Master Data Analysis with Python - Intro to Pandas Solutions

by
Teddy Petrou

Contents

I Intro to Pandas **5**

1 Solutions **7**

1.1 2. The DataFrame and Series 7
1.2 3. Data Types and Missing Values 8
1.3 4. Setting a meaningful index 9

II Selecting Subsets of Data **11**

2 Solutions **13**

2.1 1. Selecting Subsets of Data from DataFrames with just the brackets 13
2.2 2. Selecting Subsets of Data from DataFrames with `loc` 14
2.3 3. Selecting Subsets of Data from DataFrames with `iloc` 18
2.4 4. Selecting Subsets of Data from a Series 20
2.5 5. Boolean Selection Single Conditions 23
2.6 6. Boolean Selection Multiple Conditions 26
2.7 7. Boolean Selection More 30
2.8 8. Filtering with the query Method 33
2.9 9. Miscellaneous Subset Selection 34

Part I

Intro to Pandas

Chapter 1

Solutions

1.1 2. The DataFrame and Series

[1]:
```
import pandas as pd
import numpy as np

pd.options.display.max_columns = 40
bikes = pd.read_csv('../data/bikes.csv')
```

Exercise 1

Select the column `events`, the type of weather that was recorded, and assign it to a variable with the same name. Output the first 10 values of it.

[2]:
```
events = bikes['events']
events.head(10)
```

[2]:
```
0    mostlycloudy
1    partlycloudy
2    mostlycloudy
3    mostlycloudy
4    partlycloudy
5    mostlycloudy
6          cloudy
7          cloudy
8          cloudy
9    mostlycloudy
Name: events, dtype: object
```

Exercise 2

What type of object is `events`?

[3]:
```
# it's a Series
type(events)
```

[3]:
```
pandas.core.series.Series
```

Exercise 3

Select the last two rows of the `bikes` DataFrame and assign it to the variable `bikes_last_2`. What type of object is `bikes_last_2`?

[4]:
```
# it's a DataFrame
bikes_last_2 = bikes.tail(2)
type(bikes_last_2)
```

[4]:
```
pandas.core.frame.DataFrame
```

Exercise 4

Use `pd.reset_option('all')` to reset the options to their default values. Test that this worked.

[5]:
```
pd.reset_option('all')
```

```
As the xlwt package is no longer maintained, the xlwt engine will be removed in a␣
 ↪future
version of pandas. This is the only engine in pandas that supports writing in the␣
 ↪xls
format. Install openpyxl and write to an xlsx file instead.

: boolean
    use_inf_as_null had been deprecated and will be removed in a future
    version. Use `use_inf_as_na` instead.


/Users/Ted/miniconda3/lib/python3.8/site-packages/pandas/_config/config.py:630:
FutureWarning: As the xlwt package is no longer maintained, the xlwt engine will be
removed in a future version of pandas. This is the only engine in pandas that␣
 ↪supports
writing in the xls format. Install openpyxl and write to an xlsx file instead.
  warnings.warn(d.msg, FutureWarning)
/Users/Ted/miniconda3/lib/python3.8/site-packages/pandas/_config/config.py:630:
FutureWarning:
: boolean
    use_inf_as_null had been deprecated and will be removed in a future
    version. Use `use_inf_as_na` instead.

  warnings.warn(d.msg, FutureWarning)
```

1.2 3. Data Types and Missing Values

Exercise 1

What type of object is returned from the `dtypes` attribute?

[6]:
```
# a Series
type(bikes.dtypes)
```

[6]:
```
pandas.core.series.Series
```

Exercise 2

What type of object is returned from the `shape` attribute?

[7]:
```
# a tuple of rows, columns
type(bikes.shape)
```

[7]:
```
tuple
```

Exercise 3

The memory usage from the `info` method isn't correct when you have objects in your DataFrame. Read the docstrings from it and get the true memory usage.

[8]:
```
bikes.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50089 entries, 0 to 50088
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   gender             50089 non-null  object
 1   starttime          50089 non-null  object
 2   stoptime           50089 non-null  object
 3   tripduration       50089 non-null  int64
 4   from_station_name  50089 non-null  object
 5   start_capacity     50083 non-null  float64
 6   to_station_name    50089 non-null  object
 7   end_capacity       50077 non-null  float64
 8   temperature        50089 non-null  float64
 9   wind_speed         50089 non-null  float64
 10  events             50089 non-null  object
dtypes: float64(4), int64(1), object(6)
memory usage: 23.0 MB
```

1.3 4. Setting a meaningful index

Exercise 1

Read in the movie dataset and set the index to be something other than movie title. Are there any other good columns to use as an index?

[9]:
```
movies = pd.read_csv('../data/movie.csv', index_col='director_name')
movies.head(3)
```

director_name	title	year	color	content_rating	duration	...	plot_keywords	language	country	budget	imdb_score
James Cameron	Avatar	2009.0	Color	PG-13	178.0	...	avatar|future|marine|native|paraplegic	English	USA	237000000.0	7.9
Gore Verbinski	Pirates of the Caribbean: At World's End	2007.0	Color	PG-13	169.0	...	goddess|marriage ceremony|marriage proposal|pi...	English	USA	300000000.0	7.1
Sam Mendes	Spectre	2015.0	Color	PG-13	148.0	...	bomb|espionage|sequel|spy|terrorist	English	UK	245000000.0	6.8

3 rows × 21 columns

Director name isn't unique. There aren't any other good column names to use as an index.

Exercise 2

Use `set_index` to set the index and keep the column as part of the data. Read the docstrings to find the parameter that controls this functionality.

[10]:
```
movies = pd.read_csv('../data/movie.csv')
movies = movies.set_index('title', drop=False)
movies.head(3)
```

title	title	year	color	content_rating	duration	...	plot_keywords	language	country	budget	imdb_score
Avatar	Avatar	2009.0	Color	PG-13	178.0	...	avatar|future|marine|native|paraplegic	English	USA	237000000.0	7.9
Pirates of the Caribbean: At World's End	Pirates of the Caribbean: At World's End	2007.0	Color	PG-13	169.0	...	goddess|marriage ceremony|marriage proposal|pi...	English	USA	300000000.0	7.1
Spectre	Spectre	2015.0	Color	PG-13	148.0	...	bomb|espionage|sequel|spy|terrorist	English	UK	245000000.0	6.8

3 rows × 22 columns

Exercise 3

Read in the movie DataFrame and set the index as the title column. Assign the index to its own variable and output the last 10 movies titles.

[11]:
```
movies = pd.read_csv('../data/movie.csv', index_col='title')
index = movies.index
index[-10:]
```

[11]:
```
Index(['Primer', 'Cavite', 'El Mariachi', 'The Mongol King', 'Newlyweds',
       'Signed Sealed Delivered', 'The Following', 'A Plague So Pleasant',
       'Shanghai Calling', 'My Date with Drew'],
      dtype='object', name='title')
```

Exercise 4

Use an integer instead of the column name for `index_col` when reading in the data using `read_csv`. What does it do?

[12]:
```
movies = pd.read_csv('../data/movie.csv', index_col=-5)
movies.head(3)
```

plot_keywords	title	year	color	content_rating	duration	...	num_voted_users	language	country	budget	imdb_score
avatar|future|marine|native|paraplegic	Avatar	2009.0	Color	PG-13	178.0	...	886204	English	USA	237000000.0	7.9
goddess|marriage ceremony|marriage proposal|pirate|singapore	Pirates of the Caribbean: At World's End	2007.0	Color	PG-13	169.0	...	471220	English	USA	300000000.0	7.1
bomb|espionage|sequel|spy|terrorist	Spectre	2015.0	Color	PG-13	148.0	...	275868	English	UK	245000000.0	6.8

3 rows × 21 columns

It chooses the column name with that integer location.

Part II

Selecting Subsets of Data

Chapter 2

Solutions

2.1 1. Selecting Subsets of Data from DataFrames with just the brackets

[1]:
```
import pandas as pd
movie = pd.read_csv('../data/movie.csv', index_col='title')
```

Exercise 1

Select the column with the director's name as a Series

[2]:
```
movie['director_name'].head(3)
```

[2]:
```
title
Avatar                                         James Cameron
Pirates of the Caribbean: At World's End    Gore Verbinski
Spectre                                         Sam Mendes
Name: director_name, dtype: object
```

Exercise 2

Select the column with the director's name and number of Facebook likes.

[3]:
```
movie[['director_name', 'director_fb']].head(3)
```

title	director_name	director_fb
Avatar	James Cameron	0.0
Pirates of the Caribbean: At World's End	Gore Verbinski	563.0
Spectre	Sam Mendes	0.0

Exercise 3

Select a single column as a DataFrame and not a Series

[4]:
```
# make a one item list
col = ['director_name']
movie[col].head(3)
```

title	director_name
Avatar	James Cameron
Pirates of the Caribbean: At World's End	Gore Verbinski
Spectre	Sam Mendes

2.2 2. Selecting Subsets of Data from DataFrames with `loc`

[5]:
```
movie = pd.read_csv('../data/movie.csv', index_col='title')
movie.head(3)
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
Avatar	2009.0	Color	PG-13	178.0	James Cameron	...	avatar|future|marine|native|paraplegic	English	USA	237000000.0	7.9
Pirates of the Caribbean: At World's End	2007.0	Color	PG-13	169.0	Gore Verbinski	...	goddess|marriage ceremony|marriage proposal|pi...	English	USA	300000000.0	7.1
Spectre	2015.0	Color	PG-13	148.0	Sam Mendes	...	bomb|espionage|sequel|spy|terrorist	English	UK	245000000.0	6.8

3 rows × 21 columns

Exercise 1

Select columns `actor1`, `actor2`, and `actor3` for the movies 'Home Alone' and 'Top Gun'.

[6]:
```
rows = ['Home Alone', 'Top Gun']
cols = ['actor1', 'actor2', 'actor3']
movie.loc[rows, cols]
```

title	actor1	actor2	actor3
Home Alone	Macaulay Culkin	Devin Ratray	Catherine O'Hara
Top Gun	Tom Cruise	Tom Skerritt	Adrian Pasdar

Exercise 2

Select columns `actor1`, `actor2`, and `actor3` for all of the movies beginning at 'Home Alone' and ending at 'Top Gun'.

[7]:
```
cols = ['actor1', 'actor2', 'actor3']
movie.loc['Home Alone':'Top Gun', cols]
```

title	actor1	actor2	actor3
Home Alone	Macaulay Culkin	Devin Ratray	Catherine O'Hara
3 Men and a Baby	Tom Selleck	Ted Danson	Steve Guttenberg
Tootsie	Bill Murray	Sydney Pollack	Teri Garr
Top Gun	Tom Cruise	Tom Skerritt	Adrian Pasdar

Exercise 3

Select just the `director_name` column for the movies 'Home Alone' and 'Top Gun'.

[8]:
```
rows = ['Home Alone', 'Top Gun']
movie.loc[rows, 'director_name']
```

[8]:
```
title
Home Alone    Chris Columbus
Top Gun           Tony Scott
Name: director_name, dtype: object
```

Exercise 4

Repeat exercise 3, but return a DataFrame instead.

[9]:
```
rows = ['Home Alone', 'Top Gun']
cols = ['director_name']
movie.loc[rows, cols]
```

	director_name
title	
Home Alone	Chris Columbus
Top Gun	Tony Scott

Exercise 5

Select all columns for the movie 'The Dark Knight Rises'.

[10]:
```
movie.loc['The Dark Knight Rises', :]
```

[10]:
```
year                                                    2012.0
color                                                    Color
content_rating                                           PG-13
duration                                                 164.0
director_name                                  Christopher Nolan
director_fb                                              22000.0
actor1                                                Tom Hardy
actor1_fb                                                27000.0
actor2                                           Christian Bale
actor2_fb                                                23000.0
actor3                                         Joseph Gordon-Levitt
actor3_fb                                                23000.0
gross                                                  448130642.0
genres                                          Action|Thriller
num_reviews                                              813.0
num_voted_users                                         1144337
plot_keywords      deception|imprisonment|lawlessness|police offi…
language                                                 English
country                                                      USA
```

```
budget                                                    250000000.0
imdb_score                                                      8.5
Name: The Dark Knight Rises, dtype: object
```

Alternatively, don't include the empty slice.

[11]:
```
# movie.loc['The Dark Knight Rises']
```

Exercise 6

Repeat exercise 5 but return a DataFrame instead.

[12]:
```
movie.loc[['The Dark Knight Rises'], :]
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
The Dark Knight Rises	2012.0	Color	PG-13	164.0	Christopher Nolan	...	deception|imprisonment|lawlessness|police offi...	English	USA	250000000.0	8.5

1 rows × 21 columns

Exercise 7

Select all columns for the movies 'Tangled' and 'Avatar'.

[13]:
```
rows = ['Tangled', 'Avatar']
movie.loc[rows, :]
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
Tangled	2010.0	Color	PG	100.0	Nathan Greno	...	17th century|based on fairy tale|disney|flower...	English	USA	260000000.0	7.8
Avatar	2009.0	Color	PG-13	178.0	James Cameron	...	avatar|future|marine|native|paraplegic	English	USA	237000000.0	7.9

2 rows × 21 columns

Alternatively, omit the empty slice.

[14]:
```
# movie.loc[rows]
```

Exercise 8

What year was 'Tangled' and 'Avatar' made and what was their IMBD scores?

[15]:
```
movie.loc[['Tangled', 'Avatar'], ['year', 'imdb_score']]
```

title	year	imdb_score
Tangled	2010.0	7.8
Avatar	2009.0	7.9

Exercise 9

What is the data type of the `year` column?

[16]:
```
movie.dtypes.head() # Int64
```

[16]:
```
year              float64
color              object
content_rating     object
duration          float64
director_name      object
dtype: object
```

Exercise 10

Use a single method to output the data type and number of non-missing values of `year`. Is it missing any?

[17]:
```
# yes, it's missing many values. 4810 non-missing vs 4916 total
movie.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4916 entries, Avatar to My Date with Drew
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   year             4810 non-null   float64
 1   color            4897 non-null   object
 2   content_rating   4616 non-null   object
 3   duration         4901 non-null   float64
 4   director_name    4814 non-null   object
 5   director_fb      4814 non-null   float64
 6   actor1           4909 non-null   object
 7   actor1_fb        4909 non-null   float64
 8   actor2           4903 non-null   object
 9   actor2_fb        4903 non-null   float64
 10  actor3           4893 non-null   object
 11  actor3_fb        4893 non-null   float64
 12  gross            4054 non-null   float64
 13  genres           4916 non-null   object
 14  num_reviews      4867 non-null   float64
 15  num_voted_users  4916 non-null   int64
 16  plot_keywords    4764 non-null   object
 17  language         4904 non-null   object
 18  country          4911 non-null   object
 19  budget           4432 non-null   float64
 20  imdb_score       4916 non-null   float64
dtypes: float64(10), int64(1), object(10)
memory usage: 974.0+ KB
```

Exercise 11

Select every 300th movie between 'Tangled' and 'Forrest Gump'. Why doesn't 'Forrest Gump' appear in the results?

[18]:
```
# Forrest Gump is not a multiple of 300 away from Tangled
movie.loc['Tangled':'Forrest Gump':300]
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
Tangled	2010.0	Color	PG	100.0	Nathan Greno	...	17th century|based on fairy tale|disney|flower...	English	USA	260000000.0	7.8
Cloud Atlas	2012.0	Color	R	172.0	Tom Tykwer	...	composer|future|letter|nonlinear timeline|nurs...	English	Germany	102000000.0	7.5
Doom	2005.0	Color	R	113.0	Andrzej Bartkowiak	...	commando unit|extra chromosome|first person sh...	English	UK	60000000.0	5.2

3 rows × 21 columns

2.3 3. Selecting Subsets of Data from DataFrames with `iloc`

Exercise 1

Select the columns with integer location 10, 5, and 1.

[19]:
```
movie.iloc[:, [10, 5, 1]].head(3)
```

title	actor3	director_fb	color
Avatar	Wes Studi	0.0	Color
Pirates of the Caribbean: At World's End	Jack Davenport	563.0	Color
Spectre	Stephanie Sigman	0.0	Color

Exercise 2

Select the rows with integer location 10, 5, and 1.

[20]:
```
movie.iloc[[10, 5, 1], :]
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
Batman v Superman: Dawn of Justice	2016.0	Color	PG-13	183.0	Zack Snyder	...	based on comic book|batman|sequel to a reboot|...	English	USA	250000000.0	6.9
John Carter	2012.0	Color	PG-13	132.0	Andrew Stanton	...	alien|american civil war|male nipple|mars|prin...	English	USA	263700000.0	6.6
Pirates of the Caribbean: At World's End	2007.0	Color	PG-13	169.0	Gore Verbinski	...	goddess|marriage ceremony|marriage proposal|pi...	English	USA	300000000.0	7.1

3 rows × 21 columns

Exercise 3

Select rows with integer location 100 to 104 along with the column integer location 5.

[21]:
```
movie.iloc[100:105, 5]
```

[21]:
```
title
The Fast and the Furious                    357.0
The Curious Case of Benjamin Button       21000.0
X-Men: First Class                          905.0
The Hunger Games: Mockingjay - Part 2       508.0
The Sorcerer's Apprentice                   226.0
Name: director_fb, dtype: float64
```

Exercise 4

Select the value at row integer location 100 and column integer location 4.

[22]:
```
movie.iloc[100, 4]
```

[22]:
```
'Rob Cohen'
```

Exercise 5

Return the result of exercise 4 as a DataFrame.

[23]:
```
movie.iloc[[100], [4]]
```

title	director_name
The Fast and the Furious	Rob Cohen

Exercise 6

Select the last 5 rows of the last 5 columns.

[24]:
```
movie.iloc[-5:, -5:]
```

title	plot_keywords	language	country	budget	imdb_score
Signed Sealed Delivered	fraud|postal worker|prison|theft|trial	English	Canada	NaN	7.7
The Following	cult|fbi|hideout|prison escape|serial killer	English	USA	NaN	7.5
A Plague So Pleasant	NaN	English	USA	1400.0	6.3
Shanghai Calling	NaN	English	USA	NaN	6.3
My Date with Drew	actress name in title|crush|date|four word tit...	English	USA	1100.0	6.6

Exercise 7

Select every 25th row between rows with integer location 100 and 200 along with every fifth column.

[25]:
```
movie.iloc[100:200:25, ::5]
```

title	year	director_fb	actor3	num_voted_users	imdb_score
The Fast and the Furious	2001.0	357.0	Jordana Brewster	272223	6.7
Frozen	2013.0	69.0	Livvy Stubenrauch	421658	7.6
Armageddon	1998.0	0.0	Will Patton	322395	6.6
The Incredible Hulk	2008.0	255.0	William Hurt	326286	6.8

Exercise 8

Select the column with integer location 7 as a Series.

[26]:
```
movie.iloc[:, 7].head(3)
```

[26]:
```
title
Avatar                                         1000.0
Pirates of the Caribbean: At World's End      40000.0
```

```
Spectre                                          11000.0
Name: actor1_fb, dtype: float64
```

Exercise 9

Select the rows with integer location 999, 99, and 9 and the columns with integer location 9 and 19.

[27]:
```
rows = [999, 99, 9]
cols = [9, 19]
movie.iloc[rows, cols]
```

title	actor2_fb	budget
The Iron Giant	631.0	70000000.0
The Hobbit: An Unexpected Journey	972.0	180000000.0
Harry Potter and the Half-Blood Prince	11000.0	250000000.0

2.4 4. Selecting Subsets of Data from a Series

[28]:
```
import pandas as pd
movie = pd.read_csv('../data/movie.csv', index_col='title')
duration = movie['duration']
duration.head()
```

[28]:
```
title
Avatar                                         178.0
Pirates of the Caribbean: At World's End       169.0
Spectre                                        148.0
The Dark Knight Rises                          164.0
Star Wars: Episode VII - The Force Awakens       NaN
Name: duration, dtype: float64
```

Exercise 1

How long was the movie 'Titanic'?

[29]:
```
duration.loc['Titanic']
```

[29]:
```
194.0
```

Exercise 2

How long was the movie at the 999th integer location?

[30]:
```
duration.iloc[999]
```

[30]:
```
90.0
```

Exercise 3

Select the duration for the movies 'Hulk', 'Toy Story', and 'Cars'.

[31]:
```
names = ['Hulk', 'Toy Story', 'Cars']
duration.loc[names]
```

[31]:
```
title
Hulk         138.0
Toy Story     74.0
Cars         117.0
Name: duration, dtype: float64
```

Exercise 4

Select the duration for every 100th movies from 'Hulk' to 'Cars'.

[32]:
```
duration.loc['Hulk':'Cars':100]
```

[32]:
```
title
Hulk                     138.0
Live Free or Die Hard    129.0
Valkyrie                 121.0
Yogi Bear                 80.0
Dr. Dolittle 2            87.0
Name: duration, dtype: float64
```

Exercise 5

Select the duration for every 10th movie beginning from the 100th from the end.

[33]:
```
duration.iloc[-100::10]
```

[33]:
```
title
Antarctic Edge: 70° South                72.0
A Dog's Breakfast                        88.0
Penitentiary                             99.0
Peace, Propaganda & the Promised Land    80.0
Supporting Characters                    87.0
Bending Steel                            92.0
Run, Hide, Die                           75.0
Manito                                   79.0
Breaking Upwards                         88.0
Primer                                   77.0
Name: duration, dtype: float64
```

Read in the bikes dataset

Read in the bikes dataset and select the `wind_speed` column by executing the cell below and use it for the rest of the exercises. Notice that the index labels are integers, meaning that when you use `loc` you

will be using integers.

[34]:
```
bikes = pd.read_csv('../data/bikes.csv')
wind = bikes['wind_speed']
wind.head()
```

[34]:
```
0    12.7
1     6.9
2    16.1
3    16.1
4    17.3
Name: wind_speed, dtype: float64
```

Exercise 6

What type of index does the `wind` Series have?

It has a `RangeIndex`

[35]:
```
wind.index
```

[35]:
```
RangeIndex(start=0, stop=50089, step=1)
```

Exercise 7

From the `wind` Series, select the integer locations 4 through, but not including 10.

[36]:
```
wind.iloc[4:10]
```

[36]:
```
4    17.3
5    17.3
6    15.0
7     5.8
8     0.0
9    12.7
Name: wind_speed, dtype: float64
```

Exercise 8

Copy and paste your answer to Exercise 7 below but use `loc` instead. Do you get the same result? Why not?

[37]:
```
wind.loc[4:10]
```

[37]:
```
4     17.3
5     17.3
6     15.0
7      5.8
8      0.0
9     12.7
```

```
10     9.2
Name: wind_speed, dtype: float64
```

This is tricky - the index in this case contains integers and not strings. So the labels themselves are also integers and happen to be the same integers corresponding to integer location. The reason `.iloc` and `.loc` produce different results is that `.loc` always includes the last value when slicing.

2.5 5. Boolean Selection Single Conditions

[38]:
```
bikes = pd.read_csv('../data/bikes.csv')
bikes.head(3)
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events
0	Male	2013-06-28 19:01:00	2013-06-28 19:17:00	993	Lake Shore Dr & Monroe St	...	Michigan Ave & Oak St	15.0	73.9	12.7	mostlycloudy
1	Male	2013-06-28 22:53:00	2013-06-28 23:03:00	623	Clinton St & Washington Blvd	...	Wells St & Walton St	19.0	69.1	6.9	partlycloudy
2	Male	2013-06-30 14:43:00	2013-06-30 15:01:00	1040	Sheffield Ave & Kingsbury St	...	Dearborn St & Monroe St	23.0	73.0	16.1	mostlycloudy

3 rows × 11 columns

Exercise 1

Find all the rides with temperature below 0.

[39]:
```
filt = bikes['temperature'] < 0
bikes[filt].head(3)
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events
1871	Male	2013-12-12 05:13:00	2013-12-12 05:27:00	878	California Ave & North Ave	...	Carpenter St & Huron St	19.0	-2.0	6.9	mostlycloudy
2049	Female	2014-01-23 06:15:00	2014-01-23 06:29:00	828	Stave St & Armitage Ave	...	Ashland Ave & Division St	19.0	-2.0	16.1	partlycloudy
2054	Male	2014-01-23 21:15:00	2014-01-23 21:21:00	351	LaSalle St & Illinois St	...	McClurg Ct & Illinois St	31.0	-0.9	12.7	clear

3 rows × 11 columns

Exercise 2

Find all the rides with wind speed greater than 30.

[40]:
```
filt = bikes['wind_speed'] > 30
bikes[filt].head(3)
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events
2164	Male	2014-02-20 19:06:00	2014-02-20 19:07:00	66	Pine Grove Ave & Waveland Ave	...	Pine Grove Ave & Waveland Ave	23.0	46.9	31.1	mostlycloudy
2165	Male	2014-02-20 20:47:00	2014-02-20 21:14:00	1605	Millennium Park	...	Clark St & Wrightwood Ave	15.0	39.0	35.7	cloudy
2479	Male	2014-04-01 08:28:00	2014-04-01 08:29:00	82	Desplaines St & Kinzie St	...	Desplaines St & Kinzie St	19.0	33.1	31.1	mostlycloudy

3 rows × 11 columns

Exercise 3

Find all the rides that began from station 'Millennium Park'.

[41]:
```
bikes['from_station_name'] == 'Millennium Park'
bikes[filt].head(3)
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events
2164	Male	2014-02-20 19:06:00	2014-02-20 19:07:00	66	Pine Grove Ave & Waveland Ave	...	Pine Grove Ave & Waveland Ave	23.0	46.9	31.1	mostlycloudy
2165	Male	2014-02-20 20:47:00	2014-02-20 21:14:00	1605	Millennium Park	...	Clark St & Wrightwood Ave	15.0	39.0	35.7	cloudy
2479	Male	2014-04-01 08:28:00	2014-04-01 08:29:00	82	Desplaines St & Kinzie St	...	Desplaines St & Kinzie St	19.0	33.1	31.1	mostlycloudy

3 rows × 11 columns

Exercise 4

Find all the rides with wind speed less than 0. How is this possible?

[42]:
```
# these are likely missing values.
# The dataset uses -9999 to represent missing values
filt = bikes['wind_speed'] < 0
bikes[filt].head(3)
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events
22990	Male	2016-03-19 10:08:00	2016-03-19 10:20:00	702	Wood St & Division St	...	Campbell Ave & Fullerton Ave	15.0	42.8	-9999.0	mostlycloudy
27168	Female	2016-06-30 11:47:00	2016-06-30 11:51:00	240	Kimball Ave & Belmont Ave	...	Avers Ave & Belmont Ave	19.0	-9999.0	-9999.0	unknown
28368	Female	2016-07-21 21:02:29	2016-07-21 21:20:28	1079	Sedgwick St & North Ave	...	Ashland Ave & Division St	19.0	73.0	-9999.0	tstorms

3 rows × 11 columns

Exercise 5

Find all the rides where the starting number of bikes at the station (start_capacity) was more than 50.

[43]:
```
filt = bikes['start_capacity'] > 50
bikes[filt].head()
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events
36122	Male	2017-02-17 17:00:36	2017-02-17 17:23:27	1371	Field Museum	...	Lake Shore Dr & North Blvd	39.0	63.0	10.4	partlycloudy
37617	Male	2017-04-14 18:44:47	2017-04-14 19:00:53	966	Field Museum	...	Burnham Harbor	23.0	63.0	4.6	cloudy
37920	Male	2017-04-22 12:28:51	2017-04-22 12:44:14	923	Field Museum	...	Indiana Ave & Roosevelt Rd	39.0	55.9	12.7	mostlycloudy
37940	Female	2017-04-22 17:12:55	2017-04-22 17:23:44	649	Field Museum	...	Wabash Ave & Roosevelt Rd	23.0	57.9	12.7	partlycloudy
39102	Male	2017-05-21 20:40:10	2017-05-21 20:51:29	679	Field Museum	...	Buckingham Fountain	27.0	51.1	12.7	cloudy

5 rows × 11 columns

Exercise 6

Did any rides happen in temperature over 100 degrees?

[44]:
```
# no, a dataframe with no rows is returned.
filt = bikes['temperature'] > 100
bikes[filt]
```

gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events

0 rows × 11 columns

Read in new data

Read in the movie dataset by executing the cell below and use it for the following exercises.

[45]:
```
import pandas as pd
movie = pd.read_csv('../data/movie.csv', index_col='title')
movie.head(3)
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
Avatar	2009.0	Color	PG-13	178.0	James Cameron	...	avatar|future|marine|native|paraplegic	English	USA	237000000.0	7.9
Pirates of the Caribbean: At World's End	2007.0	Color	PG-13	169.0	Gore Verbinski	...	goddess|marriage ceremony|marriage proposal|pi...	English	USA	300000000.0	7.1
Spectre	2015.0	Color	PG-13	148.0	Sam Mendes	...	bomb|espionage|sequel|spy|terrorist	English	UK	245000000.0	6.8

3 rows × 21 columns

Exercise 7

Select all movies that have 'Tom Hanks' as `actor1`. How many of these movies has he starred in?

[46]:
```
filt = movie['actor1'] == 'Tom Hanks'
hanks_movies = movie[filt]
hanks_movies.head(3)
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
Toy Story 3	2010.0	Color	G	103.0	Lee Unkrich	...	college|day care|escape|teddy bear|toy	English	USA	200000000.0	8.3
The Polar Express	2004.0	Color	G	100.0	Robert Zemeckis	...	boy|christmas|christmas eve|north pole|train	English	USA	165000000.0	6.6
Angels & Demons	2009.0	Color	PG-13	146.0	Ron Howard	...	conclave|illuminati|murder|reference to bernin...	English	USA	150000000.0	6.7

3 rows × 21 columns

He's starred in 24 movies

[47]:
```
hanks_movies.shape
```

[47]:
```
(24, 21)
```

Exercise 8

Select movies with an IMDB score greater than 9.

[48]:
```
filt = movie['imdb_score'] > 9
movie[filt]
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
The Shawshank Redemption	1994.0	Color	R	142.0	Frank Darabont	...	escape from prison|first person narration|pris...	English	USA	25000000.0	9.3
Towering Inferno	NaN	Color	NaN	65.0	John Blanchard	...	NaN	English	Canada	NaN	9.5
Dekalog	NaN	Color	TV-MA	55.0	NaN	...	meaning of life|moral challenge|morality|searc...	Polish	Poland	NaN	9.1
The Godfather	1972.0	Color	R	175.0	Francis Ford Coppola	...	crime family|mafia|organized crime|patriarch|r...	English	USA	6000000.0	9.2
Kickboxer: Vengeance	2016.0	NaN	NaN	90.0	John Stockwell	...	NaN	NaN	USA	17000000.0	9.1

5 rows × 21 columns

Exercise 9

Write a function that accepts a single parameter to find the number of movies for a given content rating. Use the function to find the number of movies for ratings 'R', 'PG-13', and 'PG'.

[49]:
```
def count_rating(rating):
    filt = movie['content_rating'] == rating
    count = len(movie[filt])
    return f'There are {count} movies rated {rating}'
```

[50]:
```
count_rating('R')
```

[50]:
```
'There are 2067 movies rated R'
```

[51]:
```
count_rating('PG-13')
```

[51]:
```
'There are 1411 movies rated PG-13'
```

[52]:
```
count_rating('PG')
```

[52]:
```
'There are 686 movies rated PG'
```

2.6 6. Boolean Selection Multiple Conditions

[53]:
```
import pandas as pd
bikes = pd.read_csv('../data/bikes.csv')
bikes.head(3)
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events
0	Male	2013-06-28 19:01:00	2013-06-28 19:17:00	993	Lake Shore Dr & Monroe St	...	Michigan Ave & Oak St	15.0	73.9	12.7	mostlycloudy
1	Male	2013-06-28 22:53:00	2013-06-28 23:03:00	623	Clinton St & Washington Blvd	...	Wells St & Walton St	19.0	69.1	6.9	partlycloudy
2	Male	2013-06-30 14:43:00	2013-06-30 15:01:00	1040	Sheffield Ave & Kingsbury St	...	Dearborn St & Monroe St	23.0	73.0	16.1	mostlycloudy

3 rows × 11 columns

Exercise 1

Find all the rides where temperature was between 0 and 2.

[54]:
```
filt1 = bikes['temperature'] >= 0
filt2 = bikes['temperature'] <= 2
filt = filt1 & filt2
bikes[filt].head(3)
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events
1870	Female	2013-12-11 21:13:00	2013-12-11 21:19:00	347	Southport Ave & Roscoe St	...	Ashland Ave & Grace St	15.0	1.9	10.4	clear
1938	Male	2013-12-23 22:29:00	2013-12-23 22:37:00	455	State St & Pearson St	...	Clark St & Schiller St	19.0	-0.0	15.0	partlycloudy
2039	Male	2014-01-21 08:48:00	2014-01-21 09:13:00	1547	Damen Ave & Pierce Ave	...	Franklin St & Jackson Blvd	31.0	1.9	19.6	partlycloudy

3 rows × 11 columns

Exercise 2

Find all the rides with trip duration less than 100 done by females.

[55]:
```
filt1 = bikes['tripduration'] < 100
filt2 = bikes['gender'] == 'Female'
filt = filt1 & filt2
bikes[filt].head(3)
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events
2485	Female	2014-04-01 15:43:00	2014-04-01 15:44:00	67	Halsted St & Polk St	...	Halsted St & Polk St	19.0	46.0	23.0	partlycloudy
3366	Female	2014-05-17 14:55:00	2014-05-17 14:56:00	62	Damen Ave & Chicago Ave	...	Damen Ave & Chicago Ave	15.0	57.9	11.5	partlycloudy
3629	Female	2014-05-27 07:55:00	2014-05-27 07:56:00	70	Clark St & Schiller St	...	Clark St & Schiller St	19.0	73.0	9.2	cloudy

3 rows × 11 columns

Exercise 3

Find all the rides from 'Daley Center Plaza' to 'Michigan Ave & Washington St'.

[56]:
```
filt1 = bikes['from_station_name'] == 'Daley Center Plaza'
filt2 = bikes['to_station_name'] == 'Michigan Ave & Washington St'
filt = filt1 & filt2
bikes[filt]
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events
6276	Female	2014-08-04 17:18:00	2014-08-04 17:22:00	214	Daley Center Plaza	...	Michigan Ave & Washington St	43.0	73.0	8.1	cloudy
31038	Female	2016-09-07 07:48:30	2016-09-07 07:53:27	297	Daley Center Plaza	...	Michigan Ave & Washington St	43.0	79.0	10.4	cloudy
46116	Male	2017-09-13 16:29:00	2017-09-13 16:32:42	222	Daley Center Plaza	...	Michigan Ave & Washington St	43.0	72.0	9.2	cloudy

3 rows × 11 columns

Exercise 4

Find all the rides with temperature greater than 90 or trip duration greater than 2000 or wind speed greater than 20.

[57]:
```
filt1 = bikes['temperature'] > 90
filt2 = bikes['tripduration'] > 2000
filt3 = bikes['wind_speed'] > 20
filt = filt1 | filt2 | filt3
bikes[filt].head(3)
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events
12	Male	2013-07-05 10:02:00	2013-07-05 10:40:00	2263	Jefferson St & Monroe St	...	Jefferson St & Monroe St	19.0	79.0	0.0	partlycloudy
18	Male	2013-07-09 13:12:00	2013-07-09 14:42:00	5396	Canal St & Jackson Blvd	...	Millennium Park	35.0	79.0	13.8	cloudy
40	Female	2013-07-14 14:08:00	2013-07-14 15:53:00	6274	Wabash Ave & Roosevelt Rd	...	Lake Shore Dr & Monroe St	11.0	87.1	8.1	partlycloudy

3 rows × 11 columns

Exercise 5

Invert the condition from exercise 4.

[58]:
```
filt1 = bikes['temperature'] > 90
filt2 = bikes['tripduration'] > 2000
filt3 = bikes['wind_speed'] > 20
filt = filt1 | filt2 | filt3
bikes[~filt].head(3)
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events
0	Male	2013-06-28 19:01:00	2013-06-28 19:17:00	993	Lake Shore Dr & Monroe St	...	Michigan Ave & Oak St	15.0	73.9	12.7	mostlycloudy
1	Male	2013-06-28 22:53:00	2013-06-28 23:03:00	623	Clinton St & Washington Blvd	...	Wells St & Walton St	19.0	69.1	6.9	partlycloudy
2	Male	2013-06-30 14:43:00	2013-06-30 15:01:00	1040	Sheffield Ave & Kingsbury St	...	Dearborn St & Monroe St	23.0	73.0	16.1	mostlycloudy

3 rows × 11 columns

Exercise 6

Are there any rides where the weather event was snow and the temperature was greater than 40?

[59]:
```
# no
filt1 = bikes['events'] == 'snow'
filt2 = bikes['temperature'] > 40
filt = filt1 & filt2
bikes[filt]
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events

0 rows × 11 columns

Read in the movie dataset by executing the cell below and use it for the following exercises.

[60]:
```
import pandas as pd
movie = pd.read_csv('../data/movie.csv', index_col='title')
```

```
movie.head(3)
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
Avatar	2009.0	Color	PG-13	178.0	James Cameron	...	avatar|future|marine|native|paraplegic	English	USA	237000000.0	7.9
Pirates of the Caribbean: At World's End	2007.0	Color	PG-13	169.0	Gore Verbinski	...	goddess|marriage ceremony|marriage proposal|pi...	English	USA	300000000.0	7.1
Spectre	2015.0	Color	PG-13	148.0	Sam Mendes	...	bomb|espionage|sequel|spy|terrorist	English	UK	245000000.0	6.8

3 rows × 21 columns

Exercise 7

Select all movies with an IMDB score between 8 and 9.

[61]:
```
filt1 = movie['imdb_score'] >= 8
filt2 = movie['imdb_score'] <= 9
filt = filt1 & filt2
movie[filt].head(3)
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
The Dark Knight Rises	2012.0	Color	PG-13	164.0	Christopher Nolan	...	deception|imprisonment|lawlessness|police offi...	English	USA	250000000.0	8.5
The Avengers	2012.0	Color	PG-13	173.0	Joss Whedon	...	alien invasion|assassin|battle|iron man|soldier	English	USA	220000000.0	8.1
Captain America: Civil War	2016.0	Color	PG-13	147.0	Anthony Russo	...	based on comic book|knife|marvel cinematic uni...	English	USA	250000000.0	8.2

3 rows × 21 columns

Exercise 8

Select all movies rated 'PG-13' that had IMDB scores between 8 and 9.

[62]:
```
filt1 = movie['imdb_score'] >= 8
filt2 = movie['imdb_score'] <= 9
filt3 = movie['content_rating'] == 'PG-13'
filt = filt1 & filt2 & filt3
movie[filt].head(3)
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
The Dark Knight Rises	2012.0	Color	PG-13	164.0	Christopher Nolan	...	deception|imprisonment|lawlessness|police offi...	English	USA	250000000.0	8.5
The Avengers	2012.0	Color	PG-13	173.0	Joss Whedon	...	alien invasion|assassin|battle|iron man|soldier	English	USA	220000000.0	8.1
Captain America: Civil War	2016.0	Color	PG-13	147.0	Anthony Russo	...	based on comic book|knife|marvel cinematic uni...	English	USA	250000000.0	8.2

3 rows × 21 columns

Exercise 9

Select movies that were rated either R, PG-13, or PG.

[63]:
```
filt = movie['content_rating'].isin(['R', 'PG-13', 'PG'])
movie[filt].head(3)
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
Avatar	2009.0	Color	PG-13	178.0	James Cameron	...	avatar|future|marine|native|paraplegic	English	USA	237000000.0	7.9
Pirates of the Caribbean: At World's End	2007.0	Color	PG-13	169.0	Gore Verbinski	...	goddess|marriage ceremony|marriage proposal|pi...	English	USA	300000000.0	7.1
Spectre	2015.0	Color	PG-13	148.0	Sam Mendes	...	bomb|espionage|sequel|spy|terrorist	English	UK	245000000.0	6.8

3 rows × 21 columns

Exercise 10

Select movies that are either rated PG-13 or had an IMDB score greater than 7.

[64]:
```
filt1 = movie['content_rating'] == 'PG-13'
filt2 = movie['imdb_score'] > 7
filt = filt1 | filt2
movie[filt].head(3)
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
Avatar	2009.0	Color	PG-13	178.0	James Cameron	...	avatar|future|marine|native|paraplegic	English	USA	237000000.0	7.9
Pirates of the Caribbean: At World's End	2007.0	Color	PG-13	169.0	Gore Verbinski	...	goddess|marriage ceremony|marriage proposal|pi...	English	USA	300000000.0	7.1
Spectre	2015.0	Color	PG-13	148.0	Sam Mendes	...	bomb|espionage|sequel|spy|terrorist	English	UK	245000000.0	6.8

3 rows × 21 columns

Exercise 11

Find all the movies that have at least one of the three actors with more than 10,000 Facebook likes.

[65]:
```
filt1 = movie['actor1_fb'] > 10000
filt2 = movie['actor2_fb'] > 10000
filt3 = movie['actor3_fb'] > 10000
filt = filt1 | filt2 | filt3
movie[filt].head(3)
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
Pirates of the Caribbean: At World's End	2007.0	Color	PG-13	169.0	Gore Verbinski	...	goddess|marriage ceremony|marriage proposal|pi...	English	USA	300000000.0	7.1
Spectre	2015.0	Color	PG-13	148.0	Sam Mendes	...	bomb|espionage|sequel|spy|terrorist	English	UK	245000000.0	6.8
The Dark Knight Rises	2012.0	Color	PG-13	164.0	Christopher Nolan	...	deception|imprisonment|lawlessness|police offi...	English	USA	250000000.0	8.5

3 rows × 21 columns

Exercise 12

Invert the condition from exercise 10. In words, what have you selected?

The following selects non-PG-13 movies with IMDB score 7 or less.

[66]:
```
filt1 = movie['content_rating'] == 'PG-13'
filt2 = movie['imdb_score'] > 7
filt = filt1 | filt2
movie[~filt].head(3)
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
The Chronicles of Narnia: Prince Caspian	2008.0	Color	PG	150.0	Andrew Adamson	...	brother brother relationship|brother sister re...	English	USA	225000000.0	6.6
Alice in Wonderland	2010.0	Color	PG	108.0	Tim Burton	...	alice in wonderland|mistaking reality for drea...	English	USA	200000000.0	6.5
Oz the Great and Powerful	2013.0	Color	PG	130.0	Sam Raimi	...	circus|magic|magician|oz|witch	English	USA	215000000.0	6.4

3 rows × 21 columns

Exercise 13

Select all movies from the 1970's.

[67]:
```
filt1 = movie['year'] >= 1970
filt2 = movie['year'] <= 1979
filt = filt1 & filt2
movie[filt].head(3)
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
All That Jazz	1979.0	Color	R	123.0	Bob Fosse	...	dancer|editing|stand up comedian|surgery|vomiting	English	USA	NaN	7.8
Superman	1978.0	Color	PG	188.0	Richard Donner	...	1970s|clark kent|planet|superhero|year 1978	English	USA	55000000.0	7.3
Solaris	1972.0	Black and White	PG	115.0	Andrei Tarkovsky	...	hallucination|ocean|psychologist|scientist|spa...	Russian	Soviet Union	1000000.0	8.1

3 rows × 21 columns

2.7 7. Boolean Selection More

[68]:
```
import pandas as pd
bikes = pd.read_csv('../data/bikes.csv')
```

Exercise 1

Select the wind speed column a a Series and assign it to a variable. Are there any negative wind speeds?

[69]:
```
wind = bikes['wind_speed']
wind.head(3)
```

[69]:
```
0    12.7
1     6.9
2    16.1
Name: wind_speed, dtype: float64
```

Yes, there is really strong negative wind! Or maybe its just bad data...

[70]:
```
filt = wind < 0
wind[filt].head(3)
```

[70]:
```
22990   -9999.0
27168   -9999.0
28368   -9999.0
Name: wind_speed, dtype: float64
```

Exercise 2

Select all wind speed values between 12 and 16.

[71]:
```
filt = wind.between(12, 16)
wind[filt].head(3)
```

[71]:
```
0    12.7
6    15.0
9    12.7
Name: wind_speed, dtype: float64
```

Exercise 3

Select the events and gender columns for all trip durations longer than 1,000 seconds.

[72]:
```
filt = bikes['tripduration'] > 1000
cols = ['events', 'gender']
bikes.loc[filt, cols].head(3)
```

	events	gender
2	mostlycloudy	Male
8	cloudy	Male
10	mostlycloudy	Male

Read in the movie dataset by executing the cell below and use it for the following exercises.

[73]:
```
import pandas as pd
movie = pd.read_csv('../data/movie.csv', index_col='title')
movie.head(3)
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
Avatar	2009.0	Color	PG-13	178.0	James Cameron	...	avatar|future|marine|native|paraplegic	English	USA	237000000.0	7.9
Pirates of the Caribbean: At World's End	2007.0	Color	PG-13	169.0	Gore Verbinski	...	goddess|marriage ceremony|marriage proposal|pi...	English	USA	300000000.0	7.1
Spectre	2015.0	Color	PG-13	148.0	Sam Mendes	...	bomb|espionage|sequel|spy|terrorist	English	UK	245000000.0	6.8

3 rows × 21 columns

Exercise 4

Select all the movies such that the Facebook likes for actor 2 are greater than those for actor 1.

There are none!

[74]:
```
filt = movie['actor2_fb'] > movie['actor2_fb']
movie[filt]
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score

0 rows × 21 columns

Exercise 5

Select the year, content rating, and IMDB score columns for movies from the year 2016 with IMDB score less than 4.

[75]:
```
filt1 = movie['year'] == 2016
filt2 = movie['imdb_score'] < 4
filt = filt1 & filt2
cols = ['year', 'content_rating', 'imdb_score']

movie.loc[filt, cols]
```

title	year	content_rating	imdb_score
Fifty Shades of Black	2016.0	R	3.5
Cabin Fever	2016.0	Not Rated	3.7
God's Not Dead 2	2016.0	PG	3.4

Exercise 6

Select all the movies that are missing values for content rating.

[76]:
```
filt = movie['content_rating'].isna()
movie[filt].head(3)
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
Star Wars: Episode VII - The Force Awakens	NaN	NaN	NaN	NaN	Doug Walker	...	NaN	NaN	NaN	NaN	7.1
Godzilla Resurgence	2016.0	Color	NaN	120.0	Hideaki Anno	...	blood|godzilla|monster|sequel	Japanese	Japan	NaN	8.2
Harry Potter and the Deathly Hallows: Part II	2011.0	Color	NaN	NaN	Matt Birch	...	NaN	English	UK	NaN	7.5

3 rows × 21 columns

Exercise 7

Select all the movies that are missing values for both the gross and budget columns. Return just those columns to verify that those values are indeed missing.

[77]:
```
filt = movie['gross'].isna() & movie['budget'].isna()
cols = ['gross', 'budget']
movie.loc[filt, cols].head(3)
```

title	gross	budget
Star Wars: Episode VII - The Force Awakens	NaN	NaN
The Lovers	NaN	NaN
Godzilla Resurgence	NaN	NaN

Exercise 8

Write a function `find_missing` that has three parameters, `df`, `col1` and `col2` where `df` is a DataFrame and `col1` and `col2` are column names. This function should return all the rows of the DataFrame where `col1` and `col2` are missing. Only return the two columns as well. Answer Exercise 7 with this function.

[78]:
```
def find_missing(df, col1, col2):
    filt = df[col1].isna() & df[col2].isna()
    cols = [col1, col2]

    return df.loc[filt, cols]
```

[79]:
```
movie_missing = find_missing(movie, 'gross', 'budget')
movie_missing.head(3)
```

title	gross	budget
Star Wars: Episode VII - The Force Awakens	NaN	NaN
The Lovers	NaN	NaN
Godzilla Resurgence	NaN	NaN

2.8 8. Filtering with the query Method

Exercise 1

Use the `query` method to select trip durations between 5,000 and 10,000.

[80]:
```
bikes.query('5000 <= tripduration <= 10000').head(3)
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events
18	Male	2013-07-09 13:12:00	2013-07-09 14:42:00	5396	Canal St & Jackson Blvd	...	Millennium Park	35.0	79.0	13.8	cloudy
40	Female	2013-07-14 14:08:00	2013-07-14 15:53:00	6274	Wabash Ave & Roosevelt Rd	...	Lake Shore Dr & Monroe St	11.0	87.1	8.1	partlycloudy
77	Female	2013-07-21 11:35:00	2013-07-21 13:54:00	8299	State St & 19th St	...	Sheffield Ave & Kingsbury St	15.0	82.9	5.8	mostlycloudy

3 rows × 11 columns

Exercise 2

Use the `query` method to select trip durations between 5,000 and 10,000 when the weather was snow or rain. Retrieve the same data with boolean selection.

[81]:
```
bikes.query('5000 <= tripduration <= 10000 and events in ["snow", "rain"]').head(3)
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events
8506	Male	2014-10-04 12:33:00	2014-10-04 14:06:00	5568	Halsted St & Diversey Pkwy	...	Halsted St & Wrightwood Ave	15.0	42.1	17.3	rain
13355	Male	2015-06-15 11:41:00	2015-06-15 13:43:00	7295	Racine Ave & Belmont Ave	...	Racine Ave & Belmont Ave	15.0	75.9	4.6	rain
22155	Male	2016-02-09 10:09:00	2016-02-09 12:28:00	8309	Wabash Ave & Roosevelt Rd	...	Museum Campus	35.0	16.0	16.1	snow

3 rows × 11 columns

[82]:
```
filt1 = bikes['tripduration'].between(5000, 10000)
filt2 = bikes['events'].isin(['snow', 'rain'])
filt = filt1 & filt2
bikes[filt].head(3)
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events
8506	Male	2014-10-04 12:33:00	2014-10-04 14:06:00	5568	Halsted St & Diversey Pkwy	...	Halsted St & Wrightwood Ave	15.0	42.1	17.3	rain
13355	Male	2015-06-15 11:41:00	2015-06-15 13:43:00	7295	Racine Ave & Belmont Ave	...	Racine Ave & Belmont Ave	15.0	75.9	4.6	rain
22155	Male	2016-02-09 10:09:00	2016-02-09 12:28:00	8309	Wabash Ave & Roosevelt Rd	...	Museum Campus	35.0	16.0	16.1	snow

3 rows × 11 columns

Exercise 3

Use the `query` method to select trip durations between 5,000 and 10,000 when it was snow or rain. Create a list outside of the `query` method to hold the weather and reference that variable with `@` within `query`.

[83]:
```
weather = ["snow", "rain"]
bikes.query('5000 <= tripduration <= 10000 and events in @weather').head(3)
```

	gender	starttime	stoptime	tripduration	from_station_name	...	to_station_name	end_capacity	temperature	wind_speed	events
8506	Male	2014-10-04 12:33:00	2014-10-04 14:06:00	5568	Halsted St & Diversey Pkwy	...	Halsted St & Wrightwood Ave	15.0	42.1	17.3	rain
13355	Male	2015-06-15 11:41:00	2015-06-15 13:43:00	7295	Racine Ave & Belmont Ave	...	Racine Ave & Belmont Ave	15.0	75.9	4.6	rain
22155	Male	2016-02-09 10:09:00	2016-02-09 12:28:00	8309	Wabash Ave & Roosevelt Rd	...	Museum Campus	35.0	16.0	16.1	snow

3 rows × 11 columns

Read in the movie dataset by executing the cell below and use it for the following exercises.

[84]:
```
import pandas as pd
pd.set_option('display.max_columns', 50)
movie = pd.read_csv('../data/movie.csv', index_col='title')
```

```
movie.head(3)
```

title	year	color	content_rating	duration	director_name	director_fb	actor1	actor1_fb	actor2	actor2_fb	actor3	actor3_fb	gross	genres	num_reviews	num_voted_users	plot_keywords	language	country	budget	imdb_score
Avatar	2009.0	Color	PG-13	178.0	James Cameron	0.0	CCH Pounder	1000.0	Joel David Moore	936.0	Wes Studi	855.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	723.0	886204	avatar|future|marine|native|paraplegic	English	USA	237000000.0	7.9
Pirates of the Caribbean: At World's End	2007.0	Color	PG-13	169.0	Gore Verbinski	563.0	Johnny Depp	40000.0	Orlando Bloom	5000.0	Jack Davenport	1000.0	309404152.0	Action|Adventure|Fantasy	302.0	471220	goddess|marriage ceremony|marriage proposal|pi...	English	USA	300000000.0	7.1
Spectre	2015.0	Color	PG-13	148.0	Sam Mendes	0.0	Christoph Waltz	11000.0	Rory Kinnear	393.0	Stephanie Sigman	161.0	200074175.0	Action|Adventure|Thriller	602.0	275868	bomb|espionage|sequel|spy|terrorist	English	UK	245000000.0	6.8

Exercise 4

Use the `query` method to find all movies where the total number of Facebook likes for all three actors is greater than 50,000.

[85]:
```
movie.query('actor1_fb + actor2_fb + actor3_fb > 50000').head(3)
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
The Dark Knight Rises	2012.0	Color	PG-13	164.0	Christopher Nolan	...	deception|imprisonment|lawlessness|police offi...	English	USA	250000000.0	8.5
Avengers: Age of Ultron	2015.0	Color	PG-13	141.0	Joss Whedon	...	artificial intelligence|based on comic book|ca...	English	USA	250000000.0	7.5
The Avengers	2012.0	Color	PG-13	173.0	Joss Whedon	...	alien invasion|assassin|battle|iron man|soldier	English	USA	220000000.0	8.1

3 rows × 21 columns

Exercise 5

Select all the movies where the number of user voters is less than 10 times the number of reviews.

[86]:
```
movie.query('num_voted_users < 10 * num_reviews').head(3)
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
Indignation	2016.0	Color	R	110.0	James Schamus	...	based on novel	Hebrew	USA	NaN	7.8
Pete's Dragon	2016.0	Color	PG	102.0	David Lowery	...	NaN	English	USA	65000000.0	7.3
Kicks	2016.0	Color	R	80.0	Justin Tipping	...	NaN	English	USA	NaN	7.8

3 rows × 21 columns

Exercise 6

Select all the movies made in the 1990's that were rated R with an IMDB score greater than 8.

[87]:
```
movie.query('1990 <= year <= 1999 and content_rating == "R" and imdb_score > 8').
↪head(3)
```

title	year	color	content_rating	duration	director_name	...	plot_keywords	language	country	budget	imdb_score
Terminator 2: Judgment Day	1991.0	Color	R	153.0	James Cameron	...	future|liquid metal|multiple cameos|sexy woman...	English	USA	102000000.0	8.5
Braveheart	1995.0	Color	R	178.0	Mel Gibson	...	14th century|legend|revolt|scotland|tyranny	English	USA	72000000.0	8.4
Saving Private Ryan	1998.0	Color	R	169.0	Steven Spielberg	...	army|invasion|killed in action|normandy|soldier	English	USA	70000000.0	8.6

3 rows × 21 columns

2.9 9. Miscellaneous Subset Selection

[88]:
```
import pandas as pd
bikes = pd.read_csv('../data/bikes.csv')
```

Exercise 1

Provide several example column names that are not possible to select using dot notation.

- `min` or any other pandas method
- `some column` or any name with spaces
- `4` or any other integer
- `3cities` or any string that begins with an integer
- `(1, 2, 3)` - a tuple or any other object that is not a string

Exercise 2

Use the `%time` magic function to compare the performance difference between `loc` and `at` and between `iloc` and `iat`.

[89]:
```
%time bikes.at[35103, 'tripduration']
```

```
CPU times: user 2.62 ms, sys: 300 µs, total: 2.92 ms
Wall time: 2.37 ms
```

[89]: 487

[90]:
```
%time bikes.loc[35103, 'tripduration']
```

```
CPU times: user 61 µs, sys: 0 ns, total: 61 µs
Wall time: 63.9 µs
```

[90]: 487

[91]:
```
%time bikes.iat[35103, 4]
```

```
CPU times: user 116 µs, sys: 0 ns, total: 116 µs
Wall time: 120 µs
```

[91]: 'Clinton St & Madison St'

[92]:
```
%time bikes.iloc[35103, 4]
```

```
CPU times: user 146 µs, sys: 1 µs, total: 147 µs
Wall time: 151 µs
```

[92]: 'Clinton St & Madison St'

[93]:
```
# numpy
values = bikes['tripduration'].values
%time values[35103]
```

```
CPU times: user 4 µs, sys: 0 ns, total: 4 µs
Wall time: 9.06 µs
```

[93]: 487