Mud card

- **can you go over the merge and appending again? It was a little fast at the end of the lecture**
- **Merge was a bit tricky to understand!**
- **Concatenating Dataframes**
  - play around with the code, read the manual of merge and concatenate, and rewatch the lecture recording on canvas
- **How to append dataframes so that the indices are not repeated**
  - set ignore_index to True
- **Finding errors like in exercise 1 and then knowing which options to use to fix them was muddy.**
  - yep, that will take some practice
  - it's mostly just common sense though
  - when you read in a dataset:
    - open your dataset with excel and literally just take a look at it
    - read it in with pandas
    - print out the head or tail
    - compare what you see in python vs. what's in the file
  - when you filter/merge/append or do any other transformation:
    - come up with tests
    - create a small dataset for which you know what the correct solution of your operation should be
    - think of edge cases too
- **When would we use loc instead of iloc? is the only difference whether it includes the stop or not?**
  - loc and iloc are only the same if the default row indexing is used
  - loc filters based on the value of the index column which can be changed
  - iloc uses the row's integer position always

Exploratory data analysis in python, part 2

Learning objectives

By the end of this lecture, you will be able to

- visualize one column (continuous or categorical data)
- visualize column pairs (all variations of continuous and categorical columns)
- visualize multiple columns simultaneously

Dataset of the day

Adult dataset, see here

Packages of the day

matplotlib and pandas

By the end of this lecture, you will be able to

- **visualize one column (categorical or continuous data)**
- visualize column pairs (all variations of continuous and categorical columns)
- visualize multiple columns simultaneously

Let's load the data first!

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib
from matplotlib import pylab as plt
df = pd.read_csv('data/adult_data.csv')
print(df.dtypes)
```

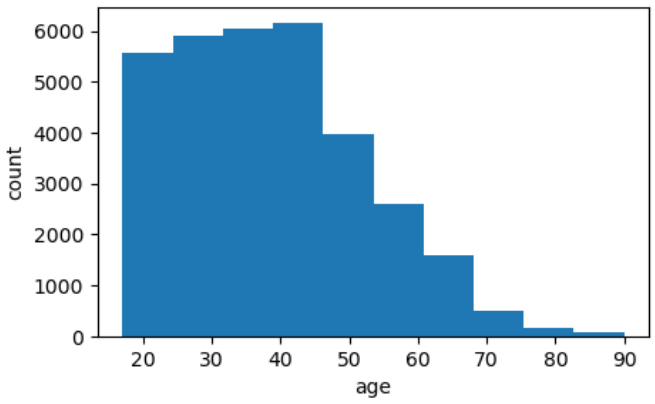
```
age                int64
workclass         object
fnlwgt             int64
education         object
education-num      int64
marital-status    object
occupation        object
relationship      object
race              object
sex               object
capital-gain       int64
capital-loss       int64
hours-per-week     int64
native-country    object
gross-income      object
dtype: object
```

Column is continuous

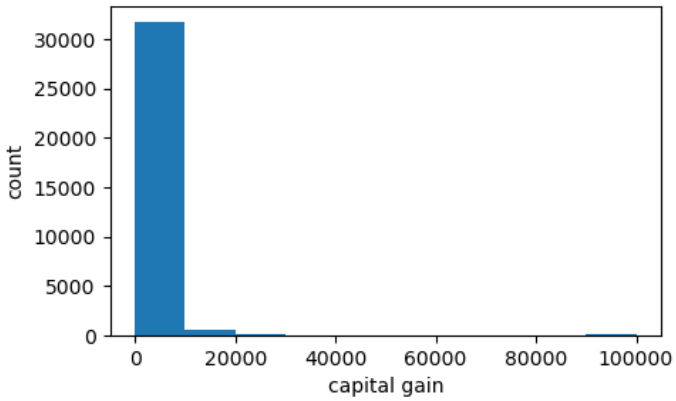
In [2]:
```
print(df['age'].describe())
```

```
count    32561.000000
mean        38.581647
std         13.640433
min         17.000000
25%         28.000000
50%         37.000000
75%         48.000000
max         90.000000
Name: age, dtype: float64
```

In [3]:
```
plt.figure(figsize=(5,3))

df['age'].plot.hist()   # bins = int(np.sqrt(df.shape[0]))
                        # bins = df['age'].nunique()
plt.xlabel('age')
plt.ylabel('count')
plt.show()
```



In [4]:
```
plt.figure(figsize=(5,3))

df['capital-gain'].plot.hist() # log=True, bins = np.logspace(np.log10(1),np.log10(np.max(df['capital-gain'])),50
#plt.semilogy()
#plt.semilogx()
plt.xlabel('capital gain')
plt.ylabel('count')
plt.show()
```

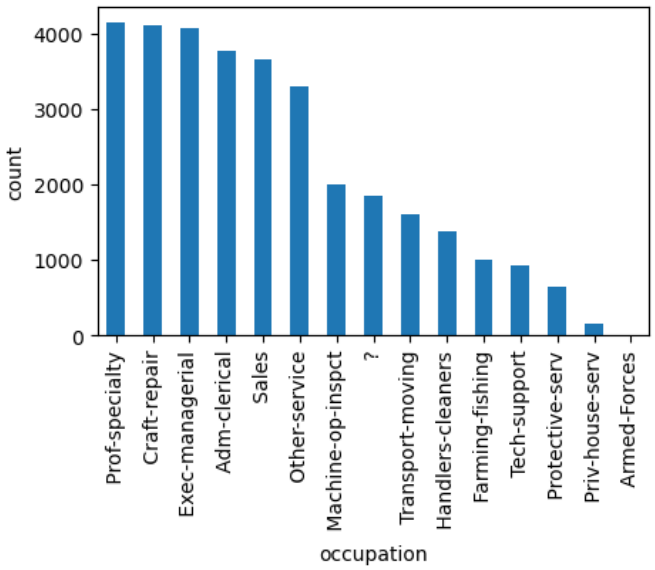


Column is categorical

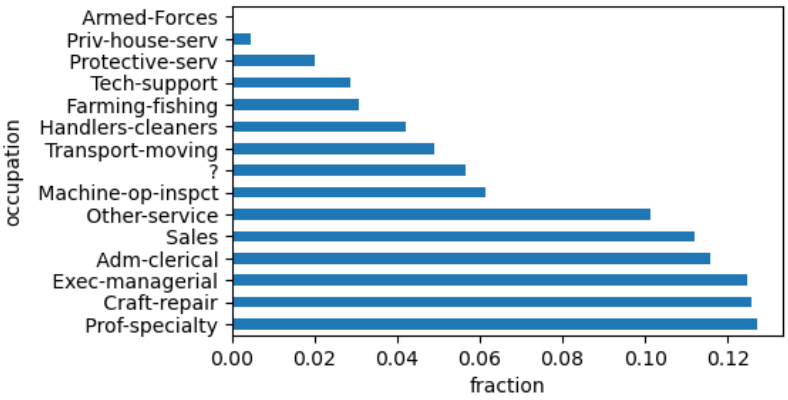
In [5]:
```
print(df['occupation'].value_counts())
```

```
occupation
 Prof-specialty       4140
 Craft-repair         4099
 Exec-managerial      4066
 Adm-clerical         3770
 Sales                3650
 Other-service        3295
 Machine-op-inspct    2002
 ?                    1843
 Transport-moving     1597
 Handlers-cleaners    1370
 Farming-fishing       994
 Tech-support          928
 Protective-serv       649
 Priv-house-serv       149
 Armed-Forces            9
Name: count, dtype: int64
```

In [6]:
```
plt.figure(figsize=(5,3))
pd.value_counts(df['occupation']).plot.bar()
plt.ylabel('count')
plt.xlabel('occupation')
plt.show()
```

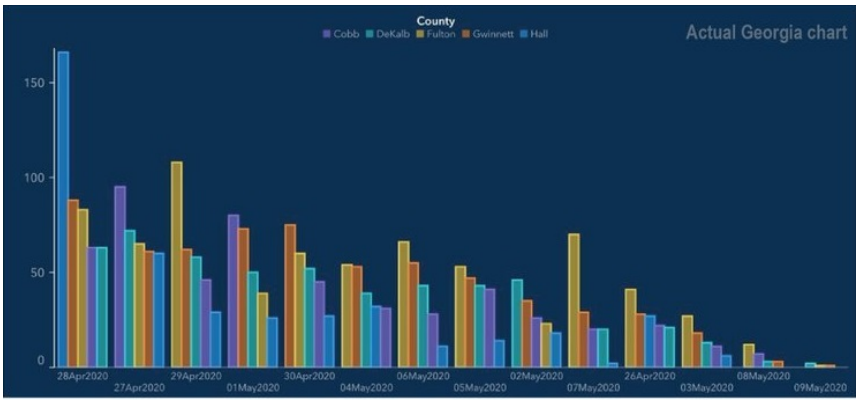


In [7]:
```
plt.figure(figsize=(5,3))
pd.value_counts(df['occupation'],normalize=True).plot.barh()
plt.xlabel('fraction')
plt.show()
```

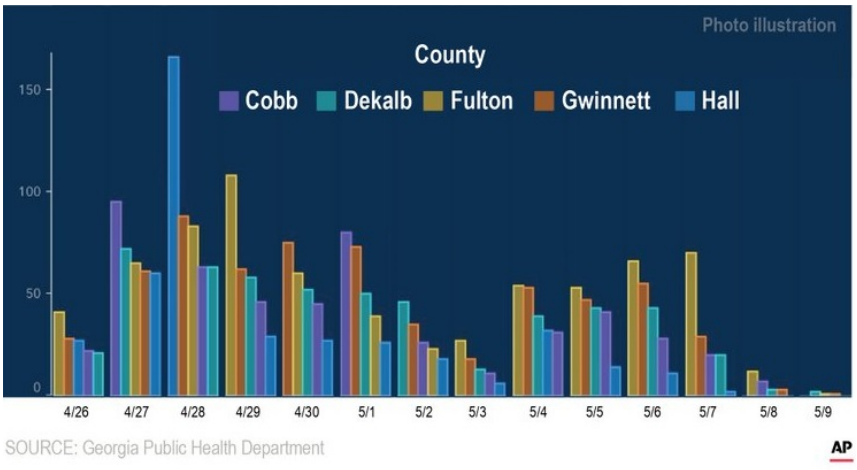


Quiz 1

- What's wrong with this figure?



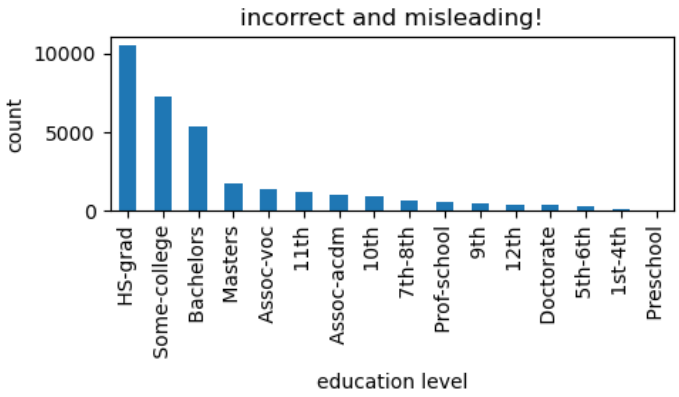
Ordinal features



- other examples of ordinal features:
  - measure of quality (e.g., bad, average, good, excellent)
  - socioeconomic status (e.g., low income, middle income, high income)
  - education level (e.g., 8th grade, high school, BSc, MSc, PhD)
  - satisfaction rating (e.g., dislike, neutral, like)
  - time (e.g., days of the week, months, years)

In [8]:
```
plt.figure(figsize=(5,3))
pd.value_counts(df['education']).plot.bar()
plt.ylabel('count')
plt.xlabel('education level')
```

```
plt.title('incorrect and misleading!')
plt.tight_layout()
plt.show()
```



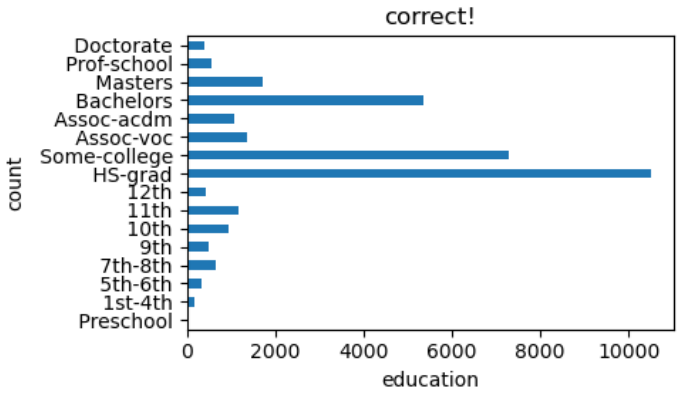
In [9]:
```
pd.value_counts(df['education'])
```

Out[9]:
```
education
 HS-grad         10501
 Some-college     7291
 Bachelors        5355
 Masters          1723
 Assoc-voc        1382
 11th             1175
 Assoc-acdm       1067
 10th              933
 7th-8th           646
 Prof-school       576
 9th               514
 12th              433
 Doctorate         413
 5th-6th           333
 1st-4th           168
 Preschool          51
Name: count, dtype: int64
```

In [10]:
```
correct_order = [' Preschool', ' 1st-4th', ' 5th-6th', ' 7th-8th', ' 9th', ' 10th', ' 11th', \
         ' 12th', ' HS-grad', ' Some-college', ' Assoc-voc', ' Assoc-acdm', ' Bachelors',\
         ' Masters', ' Prof-school', ' Doctorate']

pd.value_counts(df['education']).reindex(correct_order)
```

Out[10]:
```
education
 Preschool          51
 1st-4th           168
 5th-6th           333
 7th-8th           646
 9th               514
 10th              933
 11th             1175
 12th              433
 HS-grad         10501
 Some-college     7291
 Assoc-voc        1382
 Assoc-acdm       1067
 Bachelors        5355
 Masters          1723
 Prof-school       576
 Doctorate         413
Name: count, dtype: int64
```

In [11]:
```
plt.figure(figsize=(5,3))

pd.value_counts(df['education']).reindex(correct_order).plot.barh()
plt.ylabel('count')
plt.xlabel('education')
plt.title('correct!')
plt.tight_layout()
plt.show()
```



By the end of this lecture, you will be able to

- visualize one column (categorical or continuous data)
- **visualize column pairs (all variations of continuous and categorical columns)**
- visualize multiple columns simultaneously

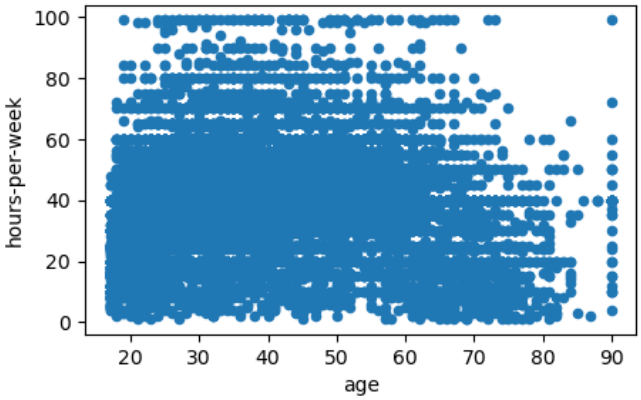
Overview

Visualization types	column continuous	column categorical
column continuous	scatter plot, heatmap	category-specific histograms, box plot, violin plot
column categorical	category-specific histograms, box plot, violin plot	stacked bar plot

Continuous vs. continuous columns

- scatter plot

In [12]:
```
df.plot.scatter('age','hours-per-week',figsize=(5,3)) # alpha=0.1,s=10
plt.show()
```



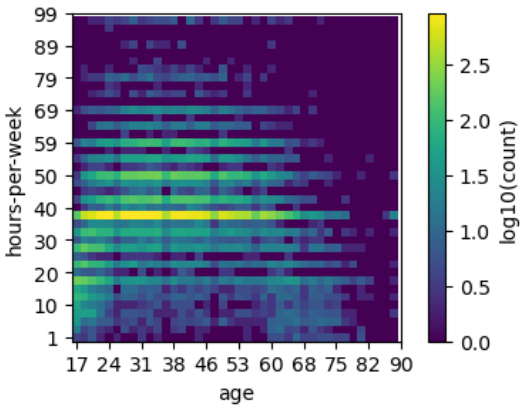
Continuous vs. continuous columns

- heatmap

In [13]:
```
nbins = 40

heatmap, xedges, yedges = np.histogram2d(df['age'], df['hours-per-week'], bins=nbins)
extent = [xedges[0], xedges[-1], yedges[0], yedges[-1]]
```

In [14]:
```
heatmap[heatmap == 0] = 0.1 # we will use log and log(0) is undefined
plt.figure(figsize=(5,3))

plt.imshow(np.log10(heatmap).T, origin='lower',vmin=0) # use log count
plt.xlabel('age')
plt.ylabel('hours-per-week')
plt.xticks(np.arange(nbins+1)[::4],xedges[::4].astype(int))
```

```
plt.yticks(np.arange(nbins+1)[::4],yedges[::4].astype(int))
plt.colorbar(label='log10(count)')
plt.show()
```



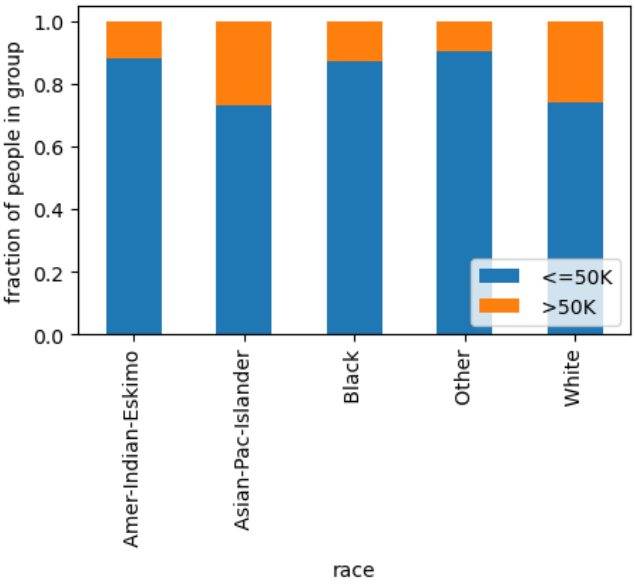
Categorical vs. categorical columns

- stacked bar plot

In [15]:
```
count_matrix = df.groupby(['race', 'gross-income']).size().unstack()
print(count_matrix)

count_matrix_norm = count_matrix.div(count_matrix.sum(axis=1),axis=0)
#print(count_matrix_norm)
```

```
gross-income          <=50K   >50K
race
 Amer-Indian-Eskimo     275     36
 Asian-Pac-Islander     763    276
 Black                 2737    387
 Other                  246     25
 White                20699   7117
```

In [16]:
```
count_matrix_norm.plot(kind='bar', stacked=True,figsize=(5,3))
plt.ylabel('fraction of people in group')
plt.legend(loc=4)
plt.show()
```

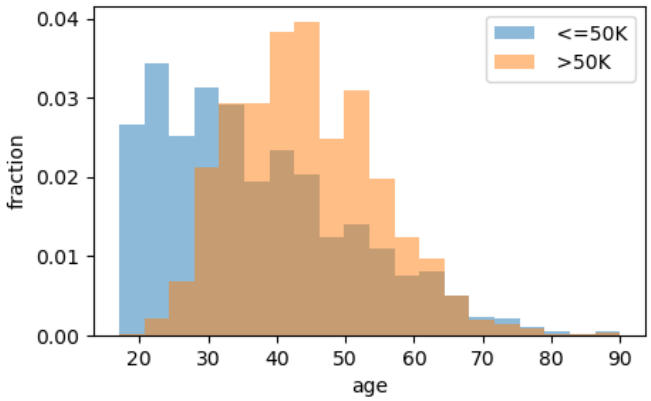


Continuous vs. categorical columns

- category-specific histograms

In [17]:
```
import matplotlib
from matplotlib import pylab as plt
```

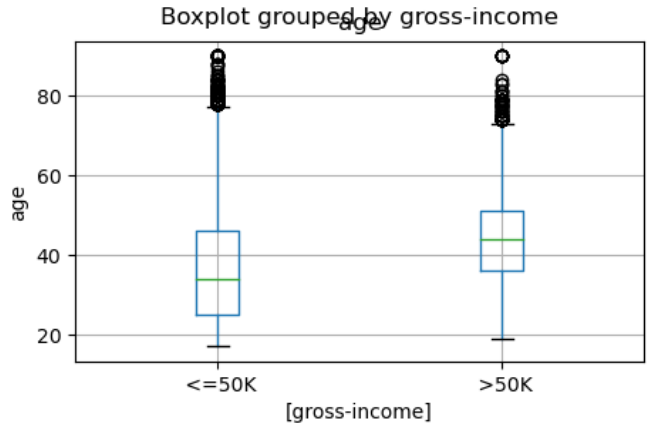
```
categories = df['gross-income'].unique()
bin_range = (df['age'].min(),df['age'].max())

plt.figure(figsize=(5,3))

for c in categories:
    plt.hist(df[df['gross-income']==c]['age'],alpha=0.5,label=c,range=bin_range,bins=20,density=True)
plt.legend()
plt.ylabel('fraction')
plt.xlabel('age')
plt.show()
```



Continuous vs. categorical columns

- box plot

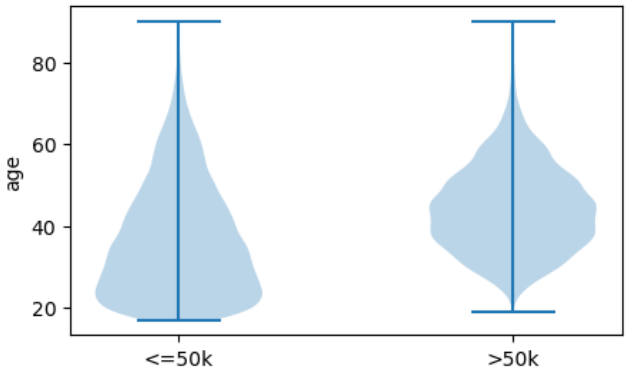
In [18]:
```
df[['age','gross-income']].boxplot(by='gross-income',figsize=(5,3))
plt.ylabel('age')
plt.show()
```



Continuous vs. categorical columns

- violin plot

In [19]:
```
dataset = [df[df['gross-income']==' <=50K']['age'].values,
           df[df['gross-income']==' >50K']['age'].values]

plt.figure(figsize=(5,3))

plt.violinplot(dataset = dataset)
plt.xticks([1,2],['<=50k','>50k'])
plt.ylabel('age')
plt.show()
```



Quiz 2

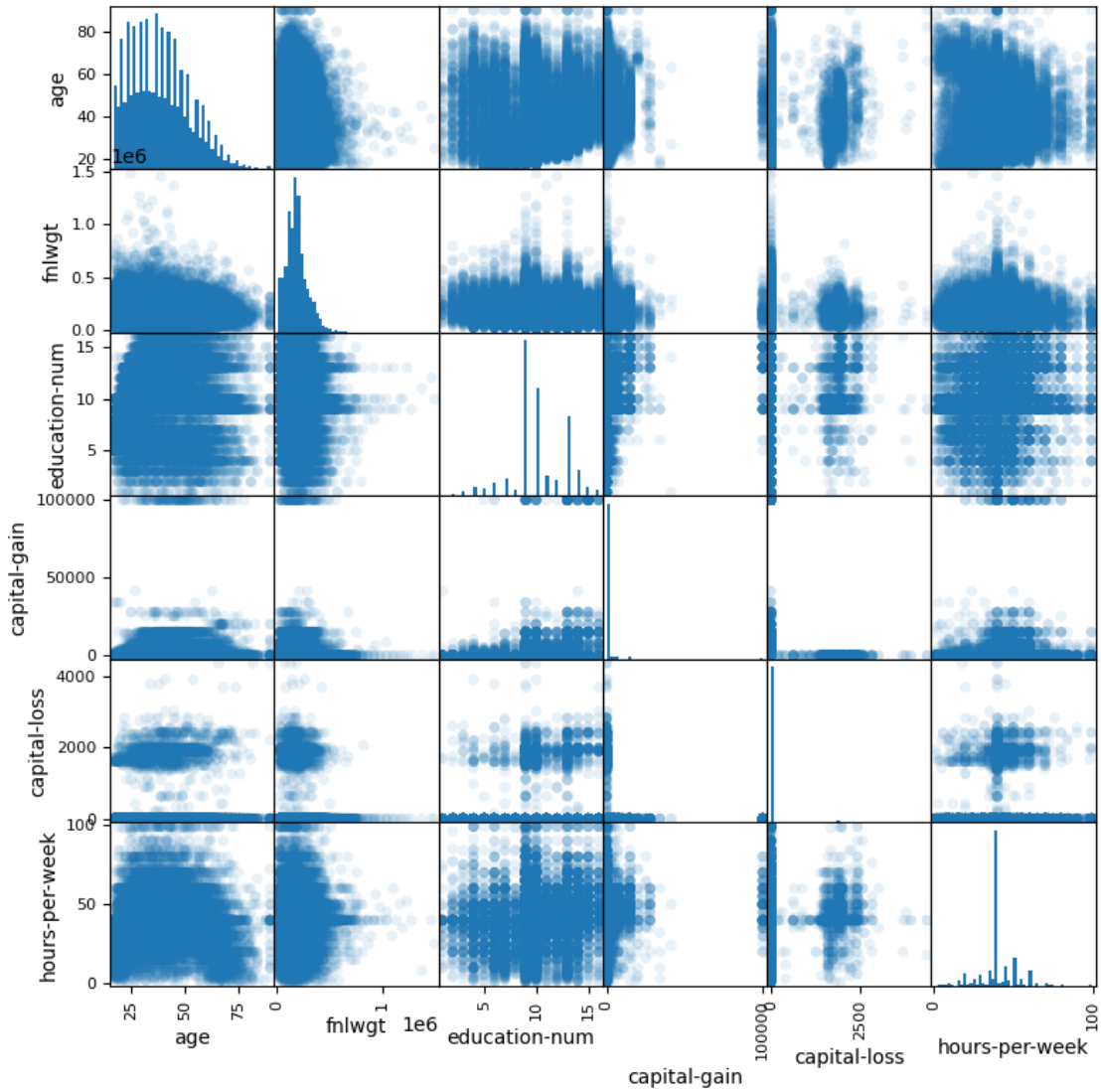
Pair the column name(s) with the appropriate visualization type!

By the end of this lecture, you will be able to

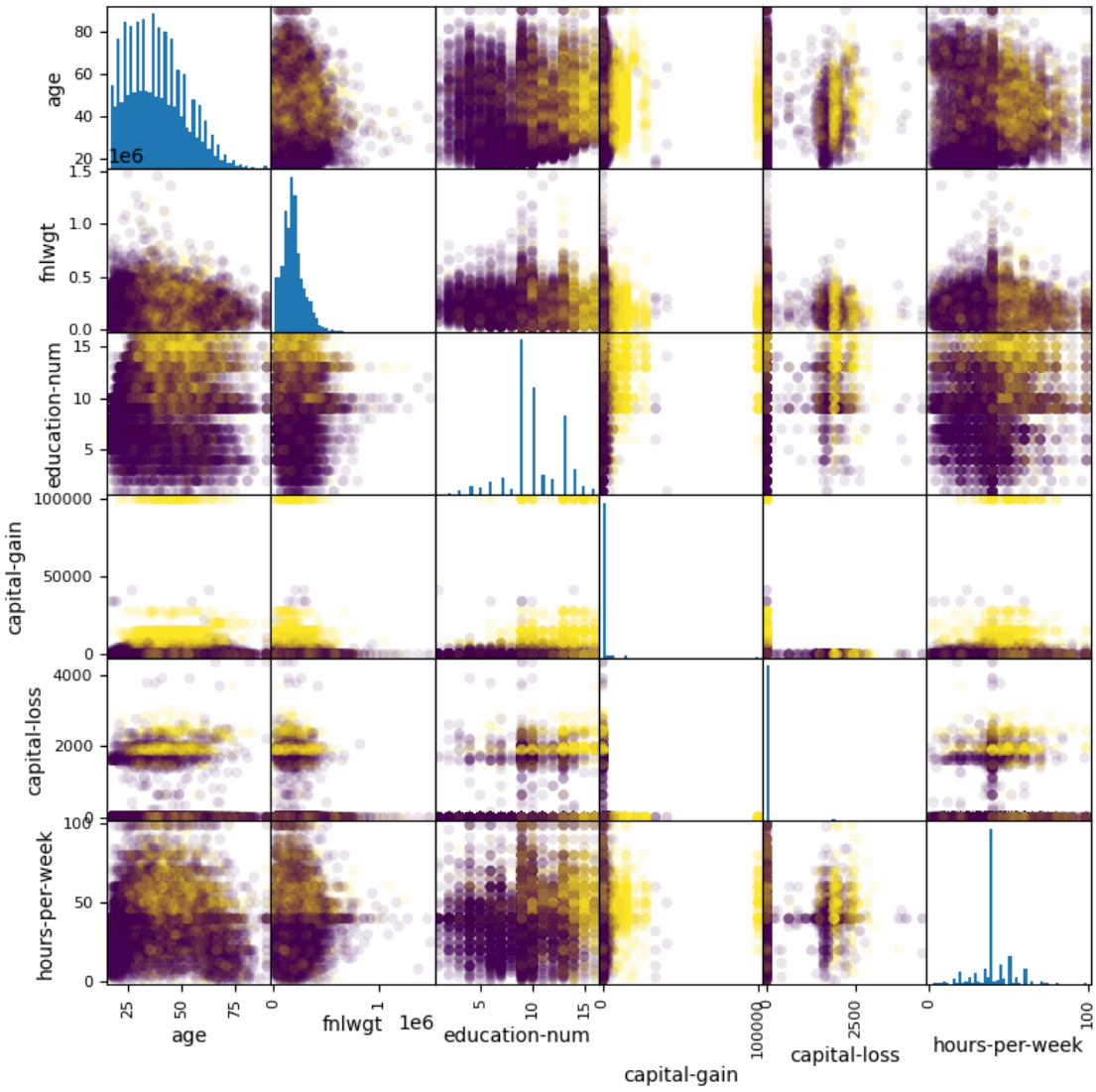
- visualize one column (categorical or continuous data)
- visualize column pairs (all variations of continuous and categorical columns)
- **visualize multiple columns simultaneously**

Scatter matrix

In [20]:
```
pd.plotting.scatter_matrix(df.select_dtypes(int), figsize=(9, 9), marker='o',hist_kwds={'bins': 50},
                           s=30, alpha=.1)
plt.show()
```



In [21]:
```
pd.plotting.scatter_matrix(df.select_dtypes(int), figsize=(9, 9),c = pd.get_dummies(df['gross-income']).iloc[:,1]
                           marker='o',hist_kwds={'bins': 50}, s=30, alpha=.1)
plt.show()
```



By now, you can

- visualize one column (continuous or categorical data)
- visualize column pairs (all variations of continuous and categorical columns)
- visualize multiple columns simultaneously

Matplotlib cheatsheets!

The cheatsheets in this repo are excellent. Feel free to use them any time!

Other great resources for visualization

https://www.data-to-viz.com/

https://pyviz.org/

Mud card

In [ ]: