Lab03 - PySpark

Wei-Lun Tseng (Eric)
ECE, NYCU
eric840610.ee02@g2.nctu.edu.tw

2022/05/26

Class Schedule

Total: 3 points

- Lab 3-1: PySpark Warmup
- Lab 3-2: PySpark Exercise

Checkpoint

Lab 3-1_PySpark WarmUp

- Checkpoint 1: Successfully install PySpark on Colab (0.5 point)
- Checkpoint 2: How many people whose age larger than 18? (0.5 point)

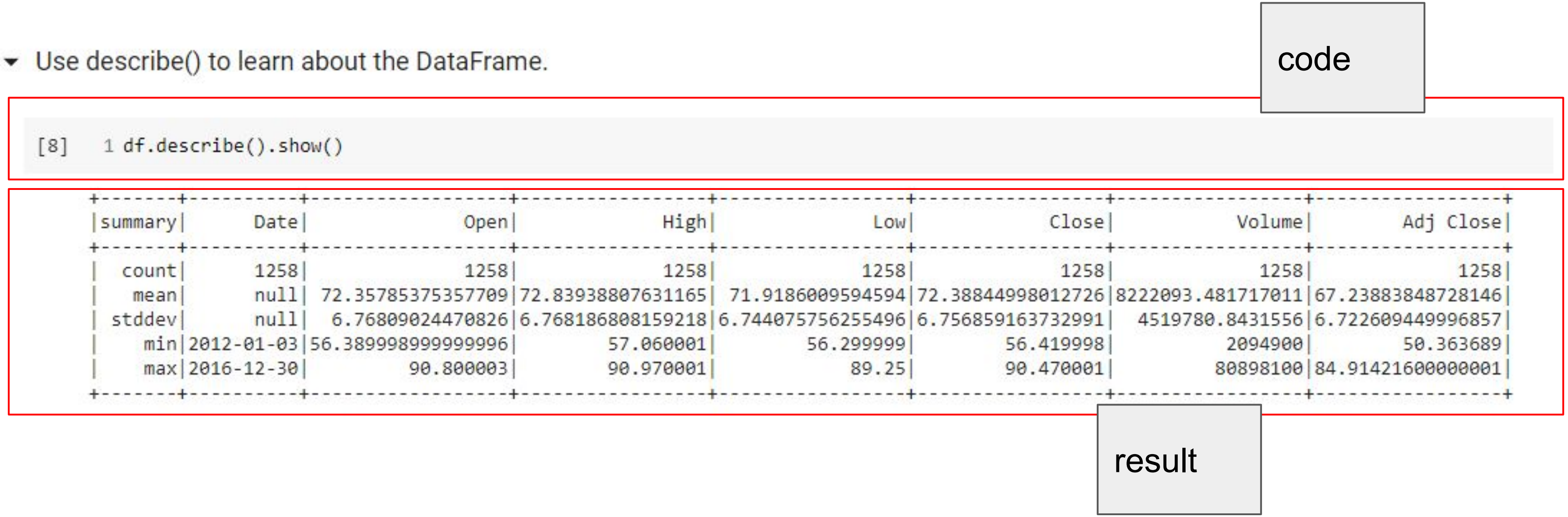
Lab 3-2

- Checkpoint 3: format Open, High, Low, Close, Volume, Adj Close (0.5 point)
- Checkpoint 4: What is the max and min of the Volume column? (0.5 point)
- Checkpoint 5: What is the average Close for each Calendar Month? (0.5 point)

Checkpoint 6: Upload screenshot in correct naming rule (0.5 point)

Reminder

- The screenshot needs contain coding block and results
- Example:



Use describe() to learn about the DataFrame.

```
[8]  df.describe().show()
```

```
+-------+----------+------------------+------------------+-----------------+------------------+-----------------+-----------------+
|summary|      Date|              Open|              High|              Low|             Close|           Volume|        Adj Close|
+-------+----------+------------------+------------------+-----------------+------------------+-----------------+-----------------+
|  count|      1258|              1258|              1258|             1258|              1258|             1258|             1258|
|   mean|      null| 72.35785375357709| 72.83938807631165| 71.9186009594594| 72.38844998012726|8222093.481717011|67.23883848728146|
| stddev|      null|  6.76809024470826| 6.768186808159218|6.744075756255496| 6.756859163732991|  4519780.8431556| 6.72260944996857|
|    min|2012-01-03|56.389998999999996|         57.060001|        56.299999|         56.419998|          2094900|        50.363689|
|    max|2016-12-30|         90.800003|         90.970001|            89.25|         90.470001|         80898100|84.91421600000001|
+-------+----------+------------------+------------------+-----------------+------------------+-----------------+-----------------+
```

code

result

Python and Spark

- In this course the main way we will be working with Python and Spark is through the DataFrame Syntax.
- If you’ve worked with pandas in Python, R, SQL or even Excel, a DataFrame will feel very familiar!

Python and Spark

- Spark DataFrames hold data in a column and row format.
- Each column represents some feature or variable.
- Each row represents an individual data point.

Python and Spark

- Spark began with something known as the “RDD” syntax which was a little ugly and tricky to learn.
- Now Spark 2.0 and higher has shifted towards a DataFrame syntax which is much cleaner and easier to work with!

Python and Spark

- Spark DataFrames are able to input and output data from a wide variety of sources.
- We can then use these DataFrames to apply various transformations on the data.

Python and Spark

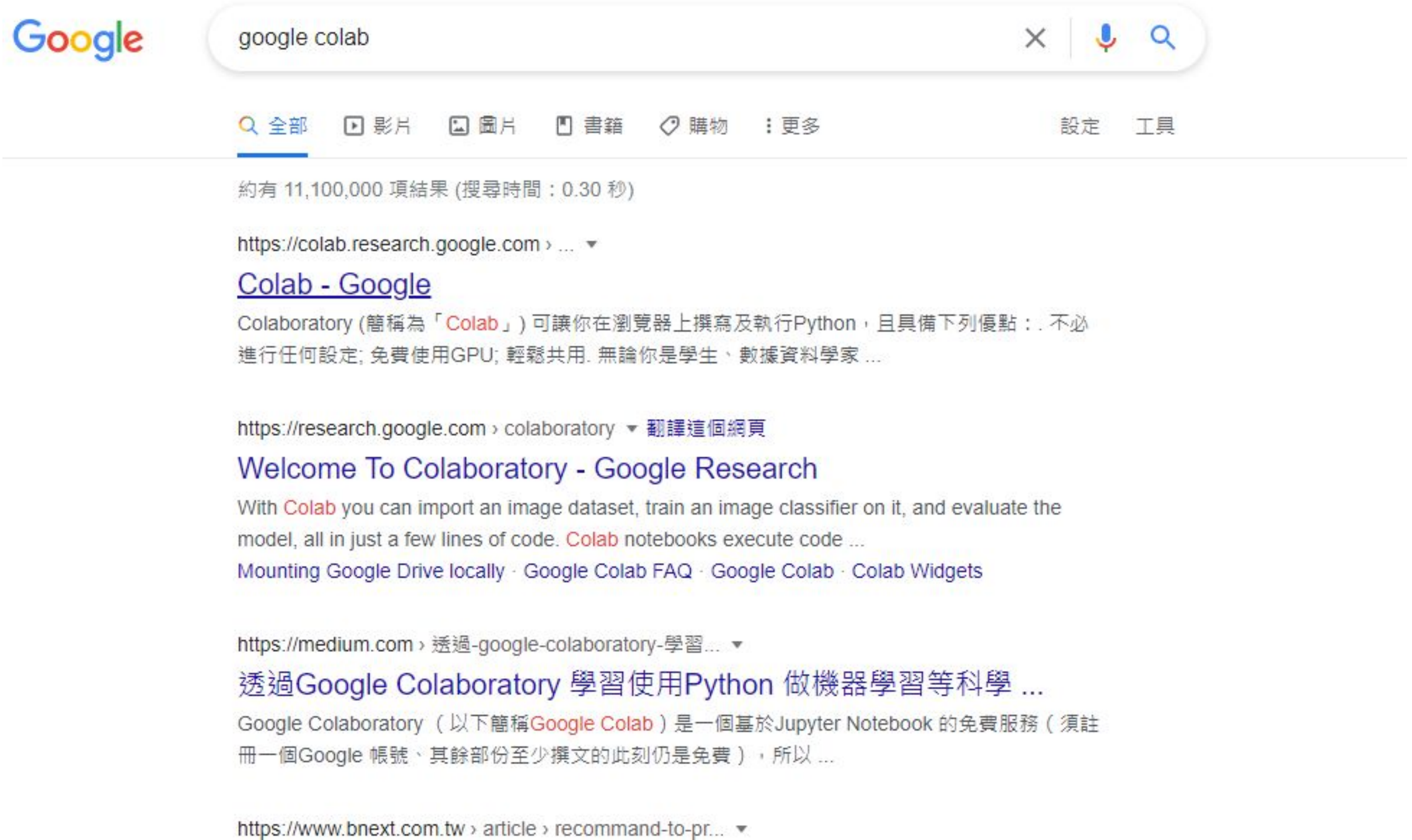
- At the end of the transformation calls, we can either show or collect the results to display or for some final processing.
- In this section we’ll cover all the main features of working with DataFrames that you need to know.

Python and Spark

- Once we have a solid understanding of Spark DataFrames, we can move on to utilizing the DataFrame MLlib API for Machine Learning.

Lab 3-1

- PySpark - Warm Up



約有 11,100,000 項結果 (搜尋時間：0.30 秒)

https://colab.research.google.com › ...

Colab - Google

Colaboratory (簡稱為「Colab」) 可讓你在瀏覽器上撰寫及執行Python，且具備下列優點：. 不必進行任何設定; 免費使用GPU; 輕鬆共用. 無論你是學生、數據資料學家 ...

https://research.google.com › colaboratory 翻譯這個網頁

Welcome To Colaboratory - Google Research

With Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code. Colab notebooks execute code ...

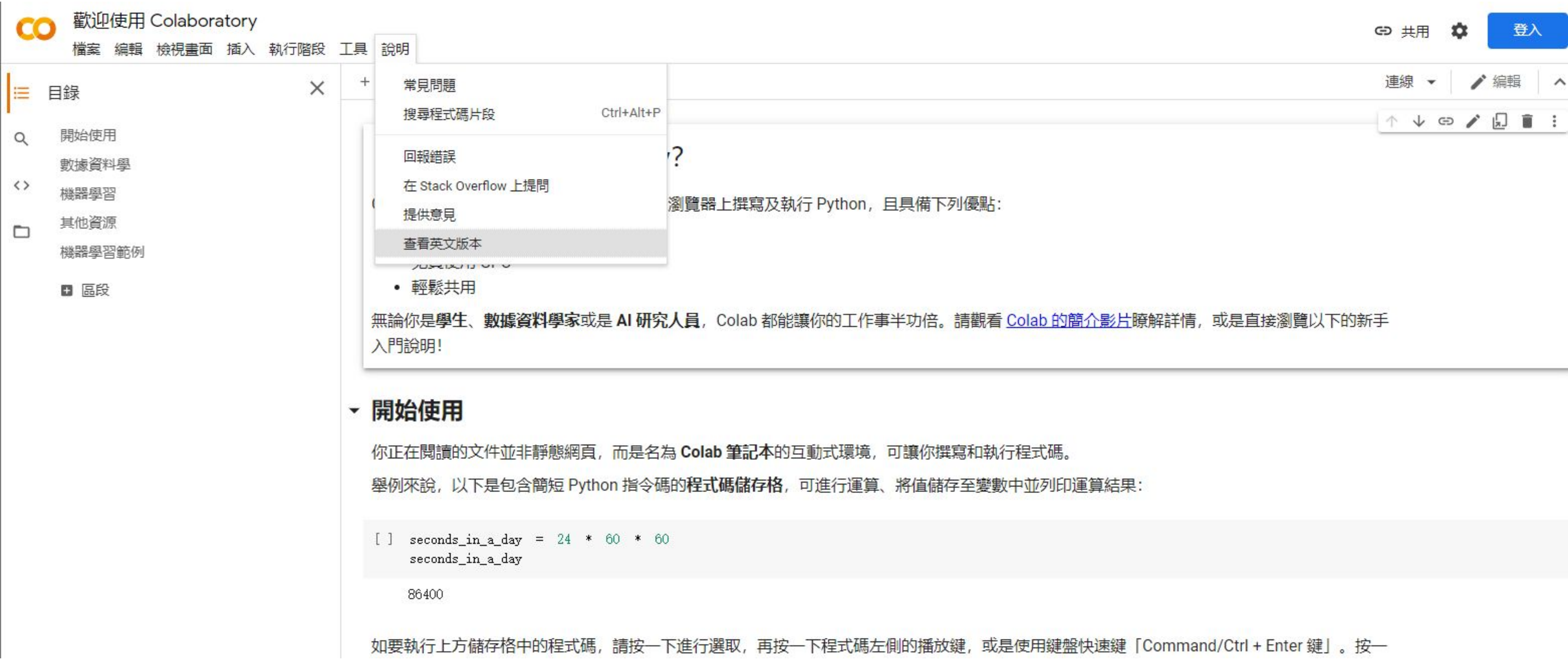
Mounting Google Drive locally · Google Colab FAQ · Google Colab · Colab Widgets

https://medium.com › 透過-google-colaboratory-學習...

透過Google Colaboratory 學習使用Python 做機器學習等科學 ...

Google Colaboratory （以下簡稱Google Colab ）是一個基於Jupyter Notebook 的免費服務（須註冊一個Google 帳號、其餘部份至少撰文的此刻仍是免費），所以 ...

https://www.bnext.com.tw › article › recommand-to-pr...



?

瀏覽器上撰寫及執行 Python，且具備下列優點：

- 輕鬆共用

無論你是**學生**、**數據資料學家**或是 **AI 研究人員**，Colab 都能讓你的工作事半功倍。請觀看 Colab 的簡介影片瞭解詳情，或是直接瀏覽以下的新手入門說明！

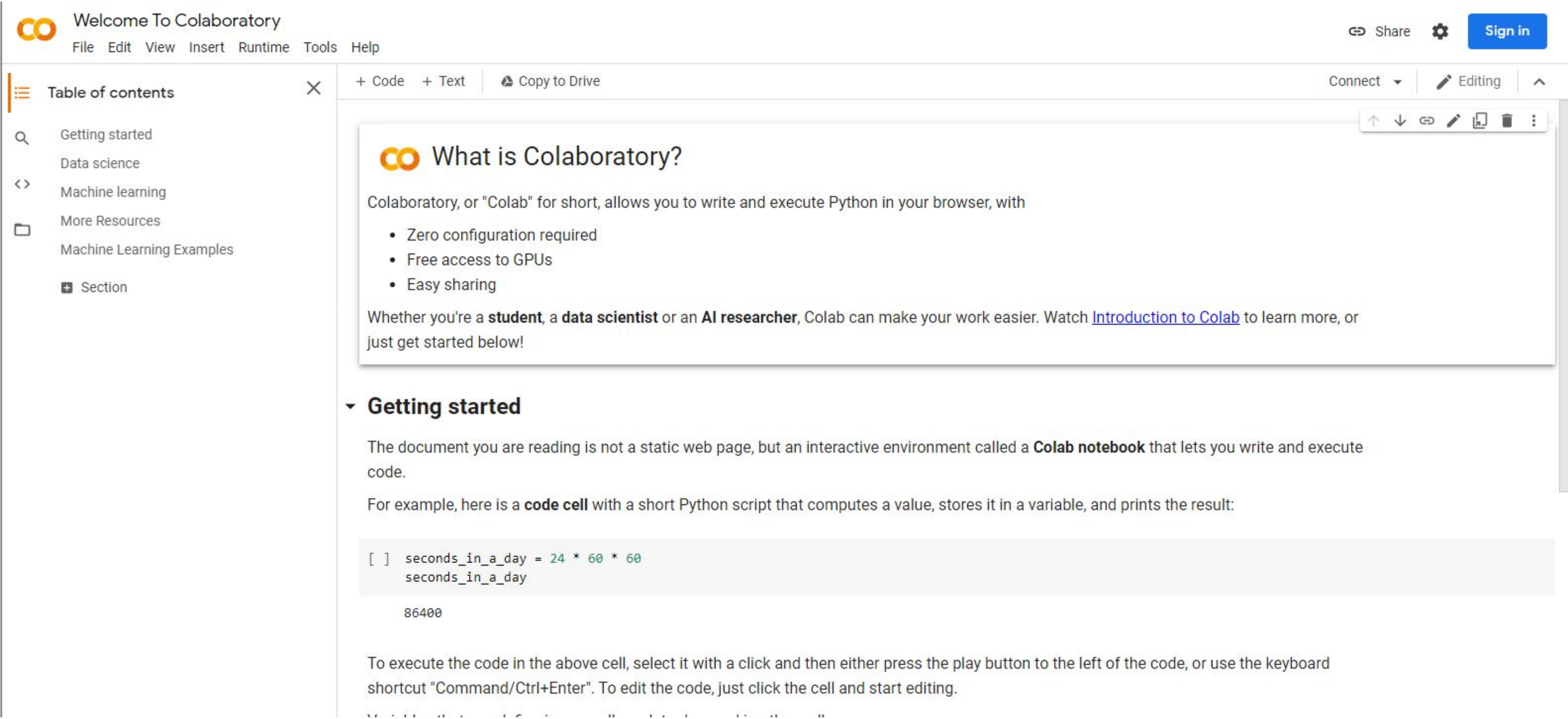
開始使用

你正在閱讀的文件並非靜態網頁，而是名為 **Colab 筆記本**的互動式環境，可讓你撰寫和執行程式碼。

舉例來說，以下是包含簡短 Python 指令碼的**程式碼儲存格**，可進行運算、將值儲存至變數中並列印運算結果：

```
[ ]  seconds_in_a_day = 24 * 60 * 60
     seconds_in_a_day

     86400
```

如要執行上方儲存格中的程式碼，請按一下進行選取，再按一下程式碼左側的播放鍵，或是使用鍵盤快速鍵「Command/Ctrl + Enter 鍵」。按一



What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab to learn more, or just get started below!

Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ]  seconds_in_a_day = 24 * 60 * 60
     seconds_in_a_day
```

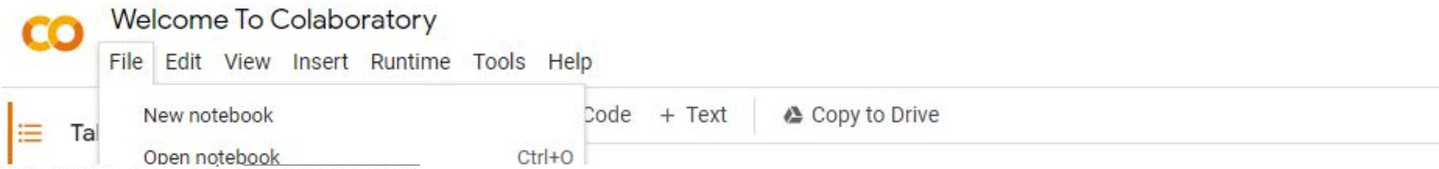
```
86400
```

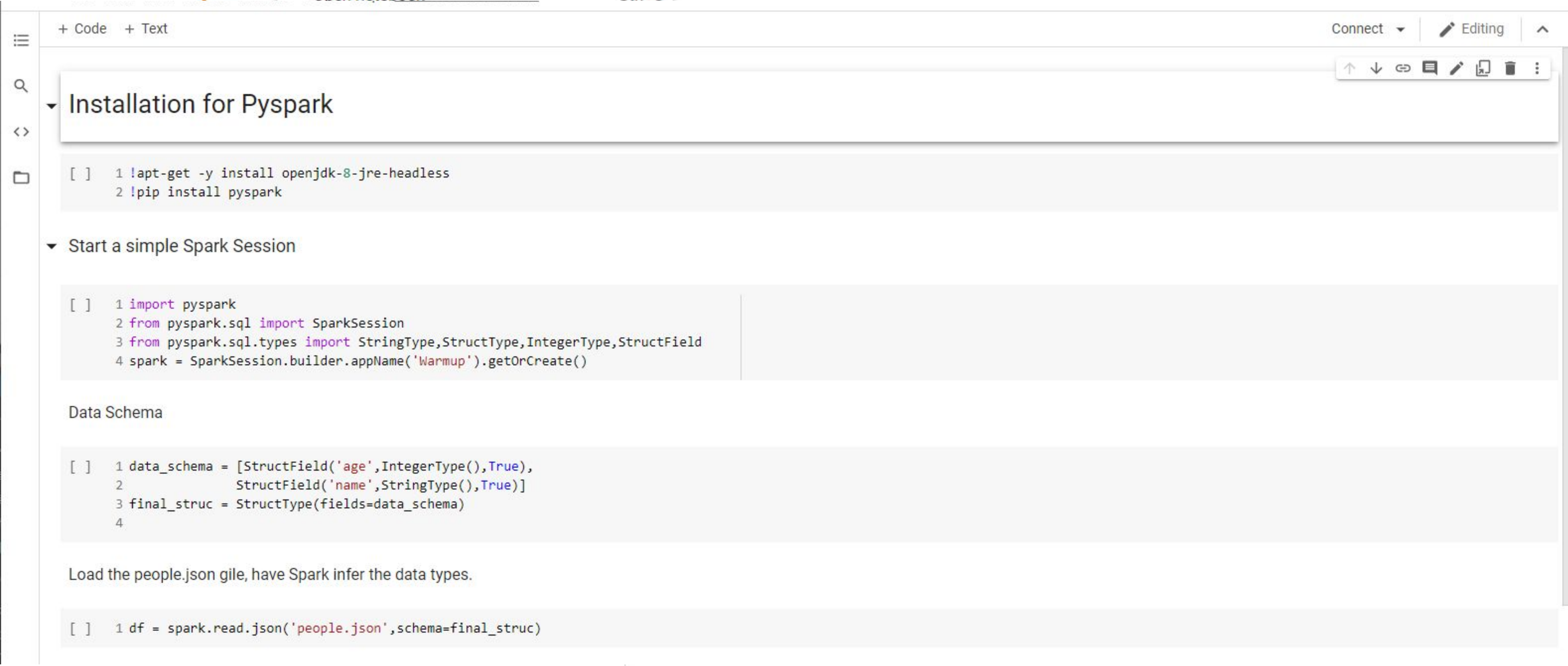
To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

Login your account…

Lab 3-1: PySpark Warm Up

- File:
  - 2022-Lab-3-1_PySpark WarmUp.ipynb
  - people.json
- Checkpoint:
  - Checkpoint 1: Successfully install PySpark on Colab (0.5 pts)
  - Checkpoint 2: How many people whose age smaller than 30? (0.5 pts)





Installation for Pyspark

```
!apt-get -y install openjdk-8-jre-headless
!pip install pyspark
```

Start a simple Spark Session

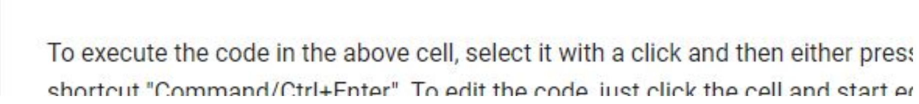
```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.types import StringType,StructType,IntegerType,StructField
spark = SparkSession.builder.appName('Warmup').getOrCreate()
```

Data Schema

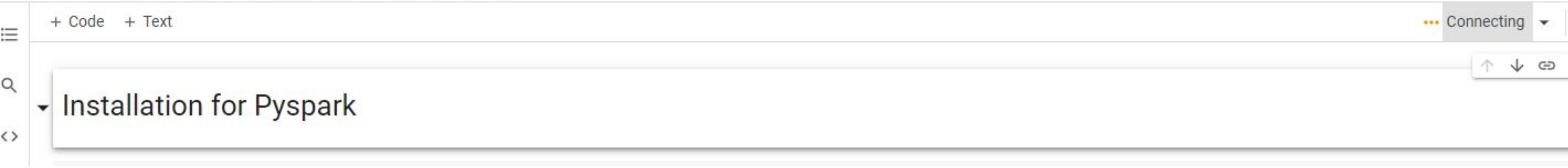
```
data_schema = [StructField('age',IntegerType(),True),
               StructField('name',StringType(),True)]
final_struc = StructType(fields=data_schema)

```

Load the people.json gile, have Spark infer the data types.

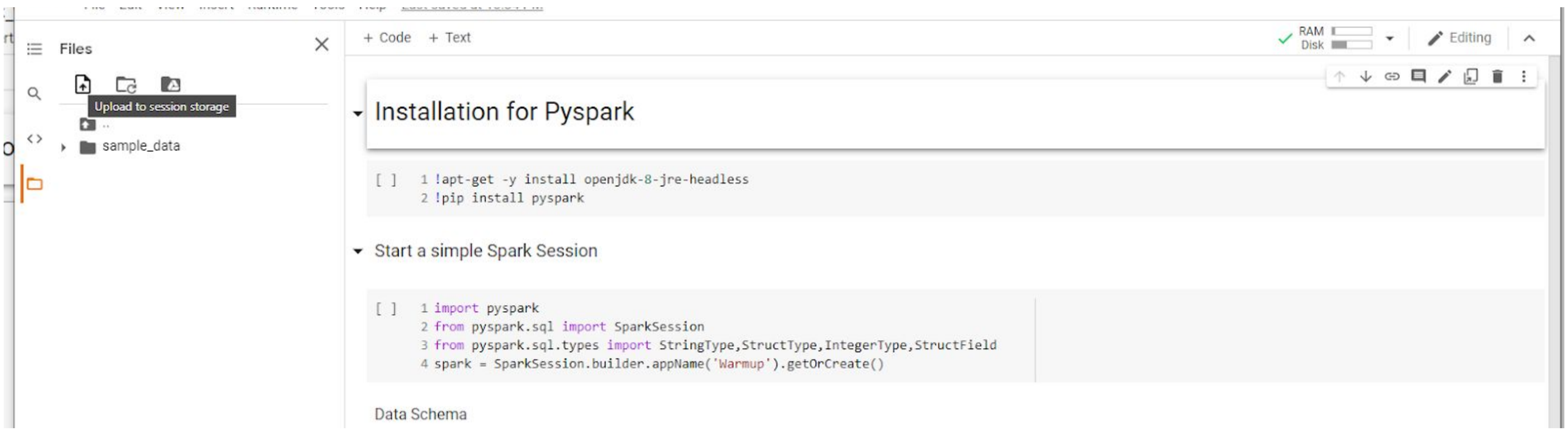
```
df = spark.read.json('people.json',schema=final_struc)
```



To execute the code in the above cell, select it with a click and then either press
shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start e



Installation for Pyspark



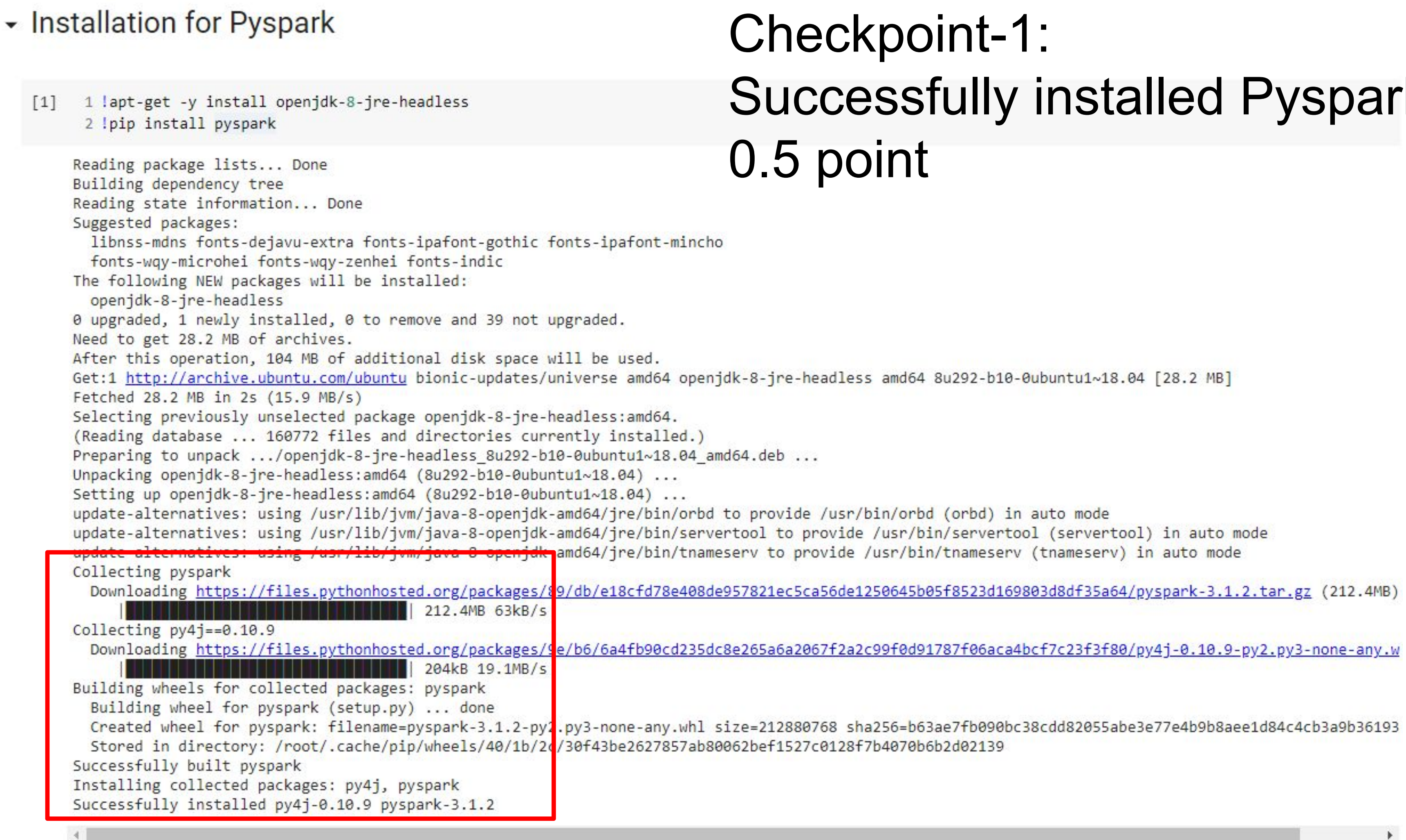
Installation for Pyspark

```
[ ]
!apt-get -y install openjdk-8-jre-headless
!pip install pyspark
```

Start a simple Spark Session

```
[ ]
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.types import StringType,StructType,IntegerType,StructField
spark = SparkSession.builder.appName('Warmup').getOrCreate()
```

Data Schema

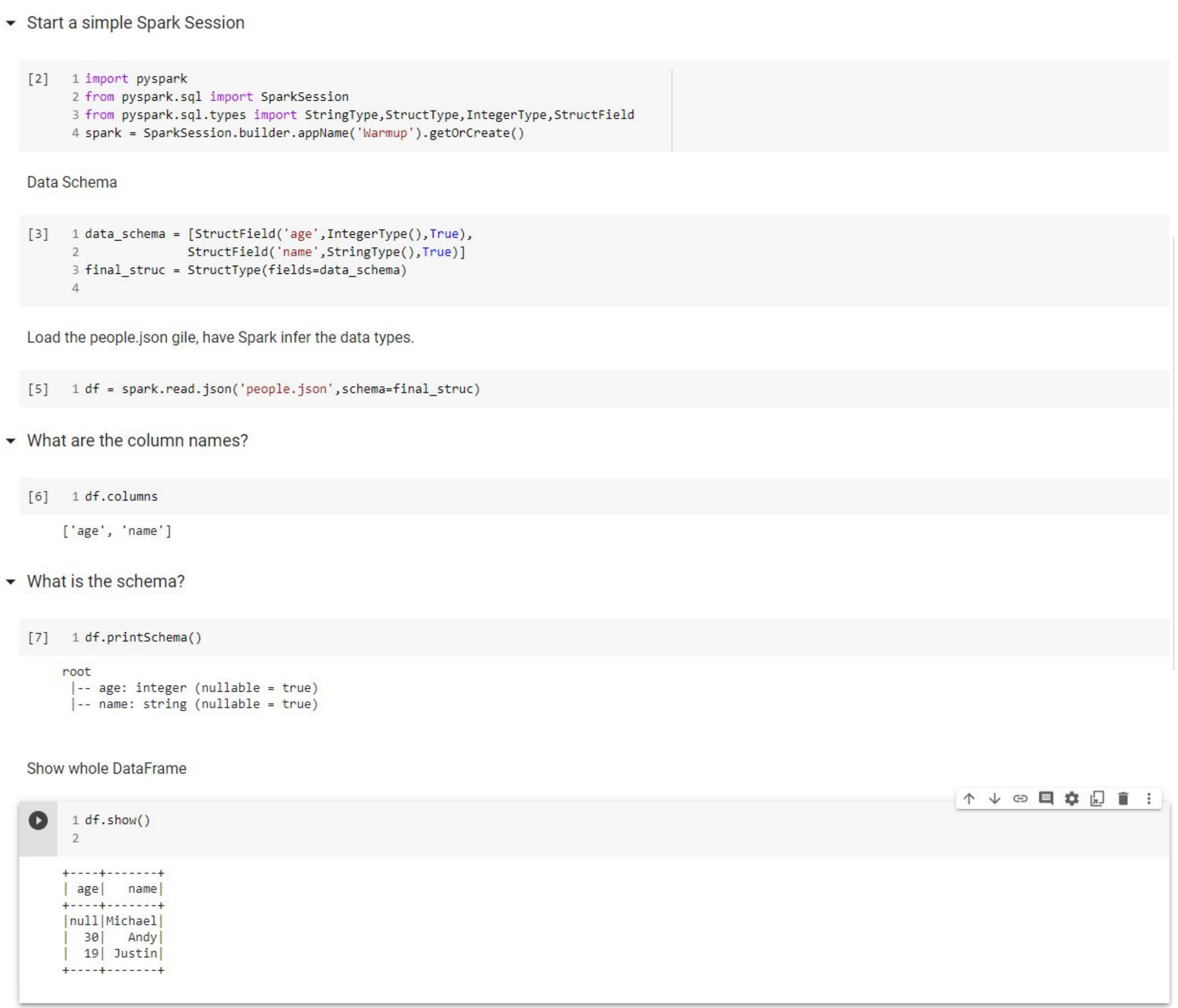


Installation for Pyspark

```
[1]  !apt-get -y install openjdk-8-jre-headless
     !pip install pyspark
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  libnss-mdns fonts-dejavu-extra fonts-ipafont-gothic fonts-ipafont-mincho
  fonts-wqy-microhei fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  openjdk-8-jre-headless
0 upgraded, 1 newly installed, 0 to remove and 39 not upgraded.
Need to get 28.2 MB of archives.
After this operation, 104 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 openjdk-8-jre-headless amd64 8u292-b10-0ubuntu1~18.04 [28.2 MB]
Fetched 28.2 MB in 2s (15.9 MB/s)
Selecting previously unselected package openjdk-8-jre-headless:amd64.
(Reading database ... 160772 files and directories currently installed.)
Preparing to unpack .../openjdk-8-jre-headless_8u292-b10-0ubuntu1~18.04_amd64.deb ...
Unpacking openjdk-8-jre-headless:amd64 (8u292-b10-0ubuntu1~18.04) ...
Setting up openjdk-8-jre-headless:amd64 (8u292-b10-0ubuntu1~18.04) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/orbd to provide /usr/bin/orbd (orbd) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/servertool to provide /usr/bin/servertool (servertool) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/tnameserv to provide /usr/bin/tnameserv (tnameserv) in auto mode
Collecting pyspark
  Downloading https://files.pythonhosted.org/packages/89/db/e18cfd78e408de957821ec5ca56de1250645b05f8523d169803d8df35a64/pyspark-3.1.2.tar.gz (212.4MB)
     |████████████████████████████████| 212.4MB 63kB/s
Collecting py4j==0.10.9
  Downloading https://files.pythonhosted.org/packages/9e/b6/6a4fb90cd235dc8e265a6a2067f2a2c99f0d91787f06aca4bcf7c23f3f80/py4j-0.10.9-py2.py3-none-any.w
     |████████████████████████████████| 204kB 19.1MB/s
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.1.2-py2.py3-none-any.whl size=212880768 sha256=b63ae7fb090bc38cdd82055abe3e77e4b9b8aee1d84c4cb3a9b36193
  Stored in directory: /root/.cache/pip/wheels/40/1b/2c/30f43be2627857ab80062bef1527c0128f7b4070b6b2d02139
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9 pyspark-3.1.2
```

Checkpoint-1:
Successfully installed Pyspark
0.5 point



Start a simple Spark Session

```
[2]
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.types import StringType,StructType,IntegerType,StructField
spark = SparkSession.builder.appName('Warmup').getOrCreate()
```

Data Schema

```
[3]
data_schema = [StructField('age',IntegerType(),True),
               StructField('name',StringType(),True)]
final_struc = StructType(fields=data_schema)

```

Load the people.json gile, have Spark infer the data types.

```
[5]
df = spark.read.json('people.json',schema=final_struc)
```

What are the column names?

```
[6]
df.columns
```

```
['age', 'name']
```

What is the schema?

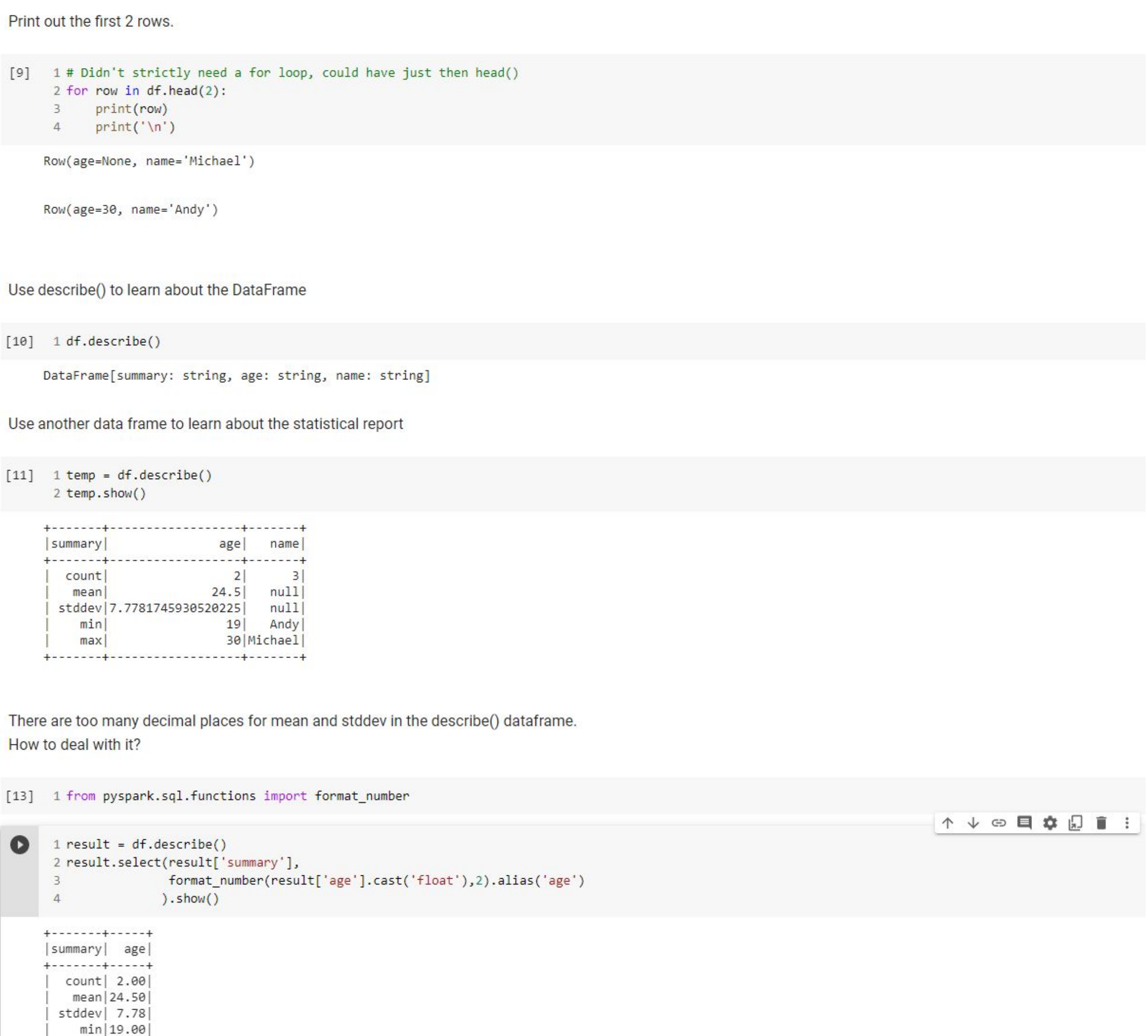
```
[7]
df.printSchema()
```

```
root
 |-- age: integer (nullable = true)
 |-- name: string (nullable = true)
```

Show whole DataFrame

```
df.show()

```

```
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```



Print out the first 2 rows.

[9]

```
# Didn't strictly need a for loop, could have just then head()
for row in df.head(2):
    print(row)
    print('\n')
```

```
Row(age=None, name='Michael')


Row(age=30, name='Andy')
```

Use describe() to learn about the DataFrame

[10]

```
df.describe()
```

```
DataFrame[summary: string, age: string, name: string]
```

Use another data frame to learn about the statistical report

[11]

```
temp = df.describe()
temp.show()
```

```
+-------+------------------+-------+
|summary|               age|   name|
+-------+------------------+-------+
|  count|                 2|      3|
|   mean|              24.5|   null|
| stddev|7.7781745930520225|   null|
|    min|                19|   Andy|
|    max|                30|Michael|
+-------+------------------+-------+
```

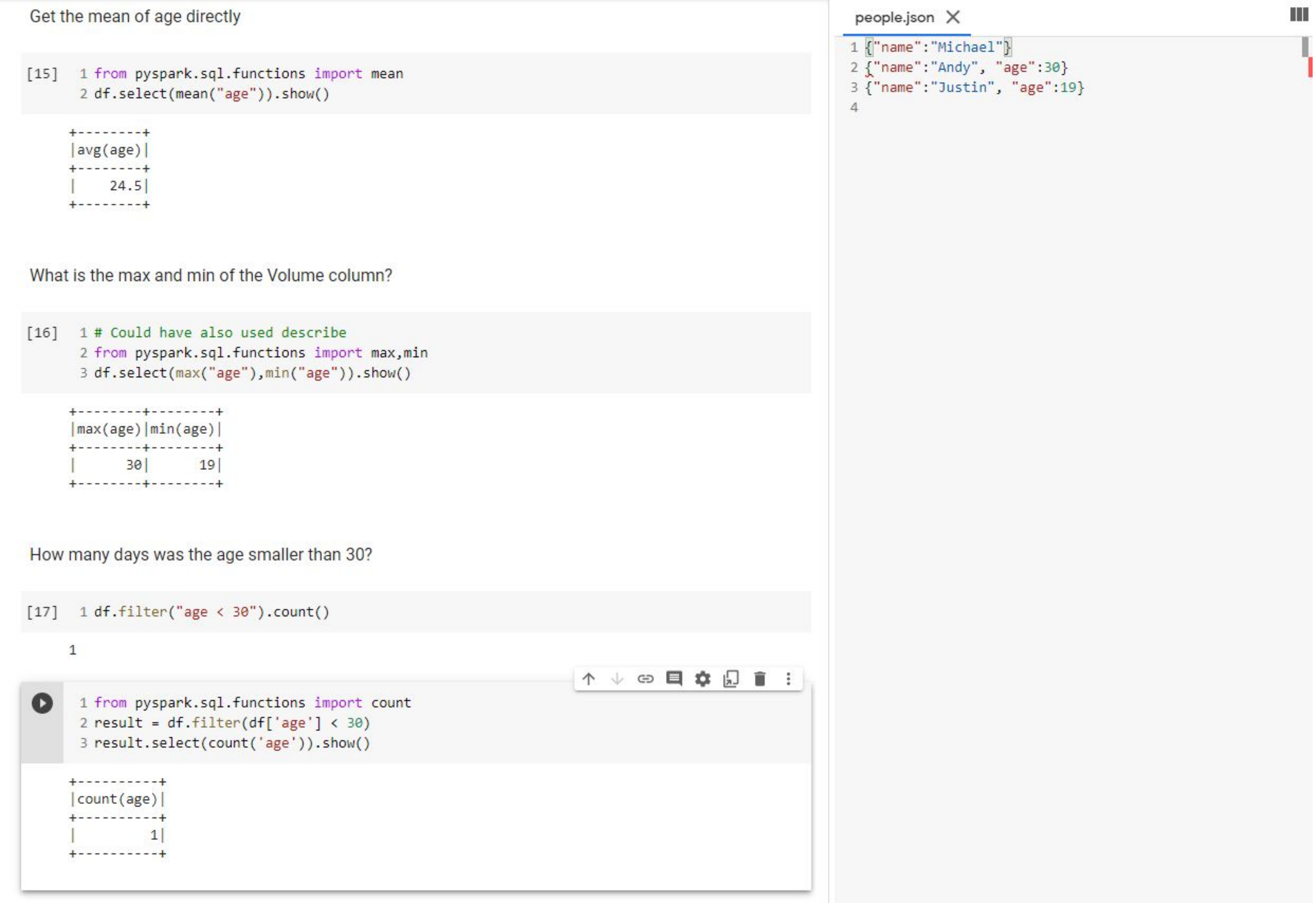
There are too many decimal places for mean and stddev in the describe() dataframe.
How to deal with it?

[13]

```
from pyspark.sql.functions import format_number
```

```
result = df.describe()
result.select(result['summary'],
              format_number(result['age'].cast('float'),2).alias('age')
             ).show()
```

```
+-------+-----+
|summary|  age|
+-------+-----+
|  count| 2.00|
|   mean|24.50|
| stddev| 7.78|
|    min|19.00|
```



Get the mean of age directly

[15]

```
from pyspark.sql.functions import mean
df.select(mean("age")).show()
```

```
+--------+
|avg(age)|
+--------+
|    24.5|
+--------+
```

What is the max and min of the Volume column?

[16]

```
# Could have also used describe
from pyspark.sql.functions import max,min
df.select(max("age"),min("age")).show()
```

```
+--------+--------+
|max(age)|min(age)|
+--------+--------+
|      30|      19|
+--------+--------+
```

How many days was the age smaller than 30?

[17]

```
df.filter("age < 30").count()
```

```
1
```

```
from pyspark.sql.functions import count
result = df.filter(df['age'] < 30)
result.select(count('age')).show()
```

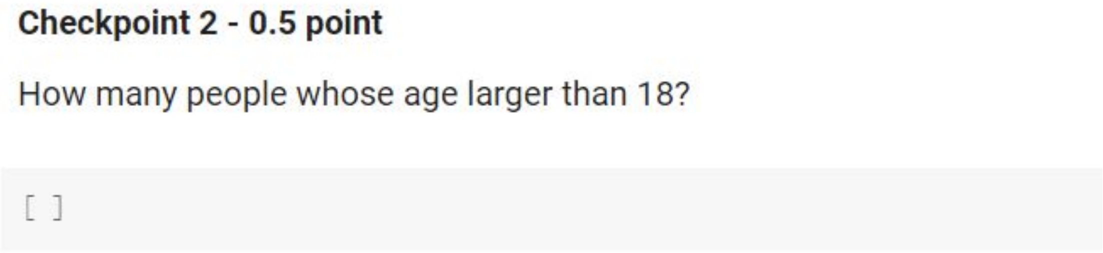
```
+----------+
|count(age)|
+----------+
|         1|
+----------+
```

people.json

```
{"name":"Michael"}
{"name":"Andy", "age":30}
{"name":"Justin", "age":19}
```

Check-Point-2:

0.5 point



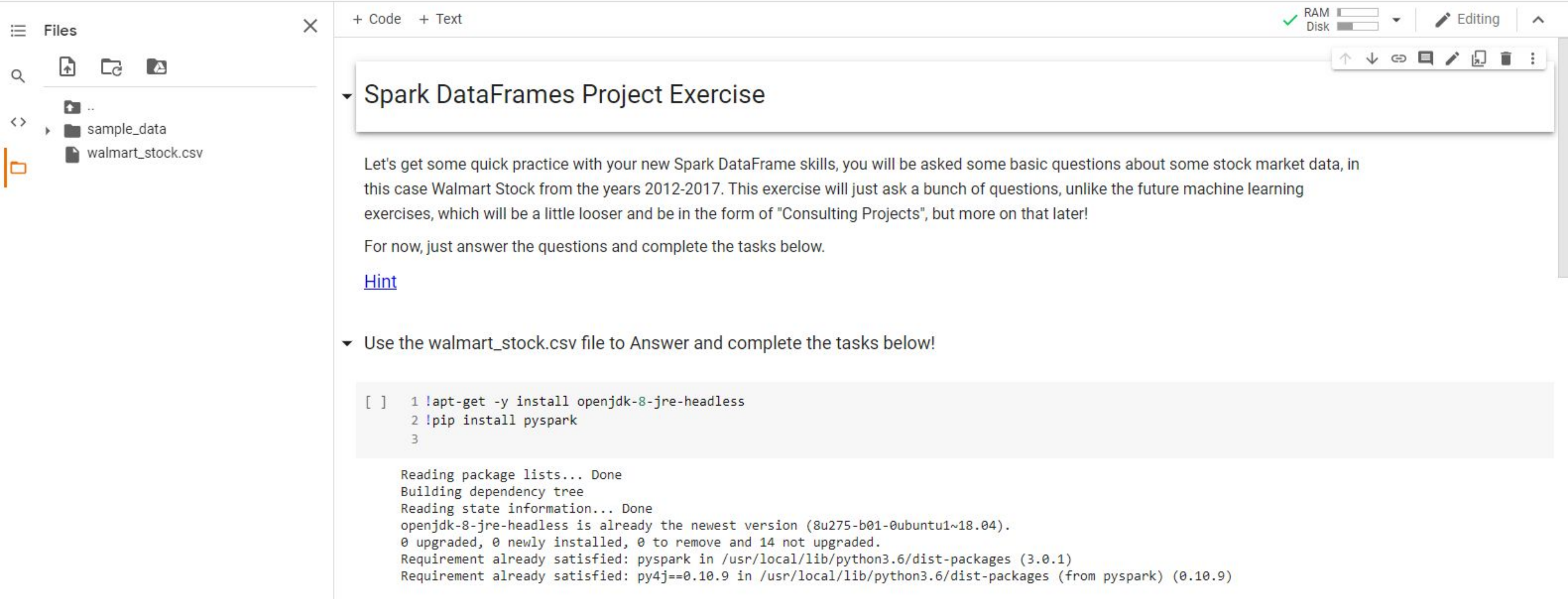
Checkpoint 2 - 0.5 point

How many people whose age larger than 18?

```
[ ]
```

Lab 3-2

- PySpark - Exercise



Spark DataFrames Project Exercise

Let's get some quick practice with your new Spark DataFrame skills, you will be asked some basic questions about some stock market data, in this case Walmart Stock from the years 2012-2017. This exercise will just ask a bunch of questions, unlike the future machine learning exercises, which will be a little looser and be in the form of "Consulting Projects", but more on that later!

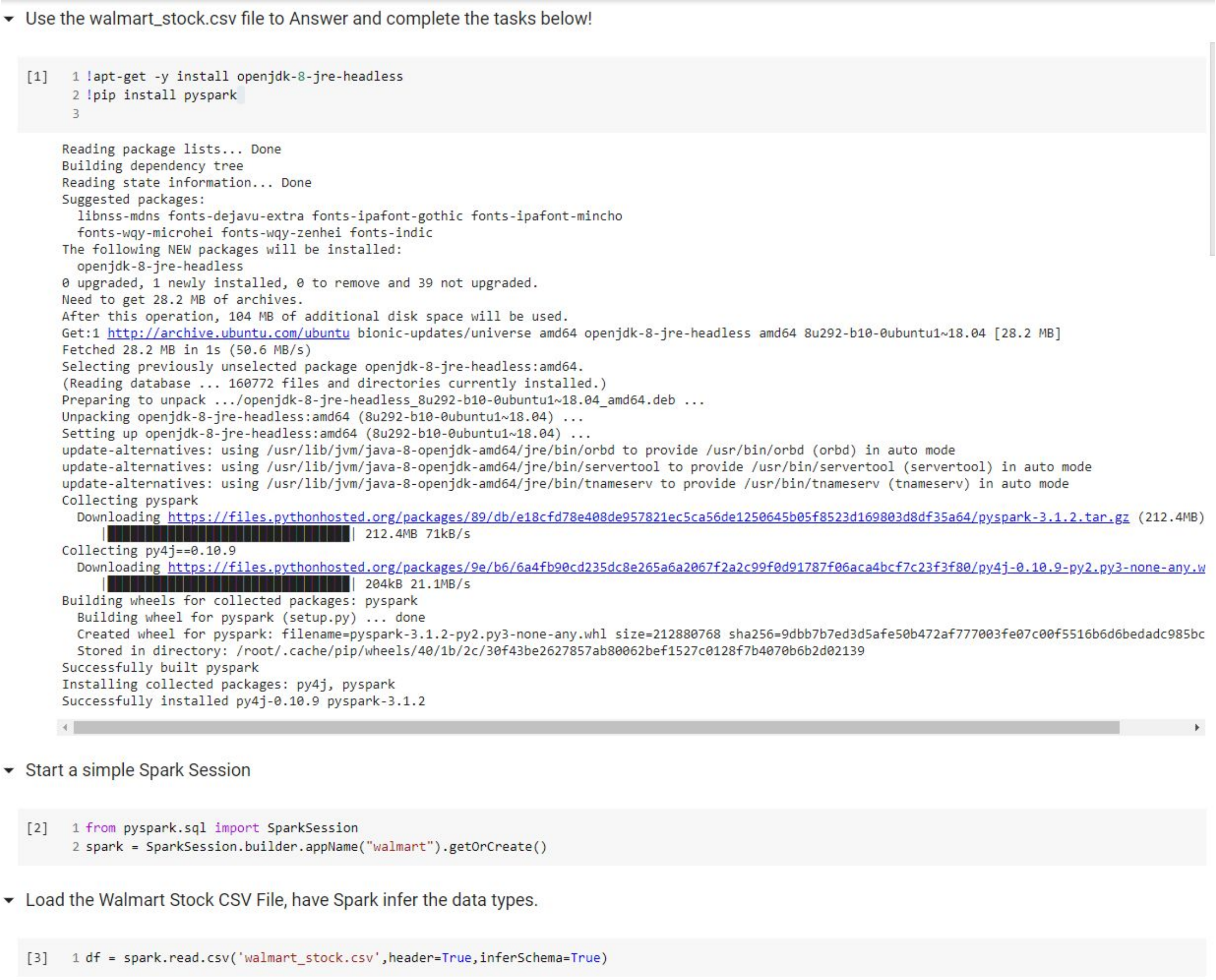
For now, just answer the questions and complete the tasks below.

Hint

Use the walmart_stock.csv file to Answer and complete the tasks below!

```
!apt-get -y install openjdk-8-jre-headless
!pip install pyspark

```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
openjdk-8-jre-headless is already the newest version (8u275-b01-0ubuntu1~18.04).
0 upgraded, 0 newly installed, 0 to remove and 14 not upgraded.
Requirement already satisfied: pyspark in /usr/local/lib/python3.6/dist-packages (3.0.1)
Requirement already satisfied: py4j==0.10.9 in /usr/local/lib/python3.6/dist-packages (from pyspark) (0.10.9)
```



Use the walmart_stock.csv file to Answer and complete the tasks below!

[1]

```
!apt-get -y install openjdk-8-jre-headless
!pip install pyspark

```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  libnss-mdns fonts-dejavu-extra fonts-ipafont-gothic fonts-ipafont-mincho
  fonts-wqy-microhei fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  openjdk-8-jre-headless
0 upgraded, 1 newly installed, 0 to remove and 39 not upgraded.
Need to get 28.2 MB of archives.
After this operation, 104 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 openjdk-8-jre-headless amd64 8u292-b10-0ubuntu1~18.04 [28.2 MB]
Fetched 28.2 MB in 1s (50.6 MB/s)
Selecting previously unselected package openjdk-8-jre-headless:amd64.
(Reading database ... 160772 files and directories currently installed.)
Preparing to unpack .../openjdk-8-jre-headless_8u292-b10-0ubuntu1~18.04_amd64.deb ...
Unpacking openjdk-8-jre-headless:amd64 (8u292-b10-0ubuntu1~18.04) ...
Setting up openjdk-8-jre-headless:amd64 (8u292-b10-0ubuntu1~18.04) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/orbd to provide /usr/bin/orbd (orbd) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/servertool to provide /usr/bin/servertool (servertool) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/tnameserv to provide /usr/bin/tnameserv (tnameserv) in auto mode
Collecting pyspark
  Downloading https://files.pythonhosted.org/packages/89/db/e18cfd78e408de957821ec5ca56de1250645b05f8523d169803d8df35a64/pyspark-3.1.2.tar.gz (212.4MB)
     |████████████████████████████████| 212.4MB 71kB/s
Collecting py4j==0.10.9
  Downloading https://files.pythonhosted.org/packages/9e/b6/6a4fb90cd235dc8e265a6a2067f2a2c99f0d91787f06aca4bcf7c23f3f80/py4j-0.10.9-py2.py3-none-any.w
     |████████████████████████████████| 204kB 21.1MB/s
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.1.2-py2.py3-none-any.whl size=212880768 sha256=9dbb7b7ed3d5afe50b472af777003fe07c00f5516b6d6bedadc985bc
  Stored in directory: /root/.cache/pip/wheels/40/1b/2c/30f43be2627857ab80062bef1527c0128f7b4070b6b2d02139
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9 pyspark-3.1.2
```

Start a simple Spark Session

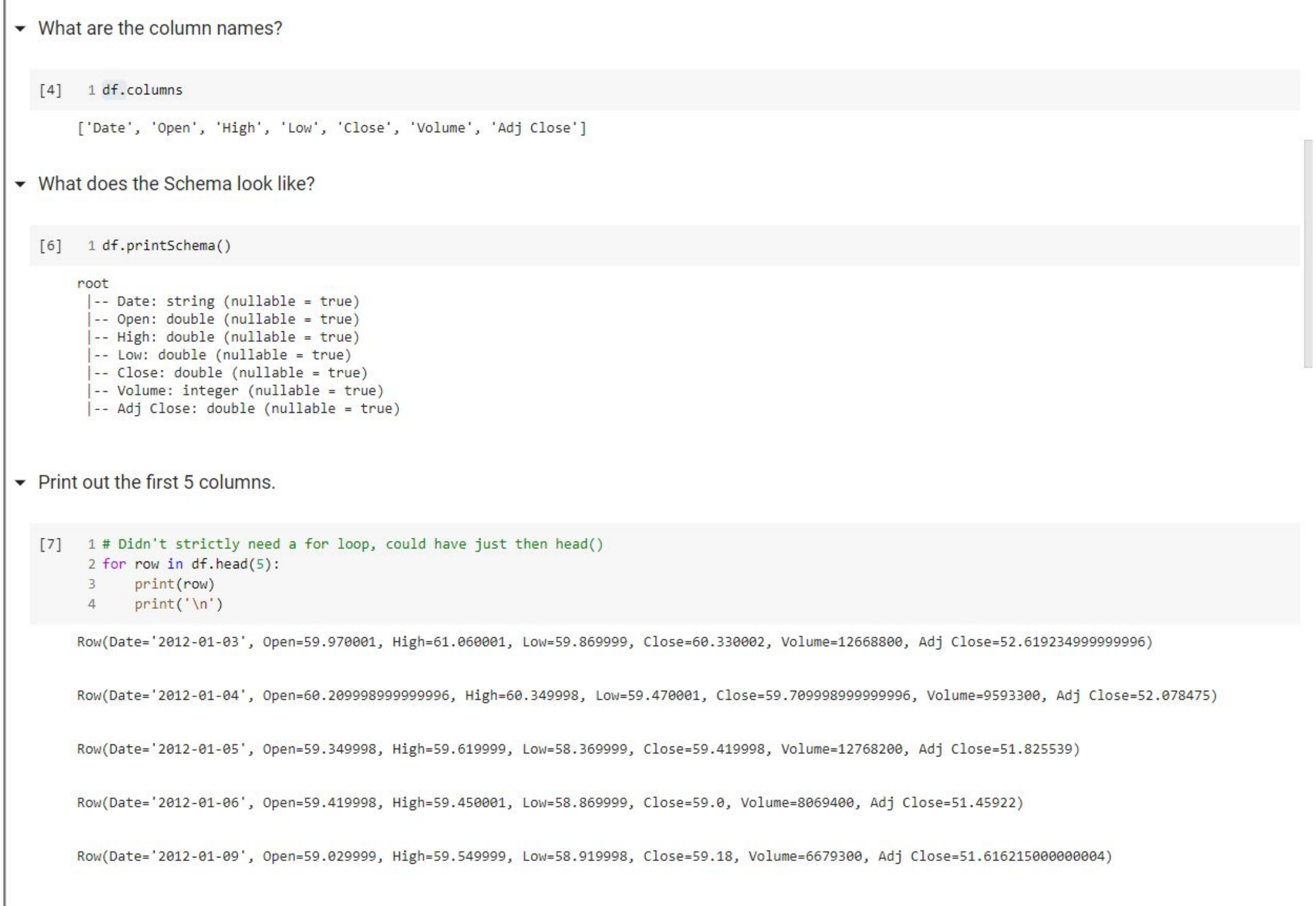
[2]

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("walmart").getOrCreate()
```

Load the Walmart Stock CSV File, have Spark infer the data types.

[3]

```
df = spark.read.csv('walmart_stock.csv',header=True,inferSchema=True)
```



What are the column names?

[4]
```
df.columns
```

```
['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close']
```

What does the Schema look like?

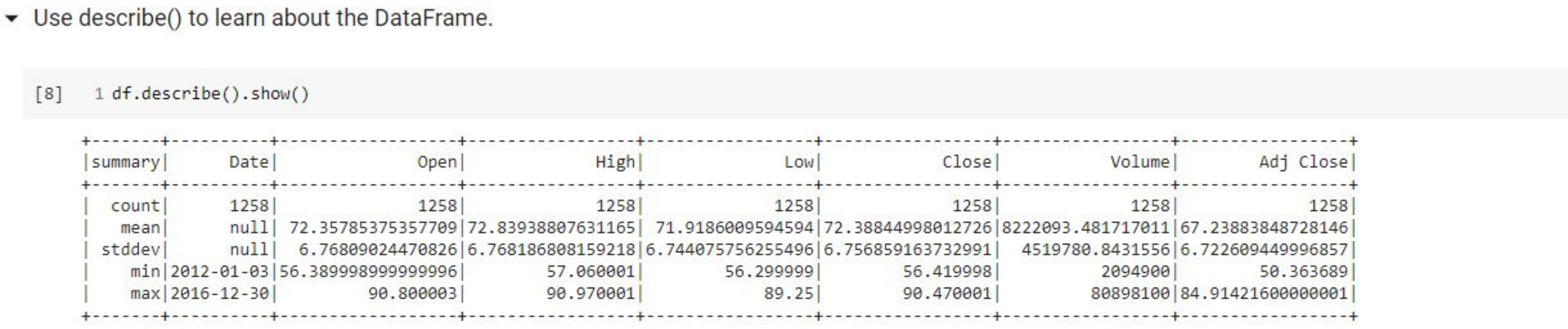
[6]
```
df.printSchema()
```

```
root
 |-- Date: string (nullable = true)
 |-- Open: double (nullable = true)
 |-- High: double (nullable = true)
 |-- Low: double (nullable = true)
 |-- Close: double (nullable = true)
 |-- Volume: integer (nullable = true)
 |-- Adj Close: double (nullable = true)
```

Print out the first 5 columns.

[7]
```
# Didn't strictly need a for loop, could have just then head()
for row in df.head(5):
    print(row)
    print('\n')
```

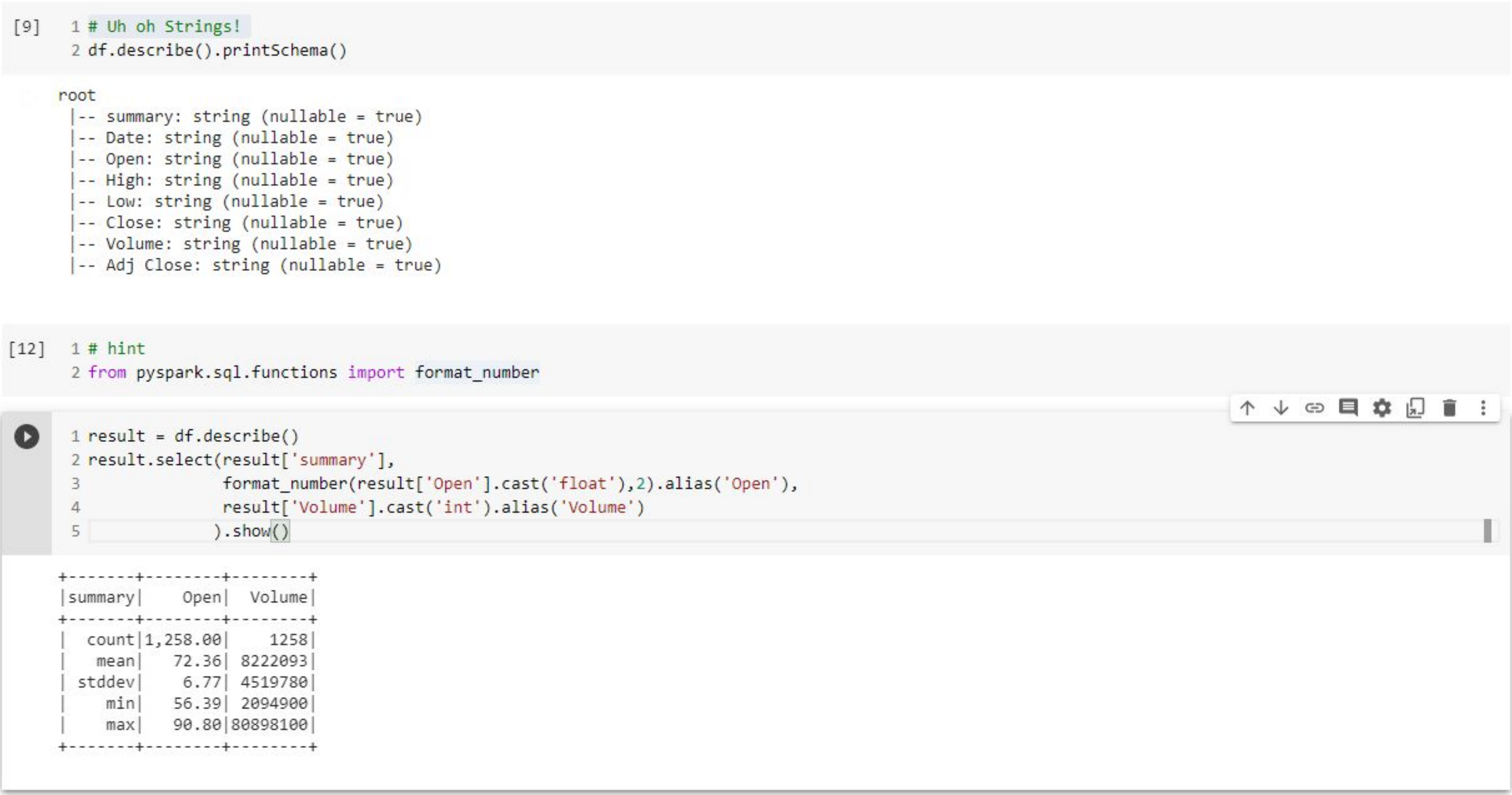
```
Row(Date='2012-01-03', Open=59.970001, High=61.060001, Low=59.869999, Close=60.330002, Volume=12668800, Adj Close=52.619234999999996)


Row(Date='2012-01-04', Open=60.209998999999996, High=60.349998, Low=59.470001, Close=59.709998999999996, Volume=9593300, Adj Close=52.078475)


Row(Date='2012-01-05', Open=59.349998, High=59.619999, Low=58.369999, Close=59.419998, Volume=12768200, Adj Close=51.825539)


Row(Date='2012-01-06', Open=59.419998, High=59.450001, Low=58.869999, Close=59.0, Volume=8069400, Adj Close=51.45922)


Row(Date='2012-01-09', Open=59.029999, High=59.549999, Low=58.919998, Close=59.18, Volume=6679300, Adj Close=51.616215000000004)
```



Use describe() to learn about the DataFrame.

```
[8]  df.describe().show()
```

```
+-------+----------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+
|summary|      Date|              Open|             High|              Low|            Close|           Volume|        Adj Close|
+-------+----------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+
|  count|      1258|              1258|             1258|             1258|             1258|             1258|             1258|
|   mean|      null| 72.35785375357709|72.83938807631165| 71.9186009594594|72.38844998012726|8222093.481717011|67.23883848728146|
| stddev|      null|  6.76809024470826|6.768186808159218|6.744075756255496|6.756859163732991|  4519780.8431556|6.722609449996857|
|    min|2012-01-03|56.389998999999996|        57.060001|        56.299999|        56.419998|          2094900|        50.363689|
|    max|2016-12-30|         90.800003|        90.970001|            89.25|        90.470001|         80898100|84.91421600000001|
+-------+----------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+
```



[9]

```
# Uh oh Strings!
df.describe().printSchema()
```

```
root
 |-- summary: string (nullable = true)
 |-- Date: string (nullable = true)
 |-- Open: string (nullable = true)
 |-- High: string (nullable = true)
 |-- Low: string (nullable = true)
 |-- Close: string (nullable = true)
 |-- Volume: string (nullable = true)
 |-- Adj Close: string (nullable = true)
```

[12]

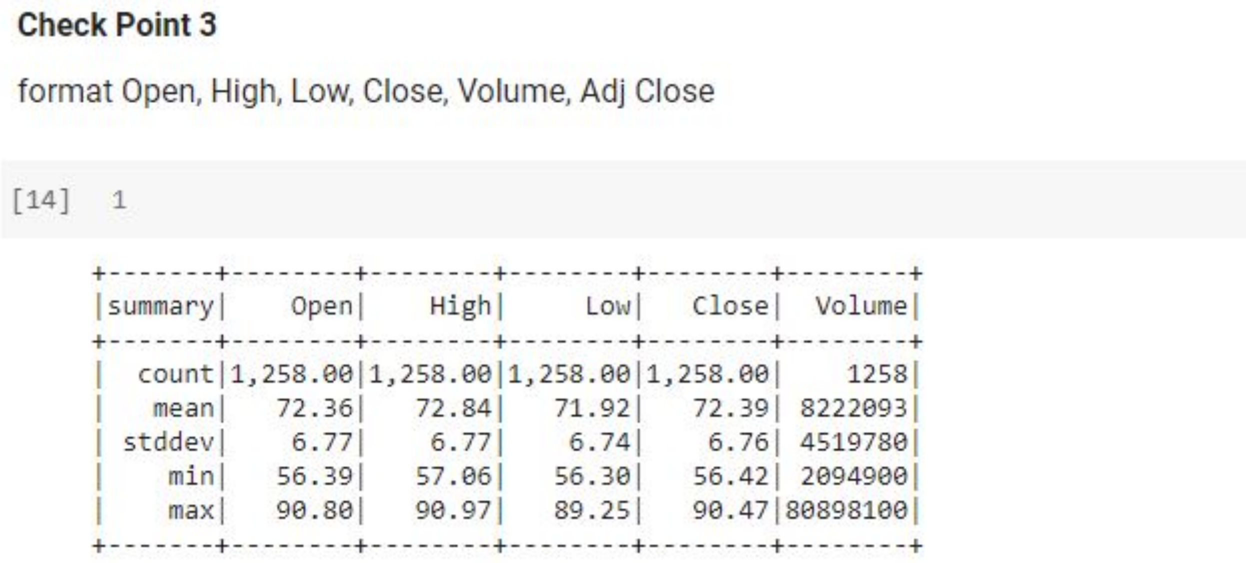
```
# hint
from pyspark.sql.functions import format_number
```

```
result = df.describe()
result.select(result['summary'],
              format_number(result['Open'].cast('float'),2).alias('Open'),
              result['Volume'].cast('int').alias('Volume')
             ).show()
```

```
+-------+--------+--------+
|summary|    Open|  Volume|
+-------+--------+--------+
|  count|1,258.00|    1258|
|   mean|   72.36| 8222093|
| stddev|    6.77| 4519780|
|    min|   56.39| 2094900|
|    max|   90.80|80898100|
+-------+--------+--------+
```

Check-Point-3:

0.5 point

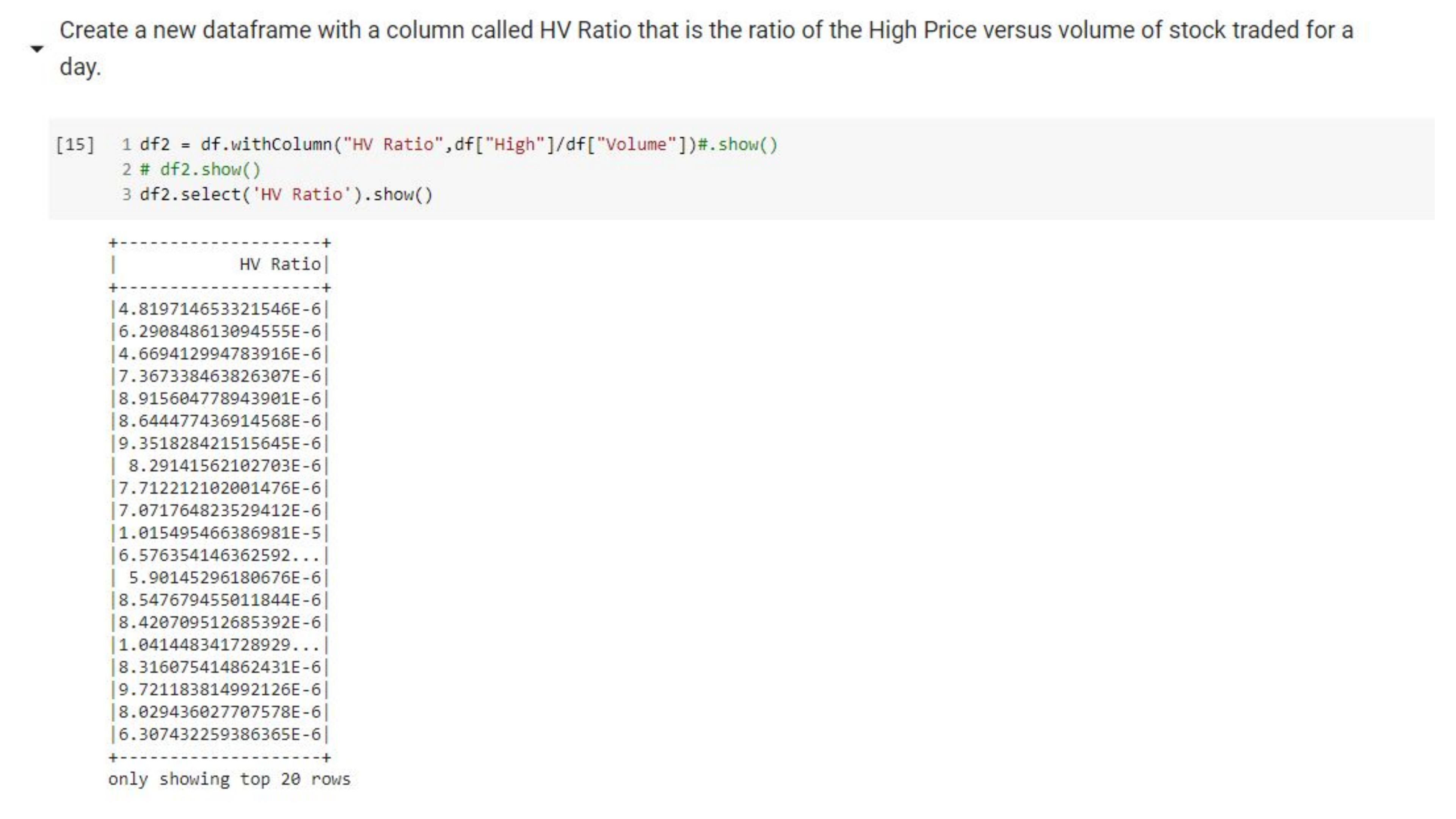


Check Point 3

format Open, High, Low, Close, Volume, Adj Close

[14]

```
+-------+--------+--------+--------+--------+--------+
|summary|    Open|    High|     Low|   Close|  Volume|
+-------+--------+--------+--------+--------+--------+
|  count|1,258.00|1,258.00|1,258.00|1,258.00|    1258|
|   mean|   72.36|   72.84|   71.92|   72.39| 8222093|
| stddev|    6.77|    6.77|    6.74|    6.76| 4519780|
|    min|   56.39|   57.06|   56.30|   56.42| 2094900|
|    max|   90.80|   90.97|   89.25|   90.47|80898100|
+-------+--------+--------+--------+--------+--------+
```

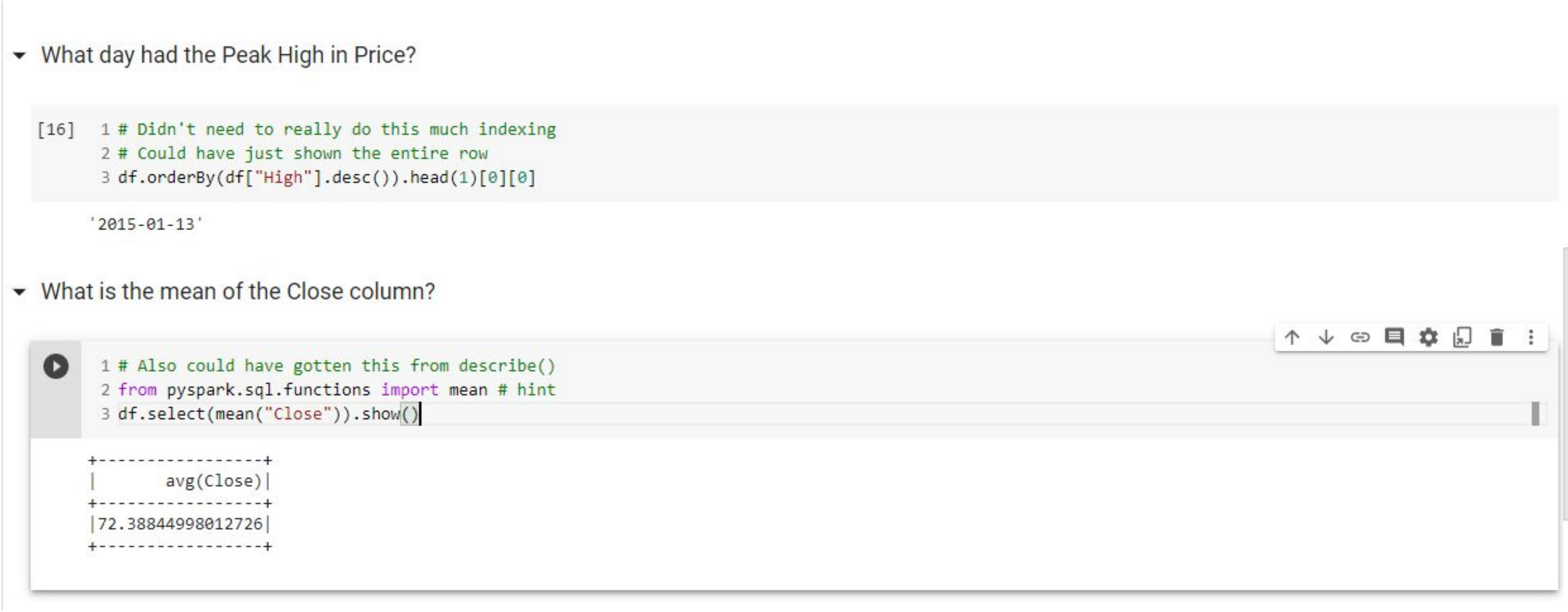


Create a new dataframe with a column called HV Ratio that is the ratio of the High Price versus volume of stock traded for a day.

[15]

```
df2 = df.withColumn("HV Ratio",df["High"]/df["Volume"])#.show()
# df2.show()
df2.select('HV Ratio').show()
```

```
+--------------------+
|            HV Ratio|
+--------------------+
|4.819714653321546E-6|
|6.290848613094555E-6|
|4.669412994783916E-6|
|7.367338463826307E-6|
|8.915604778943901E-6|
|8.644477436914568E-6|
|9.351828421515645E-6|
| 8.29141562102703E-6|
|7.712212102001476E-6|
|7.071764823529412E-6|
|1.015495466386981E-5|
|6.576354146362592...|
| 5.90145296180676E-6|
|8.547679455011844E-6|
|8.420709512685392E-6|
|1.041448341728929...|
|8.316075414862431E-6|
|9.721183814992126E-6|
|8.029436027707578E-6|
|6.307432259386365E-6|
+--------------------+
only showing top 20 rows
```



What day had the Peak High in Price?

```
[16]
# Didn't need to really do this much indexing
# Could have just shown the entire row
df.orderBy(df["High"].desc()).head(1)[0][0]
```

```
'2015-01-13'
```

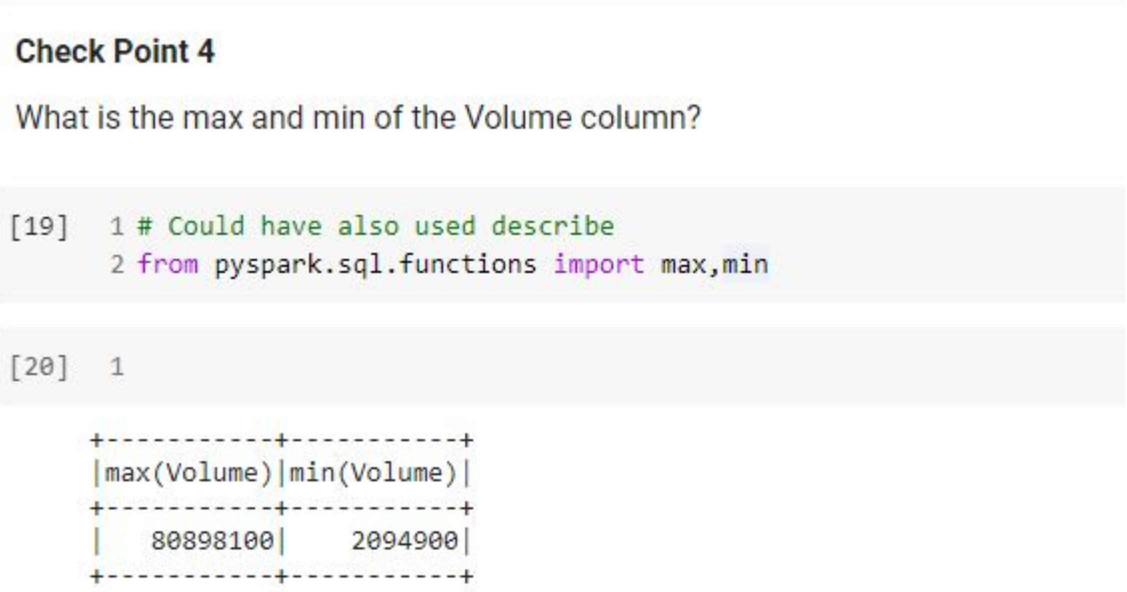
What is the mean of the Close column?

```
# Also could have gotten this from describe()
from pyspark.sql.functions import mean # hint
df.select(mean("Close")).show()
```

```
+-----------------+
|       avg(Close)|
+-----------------+
|72.38844998012726|
+-----------------+
```

Check-Point-4:

1 point



Check Point 4

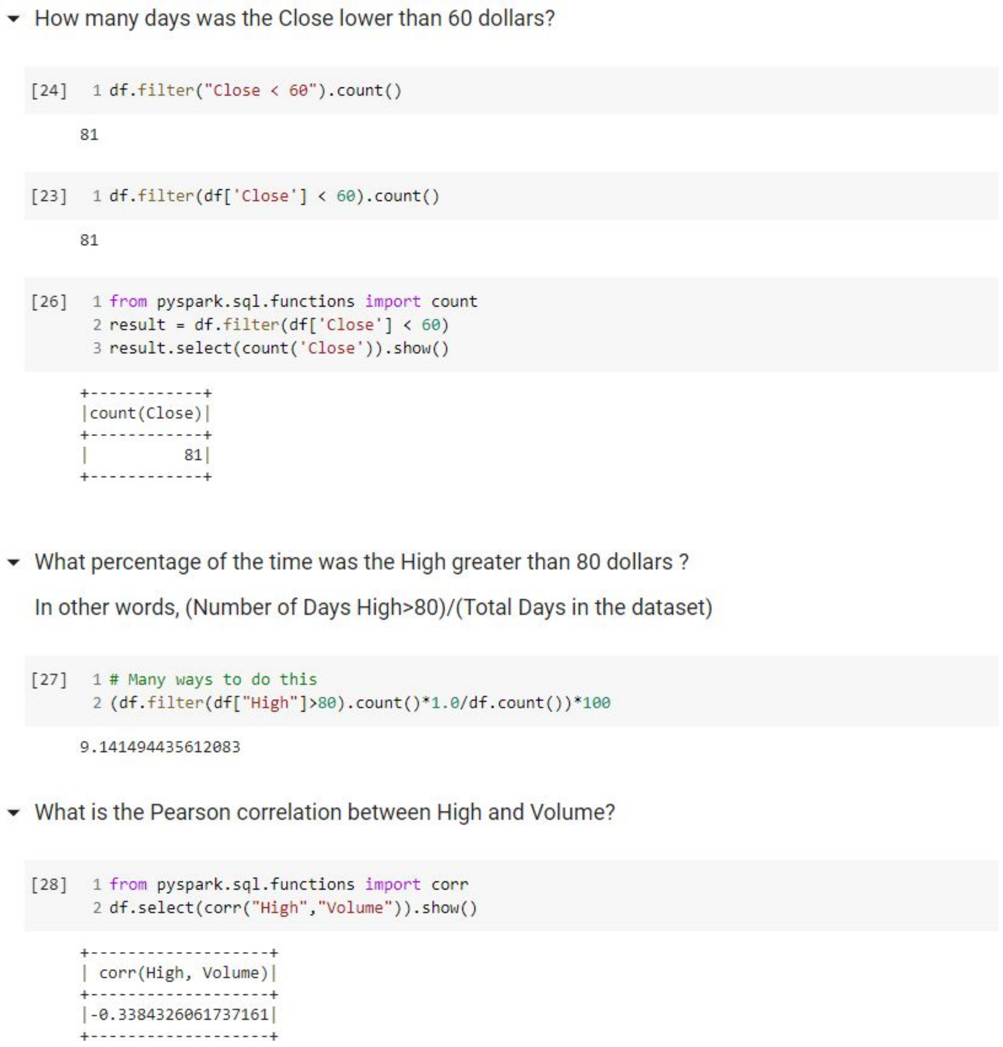
What is the max and min of the Volume column?

[19]

```
# Could have also used describe
from pyspark.sql.functions import max,min
```

[20]

```
+-----------+-----------+
|max(Volume)|min(Volume)|
+-----------+-----------+
|   80898100|    2094900|
+-----------+-----------+
```



How many days was the Close lower than 60 dollars?

[24]

```
df.filter("Close < 60").count()
```

```
81
```

[23]

```
df.filter(df['Close'] < 60).count()
```

```
81
```

[26]

```
from pyspark.sql.functions import count
result = df.filter(df['Close'] < 60)
result.select(count('Close')).show()
```

```
+------------+
|count(Close)|
+------------+
|          81|
+------------+
```

What percentage of the time was the High greater than 80 dollars ?

In other words, (Number of Days High>80)/(Total Days in the dataset)

[27]

```
# Many ways to do this
(df.filter(df["High"]>80).count()*1.0/df.count())*100
```

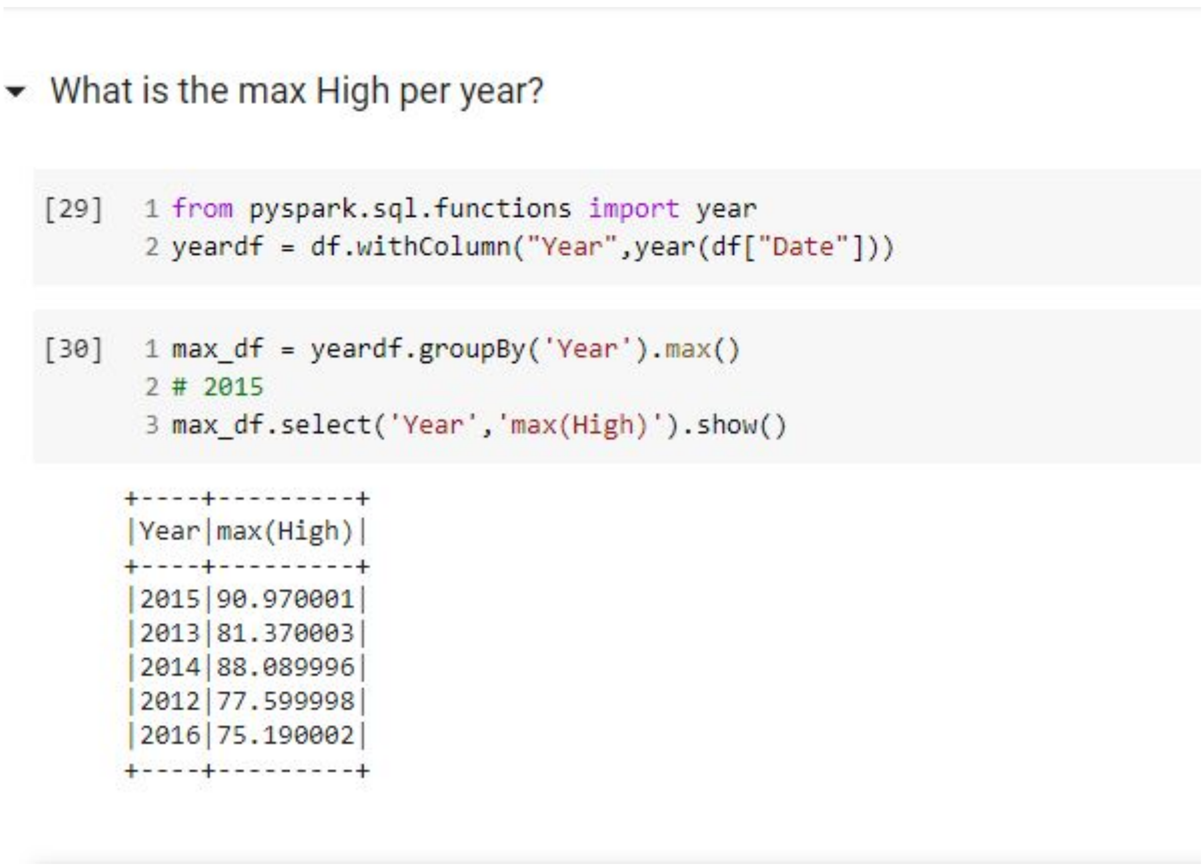
```
9.141494435612083
```

What is the Pearson correlation between High and Volume?

[28]

```
from pyspark.sql.functions import corr
df.select(corr("High","Volume")).show()
```

```
+-------------------+
| corr(High, Volume)|
+-------------------+
|-0.3384326061737161|
+-------------------+
```

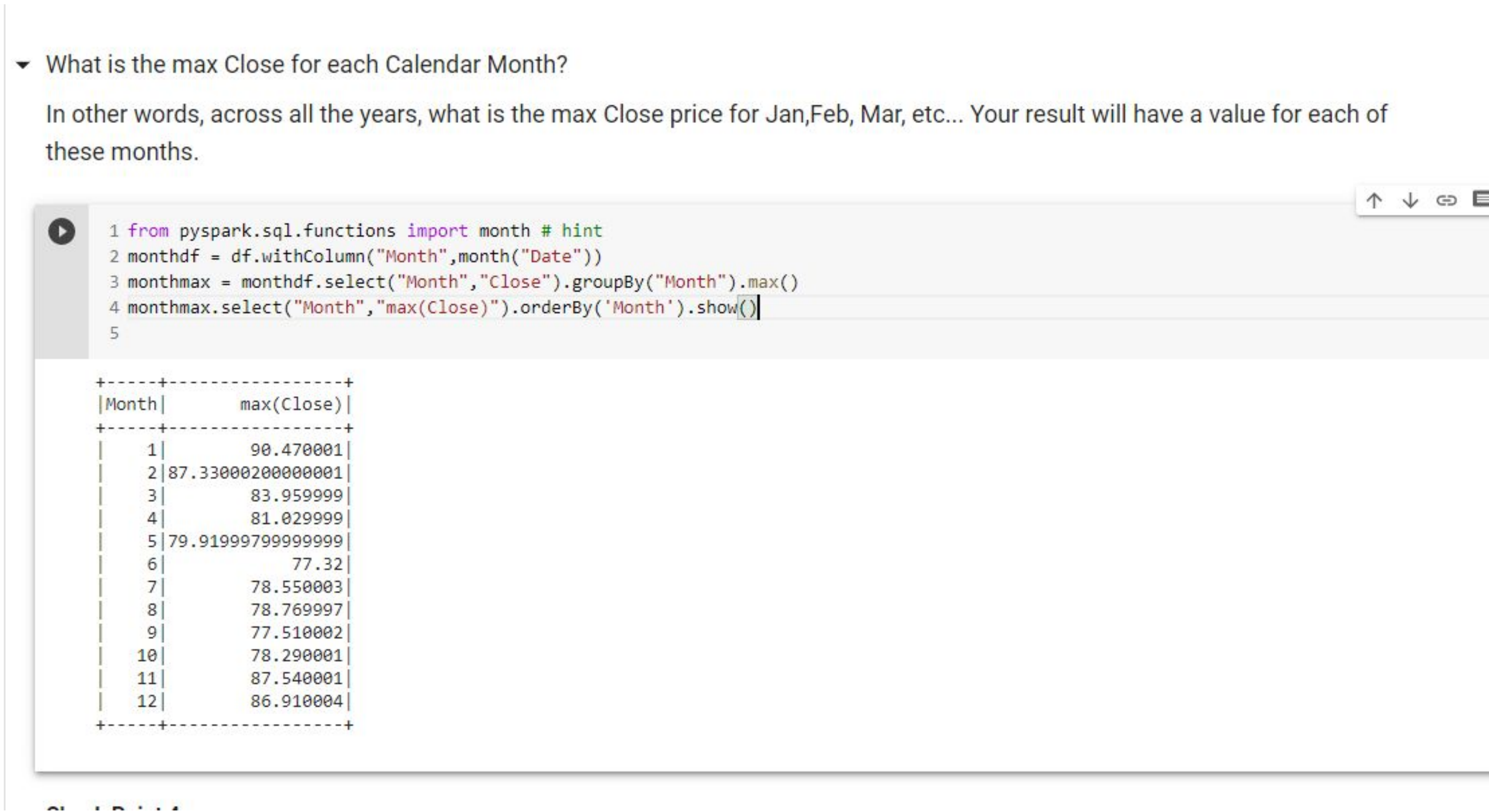


What is the max High per year?

```
[29]
from pyspark.sql.functions import year
yeardf = df.withColumn("Year",year(df["Date"]))
```

```
[30]
max_df = yeardf.groupBy('Year').max()
# 2015
max_df.select('Year','max(High)').show()
```

```
+----+---------+
|Year|max(High)|
+----+---------+
|2015|90.970001|
|2013|81.370003|
|2014|88.089996|
|2012|77.599998|
|2016|75.190002|
+----+---------+
```



What is the max Close for each Calendar Month?

In other words, across all the years, what is the max Close price for Jan,Feb, Mar, etc... Your result will have a value for each of these months.

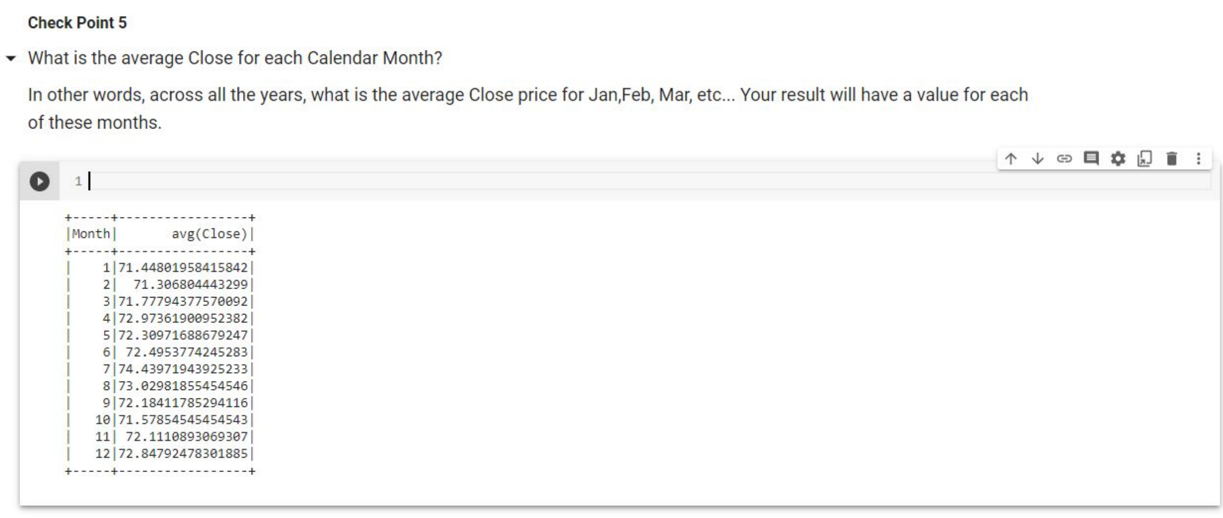
```
from pyspark.sql.functions import month # hint
monthdf = df.withColumn("Month",month("Date"))
monthmax = monthdf.select("Month","Close").groupBy("Month").max()
monthmax.select("Month","max(Close)").orderBy('Month').show()

```

```
+-----+-----------------+
|Month|       max(Close)|
+-----+-----------------+
|    1|        90.470001|
|    2|87.33000200000001|
|    3|        83.959999|
|    4|        81.029999|
|    5|79.91999799999999|
|    6|            77.32|
|    7|        78.550003|
|    8|        78.769997|
|    9|        77.510002|
|   10|        78.290001|
|   11|        87.540001|
|   12|        86.910004|
+-----+-----------------+
```

Check Point 4

Check-Point-5:

1 point



Check Point 5

What is the average Close for each Calendar Month?

In other words, across all the years, what is the average Close price for Jan,Feb, Mar, etc... Your result will have a value for each of these months.

```
+-----+-----------------+
|Month|       avg(Close)|
+-----+-----------------+
|    1|71.44801958415842|
|    2|  71.306804443299|
|    3|71.77794377570092|
|    4|72.97361900952382|
|    5|72.30971688679247|
|    6| 72.4953774245283|
|    7|74.43971943925233|
|    8|73.02981855454546|
|    9|72.18411785294116|
|   10|71.57854545454543|
|   11| 72.1110893069307|
|   12|72.84792478301885|
+-----+-----------------+
```

Great ~ !

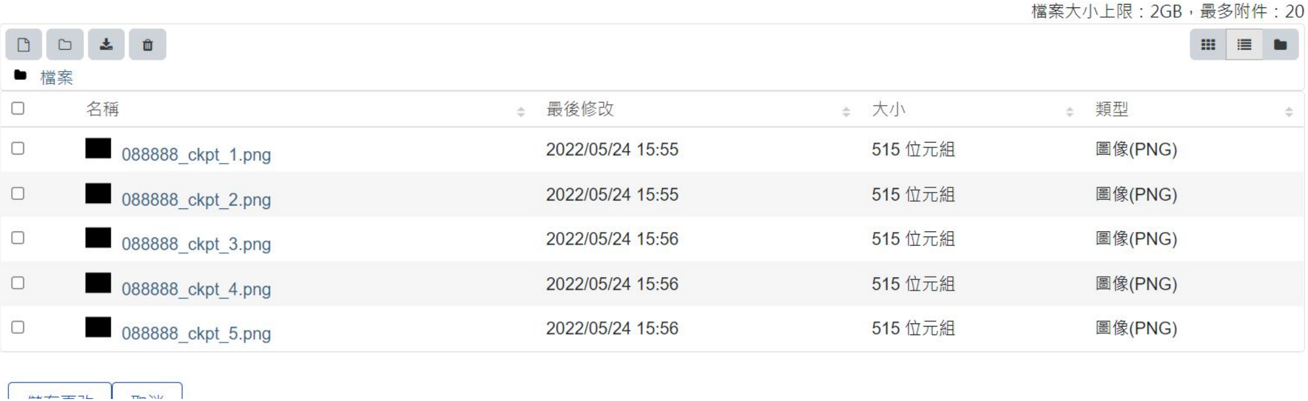
Checkpoint 6

Download your jupyter notebook and upload to new E3 (Lab.3 PySpark) (0.5 points)

Checkpoint 6

Upload screenshots of 5 checkpoint results to E3.

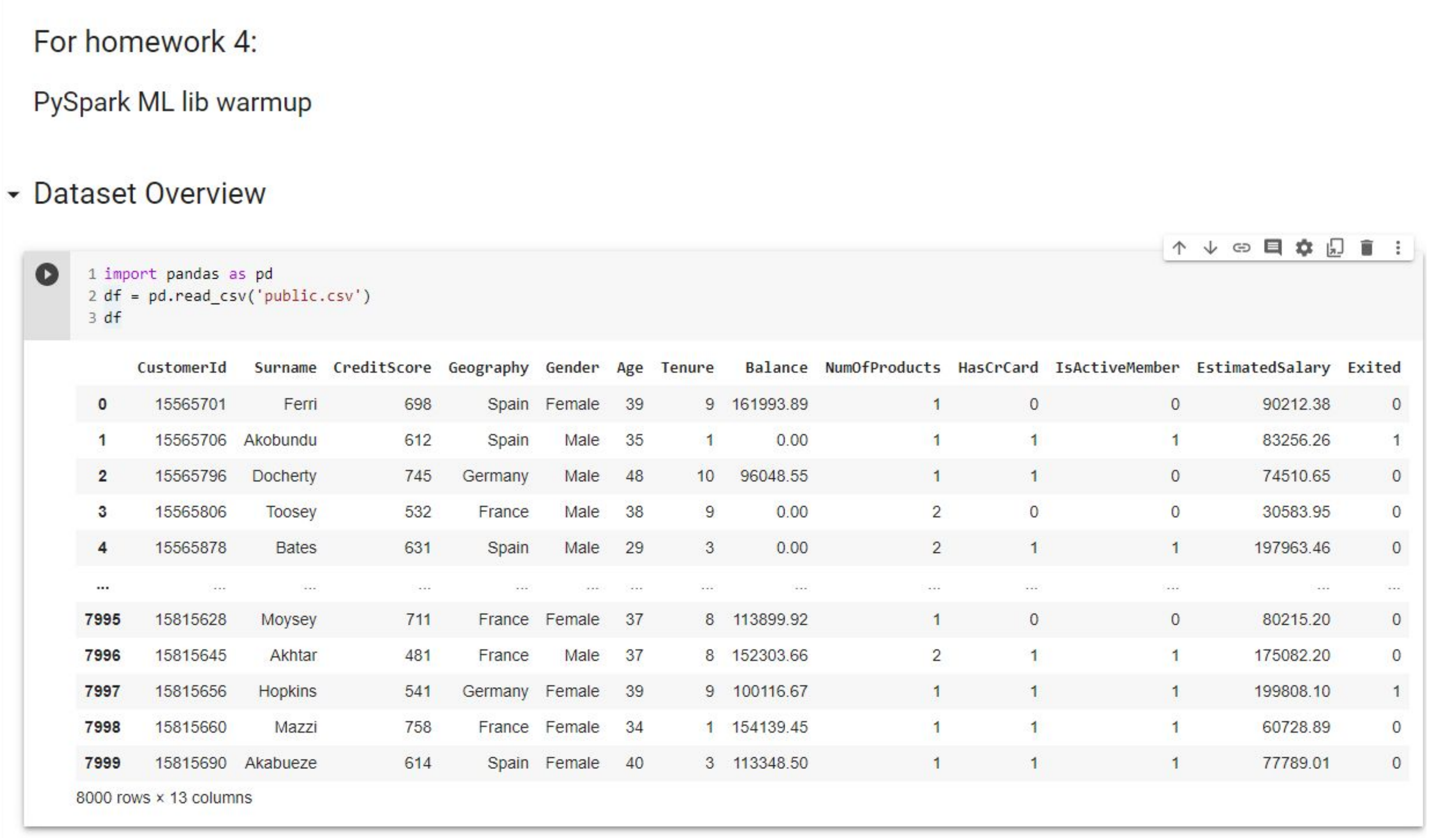
- File format: jpg/png
- Number of files: 5
- File name:
  - [student-ID]_ckpt_[no.].jpg
  - Example: 088888_ckpt_1.jpg
- Note: Do not use any folder or zip file to pack your pictures.
  - Ex.



檔案大小上限：2GB，最多附件：20

檔案

□	名稱	最後修改	大小	類型
□	088888_ckpt_1.png	2022/05/24 15:55	515 位元組	圖像(PNG)
□	088888_ckpt_2.png	2022/05/24 15:55	515 位元組	圖像(PNG)
□	088888_ckpt_3.png	2022/05/24 15:56	515 位元組	圖像(PNG)
□	088888_ckpt_4.png	2022/05/24 15:56	515 位元組	圖像(PNG)
□	088888_ckpt_5.png	2022/05/24 15:56	515 位元組	圖像(PNG)



For homework 4:

PySpark ML lib warmup

Dataset Overview

```python
import pandas as pd
df = pd.read_csv('public.csv')
df
```

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	15565701	Ferri	698	Spain	Female	39	9	161993.89	1	0	0	90212.38	0
1	15565706	Akobundu	612	Spain	Male	35	1	0.00	1	1	1	83256.26	1
2	15565796	Docherty	745	Germany	Male	48	10	96048.55	1	1	0	74510.65	0
3	15565806	Toosey	532	France	Male	38	9	0.00	2	0	0	30583.95	0
4	15565878	Bates	631	Spain	Male	29	3	0.00	2	1	1	197963.46	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
7995	15815628	Moysey	711	France	Female	37	8	113899.92	1	0	0	80215.20	0
7996	15815645	Akhtar	481	France	Male	37	8	152303.66	2	1	1	175082.20	0
7997	15815656	Hopkins	541	Germany	Female	39	9	100116.67	1	1	1	199808.10	1
7998	15815660	Mazzi	758	France	Female	34	1	154139.45	1	1	1	60728.89	0
7999	15815690	Akabueze	614	Spain	Female	40	3	113348.50	1	1	1	77789.01	0

8000 rows × 13 columns