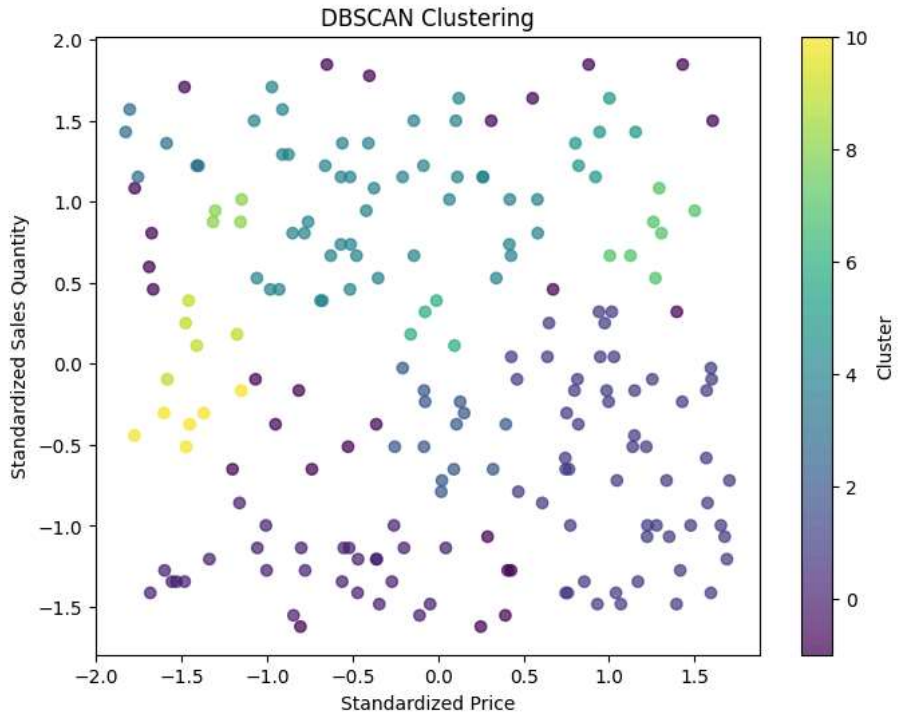
```
import pandas as pd
import numpy as np
import random
import datetime

np.random.seed(0)
random.seed(0)

Categories = ['Electronic' ,'Home Appliances' , 'Clothing ', 'Home Decor']
Brands = ['Samsung','Apple','LG','Nike','Sony', 'Adidas','IKEA','Google','Whirlpool','Home Depot']
Seasons = ['Summer','Winter','Spring','Fall']
Data = []
Starting_date = datetime.datetime(2020,1,1)

for i in range(200):
    Product_ID = i + 1
    Category = random.choice(Categories)
    Brand = random.choice(Brands)
    Price = round(random.uniform(10, 200), 2)
    Sales_quantity = random.randint(0, 50)
    Season = random.choice(Seasons)  # Choose a single season randomly
    date = Starting_date + datetime.timedelta(days=i)
    Data.append([Product_ID, Category, Brand, Price, Sales_quantity, Season, date])

df = pd.DataFrame(Data, columns=['Product_ID', 'Category', 'Brand', 'Price', 'Sales_quantity', 'Season', 'date'])
df.to_csv('sales_data.csv', index=False)

#Loading the data
import pandas as pd
df = pd.read_csv('/content/sales_data.csv')
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_score
# Feature scaling
scaler = StandardScaler()
scaled_df = scaler.fit_transform(df[['Price', 'Sales_quantity']])

# Initialize DBSCAN with adjusted parameters
dbscan = DBSCAN(eps=0.3, min_samples=5)  # Adjust parameters as needed

# Fit the model
dbscan.fit(scaled_df)

# Assign cluster labels to the data
df['Cluster'] = dbscan.labels_

# Check the number of unique clusters (including noise, labeled as -1)
unique_clusters = len(set(df['Cluster'])) - (1 if -1 in df['Cluster'] else 0)
print(f"Number of clusters: {unique_clusters}")

if unique_clusters > 1:
    # Scatter plot to visualize clusters
    plt.figure(figsize=(8, 6))
    plt.scatter(scaled_df[:, 0], scaled_df[:, 1], c=df['Cluster'], cmap='viridis', alpha=0.7)
    plt.xlabel('Standardized Price')
    plt.ylabel('Standardized Sales Quantity')
    plt.title('DBSCAN Clustering')
    plt.colorbar(label='Cluster')
    plt.show()

    # Calculate Silhouette Score
    silhouette_avg = silhouette_score(scaled_df, df['Cluster'])
    print(f"Silhouette Score: {silhouette_avg:.4f}")
else:
    print("Not enough clusters to calculate silhouette score. Try adjusting the parameters.")
```

```
Number of clusters: 12
```




```
Silhouette Score: 0.0963
```