TITLE : To implement and analyze frequent itemset mining and clustering techniques on a real-world

dataset using Python.

NAME : Shinde Shubham Dnyandev

ROLL NO. : 23107121

CLASS : TY-B

BATCH : B

In [6]:
```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import matplotlib.pyplot as plt
```

In [7]:
```
dataset = [
    ['milk', 'bread', 'eggs'],
    ['milk', 'bread'],
    ['milk', 'eggs'],
    ['bread', 'butter'],
    ['milk', 'bread', 'butter'],
    ['bread', 'eggs'],
    ['milk', 'butter']
]
```

In [8]:
```
from mlxtend.preprocessing import TransactionEncoder
te = TransactionEncoder()
te_data = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_data, columns=te.columns_)

print("\n--- Transactions (One-Hot Encoded) ---")
print(df)
```

```
--- Transactions (One-Hot Encoded) ---
   bread  butter   eggs   milk
0   True   False   True   True
1   True   False  False   True
2  False   False   True   True
3   True    True  False  False
4   True    True  False   True
5   True   False   True  False
6  False    True  False   True
```

In [10]:
```
frequent_itemsets = apriori(df, min_support=0.3, use_colnames=True)

print("\n--- Frequent Itemsets ---")
print(frequent_itemsets)

rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.0)
print("\n--- Association Rules ---")
print(rules[['antecedents','consequents','support','confidence','lift']])
```

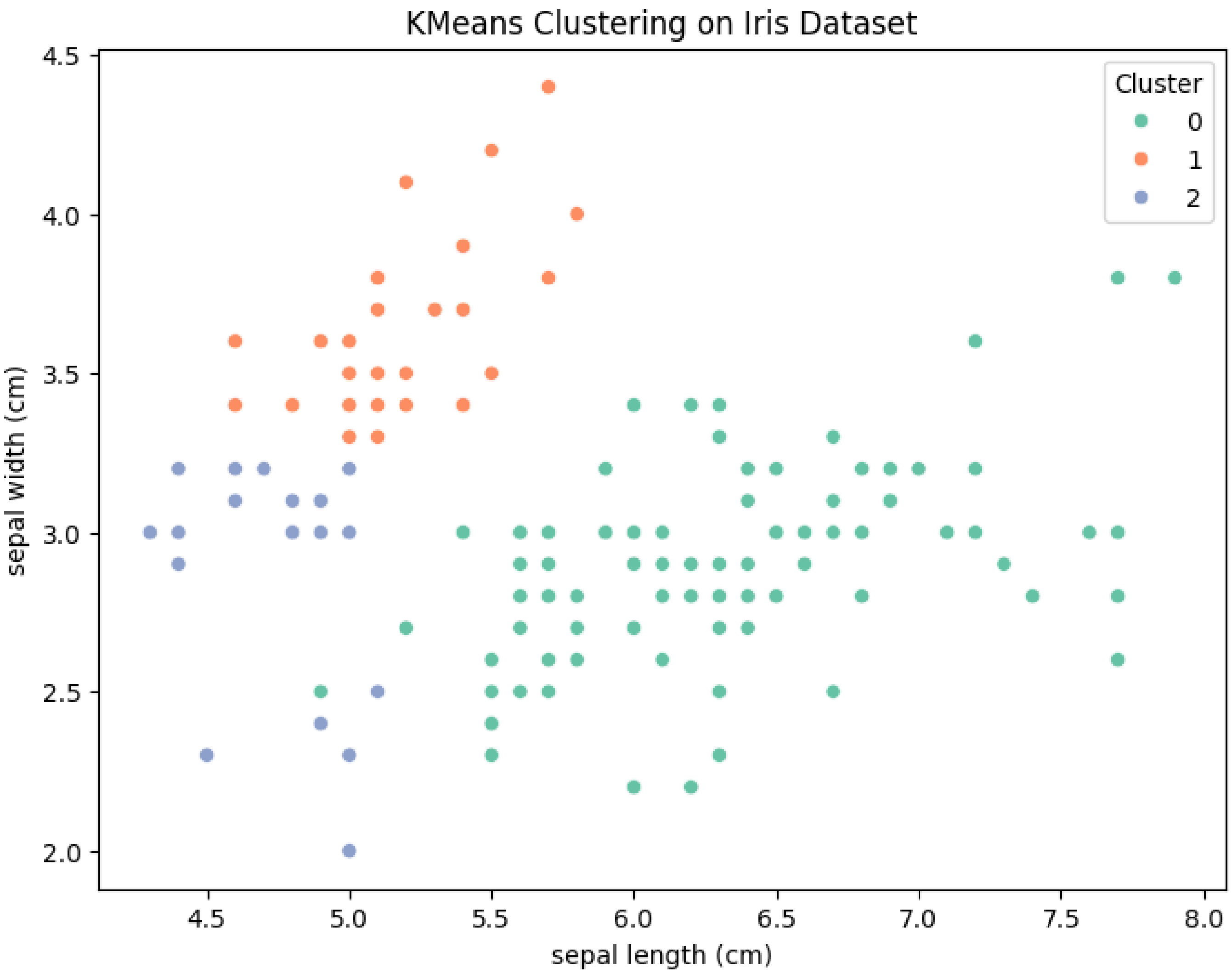
```
--- Frequent Itemsets ---
    support       itemsets
0  0.714286        (bread)
1  0.428571       (butter)
2  0.428571         (eggs)
3  0.714286         (milk)
4  0.428571  (bread, milk)

--- Association Rules ---
Empty DataFrame
Columns: [antecedents, consequents, support, confidence, lift]
Index: []
```

In [11]:
```
from sklearn.datasets import load_iris

# Load iris dataset
iris = load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(iris_df)
```

In [12]:
```
# Apply KMeans clustering
kmeans = KMeans(n_clusters=3, random_state=42)
iris_df['Cluster'] = kmeans.fit_predict(X_scaled)

print("\n--- Cluster Centers ---")
print(kmeans.cluster_centers_)
```

```
--- Cluster Centers ---
[[ 0.57100359 -0.37176778  0.69111943  0.66315198]
 [-0.81623084  1.31895771 -1.28683379 -1.2197118 ]
 [-1.32765367 -0.373138   -1.13723572 -1.11486192]]
```

In [13]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(x=iris_df['sepal length (cm)'], y=iris_df['sepal width (cm)'],hue=iri
plt.title("KMeans Clustering on Iris Dataset")
plt.show()
```


KMeans Clustering on Iris Dataset
Cluster
0
1
2
sepal width (cm)
sepal length (cm)
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

```
In [ ]:
```