**Student Performance Documentation**

Follow the instruction Steps:

**Step:-1** **Install python in your pc**

Download Python From <https://www.python.org/downloads/> or Search Python.org

Download the new updated version as your operating system,

After download install it in your main drive(c)/ in your OS.

Open CMD and run as administrator

Check python Version using this command.

**Command:- python –version**

**Step:-2 Install PyCharm Software in you pc.**

Download PyCharm Software

Install the **PyCharm Community Edition**

From this link

<https://www.jetbrains.com/pycharm/download/downloadthanks.html?platform=windows&code=PCC>

After Download Complete Install it and grant permission.

**Step:-3 Now Install Some Main Libraries**

Open CMD And write this following command

* 1. Install pandas for data manipulation and analysis

pip install pandas

3.2 Install Matplotlib for data visualization

pip install matplotlib

* 1. Install seaborn for statistical data visualization

pip install seaborn

* 1. Install scikit-learn for machine learning algorithms

pip install scikit-learn

* 1. Install OpenCv for Open-source computer vision and machine learning software.

pip install opencv-python

* 1. Install Tenserflow for machine learning and deep learning tasks.

pip install tensorflow

Now We Install all the basic Libraries for Python.

1. pandas: It is a powerful Python library used for data manipulation and analysis. It provides data structures like DataFrame and Series, which are very useful for handling structured data.

2. matplotlib: This library is used for data visualization in Python. It provides a variety of plotting functions to create static, interactive, and animated visualizations.

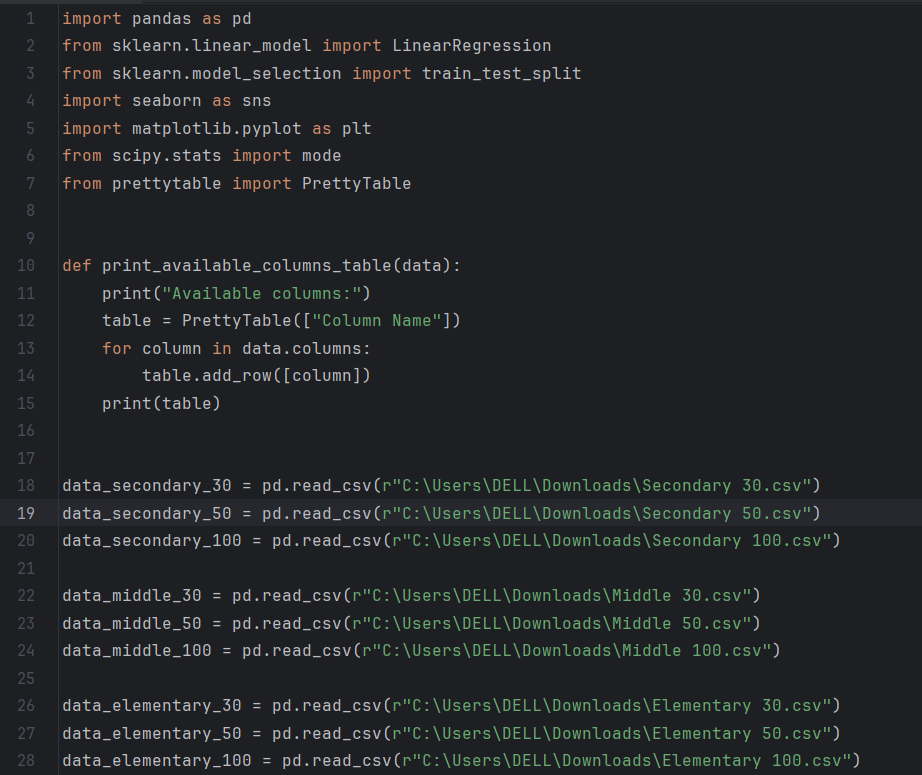
3. seaborn: Seaborn is built on top of matplotlib and provides a high-level interface for drawing attractive and informative statistical graphics. It simplifies the process of creating complex visualizations such as heatmaps, pair plots, and categorical plots.

4. scikit-learn: Also known as sklearn, scikit-learn is a machine learning library in Python. It provides simple and efficient tools for data mining and data analysis, including various machine learning algorithms for classification, regression, clustering, dimensionality reduction, and more.

5. OpenCV (opencv-python): OpenCV is an open-source computer vision and machine learning software library. It provides tools and algorithms for image processing, computer vision tasks, and machine learning applications related to images and videos.

6. TensorFlow (tensorflow): TensorFlow is an open-source machine learning framework developed by Google. It is widely used for building and training deep learning models, including neural networks for various tasks such as image classification, natural language processing, and more.

**Step:-4 Start To Code**

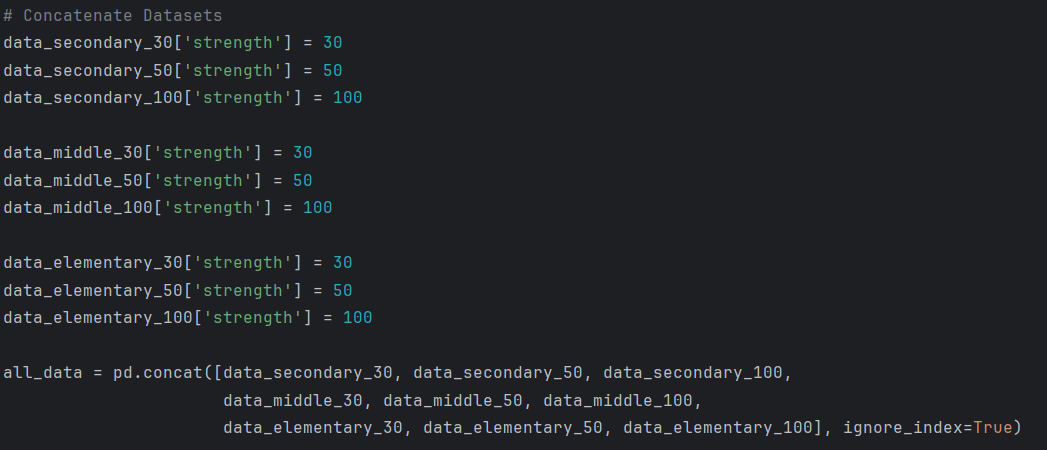


Lets break down the code step by step:

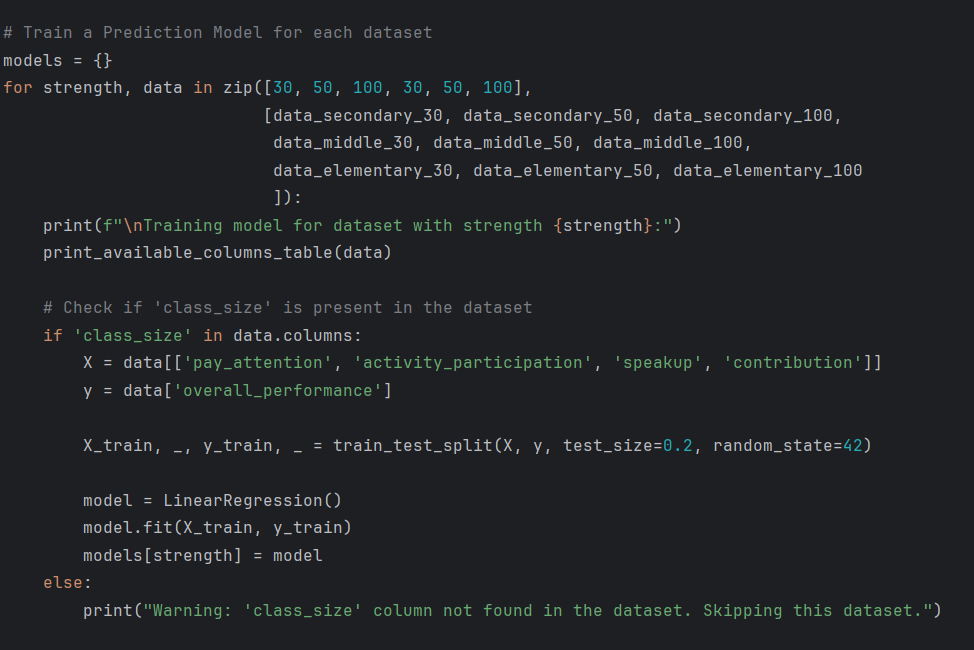
1. **Define a Function print\_available\_columns\_table:**
   * Defines a function that takes a DataFrame (**data**) as input.
   * Prints a header indicating "Available columns."
   * Creates a **PrettyTable** object with a single column ("Column Name").
   * Iterates through the columns of the input DataFrame and adds each column name as a row to the table.
   * Prints the table.
2. **Read CSV Files into DataFrames:**
   * Reads data from CSV files into separate DataFrames for different datasets (**data\_secondary\_30**, **data\_secondary\_50**, ..., **data\_elementary\_100**).
3. **Load Datasets:**
   * Each dataset corresponds to a certain type (Secondary, Middle, Elementary) and a specific strength (30, 50, 100).
4. **Example Usage of print\_available\_columns\_table:**
   * The function **print\_available\_columns\_table** can be called with any of the loaded DataFrames to display the available columns for that specific dataset.



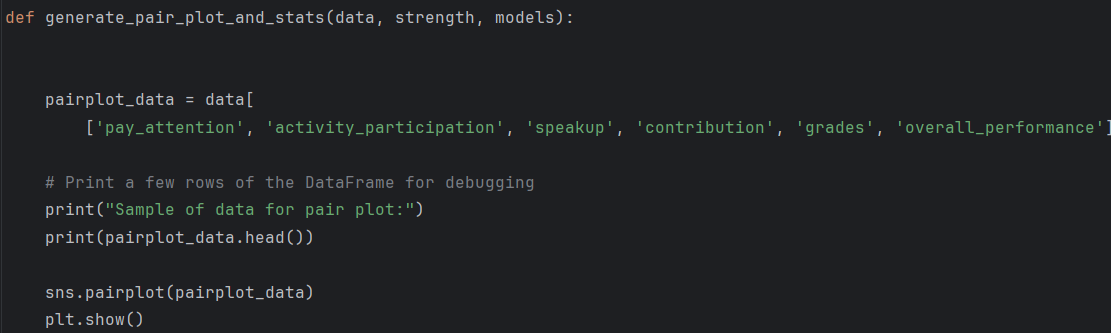
1. **Print Statements:**
   * Prints statements indicating the dataset type and strength for each section (Secondary, Middle, Elementary) and strength (30, 50, 100).
2. **Function Calls:**
   * Calls the **print\_available\_columns\_table** function for each corresponding DataFrame (**data\_secondary\_30**, **data\_secondary\_50**, ..., **data\_elementary\_100**).
3. **Output:**
   * The output of each function call displays a table with a single column ("Column Name") containing the available columns for the respective dataset.



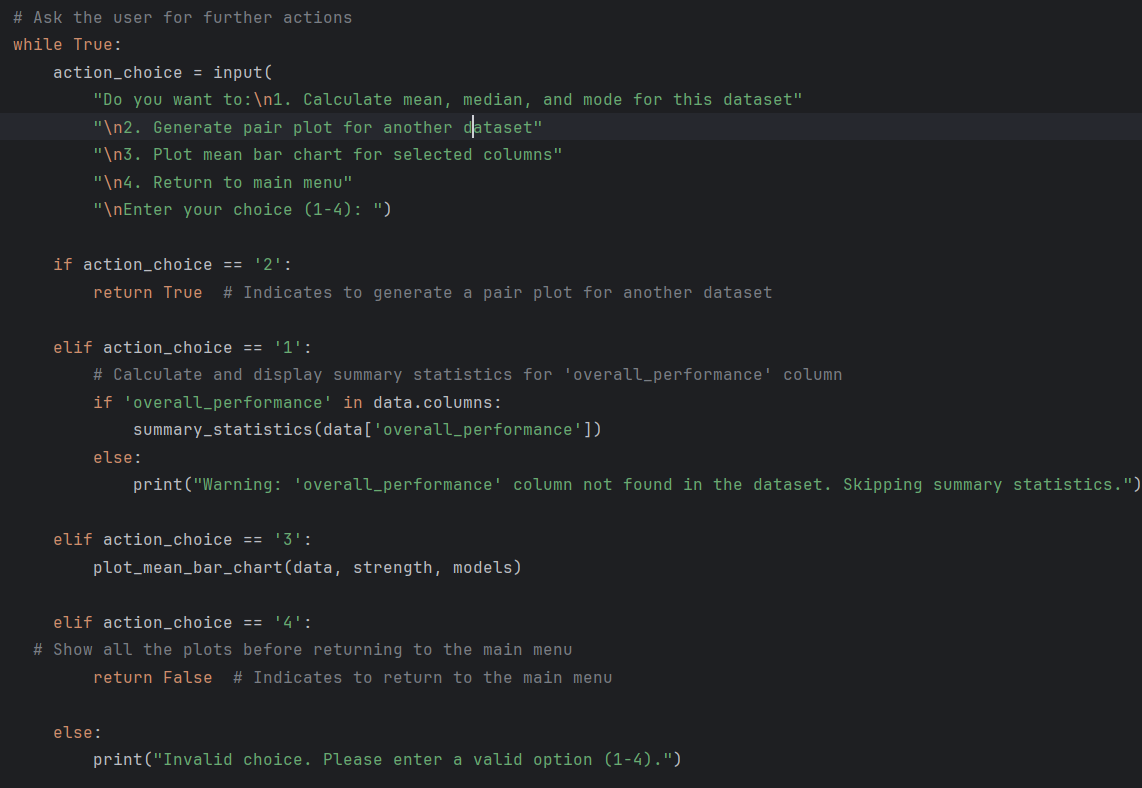
1. **Add 'strength' Column:**
   * Adds a new column named 'strength' to each individual dataset, assigning the corresponding strength value (30, 50, or 100).
2. **Concatenate Datasets:**
   * Uses the **pd.concat** function to concatenate the individual datasets along the rows (**axis=0**) into a single DataFrame (**all\_data**).
   * The **ignore\_index=True** parameter resets the index of the resulting DataFrame.
3. **Result:**
   * The resulting DataFrame **all\_data** contains all the rows from the concatenated datasets, and the 'strength' column indicates the strength value for each row.



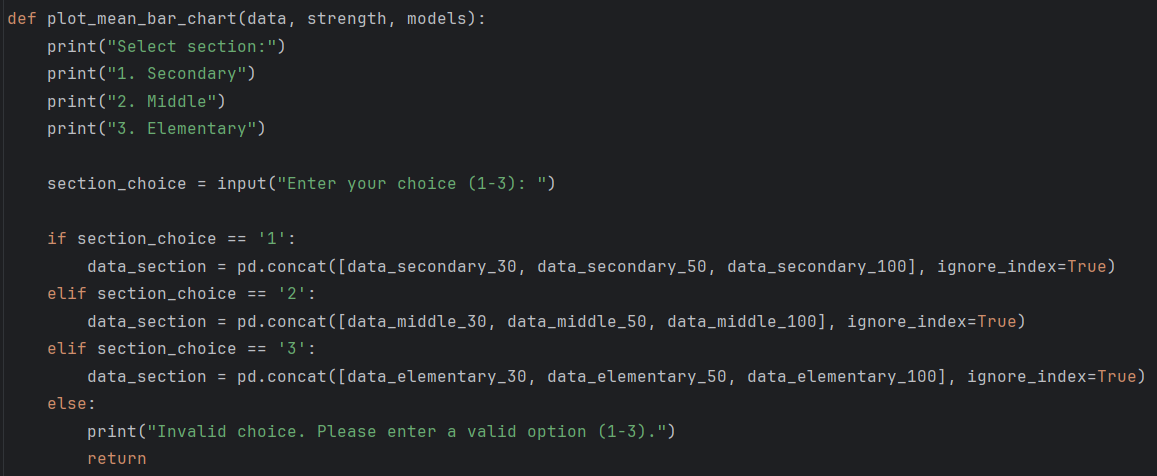
1. **Iteration Over Datasets:**
   * Uses a loop to iterate over datasets for each strength (30, 50, 100).
   * The datasets are organized in pairs, where the first three represent Secondary datasets, the next three represent Middle datasets, and the last three represent Elementary datasets.
2. **Print Training Information:**
   * Prints information indicating that a model is being trained for the current dataset strength.
   * Calls the previously defined **print\_available\_columns\_table** function to display the available columns for the current dataset.
3. **Check for 'class\_size' Column:**
   * Checks if the 'class\_size' column is present in the dataset.
4. **Feature and Target Selection:**
   * If 'class\_size' is present, selects the features ('pay\_attention', 'activity\_participation', 'speakup', 'contribution') and the target ('overall\_performance') for training the model.
5. **Train Linear Regression Model:**
   * Uses **train\_test\_split** to split the dataset into training and testing sets.
   * Initializes a Linear Regression model and fits it to the training data.
6. **Store Trained Model:**
   * Stores the trained model in the **models** dictionary, using the strength value as the key.
7. **Warning for Missing 'class\_size' Column:**
   * If the 'class\_size' column is not found in the dataset, prints a warning message and skips training a model for that dataset.



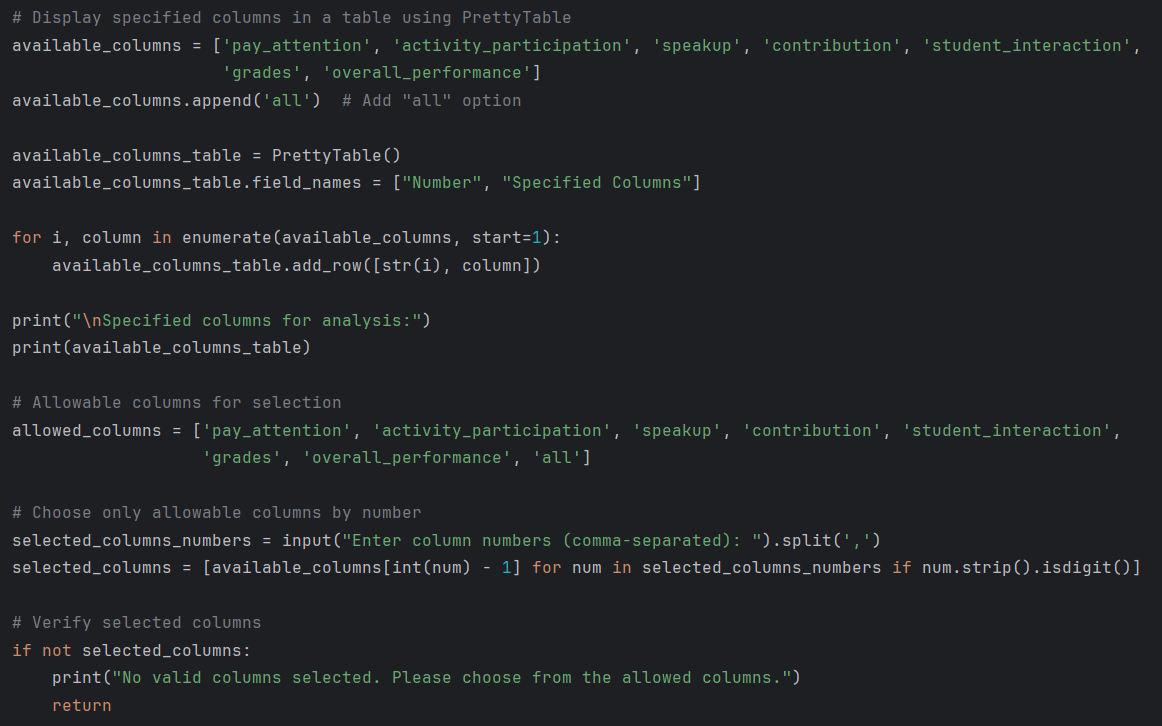
1. **Select Relevant Columns:**
   * Creates a subset (**pairplot\_data**) of the original dataset (**data**) containing specific columns: 'pay\_attention', 'activity\_participation', 'speakup', 'contribution', 'grades', 'overall\_performance'.
2. **Print Sample of Data:**
   * Prints a sample of the selected data to provide a preview of the content.
3. **Generate Pair Plot:**
   * Uses Seaborn's **pairplot** function to create a pair plot based on the selected columns in **pairplot\_data**.
4. **Display Pair Plot:**
   * Uses **plt.show()** to display the generated pair plot.



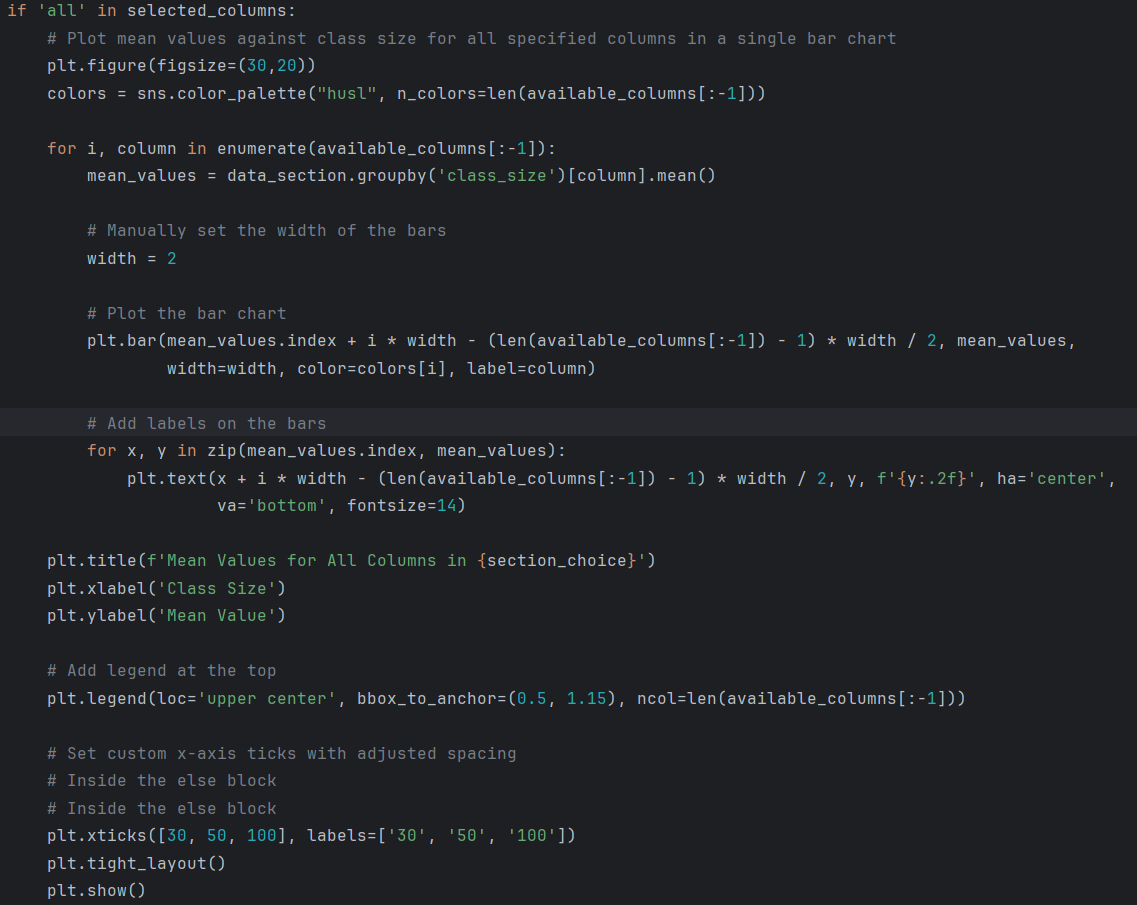
1. **User Interaction Loop (while True):**
   * The code enters an infinite loop to keep prompting the user for actions until the user chooses to return to the main menu.
2. **User Input Prompt:**
   * Asks the user for their choice of action, presenting a menu with options (1-4).
3. **Action Choices:**
   * **Choice 1 (action\_choice == '1'):**
     + If the user chooses to calculate mean, median, and mode for the dataset, it checks if the 'overall\_performance' column is present in the dataset. If present, it calls the **summary\_statistics** function to display summary statistics for that column. If not present, it prints a warning message.
   * **Choice 2 (action\_choice == '2'):**
     + If the user chooses to generate a pair plot for another dataset, it returns **True** to signal that the pair plot generation should continue.
   * **Choice 3 (action\_choice == '3'):**
     + If the user chooses to plot a mean bar chart for selected columns, it calls the **plot\_mean\_bar\_chart** function with the current dataset, strength, and models.
   * **Choice 4 (action\_choice == '4'):**
     + If the user chooses to return to the main menu, it returns **False** to indicate the intention to return.
   * **Invalid Choice:**
     + If the user enters an invalid choice, it prints an error message.



1. **Print Section Options:**
   * Prints the options for selecting a section ('Secondary', 'Middle', 'Elementary').
2. **User Section Choice:**
   * Asks the user to input their choice (1-3) for the section.
3. **Section Concatenation:**
   * If the user chooses '1', concatenates Secondary data (**data\_secondary\_30**, **data\_secondary\_50**, **data\_secondary\_100**) into **data\_section** with a reset index.
   * If the user chooses '2', concatenates Middle data (**data\_middle\_30**, **data\_middle\_50**, **data\_middle\_100**) into **data\_section** with a reset index.
   * If the user chooses '3', concatenates Elementary data (**data\_elementary\_30**, **data\_elementary\_50**, **data\_elementary\_100**) into **data\_section** with a reset index.
4. **Invalid Section Choice Handling:**
   * If the user enters an invalid choice, it prints a message asking the user to enter a valid option (1-3) and returns from the function.



1. **Specify Available Columns:**
   * Defines a list of **available\_columns** containing specific column names for analysis, and appends the option 'all' to it.
2. **Create PrettyTable for Available Columns:**
   * Initializes a **PrettyTable** named **available\_columns\_table** to display available columns in a tabular format.
3. **Populate PrettyTable:**
   * Populates the table with column numbers and names using a loop over the **available\_columns** list.
4. **Print Specified Columns Table:**
   * Prints the table containing the specified columns for analysis.
5. **Define Allowed Columns:**
   * Creates a list of **allowed\_columns** containing valid column names for selection.
6. **User Input for Selected Columns:**
   * Asks the user to enter column numbers (comma-separated) for the analysis.
7. **Verify and Extract Selected Columns:**
   * Validates the entered column numbers and extracts the corresponding column names from the **available\_columns** list.
8. **Verify Selected Columns:**
   * Checks if any valid columns are selected; if not, prints a message and returns from the function.
9. **Verify 'class\_size' Column:**
   * Ensures that the 'class\_size' column is present in the **data\_section** DataFrame; if not, prints an error message and returns from the function.



**1. Check for 'all' in Selected Columns:**

- Checks if 'all' is present in the selected columns. If true, it proceeds to plot mean values against class size for all specified columns in a single bar chart.

**2. Set Figure Size and Colors:**

- Sets the figure size for the bar chart and defines colors using Seaborn's color palette.

**3. Iterate Over Specified Columns:**

- Iterates over each specified column in `available\_columns[:-1]`.

**4. Calculate Mean Values:**

- Calculates mean values for the current column grouped by 'class\_size' in the `data\_section` DataFrame.

**5. Set Bar Width:**

- Manually sets the width of the bars in the bar chart.

**6. Plot Bar Chart:**

- Plots the bar chart for the mean values, adjusting the x-coordinate of each bar for better visualization.

**7. Add Labels on Bars:**

- Adds labels on each bar, displaying the mean value with a specific format.

**8. Set Title and Axes Labels:**

- Sets the title, x-axis label, and y-axis label for the bar chart.

**9. Add Legend:**

- Adds a legend at the top center of the plot, with labels for each specified column.

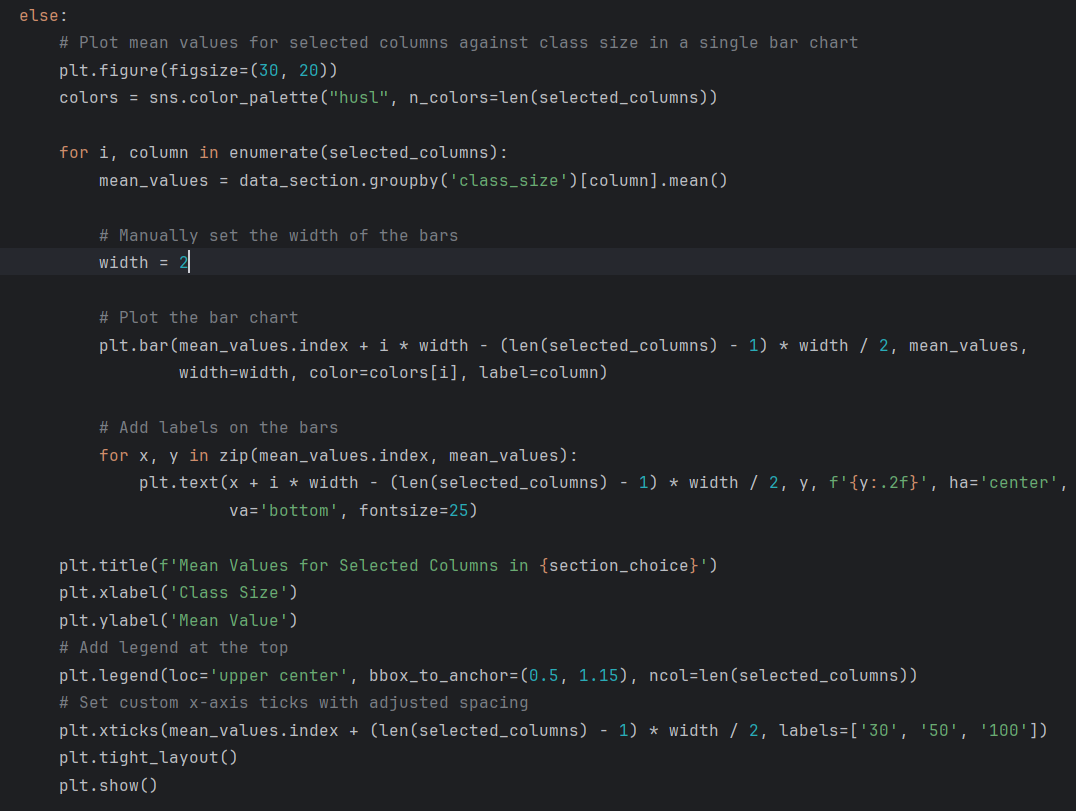
**10. Set Custom X-axis Ticks:**

- Sets custom x-axis ticks with adjusted spacing, representing the class sizes.

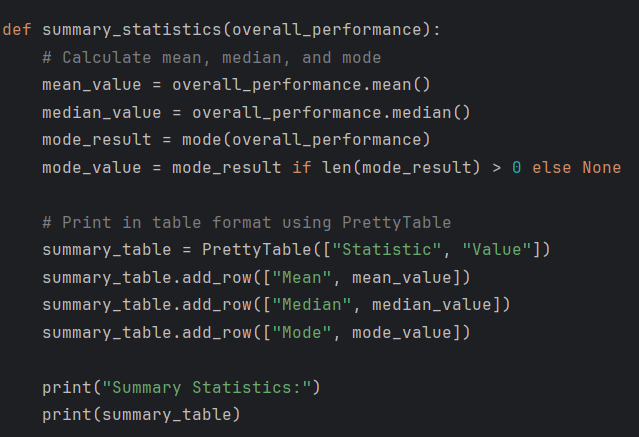
**11. Tight Layout and Show Plot:**

- Adjusts the layout to prevent clipping of titles or labels, then displays the bar chart.

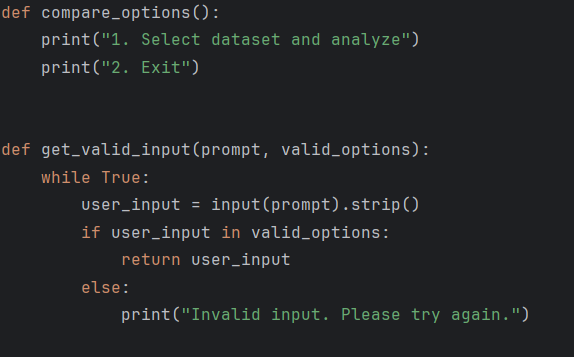
In summary, this code block plots mean values against class size for all specified columns in a single bar chart. The layout and formatting are adjusted for clarity, and a legend is included for column identification.



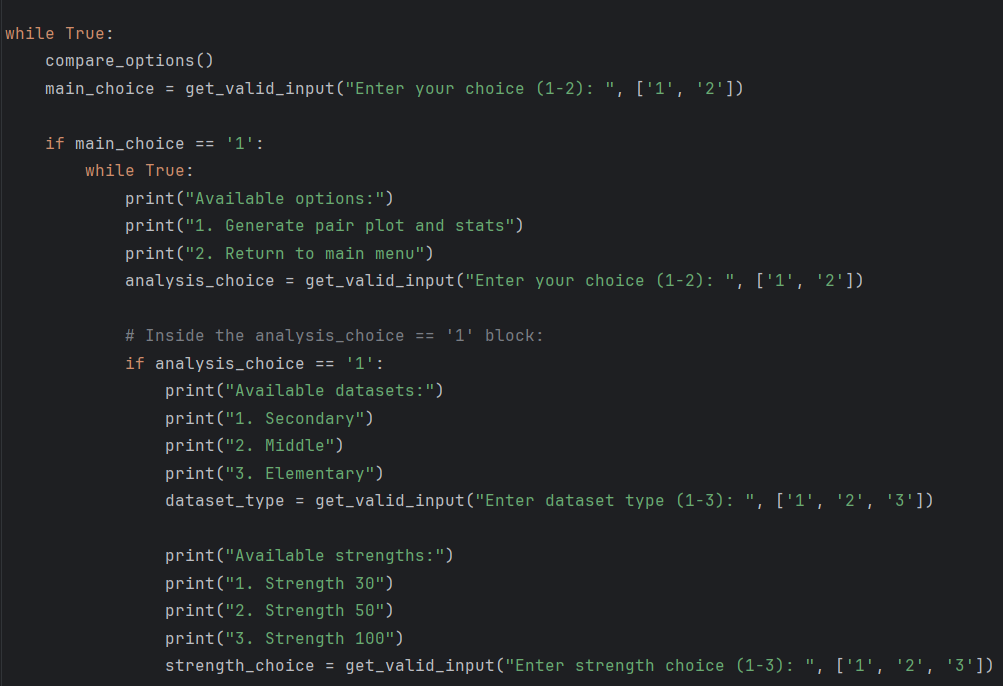
1. **Set Figure Size and Colors:**
   * Sets the figure size for the bar chart and defines colors using Seaborn's color palette based on the number of selected columns.
2. **Iterate Over Selected Columns:**
   * Iterates over each selected column in the **selected\_columns** list.
3. **Calculate Mean Values:**
   * Calculates mean values for the current column grouped by 'class\_size' in the **data\_section** DataFrame.
4. **Set Bar Width:**
   * Manually sets the width of the bars in the bar chart.
5. **Plot Bar Chart:**
   * Plots the bar chart for the mean values, adjusting the x-coordinate of each bar for better visualization.
6. **Add Labels on Bars:**
   * Adds labels on each bar, displaying the mean value with a specific format.
7. **Set Title and Axes Labels:**
   * Sets the title, x-axis label, and y-axis label for the bar chart.
8. **Add Legend:**
   * Adds a legend at the top center of the plot, with labels for each selected column.
9. **Set Custom X-axis Ticks:**
   * Sets custom x-axis ticks with adjusted spacing, representing the class sizes.
10. **Tight Layout:**
    * Adjusts the layout to prevent clipping of titles or labels.
11. **Show Plot:**
    * Displays the bar chart.



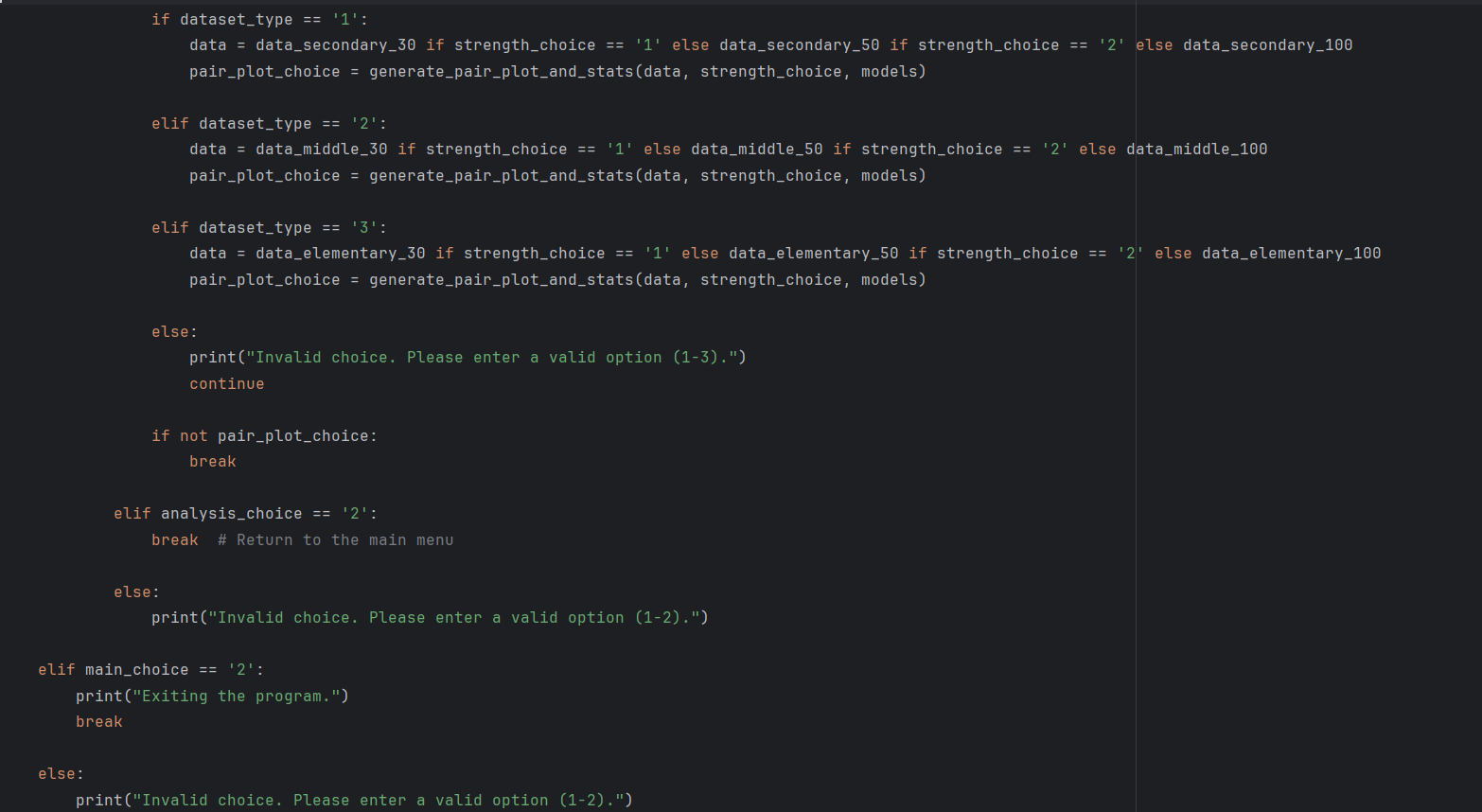
1. **Calculate Mean:**
   * Computes the mean value of the 'overall\_performance' column using the **mean()** function.
2. **Calculate Median:**
   * Computes the median value of the 'overall\_performance' column using the **median()** function.­­­
3. **Calculate Mode:**
   * Uses the **mode()** function to calculate the mode of the 'overall\_performance' column. The result is stored in **mode\_result**.
4. **Handle Mode Result:**
   * Checks if there is a mode by verifying the length of **mode\_result**. If there is a mode, the mode value is assigned to **mode\_value**; otherwise, it remains **None**.
5. **Print Summary Statistics in Table Format:**
   * Utilizes **PrettyTable** to create a table for displaying the calculated mean, median, and mode values.
6. **Print Summary Table:**
   * Prints the summary statistics table with headers "Statistic" and "Value".



1. **compare\_options Function:**
   * Prints options for the user to select from, specifically for dataset analysis or exiting.
2. **get\_valid\_input Function:**
   * Takes two parameters: **prompt** (a string to display to the user) and **valid\_options** (a list of valid input options).
   * Initiates a loop to continuously prompt the user for input until a valid option is provided.
   * Strips any leading or trailing whitespaces from the user's input.
   * Checks if the user's input is within the list of valid options.
   * If the input is valid, the function returns the user's input.
   * If the input is invalid, it prints an error message and prompts the user to try again.



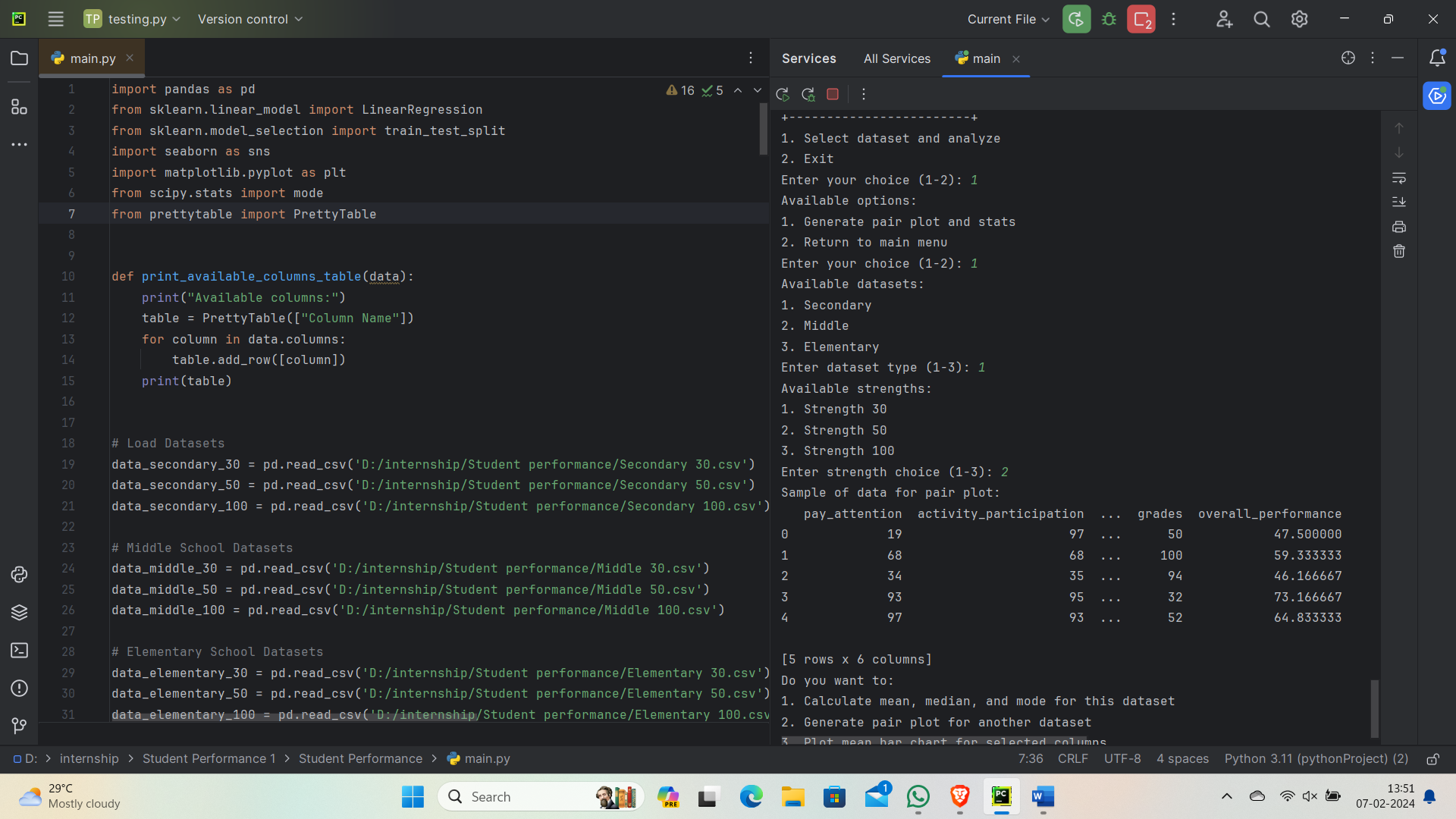
1. **Outer Loop (while True):**
   * Creates an infinite loop for the main menu where the user can choose between dataset analysis (**1**) or exiting (**2**).
2. **Print Main Menu Options:**
   * Calls the **compare\_options** function to print the main menu options.
3. **Get Valid Main Menu Input:**
   * Invokes the **get\_valid\_input** function to get valid user input for the main menu choice (either '1' or '2').
4. **Inner Loop (while True):**
   * Enters another infinite loop for dataset analysis options as long as the user does not choose to return to the main menu.
5. **Print Analysis Menu Options:**
   * Prints options for dataset analysis or returning to the main menu.
6. **Get Valid Analysis Menu Input:**
   * Invokes the **get\_valid\_input** function to get valid user input for the dataset analysis choice (either '1' or '2').
7. **Dataset Analysis Choice Block:**
   * If the user chooses dataset analysis (**analysis\_choice == '1'**), it proceeds with selecting the dataset type and strength.
8. **Print Dataset Type Options:**
   * Prints options for selecting the dataset type (Secondary, Middle, Elementary).
9. **Get Valid Dataset Type Input:**
   * Invokes the **get\_valid\_input** function to get valid user input for the dataset type choice ('1', '2', or '3').
10. **Print Strength Options:**
    * Prints options for selecting the strength (30, 50, 100).
11. **Get Valid Strength Input:**
    * Invokes the **get\_valid\_input** function to get valid user input for the strength choice ('1', '2', or '3').



1. **Dataset Analysis Block (analysis\_choice == '1'):**
   * Checks if the user's choice is to analyze a dataset (**analysis\_choice == '1'**).
2. **Determine Dataset Type and Strength:**
   * Based on the user's choices for dataset type (**dataset\_type**) and strength (**strength\_choice**), it assigns the appropriate dataset (**data**) for analysis.
3. **Generate Pair Plot and Stats:**
   * Calls the **generate\_pair\_plot\_and\_stats** function, passing the selected dataset, strength, and models. The result (**pair\_plot\_choice**) indicates whether the user wants to generate a pair plot for another dataset.
4. **Check Pair Plot Choice:**
   * If **pair\_plot\_choice** is **False**, it breaks out of the inner loop, returning to the main dataset analysis menu.
5. **Return to Main Menu (analysis\_choice == '2'):**
   * If the user chooses to return to the main menu (**analysis\_choice == '2'**), it breaks out of the inner loop.
6. **Invalid Analysis Choice:**
   * If the user enters an invalid choice, it prints an error message and continues to the next iteration of the loop.
7. **Main Menu Block (main\_choice == '2'):**
   * If the user's main menu choice is to exit (**main\_choice == '2'**), it prints an exit message and breaks out of the outer loop, terminating the program.
8. **Invalid Main Menu Choice:**
   * If the user enters an invalid choice in the main menu, it prints an error message.

**Output:**

* **Select dataset and analyze:**



* **Menu Selection:**

The user is presented with two options:

1. Select a dataset and analyze
2. Exit

The user entered 1, indicating the choice to select a dataset and analyze.

* **Submenu - Dataset Analysis:**

After choosing to analyze a dataset, the user is presented with the following options:

1. Generate pair plot and stats
2. Return to the main menu

The user entered 1, indicating the choice to generate pair plots and statistics.

* **Dataset Type Selection:**

The program presents the user with available dataset types:

1. Secondary
2. Middle
3. Elementary

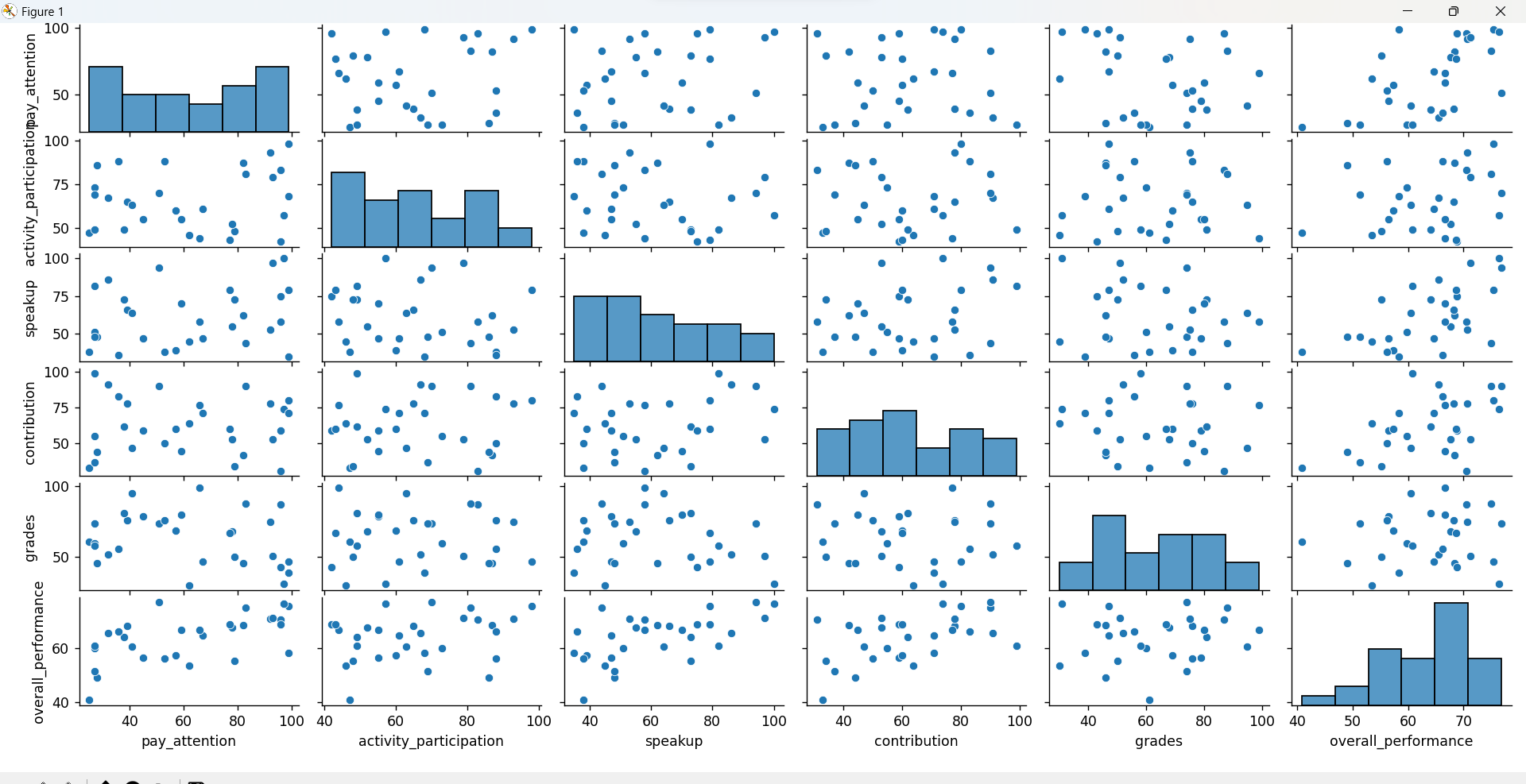
The user entered 1, indicating the choice for the Secondary dataset type.

* **Strength Choice:**

For the selected dataset type (Secondary), the program offers strength options:

1. Strength 30
2. Strength 50
3. Strength 100

The user entered 2, indicating the choice for Strength 50.



* **Sample Data for Pair Plot:**

The program then displays a sample of the dataset, presumably with the selected strength (Strength 50), containing columns such as:

pay\_attention

activity\_participation

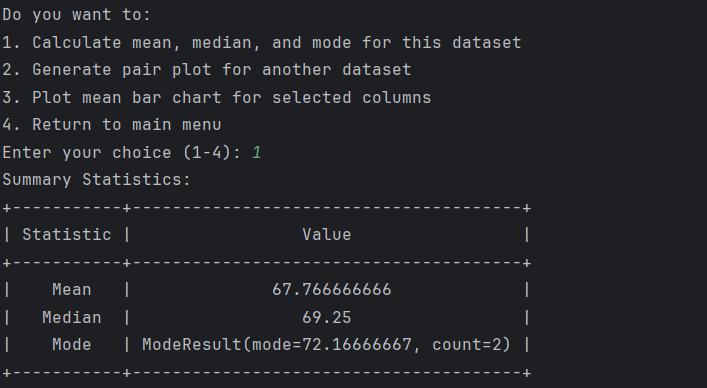
...

grades

overall\_performance

The screenshot shows a matrix of plots that visualize the relationships between different student behaviors and performance metrics

* **Calculate mean, median, and mode for this dataset:**



The summary statistics provide key numerical measures that describe the central tendency and distribution of the "overall\_performance" variable in the "Secondary" dataset with a strength of 30. Here's an explanation of each statistic:

* **Mean (Average):**

The mean is calculated by adding up all the values in the "overall\_performance" column and dividing by the number of observations.

Mean = 67.766666666

* **Median (Middle Value):**

The median is the middle value of the "overall\_performance" column when the data is sorted in ascending order.

Median = 69.25

* **Mode (Most Frequent Value):**

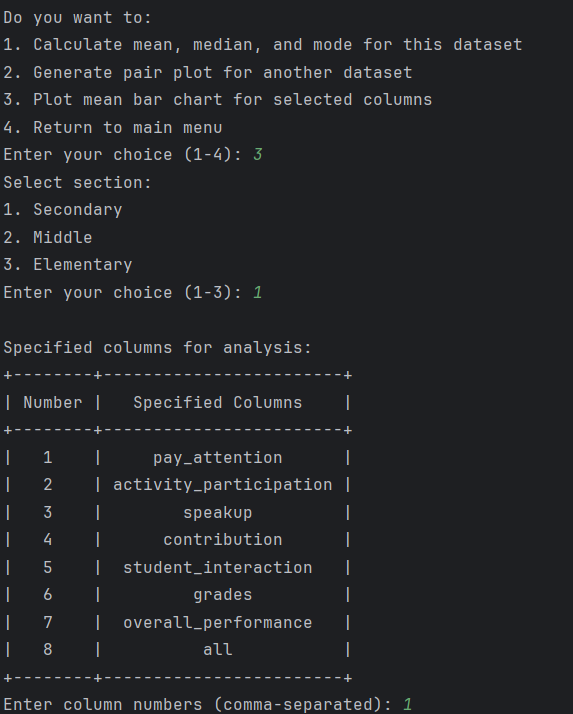
The mode represents the value that appears most frequently in the "overall\_performance" column.

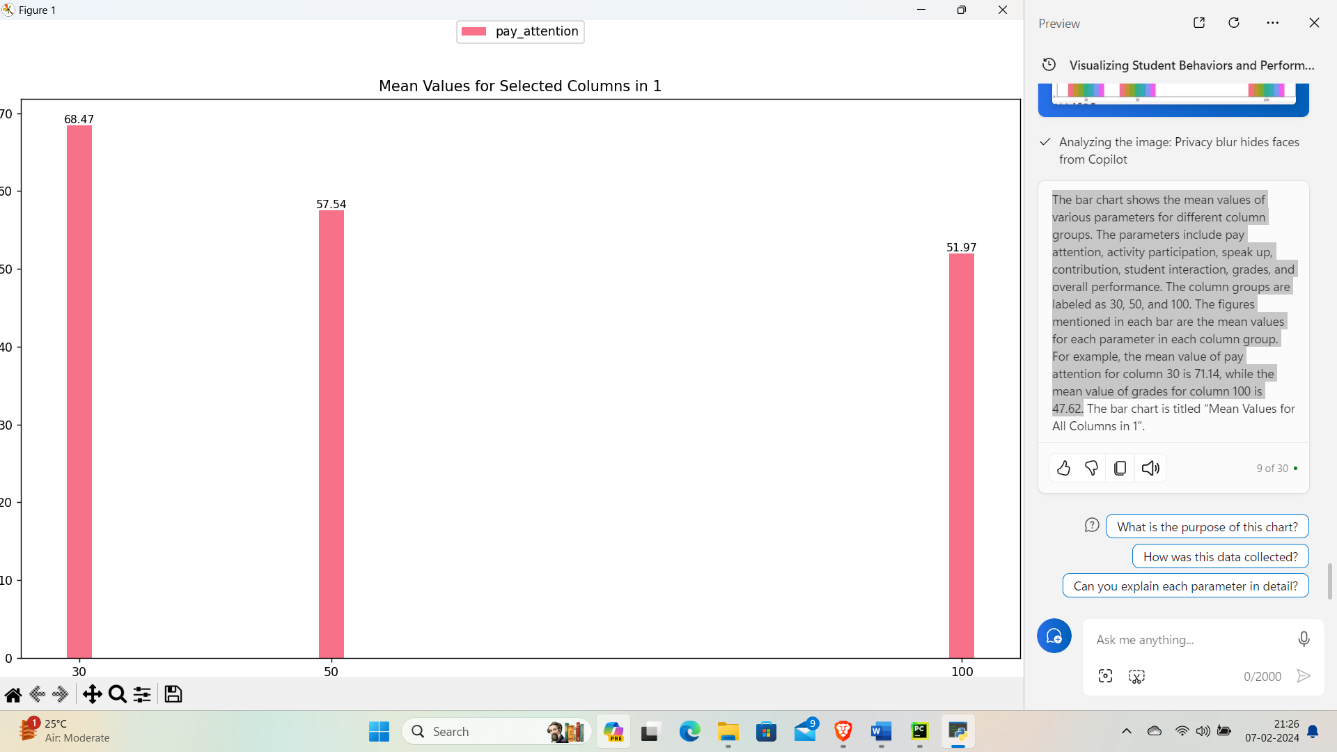
In this case, the mode is approximately 72.16666667, and it occurs 2 times.

This the output for the class Secondary with the strength of 30, similarly it works for the remaining class and all the strength.

* **Plot mean bar chart for selected columns:**

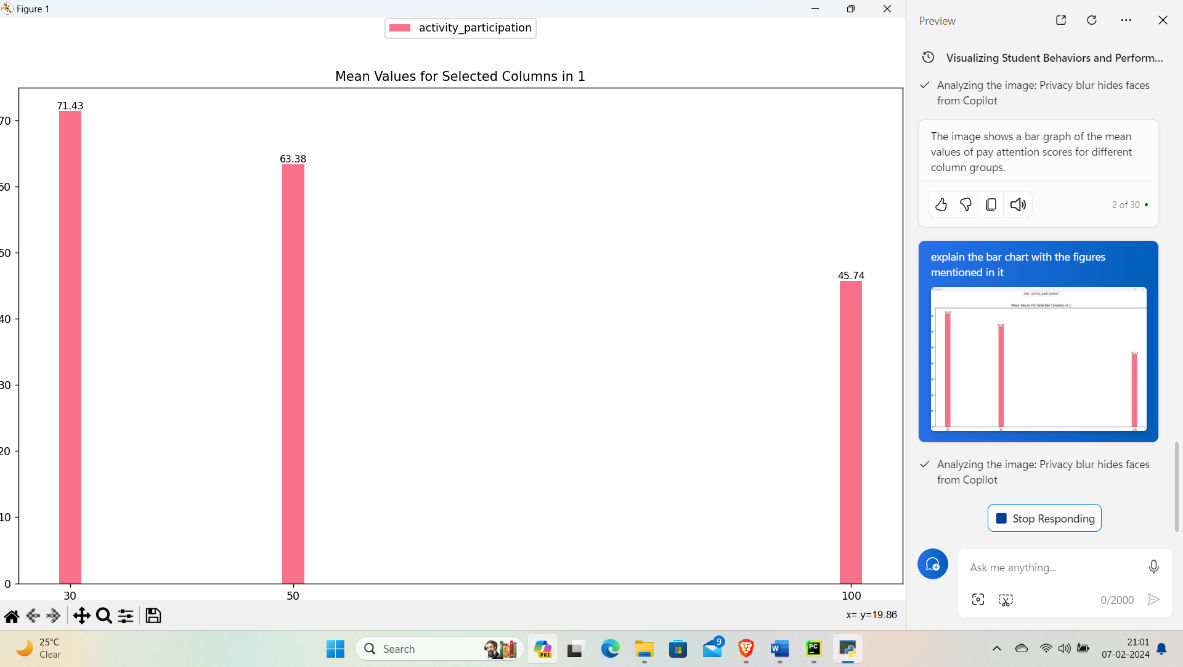
1. **Pay attention:**





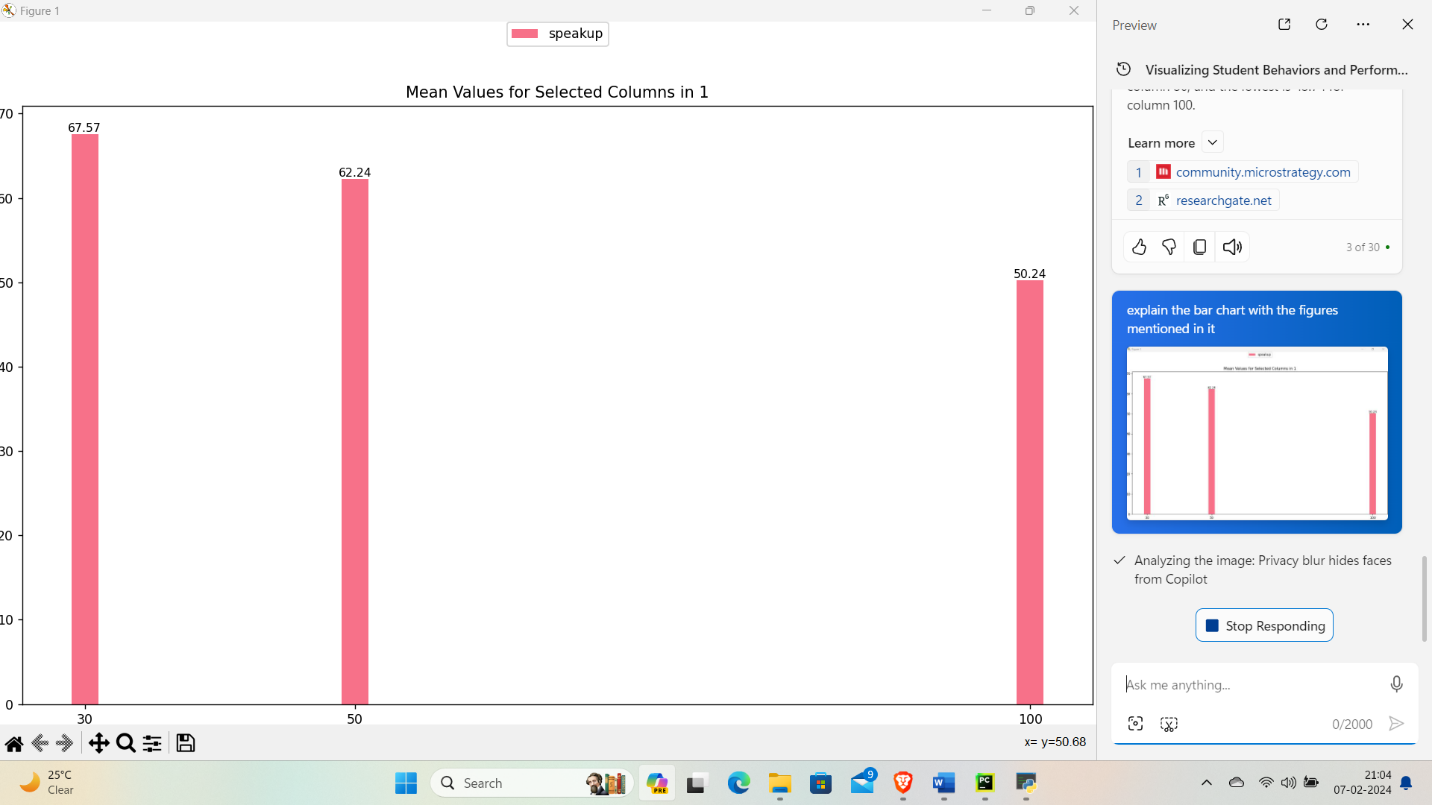
The image shows a bar graph of the mean values of pay attention scores for different column groups.

1. **Activity participation:**



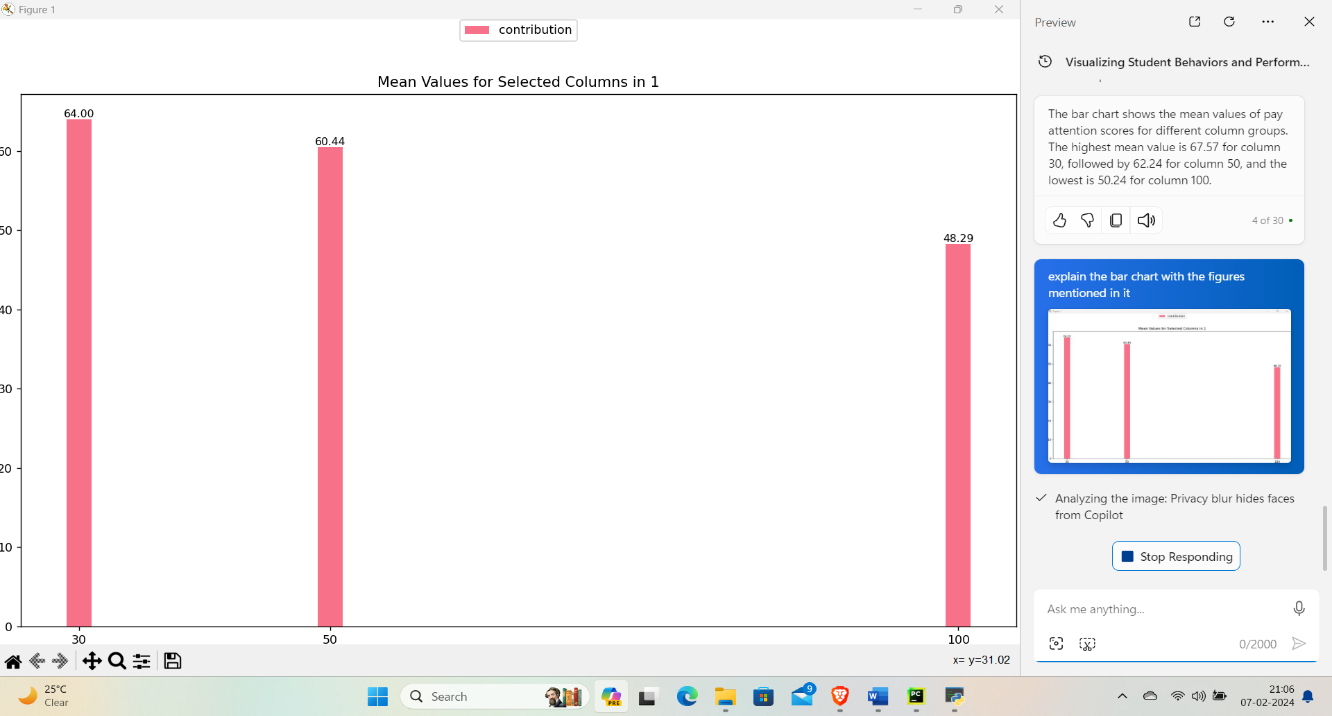
The bar chart shows that the average activity participation level decreases as the column number increases. The highest mean value is 71.43 for column 30, followed by 63.38 for column 50, and the lowest is 45.74 for column 100.

1. **Speakup:**



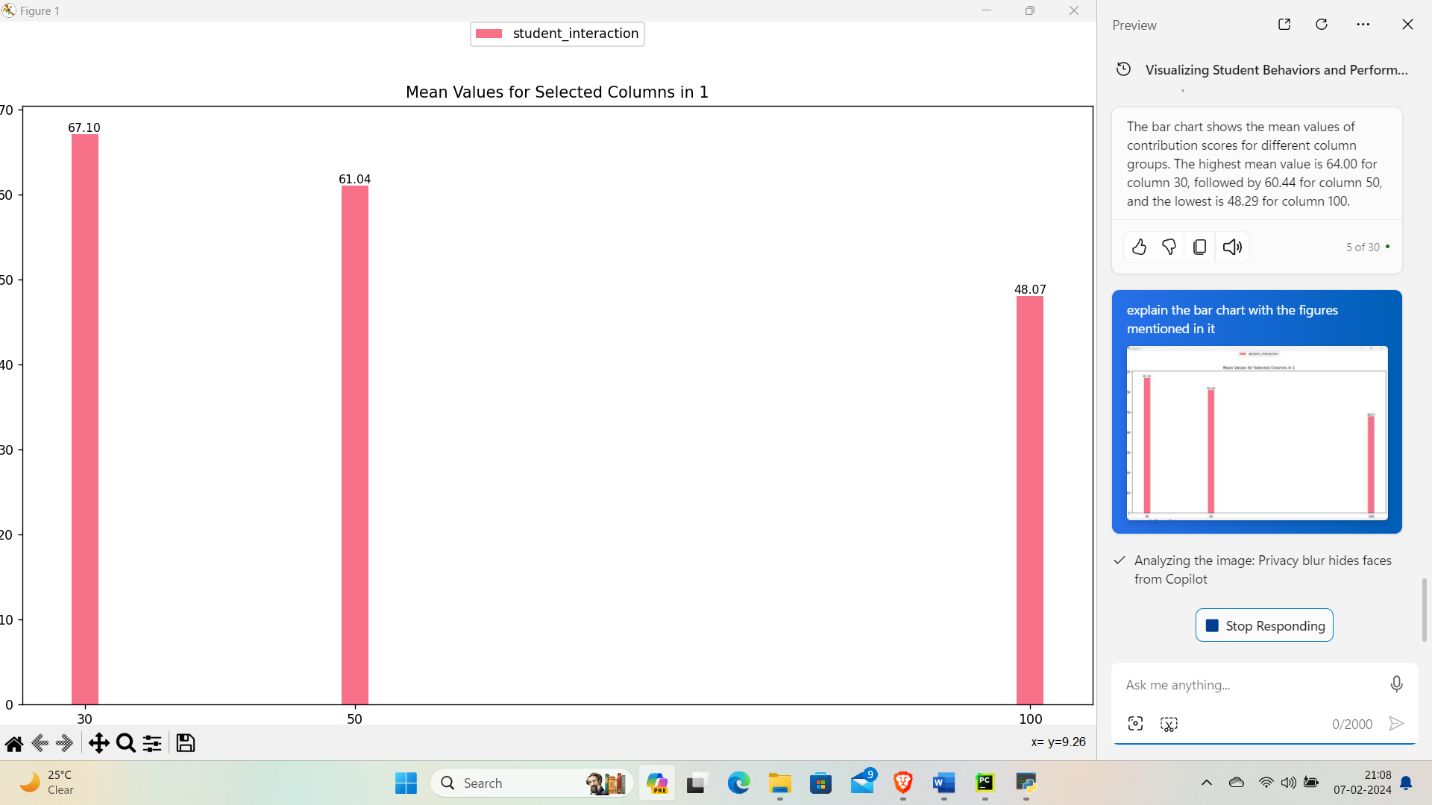
The bar chart shows the mean values of pay attention scores for different column groups. The highest mean value is 67.57 for column 30, followed by 62.24 for column 50, and the lowest is 50.24 for column 100.

1. **Contribution:**



The bar chart shows the mean values of contribution scores for different column groups. The highest mean value is 64.00 for column 30, followed by 60.44 for column 50, and the lowest is 48.29 for column 100.

1. **Student interaction:**



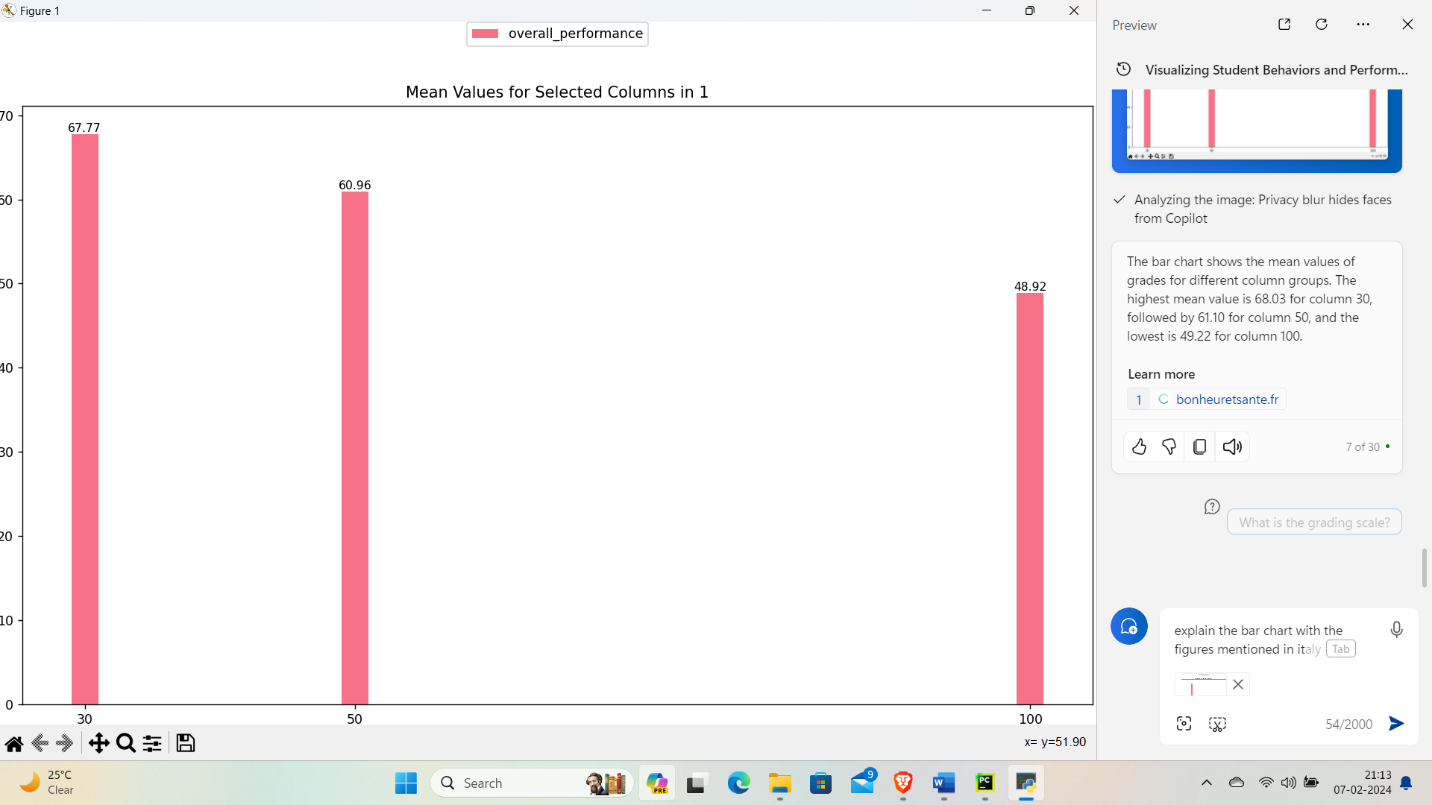
The bar chart shows the mean values of student interaction scores for different column groups. The highest mean value is 67.10 for column 30, followed by 61.04 for column 50, and the lowest is 48.07 for column 100.

1. **Grades**



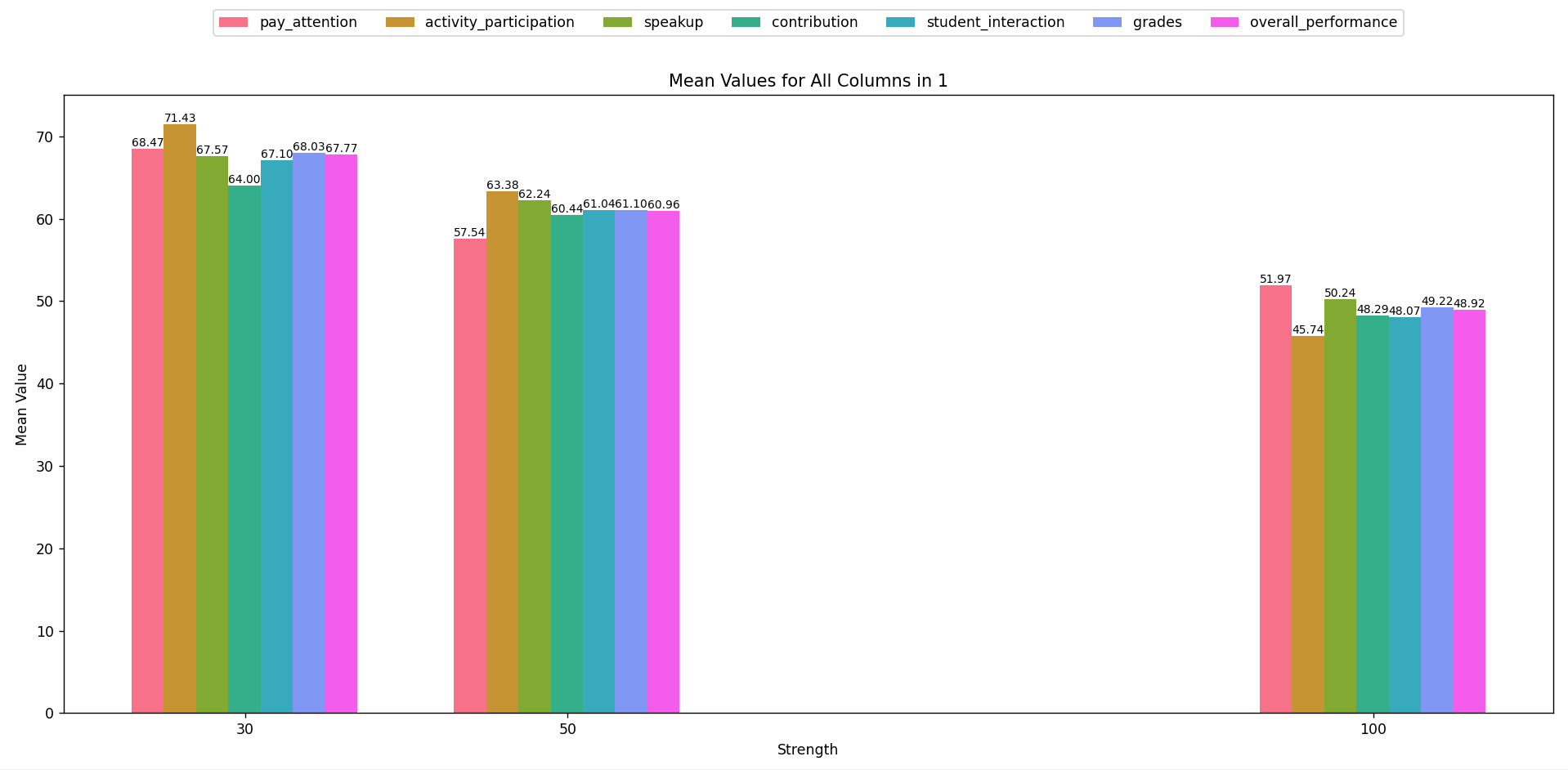
The bar chart shows the mean values of grades for different column groups. The highest mean value is 68.03 for column 30, followed by 61.10 for column 50, and the lowest is 49.22 for column 100.

1. **Overall performance:**



The bar chart shows the mean values of overall performance for different column groups. The highest mean value is 67.77 for column 30, followed by 60.96 for column 50, and the lowest is 48.92 for column 100.

1. **All**



The bar chart shows the mean values of various parameters for different column groups. The parameters include pay attention, activity participation, speak up, contribution, student interaction, grades, and overall performance. The column groups are labeled as 30, 50, and 100. The figures mentioned in each bar are the mean values for each parameter in each column group. For example, the mean value of pay attention for column 30 is 71.14, while the mean value of grades for column 100 is 47.62.

* The above Plot mean bar chart is for the selected columns shows the output of Secondary section , similarly the program runs for the Elementary and the Middle section and give the similar output.