

```
In [126...]: # DA Case Study using Pandas and Matplotlib  
# Dataset: class_marks.csv  
# Name: Subhapreet Patro  
# Roll No.: 2211CS010547  
# Group: 3
```

```
In [127...]: import pandas as pd  
import matplotlib.pyplot as plt  
  
data = pd.read_csv('class_marks.csv')  
  
print("Dataset Preview:")  
print(data.head())  
  
print("\nDataset Info:")  
print(data.info())
```

```
Dataset Preview:
```

```
Total  Q1aM4  Q1bM6  Q2aM6  Q2bM4  Q3aM5  Q3bM5  Q4aM3  Q4bM7  Q5M10  \
0      37      4.0      5.0      6.0      4.0      2.0      1.0      NaN      5.0      8.0
1      32      4.0      3.0      4.0      3.0      NaN      NaN      3.0      6.0      9.0
2      33      4.0      5.0      5.0      1.0      5.0      5.0      NaN      NaN      8.0
3      24      4.0      6.0      6.0      3.0      2.0      2.0      NaN      NaN      NaN
4      36      3.0      6.0      4.0      4.0      5.0      4.0      NaN      NaN      10.0
```

```
Q6aM4  Q6bM6
0      4.0      6.0
1      NaN      NaN
2      NaN      NaN
3      2.0      NaN
4      NaN      NaN
```

```
Dataset Info:
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 86 entries, 0 to 85
```

```
Data columns (total 12 columns):
```

#	Column	Non-Null Count	Dtype
0	Total	86 non-null	int64
1	Q1aM4	85 non-null	float64
2	Q1bM6	73 non-null	float64
3	Q2aM6	72 non-null	float64
4	Q2bM4	75 non-null	float64
5	Q3aM5	52 non-null	float64
6	Q3bM5	51 non-null	float64
7	Q4aM3	32 non-null	float64
8	Q4bM7	26 non-null	float64
9	Q5M10	56 non-null	float64
10	Q6aM4	33 non-null	float64
11	Q6bM6	28 non-null	float64

```
dtypes: float64(11), int64(1)
```

```
memory usage: 8.2 KB
```

```
None
```

This kernel imports necessary libraries, loads the dataset from a CSV file, and then displays a preview and summary information of the dataset.

In [128...]

```
print("Basic Statistical Overview:")
print(data.describe())
```

Basic Statistical Overview:

	Total	Q1aM4	Q1bM6	Q2aM6	Q2bM4	Q3aM5	\
count	86.000000	85.000000	73.000000	72.000000	75.00000	52.000000	
mean	29.360465	3.141176	4.972603	5.291667	3.20000	4.173077	
std	7.839437	1.013633	1.246914	1.155970	1.03975	1.294349	
min	3.000000	0.000000	2.000000	1.000000	1.00000	1.000000	
25%	25.000000	2.000000	4.000000	5.000000	2.00000	3.750000	
50%	31.000000	4.000000	5.000000	6.000000	4.00000	5.000000	
75%	36.000000	4.000000	6.000000	6.000000	4.00000	5.000000	
max	40.000000	4.000000	6.000000	6.000000	4.00000	5.000000	
	Q3bM5	Q4aM3	Q4bM7	Q5M10	Q6aM4	Q6bM6	
count	51.000000	32.000000	26.000000	56.000000	33.000000	28.000000	
mean	4.098039	2.468750	4.884615	8.000000	3.484848	4.964286	
std	1.345435	0.761339	2.196851	1.705606	0.972150	1.502643	
min	1.000000	1.000000	1.000000	4.000000	1.000000	1.000000	
25%	3.000000	2.000000	3.000000	7.000000	3.000000	4.000000	
50%	5.000000	3.000000	5.500000	8.000000	4.000000	6.000000	
75%	5.000000	3.000000	7.000000	10.000000	4.000000	6.000000	
max	5.000000	3.000000	7.000000	10.000000	4.000000	6.000000	

This kernel computes and displays basic statistical summary (e.g., mean, median, standard deviation) for the dataset's numerical columns.

In [129...]

```
average_score = data['Total'].mean()
max_score = data['Total'].max()
min_score = data['Total'].min()

print(f"Average Total Score: {average_score}")
print(f"Maximum Total Score: {max_score}")
print(f"Minimum Total Score: {min_score}")
```

Average Total Score: 29.36046511627907

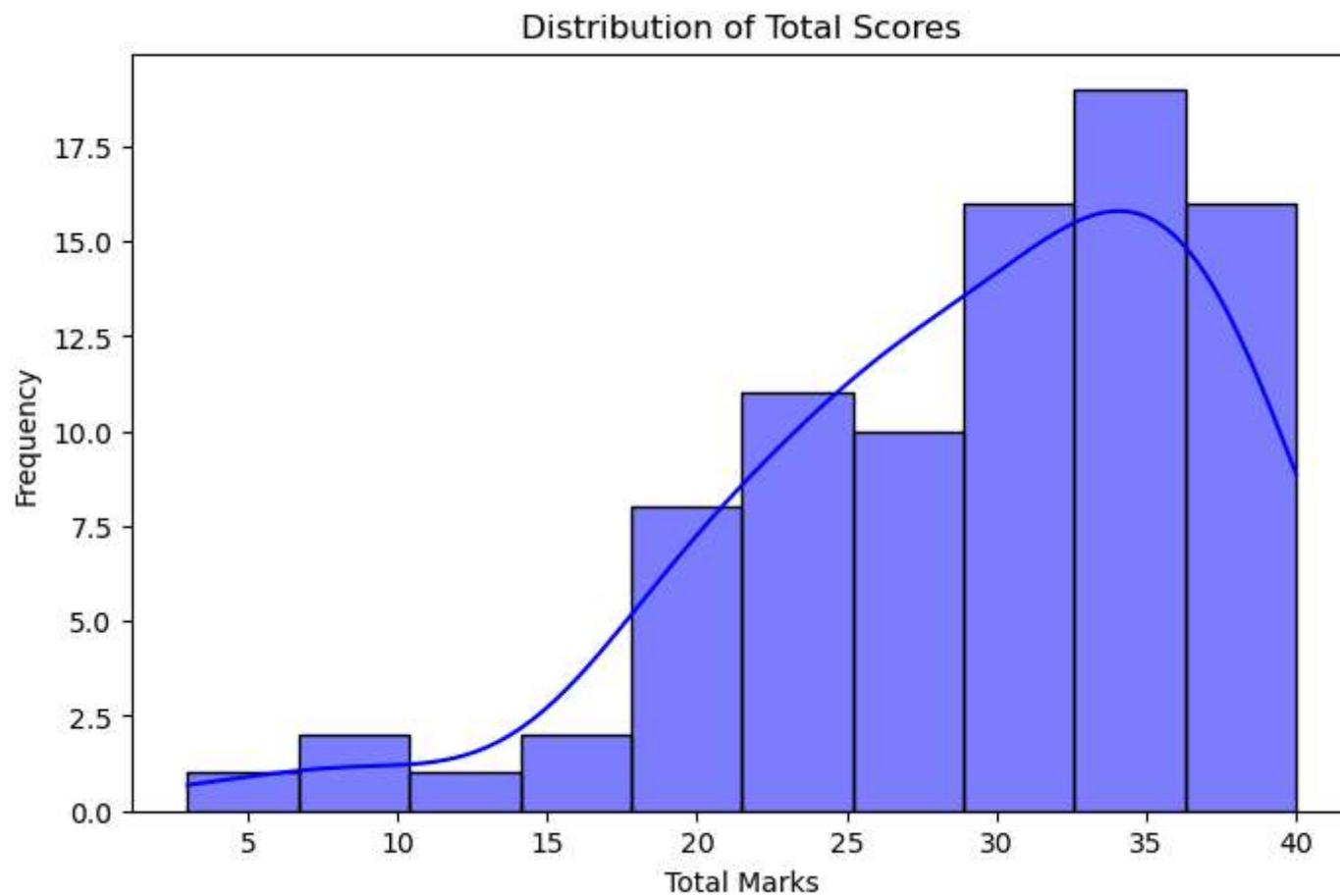
Maximum Total Score: 40

Minimum Total Score: 3

This kernel calculates and prints the average, maximum, and minimum values of the "Total" score column in the dataset.

In [130...]

```
plt.figure(figsize=(8, 5))
sns.histplot(data['Total'], kde=True, bins=10, color='blue')
plt.title('Distribution of Total Scores')
plt.xlabel('Total Marks')
plt.ylabel('Frequency')
plt.show()
```



This kernel visualizes the distribution of total scores using a histogram with KDE (Kernel Density Estimation).

In [131...]

```
threshold = 35
top_performers = data[data['Total'] > threshold]

print(f"Number of Top Performers (Total > {threshold}): {len(top_performers)}")
print(top_performers)
```

Number of Top Performers (Total > 35): 23

	Total	Q1aM4	Q1bM6	Q2aM6	Q2bM4	Q3aM5	Q3bM5	Q4aM3	Q4bM7	Q5M10	\
0	37	4.0	5.0	6.0	4.0	2.0	1.0	NaN	5.0	8.0	
4	36	3.0	6.0	4.0	4.0	5.0	4.0	NaN	NaN	10.0	
10	37	3.0	5.0	6.0	4.0	NaN	NaN	3.0	6.0	NaN	
17	36	3.0	4.0	6.0	4.0	NaN	NaN	NaN	NaN	9.0	
26	39	4.0	6.0	6.0	3.0	4.0	NaN	NaN	NaN	NaN	
28	38	2.0	6.0	6.0	4.0	5.0	5.0	NaN	NaN	10.0	
33	40	NaN	NaN	6.0	4.0	5.0	5.0	3.0	7.0	NaN	
36	37	2.0	NaN	6.0	4.0	5.0	5.0	NaN	NaN	10.0	
42	38	4.0	6.0	6.0	4.0	5.0	5.0	3.0	5.0	NaN	
44	36	3.0	6.0	6.0	2.0	NaN	NaN	2.0	7.0	10.0	
46	36	4.0	5.0	6.0	4.0	5.0	5.0	NaN	NaN	7.0	
47	38	2.0	6.0	6.0	4.0	5.0	5.0	3.0	7.0	10.0	
49	39	3.0	6.0	6.0	4.0	5.0	5.0	NaN	NaN	10.0	
51	40	0.0	NaN	6.0	4.0	NaN	NaN	3.0	7.0	10.0	
53	40	4.0	6.0	6.0	4.0	5.0	5.0	NaN	NaN	10.0	
55	38	3.0	5.0	6.0	4.0	NaN	NaN	NaN	NaN	10.0	
59	38	2.0	6.0	6.0	4.0	5.0	5.0	NaN	NaN	10.0	
62	36	3.0	4.0	6.0	4.0	5.0	5.0	NaN	NaN	7.0	
64	36	1.0	6.0	6.0	4.0	5.0	5.0	NaN	NaN	9.0	
65	40	4.0	6.0	6.0	4.0	5.0	5.0	NaN	NaN	10.0	
71	36	4.0	5.0	6.0	4.0	5.0	5.0	NaN	NaN	7.0	
73	40	4.0	6.0	NaN	NaN	5.0	5.0	3.0	NaN	10.0	
83	37	4.0	6.0	6.0	2.0	NaN	NaN	NaN	NaN	9.0	

Q6aM4 Q6bM6

0	4.0	6.0
4	NaN	NaN
10	4.0	6.0
17	4.0	6.0
26	4.0	6.0
28	NaN	NaN
33	4.0	6.0
36	4.0	5.0
42	NaN	NaN
44	NaN	NaN
46	NaN	NaN
47	NaN	NaN
49	NaN	NaN
51	NaN	NaN
53	NaN	NaN
55	4.0	6.0
59	4.0	6.0
62	NaN	NaN

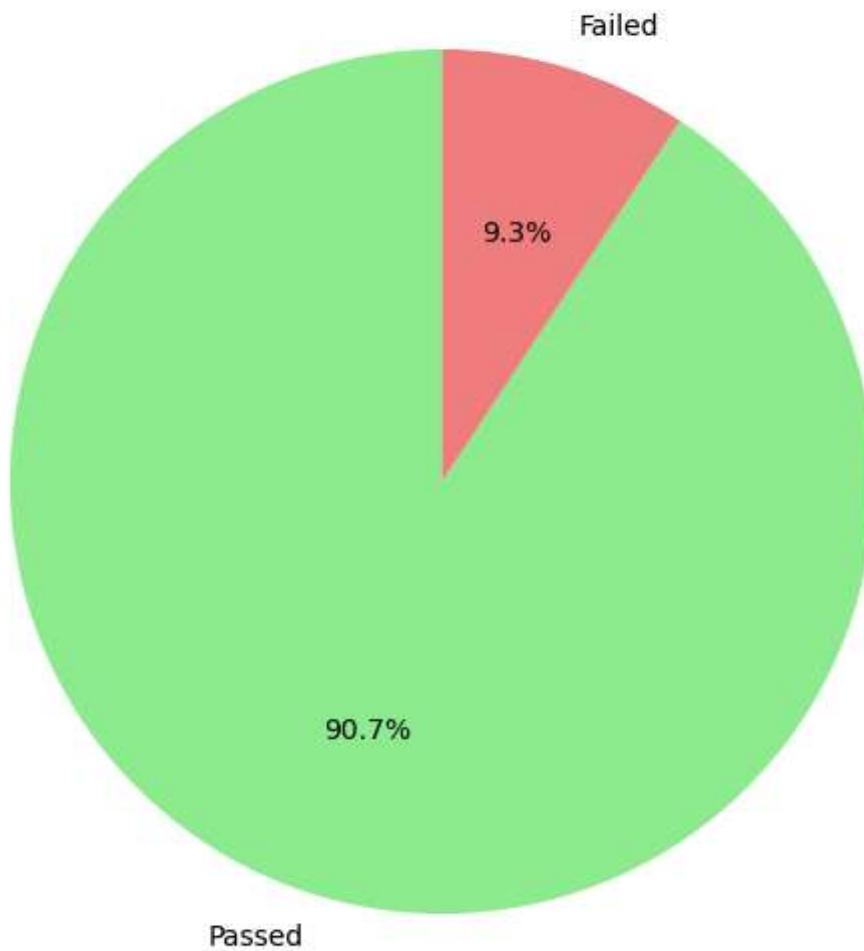
```
64    NaN    NaN
65    NaN    NaN
71    NaN    NaN
73    4.0    6.0
83    4.0    6.0
```

This kernel identifies and displays the students whose total scores are above a specified threshold (35), and prints the number of top performers.

```
In [132...]  
pass_threshold = 20  
passed_students = data[data['Total'] >= pass_threshold]  
pass_percentage = (len(passed_students) / len(data)) * 100  
  
print(f"Pass Percentage: {pass_percentage:.2f}%")  
  
data['Passed'] = data['Total'] >= 20  
  
pass_fail_counts = data['Passed'].value_counts()  
labels = ['Passed', 'Failed']  
colors = ['lightgreen', 'lightcoral']  
  
plt.figure(figsize=(7, 7))  
plt.pie(pass_fail_counts, labels=labels, autopct='%1.1f%%', colors=colors, startangle=90)  
plt.title('Class Performance: Passed vs Failed')  
plt.show()
```

Pass Percentage: 90.70%

Class Performance: Passed vs Failed



This kernel calculates the pass percentage based on a threshold and creates a new "Passed" column, followed by visualizing the pass/fail distribution using a pie chart.

In [133...]

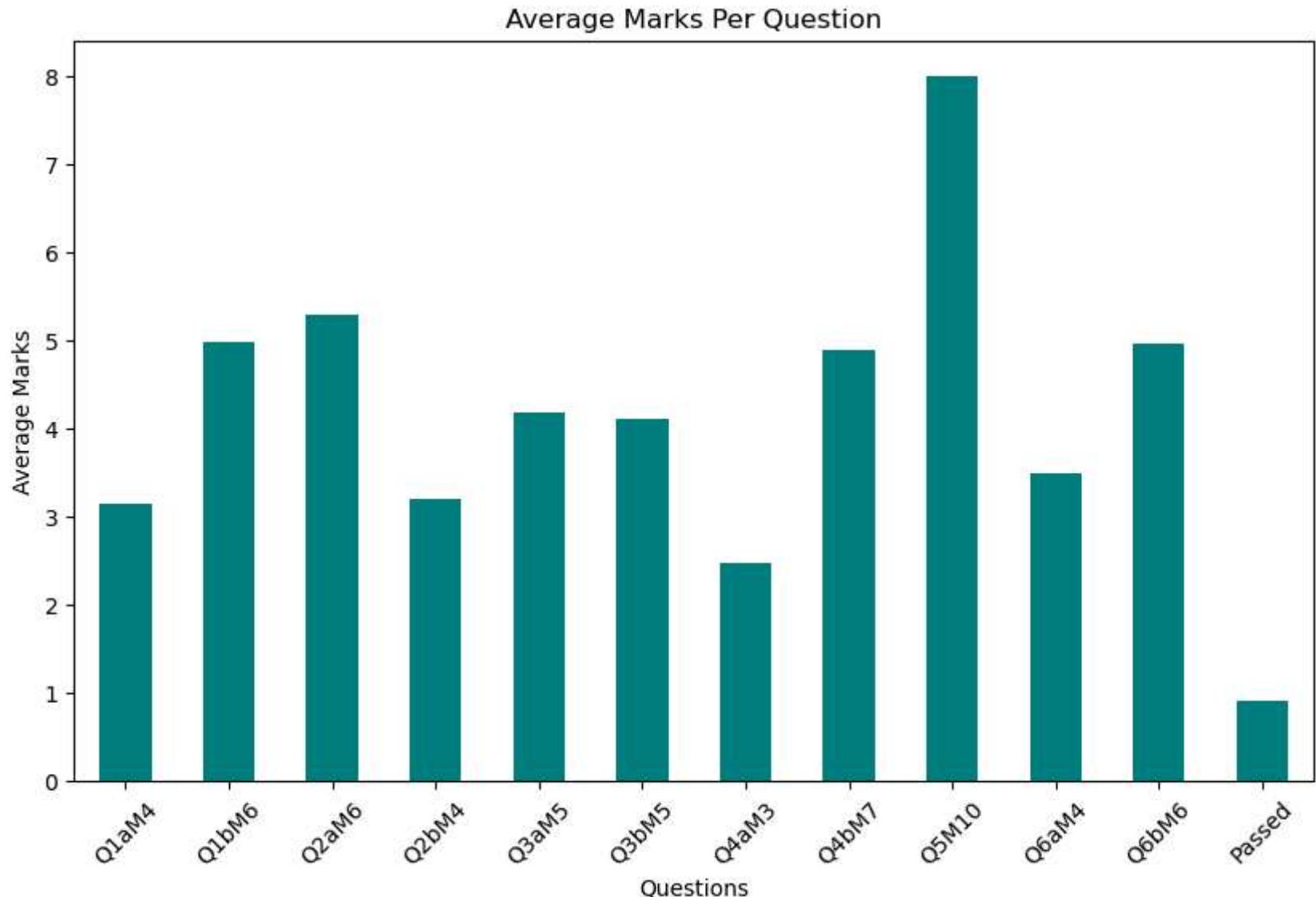
```
question_columns = [col for col in data.columns if col != 'Total']
average_marks_per_question = data[question_columns].mean()

print("Average Marks Per Question:")
print(average_marks_per_question)

plt.figure(figsize=(10, 6))
average_marks_per_question.plot(kind='bar', color='teal')
plt.title('Average Marks Per Question')
plt.xlabel('Questions')
plt.ylabel('Average Marks')
plt.xticks(rotation=45)
plt.show()
```

Average Marks Per Question:

```
Q1aM4      3.141176
Q1bM6      4.972603
Q2aM6      5.291667
Q2bM4      3.200000
Q3aM5      4.173077
Q3bM5      4.098039
Q4aM3      2.468750
Q4bM7      4.884615
Q5M10      8.000000
Q6aM4      3.484848
Q6bM6      4.964286
Passed      0.906977
dtype: float64
```



This kernel calculates and displays the average marks for each question, then visualizes the averages using a bar chart.

```
In [134]: challenging_questions = average_marks_per_question[average_marks_per_question < average_marks_per_question.mean()]
print("Challenging Questions (Below Average):")
```

```
print(challenging_questions)
```

```
Challenging Questions (Below Average):  
Q1aM4      3.141176  
Q2bM4      3.200000  
Q3bM5      4.098039  
Q4aM3      2.468750  
Q6aM4      3.484848  
Passed     0.906977  
dtype: float64
```

This kernel identifies and displays the questions with average scores below the overall average, indicating the most challenging questions.

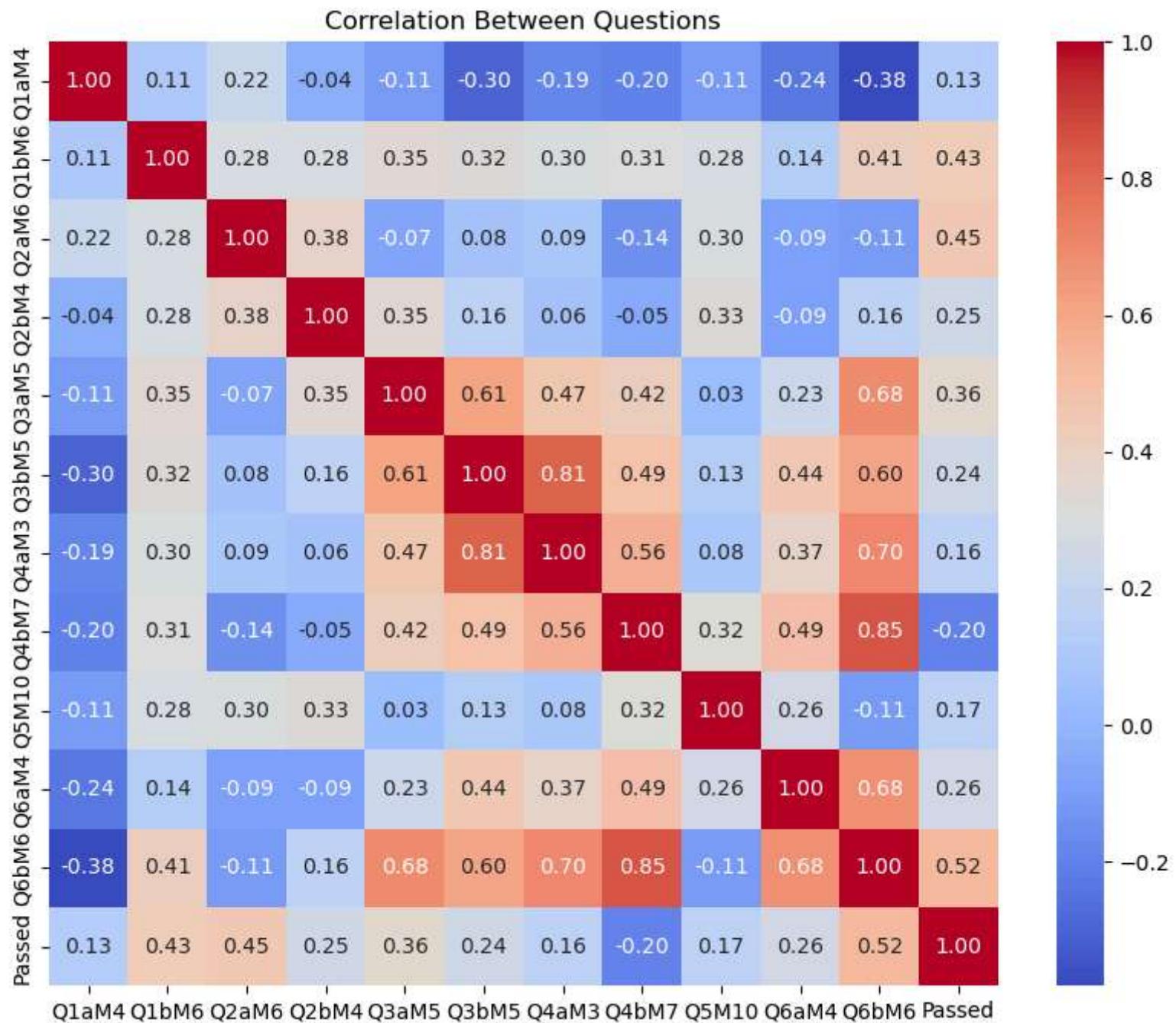
In [135...]

```
correlation_matrix = data[question_columns].corr()  
  
print("Correlation Matrix:")  
print(correlation_matrix)  
  
plt.figure(figsize=(10, 8))  
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")  
plt.title('Correlation Between Questions')  
plt.show()
```

Correlation Matrix:

	Q1aM4	Q1bM6	Q2aM6	Q2bM4	Q3aM5	Q3bM5	Q4aM3	\
Q1aM4	1.000000	0.109738	0.224234	-0.040381	-0.109641	-0.300419	-0.189504	
Q1bM6	0.109738	1.000000	0.282313	0.284455	0.351443	0.321828	0.296118	
Q2aM6	0.224234	0.282313	1.000000	0.375111	-0.069269	0.075433	0.091992	
Q2bM4	-0.040381	0.284455	0.375111	1.000000	0.352150	0.162289	0.064253	
Q3aM5	-0.109641	0.351443	-0.069269	0.352150	1.000000	0.613131	0.465130	
Q3bM5	-0.300419	0.321828	0.075433	0.162289	0.613131	1.000000	0.814191	
Q4aM3	-0.189504	0.296118	0.091992	0.064253	0.465130	0.814191	1.000000	
Q4bM7	-0.203419	0.310639	-0.144171	-0.048879	0.418330	0.486056	0.560380	
Q5M10	-0.110698	0.284444	0.299839	0.330075	0.034933	0.132515	0.082993	
Q6aM4	-0.236846	0.135749	-0.089972	-0.094541	0.231930	0.443978	0.374281	
Q6bM6	-0.381962	0.406479	-0.112426	0.158579	0.684208	0.603307	0.701932	
Passed	0.125130	0.431945	0.450176	0.247451	0.355128	0.242349	0.161515	

	Q4bM7	Q5M10	Q6aM4	Q6bM6	Passed
Q1aM4	-0.203419	-0.110698	-0.236846	-0.381962	0.125130
Q1bM6	0.310639	0.284444	0.135749	0.406479	0.431945
Q2aM6	-0.144171	0.299839	-0.089972	-0.112426	0.450176
Q2bM4	-0.048879	0.330075	-0.094541	0.158579	0.247451
Q3aM5	0.418330	0.034933	0.231930	0.684208	0.355128
Q3bM5	0.486056	0.132515	0.443978	0.603307	0.242349
Q4aM3	0.560380	0.082993	0.374281	0.701932	0.161515
Q4bM7	1.000000	0.317862	0.490960	0.847159	-0.196397
Q5M10	0.317862	1.000000	0.264512	-0.114708	0.170783
Q6aM4	0.490960	0.264512	1.000000	0.681095	0.261308
Q6bM6	0.847159	-0.114708	0.681095	1.000000	0.517040
Passed	-0.196397	0.170783	0.261308	0.517040	1.000000



This kernel computes and displays the correlation matrix between questions, followed by visualizing it using a heatmap to show relationships between the questions.

In [136]:

```
top_5_students = data.nlargest(5, 'Total')
print("Top 5 Students:")
print(top_5_students)

bottom_5_students = data.nsmallest(5, 'Total')
print("\nBottom 5 Students:")
print(bottom_5_students)
```

Top 5 Students:

	Total	Q1aM4	Q1bM6	Q2aM6	Q2bM4	Q3aM5	Q3bM5	Q4aM3	Q4bM7	Q5M10	\
33	40	NaN	NaN	6.0	4.0	5.0	5.0	3.0	7.0	NaN	
51	40	0.0	NaN	6.0	4.0	NaN	NaN	3.0	7.0	10.0	
53	40	4.0	6.0	6.0	4.0	5.0	5.0	NaN	NaN	10.0	
65	40	4.0	6.0	6.0	4.0	5.0	5.0	NaN	NaN	10.0	
73	40	4.0	6.0	NaN	NaN	5.0	5.0	3.0	NaN	10.0	

Q6aM4 Q6bM6 Passed

33	4.0	6.0	True
51	NaN	NaN	True
53	NaN	NaN	True
65	NaN	NaN	True
73	4.0	6.0	True

Bottom 5 Students:

	Total	Q1aM4	Q1bM6	Q2aM6	Q2bM4	Q3aM5	Q3bM5	Q4aM3	Q4bM7	Q5M10	\
69	3	1.0	NaN	1.0	NaN	NaN	NaN	1.0	NaN	NaN	
11	8	2.0	2.0	NaN	3.0	1.0	NaN	NaN	NaN	NaN	
23	9	4.0	3.0	NaN							
22	14	4.0	4.0	5.0	2.0	NaN	NaN	NaN	NaN	NaN	
57	17	3.0	NaN	NaN	4.0	NaN	NaN	3.0	7.0	NaN	

Q6aM4 Q6bM6 Passed

69	NaN	NaN	False
11	NaN	NaN	False
23	1.0	1.0	False
22	NaN	NaN	False
57	4.0	NaN	False

This kernel displays the top 5 and bottom 5 students based on their total marks in the dataset.

