In [3]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

In [4]:

```
df=pd.read_csv("Student_Performance.csv")
```

In [5]:

```
df.head()
```

Out[5]:

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	Yes	9	1	91.0
1	4	82	No	4	2	65.0
2	8	51	Yes	7	2	45.0
3	5	52	Yes	5	2	36.0
4	7	75	No	8	5	66.0

In [6]:

```
##check null values
df.isnull().sum()
```

Out[6]:

```
Hours Studied                       0
Previous Scores                     0
Extracurricular Activities          0
Sleep Hours                         0
Sample Question Papers Practiced    0
Performance Index                   0
dtype: int64
```

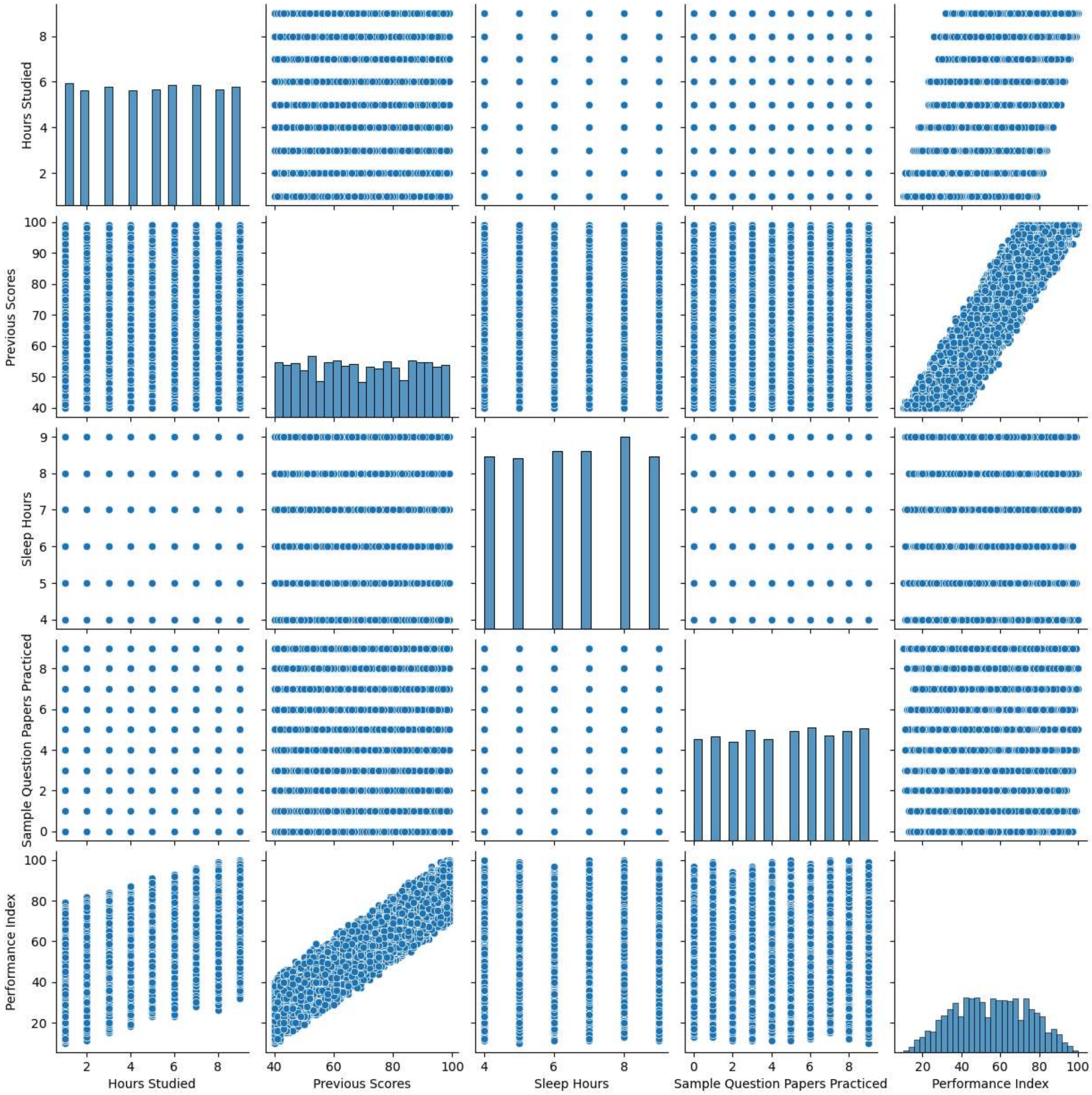
In [9]:

```
## Lets do some viualization
import seaborn as sns
sns.pairplot(df)
```

```
C:\Users\win 10\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure lay
out has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[9]:

```
<seaborn.axisgrid.PairGrid at 0x195ce978cd0>
```



In [94]:

```
df["Extracurricular Activities"]=df["Extracurricular Activities"].map({"Yes":1 , "No":2})
df
```

Out[94]:

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	1	9	1	91.0
1	4	82	2	4	2	65.0
2	8	51	1	7	2	45.0
3	5	52	1	5	2	36.0
4	7	75	2	8	5	66.0
...	...	...	...	...	...	...
9995	1	49	1	4	2	23.0
9996	7	64	1	8	5	58.0
9997	6	83	1	8	5	74.0
9998	9	97	1	7	0	95.0
9999	7	74	2	8	1	64.0

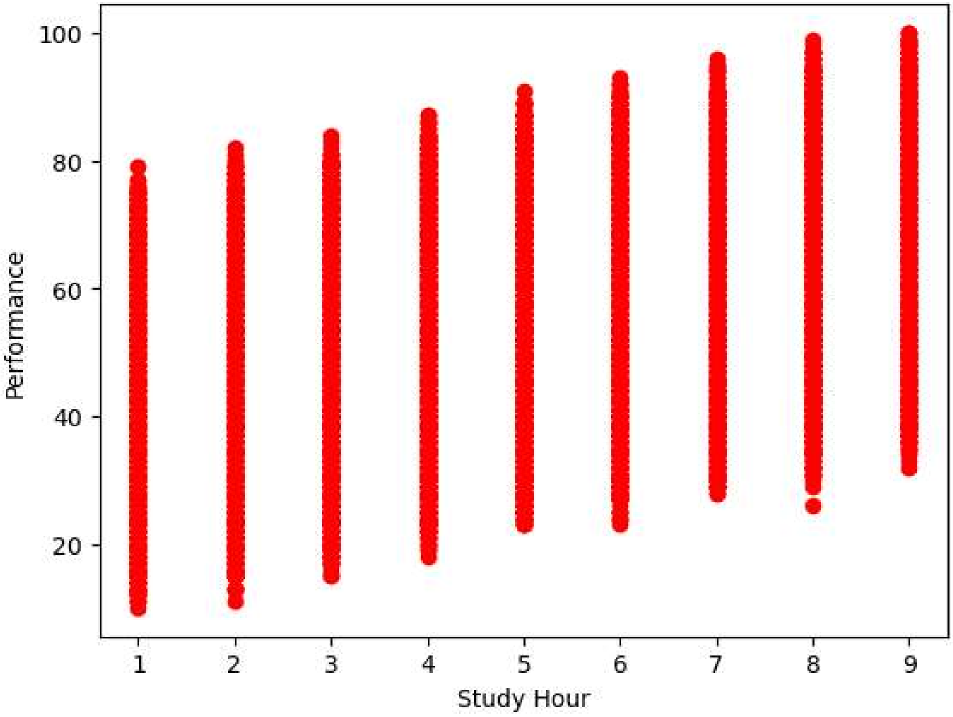
10000 rows × 6 columns

In [95]:

```
## Visualiza the datapoints more closely
plt.scatter(df['Hours Studied'],df['Performance Index'], color='r')
plt.xlabel("Study Hour")
plt.ylabel("Performance")
```

Out[95]:

```
Text(0, 0.5, 'Performance')
```



In [96]:

```
##independent and dependent features
X=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

In [97]:
```
X.head()
```

Out[97]:

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced
0	7	99	1	9	1
1	4	82	2	4	2
2	8	51	1	7	2
3	5	52	1	5	2
4	7	75	2	8	5

In [98]:
```
y
```

Out[98]:
```
0       91.0
1       65.0
2       45.0
3       36.0
4       66.0
        ...
9995    23.0
9996    58.0
9997    74.0
9998    95.0
9999    64.0
Name: Performance Index, Length: 10000, dtype: float64
```

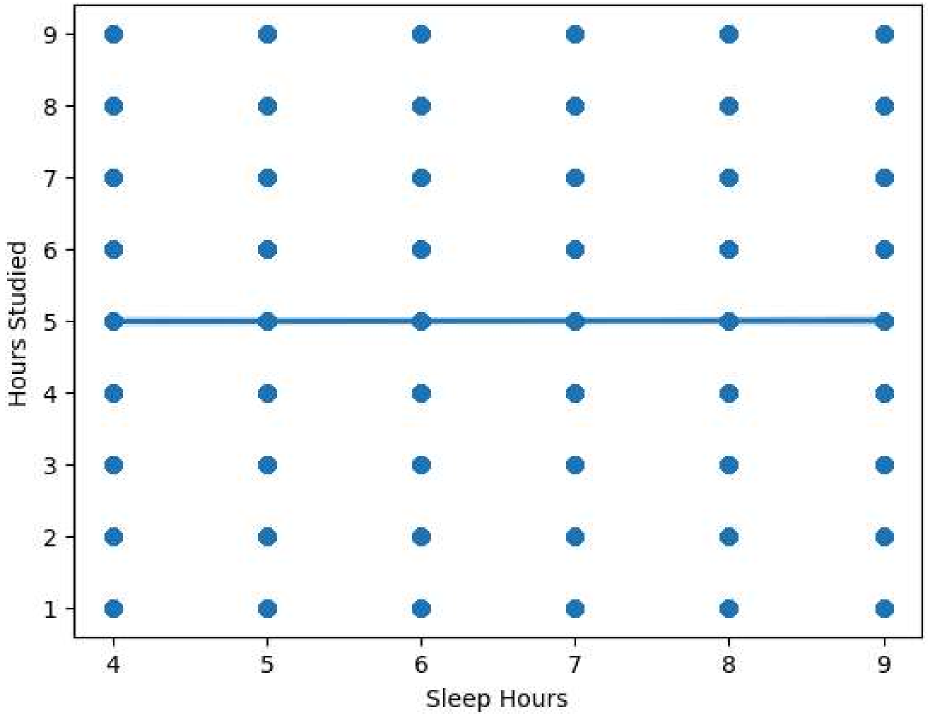
In [112]:
```
# train test split
from sklearn.model_selection import train_test_split
```

In [113]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=42)
```

In [114]:
```
import seaborn as sns
```

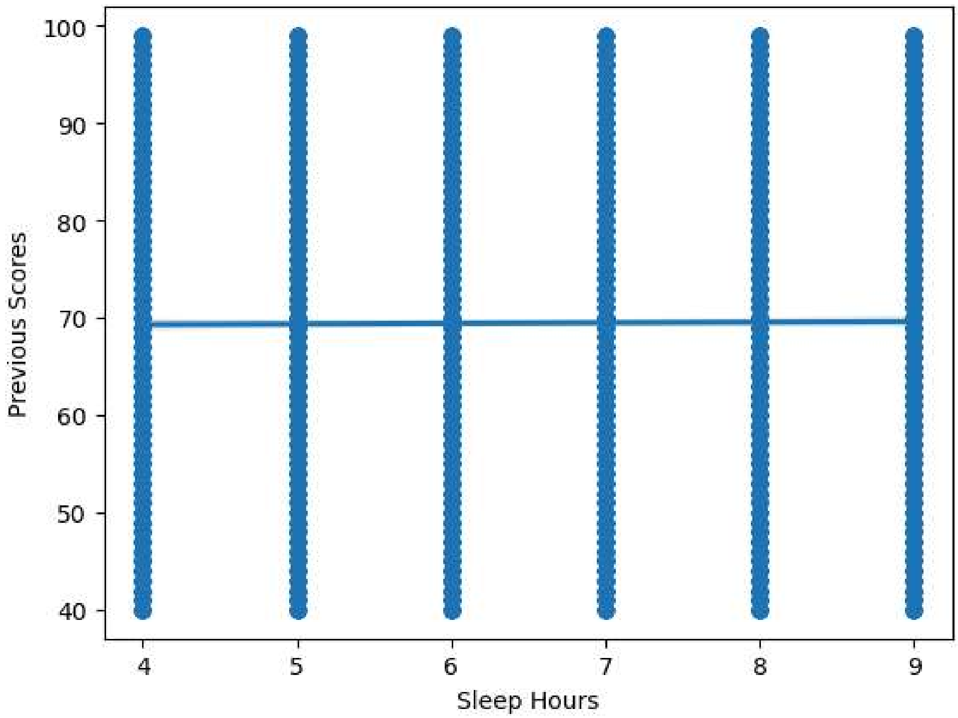
In [115]:
```
sns.regplot(x=df['Sleep Hours'],y=df['Hours Studied'])
```

Out[115]:
```
<Axes: xlabel='Sleep Hours', ylabel='Hours Studied'>
```



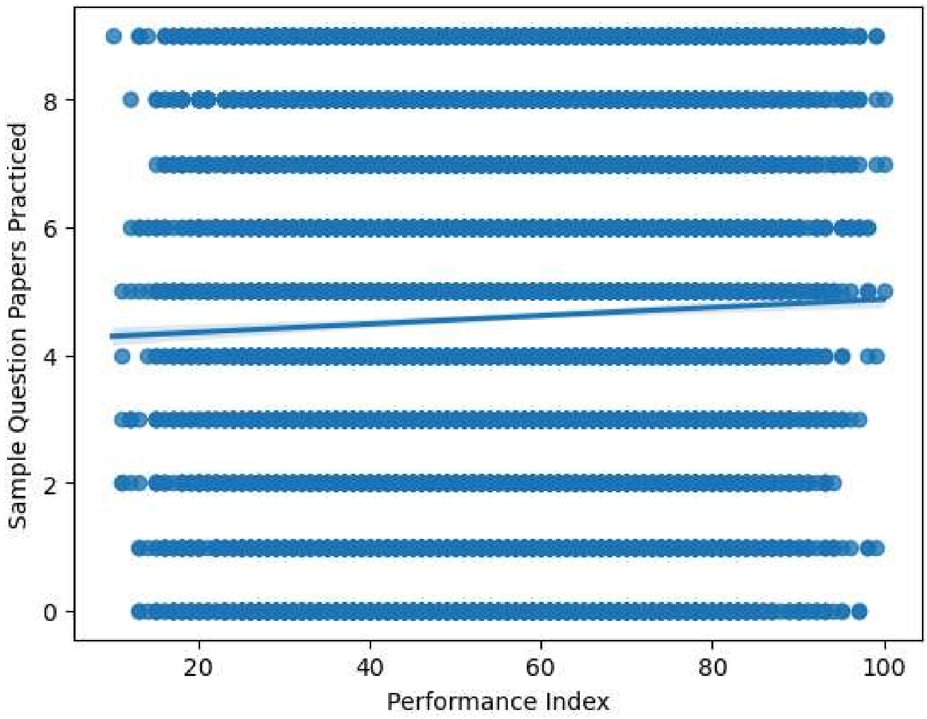
```
In [116]: sns.regplot(x=df['Sleep Hours'],y=df['Previous Scores'])
```

```
Out[116]: <Axes: xlabel='Sleep Hours', ylabel='Previous Scores'>
```



```
In [117]: sns.regplot(x=df['Performance Index'],y=df['Sample Question Papers Practiced'])
```

```
Out[117]: <Axes: xlabel='Performance Index', ylabel='Sample Question Papers Practiced'>
```



```
In [138]: x=df.drop("Performance Index",axis=1)
          y=df['Performance Index']
          train_X,test_X,train_Y,test_Y = train_test_split(x,y,test_size=0.2,shuffle=True)
```

In [139]:
```
from sklearn.linear_model import LinearRegression
```

In [140]:
```
regression=LinearRegression()
```

In [141]:
```
regression.fit(X_train,y_train)
```

Out[141]:
```
▾ LinearRegression
LinearRegression()
```

In [142]:
```
## cross validation
from sklearn.model_selection import cross_val_score
```

In [143]:
```
validation_score=cross_val_score(regression,X_train,y_train,scoring='neg_mean_squared_error',cv=3
```

In [144]:
```
np.mean(validation_score)
```

Out[144]:
```
-4.2027969495394295
```

In [145]:
```
#Prediction
y_pred=regression.predict(X_test)
```

In [146]:
```
y_pred
```

Out[146]:
```
array([54.73187888, 22.61211054, 47.90838844, ..., 68.07396952,
       53.68636805, 54.85816372])
```

In [147]:
```
## Performance Metrics
from sklearn.metrics import mean_absolute_error, mean_squared_error
mse=mean_squared_error(y_test,y_pred)
mae=mean_absolute_error(y_test,y_pred)
rmse=np.sqrt(mse)
print(mse)
print(mae)
print(rmse)
```

```
4.032544215419129
1.5975792091646137
2.0081195719924474
```

In [148]:
```
from sklearn.metrics import r2_score
score=r2_score(y_test,y_pred)
print(score)
```

```
0.9890550757439103
```

In [149]:
```
#display adjusted R-squared
print(1 - (1-score)*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1))
```

```
0.9890331332333728
```

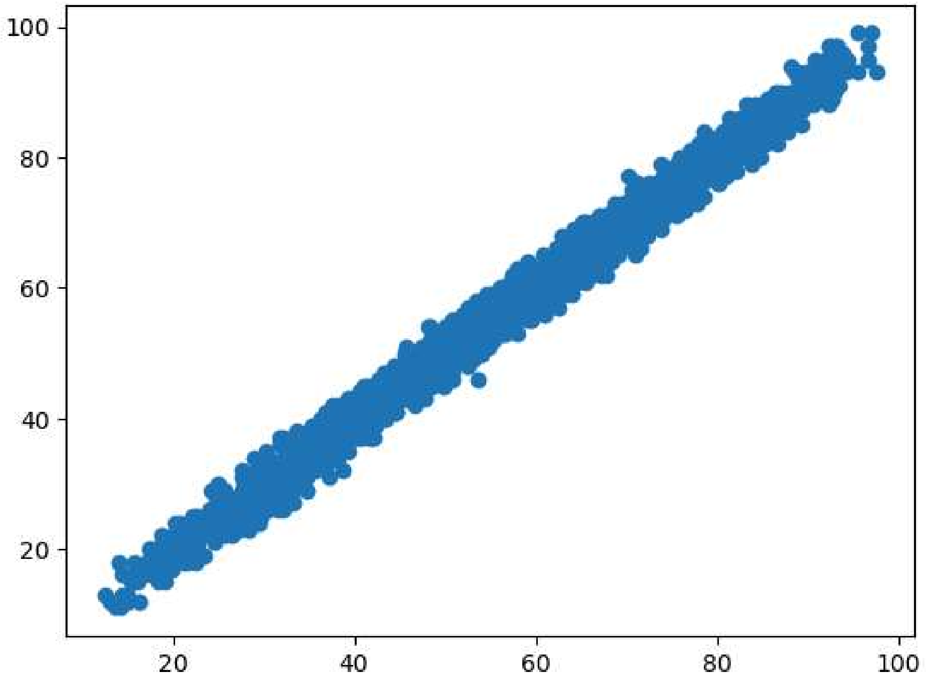
Assumptions

In [150]:

```
plt.scatter(y_pred,y_test)
```

Out[150]:

```
<matplotlib.collections.PathCollection at 0x195dc03da10>
```



In [151]:

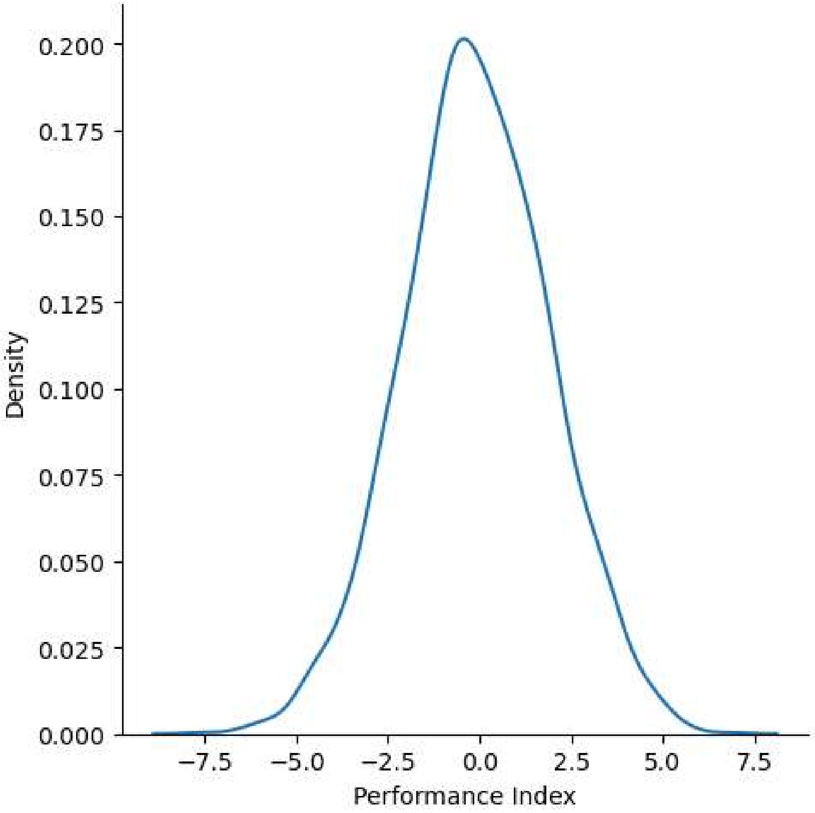
```
residuals=y_test-y_pred
print(residuals)
```

```
6252   -3.731879
4684   -2.612111
1731   -1.908388
4742   -3.301042
4521   -2.035815
          ...
4862    0.575641
7025   -1.000419
7647    1.926030
7161    2.313632
73     -2.858164
Name: Performance Index, Length: 2500, dtype: float64
```

In [152]:
```
## Plot this residuals
sns.displot(residuals,kind='kde')
```

```
C:\Users\win 10\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure lay
out has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[152]:
```
<seaborn.axisgrid.FacetGrid at 0x195da62fc10>
```

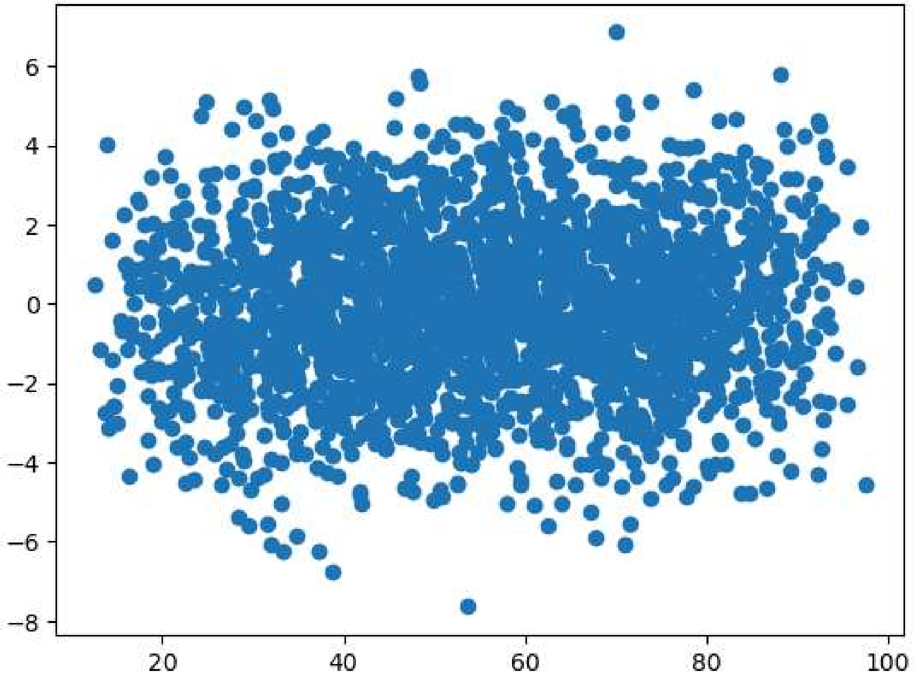


In [153]:

```
## scatter plot with respect to prediction and residuals
plt.scatter(y_pred,residuals)
```

Out[153]:

```
<matplotlib.collections.PathCollection at 0x195dc0f8390>
```



In [154]:

```
## OLS Linear Regression
import statsmodels.api as sm
model=sm.OLS(y_train,X_train).fit()
```

In [135]:
```
model.summary()
```

Out[135]:

OLS Regression Results

Dep. Variable:	Performance Index	**R-squared (uncentered):**	0.992
Model:	OLS	**Adj. R-squared (uncentered):**	0.992
Method:	Least Squares	**F-statistic:**	1.890e+05
Date:	Mon, 02 Jun 2025	**Prob (F-statistic):**	0.00
Time:	22:44:13	**Log-Likelihood:**	-23011.
No. Observations:	7500	**AIC:**	4.603e+04
Df Residuals:	7495	**BIC:**	4.607e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Hours Studied	2.3392	0.022	105.210	0.000	2.296	2.383
Previous Scores	0.8540	0.003	304.681	0.000	0.849	0.860
Extracurricular Activities	-4.7304	0.108	-43.658	0.000	-4.943	-4.518
Sleep Hours	-1.0713	0.030	-36.251	0.000	-1.129	-1.013
Sample Question Papers Practiced	-0.1889	0.020	-9.235	0.000	-0.229	-0.149

Omnibus:	18.746	**Durbin-Watson:**	1.929
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	15.050
Skew:	-0.003	**Prob(JB):**	0.000539
Kurtosis:	2.781	**Cond. No.**	131.

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [137]:
```
print(regression.coef_)
```

```
[ 2.85492123  1.01637916 -0.58370931  0.47688351  0.19092346]
```