In [1]:
```
#import libraries
import numpy as np
import pandas as pd
import scipy
import matplotlib.pyplot as plt
import seaborn as sns
color = sns.color_palette()
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import KMeans
import scipy.cluster.hierarchy as shc
from sklearn.decomposition import PCA
from sklearn.cluster import AgglomerativeClustering
```

In [2]:
```
#Load the dataset
df = pd.read_csv("medical_clean.csv")
df.head()
```

Out[2]:

	CaseOrder	Customer_id	Interaction	UID	City	State
0	1	C412403	8cd49b13-f45a-4b47-a2bd-173ffa932c2f	3a83ddb66e2ae73798bdf1d705dc0932	Eva	AL
1	2	Z919181	d2450b70-0337-4406-bdbb-bc1037f1734c	176354c5eef714957d486009feabf195	Marianna	FL
2	3	F995323	a2057123-abf5-4a2c-abad-8ffe33512562	e19a0fa00aeda885b8a436757e889bc9	Sioux Falls	SD
3	4	A879973	1dec528d-eb34-4079-adce-0d7a40e82205	cd17d7b6d152cb6f23957346d11c3f07	New Richland	MN
4	5	C544523	5885f56b-d6da-43a3-8760-83583af94266	d2f0425877b10ed6bb381f3e2579424a	West Point	VA

5 rows × 50 columns

In [3]:
```
# General info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   CaseOrder           10000 non-null  int64  
 1   Customer_id         10000 non-null  object 
 2   Interaction         10000 non-null  object 
 3   UID                 10000 non-null  object 
 4   City                10000 non-null  object 
 5   State               10000 non-null  object 
 6   County              10000 non-null  object 
 7   Zip                 10000 non-null  int64  
 8   Lat                 10000 non-null  float64
 9   Lng                 10000 non-null  float64
 10  Population          10000 non-null  int64  
 11  Area                10000 non-null  object 
 12  TimeZone            10000 non-null  object 
 13  Job                 10000 non-null  object 
 14  Children            10000 non-null  int64  
 15  Age                 10000 non-null  int64  
 16  Income              10000 non-null  float64
 17  Marital             10000 non-null  object 
 18  Gender              10000 non-null  object 
 19  ReAdmis             10000 non-null  object 
 20  VitD_levels         10000 non-null  float64
 21  Doc_visits          10000 non-null  int64  
 22  Full_meals_eaten    10000 non-null  int64  
 23  vitD_supp           10000 non-null  int64  
 24  Soft_drink          10000 non-null  object 
 25  Initial_admin       10000 non-null  object 
 26  HighBlood           10000 non-null  object 
 27  Stroke              10000 non-null  object 
 28  Complication_risk   10000 non-null  object 
 29  Overweight          10000 non-null  object 
 30  Arthritis           10000 non-null  object 
 31  Diabetes            10000 non-null  object 
 32  Hyperlipidemia      10000 non-null  object 
 33  BackPain            10000 non-null  object 
 34  Anxiety             10000 non-null  object 
 35  Allergic_rhinitis   10000 non-null  object 
 36  Reflux_esophagitis  10000 non-null  object 
 37  Asthma              10000 non-null  object 
 38  Services            10000 non-null  object 
 39  Initial_days        10000 non-null  float64
 40  TotalCharge         10000 non-null  float64
 41  Additional_charges  10000 non-null  float64
 42  Item1               10000 non-null  int64  
 43  Item2               10000 non-null  int64  
 44  Item3               10000 non-null  int64  
 45  Item4               10000 non-null  int64  
 46  Item5               10000 non-null  int64  
 47  Item6               10000 non-null  int64  
 48  Item7               10000 non-null  int64  
 49  Item8               10000 non-null  int64  
```

```
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

In [4]:
```
# Lowering upper characters and replacing blanks by underscores
df.columns = [x.lower().replace(" ", "_") for x in df.columns]
```

In [5]:
```
# Get overview of descriptive stats
df.describe()
```

Out[5]:

	caseorder	zip	lat	lng	population	children	
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	100
mean	5000.50000	50159.323900	38.751099	-91.243080	9965.253800	2.097200	
std	2886.89568	27469.588208	5.403085	15.205998	14824.758614	2.163659	
min	1.00000	610.000000	17.967190	-174.209700	0.000000	0.000000	
25%	2500.75000	27592.000000	35.255120	-97.352982	694.750000	0.000000	
50%	5000.50000	50207.000000	39.419355	-88.397230	2769.000000	1.000000	
75%	7500.25000	72411.750000	42.044175	-80.438050	13945.000000	3.000000	
max	10000.00000	99929.000000	70.560990	-65.290170	122814.000000	10.000000	

8 rows × 23 columns

In [6]:
```
# Checking for null values
df.isnull()
```

Out[6]:

	caseorder	customer_id	interaction	uid	city	state	county	zip	lat	lng	...	to
0	False	False	False	False	False	False	False	False	False	False	...	
1	False	False	False	False	False	False	False	False	False	False	...	
2	False	False	False	False	False	False	False	False	False	False	...	
3	False	False	False	False	False	False	False	False	False	False	...	
4	False	False	False	False	False	False	False	False	False	False	...	
...	...	...	...	...	...	...	...	...	...	...	...	
9995	False	False	False	False	False	False	False	False	False	False	...	
9996	False	False	False	False	False	False	False	False	False	False	...	
9997	False	False	False	False	False	False	False	False	False	False	...	
9998	False	False	False	False	False	False	False	False	False	False	...	
9999	False	False	False	False	False	False	False	False	False	False	...	

10000 rows × 50 columns

In [7]:
```
#drop columns not being used
to_drop = ['caseorder', 'customer_id', 'uid', 'age', 'interaction', 'readmis',
df.drop(to_drop, inplace=True, axis=1)
```

In [8]:
```
df.shape
```

Out[8]:
```
(10000, 5)
```

In [9]:
```
#Change floats to integers
df['income'] = df['income'].astype(int)
df['vitd_levels'] = df['vitd_levels'].astype(int)
df['initial_days'] = df['initial_days'].astype(int)
df['totalcharge'] = df['totalcharge'].astype(int)
df['full_meals_eaten'] = df['full_meals_eaten'].astype(int)
```

In [10]:
```
df.columns
```

Out[10]:
```
Index(['income', 'vitd_levels', 'full_meals_eaten', 'initial_days',
       'totalcharge'],
      dtype='object')
```

In [11]:
```
# General info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   income            10000 non-null  int32
 1   vitd_levels       10000 non-null  int32
 2   full_meals_eaten  10000 non-null  int32
 3   initial_days      10000 non-null  int32
 4   totalcharge       10000 non-null  int32
dtypes: int32(5)
memory usage: 195.4 KB
```

In [12]:
```
# Checking for null values
df.isnull()
```

Out[12]:

	income	vitd_levels	full_meals_eaten	initial_days	totalcharge
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
9995	False	False	False	False	False
9996	False	False	False	False	False
9997	False	False	False	False	False
9998	False	False	False	False	False
9999	False	False	False	False	False

