BDML CT3 PROJECT

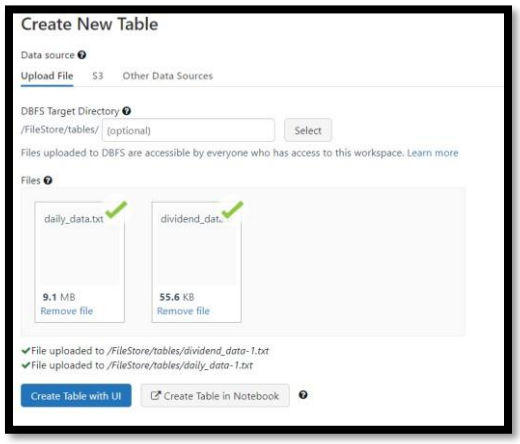
Shafeena Farheen

Performing the following tasks based on the dataset given.

Problem Statement:
A dataset from New York Stock Exchange archives containing two files is given. The file daily_data.txt contains daily details of a few companies for the years 1970 to 2010. The file dividend_data.txt contains dividends data for this period. Both the files are in tab-delimited format and contain headers giving the name of each column. We are writing a PySpark application to perform the following tasks. Using only Spark SQL's dataframe functions wherever specified. Otherwise using SQL statements by creating a temp view from the dataframe as per your preference.

Create cluster as follows:

To the tables add the two datasets and then create a new notebook:



---

1. Create a data frame from the daily data file. Display the schema of the data frame

1. Create a dataframe from the daily data file. Display the schema of the dataframe.

```
# The following three lines are required if you are running PySpark using Jupyter notebook with Python3 as kernel
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Python Spark SQL example").getOrCreate()
```

Command took 0.17 seconds -- by priya99ramesh@gmail.com at 12/22/2023, 11:11:57 AM on glsrm

Cmd 2

```
dailyDF = spark.read.load("/FileStore/tables/daily_data.txt", format="csv", sep="\t", inferSchema="true",header="true")
dividendDF = spark.read.load("/FileStore/tables/dividend_data.txt", format="csv", sep="\t", inferSchema="true",header="true")
```

▸ (4) Spark Jobs

▸ dailyDF: pyspark.sql.dataframe.DataFrame = [exchange: string, stock_symbol: string ... 7 more fields]

▸ dividendDF: pyspark.sql.dataframe.DataFrame = [exchange: string, stock_symbol: string ... 2 more fields]

Command took 2.77 seconds -- by priya99ramesh@gmail.com at 12/22/2023, 11:37:46 AM on glsrm

Cmd 3

```
dailyDF.printSchema()
```

```
root
 |-- exchange: string (nullable = true)
 |-- stock_symbol: string (nullable = true)
 |-- date: date (nullable = true)
 |-- open_price: double (nullable = true)
 |-- high_price: double (nullable = true)
 |-- low_price: double (nullable = true)
 |-- close_price: double (nullable = true)
 |-- volume: integer (nullable = true)
 |-- adjusted_close_price: double (nullable = true)
```

Command took 0.08 seconds -- by priya99ramesh@gmail.com at 12/22/2023, 11:37:52 AM on glsrm

Inference:

- Import PySpark: The first line (import pyspark) imports the PySpark library, which is the Python API for Apache Spark.

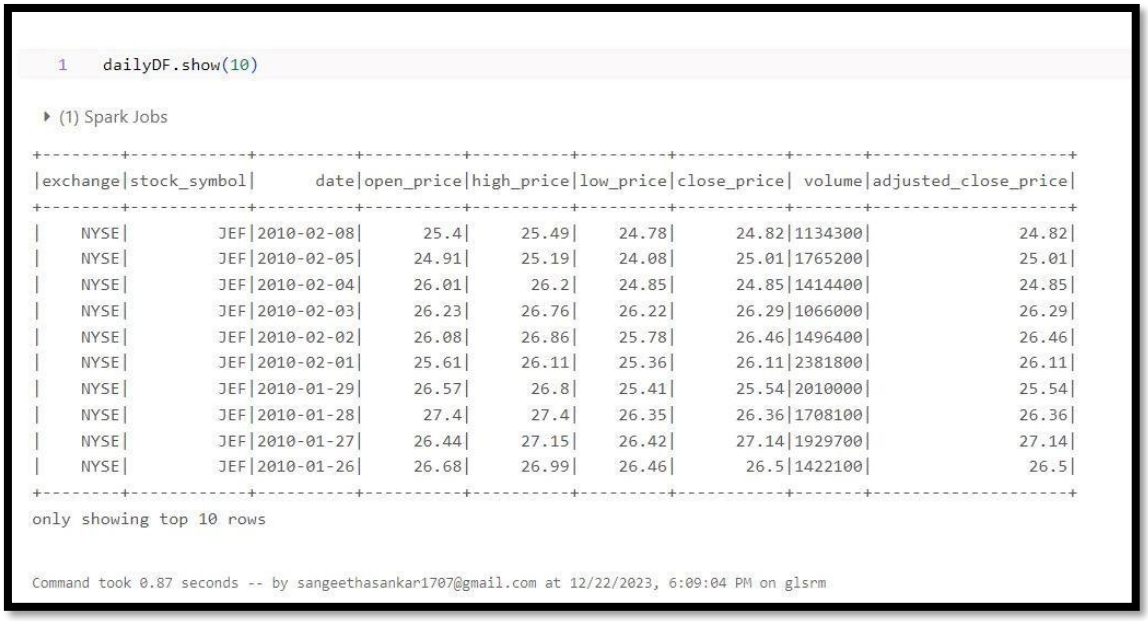
- Import SparkSession: The second line (from pyspark.sql import SparkSession) imports the SparkSession class, which is the entry point to programming Spark with the DataFrame and SQL API.

- Create SparkSession: The third line (spark = SparkSession.builder.appName("Python Spark SQL example").getOrCreate()) creates a SparkSession. The SparkSession is a unified entry point for reading data, configuring the runtime settings, and performing operations in Spark. The appName method sets a name for your application, and getOrCreate either retrieves an existing SparkSession or creates a new one if none exists
- We are using PySpark to read data from two text files (daily_data.txt and dividend_data.txt) and creating two DataFrames (dailyDF and dividendDF).
  spark.read.load: This method is used to read data from a data source. In this case, you're loading data from a CSV file.
- "/FileStore/tables/daily_data.txt": This is the path to the CSV file you want to read.
- format="csv": Specifies that the data source is in CSV format.
- sep="\t": Specifies the delimiter used in the file. In this case, it's a tab character (\t).
- inferSchema="true": Enables automatic inference of the schema (data types) of the DataFrame.
- header="true": Specifies that the first row of the file contains the header (column names).

- "Printschema" displays the schema of the DataFrame, showing the column names, data types, and whether each column allows null values

---

2. Get the number of rows and the number of columns of the dataframe.



```
1  dailyDF.show(10)
```

(1) Spark Jobs

exchange	stock_symbol	date	open_price	high_price	low_price	close_price	volume	adjusted_close_price
NYSE	JEF	2010-02-08	25.4	25.49	24.78	24.82	1134300	24.82
NYSE	JEF	2010-02-05	24.91	25.19	24.08	25.01	1765200	25.01
NYSE	JEF	2010-02-04	26.01	26.2	24.85	24.85	1414400	24.85
NYSE	JEF	2010-02-03	26.23	26.76	26.22	26.29	1066000	26.29
NYSE	JEF	2010-02-02	26.08	26.86	25.78	26.46	1496400	26.46
NYSE	JEF	2010-02-01	25.61	26.11	25.36	26.11	2381800	26.11
NYSE	JEF	2010-01-29	26.57	26.8	25.41	25.54	2010000	25.54
NYSE	JEF	2010-01-28	27.4	27.4	26.35	26.36	1708100	26.36
NYSE	JEF	2010-01-27	26.44	27.15	26.42	27.14	1929700	27.14
NYSE	JEF	2010-01-26	26.68	26.99	26.46	26.5	1422100	26.5

only showing top 10 rows

Command took 0.87 seconds -- by sangeethasankar1707@gmail.com at 12/22/2023, 6:09:04 PM on glsrm

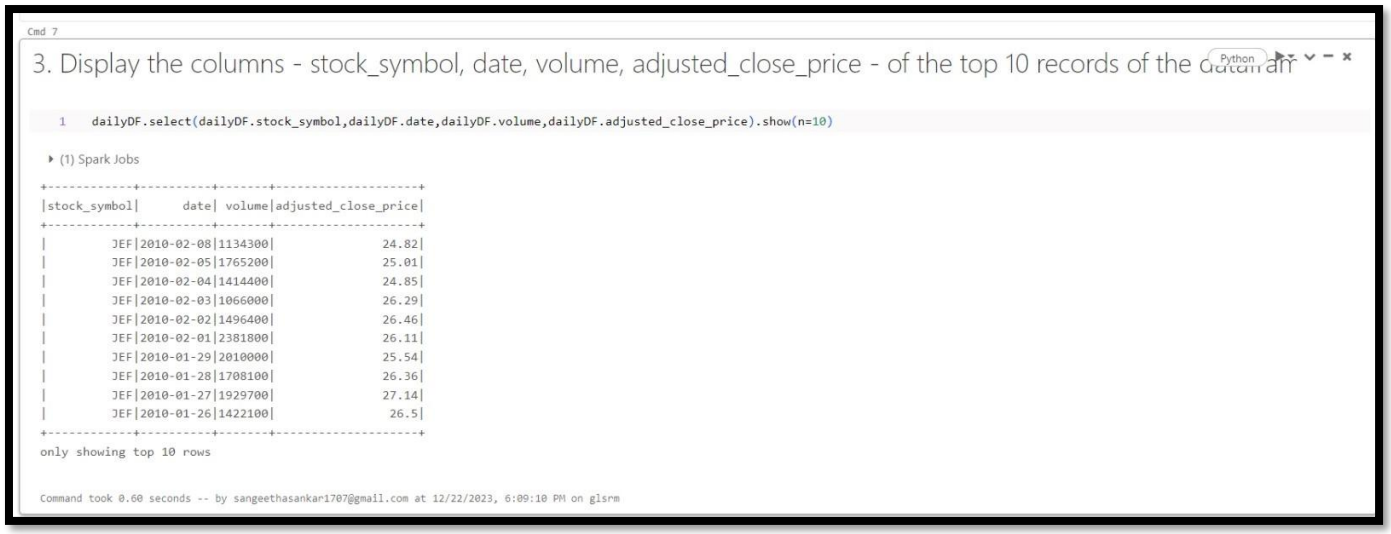
Inference:

- The count() method in PySpark is used to count the number of rows in a DataFrame.
- The len(dailyDF.columns) expression will give you the number of columns in the dailyDF DataFrame. This approach works because dailyDF.columns returns a list of column names, and len() gives you the length (number of elements) of that list.
- The show() method in PySpark is used to display the first few rows of a DataFrame.

- 171430 are the number of rows present and 9 columns are present in the dataset dailyDF.

---

3. Display the columns - stock_symbol, date, volume, adjusted_close_price - of the top 10 records of the dataframe. Use Spark SQL's dataframe function for this.



Inference:

- We are using the "select" method to choose specific columns from the dailyDF DataFrame and then using the show method to display the first 10 rows.
- Here the code selecting the columns 'stock_symbol', 'date', 'volume', and 'adjusted_close_price' from dailyDF dataframe.
- It will display the first 10 rows of the selected columns from the dailyDF DataFrame. Adjust the number in the show() method if you want to display a different number of rows.

---

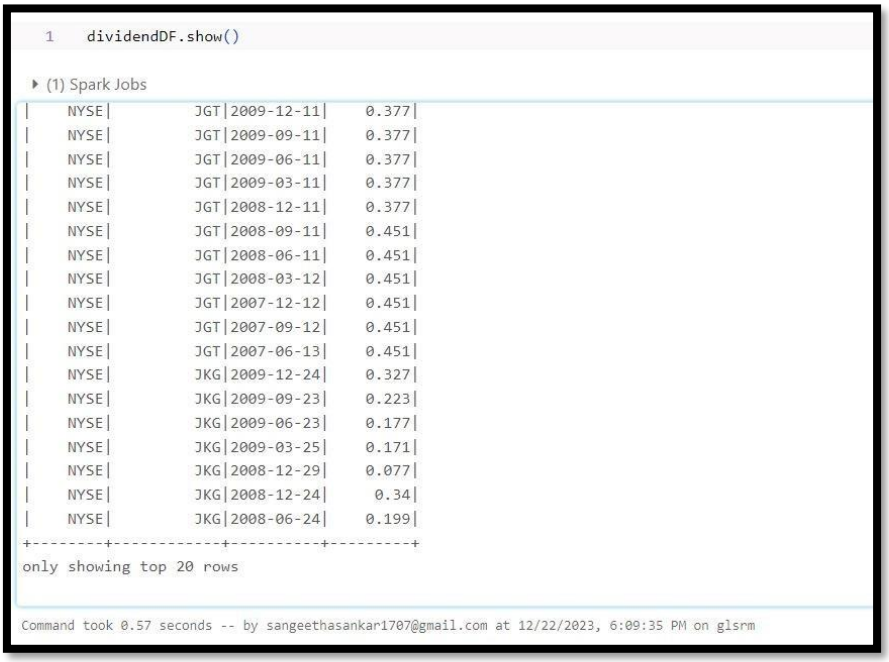
4. Display the columns - stock_symbol, date, volume, adjusted_close_price – for the top 10 rows of the dataframe after filtering rows of the dataframe for volume more than 2 million i.e. 20 lakhs.

Inference:

- we are using the "select" and "filter" methods to choose specific columns and filter rows based on a condition in the dailyDF DataFrame. In this case, we are selecting the columns 'stock_symbol', 'date', 'volume', and 'adjusted_close_price' and filtering rows where the 'volume' is greater than 2,000,000.
- The filter method in PySpark is used to filter rows from a DataFrame based on a given condition or set of conditions. It is often used in conjunction with the select method to perform both column selection and row filtering.

---

5. Create a dataframe from the dividend data file. Display the schema of the dataframe.



```
1  dividendDF.show()

▸ (1) Spark Jobs

|    NYSE|         JGT|2009-12-11|    0.377|
|    NYSE|         JGT|2009-09-11|    0.377|
|    NYSE|         JGT|2009-06-11|    0.377|
|    NYSE|         JGT|2009-03-11|    0.377|
|    NYSE|         JGT|2008-12-11|    0.377|
|    NYSE|         JGT|2008-09-11|    0.451|
|    NYSE|         JGT|2008-06-11|    0.451|
|    NYSE|         JGT|2008-03-12|    0.451|
|    NYSE|         JGT|2007-12-12|    0.451|
|    NYSE|         JGT|2007-09-12|    0.451|
|    NYSE|         JGT|2007-06-13|    0.451|
|    NYSE|         JKG|2009-12-24|    0.327|
|    NYSE|         JKG|2009-09-23|    0.223|
|    NYSE|         JKG|2009-06-23|    0.177|
|    NYSE|         JKG|2009-03-25|    0.171|
|    NYSE|         JKG|2008-12-29|    0.077|
|    NYSE|         JKG|2008-12-24|     0.34|
|    NYSE|         JKG|2008-06-24|    0.199|
+--------+------------+----------+---------+
only showing top 20 rows

Command took 0.57 seconds -- by sangeethasankar1707@gmail.com at 12/22/2023, 6:09:35 PM on glsrm
```

Inference:

- We are using PySpark to read data from two text files (daily_data.txt and dividend_data.txt) and creating two DataFrames (dailyDF and dividendDF). spark.read.load: This method is used to read data from a data source. In this case, you're loading data from a CSV file.
- "/FileStore/tables/daily_data.txt": This is the path to the CSV file you want to read.
- format="csv": Specifies that the data source is in CSV format.
- sep="\t": Specifies the delimiter used in the file. In this case, it's a tab character (\t).
- inferSchema="true": Enables automatic inference of the schema (data types) of the DataFrame.
- header="true": Specifies that the first row of the file contains the header (column names).
- "Printschema" displays the schema of the DataFrame, showing the column names, data types, and whether each column allows null values

---

6. Get the stock_symbol and the number of times it gave dividends in our data. Use Spark SQL's dataframe function for this.

---

6. Get the stock_symbol and the number of times it gave dividends in our data. Use Spark SQL's dataframe functi

```
1  dividendDF.createOrReplaceTempView("dividend_table")
2
3  result = spark.sql("SELECT stock_symbol, COUNT(*) AS dividend_count FROM dividend_table GROUP BY stock_symbol ORDER BY stock_symbol")
```

result: pyspark.sql.dataframe.DataFrame = [stock_symbol: string, dividend_count: long]

Command took 0.68 seconds -- by sangeethasankar1707@gmail.com at 12/22/2023, 6:09:44 PM on glsrm

Cmd 13

```
1  result.show()
```

(2) Spark Jobs

JBJ	13
JBK	20
JBL	15
JBN	5
JBO	11
JBR	5
JBT	5
JCE	11
JCI	97
JCP	114
JDD	50
JEF	72
JEQ	8
JFC	19
JFP	58
JFR	68
JGG	17
JGT	11

only showing top 20 rows

Command took 2.08 seconds -- by sangeethasankar1707@gmail.com at 12/22/2023, 6:09:54 PM on glsrm

Inference:

- we are creating a temporary view named "dividend_table" using the createOrReplaceTempView method and then executing a SQL query on that view using the spark.sql method. The SQL query is counting the number of dividends for each stock symbol in the "dividend_table" and ordering the results by stock symbol.
- we use SQL to perform operations on that view. The result will be a new DataFrame (result) with two columns: 'stock_symbol' and 'dividend_count'.
- show() command is used to display the content of the result DataFrame.

---

7. Get a list with stock symbol, date, volume, adjusted close price and dividends by joining the two dataframes on stock symbol and date columns.

---

7. Get a list with stock symbol, date, volume, adjusted close price and dividends by joining the two dataframes

```
joined_df=dailyDF.join(dividendDF, ['stock_symbol', 'date'], 'inner')
```

joined_df: pyspark.sql.dataframe.DataFrame = [stock_symbol: string, date: date ... 9 more fields]

Command took 0.21 seconds -- by sangeethasankar1707@gmail.com at 12/22/2023, 6:10:01 PM on glsrm

Cmd 15

```
joined_df1=joined_df.select('stock_symbol', 'date', 'volume', 'adjusted_close_price', 'dividends')
```

joined_df1: pyspark.sql.dataframe.DataFrame = [stock_symbol: string, date: date ... 3 more fields]

Command took 0.21 seconds -- by sangeethasankar1707@gmail.com at 12/22/2023, 6:10:05 PM on glsrm

```
joined_df1.show()
```

(2) Spark Jobs

JEF	2007-11-13	1488700	24.52	0.125
JEF	2007-08-13	1646400	24.33	0.125
JEF	2007-05-11	2019500	31.3	0.125
JEF	2007-02-13	519200	27.57	0.125
JEF	2006-11-13	467000	28.51	0.125
JEF	2006-08-11	385600	23.93	0.125
JEF	2006-05-23	1009300	28.4	0.063
JEF	2006-02-13	511200	25.25	0.075
JEF	2005-11-10	658200	21.65	0.075
JEF	2005-08-11	283800	19.48	0.06
JEF	2005-05-12	1393400	16.14	0.06
JEF	2005-02-11	824400	19.05	0.06
JEF	2004-11-10	561200	19.46	0.05
JEF	2004-08-12	337400	14.33	0.05
JEF	2004-05-12	1251600	15.31	0.04
JEF	2004-02-12	316200	17.57	0.04
JEF	2003-11-13	324200	15.09	0.04
JEF	2003-08-22	630400	13.96	0.04

only showing top 20 rows

Command took 1.86 seconds -- by sangeethasankar1707@gmail.com at 12/22/2023, 6:10:10 PM on glsrm

```
# Execute Spark SQL query with JOIN
result = dailyDF.join(dividendDF, (dailyDF["date"] == dividendDF["date"]) & (dailyDF["stock_symbol"] == dividendDF["stock_symbol"]), "inner") \
        .select(dailyDF["date"], dailyDF["stock_symbol"], dailyDF["volume"], dailyDF["adjusted_close_price"],dividendDF["dividends"])

# Show the result
result.show()
```

(2) Spark Jobs

result: pyspark.sql.dataframe.DataFrame = [date: date, stock_symbol: string ... 3 more fields]

2007-11-13	JEF	1488700	24.52	0.125
2007-08-13	JEF	1646400	24.33	0.125
2007-05-11	JEF	2019500	31.3	0.125
2007-02-13	JEF	519200	27.57	0.125
2006-11-13	JEF	467000	28.51	0.125
2006-08-11	JEF	385600	23.93	0.125
2006-05-23	JEF	1009300	28.4	0.063
2006-02-13	JEF	511200	25.25	0.075
2005-11-10	JEF	658200	21.65	0.075
2005-08-11	JEF	283800	19.48	0.06
2005-05-12	JEF	1393400	16.14	0.06
2005-02-11	JEF	824400	19.05	0.06
2004-11-10	JEF	561200	19.46	0.05
2004-08-12	JEF	337400	14.33	0.05
2004-05-12	JEF	1251600	15.31	0.04
2004-02-12	JEF	316200	17.57	0.04
2003-11-13	JEF	324200	15.09	0.04
2003-08-22	JEF	630400	13.96	0.04

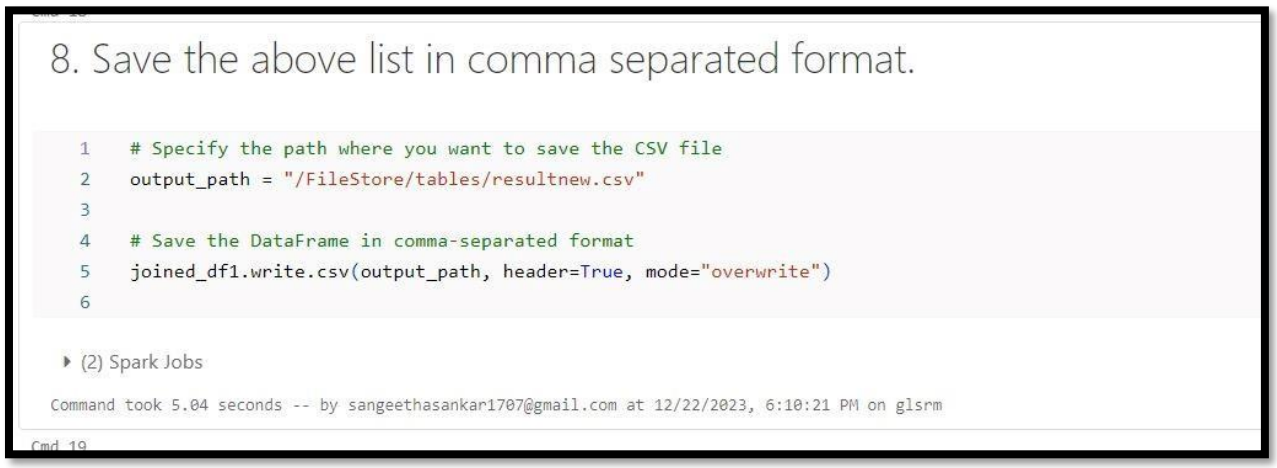
only showing top 20 rows

Inference:

- we are performing an inner join between the dailyDF and dividendDF DataFrames based on the columns 'stock_symbol' and 'date'. The resulting DataFrame, joined_df, will contain columns from both DataFrames for the matched rows.
- Here's a breakdown of your code:
- dailyDF: The left DataFrame in the join.
- dividendDF: The right DataFrame in the join.
- ['stock_symbol', 'date']: The columns used for the join condition. This specifies that the join should be based on matching values in the 'stock_symbol' and 'date' columns.
- 'inner': The type of join. In this case, it's an inner join, which means only the rows with matching values in both DataFrames will be included in the result.
- The resulting joined_df DataFrame will contain columns from both dailyDF and dividendDF for the rows where the specified columns have matching values.
- we are selecting specific columns from the joined_df DataFrame and creating a new DataFrame named joined_df1
- It selects the columns 'stock_symbol', 'date', 'volume', 'adjusted_close_price', and 'dividends' from the joined_df DataFrame and creates a new DataFrame joined_df1 with only these columns.
- show() is used to display the content of the joined_df1 DataFrame

---

8. Save the above list in comma separated format.



```
%fs ls dbfs:/FileStore/tables
```

Table +

	path	name	size	modificationTime
1	dbfs:/FileStore/tables/daily_data.txt	daily_data.txt	9105715	1703248630000
2	dbfs:/FileStore/tables/dividend_data.txt	dividend_data.txt	55598	1703248627000
3	dbfs:/FileStore/tables/resultnew.csv/	resultnew.csv/	0	0
4	dbfs:/FileStore/tables/war_and_peace.txt	war_and_peace.txt	3266939	1700300656000

4 rows | 13.86 seconds runtime — Refreshed 8 minutes ago

Command took 13.86 seconds -- by sangeethasankar1707@gmail.com at 12/22/2023, 6:10:29 PM on glsrm

Cmd 20

```
resultnewDF = spark.read.load("/FileStore/tables/resultnew.csv", format="csv", sep="\t", inferSchema="true",header="true")
```

▸ (2) Spark Jobs

▸ resultnewDF: pyspark.sql.dataframe.DataFrame = [stock_symbol,date,volume,adjusted_close_price,dividends: string]

Command took 1.72 seconds -- by sangeethasankar1707@gmail.com at 12/22/2023, 6:10:51 PM on glsrm

```
resultnewDF.show()
```

▸ (1) Spark Jobs

```
|                                   JEF,2007-11-13,14...|
|                                   JEF,2007-08-13,16...|
|                                   JEF,2007-05-11,20...|
|                                   JEF,2007-02-13,51...|
|                                   JEF,2006-11-13,46...|
|                                   JEF,2006-08-11,38...|
|                                   JEF,2006-05-23,10...|
|                                   JEF,2006-02-13,51...|
|                                   JEF,2005-11-10,65...|
|                                   JEF,2005-08-11,28...|
|                                   JEF,2005-05-12,13...|
|                                   JEF,2005-02-11,82...|
|                                   JEF,2004-11-10,56...|
|                                   JEF,2004-08-12,33...|
|                                   JEF,2004-05-12,12...|
|                                   JEF,2004-02-12,31...|
|                                   JEF,2003-11-13,32...|
|                                   JEF,2003-08-22,63...|
+-----------------------------------------------------------+
only showing top 20 rows
```

Command took 0.32 seconds -- by sangeethasankar1707@gmail.com at 12/22/2023, 6:10:58 PM on glsrm

Cmd 22

```
spark.stop()
```

The spark context has stopped and the driver is restarting. Your notebook will be automatically reattached.

Command took 2.90 seconds -- by sangeethasankar1707@gmail.com at 12/22/2023, 6:20:35 PM on glsrm

Inference:

- we are saving the joined_df1 DataFrame as a CSV file in the specified output path ("/FileStore/tables/resultnew.csv").
- Here's the breakdown of code:

- the write.csv method to save the DataFrame to a CSV file. The parameters used are:

- output_path: The path where the CSV file will be saved.
- header=True: Specifies that the CSV file should include a header with column names.
- mode="overwrite": If a file with the same name already exists, it will be overwritten.
- After running this code, we should find the CSV file with the specified columns and data in the specified output path.
- to list the contents of the directory located at dbfs:/FileStore/tables using Databricks %fs magic command.
- We are reading the resultnew csv file and display the content using show() method.
- The spark.stop() command is used to stop the SparkSession and release associated resources. When you call spark.stop(), it terminates the Spark application and releases any acquired resources.