Lab 3 (Due @ by 11:59 pm via Canvas/Gradescope)

Your Name: Diego Cicotoste

Due: Wednesday or Thursday, Oct. 23/24 @ 11:59 pm

Submission Instructions

Submit this `ipynb` file to Gradescope (this can also be done via the assignment on Canvas). To ensure that your submitted `ipynb` file represents your latest code, make sure to give a fresh `Kernel > Restart & Run All` just before uploading the `ipynb` file to gradescope. **In addition:**

- Make sure your name is entered above
- Make sure you comment your code effectively
- If problems are difficult for the TAs/Profs to grade, you will lose points

Special Note for This Lab

There are "handwritten" parts of this lab. You must show all math work/steps (no matter how trivial) to receive full credit. You may either use LaTeX typesetting within a Markdown cell, or do it by hand with pen and paper and embed the image in this .ipynb file, or submit a separate pdf file with your handwritten work. Round all decimals to three places.

Tips for success

- Collaborate: bounce ideas off of each other, if you are having trouble you can ask your classmates or Dr. Gerber for help with specific issues, however...
- Under no circumstances may one student view or share their ungraded homework or quiz with another student (see also), i.e. you are welcome to **talk about** (*not* show each other your answers to) the problems.

In [3]:
```
# you might use the below modules on this lab
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
import pandas as pd
```

```
import numpy as np
import plotly.express as px
from sympy import *
```

Part 1: Eigenvalues & Eigenvectors

Part 1.1: Computation by hand (30 Points)

Show all math work/steps (no matter how trivial) to receive full credit. You may either use LaTeX typesetting within a Markdown cell, or do it by hand with pen and paper and embed the image in this .ipynb file, or submit a separate pdf file with your handwritten work. Round all decimals to three places.

Find the eigenvalues and eigenvectors for the following matrices by hand:

a)

$$A = \begin{bmatrix} 2 & -2 \\ -1 & 3 \end{bmatrix}$$

b)

$$B = \begin{bmatrix} 7 & 3 \\ 3 & -1 \end{bmatrix}$$

Lab3

Matrix $A = \begin{bmatrix} 2 & -2 \\ -1 & 3 \end{bmatrix}$

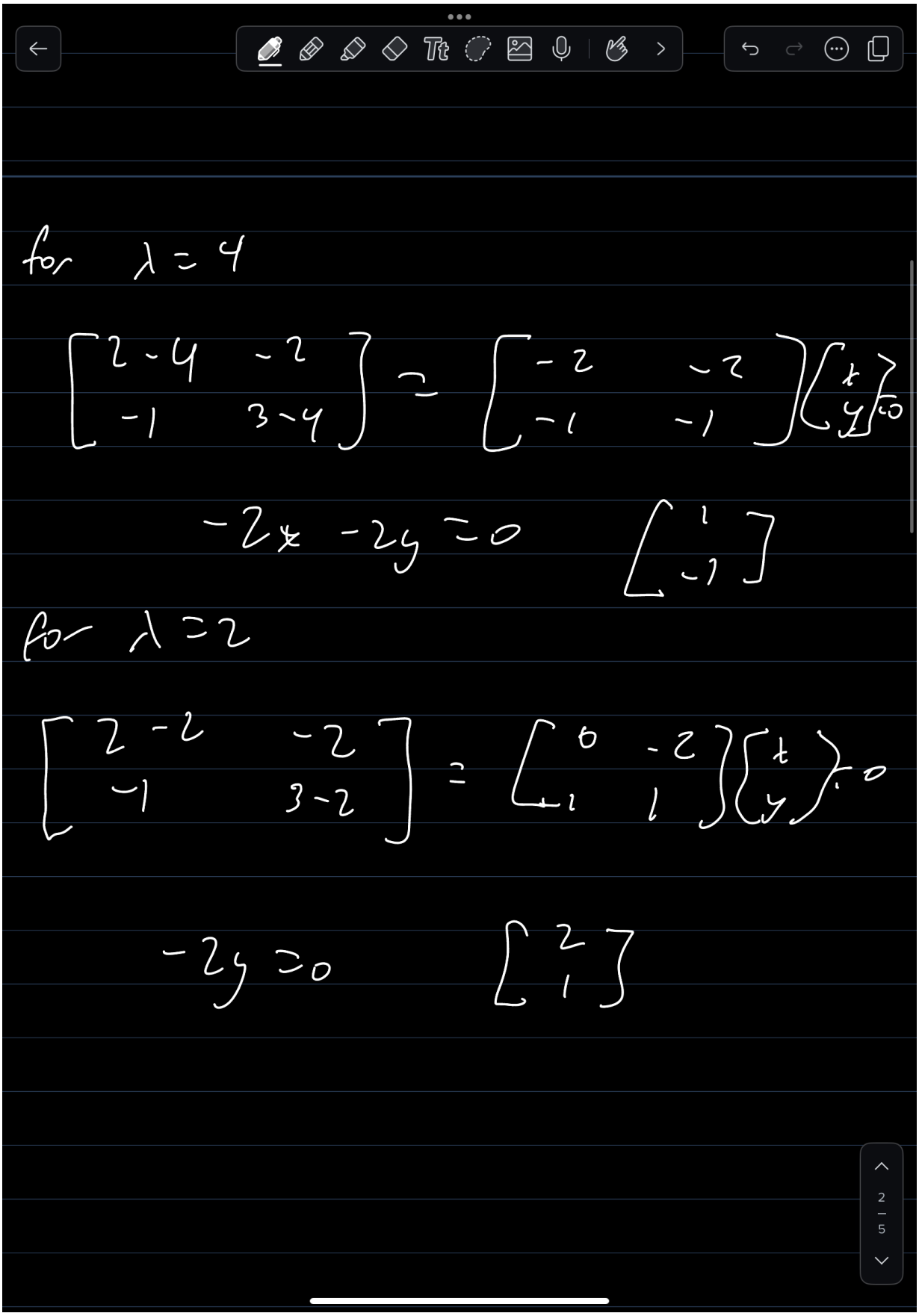
$$\det(A - \lambda I) = 0$$

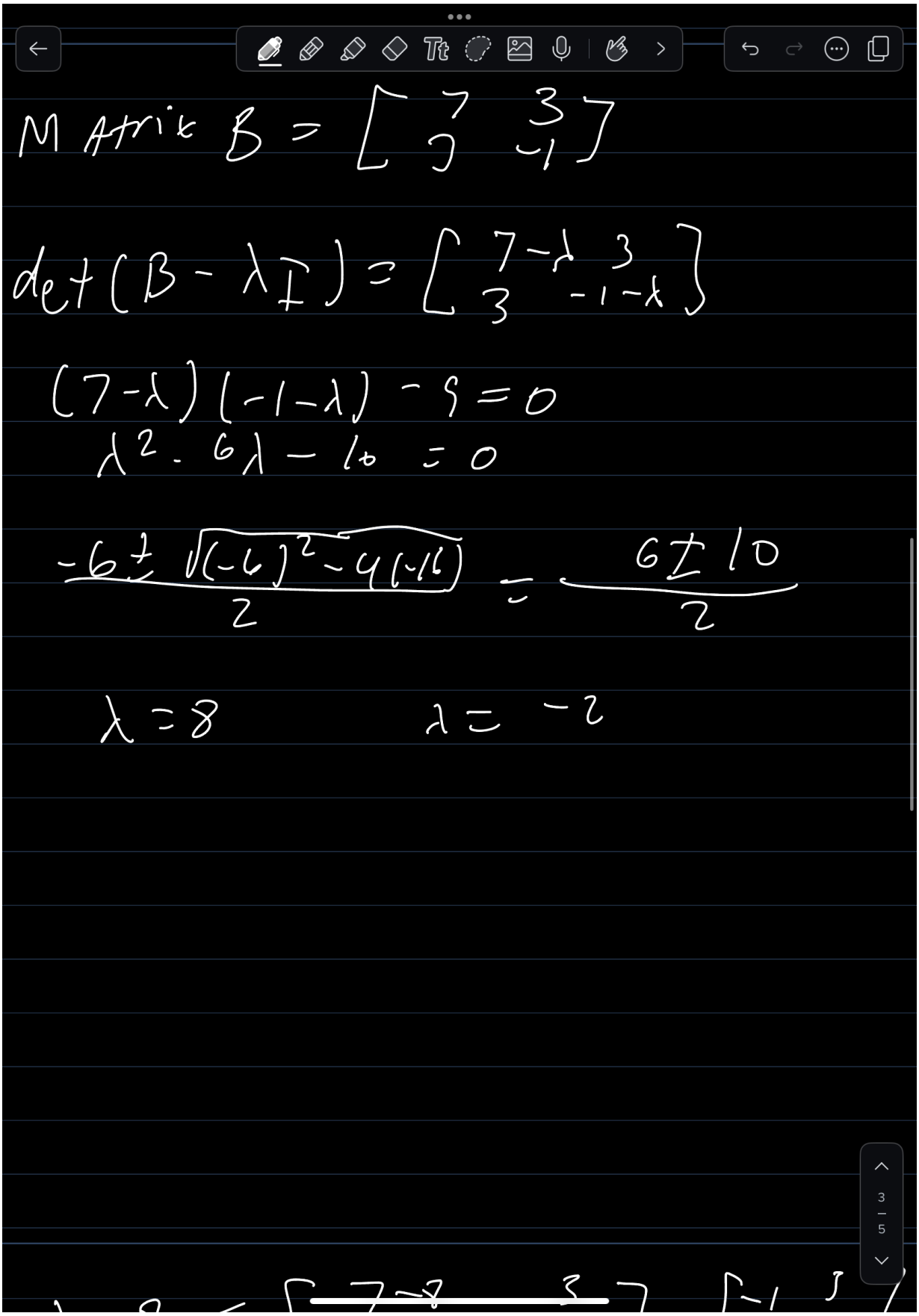
$$\begin{bmatrix} 2 & -2 \\ -1 & 3 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2-\lambda & -2 \\ -1 & 3-\lambda \end{bmatrix}$$

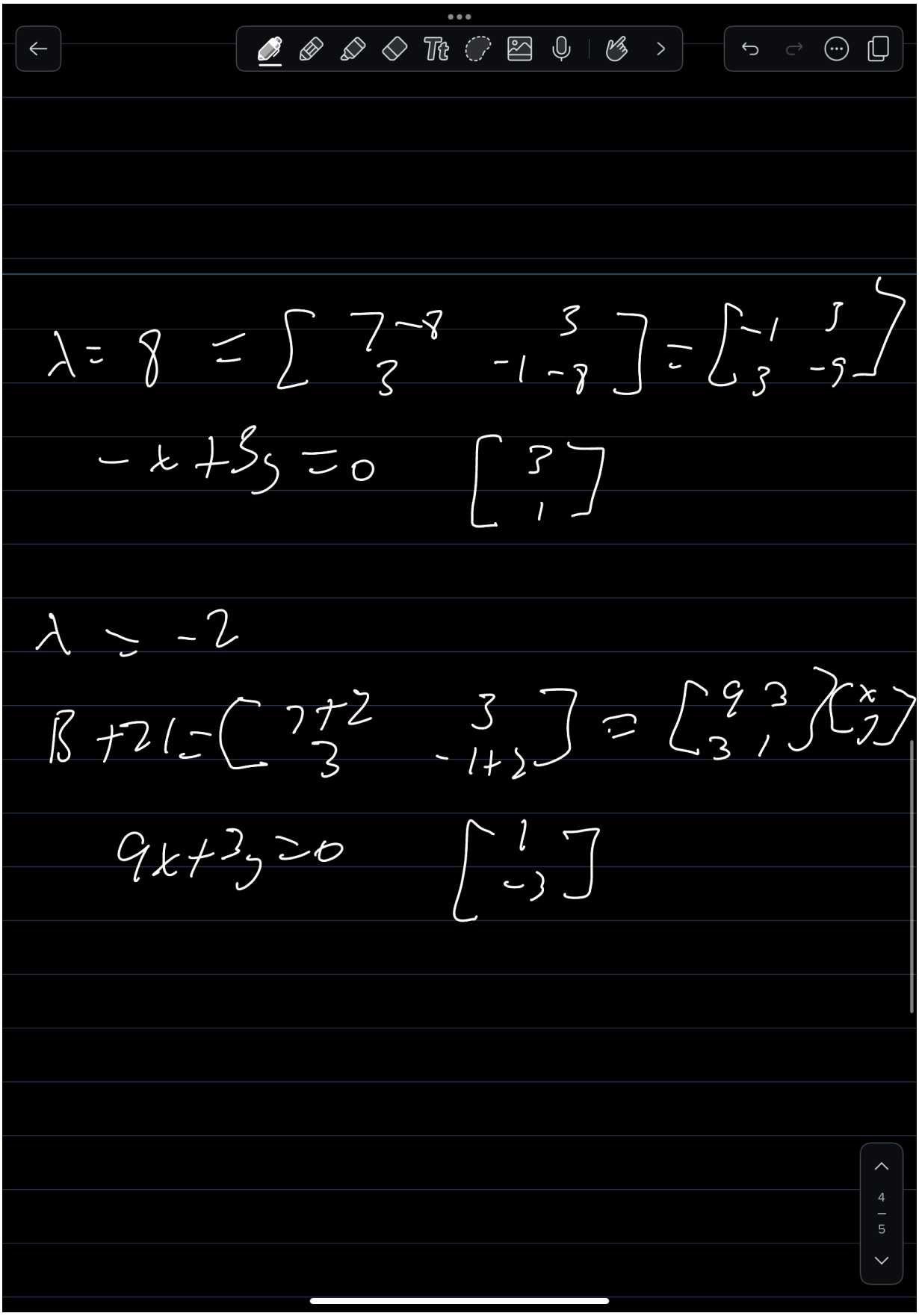
$$(2-\lambda)(3-\lambda) = 6 - \lambda + \lambda^2 = 0$$

$$\frac{-(-5) \pm \sqrt{(-5)^2 - 4(1)(4)}}{2} = \frac{5 \pm 3}{2}$$

$\lambda_1 = 4$ $\lambda = 2$


for $\lambda = 4$
$\begin{bmatrix} 2-4 & -2 \\ -1 & 3-4 \end{bmatrix} = \begin{bmatrix} -2 & -2 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} t \\ y \end{bmatrix} = 0$
$-2x - 2y = 0$
$\begin{bmatrix} 1 \\ -1 \end{bmatrix}$
for $\lambda = 2$
$\begin{bmatrix} 2-2 & -2 \\ -1 & 3-2 \end{bmatrix} = \begin{bmatrix} 0 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} t \\ y \end{bmatrix} = 0$
$-2y = 0$
$\begin{bmatrix} 2 \\ 1 \end{bmatrix}$
2 / 5


Matrix $B = \begin{bmatrix} 7 & 3 \\ 3 & -1 \end{bmatrix}$
$\det(B - \lambda I) = \begin{bmatrix} 7-\lambda & 3 \\ 3 & -1-\lambda \end{bmatrix}$
$(7-\lambda)(-1-\lambda) - 9 = 0$
$\lambda^2 - 6\lambda - 16 = 0$
$\frac{-6 \pm \sqrt{(-6)^2 - 4(-16)}}{2} = \frac{6 \pm 10}{2}$
$\lambda = 8$ $\lambda = -2$
3 / 5


λ = 8
−x + 3y = 0
λ = −2
9x + 3y = 0
4
5

Part 1.2: Computation using `numpy` (20 Points)

Now for each of the above matrices, using `numpy` find their eigenvalues and eigenvectors:

In [4]:
```
import numpy as np

A = np.array([[2, -2], [-1, 3]])
B = np.array([[7, 3], [3, -1]])
eigenvalues_A, eigenvectors_A = np.linalg.eig(A)
eigenvalues_B, eigenvectors_B = np.linalg.eig(B)

print("Eigenvalues of A:", eigenvalues_A)
print("Eigenvectors of A:\n", eigenvectors_A)
print("Eigenvalues of B:", eigenvalues_B)
print("Eigenvectors of B:\n", eigenvectors_B)
```

```
Eigenvalues of A: [1. 4.]
Eigenvectors of A:
 [[-0.89442719  0.70710678]
 [-0.4472136  -0.70710678]]
Eigenvalues of B: [ 8. -2.]
Eigenvectors of B:
 [[ 0.9486833  -0.31622777]
 [ 0.31622777  0.9486833 ]]
```

Part 2: Projections/Line of Best Fit

Part 2.1: 2-d Projection by hand (25 points)

Show all math work/steps (no matter how trivial) to receive full credit. You may either use LaTeX typesetting within a Markdown cell, or do it by hand with pen and paper and embed the image in this .ipynb file, or submit a separate pdf file with your handwritten work. Round all decimals to three places.

Find the projection of vector $\vec{a}$ onto vector $\vec{b}$:

$$\vec{a} = \begin{bmatrix} -4 \\ 1 \end{bmatrix}$$

$$\vec{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Part 2.2: Line of Best Fit using `numpy`

Now, let's extend this to a real data example. On Canvas there is the `big_recipe.csv` in the Labs Module, which contains a recipe data set that we scraped earlier in the semester. For this last part, we will try to answer the question "How would we mathematically quantify the linear relationship between Cholesterol and Total Fat in these recipes?"

Part 2.2.1: Exploring Data (5 points)

The data are read in for you, and a plot is made below. In a markdown cell, explain why you would or would not think that fitting a straight line to model this relationship makes sense.

In [9]:
```
recipe_df = pd.read_csv('big_recipe.csv')

px.scatter(data_frame=recipe_df, x='Cholesterol', y='Total Fat', color='quer
```

Adding a straight line to show the relationship of both cholesterol and total fat could make sense because they are often correlated in high-fat recipes. However, some other variables for example plant-based or animal-based fats may cause outliers. Therefore, while a linear model could provide a good approximation, it may oversimplify the complex relationship between these two variables.

Part 2.2.2: Preparing Data (5 points)

In order to fit the line of best fit through these points using NumPy, we need to store the data as vectors/matrices (arrays) in NumPy. The code below does that. Explain in a markdown cell why we are storing Cholesterol in a matrix called X with an additional column of 1s.

In [6]:
```
y = recipe_df['Total Fat'].to_numpy() # creates the vector y
X = np.vstack([recipe_df.Cholesterol.to_numpy(), np.ones(len(recipe_df.Chole
X[:6,:] # prints the first six rows of the matrix X
```

Out[6]:
```
array([[142.,   1.],
       [218.,   1.],
       [ 99.,   1.],
       [ 44.,   1.],
       [100.,   1.],
       [112.,   1.]])
```

The Cholesterol in a matrix X is being stored in the first column to facilitate the linear regression calculation. in the second column represents the intercept in the linear model. This way it allows for us to solve for both the slope and the intercept of the line of best fit. Where we could use *y=mx+b* with mx being the slope and b being the intercept. Without the additional column we would only calculate the slope and not the intercept, therefore, being an incomplete model.

Part 2.2.3: Fitting the Line of best fit (15 points)

Use NumPy and linear algebra to find the slope and intercept of the line of best fit. Print out the final vector. Then, in a markdown cell, address **in detail** the following:

1. What is the interpretation of the slope and intercept in the context of this problem?
2. Explain in a few sentences why this operation is the same thing as "projecting y onto the span of X".

In [7]:
```
import numpy as np

y = recipe_df['Total Fat'].to_numpy()
X = np.vstack([recipe_df.Cholesterol.to_numpy(), np.ones(len(recipe_df.Chole
beta = np.linalg.inv(X.T @ X) @ X.T @ y

beta
```

Out[7]:
```
array([0.18437883, 9.21885897])
```

The slope shows how much total fat increseases for every cholesterol it increases. Solving for the slope and intercept using linear algebra is equivalent to projecting the vector of Total Fat values onto the space spanned by Cholesterol and a constant intercept.