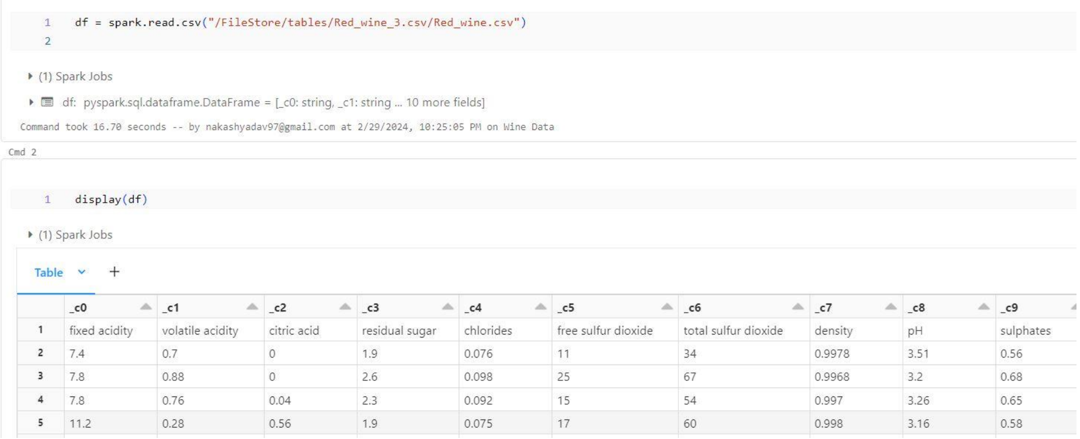
1. Reading the Dataset into Pyspark : To read any dataset into pyspark we should use “**spark.read.csv**”



```
df = spark.read.csv("/FileStore/tables/Red_wine_3.csv/Red_wine.csv")

```

▸ (1) Spark Jobs
▸ df: pyspark.sql.dataframe.DataFrame = [_c0: string, _c1: string ... 10 more fields]
Command took 16.70 seconds -- by nakashyadav97@gmail.com at 2/29/2024, 10:25:05 PM on Wine Data

Cmd 2

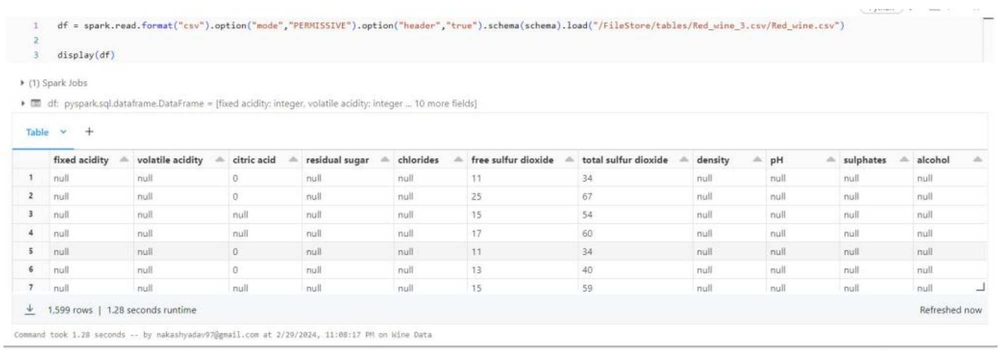
```
display(df)
```

▸ (1) Spark Jobs

Table ⌄ +

	_c0	_c1	_c2	_c3	_c4	_c5	_c6	_c7	_c8	_c9
1	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
2	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56
3	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68
4	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65
5	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58

2. Modes in the Spark.
   - permissive: Sets all fields to null when encountering a corrupted record.
   - failfast: Stops reading when encountering a corrupted record.
   - dropMalformed (default): Drops the corrupted records and continues reading.



```
df = spark.read.format("csv").option("mode","PERMISSIVE").option("header","true").schema(schema).load("/FileStore/tables/Red_wine_3.csv/Red_wine.csv")

display(df)
```

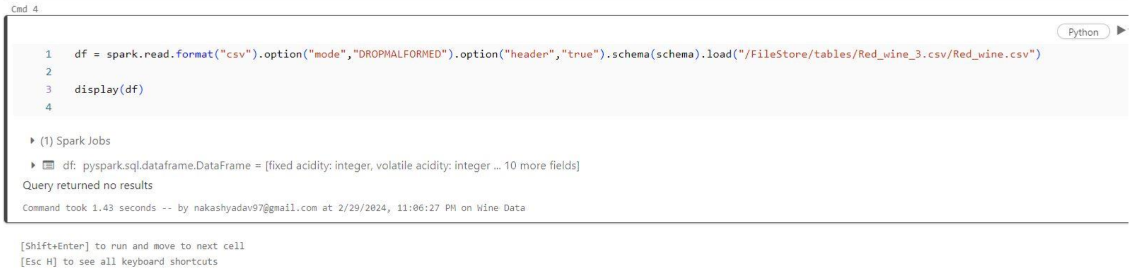
▸ (1) Spark Jobs
▸ df: pyspark.sql.dataframe.DataFrame = [fixed acidity: integer, volatile acidity: integer ... 10 more fields]

Table ⌄ +

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
1	null	null	0	null	null	11	34	null	null	null	null
2	null	null	0	null	null	25	67	null	null	null	null
3	null	null	null	null	null	15	54	null	null	null	null
4	null	null	null	null	null	17	60	null	null	null	null
5	null	null	0	null	null	11	34	null	null	null	null
6	null	null	0	null	null	13	40	null	null	null	null
7	null	null	null	null	null	15	59	null	null	null	null

1,599 rows | 1.28 seconds runtime   Refreshed now

Command took 1.28 seconds -- by nakashyadav97@gmail.com at 2/29/2024, 11:08:17 PM on Wine Data

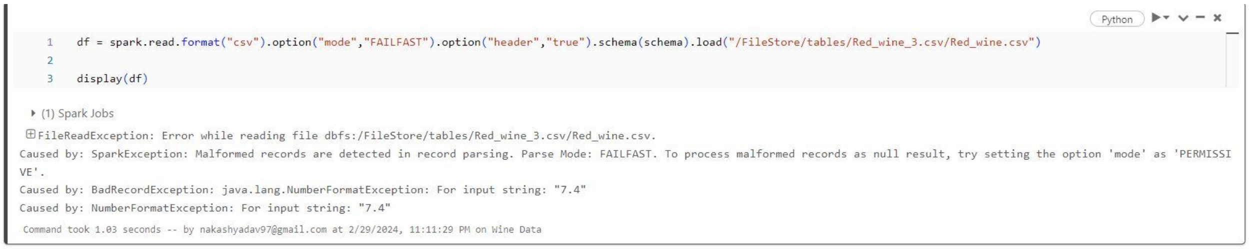


Cmd 4

```
df = spark.read.format("csv").option("mode","DROPMALFORMED").option("header","true").schema(schema).load("/FileStore/tables/Red_wine_3.csv/Red_wine.csv")

display(df)

```

▸ (1) Spark Jobs
▸ df: pyspark.sql.dataframe.DataFrame = [fixed acidity: integer, volatile acidity: integer ... 10 more fields]
Query returned no results
Command took 1.43 seconds -- by nakashyadav97@gmail.com at 2/29/2024, 11:06:27 PM on Wine Data

[Shift+Enter] to run and move to next cell
[Esc H] to see all keyboard shortcuts



```
df = spark.read.format("csv").option("mode","FAILFAST").option("header","true").schema(schema).load("/FileStore/tables/Red_wine_3.csv/Red_wine.csv")

display(df)
```

▸ (1) Spark Jobs

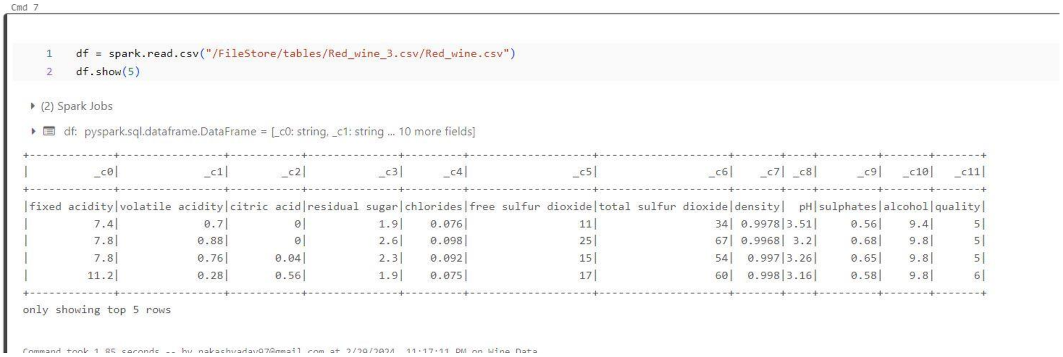
```
⊞FileReadException: Error while reading file dbfs:/FileStore/tables/Red_wine_3.csv/Red_wine.csv.
Caused by: SparkException: Malformed records are detected in record parsing. Parse Mode: FAILFAST. To process malformed records as null result, try setting the option 'mode' as 'PERMISSI
VE'.
Caused by: BadRecordException: java.lang.NumberFormatException: For input string: "7.4"
Caused by: NumberFormatException: For input string: "7.4"
```

Command took 1.03 seconds -- by nakashyadav97@gmail.com at 2/29/2024, 11:11:29 PM on Wine Data

3. Cluster : In Apache Spark, a cluster refers to a group of interconnected computers that work together to perform distributed processing of data. Spark is designed to scale horizontally, allowing you to process large datasets by distributing the workload across multiple nodes in a cluster.

4. Table : A table in Spark typically refers to a structured collection of data organized into rows and columns. Spark provides a SQL interface called Spark SQL, which allows you to query structured data using SQL-like syntax.

5. Header Code to show 5 records : To show the first 5 records of a DataFrame in PySpark, you can use the **show()** method.



Cmd 7

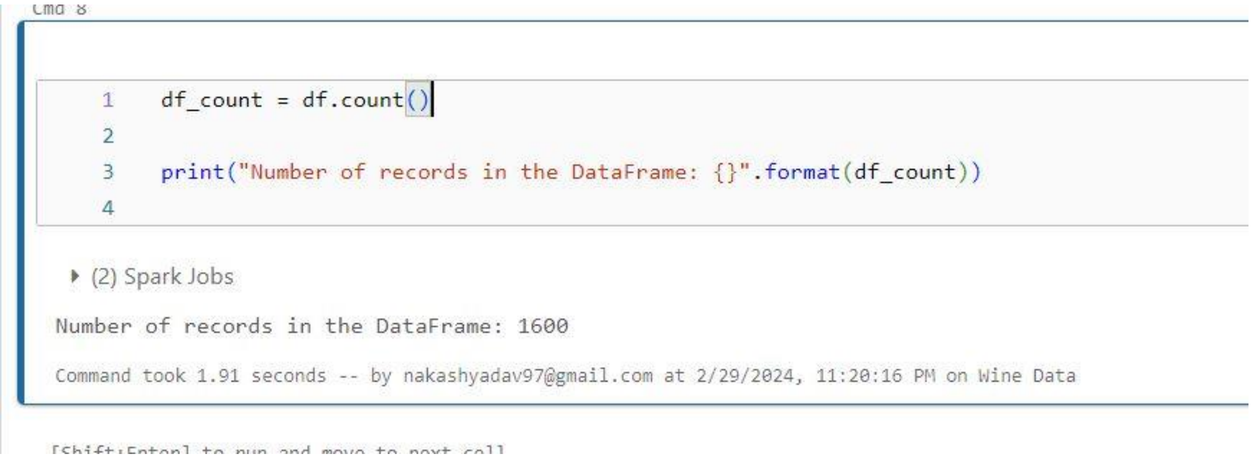
```python
df = spark.read.csv("/FileStore/tables/Red_wine_3.csv/Red_wine.csv")
df.show(5)
```

df: pyspark.sql.dataframe.DataFrame = [_c0: string, _c1: string ... 10 more fields]

```
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+-------+
|          _c0|             _c1|        _c2|           _c3|      _c4|                _c5|                 _c6|    _c7| _c8|      _c9|   _c10|   _c11|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+-------+
|fixed acidity|volatile acidity|citric acid|residual sugar|chlorides|free sulfur dioxide|total sulfur dioxide|density|  pH|sulphates|alcohol|quality|
|          7.4|             0.7|          0|           1.9|    0.076|                 11|                  34| 0.9978|3.51|     0.56|    9.4|      5|
|          7.8|            0.88|          0|           2.6|    0.098|                 25|                  67| 0.9968| 3.2|     0.68|    9.8|      5|
|          7.8|            0.76|       0.04|           2.3|    0.092|                 15|                  54|  0.997|3.26|     0.65|    9.8|      5|
|         11.2|            0.28|       0.56|           1.9|    0.075|                 17|                  60|  0.998|3.16|     0.58|    9.8|      6|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+-------+
only showing top 5 rows
```

Command took 1.85 seconds -- by nakashyadav97@gmail.com at 2/29/2024, 11:17:11 PM on Wine Data

6. Count Code : code reads the CSV file into the DataFrame **df** and then immediately calculates the count using the **count()** method.



Cmd 8

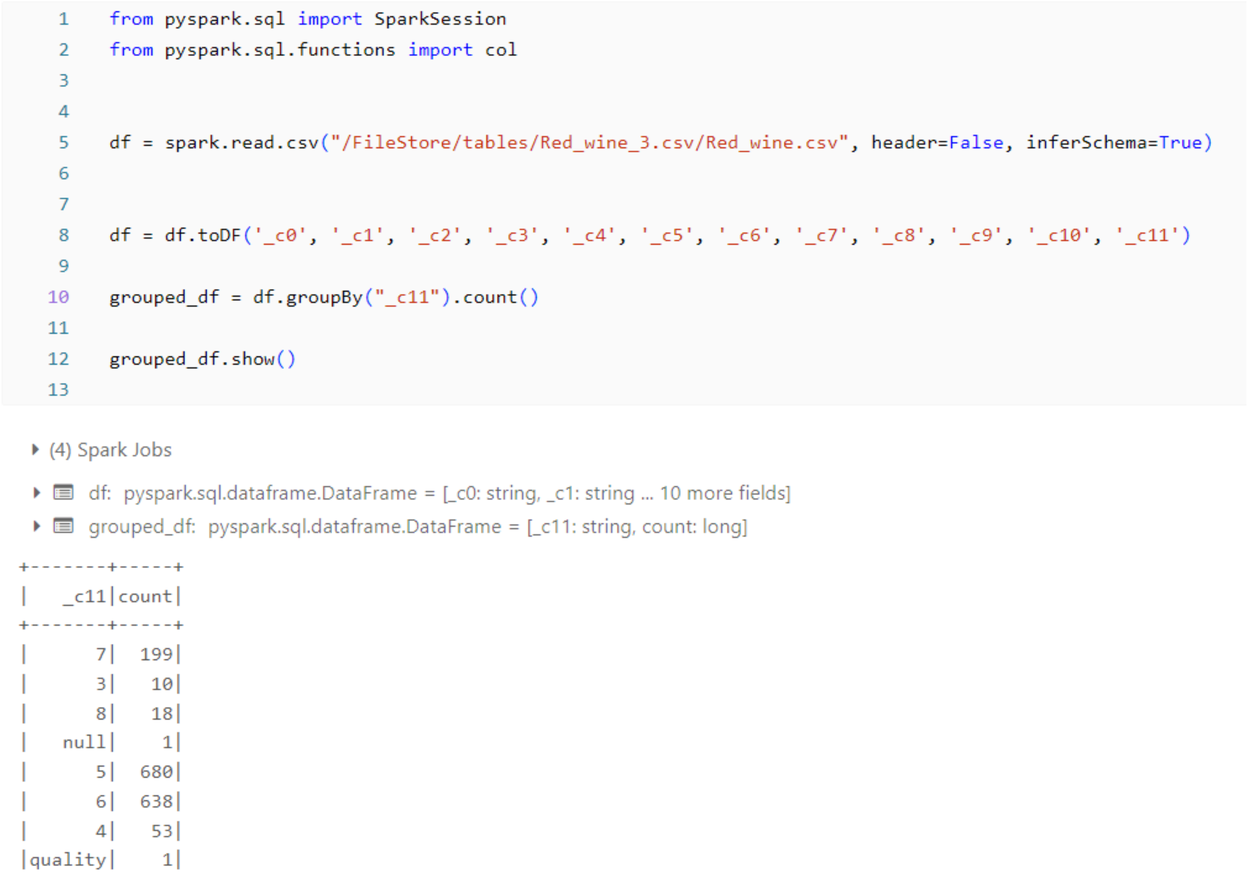
```python
df_count = df.count()

print("Number of records in the DataFrame: {}".format(df_count))

```

```
Number of records in the DataFrame: 1600
```

Command took 1.91 seconds -- by nakashyadav97@gmail.com at 2/29/2024, 11:20:16 PM on Wine Data

[Shift+Enter] to run and move to next cell

7. Groupby : The **count()** method is used to count the occurrences of each group. The resulting DataFrame grouped_**df** will have two columns: "quality" and "count", representing the distinct values in the "quality" column and the count of each value, respectively.



```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col


df = spark.read.csv("/FileStore/tables/Red_wine_3.csv/Red_wine.csv", header=False, inferSchema=True)


df = df.toDF('_c0', '_c1', '_c2', '_c3', '_c4', '_c5', '_c6', '_c7', '_c8', '_c9', '_c10', '_c11')

grouped_df = df.groupBy("_c11").count()

grouped_df.show()

```

```
▸ (4) Spark Jobs
▸ df:  pyspark.sql.dataframe.DataFrame = [_c0: string, _c1: string ... 10 more fields]
▸ grouped_df:  pyspark.sql.dataframe.DataFrame = [_c11: string, count: long]
+-------+-----+
|   _c11|count|
+-------+-----+
|      7|  199|
|      3|   10|
|      8|   18|
|   null|    1|
|      5|  680|
|      6|  638|
|      4|   53|
|quality|    1|
```