1. The oldest business in the world

Image: St. Peter Stiftskeller, founded 803. Credit: Pakeha (https://commons.wikimedia.org/wiki/File:Eingang_zum_St._Peter_Stiftskeller.jpg).

An important part of business is planning for the future and ensuring that the company survives changing market conditions. Some businesses do this really well and last for hundreds of years.

BusinessFinancing.co.uk researched (https://businessfinancing.co.uk/the-oldest-company-in-almost-every-country) the oldest company that is still in business in (almost) every country and compiled the results into a dataset. In this project, you'll explore that dataset to see what they found.

The database contains three tables.

`categories`

column	type	meaning

column	type	meaning
category_code	varchar	Code for the category of the business.
category	varchar	Description of the business category.

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.

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.

In [2]:
```
%%sql
postgresql:///oldestbusinesses

-- Select the oldest and newest founding years from the businesses table
SELECT MIN(year_founded),MAX(year_founded)
FROM businesses;
```

```
1 rows affected.
```

Out[2]:

min	max
578	1999

In [3]:
```
%%nose
last_output = _

def test_resultset():
    try:
        assert str(type(last_output)) == "<class 'sql.run.ResultSet'>"
    except AssertionError:
        assert False, "Please ensure a SQL ResultSet is the output of the code
cell."
results = last_output.DataFrame()
def test_shape():
    try:
        assert results.shape == (1, 2)
    except AssertionError:
        assert False, "The results should have two columns and a single row."
def test_colnames():
    try:
        assert results.columns.tolist() == ['min', 'max']
    except AssertionError:
        assert False, "The results should have columns named `'min` and `'ma
x'`."
def test_min_year_founded():
    try:
        assert results.loc[0, 'min'] == 578
    except AssertionError:
        assert False, "The oldest year founded is incorrect."
def test_max_year_founded():
    try:
        assert results.loc[0, 'max'] == 1999
    except AssertionError:
        assert False, "The newest year founded is incorrect."
```

Out[3]:
```
5/5 tests passed
```

2. How many businesses were founded before 1000?

Wow! That's a lot of variation between countries. In one country, the oldest business was only founded in 1999. By contrast, the oldest business in the world was founded back in 578. That's pretty incredible that a business has survived for more than a millennium.

I wonder how many other businesses there are like that.

In [4]:
```
%%sql

-- Get the count of rows in businesses where the founding year was before 1000
SELECT COUNT(business)
FROM businesses
WHERE year_founded < 1000;
```

```
 * postgresql:///oldestbusinesses
1 rows affected.
```

Out[4]:

count
6

In [5]:
```
%%nose
last_output = _

def test_resultset():
    try:
        assert str(type(last_output)) == "<class 'sql.run.ResultSet'>"
    except AssertionError:
        assert False, "Please ensure a SQL ResultSet is the output of the code
cell."
results = last_output.DataFrame()
def test_shape():
    try:
        assert results.shape == (1, 1)
    except AssertionError:
        assert False, "The results should have a single column and a single ro
w."
def test_colnames():
    try:
        assert results.columns.tolist() == ['count']
    except AssertionError:
        assert False, "The results should have a column named `'count`."
def test_count():
    try:
        assert last_output.DataFrame().loc[0, 'count'] == 6
    except AssertionError:
        assert False, "The count of businesses founded before 1000 is incorrec
t."
```

Out[5]:
```
4/4 tests passed
```

3. Which businesses were founded before 1000?

Having a count is all very well, but I'd like more detail. Which businesses have been around for more than a millennium?

