In [34]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

In [35]:
```
base_census = pd.read_csv(r'C:\Users\Master\Desktop\Jupyter\IA UDEMY\census.csv')
base_census
```

Out[35]:

	age	workclass	final-weight	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loos	hour-per-week	native-country	income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	>50K
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	United-States	>50K

32561 rows × 15 columns

In [36]:
```
base_census.describe()
```

Out[36]:

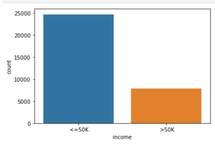
	age	final-weight	education-num	capital-gain	capital-loos	hour-per-week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

In [37]:
```
base_census.isnull().sum()# verificando se ha dados faltantes
```

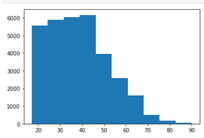
Out[37]:
```
age             0
workclass       0
final-weight    0
education       0
education-num   0
marital-status  0
occupation      0
relationship    0
race            0
sex             0
capital-gain    0
capital-loos    0
hour-per-week   0
native-country  0
income          0
dtype: int64
```

In [38]:
```
np.unique(base_census['income'], return_counts=True);
```

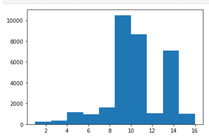
In [39]:
```
sns.countplot(x = base_census['income']);
```



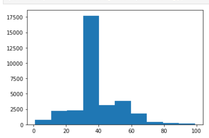
In [41]:
```
plt.hist(x = base_census['age']);
```



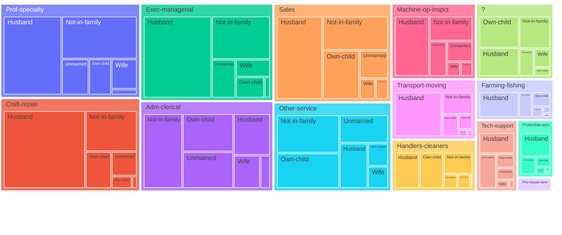
In [42]:
```
plt.hist(x = base_census['education-num']);
```



In [43]:
```
plt.hist(x = base_census['hour-per-week']);
```



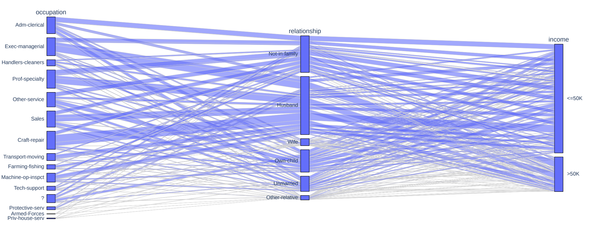
In [16]:
```
grafico = px.treemap(base_census,path=['occupation','relationship'])
grafico.show()
```



In [ ]:
```

```

In [44]:
```
grafico = px.parallel_categories(base_census,dimensions=['occupation','relationship','income'])
grafico.show()
```



#grafico = px.parallel_categories(base_census.dimensions=['workclass','occupation','income']) #grafico.show()

In [45]:
```
x_census = base_census.iloc[:,0:14].values
y_census = base_census.iloc[:, 14].values
```

In [10]:
```
#label Encoder = serve para deixar os numeros proporcionais
from sklearn.preprocessing import LabelEncoder
```

In [49]:
```
label_encoder_workclass = LabelEncoder()
label_encoder_education= LabelEncoder()
label_encoder_marital = LabelEncoder()
label_encoder_occupation=LabelEncoder()
label_encoder_relationship= LabelEncoder()
label_encoder_race = LabelEncoder()
label_encoder_sex = LabelEncoder()
label_encoder_country = LabelEncoder()

x_census[:,1 ] = label_encoder_workclass.fit_transform(x_census[:,1])
x_census[:,3 ] = label_encoder_education.fit_transform(x_census[:,3])
x_census[:,5 ] = label_encoder_marital.fit_transform(x_census[:,5])
x_census[:,6 ] = label_encoder_occupation.fit_transform(x_census[:,6])
x_census[:,7 ] = label_encoder_relationship.fit_transform(x_census[:,7])
x_census[:,8 ] = label_encoder_race.fit_transform(x_census[:,8])
x_census[:,9 ] = label_encoder_sex.fit_transform(x_census[:,9])
x_census[:,13] = label_encoder_country.fit_transform(x_census[:,13])
```

In [58]:
```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

onehotencoder_census = ColumnTransformer(transformers=[('OneHot',OneHotEncoder(), [1,3,5,7,8,9,13])],remainder='passthrough')
x_census = onehotencoder_census.fit_transform(x_census).toarray() #toarray() transforma para o numpyarray
x_census
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-58-f30d243a7691> in <module>
      3
      4 onehotencoder_census = ColumnTransformer(transformers=[('OneHot',OneHotEncoder(), [1,3,5,7,8,9,13])],remainder='passthrough')
----> 5 x_census = onehotencoder_census.fit_transform(x_census).toarray() #toarray() transforma para o numpyarray
      6 x_census

AttributeError: 'numpy.ndarray' object has no attribute 'toarray'
```

In [2]:
```
jupyter nbconvert --to html NOTEBOOK-NAME.ipynb
```

```
  File "<ipython-input-2-a908f243614a>", line 1
    jupyter nbconvert --to html NOTEBOOK-NAME.ipynb
            ^
SyntaxError: invalid syntax
```

In [ ]:
```

```

In [ ]:
```

```