In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt #likley won't be used much as i'm experimenting with plo
import plotly.graph_objects as go #you will be learning how go and px work with me!
import plotly.express as px

pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)

# Input data files are available in the read-only "../input/" directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [2]:
```
#load data
df = pd.read_csv('/Users/jalilkhan/Documents/Kaggle_Survay_Data_2020/kaggle_survey_2020_
df.shape
```

```
/var/folders/yg/h1y3gfrd6v9_sx5hgb2n7zh40000gn/T/ipykernel_14619/2581559655.py:2: DtypeW
arning: Columns (0) have mixed types. Specify dtype option on import or set low_memory=F
alse.
  df = pd.read_csv('/Users/jalilkhan/Documents/Kaggle_Survay_Data_2020/kaggle_survey_202
0_responses.csv')
```

Out[2]:
```
(20037, 355)
```

In [3]:
```
df.describe()
# describe the data
```

Out[3]:

	Time from Start to Finish (seconds)	Q1	Q2	Q3	Q4	Q5	Q6	Q7_Part_1	Q7_Part_2	Q7_Part_3	Q7_Part_4
count	20037	20037	20037	20037	19570	19278	19121	15531	4278	7536	3316
unique	5168	12	6	56	8	14	8	2	2	2	2
top	565	25-29	Man	India	Master's degree	Student	3-5 years	Python	R	SQL	C
freq	34	4011	15789	5851	7859	5171	4546	15530	4277	7535	3315

In [4]:
```
df.head()
# look into the individual data
```

Out[4]:

	Time from Start to Finish (seconds)	Q1	Q2	Q3	Q4	Q5	Q6	Q7_Part_1	Q7_Part_2	Q7_Part_3
0	Duration (in seconds)	What is your age (# years)?	What is your gender? - Selected Choice	In which country do you currently reside?	What is the highest level of formal education ...	Select the title most similar to your current ...	For how many years have you been	What programming languages do you use on a reg...	What programming languages do you use on a reg...	What programming languages do you use on a reg...

								writing code ...			
1	1838	35-39	Man	Colombia	Doctoral degree	Student	5-10 years	Python	R	SQL	
2	289287	30-34	Man	United States of America	Master's degree	Data Engineer	5-10 years	Python	R	SQL	
3	860	35-39	Man	Argentina	Bachelor's degree	Software Engineer	10-20 years	NaN	NaN	NaN	
4	507	30-34	Man	United States of America	Master's degree	Data Scientist	5-10 years	Python	NaN	SQL	

In [5]:
```
#remove the top row
# where we have the questions

df_fin = df.iloc[1:,:]
```

In [6]:
```
# now look at the head again
df_fin.head()
```

Out[6]:

	Time from Start to Finish (seconds)	Q1	Q2	Q3	Q4	Q5	Q6	Q7_Part_1	Q7_Part_2	Q7_Part_3	Q7_Part_4	Q7_P
1	1838	35-39	Man	Colombia	Doctoral degree	Student	5-10 years	Python	R	SQL	C	
2	289287	30-34	Man	United States of America	Master's degree	Data Engineer	5-10 years	Python	R	SQL	NaN	
3	860	35-39	Man	Argentina	Bachelor's degree	Software Engineer	10-20 years	NaN	NaN	NaN	NaN	
4	507	30-34	Man	United States of America	Master's degree	Data Scientist	5-10 years	Python	NaN	SQL	NaN	

5	78	30-34	Man	Japan	Master's degree	Software Engineer	3-5 years	Python	NaN	NaN	NaN

In [7]:
```
# get percent of null values in question

df_fin.isnull().sum() / df.shape[0]
```

Out[7]:
```
Time from Start to Finish (seconds)    0.000000
Q1                                     0.000000
Q2                                     0.000000
Q3                                     0.000000
Q4                                     0.023307
Q5                                     0.037880
Q6                                     0.045715
Q7_Part_1                              0.224884
Q7_Part_2                              0.786495
Q7_Part_3                              0.623896
Q7_Part_4                              0.834506
Q7_Part_5                              0.808953
Q7_Part_6                              0.831911
Q7_Part_7                              0.850477
Q7_Part_8                              0.986874
Q7_Part_9                              0.990068
Q7_Part_10                             0.911314
Q7_Part_11                             0.889305
Q7_Part_12                             0.989669
Q7_OTHER                               0.902880
Q8                                     0.110545
Q9_Part_1                              0.440435
Q9_Part_2                              0.809003
Q9_Part_3                              0.877926
Q9_Part_4                              0.706842
Q9_Part_5                              0.745471
Q9_Part_6                              0.835754
Q9_Part_7                              0.843639
Q9_Part_8                              0.877576
Q9_Part_9                              0.924989
Q9_Part_10                             0.919898
Q9_Part_11                             0.980686
Q9_OTHER                               0.941957
Q10_Part_1                             0.700903
Q10_Part_2                             0.684084
Q10_Part_3                             0.957179
Q10_Part_4                             0.990967
Q10_Part_5                             0.896541
Q10_Part_6                             0.994710
Q10_Part_7                             0.957728
Q10_Part_8                             0.975146
Q10_Part_9                             0.987723
Q10_Part_10                            0.939163
Q10_Part_11                            0.938514
Q10_Part_12                            0.980286
Q10_Part_13                            0.736338
Q10_OTHER                              0.975745
Q11                                    0.150072
Q12_Part_1                             0.585217
Q12_Part_2                             0.952039
Q12_Part_3                             0.606129
Q12_OTHER                              0.966712
Q13                                    0.162499
Q14_Part_1                             0.383990
```

Q14_Part_2	0.559715
Q14_Part_3	0.793831
Q14_Part_4	0.794330
Q14_Part_5	0.943205
Q14_Part_6	0.958577
Q14_Part_7	0.988322
Q14_Part_8	0.954035
Q14_Part_9	0.957529
Q14_Part_10	0.970205
Q14_Part_11	0.905275
Q14_OTHER	0.971752
Q15	0.182762
Q16_Part_1	0.488396
Q16_Part_2	0.653890
Q16_Part_3	0.690922
Q16_Part_4	0.790887
Q16_Part_5	0.962919
Q16_Part_6	0.989070
Q16_Part_7	0.803563
Q16_Part_8	0.910166
Q16_Part_9	0.952188
Q16_Part_10	0.973399
Q16_Part_11	0.982782
Q16_Part_12	0.953087
Q16_Part_13	0.975795
Q16_Part_14	0.995758
Q16_Part_15	0.939662
Q16_OTHER	0.981434
Q17_Part_1	0.472925
Q17_Part_2	0.560563
Q17_Part_3	0.743724
Q17_Part_4	0.817937
Q17_Part_5	0.963468
Q17_Part_6	0.832061
Q17_Part_7	0.707541
Q17_Part_8	0.948845
Q17_Part_9	0.826870
Q17_Part_10	0.935170
Q17_Part_11	0.963218
Q17_OTHER	0.979488
Q18_Part_1	0.893198
Q18_Part_2	0.899985
Q18_Part_3	0.896192
Q18_Part_4	0.824674
Q18_Part_5	0.945451
Q18_Part_6	0.942407
Q18_OTHER	0.996407
Q19_Part_1	0.894645
Q19_Part_2	0.924390
Q19_Part_3	0.972152
Q19_Part_4	0.928682
Q19_Part_5	0.947697
Q19_OTHER	0.995858
Q20	0.430853
Q21	0.436892
Q22	0.444478
Q23_Part_1	0.679493
Q23_Part_2	0.834257
Q23_Part_3	0.799970
Q23_Part_4	0.863952
Q23_Part_5	0.844288
Q23_Part_6	0.883216
Q23_Part_7	0.912462
Q23_OTHER	0.973898
Q24	0.464491
Q25	0.472426

Q26_A_Part_1	0.859909
Q26_A_Part_2	0.914858
Q26_A_Part_3	0.885861
Q26_A_Part_4	0.977941
Q26_A_Part_5	0.985676
Q26_A_Part_6	0.991516
Q26_A_Part_7	0.990368
Q26_A_Part_8	0.987573
Q26_A_Part_9	0.994211
Q26_A_Part_10	0.996157
Q26_A_Part_11	0.908270
Q26_A_OTHER	0.987723
Q27_A_Part_1	0.917553
Q27_A_Part_2	0.952588
Q27_A_Part_3	0.969257
Q27_A_Part_4	0.957229
Q27_A_Part_5	0.974747
Q27_A_Part_6	0.976893
Q27_A_Part_7	0.949294
Q27_A_Part_8	0.967460
Q27_A_Part_9	0.982632
Q27_A_Part_10	0.968957
Q27_A_Part_11	0.945950
Q27_A_OTHER	0.995009
Q28_A_Part_1	0.969407
Q28_A_Part_2	0.990418
Q28_A_Part_3	0.988821
Q28_A_Part_4	0.972900
Q28_A_Part_5	0.984678
Q28_A_Part_6	0.963917
Q28_A_Part_7	0.988322
Q28_A_Part_8	0.980686
Q28_A_Part_9	0.977691
Q28_A_Part_10	0.871538
Q28_A_OTHER	0.994909
Q29_A_Part_1	0.875780
Q29_A_Part_2	0.916355
Q29_A_Part_3	0.941259
Q29_A_Part_4	0.952238
Q29_A_Part_5	0.936867
Q29_A_Part_6	0.986325
Q29_A_Part_7	0.986475
Q29_A_Part_8	0.922194
Q29_A_Part_9	0.976743
Q29_A_Part_10	0.978290
Q29_A_Part_11	0.977841
Q29_A_Part_12	0.984529
Q29_A_Part_13	0.979488
Q29_A_Part_14	0.966562
Q29_A_Part_15	0.978540
Q29_A_Part_16	0.987823
Q29_A_Part_17	0.925388
Q29_A_OTHER	0.982283
Q30	0.824624
Q31_A_Part_1	0.991366
Q31_A_Part_2	0.917702
Q31_A_Part_3	0.968608
Q31_A_Part_4	0.993911
Q31_A_Part_5	0.908419
Q31_A_Part_6	0.986475
Q31_A_Part_7	0.996357
Q31_A_Part_8	0.984079
Q31_A_Part_9	0.997155
Q31_A_Part_10	0.995708
Q31_A_Part_11	0.991965
Q31_A_Part_12	0.996906

Q31_A_Part_13	0.991466
Q31_A_Part_14	0.846983
Q31_A_OTHER	0.982732
Q32	0.925188
Q33_A_Part_1	0.967310
Q33_A_Part_2	0.973699
Q33_A_Part_3	0.959275
Q33_A_Part_4	0.988771
Q33_A_Part_5	0.965963
Q33_A_Part_6	0.970904
Q33_A_Part_7	0.766432
Q33_A_OTHER	0.993362
Q34_A_Part_1	0.981933
Q34_A_Part_2	0.991865
Q34_A_Part_3	0.992614
Q34_A_Part_4	0.992414
Q34_A_Part_5	0.993961
Q34_A_Part_6	0.979139
Q34_A_Part_7	0.970904
Q34_A_Part_8	0.985627
Q34_A_Part_9	0.997754
Q34_A_Part_10	0.994410
Q34_A_Part_11	0.967859
Q34_A_OTHER	0.991915
Q35_A_Part_1	0.992514
Q35_A_Part_2	0.981434
Q35_A_Part_3	0.996457
Q35_A_Part_4	0.997105
Q35_A_Part_5	0.933822
Q35_A_Part_6	0.996656
Q35_A_Part_7	0.995608
Q35_A_Part_8	0.989420
Q35_A_Part_9	0.996257
Q35_A_Part_10	0.763787
Q35_A_OTHER	0.988421
Q36_Part_1	0.988621
Q36_Part_2	0.990667
Q36_Part_3	0.995359
Q36_Part_4	0.828567
Q36_Part_5	0.973649
Q36_Part_6	0.906223
Q36_Part_7	0.937715
Q36_Part_8	0.983381
Q36_Part_9	0.884314
Q36_OTHER	0.989519
Q37_Part_1	0.631532
Q37_Part_2	0.875281
Q37_Part_3	0.758247
Q37_Part_4	0.848630
Q37_Part_5	0.947148
Q37_Part_6	0.897340
Q37_Part_7	0.769077
Q37_Part_8	0.919249
Q37_Part_9	0.946249
Q37_Part_10	0.821880
Q37_Part_11	0.934771
Q37_OTHER	0.918301
Q38	0.336677
Q39_Part_1	0.856865
Q39_Part_2	0.866347
Q39_Part_3	0.893098
Q39_Part_4	0.608774
Q39_Part_5	0.886510
Q39_Part_6	0.655437
Q39_Part_7	0.925388
Q39_Part_8	0.682138

Q39_Part_9	0.867695
Q39_Part_10	0.921894
Q39_Part_11	0.965664
Q39_OTHER	0.976244
Q26_B_Part_1	0.738634
Q26_B_Part_2	0.811349
Q26_B_Part_3	0.743125
Q26_B_Part_4	0.912811
Q26_B_Part_5	0.947048
Q26_B_Part_6	0.971054
Q26_B_Part_7	0.975945
Q26_B_Part_8	0.979288
Q26_B_Part_9	0.978789
Q26_B_Part_10	0.984778
Q26_B_Part_11	0.933872
Q26_B_OTHER	0.991815
Q27_B_Part_1	0.881369
Q27_B_Part_2	0.875231
Q27_B_Part_3	0.915906
Q27_B_Part_4	0.875780
Q27_B_Part_5	0.908070
Q27_B_Part_6	0.910965
Q27_B_Part_7	0.837650
Q27_B_Part_8	0.867695
Q27_B_Part_9	0.901532
Q27_B_Part_10	0.877077
Q27_B_Part_11	0.954285
Q27_B_OTHER	0.994909
Q28_B_Part_1	0.915007
Q28_B_Part_2	0.917952
Q28_B_Part_3	0.930129
Q28_B_Part_4	0.878175
Q28_B_Part_5	0.930429
Q28_B_Part_6	0.831162
Q28_B_Part_7	0.897989
Q28_B_Part_8	0.862355
Q28_B_Part_9	0.858312
Q28_B_Part_10	0.938963
Q28_B_OTHER	0.995558
Q29_B_Part_1	0.795279
Q29_B_Part_2	0.893347
Q29_B_Part_3	0.895044
Q29_B_Part_4	0.924590
Q29_B_Part_5	0.847033
Q29_B_Part_6	0.965164
Q29_B_Part_7	0.962719
Q29_B_Part_8	0.901931
Q29_B_Part_9	0.957529
Q29_B_Part_10	0.942107
Q29_B_Part_11	0.956680
Q29_B_Part_12	0.965414
Q29_B_Part_13	0.946449
Q29_B_Part_14	0.907421
Q29_B_Part_15	0.898887
Q29_B_Part_16	0.945750
Q29_B_Part_17	0.936168
Q29_B_OTHER	0.991466
Q31_B_Part_1	0.843090
Q31_B_Part_2	0.954734
Q31_B_Part_3	0.870739
Q31_B_Part_4	0.985676
Q31_B_Part_5	0.813196
Q31_B_Part_6	0.959076
Q31_B_Part_7	0.972201
Q31_B_Part_8	0.971153
Q31_B_Part_9	0.990917

```
Q31_B_Part_10                        0.990917
Q31_B_Part_11                        0.982832
Q31_B_Part_12                        0.989819
Q31_B_Part_13                        0.949743
Q31_B_Part_14                        0.894395
Q31_B_OTHER                          0.990967
Q33_B_Part_1                         0.878774
Q33_B_Part_2                         0.873185
Q33_B_Part_3                         0.836552
Q33_B_Part_4                         0.924590
Q33_B_Part_5                         0.880321
Q33_B_Part_6                         0.842192
Q33_B_Part_7                         0.894745
Q33_B_OTHER                          0.990567
Q34_B_Part_1                         0.876229
Q34_B_Part_2                         0.957079
Q34_B_Part_3                         0.959974
Q34_B_Part_4                         0.958377
Q34_B_Part_5                         0.977691
Q34_B_Part_6                         0.880970
Q34_B_Part_7                         0.862604
Q34_B_Part_8                         0.932675
Q34_B_Part_9                         0.984229
Q34_B_Part_10                        0.964665
Q34_B_Part_11                        0.959824
Q34_B_OTHER                          0.990318
Q35_B_Part_1                         0.953286
Q35_B_Part_2                         0.941209
Q35_B_Part_3                         0.975296
Q35_B_Part_4                         0.978490
Q35_B_Part_5                         0.840295
Q35_B_Part_6                         0.972152
Q35_B_Part_7                         0.975994
Q35_B_Part_8                         0.957728
Q35_B_Part_9                         0.974048
Q35_B_Part_10                        0.846135
Q35_B_OTHER                          0.987423
dtype: float64
```

Part of EDA is finding a way to make the data useful to you. I wanted to make it easy to run analysis on individual questions if I wanted to. The most practical way I found was to put all the questions in a dictionary. Each key in the dictionary is the Question number and each value is a dataframe with the parts to the question. I could now easily pull data for individual questions rather than filtering every time. This is particularly important for questions with multiple parts.

In [8]:
```
#create a dictionary for questions
Questions = {}

#create list of questions
#not very efficient, but keeps things ordered
qnums = list(dict.fromkeys([i.split('_')[0] for i in df_fin.columns]))
qnums
```

Out[8]:
```
['Time from Start to Finish (seconds)',
 'Q1',
 'Q2',
 'Q3',
 'Q4',
 'Q5',
 'Q6',
 'Q7',
 'Q8',
 'Q9',
 'Q10',
```

```
 'Q11',
 'Q12',
 'Q13',
 'Q14',
 'Q15',
 'Q16',
 'Q17',
 'Q18',
 'Q19',
 'Q20',
 'Q21',
 'Q22',
 'Q23',
 'Q24',
 'Q25',
 'Q26',
 'Q27',
 'Q28',
 'Q29',
 'Q30',
 'Q31',
 'Q32',
 'Q33',
 'Q34',
 'Q35',
 'Q36',
 'Q37',
 'Q38',
 'Q39']
```

In [9]:
```
#add data for each question to key value pairs in dictionary
for i in qnums:
    if i in ['Q1','Q2','Q3']: #since we are using .startswith() below this prevents all
        Questions[i] = df_fin[i] #[1,2,3] from going in the key value pair (Example in v
    else:
        Questions[i] = df_fin[[q for q in df_fin.columns if q.startswith(i)]]
```

Q1 & Q7 Examples to explain px vs go plotly express (px) --> takes the data frame in as a parameter and you use other paramaters to mainipulate the columns. I think this is better that allows you to work with a full dataframe.

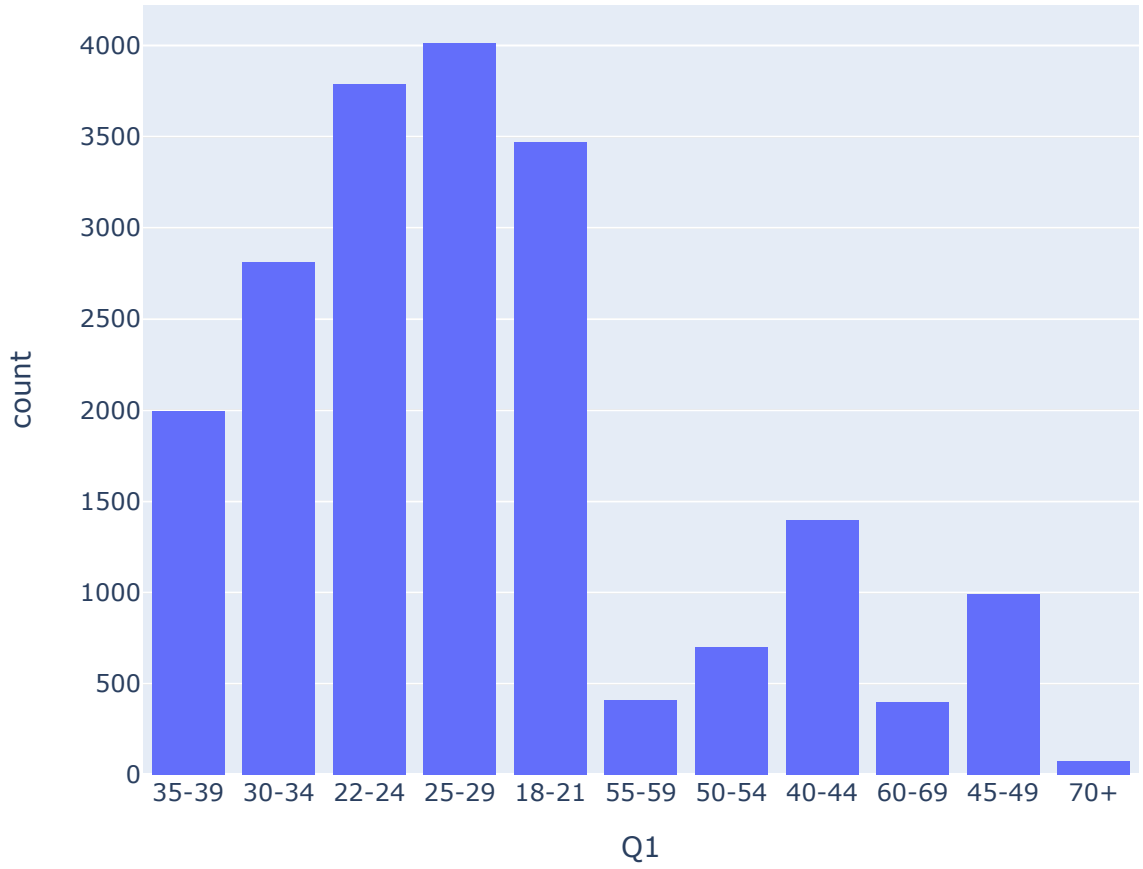
plotly graph objects (go) --> Takes in just the data as parameters. In this case you manipulate the data before passing it in. This is a bit more flexbile for questions like Q7 where there are columns for each answer type.

In [10]:
```
df_fin.Q1
```

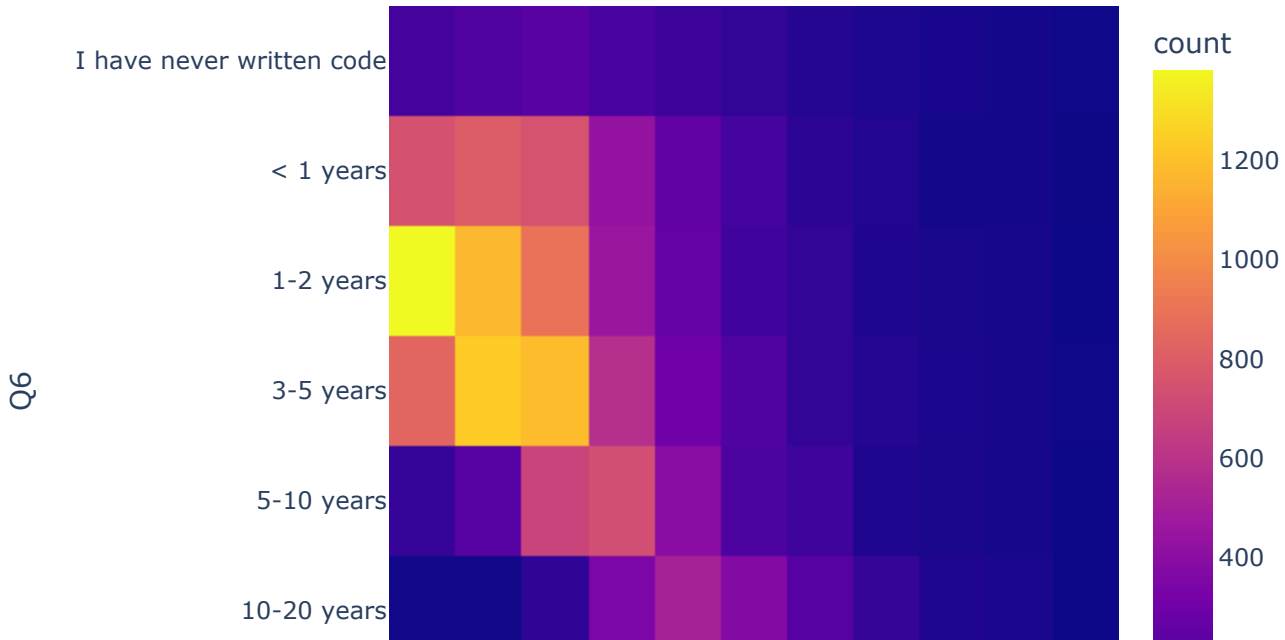
Out[10]:
```
1         35-39
2         30-34
3         35-39
4         30-34
5         30-34
          ...
20032     18-21
20033     55-59
20034     30-34
20035     22-24
20036     22-24
Name: Q1, Length: 20036, dtype: object
```

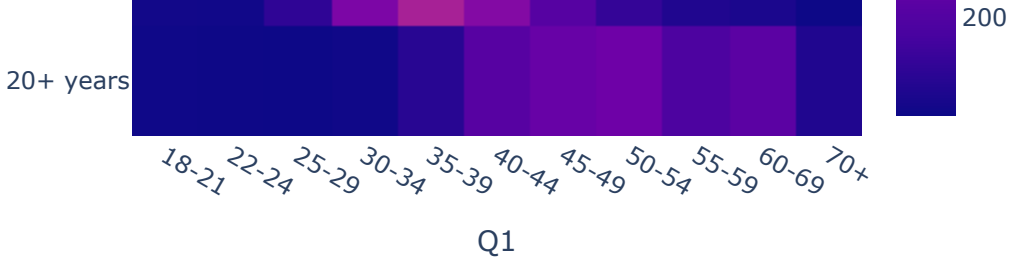
In [11]:
```
#q1 histogram using px
```

```
fig = px.histogram(df_fin, x = 'Q1')
fig.show()
```



In [12]:
```
# heatmap using px for q1 & q6

fig = px.density_heatmap(df_fin, x='Q1', y='Q6', category_orders={'Q1':['18-21','22-24',
fig.show()
```





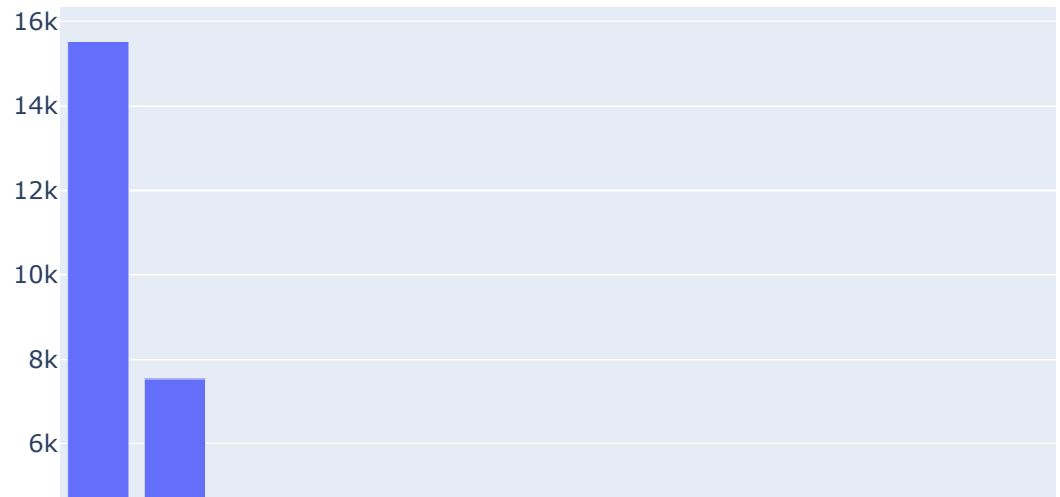
In [13]:
```
Questions['Q7']
```

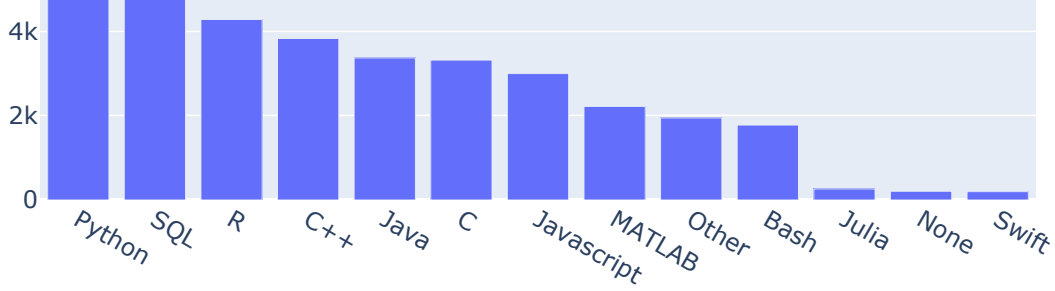
Out[13]:

	Q7_Part_1	Q7_Part_2	Q7_Part_3	Q7_Part_4	Q7_Part_5	Q7_Part_6	Q7_Part_7	Q7_Part_8	Q7_Part_9	Q7_I
1	Python	R	SQL	C	NaN	NaN	Javascript	NaN	NaN	
2	Python	R	SQL	NaN	NaN	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	Java	Javascript	NaN	NaN	
4	Python	NaN	SQL	NaN	NaN	NaN	NaN	NaN	NaN	
5	Python	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	
20032	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
20033	Python	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
20034	Python	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
20035	Python	NaN	SQL	C	NaN	Java	Javascript	NaN	NaN	
20036	Python	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

20036 rows × 13 columns

In [14]:
```
# Q7 example for go use. We aggregate the data beforehand with .value_counts()
Questions['Q7'].columns = list(Questions['Q7'].mode().iloc[0,:])
q7 = Questions['Q7'].count().reset_index()
q7.columns = ['language','Count']
q7 = q7.sort_values('Count', ascending = False)
fig = go.Figure([go.Bar(x = q7.language, y = q7.Count)])
fig.show()
```





The main thing I wanted to understand through this analysis was position by roles. I used a similar process as above to create a dictionary where they roles were the keys and the dataframes filtered by role were the value pairs. This might not have been the most efficient approach, but with a relatively small dataset like this, I valued ease of use over compute time.

In [15]:
```
#Create dictionary with role / data key value pairs
Roles = {}
for i in df_fin.Q5.unique():
    Roles[i] = df_fin[df_fin.Q5 == i]
```

In [16]:
```
Roles.keys()
```

Out[16]:
```
dict_keys(['Student', 'Data Engineer', 'Software Engineer', 'Data Scientist', 'Data Analyst', 'Research Scientist', 'Other', 'Currently not employed', 'Statistician', 'Product/Project Manager', 'Machine Learning Engineer', nan, 'Business Analyst', 'DBA/Database Engineer'])
```

dict_keys(['Student', 'Data Engineer', 'Software Engineer', 'Data Scientist', 'Data Analyst', 'Research Scientist', 'Other', 'Currently not employed', 'Statistician', 'Product/Project Manager', 'Machine Learning Engineer', nan, 'Business Analyst', 'DBA/Database Engineer'])

In [17]:
```
Roles['Student']
```

Out[17]:

	Time from Start to Finish (seconds)	Q1	Q2	Q3	Q4	Q5	Q6	Q7_Part_1	Q7_Part_2	Q7_Part_3	Q7_Part_
1	1838	35-39	Man	Colombia	Doctoral degree	Student	5-10 years	Python	R	SQL	
7	748	22-24	Man	Brazil	Bachelor's degree	Student	3-5 years	Python	R	NaN	
8	171196	25-29	Woman	China	Master's degree	Student	< 1 years	NaN	R	NaN	Nal
10	150	22-24	Man	China	No formal education past high school	Student	< 1 years	Python	NaN	SQL	Nal

11	7469	18-21	Man	India	Bachelor's degree	Student	1-2 years	Python	R	SQL	
...	...	...	...	...	...	...	...	...	...	...	
20017	374	18-21	Prefer not to say	China	Bachelor's degree	Student	1-2 years	Python	NaN	NaN	Nal
20021	238	18-21	Woman	United States of America	Bachelor's degree	Student	3-5 years	Python	NaN	NaN	Nal
20024	221	18-21	Man	India	Bachelor's degree	Student	3-5 years	Python	NaN	NaN	Nal
20026	1158	22-24	Man	United States of America	Master's degree	Student	1-2 years	Python	NaN	NaN	
20028	739	25-29	Man	India	Master's degree	Student	3-5 years	Python	NaN	SQL	

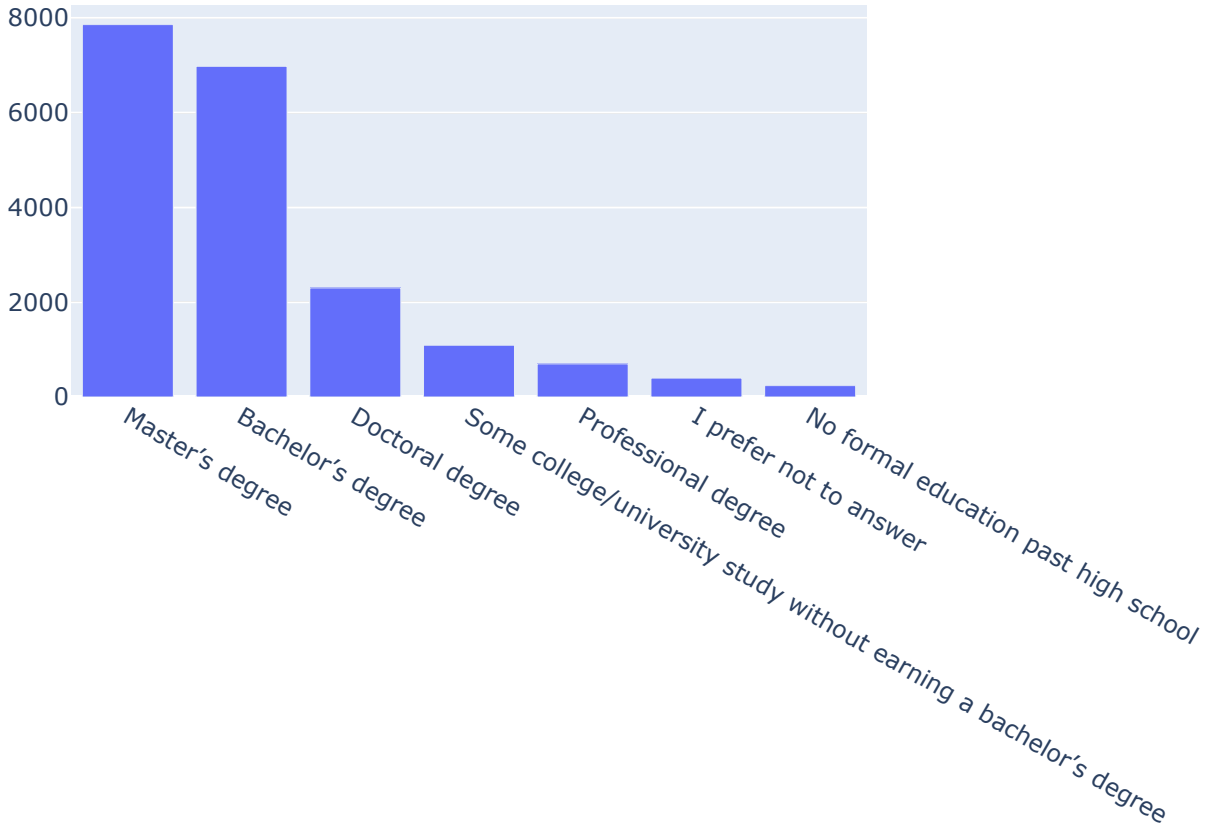
5171 rows × 355 columns

first subquestion --> How does education level vary by role

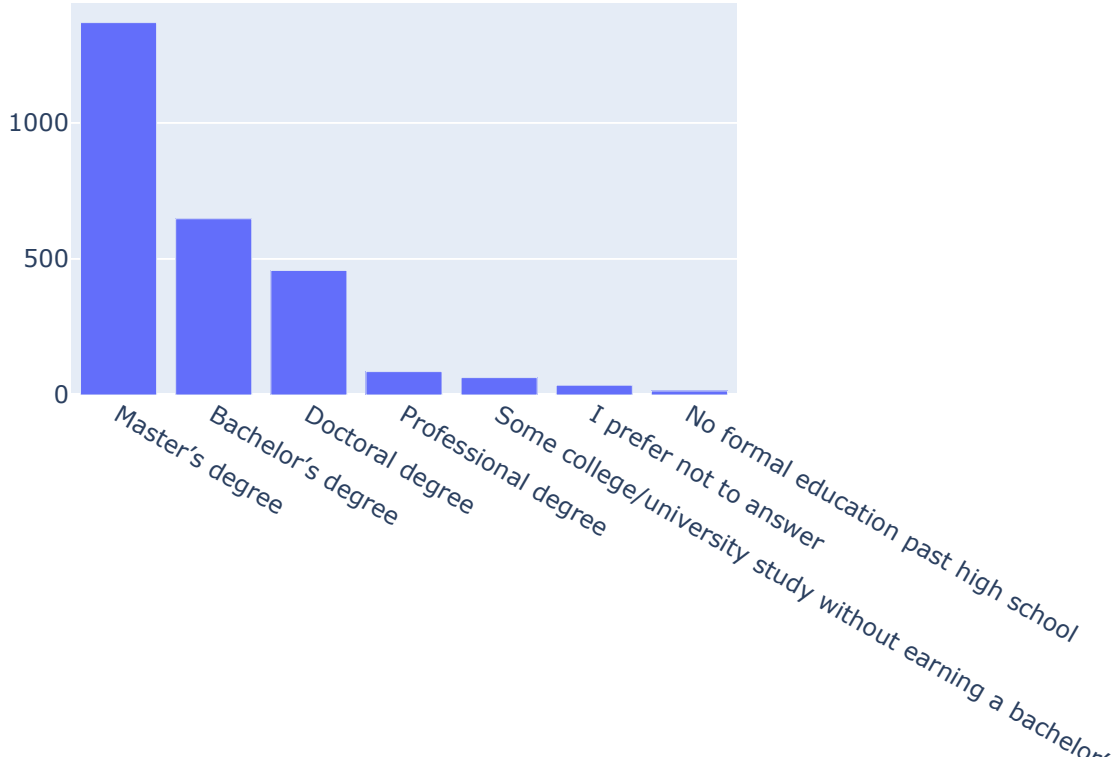
In [18]:
```
#all education graph
edu = df_fin.Q4.value_counts()
edu
```

Out[18]:
```
Master's degree                                                 7859
Bachelor's degree                                               6978
Doctoral degree                                                 2302
Some college/university study without earning a bachelor's degree 1092
Professional degree                                              699
I prefer not to answer                                           399
No formal education past high school                             240
Name: Q4, dtype: int64
```

In [19]:
```
#education across whole survey sample
fig = go.Figure([go.Bar(x=edu.index, y=edu.values)])
fig.show()
```



In [20]:
```
#education for just data scientists
ds_edu = Roles['Data Scientist'].Q4.value_counts()
fig = go.Figure([go.Bar(x= ds_edu.index, y=ds_edu.values)])
fig.show()
```

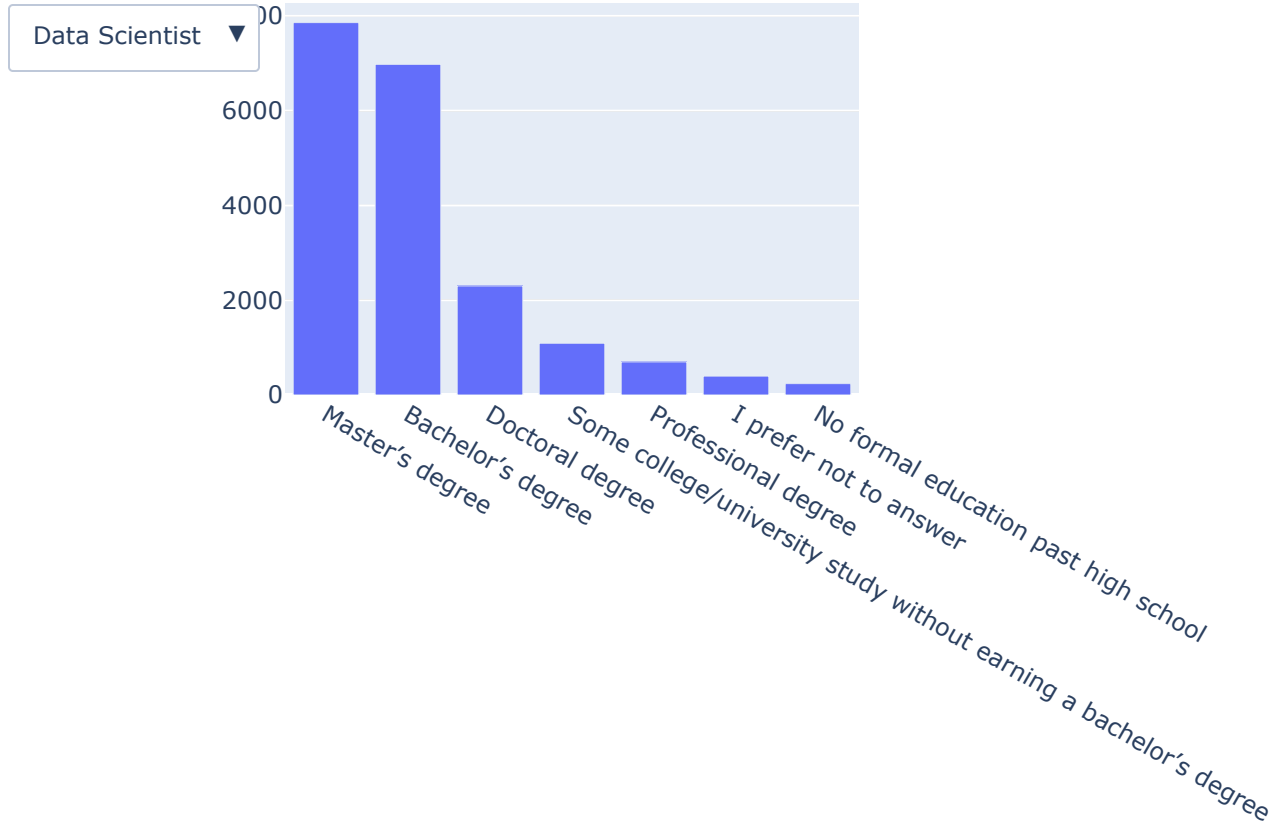


Building an Advanced Graph

I wanted to try to compare education levels between different career tracks. A great thing about plotly is that it is interactive. I wanted to explore these features to build a graph that uses a dropdown to compare different roles. The below graphs are the iterations of how I came to the final graph.

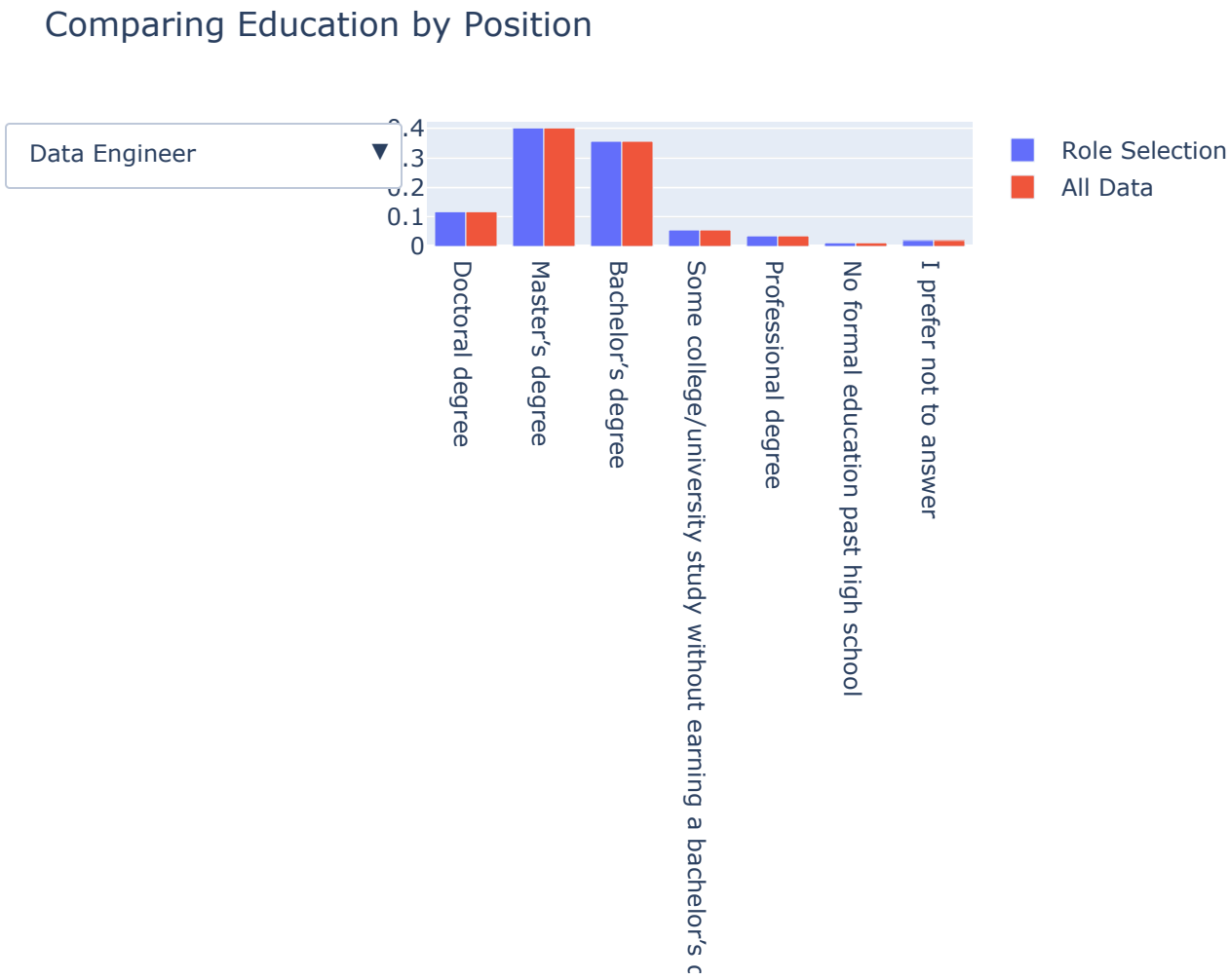
In [21]:
```
########################################
# First Iteration - Basic dropdown
########################################

#https://stackoverflow.com/questions/59406167/plotly-how-to-filter-a-pandas-dataframe-us
#https://plotly.com/python/dropdowns/

fig = go.Figure()
fig.add_trace(go.Bar(x= edu.index, y=edu.values))

#buttons are the things you see in the dropdown
buttons = []

#for each graph we want to show, we need a button for it
#you can do a lot with dropdowns, not just replace data
buttons.append(dict(method='restyle',
                    label='Data Scientist',
                    visible=True,
                    args=[{'y':[Roles['Data Scientist'].Q4.value_counts().values],
                           'x':[Roles['Data Scientist'].Q4.value_counts().index],
                           'type':'bar'}, [0]],
                    )
               )
buttons.append(dict(method='restyle',
                    label='Student',
                    visible=True,
                    args=[{'y':[Roles['Student'].Q4.value_counts().values],
                           'x':[Roles['Student'].Q4.value_counts().index],
                           'type':'bar'}, [0]],
                    )
               )
buttons.append(dict(method='restyle',
                    label='Data Analyst',
                    visible=True,
                    args=[{'y':[Roles['Data Analyst'].Q4.value_counts().values],
                           'x':[Roles['Data Analyst'].Q4.value_counts().index],
                           'type':'bar'}, [0]],
                    )
               )

#to get a menu to show, you need to create an updatemenu.
#at this point I had no clue how it worked, I just was trying to get something to run

updatemenu = []
your_menu = {}
updatemenu.append(your_menu)

updatemenu[0]['buttons'] = buttons
updatemenu[0]['direction'] = 'down'
updatemenu[0]['showactive'] = True

# add dropdown menus to the figure
```

```
fig.update_layout(showlegend=False, updatemenus=updatemenu)
fig.show()
```



In [22]:
```
########################################
# Second Iteration - Comparison Chart vs Baseline
########################################

#Added title to the figure
fig = go.Figure(layout=go.Layout(title= go.layout.Title(text="Comparing Education by Pos

#change to percent of group rather than raw numbers
fig.add_trace(go.Bar(name= 'Role Selection', x= edu.index, y=(edu.values/ edu.values.sum

#added another trace, this is the second series of bars
fig.add_trace(go.Bar(name= 'All Data',x= edu.index, y=(edu.values/ edu.values.sum())))

#updatemenu = []
buttons = []

#add all roles with a loop, in previous we added them individually.
for i in list(Roles.keys())[1:]:
    buttons.append(dict(method='restyle',
                        label= i,
                        visible=True,
                        args=[{'y':[Roles[i].Q4.value_counts().values/Roles[i].Q4.value_
                               'x':[Roles[i].Q4.value_counts().index],
                               'type':'bar'}, [0]],
                        )
                   )


#at this point I still didn't understand how this worked, I just knew it didn't add a dr
```

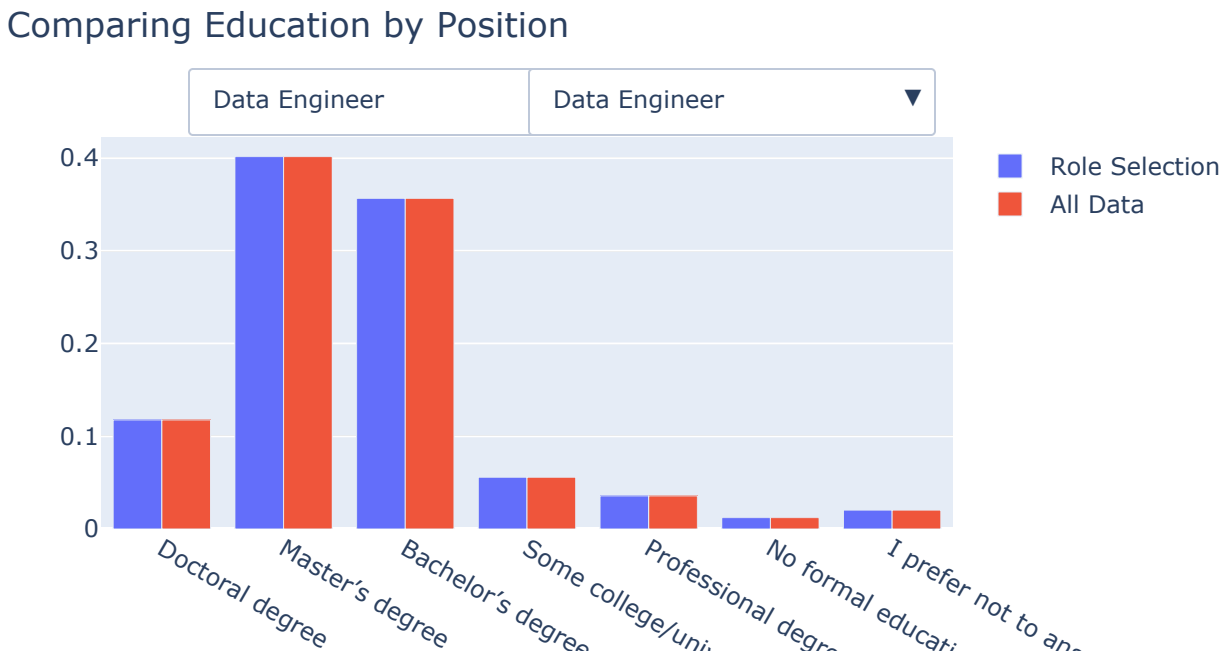
```
updatemenu = []
your_menu = {}
updatemenu.append(your_menu)

updatemenu[0]['buttons'] = buttons
updatemenu[0]['direction'] = 'down'
updatemenu[0]['showactive'] = True

# add dropdown menus to the figure
fig.update_layout( updatemenus=updatemenu)

#order axes https://plotly.com/python/categorical-axes/
fig.update_xaxes(categoryorder= 'array', categoryarray= ["Doctoral degree",'Master's deg
fig.show()
```

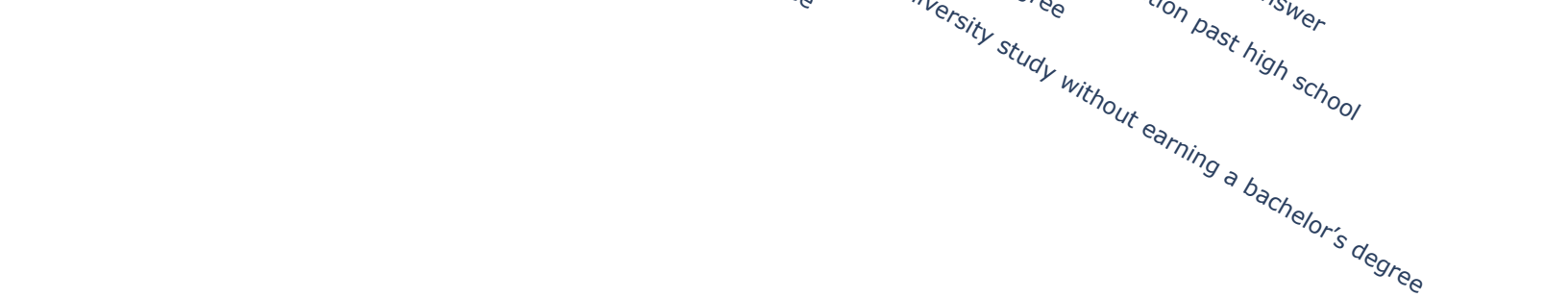


In [23]:

```
########################################
# Third Iteration - Two Drop Down Comparison
########################################

fig = go.Figure(layout=go.Layout(title= go.layout.Title(text="Comparing Education by Pos
fig.add_trace(go.Bar(name= 'Role Selection', x= edu.index, y=(edu.values/ edu.values.sum

buttons = []
# add buttons for first series of bars
for i in list(Roles.keys())[1:]:
    buttons.append(dict(method='restyle',
                        label= i,
                        visible=True,
                        args=[{'y':[Roles[i].Q4.value_counts().values/Roles[i].Q4.value_
                               'x':[Roles[i].Q4.value_counts().index],
                               'type':'bar'}, [0]], # the [0] at the end lets us know th
                        )
```

```
                    )

fig.add_trace(go.Bar(name= 'All Data',x= edu.index, y=(edu.values/ edu.values.sum())))

buttons2 = []
# add buttons for second series of bars
for i in list(Roles.keys())[1:]:
    buttons2.append(dict(method='restyle',
                         label= i,
                         visible=True,
                         args=[{'y':[Roles[i].Q4.value_counts().values/Roles[i].Q4.value_
                                'x':[Roles[i].Q4.value_counts().index],
                                'type':'bar'}, [1]], # the [1] at the end lets us know th
                         )                          #literally figured that out by just exp
                    )
# adjusted dropdown placement
#found out updatemenus take a dictionary of buttons and allow you to format how the drop
# https://plotly.com/python/dropdowns/
button_layer_1_height = 1.23
updatemenus = list([
    dict(buttons=buttons,
            direction="down",
            pad={"r": 10, "t": 10},
            showactive=True,
            x=0.1,
            xanchor="left",
            y=button_layer_1_height,
            yanchor="top"),
    dict(buttons=buttons2,
            direction="down",
            pad={"r": 10, "t": 10},
            showactive=True,
            x=0.5,
            xanchor="left",
            y=button_layer_1_height,
            yanchor="top")])

fig.update_layout( updatemenus=updatemenus)
fig.update_xaxes(categoryorder= 'array', categoryarray= ["Doctoral degree",'Master's deg
fig.show()

#add topline to each for all types
# add seleciton 1 and selection 2
```

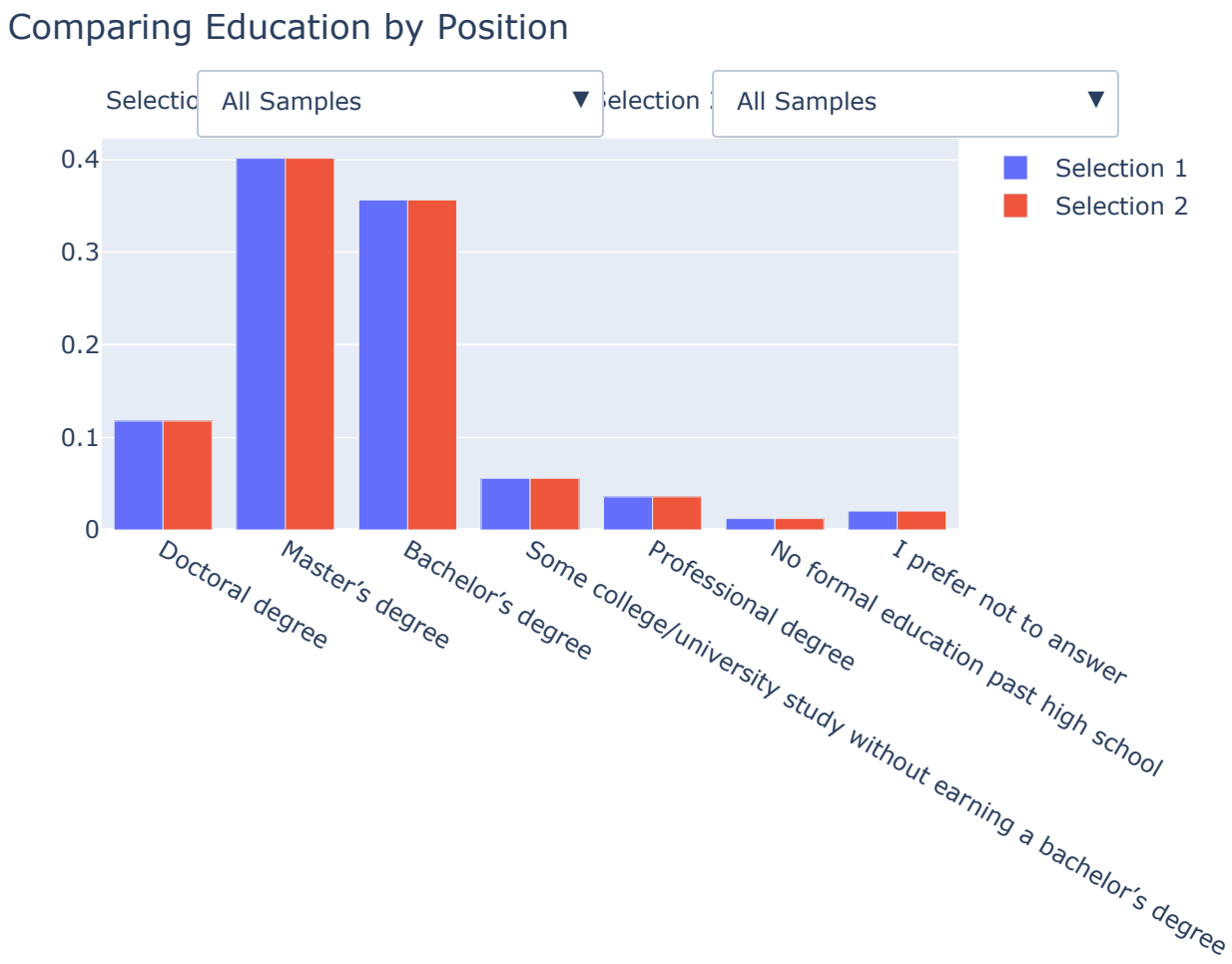




In [24]:

```
########################################
# Final Iteration - Touch-ups
########################################
fig = go.Figure(layout=go.Layout(title= go.layout.Title(text="Comparing Education by Pos
#changed from role selection to selection 1
fig.add_trace(go.Bar(name= 'Selection 1', x= edu.index, y=(edu.values/ edu.values.sum())

buttons = []

#added button for all data comparison
buttons.append(dict(method='restyle',
                        label= 'All Samples',
                        visible=True,
                        args=[{'y':[df_fin.Q4.value_counts().values/df_fin.Q4.value_coun
                               'x':[df_fin.Q4.value_counts().index],
                               'type':'bar'}, [0]], # the [0] at the end lets us know th
                        )
                    )

for i in list(Roles.keys())[1:]:
    buttons.append(dict(method='restyle',
                        label= i,
                        visible=True,
                        args=[{'y':[Roles[i].Q4.value_counts().values/Roles[i].Q4.value_
                               'x':[Roles[i].Q4.value_counts().index],
                               'type':'bar'}, [0]], # the [0] at the end lets us know th
                        )
                    )

fig.add_trace(go.Bar(name= 'Selection 2',x= edu.index, y=(edu.values/ edu.values.sum()))

buttons2 = []
#added button for all data comparison
buttons2.append(dict(method='restyle',
                        label= 'All Samples',
                        visible=True,
                        args=[{'y':[df_fin.Q4.value_counts().values/df_fin.Q4.value_coun
                               'x':[df_fin.Q4.value_counts().index],
                               'type':'bar'}, [1]], # the [0] at the end lets us know th
                        )
                    )

for i in list(Roles.keys())[1:]:
    buttons2.append(dict(method='restyle',
                        label= i,
                        visible=True,
                        args=[{'y':[Roles[i].Q4.value_counts().values/Roles[i].Q4.value_
                               'x':[Roles[i].Q4.value_counts().index],
                               'type':'bar'}, [1]], # the [1] at the end lets us know th
                        )                          #literally figured that out by just exp
                    )
# adjusted dropdown placement
#found out updatemenus take a dictionary of buttons and allow you to format how the drop
# https://plotly.com/python/dropdowns/
```

```
button_layer_1_height = 1.23
updatemenus = list([
    dict(buttons=buttons,
            direction="down",
            pad={"r": 10, "t": 10},
            showactive=True,
            x=0.11,
            xanchor="left",
            y=button_layer_1_height,
            yanchor="top"),
    dict(buttons=buttons2,
            direction="down",
            pad={"r": 10, "t": 10},
            showactive=True,
            x=0.71,
            xanchor="left",
            y=button_layer_1_height,
            yanchor="top")])

fig.update_layout( updatemenus=updatemenus)
#added annotations next to dropdowns
fig.update_layout(
    annotations=[
        dict(text="Selection 1", x=0, xref="paper", y=1.15, yref="paper",
                                align="left", showarrow=False),
        dict(text="Selection 2", x=0.65, xref="paper", y=1.15,
                                yref="paper", showarrow=False)
    ])
fig.update_xaxes(categoryorder= 'array', categoryarray= ["Doctoral degree",'Master's deg
fig.show()
```



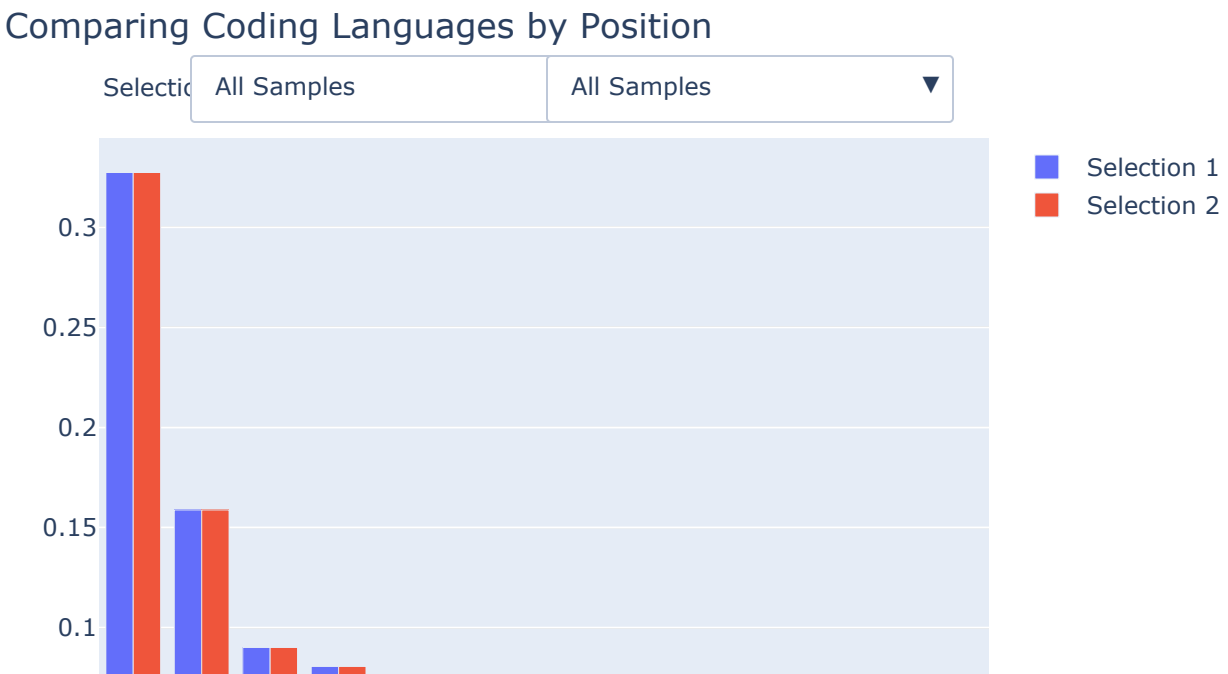
Part 3 (Building Advanced Graphs)

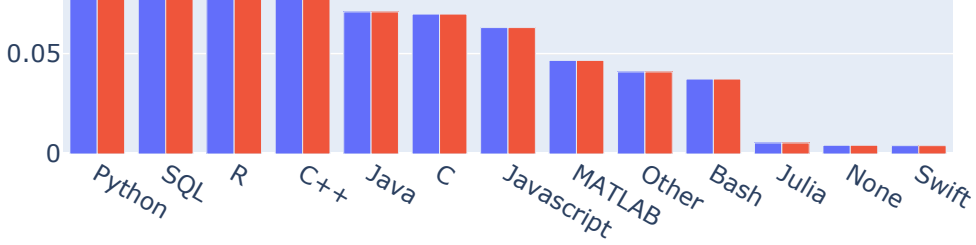
Create more advanced graphs comparing programming languages, IDE's, etc. by role Create a function to easily graph results for other comparisons Separate notebook for comparing gender differences linked here:

In [25]:
```
########################################
# Same Format But Coding Languages Q7
########################################
Questions['Q7']['Roles'] = df_fin.Q5

fig = go.Figure(layout=go.Layout(title= go.layout.Title(text="Comparing Coding Languages
#changed from role selection to selection 1
fig.add_trace(go.Bar(name= 'Selection 1', x= q7.language, y=(q7.Count/ q7.Count.sum())))

def filter_bars(role, data):
    df = data[data['Roles'] == role]
    q7 = df.drop('Roles', axis= 1).count().reset_index()
    q7.columns = ['language','Count']
    return (q7.language, q7.Count/q7.Count.sum())

buttons = []

#added button for all data comparison
buttons.append(dict(method='restyle',
                        label= 'All Samples',
                        visible=True,
                        args=[{'y':[(q7.Count/ q7.Count.sum())],
                               'x':[q7.language],
                               'type':'bar'}, [0]], # the [0] at the end lets us know th
                        )
                   )

for i in list(Roles.keys())[1:]:
    buttons.append(dict(method='restyle',
                        label= i,
                        visible=True,
                        args=[{'y':[filter_bars(i,Questions['Q7'])[1].values],
                               'x':[filter_bars(i,Questions['Q7'])[0].values],
                               'type':'bar'}, [0]], # the [0] at the end lets us know th
                        )
                   )

fig.add_trace(go.Bar(name= 'Selection 2', x= q7.language, y=(q7.Count/ q7.Count.sum())))

buttons2 = []
#added button for all data comparison
buttons2.append(dict(method='restyle',
                        label= 'All Samples',
                        visible=True,
                        args=[{'y':[(q7.Count/ q7.Count.sum())],
                               'x':[q7.language],
                               'type':'bar'}, [1]], # the [0] at the end lets us know th
                        )
                   )

for j in list(Roles.keys())[1:]:
    buttons2.append(dict(method='restyle',
                        label= j,
                        visible=True,
                        args=[{'y':[filter_bars(j,Questions['Q7'])[1].values],
                               'x':[filter_bars(j,Questions['Q7'])[0].values],
                               'type':'bar'}, [1]], # the [1] at the end lets us know th
                        )                          #literally figured that out by just exp
                   )
# adjusted dropdown placement
```

```
#found out updatemenus take a dictionary of buttons and allow you to format how the drop
# https://plotly.com/python/dropdowns/
button_layer_1_height = 1.15
updatemenus = list([
    dict(buttons=buttons,
            direction="down",
            pad={"r": 10, "t": 10},
            showactive=True,
            x=0.1,
            xanchor="left",
            y=button_layer_1_height,
            yanchor="top"),
    dict(buttons=buttons2,
            direction="down",
            pad={"r": 10, "t": 10},
            showactive=True,
            x=0.50,
            xanchor="left",
            y=button_layer_1_height,
            yanchor="top")])

fig.update_layout( updatemenus=updatemenus)
#added annotations next to dropdowns
fig.update_layout(
    annotations=[
        dict(text="Selection 1", x=0, xref="paper", y=1.1, yref="paper",
                                align="left", showarrow=False),
        dict(text="Selection 2", x=0.45, xref="paper", y=1.1,
                                yref="paper", showarrow=False)
    ])
fig.update_xaxes(categoryorder= 'array', categoryarray= q7.language)
fig.show()
```

```
/var/folders/yg/h1y3gfrd6v9_sx5hgb2n7zh40000gn/T/ipykernel_14619/1833184593.py:4: Settin
gWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
```



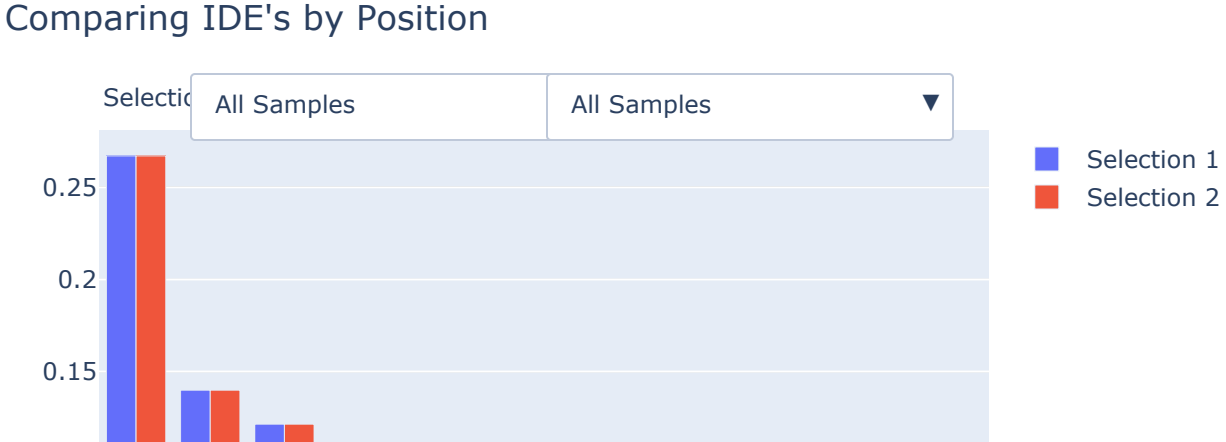


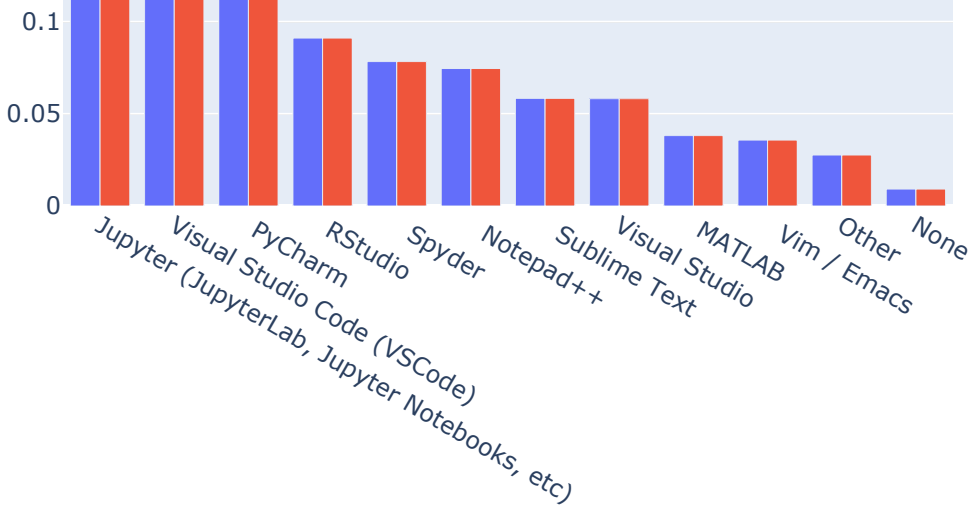
In [26]:

```
########################################
# Same Format But for IDE's Q9
########################################

# Q7 example for go use. We aggregate the data beforehand with .value_counts()
Questions['Q9'].columns = list(Questions['Q9'].mode().iloc[0,:])
q9 = Questions['Q9'].count().reset_index()
q9.columns = ['language','Count']
q9 = q9.sort_values('Count', ascending = False)

Questions['Q9']['Roles'] = df_fin.Q5

fig = go.Figure(layout=go.Layout(title= go.layout.Title(text="Comparing IDE's by Positio
#changed from role selection to selection 1
fig.add_trace(go.Bar(name= 'Selection 1', x= q9.language, y=(q9.Count/ q9.Count.sum())))

buttons = []

#added button for all data comparison
buttons.append(dict(method='restyle',
                        label= 'All Samples',
                        visible=True,
                        args=[{'y':[(q9.Count/ q9.Count.sum())],
                               'x':[q9.language],
                               'type':'bar'}, [0]], # the [0] at the end lets us know th
                        )
                   )

for i in list(Roles.keys())[1:]:
    buttons.append(dict(method='restyle',
                        label= i,
                        visible=True,
                        args=[{'y':[filter_bars(i,Questions['Q9'])[1].values],
                               'x':[filter_bars(i,Questions['Q9'])[0].values],
                               'type':'bar'}, [0]], # the [0] at the end lets us know th
                        )
                   )

fig.add_trace(go.Bar(name= 'Selection 2', x= q9.language, y=(q9.Count/ q9.Count.sum())))

buttons2 = []
#added button for all data comparison
buttons2.append(dict(method='restyle',
                        label= 'All Samples',
                        visible=True,
                        args=[{'y':[(q9.Count/ q9.Count.sum())],
                               'x':[q9.language],
                               'type':'bar'}, [1]], # the [0] at the end lets us know th
                        )
                   )

for j in list(Roles.keys())[1:]:
    buttons2.append(dict(method='restyle',
                        label= j,
```

```
                        visible=True,
                        args=[{'y':[filter_bars(j,Questions['Q9'])[1].values],
                               'x':[filter_bars(j,Questions['Q9'])[0].values],
                               'type':'bar'}, [1]], # the [1] at the end lets us know th
                        )                        #literally figured that out by just exp
                    )
# adjusted dropdown placement
#found out updatemenus take a dictionary of buttons and allow you to format how the drop
# https://plotly.com/python/dropdowns/
button_layer_1_height = 1.15
updatemenus = list([
    dict(buttons=buttons,
            direction="down",
            pad={"r": 10, "t": 10},
            showactive=True,
            x=0.1,
            xanchor="left",
            y=button_layer_1_height,
            yanchor="top"),
    dict(buttons=buttons2,
            direction="down",
            pad={"r": 10, "t": 10},
            showactive=True,
            x=0.50,
            xanchor="left",
            y=button_layer_1_height,
            yanchor="top")])

fig.update_layout( updatemenus=updatemenus)
#added annotations next to dropdowns
fig.update_layout(
    annotations=[
        dict(text="Selection 1", x=0, xref="paper", y=1.1, yref="paper",
                                align="left", showarrow=False),
        dict(text="Selection 2", x=0.45, xref="paper", y=1.1,
                                yref="paper", showarrow=False)
    ])
fig.update_xaxes(categoryorder= 'array', categoryarray= q9.language)
fig.show()
```

```
/var/folders/yg/h1y3gfrd6v9_sx5hgb2n7zh40000gn/T/ipykernel_14619/657301241.py:11: Settin
gWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
```

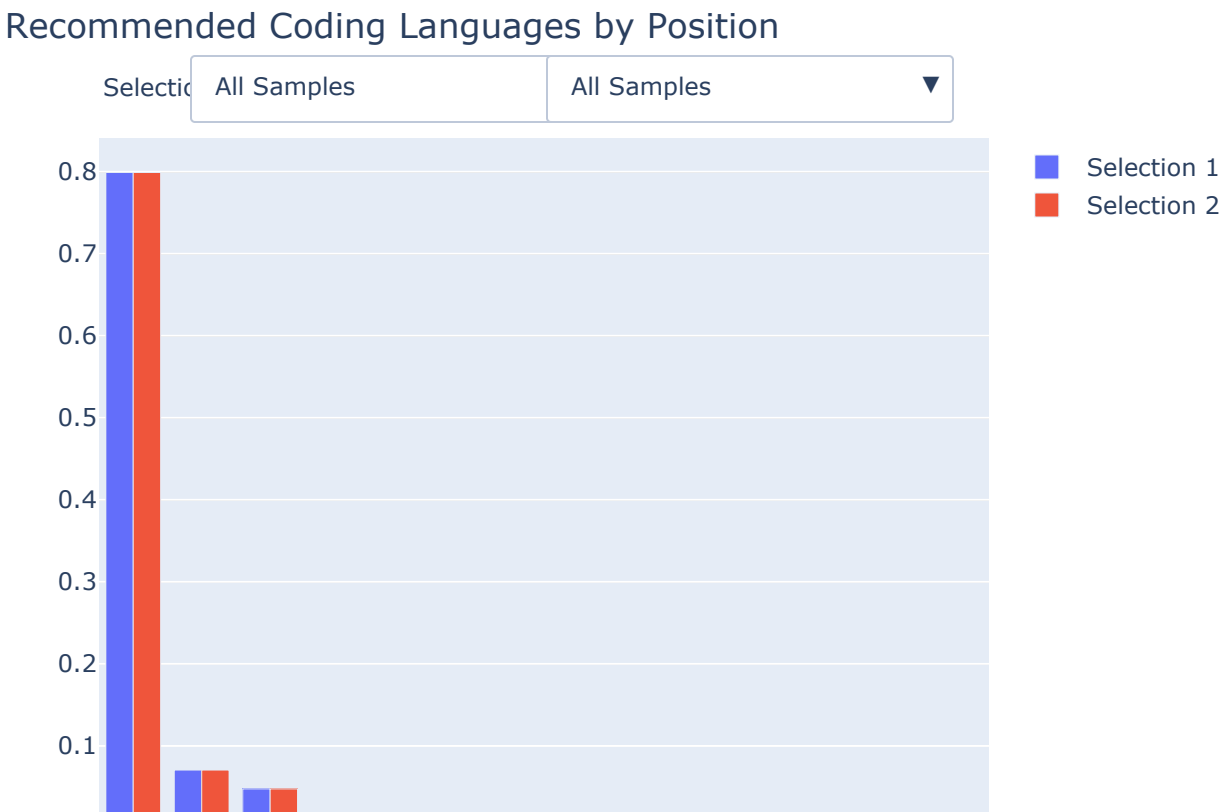




In [27]:

```
########################################
# Question 8 -- What would they recommend
########################################
edu2 = df_fin.Q8.value_counts()
fig = go.Figure(layout=go.Layout(title= go.layout.Title(text="Recommended Coding Languag
#changed from role selection to selection 1
fig.add_trace(go.Bar(name= 'Selection 1', x= edu2.index, y=(edu2.values/ edu2.values.sum

buttons = []

#added button for all data comparison
buttons.append(dict(method='restyle',
                    label= 'All Samples',
                    visible=True,
                    args=[{'y':[df_fin.Q8.value_counts().values/df_fin.Q8.value_coun
                           'x':[df_fin.Q8.value_counts().index],
                           'type':'bar'}, [0]], # the [0] at the end lets us know th
                    )
                )

for i in list(Roles.keys())[1:]:
    buttons.append(dict(method='restyle',
                    label= i,
                    visible=True,
                    args=[{'y':[Roles[i].Q8.value_counts().values/Roles[i].Q8.value_
                           'x':[Roles[i].Q8.value_counts().index],
                           'type':'bar'}, [0]], # the [0] at the end lets us know th
                    )
                )

fig.add_trace(go.Bar(name= 'Selection 2',x= edu2.index, y=(edu2.values/ edu2.values.sum(

buttons2 = []
#added button for all data comparison
buttons2.append(dict(method='restyle',
                    label= 'All Samples',
                    visible=True,
                    args=[{'y':[df_fin.Q8.value_counts().values/df_fin.Q8.value_coun
                           'x':[df_fin.Q8.value_counts().index],
                           'type':'bar'}, [1]], # the [0] at the end lets us know th
                    )
                )

for i in list(Roles.keys())[1:]:
    buttons2.append(dict(method='restyle',
                    label= i,
                    visible=True,
```

```
                        args=[{'y':[Roles[i].Q8.value_counts().values/Roles[i].Q8.value_
                               'x':[Roles[i].Q8.value_counts().index],
                               'type':'bar'}, [1]], # the [1] at the end lets us know th
                        )                          #literally figured that out by just exp
                    )
# adjusted dropdown placement
#found out updatemenus take a dictionary of buttons and allow you to format how the drop
# https://plotly.com/python/dropdowns/
button_layer_1_height = 1.15
updatemenus = list([
    dict(buttons=buttons,
            direction="down",
            pad={"r": 10, "t": 10},
            showactive=True,
            x=0.1,
            xanchor="left",
            y=button_layer_1_height,
            yanchor="top"),
    dict(buttons=buttons2,
            direction="down",
            pad={"r": 10, "t": 10},
            showactive=True,
            x=0.50,
            xanchor="left",
            y=button_layer_1_height,
            yanchor="top")])

fig.update_layout( updatemenus=updatemenus)
#added annotations next to dropdowns
fig.update_layout(
    annotations=[
        dict(text="Selection 1", x=0, xref="paper", y=1.1, yref="paper",
                                align="left", showarrow=False),
        dict(text="Selection 2", x=0.45, xref="paper", y=1.1,
                                yref="paper", showarrow=False)
    ])
#fig.update_xaxes(categoryorder= 'array', categoryarray= ["Doctoral degree",'Master's de
fig.show()
```



In [28]:

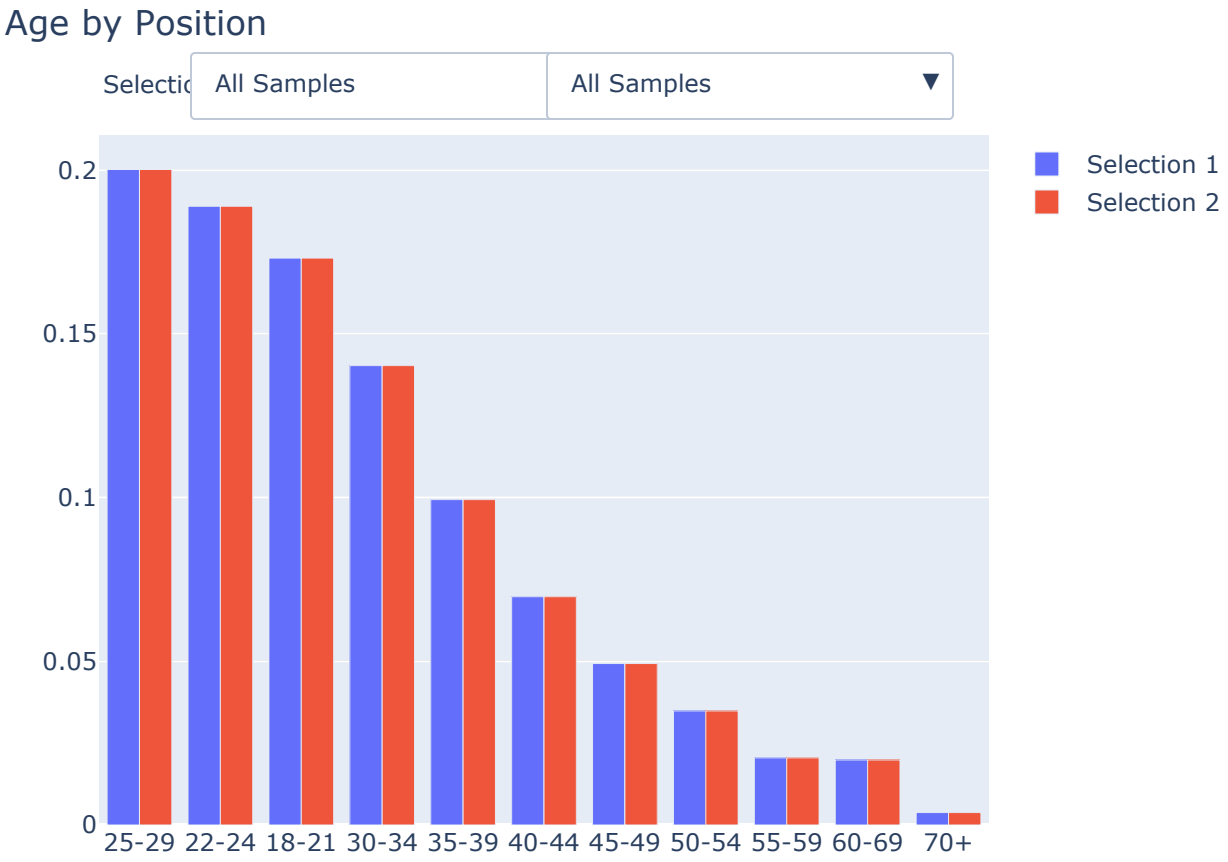
```
########################################
# Design Function
########################################

def filter_bars(role, data):
    df = data[data['Roles'] == role]
    q = df.drop('Roles', axis= 1).count().reset_index()
    q.columns = ['language','Count']
    return (q.language, q.Count/q.Count.sum())

def build_graph(q_number, Roles, Title):
    """Create dropdown visual with question data"""
    if isinstance(q_number, pd.DataFrame):
        qnumber = q_number.copy()
        qnumber.columns = list(qnumber.mode().iloc[0,:])
        qcnt = qnumber.count().reset_index()
        qcnt.columns = ['feature','cnt']
        qcnt = qcnt.sort_values('cnt', ascending = False)
        qnumber['Roles'] = df_fin.Q5


        fig = go.Figure(layout=go.Layout(title= go.layout.Title(text=Title)))
        #changed from role selection to selection 1
        fig.add_trace(go.Bar(name= 'Selection 1', x= qcnt.feature, y=(qcnt.cnt/ qcnt.cnt

        buttons = []

        #added button for all data comparison
        buttons.append(dict(method='restyle',
                            label= 'All Samples',
                            visible=True,
                            args=[{'y':[(qcnt.cnt/ qcnt.cnt.sum())],
                                   'x':[qcnt.feature],
                                   'type':'bar'}, [0]], # the [0] at the end lets us
                            )
                       )

        for i in list(Roles.keys())[1:]:
            buttons.append(dict(method='restyle',
                                label= i,
                                visible=True,
                                args=[{'y':[filter_bars(i,qnumber)[1].values],
                                       'x':[filter_bars(i,qnumber)[0].values],
                                       'type':'bar'}, [0]], # the [0] at the end lets us
                                )
                           )

        fig.add_trace(go.Bar(name= 'Selection 2', x= qcnt.feature, y=(qcnt.cnt/ qcnt.cnt

        buttons2 = []
        #added button for all data comparison
        buttons2.append(dict(method='restyle',
                            label= 'All Samples',
                            visible=True,
                            args=[{'y':[(qcnt.cnt/ qcnt.cnt.sum())],
                                   'x':[qcnt.feature],
                                   'type':'bar'}, [1]],
                            )
                       )
```

```
        for i in list(Roles.keys())[1:]:
            buttons2.append(dict(method='restyle',
                                 label= i,
                                 visible=True,
                                 args=[{'y':[filter_bars(i,qnumber)[1].values],
                                        'x':[filter_bars(i,qnumber)[0].values],
                                        'type':'bar'}, [1]],
                                 )
                            )

        # adjusted dropdown placement
        #found out updatemenus take a dictionary of buttons and allow you to format how
        # https://plotly.com/python/dropdowns/
        button_layer_1_height = 1.15
        updatemenus = list([
            dict(buttons=buttons,
                    direction="down",
                    pad={"r": 10, "t": 10},
                    showactive=True,
                    x=0.1,
                    xanchor="left",
                    y=button_layer_1_height,
                    yanchor="top"),
            dict(buttons=buttons2,
                    direction="down",
                    pad={"r": 10, "t": 10},
                    showactive=True,
                    x=0.50,
                    xanchor="left",
                    y=button_layer_1_height,
                    yanchor="top")])

        fig.update_layout( updatemenus=updatemenus)
        #added annotations next to dropdowns
        fig.update_layout(
            annotations=[
                dict(text="Selection 1", x=0, xref="paper", y=1.1, yref="paper",
                                        align="left", showarrow=False),
                dict(text="Selection 2", x=0.45, xref="paper", y=1.1,
                                        yref="paper", showarrow=False)
            ])
        fig.update_xaxes(categoryorder= 'array', categoryarray= qcnt.feature)
        fig.show()


    else:
        qnumber= q_number.copy()
        vcnts = qnumber.value_counts()
        qnumber = pd.concat([qnumber,df_fin.Q5], axis =1)
        qnumber.columns = ['feature','Roles']

        fig = go.Figure(layout=go.Layout(title= go.layout.Title(text=Title)))
        #changed from role selection to selection 1
        fig.add_trace(go.Bar(name= 'Selection 1', x= vcnts.index, y=(vcnts.values/ vcnts

        buttons = []

        #added button for all data comparison
        buttons.append(dict(method='restyle',
                                 label= 'All Samples',
                                 visible=True,
                                 args=[{'y':[vcnts.values/ vcnts.values.sum()],
                                        'x':[vcnts.index],
                                        'type':'bar'}, [0]], # the [0] at the end lets us
                                 )
```

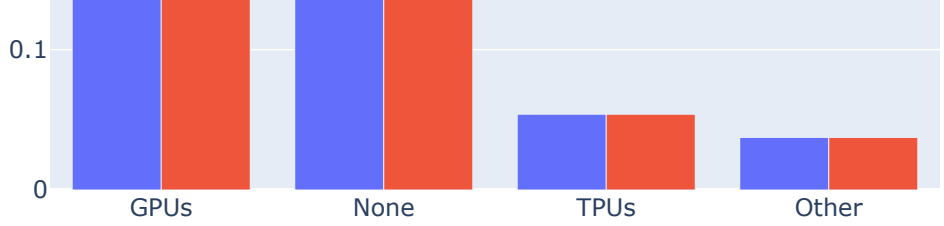
```
                    )

for i in list(Roles.keys())[1:]:
    qrole = qnumber[qnumber['Roles']==i].feature.value_counts()
    buttons.append(dict(method='restyle',
                        label= i,
                        visible=True,
                        args=[{'y':[qrole.values/qrole.values.sum()],
                               'x':[qrole.index],
                               'type':'bar'}, [0]], # the [0] at the end lets us
                        )
                    )

fig.add_trace(go.Bar(name= 'Selection 2',x= vcnts.index, y=(vcnts.values/ vcnts.

buttons2 = []
        #added button for all data comparison
buttons2.append(dict(method='restyle',
                        label= 'All Samples',
                        visible=True,
                        args=[{'y':[(vcnts.values/ vcnts.values.sum())],
                               'x':[vcnts.index],
                               'type':'bar'}, [1]], # the [0] at the end lets us
                        )
                    )

for i in list(Roles.keys())[1:]:
    qrole = qnumber[qnumber['Roles']==i].feature.value_counts()
    buttons2.append(dict(method='restyle',
                        label= i,
                        visible=True,
                        args=[{'y':[qrole.values/qrole.values.sum()],
                               'x':[qrole.index],
                               'type':'bar'}, [1]], # the [0] at the end lets us
                        )
                    )
# adjusted dropdown placement
#found out updatemenus take a dictionary of buttons and allow you to format how
# https://plotly.com/python/dropdowns/
button_layer_1_height = 1.15
updatemenus = list([
    dict(buttons=buttons,
            direction="down",
            pad={"r": 10, "t": 10},
            showactive=True,
            x=0.1,
            xanchor="left",
            y=button_layer_1_height,
            yanchor="top"),
    dict(buttons=buttons2,
            direction="down",
            pad={"r": 10, "t": 10},
            showactive=True,
            x=0.50,
            xanchor="left",
            y=button_layer_1_height,
            yanchor="top")])

fig.update_layout( updatemenus=updatemenus)
#added annotations next to dropdowns
fig.update_layout(
    annotations=[
        dict(text="Selection 1", x=0, xref="paper", y=1.1, yref="paper",
                                align="left", showarrow=False),
        dict(text="Selection 2", x=0.45, xref="paper", y=1.1,
                                yref="paper", showarrow=False)
```

```
            ])
        fig.update_xaxes(categoryorder= 'array', categoryarray= vcnts.index)
        fig.show()

    return
```

In [29]:
```
build_graph(Questions['Q1'],Roles,'Age by Position')
```



In [30]:
```
build_graph(Questions['Q12'],Roles,'Hardware by position')
```



In [ ]: