CS170 - Introduction to Data Science (Jupyter Notebook)

Instructions

Answer each line item by replacing the blanks with the necessary operator or a value. Make sure the kernel is set to Python 3** Once done, right click the actual notebook page and print as PDF. Last part of the notebook is the code for timestamp from your computer - Run it!.

In [ ]:

```
#Import the necesssary library such as pandas and matplotlib
import pandas as pd
import matplotlib.pyplot as plt
```

In [ ]:

```
#read the dataset
pokemon = pd. read_csv ("C:/Users/Jhainno Marcos/Downloads/pokemon.csv")
```

In [ ]:

```
pokemon .shape
#get the shape of the dataset
```

Out[ ]:

```
(801, 8)
```

In [ ]:

```
pokemon.head(10)
#complete the syntax to disply the first 10 rows of the record.
```

Out[ ]:

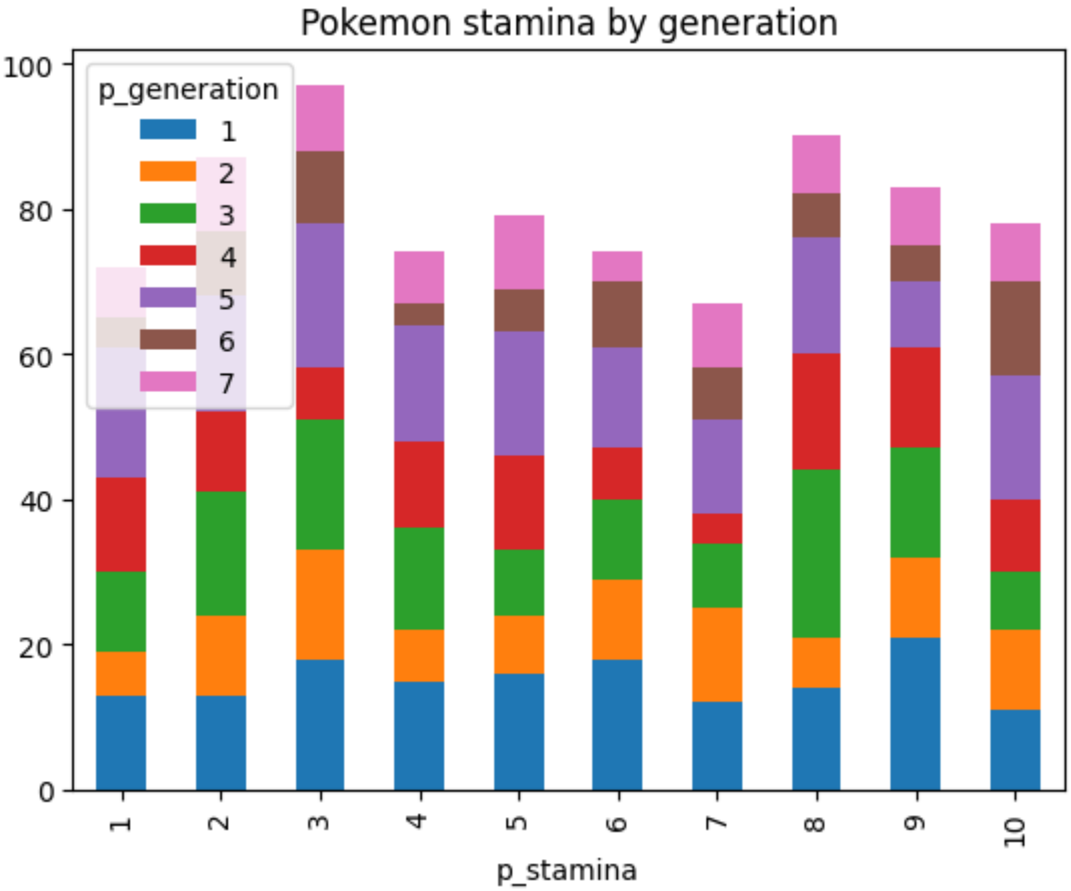
	pokedex_num	sp_attack	sp_defense	p_speed	p_generation	is_legendary	p_published
0	1	43	135	105	1	0	YES
1	2	58	196	24	1	0	YES
2	3	8	77	199	1	0	NO
3	4	73	20	69	1	0	YES
4	5	11	143	193	1	0	NO
5	6	124	174	112	1	0	NO
6	7	172	91	56	1	0	NO
7	8	109	62	75	1	0	YES
8	9	11	3	76	1	0	YES
9	10	25	15	16	1	0	NO

In [ ]:

```
# complete the syntex by creating a crosstab of the record based on stamina and gen
crosstab_01 = pd .crosstab(pokemon['p_stamina'], pokemon['p_generation'])
```

In [ ]:
```
# plot a bar graph (frequency), make sure it is stacked
crosstab_01 .plot(kind='bar', stacked = True , title = 'Pokemon stamina by generati
```

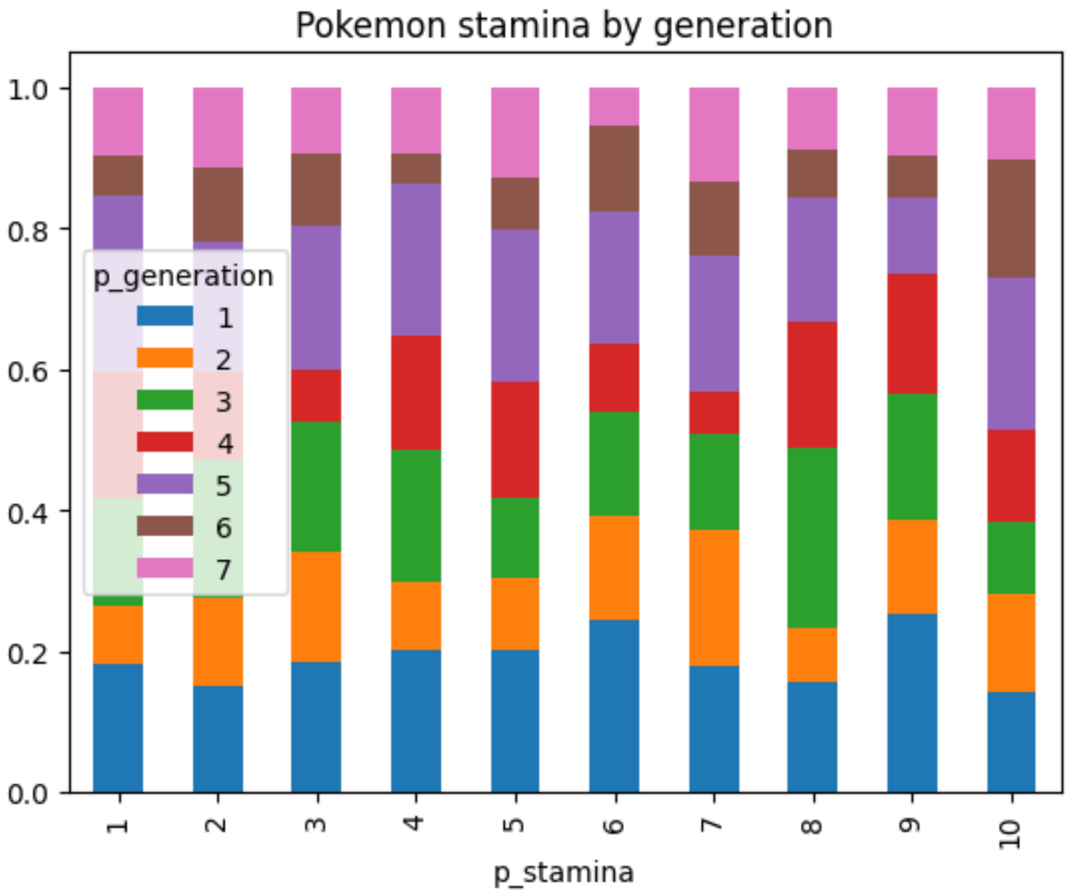
Out[ ]:
```
<Axes: title={'center': 'Pokemon stamina by generation'}, xlabel='p_stamina'>
```



In [ ]:
```
# create crosstab by using div and sum command.
crosstab_norm = crosstab_01 .div(crosstab_01.sum (1),axis=0 )
```

In [ ]:
```
# plot a normalized bar type crosstab data  with proportions
crosstab_norm.plot (kind='bar', stacked=True , title = 'Pokemon stamina by generati
```

Out[ ]:
```
<Axes: title={'center': 'Pokemon stamina by generation'}, xlabel='p_stamina'>
```



In [ ]:

```
# create a contingency table showing the generation and legendary
crosstab_02 = pd.crosstab(pokemon['p_generation'], pokemon['is_legendary' ])
pokemon .head(7)
```

Out[ ]:

	pokedex_num	sp_attack	sp_defense	p_speed	p_generation	is_legendary	p_published
0	1	43	135	105	1	0	YES
1	2	58	196	24	1	0	YES
2	3	8	77	199	1	0	NO
3	4	73	20	69	1	0	YES
4	5	11	143	193	1	0	NO
5	6	124	174	112	1	0	NO
6	7	172	91	56	1	0	NO

In [ ]:

```
# create a contingency table showing the generation and legendary by its percentage
round(crosstab_02.div(crosstab_02.sum(0),axis=1)*100,1)
```

Out[ ]:

is_legendary	0	1
p_generation		
1	20.0	7.1
2	12.9	8.6
3	17.1	14.3
4	12.9	18.6
5	19.6	18.6
6	9.0	8.6
7	8.6	24.3

In [ ]:

```
# import required package second task
import numpy as np
import matplotlib.pyplot as plt
```

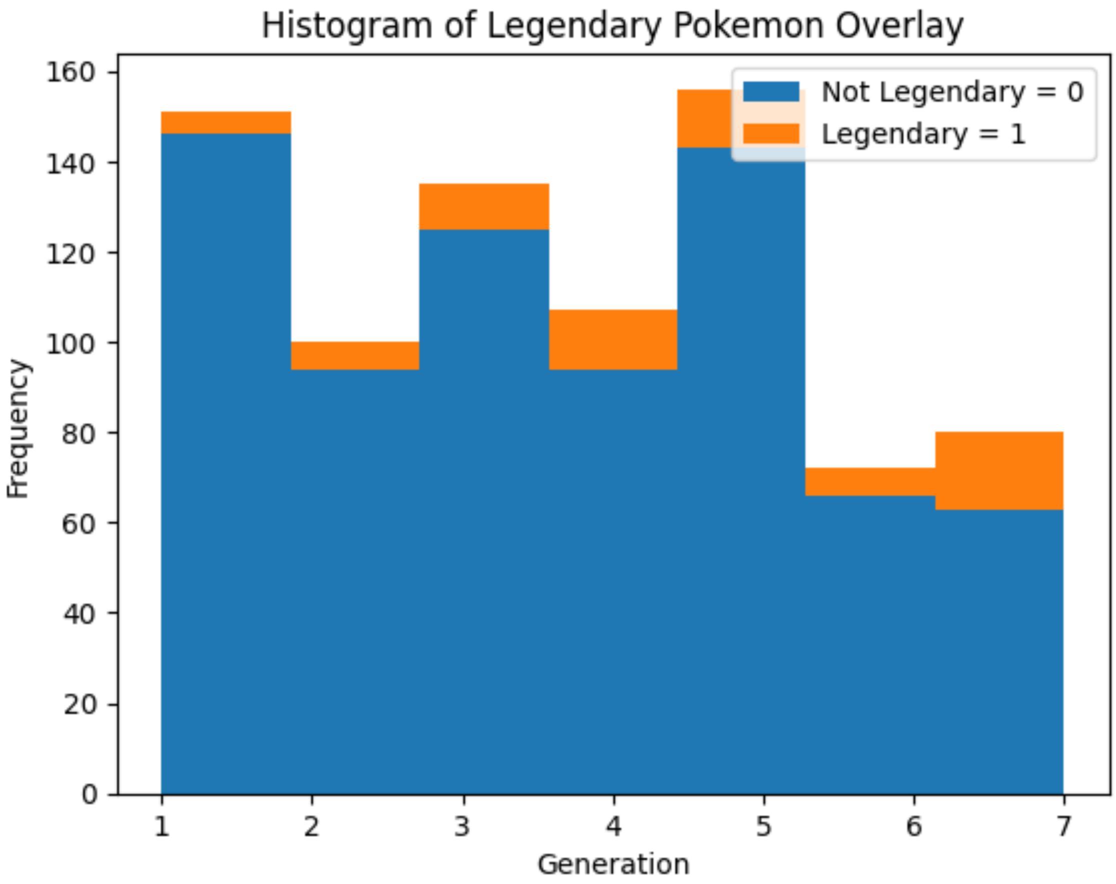
In [ ]:

```
# then using the percentage data, create a subset for each element of the overlay
# is legendary overlay generation
pok_y=pokemon[pokemon.is_legendary==0]['p_generation']
pok_n=pokemon[pokemon.is_legendary==1]['p_generation']
```

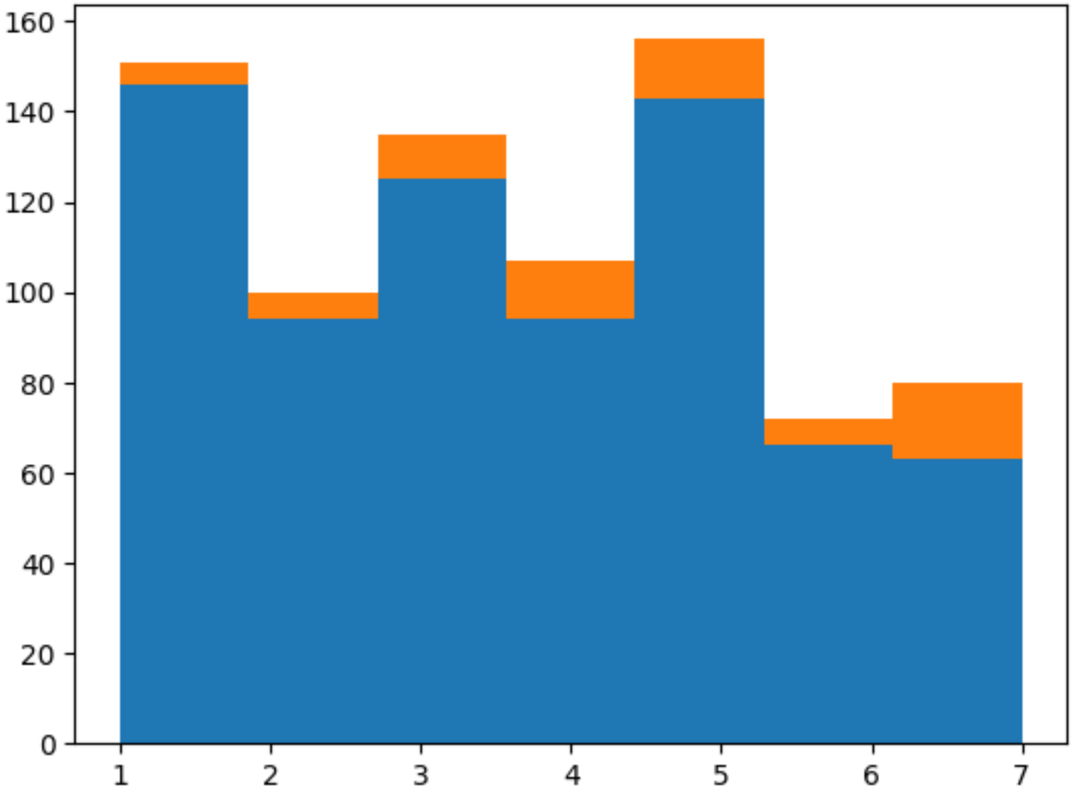
In [ ]:

```
# now create a histogram based on the two subsets, 7 bins

plt.hist([pok_y , pok_n ], bins = 7 , stacked = True)
plt.legend(['Not Legendary = 0', 'Legendary = 1'])
plt.title('Histogram of Legendary Pokemon Overlay')
plt.xlabel('Generation'); plt.ylabel('Frequency'); plt .show ()
```



```
In [ ]: # save the output from the non-normalized plot into variables
        (n, bins, patches) = plt.hist([pok_y ,pok_n ], bins =7 , stacked = True)
```



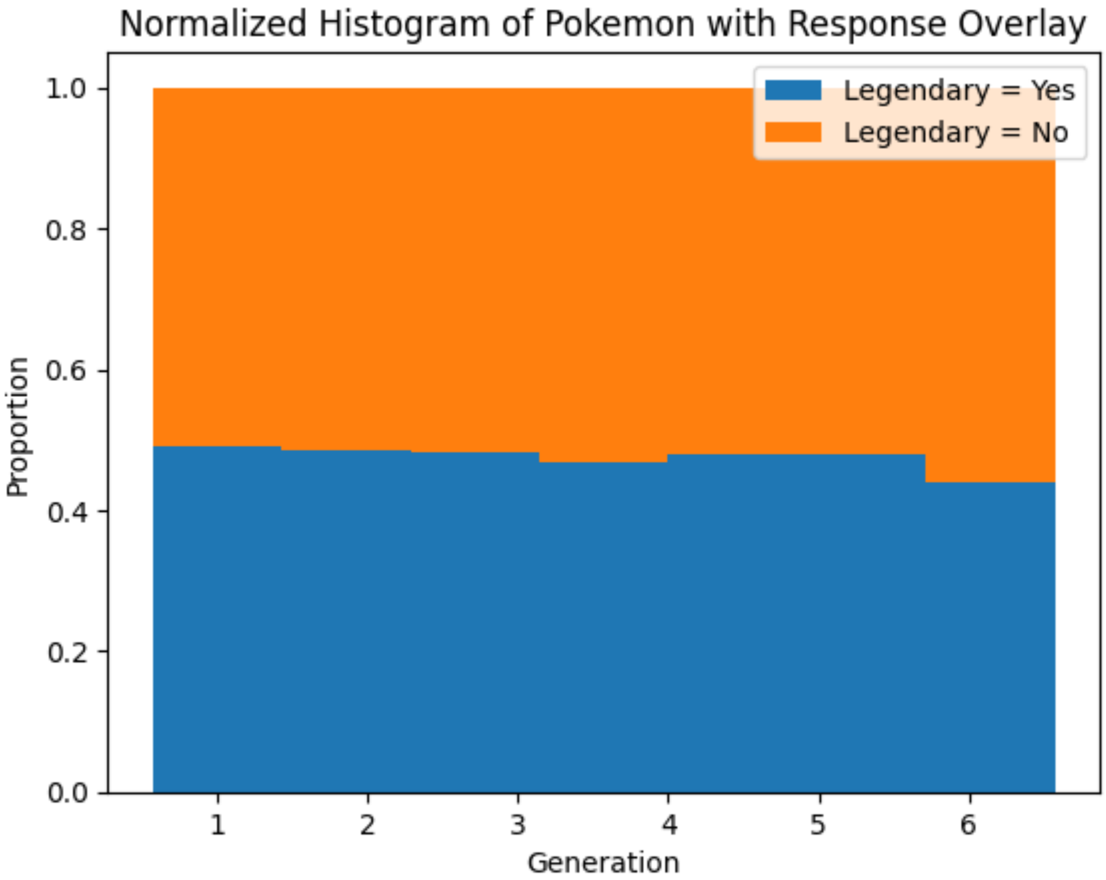
In [ ]:
```
# create a table and combine the height of the variables into single array
n_table = np.column_stack((n[0], n[1]))
```

In [ ]:
```
# divide each row by the sum of that row
# no revisions on this
n_norm = n_table  / n_table .sum(axis=1)[:, None]
```

In [ ]:
```
# determin upper and lower bounds of each bin (use the number of bins)
ourbins = np.column_stack((bins[0:7 ], bins[1:8 ]))
```

In [ ]:
```
# construct normalized plot plt.bar p1 and p2
p1 = plt.bar (x = ourbins[:,0], height = n_norm[:,0], width = ourbins[:, 1 ] - ourb
p2  = plt.bar (x = ourbins[:,0], height = n_norm[:,1 ], width = ourbins[:, 1 ] - ou

#plot the table
plt.legend(['Legendary = Yes', 'Legendary = No'])
plt.title('Normalized Histogram of Pokemon with Response Overlay')
plt.xlabel('Generation'); plt.ylabel('Proportion'); plt.show ()
```



In [ ]:
```
# use the cut function in Pandas to create the bins based on pokemon attack
# should be: Under 50, 50 to 75, 75 to 100, and over 100

pokemon['VAR'] = pd.cut (x = pokemon['sp_attack'], bins = [1 , 50 , 75 , 100 , 110
labels=["Under 50", "50 to 75", "75 to 100", "Over 100"], right = True)
```

In [ ]:
```
# create contingency table based on its type (legendary and non-legendary) and if p
crosstab_02 = pd.crosstab(pokemon['VAR'], pokemon['is_legendary'])
```

```
crosstab_02.head(4)
```

Out[ ]:

is_legendary	0	1
VAR		
Under 50	197	17
50 to 75	89	5
75 to 100	95	5
Over 100	41	3

In [ ]:

```
# craete a contingency table based on percentage
round(crosstab_02.div (crosstab_02.sum(0), axis = 1)*100, 1)
```

Out[ ]:

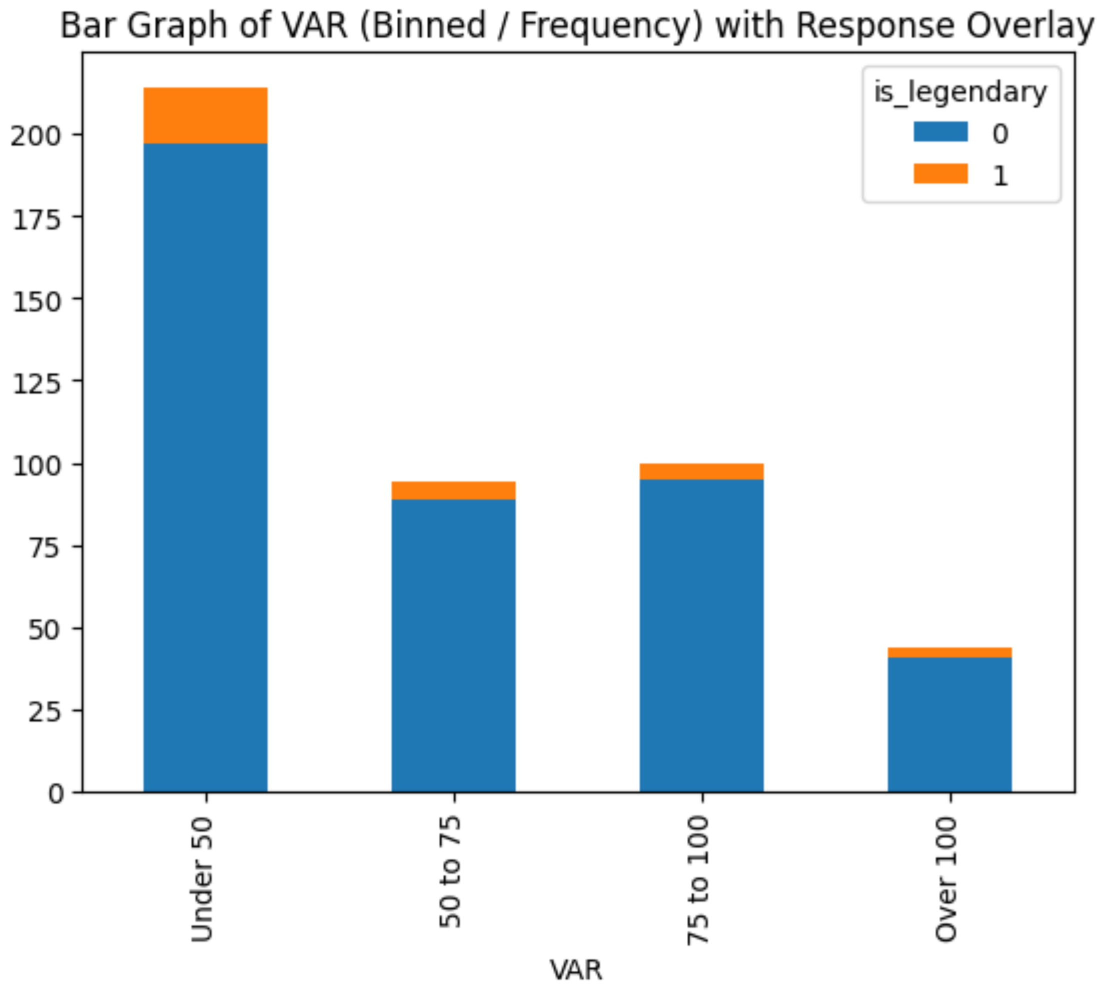
is_legendary	0	1
VAR		
Under 50	46.7	56.7
50 to 75	21.1	16.7
75 to 100	22.5	16.7
Over 100	9.7	10.0

In [ ]:

```
# then plot a binned bar graph of the crosstab data based on VAR (frequency)
crosstab_02.plot(kind='bar' , stacked = True , title = 'Bar Graph of VAR (Binned /
```

Out[ ]:

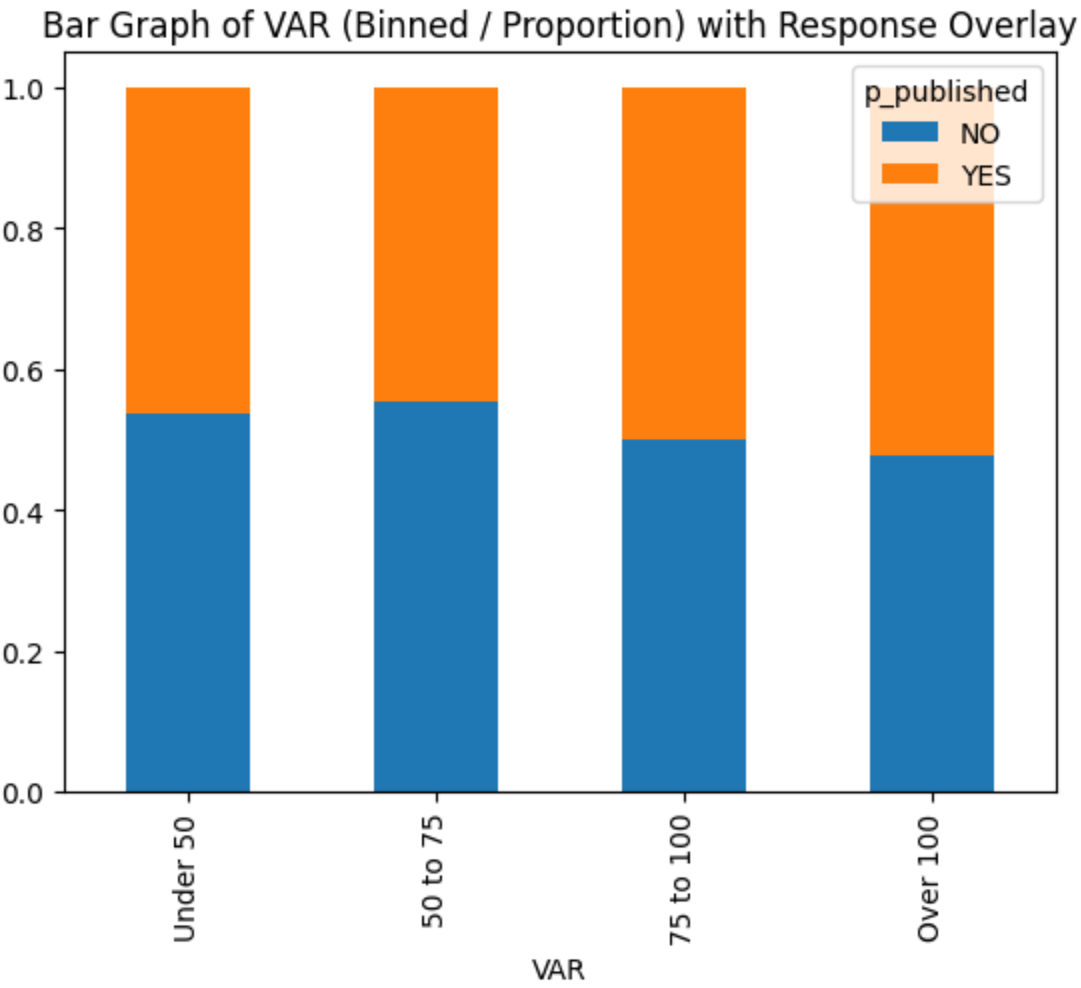
```
<Axes: title={'center': 'Bar Graph of VAR (Binned / Frequency) with Response Overl
ay'}, xlabel='VAR'>
```



```
In [ ]: crosstab_02 = pd.crosstab(pokemon['VAR'], pokemon['p_published'])
        crosstab_02_norm = crosstab_02.div (crosstab_02.sum(1), axis = 0)
```

```
In [ ]: # then plot a binned bar graph of the crosstab data based on VAR (normalized)
        crosstab_02_norm.plot (kind='bar', stacked = True, title = 'Bar Graph of VAR (Binne
```

```
Out[ ]: <Axes: title={'center': 'Bar Graph of VAR (Binned / Proportion) with Response Over
        lay'}, xlabel='VAR'>
```



In [ ]:

```
import datetime
import socket
def get_Host_name_IP():
    try:
        host_name = socket.gethostname()
        host_ip = socket.gethostbyname(host_name)
        print("Hostname-7:",host_name)
        print("IP Address:",host_ip)
    except:
        print("No visible IP Address")
get_Host_name_IP()
now = datetime.datetime.now()
print ("Time Stamp:", now.strftime("%Y-%m-%d %H:%M:%S"))
```

```
Hostname-7: LAPTOP-7LSO2L1V
IP Address: 192.168.56.1
Time Stamp: 2023-12-02 11:37:18
```

In [ ]:

```
#JHAINNO ALLRICK M. MARCOS
#FOPI01 CSS145
```

In [ ]: