MEDTOUREASY

HARSHINI D ALICE

PROJECT REPORT
OF
INTERNSHIP PROGRAM

ANALYSIS OF CHEMICAL COMPONENTS IN COSMETICS

2024

ACKNOWLDEGMENT

The traineeship opportunity that I had with MedTourEasy was a great change for learning and understanding the intricacies of the subject of Data Visualizations in Data Analytics; and also, for personal as well as professional development. I am very obliged for having a chance to interact with so many professionals who guided me throughout the traineeship project and made it a great learning curve for me.

Firstly, I express my deepest gratitude and special thanks to the Training & Development Team of MedTourEasy who gave me an opportunity to carry out my traineeship at their esteemed organization. Also, I express my thanks to the team for making me understand the details of the Data Analytics profile and training me in the same so that I can carry out the project properly and with maximum client satisfaction and also for spearing his valuable time in spite of his busy schedule.

I would also like to thank the team of MedTourEasy and my colleagues who made the working environment productive and very conducive.

Table of CONTENTS

1 Introduction

1.1 About the Company

MedTourEasy improves access to healthcare for people everywhere. It is an easy to use platform and service that helps patients to get medical second opinions and to schedule affordable, high quality medical treatment abroad.

MedTourEasy commitment to quality and transparency in healthcare is core to our mission. At MedTourEasy, we integrate the same three factors that physicians themselves agree are most important when selecting or referring a healthcare provider (patient satisfaction, experience match, and the quality of the hospital where a physician provides care.

1.2 Project Overview

This project develops an interactive recommendation system to suggest cosmetics tailored to individual needs, particularly for sensitive skin

1.3 Project Objectives

- Create a recommendation system based on product ingredients.
- Visualize ingredient similarities.
- Provide an engaging user interface.

2. Methodology

2.1 Project Flow

- Data Collection: Gather product data.
- Data Cleaning: Process and standardize data.
- Data Analysis: Tokenize ingredients and build a matrix.
- Modeling: Apply t-SNE for dimensionality reduction.
- Visualization: Use Bokeh for interactive plots

.

2.2 Tools and Technologies

- Languages: Python
- Libraries: Pandas, NumPy, scikit-learn, Bokeh
- Environment: Jupyter Notebook , Idle.

.

3. Implementation

3.1 Problem Definition

The goal is to assist users in selecting cosmetics based on skin type and personal preferences.

3.2 Data Collection

Data sourced from Cosmetic.csv includes product names, brands, and ingredients

3.3 Data Cleaning

Handled missing values and standardized ingredient names.

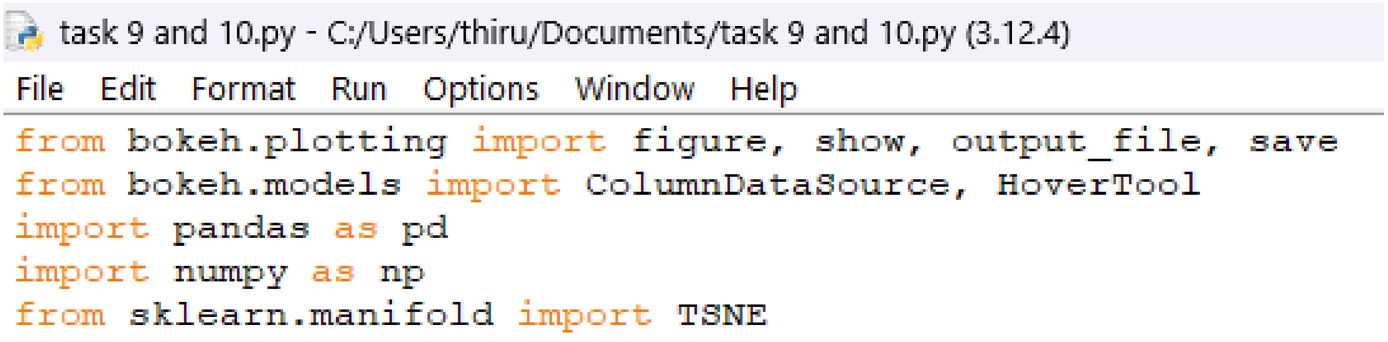
3.4 Data Filtering

Filtered to focus on moisturizers for dry skin and tasks given

4. Visualization and Analysis

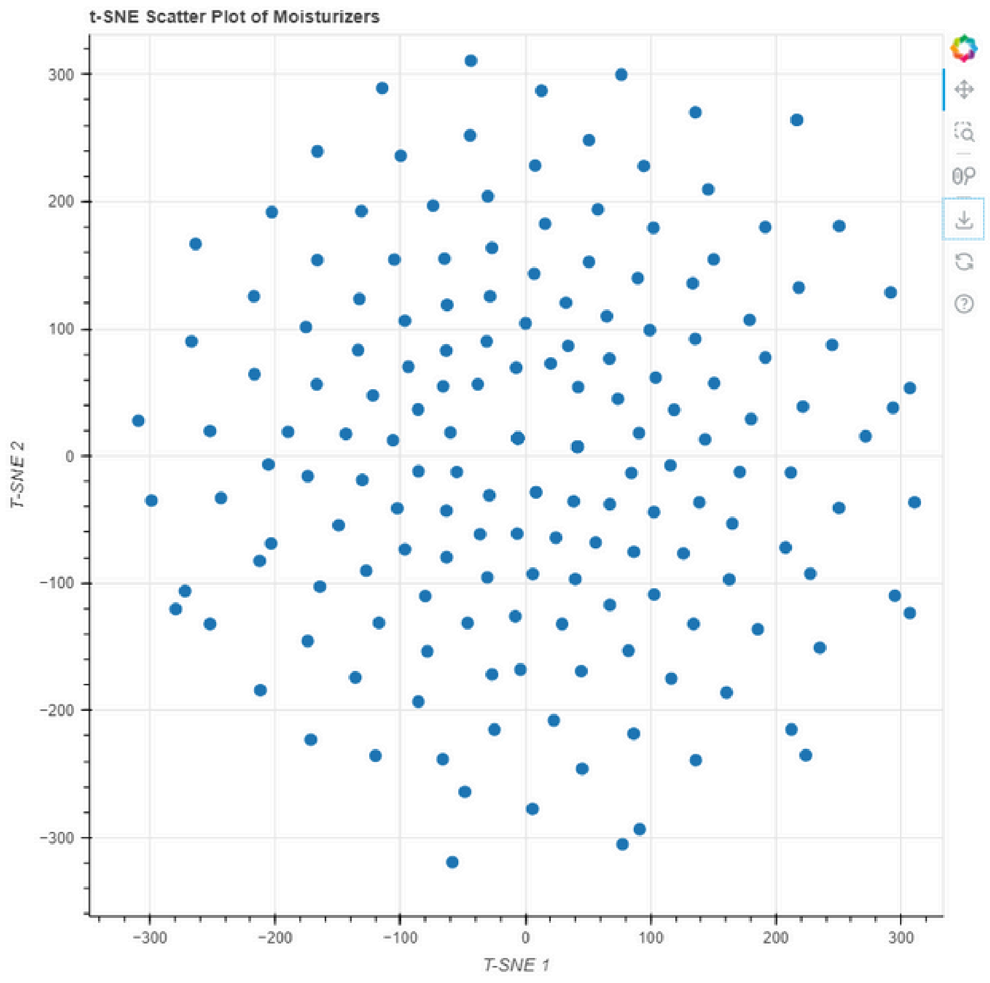
4.1 Data Visualization

Used t-SNE to visualize ingredient similarities and created scatter plots to display product distributions.

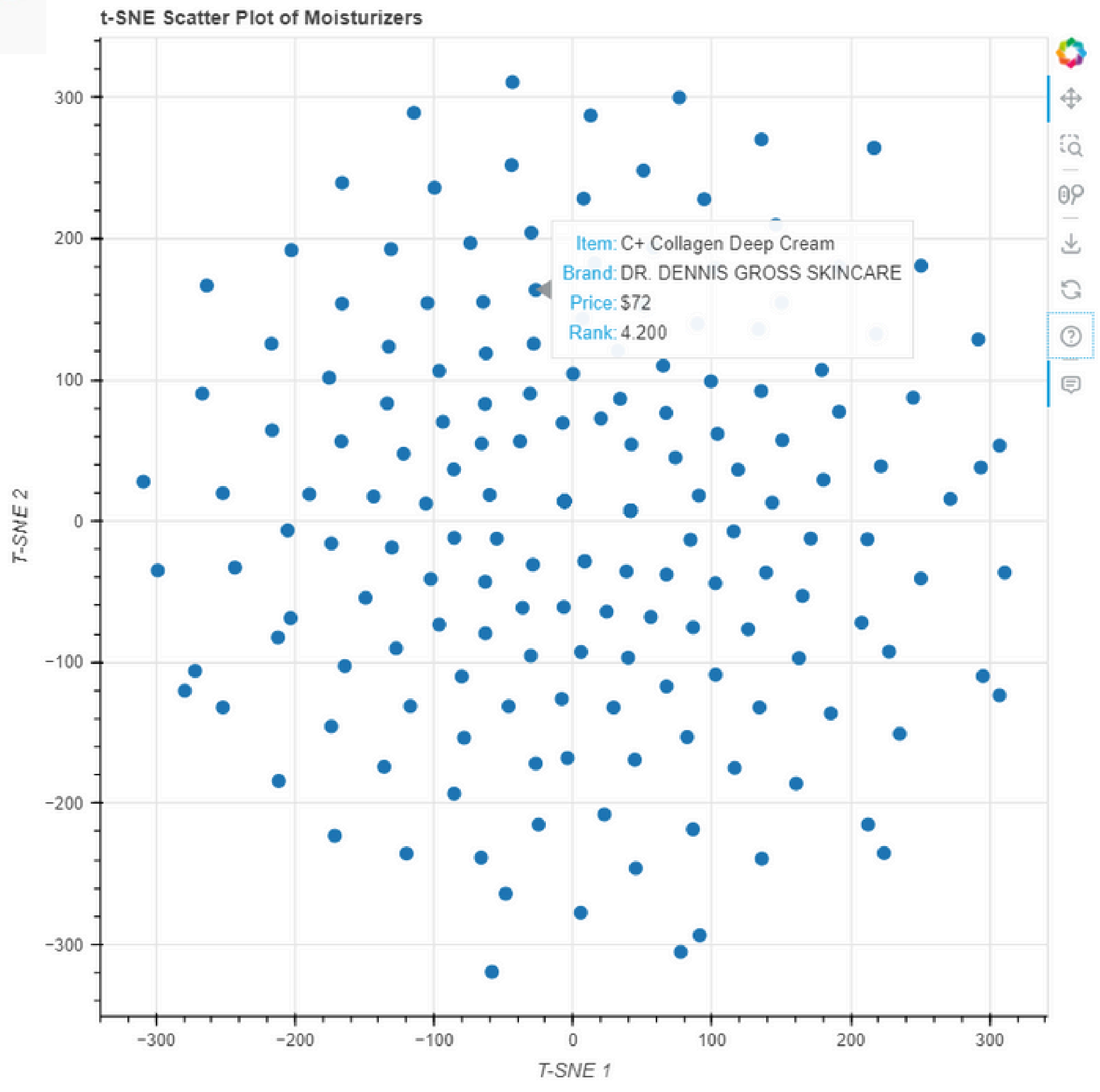

task 9 and 10.py - C:/Users/thiru/Documents/task 9 and 10.py (3.12.4)
File Edit Format Run Options Window Help

```
from bokeh.plotting import figure, show, output_file, save
from bokeh.models import ColumnDataSource, HoverTool
import pandas as pd
import numpy as np
from sklearn.manifold import TSNE
```

4.2 Sample Visuals



(without hover)



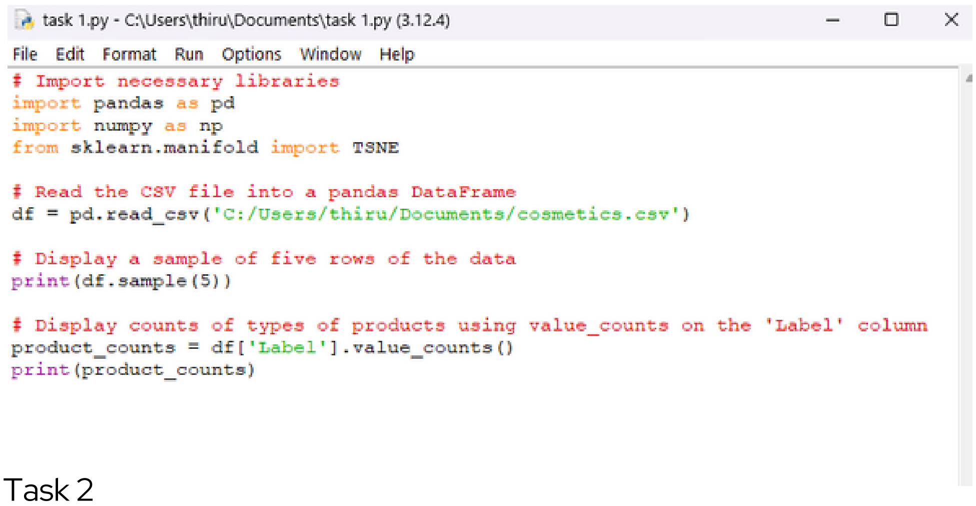
Item: C+ Collagen Deep Cream
Brand: DR. DENNIS GROSS SKINCARE
Price: $72
Rank: 4.200

(with hover)

5.Tasks and Results

5.1 Task 1

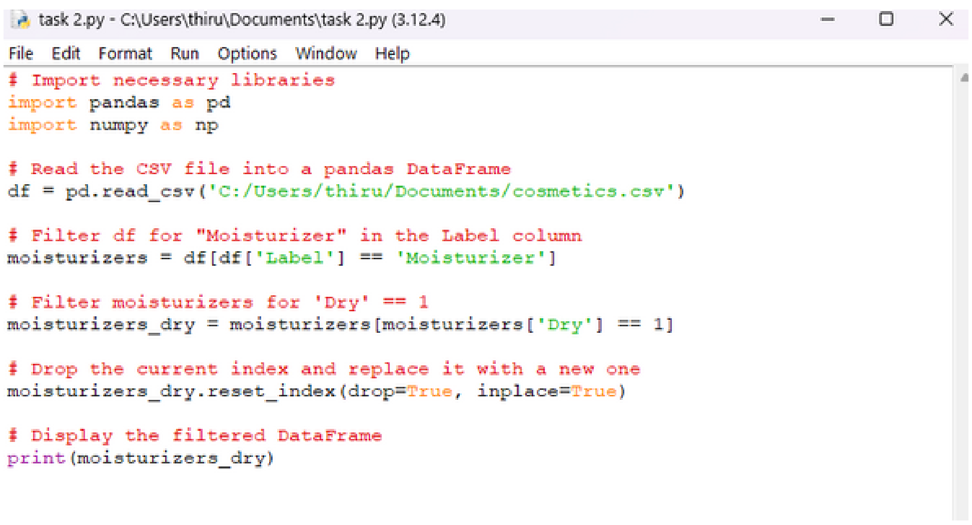
Instructions Import and inspect the dataset. Import pandas aliased as pd and numpy as np Import TSNE from sklearn.manifold Read the CSV file, "datasets/cosmetics.csv", into a pandas DataFrame and name il df. Display a sample of five rows of the data using the sample () method inside the display() function. Display counts of types of product using the value_counts () method on the Label column of df.


task 1.py - C:\Users\thiru\Documents\task 1.py (3.12.4)
File Edit Format Run Options Window Help

```python
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.manifold import TSNE

# Read the CSV file into a pandas DataFrame
df = pd.read_csv('C:/Users/thiru/Documents/cosmetics.csv')

# Display a sample of five rows of the data
print(df.sample(5))

# Display counts of types of products using value_counts on the 'Label' column
product_counts = df['Label'].value_counts()
print(product_counts)
```

Task 2

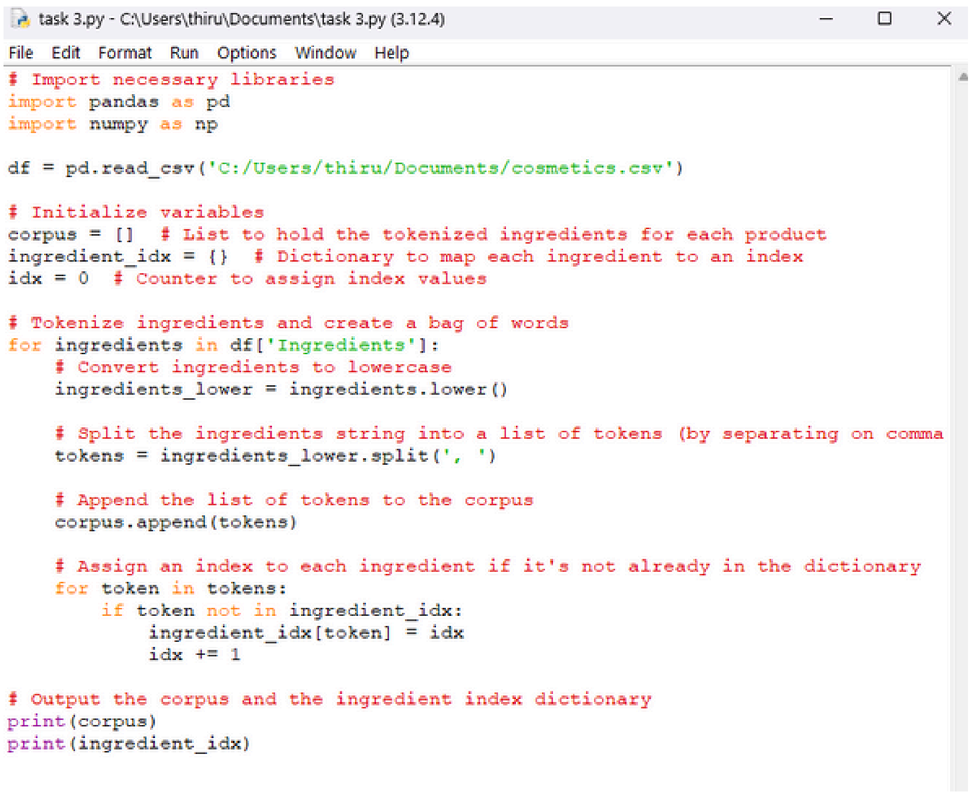
instructions Filter the data for moisturizers and dry skin. Filter df for "Moisturizer" in the Label column and store the result in moisturizers. Filter moisturizers for 1 in the Dry column and store the result in moisturizers_dry. Drop the current index of moisturizers_dry and replace it with a new one using the reset index () method, setting drop = True


task 2.py - C:\Users\thiru\Documents\task 2.py (3.12.4)
File Edit Format Run Options Window Help

```python
# Import necessary libraries
import pandas as pd
import numpy as np

# Read the CSV file into a pandas DataFrame
df = pd.read_csv('C:/Users/thiru/Documents/cosmetics.csv')

# Filter df for "Moisturizer" in the Label column
moisturizers = df[df['Label'] == 'Moisturizer']

# Filter moisturizers for 'Dry' == 1
moisturizers_dry = moisturizers[moisturizers['Dry'] == 1]

# Drop the current index and replace it with a new one
moisturizers_dry.reset_index(drop=True, inplace=True)

# Display the filtered DataFrame
print(moisturizers_dry)
```

Task 3

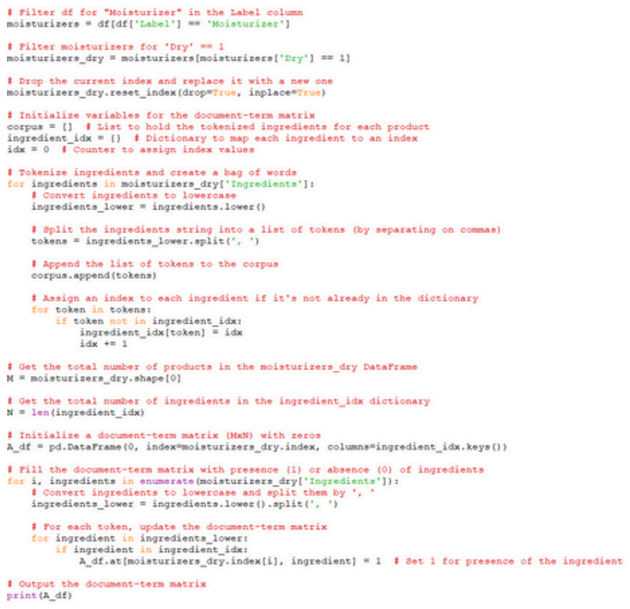
Instructions Tokenize the ingredients and create a bag of words. Inside the outer for loop: Make each product's ingredients list lowercase. Split the lowercase text into tokens by specifying, as the separator. Append tokens (which itself is a list) to the list corpus. Inside the inner for loop, if the ingredient is not yet in ingredient_idx dictionary: Add an entry to ingredient_idx with the key being the new ingredient and the value being the current idx value. Increment idx by 1


task 3.py - C:\Users\thiru\Documents\task 3.py (3.12.4)
File Edit Format Run Options Window Help

```python
# Import necessary libraries
import pandas as pd
import numpy as np

df = pd.read_csv('C:/Users/thiru/Documents/cosmetics.csv')

# Initialize variables
corpus = []  # List to hold the tokenized ingredients for each product
ingredient_idx = {}  # Dictionary to map each ingredient to an index
idx = 0  # Counter to assign index values

# Tokenize ingredients and create a bag of words
for ingredients in df['Ingredients']:
    # Convert ingredients to lowercase
    ingredients_lower = ingredients.lower()

    # Split the ingredients string into a list of tokens (by separating on comma
    tokens = ingredients_lower.split(', ')

    # Append the list of tokens to the corpus
    corpus.append(tokens)

    # Assign an index to each ingredient if it's not already in the dictionary
    for token in tokens:
        if token not in ingredient_idx:
            ingredient_idx[token] = idx
            idx += 1

# Output the corpus and the ingredient index dictionary
print(corpus)
print(ingredient_idx)
```

Task 4

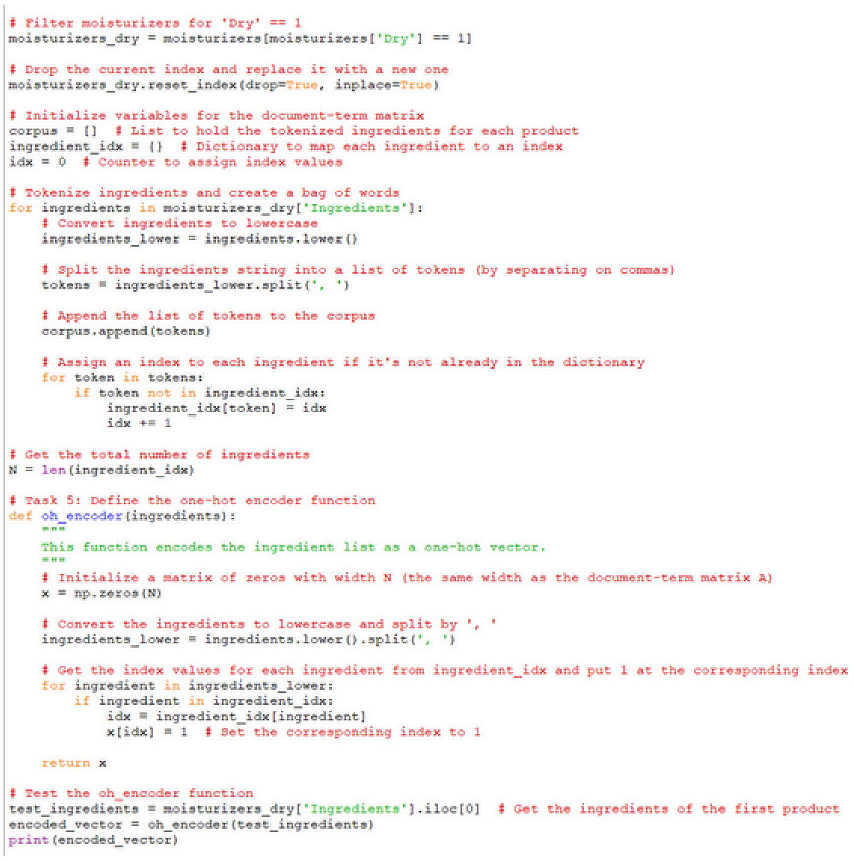
IInstructions Initialize a documont-term matrix. Get the total number of products in the moisturizers_dry Dataframe Assign it to M. Get the total number of ingredients in the ingredient_idx dictionary Assign it to N Create a matrix of zeros with sizo MxN. Assign it to A



```python
# Filter df for "Moisturizer" in the Label column
moisturizers = df[df['Label'] == 'Moisturizer']

# Filter moisturizers for 'Dry' == 1
moisturizers_dry = moisturizers[moisturizers['Dry'] == 1]

# Drop the current index and replace it with a new one
moisturizers_dry.reset_index(drop=True, inplace=True)

# Initialize variables for the document-term matrix
corpus = []  # List to hold the tokenized ingredients for each product
ingredient_idx = {}  # Dictionary to map each ingredient to an index
idx = 0  # Counter to assign index values

# Tokenize ingredients and create a bag of words
for ingredients in moisturizers_dry['Ingredients']:
    # Convert ingredients to lowercase
    ingredients_lower = ingredients.lower()

    # Split the ingredients string into a list of tokens (by separating on commas)
    tokens = ingredients_lower.split(', ')

    # Append the list of tokens to the corpus
    corpus.append(tokens)

    # Assign an index to each ingredient if it's not already in the dictionary
    for token in tokens:
        if token not in ingredient_idx:
            ingredient_idx[token] = idx
            idx += 1

# Get the total number of products in the moisturizers_dry DataFrame
M = moisturizers_dry.shape[0]

# Get the total number of ingredients in the ingredient_idx dictionary
N = len(ingredient_idx)

# Initialize a document-term matrix (MxN) with zeros
A_df = pd.DataFrame(0, index=moisturizers_dry.index, columns=ingredient_idx.keys())

# Fill the document-term matrix with presence (1) or absence (0) of ingredients
for i, ingredients in enumerate(moisturizers_dry['Ingredients']):
    # Convert ingredients to lowercase and split them by ', '
    ingredients_lower = ingredients.lower().split(', ')

    # For each token, update the document-term matrix
    for ingredient in ingredients_lower:
        if ingredient in ingredient_idx:
            A_df.at[moisturizers_dry.index[i], ingredient] = 1  # Set 1 for presence of the ingredient

# Output the document-term matrix
print(A_df)
```

Task 5

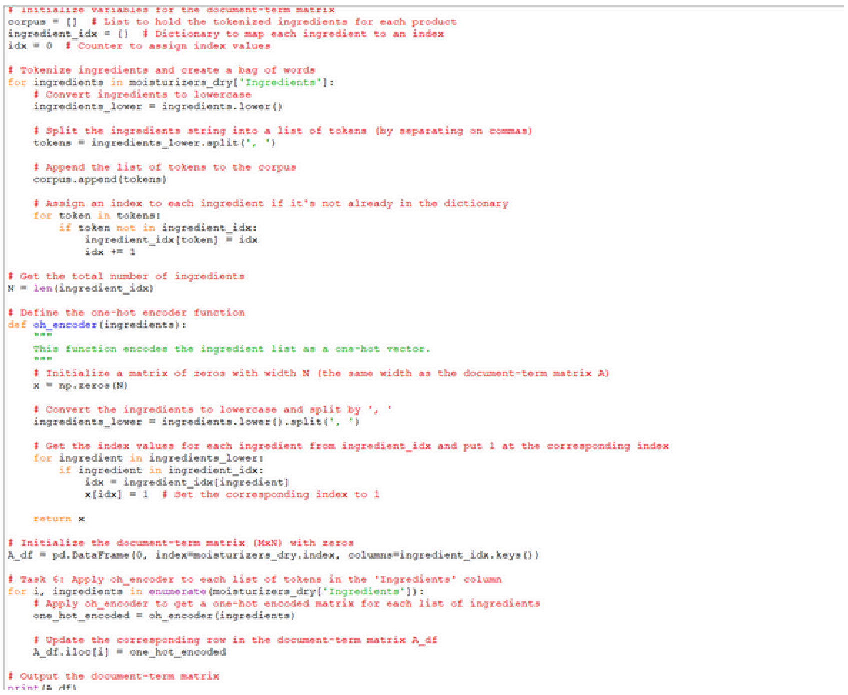
Instructions Create a function named oh_encoder. Initialize a matrix of zeros with width N (i.e., the same width as matrix A). Get the index values for each ingredient from ingredient_idx. Put 1 at the corresponding indices. Return the matrix x same read from the same directory



```
# Filter moisturizers for 'Dry' == 1
moisturizers_dry = moisturizers[moisturizers['Dry'] == 1]

# Drop the current index and replace it with a new one
moisturizers_dry.reset_index(drop=True, inplace=True)

# Initialize variables for the document-term matrix
corpus = []  # List to hold the tokenized ingredients for each product
ingredient_idx = {}  # Dictionary to map each ingredient to an index
idx = 0  # Counter to assign index values

# Tokenize ingredients and create a bag of words
for ingredients in moisturizers_dry['Ingredients']:
    # Convert ingredients to lowercase
    ingredients_lower = ingredients.lower()

    # Split the ingredients string into a list of tokens (by separating on commas)
    tokens = ingredients_lower.split(', ')

    # Append the list of tokens to the corpus
    corpus.append(tokens)

    # Assign an index to each ingredient if it's not already in the dictionary
    for token in tokens:
        if token not in ingredient_idx:
            ingredient_idx[token] = idx
            idx += 1

# Get the total number of ingredients
N = len(ingredient_idx)

# Task 5: Define the one-hot encoder function
def oh_encoder(ingredients):
    """
    This function encodes the ingredient list as a one-hot vector.
    """
    # Initialize a matrix of zeros with width N (the same width as the document-term matrix A)
    x = np.zeros(N)

    # Convert the ingredients to lowercase and split by ', '
    ingredients_lower = ingredients.lower().split(', ')

    # Get the index values for each ingredient from ingredient_idx and put 1 at the corresponding index
    for ingredient in ingredients_lower:
        if ingredient in ingredient_idx:
            idx = ingredient_idx[ingredient]
            x[idx] = 1  # Set the corresponding index to 1

    return x

# Test the oh_encoder function
test_ingredients = moisturizers_dry['Ingredients'].iloc[0]  # Get the ingredients of the first product
encoded_vector = oh_encoder(test_ingredients)
print(encoded_vector)
```

Task 6

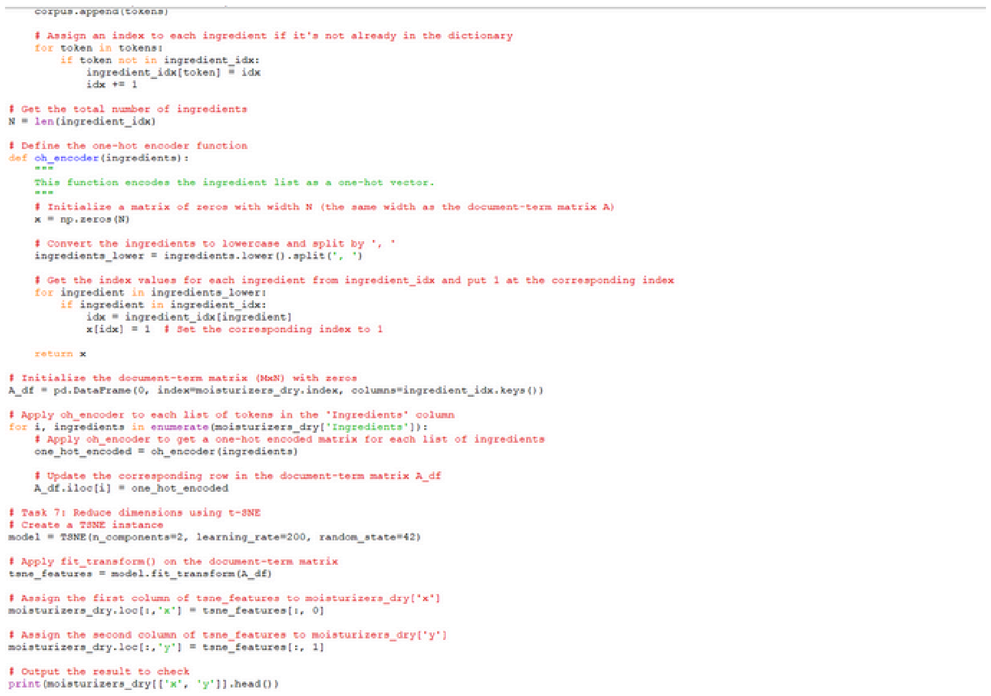
Instructions Get the binary value of the tokens for each row of the matrix A. Inside the for loop: Apply oh_encoder () to get a one-hot encoded matrix for each list of tokens in corpus (i.e., each product's ingredients list). Increment i by 1



```
# Initialize variables for the document-term matrix
corpus = []  # List to hold the tokenized ingredients for each product
ingredient_idx = {}  # Dictionary to map each ingredient to an index
idx = 0  # Counter to assign index values

# Tokenize ingredients and create a bag of words
for ingredients in moisturizers_dry['Ingredients']:
    # Convert ingredients to lowercase
    ingredients_lower = ingredients.lower()

    # Split the ingredients string into a list of tokens (by separating on commas)
    tokens = ingredients_lower.split(', ')

    # Append the list of tokens to the corpus
    corpus.append(tokens)

    # Assign an index to each ingredient if it's not already in the dictionary
    for token in tokens:
        if token not in ingredient_idx:
            ingredient_idx[token] = idx
            idx += 1

# Get the total number of ingredients
N = len(ingredient_idx)

# Define the one-hot encoder function
def oh_encoder(ingredients):
    """
    This function encodes the ingredient list as a one-hot vector.
    """
    # Initialize a matrix of zeros with width N (the same width as the document-term matrix A)
    x = np.zeros(N)

    # Convert the ingredients to lowercase and split by ', '
    ingredients_lower = ingredients.lower().split(', ')

    # Get the index values for each ingredient from ingredient_idx and put 1 at the corresponding index
    for ingredient in ingredients_lower:
        if ingredient in ingredient_idx:
            idx = ingredient_idx[ingredient]
            x[idx] = 1  # Set the corresponding index to 1

    return x

# Initialize the document-term matrix (MxN) with zeros
A_df = pd.DataFrame(0, index=moisturizers_dry.index, columns=ingredient_idx.keys())

# Task 6: Apply oh_encoder to each list of tokens in the 'Ingredients' column
for i, ingredients in enumerate(moisturizers_dry['Ingredients']):
    # Apply oh_encoder to get a one-hot encoded matrix for each list of ingredients
    one_hot_encoded = oh_encoder(ingredients)

    # Update the corresponding row in the document-term matrix A_df
    A_df.iloc[i] = one_hot_encoded

# Output the document-term matrix
print(A_df)
```

Task 7

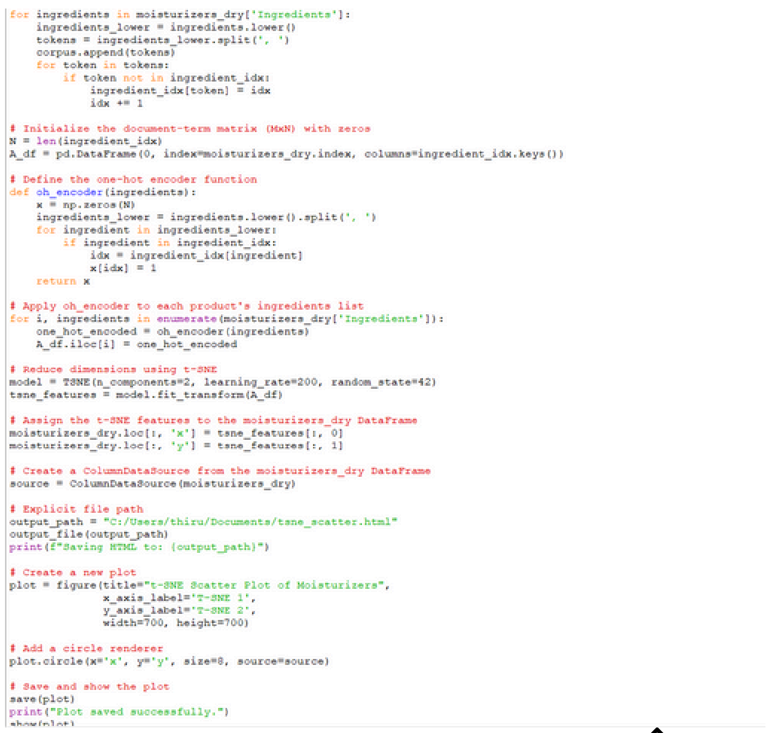
Instructions Reduce the dimensions of the matrix using t-SNE Create a TSNE Instance with n_components 2, learning_rate=200, and random_state = 42 Assign it to model. Apply the fit_transform() method of model to the matrix A. Assign the result 10 tsne_features. Assign the first column of tsne_features 10 moisturizers_dry['x']. Assign the second column of tsne_features to moisturizers_dry['Y'] with read from code like before



```python
    corpus.append(tokens)

    # Assign an index to each ingredient if it's not already in the dictionary
    for token in tokens:
        if token not in ingredient_idx:
            ingredient_idx[token] = idx
            idx += 1

# Get the total number of ingredients
N = len(ingredient_idx)

# Define the one-hot encoder function
def oh_encoder(ingredients):
    """
    This function encodes the ingredient list as a one-hot vector.
    """
    # Initialize a matrix of zeros with width N (the same width as the document-term matrix A)
    x = np.zeros(N)

    # Convert the ingredients to lowercase and split by ', '
    ingredients_lower = ingredients.lower().split(', ')

    # Get the index values for each ingredient from ingredient_idx and put 1 at the corresponding index
    for ingredient in ingredients_lower:
        if ingredient in ingredient_idx:
            idx = ingredient_idx[ingredient]
            x[idx] = 1  # Set the corresponding index to 1

    return x

# Initialize the document-term matrix (MxN) with zeros
A_df = pd.DataFrame(0, index=moisturizers_dry.index, columns=ingredient_idx.keys())

# Apply oh_encoder to each list of tokens in the 'Ingredients' column
for i, ingredients in enumerate(moisturizers_dry['Ingredients']):
    # Apply oh_encoder to get a one-hot encoded matrix for each list of ingredients
    one_hot_encoded = oh_encoder(ingredients)

    # Update the corresponding row in the document-term matrix A_df
    A_df.iloc[i] = one_hot_encoded

# Task 7: Reduce dimensions using t-SNE
# Create a TSNE instance
model = TSNE(n_components=2, learning_rate=200, random_state=42)

# Apply fit_transform() on the document-term matrix
tsne_features = model.fit_transform(A_df)

# Assign the first column of tsne_features to moisturizers_dry['x']
moisturizers_dry.loc[:,'x'] = tsne_features[:, 0]

# Assign the second column of tsne_features to moisturizers_dry['y']
moisturizers_dry.loc[:,'y'] = tsne_features[:, 1]

# Output the result to check
print(moisturizers_dry[['x', 'y']].head())
```

Task 8

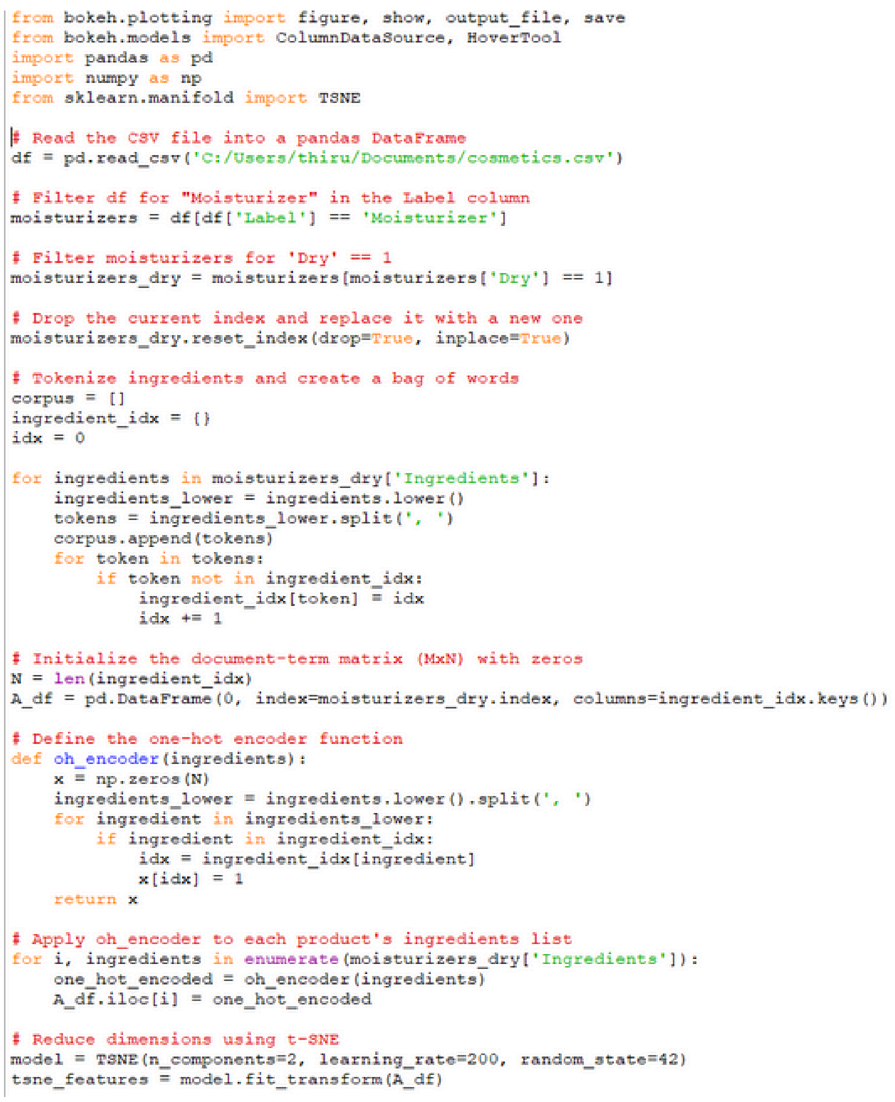
IInstructions Plot a scatter plot with the vectorized items. Create a ColumnDataSource with moisturizers_dry. Assign it to source Label the x-axis as T-SNE 1 and the y-axis as T-SNE 2. Add a circle renderer using plot.circle(), setting x = 'x',y = 'Y', and source to the ColumnDatasource you created. with read code like before



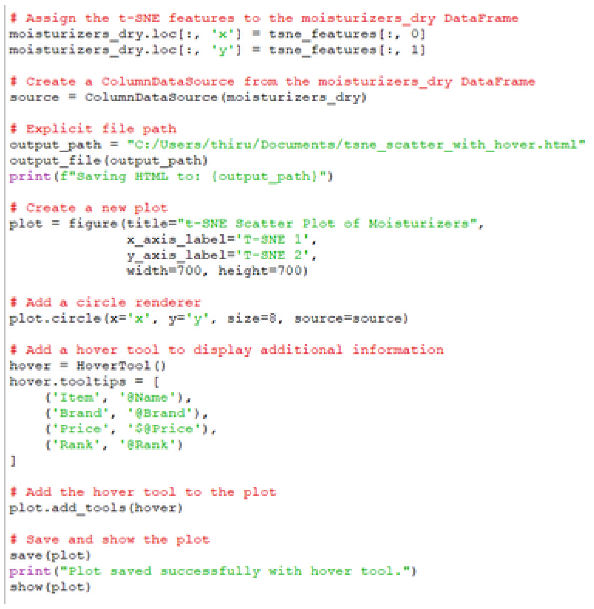
```python
for ingredients in moisturizers_dry['Ingredients']:
    ingredients_lower = ingredients.lower()
    tokens = ingredients_lower.split(', ')
    corpus.append(tokens)
    for token in tokens:
        if token not in ingredient_idx:
            ingredient_idx[token] = idx
            idx += 1

# Initialize the document-term matrix (MxN) with zeros
N = len(ingredient_idx)
A_df = pd.DataFrame(0, index=moisturizers_dry.index, columns=ingredient_idx.keys())

# Define the one-hot encoder function
def oh_encoder(ingredients):
    x = np.zeros(N)
    ingredients_lower = ingredients.lower().split(', ')
    for ingredient in ingredients_lower:
        if ingredient in ingredient_idx:
            idx = ingredient_idx[ingredient]
            x[idx] = 1
    return x

# Apply oh_encoder to each product's ingredients list
for i, ingredients in enumerate(moisturizers_dry['Ingredients']):
    one_hot_encoded = oh_encoder(ingredients)
    A_df.iloc[i] = one_hot_encoded

# Reduce dimensions using t-SNE
model = TSNE(n_components=2, learning_rate=200, random_state=42)
tsne_features = model.fit_transform(A_df)

# Assign the t-SNE features to the moisturizers_dry DataFrame
moisturizers_dry.loc[:, 'x'] = tsne_features[:, 0]
moisturizers_dry.loc[:, 'y'] = tsne_features[:, 1]

# Create a ColumnDataSource from the moisturizers_dry DataFrame
source = ColumnDataSource(moisturizers_dry)

# Explicit file path
output_path = "C:/Users/thiru/Documents/tsne_scatter.html"
output_file(output_path)
print(f"Saving HTML to: {output_path}")

# Create a new plot
plot = figure(title="t-SNE Scatter Plot of Moisturizers",
              x_axis_label='T-SNE 1',
              y_axis_label='T-SNE 2',
              width=700, height=700)

# Add a circle renderer
plot.circle(x='x', y='y', size=8, source=source)

# Save and show the plot
save(plot)
print("Plot saved successfully.")
show(plot)
```

Task 9 and 10

Instructions Add a hover tool. Set the tooltips argument to ('Item', '@Name'), ('Brand', '@Brand'), ('Price', '$@Price'), and ('Rank', '@Rank'). Add the new hover object to the plot. with full code



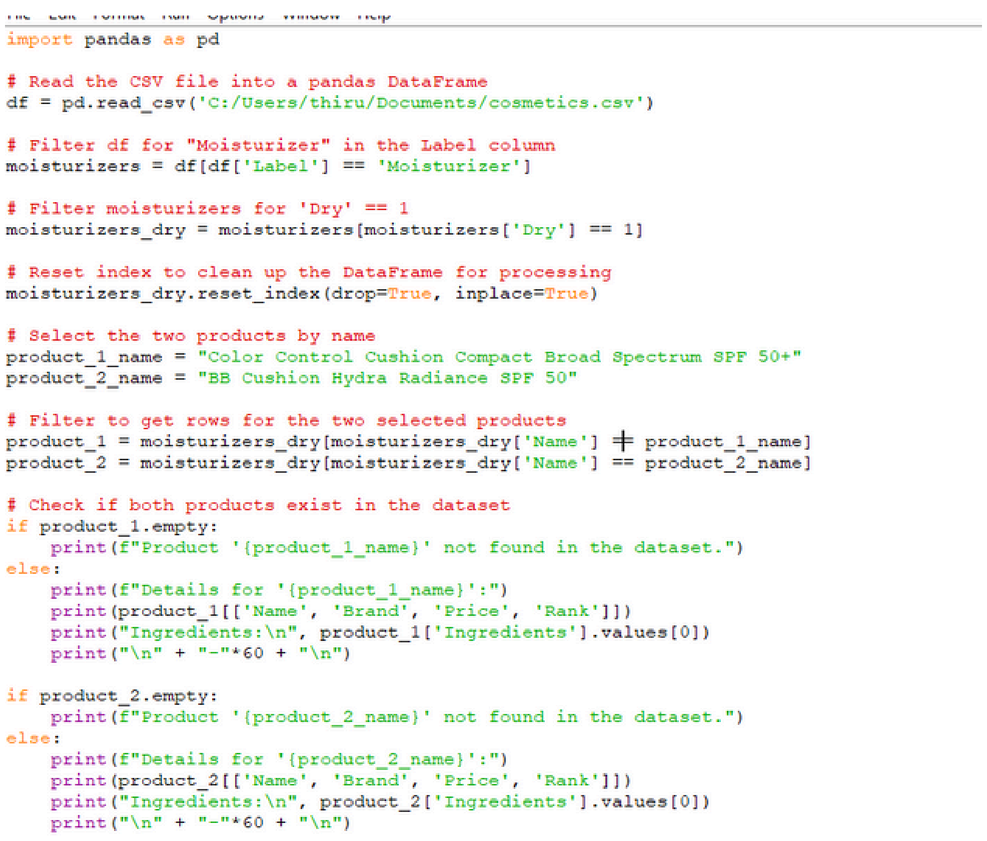
```
from bokeh.plotting import figure, show, output_file, save
from bokeh.models import ColumnDataSource, HoverTool
import pandas as pd
import numpy as np
from sklearn.manifold import TSNE

# Read the CSV file into a pandas DataFrame
df = pd.read_csv('C:/Users/thiru/Documents/cosmetics.csv')

# Filter df for "Moisturizer" in the Label column
moisturizers = df[df['Label'] == 'Moisturizer']

# Filter moisturizers for 'Dry' == 1
moisturizers_dry = moisturizers[moisturizers['Dry'] == 1]

# Drop the current index and replace it with a new one
moisturizers_dry.reset_index(drop=True, inplace=True)

# Tokenize ingredients and create a bag of words
corpus = []
ingredient_idx = {}
idx = 0

for ingredients in moisturizers_dry['Ingredients']:
    ingredients_lower = ingredients.lower()
    tokens = ingredients_lower.split(', ')
    corpus.append(tokens)
    for token in tokens:
        if token not in ingredient_idx:
            ingredient_idx[token] = idx
            idx += 1

# Initialize the document-term matrix (MxN) with zeros
N = len(ingredient_idx)
A_df = pd.DataFrame(0, index=moisturizers_dry.index, columns=ingredient_idx.keys())

# Define the one-hot encoder function
def oh_encoder(ingredients):
    x = np.zeros(N)
    ingredients_lower = ingredients.lower().split(', ')
    for ingredient in ingredients_lower:
        if ingredient in ingredient_idx:
            idx = ingredient_idx[ingredient]
            x[idx] = 1
    return x

# Apply oh_encoder to each product's ingredients list
for i, ingredients in enumerate(moisturizers_dry['Ingredients']):
    one_hot_encoded = oh_encoder(ingredients)
    A_df.iloc[i] = one_hot_encoded

# Reduce dimensions using t-SNE
model = TSNE(n_components=2, learning_rate=200, random_state=42)
tsne_features = model.fit_transform(A_df)
```



```
# Assign the t-SNE features to the moisturizers_dry DataFrame
moisturizers_dry.loc[:, 'x'] = tsne_features[:, 0]
moisturizers_dry.loc[:, 'y'] = tsne_features[:, 1]

# Create a ColumnDataSource from the moisturizers_dry DataFrame
source = ColumnDataSource(moisturizers_dry)

# Explicit file path
output_path = "C:/Users/thiru/Documents/tsne_scatter_with_hover.html"
output_file(output_path)
print(f"Saving HTML to: {output_path}")

# Create a new plot
plot = figure(title="t-SNE Scatter Plot of Moisturizers",
              x_axis_label='T-SNE 1',
              y_axis_label='T-SNE 2',
              width=700, height=700)

# Add a circle renderer
plot.circle(x='x', y='y', size=8, source=source)

# Add a hover tool to display additional information
hover = HoverTool()
hover.tooltips = [
    ('Item', '@Name'),
    ('Brand', '@Brand'),
    ('Price', '$@Price'),
    ('Rank', '@Rank')
]

# Add the hover tool to the plot
plot.add_tools(hover)

# Save and show the plot
save(plot)
print("Plot saved successfully with hover tool.")
show(plot)
```

Task 11

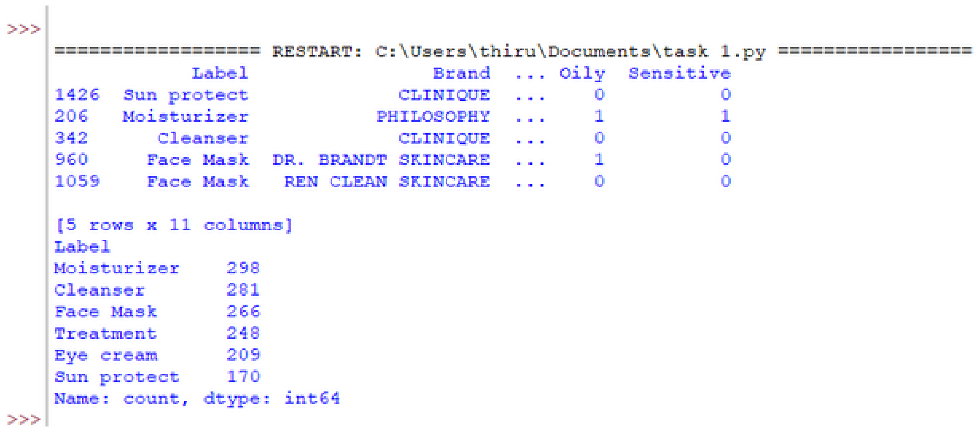
Instructions Print out the ingredients for two similar products. Run the cell as is to print out the data and ingredients for Color Control Cushion Compact Broad Spectrum SPF 50+ and BB Cushion Hydra Radiance SPF 50. with full code like before



```python
import pandas as pd

# Read the CSV file into a pandas DataFrame
df = pd.read_csv('C:/Users/thiru/Documents/cosmetics.csv')

# Filter df for "Moisturizer" in the Label column
moisturizers = df[df['Label'] == 'Moisturizer']

# Filter moisturizers for 'Dry' == 1
moisturizers_dry = moisturizers[moisturizers['Dry'] == 1]

# Reset index to clean up the DataFrame for processing
moisturizers_dry.reset_index(drop=True, inplace=True)

# Select the two products by name
product_1_name = "Color Control Cushion Compact Broad Spectrum SPF 50+"
product_2_name = "BB Cushion Hydra Radiance SPF 50"

# Filter to get rows for the two selected products
product_1 = moisturizers_dry[moisturizers_dry['Name'] == product_1_name]
product_2 = moisturizers_dry[moisturizers_dry['Name'] == product_2_name]

# Check if both products exist in the dataset
if product_1.empty:
    print(f"Product '{product_1_name}' not found in the dataset.")
else:
    print(f"Details for '{product_1_name}':")
    print(product_1[['Name', 'Brand', 'Price', 'Rank']])
    print("Ingredients:\n", product_1['Ingredients'].values[0])
    print("\n" + "-"*60 + "\n")

if product_2.empty:
    print(f"Product '{product_2_name}' not found in the dataset.")
else:
    print(f"Details for '{product_2_name}':")
    print(product_2[['Name', 'Brand', 'Price', 'Rank']])
    print("Ingredients:\n", product_2['Ingredients'].values[0])
    print("\n" + "-"*60 + "\n")
```

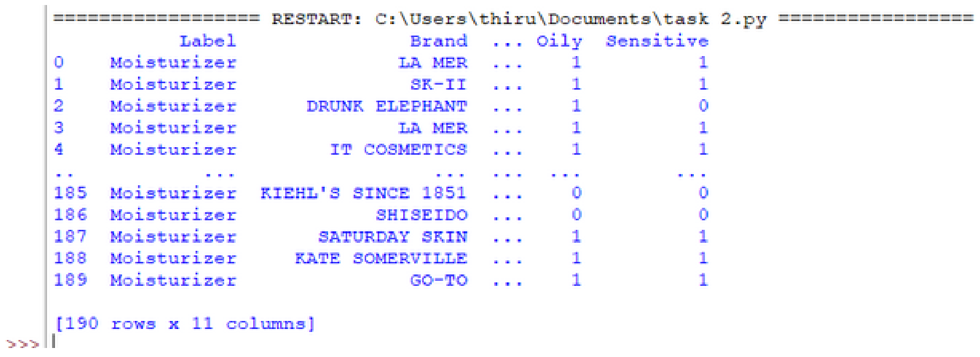
Results

Task 1



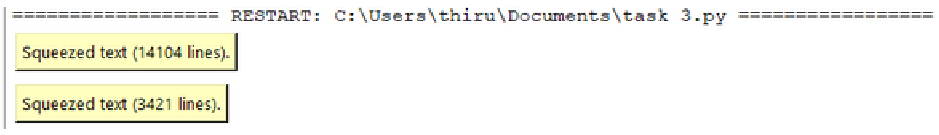
```
>>> 
================== RESTART: C:\Users\thiru\Documents\task 1.py ==================
            Label                Brand  ... Oily  Sensitive
1426  Sun protect             CLINIQUE  ...    0          0
206   Moisturizer           PHILOSOPHY  ...    1          1
342      Cleanser             CLINIQUE  ...    0          0
960     Face Mask  DR. BRANDT SKINCARE  ...    1          0
1059    Face Mask   REN CLEAN SKINCARE  ...    0          0

[5 rows x 11 columns]
Label
Moisturizer    298
Cleanser       281
Face Mask      266
Treatment      248
Eye cream      209
Sun protect    170
Name: count, dtype: int64
>>> 
```

Task 2



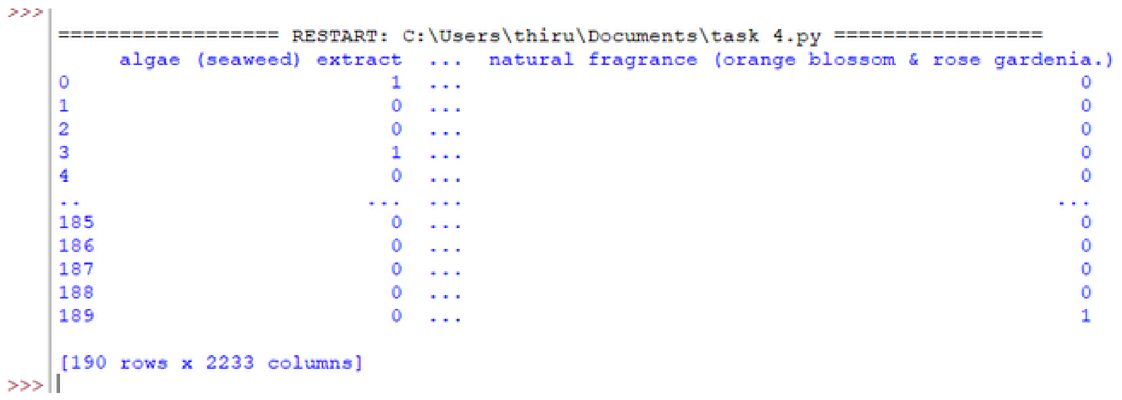
```
================== RESTART: C:\Users\thiru\Documents\task 2.py ==================
           Label               Brand  ... Oily  Sensitive
0    Moisturizer              LA MER  ...    1          1
1    Moisturizer              SK-II   ...    1          1
2    Moisturizer      DRUNK ELEPHANT  ...    1          0
3    Moisturizer              LA MER  ...    1          1
4    Moisturizer        IT COSMETICS  ...    1          1
..           ...                 ...  ...  ...        ...
185  Moisturizer  KIEHL'S SINCE 1851  ...    0          0
186  Moisturizer            SHISEIDO  ...    0          0
187  Moisturizer       SATURDAY SKIN  ...    1          1
188  Moisturizer     KATE SOMERVILLE  ...    1          1
189  Moisturizer               GO-TO  ...    1          1

[190 rows x 11 columns]
>>> 
```

Task 3



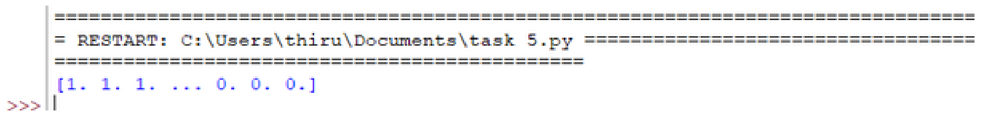
```
================== RESTART: C:\Users\thiru\Documents\task 3.py ==================
Squeezed text (14104 lines).

Squeezed text (3421 lines).
```

Task 4



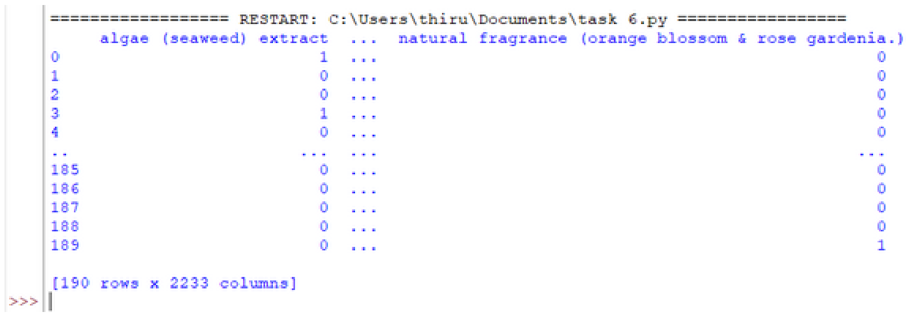
```
>>>
==================== RESTART: C:\Users\thiru\Documents\task 4.py ==================
     algae (seaweed) extract  ...  natural fragrance (orange blossom & rose gardenia.)
0                          1  ...                                                    0
1                          0  ...                                                    0
2                          0  ...                                                    0
3                          1  ...                                                    0
4                          0  ...                                                    0
..                       ...  ...                                                  ...
185                        0  ...                                                    0
186                        0  ...                                                    0
187                        0  ...                                                    0
188                        0  ...                                                    0
189                        0  ...                                                    1

[190 rows x 2233 columns]
>>>
```

Task 5



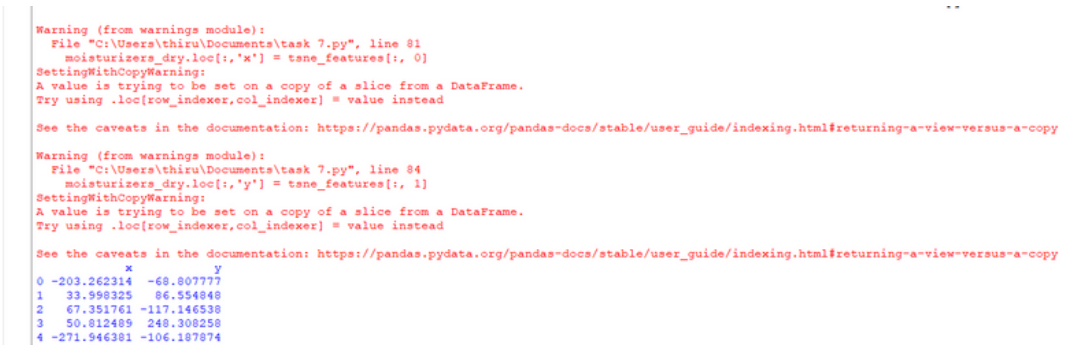
```
================================================================================
= RESTART: C:\Users\thiru\Documents\task 5.py ==================================
==============================================
[1. 1. 1. ... 0. 0. 0.]
>>>
```

Task 6



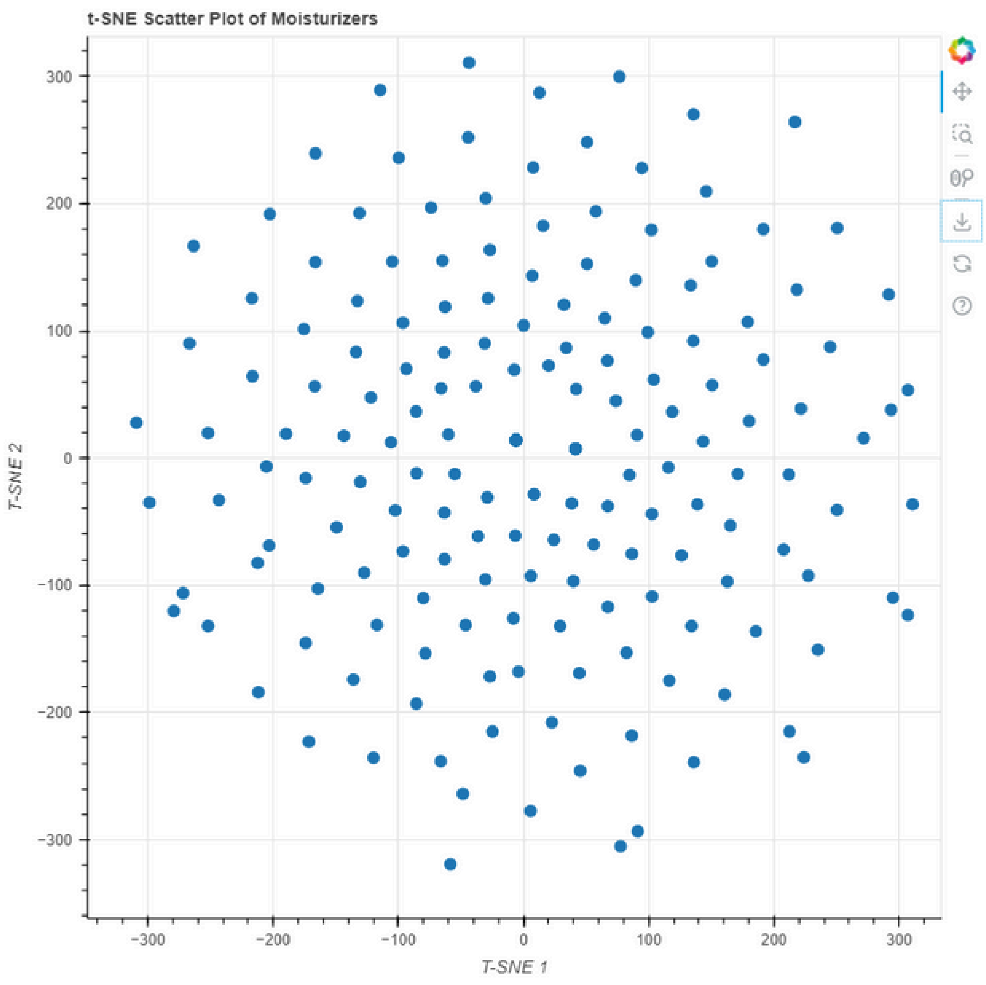
```
================== RESTART: C:\Users\thiru\Documents\task 6.py =================
     algae (seaweed) extract  ...  natural fragrance (orange blossom & rose gardenia.)
0                          1  ...                                                    0
1                          0  ...                                                    0
2                          0  ...                                                    0
3                          1  ...                                                    0
4                          0  ...                                                    0
..                       ...  ...                                                  ...
185                        0  ...                                                    0
186                        0  ...                                                    0
187                        0  ...                                                    0
188                        0  ...                                                    0
189                        0  ...                                                    1

[190 rows x 2233 columns]
>>>
```

Task 7

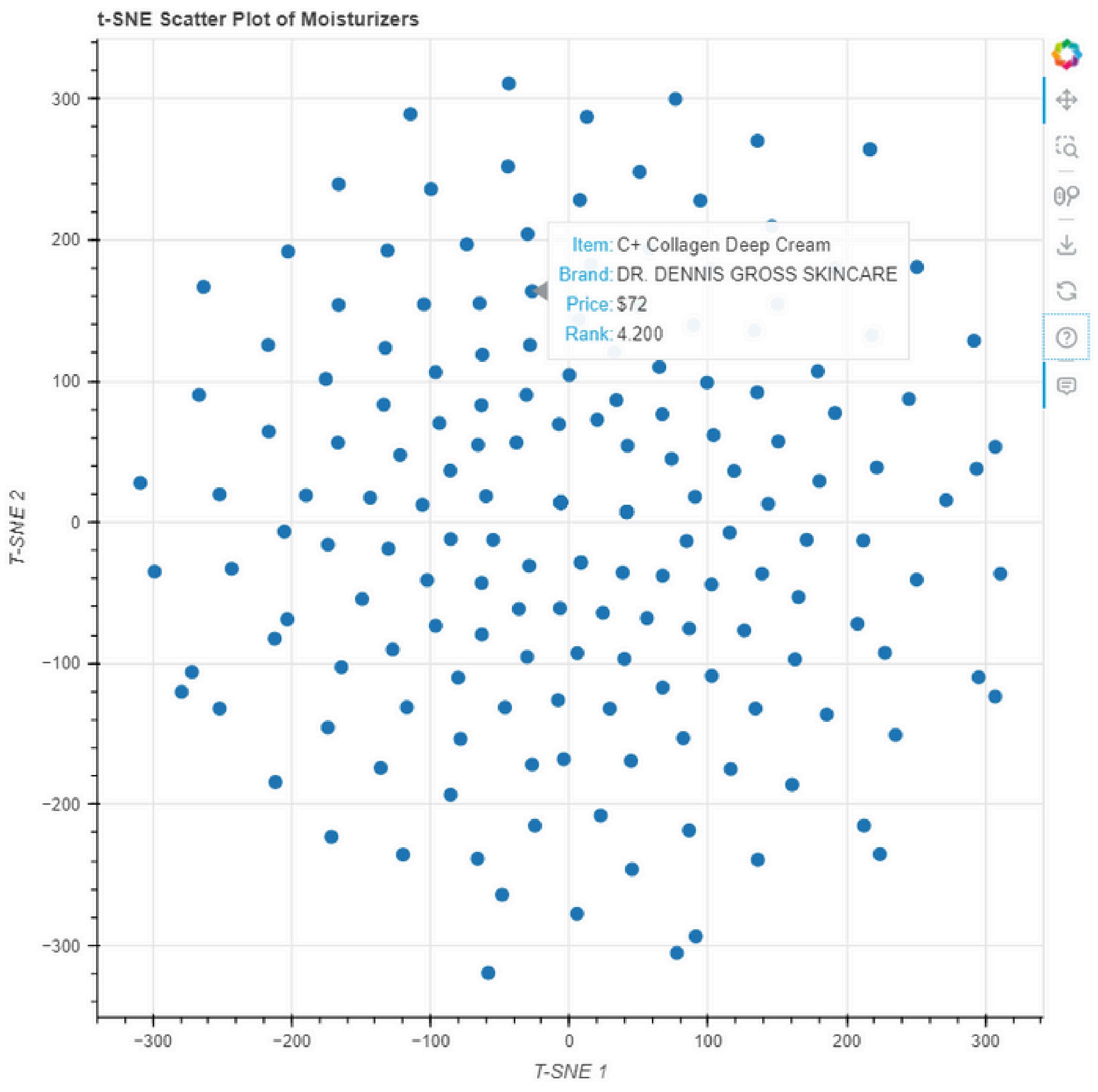


```
Warning (from warnings module):
  File "C:\Users\thiru\Documents\task 7.py", line 81
    moisturizers_dry.loc[:,'x'] = tsne_features[:, 0]
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

Warning (from warnings module):
  File "C:\Users\thiru\Documents\task 7.py", line 84
    moisturizers_dry.loc[:,'y'] = tsne_features[:, 1]
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
            x           y
0 -203.262314  -68.807777
1   33.998325   86.554848
2   67.351761 -117.146538
3   50.812489  248.308258
4 -271.946381 -106.187874
```

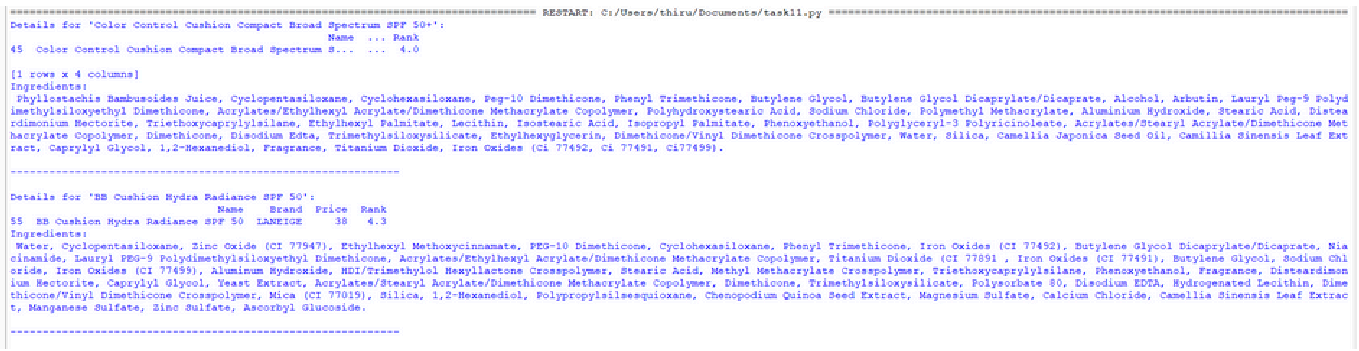
Task 8



Task 9 & 10


t-SNE Scatter Plot of Moisturizers
Item: C+ Collagen Deep Cream
Brand: DR. DENNIS GROSS SKINCARE
Price: $72
Rank: 4.200
T-SNE 1
T-SNE 2
300
200
100
0
−100
−200
−300
−300
−200
−100
0
100
200
300

Task 11



```
================================================================================ RESTART: C:/Users/thiru/Documents/task11.py ================================================================================
Details for 'Color Control Cushion Compact Broad Spectrum SPF 50+':
                                                Name  ... Rank
45  Color Control Cushion Compact Broad Spectrum S...  ...  4.0

[1 rows x 4 columns]
Ingredients:
 Phyllostachis Bambusoides Juice, Cyclopentasiloxane, Cyclohexasiloxane, Peg-10 Dimethicone, Phenyl Trimethicone, Butylene Glycol, Butylene Glycol Dicaprylate/Dicaprate, Alcohol, Arbutin, Lauryl Peg-9 Polyd
imethylsiloxyethyl Dimethicone, Acrylates/Ethylhexyl Acrylate/Dimethicone Methacrylate Copolymer, Polyhydroxystearic Acid, Sodium Chloride, Polymethyl Methacrylate, Aluminium Hydroxide, Stearic Acid, Distea
rdimonium Hectorite, Triethoxycaprylylsilane, Ethylhexyl Palmitate, Lecithin, Isostearic Acid, Isopropyl Palmitate, Phenoxyethanol, Polyglyceryl-3 Polyricinoleate, Acrylates/Stearyl Acrylate/Dimethicone Met
hacrylate Copolymer, Dimethicone, Disodium Edta, Trimethylsiloxysilicate, Ethylhexyglycerin, Dimethicone/Vinyl Dimethicone Crosspolymer, Water, Silica, Camellia Japonica Seed Oil, Camillia Sinensis Leaf Ext
ract, Caprylyl Glycol, 1,2-Hexanediol, Fragrance, Titanium Dioxide, Iron Oxides (Ci 77492, Ci 77491, Ci77499).

------------------------------------------------------------

Details for 'BB Cushion Hydra Radiance SPF 50':
                               Name    Brand  Price  Rank
55  BB Cushion Hydra Radiance SPF 50  LANEIGE     38   4.3
Ingredients:
 Water, Cyclopentasiloxane, Zinc Oxide (CI 77947), Ethylhexyl Methoxycinnamate, PEG-10 Dimethicone, Cyclohexasiloxane, Phenyl Trimethicone, Iron Oxides (CI 77492), Butylene Glycol Dicaprylate/Dicaprate, Nia
cinamide, Lauryl PEG-9 Polydimethylsiloxyethyl Dimethicone, Acrylates/Ethylhexyl Acrylate/Dimethicone Methacrylate Copolymer, Titanium Dioxide (CI 77891 , Iron Oxides (CI 77491), Butylene Glycol, Sodium Chl
oride, Iron Oxides (CI 77499), Aluminum Hydroxide, HDI/Trimethylol Hexyllactone Crosspolymer, Stearic Acid, Methyl Methacrylate Crosspolymer, Triethoxycaprylylsilane, Phenoxyethanol, Fragrance, Disteardimon
ium Hectorite, Caprylyl Glycol, Yeast Extract, Acrylates/Stearyl Acrylate/Dimethicone Methacrylate Copolymer, Dimethicone, Trimethylsiloxysilicate, Polysorbate 80, Disodium EDTA, Hydrogenated Lecithin, Dime
thicone/Vinyl Dimethicone Crosspolymer, Mica (CI 77019), Silica, 1,2-Hexanediol, Polypropylsilsesquioxane, Chenopodium Quinoa Seed Extract, Magnesium Sulfate, Calcium Chloride, Camellia Sinensis Leaf Extrac
t, Manganese Sulfate, Zinc Sulfate, Ascorbyl Glucoside.

------------------------------------------------------------
```

6. CONCLUSION AND FUTURE SCOPE

The interactive recommendation system for cosmetics has demonstrated its effectiveness in assisting users in selecting products tailored to their specific skin types and preferences. By leveraging data analysis and machine learning techniques, particularly the t-SNE algorithm for dimensionality reduction, the system provides valuable insights into ingredient similarities and product relationships.

This project successfully addressed the challenges faced by users with sensitive skin, allowing them to make informed decisions based on their individual needs. The incorporation of an interactive visualization platform enhances user engagement, making the selection process more enjoyable and efficient. Overall, the system not only aids in product discovery but also empowers users with knowledge about the ingredients in their cosmetics, fostering a more conscious approach to beauty.

the system could be enhanced by developing a mobile application, making the recommendation tool more accessible to users on-the-go. This could also include features such as barcode scanning to instantly retrieve product information and recommendations based on user input.
Lastly, collaborations with dermatologists or skincare experts could validate the recommendations, ensuring that users receive professional guidance tailored to their unique skin conditions. By addressing these areas, the recommendation system can evolve into a comprehensive tool that not only enhances user experience but also promotes healthier skincare choices.

7.References

*__Books and Articles:__**

- McKinsey & Company. (2021). "The State of Fashion 2021." [Link](https://www.mckinsey.com/industries/retail/our-insights/the-state-of-fashion-2021)
- Shankar, V., & Bolton, R. N. (2004). "An Integrative Framework for Conducting Retailing Research." Journal of Retailing, 80(2), 103-116. [Link](https://www.sciencedirect.com/science/article/pii/S0022435904000120)
- P. K. Jain and A. Jain, "A Survey on Recommendation System," International Journal of Computer Applications, vol. 139, no. 10, pp. 15-18, 2016. [Link](https://www.ijcaonline.org/research/volume139/number10/24052-2016902269)

*__Datasets:__**

- Sephora Product Data: "Sephora's API for Product Data" [Link](https://www.sephora.com/api)
- Cosmetics Ingredient Review. "Cosmetic Ingredient Dictionary." [Link](https://www.cir-safety.org/ingredients)

*__Machine Learning and Data Analysis:__**

- Aggarwal, C. C. (2016). Recommender Systems: The Textbook. Springer. [Link](https://link.springer.com/book/10.1007/978-3-319-29659-3)
- Koren, Y., Bell, R., & Volinsky, C. (2009). "Matrix Factorization Techniques for Recommender Systems." Computer Science Review, 9(3-4), 254-299. [Link](https://www.sciencedirect.com/science/article/pii/S1574013709000220)
- G. K. Gupta and M. S. Iyer, "Recommendation Systems: A Survey," International Journal of Computer Applications, vol. 152, no. 1, pp. 1-5, 2016. [Link](https://www.ijcaonline.org/research/volume152/number1/26417-2016901862)

*__Web Resources__:**

- Towards Data Science. "Building a Content-Based Recommendation System Using Python." [Link](https://towardsdatascience.com/building-a-content-based-recommendation-system-using-python-1f5c7a5b1c7c)
- Kaggle. "Cosmetic Data Analysis." [Link](https://www.kaggle.com/datasets)
- Bokeh Documentation. "Bokeh Documentation." [Link](https://docs.bokeh.org/en/latest/)

*__Cosmetic Industry Reports:__**

- Statista. (2023). "Cosmetics Market in the United States - Statistics & Facts." [Link](https://www.statista.com/topics/1009/cosmetics/)
- Grand View Research. (2023). "Cosmetics Market Size, Share & Trends Analysis Report." [Link](https://www.grandviewresearch.com/industry-analysis/cosmetics-market)