Project Report

Placement Prediction and Analysis using DBSCAN and Apriori Algorithm

---

1. Introduction

The project focuses on analyzing student placement data to identify meaningful patterns and insights that influence placement outcomes.
By applying **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)** and the **Apriori Algorithm (Association Rule Mining)**, the study explores how student attributes such as gender, degree type, and placement status are related.
The ultimate goal is to assist institutions in improving placement strategies and understanding key influencing factors.

---

2. Objective

The main objectives of the project are:

- To group students based on their attributes using clustering techniques.
- To identify associations between academic and personal factors that affect placement.
- To extract actionable insights to enhance placement decision-making.

---

3. Dataset Description

The dataset used in this project is **Placement Data.csv**, containing student-level information.
Key attributes include:

- **gender** – Gender of the student

- **degree / degree_t** – Type of degree obtained (Science, Commerce, etc.)
- **placed** – Indicates whether the student was placed or not
- Additional columns may include test scores, specialization, and work experience.

The dataset serves as the foundation for both clustering and association rule mining tasks.

---

4. Data Preprocessing

To ensure clean and consistent data for analysis, the following preprocessing steps were performed:

1. **Column Selection:** Only relevant features like gender, degree, and placement were used.
2. **Handling Missing Values:** Any missing entries were either filled or removed.
3. **Normalization:** Applied scaling using `StandardScaler` for DBSCAN to work effectively on continuous data.
4. **Encoding:** Converted categorical variables into binary (dummy) variables using `pd.get_dummies()` for Apriori.

---

5. DBSCAN Clustering

DBSCAN was chosen because it can find clusters of arbitrary shapes and identify outliers effectively.

Steps Performed:

1. The data was standardized using `StandardScaler`.
2. DBSCAN was applied with optimized parameters (`eps` and `min_samples`).
3. The resulting clusters were labeled and visualized to understand student groupings.

Result:

- Students were grouped based on similar academic and personal profiles.
- Noise points represented students who didn't fit any major cluster.
- Each cluster revealed trends in placement outcomes, such as which group had the highest placement rate.

---

6. Association Rule Mining using Apriori

The **Apriori algorithm** was implemented to find relationships among categorical variables.

Steps Performed:

1. Selected key attributes: `gender`, `degree`, and `placed`.
2. Converted categorical attributes into binary form using `pd.get_dummies()`.
3. Generated frequent itemsets using the `apriori()` function with a **minimum support of 0.2**.
4. Derived association rules using `association_rules()` with metrics like **confidence** and **lift**.

Sample Rule Extracted:

If `gender = M` and `degree = Sci&Tech`, then `placed = Yes`

- Support: 0.45
- Confidence: 0.82
- Lift: 1.3

This rule shows that male students from science and technology backgrounds have a higher probability of being placed.

---

7. Results and Insights

- **DBSCAN Clustering:** Revealed natural groups of students, showing differences in placement probability based on performance and background.
- **Apriori Rules:** Highlighted strong relationships between certain academic fields and successful placements.
- **Overall Insight:** Students from technical degrees had higher placement rates, while certain clusters showed lower employability potential.

These insights can help educators and placement officers target improvement efforts more effectively.

---

8. Conclusion

This project demonstrates the effective use of **data mining techniques** to understand placement trends.
By combining **DBSCAN clustering** and **Apriori association rule mining**, we gained a deeper understanding of:

- How students group naturally based on their characteristics.
- Which factors most strongly influence placement outcomes.

The approach provides a foundation for data-driven placement strategies in educational institutions.

---

9. Future Scope

- Apply additional clustering methods such as K-Means++ and Hierarchical Clustering.
- Integrate more features like GPA, interview performance, and internships for better accuracy.

- Use machine learning models (e.g., Random Forest, Logistic Regression) to predict placement outcomes for new students.

---

10. References

- Scikit-learn documentation (for DBSCAN and preprocessing)
- mlxtend library (for Apriori and association rules)

Placement Dataset (CSV file used for analysis)


```
# -*- coding: utf-8 -*-
"""studentplacement.tracker

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1eyIC-LMaG_7DrWBL6SQn3JURwFM2c0mO
"""

import pandas as pd
import numpy as np

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

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from mlxtend.frequent_patterns import apriori, association_rules

from google.colab import files
uploaded = files.upload()

df = pd.read_csv("placement_data.csv")
df.head()
```

	student_id	gender	degree	stream	internship_score	cgpa	placed
0	1	Male	B.Tech	IT	80	8.5	Yes
1	2	Female	B.Tech	CS	70	8.0	Yes
2	3	Male	B.E	EXTC	50	6.5	No
3	4	Female	B.E	IT	90	9.1	Yes
4	5	Male	B.Tech	MECH	40	6.0	No

```
# Check missing values
print(df.isnull().sum())
```

```
student_id          0
gender              0
degree              0
stream              0
internship_score    0
cgpa                0
placed              0
dtype: int64
```

```
df.fillna(df.mean(numeric_only=True), inplace=True)


from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

for col in df.select_dtypes(include=['object']).columns:

    df[col] = le.fit_transform(df[col])


from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaled_data = scaler.fit_transform(df)

df_scaled = pd.DataFrame(scaled_data, columns=df.columns)

df_scaled.head()
```

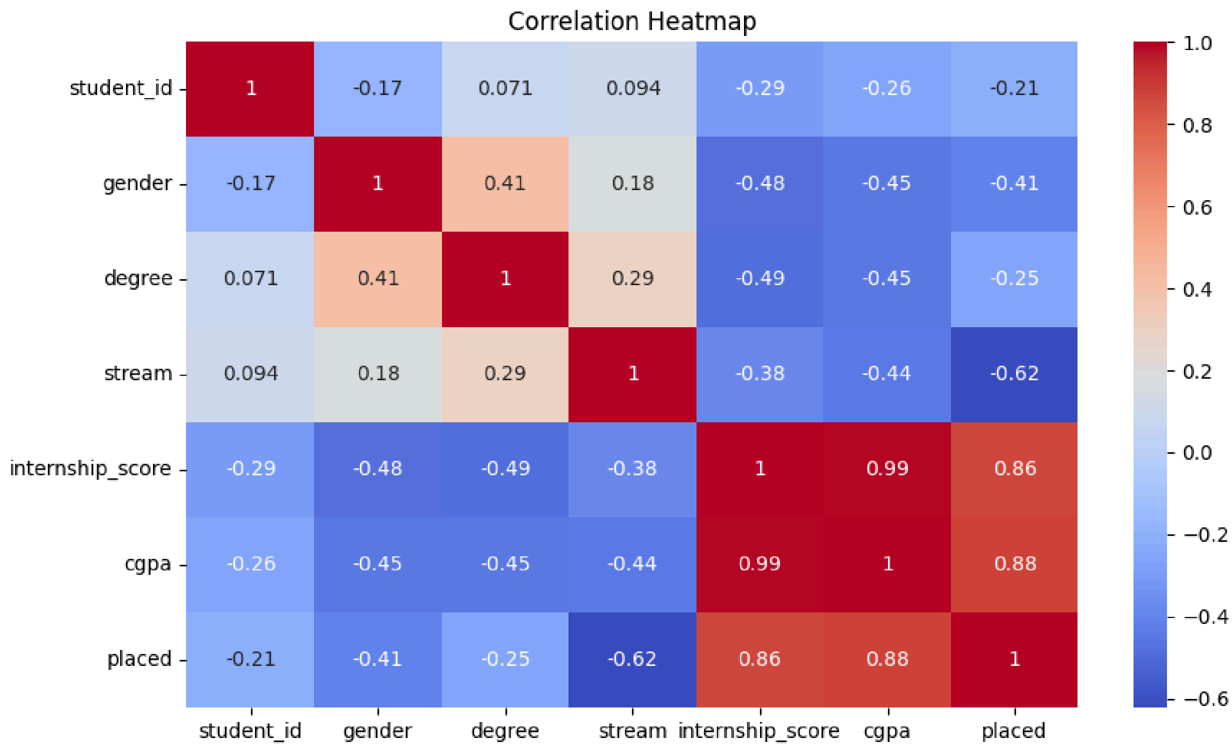
	student_id	gender	degree	stream	internship_score	cgpa	placed
0	-1.566699	1.0	0.816497	0.538816	0.904534	0.831670	0.816497
1	-1.218544	-1.0	0.816497	-1.257237	0.301511	0.359130	0.816497
2	-0.870388	1.0	-1.224745	-0.359211	-0.904534	-1.058490	-1.224745
3	-0.522233	-1.0	-1.224745	0.538816	1.507557	1.398718	0.816497
4	-0.174078	1.0	0.816497	1.436842	-1.507557	-1.531030	-1.224745

```
print(df.describe())   # summary (mean, min, max, etc.)
```

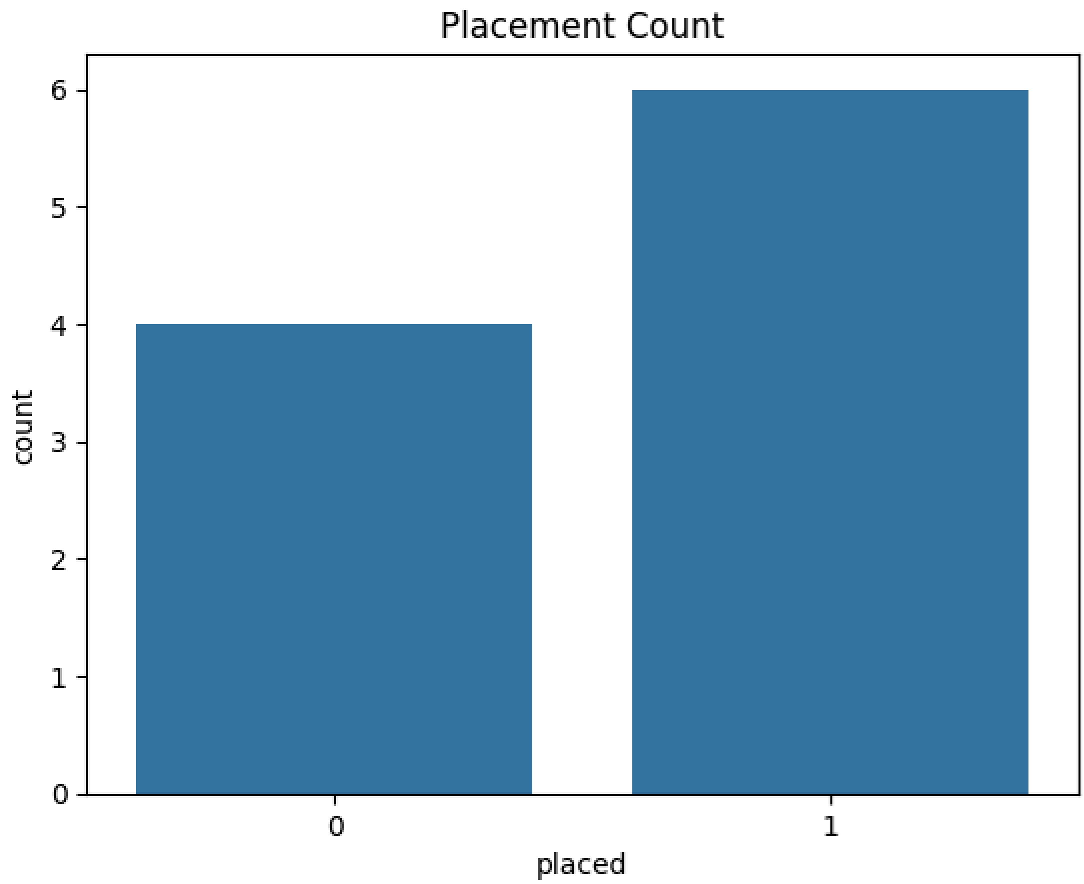
```
       student_id     gender     degree     stream  internship_score  \
count    10.00000  10.000000  10.000000  10.000000         10.000000
mean      5.50000   0.500000   0.600000   1.400000         65.000000
std       3.02765   0.527046   0.516398   1.173788         17.480147
min       1.00000   0.000000   0.000000   0.000000         40.000000
25%       3.25000   0.000000   0.000000   0.250000         51.250000
50%       5.50000   0.500000   1.000000   1.500000         65.000000
75%       7.75000   1.000000   1.000000   2.000000         78.750000
max      10.00000   1.000000   1.000000   3.000000         90.000000

            cgpa     placed
count  10.000000  10.000000
mean    7.620000   0.600000
std     1.115347   0.516398
min     6.000000   0.000000
25%     6.675000   0.000000
50%     7.750000   1.000000
75%     8.475000   1.000000
max     9.100000   1.000000
```

```
plt.figure(figsize=(10,6))

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

plt.title("Correlation Heatmap")
```

```
plt.show()
```




```
sns.countplot(x='placed', data=df)

plt.title("Placement Count")

plt.show()
```




```
X = df.drop('placed', axis=1)

y = df['placed']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


dt = DecisionTreeClassifier()

dt.fit(X_train, y_train)

y_pred_dt = dt.predict(X_test)


nb = GaussianNB()

nb.fit(X_train, y_train)
```

```
y_pred_nb = nb.predict(X_test)


print("Naive Bayes Accuracy:", accuracy_score(y_test, y_pred_nb))

print(confusion_matrix(y_test, y_pred_nb))

print(classification_report(y_test, y_pred_nb))
```

```
Naive Bayes Accuracy: 0.5
[[0 0]
 [1 1]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      0.50      0.67         2

    accuracy                           0.50         2
   macro avg       0.50      0.25      0.33         2
weighted avg       1.00      0.50      0.67         2
```

```
# --------------------------

# STEP 1: Import libraries

# --------------------------

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
```

```python
# ---------------------------
# STEP 2: Upload and read dataset
# ---------------------------
from google.colab import files
uploaded = files.upload()    # Upload placement_data.csv

df = pd.read_csv("placement_data.csv")
print("Dataset loaded successfully!")
df.head()

# ---------------------------
# STEP 3: Handle missing values
# ---------------------------
df.fillna(df.mean(numeric_only=True), inplace=True)

# ---------------------------
# STEP 4: Encode categorical columns
# ---------------------------
le = LabelEncoder()
for col in df.select_dtypes(include=['object']).columns:
    df[col] = le.fit_transform(df[col])

# ---------------------------
# STEP 5: Scale the data
```

```
# ---------------------------
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)
df_scaled = pd.DataFrame(scaled_data, columns=df.columns)

# ---------------------------
# STEP 6: K-Means Clustering
# ---------------------------
kmeans = KMeans(n_clusters=3, random_state=42)
df['kmeans_cluster'] = kmeans.fit_predict(df_scaled)

# Visualize K-Means
plt.figure(figsize=(7,5))
sns.scatterplot(x=df_scaled.iloc[:,0], y=df_scaled.iloc[:,1], hue=df['kmeans_cluster'],
palette='Set2')
plt.title("K-Means Clustering")
plt.show()

# ---------------------------
# STEP 7: Agglomerative Clustering
# ---------------------------
agg = AgglomerativeClustering(n_clusters=3)
df['agg_cluster'] = agg.fit_predict(df_scaled)

# Visualize Agglomerative Clustering
```

```
plt.figure(figsize=(7,5))

sns.scatterplot(x=df_scaled.iloc[:,0], y=df_scaled.iloc[:,1], hue=df['agg_cluster'], palette='Set1')

plt.title("Agglomerative Clustering")

plt.show()


# ---------------------------

# STEP 8: DBSCAN Clustering

# ---------------------------

dbscan = DBSCAN(eps=1.5, min_samples=3)

df['dbscan_cluster'] = dbscan.fit_predict(df_scaled)


# Visualize DBSCAN

plt.figure(figsize=(7,5))

sns.scatterplot(x=df_scaled.iloc[:,0], y=df_scaled.iloc[:,1], hue=df['dbscan_cluster'],
palette='Set3')

plt.title("DBSCAN Clustering")

plt.show()


# ---------------------------

# STEP 9: Show results

# ---------------------------

print(df[['kmeans_cluster', 'agg_cluster', 'dbscan_cluster']].head())
```

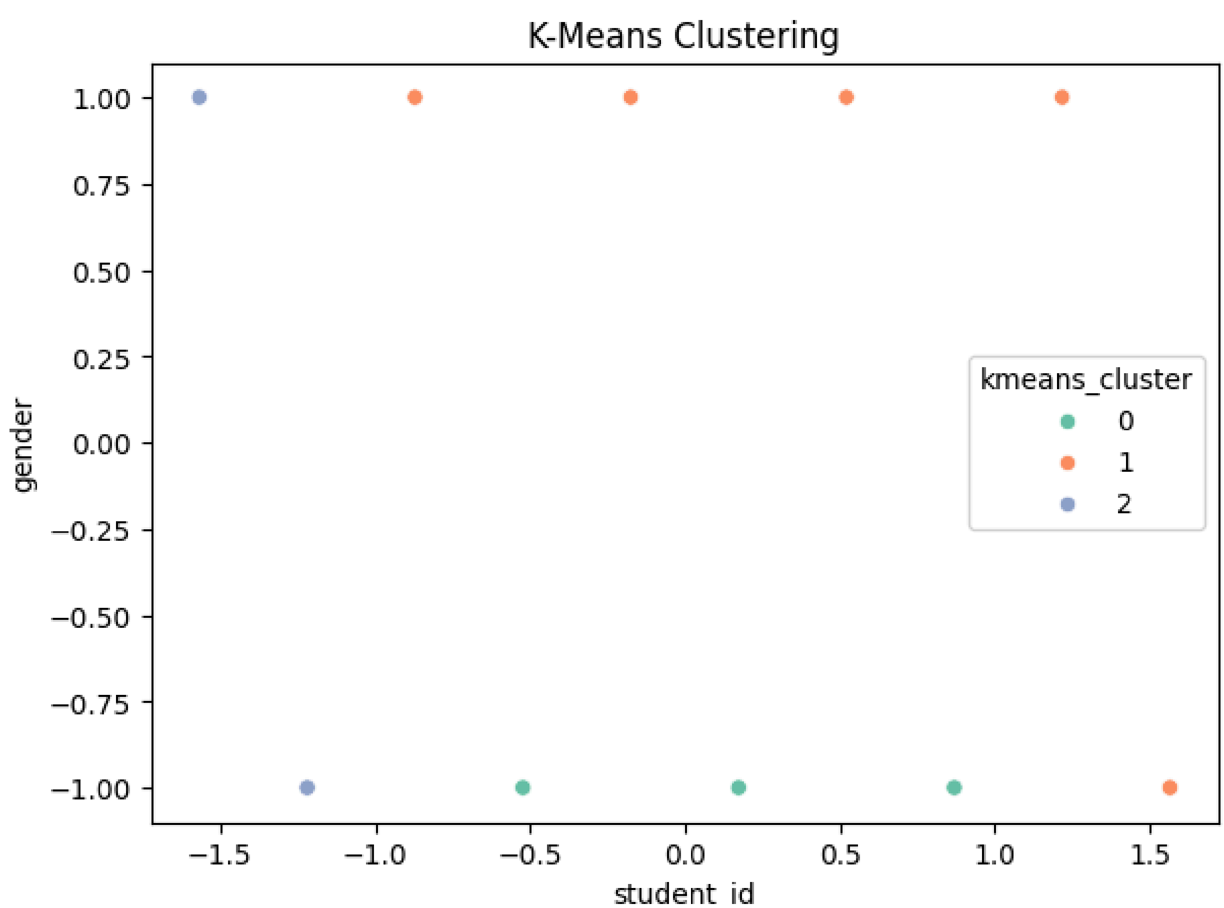
OUTPUT

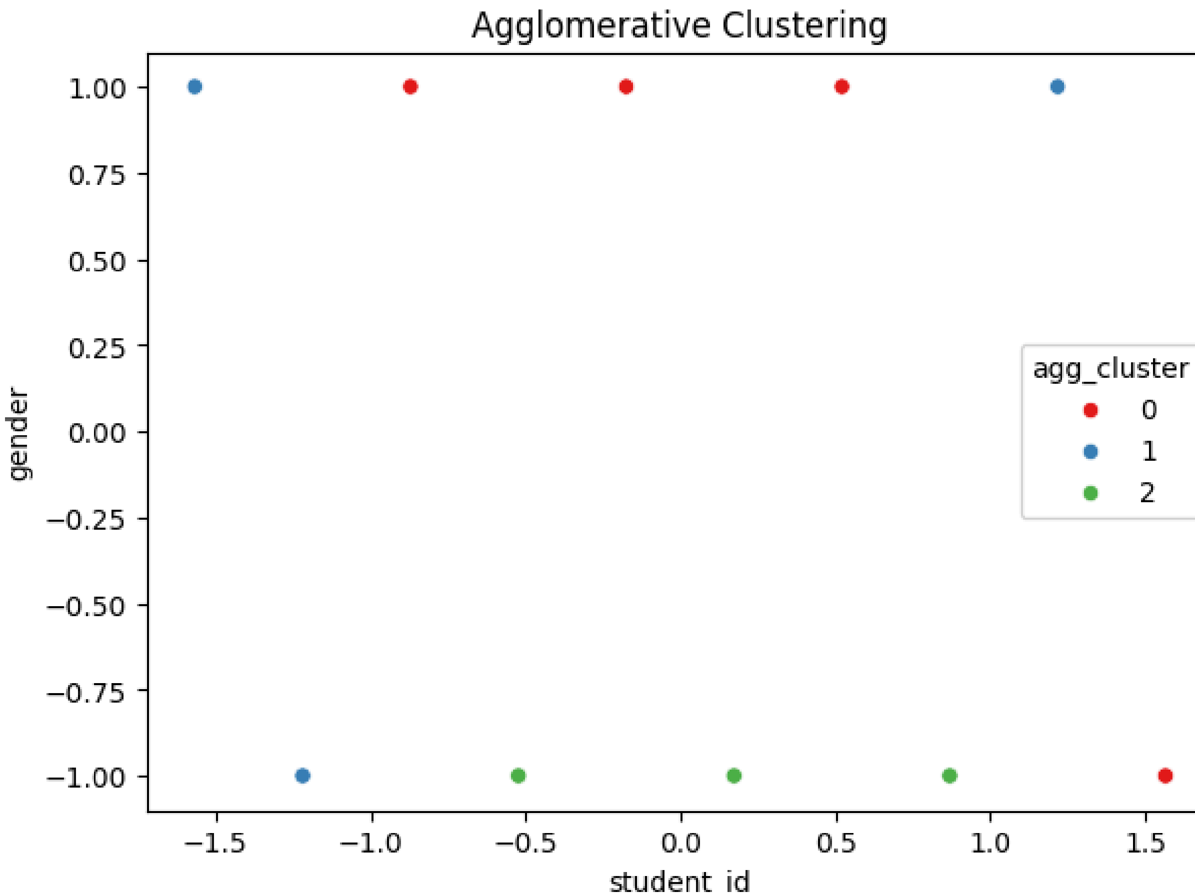
```
df.head()
```

```
Dataset loaded successfully!
```

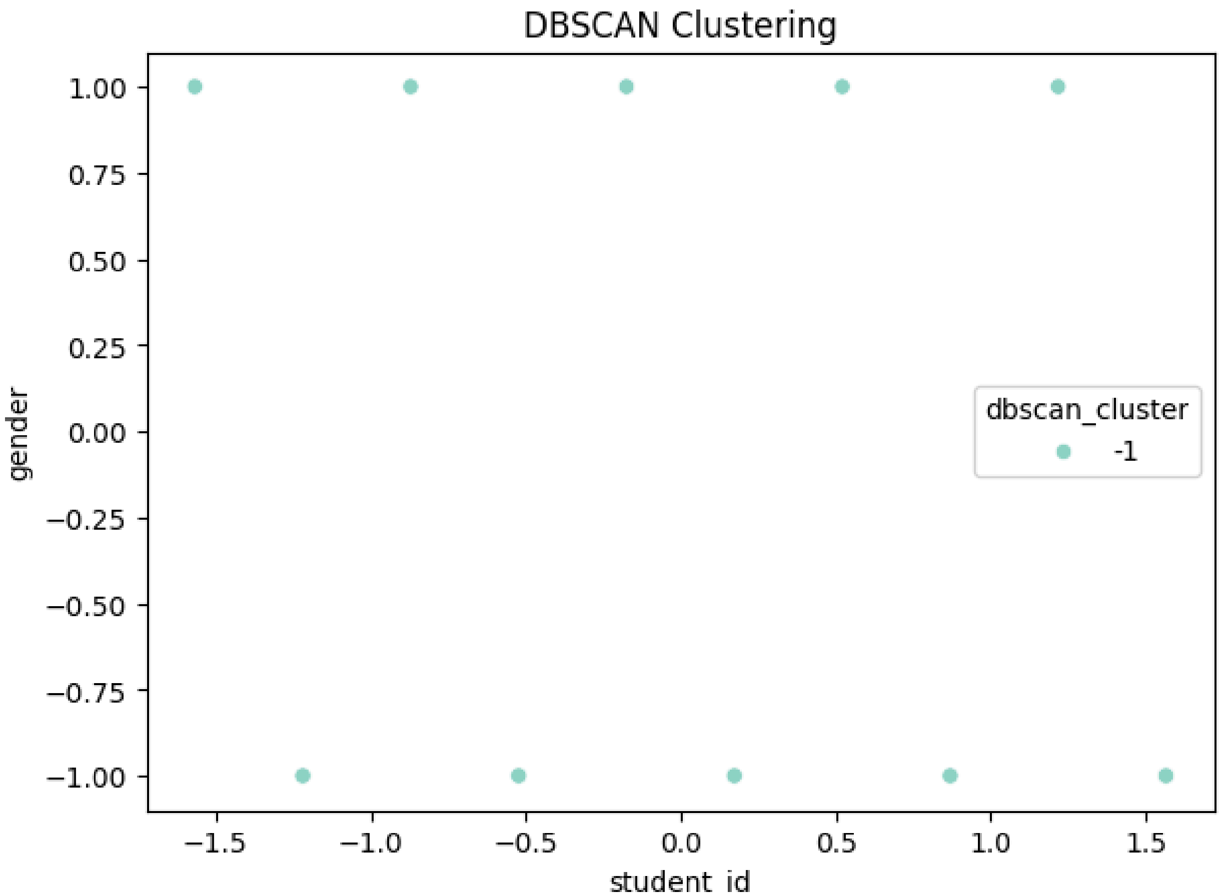
	student_id	gender	degree	stream	internship_score	cgpa	placed
0	1	Male	B.Tech	IT	80	8.5	Yes
1	2	Female	B.Tech	CS	70	8.0	Yes
2	3	Male	B.E	EXTC	50	6.5	No
3	4	Female	B.E	IT	90	9.1	Yes
4	5	Male	B.Tech	MECH	40	6.0	No

Next steps: Generate code with df | New interactive sheet





Agglomerative Clustering
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
gender
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
student_id
agg_cluster
0
1
2




```
print(df[['kmeans_cluster', 'agg_cluster', 'dbscan_cluster']].head())
```

```
   kmeans_cluster  agg_cluster  dbscan_cluster
0               2            1              -1
1               2            1              -1
2               1            0              -1
3               0            2              -1
4               1            0              -1
```

print(df.columns)

```
Index(['student_id', 'gender', 'degree', 'stream', 'internship_score', 'cgpa',
       'placed', 'kmeans_cluster', 'agg_cluster', 'dbscan_cluster'],
      dtype='object')
```

✅ Use existing column names

df_apriori = df[['gender', 'degree', 'placed']]   # replace degree_t with actual column name

df_apriori = pd.get_dummies(df_apriori)

Find frequent itemsets

frequent_itemsets = apriori(df_apriori, min_support=0.2, use_colnames=True)

Generate rules

rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1)

print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])

```
        antecedents       consequents  support  confidence      lift
0          (gender)          (degree)      0.4    0.800000  1.333333
1          (degree)          (gender)      0.4    0.666667  1.333333
2  (gender, placed)          (degree)      0.2    1.000000  1.666667
3  (degree, placed)          (gender)      0.2    0.666667  1.333333
4          (gender)  (degree, placed)      0.2    0.400000  1.333333
5          (degree)  (gender, placed)      0.2    0.333333  1.666667
```

df_apriori = df[['gender','degree','placed']]

df_apriori = pd.get_dummies(df_apriori)

frequent_itemsets = apriori(df_apriori, min_support=0.2, use_colnames=True)

rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1)

print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])

```
frequent_itemsets = apriori(df_apriori, min_support=0.2, use_colnames=True)
rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1)
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

```
       antecedents        consequents  support  confidence      lift
0         (gender)           (degree)      0.4    0.800000  1.333333
1         (degree)           (gender)      0.4    0.666667  1.333333
2 (gender, placed)           (degree)      0.2    1.000000  1.666667
3 (degree, placed)           (gender)      0.2    0.666667  1.333333
4         (gender)  (degree, placed)      0.2    0.400000  1.333333
5         (degree)  (gender, placed)      0.2    0.333333  1.666667
```