In [6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [7]:
```
df=pd.read_csv("test_data.csv")
```

In [8]:
```
df.head()
```

Out[8]:

	ID	Gender	Has a car	Has a property	Children count	Income	Employment status	Education level	Marital status	Dwelling	Age	Employment length	Has a mobile phone	Has a work phone
0	5091261	F	N	Y	0	202500.0	State servant	Secondary / secondary special	Separated	House / apartment	-16834	-1692	1	0
1	5096963	M	Y	N	0	675000.0	Commercial associate	Higher education	Married	House / apartment	-18126	-948	1	0
2	5087880	F	N	N	0	234000.0	State servant	Higher education	Civil marriage	House / apartment	-21967	-5215	1	0
3	5021949	F	Y	Y	0	445500.0	Commercial associate	Higher education	Married	House / apartment	-12477	-456	1	0
4	5105705	F	Y	N	0	225000.0	Working	Secondary / secondary special	Married	Municipal apartment	-12155	-667	1	0

In [9]:
```
df.shape
```

Out[9]:
```
(7292, 20)
```

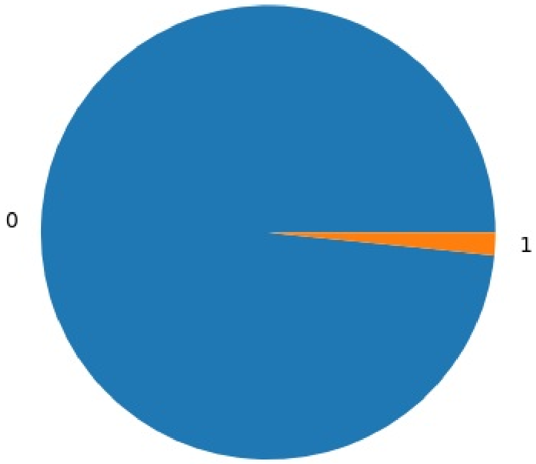
In [10]:
```
df.describe()
```

Out[10]:

	ID	Children count	Income	Age	Employment length	Has a mobile phone	Has a work phone	Has a phone	Has an email	Family member count
count	7.292000e+03	7292.000000	7.292000e+03	7292.000000	7292.000000	7292.0	7292.000000	7292.000000	7292.000000	7292.000000
mean	5.078209e+06	0.428415	1.858672e+05	-15957.958722	59283.630691	1.0	0.230389	0.294158	0.087493	2.202139
std	4.208243e+04	0.744350	1.032964e+05	4190.990010	137642.577749	0.0	0.421111	0.455695	0.282576	0.909726
min	5.008809e+06	0.000000	2.700000e+04	-25152.000000	-15661.000000	1.0	0.000000	0.000000	0.000000	1.000000
25%	5.041912e+06	0.000000	1.170000e+05	-19382.000000	-3141.000000	1.0	0.000000	0.000000	0.000000	2.000000
50%	5.069416e+06	0.000000	1.575000e+05	-15522.000000	-1534.000000	1.0	0.000000	0.000000	0.000000	2.000000
75%	5.115503e+06	1.000000	2.250000e+05	-12454.000000	-397.000000	1.0	0.000000	1.000000	0.000000	3.000000
max	5.150487e+06	14.000000	1.575000e+06	-7489.000000	365243.000000	1.0	1.000000	1.000000	1.000000	15.000000

Let's start by plotting the piechart for Is hish risk column

In [11]:
```
temp = df['Is high risk'].value_counts()
plt.pie(temp.values,
        labels=temp.index,
        )
plt.show()
```



In [15]:
```
pip install seaborn
```

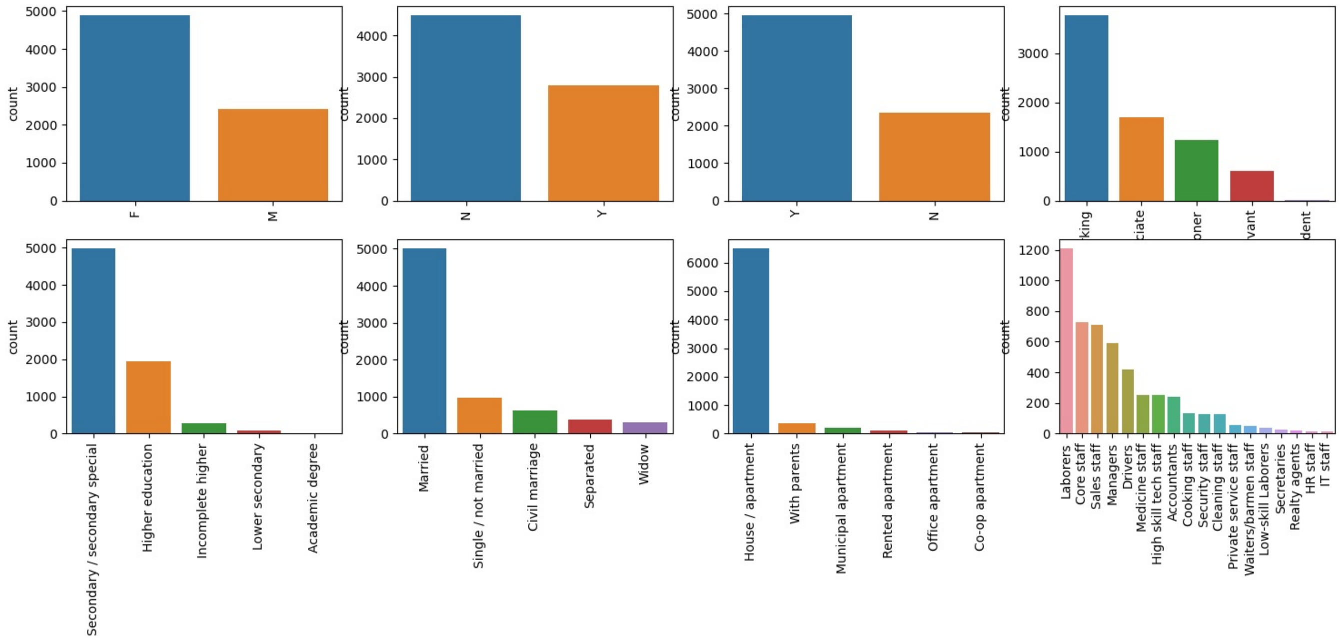
```
Requirement already satisfied: seaborn in c:\users\acer\anaconda3\lib\site-packages (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in c:\users\acer\anaconda3\lib\site-packages (from seaborn)
(1.24.3)
Requirement already satisfied: pandas>=0.25 in c:\users\acer\anaconda3\lib\site-packages (from seaborn) (2.0.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in c:\users\acer\anaconda3\lib\site-packages (from seabo
rn) (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\acer\anaconda3\lib\site-packages (from matplotlib!=
3.6.1,>=3.1->seaborn) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\acer\anaconda3\lib\site-packages (from matplotlib!=3.6.
1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\acer\anaconda3\lib\site-packages (from matplotlib!
=3.6.1,>=3.1->seaborn) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\acer\anaconda3\lib\site-packages (from matplotlib!
=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\acer\anaconda3\lib\site-packages (from matplotlib!=3
.6.1,>=3.1->seaborn) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\acer\anaconda3\lib\site-packages (from matplotlib!=3.6
.1,>=3.1->seaborn) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\acer\anaconda3\lib\site-packages (from matplot
lib!=3.6.1,>=3.1->seaborn) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\acer\anaconda3\lib\site-packages (from matplotl
ib!=3.6.1,>=3.1->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\acer\anaconda3\lib\site-packages (from pandas>=0.25->se
aborn) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\acer\anaconda3\lib\site-packages (from pandas>=0.25->
seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\acer\anaconda3\lib\site-packages (from python-dateutil>=2.7
->matplotlib!=3.6.1,>=3.1->seaborn) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [16]:
```
import seaborn as sns
```

As ID is completely unique and not correlated with any of the other column, So we will drop it using .drop() function.

In [18]:
```
obj = (df.dtypes == 'object')
object_cols = list(obj[obj].index)
plt.figure(figsize=(18,36))
index = 1

for col in object_cols:
  y = df[col].value_counts()
  plt.subplot(11,4,index)
  plt.xticks(rotation=90)
  sns.barplot(x=list(y.index), y=y)
  index +=1
```



In [19]:
```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how
# to understand word labels.
label_encoder = preprocessing.LabelEncoder()
obj = (df.dtypes == 'object')
for col in list(obj[obj].index):
  df[col] = label_encoder.fit_transform(df[col])
```

In [20]:
```
df
```

Out[20]:

	Gender	Has a car	Has a property	Children count	Income	Employment status	Education level	Marital status	Dwelling	Age	Employment length	Has a mobile phone	Has a work phone	Has a phone	H em
0	0	0	1	0	202500.0	2	4	2	1	-16834	-1692	1	0	0	
1	1	1	0	0	675000.0	0	1	1	1	-18126	-948	1	0	1	
2	0	0	0	0	234000.0	2	1	0	1	-21967	-5215	1	0	0	
3	0	1	1	0	445500.0	0	1	1	1	-12477	-456	1	0	0	
4	0	1	0	0	225000.0	4	4	1	2	-12155	-667	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7287	0	1	1	0	135000.0	4	4	1	1	-21724	-1351	1	0	0	
7288	0	1	1	0	157500.0	4	1	1	1	-14976	-3550	1	0	0	
7289	0	0	1	0	67500.0	4	4	4	1	-20482	-5030	1	1	1	
7290	0	1	0	0	95850.0	0	4	1	1	-18931	-6678	1	1	0	
7291	0	0	1	1	135000.0	0	4	0	1	-10765	-2196	1	0	0	

7292 rows × 19 columns

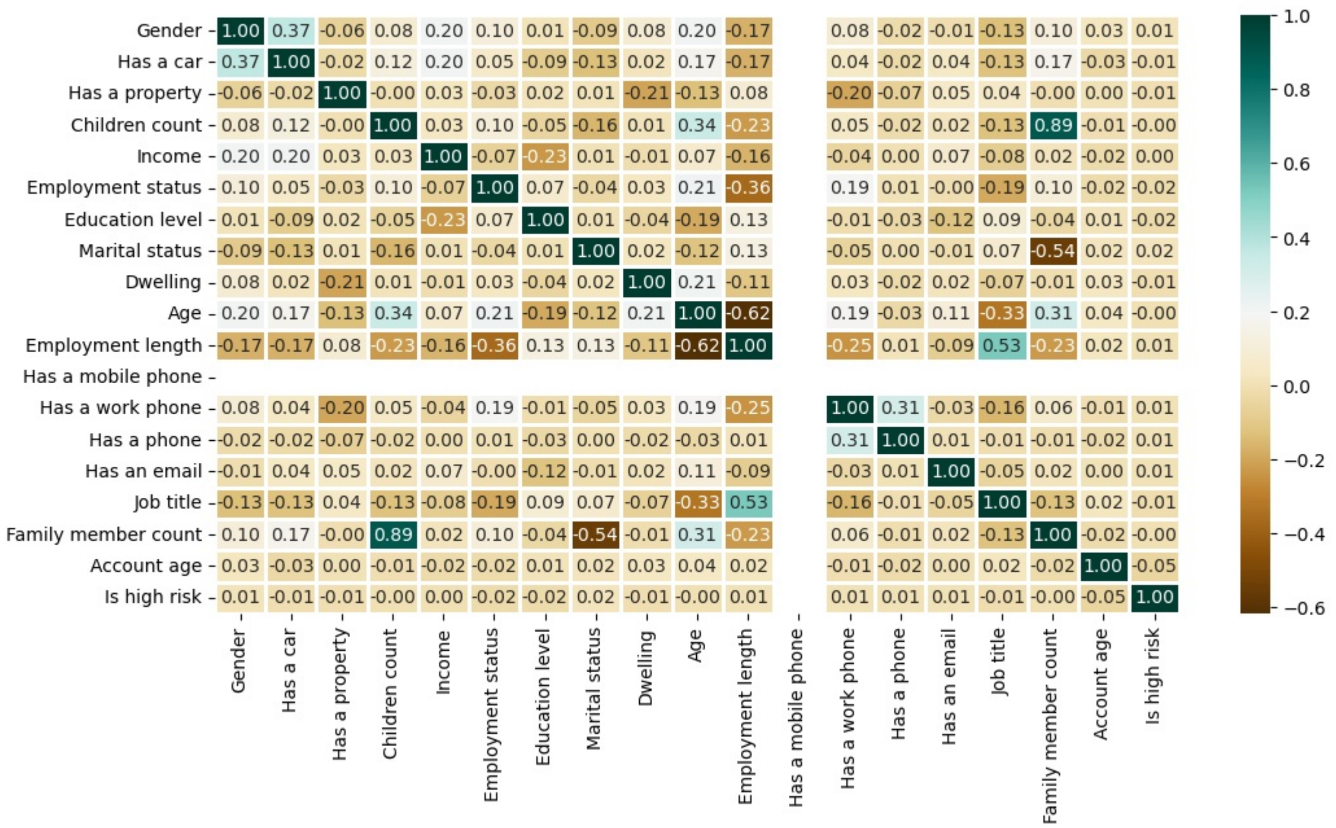
In [21]:
```
# To find the number of columns with
# datatype==object
obj = (df.dtypes == 'object')
print("Categorical variables:",len(list(obj[obj].index)))
```

```
Categorical variables: 0
```

In [22]:
```
plt.figure(figsize=(12,6))

sns.heatmap(df.corr(),cmap='BrBG',fmt='.2f',
            linewidths=2,annot=True)
```

Out[22]:
```
<Axes: >
```



```
In [23]: for col in df.columns:
           df[col] = df[col].fillna(df[col].mean())

         df.isna().sum()
```

```
Out[23]: Gender                  0
         Has a car               0
         Has a property          0
         Children count          0
         Income                  0
         Employment status       0
         Education level         0
         Marital status          0
         Dwelling                0
         Age                     0
         Employment length       0
         Has a mobile phone      0
         Has a work phone        0
         Has a phone             0
         Has an email            0
         Job title               0
         Family member count     0
         Account age             0
         Is high risk            0
         dtype: int64
```

Splitting Dataset

```
In [76]: from sklearn.model_selection import train_test_split

         X = df.iloc[:,1:-1].values
         y = df.iloc[:,18].values
         X.shape,y.shape

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)
         X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
Out[76]: ((4375, 17), (2917, 17), (4375,), (2917,))
```

```
In [77]: print(X_train)
```

```
[[  0.   1.   0. ...  16.   2. -23.]
 [  0.   0.   0. ...  18.   2. -15.]
 [  0.   1.   0. ...   0.   2. -16.]
 ...
 [  1.   0.   2. ...  18.   4. -49.]
 [  0.   1.   1. ...  17.   3.  -9.]
 [  1.   1.   0. ...   8.   2. -13.]]
```

In [78]:
```
from sklearn.tree import DecisionTreeClassifier

from sklearn.linear_model import LogisticRegression

from sklearn import metrics

dtc = DecisionTreeClassifier(criterion = 'entropy', max_depth =3)
lc = LogisticRegression()
```

In [79]:
```
# making predictions on the training set

for clf in (dtc,lc):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_train)
    print("Accuracy score of ", clf.__class__.__name__,"=",100*metrics.accuracy_score(y_train, y_pred))
```

```
Accuracy score of  DecisionTreeClassifier = 98.28571428571429
Accuracy score of  LogisticRegression = 98.24000000000001
```

In [80]:
```
# making predictions on the testing set
for clf in (dtc,lc):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print("Accuracy score of ", clf.__class__.__name__,"=", 100*metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy score of  DecisionTreeClassifier = 98.59444634898868
Accuracy score of  LogisticRegression = 98.62872814535481
```

In [81]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
print(confusion_matrix(y_test,y_pred))
```

```
[[2877    0]
 [  40    0]]
```

In [82]:
```
#Let see the Decision Tree
from sklearn import tree
```

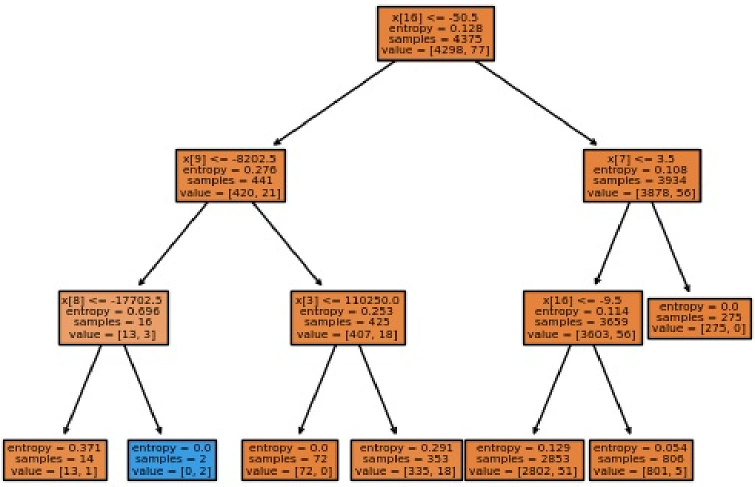
In [83]:
```
plt.figure(figsize=(20,10))
```

Out[83]:
```
<Figure size 2000x1000 with 0 Axes>

<Figure size 2000x1000 with 0 Axes>
```

In [84]:
```
tree.plot_tree(dtc,filled=True)
```

Out[84]:
```
[Text(0.5769230769230769, 0.875, 'x[16] <= -50.5\nentropy = 0.128\nsamples = 4375\nvalue = [4298, 77]'),
 Text(0.3076923076923077, 0.625, 'x[9] <= -8202.5\nentropy = 0.276\nsamples = 441\nvalue = [420, 21]'),
 Text(0.15384615384615385, 0.375, 'x[8] <= -17702.5\nentropy = 0.696\nsamples = 16\nvalue = [13, 3]'),
 Text(0.07692307692307693, 0.125, 'entropy = 0.371\nsamples = 14\nvalue = [13, 1]'),
 Text(0.23076923076923078, 0.125, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.46153846153846156, 0.375, 'x[3] <= 110250.0\nentropy = 0.253\nsamples = 425\nvalue = [407, 18]'),
 Text(0.38461538461538464, 0.125, 'entropy = 0.0\nsamples = 72\nvalue = [72, 0]'),
 Text(0.5384615384615384, 0.125, 'entropy = 0.291\nsamples = 353\nvalue = [335, 18]'),
 Text(0.8461538461538461, 0.625, 'x[7] <= 3.5\nentropy = 0.108\nsamples = 3934\nvalue = [3878, 56]'),
 Text(0.7692307692307693, 0.375, 'x[16] <= -9.5\nentropy = 0.114\nsamples = 3659\nvalue = [3603, 56]'),
 Text(0.6923076923076923, 0.125, 'entropy = 0.129\nsamples = 2853\nvalue = [2802, 51]'),
 Text(0.8461538461538461, 0.125, 'entropy = 0.054\nsamples = 806\nvalue = [801, 5]'),
 Text(0.9230769230769231, 0.375, 'entropy = 0.0\nsamples = 275\nvalue = [275, 0]')]
```


x[16] <= -50.5
entropy = 0.128
samples = 4375
value = [4298, 77]
x[9] <= -8202.5
entropy = 0.276
samples = 441
value = [420, 21]
x[7] <= 3.5
entropy = 0.108
samples = 3934
value = [3878, 56]
x[8] <= -17702.5
entropy = 0.696
samples = 16
value = [13, 3]
x[3] <= 110250.0
entropy = 0.253
samples = 425
value = [407, 18]
x[16] <= -9.5
entropy = 0.114
samples = 3659
value = [3603, 56]
entropy = 0.0
samples = 275
value = [275, 0]
entropy = 0.371
samples = 14
value = [13, 1]
entropy = 0.0
samples = 2
value = [0, 2]
entropy = 0.0
samples = 72
value = [72, 0]
entropy = 0.291
samples = 353
value = [335, 18]
entropy = 0.129
samples = 2853
value = [2802, 51]
entropy = 0.054
samples = 806
value = [801, 5]