Task

Analyze the data in "/content/store_clusters.csv", understand the text, image, and geolocation features (implicitly represented in the clusters), and provide the following output: 1) Clusters with features, 2) Embed results in a graph UI (plotly or seaborn), and 3) Easy to deploy.

```
from google.colab import drive
drive.mount ('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
import numpy as np
import pandas as pd
import seaborn as sns
```

```
df = pd.read_csv("/content/drive/MyDrive/Store Demand Forecast Weekly.csv")
```

Load and inspect data

Subtask:

Load the /content/store_clusters.csv file into a pandas DataFrame and display the first few rows and data types to understand the data structure.

```
df.head()
```

	store_nbr	sell_quantity	unit_cost_amount	final_fcst_each_qty	total_sales	p
0	1	9.806667	NaN	2.789968	0.0	1.00
1	2	6.273333	NaN	3.679712	0.0	0.66
2	3	8.955000	NaN	2.156298	0.0	0.42
3	4	5.910000	NaN	2.926490	0.0	0.13
4	5	8.955000	NaN	2.198437	0.0	0.44

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
print(df.columns)
```

```
Index(['store_nbr', 'sell_quantity', 'unit_cost_amount', 'final_fcst_each_qty',
       'total_sales', 'pca_x', 'pca_y', 'cluster'],
```

```
dtype='object')
```

```
df.describe()
```

	store_nbr	sell_quantity	unit_cost_amount	final_fcst_each_qty	total_sales
count	4313.000000	4313.000000	0.0	4313.000000	4313.0
mean	2769.713888	7.553609	NaN	2.013194	0.0
std	1872.614837	2.908155	NaN	1.204241	0.0
min	1.000000	1.940000	NaN	0.031490	0.0
25%	1209.000000	5.910000	NaN	1.163365	0.0
50%	2491.000000	6.273333	NaN	1.773341	0.0
75%	4209.000000	9.105000	NaN	2.574744	0.0
max	8958.000000	24.000000	NaN	13.175962	0.0

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4313 entries, 0 to 4312
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   store_nbr            4313 non-null   int64  
 1   sell_quantity        4313 non-null   float64
 2   unit_cost_amount     0 non-null      float64
 3   final_fcst_each_qty  4313 non-null   float64
 4   total_sales          4313 non-null   float64
 5   pca_x                4313 non-null   float64
 6   pca_y                4313 non-null   float64
 7   cluster              4313 non-null   int64  
dtypes: float64(6), int64(2)
memory usage: 269.7 KB
```

```
duplicate_rows_count = df.duplicated().sum()
if duplicate_rows_count > 0:
    print(f"Total duplicate rows found: {duplicate_rows_count}")
else:
    print("No duplicate rows found.")
```

```
No duplicate rows found.
```

```
print(store_features.isna().sum())
print(pd.DataFrame(scaled_data).isna().sum())
```

```
store_nbr                0
sell_quantity            0
unit_cost_amount       4313
final_fcst_each_qty       0
```

```
total_sales              0
dtype: int64
0       0
1    4313
2       0
3       0
dtype: int64
```

```
# Convert columns to numeric (if they contain string numbers)
df['sell_quantity'] = pd.to_numeric(df['sell_quantity'], errors='coerce')
df['unit_cost_amount'] = pd.to_numeric(df['unit_cost_amount'], errors='coerce')
df['final_fcst_each_qty'] = pd.to_numeric(df['final_fcst_each_qty'], errors='coerce')
```

```
df['total_sales'] = df['sell_quantity'] * df['unit_cost_amount']
```

```
store_features = df.groupby('store_nbr').agg({
    'sell_quantity': 'mean',
    'unit_cost_amount': 'mean',
    'final_fcst_each_qty': 'mean',
    'total_sales': 'sum'
}).reset_index()

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaled_data = scaler.fit_transform(store_features[['sell_quantity', 'unit_cost_amount', '
```

```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(store_features[['sell_quantity', 'unit_cost_amount', '
```

```
df = df.replace([np.nan, -np.inf], 0)
print(df)
```

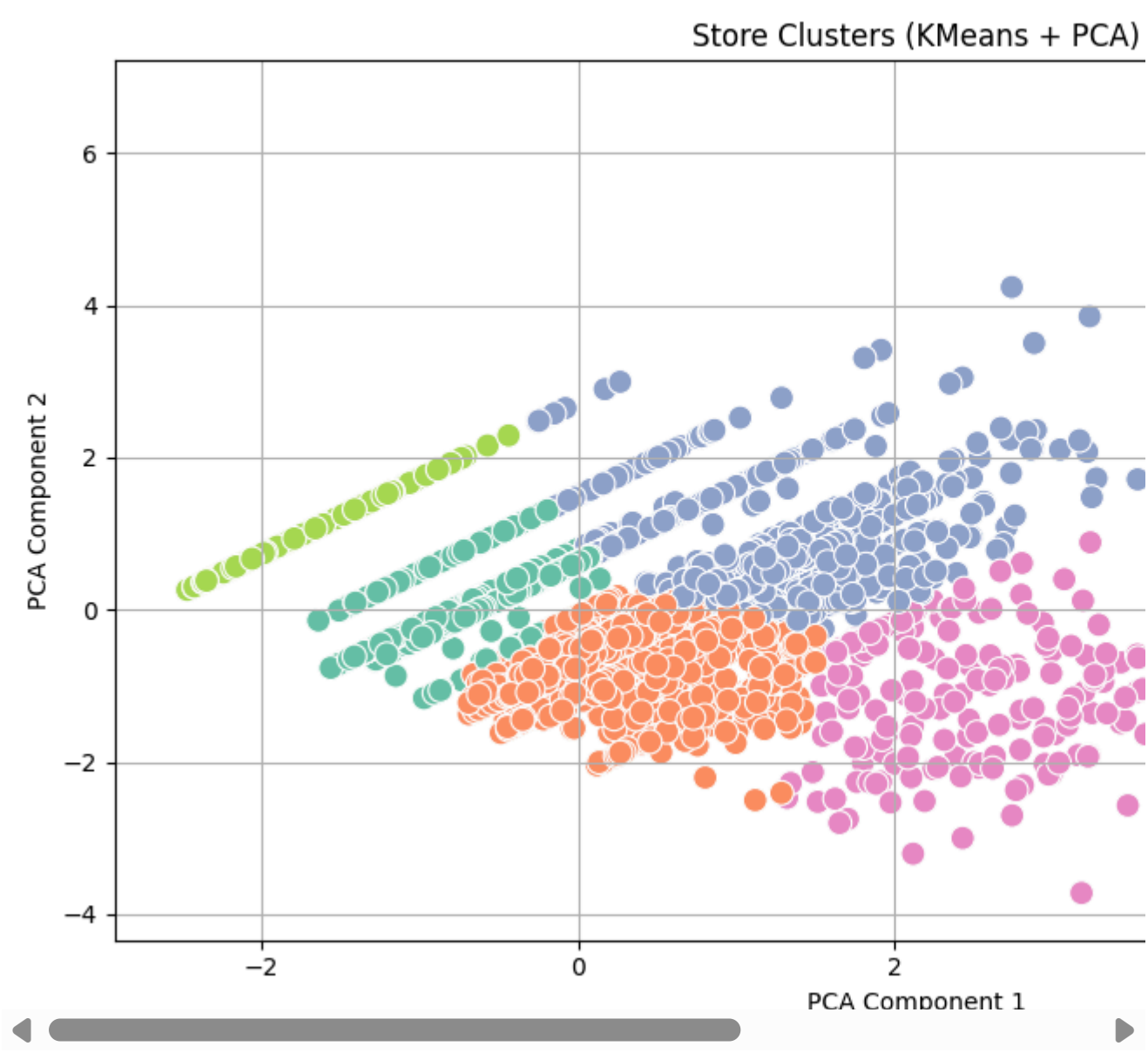
```
      store_nbr  sell_quantity  unit_cost_amount  final_fcst_each_qty  \
0             1       9.806667               0.0             2.789968
1             2       6.273333               0.0             3.679712
2             3       8.955000               0.0             2.156298
3             4       5.910000               0.0             2.926490
4             5       8.955000               0.0             2.198437
...         ...            ...               ...                  ...
4308       7601       6.273333               0.0             0.485481
4309       8331       9.488571               0.0             9.031126
4310       8861       6.273333               0.0             2.248429
4311       8930      12.000000               0.0             4.717067
4312       8958       6.273333               0.0             1.328141

      total_sales     pca_x     pca_y  cluster
0             0.0  1.004045 -0.091727        1
1             0.0  0.667330  1.289990        2
```

```
2             0.0  0.424820 -0.256745        1
3             0.0  0.136648  0.936015        0
4             0.0  0.449566 -0.231998        1
...           ...       ...       ...      ...
4308          0.0 -1.208477 -0.585817        0
4309          0.0  4.591803  3.650736        2
4310          0.0 -0.173189  0.449472        0
4311          0.0  2.669094  0.506597        2
4312          0.0 -0.713626 -0.090966        0

[4313 rows x 8 columns]
```

```
from sklearn.decomposition import PCA
from sklearn.impute import SimpleImputer

# Impute missing values with the mean
imputer = SimpleImputer(strategy='mean')
scaled_data_imputed = imputer.fit_transform(scaled_data)

pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data_imputed)

store_features['pca_x'] = pca_data[:, 0]
store_features['pca_y'] = pca_data[:, 1]


kmeans = KMeans(n_clusters=5, random_state=42)
store_features['cluster'] = kmeans.fit_predict(scaled_data_imputed)


store_features.to_csv("store_clusters.csv", index=False)
joblib.dump(kmeans, "kmeans_model.pkl")
```

```
['kmeans_model.pkl']
```

```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=store_features, x='pca_x', y='pca_y', hue='cluster', palette='Set2',
plt.title("Store Clusters (KMeans + PCA)")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.grid(True)
plt.tight_layout()
plt.savefig("cluster_plot.png")
plt.show()
```



Visualize clusters

Subtask:

Visualize the store clusters using a scatter plot with PCA components on the axes and cluster labels for color.

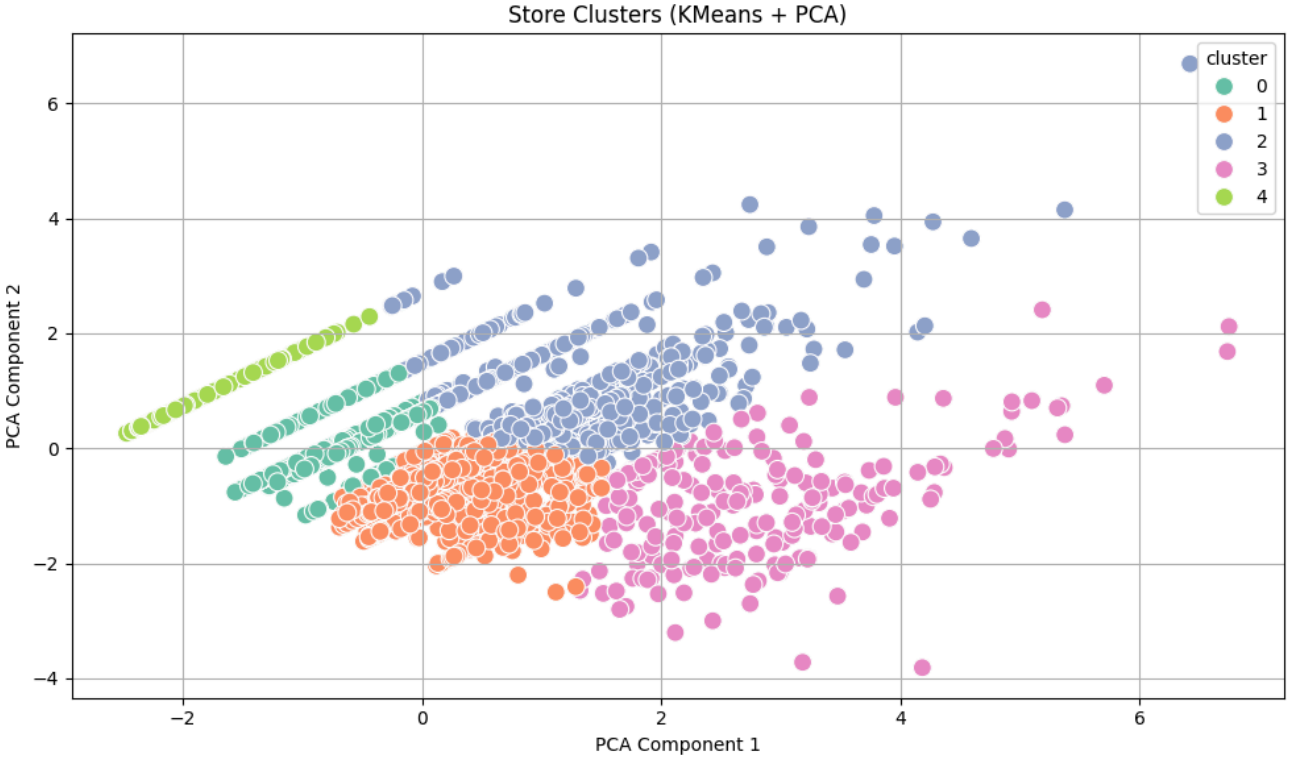
```
df_clusters = pd.read_csv('/content/store_clusters.csv')
print(df_clusters.head())
print(df_clusters.info())
```

```
   store_nbr  sell_quantity  unit_cost_amount  final_fcst_each_qty  \
0          1       9.806667               NaN             2.789968
1          2       6.273333               NaN             3.679712
2          3       8.955000               NaN             2.156298
3          4       5.910000               NaN             2.926490
4          5       8.955000               NaN             2.198437

   total_sales     pca_x     pca_y  cluster
0          0.0  1.004045 -0.091727        1
```

```
1          0.0  0.667330  1.289990        2
2          0.0  0.424820 -0.256745        1
3          0.0  0.136648  0.936015        2
4          0.0  0.449566 -0.231998        1
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4313 entries, 0 to 4312
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   store_nbr            4313 non-null   int64
 1   sell_quantity        4313 non-null   float64
 2   unit_cost_amount     0 non-null      float64
 3   final_fcst_each_qty  4313 non-null   float64
 4   total_sales          4313 non-null   float64
 5   pca_x                4313 non-null   float64
 6   pca_y                4313 non-null   float64
 7   cluster              4313 non-null   int64
dtypes: float64(6), int64(2)
memory usage: 269.7 KB
None
```

```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=store_features, x='pca_x', y='pca_y', hue='cluster', palette='Set2',
plt.title("Store Clusters (KMeans + PCA)")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.grid(True)
plt.tight_layout()
plt.savefig("cluster_plot.png")
plt.show()
```


Store Clusters (PCA Projection)
cluster
0
1
2
3
4
pca_y
pca_x
6
4
2
0
−2
−4
−2
0
2
4
6

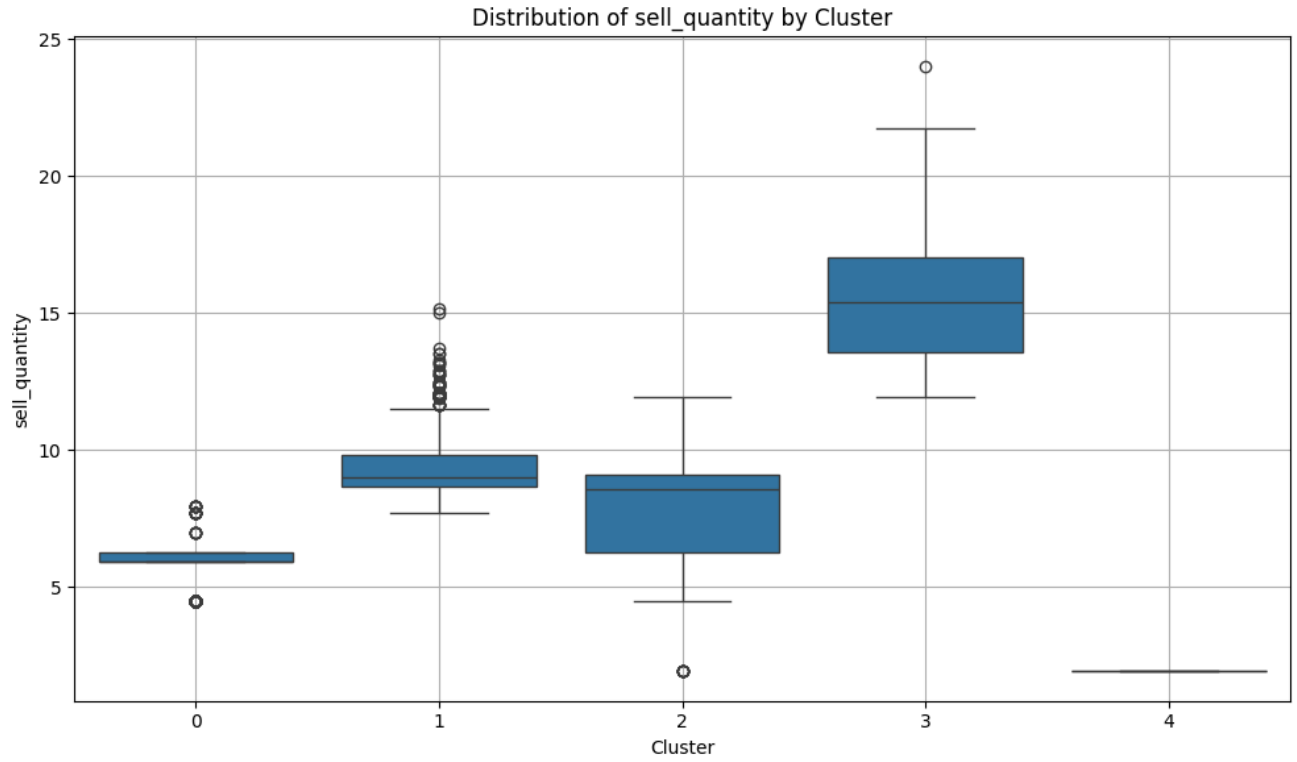

Store Clusters (KMeans + PCA)
cluster
0
1
2
3
4
PCA Component 2
PCA Component 1
6
4
2
0
−2
−4
−2
0
2
4
6

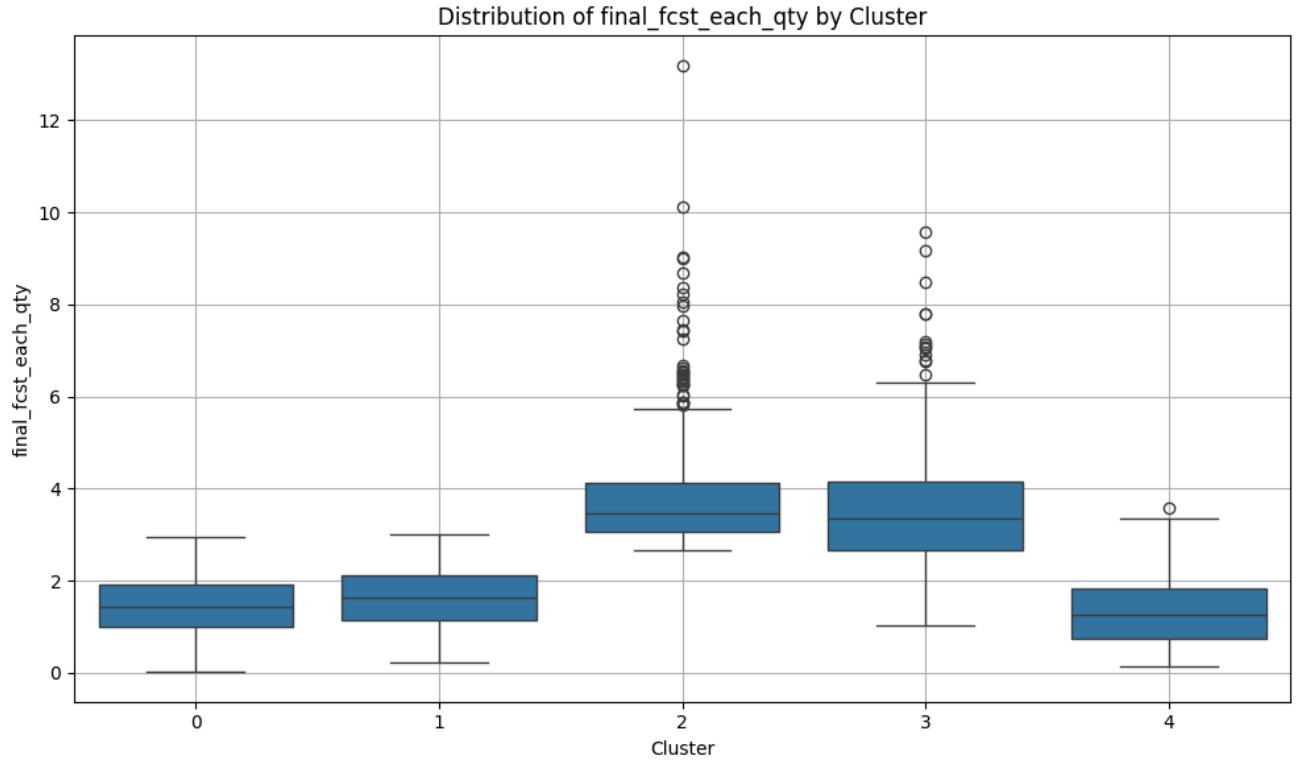
Visualize Cluster Characteristics

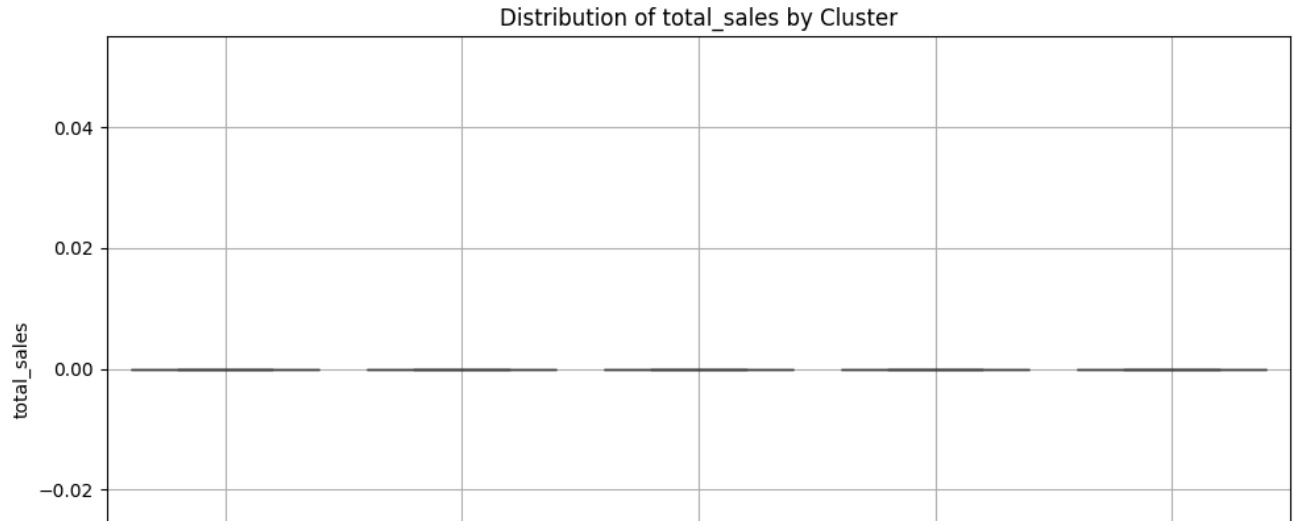
Subtask:

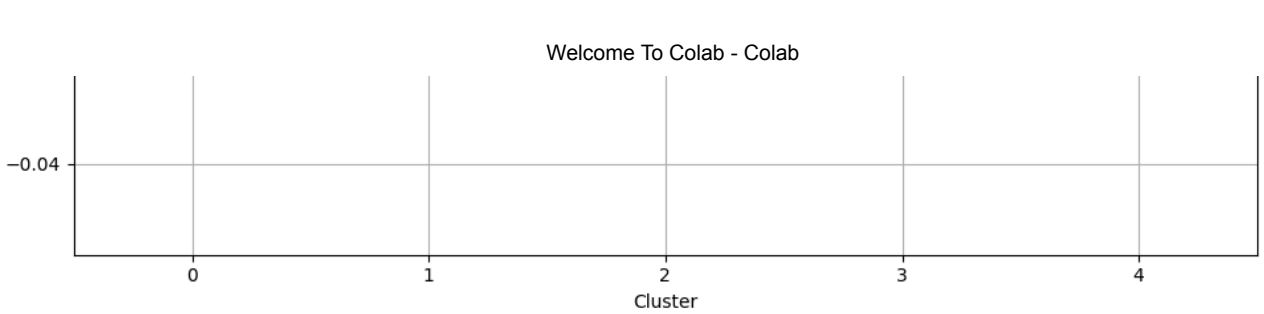
Generate box plots for key numeric features grouped by cluster to understand the distribution of values within each cluster.

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

features_to_plot = ['sell_quantity', 'final_fcst_each_qty', 'total_sales']

for feature in features_to_plot:
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='cluster', y=feature, data=df_clusters)
    plt.title(f'Distribution of {feature} by Cluster')
    plt.xlabel('Cluster')
    plt.ylabel(feature)
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```


Distribution of sell_quantity by Cluster
sell_quantity
Cluster
0
1
2
3
4
5
10
15
20
25


Distribution of final_fcst_each_qty by Cluster
final_fcst_each_qty
Cluster
0
2
4
6
8
10
12


Distribution of total_sales by Cluster
total_sales
0.04
0.02
0.00
−0.02

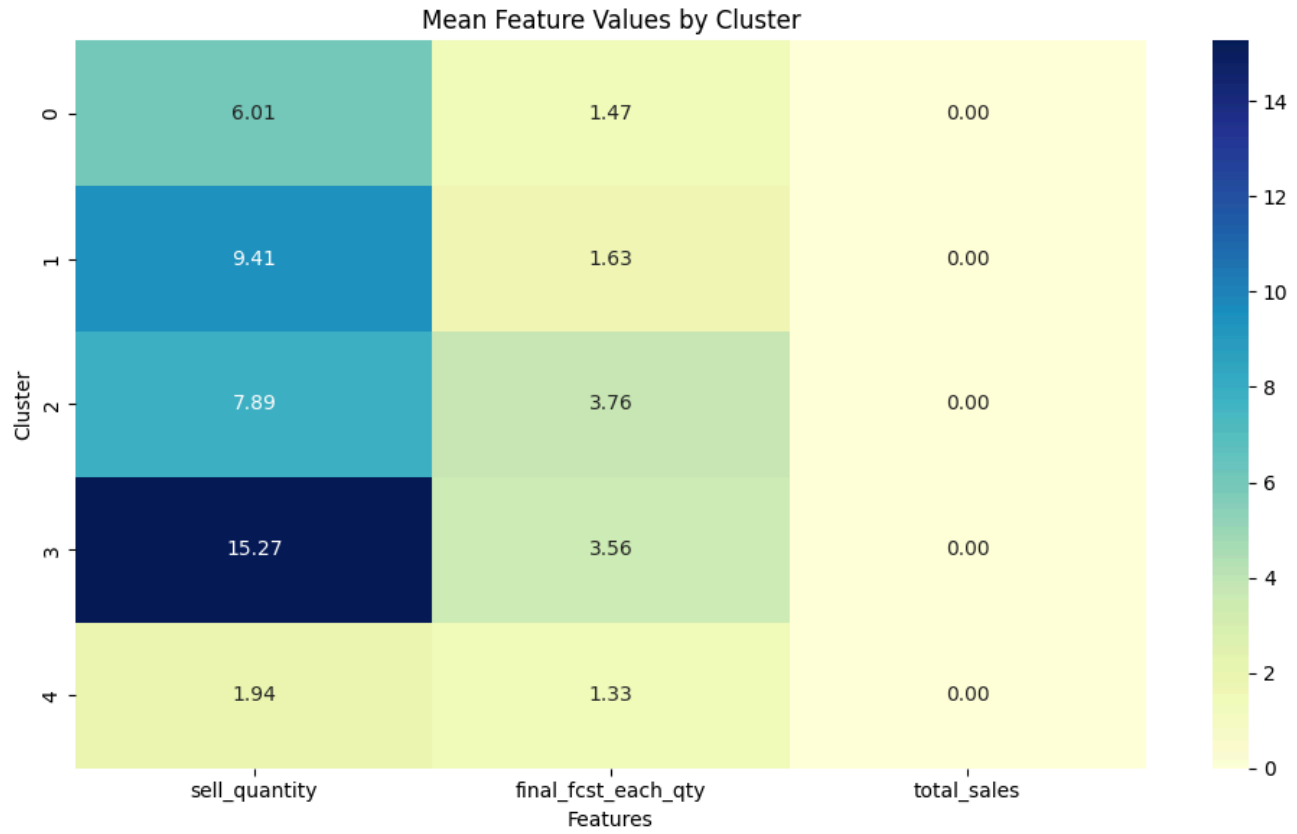

−0.04
0
1
2
3
4
Cluster

Subtask:

Generate a heatmap showing the mean values of key numeric features for each cluster to highlight average cluster characteristics.

Reasoning: Calculate the mean of the selected numeric features for each cluster and visualize these means using a heatmap. This will provide a clear overview of how the clusters differ in terms of their average performance metrics.

```
# Calculate the mean of the features for each cluster
cluster_means = df_clusters.groupby('cluster')[features_to_plot].mean()

plt.figure(figsize=(10, 6))
sns.heatmap(cluster_means, annot=True, cmap='YlGnBu', fmt=".2f")
plt.title('Mean Feature Values by Cluster')
plt.xlabel('Features')
plt.ylabel('Cluster')
plt.tight_layout()
plt.show()
```




```
!pip install streamlit
```

```
Requirement already satisfied: streamlit in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: blinker<2,>=1.5.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: cachetools<7,>=4.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: numpy<3,>=1.23 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: packaging<26,>=20 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: pandas<3,>=1.4.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: pillow<12,>=7.1.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: protobuf<7,>=3.20 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: tenacity<10,>=8.1.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: typing-extensions<5,>=4.4.0 in /usr/local/lib/python3.
Requirement already satisfied: watchdog<7,>=2.1.5 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /usr/local/lib/python3
Requirement already satisfied: pydeck<1,>=0.8.0b4 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: tornado!=6.5.0,<7,>=6.0.3 in /usr/local/lib/python3.11
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: narwhals>=1.14.2 in /usr/local/lib/python3.11/dist-pac
```

```
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/di
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (f
```

```
Start coding or generate with AI.
```

Reasoning: Save the clustered data to a CSV file and the fitted KMeans model to a pickle file.

Create a graphical user interface (ui)

Subtask:

Develop a simple web application using Streamlit to display the clustered data and the cluster visualization.