HW4: Occupation Dataset

Introduction:

Special thanks to: https://github.com/guipsamora for sharing his datasets, materials, and questions.

- https://github.com/justmarkham for sharing the dataset and materials.

In [1]:
```
### Import the necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
### I believe the following data set is from the US in the 1950s.
### Gender proportions and ages are thus reflective of that era.
### Import the dataset from this address. https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user
### Assign it to a variable called users and use the 'user_id' as index
users = pd.read_csv('https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user',
                    sep='|', index_col='user_id')
```

In [3]:
```
# Problem 1. See the first 10 entries. (done for you)
users.head(10)
```

Out[3]:

user_id	age	gender	occupation	zip_code
1	24	M	technician	85711
2	53	F	other	94043
3	23	M	writer	32067
4	24	M	technician	43537
5	33	F	other	15213
6	42	M	executive	98101
7	57	M	administrator	91344
8	36	M	administrator	05201
9	29	M	student	01002
10	53	M	lawyer	90703

In [4]:
```
# Problem 2. How many observations and columns are in the data?

# number of observations
print(len(users))
```

```
943
```

In [5]:
```
# number of columns
print(len(users.columns))
```

```
4
```

In [6]:
```
# Problem 3. How many different occupations there are in this dataset?
occupation_groups = users.groupby(['occupation'])
print(len(occupation_groups))
```

```
21
```

In [7]:
```
# Problem 4. What is the most frequent occupation?
occupation_sizes = occupation_groups.size()
print(occupation_sizes.idxmax())
```

```
student
```

In [8]:
```
# Problem 5. Discover what is the mean age per occupation.
mean_by_occupation = occupation_groups.mean()
print(mean_by_occupation)
```

```
                     age
occupation
administrator  38.746835
artist         31.392857
doctor         43.571429
educator       42.010526
engineer       36.388060
entertainment  29.222222
executive      38.718750
healthcare     41.562500
homemaker      32.571429
lawyer         36.750000
librarian      40.000000
marketing      37.615385
none           26.555556
other          34.523810
programmer     33.121212
retired        63.071429
salesman       35.666667
scientist      35.548387
student        22.081633
technician     33.148148
writer         36.311111
```

In [9]:
```
# Sort the results and find the 3 occupations with the lowest mean age and the 3 with the highest
sorted_results = mean_by_occupation.sort_values(by = 'age', ascending = True)

# 3 lowest age occupations
sorted_results[:3]
```

Out[9]:

	age
occupation	
student	22.081633
none	26.555556
entertainment	29.222222

In [10]:
```
# 3 highest age occupations
```

```
sorted_results[-3:]
```

Out[10]:

occupation	age
educator	42.010526
doctor	43.571429
retired	63.071429

In [11]:

```
# Problem 6. Find the proportion of males by occupation and sort it from the most to the least

# by_occupation = users.groupby(users['occupation'])
# for df in by_occupation:
#     sum = df.loc[df.gender == 'M'].index.size()
#     total = df.index.size()

occupation_males = users[users['gender'] == "M"].groupby(users['occupation']).size()
occupation_males_prop = occupation_males / occupation_sizes
occupation_males_prop.sort_values(ascending = False)
```

Out[11]:
```
occupation
doctor           1.000000
engineer         0.970149
technician       0.962963
retired          0.928571
programmer       0.909091
executive        0.906250
scientist        0.903226
entertainment    0.888889
lawyer           0.833333
salesman         0.750000
educator         0.726316
student          0.693878
other            0.657143
marketing        0.615385
writer           0.577778
none             0.555556
administrator    0.544304
artist           0.535714
librarian        0.431373
healthcare       0.312500
homemaker        0.142857
dtype: float64
```

In [12]:
```
# Problem 7. For each occupation, calculate the minimum and maximum ages
# See groupby and agg() to perform multiple aggregate functions at once
occupation_groups.age.agg(['min', 'max'])
```

Out[12]:

occupation	min	max
administrator	21	70
artist	19	48
doctor	28	64
educator	23	63
engineer	22	70
entertainment	15	50
executive	22	69
healthcare	22	62
homemaker	20	50
lawyer	21	53
librarian	23	69
marketing	24	55
none	11	55
other	13	64
programmer	20	63
retired	51	73
salesman	18	66
scientist	23	55
student	7	42
technician	21	55
writer	18	60

In [13]:
```
# Problem 8. For each combination of occupation and gender, calculate the mean age.
# Arrange the results in a table so each row is an occupation, and you have a
# column of the average male age and another column with the average female age.
```

```
# Sort the resulting table by Female mean age from least to greatest
users.groupby(['occupation', 'gender']).mean().unstack().sort_values(by = ('age', 'F'))
```

Out[13]:

	age	
gender	F	M
occupation		
student	20.750000	22.669118
salesman	27.000000	38.555556
scientist	28.333333	36.321429
engineer	29.500000	36.600000
artist	30.307692	32.333333
entertainment	31.000000	29.000000
programmer	32.166667	33.216667
homemaker	34.166667	23.000000
other	35.472222	34.028986
none	36.500000	18.600000
marketing	37.200000	37.875000
writer	37.631579	35.346154
technician	38.000000	32.961538
educator	39.115385	43.101449
lawyer	39.500000	36.200000
healthcare	39.818182	45.400000
librarian	40.000000	40.000000
administrator	40.638889	37.162791
executive	44.000000	38.172414
retired	70.000000	62.538462
doctor	NaN	43.571429

In [14]:
```
# Problem 9. For each occupation find the count of women and men
# Arrange the results in a table so each row is an occupation, similar to above
occupation_gender_size = occupation_groups['gender'].value_counts().unstack().fillna(0)
occupation_gender_size
```

Out[14]:

gender	F	M
occupation		
administrator	36.0	43.0
artist	13.0	15.0
doctor	0.0	7.0
educator	26.0	69.0
engineer	2.0	65.0
entertainment	2.0	16.0
executive	3.0	29.0
healthcare	11.0	5.0
homemaker	6.0	1.0
lawyer	2.0	10.0
librarian	29.0	22.0
marketing	10.0	16.0
none	4.0	5.0
other	36.0	69.0
programmer	6.0	60.0
retired	1.0	13.0
salesman	3.0	9.0
scientist	3.0	28.0
student	60.0	136.0
technician	1.0	26.0
writer	19.0	26.0

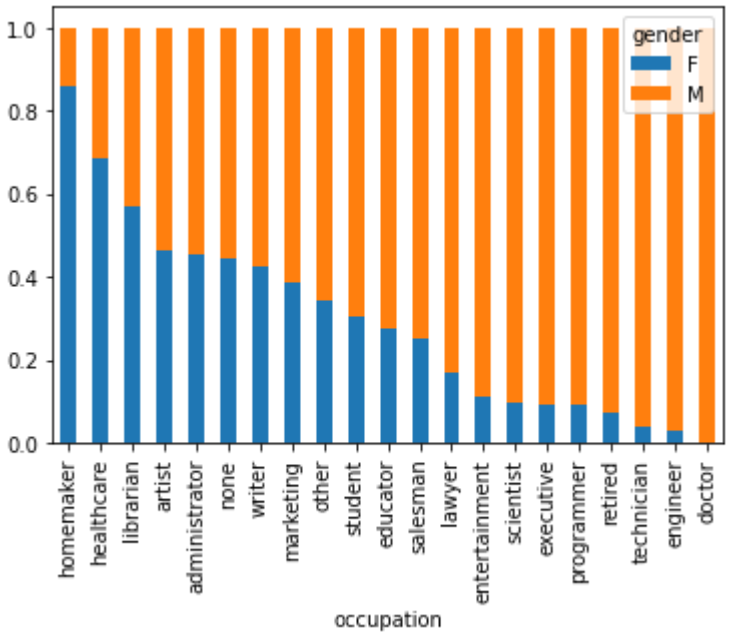
In [15]:
```
# Problem 10. Turn the counts above into proportions. e.g administrator 0.455696 0.544304
# Arrange results in increasing order of proportion men
occupation_gender_prop = occupation_groups['gender'].value_counts(normalize = True)\
.unstack().fillna(0).sort_values(by = 'M')
occupation_gender_prop
```

Out[15]:

gender	F	M
occupation		
homemaker	0.857143	0.142857
healthcare	0.687500	0.312500
librarian	0.568627	0.431373
artist	0.464286	0.535714
administrator	0.455696	0.544304
none	0.444444	0.555556
writer	0.422222	0.577778
marketing	0.384615	0.615385
other	0.342857	0.657143
student	0.306122	0.693878
educator	0.273684	0.726316
salesman	0.250000	0.750000
lawyer	0.166667	0.833333
entertainment	0.111111	0.888889
scientist	0.096774	0.903226
executive	0.093750	0.906250
programmer	0.090909	0.909091
retired	0.071429	0.928571
technician	0.037037	0.962963
engineer	0.029851	0.970149
doctor	0.000000	1.000000

In [16]:
```
# Create a stacked barchart showing the results above
occupation_gender_prop.plot.bar(stacked = True)
```

Out[16]: <AxesSubplot:xlabel='occupation'>



In [17]:
```
# Extract the first digit of each zip code
# and create a new column called 'region' that maps the
# first digit of the zip to new values using this dictionary:
d = {'0': 'New England',
'1': 'Mid-Atlantic',
'2': 'Central East Coast',
'3': 'The South',
'4': 'Midwest',
'5': 'Northern Great Plains',
'6': 'Central Great Plains',
'7': 'Southern Central',
'8': 'Mountain Desert',
'9': 'West Coast'}

# print the first 5 rows of the result
# postal codes that begin with a letter are actually Canadian but are missing the last digit. These rows can be ignored
```

In [18]:
```
def zip_to_region(zip):
    key = str(zip)[0]
    if key in d:
        return d[key]
    else:
```

```
        return "Canada"
users['region'] = users['zip_code'].apply(zip_to_region)
print(users.head(5))
```

```
         age gender  occupation zip_code           region
user_id
1         24      M  technician    85711  Mountain Desert
2         53      F       other    94043       West Coast
3         23      M      writer    32067        The South
4         24      M  technician    43537          Midwest
5         33      F       other    15213     Mid-Atlantic
```

In [19]:
```
# for the occuptation 'retired', find the mean age of each region
users.loc[users.occupation == 'retired'].groupby(['region']).mean()
```

Out[19]:

region	age
Central East Coast	60.0
Central Great Plains	59.5
Mid-Atlantic	60.0
Midwest	69.0
New England	65.0
Northern Great Plains	61.0
The South	73.0
West Coast	60.5