Pyspark Coding Challenge

Name: PRIYESHWAR

Mail: priyesh2664@gmail.com

Spark Initialization & Data Loading

Initializes a Spark session and loads the loan dataset with headers and inferred data types.

show() displays data; count() gives total number of records.

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('Coding_Challenge').getOrCreate()

df = spark.read.csv("/content/LoanData (1).csv", header=True, inferSchema=True)

df.show()

```
+--------+------+-------+----------+------------+-------------+----------------+------------------+----------+----------------+-------------+-------------+----------+
| Loan_ID|Gender|Married|Dependents|   Education|Self_Employed|ApplicantIncome|CoapplicantIncome|LoanAmount|Loan_Amount_Term|Credit_History|Property_Area|Loan_Status|
+--------+------+-------+----------+------------+-------------+----------------+------------------+----------+----------------+-------------+-------------+----------+
|LP001002|  Male|     No|         0|    Graduate|           No|            5849|               0.0|      NULL|             360|            1|        Urban|         Y|
|LP001003|  Male|    Yes|         1|    Graduate|           No|            4583|            1508.0|       128|             360|            1|        Rural|         N|
|LP001005|  Male|    Yes|         0|    Graduate|          Yes|            3000|               0.0|        66|             360|            1|        Urban|         Y|
|LP001006|  Male|    Yes|         0|Not Graduate|           No|            2583|            2358.0|       120|             360|            1|        Urban|         Y|
|LP001008|  Male|     No|         0|    Graduate|           No|            6000|               0.0|       141|             360|            1|        Urban|         Y|
|LP001011|  Male|    Yes|         2|    Graduate|          Yes|            5417|            4196.0|       267|             360|            1|        Urban|         Y|
|LP001013|  Male|    Yes|         0|Not Graduate|           No|            2333|            1516.0|        95|             360|            1|        Urban|         Y|
|LP001014|  Male|    Yes|        3+|    Graduate|           No|            3036|            2504.0|       158|             360|            0|    Semiurban|         N|
|LP001018|  Male|    Yes|         2|    Graduate|           No|            4006|            1526.0|       168|             360|            1|        Urban|         Y|
|LP001020|  Male|    Yes|         1|    Graduate|           No|           12841|           10968.0|       349|             360|            1|    Semiurban|         N|
|LP001024|  Male|    Yes|         2|    Graduate|           No|            3200|             700.0|        70|             360|            1|        Urban|         Y|
|LP001027|  Male|    Yes|         2|    Graduate|         NULL|            2500|            1840.0|       109|             360|            1|        Urban|         Y|
|LP001028|  Male|    Yes|         2|    Graduate|           No|            3073|            8106.0|       200|             360|            1|        Urban|         Y|
|LP001029|  Male|     No|         0|    Graduate|           No|            1853|            2840.0|       114|             360|            1|        Rural|         N|
|LP001030|  Male|    Yes|         2|    Graduate|           No|            1299|            1086.0|        17|             120|            1|        Urban|         Y|
|LP001032|  Male|     No|         0|    Graduate|           No|            4950|               0.0|       125|             360|            1|        Urban|         Y|
|LP001034|  Male|     No|         1|Not Graduate|           No|            3596|               0.0|       100|             240|         NULL|        Urban|         Y|
|LP001036|Female|     No|         0|    Graduate|           No|            3510|               0.0|        76|             360|            0|        Urban|         N|
|LP001038|  Male|    Yes|         0|Not Graduate|           No|            4887|               0.0|       133|             360|            1|        Rural|         N|
|LP001041|  Male|    Yes|         0|    Graduate|         NULL|            2600|            3500.0|       115|            NULL|            1|        Urban|         Y|
+--------+------+-------+----------+------------+-------------+----------------+------------------+----------+----------------+-------------+-------------+----------+
only showing top 20 rows

614
```

Transformations

1. ***Filter***

Filters the dataset to show only self-employed applicants.

df.filter(col("elf_Employed")=="Yes").show()

Filter

```
[41] from pyspark.sql.functions import col
     df.filter(col("Self_Employed")=="Yes").show()
```

```
+--------+------+-------+----------+------------+-------------+----------------+------------------+----------+----------------+-------------+-------------+----------+
| Loan_ID|Gender|Married|Dependents|   Education|Self_Employed|ApplicantIncome|CoapplicantIncome|LoanAmount|Loan_Amount_Term|Credit_History|Property_Area|Loan_Status|
+--------+------+-------+----------+------------+-------------+----------------+------------------+----------+----------------+-------------+-------------+----------+
|LP001005|  Male|    Yes|         0|    Graduate|          Yes|            3000|               0.0|        66|             360|            1|        Urban|         Y|
|LP001011|  Male|    Yes|         2|    Graduate|          Yes|            5417|            4196.0|       267|             360|            1|        Urban|         Y|
|LP001066|  Male|    Yes|         0|    Graduate|          Yes|            9560|               0.0|       191|             360|            1|    Semiurban|         Y|
|LP001097|  Male|     No|         1|    Graduate|          Yes|            4692|               0.0|       106|             360|            1|        Rural|         N|
|LP001136|  Male|    Yes|         0|Not Graduate|          Yes|            4695|               0.0|        96|            NULL|            1|        Urban|         Y|
|LP001186|Female|    Yes|         1|    Graduate|          Yes|           11500|               0.0|       286|             360|            0|        Urban|         N|
|LP001207|  Male|    Yes|         0|Not Graduate|          Yes|            2609|            3449.0|       165|             180|            0|        Rural|         N|
|LP001238|  Male|    Yes|        3+|Not Graduate|          Yes|            7100|               0.0|       125|              60|            1|        Urban|         Y|
|LP001245|  Male|    Yes|         2|Not Graduate|          Yes|            1875|            1875.0|        97|             360|            1|    Semiurban|         Y|
|LP001253|  Male|    Yes|        3+|    Graduate|          Yes|            5266|            1774.0|       187|             360|            1|    Semiurban|         Y|
|LP001259|  Male|    Yes|         1|    Graduate|          Yes|            1000|            3022.0|       110|             360|            1|        Urban|         N|
|LP001264|  Male|    Yes|        3+|Not Graduate|          Yes|            3333|            2166.0|       130|             360|         NULL|    Semiurban|         Y|
|LP001266|  Male|    Yes|         1|    Graduate|          Yes|            2395|               0.0|      NULL|             360|            1|    Semiurban|         Y|
|LP001392|Female|     No|         1|    Graduate|          Yes|            7451|               0.0|      NULL|             360|            1|    Semiurban|         Y|
|LP001451|  Male|    Yes|         1|    Graduate|          Yes|           10513|            3850.0|       160|             180|            0|        Urban|         N|
|LP001469|  Male|     No|         0|    Graduate|          Yes|           20166|               0.0|       650|             480|         NULL|        Urban|         Y|
|LP001482|  Male|    Yes|         0|    Graduate|          Yes|            3459|               0.0|        25|             120|            1|    Semiurban|         Y|
|LP001491|  Male|    Yes|         2|    Graduate|          Yes|            3316|            3500.0|        88|             360|            1|        Urban|         Y|
|LP001504|  Male|     No|         0|    Graduate|          Yes|            6950|               0.0|       175|             180|            1|    Semiurban|         Y|
|LP001529|  Male|    Yes|         0|    Graduate|          Yes|            2577|            3750.0|       152|             360|            1|        Rural|         Y|
+--------+------+-------+----------+------------+-------------+----------------+------------------+----------+----------------+-------------+-------------+----------+
only showing top 20 rows
```

---

2. ***Join***

Joins product and sales datasets on product_id to combine related records.

ProductDf = spark.read.csv("/content/products.csv", header=True, inferSchema=True)

salesDf = spark.read.csv("/content/sales.csv", header=True, inferSchema=True)

ProductDf.join(salesDf, ProductDf.product_id == salesDf.product_id).show()

```
+----------+------------+-----------+
|product_id|product_name|   category|
+----------+------------+-----------+
|       101|      Laptop|Electronics|
|       102|    Keyboard|Electronics|
|       103|       Mouse|Electronics|
|       104|        Desk|  Furniture|
|       105|       Chair|  Furniture|
|       106|     Monitor|Electronics|
|       107|    Notebook| Stationery|
+----------+------------+-----------+

+-------+----------+--------+----------+
|sale_id|product_id|quantity| sale_date|
+-------+----------+--------+----------+
|      1|       101|       2|2023-01-01|
|      2|       102|       1|2023-01-02|
|      3|       103|       4|2023-01-03|
|      4|       108|       1|2023-01-04|
|      5|       109|       2|2023-01-05|
|      6|       101|       1|2023-01-06|
|      7|       104|       3|2023-01-07|
+-------+----------+--------+----------+

+----------+------------+-----------+-------+----------+--------+----------+
|product_id|product_name|   category|sale_id|product_id|quantity| sale_date|
+----------+------------+-----------+-------+----------+--------+----------+
|       101|      Laptop|Electronics|      6|       101|       1|2023-01-06|
|       101|      Laptop|Electronics|      1|       101|       2|2023-01-01|
|       102|    Keyboard|Electronics|      2|       102|       1|2023-01-02|
|       103|       Mouse|Electronics|      3|       103|       4|2023-01-03|
|       104|        Desk|  Furniture|      7|       104|       3|2023-01-07|
+----------+------------+-----------+-------+----------+--------+----------+
```

3. ***Simple Aggregations***

Calculates total, maximum, minimum, and average applicant income.

from pyspark.sql.functions import sum, max, min, avg

df.agg(sum("ApplicantIncome").alias("Total Income")).show()

df.agg(max("ApplicantIncome").alias("Max Income")).show()

df.agg(min("ApplicantIncome").alias("Min Income")).show()

df.agg(avg("ApplicantIncome").alias("Avg Income")).show()

Simple Aggregations

```
from pyspark.sql.functions import sum,max,min,avg
df.agg(sum("ApplicantIncome").alias ("Total Income")).show()
df.agg(max("ApplicantIncome").alias ("Max Income")).show()
df.agg(min("ApplicantIncome").alias ("Min Income")).show()
df.agg(avg("ApplicantIncome").alias ("Avg Income")).show()
```

```
+------------+
|Total Income|
+------------+
|     3317724|
+------------+

+----------+
|Max Income|
+----------+
|     81000|
+----------+

+----------+
|Min Income|
+----------+
|       150|
+----------+

+-----------------+
|       Avg Income|
+-----------------+
|5403.459283387622|
+-----------------+
```

4. ***GroupBy Aggregation***

Groups data by education level and calculates total income per group.

df.groupBy("Education").agg(sum("ApplicantIncome").alias("Total Income")).show()

GroupBy

```
[17] df.groupBy("Education").agg(sum("ApplicantIncome").alias ("Total Income")).show()
```

```
+------------+------------+
|   Education|Total Income|
+------------+------------+
|Not Graduate|      506156|
|    Graduate|     2811568|
+------------+------------+
```

5. ***Window Function***

Assigns a rank to each applicant's income within their property area using windowing.

from pyspark.sql.window import Window

from pyspark.sql.functions import rank

windowSpec = Window.partitionBy("Property_Area").orderBy(col("ApplicantIncome").desc())

df_ranked = df.withColumn("Income_Rank", rank().over(windowSpec))

df_ranked.select("Loan_ID", "Property_Area", "ApplicantIncome", "Income_Rank").show()

Window Function

```
from pyspark.sql.window import Window
from pyspark.sql.functions import rank
windowSpec = Window.partitionBy("Property_Area").orderBy(col("ApplicantIncome").desc())
df_ranked = df.withColumn("Income_Rank", rank().over(windowSpec))
df_ranked.select("Loan_ID", "Property_Area", "ApplicantIncome", "Income_Rank").show()
```

```
+--------+-------------+---------------+-----------+
| Loan_ID|Property_Area|ApplicantIncome|Income_Rank|
+--------+-------------+---------------+-----------+
|LP002317|        Rural|          81000|          1|
|LP001448|        Rural|          23803|          2|
|LP001922|        Rural|          20667|          3|
|LP001996|        Rural|          20233|          4|
|LP002191|        Rural|          19730|          5|
|LP002699|        Rural|          17500|          6|
|LP002527|        Rural|          16525|          7|
|LP002065|        Rural|          15000|          8|
|LP001859|        Rural|          14683|          9|
|LP001401|        Rural|          14583|         10|
|LP001100|        Rural|          12500|         11|
|LP001519|        Rural|          10000|         12|
|LP002050|        Rural|          10000|         12|
|LP002945|        Rural|           9963|         14|
|LP001935|        Rural|           9508|         15|
|LP002262|        Rural|           9504|         16|
|LP001647|        Rural|           9328|         17|
|LP002201|        Rural|           9323|         18|
|LP002255|        Rural|           9167|         19|
|LP002140|        Rural|           8750|         20|
+--------+-------------+---------------+-----------+
```

6. ***sortBy (RDD Transformation)***

Converts DataFrame to RDD and sorts applicants by income in descending order.

rdd = df.rdd

sorted_rdd = rdd.sortBy(lambda row: row["ApplicantIncome"], ascending=False)

for row in sorted_rdd.take(5):

print(row["Loan_ID"], row["ApplicantIncome"])

sortBy()

```
rdd=df.rdd
sorted_rdd = rdd.sortBy(lambda row: row["ApplicantIncome"], ascending=False)
for row in sorted_rdd.take(5):
    print(row["Loan_ID"], row["ApplicantIncome"])
```

```
LP002317 81000
LP002101 63337
LP001585 51763
LP001536 39999
LP001640 39147
```

Actions

1. ***collect()***

Retrieves the entire DataFrame into driver memory (use cautiously with large data).

print(df.collect())

Actions

collect()

```
[30] print(df.collect())
```

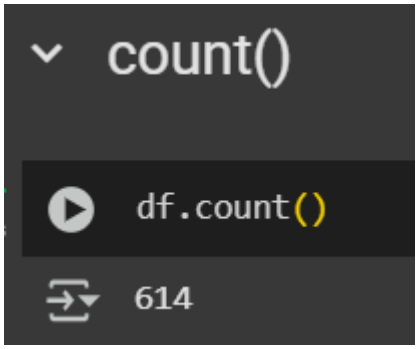
```
;='Y'), Row(Loan_ID='LP001003', Gender='Male', Married='Yes', Dependents='1', Education='Graduate', Self_Employed='No'
```

---

2. ***count()***

Returns the number of records in the DataFrame.

df.count()

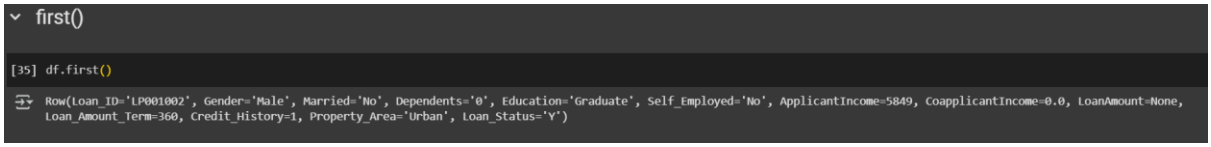


---

3. ***first()***

Retrieves the first row from the DataFrame.

df.first()



---

4. ***saveAsTextFile()***

Saves the RDD content as a text file to the specified location.

rdd.saveAsTextFile('file.txt')