Occupation

Introduction:

Special thanks to: https://github.com/justmarkham for sharing the dataset and materials.

Step 1. Import the necessary libraries

In [1]:

```
import pandas as pd
```

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called users.

In [2]:

```
users = pd.read_table('https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user',
                      sep='|', index_col='user_id')
users.head()
```

Out[2]:

	age	gender	occupation	zip_code
user_id				
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

Step 4. Discover what is the mean age per occupation

In [3]:

```
users.groupby("occupation").age.mean()
```

Out[3]:

```
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
Name: age, dtype: float64
```

Step 5. Discover the Male ratio per occupation and sort it from the most to the least

In [4]:

```
users["gender"] = users["gender"].map({"M": 1, "F":0})
```

In [5]:

```
a = users.groupby("occupation").gender.sum() / users.occupation.value_counts() * 100
a.sort_values(ascending = False)
```

Out[5]:

```
doctor           100.000000
engineer          97.014925
technician        96.296296
retired           92.857143
programmer        90.909091
executive         90.625000
scientist         90.322581
entertainment     88.888889
lawyer            83.333333
salesman          75.000000
educator          72.631579
student           69.387755
other             65.714286
marketing         61.538462
writer            57.777778
none              55.555556
administrator     54.430380
artist            53.571429
librarian         43.137255
healthcare        31.250000
homemaker         14.285714
dtype: float64
```

Step 6. For each occupation, calculate the minimum and maximum ages

In [6]:

```
users.groupby("occupation").age.agg(["min","max"])
```

Out[6]:

	min	max
occupation		
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

Step 7. For each combination of occupation and gender, calculate the mean age

In [7]:

```
users.groupby(["occupation","gender"]).age.mean()
```

Out[7]:

```
occupation     gender
administrator  0         40.638889
               1         37.162791
artist         0         30.307692
               1         32.333333
doctor         1         43.571429
educator       0         39.115385
               1         43.101449
engineer       0         29.500000
               1         36.600000
entertainment  0         31.000000
               1         29.000000
executive      0         44.000000
               1         38.172414
healthcare     0         39.818182
               1         45.400000
homemaker      0         34.166667
               1         23.000000
lawyer         0         39.500000
               1         36.200000
librarian      0         40.000000
               1         40.000000
marketing      0         37.200000
               1         37.875000
none           0         36.500000
               1         18.600000
other          0         35.472222
               1         34.028986
programmer     0         32.166667
               1         33.216667
retired        0         70.000000
               1         62.538462
salesman       0         27.000000
               1         38.555556
scientist      0         28.333333
               1         36.321429
student        0         20.750000
               1         22.669118
technician     0         38.000000
               1         32.961538
writer         0         37.631579
               1         35.346154
Name: age, dtype: float64
```

Step 8. For each occupation present the percentage of women and men

In [8]:

```
users.groupby("occupation").gender.agg(["count"/users["gender"].count().sum()])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-8c31186bcd15> in <module>
----> 1 users.groupby("occupation").gender.agg(["count"/users["gender"].count().sum()])

TypeError: ufunc 'true_divide' not supported for the input types, and the inputs could not be safely coerced to any supported types according to the casting r
ule ''safe''
```

In [ ]:

```
gender_ocup = users.groupby(['occupation', 'gender']).agg({'gender': 'count'})
gender_ocup
```

In [ ]:

```
occup_count = users.groupby(['occupation']).agg('count')
occup_count
```

In [ ]:

```
occup_gender = gender_ocup.div(occup_count, level = "occupation") * 100
occup_gender
```

In [ ]:

```
occup_gender.loc[: , 'gender']
```