Assignment

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Descriptive Analytics for Numerical Columns

```
df = pd.read_csv('sales_data_with_discounts.csv')
df.head()
```

```
         Date       Day  SKU City  Volume       BU  Brand       Model
\
0  01-04-2021  Thursday  M01    C      15  Mobiles  RealU       RU-10

1  01-04-2021  Thursday  M02    C      10  Mobiles  RealU   RU-9 Plus

2  01-04-2021  Thursday  M03    C       7  Mobiles   YouM       YM-99

3  01-04-2021  Thursday  M04    C       6  Mobiles   YouM  YM-99 Plus

4  01-04-2021  Thursday  M05    C       3  Mobiles   YouM      YM-98


   Avg Price  Total Sales Value  Discount Rate (%)  Discount Amount  \
0      12100             181500          11.654820     21153.498820
1      10100             101000          11.560498     11676.102961
2      16100             112700           9.456886     10657.910157
3      20100             120600           6.935385      8364.074702
4       8100              24300          17.995663      4372.946230

   Net Sales Value
0    160346.501180
1     89323.897039
2    102042.089843
3    112235.925298
4     19927.053770
```

```
# Identify numerical columns
numerical_cols = df.select_dtypes(include=['number']).columns
numerical_cols
```

```
Index(['Volume', 'Avg Price', 'Total Sales Value', 'Discount Rate
(%)',
       'Discount Amount', 'Net Sales Value'],
      dtype='object')
```

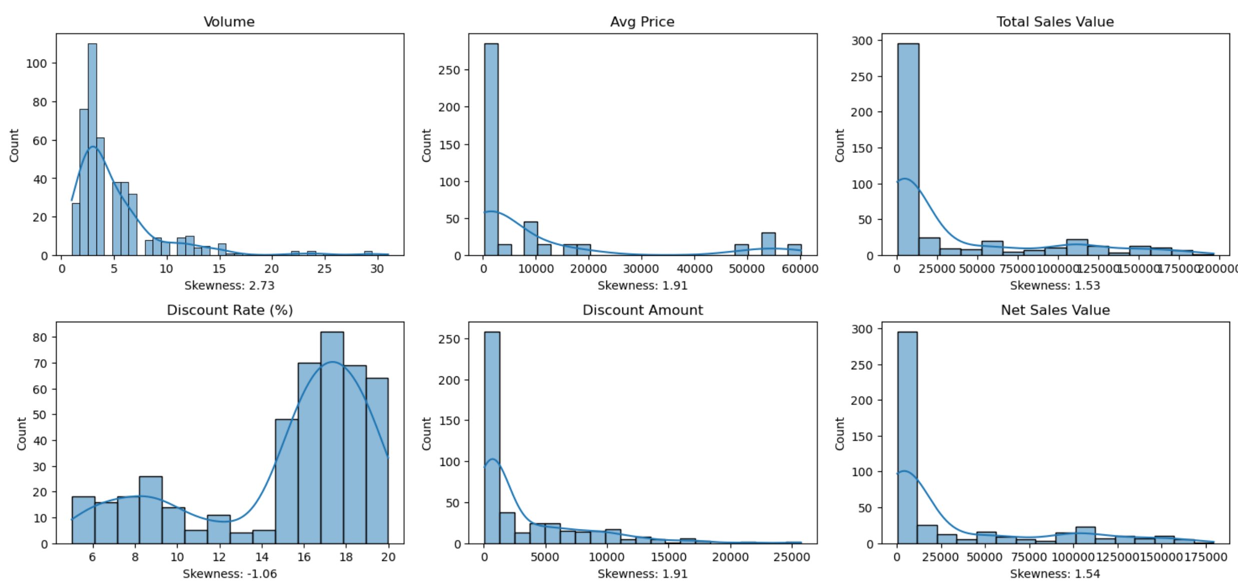
```
# Calculate basic statistical measures
stats = df[numerical_cols].agg(['mean', 'median', lambda x:
x.mode().iloc[0], 'std']).transpose()
stats.columns = ['Mean', 'Median', 'Mode', 'Standard Deviation']

# Provide interpretation
print("Basic Statistical Measures for Numerical Columns:")
print(stats)
```

```
Basic Statistical Measures for Numerical Columns:
                           Mean       Median          Mode  Standard
Deviation
Volume                 5.066667     4.000000      3.000000
4.231602
Avg Price          10453.433333  1450.000000    400.000000
18079.904840
Total Sales Value  33812.835556  5700.000000  24300.000000
50535.074173
Discount Rate (%)     15.155242    16.577766      5.007822
4.220602
Discount Amount     3346.499424   988.933733     69.177942
4509.902963
Net Sales Value    30466.336131  4677.788059    326.974801
46358.656624
```

Data Visualization

```
# Calculate skewness for each numerical column
skewness = df[numerical_cols].skew()

# Plot histograms for each numerical column
plt.figure(figsize=(15, 10))
for i, col in enumerate(numerical_cols, 1):
    plt.subplot(3, 3, i)
    sns.histplot(df[col], kde=True)
    plt.title(col)
    plt.xlabel(f"Skewness: {skewness[col]:.2f}")
plt.tight_layout()
plt.show()

# Identify outliers using IQR method and provide inferences
for col in numerical_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
    print(f"Column: {col}")
    print(f"Number of outliers: {len(outliers)}")
```

```
    print(f"Outlier values: {outliers[col].values}")
    print(f"Lower bound: {lower_bound}, Upper bound: {upper_bound}")
    print("")

# Provide inferences
print("Inferences:")
print("1. Skewness:")
for col, val in skewness.items():
    if abs(val) > 1:
        print(f"   - Column '{col}' is highly skewed ({val:.2f}).")
    elif abs(val) > 0.5:
        print(f"   - Column '{col}' is moderately skewed
({val:.2f}).")
    else:
        print(f"   - Column '{col}' is approximately symmetric
({val:.2f}).")

print("\n2. Outliers:")
for col in numerical_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    num_outliers = len(df[(df[col] < lower_bound) | (df[col] >
upper_bound)])
    if num_outliers > 0:
        print(f"   - Column '{col}' has {num_outliers} outliers.")
```

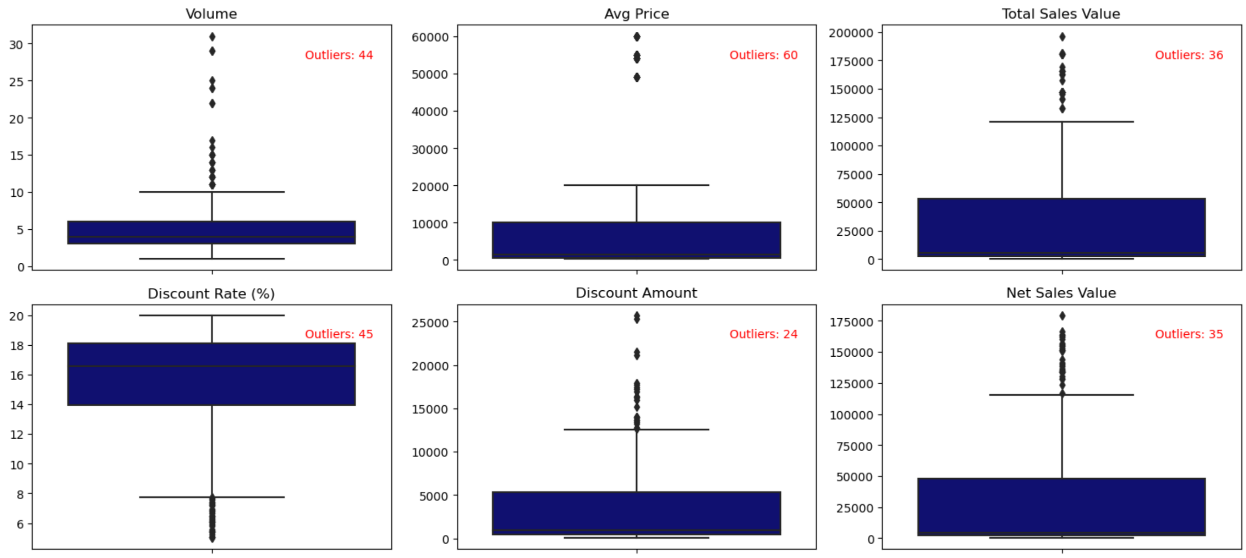


```
Column: Volume
Number of outliers: 44
Outlier values: [15 13 11 29 13 24 14 12 25 15 11 15 14 12 12 12 22 11
11 12 12 14 11 11
 15 31 12 16 24 11 12 12 22 11 13 15 12 14 14 11 29 15 13 17]
Lower bound: -1.5, Upper bound: 10.5

Column: Avg Price
Number of outliers: 60
Outlier values: [49100 54100 55100 60100 49100 54100 55100 60100 49100
54100 55100 60100
 49100 54100 55100 60100 49100 54100 55100 60100 49100 54100 55100
60100
 49100 54100 55100 60100 49100 54100 55100 60100 49100 54100 55100
60100
 49100 54100 55100 60100 49100 54100 55100 60100 49100 54100 55100
60100
 49100 54100 55100 60100 49100 54100 55100 60100 49100 54100 55100
60100]
Lower bound: -13987.5, Upper bound: 24552.5

Column: Total Sales Value
Number of outliers: 36
Outlier values: [181500 147300 180300 133100 147300 165300 180300
196400 147300 147300
 162300 162300 145200 147300 162300 165300 180300 169400 147300 181500
 140700 147300 165300 145200 147300 165300 180300 165300 180300 140700
 147300 133100 147300 157300 147300 165300]
Lower bound: -73050.0, Upper bound: 128950.0

Column: Discount Rate (%)
Number of outliers: 45
Outlier values: [6.93538533 5.55371934 7.41010449 6.2148882
5.25211255 7.62179096
 5.00782219 5.87067094 6.71045354 6.09520144 5.93508419 7.58459064
 7.73266709 7.23384674 5.42050666 6.84997564 7.25669557 7.1787259
 7.6793856  5.79480208 5.05980128 6.85825457 7.20836295 7.34187434
 6.47330471 6.43991996 7.4213256  6.26891381 6.81911066 6.17039789
 5.07212419 6.1069307  6.50871908 6.06619192 5.08410843 6.32689169
 6.41523029 5.05521841 5.41180219 5.51104232 5.48515667 5.46637934
 6.00819957 6.64259534 5.42591053]
Lower bound: 7.740578642625298, Upper bound: 24.339202378829146

Column: Discount Amount
Number of outliers: 24
Outlier values: [21153.49881959 13594.039719   17900.98373313
17445.6038281
 13951.66019446 16384.02900944 16892.52095098 15214.6433236
 12622.50365771 17178.33185948 12753.56595799 13999.93849871
 17696.81362055 25328.2242042  13608.23831923 25738.02219376
```

```
 21496.67536736 16332.91992954 14036.83865216 12734.00901241
 13275.78074114 16218.59472035 13382.22733346 15984.73228058]
Lower bound: -6823.594880316146, Upper bound: 12600.54961088833

Column: Net Sales Value
Number of outliers: 35
Outlier values: [160346.50118041 133705.960281   162399.01626687
139563.63821492
 151348.33980554 163915.971      179507.47904902 134677.49634229
 134731.95462498 152667.35835357 151182.48953317 128021.66814052
 134546.43404201 150648.92786553 151300.06150129 162603.18637945
 144071.7757958  133691.76168077 155761.97780624 130557.83332703
 136485.41909127 154937.48547455 123703.32463264 134934.94669154
 154541.08736469 163967.08007046 156895.96877157 166263.16134784
 127965.99098759 134024.21925886 116881.40527965 133917.77266654
 141315.26771942 138449.92203905 156330.96988963]
Lower bound: -66266.347664084, Upper bound: 116316.46916099661

Inferences:
1. Skewness:
   - Column 'Volume' is highly skewed (2.73).
   - Column 'Avg Price' is highly skewed (1.91).
   - Column 'Total Sales Value' is highly skewed (1.53).
   - Column 'Discount Rate (%)' is highly skewed (-1.06).
   - Column 'Discount Amount' is highly skewed (1.91).
   - Column 'Net Sales Value' is highly skewed (1.54).

2. Outliers:
   - Column 'Volume' has 44 outliers.
   - Column 'Avg Price' has 60 outliers.
   - Column 'Total Sales Value' has 36 outliers.
   - Column 'Discount Rate (%)' has 45 outliers.
   - Column 'Discount Amount' has 24 outliers.
   - Column 'Net Sales Value' has 35 outliers.
```

```
# Plot boxplots for each numerical column
plt.figure(figsize=(15, 10))
for i, col in enumerate(numerical_cols, 1):
    plt.subplot(3, 3, i)
    sns.boxplot(y=df[col], color='navy')
    plt.title(col)
    plt.ylabel("")

    # Calculate outliers using IQR method
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
```

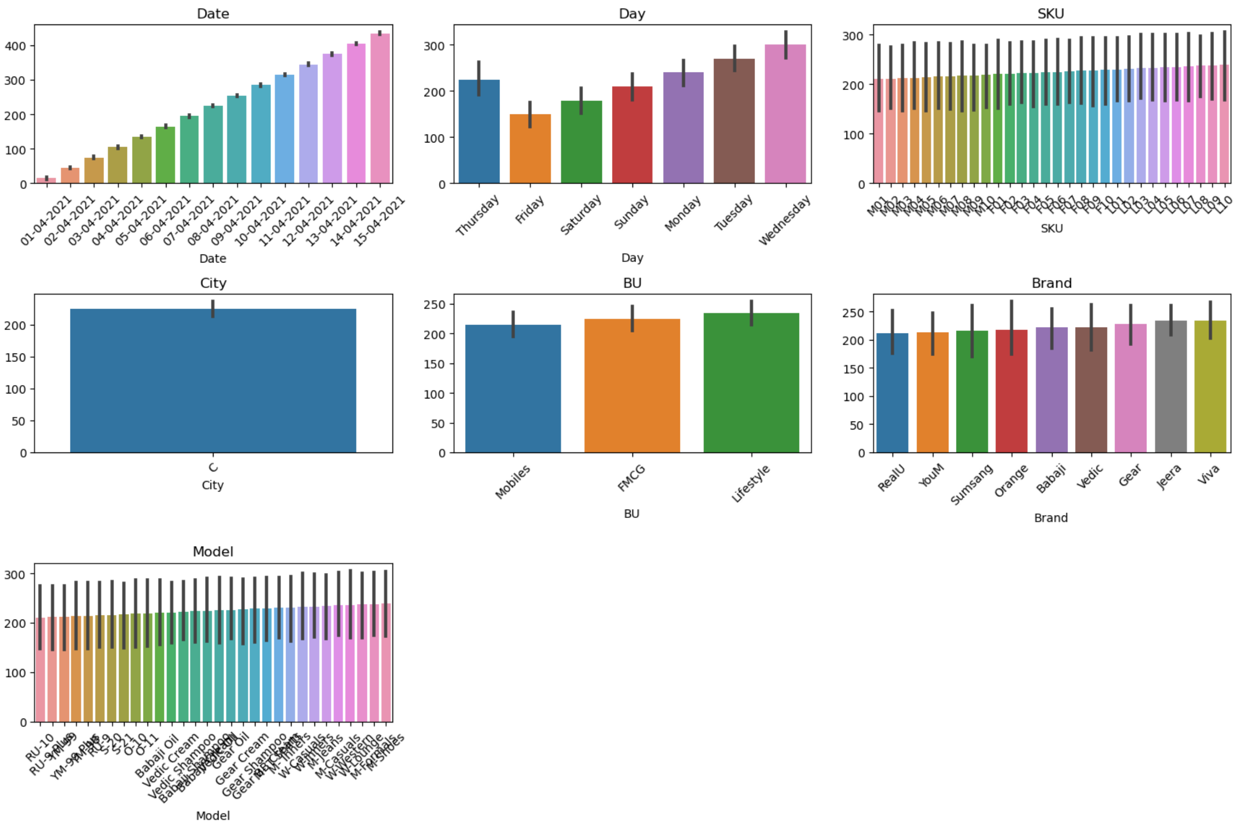
```
    plt.text(0.95, 0.9, f"Outliers: {len(outliers)}",
transform=plt.gca().transAxes, ha='right', va='top', color='red')
plt.tight_layout()
plt.show()
```




```
categorical_cols = df.select_dtypes(include=['object']).columns
categorical_cols
```

```
Index(['Date', 'Day', 'SKU', 'City', 'BU', 'Brand', 'Model'],
dtype='object')
```

```
# Plot bar charts for each categorical column
plt.figure(figsize=(15, 10))
for i, col in enumerate(categorical_cols, 1):
    plt.subplot(3, 3, i)
    sns.barplot(data=df, x=col, y=df.index)
    plt.title(col)
    plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Standardization of Numerical Variables

```
numerical_cols

Index(['Volume', 'Avg Price', 'Total Sales Value', 'Discount Rate
(%)',
       'Discount Amount', 'Net Sales Value'],
      dtype='object')

# To know the mean and standard deviation of each column
df.describe()

           Volume     Avg Price  Total Sales Value  Discount Rate (%)
\
count  450.000000    450.000000         450.000000         450.000000

mean     5.066667  10453.433333       33812.835556          15.155242

std      4.231602  18079.904840       50535.074173           4.220602

min      1.000000    290.000000         400.000000           5.007822

25%      3.000000    465.000000        2700.000000          13.965063

50%      4.000000   1450.000000        5700.000000          16.577766
```

```
75%      6.000000  10100.000000       53200.000000          18.114718

max     31.000000  60100.000000      196400.000000          19.992407


       Discount Amount  Net Sales Value
count       450.000000       450.000000
mean       3346.499424     30466.336131
std        4509.902963     46358.656624
min          69.177942       326.974801
25%         460.459304      2202.208645
50%         988.933733      4677.788059
75%        5316.495427     47847.912852
max       25738.022194    179507.479049
```

```
df[numerical_cols].agg(['std'])
```

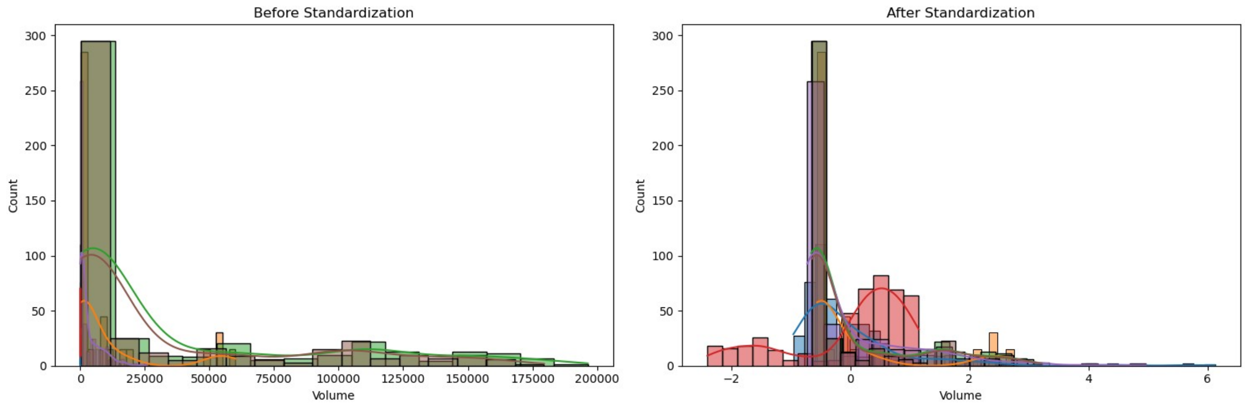
```
Index(['Volume', 'Avg Price', 'Total Sales Value', 'Discount Rate
(%)',
       'Discount Amount', 'Net Sales Value'],
      dtype='object')
```

```
# Identify numerical columns
numerical_cols = df.select_dtypes(include=['number']).columns

# Standardize numerical columns
df_standardized = df.copy()
for col in numerical_cols:
    mu = df[col].mean()  # Calculate mean for the column
    sigma = df[col].std()  # Calculate standard deviation for the
column
    df_standardized[col] = (df[col] - mu) / sigma  # Standardize the
column

# Display the standardized DataFrame
df_standardized
```

```
           Date        Day  SKU City    Volume         BU  Brand
Model  \
0    01-04-2021  Thursday  M01    C  2.347417    Mobiles  RealU
RU-10
1    01-04-2021  Thursday  M02    C  1.165831    Mobiles  RealU   RU-9
Plus
2    01-04-2021  Thursday  M03    C  0.456880    Mobiles   YouM
YM-99
3    01-04-2021  Thursday  M04    C  0.220563    Mobiles   YouM  YM-99
Plus
4    01-04-2021  Thursday  M05    C -0.488389    Mobiles   YouM
YM-98
..          ...        ...  ...  ...       ...        ...    ...
...
```

```
445  15-04-2021  Thursday  L06    C -0.724706  Lifestyle  Jeera   M-
Casuals
446  15-04-2021  Thursday  L07    C  0.220563  Lifestyle   Viva   W-
Western
447  15-04-2021  Thursday  L08    C -0.724706  Lifestyle   Viva    W-
Lounge
448  15-04-2021  Thursday  L09    C -0.488389  Lifestyle  Jeera   M-
Formals
449  15-04-2021  Thursday  L10    C -0.961023  Lifestyle  Jeera     M-
Shoes

     Avg Price  Total Sales Value  Discount Rate (%)  Discount Amount
\
0     0.091072           2.922469          -0.829365         3.948422

1    -0.019548           1.329516          -0.851714         1.846958

2     0.312312           1.561038          -1.350129         1.621190

3     0.533552           1.717365          -1.947555         1.112568

4    -0.130168          -0.188242           0.672990         0.227598

..         ...                ...                ...              ...

445  -0.506277          -0.617647           0.075924        -0.652815

446  -0.434374          -0.360400           0.450596        -0.152022

447  -0.489684          -0.605774           0.902788        -0.607464

448  -0.473091          -0.556303           0.388042        -0.529789

449  -0.406719          -0.607753           0.042188        -0.636636


     Net Sales Value
0           2.801638
1           1.269613
2           1.543957
3           1.763847
4          -0.227342
..               ...
445        -0.609783
446        -0.378079
447        -0.601252
448        -0.554881
449        -0.600571

[450 rows x 13 columns]
```

```python
# Plot histograms for each numerical column before and after
standardization
plt.figure(figsize=(15, 5))
for i, col in enumerate(numerical_cols, 1):
    plt.subplot(1, 2, 1)
    sns.histplot(df[col], kde=True)
    plt.title('Before Standardization')

    plt.subplot(1, 2, 2)
    sns.histplot(df_standardized[col], kde=True)
    plt.title('After Standardization')

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



Conversion of Categorical Data into Dummy Variables

```python
# Identify categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns

# Apply one-hot encoding to categorical columns
df_encoded = pd.get_dummies(df, columns=categorical_cols, dtype='int')

# Display a portion of the transformed dataset
df_encoded
```

```
     Volume  Avg Price  Total Sales Value  Discount Rate (%)  Discount
Amount  \
0        15      12100             181500          11.654820
21153.498820
1        10      10100             101000          11.560498
11676.102961
2         7      16100             112700           9.456886
10657.910157
3         6      20100             120600           6.935385
8364.074702
4         3       8100              24300          17.995663
```

```
4372.946230
..      ...        ...                 ...                 ...
...
445       2       1300                 2600          15.475687
402.367873
446       6       2600                15600          17.057027
2660.896242
447       2       1600                 3200          18.965550
606.897606
448       3       1900                 5700          16.793014
957.201826
449       1       3100                 3100          15.333300
475.332295

     Net Sales Value  Date_01-04-2021  Date_02-04-2021  Date_03-04-
2021  \
0      160346.501180                1                0
0
1       89323.897039                1                0
0
2      102042.089843                1                0
0
3      112235.925298                1                0
0
4       19927.053770                1                0
0
..               ...              ...              ...              ..
.
445      2197.632127                0                0
0
446     12939.103758                0                0
0
447      2593.102394                0                0
0
448      4742.798174                0                0
0
449      2624.667705                0                0
0

     Date_04-04-2021  ...  Model_Vedic Cream  Model_Vedic Oil  \
0                  0  ...                  0                0
1                  0  ...                  0                0
2                  0  ...                  0                0
3                  0  ...                  0                0
4                  0  ...                  0                0
..               ...  ...                ...              ...
445                0  ...                  0                0
446                0  ...                  0                0
447                0  ...                  0                0
```

```
448                0  ...                  0                0
449                0  ...                  0                0

     Model_Vedic Shampoo  Model_W-Casuals  Model_W-Inners  Model_W-
Lounge  \
0                      0                0               0
0
1                      0                0               0
0
2                      0                0               0
0
3                      0                0               0
0
4                      0                0               0
0
..                   ...              ...             ...
...
445                    0                0               0
0
446                    0                0               0
0
447                    0                0               0
1
448                    0                0               0
0
449                    0                0               0
0

     Model_W-Western  Model_YM-98   Model_YM-99  Model_YM-99 Plus
0                  0            0             0                 0
1                  0            0             0                 0
2                  0            0             1                 0
3                  0            0             0                 1
4                  0            1             0                 0
..               ...          ...           ...               ...
445                0            0             0                 0
446                1            0             0                 0
447                0            0             0                 0
448                0            0             0                 0
449                0            0             0                 0

[450 rows x 101 columns]
```

Conclusion