```
# Cleaning and preprocessing of data.
import numpy as np
import pandas as pd
```

```
df = pd.read_csv('products (2).csv')
```

```
print(df)
```

```
      Index                                     Name  \
0         1  Smart Fan Iron Cooker Go Wireless Portable
1         2                                      Fan
2         3   Smart Speakerphone Charger Eco Plus Clean
3         4              Premium Grill Trimmer Portable
4         5                          Keyboard Freezer
...     ...                                      ...
9995   9996                                Powerbank
9996   9997            Rechargeable Lamp Scanner Vacuum
9997   9998                                   Heater
9998   9999              Rechargeable Headphones Stove
9999  10000                    Clean Charger Treadmill

                                      Description  \
0                            Catch enough role nearly.
1                            All movement yeah tax me.
2                   Quickly inside pull line lay start.
3                              Lawyer one than fire.
4                     Remain Congress blood plan voice.
...                                             ...
9995                  Yeah none boy fund city decision.
9996    Appear country enter there pull thing computer.
9997  Owner plan approach soldier church little little.
9998                    Base account partner probably.
9999        Return reflect truth tell scientist season.

                       Brand             Category  MRP_Price Currency  Stock  \
0                 Herman Ltd       Kids' Clothing      585.0      INR  194.0
1     Braun, King and Rollins       Grooming Tools      992.0      INR  724.0
2               Peck-Coleman    Fishing & Hunting      940.0      INR  769.0
3                  Hines Ltd             Skincare      324.0      INR   93.0
4     Spence, Webster and Orr  Laptops & Computers      908.0      INR  614.0
...                       ...                  ...        ...      ...    ...
9995       Castaneda-Harrison   Books & Stationery      567.0      INR   27.0
9996                Knapp-Key              Cycling      300.0      INR  177.0
9997    Meyer, Case and Mills    Fitness Equipment      399.0      INR  996.0
9998             Jones-Bell   Kitchen Appliances       45.0      INR  274.0
9999                Davis LLC    Sports & Outdoors       31.0      INR   46.0

               EAN        Color   Availability  Internal ID    Qty  \
0     3.968600e+12     Cornsilk  limited_stock           54  548.0
1     1.911270e+11       Bisque   discontinued           49  228.0
2     7.569140e+12         Blue      pre_order           42  520.0
3     2.705140e+12        Ivory   out_of_stock           93  251.0
4     9.830390e+12  FloralWhite   discontinued           91  602.0
...            ...          ...            ...          ...    ...
9995  6.519000e+12         Cyan      pre_order            7  206.0
9996  4.707960e+12    MintCream  limited_stock           55  557.0
9997  5.286510e+12    LightGray      pre_order           37  698.0
9998  3.702650e+12    SlateBlue       in_stock           52  183.0
9999  8.879840e+12        Ivory      backorder           70  904.0

      Selling_price
0            345.15
1            585.28
2            601.60
3            178.20
4            572.04
```

```
# check how much table and column
df.shape
```

```
(10000, 14)
```

```
#check type data form each column
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Index          10000 non-null  int64
 1   Name           10000 non-null  object
 2   Description    10000 non-null  object
 3   Brand          10000 non-null  object
```

```
 4   Category       10000 non-null  object
 5   MRP_Price      9997 non-null   float64
 6   Currency       10000 non-null  object
 7   Stock          9978 non-null   float64
 8   EAN            10000 non-null  float64
 9   Color          10000 non-null  object
 10  Availability   10000 non-null  object
 11  Internal ID    10000 non-null  int64
 12  Qty            9991 non-null   float64
 13  Selling_price  9985 non-null   float64
dtypes: float64(5), int64(2), object(7)
memory usage: 1.1+ MB
```

```
#check null data
df.isna().any()
```

	0
Index	False
Name	False
Description	False
Brand	False
Category	False
MRP_Price	True
Currency	False
Stock	True
EAN	False
Color	False
Availability	False
Internal ID	False
Qty	True
Selling_price	True

dtype: bool

```
#add up the blank values
df.isna().sum()
```

	0
Index	0
Name	0
Description	0
Brand	0
Category	0
MRP_Price	3
Currency	0
Stock	22
EAN	0
Color	0
Availability	0
Internal ID	0
Qty	9
Selling_price	15

dtype: int64

```
#check duplicate data
df.duplicated().sum()
```

```
np.int64(0)
```

```
#check unique value
df.nunique()
```

	0
Index	10000
Name	7896
Description	10000
Brand	9241
Category	34
MRP_Price	999
Currency	1
Stock	999
EAN	9952
Color	140
Availability	6
Internal ID	99
Qty	951
Selling_price	7905

dtype: int64

```
#descirbe all
df.describe(include="all").T
```

	count	unique	top	freq	mean	std	min	25%
Index	10000.0	NaN	NaN	NaN	5000.5	2886.89568	1.0	2500.75
Name	10000	7896	Fan	44	NaN	NaN	NaN	NaN
Description	10000	10000	Between key father half.	1	NaN	NaN	NaN	NaN
Brand	10000	9241	Kennedy LLC	5	NaN	NaN	NaN	NaN
Category	10000	34	Clothing & Apparel	320	NaN	NaN	NaN	NaN
MRP_Price	9997.0	NaN	NaN	NaN	503.327498	289.746061	1.0	251.0
Currency	10000	1	INR	10000	NaN	NaN	NaN	NaN
Stock	9978.0	NaN	NaN	NaN	499.14071	288.02384	1.0	249.0
EAN	10000.0	NaN	NaN	NaN	5035709017000.380859	2887653409184.608887	261275896.0	2545170000000.0
Color	10000	140	Cornsilk	101	NaN	NaN	NaN	NaN
Availability	10000	6	discontinued	1706	NaN	NaN	NaN	NaN
Internal ID	10000.0	NaN	NaN	NaN	50.4755	28.555993	1.0	26.0
Qty	9991.0	NaN	NaN	NaN	523.69983	276.426889	50.0	281.5

```
df.isna().sum()
```

	0
Index	0
Name	0
Description	0
Brand	0
Category	0
MRP_Price	3
Currency	0
Stock	22
EAN	0
Color	0
Availability	0
Internal ID	0
Qty	9
Selling_price	15

dtype: int64

```
#fill na with 0
df['Stock'].fillna(0,inplace=True)
```

```
/tmp/ipython-input-2987266135.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through ch
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col]


  df['Stock'].fillna(0,inplace=True)
```

```
#fill na with 0

df['Qty'].fillna(0,inplace=True)
```

```
/tmp/ipython-input-290176259.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through cha
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col]


  df['Qty'].fillna(0,inplace=True)
```

```
# blank mrp_price analysis
```

```
mean_mrp_price = df['MRP_Price'].mean().round(2)
```

```
print(mean_mrp_price)
```

```
503.33
```

```
median_mrp_price = df['MRP_Price'].median()
print(median_mrp_price)
```

```
506.0
```

```
# brand wise calculating mean, median for mrp_price
```

```
brand_mrp_stats = df.groupby("Brand")["MRP_Price"].agg(
    mean_mrp="mean",
    median_mrp="median",
    count="count"
).reset_index()

print(brand_mrp_stats)
```

```
                          Brand  mean_mrp  median_mrp  count
0                     Abbott Ltd      95.0        95.0      1
1        Abbott, Holden and Conrad     800.0       800.0      1
2         Abbott, Lynch and Warner     521.0       521.0      1
```

```
3        Abbott, Pitts and Charles     527.0       527.0      1
4       Abbott, Zavala and Patrick     611.0       611.0      1
...                            ...       ...         ...    ...
9236   Zuniga, Schwartz and Barber     664.0       664.0      1
9237  Zuniga, Zavala and Contreras      71.0        71.0      1
9238                 Zuniga-Allen     496.0       496.0      1
9239              Zuniga-Mcpherson     969.0       969.0      1
9240                 Zuniga-Yates      32.0        32.0      1

[9241 rows x 4 columns]
```

```
# Missing MRP values were filled using brand-wise median pricing, with overall median as fallback where brand-level data wa
```

```
mrp_median = df.groupby("Brand")["MRP_Price"].median()
print(mrp_median)
```

```
Brand
Abbott Ltd                       95.0
Abbott, Holden and Conrad       800.0
Abbott, Lynch and Warner        521.0
Abbott, Pitts and Charles       527.0
Abbott, Zavala and Patrick      611.0
                                ...
Zuniga, Schwartz and Barber     664.0
Zuniga, Zavala and Contreras     71.0
Zuniga-Allen                    496.0
Zuniga-Mcpherson                969.0
Zuniga-Yates                     32.0
Name: MRP_Price, Length: 9241, dtype: float64
```

```
df["MRP_Price"] = df["MRP_Price"].fillna(
    df["Brand"].map(mrp_median)
)
```

```
overall_median = df["MRP_Price"].median()
df["MRP_Price"] = df["MRP_Price"].fillna(overall_median)
```

```
print(overall_median)
```

```
506.0
```

```
# selling price analysis
```

```
mean_selling_price = df['Selling_price'].mean()
```

```
print(mean_selling_price)
```

```
365.3695943915874
```

```
median_selling_price = df['Selling_price'].median()
```

```
print(median_selling_price)
```

```
353.4
```

```
# brand wise calculating mean, median for selling price
```

```
brand_selling_stats = df.groupby("Brand")["Selling_price"].agg(
    mean_mrp="mean",
    median_mrp="median",
    count="count"
).reset_index()

print(brand_selling_stats)
```

```
                           Brand  mean_mrp  median_mrp  count
0                     Abbott Ltd     60.80       60.80      1
1        Abbott, Holden and Conrad    488.00      488.00      1
2         Abbott, Lynch and Warner    401.17      401.17      1
3        Abbott, Pitts and Charles    447.95      447.95      1
4       Abbott, Zavala and Patrick    446.03      446.03      1
...                            ...       ...         ...    ...
9236   Zuniga, Schwartz and Barber    544.48      544.48      1
9237  Zuniga, Zavala and Contreras     46.86       46.86      1
9238                 Zuniga-Allen    381.92      381.92      1
9239              Zuniga-Mcpherson    629.85      629.85      1
9240                 Zuniga-Yates     16.64       16.64      1

[9241 rows x 4 columns]
```

```
overall_selling_price_median = df["Selling_price"].median()
df["Selling_price"] = df["Selling_price"].fillna(overall_selling_price_median)
```

```
print(overall_selling_price_median)
```

```
353.4
```

```
df.isna().sum()
```

