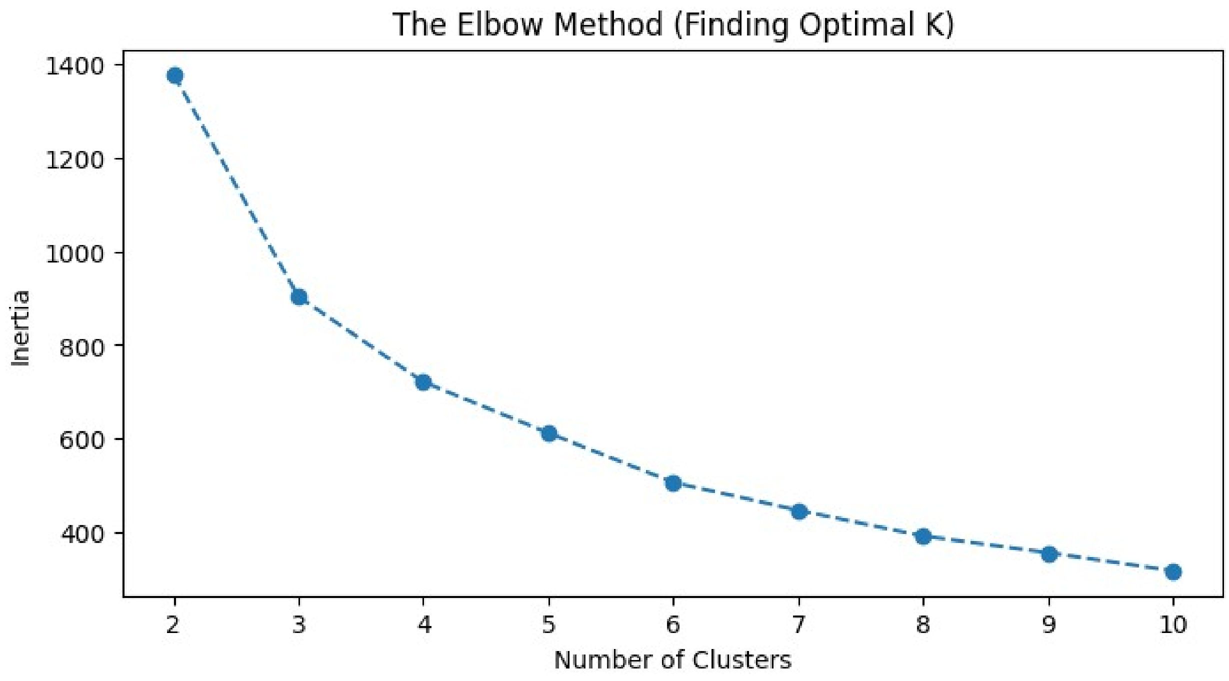
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import silhouette_score

# Generating 1000 synthetic customers
np.random.seed(42)
n_customers = 1000

data = {
    'Recency': np.random.exponential(30, n_customers), # Days since
last purchase
    'Frequency': np.random.poisson(5, n_customers),    # Number of
orders
    'Monetary': np.random.lognormal(5, 1, n_customers), # Total Spend
    'Avg_Ticket': np.random.normal(100, 20, n_customers) # Avg order
value
}

df = pd.DataFrame(data)
print("Data Head:\n", df.head())
```

```
Data Head:
      Recency  Frequency    Monetary  Avg_Ticket
0  14.078043          8  299.808082  124.545476
1  90.303643          4  252.953915   75.796790
2  39.502371          5  175.108129  108.771586
3  27.388277         10   37.262313   72.599467
4   5.088746          3   91.504016  121.307525
```

```python
# 1. Log Transform to handle skewness
df_log = np.log1p(df)

# 2. Standardize
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_log)

# 3. PCA - Dimensionality Reduction
# Here is where we define the variable that caused your error!
pca = PCA(n_components=2)
pca_transformed_data = pca.fit_transform(df_scaled)

