Шаг 1.
Импорт базовых библиотек

```
#Импорт базовых библиотек
import numpy as np
import pandas as pd
#Импорт библиотек визуализации
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
#Импорт библиотек обработки текста
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
df = pd.read_csv('Products.csv', index_col='index')
print(df.shape)
print(df.head())
```

Результат кода:

```
(27555, 9)
                                                product  ...                                        description
index                                                    ...
1                    Garlic Oil - Vegetarian Capsule 500 mg  ...  This Product contains Garlic Oil that is known...
2                                   Water Bottle - Orange  ...  Each product is microwave safe (without lid), ...
3                              Brass Angle Deep - Plain, No.2  ...  A perfect gift for all occasions, be it your m...
4      Cereal Flip Lid Container/Storage Jar - Assort...  ...  Multipurpose container with an attractive desi...
5                         Creme Soft Soap - For Hands & Body  ...  Nivea Creme Soft Soap gives your skin the best...

[5 rows x 9 columns]
```

Шаг 2. Проверка наличия отсутствующих или недостающих данных:

```
print('Количество пропущенных данных в каждом столбце')
print('-'*30)
print(df.isnull().sum())
print('-'*30)
print('Пропущенные данные в % в каждом столбце')
print('-'*30)
for col in df.columns:
    null_count = df[col].isnull().sum()
    total_count = df.shape[0]
    print("{} : {:.2f}".format(col,null_count/total_count * 100))
```

Результат:

```
Количество пропущенных данных в каждом столбце
------------------------------
product            1
category           0
sub_category       0
brand              1
sale_price         0
market_price       0
type               0
rating          8626
description      115
dtype: int64
------------------------------
Пропущенные данные в % в каждом столбце
------------------------------
product : 0.00
category : 0.00
sub_category : 0.00
brand : 0.00
sale_price : 0.00
market_price : 0.00
type : 0.00
rating : 31.30
description : 0.42
```

Таким образом: 1) есть товары без названия; 2) есть товары без бренда; 3) 115 товаров не имеют описания; 4) 8626 товаров не имеют рейтингов.

Вышеперечисленные особенности важны для построения рекомендательной системы. Отбросим строки из данных, которые содержат отсутствующие значения:

```
df = df.dropna()
print(df.shape)
```

В результате получим:

```
(18840, 9)
```

Получим общие сведения о типах данных столбцов:

```
print(df.dtypes)
```

Результат:

```
product          object
category         object
sub_category     object
brand            object
sale_price      float64
market_price    float64
type             object
rating          float64
description      object
```

Таким образом, признаки sale_price, market_price и rating являются числовыми (поскольку они представлены с помощью float64). Все остальное — это строковые объекты (представленные в виде объектов).

Шаг 3. Одномерный анализ данных

Рассмотрим распределение столбцов категорий. Для этого наберем следующий код:

```
counts = df['category'].value_counts()
count_percentage = df['category'].value_counts(1)*100
counts_df = pd.DataFrame({'Category': counts.index, 'Counts':counts.values,
                          'Percent': np.round(count_percentage.values, 2)})
display(counts_df)
pio.renderers.default = 'png'
fig = go.Figure(px.bar(data_frame=counts_df,
 x='Category',
 y='Counts',
 color='Counts',
 color_continuous_scale='blues',
 text_auto=True,
 title=f'Число вещей в каждой категории'))
fig.show()
```

Получим следующий результат:

	Category	Counts	Percent
0	Beauty & Hygiene	5460	28.98
1	Kitchen, Garden & Pets	2494	13.24
2	Snacks & Branded Foods	2468	13.10
3	Gourmet & World Food	2364	12.55
4	Foodgrains, Oil & Masala	2173	11.53
5	Cleaning & Household	2091	11.10
6	Bakery, Cakes & Dairy	665	3.53
7	Beverages	630	3.34
8	Baby Care	495	2.63

Отсюда можно сделать следующие выводы:

Красота и гигиена насчитывают в общей сложности 5460 товаров. Он охватывает 28,98% от общего продуктового портфеля.

Далее идет лучшая категория «Кухня, сад и домашние животные», в которой насчитывается 2494 товара. Она охватывает 13,24% от общего продуктового портфеля.

Категория «Уход за младенцами» имеет наименьшее количество продуктов - 495 продуктов. Она охватывает 2,63% от общего продуктового портфеля.

Теперь рассмотрим распределение товаров по брендам.

```
column = 'brand'
counts = df[column].value_counts()
count_percentage = df[column].value_counts(1)*100
counts_df = pd.DataFrame({column:counts.index,'Counts':counts.values,
                          'Percent':np.round(count_percentage.values,2)})
print('Уникальные '+str(column)+' Количество = ',df['sub_category'].nunique())
print('Топ 10 брендов')
display(counts_df.head(10))
print('Самые худшие топ 10 брендов')
display(counts_df.tail(10))
```

Получим следующие результаты:

```
Уникальные brand Количество =  77
Топ 10 брендов
            brand  Counts  Percent
0        bb Royal     278     1.48
1         BB Home     172     0.91
2            Amul     153     0.81
3        Himalaya     139     0.74
4           Cello     104     0.55
5        BIOTIQUE     103     0.55
6              DP     101     0.54
7            Keya     101     0.54
8  Organic Tattva      99     0.53
9             MTR      97     0.51
Самые худшие топ 10 брендов
               brand  Counts  Percent
1923            Yoni       1     0.01
1924        Vochelle       1     0.01
1925  Ancient Living       1     0.01
1926        Goodluck       1     0.01
1927             NNF       1     0.01
1928          Bachun       1     0.01
1929        Dhishoom       1     0.01
1930   Double Pagoda       1     0.01
1931     Muscleblaze       1     0.01
1932            Mamy       1     0.01
```

Шаг 4. Создадим новые функции, которые помогут улучшить рекомендации. У нас есть sale_price и market_price. Создадим функцию discount% по следующей формуле:

discount% = [(market_price – sale_price)/sale_price] * 100

```
df['discount'] = (df['market_price']-df['sale_price'])*100/df['market_price']
print(df['discount'].describe())
print(pd.cut(df.discount,bins = [-1,0,10,20,30,40,50,60,80,90,100]).reset_index().groupby(['discount']).size())
```

Получим:

discount	
(-1, 0]	8157
(0, 10]	3125
(10, 20]	2962
(20, 30]	2435
(30, 40]	1126
(40, 50]	700
(50, 60]	223
(60, 80]	108
(80, 90]	4
(90, 100]	0

Шаг 5. Построение простой системы рекомендаций

Шаг 5.1 Очистим несколько строковых столбцов

Колонка категорий содержит «Кухня, сад и домашние животные». Заменим его на следующие столбцы [кухня, сад, домашние животные]. Аналогичная трансформация будет применена к sub_category, type и brand.

Теперь давайте создадим одну функцию (product_classification_features), которая добавляет все вышеуказанные очищенные столбцы.

```
df2 = df.copy()
rmv_spc = lambda a:a.strip()
get_list = lambda a:list(map(rmv_spc,re.split(r'[& |, |* |\n-]', a)))
for col in ['category', 'sub_category', 'type']:
    df2[col] = df2[col].apply(get_list)
def cleaner(x):
    if isinstance(x, list):
        return [str.lower(i.replace(" ", "")) for i in x]
    else:
        if isinstance(x, str):
            return str.lower(x.replace(" ", ""))
        else:
            return ''
for col in ['category', 'sub_category', 'type','brand']:
    df2[col] = df2[col].apply(cleaner)
def couple(x):
    return ' '.join(x['category']) + ' ' + ' '.join(x['sub_category']) + ' '+x['brand']+' ' +' '.join(x['type'])
df2['product_classification_features'] = df2.apply(couple, axis=1)
```

Шаг 5.2. Построим простую логику рекомендаций на основе популярности. Мы будем использовать рекомендательные признаки – type, category или sub_category. Напишем функцию, возвращающую самый популярный товар или товар с самым высоким рейтингом:

```
def recommend_most_popular(col,col_value,top_n=5):
    return df[df[col]==col_value].sort_values(by='rating',ascending = False).head(top_n)[['product',col,'rating']]

print(recommend_most_popular(col='category',col_value='Beauty & Hygiene'))
```

Получим:

index	product	...	rating
20164	Supreme Scalp Rejuvenation Shampoo	...	5.0
5472	Vitamin C Brightening Day Cream With SPF 30 UV...	...	5.0
20936	Exfoliating Face Scrub	...	5.0
5499	Prickly Heat Powder - Cool Chandan With Sandal...	...	5.0
20860	Wheat Grass Powder	...	5.0

Шаг 6. Создадим систему рекомендаций на основе контента, то есть характеристик продуктов, которые мы хотим рекомендовать.

Мы будем использовать CountVectorizer для создания пространства признаков, с помощью которого будем определять количество вхождений поискового запроса пользователя интернет-магазина.

После этого (получения вектора признаков) будем использовать метрики сходства для определения и рекомендации похожих продуктов.

Есть 4 метрики: Евклидово расстояние, **манхэттенское расстояние (**измеряет абсолютную разницу между двумя векторами), **расстояние Жаккара (**измеряет соотношение общих слов между 2 предложениями к общему количеству уникальных слов в этих 2 предложениях) и **Cosine Distance (или косинусное сходство).**

Косинусное сходство измеряет угол между 2 векторами. Косинусное сходство может давать значение от -1 до +1. Значение -1 означает, что продукты противоположны или непохожи, а значение +1 означает, что 2 продукта одинаковы.

Рассчитаем косинусное сходство product_classification_features для всех изделий.

```
count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(df2['product_classification_features'])
cosine_sim = cosine_similarity(count_matrix, count_matrix)
cosine_sim_df = pd.DataFrame(cosine_sim)
```

Приведенная выше матрица содержит косинусное сходство каждого товара по сравнению с остальными товарами в каталоге. Теперь создадим рекомендатель, используя косинусное сходство

```
def content_recommendation_v1(title):
    a = df2.copy().reset_index().drop('index',axis=1)
    index = a[a['product']==title].index[0]
    top_n_index = list(cosine_sim_df[index].nlargest(10).index)
    try:
        top_n_index.remove(index)
    except:
        pass
    similar_df = a.iloc[top_n_index][['product']]
    similar_df['cosine_similarity'] = cosine_sim_df[index].iloc[top_n_index]
    return similar_df
```

Протестируем на темном шоколаде:

```
title = 'Dark Chocolate'
print(content_recommendation_v1(title).head)
```

Результат:

```
<bound method NDFrame.head of                                                product  cosine_similarity
105                   I Love You Fruit N Nut Chocolate              1.0
163                  Dark Chocolate- 55% Rich In Cocoa              1.0
1144  Choco Cracker - Magical Crystal With Milk Choc...              1.0
1718             Dark Chocolate - Single Origin, India              1.0
2517     Fruit N Nut, Dark Chocolate- 55% Rich In Cocoa              1.0
3117  Colombia Classique Black, Single Origin Dark C...              1.0
4013  Almondo - Roasted Almonds Coated With Milk Cho...              1.0
4358                          Sugar Free Dark Chocolate              1.0
5867                        Milk Compound Slab - MCO-11              1.0>
```

Шаг 7. Улучшение модели рекомендаций

Можно донастроить алгоритм. Для этого воспользуемся столбцом product, чтобы создать еще одно косинусное сходство. Возьмем среднее значение сходства косинуса и посмотрим, будут ли результаты лучше.

```
count2 = CountVectorizer(stop_words='english', lowercase=True)
count_matrix2 = count2.fit_transform(df2['product'])
cosine_sim2 = cosine_similarity(count_matrix2, count_matrix2)
cosine_sim_df2 = pd.DataFrame(cosine_sim2)
def content_recommendation_v2(title):
    a = df2.copy().reset_index().drop('index', axis=1)
    index = a[a['product'] == title].index[0]
    similar_basis_metric_1 = cosine_sim_df[cosine_sim_df[index]>0][index].reset_index().rename(columns={index:'sim_1'})
    similar_basis_metric_2 = cosine_sim_df2[cosine_sim_df2[index]>0][index].reset_index().rename(columns={index:'sim_2'})
    similar_df = similar_basis_metric_1.merge(similar_basis_metric_2,how='left').merge(a[['product']].reset_index(),how='left')
    similar_df['sim'] = similar_df[['sim_1','sim_2']].fillna(0).mean(axis=1)
    similar_df = similar_df[similar_df['index']!=index].sort_values(by='sim',ascending=False)
    return similar_df[['product', 'sim']].head(10)
```

Протестируем на том же темном шоколаде.

```
title = 'Dark Chocolate'
print(content_recommendation_v2(title).head)
```

Результат:

```
<bound method NDFrame.head of                                                product       sim
1423               Prebiotic Chocolate - Dark Chocolate  0.868207
906                           Sugar Free Dark Chocolate  0.853553
2952                         Best Wishes Dark Chocolate  0.853553
914          Prebiotic Chocolate - Dark Chocolate Ginger  0.836086
2898  Peru Dark Amazon, Single Origin Dark Chocolate...  0.835410
3353  Tanzania Chocolat Noir, Single Origin Dark Cho...  0.819801
646   Colombia Classique Black, Single Origin Dark C...  0.819801
1964  Madagascar Noir De Cacao, Single Origin Dark C...  0.819801
31                    Dark Chocolate- 55% Rich In Cocoa  0.816228
365               Dark Chocolate - Single Origin, India  0.816228>
```