	0
Index	0
Name	0
Description	0
Brand	0
Category	0
MRP_Price	0
Currency	0
Stock	0
EAN	0
Color	0
Availability	0
Internal ID	0
Qty	0
Selling_price	0

dtype: int64

```
df
```

	Index	Name	Description	Brand	Category	MRP_Price	Currency	Stock	EAN	Color	Availabil
0	1	Smart Fan Iron Cooker Go Wireless Portable	Catch enough role nearly.	Herman Ltd	Kids' Clothing	585.0	INR	194.0	3.968600e+12	Cornsilk	limited_st
1	2	Fan	All movement yeah tax me.	Braun, King and Rollins	Grooming Tools	992.0	INR	724.0	1.911270e+11	Bisque	discontinu
2	3	Smart Speakerphone Charger Eco Plus Clean	Quickly inside pull line lay start.	Peck-Coleman	Fishing & Hunting	940.0	INR	769.0	7.569140e+12	Blue	pre_or
3	4	Premium Grill Trimmer Portable	Lawyer one than fire.	Hines Ltd	Skincare	324.0	INR	93.0	2.705140e+12	Ivory	out_of_st
4	5	Keyboard Freezer	Remain Congress blood plan voice.	Spence, Webster and Orr	Laptops & Computers	908.0	INR	614.0	9.830390e+12	FloralWhite	discontinu
...	...	...	...	...	...	...	...	...	...	...	
9995	9996	Powerbank	Yeah none boy fund city decision.	Castaneda-Harrison	Books & Stationery	567.0	INR	27.0	6.519000e+12	Cyan	pre_or
9996	9997	Rechargeable Lamp Scanner Vacuum	Appear country enter there pull thing computer.	Knapp-Key	Cycling	300.0	INR	177.0	4.707960e+12	MintCream	limited_st
9997	9998	Heater	Owner plan approach soldier church little little	Meyer, Case and Mills	Fitness Equipment	399.0	INR	996.0	5.286510e+12	LightGray	pre_or

```
# checking if there space
```

```
df['Brand'].unique().tolist()
```

```
 'Saunders, Farley and Bradley',
 'Rosario-Velez',
 'Peters and Sons',
 'Abbott-Mccall',
 'Walker, Pittman and Wood',
 'Waller Ltd',
 'Golden, Booker and Green',
 'Norman, Dalton and Wolf',
 'Richardson, Knox and Parsons',
 'Hutchinson Ltd',
 'Mcmillan, Benton and Schneider',
 'Gilmore, Frey and Barron',
 'Carpenter Group',
 'Brady-Meyers',
 'Sanford, Pennington and Lang',
 'Barrera-Phelps',
 'Wu, Mcclure and Peters',
 'Mcdaniel-Shelton',
 'Weiss LLC',
 'Finley, Frey and Adkins',
 'Benton Inc',
 'Blevins, Meadows and Porter',
 'Pham-Robinson',
 'Archer, Bryant and Parker',
 'Lawson Group',
 'Norton LLC',
 'Randolph-Richardson',
 'Dixon-Pearson',
 'Anderson, Barker and Blevins',
 'Molina Group',
 'Drake-Stanley',
 'Simon, Andrade and Riggs',
 'West LLC',
 'Hernandez-Castro',
 'Villanueva-Steele',
 'Potter LLC',
 'Hobbs-Key',
 'Cruz-Sanford',
 'Robertson-Brown',
 'Huffman PLC',
 'Villegas-Stephenson',
 'Patton, Merritt and Baker',
 'Arnold-Melton',
 'Howe Group',
 'Hurst, Travis and Cervantes',
 'Frank and Sons',
 'Simpson LLC',
 'Ballard-Potter',
 'Keith-Howe',
 'Bradshaw LLC',
 'Brady PLC',
 'Goodwin Inc',
 'Wilkins PLC',
 'Frazier, Gamble and Meza',
 'Mcguire, Peterson and Pratt',
 'Fernandez-Russell',
 'Mann Ltd',
 'Hudson, Mueller and Vargas',
```

```
df['Category'].unique().tolist()
```

```
["Kids' Clothing",
 'Grooming Tools',
 'Fishing & Hunting',
 'Skincare',
 'Laptops & Computers',
 'Kitchen Appliances',
 'Clothing & Apparel',
 'Beauty & Personal Care',
 'Makeup',
 'Toys & Games',
 "Men's Clothing",
 'Camping & Hiking',
 'Cleaning Supplies',
 'Headphones & Earbuds',
 'Home & Kitchen',
 'Haircare',
 'Cameras & Accessories',
 'Bedding & Bath',
 'Accessories (Bags, Hats, Belts)',
 'Team Sports',
 'Smartwatches',
 'Fragrances',
 "Women's Clothing",
 'Shoes & Footwear',
```

```
 'Furniture',
 'Home Decor',
 'Fitness Equipment',
 'Health & Wellness',
 'Smartphones',
 'Office Supplies',
 'Cycling',
 'Automotive',
 'Sports & Outdoors',
 'Books & Stationery']
```

```
df['Brand'].value_counts()
```

Brand	count
Kennedy LLC	5
Buckley Inc	4
Watkins Group	4
Flores LLC	4
Johnston Ltd	4
...	...
Hendricks LLC	1
Mcconnell, Rangel and Brennan	1
Wall-Mack	1
Mcfarland-Henry	1
Graham Ltd	1

9241 rows × 1 columns

dtype: int64

```
df['Availability'].unique().tolist()
```

```
['limited_stock',
 'discontinued',
 'pre_order',
 'out_of_stock',
 'in_stock',
 'backorder']
```

```
df['Availability'].value_counts()
```

Availability	count
discontinued	1706
out_of_stock	1700
pre_order	1673
limited_stock	1644
in_stock	1644
backorder	1633

dtype: int64

```
df['Color'].unique().tolist()
```

```
['Cornsilk',
 'Bisque',
 'Blue',
 'Ivory',
 'FloralWhite',
 'LightSalmon',
 'MediumPurple',
 'PaleGreen',
 'DarkBlue',
 'SandyBrown',
 'Thistle',
 'OldLace',
 'LightSkyBlue',
 'MediumBlue',
 'DarkTurquoise',
```

```
 'Azure',
 'DeepSkyBlue',
 'DarkSeaGreen',
 'BlanchedAlmond',
 'LavenderBlush',
 'LemonChiffon',
 'SaddleBrown',
 'CadetBlue',
 'DarkSalmon',
 'HoneyDew',
 'DarkRed',
 'Peru',
 'AntiqueWhite',
 'DarkCyan',
 'LightGray',
 'RoyalBlue',
 'CornflowerBlue',
 'Red',
 'PaleGoldenRod',
 'IndianRed',
 'LimeGreen',
 'Aqua',
 'Snow',
 'SpringGreen',
 'Teal',
 'DarkOrchid',
 'LightGreen',
 'GreenYellow',
 'Plum',
 'MediumOrchid',
 'Fuchsia',
 'MediumSpringGreen',
 'LightBlue',
 'Orchid',
 'MediumTurquoise',
 'RosyBrown',
 'Crimson',
 'Tomato',
 'LightGoldenRodYellow',
 'LightSeaGreen',
 'SlateBlue',
 'PowderBlue',
 'Salmon',
```

```
df['Color'].value_counts()
```

Color	count
Cornsilk	101
Peru	93
AntiqueWhite	92
Ivory	92
Salmon	88
...	...
MediumPurple	56
FloralWhite	55
Wheat	53
White	49
Turquoise	48

140 rows × 1 columns

dtype: int64

```
df['Name'].unique().tolist()
```

```
['Smart Fan Iron Cooker Go Wireless Portable',
 'Fan',
 'Smart Speakerphone Charger Eco Plus Clean',
 'Premium Grill Trimmer Portable',
 'Keyboard Freezer',
 'Smart Fridge Plus',
 'Compact Dock One Portable Wireless',
 'Premium Cooler Digital Portable X',
 'Compact Fan Scooter Headphones',
 'Eco Dock Air',
 'Thermostat Stove Lamp',
 'Mini Powerbank Lite Plus',
 'Scooter Bicycle Oven',
```

```
 'Wireless Light Heater Clock X',
 'Ultra Powerbank Scale Automatic',
 'Projector',
 'Blender',
 'Wireless Speakerphone Oven Advanced',
 'Iron Microphone',
 'Lock',
 'Automatic Speaker Wireless Plus Edge',
 'Eco Treadmill Plus Smart',
 'Smart Microphone Clock Ultra One',
 'Digital Router Scanner One',
 'Pro Stove Light',
 'Clean Mixer Ultra Eco',
 'Ultra Fan',
 'Ultra Drone Smart',
 'Smart Lock Mini Pro',
 'Rechargeable Keyboard Toaster Monitor Portable Fast',
 'Mini Mixer Freezer Trimmer Smart',
 'Heater Cooker',
 'Fast Shaver X One',
 'Wireless Webcam Bicycle Camera',
 'Clean Scale Speaker Powerbank Clean Wireless',
 'Advanced Fan Powerbank',
 'Pro Thermostat Speaker Touch Eco',
 'Fast Blender Treadmill Tablet One Air',
 'Automatic Iron Toaster Projector',
 'Smart Scale Stove',
 'Smart Webcam Mouse Digital',
 'Compact Drone Lite',
 'Digital Scale Wireless One Portable',
 'Clean Drone Thermostat Headphones',
 'Wireless Scooter',
 'Clean Light Advanced Smart Premium',
 'Compact Fan Iron Clock Digital',
 'Ultra Watch Eco',
 'Fast Lamp Scanner Vacuum',
 'Eco Stove Edge',
 'Fast Powerbank Radio Automatic Touch Ultra',
 'Pro Oven Scale',
 'Smart Printer Charger Brush Smart Ultra Touch',
 'Wireless Watch Premium Air',
 'Advanced Thermostat Microphone Scooter Clean Smart',
 'Pro Mouse Mini Rechargeable',
 'Silent Powerbank Radio Smart Plus',
 'Clean Speakerphone Edge Rechargeable Max',
```

```
df['Name'].value_counts()
```

Name	count
Fan	44
Dock	43
Freezer	34
Lock	32
Brush	27
...	...
Eco Grill Grill Portable	1
Mini Printer Lock Thermostat	1
Smart Grill Microphone Projector 360	1
Fast Powerbank Ultra	1
Smart Microphone X Portable Fast	1

7896 rows × 1 columns

dtype: int64

```
df['Description'].value_counts()
```

Description	count
Between key father half.	1
Part perform result newspaper.	1
Agreement ten responsibility whole maybe pull.	1
Who source better.	1
Expect public dinner.	1
...	...
Remain Congress blood plan voice.	1
Lawyer one than fire.	1
Quickly inside pull line lay start.	1
All movement yeah tax me.	1
Catch enough role nearly.	1

10000 rows × 1 columns

dtype: int64

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Index          10000 non-null  int64  
 1   Name           10000 non-null  object 
 2   Description    10000 non-null  object 
 3   Brand          10000 non-null  object 
 4   Category       10000 non-null  object 
 5   MRP_Price      10000 non-null  float64
 6   Currency       10000 non-null  object 
 7   Stock          10000 non-null  float64
 8   EAN            10000 non-null  float64
 9   Color          10000 non-null  object 
 10  Availability   10000 non-null  object 
 11  Internal ID    10000 non-null  int64  
 12  Qty            10000 non-null  float64
 13  Selling_price  10000 non-null  float64
dtypes: float64(5), int64(2), object(7)
memory usage: 1.1+ MB
```

```
# change datatype
```

```
df['Stock'] = pd.to_numeric(df['Stock'], errors='coerce').astype('Int64')
df['Qty'] = pd.to_numeric(df['Qty'], errors='coerce').astype('Int64')
```

```
cat_cols = ['Brand', 'Category', 'Color', 'Availability', 'Currency']

for col in cat_cols:
    df[col] = df[col].astype('category')
```

```
df['Name'] = df['Name'].astype(str)
df['Description'] = df['Description'].astype(str)
```

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

```
df['EAN'] = df['EAN'].astype('Int64').astype(str)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Index          10000 non-null  int64  
 1   Name           10000 non-null  object 
 2   Description    10000 non-null  object 
 3   Brand          10000 non-null  category
```

```
 4   Category       10000 non-null  category
 5   MRP_Price      10000 non-null  float64
 6   Currency       10000 non-null  category
 7   Stock          10000 non-null  Int64
 8   EAN            10000 non-null  object
 9   Color          10000 non-null  category
 10  Availability   10000 non-null  category
 11  Internal ID    10000 non-null  int64
 12  Qty            10000 non-null  Int64
 13  Selling_price  10000 non-null  float64
dtypes: Int64(2), category(5), float64(2), int64(2), object(3)
memory usage: 1.1+ MB
```

```
# column for category section
```

```
menu = {
    "Men's Clothing": "Apparel & Fashion",
    "Women's Clothing": "Apparel & Fashion",
    "Kids' Clothing": "Apparel & Fashion",
    "Shoes & Footwear": "Apparel & Fashion",
    "Clothing & Apparel": "Apparel & Fashion",
    "Accessories (Bags, Hats, Belts)": "Apparel & Fashion",

    "Beauty & Personal Care": "Beauty & Personal Care",
    "Skincare": "Beauty & Personal Care",
    "Makeup": "Beauty & Personal Care",
    "Haircare": "Beauty & Personal Care",
    "Fragrances": "Beauty & Personal Care",
    "Grooming Tools": "Beauty & Personal Care",

    "Home & Kitchen": "Home & Living",
    "Kitchen Appliances": "Home & Living",
    "Furniture": "Home & Living",
    "Home Decor": "Home & Living",
    "Bedding & Bath": "Home & Living",
    "Cleaning Supplies": "Home & Living",

    "Smartphones": "Mobiles & Electronics",
    "Laptops & Computers": "Mobiles & Electronics",
    "Smartwatches": "Mobiles & Electronics",
    "Headphones & Earbuds": "Mobiles & Electronics",
    "Cameras & Accessories": "Mobiles & Electronics",

    "Sports & Outdoors": "Sports, Fitness & Outdoors",
    "Fitness Equipment": "Sports, Fitness & Outdoors",
    "Cycling": "Sports, Fitness & Outdoors",
    "Camping & Hiking": "Sports, Fitness & Outdoors",
    "Fishing & Hunting": "Sports, Fitness & Outdoors",
    "Team Sports": "Sports, Fitness & Outdoors",

    "Toys & Games": "Toys, Books & Office",
    "Books & Stationery": "Toys, Books & Office",
    "Office Supplies": "Toys, Books & Office",

    "Automotive": "Automotive",
    "Health & Wellness": "Health & Wellness"
}

df["Category_Section"] = df["Category"].map(menu)
```

```
# column for availability
```

```
availability_map = {
    "in_stock": "Available",
    "limited_stock": "Available (Limited)",
    "pre_order": "Temporarily Unavailable",
    "backorder": "Temporarily Unavailable",
    "out_of_stock": "Unavailable",
    "discontinued": "Unavailable"
}

df["Availability_Status"] = df["Availability"].map(availability_map)
```

```
# df.sort_values('Availability_Status').head()
```

Show hidden output

```
df.info()
```

Show hidden output

```
# calculating discount
```

```
df['discount'] = df['MRP_Price'] - df['Selling_price']

df[['MRP_Price', 'Selling_price', 'discount']].describe()
```

	MRP_Price	Selling_price	discount
count	10000.000000	10000.000000	10000.000000
mean	503.328300	365.351640	137.976660
std	289.702596	224.066094	111.648617
min	1.000000	0.500000	-309.400000
25%	251.000000	177.120000	47.600000
50%	506.000000	353.400000	107.470000
75%	756.000000	529.540000	204.410000
max	999.000000	949.050000	609.600000

```
# renaming column name
```

```
df.rename(columns={'Selling_price': 'Selling Price'}, inplace=True)
df.rename(columns={'MRP_Price':'MRP Price'}, inplace=True)
df.rename(columns={'Category_Section':'Category Section'}, inplace=True)
df.rename(columns={'Availability_Status':'Availability Status'}, inplace=True)
```

```
# save clean dataset
```

```
df.to_csv('Products(clean dataset).csv', index=False)
```

```
df.shape
```

Show hidden output

```
df.describe(include='all').T
```

Show hidden output

```
df.duplicated().sum()
```

Show hidden output

```
#Analysing and providing the stats of each column.
```

```
#stats for Name column
name_stats = {
    'unique_values': df['Name'].nunique(),
    'missing_values': df['Name'].isnull().sum(),
    'avg_length': df['Name'].dropna().str.len().mean(),
    'min_length': df['Name'].dropna().str.len().min(),
    'max_length': df['Name'].dropna().str.len().max(),
    'most_frequent': df['Name'].mode().iloc[0]
}

name_stats
```

```
{'unique_values': 7896,
 'missing_values': np.int64(0),
 'avg_length': np.float64(26.1114),
 'min_length': 3,
 'max_length': 64,
 'most_frequent': 'Fan'}
```

```
name_stats_df = pd.DataFrame([name_stats], index=['Name'])
name_stats_df
```

	unique_values	missing_values	avg_length	min_length	max_length	most_frequent
Name	7896	0	26.1114	3	64	Fan

```
top_5_frequent= df['Name'].value_counts().head(5).to_dict()
top_5_frequent
```

```
{'Fan': 44, 'Dock': 43, 'Freezer': 34, 'Lock': 32, 'Brush': 27}
```

```
#stats for Description column
description_stats = {
    'unique_values': df['Description'].nunique(),
    'missing_values': df['Description'].isnull().sum(),
    'avg_length': df['Description'].dropna().str.len().mean(),
    'min_length': df['Description'].dropna().str.len().min(),
    'max_length': df['Description'].dropna().str.len().max(),
    'most_frequent': df['Description'].mode().iloc[0]
}

description_stats
```

```
{'unique_values': 10000,
 'missing_values': np.int64(0),
 'avg_length': np.float64(35.9708),
 'min_length': 11,
 'max_length': 78,
 'most_frequent': 'A beyond become company blue.'}
```

```
description_stats_stats_df = pd.DataFrame([description_stats], index=['Description'])
description_stats_stats_df
```

	unique_values	missing_values	avg_length	min_length	max_length	most_frequent
Description	10000	0	35.9708	11	78	A beyond become company blue.

```
#stats for Brand column
brand_stats = {
    'unique_values': df['Brand'].nunique(),
    'missing_values': df['Brand'].isnull().sum(),
    'avg_length': df['Brand'].dropna().str.len().mean(),
    'min_length': df['Brand'].dropna().str.len().min(),
    'max_length': df['Brand'].dropna().str.len().max(),
    'most_frequent': df['Brand'].mode().iloc[0]
}

brand_stats
```

```
{'unique_values': 9241,
 'missing_values': np.int64(0),
 'avg_length': np.float64(16.5369),
 'min_length': 5,
 'max_length': 35,
 'most_frequent': 'Kennedy LLC'}
```

```
brand_stats_df = pd.DataFrame([brand_stats], index=['Brand'])
brand_stats_df
```

	unique_values	missing_values	avg_length	min_length	max_length	most_frequent
Brand	9241	0	16.5369	5	35	Kennedy LLC

```
#stats for Category column

category_stats = {
    'unique_values': df['Category'].nunique(),
    'missing_values': df['Category'].isnull().sum(),
    'avg_length': df['Category'].dropna().str.len().mean(),
    'min_length': df['Category'].dropna().str.len().min(),
    'max_length': df['Category'].dropna().str.len().max(),
    'most_frequent': df['Category'].mode().iloc[0]
}
```

```
category_stats
```

```
{'unique_values': 34,
 'missing_values': np.int64(0),
 'avg_length': np.float64(14.6677),
 'min_length': 6,
 'max_length': 31,
 'most_frequent': 'Clothing & Apparel'}
```

```
category_stats_df = pd.DataFrame([category_stats], index=['Category'])
category_stats_df
```

	unique_values	missing_values	avg_length	min_length	max_length	most_frequent
Category	34	0	14.6677	6	31	Clothing & Apparel

```
# Top categories
df['Category'].value_counts().head(5)
```

Category	count
Clothing & Apparel	320
Kitchen Appliances	319
Home & Kitchen	316
Team Sports	316
Grooming Tools	311

dtype: int64

```
#stats for MRP_Price column
mrp_stats = {
    'unique_values': df['MRP_Price'].nunique(),
    'missing_values': df['MRP_Price'].isnull().sum(),
    'mean': df['MRP_Price'].mean(),
    'median': df['MRP_Price'].median(),
    'min': df['MRP_Price'].min(),
    'max': df['MRP_Price'].max(),
    'std_dev': df['MRP_Price'].std(),
    'skewness': df['MRP_Price'].skew()
}
mrp_stats
```

```
{'unique_values': 999,
 'missing_values': np.int64(0),
 'mean': np.float64(503.3283),
 'median': 506.0,
 'min': 1.0,
 'max': 999.0,
 'std_dev': 289.702595567417,
 'skewness': np.float64(-0.016483312382078688)}
```

```
num_stats_df = pd.DataFrame([mrp_stats], index=['MRP_Price'])
num_stats_df
```

	unique_values	missing_values	mean	median	min	max	std_dev	skewness
MRP_Price	999	0	503.3283	506.0	1.0	999.0	289.702596	-0.016483

```
# stats for Stock column
stock_stats = {
    'unique_values': df['Stock'].nunique(),
    'missing_values': df['Stock'].isnull().sum(),
    'mean': df['Stock'].mean(),
    'median': df['Stock'].median(),
    'min': df['Stock'].min(),
    'max': df['Stock'].max(),
    'std_dev': df['Stock'].std(),
    'zero_stock_count': (df['Stock'] == 0).sum()
}

stock_stats
```

```
{'unique_values': 1000,
 'missing_values': np.int64(0),
```

```
 'mean': np.float64(498.0426),
 'median': np.float64(499.0),
 'min': np.int64(0),
 'max': np.int64(999),
 'std_dev': np.float64(288.65579329356996),
 'zero_stock_count': np.int64(22)}
```

```
stock_stats_df = pd.DataFrame([stock_stats], index=['Stock'])
stock_stats_df
```

	unique_values	missing_values	mean	median	min	max	std_dev	zero_stock_count
Stock	1000	0	498.0426	499.0	0	999	288.655793	22

```
# stats for EAN column
ean_stats = {
    'unique_values': df['EAN'].nunique(),
    'missing_values': df['EAN'].isnull().sum(),
    'duplicate_rows': df['EAN'].duplicated(keep=False).sum()
}

ean_stats
```

```
{'unique_values': 9952,
 'missing_values': np.int64(0),
 'duplicate_rows': np.int64(96)}
```

```
ean_stats_df = pd.DataFrame([ean_stats], index=['EAN'])
ean_stats_df
```

	unique_values	missing_values	duplicate_rows
EAN	9952	0	96

```
#stats for color column
color_stats = {
    'unique_values': df['Color'].nunique(),
    'missing_values': df['Color'].isnull().sum(),
    'avg_length': df['Color'].dropna().str.len().mean(),
    'min_length': df['Color'].dropna().str.len().min(),
    'max_length': df['Color'].dropna().str.len().max(),
    'most_frequent': df['Color'].mode().iloc[0]
}


color_stats
```

```
{'unique_values': 140,
 'missing_values': np.int64(0),
 'avg_length': np.float64(8.9051),
 'min_length': 3,
 'max_length': 20,
 'most_frequent': 'Cornsilk'}
```

```
color_stats_df = pd.DataFrame([color_stats], index=['Color'])
color_stats_df
```

	unique_values	missing_values	avg_length	min_length	max_length	most_frequent
Color	140	0	8.9051	3	20	Cornsilk

```
# Check inconsistent naming
df['Color'].dropna().str.lower().value_counts().head(10)
```

Color	count
cornsilk	101
peru	93
antiquewhite	92
ivory	92
salmon	88
plum	88
darkslategray	87
maroon	85
teal	85
lightcoral	85

dtype: int64

```
#stats for Availability column
availability_stats = {
    'unique_values': df['Availability'].nunique(),
    'missing_values': df['Availability'].isnull().sum(),
    'avg_length': df['Availability'].dropna().str.len().mean(),
    'min_length': df['Availability'].dropna().str.len().min(),
    'max_length': df['Availability'].dropna().str.len().max(),
    'most_frequent': df['Availability'].mode().iloc[0]
}


availability_stats
```

```
{'unique_values': 6,
 'missing_values': np.int64(0),
 'avg_length': np.float64(10.515),
 'min_length': 8,
 'max_length': 13,
 'most_frequent': 'discontinued'}
```

```
availability_stats_df = pd.DataFrame([availability_stats], index=['Availability'])
availability_stats_df
```

	unique_values	missing_values	avg_length	min_length	max_length	most_frequent
Availability	6	0	10.515	8	13	discontinued

```
#stats for Internal ID
internal_id_stats = {
    'unique_values': df['Internal ID'].nunique(),
    'missing_values': df['Internal ID'].isnull().sum(),
    'duplicate_rows': df['Internal ID'].duplicated(keep=False).sum(),
}

internal_id_stats
```

```
{'unique_values': 99,
 'missing_values': np.int64(0),
 'duplicate_rows': np.int64(10000)}
```

```
internal_id_stats_df = pd.DataFrame([internal_id_stats], index=['Internal ID'])
internal_id_stats_df
```

	unique_values	missing_values	duplicate_rows
Internal ID	99	0	10000

```
# ststs for Qty column
qty_stats = {
    'unique_values': df['Qty'].nunique(),
    'missing_values': df['Qty'].isnull().sum(),
    'mean': df['Qty'].mean(),
    'median': df['Qty'].median(),
    'min': df['Qty'].min(),
    'max': df['Qty'].max(),
```

```
    'std_dev': df['Qty'].std(),
    'skewness': df['Qty'].skew()
}

qty_stats
```

```
{'unique_values': 952,
 'missing_values': np.int64(0),
 'mean': np.float64(523.2285),
 'median': np.float64(522.0),
 'min': np.int64(0),
 'max': np.int64(1000),
 'std_dev': np.float64(276.74841509261734),
 'skewness': np.float64(-0.0014358144305663615)}
```

```
qty_stats_df = pd.DataFrame([qty_stats], index=['Qty'])
qty_stats_df
```

	unique_values	missing_values	mean	median	min	max	std_dev	skewness
Qty	952	0	523.2285	522.0	0	1000	276.748415	-0.001436

```
# stats for Selling_Price column
selling_price_stats = {
    'unique_values': df['Selling_price'].nunique(),
    'missing_values': df['Selling_price'].isnull().sum(),
    'mean': df['Selling_price'].mean(),
    'median': df['Selling_price'].median(),
    'min': df['Selling_price'].min(),
    'max': df['Selling_price'].max(),
    'std_dev': df['Selling_price'].std()
}
selling_price_stats
```

```
{'unique_values': 7915,
 'missing_values': np.int64(0),
 'mean': np.float64(365.69653999999997),
 'median': 353.94500000000005,
 'min': 0.5,
 'max': 963.0,
 'std_dev': 224.39278000786976}
```

```
selling_price_stats_df = pd.DataFrame([selling_price_stats], index=['Selling_price'])
selling_price_stats_df
```

	unique_values	missing_values	mean	median	min	max	std_dev
Selling_price	7915	0	365.69654	353.945	0.5	963.0	224.39278

```
# stats for column Category_Section
category_section_stats = {
    'unique_values': df['Category_Section'].nunique(),
    'missing_values': df['Category_Section'].isnull().sum(),
    'avg_length': df['Category_Section'].astype(str).str.len().mean(),
    'min_length': df['Category_Section'].astype(str).str.len().min(),
    'max_length': df['Category_Section'].astype(str).str.len().max(),
    'most_frequent':  df['Category_Section'].mode().iloc[0]
}

category_section_stats
```

```
{'unique_values': 8,
 'missing_values': np.int64(0),
 'avg_length': np.float64(19.3901),
 'min_length': 10,
 'max_length': 26,
 'most_frequent': 'Home & Living'}
```

```
category_section_stats_df = pd.DataFrame([category_section_stats], index=['Category_Section'])
category_section_stats_df
```

Show hidden output

```
# stats for column Availability_Status
availability_status_stats = {
    'unique_values': df['Availability_Status'].nunique(),
    'missing_values': df['Availability_Status'].isnull().sum(),
    'avg_length': df['Availability_Status'].astype(str).str.len().mean(),
    'min_length': df['Availability_Status'].astype(str).str.len().min(),
    'max_length': df['Availability_Status'].astype(str).str.len().max(),
```

```
    'most_frequent':  df['Availability_Status'].mode().iloc[0]
}

availability_status_stats
```

```
{'unique_values': 4,
 'missing_values': np.int64(0),
 'avg_length': np.float64(15.9536),
 'min_length': 9,
 'max_length': 23,
 'most_frequent': 'Unavailable'}
```

```
availability_status_stats_df = pd.DataFrame([availability_status_stats], index=['Availability_Status'])
availability_status_stats_df
```

Show hidden output

```
(df['Selling price'] > df['MRP Price']).sum()
```