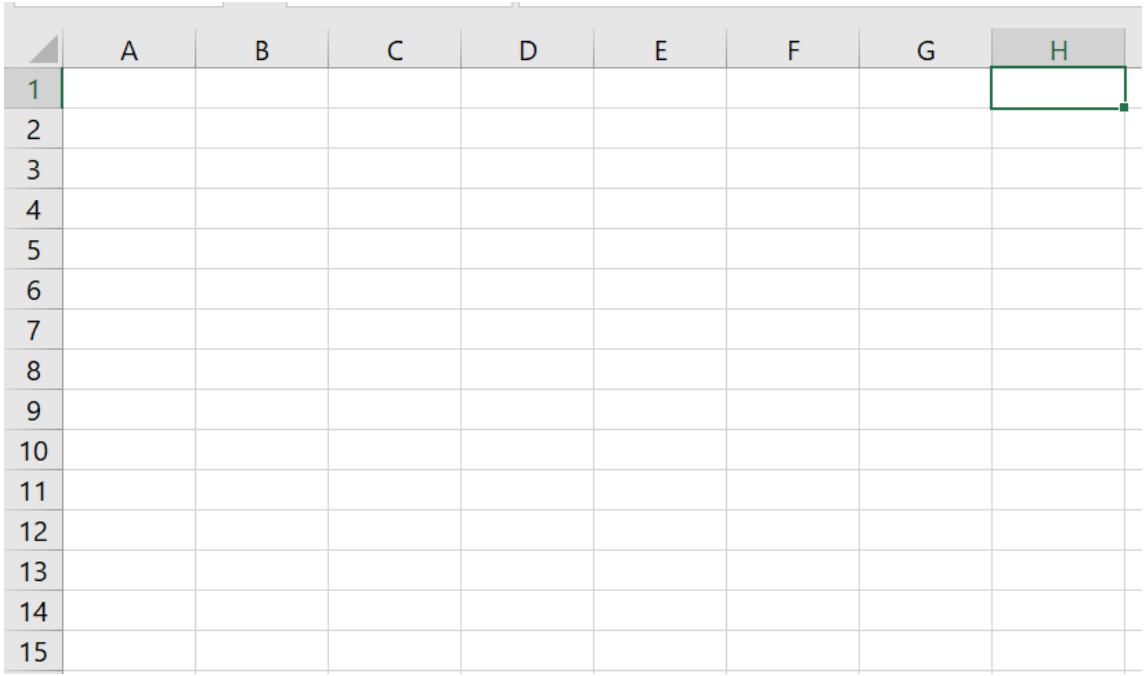
Python Pandas DataFrame

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types

Data Science: Python Pandas DataFrame

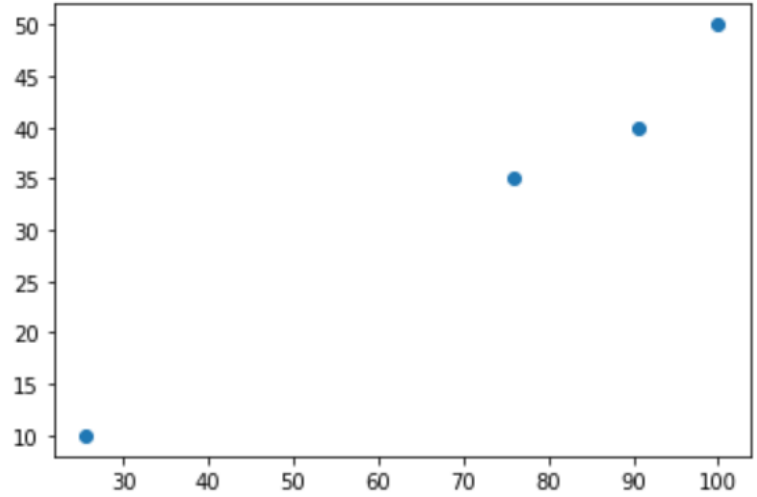


Python Pandas DataFrame

```
import pandas as pd
from matplotlib import pyplot as plt

grades = [90.5, 100.0, 75.8, 25.6]
studytime = [40, 50, 35, 10]
# Convert the List of Grades into Excel Spreadsheet Lookalike Column Format
# Use Pandas DataFrame to convert ONE list to start with (One Column)
df = pd.DataFrame(grades, columns = ["Grades"])
# Now we can add another List as second column to the dataframe created
df["Studytime"]=studytime
# Take a look at the dataframe. Doesn't it look just like Excel?
print (df)
# Use Pandas DataFrame Correlation to perform Pearson R
print(df.corr())
plt.scatter(grades,studytime)
plt.show()
```

Data Science: Python Pandas DataFrame and Correlation

```
   Grades  Studytime
0    90.5         40
1   100.0         50
2    75.8         35
3    25.6         10
            Grades  Studytime
Grades     1.00000    0.99219
Studytime  0.99219    1.00000
```



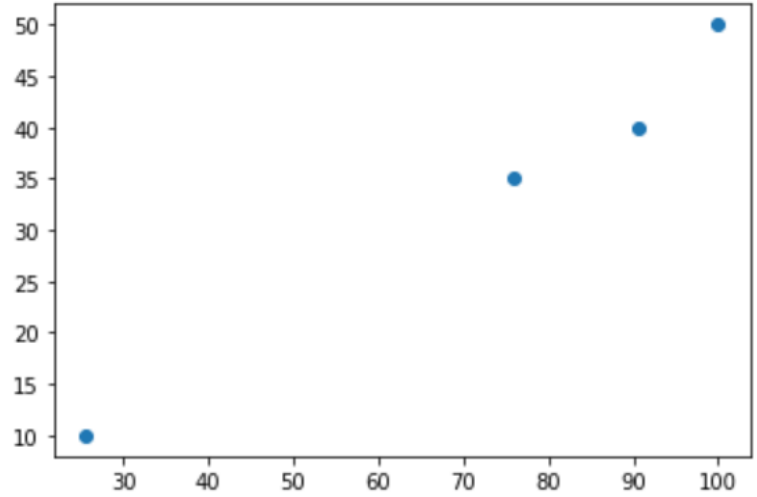
Data Science: Python Pandas DataFrame and Correlation Demo

Python Pandas DataFrame

```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline

grades = [90.5, 100.0, 75.8, 25.6]
studytime = [40, 50, 35, 10]

# Convert the List of Grades into Excel Spreadsheet Lookalike Column Format
data = list(zip(grades, studytime))

df = pd.DataFrame(data, columns = ["Grades", "StudyTime"])

print (df)

# Use Pandas DataFrame Correlation
print (df.corr())
plt.scatter(studytime, grades)
plt.show()
```

Data Science: Python Pandas DataFrame and Correlation

```
   Grades  Studytime
0    90.5         40
1   100.0         50
2    75.8         35
3    25.6         10
            Grades  Studytime
Grades     1.00000    0.99219
Studytime  0.99219    1.00000
```



Python Pandas DataFrame

```
import pandas as pd

MyClass = {'students':['Bruce', 'Jane', 'Nancy',
'Bill'],
        'grades':[10, 9, 9, 8]}

df = pd.DataFrame(MyClass)
```

	students	grades
0	Bruce	10
1	Jane	9
2	Nancy	9
3	Bill	8

Python Pandas DataFrame

```
import pandas as pd

MyClass = {'students':['Bruce', 'Jane', 'Nancy',
'Bill'],
        'grades':[10, 9, 9, 8]}

df = pd.DataFrame(MyClass, index
=["ID1","ID2","ID3","ID4"])
```

	students	grades
ID1	Bruce	10
ID2	Jane	9
ID3	Nancy	9
ID4	Bill	8

Python Pandas DataFrame

```
import pandas as pd

MyClass =
{'John':10,'Jake':9,'Jackie':8,'Jack':7,'Jane':6,'Jo'
:10,'Ja':9,'Jac':8,'Jacky':7,'Jan':6}

df = pd.DataFrame(MyClass, index=[1, 2, 3])
```

	John	Jake	Jackie	Jack	Jane	Jo	Ja	Jac	Jacky	Jan
1	10	9	8	7	6	10	9	8	7	6
2	10	9	8	7	6	10	9	8	7	6
3	10	9	8	7	6	10	9	8	7	6

Python Pandas DataFrame

```
MyInventory = {
  "Item": ["coffee", "chocolate", "tea",
"water"],
  "Promotion": [False, False, True, False],
  "Price": [5.95, 5.95, 3.95, 2.95],
  "Stock": [100, 250, 1000, 1200]
}

ddf = pd.DataFrame(MyInventory)
ddf
```

	Item	Promotion	Price	Stock
0	coffee	False	5.95	100
1	chocolate	False	5.95	250
2	tea	True	3.95	1000
3	water	False	2.95	1200

Python Pandas DataFrame

```
Inv2 = {
  "Item": ["coffee", "chocolate", "tea",
"water"],
  "Promotion": ["no","no","yes","yes"],
  "Price": [5.95, 5.95, 3.95, 2.95],
  "Stock": [100, 250, 1000, 1200]
}

InvDF = pd.DataFrame(Inv2)
InvDF
```

	Item	Promotion	Price	Stock
0	coffee	no	5.95	100
1	chocolate	no	5.95	250
2	tea	yes	3.95	1000
3	water	yes	2.95	1200

Python Pandas DataFrame

```
Inv2 = {
  "Item": ["coffee", "chocolate", "tea",
"water"],
  "Promotion": ["no","no","yes","yes"],
  "Price": [5.95, 5.95, 3.95, 2.95],
  "Stock": [100, 250, 1000, 1200]
}

InvDF = pd.DataFrame(Inv2)
InvDF

InvDF = InvDF.replace({'Promotion': {'no':
False,'yes': True}})
```

	Item	Promotion	Price	Stock
0	coffee	no	5.95	100
1	chocolate	no	5.95	250
2	tea	yes	3.95	1000
3	water	yes	2.95	1200

	Item	Promotion	Price	Stock
0	coffee	False	5.95	100
1	chocolate	False	5.95	250
2	tea	True	3.95	1000
3	water	True	2.95	1200

Python Pandas DataFrame

	Item	Promotion	Price	Stock
0	coffee	False	5.95	100
1	chocolate	False	5.95	250
2	tea	True	3.95	1000
3	water	True	2.95	1200

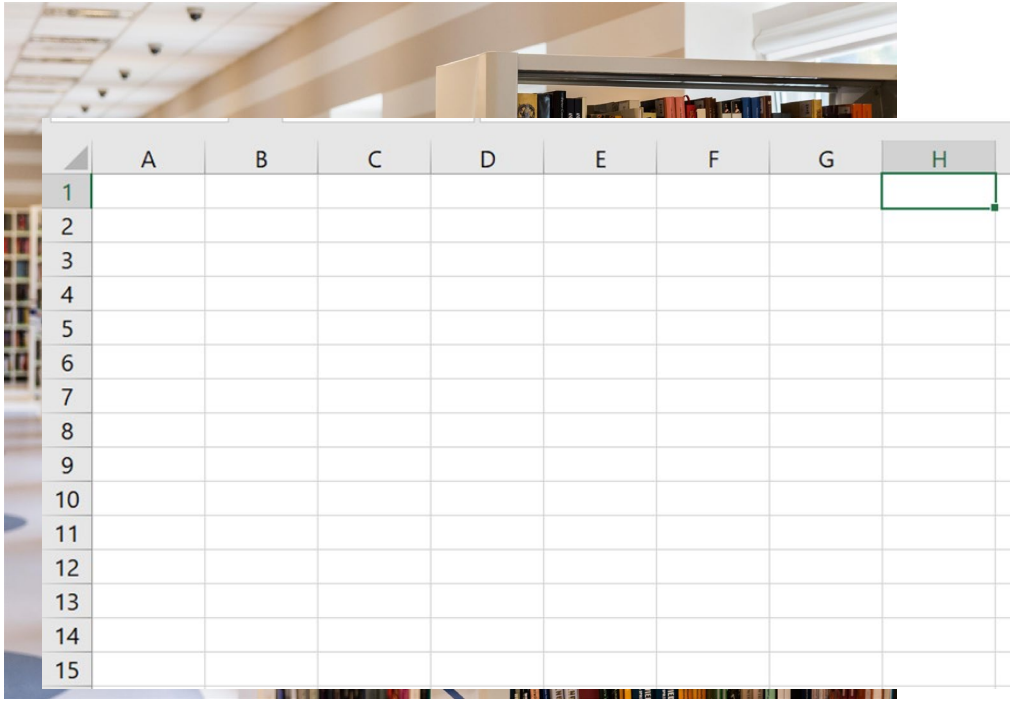
InvDF[InvDF["Promotion"] == False]

	Item	Promotion	Price	Stock
0	coffee	False	5.95	100
1	chocolate	False	5.95	250

InvDF[InvDF["Price"] < 5]

	Item	Promotion	Price	Stock
2	tea	True	3.95	1000
3	water	True	2.95	1200

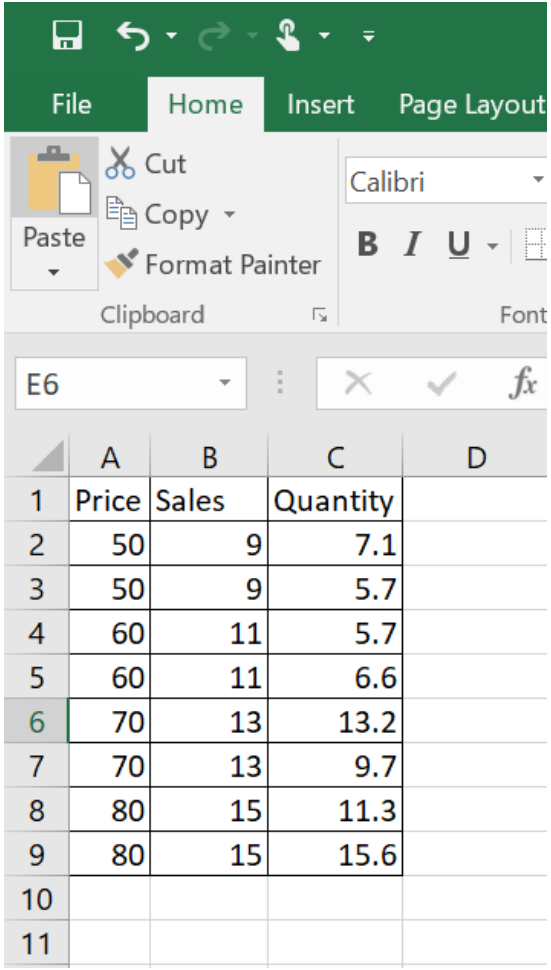
Data Science: Python Pandas DataFrame from Excel



Pandas **DataFrame** Review

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types

Data Science: Python Pandas DataFrame from Excel



	A	B	C	D
1	Price	Sales	Quantity	
2	50	9	7.1	
3	50	9	5.7	
4	60	11	5.7	
5	60	11	6.6	
6	70	13	13.2	
7	70	13	9.7	
8	80	15	11.3	
9	80	15	15.6	
10				
11				

Data Science: Python Pandas DataFrame from Excel

STEPS:

1. import the pandas library
2. Create a variable for the data frame to store all columns and values from the Excel worksheet
3. Use the API from pandas pandas.read_excel(“filename.xlsx”) to read the file “filename.xlsx” and assign all columns and values to the data frame variable

Data Science: Python Pandas DataFrame from Excel

In [19]:
```
import pandas as pd

df = pd.read_excel ("QtyDemand.xlsx")

print(df)
```

```
   Price  Sales  Quantity
0     50      9       7.1
1     50      9       5.7
2     60     11       5.7
3     60     11       6.6
4     70     13      13.2
5     70     13       9.7
6     80     15      11.3
7     80     15      15.6
```

In [ ]:

Data Science: Python Pandas DataFrame from Excel

- You can access the specific column by referencing the index (column label) of that column
  - **df["Price"]** will return the values for the entire column "Price"
  - You can also use the form df.Price
- 0, 1, 2, 3, 4, 5, 6, 7 on the left most column is the index for the rows. You can access the value stored in column Price row 0 using **df.Price[index]**
  - **df.Price[0]** will return the value of the first element of column "Price" = 50
  - **df.Price[6]** will return the value of the seventh element of column "Price" = 80
  - **df.Sales[2]** = 11
  - **df.Quantity[4]** = 13.2

```
In [19]: import pandas as pd

         df = pd.read_excel ("QtyDemand.xlsx")

         print(df)
```

	Price	Sales	Quantity
0	50	9	7.1
1	50	9	5.7
2	60	11	5.7
3	60	11	6.6
4	70	13	13.2
5	70	13	9.7
6	80	15	11.3
7	80	15	15.6

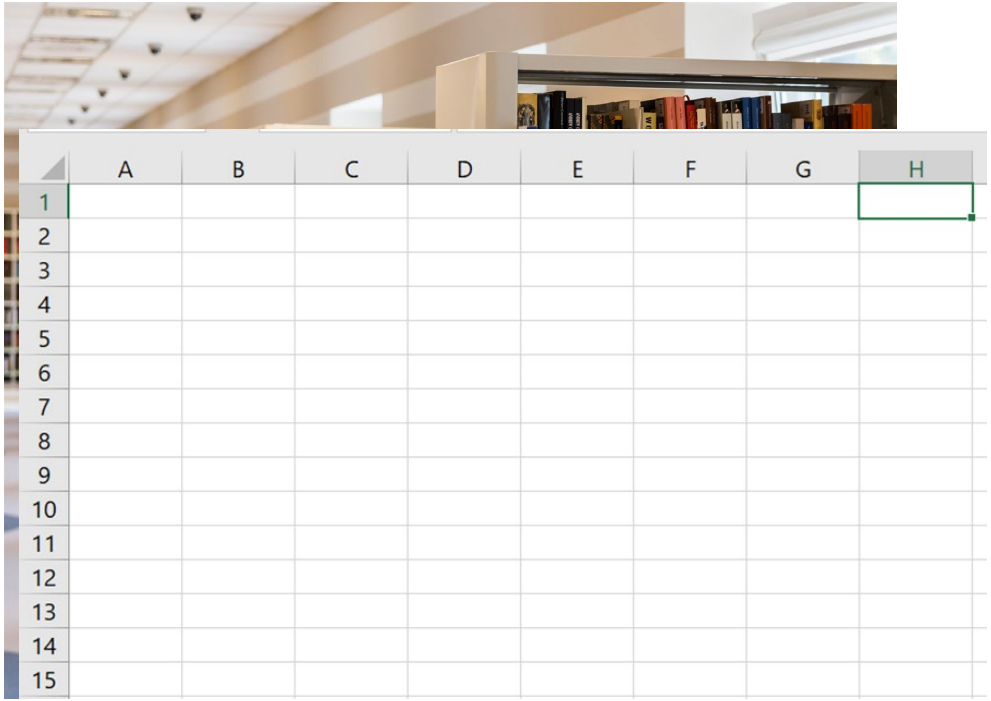
```
In [ ]:
```

Data Science: Python Pandas DataFrame from Excel

```
In [22]: print(df.Price[0])

50

In [23]: print(df.Price[6])

80

In [24]: print(df.Sales[2])

11

In [25]: print(df.Quantity[4])

13.2
```

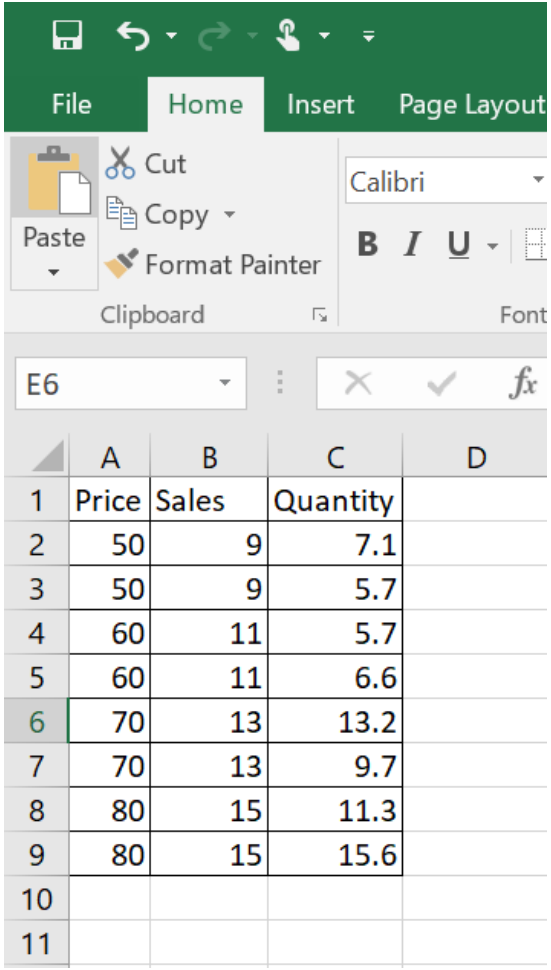
```
In [21]: print(df["Price"])

0    50
1    50
2    60
3    60
4    70
5    70
6    80
7    80
Name: Price, dtype: int64
```

Data Science: Python Pandas DataFrame from Excel Columns



pandas.read_excel(“filename.xlsx”)

Data Science: Python Pandas DataFrame from Excel



	A	B	C	D
1	Price	Sales	Quantity	
2	50	9	7.1	
3	50	9	5.7	
4	60	11	5.7	
5	60	11	6.6	
6	70	13	13.2	
7	70	13	9.7	
8	80	15	11.3	
9	80	15	15.6	
10				
11				

What if we only want to read column “Price” into the dataframe variable?

Data Science: Python Pandas DataFrame from Excel

STEPS:

1. import the pandas library
2. Create a variable for the data frame to store all columns and values from the Excel worksheet
3. Use the API from pandas pandas.read_excel(“filename.xlsx”) to read the file “filename.xlsx” and assign all columns and values to the data frame variable.
    1. This time we will use an additional argument named usecols.
    2. pandas.read_excel(“filename.xlsx”, usecols=[0]) to read first column only
    3. pandas.read_excel(“filename.xlsx”, usecols=[0, 1]) to read first column and second column only

Data Science: Python Pandas DataFrame from Excel

```python
import pandas as pd

df = pd.read_excel ("QtyDemand.xlsx", usecols = [0])

print(df)
```

```
   Price
0     50
1     50
2     60
3     60
4     70
5     70
6     80
7     80
```

```python
import pandas as pd

df = pd.read_excel ("QtyDemand.xlsx", usecols = [0, 2])

print(df)
```

```
   Price  Quantity
0     50       7.1
1     50       5.7
2     60       5.7
3     60       6.6
4     70      13.2
5     70       9.7
6     80      11.3
7     80      15.6
```

Data Science: Python Pandas DataFrame Saving to Excel File

```python
import pandas as pd

df = pd.read_excel ("QtyDemand.xlsx", usecols = [0])

print(df)
```

```
   Price
0     50
1     50
2     60
3     60
4     70
5     70
6     80
7     80
```

```python
df.to_excel ("test2.xlsx")
```

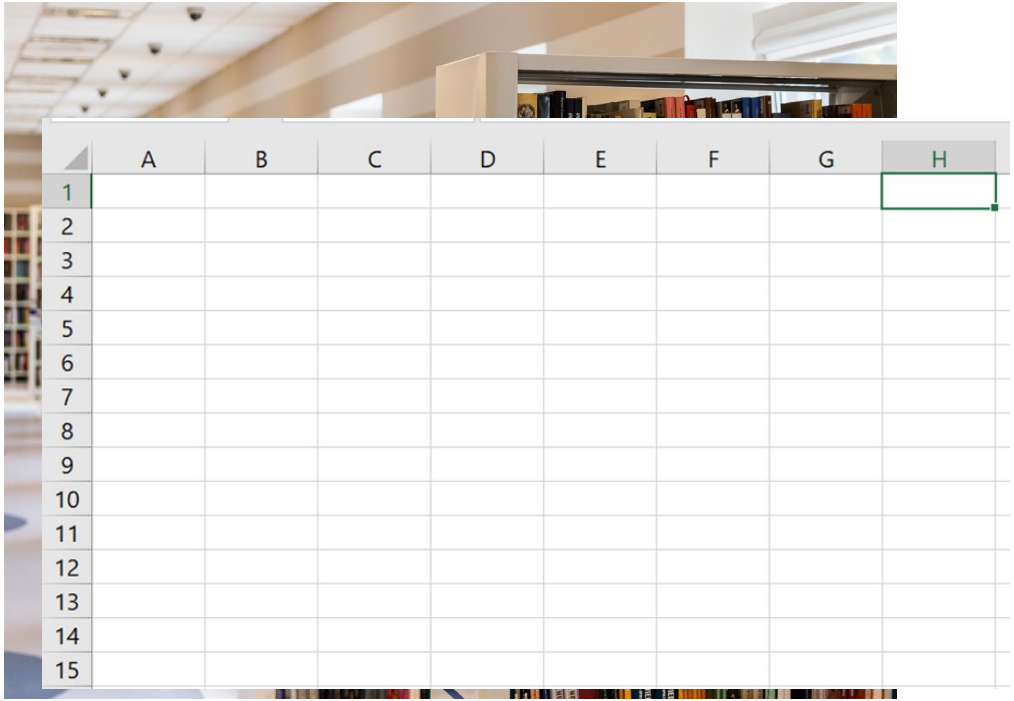
A1

	A	B	C
1		Price	
2	0	50	
3	1	50	
4	2	60	
5	3	60	
6	4	70	
7	5	70	
8	6	80	
9	7	80	

```python
df.to_excel ("test2.xlsx", index = False)
```

A	B
Price	
50	
50	
60	
60	
70	
70	
80	
80	

Data Science: Python Pandas DataFrame from csv



Data Science: Python Pandas DataFrame from csv

```
Price,Sales,Quantity
50,9,7.1
50,9,5.7
60,11,5.7
60,11,6.6
70,13,13.2
70,13,9.7
80,15,11.3
80,15,15.6
```

Data Science: Python Pandas DataFrame from csv

STEPS:

1. import the pandas library
2. Create a variable for the data frame to store all columns and values from the csv file
3. Use the API from pandas pandas.read_csv(“filename.csv”) to read the file “filename.csv” and assign all data to the data frame variable

Data Science: Python Pandas DataFrame from csv

```python
import pandas as pd

df = pd.read_csv ("QtyDemand.csv")

print(df)
```

```
   Price  Sales  Quantity
0     50      9       7.1
1     50      9       5.7
2     60     11       5.7
3     60     11       6.6
4     70     13      13.2
5     70     13       9.7
6     80     15      11.3
7     80     15      15.6
```

Data Science: Python Pandas DataFrame from csv

Specific Column

```
import pandas as pd

df = pd.read_csv ("QtyDemand.csv", usecols = [0])

print(df)
```

```
   Price
0     50
1     50
2     60
3     60
4     70
5     70
6     80
7     80
```

Data Science: Python Pandas DataFrame Saving to a csv file

```python
import pandas as pd

df = pd.read_csv ("QtyDemand.csv", usecols = [0])

print(df)
```

```
   Price
0     50
1     50
2     60
3     60
4     70
5     70
6     80
7     80
```

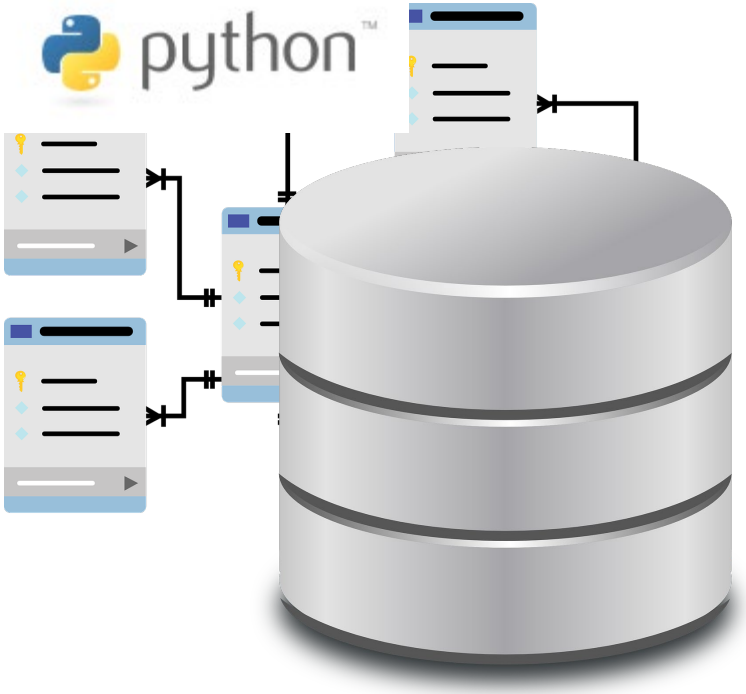
```python
df.to_csv ("test3.csv")
```

	A	B
1		Price
2	0	50
3	1	50
4	2	60
5	3	60
6	4	70
7	5	70
8	6	80
9	7	80

```python
df.to_csv ("test3.csv", index = False)
```

A	B
Price	
50	
50	
60	
60	
70	
70	
80	
80	

Python and MySQL Integration - Revisited



Python and MySQL Integration – Data Cleaning

```
import pandas as pd
import numpy as np

df = pd.read_csv('cookies dirty data.csv')

# drop rows if all elements are blank
df.dropna(how="all", inplace = True)

# remove leading/trailing spaces
df["Salesman"] = df["Salesman"].str.strip()

# replaces invalid values with NaN for one column
df['Price'] = df['Price'].replace('[^A-Za-z0-9]',np.NaN,regex=True)

# replaces invalid values with NaN for multiple columns
df[['Tweets','Sales']] = df[['Tweets','Sales']].replace('[^0-9]',np.NaN,regex=True)

# drop rows if one or more elements are blank
df = df.dropna()
```

Python and MySQL Integration – DateTime Object

	Date	Day	Temperature	Salesman	Tweets	Price	Sales
0	1/1/2019	Tuesday	72.0	John	2	0.5	177
2	1/3/2019	Thursday	69.0	John	5	0.5	172
3	1/4/2019	Friday	100.0	John	7	0.5	150
5	1/6/2019	Sunday	91.0	Ada	8	0.5	120
6	1/7/2019	Monday	81.0	Ada	3	0.3	96

```
df['Date'] = pd.to_datetime(df['Date'], format = '%m/%d/%Y')
```

	Date	Day	Temperature	Salesman	Tweets	Price	Sales
0	2019-01-01	Tuesday	72.0	John	2	0.5	177
2	2019-01-03	Thursday	69.0	John	5	0.5	172
3	2019-01-04	Friday	100.0	John	7	0.5	150
5	2019-01-06	Sunday	91.0	Ada	8	0.5	120
6	2019-01-07	Monday	81.0	Ada	3	0.3	96

Python and MySQL Integration – DataTime Object

	Date	Day	Temperature	Salesman	Tweets	Price	Sales
0	1/1/2019	Tuesday	72.0	John	2	0.5	177
2	1/3/2019	Thursday	69.0	John	5	0.5	172
3	1/4/2019	Friday	100.0	John	7	0.5	150
5	1/6/2019	Sunday	91.0	Ada	8	0.5	120
6	1/7/2019	Monday	81.0	Ada	3	0.3	96

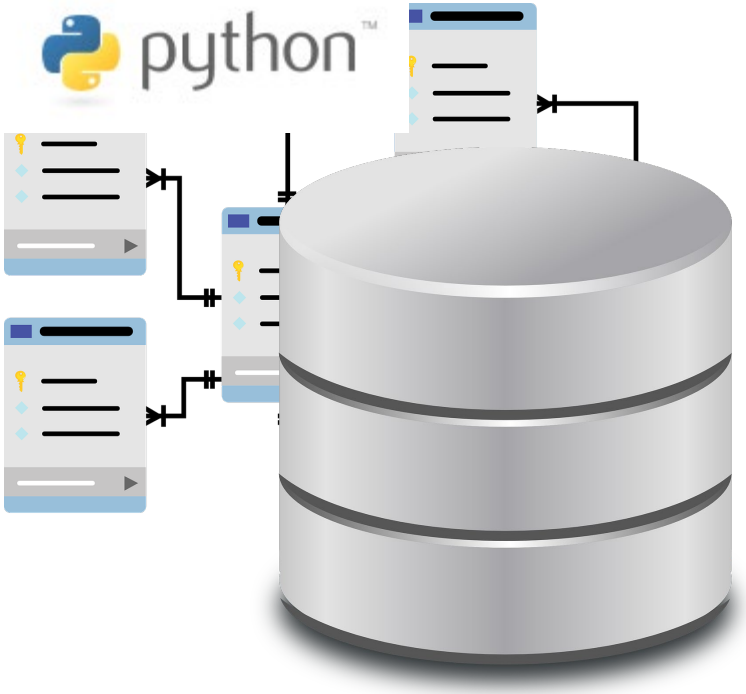
```
df = pd.read_csv('cookies dirty data.csv', parse_dates=['Date'])
```

	Date	Day	Temperature	Salesman	Tweets	Price	Sales
0	2019-01-01	Tuesday	72.0	John	2	0.5	177
2	2019-01-03	Thursday	69.0	John	5	0.5	172
3	2019-01-04	Friday	100.0	John	7	0.5	150
5	2019-01-06	Sunday	91.0	Ada	8	0.5	120
6	2019-01-07	Monday	81.0	Ada	3	0.3	96

Python and MySQL Integration – to_csv

```
df.to_csv("cookies_clean1.csv", index= False)
```

Python Data Filtering



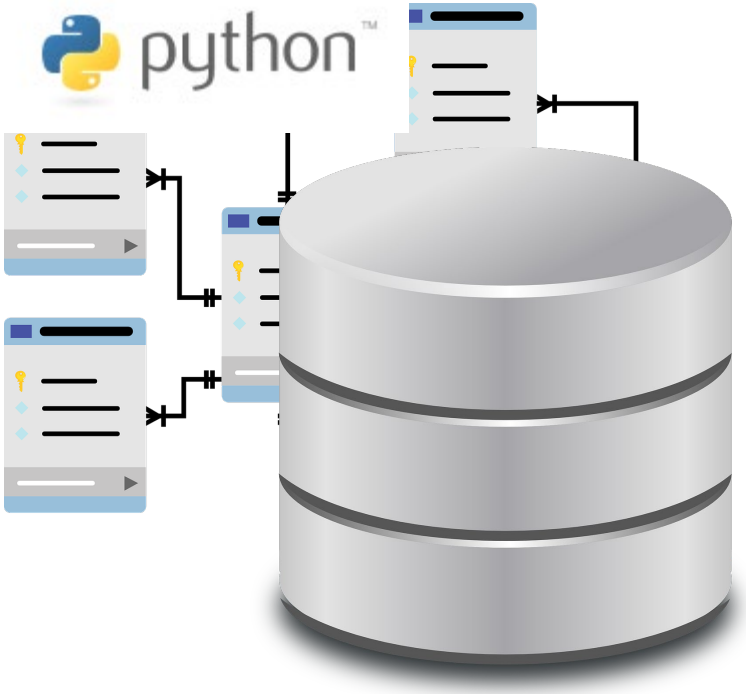
Python Data Filtering

```
df_filter = df['Salesman'] == 'Ada'

df[df_filter]

df_filter = (df['Salesman'] == 'Ada') & (df['Day'] == 'Monday')

df[df_filter]

df_filter = (df['Day'] == 'Tuesday') | (df['Day'] == 'Monday')

df[df_filter]

df_filter = (df['Sales'] >= 100) & (df['Tweets'] <= 1)
```

Python Data Filtering

```
df['Sales'] = pd.to_numeric(df['Sales'],errors='coerce')

df['Tweets'] = pd.to_numeric(df['Tweets'],errors='coerce')

df[df_filter]
```

Python and MySQL Integration – Pivot Table



Python and MySQL Integration – Pivot Table

```
df.pivot(index="Date", columns ="Salesman")

df.pivot(index="Date", columns ="Salesman", values = "Sales")

df.pivot(index="Salesman", columns ="Date", values = "Sales")
```

Python and MySQL Integration – Pivot Table

```
df.pivot_table(index="y", columns ="marital")

df.pivot_table(index="y", columns ="education", values = "age" )

df.pivot_table(index="marital", columns ="y", values = "balance")

df.pivot_table(index="marital", columns ="y", values = "age",
aggfunc="sum")
```