In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [2]:

```
df=pd.read_csv("C:/Users/sathi/Downloads/DataSet/data.csv",low_memory=False)
df=df[1:] #Remove first row as i contains longer text
df.head()
```

Out[2]:

	Time from Start to Finish (seconds)	Q1	Q2	Q3	Q4	Q5	Q6	Q7_Part_1	Q7_Part_2	Q7_Part_3	...	Q38_B_Part_3	Q38_B_P
1	910	50-54	Man	India	Bachelor’s degree	Other	5-10 years	Python	R	NaN	...	NaN	
2	784	50-54	Man	Indonesia	Master’s degree	Program/Project Manager	20+ years	NaN	NaN	SQL	...	NaN	
3	924	22-24	Man	Pakistan	Master’s degree	Software Engineer	1-3 years	Python	NaN	NaN	...	NaN	
4	575	45-49	Man	Mexico	Doctoral degree	Research Scientist	20+ years	Python	NaN	NaN	...	NaN	
5	781	45-49	Man	India	Doctoral degree	Other	< 1 years	Python	NaN	NaN	...	NaN	

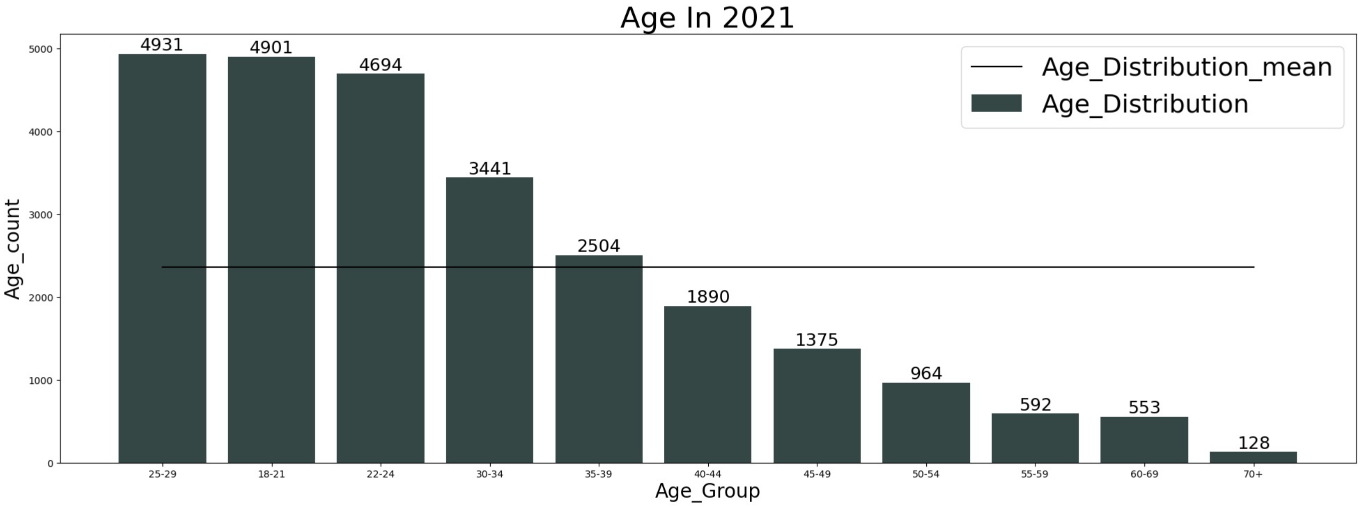
5 rows × 369 columns

In [3]:

```
def country_cleaning(x):
    '''This Function truncated the longer countries' names to the short names..
    '''
    if x=='United States of America':
        x='USA'
    elif x=='United Kingdom of Great Britain and Northern Ireland':
        x='UK'
    elif x=='Iran, Islamic Republic of...':
        x='Iran'
    elif x=='Hong Kong (S.A.R.)':
        x='Hong Kong'
    elif x=='I do not wish to disclose my location':
        x='Other'
    elif x=='United Arab Emirates':
        x='UA'
    elif x=='Viet Nam':
        x='Vietnam'
    return x

def degree_cleaning(x):
    '''This Function truncated the longer degree' names to the short names..
    '''
    if x=='Some college/university study without earning a bachelor’s degree':
        x='College without degree'
    elif x=='I prefer not to answer':
        x='No-response'
    elif x=='No formal education past high school':
        x='After high school'
    return x
def code_cleaning(x):
    if x=='I have never written code':
        x='0years'
    elif x=='I do not use machine learning methods':
        x='No experience'
    return x
#we use map function here to process the countries names
df['Q3']=df['Q3'].map(lambda x: country_cleaning(x))
df['Q4']=df['Q4'].map(lambda x: degree_cleaning(x))
df['Q6']=df['Q6'].map(lambda x: code_cleaning(x))
df['Q15']=df['Q15'].map(lambda x: code_cleaning(x))
#########################################################
```

In [4]:

```
#Define the figure size
fig,ax1=plt.subplots(1,1,figsize=(24,8))
#drwaing barplot
sns.barplot(x=df['Q1'].value_counts().index,y=df['Q1'].value_counts().values,color="#314a48",label="Age_Distribu
#Mentioning the text in bar plot
for index,value in enumerate(df['Q1'].value_counts().values):
    ax1.annotate(value,xy=(index,value+100),ha="center",va="center",fontsize=18)
#ploting the mean line
sns.lineplot(x=df['Q1'].value_counts().index,y=df['Q1'].value_counts().values.mean(),color="black",label="Age_Di
plt.legend(fontsize=26)
plt.xlabel("Age_Group",fontsize=20)
plt.ylabel("Age_count",fontsize=20)
plt.title("Age In 2021",fontsize=30)
plt.show()
```



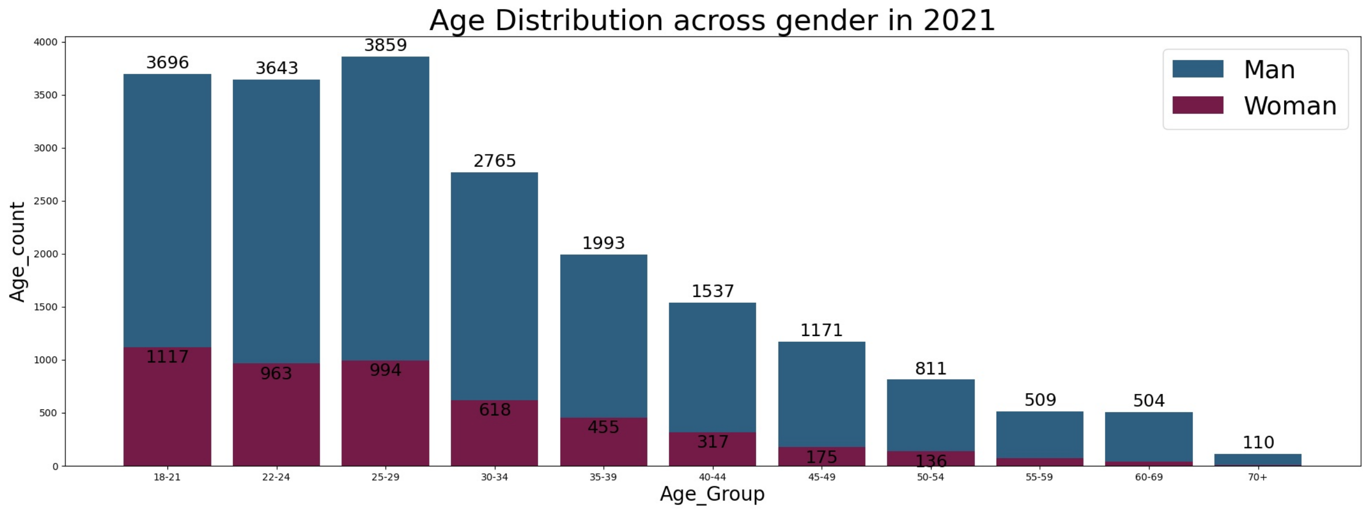
In [5]:

```
#create empty dict to store the population regarding age group of man and woman
gender_dict={}
#loop through gender column
for gender in df['Q2'].value_counts().index:
    gender_dict[gender]=df[df['Q2']==gender]['Q1'].value_counts()
#create dataframe from dictionary
age_df=pd.DataFrame(gender_dict)
#change the index name
age_df.index.rename("Age_Group",inplace=True)
#display the datafarme
display(age_df)
```

	Man	Woman	Prefer not to say	Nonbinary	Prefer to self-describe
Age_Group					
18-21	3696	1117	60	16	12.0
22-24	3643	963	66	13	9.0
25-29	3859	994	61	12	5.0
30-34	2765	618	34	17	7.0
35-39	1993	455	42	7	7.0
40-44	1537	317	31	4	1.0
45-49	1171	175	24	4	1.0
50-54	811	136	14	3	NaN
55-59	509	72	7	4	NaN
60-69	504	35	10	4	NaN
70+	110	8	6	4	NaN

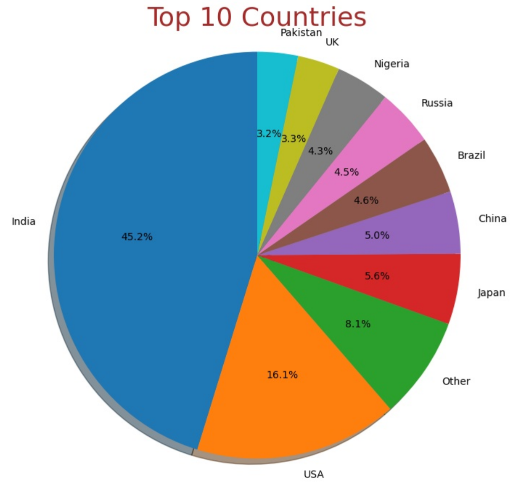
In [6]:

```
#Define the figure size
fig,ax1=plt.subplots(1,1,figsize=(24,8))
#Plot the bar chart in same axis
ax1.bar(age_df.index,age_df['Man'],color="#2e5f81",label="Man")
ax1.bar(age_df.index,age_df['Woman'],color="#741b47",label="Woman")
#Adjust the text in bar chart
for index in age_df.index:
    ax1.annotate(age_df["Man"].loc[index],xy=(index,age_df["Man"].loc[index]+100),ha="center",va="center",fontsi
    ax1.annotate(age_df["Woman"].loc[index],xy=(index,age_df["Woman"].loc[index]-100),ha="center",va="center",fo
plt.legend(fontsize=26)
plt.xlabel("Age_Group",fontsize=20)
plt.ylabel("Age_count",fontsize=20)
plt.title("Age Distribution across gender in 2021",fontsize=30)
plt.show()
```



In [7]:

```
#df['Q3'].value_counts()
df_=df['Q3'].value_counts().head(10)
#Draw the pie chart
labels=df_.index
sizes=df_.values
fig,ax1=plt.subplots(1,1,figsize=(24,8))
ax1.pie(sizes, labels=labels, autopct='%1.1f%%',shadow=True, startangle=90)
ax1.axis("equal")
plt.title("Top 10 Countries",fontsize=26,color="#a52a2a")
plt.show()
```



In [8]:

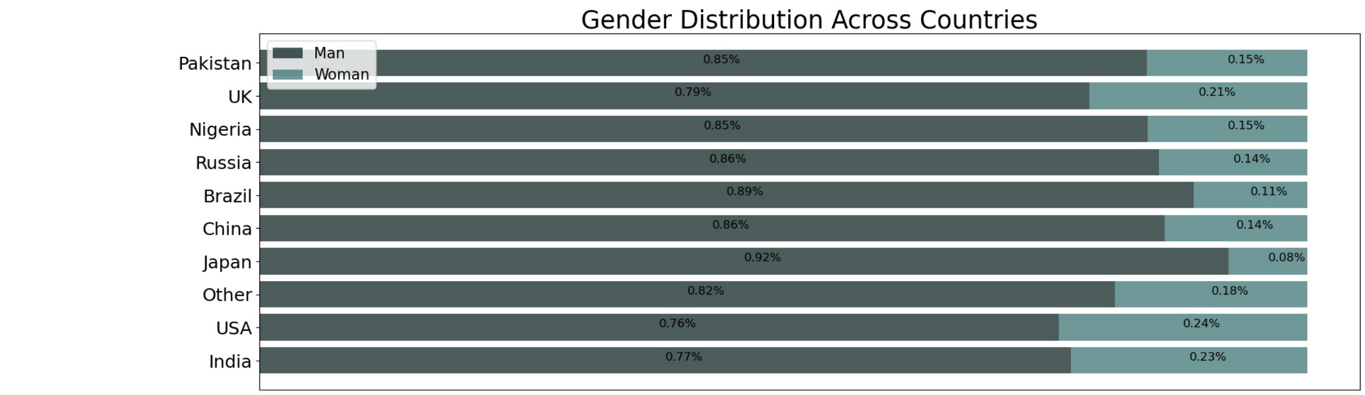
```
def gender_across_columns(col_name):
    '''This function returns the ratio of men and women participation in data science for any column in the data
    By using this function we can plot a bar chart and can visualize the distribution of men and women.
    '''
    #create the dictionary
    pop={}
    #iterate through the columns
    for index in df[col_name].value_counts().index:
        pop[index]=df[df[col_name]==index]['Q2'].value_counts()
    _df=pd.DataFrame(pop)
    #create dataframe which will calculate ratio of men and women
    new_df=_df.T
    #select only men and women from 5 categories
    new_df=new_df[['Man','Woman']]
    #find the sum=men+women
    new_df['sum']=new_df.sum(axis=1)
    #calculate the ratio
    new_df=new_df.T/new_df['sum']
    #remove the sum row from dataframe
    new_df=new_df[:-1]
    #adjust the dataframe
    new_df_ratio=new_df.T
    return new_df_ratio
```

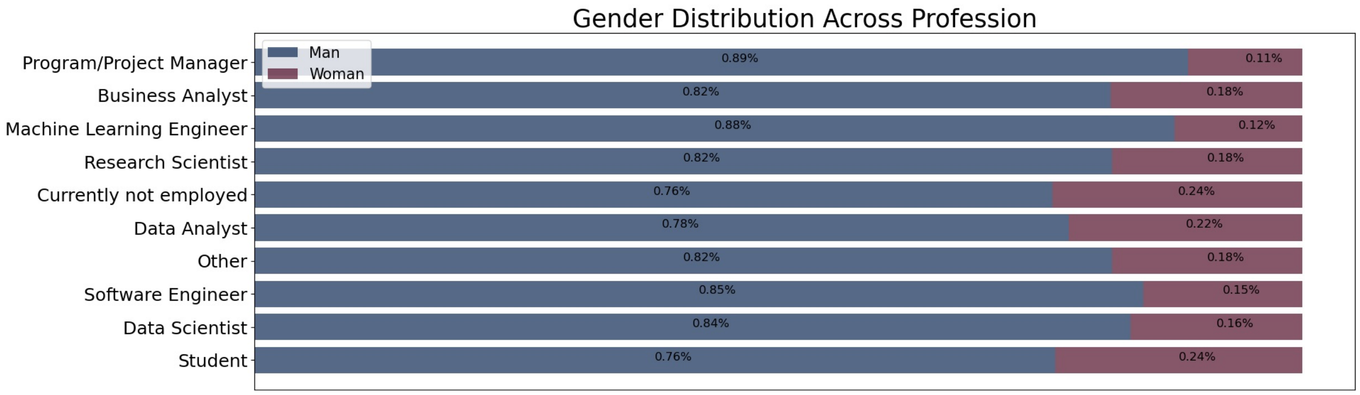
In [9]:

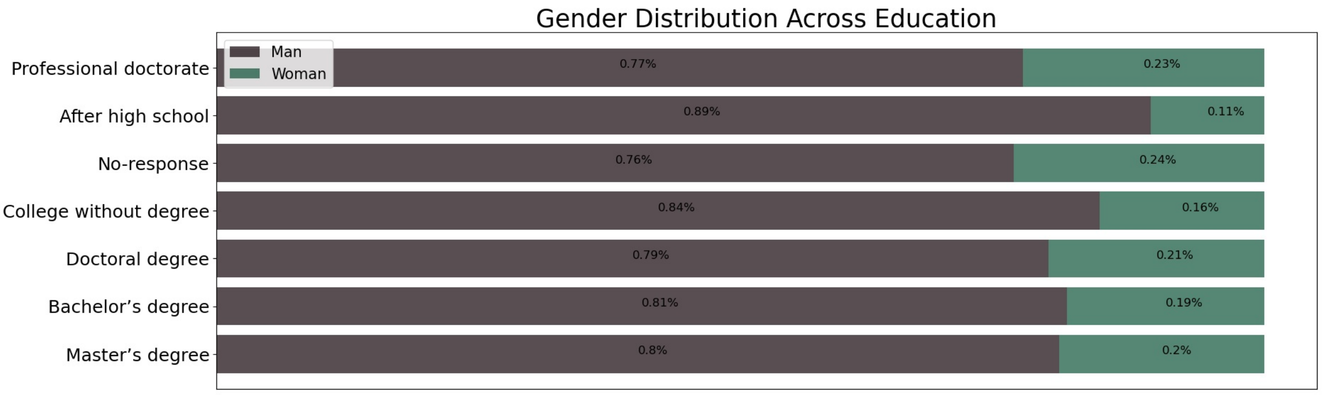
```
df3=gender_across_columns('Q3').head(10)
df4=gender_across_columns('Q4')
df5=gender_across_columns('Q5').head(10)
df6=gender_across_columns('Q6')


fig, ax1 = plt.subplots(4,figsize=(20,30))
ax1[0].barh(df3.index,df3['Man'],alpha=0.7,label='Man',color='#001817')
```

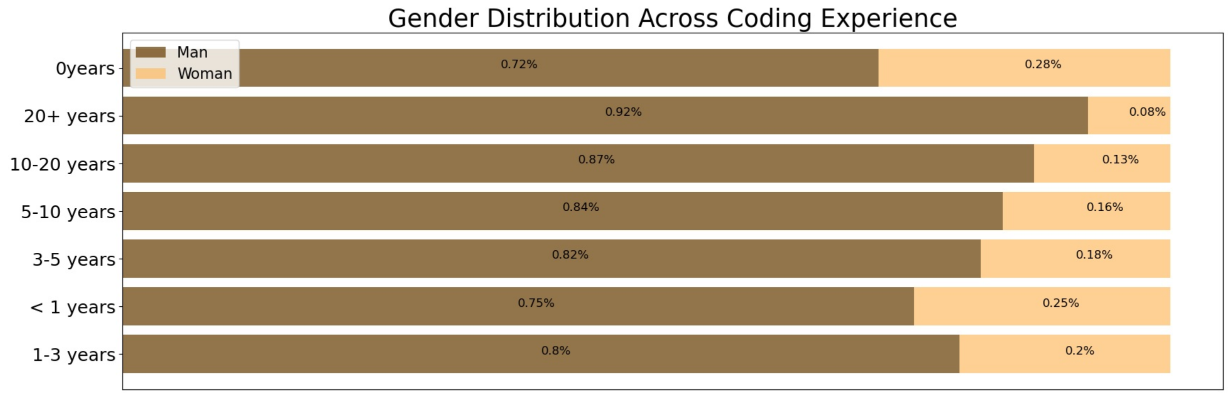
```
ax1[0].barh(df3.index,df3['Woman'],alpha=0.7,label='Woman',color='#326e6d',left=df3['Man'])

for index in df3.index:
    ax1[0].annotate(str(round(df3['Man'].loc[index],2))+'%',xy=(df3['Man'].loc[index]/2,index),fontsize=12,colo
    ax1[0].annotate(str(round(df3['Woman'].loc[index],2))+'%',xy=(df3['Man'].loc[index]+df3['Woman'].loc[index],
ax1[0].set_xticks([])
ax1[0].set_yticklabels(df3.index,fontsize=18)
ax1[0].set_title("Gender Distribution Across Countries",fontsize=25)
ax1[0].legend(loc ='upper left',fontsize=15)

#####################################################################
ax1[1].barh(df5.index,df5['Man'],alpha=0.7,label='Man',color='#0f2953')
ax1[1].barh(df5.index,df5['Woman'],alpha=0.7,label='Woman',color='#530f29',left=df5['Man'])

for index in df5.index:
    ax1[1].annotate(str(round(df5['Man'].loc[index],2))+'%',xy=(df5['Man'].loc[index]/2,index),fontsize=12,colo
    ax1[1].annotate(str(round(df5['Woman'].loc[index],2))+'%',xy=(df5['Man'].loc[index]+df5['Woman'].loc[index],
ax1[1].set_xticks([])
ax1[1].set_yticklabels(df5.index,fontsize=18)
ax1[1].set_title("Gender Distribution Across Profession",fontsize=25)
ax1[1].legend(loc ='upper left',fontsize=15)
########################################################################################
ax1[2].barh(df4.index,df4['Man'],alpha=0.7,label='Man',color='#120309')
ax1[2].barh(df4.index,df4['Woman'],alpha=0.7,label='Woman',color='#0f5339',left=df4['Man'])

for index in df4.index:
    ax1[2].annotate(str(round(df4['Man'].loc[index],2))+'%',xy=(df4['Man'].loc[index]/2,index),fontsize=12,colo
    ax1[2].annotate(str(round(df4['Woman'].loc[index],2))+'%',xy=(df4['Man'].loc[index]+df4['Woman'].loc[index],
ax1[2].set_xticks([])
ax1[2].set_yticklabels(df4.index,fontsize=18)
ax1[2].set_title("Gender Distribution Across Education",fontsize=25)
ax1[2].legend(loc ='upper left',fontsize=15)
##########################################################

ax1[3].barh(df6.index,df6['Man'],alpha=0.7,label='Man',color='#663a00')
ax1[3].barh(df6.index,df6['Woman'],alpha=0.7,label='Woman',color='#ffbd66',left=df6['Man'])

for index in df6.index:
    ax1[3].annotate(str(round(df6['Man'].loc[index],2))+'%',xy=(df6['Man'].loc[index]/2,index),fontsize=12,colo
    ax1[3].annotate(str(round(df6['Woman'].loc[index],2))+'%',xy=(df6['Man'].loc[index]+df6['Woman'].loc[index],
ax1[3].set_xticks([])
ax1[3].set_yticklabels(df6.index,fontsize=18)
ax1[3].set_title("Gender Distribution Across Coding Experience",fontsize=25)
ax1[3].legend(loc ='upper left',fontsize=15)

plt.show()
```

```
C:\Users\sathi\AppData\Local\Temp\ipykernel_21940\1864963147.py:15: UserWarning: set_ticklabels() should only be
used with a fixed number of ticks, i.e. after set_ticks() or using a FixedLocator.
  ax1[0].set_yticklabels(df3.index,fontsize=18)
C:\Users\sathi\AppData\Local\Temp\ipykernel_21940\1864963147.py:27: UserWarning: set_ticklabels() should only be
used with a fixed number of ticks, i.e. after set_ticks() or using a FixedLocator.
  ax1[1].set_yticklabels(df5.index,fontsize=18)
C:\Users\sathi\AppData\Local\Temp\ipykernel_21940\1864963147.py:38: UserWarning: set_ticklabels() should only be
used with a fixed number of ticks, i.e. after set_ticks() or using a FixedLocator.
  ax1[2].set_yticklabels(df4.index,fontsize=18)
C:\Users\sathi\AppData\Local\Temp\ipykernel_21940\1864963147.py:50: UserWarning: set_ticklabels() should only be
used with a fixed number of ticks, i.e. after set_ticks() or using a FixedLocator.
  ax1[3].set_yticklabels(df6.index,fontsize=18)
```









```
In [ ]:
```

```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js