Problem Statement

We want to segment the customers into different groups based on their annual spending on various product categories. This can help the business tailor marketing strategies to different customer segments.

Steps

1. Import necessary libraries.
2. Load and explore the dataset.
3. Preprocess the data (handle missing values, scale features, etc.).
4. Determine the optimal number of clusters using a dendrogram.
5. Apply Agglomerative Clustering.
6. Visualize the clusters.
7. Analyze the results.

Step-by-Step Code with Explanations

Cell 1: Import necessary libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage

# Print statement
print("Libraries imported successfully.")
```

Cell 2: Load and explore the dataset

```
# Load the dataset
url =
"https://archive.ics.uci.edu/ml/machine-learning-databases/00292/Whole
sale customers data.csv"
data = pd.read_csv(url)

# Print statement
print("Dataset loaded successfully.")
```

```
data.head()
```

Cell 3: Check for missing values

```
# Check for missing values
missing_values = data.isnull().sum()

# Print statement
print("Missing values in the dataset:")
print(missing_values)
```

Cell 4: Select features for clustering and scale them

```
# Select features for clustering
features = data[['Fresh', 'Milk', 'Grocery', 'Frozen',
'Detergents_Paper', 'Delicassen']]

# Scale the features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# Print statement
print("Features selected and scaled for clustering.")
print(scaled_features[:5])  # Display the first five rows of scaled
features
```

Cell 5: Determine the optimal number of clusters using a dendrogram

```
# Create the linkage matrix
linked = linkage(scaled_features, method='ward')

# Plot the dendrogram
plt.figure(figsize=(10, 7))
dendrogram(linked, orientation='top', distance_sort='descending',
show_leaf_counts=True)
plt.title('Dendrogram')
plt.xlabel('Samples')
plt.ylabel('Distance')
plt.show()

# Print statement
print("Dendrogram plotted to determine the optimal number of
clusters.")
```

Cell 6: Apply Agglomerative Clustering with the optimal number of clusters (e.g., 3)

```
# Apply Agglomerative Clustering
agg_clustering = AgglomerativeClustering(n_clusters=3)
clusters = agg_clustering.fit_predict(scaled_features)

# Add the cluster labels to the original dataframe
data['Cluster'] = clusters

# Print statement
print("Agglomerative Clustering applied and cluster labels added to
the dataset.")
data.head()
```

Cell 7: Visualize the clusters

```
# Visualize the clusters
plt.figure(figsize=(10, 7))
sns.scatterplot(x='Grocery', y='Milk', hue='Cluster',
palette='viridis', data=data)
plt.title('Customer Segments (Agglomerative Clustering)')
plt.legend()
plt.show()

# Print statement
print("Clusters visualized.")
```

Explanation of Each Cell

Cell 1: Import necessary libraries

- **pandas**: Used for data manipulation and analysis.
- **matplotlib.pyplot** and **seaborn**: Used for data visualization.
- **StandardScaler**: Used to standardize the features by removing the mean and scaling to unit variance.
- **AgglomerativeClustering**: Used to perform hierarchical/agglomerative clustering.
- **dendrogram** and **linkage**: Used to create and plot a dendrogram for hierarchical clustering.

Cell 2: Load and explore the dataset

- **pd.read_csv(url)**: Reads the CSV file from the provided URL and loads it into a DataFrame.
- **data.head()**: Displays the first five rows of the dataset.

Cell 3: Check for missing values

- **data.isnull().sum()**: Checks for missing values in each column and sums them up.
- **print(missing_values)**: Prints the count of missing values for each column.

Cell 4: Select features for clustering and scale them

- **features**: Selects the relevant columns for clustering.
- **StandardScaler()**: Initializes the StandardScaler object.
- **scaler.fit_transform(features)**: Scales the selected features to have zero mean and unit variance.
- **print(scaled_features[:5])**: Displays the first five rows of the scaled features.

Cell 5: Determine the optimal number of clusters using a dendrogram

- **linkage(scaled_features, method='ward')**: Creates the linkage matrix using the Ward method.
- **dendrogram()**: Plots the dendrogram to help determine the optimal number of clusters.

Cell 6: Apply Agglomerative Clustering with the optimal number of clusters (e.g., 3)

- **AgglomerativeClustering(n_clusters=3)**: Initializes the AgglomerativeClustering object with 3 clusters.
- **agg_clustering.fit_predict(scaled_features)**: Fits the model and predicts the cluster labels.
- **data['Cluster'] = clusters**: Adds the predicted cluster labels to the original DataFrame.

Cell 7: Visualize the clusters

- **sns.scatterplot()**: Creates a scatter plot to visualize the clusters.
- **plt.title()**, **plt.legend()**, **plt.show()**: Set the title, add the legend, and display the plot