Problem 7

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering

# Load Online Retail data (assuming you have the dataset as a CSV
file)
data = pd.read_csv('/content/drive/MyDrive/sem 7/ID5055/Assignment
3/Problem 7/OnlineRetail.csv', encoding='ISO-8859-1')

# Drop rows with missing values (you may need more extensive
preprocessing)
data = data.dropna()

data.head()
```

```
  InvoiceNo StockCode                          Description
Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6

1    536365     71053                  WHITE METAL LANTERN         6

2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8

3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6

4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6


        InvoiceDate  UnitPrice  CustomerID         Country
0  01-12-2010 08:26       2.55     17850.0  United Kingdom
1  01-12-2010 08:26       3.39     17850.0  United Kingdom
2  01-12-2010 08:26       2.75     17850.0  United Kingdom
3  01-12-2010 08:26       3.39     17850.0  United Kingdom
4  01-12-2010 08:26       3.39     17850.0  United Kingdom
```

```
data.shape
```

```
(406829, 8)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 406829 entries, 0 to 541908
```

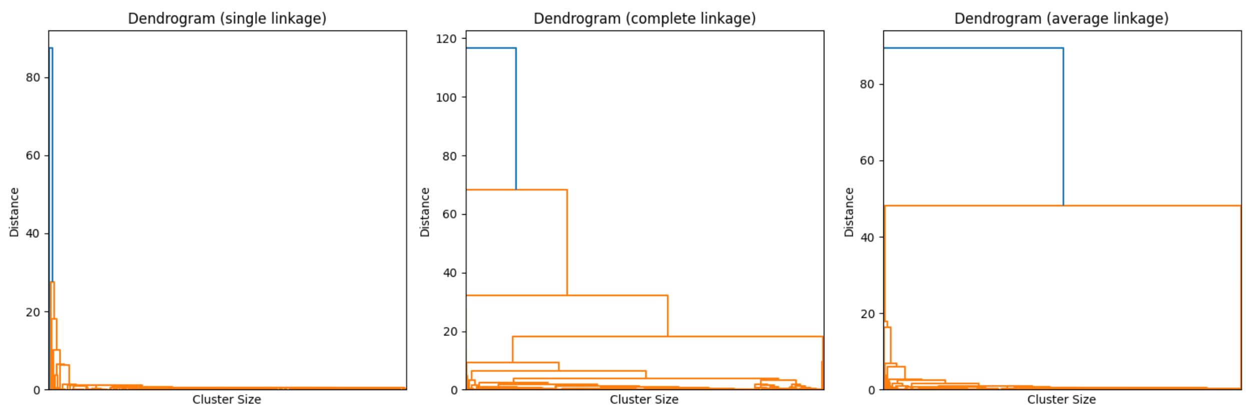
```
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    406829 non-null  object
 1   StockCode    406829 non-null  object
 2   Description  406829 non-null  object
 3   Quantity     406829 non-null  int64
 4   InvoiceDate  406829 non-null  object
 5   UnitPrice    406829 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      406829 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 27.9+ MB
```

```
for items in list(data.columns):
  if data[items].dtype == 'object':
    print(data[items].unique())
```

```
['536365' '536366' '536367' ... '581585' '581586' '581587']
['85123A' '71053' '84406B' ... '90214Z' '90089' '23843']
['WHITE HANGING HEART T-LIGHT HOLDER' 'WHITE METAL LANTERN'
 'CREAM CUPID HEARTS COAT HANGER' ... 'PINK CRYSTAL SKULL PHONE CHARM'
 'CREAM HANGING HEART T-LIGHT HOLDER' 'PAPER CRAFT , LITTLE BIRDIE']
['01-12-2010 08:26' '01-12-2010 08:28' '01-12-2010 08:34' ...
 '09-12-2011 12:31' '09-12-2011 12:49' '09-12-2011 12:50']
['United Kingdom' 'France' 'Australia' 'Netherlands' 'Germany'
'Norway'
 'EIRE' 'Switzerland' 'Spain' 'Poland' 'Portugal' 'Italy' 'Belgium'
 'Lithuania' 'Japan' 'Iceland' 'Channel Islands' 'Denmark' 'Cyprus'
 'Sweden' 'Austria' 'Israel' 'Finland' 'Greece' 'Singapore' 'Lebanon'
 'United Arab Emirates' 'Saudi Arabia' 'Czech Republic' 'Canada'
 'Unspecified' 'Brazil' 'USA' 'European Community' 'Bahrain' 'Malta'
'RSA']
```

```
df = data.iloc[:10000, :]

X = df[['Quantity', 'UnitPrice']]

# Standardize the feature matrix (important for hierarchical
clustering)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Define the number of clusters (k=3)
n_clusters = 3

# Apply Hierarchical Clustering with different linkage methods
linkage_methods = ['single', 'complete', 'average']
labels = {}

for method in linkage_methods:
```

```
    Z = linkage(X_scaled, method=method, metric='euclidean')
    labels[method] =
AgglomerativeClustering(n_clusters=n_clusters).fit_predict(Z)

# Plot the dendrograms
plt.figure(figsize=(15, 5))
for i, method in enumerate(linkage_methods):
    plt.subplot(1, 3, i + 1)
    plt.title(f'Dendrogram ({method} linkage)')
    dendrogram(linkage(X_scaled, method=method, metric='euclidean'),
no_labels=True, truncate_mode='level')
    plt.xlabel('Cluster Size')
    plt.ylabel('Distance')
plt.tight_layout()
plt.show()
```



1. Single Linkage: This method computes the distance between the closest points in the clusters. It tends to create long, chain-like clusters and is sensitive to outliers. Use cases include cases where clusters are expected to be non-convex and where there may be noise in the data.

2. Complete Linkage: This method computes the distance between the farthest points in the clusters. It tends to create compact, spherical clusters. It's suitable for situations where we expect well-separated, compact clusters in our data.

3. Average Linkage: This method calculates the average distance between all pairs of data points in the clusters. It is a compromise between single and complete linkage and is less sensitive to outliers. Average linkage is often a good choice when we have data with varying cluster shapes and sizes.

By visualizing the dendrograms and the resulting clusters, we can assess which linkage method is most suitable for our specific Online Retail dataset, depending on the characteristics of our data and our clustering objectives.