HANDLING BAD RECORDS AND FILES IN DATABRICKS USING BADRECORDSPATH

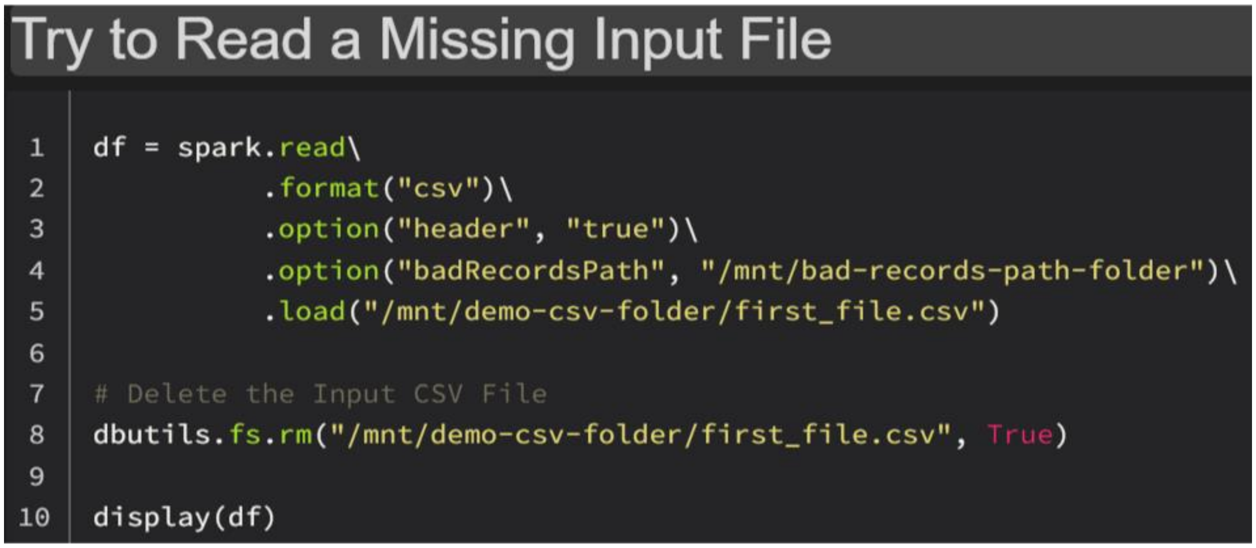
“badRecordsPath” Option

- When *reading Data* from a *File-Based Data Source*, *Apache Spark SQL* faces ***two*** typical *error cases* -
  - First, the *Files* may *Not* be *Readable* (for instance, the *Files* could be *Missing*, *Inaccessible* or *Corrupted*)
  - Second, ***even if*** the *Files* are *Processable*, some *Records* may *Not* be *Parsable*, might be due to *Syntax Errors* and *Schema Mismatch*.
- “*Azure Databricks*” ***provides*** a *Unified Interface* for ***handling*** “*Bad Records*” and “*Bad Files*” ***without interrupting*** *Spark Jobs*. It is ***possible*** to ***obtain*** the *Exception Records/Files* and ***retrieve*** the *Reason of Exception* from the “*Exception Logs*”, by ***setting*** the “***data source***” *Option* “*badRecordsPath*”. “*badRecordsPath*” specifies a *Path* to store “*Exception Files*” for *Recording* the *Information* about -
  - *Bad Records* for *CSV* and *JSON sources*.
  - *Bad Files* for ***all*** the *File-Based Built-In sources*, like - “*Parquet*”.

Limitation of “badRecordsPath” Option

- *Using* the “*badRecordsPath*” *Option* in a *File-Based Data Source* has a few important *Limitations* -
  - It is “*Non-Transactional*” and can ***lead to*** “*Inconsistent Results*”.
  - *When reading Files*, the “*Transient Errors*”, like - “*Network Connection Exception*”, “*IO Exception*” and so on, may ***occur***. These *Errors* are ***ignored***, and ***recorded*** under the “*badRecordsPath*”, and *Apache Spark* will ***continue*** to ***run*** the *Tasks*.

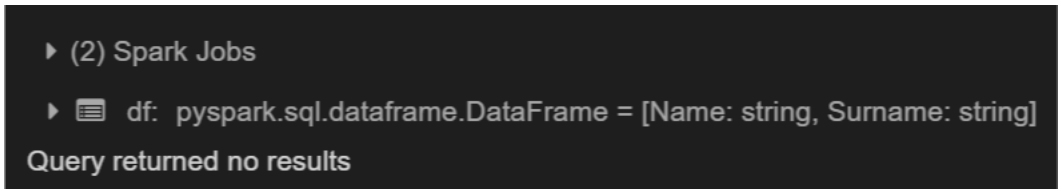
Example of "badRecordsPath" Option for Bad File (Missing File)



Try to Read a Missing Input File

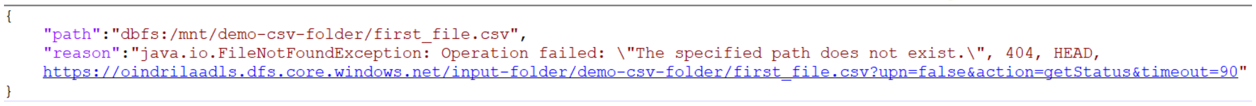
```
df = spark.read\
          .format("csv")\
          .option("header", "true")\
          .option("badRecordsPath", "/mnt/bad-records-path-folder")\
          .load("/mnt/demo-csv-folder/first_file.csv")

# Delete the Input CSV File
dbutils.fs.rm("/mnt/demo-csv-folder/first_file.csv", True)

display(df)
```

Output -



```
▸ (2) Spark Jobs
▸ df: pyspark.sql.dataframe.DataFrame = [Name: string, Surname: string]
Query returned no results
```

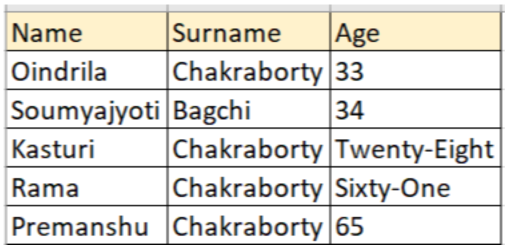
- In the above example, since "*df.show()*" is ***unable*** to ***find*** the ***Input File***, ***Apache Spark creates*** an "***Exception File***" in "***JSON Format***" to ***record*** the ***Error***. In this case, ***for*** the ***Missing Input File***, the ***Path*** of the ***generated*** "***Exception File***" is ***"/mnt/bad-records-path-folder/20220103T174849/bad_files/part-00000-a9b05b0e-75ba-4797-a319-4c82f6c2da1a***" -
  - The "***Exception File***", i.e., "*part-00000-a9b05b0e-75ba-4797-a319-4c82f6c2da1a*" is ***present inside*** the specified "***badRecordsPath***" ***Directory***, i.e., "*/mnt/bad-records-path-folder*".
  - "*20220103T174849*" is the "***Creation Time***" of the "***DataFrameReader***" that is displayed.
  - "*bad_files*" is the "***Exception Type***".
  - The "***Exception File***", i.e., "*part-00000-a9b05b0e-75ba-4797-a319-4c82f6c2da1a*" is a ***File*** that ***contains*** a ***JSON Record***, which has the ***Path*** of the ***Bad File***, and the "***Exception***" or "***Reason***" ***Message***.



```
{
    "path":"dbfs:/mnt/demo-csv-folder/first_file.csv",
    "reason":"java.io.FileNotFoundException: Operation failed: \"The specified path does not exist.\", 404, HEAD,
    https://oindrilaadls.dfs.core.windows.net/input-folder/demo-csv-folder/first_file.csv?upn=false&action=getStatus&timeout=90"
}
```

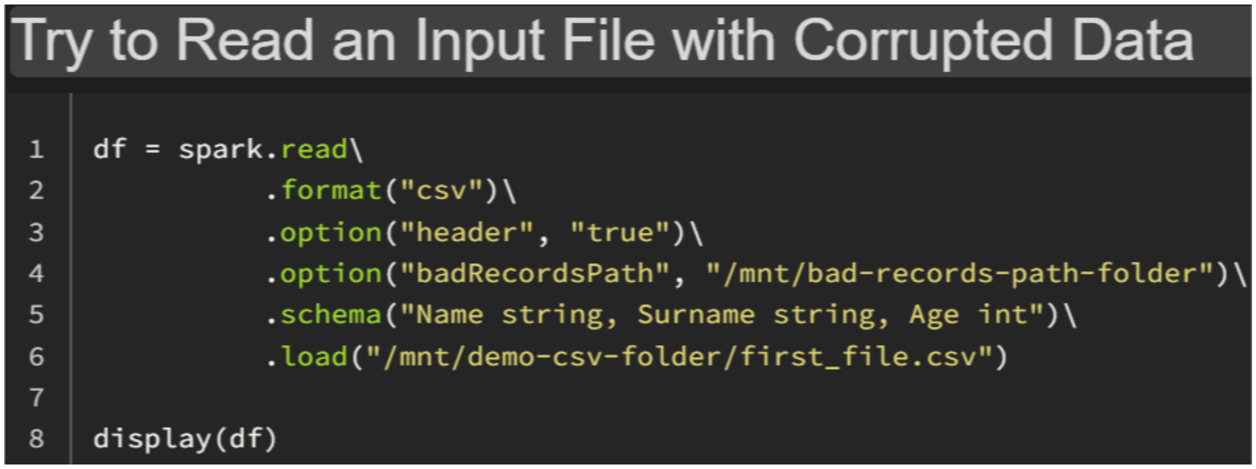
Example of “badRecordsPath” Option for Bad Records

- The *Records* in the *CSV File*, *including Corrupted Records*, are following -



Name	Surname	Age
Oindrila	Chakraborty	33
Soumyajyoti	Bagchi	34
Kasturi	Chakraborty	Twenty-Eight
Rama	Chakraborty	Sixty-One
Premanshu	Chakraborty	65

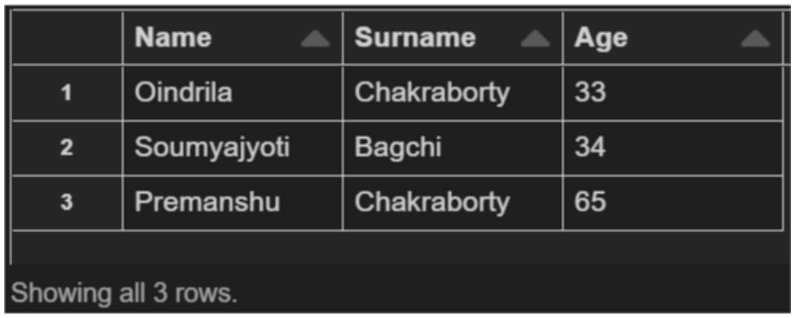
- The *code* to *read* and *display* the *Records* of the *CSV File* is following -



Try to Read an Input File with Corrupted Data

```
df = spark.read\
          .format("csv")\
          .option("header", "true")\
          .option("badRecordsPath", "/mnt/bad-records-path-folder")\
          .schema("Name string, Surname string, Age int")\
          .load("/mnt/demo-csv-folder/first_file.csv")

display(df)
```

Output -

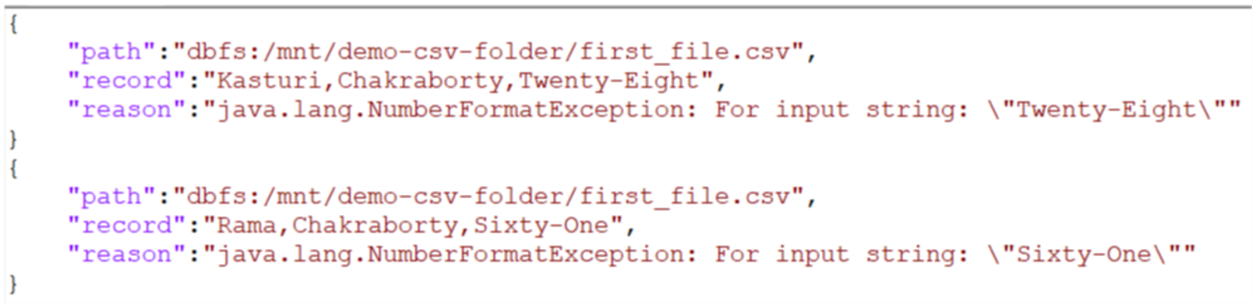


	Name	Surname	Age
1	Oindrila	Chakraborty	33
2	Soumyajyoti	Bagchi	34
3	Premanshu	Chakraborty	65

Showing all 3 rows.

- In this example, the *DataFrame contains* only the *three Records*, *matching* the provided *Schema*.

- For the ***two Bad Records***, ***not matching*** the provided ***Schema***, the ***Path*** of the ***generated*** "***Exception File***" is "***/mnt/bad-records-path-folder/20220103T184743/bad_records/part-00000-f0eeb7c2-7049-4f72-9b04-6ad87e348398***" -
  - The "***Exception File***", i.e., "***part-00000-f0eeb7c2-7049-4f72-9b04-6ad87e348398***" is ***present inside*** the specified "***badRecordsPath***" ***Directory***, i.e., "***/mnt/bad-records-path-folder***".
  - "***20220103T184743***" is the "***Creation Time***" of the "***DataFrameReader***" that is displayed.
  - "***bad_records***" is the "***Exception Type***".
  - The "***Exception File***", i.e., "***part-00000-f0eeb7c2-7049-4f72-9b04-6ad87e348398***" is a ***JSON File*** that ***contains*** -
    - ✓ the ***Bad Record***.
    - ✓ the ***Path*** of the "***Input File***" ***containing*** the ***Bad Record***.
    - ✓ the "***Exception***" or "***Reason***" ***Message***.
- ***After*** the "***Exception File***" is ***located***, a "***JSON Reader***" can be ***used*** to ***Process*** the "***Exception File***".

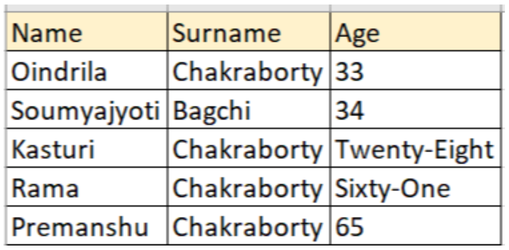


```
{
    "path":"dbfs:/mnt/demo-csv-folder/first_file.csv",
    "record":"Kasturi,Chakraborty,Twenty-Eight",
    "reason":"java.lang.NumberFormatException: For input string: \"Twenty-Eight\""
}
{
    "path":"dbfs:/mnt/demo-csv-folder/first_file.csv",
    "record":"Rama,Chakraborty,Sixty-One",
    "reason":"java.lang.NumberFormatException: For input string: \"Sixty-One\""
}
```

HANDLING BAD RECORDS IN DATABRICKS USING PARSING MODES

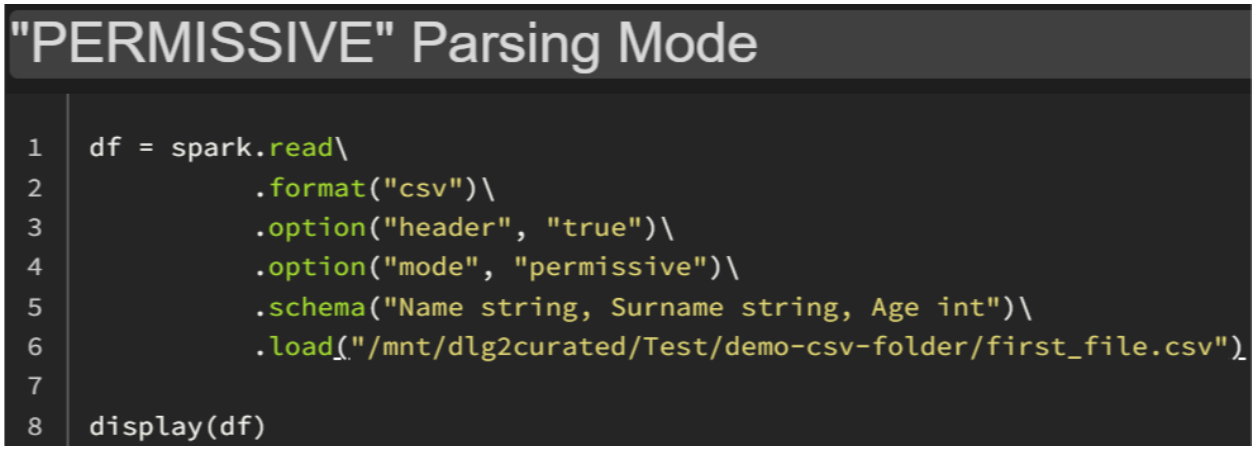
Correctness of the Data

- When *reading Data* from a *File-Based Data Source* with specified *Schema*, it is *possible* that the *Data* in the *Files* does *Not Match* the *Schema*. For example, a *field* "*Age*" *containing Integers*, will *Not Parse* as *Strings*.



Name	Surname	Age
Oindrila	Chakraborty	33
Soumyajyoti	Bagchi	34
Kasturi	Chakraborty	Twenty-Eight
Rama	Chakraborty	Sixty-One
Premanshu	Chakraborty	65

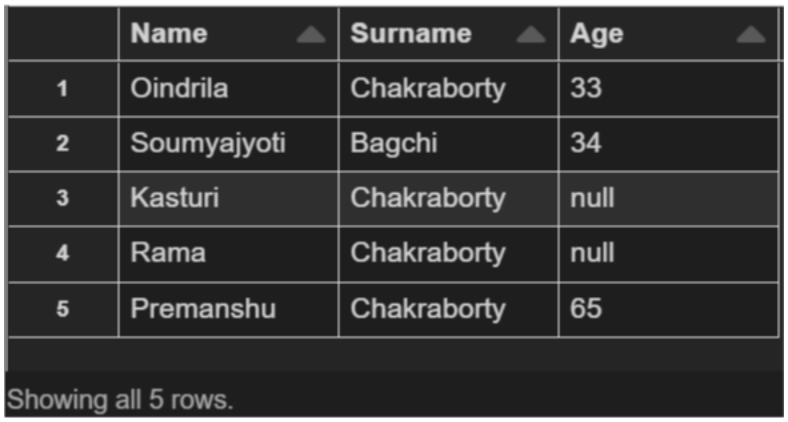
- The consequences depend on the *Mode* that the *Parser* runs on. To *set* the "*Mode*", the "*mode*" *Option* is *used* -
  - *PERMISSIVE (Default)* - In "*Permissive*" *Mode*, "*NULLs*" are *inserted* for *Fields* that could *Not* be *Parsed correctly*.



"PERMISSIVE" Parsing Mode

```
df = spark.read\
          .format("csv")\
          .option("header", "true")\
          .option("mode", "permissive")\
          .schema("Name string, Surname string, Age int")\
          .load("/mnt/dlg2curated/Test/demo-csv-folder/first_file.csv")

display(df)
```

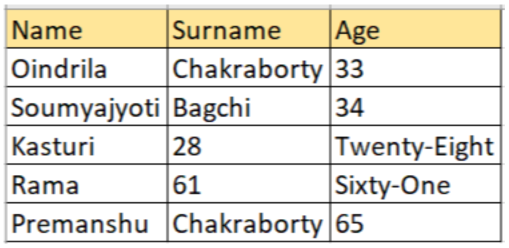
Output -



	Name	Surname	Age
1	Oindrila	Chakraborty	33
2	Soumyajyoti	Bagchi	34
3	Kasturi	Chakraborty	null
4	Rama	Chakraborty	null
5	Premanshu	Chakraborty	65

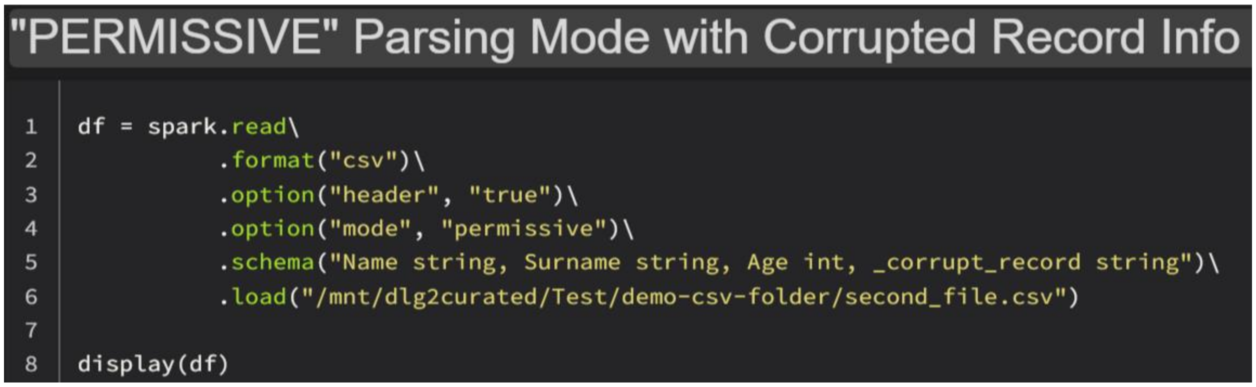
Showing all 5 rows.

In the "*Permissive*" *Mode*, it is *possible* to *inspect* the *Rows* that could *not* be *Parsed correctly*. To do that, "*_corrupt_record*" *Column* can be *added* to the *Schema*.
In the below CSV File, the *field* "*Surname*" *containing Strings*, will *Not Parse* as *Integers*. Similarly, the *field* "*Age*" *containing Integers*, will *Not Parse* as *Strings*.



Name	Surname	Age
Oindrila	Chakraborty	33
Soumyajyoti	Bagchi	34
Kasturi	28	Twenty-Eight
Rama	61	Sixty-One
Premanshu	Chakraborty	65

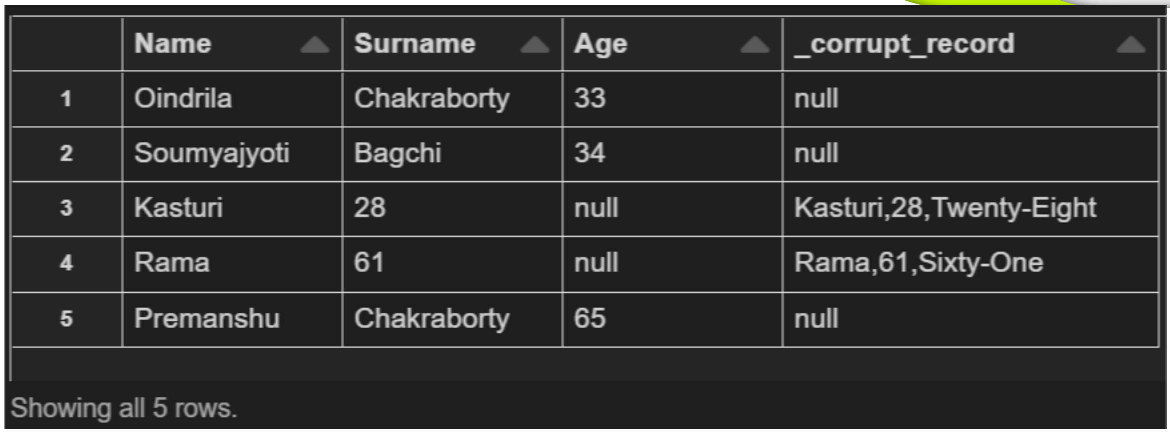
The *code*, to *read* the *CSV File* using "*Permissive*" *Mode* with the *incorrectly Parsed Data* as another *Column*, is as following -



"PERMISSIVE" Parsing Mode with Corrupted Record Info

```
df = spark.read\
          .format("csv")\
          .option("header", "true")\
          .option("mode", "permissive")\
          .schema("Name string, Surname string, Age int, _corrupt_record string")\
          .load("/mnt/dlg2curated/Test/demo-csv-folder/second_file.csv")

display(df)
```

Output -

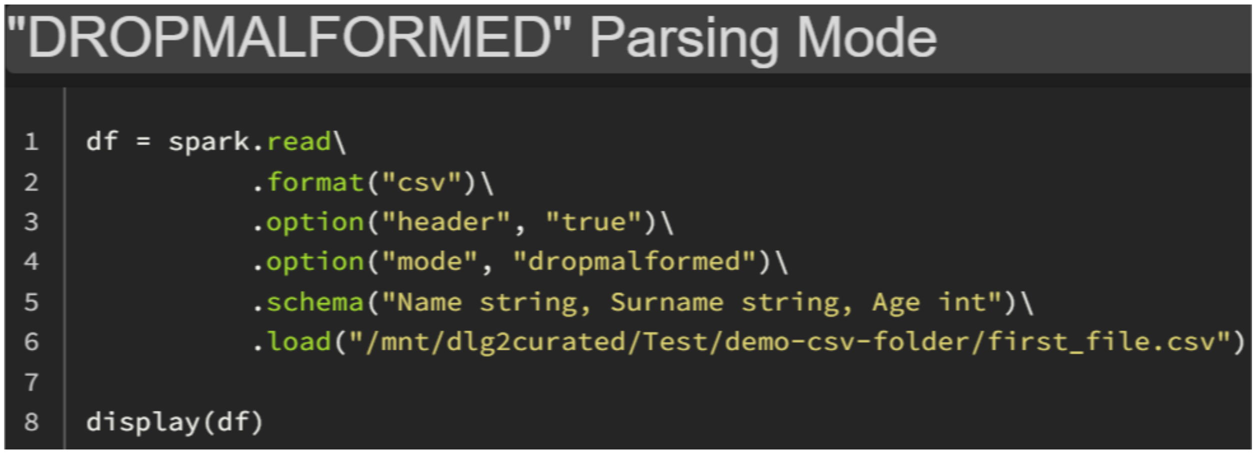


	Name	Surname	Age	_corrupt_record
1	Oindrila	Chakraborty	33	null
2	Soumyajyoti	Bagchi	34	null
3	Kasturi	28	null	Kasturi,28,Twenty-Eight
4	Rama	61	null	Rama,61,Sixty-One
5	Premanshu	Chakraborty	65	null

Showing all 5 rows.

The "*_corrupt_record*" *Column contains NULL* for the *Rows*, *having no Malformed Data*.
On the other hand, the "*_corrupt_record*" *Column contains all* the *Malformed Data*, *separated by Comma* (*;*) for the *Rows*, *having Malformed Data*.

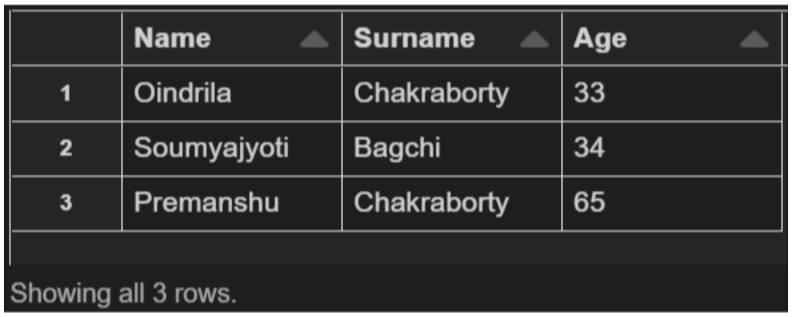
- <u>*DROPMALFORMED*</u> - In "*Dropmalformed*" *Mode*, the lines that contain *Fields* that could *Not* be *Parsed correctly*, are *Dropped*.



"DROPMALFORMED" Parsing Mode

```
df = spark.read\
          .format("csv")\
          .option("header", "true")\
          .option("mode", "dropmalformed")\
          .schema("Name string, Surname string, Age int")\
          .load("/mnt/dlg2curated/Test/demo-csv-folder/first_file.csv")

display(df)
```

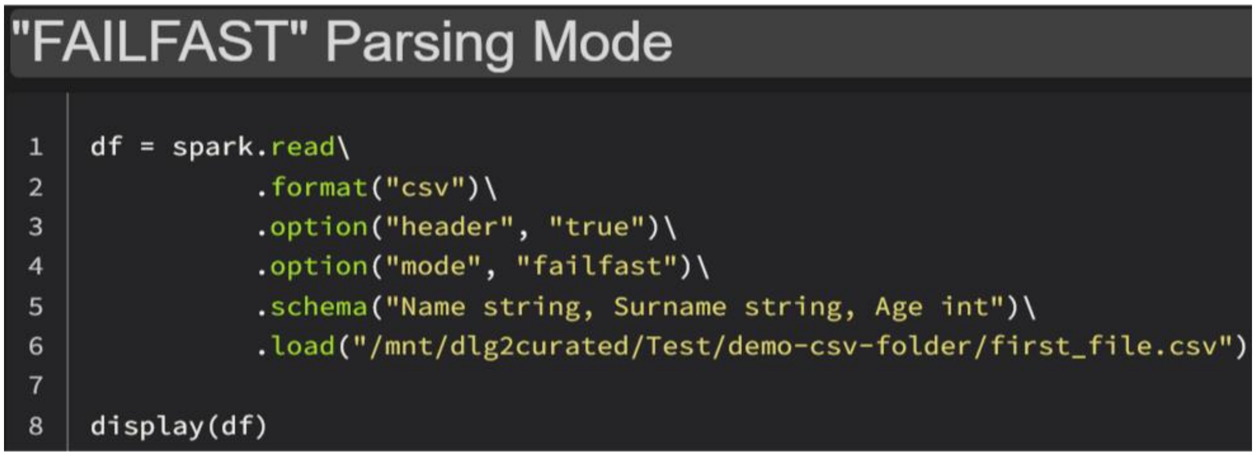
Output -



	Name	Surname	Age
1	Oindrila	Chakraborty	33
2	Soumyajyoti	Bagchi	34
3	Premanshu	Chakraborty	65

Showing all 3 rows.

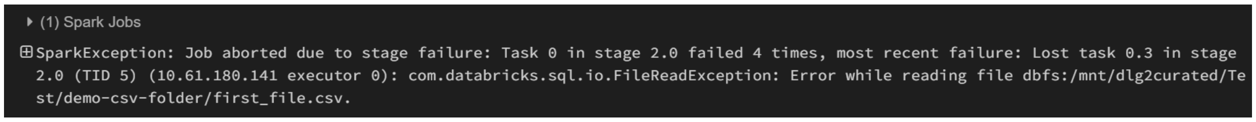
- ***FAILFAST*** - In "***Failfast***" ***Mode***, ***Apache Spark aborts*** the ***reading*** with ***Exception***, ***if*** any ***Malformed Data*** is ***found***.



"FAILFAST" Parsing Mode

```
df = spark.read\
          .format("csv")\
          .option("header", "true")\
          .option("mode", "failfast")\
          .schema("Name string, Surname string, Age int")\
          .load("/mnt/dlg2curated/Test/demo-csv-folder/first_file.csv")

display(df)
```

Output -



```
▸ (1) Spark Jobs
⊞SparkException: Job aborted due to stage failure: Task 0 in stage 2.0 failed 4 times, most recent failure: Lost task 0.3 in stage
 2.0 (TID 5) (10.61.180.141 executor 0): com.databricks.sql.io.FileReadException: Error while reading file dbfs:/mnt/dlg2curated/Te
 st/demo-csv-folder/first_file.csv.
```