In [9]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [10]:
```python
# Student data
students = pd.DataFrame({
    'student_id': [1, 2, 3, 4],
    'name': ['Alice', 'Bob', 'Charlie', 'David'],
    'batch': ['2021', '2021', '2020', '2020'],
    'department': ['CSE', 'CSE', 'ECE', 'ECE']
})

# Performance data (GPA, core course scores)
performance_metrics = pd.DataFrame({
    'student_id': [1, 2, 3, 4],
    'semester_gpa': [8.5, 9.0, 7.5, 8.2],
    'core_courses_score': [85, 90, 75, 80]
})

# Extra-curricular data (hackathons, papers, assistance)
extra_curricular = pd.DataFrame({
    'student_id': [1, 2, 3, 4],
    'hackathons_participation': [3, 1, 2, 4],
    'papers_presented': [2, 3, 1, 0],
    'teacher_assistance': [1, 0, 2, 1]
})
```

```
In [11]: # Define weights for each metric
weights = {
    'semester_gpa': 0.5,
    'core_courses_score': 0.3,
    'hackathons_participation': 0.1,
    'papers_presented': 0.05,
    'teacher_assistance': 0.05

}

# Merge all data into a single DataFrame
data = pd.merge(students, performance_metrics, on='student_id')
data = pd.merge(data, extra_curricular, on='student_id')

# Calculate the weighted score for each student
data['total_score'] = (
    data['semester_gpa'] * weights['semester_gpa'] +
    data['core_courses_score'] * weights['core_courses_score'] +
    data['hackathons_participation'] * weights['hackathons_participation'] +
    data['papers_presented'] * weights['papers_presented'] +
    data['teacher_assistance'] * weights['teacher_assistance']
)

# Rank students based on total score
data['rank'] = data['total_score'].rank(ascending=False)

# Display top 3 students
top_students = data.sort_values(by='total_score', ascending=False).head(3)
print(top_students[['name', 'total_score', 'rank']])
```

```
    name  total_score  rank
1    Bob        31.75   1.0
0  Alice        30.20   2.0
3  David        28.55   3.0
```

In [12]:
```
# Historical data (target variable = overall success metric)
historical_data = pd.DataFrame({
    'semester_gpa': [8.5, 9.0, 7.5, 8.2],
    'core_courses_score': [85, 90, 75, 80],
    'hackathons_participation': [3, 1, 2, 4],
    'papers_presented': [2, 3, 1, 0],
    'teacher_assistance': [1, 0, 2, 1],
    'overall_success': [90, 95, 85, 87]
})

# Define features and target variable
X = historical_data[['semester_gpa', 'core_courses_score', 'hackathons_parti
y = historical_data['overall_success']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar

# Train a Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict success for new data
predictions = model.predict(X_test)

# Get the feature importance (coefficients)
weights = model.coef_
print("Dynamic Weights:", weights)
```

```
Dynamic Weights: [ 0.0265236   0.48827536 -0.11845199  0.21610706  0.00693
23 ]
```

In [13]:
```
# Use dynamic weights from the model
data['total_score'] = (
    data['semester_gpa'] * weights[0] +
    data['core_courses_score'] * weights[1] +
    data['hackathons_participation'] * weights[2] +
    data['papers_presented'] * weights[3] +
    data['teacher_assistance'] * weights[4]
)

# Rank students based on the dynamic total score
data['rank'] = data['total_score'].rank(ascending=False)
top_students = data.sort_values(by='total_score', ascending=False).head(3)
print(top_students[['name', 'total_score', 'rank']])
```

```
    name  total_score  rank
1    Bob    44.713364   1.0
0  Alice    41.812647   2.0
3  David    38.812647   3.0
```

In [ ]: