Import Packages

In [1]:

```
import pandas as pd
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error, accuracy_score
from sklearn.metrics import accuracy_score, classification_report
from sklearn.ensemble import RandomForestClassifier
```

In [2]:

```
data = pd.read_csv('finalETLdata.csv')
data
```

Out[2]:

illset4	Skillset5	Location	...	Language1	Language2	Language3	Language4	Language5	ResearchInterests	Attendance	SkillCount	LanguageCount
known	Unknown	New York	...	Chinese	Japanese	Spanish	German	French	Biomedical Engineering	0.53	1	5
Data nalysis	Programming	Boston	...	French	English	Chinese	Spanish	Japanese	Urban Planning	0.37	5	5
Artistic	Problem Solving	Chicago	...	Spanish	Japanese	German	French	Unknown	Nanotechnology	0.25	5	4
known	Unknown	Chicago	...	Japanese	Chinese	Spanish	French	Unknown	Space Exploration	0.14	3	4
known	Unknown	Chicago	...	English	Unknown	Unknown	Unknown	Unknown	Climate Change	0.15	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
Public eaking	Data Analysis	Chicago	...	French	Chinese	English	Unknown	Unknown	Astrophysics	0.07	5	3
known	Unknown	New York	...	Spanish	German	Chinese	Unknown	Unknown	Data Science	0.79	2	3
lership	Problem Solving	New York	...	Spanish	German	Japanese	Chinese	French	Space Exploration	0.29	5	5
lership	Problem Solving	Houston	...	German	Unknown	Unknown	Unknown	Unknown	Space Exploration	0.38	5	1
known	Unknown	New York	...	Japanese	Spanish	Chinese	French	German	Bioinformatics	0.03	1	5

In [3]:

```
df = pd.DataFrame(data)
df
```

Out[3]:

	StudentID	Name	AcademicInterest	ExtracurricularActivities	Skillset1	Skillset2	Skillset3	Skillset4	Skillset5	Location	...	Lan
0	1	Aaron Jordan	Psychology	Debate Club	Problem Solving	Unknown	Unknown	Unknown	Unknown	New York	...	C
1	2	Kevin Smith	Psychology	Debate Club	Leadership	Problem Solving	Public Speaking	Data Analysis	Programming	Boston	...	
2	3	William May	History	Volunteer Group	Data Analysis	Leadership	Public Speaking	Artistic	Problem Solving	Chicago	...	S
3	4	Valerie Miranda	Computer Science	Volunteer Group	Public Speaking	Data Analysis	Problem Solving	Unknown	Unknown	Chicago	...	Ja
4	5	Samantha Hill	Computer Science	Sports Team	Data Analysis	Unknown	Unknown	Unknown	Unknown	Chicago	...	
...	...	...	...	...	...	...	...	...	...	...	...	
995	996	Justin Jordan	History	Sports Team	Leadership	Artistic	Programming	Public Speaking	Data Analysis	Chicago	...	
996	997	Lauren Baker	Psychology	Art Club	Data Analysis	Leadership	Unknown	Unknown	Unknown	New York	...	S
997	998	Charles Lewis	Mathematics	Volunteer Group	Public Speaking	Programming	Artistic	Leadership	Problem Solving	New York	...	S
998	999	Christopher Lewis	Physics	Sports Team	Public Speaking	Data Analysis	Artistic	Leadership	Problem Solving	Houston	...	C
999	1000	Corey Bond	History	Coding Club	Public Speaking	Unknown	Unknown	Unknown	Unknown	New York	...	Ja

1000 rows × 23 columns

Label Encoding

In [4]:

```
from sklearn.preprocessing import LabelEncoder

# Define the columns you want to apply label encoding to
columns_to_encode = ['AcademicInterest', 'ExtracurricularActivities', 'YearOfStudy','Major', 'ResearchInterests'

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Apply label encoding to selected columns
for column in columns_to_encode:
    df[column] = label_encoder.fit_transform(df[column])

# Display the modified DataFrame
df
```

Out[4]:

killset5	Location	...	Language1	Language2	Language3	Language4	Language5	ResearchInterests	Attendance	SkillCount	LanguageCount	CompositeS
nknown	New York	...	Chinese	Japanese	Spanish	German	French	4	0.53	1	5	
amming	Boston	...	French	English	Chinese	Spanish	Japanese	24	0.37	5	5	
Problem Solving	Chicago	...	Spanish	Japanese	German	French	Unknown	15	0.25	5	4	
nknown	Chicago	...	Japanese	Chinese	Spanish	French	Unknown	22	0.14	3	4	
nknown	Chicago	...	English	Unknown	Unknown	Unknown	Unknown	6	0.15	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	
Data Analysis	Chicago	...	French	Chinese	English	Unknown	Unknown	1	0.07	5	3	
nknown	New York	...	Spanish	German	Chinese	Unknown	Unknown	9	0.79	2	3	
Problem Solving	New York	...	Spanish	German	Japanese	Chinese	French	22	0.29	5	5	
Problem Solving	Houston	...	German	Unknown	Unknown	Unknown	Unknown	22	0.38	5	1	
nknown	New York	...	Japanese	Spanish	Chinese	French	German	3	0.03	1	5	

Task 1: GPA prediction

In [5]:

```
# Define predictor variables (X) and target variable (y) for GPA prediction
X_gpa = df[['StudentID', 'SkillCount', 'LanguageCount', 'CompositeScore']]
y_gpa = df['GPA']
```

In [6]:

```
# Split the data into training and testing sets for GPA prediction
X_train_gpa, X_test_gpa, y_train_gpa, y_test_gpa = train_test_split(X_gpa, y_gpa, test_size=0.2, random_state=42
```

In [7]:

```
# Train linear regression model for GPA prediction
lr_gpa = LinearRegression()
lr_gpa.fit(X_train_gpa, y_train_gpa)

# Evaluate linear regression model for GPA prediction
y_pred_gpa = lr_gpa.predict(X_test_gpa)
mse_gpa = mean_squared_error(y_test_gpa, y_pred_gpa)
print("Mean Squared Error (GPA Prediction):", mse_gpa)
```

```
Mean Squared Error (GPA Prediction): 0.006415792969882901
```

In [8]:
```
# Step 5: Make Predictions
# Use the trained model to predict the next GPA of each student
next_gpa_predictions = lr_gpa.predict(X_gpa)
df['NextGPA'] = next_gpa_predictions

# Display the DataFrame with predicted next GPA
print(df[['StudentID', 'GPA', 'Attendance', 'NextGPA']])
```

```
     StudentID   GPA  Attendance   NextGPA
0            1  3.27        0.53  3.289579
1            2  3.17        0.37  3.118852
2            3  2.09        0.25  2.324098
3            4  2.56        0.14  2.561646
4            5  2.01        0.15  2.201725
..         ...   ...         ...       ...
995        996  2.21        0.07  2.270511
996        997  3.62        0.79  3.685658
997        998  3.09        0.29  2.999678
998        999  3.17        0.38  3.116268
999       1000  2.16        0.03  2.217429

[1000 rows x 4 columns]
```

Task 2: Pass Fail Prediction

In [9]:
```
ine predictor variables (X) and target variable (y) for pass/fail prediction
hold = 3.0  # Threshold GPA value for pass/fail classification
assOrFail'] = (df['GPA'] >= threshold).astype(int)  # Create binary target variable for pass/fail classification

assOrFail']
= df[['StudentID', 'SkillCount', 'LanguageCount', 'CompositeScore']]  # Exclude GPA and PassOrFail from predictor
= df['PassOrFail']
```

In [10]:
```
# Split the data into training and testing sets for pass/fail prediction
X_train_pf, X_test_pf, y_train_pf, y_test_pf = train_test_split(X_pf, y_pf, test_size=0.2, random_state=42)
```

In [11]:
```
# Train logistic regression model for pass/fail prediction
lr_pf = LogisticRegression()
lr_pf.fit(X_train_pf, y_train_pf)

# Step 4: Evaluate Model Performance
# For pass/fail classification
y_pred_pf = lr_pf.predict(X_test_pf)
accuracy_pf = accuracy_score(y_test_pf, y_pred_pf)
print("Accuracy (Pass/Fail Classification):", accuracy_pf)
print("Classification Report (Pass/Fail Classification):\n", classification_report(y_test_pf, y_pred_pf))
```

```
Accuracy (Pass/Fail Classification): 0.985
Classification Report (Pass/Fail Classification):
               precision    recall  f1-score   support

           0       0.98      0.99      0.98        91
           1       0.99      0.98      0.99       109

    accuracy                           0.98       200
   macro avg       0.98      0.99      0.98       200
weighted avg       0.99      0.98      0.99       200
```

In [12]:
```

# Display the DataFrame with pass or fail predictions
print(df[['StudentID', 'GPA', 'Attendance', 'CompositeScore', 'NextGPA','PassOrFail']])
```

```
     StudentID   GPA  Attendance  CompositeScore   NextGPA  PassOrFail
0            1  3.27        0.53            1.90  3.289579           1
1            2  3.17        0.37            1.77  3.118852           1
2            3  2.09        0.25            1.17  2.324098           0
3            4  2.56        0.14            1.35  2.561646           0
4            5  2.01        0.15            1.08  2.201725           0
..         ...   ...         ...             ...       ...         ...
995        996  2.21        0.07            1.14  2.270511           0
996        997  3.62        0.79            2.21  3.685658           1
997        998  3.09        0.29            1.69  2.999678           1
998        999  3.17        0.38            1.78  3.116268           1
999       1000  2.16        0.03            1.10  2.217429           0

[1000 rows x 6 columns]
```

Task 3: Engagement Predictions

In [13]:

```
# Filter out students who have passed based on GPA
engaged_students = df[df['GPA'] >= threshold]
```

In [14]:

```
# Define engagement categories based on specified criteria
def define_engagement(row):
    if row['ExtracurricularActivities'] > 0 and row['ResearchInterests'] is not None:
        if row['SkillCount'] > 3 and row['LanguageCount'] > 3:
            return "Extremely Engaged"
        elif row['SkillCount'] > 1 and row['LanguageCount'] > 1:
            return "Moderately Engaged"
        elif row['SkillCount'] == 1 and row['LanguageCount'] == 1:
            return "Low Engaged"
        else:
            return "Not Engaged"
    else:
        return "Not Engaged"

# Apply the define_engagement function to create the 'Engagement' variable
engaged_students['Engagement'] = engaged_students.apply(lambda row: define_engagement(row), axis=1)

# Label encode the 'Engagement' variable
label_encoder = LabelEncoder()
engaged_students['Engagement'] = label_encoder.fit_transform(engaged_students['Engagement'])

engaged_students
```

```
/var/folders/33/03_543s9719fh3y8mrxhst580000gn/T/ipykernel_46941/1561929157.py:16: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  engaged_students['Engagement'] = engaged_students.apply(lambda row: define_engagement(row), axis=1)
/var/folders/33/03_543s9719fh3y8mrxhst580000gn/T/ipykernel_46941/1561929157.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  engaged_students['Engagement'] = label_encoder.fit_transform(engaged_students['Engagement'])
```

Out[14]:

	StudentID	Name	AcademicInterest	ExtracurricularActivities	Skillset1	Skillset2	Skillset3	Skillset4	Skillset5	Location	...	Langu
0	1	Aaron Jordan	5	2	Problem Solving	Unknown	Unknown	Unknown	Unknown	New York	...	G
1	2	Kevin Smith	5	2	Leadership	Problem Solving	Public Speaking	Data Analysis	Programming	Boston	...	Sp
6	7	Jason Hale	2	0	Public Speaking	Problem Solving	Data Analysis	Artistic	Unknown	New York	...	G
7	8	Anthony Wright	1	5	Programming	Public Speaking	Artistic	Leadership	Unknown	Houston	...	Unl
9	10	Peter Yu	0	4	Programming	Data Analysis	Problem Solving	Leadership	Unknown	Los Angeles	...	Unl
...	...	...	...	...	...	...	...	...	...	...	...	
990	991	Gabriel Nolan	5	4	Programming	Problem Solving	Unknown	Unknown	Unknown	Los Angeles	...	Unl
993	994	Jeffrey Williams	1	4	Data Analysis	Artistic	Unknown	Unknown	Unknown	New York	...	Unl
996	997	Lauren Baker	5	0	Data Analysis	Leadership	Unknown	Unknown	Unknown	New York	...	Unl
997	998	Charles Lewis	3	5	Public Speaking	Programming	Artistic	Leadership	Problem Solving	New York	...	Cl
998	999	Christopher Lewis	4	4	Public Speaking	Data Analysis	Artistic	Leadership	Problem Solving	Houston	...	Unl