In [6]:
```
%%sql

-- Select all columns from businesses where the founding year was before 1000
-- Arrange the results from oldest to newest
SELECT *
FROM businesses
WHERE year_founded < 1000
ORDER BY year_founded;
```

```
 * postgresql:///oldestbusinesses
6 rows affected.
```

Out[6]:

business	year_founded	category_code	country_code
Kongō Gumi	578	CAT6	JPN
St. Peter Stifts Kulinarium	803	CAT4	AUT
Staffelter Hof Winery	862	CAT9	DEU
Monnaie de Paris	864	CAT12	FRA
The Royal Mint	886	CAT12	GBR
Sean's Bar	900	CAT4	IRL

In [7]:
```
%%nose
last_output = _

def test_resultset():
    try:
        assert str(type(last_output)) == "<class 'sql.run.ResultSet'>"
    except AssertionError:
        assert False, "Please ensure a SQL ResultSet is the output of the code
cell."
results = last_output.DataFrame()
def test_shape():
    try:
        assert results.shape == (6, 4)
    except AssertionError:
        assert False, "The results should have four columns and six rows."
def test_colnames():
    try:
        assert results.columns.tolist() == ['business', 'year_founded', 'categ
ory_code', 'country_code']
    except AssertionError:
        assert False, "The results should have the four columns from the `busi
nesses` table."
def test_where_year_founded_lt_1000():
    try:
        assert results.loc[:, 'year_founded'].max() < 1000
    except AssertionError:
        assert False, "The most recent year founded is not before 1000."
def test_ordered_by_year_founded():
    try:
        assert results.loc[:, 'year_founded'].is_monotonic
    except AssertionError:
        assert False, "The rows are not ordered by increasing year founded."
```

Out[7]:
```
5/5 tests passed
```

4. Exploring the categories

Now we know that the oldest, continuously operating company in the world is called Kongō Gumi. But was does that company do? The category codes in the `businesses` table aren't very helpful: the descriptions of the categories are stored in the `categories` table.

This is a common problem: for data storage, it's better to keep different types of data in different tables, but for analysis, you want all the data in one place. To solve this, you'll have to join the two tables together.

In [8]:
```
%%sql

-- Select business name, founding year, and country code from businesses; and
 category from categories
-- where the founding year was before 1000, arranged from oldest to newest
SELECT bus.business, bus.year_founded, bus.country_code, cat.category
    FROM businesses AS bus
    INNER JOIN categories AS cat
        ON bus.category_code = cat.category_code
    WHERE year_founded < 1000
    ORDER BY year_founded;
```

```
 * postgresql:///oldestbusinesses
6 rows affected.
```

Out[8]:

business	year_founded	country_code	category
Kongō Gumi	578	JPN	Construction
St. Peter Stifts Kulinarium	803	AUT	Cafés, Restaurants & Bars
Staffelter Hof Winery	862	DEU	Distillers, Vintners, & Breweries
Monnaie de Paris	864	FRA	Manufacturing & Production
The Royal Mint	886	GBR	Manufacturing & Production
Sean's Bar	900	IRL	Cafés, Restaurants & Bars

In [9]:
```
%%nose
last_output = _

def test_resultset():
    try:
        assert str(type(last_output)) == "<class 'sql.run.ResultSet'>"
    except AssertionError:
        assert False, "Please ensure a SQL ResultSet is the output of the code
cell."
results = last_output.DataFrame()
def test_shape():
    try:
        assert results.shape == (6, 4)
    except AssertionError:
        assert False, "The results should have four columns and six rows."
def test_colnames():
    try:
        assert results.columns.tolist() == ['business', 'year_founded', 'count
ry_code', 'category']
    except AssertionError:
        assert False, "The results should have business, year founded, and cou
ntry code columns from the `businesses` table and category from the `categorie
s` table."
def test_where_year_founded_lt_1000():
    try:
        assert results.loc[:, 'year_founded'].max() < 1000
    except AssertionError:
        assert False, "The most recent year founded is not before 1000."
def test_ordered_by_year_founded():
    try:
        assert results.loc[:, 'year_founded'].is_monotonic
    except AssertionError:
        assert False, "The rows are not ordered by increasing year founded."
```

Out[9]:
```
5/5 tests passed
```

5. Counting the categories

With that extra detail about the oldest businesses, we can see that Kongō Gumi is a construction company. In that list of six businesses, we also see a café, a winery, and a bar. The two companies recorded as "Manufacturing and Production" are both mints. That is, they produce currency.

I'm curious as to what other industries constitute the oldest companies around the world, and which industries are most common.

In [10]:
```
%%sql

-- Select the category and count of category (as "n")
-- arranged by descending count, limited to 10 most common categories
SELECT cat.category, COUNT(cat.category) AS n
    FROM businesses AS bus
    INNER JOIN categories AS cat
        ON bus.category_code = cat.category_code
    GROUP BY cat.category
    ORDER BY n DESC
    LIMIT 10;
```

```
 * postgresql:///oldestbusinesses
10 rows affected.
```

Out[10]:

category	n
Banking & Finance	37
Distillers, Vintners, & Breweries	22
Aviation & Transport	19
Postal Service	16
Manufacturing & Production	15
Media	7
Agriculture	6
Cafés, Restaurants & Bars	6
Food & Beverages	6
Tourism & Hotels	4

In [11]:
```
%%nose
last_output = _

def test_resultset():
    try:
        assert str(type(last_output)) == "<class 'sql.run.ResultSet'>"
    except AssertionError:
        assert False, "Please ensure a SQL ResultSet is the output of the code
cell."
results = last_output.DataFrame()
def test_shape():
    try:
        assert results.shape == (10, 2)
    except AssertionError:
        assert False, "The results should have two columns and ten rows."
def test_colnames():
    try:
        assert results.columns.tolist() == ['category', 'n']
    except AssertionError:
        assert False, "The results should have a category column and a count c
olumn named `'n'`."
def test_ordered_by_desc_n():
    try:
        assert results.loc[:, 'n'].is_monotonic_decreasing
    except AssertionError:
        assert False, "The rows are not ordered by descending count."
def test_count():
    try:
        assert results.loc[:, 'n'].values.tolist() == [37, 22, 19, 16, 15, 7,
 6, 6, 6, 4]
    except AssertionError:
        assert False, "The category counts are not correct."
```

Out[11]:
```
5/5 tests passed
```

6. Oldest business by continent

It looks like "Banking & Finance" is the most popular category. Maybe that's where you should aim if you want to start a thousand-year business.

One thing we haven't looked at yet is where in the world these really old businesses are. To answer these questions, we'll need to join the `businesses` table to the `countries` table. Let's start by asking how old the oldest business is on each continent.

In [12]:
```
%%sql

-- Select the oldest founding year (as "oldest") from businesses,
-- and continent from countries
-- for each continent, ordered from oldest to newest
SELECT MIN(b.year_founded) as oldest, a.continent
    FROM businesses as b
    INNER JOIN countries AS a
        ON b.country_code = a.country_code
    GROUP BY continent
    ORDER BY oldest;
```

```
 * postgresql:///oldestbusinesses
6 rows affected.
```

Out[12]:

oldest	continent
578	Asia
803	Europe
1534	North America
1565	South America
1772	Africa
1809	Oceania

In [13]:
```
%%nose
last_output = _

def test_resultset():
    try:
        assert str(type(last_output)) == "<class 'sql.run.ResultSet'>"
    except AssertionError:
        assert False, "Please ensure a SQL ResultSet is the output of the code
cell."
results = last_output.DataFrame()
def test_shape():
    try:
        assert results.shape == (6, 2)
    except AssertionError:
        assert False, "The results should have two columns and six rows."
def test_colnames():
    try:
        assert results.columns.tolist() == ['oldest', 'continent']
    except AssertionError:
        assert False, "The results should have columns named oldest, and conti
nent."
def test_ordered_by_min_year_founded():
    try:
        assert results.loc[:, 'oldest'].is_monotonic
    except AssertionError:
        assert False, "The rows are not ordered by year founded."
def test_count():
    try:
        assert results.loc[:, 'oldest'].values.tolist() == [578, 803, 1534, 15
65, 1772, 1809]
    except AssertionError:
        assert False, "The year founded values are not correct."
```

Out[13]:
```
5/5 tests passed
```

7. Joining everything for further analysis

Interesting. There's a jump in time from the older businesses in Asia and Europe to the 16th Century oldest businesses in North and South America, then to the 18th and 19th Century oldest businesses in Africa and Oceania.

As mentioned earlier, when analyzing data it's often really helpful to have all the tables you want access to joined together into a single set of results that can be analyzed further. Here, that means we need to join all three tables.

In [14]:
```
%%sql

-- Select the business, founding year, category, country, and continent
SELECT b.business, b.year_founded, cat.category, c.country,c.continent
    FROM businesses AS b
    INNER JOIN categories AS cat
        ON b.category_code = cat.category_code
    INNER JOIN countries AS c
        ON b.country_code = c.country_code;
```

```
 * postgresql:///oldestbusinesses
163 rows affected.
```

Out[14]:

business	year_founded	category	country	continent
Hamoud Boualem	1878	Food & Beverages	Algeria	Africa
Communauté Électrique du Bénin	1968	Energy	Benin	Africa
Botswana Meat Commission	1965	Agriculture	Botswana	Africa
Air Burkina	1967	Aviation & Transport	Burkina Faso	Africa
Brarudi	1955	Distillers, Vintners, & Breweries	Burundi	Africa
Cameroon Development Corporation	1947	Agriculture	Cameroon	Africa
Correios de Cabo Verde	1849	Postal Service	Cabo Verde	Africa
Banque Internationale pour la Centrafrique	1946	Banking & Finance	Central African Republic	Africa
Cotontchad	1971	Agriculture	Chad	Africa
Central Bank of the Comoros	1981	Banking & Finance	Comoros	Africa
Société nationale des Chemins de fer du Congo	1889	Aviation & Transport	Congo, The Democratic Republic of the	Africa
Development Bank of the Central African States	1975	Banking & Finance	Congo	Africa
La Poste	1945	Postal Service	Côte d'Ivoire	Africa
Ethio-Djibouti Railways	1901	Aviation & Transport	Djibouti	Africa
Egyptian National Railways	1854	Aviation & Transport	Egypt	Africa
Guinea Ecuatorial Airlines	1996	Aviation & Transport	Equatorial Guinea	Africa
Asmara Brewery	1939	Distillers, Vintners, & Breweries	Eritrea	Africa
National Bank of Ethiopia	1906	Banking & Finance	Ethiopia	Africa
BGFIBank Group	1971	Banking & Finance	Gabon	Africa
Halco Mining	1962	Mining	Guinea	Africa
Correios da Guiné-Bissau	1973	Postal Service	Guinea-Bissau	Africa
KCB Group Limited	1896	Banking & Finance	Kenya	Africa
Central Bank of Lesotho	1978	Banking & Finance	Lesotho	Africa
National Port Authority	1921	Aviation & Transport	Liberia	Africa
Umma Bank	1907	Banking & Finance	Libya	Africa
Air Madagascar	1962	Aviation & Transport	Madagascar	Africa
Malawi Broadcasting Corporation	1964	Media	Malawi	Africa
Office de Radiodiffusion-Télévision du Mali	1957	Media	Mali	Africa
Central Bank of Mauritania	1973	Banking & Finance	Mauritania	Africa
Mauritius Post	1772	Postal Service	Mauritius	Africa
Attijariwafa Bank	1904	Banking & Finance	Morocco	Africa

Beira Railroad Corporation	1892	Aviation & Transport	Mozambique	Africa
NamPost	1814	Postal Service	Namibia	Africa
Office of Radio and Television of Niger	1960	Media	Niger	Africa
First Bank of Nigeria	1894	Banking & Finance	Nigeria	Africa
National Post Office	1922	Postal Service	Rwanda	Africa
Central Bank of São Tomé and Príncipe	1975	Banking & Finance	Sao Tome and Principe	Africa
Dakar–Niger Railway	1924	Aviation & Transport	Senegal	Africa
Air Seychelles	1977	Aviation & Transport	Seychelles	Asia
Rokel Commercial Bank	1917	Banking & Finance	Sierra Leone	Africa
Radio Mogadishu	1943	Media	Somalia	Africa
Premier FMCG	1820	Manufacturing & Production	South Africa	Africa
Ivory Bank	1994	Banking & Finance	South Sudan	Africa
Bank of Khartoum	1913	Banking & Finance	Sudan	Africa
Swazi Rail	1963	Aviation & Transport	Eswatini	Africa
Tanzania Breweries Limited	1933	Distillers, Vintners, & Breweries	Tanzania, United Republic of	Africa
La Poste du Togo	1883	Postal Service	Togo	Africa
La Poste Tunisienne	1847	Postal Service	Tunisia	Africa
Stanbic Bank Uganda Limited	1906	Banking & Finance	Uganda	Africa
ZamPost	1896	Postal Service	Zambia	Africa
Standard Chartered Zimbabwe	1892	Banking & Finance	Zimbabwe	Africa
Spinzar Cotton Company	1930	Agriculture	Afghanistan	Asia
Azerbaijan Caspian Shipping Company	1858	Aviation & Transport	Azerbaijan	Asia
BMMI	1883	Retail	Bahrain	Asia
M. M. Ispahani Limited	1820	Food & Beverages	Bangladesh	Asia
Tashi Group	1959	Aviation & Transport	Bhutan	Asia
Hua Ho Department Store	1947	Retail	Brunei Darussalam	Asia
National Bank of Cambodia	1954	Banking & Finance	Cambodia	Asia
Ma Yu Ching's Bucket Chicken House	1153	Cafés, Restaurants & Bars	China	Asia
Bank of Georgia	1903	Banking & Finance	Georgia	Asia
Wadia Group	1736	Manufacturing & Production	India	Asia
Pindad	1808	Defense	Indonesia	Asia
North Oil Company	1928	Energy	Iraq	Asia
Café Abu Salem	1914	Cafés, Restaurants & Bars	Israel	Asia

Kongō Gumi	578	Construction	Japan	Asia
Arab Bank	1930	Banking & Finance	Jordan	Asia
Bogatyr Access Komir	1913	Mining	Kazakhstan	Asia
M.H. Alshaya Co.	1890	Retail	Kuwait	Asia
Electricite du Laos	1959	Energy	Lao People's Democratic Republic	Asia
Bank Audi	1830	Banking & Finance	Lebanon	Asia
Pos Malaysia	1800	Postal Service	Malaysia	Asia
Mongolian National Broadcaster	1931	Media	Mongolia	Asia
Myanmar National Airlines	1948	Aviation & Transport	Myanmar	Asia
Nepal Bank Limited	1937	Banking & Finance	Nepal	Asia
Kim Chong-t'ae Electric Locomotive Works	1945	Aviation & Transport	Korea, Democratic People's Republic of	Asia
Petroleum Development Oman	1937	Energy	Oman	Asia
House of Habib	1841	Conglomerate	Pakistan	Asia
Destileria Limtuaco	1853	Distillers, Vintners, & Breweries	Philippines	Asia
Salam International Investment Limited	1952	Conglomerate	Qatar	Asia
Petrodvorets Watch Factory	1721	Consumer Goods	Russian Federation	Europe
House of Alireza	1845	Construction	Saudi Arabia	Asia
Singapore Post	1819	Postal Service	Singapore	Asia
KT Corporation	1885	Telecommunications	Korea, Republic of	Asia
George Steuart Group	1835	Food & Beverages	Sri Lanka	Asia
Bakdash	1885	Cafés, Restaurants & Bars	Syrian Arab Republic	Asia
Bank of Taiwan	1897	Banking & Finance	Taiwan, Province of China	Asia
B.Grimm Group	1878	Conglomerate	Thailand	Asia
Liwa Chemicals	1939	Manufacturing & Production	United Arab Emirates	Asia
Tashkent Aviation Production Association	1932	Manufacturing & Production	Uzbekistan	Asia
Vietnam Railways	1881	Aviation & Transport	Viet Nam	Asia
Yemenia Airways	1962	Aviation & Transport	Yemen	Asia
ALBtelecom	1912	Telecommunications	Albania	Europe
Andbank	1930	Banking & Finance	Andorra	Europe
Yerevan Ararat Brandy-Wine-Vodka Factory	1877	Distillers, Vintners, & Breweries	Armenia	Asia
St. Peter Stifts Kulinarium	803	Cafés, Restaurants & Bars	Austria	Europe

Olivaria Brewery	1864	Distillers, Vintners, & Breweries	Belarus	Europe
Affligem Brewery	1074	Distillers, Vintners, & Breweries	Belgium	Europe
Sarajevska Pivara	1864	Distillers, Vintners, & Breweries	Bosnia and Herzegovina	Europe
Arsenal AD	1878	Defense	Bulgaria	Europe
Kraljevica Shipyard	1729	Manufacturing & Production	Croatia	Europe
Bank of Cyprus	1899	Banking & Finance	Cyprus	Europe
Pivovar Broumov	1348	Distillers, Vintners, & Breweries	Czechia	Europe
Munke Mølle	1135	Manufacturing & Production	Denmark	Europe
The Royal Mint	886	Manufacturing & Production	United Kingdom	Europe
Raeapteek	1422	Medical	Estonia	Europe
Fiskars	1649	Consumer Goods	Finland	Europe
Monnaie de Paris	864	Manufacturing & Production	France	Europe
Staffelter Hof Winery	862	Distillers, Vintners, & Breweries	Germany	Europe
Piraeus Bank	1606	Banking & Finance	Greece	Europe
Zwack	1790	Distillers, Vintners, & Breweries	Hungary	Europe
Íslandspóstur	1776	Postal Service	Iceland	Europe
Sean's Bar	900	Cafés, Restaurants & Bars	Ireland	Europe
Marinelli Bell Foundry	1040	Manufacturing & Production	Italy	Europe
Meridian Corporation	1999	Media	Kosovo	Europe
Cēsu Alus	1590	Distillers, Vintners, & Breweries	Latvia	Europe
National Bank of Liechtenstein	1861	Banking & Finance	Liechtenstein	Europe
Gubernija	1665	Distillers, Vintners, & Breweries	Lithuania	Europe
Mousel	1511	Distillers, Vintners, & Breweries	Luxembourg	Europe
HSBC Bank Malta	1882	Banking & Finance	Malta	Europe
Pošta Crne Gore	1841	Postal Service	Montenegro	Europe
The Brand Brewery	1340	Distillers, Vintners, & Breweries	Netherlands	Europe
Tutunski kombinat Prilep	1873	Consumer Goods	North Macedonia	Europe
Posten Norge	1647	Postal Service	Norway	Europe
Bochnia Salt Mine	1248	Mining	Poland	Europe

CTT-Correios de Portugal SA	1520	Postal Service	Portugal	Europe
Ursus Breweries	1878	Distillers, Vintners, & Breweries	Romania	Europe
Apatin Brewery	1756	Distillers, Vintners, & Breweries	Serbia	Europe
Kremnica Mint	1328	Manufacturing & Production	Slovakia	Europe
Gostilna Gastuž	1467	Tourism & Hotels	Slovenia	Europe
Casa de Ganaderos	1218	Agriculture	Spain	Europe
Skyllbergs bruk	1346	Manufacturing & Production	Sweden	Europe
Gasthof Sternen	1230	Tourism & Hotels	Switzerland	Europe
Çemberlitaş Hamamı	1584	Tourism & Hotels	Turkey	Asia
Drohobych salt plant	1250	Manufacturing & Production	Ukraine	Europe
Mount Gay Rum	1703	Distillers, Vintners, & Breweries	Barbados	North America
Belize Bank	1902	Banking & Finance	Belize	North America
Hudson's Bay Company	1670	Retail	Canada	North America
Florida Ice and Farm Company	1908	Distillers, Vintners, & Breweries	Costa Rica	North America
Cubana de Aviación	1929	Aviation & Transport	Cuba	North America
The Chronicle (Dominica)	1909	Media	Dominica	North America
HSBC El Salvador	1891	Banking & Finance	El Salvador	North America
Corporación Multi Inversiones	1920	Food & Beverages	Guatemala	North America
Rhum Barbancourt	1862	Distillers, Vintners, & Breweries	Haiti	North America
National Railroad of Honduras	1870	Aviation & Transport	Honduras	North America
Rose Hall	1770	Tourism & Hotels	Jamaica	North America
La Casa de Moneda de México	1534	Manufacturing & Production	Mexico	North America
Flor de Caña	1890	Distillers, Vintners, & Breweries	Nicaragua	North America
National Bank of Panama	1904	Banking & Finance	Panama	North America
1st National Bank of St Lucia	1938	Banking & Finance	Saint Lucia	North America
House of Angostura	1830	Distillers, Vintners, & Breweries	Trinidad and Tobago	North America

Shirley Plantation	1638	Agriculture	United States	North America
Bank of the Province of Buenos Aires	1822	Banking & Finance	Argentina	South America
Banco Nacional de Bolivia	1871	Banking & Finance	Bolivia, Plurinational State of	South America
Casa da Moeda do Brasil	1694	Manufacturing & Production	Brazil	South America
Famae	1811	Defense	Chile	South America
Casa de Moneda de Colombia	1621	Manufacturing & Production	Colombia	South America
Banks DIH	1840	Food & Beverages	Guyana	South America
Casa Nacional de Moneda	1565	Banking & Finance	Peru	South America
Cafe Brasilero	1877	Cafés, Restaurants & Bars	Uruguay	South America
Hacienda Chuao	1660	Food & Beverages	Venezuela, Bolivarian Republic of	South America
Australia Post	1809	Postal Service	Australia	Oceania
Bank of New Zealand	1861	Banking & Finance	New Zealand	Oceania
European Trust Company	1991	Banking & Finance	Vanuatu	Oceania

In [15]:
```
%%nose
last_output = _

def test_resultset():
    try:
        assert str(type(last_output)) == "<class 'sql.run.ResultSet'>"
    except AssertionError:
        assert False, "Please ensure a SQL ResultSet is the output of the code
cell."
results = last_output.DataFrame()
def test_shape():
    try:
        assert results.shape == (163, 5)
    except AssertionError:
        assert False, "The results should have five columns and one hundred an
d sixty three rows."
def test_colnames():
    try:
        assert results.columns.tolist() == ['business', 'year_founded', 'categ
ory', 'country', 'continent']
    except AssertionError:
        assert False, "The results should have columns named business, year_fo
unded, category, country, and continent."
```

Out[15]:
```
3/3 tests passed
```

8. Counting categories by continent

Having `businesses` joined to `categories` and `countries` together means we can ask questions about both these things together. For example, which are the most common categories for the oldest businesses on each continent?

In [16]:
```
%%sql

-- Count the number of businesses in each continent and category
SELECT cnt.continent, cat.category, COUNT(*) AS n
    FROM businesses AS bus
    INNER JOIN categories as cat
        ON bus.category_code = cat.category_code
    INNER JOIN countries as cnt
        ON bus.country_code = cnt.country_code
    GROUP BY cnt.continent, cat.category;
```

```
 * postgresql:///oldestbusinesses
56 rows affected.
```

Out[16]:

continent	category	n
North America	Banking & Finance	4
Asia	Media	1
Asia	Defense	1
Europe	Postal Service	4
North America	Food & Beverages	1
Europe	Manufacturing & Production	8
Africa	Food & Beverages	1
Europe	Consumer Goods	3
Oceania	Banking & Finance	2
Asia	Agriculture	1
Africa	Mining	1
Asia	Retail	3
South America	Manufacturing & Production	2
Africa	Banking & Finance	17
North America	Tourism & Hotels	1
South America	Defense	1
Africa	Energy	1
Europe	Agriculture	1
Asia	Aviation & Transport	7
North America	Distillers, Vintners, & Breweries	5
Africa	Distillers, Vintners, & Breweries	3
Asia	Postal Service	2
Europe	Defense	1
Asia	Manufacturing & Production	3
Europe	Media	1
Oceania	Postal Service	1
South America	Food & Beverages	2
Europe	Tourism & Hotels	2
Europe	Medical	1
Africa	Manufacturing & Production	1
North America	Aviation & Transport	2
Asia	Distillers, Vintners, & Breweries	2
South America	Banking & Finance	3
Africa	Postal Service	9
Asia	Telecommunications	1
Europe	Mining	1

North America	Manufacturing & Production	1
Africa	Aviation & Transport	10
Asia	Construction	2
Asia	Energy	3
Asia	Cafés, Restaurants & Bars	3
Europe	Banking & Finance	5
Europe	Cafés, Restaurants & Bars	2
Asia	Banking & Finance	6
Asia	Conglomerate	3
North America	Media	1
Asia	Mining	1
Asia	Food & Beverages	2
Africa	Agriculture	3
North America	Agriculture	1
Europe	Telecommunications	1
Europe	Distillers, Vintners, & Breweries	12
South America	Cafés, Restaurants & Bars	1
North America	Retail	1
Africa	Media	4
Asia	Tourism & Hotels	1

In [17]:
```
%%nose
last_output = _

def test_resultset():
    try:
        assert str(type(last_output)) == "<class 'sql.run.ResultSet'>"
    except AssertionError:
        assert False, "Please ensure a SQL ResultSet is the output of the code
cell."
results = last_output.DataFrame()
def test_shape():
    try:
        assert results.shape == (56, 3)
    except AssertionError:
        assert False, "The results should have three columns and fifty six row
s."
def test_colnames():
    try:
        assert results.columns.tolist() == ['continent', 'category', 'n']
    except AssertionError:
        assert False, "The results should have continent, category, and count
 (as 'n')."
def test_count():
    try:
        assert results.loc[:, 'n'].sort_values(ascending=False).values.tolist
() == [17, 12, 10, 9, 8, 7, 6, 5, 5, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2,
2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1]
    except AssertionError:
        assert False, "The counts are not correct."
```

Out[17]:
```
4/4 tests passed
```

9. Filtering counts by continent and category

Combining continent and business category led to a lot of results. It's difficult to see what is important. To trim this down to a manageable size, let's restrict the results to only continent/category pairs with a high count.

In [18]:
```
%%sql

-- Repeat that previous query, filtering for results having a count greater th
an 5
SELECT cnt.continent, cat.category, COUNT(*) AS n
    FROM businesses AS bus
    INNER JOIN categories as cat
        ON bus.category_code = cat.category_code
    INNER JOIN countries as cnt
        ON bus.country_code = cnt.country_code
    GROUP BY cnt.continent, cat.category
    HAVING COUNT(*) > 5
    ORDER BY n DESC;
```

```
 * postgresql:///oldestbusinesses
7 rows affected.
```

Out[18]:

continent	category	n
Africa	Banking & Finance	17
Europe	Distillers, Vintners, & Breweries	12
Africa	Aviation & Transport	10
Africa	Postal Service	9
Europe	Manufacturing & Production	8
Asia	Aviation & Transport	7
Asia	Banking & Finance	6

In [19]:
```
%%nose
last_output = _

def test_resultset():
    try:
        assert str(type(last_output)) == "<class 'sql.run.ResultSet'>"
    except AssertionError:
        assert False, "Please ensure a SQL ResultSet is the output of the code
cell."
results = last_output.DataFrame()
def test_shape():
    try:
        assert results.shape == (7, 3)
    except AssertionError:
        assert False, "The results should have three columns and seven rows."
def test_colnames():
    try:
        assert results.columns.tolist() == ['continent', 'category', 'n']
    except AssertionError:
        assert False, "The results should have continent, category, and count
 (as 'n')."
def test_count():
    try:
        assert results.loc[:, 'n'].values.tolist() == [17, 12, 10, 9, 8, 7, 6]
    except AssertionError:
        assert False, "The counts are not correct."
```

Out[19]:
```
4/4 tests passed
```