-null   object        
 7   Transaction Date  9540 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(3), object(4)
memory usage: 625.1+ KB
```

In [133…

```python
columns_to_use = df.columns[1:7]

for x in columns_to_use:
    print("="*40)
    print(f"Column: {x}")
    print(f"Number of unique values (incl. NaN): {len(df[x].unique())}")
    print(f"Number of distinct values (excl. NaN): {df[x].nunique()}")
    print("List of values:")
    print(df[x].unique())
    print()
```

```
========================================
Column: Item
Number of unique values (incl. NaN): 9
Number of distinct values (excl. NaN): 8
List of values:
['Coffee' 'Cake' 'Cookie' 'Salad' 'Smoothie' nan 'Sandwich' 'Juice' 'Tea']

========================================
Column: Quantity
Number of unique values (incl. NaN): 6
Number of distinct values (excl. NaN): 5
List of values:
[ 2.  4.  5.  3.  1. nan]

========================================
Column: Price Per Unit
Number of unique values (incl. NaN): 7
Number of distinct values (excl. NaN): 6
List of values:
[2.  3.  1.  5.  4.  1.5 nan]

========================================
Column: Total Spent
Number of unique values (incl. NaN): 18
Number of distinct values (excl. NaN): 17
List of values:
[ 4.  12.  10.  20.   9.  16.  15.  25.   8.   5.   3.   6.   2.   nan
  1.   7.5  4.5  1.5]

========================================
Column: Payment Method
Number of unique values (incl. NaN): 4
Number of distinct values (excl. NaN): 3
List of values:
['Credit Card' 'Cash' nan 'Digital Wallet']

========================================
Column: Location
Number of unique values (incl. NaN): 3
Number of distinct values (excl. NaN): 2
List of values:
['Takeaway' 'In-store' nan]
```

In [134…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype         
---  ------            --------------  -----         
 0   Transaction ID    10000 non-null  object        
 1   Item              9031 non-null   object        
 2   Quantity          9962 non-null   float64       
 3   Price Per Unit    9962 non-null   float64       
 4   Total Spent       9960 non-null   float64       
 5   Payment Method    6822 non-null   object        
 6   Location          6039 non-null   object        
 7   Transaction Date  9540 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(3), object(4)
memory usage: 625.1+ KB
```

In [135…
```
df_no_null = df.dropna()
df_no_null.to_csv('cafe_sales_no_nulls.csv', index=False)
```

In [136…
```
print ("If you drop all null values you will be left with " + str(df_no_null.shape)
print("Percentage of data kept: " + str(round(df_no_null.shape[0] / df.shape[0] * 1
```

```
If you drop all null values you will be left with (3556, 8)
Percentage of data kept: 35.56%
```

In [137…
```
df_no_null_numbers = df.dropna(subset=["Quantity", "Price Per Unit", "Total Spent"]
df_no_null_numbers.to_csv('cafe_sales_no_numeric_nulls.csv', index=False)
```

In [138…
```
print ("If you drop null values only in the numeric columns you will get a shape of
print("Percentage of data kept: " + str(round(df_no_null_numbers.shape[0] / df.shap
```

```
If you drop null values only in the numeric columns you will get a shape of (9942,
8)
Percentage of data kept: 99.42%
```