In [1]:

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

In [2]:

```
dataset = pd.read_csv("C:/Users/BHARATH/Downloads/sample_images/census.csv")
```

In [3]:

```
dataset.head(15)
```

Out[3]:

	House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Rel
0	1	Rhubarb Drive	Amy	Hall	41	Head	Single	Female	Drilling engineer	NaN	Ca
1	1	Rhubarb Drive	Bruce	Murphy	49	Partner	Single	Male	Scientist, research (physical sciences)	NaN	Ca
2	1	Rhubarb Drive	Billy	Hall	17	Son	NaN	Male	Student	NaN	
3	1	Rhubarb Drive	Mary	Hall	15	Daughter	NaN	Female	Student	NaN	
4	1	Rhubarb Drive	Elliott	Hall	13	Son	NaN	Male	Student	NaN	
5	1	Rhubarb Drive	Clifford	Hall	7	Son	NaN	Male	Student	NaN	
6	1	Rhubarb Drive	Kirsty	Hall	6	Daughter	NaN	Female	Student	NaN	
7	1	Rhubarb Drive	Nathan	Hall	5	Son	NaN	Male	Student	NaN	
8	1	Rhubarb Drive	Sophie	Hall	4	Daughter	NaN	Female	Child	NaN	
9	1	Rhubarb Drive	Sarah	Hall	2	Daughter	NaN	Female	Child	NaN	
10	2	Rhubarb Drive	Terry	Doherty	58	Head	Single	Male	Arts administrator	NaN	Chr
11	2	Rhubarb Drive	Raymond	Carter	33	NaN	Single	Male	Electrical engineer	NaN	
12	2	Rhubarb Drive	Ryan	Farrell	26	NaN	Single	Male	Psychologist, occupational	NaN	
13	2	Rhubarb Drive	Bethany	Jones	36	NaN	Single	Female	Surveyor, land/geomatics	NaN	Ca
14	2	Rhubarb Drive	Kieran	Wood	47	NaN	Single	Male	Programme researcher, broadcasting/film/video	NaN	Ca

In [4]:

```
dataset.columns = ['House_Number', 'Street', 'First_Name', 'Surname', 'Age',
       'Relationship_to_Head_of_House', 'Marital_Status', 'Gender',
       'Occupation', 'Infirmity', 'Religion']
```

In [5]:

```
#converting columns like House_Number, Age to numeric type from string type
try:
    dataset["Age"] = pd.to_numeric(dataset["Age"])
except:
    print("Some values in Age column can not be converted to numeric values")
    print(dataset.loc[dataset.Age.str.isnumeric() == False, "Age"])
    ind = dataset.loc[dataset.Age.str.isnumeric() == False, "Age"].index

try:
    dataset["House_Number"] = pd.to_numeric(dataset["House_Number"])
except:
    print("some values in House_Number column can not be converted to numeric values")
```

```
Some values in Age column can not be converted to numeric values
2774    95.20845345950883
2775    97.20845345950883
2776    46.99999999999999
4189    50.68642589738663
4190    50.68642589738663
4191                  0.0
4192                  0.0
5038    71.70511965460919
5039    74.70511965460919
5040                 13.0
5284     84.3600933267044
6819
8005    78.78623363918231
8006                 30.0
8007                 27.0
8008                 24.0
9422
Name: Age, dtype: object
```

In [6]:

```
dataset["Age"] = dataset["Age"].astype("int64", errors = 'ignore')
dataset.iloc[ind, : ]
```

Out[6]:

	House_Number	Street	First_Name	Surname	Age	Relationship_to_Head_of_House	Mar
2774	7	Orchard Crescent	Kim	Mitchell	95.20845345950883	Head	
2775	7	Orchard Crescent	Antony	Mitchell	97.20845345950883	Husband	
2776	7	Orchard Crescent	Leigh	Mitchell	46.99999999999999	Son	
4189	26	Kelly Shoals	Claire	Khan	50.68642589738663	Head	
4190	26	Kelly Shoals	Douglas	Khan	50.68642589738663	Husband	
4191	26	Kelly Shoals	Christian	Khan	0.0	Son	
4192	26	Kelly Shoals	Geraldine	Khan	0.0	Daughter	
5038	33	Brightonshot Lane	Emily	Manning	71.70511965460919	Head	
5039	33	Brightonshot Lane	Graham	Manning	74.70511965460919	Husband	
5040	33	Brightonshot Lane	Irene	Manning	13.0	Daughter	
5284	25	Smith Mountain	Janet	Storey	84.3600933267044	Head	
6819	33	Bank Street	Georgia	Poole		Head	
8005	19	St.Luke Locks	Joanne	Price	78.78623363918231	Head	
8006	19	St.Luke Locks	Aimee	Price	30.0	Daughter	
8007	19	St.Luke Locks	Jade	Price	27.0	Daughter	
8008	19	St.Luke Locks	Gregory	Price	24.0	Son	
9422	32	Bank Burg	Mitchell	Jordan		Partner	

In [7]:

```
for i in ind:
    name = dataset.First_Name[i]
    gender = dataset.Gender[i]
    ocp = dataset.Occupation[i]
    if len(pd.Series.mode(dataset["Age"].loc[(dataset.Occupation == ocp) & (dataset.Gender == gender) & (
        dataset.at[i, "Age"] = pd.Series.mode(dataset["Age"].loc[(dataset.Occupation == ocp) & (dataset.G
    else:
        dataset.at[i,"Age"] = 80
dataset["Age"] = pd.to_numeric(dataset["Age"])
```

In [8]:

```
dataset.iloc[ind, : ]
```

Out[8]:

	House_Number	Street	First_Name	Surname	Age	Relationship_to_Head_of_House	Marital_Status	Ge
2774	7	Orchard Crescent	Kim	Mitchell	80	Head	Married	Fe
2775	7	Orchard Crescent	Antony	Mitchell	79	Husband	Married	
2776	7	Orchard Crescent	Leigh	Mitchell	28	Son	Single	
4189	26	Kelly Shoals	Claire	Khan	24	Head	Married	Fe
4190	26	Kelly Shoals	Douglas	Khan	42	Husband	Married	
4191	26	Kelly Shoals	Christian	Khan	3	Son	NaN	
4192	26	Kelly Shoals	Geraldine	Khan	2	Daughter	NaN	Fe
5038	33	Brightonshot Lane	Emily	Manning	80	Head	Married	Fe
5039	33	Brightonshot Lane	Graham	Manning	80	Husband	Married	
5040	33	Brightonshot Lane	Irene	Manning	9	Daughter	NaN	Fe
5284	25	Smith Mountain	Janet	Storey	80	Head	Widowed	Fe
6819	33	Bank Street	Georgia	Poole	80	Head	Married	Fe
8005	19	St.Luke Locks	Joanne	Price	73	Head	Divorced	Fe
8006	19	St.Luke Locks	Aimee	Price	29	Daughter	Single	Fe
8007	19	St.Luke Locks	Jade	Price	31	Daughter	Divorced	Fe
8008	19	St.Luke Locks	Gregory	Price	27	Son	Single	
9422	32	Bank Burg	Mitchell	Jordan	40	Partner	Single	

In [9]:

```
#getting number of missing data poijnts in each column
dataset.isnull().sum()
```

Out[9]:

```
House_Number                        0
Street                              0
First_Name                          0
Surname                             0
Age                                 0
Relationship_to_Head_of_House     722
Marital_Status                   2273
Gender                              0
Occupation                          0
Infirmity                        9657
Religion                         5561
dtype: int64
```

In [10]:

```
#Handling missing values
# Relationship to Head of House column missing values can be handled
# by some assumptions
# 1. If there is no the other person with same surname with in the same house then we fill the column valu
index_of_rows_with_NONE_Relationship_to_head = dataset.loc[dataset.Relationship_to_Head_of_House.isnull()
for i in index_of_rows_with_NONE_Relationship_to_head:
    house_no, street, surname = dataset.loc[i, ["House_Number", "Street", "Surname"]]
    req = dataset.loc[(dataset.House_Number == house_no) & (dataset.Street == street) & (dataset.Surname =
    if(req.shape[0] == 1):
        dataset.loc[i, "Relationship_to_Head_of_House"] = "Head"
```

In [11]:

```
dataset["Relationship_to_Head_of_House"].isnull().sum()
```

Out[11]:

```
15
```

In [12]:

```
dataset.isnull().sum()
```

Out[12]:

```
House_Number                        0
Street                              0
First_Name                          0
Surname                             0
Age                                 0
Relationship_to_Head_of_House      15
Marital_Status                   2273
Gender                              0
Occupation                          0
Infirmity                        9657
Religion                         5561
dtype: int64
```

In [13]:

```
dataset.loc[(dataset.Marital_Status.isnull() == True) & (dataset.Age <= 18), "Marital_Status"] = "None"
```

In [14]:

```
dataset.isnull().sum()
```

Out[14]:

```
House_Number                       0
Street                             0
First_Name                         0
Surname                            0
Age                                0
Relationship_to_Head_of_House     15
Marital_Status                     0
Gender                             0
Occupation                         0
Infirmity                       9657
Religion                        5561
dtype: int64
```

In [15]:

```
ind = dataset.loc[(dataset.Religion.isnull() == True) & (dataset.Age < 18)].index
for i in ind:
    surn = dataset.Surname[i]
    street = dataset.Street[i]
    hno = dataset.House_Number[i]
    if (len(dataset.loc[(dataset.Age > 18) & (dataset.Surname == surn) & (dataset.Street == street) & (da
        dataset.at[i,"Religion"] = (dataset.loc[(dataset.Age > 18) & (dataset.Surname == surn) & (dataset
```

In [16]:

```
dataset.Religion = dataset.Religion.fillna(value = "None")
```

In [17]:

```
dataset.iloc[ind, : ]
```

Out[17]:

	House_Number	Street	First_Name	Surname	Age	Relationship_to_Head_of_House	Marital_Status	Gende
2	1	Rhubarb Drive	Billy	Hall	17	Son	None	Mal
3	1	Rhubarb Drive	Mary	Hall	15	Daughter	None	Femal
4	1	Rhubarb Drive	Elliott	Hall	13	Son	None	Mal
5	1	Rhubarb Drive	Clifford	Hall	7	Son	None	Mal
6	1	Rhubarb Drive	Kirsty	Hall	6	Daughter	None	Femal
...	...	...	...	...	...	...	...	.
9707	1	Beech Villa	Jacob	King	7	Son	None	Mal
9708	1	Beech Villa	Brenda	Brown-King	5	Daughter	None	Femal
9709	1	Beech Villa	Laura	Brown-King	1	Daughter	None	Femal
9726	1	Taylor Barracks	Rita	Turner-Rose	14	Daughter	None	Femal
9727	1	Taylor Barracks	Terry	Turner-Rose	12	Son	None	Mal

2278 rows × 11 columns

In [18]:

```
dataset.isnull().sum()
```

Out[18]:

```
House_Number                      0
Street                            0
First_Name                        0
Surname                           0
Age                               0
Relationship_to_Head_of_House    15
Marital_Status                    0
Gender                            0
Occupation                        0
Infirmity                      9657
Religion                          0
dtype: int64
```

In [19]:

```
dataset["Age_band"] = [str(((age // 5))* 5) + "-" + str(((age // 5) + 1)* 5) for age in dataset["Age"]]
```

In [20]:

```
dataset.head()
```

Out[20]:

	House_Number	Street	First_Name	Surname	Age	Relationship_to_Head_of_House	Marital_Status	Gender	(
0	1	Rhubarb Drive	Amy	Hall	41	Head	Single	Female	
1	1	Rhubarb Drive	Bruce	Murphy	49	Partner	Single	Male	
2	1	Rhubarb Drive	Billy	Hall	17	Son	None	Male	
3	1	Rhubarb Drive	Mary	Hall	15	Daughter	None	Female	
4	1	Rhubarb Drive	Elliott	Hall	13	Son	None	Male	

In [21]:

```
cat = ["Student", "Child", "Retired", "Unemployed"]
def categ(val, cat):
    req = "Employed"
    for i in cat:
        if(i in val):
            req = i
    return req
dataset["Employement_Category"] = [categ(i, cat) for i in dataset.Occupation.values]
```

In [22]:

```
dataset.head()
```

Out[22]:

	House_Number	Street	First_Name	Surname	Age	Relationship_to_Head_of_House	Marital_Status	Gender	(
0	1	Rhubarb Drive	Amy	Hall	41	Head	Single	Female	
1	1	Rhubarb Drive	Bruce	Murphy	49	Partner	Single	Male	
2	1	Rhubarb Drive	Billy	Hall	17	Son	None	Male	
3	1	Rhubarb Drive	Mary	Hall	15	Daughter	None	Female	
4	1	Rhubarb Drive	Elliott	Hall	13	Son	None	Male	

In [23]:

```
def hhn(i):
    sn = dataset.Surname[i]
    hn = dataset.House_Number[i]
    st = dataset.Street[i]
    return len(dataset.loc[(dataset.House_Number == hn) & (dataset.Surname == sn) & (dataset.Street == st
dataset["Household_Occupancy"] = [hhn(i) for i in range(len(dataset))]
```

In [24]:

```
dataset.head(20)
```

Out[24]:

	House_Number	Street	First_Name	Surname	Age	Relationship_to_Head_of_House	Marital_Status	Gender
0	1	Rhubarb Drive	Amy	Hall	41	Head	Single	Female
1	1	Rhubarb Drive	Bruce	Murphy	49	Partner	Single	Male
2	1	Rhubarb Drive	Billy	Hall	17	Son	None	Male
3	1	Rhubarb Drive	Mary	Hall	15	Daughter	None	Female
4	1	Rhubarb Drive	Elliott	Hall	13	Son	None	Male
5	1	Rhubarb Drive	Clifford	Hall	7	Son	None	Male
6	1	Rhubarb Drive	Kirsty	Hall	6	Daughter	None	Female
7	1	Rhubarb Drive	Nathan	Hall	5	Son	None	Male
8	1	Rhubarb Drive	Sophie	Hall	4	Daughter	None	Female
9	1	Rhubarb Drive	Sarah	Hall	2	Daughter	None	Female
10	2	Rhubarb Drive	Terry	Doherty	58	Head	Single	Male
11	2	Rhubarb Drive	Raymond	Carter	33	Head	Single	Male
12	2	Rhubarb Drive	Ryan	Farrell	26	Head	Single	Male
13	2	Rhubarb Drive	Bethany	Jones	36	Head	Single	Female
14	2	Rhubarb Drive	Kieran	Wood	47	Head	Single	Male
15	3	Rhubarb Drive	Holly	Greenwood	47	Head	Married	Female
16	3	Rhubarb Drive	Tom	Greenwood	51	Husband	Married	Male
17	3	Rhubarb Drive	Scott	Greenwood	19	Son	Single	Male
18	3	Rhubarb Drive	Jonathan	Greenwood	17	Son	None	Male
19	3	Rhubarb Drive	Marian	Greenwood	17	Daughter	None	Female

In [25]:

```
dataset.loc[dataset.Employement_Category == "Employed"]
```

Out[25]:

	House_Number	Street	First_Name	Surname	Age	Relationship_to_Head_of_House	Marital_Status	Gend
0	1	Rhubarb Drive	Amy	Hall	41	Head	Single	Fema
1	1	Rhubarb Drive	Bruce	Murphy	49	Partner	Single	Ma
10	2	Rhubarb Drive	Terry	Doherty	58	Head	Single	Ma
11	2	Rhubarb Drive	Raymond	Carter	33	Head	Single	Ma
12	2	Rhubarb Drive	Ryan	Farrell	26	Head	Single	Ma
...	...	...	...	...	...	...	...	
9720	1	Stephens Rectory	Angela	Mitchell	39	Daughter	Single	Fema
9721	1	Stephens Rectory	Sharon	Mitchell	39	Daughter	Single	Fema
9722	1	Stephens Rectory	Hollie	Mitchell	36	Daughter	Single	Fema
9723	1	Taylor Barracks	Anne	Turner	42	Head	Married	Fema
9724	1	Taylor Barracks	Graham	Turner-Rose	44	Husband	Married	Ma

5271 rows × 14 columns

In [26]:

```
dataset.Occupation.value_counts()
```

Out[26]:

```
Occupation
Student                                          1871
Unemployed                                        593
University Student                                571
Child                                             537
PhD Student                                        20
                                                 ...
Retired Scientist, water quality                    1
Retired Nurse, learning disability                  1
Retired Energy manager                              1
Retired English as a second language teacher        1
Retired Research scientist (life sciences)          1
Name: count, Length: 1113, dtype: int64
```

In [27]:

```
dataset.Employement_Category.value_counts()
```

Out[27]:

```
Employement_Category
Employed      5271
Student       2462
Retired        852
Unemployed     593
Child          550
Name: count, dtype: int64
```

In [28]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9728 entries, 0 to 9727
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype 
---  ------                          --------------  ----- 
 0   House_Number                    9728 non-null   int64 
 1   Street                          9728 non-null   object
 2   First_Name                      9728 non-null   object
 3   Surname                         9728 non-null   object
 4   Age                             9728 non-null   int64 
 5   Relationship_to_Head_of_House   9713 non-null   object
 6   Marital_Status                  9728 non-null   object
 7   Gender                          9728 non-null   object
 8   Occupation                      9728 non-null   object
 9   Infirmity                       71 non-null     object
 10  Religion                        9728 non-null   object
 11  Age_band                        9728 non-null   object
 12  Employement_Category            9728 non-null   object
 13  Household_Occupancy             9728 non-null   int64 
dtypes: int64(3), object(11)
memory usage: 1.0+ MB
```

In [29]:

```
male = dataset.loc[dataset.Gender == "Male"].Age_band.value_counts()
female = dataset.loc[dataset.Gender == "Female"].Age_band.value_counts()
ages = pd.DataFrame()
ages["bands"] = [str((i - 1) * 5) + "-" + str(i * 5) for i in range(1,23)]
ages["Male"] = [male[ages["bands"][i]] for i in range(len(ages))]
ages["Female"] = [female[ages["bands"][i]] for i in range(len(ages))]
ages.head()
```

Out[29]:

	bands	Male	Female
0	0-5	275	262
1	5-10	356	323
2	10-15	360	309
3	15-20	378	343
4	20-25	323	325

In [30]:

```
len(ages)
```

Out[30]:

```
22
```

In [31]:

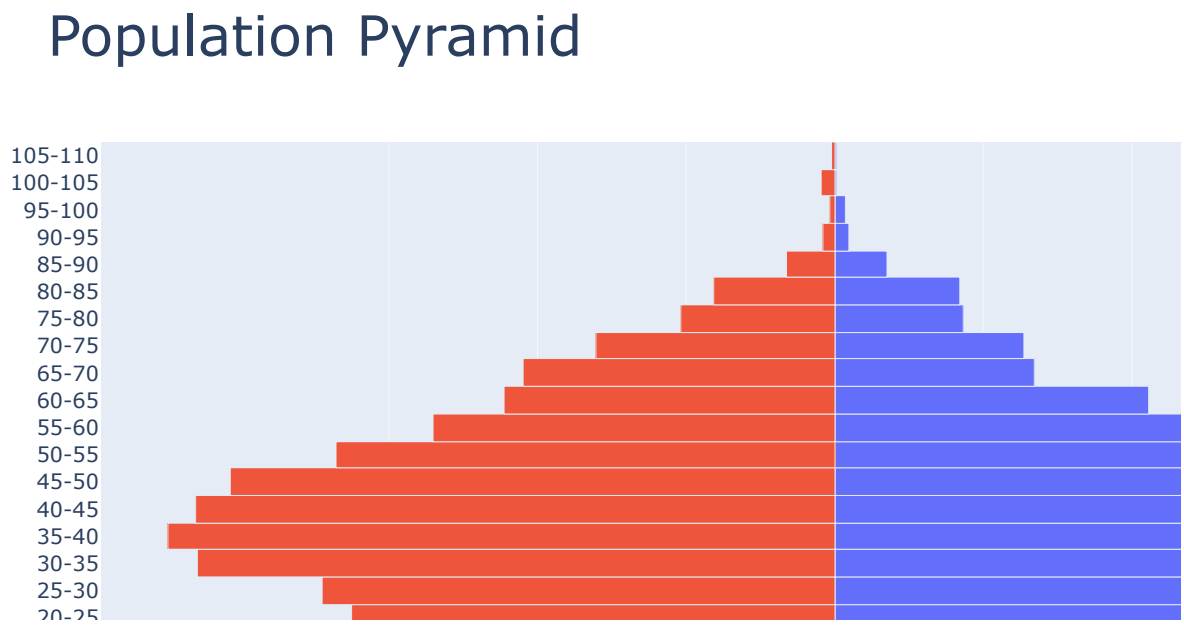
```
import plotly.graph_objects as gp
```

In [32]:

```
y_age = ages['bands']
x_M = ages["Male"]
x_F = ages['Female'] * -1
```

In [33]:

```
# Creating instance of the figure
fig = gp.Figure()

# Adding Male data to the figure
fig.add_trace(gp.Bar(y= y_age, x = x_M,
                     name = 'Male',
                     orientation = 'h'))

# Adding Female data to the figure
fig.add_trace(gp.Bar(y = y_age, x = x_F,
                     name = 'Female', orientation = 'h'))

# Updating the layout for our graph
fig.update_layout(title = 'Population Pyramid',
                  title_font_size = 32, barmode = 'relative',
                  bargap = 0.0, bargroupgap = 0.0,
                  xaxis = dict(tickvals = [-300, -200, -100,
                                           0, 100, 200, 300],

                               ticktext = ['300', '200', '100', '0',
                                           '100', '200', '300'],

                               title = 'Population',
                               title_font_size = 24)
                  )

fig.show()
```

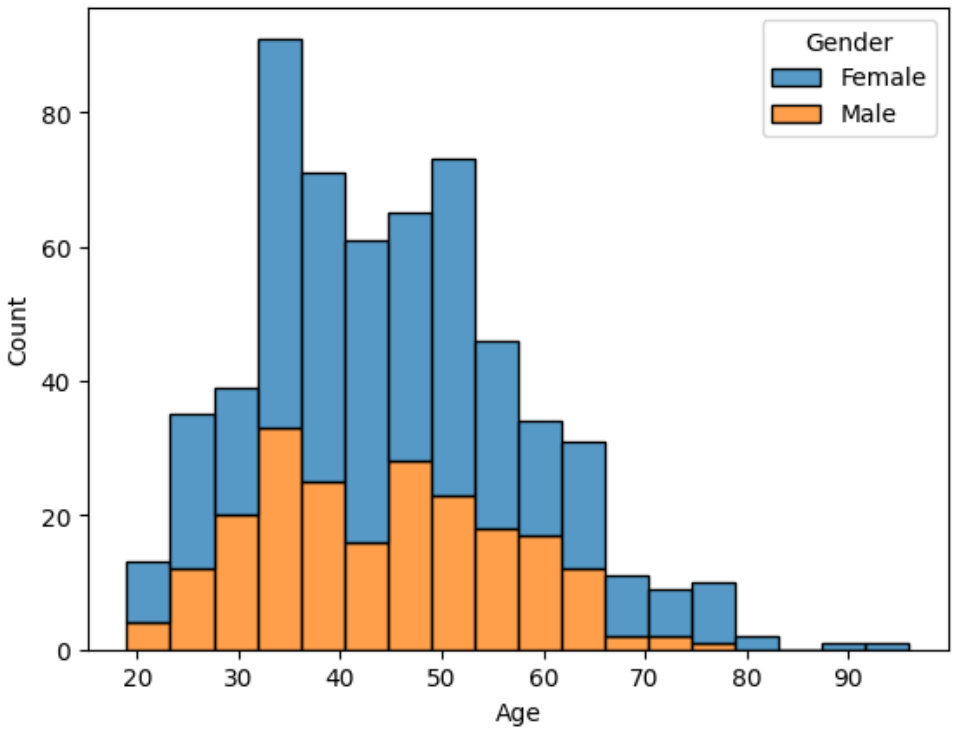


In [34]:

```
sns.histplot(data = dataset.loc[dataset.Employement_Category == "Unemployed",["Employement_Category", "Ag
```

Out[34]:

```
<Axes: xlabel='Age', ylabel='Count'>
```

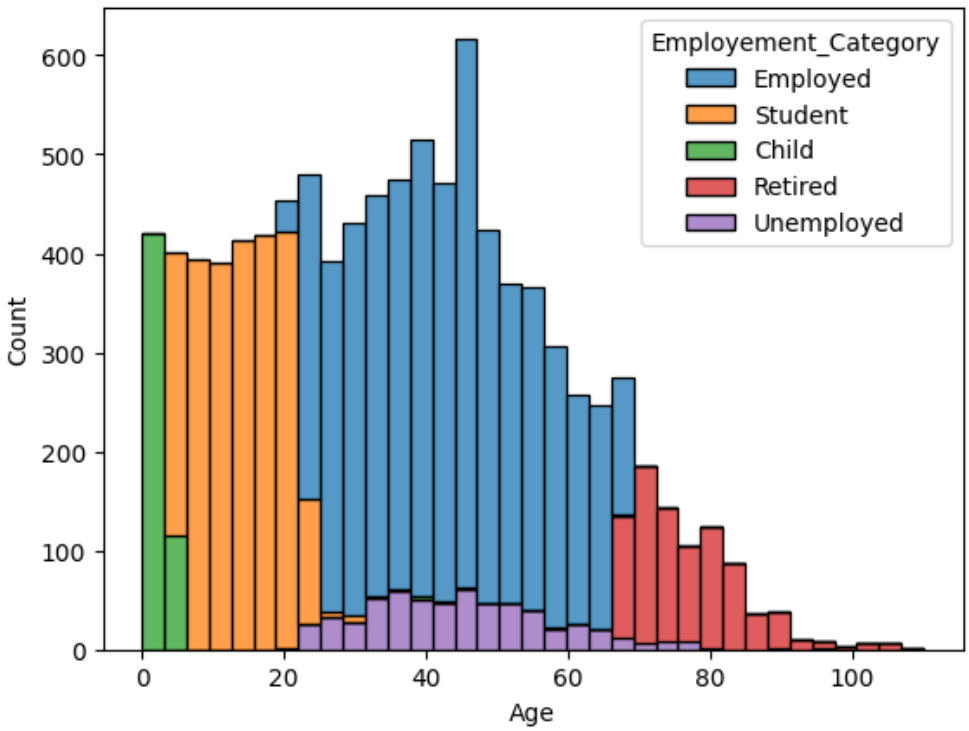


In [35]:

```
sns.histplot(data = dataset.loc[: ,["Employement_Category", "Age"]], x = "Age", hue = "Employement_Categor
```

Out[35]:

```
<Axes: xlabel='Age', ylabel='Count'>
```



In [36]:

```
yc = len(dataset.loc[dataset.Age == 1])
yl = len(dataset.loc[dataset.Age == 6])
current_year_birth_rate = (yc / len(dataset)) * 1000
previous_year_birth_rate = (yl / (len(dataset) - dataset.Age_band.value_counts()["0-5"])) * 1000
print("Current Birth rate : ", current_year_birth_rate, "Birth rate 5 years back: ", previous_year_birth_r
l = dataset.Age_band.value_counts()
req = [l[str(i * 5) + "-" + str((i + 1) * 5)] - l[str((i - 1) * 5) + "-" + str((i) * 5)] for i in range(1
req = sum(req) / 5
death_rate = (req/ 9728) * 1000
print("Death rate: ", death_rate)
```

```
Current Birth rate :  9.457236842105264 Birth rate 5 years back:  15.776302905015777
Death rate:  -8.84046052631579
```

In [37]:

```
#Given the current Birth rate is 9.5 which is decreasing when compared with birth rate 5 years back and ar
#increase in population of school going children will be close to birth_rate * population / 1000
#in next 5 years no of school going children will be close to 5 * birth_rate * population / 1000 + frequer
#current population in school is close to population in within 5 - 20
expected_change_in_students_in_5_years = ((5 *9.5* 9728) / 1000) - 1 * (dataset.Age_band.value_counts()["
expected_change_in_students_in_5_years
```

Out[37]:

```
278.08
```

In [38]:

```
current_retired = len(dataset.loc[dataset.Employement_Category == "Retired"])
retired_in_next5yrs = len(dataset.loc[(dataset.Employement_Category == "Employed") & (dataset.Age >= 60)]
dead_in_next5yrs = (8.8 * 9728 * 5) / 1000
expected_change_in_retired = retired_in_next5yrs + current_retired - dead_in_next5yrs
```

In [39]:

```
print(expected_change_in_retired)
```

```
1019.968
```

In [40]:

```
print(len(dataset.loc[dataset.Employement_Category == "Unemployed"]))
```

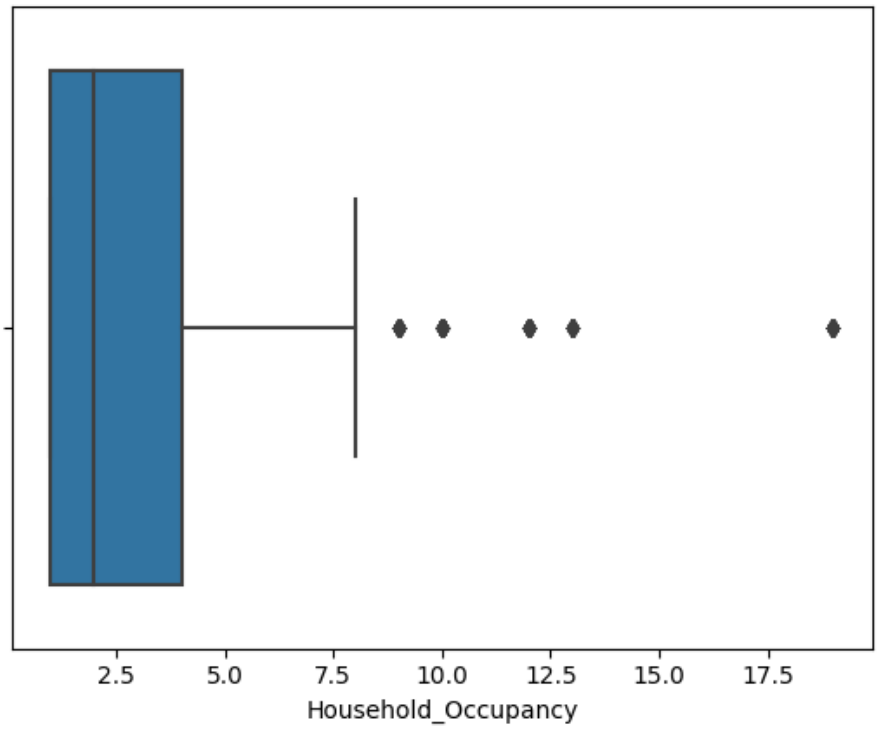
```
593
```

In [41]:

```
sns.boxplot(data=dataset,  x= "Household_Occupancy")
```

Out[41]:

```
<Axes: xlabel='Household_Occupancy'>
```



In [42]:

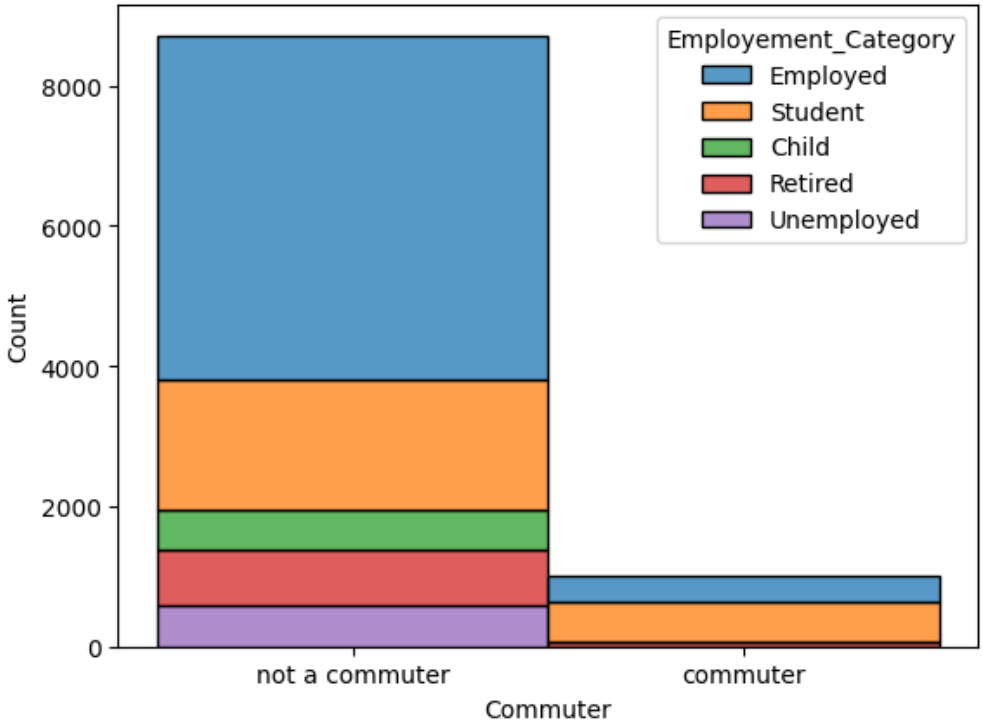
```
l = ["university", "scientist", "professor", "research", "phd"]
def commuter(row):
    req = "not a commuter"
    for i in l:
        if i in dataset["Occupation"][row].lower():
            req = "commuter"
            break
    return req
dataset["Commuter"] = [commuter(i) for i in range(len(dataset))]
```

In [43]:

```
sns.histplot(data = dataset, x = "Commuter", hue = "Employement_Category" , multiple = "stack")
```

Out[43]:

```
<Axes: xlabel='Commuter', ylabel='Count'>
```



In [44]:

```
len(dataset.loc[dataset.Commuter == "commuter" ])
```

Out[44]:

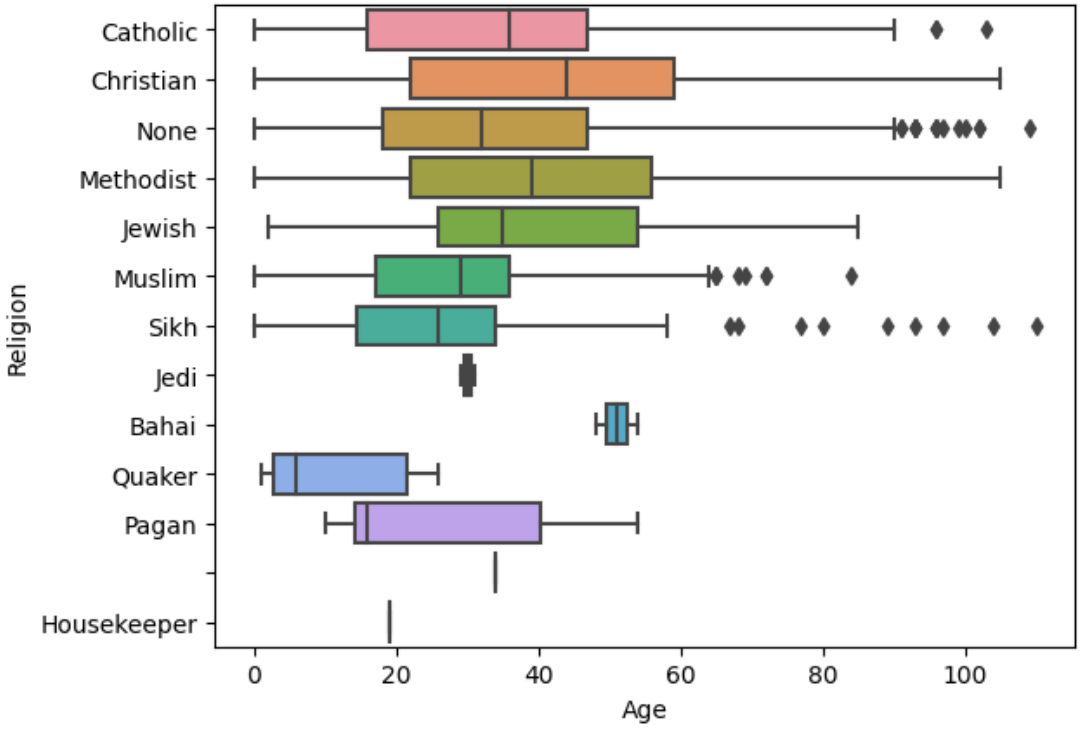
```
1014
```

In [45]:

```
sns.boxplot(data=dataset, y ="Religion", x="Age")
```

Out[45]:

```
<Axes: xlabel='Age', ylabel='Religion'>
```



In [46]:

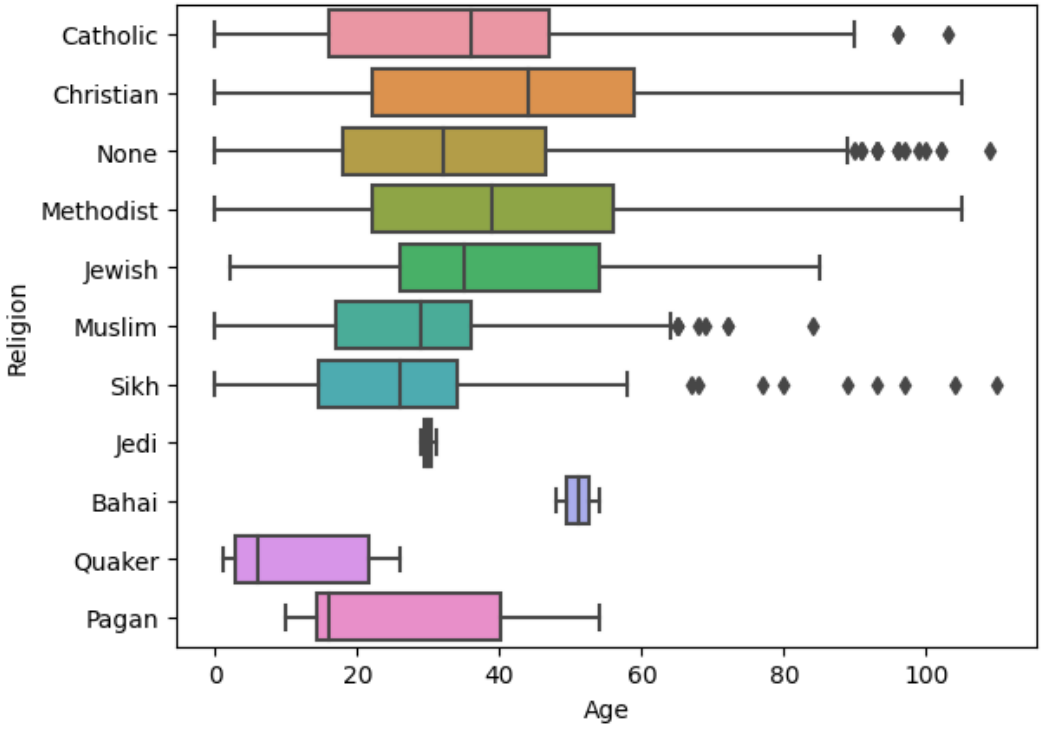
```
dataset["Religion"] = dataset["Religion"].replace([" ", "Housekeeper"], "None")
```

In [47]:

```
sns.boxplot(data=dataset, y ="Religion", x="Age")
```

Out[47]:

```
<Axes: xlabel='Age', ylabel='Religion'>
```

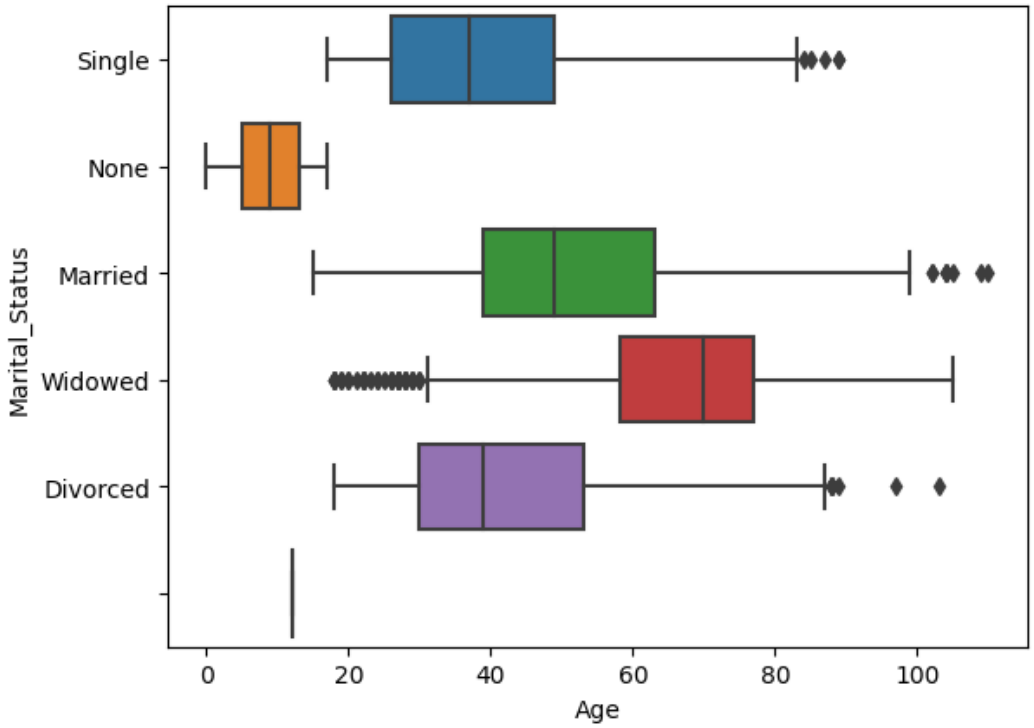


In [48]:

```
sns.boxplot(data=dataset, y ="Marital_Status", x="Age")
```

Out[48]:

```
<Axes: xlabel='Age', ylabel='Marital_Status'>
```

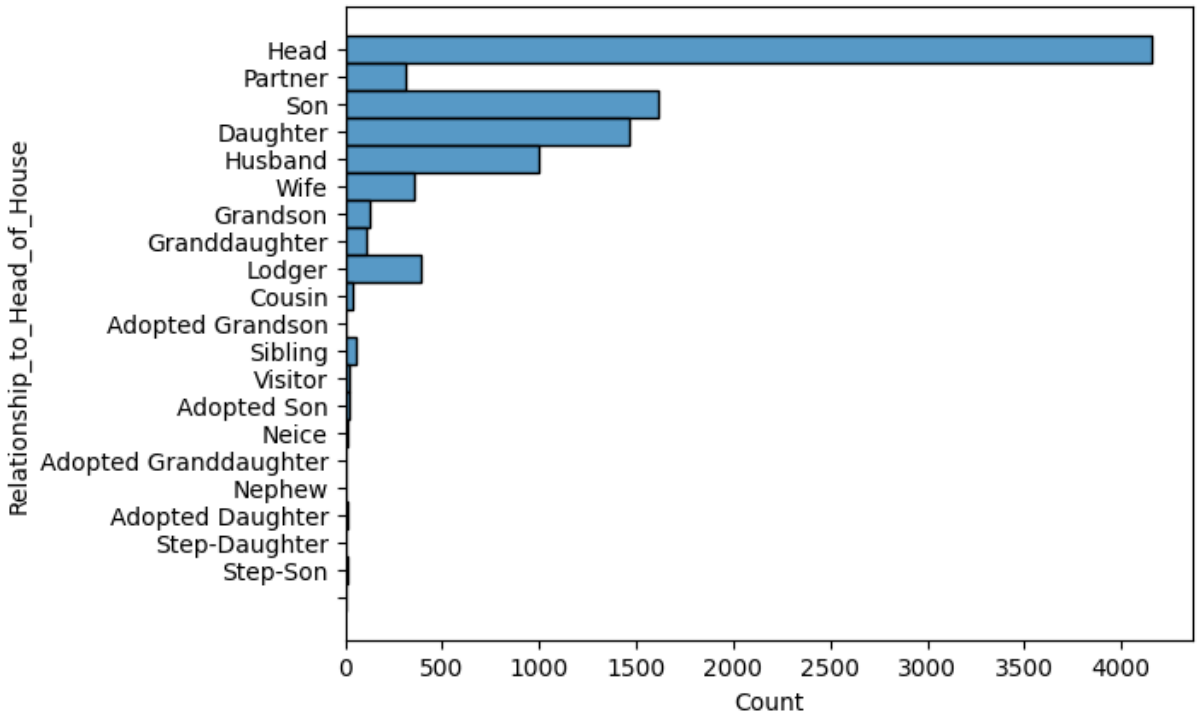


In [49]:

```
sns.histplot(data = dataset, y = "Relationship_to_Head_of_House")
```

Out[49]:

```
<Axes: xlabel='Count', ylabel='Relationship_to_Head_of_House'>
```



In [50]:

```
dataset.Relationship_to_Head_of_House.value_counts()
```

Out[50]:

```
Relationship_to_Head_of_House
Head                        4155
Son                         1611
Daughter                    1462
Husband                      998
Lodger                       388
Wife                         354
Partner                      314
Grandson                     123
Granddaughter                109
Sibling                       58
Cousin                        37
Visitor                       26
Adopted Son                   25
Neice                         14
Step-Son                      10
Adopted Daughter               9
Step-Daughter                  8
Nephew                         7
Adopted Grandson               2
Adopted Granddaughter          2
                               1
Name: count, dtype: int64
```

In [51]:

```
import math
emp_cat = list(dataset.Employement_Category.unique())
l = dataset.Employement_Category.value_counts()
count  = [l[i] for i in emp_cat]
per = [round((i / sum(count) * 100), 2) for i in count]
print(per)
```

```
[54.18, 25.31, 5.65, 8.76, 6.1]
```

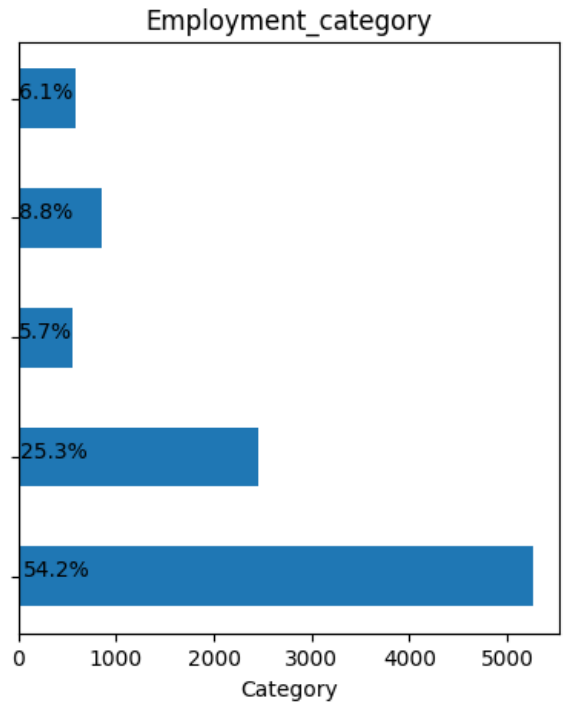
In [52]:

```
l = dataset.Employement_Category.value_counts()
```

In [53]:

```
import matplotlib.pyplot as plt
import pandas as pd

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(10, 5))
ds = pd.DataFrame()
ds["emp_cat"] = emp_cat[::-1]

ds["count"] = count[::-1]
ds["perc"] = per[::-1]
table = ds.values.tolist()
ax1.axis('off')
table_obj = ax1.table(cellText=table, cellLoc="right", bbox=[0, 0, 1, 1])
table_obj.auto_set_font_size(False)
table_obj.set_fontsize(12)

ax2.barh(emp_cat, count, height=0.5)
ax2.set_xlabel('Category')
ax2.set_title('Employment_category')
ax2.set_yticklabels([])  # Remove y-axis tick labels

# Add percentage labels to the bars
for i, v in enumerate(per):
    ax2.text(v + 0.1, i, f'{v:.1f}%', color='black')

plt.show()
```

Unemployed	593	6.1
Retired	852	8.76
Child	550	5.65
Student	2462	25.31
Employed	5271	54.18



In [54]:

```
dataset["Marital_Status"] = dataset["Marital_Status"].replace([" "], "None")
```

In [55]:

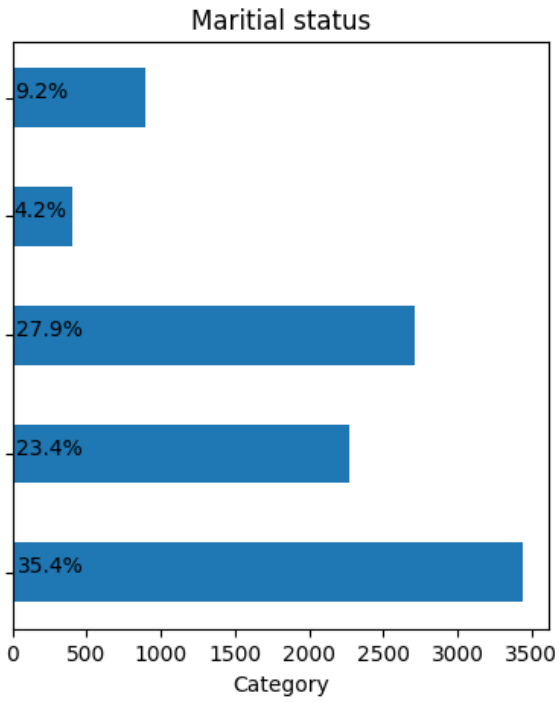
```
mar_cat = list(dataset.Marital_Status.unique())
l = dataset.Marital_Status.value_counts()
count  = [l[i] for i in mar_cat]
per = [round((i / sum(count) * 100), 2) for i in count]
print(per)
```

```
[35.39, 23.38, 27.86, 4.17, 9.2]
```

In [56]:

```
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(10, 5))
ds = pd.DataFrame()
ds["mar_cat"] = mar_cat[::-1]
ds["count"] = count[::-1]
ds["perc"] = per[::-1]
table = ds.values.tolist()
ax1.axis('off')
table_obj = ax1.table(cellText=table, cellLoc="right", bbox=[0, 0, 1, 1])
table_obj.auto_set_font_size(False)
table_obj.set_fontsize(12)

ax2.barh(emp_cat, count, height=0.5)
ax2.set_xlabel('Category')
ax2.set_title('Maritial status')
ax2.set_yticklabels([])  # Remove y-axis tick labels

# Add percentage labels to the bars
for i, v in enumerate(per):
    ax2.text(v + 0.1, i, f'{v:.1f}%', color='black')

plt.show()
```

Divorced	895	9.2
Widowed	406	4.17
Married	2710	27.86
None	2274	23.38
Single	3443	35.39



In [57]:

```
reli_cat = list(dataset.Religion.unique())
l = dataset.Religion.value_counts()
count  = [l[i] if(i in l.index) else (4357) for i in reli_cat]
per = [round((i / sum(count) * 100), 2) for i in count]
print(len(reli_cat), len(count), len(per))
```

```
11 11 11
```

In [58]:

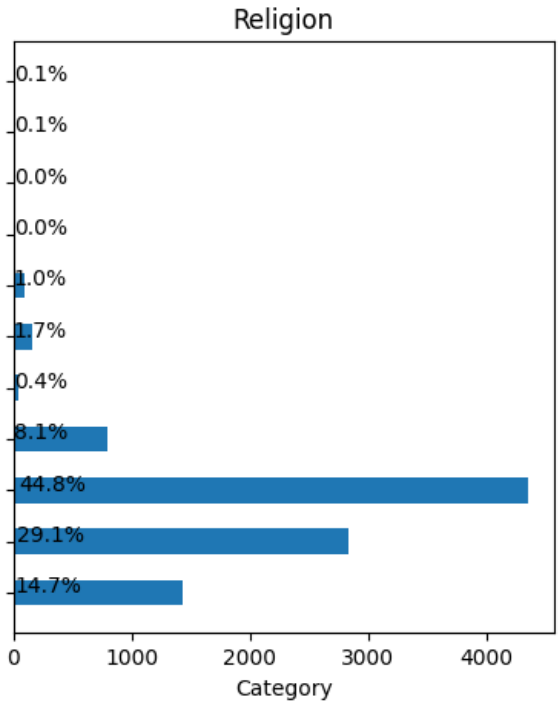
```
print(reli_cat, count, per)
```

```
['Catholic', 'Christian', 'None', 'Methodist', 'Jewish', 'Muslim', 'Sikh', 'Jedi', 'Bahai',
'Quaker', 'Pagan'] [1431, 2830, 4359, 790, 43, 162, 95, 2, 2, 8, 6] [14.71, 29.09, 44.81,
8.12, 0.44, 1.67, 0.98, 0.02, 0.02, 0.08, 0.06]
```

In [59]:

```
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(10, 5))
ds = pd.DataFrame()
ds["reli_cat"] = reli_cat[::-1]
ds["count"] = count[::-1]
ds["perc"] = per[::-1]
table = ds.values.tolist()
ax1.axis('off')
table_obj = ax1.table(cellText=table, cellLoc="right", bbox=[0, 0, 1, 1])
table_obj.auto_set_font_size(False)
table_obj.set_fontsize(12)

ax2.barh(reli_cat, count, height=0.5)
ax2.set_xlabel('Category')
ax2.set_title('Religion')
ax2.set_yticklabels([])  # Remove y-axis tick labels

# Add percentage labels to the bars
for i, v in enumerate(per):
    ax2.text(v + 0.1, i, f'{v:.1f}%', color='black')

plt.show()
```

Pagan	6	0.06
Quaker	8	0.08
Bahai	2	0.02
Jedi	2	0.02
Sikh	95	0.98
Muslim	162	1.67
Jewish	43	0.44
Methodist	790	8.12
None	4359	44.81
Christian	2830	29.09
Catholic	1431	14.71



In [60]:

```
dataset.Infirmity = dataset.Infirmity.fillna(value = "None")
```

In [61]:

```
inf_cat = list(dataset.Infirmity.unique())
l = dataset.Infirmity.value_counts()
count  = [l[i] for i in inf_cat]
per = [round((i / sum(count) * 100), 2) for i in count]
print(per)
```

```
[99.27, 0.1, 0.14, 0.16, 0.06, 0.05, 0.12, 0.08]
```

In [62]:

```
dataset.Infirmity.value_counts()
```

Out[62]:

```
Infirmity
None                   9657
Physical Disability      16
Blind                    14
Mental Disability        12
Deaf                     10
Unknown Infection         8
                          6
Disabled                  5
Name: count, dtype: int64
```

In [63]:

```
dataset["Infirmity"] = dataset["Infirmity"].replace([""], "Missing")
```

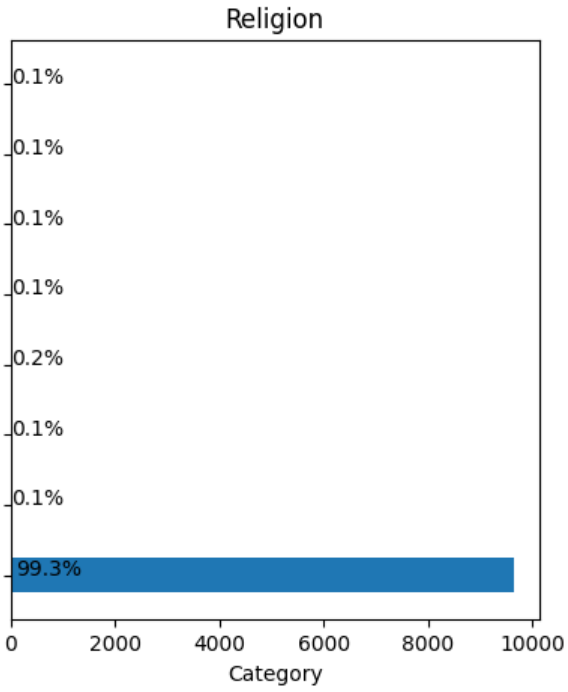
In [64]:

```
dataset["Infirmity"] = dataset["Infirmity"].replace([""], "Missing")
```

In [65]:

```
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(10, 5))
ds = pd.DataFrame()
ds["inf_cat"] = inf_cat[::-1]
ds["count"] = count[::-1]
ds["perc"] = per[::-1]
table = ds.values.tolist()
ax1.axis('off')
table_obj = ax1.table(cellText=table, cellLoc="right", bbox=[0, 0, 1, 1])
table_obj.auto_set_font_size(False)
table_obj.set_fontsize(8)

ax2.barh(inf_cat, count, height=0.5)
ax2.set_xlabel('Category')
ax2.set_title('Religion')
ax2.set_yticklabels([])  # Remove y-axis tick labels

# Add percentage labels to the bars
for i, v in enumerate(per):
    ax2.text(v + 0.1, i, f'{v:.1f}%', color='black')

plt.show()
```

Unknown Infection	8	0.08
Mental Disability	12	0.12
Disabled	5	0.05
	6	0.06
Physical Disability	16	0.16
Blind	14	0.14
Deaf	10	0.1
None	9657	99.27



In [66]:

```
yc = len(dataset.loc[dataset.Age == 1])
yl = len(dataset.loc[dataset.Age == 6])
current_year_birth_rate = (yc / len(dataset)) * 1000
previous_year_birth_rate = (yl / len(dataset)) * 1000
print("Current Birth rate : ", current_year_birth_rate, "Birth rate 5 years back: ", previous_year_birth_r
l = dataset.Age_band.value_counts()
req = [l[str(i * 5) + "-" + str((i + 1) * 5)] - l[str((i - 1) * 5) + "-" + str((i) * 5)] for i in range(1
req = sum(req) / 5
death_rate = (req/ 9728) * 1000
print("Death rate: ", death_rate)
```

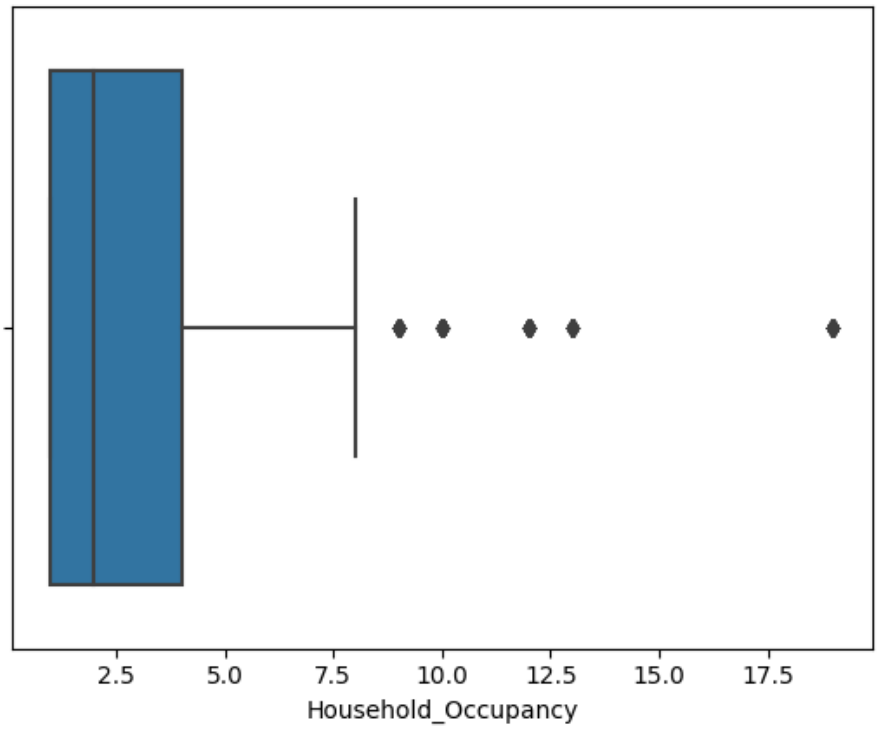
```
Current Birth rate :  9.457236842105264 Birth rate 5 years back:  14.905427631578947
Death rate:  -8.84046052631579
```

In [67]:

```
sns.boxplot(data = dataset, x = "Household_Occupancy")
```

Out[67]:

```
<Axes: xlabel='Household_Occupancy'>
```



In [ ]: