Topic summary/Purpose:

These instructions will walk my target through using Google Colab to create a scatterplot with a line of best fit and a visually appealing design that could be used in a presentation.

Audience Analysis:

My target audience is students in the 'Introduction to Data Science' class, CMSC320. The class has two prerequisites of 'Introduction to Computer Systems' and 'Discrete Structures', CMSC216 and CMSC250. So the students in this class have a strong understanding of object-oriented programming languages and the mathematics behind computer science. CMSC320 prepares students for a job as a data analyst, which is a job which partially consists of giving presentations of the findings of one's data analysis to colleagues, superiors, and clients. These presentations benefit from clear visualizations of the findings, and a good-looking layout will make one's audience more receptive to the findings.

In the class, we were expected to make visually appealing graphs to corroborate our findings, but since this was secondary to the data analysis the class was focused on teaching, students were expected to figure that out without guidance. As I've gone through this class before, I figured out how to do so, and can share the knowledge with future CMSC320 students so they can focus more on data analysis.

How to Create a Good-Looking Graph from a Pandas Database using Google Colab

Introduction:

A graph is a useful tool to visualize a large amount of data and make it easier to draw conclusions from. Graphs may have been used earlier in the data analysis process to give you a gist of the trends in the data, but the ones you make for yourself typically wouldn't be shown to an audience. In a presentation, there's an additional level of style and professionalism that needs to be incorporated into the graph to make it an effective visual to use on others. This will be a step-by-step guide on how to make a scatter plot with a line of best-fit that will look professional in a presentation.

Technical Background:

- Must know how to read graphs
- Should have basic knowledge of Python, the pandas library, and scikit-learn

Materials:

- Google Colab and Google Slides
- .csv file with the data being used
- Imports:

```
from matplotlib import pyplot as plt

from sklearn.datasets import make_regression

from sklearn.linear_model import LinearRegression

import pandas as pd
```

Warning:

Use data that's been cleaned of outliers. If you have outliers, that could throw off your line of best fit and worsen the graph's ability to get your point across.

Steps:

Creating the Graph

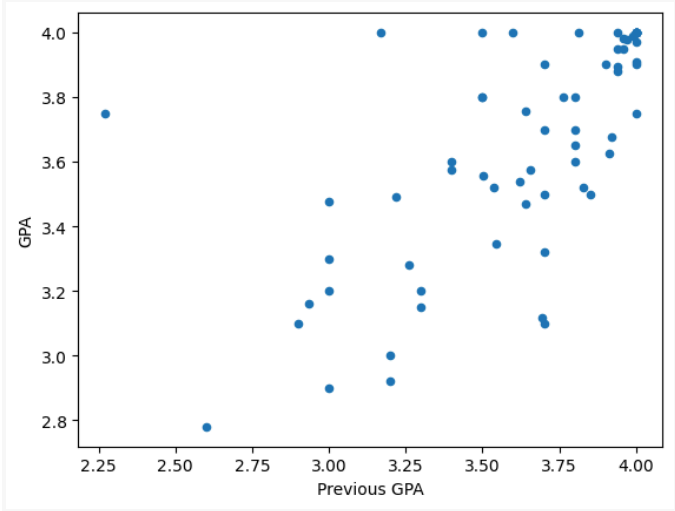
1. Decide which finding you're representing. Scatter plots are most useful for displaying correlations, so it's recommended that you choose two features, or a feature and a label, that have a strong positive or negative correlation.

Factor	Coefficient
State	0.01502
Major	0.07261
Start	0.0008
Private school	0.03272
Scholarship	0.09183
Parent in tech	0.01021
Study with friends	0.01463
Notetaking	0.00945
Credits	0.01294
Screentime	0.00126
TikTok	0.00951
Sleep	0.02259
Previous GPA	0.53523

Previous GPA has the highest coefficient with GPA in a linear regression, so it will likely look the best in our graph

2. Create the scatterplot of your two features, or feature and label. If you're graphing a feature and label, the feature should be the x-axis, and the label should be the y-axis. Otherwise, you can pick the order you prefer.
    - Your graph will look like this if the style is set to the default:

```
df_vector.plot.scatter(x='Previous GPA', y='GPA')
```



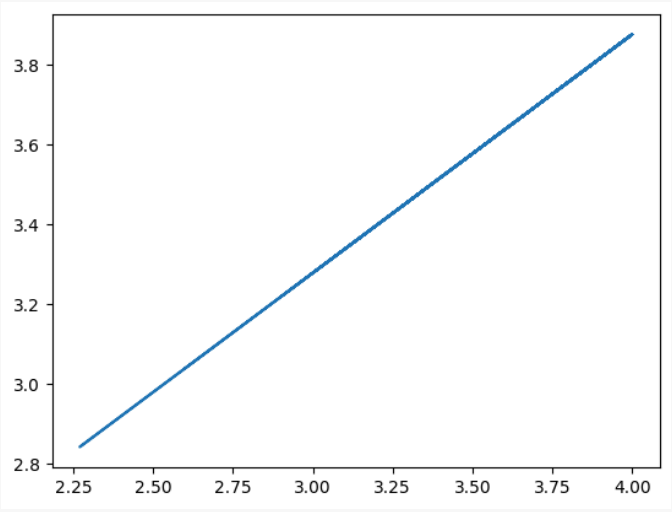
3. Create ‘x’ and ‘y’ variables to match what you decided on in step 2.

```
x = df_vector["Previous GPA"].values.reshape(-1, 1)
y = df_vector["GPA"]
```

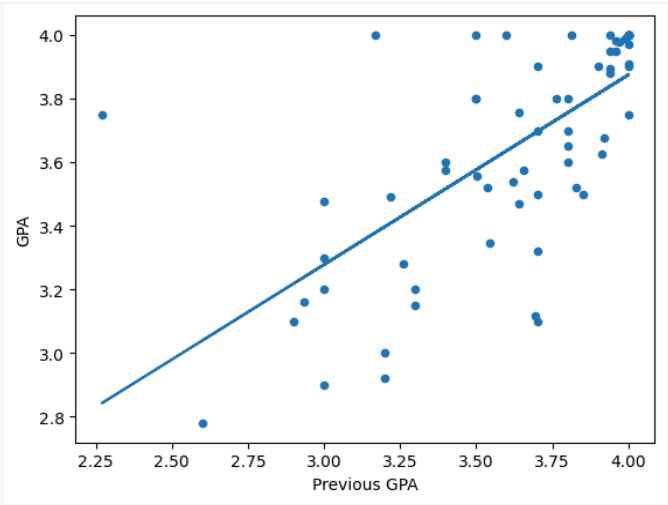
★ The reshape function is necessary to turn the pandas series into a 2d container for the next step.

4. Use linear regression to create a prediction for y based on x in the form of a linear function.

```
lin_reg = LinearRegression()
lin_reg.fit(x, y)
y_pred = lin_reg.predict(x)
plt.plot(x, y_pred)
```



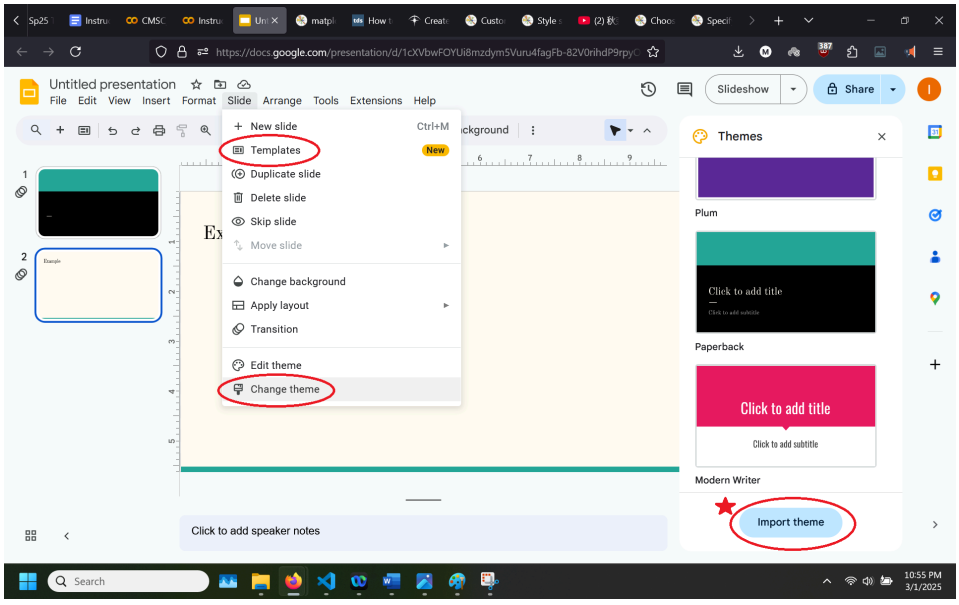
5. Plot the two graphs together using `df_vector.plot.scatter(x=, y=)` and `plt.plot(x, y_pred)` from steps 2 and 4. In Google Colab, they need to be in the same code block to appear together.



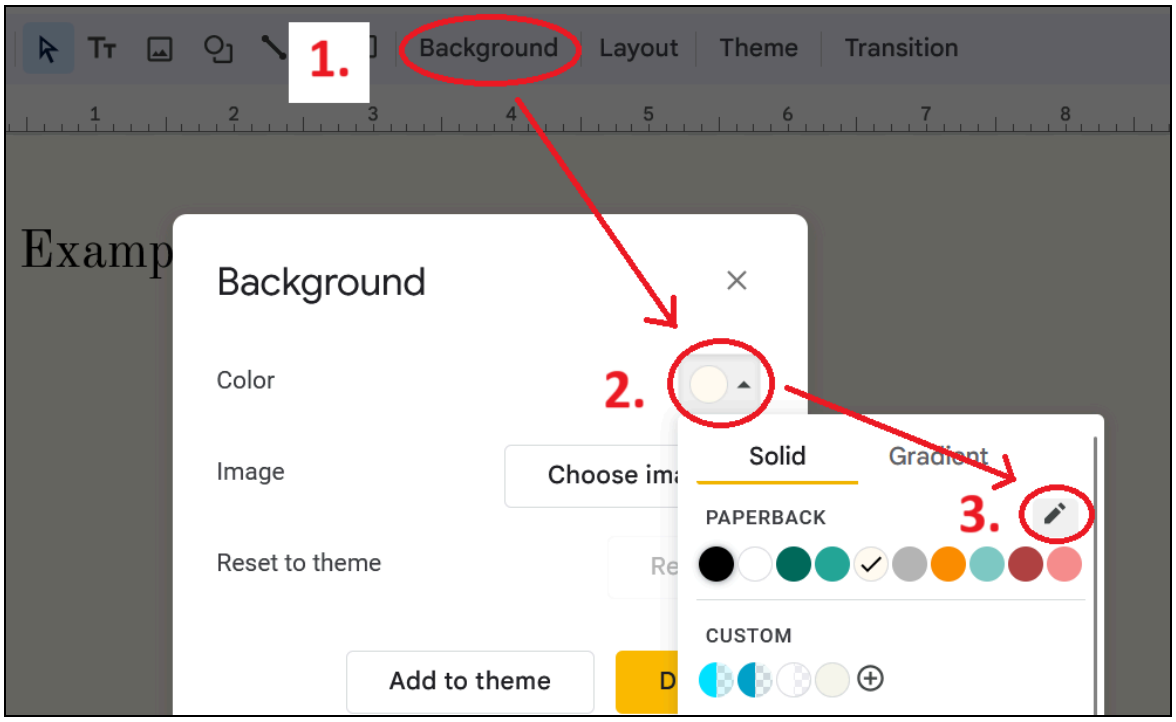
Adding Visual Appeal

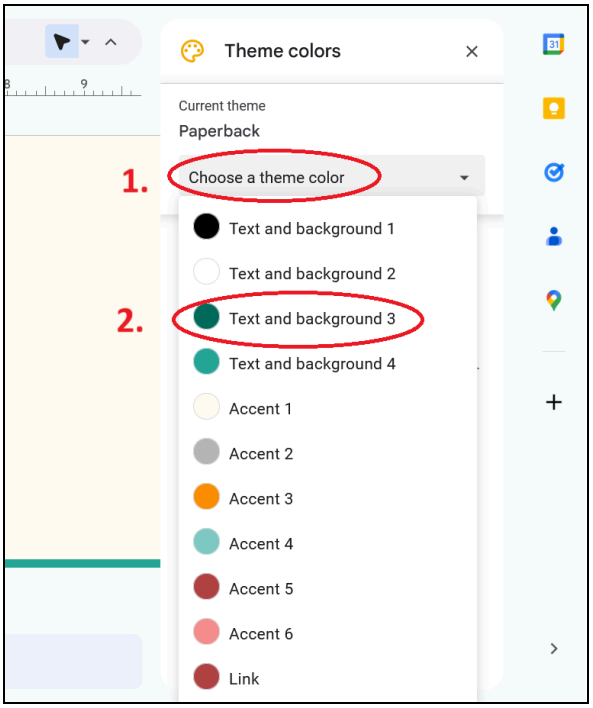
6. Decide on the look of your presentation. Find a style you like by going to Slide > Template or Side > Change Theme.

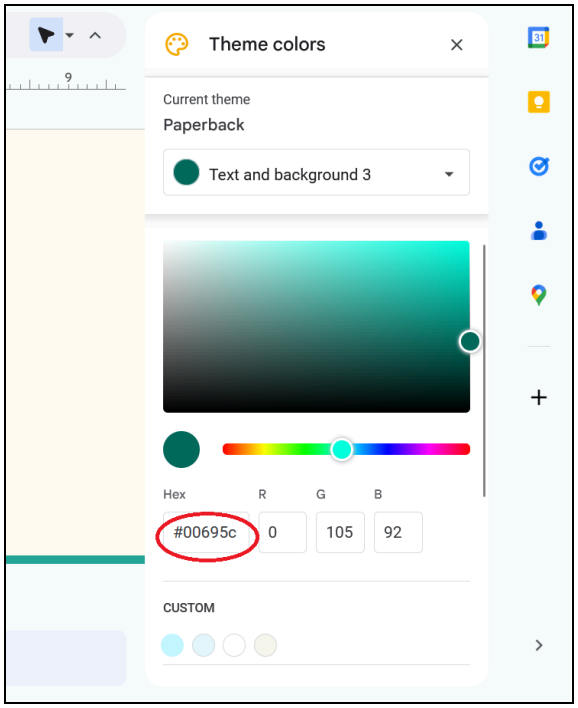
   ★ You can also download a theme elsewhere and go to Slide > Change Theme > Import Theme to add it to Google Slides.



7. Find the colors comprising your template. Click Slide, then Change Background, Color, and the pencil icon. From there, you can click on ‘Choose a theme color’ and pick a color to view its hex value.

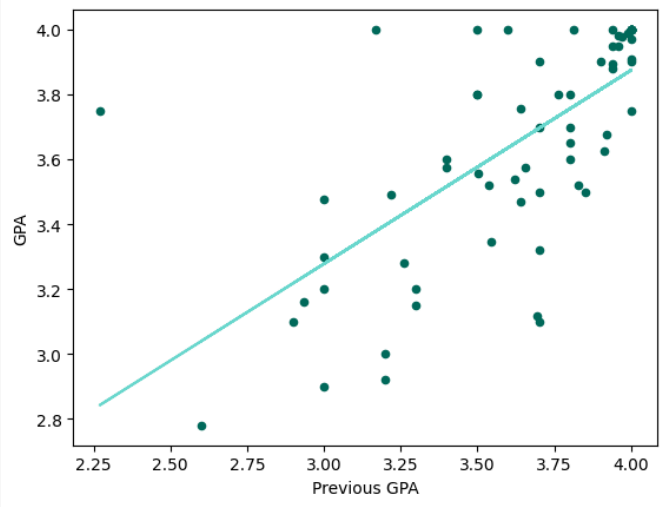






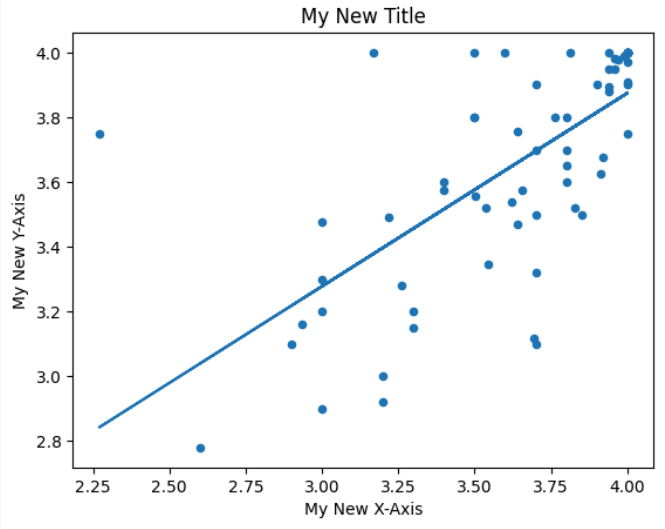
8. Change the colors of the scatter plot or line of best-fit by adding the parameter `c = [Hex Code]` to the plot functions.

```
df_vector.plot.scatter(x='Previous GPA', y='GPA', c='#00695c')
plt.plot(x, y_pred, color='#6ad7cd')
```



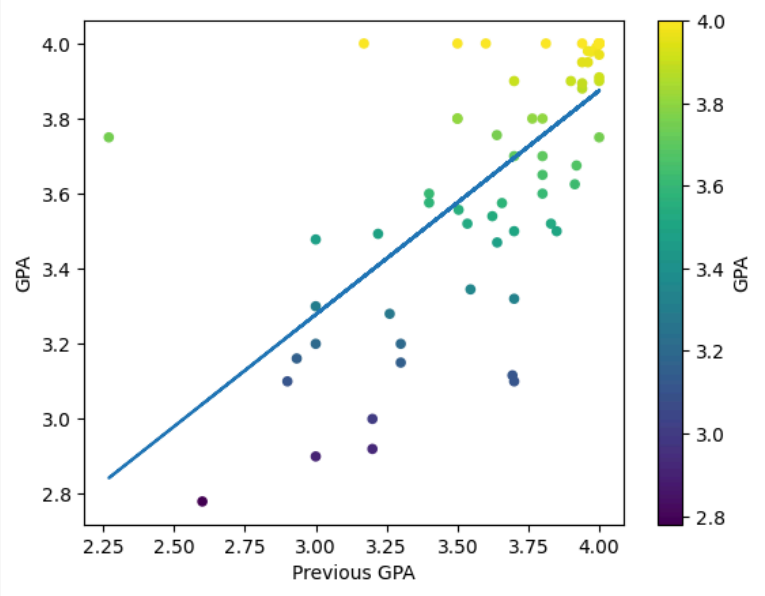
9. Add a title and rename the axes of your graph using the '`title`', '`xlabel`', and '`ylabel`' parameters.

```
df_vector.plot.scatter(x='Previous GPA', y='GPA', title='My New Title',
xlabel='My New X-Axis', ylabel='My New Y-Axis')
plt.plot(x, y_pred)
```



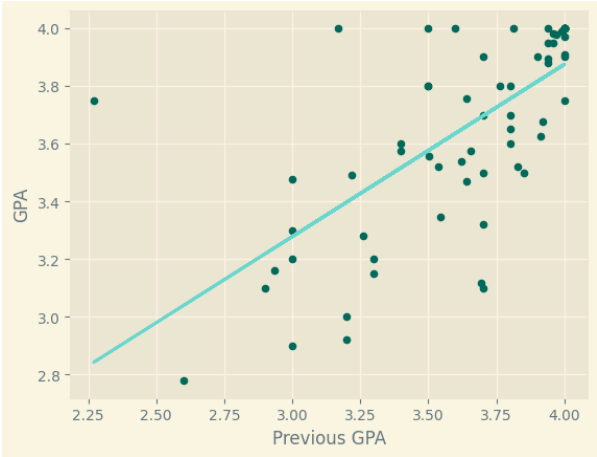
10. Create a gradient of color in your scatter plot by setting the parameter ‘`c`’ to be a feature or label in the dataframe, for which the values will map to an additional parameter `'colormap'`, which will be a set of colors from this list.

```
df_vector.plot.scatter(x='Previous GPA', y='GPA', c='GPA', colormap='viridis')
plt.plot(x, y_pred)
```



11. Choose a background color and design for your graph with `plt.style.use()` where the parameter is a style from [this list](#).

```
plt.style.use('Solarize_Light2')
df_vector.plot.scatter(x='Previous GPA', y='GPA', c='#00695c')
plt.plot(x, y_pred, color='#6ad7cd')
```




```
plt.style.use('dark_background')
df_vector.plot.scatter(x='Previous GPA', y='GPA', c='GPA', colormap='viridis')
plt.plot(x, y_pred)
```