Dataset setup

In [2]:
```
import sqlite3
conn = sqlite3.connect('student_recognition.db')
cursor = conn.cursor()
cursor.execute('''
CREATE TABLE IF NOT EXISTS students (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    batch INTEGER NOT NULL
);
''')

cursor.execute('''
CREATE TABLE IF NOT EXISTS achievements (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    student_id INTEGER,
    academic_score FLOAT,
    hackathon_score FLOAT,
    paper_presentation_score FLOAT,
    contribution_score FLOAT,
    FOREIGN KEY(student_id) REFERENCES students(id)
);
''')

cursor.execute('''
CREATE TABLE IF NOT EXISTS weights (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    academic_weight FLOAT,
    hackathon_weight FLOAT,
    paper_weight FLOAT,
    contribution_weight FLOAT
);
''')

cursor.execute('''
INSERT INTO weights (academic_weight, hackathon_weight, paper_weight, contri
VALUES (0.5, 0.2, 0.2, 0.1);
''')

conn.commit()
print("Database and tables created successfully.")
```

```
Database and tables created successfully.
```

In [3]:
```
def add_student(name, batch):
    cursor.execute('''
    INSERT INTO students (name, batch) VALUES (?, ?)
    ''', (name, batch))
    conn.commit()
def add_achievement(student_id, academic_score, hackathon_score, paper_score
    cursor.execute('''
    INSERT INTO achievements (student_id, academic_score, hackathon_score, p
    VALUES (?, ?, ?, ?, ?)
    ''', (student_id, academic_score, hackathon_score, paper_score, contribu
    conn.commit()

# Example: Adding student and their achievements
add_student('Alice', 2022)
add_achievement(1, 85.5, 90.0, 80.0, 70.0)

add_student('Bob', 2022)
add_achievement(2, 78.0, 85.0, 88.0, 72.0)

add_student('Charlie', 2022)
add_achievement(3, 92.0, 87.0, 91.0, 80.0)

print("Students and achievements added successfully.")
```

```
Students and achievements added successfully.
```

In [5]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression

# Fetch student data and their achievements
def fetch_student_data():
    cursor.execute('''
    SELECT students.name, achievements.academic_score, achievements.hackatho
           achievements.paper_presentation_score, achievements.contribution_
    FROM students
    JOIN achievements ON students.id = achievements.student_id;
    ''')
    data = cursor.fetchall()
    return pd.DataFrame(data, columns=['Name', 'Academic', 'Hackathon', 'Pap

# Fetch weights for different criteria
def fetch_weights():
    cursor.execute('SELECT academic_weight, hackathon_weight, paper_weight,
    return cursor.fetchone()

# Calculate weighted scores and rankings using a machine learning model
def calculate_rankings():
    data = fetch_student_data()
    weights = fetch_weights()

    # Calculate weighted total for each student
    X = data[['Academic', 'Hackathon', 'Paper', 'Contribution']].values
    y = np.dot(X, weights)  # Apply weights

    # Add rankings to the dataframe
    data['Weighted_Score'] = y
    data['Rank'] = data['Weighted_Score'].rank(ascending=False)
    return data.sort_values('Rank')
```

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

# Mock function to fetch data
def fetch_student_data():
    # Example student data with 4 features and a weighted score as the targe
    data = {
        'Academic': [85, 90, 88, 78, 92],
        'Hackathon': [2, 3, 4, 1, 2],
        'Paper': [1, 2, 0, 1, 3],
        'Contribution': [10, 20, 5, 8, 25],
        'Weighted_Score': [80, 85, 78, 75, 90]
    }
    return pd.DataFrame(data)

def train_model():
    data = fetch_student_data()

    # Print the data for inspection
    print("Fetched student data:")
    print(data)

    # Features (academic, hackathon, paper, contribution)
    X = data[['Academic', 'Hackathon', 'Paper', 'Contribution']].values

    # Labels (can be past rankings if available)
    y = data['Weighted_Score']  # Using weighted scores as labels for simpli

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

    # Train the model
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Evaluate the model
    print(f"Model R² score: {model.score(X_test, y_test)}")

train_model()
```

```
Fetched student data:
   Academic  Hackathon  Paper  Contribution  Weighted_Score
0        85          2      1            10              80
1        90          3      2            20              85
2        88          4      0             5              78
3        78          1      1             8              75
4        92          2      3            25              90
Model R² score: nan

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_regression.py:
996: UndefinedMetricWarning: R^2 score is not well-defined with less than
two samples.
  warnings.warn(msg, UndefinedMetricWarning)
```

In [ ]: