# DCS 530 : Final Project

## "Are marriages made in heaven?"

### Aniruddha Joshi

#### Date: Aug 07, 2023

Data set Reference : https://gssdataexplorer.norc.org/ (https://gssdataexplorer.norc.org/)

A few months ago, one of my friends got engaged, and we celebrated with an engagement party at a nice restaurant. "It looks like your marriage is created in heaven," said one of my friends. This sentence made me wonder: Why do people say things like that? Are marriage truly created in heaven? If so, why do relationships end in divorce? Is it possible to foresee whether a couple will be happy together forever? I choose to conduct some data analysis to find out. The link to the General Social Survey information found at https://gssdataexplorer.norc.org/adfdsf (https://gssdataexplorer.norc.org/adfdsf).

In [44]:
```
#Download the required files for execution
from os.path import basename, exists


def download(url):
    filename = basename(url)
    if not exists(filename):
        from urllib.request import urlretrieve

        local, _ = urlretrieve(url, filename)
        print("Downloaded " + local)


download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkstats2.py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkplot.py")
```

### Import the required libraries

In [45]:
```
#import the required libraries
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import thinkstats2
import thinkplot
import IPython.display

import seaborn as sns

from sklearn.preprocessing import MinMaxScaler

from scipy.stats import pearsonr
from scipy.stats import chi2_contingency
import statsmodels.api as sm
from sklearn.linear_model import LogisticRegression
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

### Import the dataset

In [46]:
```
#Import datasets

dataset_gss = pd.read_csv('C:\\Users\\aniruddha.joshi\\OneDrive - Emerson\\Personal\\MS Data Science Emerson\\DCS 530\\Project
```

In [47]:
```python
# A quick review of the data
dataset_gss
```

Out[47]:

| | year | id_ | hrs1 | wrkslf | marital | divorce | spwrksta | childs | age | educ | spdeg | sex | race | family16 | income | re |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1972 | 1 | .i: Inapplicable | Someone else | Never married | .i: Inapplicable | .i: Inapplicable | 0 | 23 | 16 | .i: Inapplicable | FEMALE | White | FATHER | .i: Inapplicable | Inappli |
| **1** | 1972 | 2 | .i: Inapplicable | Someone else | Married | NO | KEEPING HOUSE | 5 | 70 | 10 | HIGH SCHOOL | MALE | White | M AND F RELATIVES | .i: Inapplicable | Inappli |
| **2** | 1972 | 3 | .i: Inapplicable | Someone else | Married | NO | WORKING FULLTIME | 4 | 48 | 12 | .n: No answer | FEMALE | White | MOTHER & FATHER | .i: Inapplicable | Inappli |
| **3** | 1972 | 4 | .i: Inapplicable | Someone else | Married | NO | WORKING FULLTIME | 0 | 27 | 17 | GRADUATE | FEMALE | White | MOTHER & FATHER | .i: Inapplicable | Inappli |
| **4** | 1972 | 5 | .i: Inapplicable | Someone else | Married | NO | TEMP NOT WORKING | 2 | 61 | 12 | HIGH SCHOOL | FEMALE | White | MOTHER & FATHER | .i: Inapplicable | Inappli |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **72385** | 2022 | 3541 | 48 | Someone else | Never married | .i: Inapplicable | .x: Not available in this release | 0 | 22 | 12 | .x: Not available in this release | FEMALE | White | MOTHER & STPFATHER | $25,000 or more | .: availa this re |
| **72386** | 2022 | 3542 | 50 | Someone else | Married | YES | .x: Not available in this release | 2 | 29 | 19 | .x: Not available in this release | FEMALE | White | MOTHER & FATHER | $25,000 or more | .: availa this re |
| **72387** | 2022 | 3543 | 38 | Someone else | Never married | .i: Inapplicable | .x: Not available in this release | 1 | 32 | 15 | .x: Not available in this release | MALE | White | MOTHER & STPFATHER | $25,000 or more | .: availa this re |
| **72388** | 2022 | 3544 | 40 | Someone else | Married | NO | .x: Not available in this release | 0 | 49 | 17 | .x: Not available in this release | FEMALE | White | MOTHER & FATHER | $25,000 or more | .: availa this re |
| **72389** | 2022 | 3545 | 40 | Someone else | Married | NO | .x: Not available in this release | 1 | 50 | 20 | .x: Not available in this release | MALE | White | MOTHER & FATHER | $25,000 or more | .: availa this re |

72390 rows × 19 columns

In [48]:
```python
# Listing the available columns
data=dataset_gss.columns
df = pd.DataFrame(data)
df
```

Out[48]:

| | 0 |
|---|---|
| **0** | year |
| **1** | id_ |
| **2** | hrs1 |
| **3** | wrkslf |
| **4** | marital |
| **5** | divorce |
| **6** | spwrksta |
| **7** | childs |
| **8** | age |
| **9** | educ |
| **10** | spdeg |
| **11** | sex |
| **12** | race |
| **13** | family16 |
| **14** | income |
| **15** | relig16 |
| **16** | hapmar |
| **17** | ballot |
| **18** | incomeUSD |

# Vailable Definations

To find out if the couple will stay together happily forever lets analyze different sets of parametersm. Below is the explanation of each variable

```
-ballot   :  ballot used for interview
-hrs1     :  number of hours worked last week. To check if couples have a time for each other.
-wrkslf   :  self-emp or works for somebody. To check if they work for someone else.
-marital :  marital status. The dependent varilable 1.
-divorce :  ever been divorced or separated. The dependent varilable 2.
-spwrksta:  spouse labor force status
-childs   :  number of children. Checking the number of childrens helps couple to stay together.
-age      :  age of respondent. Is there any age factor in divorce.
-educ     :  highest year of school completed. Is education playing any role for couple to stay together.
-spdeg    :  spouse's highest degree.
-sex      :  respondents sex. Are more males divorced than females?
-race     :  race of respondent. Are divorce % larger in any specific race?
-family16:  living with parents when 16 yrs old. Staying long with parents helps achieve family bonding?
-income   :  total family income. Is there a role of income in happy marriages?
-relig16 :  religion in which raised. Do certain religions having lesser % of divorce?
-hapmar   :  happiness of marriage. Dependant variable 3.
-year     :  GSS year for this respondent
-incomUSD:  Income in USD
-id       :  Respondent id number
```

## Clean up the Database

In [49]:
```python
#Clean up the database
#Replaced invalid/incorrect strings with NaN
#This makes replacement in the complete database
dataset_gss.replace('.i:  Inapplicable', np.nan, inplace=True)
dataset_gss.replace('.d:  Do not Know/Cannot Choose', np.nan, inplace=True)
dataset_gss.replace('.f:  Missing Birthdate Information', np.nan, inplace=True)
dataset_gss.replace('.i:  Inapplicable', np.nan, inplace=True)
dataset_gss.replace('.j:  I do not have a job', np.nan, inplace=True)
dataset_gss.replace('.m:  DK, NA, IAP', np.nan, inplace=True)
dataset_gss.replace('.n:  No answer', np.nan, inplace=True)
dataset_gss.replace('.p:  Not applicable (I have not faced this decision)/Not imputable', np.nan, inplace=True)
dataset_gss.replace('.q:  Not imputable', np.nan, inplace=True)
dataset_gss.replace('.r:  Refused', np.nan, inplace=True)
dataset_gss.replace('.s:  Skipped on Web', np.nan, inplace=True)
dataset_gss.replace('.u:  Uncodable', np.nan, inplace=True)
dataset_gss.replace('.x:  Not available in this release', np.nan, inplace=True)
dataset_gss.replace('.y:  Not available in this year', np.nan, inplace=True)
dataset_gss.replace('.z:  Variable-specific reserve code', np.nan, inplace=True)
dataset_gss.replace('8 or more', int(8), inplace=True)
dataset_gss.replace('89 or older', int(89), inplace=True)
dataset_gss.replace('No formal schooling', int(0), inplace=True)
```

### Converting the columns to appropriate data format

In [50]:
```python
#Convert fields to appropriate format
dataset_gss['hrs1'] = pd.to_numeric(dataset_gss['hrs1'], errors='coerce')
dataset_gss['wrkslf'] = dataset_gss['wrkslf'].astype('category')
dataset_gss['marital'] = dataset_gss['marital'].astype('category')
dataset_gss['divorce'] = dataset_gss['divorce'].astype('category')
dataset_gss['spwrksta'] = dataset_gss['spwrksta'].astype('category')
dataset_gss['childs'] = pd.to_numeric(dataset_gss['childs'], errors='coerce')
dataset_gss['age'] = pd.to_numeric(dataset_gss['age'], errors='coerce')
dataset_gss['educ'] = pd.to_numeric(dataset_gss['educ'], errors='coerce')
dataset_gss['spdeg'] = dataset_gss['spdeg'].astype('category')
dataset_gss['sex'] = dataset_gss['sex'].astype('category')
dataset_gss['race'] = dataset_gss['race'].astype('category')
dataset_gss['family16'] = dataset_gss['family16'].astype('category')
dataset_gss['income'] = dataset_gss['income'].astype('category')
dataset_gss['relig16'] = dataset_gss['relig16'].astype('category')
dataset_gss['hapmar'] = dataset_gss['hapmar'].astype('category')
dataset_gss['incomeUSD'] = pd.to_numeric(dataset_gss['incomeUSD'], errors='coerce')
dataset_gss['ballot'] = dataset_gss['ballot'].astype('string')
```

In [51]:
```
#QUick review of numeric fields
#descriptive characteristics about the variables: Mean, Mode, Spread, and Tail
dataset_gss.describe()
```

Out[51]:

| | year | id_ | hrs1 | childs | age | educ | incomeUSD |
|---|---|---|---|---|---|---|---|
| **count** | 72390.000000 | 72390.000000 | 41266.000000 | 72129.000000 | 71621.000000 | 72127.000000 | 63439.000000 |
| **mean** | 1997.715541 | 1241.796395 | 40.843285 | 1.916538 | 46.555982 | 13.034633 | 25948.477378 |
| **std** | 15.109995 | 912.273245 | 13.584545 | 1.759511 | 17.600417 | 3.182372 | 9403.188921 |
| **min** | 1972.000000 | 1.000000 | 0.000000 | 0.000000 | 18.000000 | 0.000000 | 1011.392793 |
| **25%** | 1985.000000 | 534.000000 | 37.000000 | 0.000000 | 32.000000 | 12.000000 | 19058.115660 |
| **50%** | 1998.000000 | 1083.000000 | 40.000000 | 2.000000 | 44.000000 | 12.000000 | 30941.325100 |
| **75%** | 2010.000000 | 1722.000000 | 48.000000 | 3.000000 | 60.000000 | 16.000000 | 33203.877585 |
| **max** | 2022.000000 | 4510.000000 | 88.000000 | 8.000000 | 89.000000 | 20.000000 | 35499.895520 |

In [94]:
```
from tabulate import tabulate
data_calc = dataset_gss[['hrs1', 'age', 'educ','incomeUSD', 'childs']]
mean = data_calc.mean()
mode = data_calc.mode().iloc[0]  # Get the first row as mode
spread = data_calc.describe().loc[['mean', 'std', 'min', '25%', '50%', '75%', 'max']]
tails = data_calc.quantile([0.05, 0.95])

# Create a DataFrame to display the results in a tabular format
result_table = pd.DataFrame({
    'Statistic': ['Mean', 'Mode', 'Standard Deviation', 'Minimum', '25th Percentile', 'Median (50th Percentile)', '75th Percer
    'Hours Worked (hrs1)': [mean['hrs1'], mode['hrs1'], spread.loc['std', 'hrs1'], spread.loc['min', 'hrs1'], spread.loc['25%
    'Age': [mean['age'], mode['age'], spread.loc['std', 'age'], spread.loc['min', 'age'], spread.loc['25%', 'age'], spread.loc
    'Education (educ)': [mean['educ'], mode['educ'], spread.loc['std', 'educ'], spread.loc['min', 'educ'], spread.loc['25%',
    'Income (USD)': [mean['incomeUSD'], mode['incomeUSD'], spread.loc['std', 'incomeUSD'], spread.loc['min', 'incomeUSD'], spr
    'Number of Children (childs)': [mean['childs'], mode['childs'], spread.loc['std', 'childs'], spread.loc['min', 'childs'],
})

# Print the result table with headers
result_table
```

Out[94]:

| | Statistic | Hours Worked (hrs1) | Age | Education (educ) | Income (USD) | Number of Children (childs) |
|---|---|---|---|---|---|---|
| **0** | Mean | 40.843285 | 46.555982 | 13.034633 | 25948.477378 | 1.916538 |
| **1** | Mode | 40.000000 | 30.000000 | 12.000000 | 32322.542030 | 0.000000 |
| **2** | Standard Deviation | 13.584545 | 17.600417 | 3.182372 | 9403.188921 | 1.759511 |
| **3** | Minimum | 0.000000 | 18.000000 | 0.000000 | 1011.392793 | 0.000000 |
| **4** | 25th Percentile | 37.000000 | 32.000000 | 12.000000 | 19058.115660 | 0.000000 |
| **5** | Median (50th Percentile) | 40.000000 | 44.000000 | 12.000000 | 30941.325100 | 2.000000 |
| **6** | 75th Percentile | 48.000000 | 60.000000 | 16.000000 | 33203.877585 | 3.000000 |
| **7** | Maximum | 88.000000 | 89.000000 | 20.000000 | 35499.895520 | 8.000000 |
| **8** | 5th Percentile | 15.000000 | 22.000000 | 8.000000 | 7085.611358 | 0.000000 |
| **9** | 95th Percentile | 64.000000 | 78.000000 | 18.000000 | 35034.412535 | 5.000000 |

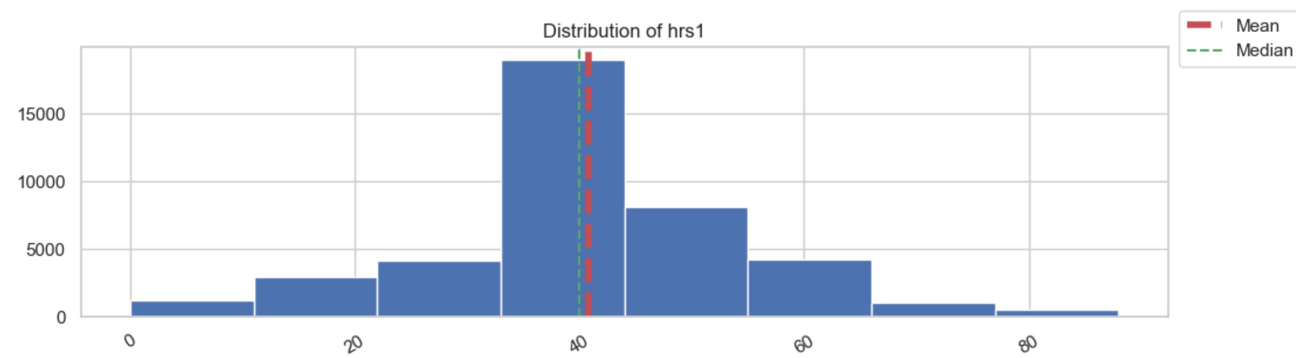
In [52]:
```python
# Print unique categorical values for each column
unique_values = {}
for column in dataset_gss.columns:
    if dataset_gss[column].dtype == 'category':
        unique_values[column] = dataset_gss[column].cat.categories.tolist()

max_len = max(len(val) for val in unique_values.values())
for col in unique_values:
    unique_values[col] += [' '] * (max_len - len(unique_values[col]))

strPrint=(pd.DataFrame(unique_values).transpose())

strPrint
```

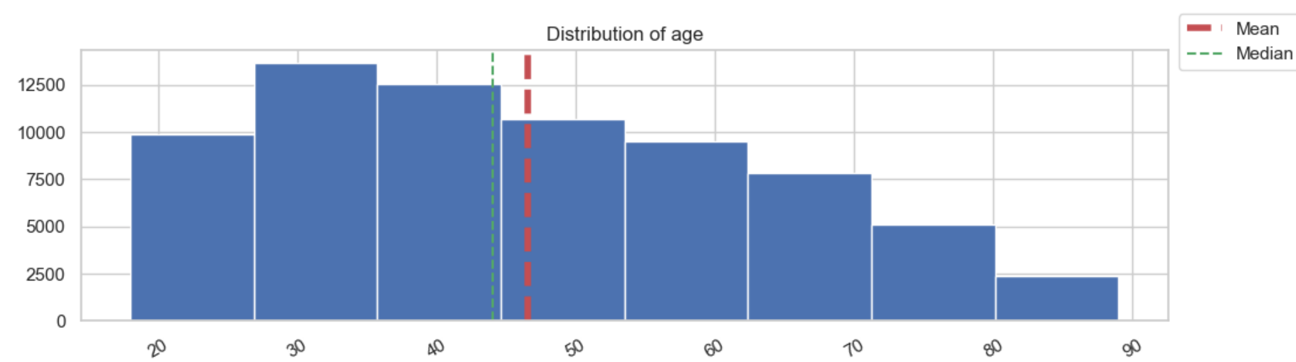
Out[52]:

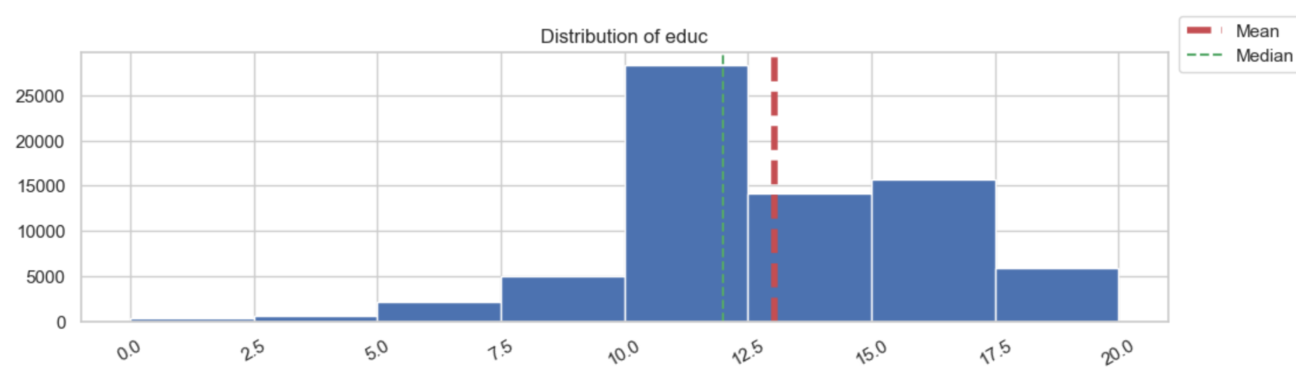
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **wrkslf** | Self-employed | Someone else | | | | | | | | |
| **marital** | Divorced | Married | Never married | Separated | Widowed | | | | | |
| **divorce** | NO | YES | | | | | | | | |
| **spwrksta** | KEEPING HOUSE | OTHER | RETIRED | SCHOOL | TEMP NOT WORKING | UNEMPL, LAID OFF | WORKING FULLTIME | WORKING PARTTIME | | |
| **spdeg** | ASSOCIATE/JUNIOR COLLEGE | BACHELOR | GRADUATE | HIGH SCHOOL | LT HIGH SCHOOL | | | | | |
| **sex** | FEMALE | MALE | | | | | | | | |
| **race** | Black | Other | White | | | | | | | |
| **family16** | FATHER | FATHER & STPMOTHER | FEMALE RELATIVE | M AND F RELATIVES | MALE RELATIVE | MOTHER | MOTHER & FATHER | MOTHER & STPFATHER | OTHER | |
| **income** | $1,000to$2,999 | $10,000to$ 14,999 | $15,000to$ 19,999 | $20,000to$ 24,999 | $25,000 or more | $3,000to$ 3,999 | $4,000to$4,999 | $5,000to$ 5,999 | $6,$ $000to$ 6,999 | $7,000to$ 7,999 |
| **relig16** | BUDDHISM | CATHOLIC | CHRISTIAN | HINDUISM | INTER-NONDENOMINATIONAL | JEWISH | MUSLIM/ISLAM | NATIVE AMERICAN | NONE | ORTHODOX-CHRISTIAN |
| **hapmar** | NOT TOO HAPPY | PRETTY HAPPY | VERY HAPPY | | | | | | | |

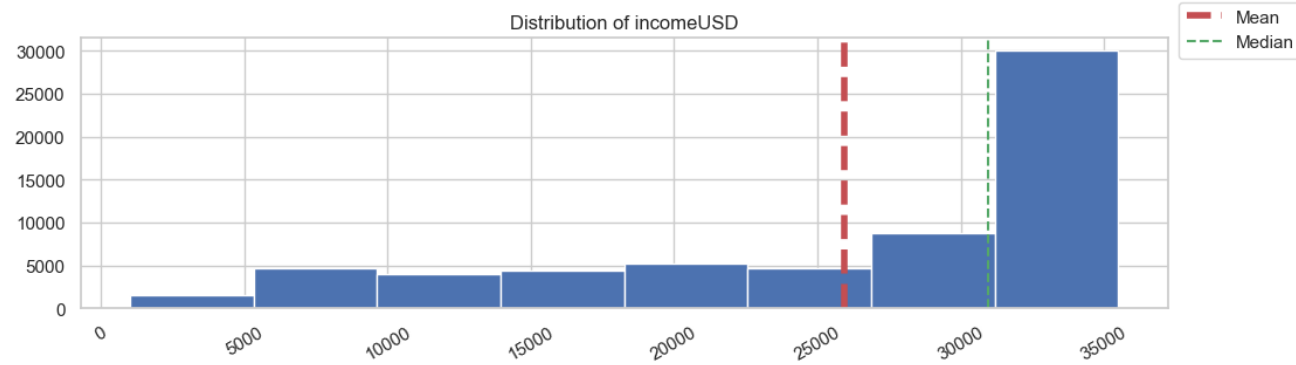
### Plotting Histograms of general polulation

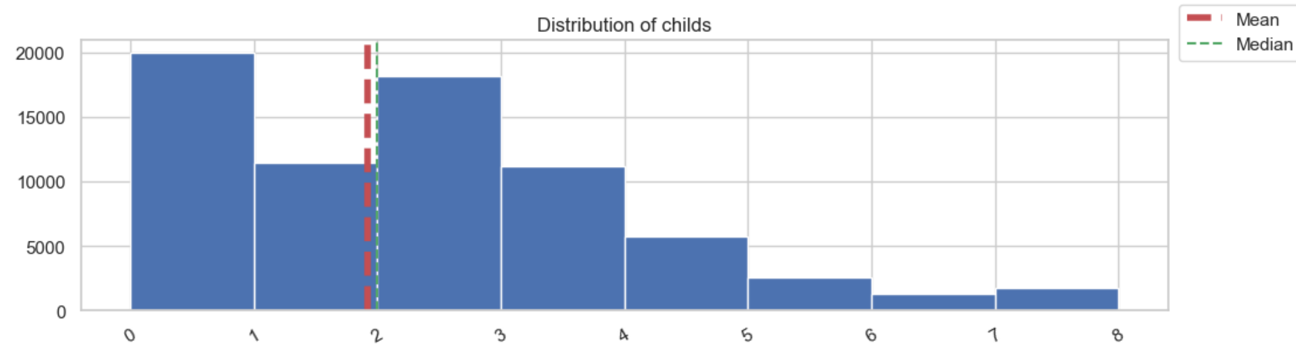
In [85]:
```
#Index(['year', 'id_', 'hrs1', 'wrkslf', 'marital', 'divorce', 'spwrksta',_
#'childs', 'age', 'educ', 'spdeg', 'sex', 'race', 'family16', 'income',
#       'relig16', 'hapmar', 'ballot', 'hsr1'],
#      dtype='object')

# Plot histograms of the height, weight, and age data
#dataset_gss[['hrs1', 'wrkslf', 'marital', 'divorce']].hist(figsize=(5, 5))
    # Add mean and median lines


for column in ['hrs1', 'age', 'educ','incomeUSD', 'childs', 'marital', 'divorce', 'spwrksta',
                 'sex', 'race', 'family16', 'income',
               'hapmar','wrkslf' ]:
    fig, ax = plt.subplots()
    dataset_gss[column].hist(ax=ax, bins=8, figsize=(12, 3))
    plt.title(f"Distribution of {column}")

    if is_numeric(dataset_gss[column]):
        ax.axvline(dataset_gss[column].mean(), color='r', linestyle='dashed', label='Mean', linewidth = 4)
        ax.axvline(dataset_gss[column].median(), color='g', linestyle='dashed', label='Median')

    handles, labels = ax.get_legend_handles_labels()
    fig.legend(handles, labels, loc="upper right")
    # Rotate x-axis labels by 30 degrees
    plt.xticks(rotation=30)
    plt.show()
```

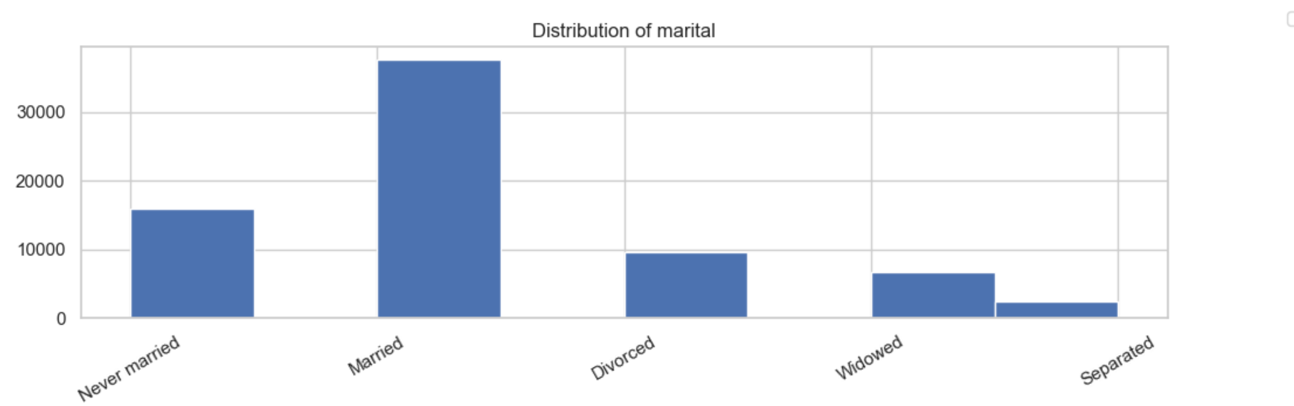


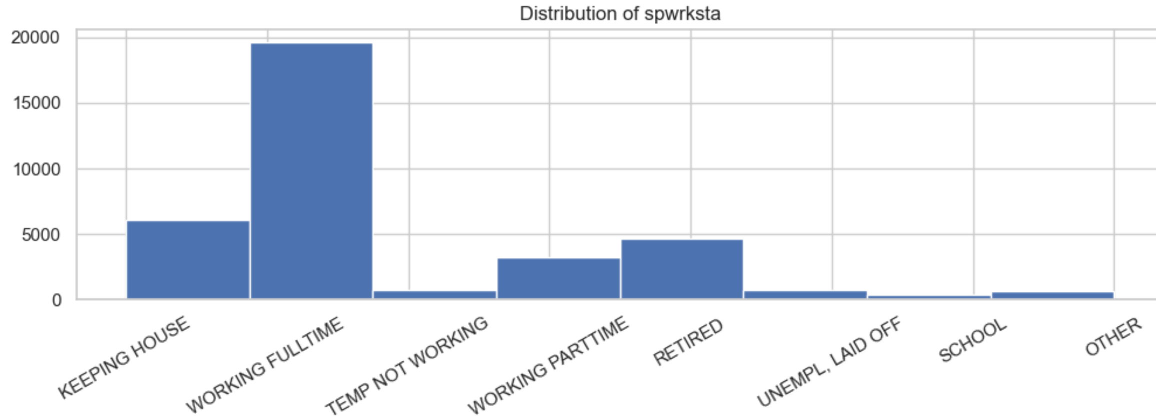




Distribution of incomeUSD
Mean
Median
0
5000
10000
15000
20000
25000
30000
35000

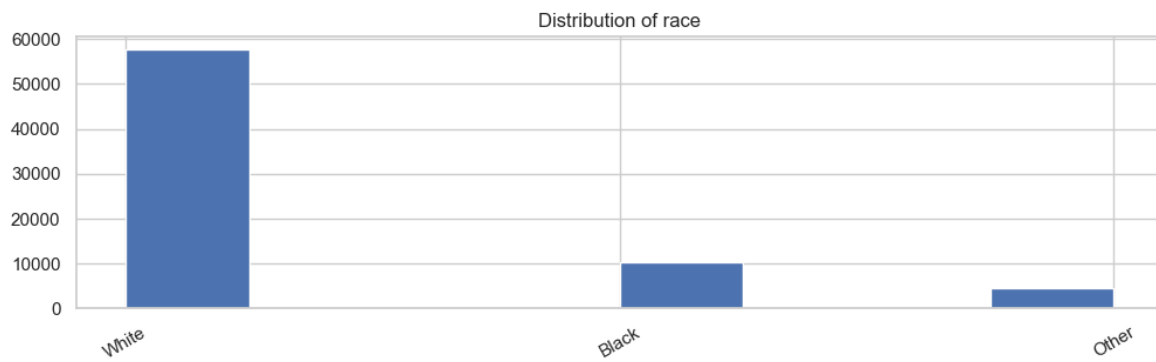

Distribution of childs
Mean
Median
0
5000
10000
15000
20000
1
2
3
4
5
6
7
8

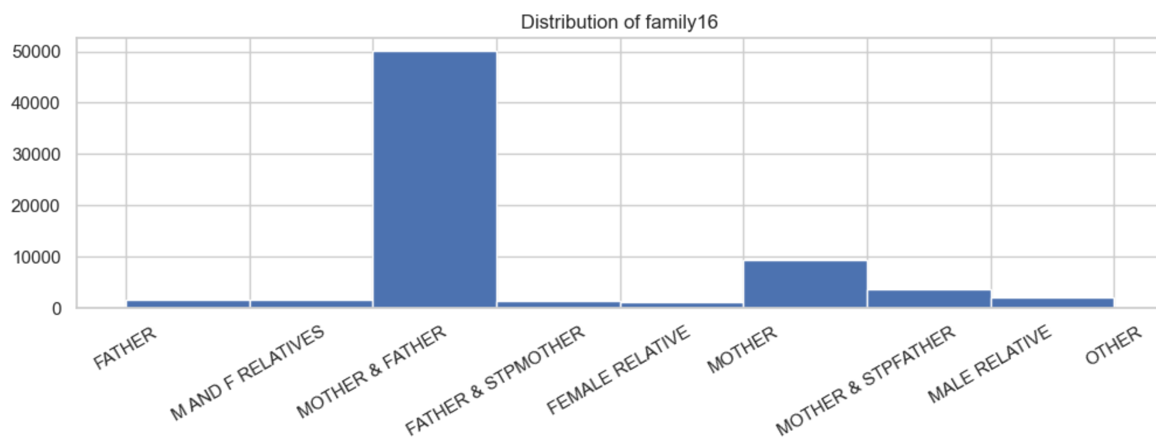

Distribution of marital
0
10000
20000
30000
Never married
Married
Divorced
Widowed
Separated

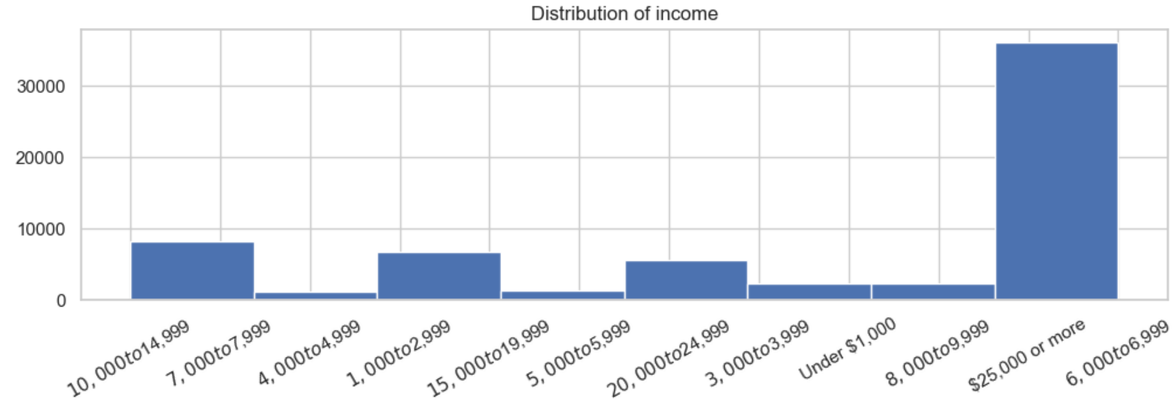

Distribution of divorce
0
10000
20000
30000
NO
YES

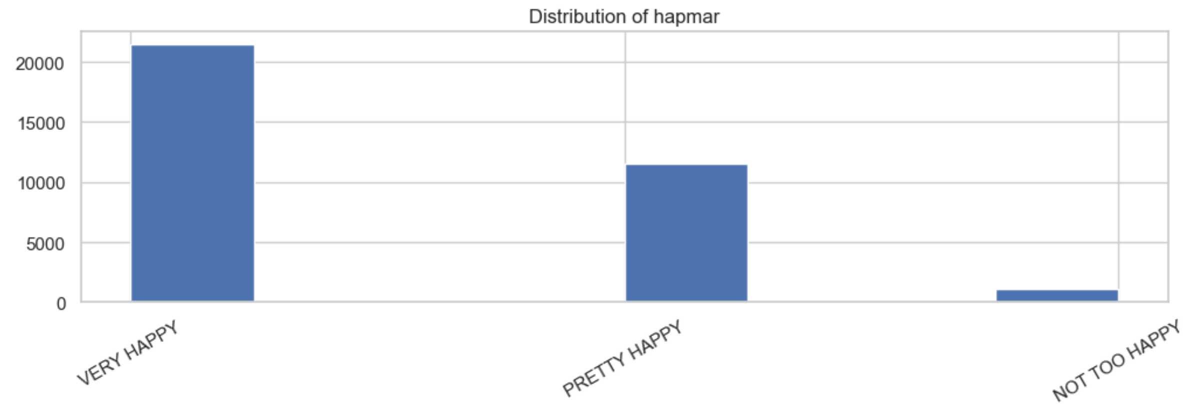

Distribution of spwrksta
20000
15000
10000
5000
0
KEEPING HOUSE
WORKING FULLTIME
TEMP NOT WORKING
WORKING PARTTIME
RETIRED
UNEMPL, LAID OFF
SCHOOL
OTHER


Distribution of sex
40000
30000
20000
10000
0
FEMALE
MALE


Distribution of race
60000
50000
40000
30000
20000
10000
0
White
Black
Other


Distribution of family16
50000
40000
30000
20000
10000
0
FATHER
M AND F RELATIVES
MOTHER & FATHER
FATHER & STPMOTHER
FEMALE RELATIVE
MOTHER
MOTHER & STPFATHER
MALE RELATIVE
OTHER

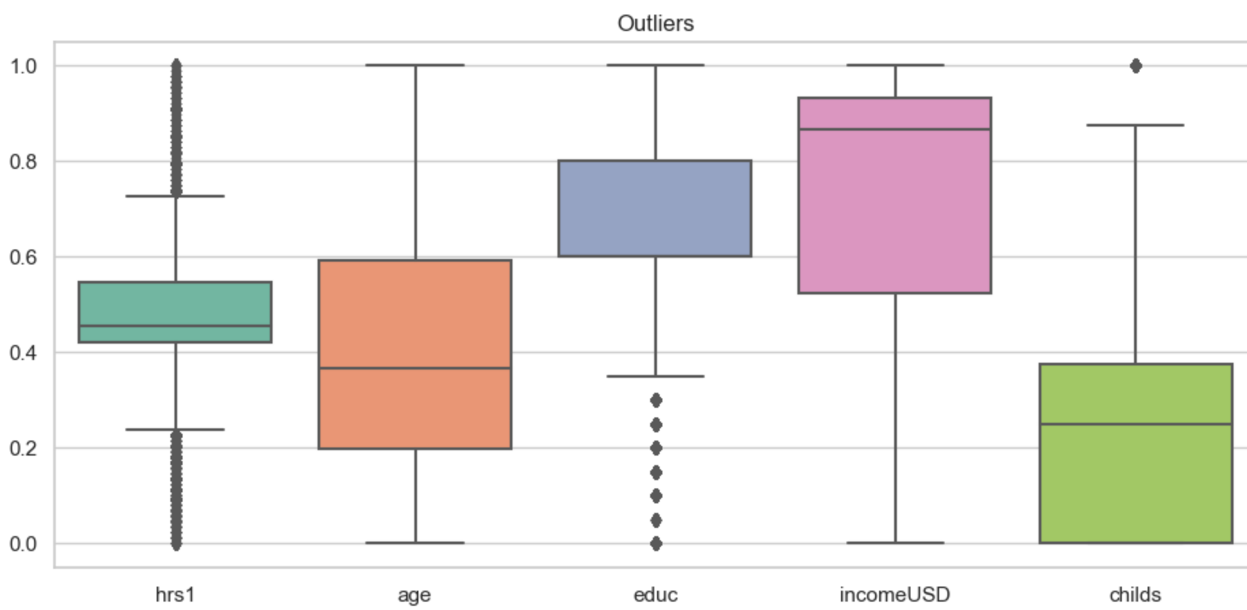

Distribution of income
30000
20000
10000
0
10,000to14,999
7,000to7,999
4,000to4,999
1,000to2,999
15,000to19,999
5,000to5,999
20,000to24,999
3,000to3,999
Under $1,000
8,000to9,999
$25,000 or more
6,000to6,999


Distribution of hapmar
20000
15000
10000
5000
0
VERY HAPPY
PRETTY HAPPY
NOT TOO HAPPY


Distribution of wrkslf
60000
50000
40000
30000
20000
10000
0
Someone else
Self-employed

### Identifying the outliers

In [54]:
```
#Plotting the outliers of 5 numeric variables

# Load the dataset
dataset = dataset_gss[['hrs1', 'age', 'educ',  'incomeUSD','childs']]

df = dataset

scaler = MinMaxScaler()
df_normalized = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)

# Create box plots for each normalized numerical variable
plt.figure(figsize=(25, 5))
sns.set(style="whitegrid")
plt.subplot(1, 2,2)
box_plot = sns.boxplot(data=df_normalized, palette="Set2")
plt.title('Outliers')

# Adding legend
# Create a custom legend
legend_labels = ['hrs1', 'age', 'educ', 'incomeUSD', 'childs']
legend_handles = [plt.Line2D([0], [0], marker='o', color='w', markerfacecolor=sns.color_palette("Set2")[i], markersize=10) for

#plt.legend(legend_handles, legend_labels, loc='right')

plt.show()

# Create scatter plots for each normalized numerical variable
#plt.subplot(1, 2, 2)
#sns.pairplot(dataset_gss)
#plt.title('Scatter Plot of Normalized Numerical Variables')

#plt.tight_layout()
#plt.show()
```

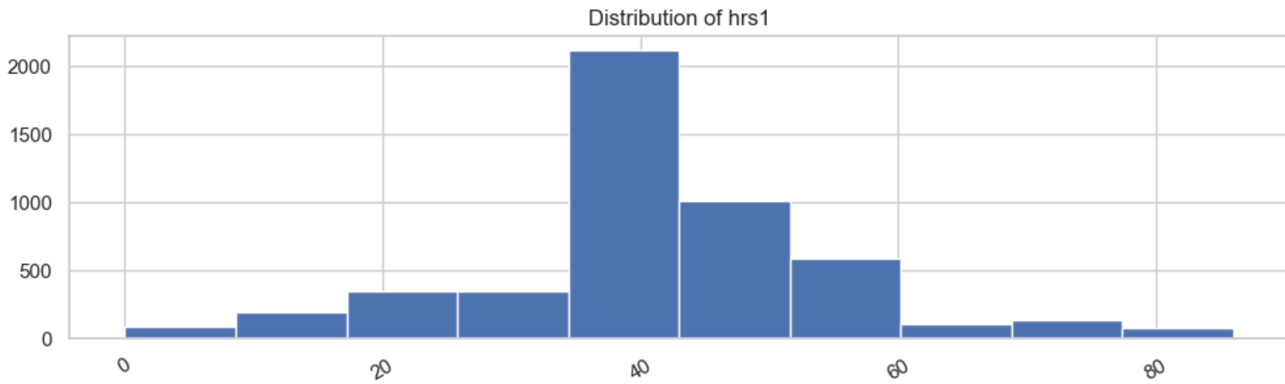


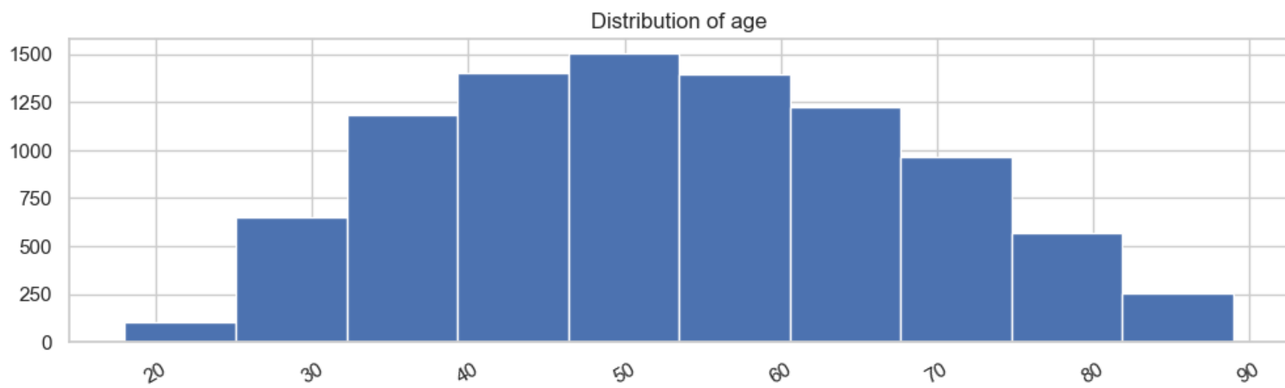

### Plotting histogram of divorced population

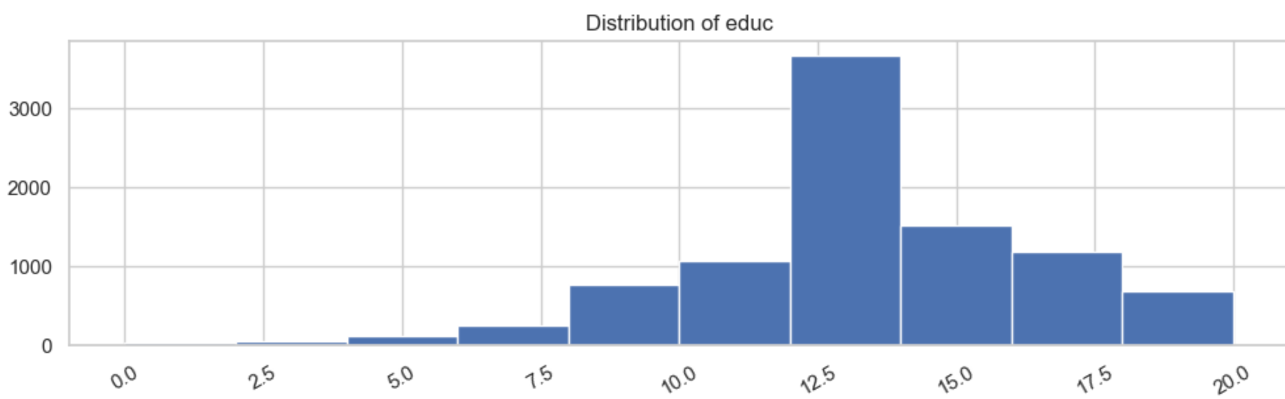
In [55]:
```
# Filtering the dataset based on fivorce status
ds_divorced = dataset_gss[dataset_gss.divorce == 'YES']
```

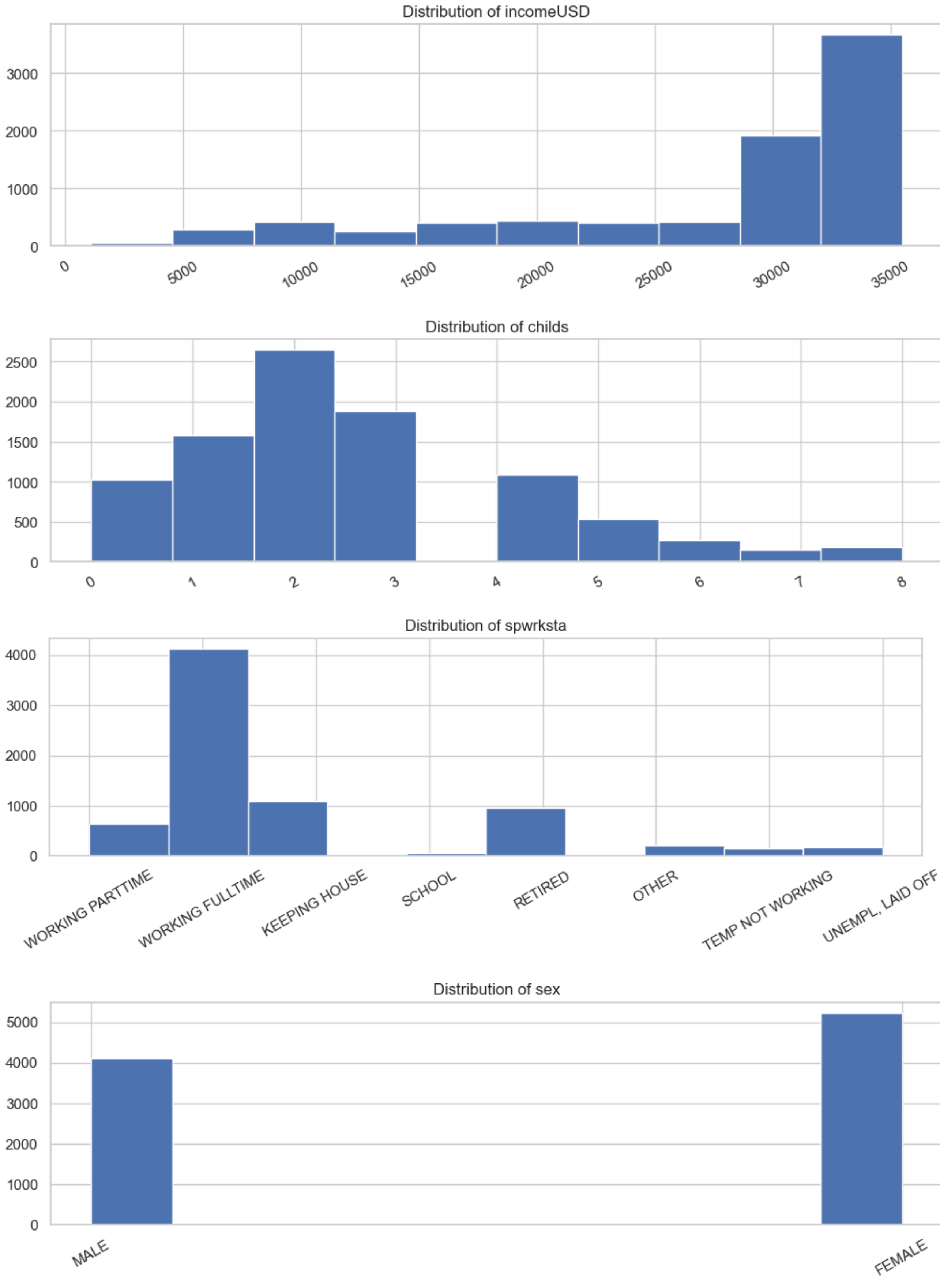
In [56]:

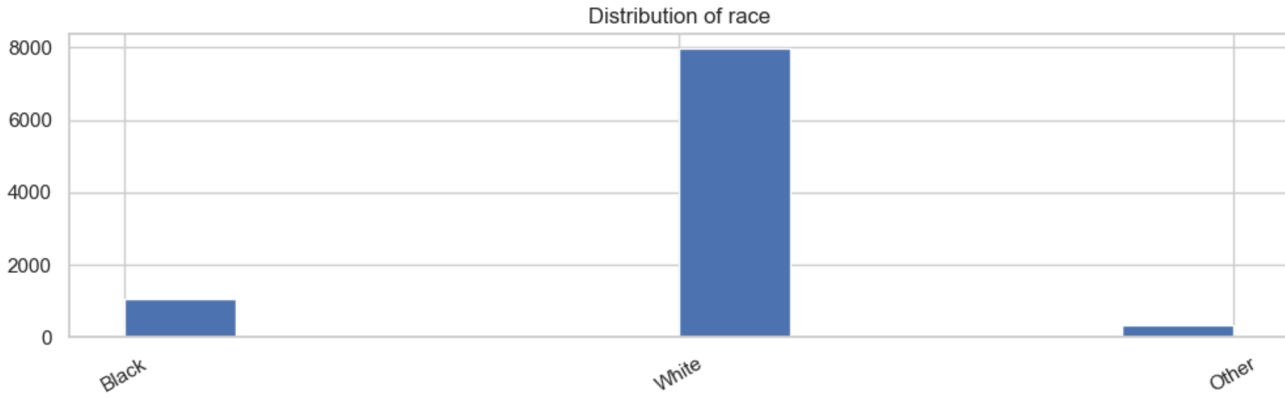
```
for column in ['hrs1', 'age', 'educ',  'incomeUSD','childs', 'spwrksta',
                 'sex', 'race', 'family16','wrkslf']:
    fig, ax = plt.subplots()
    ds_divorced[column].hist(ax=ax, bins=10, figsize=(12, 3))
    plt.title(f"Distribution of {column}")

    # Rotate x-axis labels by 30 degrees
    plt.xticks(rotation=30)

    plt.show()
    #hist = thinkstats2.Hist(ds_divorced[column])
    #thinkplot.Hist(hist)
```

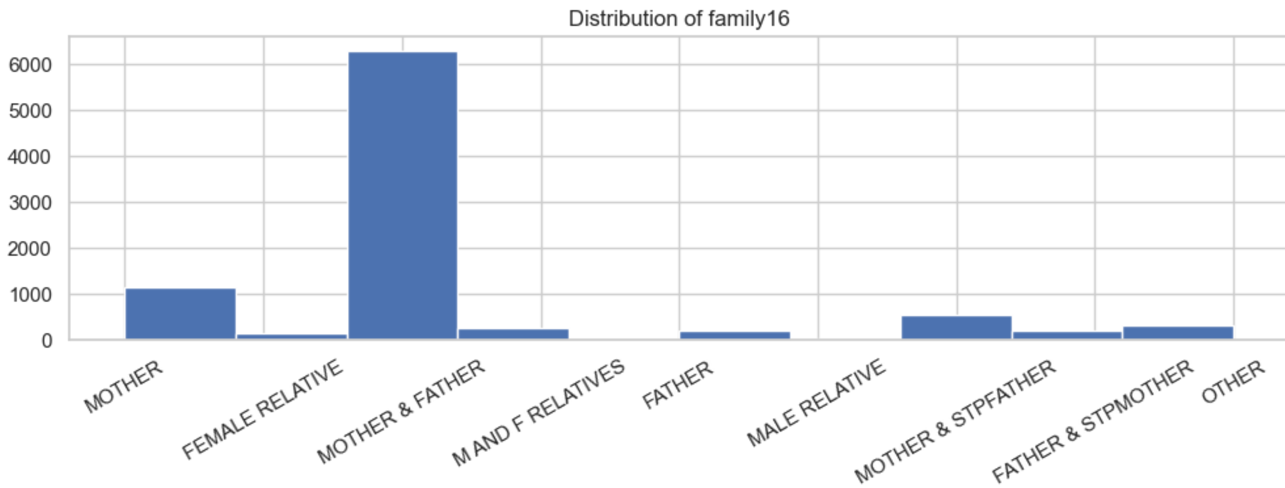



Distribution of incomeUSD
0
1000
2000
3000
0
5000
10000
15000
20000
25000
30000
35000
Distribution of childs
0
500
1000
1500
2000
2500
0
1
2
3
4
5
6
7
8
Distribution of spwrksta
0
1000
2000
3000
4000
WORKING PARTTIME
WORKING FULLTIME
KEEPING HOUSE
SCHOOL
RETIRED
OTHER
TEMP NOT WORKING
UNEMPL, LAID OFF
Distribution of sex
0
1000
2000
3000
4000
5000
MALE
FEMALE

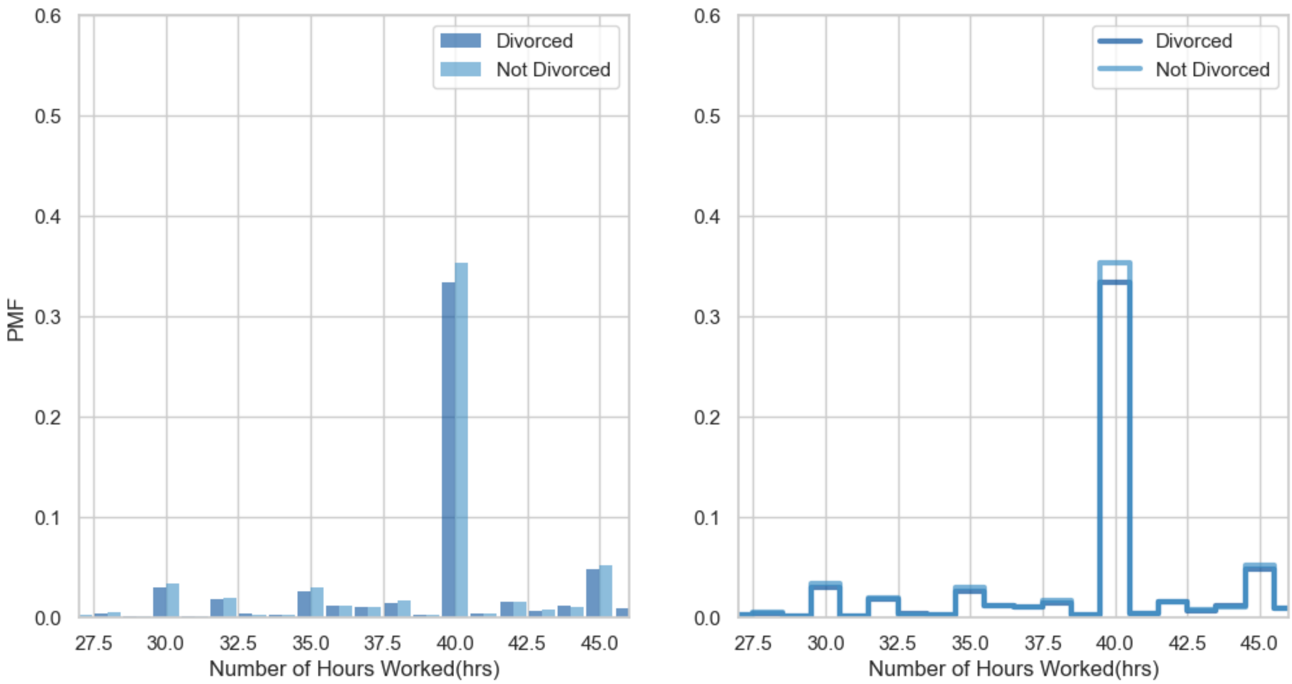

Distribution of race
8000
6000
4000
2000
0
Black
White
Other


Distribution of family16
6000
5000
4000
3000
2000
1000
0
MOTHER
FEMALE RELATIVE
MOTHER & FATHER
M AND F RELATIVES
FATHER
MALE RELATIVE
MOTHER & STPFATHER
FATHER & STPMOTHER
OTHER


Distribution of wrkslf
8000
6000
4000
2000
0
Someone else
Self-employed

### Two scenario comparison by using PMF

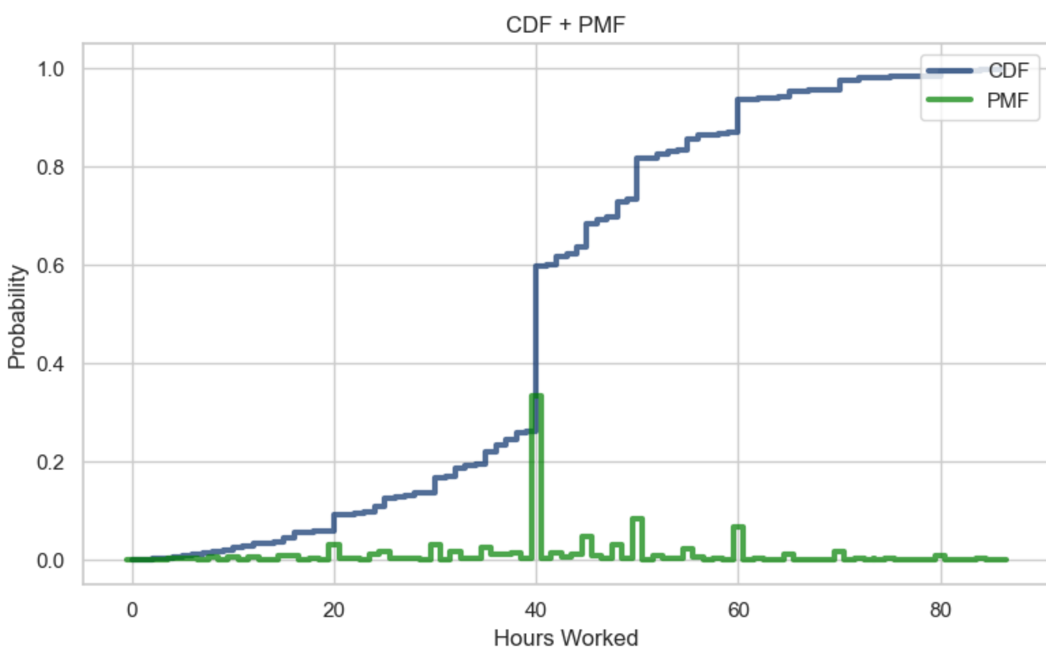
In [57]:
```
ds_divorced = dataset_gss[dataset_gss.divorce == 'YES']
ds_NOTdivorced = dataset_gss[dataset_gss.divorce != 'YES']

first_pmf = thinkstats2.Pmf(ds_divorced['hrs1'], label="Divorced")
other_pmf = thinkstats2.Pmf(ds_NOTdivorced['hrs1'], label="Not Divorced")

width = 0.45
axis = [27, 46, 0, 0.6]
thinkplot.PrePlot(2, cols=2)
thinkplot.Hist(first_pmf, align="right", width=width)
thinkplot.Hist(other_pmf, align="left", width=width)
thinkplot.Config(xlabel="Number of Hours Worked(hrs)", ylabel="PMF", axis=axis)

thinkplot.PrePlot(2)
thinkplot.SubPlot(2)
thinkplot.Pmfs([first_pmf, other_pmf])
thinkplot.Config(xlabel="Number of Hours Worked(hrs)", axis=axis)
```



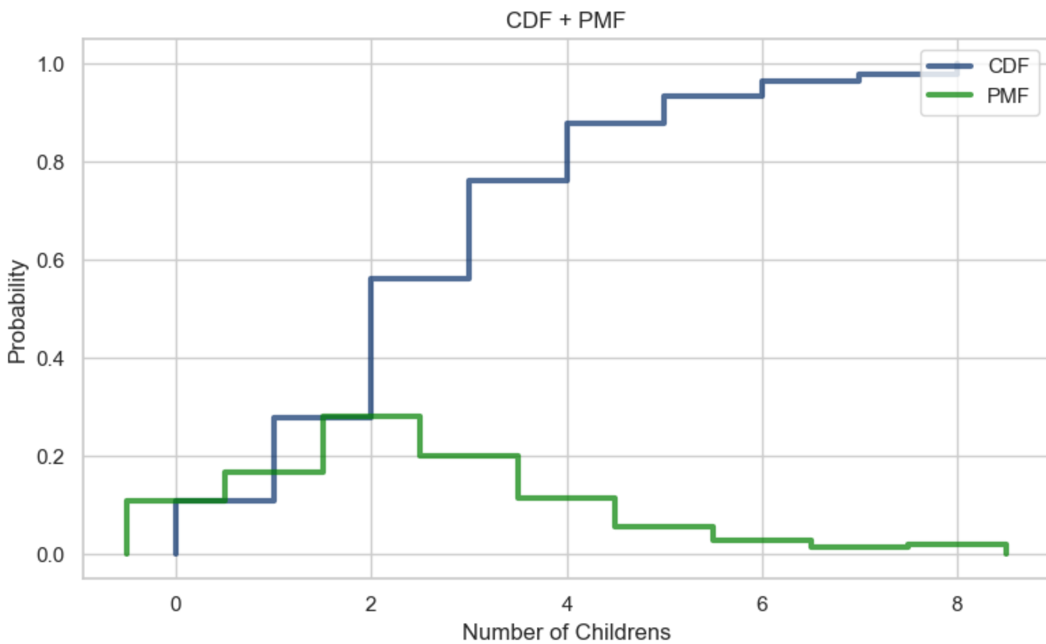
### Calculating PMF and CDF

In [58]:
```
def PlotCDFAndPMF(dataset, xlabel):
    cdf = thinkstats2.Cdf(dataset)

    #     Calculate PMF
    pmf = thinkstats2.Pmf(dataset)

    # Create a figure with two subplots
    fig, (ax1) = plt.subplots(1, 1, figsize=(8, 5))

    # Plot CDF on the first subplot
    thinkplot.Cdf(cdf, label='CDF')
    ax1.set_xlabel(xlabel)
    ax1.set_title("CDF + PMF")
    ax1.set_ylabel("Probability")

    # Plot PMF on the second subplot
    thinkplot.Pmf(pmf, label='PMF', color='green')

    # Adjust layout and display the plots
    lines, labels = ax1.get_legend_handles_labels()
    #lines2, labels2 = ax2.get_legend_handles_labels()
    ax1.legend(lines , labels , loc="upper right")

    plt.tight_layout()
    plt.show()
```

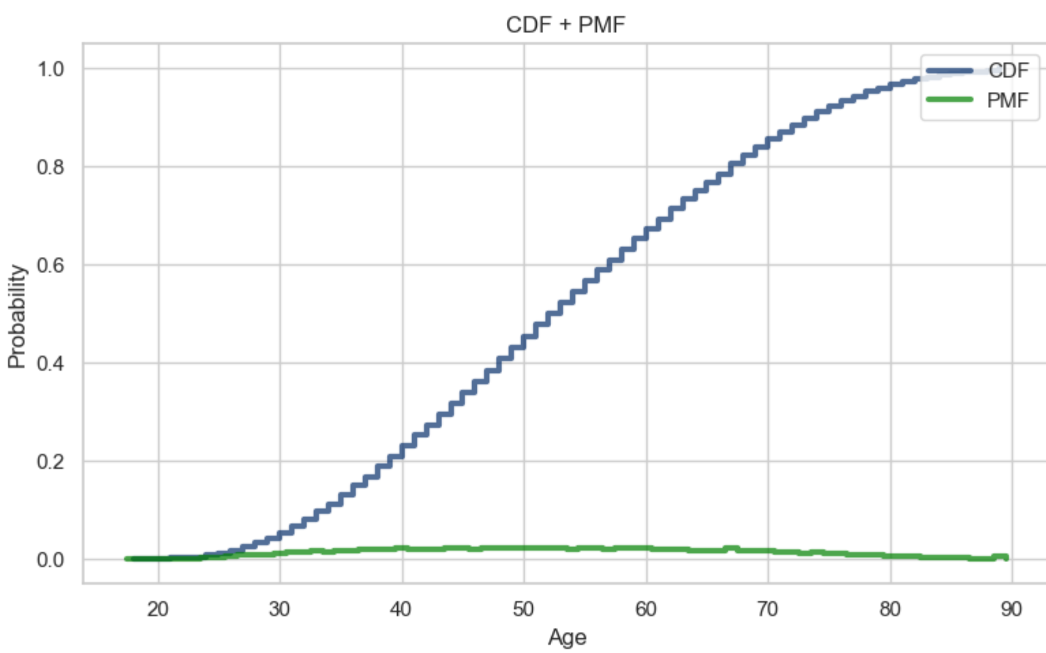
In [59]:
```
PlotCDFAndPMF(ds_divorced['hrs1'],"Hours Worked")
```



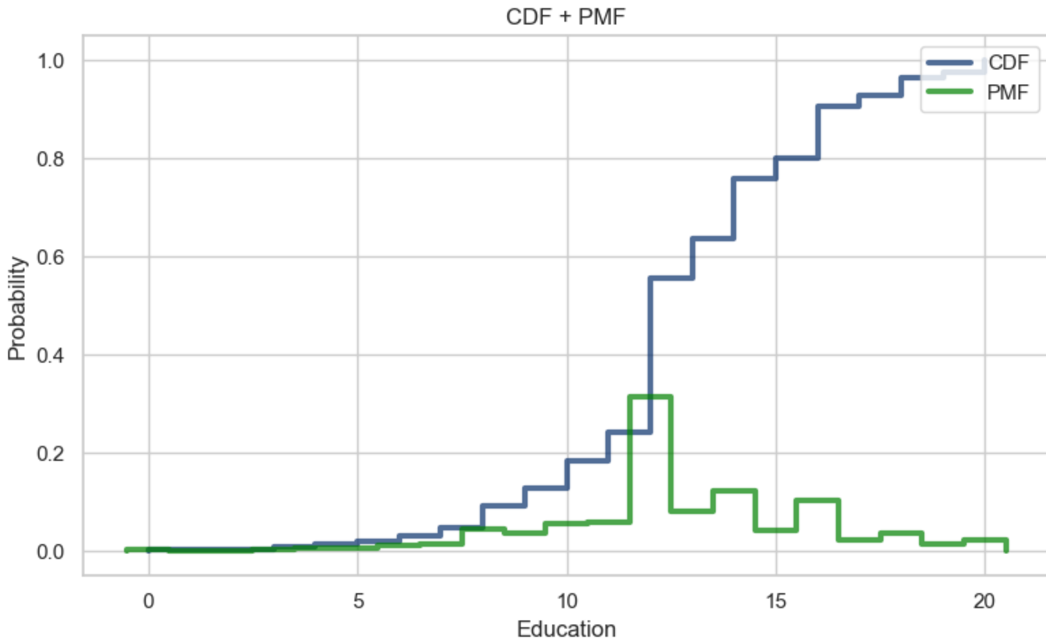
In [60]:
```
#['hrs1', 'wrkslf', 'spwrksta', 'childs', 'age', 'educ', 'sex', 'race', 'family16', 'income']
PlotCDFAndPMF(ds_divorced['childs'],"Number of Childrens")
#cdf = thinkstats2.Cdf(ds_divorced['childs'])
#thinkplot.Cdf(cdf)
#thinkplot.Show(xlabel="Number of Childrens")
```



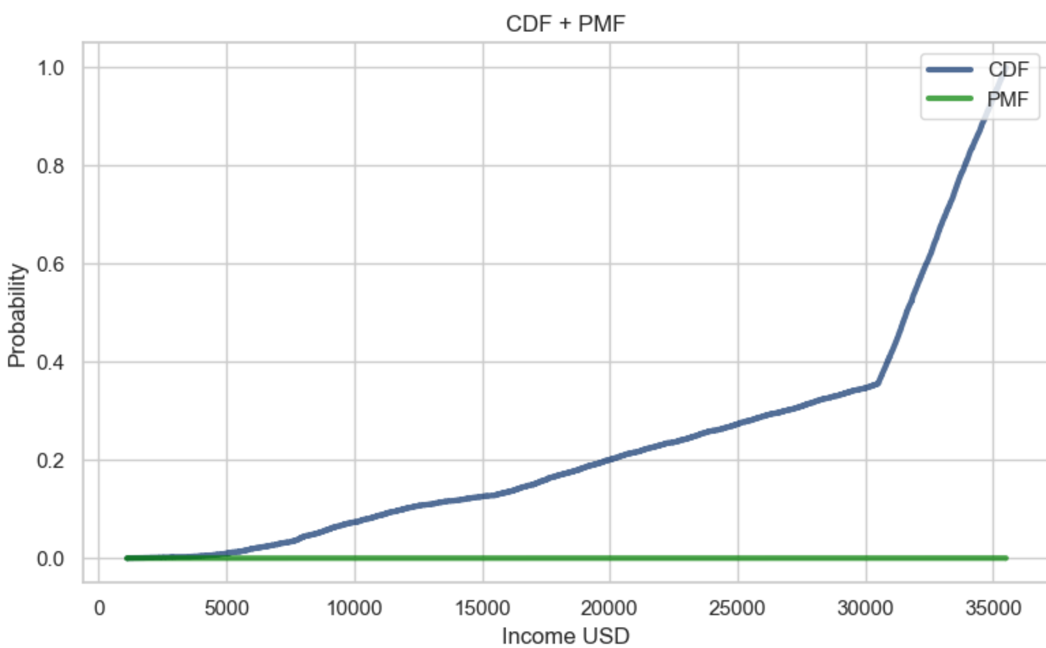

In [61]:
```
#['hrs1', 'wrkslf', 'spwrksta', 'childs', 'age', 'educ', 'sex', 'race', 'family16', 'income']

#cdf = thinkstats2.Cdf(ds_divorced['age'])
#thinkplot.Cdf(cdf)
#thinkplot.Show(xlabel="Age")
PlotCDFAndPMF(ds_divorced['age'],"Age")
```



In [62]:
```
#['hrs1', 'wrkslf', 'spwrksta', 'childs', 'age', 'educ', 'sex', 'race', 'family16', 'income']

PlotCDFAndPMF(ds_divorced['educ'],"Education")
```



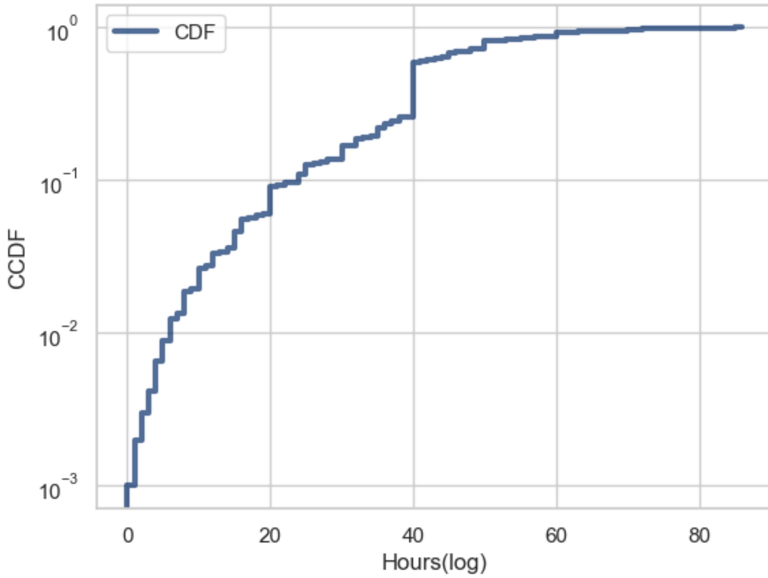

In [63]:
```
PlotCDFAndPMF(ds_divorced['incomeUSD'],"Income USD")
```




### Analytical distribution

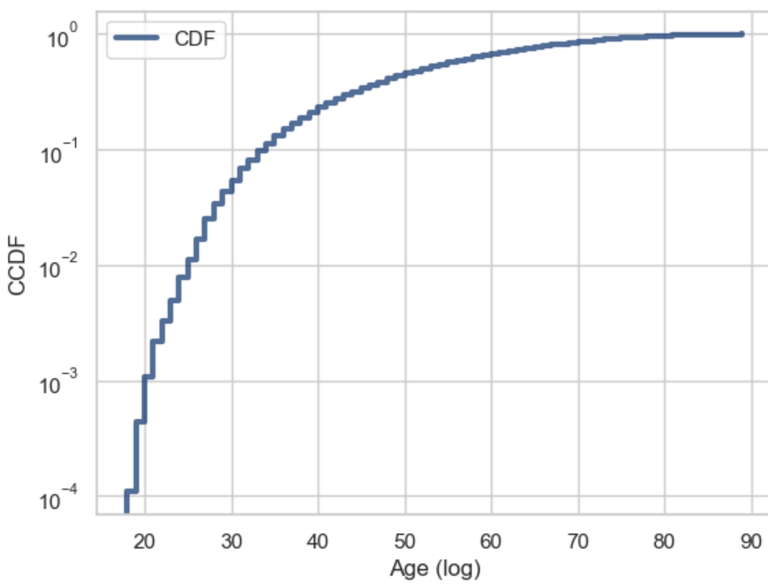
In [64]:
```
def PlotCDFAnalytical(dataset, xlabel):
    cdf = thinkstats2.Cdf(dataset)
    thinkplot.Cdf(cdf, label='CDF')
    thinkplot.Show(xlabel=xlabel, ylabel='CCDF', yscale='log')
```

In [65]:
```
PlotCDFAnalytical(ds_divorced['hrs1'],"Hours(log)")
```



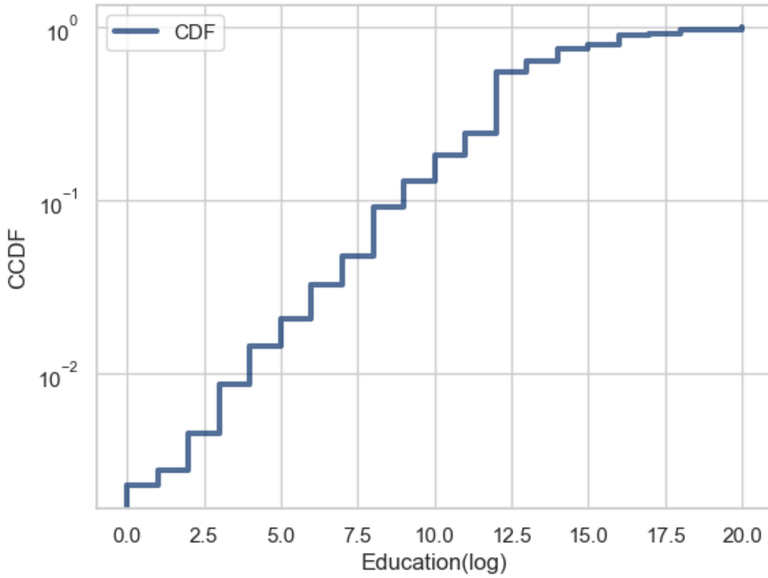

```
<Figure size 800x600 with 0 Axes>
```

In [66]:
```
PlotCDFAnalytical(ds_divorced['age'],"Age (log)")
```




```
<Figure size 800x600 with 0 Axes>
```

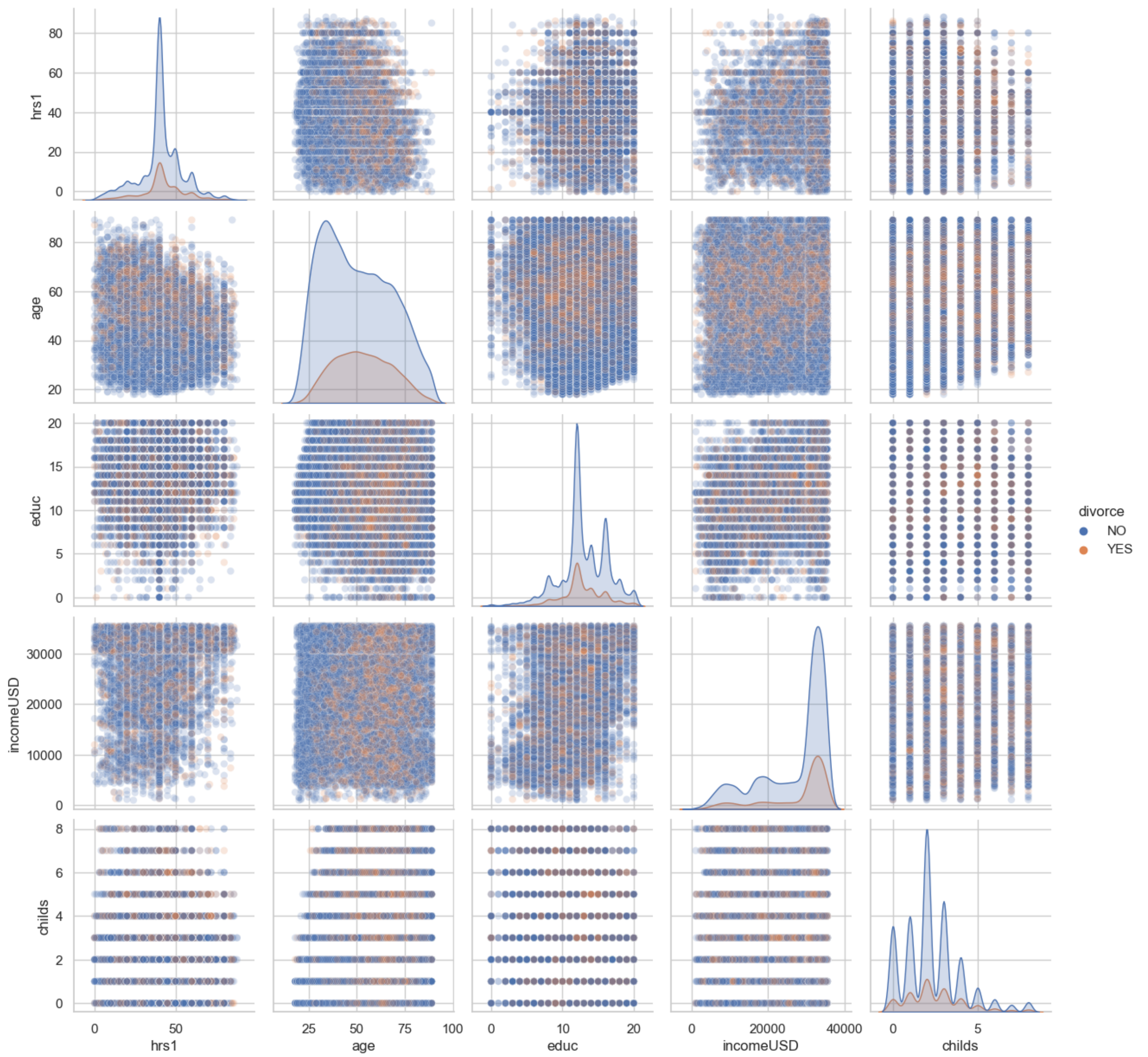
In [67]:
```
PlotCDFAnalytical(ds_divorced['educ'],"Education(log)")
```



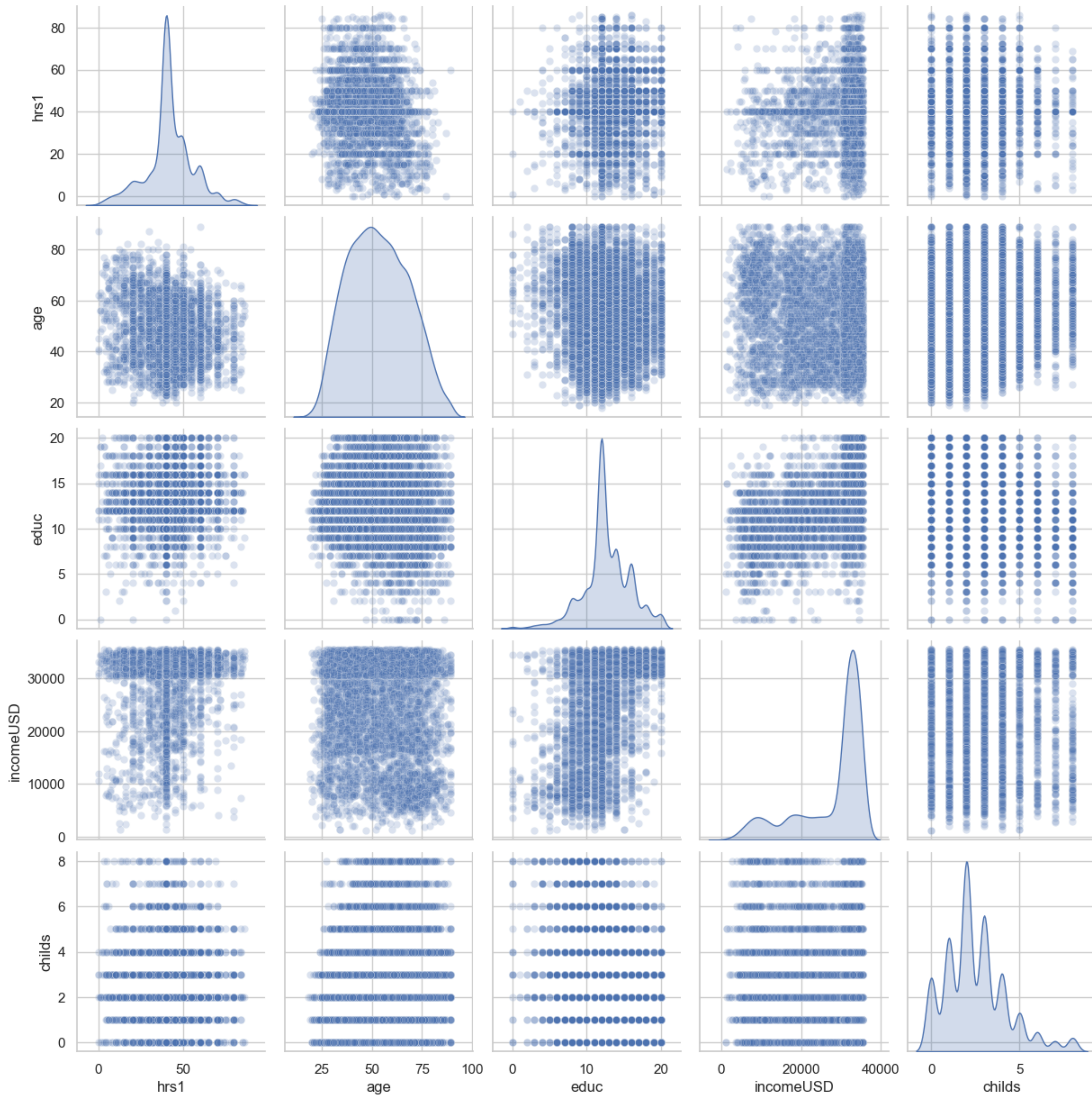

```
<Figure size 800x600 with 0 Axes>
```

### Scatter plot of different variables

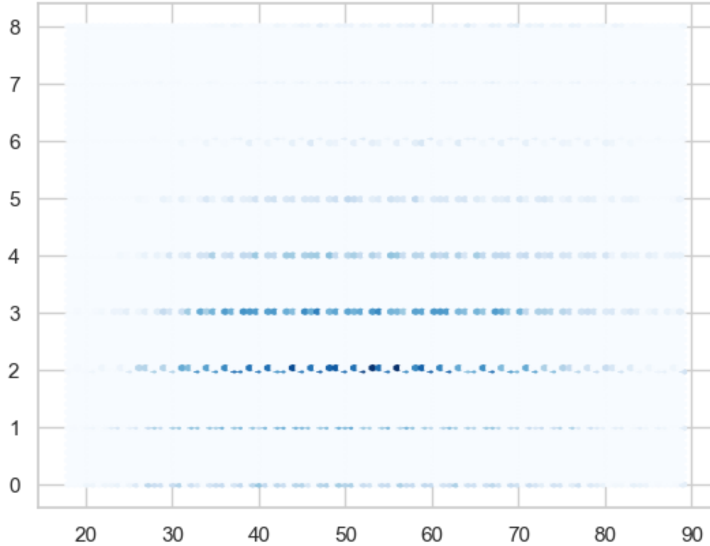
In [68]:
```
#Plotting scatter plot of different variables
#the blue shows not divorced, the orange shows the divorce

# Load the dataset
dataset = dataset_gss[['divorce','hrs1', 'age', 'educ',  'incomeUSD','childs', 'spwrksta',
                 'sex', 'race', 'family16','wrkslf']]

df = dataset.copy()

# Set the style for the plots (optional, just for aesthetics)
sns.set(style='whitegrid')

# Create a pair plot
sns.pairplot(df,  diag_kind='kde', hue='divorce',  plot_kws={'alpha': 0.2})

# Show the plot
plt.show()
```



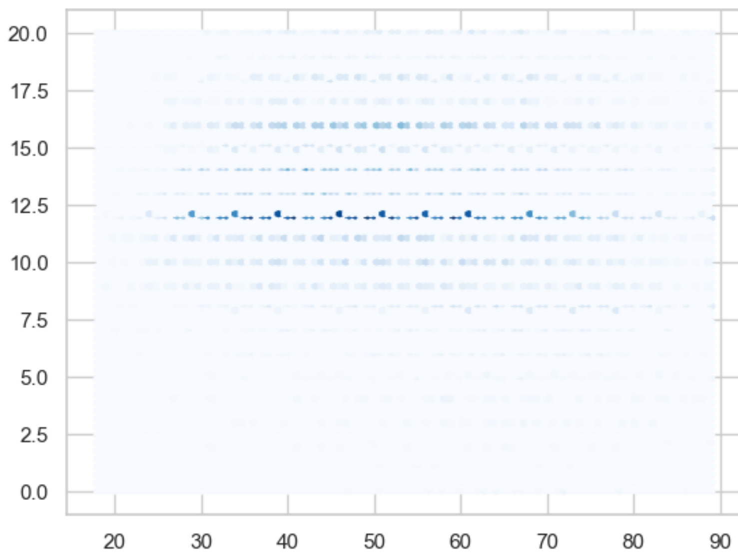
In [69]:
```
#Plotting scatter plot of divorce  variables

dataset = ds_divorced[['divorce','hrs1', 'age', 'educ',  'incomeUSD','childs', 'spwrksta',
                 'sex', 'race', 'family16','wrkslf']]

df = dataset

# Set the style for the plots (optional, just for aesthetics)
sns.set(style='whitegrid')

# Create a pair plot
sns.pairplot(df,  diag_kind='kde',  plot_kws={'alpha': 0.2})

# Show the plot
plt.show()
```



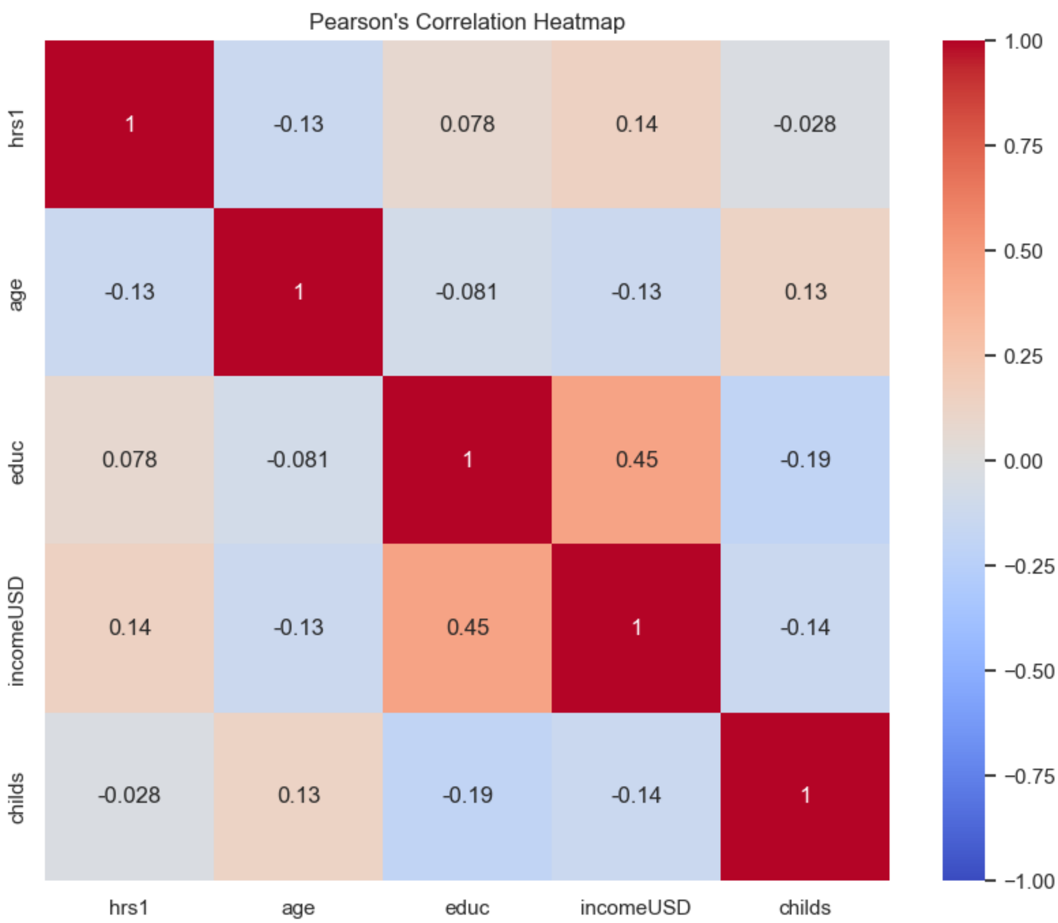
In [70]:
```
#Hexbin plot, that shows the age and childs relationship for divorced couples
thinkplot.HexBin(ds_divorced['age'],ds_divorced['childs'])
#[['divorce','hrs1', 'age', 'educ',  'incomeUSD','childs', 'spwrksta',
#                 'sex', 'race', 'family16','wrkslf']]
```




In [71]:
```
#Hexbin plot, that shows the age and childs relationship for divorced couples
thinkplot.HexBin(ds_divorced['age'],ds_divorced['educ'])
#[['divorce','hrs1', 'age', 'educ',  'incomeUSD','childs', 'spwrksta',
#                 'sex', 'race', 'family16','wrkslf']]
```



In [72]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import pearsonr
import statsmodels.api as sm

# Load the dataset
dataset = ds_divorced[[ 'hrs1', 'age', 'educ', 'incomeUSD', 'childs']]

# Compute Pearson's correlation matrix
correlation_matrix = dataset.corr()

# Display the correlation matrix as a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title("Pearson's Correlation Heatmap")
plt.show()
```




**Running OLS regression**

In [73]:
```
# Perform regression analysis to assess relationships

dataset_analysis=dataset_gss.copy()
# Convert 'divorce' to integer

# Convert 'divorce' to numeric type and fill NaN with a placeholder value
dataset_analysis['divorce'] = pd.to_numeric(dataset_analysis['divorce'], errors='coerce')
dataset_analysis['divorce'] = dataset_analysis['divorce'].fillna(-999).astype(int)

# Perform regression analysis to assess relationships
y = dataset_analysis['divorce']

X = dataset_analysis[['hrs1', 'age', 'educ', 'incomeUSD', 'childs']]
X = X.replace([np.inf, -np.inf], np.nan).fillna(0)
X = sm.add_constant(X)  # Add a constant term for the intercept


model = sm.OLS(y, X).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                divorce   R-squared:                        -inf
Model:                            OLS   Adj. R-squared:                   -inf
Method:                 Least Squares   F-statistic:                -1.448e+04
Date:                Sat, 12 Aug 2023   Prob (F-statistic):               1.00
Time:                        11:50:28   Log-Likelihood:             1.8772e+06
No. Observations:               72390   AIC:                        -3.754e+06
Df Residuals:                   72384   BIC:                        -3.754e+06
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       -999.0000   2.65e-14  -3.78e+16      0.000    -999.000    -999.000
hrs1       -2.043e-14   2.42e-16    -84.341      0.000   -2.09e-14      -2e-14
age          1.41e-14   3.01e-16     46.823      0.000    1.35e-14    1.47e-14
educ        1.066e-13   1.65e-15     64.628      0.000    1.03e-13     1.1e-13
incomeUSD   4.033e-17   4.43e-19     90.961      0.000    3.95e-17    4.12e-17
childs      5.063e-14   3.03e-15     16.690      0.000    4.47e-14    5.66e-14
==============================================================================
Omnibus:                       35.174   Durbin-Watson:                   0.565
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               38.698
Skew:                           0.013   Prob(JB):                     3.95e-09
Kurtosis:                       3.110   Cond. No.                     1.39e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.39e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

C:\ProgramData\anaconda3\lib\site-packages\statsmodels\regression\linear_model.py:1752: RuntimeWarning: divide by zero encou
ntered in double_scalars
  return 1 - self.ssr/self.centered_tss
```

### The Null Hypothesis

**The Null hypothesis 1: There is no co-rellation between divorce and the number of childrens**

In [74]:
```
# Create a DataFrame
df = dataset_gss.copy()

# Create a contingency table
contingency_table = pd.crosstab(df['divorce'], df['childs'])

# Perform the Chi-Square test
chi2, p, dof, expected = chi2_contingency(contingency_table)

# Define significance level
alpha = 0.05

# Print the results
print("\nChi-Square:", chi2)
print("p-value:", p)
print("Degrees of Freedom:", dof)

# Compare p-value with significance level
if p < alpha:
    print("\nReject the null hypothesis. There is a significant correlation between divorce and the number of children.")
else:
    print("\nFail to reject the null hypothesis. There is no significant correlation between divorce and the number of childre
```

```
Chi-Square: 212.3517343739779
p-value: 1.5873059852030754e-41
Degrees of Freedom: 8

Reject the null hypothesis. There is a significant correlation between divorce and the number of children.
```

## Reject the null hypothesis. There is a significant correlation between divorce and the number of children.

**The Null hypothesis 2: There is no co-rellation between divorce and the number of hours worked in a week**

In [75]:
```
# Create a DataFrame
#[ 'hrs1', 'age', 'educ', 'incomeUSD', 'childs']
# Create a DataFrame
df = dataset_gss.copy()

# Create a contingency table
contingency_table = pd.crosstab(df['divorce'], df['hrs1'])

# Perform the Chi-Square test
chi2, p, dof, expected = chi2_contingency(contingency_table)

# Define significance level
alpha = 0.05

# Print the results
print("\nChi-Square:", chi2)
print("p-value:", p)
print("Degrees of Freedom:", dof)

# Compare p-value with significance level
if p < alpha:
    print("\nReject the null hypothesis. There is a significant correlation between divorce and the number of hours worked.")
else:
    print("\nFail to reject the null hypothesis. There is no significant correlation between divorce and the number of hours w
```

```
Chi-Square: 103.43880050111484
p-value: 0.12476633823367769
Degrees of Freedom: 88

Fail to reject the null hypothesis. There is no significant correlation between divorce and the number of hours worked.
```

**<u>Fail to reject the null hypothesis. There is no significant correlation between divorce and the number of hours worked.</u>**

## Logistic Regression

In [76]:
```
# Check if the column is numeric
def is_numeric(column):
    return pd.api.types.is_numeric_dtype(column)

# Impute the missing values
def impute_missing_values(df):
    for column in df.columns:
        if is_numeric(column):
            imputer = SimpleImputer(missing_values='NaN', strategy='mean')
            df[column] = imputer.fit_transform(df[column].values.reshape(-1, 1))
        else:
            #df[column] = df[column].fillna('missing')
            a=2

# Impute the missing values in the dataset
```

In [77]:
```
dataset_gss
```

Out[77]:

| | year | id_ | hrs1 | wrkslf | marital | divorce | spwrksta | childs | age | educ | spdeg | sex | race | family16 | income | relig16 | hapmar | ba |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1972 | 1 | NaN | Someone else | Never married | NaN | NaN | 0.0 | 23.0 | 16.0 | NaN | FEMALE | White | FATHER | NaN | NaN | NaN | < |
| **1** | 1972 | 2 | NaN | Someone else | Married | NO | KEEPING HOUSE | 5.0 | 70.0 | 10.0 | HIGH SCHOOL | MALE | White | M AND F RELATIVES | NaN | NaN | NaN | < |
| **2** | 1972 | 3 | NaN | Someone else | Married | NO | WORKING FULLTIME | 4.0 | 48.0 | 12.0 | NaN | FEMALE | White | MOTHER & FATHER | NaN | NaN | NaN | < |
| **3** | 1972 | 4 | NaN | Someone else | Married | NO | WORKING FULLTIME | 0.0 | 27.0 | 17.0 | GRADUATE | FEMALE | White | MOTHER & FATHER | NaN | NaN | NaN | < |
| **4** | 1972 | 5 | NaN | Someone else | Married | NO | TEMP NOT WORKING | 2.0 | 61.0 | 12.0 | HIGH SCHOOL | FEMALE | White | MOTHER & FATHER | NaN | NaN | NaN | < |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **72385** | 2022 | 3541 | 48.0 | Someone else | Never married | NaN | NaN | 0.0 | 22.0 | 12.0 | NaN | FEMALE | White | MOTHER & STPFATHER | $25,000 or more | NaN | NaN | B |
| **72386** | 2022 | 3542 | 50.0 | Someone else | Married | YES | NaN | 2.0 | 29.0 | 19.0 | NaN | FEMALE | White | MOTHER & FATHER | $25,000 or more | NaN | VERY HAPPY | B |
| **72387** | 2022 | 3543 | 38.0 | Someone else | Never married | NaN | NaN | 1.0 | 32.0 | 15.0 | NaN | MALE | White | MOTHER & STPFATHER | $25,000 or more | NaN | NaN | B |
| **72388** | 2022 | 3544 | 40.0 | Someone else | Married | NO | NaN | 0.0 | 49.0 | 17.0 | NaN | FEMALE | White | MOTHER & FATHER | $25,000 or more | NaN | VERY HAPPY | B |
| **72389** | 2022 | 3545 | 40.0 | Someone else | Married | NO | NaN | 1.0 | 50.0 | 20.0 | NaN | MALE | White | MOTHER & FATHER | $25,000 or more | NaN | PRETTY HAPPY | B |

72390 rows × 19 columns

In [78]:
```
# Create the dependent variable
df = dataset_gss.copy()
# Create the independent variables
df = df.drop('id_', axis=1)
df = df.drop('ballot', axis=1)
df = df.drop('year', axis=1)
#df = df.drop('marital', axis=1)



data = df
# Impute NaN values with the mean for numeric columns and the mode for categorical columns
for col in data.columns:
    if data[col].dtype.name == "float64":
        data[col].fillna(data[col].mean(), inplace=True)
    elif not pd.api.types.is_numeric_dtype(data[col]):
        data = pd.get_dummies(data, columns=[col])
        #print(col)

        # Split the data into training and testing sets
data = data.drop('divorce_NO', axis=1)
#X_train, X_test, y_train, y_test = train_test_split(data.drop(columns="marital_Divorced"), data["marital_Divorced"], test_siz
X_train, X_test, y_train, y_test = train_test_split(data.drop(columns="divorce_YES"), data["divorce_YES"], test_size=0.25)


# Create a logistic regression model
log_reg = LogisticRegression(max_iter =10000)

# Fit the model to the training data
result= log_reg.fit(X_train, y_train)

# Make predictions on the testing data
predictions = log_reg.predict(X_test)

# Calculate the accuracy of the model
accuracy = np.mean(predictions == y_test)

print("The accuracy of the model is:", accuracy)
```

```
The accuracy of the model is: 0.8681622278704829
```

In [79]:
```
import statsmodels.api as sm
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge
# Scale the input features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Create a Ridge regression model with alpha parameter for regularization
ridge_reg = Ridge(alpha=1.0)
ridge_reg.fit(X_train_scaled, y_train)

# Print coefficients and summary
#print("Ridge Regression Coefficients:")
#for col, coef in zip(X_train.columns, ridge_reg.coef_):
#    print(f"{col}: {coef:.4f}")


# Using statsmodels to fit the Ridge regression model and print summary
X_train_scaled_with_const = sm.add_constant(X_train_scaled)
ridge_reg_sm = sm.OLS(y_train, X_train_scaled_with_const)
result_sm = ridge_reg_sm.fit()

# Print the summary including coefficients and p-values
print(result_sm.summary())

# Make predictions on the testing data
X_test_scaled_with_const = sm.add_constant(X_test_scaled)
y_pred = result_sm.predict(X_test_scaled_with_const)
y_pred_binary = np.round(y_pred)  # Convert predicted probabilities to binary predictions

# Calculate accuracy
accuracy = np.mean(y_pred_binary == y_test)
print("Accuracy:", accuracy)

# Calculate precision
precision = np.sum((y_pred_binary == 1) & (y_test == 1)) / np.sum(y_pred_binary == 1)
print("Precision:", precision)

# List column names with their p-values
#p_values = result_sm.pvalues[1:]  # Exclude the constant term
#print("\nColumn Names with P-values:")
#for col, p_value in zip(X_train.columns, p_values):
#    print(f"{col}: p-value={p_value:.4f}")
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:            divorce_YES   R-squared:                       0.116
Model:                            OLS   Adj. R-squared:                  0.115
Method:                 Least Squares   F-statistic:                     106.3
Date:                Sat, 12 Aug 2023   Prob (F-statistic):               0.00
Time:                        11:50:30   Log-Likelihood:                -14430.
No. Observations:               54292   AIC:                         2.900e+04
Df Residuals:                   54224   BIC:                         2.960e+04
Df Model:                          67                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1295      0.001     95.526      0.000       0.127       0.132
x1             0.0055      0.001      3.956      0.000       0.003       0.008
x2             0.0133      0.002      8.299      0.000       0.010       0.016
x3             0.0211      0.002     11.360      0.000       0.017       0.025
x4            -0.0081      0.002     -4.875      0.000      -0.011      -0.005
x5             0.0076      0.009      0.859      0.391      -0.010       0.025
x6             0.0207      0.002      9.018      0.000       0.016       0.025
x7             0.0195      0.002      8.553      0.000       0.015       0.024
x8            -0.0203      0.017     -1.196      0.232      -0.053       0.013
x9             0.0954      0.026      3.693      0.000       0.045       0.146
x10           -0.0082      0.021     -0.392      0.695      -0.049       0.033
x11           -0.0100      0.009     -1.095      0.273      -0.028       0.008
x12            0.0425      0.015      2.902      0.004       0.014       0.071
x13           -0.0157      0.005     -3.204      0.001      -0.025      -0.006
x14            0.0069      0.002      3.282      0.001       0.003       0.011
x15           -0.0105      0.004     -2.377      0.017      -0.019      -0.002
x16           -0.0020      0.002     -1.095      0.274      -0.006       0.002
x17        -1.436e-05      0.002     -0.007      0.995      -0.004       0.004
x18            0.0043      0.002      1.937      0.053   -5.03e-05       0.009
x19           -0.0036      0.008     -0.467      0.641      -0.019       0.011
x20           -0.0056      0.004     -1.493      0.135      -0.013       0.002
x21           -0.0034      0.003     -1.309      0.190      -0.008       0.002
x22           -0.0185      0.004     -4.784      0.000      -0.026      -0.011
x23           -0.0121      0.003     -3.918      0.000      -0.018      -0.006
x24           -0.0047      0.006     -0.775      0.438      -0.016       0.007
x25           -0.0121      0.004     -2.912      0.004      -0.020      -0.004
x26           -0.0527      0.017     -3.032      0.002      -0.087      -0.019
x27           -0.0508      0.017     -2.916      0.004      -0.085      -0.017
x28            0.0016      0.012      0.135      0.893      -0.022       0.026
x29           -0.0024      0.008     -0.282      0.778      -0.019       0.014
x30            0.0044      0.014      0.314      0.754      -0.023       0.032
x31           -0.0005      0.002     -0.219      0.826      -0.005       0.004
x32            0.0004      0.002      0.209      0.835      -0.004       0.005
x33            0.0007      0.002      0.328      0.743      -0.003       0.005
x34            0.0028      0.002      1.255      0.209      -0.002       0.007
x35           -0.0008      0.002     -0.509      0.611      -0.004       0.002
x36            0.0030      0.004      0.688      0.492      -0.006       0.011
x37           -0.0101      0.006     -1.751      0.080      -0.021       0.001
x38            0.0073      0.003      2.409      0.016       0.001       0.013
x39            0.0100      0.002      4.257      0.000       0.005       0.015
x40            0.0037      0.003      1.171      0.242      -0.002       0.010
x41            0.0007      0.003      0.249      0.803      -0.005       0.007
x42            0.0028      0.002      1.476      0.140      -0.001       0.006
x43            0.0016      0.002      0.907      0.364      -0.002       0.005
x44            0.0156      0.004      3.675      0.000       0.007       0.024
x45            0.0038      0.003      1.275      0.202      -0.002       0.010
x46            0.0007      0.003      0.254      0.799      -0.005       0.006
x47            0.0009      0.003      0.337      0.736      -0.004       0.006
x48            0.0030      0.003      1.194      0.232      -0.002       0.008
x49            0.0009      0.002      0.347      0.728      -0.004       0.006
x50            0.0007      0.003      0.259      0.796      -0.005       0.006
x51            0.0039      0.003      1.382      0.167      -0.002       0.010
x52           -0.0004      0.001     -0.308      0.758      -0.003       0.002
x53           -0.0068      0.003     -2.290      0.022      -0.013      -0.001
x54            0.0008      0.001      0.588      0.556      -0.002       0.004
x55           -0.0024      0.001     -1.699      0.089      -0.005       0.000
x56            0.0006      0.001      0.404      0.686      -0.002       0.003
x57           -0.0026      0.002     -1.632      0.103      -0.006       0.001
x58            0.0005      0.001      0.391      0.696      -0.002       0.003
x59            0.0009      0.001      0.639      0.523      -0.002       0.004
x60            0.0028      0.002      1.480      0.139      -0.001       0.007
x61            0.0006      0.001      0.445      0.656      -0.002       0.003
x62           -0.0017      0.001     -1.153      0.249      -0.005       0.001
x63           -0.0012      0.001     -0.847      0.397      -0.004       0.002
x64            0.0104      0.003      3.215      0.001       0.004       0.017
x65            0.0100      0.002      6.249      0.000       0.007       0.013
x66            0.0044      0.003      1.511      0.131      -0.001       0.010
x67            0.0008      0.003      0.229      0.819      -0.006       0.008
==============================================================================
Omnibus:                    15858.606   Durbin-Watson:                   2.006
```

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            34941.170
Skew:                           1.740   Prob(JB):                         0.00
Kurtosis:                       4.825   Cond. No.                         60.3
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Accuracy: 0.8709249640844292
Precision: nan

C:\Users\aniruddha.joshi\AppData\Local\Temp\ipykernel_25488\2104285088.py:37: RuntimeWarning: invalid value encountered in l
onglong_scalars
  precision = np.sum((y_pred_binary == 1) & (y_test == 1)) / np.sum(y_pred_binary == 1)
```

In [80]:
```
from tabulate import tabulate

# ... (previous code) ...

# List column names with their p-values, coefficients, and R-squared
p_values = result_sm.pvalues[1:]  # Exclude the constant term
coefficients = result_sm.params[1:]  # Exclude the constant term
r_squared = result_sm.rsquared

# Prepare data for tabular format
table_data = []
for col, coef, p_value in zip(X_train.columns, coefficients, p_values):
    table_data.append([col, coef, p_value])

# Add R-squared to the table
table_data.append(['R-squared', r_squared, ''])

# Print column names, coefficients, p-values, and R-squared in tabular format
headers = ["Column Name", "Coefficient", "P-value"]
print("\nSummary:")
print(tabulate(table_data, headers=headers, floatfmt=(".4f", ".4f", ".4f")))
```

```
Summary:
Column Name                      Coefficient  P-value
------------------------------  -------------  ----------------------
hrs1                                   0.0055  7.620459765753077e-05
childs                                 0.0133  1.0707931742194831e-16
age                                    0.0211  7.170395867519049e-30
educ                                  -0.0081  1.0892364007125017e-06
incomeUSD                              0.0076  0.39057106719663826
wrkslf_Self-employed                   0.0207  1.969026992589378e-19
wrkslf_Someone else                    0.0195  1.2257570155047025e-17
marital_Divorced                      -0.0203  0.23154110157380986
marital_Married                        0.0954  0.00022199688382626233
marital_Never married                 -0.0082  0.6954071549873602
marital_Separated                     -0.0100  0.2734943796529368
marital_Widowed                        0.0425  0.003707304727193523
spwrksta_KEEPING HOUSE                -0.0157  0.001355395561627798
spwrksta_OTHER                         0.0069  0.0010331660646010002
spwrksta_RETIRED                      -0.0105  0.017449143709530247
spwrksta_SCHOOL                       -0.0020  0.2737100273559191
spwrksta_TEMP NOT WORKING             -0.0000  0.9946821541930824
spwrksta_UNEMPL, LAID OFF              0.0043  0.052722356874633225
spwrksta_WORKING FULLTIME             -0.0036  0.6407888968926208
spwrksta_WORKING PARTTIME             -0.0056  0.13543520459486624
spdeg_ASSOCIATE/JUNIOR COLLEGE        -0.0034  0.19043024829369867
spdeg_BACHELOR                        -0.0185  1.7202075529369903e-06
spdeg_GRADUATE                        -0.0121  8.932172580960702e-05
spdeg_HIGH SCHOOL                     -0.0047  0.4382041232576087
spdeg_LT HIGH SCHOOL                  -0.0121  0.0035876637112035345
sex_FEMALE                            -0.0527  0.002428176600103502
sex_MALE                              -0.0508  0.0035514124097361945
race_Black                             0.0016  0.8926920953041122
race_Other                            -0.0024  0.7777678993970554
race_White                             0.0044  0.7537838933933967
family16_FATHER                       -0.0005  0.8262792298330345
family16_FATHER & STPMOTHER            0.0004  0.8348212956813824
family16_FEMALE RELATIVE               0.0007  0.742852189237496
family16_M AND F RELATIVES             0.0028  0.20943306531849945
family16_MALE RELATIVE                -0.0008  0.610893507412598
family16_MOTHER                        0.0030  0.4916856459009773
family16_MOTHER & FATHER              -0.0101  0.07992793549019443
family16_MOTHER & STPFATHER            0.0073  0.015978661101395963
family16_OTHER                         0.0100  2.072576362731709e-05
income_$1,000 to $2,999                0.0037  0.24172531458132138
income_$10,000 to $14,999              0.0007  0.8032990284171928
income_$15,000 to $19,999              0.0028  0.1400385808857833
income_$20,000 to $24,999              0.0016  0.36441982611061063
income_$25,000 or more                 0.0156  0.0002382057572925851
income_$3,000 to $3,999                0.0038  0.2023103669463342
income_$4,000 to $4,999                0.0007  0.7994139869948655
income_$5,000 to $5,999                0.0009  0.7358917650276431
income_$6,000 to $6,999                0.0030  0.2324851938528776
income_$7,000 to $7,999                0.0009  0.7283643349741542
income_$8,000 to $9,999                0.0007  0.79563483944189
income_Under $1,000                    0.0039  0.1669678928170715
relig16_BUDDHISM                      -0.0004  0.7584393042582245
relig16_CATHOLIC                      -0.0068  0.021997904747485846
relig16_CHRISTIAN                      0.0008  0.5562945876407357
relig16_HINDUISM                      -0.0024  0.08935492394359891
relig16_INTER-NONDENOMINATIONAL        0.0006  0.6859964983459588
relig16_JEWISH                        -0.0026  0.10274141372535493
relig16_MUSLIM/ISLAM                   0.0005  0.6955463804879863
relig16_NATIVE AMERICAN                0.0009  0.5225048231474876
relig16_NONE                           0.0028  0.13895096742026966
relig16_ORTHODOX-CHRISTIAN             0.0006  0.6560301866995284
relig16_OTHER                         -0.0017  0.24892654177831486
relig16_OTHER EASTERN                 -0.0012  0.3971148018798719
relig16_PROTESTANT                     0.0104  0.0013031136007823194
hapmar_NOT TOO HAPPY                   0.0100  4.15132086125714e-10
hapmar_PRETTY HAPPY                    0.0044  0.13066791693248
hapmar_VERY HAPPY                      0.0008  0.818735067547654
R-squared                              0.1161
```

In [ ]: