1. **Use a previously created dictionary to get an example list, in this case for the MEAT product family. Through this pdf we'll go through the loop:**

```
#initiate loop using family dictionary
for designation in family_dict.keys():

    #get list of items in ProdutDesignation
1   items = family_dict[designation]

    #use list to create dataframe with the corresponding values
2   items_df = df[df['ProductDesignation'].isin(items)]

    ##One hot encode get dummi variables for each Designation
3   one_hot_encoded = pd.get_dummies(items_df['ProductDesignation'])
    #
    ##Add 'OrderNumber','TotalAmount' columns
4   one_hot_encoded_items_df = pd.concat([items_df[[ 'OrderNumber','TotalAmount','Qty']], one_hot_encoded], axis=1)

    # iterate through each row in the dataframe
    for index, row in one_hot_encoded_items_df.iterrows():
        # iterate through each column in the row
5       for col in one_hot_encoded_items_df.columns:
            # check if the column is in the list and the value is 1
            if col in items and row[col] == 1:
                # change the value to the TotalAmount value on that row
                one_hot_encoded_items_df.at[index, col] = row['TotalAmount']
6               #take into account the number of products
                one_hot_encoded_items_df.at[index, col] = one_hot_encoded_items_df.at[index, col] * row['Qty']

    #THIS WILL CALCULATE THE AVERAGE AMOUNTY WHERE THE DIVISOR TAKES THE QUANTITIES ORDERED INTO A ACCOUNT
    #Describe the calculated df, and summing Amounts, describe wll generate mean, which equals the metric we're looking for
7/8 items_df_Qty_and_Amount_metric = one_hot_encoded_items_df.groupby('OrderNumber').sum().describe().T

    #loc by mean, sort values descending
9   items_Array_Qty_and_Amount_metric = items_df_Qty_and_Amount_metric['mean'].sort_values(ascending = False)

    #values below threshold are appended to list
10  Low_Relative_Demand_Products.extend(items_Array_Qty_and_Amount_metric[items_Array_Qty_and_Amount_metric < 1].index.tolist())


11 print('Low Relative Demand Products Represent',round(len(Low_Relative_Demand_Products)/len(df['ProductDesignation'].unique()),4)*100,'% of the existing Product Ra
```

```
Low Relative Demand Products Represent 63.92 % of the existing Product Range
```

```
family_dict['MEAT']
```

```
['SWEET SOUR CHICKEN',
 'LEMON CHICKEN',
 'CHICKEN CASHEW NUTS',
 'BABY PORK',
 'CHICKEN MIXED VEGETABLES',
 'TIN JIAN STYLE CHICKEN',
 'SWEET SOUR PORK',
 'FORTUNE PORK',
 'CHICKEN WITH MUSHROOMS',
 'CHICKEN CANTONESE STYLE',
 'CHICKEN GREEN PEPPERS',
 'VALENTINES BUFET',
 'SPECIAL BUFFET']
```

2. Get items df that contains observations with the with the list form dictionary created before in the notebook

```
items_df
```

	DocNumber	ProductDesignation	ProductFamily	Qty	TotalAmount	EmployeeID	IsDelivery	Pax	CustomerID	CustomerCity	Cus
2	TKD0320000012018	SWEET SOUR CHICKEN	MEAT	2	20.2	2	1	1	53908	LEYKOSIA	
3	TKD0320000012018	LEMON CHICKEN	MEAT	1	10.1	2	1	1	53908	LEYKOSIA	
17	TKD0320000022018	SWEET SOUR CHICKEN	MEAT	1	10.1	2	1	1	1072	LEYKOSIA	2
20	TKD0320000032018	LEMON CHICKEN	MEAT	1	10.1	2	1	1	45483	LAKATAMEIA	1
21	TKD0320000032018	CHICKEN CASHEW NUTS	MEAT	1	10.1	2	1	1	45483	LAKATAMEIA	1
...	...	...	...	...	...	...	...	...	...	...	
84070	TK0110054622018	BABY PORK	MEAT	1	17.7	2	0	4	0	<NA>	
84082	TK0110054632018	SWEET SOUR CHICKEN	MEAT	1	12.6	2	0	7	0	<NA>	
84084	TK0110054632018	BABY PORK	MEAT	1	17.7	2	0	7	0	<NA>	
84097	TK0110054642018	LEMON CHICKEN	MEAT	1	12.6	2	0	9	0	<NA>	
84104	TK0110054642018	SWEET SOUR CHICKEN	MEAT	1	12.6	2	0	9	0	<NA>	

3. Get df with Dummy Variables that correspond to unique values of Product Designation

one_hot_encoded

	BABY PORK	CHICKEN CANTONESE STYLE	CHICKEN CASHEW NUTS	CHICKEN GREEN PEPPERS	CHICKEN MIXED VEGETABLES	CHICKEN WITH MUSHROOMS	FORTUNE PORK	LEMON CHICKEN	SPECIAL BUFFET	SWEET SOUR CHICKEN	SWEET SOUR PORK	TIN JIAN STYLE CHICKEN
2	0	0	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0	1	0	0	0	0
17	0	0	0	0	0	0	0	0	0	1	0	0
20	0	0	0	0	0	0	0	1	0	0	0	0
21	0	0	1	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
84070	1	0	0	0	0	0	0	0	0	0	0	0
84082	0	0	0	0	0	0	0	0	0	1	0	0
84084	1	0	0	0	0	0	0	0	0	0	0	0
84097	0	0	0	0	0	0	0	1	0	0	0	0
84104	0	0	0	0	0	0	0	0	0	1	0	0

4. Add OrderNumber, TotalAmount, and Qty columns to df generated in #3

one_hot_encoded_items_df

	OrderNumber	TotalAmount	Qty	BABY PORK	CHICKEN CANTONESE STYLE	CHICKEN CASHEW NUTS	CHICKEN GREEN PEPPERS	CHICKEN MIXED VEGETABLES	CHICKEN WITH MUSHROOMS	FORTUNE PORK	LEMON CHICKEN	SPECIAL BUFFET	SWEET SOUR CHICKEN	SWEET SOUR PORK	TIN JIAN STYLE CHICKEN
2	1	20.2	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	40.4	0.0	0.0
3	1	10.1	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	10.1	0	0.0	0.0	0.0
17	2	10.1	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	10.1	0.0	0.0
20	3	10.1	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	10.1	0	0.0	0.0	0.0
21	3	10.1	1	0.0	0.0	10.1	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
84070	11145	17.7	1	17.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0
84082	11146	12.6	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	12.6	0.0	0.0
84084	11146	17.7	1	17.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0
84097	11147	12.6	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	12.6	0	0.0	0.0	0.0
84104	11147	12.6	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	12.6	0.0	0.0

5. With this loop we iterate with indexes and rows through #4 df. for each column we check if the value is 1 and use the index and row for next step (see output on image)

```
5]: # iterate through each row in the dataframe
    for index, row in one_hot_encoded_items_df.iterrows():
         #iterate through each column in the row
        for col in one_hot_encoded_items_df.columns:
            # check if the column is in the list and the value is 1
            if col in items and row[col] == 1:
                print(index, col)
```

```
2 SWEET SOUR CHICKEN
3 LEMON CHICKEN
17 SWEET SOUR CHICKEN
20 LEMON CHICKEN
21 CHICKEN CASHEW NUTS
42 SWEET SOUR CHICKEN
55 CHICKEN CASHEW NUTS
58 LEMON CHICKEN
65 LEMON CHICKEN
69 SWEET SOUR CHICKEN
75 BABY PORK
76 CHICKEN MIXED VEGETABLES
90 SWEET SOUR CHICKEN
99 LEMON CHICKEN
103 SWEET SOUR CHICKEN
108 LEMON CHICKEN
119 LEMON CHICKEN
128 SWEET SOUR CHICKEN
138 SWEET SOUR CHICKEN
149 SWEET SOUR CHICKEN
164 SWEET SOUR CHICKEN
165 BABY PORK
178 LEMON CHICKEN
179 SWEET SOUR CHICKEN
185 SWEET SOUR CHICKEN
210 LEMON CHICKEN
214 SWEET SOUR CHICKEN
219 TIN JIAN STYLE CHICKEN
249 CHICKEN MIXED VEGETABLES
```

6. At the end of the loop, if the value of the cell is 1, we change it to the Total Amount value 1st print is targetted cell before, 2nd is after =TotalAmount)

8561 rows × 16 columns

[86]:
```
# iterate through each row in the dataframe
for index, row in one_hot_encoded_items_df.iterrows():
    # iterate through each column in the row
    for col in one_hot_encoded_items_df.columns:
        # check if the column is in the list and the value is 1
        if col in items and row[col] == 1:
            print(one_hot_encoded_items_df.at[index, col])
            # change the value to the TotalAmount value on that row
            one_hot_encoded_items_df.at[index, col] = row['TotalAmount']
            #take into account the number of products
            one_hot_encoded_items_df.at[index, col] = one_hot_encoded_items_df.at[index, col] * row['Qty']
            print(one_hot_encoded_items_df.at[index, col])
```

```
1
40.4
1
10.1
1.0
10.1
1.0
10.1
1
10.1
1.0
10.1
1.0
10.1
```

7. Now we group by Order Amount and sum the values so we aggregate rows that represent the same order into 1, with all the amount spend on each product

```
#THIS WILL CALCULATE THE AVERAGE AMOUNTY WHERE THE DIVISOR TAKES THE QUANTITIES ORDERED INTO A ACCOUNT
#Describe the calculated df, and summing Amounts, describe wll generate mean, which equals the metric we're looking for
items_df_Qty_and_Amount_metric = one_hot_encoded_items_df.groupby('OrderNumber').sum().describe().T
viz1 = one_hot_encoded_items_df.groupby('OrderNumber').sum()
viz1
```

OrderNumber	TotalAmount	Qty	BABY PORK	CHICKEN CANTONESE STYLE	CHICKEN CASHEW NUTS	CHICKEN GREEN PEPPERS	CHICKEN MIXED VEGETABLES	CHICKEN WITH MUSHROOMS	FORTUNE PORK	LEMON CHICKEN	SPECIAL BUFFET	SWEET SOUR CHICKEN	SWEET SOUR PORK	TIN JIAN STYLE CHICKEN	V
1	30.3	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	10.1	0	40.4	0.0	0.0	
2	10.1	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	10.1	0.0	0.0	
3	20.2	2	0.0	0.0	10.1	0.0	0.0	0.0	0.0	10.1	0	0.0	0.0	0.0	
6	10.1	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	10.1	0.0	0.0	
8	10.1	1	0.0	0.0	10.1	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
11143	30.3	2	17.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	12.6	0.0	
11144	37.8	3	0.0	0.0	12.6	0.0	0.0	0.0	12.6	0.0	0	0.0	12.6	0.0	
11145	30.3	2	17.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	12.6	0.0	0.0	
11146	30.3	2	17.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	12.6	0.0	0.0	
11147	25.2	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	12.6	0	12.6	0.0	0.0	

6010 rows × 15 columns

8. By using describe() we obtain all metrics but we want to focus on the mean later since it's a result of the following equation:

```
#THIS WILL CALCULATE THE AVERAGE AMOUNTY WHERE THE DIVISOR TAKES THE QUANTITIES ORDERED INTO A ACCOUNT
#Describe the calculated df, and summing Amounts, describe wll generate mean, which equals the metric we'r
items_df_Qty_and_Amount_metric = one_hot_encoded_items_df.groupby('OrderNumber').sum().describe().T
viz1 = one_hot_encoded_items_df.groupby('OrderNumber').sum().describe().T
viz1
```

	count	mean	std	min	25%	50%	75%	max
TotalAmount	6010.0	18.973045	14.524211	6.3	10.1	12.6	25.2	450.0
Qty	6010.0	1.535108	0.850743	1.0	1.0	1.0	2.0	18.0
BABY PORK	6010.0	2.732130	7.867925	0.0	0.0	0.0	0.0	159.3
CHICKEN CANTONESE STYLE	6010.0	0.042230	0.888779	0.0	0.0	0.0	0.0	50.4
CHICKEN CASHEW NUTS	6010.0	1.367854	4.312608	0.0	0.0	0.0	0.0	90.9
CHICKEN GREEN PEPPERS	6010.0	0.175807	2.096637	0.0	0.0	0.0	0.0	113.4
CHICKEN MIXED VEGETABLES	6010.0	1.126656	4.265158	0.0	0.0	0.0	0.0	113.4
CHICKEN WITH MUSHROOMS	6010.0	0.161780	1.582088	0.0	0.0	0.0	0.0	50.4
FORTUNE PORK	6010.0	0.436373	3.040215	0.0	0.0	0.0	0.0	86.4
LEMON CHICKEN	6010.0	2.668869	9.810107	0.0	0.0	0.0	0.0	617.4
SPECIAL BUFFET	6010.0	0.410483	8.012454	0.0	0.0	0.0	0.0	225.0
SWEET SOUR CHICKEN	6010.0	8.674526	11.957727	0.0	0.0	10.1	12.6	201.6
SWEET SOUR PORK	6010.0	1.979135	6.160717	0.0	0.0	0.0	0.0	90.9
TIN JIAN STYLE CHICKEN	6010.0	0.644892	2.970831	0.0	0.0	0.0	0.0	50.4
VALENTINES BUFET	6010.0	1.063062	11.474422	0.0	0.0	0.0	0.0	225.0

Mean of Product Designation:

$$\frac{\sum \text{(Total Amount)}}{\text{Number Of Orders}}$$

Product's Demand Relative to Family via Frequency & Amount Spend

9. Now we produce an Array off of the Mean column of the describe () df, sort it by descending values

8561 rows × 16 columns

[94]:
```
#THIS WILL CALCULATE THE AVERAGE AMOUNTY WHERE THE DIVISOR TAKES THE QUANTITIES ORDERED INTO A ACCOUNT
#Describe the calculated df, and summing Amounts, describe wll generate mean, which equals the metric we're looking
items_df_Qty_and_Amount_metric = one_hot_encoded_items_df.groupby('OrderNumber').sum().describe().T

#loc by mean, sort values descending
items_Array_Qty_and_Amount_metric = items_df_Qty_and_Amount_metric['mean'].sort_values(ascending = False)
items_Array_Qty_and_Amount_metric
```

[94]:
```
TotalAmount                 18.973045
SWEET SOUR CHICKEN           8.674526
BABY PORK                    2.732130
LEMON CHICKEN                2.668869
SWEET SOUR PORK              1.979135
Qty                          1.535108
CHICKEN CASHEW NUTS          1.367854
CHICKEN MIXED VEGETABLES     1.126656
VALENTINES BUFET             1.063062
TIN JIAN STYLE CHICKEN       0.644892
FORTUNE PORK                 0.436373
SPECIAL BUFFET               0.410483
CHICKEN GREEN PEPPERS        0.175807
CHICKEN WITH MUSHROOMS       0.161780
CHICKEN CANTONESE STYLE      0.042230
Name: mean, dtype: float64
```

10. With a Threshold of 1, meaning the Amount are higher or equal to the Number of Orders we dictate what products are associated to inefficiency due to Low Demand Relative to the Family. We then extend the initial list, as we go through all Product Families and add each items.

```
5]: #values below threshold are appended to list
    Low_Relative_Demand_Products.extend(items_Array_Qty_and_Amount_metric[items_Array_Qty_and_Amount_metric < 1].index.tolist())
    items_Array_Qty_and_Amount_metric[items_Array_Qty_and_Amount_metric < 0.75].index.tolist()

6]: ['TIN JIAN STYLE CHICKEN',
     'FORTUNE PORK',
     'SPECIAL BUFFET',
     'CHICKEN GREEN PEPPERS',
     'CHICKEN WITH MUSHROOMS',
     'CHICKEN CANTONESE STYLE']
```

11. By this logic we remove 64% of the products, here's a sample of the resulting list

```
print('Low Relative Demand Products Represent',round(len(Low_Relative_Demand_Products)/len(df['ProductDesignation'].unique()),4)*100,'% of the existing Product Ra
Low Relative Demand Products Represent 63.92 % of the existing Product Range

[98]: Low_Relative_Demand_Products

[98]: ['WHOLE DUCK',
       'CHICKEN SATAY',
       'SEAWEED',
       'PECKING STYLE PRAWNS',
       'BBQ SPARE RIBS',
       'SELCTION FOR  TWO',
       'PRAWN CRACKERS',
       'CRISPY AROMATIC DUCK',
       'TIN JIAN STYLE CHICKEN',
       'FORTUNE PORK',
       'SPECIAL BUFFET',
       'CHICKEN GREEN PEPPERS',
       'CHICKEN WITH MUSHROOMS',
       'CHICKEN CANTONESE STYLE',
       'CHICKEN OYSTER SAUCE',
       'BEEF WITH GINGER',
       'PRAWNS WITH GINGER',
       'PORK BBS',
       'CHICKEN WITH GINGER',
       'PORK OYSTER SAUCE',
       'PORK WITH GINGER',
       'CURRY',
       'VEG RICE',
       'SHRIMP FRIED RICE',
       'SPICY RICE',
       'SPICY NOODLES',
       'CHOW MEIN CHICKEN',
       'CHOW MEIN BEEF',
       'TotalAmount',
       ' NO BABY CORN',
       ' NO BEANSPROUTS',
       ' NO CHILLY',
       ' NO EGG',
       ' NO MEAT',
       ' NO MUSHROOMS',
       ' NO ONION',
       ' NO PORK',
       ' NO PRAWNS',
       ' NO SAUCE',
```