Step 1: Load the Dataset

```
from sklearn.datasets import load_iris import pandas as
pd # Load Iris Dataset data = load_iris() df =
pd.DataFrame(data.data, columns=data.feature_names)
df['species'] = data.target
```
Step 2: Standardize the Data

```
from sklearn.preprocessing import StandardScaler
# Standardizing the features
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df.iloc[:, :-1])
```

Step 3: Apply PCA

```
from sklearn.decomposition import PCA # PCA to reduce dimensions to 2 pca = PCA(n_components=2) principalComponents
= pca.fit_transform(df_scaled) principalDf = pd.DataFrame(data=principalComponents, columns=['principal component
1', 'principal component 2']) finalDf = pd.concat([principalDf, df[['species']]], axis=1)
```
Step 4: Visualize the Results

```
import matplotlib.pyplot as plt import seaborn as sns # Visualizing the PCA result
plt.figure(figsize=(8, 6)) sns.scatterplot(x="principal component 1", y="principal
component 2", hue=df['species'].apply(lambda x: data.target_names[x]), data=finalDf,
palette='Set2') plt.title('PCA of Iris Dataset') plt.show()
```

Step 5: Interpret the Results

Q.3. Consider the given problem statement and apply the steps given in question 2: Problem Statement: A retail company has accumulated a large dataset through its customer relationship management system, encompassing various customer behaviours, demographic pro les, and transaction histories. The dataset features over 100 variables, including age, income, purchase history, online engagement metrics, and product preferences. The complexity and high dimensionality of this dataset pose signi cant challenges in extracting actionable insights and effectively segmenting customers to tailor marketing strategies.

Making a DATASET in python and reading it to use the data

```
import pandas as pd
import numpy as np
# Set random seed for reproducibility np.random.seed(42) # Generate synthetic data num_customers = 1000
customer_ids = np.arange(1, num_customers + 1) ages = np.random.randint(18, 70, size=num_customers) incomes =
np.random.randint(30000, 150000, size=num_customers) purchase_history = np.random.randint(1, 50,
size=num_customers) online_engagement = np.random.randint(50, 500, size=num_customers) product_preferences =
np.random.choice(['Electronics', 'Clothing', 'Home Decor', 'Books'], size=num_customers)
# Create DataFrame
df = pd.DataFrame({
'CustomerID': customer_ids,
'Age': ages,
'Income': incomes,
'Purchase_History': purchase_history,
'Online_Engagement': online_engagement,
```

```
'Product_Preferences': product_preferences
}) # Save to CSV
df.to_csv('synthetic_customer_data.csv',
index=False)
print("Synthetic dataset created and saved as 'synthetic_customer_data.csv'")
```

```
Synthetic dataset created and saved as 'synthetic_customer_data.csv'
```
Part

1: Importing Libraries and Loading the Dataset

```
import pandas as pd from sklearn.preprocessing
import StandardScaler from
sklearn.decomposition import PCA import
matplotlib.pyplot as plt import seaborn as sns
# Load the synthetic dataset
df = pd.read_csv('synthetic_customer_data.csv')
```

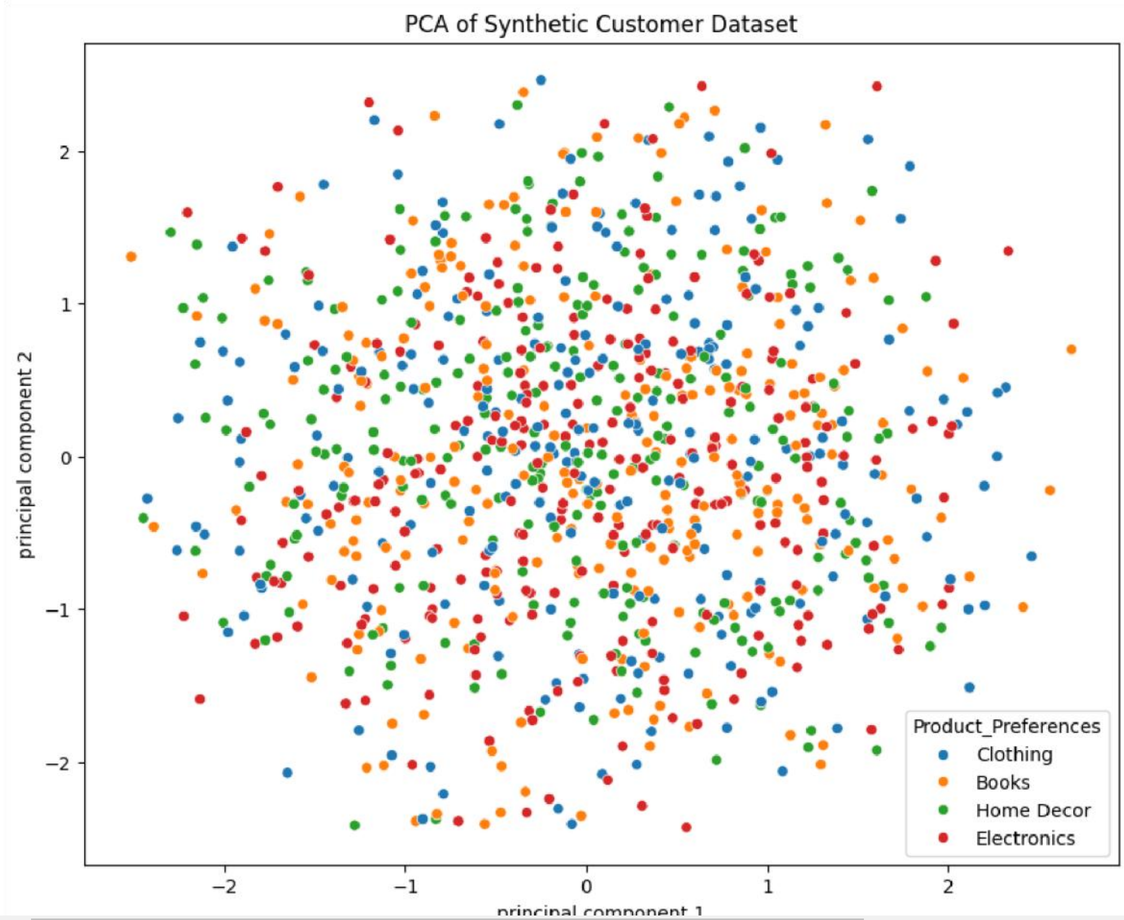
Part 2: Standardizing the Features

```
# Selecting relevant features for PCA features = ['Age', 'Income',
'Purchase_History', 'Online_Engagement'] df_selected = df[features]
# Standardizing the features scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_selected)
```

Part 3: Applying PCA to Reduce Dimensions

```
# PCA to reduce dimensions to 2 pca = PCA(n_components=2) principalComponents = pca.fit_transform(df_scaled)
principalDf = pd.DataFrame(data=principalComponents, columns=['principal component 1', 'principal component 2'])
finalDf = pd.concat([principalDf, df[['CustomerID', 'Product_Preferences']]], axis=1)
```
Part 4: Visualizing the PCA Result

```
# Visualizing the PCA result plt.figure(figsize=(10, 8)) sns.scatterplot(x="principal component 1",
y="principal component 2", hue='Product_Preferences', data=finalDf) plt.title('PCA of Synthetic Customer
Dataset') plt.show()
```



Part 5: Documentation and Visualization of Explained Variance

```
# Explained variance ratio explained_variance = pca.explained_variance_ratio_
print(f'Explained variance by each principal component:
{explained_variance}')
# Visualizing the explained variance plt.figure(figsize=(8, 6)) plt.bar(range(1,
len(explained_variance) + 1), explained_variance, alpha=0.5, align='center')
plt.title('Explained Variance by Principal Components') plt.xlabel('Principal Components')
plt.ylabel('Variance Ratio')
plt.show()
```

```
Explained variance by each principal component: [0.26441139 0.2619286 ]
```