print("PCA successful. Shape:", pca_transformed_data.shape)
```

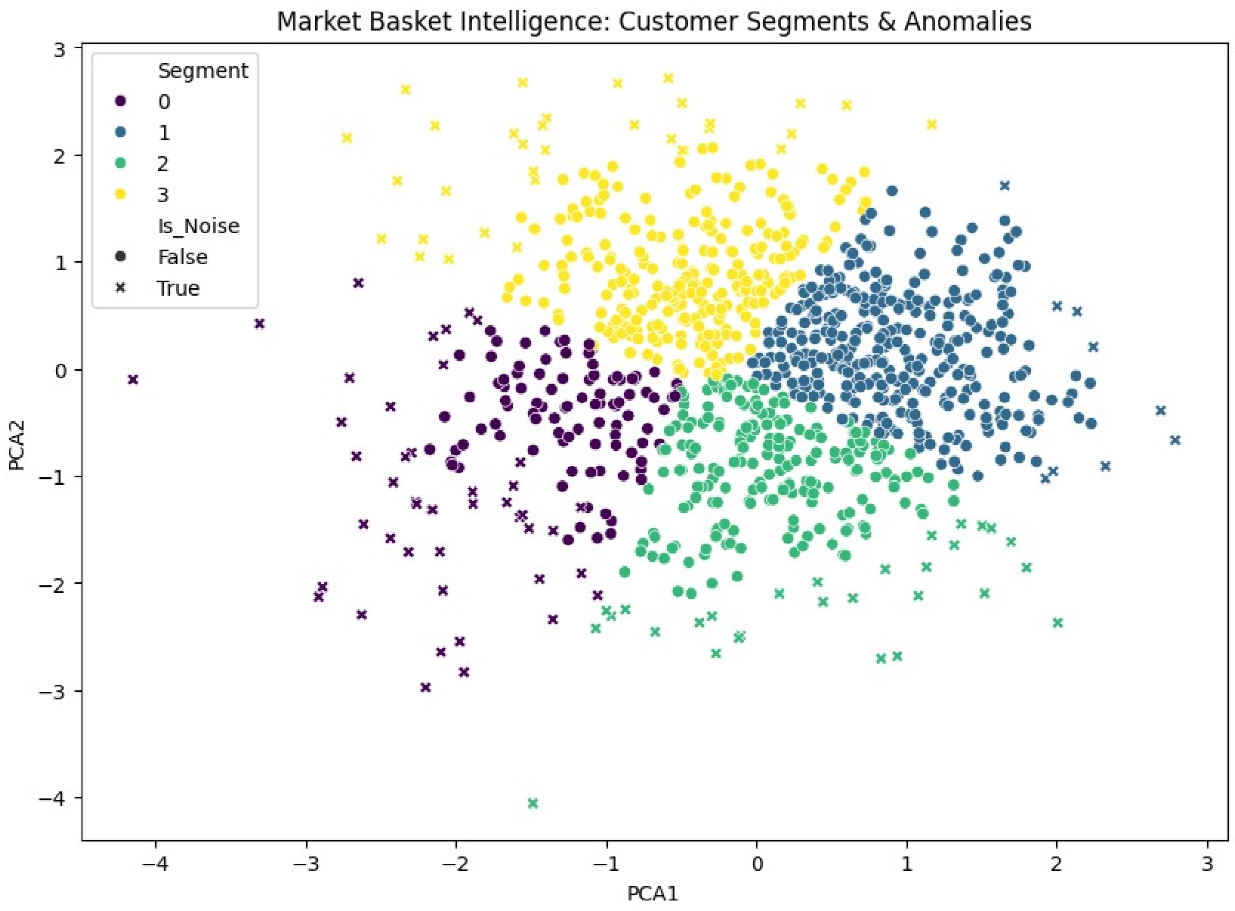
```
PCA successful. Shape: (1000, 2)
```

```python
inertia = []
K_range = range(2, 11)

for k in K_range:
    # We use pca_transformed_data defined in the previous step
    km = KMeans(n_clusters=k, init='k-means++', n_init=10,
random_state=42)
    km.fit(pca_transformed_data)
    inertia.append(km.inertia_)

# Plotting the Elbow
plt.figure(figsize=(8, 4))
plt.plot(K_range, inertia, marker='o', linestyle='--')
plt.title('The Elbow Method (Finding Optimal K)')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()
```



```python
# Detect anomalies using DBSCAN
dbscan = DBSCAN(eps=0.3, min_samples=10)
dbscan_labels = dbscan.fit_predict(pca_transformed_data)

# Add results back to main dataframe
df['PCA1'] = pca_transformed_data[:, 0]
df['PCA2'] = pca_transformed_data[:, 1]
df['Is_Noise'] = (dbscan_labels == -1)

# Final K-Means with k=4 (typical for e-commerce)
```

```
kmeans = KMeans(n_clusters=4, random_state=42)
df['Segment'] = kmeans.fit_predict(pca_transformed_data)

plt.figure(figsize=(10, 7))
sns.scatterplot(data=df, x='PCA1', y='PCA2', hue='Segment',
palette='viridis', style='Is_Noise')
plt.title('Market Basket Intelligence: Customer Segments & Anomalies')
plt.show()

# Business Summary
print("\n--- Cluster Profile Summary ---")
print(df.groupby('Segment')[['Recency', 'Frequency',
'Monetary']].mean())
```



```
--- Cluster Profile Summary ---
            Recency  Frequency    Monetary
Segment
0         17.969306   3.034965  123.940752
1         38.117509   6.101156  311.243041
```

```
2         45.498552   3.400000  364.686630
3         10.506201   5.868327  109.502710
```

```
# Create a 'Non-Actionable' flag
# We exclude:
# 1. DBSCAN Outliers (The 'Noise')
# 2. Customers with only 1 purchase (Low Frequency) AND high recency
(Lapsed)

df['Actionable'] = True

# Mark Noise as non-actionable
df.loc[df['Is_Noise'] == True, 'Actionable'] = False

# Mark "One-Time Lapsed" shoppers as non-actionable
# (e.g., Bought once more than 90 days ago)
df.loc[(df['Frequency'] <= 1) & (df['Recency'] > 90), 'Actionable'] =
False

# Only perform Market Basket Analysis on Actionable segments
actionable_customers = df[df['Actionable'] == True]

# Calculate Silhouette Score to prevent over-clustering
score = silhouette_score(pca_transformed_data, df['Segment'])
print(f"Silhouette Score for {kmeans.n_clusters} clusters:
{score:.2f}")

if score < 0.3:
    print("Warning: Clusters are overlapping. Consider reducing K or
increasing Denoising.")
```

```
Silhouette Score for 4 clusters: 0.33
```