```
# Load and inspect the data
df = pd.read_csv('student_feedback.csv')
print("Dataset Shape:", df.shape)
print("\nFirst few rows:")
print(df.head())
print("\nColumn Info:")
print(df.info())
print("\nBasic Statistics:")
print(df.describe())
```

```
Dataset Shape: (1001, 10)

First few rows:
   Unnamed: 0  Student ID  Well versed with the subject  \
0           0         340                             5
1           1         253                             6
2           2         680                             7
3           3         806                             9
4           4         632                             8

   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

Column Info:
<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
```

IMPORTING LIBRARIES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.cluster import hierarchy
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, DBSCAN
from sklearn.manifold import TSNE
from sklearn.metrics import silhouette_score
from sklearn.feature_selection import mutual_info_classif
import warnings
warnings.filterwarnings('ignore')
```

SETTING MY PLOT STYLE

```
plt.style.use('seaborn-v0_8')
sns.set_palette("husl")
```

DATA CLEANING

```
# Cleaning column names
df.columns = [c.strip().lower().replace(" ", "_") for c in df.columns]

print("\n📄 Columns in dataset:")
print(df.columns.tolist())
```

```
# Removing duplicates
df.drop_duplicates(inplace=True)

# Handling missing values
df.fillna("", inplace=True)
```

```
📄 Columns in dataset:
['unnamed:_0', 'student_id', 'well_versed_with_the_subject', 'explains_concepts_ir
```

HANDLING MULTIPLE RATING COLUMNS

```
# Standardize column names
df.columns = df.columns.str.strip().str.lower().str.replace(" ", "_").str.replac

# Remove unnamed columns
df = df.loc[:, ~df.columns.str.contains('unnamed')]
print("\n🧹 Cleaned column names:")
print(list(df.columns))

# Handle missing values
missing = df.isnull().sum()
print("\nMissing values per column:")
print(missing[missing > 0])

# Fill numeric NaNs with column mean
num_cols = df.select_dtypes(include=[np.number]).columns
df[num_cols] = df[num_cols].apply(lambda x: x.fillna(x.mean()))

# Drop duplicates
df.drop_duplicates(inplace=True)
```

```
🧹 Cleaned column names:
['student_id', 'well_versed_with_the_subject', 'explains_concepts_in_an_understanc

Missing values per column:
Series([], dtype: int64)
```

DETECTING RATING COLUMNS

```
# ==============================================
#  dataset has only numeric ratings + overall_rating
rating_columns = [
    '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',
    'overall_rating'
]

# Ensuring they exist
rating_columns = [col for col in rating_columns if col in df.columns]
print("\n✅ Using these rating columns:")
print(rating_columns)

# Computing overall average rating
df['average_rating'] = df[rating_columns].mean(axis=1)
```

```
✅ Using these rating columns:
['well_versed_with_the_subject', 'explains_concepts_in_an_understandable_way', 'us
```

DETECTING TEXT FEEDBACK COLUMN

```
text_columns = [col for col in df.columns if df[col].dtype == 'object' and col n

if len(text_columns) > 0:
    text_col = text_columns[0]
    print(f"\n🗣️ Detected text feedback column: {text_col}")
    has_text = True
else:
    print("\n⚠️ No text feedback column found. Skipping sentiment analysis...")
    has_text = False
```

```
⚠️ No text feedback column found. Skipping sentiment analysis...
```

EXPLORATORY DATA ANALYSIS (EDA)

```
print("\n📊 Summary Statistics:")
display(df[rating_columns + ['average_rating']].describe())

# Correlation heatmap
plt.figure(figsize=(10,6))
sns.heatmap(df[rating_columns + ['average_rating']].corr(), annot=True, cmap='co
plt.title("Correlation Heatmap of Ratings")
plt.show()
```

```
# Rating distribution
plt.figure(figsize=(8,5))
sns.histplot(df['average_rating'], bins=5, kde=True)
plt.title("Distribution of Average Ratings")
plt.xlabel("Average Rating")
plt.ylabel("Number of Students")
plt.show()

# Boxplots for individual criteria
plt.figure(figsize=(12,6))
sns.boxplot(data=df[rating_columns])
plt.title("Distribution of Ratings by Category")
plt.xticks(rotation=45)
plt.show()
```

INSIGHTS & RECOMMENDATIONS

```
avg_ratings = df[rating_columns + ['average_rating']].mean().sort_values(ascendi
print("\n📈 Average Ratings per Category:")
display(avg_ratings)
```

```
top_aspects = avg_ratings.head(3).index.tolist()
weak_aspects = avg_ratings.tail(3).index.tolist()

print("\n✅ Strengths (Top-rated aspects):", top_aspects)
print("⚠️ Areas for Improvement:", weak_aspects)

# Z-score analysis for outliers
from scipy import stats
z_scores = np.abs(stats.zscore(df[rating_columns]))
outliers = (z_scores > 3).sum().sum()
print(f"\n🔍 Outlier count across all ratings: {outliers}")
```

SUMMARY REPORT

```
print("\n📋 SUMMARY REPORT")
print("="*60)
print(f"Total Students: {len(df)}")
print(f"Average Overall Rating: {df['average_rating'].mean():.2f} / 5")
if has_text:
    print(f"Most Common Sentiment: {df['sentiment'].mode()[0]}")
print(f"Top Strengths: {', '.join(top_aspects)}")
print(f"Needs Improvement: {', '.join(weak_aspects)}")
```

```
print("="*60)

print("\n🎯 Recommendations:")
for col in weak_aspects:
    print(f"- Consider improving '{col.replace('_',' ')}' through better engagemen

print("\n✅ Analysis Complete!")
```

```
📋 SUMMARY REPORT
============================================================
Total Students: 1001
Average Overall Rating: 5.92 / 5
Top Strengths: well_versed_with_the_subject, explains_concepts_in_an_understandabl
Needs Improvement: course_recommendation_based_on_relevance, solves_doubts_willing
============================================================

🎯 Recommendations:
- Consider improving 'course recommendation based on relevance' through better eng
- Consider improving 'solves doubts willingly' through better engagement or cleare
- Consider improving 'degree of difficulty of assignments' through better engageme

✅ Analysis Complete!
```