```
# Pre-requisite 1
# ---
# The first thing that we will do in this notebook is
# to import the pandas library for data manipulation.
# ---
# OUR CODE GOES BELOW
#
import pandas as pd


# Pre-requisite 2
# ---
# We will also import the numpy library which
# will allow us to perform scientific computations.
# ---
# For those who are not familiar with what a library is,
# A library is a collection of related pieces of code that
# have been compiled and stored together for reuse.
# ---
# OUR CODE GOES BELOW
#
import numpy as np
```

Discovery Step

Reading Data

```
# Example 1
# ---
# Loading a dataset (csv file) from a url
# ---
# Dataset url (csv file) = http://bit.ly/IrisDataset
# ---
# OUR CODE GOES BELOW
#

# Reading our csv file from the given url and storing it to a dataframe.
# A data frame is a two-dimensional data structure that is used to
# represent a table of data with rows and columns.
# ---
#
df = pd.read_csv("http://bit.ly/IrisDataset")

# Previewing the first 5 records
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```
# Example 2
# ---
# We can also load A CSV Into pandas as shown below
# ---
# Dataset url = http://bit.ly/CitiesDataset1
# ----
# Instructions:
# 1. Visit the above dataset url with a browser. Then download the file.
# 2. Within this notebook you will need to have the Table of contents left sidebar open.
#    If its not open, you can open the sidebar clicking at the top of the notebook View -> Table of Contents.
# 3. Once its open click on the Files tab within the sidebar (this is on the farmost right).
# 4. Then upload the downloaded file to this location/ Or drag the file
# 5. Once uploaded you will use the name of the file to open the file as shown below.
#    You can also store in a directory/folder then reference that directory while reading the file.
#    i.e. the file cities stored in a directory named finance, then the reference pd.read_csv("/finance/cities.csv")
# ---
#
```

```
# Let's read the cities csv file
df_cities = pd.read_csv("http://bit.ly/CitiesDataset1")

# Previewing the first five records
df_cities.head(10)
```

	city	country	latitude	longitude	temperature
0	Aalborg	Denmark	57.03	9.92	7.52
1	Aberdeen	United Kingdom	57.17	-2.08	8.10
2	Abisko	Sweden	63.35	18.83	0.20
3	Adana	Turkey	36.99	35.32	18.67
4	Albacete	Spain	39.00	-1.87	12.62
5	Algeciras	Spain	36.13	-5.47	17.38
6	Amiens	France	49.90	2.30	10.17
7	Amsterdam	Netherlands	52.35	4.92	8.93
8	Ancona	Italy	43.60	13.50	13.52
9	Andorra	Andorra	42.50	1.52	9.60

```
# Challenge 1a
# ---
# Question: Load the following hotels dataset and preview the first 5 records.
# Hint: Use a different dataframe that the one used in the example dataset
#       i.e. using the dataframe name "hotel_df" rather using the dataframe name "df"
#       If you use the same dataframe names for the examples and challenges you
#       might experience a crash.
#       We will be using this dataset for most of the challenges.
#     : To work quickly on theses challenges, copy paste example code and modify the your need.
# ---
# Dataset url (csv file) = https://bit.ly/HotelBookingsDB
# This data set contains booking information for a city hotel and a resort hotel,
# and includes information such as when the booking was made, length of stay,
# the number of adults, children, and/or babies, and the number of available
# parking spaces, among other things.
# ---
# OUR CODE GOES BELOW
#
hotels_df = pd.read_csv("https://bit.ly/HotelBookingsDB")

hotels_df.head()
```

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_c
0	Resort Hotel	0	342	2015	July	
1	Resort Hotel	0	737	2015	July	
2	Resort Hotel	0	7	2015	July	
3	Resort Hotel	0	13	2015	July	
4	Resort Hotel	0	14	2015	July	

5 rows × 32 columns

```
# Challenge 1b
# ---
# Question: Load a dataset (excel file) from the url below
# Hint: - Use the read_excel() function rather than using the read_csv() function
#       - Use a new dataframe windmill_df
# ---
# Dataset url = http://bit.ly/WindmillDataset
# ---
# OUR CODE GOES BELOW
#
```

```
windmill_df= pd.read_excel("http://bit.ly/WindmillDataset")

windmill_df.head()
```

	Windmill	Wind Speed (m/s)	Power output (MW)
0	a1	1.096875	0.000000
1	a2	1.231528	0.000000
2	a3	1.275139	0.005479
3	a4	1.365486	0.010104
4	a5	1.387778	0.010812

Data Structuring

Data Exploration

```
# Example 2a
# ---
# Determining the no. of records in the dataset
# NB: We will use the above loaded dataset in example 1
# ---
# OUR CODE GOES BELOW
#

# Shape returns no. or records/instances (left) and columns/variables (right)
#
df.shape
```

```
(150, 5)
```

```
# Challenge 2a
# ---
# Question: Determine the no. of records in the our hotels dataset.
# NB: .shape gives us the no. of records.
# ---
# OUR CODE GOES BELOW
#
hotels_df.shape
```

```
(119390, 32)
```

```
# Example 2b
# ---
# Previewing the last few records/instances of our dataset
# ---
# OUR CODE GOES BELOW
#
df.tail()
```

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

```
# Challenge 2c
# ---
# Question: Preview the last few records in the hotels dataset.
# Hint = Use the dataframe you created for this dataset.
# ---
# OUR CODE GOES BELOW
#
hotels_df.tail()
```

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arri
119385	City Hotel	0	23	2017	August	
119386	City Hotel	0	102	2017	August	
119387	City Hotel	0	34	2017	August	
119388	City Hotel	0	109	2017	August	
119389	City Hotel	0	205	2017	August	

5 rows × 32 columns

```
# Challenge 2d
# ---
# Question: Preview a sample of 10 records from the hotels dataset.
# Hint: Use the dataframe you created for this dataset.
#     : Use the sample() function, and add no. of desired records as the parameter.
# ---
# OUR CODE GOES BELOW
#
hotels_df.head(10)
```

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_in_w
0	Resort Hotel	0	342	2015	July	27	1	
1	Resort Hotel	0	737	2015	July	27	1	
2	Resort Hotel	0	7	2015	July	27	1	
3	Resort Hotel	0	13	2015	July	27	1	
4	Resort Hotel	0	14	2015	July	27	1	
5	Resort Hotel	0	14	2015	July	27	1	
6	Resort Hotel	0	0	2015	July	27	1	
7	Resort Hotel	0	9	2015	July	27	1	
8	Resort Hotel	1	85	2015	July	27	1	
9	Resort Hotel	1	75	2015	July	27	1	

10 rows × 32 columns

```
# Example 2c
# ---
# Checking the datatypes of df variables (columns)
# ---
# OUR CODE GOES BELOW
#
df.dtypes
```

```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

```
# Challenge 2e
# ---
# Question: Check the datatypes of the hotels dataset.
# ---
```

```
# OUR CODE GOES BELOW
#
hotels_df.dtypes
```

```
hotel                           object
is_canceled                      int64
lead_time                        int64
arrival_date_year                int64
arrival_date_month              object
arrival_date_week_number         int64
arrival_date_day_of_month        int64
stays_in_weekend_nights          int64
stays_in_week_nights             int64
adults                           int64
children                       float64
babies                           int64
meal                            object
country                         object
market_segment                  object
distribution_channel            object
is_repeated_guest                int64
previous_cancellations           int64
previous_bookings_not_canceled   int64
reserved_room_type              object
assigned_room_type              object
booking_changes                  int64
deposit_type                    object
agent                          float64
company                        float64
days_in_waiting_list             int64
customer_type                   object
adr                            float64
required_car_parking_spaces      int64
total_of_special_requests        int64
reservation_status              object
reservation_status_date         object
dtype: object
```

Data Structuring

Standardization

```
# Example 1
# ---
# Renaming column names
# ---
# Dataset url = http://bit.ly/DataCleaningDataset
# ---
# OUR CODE GOES BELOW
#

# Reading our dataset from the url
# ---
#
df = pd.read_csv('http://bit.ly/DataCleaningDataset')
df.head()
```

	NAME;CITY;COUNTRY;HEIGHT;WEIGHT;ACCOUNT A;ACCOUNT B;TOTAL ACCOUNT
0	Adi Dako ;LISBON ;PORTUGAL ;56;132;2390...
1	John Paul;LONDON ;UNITED KINGDOM;62;165;4500...
2	Cindy Jules;Stockholm;Sweden;48;117;;5504;8949
3	Arthur Kegels;BRUSSELS;BELGIUM;59;121;4344...
4	Freya Bismark; Berlin;GERMANYY;53;126;7000;19...

```
# We specify the character ; as our separator so that we can
# be able to ready the above file that has ";" as a separator
# for our columns. We should note that many dataset may note have
# such a structure, but if we come across this kind of a scenario
# this is how we would read our file.
# ---
#
df = pd.read_csv('http://bit.ly/DataCleaningDataset', ';')
df.head()
```

```
<ipython-input-34-504614e841ab>:8: FutureWarning: In a future version of pandas all arguments of read_csv except for the argument 'filep
  df = pd.read_csv('http://bit.ly/DataCleaningDataset', ';')
```

	NAME	CITY	COUNTRY	HEIGHT	WEIGHT	ACCOUNT A	ACCOUNT B	TOTAL ACCOUNT
0	Adi Dako	LISBON	PORTUGAL	56	132.0	2390.0	4340	6730
1	John Paul	LONDON	UNITED KINGDOM	62	165.0	4500.0	34334	38834
2	Cindy Jules	Stockholm	Sweden	48	117.0	NaN	5504	8949
3	Arthur Kegels	BRUSSELS	BELGIUM	59	121.0	4344.0	8999	300
4	Freya Bismark	Berlin	GERMANYY	53	126.0	7000.0	19000	26000

```
# Example 1a
# ---
# We now rename our columns.
# We can use this method if we have many column names.
# We will use the str.strip(), str.lower(), str.replace() functions
# to ensure that our column names are in lowercase format that we easily
# reference while performing further analysis.
# ---
# str.strip() - This fuction is used to remove leading and trailing characters.
# str.lower() - This function is used to convert all characters to lowercase
# str.replace() - This fuction is used to replace text with some other text.
# ---
#
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_').str.replace('(', '').str.replace(')', '')

# Then preview our resulting dataframe
# ---
df.head()
```

```
<ipython-input-35-4ab1fc01f037>:14: FutureWarning: The default value of regex will
  df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_').str.replac
<ipython-input-35-4ab1fc01f037>:14: FutureWarning: The default value of regex will
  df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_').str.replac
```

	name	city	country	height	weight	account_a	account_b	total_acc
0	Adi Dako	LISBON	PORTUGAL	56	132.0	2390.0	4340	
1	John Paul	LONDON	UNITED KINGDOM	62	165.0	4500.0	34334	3
2	Cindy Jules	Stockholm	Sweden	48	117.0	NaN	5504	
3	Arthur	BRUSSELS	BELGIUM	59	121.0	4344.0	8999	

```
# Example 1b
# ---
# Alternatively, we can rename column names in a dataframe manually by
# specifying new column names that would replace the original column names.
# You should note that this method is cumbersome when
# the no. of features/varibles/columns become large.
# ---
#

# To demonstrate this, we will need to import our dataset again
# so that we work with our original dataset.
# ---
#
df = pd.read_csv('http://bit.ly/DataCleaningDataset', ';')

# We then specify our columns names, store them in a list, then afterwards
# assign this list to the original column names.

# Lets first see our original column names below
# ---
#
df.columns
```

```
<ipython-input-36-dce6b6ba67cd>:14: FutureWarning: In a future version of pandas all arguments of read_csv except for the argument 'file
  df = pd.read_csv('http://bit.ly/DataCleaningDataset', ';')
Index(['NAME', 'CITY', 'COUNTRY', 'HEIGHT', 'WEIGHT', 'ACCOUNT A', 'ACCOUNT B',
       'TOTAL ACCOUNT'],
      dtype='object')
```

```
# We then perform our column values replacement as shown below
# ---
#
df.columns = ['name', 'city', 'country', 'height', 'weight', 'account_a', 'account_b', 'total_account']

# We then preview our dataframe as shown
# ---
#
df.head()
```

	name	city	country	height	weight	account_a	account_b	total_account
0	Adi Dako	LISBON	PORTUGAL	56	132.0	2390.0	4340	6730
1	John Paul	LONDON	UNITED KINGDOM	62	165.0	4500.0	34334	38834
2	Cindy Jules	Stockholm	Sweden	48	117.0	NaN	5504	8949
3	Arthur Kegels	BRUSSELS	BELGIUM	59	121.0	4344.0	8999	300
4	Freya Bismark	Berlin	GERMANYY	53	126.0	7000.0	19000	26000

```
# Example 2
# ---
# We can also perform string conversion to a particular column.
# This allows us to have uniformity across all values of a column.
# In this example, we will convert the values of the column "city" to lower case values.
# ---
# OUR CODE GOES BELOW
#

# Lets convert the city column to comprise of only lowercase characters
# ---
#
df['city'] = df['city'].str.lower()
df.head()
```

	name	city	country	height	weight	account_a	account_b	total_acco
0	Adi Dako	lisbon	PORTUGAL	56	132.0	2390.0	4340	6
1	John Paul	london	UNITED KINGDOM	62	165.0	4500.0	34334	38
2	Cindy Jules	stockholm	Sweden	48	117.0	NaN	5504	8

```
# Example 3
# ---
# We can also perform types of conversion that we would want i.e. metric conversion.
# In this example, we will convert our height values to centimeters noting
# that 1 inch = 2.54 cm.
# ---
# Dataset url = http://bit.ly/DataCleaningDataset
# ---
#

# We can perform our conversion across the column that we would want
# then replace the column with the outcome of our conversion.
# ---
#
df['height'] = df['height'] * 2.54
df.head()
```

	name	city	country	height	weight	account_a	account_b	total_acco
0	Adi Dako	lisbon	PORTUGAL	142.24	132.0	2390.0	4340	6
1	John Paul	london	UNITED KINGDOM	157.48	165.0	4500.0	34334	38
2	Cindy Jules	stockholm	Sweden	121.92	117.0	NaN	5504	8

```
# Example 4
# ---
# We can also perform other types of conversion such as datatype conversion as shown
# in the next cell.
# ---


# But before we do that, let's first determine the column/feature datatypes
# ---
#
df.dtypes
```

```
name              object
city              object
country           object
height           float64
weight           float64
account_a        float64
account_b          int64
total_account      int64
dtype: object
```

```
# Then perform a conversion by converting our column/feature
# through the use of the apply() function, passing the numerical
# type provided by numpy.
# To get an understanding of other datatypes provided by numpy we can visit:
# https://docs.scipy.org/doc/numpy/user/basics.types.html
# ---
# Other
# ---
#
df['height'] = df['height'].apply(np.int64)

# Let's now check whether our conversion happened by checking our updated datatypes
# ---
#
df.dtypes
```

```
name              object
city              object
country           object
height             int64
weight           float64
account_a        float64
account_b          int64
total_account      int64
dtype: object
```

Challenges

```
# Challenge 1
# ---
# Question: Convert the variables account_a and account_b to integer datatype.
# ---
# Hint: You can refer to the df dataframe in the example.
# ---
# OUR CODE GOES BELOW
#
df['account_b'] = df['account_b'].astype(int)
df.dtypes
```

```
name              object
city              object
country           object
height             int64
weight           float64
account_a        float64
account_b          int64
total_account      int64
dtype: object
```

```
# Challenge 2
# ---
# Question: Convert the given weight feature in the dataset from pounds to grams.
# ---
# Hint: 1 pound = 453.592 grams
```

```
#     : You can refer to the df dataframe in the example.
# ---
# ---
# OUR CODE GOES BELOW
#

df['weight']=df['weight']*2.2046

df.head()
```

	name	city	country	height	weight	account_a	account_b	total_acc
0	Adi Dako	lisbon	PORTUGAL	142	291.0072	2390.0	4340	
1	John Paul	london	UNITED KINGDOM	157	363.7590	4500.0	34334	3
2	Cindy Jules	stockholm	Sweden	121	257.9382	NaN	5504	

DATA CLEANING STEP to find syntax errors

Syntax Errors

```
# Example 1
# ---
# While performing our analysis, we can get to a point where we need to
# fix spelling mistakes or typos. This example will show us how we can
# go about this.
# ---
# Dataset url = http://bit.ly/DataCleaningDataset
# ---
# OUR CODE GOES BELOW
#

# Let's replacing any value "GERMANYY" with the correct value "GERMANY".
# We use the string replace() function to perform our operation as shown.
# ---
#
df['country'] = df['country'].str.replace('GERMANYY', 'GERMANY')
df.head()
```

	name	city	country	height	weight	account_a	account_b	total_acc
0	Adi Dako	lisbon	PORTUGAL	142	291.0072	2390.0	4340	
1	John Paul	london	UNITED KINGDOM	157	363.7590	4500.0	34334	3
2	Cindy Jules	stockholm	Sweden	121	257.9382	NaN	5504	

```
# Example 2
# ---
# We can also decide to strip or remove leading spaces (space infront)
# and trailing spaces (spaces at the end) by using the string strip() function
# covered in this example.
# ---
# Dataset = http://bit.ly/DataCleaningDataset
# ---
# OUR CODE GOES BELOW
#

# We first load our dataframe column with the intention to observing leading
# and trailing spaces in the city column
# ---
#
df['city']
```

```
0          lisbon
1           london
2           stockholm
3            brussels
4              berlin
```

```
5         brasilia
6            stockholm
7            london
Name: city, dtype: object
```

```
# Then later we strip the leading and trailing spaces as shown and lastly
# confirm our changes
# ---
#
df['city'] = df['city'].str.strip()
df['city']
```

```
0       lisbon
1       london
2    stockholm
3     brussels
4       berlin
5     brasilia
6    stockholm
7       london
Name: city, dtype: object
```

Challenges

```
# Challenge 1
# ---
# Question: Deal with the leading and trailing whitespaces from Name
# and Team variables in the following dataset.
# ---
# Dataset url = http://bit.ly/NBABasketballDataset
# ---
# OUR CODE GOES BELOW
#
df=pd.read_csv("http://bit.ly/NBABasketballDataset")

df['Name'] = df['Name'].str.strip()

df['Name']

df['Team'] = df['Team'].str.strip()

df['Team']
```

```
0      Boston Celtics
1      Boston Celtics
2      Boston Celtics
3       oston Celtics
4      Boston Celtics
             ...
453         Utah Jazz
454         Utah Jazz
455         Utah Jazz
456         Utah Jazz
457               NaN
Name: Team, Length: 458, dtype: object
```

DATA CLEANING Step to find irrelevant data in the dataset

Irrelevent Data

```
# Example 1
# ---
# We can also deleting/dropping irrelevant columns/features.
# By irrelevant we mean dataset features that we don't need
# to answer a research question.
# ---
# Dataset url = http://bit.ly/DataCleaningDataset
# Hint: Use the df dataframe you created earlier
# ---
# OUR CODE GOES BELOW
#
df = pd.read_csv('http://bit.ly/NBABasketballDataset')

df
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	oston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
...	...	...	...	...	...	...	...	...	...
453	Shelvin Mack	Utah Jazz	8.0	PG	26.0	6-3	203.0	Butler	2433333.0
454	Raul Neto	Utah Jazz	25.0	PG	24.0	6-1	179.0	NaN	900000.0

```
# Deleting an Irrelevant Column i.e. if we didn't require the column city
# to answer our research question.
# ---
# While dropping/deleting those two columns:
# a) We set axis = 1
#    A dataframe has two axes: “axis 0” and “axis 1”.
#    “axis 0” represents rows and “axis 1” represents columns.
# b) We can also set Inplace = True.
#    This means the changes would be made to the original dataframe.
# Dropping the irrelevant columns i.e. Team and Weight
# Those values were dropped since axis was set equal to 1 and
# the changes were made in the original data frame since inplace was True.
#
df.drop(["Team"] , axis = 1 , inplace = True)
df.head()
```

	Name	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	30.0	SG	27.0	6-5	205.0	Boston University	NaN
							Georgia	

```
# We can drop multiple columns as shown
# ---
#
df.drop(["Position" , "College"], axis = 1, inplace = True)

# And preview our resulting dataset
# ---
#
df.head()
```

	Name	Number	Age	Height	Weight	Salary
0	Avery Bradley	0.0	25.0	6-2	180.0	7730337.0
1	Jae Crowder	99.0	25.0	6-6	235.0	6796117.0
2	John Holland	30.0	27.0	6-5	205.0	NaN
3	R.J. Hunter	28.0	22.0	6-5	185.0	1148640.0
4	Jonas Jerebko	8.0	29.0	6-10	231.0	5000000.0

```
# Example 2
# ---
# We can also fix in-record & cross-datasets errors.
# These kinds errors result from having two or more values in the same row
# or across datasets contradicting with each other.
# ---
```

```
# Dataset = http://bit.ly/DataCleaningDataset
# ---
# OUR CODE GOES BELOW
#
df = pd.read_csv('http://bit.ly/NBABasketballDataset')
df['total_account'] = df['Number'] + df['Weight']

# Previewing our resulting dataframe
# ---
#
df.head(10)
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary	total_account
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0	180.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0	334.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN	235.0
3	R.J. Hunter	oston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0	213.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0	239.0
5	Amir Johnson	Boston Celtics	90.0	PF	29.0	6-9	240.0	NaN	12000000.0	330.0
6	Jordan Mickey	oston Celtics	55.0	PF	21.0	6-8	235.0	LSU	1170960.0	290.0
7	Kelly Olynyk	Boston Celtics	41.0	C	25.0	7-0	238.0	Gonzaga	2165160.0	279.0
8	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0	202.0
9	Marcus Smart	Boston Celtics	36.0	PG	22.0	6-4	220.0	Oklahoma State	3431040.0	256.0

```
# Create another column to tell us whether if the two columns match.
# We will use the numpy library through use of np.
# ---
#
df['total_account?'] = np.where(df['total_account'] == df['Weight'], 'True', 'False')

# Previewing our resulting dataframe
# ---
#
df.head()
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary	tot
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0	
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0	
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN	

```
# Let's now select the records which don't match
# ---
#
df.loc[df['total_account?'] == "False"]
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0

```
# At this point we can do several things
# 1. Correct the values,
# 2. Drop/Delete the values,
# 3. Or even decide to leave them as they are for certain reasons
# ---
# If we had a large dataset, we could get the no. of records using len(),
# this would help us in our decision making process.
# ---
#
len(df.loc[df['total_account?'] == "False"])
```

```
438
```

453	[illegible]	[illegible]	8.0	PG	26.0	6-3	203.0	Butler	2433333.0
454	Neto	Jazz	25.0	PG	24.0	6-4	179.0	NaN	900000.0

Challenges

```
# Challenge 1
# ---
# Question: While perfoming some analysis to answer a research question,
# we realize that we don't need the Date and Time features in our dataset.
# Let's drop those two features below
# ---
# Dataset url = https://bit.ly/SuperMarketSalesDB
# ---
# OUR CODE GOES BELOW
#
df = pd.read_csv('https://bit.ly/SuperMarketSalesDB')
df.drop(["Date", "Time"] , axis = 1 , inplace = True)
df.head()
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.141
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.820
	631-41-					Home and			

Data Cleaning to find duplicates in the dataset

Duplicates

```
# Example 1
# ---
# Finding duplicate records
# -> Duplicate records are repeated records in a dataset.
# ---
# Dataset url = http://bit.ly/NBABasketballDataset
# ---
# OUR CODE GOES BELOW
#

nba_df = pd.read_csv('http://bit.ly/NBABasketballDataset')

# Again, we first explore our dataset by determining the shape of
# our dataset (records/instances, columns/variables)
# ---
#
nba_df.head()
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
2	Holland	Celtics	30.0	SG	27.0	6-5	205.0	University	NaN

```
nba_df.shape
```

```
(458, 9)
```

```
# We can then identify which observations are duplicates
# through the duplicated() function and sum() to know how many
# duplicate records there are.
# Normally, duplicate records are dropped from the dataset.
# But in our case we don't have any duplicate records.
# ---
#
nba_df = nba_df[nba_df.duplicated()]


# Finding the no. of duplicates
# ---
#
sum(nba_df.duplicated())
```

```
0
```

```
# Example 2
# ---
# Dropping duplicate columns
# ---
# Dataset = http://bit.ly/NBABasketballDataset
# ---
# OUR CODE GOES BELOW
#
nba_df = pd.read_csv('http://bit.ly/NBABasketballDataset')
# In our previous dataset, if there were duplicates we
# could have dropped the through the use of the drop_duplicates() function
# as shown in this example
# ---
#
nba_df_duplicates = nba_df.drop_duplicates()


# Example 3
# ---
# Dropping duplicates in a specific column in thed df dataframe
# ---
# Dataset url = http://bit.ly/NBABasketballDataset
# ---
#

duplicates_df = pd.read_csv("http://bit.ly/NBABasketballDataset")


# We can also consider records with repeated variables/columns
# as duplicates and deal with them. For example, we can
# identify duplicates in our dataset based on city.
# ---
#
duplicates_df = df[df.duplicated(['City'])]

duplicates_df
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2

```
# Then dropping the duplicates as shown below.
# NB: We will create in a new dataframe object which will contain our unique dataframe
# which won't have any duplicates.
# ---
#
unique_df = df.drop_duplicates(['City'])

# Determining the size of our new dataset
# We note that the two records were dropped from our original dataset
# ---
#
unique_df.shape
```

```
(3, 15)
```

Challenges

```
# Challenge 1
# ---
# Question: Find the duplicates in the following dataset.
# ---
# Dataset url = https://bit.ly/ShoprityDS
# ---
# OUR CODE GOES BELOW
#

df = pd.read_csv('https://bit.ly/ShoprityDS')

duplicate=df[df.duplicated(keep='last')]

sum(df.duplicated())
```

```
6
```

```
duplicate
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type
6	FDO10	13.650	Regular	0.012741	Snack Foods
20	FDU02	13.350	Low Fat	0.102492	Dairy
43	FDK43	9.800	Low Fat	0.026818	Meat
110	FDG12	6.635	Regular	0.000000	Baking Goods
					Frozen

```
df.shape
```

```
(8529, 11)
```

```
df_duplicates = df.drop_duplicates(inplace=True)
```

```
df.shape
```

```
(8523, 11)
```

```
# Challenge 2
#
```

```
# ---
# Question: From your understanding of the features, deal with the duplicates found in the given dataset.
# ---
# Dataset url = https://bit.ly/ShoprityDS
# ---
# OUR CODE GOES BELOW
#
nba_df =pd.read_csv('https://bit.ly/ShoprityDS')
```

```
nba_df_duplicates = nba_df.drop_duplicates()
```

Data Cleaning to find the missing data

Missing Data

```
# Example 3a
# ---
# Checking for missing data
# NB: This method may not be the most convenient. Why?
# ---
# We can check if there is any missing values in the entire dataframe as shown
# ---
# OUR CODE GOES BELOW
#
df.isnull()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
8524	False	False	False	False	False
8525	False	False	False	False	False
8526	False	False	False	False	False
8527	False	False	False	False	False
8528	False	False	False	False	False

8523 rows × 11 columns

```
# Example 3b
# ---
# We can also check for missing values in each column
# ---
# OUR CODE GOES BELOW
#
df.isnull().any()
```

```
Invoice ID                False
Branch                    False
City                      False
Customer type             False
Gender                    False
Product line              False
Unit price                False
Quantity                  False
Tax 5%                    False
Total                     False
Payment                   False
cogs                      False
gross margin percentage   False
gross income              False
Rating                    False
dtype: bool
```

```
# Example 3c
# ---
# We can check how many missing values there are across each column by
```

```
# ---
# OUR CODE GOES BELOW
#
df.isnull().sum()
```

```
Invoice ID                0
Branch                    0
City                      0
Customer type             0
Gender                    0
Product line              0
Unit price                0
Quantity                  0
Tax 5%                    0
Total                     0
Payment                   0
cogs                      0
gross margin percentage   0
gross income              0
Rating                    0
dtype: int64
```

```
# Example 3d
# ---
# We can also check to see if we have any missing values in the dataframe
# ---
# OUR CODE GOES BELOW
#
print(df.isnull().values.any())
```

```
False
```

```
# Example 3e: Method 1
# ---
# Dealing with the missing data
# ---
# OUR CODE GOES BELOW
#

# We can drop the missing observations
# ---
#
df_no_missing = df.dropna()

# Checking for missing data
df_no_missing.isnull().sum()
```

```
Invoice ID                0
Branch                    0
City                      0
Customer type             0
Gender                    0
Product line              0
Unit price                0
Quantity                  0
Tax 5%                    0
Total                     0
Payment                   0
cogs                      0
gross margin percentage   0
gross income              0
Rating                    0
dtype: int64
```

```
# Example 3e: Method 2
# ---
# We can drop rows where all cells in that row is NA
# ---
# OUR CODE GOES BELOW
#
df_cleaned = df.dropna(how='all')

# Checking the shape of our dataset
df_cleaned.shape
```

```
(1000, 15)
```

```
# Example 3e: Method 3
# ---
# We could drop columns if they only contain missing values
# ---
# OUR CODE GOES BELOW
#
df_without_columns = df.dropna(axis=1, how='all')

# Checking the shape of our dataset
df_without_columns.shape
```

```
(1000, 15)
```

```
# Example 3e: Method 4
# ---
# We could drop rows that contain less than five observations
# ---
# OUR CODE GOES BELOW
#
df.dropna(thresh=5)

# Checking the shape of our dataset
df.shape


# Further reading
# ----
# Above are only a few methods of dealing with missing data.
```

```
(1000, 15)
```

Challenges

```
# Challenge 3a
# ---
# Question: Find the missing values in the following datset.
# You can use any of the above methods to you see fit.
# ---
# Dataset url = https://bit.ly/ShoprityDS
# ---
# OUR CODE GOES BELOW
#

df= pd.read_csv('https://bit.ly/ShoprityDS')

df.isnull()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
8524	False	False	False	False	False
8525	False	False	False	False	False
8526	False	False	False	False	False
8527	False	False	False	False	False
8528	False	False	False	False	False

8529 rows × 11 columns

```
# Challenge 3b
# ---
# Question: Handle the missing values in the following dataset.
# You can any of the above methods that you see fit.
# ---
```

```
# Dataset url = https://bit.ly/ShoprityDS
# ---
# OUR CODE GOES BELOW
#

df.dropna(thresh=20)

df.shape
```

```
(8529, 11)
```

Filtering

Examples and Challenges

```
# Example 4a
# ---
# Selecting rows when columns contain certain values
# ---
# NB: Selecting records where petal_length is 5.0
# ---
# OUR CODE GOES BELOW
#

# Reading our csv file from the given url and storing it to a dataframe.
iris_df = pd.read_csv("http://bit.ly/IrisDataset")

# Previewing the first 5 records
iris_df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```
# Selecting the records where column has certain values
# ---
#
iris_df[iris_df.petal_length.isin(['5.0'])]
```

	sepal_length	sepal_width	petal_length	petal_width	species

```
# Challenge 4a
# ---
# Question: From the given dataset, find observations with outlets
# established in 2002.
# ---
# Dataset url = https://bit.ly/ShoprityDS
# ---
# OUR CODE GOES BELOW
#
df_shop = pd.read_csv("https://bit.ly/ShoprityDS")
df_shop[df_shop.Outlet_Establishment_Year.isin(['2002'])]
#df_shop.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item

```
# Example 4b
# ---
# Selecting the records where column doesn't have certain values
# in our case, where petal_leghth is not 5.0
# ---
# OUR CODE GOES BELOW
#

iris_df[~iris_df.petal_length.isin(['5.0'])]
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

```
# Challenge 4b
# ---
# Question: Select all the Dairy observations from the given dataset.
# ---
# Dataset url = https://bit.ly/ShoprityDS
# ---
# OUR CODE GOES BELOW
#
df_shop[df_shop.Item_Type.isin(['Dairy'])]
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type
0	FDA15	9.300	Low Fat	0.016047	Dairy
12	FDA03	18.500	Regular	0.045464	Dairy
20	FDU02	13.350	Low Fat	0.102492	Dairy
21	FDU02	13.350	Low Fat	0.102492	Dairy
30	FDE51	5.925	Regular	0.161467	Dairy
...	...	...	...	...	...
8430	FDC39	7.405	Low Fat	0.159165	Dairy
8453	FDS26	20.350	Low Fat	0.089975	Dairy
8454	FDV50	14.300	Low Fat	0.123071	Dairy
8463	FDY50	5.800	Low Fat	0.130931	Dairy
8518	FDR26	20.700	Low Fat	0.042801	Dairy

683 rows × 11 columns

```
# Example 4c
# ---
# Selecting records using filters where petal_width greater than 1.9
# ---
# OUR CODE GOES BELOW
#

iris_df[(iris_df['petal_width'] > 1.9)]
```

	sepal_length	sepal_width	petal_length	petal_width	species
100	6.3	3.3	6.0	2.5	Iris-virginica
102	7.1	3.0	5.9	2.1	Iris-virginica
104	6.5	3.0	5.8	2.2	Iris-virginica
105	7.6	3.0	6.6	2.1	Iris-virginica
109	7.2	3.6	6.1	2.5	Iris-virginica
110	6.5	3.2	5.1	2.0	Iris-virginica
112	6.8	3.0	5.5	2.1	Iris-virginica
113	5.7	2.5	5.0	2.0	Iris-virginica
114	5.8	2.8	5.1	2.4	Iris-virginica
115	6.4	3.2	5.3	2.3	Iris-virginica
117	7.7	3.8	6.7	2.2	Iris-virginica
118	7.7	2.6	6.9	2.3	Iris-virginica
120	6.9	3.2	5.7	2.3	Iris-virginica
121	5.6	2.8	4.9	2.0	Iris-virginica
122	7.7	2.8	6.7	2.0	Iris-virginica
124	6.7	3.3	5.7	2.1	Iris-virginica
128	6.4	2.8	5.6	2.1	Iris-virginica
131	7.9	3.8	6.4	2.0	Iris-virginica
132	6.4	2.8	5.6	2.2	Iris-virginica
135	7.7	3.0	6.1	2.3	Iris-virginica
136	6.3	3.4	5.6	2.4	Iris-virginica
139	6.9	3.1	5.4	2.1	Iris-virginica

```
# Challenge 4c
# ---
# Question: Which observations had items outlet sales greater than 2000?
# ---
# Dataset url = https://bit.ly/ShoprityDS
# ---
# OUR CODE GOES BELOW
#
df_shop[(df_shop['Outlet_Establishment_Year'] > 2000)]
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type
1	DRC01	5.920	Regular	0.019278	Soft Drinks
5	FDP36	10.395	Regular	0.000000	Baking Goods
9	FDH17	16.200	Regular	0.016687	Frozen Foods
10	FDU28	19.200	Regular	0.094450	Frozen Foods
17	NCB42	11.800	Low Fat	0.008596	Health and Hygiene
...	...	...	...	...	...
8521	FDH24	20.700	Low Fat	0.021518	Baking Goods
8522	NCJ19	18.600	Low Fat	0.118661	Others
8525	FDS36	8.380	Regular	0.046982	Baking

```
# Example 5d
# ---
# We can also use the query method to get for data where petal_width is equal to 1.0
# Once you run this cell, replace the equals operator == to with less than <
# or greater than > operators to see their applications as well.
```

```
# ---
# The parameter inplace makes changes in the original dataframe if True.
# We should also note the query method works if the column name doesn’t have any empty spaces.
# Hence the need replace blank spaces in our column names with '_'
# ---
#

# Let's filter our data
#
iris_df.query('petal_width == 1.9', inplace = True)

# Previewing our dataset
iris_df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
101	5.8	2.7	5.1	1.9	Iris-virginica
111	6.4	2.7	5.3	1.9	Iris-virginica
130	7.4	2.8	6.1	1.9	Iris-virginica
142	5.8	2.7	5.1	1.9	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica

```
# We can also perform multiple condition filtering as shown below
# ---
#

iris_df.query('sepal_width == 2.7 and petal_length == 5.1', inplace = True)

# Previewing our dataset
iris_df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
101	5.8	2.7	5.1	1.9	Iris-virginica
142	5.8	2.7	5.1	1.9	Iris-virginica

```
# Challenge 5d
# ---
# Question: Which observations had items outlet sales greater than 2000 and less than 3000?
# ---
# Dataset url = https://bit.ly/ShoprityDS
# ---
# OUR CODE GOES BELOW
#
df_shop.query('Item_Outlet_Sales > 2000 and Item_Outlet_Sales < 3000', inplace=True)

df_shop.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	It
2	FDN15	17.50	Low Fat	0.016760	Meat	1
12	FDA03	18.50	Regular	0.045464	Dairy	1
14	FDS46	17.60	Regular	0.047257	Snack Foods	1
19	DRI11	NaN	Low Fat	0.034238	Hard Drinks	1

Sorting

```
# Example 5a
# ---
# Load the given dataframe in ascending order by reports
# ---
# OUR CODE GOES BELOW
#

# Sorting our dataset in ascending order by sepal_length
# ---
```

```
#
iris_df.sort_values(by='sepal_length', ascending=1)
```

	sepal_length	sepal_width	petal_length	petal_width	species
101	5.8	2.7	5.1	1.9	Iris-virginica
142	5.8	2.7	5.1	1.9	Iris-virginica

```
# Challenge 5a
# ---
# Sort items by weight in descending order given the following dataset.
# Hint: ascending = 0
# ---
# Dataset url = https://bit.ly/ShoprityDS
# ---
# OUR CODE GOES BELOW
#
df_shop.sort_values(by='Item_Weight', ascending=0)
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year
5117	NCO42	21.25	LF	0.024651	Household	146.0102	OUT035	2004
5788	FDA45	21.25	Low Fat	0.155250	Snack Foods	175.7370	OUT013	1987
4770	FDT03	21.25	Low Fat	0.010055	Meat	183.1608	OUT017	2007
6711	FDA45	21.25	Low Fat	0.156013	Snack Foods	177.3370	OUT018	2009
6107	FDA45	21.25	Low Fat	0.155695	Snack Foods	177.6370	OUT045	2002
...	...	...	...	...	...	...	...	...
8277	FDX46	NaN	Regular	0.057835	Snack Foods	57.5562	OUT027	1985
8355	FDN15	NaN	Low Fat	0.016653	Meat	139.5180	OUT027	1985
8368	FDY37	NaN	Regular	0.026440	Canned	143.6470	OUT027	1985
					Fruits and			

Splitting, Merging and Concatenation

Examples

```
# Example 6a
# ---
# Split the dataframe species column into two columns by using the "-" character
# ---
# OUR CODE GOES BELOW
#

# Dropping null value columns to avoid errors
iris_df.dropna(inplace = True)

# New data frame with split value columns
new_iris_df = iris_df["species"].str.split("-", n = 1, expand = True)

# Naking separate first name column from new data frame
iris_df["family"]= new_iris_df[0]

# Making separate last name column from new data frame
iris_df["sub-species"]= new_iris_df[1]

# Dropping old Name columns
iris_df.drop(columns =["species"], inplace = True)

# Displaying our dataframe
iris_df
```

	sepal_length	sepal_width	petal_length	petal_width	family	sub-species

```
# Example 2
# ---
# Concatenating (merging) two columns in a dataframe
# ---
# OUR CODE GOES BELOW
#

# Concatenating our column
#
iris_df['species'] = iris_df['family'].str.cat(iris_df['sub-species'],sep="-")

# Dropping old name columns
#
iris_df.drop(columns = ["family", "sub-species"], inplace = True)

# Previewing our new dataframe
#
iris_df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
101	5.8	2.7	5.1	1.9	Iris-virginica
142	5.8	2.7	5.1	1.9	Iris-virginica

```
# Challenge 6a
# ---
# Concatenate the the city and state columns from the dataset below.
# The resulting region column should have city and state seperated by a comma and whitespace.
# ---
# Dataset url = http://bit.ly/SchoolShootingsDataset
# ---
# Step 1: Load the dataset
# Step 2: Preview the dataset
# Step 3: Perform your concatenation
# Step 4: Dropping old columns
# Step 4: Preview your final dataframe
# ---
# OUR CODE GOES BELOW
#
df = pd.read_csv('http://bit.ly/SchoolShootingsDataset',  encoding = "latin-1")


# perform concatenation
df['region'] = df['city'].str.cat(df['state'], sep=", ")

# Dropping old name columns
#
df.drop(columns = ["city", "state"], inplace = True)

# Previewing our new dataframe
#
df.head()
```

	uid	nces_school_id	school_name	nces_district_id	district_name	date	school_y
0	1	080480000707	Columbine High School	804800.0	Jefferson County R-1	4/20/1999	1998-1
1	2	220054000422	Scotlandville Middle School	2200540.0	East Baton Rouge Parish School Board	4/22/1999	1998-1
2	3	130441001591	Heritage High School	1304410.0	Rockdale County	5/20/1999	1998-1
3	4	421899003847	John Bartram High School	4218990.0	Philadelphia City SD	10/4/1999	1999-2
4	5	250279000225	Dorchester High School	2502790.0	Boston	11/3/1999	1999-2

5 rows × 49 columns

```
# Example 6b
# ---
# Merging two dataframes.
# In this example, we will create two simple dataframes for demonstration purposes.
# ---
# OUR CODE GOES BELOW
#

# Let's create our first dataframe and preview it
#
df1 = pd.DataFrame([[2, 3], [41, 51]], columns=['a', 'b'])
df1.head()
```

	a	b
0	2	3
1	41	51

```
 # Let our second dataframe and preview it
#
df2 = pd.DataFrame([[2, 5], [41, 6]], columns=['a', 'c'])
df2.head()
```

	a	c
0	2	5
1	41	6

```
# Merging our dataframes and storing our resulting dataframe in df3, then previewing it
#
df3 = df1.merge(df2, how='left', on='a')
df3.head()
```

	a	b	c
0	2	3	5
1	41	51	6

```
# Challenge 6b
# ---
# Merge the following two datasets
# ---
# Dataset 1 url = http://bit.ly/CitiesDataset
# Dataset 2 url = http://bit.ly/CountriesDataset1
# ---
# OUR CODE GOES BELOW
#
df_city = pd.read_csv("http://bit.ly/CitiesDataset")
df_city.head()

df_country = pd.read_csv("http://bit.ly/CountriesDataset1")
df_country.head()

df1 = pd.DataFrame([['Albania',2.90,'no','yes'], ['Andorra',0.07,'no','no']], columns=['country', 'population', 'EU', "coastline"])
df1.head()

df2 = pd.DataFrame([['Albacete','spain',39,-1.87,12.62], ['Arad','Romania',46.17,21.32,9.32]], columns=['city','country', 'latitude', 'logitu
df2.head()

df3= df1.merge(df2, how='right')
df3.head()
```

	country	population	EU	coastline	city	latitude	logitude	temperature
0	spain	NaN	NaN	NaN	Albacete	39.00	-1.87	12.62
1	Romania	NaN	NaN	NaN	Arad	46.17	21.32	9.32

Outliers

```
# Example 1
# ---
# Given the following dataset, find and deal with outliers.
# ---
# Dataset url = http://bit.ly/CountryDataset1
# ---
# OUR CODE GOES BELOW
#

# Let's read data from url as dataframe
#
outliersc_df = pd.read_csv("http://bit.ly/CountryDataset1")

# Lets preview our our dataframe below
#
outliersc_df.head()
```

	country	year	pop	continent	lifeExp	gdpPercap
0	Afghanistan	1952	8425333.0	Asia	28.801	779.445314
1	Afghanistan	1957	9240934.0	Asia	30.332	820.853030
2	Afghanistan	1962	10267083.0	Asia	31.997	853.100710
3	Afghanistan	1967	11537966.0	Asia	34.020	836.197138
4	Afghanistan	1972	13079460.0	Asia	36.088	739.981106

```
# Checking the size of our dataset for cleaning purposes
# ---
#
outliersc_df.shape
```

```
(1704, 6)
```

```
# There are many ways of dealing with the outliers however in this session we wiil
# use the interquartile range (IQR).
# A data point is considered to be an outiler if it is more
# than 1.5 * IQR above the third quartile (i.e. Q3  + 1.5 * IQR)
# or below the first quartile (i.e. Q1 - 1.5 * IQR).
# If we were to use a box plot visualisation then,
# we would be able to visually see those values outside this range.
# Something to note is that this method will consider only the numerical values in
# our dataset. Lets now calculate the IQR for each column.
# ---
#

# We first defining our quantiles using the quantile() function
# ---
#
Q1 = outliersc_df.quantile(0.25)
Q3 = outliersc_df.quantile(0.75)
IQR = Q3 - Q1
IQR

# Then filtering out our outliers by getting values which are outside our IQR Range.
# ---
#
outliers_df_iqr = outliersc_df[((outliersc_df < (Q1 - 1.5 * IQR)) | (outliersc_df > (Q3 + 1.5 * IQR))).any(axis=1)]

# Checking the size of the dataset with outliers for cleaning purposes
# ---
#
outliers_df_iqr.shape
```

```
<ipython-input-181-cf3dc7b4fc40>:16: FutureWarning: The default value of numeric_only in DataFrame.quantile is deprecated. In a future v
  Q1 = outliersc_df.quantile(0.25)
<ipython-input-181-cf3dc7b4fc40>:17: FutureWarning: The default value of numeric_only in DataFrame.quantile is deprecated. In a future v
  Q3 = outliersc_df.quantile(0.75)
<ipython-input-181-cf3dc7b4fc40>:24: FutureWarning: Automatic reindexing on DataFrame vs Series comparisons is deprecated and will raise
  outliers_df_iqr = outliersc_df[((outliersc_df < (Q1 - 1.5 * IQR)) | (outliersc_df > (Q3 + 1.5 * IQR))).any(axis=1)]
(317, 6)
```

```
# We can also explore our outliers by doing the following
# ---
```

```
#
outliers_df_iqr
```

	country	year	pop	continent	lifeExp	gdpPercap
67	Australia	1987	16257249.0	Oceania	76.320	21888.889030
68	Australia	1992	17481977.0	Oceania	77.560	23424.766830
69	Australia	1997	18565243.0	Oceania	78.830	26997.936570
70	Australia	2002	19546792.0	Oceania	80.370	30687.754730
71	Australia	2007	20434176.0	Oceania	81.235	34435.367440
...	...	...	...	...	...	...
1651	Vietnam	1987	62826491.0	Asia	62.820	820.799445
1652	Vietnam	1992	69940728.0	Asia	67.662	989.023149
1653	Vietnam	1997	76048996.0	Asia	70.672	1385.896769
1654	Vietnam	2002	80908147.0	Asia	73.017	1764.456677
1655	Vietnam	2007	85262356.0	Asia	74.249	2441.576404

317 rows × 6 columns

```
# Lastly, the most common method of handling our outliers is to drop them.
# In some cases we can:
# 1. Leave them if they are genuine
# 2. Replace them with values within the IQR range
# 3. Drop them
# ---
# In our case, we will drop them.
# We just use the "~" character to refer to the other part of the dataset that
# does not have outliers
# ---
#
clean_dfc_iqr = outliersc_df[~((outliersc_df < (Q1 - 1.5 * IQR)) | (outliersc_df > (Q3 + 1.5 * IQR))).any(axis=1)]

# Checking the size of our final dataset.
clean_dfc_iqr.shape
```

```
<ipython-input-183-88230a30cb17>:12: FutureWarning: Automatic reindexing on DataFrame vs Series comparisons is deprecated and will raise
  clean_dfc_iqr = outliersc_df[~((outliersc_df < (Q1 - 1.5 * IQR)) | (outliersc_df > (Q3 + 1.5 * IQR))).any(axis=1)]
(1387, 6)
```

Challenges

```
# Challenge 1
# ---
# Question: Find and Deal with the outliers in the given df dataframe.
# ---
# Dataset url = http://bit.ly/DataCleaningDataset
# You can use the df dataframe you created ealier on.
# ---
# OUR CODE GOES BELOW
#

df_clean = pd.read_csv("http://bit.ly/DataCleaningDataset")

df_clean.head()

#df_clean.shape
```

	NAME;CITY;COUNTRY;HEIGHT;WEIGHT;ACCOUNT A;ACCOUNT B;TOTAL ACCOUNT
0	Adi Dako ;LISBON ;PORTUGAL ;56;132;2390...
1	John Paul;LONDON ;UNITED KINGDOM;62;165;4500...
2	Cindy Jules;Stockholm;Sweden;48;117;;5504;8949
3	Arthur Kegels;BRUSSELS;BELGIUM;59;121;4344...
4	Freya Bismark; Berlin;GERMANYY;53;126;7000;19...

Exporting Data

```
# Pre-requisite 7
# ---
# Importing files module
# ---
# OUR CODE GOES BELOW
#

# Files module will allow us to download our file from colaboratory
from google.colab import files


# Example 7a
# ---
# Load the given dataframe in ascending order by reports
# ---
# OUR CODE GOES BELOW
#

# Create our sample dataframe
data = {'name': ['Daniel', 'Joyce', 'Elizabeth', 'Sanni', 'Sefu'],
        'year': [2012, 2012, 2013, 2014, 2014],
        'reports': [4, 24, 31, 2, 3]}
df = pd.DataFrame(data, index = ['Nairobi', 'Cairo', 'Cape Town', 'Adis Ababa', 'Mombasa'])
df.head()
```

	name	year	reports
Nairobi	Daniel	2012	4
Cairo	Joyce	2012	24
Cape Town	Elizabeth	2013	31
Adis Ababa	Sanni	2014	2
Mombasa	Sefu	2014	3

```
# Exporting our sample dataframe as a csv
# ---
#

df.to_csv('example.csv')

# NB: In order to download our files from colaboratory we need to run the following
#
files.download('example.csv')



# Challenge 7a
# ---
# Question: Export a csv file of your resulting shoprity dataframe below
# ---
# OUR CODE GOES BELOW
#
df = pd.read_csv('example.csv')

df
```

	Unnamed: 0	name	year	reports
0	Nairobi	Daniel	2012	4
1	Cairo	Joyce	2012	24
2	Cape Town	Elizabeth	2013	31
3	Adis Ababa	Sanni	2014	2
4	Mombasa	Sefu	2014	3

```
# Example 7b
# ---
# Exporting our sample dataframe as a excel file (xlsx)
```

```
# ---
# ---
# OUR CODE GOES BELOW
#

df.to_excel('example.xlsx')

# NB: In order to download our files from colab we need to run the following
#
files.download('example.xlsx')



# Challenge 7b
# ---
# Question: Export a excel file of your resulting shoprity dataframe
# ---
# OUR CODE GOES BELOW
#
df = pd.read_excel('example.xlsx')
df
```

	Unnamed: 0.1	Unnamed: 0	name	year	reports
0	0	Nairobi	Daniel	2012	4
1	1	Cairo	Joyce	2012	24
2	2	Cape Town	Elizabeth	2013	31
3	3	Adis Ababa	Sanni	2014	2
4	4	Mombasa	Sefu	2014	3

✓ 0s completed at 11:15 AM