PYSPARK PRACTICE

🡪Creating a Pyspark session.

🡪Giving the required credentials to connect to the AWS.

🡪Importing data from a s3 bucket to spark environment.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

🡪Displaying a particular column of the dataset

A screen shot of a computer screen

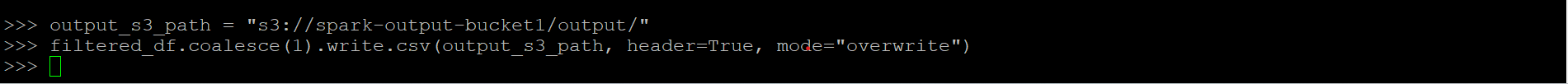
Description automatically generated

A screen shot of a computer

Description automatically generated

🡪Define the output S3 path where you want to store the results.

🡪Writing the filtered Data Frame to a new S3 file in CSV format.



🡪Below we can see that the dataframe we filtered stored in the specified s3 location.

A screenshot of a computer

Description automatically generated

🡪Below is the output file of Avg ratings<3

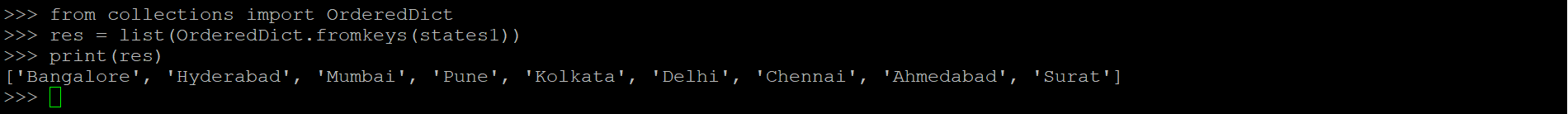
A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

🡪below we perform the distinct values in a column.

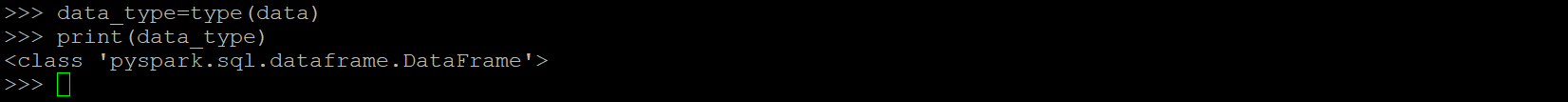


🡪Date incrementation

A black background with many small lights

Description automatically generated

🡪To check the type of the data

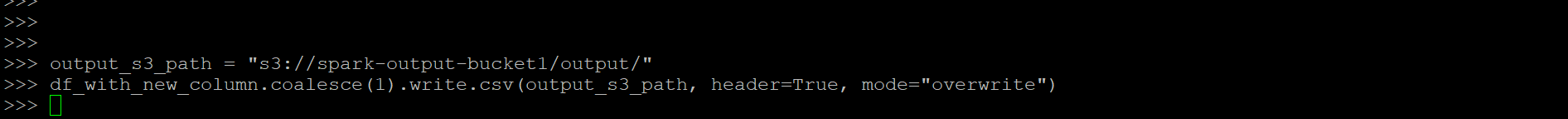


🡪To add a new column date with constant value

A screen shot of a computer

Description automatically generated

🡪storing the output in the s3



🡪New column with date

A screenshot of a computer

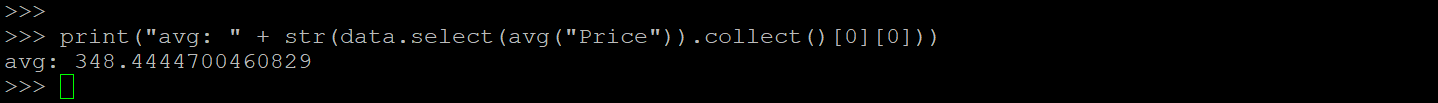
Description automatically generated

🡪**approx\_count\_distinct**: Estimates the approximate number of distinct values in a column.

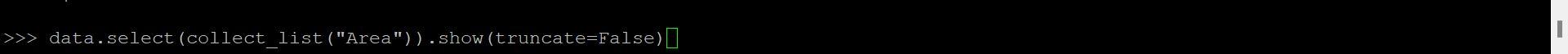
A black screen with white text

Description automatically generated

🡪**avg**: Calculates the average of a numerical column.



🡪**collect\_list**: Collects the elements of a column into a list.



A screen shot of a computer screen

Description automatically generated

🡪**collect\_set**: Collects the unique elements of a column into a set.

A computer screen with a keyboard

Description automatically generated

🡪**countDistinct**: Counts the distinct values of selected columns.

A black background with a black square

Description automatically generated

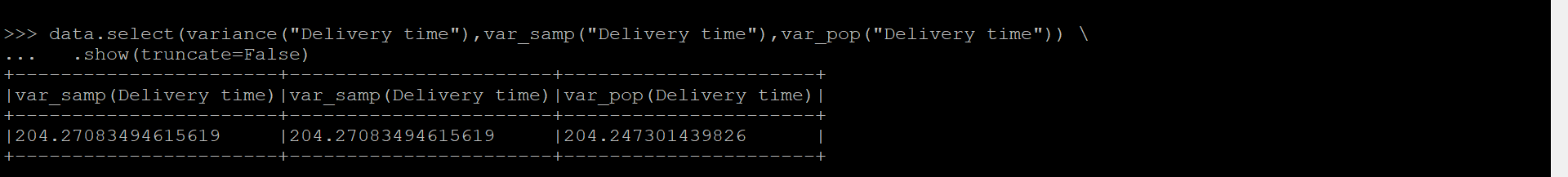
* **first**: Returns the first element of a column.
* **last**: Returns the last element of a column.
* **kurtosis**: Computes the kurtosis of a numerical column.
* **max**: Finds the maximum value in a numerical column.
* **min**: Finds the minimum value in a numerical column.
* **mean**: Computes the mean of a numerical column.
* **skewness**: Computes the skewness of a numerical column.
* **stddev**: Calculates the standard deviation of a numerical column.
* **stddev\_samp**: Calculates the sample standard deviation of a numerical column.
* **stddev\_pop**: Calculates the population standard deviation of a numerical column.
* **sum**: Calculates the sum of a numerical column.
* **sumDistinct**: Calculates the sum of distinct values in a numerical column.
* **variance**: Computes the variance of a numerical column.
* **var\_samp**: Computes the sample variance of a numerical column.
* **var\_pop**: Computes the population variance of a numerical column.

A screenshot of a computer

Description automatically generated

A computer screen with text on it

Description automatically generated



🡪Broadcast join

🡪Took a second data set which has a similar column.

A screen shot of a computer program

Description automatically generated

🡪Performed data1 on data2 and then stored the output to a s3 location.

A black screen with many small text

Description automatically generated

A screenshot of a computer

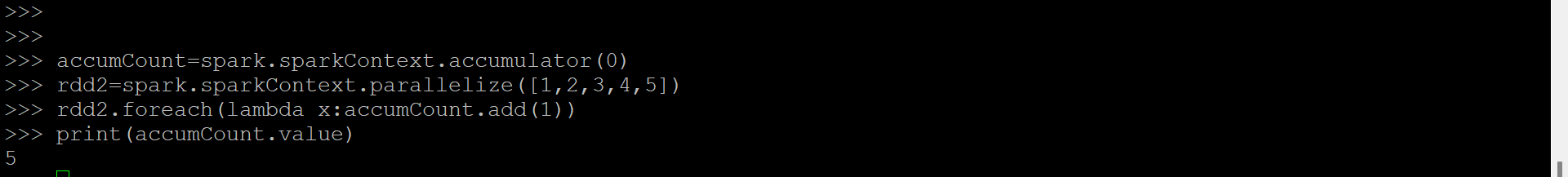
Description automatically generated

🡪Accumulator--The accumulator is initialized to 0 and then used to accumulate the values from the RDD. The accumulator accumulates the sum of the elements in the RDD using the foreach transformation.

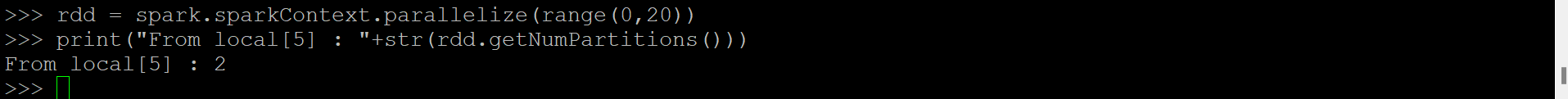
A blurry image of a city

Description automatically generated

🡪accumulators to do a counter.

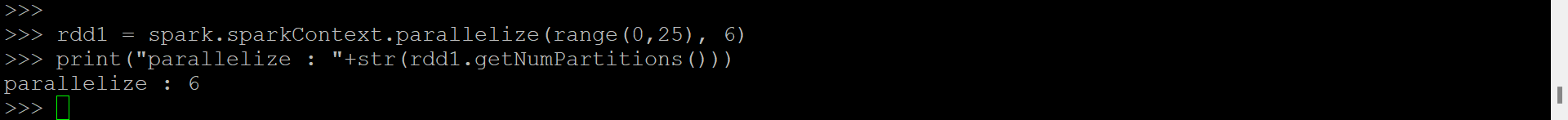


🡪repartition() vs coalesce(), repartition() is used to increase or decrease the RDD/DataFrame partitions whereas the PySpark coalesce() is used to only decrease the number of partitions in an efficient way.



for the above code the partitions will be based on our configurations and cluster settings.

🡪In the below, the parallelize() method is called with two arguments. The first argument is range(0, 25), which generates a sequence of numbers from 0 to 24. The second argument, 6, specifies the desired number of partitions for the RDD.)

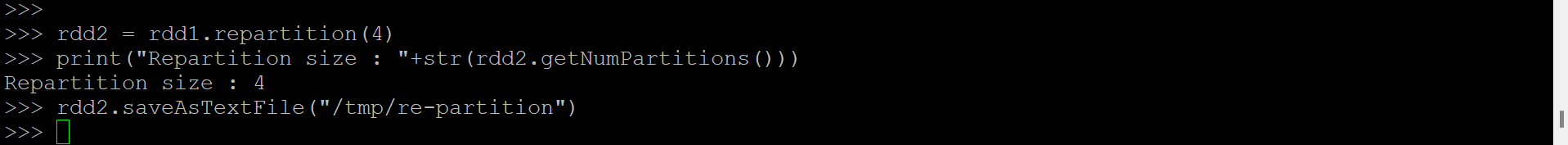


🡪rddFromFile = spark.sparkContext.textFile("src/main/resources/test.txt",10)

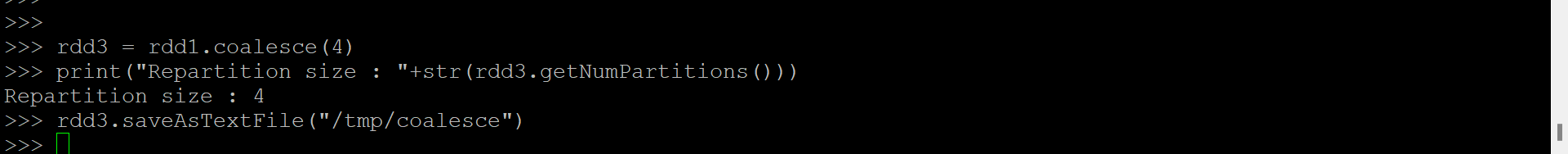
print("TextFile : "+str(rddFromFile.getNumPartitions()))

( RDD using the textFile() method and printing the number of partitions in the RDD.In this line, the textFile() method is called on the sparkContext object to read the text file located at "src/main/resources/test.txt". The second argument, 10, specifies the desired number of partitions for the RDD. In this case, the RDD will be divided into 10 partitions.)

🡪repartition()



🡪coalesce()



🡪List to column

A computer screen with text on it

Description automatically generated

🡪Groupby function

A screen shot of a computer

Description automatically generated

🡪Using lit function to create a new dataframe and adding a new column to it with a constant value.

A screen shot of a computer

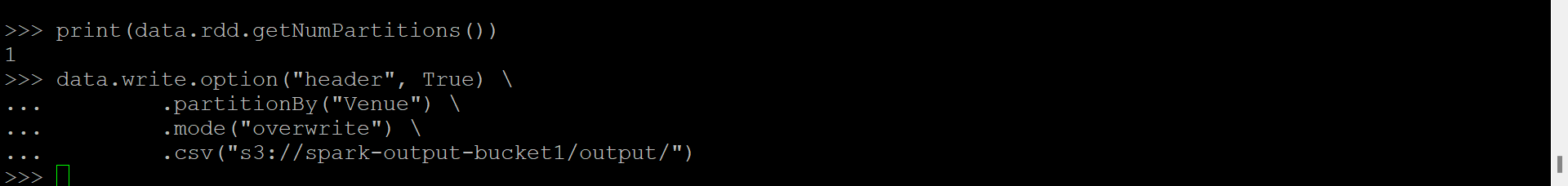
Description automatically generated

🡪concat ws expects the first parameter to be the separator string and the subsequent parameters to be the columns to concatenate.

A black screen with white text

Description automatically generated

🡪PartitionBy



A screenshot of a computer

Description automatically generated

🡪Spark pivot() function is used to pivot/rotate the data from one DataFrame/Dataset column into multiple columns

A black screen with many small colored text

Description automatically generated

A black screen with white text

Description automatically generated

A black background with a black square

Description automatically generated