Topic	Data-Science 1	
Class Description	**Students will be applying statistics to the exo-planets data and plotting charts to find interesting insights.**	
Class	**C131**	
Class time	**45 mins**	
Goal	• Plot various charts from our data • Find interesting insights about the exo-planets	
Resources Required	• Teacher Resources ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen • Student Resources ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen	
Class structure	**Warm Up** **Teacher-led Activity** **Student-led Activity** **Wrap up**	**5 mins** **15 min** **15 min** **5 min**

CONTEXT

- **Review the concepts learned in the earlier classes**

Class Steps	Teacher Action	Student Action
Step 1: Warm Up (5 mins)	Hi <Student Name>! In the last class, we understood and cleaned our data. We first understood the meaning of the fields, determined which ones do we need and removed the rest. We then re-named all the	

	columns to make our data more readable for us.	
	In today's class, we will jump to the exciting stuff! We will be plotting different graphs and apply statistics to our data to see if we can find any interesting insights. Are you excited about it?	**ESR:** "Yes!"
	Let's get right into it.	
Teacher Initiates Screen Share		
CHALLENGE • **Discussing with the student on what all can we plot** • **Plotting the charts and applying statistics** • **Finding interesting insights**		
Step 2: Teacher-led Activity (15 min)	*(Before beginning the class, please make the student download the CSV from the link below. This CSV is the latest version of the data on which we have to perform data cleaning.)* *<Teacher can download from **Teacher Activity 1**>* https://raw.githubusercontent.com/whitehatjr/Data-cleaning/master/main.csv	*<Student can download from **Student Activity 1**>*
	Let's just start by recalling the meaning of all the columns we have left in our CSV. *<Discuss all the columns one by one with the student>*	*The student discusses the columns with the teacher.*

	name - name of the planet. ***light_years_from_earth*** - Distance of the exo-planet from earth in light years. ***planet_mass*** - Mass of the planet. ***stellar_magnitude -*** This is the brightness of the host star of the planet when observed from Earth (just as the sun is our host star). ***discovery_date -*** This is the year of discovery for the exo-planet. ***planet_type -*** This is the type of the planet (Gas Giant, Super Earth, etc.). ***planet_radius -*** This is the radius of the exo-planet with respect to Earth or Jupiter. ***orbital_radius -*** This is the average distance of this exo-planet from its sun. Just like our solar system has 1 sun, there are multiple solar systems that contain many planets and sun(s). ***orbital_period -*** This is the time it takes to complete one orbit of it's sun. ***eccentricity -*** This denotes how circular the orbit is. It might be oval in shape too. The lower the eccentricity, the more circular is the orbit. ***solar_system_name -*** The name of the host solar system. ***planet_discovery_method -*** This is the discovery method which was used to find this exo-planet. ***planet_orbital_inclination -*** This is the orbital inclination, which means that it is the tilt of the exo-planet's orbit when it revolves around its sun. ***planet_density -*** This is the density of the planet. ***right_ascension -*** This is the right ascension of the planetary system, which is the east-west coordinate by which the position of this planet is	

	measured. ***declination -*** This is the north-south coordinate by which the position of the planet is measured. ***host_temperature -*** This is the temperature of the host star in Kelvin. ***host_mass -*** This is the amount of mass contained in the host star. ***host_radius -*** This is the radius of the host star.	
	Now, in our Solar System, we have about 8 planets. Can you name them?	**ESR:** - Mercury - Venus - Earth - Mars - Jupiter - Saturn - Neptune - Uranus
	Awesome. Just like we have 8 planets in our solar system, let's try to find out how many planets have we found in other solar systems! For this, we have a column known as ***solar_system_name*** in our CSV. Using this, we can create a dictionary to maintain the count of planets each solar system has! Before we do that, if we look at the csv we just downloaded, we can see that there is index mentioned in the rows as the first element, but the header's first element is empty - `import csv`	**ESR:** Using Game States

```
rows = []

with open("main.csv", "r") as f:
 csvreader = csv.reader(f)
 for row in csvreader:
   rows.append(row)

headers = rows[0]
planet_data_rows = rows[1:]
print(headers)
print(planet_data_rows[0])
```

Here, we are just reading the CSV file and segregating the two into headers and planet_data

Let's fix that and then find the solar system with the most number of planets!

```
headers[0] = "row_num"

solar_system_planet_count = {}
```

```
for planet_data in
planet_data_rows:
 if
solar_system_planet_count.get(pl
anet_data[11]):

solar_system_planet_count[planet
_data[11]] += 1
 else:

solar_system_planet_count[planet
_data[11]] = 1

max_solar_system =
max(solar_system_planet_count,
key=solar_system_planet_count.ge
t)
print("Solar system {} has
maximum planets {} out of all
the solar systems we have
discovered so
far!".format(max_solar_system,
solar_system_planet_count[max_so
lar_system]))
```

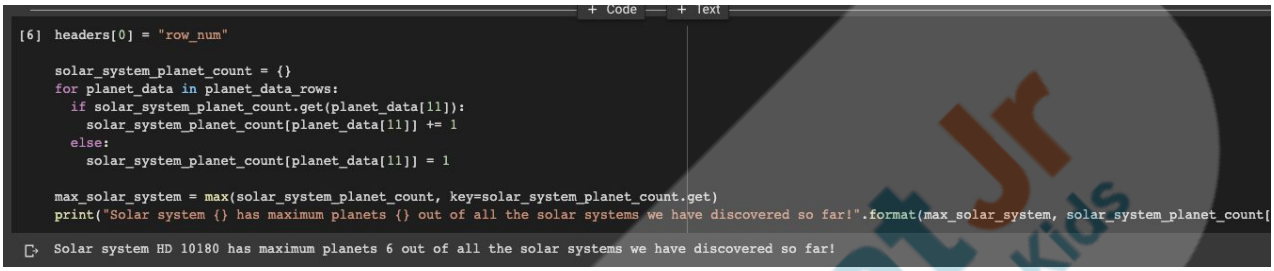
Here, we are first adding the header, to make our data consistent. Then, we are creating an empty dictionary where we can store the number of planets in each solar system.

We are iterating over all the planets and we are checking if we already have an entry for that solar system in our dictionary or not.

If we do, we are increasing the count by 1 else we are adding this solar

system into our dictionary with count as 1.

Finally, we are just finding the name of the solar system (key in our dictionary) with maximum number of planets (values in our dictionary).



Here, we can see that there is a solar system known as **HD 10180** where there are 6 planets! That's awesome! Could this be our next home? Let's get the list of all the planets from this solar system!

```
hd_10180_planets = []
for planet_data in
planet_data_rows:
 if max_solar_system ==
planet_data[11]:

hd_10180_planets.append(planet_d
ata)

print(len(hd_10180_planets))
print(hd_10180_planets)
```

This code would simply check if a given planet's solar system is equal to the solar system with maximum planets and if it is, we are adding the details of that planet into a list.

```
hd_10180_planets = []
for planet_data in planet_data_rows:
  if max_solar_system == planet_data[11]:
    hd_10180_planets.append(planet_data)

print(len(hd_10180_planets))
print(hd_10180_planets)
```

```
6
[['398', 'HD 10180 c', '127.0', '13.22173 Earths', '7.321000000000001', '2010', 'Neptune-like', '0.33 x Jupiter', '0.06412 AU', '5.8 days', '0.07', 'HD
```

Great! Now that we have the list of planets in this solar system, we can plot different charts to study them.

Before we proceed with that, we need to make sure that our data is uniform! We have 2 columns, ***planet_mass & planet_radius*** where in some rows, the values are given with reference to ***Jupiter*** while at some places, it's given with the reference of ***Earth***.

We have to make it uniform in our CSV. Let's make it ***Earth*** for all, since it would be easier for us to compare that way!

For this, we need to look at our 3rd and 7th element and see if they are measured in ***Jupiter or Earth.*** If they are measured in Jupiter, we need to change it. There are also a few rows with value as ***Unknown.*** We need to remove these rows as well!

We will do this in the same cell in which we found out the 6 planets in the same solar system.

Doing this, we will make this data uniform.

```
temp_planet_data_rows =
list(planet_data_rows)
```

```
for planet_data in
temp_planet_data_rows:
 planet_mass = planet_data[3]
 if planet_mass.lower() ==
"unknown":

planet_data_rows.remove(planet_d
ata)
   continue
 else:
   planet_mass_value =
planet_mass.split(" ")[0]
   planet_mass_ref =
planet_mass.split(" ")[1]
   if planet_mass_ref ==
"Jupiters":
     planet_mass_value =
float(planet_mass_value) * 317.8
   planet_data[3] =
planet_mass_value

 planet_radius = planet_data[7]
 if planet_radius.lower() ==
"unknown":

planet_data_rows.remove(planet_d
ata)
   continue
 else:
   planet_radius_value =
planet_radius.split(" ")[0]
   planet_radius_ref =
planet_radius.split(" ")[2]
   if planet_radius_ref ==
"Jupiter":
```

	`` planet_radius_value = float(planet_radius_value) * 11.2 planet_data[7] = planet_radius_value ``` Great! We will be left with 4,251 planet data! Let's understand this code line by line - At first, we created a temporary planet list (using list function) and iterated over it. Then, we just checked if the third element of the list (***planet_mass***) is ***unknown*** or not. If it's not, we take out the value and reference (***Earth or Jupiter***) for the planet. If it's Jupiter, we replace the values and then we change the value of the element in the list. If in case the value was ***unknown***, we are removing the ***planet_data*** from our main list ***planet_data_rows.*** We did the same for the 7th element (***planet_radius***) as well! 1 Jupiter Mass = 317.8 Earth Mass 1 Jupiter Radius = 11.2 Earth Radius Can you tell why we created a temporary list of all planet rows?	**ESR:** Because we are removing the element from the list inside the for loop. If we iterate over the same list and remove from the same list too, then our code will perform unexpectedly.

```
for planet_data in planet_data_rows:
  planet_mass = planet_data[3]
  if planet_mass.lower() == "unknown":
    planet_data_rows.remove(planet_data)
    continue
  else:
    planet_mass_value = planet_mass.split(" ")[0]
    planet_mass_ref = planet_mass.split(" ")[1]
    if planet_mass_ref == "Jupiters":
      planet_mass_value = float(planet_mass_value) * 317.8
    planet_data[3] = planet_mass_value

  planet_radius = planet_data[7]
  if planet_radius.lower() == "unknown":
    planet_data_rows.remove(planet_data)
    continue
  else:
    planet_radius_value = planet_radius.split(" ")[0]
    planet_radius_ref = planet_radius.split(" ")[2]
    if planet_radius_ref == "Jupiter":
      planet_radius_value = float(planet_radius_value) * 11.2
    planet_data[7] = planet_radius_value

print(len(planet_data_rows))

hd_10180_planets = []
for planet_data in planet_data_rows:
  if max_solar_system == planet_data[11]:
    hd_10180_planets.append(planet_data)

print(len(hd_10180_planets))
print(hd_10180_planets)
```

```
4255
7
[['3665', 'Kepler-903 b', '2704.0', '4.7', '14.615', '2016', 'Super Earth', '2.01', 'Unknown', '10.4 days', '0.0', 'KOI-351', 'Transit', '89.2', '', '18h57m4
```

Now, let's plot a bar chart on the planet mass -

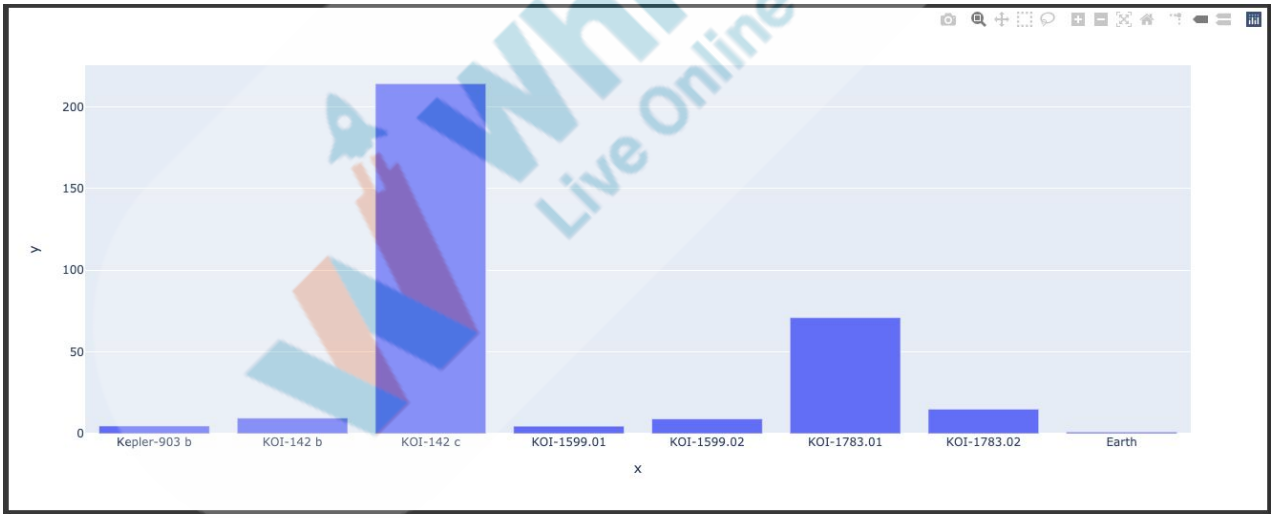
```
import plotly.express as px

hd_10180_planet_masses = []
hd_10180_planet_names = []
for planet_data in
hd_10180_planets:

hd_10180_planet_masses.append(pl
anet_data[3])

hd_10180_planet_names.append(pla
net_data[1])

hd_10180_planet_masses.append(1)
hd_10180_planet_names.append("Ea
rth")
```

```
fig =
px.bar(x=hd_10180_planet_names,
y=hd_10180_planet_masses)
fig.show()
```

We are creating separate lists of planet masses and names, and then we are adding the details of our planet Earth as well, just to compare.

We are finally plotting a bar chart with names on the X-Coordinate and masses on the Y-Coordinate.

We can see that there are 2 planets - ***Kepler-903b and KOI-1599.01*** who are close to earth in terms of mass (***Earth's*** mass would be 1 since we converted these metrics with reference to Earth!)

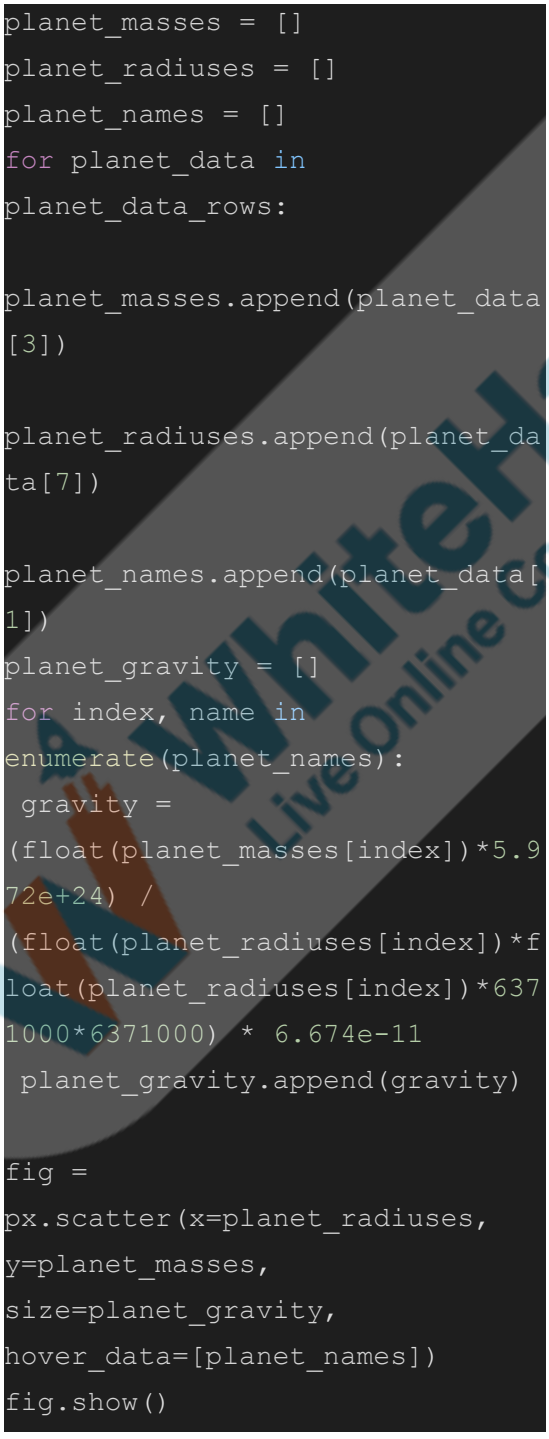


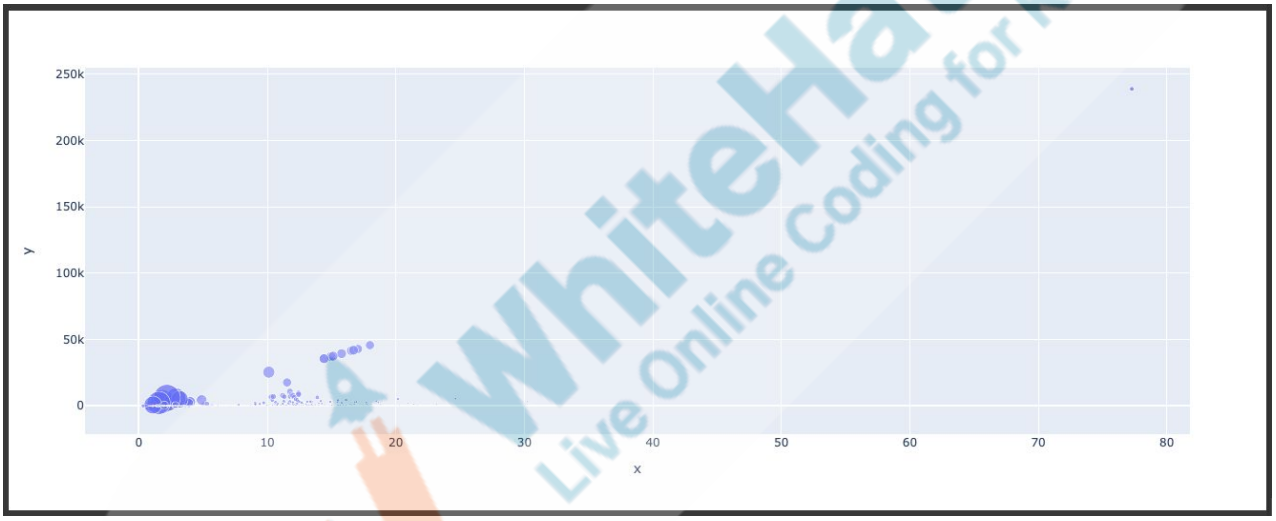
Okay, now let's try to have this with all the planets we have discovered so far and see if we can find more earth like planets.

-

	Before we do that, let's understand one more thing. Great Scientist Albert Einstein gave us a formula with which we can calculate the gravity of any planet. The formula is this - $g = \frac{G * M_{earth}}{d^2}$ Here, G is a gravitational constant, which means that it will always be the same. M(earth) is the mass of Earth (or any other planet if we are calculating it for another planet) d is the radius of the planet! Here, we can see an inverse relation between the radius of the planet and the gravity. The more the radius (and bigger the planet), the lesser would be Gravity. But then, we also see a direct relation between the mass of the planet and the gravity. The more the mass of the planet, the more will be the gravity. Our Earth's gravity is **9.8 m/s,** and we as humans are accustomed to it.	

	In order for us to exist on any other planet, the gravity should be close to what we have here. ***Mars*** has a gravity of ***3.711 m/s*** and ***Moon*** has a gravity of ***1.62 m/s***.	
Teacher Stops Screen Share		
	Now it's your turn. Please share your screen with me.	
• **Ask Student to press ESC key to come back to panel** • **Guide Student to start Screen Share** • **Teacher gets into Fullscreen**		
ACTIVITY • **Student code to build the complete game** • **Add optional challenges for the student**		
Step 3: Student-Led Activity (15 min)	Given what we have just learned, let's try to plot a scatter plot for all the planets, where we will keep the mass of the planet as the Y-Coordinate, the Radius of the Planet as X-Coordinate, and the size of the blob as the gravity of it. Let's see if we can find anything interesting! *<Help the student in writing the code for this>* ***The value of G (Gravitational Constant) is 6.674e-11*** **Note - Since we have the *planet_mass* and *planet_radius* with Reference to Earth, don't forget to multiple the mass of the**	*The student writes code to create the following scatter plot.*

	earth and the radius of the earth with these values - Mass of Earth = 5.972e+24 Radius of Earth = 6371000	
	 ``` planet_masses = [] planet_radiuses = [] planet_names = [] for planet_data in planet_data_rows: planet_masses.append(planet_data [3]) planet_radiuses.append(planet_da ta[7]) planet_names.append(planet_data[1]) planet_gravity = [] for index, name in enumerate(planet_names): gravity = (float(planet_masses[index])*5.9 72e+24) / (float(planet_radiuses[index])*f loat(planet_radiuses[index])*637 1000*6371000) * 6.674e-11 planet_gravity.append(gravity) fig = px.scatter(x=planet_radiuses, y=planet_masses, size=planet_gravity, hover_data=[planet_names]) fig.show() ```	

	Let's understand this code. We are first segregating the names, masses and radiuses of the planets. We are then applying the formula by filling the values we know and creating a list of gravities for all the planets. Finally, we are plotting the graph with X as planet_radiuses, Y as planet_masses, size as planet_gravity and hover_data as the name of the planet!	
		
	As we can see, there is this one humongous planet plotted on the top right of the chart with name **HD 100546 b**. Let's remove this planet from our main planet_list to have a better plotted chart. `temp_planet_data_rows = list(planet_data_rows)` `for planet_data in temp_planet_data_rows:`	*Student finds and removes this planet_row and re-runs the code.*

```
 if planet_data[1].lower() ==
"hd 100546 b":

planet_data_rows.remove(planet_d
ata)
```

This code again creates a temporary list (since we do not want to remove an element from the same list on which we are iterating), iterates over the planet_data and checks if the name of the planet is the one we want to remove (**Remember, removing outliers?**). In our case, it is `hd 100546 b`

Here, now we see a better Graph!

	Fun Fact - Our standing human bodies can withstand a gravitational force 90 times stronger than earth! Although that is going to be a bit extreme, we can still survive at 10 times the gravity we have at Earth. Let's list down all the names of the planets that have Gravity of 100 or less! `low_gravity_planets = []` `for index, gravity in enumerate(planet_gravity):` `  if gravity < 100:` `low_gravity_planets.append(planet_data_rows[index])` `print(len(low_gravity_planets))` (Here, we are iterating over the gravities of the planet with enumerate function (to get the index of the planet as well) and then if the planet's gravity is less than 100, we are adding the planet data into a list.) 3,951 Planets! That's insane! Let's see how many are there with gravity less than 10? `low_gravity_planets = []` `for index, gravity in enumerate(planet_gravity):`	*The student creates a list of planets.*

	`` if gravity < 10: low_gravity_planets.append(plane t_data_rows[index]) print(len(low_gravity_planets)) ``` 1,012 Planets! Wow! We have a lot of potential options!	
	`` [101] low_gravity_planets = [] for index, gravity in enumerate(planet_gravity): if gravity < 100: low_gravity_planets.append(planet_data_rows[index]) print(len(low_gravity_planets)) 3951 ``` **Less than 100 Gravity** ``` [102] low_gravity_planets = [] for index, gravity in enumerate(planet_gravity): if gravity < 10: low_gravity_planets.append(planet_data_rows[index]) print(len(low_gravity_planets)) 1012 ``` **Less than 10 Gravity**	
	Great! That's it!	
Teacher Guides Student to Stop Screen Share		
FEEDBACK • **Appreciate the student for their efforts** • **Identify 2 strengths and 1 area of progress for the student**		
Step 4: Wrap-Up (5 min)	So, in this class, we converted all the planet's mass and radius with reference to earth and understood how gravity is calculated. We also	

	found out a list of 3,951 planets (1,012 planets heavily favourable) which could be our next home! Our human bodies are amazing, but they have limitations too. We will try to understand what all we need to survive as humans in the next class! How was your experience?	**ESR:** varied
	Amazing. While working on this project, we also made sure that we are on top of all the concepts we have acquired so far. Next class, we will dive more deep into our data and filter out some other planets!	-
Teacher Clicks 		

Activity	Activity Name	Links
Student Activity 1	Data from last class	https://raw.githubusercontent.com/whitehatjr/Data-cleaning/master/main.csv
Teacher Activity 1	Data from last class	https://raw.githubusercontent.com/whitehatjr/Data-cleaning/master/main.csv
Teacher Activity 2	Solution	https://colab.research.google.com/drive/1SQUQYIrhJ--bxxLyd_abCKVC-6a2t93n?usp=sharing