Student Performance Factors

Amy Fishencord

This dataset provides a comprehensive overview of various factors affecting student performance in exams. It includes information on study habits, attendance, parental involvement, and other aspects influencing academic success.

In [8]:
```
#importing required libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

#loading in data set
data = pd.read_csv('StudentPerformanceFactors.csv')
print(data.info())
data.head()
```

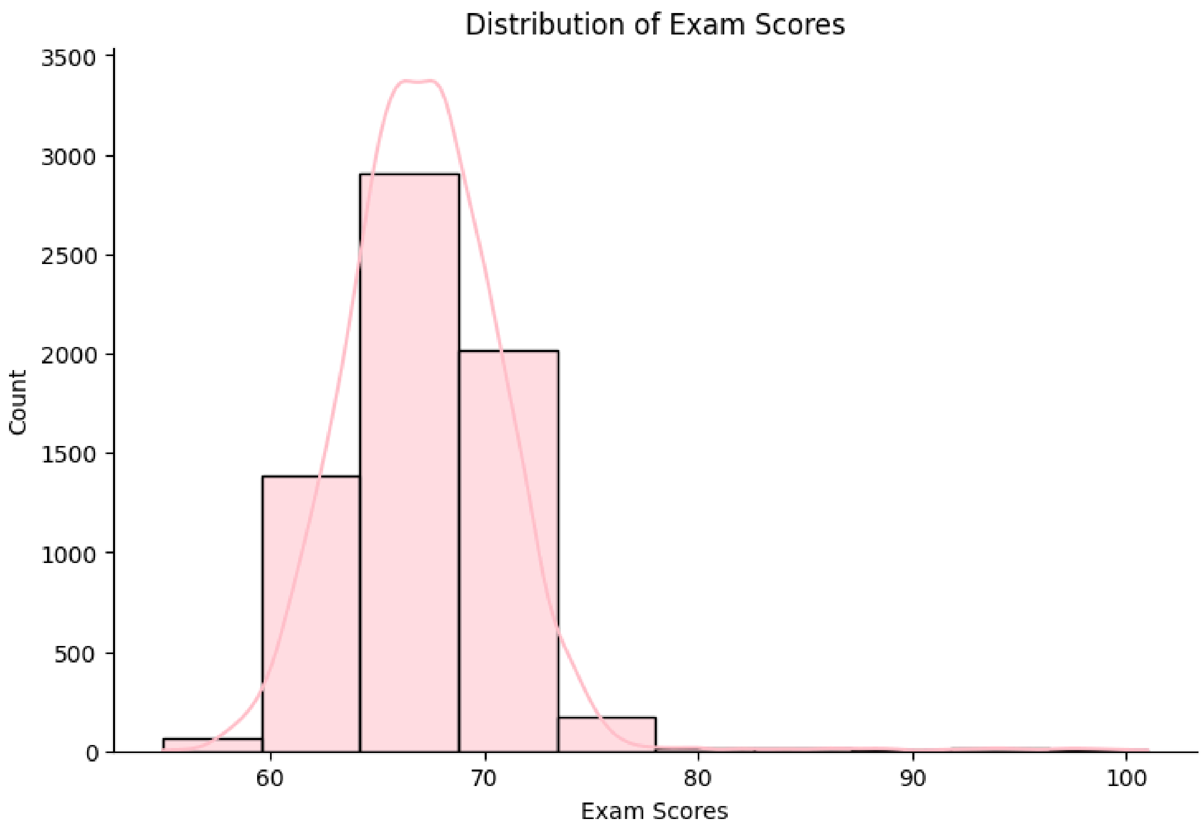
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6607 entries, 0 to 6606
Data columns (total 20 columns):
 #   Column                      Non-Null Count  Dtype 
---  ------                      --------------  ----- 
 0   Hours_Studied               6607 non-null   int64 
 1   Attendance                  6607 non-null   int64 
 2   Parental_Involvement        6607 non-null   object
 3   Access_to_Resources         6607 non-null   object
 4   Extracurricular_Activities  6607 non-null   object
 5   Sleep_Hours                 6607 non-null   int64 
 6   Previous_Scores             6607 non-null   int64 
 7   Motivation_Level            6607 non-null   object
 8   Internet_Access             6607 non-null   object
 9   Tutoring_Sessions           6607 non-null   int64 
 10  Family_Income               6607 non-null   object
 11  Teacher_Quality             6529 non-null   object
 12  School_Type                 6607 non-null   object
 13  Peer_Influence              6607 non-null   object
 14  Physical_Activity           6607 non-null   int64 
 15  Learning_Disabilities       6607 non-null   object
 16  Parental_Education_Level    6517 non-null   object
 17  Distance_from_Home          6540 non-null   object
 18  Gender                      6607 non-null   object
 19  Exam_Score                  6607 non-null   int64 
dtypes: int64(7), object(13)
memory usage: 1.0+ MB
None
```

Out[8]:

	Hours_Studied	Attendance	Parental_Involvement	Access_to_Resources	Extracurri
0	23	84	Low	High	
1	19	64	Low	Medium	
2	24	98	Medium	Medium	
3	29	89	Low	Medium	
4	19	92	Medium	Medium	

Overview of Variables

In [9]:
```
#distribution plot
sns.displot(data['Exam_Score'], kde=True, color='pink', bins=10, aspect=1.5)

plt.title('Distribution of Exam Scores') #adding plot labels
plt.xlabel('Exam Scores')
plt.ylabel('Count')
plt.show()
```



Most students score in the 60's and 70's

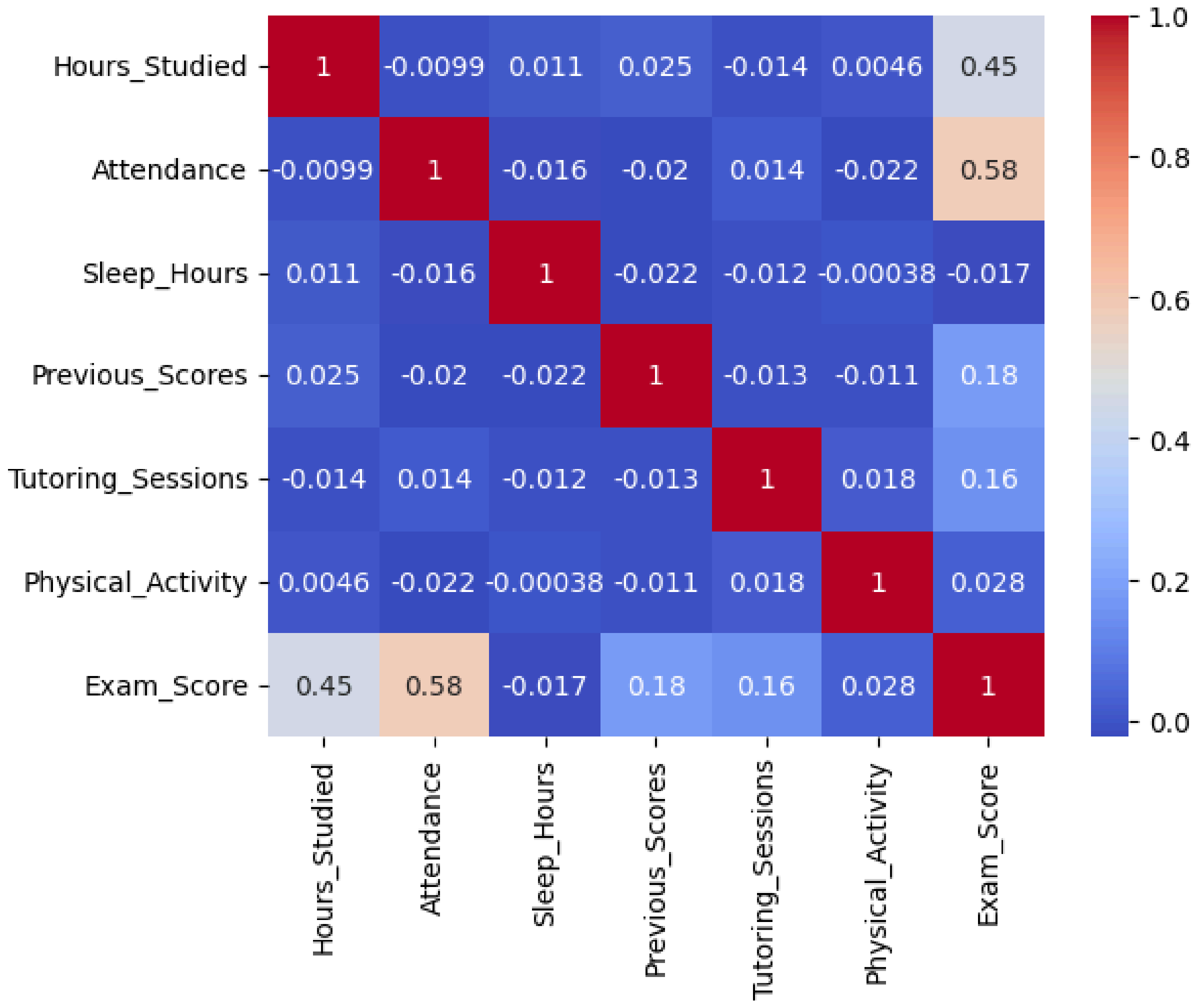
In [6]:
```
print(data.describe())
```

```
       Hours_Studied   Attendance  Sleep_Hours  Previous_Scores  \
count    6607.000000  6607.000000   6607.00000      6607.000000
mean       19.975329    79.977448      7.02906        75.070531
std         5.990594    11.547475      1.46812        14.399784
min         1.000000    60.000000      4.00000        50.000000
25%        16.000000    70.000000      6.00000        63.000000
50%        20.000000    80.000000      7.00000        75.000000
75%        24.000000    90.000000      8.00000        88.000000
max        44.000000   100.000000     10.00000       100.000000

