
0.1 Question 3c

In the cell below, run the following line of code: `q3c_df = cake_at_least_3.sort_values('timestamp').groupby('bid').`

Is the granularity of `cake_at_least_3` the same as the granularity of `q3c_df`? In other words, what does a single row of `q3c_df` represent, and what does a single row in `cake_at_least_3` represent? Explain the granularity of each `DataFrame`. Your answer does not need to be more than 2-3 lines, but you should be specific.

Note: For more details on what the granularity of a `DataFrame` means, feel free to check Section 5.2.1 in the course notes!

```
In [ ]: q3c_df = cake_at_least_3.sort_values('timestamp').groupby('bid').agg('first')
        q3c_df.head()
```

The granularity of `cake_at_least_3` is not the same as `q3c_df`. `cake_at_least_3` contains information from every single inspection for cake shops that had at least 3 total inspections in the whole dataset. Meanwhile, `q3c_df` only contains the data from the very first inspection of cake shops that have had at least 3 total inspections. The data in `q3c_df` is thus contained in `cake_at_least_3`, while `cake_at_least_3` contains additional data (the two or more additional inspections performed for the same cake shop)

0.2 Question 3e

Finally, to examine different parts of a chained `pandas` statement, describe the purpose of each of the functions used (`.loc`, `.groupby`, `idxmax()`) in words.

Secondly, share what you think this line of code accomplishes. In other words, write a question that could be answered using this statement.

While the first part of this question will be graded for correctness, the second part of this question is a bit more open-ended. Answers demonstrating your understanding will get full credit.

```
In [ ]: cake_at_least_3.loc[cake_at_least_3.groupby("bid")["score"].idxmax()].head()
```

0.3 Part 1

The `.loc` accessor allows us to select a subset of data from the `cake_at_least_3` dataframe. If only given one parameter, the `.loc` accessor will assume that that parameter corresponds to the labels of rows to select. The `.loc` accessor can take a single label, a list or array of labels, or a slice of labels. In this scenario, we will pass in a series of labels, which is essentially an array of labels, that corresponds to the labels of the rows (the index labels, not the actual numerical index) that we want.

The `.groupby` function takes in a column as a parameter by which to organize the dataframe. It returns a `GroupBy` object, which cannot be visualized like a normal dataframe, but must instead have some aggregate function be performed on each grouped set to return a dataframe or series. In this case, we grouped by the `bid` column (business ID) of the dataframe.

Next, we used the context-dependent accessor `[]`, which determines by context what subset of the data we wanted to retrieve (what rows, columns, or subset of rows/columnmns). Here, we passed in the label of a column with `["score"]`, indicating that we wanted only the inspection scores of for each group of business id's, and received a SeriesGroupBy object which contained only the scores for each group of business IDs.

We called `.idxmax()` as the aggregate function over the GroupBy object of scores of the business IDs, which gives us the index label of the maximum of the series it is called on. In this case, because we called it as the aggregating function for the GroupBy object of scores of business IDs, it returns a series of the corresponding index label in `cake_at_least_3` of the highest score corresponding to each group of business IDs.

Thus, a series of labels of indexes is passed to the `.loc[]` accessor, and is used to select from the `cake_at_least_3` dataframe a subset of cakeshops. Finally, `.head()` returns the first 5 rows of the new subset dataframe of `cake_at_least_3`.

0.4 Part 2

What this overall code is doing is finding the inspection with the highest score for each of the businesses with at least 3 total inspections. We group by bid, find the index label corresponding to the highest score for each bid, and then filter the `cake_at_least_3` dataframe to only show those inspections (the highest-scored inspection for each unique BID with at least 3 inspections).

In [ ]:
```
# You may do some scratch work in this cell, however, only your written answer will be graded.
# Any outputs or dataframes you generate here will not be counted as part of your explanation.
display(cake_at_least_3.groupby("bid"))
display(cake_at_least_3.groupby("bid")["score"])
display(cake_at_least_3.groupby("bid")["score"].idxmax())
# cake_at_least_3.drop_duplicates(subset="bid")
```