501 rows × 26 columns

In [15]:

```
# Step 4: Define predictor variables (X) and target variable (y)
X = engaged_students[['StudentID','GPA', 'CompositeScore']]
y = engaged_students['Engagement']
```

In [16]:

```
# Step 5: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [17]:

```
# Step 6: Train the Random Forest Classifier
rf_classifier = RandomForestClassifier()
rf_classifier.fit(X_train, y_train)
```

Out[17]:

```
▾ RandomForestClassifier

RandomForestClassifier()
```

In [18]:

```
# Step 7: Make predictions
y_pred = rf_classifier.predict(X_test)
```

In [19]:

```
# Step 8: Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.26732673267326734
Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00        28
           1       0.00      0.00      0.00         2
           2       0.35      0.38      0.36        37
           3       0.27      0.38      0.31        34

    accuracy                           0.27       101
   macro avg       0.15      0.19      0.17       101
weighted avg       0.22      0.27      0.24       101


/Users/amjadalikudsi/anaconda3/lib/python3.11/site-packages/sklearn/metrics/_classification.py:1469: UndefinedM
etricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. U
se `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/Users/amjadalikudsi/anaconda3/lib/python3.11/site-packages/sklearn/metrics/_classification.py:1469: UndefinedM
etricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. U
se `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/Users/amjadalikudsi/anaconda3/lib/python3.11/site-packages/sklearn/metrics/_classification.py:1469: UndefinedM
etricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. U
se `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Bonus Task: Prediction that nextGPA is not being hampered based on Engagement

In [20]:

```
# Step 5: Predict whether their next GPA will not be hampered
# Here, you can use additional features such as GPA, attendance, composite score
# Define predictor variables (X) and target variable (y)
X_gpa = engaged_students[['StudentID', 'GPA', 'CompositeScore']]
# Assuming you have the target variable 'NextGPA' which indicates whether the next GPA is hampered or not

# Make predictions
next_gpa_predictions = rf_classifier.predict(X_gpa)

# Add predictions as a new column to the DataFrame
engaged_students['EngagementMetrics'] = next_gpa_predictions

# Display the DataFrame with predictions
print(engaged_students[['StudentID', 'GPA', 'NextGPA', 'EngagementMetrics']])
```

```
     StudentID   GPA   NextGPA  EngagementMetrics
0            1  3.27  3.289579                  0
1            2  3.17  3.118852                  0
6            7  3.35  3.356055                  0
7            8  3.64  3.673059                  2
9           10  3.03  2.971565                  2
..         ...   ...       ...                ...
990        991  3.37  3.354229                  3
993        994  3.23  3.181503                  3
996        997  3.62  3.685658                  3
997        998  3.09  2.999678                  0
998        999  3.17  3.116268                  3

[501 rows x 4 columns]

/var/folders/33/03_543s9719fh3y8mrxhst580000gn/T/ipykernel_46941/2134801628.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  engaged_students['EngagementMetrics'] = next_gpa_predictions
```

In [21]:

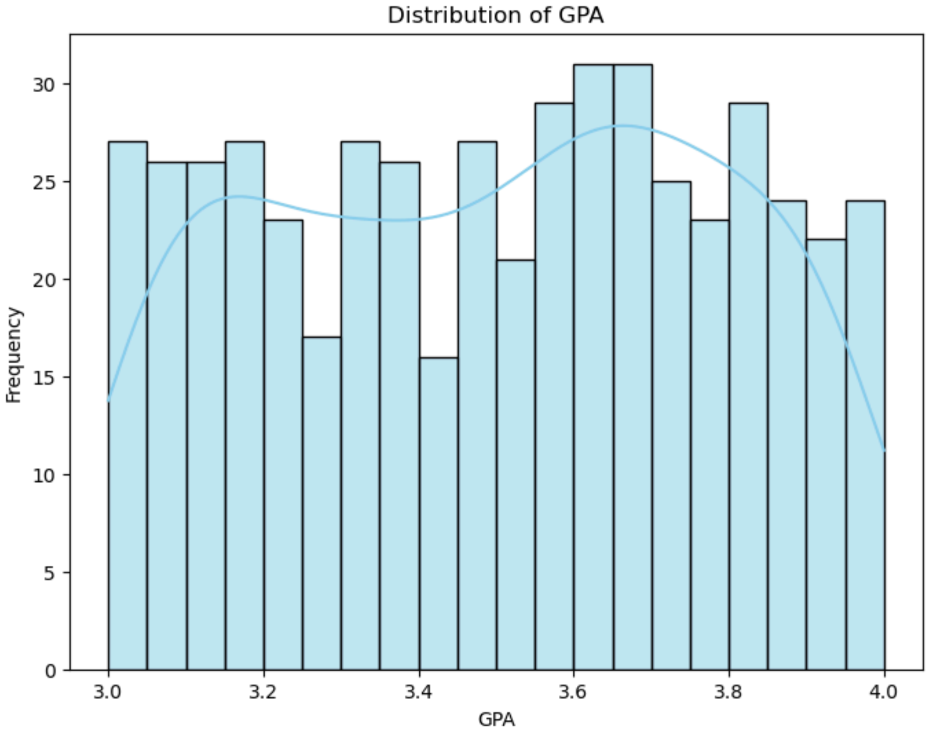
```
# Assuming 'new_students' DataFrame contains the data including the predicted next GPA and the current GPA

# Define a function to check if next GPA is less than current GPA
def check_pass_fail(row):
    if row['NextGPA'] < threshold:
        return 0  # If next GPA is less than current GPA, return 0 (Fail)
    else:
        return 1  # If next GPA is greater than or equal to current GPA, return 1 (Pass)

# Apply the function to create the new_pass_fail column
engaged_students['new_pass_fail'] = engaged_students.apply(check_pass_fail, axis=1)

# Display the updated DataFrame with the new_pass_fail column
print(engaged_students[['StudentID', 'GPA', 'NextGPA', 'EngagementMetrics','new_pass_fail']])
```

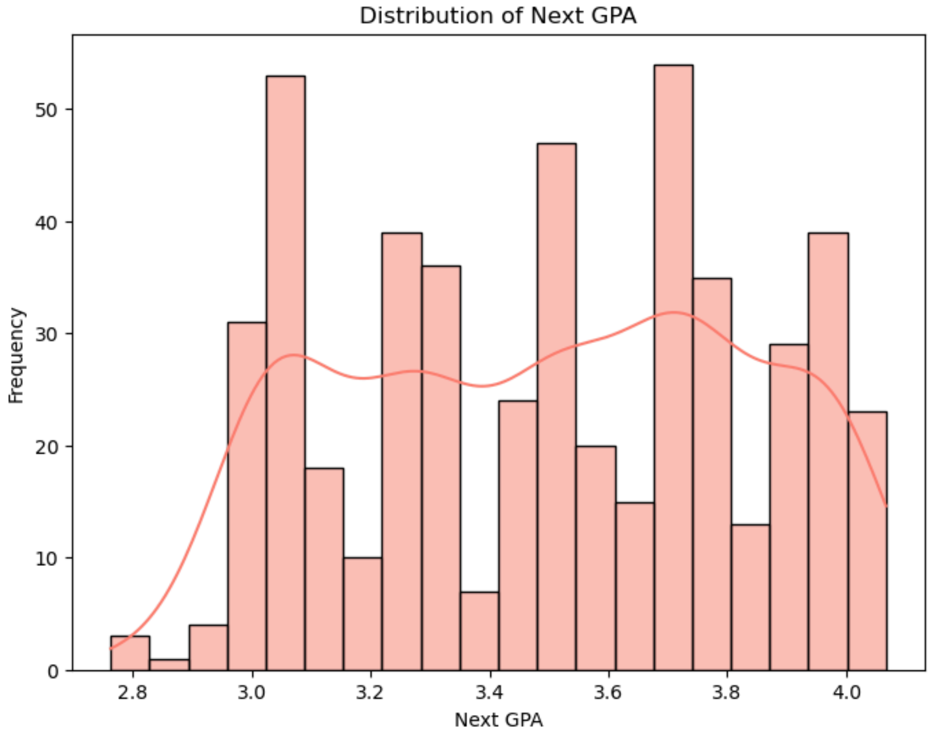
```
     StudentID   GPA   NextGPA  EngagementMetrics  new_pass_fail
0            1  3.27  3.289579                  0              1
1            2  3.17  3.118852                  0              1
6            7  3.35  3.356055                  0              1
7            8  3.64  3.673059                  2              1
9           10  3.03  2.971565                  2              0
..         ...   ...       ...                ...            ...
990        991  3.37  3.354229                  3              1
993        994  3.23  3.181503                  3              1
996        997  3.62  3.685658                  3              1
997        998  3.09  2.999678                  0              0
998        999  3.17  3.116268                  3              1

[501 rows x 5 columns]

/var/folders/33/03_543s9719fh3y8mrxhst580000gn/T/ipykernel_46941/3552749785.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  engaged_students['new_pass_fail'] = engaged_students.apply(check_pass_fail, axis=1)
```

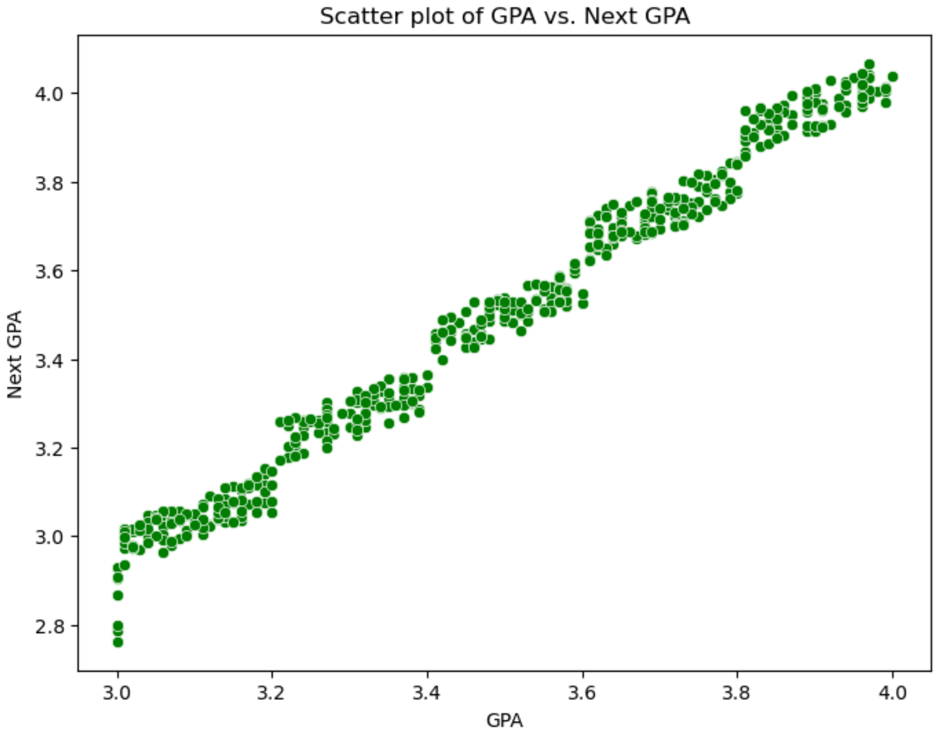
Visualizations

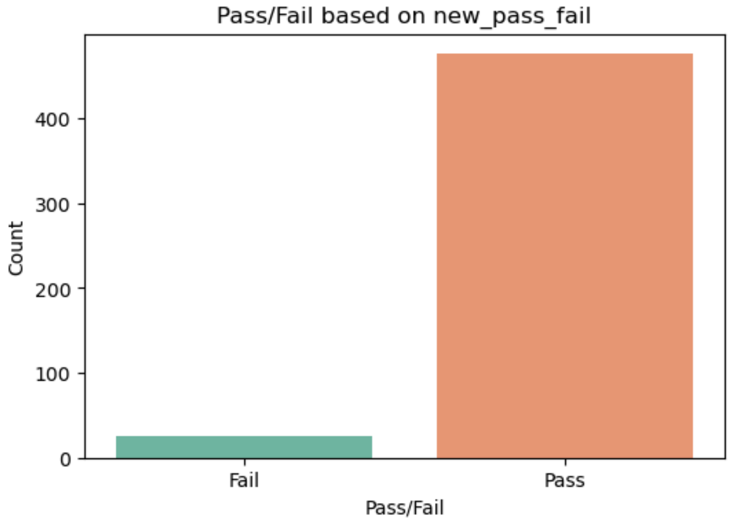
In [22]:

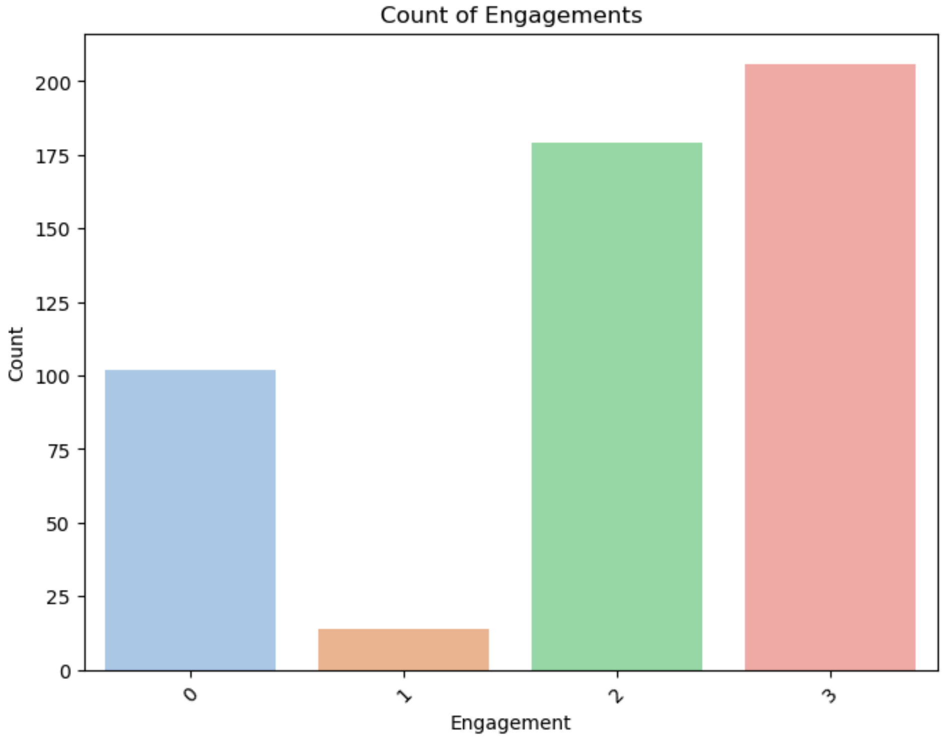
```
import matplotlib.pyplot as plt
import seaborn as sns

# Visualization 1: Distribution of GPA
plt.figure(figsize=(8, 6))
sns.histplot(data=engaged_students, x='GPA', bins=20, kde=True, color='skyblue')
plt.title('Distribution of GPA')
plt.xlabel('GPA')
plt.ylabel('Frequency')
plt.show()

# Visualization 2: Distribution of Next GPA
plt.figure(figsize=(8, 6))
sns.histplot(data=engaged_students, x='NextGPA', bins=20, kde=True, color='salmon')
plt.title('Distribution of Next GPA')
plt.xlabel('Next GPA')
plt.ylabel('Frequency')
plt.show()

# Visualization 3: Scatter plot of GPA vs. Next GPA
plt.figure(figsize=(8, 6))
sns.scatterplot(data=engaged_students, x='GPA', y='NextGPA', color='green')
plt.title('Scatter plot of GPA vs. Next GPA')
plt.xlabel('GPA')
plt.ylabel('Next GPA')
plt.show()

# Visualization 4: Bar plot of Pass/Fail based on new_pass_fail
plt.figure(figsize=(6, 4))
sns.countplot(data=engaged_students, x='new_pass_fail', palette='Set2')
plt.title('Pass/Fail based on new_pass_fail')
plt.xlabel('Pass/Fail')
plt.ylabel('Count')
plt.xticks(ticks=[0, 1], labels=['Fail', 'Pass'])
plt.show()

# Visualization 5: Count of Engagements
plt.figure(figsize=(8, 6))
sns.countplot(data=engaged_students, x='Engagement', palette='pastel')
plt.title('Count of Engagements')
plt.xlabel('Engagement')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```




Distribution of Next GPA
Frequency
0
10
20
30
40
50
2.8
3.0
3.2
3.4
3.6
3.8
4.0
Next GPA


Scatter plot of GPA vs. Next GPA
Next GPA
2.8
3.0
3.2
3.4
3.6
3.8
4.0
3.0
3.2
3.4
3.6
3.8
4.0
GPA


Pass/Fail based on new_pass_fail
Count
0
100
200
300
400
Fail
Pass
Pass/Fail


Count of Engagements
Count
0
25
50
75
100
125
150
175
200
0
1
2
3
Engagement