Activity_Address missing data

July 16, 2024

1 Activity: Address missing data

1.1 Introduction

The datasets that data professionals use to solve problems typically contain missing values, which must be dealt with in order to achieve clean, useful data. This is particularly crucial in exploratory data analysis (EDA). In this activity, you will learn how to address missing data.

You are a financial data consultant, and an investor has tasked your team with identifying new business opportunities. To help them decide which future companies to invest in, you will provide a list of current businesses valued at more than $1 billion. These are sometimes referred to as "unicorns." Your client will use this information to learn about profitable businesses in general.

The investor has asked you to provide them with the following data: - Companies in the `hardware` industry based in either `Beijing`, `San Francisco`, or `London` - Companies in the `artificial intelligence` industry based in `London` - A list of the top 20 countries sorted by sum of company valuations in each country, excluding `United States`, `China`, `India`, and `United Kingdom` - A global valuation map of all countries with companies that joined the list after 2020 - A global valuation map of all countries except `United States`, `China`, `India`, and `United Kingdom` (a separate map for Europe is also required)

Your dataset includes a list of businesses and data points, such as the year they were founded; their industry; and their city, country, and continent.

1.2 Step 1: Imports

1.2.1 Import libraries

Import the following relevant Python libraries: * `numpy` * `pandas` * `matplotlib.pyplot` * `plotly.express` * `seaborn`

[2]:
```
# Import libraries and modules.

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

1.2.2 Load the dataset

The dataset is currently in CSV format and in a file named `Unicorn_Companies.csv`. As shown in this cell, the dataset has been automatically loaded in for you. You do not need to download the .csv file, or provide more code, in order to access the dataset and proceed with this lab. Please continue with this activity by completing the following instructions.

[4]:
```
# RUN THIS CELL TO IMPORT YOUR DATA.

### YOUR CODE HERE ###
df_companies = pd.read_csv("Unicorn_Companies.csv")
```

1.3 Step 2: Data exploration

Explore the dataset and answer questions that will guide your management of missing values.

1.3.1 Display top rows

Display the first 10 rows of the data to understand how the dataset is structured.

[5]:
```
# Display the first 10 rows of the data.

df_companies.head(10)
```

[5]:

	Company	Valuation	Date Joined	Industry	City	Country/Region	Continent	Year Founded	Funding
0	Bytedance	$180B	4/7/17	Artificial intelligence	Beijing	China	Asia	2012	$8B
1	SpaceX	$100B	12/1/12	Other	Hawthorne	United States	North America	2002	$7B
2	SHEIN	$100B	7/3/18	E-commerce & direct-to-consumer	Shenzhen	China	Asia	2008	$2B
3	Stripe	$95B	1/23/14	Fintech	San Francisco	United States	North America	2010	$2B
4	Klarna	$46B	12/12/11	Fintech	Stockholm	Sweden	Europe	2005	$4B
5	Canva	$40B	1/8/18	Internet software & services	Surry Hills	Australia	Oceania	2012	$572M
6	Checkout.com	$40B	5/2/19	Fintech	London	United Kingdom	Europe	2012	$2B
7	Instacart	$39B	12/30/14	Supply chain, logistics, & delivery	San Francisco	United States	North America	2012	$3B
8	JUUL Labs	$38B	12/20/17	Consumer & retail	San Francisco	United States	North America	2015	$14B
9	Databricks	$38B	2/5/19	Data management & analytics					

```
9  San Francisco   United States  North America          2013     $3B

                                    Select Investors
0  Sequoia Capital China, SIG Asia Investments, S…
1  Founders Fund, Draper Fisher Jurvetson, Rothen…
2  Tiger Global Management, Sequoia Capital China…
3        Khosla Ventures, LowercaseCapital, capitalG
4  Institutional Venture Partners, Sequoia Capita…
5  Sequoia Capital China, Blackbird Ventures, Mat…
6  Tiger Global Management, Insight Partners, DST…
7  Khosla Ventures, Kleiner Perkins Caufield & By…
8                            Tiger Global Management
9  Andreessen Horowitz, New Enterprise Associates…
```

Hint 1

Refer to the materials about exploratory data analysis in Python.

Hint 2

There is a function in the `pandas` library that allows you to get a specific number of rows from the top of a DataFrame.

Hint 3

Call the `head(10)` function from the `pandas` library.

1.3.2 Statistical properties of the dataset

Use methods and attributes of the dataframe to get information and descriptive statistics for the data, including its range, data types, mean values, and shape.

[7]:
```
# Get the shape of the dataset.

df_companies.shape
```

[7]:
```
(1074, 10)
```

Hint 1

Refer to the material about exploratory data analysis in Python.

Hint 2

Call the `shape` attribute of the dataframe.

Question: What is the shape of the dataset?

Ans: (1074, 10)

[8]:
```
# Get the data types and number of non-null values in the dataset.

```

```
df_companies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1074 entries, 0 to 1073
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Company           1074 non-null   object
 1   Valuation         1074 non-null   object
 2   Date Joined       1074 non-null   object
 3   Industry          1074 non-null   object
 4   City              1058 non-null   object
 5   Country/Region    1074 non-null   object
 6   Continent         1074 non-null   object
 7   Year Founded      1074 non-null   int64 
 8   Funding           1074 non-null   object
 9   Select Investors  1073 non-null   object
dtypes: int64(1), object(9)
memory usage: 84.0+ KB
```

Hint 1

Refer to the material about exploratory data analysis in Python.

Hint 2

Use the `info()` method.

Question: What are the data types of various columns?

Except for the Year Funded column, which is int64, the data type for all other columns is object. You can get this information using the info() method.

Question: How many columns contain null values?

Ans: City and select investors contain null values.

[9]:
```
# Get descriptive statistics such as mean, standard deviation, and range of the␣
↪numerical columns in the dataset.

df_companies.describe()
```

[9]:
```
       Year Founded
count   1074.000000
mean    2012.895717
std        5.698573
min     1919.000000
25%     2011.000000
50%     2014.000000
75%     2016.000000
max     2021.000000
```

Hint 1

Refer to the material about exploratory data analysis in Python.

Hint 2

There is a function in the `pandas` library that allows you to find descriptive statistics for the numeric columns in a DataFrame.

Hint 3

Call the `describe()` function from the `pandas` library.

Question: In what year was the oldest company founded?

The oldest company in the list was founded in 1919

1.3.3 Data Preprocessing

In order to answer the investor's questions, some data preprocessing steps are required. The first step is to add a new column to the dataframe containing just the year each company became a unicorn company. Call this new column `Year Joined`.

[10]:
```
# Create a new column "Year Joined" from "Date Joined".

df_companies['Year_Joined'] = pd.to_datetime(df_companies['Date Joined']).dt.
↪year
```

For each country, you want to calculate the sum of all valuations of companies from that country. However, in order to do this, you'll need to first prepare the data. Currently, the data in the `Valuation` column is a string that starts with a `$` and ends with a `B`. Because this column is not in a numeric datatype, pandas cannot perform mathematical operations on its values. The data in this column needs to be converted to a numeric datatype.

In this step, define a function called `str_to_num()` that accepts as an argument:

- `x`: a string in the format of the values contained in the `Valuation` column

And returns:

- `x`: an `int` of the number represented by the input string

`Example:`

```
 [IN]:  str_to_num('$4B')
[OUT]:  4
```

To do this, use the string `strip()` method. This method is applied to a string. Its argument is a string that contains all the characters that you want to remove from the beginning and end of a given string—in any order. The specified characters will be removed until a valid character is encountered. This process is applied moving forward from the beginning of the string and also moving in reverse from the end of the string, thus removing unwanted beginning and trailing characters.

Example:

```
[IN]:  my_string = '#....... Section 3.2.1 Issue #32 .......'
       my_string = my_string.strip('.#! ')
       print(my_string)

[OUT]:  'Section 3.2.1 Issue #32'
```

Note that you must reassign the result back to a variable or else the change will not be permanent.

[11]:
```
# Define the `str_to_num()` function

def str_to_num(x):
    x = x.strip('$B')
    x = int(x)

    return x
```

Hint 1

The unwanted characters in the values contained in the `Valuation` column are `'$'` and `'B'`.

Hint 2

Pass a string of the unwanted values as an argument to the `strip()` string method.

Hint 3

The final step before returning `x` should be converting it to an integer.

Now, use this function to create a new column called `valuation_num` that represents the `Valuation` column as an integer value. To do this, use the series method `apply()` to apply the `str_to_num()` function to the `Valuation` column.

`apply()` is a method that can be used on a `DataFrame` or `Series` object. In this case, you're using it on the `Valuation` series. The method accepts a function as an argument and applies that function to each value in the series.

Example:

```
[IN]: def square(x):
          return x ** 2

      my_series = pd.Series([0, 1, 2, 3])
      my_series

[OUT]: 0    0
       1    1
       2    2
       3    3
       dtype: int64
```

```
[IN]: my_series = my_series.apply(square)
      my_series

[OUT]: 0    0
       1    1
       2    4
       3    9
       dtype: int64
```

Notice that the function passed as an argument to the `apply()` method does not have parentheses. It's just the function name.

[12]:
```
# Apply the `str_to_num()` function to the `Valuation` column
# and assign the result back to a new column called `valuation_num`

df_companies['valuation_num'] = df_companies['Valuation'].apply(str_to_num)
df_companies[['Valuation', 'valuation_num']].head()
```

[12]:

	Valuation	valuation_num
0	$180B	180
1	$100B	100
2	$100B	100
3	$95B	95
4	$46B	46

1.3.4 Find missing values

The unicorn companies dataset is fairly clean, with few missing values.

[13]:
```
# Find the number of missing values in each column in this dataset.

df_companies.isna().sum()
```

[13]:
```
Company              0
Valuation            0
Date Joined          0
Industry             0
City                16
Country/Region       0
Continent            0
Year Founded         0
Funding              0
Select Investors     1
Year_Joined          0
valuation_num        0
dtype: int64
```

Hint 1

The `isna()` `DataFrame` method will return a dataframe of Boolean values in the same shape as your original dataframe. Values are `True` if the data is missing and `False` if it is not missing.

Hint 2

You'll need to convert Boolean values into numerical values. Remember that `True` values are considered `1` and `False` values are considered `0`.

Hint 3

After applying the `isna()` method to the `df_companies` dataframe, apply the `sum()` method to the results to return a pandas `Series` object with each column name and the number of `NaN` values it contains.

Question: How many missing values are in each column in the dataset?

There is a single missing value in the Select Investors column and 16 missing cities. There are no missing values in other columns.

1.3.5 Review rows with missing values

Before dealing with missing values, it's important to understand the nature of the missing value that is being filled. Display all rows with missing values from `df_companies`. To do this, perform the following three steps:

1. Apply the `isna()` method to the `df_companies` dataframe as you did in the last step. Remember, this results in a dataframe of the same shape as `df_companies` where each value is `True` if its contents are `NaN` and a `False` if its contents are not `NaN`. Assign the results to a variable called `mask`.

[14]:
```
# 1. Apply the `isna()` method to the `df_companies` dataframe and assign back␣
↪to `mask`

mask = df_companies.isna()
mask.tail()
```

[14]:

	Company	Valuation	Date Joined	Industry	City	Country/Region	\
1069	False	False	False	False	False	False	
1070	False	False	False	False	False	False	
1071	False	False	False	False	False	False	
1072	False	False	False	False	False	False	
1073	False	False	False	False	False	False	

	Continent	Year Founded	Funding	Select Investors	Year_Joined	\
1069	False	False	False	False	False	
1070	False	False	False	False	False	
1071	False	False	False	False	False	
1072	False	False	False	False	False	
1073	False	False	False	False	False	

```
      valuation_num
1069          False
1070          False
1071          False
1072          False
1073          False
```

You're not done yet. You still need to go from this dataframe of Boolean values to a dataframe of just the rows of `df_companies` that contain at least one `NaN` value. This means that you need a way to find the indices of the rows of the Boolean dataframe that contain at least one `True` value, then extract those indices from `df_companies`.

You can do this using the `any()` method for `DataFrame` objects. This method returns a Boolean `Series` indicating whether any value is `True` over a specified axis.

`Example:`

```
df =
        A      B    C
    0   0      a    10
    1   False  0    1
    2   NaN    NaN  NaN


 [IN]: df.any(axis=0)

[OUT]: A    False
       B     True
       C     True
       dtype: bool

 [IN]: df.any(axis=1)

[OUT]: 0     True
       1     True
       2    False
       dtype: bool
```

Note that `0`, `False`, and `NaN` are considered `False` and anything else is considered `True`.

2. Apply the `any()` method to the Boolean dataframe you created to make a Boolean series where each element in the series represents `True` if a row of the dataframe contains any `True` values and `False` if any row in the dataframe contains any `False` values. Assign the results back to `mask`.

[15]:
```
# 2. Apply the `any()` method to `mask` and assign the results back to `mask`

mask = mask.any(axis=1)
mask.head()
```

[15]:
```
0    False
1    False
2    False
3    False
4    False
dtype: bool
```

Hint 1

Refer to the example given for how to use the `any()` method for dataframes.

Hint 2

Using the provided example as a guide, which axis returns **rows** that have at least one `True` value?

Hint 3

`mask.any(axis=1)` will return a Boolean series that represents whether each row of `mask` contains at least one `True` value.

3. Because `mask` is now a series of Boolean values, you can use it as a Boolean mask. Apply the Boolean mask to the `df_companies` dataframe to return a filtered dataframe containing just the rows that contain a missing value. Assign the results to a variable called `df_missing_rows`.

[16]:
```
# 3. Apply `mask` as a Boolean mask to `df_companies` and assign results to␣
↪`df_missing_rows`

df_missing_rows = df_companies[mask]
df_missing_rows
```

[16]:

	Company	Valuation	Date Joined
12	FTX	$32B	7/20/21
170	HyalRoute	$4B	5/26/20
242	Moglix	$3B	5/17/21
251	Trax	$3B	7/22/19
325	Amber Group	$3B	6/21/21
382	Ninja Van	$2B	9/27/21
541	Advance Intelligence Group	$2B	9/23/21
629	LinkSure Network	$1B	1/1/15
811	Carousell	$1B	9/15/21
848	Matrixport	$1B	6/1/21
880	bolttech	$1B	7/1/21
889	Carro	$1B	6/14/21
893	Cider	$1B	9/2/21
980	NIUM	$1B	7/13/21
986	ONE	$1B	12/8/21
994	PatSnap	$1B	3/16/21
1061	WeLab	$1B	11/8/17

	Industry	City	Country/Region	Continent	Year Founded	Funding	Select Investors	Year_Joined
12	Fintech	NaN	Bahamas	North America	2018	$2B	Sequoia Capital, Thoma Bravo, Softbank	2021
170	Mobile & telecommunications	NaN	Singapore	Asia	2015	$263M	Kuang-Chi	2020
242	E-commerce & direct-to-consumer	NaN	Singapore	Asia	2015	$471M	Jungle Ventures, Accel, Venture Highway	2021
251	Artificial intelligence	NaN	Singapore	Asia	2010	$1B	Hopu Investment Management, Boyu Capital, DC T…	2019
325	Fintech	NaN	Hong Kong	Asia	2015	$328M	Tiger Global Management, Tiger Brokers, DCM Ve…	2021
382	Supply chain, logistics, & delivery	NaN	Singapore	Asia	2014	$975M	B Capital Group, Monk's Hill Ventures, Dynamic…	2021
541	Artificial intelligence	NaN	Singapore	Asia	2016	$536M	Vision Plus Capital, GSR Ventures, ZhenFund	2021
629	Mobile & telecommunications	Shanghai	China	Asia	2013	$52M	NaN	2015
811	E-commerce & direct-to-consumer	NaN	Singapore	Asia	2012	$288M		
848	Fintech	NaN	Singapore	Asia	2019	$100M		
880	Fintech	NaN	Singapore	Asia	2018	$210M		
889	E-commerce & direct-to-consumer	NaN	Singapore	Asia	2015	$595M		
893	E-commerce & direct-to-consumer	NaN	Hong Kong	Asia	2020	$140M		
980	Fintech	NaN	Singapore	Asia	2014	$285M		
986	Internet software & services	NaN	Singapore	Asia	2011	$515M		
994	Internet software & services	NaN	Singapore	Asia	2007	$352M		
1061	Fintech	NaN	Hong Kong	Asia	2013	$871M		

```
811   500 Global, Rakuten Ventures, Golden Gate Vent…         2021
848   Dragonfly Captial, Qiming Venture Partners, DS…         2021
880   Mundi Ventures, Doqling Capital Partners, Acti…         2021
889   SingTel Innov8, Alpha JWC Ventures, Golden Gat…         2021
893        Andreessen Horowitz, DST Global, IDG Capital          2021
980   Vertex Ventures SE Asia, Global Founders Capit…         2021
986   Temasek, Guggenheim Investments, Qatar Investm…         2021
994   Sequoia Capital China, Shunwei Capital Partner…         2021
1061  Sequoia Capital China, ING, Alibaba Entreprene…         2017

      valuation_num
12               32
170               4
242               3
251               3
325               3
382               2
541               2
629               1
811               1
848               1
880               1
889               1
893               1
980               1
986               1
994               1
1061              1
```

Question: Is there a specific country/region that shows up a lot in this missing values dataframe? Which one?

Twelve of the 17 rows with missing values are for companies from Singapore.

Question: What steps did you take to find missing data?

DataFrame.isna() will return a Boolean dataframe indicating every location that is NaN with True

You can use sum() in conjunction with isna() to get the counts of NaN values in each column.

You can use any() in conjunction with isna() to create a Boolean mask, which can be applied to the original dataframe to obtain just the rows with at least one NaN value.

Question: What observations can be made about the forms and context of missing data?

Missing values can take different forms and are usually context-specific. Not every missing value is labeled as na or None, or Null.

Question: What other methods could you use to address missing data?

If possible, ask the business users for insight into the causes of missing values and, if possible, get domain knowledge to intelligently impute these values.

1.4 Step 3: Model building

Think of the model you are building as the completed dataset, which you will then use to inform the questions the investor has asked of you.

1.4.1 Two ways to address missing values

There are several ways to address missing values, which is critical in EDA. The two primary methods are removing them and imputing other values in their place. Choosing the proper method depends on the business problem and the value the solution will add or take away from the dataset.

Here, you will try both.

To compare the the effect of different actions, first store the original number of values in a variable. Create a variable called `count_total` that is an integer representing the total number of values in `df_companies`. For example, if the dataframe had 5 rows and 2 columns, then this number would be 10.

[17]:
```
# Store the total number of values in a variable called `count_total`

count_total = df_companies.size
count_total
```

[17]: 12888

Now, remove all rows containing missing values and store the total number of remaining values in a variable called `count_dropna_rows`.

[18]:
```
# Drop the rows containing missing values, determine number of remaining values

count_dropna_rows = df_companies.dropna().size
count_dropna_rows
```

[18]: 12684

Hint

Use the `dropna()` dataframe method to drop rows with missing values.

Now, remove all columns containing missing values and store the total number of cells in a variable called `count_dropna_columns`.

[19]:
```
# Drop the columns containing missing values, determine number of remaining␣
↪values

count_dropna_columns = df_companies.dropna(axis=1).size
```

```
count_dropna_columns
```

[19]: 10740

Hint

Specify `axis=1` to the `dropna()` method to drop columns with missing values.

Next, print the percentage of values removed by each method and compare them.

[20]:
```
# Print the percentage of values removed by dropping rows.

row_percent = ((count_total - count_dropna_rows) / count_total) * 100
print(f'Percentage removed, rows: {row_percent:.3f}')

# Print the percentage of values removed by dropping columns.

col_percent = ((count_total - count_dropna_columns) / count_total) * 100
print(f'Percentage removed, columns: {col_percent:.3f}')
```

```
Percentage removed, rows: 1.583
Percentage removed, columns: 16.667
```

Question: Which method was most effective? Why?

The percentage removed was significantly higher for columns than it was for rows. Since both approaches result in a dataset with no missing values, the "most effective" method depends on how much data you have and what you want to do with it. It might be best to use the way that leaves the most data intact—in this case, dropping rows. Or, if you don't have many samples and don't want to lose any, but you don't need all your columns, then dropping columns might be best. With this data, it would probably be best to drop rows in the majority of cases.

Now, practice the second method: imputation. Perform the following steps:

1. Use the `fillna()` dataframe method to fill each missing value with the next non-NaN value in its column. Assign the results to a new dataframe called `df_companies_backfill`.

Example:

```
df =
        A    B    C
    0   5    a    NaN
    1   10   NaN  False
    2   NaN  c    True

 [IN]: df.fillna(method='backfill')

[OUT]:
        A    B    C
    0   5    a    False
    1   10   c    False
```

```
2   NaN  c    True
```

Notice that if there is a `NaN` value in the last row, it will not backfill because there is no subsequent value in the column to refer to.

2. Show the rows that previously had missing values.

[21]:
```
# 1. Fill missing values using the 'fillna()' method, back-filling

df_companies_backfill = df_companies.fillna(method='backfill')

# 2. Show the rows that previously had missing values


df_companies_backfill.iloc[df_missing_rows.index, :]
```

[21]:

	Company	Valuation	Date Joined	Industry	City	Country/Region
12	FTX	$32B	7/20/21	Fintech	Jacksonville	Bahamas
170	HyalRoute	$4B	5/26/20	Mobile & telecommunications	El Segundo	Singapore
242	Moglix	$3B	5/17/21	E-commerce & direct-to-consumer	San Francisco	Singapore
251	Trax	$3B	7/22/19	Artificial intelligence	Amsterdam	Singapore
325	Amber Group	$3B	6/21/21	Fintech	San Francisco	Hong Kong
382	Ninja Van	$2B	9/27/21	Supply chain, logistics, & delivery	San Francisco	Singapore
541	Advance Intelligence Group	$2B	9/23/21	Artificial intelligence	Helsinki	Singapore
629	LinkSure Network	$1B	1/1/15	Mobile & telecommunications	Shanghai	China
811	Carousell	$1B	9/15/21	E-commerce & direct-to-consumer	New York	Singapore
848	Matrixport	$1B	6/1/21	Fintech	San Francisco	Singapore
880	bolttech	$1B	7/1/21	Fintech	Englewood	Singapore
889	Carro	$1B	6/14/21	E-commerce & direct-to-consumer	Lincoln	Singapore
893	Cider	$1B	9/2/21			
980	NIUM	$1B	7/13/21			
986	ONE	$1B	12/8/21			
994	PatSnap	$1B	3/16/21			
1061	WeLab	$1B	11/8/17			

```
893       E-commerce & direct-to-consumer    Mexico City      Hong Kong
980                                Fintech      Bengaluru      Singapore
986           Internet software & services       New York      Singapore
994           Internet software & services         London      Singapore
1061                               Fintech        Beijing      Hong Kong

          Continent  Year Founded Funding  \
12    North America          2018     $2B
170            Asia          2015   $263M
242            Asia          2015   $471M
251            Asia          2010     $1B
325            Asia          2015   $328M
382            Asia          2014   $975M
541            Asia          2016   $536M
629            Asia          2013    $52M
811            Asia          2012   $288M
848            Asia          2019   $100M
880            Asia          2018   $210M
889            Asia          2015   $595M
893            Asia          2020   $140M
980            Asia          2014   $285M
986            Asia          2011   $515M
994            Asia          2007   $352M
1061           Asia          2013   $871M

                                       Select Investors  Year_Joined  \
12               Sequoia Capital, Thoma Bravo, Softbank         2021
170                                          Kuang-Chi         2020
242             Jungle Ventures, Accel, Venture Highway         2021
251   Hopu Investment Management, Boyu Capital, DC T…         2019
325   Tiger Global Management, Tiger Brokers, DCM Ve…         2021
382   B Capital Group, Monk's Hill Ventures, Dynamic…         2021
541         Vision Plus Capital, GSR Ventures, ZhenFund         2021
629   Sequoia Capital India, The Times Group, GMO Ve…         2015
811   500 Global, Rakuten Ventures, Golden Gate Vent…         2021
848   Dragonfly Captial, Qiming Venture Partners, DS…         2021
880   Mundi Ventures, Doqling Capital Partners, Acti…         2021
889   SingTel Innov8, Alpha JWC Ventures, Golden Gat…         2021
893        Andreessen Horowitz, DST Global, IDG Capital         2021
980   Vertex Ventures SE Asia, Global Founders Capit…         2021
986   Temasek, Guggenheim Investments, Qatar Investm…         2021
994   Sequoia Capital China, Shunwei Capital Partner…         2021
1061  Sequoia Capital China, ING, Alibaba Entreprene…         2017

      valuation_num
12               32
170               4
```

```
242             3
251             3
325             3
382             2
541             2
629             1
811             1
848             1
880             1
889             1
893             1
980             1
986             1
994             1
1061            1
```

Hint 1

To backfill missing values, refer to the example provided.

Hint 2

To show the rows that previously had missing values, you'll need the indices of the rows that had missing values.

Hint 3

- You already have a dataframe of rows with missing values. It's stored in a variable called `df_missing_rows`.
- To access its index, call `df_missing_rows.index`. This will give you the row numbers of rows with missing values.
- Use these index numbers in an iloc[] selection statement on the `df_companies_backfill` dataframe to extract those row numbers.

Question: Do the values that were used to fill in for the missing values make sense?

No, the values seem to be added without consideration of the country those cities are located in.

Another option is to fill the values with a certain value, such as 'Unknown'. However, doing so doesn't add any value to the dataset and could make finding the missing values difficult in the future. Reviewing the missing values in this dataset determines that it is fine to leave the values as they are. This also avoids adding bias to the dataset.

1.5 Step 4: Results and evaluation

Now that you've addressed your missing values, provide your investor with their requested data points.

1.5.1 Companies in the Hardware Industry

Your investor is interested in identifying unicorn companies in the Hardware industry in the following cities: Beijing, San Francisco, and London. They are also interested in companies in the Artificial intelligence industry in London.

Write a selection statement that extracts the rows that meet these criteria. This task requires complex conditional logic. Break the process into the following parts.

1. Create a mask to apply to the df_companies dataframe. The following logic is a pseudo-code representation of how this mask could be structured.

```
((Industry==Hardware) and (City==Beijing, San Francisco, or London))
OR
((Industry==Artificial intelligence) and (City==London))
```

You're familiar with how to create Boolean masks based on conditional logic in pandas. However, you might not know how to write a conditional statement that selects rows that have *any one of several possible values* in a given column. In this case, this is the (City==Beijing, San Francisco, or London) part of the expression.

For this type of construction, use the isin() Series method. This method is applied to a pandas series and, for each value in the series, checks whether it is a member of whatever is passed as its argument.

Example:

```
 [IN]: my_series = pd.Series([0, 1, 2, 3])
       my_series

[OUT]: 0    0
       1    1
       2    2
       3    3
       dtype: int64

 [IN]: my_series.isin([1, 2])

[OUT]: 0    False
       1     True
       2     True
       3    False
       dtype: bool
```

2. Apply the mask to the df_companies dataframe and assign the result to a new variable called df_invest.

[22]:
```
# 1. Create a Boolean mask using conditional logic

cities = ['Beijing', 'San Francisco', 'London']
```

```
mask = (
    (df_companies['Industry']=='Hardware') & (df_companies['City'].isin(cities))
) | (
    (df_companies['Industry']=='Artificial intelligence') &␣
 ↪(df_companies['City']=='London')
)

# 2. Apply the mask to the `df_companies` dataframe and assign the results to␣
 ↪`df_invest`

df_invest = df_companies[mask]
df_invest
```

[22]:

	Company	Valuation	Date Joined	Industry	City	Country/Region	Continent	Year Founded	Funding	Select Investors	Year_Joined	valuation_num
36	Bitmain	$12B	7/6/18	Hardware	Beijing	China	Asia	2015	$765M	Coatue Management, Sequoia Capital China, IDG …	2018	12
43	Global Switch	$11B	12/22/16	Hardware	London	United Kingdom	Europe	1998	$5B	Aviation Industry Corporation of China, Essenc…	2016	11
147	Chipone	$5B	12/16/21	Hardware	Beijing	China	Asia	2008	$1B	China Grand Prosperity Investment, Silk Road H…	2021	5
845	Density	$1B	11/10/21	Hardware	San Francisco	United States	North America	2014	$217M	Founders Fund, Upfront Ventures, 01 Advisors	2021	
873	BenevolentAI	$1B	6/2/15	Artificial intelligence	London	United Kingdom	Europe	2013	$292M	Woodford Investment Management	2015	
923	Geek+	$1B	11/21/18	Hardware	Beijing	China	Asia	2015	$439M	Volcanics Ventures, Vertex Ventures China, War…	2018	
1040	TERMINUS Technology	$1B	10/25/18	Hardware	Beijing	China	Asia	2015	$623M	China Everbright Limited, IDG Capital, iFLYTEK	2018	
1046	Tractable	$1B	6/16/21	Artificial intelligence	London	United Kingdom	Europe	2014	$120M	Insight Partners, Ignition Partners, Georgian …	2021	

```
845     1
873     1
923     1
1040    1
1046    1
```

Hint 1

- Remember that pandas uses `&` for "and", `|` for "or", and `~` for "not".
  - Remember that each condition needs to be in its own set of parentheses. Refer to the above pseudo-code for an example.

Hint 2

- Use `(Series.isin(list_of_cities))` to represent the logic: (City==Beijing, San Francisco, or London)'.
  - There are two sets of conditional pairs: ((A) and (B)) or ((C) and (D)). Make sure the parentheses reflect this logic.

Hint 3

Consider using the following code:

```
cities = ['Beijing', 'San Francisco', 'London']     mask = (
(df_companies['Industry']=='Hardware') & (df_companies['City'].isin(cities))
) | (         (df_companies['Industry']=='Artificial intelligence') &
(df_companies['City']=='London')     )     df_invest = df_companies[mask]
```

Question: How many companies meet the criteria given by the investor?

Eight companies

1.5.2 List of countries by sum of valuation

For each country, sum the valuations of all companies in that country, then sort the results in descending order by summed valuation. Assign the results to a variable called `national_valuations`.

[23]:
```
# Group the data by`Country/Region`

national_valuations = df_companies.groupby(['Country/Region'])['valuation_num'].
 ↪sum(
).sort_values(ascending=False).reset_index()

# Print the top 15 values of the DataFrame.


national_valuations.head(15)
```

[23]:

	Country/Region	valuation_num
0	United States	1933
1	China	696

2	India	196
3	United Kingdom	195
4	Germany	72
5	Sweden	63
6	Australia	56
7	France	55
8	Canada	49
9	South Korea	41
10	Israel	39
11	Brazil	37
12	Bahamas	32
13	Indonesia	28
14	Singapore	21

Hint

Use a `groupby()` statement to group by `Country/Region`, then isolate the `valuation_num` column, sum it, and use the `sort_values()` method to sort the results.

Question: Which countries have the highest sum of valuation?

The sorted data indicates that the four countries with highest total company valuations are the United States, China, India, and the United Kingdom. However, your investor specified that these countries should not be included in the list because they are outliers.

1.5.3 Filter out top 4 outlying countries

Use this grouped and summed data to plot a barplot. However, to meet the needs of your stakeholder, you must first remove the United States, China, India, and the United Kingdom. Remove these countries from `national_valuations` and reassign the results to a variable called `national_valuations_no_big4`.

[24]:
```
# Remove outlying countries

national_valuations_no_big4 = national_valuations.iloc[4:, :]

national_valuations_no_big4.head()
```

[24]:

	Country/Region	valuation_num
4	Germany	72
5	Sweden	63
6	Australia	56
7	France	55
8	Canada	49

Hint

There are a number of ways to accomplish this task. One of the easiest ways is to use a simple iloc[] selection statement to select row indices 4–end and all columns of `national_valuations`.

1.5.4 BONUS CONTENT: Alternative approach (optional)

You can also use `isin()` to create a Boolean mask to filter out specific values of the `Country/Region` column. In this case, this process is longer and more complicated than simply using the iloc[] statement. However, there will be situations where this is the most direct approach.

How could you use `isin()` and your knowledge of pandas conditional operators and Boolean masks to accomplish the same task?

[25]:
```
# (Optional) Use `isin()` to create a Boolean mask to accomplish the same task

mask = ~national_valuations['Country/Region'].isin(['United States', 'China',␣
↪'India', 'United Kingdom'])
national_valuations_no_big4 = national_valuations[mask]
national_valuations_no_big4.head()
```

[25]:

	Country/Region	valuation_num
4	Germany	72
5	Sweden	63
6	Australia	56
7	France	55
8	Canada	49

Answer

In this case, there are 46 total countries and you want to keep countries 5–46 and filter out countries 1–4. To use `isin()` would require you to list out 42 countries:

```
mask = national_valuations['Country/Region'].isin(['country_5', 'country_6', ... 'country_46']
```

This is very impractical. However, you can invert the statement to simplify the job. The above impractical statement becomes:

```
mask = ~national_valuations['Country/Region'].isin(['country_1', 'country_2', 'country_3', 'cou
```

Notice the `~` that precedes the whole statement. This transforms the meaning from "country is in this list" to "country is NOT in this list."

Then, simply apply the mask to `national_valuations` and assign the result back to `national_valuations_no_big4`.

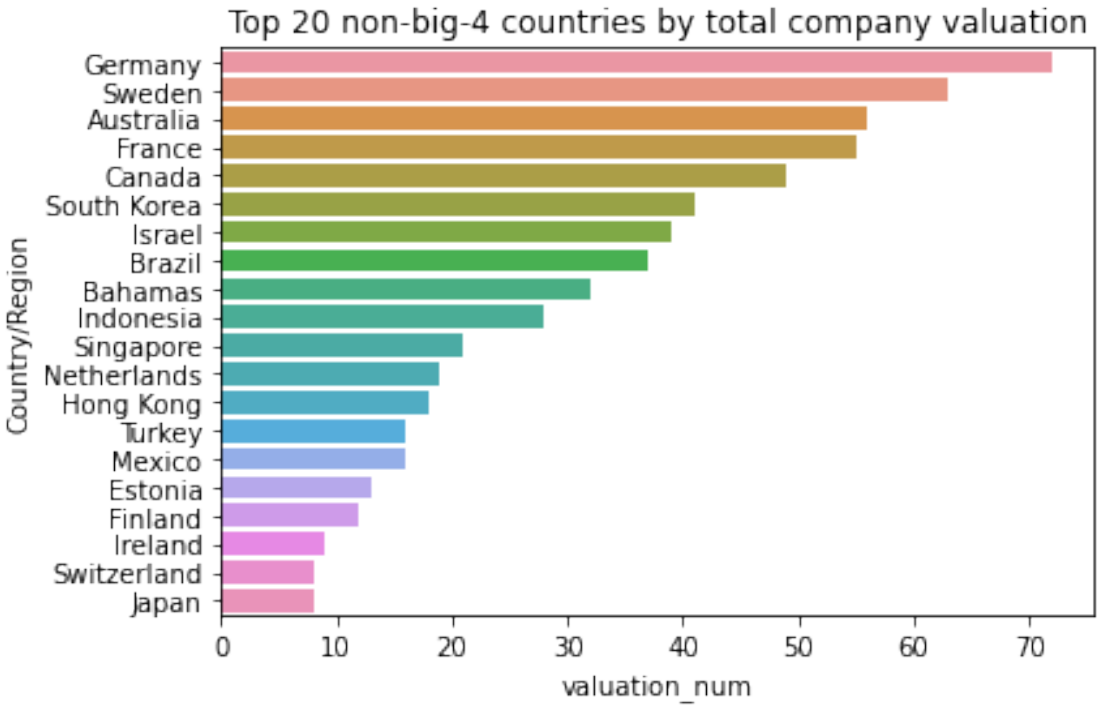
1.5.5 Create barplot for top 20 non-big-4 countries

Now, the data is ready to reveal the top 20 non-big-4 countries with the highest total company valuations. Use seaborn's `barplot()` function to create a plot showing national valuation on one axis and country on the other.

[26]:
```
# Create a barplot to compare the top 20 countries with highest company␣
↪valuations.

sns.barplot(data=national_valuations_no_big4.head(20),
```

```
            y='Country/Region',
            x='valuation_num')
plt.title('Top 20 non-big-4 countries by total company valuation')
plt.show();
```



Hint 1

Select the top 20 rows in `national_valuations_no_big4`.

Hint 2

- Select the top 20 rows in `df_companies_sum_outliers_removed` by using the `head(20)` method.
- Specify `Country/Region` for the `x` parameter of the function and `valuation_num` for the `y` parameter of the function (or vice versa).

1.5.6 Plot maps

Your investor has also asked for a global valuation map of all countries except `United States`, `China`, `India`, and `United Kingdom` (a.k.a. "big-four countries").

You have learned about using `scatter_geo()` from the `plotly.express` library to create plot data on a map. Create a `scatter_geo()` plot that depicts the total valuations of each non-big-four country on a world map, where each valuation is shown as a circle on the map, and the size of the circle is proportional to that country's summed valuation.

[27]:
```
# Plot the sum of valuations per country.

data = national_valuations_no_big4

px.scatter_geo(data,
               locations='Country/Region',
               size='valuation_num',
               locationmode='country names',
               color='Country/Region',
               title='Total company valuations by country (non-big-four)')
```

Hint 1

Use the `national_valuations_no_big4` dataframe that you already created.

Hint 2

To plot the data: * Use `national_valuations_no_big4` as the `data_frame` argument of the `scatter_geo()` function. * Use `'Country/Region'` as the `locations` argument. * Use `'country names'` as the `locationmode` argument. * Use `'Country/Region'` as the `color` argument.

Don't forget to include a title!

Question: How is the valuation sum per country visualized in the plot?

Valuation sum per country is visualized by the size of circles around the map.

Question: Does any region stand out as having a lot of activity?

Europe has a lot of unicorn companies in a concentrated area.

1.6 Conclusion

What are some key takeaways that you learned during this lab?

- Missing data is a common problem for data professionals anytime they work with a data sample.
- Addressing missing values is a part of the data-cleaning process and an important step in EDA.
- Address missing values by either removing them or filling them in.
- When considering how to address missing values, keep in mind the business, the data, and the questions to be answered. Always ensure you are not introducing bias into the dataset.
- Addressing the missing values enabled you to answer your investor's questions.

How would you present your findings from this lab to others? Consider the information you would provide (and what you would omit), how you would share the various data insights, and how data visualizations could help your presentation.

- For the industry specific companies in certain locations, you could provide a short list of company names and locations.
- For the top 20 countries by sum of valuations, you could use the plot you created in this lab or share a list.

- For the top 20 countries sorted by sum of company valuations in each country, you would exclude United States, China, India, and United Kingdom.
- For the questions concerning the valuation map, in addition to your visuals, you would provide a short summary of the data points. This is because the investor did not request a further breakdown of this data.

Reference

Bhat, M.A. *Unicorn Companies*

Congratulations! You've completed this lab. However, you may not notice a green check mark next to this item on Coursera's platform. Please continue your progress regardless of the check mark. Just click on the "save" icon at the top of this notebook to ensure your work has been logged.