# 📚 Student Habits and Academic Performance Analysis

## Project Definition

This project aims to investigate the relationship between students' daily habits and their academic performance, particularly their exam scores. Using a real-world dataset containing lifestyle, behavioral, and academic attributes of 1000 students, we explore how factors such as study hours, sleep patterns, social media usage, mental health, diet quality, parental education, and extracurricular participation impact a student's academic success.

The project follows a structured data science workflow:

Data Cleaning: Handling missing values and preparing the dataset for analysis.

Exploratory Data Analysis (EDA): Understanding the distributions, patterns, and correlations within the data.

Feature Engineering: Converting categorical variables into numerical formats through one-hot encoding.

Statistical Modeling (OLS Regression): Building a basic linear model to understand simple relationships between key variables.

Machine Learning Modeling: Implementing and evaluating models such as Linear Regression and Random Forest to predict student exam scores more accurately.

Model Evaluation: Using metrics like $R^2$ Score and Root Mean Squared Error (RMSE) to assess the performance of each model.

Feature Importance Analysis: Identifying which student habits have the most significant impact on academic outcomes.

Through this project, we not only aim to predict exam scores but also extract meaningful insights about student behavior patterns, helping educators and students make informed decisions to enhance academic performance.

In [1]:
```
# Import all the Libraries
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.multicomp import pairwise_tukeyhsd
```

In [2]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
```

In [3]:
```
# Ignore the Warnings
warnings.filterwarnings("ignore")
```

In [4]:
```
# Read the Dataset
data = pd.read_csv("C:/Users/MK/Downloads/student_habits_performance.csv")
```

In [5]:
```
# Make a copy of the dataset
data_copy = data
```

# Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is the process of examining the data to understand its structure, spot patterns, detect anomalies, check assumptions, and test hypotheses. EDA involves summarizing the main characteristics of the dataset using statistics and visualizations like histograms, pairplots, heatmaps, and boxplots.

✅ Purpose:

Understand distributions and relationships

Identify missing values, outliers, or errors

Guide decisions for cleaning and modeling

In [6]:
```
# Print the first five rows of the dataset
data_copy.head()
```

Out[6]:

| | student_id | age | gender | study_hours_per_day | social_media_hours | netflix_hours | par |
|---|---|---|---|---|---|---|---|
| **0** | S1000 | 23 | Female | 0.0 | 1.2 | 1.1 | |
| **1** | S1001 | 20 | Female | 6.9 | 2.8 | 2.3 | |
| **2** | S1002 | 21 | Male | 1.4 | 3.1 | 1.3 | |
| **3** | S1003 | 23 | Female | 1.0 | 3.9 | 1.0 | |
| **4** | S1004 | 19 | Female | 5.0 | 4.4 | 0.5 | |

In [7]:
```
# Total no of Rows and Columns
data_copy.shape
```

Out[7]:
```
(1000, 16)
```

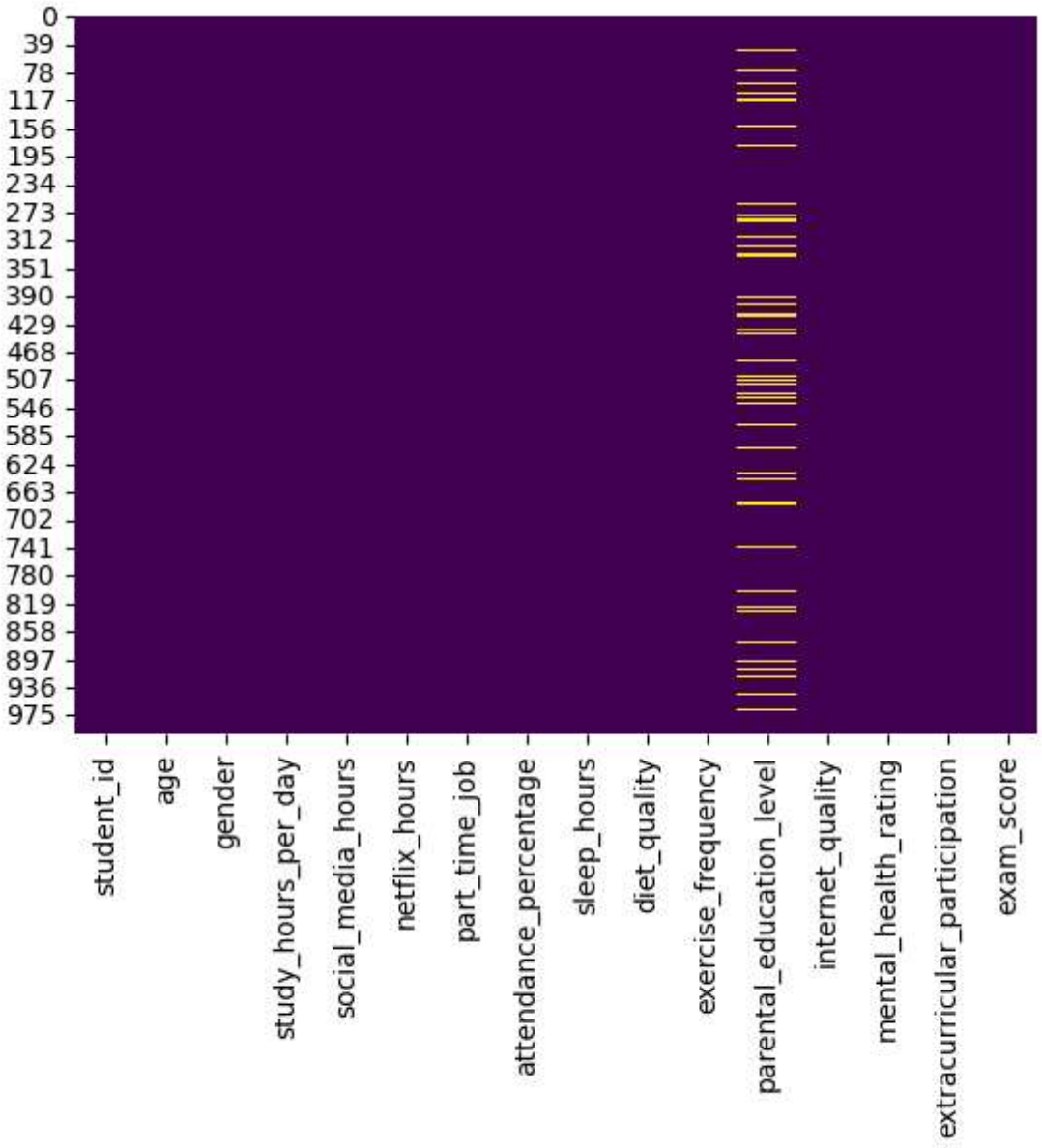
In [8]:
```
# Datatypes of every columns
data_copy.dtypes
```

Out[8]:
```
student_id                      object
age                              int64
gender                          object
study_hours_per_day            float64
social_media_hours             float64
netflix_hours                  float64
part_time_job                   object
attendance_percentage          float64
sleep_hours                    float64
diet_quality                    object
exercise_frequency               int64
parental_education_level        object
internet_quality                object
mental_health_rating             int64
extracurricular_participation   object
exam_score                     float64
dtype: object
```

In [9]:
```
# Check for the null values in every colums
data_copy.isna().sum()
```

Out[9]:
```
student_id                      0
age                             0
gender                          0
study_hours_per_day             0
social_media_hours              0
netflix_hours                   0
part_time_job                   0
attendance_percentage           0
sleep_hours                     0
diet_quality                    0
exercise_frequency              0
parental_education_level       91
internet_quality                0
mental_health_rating            0
extracurricular_participation   0
exam_score                      0
dtype: int64
```

In [10]:
```
# plot to check the null values places in the columns
sns.heatmap(data_copy.isnull(), cbar = False, cmap = 'viridis')

# Show the plot
plt.show()
```




In [11]:
```
# Data Cleaning
data_copy['parental_education_level'].mode()[0] # Checking the most repeated val
```

Out[11]: 'High School'

In [12]:
```
# Replace the null values with the most repeated values
data_copy['parental_education_level'] = data_copy['parental_education_level'].fi
```

In [13]:
```
# Again check the null values
data_copy.isna().sum()
```

Out[13]:
```
student_id                      0
age                             0
gender                          0
study_hours_per_day             0
social_media_hours              0
netflix_hours                   0
part_time_job                   0
attendance_percentage           0
sleep_hours                     0
diet_quality                    0
exercise_frequency              0
parental_education_level        0
internet_quality                0
mental_health_rating            0
extracurricular_participation   0
exam_score                      0
dtype: int64
```

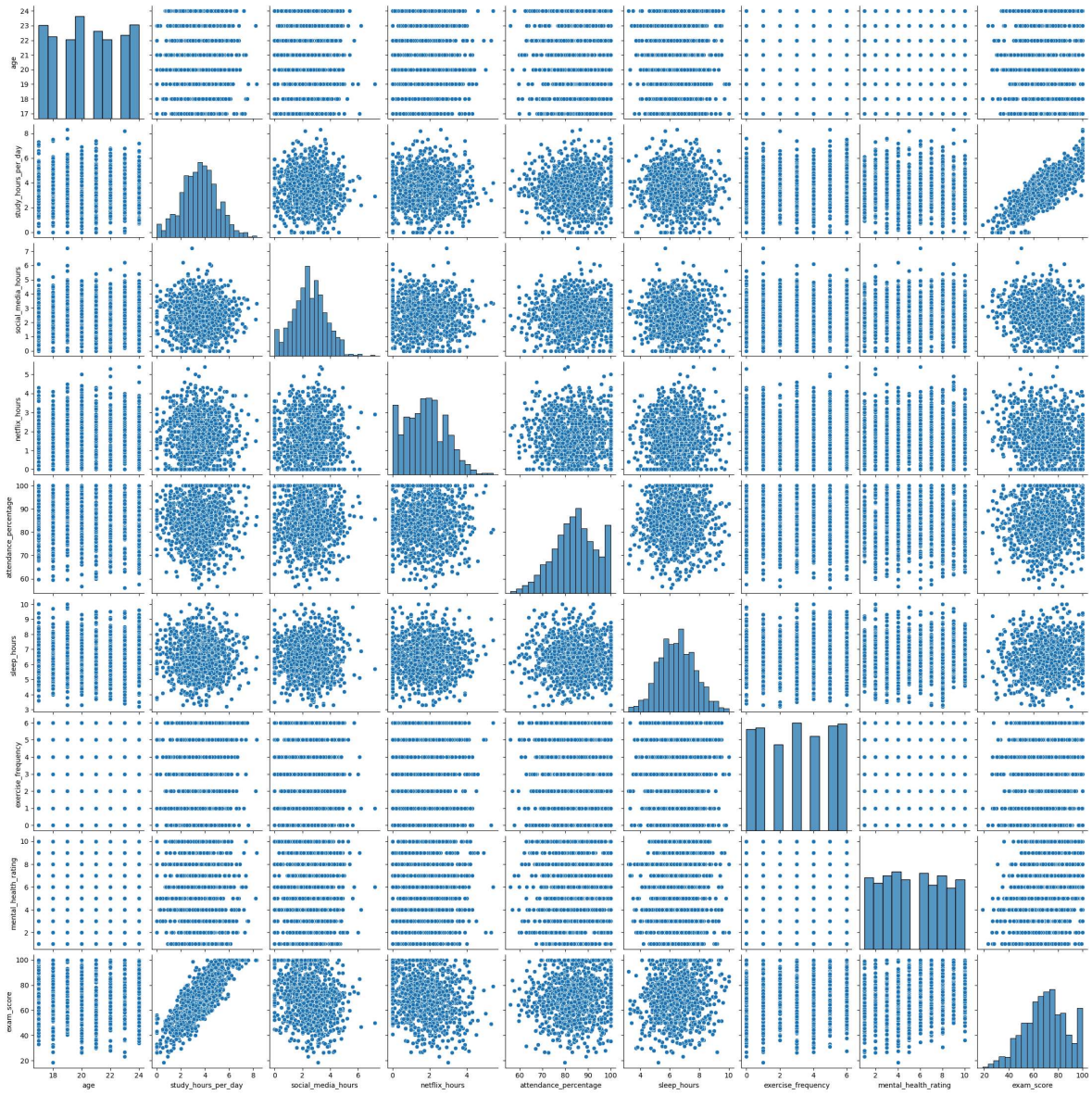
In [14]:
```
# describe the dataset
data_copy.describe(include='all')
```

Out[14]:

| | student_id | age | gender | study_hours_per_day | social_media_hours | netflix |
|---|---|---|---|---|---|---|
| **count** | 1000 | 1000.0000 | 1000 | 1000.00000 | 1000.000000 | 1000.( |
| **unique** | 1000 | NaN | 3 | NaN | NaN | |
| **top** | S1000 | NaN | Female | NaN | NaN | |
| **freq** | 1 | NaN | 481 | NaN | NaN | |
| **mean** | NaN | 20.4980 | NaN | 3.55010 | 2.505500 | 1.8 |
| **std** | NaN | 2.3081 | NaN | 1.46889 | 1.172422 | 1.( |
| **min** | NaN | 17.0000 | NaN | 0.00000 | 0.000000 | 0.( |
| **25%** | NaN | 18.7500 | NaN | 2.60000 | 1.700000 | 1.( |
| **50%** | NaN | 20.0000 | NaN | 3.50000 | 2.500000 | 1.8 |
| **75%** | NaN | 23.0000 | NaN | 4.50000 | 3.300000 | 2.5 |
| **max** | NaN | 24.0000 | NaN | 8.30000 | 7.200000 | 5.4 |

In [15]:
```
# Create a pairplot to show the linear relation
sns.pairplot(data_copy)
```

Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x22aef783650>
```




# Feature Engineering

Feature Engineering is the process of transforming raw data into a format that is better suited for modeling. It includes creating new features, modifying existing ones, encoding categorical variables, handling missing values, and scaling numeric data to improve model performance.

✅ Purpose:

Make data machine-readable

Highlight important patterns

Improve the predictive power of models

In [16]:
```
# Creating the list of Categorical columns
categorical_column = ['gender','part_time_job','diet_quality','parental_educatio
                      'internet_quality','extracurricular_participation']
```

In [17]:
```
# Encode the categorical data with dummy variables
encoded_data = pd.get_dummies(data_copy, columns = categorical_column, drop_firs
```

In [18]:
```
# Print the first five rows of the encoded data
encoded_data.head()
```

Out[18]:

| | student_id | age | study_hours_per_day | social_media_hours | netflix_hours | attendance_ |
|---|---|---|---|---|---|---|
| **0** | S1000 | 23 | 0.0 | 1.2 | 1.1 | |
| **1** | S1001 | 20 | 6.9 | 2.8 | 2.3 | |
| **2** | S1002 | 21 | 1.4 | 3.1 | 1.3 | |
| **3** | S1003 | 23 | 1.0 | 3.9 | 1.0 | |
| **4** | S1004 | 19 | 5.0 | 4.4 | 0.5 | |

In [19]:
```
# Total rows and columns in the encoded data
encoded_data.shape
```

Out[19]:
```
(1000, 20)
```

# Machine Learning Models

Machine Learning Models are algorithms that learn patterns from historical data to make predictions or classifications on new, unseen data. In this project, Linear Regression and Random Forest Regressor models are built to predict students' exam scores based on multiple lifestyle features.

✅ Purpose:

Create predictive systems that generalize well

Capture complex patterns that may not be obvious through simple statistics

Compare different models to find the best performing one

In [20]:
```
# Setting X and Y variables
X = encoded_data.drop(columns = ['student_id','exam_score'])
y = encoded_data['exam_score']
```

In [21]:
```
# Set the Training and test data (Data size and Random state)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, rando
```

In [22]:
```
# Scale the data
scaler = StandardScaler()
```

In [23]:
```
# Scale the data
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [24]:
```
# Deploy regression model
lr_model = LinearRegression()
```

In [25]:
```
lr_model.fit(X_train_scaled, y_train)
y_pred_lr = lr_model.predict(X_test_scaled)
```

In [26]:
```
print("Linear Regression Results:")
print("R² Score:", r2_score(y_test, y_pred_lr))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_lr)))
```

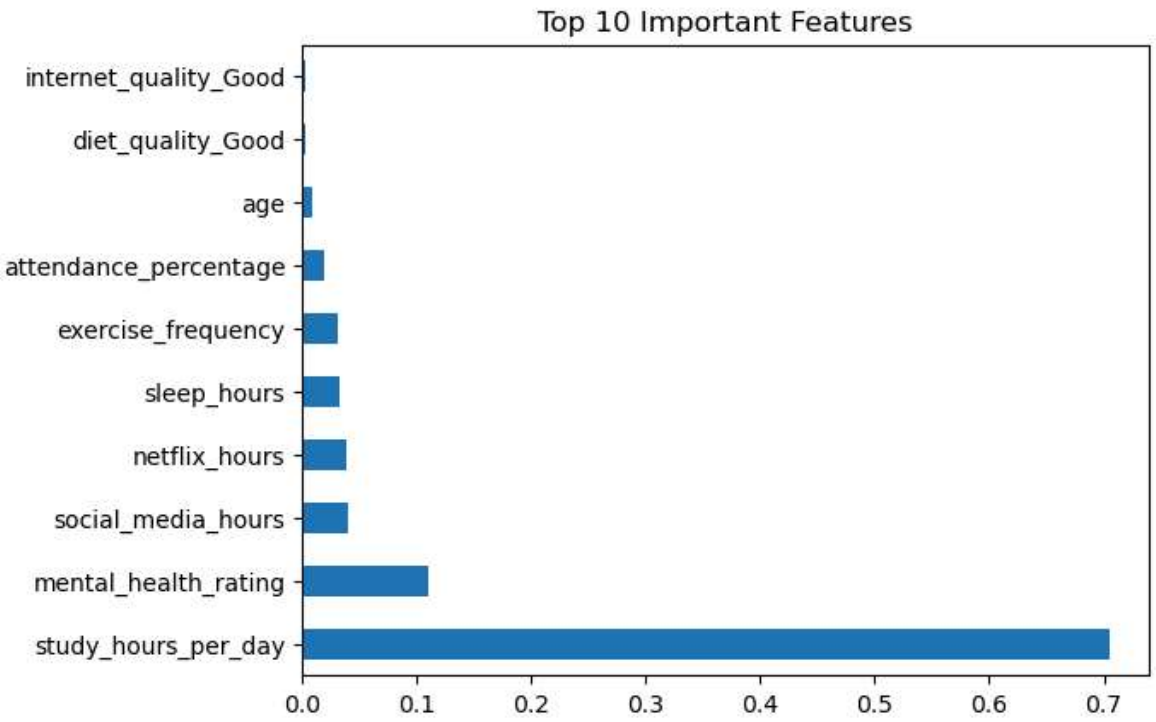
```
Linear Regression Results:
R² Score: 0.8880203736185235
RMSE: 5.54465546278495
```

In [27]:
```
rf_model = RandomForestRegressor(n_estimators = 100, random_state = 4)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
```

In [28]:
```
print("Random Forest Results:")
print("R² Score:", r2_score(y_test, y_pred_rf))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_rf)))
```

```
Random Forest Results:
R² Score: 0.8432848499277541
RMSE: 6.559346137001766
```

In [29]:
```
# Getting most important features
feat_imp = pd.Series(rf_model.feature_importances_, index=X.columns)
feat_imp.nlargest(10).plot(kind='barh')
plt.title("Top 10 Important Features")
plt.show()
```




# Statistical Modeling

Statistical Modeling is the process of creating mathematical representations of real-world data relationships using formulas and statistics. In this project, Ordinary Least Squares (OLS) Regression is used to model the linear relationship between study_hours_per_day and exam_score.

✅ Purpose:

Quantify and understand relationships between variables

Check assumptions like linearity and normality

Provide interpretable results (e.g., how much exam scores affect study hours)

In [30]:
```
data_copy['gender'].value_counts()
```

Out[30]:
```
gender
Female    481
Male      477
Other      42
Name: count, dtype: int64
```

In [31]:
```
data_copy.groupby(['gender','part_time_job'])['study_hours_per_day'].mean()
```

Out[31]:
```
gender  part_time_job
Female  No               3.607792
        Yes              3.487500
Male    No               3.517663
        Yes              3.487156
Other   No               3.778125
        Yes              3.080000
Name: study_hours_per_day, dtype: float64
```

In [32]:
```
# Ordinary Least Squares (OLS)
ols_formula = "study_hours_per_day ~ exam_score"
```

In [33]:
```
OLS = ols(formula=ols_formula, data = data_copy)
```

In [34]:
```
model = OLS.fit()
```

In [35]:
```
model_results = model.summary()
model_results
```

Out[35]:

OLS Regression Results

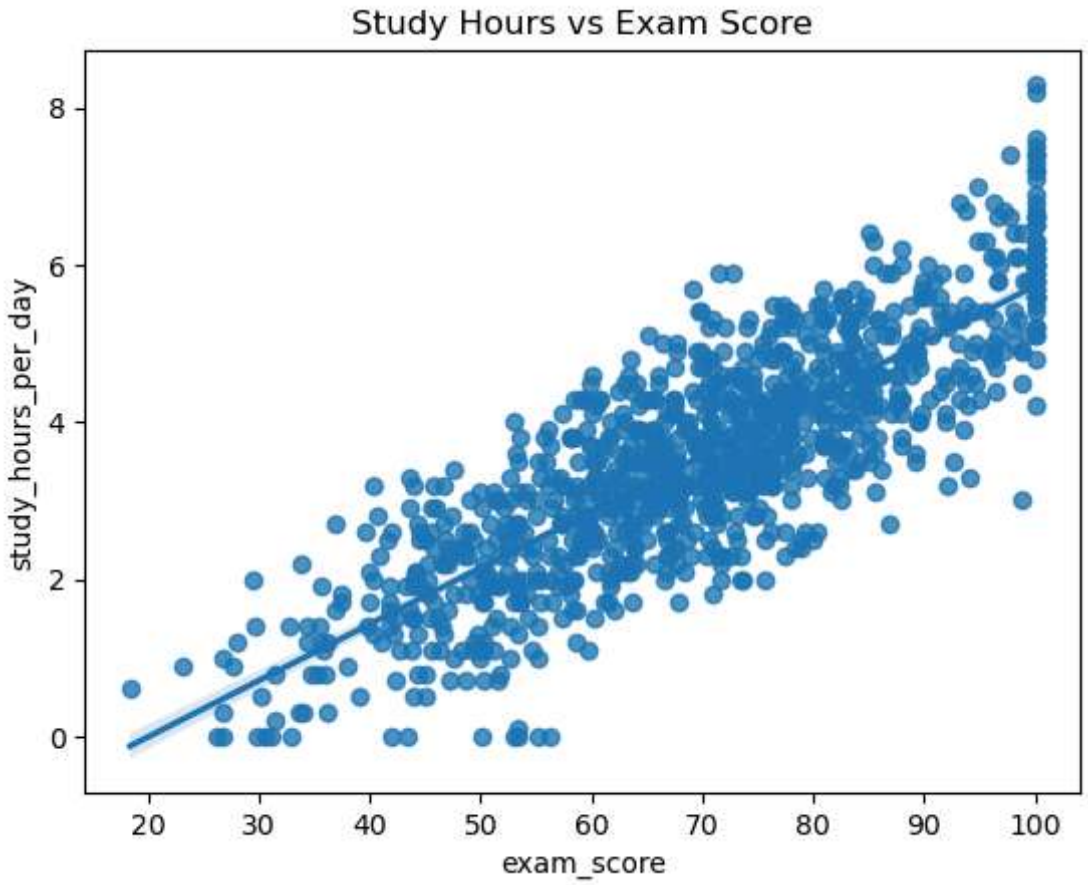
| | | | |
|---|---|---|---|
| **Dep. Variable:** | study_hours_per_day | **R-squared:** | 0.681 |
| **Model:** | OLS | **Adj. R-squared:** | 0.681 |
| **Method:** | Least Squares | **F-statistic:** | 2134. |
| **Date:** | Sat, 26 Apr 2025 | **Prob (F-statistic):** | 4.60e-250 |
| **Time:** | 21:00:11 | **Log-Likelihood:** | -1231.2 |
| **No. Observations:** | 1000 | **AIC:** | 2466. |
| **Df Residuals:** | 998 | **BIC:** | 2476. |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | -1.4467 | 0.111 | -12.997 | 0.000 | -1.665 | -1.228 |
| **exam_score** | 0.0718 | 0.002 | 46.191 | 0.000 | 0.069 | 0.075 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 0.516 | **Durbin-Watson:** | 2.063 |
| **Prob(Omnibus):** | 0.773 | **Jarque-Bera (JB):** | 0.594 |
| **Skew:** | -0.045 | **Prob(JB):** | 0.743 |
| **Kurtosis:** | 2.921 | **Cond. No.** | 304. |

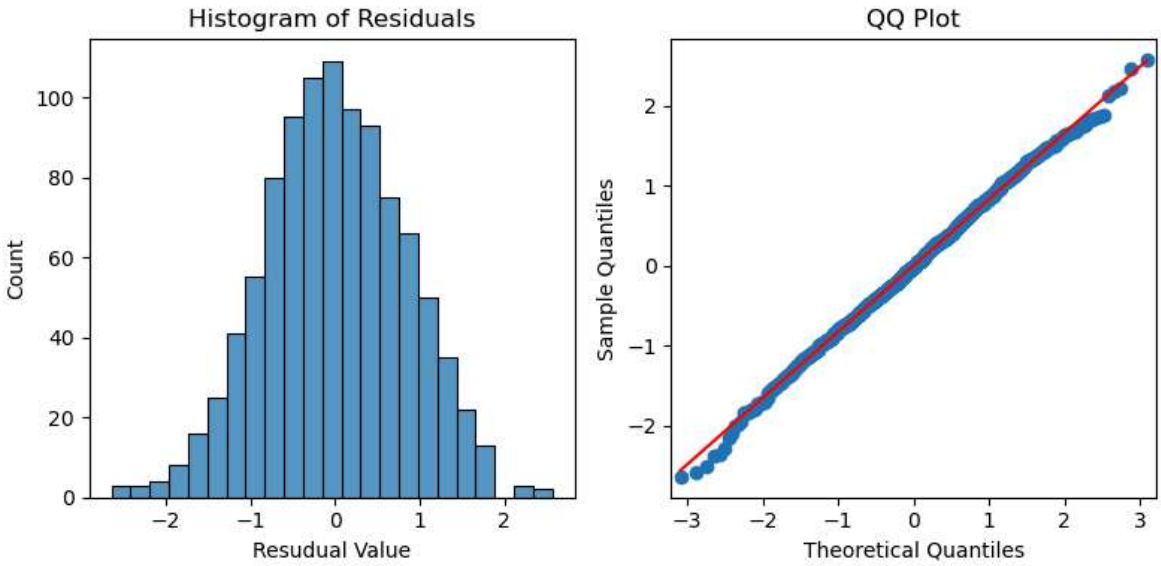
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [36]:
```
# Regression plot
sns.regplot(x='exam_score', y='study_hours_per_day', data=data_copy)
plt.title("Study Hours vs Exam Score")
plt.show()
```

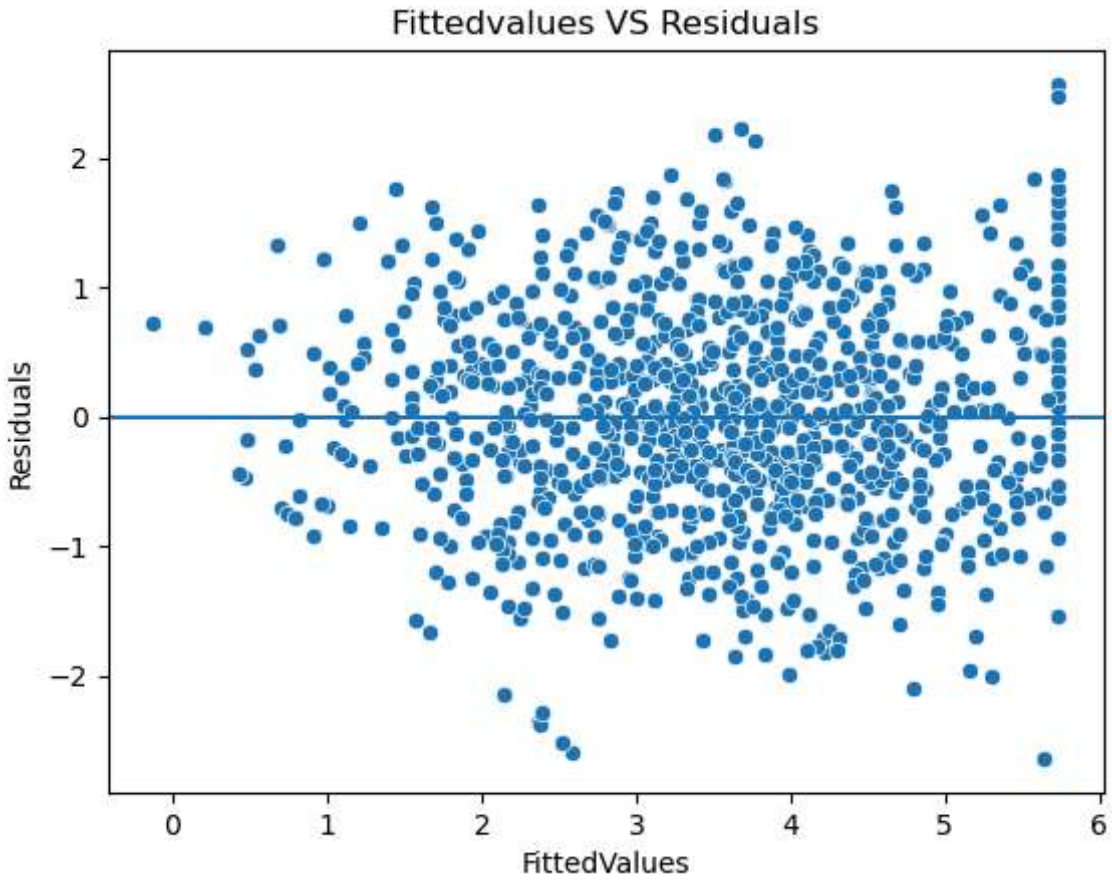



In [37]:
```
residuals = model.resid
```

In [38]:
```
# Creating a sub-plot
fig, axes = plt.subplots(1, 2, figsize = (8,4))

# Plotting a Histogram
sns.histplot(residuals, ax = axes[0]);

# Setting Title and X label
axes[0].set_title("Histogram of Residuals")
axes[0].set_xlabel("Resudual Value")

# QQ plot
sm.qqplot(residuals, line = 's', ax = axes[1]);

# QQ plot in axes 1
axes[1].set_title("QQ Plot")

plt.tight_layout()

# Plotting
plt.show()
```




In [39]:
```
# Plotting a scatter plot
fig = sns.scatterplot(x = model.fittedvalues, y = model.resid)

# Fixing X label
fig.set_xlabel("FittedValues")

# Fixing Y label
fig.set_ylabel("Residuals")

# Fixing Title
fig.set_title("Fittedvalues VS Residuals")

# Setting Axhline
fig.axhline(0)

# Plotting
plt.show()
```




In [40]:
```
model_results
```

Out[40]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | study_hours_per_day | **R-squared:** | 0.681 |
| **Model:** | OLS | **Adj. R-squared:** | 0.681 |
| **Method:** | Least Squares | **F-statistic:** | 2134. |
| **Date:** | Sat, 26 Apr 2025 | **Prob (F-statistic):** | 4.60e-250 |
| **Time:** | 21:00:11 | **Log-Likelihood:** | -1231.2 |
| **No. Observations:** | 1000 | **AIC:** | 2466. |
| **Df Residuals:** | 998 | **BIC:** | 2476. |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | -1.4467 | 0.111 | -12.997 | 0.000 | -1.665 | -1.228 |
| **exam_score** | 0.0718 | 0.002 | 46.191 | 0.000 | 0.069 | 0.075 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 0.516 | **Durbin-Watson:** | 2.063 |
| **Prob(Omnibus):** | 0.773 | **Jarque-Bera (JB):** | 0.594 |
| **Skew:** | -0.045 | **Prob(JB):** | 0.743 |
| **Kurtosis:** | 2.921 | **Cond. No.** | 304. |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

# Model Evaluation

Model Evaluation is the process of measuring how well a model performs on unseen or test data. It involves using metrics like $R^2$ Score (how much variance is explained) and Root Mean Squared Error (RMSE) (average prediction error) to judge the model's accuracy.

✅ Purpose:

Ensure that the model is not underfitting or overfitting

Quantify the model's predictive power

Compare multiple models based on performance metrics

In [41]:
```
sm.stats.anova_lm(model, typ = 2)
```

Out[41]:

| | sum_sq | df | F | PR(>F) |
|---|---|---|---|---|
| **exam_score** | 1468.562392 | 1.0 | 2133.626029 | 4.595701e-250 |
| **Residual** | 686.917598 | 998.0 | NaN | NaN |

In [42]:
```
tukey_result = pairwise_tukeyhsd(
    endog = data_copy["study_hours_per_day"],
    groups = data_copy["parental_education_level"],  # must be categorical
    alpha = 0.05
)

print(tukey_result)
```

```
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
=============================================================
   group1      group2   meandiff p-adj   lower  upper  reject
-------------------------------------------------------------
   Bachelor High School   0.0673  0.791 -0.1749 0.3095  False
   Bachelor      Master   0.0745 0.8522   -0.25  0.399  False
High School      Master   0.0072 0.9984 -0.3026 0.3169  False
-------------------------------------------------------------
```

In [43]:
```
tukey_result.summary()
```

Out[43]:

Multiple Comparison of Means - Tukey HSD, FWER=0.05

| group1 | group2 | meandiff | p-adj | lower | upper | reject |
|---|---|---|---|---|---|---|
| Bachelor | High School | 0.0673 | 0.791 | -0.1749 | 0.3095 | False |
| Bachelor | Master | 0.0745 | 0.8522 | -0.25 | 0.399 | False |
| High School | Master | 0.0072 | 0.9984 | -0.3026 | 0.3169 | False |