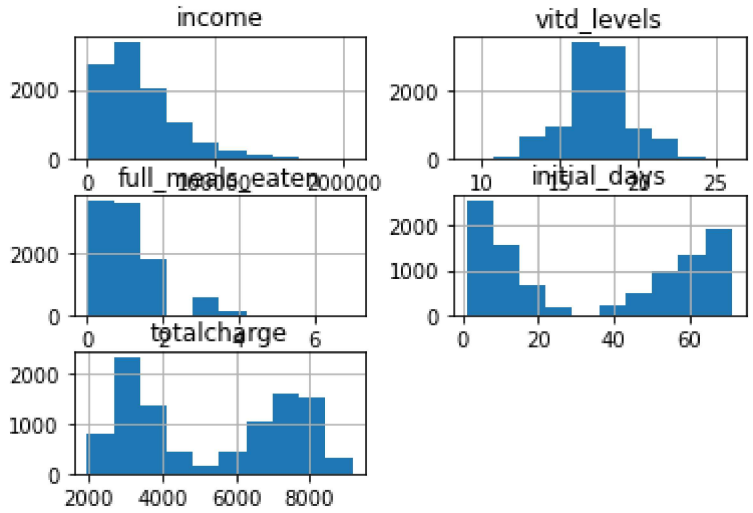
10000 rows × 5 columns

In [13]:
```
#Change floats to integers
df['income'] = df['income'].astype(int)
df['vitd_levels'] = df['vitd_levels'].astype(int)
df['initial_days'] = df['initial_days'].astype(int)
df['totalcharge'] = df['totalcharge'].astype(int)
df['full_meals_eaten'] = df['full_meals_eaten'].astype(int)
```

In [14]:
```
#save prepared data
df.to_csv('Documents/PreparedDataD212.csv')
```

In [15]:
```
df.hist()
```

Out[15]:
```
array([[<AxesSubplot:title={'center':'income'}>,
        <AxesSubplot:title={'center':'vitd_levels'}>],
       [<AxesSubplot:title={'center':'full_meals_eaten'}>,
        <AxesSubplot:title={'center':'initial_days'}>],
       [<AxesSubplot:title={'center':'totalcharge'}>, <AxesSubplot:>]],
      dtype=object)
```



In [16]:
```
# transpose to compare variables
df.describe().transpose()
```

Out[16]:

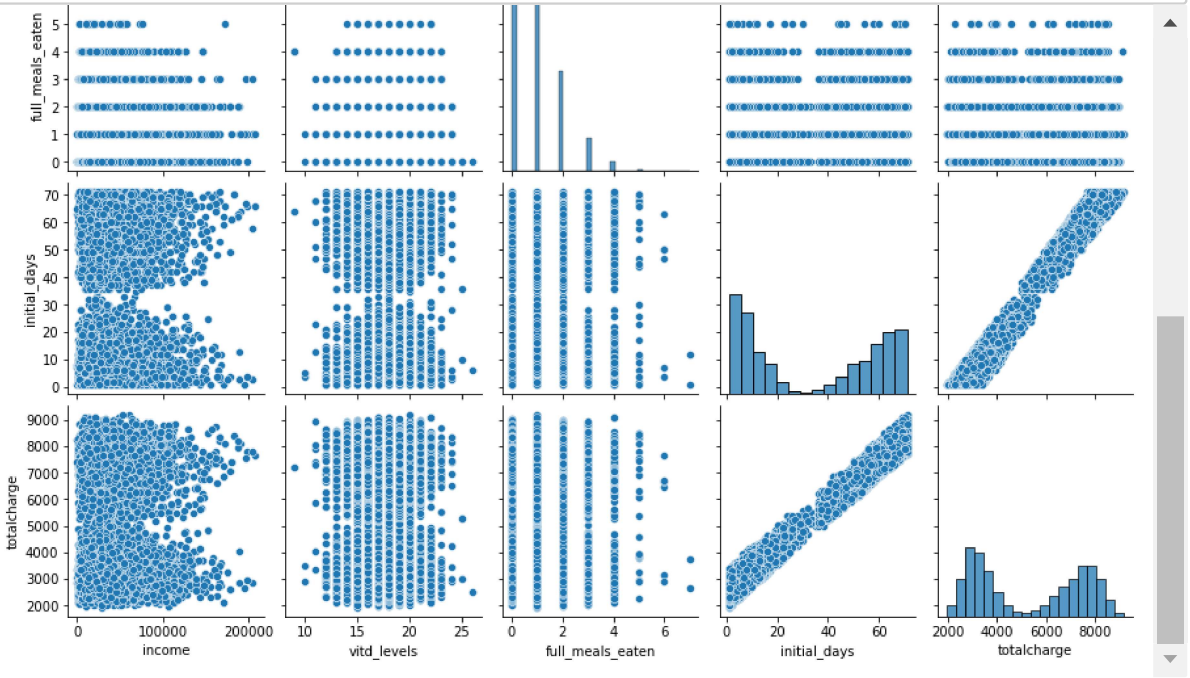
	count	mean	std	min	25%	50%	75%	ma
income	10000.0	40490.0021	28521.152883	154.0	19598.25	33768.0	54295.75	207249
vitd_levels	10000.0	17.4612	2.040370	9.0	16.00	17.0	19.00	26
full_meals_eaten	10000.0	1.0014	1.008117	0.0	0.00	1.0	2.00	7
initial_days	10000.0	33.9560	26.301628	1.0	7.00	35.5	61.00	71
totalcharge	10000.0	5311.6735	2180.391406	1938.0	3179.00	5213.5	7459.25	9180

In [17]:
```
df.head()
```

Out[17]:

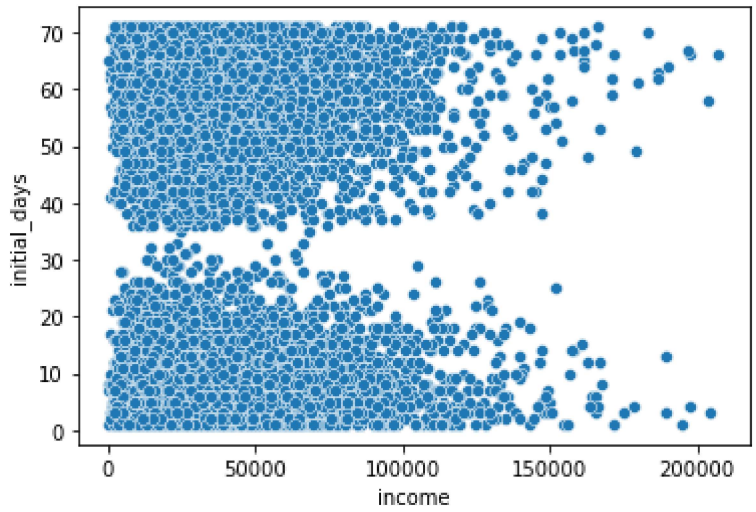
	income	vitd_levels	full_meals_eaten	initial_days	totalcharge
0	86575	19	0	10	3726
1	46805	18	2	15	4193
2	14370	18	1	4	2434
3	39741	16	1	1	2127
4	1209	17	0	1	2113

In [18]:
```
#plotting the variables with pair plot
sns.pairplot(df)
```



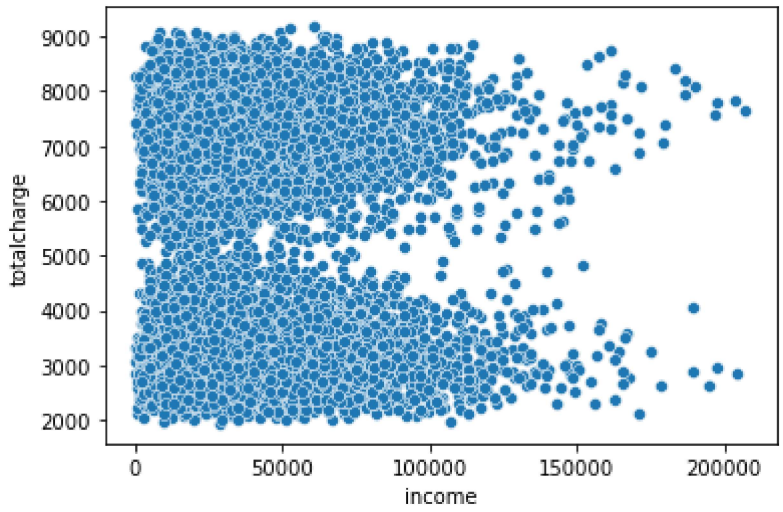
In [19]:
```
#look at similar variables
sns.scatterplot(x=df['income'],
                y=df['initial_days'])
```

Out[19]: <AxesSubplot:xlabel='income', ylabel='initial_days'>



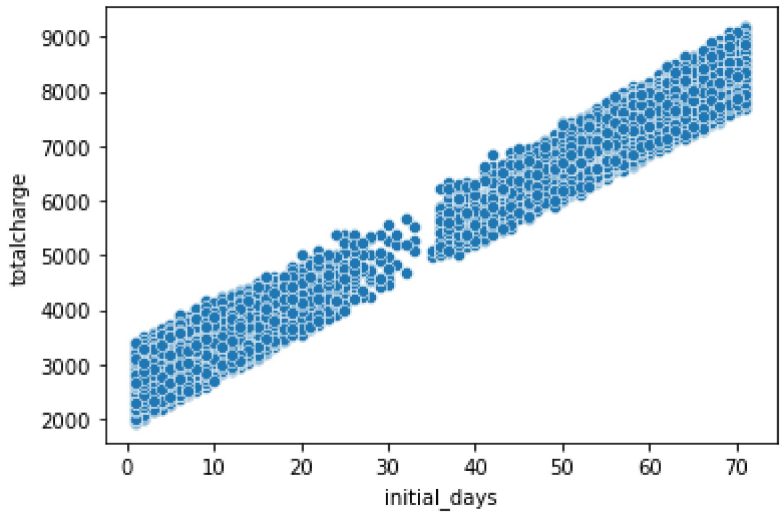
In [20]:
```
#Look at similar variables
sns.scatterplot(x=df['income'],
                y=df['totalcharge'])
```

Out[20]: <AxesSubplot:xlabel='income', ylabel='totalcharge'>

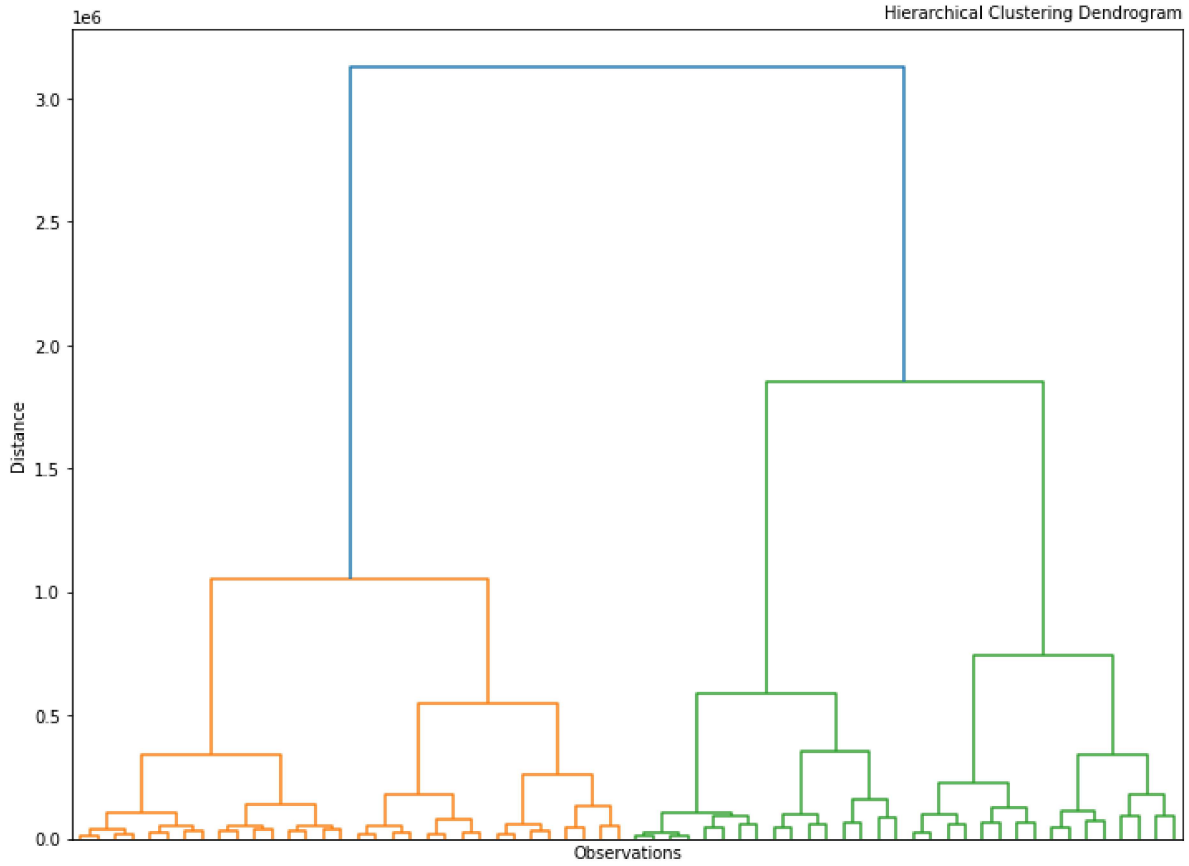


In [21]:
```
#Look at similar variables
sns.scatterplot(x=df['initial_days'],
                y=df['totalcharge'])
```

Out[21]: <AxesSubplot:xlabel='initial_days', ylabel='totalcharge'>



In [22]:
```
#dendrogram before PCA
# Building the Dendrogram
hier_clust = linkage(df, metric = "euclidean", method = "ward")
plt.figure(figsize= (12, 9))
plt.title("Hierarchical Clustering Dendrogram", fontsize=10, loc='right')
plt.xlabel("Observations")
plt.ylabel("Distance")
dendrogram(hier_clust, truncate_mode = "level", p = 5, show_leaf_counts = Fals
plt.show()
```

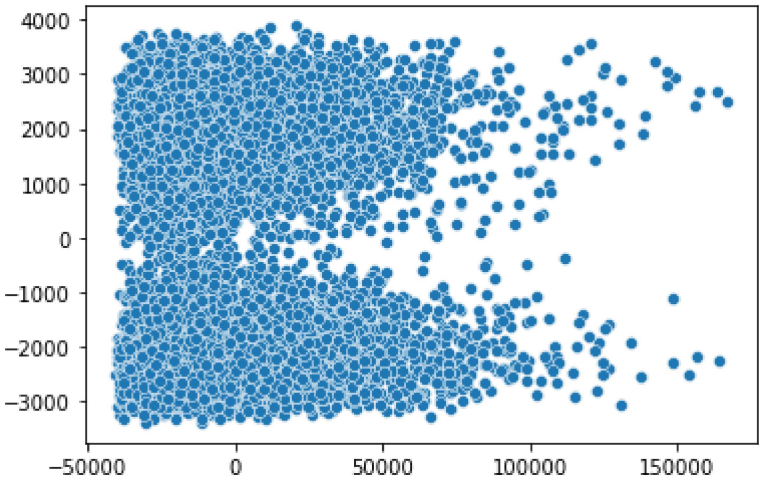


In [23]:
```
#pca fit and transform

pca = PCA(n_components=5)
pca.fit_transform(df)
pca.explained_variance_ratio_.cumsum()
```

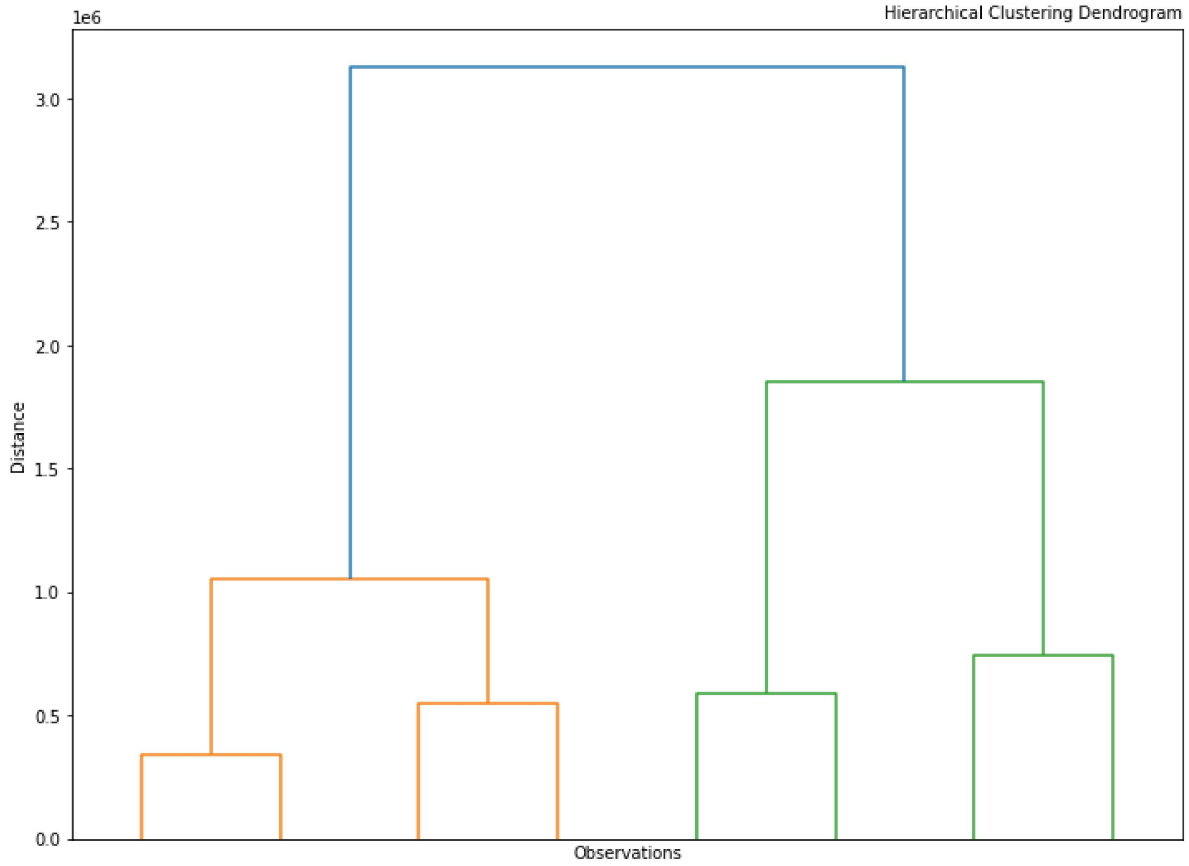
Out[23]:
```
array([0.99418998, 0.99999997, 0.99999999, 1.        , 1.        ])
```

In [24]:
```
#pca steps

pca = PCA(n_components=2)
pcs = pca.fit_transform(df)

pc1_values = pcs[:,0]
pc2_values = pcs[:,1]
sns.scatterplot(x=pc1_values, y=pc2_values)
```

Out[24]: <AxesSubplot:>



In [25]:
```
#dendrogram after PCA
# Building the Dendrogram
hier_clust = linkage(df, metric = "euclidean", method = "ward")
plt.figure(figsize= (12, 9))
plt.title("Hierarchical Clustering Dendrogram", fontsize=10, loc='right')
plt.xlabel("Observations")
plt.ylabel("Distance")
dendrogram(hier_clust, truncate_mode = "level", p = 2, show_leaf_counts = Fals
plt.show()
```

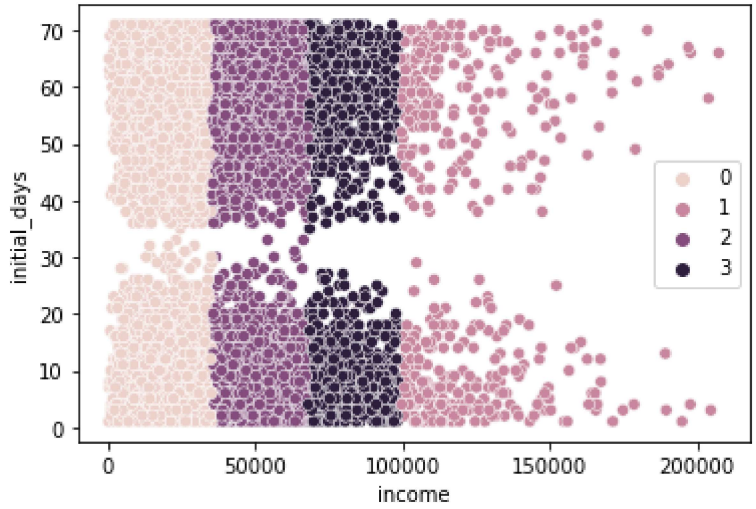


In [26]:
```
# AgglomerativeClustering

clustering_model = AgglomerativeClustering(n_clusters=4, affinity='euclidean',
clustering_model.fit(df)
clustering_model.labels_
```

Out[26]:
```
array([3, 2, 0, ..., 2, 0, 2], dtype=int64)
```

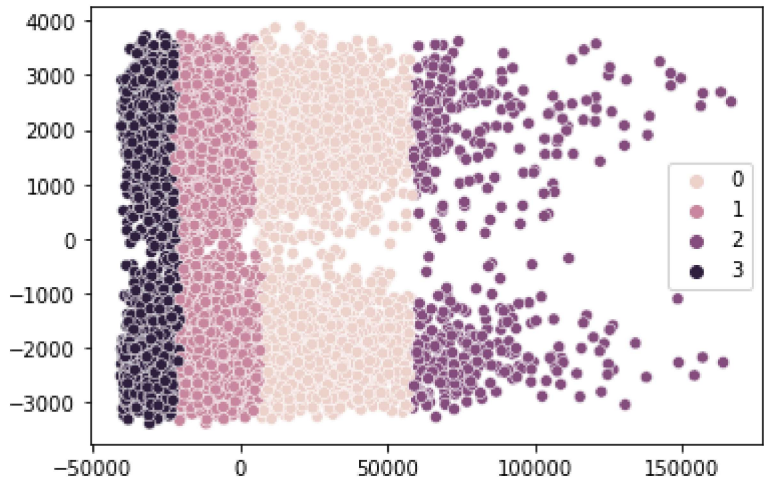
In [27]:
```
#scatter plot before PCA
data_labels = clustering_model.labels_
sns.scatterplot(x='income',
                y='initial_days',
                data=df,
                hue=data_labels)
```

Out[27]: <AxesSubplot:xlabel='income', ylabel='initial_days'>



In [28]:
```
#scatterplot after PCA
clustering_model_pca = AgglomerativeClustering(n_clusters=4, affinity='euclide
clustering_model_pca.fit(pcs)

data_labels_pca = clustering_model_pca.labels_

sns.scatterplot(x=pc1_values,
                y=pc2_values,
                hue=data_labels_pca)
```

Out[28]: <AxesSubplot:>



In [ ]: