1. Importing the Data into Google Colaboratory

```
import pandas as pd

df = pd.read_csv("student feedback.csv")
df.head()
```

Result:

```
import pandas as pd

df = pd.read_csv("student feedback.csv")
df.head()
```

	Unnamed: 0	Student ID	Well versed with the subject	Explains concepts in an understandable way	Use of presentations	Degree of difficulty of assignments	Solves doubts willingly	Structuring of the course	Provides support for students going above and beyond	Course recommendation based on relevance
0	0	340	5	2	7	6	9	2	1	8
1	1	253	6	5	8	6	2	1	2	9
2	2	680	7	7	6	5	4	2	3	1
3	3	806	9	6	7	1	5	9	4	6
4	4	632	8	10	8	4	6	6	9	9

#Check column names and data info

```
print(df.info())
```

Result:

```
Explains concepts in an understandable way  Use of presentations  \
0                                        2                    7
1                                        5                    8
2                                        7                    6
3                                        6                    7
4                                       10                    8

   Degree of difficulty of assignments  Solves doubts willingly  \
0                                    6                        9
1                                    6                        2
2                                    5                        4
3                                    1                        5
4                                    4                        6

   Structuring of the course  \
0                          2
1                          1
2                          2
```

```
3                    9
4                    6

   Provides support for students going above and beyond  \
0                                                   1
1                                                   2
2                                                   3
3                                                   4
4                                                   9

   Course recommendation based on relevance
0                                         8
1                                         9
2                                         1
3                                         6
4                                         9
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1001 entries, 0 to 1000
Data columns (total 10 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   Unnamed: 0                                  1001 non-null   int64
 1   Student ID                                  1001 non-null   int64
 2   Well versed with the subject                1001 non-null   int64
 3   Explains concepts in an understandable way  1001 non-null   int64
 4   Use of presentations                        1001 non-null   int64
 5   Degree of difficulty of assignments         1001 non-null   int64
 6   Solves doubts willingly                     1001 non-null   int64
 7   Structuring of the course                   1001 non-null   int64
 8   Provides support for students going above and beyond  1001 non-null   int64
 9   Course recommendation based on relevance    1001 non-null   int64
dtypes: int64(10)
memory usage: 78.3 KB
None
```

2. Understanding the Data

```
# Check number of rows, columns

print("Shape:", df.shape)
#Checking for number of rows and columns

print("Columns:", df.columns.tolist())          # To see the column names

print(df.info())                                #Basic information like data types and nulls

print(df.describe())                            # Quick stats for numbers like range
```

Result:

```
Shape: (1001, 10)
Columns: ['Unnamed: 0', 'Student ID', 'Well versed with the subject', 'Explains
concepts in an understandable way', 'Use of presentations', 'Degree of difficulty of
assignments', 'Solves doubts willingly', 'Structuring of the course', 'Provides support
for students going above and beyond', 'Course recommendation based on relevance']
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1001 entries, 0 to 1000
Data columns (total 10 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   Unnamed: 0                                    1001 non-null   int64
 1   Student ID                                    1001 non-null   int64
 2   Well versed with the subject                  1001 non-null   int64
 3   Explains concepts in an understandable way    1001 non-null   int64
 4   Use of presentations                          1001 non-null   int64
 5   Degree of difficulty of assignments           1001 non-null   int64
 6   Solves doubts willingly                       1001 non-null   int64
 7   Structuring of the course                     1001 non-null   int64
 8   Provides support for students going above and beyond  1001 non-null   int64
 9   Course recommendation based on relevance      1001 non-null   int64
dtypes: int64(10)
memory usage: 78.3 KB
None
       Unnamed: 0   Student ID  Well versed with the subject  \
count  1001.000000  1001.000000                   1001.000000
mean    500.000000   500.000000                      7.497502
std     289.108111   289.108111                      1.692998
min       0.000000     0.000000                      5.000000
```

25%	250.000000	250.000000	6.000000
50%	500.000000	500.000000	8.000000
75%	750.000000	750.000000	9.000000
max	1000.000000	1000.000000	10.000000

Explains concepts in an understandable way Use of presentations \

count	1001.000000	1001.000000
mean	6.081918	5.942058
std	2.597168	1.415853
min	2.000000	4.000000
25%	4.000000	5.000000
50%	6.000000	6.000000
75%	8.000000	7.000000
max	10.000000	8.000000

Degree of difficulty of assignments Solves doubts willingly \

count	1001.000000	1001.000000
mean	5.430569	5.474525
std	2.869046	2.874648
min	1.000000	1.000000
25%	3.000000	3.000000
50%	5.000000	6.000000
75%	8.000000	8.000000
max	10.000000	10.000000

Structuring of the course \

count	1001.000000
mean	5.636364
std	2.920212
min	1.000000
25%	3.000000
50%	6.000000
75%	8.000000
max	10.000000

Provides support for students going above and beyond \

count	1001.000000
mean	5.662338
std	2.891690
min	1.000000
25%	3.000000
50%	6.000000
75%	8.000000
max	10.000000

	Course recommendation based on relevance
count	1001.000000
mean	5.598402
std	2.886617
min	1.000000
25%	3.000000
50%	6.000000
75%	8.000000
max	10.000000

3. Cleaning the Dataset (Removing the unnecessary columns)

```
df_clean = df.drop(columns=['Unnamed: 0', 'Student ID'])

# Dropping unwanted columns

df_clean.head()
```

Result:

```
df_clean = df.drop(columns=['Unnamed: 0', 'Student ID'])

df_clean.head()
```

	Well versed with the subject	Explains concepts in an understandable way	Use of presentations	Degree of difficulty of assignments	Solves doubts willingly	Structuring of the course	Provides support for students going above and beyond	Course recommendation based on relevance
0	5	2	7	6	9	2	1	8
1	6	5	8	6	2	1	2	9
2	7	7	6	5	4	2	3	1
3	9	6	7	1	5	9	4	6
4	8	10	8	4	6	6	9	9

4. Finding the average rating for each question.

```
avg_ratings = df_clean.mean()                    # Average rating for each question

print("Average Ratings:")
print(avg_ratings)                               # Printing average ratings
```

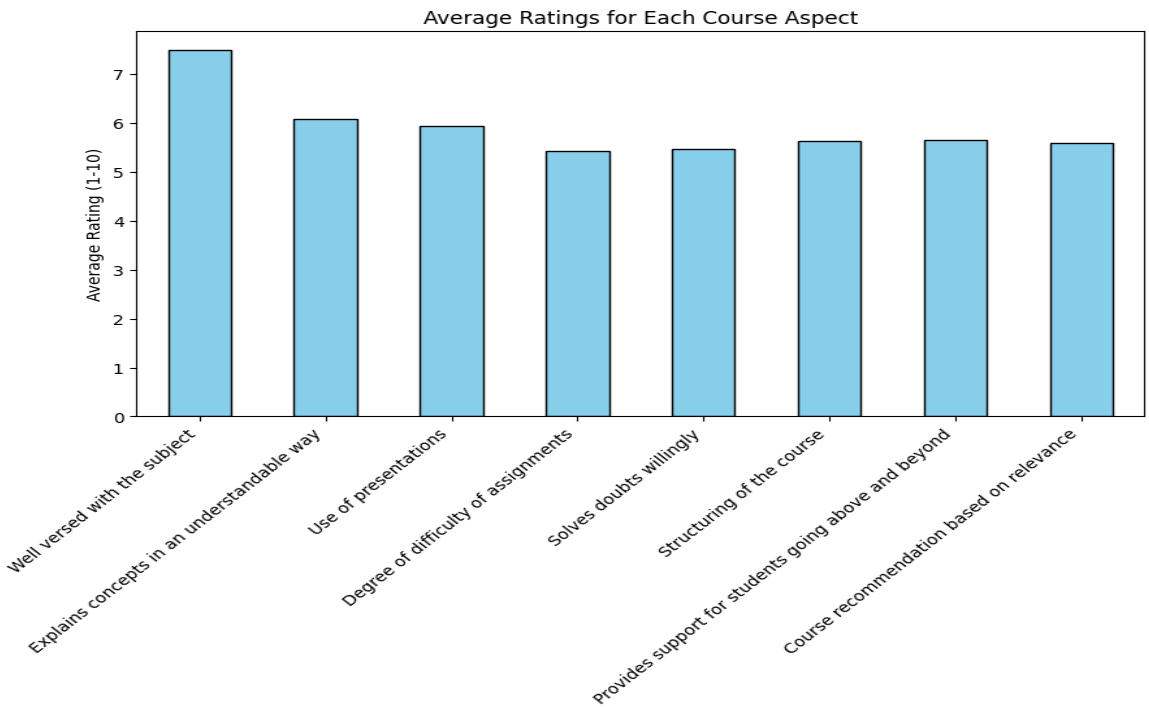
Result:

```
Average Ratings:
Well versed with the subject                          7.497502
Explains concepts in an understandable way            6.081918
Use of presentations                                  5.942058
Degree of difficulty of assignments                   5.430569
Solves doubts willingly                               5.474525
Structuring of the course                             5.636364
Provides support for students going above and beyond  5.662338
Course recommendation based on relevance              5.598402
dtype: float64
```

5. Visualize the Averages

```
import matplotlib.pyplot as plt

avg_ratings.plot(kind='bar', figsize=(10,5), color='skyblue', edgecolor='black')
plt.title("Average Ratings for Each Course Aspect")
plt.ylabel("Average Rating (1-10)")
plt.xticks(rotation=45, ha='right')
plt.show()
```

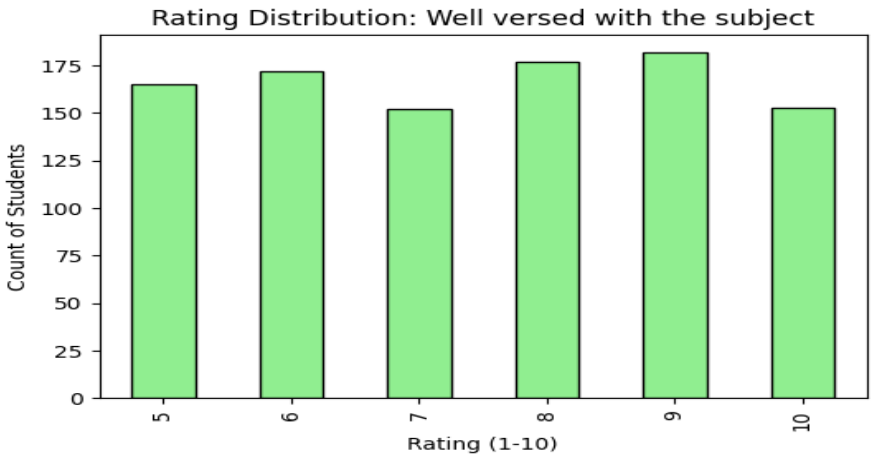
Result:

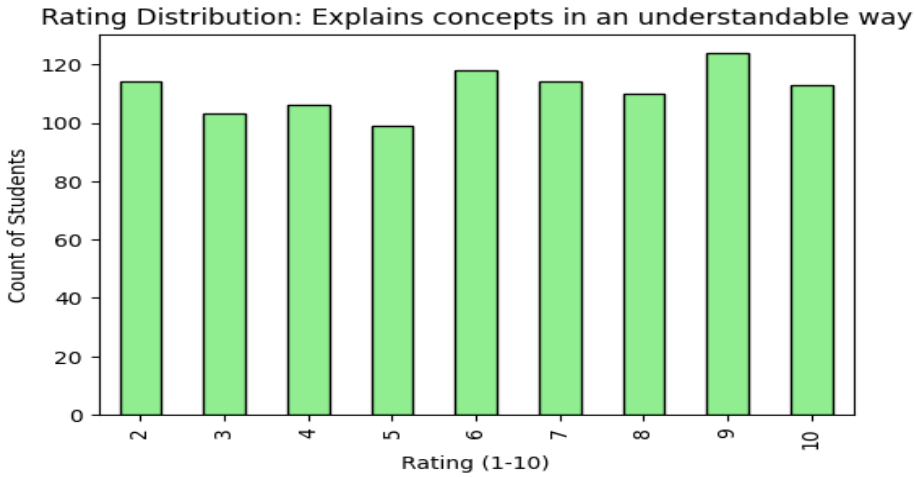


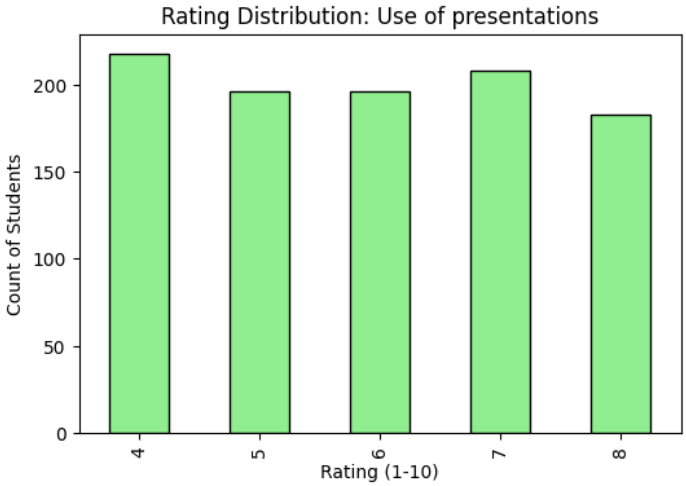
6. Distribution of Ranges

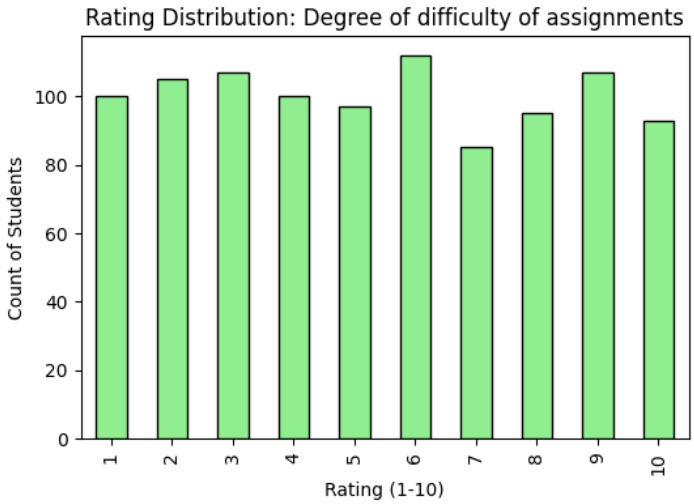
```
import matplotlib.pyplot as plt

for col in df_clean.columns:
    plt.figure(figsize=(6,4))
    df_clean[col].value_counts().sort_index().plot(kind='bar', color='lightgreen',
edgecolor='black')
    plt.title(f"Rating Distribution: {col}")
    plt.xlabel("Rating (1-10)")
    plt.ylabel("Count of Students")
    plt.show()
```

Result:

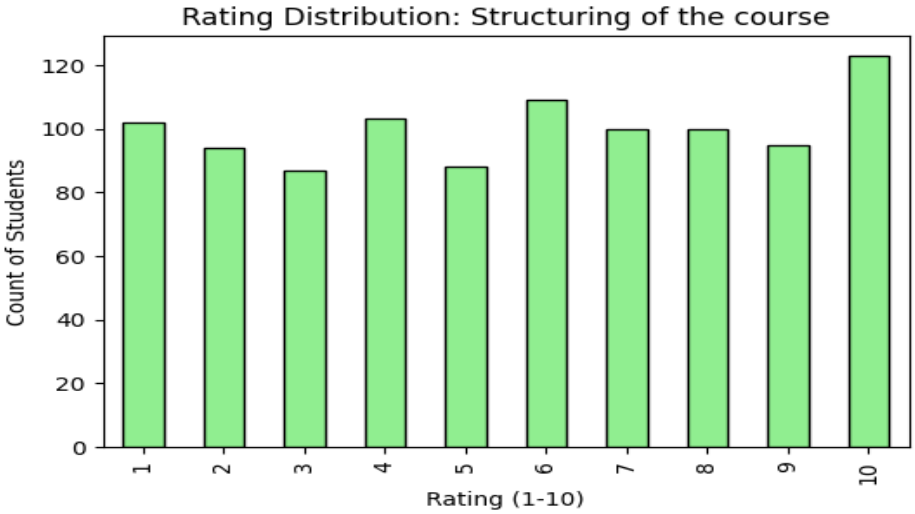


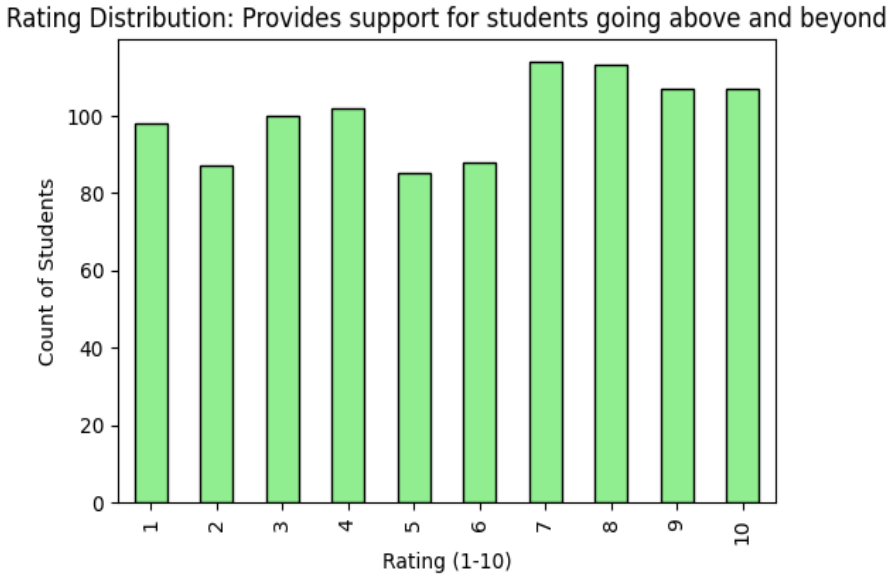



Rating Distribution: Use of presentations
Count of Students
0
50
100
150
200
4
5
6
7
8
Rating (1-10)


Rating Distribution: Degree of difficulty of assignments
Count of Students
0
20
40
60
80
100
1
2
3
4
5
6
7
8
9
10
Rating (1-10)


Rating Distribution: Solves doubts willingly
Count of Students
0
20
40
60
80
100
1
2
3
4
5
6
7
8
9
10
Rating (1-10)


Rating Distribution: Structuring of the course
Count of Students
0
20
40
60
80
100
120
1
2
3
4
5
6
7
8
9
10
Rating (1-10)


Rating Distribution: Provides support for students going above and beyond
Count of Students
0
20
40
60
80
100
1
2
3
4
5
6
7
8
9
10
Rating (1-10)


Rating Distribution: Course recommendation based on relevance
Count of Students
0
20
40
60
80
100
1
2
3
4
5
6
7
8
9
10
Rating (1-10)

7. Correlation Analysis

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

# Compute correlation matrix (ignoring ID columns)
corr = df.drop(columns=["Unnamed: 0", "Student ID"]).corr()

plt.figure(figsize=(10, 6))                # Plot the Heatmap
sns.heatmap(corr, annot=True, cmap="coolwarm", center=0)
plt.title("Correlation between Feedback Questions")
plt.show()
```

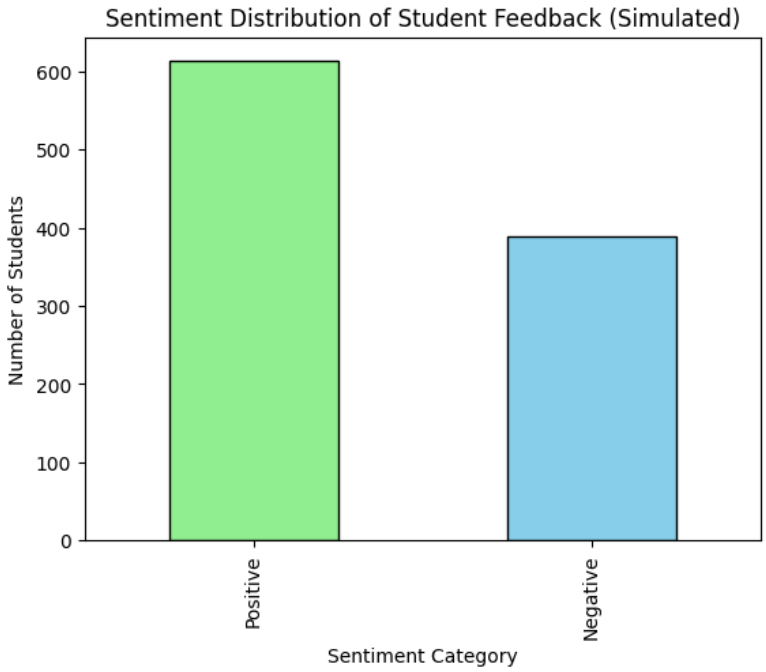
Result:

8. Sentiment Analysis (NLP)

```
import pandas as pd
import numpy as np
from textblob import TextBlob
import matplotlib.pyplot as plt
from wordcloud import WordCloud

# Load your dataset again
df = pd.read_csv("/feedback.csv")

# Step 1: Create synthetic comments based on ratings
# We'll map overall Course recommendation score to sentiment-like comments
def generate_comment(score):
    if score >= 8:
        return np.random.choice([
            "The event was excellent and very useful",
            "Loved the session, very engaging and informative",
            "Great experience, learned a lot"
        ])
    elif score >= 5:
        return np.random.choice([
            "It was okay, but could be improved",
            "The session was average, some parts were useful",
            "Not bad, but expected a bit more"
        ])
    else:
        return np.random.choice([
            "I did not like it, too boring",
            "The workshop was confusing and not helpful",
            "Poorly conducted, needs improvement"
        ])

df['Comments'] = df['Course recommendation based on
relevance'].apply(generate_comment)

# Step 2: Perform sentiment analysis with TextBlob
df['sentiment_score'] = df['Comments'].apply(lambda x:
TextBlob(str(x)).sentiment.polarity)
df['sentiment_label'] = df['sentiment_score'].apply(
    lambda x: 'Positive' if x > 0 else ('Negative' if x < 0 else 'Neutral')
)
```

```
# Step 3: Visualize sentiment distribution
df['sentiment_label'].value_counts().plot(
    kind='bar', color=['lightgreen', 'skyblue', 'salmon'], edgecolor='black'
)
plt.title("Sentiment Distribution of Student Feedback (Simulated)")
plt.ylabel("Number of Students")
plt.xlabel("Sentiment Category")
plt.show()

# Step 4: Word cloud
text = " ".join(comment for comment in df['Comments'])
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(text)

plt.figure(figsize=(10,5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title("Most Common Words in Feedback Comments (Simulated)")
plt.show()
```

Result:



9. Recommendations based on Findings

```
import pandas as pd

# Load your dataset
df = pd.read_csv("feedback.csv")

# Remove unwanted ID columns
df_clean = df.drop(columns=['Unnamed: 0', 'Student ID'])

# Calculate average ratings
avg_ratings = df_clean.mean()

# Recommendation system function
def generate_recommendations(avg_ratings):
    recs = []
    high_threshold = 7.5
    low_threshold = 6.0

    if avg_ratings['Well versed with the subject'] >= high_threshold:
        recs.append("Maintain instructors' subject expertise and encourage continuous
learning.")
    else:
        recs.append("Improve the subject knowledge of instructors through training.")

    if avg_ratings['Use of presentations'] < low_threshold:
        recs.append("Enhance presentation quality using better visuals and interactive
materials.")
```

```
    if avg_ratings['Degree of difficulty of assignments'] < low_threshold:
        recs.append("Adjust assignment difficulty to balance challenge and feasibility.")

    if avg_ratings['Solves doubts willingly'] < low_threshold:
        recs.append("Increase availability of instructors for doubt solving sessions.")

    if avg_ratings['Structuring of the course'] < low_threshold:
        recs.append("Revisit course structure, pacing, and flow to improve student
experience.")

    if avg_ratings['Provides support for students going above and beyond'] <
low_threshold:
        recs.append("Offer additional support for advanced and struggling students.")

    if avg_ratings['Course recommendation based on relevance'] < low_threshold:
        recs.append("Ensure course content is relevant and updated to match student and
industry needs.")

    return recs

# Generate and print recommendations
recommendations = generate_recommendations(avg_ratings)
print("Recommendations based on feedback ratings:\n")
for i, rec in enumerate(recommendations, 1):
    print(f"{i}. {rec}")
```

Result:

Recommendations based on feedback ratings:

1. Improve the subject knowledge of instructors through training.
2. Enhance presentation quality using better visuals and interactive materials.
3. Adjust assignment difficulty to balance challenge and feasibility.
4. Increase availability of instructors for doubt solving sessions.
5. Revisit course structure, pacing, and flow to improve student experience.
6. Offer additional support for advanced and struggling students.
7. Ensure course content is relevant and updated to match student and industry needs.