In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:
```
df=pd.read_csv("data30.csv")
df.head()
```

Out[ ]:

	Unnamed: 0	Age	Accessibility	EdLevel	Employment	Gender	MentalHealth	MainBranch
0	0	<35	No	Master	1	Man	No	Dev
1	1	<35	No	Undergraduate	1	Man	No	Dev
2	2	<35	No	Master	1	Man	No	Dev
3	3	<35	No	Undergraduate	1	Man	No	Dev
4	4	>35	No	PhD	0	Man	No	NotDev

Data Preprocessing Part 1

In [ ]:
```
#Remove "Unnamed: 0" column
df=df.drop("Unnamed: 0",axis=1)
df.head()
```

Out[ ]:

	Age	Accessibility	EdLevel	Employment	Gender	MentalHealth	MainBranch	YearsCode
0	<35	No	Master	1	Man	No	Dev	7
1	<35	No	Undergraduate	1	Man	No	Dev	12
2	<35	No	Master	1	Man	No	Dev	15
3	<35	No	Undergraduate	1	Man	No	Dev	9
4	>35	No	PhD	0	Man	No	NotDev	40

In [ ]:
```
df.shape
```

Out[ ]:
```
(73462, 14)
```

In [ ]:
```
df.info()
```

Out[ ]:
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73462 entries, 0 to 73461
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Age             73462 non-null  object
 1   Accessibility   73462 non-null  object
 2   EdLevel         73462 non-null  object
 3   Employment      73462 non-null  int64 
 4   Gender          73462 non-null  object
 5   MentalHealth    73462 non-null  object
 6   MainBranch      73462 non-null  object
 7   YearsCode       73462 non-null  int64 
 8   YearsCodePro    73462 non-null  int64 
 9   Country         73462 non-null  object
 10  PreviousSalary  73462 non-null  int64 
 11  HaveWorkedWith  73399 non-null  object
 12  ComputerSkills  73462 non-null  int64 
 13  Employed        73462 non-null  int64 
dtypes: int64(6), object(8)
memory usage: 7.8+ MB
```

In [ ]:
```
#Check the number of unique value from all the object datatype
df.select_dtypes("object").nunique()
```

Out[ ]:
```
Age                 2
Accessibility       2
EdLevel             5
Gender              3
MentalHealth        2
MainBranch          2
Country           172
HaveWorkedWith    69980
dtype: int64
```

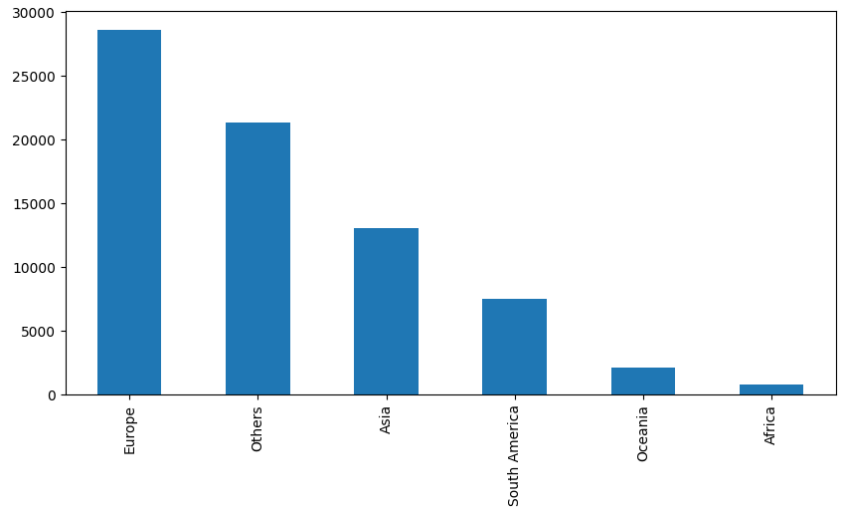
In [ ]:
```
#Segment country into smaller unique values
df.Country.unique()
```

Out[ ]:
```
array(['Sweden', 'Spain', 'Germany', 'Canada', 'Singapore', 'France',
       'Switzerland',
       'United Kingdom of Great Britain and Northern Ireland',
       'Russian Federation', 'Israel', 'Turkey',
       'United States of America', 'Brazil', 'Bulgaria', 'Greece',
       'Italy', 'Netherlands', 'Poland', 'Hungary', 'Pakistan', 'Nigeria',
       'Albania', 'Bangladesh', 'Viet Nam', 'Romania', 'Sri Lanka',
       'India', 'Lithuania', 'Ukraine', 'Croatia', 'Georgia', 'Denmark',
       'Ireland', 'Lebanon', 'Bahrain', 'Egypt', 'Colombia', 'Australia',
       'Chile', 'Indonesia', 'Iran, Islamic Republic of...', 'Portugal',
       'Slovakia', 'Armenia', 'Finland', 'Hong Kong (S.A.R.)',
       'Argentina', 'Costa Rica', 'Peru', 'Japan', 'Belgium',
       'United Arab Emirates', 'Bolivia', 'Austria', 'South Africa',
       'Norway', 'Serbia', 'Malta', 'Malaysia', 'Czech Republic',
       'Belarus', 'Madagascar', 'Kenya', 'Slovenia', 'Uruguay',
       'The former Yugoslav Republic of Macedonia', 'Botswana', 'Algeria',
       'China', 'Mexico', 'Cyprus',
       'Venezuela, Bolivarian Republic of...', 'Jordan',
       'Dominican Republic', 'Ecuador', 'Luxembourg', 'Uzbekistan',
       'Syrian Arab Republic', 'Zambia', 'Taiwan', 'Tunisia',
       'South Korea', 'Paraguay', 'Iceland', 'Morocco', 'Guatemala',
       'Cameroon', 'Republic of Moldova', 'Nepal', 'Kazakhstan',
       'Estonia', 'Ethiopia', 'Cuba', 'Latvia',
       "Lao People's Democratic Republic", 'Yemen',
       'Democratic Republic of the Congo', 'Somalia', 'Philippines',
       'Azerbaijan', 'Saudi Arabia', 'Honduras', 'Angola', 'Thailand',
       'Bosnia and Herzegovina', 'United Republic of Tanzania',
       'El Salvador', 'Iraq', 'Kosovo', 'Andorra', 'New Zealand',
       'Dominica', 'Nicaragua', 'Mozambique', 'Trinidad and Tobago',
       'Nomadic', 'Panama', 'Barbados', 'Mauritius', 'Kuwait', 'Ghana',
       'Congo, Republic of the...', 'Myanmar', 'Republic of Korea',
       'Cambodia', 'Kyrgyzstan', 'Afghanistan', 'Uganda', 'Swaziland',
       'Saint Kitts and Nevis', 'Rwanda', 'Monaco', 'Turkmenistan',
       'Libyan Arab Jamahiriya', 'Sudan', 'Montenegro', 'Haiti', 'Belize',
       'Benin', 'Bhutan', 'Palestine', 'Tajikistan', 'Isle of Man',
       'Cape Verde', 'Oman', "Côte d'Ivoire", 'Zimbabwe', 'Jamaica',
       'Guyana', 'Namibia', 'Togo', 'Lesotho', 'Guinea', 'Senegal',
       'Mongolia', 'Liberia', 'Suriname', 'Niger', 'Maldives', 'Qatar',
       'Saint Lucia', 'Djibouti', 'Malawi',
       'Saint Vincent and the Grenadines', 'Gambia', 'Burkina Faso',
       'Fiji', 'Mauritania', 'Burundi', 'Timor-Leste', 'Mali',
       'Seychelles'], dtype=object)
```

In [ ]:
```
#define a function to segment countries
def segment_country(country):
  if country in ['Sweden', 'Spain', 'Germany', 'Switzerland', 'United Kingdom of Gr
    return "Europe"
  elif country in  ['Canada', 'Brazil', 'Argentina', 'Chile', 'Peru', 'Bolivia', '(
    return "South America"
  elif country in ['Singapore', 'Israel', 'Turkey', 'Russian Federation', 'India',
    return "Asia"
  elif country in  ['Australia', 'New Zealand', 'Fiji', 'Papua New Guinea', 'Solomo
    return "Oceania"
  elif country in ['Nigeria', 'Madagascar', 'Kenya', 'Algeria', 'Morocco', 'Camerod
    return "Africa"
  else:
    return "Others"


df["Region"]=df["Country"].apply(segment_country)
```

In [ ]:
```
plt.figure(figsize=(10,5))
df["Region"].value_counts().plot(kind="bar")
```

Out[ ]: <Axes: >



In [ ]:
```
#Removed HaveWorkedWith column
df=df.drop("HaveWorkedWith",axis=1)
```

In [ ]:
```
#Replace 1=Yes and 0=No
df["Employment"]=df["Employment"].replace({0:"No",1:"Yes"})
df["Employed"]=df["Employed"].replace({0:"No",1:"Yes"})
```

In [ ]:
```
df.tail()
```

Out[ ]:

	Age	Accessibility	EdLevel	Employment	Gender	MentalHealth	MainBranch	Yea
73457	<35	No	Undergraduate	Yes	Man	No	Dev	
73458	>35	No	Undergraduate	Yes	Man	No	Dev	
73459	<35	No	Undergraduate	Yes	Man	No	Dev	
73460	<35	Yes	Undergraduate	Yes	Man	Yes	Dev	
73461	<35	No	Master	Yes	NonBinary	No	Dev	

In [ ]:

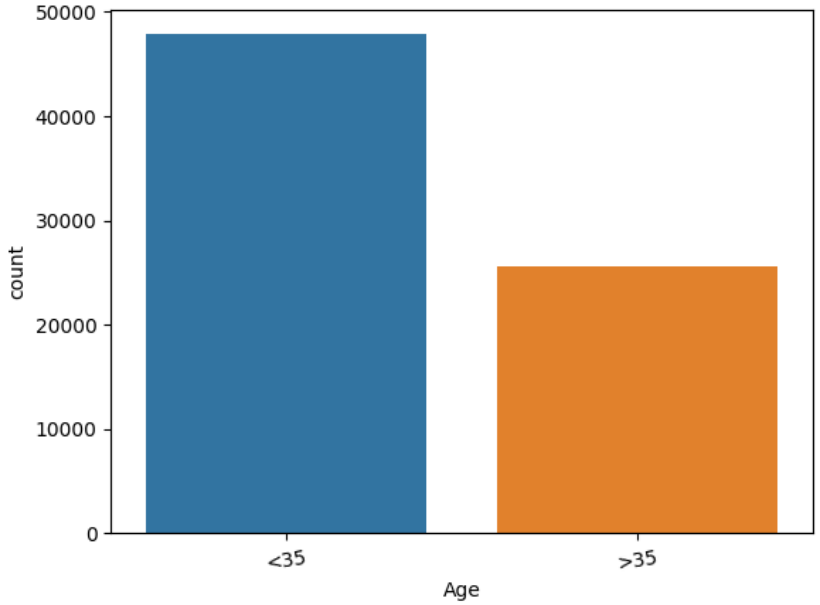
Exploratory Data Analysis

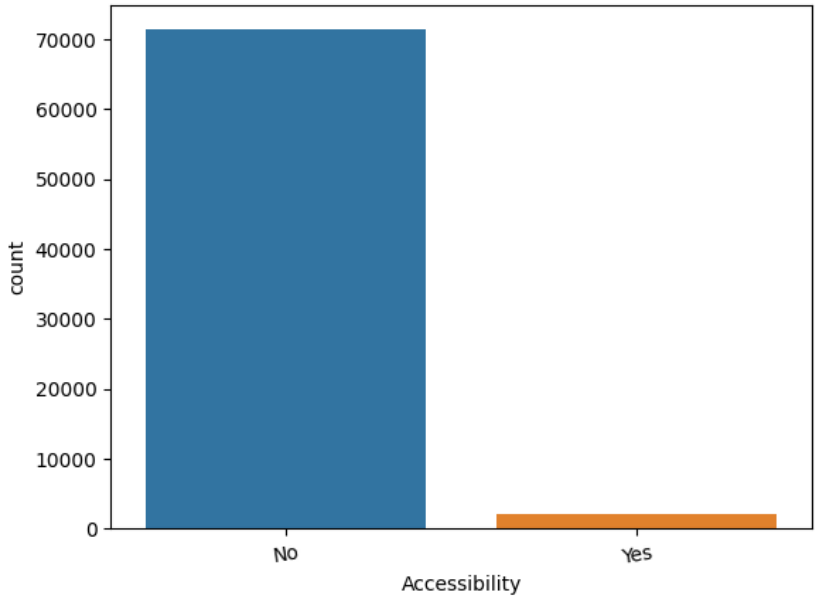
In [ ]:
```
# Get the names of all the columns with data type "Object" (categorical column)
categorical=[x for x in df.select_dtypes(include="object").columns if x !="Country"
```

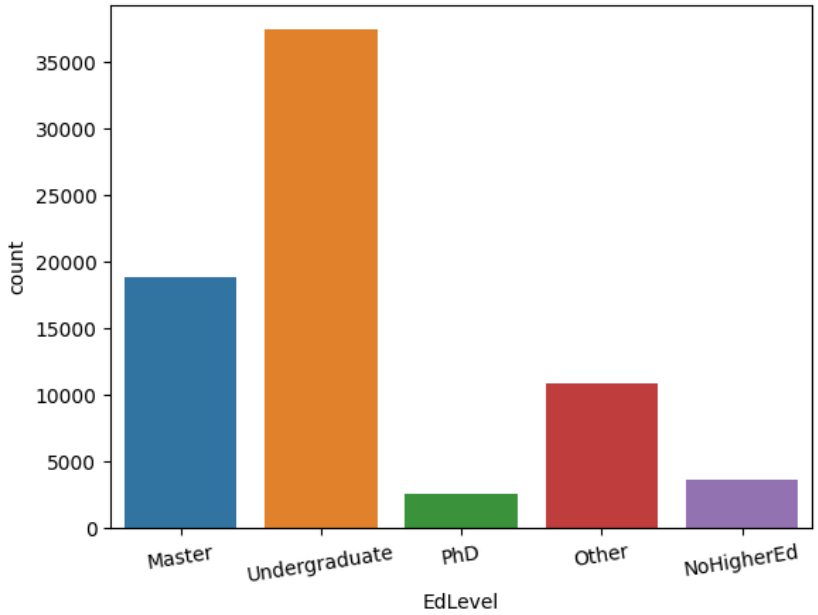
In [ ]:
```
categorical
```

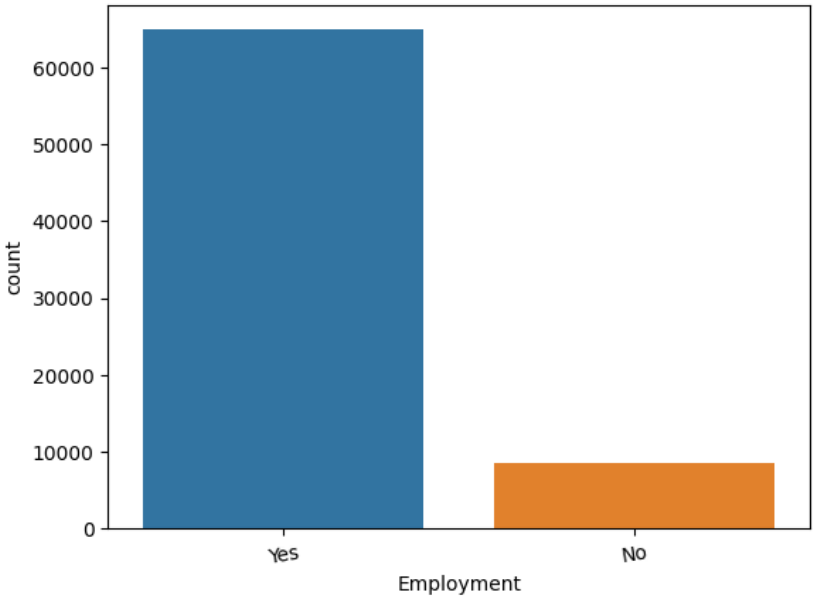
Out[ ]:
```
['Age',
 'Accessibility',
 'EdLevel',
 'Employment',
 'Gender',
 'MentalHealth',
 'MainBranch',
 'Employed',
 'Region']
```

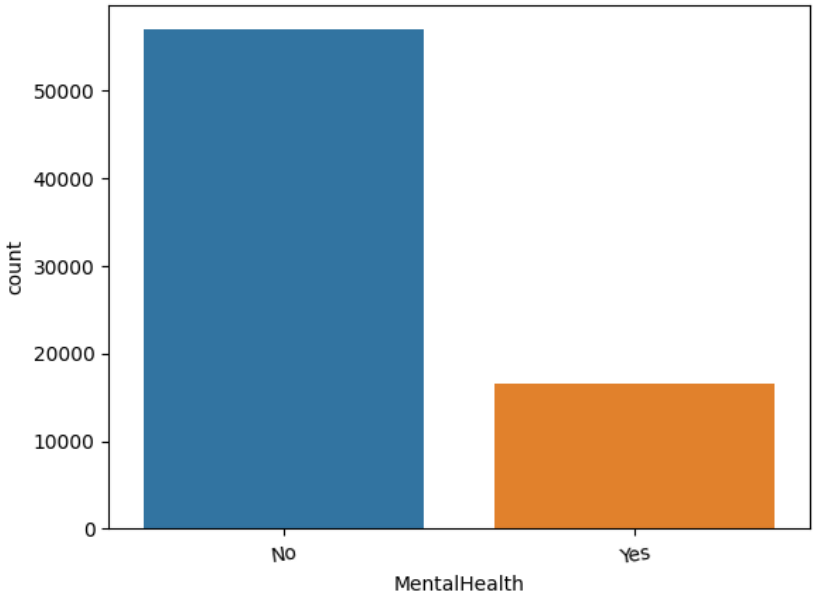
In [ ]:
```
for col in categorical:
  sns.countplot(data=df,x=col)
  plt.xlabel(col)
  plt.ylabel("count")
  plt.xticks(rotation=9)
  plt.show()
```

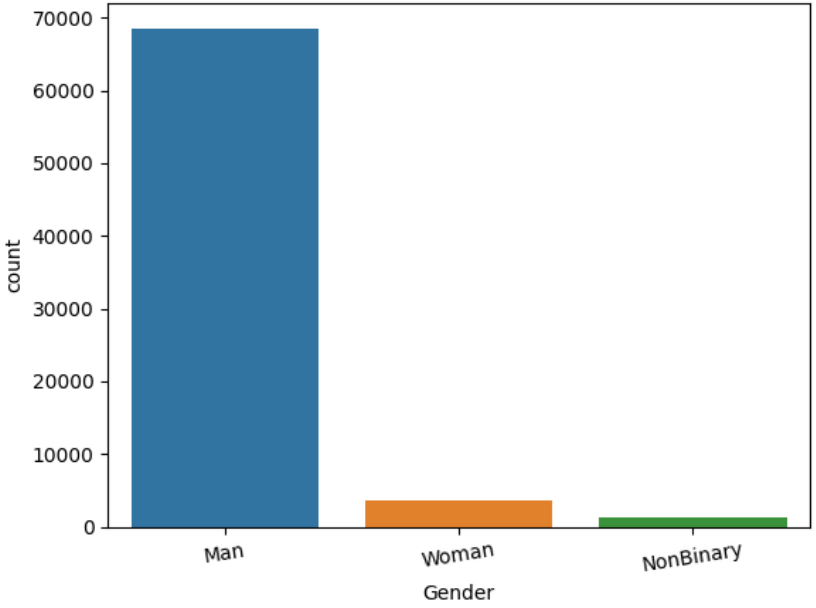


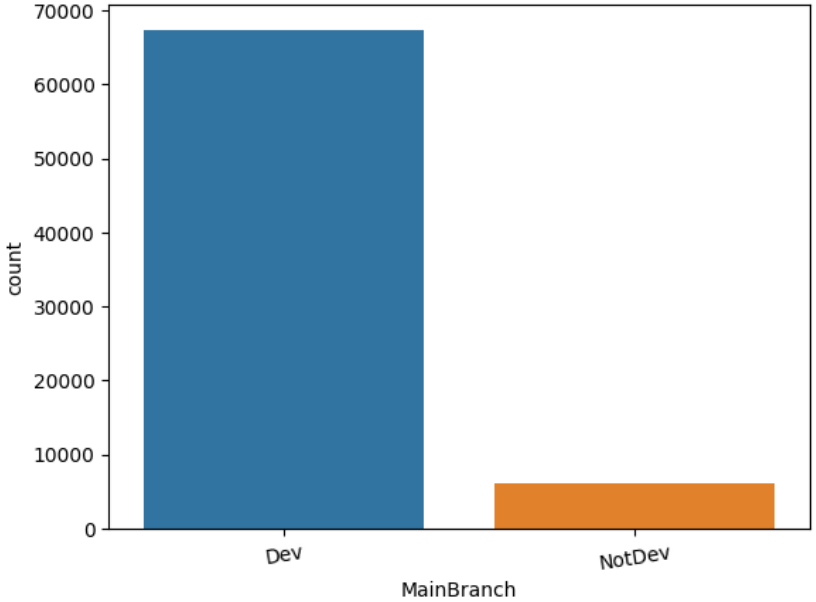


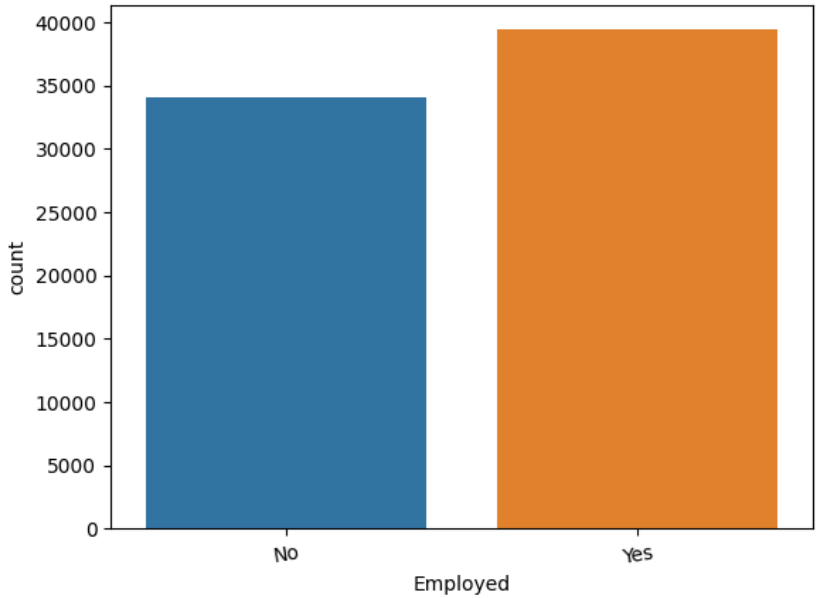


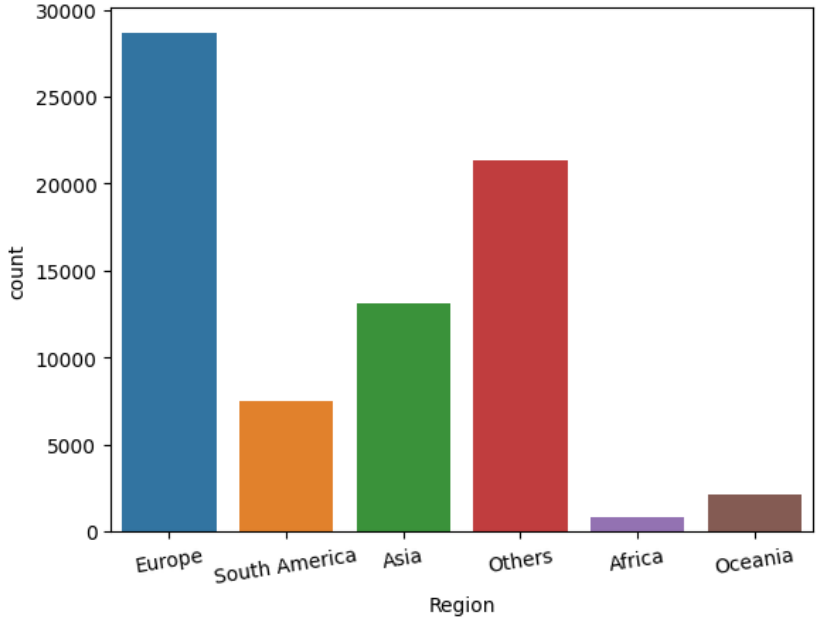

60000
50000
40000
30000
20000
10000
0
count
Yes
No
Employment


50000
40000
30000
20000
10000
0
count
No
Yes
MentalHealth


70000
60000
50000
40000
30000
20000
10000
0
count
Man
Woman
NonBinary
Gender


70000
60000
50000
40000
30000
20000
10000
0
count
Dev
NotDev
MainBranch

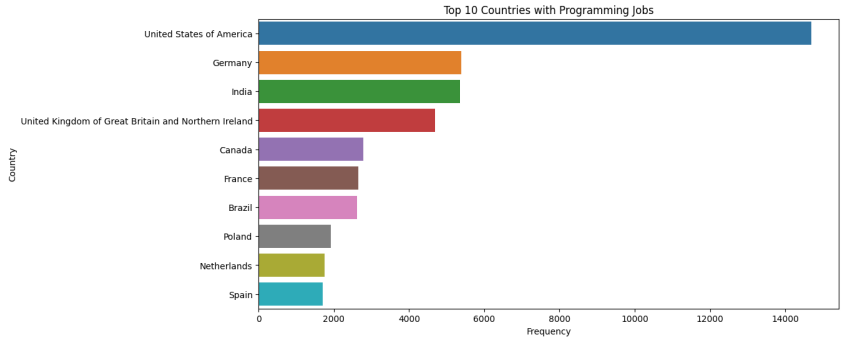




In [ ]:
```
# Get the top 10 most frequesnt countries
top_countries=df["Country"].value_counts().nlargest(10)
top_countries
```

Out[ ]:
```
United States of America                              14696
Germany                                                5395
India                                                  5360
United Kingdom of Great Britain and Northern Ireland   4688
Canada                                                 2779
France                                                 2650
Brazil                                                 2624
Poland                                                 1922
Netherlands                                            1761
Spain                                                  1712
Name: Country, dtype: int64
```

In [ ]:
```
#creating a bar chart
plt.figure(figsize=(12,6))
sns.barplot(x=top_countries,y=top_countries.index)
#labels.....
plt.xlabel("Frequency")
plt.ylabel("Country")
plt.title("Top 10 Countries with Programming Jobs")
#show plot.....
plt.show()
```



In [ ]:
```
top_countries.index
```

Out[ ]:
```
Index(['United States of America', 'Germany', 'India',
       'United Kingdom of Great Britain and Northern Ireland', 'Canada',
       'France', 'Brazil', 'Poland', 'Netherlands', 'Spain'],
      dtype='object')
```

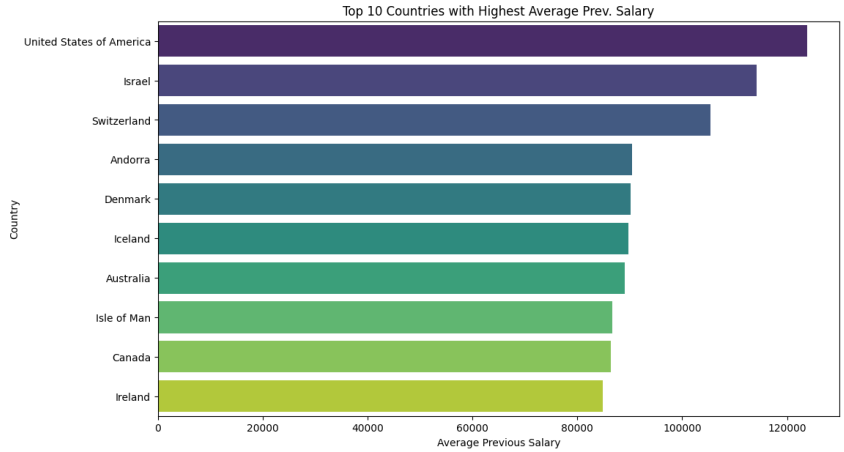
In [ ]:
```
# Group the data by "Country" and calculate the average "PreviousSalary"
avg_salary_by_country=df.groupby("Country")["PreviousSalary"].mean().nlargest(10).r
avg_salary_by_country
```

Out[ ]:

	Country	PreviousSalary
0	United States of America	123776.012520
1	Israel	114151.068653
2	Switzerland	105379.507592
3	Andorra	90379.272727
4	Denmark	90197.163324
5	Iceland	89739.400000
6	Australia	89010.656669
7	Isle of Man	86608.500000
8	Canada	86352.826556
9	Ireland	84811.465181

In [ ]:
```
# Creating a bar plot
plt.figure(figsize=(12,7))
sns.barplot(x="PreviousSalary",y="Country",data=avg_salary_by_country,palette="vir

#Set labels and title
plt.xlabel("Average Previous Salary")
plt.ylabel("Country")
plt.title("Top 10 Countries with Highest Average Prev. Salary")
```

Out[ ]:
```
Text(0.5, 1.0, 'Top 10 Countries with Highest Average Prev. Salary')
```



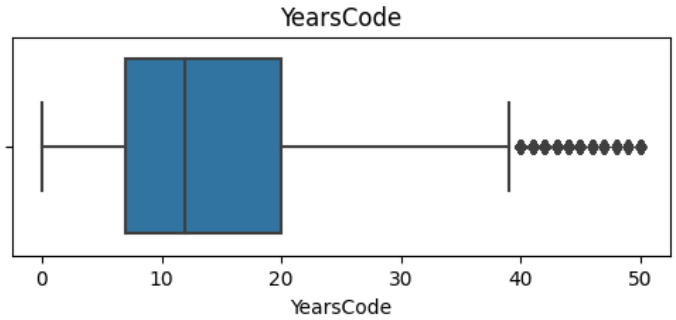
In [ ]:
```
df.info()
```

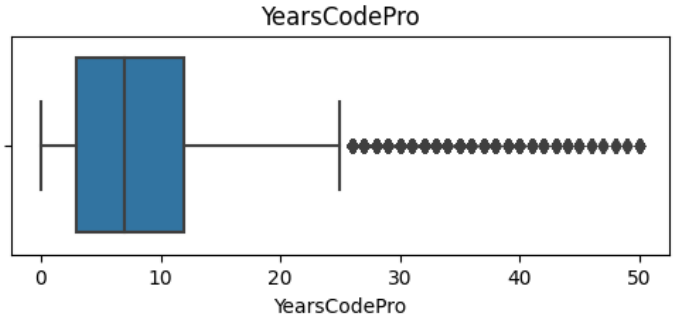
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73462 entries, 0 to 73461
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             73462 non-null  object
 1   Accessibility   73462 non-null  object
 2   EdLevel         73462 non-null  object
 3   Employment      73462 non-null  object
 4   Gender          73462 non-null  object
 5   MentalHealth    73462 non-null  object
 6   MainBranch      73462 non-null  object
 7   YearsCode       73462 non-null  int64
 8   YearsCodePro    73462 non-null  int64
 9   Country         73462 non-null  object
 10  PreviousSalary  73462 non-null  int64
 11  ComputerSkills  73462 non-null  int64
 12  Employed        73462 non-null  object
 13  Region          73462 non-null  object
dtypes: int64(4), object(10)
memory usage: 7.8+ MB
```

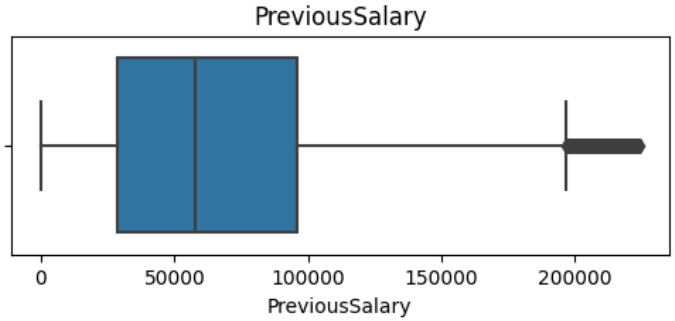
In [ ]:
```
#Get the names of all the columns with data type "int" or "float"
numerical=df.select_dtypes(["int","float"]).columns.tolist()
numerical
```

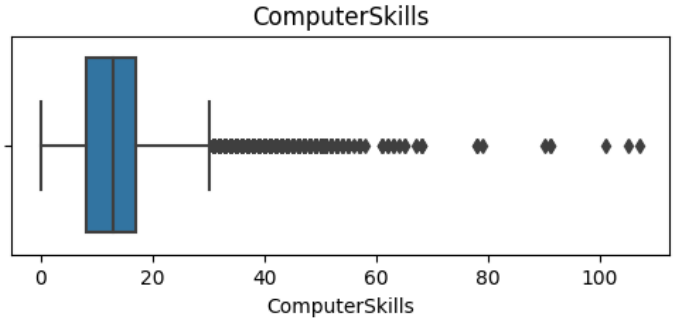
Out[ ]:
```
['YearsCode', 'YearsCodePro', 'PreviousSalary', 'ComputerSkills']
```

In [ ]:
```
#Create a boxplot
for i in numerical:
  plt.figure(figsize=(6,2))
  sns.boxplot(x=i,data=df)
  plt.title(i)
  plt.show()
```

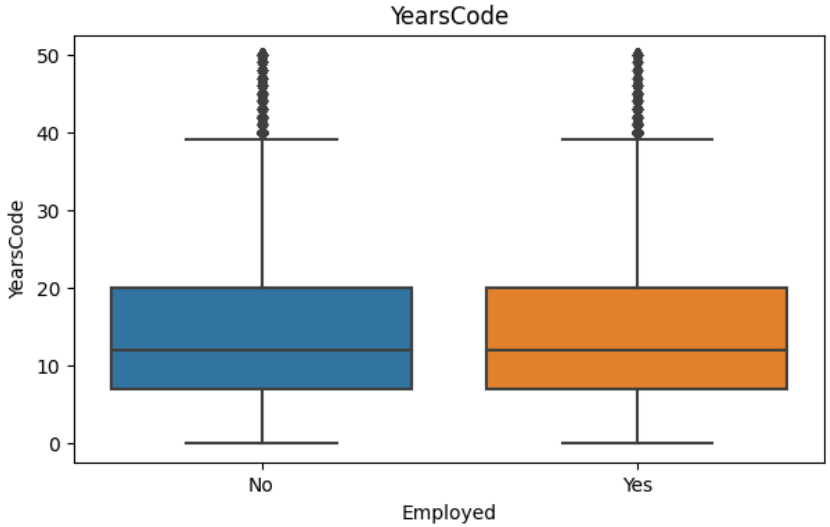


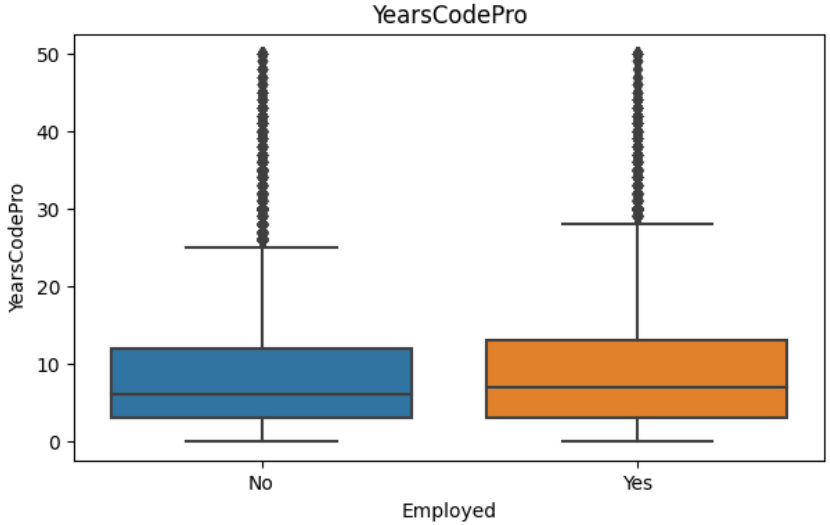


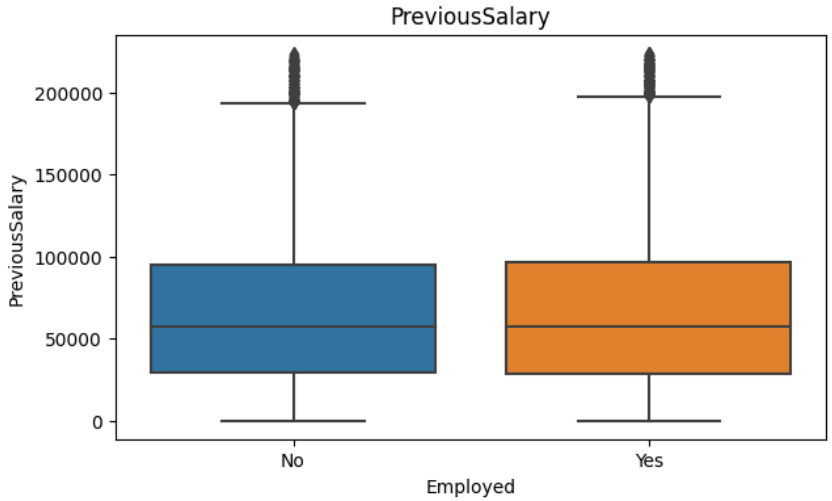


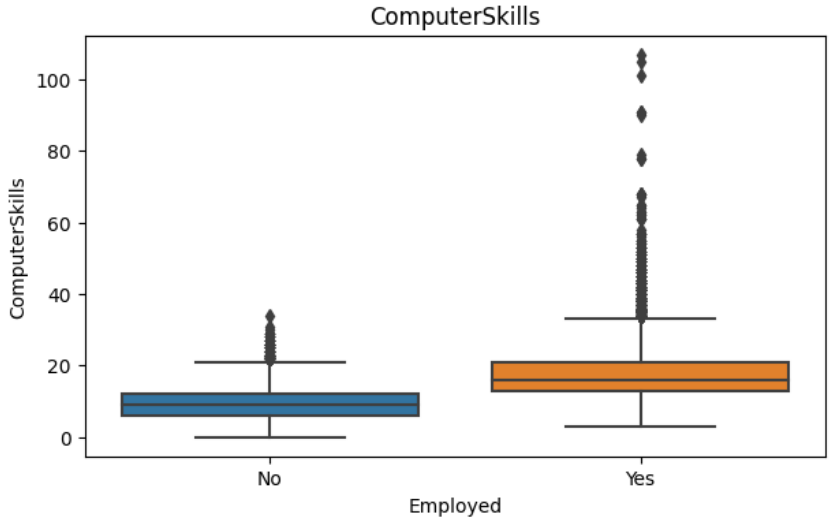


In [ ]:
```
#Create a boxplot
for i in numerical:
  plt.figure(figsize=(7,4))
  sns.boxplot(x="Employed",y=i,data=df)
  plt.title(i)
  plt.show()
```

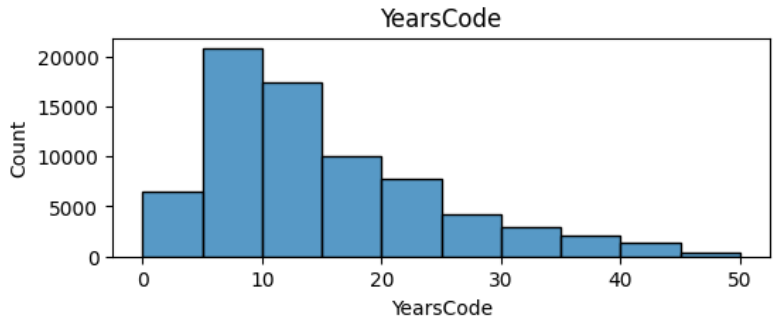


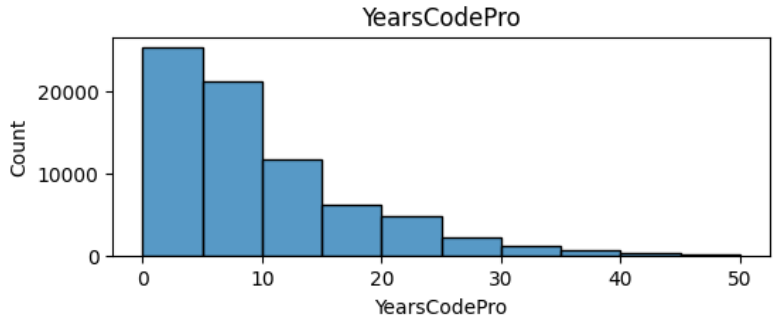


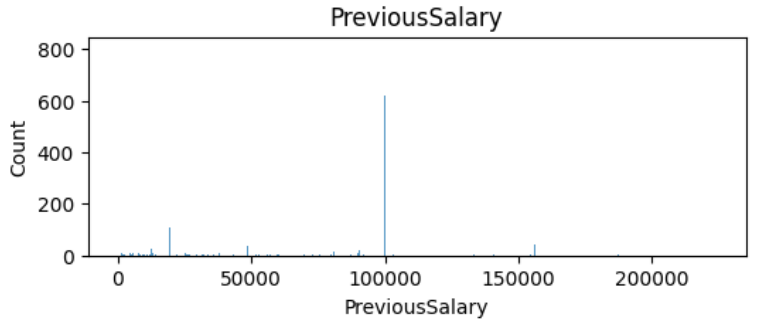


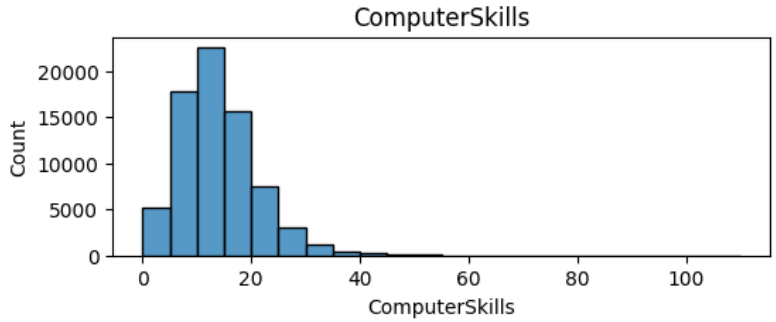


In [ ]:
```
#Creating a Histogram
for i in numerical:
  plt.figure(figsize=(6,2))
  sns.histplot(x=i,data=df,binwidth=5)
  plt.title(i)
  plt.show()
```



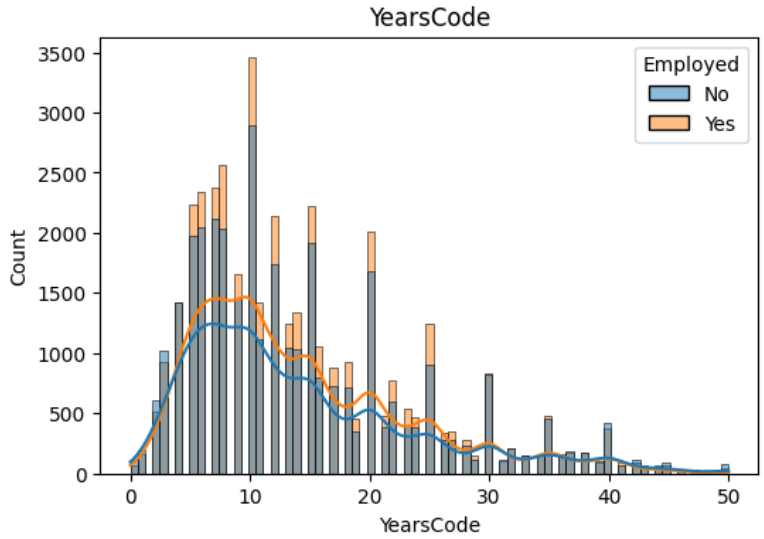


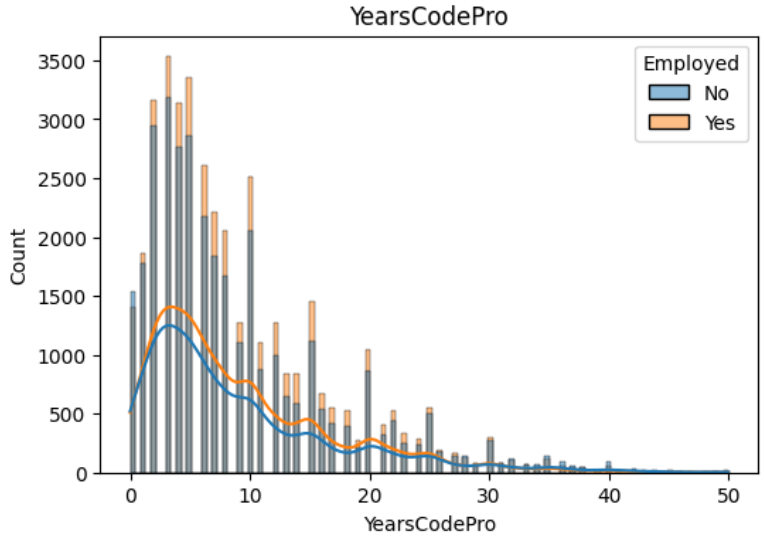


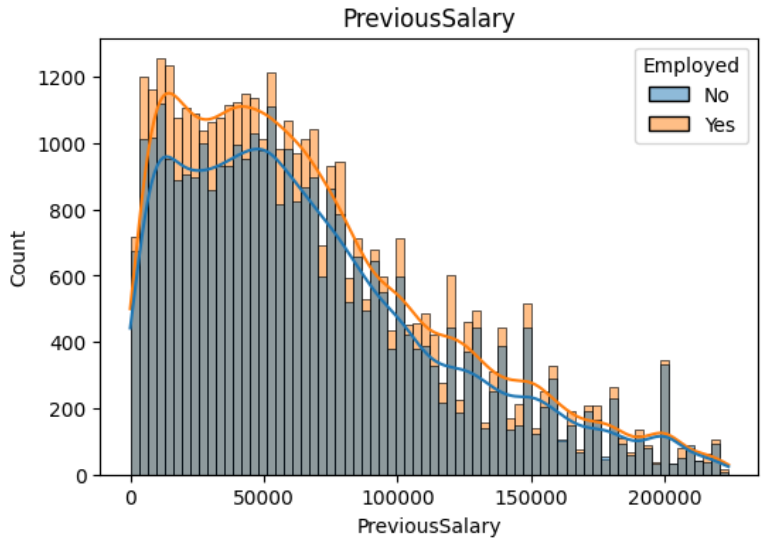


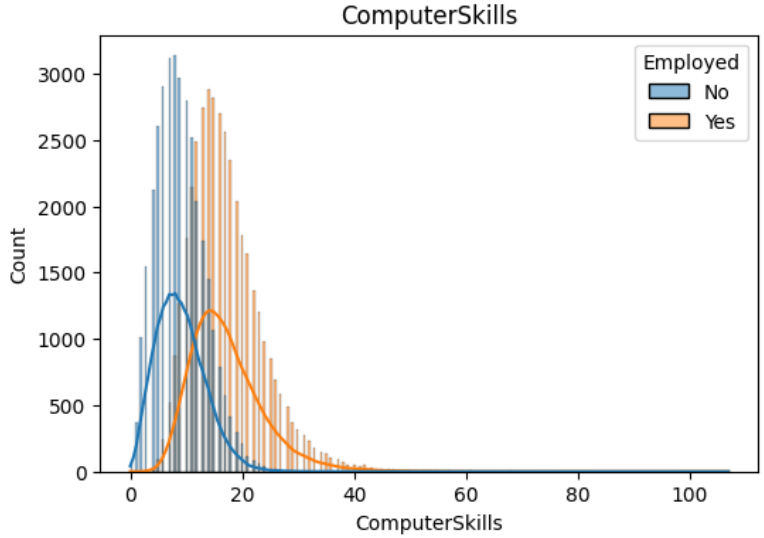
In [ ]:
```
#Creating a Histplot
for i in numerical:
  plt.figure(figsize=(6,4))
```

```
  sns.histplot(x=i,data=df,kde="True",hue="Employed")
  plt.title(i)
  plt.show()
```

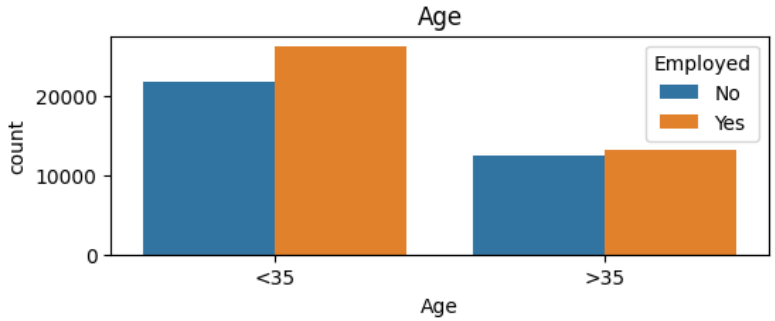


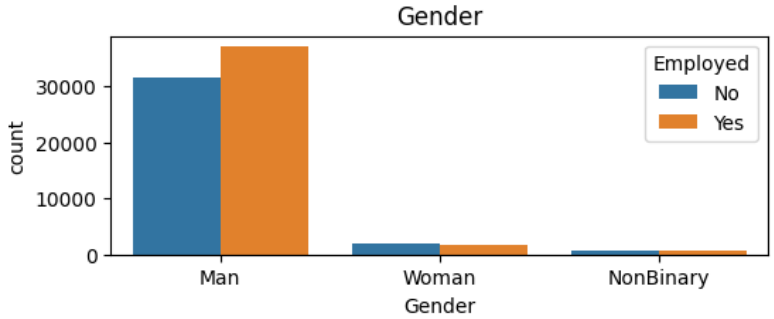


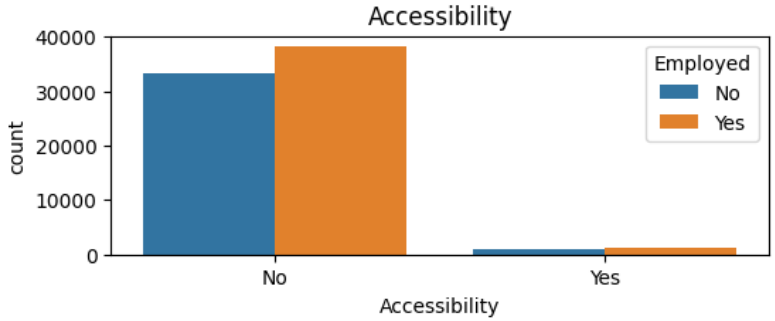


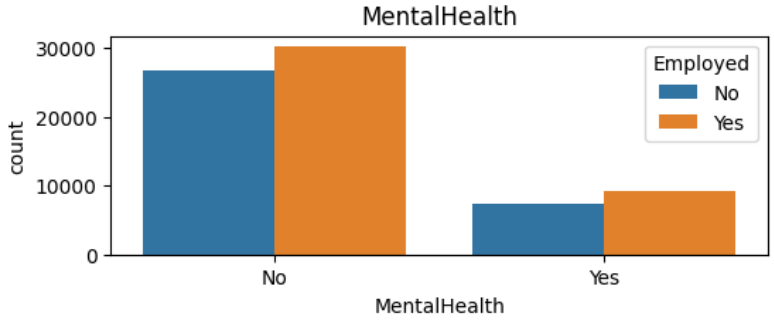


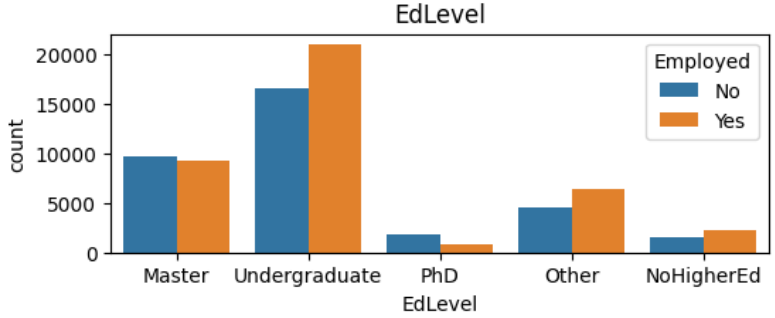
In [ ]:
```
#Creating a countplot for each categorical variable
for i in categorical:
  plt.figure(figsize=(6,2))
  sns.countplot(x=i,data=df,hue="Employed")
  plt.title(i)
  plt.show()
```

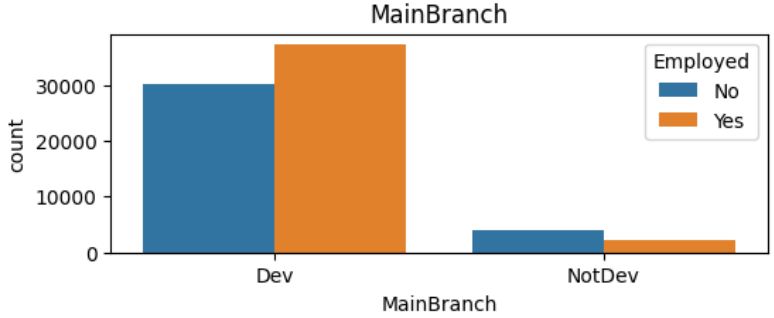

Age
Employed
No
Yes
count
20000
10000
0
<35
>35
Age

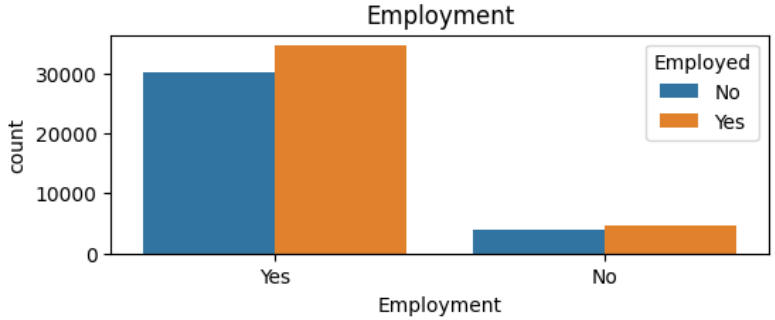

Gender
Employed
No
Yes
count
30000
20000
10000
0
Man
Woman
NonBinary
Gender

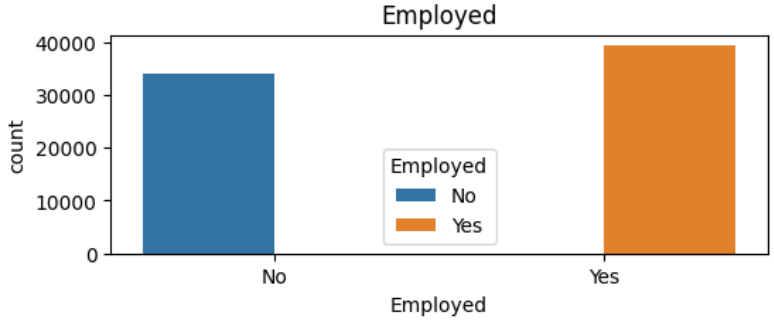

Accessibility
Employed
No
Yes
count
40000
30000
20000
10000
0
No
Yes
Accessibility

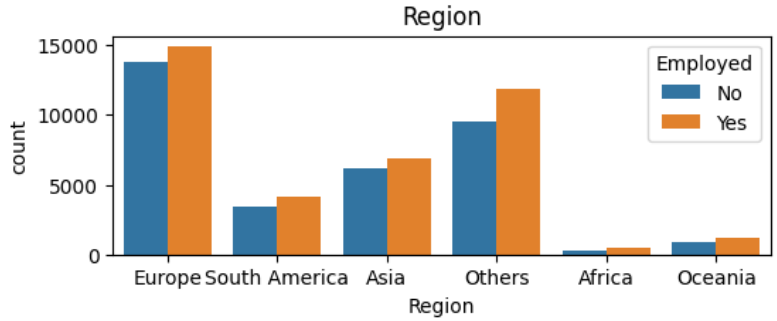

MentalHealth
Employed
No
Yes
count
30000
20000
10000
0
No
Yes
MentalHealth


EdLevel
Employed
No
Yes
count
20000
15000
10000
5000
0
Master
Undergraduate
PhD
Other
NoHigherEd
EdLevel

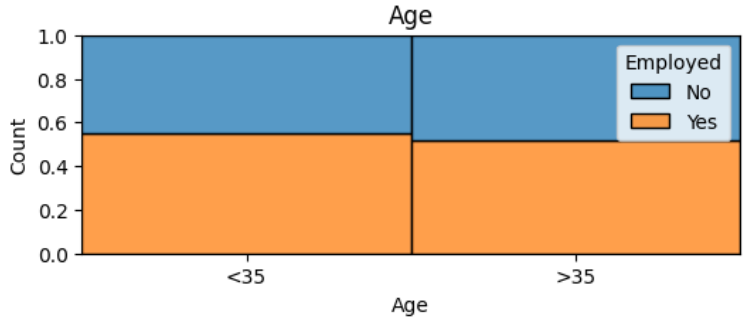

MainBranch
Employed
No
Yes
count
30000
20000
10000
0
Dev
NotDev
MainBranch

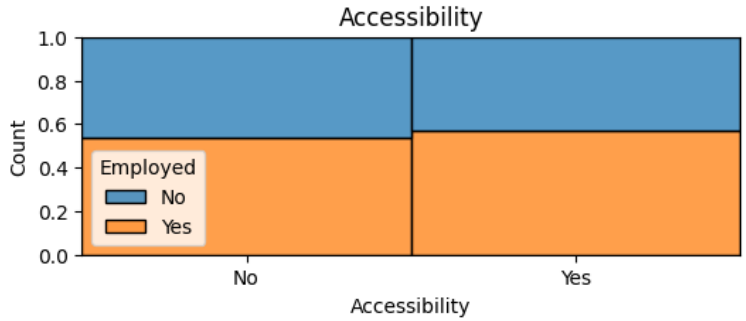

Employment
Employed
No
Yes
count
30000
20000
10000
0
Yes
No
Employment

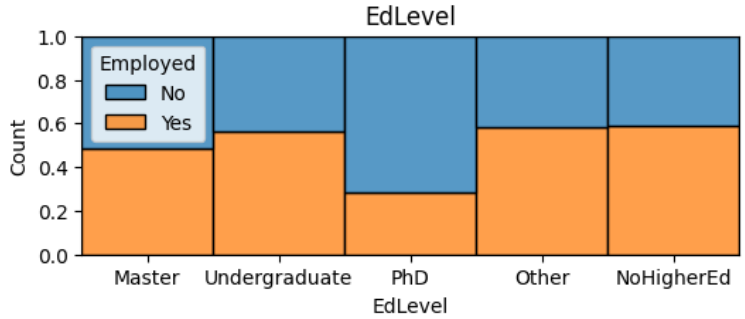

Employed
Employed
No
Yes
count
40000
30000
20000
10000
0
No
Yes
Employed

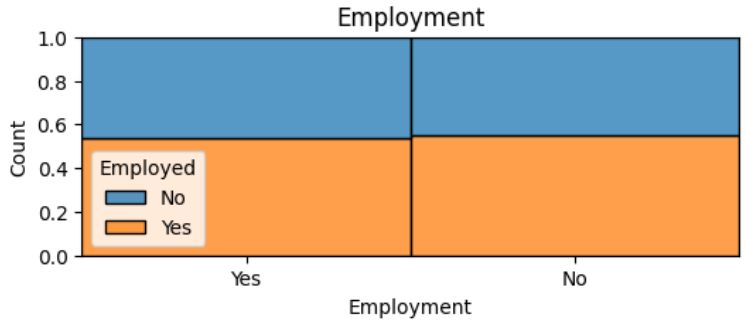


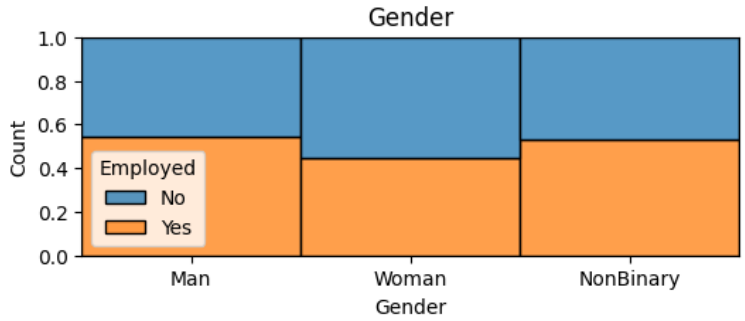
In [ ]:
```
#Creating a Density Plot for categorical values
for i in categorical:
  plt.figure(figsize=(6,2))
  sns.histplot(x=i,data=df,hue="Employed",kde=False,multiple="fill")

  plt.title(i)
  plt.show()
```

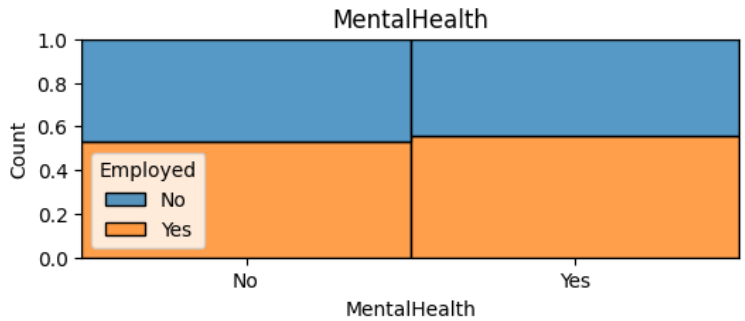


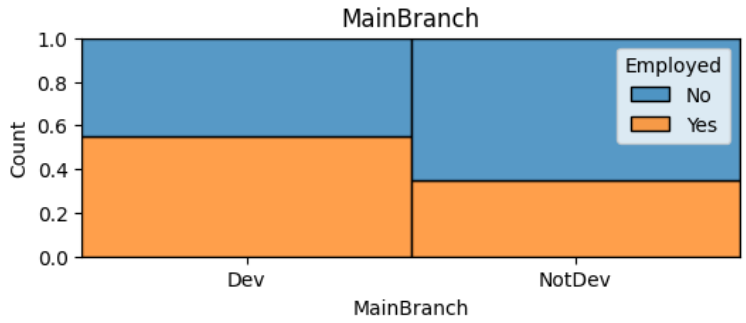


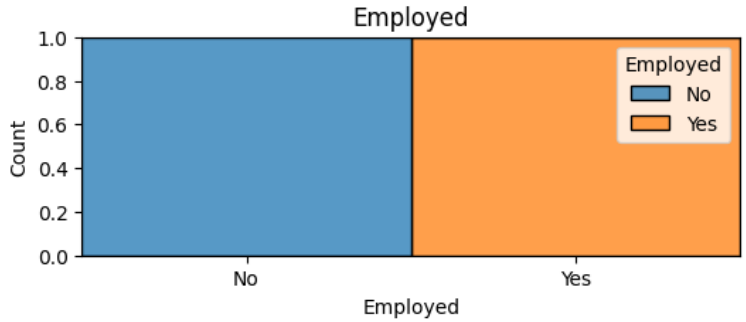


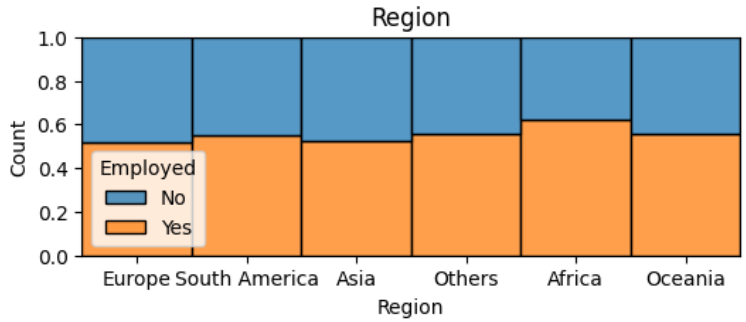












Data Preprocessing Part 2

In [ ]:
```
#Check the amount of missing values
missing=df.isnull().sum()*100/df.shape[0]
missing[missing>0].sort_values(ascending=False)
```

Out[ ]:
```
Series([], dtype: float64)
```

In [ ]:
```
#Drop Country
#df.drop(columns="Country",inplace=True)
df.head()
```

Out[ ]:

	Age	Accessibility	EdLevel	Employment	Gender	MentalHealth	MainBranch	YearsCode
0	<35	No	Master	Yes	Man	No	Dev	7
1	<35	No	Undergraduate	Yes	Man	No	Dev	12
2	<35	No	Master	Yes	Man	No	Dev	15
3	<35	No	Undergraduate	Yes	Man	No	Dev	9
4	>35	No	PhD	No	Man	No	NotDev	40

In [ ]:
```

```

Label encoding for Object Data Types

In [ ]:
```
# loop over each column where dtype is "object"
for col in df.select_dtypes("object").columns:
  print(col,df[col].unique())
```

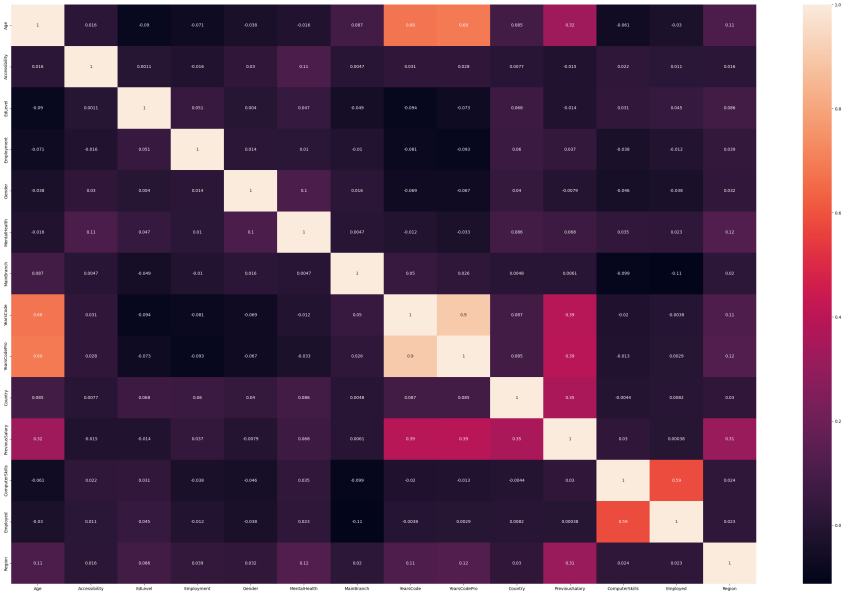
```
Age ['<35' '>35']
Accessibility ['No' 'Yes']
EdLevel ['Master' 'Undergraduate' 'PhD' 'Other' 'NoHigherEd']
Employment ['Yes' 'No']
Gender ['Man' 'Woman' 'NonBinary']
MentalHealth ['No' 'Yes']
MainBranch ['Dev' 'NotDev']
Country ['Sweden' 'Spain' 'Germany' 'Canada' 'Singapore' 'France' 'Switzerland'
 'United Kingdom of Great Britain and Northern Ireland'
 'Russian Federation' 'Israel' 'Turkey' 'United States of America'
 'Brazil' 'Bulgaria' 'Greece' 'Italy' 'Netherlands' 'Poland' 'Hungary'
 'Pakistan' 'Nigeria' 'Albania' 'Bangladesh' 'Viet Nam' 'Romania'
 'Sri Lanka' 'India' 'Lithuania' 'Ukraine' 'Croatia' 'Georgia' 'Denmark'
 'Ireland' 'Lebanon' 'Bahrain' 'Egypt' 'Colombia' 'Australia' 'Chile'
 'Indonesia' 'Iran, Islamic Republic of...' 'Portugal' 'Slovakia'
 'Armenia' 'Finland' 'Hong Kong (S.A.R.)' 'Argentina' 'Costa Rica' 'Peru'
 'Japan' 'Belgium' 'United Arab Emirates' 'Bolivia' 'Austria'
 'South Africa' 'Norway' 'Serbia' 'Malta' 'Malaysia' 'Czech Republic'
 'Belarus' 'Madagascar' 'Kenya' 'Slovenia' 'Uruguay'
 'The former Yugoslav Republic of Macedonia' 'Botswana' 'Algeria' 'China'
 'Mexico' 'Cyprus' 'Venezuela, Bolivarian Republic of...' 'Jordan'
 'Dominican Republic' 'Ecuador' 'Luxembourg' 'Uzbekistan'
 'Syrian Arab Republic' 'Zambia' 'Taiwan' 'Tunisia' 'South Korea'
 'Paraguay' 'Iceland' 'Morocco' 'Guatemala' 'Cameroon'
 'Republic of Moldova' 'Nepal' 'Kazakhstan' 'Estonia' 'Ethiopia' 'Cuba'
 'Latvia' "Lao People's Democratic Republic" 'Yemen'
 'Democratic Republic of the Congo' 'Somalia' 'Philippines' 'Azerbaijan'
 'Saudi Arabia' 'Honduras' 'Angola' 'Thailand' 'Bosnia and Herzegovina'
 'United Republic of Tanzania' 'El Salvador' 'Iraq' 'Kosovo' 'Andorra'
 'New Zealand' 'Dominica' 'Nicaragua' 'Mozambique' 'Trinidad and Tobago'
 'Nomadic' 'Panama' 'Barbados' 'Mauritius' 'Kuwait' 'Ghana'
 'Congo, Republic of the...' 'Myanmar' 'Republic of Korea' 'Cambodia'
 'Kyrgyzstan' 'Afghanistan' 'Uganda' 'Swaziland' 'Saint Kitts and Nevis'
 'Rwanda' 'Monaco' 'Turkmenistan' 'Libyan Arab Jamahiriya' 'Sudan'
 'Montenegro' 'Haiti' 'Belize' 'Benin' 'Bhutan' 'Palestine' 'Tajikistan'
 'Isle of Man' 'Cape Verde' 'Oman' "Côte d'Ivoire" 'Zimbabwe' 'Jamaica'
 'Guyana' 'Namibia' 'Togo' 'Lesotho' 'Guinea' 'Senegal' 'Mongolia'
 'Liberia' 'Suriname' 'Niger' 'Maldives' 'Qatar' 'Saint Lucia' 'Djibouti'
 'Malawi' 'Saint Vincent and the Grenadines' 'Gambia' 'Burkina Faso'
 'Fiji' 'Mauritania' 'Burundi' 'Timor-Leste' 'Mali' 'Seychelles']
Employed ['No' 'Yes']
Region ['Europe' 'South America' 'Asia' 'Others' 'Africa' 'Oceania']
```

In [ ]:
```
from sklearn import preprocessing

#Loop over each column

for col in df.select_dtypes("object").columns:
  #Initialize a label encoder object
  label_encoder=preprocessing.LabelEncoder()

  #Fit the encoder to the unique values
  label_encoder.fit(df[col].unique())

  #Transform the column using the encoder
  df[col]=label_encoder.transform(df[col])

  print(col,df[col].unique())
```

```
Age [0 1]
Accessibility [0 1]
EdLevel [0 4 3 2 1]
Employment [1 0]
Gender [0 2 1]
MentalHealth [0 1]
MainBranch [0 1]
Country [146 141  54  27 135  51 147 162 126  71 157 164  21  22  56  72 106 120
  63 114 110   1  11 168 125 142  65  87 160  34  53  40  69  83  10  45
  31   7  29  66  67 121 136   6  50  62   5  33 118  74  14 161  18   8
 139 112 133  94  91  37  13  89  77 137 165 152  20   2  30  97  36 167
  75  43  44  88 166 148 170 149 156 140 117  64 101  57  26 124 105  76
  47  48  35  82  81 169  39 138 119   9 131  61   4 151  19 163  46  68
  78   3 107  42 108 102 155 111 116  12  96  79  55  32 103 123  25  80
   0 159 145 128 127  98 158  86 143 100  60  15  16  17 115 150  70  28
 113  38 171  73  59 104 154  84  58 132  99  85 144 109  92 122 129  41
  90 130  52  23  49  95  24 153  93 134]
Employed [0 1]
Region [2 5 1 4 0 3]
```

In [ ]:
```
#Correlation heatmap
plt.figure(figsize=(40, 25))
sns.heatmap(df.corr(),annot=True,fmt='.2g')
```

Out[ ]:
```
<Axes: >
```



In [ ]:
```
# Remove YearCodePRo column bcz it has high correlation with YearsCode
df=df.drop("YearsCodePro",axis=1)
```

In [ ]:
```
df.head()
```

Out[ ]:

	Age	Accessibility	EdLevel	Employment	Gender	MentalHealth	MainBranch	YearsCode	Cour
0	0	0	0	1	0	0	0	7	
1	0	0	4	1	0	0	0	12	
2	0	0	0	1	0	0	0	15	
3	0	0	4	1	0	0	0	9	
4	1	0	3	0	0	0	1	40	

In [ ]:

Train Test Split

In [ ]:
```
x=df.drop("Employed",axis=1)
y=df["Employed"]#target column

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

In [ ]:

Remove Outliers from Train Data Using Z-score

In [ ]:
```
from scipy import stats
#Define the column for which you want to remove the outliers
selected_column=["YearsCode","PreviousSalary","ComputerSkills"]

#Calculate the z-scores for the selected colums in the training data
z_stats=stats.zscore(x_train[selected_column])
z_scores=np.abs(z_stats)

#Set a threshold value =3 for outlier detection
threshold=3

#Find the indices of outliers
outlier_indices=np.where(z_scores>threshold)[0]

#Remove the outliers from the training data
x_train=x_train.drop(x_train.index[outlier_indices])
y_train=y_train.drop(y_train.index[outlier_indices])
```

In [ ]:

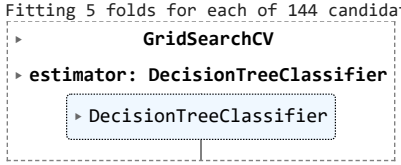
Decision Tree Classifier

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
```

```
dtree=DecisionTreeClassifier(class_weight="balanced")
param_grid = {
    'max_depth': [3, 4, 5, 6, 7, 8],
    'min_samples_split': [2, 3, 4],
    'min_samples_leaf': [1, 2, 3, 4],
    'random_state': [0, 42]
}

gscv=GridSearchCV(dtree,param_grid,cv=5,verbose=1)
gscv.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 144 candidates, totalling 720 fits
```

Out[ ]:

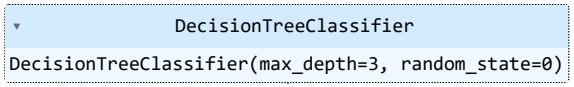


In [ ]:
```
gscv.best_params_
```

Out[ ]:
```
{'max_depth': 3,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'random_state': 0}
```

In [ ]:
```
dtree=DecisionTreeClassifier(max_depth=3,min_samples_leaf=1,min_samples_split=2,ran
dtree.fit(x_train,y_train)
```

Out[ ]:



In [ ]:
```
y_pred=dtree.predict(x_test)
accu_score=round(accuracy_score(y_test,y_pred),2)
print(accu_score*100,"%")
```

```
79.0 %
```

In [ ]:
```
from sklearn.metrics import accuracy_score,f1_score,precision_score,recall_score,lo

fscore=f1_score(y_test,y_pred,average="micro")
pscore=precision_score(y_test,y_pred,average="micro")
recallScore=recall_score(y_test,y_pred,average="micro")
jscore=jaccard_score(y_test,y_pred,average="micro")
logLoss=log_loss(y_test,y_pred)


print("F1_Score",fscore)
print("Precision_score",pscore)
print("Recall Score",recallScore)
print("Jaccard_score",jscore)
print("Log loss",logLoss)
```

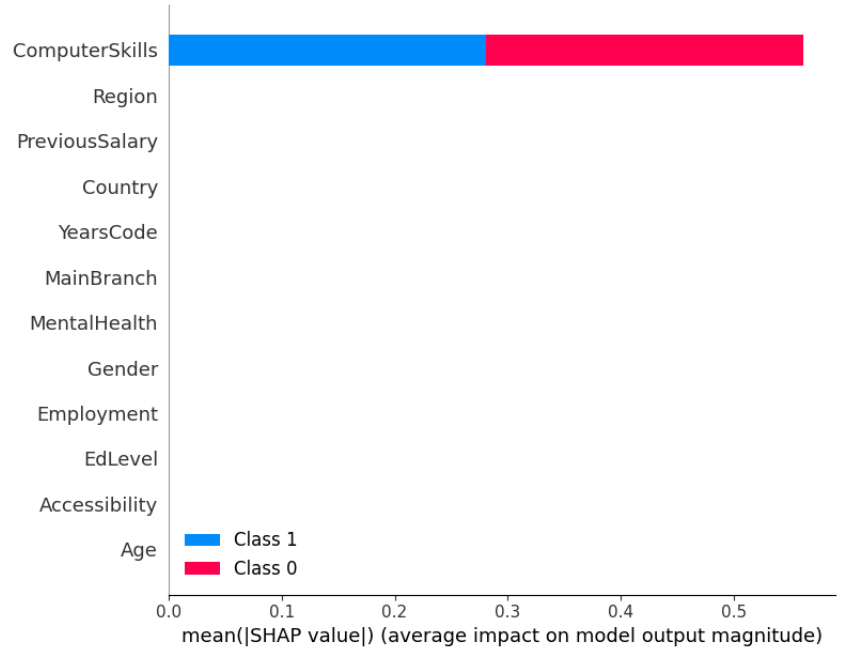
```
F1_Score 0.785680255904172
Precision_score 0.785680255904172
Recall Score 0.785680255904172
Jaccard_score 0.6470126667413967
Log loss 7.724866570634309
```

In [ ]:
```
!pip install shap
import shap
explainer=shap.TreeExplainer(dtree)
```

```
shap_values=explainer.shap_values(x_test)
shap.summary_plot(shap_values,x_test)
```

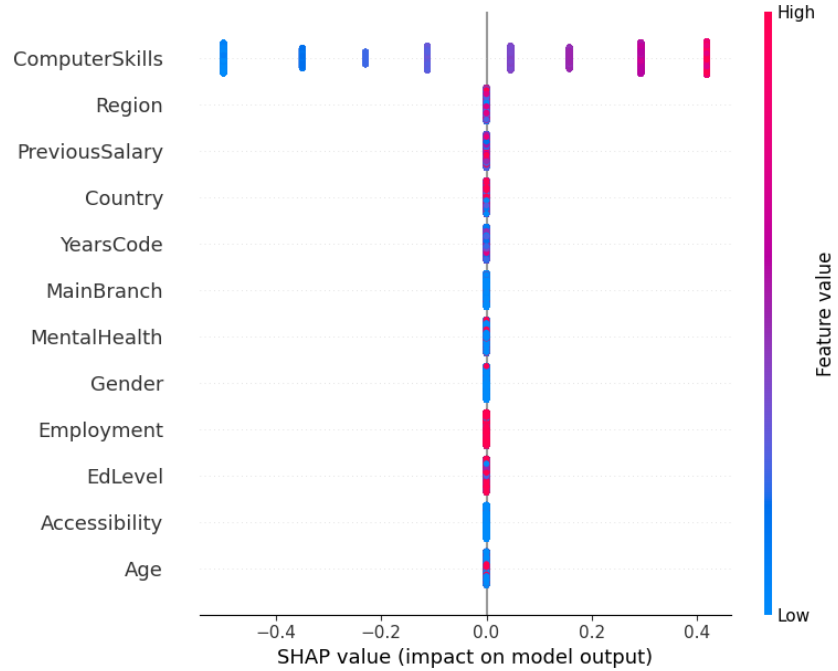
```
Collecting shap
  Downloading shap-0.42.1-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (547 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 547.9/547.9 kB 5.1 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from shap) (1.23.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from shap) (1.11.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from shap) (1.2.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from shap) (1.5.3)
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.10/dist-packages (from shap) (4.66.1)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.10/dist-packages (from shap) (23.1)
Collecting slicer==0.0.7 (from shap)
  Downloading slicer-0.0.7-py3-none-any.whl (14 kB)
Requirement already satisfied: numba in /usr/local/lib/python3.10/dist-packages (from shap) (0.56.4)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-packages (from shap) (2.2.1)
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/python3.10/dist-packages (from numba->shap) (0.39.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from numba->shap) (67.7.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->shap) (2023.3.post1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->shap) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->shap) (3.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas->shap) (1.16.0)
Installing collected packages: slicer, shap
Successfully installed shap-0.42.1 slicer-0.0.7
```

```
Using `tqdm.autonotebook.tqdm` in notebook mode. Use `tqdm.tqdm` instead to force console mode (e.g. in jupyter console)
```



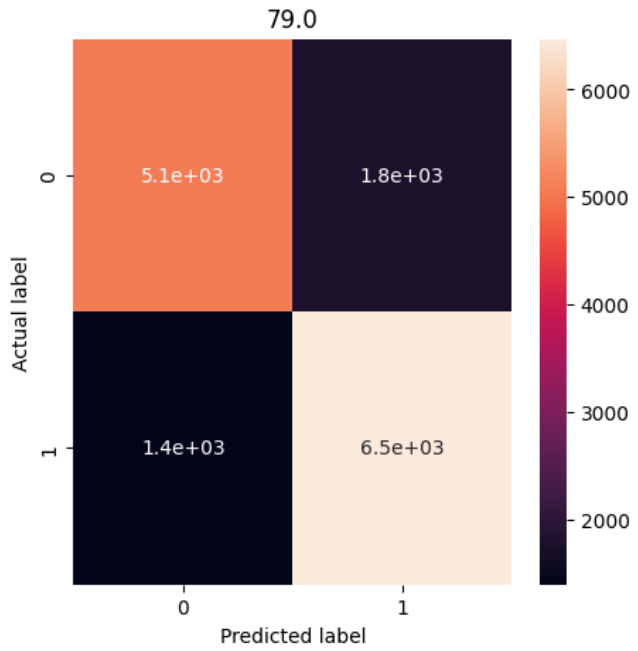
In [ ]:
```
#compute SHAP values

shap.summary_plot(shap_values[1],x_test.values,feature_names=x_test.columns)
```

```
No data for colormapping provided via 'c'. Parameters 'vmin', 'vmax' will be ignored
```



In [ ]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,annot=True)

plt.xlabel("Predicted label")
plt.ylabel("Actual label")

plt.title(accu_score*100)
```

Out[ ]:
```
Text(0.5, 1.0, '79.0')
```



In [ ]:

Random Forest Classifier

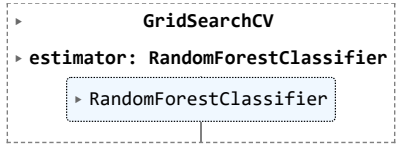
In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

rf=RandomForestClassifier(class_weight="balanced")
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 5, 10],
    'max_features': ['sqrt', 'log2', None],
    'random_state': [0, 42]
}

gscv=GridSearchCV(rf,param_grid,cv=5,verbose=2)
gscv.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
[CV] END max_depth=None, max_features=sqrt, n_estimators=100, random_state=0; tota
l time=  14.1s
[CV] END max_depth=None, max_features=sqrt, n_estimators=100, random_state=0; tota
l time=  15.5s
[CV] END max_depth=None, max_features=sqrt, n_estimators=100, random_state=0; tota
l time=  14.5s
[CV] END max_depth=None, max_features=sqrt, n_estimators=100, random_state=0; tota
l time=   8.6s
[CV] END max_depth=None, max_features=sqrt, n_estimators=100, random_state=0; tota
l time=   6.3s
[CV] END max_depth=None, max_features=sqrt, n_estimators=100, random_state=42; tot
al time=   7.2s
[CV] END max_depth=None, max_features=sqrt, n_estimators=100, random_state=42; tot
al time=   6.2s
[CV] END max_depth=None, max_features=sqrt, n_estimators=100, random_state=42; tot
al time=   7.1s
[CV] END max_depth=None, max_features=sqrt, n_estimators=100, random_state=42; tot
al time=   6.2s
[CV] END max_depth=None, max_features=sqrt, n_estimators=100, random_state=42; tot
al time=   7.2s
[CV] END max_depth=None, max_features=sqrt, n_estimators=200, random_state=0; tota
l time=  13.9s
[CV] END max_depth=None, max_features=sqrt, n_estimators=200, random_state=0; tota
l time=  13.5s
[CV] END max_depth=None, max_features=sqrt, n_estimators=200, random_state=0; tota
l time=  13.3s
[CV] END max_depth=None, max_features=sqrt, n_estimators=200, random_state=0; tota
l time=  16.0s
[CV] END max_depth=None, max_features=sqrt, n_estimators=200, random_state=0; tota
l time=  13.2s
[CV] END max_depth=None, max_features=sqrt, n_estimators=200, random_state=42; tot
al time=  13.3s
[CV] END max_depth=None, max_features=sqrt, n_estimators=200, random_state=42; tot
al time=  13.2s
[CV] END max_depth=None, max_features=sqrt, n_estimators=200, random_state=42; tot
al time=  13.2s
[CV] END max_depth=None, max_features=sqrt, n_estimators=200, random_state=42; tot
al time=  15.9s
[CV] END max_depth=None, max_features=sqrt, n_estimators=200, random_state=42; tot
al time=  25.9s
[CV] END max_depth=None, max_features=log2, n_estimators=100, random_state=0; tota
l time=   6.2s
[CV] END max_depth=None, max_features=log2, n_estimators=100, random_state=0; tota
l time=   7.1s
[CV] END max_depth=None, max_features=log2, n_estimators=100, random_state=0; tota
l time=   6.1s
[CV] END max_depth=None, max_features=log2, n_estimators=100, random_state=0; tota
l time=   7.1s
[CV] END max_depth=None, max_features=log2, n_estimators=100, random_state=0; tota
l time=   6.2s
[CV] END max_depth=None, max_features=log2, n_estimators=100, random_state=42; tot
al time=   7.1s
[CV] END max_depth=None, max_features=log2, n_estimators=100, random_state=42; tot
al time=   6.1s
[CV] END max_depth=None, max_features=log2, n_estimators=100, random_state=42; tot
al time=   7.0s
[CV] END max_depth=None, max_features=log2, n_estimators=100, random_state=42; tot
al time=   6.1s
[CV] END max_depth=None, max_features=log2, n_estimators=100, random_state=42; tot
al time=   7.1s
[CV] END max_depth=None, max_features=log2, n_estimators=200, random_state=0; tota
l time=  13.3s
[CV] END max_depth=None, max_features=log2, n_estimators=200, random_state=0; tota
l time=  13.3s
[CV] END max_depth=None, max_features=log2, n_estimators=200, random_state=0; tota
l time=  13.9s
[CV] END max_depth=None, max_features=log2, n_estimators=200, random_state=0; tota
l time=  15.0s
[CV] END max_depth=None, max_features=log2, n_estimators=200, random_state=0; tota
l time=  13.3s
[CV] END max_depth=None, max_features=log2, n_estimators=200, random_state=42; tot
al time=  13.3s
[CV] END max_depth=None, max_features=log2, n_estimators=200, random_state=42; tot
al time=  13.1s
[CV] END max_depth=None, max_features=log2, n_estimators=200, random_state=42; tot
al time=  13.2s
[CV] END max_depth=None, max_features=log2, n_estimators=200, random_state=42; tot
al time=  13.2s
[CV] END max_depth=None, max_features=log2, n_estimators=200, random_state=42; tot
al time=  13.5s
[CV] END max_depth=None, max_features=None, n_estimators=100, random_state=0; tota
l time=  17.7s
[CV] END max_depth=None, max_features=None, n_estimators=100, random_state=0; tota
l time=  17.2s
[CV] END max_depth=None, max_features=None, n_estimators=100, random_state=0; tota
l time=  17.3s
[CV] END max_depth=None, max_features=None, n_estimators=100, random_state=0; tota
l time=  17.5s
[CV] END max_depth=None, max_features=None, n_estimators=100, random_state=0; tota
l time=  17.0s
[CV] END max_depth=None, max_features=None, n_estimators=100, random_state=42; tot
al time=  17.9s
[CV] END max_depth=None, max_features=None, n_estimators=100, random_state=42; tot
al time=  17.1s
[CV] END max_depth=None, max_features=None, n_estimators=100, random_state=42; tot
al time=  16.8s
[CV] END max_depth=None, max_features=None, n_estimators=100, random_state=42; tot
al time=  17.9s
[CV] END max_depth=None, max_features=None, n_estimators=100, random_state=42; tot
al time=  18.7s
[CV] END max_depth=None, max_features=None, n_estimators=200, random_state=0; tota
l time=  34.7s
[CV] END max_depth=None, max_features=None, n_estimators=200, random_state=0; tota
l time=  34.1s
[CV] END max_depth=None, max_features=None, n_estimators=200, random_state=0; tota
l time=  34.2s
[CV] END max_depth=None, max_features=None, n_estimators=200, random_state=0; tota
l time=  34.8s
[CV] END max_depth=None, max_features=None, n_estimators=200, random_state=0; tota
l time=  34.9s
[CV] END max_depth=None, max_features=None, n_estimators=200, random_state=42; tot
al time=  33.9s
[CV] END max_depth=None, max_features=None, n_estimators=200, random_state=42; tot
al time=  34.9s
[CV] END max_depth=None, max_features=None, n_estimators=200, random_state=42; tot
al time=  34.4s
[CV] END max_depth=None, max_features=None, n_estimators=200, random_state=42; tot
al time=  34.2s
[CV] END max_depth=None, max_features=None, n_estimators=200, random_state=42; tot
al time=  34.9s
[CV] END max_depth=5, max_features=sqrt, n_estimators=100, random_state=0; total t
ime=   2.1s
[CV] END max_depth=5, max_features=sqrt, n_estimators=100, random_state=0; total t
ime=   2.1s
[CV] END max_depth=5, max_features=sqrt, n_estimators=100, random_state=0; total t
ime=   2.9s
[CV] END max_depth=5, max_features=sqrt, n_estimators=100, random_state=0; total t
```

ime=   2.1s
[CV] END max_depth=5, max_features=sqrt, n_estimators=100, random_state=0; total t
ime=   2.1s
[CV] END max_depth=5, max_features=sqrt, n_estimators=100, random_state=42; total
time=   2.1s
[CV] END max_depth=5, max_features=sqrt, n_estimators=100, random_state=42; total
time=   2.0s
[CV] END max_depth=5, max_features=sqrt, n_estimators=100, random_state=42; total
time=   2.4s
[CV] END max_depth=5, max_features=sqrt, n_estimators=100, random_state=42; total
time=   2.6s
[CV] END max_depth=5, max_features=sqrt, n_estimators=100, random_state=42; total
time=   2.1s
[CV] END max_depth=5, max_features=sqrt, n_estimators=200, random_state=0; total t
ime=   4.1s
[CV] END max_depth=5, max_features=sqrt, n_estimators=200, random_state=0; total t
ime=   4.7s
[CV] END max_depth=5, max_features=sqrt, n_estimators=200, random_state=0; total t
ime=   4.4s
[CV] END max_depth=5, max_features=sqrt, n_estimators=200, random_state=0; total t
ime=   4.1s
[CV] END max_depth=5, max_features=sqrt, n_estimators=200, random_state=0; total t
ime=   5.0s
[CV] END max_depth=5, max_features=sqrt, n_estimators=200, random_state=42; total
time=   4.1s
[CV] END max_depth=5, max_features=sqrt, n_estimators=200, random_state=42; total
time=   4.2s
[CV] END max_depth=5, max_features=sqrt, n_estimators=200, random_state=42; total
time=   4.9s
[CV] END max_depth=5, max_features=sqrt, n_estimators=200, random_state=42; total
time=   4.0s
[CV] END max_depth=5, max_features=sqrt, n_estimators=200, random_state=42; total
time=   4.1s
[CV] END max_depth=5, max_features=log2, n_estimators=100, random_state=0; total t
ime=   2.4s
[CV] END max_depth=5, max_features=log2, n_estimators=100, random_state=0; total t
ime=   2.7s
[CV] END max_depth=5, max_features=log2, n_estimators=100, random_state=0; total t
ime=   2.1s
[CV] END max_depth=5, max_features=log2, n_estimators=100, random_state=0; total t
ime=   2.1s
[CV] END max_depth=5, max_features=log2, n_estimators=100, random_state=0; total t
ime=   2.1s
[CV] END max_depth=5, max_features=log2, n_estimators=100, random_state=42; total
time=   2.1s
[CV] END max_depth=5, max_features=log2, n_estimators=100, random_state=42; total
time=   2.7s
[CV] END max_depth=5, max_features=log2, n_estimators=100, random_state=42; total
time=   2.3s
[CV] END max_depth=5, max_features=log2, n_estimators=100, random_state=42; total
time=   2.0s
[CV] END max_depth=5, max_features=log2, n_estimators=100, random_state=42; total
time=   2.0s
[CV] END max_depth=5, max_features=log2, n_estimators=200, random_state=0; total t
ime=   4.1s
[CV] END max_depth=5, max_features=log2, n_estimators=200, random_state=0; total t
ime=   5.0s
[CV] END max_depth=5, max_features=log2, n_estimators=200, random_state=0; total t
ime=   4.1s
[CV] END max_depth=5, max_features=log2, n_estimators=200, random_state=0; total t
ime=   4.2s
[CV] END max_depth=5, max_features=log2, n_estimators=200, random_state=0; total t
ime=   5.0s
[CV] END max_depth=5, max_features=log2, n_estimators=200, random_state=42; total
time=   4.0s
[CV] END max_depth=5, max_features=log2, n_estimators=200, random_state=42; total
time=   4.3s
[CV] END max_depth=5, max_features=log2, n_estimators=200, random_state=42; total
time=   4.6s
[CV] END max_depth=5, max_features=log2, n_estimators=200, random_state=42; total
time=   4.0s
[CV] END max_depth=5, max_features=log2, n_estimators=200, random_state=42; total
time=   4.4s
[CV] END max_depth=5, max_features=None, n_estimators=100, random_state=0; total t
ime=   6.0s
[CV] END max_depth=5, max_features=None, n_estimators=100, random_state=0; total t
ime=   5.5s
[CV] END max_depth=5, max_features=None, n_estimators=100, random_state=0; total t
ime=   6.3s
[CV] END max_depth=5, max_features=None, n_estimators=100, random_state=0; total t
ime=   5.4s
[CV] END max_depth=5, max_features=None, n_estimators=100, random_state=0; total t
ime=   6.4s
[CV] END max_depth=5, max_features=None, n_estimators=100, random_state=42; total
time=   5.5s
[CV] END max_depth=5, max_features=None, n_estimators=100, random_state=42; total
time=   6.4s
[CV] END max_depth=5, max_features=None, n_estimators=100, random_state=42; total
time=   5.4s
[CV] END max_depth=5, max_features=None, n_estimators=100, random_state=42; total
time=   6.3s
[CV] END max_depth=5, max_features=None, n_estimators=100, random_state=42; total
time=   5.4s
[CV] END max_depth=5, max_features=None, n_estimators=200, random_state=0; total t
ime= 11.8s
[CV] END max_depth=5, max_features=None, n_estimators=200, random_state=0; total t
ime= 11.7s
[CV] END max_depth=5, max_features=None, n_estimators=200, random_state=0; total t
ime= 11.6s
[CV] END max_depth=5, max_features=None, n_estimators=200, random_state=0; total t
ime= 11.8s
[CV] END max_depth=5, max_features=None, n_estimators=200, random_state=0; total t
ime= 11.7s
[CV] END max_depth=5, max_features=None, n_estimators=200, random_state=42; total
time= 11.8s
[CV] END max_depth=5, max_features=None, n_estimators=200, random_state=42; total
time= 11.2s
[CV] END max_depth=5, max_features=None, n_estimators=200, random_state=42; total
time= 11.4s
[CV] END max_depth=5, max_features=None, n_estimators=200, random_state=42; total
time= 11.8s
[CV] END max_depth=5, max_features=None, n_estimators=200, random_state=42; total
time= 11.8s
[CV] END max_depth=10, max_features=sqrt, n_estimators=100, random_state=0; total
time=   3.8s
[CV] END max_depth=10, max_features=sqrt, n_estimators=100, random_state=0; total
time=   3.7s
[CV] END max_depth=10, max_features=sqrt, n_estimators=100, random_state=0; total
time=   3.3s
[CV] END max_depth=10, max_features=sqrt, n_estimators=100, random_state=0; total
time=   3.3s
[CV] END max_depth=10, max_features=sqrt, n_estimators=100, random_state=0; total
time=   4.3s
[CV] END max_depth=10, max_features=sqrt, n_estimators=100, random_state=42; total
time=   3.3s
[CV] END max_depth=10, max_features=sqrt, n_estimators=100, random_state=42; total
time=   3.3s
[CV] END max_depth=10, max_features=sqrt, n_estimators=100, random_state=42; total

```
time=   3.5s
[CV] END max_depth=10, max_features=sqrt, n_estimators=100, random_state=42; total
time=   3.9s
[CV] END max_depth=10, max_features=sqrt, n_estimators=100, random_state=42; total
time=   3.3s
[CV] END max_depth=10, max_features=sqrt, n_estimators=200, random_state=0; total
time=   7.6s
[CV] END max_depth=10, max_features=sqrt, n_estimators=200, random_state=0; total
time=   6.7s
[CV] END max_depth=10, max_features=sqrt, n_estimators=200, random_state=0; total
time=   7.6s
[CV] END max_depth=10, max_features=sqrt, n_estimators=200, random_state=0; total
time=   6.7s
[CV] END max_depth=10, max_features=sqrt, n_estimators=200, random_state=0; total
time=   7.6s
[CV] END max_depth=10, max_features=sqrt, n_estimators=200, random_state=42; total
time=   6.6s
[CV] END max_depth=10, max_features=sqrt, n_estimators=200, random_state=42; total
time=   7.4s
[CV] END max_depth=10, max_features=sqrt, n_estimators=200, random_state=42; total
time=   6.8s
[CV] END max_depth=10, max_features=sqrt, n_estimators=200, random_state=42; total
time=   7.1s
[CV] END max_depth=10, max_features=sqrt, n_estimators=200, random_state=42; total
time=   7.4s
[CV] END max_depth=10, max_features=log2, n_estimators=100, random_state=0; total
time=   3.3s
[CV] END max_depth=10, max_features=log2, n_estimators=100, random_state=0; total
time=   3.3s
[CV] END max_depth=10, max_features=log2, n_estimators=100, random_state=0; total
time=   3.3s
[CV] END max_depth=10, max_features=log2, n_estimators=100, random_state=0; total
time=   4.2s
[CV] END max_depth=10, max_features=log2, n_estimators=100, random_state=0; total
time=   3.3s
[CV] END max_depth=10, max_features=log2, n_estimators=100, random_state=42; total
time=   3.3s
[CV] END max_depth=10, max_features=log2, n_estimators=100, random_state=42; total
time=   4.1s
[CV] END max_depth=10, max_features=log2, n_estimators=100, random_state=42; total
time=   3.4s
[CV] END max_depth=10, max_features=log2, n_estimators=100, random_state=42; total
time=   3.3s
[CV] END max_depth=10, max_features=log2, n_estimators=100, random_state=42; total
time=   3.3s
[CV] END max_depth=10, max_features=log2, n_estimators=200, random_state=0; total
time=   7.5s
[CV] END max_depth=10, max_features=log2, n_estimators=200, random_state=0; total
time=   7.0s
[CV] END max_depth=10, max_features=log2, n_estimators=200, random_state=0; total
time=   7.1s
[CV] END max_depth=10, max_features=log2, n_estimators=200, random_state=0; total
time=   7.5s
[CV] END max_depth=10, max_features=log2, n_estimators=200, random_state=0; total
time=   6.5s
[CV] END max_depth=10, max_features=log2, n_estimators=200, random_state=42; total
time=   7.4s
[CV] END max_depth=10, max_features=log2, n_estimators=200, random_state=42; total
time=   6.5s
[CV] END max_depth=10, max_features=log2, n_estimators=200, random_state=42; total
time=   7.4s
[CV] END max_depth=10, max_features=log2, n_estimators=200, random_state=42; total
time=   6.5s
[CV] END max_depth=10, max_features=log2, n_estimators=200, random_state=42; total
time=   7.4s
[CV] END max_depth=10, max_features=None, n_estimators=100, random_state=0; total
time=  10.5s
[CV] END max_depth=10, max_features=None, n_estimators=100, random_state=0; total
time=  10.2s
[CV] END max_depth=10, max_features=None, n_estimators=100, random_state=0; total
time=  10.0s
[CV] END max_depth=10, max_features=None, n_estimators=100, random_state=0; total
time=  10.5s
[CV] END max_depth=10, max_features=None, n_estimators=100, random_state=0; total
time=  10.8s
[CV] END max_depth=10, max_features=None, n_estimators=100, random_state=42; total
time=  10.9s
[CV] END max_depth=10, max_features=None, n_estimators=100, random_state=42; total
time=  10.8s
[CV] END max_depth=10, max_features=None, n_estimators=100, random_state=42; total
time=   9.9s
[CV] END max_depth=10, max_features=None, n_estimators=100, random_state=42; total
time=  10.3s
[CV] END max_depth=10, max_features=None, n_estimators=100, random_state=42; total
time=  10.7s
[CV] END max_depth=10, max_features=None, n_estimators=200, random_state=0; total
time=  21.0s
[CV] END max_depth=10, max_features=None, n_estimators=200, random_state=0; total
time=  20.7s
[CV] END max_depth=10, max_features=None, n_estimators=200, random_state=0; total
time=  21.2s
[CV] END max_depth=10, max_features=None, n_estimators=200, random_state=0; total
time=  20.8s
[CV] END max_depth=10, max_features=None, n_estimators=200, random_state=0; total
time=  19.7s
[CV] END max_depth=10, max_features=None, n_estimators=200, random_state=42; total
time=  20.8s
[CV] END max_depth=10, max_features=None, n_estimators=200, random_state=42; total
time=  20.7s
[CV] END max_depth=10, max_features=None, n_estimators=200, random_state=42; total
time=  20.4s
[CV] END max_depth=10, max_features=None, n_estimators=200, random_state=42; total
time=  19.8s
[CV] END max_depth=10, max_features=None, n_estimators=200, random_state=42; total
time=  20.9s
```

Out[ ]:



In [ ]:
```
gscv.best_params_
```

Out[ ]:
```
{'max_depth': 10,
 'max_features': 'sqrt',
 'n_estimators': 100,
 'random_state': 0}
```

In [ ]:
```
rf=RandomForestClassifier(max_depth=10,max_features="sqrt",n_estimators=10,random_s
```

In [ ]:
```
rf.fit(x_train,y_train)
```

Out[ ]:

RandomForestClassifier

RandomForestClassifier(max_depth=10, n_estimators=10, random_state=42)

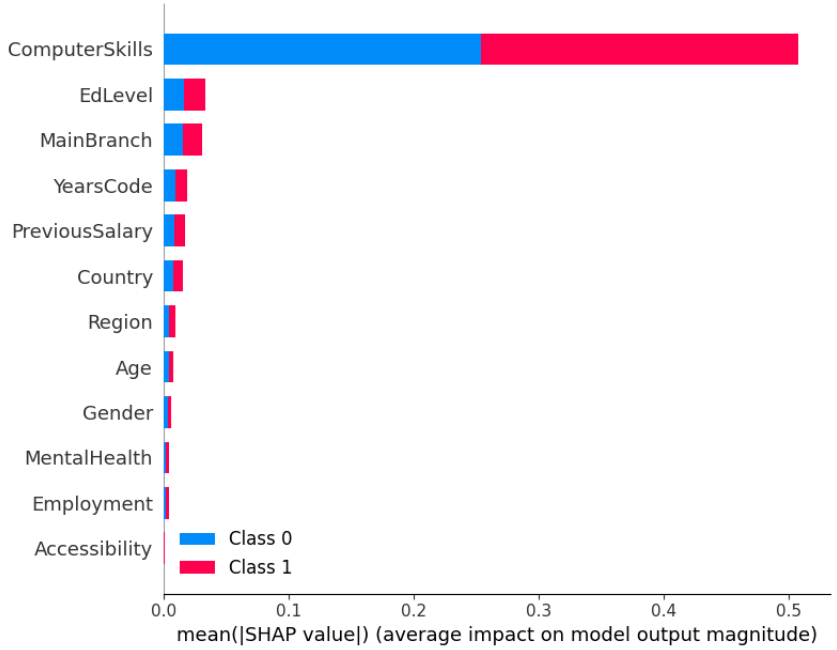
In [ ]:
```
y_pred2=rf.predict(x_test)
accuracy_score(y_test,y_pred2)
```

Out[ ]:
```
0.784591301980535
```

In [ ]:
```
from sklearn.metrics import accuracy_score,f1_score,precision_score,recall_score,lo

fscore=f1_score(y_test,y_pred2,average="micro")
pscore=precision_score(y_test,y_pred2,average="micro")
recallScore=recall_score(y_test,y_pred2,average="micro")
jscore=jaccard_score(y_test,y_pred2,average="micro")
logLoss=log_loss(y_test,y_pred2)


print("F1_Score",fscore)
print("Precision_score",pscore)
print("Recall Score",recallScore)
print("Jaccard_score",jscore)
print("Log loss",logLoss)
```

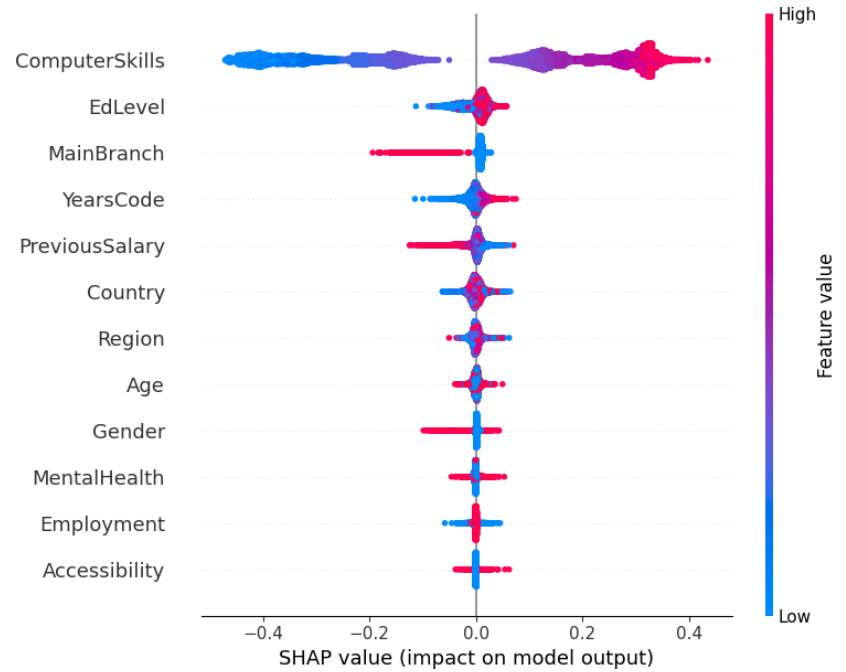
```
F1_Score 0.784591301980535
Precision_score 0.784591301980535
Recall Score 0.784591301980535
Jaccard_score 0.6455370142233173
Log loss 7.764116448414605
```

In [ ]:
```
import shap
explainer=shap.TreeExplainer(rf)
shap_values=explainer.shap_values(x_test)
shap.summary_plot(shap_values,x_test)
```



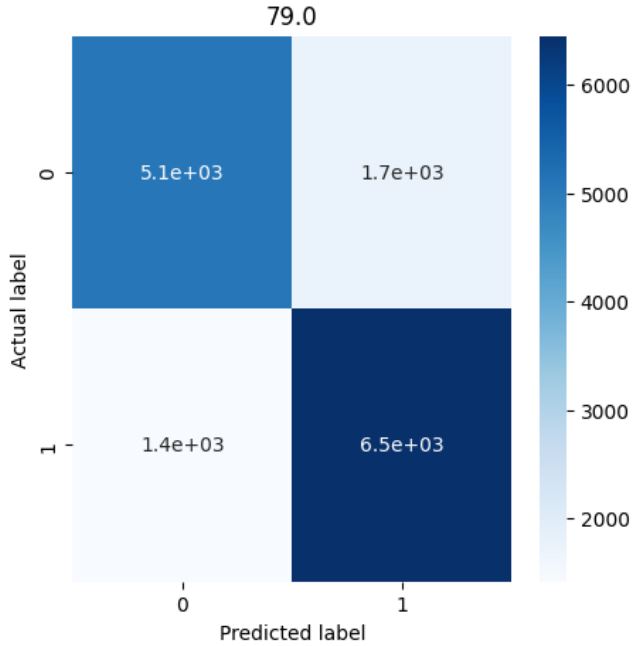
In [ ]:
```
#compute SHAP values
shap.summary_plot(shap_values[1],x_test.values,feature_names=x_test.columns)
```

```
No data for colormapping provided via 'c'. Parameters 'vmin', 'vmax' will be ignored
```



In [ ]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred2)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,annot=True,cmap="Blues")

plt.xlabel("Predicted label")
plt.ylabel("Actual label")
plt.title(accu_score*100)
```

Out[ ]:
```
Text(0.5, 1.0, '79.0')
```


79.0
5.1e+03
1.7e+03
1.4e+03
6.5e+03
0
1
Actual label
0
1
Predicted label
6000
5000
4000
3000
2000

In [ ]:

In [ ]: