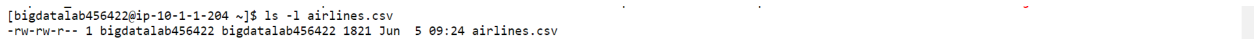
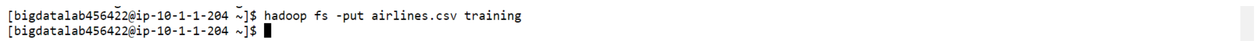
→ Airlines project in PySpark

```
[bigdatalab456422@ip-10-1-1-204 ~]$ ls -l airlines.csv
```

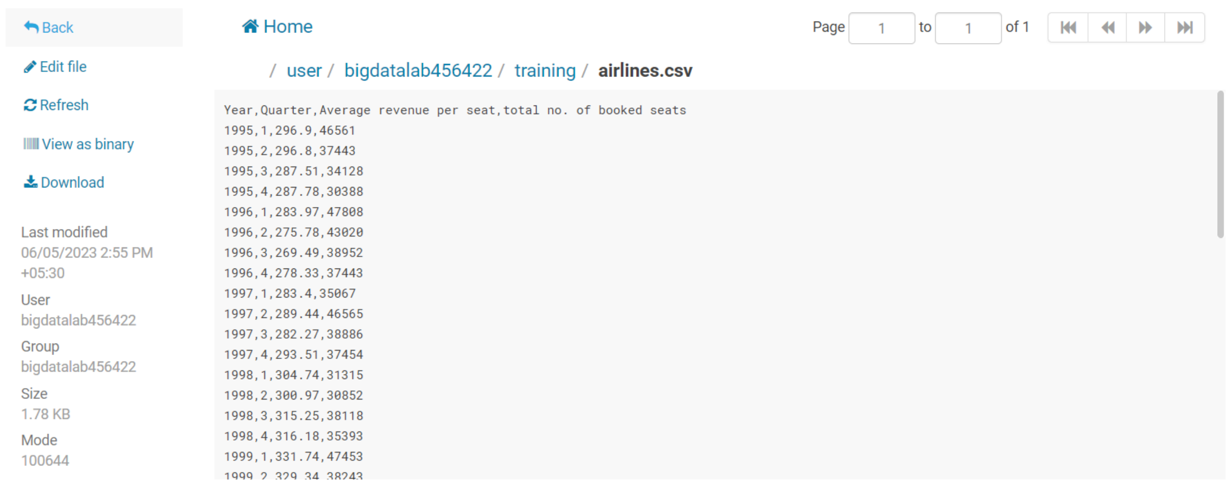


```
[bigdatalab456422@ip-10-1-1-204 ~]$ ls -l airlines.csv
-rw-rw-r-- 1 bigdatalab456422 bigdatalab456422 1821 Jun  5 09:24 airlines.csv
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -put airlines.csv training
```



```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -put airlines.csv training
[bigdatalab456422@ip-10-1-1-204 ~]$
```

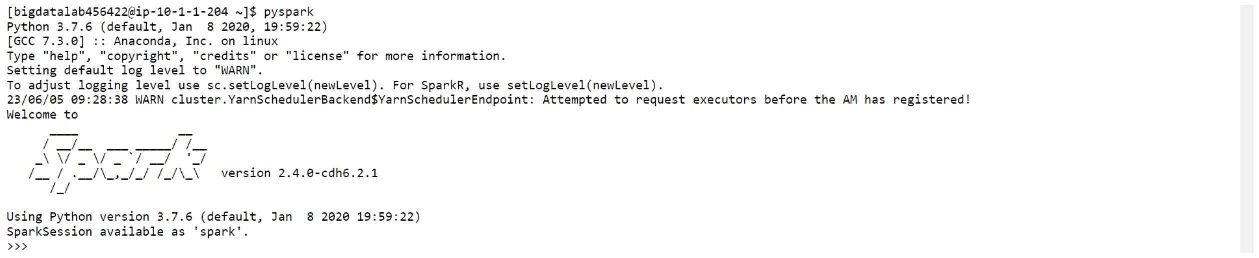


/ user / bigdatalab456422 / training / airlines.csv

Last modified
06/05/2023 2:55 PM
+05:30
User
bigdatalab456422
Group
bigdatalab456422
Size
1.78 KB
Mode
100644

```
Year,Quarter,Average revenue per seat,total no. of booked seats
1995,1,296.9,46561
1995,2,296.8,37443
1995,3,287.51,34128
1995,4,287.78,30388
1996,1,283.97,47808
1996,2,275.78,43020
1996,3,269.49,38952
1996,4,278.33,37443
1997,1,283.4,35067
1997,2,289.44,46565
1997,3,282.27,38886
1997,4,293.51,37454
1998,1,304.74,31315
1998,2,300.97,30852
1998,3,315.25,38118
1998,4,316.18,35393
1999,1,331.74,47453
1999,2,329.34,38243
```

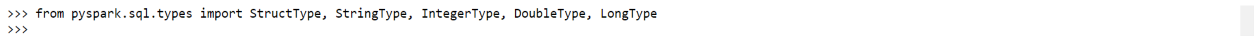
```
[bigdatalab456422@ip-10-1-1-204 ~]$ pyspark
```



```
[bigdatalab456422@ip-10-1-1-204 ~]$ pyspark
Python 3.7.6 (default, Jan  8 2020, 19:59:22)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/06/05 09:28:38 WARN cluster.YarnSchedulerBackend$YarnSchedulerEndpoint: Attempted to request executors before the AM has registered!
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.0-cdh6.2.1
      /_/

Using Python version 3.7.6 (default, Jan  8 2020 19:59:22)
SparkSession available as 'spark'.
>>>
```

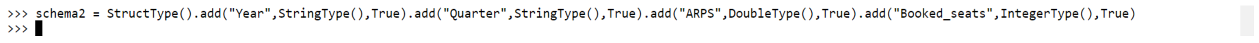
→ find the year with highest revenue

```
>>> from pyspark.sql.types import StructType, StringType, IntegerType, DoubleType, LongType
```



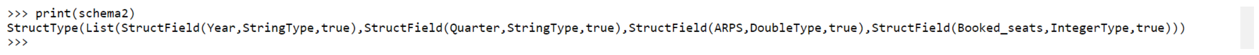
```
>>> from pyspark.sql.types import StructType, StringType, IntegerType, DoubleType, LongType
>>>
```

```
>>> schema2 = StructType().add("Year",StringType(),True).add("Quarter",StringType(),True).add("ARPS",DoubleType(),True).add("Booked_seats",IntegerType(),True)
```



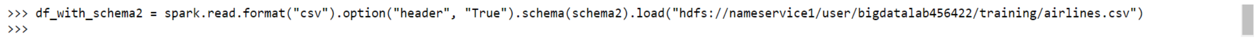
```
>>> schema2 = StructType().add("Year",StringType(),True).add("Quarter",StringType(),True).add("ARPS",DoubleType(),True).add("Booked_seats",IntegerType(),True)
>>>
```

```
>>> print(schema2)
```



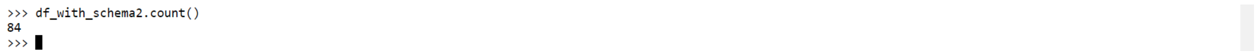
```
>>> print(schema2)
StructType(List(StructField(Year,StringType,true),StructField(Quarter,StringType,true),StructField(ARPS,DoubleType,true),StructField(Booked_seats,IntegerType,true)))
>>>
```

```
>>> df_with_schema2 = spark.read.format("csv").option("header",
"True").schema(schema2).load("hdfs://nameservice1/user/bigdatalab4564
22/training/airlines.csv")
```



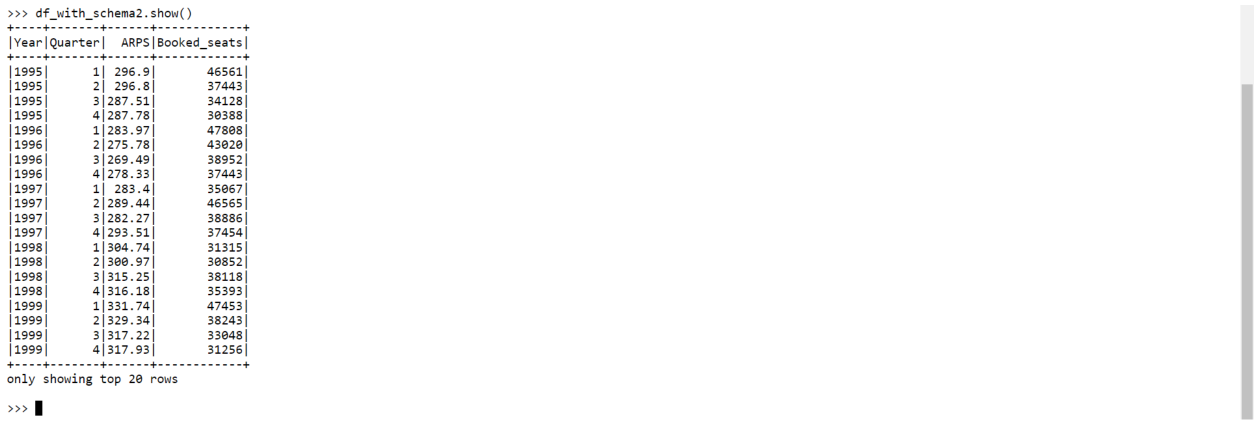
```
>>> df_with_schema2 = spark.read.format("csv").option("header", "True").schema(schema2).load("hdfs://nameservice1/user/bigdatalab456422/training/airlines.csv")
>>>
```

```
>>> df_with_schema2.count()
```



```
>>> df_with_schema2.count()
84
>>>
```

```
>>> df_with_schema2.show()
```

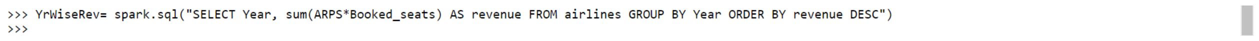


```
>>> df_with_schema2.show()
+----+-------+------+------------+
|Year|Quarter|  ARPS|Booked_seats|
+----+-------+------+------------+
|1995|      1| 296.9|       46561|
|1995|      2| 296.8|       37443|
|1995|      3|287.51|       34128|
|1995|      4|287.78|       30388|
|1996|      1|283.97|       47808|
|1996|      2|275.78|       43020|
|1996|      3|269.49|       38952|
|1996|      4|278.33|       37443|
|1997|      1| 283.4|       35067|
|1997|      2|289.44|       46565|
|1997|      3|282.27|       38886|
|1997|      4|293.51|       37454|
|1998|      1|304.74|       31315|
|1998|      2|300.97|       30852|
|1998|      3|315.25|       38118|
|1998|      4|316.18|       35393|
|1999|      1|331.74|       47453|
|1999|      2|329.34|       38243|
|1999|      3|317.22|       33048|
|1999|      4|317.93|       31256|
+----+-------+------+------------+
only showing top 20 rows

>>>
```

```
>>> df_with_schema2.registerTempTable("airlines")
```

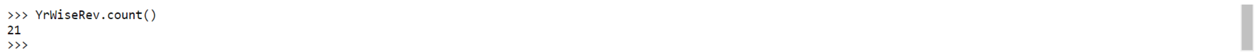
```
>>> df_with_schema2.registerTempTable("airlines")
>>>
```

```
>>> YrWiseRev= spark.sql("SELECT Year, sum(ARPS*Booked_seats) AS
revenue FROM airlines GROUP BY Year ORDER BY revenue DESC")
```



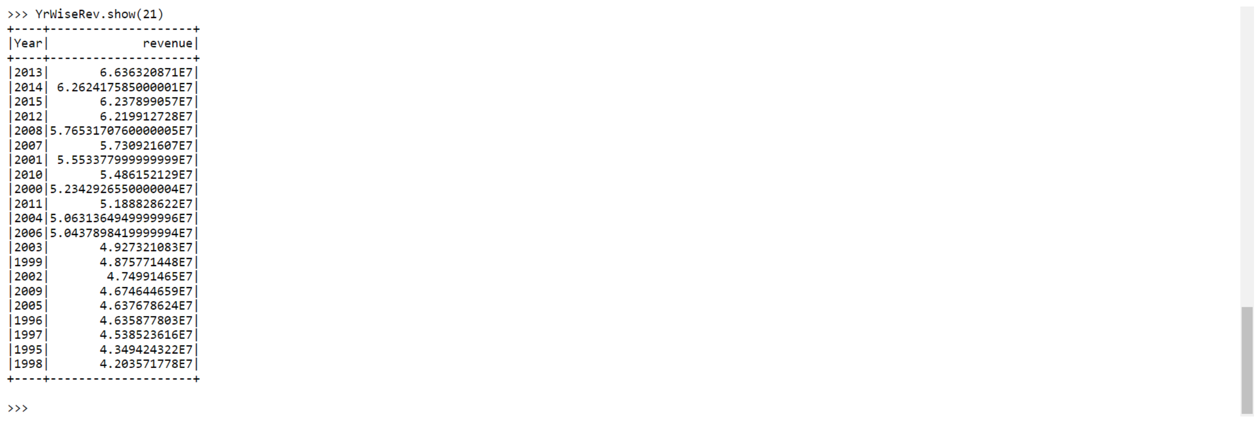
```
>>> YrWiseRev= spark.sql("SELECT Year, sum(ARPS*Booked_seats) AS revenue FROM airlines GROUP BY Year ORDER BY revenue DESC")
>>>
```

```
>>> YrWiseRev.count()
```



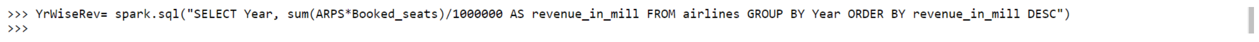
```
>>> YrWiseRev.count()
21
>>>
```

```
>>> YrWiseRev.show(21)
```



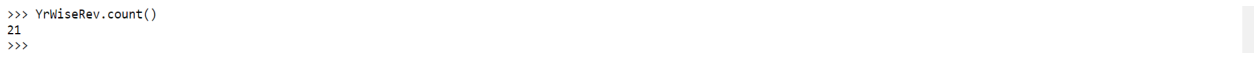
```
>>> YrWiseRev.show(21)
+----+--------------------+
|Year|             revenue|
+----+--------------------+
|2013|       6.636320871E7|
|2014| 6.262417585000001E7|
|2015|       6.237899057E7|
|2012|       6.219912728E7|
|2008|5.7653170760000005E7|
|2007|       5.730921607E7|
|2001| 5.553377999999999E7|
|2010|       5.486152129E7|
|2000|5.2342926550000004E7|
|2011|       5.188828622E7|
|2004|5.0631364949999996E7|
|2006|5.0437898419999994E7|
|2003|       4.927321083E7|
|1999|       4.875771448E7|
|2002|        4.74991465E7|
|2009|       4.674644659E7|
|2005|       4.637678624E7|
|1996|       4.635877803E7|
|1997|       4.538523616E7|
|1995|       4.349424322E7|
|1998|       4.203571778E7|
+----+--------------------+

>>>
```

```
>>> YrWiseRev= spark.sql("SELECT Year, sum(ARPS*Booked_seats)/1000000
AS revenue_in_mill FROM airlines GROUP BY Year ORDER BY
revenue_in_mill DESC")
```



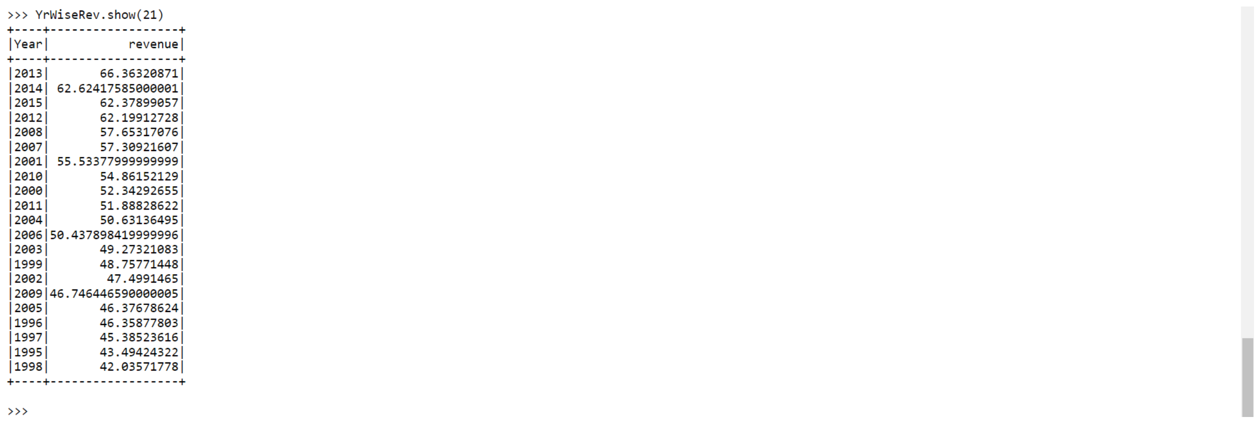
```
>>> YrWiseRev= spark.sql("SELECT Year, sum(ARPS*Booked_seats)/1000000 AS revenue_in_mill FROM airlines GROUP BY Year ORDER BY revenue_in_mill DESC")
>>>
```

```
>>> YrWiseRev.count()
```



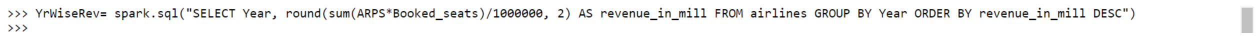
```
>>> YrWiseRev.count()
21
>>>
```

```
>>> YrWiseRev.show(21)
```



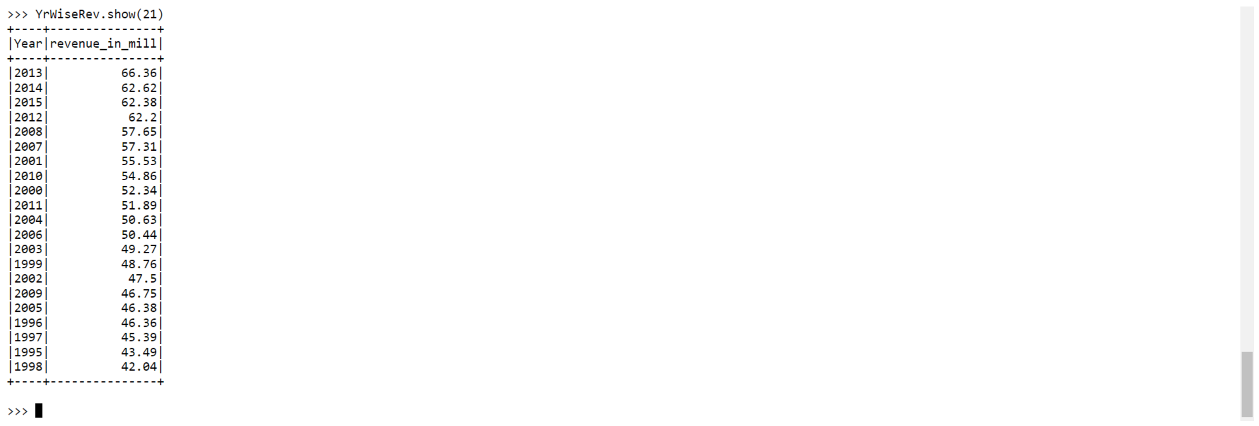
```
>>> YrWiseRev.show(21)
+----+------------------+
|Year|           revenue|
+----+------------------+
|2013|       66.36320871|
|2014| 62.62417585000001|
|2015|       62.37899057|
|2012|       62.19912728|
|2008|       57.65317076|
|2007|       57.30921607|
|2001| 55.53377999999999|
|2010|       54.86152129|
|2000|       52.34292655|
|2011|       51.88828622|
|2004|       50.63136495|
|2006|50.437898419999996|
|2003|       49.27321083|
|1999|       48.75771448|
|2002|        47.4991465|
|2009|46.746446590000005|
|2005|       46.37678624|
|1996|       46.35877803|
|1997|       45.38523616|
|1995|       43.49424322|
|1998|       42.03571778|
+----+------------------+

>>>
```

```
>>> YrWiseRev= spark.sql("SELECT Year,
round(sum(ARPS*Booked_seats)/1000000, 2) AS revenue_in_mill FROM
airlines GROUP BY Year ORDER BY revenue_in_mill DESC")
```



```
>>> YrWiseRev= spark.sql("SELECT Year, round(sum(ARPS*Booked_seats)/1000000, 2) AS revenue_in_mill FROM airlines GROUP BY Year ORDER BY revenue_in_mill DESC")
>>>
```

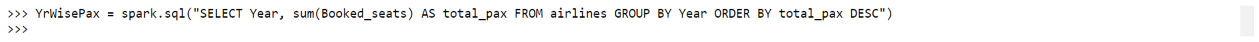
```
>>> YrWiseRev.show(21)
```



```
>>> YrWiseRev.show(21)
+----+---------------+
|Year|revenue_in_mill|
+----+---------------+
|2013|          66.36|
|2014|          62.62|
|2015|          62.38|
|2012|           62.2|
|2008|          57.65|
|2007|          57.31|
|2001|          55.53|
|2010|          54.86|
|2000|          52.34|
|2011|          51.89|
|2004|          50.63|
|2006|          50.44|
|2003|          49.27|
|1999|          48.76|
|2002|           47.5|
|2009|          46.75|
|2005|          46.38|
|1996|          46.36|
|1997|          45.39|
|1995|          43.49|
|1998|          42.04|
+----+---------------+

>>> █
```

→ find the year with highest number of PAX flown

```
>>> YrWisePax = spark.sql("SELECT Year, sum(Booked_seats) AS
total_pax FROM airlines GROUP BY Year ORDER BY total_pax DESC")
```

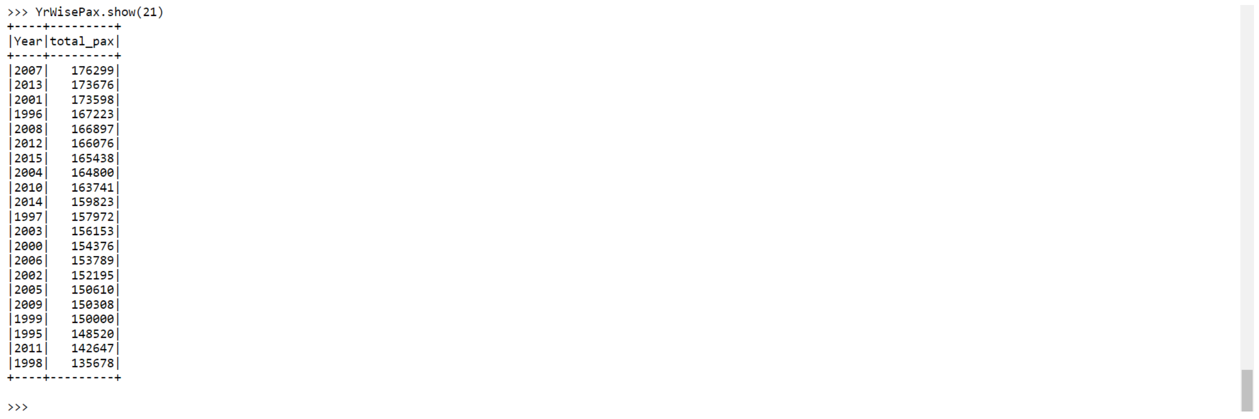


```
>>> YrWisePax = spark.sql("SELECT Year, sum(Booked_seats) AS total_pax FROM airlines GROUP BY Year ORDER BY total_pax DESC")
>>>
```

```
>>> YrWisePax.count()
```

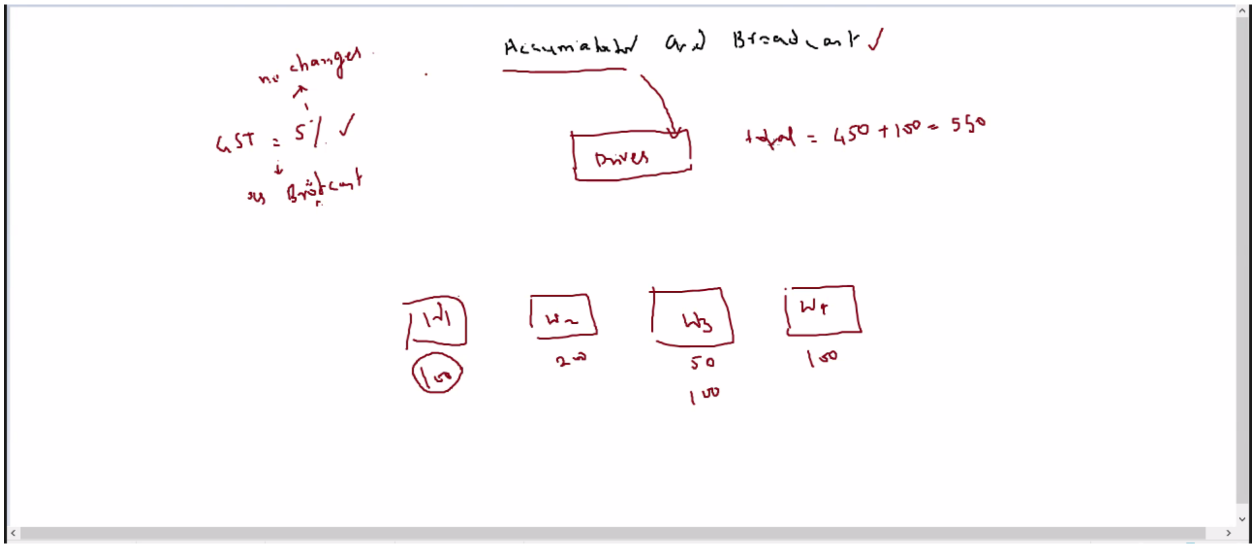
```
>>> YrWisePax.count()
21
>>>
```

```
>>> YrWisePax.show(21)
```



```
>>> YrWisePax.show(21)
+----+---------+
|Year|total_pax|
+----+---------+
|2007|   176299|
|2013|   173676|
|2001|   173598|
|1996|   167223|
|2008|   166897|
|2012|   166076|
|2015|   165438|
|2004|   164800|
|2010|   163741|
|2014|   159823|
|1997|   157972|
|2003|   156153|
|2000|   154376|
|2006|   153789|
|2002|   152195|
|2005|   150610|
|2009|   150308|
|1999|   150000|
|1995|   148520|
|2011|   142647|
|1998|   135678|
+----+---------+

>>>
```

→ Types of variables in Spark



a. Accumulators
    i. Are at driver/spark app
    ii. Value can change as it sums up the values at each of the worker nodes
    iii. Active throughout entire session
    iv. Need to re-initialize it to zero to reset accumulator

b. Broadcast variables
    i. Are like constant/immutable variable
    ii. Can be used in any program , in any stage
    iii. Declared at driver node, Can be used from any of the worker nodes

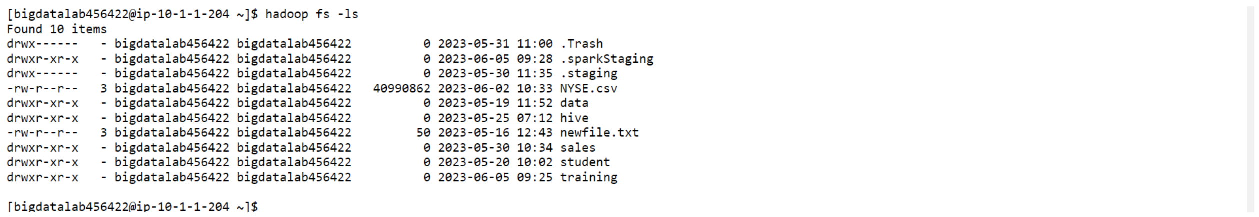
→ for port number ,

a. check core-site.xml
b. hdfs://localhost:54310
c.

→ Retail Project for accumulator & broadcast variables

→ write a program using shared variables like broadcast and accumulators to calculate total tax and total profit using Retail data

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -ls
```



```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -ls
Found 10 items
drwx------   - bigdatalab456422 bigdatalab456422          0 2023-05-31 11:00 .Trash
drwxr-xr-x   - bigdatalab456422 bigdatalab456422          0 2023-06-05 09:28 .sparkStaging
drwx------   - bigdatalab456422 bigdatalab456422          0 2023-05-30 11:35 .staging
-rw-r--r--   3 bigdatalab456422 bigdatalab456422   40990862 2023-06-02 10:33 NYSE.csv
drwxr-xr-x   - bigdatalab456422 bigdatalab456422          0 2023-05-19 11:52 data
drwxr-xr-x   - bigdatalab456422 bigdatalab456422          0 2023-05-25 07:12 hive
-rw-r--r--   3 bigdatalab456422 bigdatalab456422         50 2023-05-16 12:43 newfile.txt
drwxr-xr-x   - bigdatalab456422 bigdatalab456422          0 2023-05-30 10:34 sales
drwxr-xr-x   - bigdatalab456422 bigdatalab456422          0 2023-05-20 10:02 student
drwxr-xr-x   - bigdatalab456422 bigdatalab456422          0 2023-06-05 09:25 training

[bigdatalab456422@ip-10-1-1-204 ~]$
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -mkdir retail
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -mkdir retail
[bigdatalab456422@ip-10-1-1-204 ~]$
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -ls retail
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -ls retail
[bigdatalab456422@ip-10-1-1-204 ~]$
```

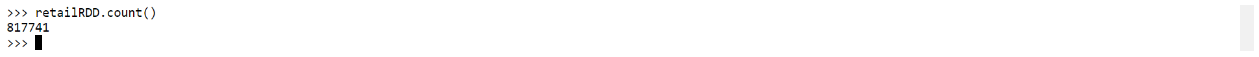
```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -put D11 D12 D01 D02
retail
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -put D11 D12 D01 D02 retail
[bigdatalab456422@ip-10-1-1-204 ~]$
```

```
>>> retailRDD =
sc.textFile("hdfs://nameservice1/user/bigdatalab456422/retail")
```

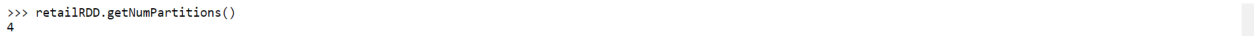
```
>>> retailRDD = sc.textFile("hdfs://nameservice1/user/bigdatalab456422/retail")
>>>
```

```
>>> retailRDD.count()
```



```
>>> retailRDD.count()
817741
>>>
```

```
>>> retailRDD.getNumPartitions()
```



```
>>> retailRDD.getNumPartitions()
4
```

```
>>> gst = sc.broadcast(5.00)
```

```
>>> gst = sc.broadcast(5.00)
>>>
```

```
>>> totalTax = sc.accumulator(0.00)
```

```
>>> totalTax = sc.accumulator(0.00)
>>>
```

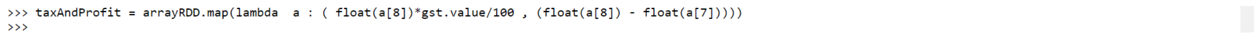
```
>>> totalProfit = sc.accumulator(0.00)
```

```
>>> totalProfit = sc.accumulator(0.00)
>>>
```

```
>>> arrayRDD = retailRDD.map(lambda a : a.split(";"))
```

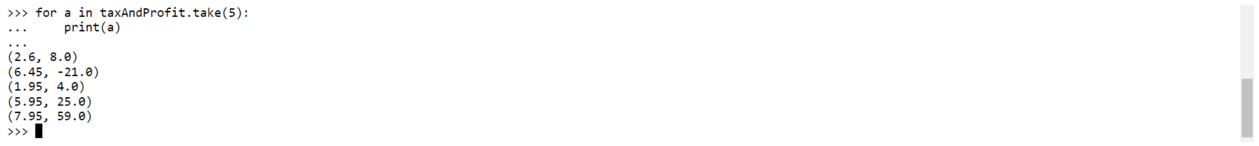
```
>>> arrayRDD = retailRDD.map(lambda a : a.split(";"))
>>>
```

```
>>> taxAndProfit = arrayRDD.map(lambda  a : (
float(a[8])*gst.value/100 , (float(a[8]) - float(a[7]))))
```



```
>>> taxAndProfit = arrayRDD.map(lambda  a : ( float(a[8])*gst.value/100 , (float(a[8]) - float(a[7]))))
>>>
```

```
>>> for a in taxAndProfit.take(5):
...     print(a)
```

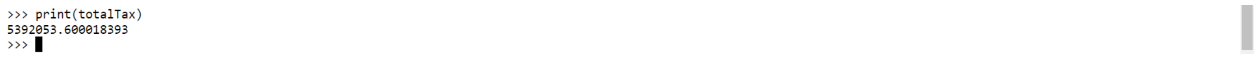


```
>>> for a in taxAndProfit.take(5):
...     print(a)
...
(2.6, 8.0)
(6.45, -21.0)
(1.95, 4.0)
(5.95, 25.0)
(7.95, 59.0)
>>>
```

```
>>> for line in taxAndProfit.collect():
...     totalTax += line[0]
```

```
>>> for line in taxAndProfit.collect():
...     totalTax += line[0]
...
>>>
```

```
>>> print(totalTax)
```

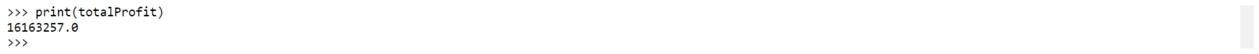


```
>>> print(totalTax)
5392053.600018393
>>>
```

```
>>> for line in taxAndProfit.collect():
...     totalProfit += line[1]
```

```
>>> for line in taxAndProfit.collect():
...     totalProfit += line[1]
...
>>>
```

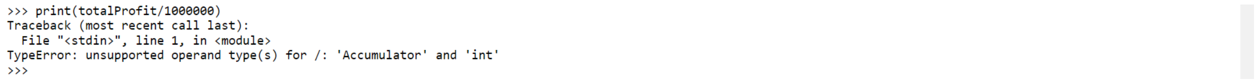
```
>>> print(totalProfit)
```



```
>>> print(totalProfit)
16163257.0
>>>
```

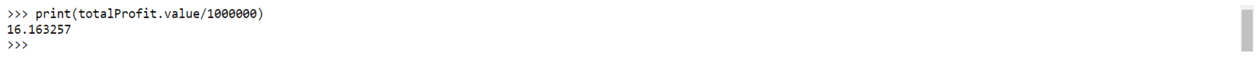
# does not work as totalProfit is of type Accumulator, so it needs to accessed using value

```
>>> print(totalProfit/1000000)
```



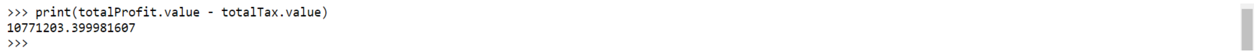
```
>>> print(totalProfit/1000000)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for /: 'Accumulator' and 'int'
>>>
```

```
>>> print(totalProfit.value/1000000)
```



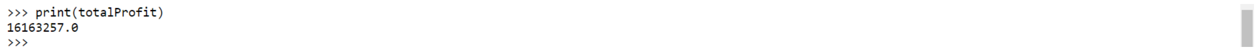
```
>>> print(totalProfit.value/1000000)
16.163257
>>>
```

```
# find net profit after deducting tax
>>> print(totalProfit.value - totalTax.value)
```



```
>>> print(totalProfit.value - totalTax.value)
10771203.399981607
>>>
```

```
>>> print(totalProfit)
```



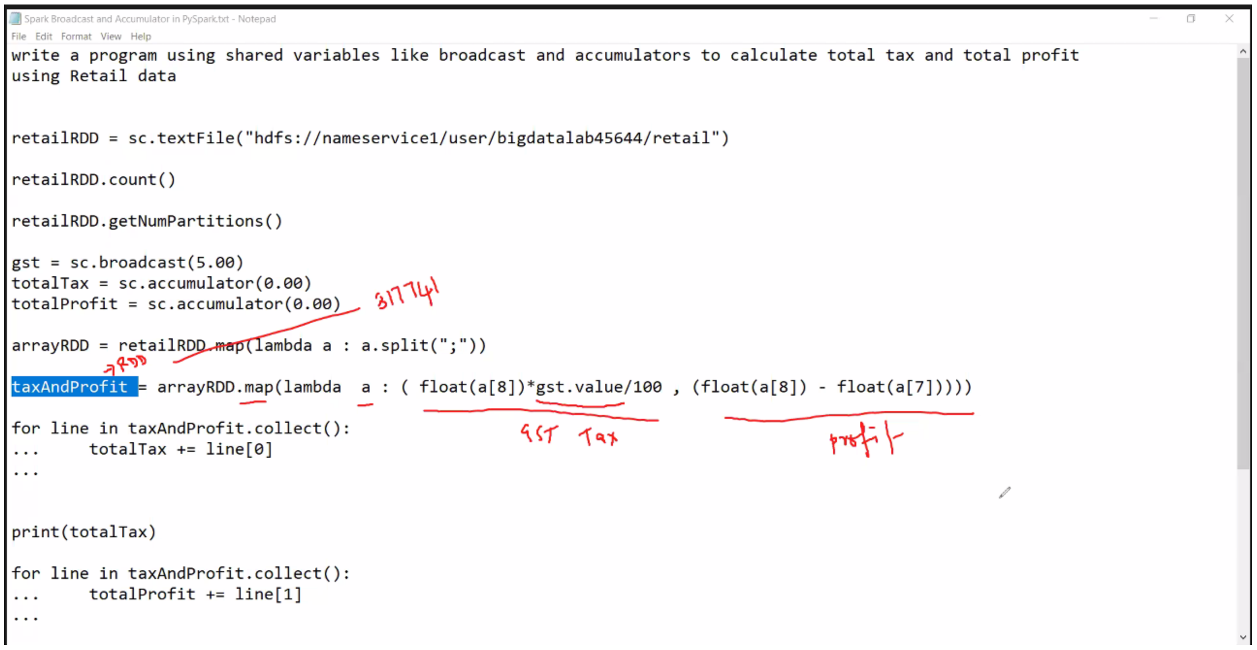
```
>>> print(totalProfit)
16163257.0
>>>
```

```
>>> totalProfit = sc.accumulator(0.0)
```

```
>>> totalProfit = sc.accumulator(0.0)
>>>
```

```
>>> print(totalProfit)
```

```
>>> print(totalProfit)
0.0
>>>
```


Spark Broadcast and Accumulator in PySpark.txt - Notepad
File Edit Format View Help
317741
→RDD
GST Tax
profit

```
write a program using shared variables like broadcast and accumulators to calculate total tax and total profit
using Retail data


retailRDD = sc.textFile("hdfs://nameservice1/user/bigdatalab45644/retail")

retailRDD.count()

retailRDD.getNumPartitions()

gst = sc.broadcast(5.00)
totalTax = sc.accumulator(0.00)
totalProfit = sc.accumulator(0.00)

arrayRDD = retailRDD.map(lambda a : a.split(";"))

taxAndProfit = arrayRDD.map(lambda  a : ( float(a[8])*gst.value/100 , (float(a[8]) - float(a[7]))))

for line in taxAndProfit.collect():
...     totalTax += line[0]
...


print(totalTax)

for line in taxAndProfit.collect():
...     totalProfit += line[1]
...
```

Previous exam pattern

1. MapReduce Problem 1 no
2. Hive Queries 5 no
3. Spark Queries 5 no