

```
#Alaa Waleed Ali Humaid
# ID:261002982
# Cafe Sales dataset

from google.colab import files
import pandas as pd
import numpy as np
import io
df = pd.read_csv('/dirty_cafe_sales.csv')
df.head()
```

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location
0	TXN_1961373	Coffee	2	2	4	Credit Card	Takeaway
1	TXN_4977031	Cake	4	3	12	Cash	In-store
2	TXN_4271903	Cookie	4	1	ERROR	Credit Card	In-store
3	TXN_7034554	Salad	2	5	10	UNKNOWN	UNKNOWN
4	TXN_3160411	Coffee	2	2	4	Digital Wallet	In-store

```
# Check for missing values
df.isnull().sum()
```

	0
Transaction ID	0
Item	333
Quantity	138
Price Per Unit	179
Total Spent	173
Payment Method	2579
Location	3265

dtype: int64

```
# show the columns with the large missing values
missing_percentage = df.isnull().mean() * 100
# Filter columns with more than 30% (or 50%) missing values
columns_with_missing_30 = missing_percentage[missing_percentage > 30]
# Display the columns with more than 30% missing values
print("Columns with more than 30% missing values:")
print(columns_with_missing_30)
```

```
Columns with more than 30% missing values:
Location    32.65
dtype: float64
```

```
# Drop the column with the most missing values
missing_percentage = df.isnull().mean() * 100
threshold = 30
# Filter columns with more than the threshold percentage of missing values
columns_to_drop = missing_percentage[missing_percentage >
threshold].index
# Drop the columns
df = df.drop(columns=columns_to_drop)
# Display the resulting dataframe after removing columns with too many missing values
print(f"Columns dropped: {list(columns_to_drop)}")
```

```
df.head()
```

Columns dropped: ['Location']

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method
0	TXN_1961373	Coffee	2	2	4	Credit Card
1	TXN_4977031	Cake	4	3	12	Cash
2	TXN_4271903	Cookie	4	1	ERROR	Credit Card
3	TXN_7034554	Salad	2	5	10	UNKNOWN
4	TXN_3160411	Coffee	2	2	4	Digital Wallet

```
# Replace error, unknown, and empty strings with NaN
df.replace(['ERROR', 'UNKNOWN', ''], np.nan, inplace=True)
```

```
df.head()
```

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method
0	TXN_1961373	Coffee	2	2	4	Credit Card
1	TXN_4977031	Cake	4	3	12	Cash
2	TXN_4271903	Cookie	4	1	NaN	Credit Card
3	TXN_7034554	Salad	2	5	10	NaN
4	TXN_3160411	Coffee	2	2	4	Digital Wallet

```
# handel numrical null values
cols = ['Quantity', 'Price Per Unit', 'Total Spent']

for col in cols:
    df[col] = pd.to_numeric(df[col], errors='coerce')
    df[col] = df[col].fillna(df[col].median())
```

```
df.head()
```

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method
0	TXN_1961373	Coffee	2.0	2.0	4.0	Credit Card
1	TXN_4977031	Cake	4.0	3.0	12.0	Cash
2	TXN_4271903	Cookie	4.0	1.0	8.0	Credit Card
3	TXN_7034554	Salad	2.0	5.0	10.0	NaN
4	TXN_3160411	Coffee	2.0	2.0	4.0	Digital Wallet

```
# handel categorical null values
cols = ['Transaction ID', 'Item', 'Payment Method']

for col in cols:
    df[col] = df[col].fillna(df[col].mode()[0])
```

```
df.head()
```

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method
0	TXN_1961373	Coffee	2.0	2.0	4.0	Credit Card
1	TXN_4977031	Cake	4.0	3.0	12.0	Cash
2	TXN_4271903	Cookie	4.0	1.0	8.0	Credit Card
3	TXN_7034554	Salad	2.0	5.0	10.0	Digital Wallet
4	TXN_3160411	Coffee	2.0	2.0	4.0	Digital Wallet

```
# handle the outlier
data = df.copy()

def detect_outliers_iqr(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
```

```
return (column < lower_bound) | (column > upper_bound)

# Select only numeric columns
numeric_cols = data.select_dtypes(include='number').columns

# Apply IQR only to numeric columns
for col in numeric_cols:
    data[f'{col}_Outlier'] = detect_outliers_iqr(data[col])

# Show rows containing any outlier
outliers_only = data[data.filter(like='_Outlier').any(axis=1)]

outliers_only.head(10)
```

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Quantity_Outlier	Price Per Unit_Outlier	Total Spent_Outlier
10	TXN_2548360	Salad	5.0	5.0	25.0	Cash	False	False	True
51	TXN_6342161	Salad	5.0	5.0	25.0	Digital Wallet	False	False	True
52	TXN_8914892	Juice	5.0	5.0	25.0	Digital Wallet	False	False	True
96	TXN_5220895	Salad	5.0	5.0	25.0	Cash	False	False	True
100	TXN_9517146	Juice	5.0	5.0	25.0	Cash	False	False	True
150	TXN_8687151	Salad	5.0	5.0	25.0	Cash	False	False	True
157	TXN_4283157	Salad	5.0	5.0	25.0	Digital Wallet	False	False	True
477	TXN_4892055	Salad	5.0	5.0	25.0	Digital	False	False	True

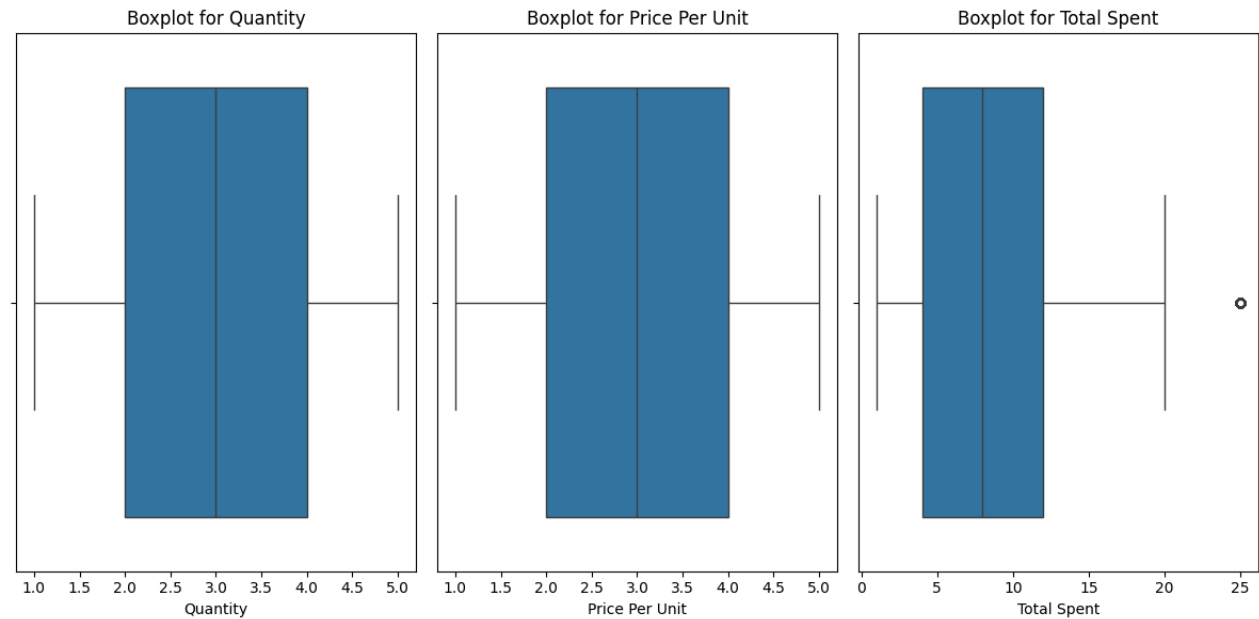
```
# Perform data visualization.
import matplotlib.pyplot as plt
import seaborn as sns

numeric_cols = df.select_dtypes(include='number').columns

plt.figure(figsize=(12,6))

for i, col in enumerate(numeric_cols[:3]):
    plt.subplot(1,3,i+1)
    sns.boxplot(x=df[col])
    plt.title(f"Boxplot for {col}")

plt.tight_layout()
plt.show()
```



```
#Replace outliers with NaN.
data=df.copy()
def replace_outliers_with_nan(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Replace outliers with NaN
    return column.where((column >= lower_bound) & (column <= upper_bound), np.nan)

# Select only numeric columns
numeric_cols = data.select_dtypes(include='number').columns

# Replace outliers with NaN for numeric features
for col in numeric_cols:
    data[col] = replace_outliers_with_nan(data[col])

null_counts = data.isnull().sum()
print("Sum of null values in each column:")
print(null_counts)
```

```
Sum of null values in each column:
Transaction ID    0
Item              0
Quantity          0
Price Per Unit    0
Total Spent       259
Payment Method    0
dtype: int64
```

```
# Handling missing values using K-Nearest Neighbors (KNN) Imputation.
from sklearn.impute import KNNImputer
import pandas as pd
import numpy as np

# Select only numeric columns from the DataFrame for KNN imputation
numeric_cols = data.select_dtypes(include=np.number).columns.tolist()

# Create a temporary DataFrame with only numeric columns
data_numeric = data[numeric_cols].copy()

# Initialize KNNImputer
imputer = KNNImputer(n_neighbors=2)

# Apply KNNImputer to the numeric data
data_imputed_knn_numeric = pd.DataFrame(imputer.fit_transform(data_numeric),
                                         columns=numeric_cols,
                                         index=data_numeric.index)

# Update the original 'data' DataFrame with the imputed numeric values
data[numeric_cols] = data_imputed_knn_numeric

# Check for null values in the updated DataFrame
data_imputed_knn = data # Keep the variable name consistent for output
data_imputed_knn.isnull().sum()
```

```
0
Transaction ID  0
Item           0
Quantity       0
Price Per Unit  0
Total Spent    0
Payment Method  0
dtype: int64
```