       Tutoring_Sessions  Physical_Activity   Exam_Score
count        6607.000000        6607.000000  6607.000000
mean            1.493719           2.967610    67.235659
std             1.230570           1.031231     3.890456
min             0.000000           0.000000    55.000000
25%             1.000000           2.000000    65.000000
50%             1.000000           3.000000    67.000000
75%             2.000000           4.000000    69.000000
max             8.000000           6.000000   101.000000
```

Question 1:

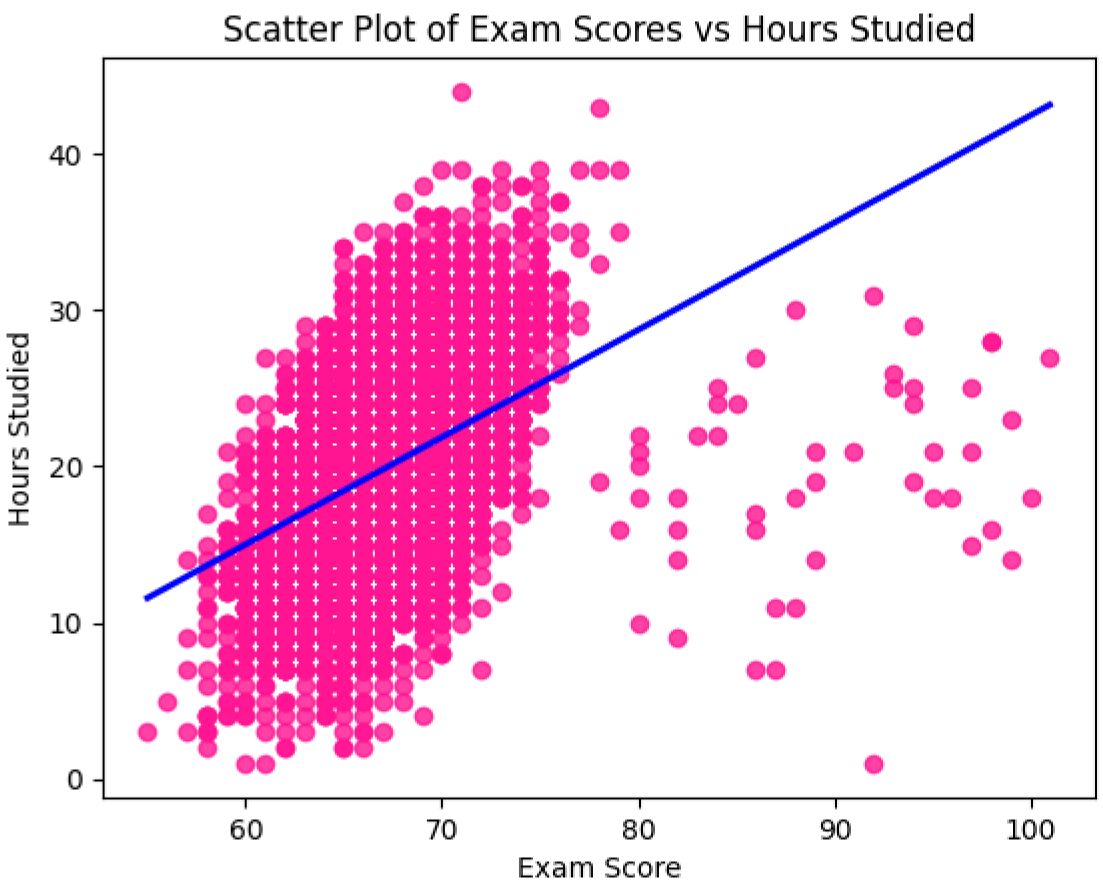
What factors have the most significant impact on students' exam scores?

In [12]:
```
#correlation matrix
numeric_data = data.select_dtypes(include=[np.number])
corr_matrix = numeric_data.corr()

#add heatmap
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
plt.show()
```



Positive correlation between exam scores and other variables that we will be looking at further.

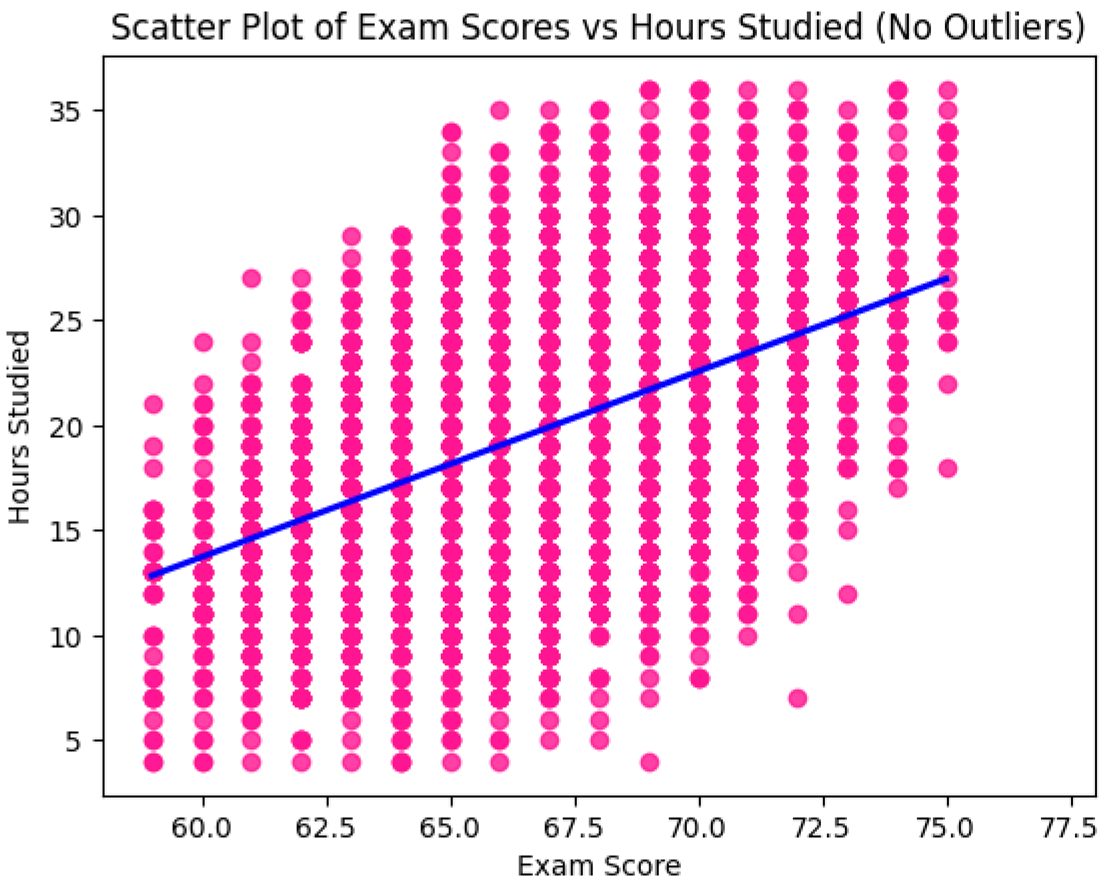
In [10]:
```
#linear regression plot
sns.regplot(x='Exam_Score', y='Hours_Studied', data=data, color='deeppink',
plt.title('Scatter Plot of Exam Scores vs Hours Studied')
plt.xlabel('Exam Score')
plt.ylabel('Hours Studied')
plt.show()

#Calculated correlation coeff
correlation = data['Exam_Score'].corr(data['Hours_Studied'])
print(correlation)
```



```
0.4454549540752825
```

A calculated correlation coefficiant of 0.45 showing a moderately positive correlation between exam scores and hours studied. Would this corrleation be stronger without the outliers?

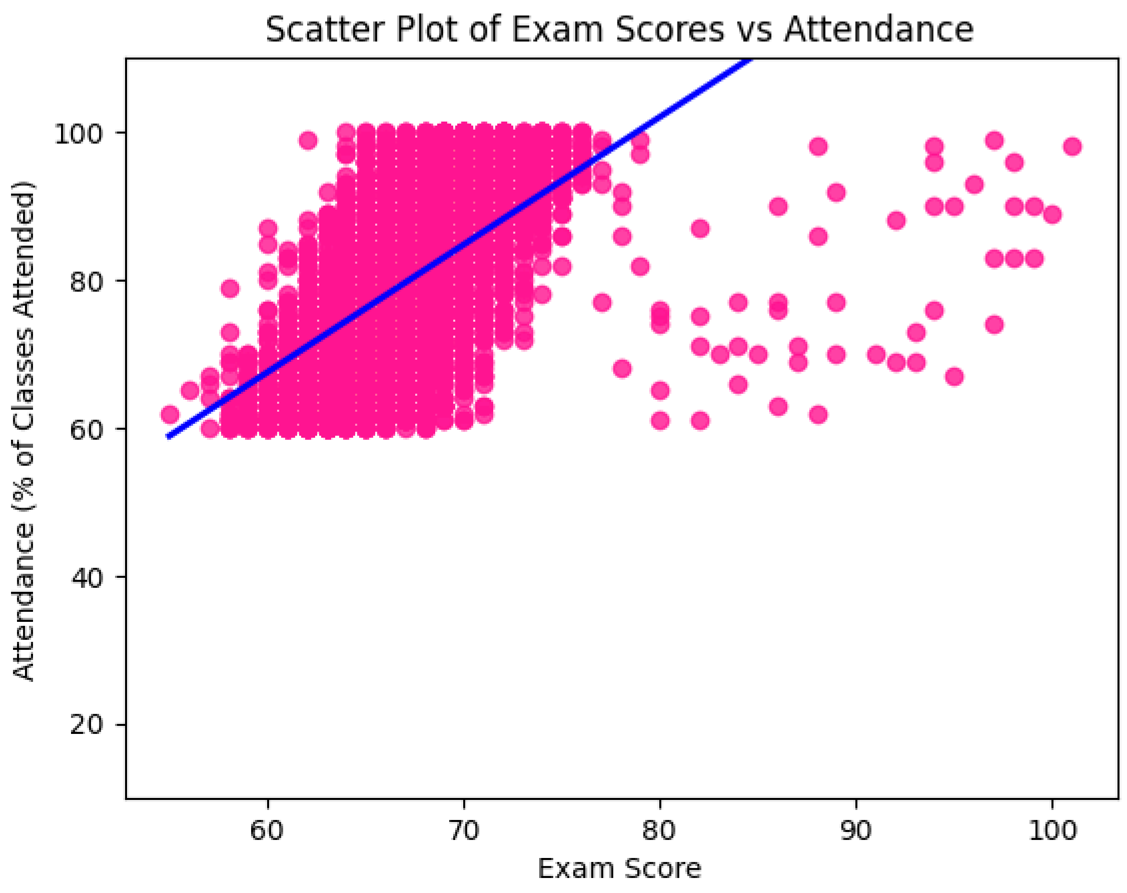
In [13]:
```
#calculate IQR to remove outliers
Q1 = data[['Exam_Score', 'Hours_Studied']].quantile(0.25)
Q3 = data[['Exam_Score', 'Hours_Studied']].quantile(0.75)
IQR = Q3 - Q1

#filter out the outliers using the IQR method
filtered_data = data[~((data[['Exam_Score', 'Hours_Studied']] < (Q1 - 1.5 *
                       (data[['Exam_Score', 'Hours_Studied']] > (Q3 + 1.5 *

#linear reg plot (without outliers)
sns.regplot(x='Exam_Score', y='Hours_Studied', data=filtered_data, color='de
plt.title('Scatter Plot of Exam Scores vs Hours Studied (No Outliers)')
plt.xlabel('Exam Score')
plt.ylabel('Hours Studied')
plt.xlim(58,78)
plt.show()

#calculated correlation coefficient (without outliers)
correlation = filtered_data['Exam_Score'].corr(filtered_data['Hours_Studied'
print("Correlation coefficient (without outliers):", correlation)
```



```
Correlation coefficient (without outliers): 0.494991169076997
```

When we remove our outliers in the exam score varaible, our calculated correlation coefficiant is 0.5. This shows a moderately positive correlation that is higher without the outliers.

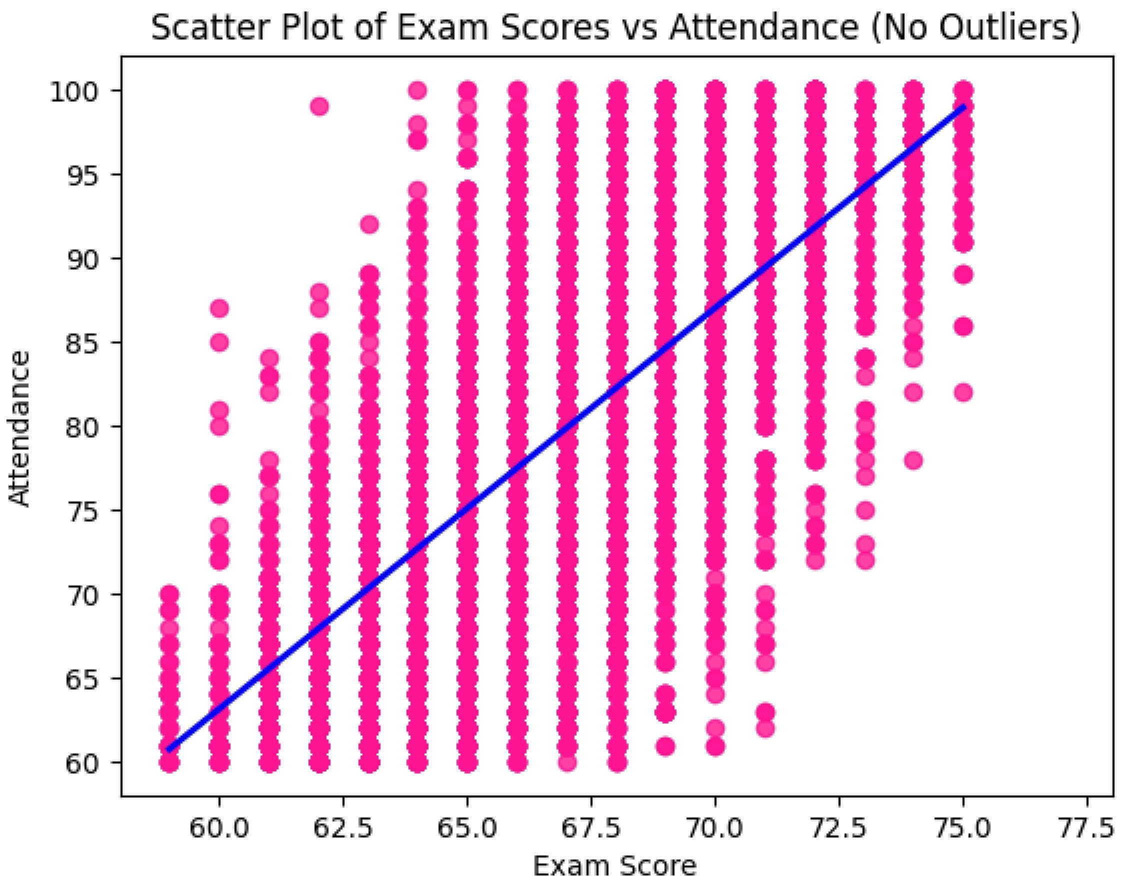
In [14]:
```
#linear regression plot
sns.regplot(x='Exam_Score', y='Attendance', data=data, color='deeppink', ci=
plt.title('Scatter Plot of Exam Scores vs Attendance')
plt.xlabel('Exam Score')
plt.ylabel('Attendance (% of Classes Attended)')
plt.ylim(10,110)
plt.show()

#corelation coefficient
correlation = data['Exam_Score'].corr(data['Attendance'])
print(correlation)
```



```
0.5810718633120647
```

A calculated correlation coefficiant of 0.58 showing a moderately positive correlation between exam scores and attendance.

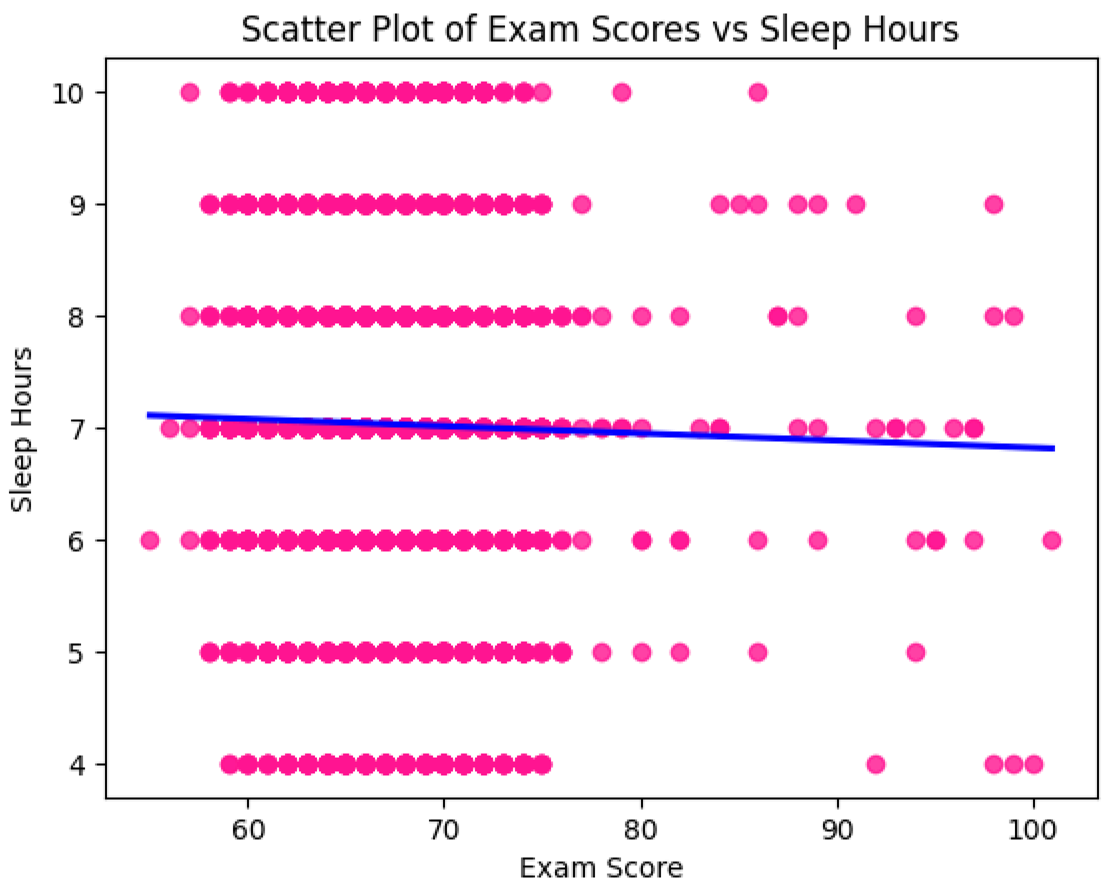
In [15]:
```
#calculate IQR to remove outliers
Q1 = data[['Exam_Score', 'Attendance']].quantile(0.25)
Q3 = data[['Exam_Score', 'Attendance']].quantile(0.75)
IQR = Q3 - Q1

#filter out the outliers using the IQR method
filtered_data = data[~((data[['Exam_Score', 'Attendance']] < (Q1 - 1.5 * IQF
                       (data[['Exam_Score', 'Attendance']] > (Q3 + 1.5 * IQF

#linear reg plot (without outliers)
sns.regplot(x='Exam_Score', y='Attendance', data=filtered_data, color='deepp
plt.title('Scatter Plot of Exam Scores vs Attendance (No Outliers)')
plt.xlabel('Exam Score')
plt.ylabel('Attendance')
plt.xlim(58,78)
plt.show()

#calculated correlation coefficient (without outliers)
correlation = filtered_data['Exam_Score'].corr(filtered_data['Attendance'])
print("Correlation coefficient (without outliers):", correlation)
```



```
Correlation coefficient (without outliers): 0.6738707423757742
```

When we remove our outliers in the attendance varaible, our calculated correlation coefficiant is 0.67. This shows a high correlation that is stronger without the outliers.

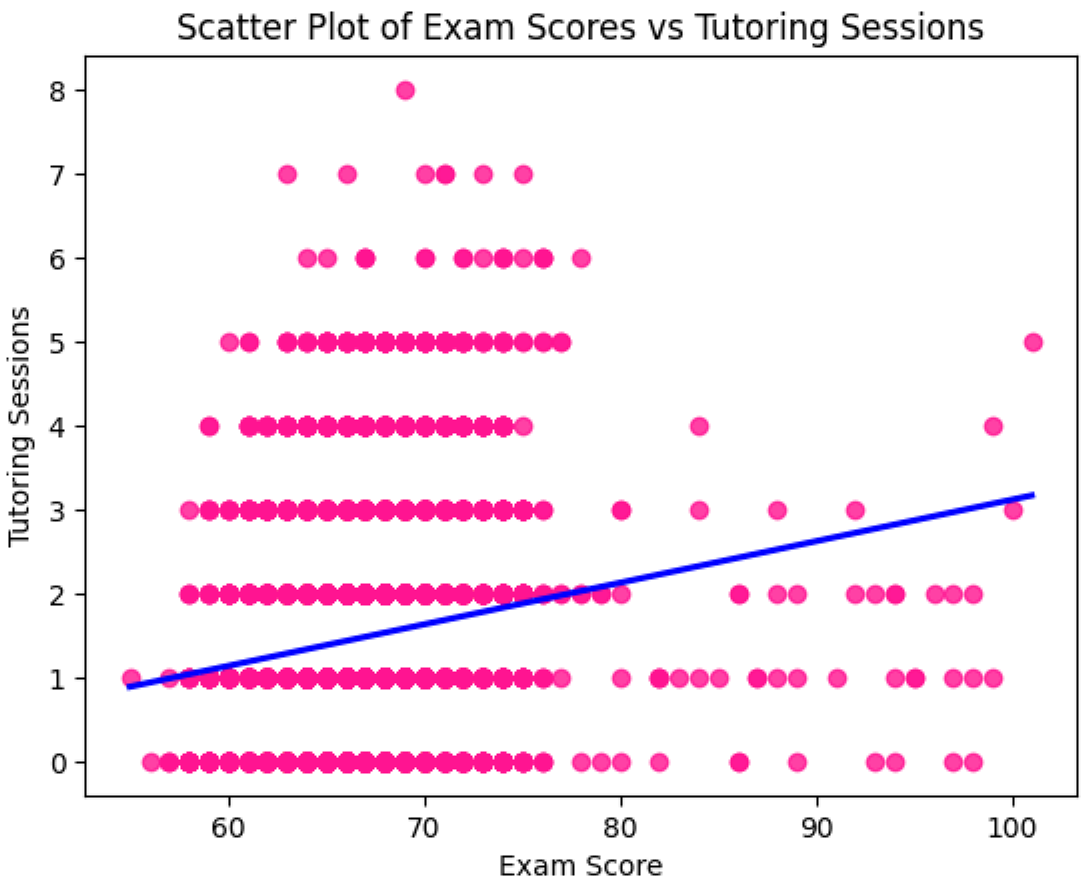
```
In [16]: #linear regression plot
sns.regplot(x='Exam_Score', y='Sleep_Hours', data=data, color='deeppink', ci
plt.title('Scatter Plot of Exam Scores vs Sleep Hours')
plt.xlabel('Exam Score')
plt.ylabel('Sleep Hours')
plt.show()

#correlation coefficiant
correlation = data['Exam_Score'].corr(data['Sleep_Hours'])
print(correlation)
```



```
-0.01702162857150254
```

A calculated correlation coefficiant shows a weak negative correlation close to zero. The amount of hours of sleep you get will have almost no impact on your exam score

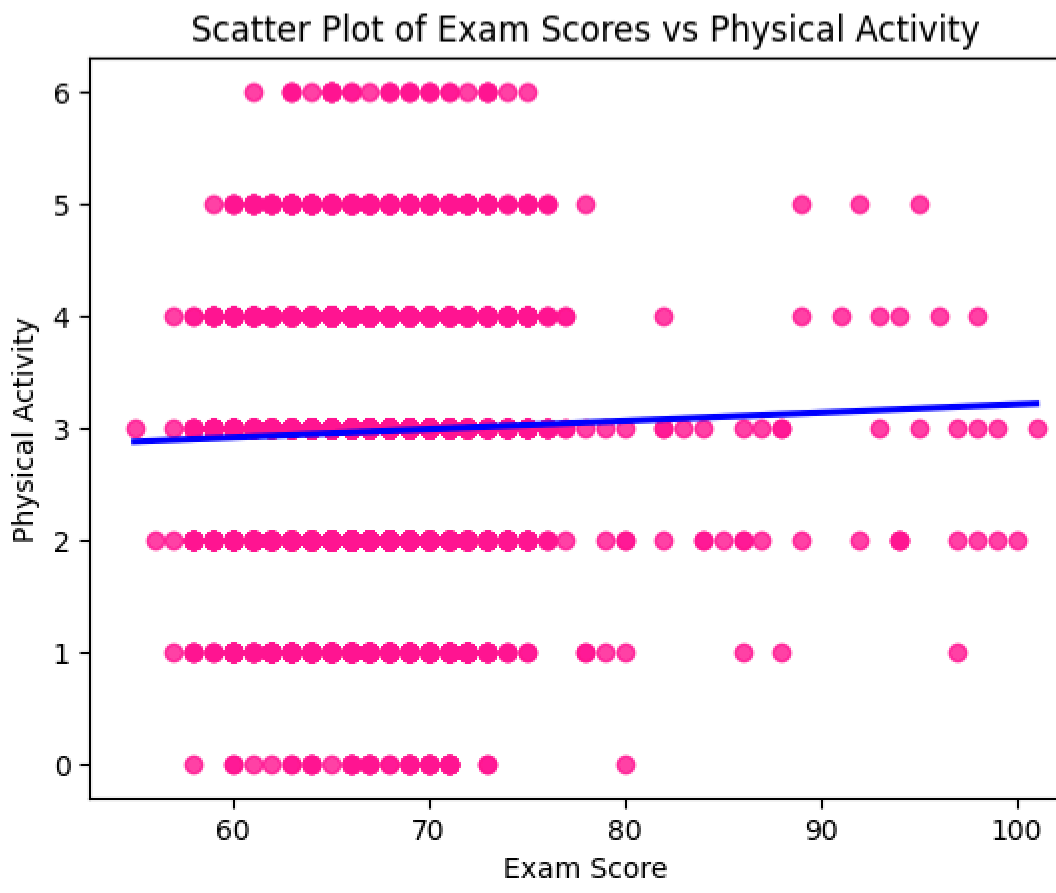
In [17]:
```
#linear regression
sns.regplot(x='Exam_Score', y='Tutoring_Sessions', data=data, color='deeppir
plt.title('Scatter Plot of Exam Scores vs Tutoring Sessions')
plt.xlabel('Exam Score')
plt.ylabel('Tutoring Sessions')
plt.show()

#correlation coefficiant
correlation = data['Exam_Score'].corr(data['Tutoring_Sessions'])
print(correlation)
```



```
0.15652518539225332
```

A calculated correlation coefficiant of 0.15 shows a weak positive correlation. The amount of tutoring sessions you attend will not have much of an impact on your exam score.

```
In [18]: #Linear Reg plot
sns.regplot(x='Exam_Score', y='Physical_Activity', data=data, color='deeppir
plt.title('Scatter Plot of Exam Scores vs Physical Activity')
plt.xlabel('Exam Score')
plt.ylabel('Physical Activity')
plt.show()

#Correlation coefficiant
correlation = data['Exam_Score'].corr(data['Physical_Activity'])
print(correlation)
```



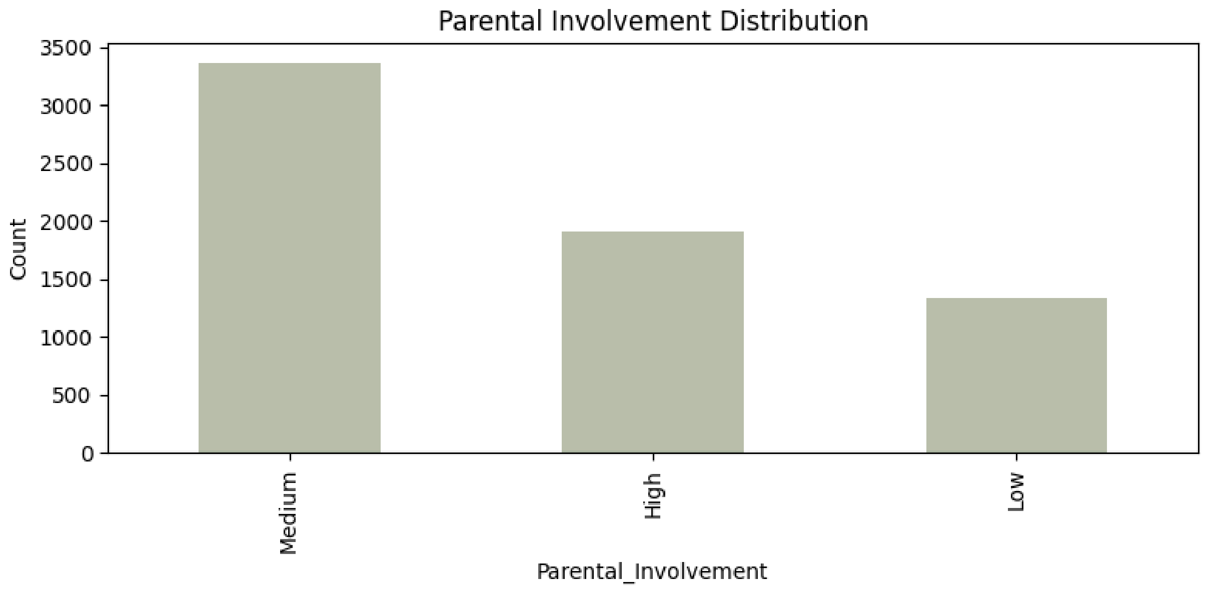
```
0.02782443618025749
```

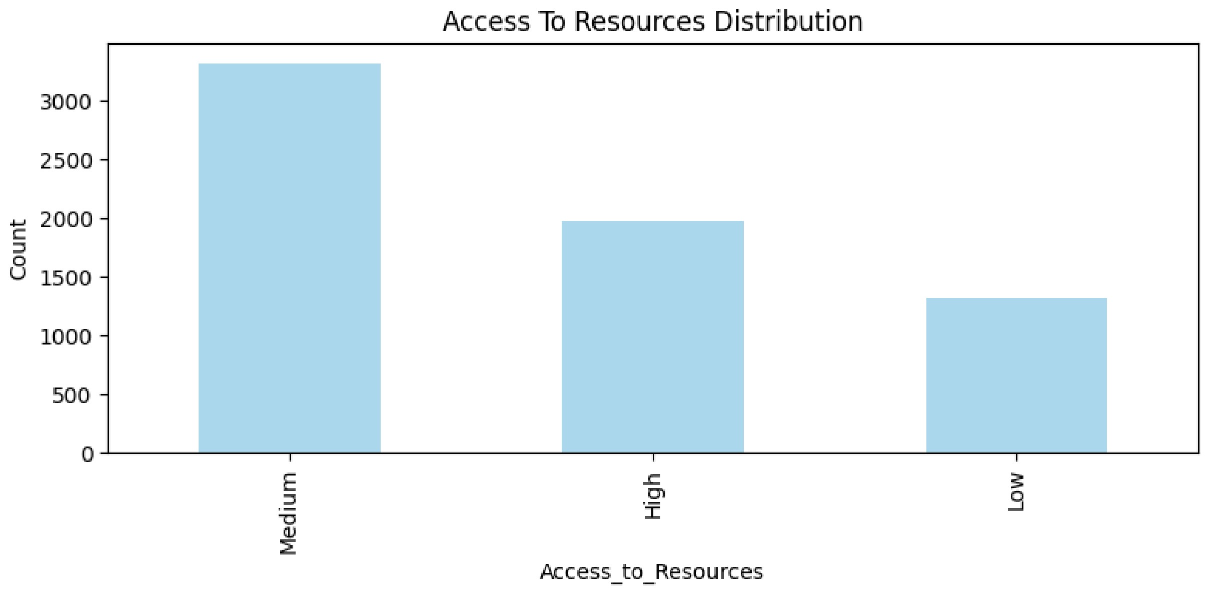
A calculated correlation coefficiant of 0.29 shows a low correlation. It is not likely that changes in physical activity do not significantly predict changes in exam scores.

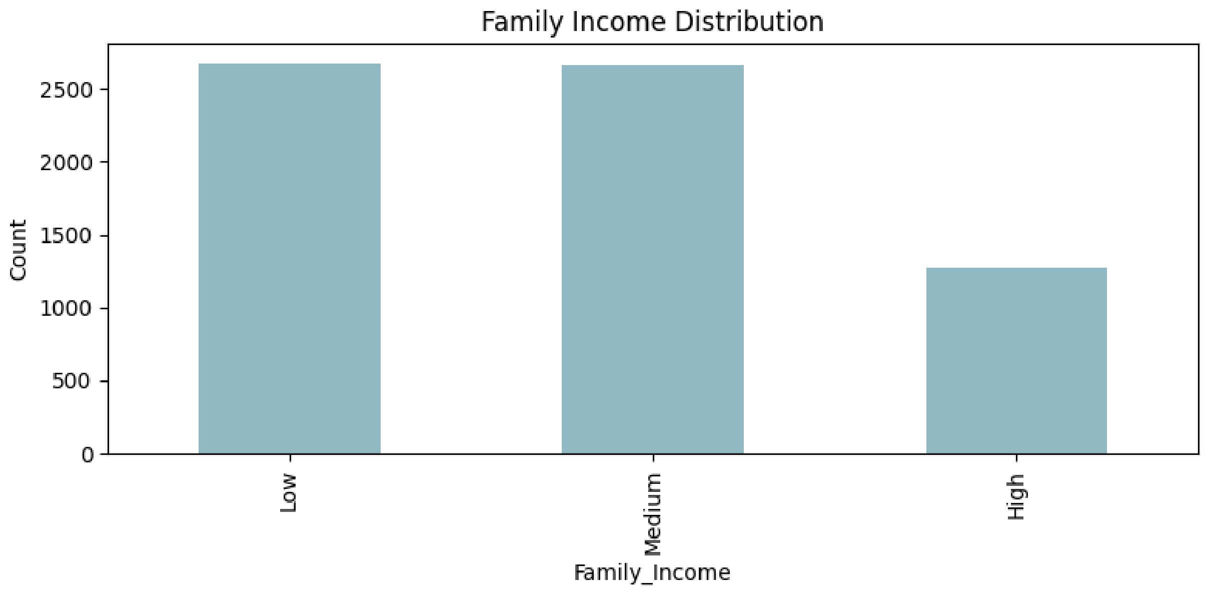
To answer our question the factors that impact students exam scores the most are hours studied and attendance. Attendance affects students exam scores the most.

Question 2: How do family related factors influence exam scores?

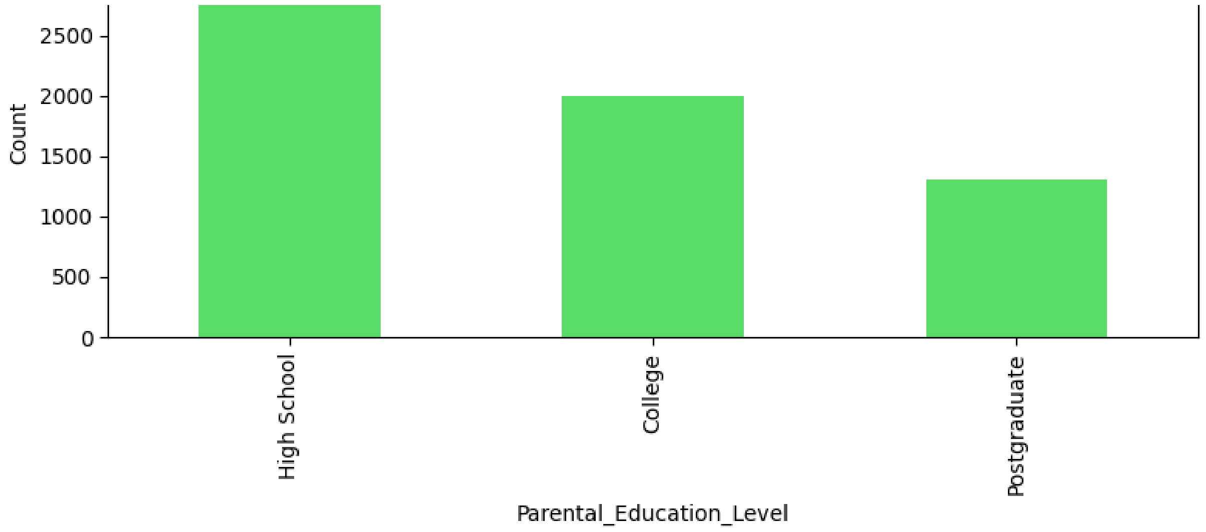
In [19]:
```
categorical_vars = ['Parental_Involvement', 'Access_to_Resources', 'Family_I

#create subplots for all variables
fig, axes = plt.subplots(len(categorical_vars), 1, figsize=(8, len(categoric

#loop through each variable and create a bar plot
for i, var in enumerate(categorical_vars):
    counts = data[var].value_counts()  #get counts of each category
    counts.plot.bar(ax=axes[i], color=np.random.rand(3,), alpha=0.7)
    axes[i].set_title(f'{var.replace("_", " ").title()} Distribution') #addi
    axes[i].set_ylabel('Count')

plt.tight_layout() #adjust layout
plt.show()
```


Parental Involvement Distribution
Count
3500
3000
2500
2000
1500
1000
500
0
Medium
High
Low
Parental_Involvement


Access To Resources Distribution
Count
3000
2500
2000
1500
1000
500
0
Medium
High
Low
Access_to_Resources

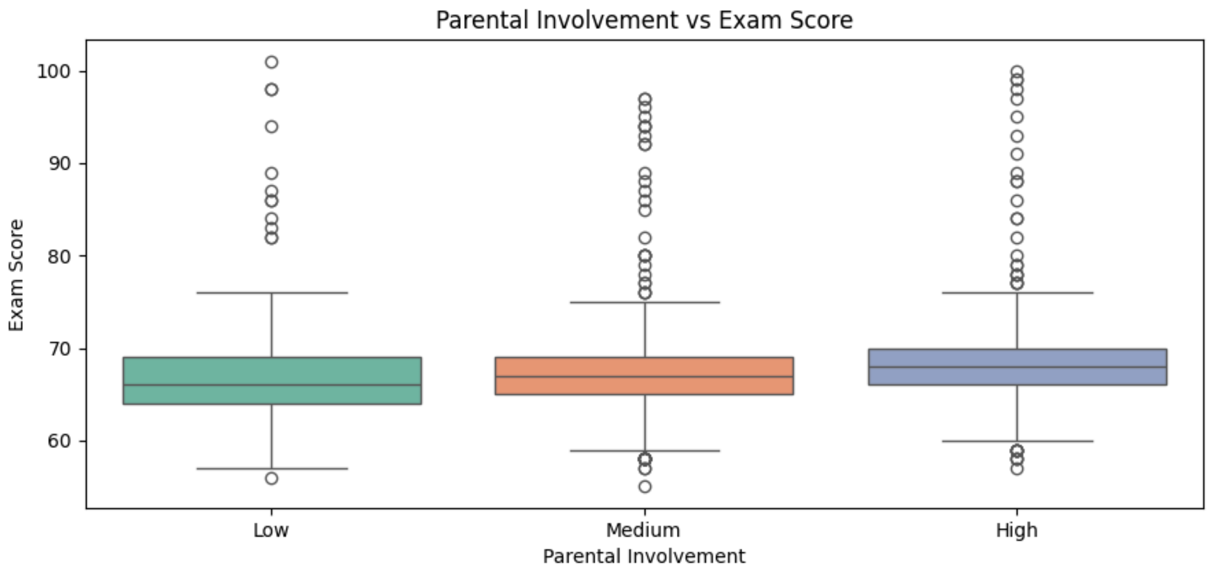

Family Income Distribution
Count
2500
2000
1500
1000
500
0
Low
Medium
High
Family_Income

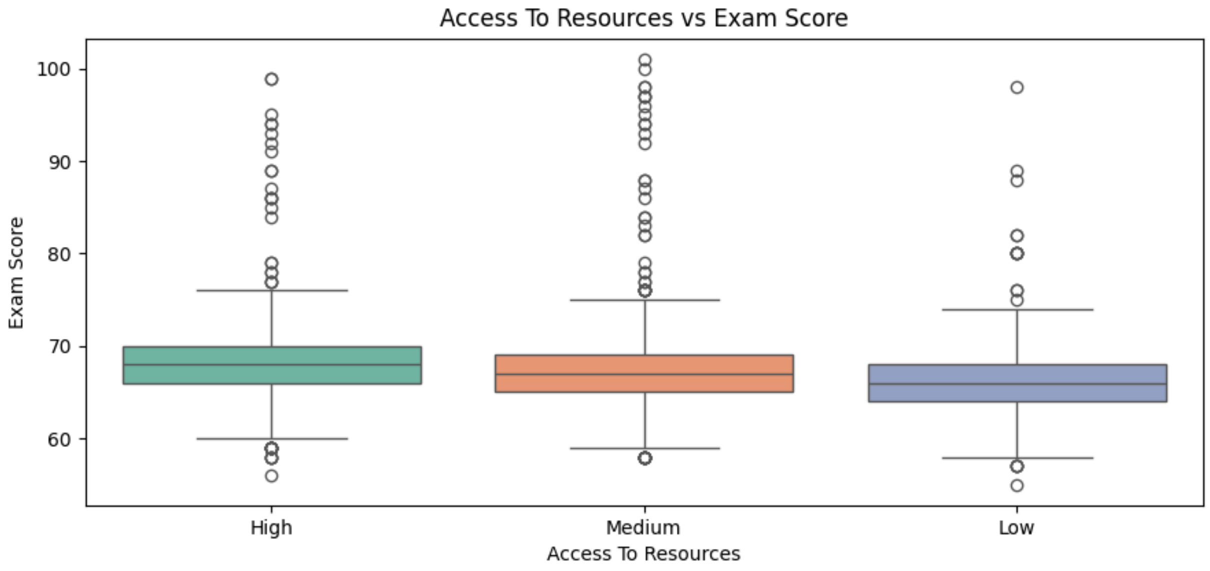
Parental Education Level Distribution
3000

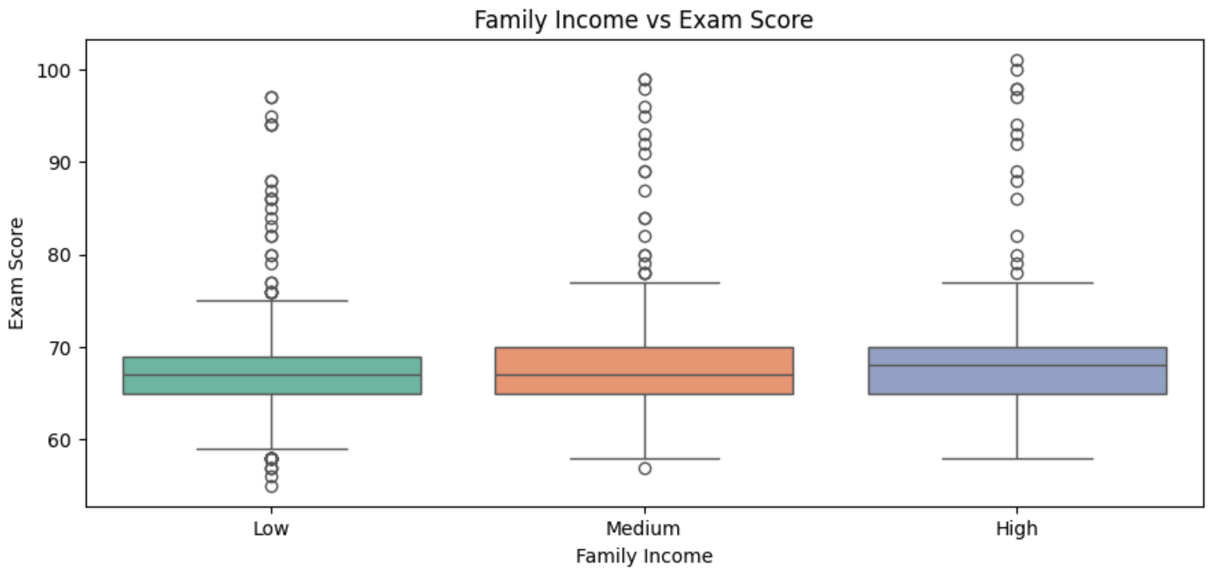


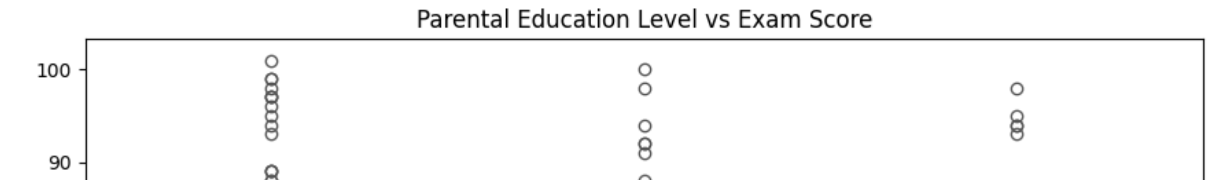
In [20]:
```
fig, axes = plt.subplots(len(categorical_vars), 1, figsize=(9, 17))

#loop through each variable and create a boxplot
for i, var in enumerate(categorical_vars):
    sns.boxplot(x=data[var], y=data['Exam_Score'], ax=axes[i], palette='Set2
    axes[i].set_title(f'{var.replace("_", " ").title()} vs Exam Score')  #ti
    axes[i].set_xlabel(f'{var.replace("_", " ").title()}')  #xaxis label
    axes[i].set_ylabel('Exam Score')  #xaxis label

#adjust layout and show the plot
plt.tight_layout()
plt.show()
```

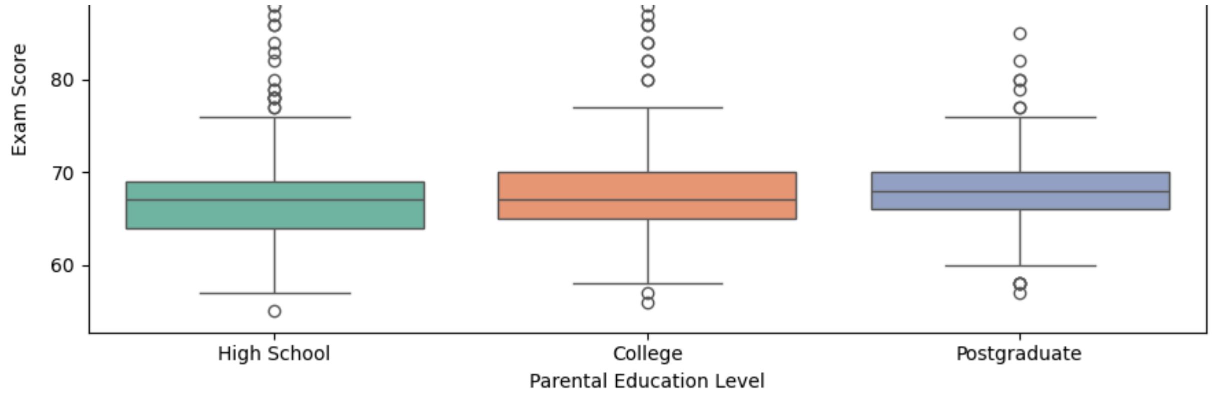
```
/tmp/ipykernel_632366/1621321702.py:5: FutureWarning: 

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  sns.boxplot(x=data[var], y=data['Exam_Score'], ax=axes[i], palette='Set2')
/tmp/ipykernel_632366/1621321702.py:5: FutureWarning: 

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  sns.boxplot(x=data[var], y=data['Exam_Score'], ax=axes[i], palette='Set2')
/tmp/ipykernel_632366/1621321702.py:5: FutureWarning: 

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  sns.boxplot(x=data[var], y=data['Exam_Score'], ax=axes[i], palette='Set2')
/tmp/ipykernel_632366/1621321702.py:5: FutureWarning: 

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  sns.boxplot(x=data[var], y=data['Exam_Score'], ax=axes[i], palette='Set2')
```


Parental Involvement vs Exam Score
Exam Score
100
90
80
70
60
Low
Medium
High
Parental Involvement


Access To Resources vs Exam Score
Exam Score
100
90
80
70
60
High
Medium
Low
Access To Resources


Family Income vs Exam Score
Exam Score
100
90
80
70
60
Low
Medium
High
Family Income

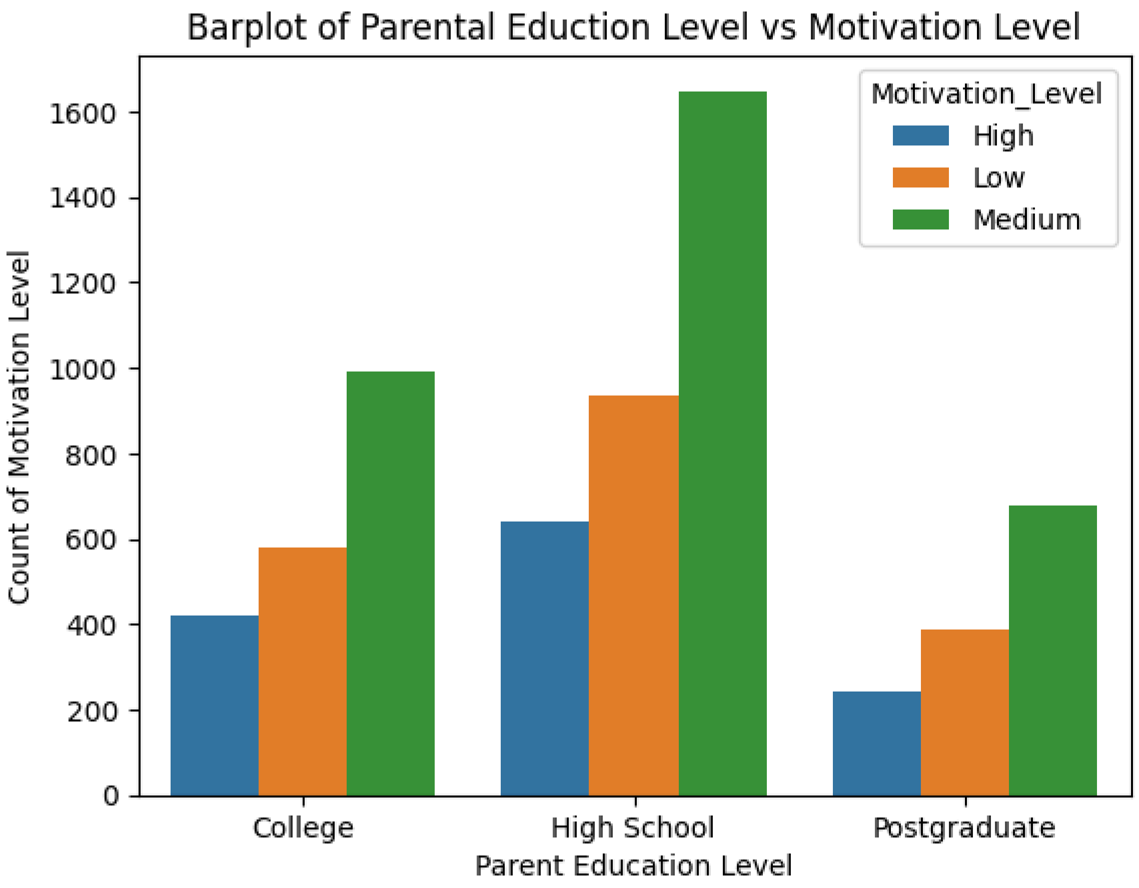

Parental Education Level vs Exam Score
100
90



To answer our question, almost all students who have higher parent involvement, access to resources, family income, and education level tend to have higher exam scores. We see this by focusing on the mean values, which show a positive trend, and by examining our outlier variables, which further emphasize the stronger performance of students with these factors. The presence of outliers in the data may indicate extreme cases, but the overall pattern remains consistent with the influence of these variables on exam scores.

Question 3: Do parents education levels affect students motivational levels?

In [21]:
```
#calculate the counts of Motivation_Level by Parental_Education_Level
count_data = data.groupby(['Parental_Education_Level', 'Motivation_Level']).

#create barplot
sns.barplot(x='Parental_Education_Level', y='count', hue='Motivation_Level',
plt.title('Barplot of Parental Eduction Level vs Motivation Level')
plt.xlabel('Parent Education Level')
plt.ylabel('Count of Motivation Level')
plt.show()
```

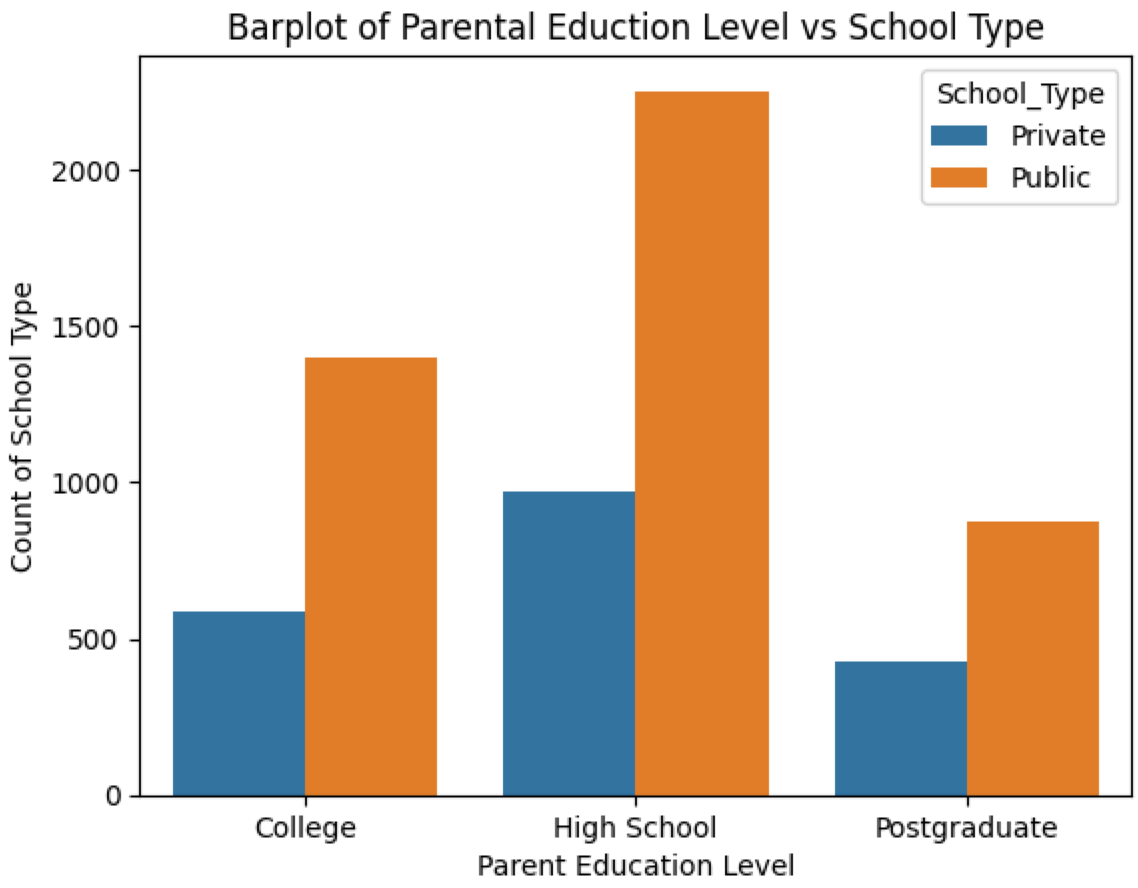


Parents education levels do not have an affect on students motivational levels, most student have an average stress level.

Question 4: Do parents education levels affect the type of school students go to?

In [22]:
```
#calculate the counts of School_Type by Parental_Education_Level
count_data = data.groupby(['Parental_Education_Level', 'School_Type']).size(

#pivot the table to restructure counts
pivot_table = count_data.pivot(index='Parental_Education_Level', columns='Sc

#add a column for the public to private ratio
pivot_table['Public_to_Private_Ratio'] = pivot_table['Public'] / pivot_table

print(pivot_table)

#create barplot
sns.barplot(x='Parental_Education_Level', y='count', hue='School_Type', data
plt.title('Barplot of Parental Eduction Level vs School Type')
plt.xlabel('Parent Education Level')
plt.ylabel('Count of School Type')
plt.show()
```

```
School_Type                Private  Public  Public_to_Private_Ratio
Parental_Education_Level                                           
College                        589    1400                 2.376910
High School                    972    2251                 2.315844
Postgraduate                   428     877                 2.049065
```



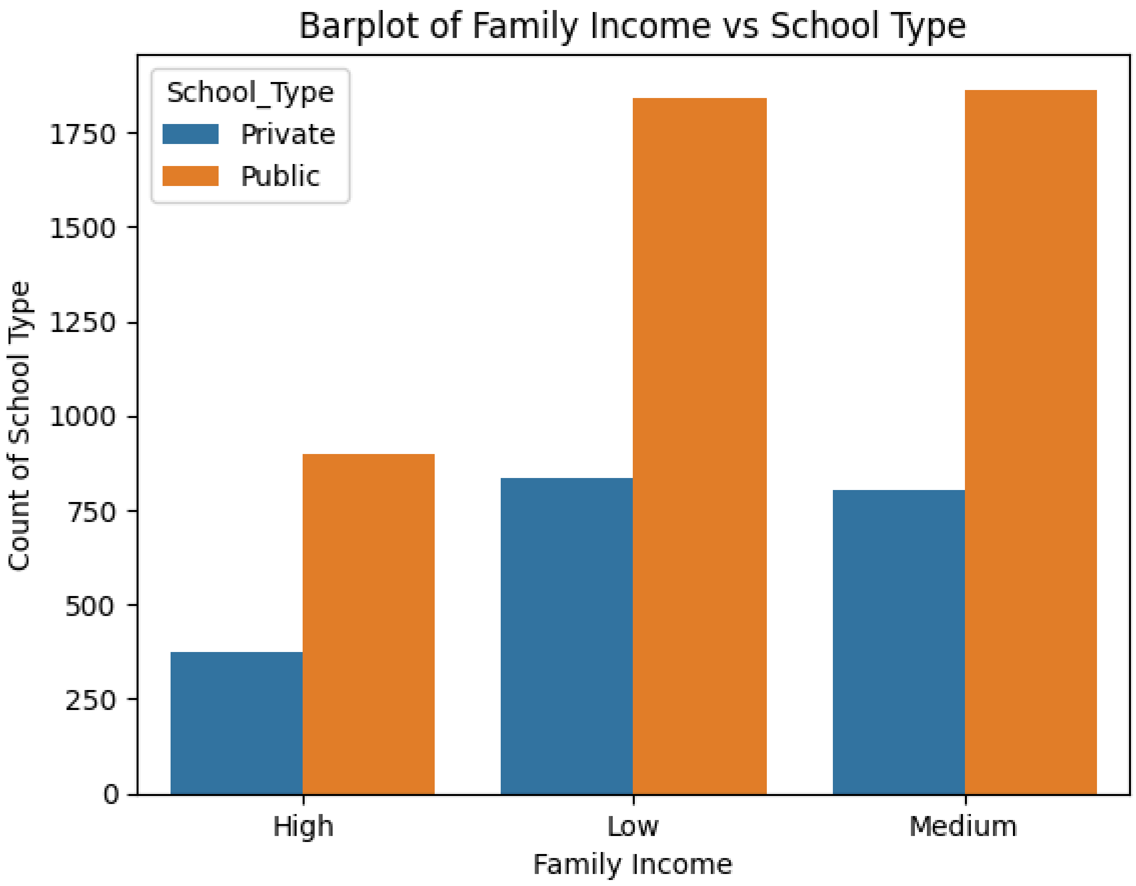
Parent educational levels do not affect the type of school students go to. The ratio of public to private school are similar by all education levels and most students attend public schools.

Question 5: Does family income affect the type of school students go to?

In [23]:
```
#calculate the counts of School_Type by Famil
count_data = data.groupby(['Family_Income', 'School_Type']).size().reset_inc

#pivot the table to restructure counts
pivot_table = count_data.pivot(index='Family_Income', columns='School_Type',

#add a column for the public to private ratio
pivot_table['Public_to_Private_Ratio'] = pivot_table['Public'] / pivot_table

print(pivot_table)

#create barplot
```

```
sns.barplot(x='Family_Income', y='count', hue='School_Type', data=count_data
plt.title('Barplot of Family Income vs School Type')
plt.xlabel('Family Income')
plt.ylabel('Count of School Type')
plt.show()
```

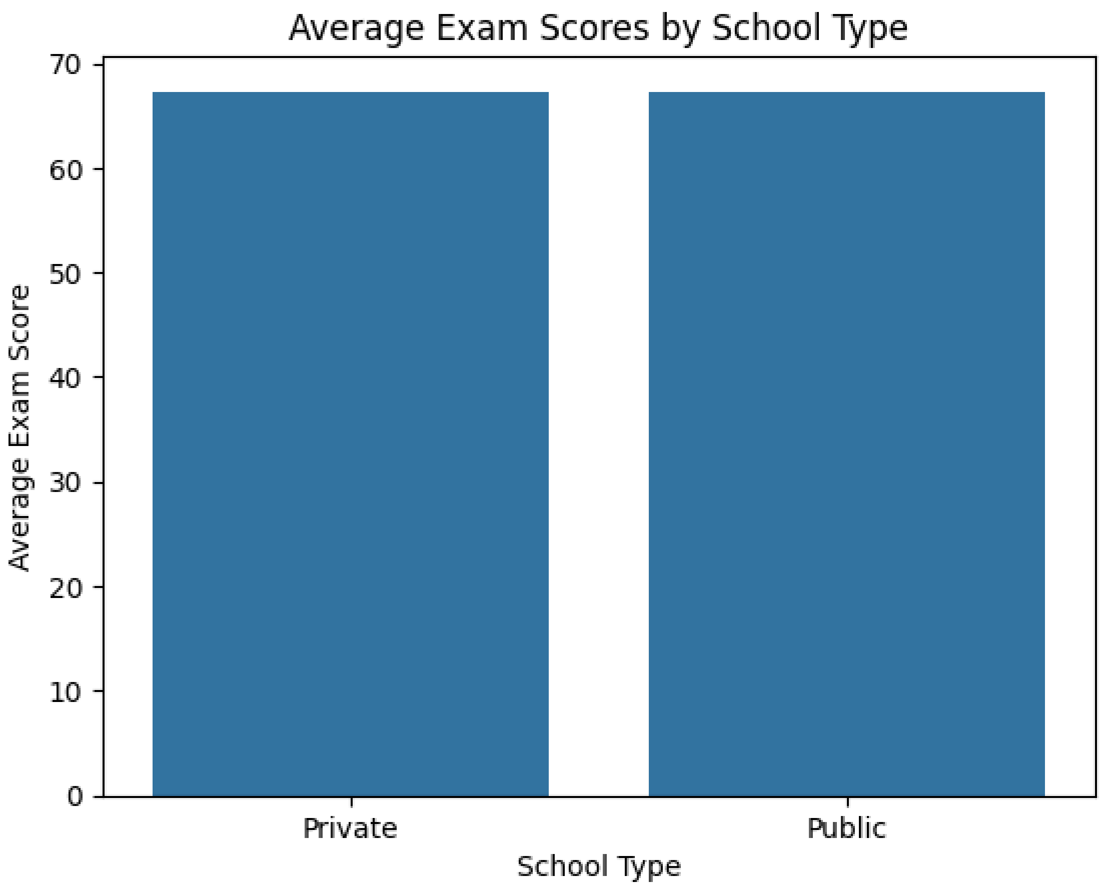
```
School_Type    Private  Public  Public_to_Private_Ratio
Family_Income                                          
High               373     896                 2.402145
Low                833    1839                 2.207683
Medium             803    1863                 2.320050
```



Family income does not affect the type of school students go to. The ratios of public to private schools are also similar by family income and most students attend public school.

Question 6: Which type of schools have the higher exam scores on average?

```
In [24]: #group by school type and calc mean exam score
average_scores = data.groupby('School_Type')['Exam_Score'].mean().reset_inde

#create barplot
sns.barplot(x='School_Type', y='Exam_Score', data=average_scores)

plt.title('Average Exam Scores by School Type') #adding title and labels
plt.xlabel('School Type')
plt.ylabel('Average Exam Score')
plt.show()
```

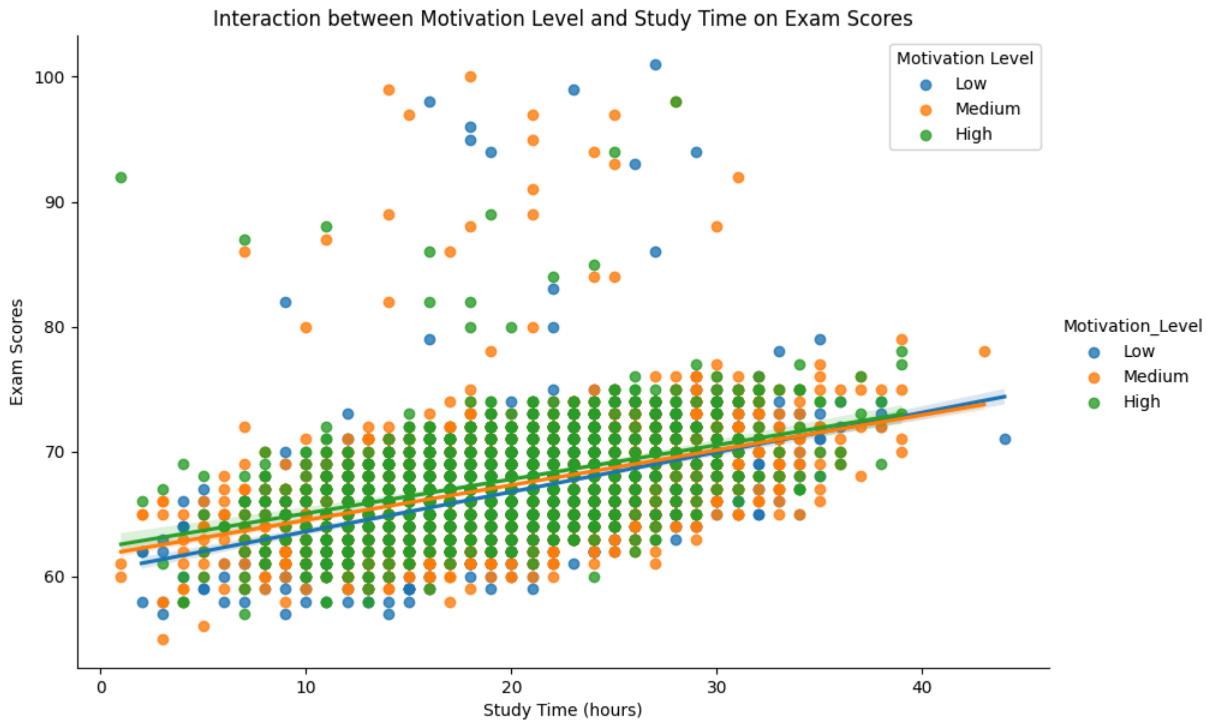


As we can see neither public or private schools are producing higher scores. Public and private schools have the same average exam scores, showing the type of school you go to will most likely not have an impact on your exam score.

Question 7: How do motivation levels and study time interact?

In [25]:
```
plt.figure(figsize=(10, 6))

#scatterplot grouped by motivational level
sns.lmplot(x='Hours_Studied', y='Exam_Score', hue='Motivation_Level', data=d

#labels and title
plt.title('Interaction between Motivation Level and Study Time on Exam Score
plt.xlabel('Study Time (hours)')
plt.ylabel('Exam Scores')
plt.legend(title='Motivation Level')
plt.show()

#calculate correlation coefficients for each motivation level
for level in data['Motivation_Level'].unique():
    subset = data[data['Motivation_Level'] == level]
    correlation = subset['Hours_Studied'].corr(subset['Exam_Score'])
    print(f"Correlation for Motivation Level {level}: {correlation:.2f}")
```

```
<Figure size 1000x600 with 0 Axes>
```



```
Correlation for Motivation Level Low: 0.47
Correlation for Motivation Level Medium: 0.44
Correlation for Motivation Level High: 0.42
```

Low Motivation Level (0.47): A moderate positive correlation. This suggests that for students with low motivation, there is a noticeable relationship between study time and exam scores—students who study more tend to perform better, though the relationship is not very strong.

Medium Motivation Level (0.44): Another moderate positive correlation, slightly lower than for low motivation. This implies a similar trend, but with slightly weaker predictive power.

High Motivation Level (0.42): The weakest positive correlation among the three levels. This suggests that for highly motivated students, study time is slightly less predictive of exam scores compared to those with medium or low motivation.

Predictive Model

Next, I will be creating a predictive model to predict the a students exam score based on their attendance and hours studied.

```
In [26]: #calculate IQR to remove outliers
Q1 = data[['Attendance', 'Hours_Studied', 'Exam_Score']].quantile(0.25)
```

```
Q3 = data[['Attendance', 'Hours_Studied', 'Exam_Score']].quantile(0.75)
IQR = Q3- Q1

#filter out outliers using IQR method
filtered_data = data[~(
    ((data[['Attendance', 'Hours_Studied', 'Exam_Score']] < (Q1 - 1.5 * IQR)
     (data[['Attendance', 'Hours_Studied', 'Exam_Score']] > (Q3 + 1.5 * IQR)
)]

#set x (features) and y (target)
X = filtered_data[['Attendance', 'Hours_Studied']]
y = filtered_data[['Exam_Score']]
```

Next, I will split the data into training and test sets. Will use 80% of the data for training and 20% of the data for testing to evaluate the model's performance.

In [27]:
```
from sklearn.model_selection import train_test_split

#split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar
```

Next, I will use the LinearRegression class to fit the model.

In [28]:
```
from sklearn.linear_model import LinearRegression

#initialize model
model = LinearRegression()

#train model
model.fit(X_train, y_train)
```

Out[28]:
```
▾ LinearRegression
LinearRegression()
```

Next, I will apply the model to make a prediction.

In [29]:
```
#make predictions on the test set
y_pred = model.predict(X_test)
```

Next, I will test the performance of the model.

In [31]:
```
from sklearn.metrics import mean_absolute_error, r2_score

mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Absolute Error (MAE):", mae)
print("R-squared (R2):", r2)
```

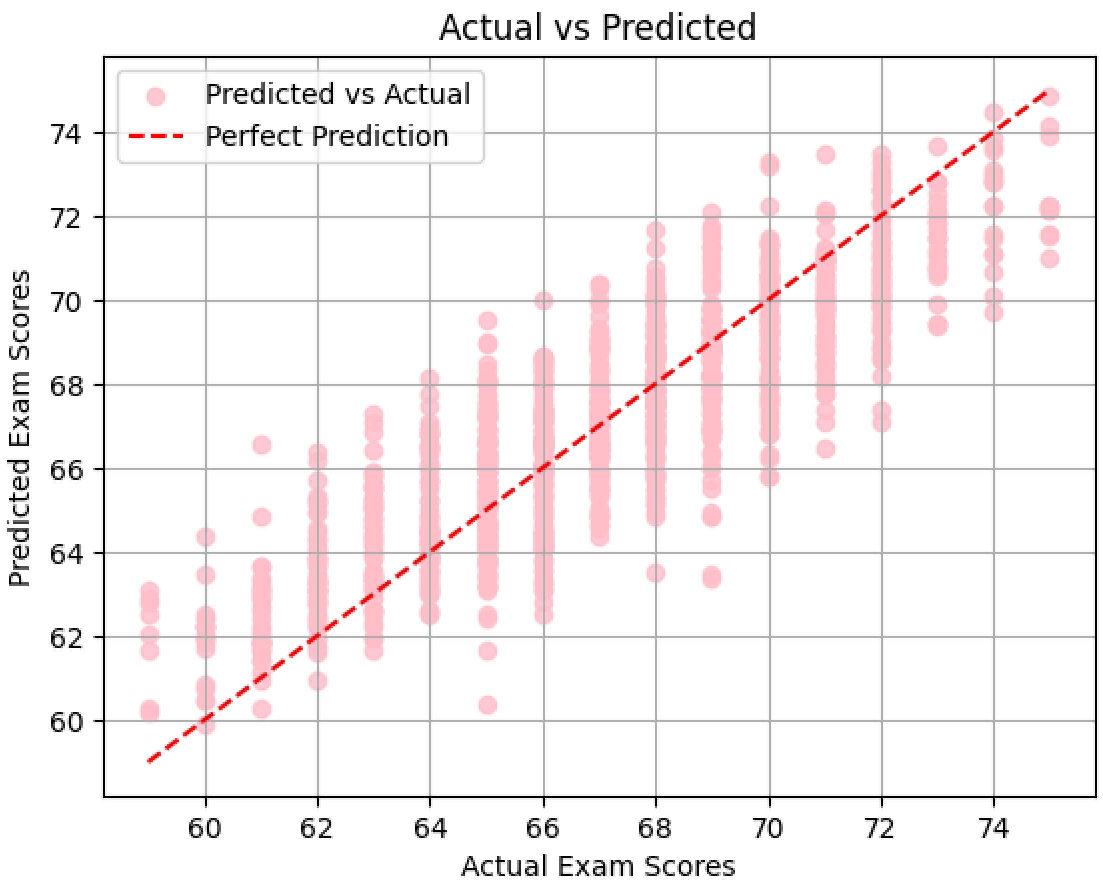
```
Mean Absolute Error (MAE): 1.3290443509773582
R-squared (R2): 0.7289459523708635
```

The Mean Absolute Error (MAE) represents the average difference between the predicted and actual exam scores. In this case, the MAE of 1.33 indicates that the model's predictions are, on average, 1.33 points away from the actual scores. This suggests the model provides fairly accurate predictions, especially when considering the scale of exam scores.

The R-squared value (0.73) indicates that 73% of the variation in exam scores is explained by the model's predictors (attendance and hours studied). This suggests that the model captures a significant portion of the relationship between these variables and exam performance.

Actual vs Predicted Comparison

In [33]:
```
plt.scatter(y_test, y_pred, color = 'pink', alpha=0.8, label='Predicted vs A
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='
plt.title('Actual vs Predicted')
plt.xlabel('Actual Exam Scores')
plt.ylabel('Predicted Exam Scores')
plt.legend()
plt.grid(True)
plt.show()
```



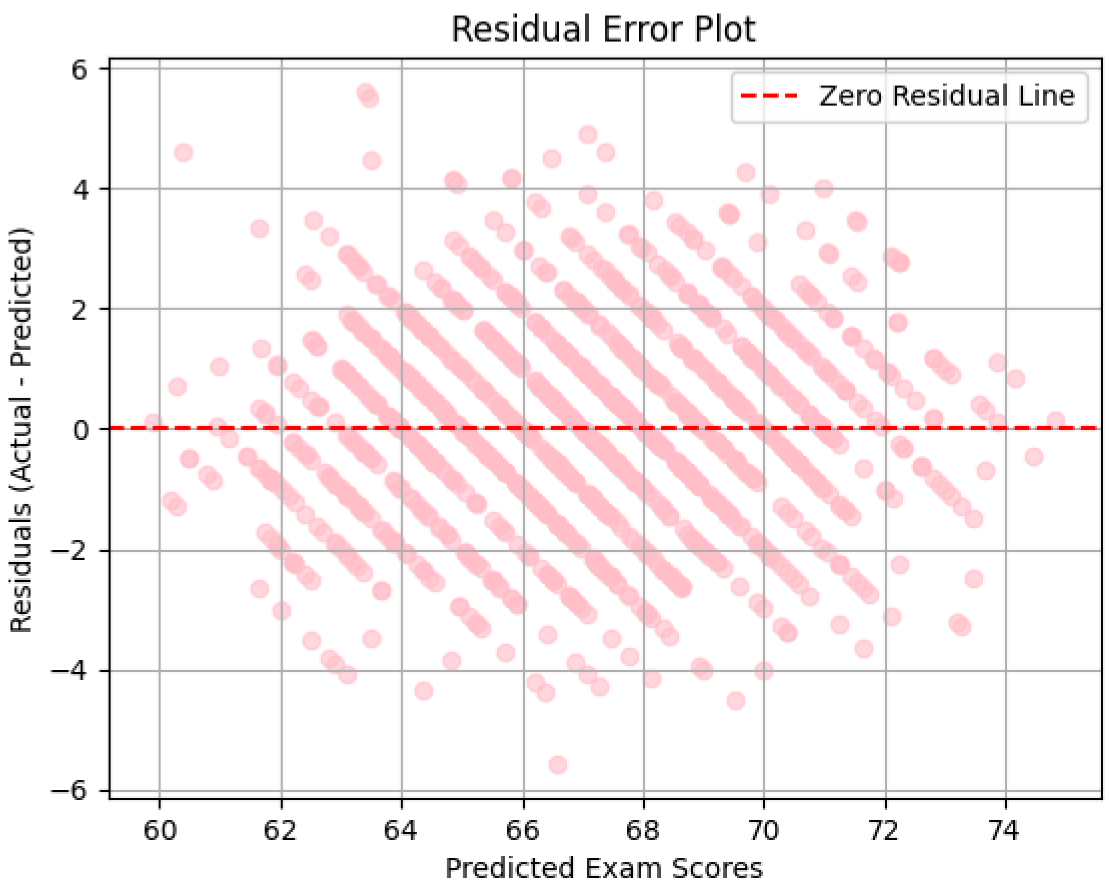
The pink points on the graph represent the actual exam scores on the x axis vs the predicted exam score on the y axis. The red dashed line represents a perfect prediction where the predicted score exactly matches the actual score. If the model predicted perfectly, all points would lie exactly on this line. If a data point lies exactly on this line, it means the model predicted that score perfectly, points above the line mean the model predicted a lower score than the actual value (underprediction), and points below the line mean the model predicted a higher score than the actual value (overprediction). Most data points are scattered around the perfect fit line representing a good model.

Residual Plot

Next, I will create a residual plot to show the difference between the actual values and the predicted values.

In [35]:
```python
#calculate residuals
residuals = y_test - y_pred

#plot residual error
plt.scatter(y_pred, residuals, color = 'pink', alpha = 0.6)
plt.axhline(y=0, color='red', linestyle='--', label='Zero Residual Line')
plt.title('Residual Error Plot')
plt.xlabel('Predicted Exam Scores')
plt.ylabel('Residuals (Actual - Predicted)')
plt.grid(True)
plt.legend()
```

Out[35]:
```
<matplotlib.legend.Legend at 0x72d506f97cb0>
```



The red dashed line represents the zero residual line, which indicates where the predicted values would exactly match the actual values. Most of the points are scattered around this line with no clear pattern, indicating that the model's residuals are randomly distributed. This suggests that the model has effectively captured the relationship between the features (attendance and hours studied) and the exam scores, without any significant bias or unaccounted patterns.

To understand the model easier, lets provide a simple example using it. Let us say student 1 attends class 85 percent of the time and studies for 15 hours a week. What is there predicted exam score?

In [40]:
```
student_data = pd.DataFrame([[85, 15]], columns=['Attendance', 'Hours_Studie
predicted_score = model.predict(student_data)

print(f"Predicted exam score for Student 1 (85% attendance, 15 hours studied
```

```
Predicted exam score for Student 1 (85% attendance, 15 hours studied): 66.60
```

Thank you!!