In [ ]:

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error, accuracy_score
import statsmodels.api as sm
```

In [14]:
```
std_prfmnce =pd.read_csv("C:/Users/ardra/Downloads/archive (14)/StudentPerforman
std_prfmnce
```

Out[14]:

	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	
...	...	...	...	...	
6602	25	69	High	Medium	
6603	23	76	High	Medium	
6604	20	90	Medium	Low	
6605	10	86	High	High	
6606	15	67	Medium	Low	

6607 rows × 20 columns

In [15]:
```
print(std_prfmnce.columns)
```

```
Index(['Hours_Studied', 'Attendance', 'Parental_Involvement',
       'Access_to_Resources', 'Extracurricular_Activities', 'Sleep_Hours',
       'Previous_Scores', 'Motivation_Level', 'Internet_Access',
       'Tutoring_Sessions', 'Family_Income', 'Teacher_Quality', 'School_Type',
       'Peer_Influence', 'Physical_Activity', 'Learning_Disabilities',
       'Parental_Education_Level', 'Distance_from_Home', 'Gender',
       'Exam_Score'],
      dtype='object')
```

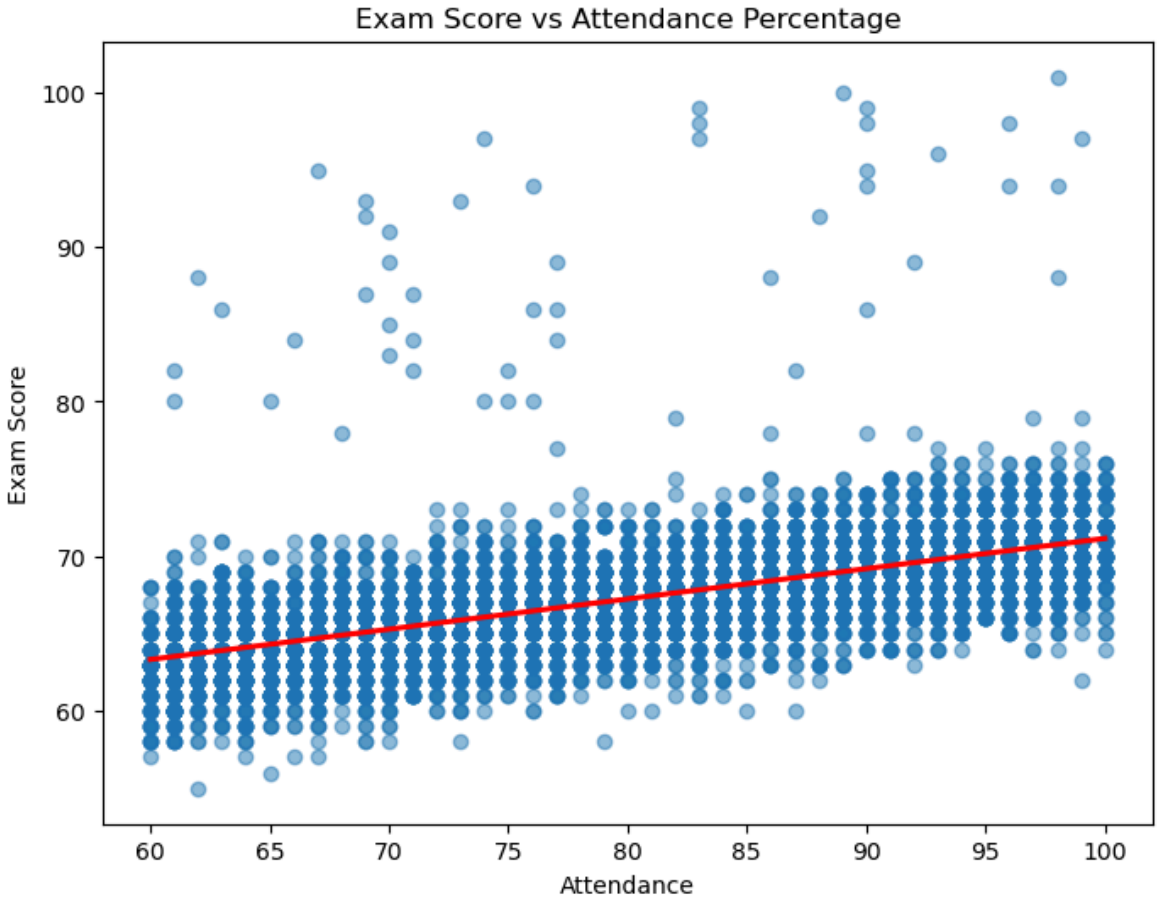
In [16]:
```
print("\nDescriptive Statistics:")
print(std_prfmnce.describe())
```

```
Descriptive Statistics:
       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
```

In [17]:
```
plt.figure(figsize=(8, 6))
sns.regplot(x=std_prfmnce['Attendance'], y=std_prfmnce['Exam_Score'], scatter_kw
plt.xlabel("Attendance")
plt.ylabel("Exam Score")
plt.title("Exam Score vs Attendance Percentage")
plt.show()

print("The correlation between the two variables is")
print(std_prfmnce['Attendance'].corr(std_prfmnce['Exam_Score']))
```




```
The correlation between the two variables is
0.5810718633120644
```

In [20]:
```
# Prepare Data for Regression
X = std_prfmnce[['Attendance']]  # Independent variable
Y = std_prfmnce['Exam_Score']  # Dependent variable

X = sm.add_constant(X)
X.head()
model = sm.OLS(Y,X, missing='drop')
model_result=model.fit()
model_result.summary()
```

Out[20]:

OLS Regression Results

Dep. Variable:	Exam_Score	**R-squared:**	0.338
Model:	OLS	**Adj. R-squared:**	0.338
Method:	Least Squares	**F-statistic:**	3367.
Date:	Sat, 15 Mar 2025	**Prob (F-statistic):**	0.00
Time:	20:42:32	**Log-Likelihood:**	-16989.
No. Observations:	6607	**AIC:**	3.398e+04
Df Residuals:	6605	**BIC:**	3.400e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	51.5786	0.273	189.191	0.000	51.044	52.113
Attendance	0.1958	0.003	58.026	0.000	0.189	0.202

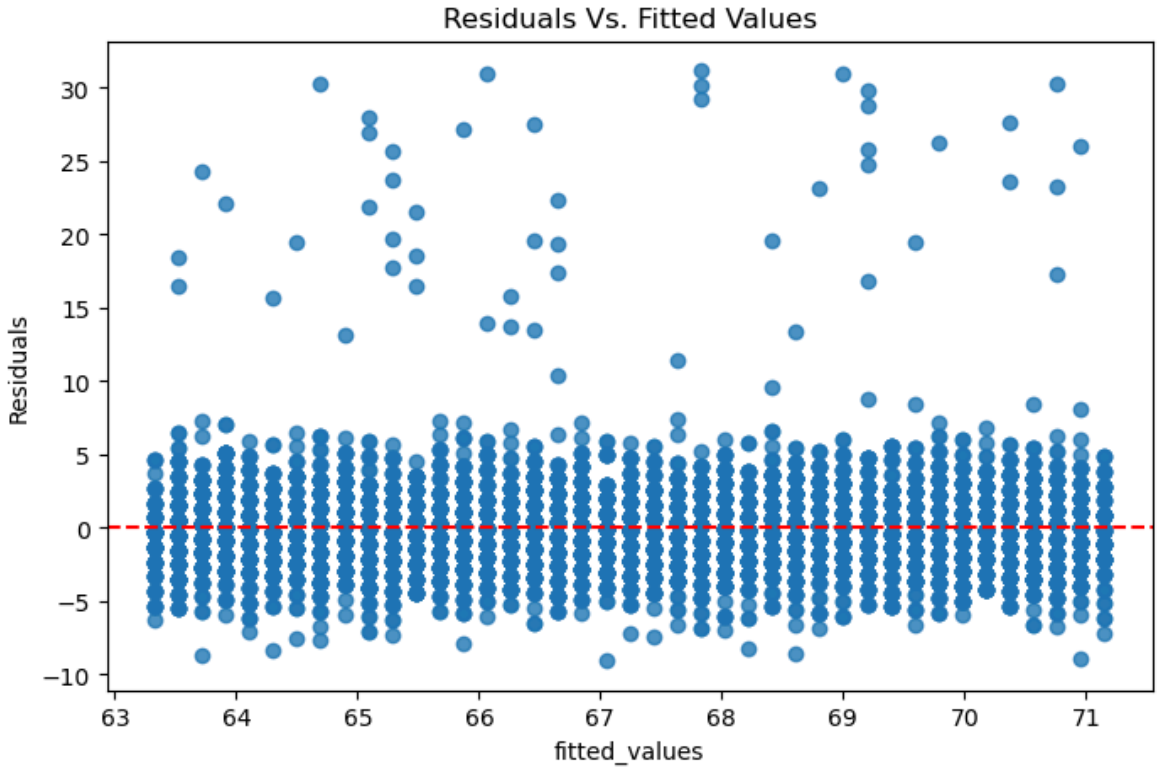
Omnibus:	4752.483	**Durbin-Watson:**	2.023
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	163501.063
Skew:	3.038	**Prob(JB):**	0.00
Kurtosis:	26.601	**Cond. No.**	566.

Notes:

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

In [22]:
```
fitted_values = model_result.fittedvalues
residuals = model_result.resid

plt.figure(figsize=(8,5))
plt.scatter(fitted_values, residuals, alpha=0.8)
plt.axhline(y=0, color='r', linestyle='dashed')
plt.xlabel('fitted_values')
plt.ylabel('Residuals')
plt.title('Residuals Vs. Fitted Values')
plt.show()
```



In [24]:
```
import statsmodels.stats.diagnostic as smd
import numpy as np

# Get residuals and independent variables (exog)
residuals = model_result.resid
exog = model_result.model.exog  # Independent variables (including constant)

# Perform the Breusch-Pagan test
bp_test = smd.het_breuschpagan(residuals, exog)
p_value = bp_test[1]  # Extract the p-value

print(f"Breusch-Pagan test p-value: {p_value}")

# Interpretation
if p_value < 0.05:
    print("Heteroscedasticity detected! Consider transformation or robust standa
else:
    print("No significant heteroscedasticity detected.")
```

```
Breusch-Pagan test p-value: 0.8057839143058365
No significant heteroscedasticity detected.
```

In [ ]: