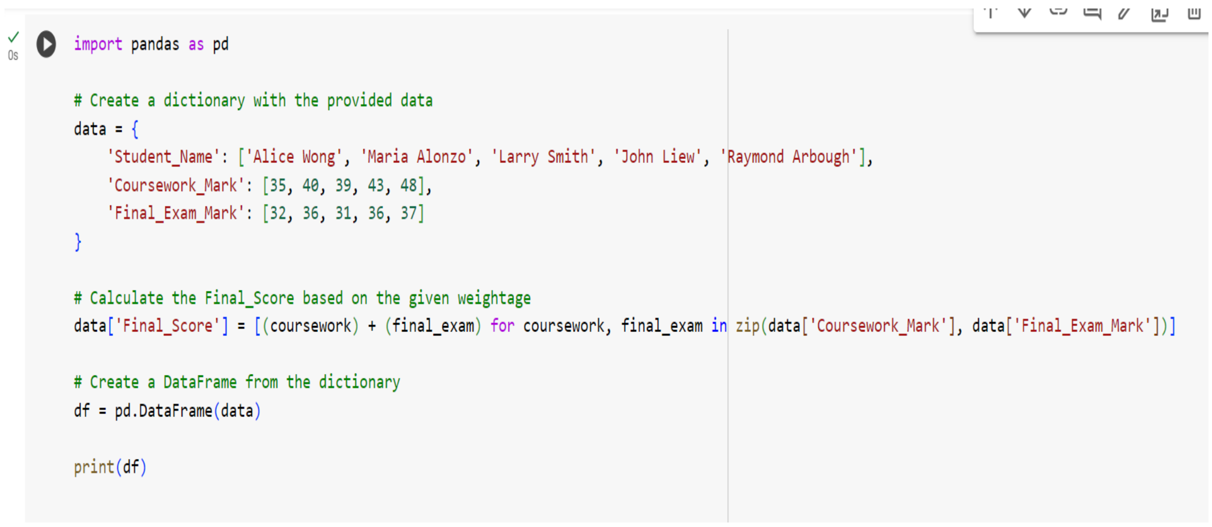
IST2334 Web and Network Analytics PRACTICAL 1

1. Creating a data frame

a. Create a data frame consisting of the following data for students taking the course IST2334 Web and Network Analytics. [Assume that the marks provided are according to the weightage assigned to both assessment components. The coursework component is 60% and the final exam component 40%.]

Student_Name	Coursework_Mark	Final_Exam_Mark	Final_Score
Alice Wong	35	32	67
Maria Alonzo	40	36	76
Larry Smith	39	31	70
John Liew	43	36	79
Raymond Arbough	48	37	85

Solution



```python
import pandas as pd

# Create a dictionary with the provided data
data = {
    'Student_Name': ['Alice Wong', 'Maria Alonzo', 'Larry Smith', 'John Liew', 'Raymond Arbough'],
    'Coursework_Mark': [35, 40, 39, 43, 48],
    'Final_Exam_Mark': [32, 36, 31, 36, 37]
}

# Calculate the Final_Score based on the given weightage
data['Final_Score'] = [(coursework) + (final_exam) for coursework, final_exam in zip(data['Coursework_Mark'], data['Final_Exam_Mark'])]

# Create a DataFrame from the dictionary
df = pd.DataFrame(data)

print(df)
```

Explanation:

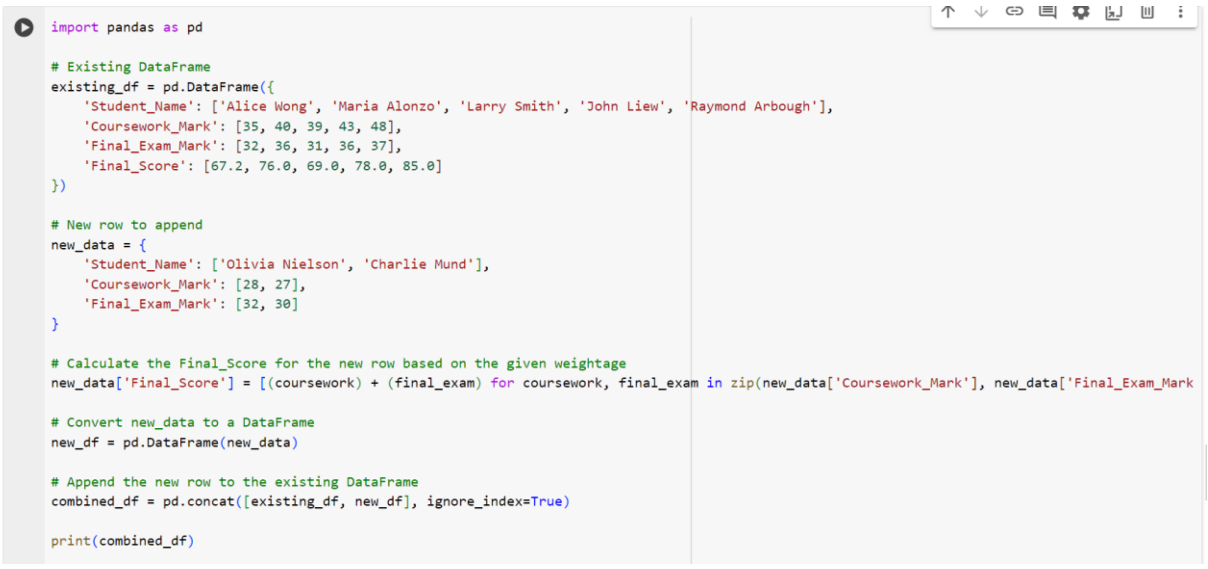
- We first create a dictionary **‘data’** containing the student names, coursework marks, and final exam marks.
- We then calculate the final score for each student by applying the weightage (60% for coursework and 40% for the final exam) and add it to the dictionary as the 'Final_Score' column.
- Finally, we create a DataFrame **‘df’** from the dictionary using **‘pd.DataFrame(data)’.**

2. Append to a Data Frame

a. Append the following information as a new row into the data frame created in Question 1.

Student_Name	Coursework_Mark	Final_Exam_Mark	Final_Score
Olivia Nielson	28	32	60
Charlie Mund	27	30	57

Solution



```python
import pandas as pd

# Existing DataFrame
existing_df = pd.DataFrame({
    'Student_Name': ['Alice Wong', 'Maria Alonzo', 'Larry Smith', 'John Liew', 'Raymond Arbough'],
    'Coursework_Mark': [35, 40, 39, 43, 48],
    'Final_Exam_Mark': [32, 36, 31, 36, 37],
    'Final_Score': [67.2, 76.0, 69.0, 78.0, 85.0]
})

# New row to append
new_data = {
    'Student_Name': ['Olivia Nielson', 'Charlie Mund'],
    'Coursework_Mark': [28, 27],
    'Final_Exam_Mark': [32, 30]
}

# Calculate the Final_Score for the new row based on the given weightage
new_data['Final_Score'] = [(coursework) + (final_exam) for coursework, final_exam in zip(new_data['Coursework_Mark'], new_data['Final_Exam_Mark

# Convert new_data to a DataFrame
new_df = pd.DataFrame(new_data)

# Append the new row to the existing DataFrame
combined_df = pd.concat([existing_df, new_df], ignore_index=True)

print(combined_df)
```

Explanation:

We first define the existing DataFrame **'existing_df'** with the data provided previously.

Then, we define the new row **'new_data'** with the provided information for Olivia Nielson and Charlie Mund.

We calculate the Final_Score for the new row based on the given weightage and add it to **'new_data'.**

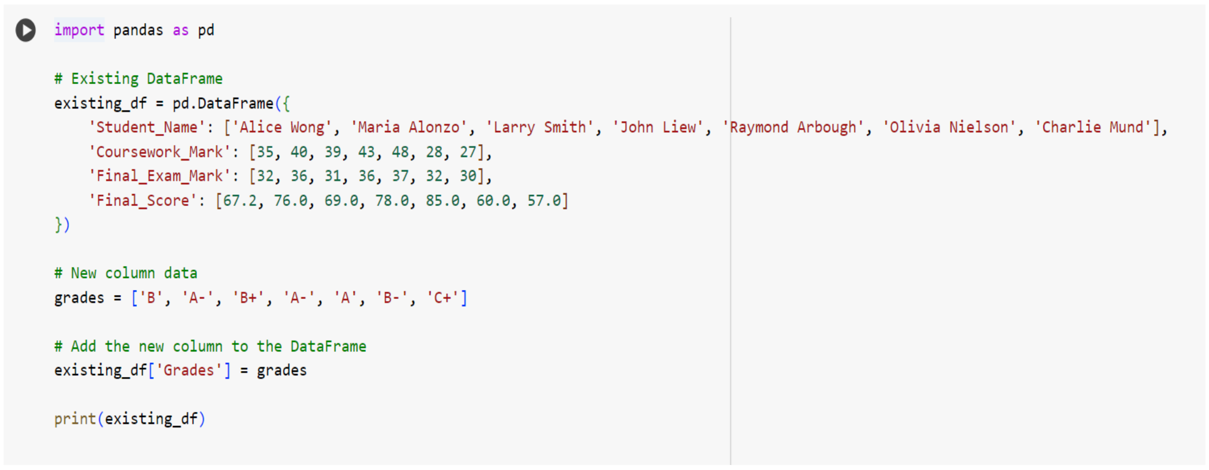
Next, we convert **'new_data'** to a DataFrame **'new_df'**.

Finally, we append **'new_df'** to **'existing_df'** using the **'.concat()'** method with **'ignore_index=True'** to reindex the resulting DataFrame. The combined DataFrame is stored in **'combined_df'.**

b. Append/Add the following information as a new column into the data frame in Question 2 (a).

Grades	B	A-	B+	A-	A	B-	C+

Solution



```python
import pandas as pd

# Existing DataFrame
existing_df = pd.DataFrame({
    'Student_Name': ['Alice Wong', 'Maria Alonzo', 'Larry Smith', 'John Liew', 'Raymond Arbough', 'Olivia Nielson', 'Charlie Mund'],
    'Coursework_Mark': [35, 40, 39, 43, 48, 28, 27],
    'Final_Exam_Mark': [32, 36, 31, 36, 37, 32, 30],
    'Final_Score': [67.2, 76.0, 69.0, 78.0, 85.0, 60.0, 57.0]
})

# New column data
grades = ['B', 'A-', 'B+', 'A-', 'A', 'B-', 'C+']

# Add the new column to the DataFrame
existing_df['Grades'] = grades

print(existing_df)
```

Explanation:

We define the existing DataFrame **‘existing_df’** with the data provided previously.

Then, we define a list of grades **‘grades’.**

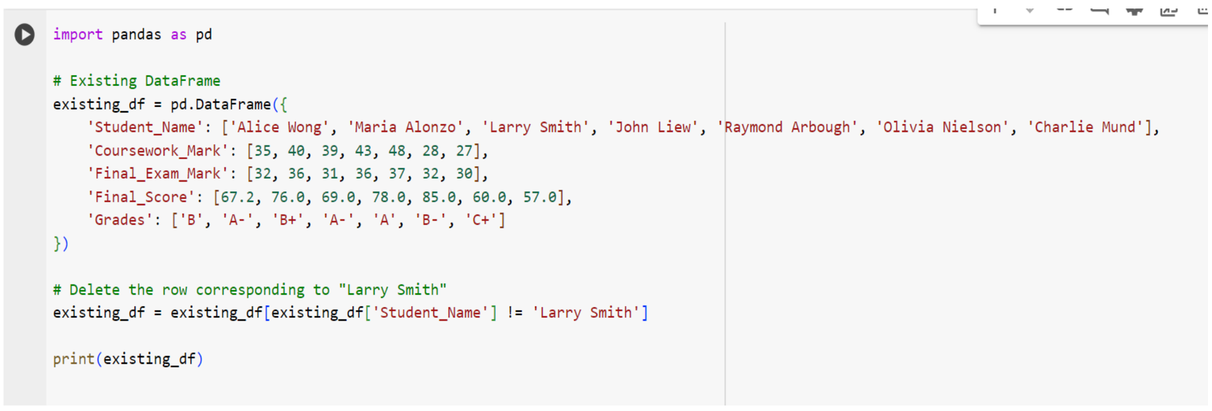
We add the new column **'Grades'** to the DataFrame **‘existing_df’** and assign **‘grades’** as its values.

The DataFrame **‘existing_df’** now includes the new column **'Grades'.**

3. Deleting data from a Data Frame.

a. Delete the student “Larry Smith” from the data frame.

Solution



```python
import pandas as pd

# Existing DataFrame
existing_df = pd.DataFrame({
    'Student_Name': ['Alice Wong', 'Maria Alonzo', 'Larry Smith', 'John Liew', 'Raymond Arbough', 'Olivia Nielson', 'Charlie Mund'],
    'Coursework_Mark': [35, 40, 39, 43, 48, 28, 27],
    'Final_Exam_Mark': [32, 36, 31, 36, 37, 32, 30],
    'Final_Score': [67.2, 76.0, 69.0, 78.0, 85.0, 60.0, 57.0],
    'Grades': ['B', 'A-', 'B+', 'A-', 'A', 'B-', 'C+']
})

# Delete the row corresponding to "Larry Smith"
existing_df = existing_df[existing_df['Student_Name'] != 'Larry Smith']

print(existing_df)
```

Explanation:

We define the existing DataFrame **‘existing_df’** with the data provided previously.

We use boolean indexing to filter out the row where the **'Student_Name'** column is equal to "Larry Smith" using the condition **‘existing_df['Student_Name'] != 'Larry Smith'.**

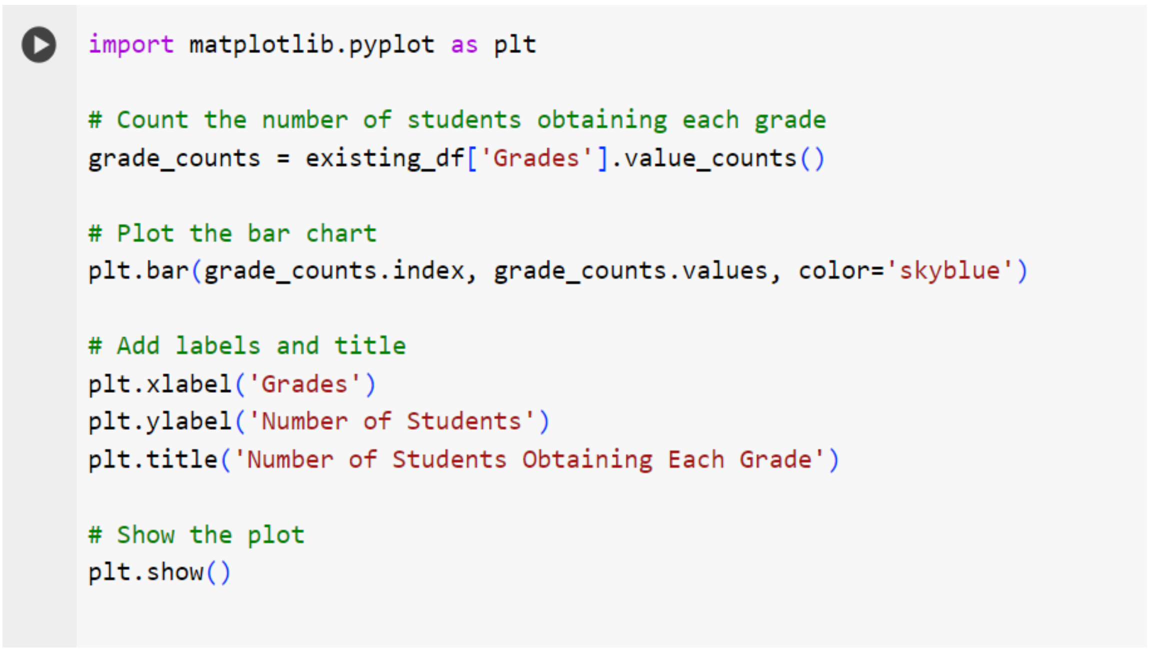
The resulting DataFrame, excluding the row corresponding to **"Larry Smith",** is stored back in **‘existing_df’.**

The DataFrame ‘**existing_df’** now no longer includes the row corresponding to "Larry Smith".

4. Displaying and Plotting the Data

a. Plot a bar chart with the number of students obtaining each of the grades on the y-axis and the list of possible grades on the x-axis

Solution



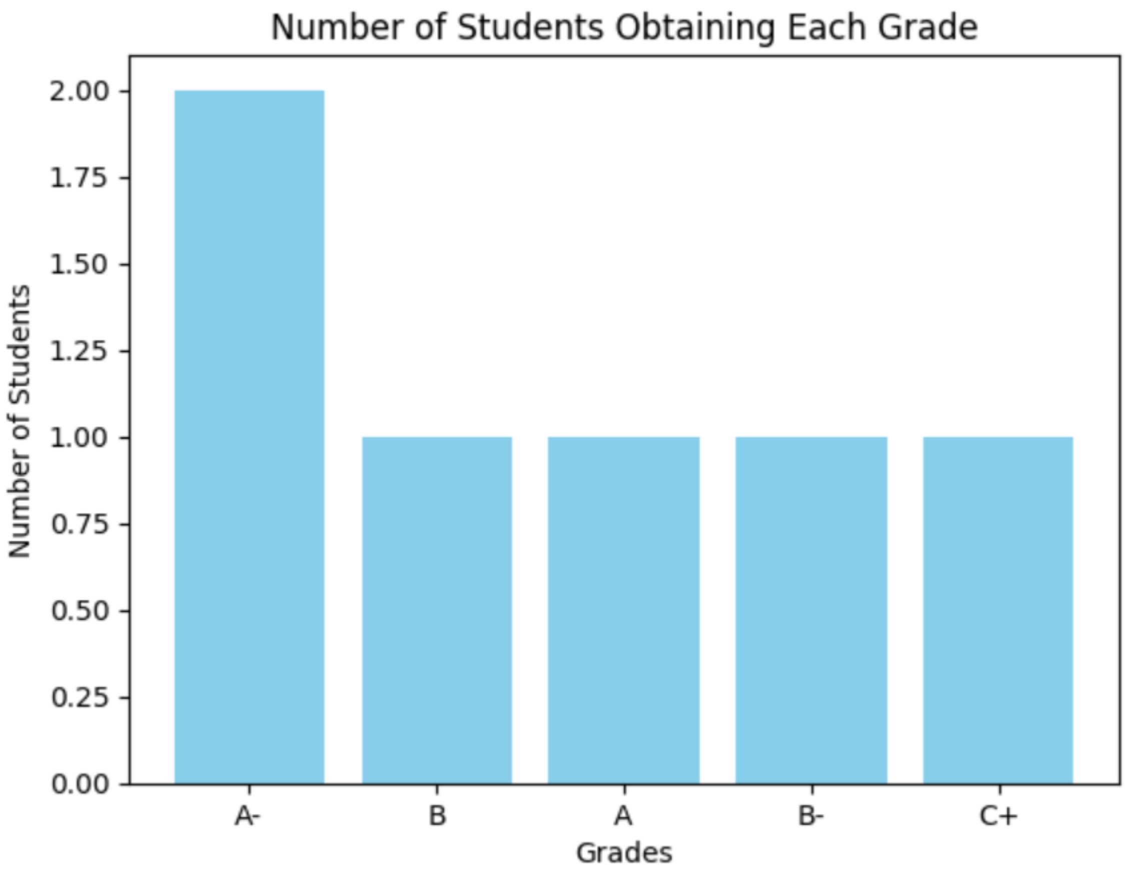
```python
import matplotlib.pyplot as plt

# Count the number of students obtaining each grade
grade_counts = existing_df['Grades'].value_counts()

# Plot the bar chart
plt.bar(grade_counts.index, grade_counts.values, color='skyblue')

# Add labels and title
plt.xlabel('Grades')
plt.ylabel('Number of Students')
plt.title('Number of Students Obtaining Each Grade')

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



Explanation:

We use the **'value_counts()'** method to count the number of occurrences of each grade in the 'Grades' column of the DataFrame.

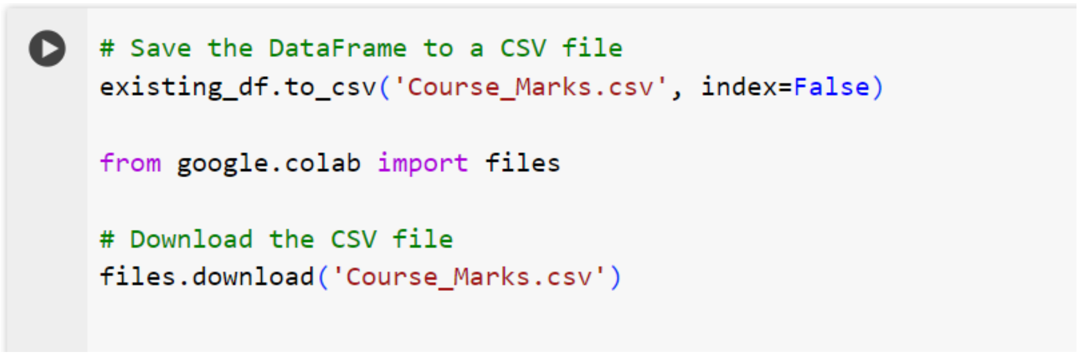
Then, we plot a bar chart using **'plt.bar()',** where **'grade_counts.index'** contains the grades on the x-axis, and **'grade_counts.values'** contains the corresponding counts on the y-axis.

We add labels to the x-axis and y-axis using **'plt.xlabel()'** and **'plt.ylabel()'** respectively.

We add a title to the plot using **'plt.title()'**.

Finally, we display the plot using **'plt.show()'.**

5. Save and download the csv file



```python
# Save the DataFrame to a CSV file
existing_df.to_csv('Course_Marks.csv', index=False)

from google.colab import files

# Download the CSV file
files.download('Course_Marks.csv')
```