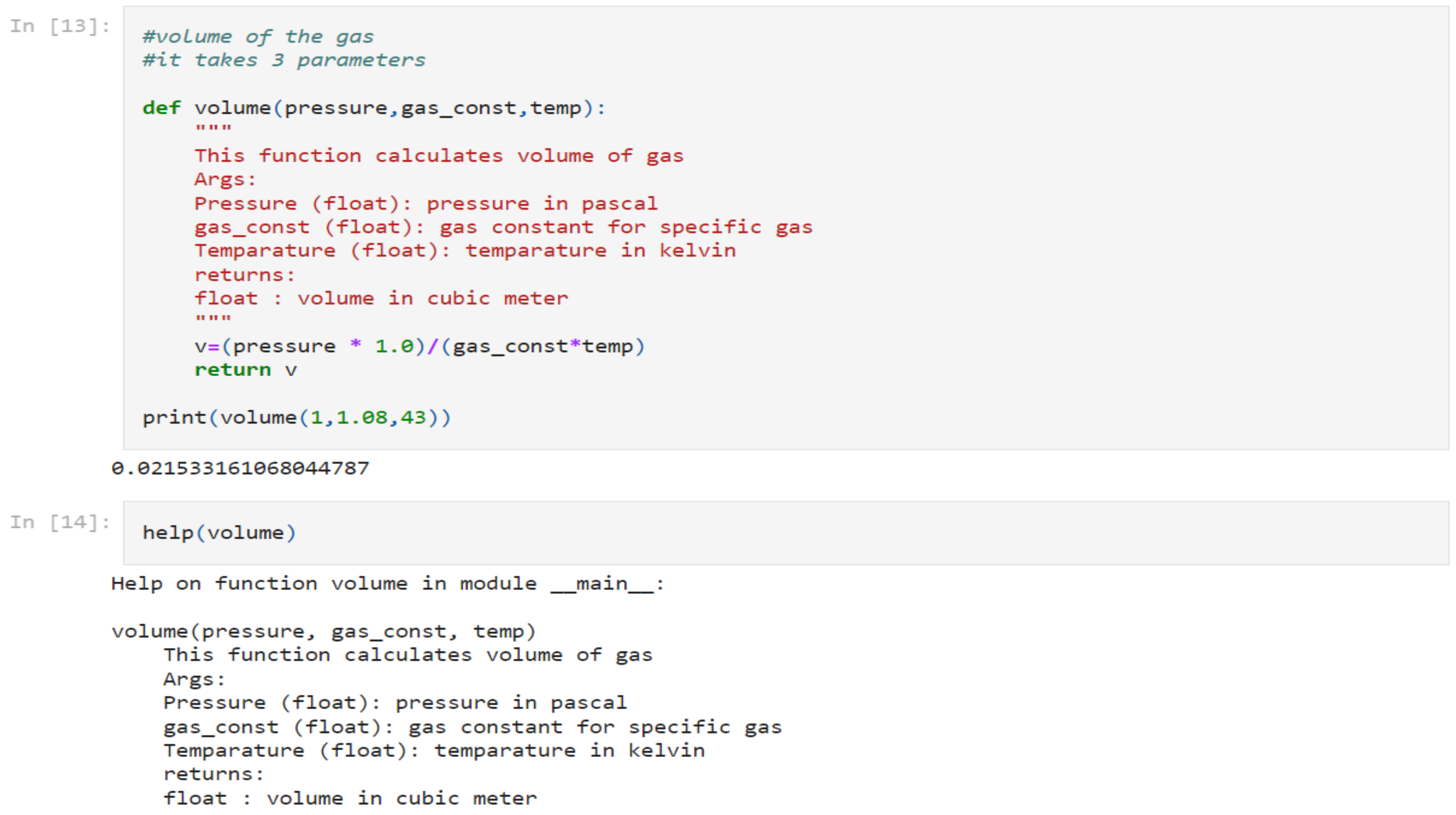
Custom Bootcamp week-4

19-09-2023 to 22-09-2023

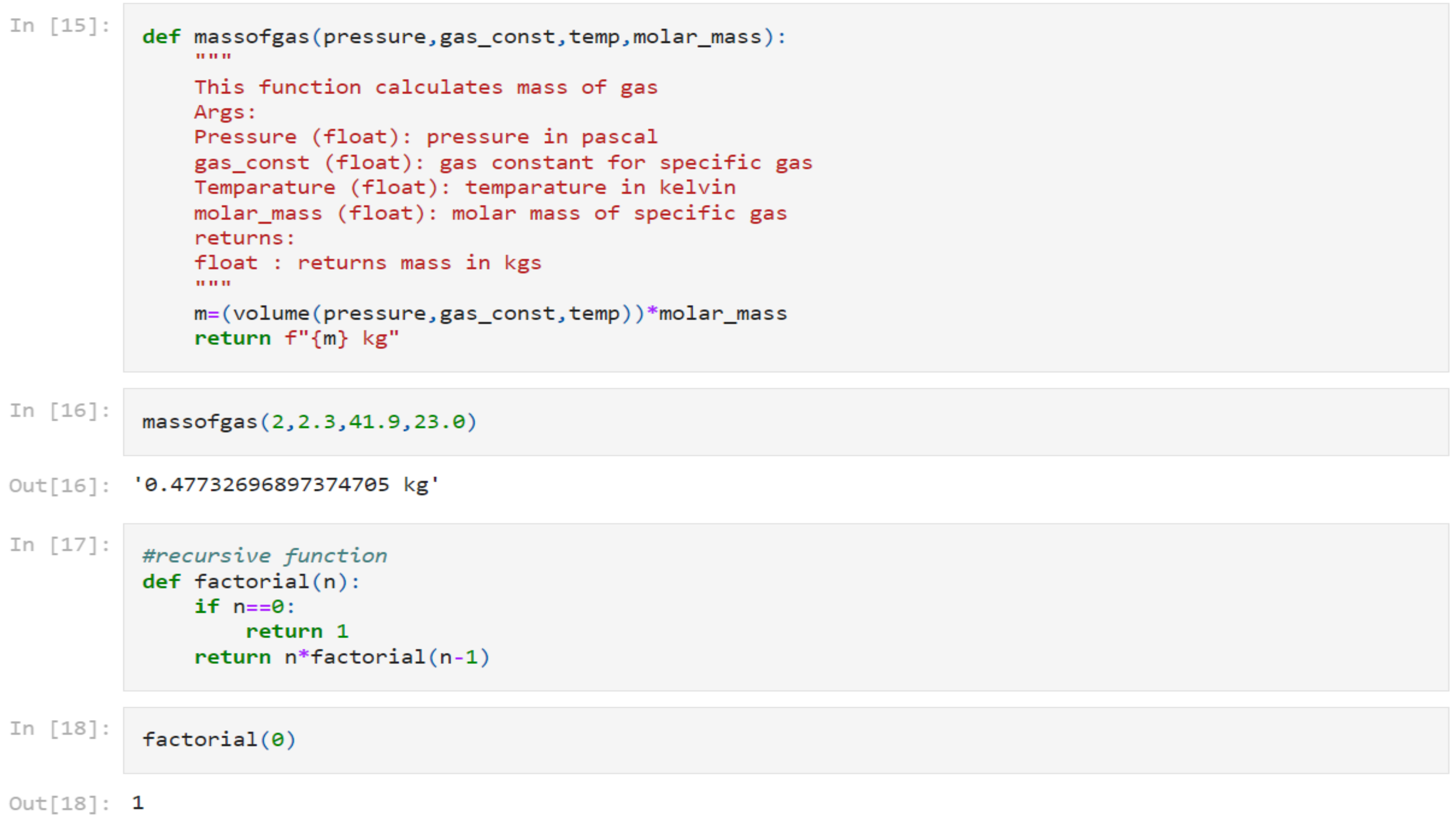
19-09-2023



```
In [13]: #volume of the gas
         #it takes 3 parameters

         def volume(pressure,gas_const,temp):
             """
             This function calculates volume of gas
             Args:
             Pressure (float): pressure in pascal
             gas_const (float): gas constant for specific gas
             Temparature (float): temparature in kelvin
             returns:
             float : volume in cubic meter
             """
             v=(pressure * 1.0)/(gas_const*temp)
             return v

         print(volume(1,1.08,43))

        0.021533161068044787
```

```
In [14]: help(volume)

        Help on function volume in module __main__:

        volume(pressure, gas_const, temp)
            This function calculates volume of gas
            Args:
            Pressure (float): pressure in pascal
            gas_const (float): gas constant for specific gas
            Temparature (float): temparature in kelvin
            returns:
            float : volume in cubic meter
```

19-09-2023



```
In [15]:
def massofgas(pressure,gas_const,temp,molar_mass):
    """
    This function calculates mass of gas
    Args:
    Pressure (float): pressure in pascal
    gas_const (float): gas constant for specific gas
    Temparature (float): temparature in kelvin
    molar_mass (float): molar mass of specific gas
    returns:
    float : returns mass in kgs
    """
    m=(volume(pressure,gas_const,temp))*molar_mass
    return f"{m} kg"
```

```
In [16]:
massofgas(2,2.3,41.9,23.0)
```

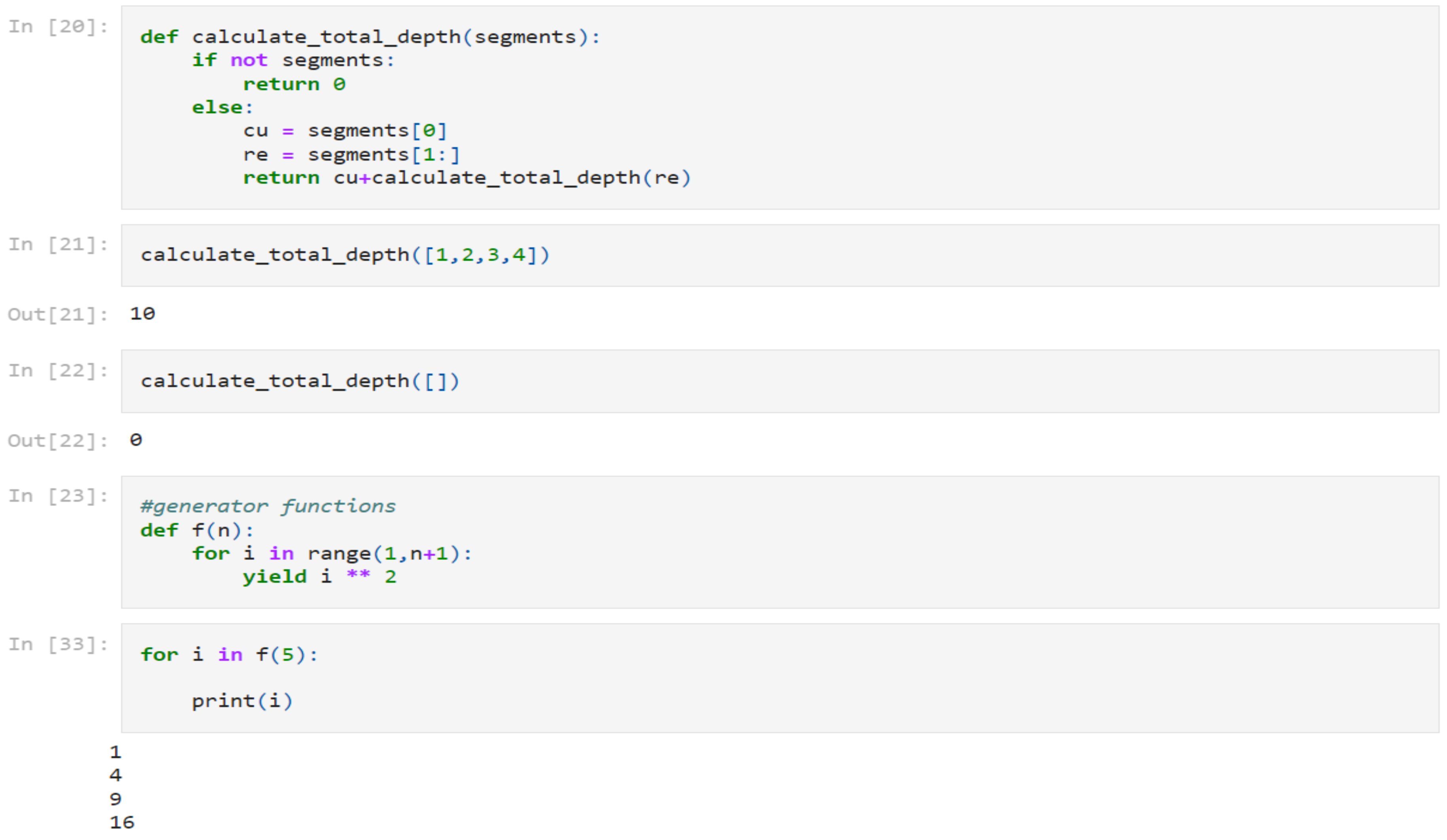
```
Out[16]: '0.47732696897374705 kg'
```

```
In [17]:
#recursive function
def factorial(n):
    if n==0:
        return 1
    return n*factorial(n-1)
```

```
In [18]:
factorial(0)
```

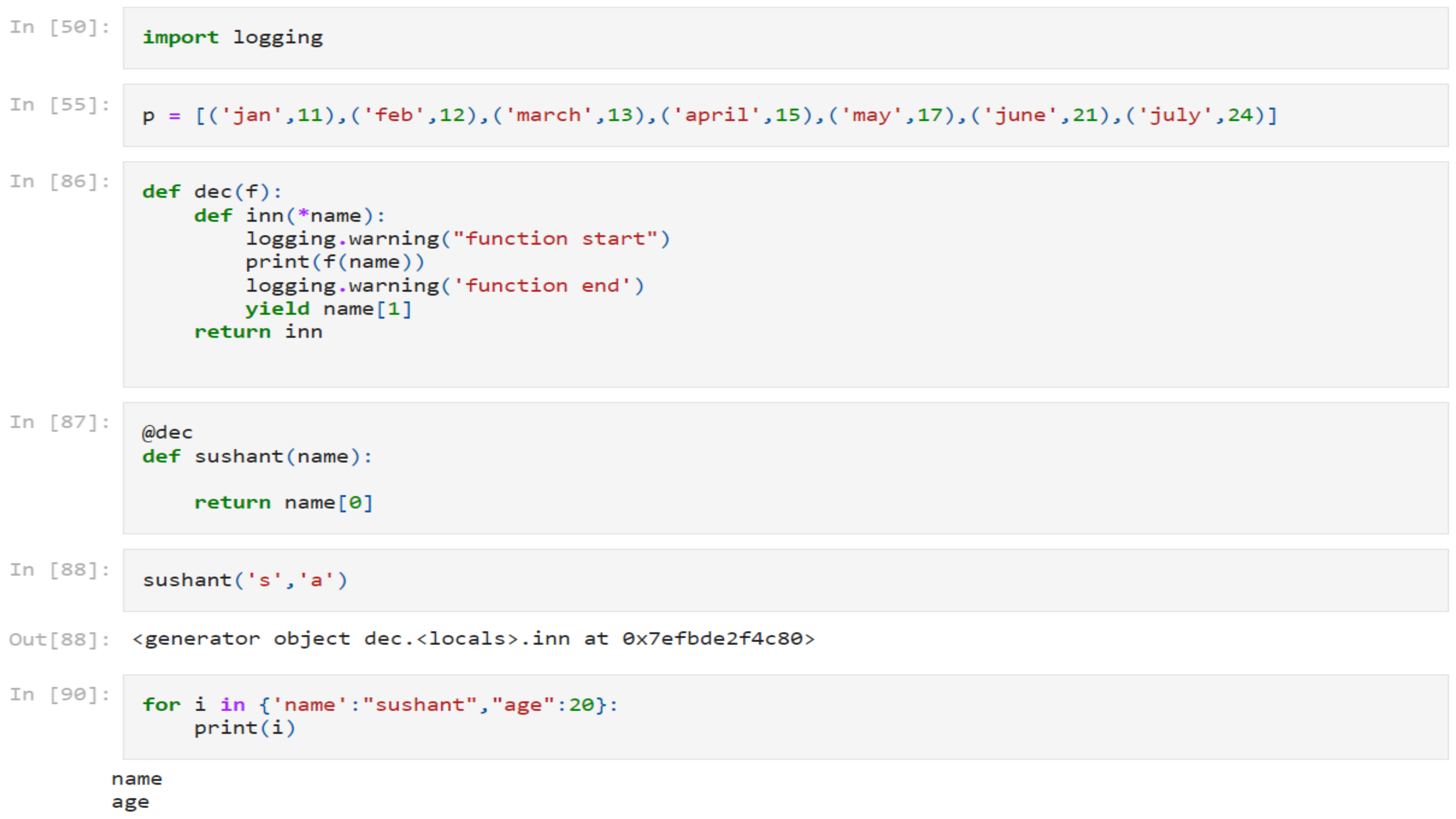
```
Out[18]: 1
```

19-09-2023



```
In [20]: def calculate_total_depth(segments):
             if not segments:
                 return 0
             else:
                 cu = segments[0]
                 re = segments[1:]
                 return cu+calculate_total_depth(re)

In [21]: calculate_total_depth([1,2,3,4])

Out[21]: 10

In [22]: calculate_total_depth([])

Out[22]: 0

In [23]: #generator functions
         def f(n):
             for i in range(1,n+1):
                 yield i ** 2

In [33]: for i in f(5):

             print(i)

        1
        4
        9
        16
```

19-09-2023



```
In [50]:  import logging
```

```
In [55]:  p = [('jan',11),('feb',12),('march',13),('april',15),('may',17),('june',21),('july',24)]
```

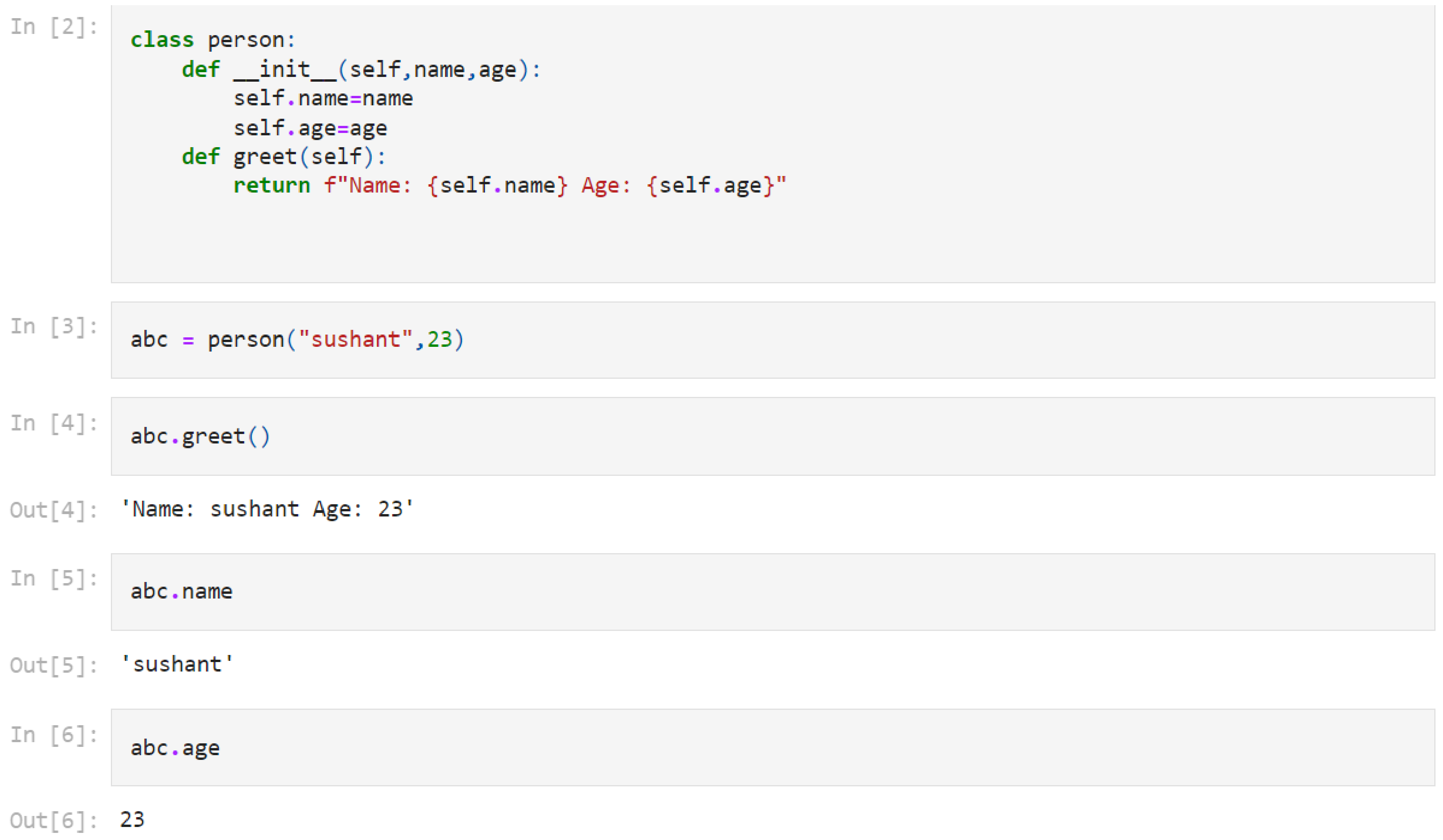
```
In [86]:  def dec(f):
              def inn(*name):
                  logging.warning("function start")
                  print(f(name))
                  logging.warning('function end')
                  yield name[1]
              return inn
```

```
In [87]:  @dec
          def sushant(name):

              return name[0]
```

```
In [88]:  sushant('s','a')

Out[88]:  <generator object dec.<locals>.inn at 0x7efbde2f4c80>
```

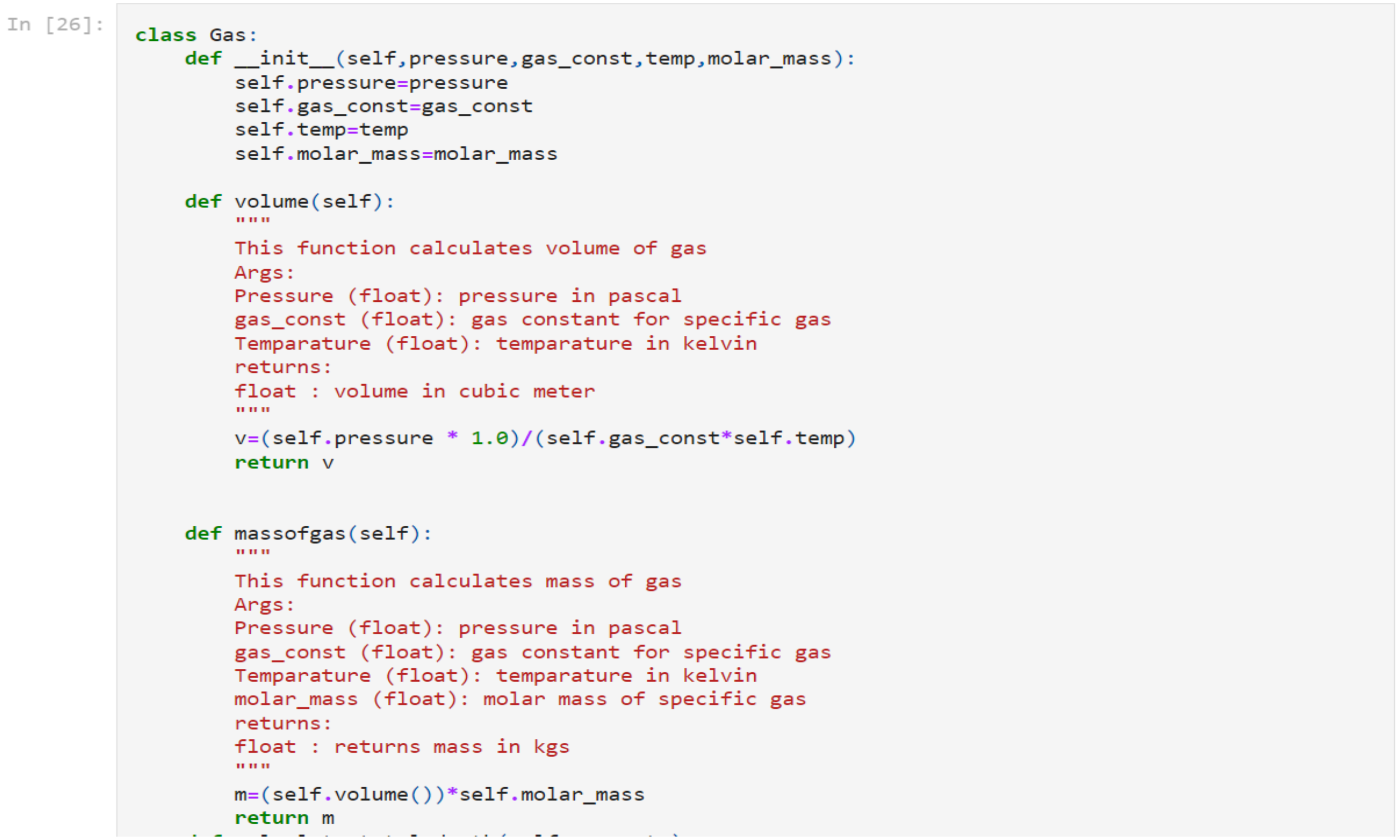
```
In [90]:  for i in {'name':"sushant","age":20}:
              print(i)

          name
          age
```

19-09-2023



```
In [2]: class person:
            def __init__(self,name,age):
                self.name=name
                self.age=age
            def greet(self):
                return f"Name: {self.name} Age: {self.age}"

In [3]: abc = person("sushant",23)

In [4]: abc.greet()

Out[4]: 'Name: sushant Age: 23'

In [5]: abc.name

Out[5]: 'sushant'

In [6]: abc.age

Out[6]: 23
```

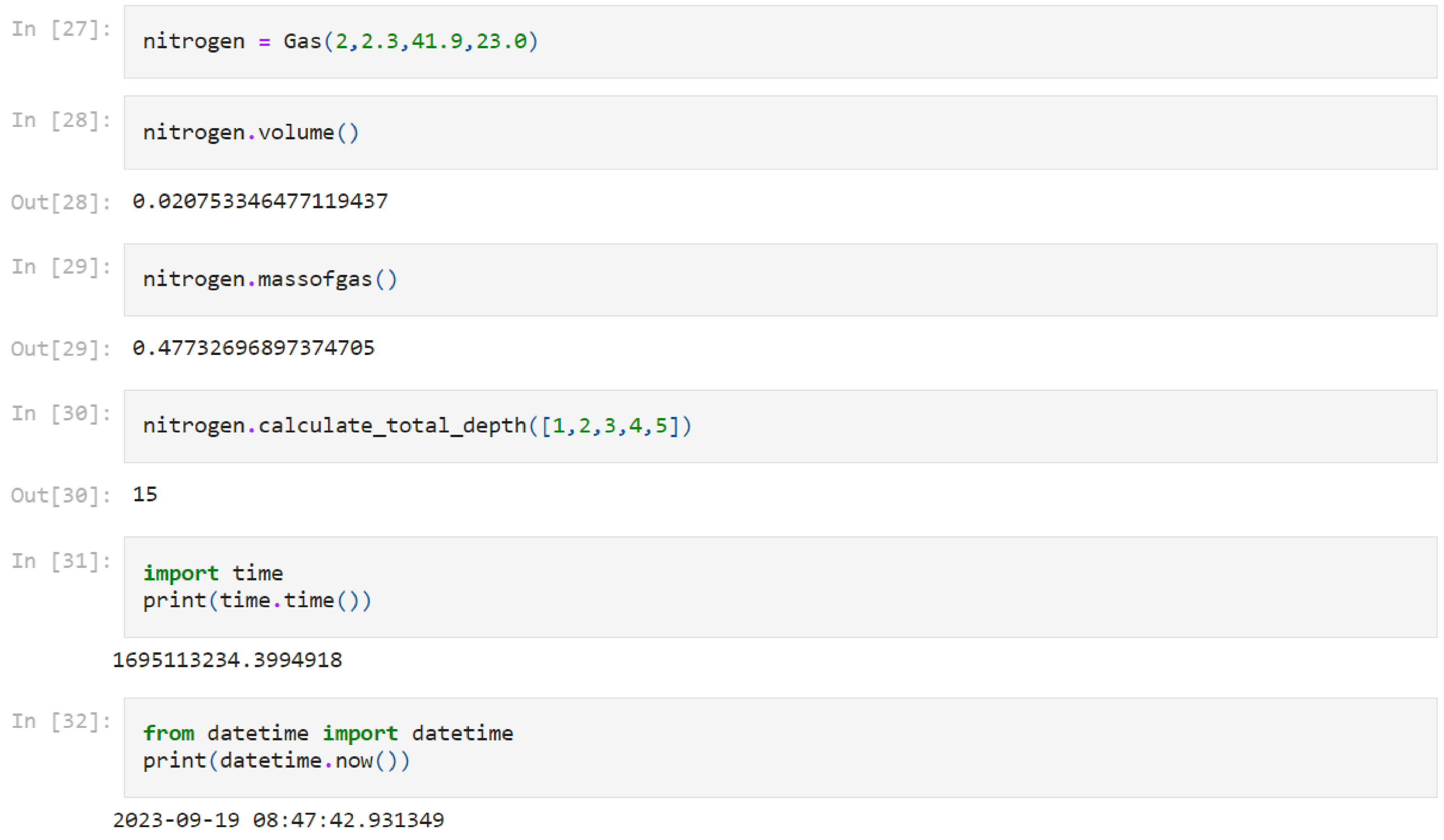
19-09-2023



In [26]:

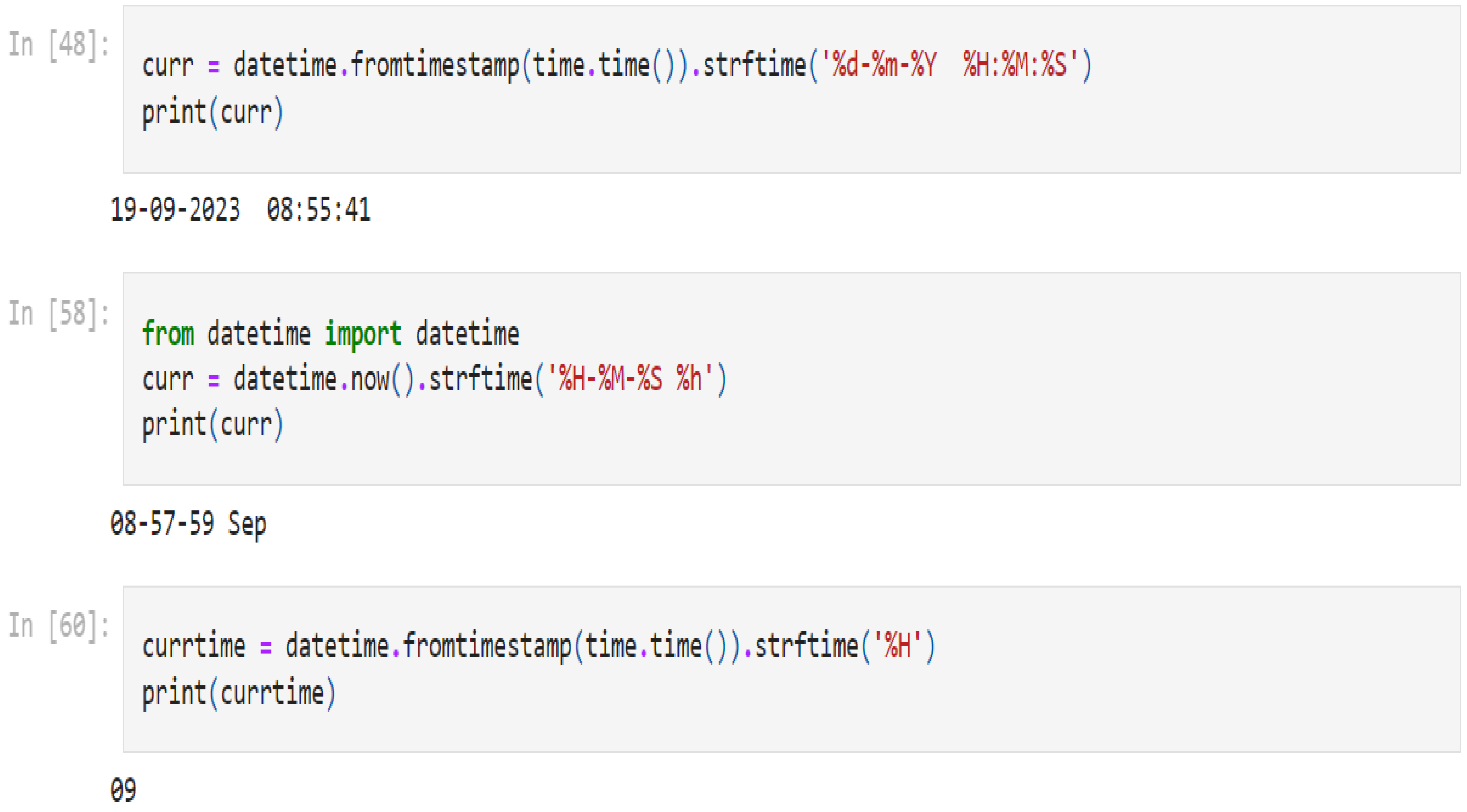
```
class Gas:
    def __init__(self,pressure,gas_const,temp,molar_mass):
        self.pressure=pressure
        self.gas_const=gas_const
        self.temp=temp
        self.molar_mass=molar_mass

    def volume(self):
        """
        This function calculates volume of gas
        Args:
        Pressure (float): pressure in pascal
        gas_const (float): gas constant for specific gas
        Temparature (float): temparature in kelvin
        returns:
        float : volume in cubic meter
        """
        v=(self.pressure * 1.0)/(self.gas_const*self.temp)
        return v


    def massofgas(self):
        """
        This function calculates mass of gas
        Args:
        Pressure (float): pressure in pascal
        gas_const (float): gas constant for specific gas
        Temparature (float): temparature in kelvin
        molar_mass (float): molar mass of specific gas
        returns:
        float : returns mass in kgs
        """
        m=(self.volume())*self.molar_mass
        return m
```

19-09-2023



```
In [27]: nitrogen = Gas(2,2.3,41.9,23.0)

In [28]: nitrogen.volume()

Out[28]: 0.020753346477119437

In [29]: nitrogen.massofgas()

Out[29]: 0.47732696897374705

In [30]: nitrogen.calculate_total_depth([1,2,3,4,5])

Out[30]: 15

In [31]: import time
         print(time.time())

        1695113234.3994918

In [32]: from datetime import datetime
         print(datetime.now())

        2023-09-19 08:47:42.931349
```

19-09-2023



In [48]:
```python
curr = datetime.fromtimestamp(time.time()).strftime('%d-%m-%Y  %H:%M:%S')
print(curr)
```

```
19-09-2023  08:55:41
```

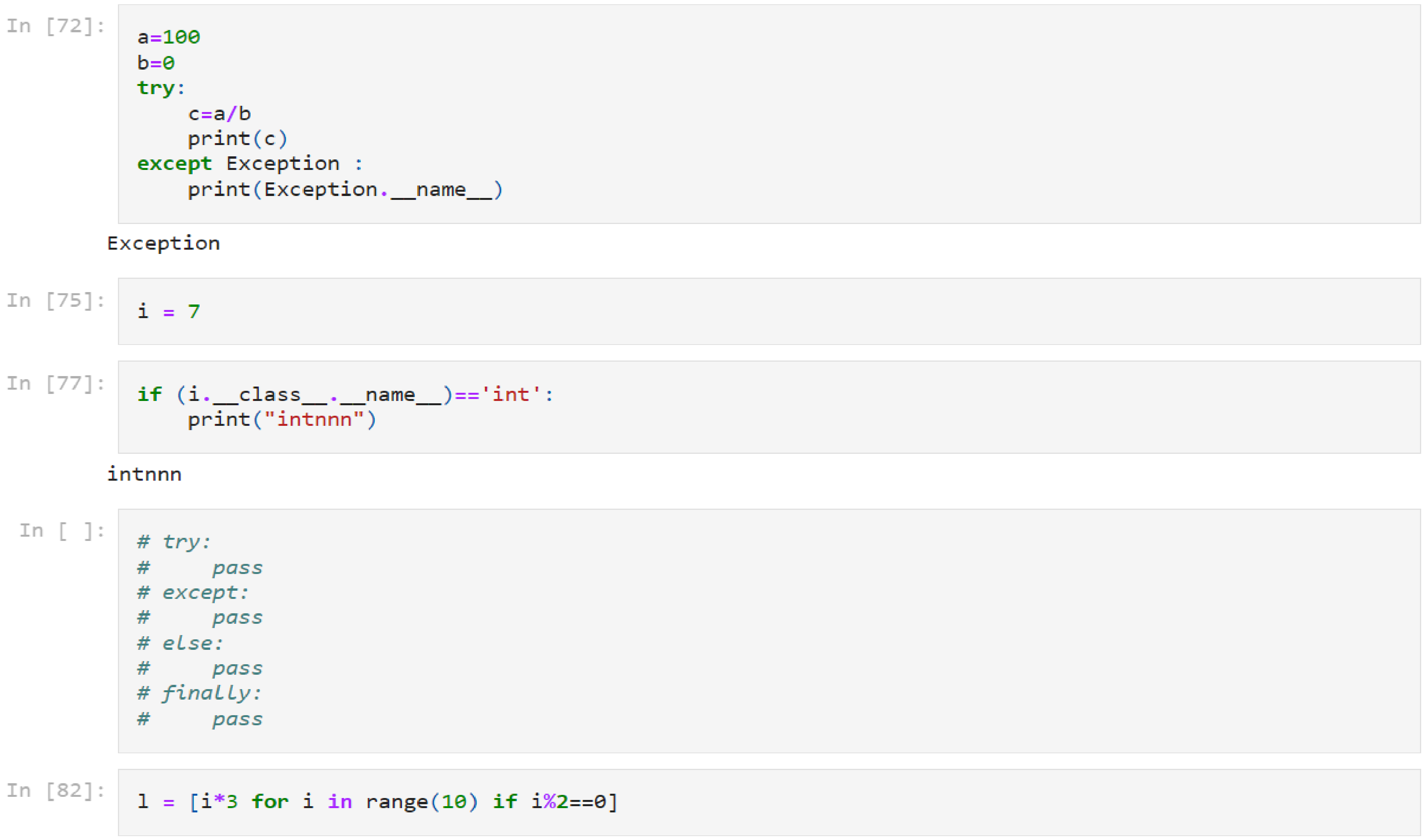
In [58]:
```python
from datetime import datetime
curr = datetime.now().strftime('%H-%M-%S %h')
print(curr)
```

```
08-57-59 Sep
```

In [60]:
```python
currtime = datetime.fromtimestamp(time.time()).strftime('%H')
print(currtime)
```

```
09
```

19-09-2023



```
In [72]:
a=100
b=0
try:
    c=a/b
    print(c)
except Exception :
    print(Exception.__name__)
```

```
Exception
```

```
In [75]:
i = 7
```

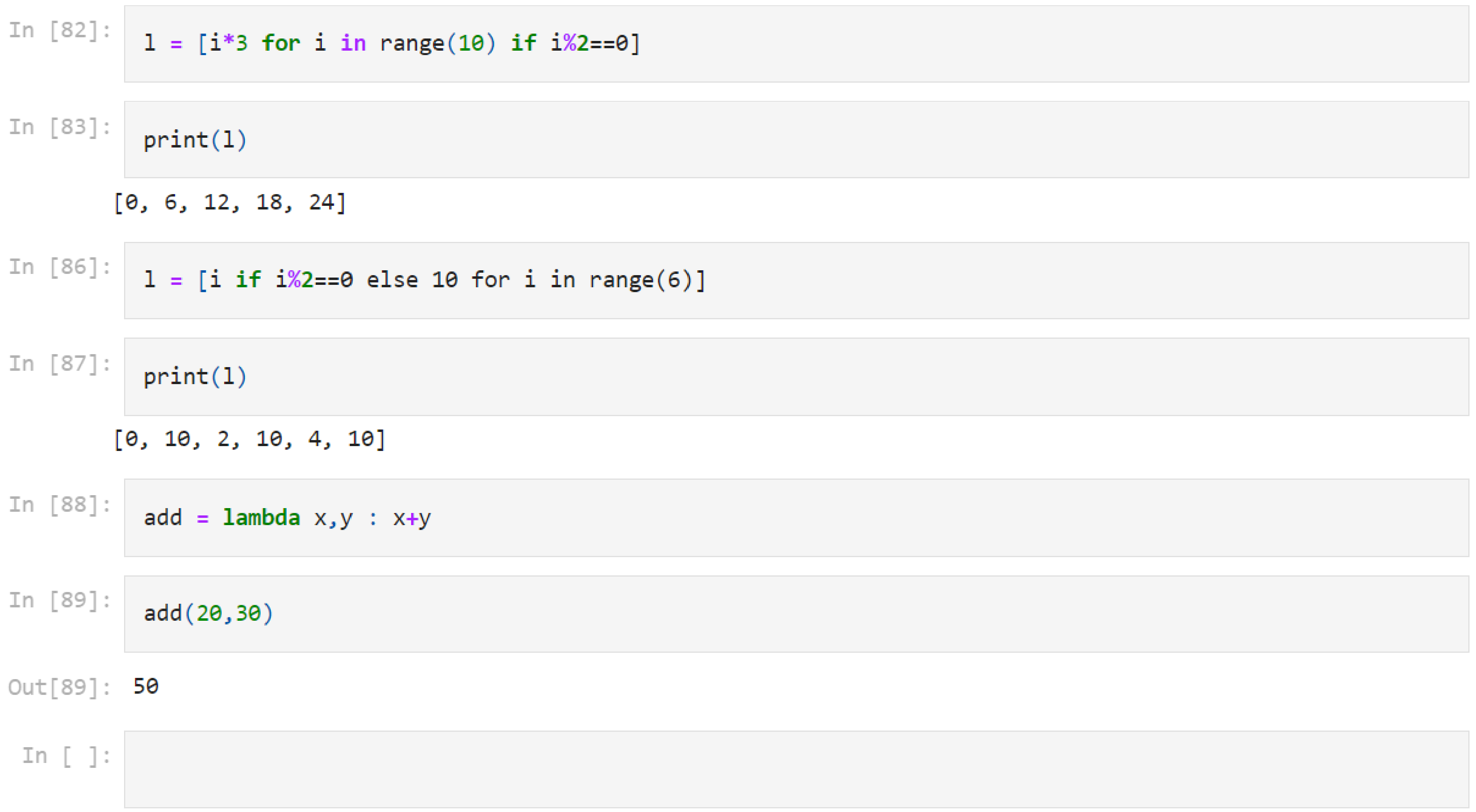
```
In [77]:
if (i.__class__.__name__)=='int':
    print("intnnn")
```

```
intnnn
```

```
In [ ]:
# try:
#     pass
# except:
#     pass
# else:
#     pass
# finally:
#     pass
```

```
In [82]:
l = [i*3 for i in range(10) if i%2==0]
```

19-09-2023

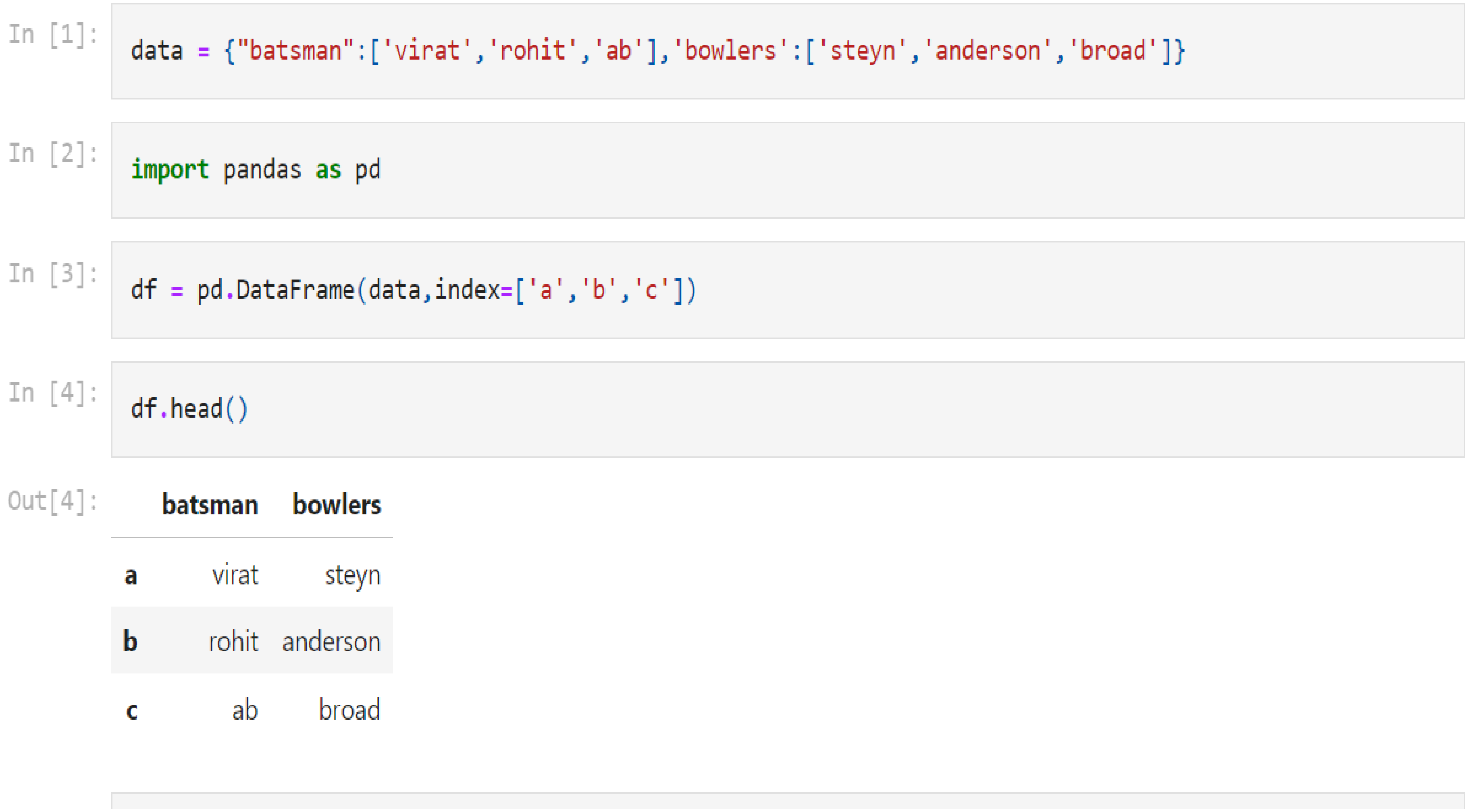


```
In [82]: l = [i*3 for i in range(10) if i%2==0]

In [83]: print(l)

         [0, 6, 12, 18, 24]

In [86]: l = [i if i%2==0 else 10 for i in range(6)]

In [87]: print(l)

         [0, 10, 2, 10, 4, 10]

In [88]: add = lambda x,y : x+y

In [89]: add(20,30)

Out[89]: 50

In [ ]:
```

20-09-2023

- Pandas is python library used to analyze the data.
- Series is one dimensional data structure used to store the data.
- Series data structure contains the index or label associated with it.
- Pandas is collection of series data structure.
- Pandas provides lot of functions to analyze the data.
- Along with this we can plot some graphs using the pandas.

20-09-2023



```
In [1]: data = {"batsman":['virat','rohit','ab'],'bowlers':['steyn','anderson','broad']}
```

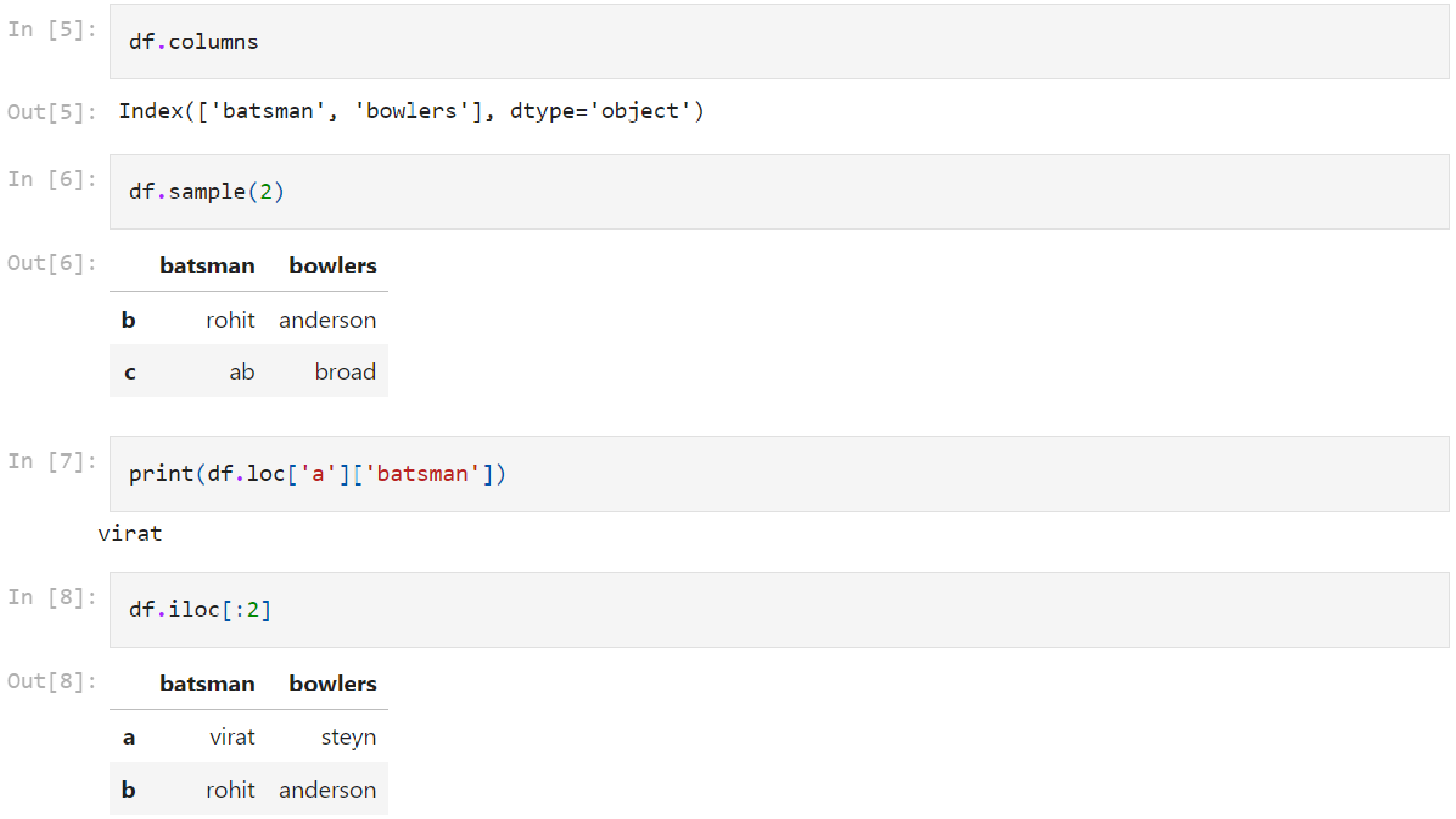
```
In [2]: import pandas as pd
```

```
In [3]: df = pd.DataFrame(data,index=['a','b','c'])
```

```
In [4]: df.head()
```

```
Out[4]:
     batsman   bowlers
  a    virat     steyn
  b    rohit  anderson
  c       ab     broad
```

20-09-2023



```
In [5]: df.columns
```

```
Out[5]: Index(['batsman', 'bowlers'], dtype='object')
```

```
In [6]: df.sample(2)
```

Out[6]:

	batsman	bowlers
b	rohit	anderson
c	ab	broad

```
In [7]: print(df.loc['a']['batsman'])
```

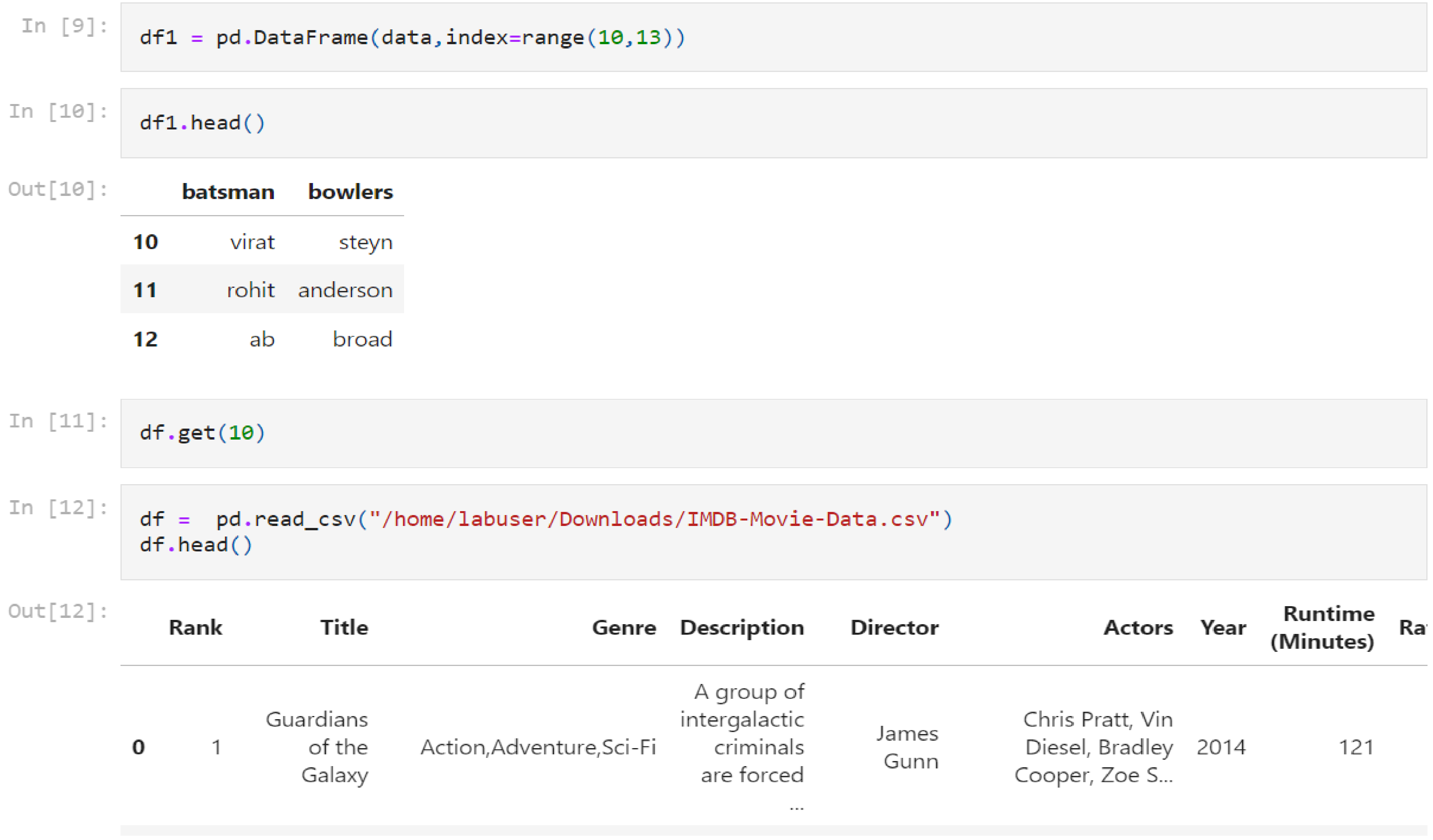
```
virat
```

```
In [8]: df.iloc[:2]
```

Out[8]:

	batsman	bowlers
a	virat	steyn
b	rohit	anderson

20-09-2023



```
In [9]:  df1 = pd.DataFrame(data,index=range(10,13))
```

```
In [10]: df1.head()
```

Out[10]:

	batsman	bowlers
10	virat	steyn
11	rohit	anderson
12	ab	broad

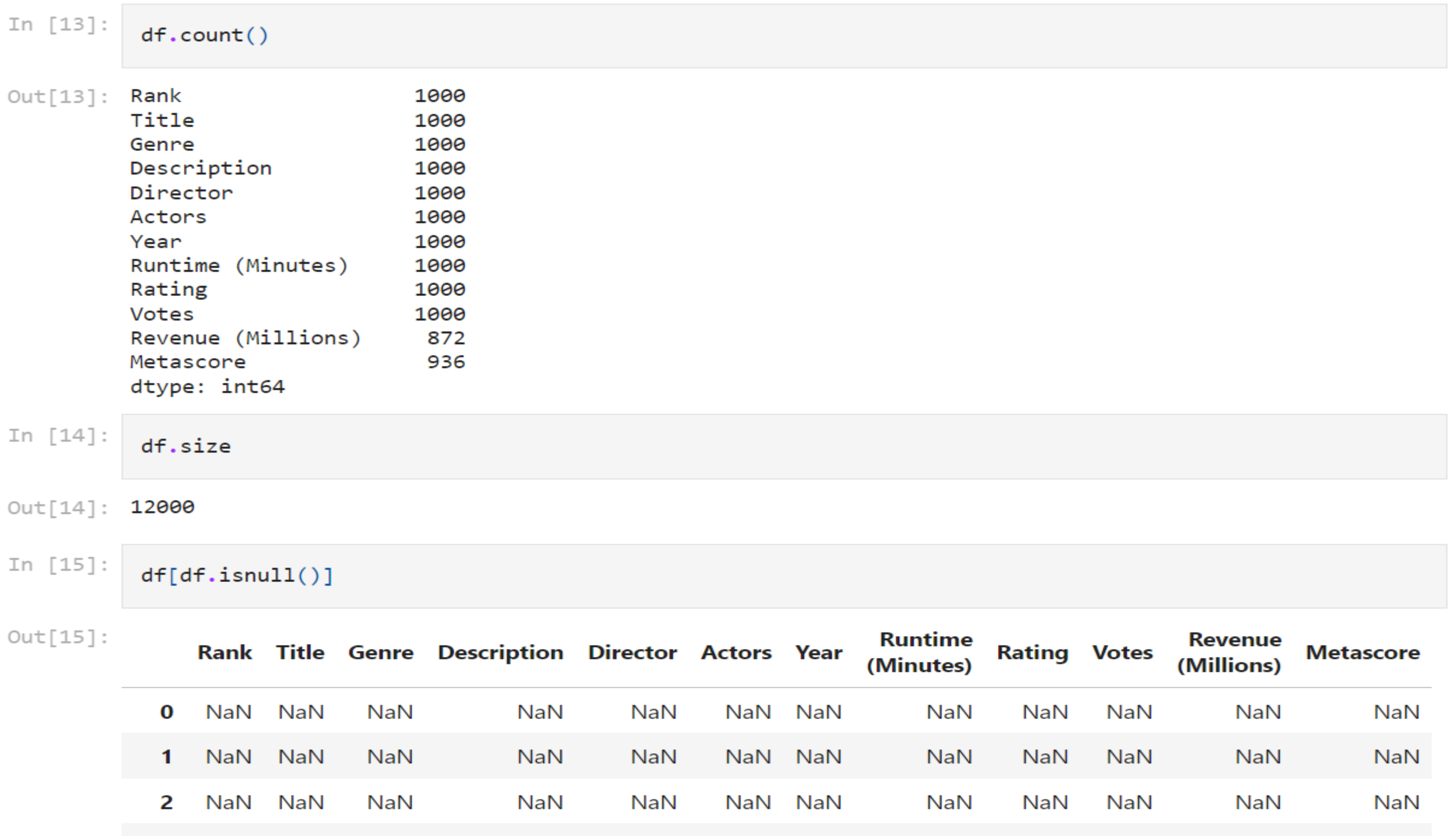
```
In [11]: df.get(10)
```

```
In [12]: df =  pd.read_csv("/home/labuser/Downloads/IMDB-Movie-Data.csv")
         df.head()
```

Out[12]:

	Rank	Title	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Ra
0	1	Guardians of the Galaxy	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2014	121	

20-09-2023



```
In [13]: df.count()
```

```
Out[13]: Rank                  1000
         Title                 1000
         Genre                 1000
         Description           1000
         Director              1000
         Actors                1000
         Year                  1000
         Runtime (Minutes)     1000
         Rating                1000
         Votes                 1000
         Revenue (Millions)     872
         Metascore              936
         dtype: int64
```

```
In [14]: df.size
```

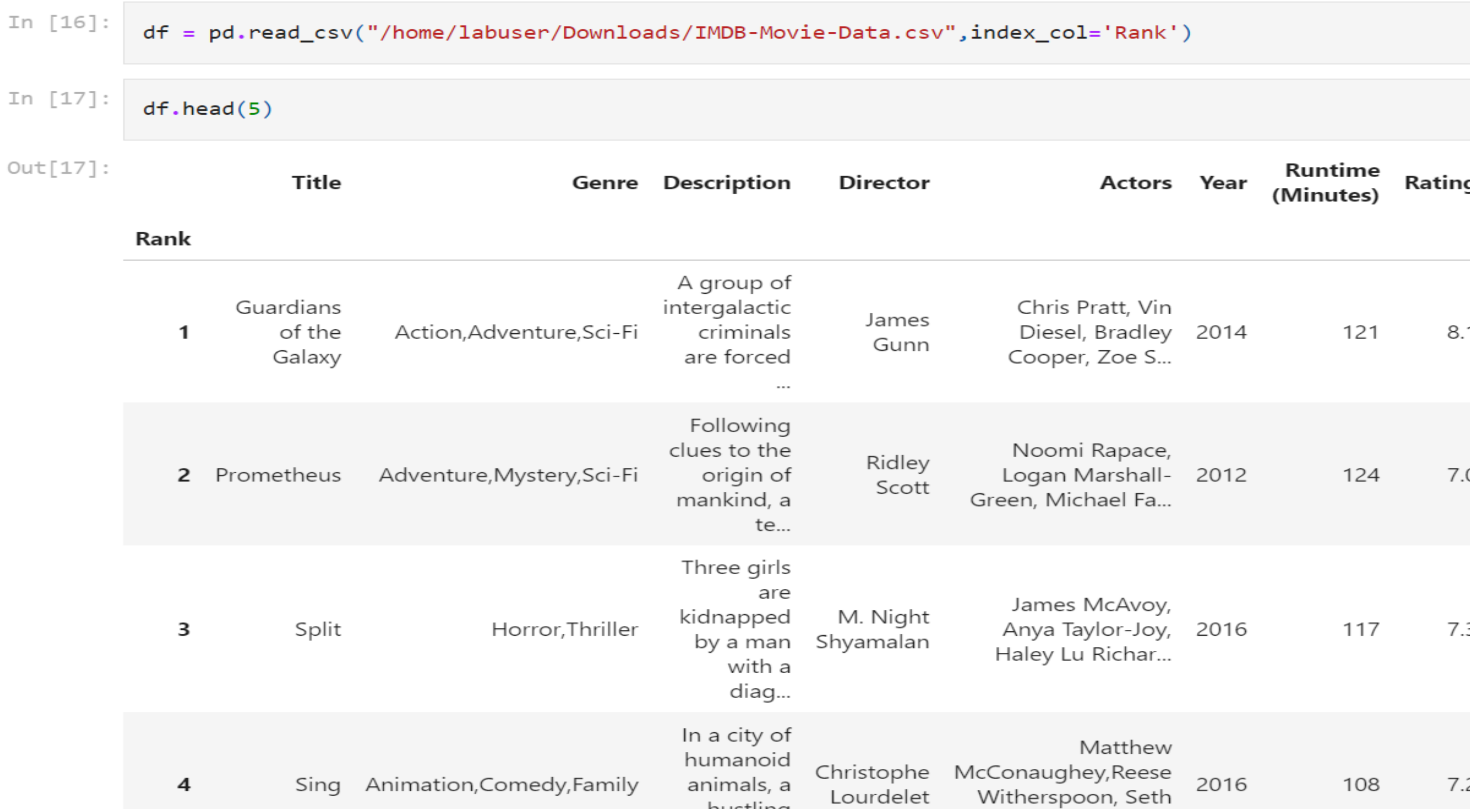
```
Out[14]: 12000
```

```
In [15]: df[df.isnull()]
```

Out[15]:

	Rank	Title	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions)	Metascore
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

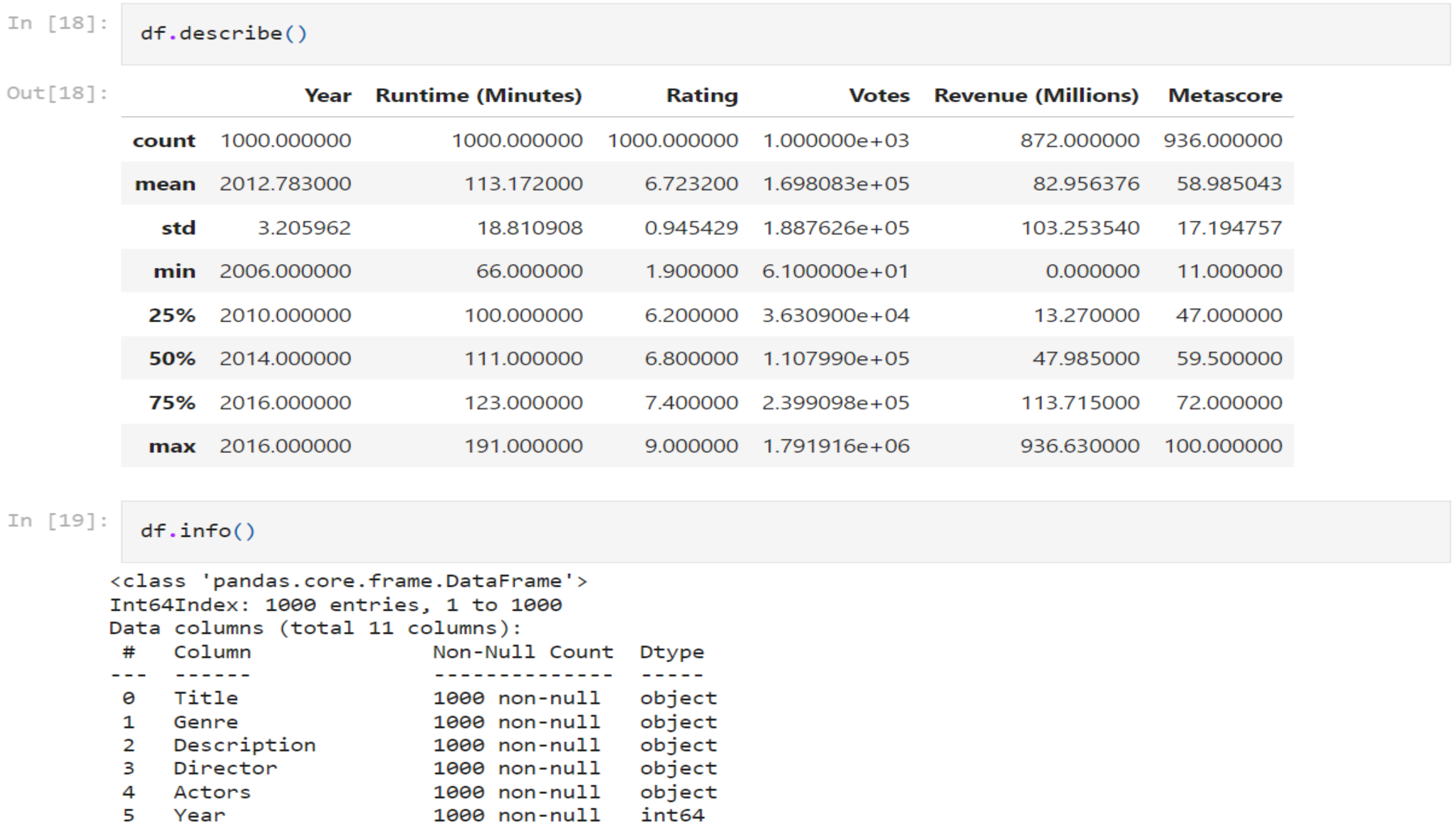
20-09-2023



```
In [16]: df = pd.read_csv("/home/labuser/Downloads/IMDB-Movie-Data.csv",index_col='Rank')

In [17]: df.head(5)

Out[17]:
```

	Title	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Rating
Rank								
1	Guardians of the Galaxy	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2014	121	8.
2	Prometheus	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7.
3	Split	Horror,Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...	2016	117	7.
4	Sing	Animation,Comedy,Family	In a city of humanoid animals, a hustling	Christophe Lourdelet	Matthew McConaughey,Reese Witherspoon, Seth	2016	108	7.

20-09-2023



In [18]:
```
df.describe()
```

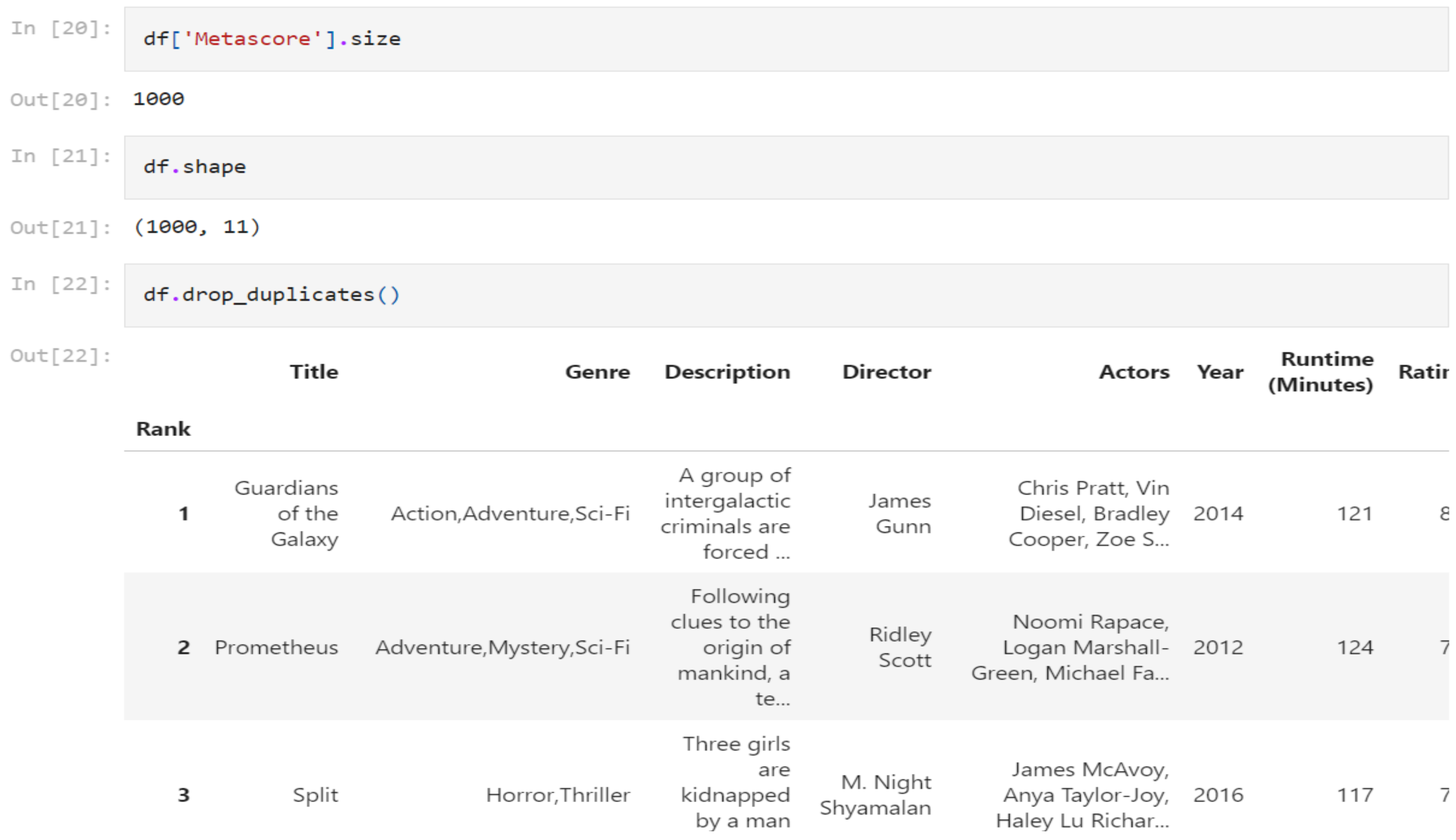
Out[18]:

	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions)	Metascore
count	1000.000000	1000.000000	1000.000000	1.000000e+03	872.000000	936.000000
mean	2012.783000	113.172000	6.723200	1.698083e+05	82.956376	58.985043
std	3.205962	18.810908	0.945429	1.887626e+05	103.253540	17.194757
min	2006.000000	66.000000	1.900000	6.100000e+01	0.000000	11.000000
25%	2010.000000	100.000000	6.200000	3.630900e+04	13.270000	47.000000
50%	2014.000000	111.000000	6.800000	1.107990e+05	47.985000	59.500000
75%	2016.000000	123.000000	7.400000	2.399098e+05	113.715000	72.000000
max	2016.000000	191.000000	9.000000	1.791916e+06	936.630000	100.000000

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 1 to 1000
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Title               1000 non-null   object
 1   Genre               1000 non-null   object
 2   Description         1000 non-null   object
 3   Director            1000 non-null   object
 4   Actors              1000 non-null   object
 5   Year                1000 non-null   int64
```

20-09-2023



```
In [20]: df['Metascore'].size
```

```
Out[20]: 1000
```

```
In [21]: df.shape
```

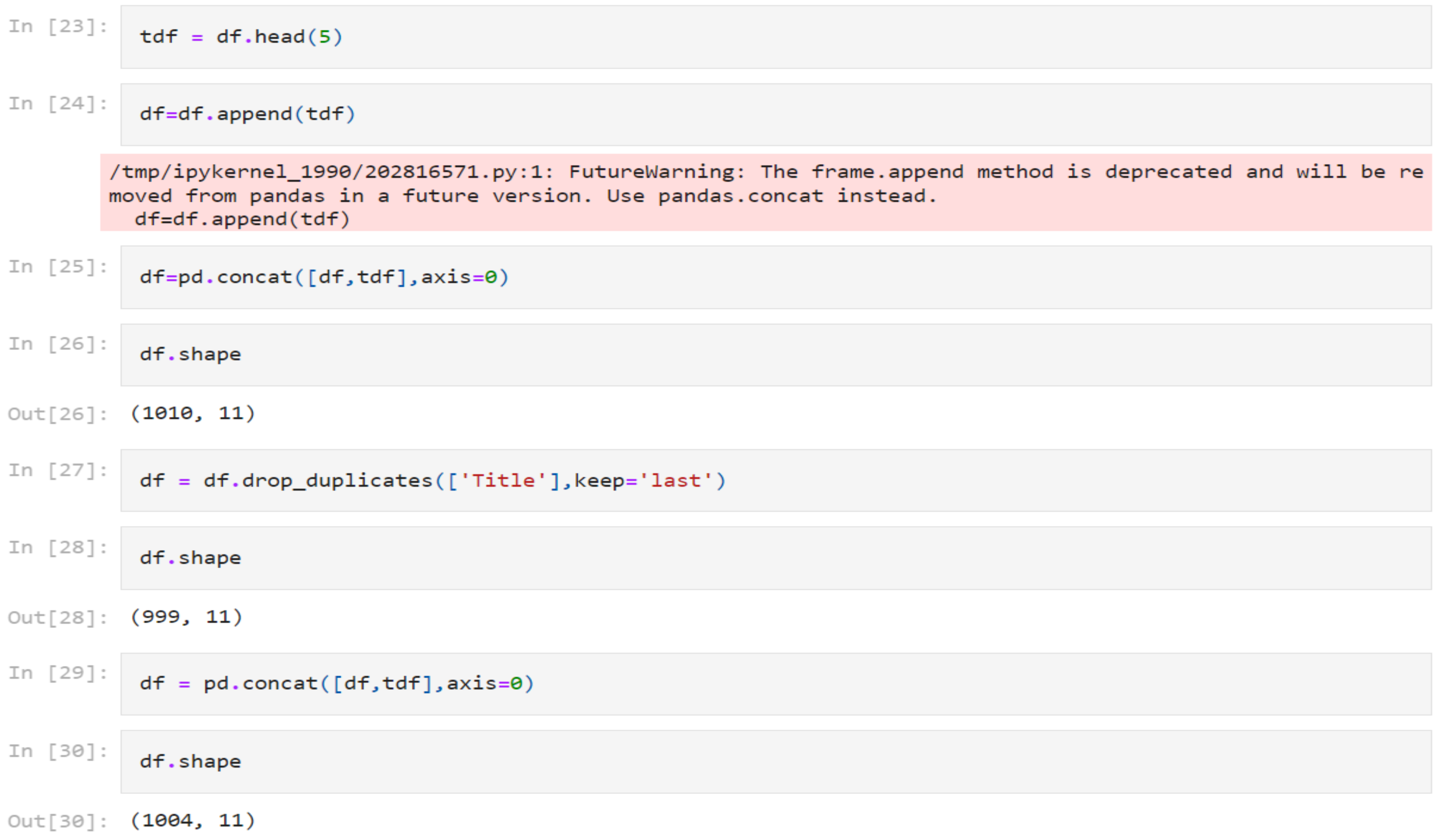
```
Out[21]: (1000, 11)
```

```
In [22]: df.drop_duplicates()
```

Out[22]:

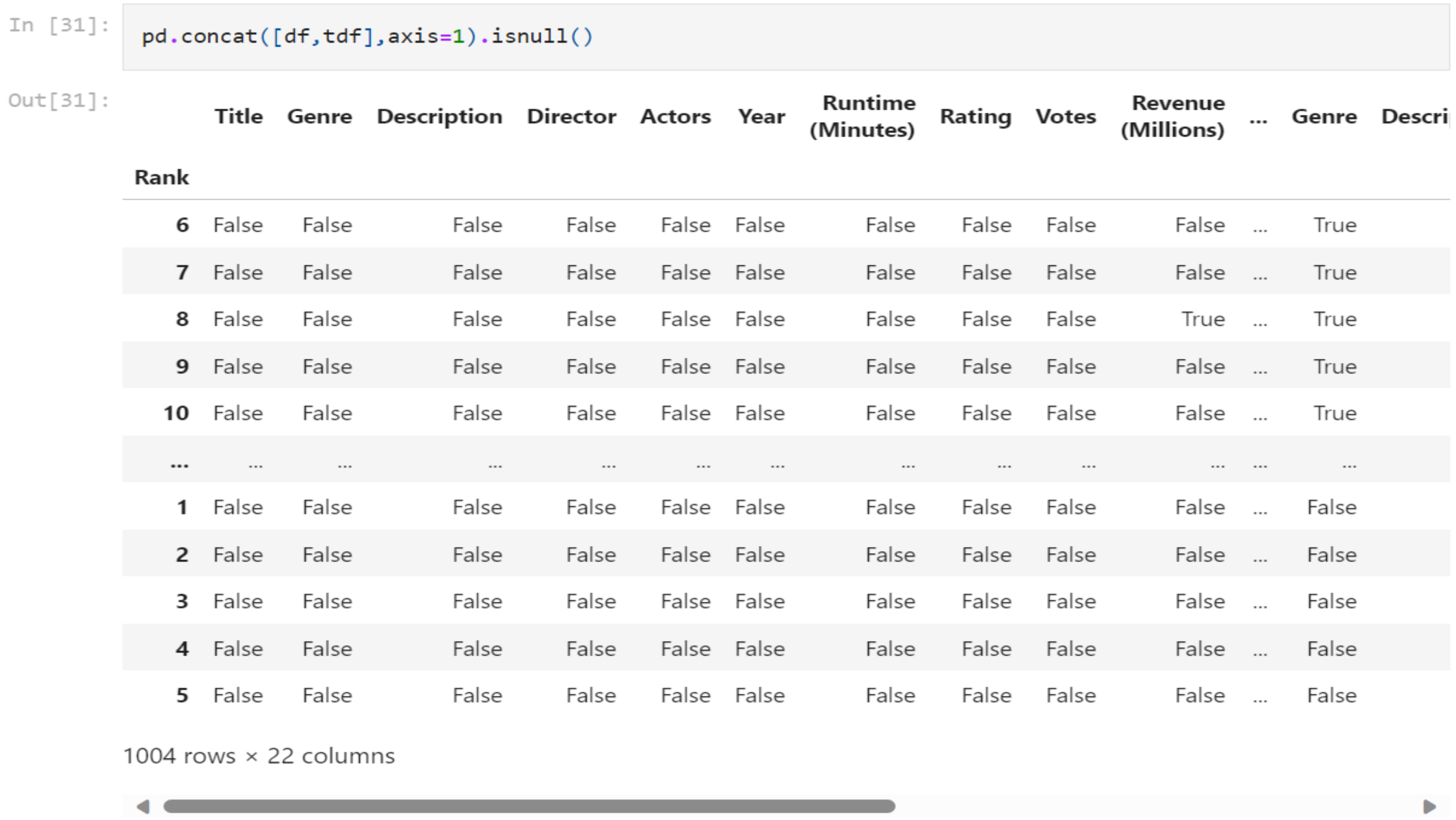
	Title	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Ratir
Rank								
1	Guardians of the Galaxy	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2014	121	8
2	Prometheus	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7
3	Split	Horror,Thriller	Three girls are kidnapped by a man	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...	2016	117	7

20-09-2023



```
In [23]:  tdf = df.head(5)

In [24]:  df=df.append(tdf)

/tmp/ipykernel_1990/202816571.py:1: FutureWarning: The frame.append method is deprecated and will be re
moved from pandas in a future version. Use pandas.concat instead.
  df=df.append(tdf)

In [25]:  df=pd.concat([df,tdf],axis=0)

In [26]:  df.shape

Out[26]:  (1010, 11)

In [27]:  df = df.drop_duplicates(['Title'],keep='last')

In [28]:  df.shape

Out[28]:  (999, 11)

In [29]:  df = pd.concat([df,tdf],axis=0)

In [30]:  df.shape

Out[30]:  (1004, 11)
```

20-09-2023



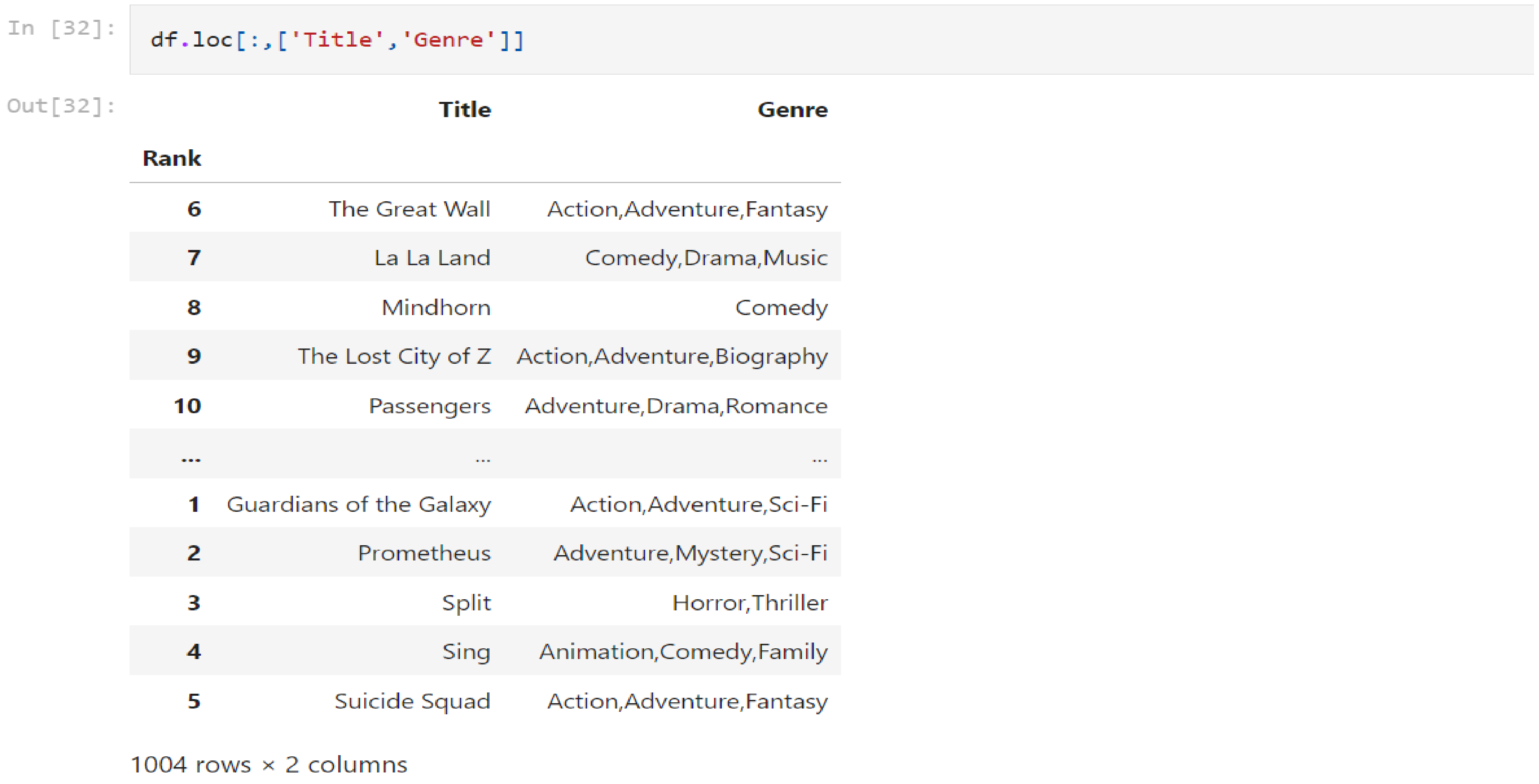
```
In [31]:  pd.concat([df,tdf],axis=1).isnull()
```

Out[31]:

	Title	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions)	...	Genre	Descri
Rank													
6	False	False	False	False	False	False	False	False	False	False	...	True	
7	False	False	False	False	False	False	False	False	False	False	...	True	
8	False	False	False	False	False	False	False	False	False	True	...	True	
9	False	False	False	False	False	False	False	False	False	False	...	True	
10	False	False	False	False	False	False	False	False	False	False	...	True	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1	False	False	False	False	False	False	False	False	False	False	...	False	
2	False	False	False	False	False	False	False	False	False	False	...	False	
3	False	False	False	False	False	False	False	False	False	False	...	False	
4	False	False	False	False	False	False	False	False	False	False	...	False	
5	False	False	False	False	False	False	False	False	False	False	...	False	

1004 rows × 22 columns

20-09-2023



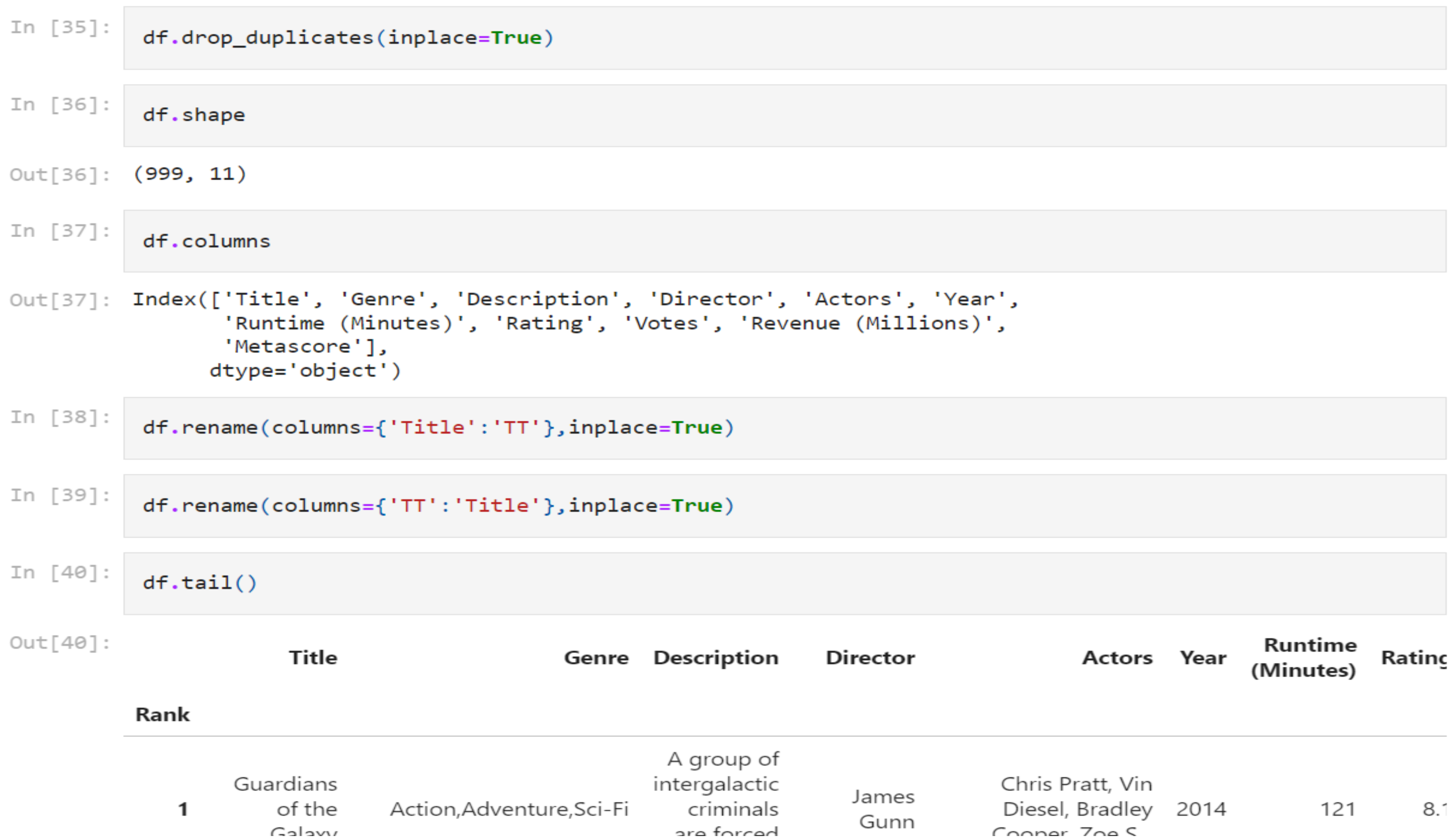
```
In [32]: df.loc[:,['Title','Genre']]
```

Out[32]:

Rank	Title	Genre
6	The Great Wall	Action,Adventure,Fantasy
7	La La Land	Comedy,Drama,Music
8	Mindhorn	Comedy
9	The Lost City of Z	Action,Adventure,Biography
10	Passengers	Adventure,Drama,Romance
...	...	...
1	Guardians of the Galaxy	Action,Adventure,Sci-Fi
2	Prometheus	Adventure,Mystery,Sci-Fi
3	Split	Horror,Thriller
4	Sing	Animation,Comedy,Family
5	Suicide Squad	Action,Adventure,Fantasy

1004 rows × 2 columns

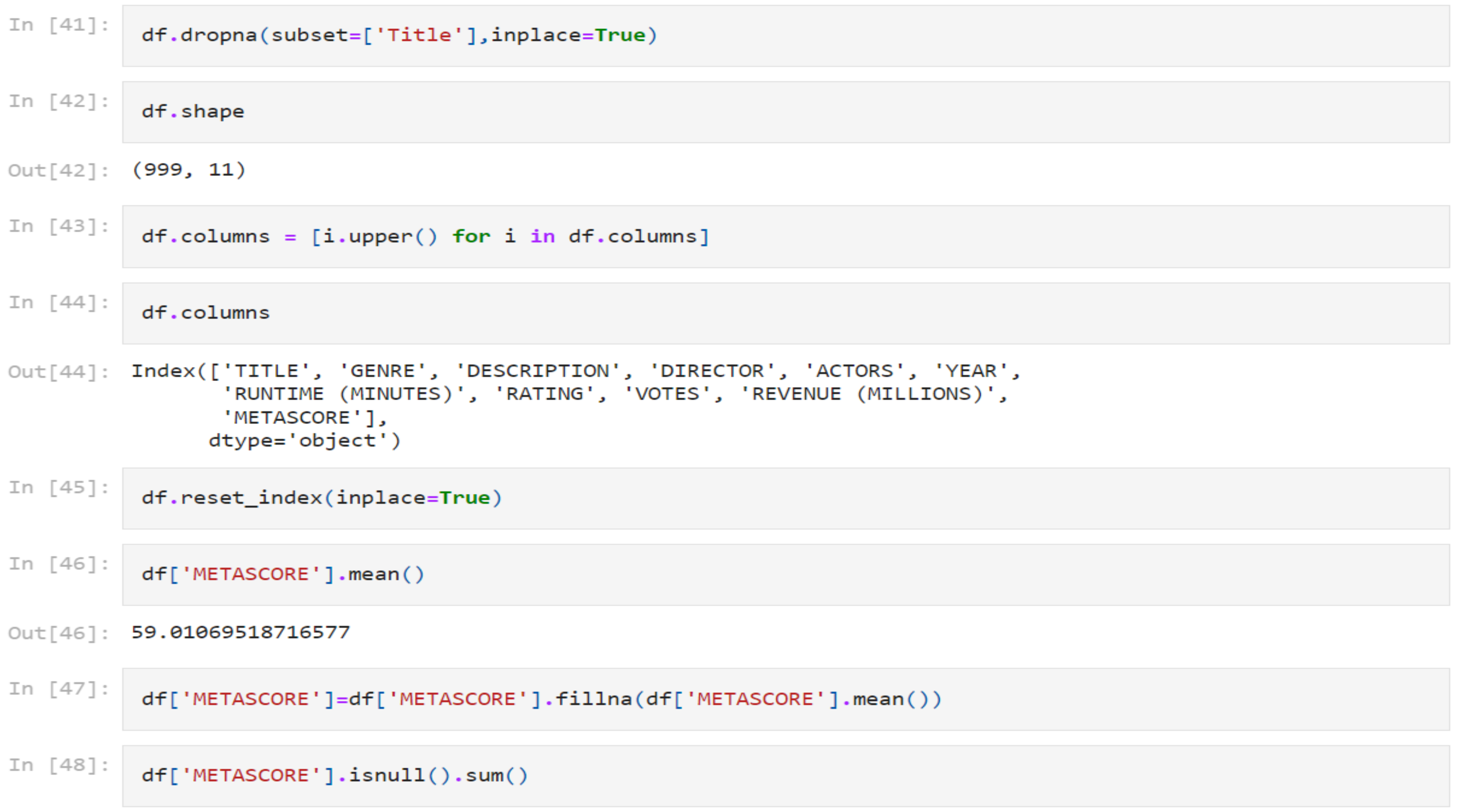
20-09-2023



```
In [35]: df.drop_duplicates(inplace=True)

In [36]: df.shape

Out[36]: (999, 11)

In [37]: df.columns

Out[37]: Index(['Title', 'Genre', 'Description', 'Director', 'Actors', 'Year',
               'Runtime (Minutes)', 'Rating', 'Votes', 'Revenue (Millions)',
               'Metascore'],
              dtype='object')

In [38]: df.rename(columns={'Title':'TT'},inplace=True)

In [39]: df.rename(columns={'TT':'Title'},inplace=True)

In [40]: df.tail()

Out[40]:
```

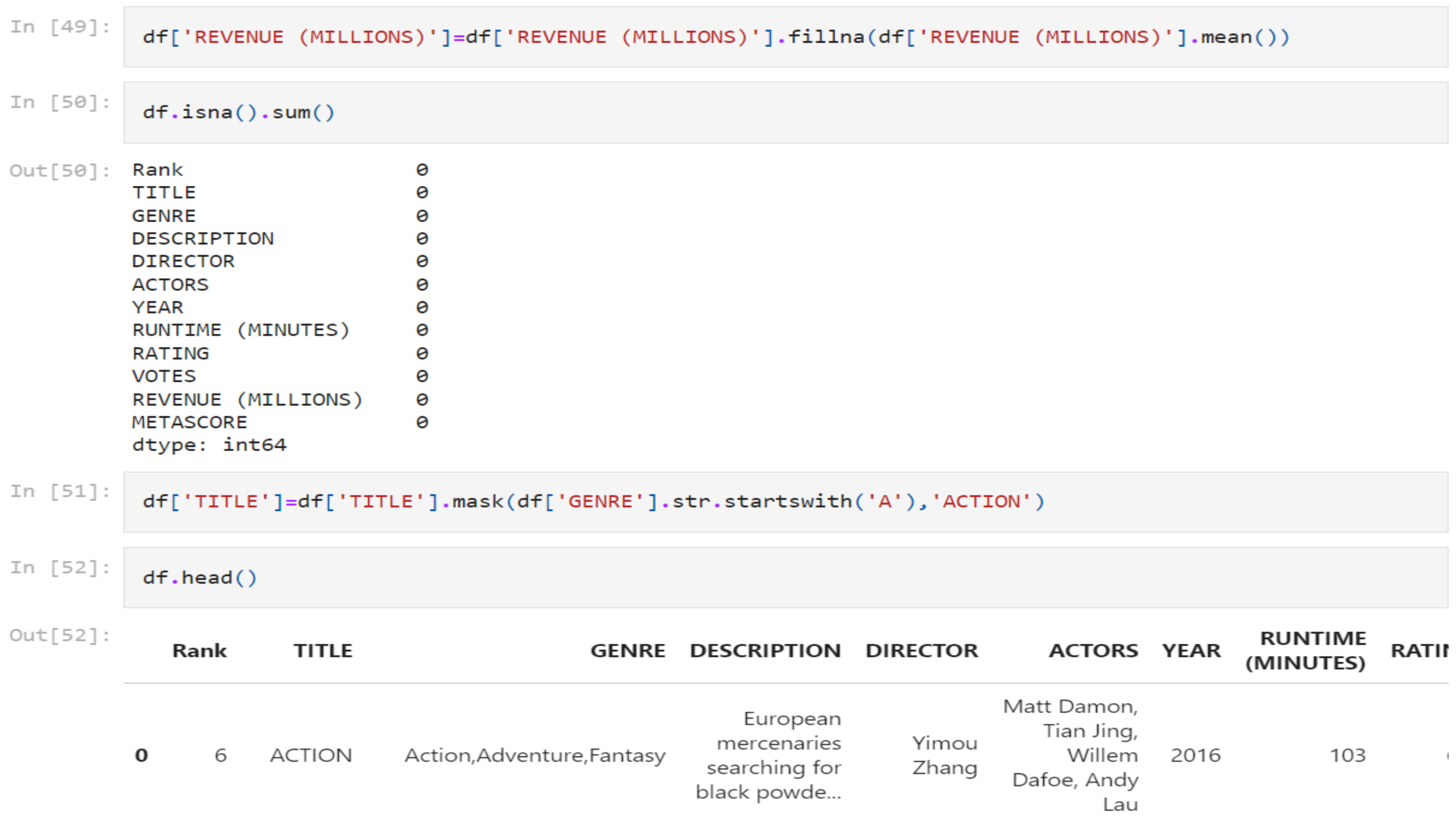
	Title	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Ratin
Rank								
1	Guardians of the	Action,Adventure,Sci-Fi	A group of intergalactic criminals	James Gunn	Chris Pratt, Vin Diesel, Bradley	2014	121	8.

20-09-2023



```
In [41]: df.dropna(subset=['Title'],inplace=True)

In [42]: df.shape

Out[42]: (999, 11)

In [43]: df.columns = [i.upper() for i in df.columns]

In [44]: df.columns

Out[44]: Index(['TITLE', 'GENRE', 'DESCRIPTION', 'DIRECTOR', 'ACTORS', 'YEAR',
               'RUNTIME (MINUTES)', 'RATING', 'VOTES', 'REVENUE (MILLIONS)',
               'METASCORE'],
              dtype='object')

In [45]: df.reset_index(inplace=True)

In [46]: df['METASCORE'].mean()

Out[46]: 59.01069518716577

In [47]: df['METASCORE']=df['METASCORE'].fillna(df['METASCORE'].mean())

In [48]: df['METASCORE'].isnull().sum()
```

20-09-2023



```
In [49]: df['REVENUE (MILLIONS)']=df['REVENUE (MILLIONS)'].fillna(df['REVENUE (MILLIONS)'].mean())
```

```
In [50]: df.isna().sum()
```

```
Out[50]: Rank                  0
         TITLE                 0
         GENRE                 0
         DESCRIPTION           0
         DIRECTOR              0
         ACTORS                0
         YEAR                  0
         RUNTIME (MINUTES)     0
         RATING                0
         VOTES                 0
         REVENUE (MILLIONS)    0
         METASCORE             0
         dtype: int64
```

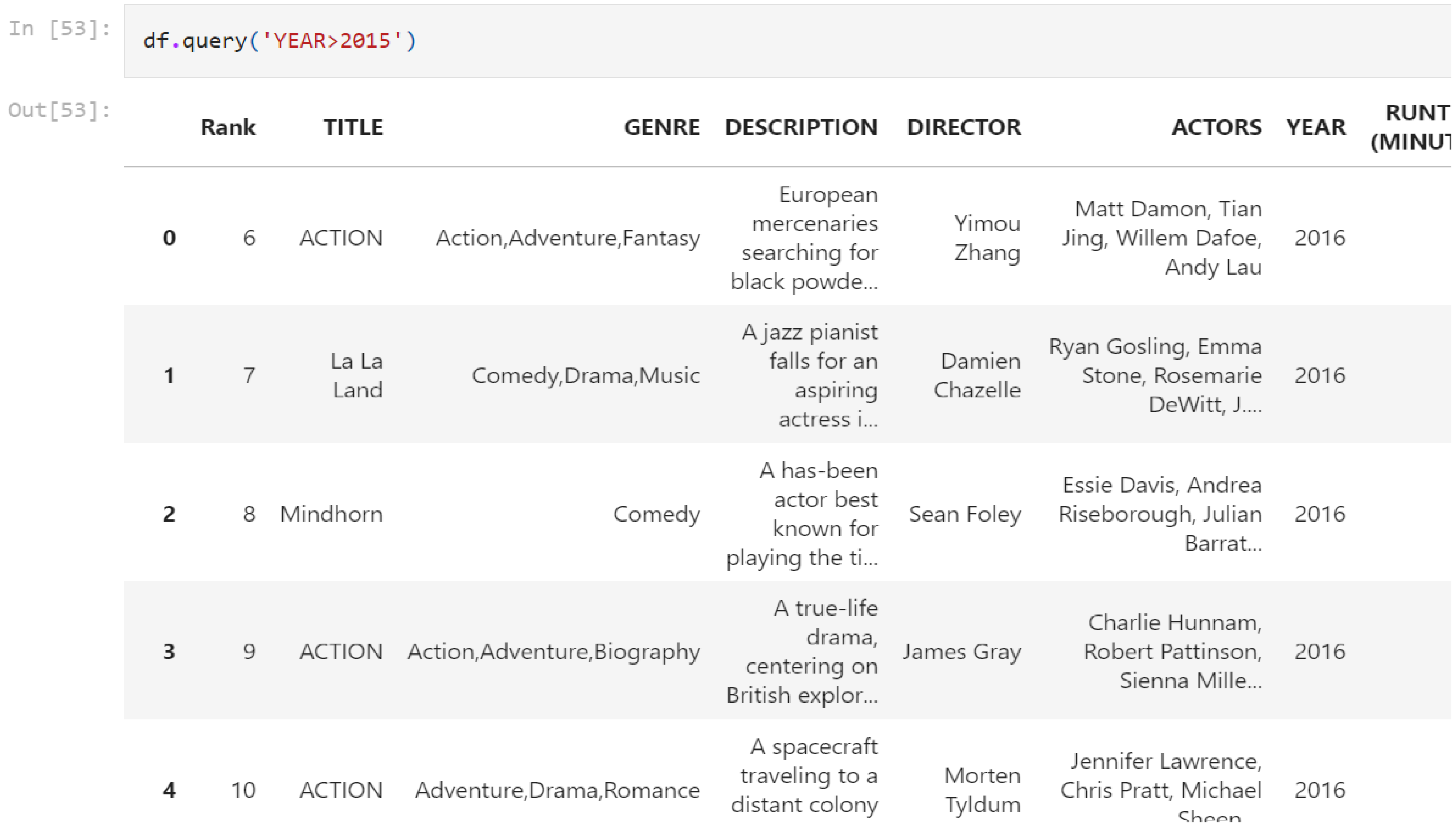
```
In [51]: df['TITLE']=df['TITLE'].mask(df['GENRE'].str.startswith('A'),'ACTION')
```

```
In [52]: df.head()
```

Out[52]:

	Rank	TITLE	GENRE	DESCRIPTION	DIRECTOR	ACTORS	YEAR	RUNTIME (MINUTES)	RATI
0	6	ACTION	Action,Adventure,Fantasy	European mercenaries searching for black powde...	Yimou Zhang	Matt Damon, Tian Jing, Willem Dafoe, Andy Lau	2016	103	

20-09-2023

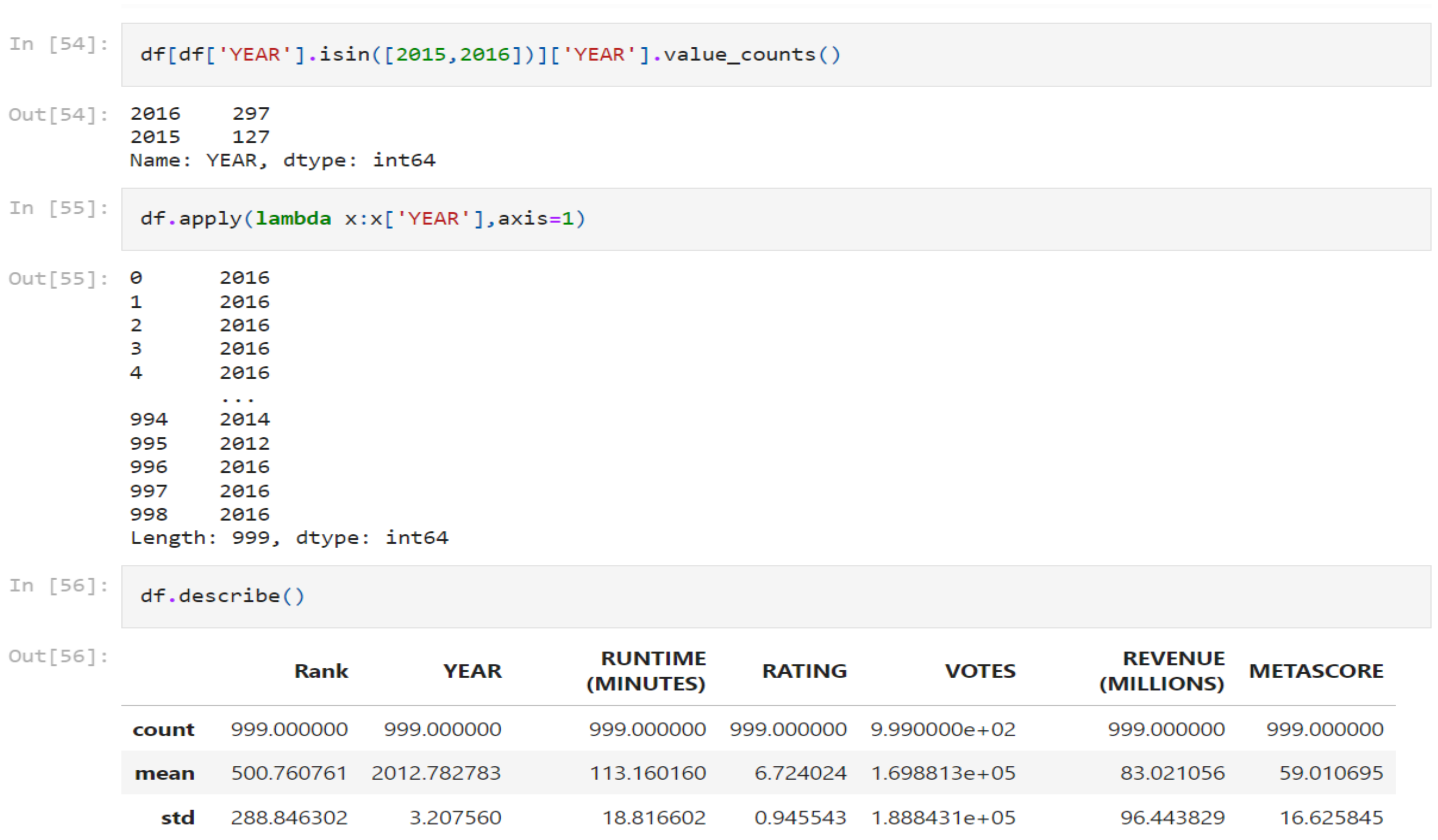


```
In [53]: df.query('YEAR>2015')
```

Out[53]:

	Rank	TITLE	GENRE	DESCRIPTION	DIRECTOR	ACTORS	YEAR	RUNT (MINUT
0	6	ACTION	Action,Adventure,Fantasy	European mercenaries searching for black powde...	Yimou Zhang	Matt Damon, Tian Jing, Willem Dafoe, Andy Lau	2016	
1	7	La La Land	Comedy,Drama,Music	A jazz pianist falls for an aspiring actress i...	Damien Chazelle	Ryan Gosling, Emma Stone, Rosemarie DeWitt, J....	2016	
2	8	Mindhorn	Comedy	A has-been actor best known for playing the ti...	Sean Foley	Essie Davis, Andrea Riseborough, Julian Barrat...	2016	
3	9	ACTION	Action,Adventure,Biography	A true-life drama, centering on British explor...	James Gray	Charlie Hunnam, Robert Pattinson, Sienna Mille...	2016	
4	10	ACTION	Adventure,Drama,Romance	A spacecraft traveling to a distant colony	Morten Tyldum	Jennifer Lawrence, Chris Pratt, Michael Sheen	2016	

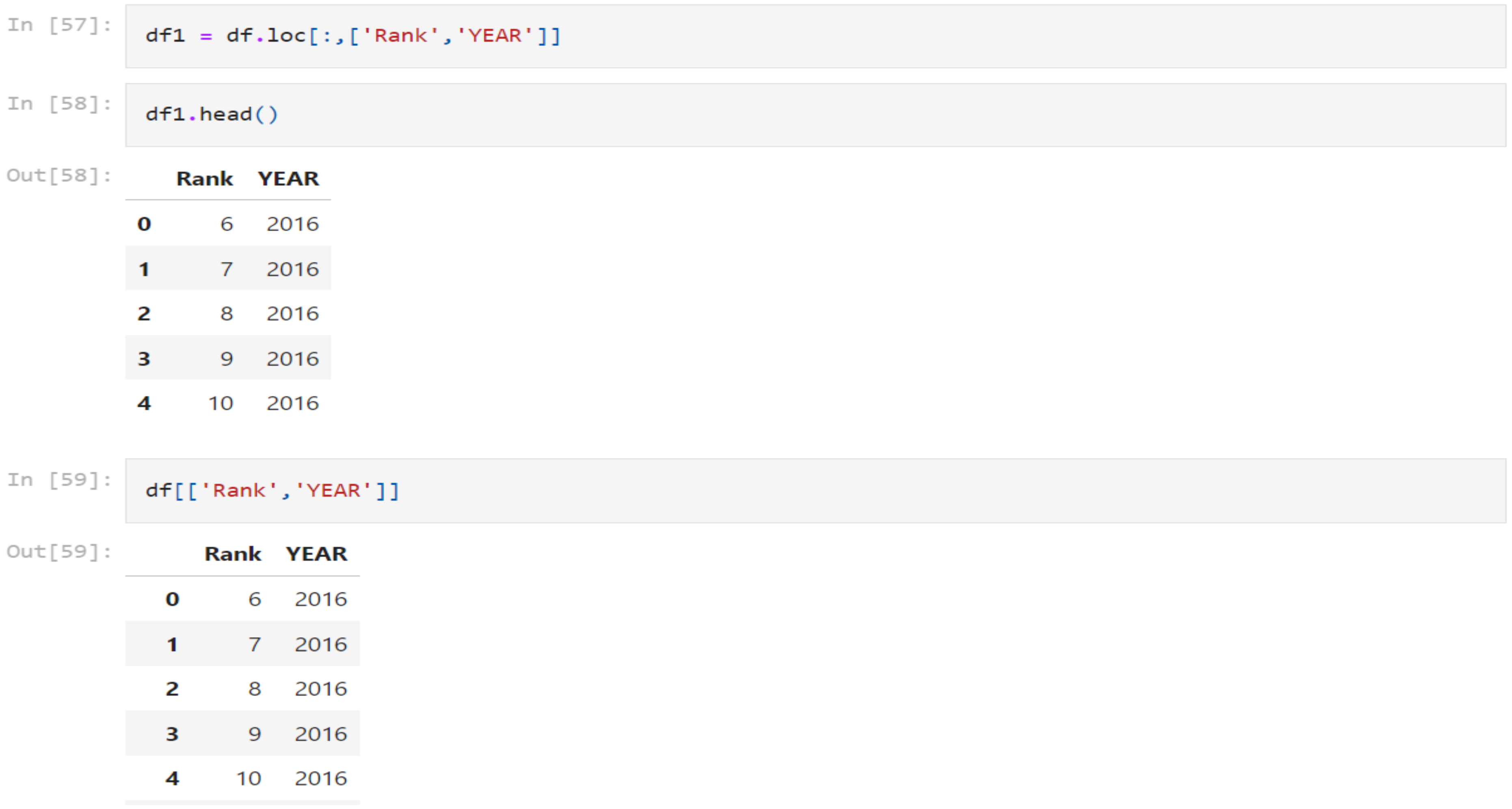
20-09-2023



```
In [54]: df[df['YEAR'].isin([2015,2016])]['YEAR'].value_counts()

Out[54]: 2016    297
         2015    127
         Name: YEAR, dtype: int64

In [55]: df.apply(lambda x:x['YEAR'],axis=1)

Out[55]: 0      2016
         1      2016
         2      2016
         3      2016
         4      2016
                ...
         994    2014
         995    2012
         996    2016
         997    2016
         998    2016
         Length: 999, dtype: int64

In [56]: df.describe()

Out[56]:
```

	Rank	YEAR	RUNTIME (MINUTES)	RATING	VOTES	REVENUE (MILLIONS)	METASCORE
count	999.000000	999.000000	999.000000	999.000000	9.990000e+02	999.000000	999.000000
mean	500.760761	2012.782783	113.160160	6.724024	1.698813e+05	83.021056	59.010695
std	288.846302	3.207560	18.816602	0.945543	1.888431e+05	96.443829	16.625845

20-09-2023



```
In [57]: df1 = df.loc[:,['Rank','YEAR']]
```

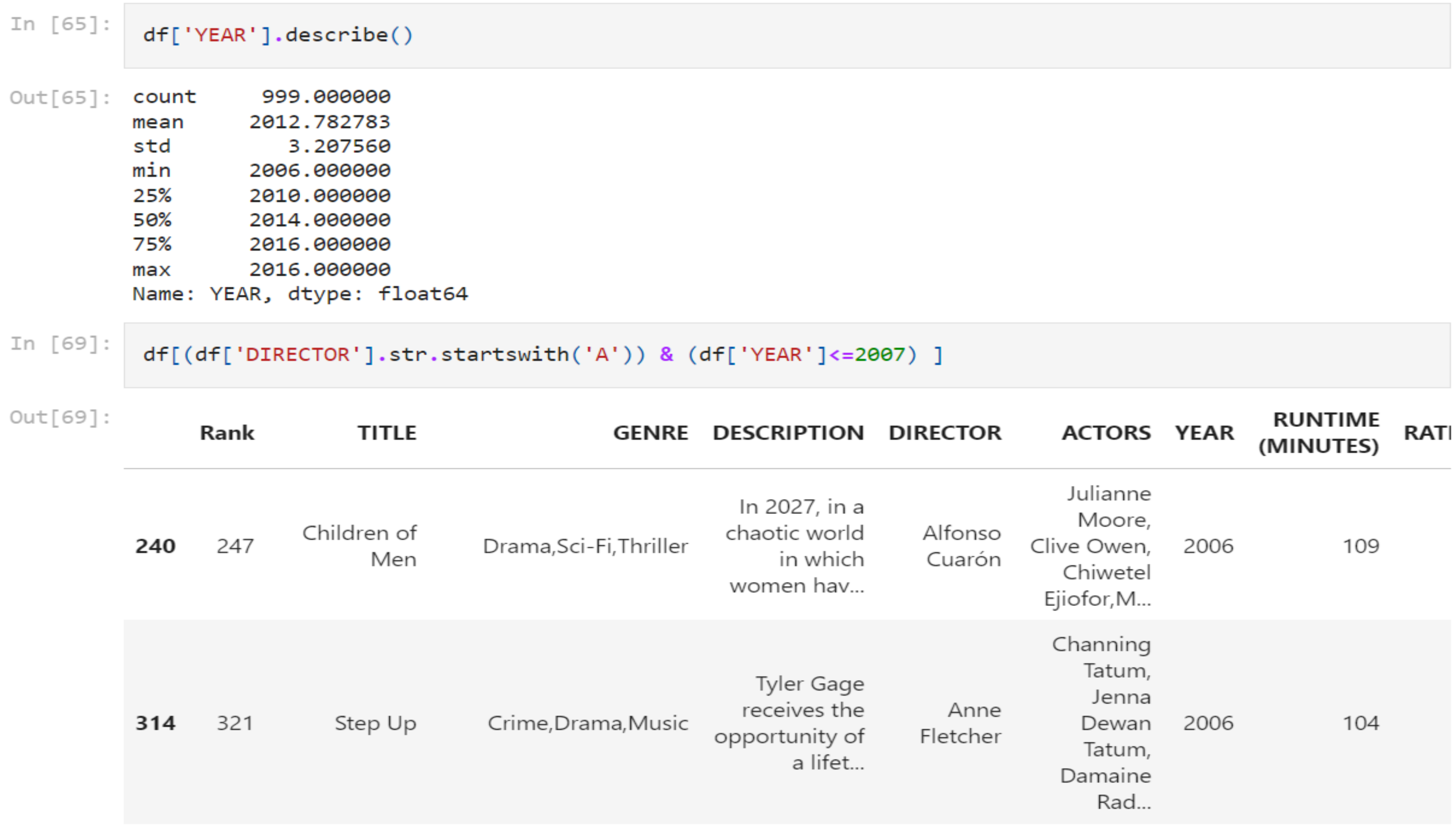
```
In [58]: df1.head()
```

```
Out[58]:
   Rank  YEAR
0     6  2016
1     7  2016
2     8  2016
3     9  2016
4    10  2016
```

```
In [59]: df[['Rank','YEAR']]
```

```
Out[59]:
   Rank  YEAR
0     6  2016
1     7  2016
2     8  2016
3     9  2016
4    10  2016
```

20-09-2023



```
In [65]: df['YEAR'].describe()
```

```
Out[65]: count     999.000000
         mean     2012.782783
         std         3.207560
         min      2006.000000
         25%      2010.000000
         50%      2014.000000
         75%      2016.000000
         max      2016.000000
         Name: YEAR, dtype: float64
```

```
In [69]: df[(df['DIRECTOR'].str.startswith('A')) & (df['YEAR']<=2007) ]
```

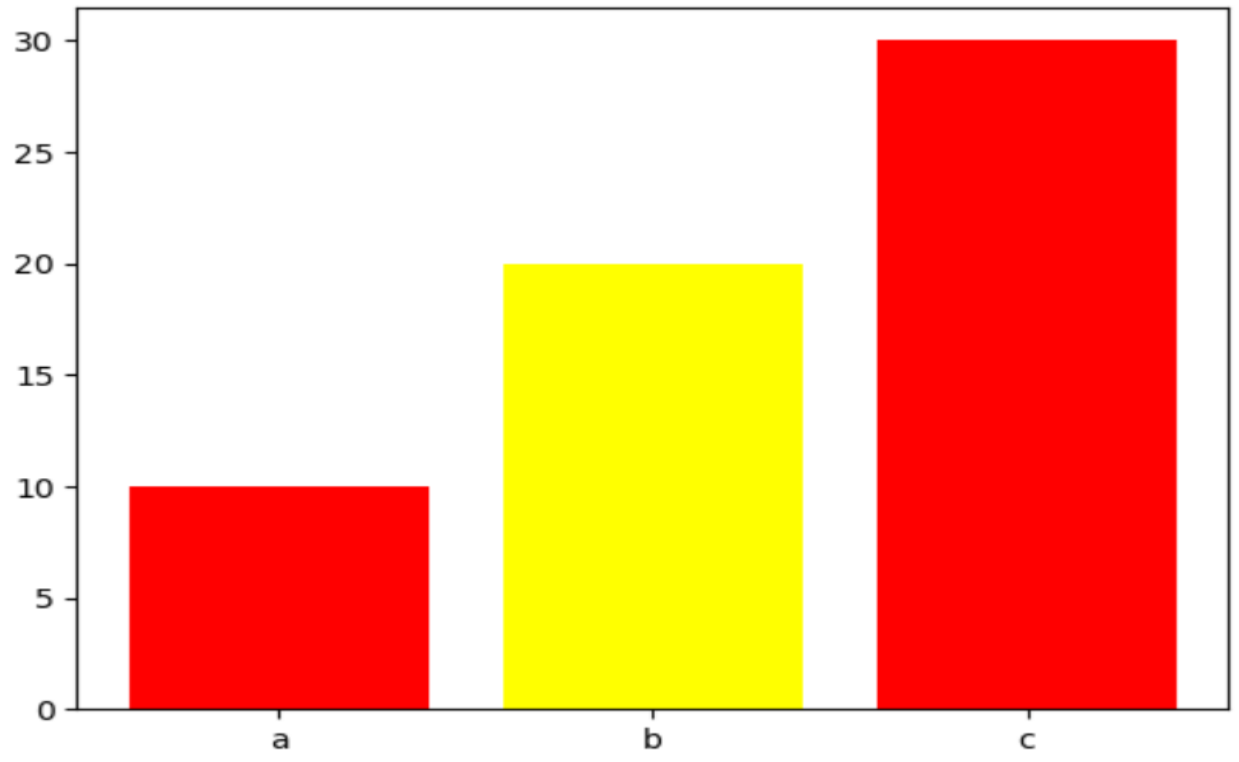
Out[69]:

	Rank	TITLE	GENRE	DESCRIPTION	DIRECTOR	ACTORS	YEAR	RUNTIME (MINUTES)	RATI
240	247	Children of Men	Drama,Sci-Fi,Thriller	In 2027, in a chaotic world in which women hav...	Alfonso Cuarón	Julianne Moore, Clive Owen, Chiwetel Ejiofor,M...	2006	109	
314	321	Step Up	Crime,Drama,Music	Tyler Gage receives the opportunity of a lifet...	Anne Fletcher	Channing Tatum, Jenna Dewan Tatum, Damaine Rad...	2006	104	

20-09-2023

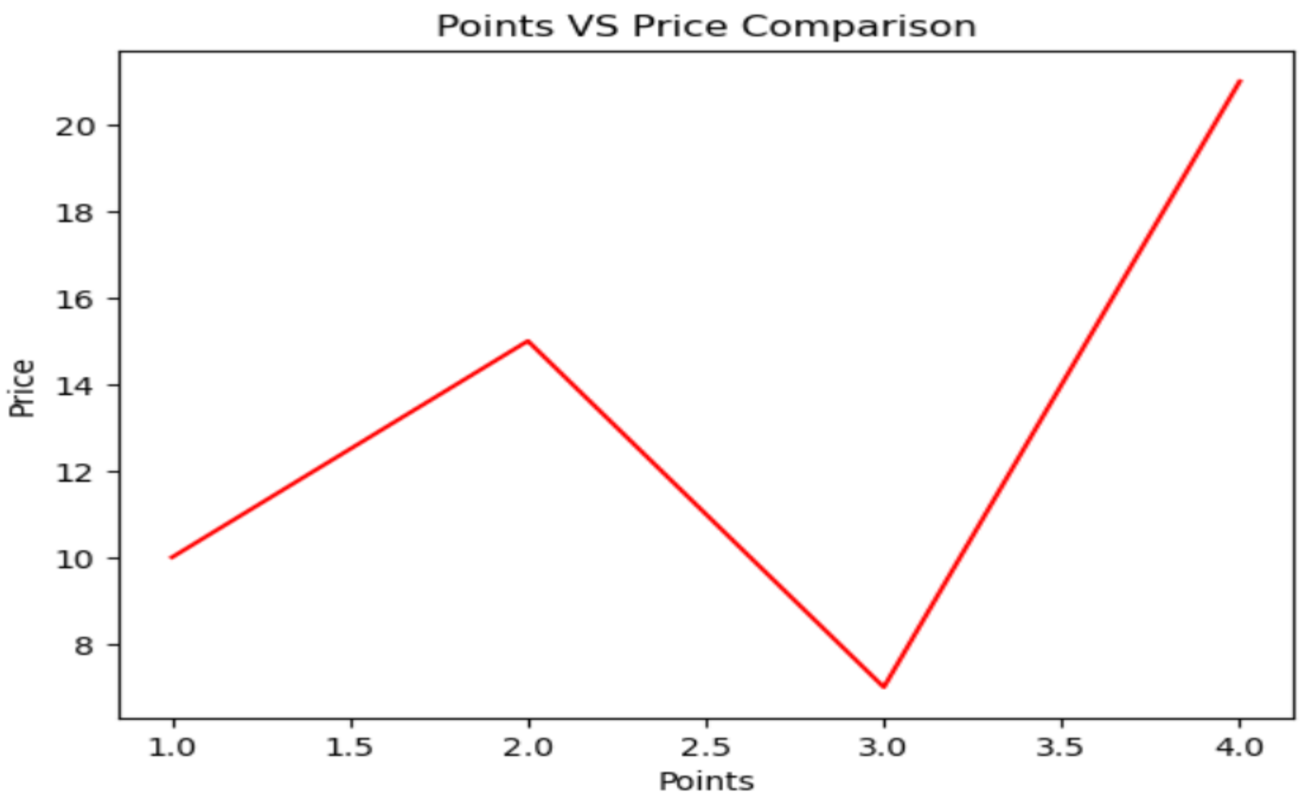
Matplotlib

In [89]:
```
import matplotlib.pyplot as plt
plt.bar(['a','b','c'],[10,20,30],color=['red','yellow'])
plt.show()
```

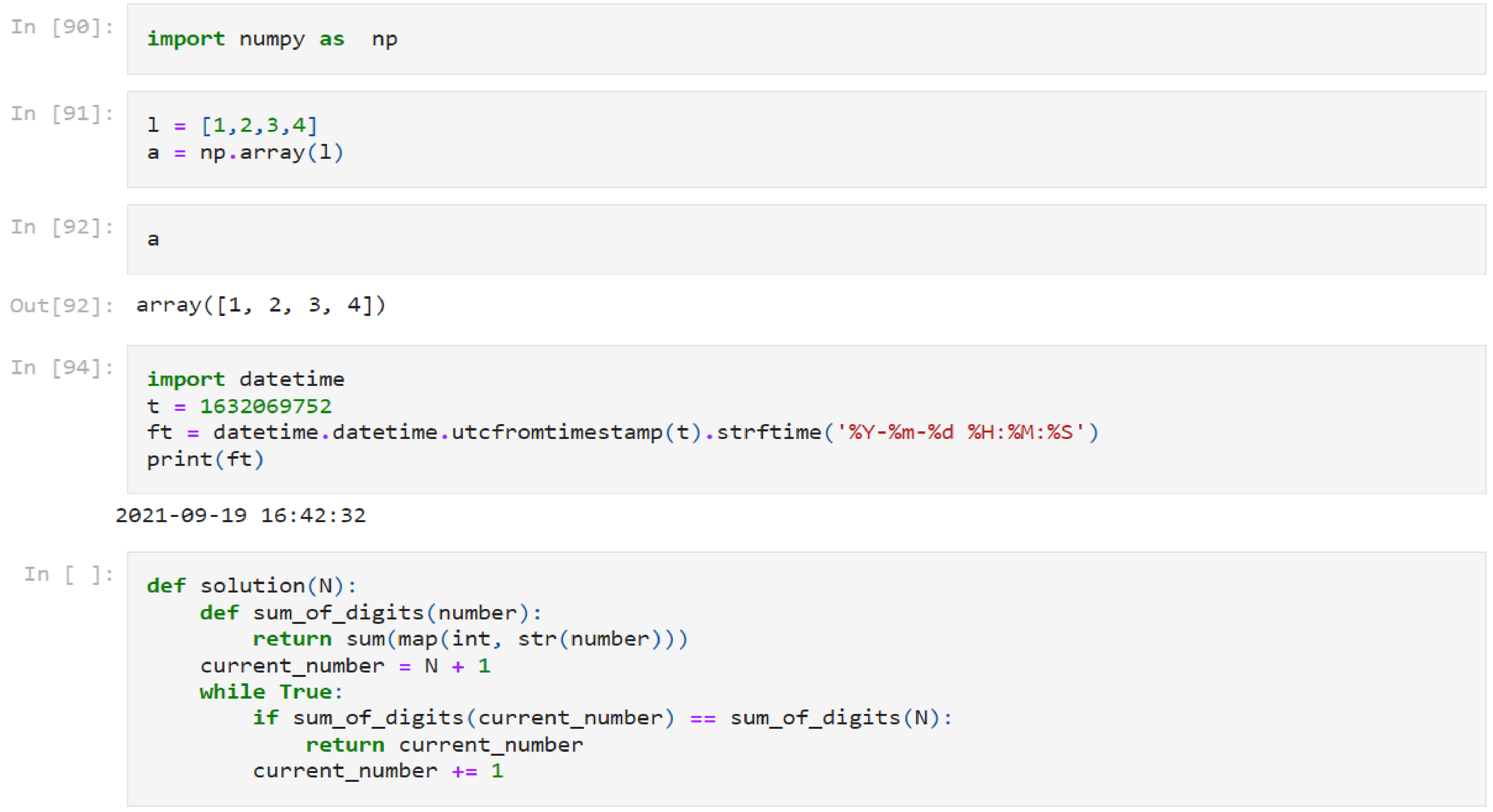


20-09-2023

```
In [88]:  plt.plot([1,2,3,4],[10,15,7,21],color='red')
          plt.xlabel("Points")
          plt.ylabel("Price")
          plt.title("Points VS Price Comparison")

Out[88]:  Text(0.5, 1.0, 'Points VS Price Comparison')
```



20-09-2023



```
In [90]: import numpy as  np
```

```
In [91]: l = [1,2,3,4]
         a = np.array(l)
```

```
In [92]: a

Out[92]: array([1, 2, 3, 4])
```

```
In [94]: import datetime
         t = 1632069752
         ft = datetime.datetime.utcfromtimestamp(t).strftime('%Y-%m-%d %H:%M:%S')
         print(ft)

        2021-09-19 16:42:32
```

```
In [ ]: def solution(N):
            def sum_of_digits(number):
                return sum(map(int, str(number)))
            current_number = N + 1
            while True:
                if sum_of_digits(current_number) == sum_of_digits(N):
                    return current_number
                current_number += 1
```

21-09-2023

- Spark is computing engine used to process the big data. It uses to process the data in distributed manner.
- Py-spark is python library that wraps spark.
- Spark supports 4 Languages.
- Scala
- Java
- Python
- R
- We can use cloud platforms to create the virtual machines and use it as nodes

21-09-2023

- driver node creates the job according to actions submitted.
- Each action creates one or more stages according to shuffling that takes place.
- Each stage can be divided into the multiple tasks and each task can be assigned to the core of the worker node.
- Worker node performs all the tasks required to complete the job.
- Cluster manager manages the resources required to complete the task by the worker.

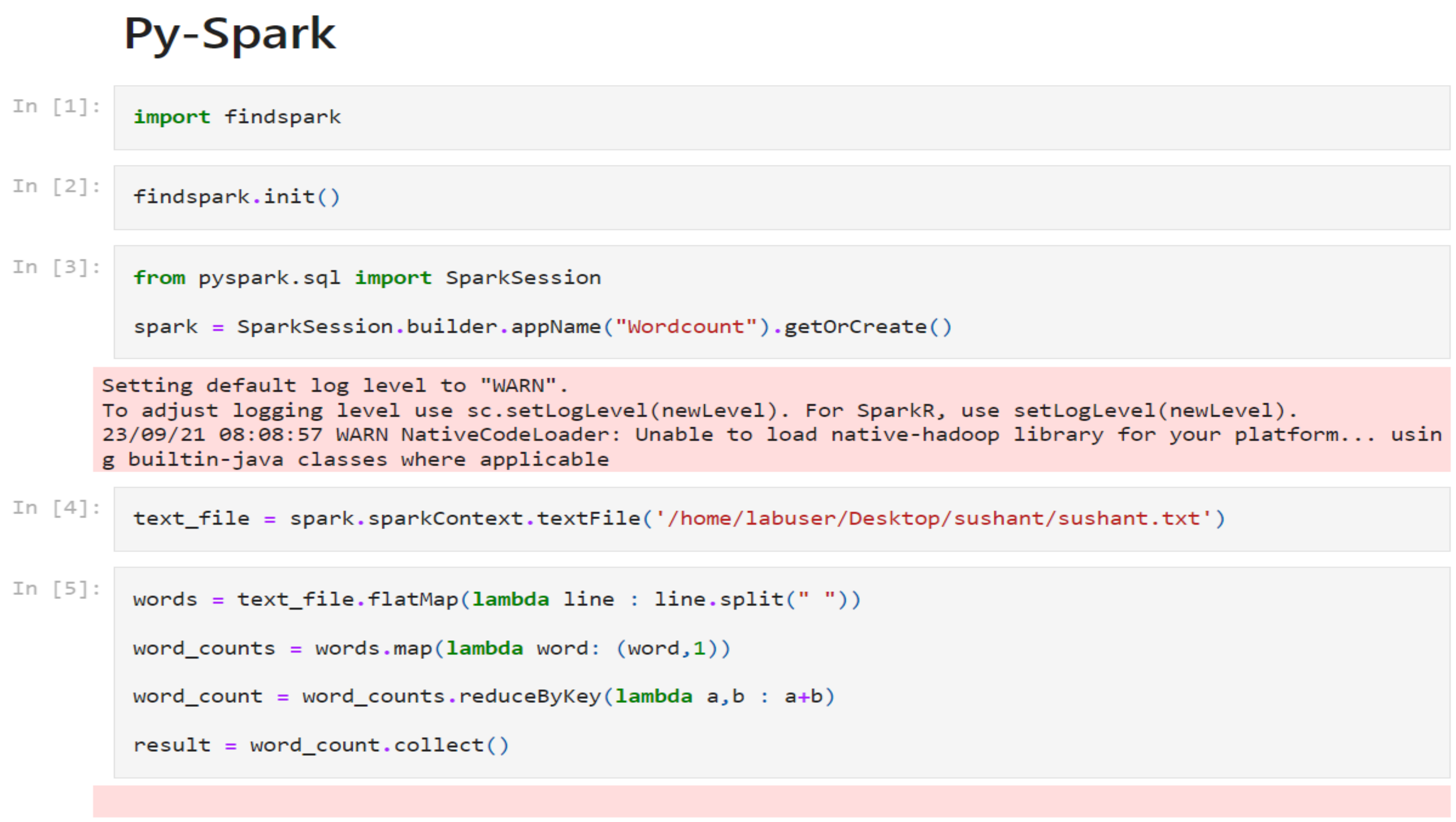
21-09-2023

- Transformation
- 1. Narrow transformation
- 2. Wide transformation
- Narrow transformation do not require any shuffling of between the worker nodes.
- It takes less time to complete the task
- Wide transformation require shuffling of data between the worker node.
- It takes more time to shuffling the data between the worker nodes.

21-09-2023

- RDD resilient distributed dataset
- It is immutable it helps in recovering in case of failure.
- RDD do not support rows and columns
- DAG directed acyclic graph
- Directed acyclic graphs will be used to refer transformation.
- Caching is used to cache the data so it will reduce the latency of accessing the data.

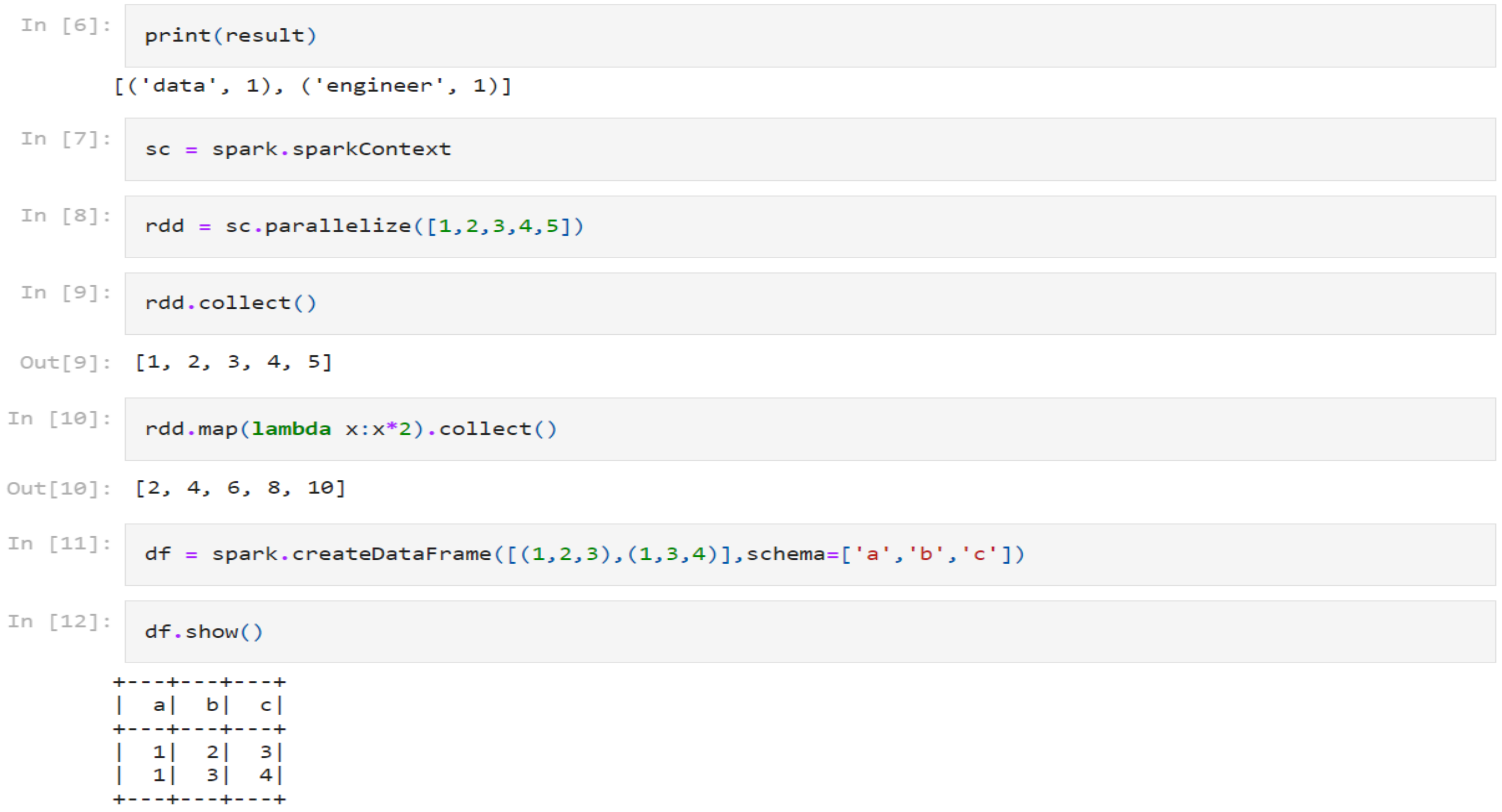
21-09-2023



Py-Spark

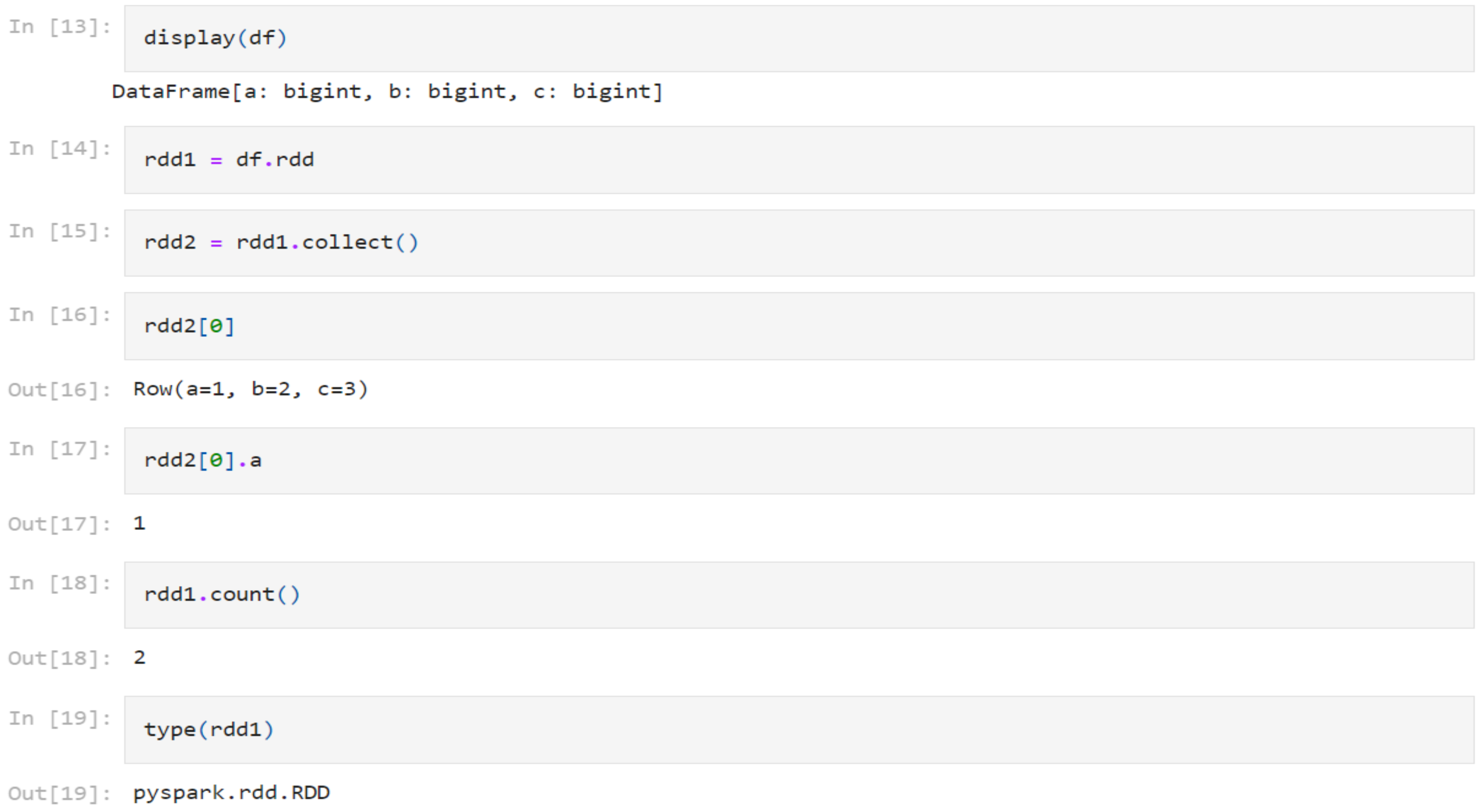
```
In [1]: import findspark

In [2]: findspark.init()

In [3]: from pyspark.sql import SparkSession

        spark = SparkSession.builder.appName("Wordcount").getOrCreate()

Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/09/21 08:08:57 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... usin
g builtin-java classes where applicable

In [4]: text_file = spark.sparkContext.textFile('/home/labuser/Desktop/sushant/sushant.txt')

In [5]: words = text_file.flatMap(lambda line : line.split(" "))

        word_counts = words.map(lambda word: (word,1))

        word_count = word_counts.reduceByKey(lambda a,b : a+b)

        result = word_count.collect()
```

21-09-2023



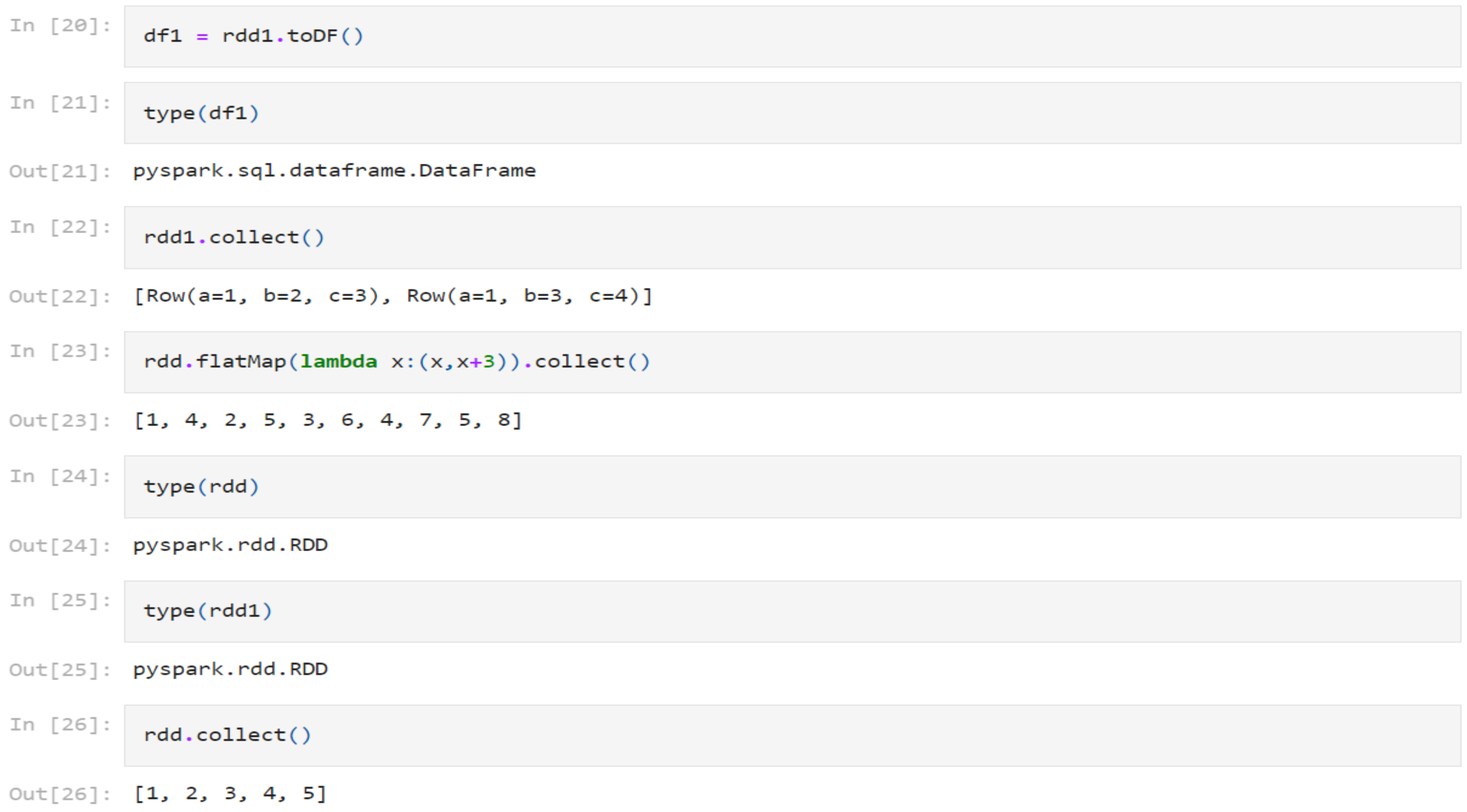
```
In [6]:  print(result)

         [('data', 1), ('engineer', 1)]

In [7]:  sc = spark.sparkContext

In [8]:  rdd = sc.parallelize([1,2,3,4,5])

In [9]:  rdd.collect()

Out[9]:  [1, 2, 3, 4, 5]

In [10]: rdd.map(lambda x:x*2).collect()

Out[10]: [2, 4, 6, 8, 10]

In [11]: df = spark.createDataFrame([(1,2,3),(1,3,4)],schema=['a','b','c'])

In [12]: df.show()

         +---+---+---+
         |  a|  b|  c|
         +---+---+---+
         |  1|  2|  3|
         |  1|  3|  4|
         +---+---+---+
```

21-09-2023



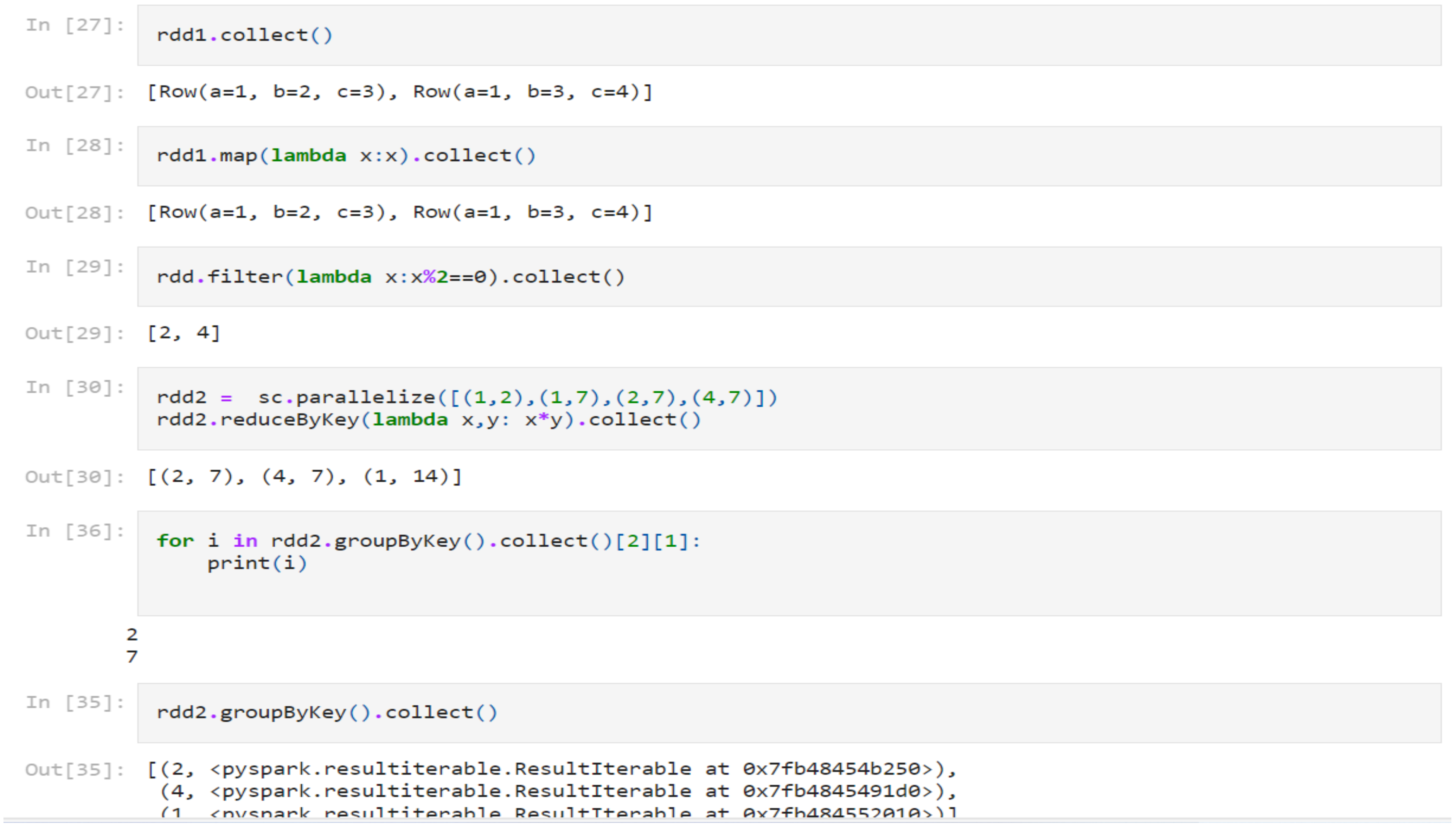
```
In [13]: display(df)

         DataFrame[a: bigint, b: bigint, c: bigint]

In [14]: rdd1 = df.rdd

In [15]: rdd2 = rdd1.collect()

In [16]: rdd2[0]

Out[16]: Row(a=1, b=2, c=3)

In [17]: rdd2[0].a

Out[17]: 1

In [18]: rdd1.count()

Out[18]: 2

In [19]: type(rdd1)

Out[19]: pyspark.rdd.RDD
```

21-09-2023



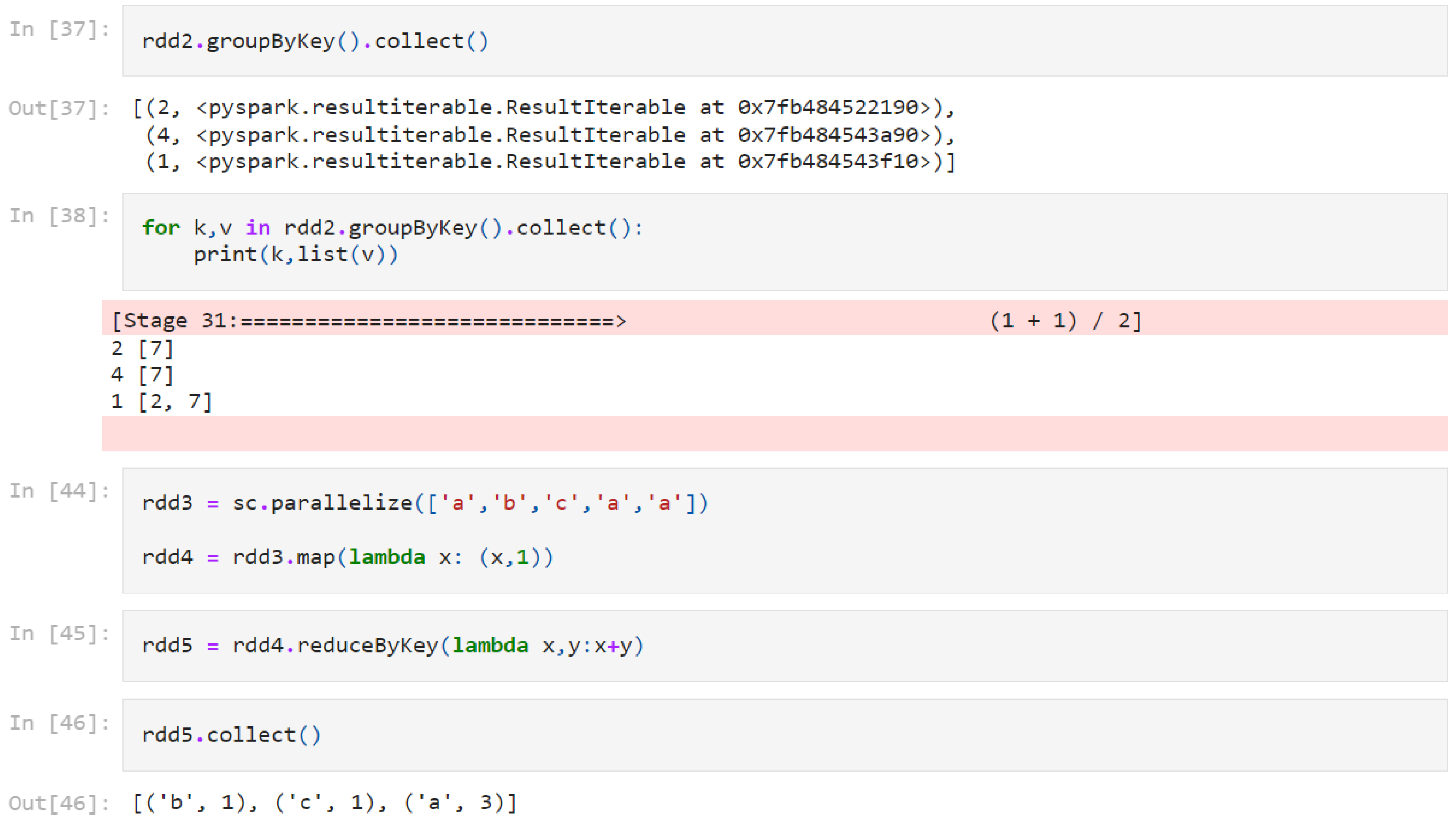
```
In [20]: df1 = rdd1.toDF()

In [21]: type(df1)

Out[21]: pyspark.sql.dataframe.DataFrame

In [22]: rdd1.collect()

Out[22]: [Row(a=1, b=2, c=3), Row(a=1, b=3, c=4)]

In [23]: rdd.flatMap(lambda x:(x,x+3)).collect()

Out[23]: [1, 4, 2, 5, 3, 6, 4, 7, 5, 8]

In [24]: type(rdd)

Out[24]: pyspark.rdd.RDD

In [25]: type(rdd1)

Out[25]: pyspark.rdd.RDD

In [26]: rdd.collect()

Out[26]: [1, 2, 3, 4, 5]
```

21-09-2023



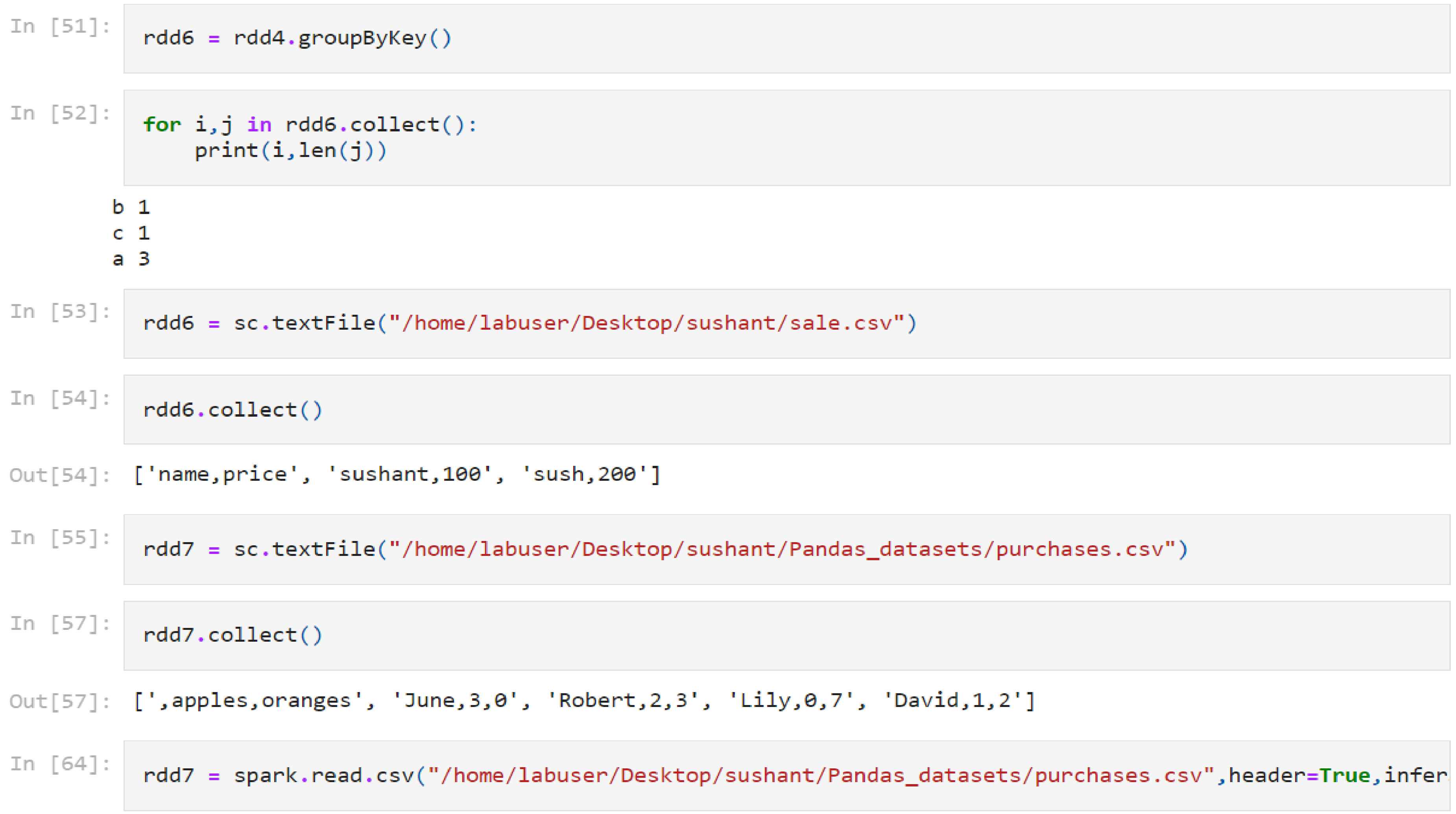
```
In [27]:  rdd1.collect()

Out[27]:  [Row(a=1, b=2, c=3), Row(a=1, b=3, c=4)]

In [28]:  rdd1.map(lambda x:x).collect()

Out[28]:  [Row(a=1, b=2, c=3), Row(a=1, b=3, c=4)]

In [29]:  rdd.filter(lambda x:x%2==0).collect()

Out[29]:  [2, 4]

In [30]:  rdd2 =  sc.parallelize([(1,2),(1,7),(2,7),(4,7)])
          rdd2.reduceByKey(lambda x,y: x*y).collect()

Out[30]:  [(2, 7), (4, 7), (1, 14)]

In [36]:  for i in rdd2.groupByKey().collect()[2][1]:
              print(i)

         2
         7

In [35]:  rdd2.groupByKey().collect()

Out[35]:  [(2, <pyspark.resultiterable.ResultIterable at 0x7fb48454b250>),
           (4, <pyspark.resultiterable.ResultIterable at 0x7fb4845491d0>),
           (1, <pyspark.resultiterable.ResultIterable at 0x7fb484552010>)]
```

21-09-2023



```
In [37]: rdd2.groupByKey().collect()

Out[37]: [(2, <pyspark.resultiterable.ResultIterable at 0x7fb484522190>),
          (4, <pyspark.resultiterable.ResultIterable at 0x7fb484543a90>),
          (1, <pyspark.resultiterable.ResultIterable at 0x7fb484543f10>)]

In [38]: for k,v in rdd2.groupByKey().collect():
             print(k,list(v))

         [Stage 31:==============================>                             (1 + 1) / 2]
         2 [7]
         4 [7]
         1 [2, 7]

In [44]: rdd3 = sc.parallelize(['a','b','c','a','a'])

         rdd4 = rdd3.map(lambda x: (x,1))

In [45]: rdd5 = rdd4.reduceByKey(lambda x,y:x+y)

In [46]: rdd5.collect()

Out[46]: [('b', 1), ('c', 1), ('a', 3)]
```

21-09-2023



```
In [51]: rdd6 = rdd4.groupByKey()
```

```
In [52]: for i,j in rdd6.collect():
             print(i,len(j))
```

```
b 1
c 1
a 3
```

```
In [53]: rdd6 = sc.textFile("/home/labuser/Desktop/sushant/sale.csv")
```

```
In [54]: rdd6.collect()
```

```
Out[54]: ['name,price', 'sushant,100', 'sush,200']
```

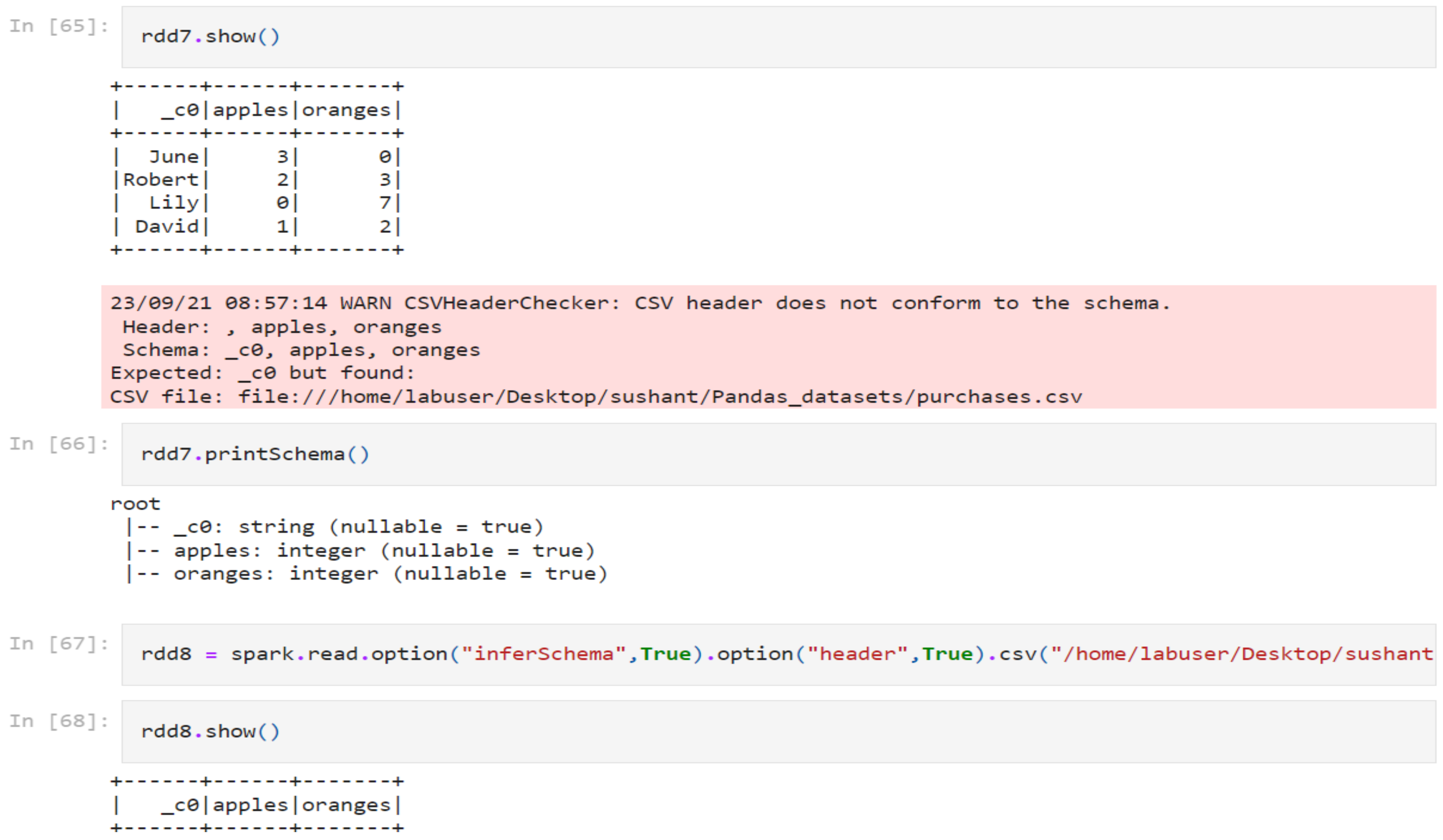
```
In [55]: rdd7 = sc.textFile("/home/labuser/Desktop/sushant/Pandas_datasets/purchases.csv")
```

```
In [57]: rdd7.collect()
```

```
Out[57]: [',apples,oranges', 'June,3,0', 'Robert,2,3', 'Lily,0,7', 'David,1,2']
```

```
In [64]: rdd7 = spark.read.csv("/home/labuser/Desktop/sushant/Pandas_datasets/purchases.csv",header=True,infer
```

21-09-2023



In [65]:
```
rdd7.show()
```

```
+------+------+-------+
|   _c0|apples|oranges|
+------+------+-------+
|  June|     3|      0|
|Robert|     2|      3|
|  Lily|     0|      7|
| David|     1|      2|
+------+------+-------+
```

```
23/09/21 08:57:14 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: , apples, oranges
 Schema: _c0, apples, oranges
Expected: _c0 but found:
CSV file: file:///home/labuser/Desktop/sushant/Pandas_datasets/purchases.csv
```

In [66]:
```
rdd7.printSchema()
```

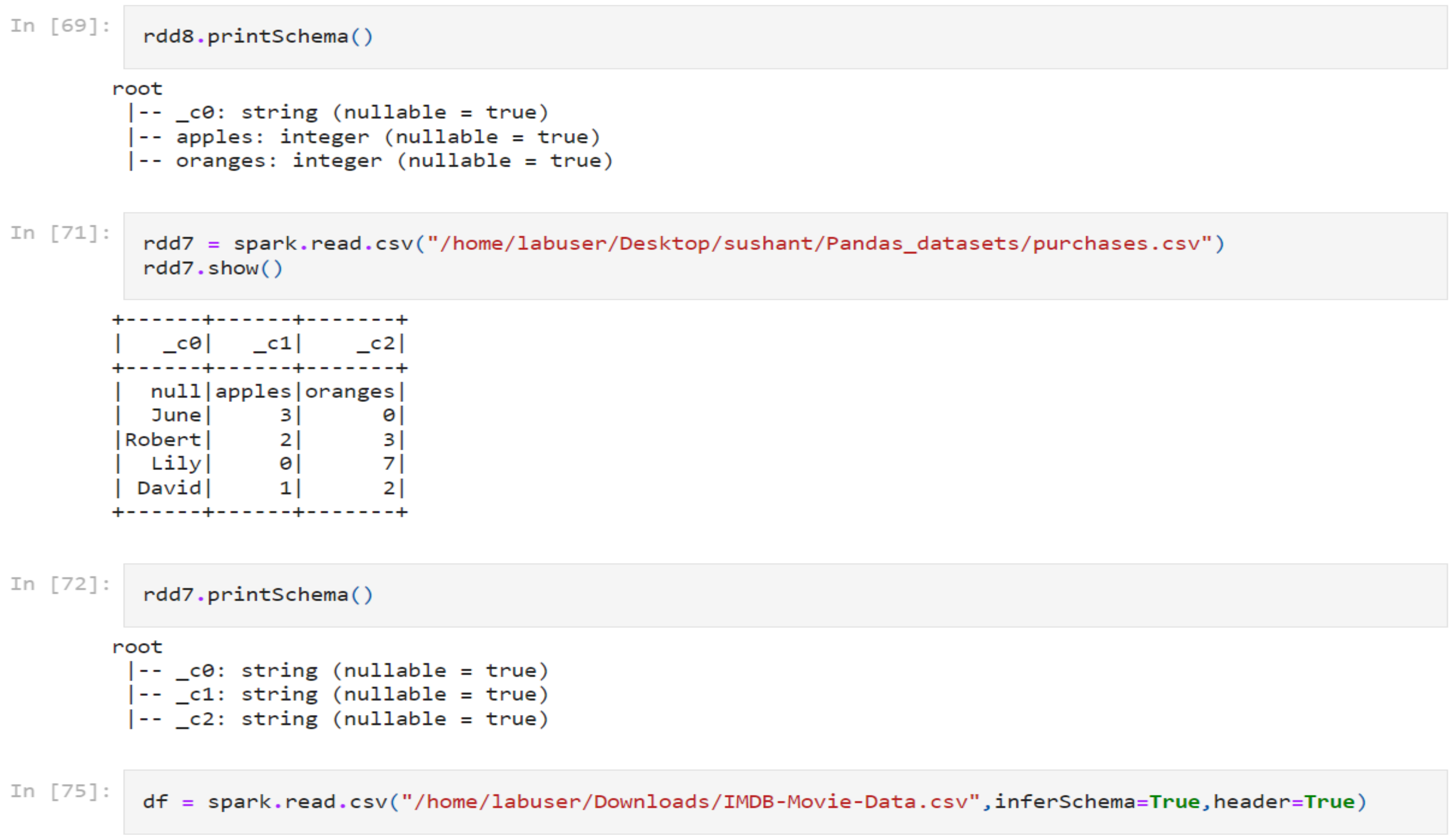
```
root
 |-- _c0: string (nullable = true)
 |-- apples: integer (nullable = true)
 |-- oranges: integer (nullable = true)
```

In [67]:
```
rdd8 = spark.read.option("inferSchema",True).option("header",True).csv("/home/labuser/Desktop/sushant
```

In [68]:
```
rdd8.show()
```

```
+------+------+-------+
|   _c0|apples|oranges|
+------+------+-------+
```

21-09-2023



In [69]:
```
rdd8.printSchema()
```

```
root
 |-- _c0: string (nullable = true)
 |-- apples: integer (nullable = true)
 |-- oranges: integer (nullable = true)
```

In [71]:
```
rdd7 = spark.read.csv("/home/labuser/Desktop/sushant/Pandas_datasets/purchases.csv")
rdd7.show()
```

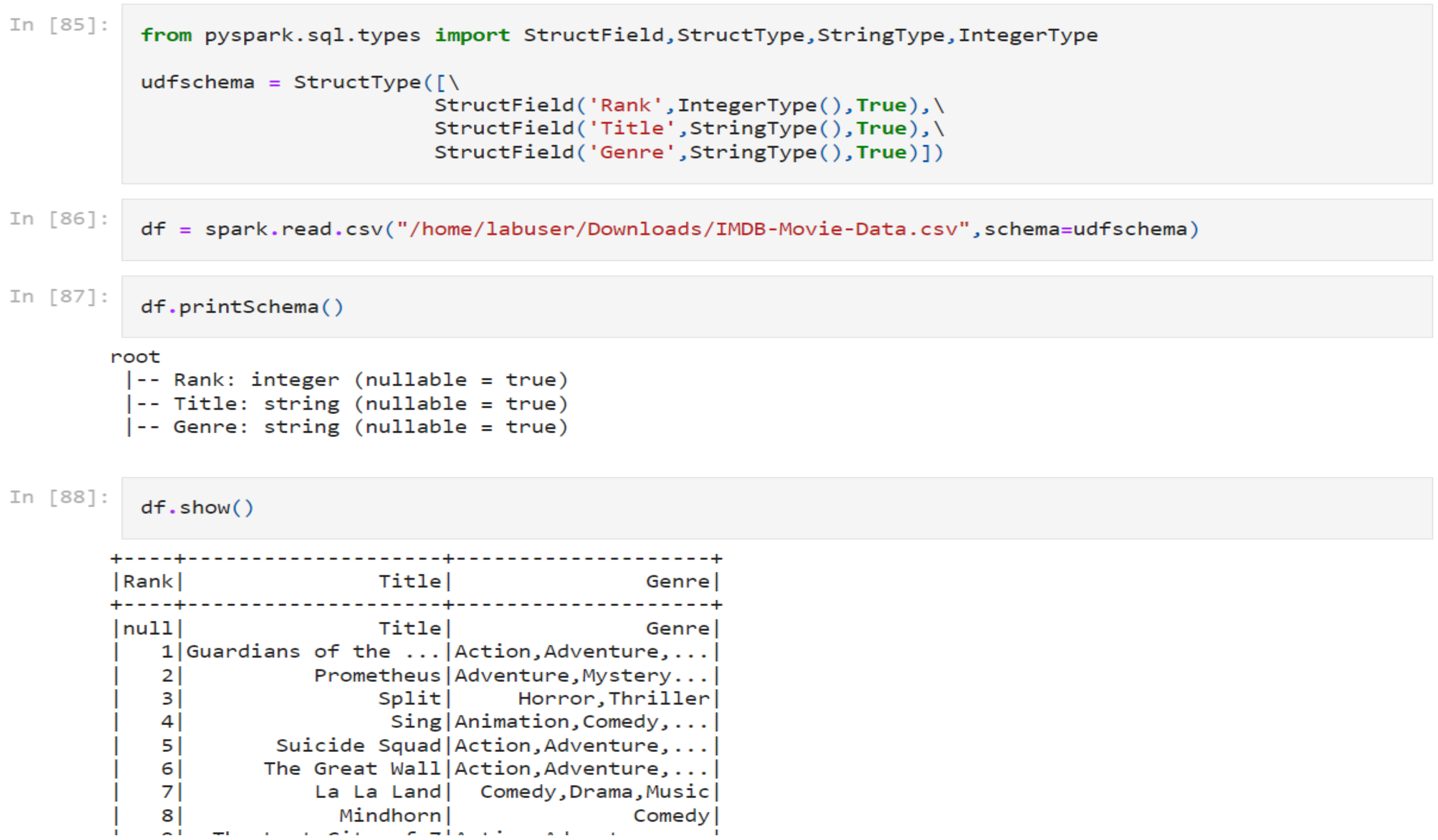
```
+------+------+-------+
|   _c0|   _c1|    _c2|
+------+------+-------+
|  null|apples|oranges|
|  June|     3|      0|
|Robert|     2|      3|
|  Lily|     0|      7|
| David|     1|      2|
+------+------+-------+
```

In [72]:
```
rdd7.printSchema()
```

```
root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
```

In [75]:
```
df = spark.read.csv("/home/labuser/Downloads/IMDB-Movie-Data.csv",inferSchema=True,header=True)
```

21-09-2023



In [85]:
```python
from pyspark.sql.types import StructField,StructType,StringType,IntegerType

udfschema = StructType([\
                        StructField('Rank',IntegerType(),True),\
                        StructField('Title',StringType(),True),\
                        StructField('Genre',StringType(),True)])
```

In [86]:
```python
df = spark.read.csv("/home/labuser/Downloads/IMDB-Movie-Data.csv",schema=udfschema)
```

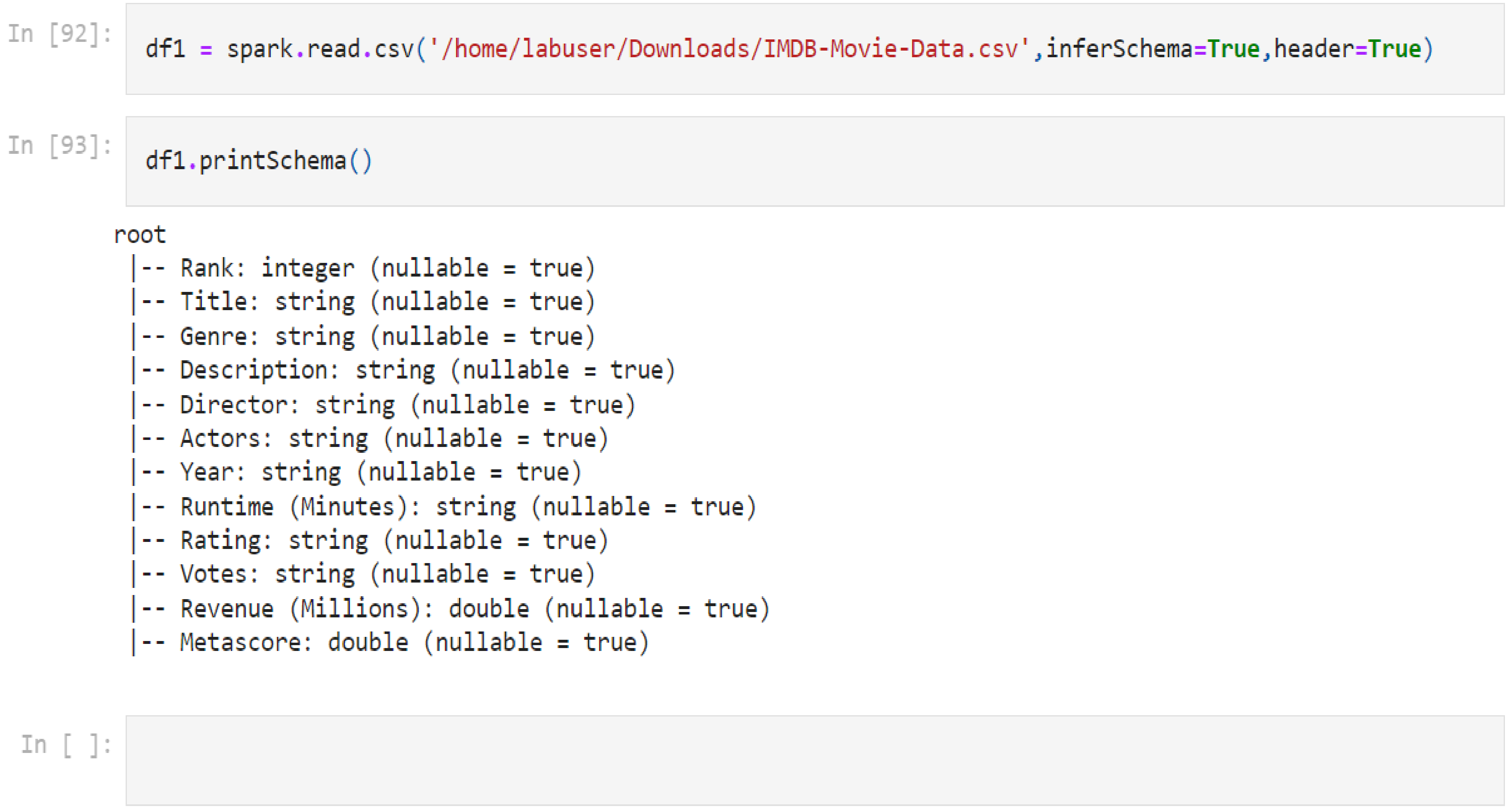
In [87]:
```python
df.printSchema()
```

```
root
 |-- Rank: integer (nullable = true)
 |-- Title: string (nullable = true)
 |-- Genre: string (nullable = true)
```

In [88]:
```python
df.show()
```

```
+----+--------------------+--------------------+
|Rank|               Title|               Genre|
+----+--------------------+--------------------+
|null|               Title|               Genre|
|   1|Guardians of the ...|Action,Adventure,...|
|   2|          Prometheus|Adventure,Mystery...|
|   3|               Split|     Horror,Thriller|
|   4|                Sing|Animation,Comedy,...|
|   5|       Suicide Squad|Action,Adventure,...|
|   6|      The Great Wall|Action,Adventure,...|
|   7|          La La Land| Comedy,Drama,Music|
|   8|            Mindhorn|              Comedy|
```

21-09-2023



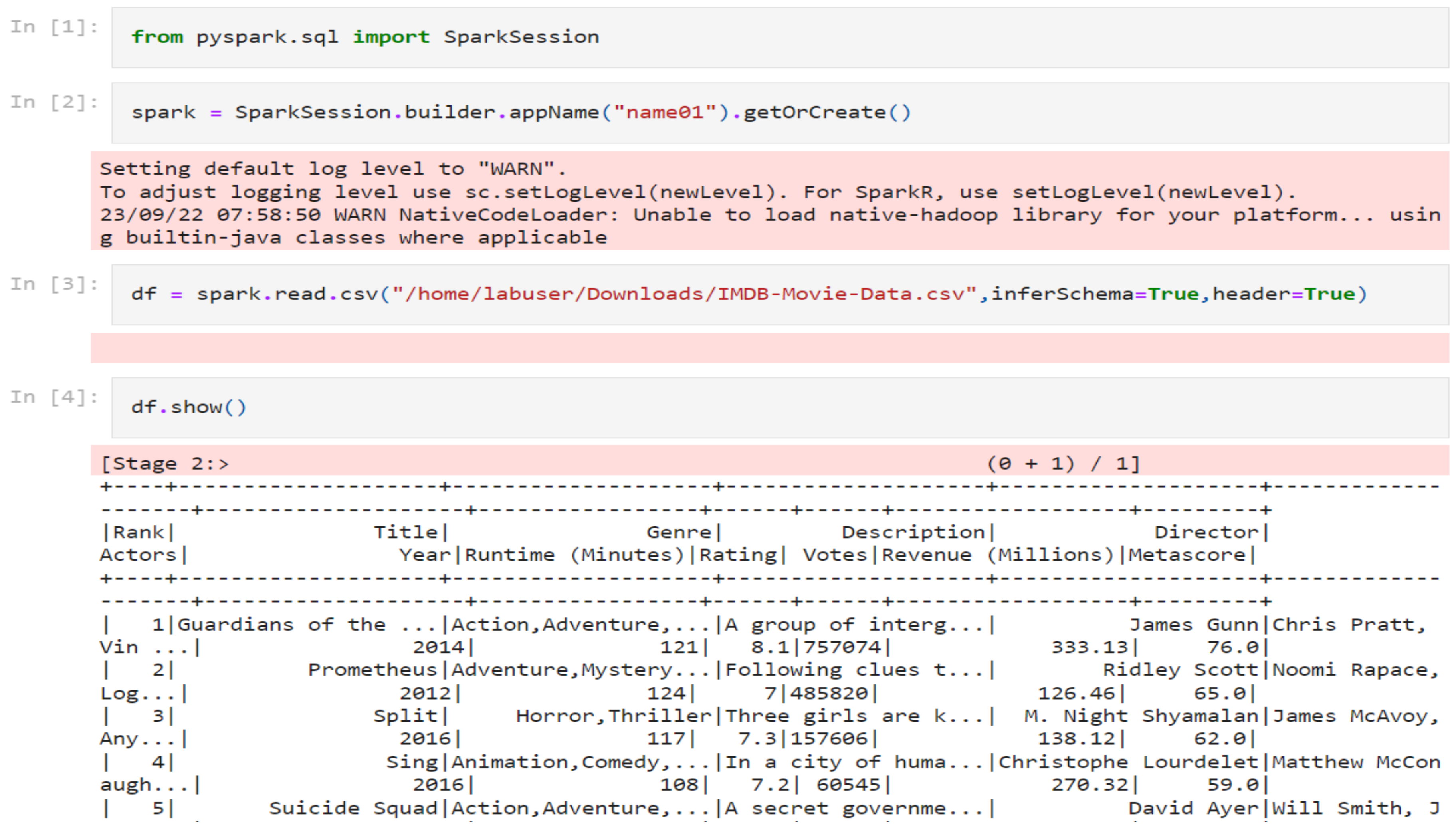
In [92]:
```python
df1 = spark.read.csv('/home/labuser/Downloads/IMDB-Movie-Data.csv',inferSchema=True,header=True)
```

In [93]:
```python
df1.printSchema()
```

```
root
 |-- Rank: integer (nullable = true)
 |-- Title: string (nullable = true)
 |-- Genre: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Director: string (nullable = true)
 |-- Actors: string (nullable = true)
 |-- Year: string (nullable = true)
 |-- Runtime (Minutes): string (nullable = true)
 |-- Rating: string (nullable = true)
 |-- Votes: string (nullable = true)
 |-- Revenue (Millions): double (nullable = true)
 |-- Metascore: double (nullable = true)
```

In [ ]:

22-09-2023



```
In [1]: from pyspark.sql import SparkSession
```

```
In [2]: spark = SparkSession.builder.appName("name01").getOrCreate()
```

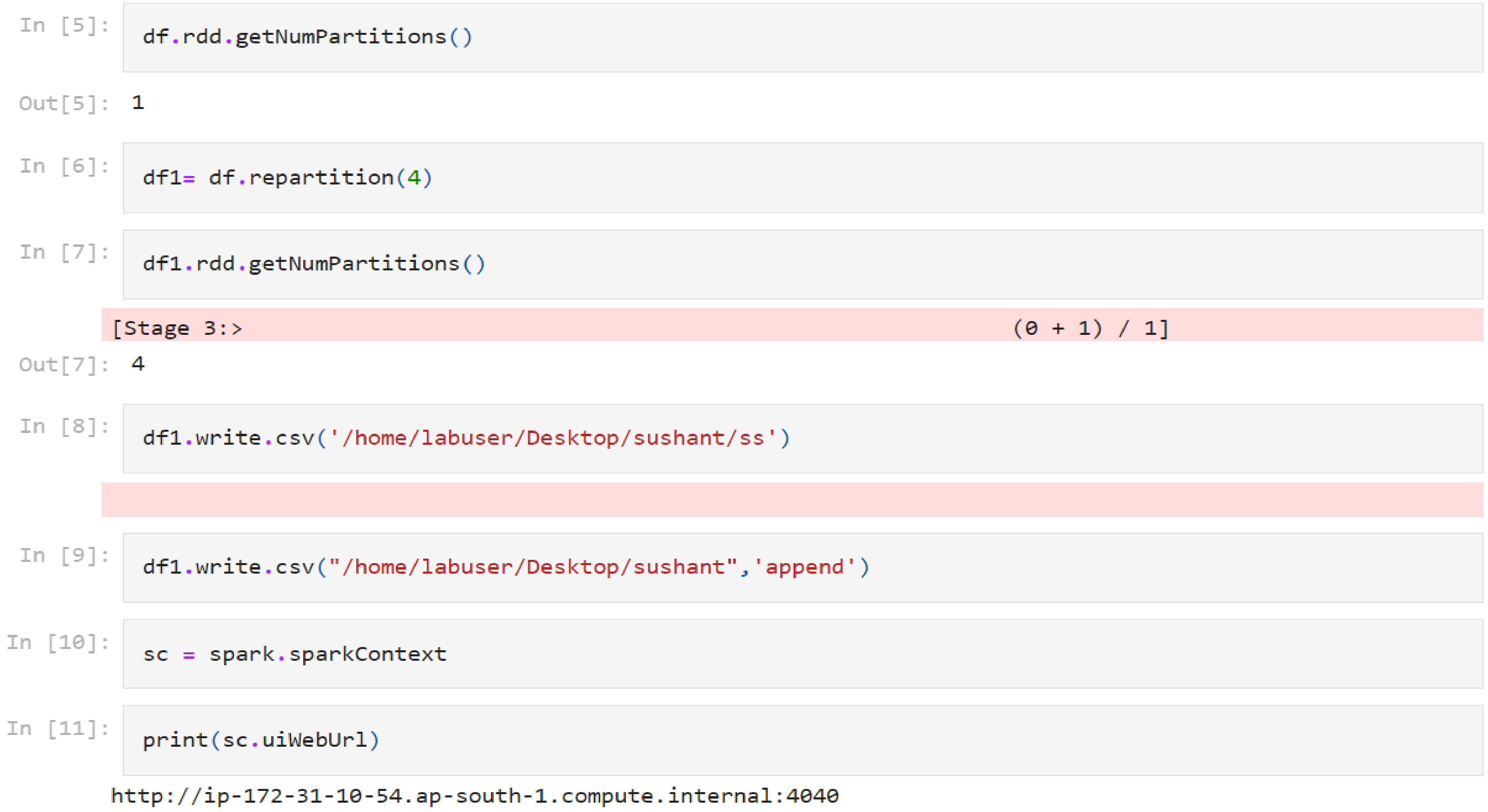
```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/09/22 07:58:50 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... usin
g builtin-java classes where applicable
```

```
In [3]: df = spark.read.csv("/home/labuser/Downloads/IMDB-Movie-Data.csv",inferSchema=True,header=True)
```

```
In [4]: df.show()
```

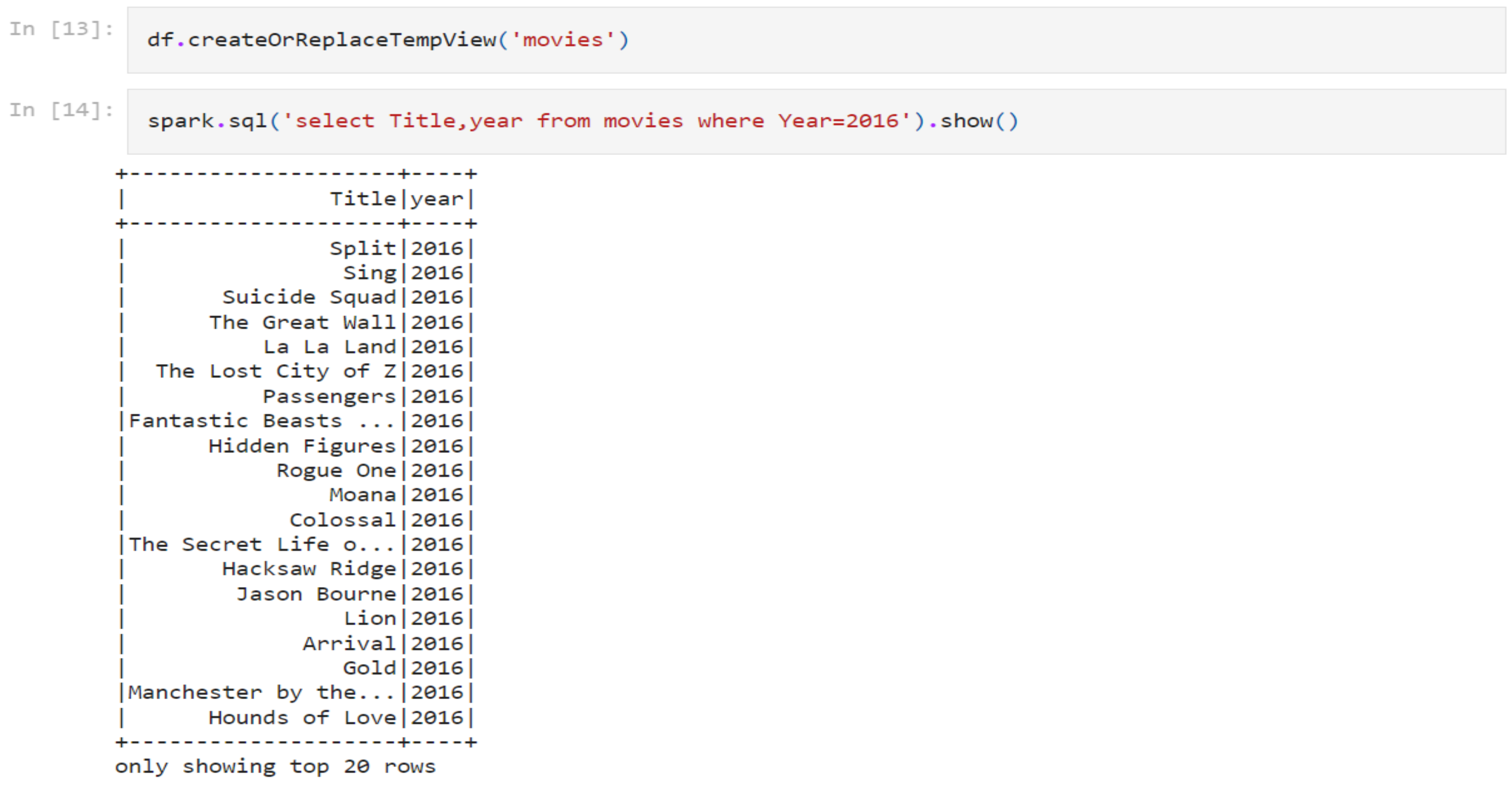
```
[Stage 2:>                                                          (0 + 1) / 1]
+----+--------------------+--------------------+--------------------+--------------------+-------------
-------+-------------------+-----------------+------+------+------------------+---------+
|Rank|               Title|               Genre|         Description|            Director|
Actors|               Year|Runtime (Minutes)|Rating| Votes|Revenue (Millions)|Metascore|
+----+--------------------+--------------------+--------------------+--------------------+-------------
-------+-------------------+-----------------+------+------+------------------+---------+
|   1|Guardians of the ...|Action,Adventure,...|A group of interg...|          James Gunn|Chris Pratt, 
Vin ...|               2014|              121|   8.1|757074|            333.13|     76.0|
|   2|          Prometheus|Adventure,Mystery...|Following clues t...|        Ridley Scott|Noomi Rapace,
Log...|               2012|              124|     7|485820|            126.46|     65.0|
|   3|               Split|     Horror,Thriller|Three girls are k...|  M. Night Shyamalan|James McAvoy,
Any...|               2016|              117|   7.3|157606|            138.12|     62.0|
|   4|                Sing|Animation,Comedy,...|In a city of huma...|Christophe Lourdelet|Matthew McCon
augh...|               2016|              108|   7.2| 60545|            270.32|     59.0|
|   5|       Suicide Squad|Action,Adventure,...|A secret governme...|          David Ayer|Will Smith, J
```

22-09-2023



```
In [5]:  df.rdd.getNumPartitions()

Out[5]:  1

In [6]:  df1= df.repartition(4)

In [7]:  df1.rdd.getNumPartitions()

         [Stage 3:>                                                        (0 + 1) / 1]
Out[7]:  4

In [8]:  df1.write.csv('/home/labuser/Desktop/sushant/ss')

In [9]:  df1.write.csv("/home/labuser/Desktop/sushant",'append')

In [10]: sc = spark.sparkContext

In [11]: print(sc.uiWebUrl)

         http://ip-172-31-10-54.ap-south-1.compute.internal:4040
```

22-09-2023

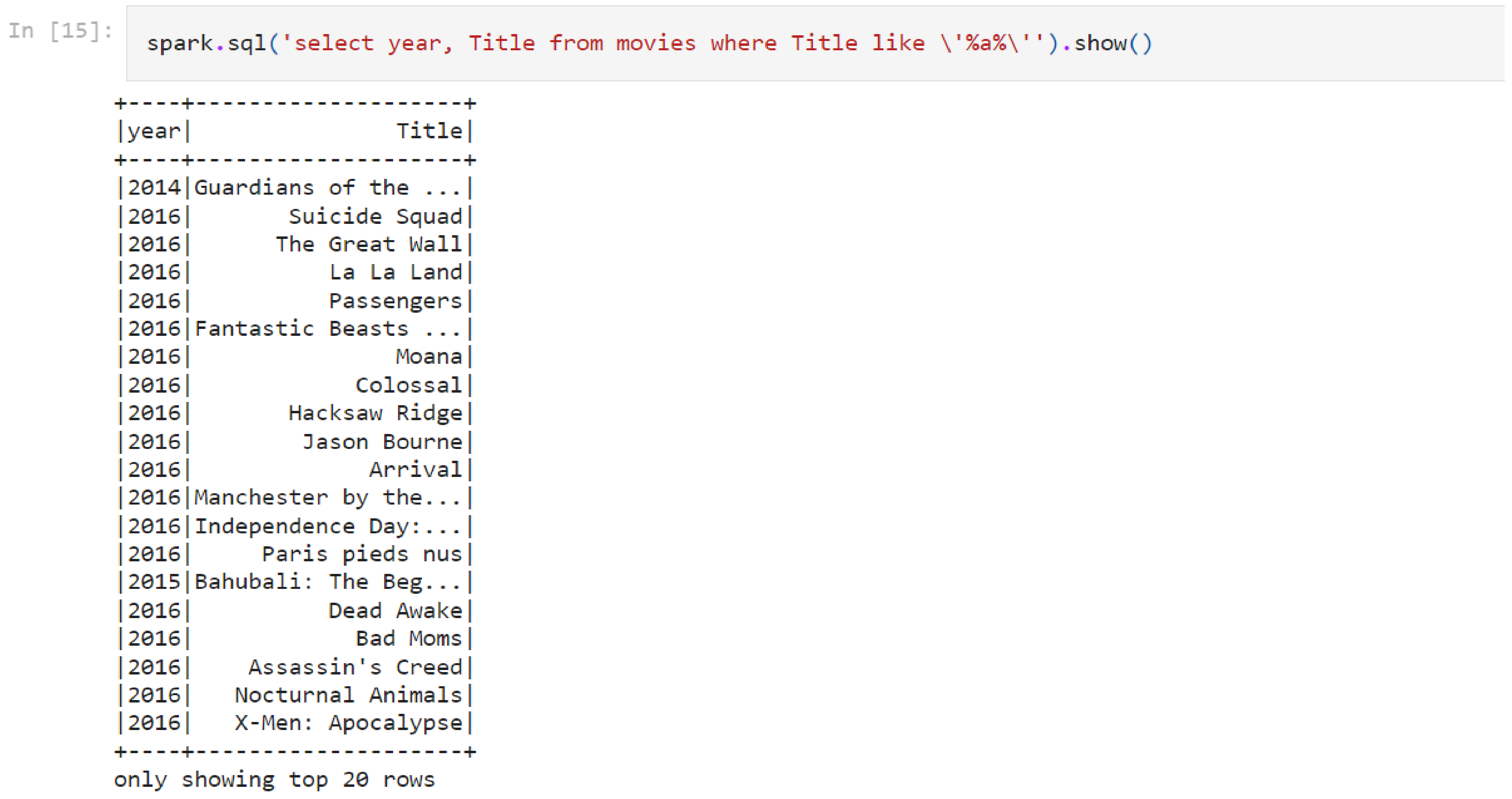


```
In [13]: df.createOrReplaceTempView('movies')
```

```
In [14]: spark.sql('select Title,year from movies where Year=2016').show()
```

```
+--------------------+----+
|               Title|year|
+--------------------+----+
|               Split|2016|
|                Sing|2016|
|       Suicide Squad|2016|
|      The Great Wall|2016|
|          La La Land|2016|
|  The Lost City of Z|2016|
|          Passengers|2016|
|Fantastic Beasts ...|2016|
|      Hidden Figures|2016|
|           Rogue One|2016|
|               Moana|2016|
|            Colossal|2016|
|The Secret Life o...|2016|
|       Hacksaw Ridge|2016|
|        Jason Bourne|2016|
|                Lion|2016|
|             Arrival|2016|
|                Gold|2016|
|Manchester by the...|2016|
|      Hounds of Love|2016|
+--------------------+----+
only showing top 20 rows
```

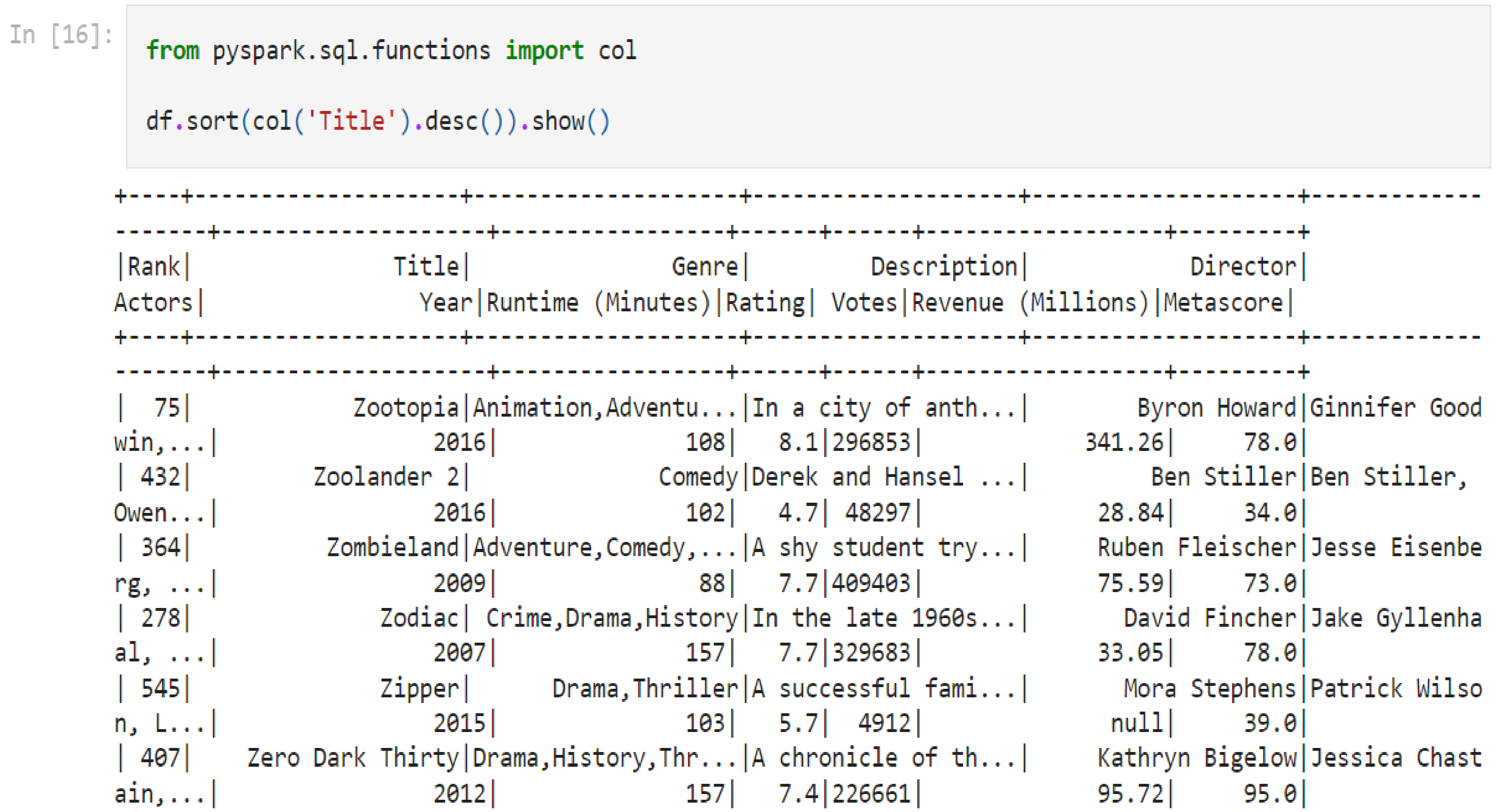
22-09-2023



```
In [15]: spark.sql('select year, Title from movies where Title like \'%a%\'').show()
```

```
+----+--------------------+
|year|               Title|
+----+--------------------+
|2014|Guardians of the ...|
|2016|       Suicide Squad|
|2016|      The Great Wall|
|2016|          La La Land|
|2016|          Passengers|
|2016|Fantastic Beasts ...|
|2016|               Moana|
|2016|            Colossal|
|2016|       Hacksaw Ridge|
|2016|        Jason Bourne|
|2016|             Arrival|
|2016|Manchester by the...|
|2016|Independence Day:...|
|2016|      Paris pieds nus|
|2015|Bahubali: The Beg...|
|2016|          Dead Awake|
|2016|            Bad Moms|
|2016|      Assassin's Creed|
|2016|     Nocturnal Animals|
|2016|     X-Men: Apocalypse|
+----+--------------------+
only showing top 20 rows
```

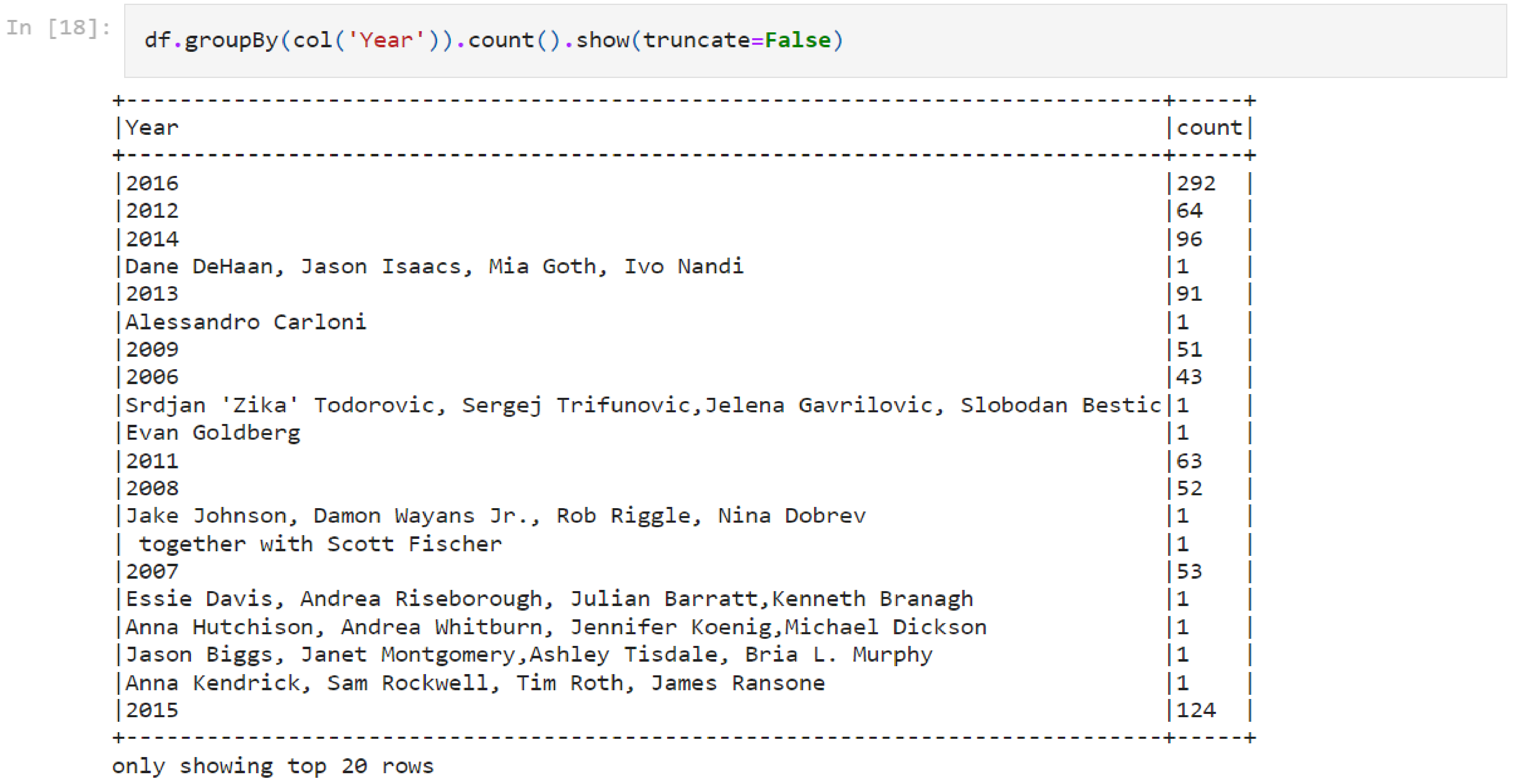
22-09-2023



In [16]:
```
from pyspark.sql.functions import col

df.sort(col('Title').desc()).show()
```

```
+----+--------------------+--------------------+--------------------+--------------------+-------------
-------+--------------------+-----------------+------+------+------------------+---------+
|Rank|               Title|               Genre|         Description|            Director|
Actors|                Year|Runtime (Minutes)|Rating| Votes|Revenue (Millions)|Metascore|
+----+--------------------+--------------------+--------------------+--------------------+-------------
-------+--------------------+-----------------+------+------+------------------+---------+
|  75|            Zootopia|Animation,Adventu...|In a city of anth...|        Byron Howard|Ginnifer Good
win,...|                2016|              108|   8.1|296853|            341.26|     78.0|
| 432|         Zoolander 2|              Comedy|Derek and Hansel ...|         Ben Stiller|Ben Stiller,
Owen...|                2016|              102|   4.7| 48297|             28.84|     34.0|
| 364|          Zombieland|Adventure,Comedy,...|A shy student try...|     Ruben Fleischer|Jesse Eisenbe
rg, ...|                2009|               88|   7.7|409403|             75.59|     73.0|
| 278|              Zodiac| Crime,Drama,History|In the late 1960s...|       David Fincher|Jake Gyllenha
al, ...|                2007|              157|   7.7|329683|             33.05|     78.0|
| 545|              Zipper|      Drama,Thriller|A successful fami...|       Mora Stephens|Patrick Wilso
n, L...|                2015|              103|   5.7|  4912|              null|     39.0|
| 407|    Zero Dark Thirty|Drama,History,Thr...|A chronicle of th...|     Kathryn Bigelow|Jessica Chast
ain,...|                2012|              157|   7.4|226661|             95.72|     95.0|
```

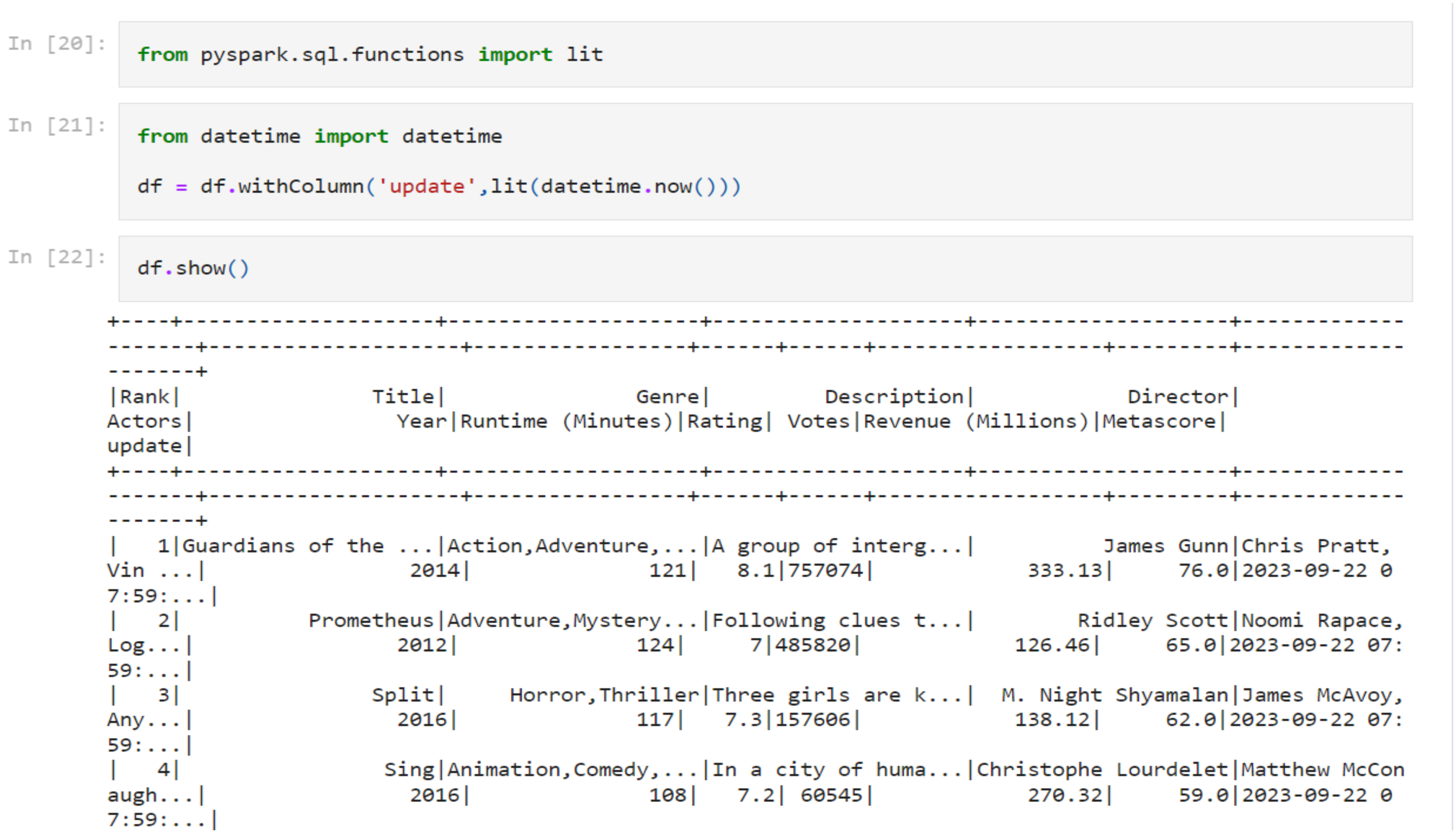
22-09-2023



```
In [18]: df.groupBy(col('Year')).count().show(truncate=False)
```

```
+-----------------------------------------------------------------------------+-----+
|Year                                                                         |count|
+-----------------------------------------------------------------------------+-----+
|2016                                                                         |292  |
|2012                                                                         |64   |
|2014                                                                         |96   |
|Dane DeHaan, Jason Isaacs, Mia Goth, Ivo Nandi                               |1    |
|2013                                                                         |91   |
|Alessandro Carloni                                                           |1    |
|2009                                                                         |51   |
|2006                                                                         |43   |
|Srdjan 'Zika' Todorovic, Sergej Trifunovic,Jelena Gavrilovic, Slobodan Bestic|1    |
|Evan Goldberg                                                                |1    |
|2011                                                                         |63   |
|2008                                                                         |52   |
|Jake Johnson, Damon Wayans Jr., Rob Riggle, Nina Dobrev                      |1    |
| together with Scott Fischer                                                 |1    |
|2007                                                                         |53   |
|Essie Davis, Andrea Riseborough, Julian Barratt,Kenneth Branagh              |1    |
|Anna Hutchison, Andrea Whitburn, Jennifer Koenig,Michael Dickson             |1    |
|Jason Biggs, Janet Montgomery,Ashley Tisdale, Bria L. Murphy                 |1    |
|Anna Kendrick, Sam Rockwell, Tim Roth, James Ransone                         |1    |
|2015                                                                         |124  |
+-----------------------------------------------------------------------------+-----+
only showing top 20 rows
```

22-09-2023



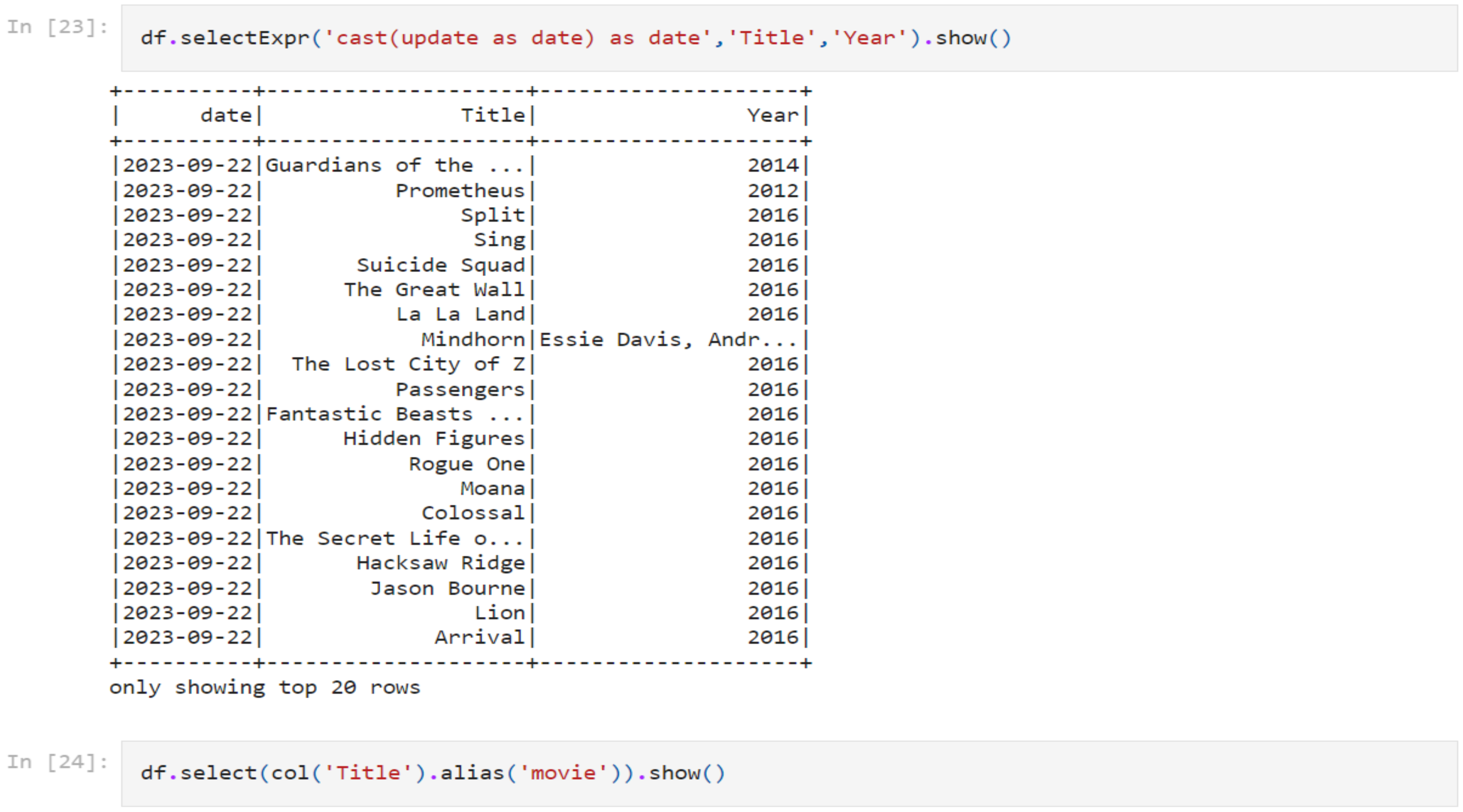
```
In [20]: from pyspark.sql.functions import lit
```

```
In [21]: from datetime import datetime

         df = df.withColumn('update',lit(datetime.now()))
```

```
In [22]: df.show()
```

```
+----+--------------------+--------------------+--------------------+--------------------+------------
-------+--------------------+-----------------+------+------+------------------+---------+------------
-------+
|Rank|               Title|               Genre|         Description|            Director|
Actors|               Year|Runtime (Minutes)|Rating| Votes|Revenue (Millions)|Metascore|
update|
+----+--------------------+--------------------+--------------------+--------------------+------------
-------+--------------------+-----------------+------+------+------------------+---------+------------
-------+
|   1|Guardians of the ...|Action,Adventure,...|A group of interg...|          James Gunn|Chris Pratt,
Vin ...|                2014|                121|   8.1|757074|            333.13|     76.0|2023-09-22 0
7:59:...|
|   2|          Prometheus|Adventure,Mystery...|Following clues t...|        Ridley Scott|Noomi Rapace,
Log...|               2012|                124|      7|485820|            126.46|     65.0|2023-09-22 07:
59:...|
|   3|               Split|     Horror,Thriller|Three girls are k...|  M. Night Shyamalan|James McAvoy,
Any...|               2016|                117|   7.3|157606|            138.12|     62.0|2023-09-22 07:
59:...|
|   4|                Sing|Animation,Comedy,...|In a city of huma...|Christophe Lourdelet|Matthew McCon
augh...|                2016|                108|   7.2| 60545|            270.32|     59.0|2023-09-22 0
7:59:...|
```

22-09-2023

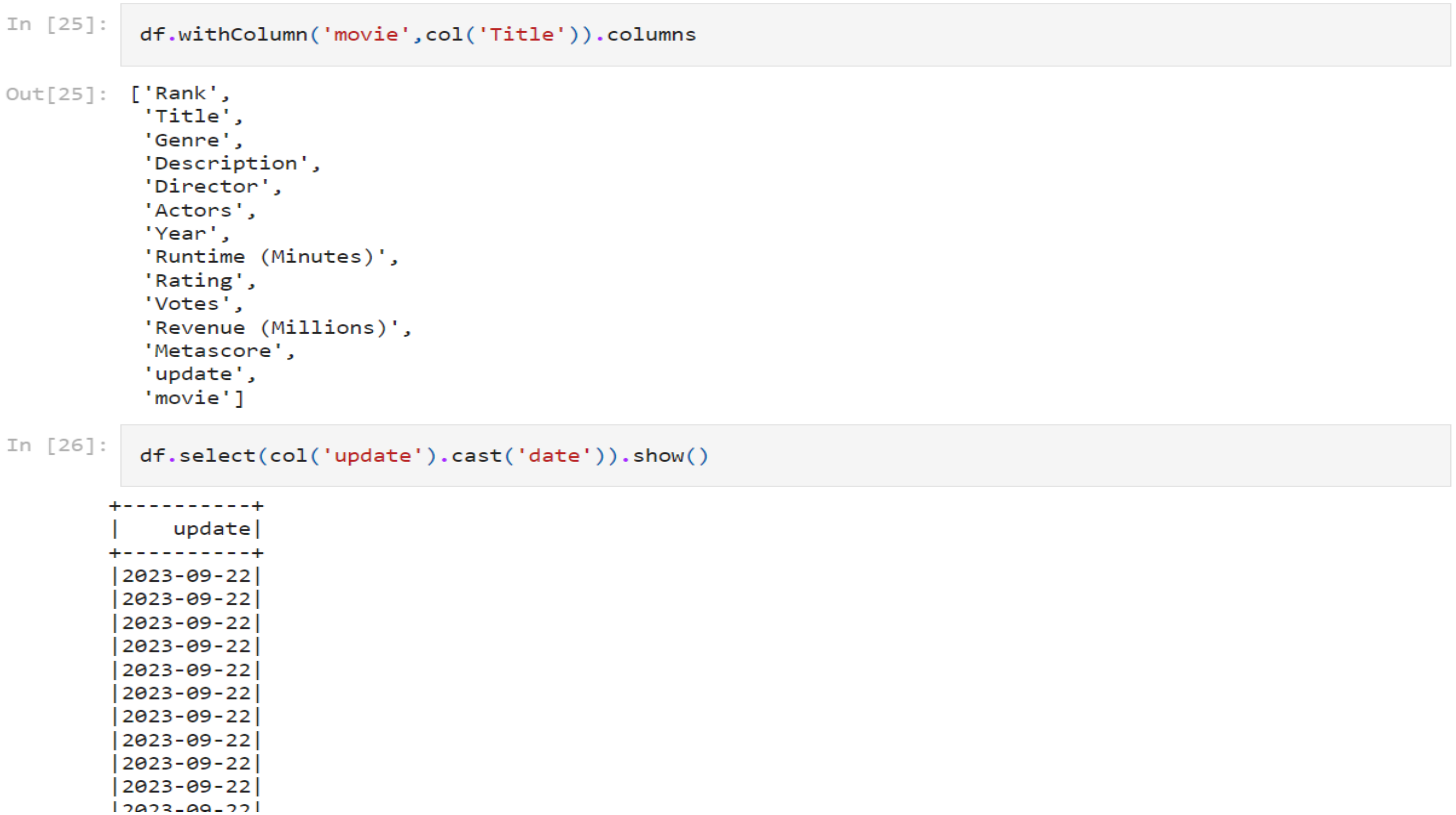


```
In [23]: df.selectExpr('cast(update as date) as date','Title','Year').show()
```

```
+----------+--------------------+--------------------+
|      date|               Title|                Year|
+----------+--------------------+--------------------+
|2023-09-22|Guardians of the ...|                2014|
|2023-09-22|          Prometheus|                2012|
|2023-09-22|               Split|                2016|
|2023-09-22|                Sing|                2016|
|2023-09-22|       Suicide Squad|                2016|
|2023-09-22|      The Great Wall|                2016|
|2023-09-22|          La La Land|                2016|
|2023-09-22|            Mindhorn|Essie Davis, Andr...|
|2023-09-22|   The Lost City of Z|                2016|
|2023-09-22|          Passengers|                2016|
|2023-09-22|Fantastic Beasts ...|                2016|
|2023-09-22|      Hidden Figures|                2016|
|2023-09-22|           Rogue One|                2016|
|2023-09-22|               Moana|                2016|
|2023-09-22|            Colossal|                2016|
|2023-09-22|The Secret Life o...|                2016|
|2023-09-22|       Hacksaw Ridge|                2016|
|2023-09-22|        Jason Bourne|                2016|
|2023-09-22|                Lion|                2016|
|2023-09-22|             Arrival|                2016|
+----------+--------------------+--------------------+
only showing top 20 rows
```

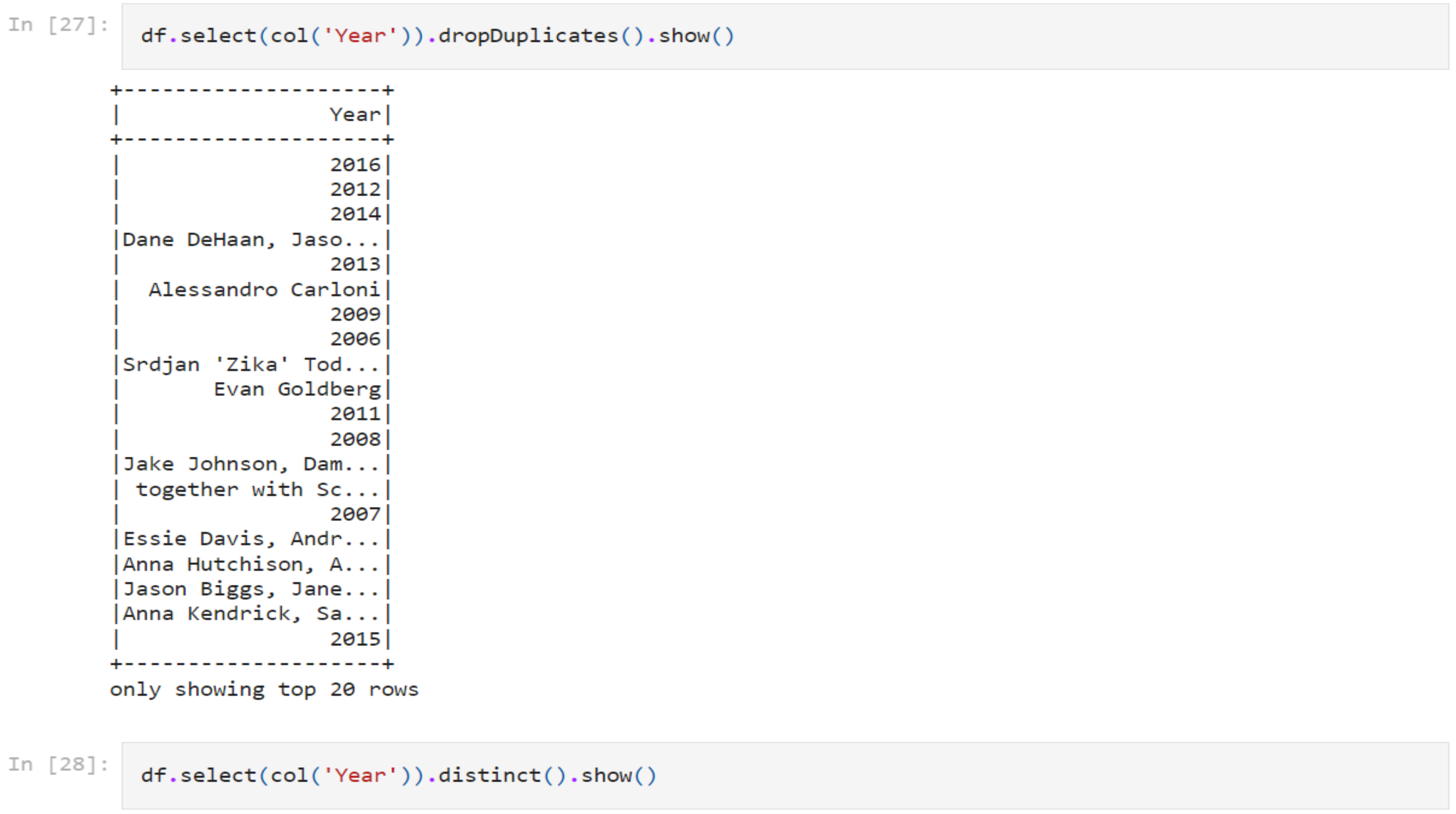
```
In [24]: df.select(col('Title').alias('movie')).show()
```

22-09-2023



```
In [25]: df.withColumn('movie',col('Title')).columns

Out[25]: ['Rank',
          'Title',
          'Genre',
          'Description',
          'Director',
          'Actors',
          'Year',
          'Runtime (Minutes)',
          'Rating',
          'Votes',
          'Revenue (Millions)',
          'Metascore',
          'update',
          'movie']

In [26]: df.select(col('update').cast('date')).show()

+----------+
|    update|
+----------+
|2023-09-22|
|2023-09-22|
|2023-09-22|
|2023-09-22|
|2023-09-22|
|2023-09-22|
|2023-09-22|
|2023-09-22|
|2023-09-22|
|2023-09-22|
```

22-09-2023

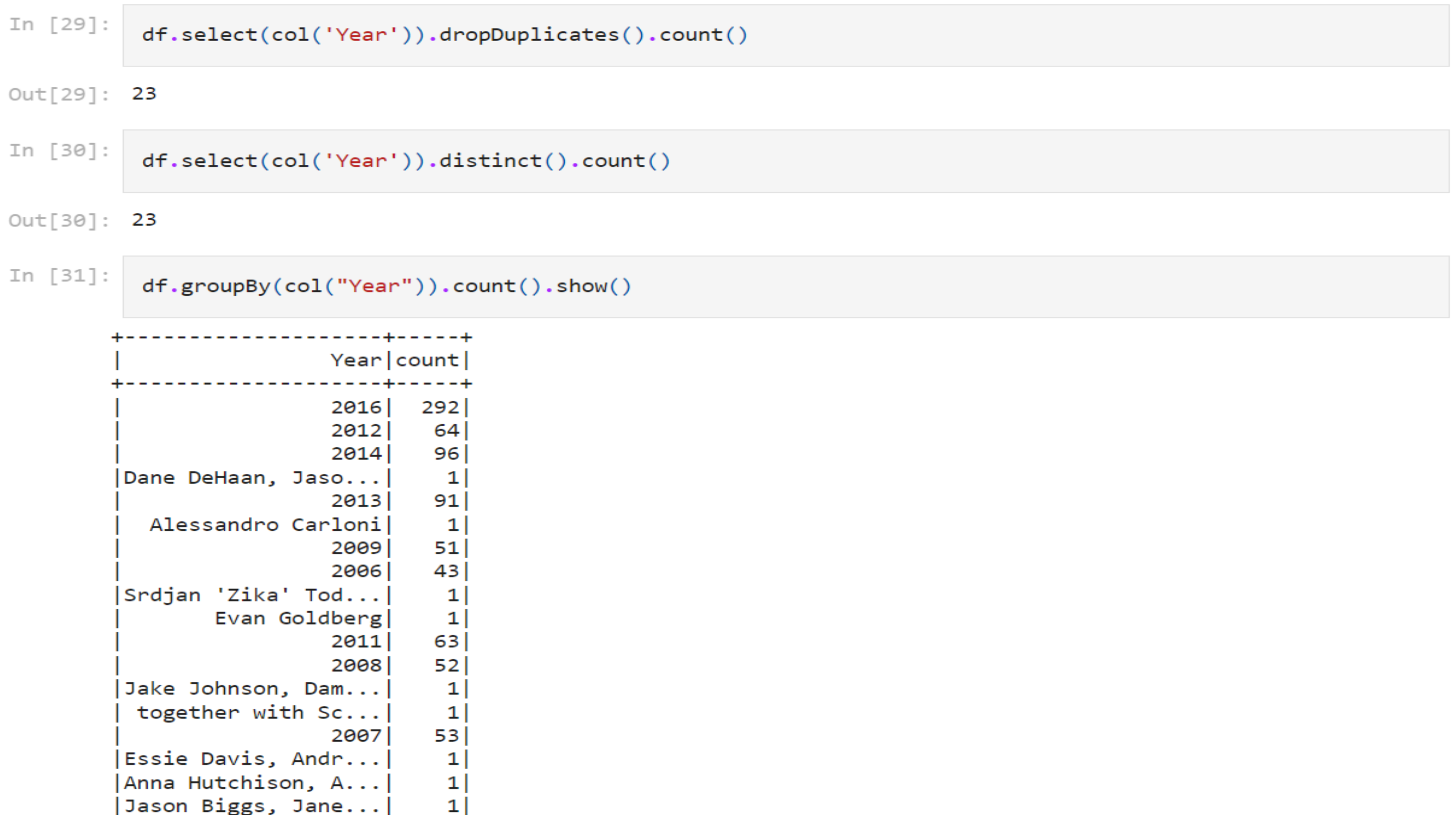


```
In [27]: df.select(col('Year')).dropDuplicates().show()
```

```
+--------------------+
|                Year|
+--------------------+
|                2016|
|                2012|
|                2014|
|Dane DeHaan, Jaso...|
|                2013|
|  Alessandro Carloni|
|                2009|
|                2006|
|Srdjan 'Zika' Tod...|
|       Evan Goldberg|
|                2011|
|                2008|
|Jake Johnson, Dam...|
| together with Sc...|
|                2007|
|Essie Davis, Andr...|
|Anna Hutchison, A...|
|Jason Biggs, Jane...|
|Anna Kendrick, Sa...|
|                2015|
+--------------------+
only showing top 20 rows
```

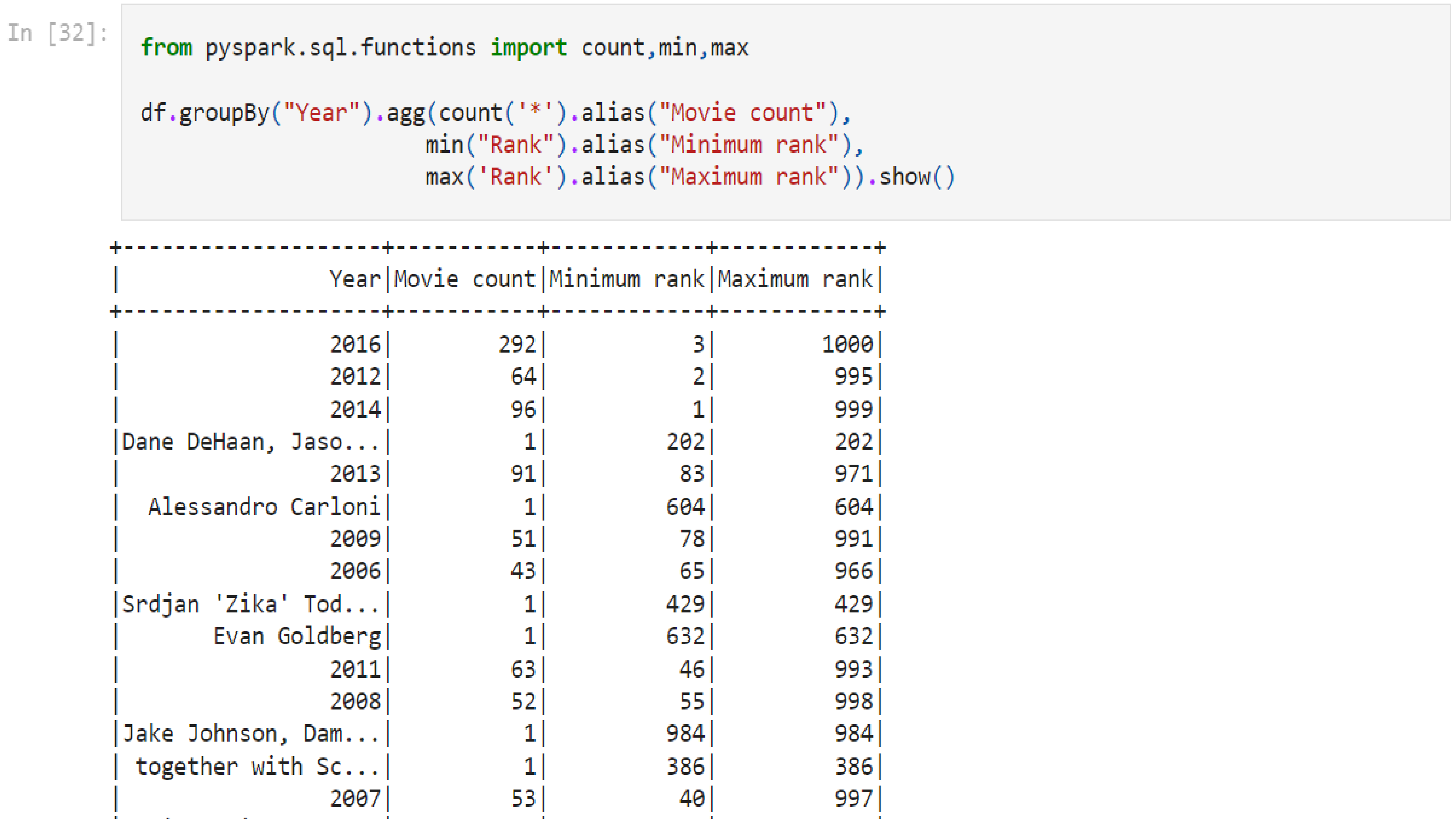
```
In [28]: df.select(col('Year')).distinct().show()
```

22-09-2023



```
In [29]: df.select(col('Year')).dropDuplicates().count()

Out[29]: 23

In [30]: df.select(col('Year')).distinct().count()

Out[30]: 23

In [31]: df.groupBy(col("Year")).count().show()

+--------------------+-----+
|                Year|count|
+--------------------+-----+
|                2016|  292|
|                2012|   64|
|                2014|   96|
|Dane DeHaan, Jaso...|    1|
|                2013|   91|
|  Alessandro Carloni|    1|
|                2009|   51|
|                2006|   43|
|Srdjan 'Zika' Tod...|    1|
|       Evan Goldberg|    1|
|                2011|   63|
|                2008|   52|
|Jake Johnson, Dam...|    1|
| together with Sc...|    1|
|                2007|   53|
|Essie Davis, Andr...|    1|
|Anna Hutchison, A...|    1|
|Jason Biggs, Jane...|    1|
```

22-09-2023

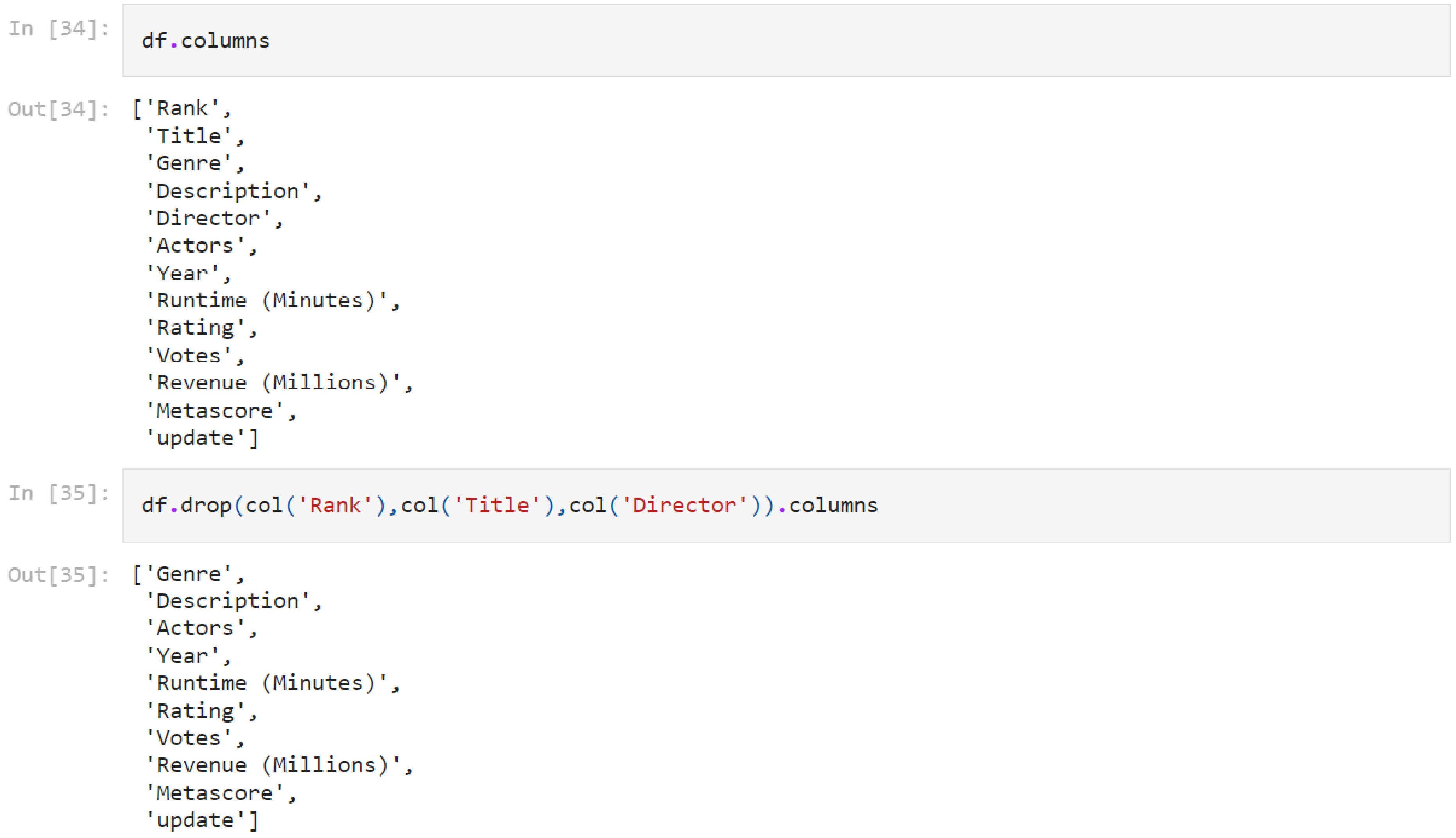


In [32]:

```
from pyspark.sql.functions import count,min,max

df.groupBy("Year").agg(count('*').alias("Movie count"),
                      min("Rank").alias("Minimum rank"),
                      max('Rank').alias("Maximum rank")).show()
```

```
+--------------------+-----------+------------+------------+
|                Year|Movie count|Minimum rank|Maximum rank|
+--------------------+-----------+------------+------------+
|                2016|        292|           3|        1000|
|                2012|         64|           2|         995|
|                2014|         96|           1|         999|
|Dane DeHaan, Jaso...|          1|         202|         202|
|                2013|         91|          83|         971|
|  Alessandro Carloni|          1|         604|         604|
|                2009|         51|          78|         991|
|                2006|         43|          65|         966|
|Srdjan 'Zika' Tod...|          1|         429|         429|
|        Evan Goldberg|          1|         632|         632|
|                2011|         63|          46|         993|
|                2008|         52|          55|         998|
|Jake Johnson, Dam...|          1|         984|         984|
| together with Sc...|          1|         386|         386|
|                2007|         53|          40|         997|
```

22-09-2023



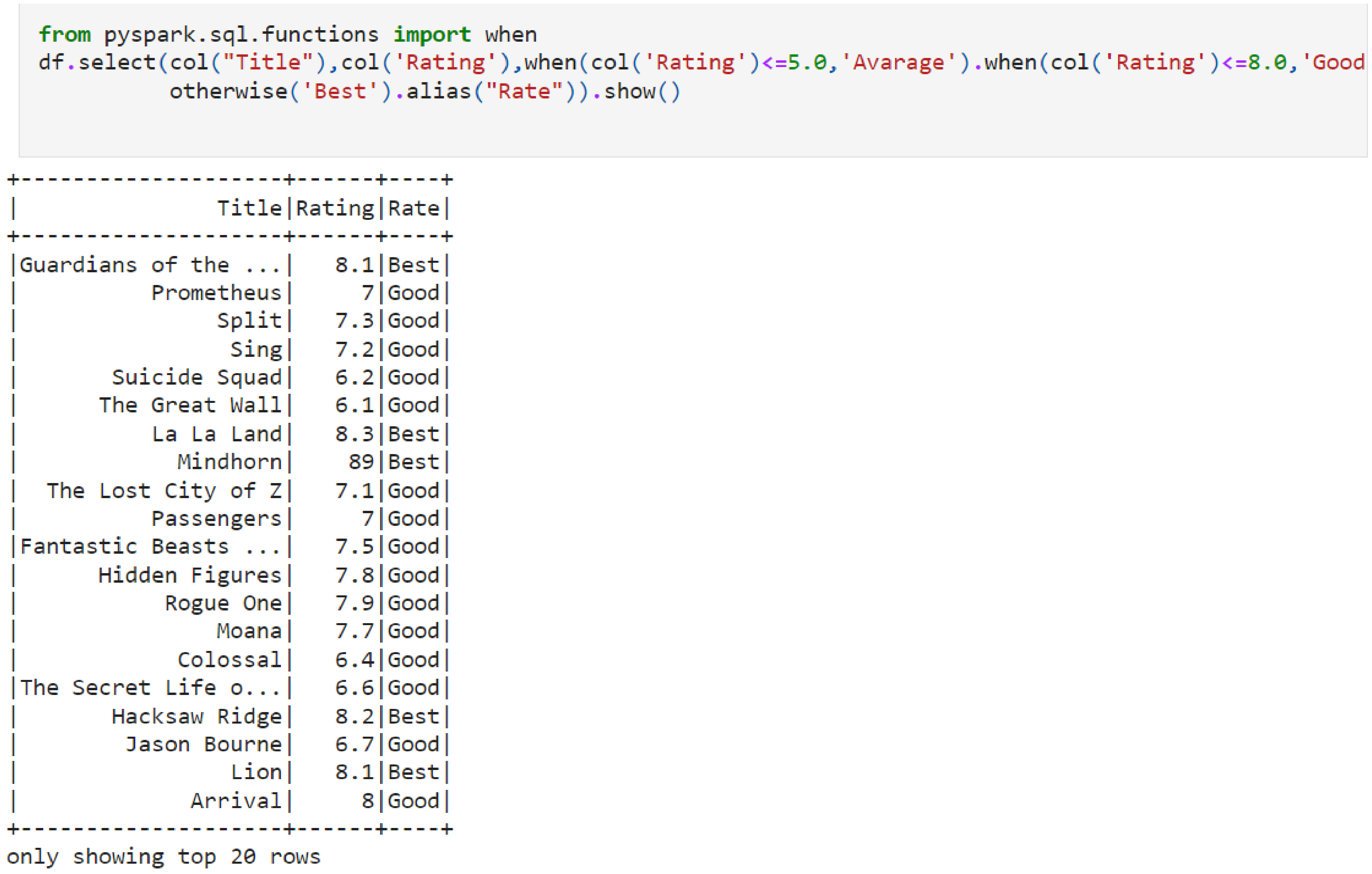
```
In [34]: df.columns
```

```
Out[34]: ['Rank',
          'Title',
          'Genre',
          'Description',
          'Director',
          'Actors',
          'Year',
          'Runtime (Minutes)',
          'Rating',
          'Votes',
          'Revenue (Millions)',
          'Metascore',
          'update']
```

```
In [35]: df.drop(col('Rank'),col('Title'),col('Director')).columns
```

```
Out[35]: ['Genre',
          'Description',
          'Actors',
          'Year',
          'Runtime (Minutes)',
          'Rating',
          'Votes',
          'Revenue (Millions)',
          'Metascore',
          'update']
```

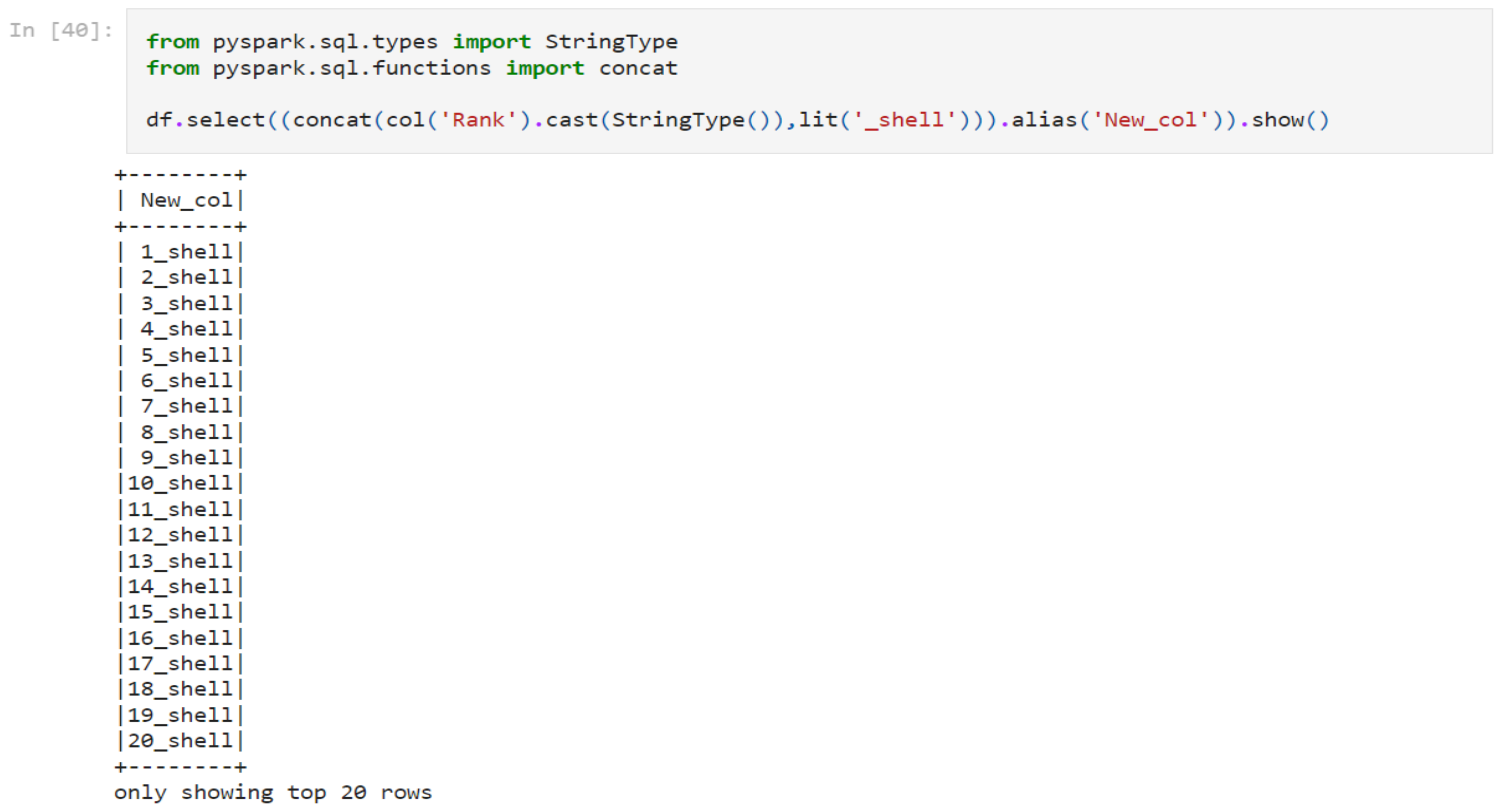
22-09-2023



```
from pyspark.sql.functions import when
df.select(col("Title"),col('Rating'),when(col('Rating')<=5.0,'Avarage').when(col('Rating')<=8.0,'Good
          otherwise('Best').alias("Rate")).show()
```

```
+--------------------+------+----+
|               Title|Rating|Rate|
+--------------------+------+----+
|Guardians of the ...|   8.1|Best|
|          Prometheus|     7|Good|
|               Split|   7.3|Good|
|                Sing|   7.2|Good|
|       Suicide Squad|   6.2|Good|
|      The Great Wall|   6.1|Good|
|          La La Land|   8.3|Best|
|            Mindhorn|    89|Best|
|    The Lost City of Z|   7.1|Good|
|          Passengers|     7|Good|
|Fantastic Beasts ...|   7.5|Good|
|      Hidden Figures|   7.8|Good|
|           Rogue One|   7.9|Good|
|               Moana|   7.7|Good|
|            Colossal|   6.4|Good|
|The Secret Life o...|   6.6|Good|
|       Hacksaw Ridge|   8.2|Best|
|        Jason Bourne|   6.7|Good|
|                Lion|   8.1|Best|
|             Arrival|     8|Good|
+--------------------+------+----+
only showing top 20 rows
```

22-09-2023

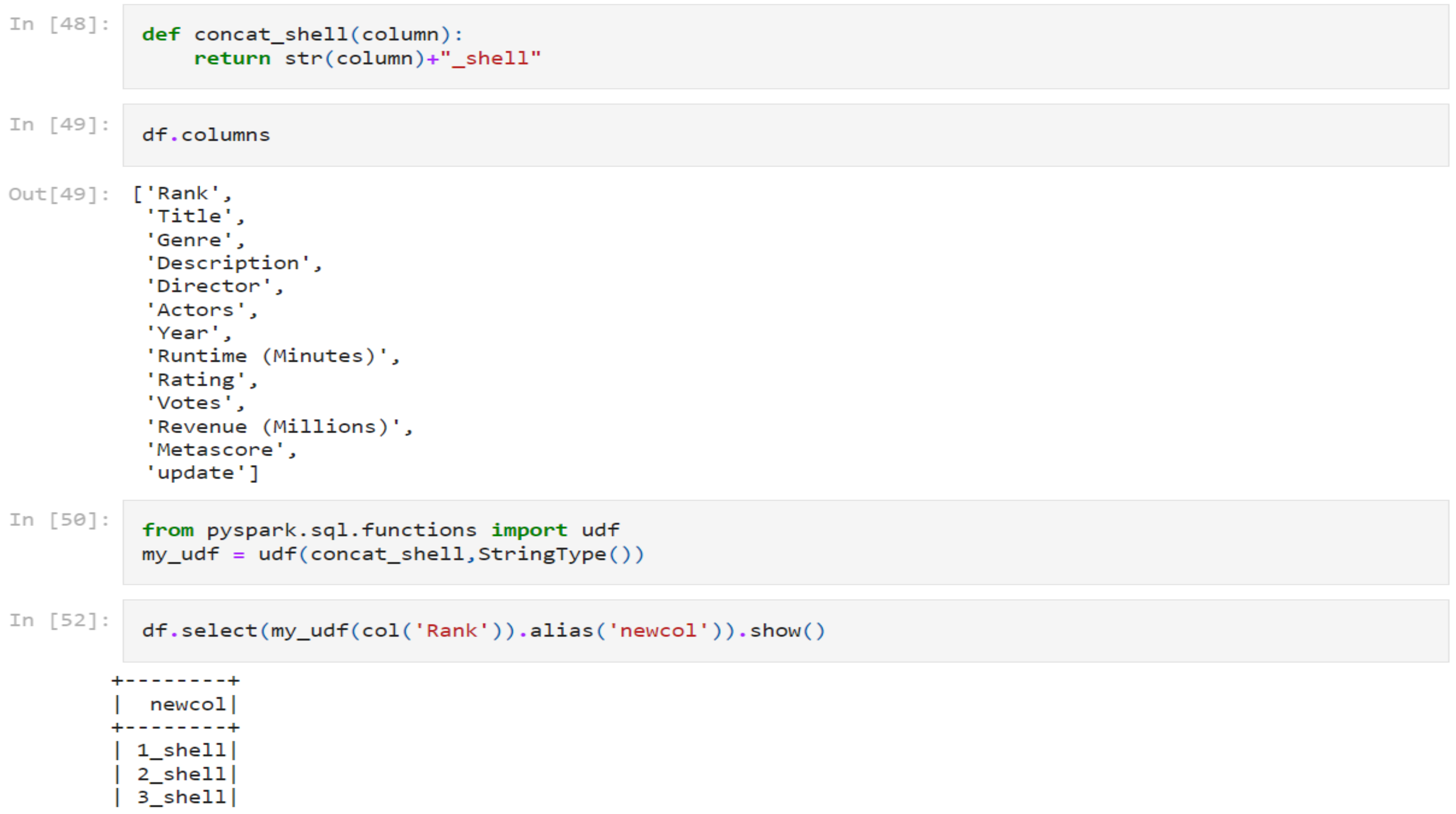


In [40]:

```
from pyspark.sql.types import StringType
from pyspark.sql.functions import concat

df.select((concat(col('Rank').cast(StringType()),lit('_shell'))).alias('New_col')).show()
```

```
+--------+
| New_col|
+--------+
| 1_shell|
| 2_shell|
| 3_shell|
| 4_shell|
| 5_shell|
| 6_shell|
| 7_shell|
| 8_shell|
| 9_shell|
|10_shell|
|11_shell|
|12_shell|
|13_shell|
|14_shell|
|15_shell|
|16_shell|
|17_shell|
|18_shell|
|19_shell|
|20_shell|
+--------+
only showing top 20 rows
```

22-09-2023



```
In [48]: def concat_shell(column):
             return str(column)+"_shell"
```

```
In [49]: df.columns
```

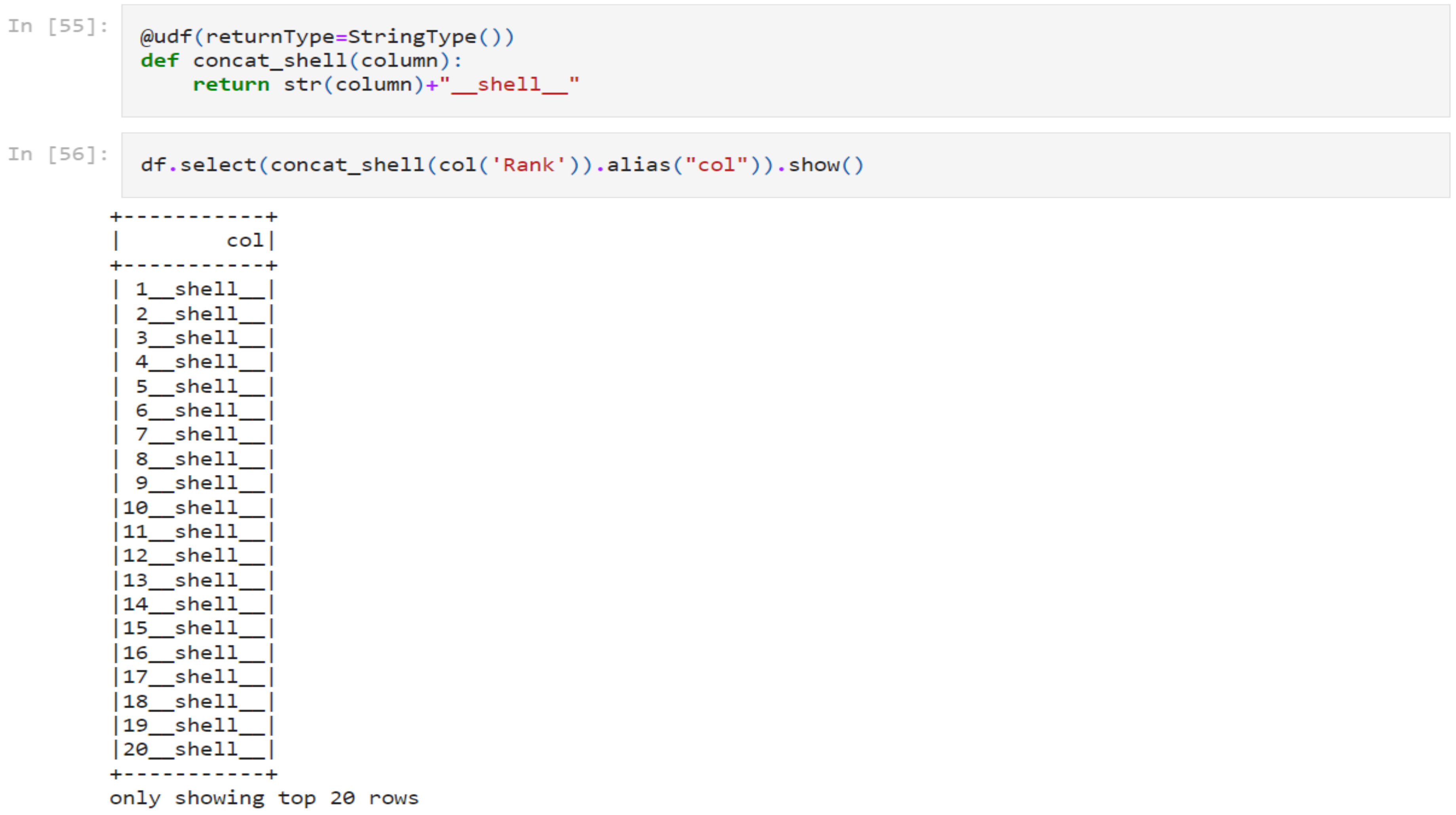
```
Out[49]: ['Rank',
          'Title',
          'Genre',
          'Description',
          'Director',
          'Actors',
          'Year',
          'Runtime (Minutes)',
          'Rating',
          'Votes',
          'Revenue (Millions)',
          'Metascore',
          'update']
```

```
In [50]: from pyspark.sql.functions import udf
         my_udf = udf(concat_shell,StringType())
```

```
In [52]: df.select(my_udf(col('Rank')).alias('newcol')).show()
```

```
+---------+
|   newcol|
+---------+
|  1_shell|
|  2_shell|
|  3_shell|
```

22-09-2023

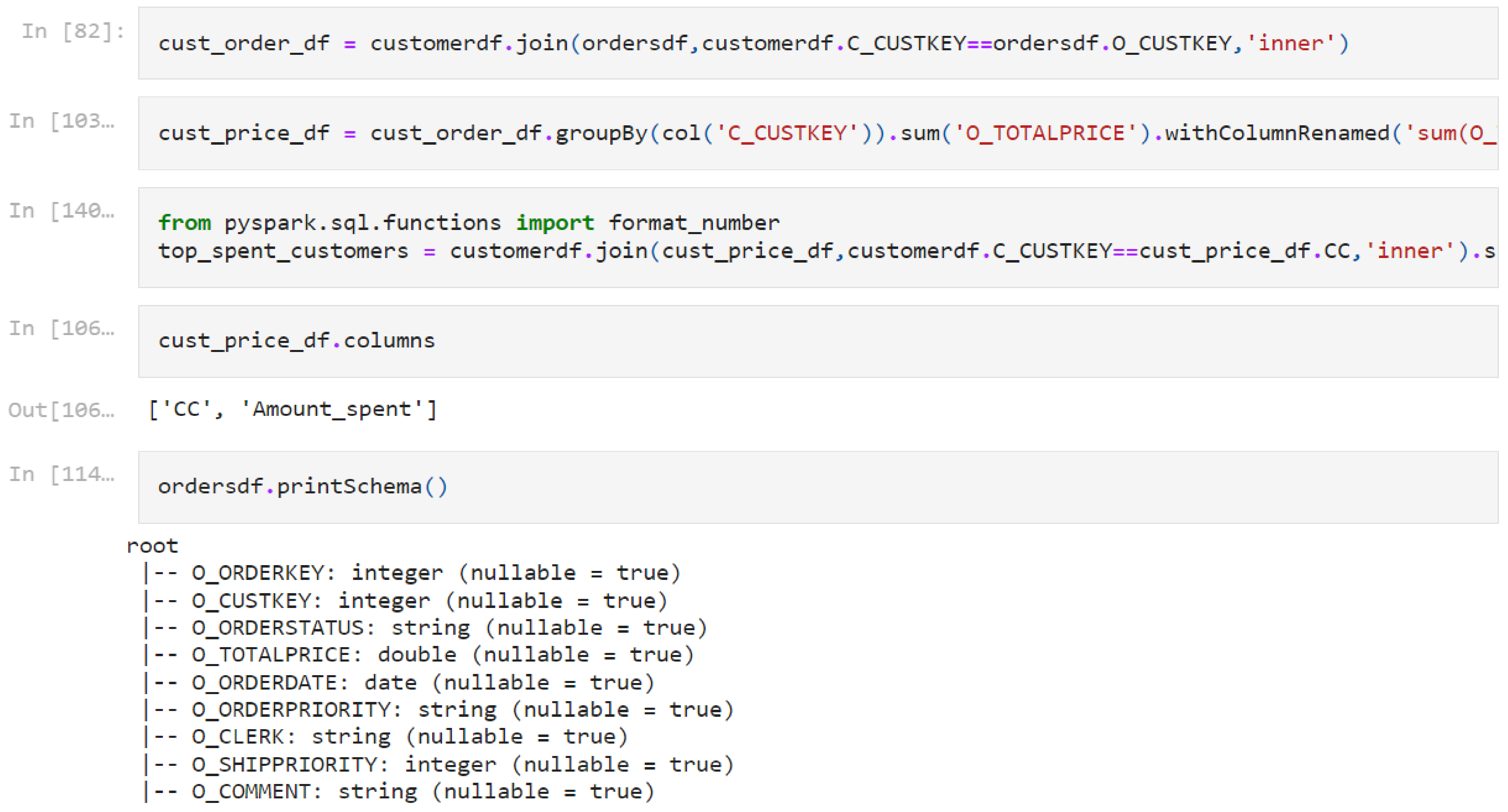


```
In [55]:
@udf(returnType=StringType())
def concat_shell(column):
    return str(column)+"__shell__"
```

```
In [56]:
df.select(concat_shell(col('Rank')).alias("col")).show()
```

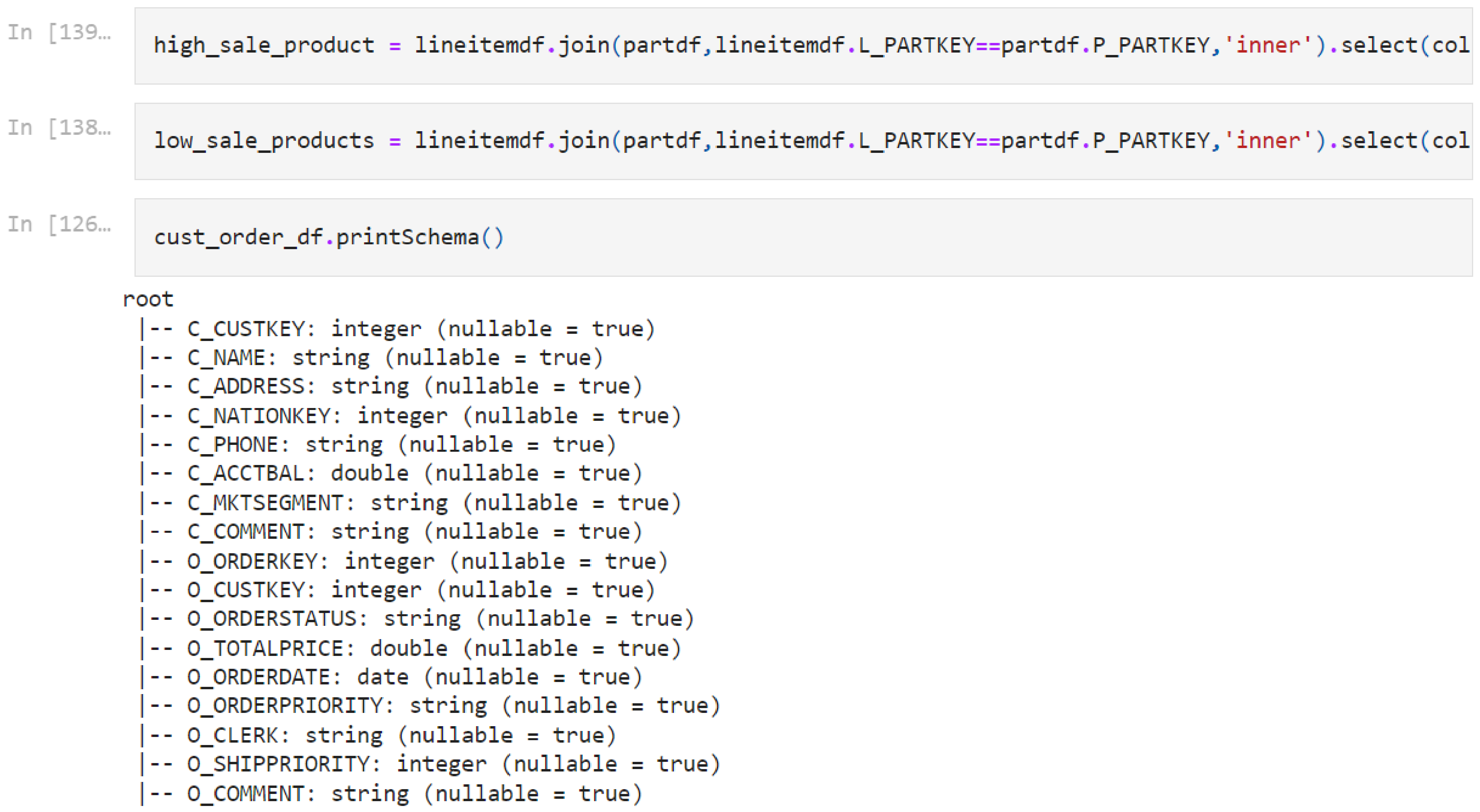
```
+-----------+
|        col|
+-----------+
| 1__shell__|
| 2__shell__|
| 3__shell__|
| 4__shell__|
| 5__shell__|
| 6__shell__|
| 7__shell__|
| 8__shell__|
| 9__shell__|
|10__shell__|
|11__shell__|
|12__shell__|
|13__shell__|
|14__shell__|
|15__shell__|
|16__shell__|
|17__shell__|
|18__shell__|
|19__shell__|
|20__shell__|
+-----------+
only showing top 20 rows
```

22-09-2023



```
In [82]:  cust_order_df = customerdf.join(ordersdf,customerdf.C_CUSTKEY==ordersdf.O_CUSTKEY,'inner')

In [103…  cust_price_df = cust_order_df.groupBy(col('C_CUSTKEY')).sum('O_TOTALPRICE').withColumnRenamed('sum(O_

In [140…  from pyspark.sql.functions import format_number
          top_spent_customers = customerdf.join(cust_price_df,customerdf.C_CUSTKEY==cust_price_df.CC,'inner').s

In [106…  cust_price_df.columns

Out[106…  ['CC', 'Amount_spent']

In [114…  ordersdf.printSchema()

root
 |-- O_ORDERKEY: integer (nullable = true)
 |-- O_CUSTKEY: integer (nullable = true)
 |-- O_ORDERSTATUS: string (nullable = true)
 |-- O_TOTALPRICE: double (nullable = true)
 |-- O_ORDERDATE: date (nullable = true)
 |-- O_ORDERPRIORITY: string (nullable = true)
 |-- O_CLERK: string (nullable = true)
 |-- O_SHIPPRIORITY: integer (nullable = true)
 |-- O_COMMENT: string (nullable = true)
```

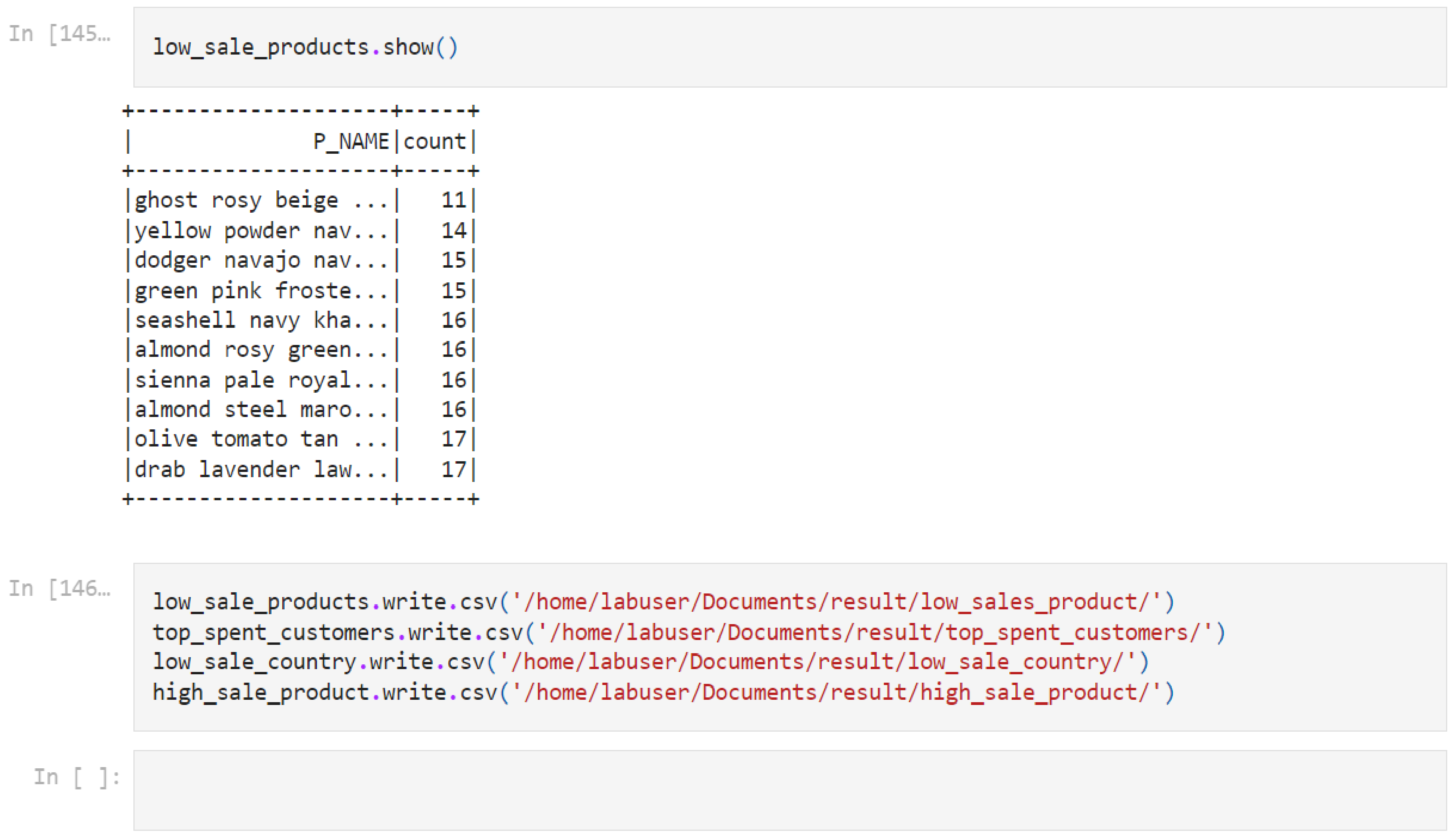
22-09-2023



```
In [139…  high_sale_product = lineitemdf.join(partdf,lineitemdf.L_PARTKEY==partdf.P_PARTKEY,'inner').select(col

In [138…  low_sale_products = lineitemdf.join(partdf,lineitemdf.L_PARTKEY==partdf.P_PARTKEY,'inner').select(col

In [126…  cust_order_df.printSchema()
```

```
root
 |-- C_CUSTKEY: integer (nullable = true)
 |-- C_NAME: string (nullable = true)
 |-- C_ADDRESS: string (nullable = true)
 |-- C_NATIONKEY: integer (nullable = true)
 |-- C_PHONE: string (nullable = true)
 |-- C_ACCTBAL: double (nullable = true)
 |-- C_MKTSEGMENT: string (nullable = true)
 |-- C_COMMENT: string (nullable = true)
 |-- O_ORDERKEY: integer (nullable = true)
 |-- O_CUSTKEY: integer (nullable = true)
 |-- O_ORDERSTATUS: string (nullable = true)
 |-- O_TOTALPRICE: double (nullable = true)
 |-- O_ORDERDATE: date (nullable = true)
 |-- O_ORDERPRIORITY: string (nullable = true)
 |-- O_CLERK: string (nullable = true)
 |-- O_SHIPPRIORITY: integer (nullable = true)
 |-- O_COMMENT: string (nullable = true)
```

22-09-2023



```
In [145…  low_sale_products.show()

+--------------------+-----+
|              P_NAME|count|
+--------------------+-----+
|ghost rosy beige ...|   11|
|yellow powder nav...|   14|
|dodger navajo nav...|   15|
|green pink froste...|   15|
|seashell navy kha...|   16|
|almond rosy green...|   16|
|sienna pale royal...|   16|
|almond steel maro...|   16|
|olive tomato tan ...|   17|
|drab lavender law...|   17|
+--------------------+-----+
```

```
In [146…  low_sale_products.write.csv('/home/labuser/Documents/result/low_sales_product/')
          top_spent_customers.write.csv('/home/labuser/Documents/result/top_spent_customers/')
          low_sale_country.write.csv('/home/labuser/Documents/result/low_sale_country/')
          high_sale_product.write.csv('/home/labuser/Documents/result/high_sale_product/')

In [ ]:
```