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

```
df  = pd.read_csv('/content/Student_Performance.csv')
```

```
df
```

	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
...	...	...	...	...	...	...
9995	1	49	Yes	4	2	23.0
9996	7	64	Yes	8	5	58.0
9997	6	83	Yes	8	5	74.0
9998	9	97	Yes	7	0	95.0
9999	7	74	No	8	1	64.0

10000 rows × 6 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.columns
```

```
Index(['Hours Studied', 'Previous Scores', 'Extracurricular Activities',
       'Sleep Hours', 'Sample Question Papers Practiced', 'Performance Index'],
      dtype='object')
```

```
# Drop unnecessary columns
# df.drop(['race/ethnicity', 'parental level of education', 'lunch', 'test preparation course'], axis=1, inplace=True)
```

```
df.head()
```

	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

Next steps: Generate code with df | View recommended plots | New interactive sheet

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

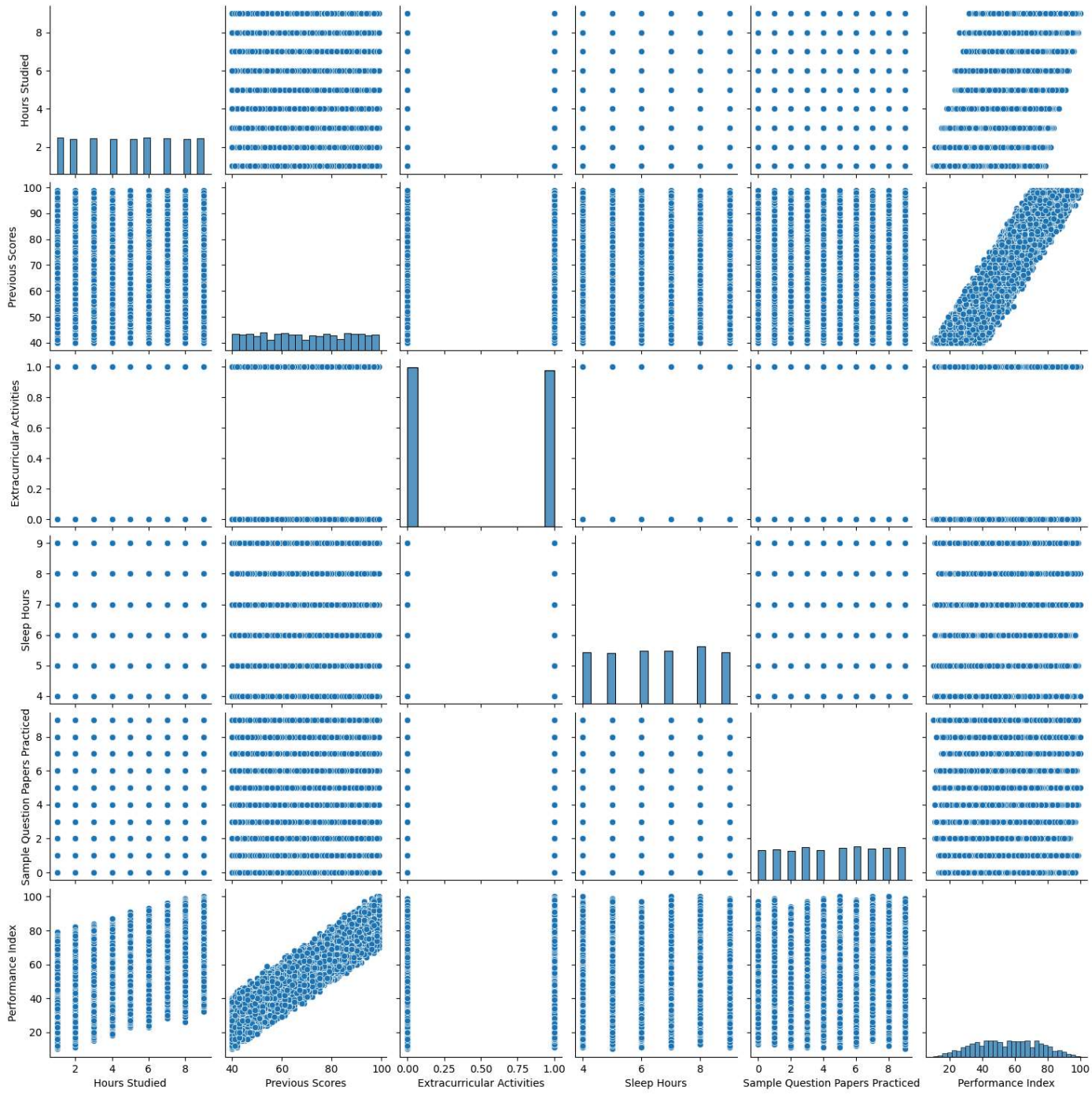
	0
Hours Studied	0
Previous Scores	0
Extracurricular Activities	0
Sleep Hours	0
Sample Question Papers Practiced	0
Performance Index	0

```
# Convert boolean values to integers
# Mapping 'Yes' to 1 and 'No' to 0
df['Extracurricular Activities'] = df['Extracurricular Activities'].map({'Yes': 1, 'No': 0})


df.drop('Extracurricular Activities Encoded', axis=1, inplace=True)


import seaborn as sns
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x78400fdbb940>
```




```
df.corr()
```

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
Hours Studied	1.000000	-0.012390	0.003873	0.001245	0.017463	0.373730
Previous Scores	-0.012390	1.000000	0.008369	0.005944	0.007888	0.915189
Extracurricular Activities	0.003873	0.008369	1.000000	-0.023284	0.013103	0.024525
Sleep Hours	0.001245	0.005944	-0.023284	1.000000	0.003990	0.048106
Sample Question Papers Practiced	0.017463	0.007888	0.013103	0.003990	1.000000	0.043268
Performance Index	0.373730	0.915189	0.024525	0.048106	0.043268	1.000000

```
df.describe()
```

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	4.992900	69.445700	0.494800	6.530600	4.583300	55.224800
std	2.589309	17.343152	0.499998	1.695863	2.867348	19.212558
min	1.000000	40.000000	0.000000	4.000000	0.000000	10.000000
25%	3.000000	54.000000	0.000000	5.000000	2.000000	40.000000
50%	5.000000	69.000000	0.000000	7.000000	5.000000	55.000000
75%	7.000000	85.000000	1.000000	8.000000	7.000000	71.000000

```
## V;isuaize the datapoints more closely
plt.scatter(df['Previous Scores'], df['Performance Index'])
plt.xlabel('Previous Scores')
plt.ylabel('Performance Index')
plt.title('Previous Scores vs Performance Index')
```

```
Text(0.5, 1.0, 'Previous Scores vs Performance Index')
```


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

```
X.head()
```

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

Next steps: Generate code with X View recommended plots New interactive sheet

y

	Performance Index
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

10000 rows × 1 columns

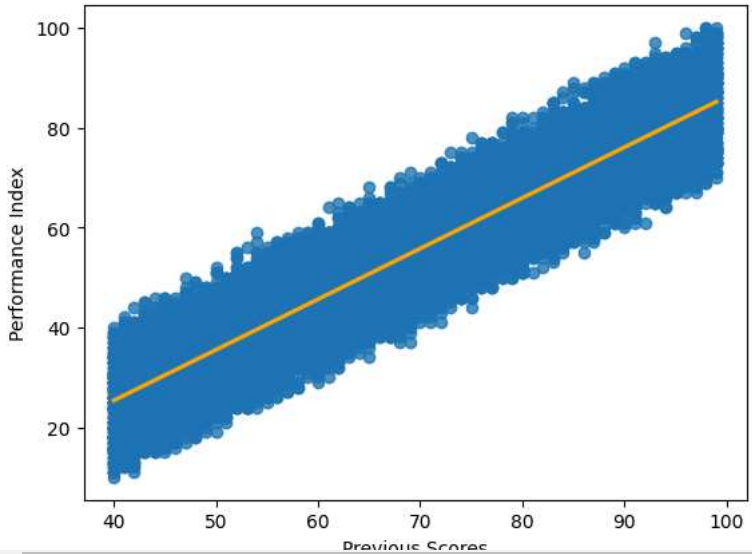
```
# Train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)


import seaborn as sns
import matplotlib.pyplot as plt

# Example data
# sns.regplot(x='Previous Scores', y='Performance Index', data=df, line_kws={'color': 'red'})

# Plot with a customized color for the regression line
sns.regplot(x='Previous Scores', y='Performance Index', data=df, line_kws={'color': 'orange'})

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




```
# Standarization
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()


X_train = scalar.fit_transform(X_train)
X_test = scalar.transform(X_test)


X_train
```

```
array([[ 0.00796272, -1.19062085, -0.98912579, -0.90874945,  0.13314392],
       [ 0.77854894,  1.05902612,  1.01099376, -1.49731775,  1.53122497],
       [-0.76262349,  1.40512565,  1.01099376,  0.26838714, -1.26493713],
       ...,
       [ 1.54913515, -1.24830411, -0.98912579,  0.26838714,  0.48266418],
       [-1.5332097 , -1.30598737, -0.98912579,  1.44552374, -1.6144574 ],
       [-1.14791659, -1.36367062, -0.98912579, -0.32018115,  0.48266418]])
```

```
from sklearn.linear_model import LinearRegression
regression = LinearRegression()


regression.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
# Cross Validation
from sklearn.model_selection import cross_val_score
validation_score = cross_val_score(regression, X_train, y_train, scoring='neg_mean_squared_error',cv=3)


validation_score
```

```
array([-4.36735143, -4.0421938 , -4.19884562])
```

```
np.mean(validation_score)
```

```
-4.202796949539429
```

```
## Prediction

y_pred = regression.predict(X_test)


y_pred
```

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

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)


print('Mean Squared Error:', mse)
print('Mean Absolute Error:', mae)
print('R-squared:', r2)
```

```
Mean Squared Error: 4.032544215419107
Mean Absolute Error: 1.5975792091646093
R-squared: 0.9890550757439104
```

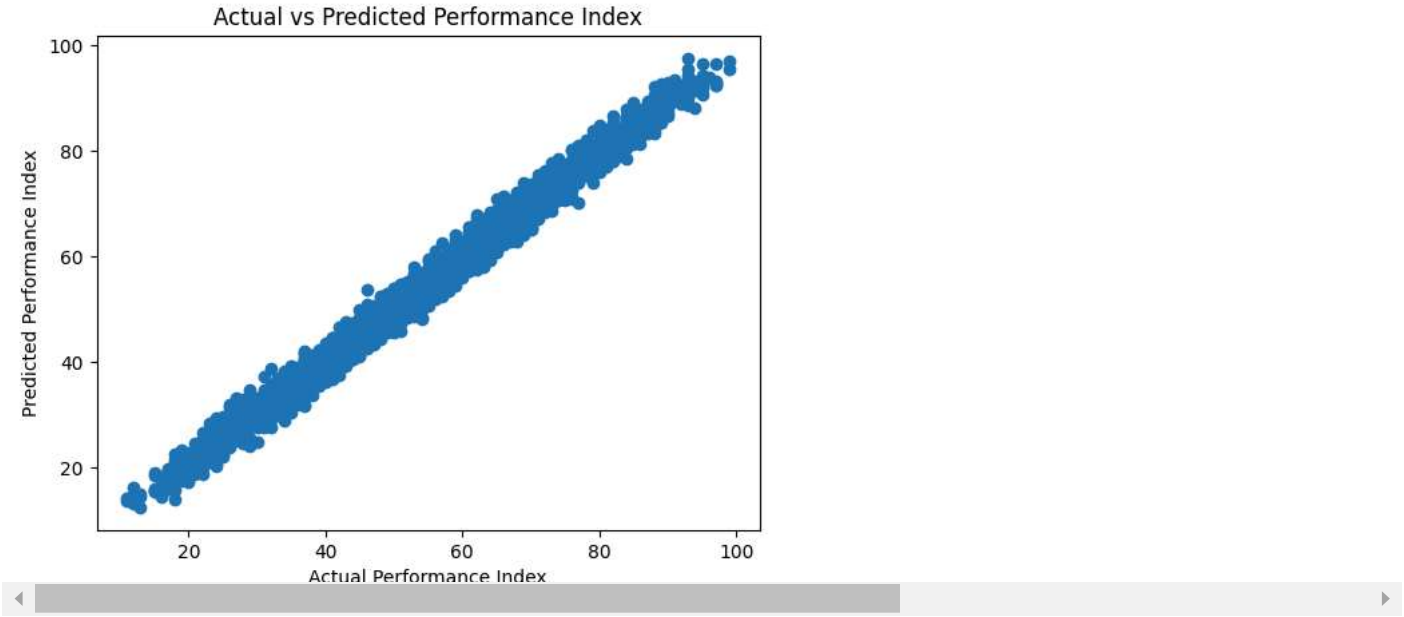
```
# Display adjusted R-Squared
adjusted_r2 = 1 - (1 - r2) * (len(y_test) - 1)/ (len(y_test) - X_test.shape[1] - 1)


print('Adjusted R-squared:', adjusted_r2)
```

```
Adjusted R-squared: 0.9890331332333729
```

```
## Assumptions
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Performance Index')
plt.ylabel('Predicted Performance Index')
plt.title('Actual vs Predicted Performance Index')
```

```
Text(0.5, 1.0, 'Actual vs Predicted Performance Index')
```




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

```
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
```

```
# Plot this residual
sns.distplot(residual)
```

```
<ipython-input-67-3ddeb65683ee>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(residual)
<Axes: xlabel='Performance Index', ylabel='Density'>
```