```
import pandas as pd
import sqlite3

df = pd.read_csv("Sample - Superstore.csv", encoding="ISO-8859-1")

conn = sqlite3.connect(r"C:\Users\HP\Downloads\sqlite-tools-win-x64-3490200\project1.sqlite")

df.to_sql("sales_data", conn, if_exists="replace", index=False)

print("Row count:", pd.read_sql_query("SELECT COUNT(*) FROM sales_data", conn))

conn.close()
```

```
Row count:    COUNT(*)
0      9994
```

```
df['Turnover_Rate'] = df['Quantity'] / (df['Sales'] + 1e-6)
df['Days_Inventory'] = 365 / df['Turnover_Rate']
df.head()
```

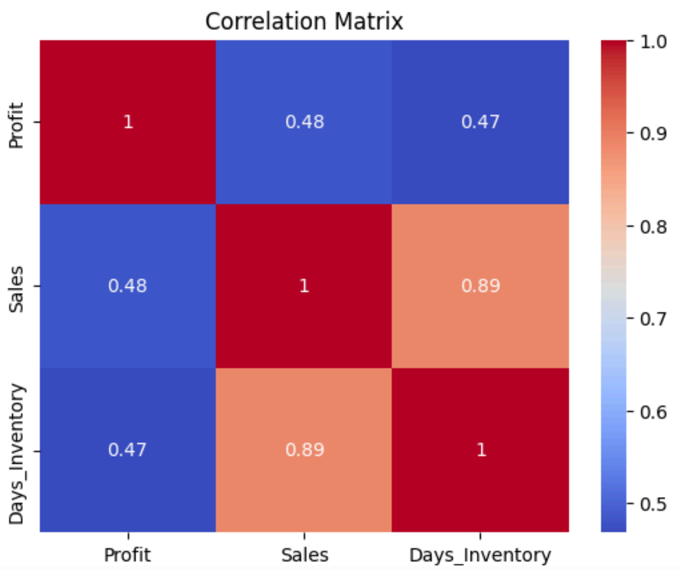
	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Product ID	Category	Sub-Category	I
0	1	CA-2016-152156	11-08-2016	11-11-2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	FUR-BO-10001798	Furniture	Bookcases	S C B
1	2	CA-2016-152156	11-08-2016	11-11-2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	FUR-CH-10000454	Furniture	Chairs	Hor Uph : (
2	3	CA-2016-138688	06-12-2016	6/16/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	OFF-LA-10000240	Office Supplies	Labels	A La Typ
3	4	US-2015-108966	10-11-2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	FUR-TA-10000577	Furniture	Tables	Se Rec
4	5	US-2015-108966	10-11-2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	OFF-ST-10000760	Office Supplies	Storage	El 'N I

5 rows × 23 columns

1) Calculates how quickly products are sold relative to their value. A higher turnover rate means faster movement of stock. The small 1e-6 prevents division by zero errors.

2) Estimates how many days, on average, it takes to sell the inventory. Useful for identifying slow-moving or overstocked items.

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(df[['Profit', 'Sales', 'Days_Inventory']].corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()
```



This heatmap visually displays the correlation coefficients between key numeric variables:

1) Profit

2) Sales

3) Days_Inventory

The values range from:

1) +1 (strong positive correlation)

2) 0 (no correlation)

3) -1 (strong negative correlation)

It helps identify relationships such as:

1) Whether higher sales lead to higher profit

2) Whether longer inventory days reduce profitability

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

features = df[['Profit', 'Sales', 'Days_Inventory']].fillna(0)
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(scaled_features)
df.head(3)
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Category	Sub-Category	Product Name	Sal
0	1	CA-2016-152156	11-08-2016	11-11-2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	Furniture	Bookcases	Bush Somerset Collection Bookcase	261.
1	2	CA-2016-152156	11-08-2016	11-11-2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	731.
2	3	CA-2016-138688	06-12-2016	6/16/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...	14.

3 rows × 24 columns

1) Selects the numeric variables to use for KMeans clustering. This includes: Profit, Sales, Days_Inventory.

2) NaN values are dropped to avoid model errors. Normalizes the data so that all features contribute equally to clustering. (Without this, larger values like Sales would dominate).

3) Groups the data into 3 clusters based on sales, profit, and inventory speed. Adds a Cluster column (0, 1, 2) to your dataset.

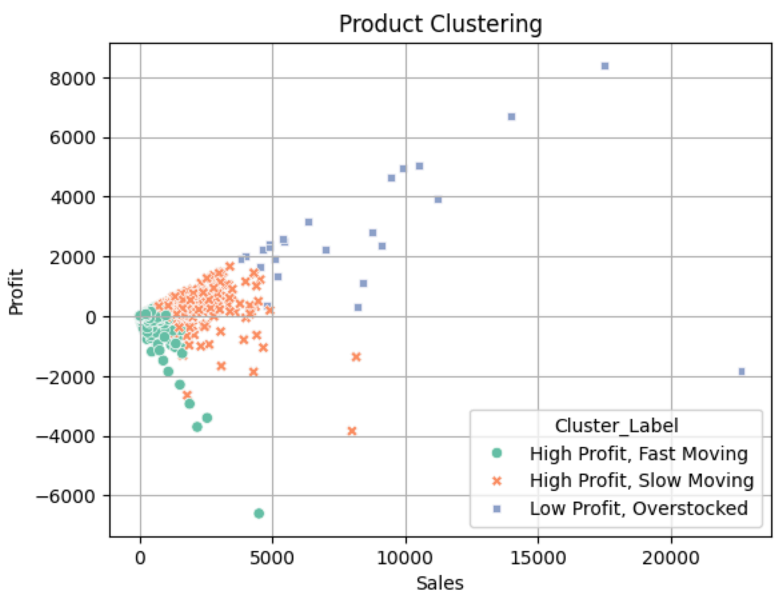
```
def interpret_cluster(row):
    if row['Cluster'] == 0:
        return 'High Profit, Fast Moving'
    elif row['Cluster'] == 1:
        return 'Low Profit, Overstocked'
    else:
        return 'High Profit, Slow Moving'

df['Cluster_Label'] = df.apply(interpret_cluster, axis=1)
df.head(3)
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Sub-Category	Product Name	Sales	Quanti
0	1	CA-2016-152156	11-08-2016	11-11-2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	Bookcases	Bush Somerset Collection Bookcase	261.96	
1	2	CA-2016-152156	11-08-2016	11-11-2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	731.94	
2	3	CA-2016-138688	06-12-2016	6/16/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	Labels	Self-Adhesive Address Labels for Typewriters b...	14.62	

3 rows × 25 columns

Maps numerical cluster IDs to meaningful business labels, making insights understandable and actionable for stakeholders.

```
sns.scatterplot(data=df, x='Sales', y='Profit', hue='Cluster_Label', style='Cluster_Label', palette='Set2')
plt.title("Product Clustering")
plt.xlabel("Sales")
plt.ylabel("Profit")
plt.grid(True)
plt.show()
```



This scatter plot visualizes products segmented by clusters on a Sales vs. Profit axis. Each point represents a product or transaction, and:

1) x = Sales → revenue generated

2) y = Profit → net gain or loss

3) hue and style = Cluster_Label → different clusters (e.g., High Profit, Slow Moving)

Purpose:

1) Visually separate product clusters for strategic decisions.

2) Identify outliers, underperformers, and top-sellers.

3) Confirm if your KMeans clustering meaningfully separates product behavior.