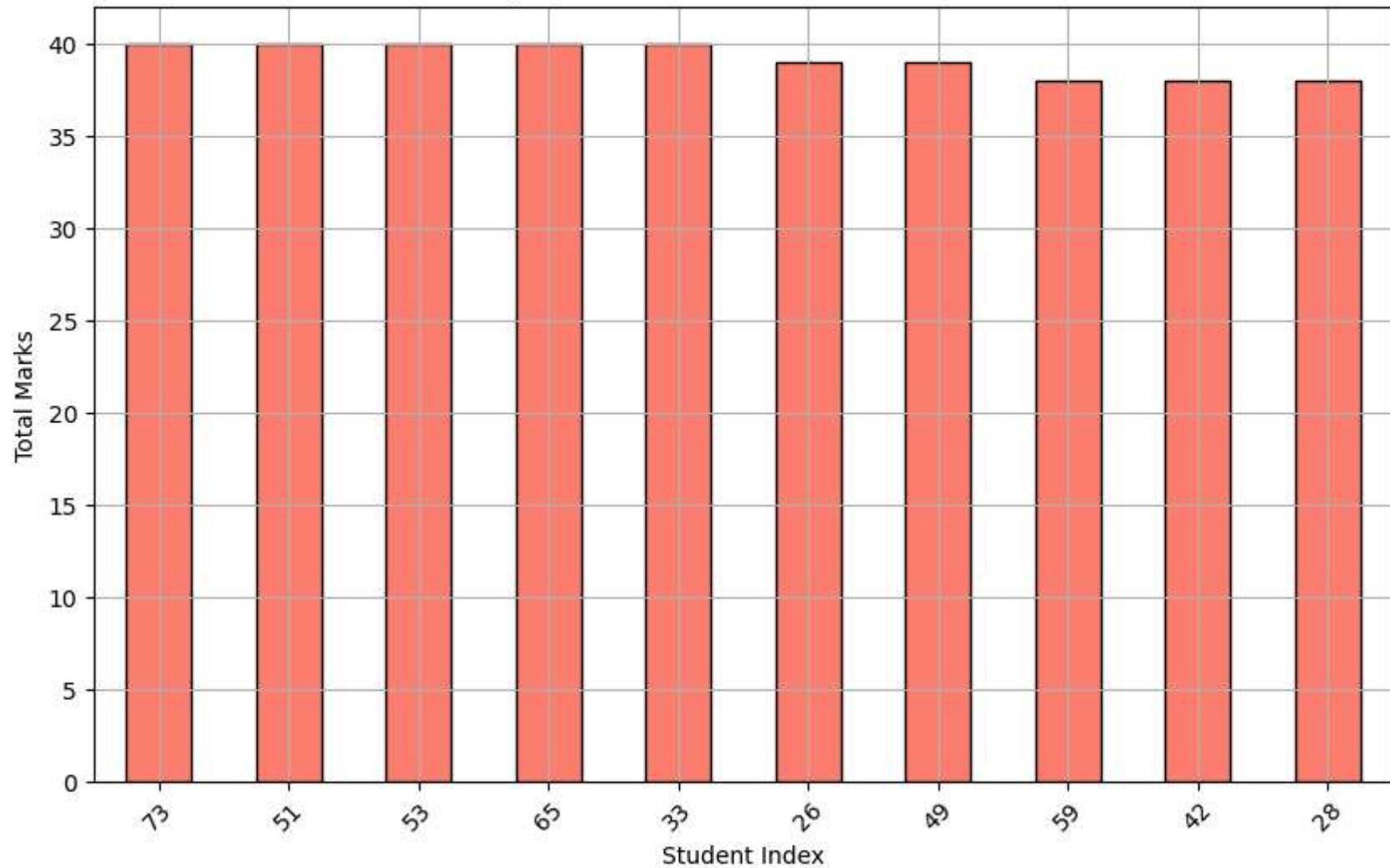
In [137...]

```
top_students = data['Total'].sort_values(ascending=False).head(10)

print(top_students)
plt.figure(figsize=(10, 6))
top_students.plot(kind='bar', color='salmon', edgecolor='black')
plt.title('Top 10 Students Based on Total Marks')
plt.xlabel('Student Index')
plt.ylabel('Total Marks')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

```
73    40
51    40
53    40
65    40
33    40
26    39
49    39
59    38
42    38
28    38
Name: Total, dtype: int64
```

Top 10 Students Based on Total Marks



This kernel displays the top 10 students based on their total marks and visualizes the results using a bar chart.

In [138]:

```
median_score = data['Total'].median()
high_performers = data[data['Total'] > median_score]
low_performers = data[data['Total'] <= median_score]
```

```
high_avg = high_performers[question_columns].mean()
low_avg = low_performers[question_columns].mean()

print("Average Marks of High Performers:")
print(high_avg)
print("\nAverage Marks of Low Performers:")
print(low_avg)

plt.figure(figsize=(12, 6))
plt.bar(high_avg.index, high_avg.values, alpha=0.7, label='High Performers', color='green')
plt.bar(low_avg.index, low_avg.values, alpha=0.7, label='Low Performers', color='red')
plt.title('Comparison of Average Marks: High vs Low Performers')
plt.xlabel('Questions')
plt.ylabel('Average Marks')
plt.legend()
plt.xticks(rotation=45)
plt.show()
```

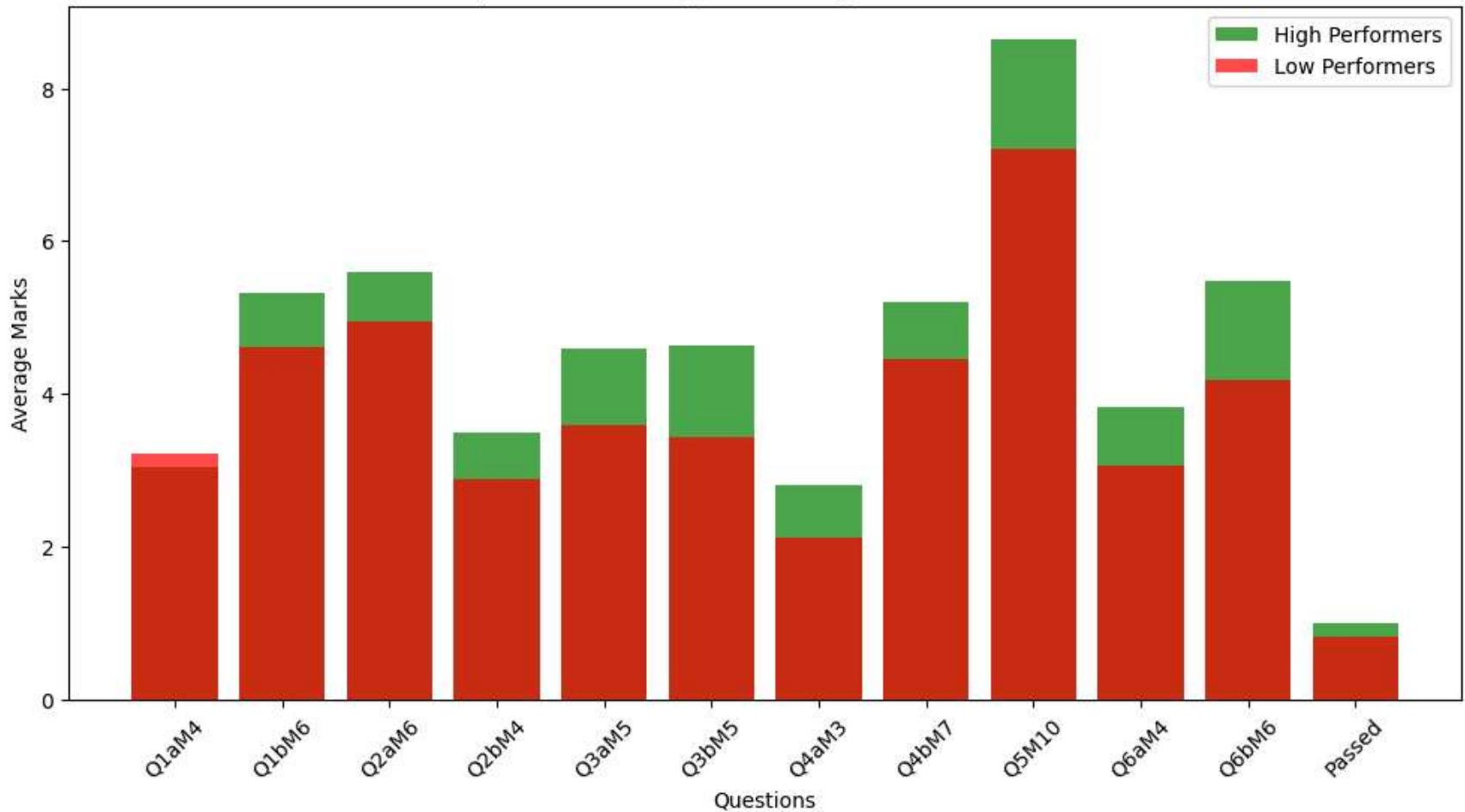
Average Marks of High Performers:

```
Q1aM4      3.050000
Q1bM6      5.324324
Q2aM6      5.605263
Q2bM4      3.500000
Q3aM5      4.600000
Q3bM5      4.642857
Q4aM3      2.812500
Q4bM7      5.200000
Q5M10      8.645161
Q6aM4      3.833333
Q6bM6      5.470588
Passed     1.000000
dtype: float64
```

Average Marks of Low Performers:

```
Q1aM4      3.222222
Q1bM6      4.611111
Q2aM6      4.941176
Q2bM4      2.891892
Q3aM5      3.590909
Q3bM5      3.434783
Q4aM3      2.125000
Q4bM7      4.454545
Q5M10      7.200000
Q6aM4      3.066667
Q6bM6      4.181818
Passed     0.822222
dtype: float64
```

Comparison of Average Marks: High vs Low Performers



This kernel divides students into high and low performers based on the median total marks, calculates the average marks for each group, and visualizes the comparison using a bar chart.

In [139...]

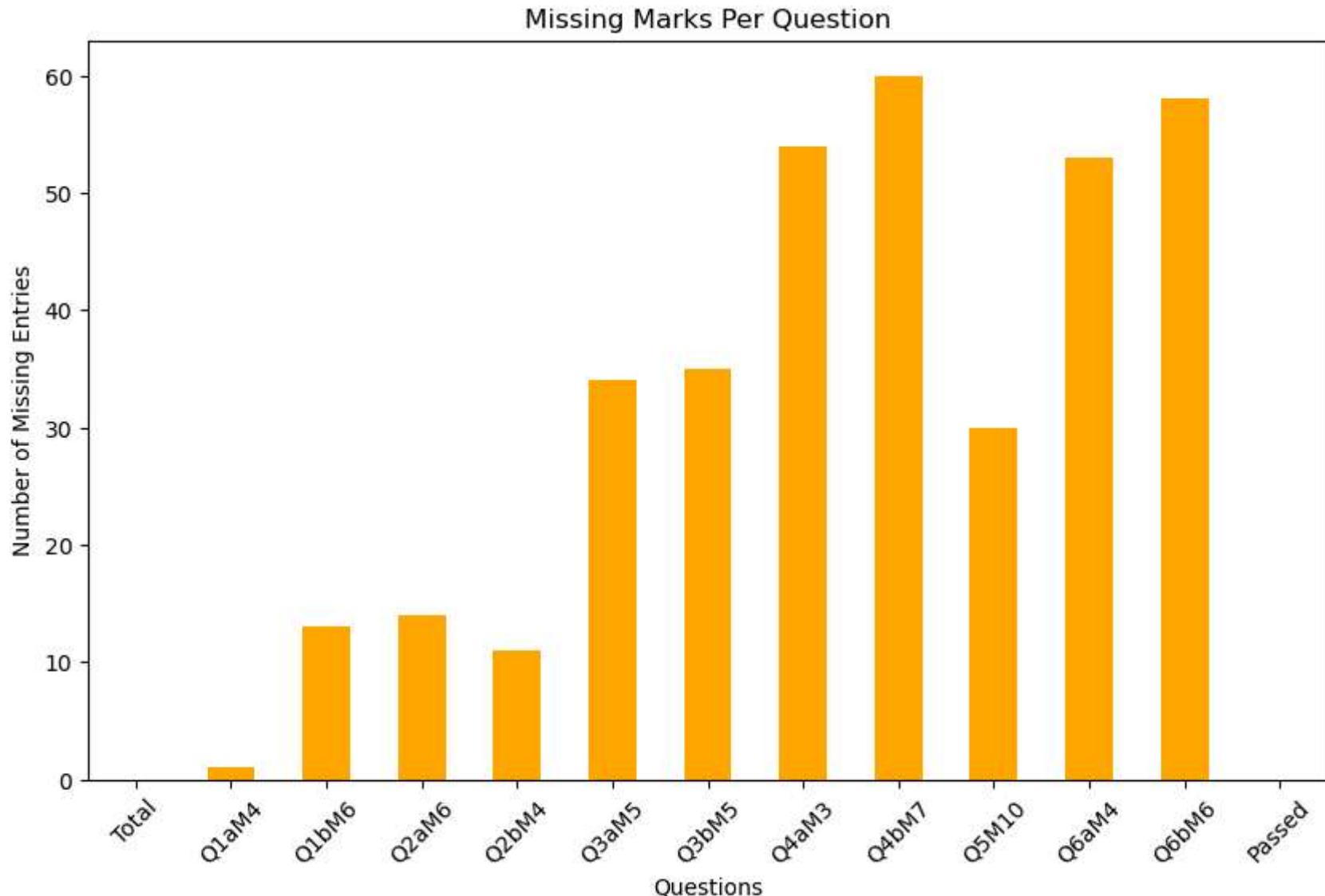
```
missing_marks = data.isnull().sum()  
print("Missing Marks Per Question:")
```

```
print(missing_marks)

plt.figure(figsize=(10, 6))
missing_marks.plot(kind='bar', color='orange')
plt.title('Missing Marks Per Question')
plt.xlabel('Questions')
plt.ylabel('Number of Missing Entries')
plt.xticks(rotation=45)
plt.show()
```

Missing Marks Per Question:

```
Total      0
Q1aM4     1
Q1bM6    13
Q2aM6    14
Q2bM4    11
Q3aM5    34
Q3bM5    35
Q4aM3    54
Q4bM7    60
Q5M10    30
Q6aM4    53
Q6bM6    58
Passed     0
dtype: int64
```



This kernel identifies and displays the number of missing marks for each question, then visualizes the missing data using a bar chart.

```
In [140...]: max_marks_per_question = {
    'Q1aM4': 4,
    'Q1bM6': 6,
    'Q2aM6': 6,
    'Q2bM4': 4,
    'Q3aM5': 5,
    'Q3bM5': 5,
    'Q4aM3': 3,
    'Q4bM7': 7,
    'Q5M10': 10,
    'Q6aM4': 4,
    'Q6bM6': 6
}

difficulty_threshold = 0.7
difficult_questions = []

for question, max_marks in max_marks_per_question.items():
    avg_marks = data[question].mean()
    if avg_marks / max_marks < difficulty_threshold:
        difficult_questions.append((question, avg_marks, max_marks))

print("Difficult Questions (Average Marks < 70% of Max):")
for q in difficult_questions:
    print(f"Question: {q[0]}, Average Marks: {q[1]:.2f}, Max Marks: {q[2]}")
```

Difficult Questions (Average Marks < 70% of Max):
 Question: Q4bM7, Average Marks: 4.88, Max Marks: 7

This kernel identifies and prints the questions where the average marks are less than 70% of the maximum possible marks, labeling them as "difficult" questions.

```
In [141...]: missed_counts = data[question_columns].isnull().sum()

most_missed = missed_counts.sort_values(ascending=False)
print("Most Commonly Missed Questions:")
print(most_missed)

plt.figure(figsize=(10, 6))
most_missed.plot(kind='bar', color='purple')
```

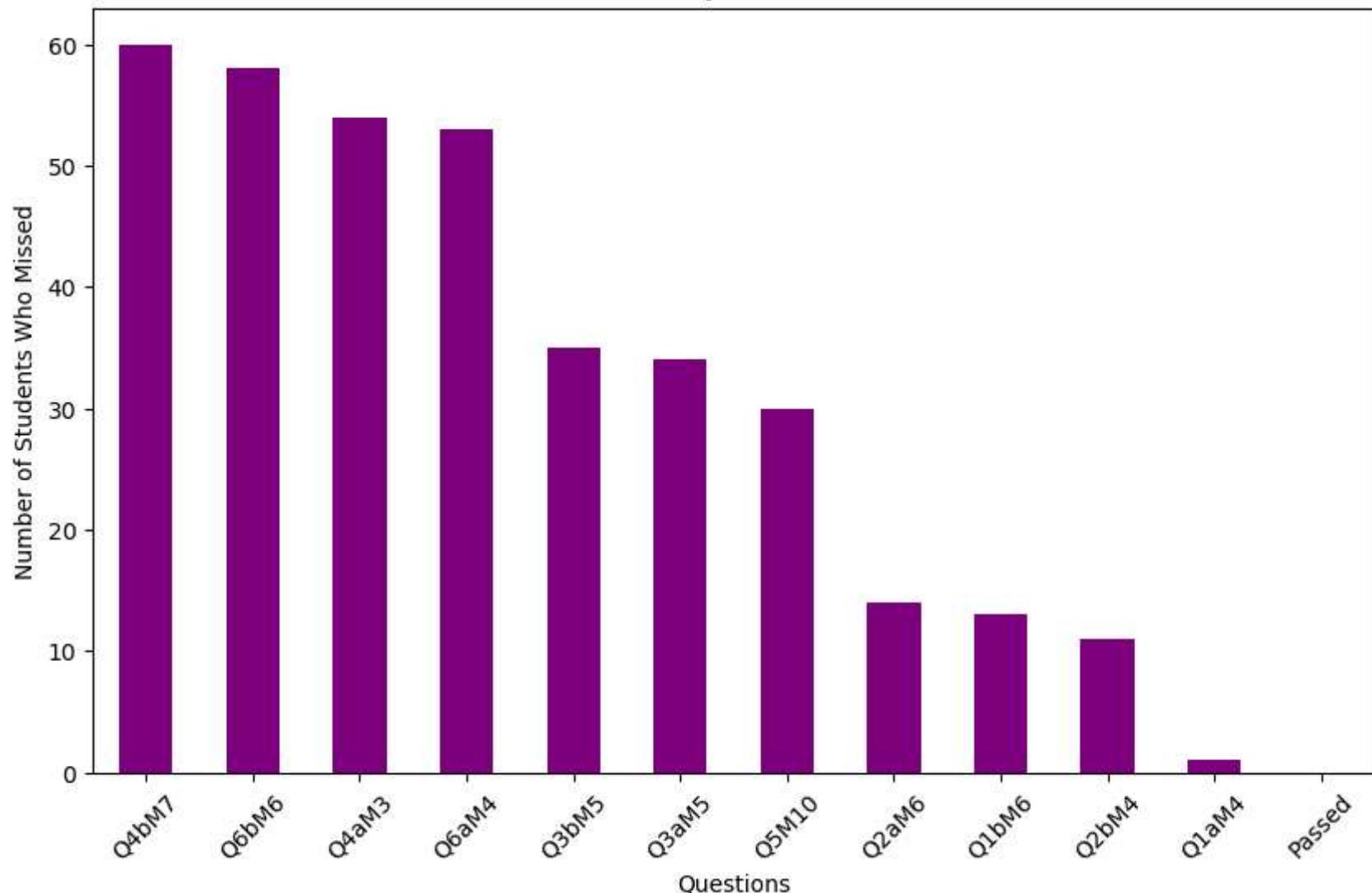
```
plt.title('Most Commonly Missed Questions')
plt.xlabel('Questions')
plt.ylabel('Number of Students Who Missed')
plt.xticks(rotation=45)
plt.show()
```

Most Commonly Missed Questions:

Q4bM7	60
Q6bM6	58
Q4aM3	54
Q6aM4	53
Q3bM5	35
Q3aM5	34
Q5M10	30
Q2aM6	14
Q1bM6	13
Q2bM4	11
Q1aM4	1
Passed	0

dtype: int64

Most Commonly Missed Questions



This kernel counts and displays the number of students who missed each question, then visualizes the most commonly missed questions using a bar chart.

In [142...]

```
bins = [0, 10, 20, 30, 40]
labels = ['0-10', '11-20', '21-30', '31-40']

data['Score_Range'] = pd.cut(data['Total'], bins=bins, labels=labels, include_lowest=True)

score_range_counts = data['Score_Range'].value_counts()

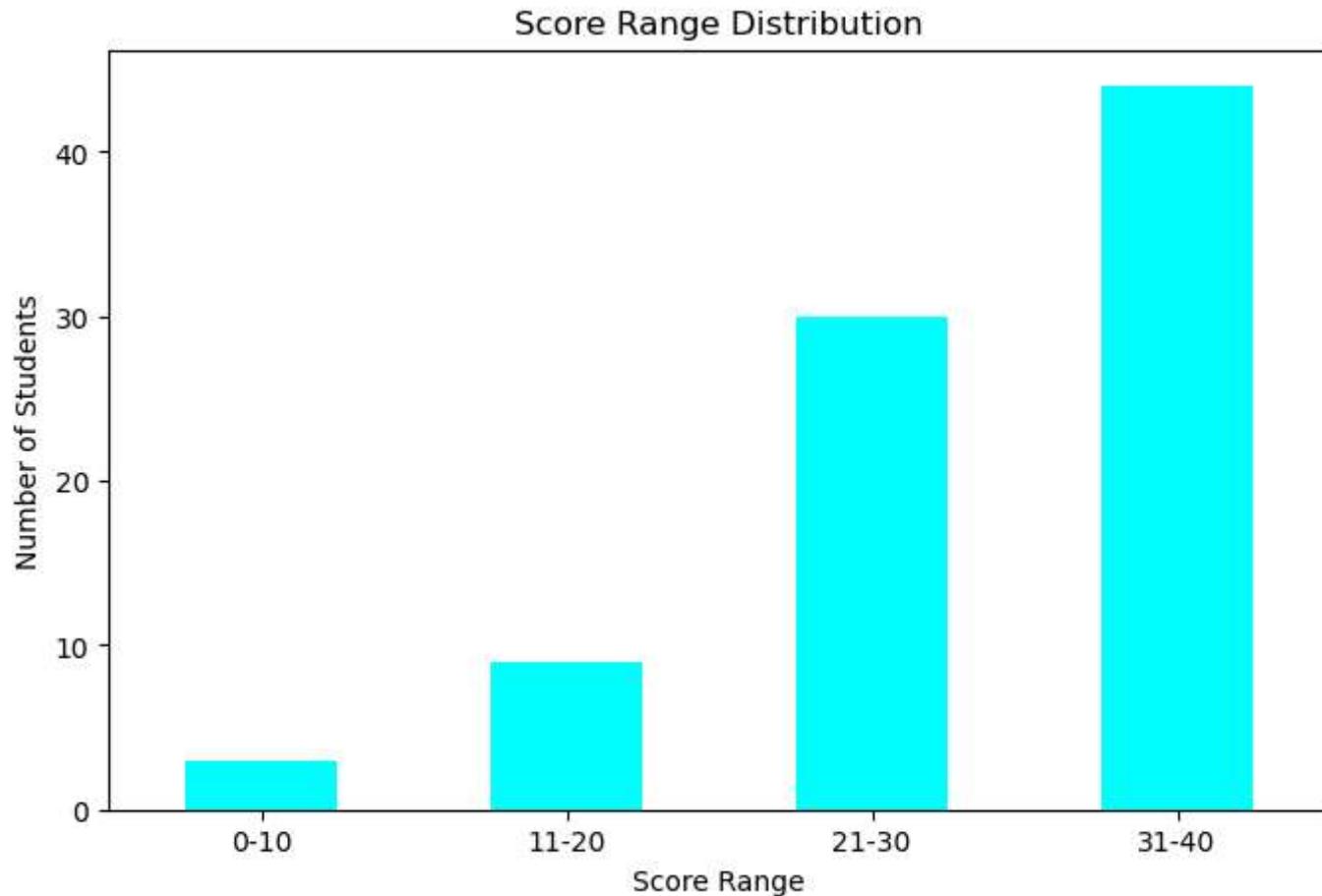
print("Number of Students in Each Score Range:")
print(score_range_counts)

plt.figure(figsize=(8, 5))
score_range_counts.sort_index().plot(kind='bar', color='cyan')
plt.title('Score Range Distribution')
plt.xlabel('Score Range')
plt.ylabel('Number of Students')
plt.xticks(rotation=0)
plt.show()
```

Number of Students in Each Score Range:

```
31-40    44
21-30    30
11-20     9
0-10      3
```

Name: Score_Range, dtype: int64



This kernel defines score ranges for total marks, assigns students to these ranges, and visualizes the distribution of students across the defined score ranges using a bar chart.

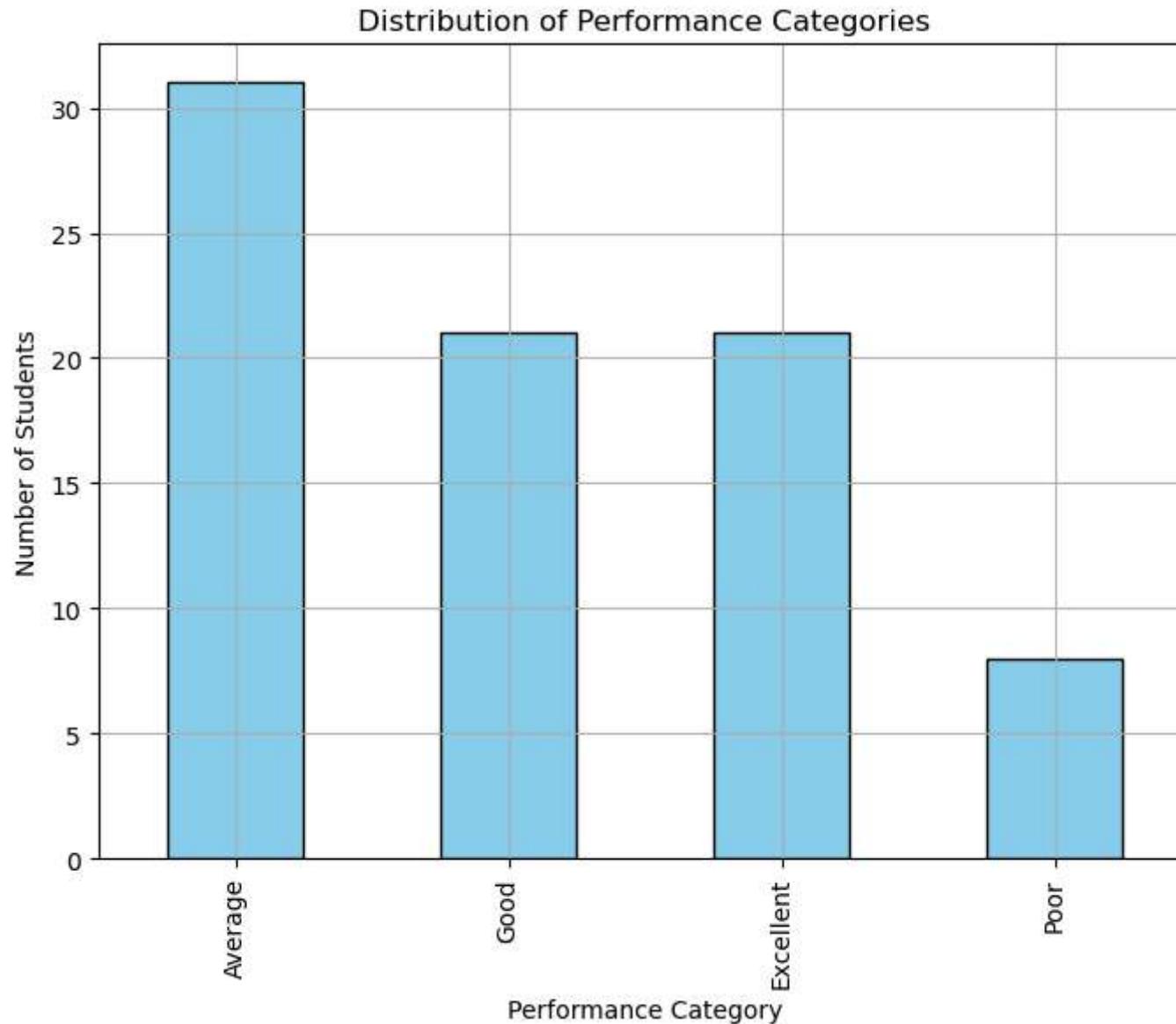
In [143]:

```
bins = [0, 20, 30, 35, 40]
labels = ['Poor', 'Average', 'Good', 'Excellent']
data['Performance_Category'] = pd.cut(data['Total'], bins=bins, labels=labels, right=False)

data[['Total', 'Performance_Category']].head()

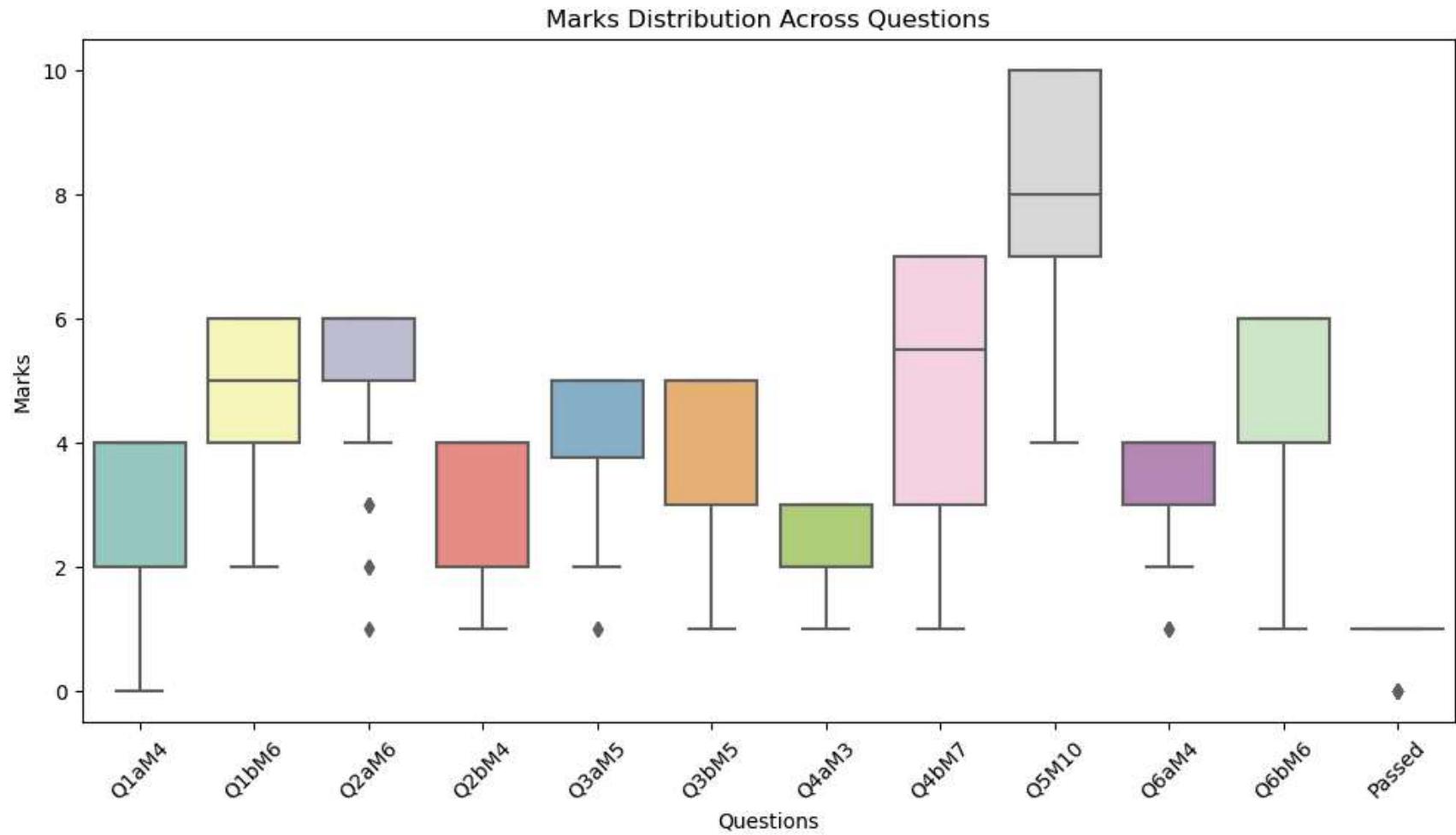
plt.figure(figsize=(8, 6))
```

```
data['Performance_Category'].value_counts().plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('Distribution of Performance Categories')
plt.xlabel('Performance Category')
plt.ylabel('Number of Students')
plt.grid(True)
plt.show()
```



This kernel categorizes students into performance categories based on their total marks, displays the first few rows with these categories, and visualizes the distribution of students across the categories using a bar chart.

```
In [144...]  
plt.figure(figsize=(12, 6))  
sns.boxplot(data=data[question_columns], palette='Set3')  
plt.title('Marks Distribution Across Questions')  
plt.xlabel('Questions')  
plt.ylabel('Marks')  
plt.xticks(rotation=45)  
plt.show()
```

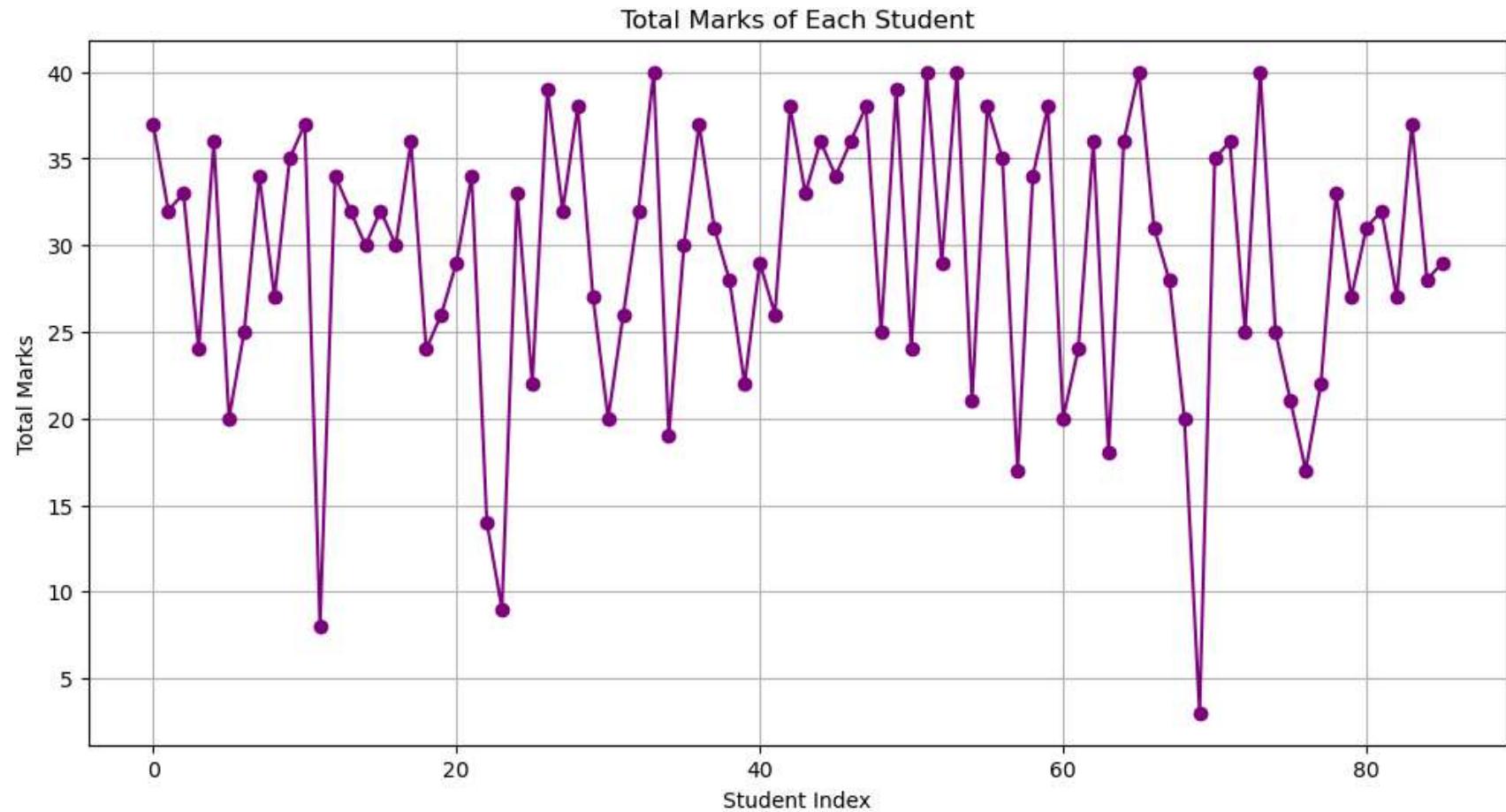


This kernel creates a boxplot to compare the marks distribution across all questions, highlighting the spread, median, and outliers for each question.

In [145]:

```
plt.figure(figsize=(12, 6))
plt.plot(data['Total'], marker='o', color='purple')
plt.title('Total Marks of Each Student')
plt.xlabel('Student Index')
plt.ylabel('Total Marks')
```

```
plt.grid(True)  
plt.show()
```

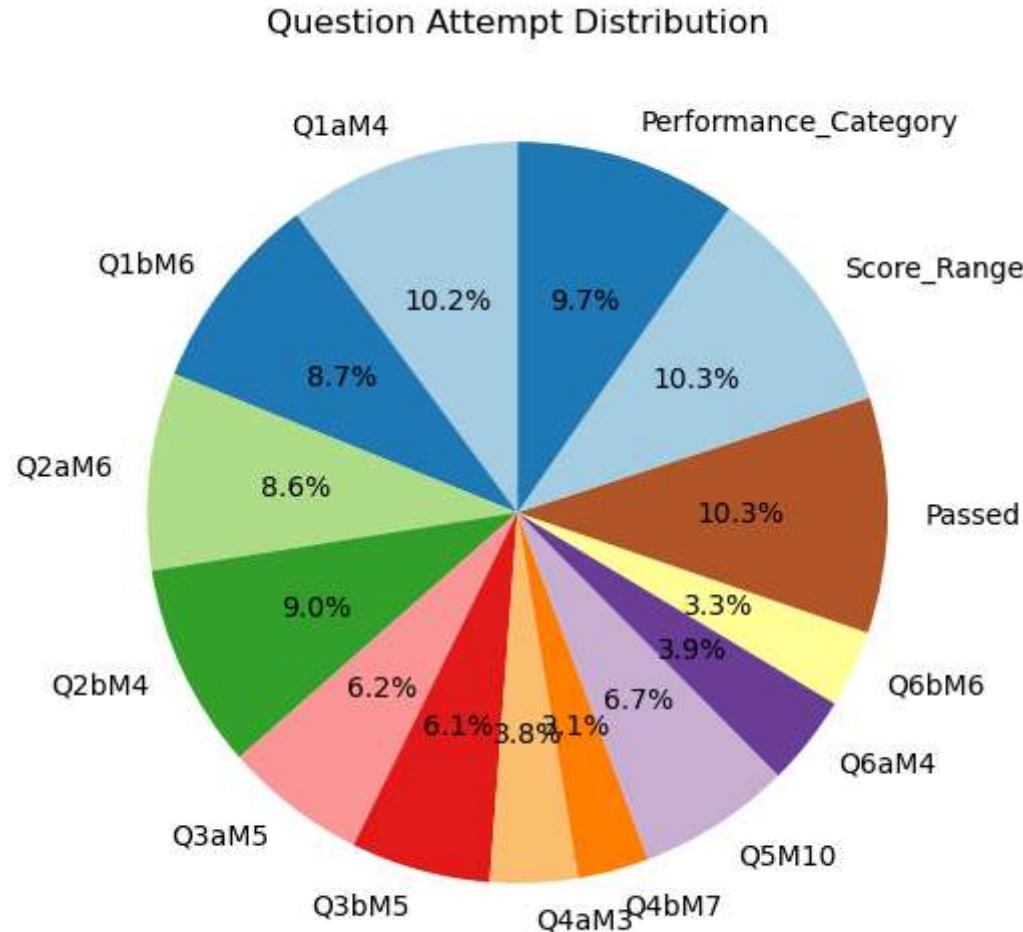


This kernel visualizes the total marks of each student as a line plot, showing how individual student scores vary across the dataset.

In [146]:

```
question_attempts = data.drop(columns='Total').notna().sum()  
  
plt.figure(figsize=(6, 6))  
plt.pie(question_attempts, labels=question_attempts.index, autopct='%1.1f%%', startangle=90, colors=plt.cm.Paired.colors)
```

```
plt.title('Question Attempt Distribution')
plt.show()
```



This kernel counts the number of non-null (attempted) entries for each question and visualizes the distribution of attempts across the questions using a pie chart.

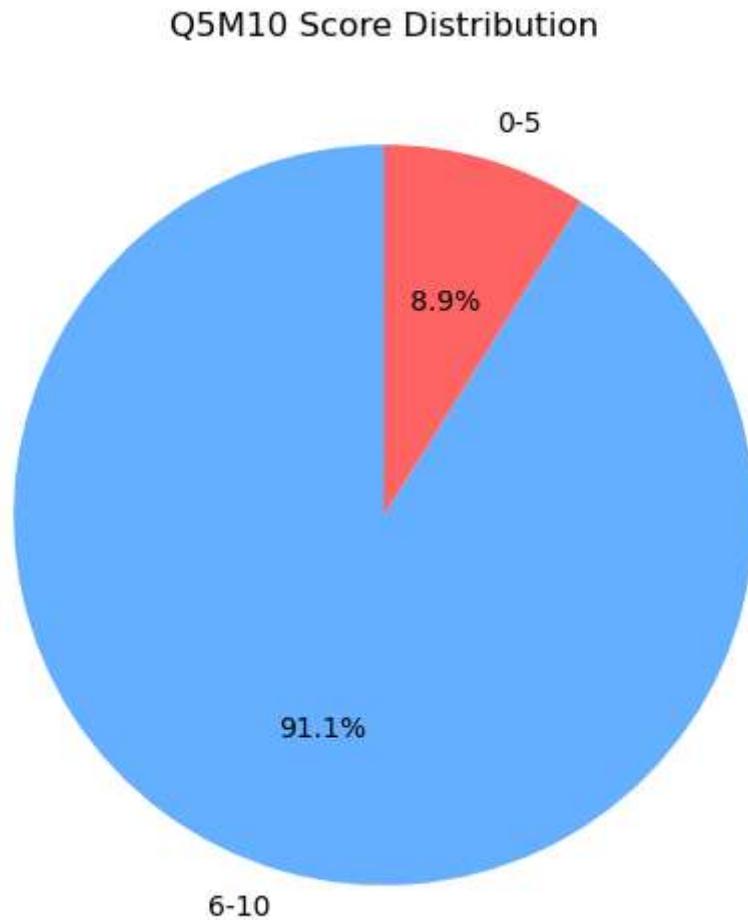
In [147]:

```
q5_marks = data['Q5M10'].dropna()
bins = [0, 5, 10]
```

```
labels = ['0-5', '6-10']
q5_bins = pd.cut(q5_marks, bins=bins, labels=labels)

q5_distribution = q5_bins.value_counts()

plt.figure(figsize=(6, 6))
plt.pie(q5_distribution, labels=q5_distribution.index, autopct='%1.1f%%', startangle=90, colors=['#66b3ff', '#ff6666'])
plt.title('Q5M10 Score Distribution')
plt.show()
```



This kernel analyzes the distribution of marks for the question "Q5M10" by grouping scores into two ranges (0-5, 6-10) and visualizes the distribution using a pie chart.

In [148...]

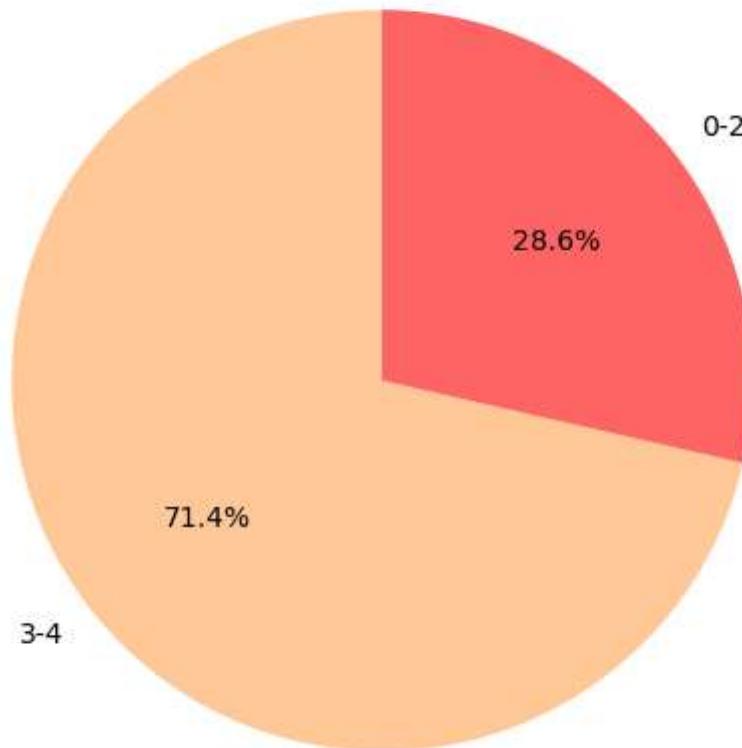
```
q1a_marks = data['Q1aM4'].dropna()

bins = [0, 2, 4]
labels = ['0-2', '3-4']
q1a_bins = pd.cut(q1a_marks, bins=bins, labels=labels)

q1a_distribution = q1a_bins.value_counts()

plt.figure(figsize=(6, 6))
plt.pie(q1a_distribution, labels=q1a_distribution.index, autopct='%1.1f%%', startangle=90, colors=['#ffcc99', '#ff6666'])
plt.title('Q1aM4 Score Distribution')
plt.show()
```

Q1aM4 Score Distribution



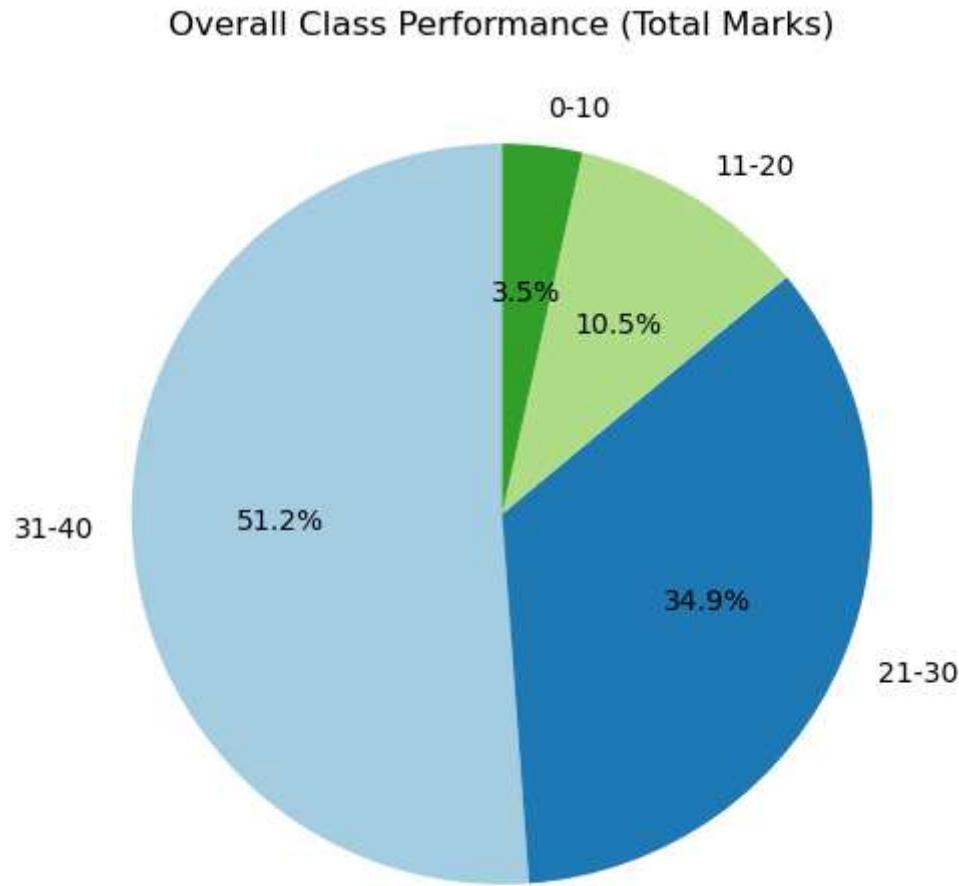
This kernel analyzes the distribution of marks for the question "Q1aM4" by grouping scores into two ranges (0-2, 3-4) and visualizes the distribution using a pie chart.

In [149...]

```
bins = [0, 10, 20, 30, 40]
labels = ['0-10', '11-20', '21-30', '31-40']
total_bins = pd.cut(data['Total'], bins=bins, labels=labels)

total_distribution = total_bins.value_counts()
```

```
plt.figure(figsize=(6, 6))
plt.pie(total_distribution, labels=total_distribution.index, autopct='%1.1f%%', startangle=90, colors=plt.cm.Paired.colors)
plt.title('Overall Class Performance (Total Marks)')
plt.show()
```



This kernel categorizes students based on their total marks into predefined score ranges (0-10, 11-20, 21-30, 31-40) and visualizes the overall class performance using a pie chart.

