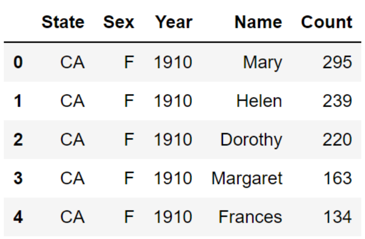
DS 100/200: Principles and Techniques of Data Science Date: Sep 18, 2019

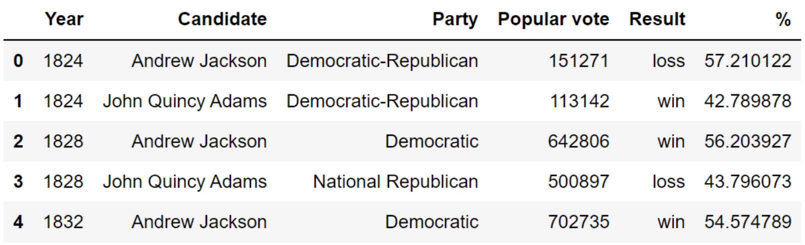
Discussion #4

Name:

Throughout this worksheet you'll be working with the babynames (left) and elections (right) datasets from lecture as shown below:



	State	Sex	Year	Name	Count
0	CA	F	1910	Mary	295
1	CA	F	1910	Helen	239
2	CA	F	1910	Dorothy	220
3	CA	F	1910	Margaret	163
4	CA	F	1910	Frances	134



	Year	Candidate	Party	Popular vote	Result	%
0	1824	Andrew Jackson	Democratic-Republican	151271	loss	57.210122
1	1824	John Quincy Adams	Democratic-Republican	113142	win	42.789878
2	1828	Andrew Jackson	Democratic	642806	win	56.203927
3	1828	John Quincy Adams	National Republican	500897	loss	43.796073
4	1832	Andrew Jackson	Democratic	702735	win	54.574789

Using `value_counts`

(a) Use `value_counts` to count the number of different baby names for each year in California. For example, `num_of_names_per_year.loc[2007]` should be 7,248.

`num_of_names_per_year =`

(b) Count the number of different names for each sex in the state 'CA'. The result should be a Series with 2 entries.

`num_of_names_per_sex =`

Using `groupby`

(a) Using `groupby.agg` or one of the shorthand methods (`groupby.min`, `groupby.first`, etc.), create a Series `best_result` that gives the highest percentage vote ever attained by each party. For example, `best_result['Libertarian']` should return 3.3. The order of your Series does not matter.

`best_result =`

(b) Again using `groupby.agg` or one of the its shorthand methods, create a DataFrame `last_result` that gives the result for a party in its most recent year of participation, with `Party` as its index. For example `last_result.query("Party == 'Whig'")` should give you a row showing that the Whigs last participated in an election in 1852 with Winfield Scott as their candidate, earning 44% of the vote. This might take more than one line of code. Write your answer below.

`last_result =`

(c) Using `filter`, create a DataFrame `major_party_results_since_1988` that includes all election results starting in 1988, but only include a row if the Party it belongs to has earned at least 1% of the popular vote in ANY election since 1988.

For example, in 1988, you should not include the `'New Alliance'` candidate since this party has not earned 1% of the vote since 1988. However, you should include the 'Libertarian' candidate from 1988 despite only having 0.47 percent of the vote in 1988 because in 2016 the Libertarian candidate Gary Johnson had 3.3% of the vote.

`major_party_results_since_1988 =`

(d) Create a Series `female_name_since_2000_count` which gives the total number of occurrences of each name for female babies born in California from the year 2000 or later. The index should be the name, and the value should be the total number of births. Your series should be ordered in decreasing order of count. For example, your first row should have index "Emily"and value 49605, because 49,605 Emilys have been born since the year 2000 in California.

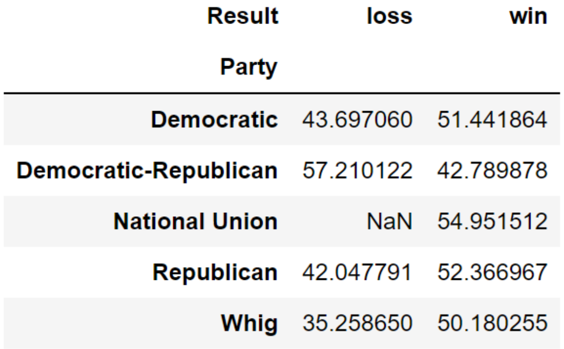
`female_name_since_2000_count =`

(e) Using `groupby`, create a Series `count_for_names_2018` listing all baby names from 2018 in decreasing order of popularity. The result should not be broken down by gender! If a name is used by both male and female babies, the number you provide should be the total across both genders. For example, `count_for_names_2018["Noah"]` should be the number 2567 because in 2018 there were 2567 Noahs born (12 female and 2555 male).

`count_for_names_2018 =`

Grouping Multiple Columns

(a) Which of the following lines of code will output the following dataframe? Recall that the arguments to `pd.pivot_table` are as follows: `data` is the input dataframe, `index` includes the values we use as rows, `columns` are the columns of the pivot table, `values` are the values in the pivot table, and `aggfunc` is the aggregation function that we use to aggregate `values`.



Result	loss	win
Party		
Democratic	43.697060	51.441864
Democratic-Republican	57.210122	42.789878
National Union	NaN	54.951512
Republican	42.047791	52.366967
Whig	35.258650	50.180255

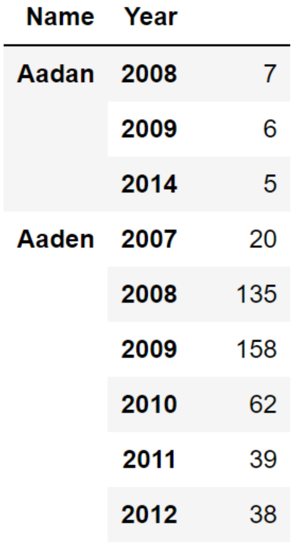
○ A. `pd.pivot_table(data=winners_only, index='Party', columns='Result', values='%', aggfunc=np.mean)`

○ B. `winners_only.groupby(['Party', 'Result'])['%'].mean()`

○ C. `pd.pivot_table(data=winners_only, index='Result', columns='Party', values='%', aggfunc=np.mean)`

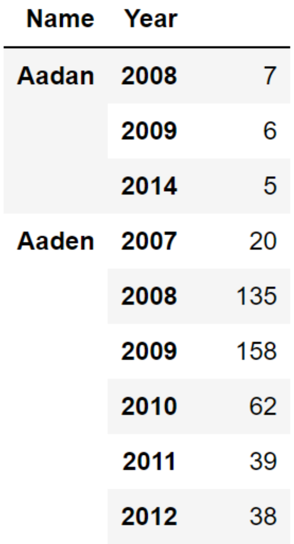
○ D. `winners_only.groupby('%')[['Party', 'Result']].mean()`

(b) `name_counts_since_1940 = babynames[babynames["Year"] >= 1940].groupby(["Name", "Year"]).sum()` generates the multi-indexed DataFrame below.



Name	Year	
Aadan	2008	7
	2009	6
	2014	5
Aaden	2007	20
	2008	135
	2009	158
	2010	62
	2011	39
	2012	38

We can index into multi-indexed DataFrames using `loc` with slightly different syntax. For example `name_counts_since_1940.loc[("Aahna", 2008):("Aaiden", 2014)]` yields the DataFrame below.



Name	Year	
Aadan	2008	7
	2009	6
	2014	5
Aaden	2007	20
	2008	135
	2009	158
	2010	62
	2011	39
	2012	38

Using `name_counts_since_1940`, set `imani_2013_count` equal to the number of babies born with the name `'Imani'` in the year 2013. You may use either '.loc'. or '.query'. Make sure you're returning a value and not a Series or DataFrame.

`imani_2013_count =`

String Operations and Table Joining

(a) Create a new DataFrame called `elections_with_first_name` with a new column `'First Name'` that is equal to the Candidate's first name. Hint: Use `.str.split`.

`elections_with_first_name =`

(b) Now create `elections_and_names` by joining `elections_with_first_name` with `name_counts_since_1940_numerical_index` (the modified version of `name_counts_since_1940` with the index reset) on both the first names of each person along and the year.

`elections_and_names =`