In [72]:
```
import os
import pandas as pd
import numpy as np
```

1. Load the datasets in Pandas dataframe

In [73]:
```
x = pd.read_csv(r'C:\Users\muham\Downloads\test_labels.csv')
y = pd.read_csv(r'C:\Users\muham\Downloads\train_values.csv')
```

In [74]:
```
x
```

Out[74]:

	row_id	poverty_probability
0	0	0.515
1	1	0.981
2	2	0.982
3	3	0.879
4	4	0.796
...	...	...
12595	12595	0.990
12596	12596	0.950
12597	12597	0.342
12598	12598	0.846
12599	12599	0.569

12600 rows × 2 columns

In [75]:
```
y
```

Out[75]:

	row_id	country	is_urban	age	female	married	religion	relationship_to_hh_head	education_level	literacy	...	reg_formal_nbfi_account	fir

	row_id	country	is_urban	age	female	married	religion	relationship_to_hh_head	education_level	literacy	...	reg_formal_nbfi_account	fir
0	0	C	False	18	True	True	P	Other	1.0	True	...	False	
1	1	C	True	30	True	True	P	Other	1.0	True	...	False	
2	2	A	False	20	True	True	Q	Spouse	1.0	True	...	False	
3	3	A	False	61	False	True	Q	Head	0.0	False	...	False	
4	4	D	False	26	True	True	X	Spouse	1.0	True	...	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	
12595	12595	C	True	50	False	True	P	Head	1.0	True	...	False	
12596	12596	D	False	90	False	False	O	Head	0.0	True	...	False	
12597	12597	J	False	52	True	False	X	Head	1.0	False	...	False	
12598	12598	I	False	40	False	True	Q	Head	0.0	False	...	False	
12599	12599	D	True	24	False	False	X	Son/Daughter	2.0	True	...	True	

12600 rows × 59 columns

2. Join the 2 data sets using pd.merge

In [76]:
```
merged_data = pd.merge(x, y,how='left', on='row_id')
merged_data
```

Out[76]:

	row_id	poverty_probability	country	is_urban	age	female	married	religion	relationship_to_hh_head	education_level	...	reg_formal_nbfi_
0	0	0.515	C	False	18	True	True	P	Other	1.0	...	
1	1	0.981	C	True	30	True	True	P	Other	1.0	...	
2	2	0.982	A	False	20	True	True	Q	Spouse	1.0	...	
3	3	0.879	A	False	61	False	True	Q	Head	0.0	...	
4	4	0.796	D	False	26	True	True	X	Spouse	1.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	

	row_id	poverty_probability	country	is_urban	age	female	married	religion	relationship_to_hh_head	education_level	...	reg_formal_nbfi_
12595	12595	0.990	C	True	50	False	True	P	Head	1.0	...	
12596	12596	0.950	D	False	90	False	False	O	Head	0.0	...	
12597	12597	0.342	J	False	52	True	False	X	Head	1.0	...	
12598	12598	0.846	I	False	40	False	True	Q	Head	0.0	...	
12599	12599	0.569	D	True	24	False	False	X	Son/Daughter	2.0	...	

12600 rows × 60 columns

3. Check for any data type changes that you can make.

Make effective use of 3 data types
a. Numerical which can be int or float
b. Boolean which is true/false
c. Categorical

In [91]:
```
merged_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12600 entries, 0 to 12599
Data columns (total 60 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   row_id                                12600 non-null  int64
 1   poverty_probability                   12600 non-null  float64
 2   country                               12600 non-null  object
 3   is_urban                              12600 non-null  bool
 4   age                                   12600 non-null  int64
 5   female                                12600 non-null  bool
 6   married                               12600 non-null  bool
 7   religion                              12600 non-null  object
 8   relationship_to_hh_head               12600 non-null  object
 9   education_level                       12364 non-null  float64
 10  literacy                              12600 non-null  bool
 11  can_add                               12600 non-null  bool
 12  can_divide                            12600 non-null  bool
```

```
13  can_calc_percents                       12600 non-null  bool
14  can_calc_compounding                    12600 non-null  bool
15  employed_last_year                      12600 non-null  bool
16  employment_category_last_year           12600 non-null  object
17  employment_type_last_year               12600 non-null  object
18  share_hh_income_provided                12295 non-null  float64
19  income_ag_livestock_last_year           12600 non-null  bool
20  income_friends_family_last_year         12600 non-null  bool
21  income_government_last_year             12600 non-null  bool
22  income_own_business_last_year           12600 non-null  bool
23  income_private_sector_last_year         12600 non-null  bool
24  income_public_sector_last_year          12600 non-null  bool
25  num_times_borrowed_last_year            12600 non-null  int64
26  borrowing_recency                       12600 non-null  int64
27  formal_savings                          12600 non-null  bool
28  informal_savings                        12600 non-null  bool
29  cash_property_savings                   12600 non-null  bool
30  has_insurance                           12600 non-null  bool
31  has_investment                          12600 non-null  bool
32  bank_interest_rate                      289 non-null    float64
33  mm_interest_rate                        151 non-null    float64
34  mfi_interest_rate                       201 non-null    float64
35  other_fsp_interest_rate                 239 non-null    float64
36  num_shocks_last_year                    12600 non-null  int64
37  avg_shock_strength_last_year            12600 non-null  float64
38  borrowed_for_emergency_last_year        12600 non-null  bool
39  borrowed_for_daily_expenses_last_year   12600 non-null  bool
40  borrowed_for_home_or_biz_last_year      12600 non-null  bool
41  phone_technology                        12600 non-null  int64
42  can_call                                12600 non-null  bool
43  can_text                                12600 non-null  bool
44  can_use_internet                        12600 non-null  bool
45  can_make_transaction                    12600 non-null  bool
46  phone_ownership                         12600 non-null  int64
47  advanced_phone_use                      12600 non-null  bool
48  reg_bank_acct                           12600 non-null  bool
49  reg_mm_acct                             12600 non-null  bool
50  reg_formal_nbfi_account                 12600 non-null  bool
51  financially_included                    12600 non-null  bool
52  active_bank_user                        12600 non-null  bool
53  active_mm_user                          12600 non-null  bool
54  active_formal_nbfi_user                 12600 non-null  bool
55  active_informal_nbfi_user               12600 non-null  bool
56  nonreg_active_mm_user                   12600 non-null  bool
57  num_formal_institutions_last_year       12600 non-null  int64
```

```
 58  num_informal_institutions_last_year    12600 non-null  int64
 59  num_financial_activities_last_year     12600 non-null  int64
dtypes: bool(37), float64(8), int64(10), object(5)
memory usage: 2.8+ MB
```

We will drop the rows which have any 'NaN' values in it w.r.t "mm_interest_rate" as it has the lowest number of rows with non-null values.

In [92]:
```
non_null = merged_data.dropna(subset=['mm_interest_rate'])
non_null.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 151 entries, 175 to 12534
Data columns (total 60 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   row_id                                 151 non-null    int64
 1   poverty_probability                    151 non-null    float64
 2   country                                151 non-null    object
 3   is_urban                               151 non-null    bool
 4   age                                    151 non-null    int64
 5   female                                 151 non-null    bool
 6   married                                151 non-null    bool
 7   religion                               151 non-null    object
 8   relationship_to_hh_head                151 non-null    object
 9   education_level                        151 non-null    float64
 10  literacy                               151 non-null    bool
 11  can_add                                151 non-null    bool
 12  can_divide                             151 non-null    bool
 13  can_calc_percents                      151 non-null    bool
 14  can_calc_compounding                   151 non-null    bool
 15  employed_last_year                     151 non-null    bool
 16  employment_category_last_year          151 non-null    object
 17  employment_type_last_year              151 non-null    object
 18  share_hh_income_provided               151 non-null    float64
 19  income_ag_livestock_last_year          151 non-null    bool
 20  income_friends_family_last_year        151 non-null    bool
 21  income_government_last_year            151 non-null    bool
 22  income_own_business_last_year          151 non-null    bool
 23  income_private_sector_last_year        151 non-null    bool
 24  income_public_sector_last_year         151 non-null    bool
 25  num_times_borrowed_last_year           151 non-null    int64
 26  borrowing_recency                      151 non-null    int64
 27  formal_savings                         151 non-null    bool
 28  informal_savings                       151 non-null    bool
 29  cash_property_savings                  151 non-null    bool
```

```
 30  has_insurance                          151 non-null    bool
 31  has_investment                         151 non-null    bool
 32  bank_interest_rate                     24 non-null     float64
 33  mm_interest_rate                       151 non-null    float64
 34  mfi_interest_rate                      14 non-null     float64
 35  other_fsp_interest_rate                13 non-null     float64
 36  num_shocks_last_year                   151 non-null    int64
 37  avg_shock_strength_last_year           151 non-null    float64
 38  borrowed_for_emergency_last_year       151 non-null    bool
 39  borrowed_for_daily_expenses_last_year  151 non-null    bool
 40  borrowed_for_home_or_biz_last_year     151 non-null    bool
 41  phone_technology                       151 non-null    int64
 42  can_call                               151 non-null    bool
 43  can_text                               151 non-null    bool
 44  can_use_internet                       151 non-null    bool
 45  can_make_transaction                   151 non-null    bool
 46  phone_ownership                        151 non-null    int64
 47  advanced_phone_use                     151 non-null    bool
 48  reg_bank_acct                          151 non-null    bool
 49  reg_mm_acct                            151 non-null    bool
 50  reg_formal_nbfi_account                151 non-null    bool
 51  financially_included                   151 non-null    bool
 52  active_bank_user                       151 non-null    bool
 53  active_mm_user                         151 non-null    bool
 54  active_formal_nbfi_user                151 non-null    bool
 55  active_informal_nbfi_user              151 non-null    bool
 56  nonreg_active_mm_user                  151 non-null    bool
 57  num_formal_institutions_last_year      151 non-null    int64
 58  num_informal_institutions_last_year    151 non-null    int64
 59  num_financial_activities_last_year     151 non-null    int64
dtypes: bool(37), float64(8), int64(10), object(5)
memory usage: 33.8+ KB
```

As there are still some columns with null values, so we will replace them with zeros

In [79]:
```
non_null['bank_interest_rate'] = non_null['bank_interest_rate'].replace(np.nan, 0)
non_null['mfi_interest_rate'] = non_null['mfi_interest_rate'].replace(np.nan, 0)
non_null['other_fsp_interest_rate'] = non_null['other_fsp_interest_rate'].replace(np.nan, 0)
non_null.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 151 entries, 175 to 12534
Data columns (total 60 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
```

```
0   row_id                                  151 non-null    int64
1   poverty_probability                     151 non-null    float64
2   country                                 151 non-null    object
3   is_urban                                151 non-null    bool
4   age                                     151 non-null    int64
5   female                                  151 non-null    bool
6   married                                 151 non-null    bool
7   religion                                151 non-null    object
8   relationship_to_hh_head                 151 non-null    object
9   education_level                         151 non-null    float64
10  literacy                                151 non-null    bool
11  can_add                                 151 non-null    bool
12  can_divide                              151 non-null    bool
13  can_calc_percents                       151 non-null    bool
14  can_calc_compounding                    151 non-null    bool
15  employed_last_year                      151 non-null    bool
16  employment_category_last_year           151 non-null    object
17  employment_type_last_year               151 non-null    object
18  share_hh_income_provided                151 non-null    float64
19  income_ag_livestock_last_year           151 non-null    bool
20  income_friends_family_last_year         151 non-null    bool
21  income_government_last_year             151 non-null    bool
22  income_own_business_last_year           151 non-null    bool
23  income_private_sector_last_year         151 non-null    bool
24  income_public_sector_last_year          151 non-null    bool
25  num_times_borrowed_last_year            151 non-null    int64
26  borrowing_recency                       151 non-null    int64
27  formal_savings                          151 non-null    bool
28  informal_savings                        151 non-null    bool
29  cash_property_savings                   151 non-null    bool
30  has_insurance                           151 non-null    bool
31  has_investment                          151 non-null    bool
32  bank_interest_rate                      151 non-null    float64
33  mm_interest_rate                        151 non-null    float64
34  mfi_interest_rate                       151 non-null    float64
35  other_fsp_interest_rate                 151 non-null    float64
36  num_shocks_last_year                    151 non-null    int64
37  avg_shock_strength_last_year            151 non-null    float64
38  borrowed_for_emergency_last_year        151 non-null    bool
39  borrowed_for_daily_expenses_last_year   151 non-null    bool
40  borrowed_for_home_or_biz_last_year      151 non-null    bool
41  phone_technology                        151 non-null    int64
42  can_call                                151 non-null    bool
43  can_text                                151 non-null    bool
44  can_use_internet                        151 non-null    bool
```

```
 45  can_make_transaction                   151 non-null    bool
 46  phone_ownership                        151 non-null    int64
 47  advanced_phone_use                     151 non-null    bool
 48  reg_bank_acct                          151 non-null    bool
 49  reg_mm_acct                            151 non-null    bool
 50  reg_formal_nbfi_account                151 non-null    bool
 51  financially_included                   151 non-null    bool
 52  active_bank_user                       151 non-null    bool
 53  active_mm_user                         151 non-null    bool
 54  active_formal_nbfi_user                151 non-null    bool
 55  active_informal_nbfi_user              151 non-null    bool
 56  nonreg_active_mm_user                  151 non-null    bool
 57  num_formal_institutions_last_year      151 non-null    int64
 58  num_informal_institutions_last_year    151 non-null    int64
 59  num_financial_activities_last_year     151 non-null    int64
dtypes: bool(37), float64(8), int64(10), object(5)
memory usage: 33.8+ KB
```

```
<ipython-input-79-0eaa16bcc259>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  non_null['bank_interest_rate'] = non_null['bank_interest_rate'].replace(np.nan, 0)
<ipython-input-79-0eaa16bcc259>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  non_null['mfi_interest_rate'] = non_null['mfi_interest_rate'].replace(np.nan, 0)
<ipython-input-79-0eaa16bcc259>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  non_null['other_fsp_interest_rate'] = non_null['other_fsp_interest_rate'].replace(np.nan, 0)
```

4. Run various pandas functions/methods to know more about the data such as describe, info

Write comments about what came to your mind when you saw the results of describe and info

In [80]:
```
non_null.describe(include=['object','integer','bool','float'])
```

Out[80]:

	row_id	poverty_probability	country	is_urban	age	female	married	religion	relationship_to_hh_head	education_level	...	re
count	151.000000	151.000000	151	151	151.000000	151	151	151	151	151.000000	...	
unique	NaN	NaN	6	2	NaN	2	2	3	6	NaN	...	
top	NaN	NaN	G	True	NaN	False	False	X	Head	NaN	...	
freq	NaN	NaN	103	76	NaN	86	79	126	93	NaN	...	
mean	6016.887417	0.481967	NaN	NaN	32.556291	NaN	NaN	NaN	NaN	1.966887	...	
std	3446.937206	0.236096	NaN	NaN	9.549615	NaN	NaN	NaN	NaN	0.795129	...	
min	175.000000	0.027000	NaN	NaN	18.000000	NaN	NaN	NaN	NaN	0.000000	...	
25%	2931.500000	0.309000	NaN	NaN	25.000000	NaN	NaN	NaN	NaN	1.000000	...	
50%	5590.000000	0.456000	NaN	NaN	30.000000	NaN	NaN	NaN	NaN	2.000000	...	
75%	8657.500000	0.656000	NaN	NaN	36.000000	NaN	NaN	NaN	NaN	3.000000	...	
max	12534.000000	0.967000	NaN	NaN	68.000000	NaN	NaN	NaN	NaN	3.000000	...	

11 rows × 60 columns

This function gives the mean, standar deviation, frequency, range of the quartiles etc.

In [82]:
```
non_null.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 151 entries, 175 to 12534
Data columns (total 60 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   row_id                                151 non-null    int64
 1   poverty_probability                   151 non-null    float64
 2   country                               151 non-null    object
 3   is_urban                              151 non-null    bool
 4   age                                   151 non-null    int64
 5   female                                151 non-null    bool
```

```
6   married                                 151 non-null    bool
7   religion                                151 non-null    object
8   relationship_to_hh_head                 151 non-null    object
9   education_level                         151 non-null    float64
10  literacy                                151 non-null    bool
11  can_add                                 151 non-null    bool
12  can_divide                              151 non-null    bool
13  can_calc_percents                       151 non-null    bool
14  can_calc_compounding                    151 non-null    bool
15  employed_last_year                      151 non-null    bool
16  employment_category_last_year           151 non-null    object
17  employment_type_last_year               151 non-null    object
18  share_hh_income_provided                151 non-null    float64
19  income_ag_livestock_last_year           151 non-null    bool
20  income_friends_family_last_year         151 non-null    bool
21  income_government_last_year             151 non-null    bool
22  income_own_business_last_year           151 non-null    bool
23  income_private_sector_last_year         151 non-null    bool
24  income_public_sector_last_year          151 non-null    bool
25  num_times_borrowed_last_year            151 non-null    int64
26  borrowing_recency                       151 non-null    int64
27  formal_savings                          151 non-null    bool
28  informal_savings                        151 non-null    bool
29  cash_property_savings                   151 non-null    bool
30  has_insurance                           151 non-null    bool
31  has_investment                          151 non-null    bool
32  bank_interest_rate                      151 non-null    float64
33  mm_interest_rate                        151 non-null    float64
34  mfi_interest_rate                       151 non-null    float64
35  other_fsp_interest_rate                 151 non-null    float64
36  num_shocks_last_year                    151 non-null    int64
37  avg_shock_strength_last_year            151 non-null    float64
38  borrowed_for_emergency_last_year        151 non-null    bool
39  borrowed_for_daily_expenses_last_year   151 non-null    bool
40  borrowed_for_home_or_biz_last_year      151 non-null    bool
41  phone_technology                        151 non-null    int64
42  can_call                                151 non-null    bool
43  can_text                                151 non-null    bool
44  can_use_internet                        151 non-null    bool
45  can_make_transaction                    151 non-null    bool
46  phone_ownership                         151 non-null    int64
47  advanced_phone_use                      151 non-null    bool
48  reg_bank_acct                           151 non-null    bool
49  reg_mm_acct                             151 non-null    bool
50  reg_formal_nbfi_account                 151 non-null    bool
```

```
 51  financially_included                  151 non-null    bool
 52  active_bank_user                      151 non-null    bool
 53  active_mm_user                        151 non-null    bool
 54  active_formal_nbfi_user               151 non-null    bool
 55  active_informal_nbfi_user             151 non-null    bool
 56  nonreg_active_mm_user                 151 non-null    bool
 57  num_formal_institutions_last_year     151 non-null    int64
 58  num_informal_institutions_last_year   151 non-null    int64
 59  num_financial_activities_last_year    151 non-null    int64
dtypes: bool(37), float64(8), int64(10), object(5)
memory usage: 33.8+ KB
```

This command prints information about a DataFrame including the dtypes and columns, non-null values and memory usage.

5. Use data aggregation techniques such as the groupby and qcut functions to learn more insights about the data and include them in your notebook.

In [84]:
```
non_null.groupby(['country']).sum()
```

Out[84]:

country	row_id	poverty_probability	is_urban	age	female	married	education_level	literacy	can_add	can_divide	...	reg_formal_nbfi_account
A	19183	2.735	3	118	1	3	10.0	4	4	4	...	1
C	21971	1.420	3	70	0	1	6.0	3	3	2	...	1
D	190871	17.311	10	882	7	13	46.0	28	22	24	...	6
F	5672	1.505	0	50	0	1	5.0	2	2	2	...	0
G	628676	46.058	55	3443	51	48	211.0	88	99	96	...	39
J	42177	3.748	5	353	6	6	19.0	10	10	7	...	1

6 rows × 55 columns

Country C & F have no female. Country F has the lowest poverty probability.

In [86]:
```
non_null.groupby(['country']).mean()
```

Out[86]:

country	row_id	poverty_probability	is_urban	age	female	married	education_level	literacy	can_add	can_divide	...	reg_formal_
A	4795.750000	0.683750	0.750000	29.500000	0.250000	0.750000	2.500000	1.000000	1.000000	1.000000	...	
C	7323.666667	0.473333	1.000000	23.333333	0.000000	0.333333	2.000000	1.000000	1.000000	0.666667	...	
D	6816.821429	0.618250	0.357143	31.500000	0.250000	0.464286	1.642857	1.000000	0.785714	0.857143	...	
F	2836.000000	0.752500	0.000000	25.000000	0.000000	0.500000	2.500000	1.000000	1.000000	1.000000	...	
G	6103.650485	0.447165	0.533981	33.427184	0.495146	0.466019	2.048544	0.854369	0.961165	0.932039	...	
J	3834.272727	0.340727	0.454545	32.090909	0.545455	0.545455	1.727273	0.909091	0.909091	0.636364	...	

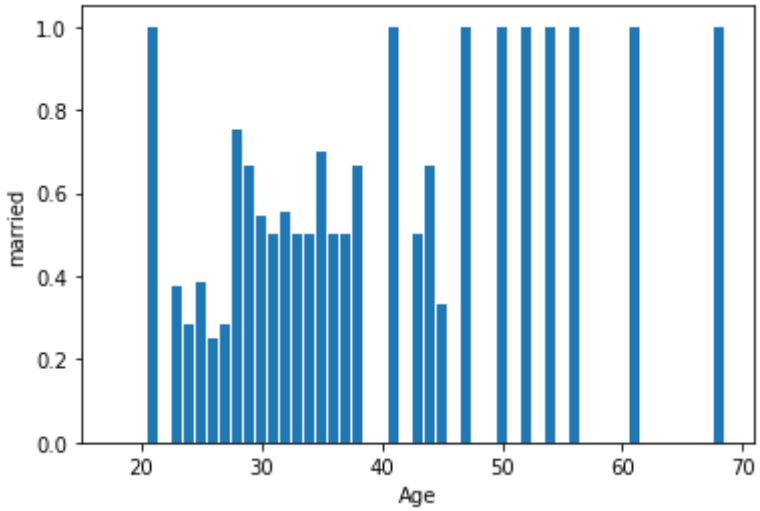
6 rows × 55 columns

Poverty probability of F is the highest in all countries.

In [88]:
```
import matplotlib.pyplot as plt

plot_married=non_null['married'].groupby(non_null['age'] ).mean()

plt.bar(plot_married.index,plot_married)
plt.xlabel('Age')
plt.ylabel("married")
plt.show()
```



6. Finally, talk about the issues with data and challenges that you see ahead and how would you plan to address them?

The issue that I had earlier was to extract the meaningful data from given data sets. The main challenge is to fill the columns with the values that are closest as per prediction because most of the datasets have some null values in them.