Reference Sheet

Pandas

In the entries below suppose `df` is a DataFrame, `s` is a Series and `pd` is the Pandas package

Attribute	Description
`df.index`	Returns the index labels of the DataFrame.
`df.columns`	Returns the column labels of the DataFrame.
`df.shape` or `s.shape`	Returns the shape of a DataFrame or Series in the form number of rows, number of columns
`df.size` or `s.size`	Returns the total number of entries in a DataFrame or Series (number of rows times number of columns)

Function	Description
`df[col]`	Returns the column labeled `col` from `df` as a Series.
`df[[col1, col2]]`	Returns a DataFrame containing the columns labeled `col1` and `col2`.
`s.loc[rows]` or `df.loc[rows, cols]`	Returns a Series/DataFrame with rows (and columns) selected by their index values.
`s.iloc[rows]` or `df.iloc[rows, cols]`	Returns a Series/DataFrame with rows (and columns) selected by their positions.
`s.isnull()` or `df.isnull()`	Returns boolean Series/DataFrame identifying missing values
`df.info()`	displays name and type of each column, the number of non-null entries, and size of dataframe
`s.fillna(value)` or `df.fillna(value)`	Returns a Series/DataFrame where missing values are replaced by `value`
`df.drop(labels, axis)`	Returns a DataFrame without the rows or columns named `labels` along `axis` (either 0 or 1)
`df.rename(index=None, columns=None)`	Returns a DataFrame with renamed columns from a dictionary `index` and/or `columns`
`df.sort_values(by, ascending=True)`	Returns a DataFrame where rows are sorted by the values in columns `by`
`s.sort_values(ascending=True)`	Returns a sorted Series.
`s.unique()`	Returns a NumPy array of the unique values
`s.value_counts()`	Returns the number of times each unique value appears in a Series
`pd.merge(left, right, how='inner', left_on='a', right_on='b')`	Returns a DataFrame joining DataFrames `left` and `right` on the columns labeled `a` in the left database and 'b' in the right database; the join is of type `inner`
`df.pivot_table(index, columns, values=None, aggfunc='mean')`	Returns a DataFrame pivot table where columns are unique values from `columns` (column name or list), and rows are unique values from `index` (column name or list); cells are collected `values` using `aggfunc`. If `values` is not provided, cells are collected for each remaining column with multi-level column indexing.
`df.set_index(col)`	Returns a DataFrame that uses the values in the column labeled `col` as the row index.
`df.reset_index()`	Returns a DataFrame that has row index 0, 1, etc., and adds the current index as a column.
`s.str.len()`	Returns a Series containing length of each string
`s.str.lower()` or `s.str.upper()`	Returns a Series containing lowercase/uppercase version of each string
`s.str.split(pat)`	Split strings around given separator/delimiter `pat`. If not specified, split on whitespace

Groupby

In the groupby entries below, `col` can be a column label or a list of column labels:

Function	Description
`df.groupby(col).count()`	Returns a Series/DataFrame with the counts of non-missing values in each column
`df.groupby(col).size()`	Returns a Series counting the number of rows in each group, including missing values
`df.groupby(col).mean()` or `df.groupby(col).min()` or `df.groupby(col).max()`	Returns a Series/DataFrame containing mean/min/max of each group for each column, excluding missing values
`df.groupby(col).first()` or `df.groupby(col).last()`	Returns a Series/DataFrame containing the first/last non-null entry of each group for each column
`df.groupby(col).agg(f)`	Returns a DataFrame with index `col`. Aggregates other columns using the given function f.

You are given a Pandas DataFrame `cereal` with **information per serving about** 80 **different breakfast cereals**. Here are the first 5 rows of the DataFrame:

	name	manufacturer	type	calories	protein	fat	fiber	carbo	sugars
0	100% Bran	Nabisco	cold	70	4	1	10.03	5.0	6
1	100% Natural Bran	Quaker Oats	cold	120	3	5	1.93	8.0	8
2	All-Bran	Kelloggs	cold	70	4	1	8.80	7.0	5
3	All-Bran with Extra Fiber	Kelloggs	cold	50	4	0	14.04	8.0	0
4	Almond Delight	Ralston Purina	cold	110	2	2	1.00	14.0	8

1. (5 pts) Fill in the blanks labeled below to add a new column to `cereal` named `low_calorie` which has the boolean value `True` if the cereal is low-calorie and `False` otherwise. Assume a cereal is low-calorie if the value of `calories` in the `cereal` DataFrame is less than or equal to 100.

   `cereal[___A___ ] = ___B___`

   Fill in blank A:

   `'low_calorie'`

   Fill in blank B:

   `cereal['calories'] <= 100`

2. (12 pts) Write code below to construct a DataFrame called `max_sugar` indexed by manufacturer, that has one column whose value is equal to the the maximum `sugars` value of all cereals by that manufacturer. Your DataFrame should be sorted by the max sugar value in **decreasing** order.

   For example, the first few entries of the DataFrame would be:

	sugars
Post	16
Kelloggs	15
Quaker Oats	14

   Write your code directly in the blanks provided below. **You can leave lines inside parantheses blank to represent a function call with no arguments.** You may not need all lines.

```
max_sugar =(cereal['sugars'].groupby('manufacturer')
           .max()
           .sort_values(Ascending=False)
           )
```

3. (3 pts)

   Suppose you are given a function named `test` that takes a DataFrame as its argument and returns a Boolean.

   What will be the output of `cereal.groupby("type").filter(test)` if the `test` function returns `True` for a group?

   - (✓) All rows in the group will be included in the result
   - ( ) All rows in the group will be excluded from the result
   - ( ) Only the first row in the group will be included in the result
   - ( ) Only the last row in the group will be included in the result
   - ( ) None of the above.

END OF QUIZ