Google Colab and our Data

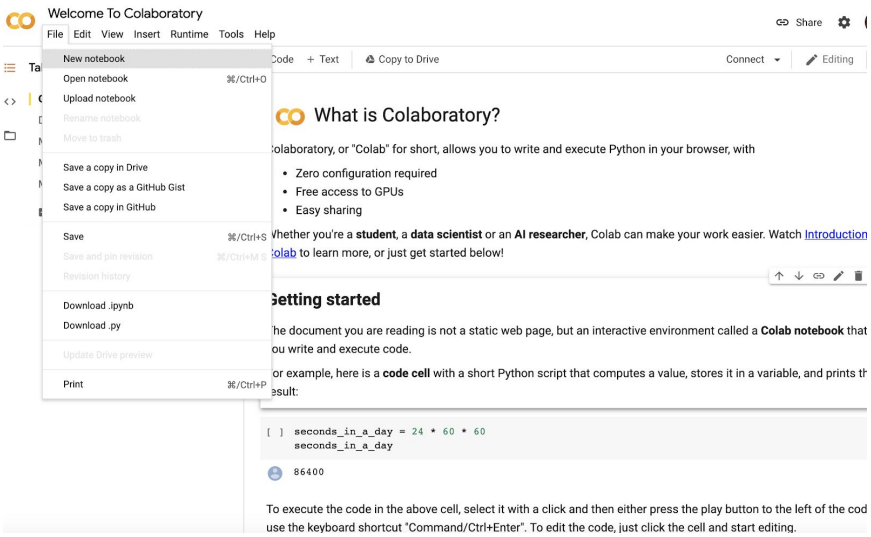
Given our recent dive into database queries and learning what tools we have available to us to help manage large datasets, we've learned a little SQLite, SQL commands, and how to use the DB Browser client to help manage our database projects. Aside from the DB Browser, we did most of our work from the command-line in our CS50 IDEs. But how do we share our work, aside from pushing code to GitHub (we've done this when we submitted CS50 problem sets), or sharing our CS50 IDE workspaces?

This is where Google Colab and the Python programming language can be useful, especially when working on any sort of data science or machine learning project, or when we want to visualize our data with charts and graphs. The tutorial I created will help us scratch the surface, but if you have further curiosity about using Google Colab in your own machine learning projects, I recommend you check out their Machine Learning Crash Course.

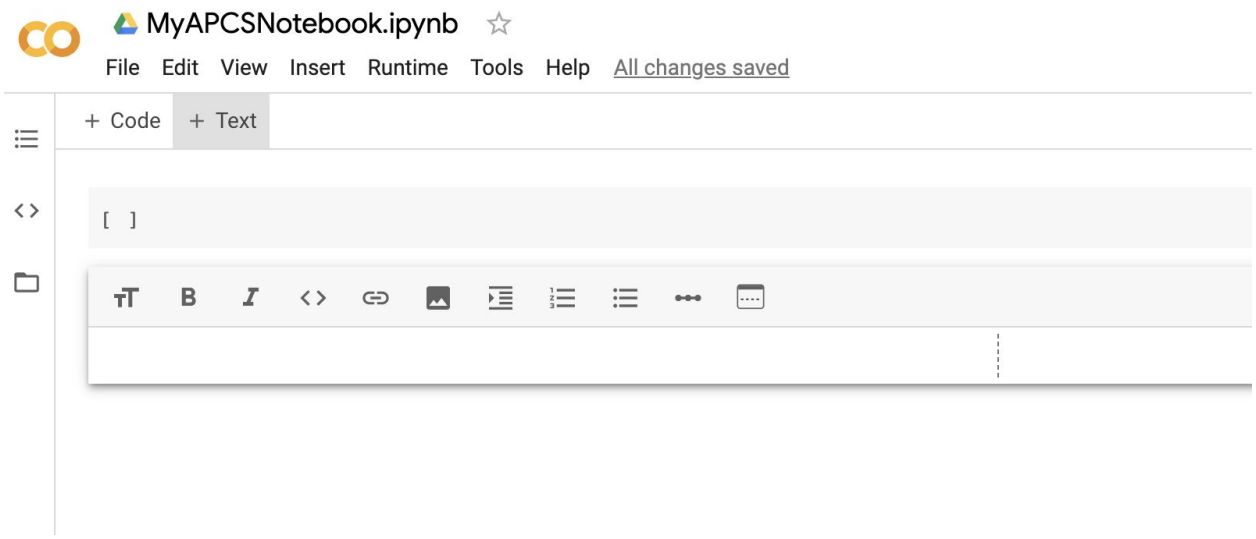
Steps To Take:

1. First watch this video to get an understanding of what Google Colab is. Essentially, it's an interactive notebook that lets you write code and documentation, and include data visualizations, that you can easily share via Google Drive or GitHub. Google Colab uses something called Markdown to style your text cells and documentation.

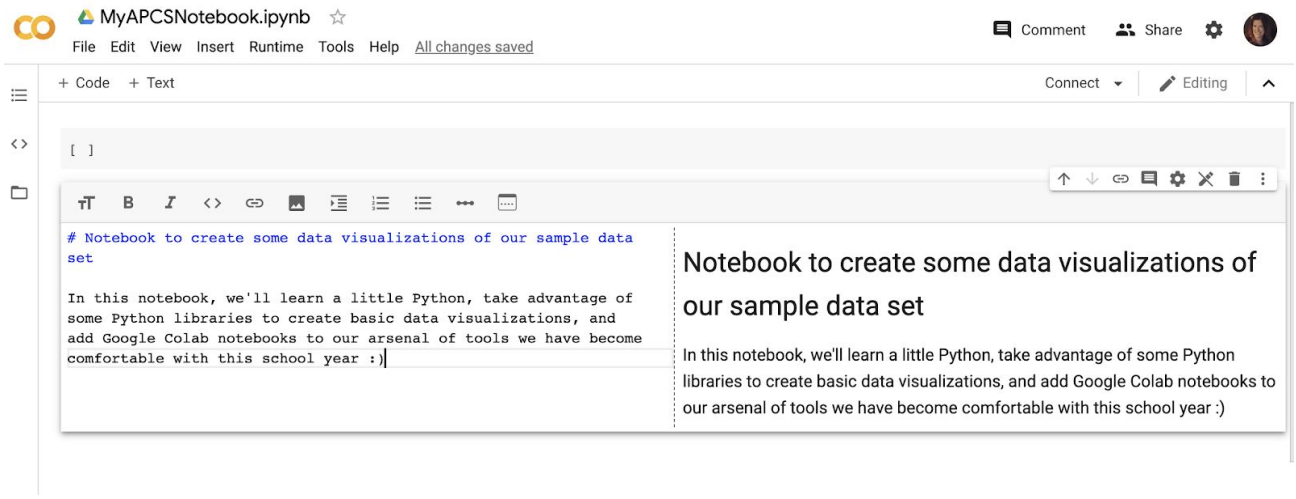
2. Next, go to Google Colab and create a new notebook. We're going to learn a little Python in this notebook that will help us visualize some of that data contained in the sample data set we used for our SQL tutorials. If you don't still have that `.csv` file downloaded to your computer, you'll need to download it again. Rename the file to APCS.csv.



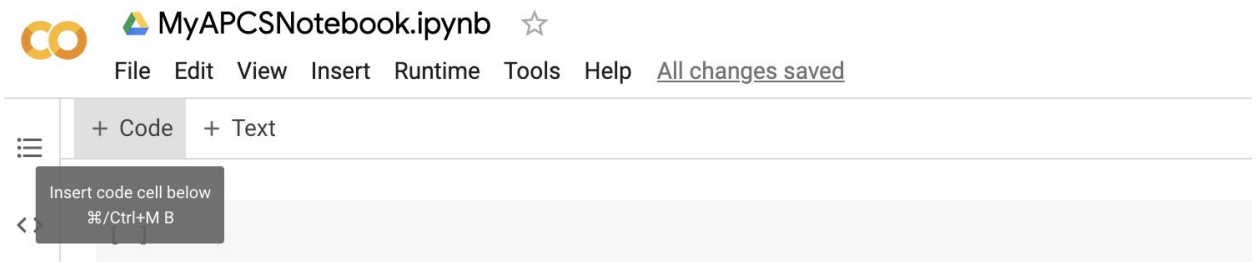
3. Let's start with the basics and just add a text cell to explain what we plan on doing inside this Colab notebook. If you rename your notebook like I did, just make sure you keep the `.ipynb` extension. Then just click on the '+' sign next to 'Text' to add that text cell.



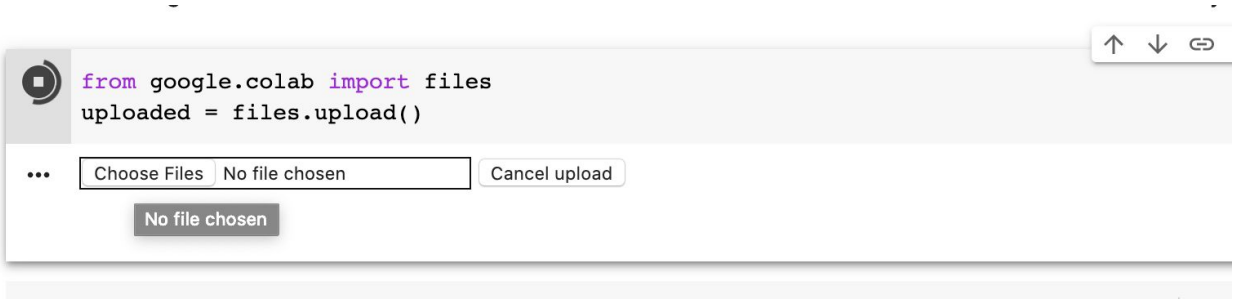
Then let's just add our Markdown to explain what we plan to do. Notice how I used the '#' to indicate which text was my heading, and then just regular text to indicate my paragraph text. The [Markdown guide] is really helpful for learning that syntax.



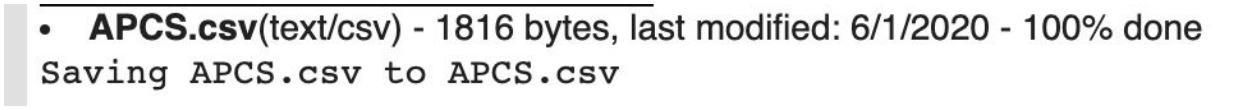
4. Time to add a code cell. You can add one the same way that you added a text cell.



In the code cell, we first want to import any files we plan on using, like the `.csv` file we have downloaded. Type what you see in the screenshot below, and choose your `.csv` file when prompted.



If your file uploads correctly, you should see this:



5. Now we have to add another code cell to add our Python libraries that we plan on using. Two very useful libraries are pandas (a data analysis library) and matplotlib (a visualization library). We'll also use the io library which will let us import our .csv file (and any others we want to work with).

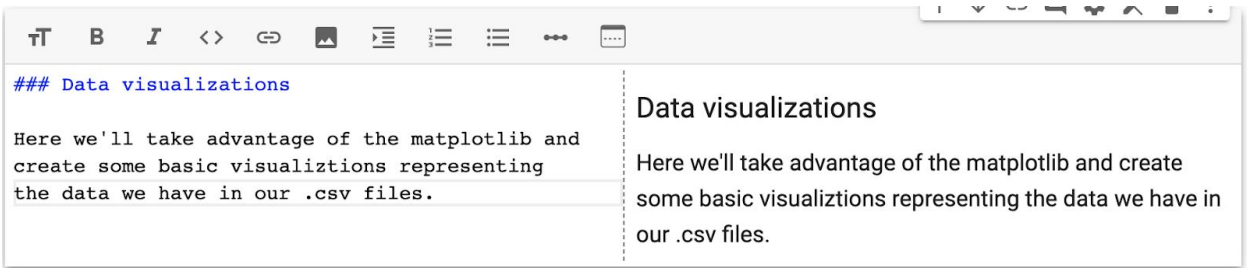
```
[46] #importing libraries we'll be using
     import pandas as pd
     import matplotlib.pyplot as plot
     import io

     #creating variable (df is short for data frame) to hold the contents
     #of our .csv file

     df = pd.read_csv(io.StringIO(uploaded['APCS.csv'].decode('utf-8')), sep = ',')

     #To check and make sure my .csv file uploaded, I can use the head()
     #function to see the first few lines of my file
     df.head()
```

After you type what you see in the window above, click the arrow icon in the top left to run that cell, and you should now see the following:

	name	age	height	hair color	eye color
0	Mark	44	70	brown	hazel
1	John	45	70	brown	hazel
2	Sara	46	70	brown	hazel
3	Megan	57	67	brown	hazel
4	Jeanine	48	67	brown	hazel

This means our data loaded!

6. Ok, let's try creating some basic data visualizations. Before we do that, let's add another text field to indicate that our data visualizations will have their own section in our notebook.



7. Next, I am going to add another code cell so I can double check that the columns in my .csv file imported correctly. Typing and running:

```
df.columns
```

should produce:

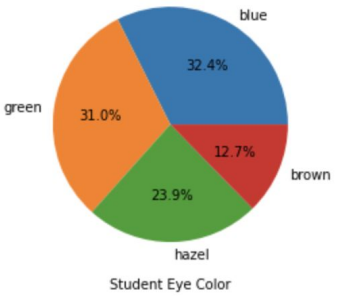
```
Index(['name', 'age', 'height', 'hair color', 'eye color'], dtype='object')
```

8. Now, I am actually going to create my first data visualization! Say I want to get counts of different eye colors, and I wanted to represent those counts in a pie chart. I would need to add another code cell, and type the following:

```
#Getting counts of different eye colors and representing them with
#a pie chart
df['eye color'].value_counts().plot(kind='pie',autopct='%1.1f%%')

#Giving my pie chart a label on its x axis
plot.xlabel('Student Eye Color')

#using a space to remove the default label on the y axis, which is would have
#been the column name 'eye color'
plot.ylabel('')
```

When I run this cell, I should see:

```
Text(0, 0.5, '')
```



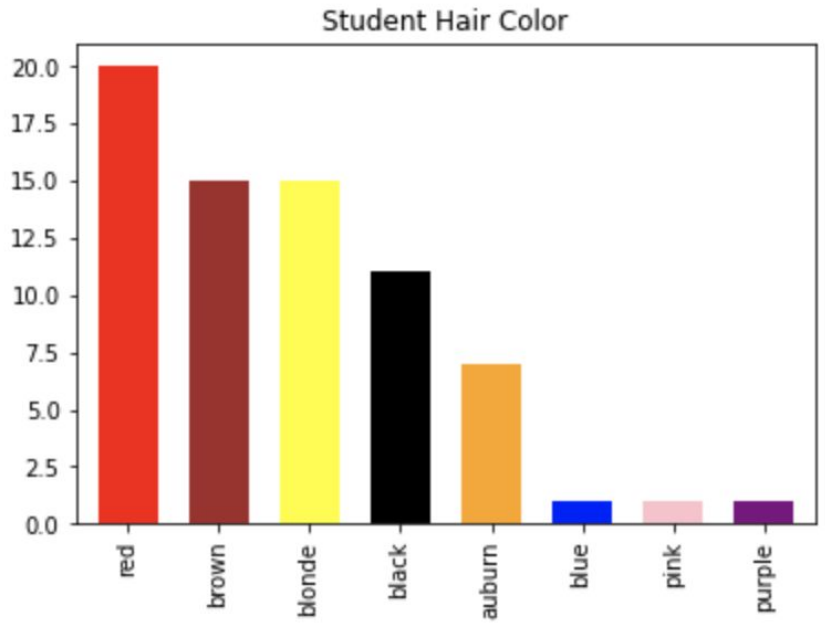
There are also ways to change the colors and labels of your pie pieces, as well as make certain wedges of your pie chart stand out. For more information on how to do this, you might want to read this article.

9. Let's try representing `hair color` in a bar graph. To do that, we need to add another code cell and type the following:

```
#Getting counts of hair color and representing each category in a
#bar chart, while also setting each bar equal to the color the
#category represents
df['hair color'].value_counts().plot(kind='bar', width=0.65, color=
['red', 'brown', 'yellow', 'black', 'orange', 'blue', 'pink', 'purple'])

#Giving my pie chart a title
plot.title('Student Hair Color')

#using a space to remove the default label on the y axis, which would have
#been the column name 'hair color'
plot.ylabel('')
```

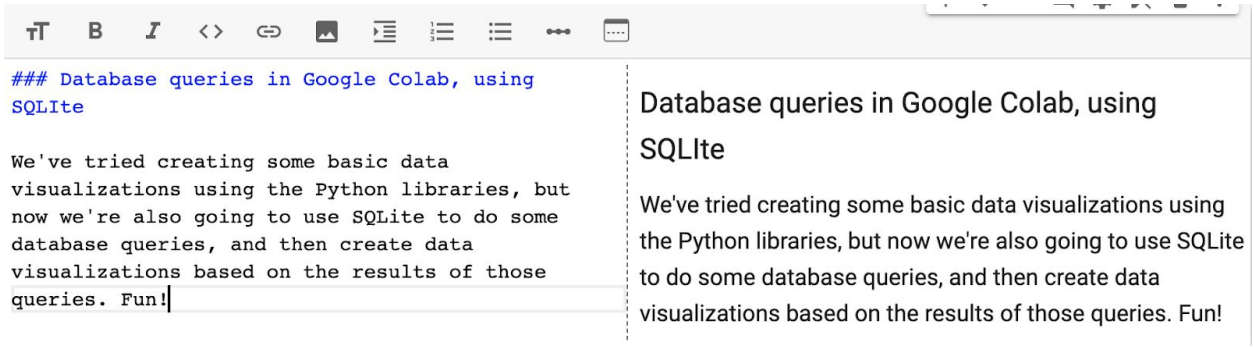
And when we run that cell, we should see a nice bar graph that looks like:

```
Text(0, 0.5, '')
```

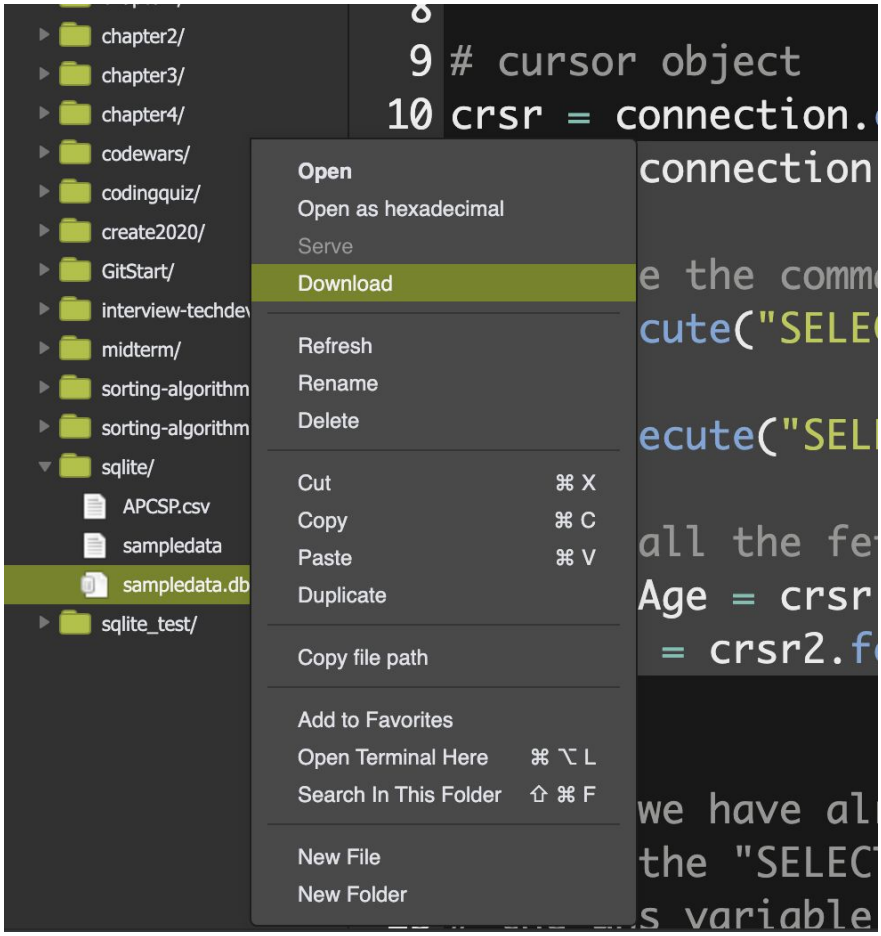


What else can we do in Google Colab?

1. If you guessed database queries, you're right! Let's add another text window to indicate our next plan.



2. First we need to import our data, as it is housed in the `sampledata.db` database we created in our CS50 IDE during part 1 of our SQLite tutorial. To do that, we must first download that `sampledata.db` file from our CS50 IDE.



Once we've done that, we need to use the same `files_upload()` function we used earlier in this Google Colab tutorial. Add another code window and type the following:

```
[78] from google.colab import files
     database = files.upload()
```

Run that cell to ensure that your sampledata.db file uploaded successfully.

- **sampledata.db**(n/a) - 12288 bytes, last modified: 6/1/2020 - 100% done

```
Saving sampledata.db to sampledata.db
```

3. Now let's see if we can run a database query. Just like we imported the `pandas` and `matplotlib` libraries earlier, if we want to write some database queries, we also have to import the `sqlite3` library.

Say I wanted to get a frequency count of the number of students in the `sampledata` database who are 35. I would need to create a new cell window and type the following (pay close attention to the comments to help you understand what is going on...Python syntax is a little different than C):

```
[82] #importing sqlite3 library
     import sqlite3

     #creating a variable called connection and setting it equal to the contents
     #of my sampledata.db table
     connection = sqlite3.connect("sampledata.db")

     #creating a new variable called crsr that creates a cursor object which lets
     #you traverse through the dataset
     crsr = connection.cursor()

     #execute the command to fetch all the data from the table sampledata
     #where student age is 35
     crsr.execute("SELECT COUNT(age) FROM sampledata WHERE age = 35")

     #storing the result of the fetchall function in a variable called countAge
     countAge = crsr.fetchall()
     #prints countAge (the number of students in the database who are 35)
     print(countAge)
```

Add text cell

When I run that cell, I should see the following:

```
[(5,)]
```

Meaning there are 5 people in our database who are 35!

4. Now let's say I want to get counts for each distinct height measurement in the database. I would need to add another cell window, and type the following:

```
[ ]  crsr2 = connection.cursor()

     #Selecting distinct height categories and sorting the heights in descening order
     crsr2.execute("SELECT DISTINCT height FROM sampledata ORDER BY height DESC")

     distinctHeight = crsr2.fetchall()
     #prints countAge (the number of students in the database who are 35)
     print(distinctHeight)
```

And I should see the following:

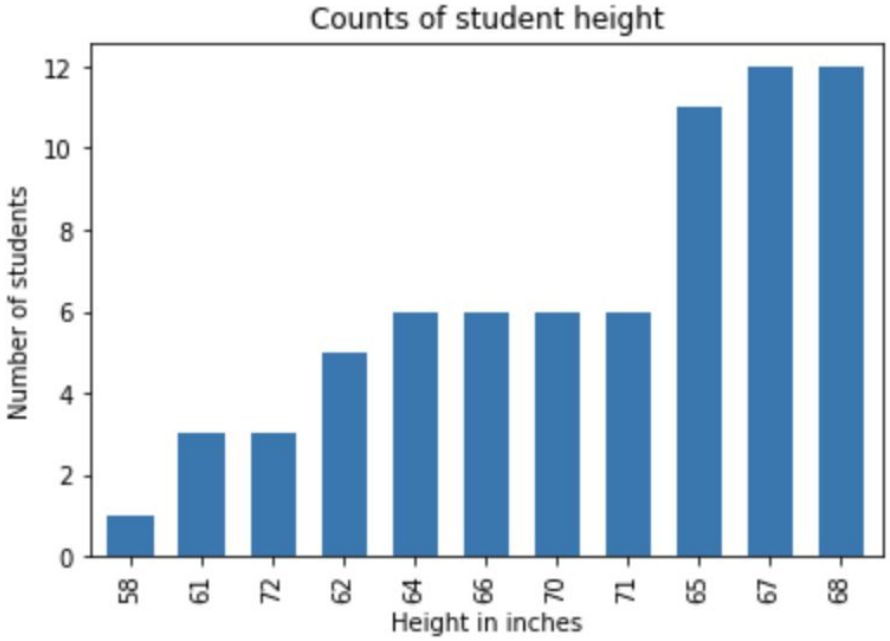
```
[('72',), ('71',), ('70',), ('68',), ('67',), ('66',), ('65',), ('64',), ('62',), ('61',), ('58',)]
```

5. Now let's make a bar graph of those distinct heights, so we can eyeball the most commonly recorded heights in the database of students. We need to add another code cell and type the following:

```
[97] #Getting counts of height and representing each distinct category in a
     #bar chart. I also put the bars in ascending height so we could easily
     #see the largest height group in the dataset
     df['height'].value_counts(ascending=True).plot(kind='bar', width=0.65)

     #Giving my bar chart a title
     plot.title('Counts of student height')

     #I gave some x and y axis labels this time too
     plot.ylabel('Number of students')
     plot.xlabel('Height in inches')
```

And our bar chart should look like:

```
Text(0.5, 0, 'Height in inches')
```

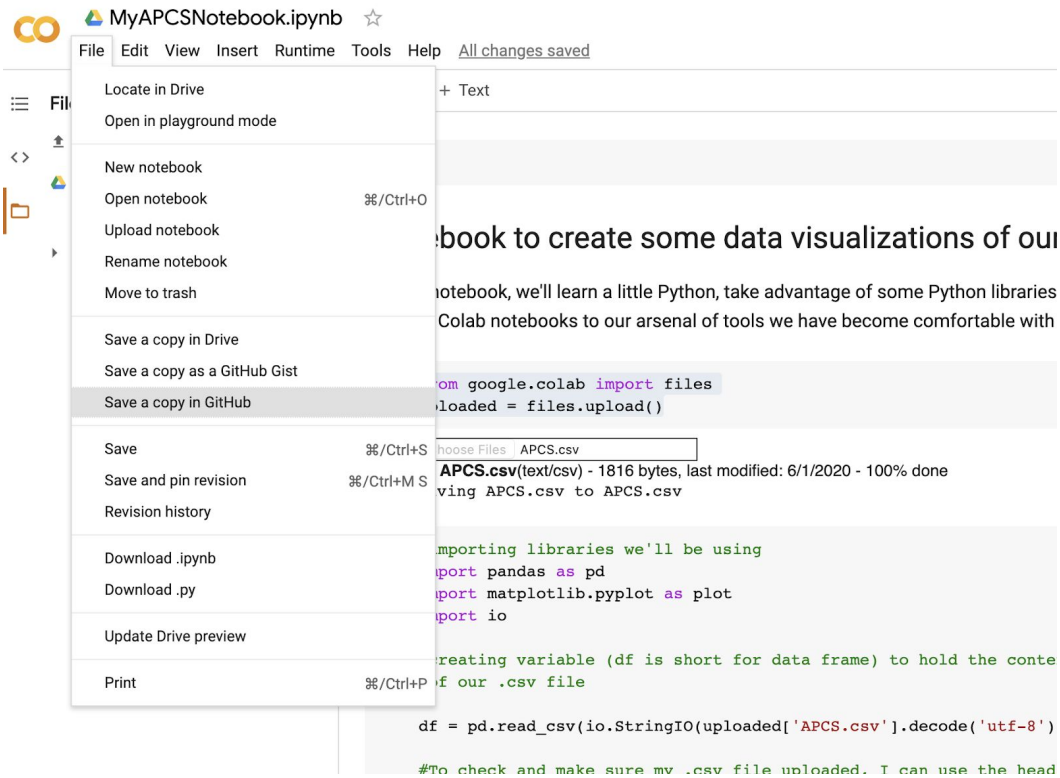


So to summarize, what conclusion can we draw from this bar chart? Looks like the greatest number of students per category are either 5'7" or 5'8".

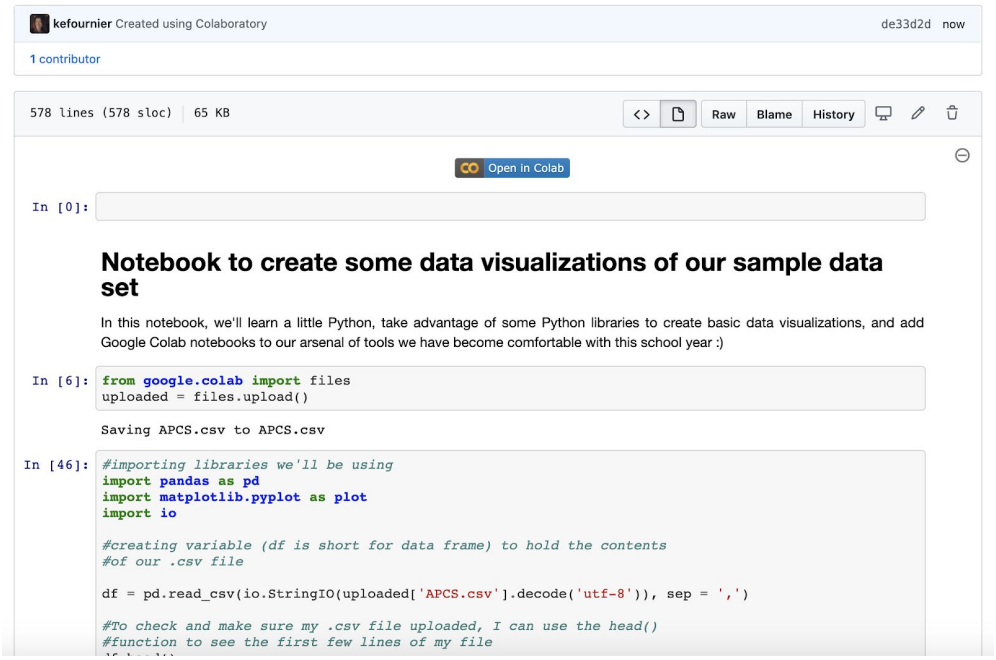
Finals Steps:

If you were able to get yourself through this tutorial start to finish, give yourself a pat on the back for learning a new tool (Google Colab) in a new language (Python)! I saved mine to my GitHub, and I suggest you do the same...it'll make you feel like a real "developer" :) And you'll have it in a spot you can easily share with others should you want to show off your new skills, or revisit for a future project (I can envision some of you creating Science Fair projects using Google Colab, or Jupyter Notebooks...).

Here's how you save it to GitHub (follow the prompts):



And here's what it will look like in your GitHub repository once you've saved it:



By default, you may already have a repo in GitHub that you can save to, but if not, here are the directions to create one. And then once you have a repository, you can create a branch which is

where your Google Colab notebook will save to. If you don’t save your notebook to GitHub, it will still be saved in Google Drive, and you can easily save it that way.

Nice work!