The oldest businesses in the world

August 22, 2024

0.1 1. The oldest businesses in the world

This is Staffelter Hof Winery, Germany's oldest business, which was established in 862 under the Carolingian dynasty. It has continued to serve customers through dramatic changes in Europe such as the Holy Roman Empire, the Ottoman Empire, and both world wars. What characteristics enable a business to stand the test of time? Image credit: Martin Kraft

To help answer this question, BusinessFinancing.co.uk researched the oldest company that is still in business in almost every country and compiled the results into a dataset. Let's explore this work to to better understand these historic businesses. Our datasets, which are all located in the datasets directory, contain the following information:

businesses and new_businesses

column

type

meaning

business

varchar

Name of the business.

year_founded

int

Year the business was founded.

category_code

varchar

Code for the category of the business.

country_code

char

ISO 3166-1 3-letter country code.

countries

column

type

meaning

country_code

varchar

ISO 3166-1 3-letter country code.

country

varchar

Name of the country.

continent

varchar

Name of the continent that the country exists in.

categories

column

type

meaning

category_code

varchar

Code for the category of the business.

category

varchar

Description of the business category.

Now let's learn about some of the world's oldest businesses still in operation!

[114]:
```
# Import the pandas library under its usual alias
import pandas as pd

# Load the business.csv file as a DataFrame called businesses
businesses = pd.read_csv('datasets/businesses.csv')

# Sort businesses from oldest businesses to youngest
sorted_businesses = businesses.sort_values(by='year_founded', ascending=True)

# Display the first few lines of sorted_businesses
sorted_businesses.head()
```

[114]:

	business	year_founded	category_code	country_code
64	Kongō Gumi	578	CAT6	JPN

```
94   St. Peter Stifts Kulinarium          803         CAT4          AUT
107        Staffelter Hof Winery          862         CAT9          DEU
106            Monnaie de Paris           864        CAT12          FRA
103               The Royal Mint          886        CAT12          GBR
```

[115]:
```
%%nose

def test_pandas_loaded():
    assert "pd" in globals(), \
    "Did you correctly import the `pandas` library under the alias `pd`?"

import pandas as pd
test_businesses = pd.read_csv("datasets/businesses.csv")
test_sorted_businesses = test_businesses.sort_values("year_founded")

def test_bus():
    assert isinstance(businesses, pd.DataFrame), \
    "Did you create a `businesses` DataFrame using `pd.read_csv()`?"
    assert test_businesses.reset_index().equals(businesses.reset_index()), \
    "Your `businesses` DataFrame is not equal to the expected DataFrame. Did␣
 ↪you load your `businesses` DataFrame from `datasets/businesses.csv` using␣
 ↪`pd.read_csv()`?"

def test_sorted_bus():
    assert sorted_businesses.iloc[-1:].equals(test_sorted_businesses.iloc[-1:
 ↪]), \
    "Did you create `sorted_businesses` by sorting `year_founded` in␣
 ↪_ascending_ order?"
    assert test_sorted_businesses.reset_index().equals(sorted_businesses.
 ↪reset_index()), \
    "Your `sorted_businesses` DataFrame is not equal to the expected DataFrame.␣
 ↪Did you create it by calling `.sort_values()` on `businesses` and sorting by␣
 ↪`year_founded`?"
```

[115]:
```
3/3 tests passed
```

0.2 2. The oldest businesses in North America

So far we've learned that Kongō Gumi is the world's oldest continuously operating business, beating out the second oldest business by well over 100 years! It's a little hard to read the country codes, though. Wouldn't it be nice if we had a list of country names to go along with the country codes?

Enter countries.csv, which is also located in the datasets folder. Having useful information in different files is a common problem: for data storage, it's better to keep different types of data separate, but for analysis, we want all the data in one place. To solve this, we'll have to join the two tables together.

countries

column

type

meaning

country_code

varchar

ISO 3166-1 3-letter country code.

country

varchar

Name of the country.

continent

varchar

Name of the continent that the country exists in.

Since countries.csv contains a continent column, merging the datasets will also allow us to look at the oldest business on each continent!

[116]:
```
# Load countries.csv to a DataFrame
countries = pd.read_csv('datasets/countries.csv')

# Merge sorted_businesses with countries
businesses_countries = sorted_businesses.merge(countries,on='country_code')

# Filter businesses_countries to include countries in North America only
north_america = businesses_countries[businesses_countries["continent"] ==␣
 ↪"North America"]
north_america.head()
```

[116]:

	business	year_founded	category_code	country_code	country	continent
22	La Casa de Moneda de México	1534	CAT12	MEX	Mexico	North America
28	Shirley Plantation	1638	CAT1	USA	United States	North America
33	Hudson's Bay Company	1670	CAT17	CAN	Canada	North America
35	Mount Gay Rum	1703	CAT9	BRB	Barbados	North America
40	Rose Hall	1770	CAT19	JAM	Jamaica	North America

[117]:
```
%%nose

import pandas as pd
test_businesses = pd.read_csv("datasets/businesses.csv")
test_sorted_businesses = test_businesses.sort_values("year_founded")
test_countries = pd.read_csv("datasets/countries.csv")
test_businesses_countries = test_sorted_businesses.merge(test_countries, on=␣
 ↪"country_code")
test_north_america =␣
 ↪test_businesses_countries[test_businesses_countries["continent"]== "North␣
 ↪America"]

def test_cntries():
    assert isinstance(countries, pd.DataFrame), \
    "Did you create a `countries` DataFrame using `pd.read_csv()`?"
    assert test_countries.reset_index().equals(countries.reset_index()), \
    "Your `countries` DataFrame is not equal to the expected DataFrame."

def test_bus_countries():
    assert len(businesses_countries.columns) == len(test_businesses_countries.
 ↪columns), \
    "Your `businesses_countries` DataFrame should have six columns: `business`,␣
 ↪`year_founded`, `category_code`, `country_code`, `country`, and `continent`."
    assert test_businesses_countries.reset_index().equals(businesses_countries.
 ↪reset_index()), \
    "Your `businesses_countries` DataFrame is not equal to the expected␣
 ↪DataFrame."

def test_north_am():
    assert north_america.shape[0] == test_north_america.shape[0], \
    "Did you filter `businesses_countries` to include only countries in North␣
 ↪America? It looks like your `north_america` DataFrame has a different number␣
 ↪of rows than expected."
    assert test_north_america.reset_index().equals(north_america.
 ↪reset_index()), \
    "Your `north_america` DataFrame is not equal to the expected DataFrame."
```

[117]:
```
3/3 tests passed
```

0.3 3. The oldest business on each continent

Now we can see that the oldest company in North America is La Casa de Moneda de México, founded in 1534. Why stop there, though, when we could easily find out the oldest business on every continent?

[118]:
```
# Create continent, which lists only the continent and oldest year_founded
continent = businesses_countries.groupby("continent").agg({"year_founded":
 ↪"min"})

# Merge continent with businesses_countries
merged_continent = continent.merge(businesses_countries, on=["continent",␣
 ↪"year_founded"])

# Subset continent so that only the four columns of interest are included
subset_merged_continent =␣
 ↪merged_continent[['continent','country','business','year_founded']]
subset_merged_continent
```

[118]:

	continent	country	business	year_founded
0	Africa	Mauritius	Mauritius Post	1772
1	Asia	Japan	Kongō Gumi	578
2	Europe	Austria	St. Peter Stifts Kulinarium	803
3	North America	Mexico	La Casa de Moneda de México	1534
4	Oceania	Australia	Australia Post	1809
5	South America	Peru	Casa Nacional de Moneda	1565

[119]:
```
%%nose

import pandas as pd
test_businesses = pd.read_csv("datasets/businesses.csv")
test_countries = pd.read_csv("datasets/countries.csv")
test_businesses_countries = test_businesses.merge(test_countries,␣
 ↪on="country_code")
test_continent = test_businesses_countries.groupby("continent").
 ↪agg({"year_founded":"min"})
test_merged_continent = test_continent.merge(test_businesses_countries,␣
 ↪on=["continent", "year_founded"])
test_subset_merged_continent = test_merged_continent[["continent", "country",␣
 ↪"business", "year_founded"]]

def test_cont():
    assert isinstance(continent, pd.DataFrame), \
    "Your `continent` DataFrame needs to be a DataFrame and not a Series. The␣
 ↪index of the DataFrame will be continent names. You can use the .agg()␣
 ↪function to do this."
    assert len(continent.columns) == len(test_continent.columns), \
    "Your `continent` DataFrame should have `continent` as its index and should␣
 ↪have a single column called `year_founded`."
    assert continent.shape[0] == test_continent.shape[0], \
    "Did you group `businesses_countries` by `continent`? It looks like your␣
 ↪`continent` DataFrame has a different number of rows than expected."
```

```
def test_subset_merged_cont():
    assert test_subset_merged_continent.columns.all() ==␣
 ↪subset_merged_continent.columns.all(), \
    "Did you correctly subset the `continent` DataFrame by selecting the␣
 ↪columns of interest (in order)?"
    assert subset_merged_continent.shape[0] == test_subset_merged_continent.
 ↪shape[0], \
    "Did you subset the `merged_continent` DataFrame _columns_? The number of␣
 ↪rows in `subset_merged_continent` and `merged_continent` should be the same."
```

[119]: 2/2 tests passed

0.4 4. Unknown oldest businesses

BusinessFinancing.co.uk wasn't able to determine the oldest business for some countries, and those countries are simply left off of businesses.csv and, by extension, businesses. However, the countries that we created does include all countries in the world, regardless of whether the oldest business is known.

We can compare the two datasets in one DataFrame to find out which countries don't have a known oldest business!

[120]:
```
# Use .merge() to create a DataFrame, all_countries
all_countries = businesses.merge(countries, on="country_code", how="right", ␣
 ↪indicator=True)

# Filter to include only countries without oldest businesses
missing_countries = all_countries[all_countries["_merge"] != "both"]

# Create a series of the country names with missing oldest business data
missing_countries_series = missing_countries["country"]

# Display the series
missing_countries_series
```

[120]:
```
163                              Angola
164                 Antigua and Barbuda
165                             Bahamas
166                  Dominican Republic
167                             Ecuador
168                                Fiji
169    Micronesia, Federated States of
170                               Ghana
171                              Gambia
```

```
172                              Grenada
173          Iran, Islamic Republic of
174                            Kyrgyzstan
175                              Kiribati
176                Saint Kitts and Nevis
177                               Monaco
178                Moldova, Republic of
179                            Maldives
180                    Marshall Islands
181                               Nauru
182                               Palau
183                    Papua New Guinea
184                            Paraguay
185                  Palestine, State of
186                      Solomon Islands
187                             Suriname
188                           Tajikistan
189                         Turkmenistan
190                          Timor-Leste
191                               Tonga
192                              Tuvalu
193    Saint Vincent and the Grenadines
194                               Samoa
Name: country, dtype: object
```

[121]:
```
%%nose
import pandas as pd
output = _
test_businesses = pd.read_csv("datasets/businesses.csv")

test_countries = pd.read_csv("datasets/countries.csv")
test_all_countries = test_businesses.merge(test_countries, on="country_code",␣
 ↪how="right",  indicator = True)
test_missing_countries = test_all_countries[test_all_countries["_merge"] !=␣
 ↪"both"]
test_missing_countries_series = test_missing_countries["country"]

def test_all():
    assert {'business', 'year_founded', 'category_code', 'country_code',␣
 ↪'country', 'continent'}.issubset(all_countries.columns), \
    "Your `all_countries` DataFrame should include the following columns:␣
 ↪`business`, `year_founded`, `category_code`, `country_code`, `country`,␣
 ↪`continent`, and `_merge`. Did you create the DataFrame using an outer merge␣
 ↪with `indicator = True`?"
    assert all_countries.shape[0] == test_all_countries.shape[0], \
    "Did you use an outer merge to create `all_countries`? It looks like your␣
 ↪`all_countries` DataFrame has a different number of rows than expected."
```

```
def test_missing():
    assert missing_countries.shape[0] == test_missing_countries.shape[0], \
    "Did you filter `all_countries` to include only countries that _don't_ have␣
 ↪a 'both' value in the `_merge` column? It looks like your␣
 ↪`missing_countries` DataFrame has a different number of rows than expected."

def test_series():
    assert isinstance(missing_countries_series, pd.Series), \
    "Are you sure your `missing_countries_series` is a _Series_ and not a␣
 ↪DataFrame?"
    assert len(missing_countries_series) == len(test_missing_countries), \
    "The number of missing countries is different than expected. It should not␣
 ↪change between the `missing_countries` DataFrame and␣
 ↪`missing_countries_series`."
    assert test_missing_countries_series.equals(missing_countries_series), \
    "Your `missing_countries_series` is not equal to the expected series."

def test_display_series():
    assert test_missing_countries_series.equals(_), \
    "Did you display `missing_countries_series` as cell output?"
```

[121]: 4/4 tests passed

0.5 5. Adding new oldest business data

It looks like we've got some holes in our dataset! Fortunately, we've taken it upon ourselves to improve upon BusinessFinancing.co.uk's work and find oldest businesses in a few of the missing countries. We've stored the newfound oldest businesses in new_businesses, located at "datasets/new_businesses.csv". It has the exact same structure as our businesses dataset.

new_businesses

column

type

meaning

business

varchar

Name of the business.

year_founded

int

Year the business was founded.

category_code

varchar

Code for the category of the business.

country_code

char

ISO 3166-1 3-letter country code.

All we have to do is combine the two so that we've got one more complete list of businesses!

[122]:
```
# Import new_businesses.csv
new_businesses = pd.read_csv("datasets/new_businesses.csv")

# Add the data in new_businesses to the existing businesses
all_businesses = pd.concat([new_businesses, businesses])

# Merge and filter to find countries with missing business data
new_all_countries = all_businesses.merge(countries, on="country_code",␣
 ↪how="outer",  indicator=True)
new_missing_countries = new_all_countries[new_all_countries["_merge"] != "both"]

# Group by continent and create a "count_missing" column
count_missing = new_missing_countries.groupby("continent").agg({"country":
 ↪"count"})
count_missing.columns = ["count_missing"]
count_missing
```

[122]:

continent	count_missing
Africa	3
Asia	7
Europe	2
North America	5
Oceania	10
South America	3

[123]:
```
%%nose
import pandas as pd
test_businesses = pd.read_csv("datasets/businesses.csv")
test_countries = pd.read_csv("datasets/countries.csv")
test_new_businesses = pd.read_csv("datasets/new_businesses.csv")
test_all_businesses = pd.concat([test_new_businesses, test_businesses])
test_new_all_countries = test_all_businesses.merge(test_countries,␣
 ↪on="country_code", how="outer",  indicator = True)
test_new_missing_countries =␣
 ↪test_new_all_countries[test_new_all_countries["_merge"] != "both"]
test_count_missing = test_new_missing_countries.groupby("continent").
 ↪agg({"country":"count"})
```

```
test_count_missing.columns = ["count_missing"]

def test_import():
    assert isinstance(new_businesses, pd.DataFrame), \
    "Did you create a `new_businesses` DataFrame using `pd.read_csv()`?"

def test_all_bus():
    assert all_businesses.shape[0] == test_all_businesses.shape[0], \
    "Did you use `pd.concat()` to create `all_businesses` from `new_businesses`␣
 ↪and `businesses`? It looks like your `all_businesses` DataFrame has a␣
 ↪different number of rows than expected."
    assert all_businesses.shape[1] == test_all_businesses.shape[1], \
    "Did you create `all_businesses` from `new_businesses` and `businesses` by␣
 ↪stacking them _vertically_? It looks like your `all_businesses` DataFrame␣
 ↪has a different number of columns than expected."
    assert test_all_businesses.reset_index().equals(all_businesses.
 ↪reset_index()), \
    "Your `all_businesses` DataFrame is not equal to the expected DataFrame.␣
 ↪Did you use `pd.concat()` to stack `new_businesses` and `businesses`␣
 ↪vertically?"

def test_new_all_cntries():
    assert {'business', 'year_founded', 'category_code', 'country_code',␣
 ↪'country', 'continent'}.issubset(new_all_countries.columns), \
    "Your `new_all_countries` DataFrame should include the following columns:␣
 ↪`business`, `year_founded`, `category_code`, `country_code`, `country`,␣
 ↪`continent`, and `_merge`. Did you create the DataFrame using an outer merge␣
 ↪with `indicator = True`?"
    assert new_all_countries.shape[0] == test_new_all_countries.shape[0], \
    "Did you use a right or outer merge to create `new_all_countries`? It looks␣
 ↪like your `new_all_countries` DataFrame has a different number of rows than␣
 ↪expected."

def test_new_missing_cntries():
    assert new_missing_countries.shape[0] == test_new_missing_countries.
 ↪shape[0], \
    "Did you filter `new_all_countries` to include only countries that _don't_␣
 ↪have a 'both' value in the `_merge` column? It looks like your␣
 ↪`new_missing_countries` DataFrame has a different number of rows than␣
 ↪expected."

def test_count_miss():
    assert isinstance(count_missing, pd.DataFrame), \
    "Your `count_missing` DataFrame needs to be a DataFrame and not a Series.␣
 ↪The index will be continent names. You can use the .agg() function to do␣
 ↪this."
```

```
    assert count_missing.shape[0] == test_count_missing.shape[0], \
    "Did you create `count_missing` by grouping `new_missing_countries` by␣
↪continent? It looks like your `count_missing` DataFrame has a different␣
↪number of rows than expected."
    try:
        assert count_missing.columns == ["count_missing"]
    except AssertionError:
        assert False, "Does `count_missing` have only one column? Did you␣
↪forget to rename the column to `count_missing`?"
    assert test_count_missing.equals(count_missing), \
    "Your `count_missing` DataFrame is not equal to the expected DataFrame. Did␣
↪you aggregate the data by _counting_ the number of countries missing from␣
↪each continent group?"
```

[123]: 5/5 tests passed

0.6 6. The oldest industries

Remember our oldest business in the world, Kongō Gumi?

business

year_founded

category_code

country_code

64

Kongō Gumi

578

CAT6

JPN

We know Kongō Gumi was founded in the year 578 in Japan, but it's a little hard to decipher which industry it's in. Information about what the category_code column refers to is in "datasets/categories.csv":

categories

column

type

meaning

category_code

varchar

Code for the category of the business.

category

varchar

Description of the business category.

Let's use categories.csv to understand how many oldest businesses are in each category of industry.

[124]:
```
# Import categories.csv and merge to businesses
categories = pd.read_csv("datasets/categories.csv")
businesses_categories = businesses.merge(categories, on="category_code")

# Create a DataFrame which lists the number of oldest businesses in each␣
 ↪category
count_business_cats = businesses_categories.groupby("category").agg({"business":
 ↪"count"})

# Rename column and display the first five rows of the DataFrame
count_business_cats.columns = ["count"]
display(count_business_cats.head())
```

category	count
Agriculture	6
Aviation & Transport	19
Banking & Finance	37
Cafés, Restaurants & Bars	6
Conglomerate	3

[125]:
```
%%nose

import pandas as pd
test_businesses = pd.read_csv("datasets/businesses.csv")
test_categories = pd.read_csv("datasets/categories.csv")
test_businesses_categories = test_businesses.merge(test_categories,␣
 ↪on="category_code")
test_count_business_cats = test_businesses_categories.groupby("category").
 ↪agg({"business":"count"})
test_count_business_cats.columns = ["count"]

def test_import():
    assert isinstance(categories, pd.DataFrame), \
    "Did you create a `categories` DataFrame using `pd.read_csv()`?"

def test_count_bus_cats():
    assert isinstance(count_business_cats, pd.DataFrame), \
    "Your `count_business_cats` DataFrame needs to be a DataFrame and not a␣
 ↪Series. The index will be category names. You can use the .agg() function to␣
 ↪do this."
```

```
    assert count_business_cats.shape[0] == test_count_business_cats.shape[0], \
    "Did you group `count_business_cats` by `category`? It looks like your␣
↪`count_business_cats` DataFrame has a different number of rows than expected.
↪"
    assert test_count_business_cats.index.all() == count_business_cats.index.
↪all(), \
    "Did you group `count_business_cats` by `category`? It looks like your␣
↪`count_business_cats` DataFrame indexes aren't the business category names."
    assert test_count_business_cats.columns.all() == count_business_cats.
↪columns.all(), \
    "Did you rename the column in `count_business_cats`?"
    assert test_count_business_cats.equals(count_business_cats), \
    "Did you aggregate the `business` column using `count`?"
```

[125]:
```
2/2 tests passed
```

0.7 7. Restaurant representation

No matter how we measure it, looks like Banking and Finance is an excellent industry to be in if longevity is our goal! Let's zoom in on another industry: cafés, restaurants, and bars. Which restaurants in our dataset have been around since before the year 1800?

[126]:
```
# Filter using .query() for CAT4 businesses founded before 1800; sort results
old_restaurants = businesses_categories.query('year_founded < 1800 and␣
 ↪category_code == "CAT4"')

# Sort the DataFrame
old_restaurants = old_restaurants.sort_values("year_founded")
old_restaurants
```

[126]:

	business	year_founded	category_code	country_code	category
142	St. Peter Stifts Kulinarium	803	CAT4	AUT	Cafés, Restaurants & Bars
143	Sean's Bar	900	CAT4	IRL	Cafés, Restaurants & Bars
139	Ma Yu Ching's Bucket Chicken House	1153	CAT4	CHN	Cafés, Restaurants & Bars

[127]:
```
%%nose

import pandas as pd
test_businesses = pd.read_csv("datasets/businesses.csv")
test_categories = pd.read_csv("datasets/categories.csv")
```

```
test_businesses_categories = test_businesses.merge(test_categories,␣
 ↪on="category_code")
test_old_restaurants = test_businesses_categories.query('year_founded < 1800␣
 ↪and category_code == "CAT4"').sort_values("year_founded")

def test_old_rests():
    assert old_restaurants.shape[0] == test_old_restaurants.shape[0], \
    "Did you filter using `.query()` where `year_founded` is less than 1800 and␣
 ↪`category_code` is 'CAT4'? It looks like your `old_restaurants` DataFrame␣
 ↪has a different number of rows than expected."
    assert old_restaurants.iloc[-1:].equals(test_old_restaurants.iloc[-1:]), \
    "Did you sort `old_restaurants` by `year_founded` in _ascending_ order?"
    assert test_old_restaurants.reset_index().equals(old_restaurants.
 ↪reset_index()), \
    "Your `old_restaurants` DataFrame is not equal to the expected DataFrame.␣
 ↪Did you filter using `.query()` where `year_founded` is less than 1800 and␣
 ↪`category_code` is 'CAT4'? Did you sort from oldest to newest?"
```

[127]:
```
1/1 tests passed
```

0.8 8. Categories and continents

St. Peter Stifts Kulinarium is old enough that the restaurant is believed to have served Mozart - and it would have been over 900 years old even when he was a patron! Let's finish by looking at the oldest business in each category of commerce for each continent.

[128]:
```
# Merge all businesses, countries, and categories together
businesses_categories_countries = businesses_categories.merge(countries,␣
 ↪on="country_code")

# Sort businesses_categories_countries from oldest to most recent
businesses_categories_countries = businesses_categories_countries.
 ↪sort_values("year_founded")

# Create the oldest by continent and category DataFrame
oldest_by_continent_category = businesses_categories_countries.
 ↪groupby(["continent", "category"]).agg({"year_founded":"min"})
oldest_by_continent_category.head()
```

[128]:

continent	category	year_founded
Africa	Agriculture	1947
	Aviation & Transport	1854
	Banking & Finance	1892
	Distillers, Vintners, & Breweries	1933
	Energy	1968

[129]:
```
%%nose
import pandas as pd
test_businesses = pd.read_csv("datasets/businesses.csv")
test_categories = pd.read_csv("datasets/categories.csv")
test_countries = pd.read_csv("datasets/countries.csv")
test_businesses_categories = test_businesses.merge(test_categories,␣
 ↪on="category_code")
test_businesses_categories_countries = test_businesses_categories.
 ↪merge(test_countries, on="country_code").sort_values("year_founded")
test_oldest_by_continent_category = test_businesses_categories_countries.
 ↪groupby(["continent", "category"]).agg({"year_founded":"min"})

def test_bus_cat_countries():
    assert len(businesses_categories_countries.columns) ==␣
 ↪len(businesses_categories_countries.columns), \
    "Your `businesses_categories_countries` DataFrame should have seven columns:
 ↪ `business`, `year_founded`, `category_code`, `country_code`, `category`,␣
 ↪`country`, and `continent`."
    assert businesses_categories_countries.shape[0] ==␣
 ↪test_businesses_categories_countries.shape[0], \
    "It looks like your `businesses_categories_countries` DataFrame has a␣
 ↪different number of rows than expected."
    assert test_businesses_categories_countries.iloc[-1:].
 ↪equals(businesses_categories_countries.iloc[-1:]),\
    "Did you sort `businesses_categories_countries` by `year_founded`?"

def test_grouped():
    assert isinstance(oldest_by_continent_category, pd.DataFrame), \
    "Your `oldest_by_continent_category` DataFrame needs to be a DataFrame and␣
 ↪not a Series. The index will be continent and category names since it is␣
 ↪grouped by both. You can use the .agg() function to do this."
    assert test_oldest_by_continent_category.index.get_level_values("category").
 ↪all() == oldest_by_continent_category.sort_index().index.
 ↪get_level_values("category").all(),\
    "Did you group `oldest_by_continent_category` by `continent` and `category`?
 ↪"

def test_values():
    assert test_oldest_by_continent_category.
 ↪equals(oldest_by_continent_category),\
    "Your `old_restaurants` DataFrame is not equal to the expected DataFrame.␣
 ↪Did you aggregate the `year_founded` column by finding its minimum?"
```

[129]:
```
3/3 tests passed
```