Writing and Adding data in Pyspark Dataframes

Table of Contents

Note → Arrow  is used to go back to "table of contents".

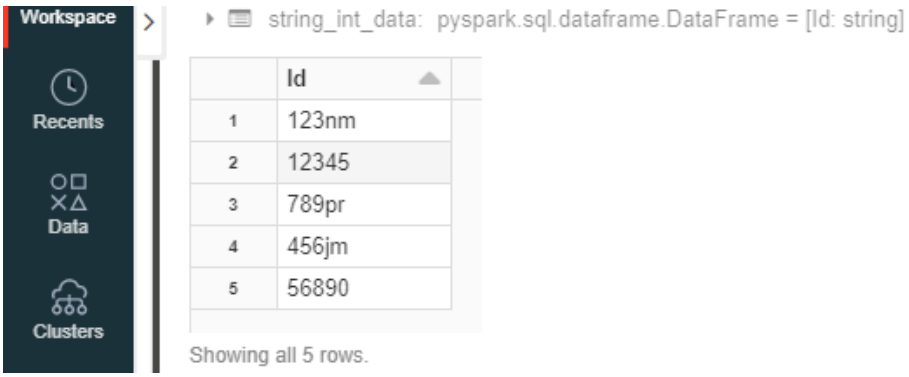
Importing libraries

```
from pyspark.sql.types import *

from pyspark.sql.functions import *

import datetime
```

Adding new data to a dataframe using union method

Create a dataframe -

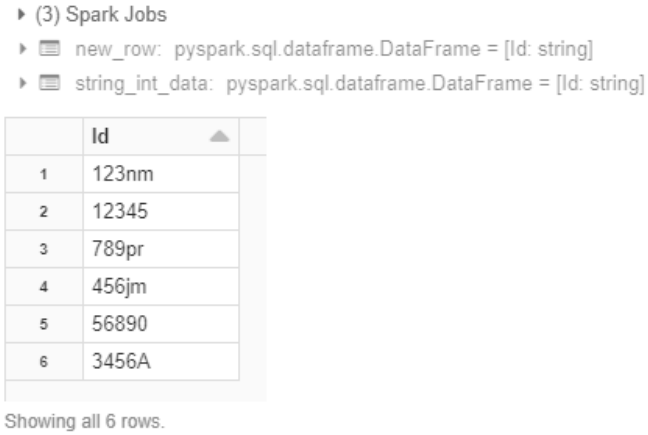
```
string_int_data = spark.createDataFrame([("123nm",), ("12345",), ("789pr",), ("456jm",), ("56890",)], ["Id",])

display(string_int_data)
```



Creating a new row and adding it to the dataframe –

union() - joins two dataframes row wise

```
new_row = spark.createDataFrame([("3456A",)],  ["Id",])

string_int_data = string_int_data.union(new_row)

display(string_int_data)
```



Setting the partition type to dynamic

```
spark.conf.set("spark.sql.sources.partitionOverwriteMode", "dynamic")
```

Creating a dataframe and writing it into a parquet file (to be used later)

Loading data into dataframe using csv format with proper datatypes

```
data = spark.read.csv("dbfs:/FileStore/tables/Date_Item_Data.txt",  header=True)

data = (data

    .withColumn("Item_id", col("Item_id").cast(IntegerType()))

    .withColumn("Amount", col("Amount").cast(IntegerType()))
```

```
    .withColumn("Date", to_date(col("Date"), "yyyy-MM-dd")))
display(data)
```

Writing data into parquet file without partitioning

```
(data.write
 .mode("overwrite")
 .parquet("dbfs:/FileStore/tables/ItemDateNoPartition.parquet"))
display(spark.read.parquet("dbfs:/FileStore/tables/ItemDateNoPartition.parquet"))
```

By overwrite mode, we erase the previous data and write the new data.

	Item_id	Amount	Date
1	1	250	2020-12-05
2	2	456	2020-12-10
3	3	500	2020-12-15
4	4	700	2020-12-15
5	5	200	2020-12-20
6	6	432	2020-12-25
7	7	567	2020-12-25
8	8	675	2020-12-25
9	9	230	2021-01-02
10	10	455	2021-01-04

Showing all 10 rows.

Writing data into parquet file with partitioning by date column

```
(data.write.partitionBy("Date")
 .mode("overwrite")
 .parquet("dbfs:/FileStore/tables/ItemDate.parquet"))
display(spark.read.parquet("dbfs:/FileStore/tables/ItemDate.parquet"))
```

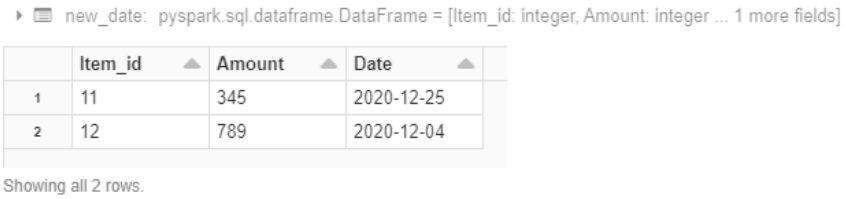
	Item_id	Amount	Date
1	1	250	2020-12-05
2	2	456	2020-12-10
3	3	500	2020-12-15
4	4	700	2020-12-15
5	5	200	2020-12-20
6	6	432	2020-12-25
7	7	567	2020-12-25
8	8	675	2020-12-25
9	9	230	2021-01-02
10	10	455	2021-01-04

Showing all 10 rows.

Writing data into parquet file without partitioning

Creating new rows of data -

```
new_date = (spark

        .createDataFrame([(11, 345, datetime.date(2020,12,25)),(12, 789, datetime.date(2020, 12, 4))], ["Item_id","Amount","Date"])

        .withColumn("Item_id", col("Item_id").cast(IntegerType()))

        .withColumn("Amount", col("Amount").cast(IntegerType()))

       )

display(new_date)
```



new_date: pyspark.sql.dataframe.DataFrame = [Item_id: integer, Amount: integer ... 1 more fields]

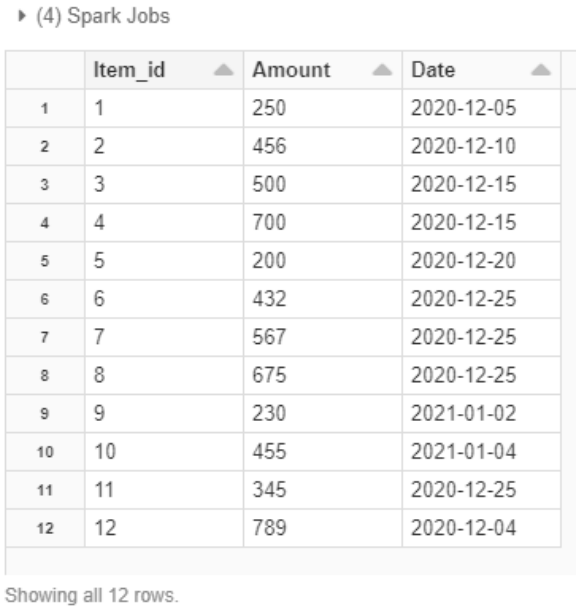
	Item_id	Amount	Date
1	11	345	2020-12-25
2	12	789	2020-12-04

Showing all 2 rows.

Writing data in append mode –

Append mode – add the new data to previous data(if exists), otherwise write the new data.

```
new_date.write.mode("append").save("dbfs:/FileStore/tables/ItemDateNoPartition.parquet")

display(spark.read.parquet("dbfs:/FileStore/tables/ItemDateNoPartition.parquet"))
```



(4) Spark Jobs

	Item_id	Amount	Date
1	1	250	2020-12-05
2	2	456	2020-12-10
3	3	500	2020-12-15
4	4	700	2020-12-15
5	5	200	2020-12-20
6	6	432	2020-12-25
7	7	567	2020-12-25
8	8	675	2020-12-25
9	9	230	2021-01-02
10	10	455	2021-01-04
11	11	345	2020-12-25
12	12	789	2020-12-04

Showing all 12 rows.

Writing data in overwrite mode –

Overwrite mode – erase the previous data and write the new one

```
new_date.write.mode("overwrite").save("dbfs:/FileStore/tables/ItemDateNoPartition.parquet")

display(spark.read.parquet("dbfs:/FileStore/tables/ItemDateNoPartition.parquet"))
```

▸ (4) Spark Jobs

	Item_id	Amount	Date
1	11	345	2020-12-25
2	12	789	2020-12-04

Showing all 2 rows.

Writing data into parquet file with partitioning

Creating new rows of data

```
new_date = (spark

        .createDataFrame([(11, 345, datetime.date(2020,12,25))],
["Item_id","Amount","Date"])

        .withColumn("Item_id", col("Item_id").cast(IntegerType()))

        .withColumn("Amount", col("Amount").cast(IntegerType()))

       )

display(new_date)
```

	Item_id	Amount	Date
1	11	345	2020-12-25

Showing all 1 rows.

Writing data in append mode in a particular partition -

Check the file path carefully.

```
new_date.write.mode("append").save("dbfs:/FileStore/tables/ItemDate.parquet/Date=2020
-12-25")

display(spark.read.parquet("dbfs:/FileStore/tables/ItemDate.parquet"))
```

	Item_id	Amount	Date
1	11	345	2020-12-25
2	6	432	2020-12-25
3	7	567	2020-12-25
4	8	675	2020-12-25
5	3	500	2020-12-15
6	4	700	2020-12-15
7	1	250	2020-12-05
8	10	455	2021-01-04
9	9	230	2021-01-02
10	5	200	2020-12-20
11	2	456	2020-12-10

Showing all 11 rows.

Writing data in overwrite mode in a particular partition –

```
new_date.write.mode("overwrite").save("dbfs:/FileStore/tables/ItemDate.parquet/Date=2020-12-25")

display(spark.read.parquet("dbfs:/FileStore/tables/ItemDate.parquet"))
```

Check the output for date “2020-12-25”.

▸ (5) Spark Jobs

	Item_id	Amount	Date
1	11	345	2020-12-25
2	3	500	2020-12-15
3	4	700	2020-12-15
4	1	250	2020-12-05
5	10	455	2021-01-04
6	9	230	2021-01-02
7	5	200	2020-12-20
8	2	456	2020-12-10

Showing all 8 rows.

Keep Learning!!!

Nidhi Mantri

Specialist Programmer – Big Data