In [1]:
```
# Initialize Otter
import otter
grader = otter.Notebook("hw1-brfss.ipynb")
```

In [2]:
```
import numpy as np
import pandas as pd
import altair as alt
```

Background

The Behavioral Risk Factor Surveillance System (BRFSS) is a long-term effort administered by the CDC to collect data on behaviors affecting physical and mental health, past and present health conditions, and access to healthcare among U.S. residents. The BRFSS comprises telephone surveys of U.S. residents conducted annually since 1984; in the last decade, over half a million interviews have been conducted each year. This is the largest such data collection effort in the world, and many countries have developed similar programs. The objective of the program is to support monitoring and analysis of factors influencing public health in the United States.

Each year, a standard survey questionnaire is developed that includes a core component comprising questions about: demographic and household information; health-related perceptions, conditions, and behaviors; substance use; and diet. Trained interviewers in each state call randomly selected telephone (landline and cell) numbers and administer the questionnaire; the phone numbers are chosen so as to obtain a representative sample of all households with telephone numbers. Take a moment to read about the 2019 survey here.

In this assignment you'll import and subsample the BRFSS 2019 data and perform a simple descriptive analysis exploring associations between adverse childhood experiences, health perceptions, tobacco use, and depressive disorders. This is an opportunity to practice:

- review of data documentation
- data assessment and critical thinking about data collection
- dataframe transformations in pandas
- communicating and interpreting grouped summaries

Data import and assessment

The cell below imports select columns from the 2019 dataset as a pandas DataFrame. The file is big, so this may take a few moments. Run the cell and then have a quick look at the first few rows and columns.

In [3]:
```
# store variable names of interest
selected_vars = ['_SEX', '_AGEG5YR',
                 'GENHLTH', 'ACEPRISN',
                 'ACEDRUGS', 'ACEDRINK',
                 'ACEDEPRS', 'ADDEPEV3',
                 '_SMOKER3', '_LLCPWT']

# import full 2019 BRFSS dataset
brfss = pd.read_csv('data/brfss2019.zip', compression = 'zip', usecols = sel

# invert sampling weights
brfss['_LLCPWT'] = 1/brfss._LLCPWT

# print first few rows
brfss.head()
```

Out[3]:

	GENHLTH	ADDEPEV3	ACEDEPRS	ACEDRINK	ACEDRUGS	ACEPRISN	_LLCPWT	_SEX
0	3.0	2.0	2.0	2.0	2.0	2.0	0.007391	2.0
1	4.0	2.0	2.0	1.0	2.0	2.0	0.000687	2.0
2	3.0	2.0	2.0	2.0	2.0	2.0	0.004639	2.0
3	4.0	2.0	NaN	NaN	NaN	NaN	0.003827	2.0
4	2.0	2.0	2.0	2.0	2.0	2.0	0.001868	2.0

Question 1: Data dimensions

Check the dimensions of the dataset. Store the dimensions as `nrows` and `ncolumns`.

In [4]:
```
nrows, ncolumns = brfss.shape

print(nrows, ncolumns)
```

```
418268 10
```

In [5]:
```
grader.check("q1")
```

Out[5]:

q1 passed! 🌟

Question 2: Row and column information

Now that you've imported the data, you should verify that the dimensions conform to the format you expect based on data documentation and ensure you understand what each row and each column represents.

Check the number of records (interviews conducted) reported and variables measured for 2019 by reviewing the surveillance summaries by year, and then answer the following questions in a few sentences:

- Does the number of rows match the number of reported records?
- How many columns were imported, and how many columns are reported in the full dataset?
- What does each row in the `brfss` dataframe represent?
- What does each column in the `brfss` dataframe represent

Answer

1. The number of rows does match the number of reported records.
2. 10 columns were reported in the full dataset and 10 collumns were imported.
3. Each row in the brfss dataframe represents a respondent's responses.
4. Each column in the brfss dataframe represents a question.

Question 3: Sampling design and data collection

Skim the overview documentation for the 2019 BRFSS data. Focus specifically the 'Background' and 'Data Collection' sections, read selectively for relevant details, and answer the following questions in a few sentences:

i. Who conducts the interviews and how long does a typical interview last?

ii. Who does an interviewer speak to in each household?

iii. What criteria must a person meet to be interviewed?

iv. Who *can't* appear in the survey? Give two examples.

v. What is the study population (*i.e.*, all individuals who could possibly be sampled)?

vi. Does the data contain any identifying information?

Answer

i. The interviews for the 2019 BRFSS data were conducted by phone by interviewers from the CDC and state health departments. A typical interview lasted about 25 minutes.

ii. Interviewers spoke to one randomly selected adult per household.

iii. To be interviewed, a person must have been aged 18 years or older and have been living in a college/residential housing for 6 or more months.

iv. The survey does not include people under the age of 18 and households without telephones.

v. The study population for the 2019 BRFSS data is all adults aged 18 years and older who own telephones and reside in the United States.

vi. The data does not contain any identifying information, but it does include geographic information and sex information.

Question 4: Variable descriptions

You'll work with the small subset variables imported above: sex, age, general health self-assessment, smoking status, depressive disorder, and adverse childhood experiences (ACEs). The names of these variables as they appear in the raw dataset are defined in the cell in which you imported the data as `selected_vars`. It is often useful, and therefore good practice, to include a brief description of each variable at the outset of any reported analyses, both for your own clarity and for that of any potential readers. Open the 2019 BRFSS codebook in your browser and use text searching to locate each of the variable names of interest. Read the codebook entries and fill in the second column in the table below with a one-sentence description of each variable identified in `selected_vars`. Rephrase the descriptions in your own words -- do not copy the codebook descriptions verbatim.

Variable name	Description
`GENHLTH`	Self-Assessed General Health Status
`_SEX`	Gender
`_AGEG5YR`	Age group (5 year intervals)
`ACEPRISN`	Incarcinerated as a child
`ACEDRUGS`	ACE of living with a a person who has drug problems
`ACEDRINK`	ACE of living with someone who has drinking problems
`ACEDEPRS`	ACE of facing depression below the age of 18
`ADDEPEV3`	Has major depressive dissorder
`_SMOKER3`	Smoking status

Subsampling

To simplify life a little, we'll draw a large random sample of the rows and work with that in place of the full dataset. This is known as **subsampling**.

The cell below draws a random subsample of 10k records. Because the subsample is randomly drawn, we should not expect it to vary in any systematic way from the overall dataset, and distinct subsamples should have similar properties -- therefore, results downstream should be similar to an analysis of the full dataset, and should also be possible to replicate using distinct subsamples.

In [6]:
```
# for reproducibility
np.random.seed(32221)

# randomly sample 10k records
samp = brfss.sample(n = 10000,
                    replace = False,
                    weights = '_LLCPWT')
```

Asides:

- Notice that the random number generator seed is set before carrying out this task -- this ensures that every time the cell is run, the same subsample is drawn. As a result, the computations in this notebook are *reproducible*: when I run the notebook on my computer, I get the same results as you get when you run the notebook on your computer.

- Notice also that *sampling weights* provided with the dataset are used to draw a weighted sample. Some respondents are more likely to be selected than others from the general population of U.S. adults with phone numbers, so the BRFSS calculates derived weights that are inversely proportional to estimates of the probability that the respondent is included in the survey. This is a somewhat sophisticated calculation, however if you're interested, you can read about how these weights are calculated and why in the overview documentation you used to answer the questions above. We use the sampling weights in drawing the subsample so that we get a representative sample of U.S. adults with phone numbers.

- Notice the missing values. How many entries are missing in each column? The cell below computes the proportion of missing values for each of the selected variables. We'll return to this issue later on.

In [7]:
```
# proportions of missingness
samp.isna().mean()
```

Out[7]:
```
GENHLTH     0.0000
ADDEPEV3    0.0000
ACEDEPRS    0.8086
ACEDRINK    0.8088
ACEDRUGS    0.8088
ACEPRISN    0.8088
_LLCPWT     0.0000
_SEX        0.0000
_AGEG5YR    0.0000
_SMOKER3    0.0000
dtype: float64
```

Tidying

In the following series of questions you'll tidy up the subsample by performing these steps:

- selecting columns of interest;
- replacing coded values of question responses with responses;
- defining new variables based on existing ones;
- renaming columns.

The goal of this is to produce a clean version of the dataset that is well-organized, intuitive to navigate, and ready for analysis.

The variable entries are coded numerically to represent certain responses. These should be replaced by more informative entries. We can use the codebook to determine which number means what, and replace the values accordingly.

The cell below replaces the numeric values for `_AGEG5YR` by their meanings, illustrating how to use `.replace()` with a dictionary to convert the numeric coding to interpretable values. The basic strategy is:

1. Store the variable coding for `VAR` as a dictionary `var_codes`.
2. Use `.replace({'VAR': var_codes})` to modify values.

If you need additional examples, check the pandas documentation for `.replace()`.

In [8]:

```
# dictionary representing variable coding
age_codes = {
    1: '18-24', 2: '25-29', 3: '30-34',
    4: '35-39', 5: '40-44', 6: '45-49',
    7: '50-54', 8: '55-59', 9: '60-64',
    10: '65-69', 11: '70-74', 12: '75-79',
    13: '80+', 14: 'Unsure/refused/missing'
}

# recode age categories
samp_mod1 = samp.replace({'_AGEG5YR': age_codes})

# check result
samp_mod1.head()
```

Out[8]:

	GENHLTH	ADDEPEV3	ACEDEPRS	ACEDRINK	ACEDRUGS	ACEPRISN	_LLCPWT	_
237125	5.0	2.0	NaN	NaN	NaN	NaN	0.057004	
329116	5.0	2.0	NaN	NaN	NaN	NaN	0.108336	
178937	3.0	2.0	NaN	NaN	NaN	NaN	0.000998	
410081	4.0	1.0	NaN	NaN	NaN	NaN	0.021973	
184555	2.0	2.0	2.0	2.0	2.0	2.0	0.027175	

Question 5: Recoding variables

Following the example immediately above and referring to the 2019 BRFSS codebook, replace the numeric codings with response categories for each of the following variables:

- `_SEX`
- `GENHLTH`
- `_SMOKER3`

Notice that above, the first modification (slicing) was stored as `samp_mod1`, and was a function of `samp`. You'll follow this pattern, creating `samp_mod2`, `samp_mod3`, and so on so that each step (modification) of your data manipulations is stored separately, for easy troubleshooting.

i. Recode `_SEX`: define a new dataframe `samp_mod2` that is the same as `samp_mod1` but with the `_SEX` variable recoded as `M` and `F`.

ii. Recode `GENHLTH`: define a new dataframe `samp_mod3` that is the same as `samp_mod2` but with the `GENHLTH` variable recoded as `Excellent`, `Very good`, `Good`, `Fair`, `Poor`, `Unsure`, and `Refused`.

iii. Recode `_SMOKER3`: define a new dataframe `samp_mod4` that is the same as `samp_mod3` but with `_SMOKER3` recoded as `Daily`, `Some days`, `Former`, `Never`, and `Unsure/refused/missing`.

iv. Print the first few rows of `samp_mod4`.

In [9]:
```
# define dictionary for sex
sex_codes = {1: 'M', 2: 'F'}

# recode sex
samp_mod2 = samp_mod1.replace({'_SEX': sex_codes})

# define dictionary for health
health_codes = {1: 'Excellent',
                2: 'Very good',
                3: 'Good',
                4: 'Fair',
                5: 'Poor',
                7: 'Unsure',
                9: 'Refused'}

# recode health
samp_mod3 = samp_mod2.replace({'GENHLTH': health_codes})

# define dictionary for smoking
smoke_codes = {1: 'Daily',
               2: 'Some days',
               3: 'Former',
               4: 'Never',
               7: 'Unsure/refused/missing'}

# recode smoking
samp_mod4 = samp_mod3.replace({'_SMOKER3': smoke_codes})

# print a few rows
print(samp_mod4.head())
```

```
          GENHLTH  ADDEPEV3  ACEDEPRS  ACEDRINK  ACEDRUGS  ACEPRISN   _LLCP
WT  \
237125       Poor       2.0       NaN       NaN       NaN       NaN  0.0570
04
329116       Poor       2.0       NaN       NaN       NaN       NaN  0.1083
36
178937       Good       2.0       NaN       NaN       NaN       NaN  0.0009
98
410081       Fair       1.0       NaN       NaN       NaN       NaN  0.0219
73
184555  Very good       2.0       2.0       2.0       2.0       2.0  0.0271
75

       _SEX _AGEG5YR   _SMOKER3
237125    F    25-29     Former
329116    F      80+     Former
178937    M    18-24      Never
410081    F    45-49  Some days
184555    F      80+     Former
```

In [10]:
```
grader.check("q5")
```

Out[10]:

q5 passed! 🚀

Question 6: Value replacement

Now all the variables *except* the adverse childhood experience and depressive disorder question responses are represented interpretably. In the codebook that the answer key is identical for these remaining variables.

The numeric codings can be replaced all at once by applying `.replace()` to the dataframe with an argument of the form

- `df.replace({'var1': varcodes1, 'var2': varcodes1, ..., 'varp': varcodesp})`

Define a new dataframe `samp_mod5` that is the same as `samp_mod4` but with the remaining variables recoded according to the answer key `Yes`, `No`, `Unsure`, `Refused`. Print the first few rows of the result using `.head()`.

In [11]:

```
# define dictionary
answer_codes = {1: 'Yes',
                2: 'No',
                7: 'Unsure',
                9: 'Refused'}

# recode
samp_mod5 = samp_mod4.replace({'ACEPRISN': answer_codes,
                               'ACEDRUGS': answer_codes,
                               'ACEDRINK': answer_codes,
                               'ACEDEPRS': answer_codes,
                               'ADDEPEV3': answer_codes})

# check using head()
print(samp_mod5.head())
```

```
         GENHLTH ADDEPEV3 ACEDEPRS ACEDRINK ACEDRUGS ACEPRISN   _LLCPWT _S
EX  \
237125        Poor       No      NaN      NaN      NaN      NaN  0.057004
F
329116        Poor       No      NaN      NaN      NaN      NaN  0.108336
F
178937        Good       No      NaN      NaN      NaN      NaN  0.000998
M
410081        Fair      Yes      NaN      NaN      NaN      NaN  0.021973
F
184555   Very good       No       No       No       No       No  0.027175
F

       _AGEG5YR   _SMOKER3
237125    25-29     Former
329116      80+     Former
178937    18-24      Never
410081    45-49  Some days
184555      80+     Former
```

In [12]:

```python
grader.check("q6")
```

Out[12]:

q6 passed! 🙌

Finally, all the variables in the dataset are categorical. Notice that the current data types do not reflect this.

In [13]:

```python
samp_mod5.dtypes
```

Out[13]:

```
GENHLTH      object
ADDEPEV3     object
ACEDEPRS     object
ACEDRINK     object
ACEDRUGS     object
ACEPRISN     object
_LLCPWT     float64
_SEX         object
_AGEG5YR     object
_SMOKER3     object
dtype: object
```

Let's coerce the variables to `category` data types using `.astype()`.

In [14]:

```python
# coerce to categorical
samp_mod6 = samp_mod5.astype('category')

# check new data types
samp_mod6.dtypes
```

Out[14]:
```
GENHLTH     category
ADDEPEV3    category
ACEDEPRS    category
ACEDRINK    category
ACEDRUGS    category
ACEPRISN    category
_LLCPWT     category
_SEX        category
_AGEG5YR    category
_SMOKER3    category
dtype: object
```

Question 7: Define ACE indicator variable

Downstream analysis of ACEs will be facilitated by having an indicator variable that is a `1` if the respondent answered 'Yes' to any ACE question, and a `0` otherwise -- that way, you can easily count the number of respondents reporting ACEs by summing up the indicator or compute the proportion by taking an average.

To this end, define a new logical variable:

- `adverse_conditions`: did the respondent answer yes to any of the adverse childhood condition questions?

You can accomplish this task in several steps:

1. Obtain a logical array indicating the positions of the ACE variables (hint: use `.columns` to obtain the column index and operate on the result with `.str.startswith(...)`.). Store this as `ace_positions`.
2. Use the logical array `ace_positions` to select the ACE columns via `.loc[]`. Store this as `ace_data`.
3. Obtain a dataframe that indicates whether each entry is a 'Yes' (hint: use the boolean operator `==`, which is a vectorized operation). Store this as `ace_yes`.
4. Compute the row sums using `.sum()`. Store this as `ace_numyes`.
5. Define the new variable as `ace_numyes > 0`.

Store the result as `samp_mod7`, and print the first few rows using `.head()`.

In [15]:

```
# copy samp_mod6
samp_mod7 = samp_mod6.copy()

# ace column positions
ace_positions = samp_mod7.columns.str.startswith('ACE')

# ace data
ace_data = samp_mod7.loc[:, ace_positions]

# ace yes indicators
ace_yes = ace_data == 'Yes'

# number of yesses
ace_numyes = ace_yes.sum(axis=1)

# assign new variable
samp_mod7['adverse_conditions'] = ace_numyes > 0

# check result using .head()
print(samp_mod7.head())
```

```
          GENHLTH ADDEPEV3 ACEDEPRS ACEDRINK ACEDRUGS ACEPRISN   _LLCPWT _S
EX  \
237125       Poor       No      NaN      NaN      NaN      NaN  0.057004
F
329116       Poor       No      NaN      NaN      NaN      NaN  0.108336
F
178937       Good       No      NaN      NaN      NaN      NaN  0.000998
M
410081       Fair      Yes      NaN      NaN      NaN      NaN  0.021973
F
184555  Very good       No       No       No       No       No  0.027175
F

       _AGEG5YR   _SMOKER3  adverse_conditions
237125    25-29     Former               False
329116      80+     Former               False
178937    18-24      Never               False
410081    45-49  Some days               False
184555      80+     Former               False
```

In [16]:

```
grader.check("q7")
```

Out[16]:

q7 passed! 🌟

Question 8: Define missingness indicator variable

As you saw earlier, there are some missing values for the ACE questions. These arise whenever a respondent is not asked these questions. In fact, answers are missing for nearly 80% of the respondents in our subsample. We should keep track of this information. Define a missing indicator:

- `adverse_missing` : is a response missing for at least one of the ACE questions?

In [17]:
```
# copy modification 7
samp_mod8 = samp_mod7.copy()

# define missing indicator
ace_missing = samp_mod8.loc[:,samp_mod8.columns.str.startswith('ACE')].isna(
ace_summissing = np.sum(ace_missing, axis=1)
samp_mod8['adverse_missing'] = (ace_summissing > 0).astype(int)
# check using head()
print(samp_mod8.head())
```

```
          GENHLTH ADDEPEV3 ACEDEPRS ACEDRINK ACEDRUGS ACEPRISN   _LLCPWT _S
EX  \
237125       Poor       No      NaN      NaN      NaN      NaN  0.057004
F
329116       Poor       No      NaN      NaN      NaN      NaN  0.108336
F
178937       Good       No      NaN      NaN      NaN      NaN  0.000998
M
410081       Fair      Yes      NaN      NaN      NaN      NaN  0.021973
F
184555  Very good       No       No       No       No       No  0.027175
F

       _AGEG5YR   _SMOKER3  adverse_conditions  adverse_missing
237125    25-29     Former               False                1
329116      80+     Former               False                1
178937    18-24      Never               False                1
410081    45-49  Some days               False                1
184555      80+     Former               False                0
```

In [18]:
```
grader.check("q8")
```

Out[18]:

q8 passed! 🚀

Question 9: Filter respondents who did not answer ACE questions

Since values are missing for the ACE question if a respondent was not asked, we can remove these observations and do any analysis *conditional on respondents having been asked the ACE questions*. Use your indicator variable `adverse_missing` to filter out respondents who were not asked the ACE questions.

Note that this dramatically limits the scope of inference for subsequent analyses to only those locations where the ACE module was included in the survey.

In [19]:
```
samp_mod9 = samp_mod8[samp_mod8['adverse_missing']==0]
```

In [20]:
```
grader.check("q9")
```

Out[20]:

q9 passed! 🎉

Question 10: Define depression indicator variable

It will prove similarly helpful to define an indicator for reported depression:

- `depression` : did the respondent report having been diagnosed with a depressive disorder?

Follow the same strategy as above for the ACE variables, and store the result as `samp_mod10`. See if you can perform the calculation of the new variable in a single line of code. Print the first few rows using `.head()`.

In [21]:
```
# create a new DataFrame with the same contents as samp_mod10
samp_mod10 = pd.DataFrame(samp_mod9)

# add a new column called 'depression' to the DataFrame
samp_mod10['depression'] = np.where(samp_mod10['ADDEPEV3'] == 'Yes', True, F

# display the first few rows of the updated DataFrame
print(samp_mod10.head())
```

```
          GENHLTH ADDEPEV3 ACEDEPRS ACEDRINK ACEDRUGS ACEPRISN   _LLCPWT _S
EX  \
184555  Very good       No       No       No       No       No  0.027175
F
315931       Poor       No       No       No       No       No  0.019520
F
326538  Excellent       No      Yes       No       No       No  0.001009
F
61521   Very good       No       No       No       No       No  0.012117
F
74165        Good      Yes      Yes      Yes      Yes      Yes  0.000891
F

       _AGEG5YR _SMOKER3  adverse_conditions  adverse_missing  depression
184555      80+   Former               False                0       False
315931      80+   Former               False                0       False
326538    50-54   Former                True                0       False
61521       80+    Never               False                0       False
74165     18-24    Never                True                0        True
```

In [22]:
```
grader.check("q10")
```

Out[22]:

q10 passed! 🚀

Question 11: Final dataset

For the final dataset, drop the respondent answers to individual questions, the missingness indicator, and select just the derived indicator variables along with general health, sex, age, and smoking status. Check the pandas documentation for `.rename()` and follow the examples to rename the latter variables:

- general_health
- sex
- age
- smoking

See if you can perform both operations (slicing and renaming) in a single chain. Store the result as `data`.

In [23]:
```
samp_mod10.columns
```

Out[23]:
```
Index(['GENHLTH', 'ADDEPEV3', 'ACEDEPRS', 'ACEDRINK', 'ACEDRUGS', 'ACEPRISN
',
       '_LLCPWT', '_SEX', '_AGEG5YR', '_SMOKER3', 'adverse_conditions',
       'adverse_missing', 'depression'],
      dtype='object')
```

In [24]:
```
# slice and rename
data = samp_mod10.drop(ace_data.columns, axis=1).drop('adverse_missing', axi

# check using .head()
data.head()
```

Out[24]:

	general_health	sex	age	smoking	adverse_conditions	depression
184555	Very good	F	80+	Former	False	False
315931	Poor	F	80+	Former	False	False
326538	Excellent	F	50-54	Former	True	False
61521	Very good	F	80+	Never	False	False
74165	Good	F	18-24	Never	True	True

In [25]:
```
grader.check("q11")
```

Out[25]:

q11 passed! 🍀

Descriptive analysis

Now that you have a clean dataset, you'll use grouping and aggregation to compute several summary statistics that will help you explore whether there is an apparent association between experiencing adverse childhood conditions and self-reported health, smoking status, and depressive disorders in areas where the ACE module was administered.

The basic strategy will be to calculate the proportions of respondents who answered yes to one of the adverse experience questions when respondents are grouped by the other variables.

Question 12: Proportion of respondents reporting ACEs

Calculate the overall proportion of respondents in the subsample that reported experiencing at least one adverse condition (given that they answered the ACE questions). Use `.mean()`; store the result as `mean_ace` and print.

In [26]:
```
# proportion of respondents reporting at least one adverse condition
# calculate the proportion of respondents reporting at least one adverse cor
mean_ace = samp_mod10['adverse_conditions'].mean()

# print the proportion
print(mean_ace)
```

```
0.3070083682008368
```

In [27]:

```python
grader.check("q12")
```

Out[27]:

q12 passed! 🎉

Does the proportion of respondents who reported experiencing adverse childhood conditions vary by general health?

The cell below computes the porportion separately by general health self-rating. Notice that the depression variable is dropped so that the result doesn't also report the proportion of respondents reporting having been diagnosed with a depressive disorder. Notice also that the proportion of missing values for respondents indicating each general health rating is shown.

In [28]:

```python
# proportions grouped by general health
data.drop(
    columns = 'depression'
).groupby(
    'general_health'
).mean(numeric_only = True)
```

Out[28]:

	adverse_conditions
general_health	
Excellent	0.300000
Fair	0.355491
Good	0.299174
Poor	0.441667
Refused	0.000000
Unsure	0.000000
Very good	0.264957

Notice that the row index lists the general health rating out of order. This can be fixed using a `.loc[]` call and the dictionary that was defined for the variable coding.

In [29]:
```
# same as above, rearranging index
ace_health = data.drop(
    columns = 'depression'
).groupby(
    'general_health'
).mean(
    numeric_only = True
).loc[list(health_codes.values()), :]

# print
ace_health
```

Out[29]:

	adverse_conditions
general_health	
Excellent	0.300000
Very good	0.264957
Good	0.299174
Fair	0.355491
Poor	0.441667
Unsure	0.000000
Refused	0.000000

Question 13: Association between smoking status and ACEs

Does the proportion of respondents who reported experiencing adverse childhood conditions vary by smoking status?

Following the example above for computing the proportion of respondents reporting ACEs by general health rating, calculate the proportion of respondents reporting ACEs by smoking status (be sure to arrange the rows in appropriate order of smoking status) and store as `ace_smoking`.

In [30]:
```
# proportions grouped by smoking status

ace_smoking = data.drop(
    columns = 'depression'
).groupby(
    'smoking'
).mean(
    numeric_only = True
)


# print
ace_smoking
```

Out[30]:

	adverse_conditions
smoking	
9.0	0.100000
Daily	0.453125
Former	0.334459
Never	0.251434
Some days	0.527778

In [31]:
```
grader.check("q13")
```

Out[31]:

q13 passed! 🌟

Question 14: Association between depression and ACEs

Does the proportion of respondents who reported experiencing adverse childhood conditions vary by smoking status?

Calculate the proportion of respondents reporting ACEs by whether respondents had been diagnosed with a depressive disorder and store as `ace_depr` .

In [32]:
```
# proportions grouped by having experienced depression
ace_depr = data.groupby(
    'depression'
).mean(
    numeric_only = True)

# print
ace_depr
```

Out[32]:

	adverse_conditions
depression	
False	0.250975
True	0.537433

In [33]:
```
grader.check("q14")
```

Out[33]:

q14 passed! 🎉

Question 15: Exploring subgroupings

Does the apparent association between general health and ACEs persist after accounting for sex?

Repeat the calculation of the proportion of respondents reporting ACEs by general health rating, but also group by sex. Store the result as `ace_health_sex` .

In [34]:
```
# group by general health and sex
ace_health_sex = data.drop(
    columns = 'depression'
).groupby(
    ['general_health', 'sex']).mean().loc[list(health_codes.values()), :]
```

```
/tmp/ipykernel_209/582600695.py:5: FutureWarning: The default value of nume
ric_only in DataFrameGroupBy.mean is deprecated. In a future version, numer
ic_only will default to False. Either specify numeric_only or select only c
olumns which should be valid for the function.
  ['general_health', 'sex']).mean().loc[list(health_codes.values()), :]
```

In [35]:
```
grader.check("q15")
```

Out[35]:

q15 passed! 💯

The cell below rearranges the table a little for better readability.

In [36]:
```
# pivot table for better display
ace_health_sex.reset_index().pivot(columns = 'sex', index = 'general_health'
```

Out[36]:

sex	F	M
general_health		
Excellent	0.328671	0.261682
Very good	0.282123	0.237885
Good	0.308108	0.285106
Fair	0.367150	0.338129
Poor	0.549296	0.285714
Unsure	NaN	0.000000
Refused	0.000000	NaN

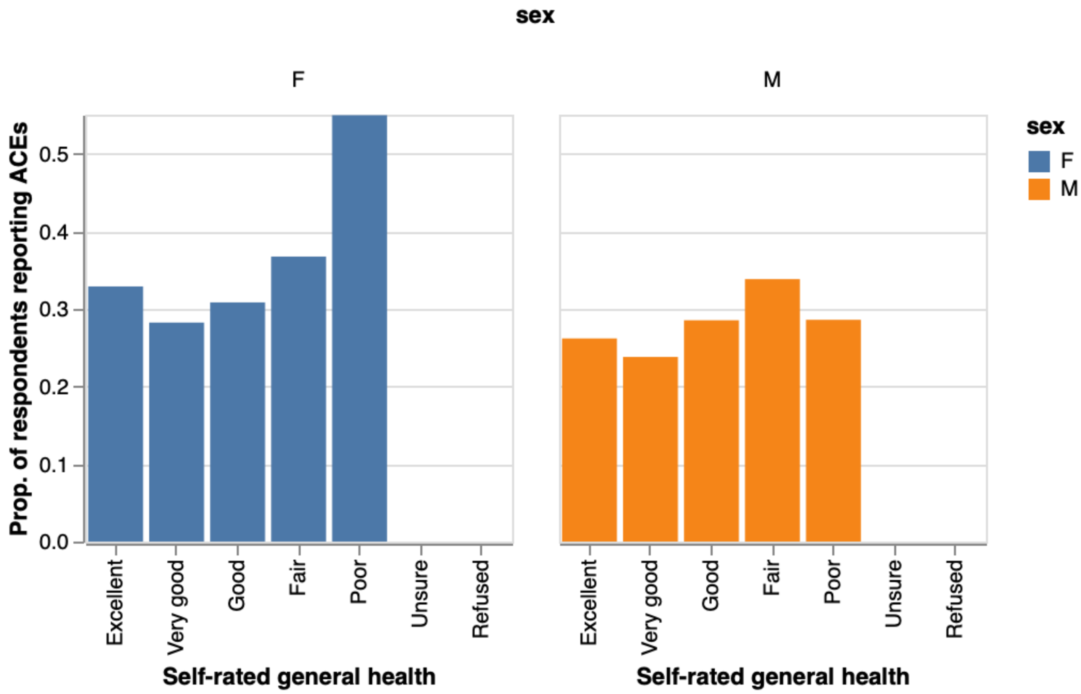
Even after rearrangement, the table in the last question is a little tricky to read (few people like visually scanning tables). This information would be better displayed in a plot. The example below generates a bar chart showing the summaries you calculated in Q2(d), with the proportion on the y axis, the health rating on the x axis, and separate bars for the two sexes.

In [37]:

```
# coerce indices to columns for plotting
plot_df = ace_health_sex.reset_index()

# specify order of general health categories
genhealth_order = list(health_codes.values())
plot_df.general_health.cat.set_categories(genhealth_order, inplace=True)
plot_df.sort_values(["general_health"], inplace=True)

# plot
alt.Chart(plot_df).mark_bar().encode(
    x = alt.X('general_health',
              sort = ['general_health'],
              title = 'Self-rated general health'),
    y = alt.Y('adverse_conditions',
              title = 'Prop. of respondents reporting ACEs'),
    color = 'sex',
    column = 'sex'
).properties(
    width = 200,
    height = 200
)
```

```
/tmp/ipykernel_209/2150558614.py:6: FutureWarning: The `inplace` parameter in pandas.Categorical.set_categories is deprecated and will be removed in a future version. Removing unused categories will always return a new Categorical object.
  plot_df.general_health.cat.set_categories(genhealth_order, inplace=True)
```

Out[37]:

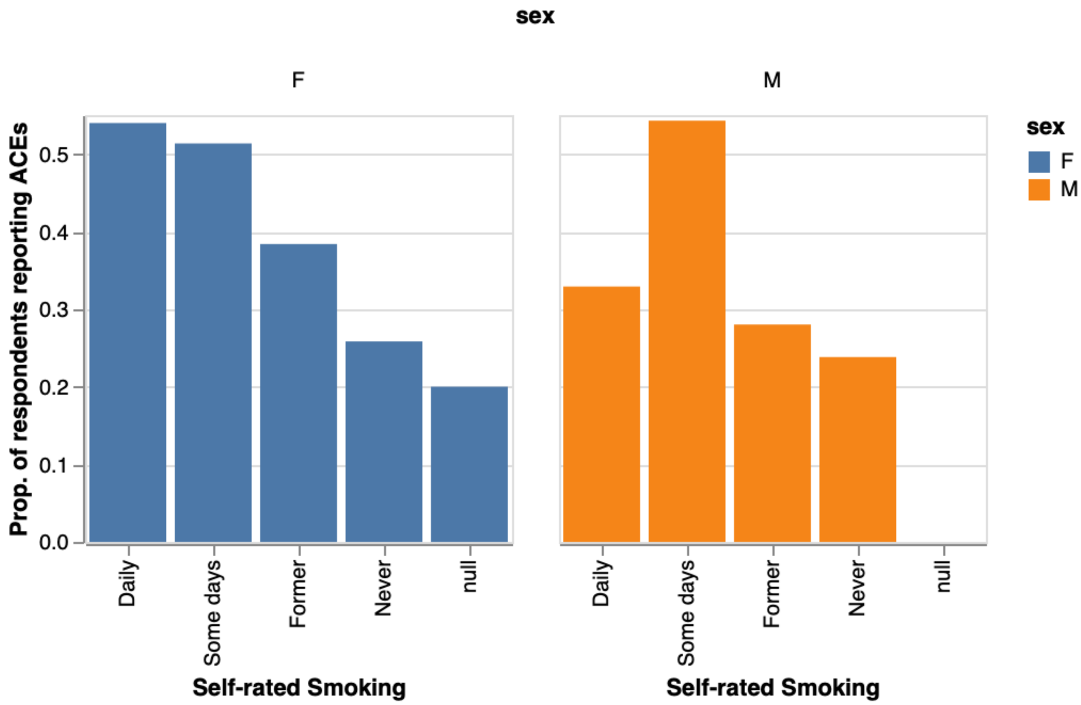


In [38]:

```
# dataframe of proportions grouped by smoking status
ace_smoking_sex = data.drop(
    columns = 'depression'
).groupby(
    ['smoking','sex']
).mean(numeric_only = True)

# coerce indices to columns for plotting
plot_df = ace_smoking_sex.reset_index()

# specify order of general health categories
genhealth_order = list(smoke_codes.values())
plot_df.smoking.cat.set_categories(genhealth_order, inplace=True)
plot_df.sort_values(["smoking"], inplace=True)

# plot
alt.Chart(plot_df).mark_bar().encode(
    x = alt.X('smoking',
              sort = ['smoking'],
              title = 'Self-rated Smoking'),
    y = alt.Y('adverse_conditions',
              title = 'Prop. of respondents reporting ACEs'),
    color = 'sex',
    column = 'sex'
).properties(
    width = 200,
    height = 200
)
```

```
/tmp/ipykernel_209/1593214795.py:13: FutureWarning: The `inplace` parameter
in pandas.Categorical.set_categories is deprecated and will be removed in a
future version. Removing unused categories will always return a new Categor
ical object.
  plot_df.smoking.cat.set_categories(genhealth_order, inplace=True)
```

Out[38]:



Communicating results

Here you'll be asked to reflect briefly on your findings.

Question 17: Summary

Is there an observed association between reporting ACEs and general health, smoking status, and depression among survey respondents who answered the ACE questions?

Write a two to three sentence answer to the above question summarizing your findings. State an answer to the question in your first sentence, and then in your second/third sentences describe exactly what you observed in the foregoing descriptive analysis of the BRFSS data. Be precise, but also concise. There is no need to describe any of the data manipulations, survey design, or the like.

Answer

There is an association between health and smoking status. Those who are smokers are likely to report lower health conditions. Furthermore, there is a correlation between depression and those who reported smoking.

Question 18: Scope of inference

Recall from the overview documentation all the care that the BRFSS dedicates to collecting a representative sample of the U.S. adult population with phone numbers. Do you think that your findings provide evidence of an association among the general public (not just the individuals survey)? Why or why not? Answer in two sentences.

Answer

I do not think these findings provide any evidence for association for the general public. Since children are included in the general public and did not get counted in the survey it can only be applied to the selected population.

Question 19: Bias

What is a potential source of bias in the survey results, and how might this affect the proportions you've calculated?

Answer in one or two sentences.

Answer

One source of bias in the survey is the order of which the questions are presented. Each question can build off the previous question pushing for an answer that might not have normally come out of the interviewers mouth. It could also be on the otherhand where the interviewer did not ask questions that were related as much which is less likely to induce an answer. This biases can effect the proporitons of the ACE questions.

Comment

Notice that the language 'association' is non-causual: we don't say that ACEs cause (or don't cause) poorer health outcomes. This is intentional, because the BRFSS data are what are known as 'observational' data, *i.e.* not originating from a controlled experiment. There could be unobserved factors that explain the association.

To take a simple example, dog owners live longer, but the reason is simply that dog owners walk more -- so it's the exercise, not the dogs, that cause an increase in longevity. An observational study that doesn't measure exercise would show a positive association between dog ownership and lifespan, but it's a non-causal relationship.

(As an interesting/amusing aside, there is a well known study that established an association between birdkeeping and lung cancer; obviously this is non-causal, yet the study authors recommended that individuals at high risk for cancer avoid 'avian exposure', as they were unsure of the mechanism.)

So there could easily be unobserved factors that account for the observed association in the BRFSS data. We guard against over-interpreting the results by using causally-neutral language.

Submission

1. Save the notebook.
2. Restart the kernel and run all cells. (**CAUTION**: if your notebook is not saved, you will lose your work.)
3. Carefully look through your notebook and verify that all computations execute correctly. You should see **no errors**; if there are any errors, make sure to correct them before you submit the notebook.
4. Download the notebook as an `.ipynb` file. This is your backup copy.
5. Export the notebook as PDF and upload to Gradescope.

To double-check your work, the cell below will rerun all of the autograder tests.

In [39]:
```
grader.check_all()
```

Out[39]:

```
q1 results: All test cases passed!

q10 results: All test cases passed!

q11 results: All test cases passed!

q12 results: All test cases passed!

q13 results: All test cases passed!

q14 results: All test cases passed!

q15 results: All test cases passed!

q5 results: All test cases passed!

q6 results: All test cases passed!

q7 results: All test cases passed!

q8 results: All test cases passed!

q9 results: All test cases passed!
```