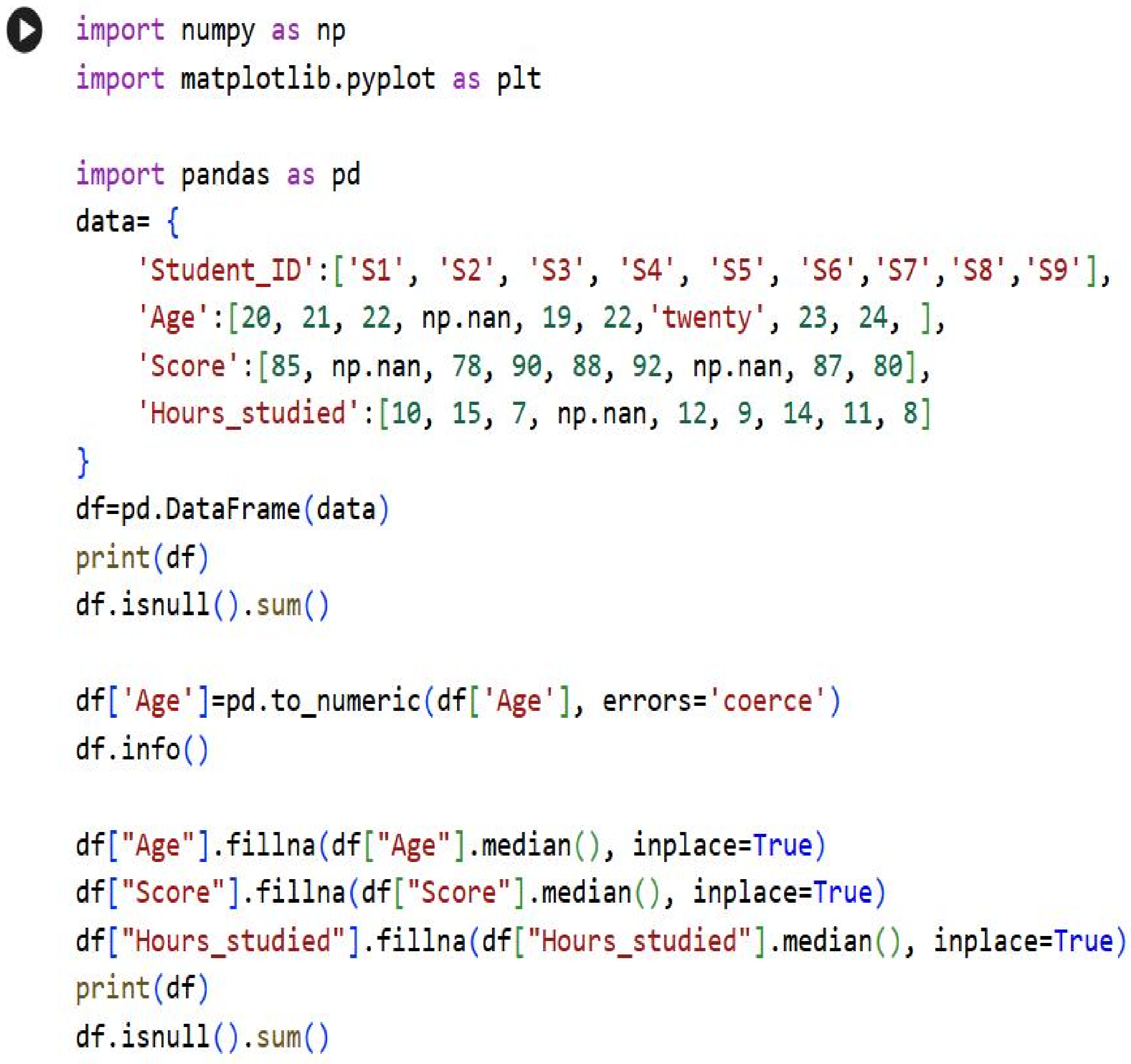
GROUP ASSIGNMENT

GROUP MEMBERS	REG NO
1. LINO NGOR MAJOK MANGOR	2405000678
2. YVES BAHATI	2305000895
3. EDGAR GAPFIZI BAHATI	25011938
4. ELYSÉE NIYIBIZI	2305000921
5. DANIEL NIZEYIMANA	2401001868

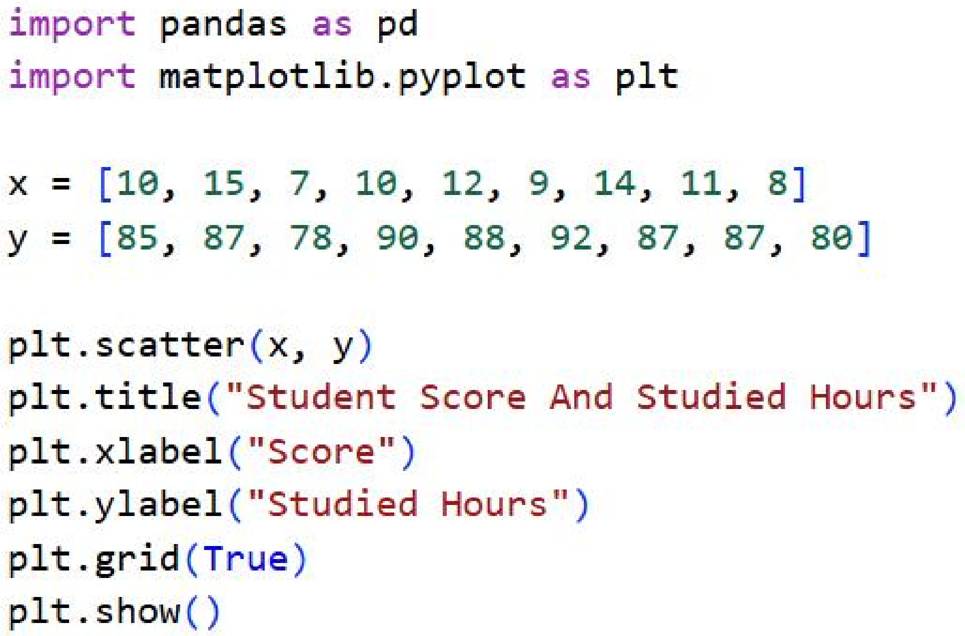
Q1.

Creating a pandas data frame

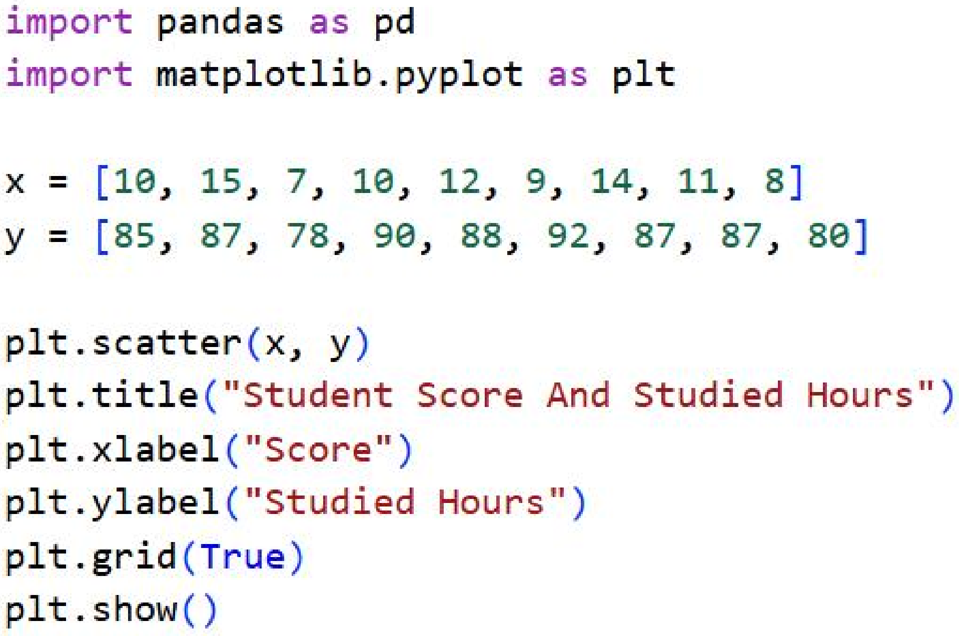
Viewing the dataset,
summarizing contents, identifying invalid data,
converting data type columns,
and treating all missing values



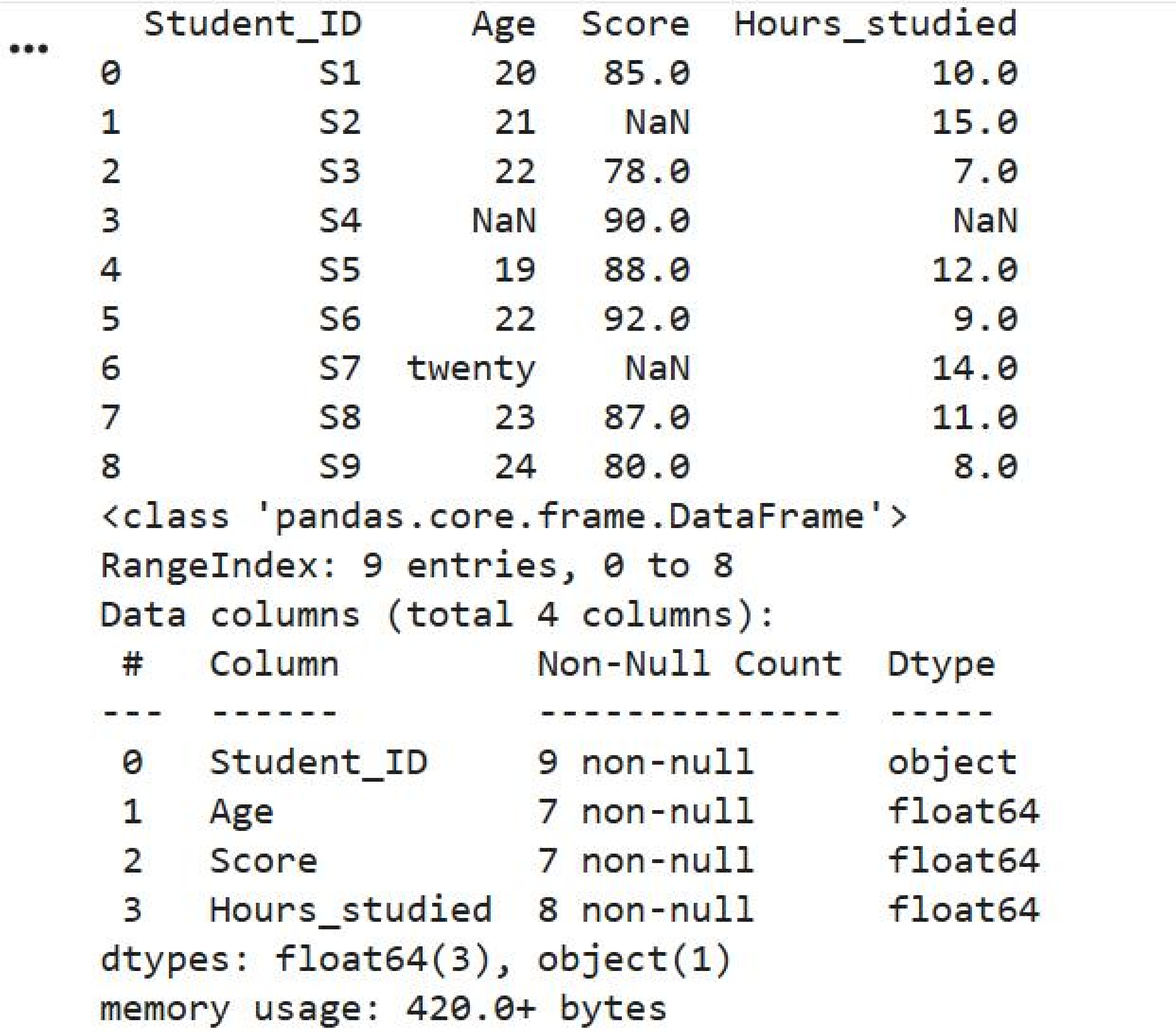
```
import numpy as np
import matplotlib.pyplot as plt

import pandas as pd
data= {
    'Student_ID':['S1', 'S2', 'S3', 'S4', 'S5', 'S6','S7','S8','S9'],
    'Age':[20, 21, 22, np.nan, 19, 22,'twenty', 23, 24, ],
    'Score':[85, np.nan, 78, 90, 88, 92, np.nan, 87, 80],
    'Hours_studied':[10, 15, 7, np.nan, 12, 9, 14, 11, 8]
}
df=pd.DataFrame(data)
print(df)
df.isnull().sum()

df['Age']=pd.to_numeric(df['Age'], errors='coerce')
df.info()

df["Age"].fillna(df["Age"].median(), inplace=True)
df["Score"].fillna(df["Score"].median(), inplace=True)
df["Hours_studied"].fillna(df["Hours_studied"].median(), inplace=True)
print(df)
df.isnull().sum()
```



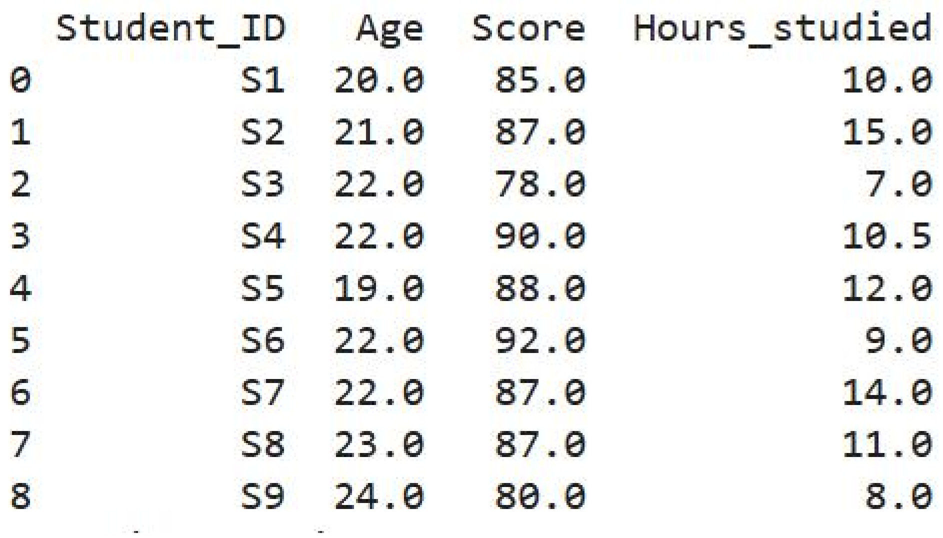
```python
import pandas as pd
import matplotlib.pyplot as plt

x = [10, 15, 7, 10, 12, 9, 14, 11, 8]
y = [85, 87, 78, 90, 88, 92, 87, 87, 80]

plt.scatter(x, y)
plt.title("Student Score And Studied Hours")
plt.xlabel("Score")
plt.ylabel("Studied Hours")
plt.grid(True)
plt.show()
```



```
import pandas as pd
import matplotlib.pyplot as plt

x = [10, 15, 7, 10, 12, 9, 14, 11, 8]
y = [85, 87, 78, 90, 88, 92, 87, 87, 80]

plt.scatter(x, y)
plt.title("Student Score And Studied Hours")
plt.xlabel("Score")
plt.ylabel("Studied Hours")
plt.grid(True)
plt.show()
```

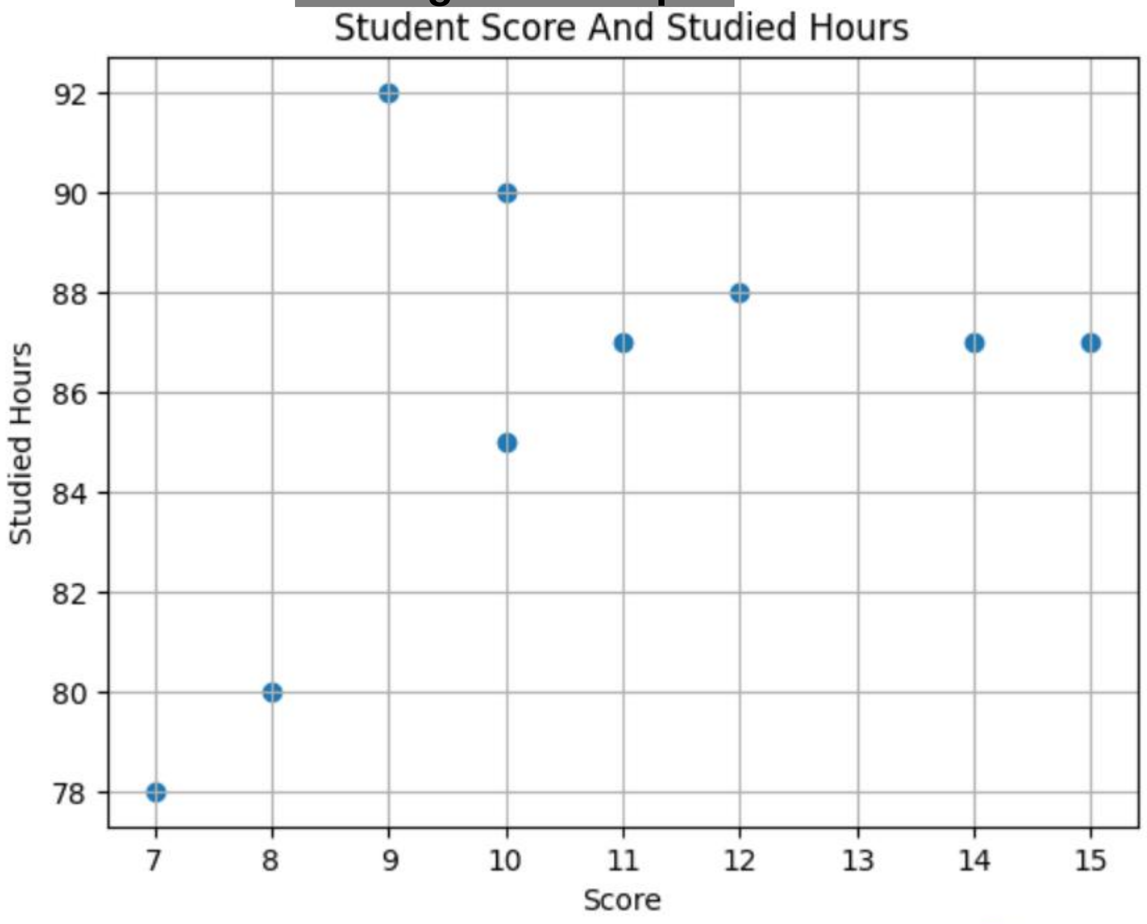


```
  Student_ID     Age  Score  Hours_studied
0         S1      20   85.0           10.0
1         S2      21    NaN           15.0
2         S3      22   78.0            7.0
3         S4     NaN   90.0            NaN
4         S5      19   88.0           12.0
5         S6      22   92.0            9.0
6         S7  twenty    NaN           14.0
7         S8      23   87.0           11.0
8         S9      24   80.0            8.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Student_ID     9 non-null      object 
 1   Age            7 non-null      float64
 2   Score          7 non-null      float64
 3   Hours_studied  8 non-null      float64
dtypes: float64(3), object(1)
memory usage: 420.0+ bytes
```



```
  Student_ID   Age  Score  Hours_studied
0         S1  20.0   85.0           10.0
1         S2  21.0   87.0           15.0
2         S3  22.0   78.0            7.0
3         S4  22.0   90.0           10.5
4         S5  19.0   88.0           12.0
5         S6  22.0   92.0            9.0
6         S7  22.0   87.0           14.0
7         S8  23.0   87.0           11.0
8         S9  24.0   80.0            8.0
```

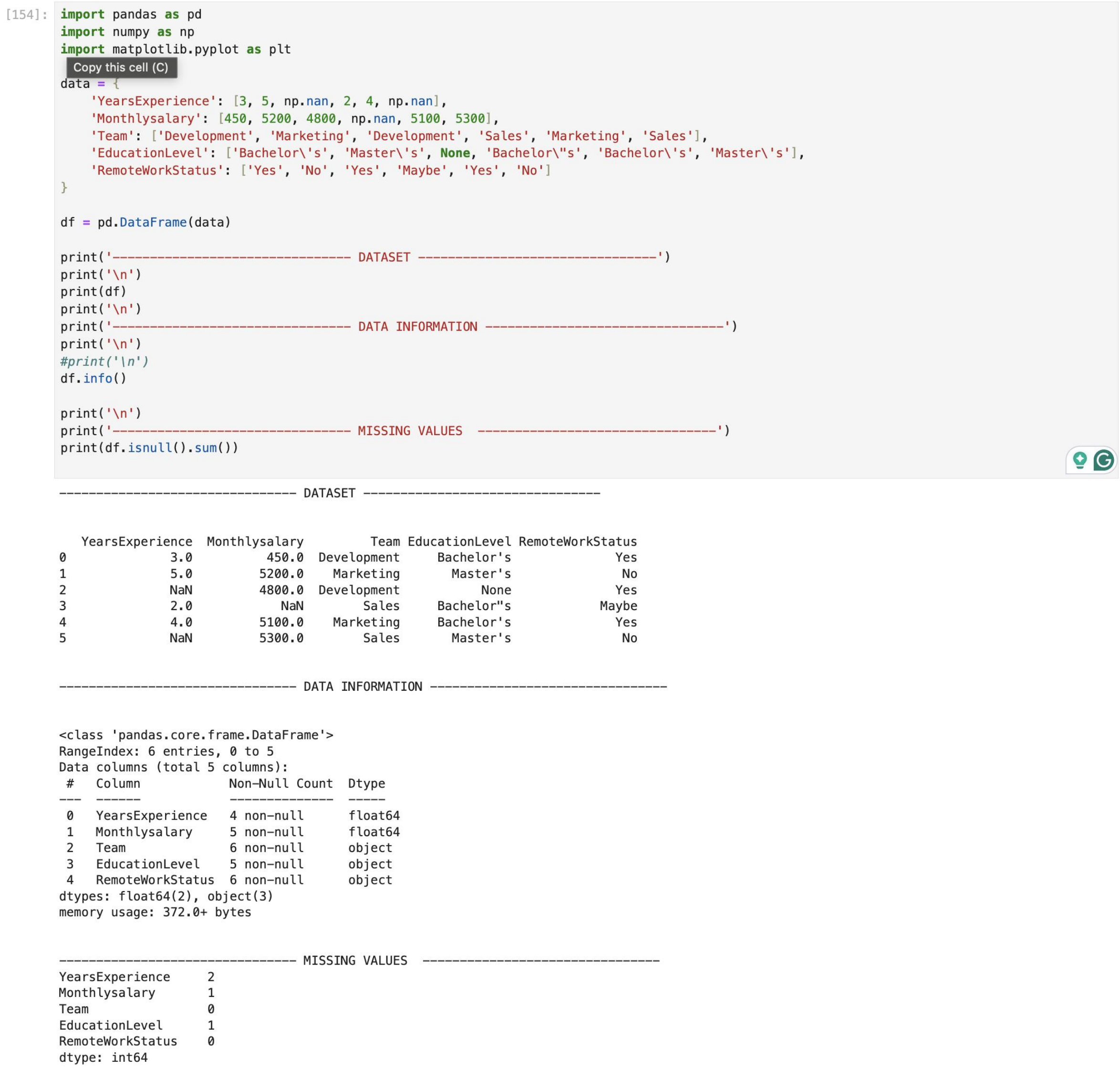
Plotting a scatter plot



Q2.

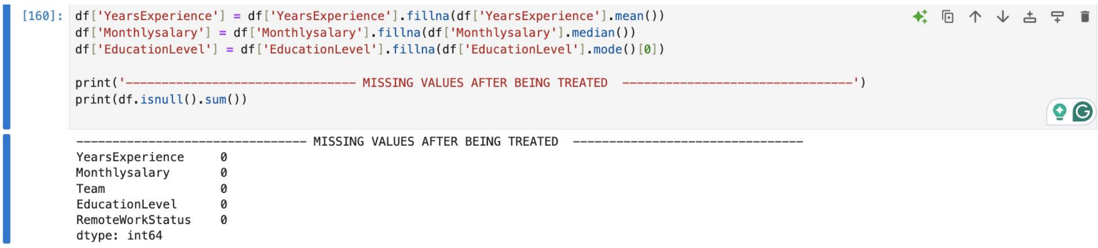
A.

A cleaner Dataset with fewer missing values is more reliable for analysis and model training. Enhanced model performance: Properly handling missing values helps models perform better by training on complete data, leading to more accurate predictions.



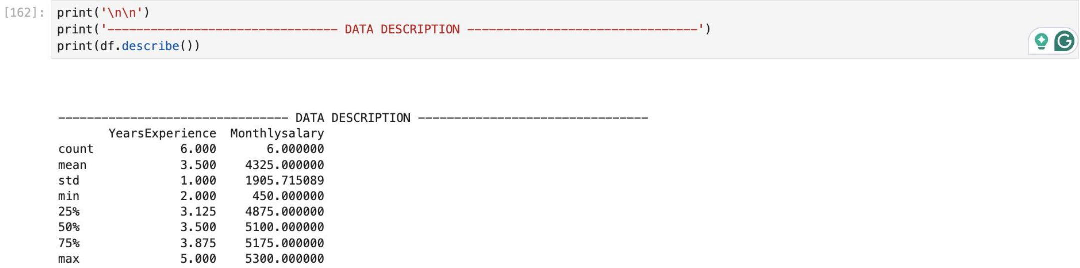
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = {
    'YearsExperience': [3, 5, np.nan, 2, 4, np.nan],
    'Monthlysalary': [450, 5200, 4800, np.nan, 5100, 5300],
    'Team': ['Development', 'Marketing', 'Development', 'Sales', 'Marketing', 'Sales'],
    'EducationLevel': ['Bachelor\'s', 'Master\'s', None, 'Bachelor\"s', 'Bachelor\'s', 'Master\'s'],
    'RemoteWorkStatus': ['Yes', 'No', 'Yes', 'Maybe', 'Yes', 'No']
}

df = pd.DataFrame(data)

print('-------------------------------- DATASET --------------------------------')
print('\n')
print(df)
print('\n')
print('-------------------------------- DATA INFORMATION --------------------------------')
print('\n')
#print('\n')
df.info()

print('\n')
print('-------------------------------- MISSING VALUES  --------------------------------')
print(df.isnull().sum())
```

```
-------------------------------- DATASET --------------------------------


   YearsExperience  Monthlysalary         Team EducationLevel RemoteWorkStatus
0              3.0          450.0  Development     Bachelor's              Yes
1              5.0         5200.0    Marketing       Master's               No
2              NaN         4800.0  Development           None              Yes
3              2.0            NaN        Sales     Bachelor"s            Maybe
4              4.0         5100.0    Marketing     Bachelor's              Yes
5              NaN         5300.0        Sales       Master's               No


-------------------------------- DATA INFORMATION --------------------------------


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   YearsExperience   4 non-null      float64
 1   Monthlysalary     5 non-null      float64
 2   Team              6 non-null      object 
 3   EducationLevel    5 non-null      object 
 4   RemoteWorkStatus  6 non-null      object 
dtypes: float64(2), object(3)
memory usage: 372.0+ bytes


-------------------------------- MISSING VALUES  --------------------------------
YearsExperience     2
Monthlysalary       1
Team                0
EducationLevel      1
RemoteWorkStatus    0
dtype: int64
```



```
[160]: df['YearsExperience'] = df['YearsExperience'].fillna(df['YearsExperience'].mean())
       df['Monthlysalary'] = df['Monthlysalary'].fillna(df['Monthlysalary'].median())
       df['EducationLevel'] = df['EducationLevel'].fillna(df['EducationLevel'].mode()[0])

       print('-------------------------------- MISSING VALUES AFTER BEING TREATED  --------------------------------')
       print(df.isnull().sum())
```

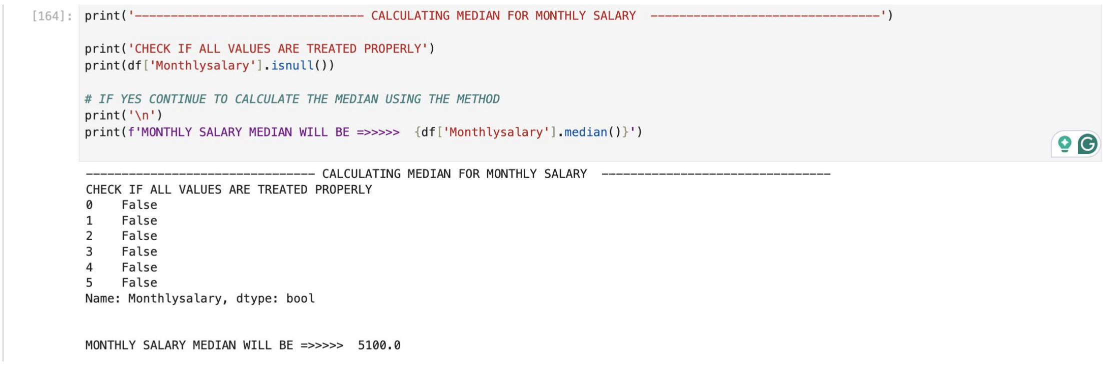
```
-------------------------------- MISSING VALUES AFTER BEING TREATED  --------------------------------
YearsExperience     0
Monthlysalary       0
Team                0
EducationLevel      0
RemoteWorkStatus    0
dtype: int64
```



```
[162]: print('\n\n')
       print('-------------------------------- DATA DESCRIPTION --------------------------------')
       print(df.describe())
```

```



-------------------------------- DATA DESCRIPTION --------------------------------
       YearsExperience  Monthlysalary
count            6.000       6.000000
mean             3.500    4325.000000
std              1.000    1905.715089
min              2.000     450.000000
25%              3.125    4875.000000
50%              3.500    5100.000000
75%              3.875    5175.000000
max              5.000    5300.000000
```

B. CALCULATING MEDIAN FOR MONTHLY SALARY

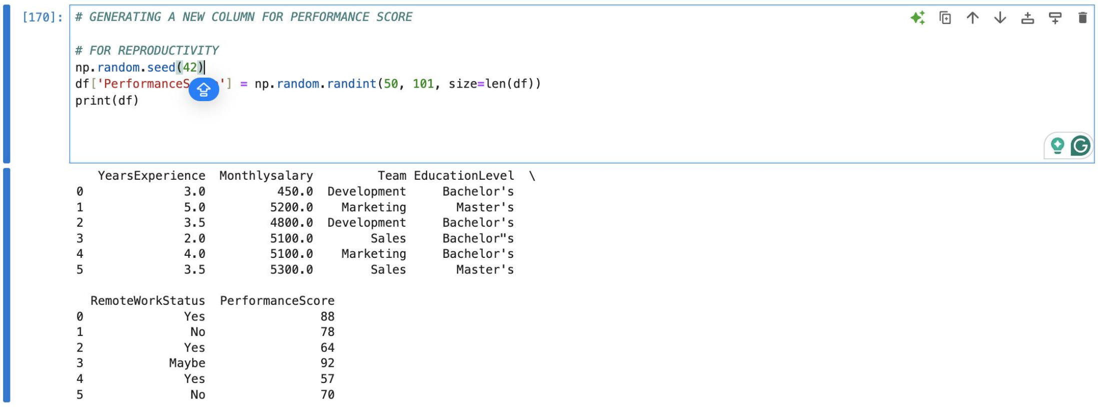


```
[164]: print('-------------------------------- CALCULATING MEDIAN FOR MONTHLY SALARY  --------------------------------')

       print('CHECK IF ALL VALUES ARE TREATED PROPERLY')
       print(df['Monthlysalary'].isnull())

       # IF YES CONTINUE TO CALCULATE THE MEDIAN USING THE METHOD
       print('\n')
       print(f'MONTHLY SALARY MEDIAN WILL BE =>>>>>  {df['Monthlysalary'].median()}')
```

```
-------------------------------- CALCULATING MEDIAN FOR MONTHLY SALARY  --------------------------------
CHECK IF ALL VALUES ARE TREATED PROPERLY
0    False
1    False
2    False
3    False
4    False
5    False
Name: Monthlysalary, dtype: bool


MONTHLY SALARY MEDIAN WILL BE =>>>>>  5100.0
```

C.

I. GENERATING A NEW COLUMN FOR PERFORMANCE SCORE

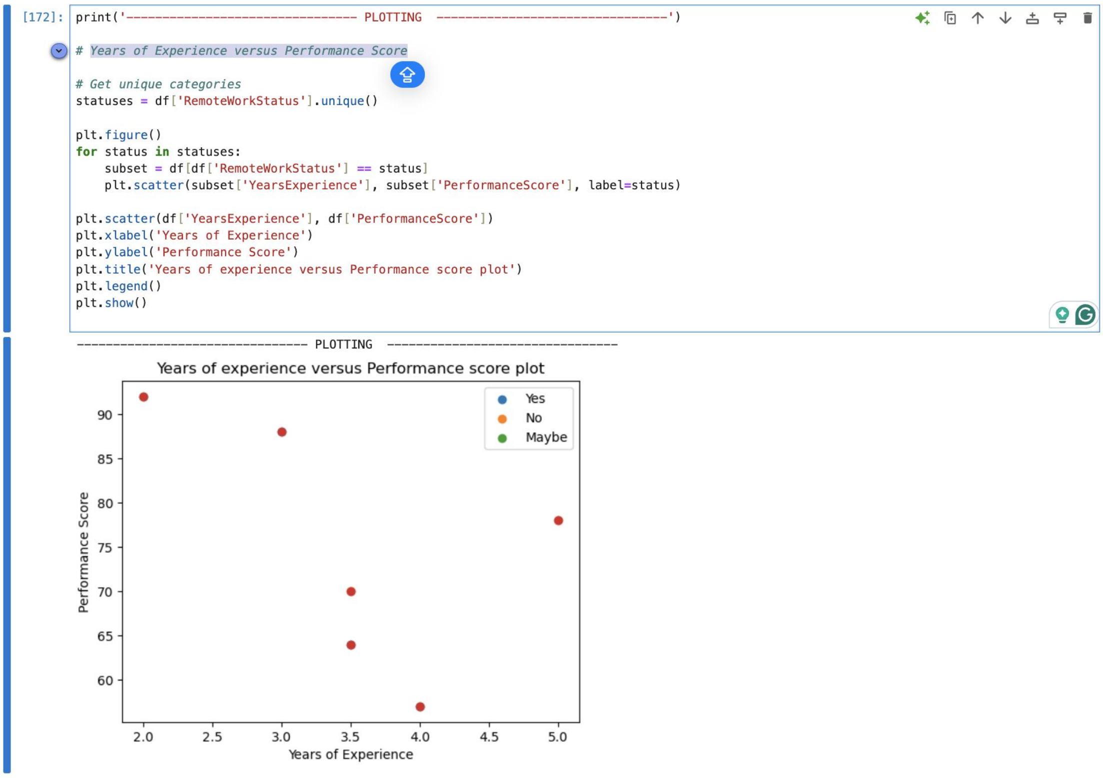


[170]:

```
# GENERATING A NEW COLUMN FOR PERFORMANCE SCORE

# FOR REPRODUCTIVITY
np.random.seed(42)
df['PerformanceScore'] = np.random.randint(50, 101, size=len(df))
print(df)
```

```
   YearsExperience  Monthlysalary         Team EducationLevel  \
0              3.0          450.0  Development     Bachelor's
1              5.0         5200.0    Marketing       Master's
2              3.5         4800.0  Development     Bachelor's
3              2.0         5100.0        Sales     Bachelor"s
4              4.0         5100.0    Marketing     Bachelor's
5              3.5         5300.0        Sales       Master's

  RemoteWorkStatus  PerformanceScore
0              Yes                88
1               No                78
2              Yes                64
3            Maybe                92
4              Yes                57
5               No                70
```

II. YEARS OF EXPERIENCE VERSUS PERFORMANCE SCORE



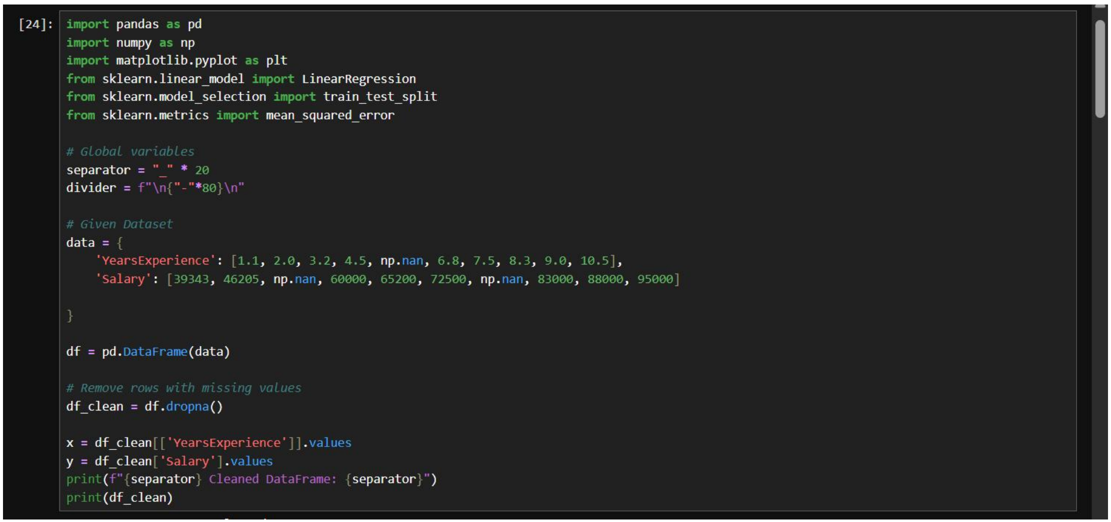
[172]:

```
print('--------------------------------- PLOTTING  ---------------------------------')

# Years of Experience versus Performance Score

# Get unique categories
statuses = df['RemoteWorkStatus'].unique()

plt.figure()
for status in statuses:
    subset = df[df['RemoteWorkStatus'] == status]
    plt.scatter(subset['YearsExperience'], subset['PerformanceScore'], label=status)

plt.scatter(df['YearsExperience'], df['PerformanceScore'])
plt.xlabel('Years of Experience')
plt.ylabel('Performance Score')
plt.title('Years of experience versus Performance score plot')
plt.legend()
plt.show()
```

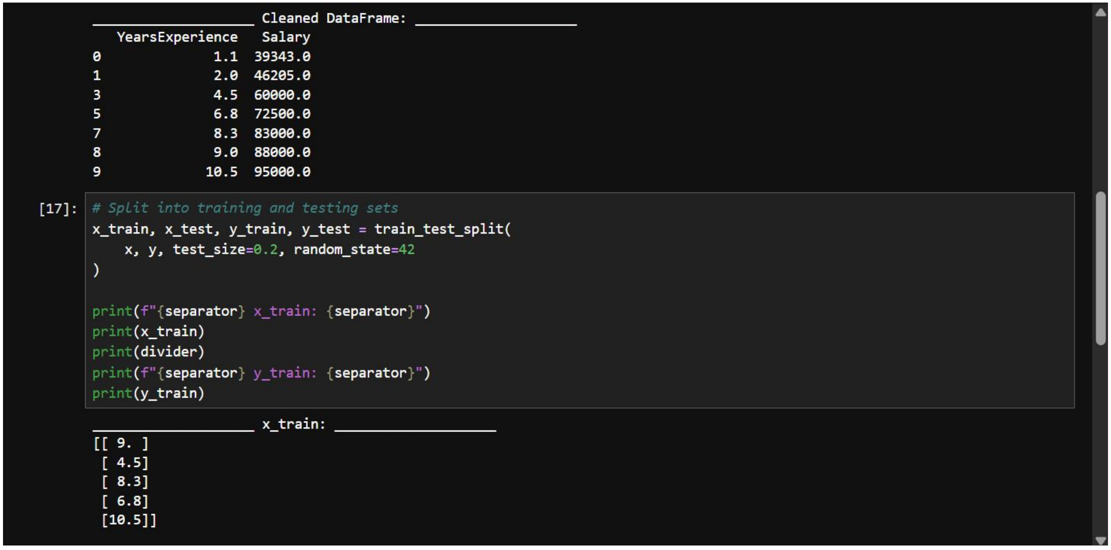
```
--------------------------------- PLOTTING  ---------------------------------
```

Question 3.

A.



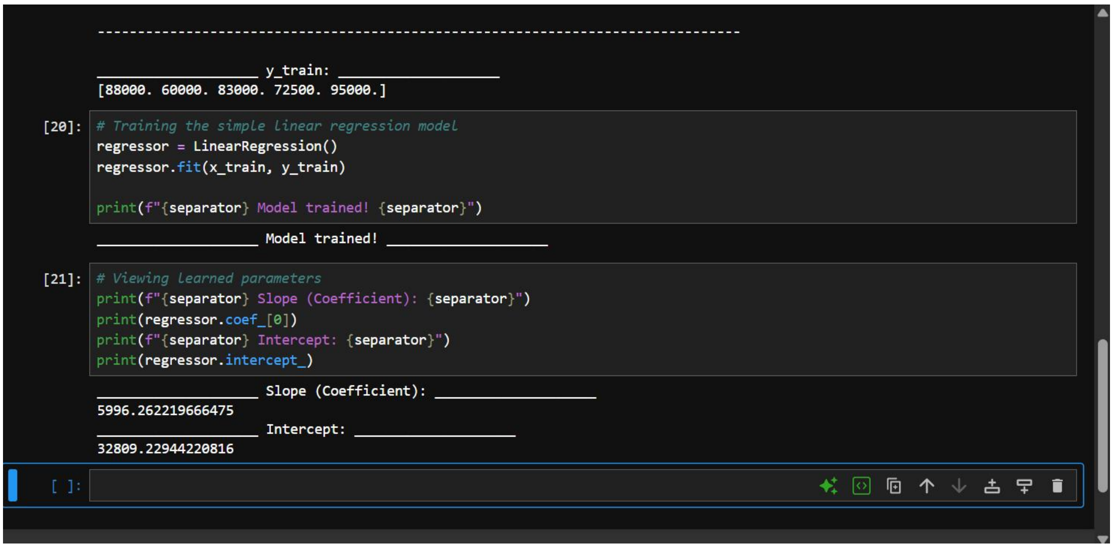
```
[24]: import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt
      from sklearn.linear_model import LinearRegression
      from sklearn.model_selection import train_test_split
      from sklearn.metrics import mean_squared_error

      # Global variables
      separator = "_" * 20
      divider = f"\n{"-"*80}\n"

      # Given Dataset
      data = {
          'YearsExperience': [1.1, 2.0, 3.2, 4.5, np.nan, 6.8, 7.5, 8.3, 9.0, 10.5],
          'Salary': [39343, 46205, np.nan, 60000, 65200, 72500, np.nan, 83000, 88000, 95000]

      }

      df = pd.DataFrame(data)

      # Remove rows with missing values
      df_clean = df.dropna()

      x = df_clean[['YearsExperience']].values
      y = df_clean['Salary'].values
      print(f"{separator} Cleaned DataFrame: {separator}")
      print(df_clean)
```



```
____________________ Cleaned DataFrame: ____________________
   YearsExperience   Salary
0              1.1  39343.0
1              2.0  46205.0
3              4.5  60000.0
5              6.8  72500.0
7              8.3  83000.0
8              9.0  88000.0
9             10.5  95000.0
```

```
[17]: # Split into training and testing sets
      x_train, x_test, y_train, y_test = train_test_split(
          x, y, test_size=0.2, random_state=42
      )

      print(f"{separator} x_train: {separator}")
      print(x_train)
      print(divider)
      print(f"{separator} y_train: {separator}")
      print(y_train)
```

```
____________________ x_train: ____________________
[[ 9. ]
 [ 4.5]
 [ 8.3]
 [ 6.8]
 [10.5]]
```



```
--------------------------------------------------------------------------------

____________________ y_train: ____________________
[88000. 60000. 83000. 72500. 95000.]
```

```
[20]: # Training the simple linear regression model
      regressor = LinearRegression()
      regressor.fit(x_train, y_train)

      print(f"{separator} Model trained! {separator}")
```

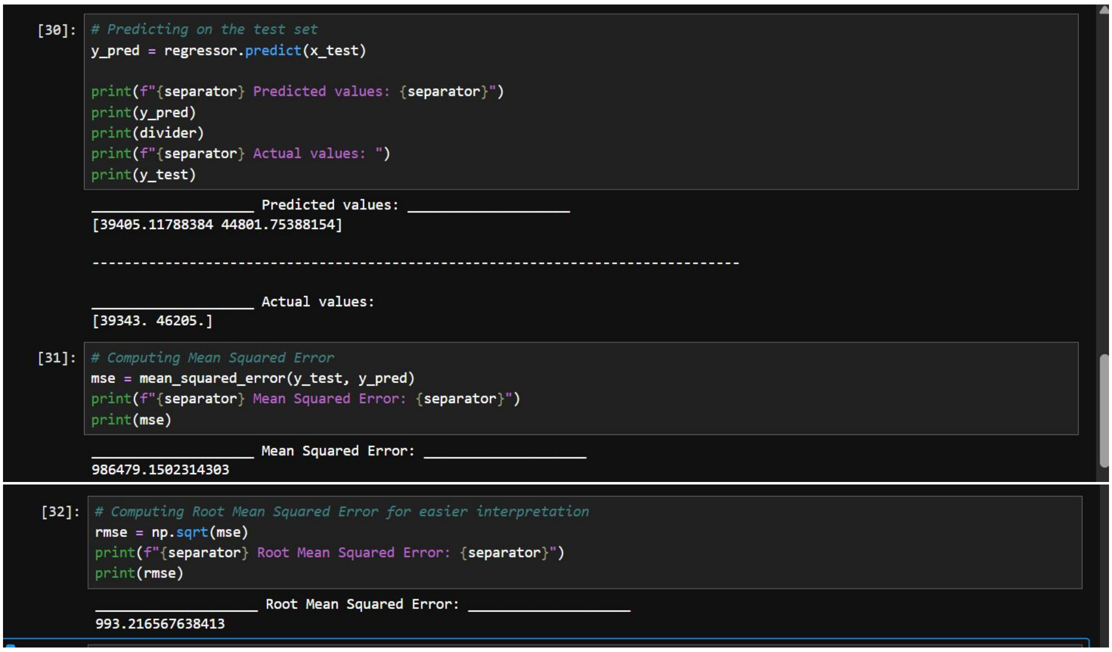
```
____________________ Model trained! ____________________
```

```
[21]: # Viewing learned parameters
      print(f"{separator} Slope (Coefficient): {separator}")
      print(regressor.coef_[0])
      print(f"{separator} Intercept: {separator}")
      print(regressor.intercept_)
```

```
____________________ Slope (Coefficient): ____________________
5996.262219666475
____________________ Intercept: ____________________
32809.22944220816
```

```
[ ]:
```

B.



```
[30]: # Predicting on the test set
      y_pred = regressor.predict(x_test)

      print(f"{separator} Predicted values: {separator}")
      print(y_pred)
      print(divider)
      print(f"{separator} Actual values: ")
      print(y_test)
```

```
____________________ Predicted values: ____________________
[39405.11788384 44801.75388154]

--------------------------------------------------------------------------------

____________________ Actual values:
[39343. 46205.]
```

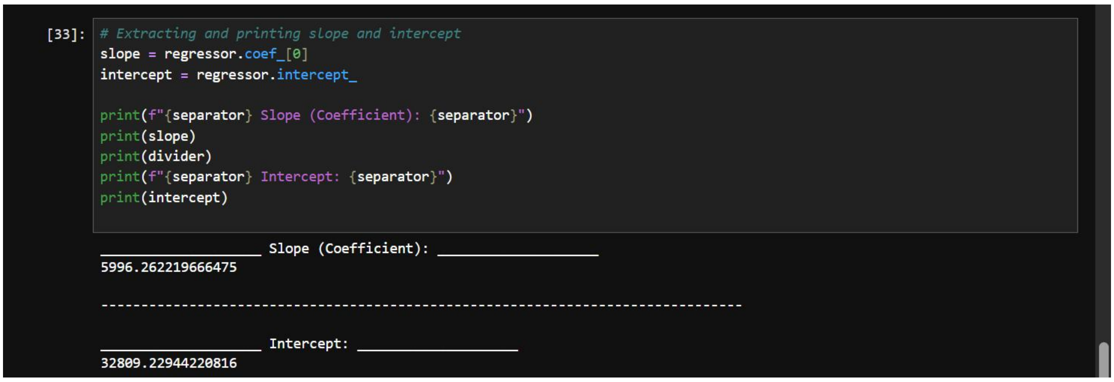
```
[31]: # Computing Mean Squared Error
      mse = mean_squared_error(y_test, y_pred)
      print(f"{separator} Mean Squared Error: {separator}")
      print(mse)
```

```
____________________ Mean Squared Error: ____________________
986479.1502314303
```

```
[32]: # Computing Root Mean Squared Error for easier interpretation
      rmse = np.sqrt(mse)
      print(f"{separator} Root Mean Squared Error: {separator}")
      print(rmse)
```

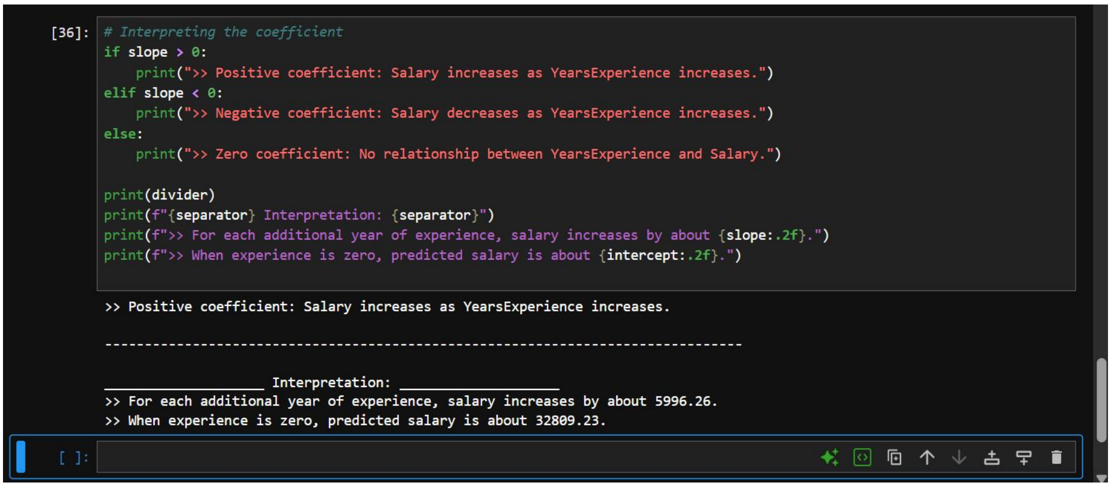
```
____________________ Root Mean Squared Error: ____________________
993.216567638413
```

C.



```
[33]: # Extracting and printing slope and intercept
      slope = regressor.coef_[0]
      intercept = regressor.intercept_

      print(f"{separator} Slope (Coefficient): {separator}")
      print(slope)
      print(divider)
      print(f"{separator} Intercept: {separator}")
      print(intercept)
```

```
____________________ Slope (Coefficient): ____________________
5996.262219666475

--------------------------------------------------------------------------------

____________________ Intercept: ____________________
32809.22944220816
```



[36]:

```python
# Interpreting the coefficient
if slope > 0:
    print(">> Positive coefficient: Salary increases as YearsExperience increases.")
elif slope < 0:
    print(">> Negative coefficient: Salary decreases as YearsExperience increases.")
else:
    print(">> Zero coefficient: No relationship between YearsExperience and Salary.")

print(divider)
print(f"{separator} Interpretation: {separator}")
print(f">> For each additional year of experience, salary increases by about {slope:.2f}.")
print(f">> When experience is zero, predicted salary is about {intercept:.2f}.")
```

```
>> Positive coefficient: Salary increases as YearsExperience increases.

--------------------------------------------------------------------------------

____________________ Interpretation: ____________________
>> For each additional year of experience, salary increases by about 5996.26.
>> When experience is zero, predicted salary is about 32809.23.
```

[ ]: