

# 1. READ THE DATASET IN DATABRICKS COMMUNITY

The screenshot shows a Databricks Community notebook titled "SparkTest". The first command (Cmd 1) is executed, reading a CSV file from the FileStore. The output shows the Spark job completion and the creation of a DataFrame with 12 fields.

```
1 df=spark.read.format("csv")\
2   .option("header", "false")\
3   .option("inferSchema", "false")\
4   .option("mode", "FAILFAST")\
5   .load("/FileStore/tables/population.csv")
6
```

Command took 1.31 seconds -- by rahul04ar@gmail.com at 2/29/2024, 6:42:58 PM on Spark

Cmd 2

```
1 df.display()
```

Table

	_c5	_c6	_c7	_c8	_c9	_c10	_c11	_c12	_c13
30	555,339	541,867	976	91.33	529,037	561,997	21,081	52	22.8%
31	580,663	474,787	818	86.05	29,004	1,025,682	114	9,252	17.1%
32	323,070	287,507	890	81.42	455,962	151,726	7,096	86	12.4%

The screenshot shows the same Databricks Community notebook. The second command (Cmd 2) is executed, displaying the DataFrame. The output shows the Spark job completion and the display of the DataFrame with 36 rows and 10 columns.

```
1 df.display()
```

Table

	_c5	_c6	_c7	_c8	_c9	_c10	_c11	_c12	_c13
30	555,339	541,867	976	91.33	529,037	561,997	21,081	52	22.8%
31	580,663	474,787	818	86.05	29,004	1,025,682	114	9,252	17.1%
32	323,070	287,507	890	81.42	455,962	151,726	7,096	86	12.4%
33	202,871	177,710	876	86.63	244,411	135,533	8,249	46	6.7%
34	193,760	149,949	774	76.24	183,024	159,829	491	698	55.5%
35	150,301	92,946	618	87.10	60,331	182,580	112	2,169	53.5%
36	33,123	31,350	946	91.85	14,121	50,308	32	2,013	6.2%

36 rows | 0.49 seconds runtime

Command took 0.49 seconds -- by rahul04ar@gmail.com at 2/29/2024, 6:43:05 PM on Spark

1 flight\_df.show(5)

(1) Spark Jobs

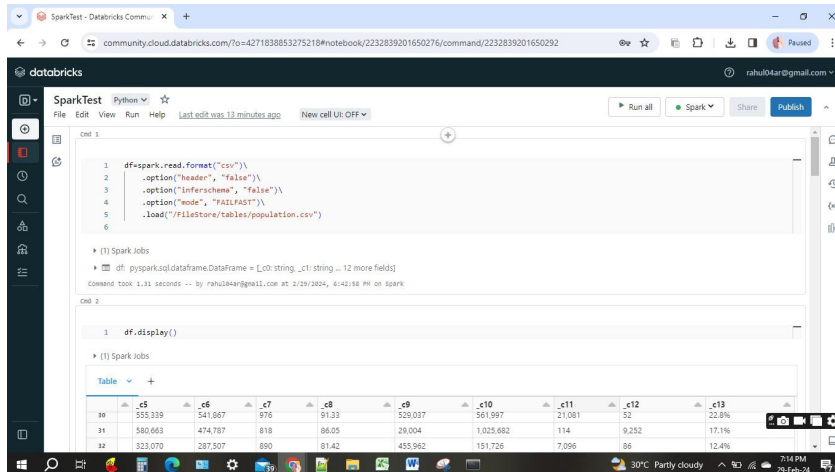
_c0	_c1	_c2	_c3	_c4	_c5	_c6	_c7	_c8	_c9	_c10	_c11
Rank	State	Capital	Population	% of Total Popula...	Males	Females	Sex Ratio	Literacy Rate (%)	Rural Population	Urban Population	Area (km²)
Density (1/km²)	Decadal Growth (%)										
1	Uttar Pradesh	Lucknow	199,812,341	16.50	104,480,510	95,331,831	912	67.68	155,111,022	44,470,455	240,928
828	20.1%										
2	Maharashtra	Mumbai	112,374,333	9.28	58,243,056	54,131,277	929	82.34	61,545,441	50,827,531	307,713
365	16.0%										
3	Bihar	Patna	104,099,452	8.60	54,278,157	49,821,295	918	61.80	92,075,028	11,729,609	94,163
1,102	25.1%										
4	West Bengal	Kolkata	91,276,115	7.54	46,809,027	44,467,088	950	76.26	62,213,676	29,134,060	88,752
1,030	13.9%										

only showing top 5 rows

Command took 0.51 seconds -- by rahul04ar@gmail.com at 2/29/2024, 6:43:10 PM on Spark

## 2. HOW MANY TYPES OF MODES WE HAVE IN SPARK?

1. **Failfast Mode:** This mode is used to instruct Spark to fail the entire operation (e.g., reading data into a DataFrame) if it encounters any malformed records or parsing errors. It stops processing the data immediately upon encountering the first error and throws an exception. This mode ensures data integrity but may result in the loss of valid records if the errors are not handled appropriately.
2. **Dropmalformed Mode:** In this mode, Spark skips and drops any rows that contain malformed records or parsing errors while reading data into a DataFrame. It allows you to continue processing the remaining valid records while ignoring the problematic ones. This mode can be useful when you want to prioritize processing valid data and handle errors separately.
3. **Permissive Mode:** Permissive mode is a combination of both failfast and dropmalformed modes. It attempts to parse all records in the input data, even if some records are malformed or contain parsing errors. Spark generates a DataFrame with a column containing the malformed records along with a new column indicating the parsing status (valid or invalid) of each record. This mode allows you to capture and handle parsing errors while still processing the valid data.



### 3. WHAT IS CLUSTER IN SPARK?

A Spark cluster is a group of interconnected computers (nodes) working together to process data. It consists of a master node, which coordinates tasks, and multiple worker nodes, where data processing occurs. Executors, running on worker nodes, execute tasks assigned by the master node. The driver program, running on the master node, coordinates the overall execution of a Spark application. Cluster managers allocate resources and manage the cluster infrastructure. Spark clusters enable distributed computing, allowing large datasets to be processed efficiently across multiple nodes.

### 4. WHAT IS TABLE IN SPARK ?

A table in Spark typically refers to structured data organized into rows and columns, represented using Data Frame or Dataset APIs. Tables can be created from various data sources and manipulated using Spark's rich functionality for querying and transforming structured data. They provide a convenient way to work with structured data in Spark, enabling tasks like data analysis, processing, and querying.

### 5. WHAT WOULD YOU DO IF YOU WANT TO SHOW THE HEADER WHILE SHOWING UP 5 RECORDS OF TABLE? WRITE THE CODE

```
dfheader=spark.read.format("csv")\
    .option("header", "true")\
    .option("inferSchema", "false")\
    .option("mode", "FAILFAST")\
    .load("/FileStore/tables/population.csv")

dfheader.show(5)
```

The screenshot shows a Databricks notebook interface. The top bar indicates the notebook is named 'SparkTest' and is using Python. The code cell contains the following Spark SQL query:

```
dfheader=spark.read.format("csv")\
    .option("header", "true")\
    .option("inferSchema", "false")\
    .option("mode", "FAILFAST")\
    .load("/FileStore/tables/population.csv")
dfheader.show(5)
```

The output shows the first 5 rows of the CSV file. The data is as follows:

Rank	State	Capital	Population	% of Total Population	Males	Females	Sex Ratio	Literacy Rate (%)	Rural Population	Urban Population	Area (km²)
1	Uttar Pradesh	Lucknow	199,812,341	16.50	104,480,510	95,331,831	912	67.68	155,111,022	44,470,455	240,928
2	Maharashtra	Mumbai	112,374,333	9.28	58,243,056	54,131,277	929	82.34	61,545,441	50,827,531	307,713
3	Bihar	Patna	104,099,452	8.60	54,278,157	49,821,295	918	61.80	92,075,028	11,729,609	94,161
4	West Bengal	Kolkata	91,276,115	7.54	46,809,027	44,467,088	950	76.26	62,213,676	29,134,060	88,197
5	Andhra Pradesh	Hyderabad	84,580,777	6.99	42,442,146	42,138,631	993	67.02	56,361,702	28,219,075	275,044

## 5. WHAT IS COUNT ? PERFORM IN SPARK

In the context of Apache Spark, "count" refers to an operation that calculates the number of elements in a dataset. It is a basic aggregation function used to determine the size of the dataset, whether it's an RDD (Resilient Distributed Dataset), DataFrame, or Dataset.

In Spark, the count() function is available as part of the DataFrame API and RDD API. When called on a DataFrame, it returns the number of rows in the DataFrame. Similarly, when called on an RDD, it returns the total number of elements in the RDD.

SparkTest - Databricks Commu x +

community.cloud.databricks.com/?o=4271838853275218#notebook/2232839201650276/command/2232839201650280

databricks rahul04ar@gmail.com

SparkTest Python ☆

File Edit View Run Help Last edit was 25 minutes ago New cell UI: OFF

Run all Spark Share Publish

Command took 0.11 seconds -- by rahul04ar@gmail.com at 2/29/2024, 6:43:36 PM on Spark

Cmd 7

Display number of rows using count( )

```
1 df.count()
```

(2) Spark Jobs

Out[23]: 36

Command took 0.49 seconds -- by rahul04ar@gmail.com at 2/29/2024, 6:45:38 PM on Spark

Cmd 8

Finding Null Occurrences of each column in data frame using Count( )

```
1 from pyspark.sql.functions import col,count,when
2
3 result = df.select ([count(when(col(c).isNull(),c)).alias(c) for c in df.columns])
4
5 display(result)
```

(2) Spark Jobs

result: pyspark.sql.dataframe.DataFrame = [\_c0: long, \_c1: long ... 12 more fields]

30°C Partly cloudy 7:26 PM 29-Feb-24

SparkTest - Databricks Commu x +

community.cloud.databricks.com/?o=4271838853275218#notebook/2232839201650276/command/2232839201650280

databricks rahul04ar@gmail.com

SparkTest Python ☆

File Edit View Run Help Last edit was 26 minutes ago New cell UI: OFF

Run all Spark Share Publish

Cmd 8

Finding Null Occurrences of each column in data frame using Count( )

```
1 from pyspark.sql.functions import col,count,when
2
3 result = df.select ([count(when(col(c).isNull(),c)).alias(c) for c in df.columns])
4
5 display(result)
```

(2) Spark Jobs

result: pyspark.sql.dataframe.DataFrame = [\_c0: long, \_c1: long ... 12 more fields]

Table +

	_c0	_c1	_c2	_c3	_c4	_c5	_c6	_c7	_c8	_c9	_c10
1	0	0	1	0	0	0	0	0	0	0	0

1 row | 1.74 seconds runtime Refreshed 27 minutes ago

Command took 1.74 seconds -- by rahul04ar@gmail.com at 2/29/2024, 7:08:29 PM on Spark

Cmd 9

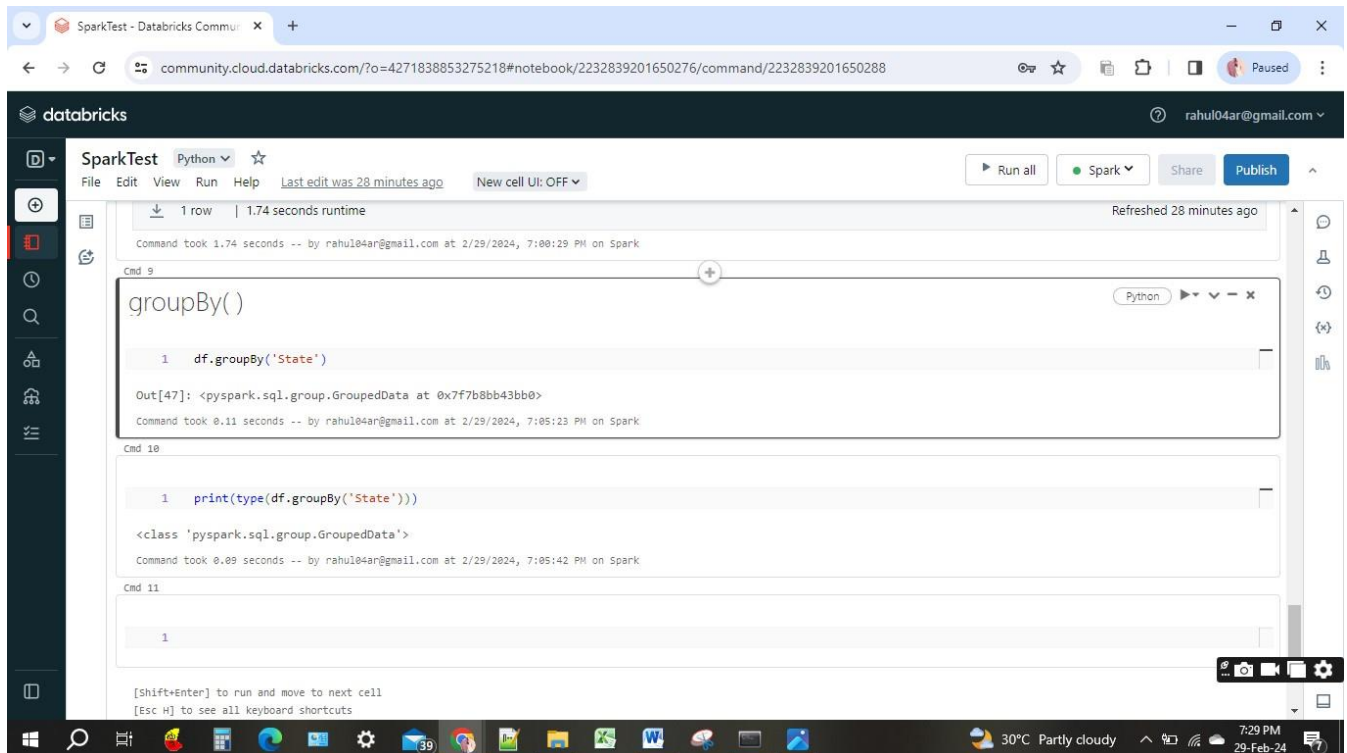
```
1 df.groupby('State')
```

30°C Partly cloudy 7:27 PM 29-Feb-24

## 6. WHAT IS GROUP BY ? PERFORM IN SPARK

In Apache Spark, "group by" is a data manipulation operation that groups the rows of a DataFrame or Dataset based on one or more columns. It is commonly used for aggregation tasks where you want to group data by certain attributes and perform aggregate functions (such as sum, count, average, etc.) on each group.

The `groupBy()` function in Spark is used to specify the column(s) by which you want to group the data. After grouping, you typically apply aggregation functions to compute summary statistics or derive new insights from the grouped data.



The screenshot shows a Databricks notebook interface with the following content:

- SparkTest** - Python
- File Edit View Run Help Last edit was 28 minutes ago New cell UI: OFF
- Run all Spark Share Publish
- 1 row | 1.74 seconds runtime Refreshed 28 minutes ago
- Command took 1.74 seconds -- by rahul04ar@gmail.com at 2/29/2024, 7:00:29 PM on Spark
- Cmd 9: `groupBy()`
- 1 `df.groupBy('State')`
- Out[47]: `<pyspark.sql.group.GroupedData at 0x7f7b8bb43bb0>`
- Command took 0.11 seconds -- by rahul04ar@gmail.com at 2/29/2024, 7:05:23 PM on Spark
- Cmd 10: `print(type(df.groupBy('State')))`
- `<class 'pyspark.sql.group.GroupedData'>`
- Command took 0.09 seconds -- by rahul04ar@gmail.com at 2/29/2024, 7:05:42 PM on Spark
- Cmd 11: `1`
- [Shift+Enter] to run and move to next cell
- [Esc H] to see all keyboard shortcuts
- 30°C Partly cloudy 7:29 PM 29-Feb-24