Final Project

Data Pre-Processing

Investigating The Data

This Process Includes A Few Steps:

- Identifying Irrelevant Columns.
- Handling Nulls and Invalid Values.
- Handling Invalid Columns Data types.

Identifying Irrelevant Columns :

After Investigating The Data Carefully I was able to Identify The Irrelevant Columns as Follows:

- asin , currency , delivery , has_variations,

  unit_ price , unit_count, coupon_text, product_avilability.

```
df.drop(columns = 'asin', inplace = True)
df.drop(columns = 'currency', inplace = True)
df.drop(columns = 'delivery', inplace = True)
df.drop(columns = 'has_variations', inplace = True)
df.drop(columns = 'unit_price', inplace = True)
df.drop(columns = 'unit_count', inplace = True)
df.drop(columns = 'coupon_text', inplace = True)
```

```
[52]:
```

```
df.drop(columns = 'product_availability', inplace = True)
```

Handling Invalid Values

First I had to deal with the (sales_volume) column Because I noted That it has many Invalid Values like (List:, Typical:,.....) So I Mapped This Column to Another Column using the (.map) function.

Named (Sales_Last_Month) and Ignored all the

Invalid Values also mapped the new values using proper data types Then Dropped The Original Column.

```
df['Sales_per_last_month']=df['sales_volume'].map({"100+ bought in past month":100,
                                                    "200+ bought in past month":200,
                                                   "500+ bought in past month":500,
                                                   "50+ bought in past month":50,
                                                   "1K+ bought in past month":1000,
                                                   "300+ bought in past month":300,
                                                   "400+ bought in past month":400,
                                                   "2K+ bought in past month":2000,
                                                   "4K+ bought in past month":4000,
                                                   "3K+ bought in past month":3000,
                                                   "900+ bought in past month":900,
                                                   "600+ bought in past month":600,
                                                   "700+ bought in past month":700})
```

Handling Invalid Column Data types:

These Columns were (product price) (product minimum offer price) (product original price).

This Process applies to Three of them to handle inappropriate data type:

1- having ($) and (,) in a value wont allow the type converting:

using the replace function to replace em with ('') .

looping through the data to avoid the nulls as they read by default as float values and the replace function won't work.

2- converting the column data type to float to handle the nulls.

Product Price Column

```
for index, value in df['product_price'].items():
    dt = type(value)
    if dt == float:
        pass
    else:
        df.at[index, 'product_price'] = value.replace('$','')
```

[64]:

```
for index, value in df['product_price'].items():
    dt = type(value)
    if dt == float:
        pass
    else:
        df.at[index, 'product_price'] = value.replace(',','')
```

[46]:

```
df['product_price'] = df['product_price'].astype('float')
```

Product Minimum Price Offer

```
for index, value in df['product_minimum_offer_price'].items():
    dt = type(value)
    if dt == float:
        pass
    else:
        df.at[index, 'product_minimum_offer_price'] = value.replace(',','')
```

```
for index, value in df['product_minimum_offer_price'].items():
    dt = type(value)
    if dt == float:
        pass
    else:
        df.at[index, 'product_minimum_offer_price'] = value.replace('$','')
```

```
df['product_minimum_offer_price'] = df['product_minimum_offer_price'].astype('float')
```

Product Original Price

```
for index, value in df['product_original_price'].items():
    dt = type(value)
    if dt == float:
        pass
    else:
        df.at[index, 'product_original_price'] = value.replace('$','')
```

```
for index, value in df['product_original_price'].items():
    dt = type(value)
    if dt == float:
        pass
    else:
        df.at[index, 'product_original_price'] = value.replace(',','')
```

```
df['product_original_price'] = df['product_original_price'].astype('float')
```

Identifying The Outliers

Identifying if The Data has Outliers

This Step is necessary for The replacing the nulls whether we gonna use the (Mean) or (Median).

```
Q1 = df['product_price'].quantile(0.25)
Q3 = df['product_price'].quantile(0.75)
IQR = Q3 - Q1

# identify outliers
threshold = 1.5
outliers = df[(df['product_price'] < Q1 - threshold * IQR) | (df['product_price'] > Q3 + threshold * IQR)]
```

Using .describe() Function

```
df.describe()
```

	product_price	product_original_price	product_star_rating	product_num_ratings	product_num_offers	product_minimum_offer_price	Sales_per_last_month
count	336.000000	148.000000	337.000000	340.000000	340.000000	336.000000	340.000000
mean	172.387768	238.589527	4.087537	2744.050000	8.426471	146.787143	460.147059
std	201.455675	286.515857	0.383544	7331.816833	10.798389	190.479132	698.834877
min	7.990000	15.800000	1.600000	0.000000	1.000000	7.990000	0.000000
25%	59.990000	83.422500	3.900000	135.250000	1.750000	44.252500	50.000000
50%	114.280000	146.470000	4.100000	503.500000	4.000000	84.995000	200.000000
75%	199.990000	287.242500	4.300000	1874.250000	11.000000	164.612500	500.000000
max	1614.990000	2019.990000	5.000000	64977.000000	83.000000	1583.650000	4000.000000

Replacing The Nulls

Replacing The Nulls

Given That The Data has Outliers so I Chose to replace the Nulls With The Median Value To The Avoid The Influence of The Extreme Values.

```
df.fillna({'product_price':df['product_price'].median()}, inplace = True)
```

```
df.fillna({'product_star_rating':df['product_star_rating'].median()}, inplace = True)
```

```
df.fillna({'product_minimum_offer_price':df['product_minimum_offer_price'].median()}, inplace = True)
```

Replacing The Nulls

I Chose To Fill The Sales per Month Column with (0) Given That it has ratings that the sales quantity referred for the last month .

```
df.fillna({'Sales_per_last_month':0.0}, inplace = True)
```

Thank You