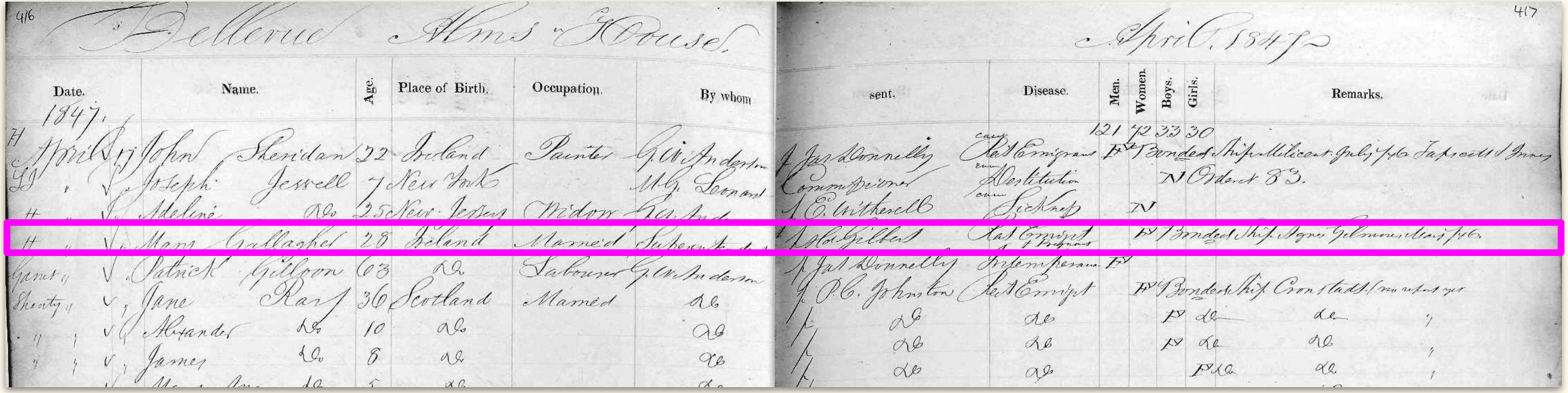
- Bellevue Almshouse dataset
  - Description of the dataset
  - Explore and filter data
  - Dealing with missing data
  - Manipulating columns

Bellevue Almshouse Dataset

Bellevue Almshouse admission ledger



Digitizing the Bellevue Almshouse admission ledger

	date_in	first_name	last_name	age	disease	profession	gender	children
0	1847-04-17	Mary	Gallagher	28.0	recent emigrant	married	w	Child Alana 10 days
1	1847-04-08	John	Sanin (?)	19.0	recent emigrant	laborer	m	Catherine 2 mo
2	1847-04-17	Anthony	Clark	60.0	recent emigrant	laborer	m	Charles Riley afed 10 days
3	1847-04-08	Lawrence	Feeney	32.0	recent emigrant	laborer	m	Child
4	1847-04-13	Henry	Joyce	21.0	recent emigrant	NaN	m	Child 1 mo
...	...	...	...	...	...	...	...	...
9579	1847-06-17	Mary	Smith	47.0	NaN	NaN	w	NaN
9580	1847-06-22	Francis	Riley	29.0	lame	superintendent	m	NaN
9581	1847-07-02	Martin	Dunn	4.0	NaN	NaN	m	NaN
9582	1847-07-08	Elizabeth	Post	32.0	NaN	NaN	w	NaN
9583	1847-04-28	Bridget	Ryan	28.0	destitution	spinster	w	NaN

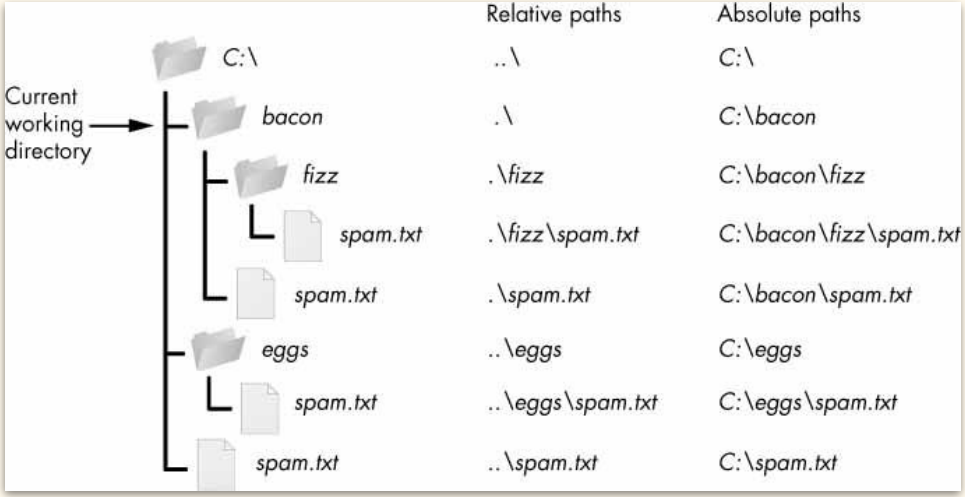
9584 rows × 8 columns

Import pandas and read in a CSV file

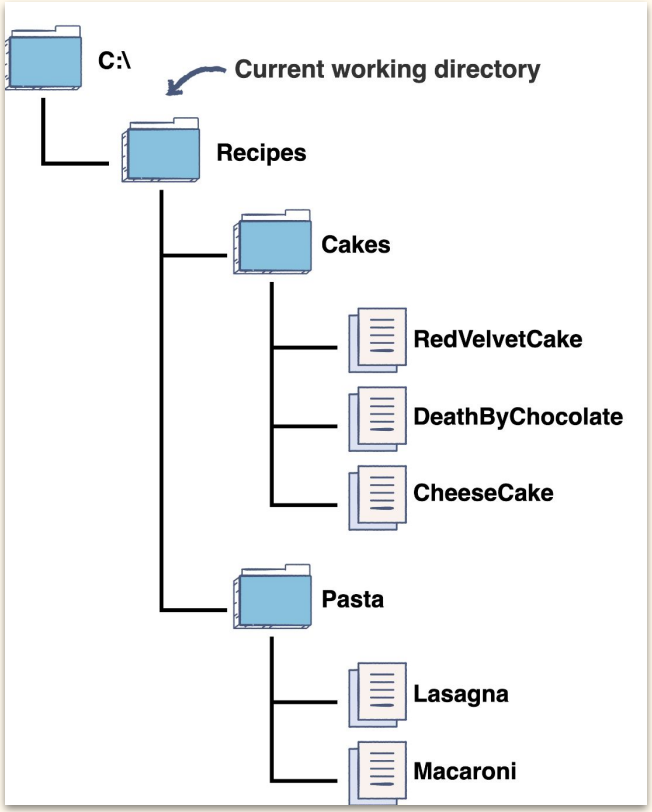
- import pandas as pd
- pd.read_csv(‘filepath’)

File paths

- There are two ways to specify a file path:
  - An **absolute path**, which always begins with the root folder
  - A **relative path**, which is relative to the current working directory



Let’s practice: Finding our way to our files!



For each file, write out the absolute and relative file path

- Path to CheeseCake
- Path to Lasagna

Description of the dataset

- .info():
  - Displays the total count of non-N/A, non-blank items, and the datatype of each column
- .head(n):
  - Provides the first *n* of rows
- .sample(n)
  - Provides a random *n* of rows

Summary statistics

- .describe(include = ‘all’)
  - Provides the summary statistics of all the variables in the dataframe
- Measures central tendency
  - Mean, median, mode
  - A value that represents the middle or centre of its distribution
- Measures spread of distribution
  - How far are the values spread from the smallest value to the largest value

Dealing with duplicates

- .duplicated(keep = ‘first’/’last’/False):
  - Creates a True/False dataframe to check which rows in the original dataframe are duplicated
  - keep
    - first: considers the first entry in the dataframe as the unique entry
    - last: considers the last entry in the dataframe as the unique entry
    - False: considers all entry as duplicates
  - Default argument: keep = ‘first’

Dealing with duplicates

- df[df.duplicated(keep=False)]
  - Selects duplicated rows from the original dataframe that fulfills the True/False dataframe conditions
- .drop_duplicate(keep = ‘first’/’last’/False):
  - Drops all the duplicated rows and keeps the first entry, last entry, or none of the entries
  - Default argument: keep = ‘first’

Frequency: Most common items in a column

- df[“column_name”].value_counts()
  - To count the number of unique values in a column

Missing Data

- .isna() / .notna()
  - Creates True/False table for values with/out NA
    - dataframe_variable[‘column name'].notna()]
    - bellevue_df['professions'].notna()
  - Filters out NA values by comparing to original df
    - dataframe_variable[dataframe_variable[‘column name'].notna()]
    - e.g. bellevue_df[bellevue_df['professions'].notna()]

Missing Data

- .count()
  - count() method always excludes NaN values
  - To find the percentage of not blank data in every column:
    - bellevue_df.count() / len(bellevue_df)
- .fillna()
  - Fill the NaN values in the DataFrame with a different value by using the .fillna() method
    - bellevue_df['professions'].fillna('no profession information recorded')

Rename Columns

- .rename(columns={})
  - bellevue_df.rename(columns={‘professions': ‘jobs’})
  - To save the new column name to the dataframe, we need to overwrite the variable
    - bellevue_df = bellevue_df.rename(columns={‘professions': ‘jobs’})

Drop Columns

- .drop(columns=”column name”)
  - bellevue_df = bellevue_df.drop(columns="children”)

Sorting Columns

- .sort_values(by=’column_name’)
  - bellevue_df.sort_values(by='date_in', ascending=True)””)

Filter/Subset Data

- data_frame[‘column_name’] == ‘value’
  - Produces a True/False table based on condition
    - e.g. bellevue_df['profession'] == 'teacher’
- data_frame[data_frame[‘column_name’] == ‘value’
  - Filters out the rows from the original data frame that fits the condition
    - e.g. bellevue_df[bellevue_df['profession'] == 'teacher']