Student Course Feedback Analysis Project

(Python | Pandas | Visualization | Rating-based Sentiment)

In [ ]:
```
#  1. Project Overview

# This project analyzes student course feedback collected via Google Forms.
# The goal is to:

# Clean and prepare feedback data

# Analyze satisfaction using rating patterns

# Convert ratings into sentiment categories

# Visualize trends

# Provide actionable recommendations
```

In [1]:
```
# 2. Import Required Libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# 3. Load Dataset
# Upload CSV in Colab first, then run this
df = pd.read_csv("student_feedback.csv")

df.head()
```

Out[2]:

	Unnamed: 0	Student ID	Well versed with the subject	Explains concepts in an understandable way	Use of presentations	Degree of difficulty of assignments	w
0	0	340	5	2	7	6	
1	1	253	6	5	8	6	
2	2	680	7	7	6	5	
3	3	806	9	6	7	1	
4	4	632	8	10	8	4	

In [3]:
```
# 4. Dataset Information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1001 entries, 0 to 1000
Data columns (total 10 columns):
 #   Column                                        Non-Null Count  Dtyp
e
---  ------                                        --------------
-----
 0   Unnamed: 0                                    1001 non-null   int6
4
 1   Student ID                                    1001 non-null   int6
4
 2   Well versed with the subject                  1001 non-null   int6
4
 3   Explains concepts in an understandable way    1001 non-null   int6
4
 4   Use of presentations                          1001 non-null   int6
4
 5   Degree of difficulty of assignments           1001 non-null   int6
4
 6   Solves doubts willingly                       1001 non-null   int6
4
 7   Structuring of the course                     1001 non-null   int6
4
 8   Provides support for students going above and beyond  1001 non-null   int6
4
 9   Course recommendation based on relevance      1001 non-null   int6
4
dtypes: int64(10)
memory usage: 78.3 KB
```

In [4]:
```
# 5. Data Cleaning
# Drop unwanted index column
df.drop(columns=["Unnamed: 0"], inplace=True)
```

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

Out[4]:

	0
Student ID	0
Well versed with the subject	0
Explains concepts in an understandable way	0
Use of presentations	0
Degree of difficulty of assignments	0
Solves doubts willingly	0
Structuring of the course	0
Provides support for students going above and beyond	0
Course recommendation based on relevance	0

dtype: int64

In [6]:
```
# 6. Rename Columns
df.columns = [
    "Student_ID",
    "Subject_Knowledge",
    "Concept_Clarity",
    "Use_of_Presentations",
    "Assignment_Difficulty",
    "Doubt_Solving",
    "Course_Structure",
    "Extra_Student_Support",
    "Course_Relevance"
]

df.head()
```

Out[6]:

	Student_ID	Subject_Knowledge	Concept_Clarity	Use_of_Presentations	Assig
0	340	5	2	7	
1	253	6	5	8	
2	680	7	7	6	
3	806	9	6	7	
4	632	8	10	8	

In [7]:
```
# 7. Descriptive Statistics
df.describe()
```

Out[7]:

	Student_ID	Subject_Knowledge	Concept_Clarity	Use_of_Presentations
count	1001.000000	1001.000000	1001.000000	1001.000000
mean	500.000000	7.497502	6.081918	5.942058
std	289.108111	1.692998	2.597168	1.415853
min	0.000000	5.000000	2.000000	4.000000
25%	250.000000	6.000000	4.000000	5.000000
50%	500.000000	8.000000	6.000000	6.000000
75%	750.000000	9.000000	8.000000	7.000000
max	1000.000000	10.000000	10.000000	8.000000

In [8]:
```
# 8. Average Rating per Student
rating_cols = df.columns[1:]

df["Average_Rating"] = df[rating_cols].mean(axis=1)

df.head()
```

Out[8]:

	Student_ID	Subject_Knowledge	Concept_Clarity	Use_of_Presentations	Assig
0	340	5	2	7	
1	253	6	5	8	
2	680	7	7	6	
3	806	9	6	7	
4	632	8	10	8	

In [9]:
```
# 9. Convert Rating → Sentiment
def rating_to_sentiment(score):
    if score >= 7:
        return "Positive"
    elif score >= 4:
        return "Neutral"
    else:
        return "Negative"

df["Sentiment"] = df["Average_Rating"].apply(rating_to_sentiment)

df.head()
```

Out[9]:

	Student_ID	Subject_Knowledge	Concept_Clarity	Use_of_Presentations	Assig
0	340	5	2	7	
1	253	6	5	8	
2	680	7	7	6	
3	806	9	6	7	
4	632	8	10	8	

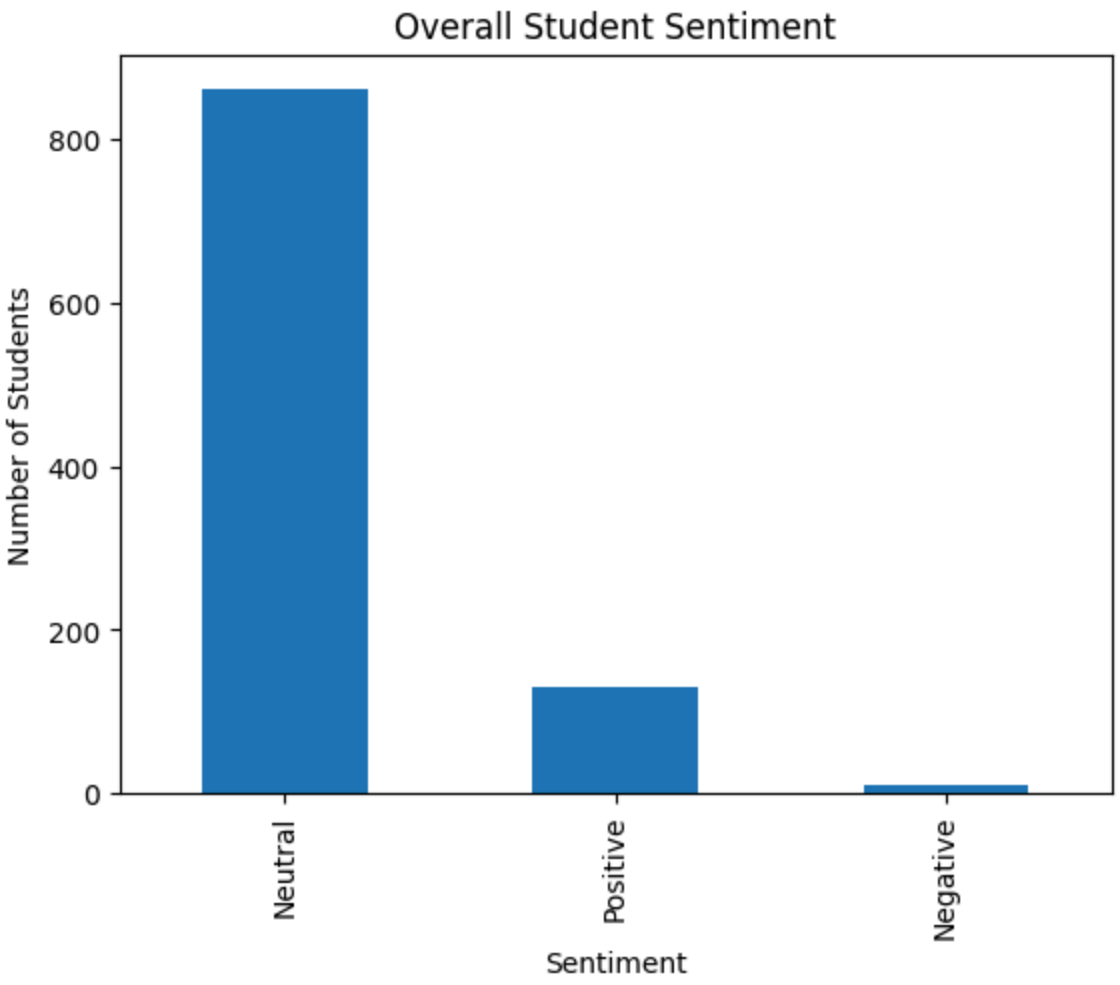
In [10]:
```
# 10. Sentiment Distribution
df["Sentiment"].value_counts()
```

Out[10]:

Sentiment	count
Neutral	860
Positive	131
Negative	10

dtype: int64

In [11]:
```
# 11. Visualization – Sentiment Analysis
df["Sentiment"].value_counts().plot(kind="bar")
plt.title("Overall Student Sentiment")
plt.xlabel("Sentiment")
plt.ylabel("Number of Students")
plt.show()
```



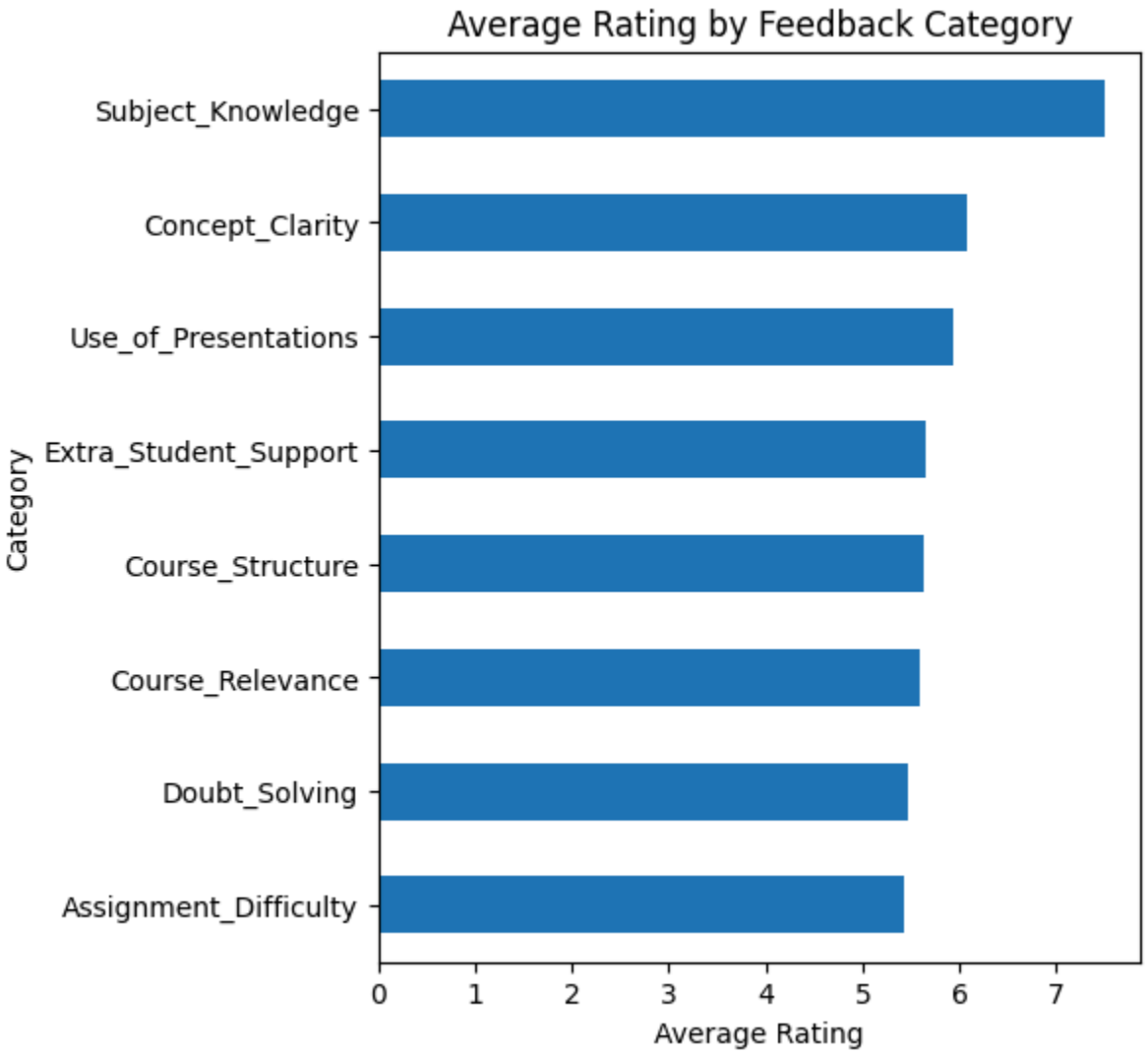
In [12]:
```
#  12. Average Rating per Question
avg_scores = df[rating_cols].mean().sort_values()

avg_scores
```

Out[12]:

	0
Assignment_Difficulty	5.430569
Doubt_Solving	5.474525
Course_Relevance	5.598402
Course_Structure	5.636364
Extra_Student_Support	5.662338
Use_of_Presentations	5.942058
Concept_Clarity	6.081918
Subject_Knowledge	7.497502

dtype: float64

In [15]:
```
# 13. Visualization – Question Wise Ratings
plt.figure(figsize=(5,6))
avg_scores.plot(kind="barh")
plt.title("Average Rating by Feedback Category")
plt.xlabel("Average Rating")
plt.ylabel("Category")
plt.show()
```



In [17]:
```
# 14. Key Insights
lowest_rated = avg_scores.head(3)
highest_rated = avg_scores.tail(3)

lowest_rated, highest_rated
```

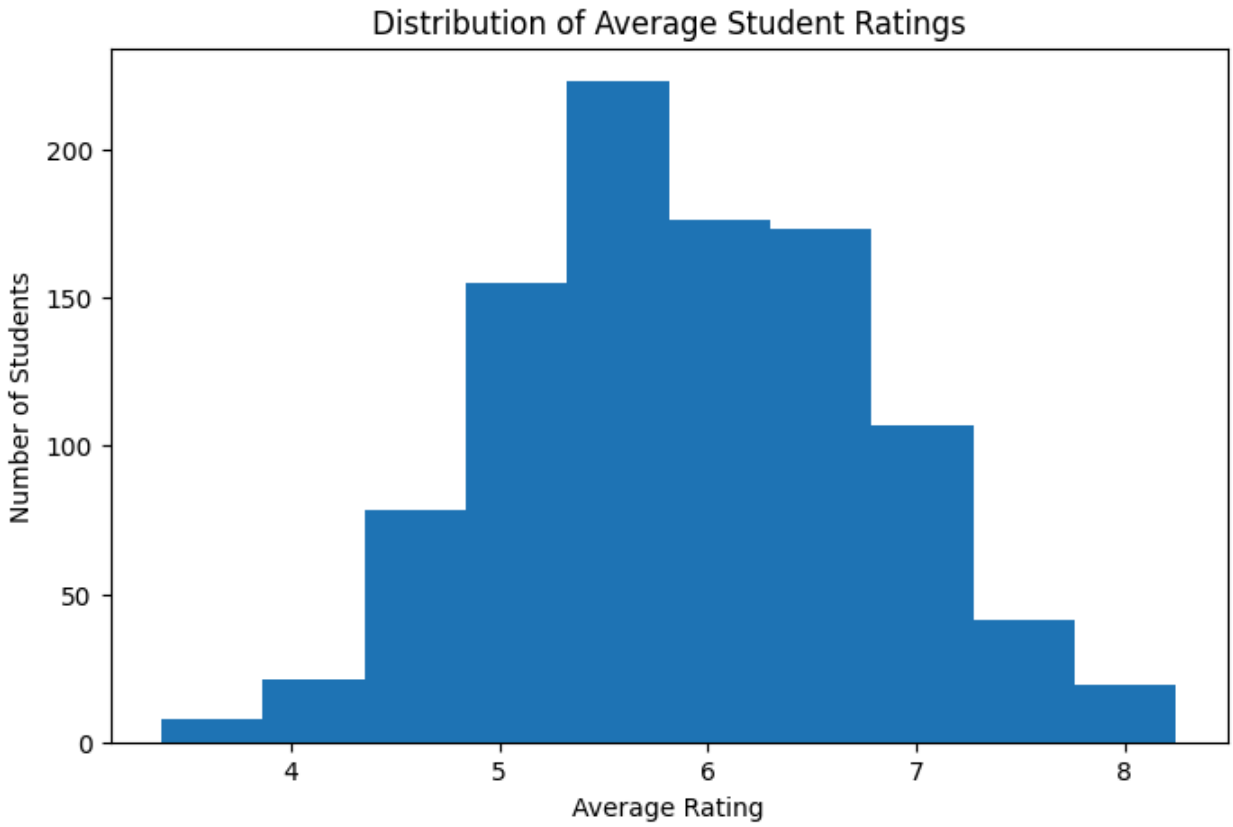
Out[17]:
```
(Assignment_Difficulty    5.430569
 Doubt_Solving            5.474525
 Course_Relevance         5.598402
 dtype: float64,
 Use_of_Presentations    5.942058
 Concept_Clarity         6.081918
 Subject_Knowledge       7.497502
 dtype: float64)
```

In [18]:
```
# 15. Business Recommendations
recommendations = {
    "Assignment_Difficulty": "Simplify or better explain assignments",
    "Use_of_Presentations": "Increase use of visual learning materials",
    "Course_Structure": "Improve course flow and planning"
}

recommendations
```

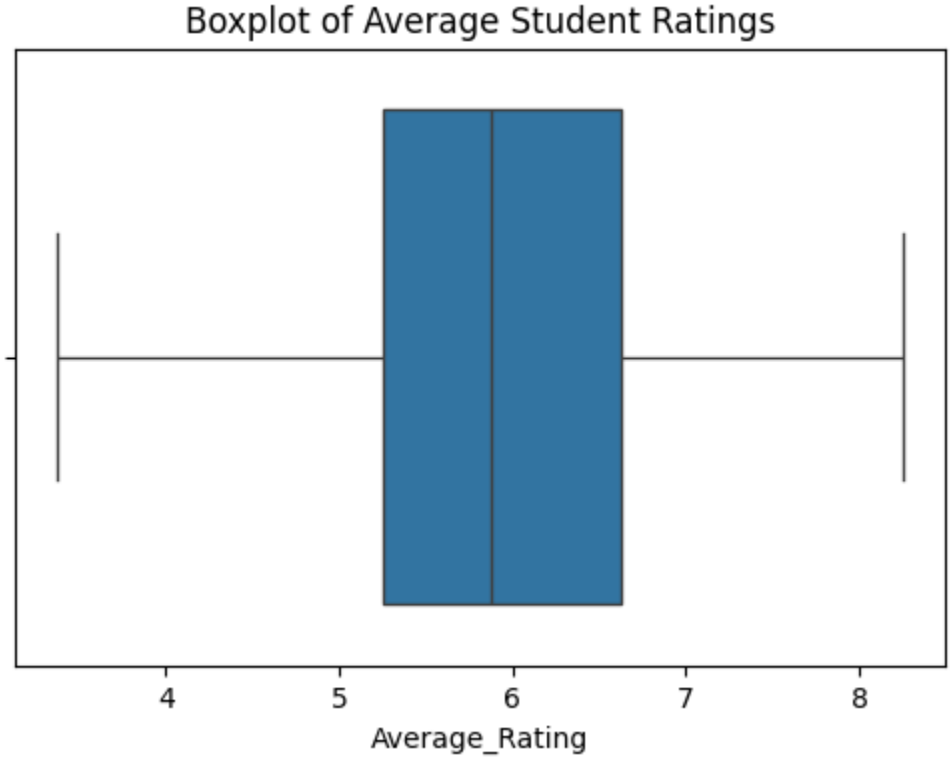
Out[18]:
```
{'Assignment_Difficulty': 'Simplify or better explain assignments',
 'Use_of_Presentations': 'Increase use of visual learning materials',
 'Course_Structure': 'Improve course flow and planning'}
```

In [20]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

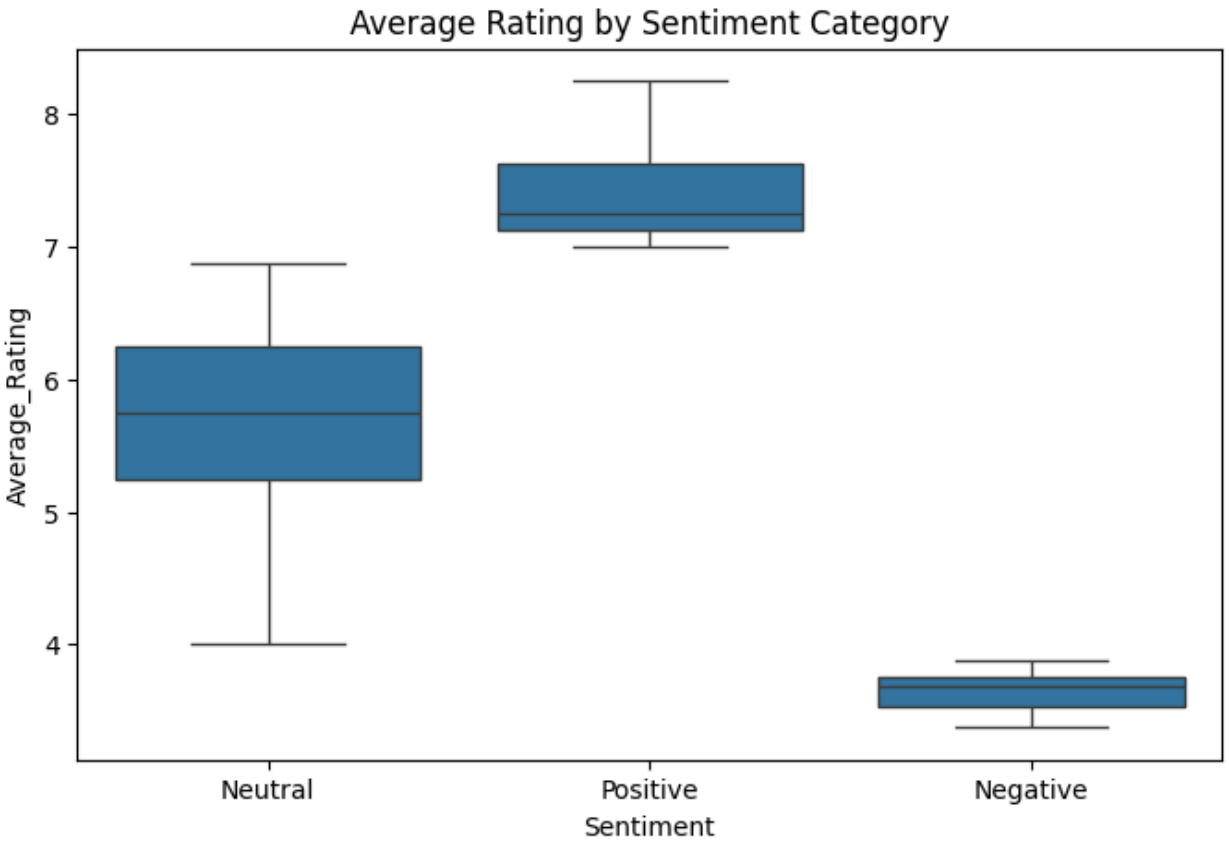
In [21]:
```
plt.figure(figsize=(8,5))
plt.hist(df["Average_Rating"], bins=10)
plt.title("Distribution of Average Student Ratings")
plt.xlabel("Average Rating")
plt.ylabel("Number of Students")
plt.show()
```



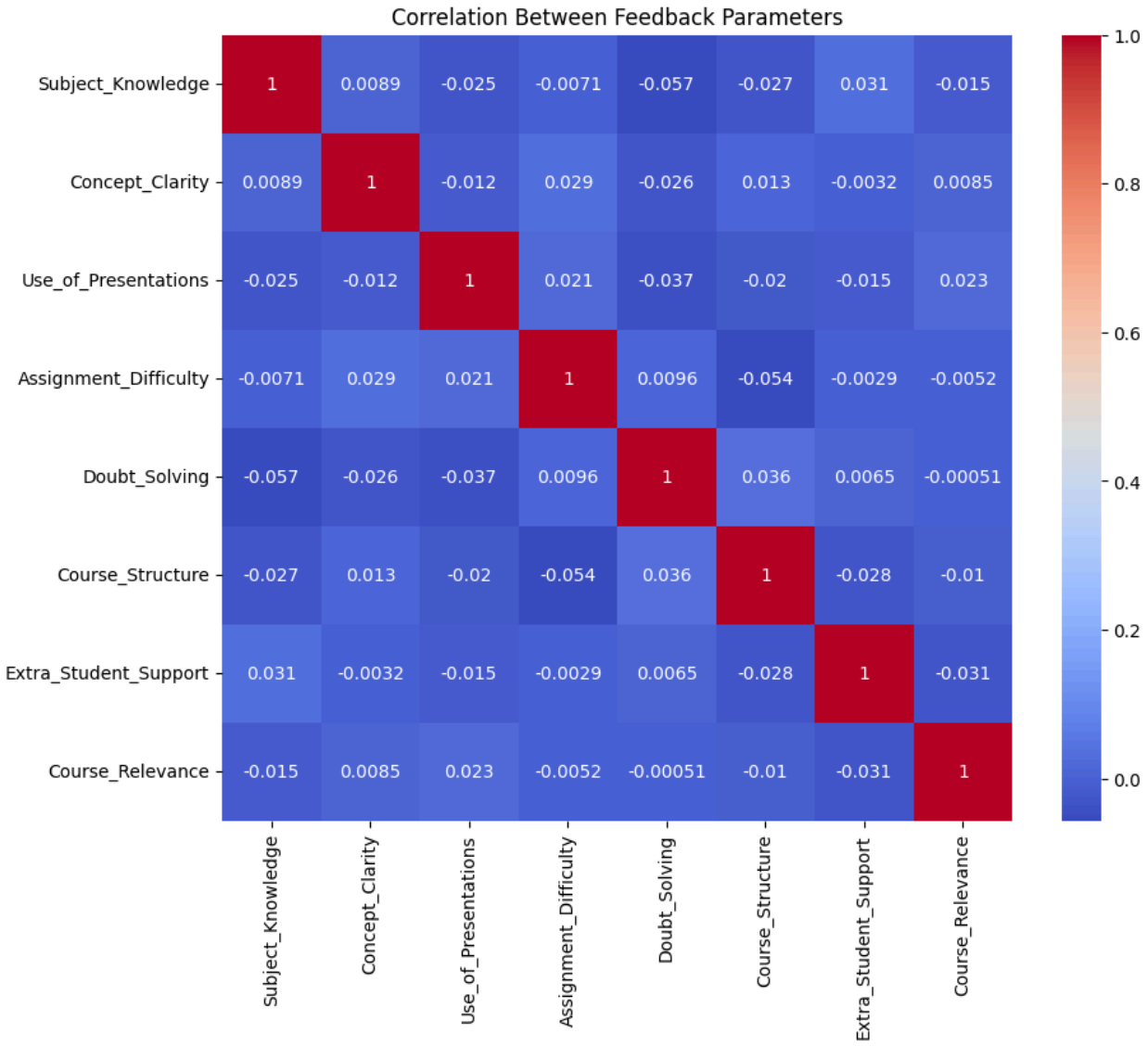
In [22]:
```
plt.figure(figsize=(6,4))
sns.boxplot(x=df["Average_Rating"])
plt.title("Boxplot of Average Student Ratings")
plt.show()
```



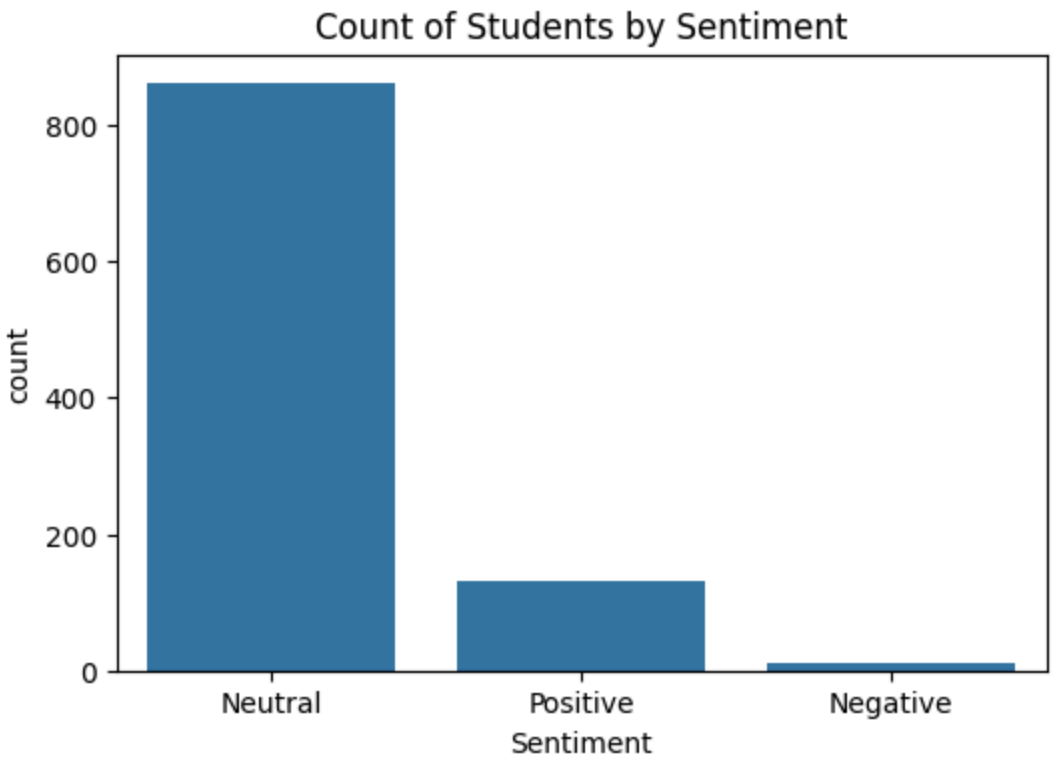
In [23]:
```
plt.figure(figsize=(8,5))
sns.boxplot(x="Sentiment", y="Average_Rating", data=df)
plt.title("Average Rating by Sentiment Category")
plt.show()
```



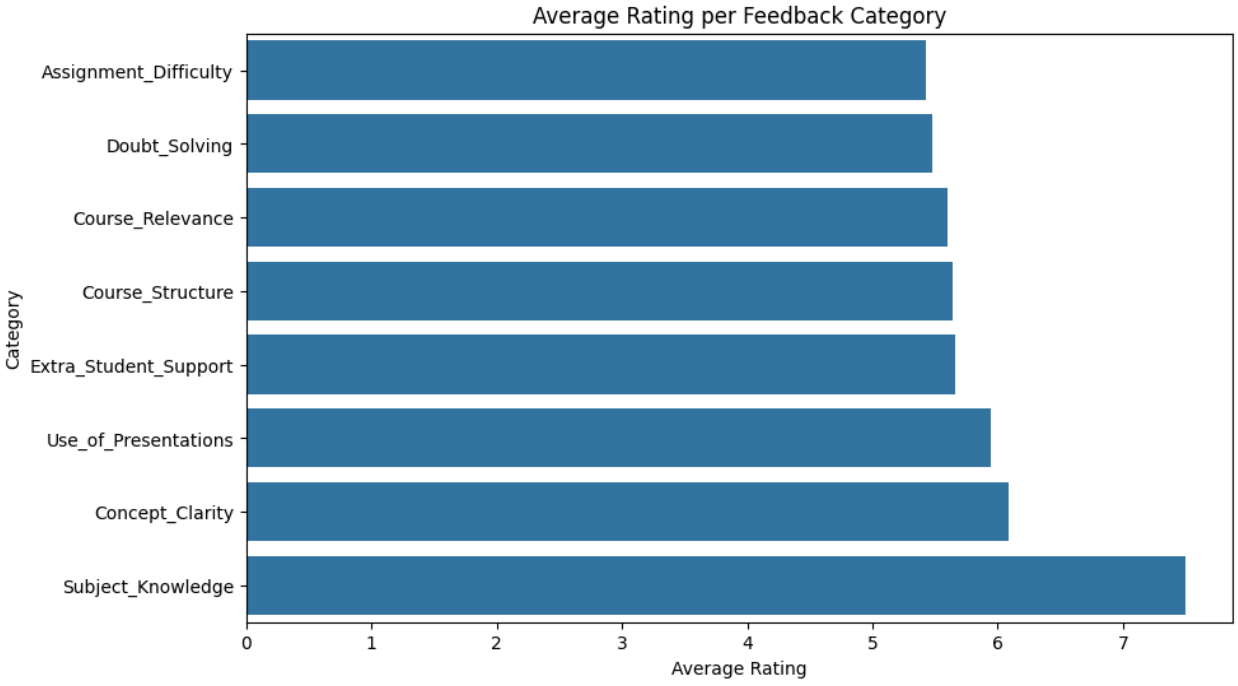
In [24]:
```
plt.figure(figsize=(10,8))
sns.heatmap(df[rating_cols].corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Between Feedback Parameters")
plt.show()
```



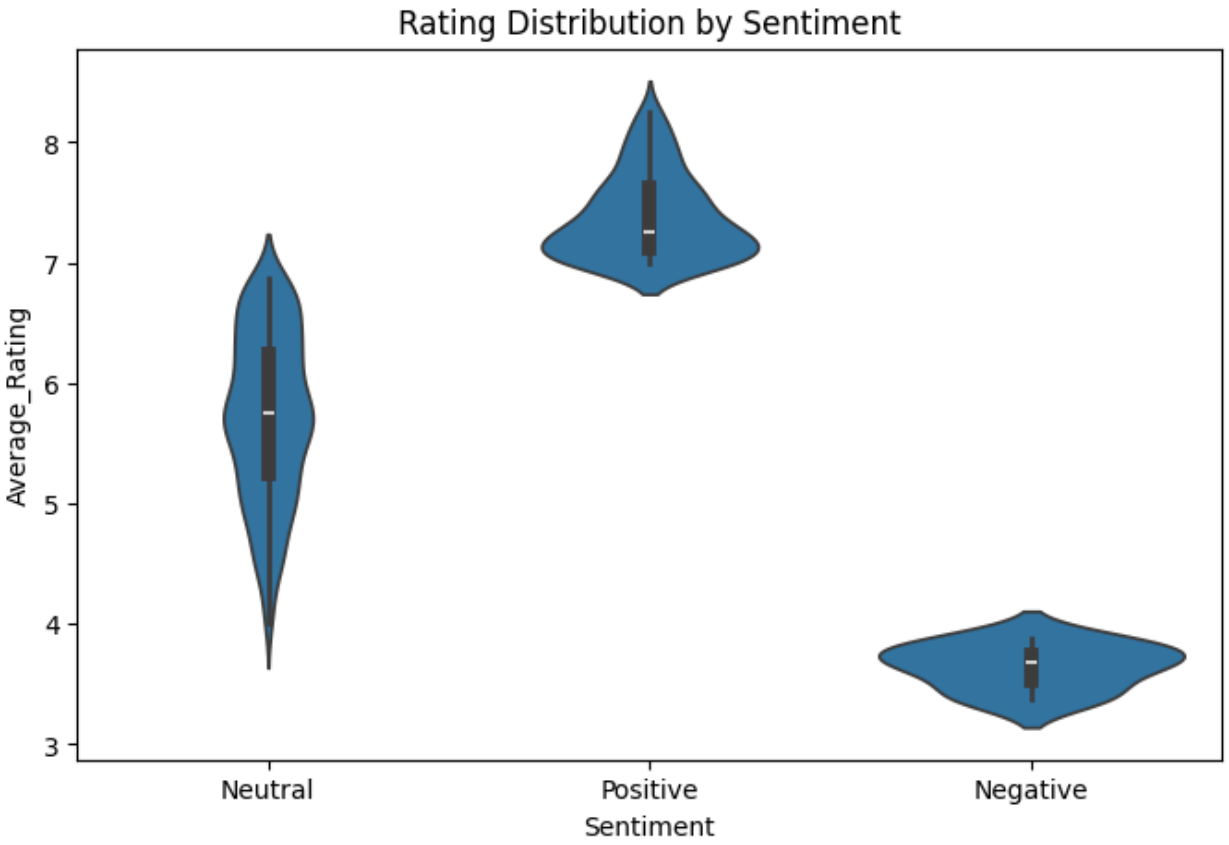
In [25]:
```
plt.figure(figsize=(6,4))
sns.countplot(x="Sentiment", data=df)
plt.title("Count of Students by Sentiment")
plt.show()
```



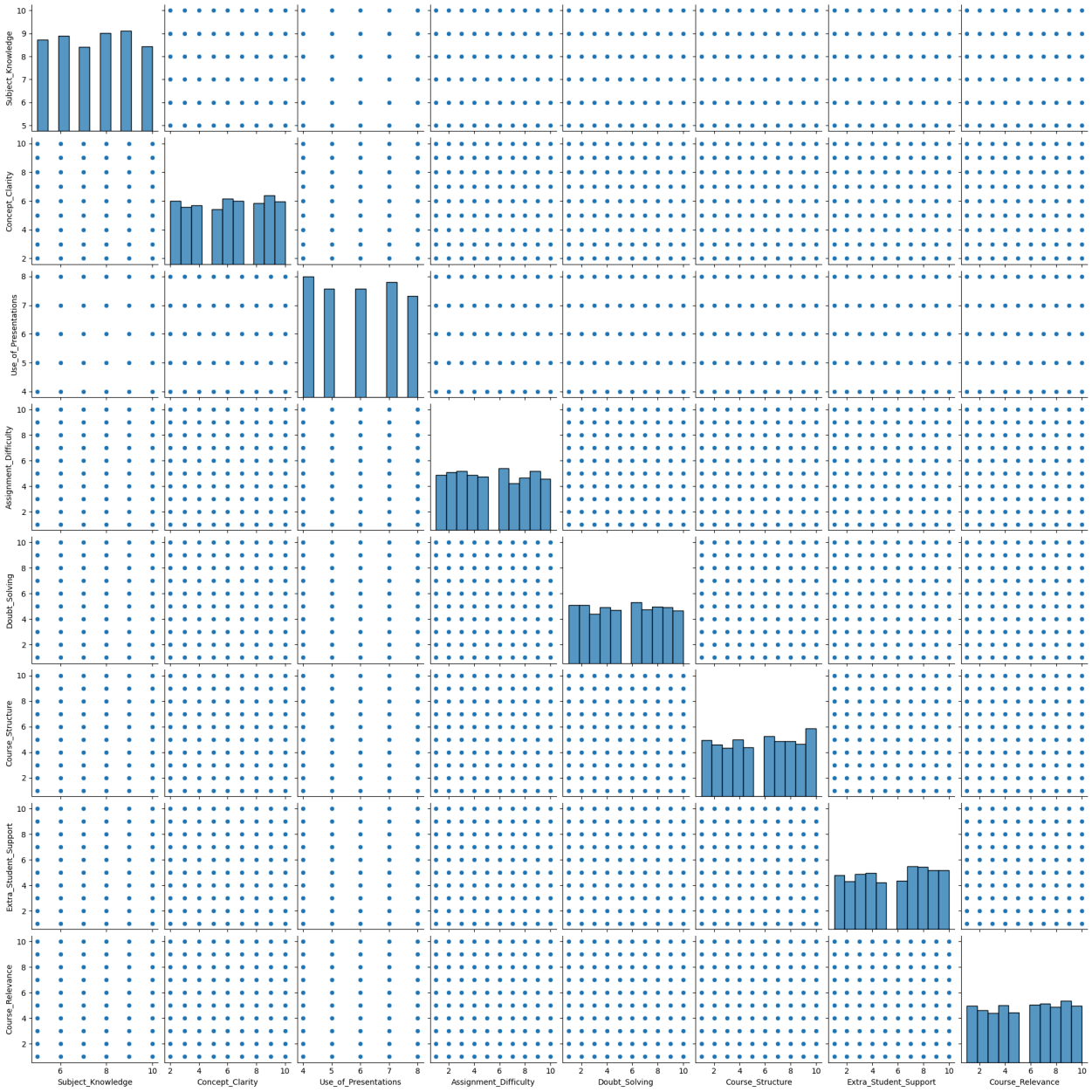
In [26]:
```
avg_df = avg_scores.reset_index()
avg_df.columns = ["Category", "Average Rating"]

plt.figure(figsize=(10,6))
sns.barplot(x="Average Rating", y="Category", data=avg_df)
plt.title("Average Rating per Feedback Category")
plt.show()
```



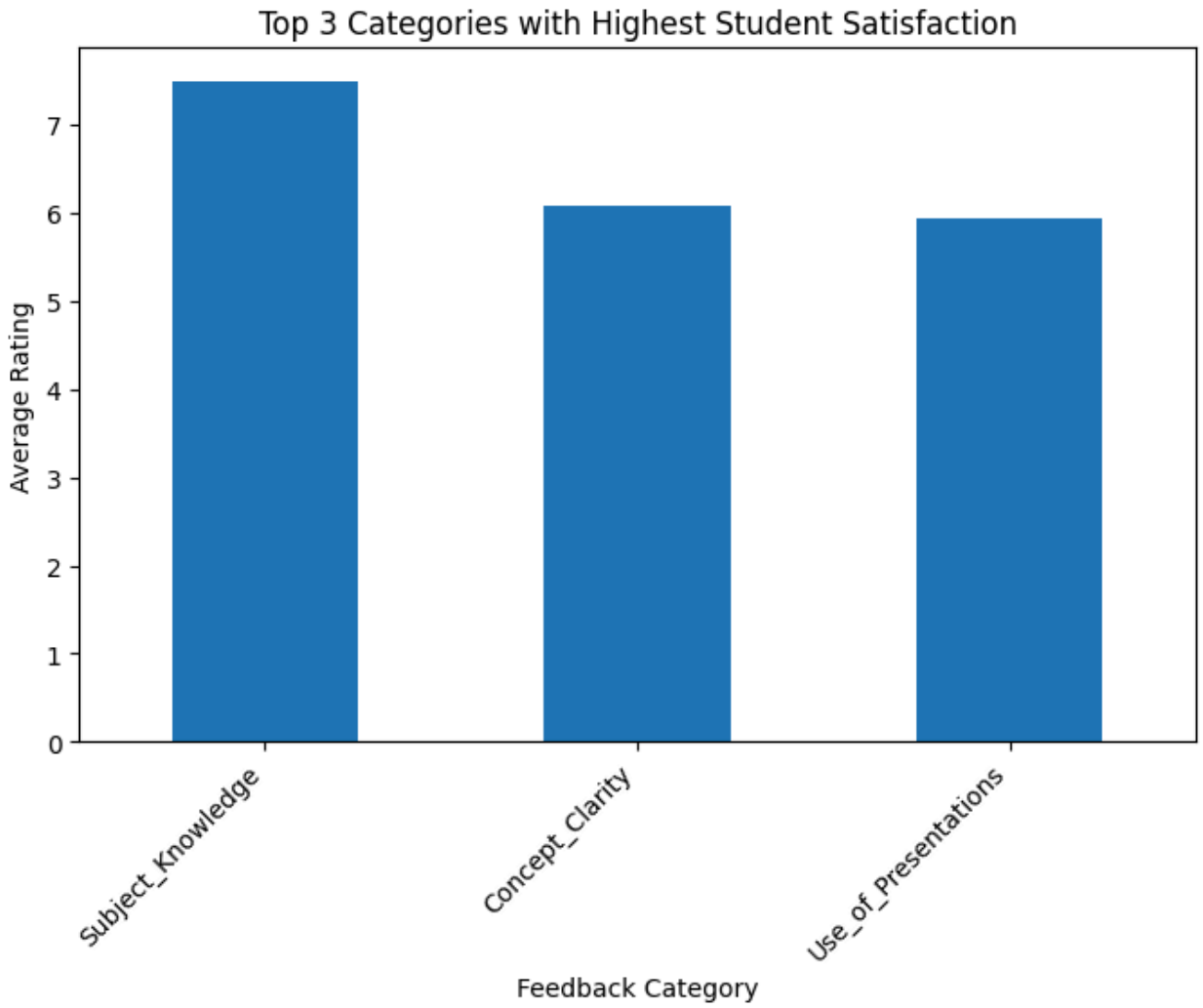
In [27]:
```
plt.figure(figsize=(8,5))
sns.violinplot(x="Sentiment", y="Average_Rating", data=df)
plt.title("Rating Distribution by Sentiment")
plt.show()
```



In [28]:
```
sns.pairplot(df[rating_cols])
plt.show()
```



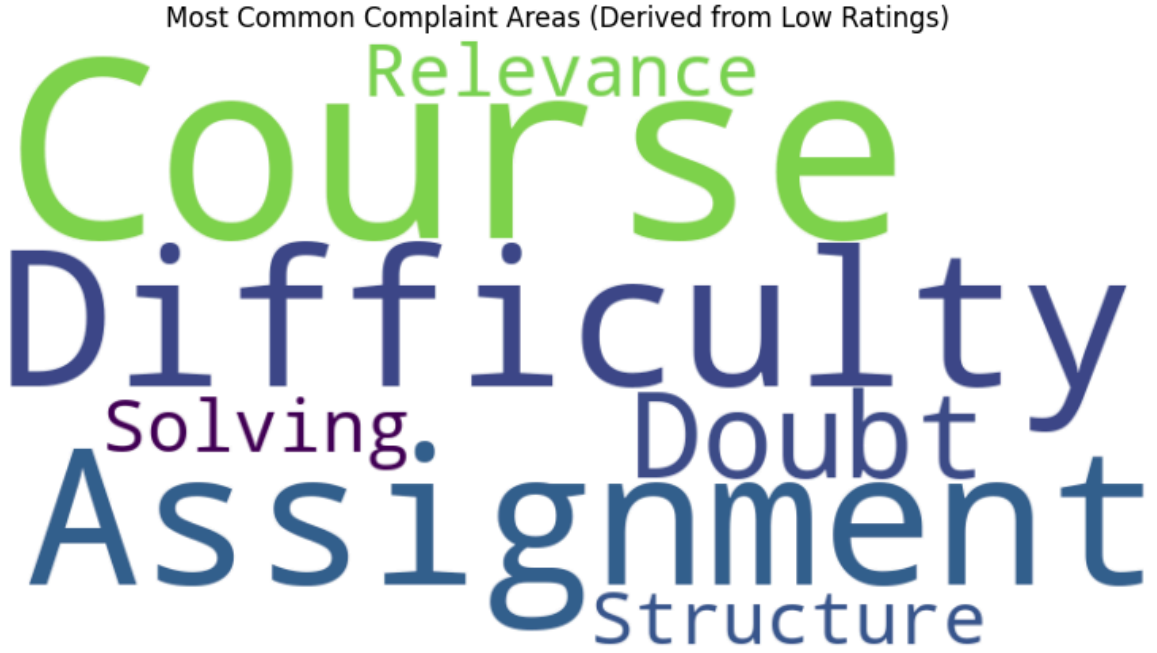
In [29]:
```
avg_scores = df[rating_cols].mean().sort_values(ascending=False)
avg_scores
top_3 = avg_scores.head(3)
top_3
plt.figure(figsize=(8,5))
top_3.plot(kind="bar")
plt.title("Top 3 Categories with Highest Student Satisfaction")
plt.xlabel("Feedback Category")
plt.ylabel("Average Rating")
plt.xticks(rotation=45, ha="right")
plt.show()
```



In [30]:

```
low_scores = avg_scores.sort_values().head(4)
low_scores
complaint_text = " ".join(low_scores.index.str.replace("_", " "))
complaint_text
!pip install wordcloud
from wordcloud import WordCloud
import matplotlib.pyplot as plt
wordcloud = WordCloud(
    width=800,
    height=400,
    background_color="white"
).generate(complaint_text)

plt.figure(figsize=(10,5))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title("Most Common Complaint Areas (Derived from Low Ratings)")
plt.show()
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.12/dist-pack
ages (1.9.5)
Requirement already satisfied: numpy>=1.19 in /usr/local/lib/python3.12/dist-pa
ckages (from wordcloud) (2.0.2)
Requirement already satisfied: pillow in /usr/local/lib/python3.12/dist-package
s (from wordcloud) (11.3.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-pac
kages (from wordcloud) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/di
st-packages (from matplotlib->wordcloud) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-p
ackages (from matplotlib->wordcloud) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/d
ist-packages (from matplotlib->wordcloud) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/d
ist-packages (from matplotlib->wordcloud) (1.4.9)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dis
t-packages (from matplotlib->wordcloud) (25.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/di
st-packages (from matplotlib->wordcloud) (3.3.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.1
2/dist-packages (from matplotlib->wordcloud) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packa
ges (from python-dateutil>=2.7->matplotlib->wordcloud) (1.17.0)
```



In [ ]:

In [ ]: