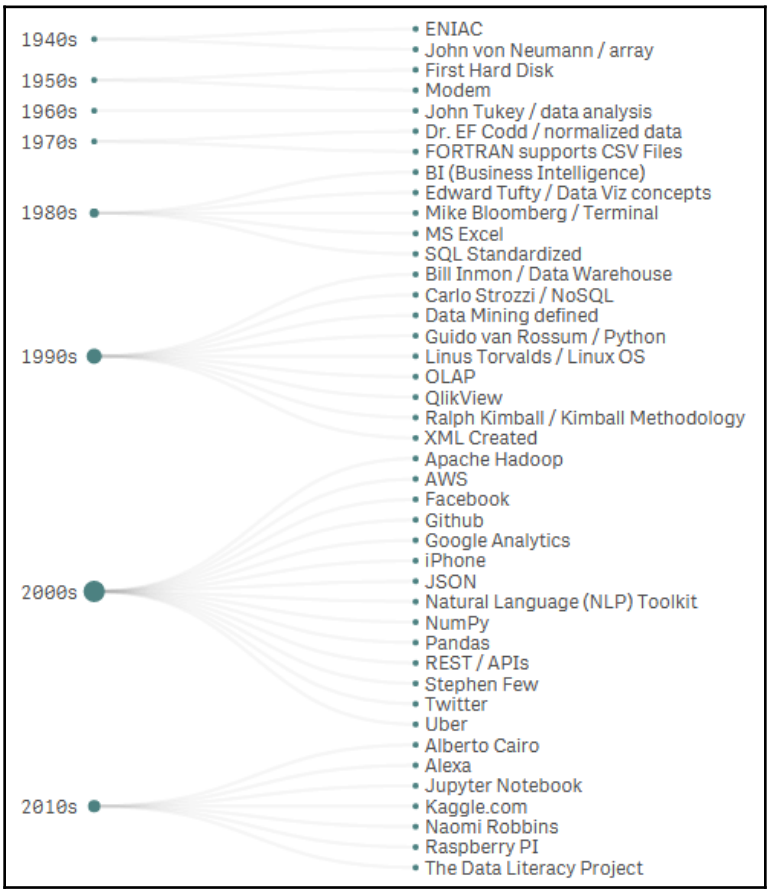
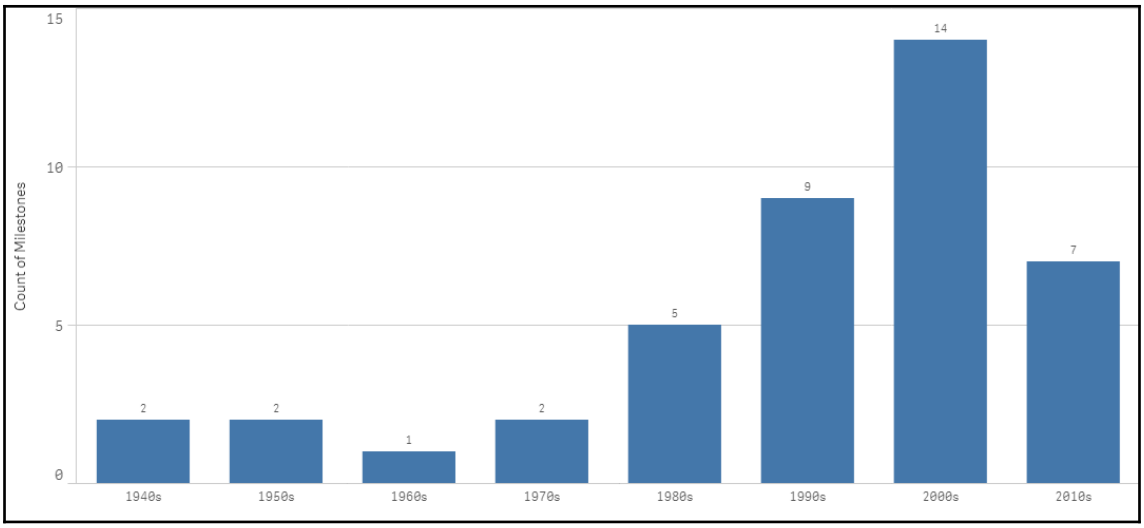
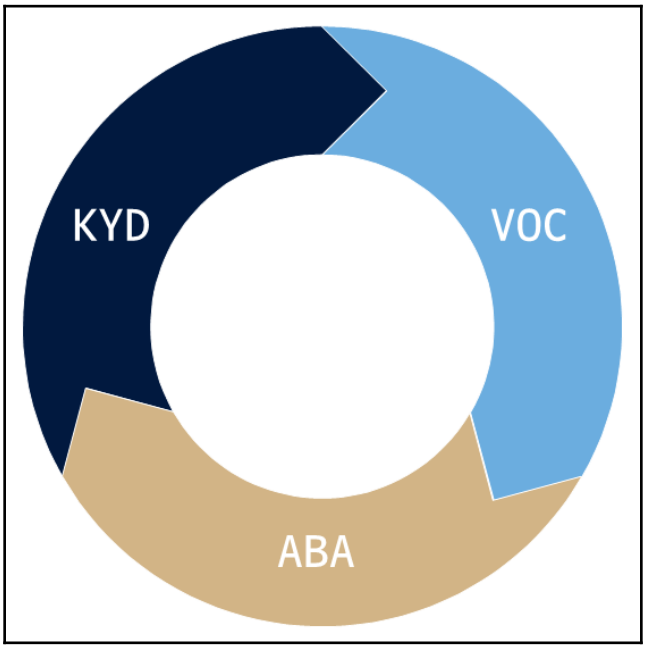
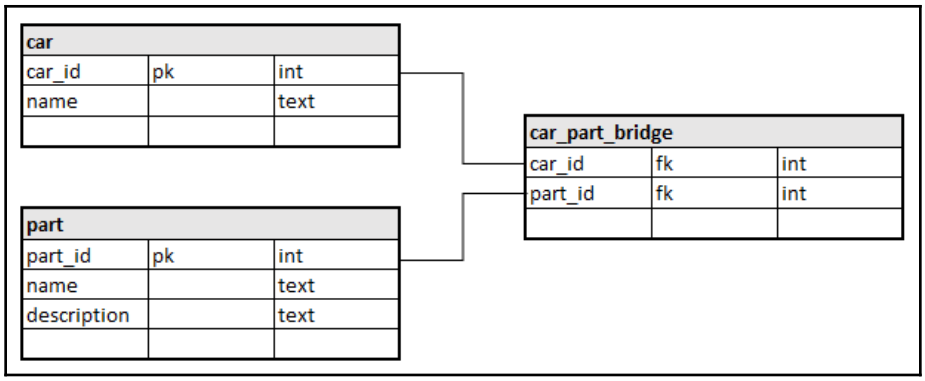
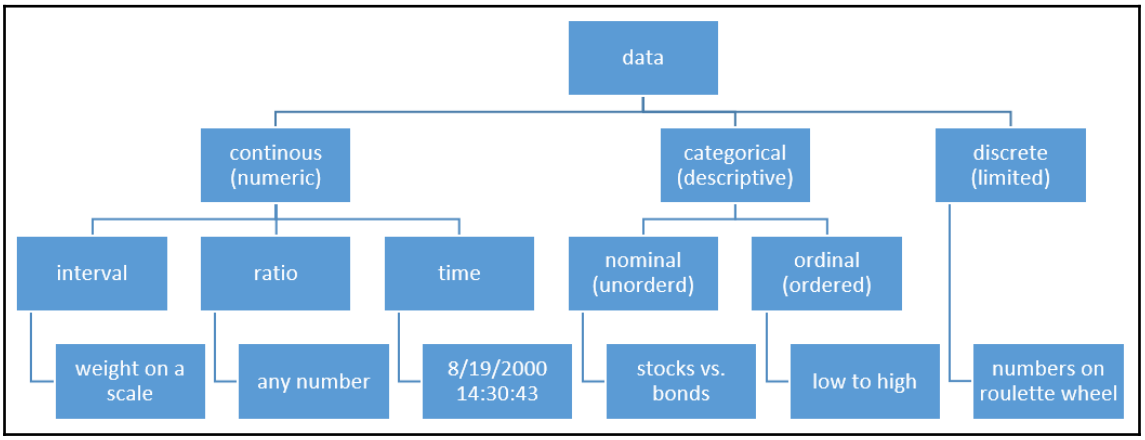
Chapter 1: Fundamentals of Data Analysis

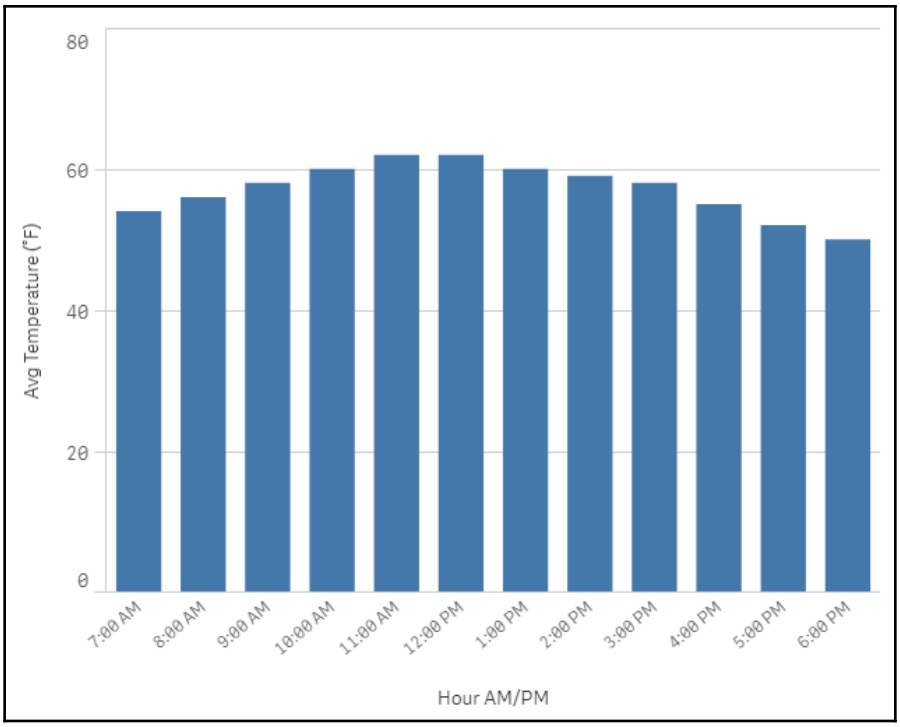



Count of Milestones
15
10
5
0
2
2
1
2
5
9
14
7
1940s
1950s
1960s
1970s
1980s
1990s
2000s
2010s

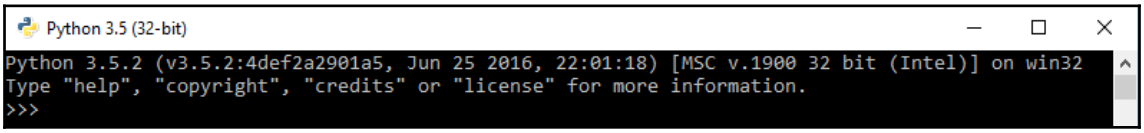

KYD
VOC
ABA

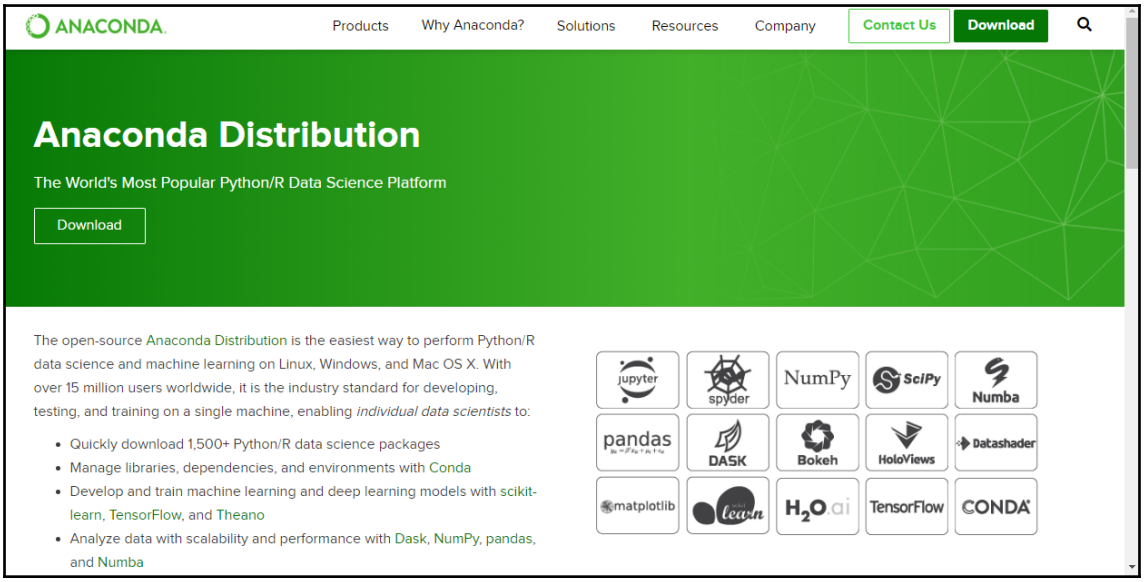

car
car_id pk int
name text
part
part_id pk int
name text
description text
car_part_bridge
car_id fk int
part_id fk int


data
continous (numeric)
categorical (descriptive)
discrete (limited)
interval
ratio
time
nominal (unorderd)
ordinal (ordered)
weight on a scale
any number
8/19/2000 14:30:43
stocks vs. bonds
low to high
numbers on roulette wheel

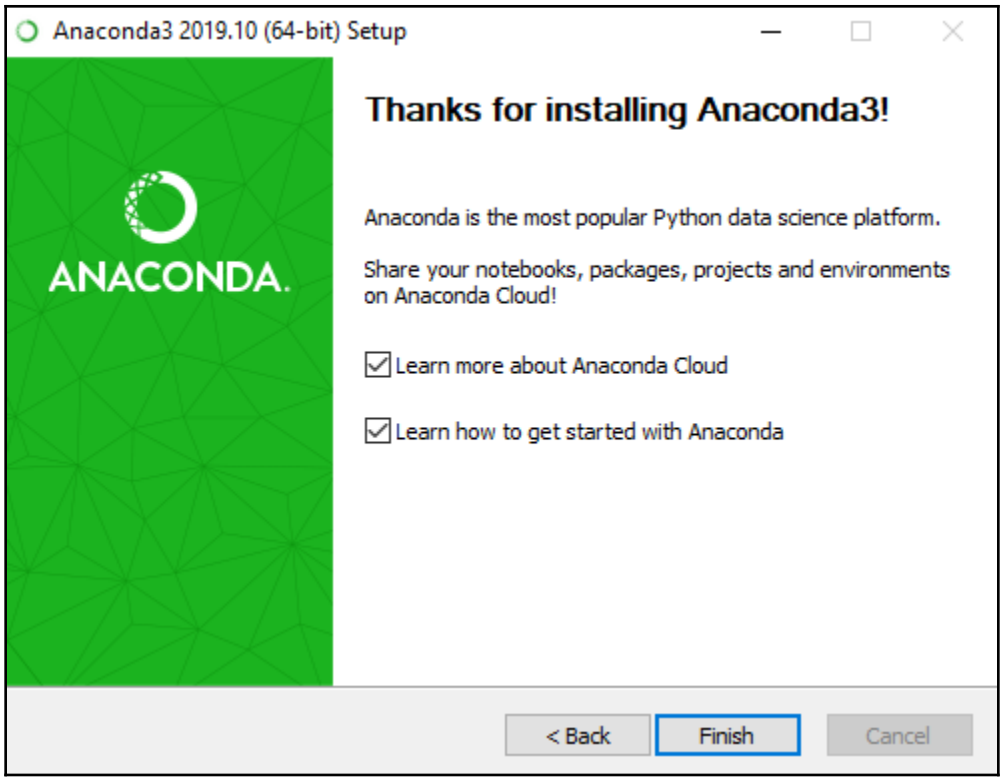

80
60
40
20
0
Avg Temperature (°F)
7:00 AM
8:00 AM
9:00 AM
10:00 AM
11:00 AM
12:00 PM
1:00 PM
2:00 PM
3:00 PM
4:00 PM
5:00 PM
6:00 PM
Hour AM/PM

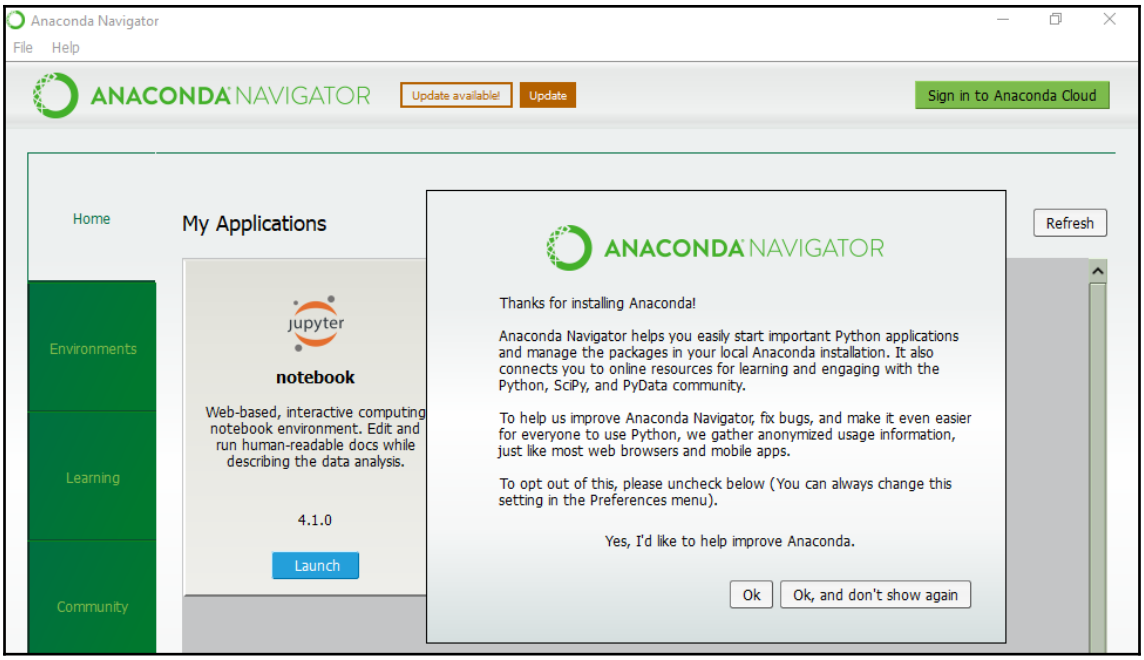
Chapter 2: Overview of Python and Installing Jupyter Notebook

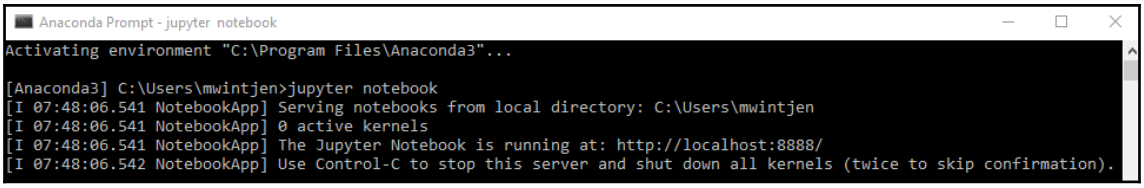


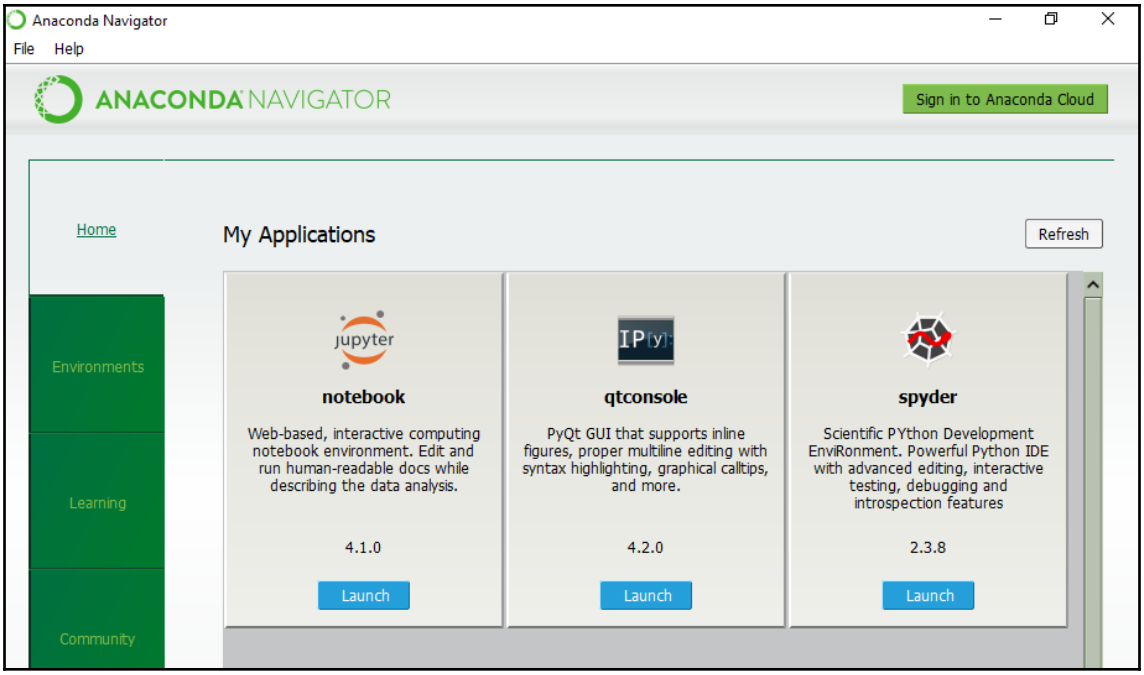


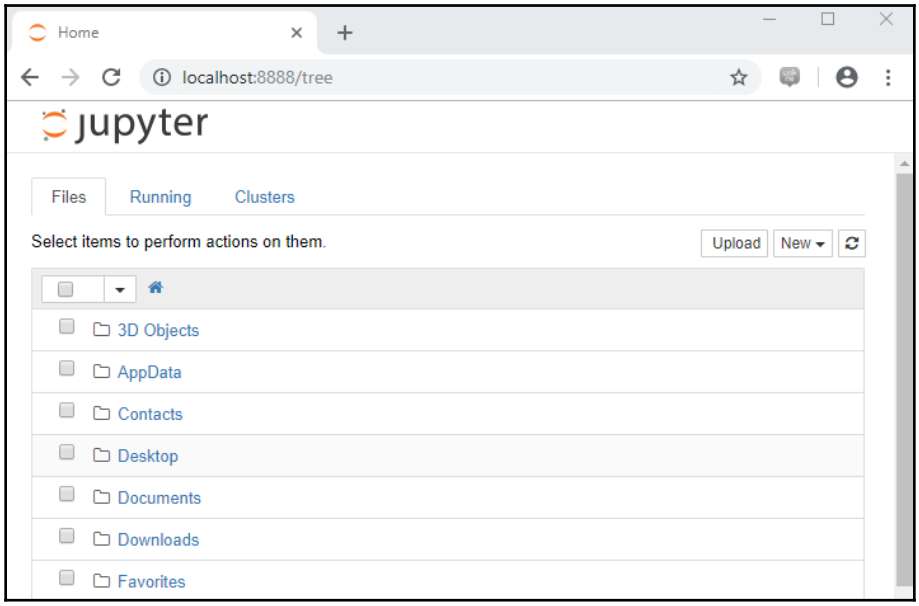

Anaconda3 2019.10 (64-bit) Setup
ANACONDA.
Welcome to Anaconda3 2019.10 (64-bit) Setup
Setup will guide you through the installation of Anaconda3 2019.10 (64-bit).
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel

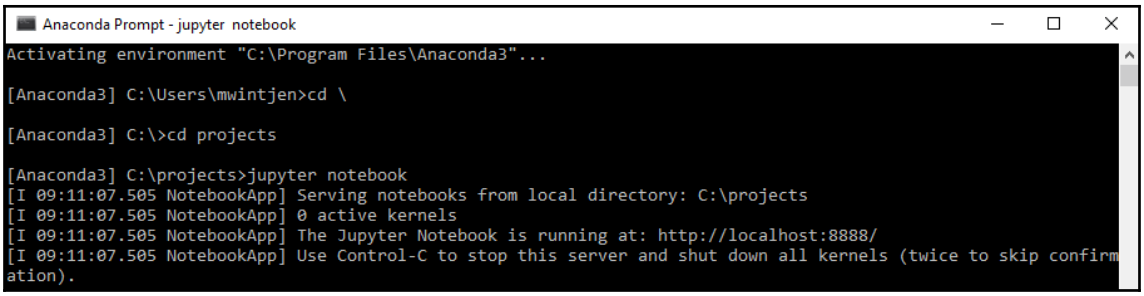

Anaconda3 2019.10 (64-bit) Setup
ANACONDA.
Thanks for installing Anaconda3!
Anaconda is the most popular Python data science platform.
Share your notebooks, packages, projects and environments on Anaconda Cloud!
Learn more about Anaconda Cloud
Learn how to get started with Anaconda
< Back
Finish
Cancel

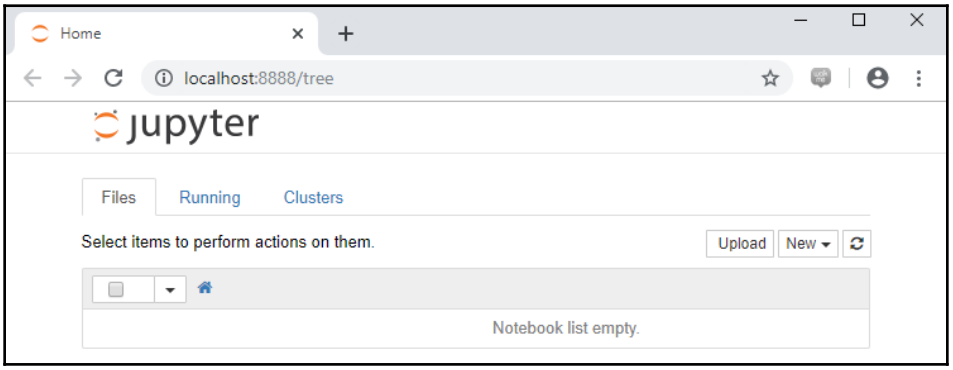

Anaconda Navigator
File Help
ANACONDA NAVIGATOR
Update available!
Update
Sign in to Anaconda Cloud
Home
My Applications
Refresh
Environments
Learning
Community
jupyter
notebook
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
4.1.0
Launch
ANACONDA NAVIGATOR
Thanks for installing Anaconda!
Anaconda Navigator helps you easily start important Python applications and manage the packages in your local Anaconda installation. It also connects you to online resources for learning and engaging with the Python, SciPy, and PyData community.
To help us improve Anaconda Navigator, fix bugs, and make it even easier for everyone to use Python, we gather anonymized usage information, just like most web browsers and mobile apps.
To opt out of this, please uncheck below (You can always change this setting in the Preferences menu).
Yes, I'd like to help improve Anaconda.
Ok
Ok, and don't show again

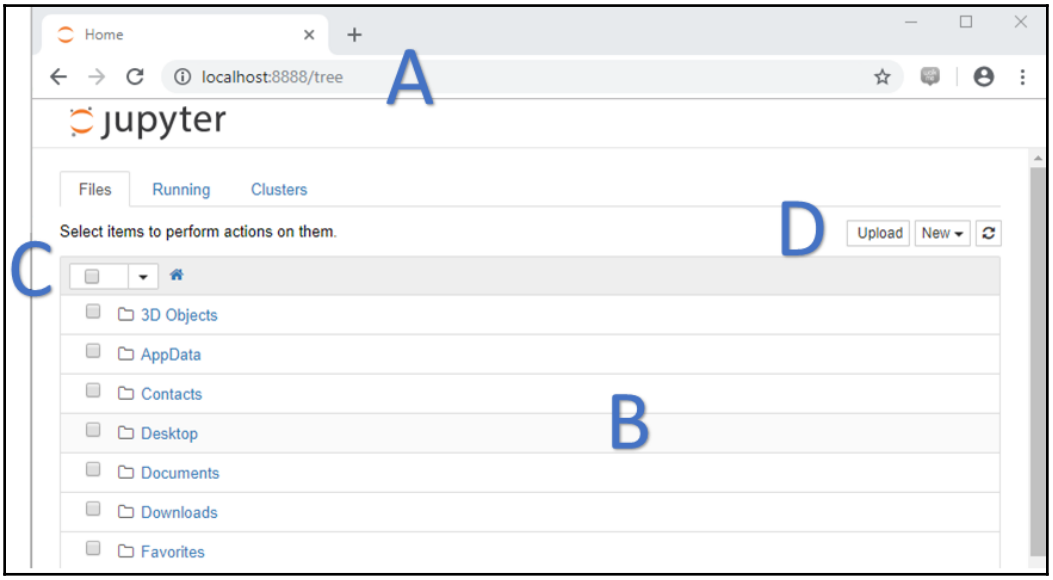

Anaconda Prompt - jupyter notebook
Activating environment "C:\Program Files\Anaconda3"...
[Anaconda3] C:\Users\mwintjen>jupyter notebook
[I 07:48:06.541 NotebookApp] Serving notebooks from local directory: C:\Users\mwintjen
[I 07:48:06.541 NotebookApp] 0 active kernels
[I 07:48:06.541 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/
[I 07:48:06.542 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).

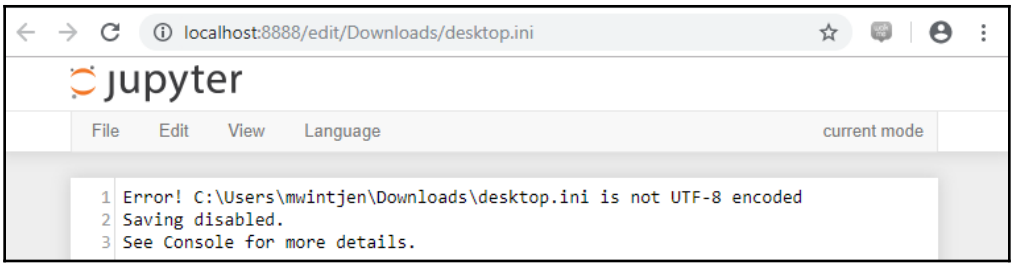

Anaconda Navigator
File Help
ANACONDA NAVIGATOR
Sign in to Anaconda Cloud
Home
My Applications
Refresh
Environments
Learning
Community
notebook
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
4.1.0
Launch
qtconsole
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
4.2.0
Launch
spyder
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
2.3.8
Launch

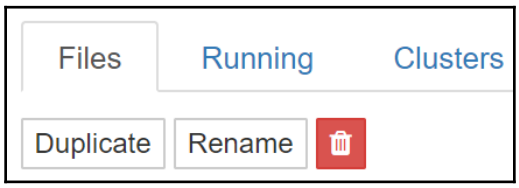

Home
localhost:8888/tree
jupyter
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
3D Objects
AppData
Contacts
Desktop
Documents
Downloads
Favorites

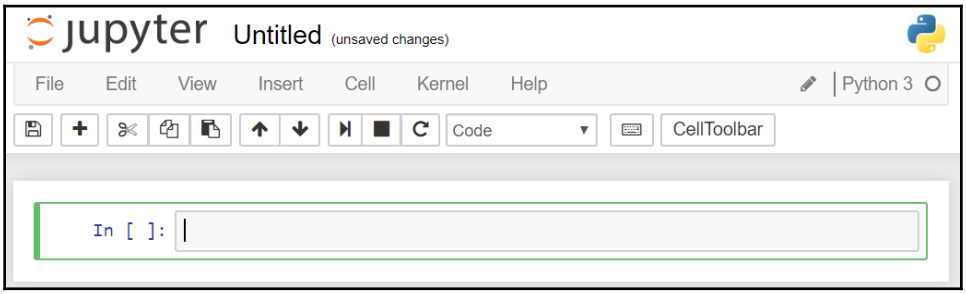

Anaconda Prompt - jupyter notebook
Activating environment "C:\Program Files\Anaconda3"...
[Anaconda3] C:\Users\mwintjen>cd \
[Anaconda3] C:\>cd projects
[Anaconda3] C:\projects>jupyter notebook
[I 09:11:07.505 NotebookApp] Serving notebooks from local directory: C:\projects
[I 09:11:07.505 NotebookApp] 0 active kernels
[I 09:11:07.505 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/
[I 09:11:07.505 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).

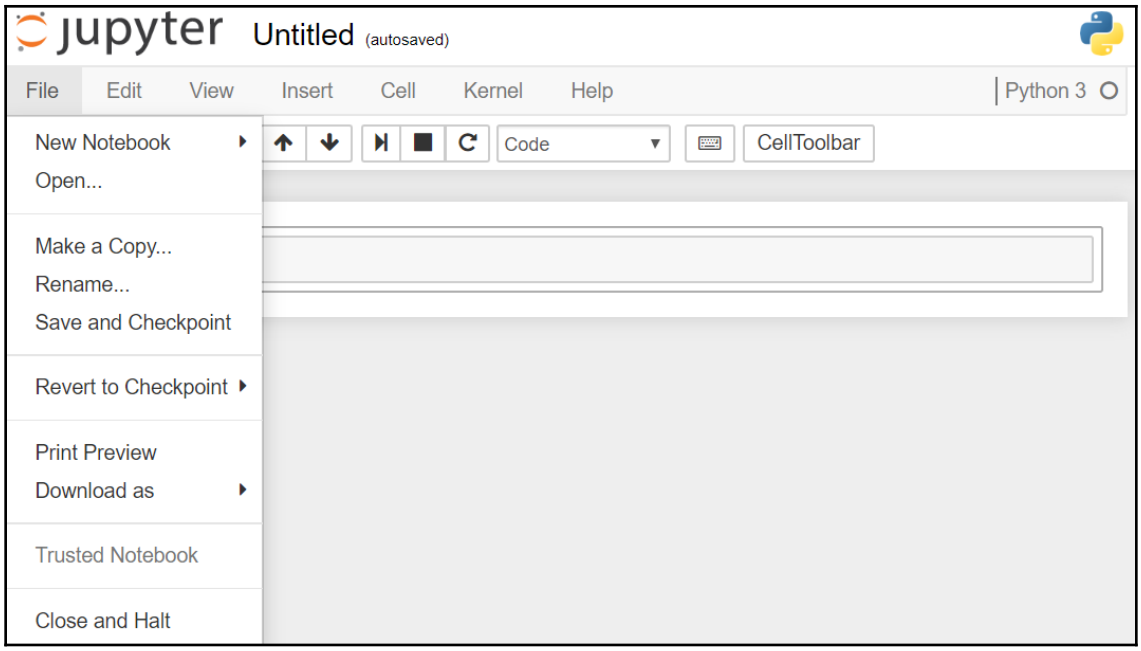

Home
localhost:8888/tree
jupyter
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
Notebook list empty.

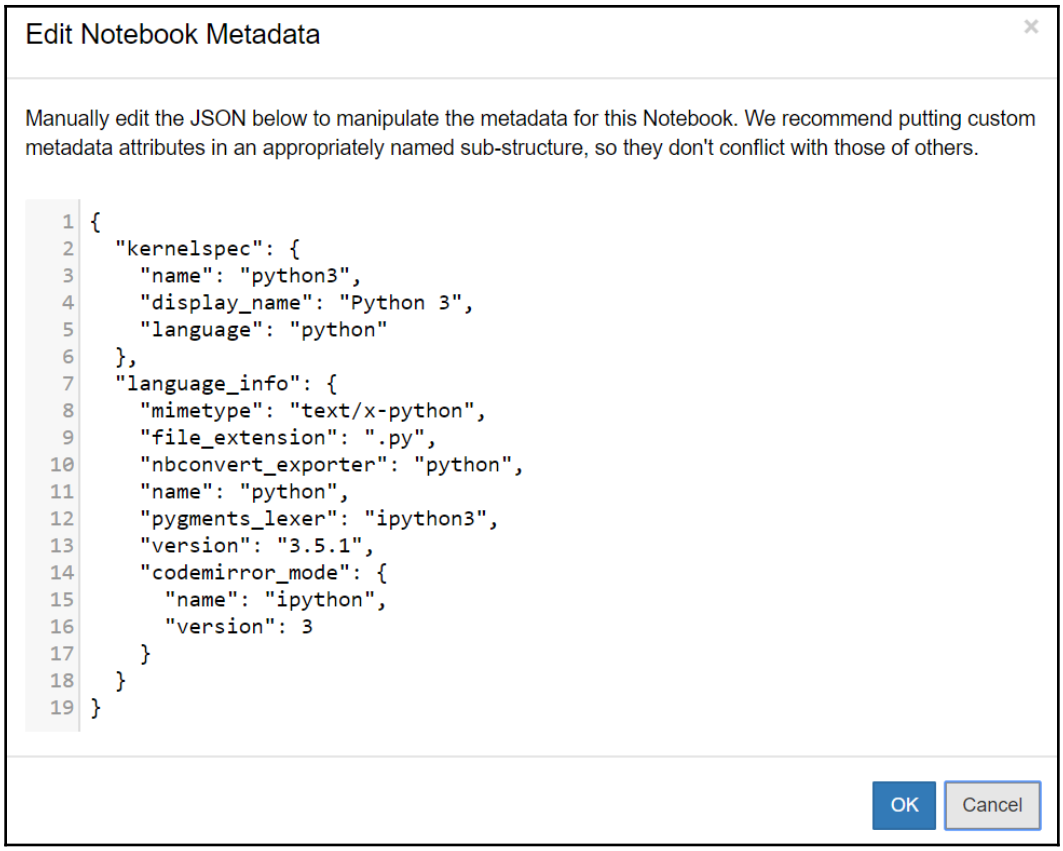

Home
localhost:8888/tree
A
jupyter
Files
Running
Clusters
Select items to perform actions on them.
D
Upload
New
C
3D Objects
AppData
Contacts
Desktop
B
Documents
Downloads
Favorites

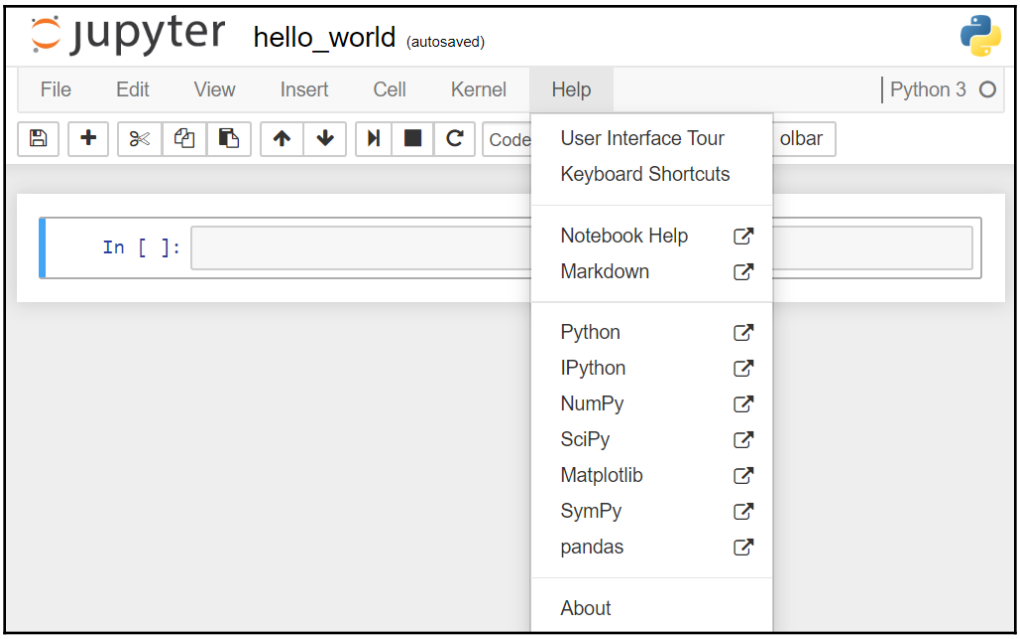

localhost:8888/edit/Downloads/desktop.ini
jupyter
File Edit View Language current mode
1 Error! C:\Users\mwintjen\Downloads\desktop.ini is not UTF-8 encoded
2 Saving disabled.
3 See Console for more details.

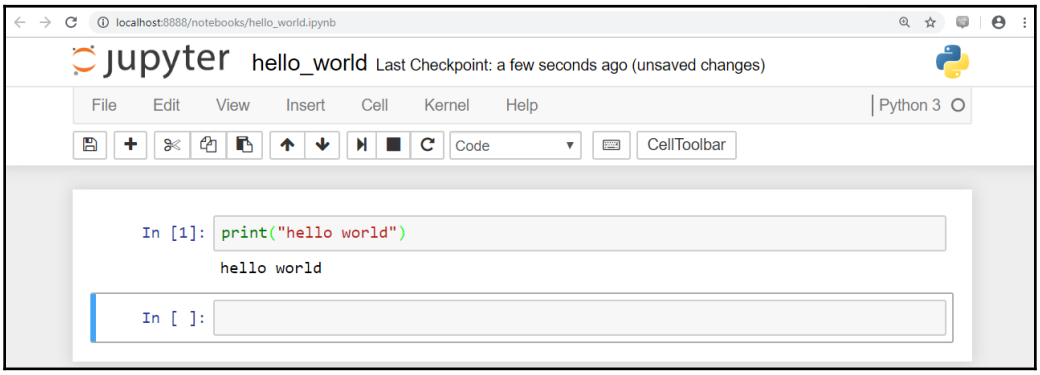

Files
Running
Clusters
Duplicate
Rename


jupyter
Untitled
(unsaved changes)
File Edit View Insert Cell Kernel Help
Python 3
Code
CellToolbar
In [ ]:

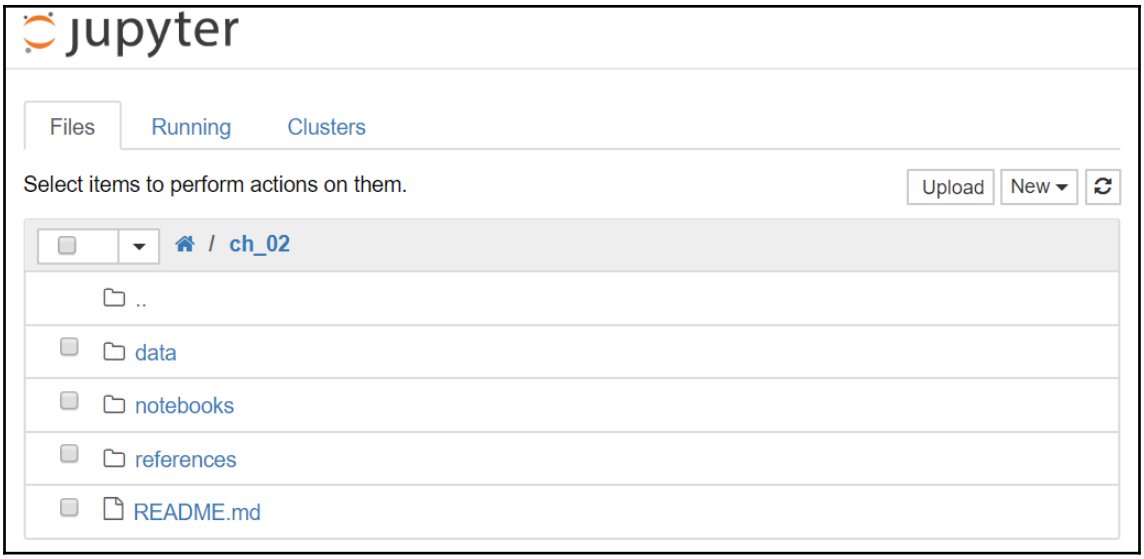

jupyter
Untitled
(autosaved)
File
Edit
View
Insert
Cell
Kernel
Help
Python 3
New Notebook
Open...
Make a Copy...
Rename...
Save and Checkpoint
Revert to Checkpoint
Print Preview
Download as
Trusted Notebook
Close and Halt
Code
CellToolbar

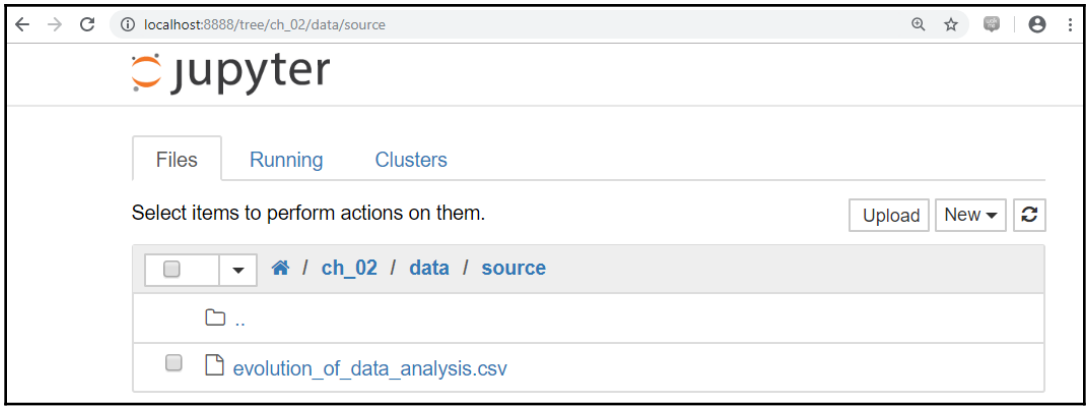

Edit Notebook Metadata
Manually edit the JSON below to manipulate the metadata for this Notebook. We recommend putting custom metadata attributes in an appropriately named sub-structure, so they don't conflict with those of others.
{
"kernelspec": {
"name": "python3",
"display_name": "Python 3",
"language": "python"
},
"language_info": {
"mimetype": "text/x-python",
"file_extension": ".py",
"nbconvert_exporter": "python",
"name": "python",
"pygments_lexer": "ipython3",
"version": "3.5.1",
"codemirror_mode": {
"name": "ipython",
"version": 3
}
}
}
OK
Cancel

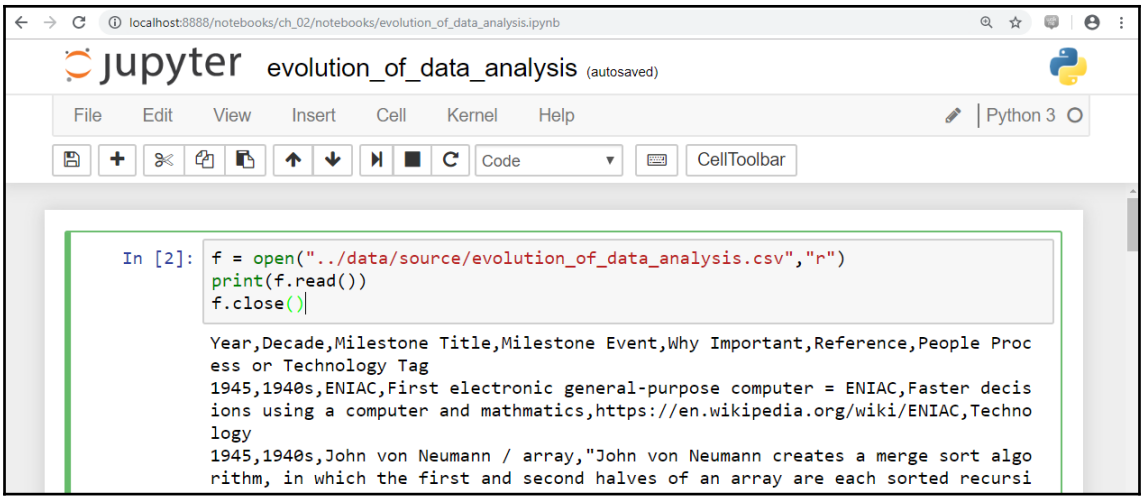


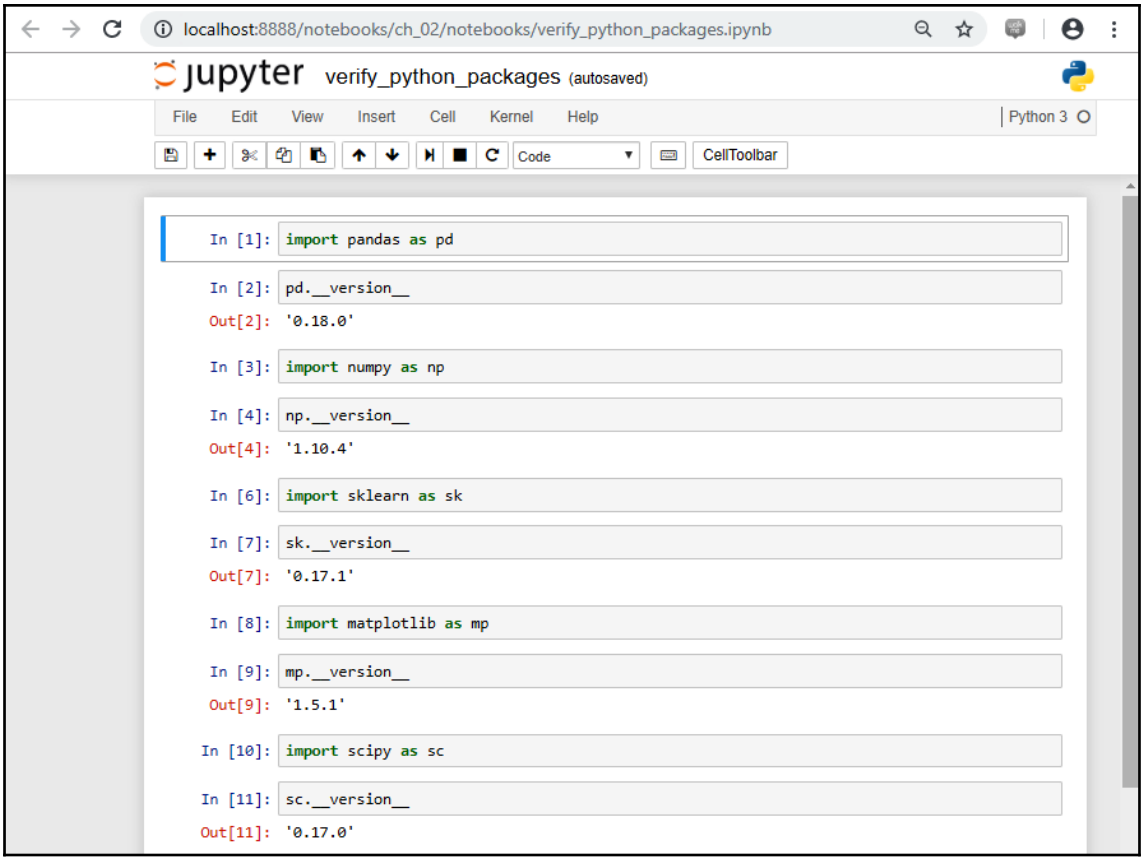



```
projectname
|
├── README.md         <- About the data analysis project and summary information
├── data
|   ├── source       <- Downloaded source raw data files as received
|   ├── wip          <- Work-In-Progress folder used for changes to the source data files
|   └── target       <- Output data files after processed
|
├── notebooks        <- Any Jupyter notebook files
|
└── references       <- Data dictionary, detail project artifacts and supporting information
```

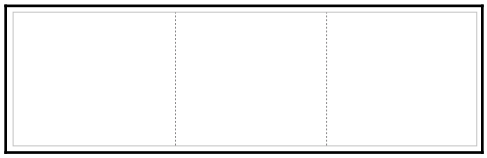

jupyter
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
/ ch_02
..
data
notebooks
references
README.md

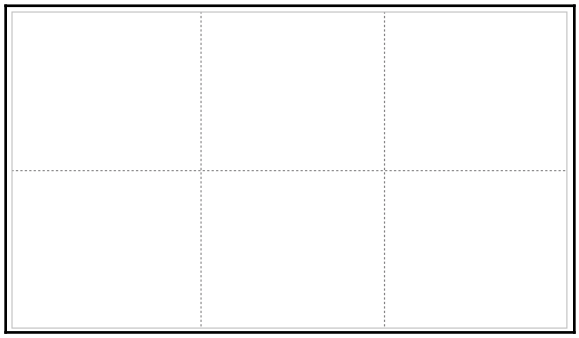

localhost:8888/tree/ch_02/data/source
jupyter
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
/ ch_02 / data / source
..
evolution_of_data_analysis.csv

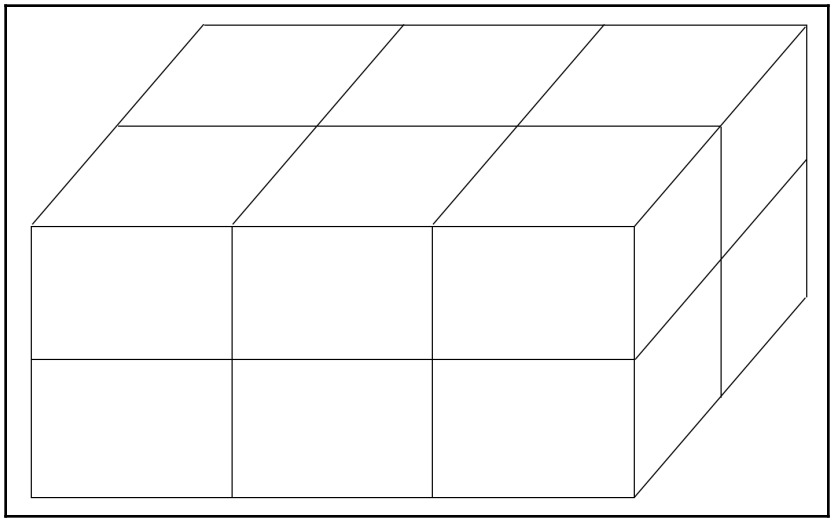

localhost:8888/notebooks/ch_02/notebooks/evolution_of_data_analysis.ipynb
jupyter evolution_of_data_analysis (autosaved)
File Edit View Insert Cell Kernel Help
Python 3
Code
CellToolbar
In [2]:
f = open("../data/source/evolution_of_data_analysis.csv","r")
print(f.read())
f.close()
Year,Decade,Milestone Title,Milestone Event,Why Important,Reference,People Proc
ess or Technology Tag
1945,1940s,ENIAC,First electronic general-purpose computer = ENIAC,Faster decis
ions using a computer and mathmatics,https://en.wikipedia.org/wiki/ENIAC,Techno
logy
1945,1940s,John von Neumann / array,"John von Neumann creates a merge sort algo
rithm, in which the first and second halves of an array are each sorted recursi


localhost:8888/notebooks/ch_02/notebooks/verify_python_packages.ipynb
jupyter verify_python_packages (autosaved)
File Edit View Insert Cell Kernel Help
Python 3
Code
CellToolbar
In [1]: import pandas as pd
In [2]: pd.__version__
Out[2]: '0.18.0'
In [3]: import numpy as np
In [4]: np.__version__
Out[4]: '1.10.4'
In [6]: import sklearn as sk
In [7]: sk.__version__
Out[7]: '0.17.1'
In [8]: import matplotlib as mp
In [9]: mp.__version__
Out[9]: '1.5.1'
In [10]: import scipy as sc
In [11]: sc.__version__
Out[11]: '0.17.0'

Chapter 3: Getting Started with NumPy







```
In [1]: import numpy as np

In [2]: my_first_array = np.array([1,2,3])

In [3]: print(my_first_array)
        [1 2 3]
```

```
In [1]: import numpy as np

In [2]: my_first_array = np.array([1,2,3])

In [3]: print(my_first_array)
        [1 2 3]

In [4]: my_first_array.shape
Out[4]: (3,)

In [5]: my_first_array.size
Out[5]: 3

In [6]: len(my_first_array)
Out[6]: 3

In [7]: my_first_array.dtype.name
Out[7]: 'int64'

In [8]: my_first_array.astype(int)
Out[8]: array([1, 2, 3])
```

```
In [9]: print(my_first_array[0])
        1
```

```
In [11]: my_first_array.sum()
Out[11]: 6

In [12]: my_first_array.min()
Out[12]: 1

In [13]: my_first_array.max()
Out[13]: 3

In [14]: my_first_array.mean()
Out[14]: 2.0
```

```
Date,Close
2019-01-02,157.919998
2019-01-03,142.190002
2019-01-04,148.259995
2019-01-07,147.929993
2019-01-08,150.75
2019-01-09,153.309998
2019-01-10,153.800003
2019-01-11,152.289993
2019-01-14,150
2019-01-15,153.070007
2019-01-16,154.940002
2019-01-17,155.860001
2019-01-18,156.820007
2019-01-22,153.300003
2019-01-23,153.919998
2019-01-24,152.699997
2019-01-25,157.759995
2019-01-28,156.300003
2019-01-29,154.679993
```

```
In [1]: import numpy as np

In [2]: input_stock_price_array = np.array([142.19,148.26,147.93,150.75,153.31,153.8,152.28,150,153.07,154.94])

In [10]: sorted_stock_price_array = np.sort(input_stock_price_array)[::-1]

In [16]: print('Closing stock price in order of day traded:    ', input_stock_price_array)
         print('Closing stock price in order from high to low: ', sorted_stock_price_array)

         Closing stock price in order of day traded:     [142.19 148.26 147.93 150.75 153.31 153.8  152.28 150.   153.07 154.94]
         Closing stock price in order from high to low:  [154.94 153.8  153.31 153.07 152.28 150.75 150.   148.26 147.93 142.19]

In [17]: print('Highest closing stock price: ', sorted_stock_price_array[0])

         Highest closing stock price:  154.94
```

```
In [1]: import numpy as np

In [8]: input_stock_price_array = np.genfromtxt('AAPL_stock_price_example.csv', delimiter=',', names=True, usecols = (1))

In [20]: input_stock_price_array.size

Out[20]: 229

In [10]: sorted_stock_price_array = np.sort(input_stock_price_array)[::-1]

In [18]: print('Closing stock price in order of day traded:    ', input_stock_price_array[:5])
         print('Closing stock price in order from high to low: ', sorted_stock_price_array[:5])

         Closing stock price in order of day traded:     [(157.919998,) (142.190002,) (148.259995,) (147.929993,) (150.75    ,)]
         Closing stock price in order from high to low:  [(267.100006,) (266.369995,) (266.290009,) (265.76001 ,) (264.470001,)]

In [19]: print('Highest closing stock price: ', sorted_stock_price_array[0])

         Highest closing stock price:  (267.100006,)
```

```
In [23]: print(temp_array[:5])
         ['Close', '157.919998', '142.190002', '148.259995', '147.929993']
```

```
In [26]: temp_array.size

Out[26]: 229
```

```
In [32]: print(input_stock_price_array[:5])
         [ 157.919998  142.190002  148.259995  147.929993  150.75    ]
```

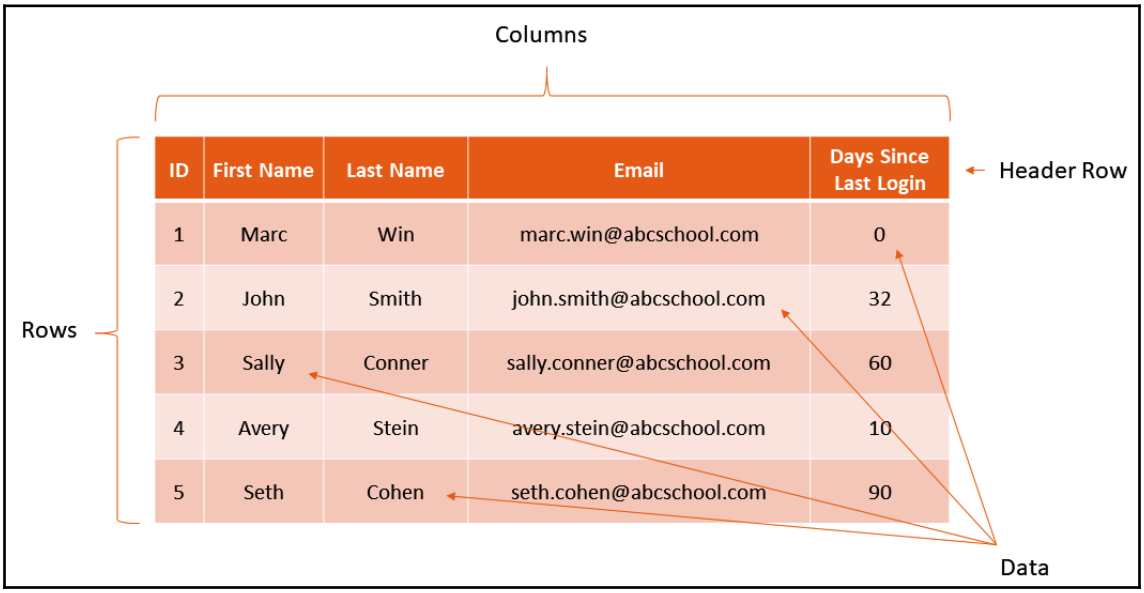
```
In [30]: print('Closing stock price in order of day traded:    ', input_stock_price_array[:5])
         print('Closing stock price in order from high to low: ', sorted_stock_price_array[:5])

         Closing stock price in order of day traded:     [ 157.919998  142.190002  148.259995  147.929993  150.75    ]
         Closing stock price in order from high to low:  [ 267.100006  266.369995  266.290009  265.76001   264.470001]
```

```
In [31]: print('Highest closing stock price: ', sorted_stock_price_array[0])

         Highest closing stock price:  267.100006
```

Chapter 4: Creating Your First pandas DataFrame



	product a	project b	project c
0	13	10	6
1	20	30	9
2	0	17	10
3	10	20	0

	product a	project b	project c
Ronny	13	10	6
Bobby	20	30	9
Ricky	0	17	10
Mike	10	20	0

```
product a    13
project b    10
project c     6
Name: Ronny, dtype: int64
```

```
Year,Decade,Milestone Title
1945,1940s,ENIAC,First elec
1945,1940s,John von Neumann
1956,1950s,First Hard Disk,
1958,1950s,Modem,The modem
1961,1960s,John Tukey / dat
1970,1970s,Dr. EF Codd / no
1972,1970s,FORTRAN supports
1982,1980s,Mike Bloomberg /
1983,1980s,Edward Tufty / D
```

Comma Delimited

```
Year    Decade  Milestone Title
1945    1940s   ENIAC   First e
1945    1940s   John von Neuman
1956    1950s   First Hard Disk
1958    1950s   Modem   The mod
1961    1960s   John Tukey / da
1970    1970s   Dr. EF Codd / n
1972    1970s   FORTRAN support
1982    1980s   Mike Bloomberg
1983    1980s   Edward Tufty /
```

Tab Delimited

```
Year|Decade|Milestone Title
1945|1940s|ENIAC|First elec
1945|1940s|John von Neumann
1956|1950s|First Hard Disk|
1958|1950s|Modem|The modem
1961|1960s|John Tukey / dat
1970|1970s|Dr. EF Codd / no
1972|1970s|FORTRAN supports
1982|1980s|Mike Bloomberg /
1983|1980s|Edward Tufty / D
```

Pipe Delimited

```
<?xml version="1.0" encoding="UTF-8" ?>
<Evolution_of_Data_Milestone="ENIAC">
    <Year>1945</Year>
    <Decade>1940s</Decade>
    <Milestone_Title>ENIAC</Milestone_Title>
    <Milestone_Event>First electronic general-purpose computer = ENIAC</Mil
    <Why_Important>Faster decisions using a computer and mathmatics</Why_Im
    <Reference>https://en.wikipedia.org/wiki/ENIAC</Reference>
    <People_Process_or_Technology_Tag>Technology</People_Process_or_Technol
</Evolution_of_Data_Milestone>
<Evolution_of_Data_Milestone="John von Neumann / array">
    <Year>1945</Year>
    <Decade>1940s</Decade>
    <Milestone_Title>John von Neumann / array</Milestone_Title>
    <Milestone_Event>John von Neumann creates a merge sort algorithm, in wh
    <Why_Important>Arrays allow mathmatics, grouping and sorting of data</W
    <Reference>https://en.wikipedia.org/wiki/John_von_Neumann</Reference>
    <People_Process_or_Technology_Tag>People</People_Process_or_Technology_
</Evolution_of_Data_Milestone>
```

```
[
  {
    "Year": 1945,
    "Decade": "1940s",
    "Milestone Title": "ENIAC",
    "Milestone Event": "First electronic general-purpose computer = ENIAC",
    "Why Important": "Faster decisions using a computer and mathmatics",
    "Reference": "https://en.wikipedia.org/wiki/ENIAC",
    "People Process or Technology Tag": "Technology"
  },
  {
    "Year": 1945,
    "Decade": "1940s",
    "Milestone Title": "John von Neumann / array",
    "Milestone Event": "John von Neumann creates a merge sort algorithm, in which th
    "Why Important": "Arrays allow mathmatics, grouping and sorting of data",
    "Reference": "https://en.wikipedia.org/wiki/John_von_Neumann",
    "People Process or Technology Tag": "People"
  },
```


```
In [1]: import pandas as pd

In [15]: my_df = pd.read_csv('evolution_of_data_analysis.csv', header=0, sep="|")

In [16]: my_df.shape
Out[16]: (42, 7)
```

```
In [17]: my_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42 entries, 0 to 41
Data columns (total 7 columns):
Year                                42 non-null int64
Decade                              42 non-null object
Milestone Title                     42 non-null object
Milestone Event                     42 non-null object
Why Important                       42 non-null object
Reference                           39 non-null object
People Process or Technology Tag    42 non-null object
dtypes: int64(1), object(6)
memory usage: 2.4+ KB
```

```
In [20]: my_df.describe()
```

Out[20]:

	Year
count	42.000000
mean	1993.833333
std	18.876901
min	1945.000000
25%	1987.500000
50%	1999.000000
75%	2006.750000
max	2018.000000

```
In [23]: my_df.head(2)
```

Out[23]:

	Year	Decade	Milestone Title	Milestone Event	Why Important	Reference	People Process or Technology Tag
0	1945	1940s	ENIAC	First electronic general-purpose computer = ENIAC	Faster decisions using a computer and mathmatics	https://en.wikipedia.org/wiki/ENIAC	Technology
1	1945	1940s	John von Neumann / array	John von Neumann creates a merge sort algorith...	Arrays allow mathmatics| grouping and sorting ...	https://en.wikipedia.org/wiki/John_von_Neumann	People

```
In [7]: my_df.tail(2)
```

Out[7]:

	Year	Decade	Milestone Title	Milestone Event	Why Important	Reference	People Process or Technology Tag
40	2015	2010s	Alexa	Amazon Alexa Ecosystem	Making voice to data mainstream and allowing A...	https://www.zdnet.com/article/technology-that-...	Technology
41	2018	2010s	The Data Literacy Project	The Data Literacy Project launches	Building a data driven culture where anyone ca...	https://www.businesswire.com/news/home/2018101...	Process

```
In [8]: my_df.sort_index(1)
```

Out[8]:

	Decade	Milestone Event	Milestone Title	People Process or Technology Tag	Reference	Why Important	Year
0	1940s	First electronic general-purpose computer = ENIAC	ENIAC	Technology	https://en.wikipedia.org/wiki/ENIAC	Faster decisions using a computer and mathmatics	1945
1	1940s	John von Neumann creates a merge sort algorith...	John von Neumann / array	People	https://en.wikipedia.org/wiki/John_von_Neumann	Arrays allow mathmatics| grouping and sorting ...	1945
2	1950s	The first computer hard disk used	First Hard Disk	Technology	https://fiftiesweb.com/pop/inventions/	Storing data for reuse = reduces time to recre...	1956

```
In [9]: import pandas as pd
```

```
In [10]: my_df = pd.read_csv('evolution_of_data_analysis.csv', header=0, sep="|")
```

```
In [11]: my_df.head(2)
```

Out[11]:

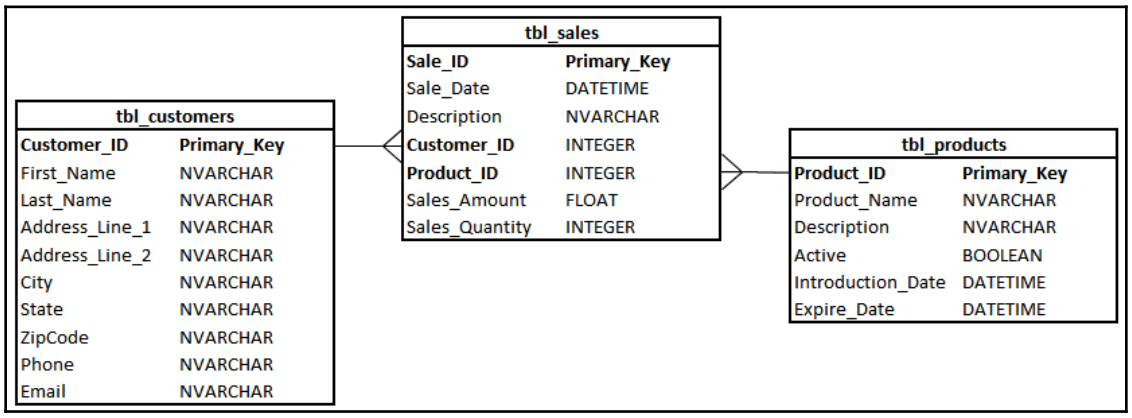
	Year	Decade	Milestone Title	Milestone Event	Why Important	Reference	People Process or Technology Tag
0	1945	1940s	ENIAC	First electronic general-purpose computer = ENIAC	Faster decisions using a computer and mathmatics	https://en.wikipedia.org/wiki/ENIAC	Technology
1	1945	1940s	John von Neumann / array	John von Neumann creates a merge sort algorith...	Arrays allow mathmatics| grouping and sorting ...	https://en.wikipedia.org/wiki/John_von_Neumann	People

```
In [12]: my_df.groupby(['Decade']).agg({'Milestone Event':'count'})
```

Out[12]:

	Milestone Event
Decade	
1940s	2
1950s	2
1960s	1
1970s	2
1980s	5
1990s	9
2000s	14
2010s	7

Chapter 5: Gathering and Loading Data in Python




```
In [13]: df_sales.head()
```

Out[13]:

	Sale_ID	Sale_Date	Description	Customer_ID	Product_ID	Sales_Amount	Sales_Quantity
0	1	12/31/2014	Purchased from Store	2	2	20	1
1	2	1/15/2015	Phone Purchase	1	1	30	2
2	3	6/14/2015	Internet Purchase	3	3	5	1
3	4	11/11/2015	Sales Convention Purchase	3	3	500	100
4	5	4/18/2016	Internet Purchase	4	1	20	2

```
In [14]: df_sales.sort_values(by='Sale_Date')
```

Out[14]:

	Sale_ID	Sale_Date	Description	Customer_ID	Product_ID	Sales_Amount	Sales_Quantity
1	2	1/15/2015	Phone Purchase	1	1	30	2
5	6	10/15/2016	Purchased from Store	5	1	20	1
3	4	11/11/2015	Sales Convention Purchase	3	3	500	100
0	1	12/31/2014	Purchased from Store	2	2	20	1
6	7	3/17/2017	Internet Purchase	4	1	20	1
4	5	4/18/2016	Internet Purchase	4	1	20	2
8	9	5/25/2019	Internet Purchase	1	3	10	2
2	3	6/14/2015	Internet Purchase	3	3	5	1
7	8	6/15/2018	Purchased from Store	3	3	5	1
9	10	6/9/2019	Internet Purchase	2	3	10	2

```
In [61]: df_sales.loc[0]

Out[61]: Sale_ID                1
         Sale_Date         12/31/2014
         Description    Purchased from Store
         Customer_ID                2
         Product_ID                 2
         Sales_Amount              20
         Sales_Quantity             1
         Name: 0, dtype: object
```

```
In [52]: df_sales[(df_sales['Sales_Amount'] > 100)]
```

Out[52]:

	Sale_ID	Sale_Date	Description	Customer_ID	Product_ID	Sales_Amount	Sales_Quantity
3	4	11/11/2015	Sales Convention Purchase	3	3	500	100

```
In [57]: df_sales[(df_sales['Sales_Quantity'] == 1)]
```

Out[57]:

	Sale_ID	Sale_Date	Description	Customer_ID	Product_ID	Sales_Amount	Sales_Quantity
0	1	12/31/2014	Purchased from Store	2	2	20	1
2	3	6/14/2015	Internet Purchase	3	3	5	1
5	6	10/15/2016	Purchased from Store	5	1	20	1
6	7	3/17/2017	Internet Purchase	4	1	20	1
7	8	6/15/2018	Purchased from Store	3	3	5	1

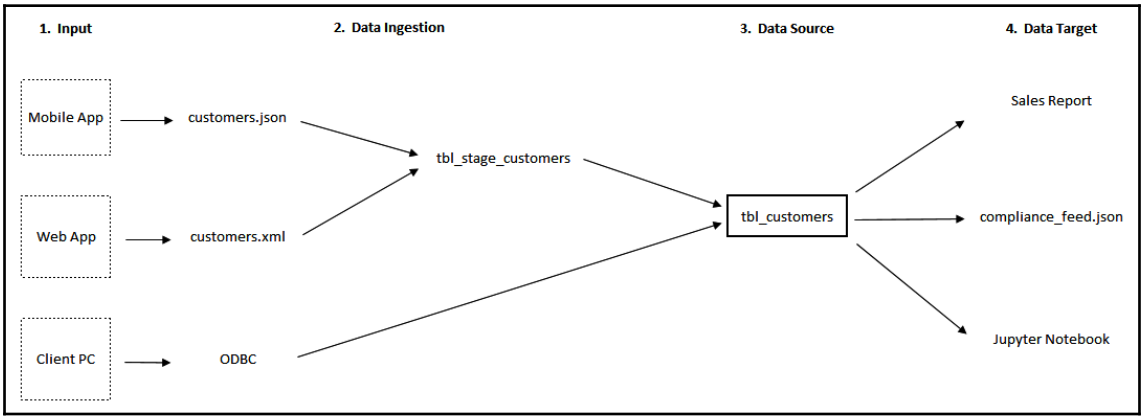
Out[10]:

	Customer_ID	First_Name	Last_Name	Address_Line_1	Address_Line_2	City	State	ZipCode	Phone	Email
0	1	Johnny	Smith	123 Main Street	None	Miami	FL	12345	302-555-1212	jsmith@email.com
1	2	Debbie	Winner	31 Roundtree Lane	None	Dover	NJ	18888	None	debbie_winner@email.com
2	3	Seth	Winer	310 Roundtree Lane	None	Dover	NJ	18888	None	sw@email.com
3	4	Anthony	Leedessa	Dallas Drive	Unit 806	El Paso	TX	99928	None	alligator@email.com
4	5	Pete	Einstein	Morton Street	None	Philadelphia	PA	28373	215-555-1212	peter_einstein@email.com

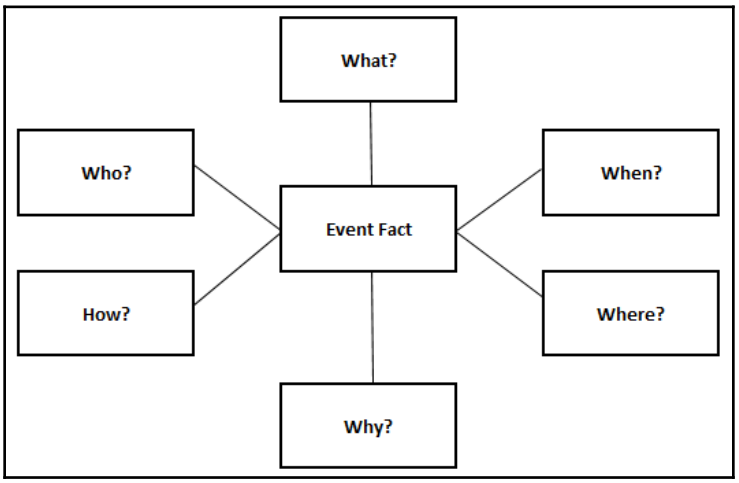
In [12]:
```
pd.isnull(df_customers)
```

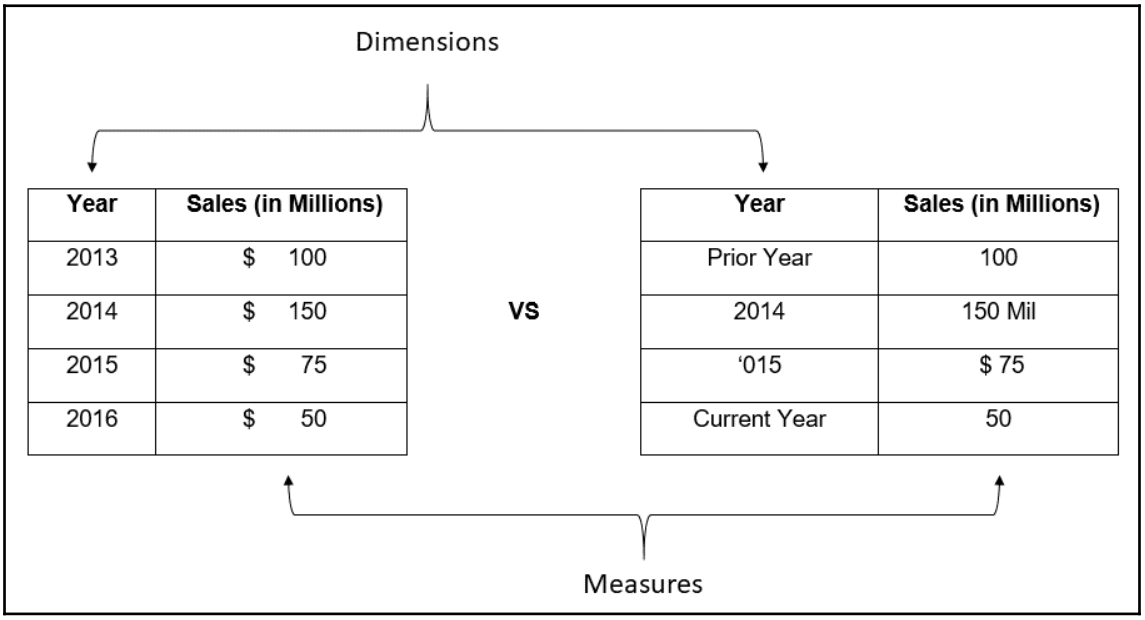
Out[12]:

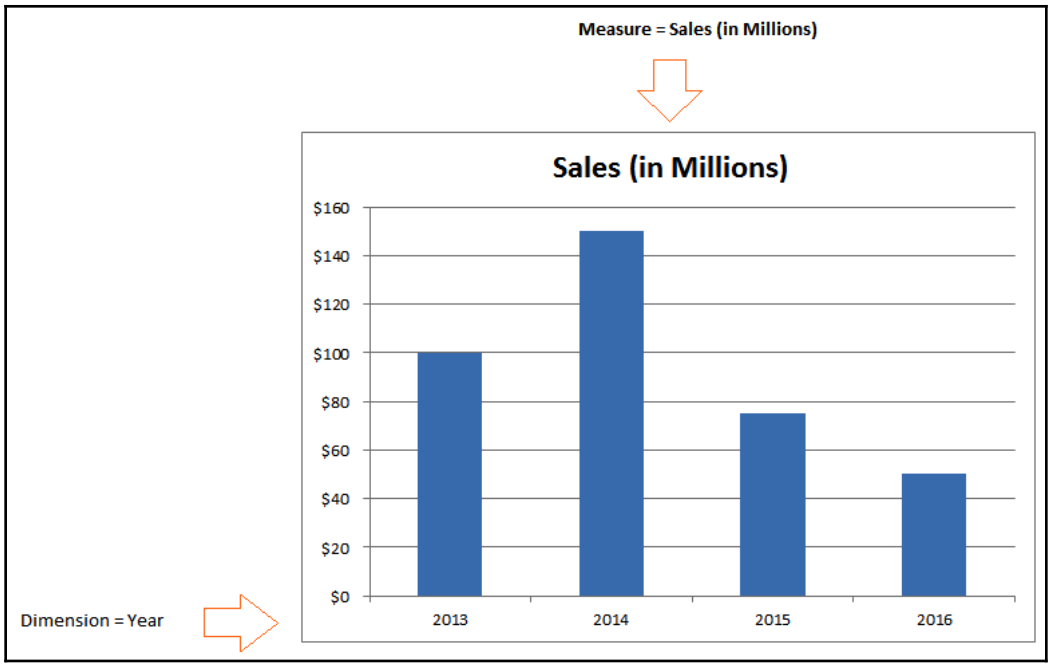
	Customer_ID	First_Name	Last_Name	Address_Line_1	Address_Line_2	City	State	ZipCode	Phone	Email
0	False	False	False	False	True	False	False	False	False	False
1	False	False	False	False	True	False	False	False	True	False
2	False	False	False	False	True	False	False	False	True	False
3	False	False	False	False	False	False	False	False	True	False
4	False	False	False	False	True	False	False	False	False	False

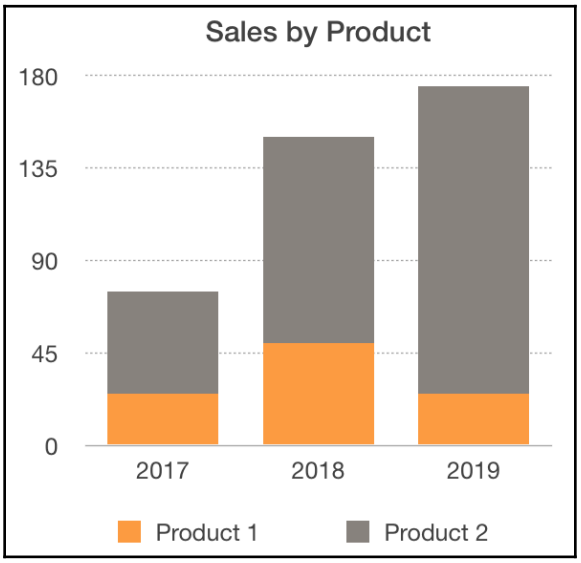


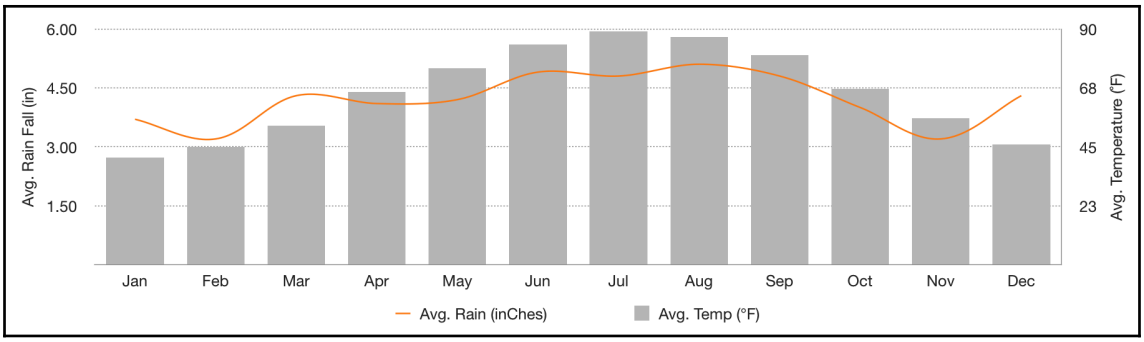
Chapter 6: Visualizing and Working with Time Series Data

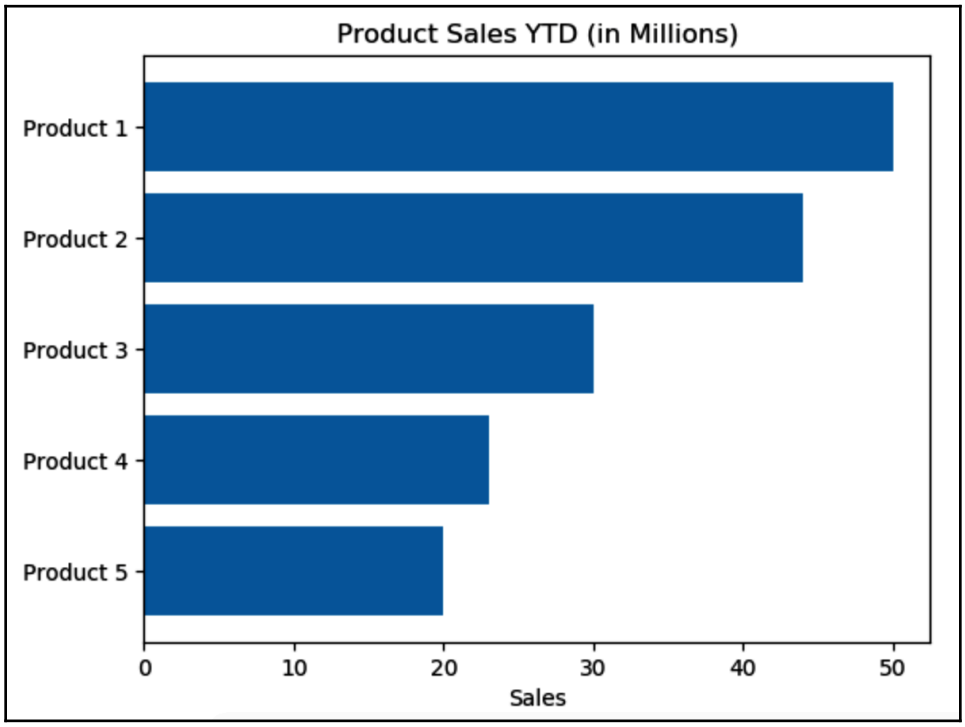


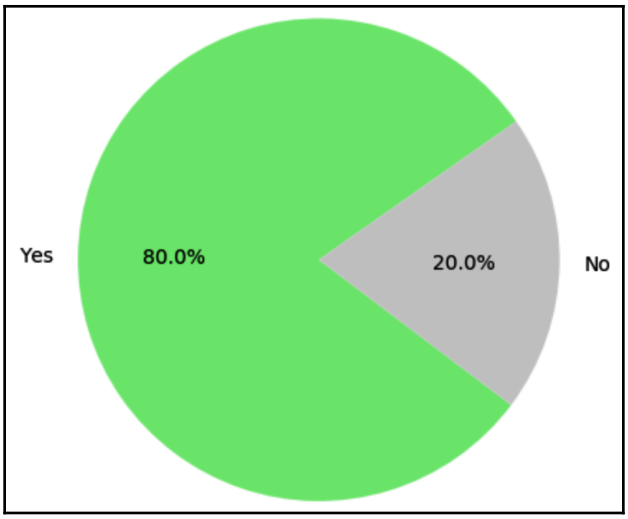


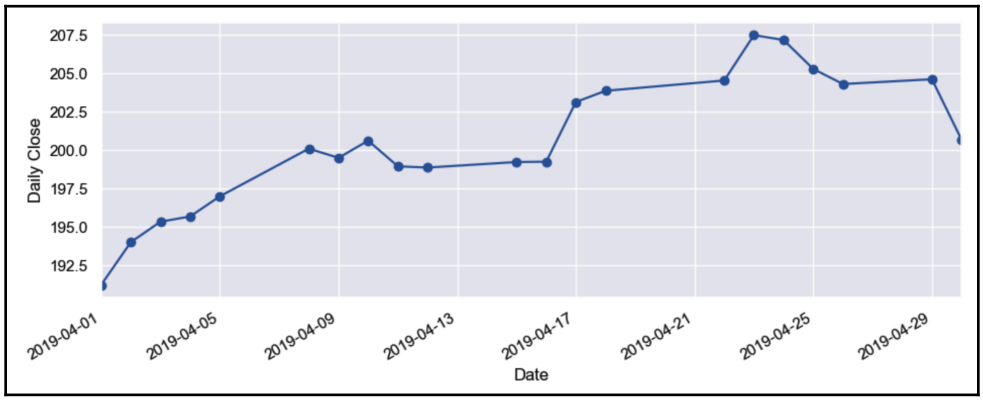

Measure = Sales (in Millions)
Sales (in Millions)
$160
$140
$120
$100
$80
$60
$40
$20
$0
2013
2014
2015
2016
Dimension = Year

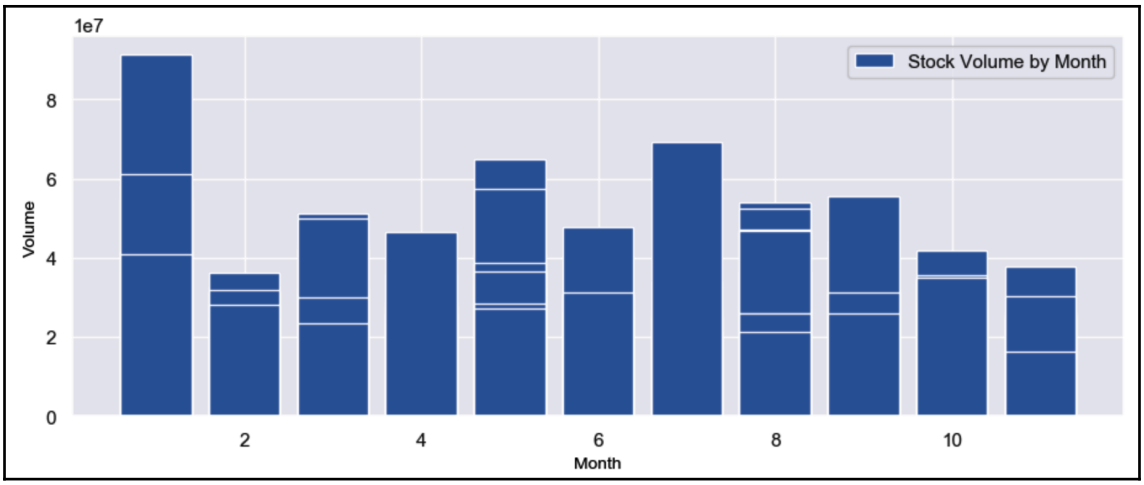

Sales by Product
180
135
90
45
0
2017
2018
2019
Product 1
Product 2

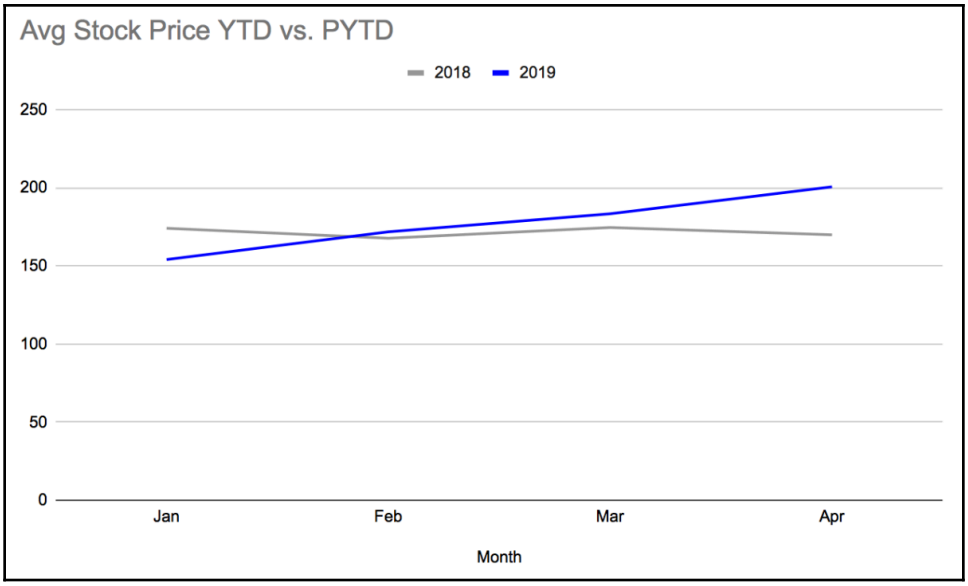

6.00
4.50
3.00
1.50
Avg. Rain Fall (in)
90
68
45
23
Avg. Temperature (°F)
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Avg. Rain (inChes)
Avg. Temp (°F)


Product Sales YTD (in Millions)
Product 1
Product 2
Product 3
Product 4
Product 5
0
10
20
30
40
50
Sales


Yes
80.0%
20.0%
No


Daily Close
207.5
205.0
202.5
200.0
197.5
195.0
192.5
2019-04-01
2019-04-05
2019-04-09
2019-04-13
2019-04-17
2019-04-21
2019-04-25
2019-04-29
Date





Month	2018	2019
Jan	174.01	154.00
Feb	167.64	171.73
Mar	174.50	183.29
Apr	169.83	200.52

```
[3]: df_stock_price.head(3)
```

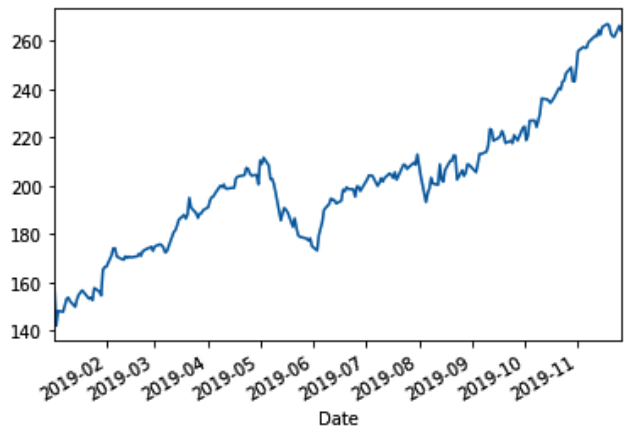
[3]:

Date	Open	High	Low	Close	Adj Close	Volume
2019-01-02	154.889999	158.850006	154.229996	157.919998	155.582367	37039700
2019-01-03	143.979996	145.720001	142.000000	142.190002	140.085220	91312200
2019-01-04	144.529999	148.550003	143.800003	148.259995	146.065353	58607100

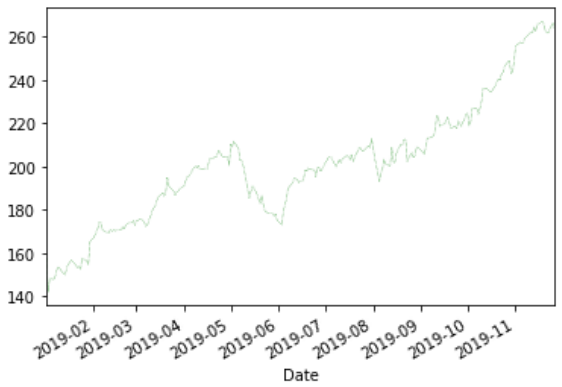
```
[4]: df_stock_price.index
```

```
[4]: DatetimeIndex(['2019-01-02', '2019-01-03', '2019-01-04', '2019-01-07',
                    '2019-01-08', '2019-01-09', '2019-01-10', '2019-01-11',
                    '2019-01-14', '2019-01-15',
                    ...
                    '2019-11-13', '2019-11-14', '2019-11-15', '2019-11-18',
                    '2019-11-19', '2019-11-20', '2019-11-21', '2019-11-22',
                    '2019-11-25', '2019-11-26'],
                   dtype='datetime64[ns]', name='Date', length=229, freq=None)
```

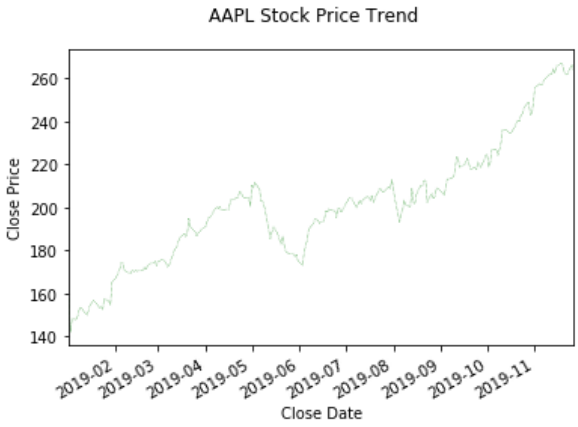
```
[34]: df_stock_price['Close'].plot();
```



```
[44]: df_stock_price['Close'].plot(linewidth=.5, color='green',linestyle='dotted');
```



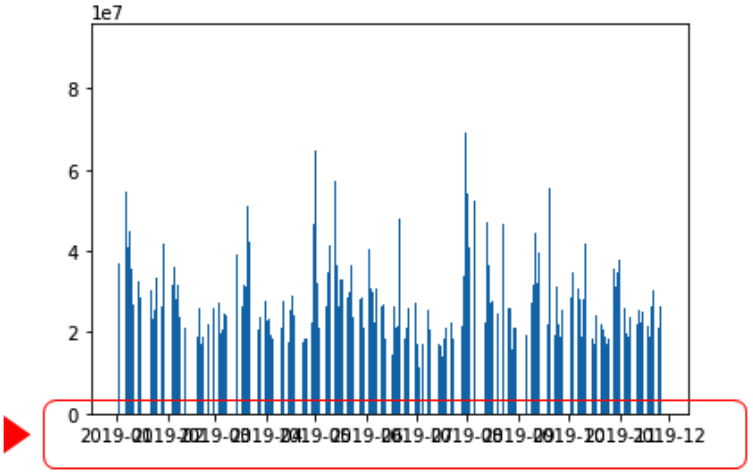

```
[47]: df_stock_price['Close'].plot(linewidth=.5, color='green',linestyle='dotted');
      plt.xlabel('Close Date')
      plt.ylabel('Close Price')
      plt.suptitle('AAPL Stock Price Trend');
```



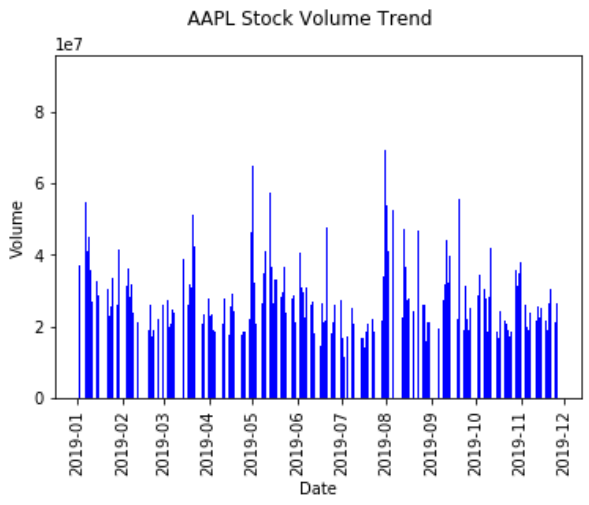
```
[80]: plt.bar(df_stock_price.index,df_stock_price['Volume']);
```

Notice the overlap on the x-axis.

This is the default behavior we will fix in the next step.




```
[86]: plt.bar(df_stock_price.index,df_stock_price['Volume'], color='blue')
      plt.xticks(rotation = 90)
      plt.yticks(fontsize = 10)
      plt.xlabel('Date')
      plt.ylabel('Volume')
      plt.suptitle('AAPL Stock Volume Trend');
```



Chapter 7: Exploring, Cleaning, Refining, and Blending Datasets

```
In [8]: import sqlite3

In [9]: conn = sqlite3.connect('user_hits.db')

In [10]: import pandas as pd

In [11]: df_user_churn = pd.read_sql_query("SELECT * FROM tbl_user_hits;", conn)
```

```
In [7]: df_user_churn.head()
```

Out[7]:

	userid	date
0	1	1/1/2017
1	2	1/2/2017
2	3	1/3/2017
3	4	1/1/2018
4	5	1/2/2018

```
In [13]: df_user_churn.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 2 columns):
userid    9 non-null int64
date      9 non-null object
dtypes: int64(1), object(1)
memory usage: 224.0+ bytes
```

Name	Date modified	Type	Size
ch_07_retrieve_sql_and_create_dataframe.ipynb	2/20/2020 4:39 PM	IPYNB File	4 KB
user_hits_export.csv	2/20/2020 4:38 PM	Microsoft Excel Comma Separated Values File	1 KB

```
In [38]: df_user_restricted = df_user_churn[df_user_churn['userid']==1]

In [39]: df_user_restricted.head()
```

Out[39]:

	userid	date
0	1	1/1/2017
6	1	1/1/2019

```
In [41]: df_user_restricted.sort_values(by='date')
```

Out[41]:

	userid	date
0	1	1/1/2017
6	1	1/1/2019

```
In [42]: df_user_restricted.sort_values(by='date', ascending=False)
```

Out[42]:

	userid	date
6	1	1/1/2019
0	1	1/1/2017

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 2 columns):
userid    9 non-null float64
date      12 non-null object
dtypes: float64(1), object(1)
memory usage: 272.0+ bytes
```

Out[26]:

	userid	date
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
5	False	False
6	False	False
7	False	False
8	False	False
9	True	False
10	True	False
11	True	False

Out[11]:

	userid	date
0	1.0	1/1/2017
1	2.0	1/2/2017
2	3.0	1/3/2017
3	4.0	1/1/2018
4	5.0	1/2/2018
5	6.0	1/3/2018
6	1.0	1/1/2019
7	3.0	1/2/2019
8	6.0	1/3/2019

user_data_2017.cvs		
userid	**date**	**year**
1	1/1/2017	2017
2	1/2/2017	2017
3	1/3/2017	2017

user_data_2018.cvs		
userid	**date**	**year**
4	1/1/2018	2018
5	1/2/2018	2018
6	1/3/2018	2018

user_data_2019.cvs		
userid	**date**	**year**
1	1/1/2019	2019
3	1/2/2019	2019
6	1/3/2019	2019

tbl_user_data_stage			
userid	**date**	**year**	**filesource**
1	1/1/2017	2017	user_data_2017.cvs
2	1/2/2017	2017	user_data_2017.cvs
3	1/3/2017	2017	user_data_2017.cvs
4	1/1/2018	2018	user_data_2018.cvs
5	1/2/2018	2018	user_data_2018.cvs
6	1/3/2018	2018	user_data_2018.cvs
1	1/1/2019	2019	user_data_2019.cvs
3	1/2/2019	2019	user_data_2019.cvs
6	1/3/2019	2019	user_data_2019.cvs

```
In [3]: df_user_data_2017.head()
```

Out[3]:

	userid	date	year
0	1	1/1/2017	2017
1	2	1/2/2017	2017
2	3	1/3/2017	2017

```
In [4]: df_user_data_2018 = pd.read_csv('user_data_2018.csv')
```

```
In [17]: df_user_data_2018.head()
```

Out[17]:

	userid	date	year
0	4	1/1/2018	2018
1	5	1/2/2018	2018
2	6	1/3/2018	2018

```
In [5]: df_user_data_2019 = pd.read_csv('user_data_2019.csv')
```

```
In [18]: df_user_data_2019.head()
```

Out[18]:

	userid	date	year
0	1	1/1/2019	2019
1	3	1/2/2019	2019
2	6	1/3/2019	2019

```
In [19]: df_user_data_combined = pd.concat([df_user_data_2017, df_user_data_2018, df_user_data_2019], ignore_index=True)

In [20]: df_user_data_combined.head(10)
```

Out[20]:

	userid	date	year
0	1	1/1/2017	2017
1	2	1/2/2017	2017
2	3	1/3/2017	2017
3	4	1/1/2018	2018
4	5	1/2/2018	2018
5	6	1/3/2018	2018
6	1	1/1/2019	2019
7	3	1/2/2019	2019
8	6	1/3/2019	2019

Q: What is your age?

- ○ Under 18 years old
- ○ 18 – 24
- ○ 15 – 44
- ○ 45 – 74
- ○ 75 years or older

Out[231]:

	userid	date
0	1	2017-01-01
1	2	2017-01-02
2	3	2017-01-03
3	4	2018-01-01
4	5	2018-01-02
5	6	2018-10-03
6	1	2019-10-01
7	3	2019-10-02
8	7	2019-10-03
9	8	2020-01-01

Out[214]:

	userid	date	age
0	1	2017-01-01	1153
1	2	2017-01-02	1152
2	3	2017-01-03	1151
3	4	2018-01-01	788
4	5	2018-01-02	787
5	6	2018-10-03	513
6	1	2019-10-01	150
7	3	2019-10-02	149
8	7	2019-10-03	148
9	8	2020-01-01	58
10	1	2020-01-02	57
11	2	2020-01-03	56

Out[216]:

	date	age
userid		
1	2020-01-02	1153
2	2020-01-03	1152
3	2019-10-02	1151
4	2018-01-01	788
5	2018-01-02	787
6	2018-10-03	513
7	2019-10-03	148
8	2020-01-01	58

Out[218]:

	date	age	age_bin
userid			
1	2020-01-02	1153	> 3 years
2	2020-01-03	1152	> 3 years
3	2019-10-02	1151	> 3 years
4	2018-01-01	788	> 3 years
5	2018-01-02	787	> 3 years
6	2018-10-03	513	1 to 2 years
7	2019-10-03	148	< 1 year
8	2020-01-01	58	< 1 year

Chapter 8: Understanding Joins, Relationships, and Aggregates

tbl_students		
student_id	student name	acceptance date
1	Mark Jo	9/1/2010
2	Joe Smith	12/1/2009
3	Roger Stein	6/1/2012

tbl_roster		
student_id	class_id	seminster date
1	1	Fall 2012
2	1	Fall 2012
3	1	Fall 2012
1	2	Fall 2012
2	2	Fall 2012
3	2	Fall 2012
1	3	Fall 2012
2	3	Fall 2012
3	3	Fall 2012

tbl_classes		
class_id	class name	teacher_id
1	Bio 101	1
2	Econ 201	2
3	English 102	1

tbl_user_hits	
userid	date
1	1/1/2019
2	1/2/2019
3	1/3/2019

tbl_user_geo		
userid	city	state
1	Dover	DE
2	Orlando	FL
3	El Paso	TX
4	Lancaster	PA
5	Ewing	NJ

Join Result			
userid	date	city	state
1	1/1/2019	Dover	DE
2	1/2/2019	Orlando	FL
3	1/3/2019	El Paso	TX

Join Result			
userid	city	state	date
1	Dover	DE	1/1/2019
2	Orlando	FL	1/2/2019
3	El Paso	TX	1/3/2019
4	Lancaster	PA	
5	Ewing	NJ	

tbl_user_hits	
userid	date
1	1/1/2019
2	1/2/2019
3	1/3/2019

tbl_user_geo		
userid	city	state
1	Dover	DE
3	El Paso	TX
5	Ewing	NJ

Join Result			
userid	date	city	state
1	1/1/2019	Dover	DE
3	1/3/2019	El Paso	TX

Join Result			
userid	date	city	state
1	1/1/2019	Dover	DE
2	1/2/2019		
3	1/3/2019	El Paso	TX
5		Ewing	NJ

```
In [8]: df_left_join.head()
```

Out[8]:

	userid	date	city	state
0	1	1/1/2019	Dover	DE
1	2	1/2/2019	None	None
2	3	1/3/2019	El Paso	TX

```
In [11]: df_inner_join.head()
```

Out[11]:

	userid	date	city	state
0	1	1/1/2019	Dover	DE
1	3	1/3/2019	El Paso	TX

```
In [32]: df_user_hits.head()
```

Out[32]:

	userid	date
0	1	1/1/2019
1	2	1/2/2019
2	3	1/3/2019

```
In [34]: df_user_geo.head()
```

Out[34]:

	userid	city	state
0	1	Dover	DE
1	3	El Paso	TX
2	5	Ewing	NJ

Out[36]:

	userid	date	city	state
0	1	1/1/2019	Dover	DE
1	2	1/2/2019	NaN	NaN
2	3	1/3/2019	El Paso	TX

Out[40]:

	userid	date	city	state
0	1.0	1/1/2019	Dover	DE
1	3.0	1/3/2019	El Paso	TX
2	5.0	NaN	Ewing	NJ

Out[41]:

	userid	date	city	state
0	1	1/1/2019	Dover	DE
1	3	1/3/2019	El Paso	TX

Out[42]:

	userid	date	city	state
0	1.0	1/1/2019	Dover	DE
1	2.0	1/2/2019	NaN	NaN
2	3.0	1/3/2019	El Paso	TX
3	5.0	NaN	Ewing	NJ

Source Table

tbl_user_hits

userid	date
1	1/1/2019
2	1/1/2019
3	1/1/2019
1	1/2/2019
2	1/2/2019
3	1/2/2019
1	1/3/2019
2	1/3/2019
3	1/3/2019

Aggregation Example by User

userid	min_date	max_date	total_hits
1	1/1/2019	1/3/2019	3
2	1/1/2019	1/3/2019	3
3	1/1/2019	1/3/2019	3

Aggregation Example by Date

date	user_count	total_hits
1/1/2019	3	3
1/2/2019	3	3
1/3/2019	3	3

Out[6]:

	userid	date	city	state
0	1	1/1/2019	Dover	DE
1	3	1/1/2019	El Paso	TX
2	1	1/2/2019	Dover	DE
3	2	1/2/2019	Philadelphia	PA
4	3	1/2/2019	El Paso	TX
5	1	1/3/2019	Dover	DE
6	2	1/3/2019	Philadelphia	PA

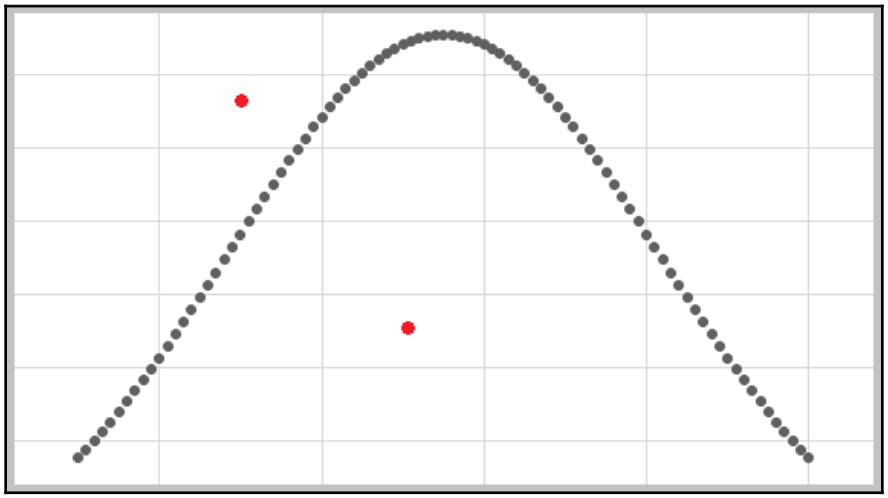
```
In [20]: df_groupby_SQL
```

Out[20]:

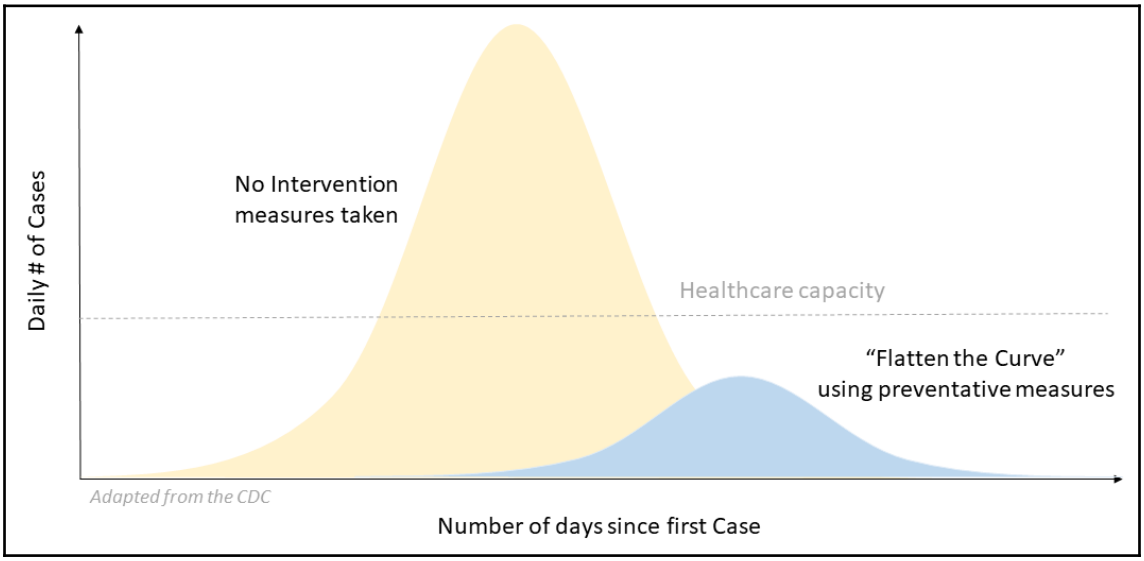
	city	state	hits
0	Dover	DE	3
1	El Paso	TX	2
2	Philadelphia	PA	2

```
In [47]: df_groupby_city_state.head()
```

```
Out[47]: city          state
         Dover         DE       3
         El Paso       TX       2
         Philadelphia  PA       2
         Name: userid, dtype: int64
```



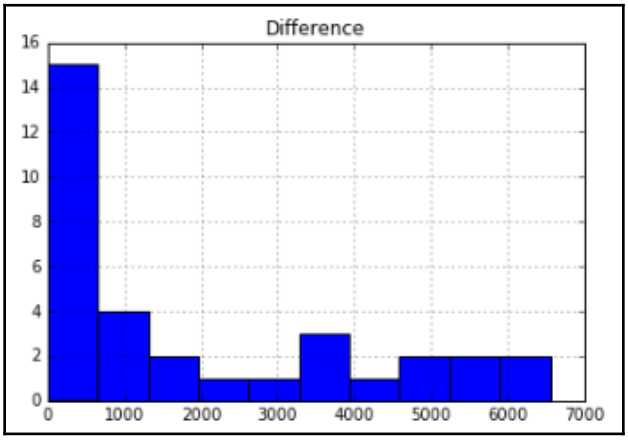
Chapter 9: Plotting, Visualization, and Storytelling



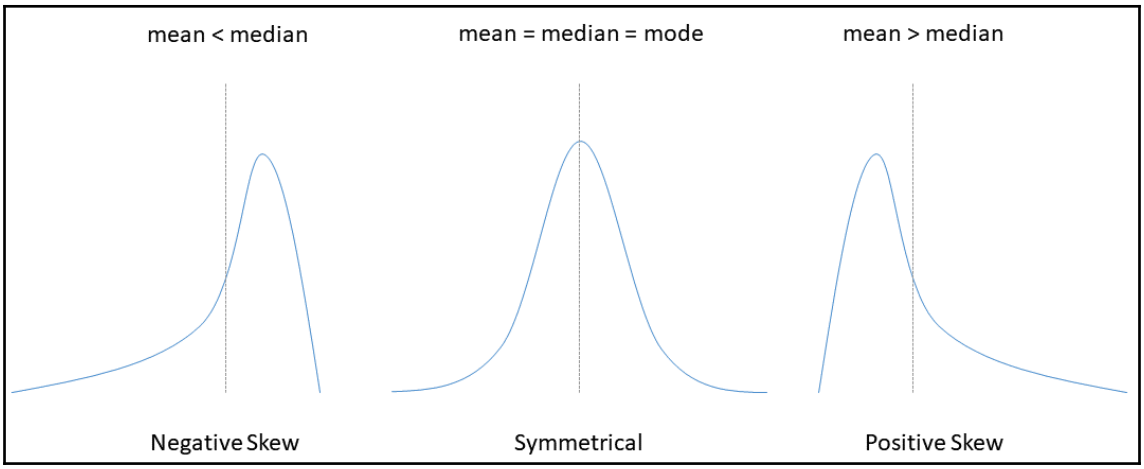

```
Date,Country_Region,Province_State,Difference,Prep_Flow_Runtime,Latest_Date,Case_Type,Cases,Lat,Long
3/9/2020,India,N/A,0,3/24/2020 9:39:03 AM,3/23/2020,Deaths,0,21,78
3/8/2020,India,N/A,0,3/24/2020 9:39:03 AM,3/23/2020,Deaths,0,21,78
3/7/2020,India,N/A,0,3/24/2020 9:39:03 AM,3/23/2020,Deaths,0,21,78
3/6/2020,India,N/A,0,3/24/2020 9:39:03 AM,3/23/2020,Deaths,0,21,78
3/5/2020,India,N/A,0,3/24/2020 9:39:03 AM,3/23/2020,Deaths,0,21,78
3/4/2020,India,N/A,0,3/24/2020 9:39:03 AM,3/23/2020,Deaths,0,21,78
3/3/2020,India,N/A,0,3/24/2020 9:39:03 AM,3/23/2020,Deaths,0,21,78
3/23/2020,India,N/A,3,3/24/2020 9:39:03 AM,3/23/2020,Deaths,10,21,78
3/22/2020,India,N/A,3,3/24/2020 9:39:03 AM,3/23/2020,Deaths,7,21,78
```

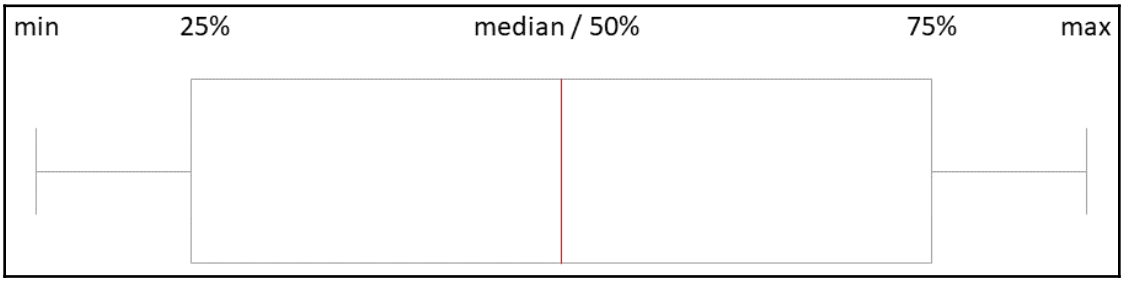
	Date	Country_Region	Province_State	Difference	Prep_Flow_Runtime	Latest_Date	Case_Type	Cases	Lat	Long
0	3/9/2020	India	NaN	0	3/24/2020 9:39:03 AM	3/23/2020	Deaths	0	21.0	78.0
1	3/8/2020	India	NaN	0	3/24/2020 9:39:03 AM	3/23/2020	Deaths	0	21.0	78.0
2	3/7/2020	India	NaN	0	3/24/2020 9:39:03 AM	3/23/2020	Deaths	0	21.0	78.0
3	3/6/2020	India	NaN	0	3/24/2020 9:39:03 AM	3/23/2020	Deaths	0	21.0	78.0
4	3/5/2020	India	NaN	0	3/24/2020 9:39:03 AM	3/23/2020	Deaths	0	21.0	78.0

	Date	Country_Region	Province_State	Difference	Prep_Flow_Runtime	Latest_Date	Case_Type	Cases	Lat	Long
28982	3/23/2020	Italy	NaN	4789	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	63927	43.0	12.0
28983	3/22/2020	Italy	NaN	5560	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	59138	43.0	12.0
28984	3/21/2020	Italy	NaN	6557	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	53578	43.0	12.0
28985	3/20/2020	Italy	NaN	5986	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	47021	43.0	12.0
28987	3/19/2020	Italy	NaN	5322	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	41035	43.0	12.0
28988	3/18/2020	Italy	NaN	4207	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	35713	43.0	12.0
28989	3/17/2020	Italy	NaN	3526	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	31506	43.0	12.0




```
Out[98]: count      33.000000
         mean     1937.181818
         std      2132.965299
         min         1.000000
         25%       202.000000
         50%       778.000000
         75%      3526.000000
         max      6557.000000
         Name: Difference, dtype: float64
```



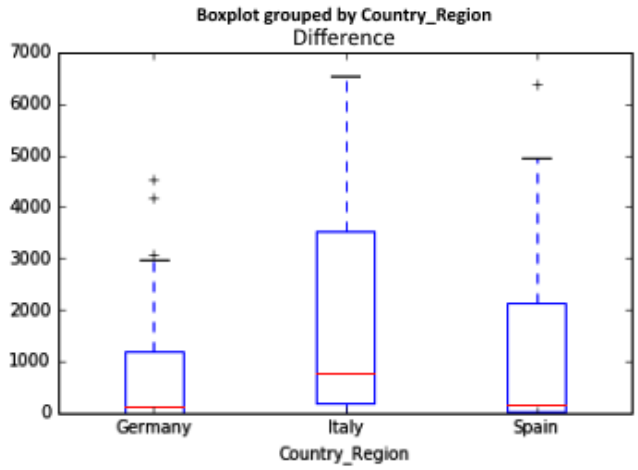


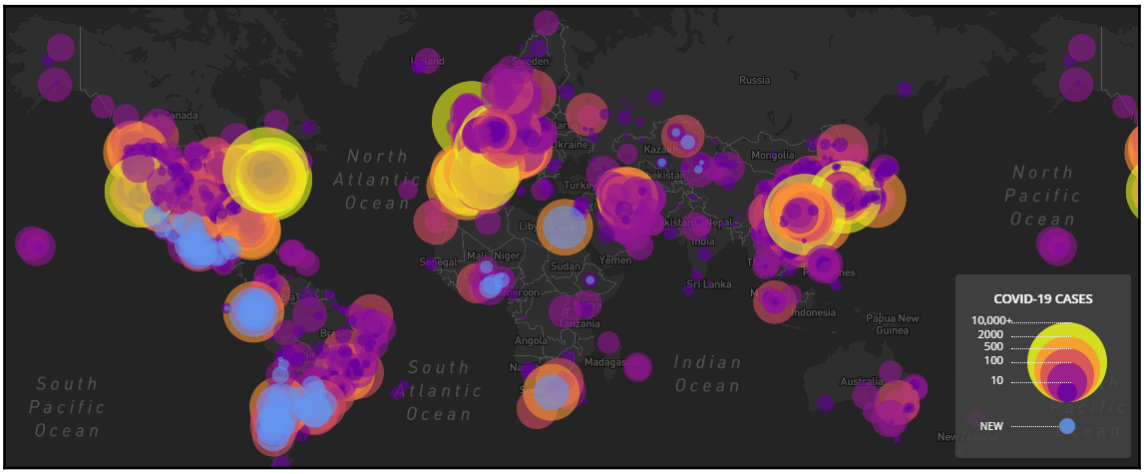
	Date	Country_Region	Province_State	Difference	Prep_Flow_Runtime	Latest_Date	Case_Type	Cases	Lat	Long
0	3/9/2020	India	NaN	0	3/24/2020 9:39:03 AM	3/23/2020	Deaths	0	21.0	78.0
1	3/8/2020	India	NaN	0	3/24/2020 9:39:03 AM	3/23/2020	Deaths	0	21.0	78.0
2	3/7/2020	India	NaN	0	3/24/2020 9:39:03 AM	3/23/2020	Deaths	0	21.0	78.0
3	3/6/2020	India	NaN	0	3/24/2020 9:39:03 AM	3/23/2020	Deaths	0	21.0	78.0
4	3/5/2020	India	NaN	0	3/24/2020 9:39:03 AM	3/23/2020	Deaths	0	21.0	78.0

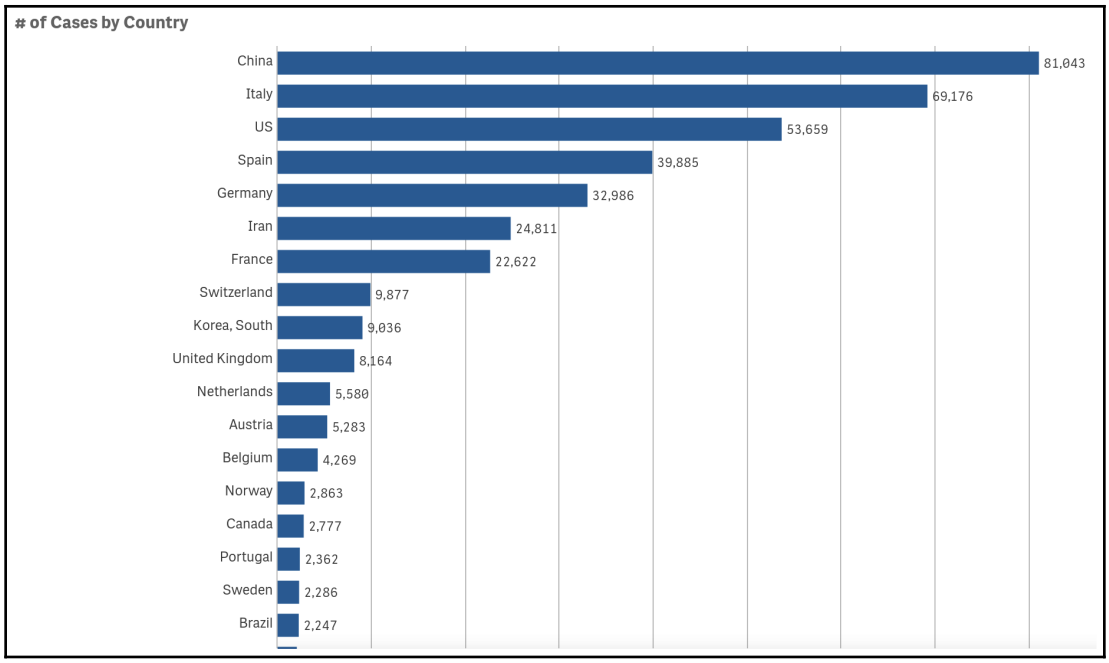
Out[78]:

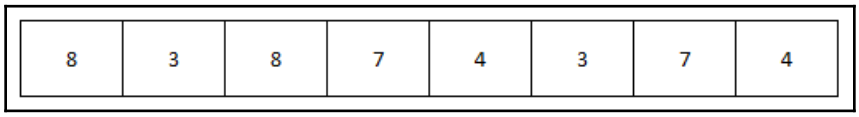
	Date	Country_Region	Province_State	Difference	Prep_Flow_Runtime	Latest_Date	Case_Type	Cases	Lat	Long
20191	3/9/2020	Germany	NaN	136	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	1176	51.0	9.0
20192	3/8/2020	Germany	NaN	241	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	1040	51.0	9.0
20193	3/7/2020	Germany	NaN	129	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	799	51.0	9.0
20194	3/6/2020	Germany	NaN	188	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	670	51.0	9.0
20195	3/5/2020	Germany	NaN	220	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	482	51.0	9.0

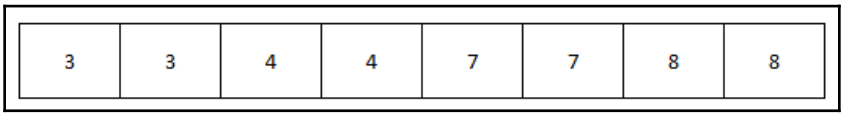
```
Out[79]: <matplotlib.axes._subplots.AxesSubplot at 0x28dd233aeb8>
```

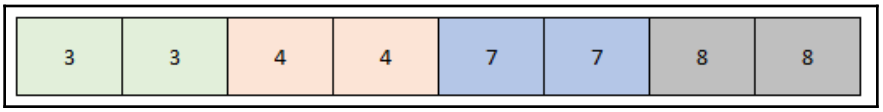


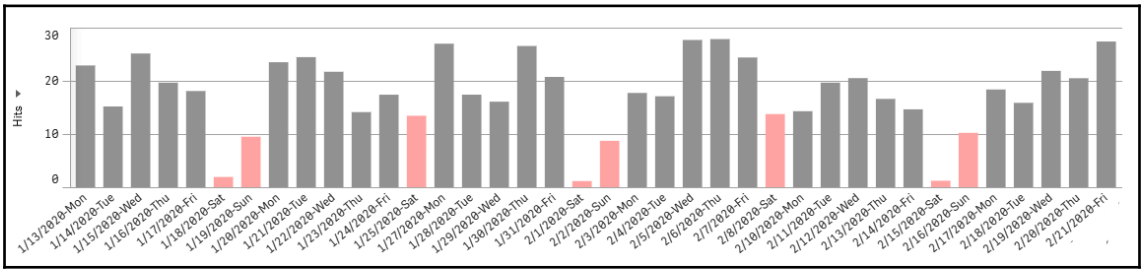



# of Cases by Country
China 81,043
Italy 69,176
US 53,659
Spain 39,885
Germany 32,986
Iran 24,811
France 22,622
Switzerland 9,877
Korea, South 9,036
United Kingdom 8,164
Netherlands 5,580
Austria 5,283
Belgium 4,269
Norway 2,863
Canada 2,777
Portugal 2,362
Sweden 2,286
Brazil 2,247


8 3 8 7 4 3 7 4


3 3 4 4 7 7 8 8


3 3 4 4 7 7 8 8


Hits
30
20
10
0
1/13/2020-Mon
1/14/2020-Tue
1/15/2020-Wed
1/16/2020-Thu
1/17/2020-Fri
1/18/2020-Sat
1/19/2020-Sun
1/20/2020-Mon
1/21/2020-Tue
1/22/2020-Wed
1/23/2020-Thu
1/24/2020-Fri
1/25/2020-Sat
1/27/2020-Mon
1/28/2020-Tue
1/29/2020-Wed
1/30/2020-Thu
1/31/2020-Fri
2/1/2020-Sat
2/2/2020-Sun
2/3/2020-Mon
2/4/2020-Tue
2/5/2020-Wed
2/6/2020-Thu
2/7/2020-Fri
2/8/2020-Sat
2/10/2020-Mon
2/11/2020-Tue
2/12/2020-Wed
2/13/2020-Thu
2/14/2020-Fri
2/15/2020-Sat
2/16/2020-Sun
2/17/2020-Mon
2/18/2020-Tue
2/19/2020-Wed
2/20/2020-Thu
2/21/2020-Fri

Out[78]:

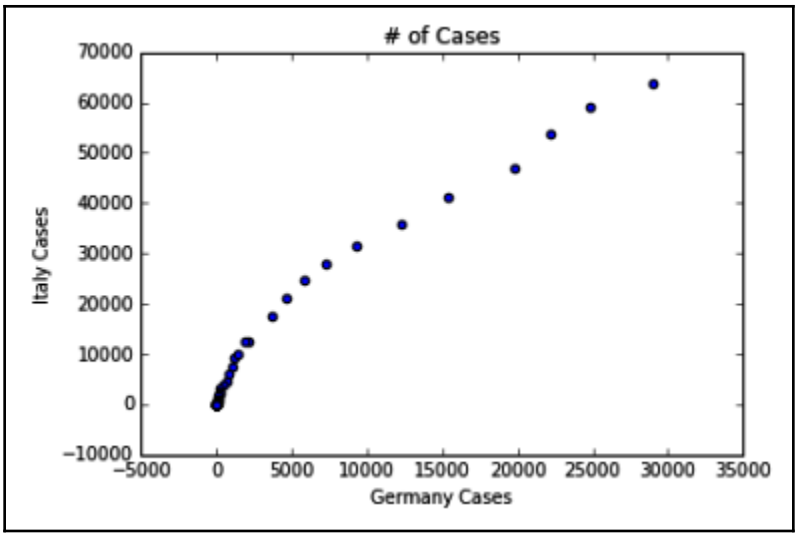
	Date	Country_Region	Province_State	Difference	Prep_Flow_Runtime	Latest_Date	Case_Type	Cases	Lat	Long
20191	3/9/2020	Germany	NaN	136	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	1176	51.0	9.0
20192	3/8/2020	Germany	NaN	241	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	1040	51.0	9.0
20193	3/7/2020	Germany	NaN	129	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	799	51.0	9.0
20194	3/6/2020	Germany	NaN	188	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	670	51.0	9.0
20195	3/5/2020	Germany	NaN	220	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	482	51.0	9.0

Out[43]:

	Date	Country_Region	Province_State	Difference	Prep_Flow_Runtime	Latest_Date	Case_Type	Cases	Lat	Long
28975	3/9/2020	Italy	NaN	1797	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	9172	43.0	12.0
28976	3/8/2020	Italy	NaN	1492	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	7375	43.0	12.0
28977	3/7/2020	Italy	NaN	1247	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	5883	43.0	12.0
28978	3/6/2020	Italy	NaN	778	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	4636	43.0	12.0
28979	3/5/2020	Italy	NaN	769	3/24/2020 9:39:03 AM	3/23/2020	Confirmed	3858	43.0	12.0

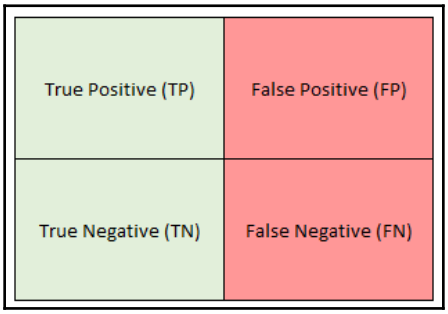
```
Out[50]: count       61.000000
         mean      2707.491803
         std       6468.499823
         min          0.000000
         25%         13.000000
         50%         16.000000
         75%       1040.000000
         max      29056.000000
         Name: Cases, dtype: float64
```

```
Out[51]: count       61.000000
         mean      8241.770492
         std      15956.982677
         min          0.000000
         25%          3.000000
         50%         62.000000
         75%       7375.000000
         max      63927.000000
         Name: Cases, dtype: float64
```


# of Cases
Italy Cases
Germany Cases
70000
60000
50000
40000
30000
20000
10000
0
−10000
−5000
0
5000
10000
15000
20000
25000
30000
35000

Chapter 10: Exploring Text Data and Unstructured Data

id	phrase	key_word_found_flag
1	Automated client-driven internet solution is balanced	1
2	Configurable balanced content-based toolset	1
3	Balanced 5th generation pricing structure	1
4	Virtual bandwidth-monitored website is well-balanced	1
5	Visionary needs-based balancing act	1




```
In [2]: nltk.download('brown')
        [nltk_data] Downloading package brown to /home/nbuser/nltk_data...
        [nltk_data]   Unzipping corpora/brown.zip.
Out[2]: True
```

```
In [4]: from nltk.corpus import brown
In [5]: brown.words()
Out[5]: ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
```

```
In [15]: count_of_words = len(brown.words())

In [16]: print('Count of all the words found the Brown Corpus =',format(count_of_words,',d'))
         Count of all the words found the Brown Corpus = 1,161,192
```

```
In [20]: nltk.download('punkt')
         [nltk_data] Downloading package punkt to /home/nbuser/nltk_data...
         [nltk_data]   Unzipping tokenizers/punkt.zip.
Out[20]: True
```

```
In [24]: input_sentence = "Seth and Becca love to run down to the playground when the weather is nice."

In [26]: nltk.word_tokenize(input_sentence)
Out[26]: ['Seth',
          'and',
          'Becca',
          'love',
          'to',
          'run',
          'down',
          'to',
          'the',
          'playground',
          'when',
          'the',
          'weather',
          'is',
          'nice',
          '.']
```

```
In [38]: from nltk.tokenize import sent_tokenize

In [41]: input_data = "Seth and Becca love the playground.  When it sunny, they head down there to play."

In [42]: print(sent_tokenize(input_data))
         ['Seth and Becca love the playground.', 'When it sunny, they head down there to play.']
```

```
In [29]: input_data = FreqDist(brown.words())
         print(input_data)
         <FreqDist with 56057 samples and 1161192 outcomes>
```

```
In [25]: input_data.most_common(10)

Out[25]: [('the', 62713),
          (',', 58334),
          ('.', 49346),
          ('of', 36080),
          ('and', 27915),
          ('to', 25732),
          ('a', 21881),
          ('in', 19536),
          ('that', 10237),
          ('is', 10011)]
```

```
In [61]: my_word_stemmer.stem('fishing')

Out[61]: 'fish'
```

```
In [78]: my_word_lemmatizer.lemmatize('fishing')

Out[78]: 'fishing'
```

```
In [113]: for x in my_list_of_words :
              print('word =', x, ': stem =', my_word_stemmer.stem(x), ': lemma =', my_word_lemmatizer.lemmatize(x))

word = The : stem = The : lemma = The
word = Fulton : stem = Fulton : lemma = Fulton
word = County : stem = Counti : lemma = County
word = Grand : stem = Grand : lemma = Grand
word = Jury : stem = Juri : lemma = Jury
word = said : stem = said : lemma = said
word = Friday : stem = Friday : lemma = Friday
word = an : stem = an : lemma = an
word = investigation : stem = investig : lemma = investigation
word = of : stem = of : lemma = of
```

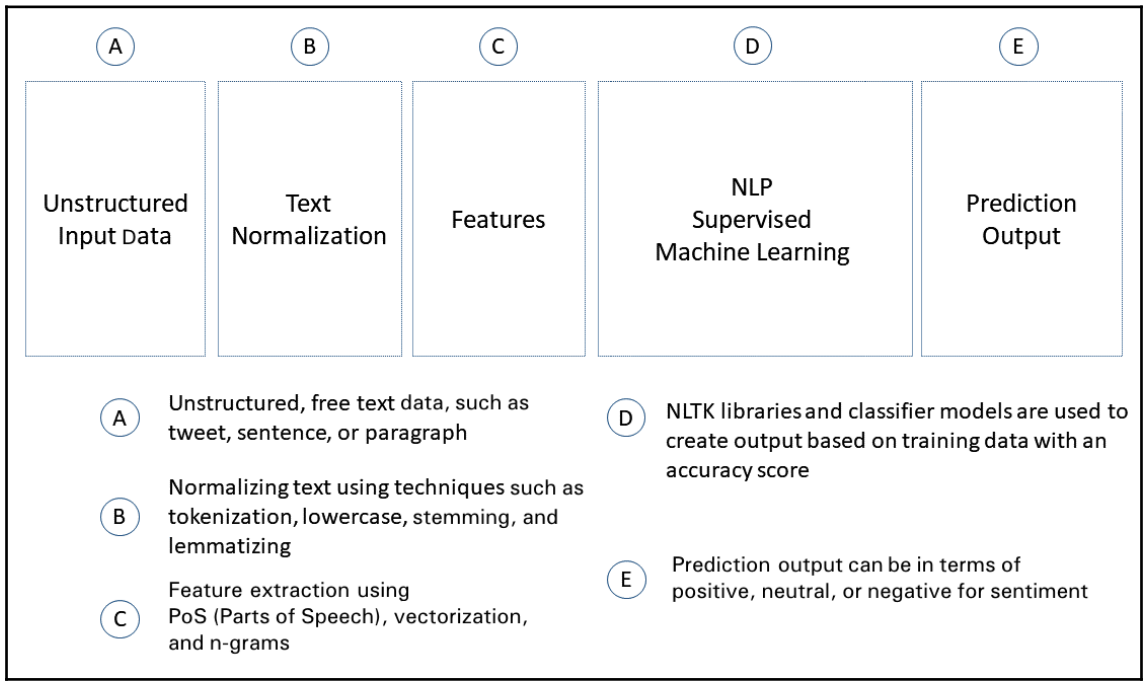
```
In [11]: input_data_cleaned = [x for x in word_tokens if not x in stop_words]
         input_data_cleaned = []

         for x in word_tokens:
             if x not in stop_words:
                 input_data_cleaned.append(x)

         print(word_tokens)
         print(input_data_cleaned)

['Seth', 'and', 'Becca', 'love', 'the', 'playground', '.', 'When', 'it', 'sunny', ',', 'they', 'head', 'down', 'there', 'to',
'play', '.']
['Seth', 'Becca', 'love', 'playground', '.', 'When', 'sunny', ',', 'head', 'play', '.']
```

Chapter 11: Practical Sentiment Analysis




```
In [19]: nltk.download("names")

         [nltk_data] Downloading package names to /home/nbuser/nltk_data...
         [nltk_data]   Package names is already up-to-date!

Out[19]: True
```

```
In [37]: print("Count of Words in male.txt:", len(names.words('male.txt')))
         print("Count of Words in female.txt:", len(names.words('female.txt')))

         Count of Words in male.txt: 2943
         Count of Words in female.txt: 5001
```

```
In [40]: print("Sample list Male names:", names.words('male.txt')[0:5])
         print("Sample list Female names:", names.words('female.txt')[0:5])

         Sample list Male names: ['Aamir', 'Aaron', 'Abbey', 'Abbie', 'Abbot']
         Sample list Female names: ['Abagael', 'Abagail', 'Abbe', 'Abbey', 'Abbi']
```

```
In [16]: gender_features('Debra')

Out[16]: {'last_letter': 'a'}
```

```
In [44]: labeled_names = ([(name, 'male') for name in names.words('male.txt')] +
                         [(name, 'female') for name in names.words('female.txt')])
         print(labeled_names[0:5])

         [('Aamir', 'male'), ('Aaron', 'male'), ('Abbey', 'male'), ('Abbie', 'male'), ('Abbot', 'male')]
```

```
In [46]: import random
         random.shuffle(labeled_names)
         print(labeled_names[0:5])

         [('Lindie', 'female'), ('Krysta', 'female'), ('Cathy', 'female'), ('Orin', 'male'), ('Siouxie', 'female')]
```

```
In [61]: featuresets = [(gender_features(n), gender) for (n, gender) in labeled_names]
         print(featuresets[0:5])

         [({'last_letter': 'e'}, 'female'), ({'last_letter': 'a'}, 'female'), ({'last_letter': 'y'},
         'female'), ({'last_letter': 'n'}, 'male'), ({'last_letter': 'e'}, 'female')]
```

```
In [96]: train_set, test_set = featuresets[500:], featuresets[:500]
         print("Count of features in Training Set:", len(train_set))
         print("Count of features in Test Set:", len(test_set))

         Count of features in Training Set: 7444
         Count of features in Test Set: 500
```

```
In [103]: classifier = nltk.NaiveBayesClassifier.train(train_set)

In [104]: classifier.classify(gender_features('Aaron'))
Out[104]: 'male'

In [105]: classifier.classify(gender_features('Marc'))
Out[105]: 'male'

In [106]: classifier.classify(gender_features('Debra'))
Out[106]: 'female'

In [107]: classifier.classify(gender_features('Deb'))
Out[107]: 'male'

In [108]: classifier.classify(gender_features('Seth'))
Out[108]: 'female'
```

```
In [34]: my_input_sentence = "I HATE my school!"
         my_analyzer.polarity_scores(my_input_sentence)
Out[34]: {'compound': -0.6932, 'neg': 0.703, 'neu': 0.297, 'pos': 0.0}
```

```
In [21]: import nltk

In [22]: nltk.download('vader_lexicon')
         [nltk_data] Downloading package vader_lexicon to
         [nltk_data]     /home/nbuser/nltk_data...
         [nltk_data]   Package vader_lexicon is already up-to-date!
Out[22]: True
```

```
In [34]: my_input_sentence = "I HATE my school!"
         my_analyzer.polarity_scores(my_input_sentence)

Out[34]: {'compound': -0.6932, 'neg': 0.703, 'neu': 0.297, 'pos': 0.0}
```

```
id,text
1,I Hate my School!!!
2,@socialmediahandle I learned something new today
3,I need to take a cool trip to Austraila!
4,The restaurant service was amazing!
5,I will never go back there again!
6,You learn something new every day - this place is great
7,First Impressions - this is good but then it went downhill from there
8,A bit pricey
9,Love them
10,meh
```

```
In [30]: !pip install twython

Requirement already satisfied: twython in /home/nbuser/anaconda3_420/lib/python3.5/site-packages (3.8.2)
Requirement already satisfied: requests>=2.1.0 in /home/nbuser/anaconda3_420/lib/python3.5/site-packages (from twython) (2.14.
2)
Requirement already satisfied: requests-oauthlib>=0.4.0 in /home/nbuser/anaconda3_420/lib/python3.5/site-packages (from twytho
n) (1.3.0)
Requirement already satisfied: oauthlib>=3.0.0 in /home/nbuser/anaconda3_420/lib/python3.5/site-packages (from requests-oauthli
b>=0.4.0->twython) (3.1.0)
WARNING: You are using pip version 19.3.1; however, version 20.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

```
In [33]: nltk.download('vader_lexicon')

         [nltk_data] Downloading package vader_lexicon to
         [nltk_data]     /home/nbuser/nltk_data...
         [nltk_data]   Package vader_lexicon is already up-to-date!

Out[33]: True
```

```
In [34]: sentences = pd.read_csv('social_media_sample_file.csv')
         len(sentences)

Out[34]: 10
```

```
In [35]: sentences.head()
```

Out[35]:

	id	text
0	1	I Hate my School!!!
1	2	@socialmediahandle I learned something new today
2	3	I need to take a cool trip to Austraila!
3	4	The restaurant service was amazing!
4	5	I will never go back there again!

```
In [40]: sentences.head(10)
```

Out[40]:

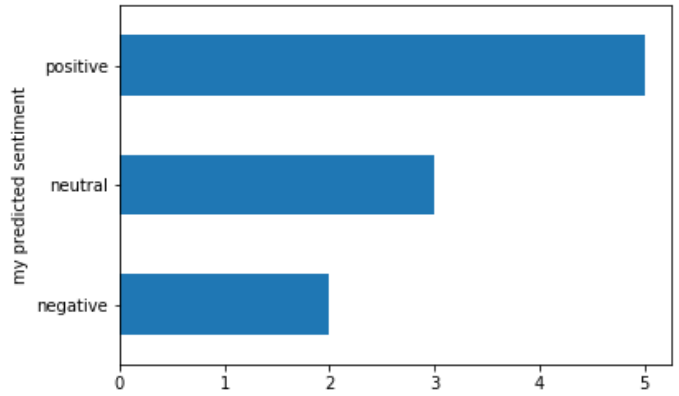
	id	text	my VADER Score	my VADER score - positive	my VADER score - negative	my VADER score - neutral
0	1	I Hate my School!!!	-0.6784	0.000	0.696	0.304
1	2	@socialmediahandle I learned something new today	0.0000	0.000	0.000	1.000
2	3	I need to take a cool trip to Austraila!	0.3802	0.302	0.000	0.698
3	4	The restaurant service was amazing!	0.6239	0.506	0.000	0.494
4	5	I will never go back there again!	0.0000	0.000	0.000	1.000
5	6	You learn something new every day - this place...	0.6249	0.313	0.000	0.687
6	7	First Impressions - this is good but then it w...	0.3400	0.254	0.000	0.746
7	8	A bit pricey	0.0000	0.000	0.000	1.000
8	9	Love them	0.6369	0.808	0.000	0.192
9	10	meh	-0.0772	0.000	1.000	0.000

```
In [62]: sentences.head(10)
```

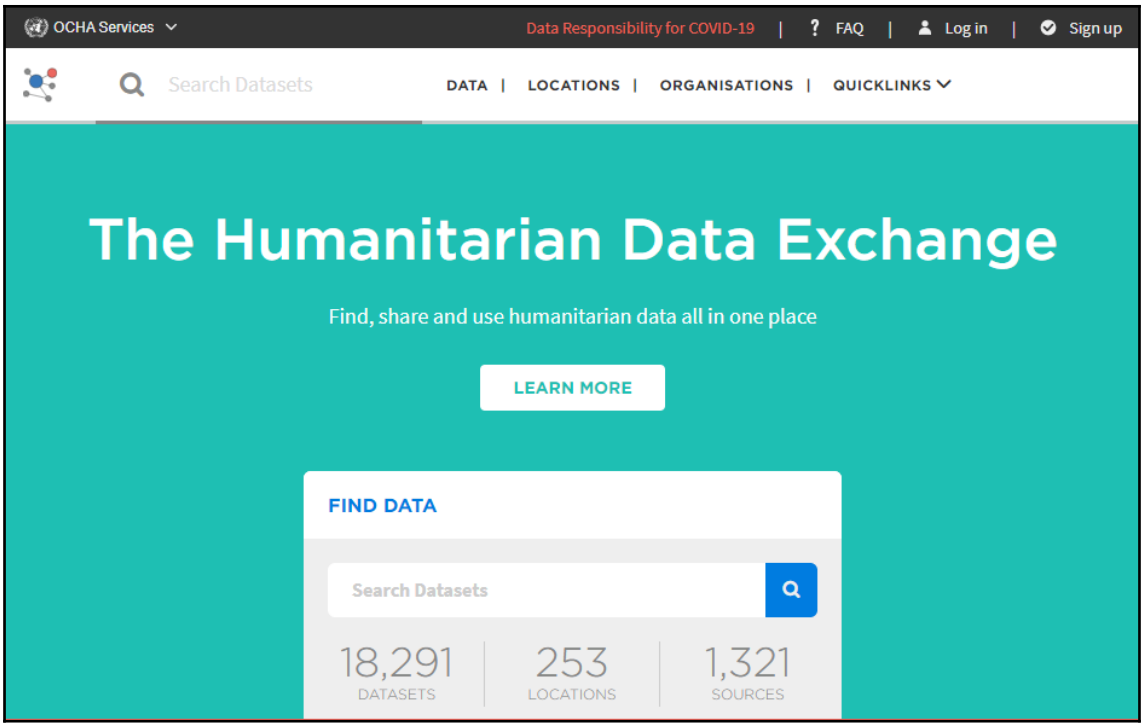
Out[62]:

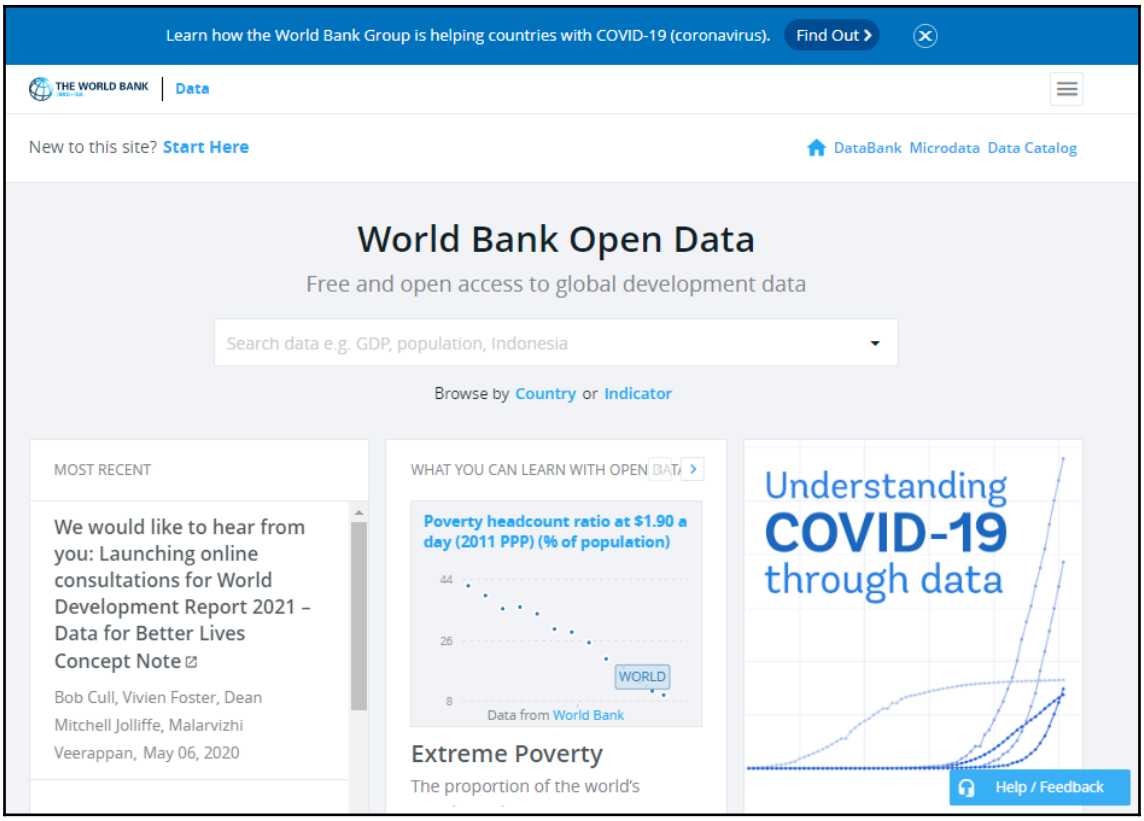
	id	text	my VADER Score	my VADER score - positive	my VADER score - negative	my VADER score - neutral	predicted sentiment	my predicted sentiment
0	1	I Hate my School!!!	-0.6784	0.000	0.696	0.304	negative	negative
1	2	@socialmediahandle I learned something new today	0.0000	0.000	0.000	1.000	neutral	neutral
2	3	I need to take a cool trip to Austraila!	0.3802	0.302	0.000	0.698	positive	positive
3	4	The restaurant service was amazing!	0.6239	0.506	0.000	0.494	positive	positive
4	5	I will never go back there again!	0.0000	0.000	0.000	1.000	neutral	neutral
5	6	You learn something new every day - this place...	0.6249	0.313	0.000	0.687	positive	positive
6	7	First Impressions - this is good but then it w...	0.3400	0.254	0.000	0.746	positive	positive
7	8	A bit pricey	0.0000	0.000	0.000	1.000	neutral	neutral
8	9	Love them	0.6369	0.808	0.000	0.192	positive	positive
9	10	meh	-0.0772	0.000	1.000	0.000	negative	negative

```
In [63]: sentences.groupby('my predicted sentiment').size().plot(kind='barh');
```



Chapter 12: Bringing It All Together


OCHA Services
Data Responsibility for COVID-19
FAQ
Log in
Sign up
Search Datasets
DATA | LOCATIONS | ORGANISATIONS | QUICKLINKS
The Humanitarian Data Exchange
Find, share and use humanitarian data all in one place
LEARN MORE
FIND DATA
Search Datasets
18,291
DATASETS
253
LOCATIONS
1,321
SOURCES


Learn how the World Bank Group is helping countries with COVID-19 (coronavirus).
Find Out
THE WORLD BANK | Data
New to this site? Start Here
DataBank Microdata Data Catalog
World Bank Open Data
Free and open access to global development data
Search data e.g. GDP, population, Indonesia
Browse by Country or Indicator
MOST RECENT
We would like to hear from you: Launching online consultations for World Development Report 2021 – Data for Better Lives Concept Note
Bob Cull, Vivien Foster, Dean Mitchell Jolliffe, Malarvizhi Veerappan, May 06, 2020
WHAT YOU CAN LEARN WITH OPEN DATA
Poverty headcount ratio at $1.90 a day (2011 PPP) (% of population)
44
26
8
WORLD
Data from World Bank
Extreme Poverty
The proportion of the world's
Understanding COVID-19 through data
Help / Feedback

	A	B	C	D	E	F	G	H	I	J
1	Type	Maturity	Currency	Volume	Coupon	Settlement Date	Maturity Date	USD Equivalent	ISIN	Final Terms
2	Green	5	INR	80000000	0.045	4/5/2018 0:00	3/24/2023 0:00	1230000	XS1801822419	
3	Green	5	INR	80000000	0.0405	11/28/2017 0:00	11/28/2022 0:00	1230000	XS1724511958	
4	Green	5	INR	1608700000	0.05	9/27/2017 0:00		25100000	XS1687324068	http://pubdocs.worldbank.org/en/206211544553172402/ft-2017-XS1687324068.pd
5	Green	3	MYR	65000000	0.0295	3/13/2017 0:00	3/13/2020 0:00	14620000	XS1576723552	http://pubdocs.worldbank.org/en/302211544553170997/ft-2017-XS1576723552.pd
6	Green	3	USD	100000000	0.01181	11/14/2016 0:00	12/15/2019 0:00	100000000	XS1517268105	
7	Green	5	INR	200000000	0.055	6/29/2016 0:00	6/28/2021 0:00	2990000	XS1432564133	http://pubdocs.worldbank.org/en/188831550243927511/ft-2016-XS1432564133.pd
8	Green	5	INR	140000000	0.054	6/29/2016 0:00	6/29/2021 0:00	2090000	XS1432390125	
9	Green	12	USD	50000000	0	4/6/2016 0:00		50000000	XS1386304395	http://pubdocs.worldbank.org/en/833271544553164048/ft-2016-XS1386304395.pd
10	Green	9	USD	16390000	0	11/13/2015 0:00	1/8/2016 0:00	16390000	XS1319581960	
11	Green	5	BRL	100000000	0.1175	11/4/2015 0:00		25640000	XS1315186921	http://pubdocs.worldbank.org/en/782911544553161009/ft-2015-XS1315186921.pd
12	Green	10	USD	1000000	0.02	11/2/2015 0:00	11/4/2025 0:00	1000000	XS1296661298	http://pubdocs.worldbank.org/en/306781544553159603/ft-2015-XS1296661298.pd
13	Green	7	USD	1000000	0.016	11/2/2015 0:00	11/4/2022 0:00	1000000	XS1296575969	
14	Green	5	USD	1000000	0.012	11/2/2015 0:00	11/4/2020 0:00	1000000	XS1296575373	
15	Green	10	USD	50000000	0	7/3/2015 0:00		50000000	XS1249904597	http://pubdocs.worldbank.org/en/425101544553155253/ft-2015-XS1249904597.pd
16	Green	5	INR	260000000	0.056	6/25/2015 0:00	6/25/2020 0:00	4080000	XS1241051967	
17	Green	10	MXN	263000000	0.0525	5/8/2015 0:00	5/18/2015 0:00	17340000	XS1233786950	http://pubdocs.worldbank.org/en/127361544553152382/ft-2015-XS1233786950.pd
18	Green	8	USD	83540000	0	6/29/2015 0:00		83540000	XS1233613188	http://pubdocs.worldbank.org/en/215251544553150986/ft-2015-XS1233613188.pd
19	Green	10	USD	50000000	0	5/15/2015 0:00		50000000	XS1225604815	http://pubdocs.worldbank.org/en/369311544553149492/ft-2015-XS1225604815.pd
20	Green	10	USD	80000000	0.0195	5/8/2015 0:00		80000000	XS1225179420	http://pubdocs.worldbank.org/en/398531550249432973/ft-2015-XS1225179420.pd

	A	B	C	D	E	F	G	H	I	J
1	Date	EUR	JPY	BGN	CZK	DKK	GBP	HUF	PLN	RON
2	#date	#value+eur	#value+jpy	#value+bgn	#value+czk	#value+dkk	#value+gbp	#value+huf	#value+pln	#value+ron
3	5/7/2020	1.0783	0.009383866	0.551334492	0.039704691	0.144534549	1.232652781	0.003086501	0.237161018	0.223551363
4	5/6/2020	1.0807	0.009426079	0.552561612	0.040045207	0.144831006	1.238582054	0.003091336	0.238071111	0.224058217
5	5/5/2020	1.0843	0.009370841	0.554402291	0.040192008	0.145325149	1.245462899	0.003095612	0.239275311	0.224678823
6	5/4/2020	1.0942	0.009364944	0.559464158	0.040348095	0.14663236	1.244851988	0.003097348	0.239598844	0.22622395
7	4/30/2020	1.0876	0.009386381	0.55608958	0.040137285	0.14582216	1.251481503	0.003083466	0.239897653	0.22456691
8	4/29/2020	1.0842	0.009385388	0.554351161	0.039970507	0.14539164	1.240815766	0.00304936	0.238589851	0.223846392
9	4/28/2020	1.0877	0.009371877	0.55614071	0.039949315	0.145862948	1.249109993	0.003055852	0.23922319	0.224522655
10	4/27/2020	1.0852	0.009337463	0.55486246	0.039920541	0.145486721	1.243596943	0.003060609	0.239627266	0.224353938
11	4/24/2020	1.08	0.009292721	0.552203702	0.039545954	0.144816767	1.234313927	0.003040626	0.238494833	0.223052934
12	4/23/2020	1.0772	0.009306263	0.550772063	0.039098399	0.144441316	1.235321101	0.003013063	0.237378523	0.222447083
13	4/22/2020	1.0867	0.00928724	0.55562941	0.039467567	0.145691724	1.236010009	0.003060179	0.239630422	0.224566552
14	4/21/2020	1.0837	0.009310937	0.554095511	0.039483368	0.145303156	1.229800272	0.003052504	0.239274911	0.224029934
15	4/20/2020	1.086	0.009273333	0.5552715	0.039739461	0.145607637	1.243373825	0.003062865	0.239883372	0.224486843
16	4/17/2020	1.086	0.009293171	0.5552715	0.039980856	0.145570553	1.248591598	0.003086893	0.240345247	0.224584333
17	4/16/2020	1.0888	0.009296448	0.556703139	0.040246923	0.145914579	1.249153884	0.003114773	0.240416887	0.225084241
18	4/15/2020	1.0903	0.009309255	0.557470089	0.040394946	0.146080362	1.247696973	0.003112475	0.240254732	0.225371036
19	4/14/2020	1.0963	0.009317525	0.560537887	0.040809261	0.14691181	1.256461096	0.003126034	0.241088118	0.226775335
20	4/9/2020	1.0867	0.009183639	0.55562941	0.040384258	0.145559023	1.241020956	0.003063198	0.238384592	0.22484999

```
In [132]: df_greenbonds.shape
Out[132]: (115, 10)
```

```
In [134]: df_greenbonds.head()
```

Out[134]:

	Type	Maturity	Currency	Volume	Coupon	Settlement Date	Maturity Date	USD Equivalent	ISIN	Final Terms
0	Green	5	INR	80000000	0.04500	04/05/2018 12:00:00 AM	03/24/2023 12:00:00 AM	1230000	XS1801822419	NaN
1	Green	5	INR	80000000	0.04050	11/28/2017 12:00:00 AM	11/28/2022 12:00:00 AM	1230000	XS1724511958	NaN
2	Green	5	INR	1608700000	0.05000	09/27/2017 12:00:00 AM	NaN	25100000	XS1687324068	http://pubdocs.worldbank.org/en/20621154455
3	Green	3	MYR	65000000	0.02950	03/13/2017 12:00:00 AM	03/13/2020 12:00:00 AM	14620000	XS1576723552	http://pubdocs.worldbank.org/en/30221154455
4	Green	3	USD	100000000	0.01181	11/14/2016 12:00:00 AM	12/15/2019 12:00:00 AM	100000000	XS1517268105	NaN

```
In [136]: df_fx_rates.shape
Out[136]: (5464, 34)
```

```
In [138]: df_fx_rates.head()
```

Out[138]:

	Date	EUR	JPY	BGN	CZK	DKK	GBP	HUF
0	#date	#value+eur	#value+jpy	#value+bgn	#value+czk	#value+dkk	#value+gbp	#value+huf
1	2020-05-07	1.0783	0.009383865633974415	0.5513344922793741	0.03970469106708888	0.14453454862274648	1.2326527812707195	0.003086501
2	2020-05-06	1.0807	0.009426079372001744	0.5525616116167297	0.04004520695149517	0.1448310059235037	1.2385820544852326	0.003091335
3	2020-05-05	1.0843	0.009370840895341804	0.5544022906227631	0.04019200830306175	0.14532514876963493	1.2454628991500114	0.003095611
4	2020-05-04	1.0942	0.00936494351249572	0.5594641578893548	0.04034809543124747	0.14663236042990002	1.244851987531002	0.003097347

5 rows × 34 columns

```
In [139]: # Delete first row because it contains hdx hashtag metadata values
          df_fx_rates = df_fx_rates.drop(0)
          df_fx_rates.head()
```

Out[139]:

	Date	EUR	JPY	BGN	CZK	DKK	GBP	HUF
1	2020-05-07	1.0783	0.009383865633974415	0.5513344922793741	0.03970469106708888	0.14453454862274648	1.2326527812707195	0.003086501030
2	2020-05-06	1.0807	0.009426079372001744	0.5525616116167297	0.04004520695149517	0.1448310059235037	1.2385820544852326	0.003091335564
3	2020-05-05	1.0843	0.009370840895341804	0.5544022906227631	0.04019200830306175	0.14532514876963493	1.2454628991500114	0.003095611956
4	2020-05-04	1.0942	0.00936494351249572	0.5594641578893548	0.04034809543124747	0.14663236042990002	1.244851987531002	0.003097347637
5	2020-04-30	1.0876	0.009386381289376024	0.5560895797116269	0.04013728457024762	0.14582216024884692	1.2514815027904032	0.003083465638

5 rows × 34 columns

```
In [10]: df_fx_rates['Date'].max()
Out[10]: '2020-05-07'
```

```
In [11]: df_fx_rates_max_date = df_fx_rates[df_fx_rates.Date==df_fx_rates['Date'].max()]
         df_fx_rates_max_date.head()
```

Out[11]:

	Date	EUR	JPY	BGN	CZK	DKK	GBP	HUF
1	2020-05-07	1.0783	0.009383865633974415	0.5513344922793741	0.03970469106708888	0.14453454862274648	1.2326527812707195	0.003086501030

1 rows × 34 columns

```
In [39]: df_rates_transposed = df_fx_rates_max_date.transpose()
         df_rates_transposed.columns.name = 'Currency'
         df_rates_transposed.columns = ['Currency_Value']
         df_rates_transposed.head(10)
```

Out[39]:

	Currency_Value
Date	2020-05-07
EUR	1.0783
JPY	0.009383865633974415
BGN	0.5513344922793741
CZK	0.03970469106708888
DKK	0.14453454862274648
GBP	1.2326527812707195
HUF	0.0030865010304556902
PLN	0.23716101788110056
RON	0.2235513631180678

```
In [46]: df_rates_transposed = df_rates_transposed.drop('Date')
         df_rates_transposed = df_rates_transposed.reindex()
         df_rates_transposed.head()
```

Out[46]:

	Currency_Value
EUR	1.0783
JPY	0.009383865633974415
BGN	0.5513344922793741
CZK	0.03970469106708888
DKK	0.14453454862274648

```
In [59]: df_greenbonds_revised = df_greenbonds.merge(df_rates_transposed, how='left', left_on='Currency', right_index=True)
         df_greenbonds_revised.head()
```

Out[59]:

Volume	Coupon	Settlement Date	Maturity Date	USD Equivalent	ISIN	Final Terms	Currency_Value
80000000	0.04500	04/05/2018 12:00:00 AM	03/24/2023 12:00:00 AM	1230000	XS1801822419	NaN	0.013187229801207066
80000000	0.04050	11/28/2017 12:00:00 AM	11/28/2022 12:00:00 AM	1230000	XS1724511958	NaN	0.013187229801207066
1608700000	0.05000	09/27/2017 12:00:00 AM	NaN	25100000	XS1687324068	http://pubdocs.worldbank.org/en/20621154455317...	0.013187229801207066
65000000	0.02950	03/13/2017 12:00:00 AM	03/13/2020 12:00:00 AM	14620000	XS1576723552	http://pubdocs.worldbank.org/en/30221154455317...	0.23124101992236926
100000000	0.01181	11/14/2016 12:00:00 AM	12/15/2019 12:00:00 AM	100000000	XS1517268105	NaN	NaN

```
In [95]: df_greenbonds_revised['Local CCY'] = df_greenbonds_revised['Local CCY'].astype(int)
         df_greenbonds_revised['USD Equivalent'] = df_greenbonds_revised['USD Equivalent'].astype(int)
         df_greenbonds_revised[['ISIN', 'Currency', 'USD Equivalent', 'Currency_Value', 'Local CCY']]
```

Out[95]:

	ISIN	Currency	USD Equivalent	Currency_Value	Local CCY
0	XS1801822419	INR	1230000	0.013187	93272053
1	XS1724511958	INR	1230000	0.013187	93272053
2	XS1687324068	INR	25100000	0.013187	1903356533
3	XS1576723552	MYR	14620000	0.231241	63224076
4	XS1517268105	USD	100000000	1.000000	100000000
5	XS1432564133	INR	2990000	0.013187	226734503
6	XS1432390125	INR	2090000	0.013187	158486659
7	XS1386304395	USD	50000000	1.000000	50000000
8	XS1319581960	USD	16390000	1.000000	16390000
9	XS1315186921	BRL	25640000	0.173162	148069038
10	XS1296661298	USD	1000000	1.000000	1000000

```
In [109]: df_greenbonds_revised[['Currency', 'USD Equivalent', 'Local CCY']].groupby(['Currency']).sum().astype(int)
```

Out[109]:

	USD Equivalent	Local CCY
Currency		
AUD	606880000	940120883
BRL	288510000	1666123174
CAD	10570000	14877203
COP	12910000	12910000
EUR	758690000	703598253

```
In [114]: df_greenbonds_revised[['Currency', 'USD Equivalent']].groupby(['Currency']).size().plot(kind='barh');
```