Basic questions:

1. How many unique students are included in the dataset?
   1. Code:

uniquestudents = studentsdata['StudentID'].nunique()

print("Number of unique students:", uniquestudents)

* 1. Conclusion: We have [number of unique students] included in the dataset.

1. What is the average GPA of the students?
   1. Code: averagegpa = studentsdata['GPA'].mean()

print("Average GPA of the students:", average\_gpa)

* 1. Conclusion: The average GPA of the students is [average GPA].

1. What is the distribution of students across different graduation years?
   1. Code:

graduationyeardistribution = studentsdata['GraduationYear'].valuecounts()

print("Distribution of students across graduation years:\n", graduationyear\_distribution)

* 1. Conclusion: The distribution of students across different graduation years is as follows: [Graduation Year 1]: [Number of students] [Graduation Year 2]: [Number of students] …

1. What is the distribution of students' experience with Python programming?
   1. Code:

pythonexperiencedistribution = studentsdata['PythonExperience'].valuecounts()

print("Distribution of students' experience with Python programming:\n", pythonexperience\_distribution)

* 1. Conclusion: The distribution of students' experience with Python programming is as follows: [Experience Level 1]: [Number of students] [Experience Level 2]: [Number of students] …

1. What is the average family income of the students?
   1. Code: averagefamilyincome = studentsdata['FamilyIncome'].mean() print("Average family income of the students:", averagefamilyincome)
   2. Conclusion: The average family income of the students is [average income].
2. How does the GPA vary among different colleges?
   1. Code: gpabycollege = studentsdata.groupby('College')['GPA'].mean()

print("GPA variation among different colleges:\n", gpaby\_college)

* 1. Conclusion: The GPA varies among different colleges as shown in the output.

1. What is the GPA for students from each city?
   1. Code:

gpabycity = studentsdata.groupby('City')['GPA'].mean()

print("GPA for students from each city:\n", gpaby\_city);

* 1. Conclusion: The GPA for students from each city is provided in the output.

1. Can we identify any relationship between family income and GPA?
   1. Code: correlation = studentsdata['FamilyIncome'].corr(students\_data['GPA']) print("Correlation between family income and GPA:", correlation)
   2. Conclusion: There [is/is not] a significant relationship between family income and GPA.

Moderate questions:

1. How does the distribution of students vary across various cities?
   1. Code:

import matplotlib.pyplot as plt

citydistribution = studentsdata['City'].valuecounts() plt.bar(citydistribution.index, city\_distribution.values)

plt.xlabel('City') plt.ylabel('Number of Students')

plt.title('Distribution of Students Across Cities')

plt.xticks(rotation=45) plt.show()

* 1. Conclusion: The visualization illustrates the distribution of students across various cities, providing insights into the geographical representation of the student population.

1. How does the expected salary vary among factors like GPA?
   1. Code:

import seaborn as sns

sns.scatterplot(x='GPA', y='ExpectedSalary', data=studentsdata)

plt.xlabel('GPA') plt.ylabel('Expected Salary')

plt.title('Variation of Expected Salary with GPA')

plt.show()

* 1. Conclusion: The scatterplot depicts the relationship between GPA and expected salary, indicating whether there is any discernible trend or correlation between the two factors.

1. Relationship between family income, experience with Python, and expected salary
   1. Code: sns.pairplot(studentsdata, vars=['FamilyIncome', 'PythonExperience', 'ExpectedSalary'])

plt.title('Relationship between Family Income, Python Experience, and Expected Salary')

plt.show()

* 1. Conclusion: The pairplot visualizes the relationships between family income, experience with Python, and expected salary, providing insights into potential correlations or patterns among these variables

1. Which event tends to attract the most students from specific fields of study?
   1. Code: eventfieldcount = studentsdata.groupby('FieldofStudy')['Event'].valuecounts().unstack().idxmax()

print("Event attracting most students from each field of study:\n", eventfieldcount)

* 1. Conclusion: The output identifies the event that tends to attract the most students from specific fields of study, shedding light on the preferences and interests of students across different disciplines.

1. Do students in leadership positions during college years tend to have higher GPAs or better expected salaries?
   1. Code:

leadershipstudents = studentsdata[studentsdata['LeadershipPosition'] == True]

avggpaleadership = leadershipstudents['GPA'].mean()

avgsalaryleadership = leadershipstudents['ExpectedSalary'].mean()

print("Average GPA of students in leadership positions:", avggpaleadership)

print("Average expected salary of students in leadership positions:", avgsalary\_leadership)

* 1. Conclusion: The analysis compares the average GPA and expected salary of students in leadership positions, providing insights into the potential impact of leadership experience on academic and career success.

1. How many students are graduating by the end of 2024?
   1. Code:

studentsgraduating2024 = studentsdata[studentsdata['GraduationYear'] == 2024].shape[0] print("Number of students graduating by the end of 2024:", studentsgraduating\_2024)

* 1. Conclusion: The analysis determines the number of students expected to graduate by the end of 2024, providing insights into the cohort size for that year.

1. Which promotion channel brings in the greatest number of students for the event?
   1. Code:

promotionchannelcount = studentsdata['PromotionChannel'].valuecounts().idxmax() print("Promotion channel bringing the greatest number of students:"); promotionchannel\_count)

* 1. Conclusion: The output identifies the promotion channel that attracts the highest number of students for the event, highlighting the effectiveness of different marketing strategies.

1. Find the total number of students who attended events related to Data Science
   1. Code:

datascienceeventattendance = studentsdata[studentsdata['Event'].str.contains('Data Science')].shape[0]

print("Total number of students attending events related to Data Science:", datascienceeventattendance)

* 1. Conclusion: The analysis calculates the total number of students who attended events specifically related to Data Science, providing insights into the level of interest in this field among students.

1. Average expected salary for students with higher CGPA and more experience in Python who have high salary expectations
   1. Code:

highcriteriastudents = studentsdata[(studentsdata['CGPA'] > threshold) & (studentsdata['PythonExperience'] > threshold) & (studentsdata['ExpectedSalary'] > threshold)]

avgexpectedsalaryhighcriteria = highcriteriastudents['ExpectedSalary'].mean()

print("Average expected salary for students meeting the criteria:", avgexpectedsalaryhigh\_criteria)

* 1. Conclusion: The analysis calculates the average expected salary for students who meet specified criteria related to CGPA, experience with Python, and salary expectations, providing insights into the salary expectations of high-achieving students with strong technical skills.