1. Importing Clean and prepare feedback dataset

```
import pandas as pd
from google.colab import files
uploaded = files.upload()
data = pd.read_csv("feedback.csv")
data.head()
```

Choose Files feedback.csv

- **feedback.csv**(text/csv) - 33786 bytes, last modified: 27/6/2025 - 100% done

```
Saving feedback.csv to feedback.csv
```

	StudentID	Name	Course	Overall_Event_Rating	Speaker_Clarity	Organization	Content_Usefulness	Recommend	Additional_Co
0	1	Fatima Martin	BA Economics	1	1	2	5	5	Excell mana
1	2	Anjali Fernandez	BBA	5	4	2	2	5	Too long, c
2	3	John Taylor	BA Economics	5	5	3	3	3	Speakers were hear a
3	4	Nisha Martin	BCom	4	3	2	5	3	Not clear, ra
4	5	Daniel Martin	BCA	3	2	1	4	1	Excell mana

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

2. Analyze ratings (1–5 scale) to find patterns of satisfaction

```
print(" Ratings Summary:")
print(data['Overall_Event_Rating'].describe())
```

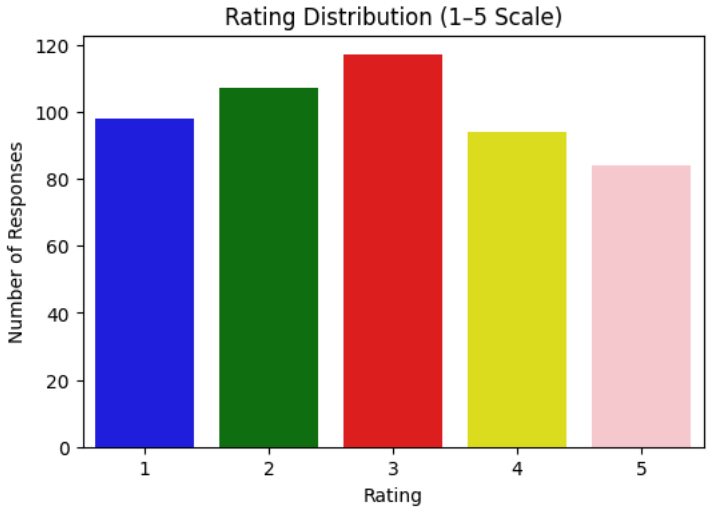
```
 Ratings Summary:
count    500.000000
mean       2.918000
std        1.361979
min        1.000000
25%        2.000000
50%        3.000000
75%        4.000000
max        5.000000
Name: Overall_Event_Rating, dtype: float64
```

```
print("\n Ratings Counts:")
print(data['Overall_Event_Rating'].value_counts().sort_index())
```

```
 Ratings Counts:
Overall_Event_Rating
1     98
2    107
3    117
4     94
5     84
Name: count, dtype: int64
```

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(6,4))
sns.countplot(x='Overall_Event_Rating', data=data, palette=['Blue','Green','Red','Yellow','Pink'])
plt.title('Rating Distribution (1–5 Scale)')
plt.xlabel('Rating')
plt.ylabel('Number of Responses')
plt.show()
```

```
/tmp/ipython-input-4-1967251060.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(x='Overall_Event_Rating', data=data, palette=['Blue','Green','Red','Yellow','Pink'])
```




```
mean_rating = data['Overall_Event_Rating'].mean()
if mean_rating >= 4:
    print(f"\n Average rating is {mean_rating:.2f}. High overall satisfaction.")
elif mean_rating >= 3:
    print(f"\n Average rating is {mean_rating:.2f}. Mixed/neutral satisfaction.")
else:
    print(f"\n Average rating is {mean_rating:.2f}. Low satisfaction. Needs improvement.")
```

```
 Average rating is 2.92. Low satisfaction. Needs improvement.
```

3. Use NLP tools to score sentiment in comments (positive/neutral/negative)

```
#installing textblob library
!pip install textblob
from textblob import TextBlob
```

```
Requirement already satisfied: textblob in /usr/local/lib/python3.11/dist-packages (0.19.0)
Requirement already satisfied: nltk>=3.9 in /usr/local/lib/python3.11/dist-packages (from textblob) (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk>=3.9->textblob) (8.2.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk>=3.9->textblob) (1.5.1)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk>=3.9->textblob) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk>=3.9->textblob) (4.67.1)
```

```
print(data.columns)
```

```
Index(['StudentID', 'Name', 'Course', 'Overall_Event_Rating',
       'Speaker_Clarity', 'Organization', 'Content_Usefulness', 'Recommend',
       'Additional_Comments'],
      dtype='object')
```

```
def score_sentiment(text):
    if pd.isna(text) or text.strip() == '':
        return 'Neutral'
    polarity = TextBlob(text).sentiment.polarity
    if polarity > 0.1:
        return 'Positive'
    elif polarity < -0.1:
        return 'Negative'
    else:
        return 'Neutral'

data['additional_comment_sentiment'] = data['Additional_Comments'].apply(score_sentiment)
print(data[['Additional_Comments', 'additional_comment_sentiment']].head())
print(data['additional_comment_sentiment'].value_counts())
```

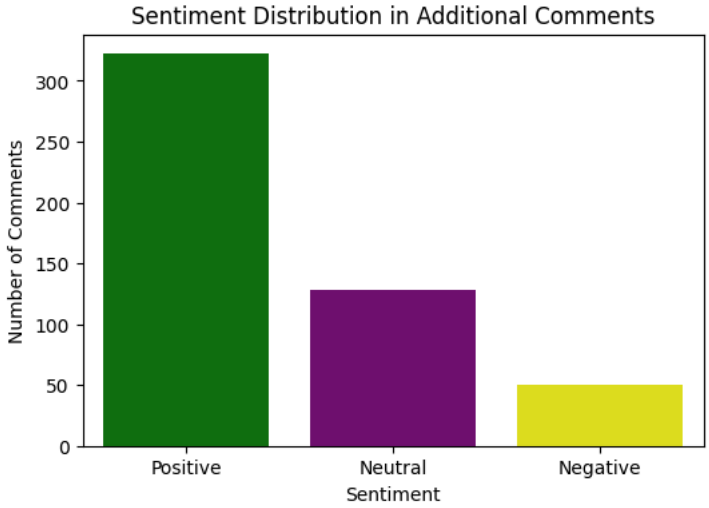
```
                     Additional_Comments additional_comment_sentiment
0            Excellent time management.                     Positive
1           Too long, could be shorter.                      Neutral
2  Speakers were hard to hear at times.                     Negative
```

```
3           Not clear, ran out of time.                      Neutral
4            Excellent time management.                     Positive
additional_comment_sentiment
Positive    322
Neutral     128
Negative     50
Name: count, dtype: int64
```

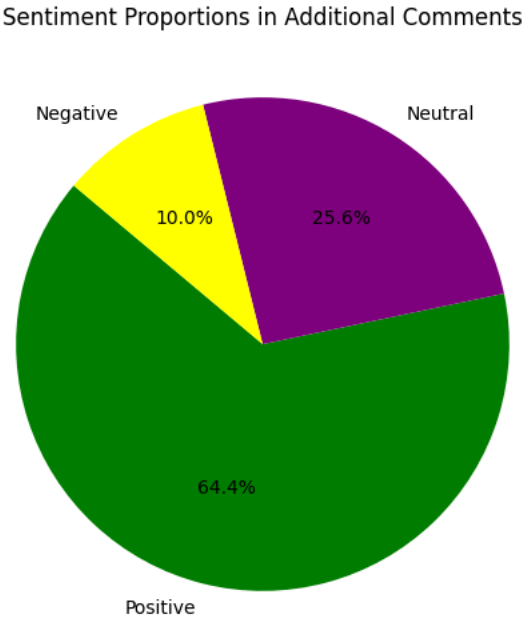
4. Visualize trends with beautiful charts and graphs

```
plt.figure(figsize=(6,4))
sns.countplot(
    x='additional_comment_sentiment',
    data=data,
    order=['Positive', 'Neutral', 'Negative'],
    palette=['green','purple','yellow']
)
plt.title('Sentiment Distribution in Additional Comments')
plt.xlabel('Sentiment')
plt.ylabel('Number of Comments')
plt.show()
```

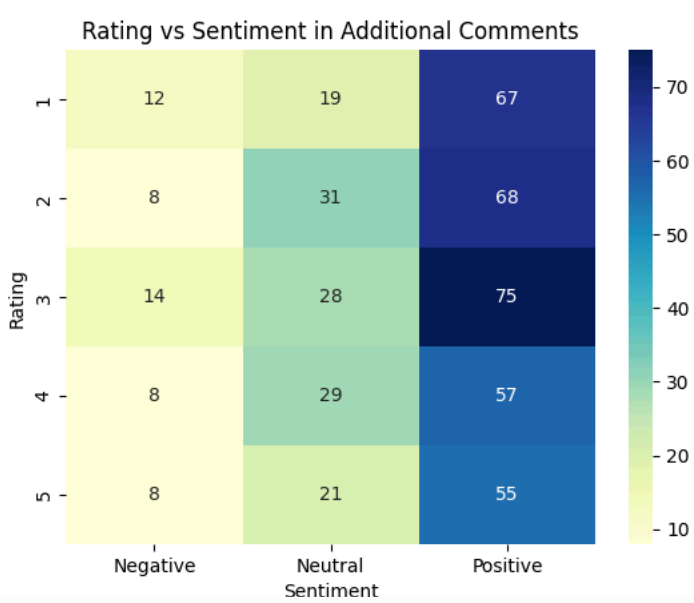
```
/tmp/ipython-input-10-2783385026.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(
```



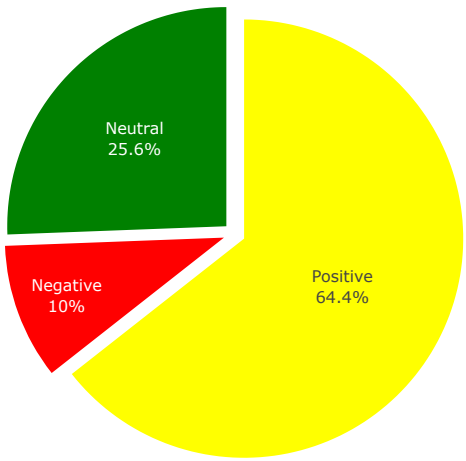

```
sentiment_counts = data['additional_comment_sentiment'].value_counts()
plt.figure(figsize=(6,6))
plt.pie(
    sentiment_counts,
    labels=sentiment_counts.index,
    autopct='%1.1f%%',
    colors=['green','purple','yellow'],
    startangle=140
)
plt.title('Sentiment Proportions in Additional Comments')
plt.show()
```



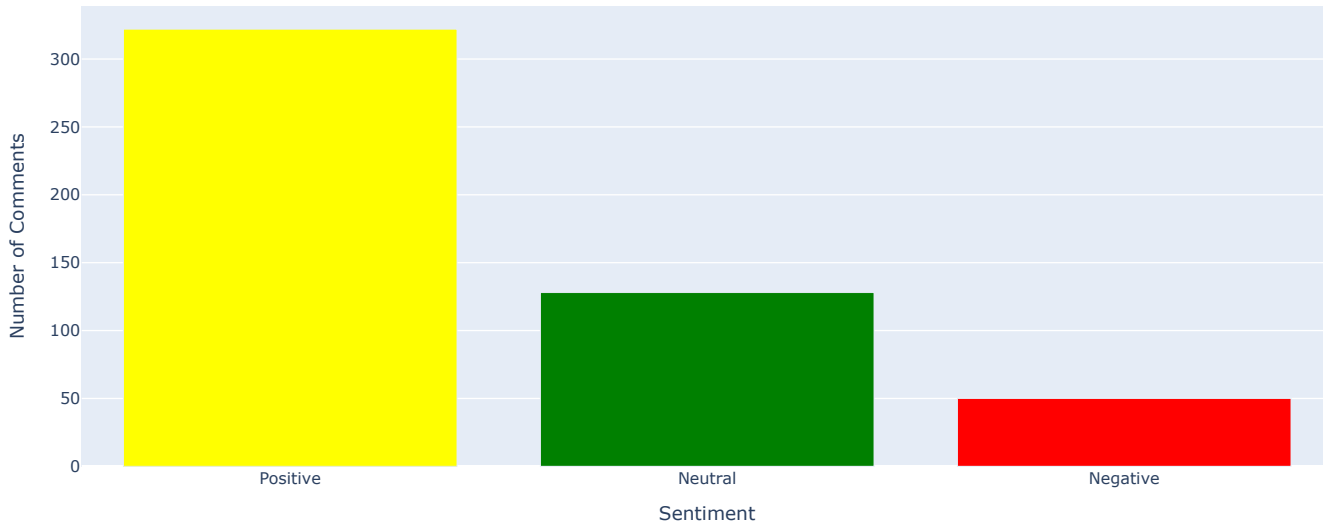

```
heat_data = pd.crosstab(data['Overall_Event_Rating'], data['additional_comment_sentiment'])
sns.heatmap(heat_data, annot=True, cmap='YlGnBu')
plt.title('Rating vs Sentiment in Additional Comments')
plt.xlabel('Sentiment')
plt.ylabel('Rating')
plt.show()
```




```
import plotly.express as px

fig = px.pie(
    values=sentiment_counts.values,
    names=sentiment_counts.index,
    title='Interactive Sentiment Proportions in Additional Comments',
    color_discrete_sequence=['yellow', 'green', 'red']
)
fig.update_traces(textinfo='percent+label', pull=[0.05, 0.05, 0.05])
fig.show()
```

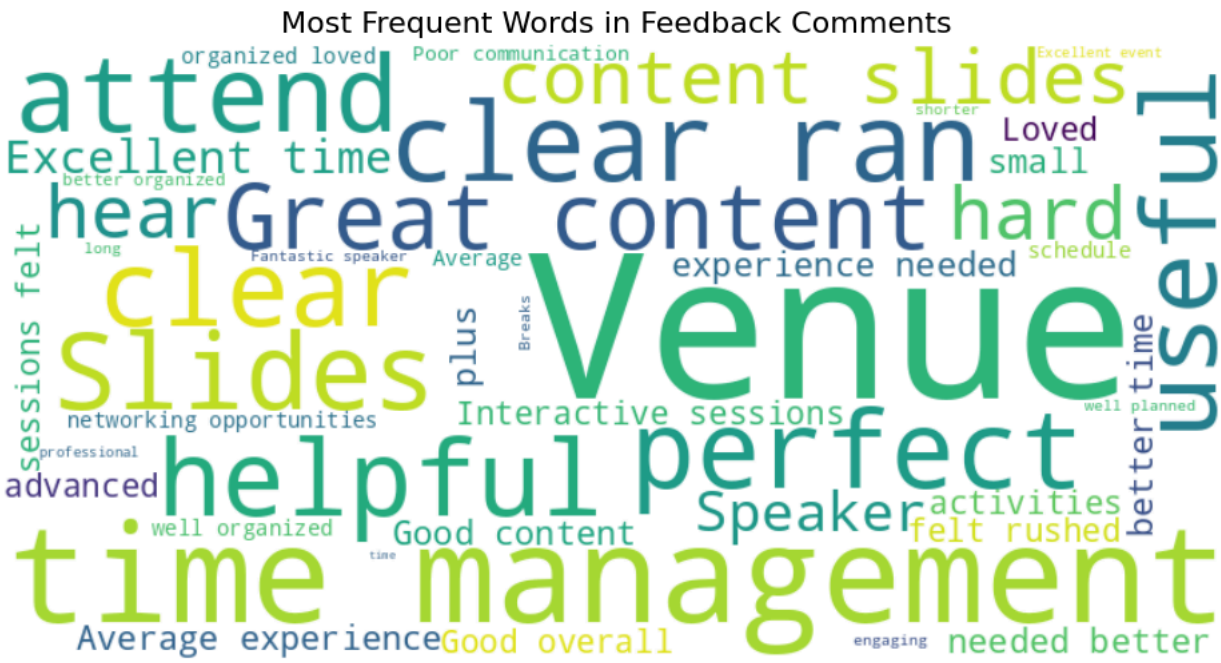



```
fig = px.bar(
    x=sentiment_counts.index,
    y=sentiment_counts.values,
    color=sentiment_counts.index,
    color_discrete_sequence=['yellow', 'green', 'red'],
    title='Interactive Sentiment Distribution in Additional Comments',
    labels={'x': 'Sentiment', 'y': 'Number of Comments'}
)
fig.show()
```




```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Combine all non-empty comments into one string
text = ' '.join(data['Additional_Comments'].dropna().astype(str))

# Generate a WordCloud
wordcloud = WordCloud(
    width=800,
    height=400,
    background_color='white',
    colormap='viridis',
```

```
    max_words=100
).generate(text)

# Display the WordCloud
plt.figure(figsize=(12, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Most Frequent Words in Feedback Comments', fontsize=16)
plt.show()
```



5. **Suggest Improvements for future events**

a. **Improve Speaker Clarity**

Use better audio systems and guide speakers for clearer delivery.

b. **Fix Organizational Gaps**

Ensure smooth coordination, session timing, and flow.

c. **Make Content More Relevant**

Align topics with student interests through pre-event surveys.

d. **Shorten and Structure Sessions**

Keep sessions concise and well-timed to avoid fatigue.

e. **Encourage Interactive Sessions**

Include Q&A, polls, and discussions to boost engagement.