databricks 1 - Introduction To PySpark

(https://databricks.com)

# Introduction To PySpark

PySpark is a Python library and framework for distributed data processing and analysis, specifically designed to work with big data. It is built on top of Apache Spark, which is an open-source, general-purpose cluster-computing framework. PySpark provides an interface for programming Spark with Python, allowing developers to leverage the power of Spark's distributed computing capabilities using Python's familiar syntax.

# Choose Data Source To Work On

Databricks includes built-in sample datasets such as the "databricks-datasets" library, which provides access to popular datasets like the "Iris" dataset, "California housing" dataset, "NYC Taxi Trips" dataset, and more. These datasets are readily available within the Databricks environment, allowing you to load them directly into Spark DataFrames or RDDs and perform data processing tasks.

**%fs** : In Databricks, **%fs** is a Databricks-specific command that allows you to interact with the file system (DBFS - Databricks File System) within the Databricks environment. It provides a convenient way to perform file system operations such as listing files, creating directories, uploading files, and more, directly from Databricks notebooks or the command-line interface.

We will use this command to perform file operations. We used ls command to list all directories.

```
%fs

ls
```

Table

| | path | name | size | modificationTime |
|---|---|---|---|---|
| 1 | dbfs:/databricks-datasets/ | databricks-datasets/ | 0 | 0 |
| 2 | dbfs:/databricks-results/ | databricks-results/ | 0 | 0 |

2 rows

```
%fs

ls dbfs:/databricks-datasets/
```

Table

| | path | name | size | modificationTim |
|---|---|---|---|---|
| 1 | dbfs:/databricks-datasets/COVID/ | COVID/ | 0 | 0 |
| 2 | dbfs:/databricks-datasets/README.md | README.md | 976 | 1532468253000 |
| 3 | dbfs:/databricks-datasets/Rdatasets/ | Rdatasets/ | 0 | 0 |
| 4 | dbfs:/databricks-datasets/SPARK_README.md | SPARK_README.md | 3359 | 1455043490000 |
| 5 | dbfs:/databricks-datasets/adult/ | adult/ | 0 | 0 |
| 6 | dbfs:/databricks-datasets/airlines/ | airlines/ | 0 | 0 |
| 7 | dbfs:/databricks-datasets/amazon/ | amazon/ | 0 | 0 |

55 rows

Now we can choose any dataset from these. Let's go with OnlineRetail Dataset.

```
%fs

ls dbfs:/databricks-datasets/online_retail/
```

Table

| | path | name | size | modificationTime |
|---|---|---|---|---|
| 1 | dbfs:/databricks-datasets/online_retail/data-001/ | data-001/ | 0 | 0 |

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | dbfs:/databricks-datasets/online_retail/data-001/ | data-001/ | 0 | 0 |

1 row

```
%fs

ls dbfs:/databricks-datasets/online_retail/data-001/data.csv
```

Table

|   | path | name | size | modificationTime |
|---|---|---|---|---|
| 1 | dbfs:/databricks-datasets/online_retail/data-001/data.csv | data.csv | 5357240 | 1466107812000 |

1 row

# How To Read CSV Files

## Pyspark SQL :

PySpark SQL is a module within PySpark that provides a programming interface for working with structured and tabular data using SQL queries, as well as DataFrame and Dataset APIs. It allows you to perform SQL-like operations on distributed datasets.With PySpark SQL, you can manipulate structured data using SQL statements, which makes it easier for those familiar with SQL to interact with big data.

## SparkSession :

In Apache Spark, a SparkSession is the entry point for programming with Spark and provides a unified interface for interacting with Spark functionalities. It is the primary way of interacting with various Spark APIs, including DataFrame, Dataset, SQL.

```
from pyspark.sql import SparkSession

# Create a SparkSession
spark = SparkSession.builder.appName("ReadCSV").getOrCreate()

# Read the CSV file into a DataFrame
df = spark.read.csv("dbfs:/databricks-datasets/online_retail/data-001/data.csv", header=True, inferSchema=True)

# Show the first few rows of the DataFrame
df.show()
```

```
+---------+---------+--------------------+--------+------------+---------+----------+--------------+
|InvoiceNo|StockCode|         Description|Quantity| InvoiceDate|UnitPrice|CustomerID|       Country|
+---------+---------+--------------------+--------+------------+---------+----------+--------------+
|   536365|   85123A|WHITE HANGING HEA...|       6|12/1/10 8:26|     2.55|     17850|United Kingdom|
|   536365|    71053| WHITE METAL LANTERN|       6|12/1/10 8:26|     3.39|     17850|United Kingdom|
|   536365|   84406B|CREAM CUPID HEART...|       8|12/1/10 8:26|     2.75|     17850|United Kingdom|
|   536365|   84029G|KNITTED UNION FLA...|       6|12/1/10 8:26|     3.39|     17850|United Kingdom|
|   536365|   84029E|RED WOOLLY HOTTIE...|       6|12/1/10 8:26|     3.39|     17850|United Kingdom|
|   536365|    22752|SET 7 BABUSHKA NE...|       2|12/1/10 8:26|     7.65|     17850|United Kingdom|
|   536365|    21730|GLASS STAR FROSTE...|       6|12/1/10 8:26|     4.25|     17850|United Kingdom|
|   536366|    22633|HAND WARMER UNION...|       6|12/1/10 8:28|     1.85|     17850|United Kingdom|
|   536366|    22632|HAND WARMER RED P...|       6|12/1/10 8:28|     1.85|     17850|United Kingdom|
|   536367|    84879|ASSORTED COLOUR B...|      32|12/1/10 8:34|     1.69|     13047|United Kingdom|
|   536367|    22745|POPPY'S PLAYHOUSE...|       6|12/1/10 8:34|      2.1|     13047|United Kingdom|
|   536367|    22748|POPPY'S PLAYHOUSE...|       6|12/1/10 8:34|      2.1|     13047|United Kingdom|
|   536367|    22749|FELTCRAFT PRINCES...|       8|12/1/10 8:34|     3.75|     13047|United Kingdom|
|   536367|    22310|IVORY KNITTED MUG...|       6|12/1/10 8:34|     1.65|     13047|United Kingdom|
|   536367|    84969|BOX OF 6 ASSORTED...|       6|12/1/10 8:34|     4.25|     13047|United Kingdom|
|   536367|    22623|BOX OF VINTAGE JI...|       3|12/1/10 8:34|     4.95|     13047|United Kingdom|
|   536367|    22622|BOX OF VINTAGE AL...|       2|12/1/10 8:34|     9.95|     13047|United Kingdom|
|   536367|    21754|HOME BUILDING BLO...|       3|12/1/10 8:34|     5.95|     13047|United Kingdom|
```

In the code above, the SparkSession is created using the SparkSession.builder method. You can specify the application name with appName. Then, you can use the read.csv() method of the SparkSession object to read the CSV file.

The read.csv() method takes several parameters:

1.The first parameter is the path to the CSV file.
2.The header parameter is set to True to indicate that the first row of the CSV file contains column headers.
3.The inferSchema parameter is set to True to automatically infer the data types of the columns.

After reading the CSV file, the data is loaded into a DataFrame object (df in the example). You can perform various operations on the DataFrame, such as filtering, aggregating, or transforming the data.

Finally, you can use the show() method to display the first few rows of the DataFrame.

# How To Import Other File Formats

You can also read and analyze data in all formats, be it SQL Tables, JSON, HTML, or Parquet. All we need to use are following methods i.e.

## format() :

we use the format() method to specify the file format. It may be csv , xml , json etc.

## option :

the option() method is used to set configuration options when reading data or performing operations on DataFrames. It allows you to specify various parameters and options to customize the behavior of the operations.

## load :

The load() method allows you to read data from various file formats and data sources such as CSV, JSON, Parquet, Avro, JDBC, and more.

```
from pyspark.sql import SparkSession

# Create a SparkSession
spark = SparkSession.builder.appName("ReadCSV").getOrCreate()

# Read the CSV file into a DataFrame
df = spark.read.format('csv').option('header','True').option('inferSchema','True').load("dbfs:/databricks-
datasets/online_retail/data-001/data.csv")

# Show the first few rows of the DataFrame
df.show()
```

```
+---------+---------+--------------------+--------+------------+---------+----------+--------------+
|InvoiceNo|StockCode|         Description|Quantity| InvoiceDate|UnitPrice|CustomerID|       Country|
+---------+---------+--------------------+--------+------------+---------+----------+--------------+
|   536365|   85123A|WHITE HANGING HEA...|       6|12/1/10 8:26|     2.55|     17850|United Kingdom|
|   536365|    71053| WHITE METAL LANTERN|       6|12/1/10 8:26|     3.39|     17850|United Kingdom|
|   536365|   84406B|CREAM CUPID HEART...|       8|12/1/10 8:26|     2.75|     17850|United Kingdom|
|   536365|   84029G|KNITTED UNION FLA...|       6|12/1/10 8:26|     3.39|     17850|United Kingdom|
|   536365|   84029E|RED WOOLLY HOTTIE...|       6|12/1/10 8:26|     3.39|     17850|United Kingdom|
|   536365|    22752|SET 7 BABUSHKA NE...|       2|12/1/10 8:26|     7.65|     17850|United Kingdom|
|   536365|    21730|GLASS STAR FROSTE...|       6|12/1/10 8:26|     4.25|     17850|United Kingdom|
|   536366|    22633|HAND WARMER UNION...|       6|12/1/10 8:28|     1.85|     17850|United Kingdom|
|   536366|    22632|HAND WARMER RED P...|       6|12/1/10 8:28|     1.85|     17850|United Kingdom|
|   536367|    84879|ASSORTED COLOUR B...|      32|12/1/10 8:34|     1.69|     13047|United Kingdom|
|   536367|    22745|POPPY'S PLAYHOUSE...|       6|12/1/10 8:34|      2.1|     13047|United Kingdom|
|   536367|    22748|POPPY'S PLAYHOUSE...|       6|12/1/10 8:34|      2.1|     13047|United Kingdom|
|   536367|    22749|FELTCRAFT PRINCES...|       8|12/1/10 8:34|     3.75|     13047|United Kingdom|
|   536367|    22310|IVORY KNITTED MUG...|       6|12/1/10 8:34|     1.65|     13047|United Kingdom|
|   536367|    84969|BOX OF 6 ASSORTED...|       6|12/1/10 8:34|     4.25|     13047|United Kingdom|
|   536367|    22623|BOX OF VINTAGE JI...|       3|12/1/10 8:34|     4.95|     13047|United Kingdom|
|   536367|    22622|BOX OF VINTAGE AL...|       2|12/1/10 8:34|     9.95|     13047|United Kingdom|
|   536367|    21754|HOME BUILDING BLO...|       3|12/1/10 8:34|     5.95|     13047|United Kingdom|
```

# How To Read Headers Or Metadata

To read headers , we use same function **columns** just like in pandas. It returns a **list of columns**.

```
df.columns
```

```
Out[93]: ['InvoiceNo',
 'StockCode',
 'Description',
 'Quantity',
 'InvoiceDate',
 'UnitPrice',
 'CustomerID',
 'Country']
```

# Read Top Few Rows and Bottom Few Rows :

We will use head() and tail() functions just like we did in pandas. It returns a list of Rows.

```
df.head(5)
```

```
Out[94]: [Row(InvoiceNo='536365', StockCode='85123A', Description='WHITE HANGING HEART T-LIGHT HOLDER', Quantity=6, I
nvoiceDate='12/1/10 8:26', UnitPrice=2.55, CustomerID=17850, Country='United Kingdom'),
 Row(InvoiceNo='536365', StockCode='71053', Description='WHITE METAL LANTERN', Quantity=6, InvoiceDate='12/1/10 8:2
6', UnitPrice=3.39, CustomerID=17850, Country='United Kingdom'),
 Row(InvoiceNo='536365', StockCode='84406B', Description='CREAM CUPID HEARTS COAT HANGER', Quantity=8, InvoiceDate='1
2/1/10 8:26', UnitPrice=2.75, CustomerID=17850, Country='United Kingdom'),
 Row(InvoiceNo='536365', StockCode='84029G', Description='KNITTED UNION FLAG HOT WATER BOTTLE', Quantity=6, InvoiceDa
te='12/1/10 8:26', UnitPrice=3.39, CustomerID=17850, Country='United Kingdom'),
 Row(InvoiceNo='536365', StockCode='84029E', Description='RED WOOLLY HOTTIE WHITE HEART.', Quantity=6, InvoiceDate='1
2/1/10 8:26', UnitPrice=3.39, CustomerID=17850, Country='United Kingdom')]
```

```
df.tail(5)
```

```
Out[95]: [Row(InvoiceNo='541695', StockCode='85095', Description='THREE CANVAS LUGGAGE TAGS', Quantity=2, InvoiceDate
='1/20/11 18:01', UnitPrice=1.25, CustomerID=None, Country='United Kingdom'),
 Row(InvoiceNo='541695', StockCode='85099B', Description='JUMBO BAG RED RETROSPOT', Quantity=1, InvoiceDate='1/20/11
18:01', UnitPrice=4.13, CustomerID=None, Country='United Kingdom'),
 Row(InvoiceNo='541695', StockCode='85103', Description='SILVER T-LIGHT SETTING', Quantity=10, InvoiceDate='1/20/11 1
8:01', UnitPrice=5.79, CustomerID=None, Country='United Kingdom'),
 Row(InvoiceNo='541695', StockCode='85106', Description='CUT GLASS HEXAGON T-LIGHT HOLDER', Quantity=1, InvoiceDate
='1/20/11 18:01', UnitPrice=1.63, CustomerID=None, Country='United Kingdom'),
 Row(InvoiceNo='541695', StockCode='85125', Description='SMALL ROUND CUT GLASS CANDLESTICK', Quantity=2, InvoiceDate
='1/20/11 18:01', UnitPrice=3.29, CustomerID=None, Country='United Kingdom')]
```

# How To Read Data Separated By Any Delimiter

To read data separated by a custom delimiter other than the default comma (,) in PySpark, you can use the option() method of the DataFrameReader class and specify the delimiter using the delimiter parameter.

```
# from pyspark.sql import SparkSession

# # Create a SparkSession
# spark = SparkSession.builder.appName("ReadCSV").getOrCreate()

# # Read the CSV file into a DataFrame
# df =
spark.read.format('csv').option('delimiter','|').option('header','True').option('inferSchema','True').load("dbfs:/data
bricks-datasets/online_retail/data-001/data.csv")

# # Show the first few rows of the DataFrame
# df.show()
```

# Describe Statistical Summary

We will use describe function to get all statistical summary all columns

```
ndf = df.describe()

ndf.show()
```

```
+-------+------------------+------------------+--------------------+------------------+-------------+----------------
--+------------------+--------------+
|summary|         InvoiceNo|         StockCode|         Description|          Quantity|  InvoiceDate|         UnitPri
ce|        CustomerID|       Country|
+-------+------------------+------------------+--------------------+------------------+-------------+----------------
--+------------------+--------------+
|  count|             65499|             65499|               65333|             65499|        65499|             654
99|             40218|         65499|
|   mean| 539091.6921058759|29165.654135469875|                null| 8.366234599001512|         null| 5.8575856119946
96|15384.033517330548|          null|
| stddev|1586.8350514333126|19298.622465903674|                null|413.80812814338367|         null|145.795962655818
22|1766.8634991790627|          null|
|    min|            536365|             10002| 4 PURPLE FLOCK D...|            -74215|1/10/11 10:04|
0.0|             12346|     Australia|
|    max|           C541694|                 m|reverse 21/5/10 a...|             74215| 12/9/10 9:49|          16888.
02|             18283|United Kingdom|
+-------+------------------+------------------+--------------------+------------------+-------------+----------------
--+------------------+--------------+
```

# How To Access A Single Column

To access a Single Column,we can use select function.

```
from pyspark.sql.functions import col

inv = df.select('InvoiceNo')
inv.show()
```

```
+---------+
|InvoiceNo|
+---------+
|   536365|
|   536365|
|   536365|
|   536365|
|   536365|
|   536365|
|   536365|
|   536366|
|   536366|
|   536367|
|   536367|
|   536367|
|   536367|
|   536367|
|   536367|
|   536367|
|   536367|
|   536367|
```

# How To Access More than One Columns

It is pretty simple to access more than one column , just write down all the columns inside select function.

```
df.select('InvoiceNo','CustomerId','UnitPrice').display()
```

Table

| | InvoiceNo | CustomerId | UnitPrice |
|---|---|---|---|
| 1 | 536365 | 17850 | 2.55 |
| 2 | 536365 | 17850 | 3.39 |
| 3 | 536365 | 17850 | 2.75 |
| 4 | 536365 | 17850 | 3.39 |
| 5 | 536365 | 17850 | 3.39 |

| | | | |
|---|---|---|---|
| 5 | 536365 | 17850 | 3.39 |
| 6 | 536365 | 17850 | 7.65 |
| 7 | 536365 | 17850 | 4.25 |

10,000 rows | Truncated data

# How To Filter Data based on Condition

There are two Alternatives to filter data based on condition. We can use **filter** and **where** functions.

## Products Where Unit Price Is More Than 100 Dollars

```
filtered_data = df.filter(col("UnitPrice")>5)
filtered_data.show()
```

```
+---------+---------+--------------------+--------+------------+---------+----------+--------------+
|InvoiceNo|StockCode|         Description|Quantity| InvoiceDate|UnitPrice|CustomerID|       Country|
+---------+---------+--------------------+--------+------------+---------+----------+--------------+
|   536365|    22752|SET 7 BABUSHKA NE...|       2|12/1/10 8:26|     7.65|     17850|United Kingdom|
|   536367|    22622|BOX OF VINTAGE AL...|       2|12/1/10 8:34|     9.95|     13047|United Kingdom|
|   536367|    21754|HOME BUILDING BLO...|       3|12/1/10 8:34|     5.95|     13047|United Kingdom|
|   536367|    21755|LOVE BUILDING BLO...|       3|12/1/10 8:34|     5.95|     13047|United Kingdom|
|   536367|    21777|RECIPE BOX WITH M...|       4|12/1/10 8:34|     7.95|     13047|United Kingdom|
|   536367|    48187| DOORMAT NEW ENGLAND|       4|12/1/10 8:34|     7.95|     13047|United Kingdom|
|   536369|    21756|BATH BUILDING BLO...|       3|12/1/10 8:35|     5.95|     13047|United Kingdom|
|   536370|     POST|             POSTAGE|       3|12/1/10 8:45|     18.0|     12583|        France|
|   536373|    82486|WOOD S/3 CABINET ...|       4|12/1/10 9:02|     6.95|     17850|United Kingdom|
|   536373|    22752|SET 7 BABUSHKA NE...|       2|12/1/10 9:02|     7.65|     17850|United Kingdom|
|   536374|    21258|VICTORIAN SEWING ...|      32|12/1/10 9:09|    10.95|     15100|United Kingdom|
|   536375|    82486|WOOD S/3 CABINET ...|       4|12/1/10 9:32|     6.95|     17850|United Kingdom|
|   536375|    22752|SET 7 BABUSHKA NE...|       2|12/1/10 9:32|     7.65|     17850|United Kingdom|
|   536381|  15056BL|EDWARDIAN PARASOL...|       2|12/1/10 9:41|     5.95|     15311|United Kingdom|
|   536381|   15056N|EDWARDIAN PARASOL...|       2|12/1/10 9:41|     5.95|     15311|United Kingdom|
|   536381|    21523|DOORMAT FANCY FON...|      10|12/1/10 9:41|     6.75|     15311|United Kingdom|
|  C536379|        D|            Discount|      -1|12/1/10 9:41|     27.5|     14527|United Kingdom|
|   536382|    22926|IVORY GIANT GARDE...|      12|12/1/10 9:45|     5.95|     16098|United Kingdom|
```

You can also use **where** method to do the same.

```
filtered_data = df.where(col("UnitPrice")>5)
filtered_data.show()
```

```
+---------+---------+--------------------+--------+------------+---------+----------+--------------+
|InvoiceNo|StockCode|         Description|Quantity| InvoiceDate|UnitPrice|CustomerID|       Country|
+---------+---------+--------------------+--------+------------+---------+----------+--------------+
|   536365|    22752|SET 7 BABUSHKA NE...|       2|12/1/10 8:26|     7.65|     17850|United Kingdom|
|   536367|    22622|BOX OF VINTAGE AL...|       2|12/1/10 8:34|     9.95|     13047|United Kingdom|
|   536367|    21754|HOME BUILDING BLO...|       3|12/1/10 8:34|     5.95|     13047|United Kingdom|
|   536367|    21755|LOVE BUILDING BLO...|       3|12/1/10 8:34|     5.95|     13047|United Kingdom|
|   536367|    21777|RECIPE BOX WITH M...|       4|12/1/10 8:34|     7.95|     13047|United Kingdom|
|   536367|    48187| DOORMAT NEW ENGLAND|       4|12/1/10 8:34|     7.95|     13047|United Kingdom|
|   536369|    21756|BATH BUILDING BLO...|       3|12/1/10 8:35|     5.95|     13047|United Kingdom|
|   536370|     POST|             POSTAGE|       3|12/1/10 8:45|     18.0|     12583|        France|
|   536373|    82486|WOOD S/3 CABINET ...|       4|12/1/10 9:02|     6.95|     17850|United Kingdom|
|   536373|    22752|SET 7 BABUSHKA NE...|       2|12/1/10 9:02|     7.65|     17850|United Kingdom|
|   536374|    21258|VICTORIAN SEWING ...|      32|12/1/10 9:09|    10.95|     15100|United Kingdom|
|   536375|    82486|WOOD S/3 CABINET ...|       4|12/1/10 9:32|     6.95|     17850|United Kingdom|
|   536375|    22752|SET 7 BABUSHKA NE...|       2|12/1/10 9:32|     7.65|     17850|United Kingdom|
|   536381|  15056BL|EDWARDIAN PARASOL...|       2|12/1/10 9:41|     5.95|     15311|United Kingdom|
|   536381|   15056N|EDWARDIAN PARASOL...|       2|12/1/10 9:41|     5.95|     15311|United Kingdom|
|   536381|    21523|DOORMAT FANCY FON...|      10|12/1/10 9:41|     6.75|     15311|United Kingdom|
|  C536379|        D|            Discount|      -1|12/1/10 9:41|     27.5|     14527|United Kingdom|
|   536382|    22926|IVORY GIANT GARDE...|      12|12/1/10 9:45|     5.95|     16098|United Kingdom|
```

# isin() :

We can use isin() function just like we did in pandas to apply filter on multiple values at once.

# Get Data Of India and France

```
filtered_data = df.filter(col('Country').isin(['UnitedKingdom','France']))
filtered_data.display()
```

Table

| | InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | Cust |
|---|---|---|---|---|---|---|---|
| 1 | 536370 | 22728 | ALARM CLOCK BAKELIKE PINK | 24 | 12/1/10 8:45 | 3.75 | 1258 |
| 2 | 536370 | 22727 | ALARM CLOCK BAKELIKE RED | 24 | 12/1/10 8:45 | 3.75 | 1258 |
| 3 | 536370 | 22726 | ALARM CLOCK BAKELIKE GREEN | 12 | 12/1/10 8:45 | 3.75 | 1258 |
| 4 | 536370 | 21724 | PANDA AND BUNNIES STICKER SHEET | 12 | 12/1/10 8:45 | 0.85 | 1258 |
| 5 | 536370 | 21883 | STARS GIFT TAPE | 24 | 12/1/10 8:45 | 0.65 | 1258 |
| 6 | 536370 | 10002 | INFLATABLE POLITICAL GLOBE | 48 | 12/1/10 8:45 | 0.85 | 1258 |
| 7 | 536370 | 21791 | VINTAGE HEADS AND TAILS CARD GAME | 24 | 12/1/10 8:45 | 1.25 | 1258 |

967 rows

# Sorting In Spark

In PySpark, you can use the sort or orderBy methods to sort a DataFrame based on one or more columns. Here's an example of how to sort data in PySpark:

## Sort By Country In Ascending Order

```
ndf = df.sort(col('Country'))
ndf.show()
```

```
+---------+---------+--------------------+--------+-------------+---------+----------+---------+
|InvoiceNo|StockCode|         Description|Quantity|  InvoiceDate|UnitPrice|CustomerID|  Country|
+---------+---------+--------------------+--------+-------------+---------+----------+---------+
|   541149|    22449|SILK PURSE BABUSH...|       6|1/14/11 11:36|     2.95|     12393|Australia|
|   536389|    22191|IVORY DINER WALL ...|       2|12/1/10 10:03|      8.5|     12431|Australia|
|   541271|    22722|SET OF 6 SPICE TI...|      12|1/17/11 11:12|     3.95|     12388|Australia|
|   537676|    22557|PLASTERS IN TIN V...|      12| 12/8/10 9:53|     1.65|     12386|Australia|
|   541149|    22450|SILK PURSE BABUSH...|       6|1/14/11 11:36|     2.95|     12393|Australia|
|   536389|    22195|LARGE HEART MEASU...|      24|12/1/10 10:03|     1.65|     12431|Australia|
|   540700|    21581|SKULLS  DESIGN  C...|       6| 1/11/11 9:47|     2.25|     12393|Australia|
|   536389|    22941|CHRISTMAS LIGHTS ...|       6|12/1/10 10:03|      8.5|     12431|Australia|
|   541149|    22451|SILK PURSE BABUSH...|      48|1/14/11 11:36|     2.95|     12393|Australia|
|   536389|    22196|SMALL HEART MEASU...|      24|12/1/10 10:03|     0.85|     12431|Australia|
|   540700|   84997B|RED 3 PIECE RETRO...|       6| 1/11/11 9:47|     3.75|     12393|Australia|
|   536389|   35004C|SET OF 3 COLOURED...|       6|12/1/10 10:03|     5.45|     12431|Australia|
|   541271|   84970L|SINGLE HEART ZINC...|      12|1/17/11 11:12|     0.95|     12388|Australia|
|   537676|    22567|20 DOLLY PEGS RET...|      24| 12/8/10 9:53|     1.25|     12386|Australia|
|   540700|    20726|  LUNCH BAG WOODLAND|      20| 1/11/11 9:47|     1.65|     12393|Australia|
|   536389|   85014B|RED RETROSPOT UMB...|       6|12/1/10 10:03|     5.95|     12431|Australia|
|   541271|    71459|HANGING JAM JAR T...|      24|1/17/11 11:12|     0.85|     12388|Australia|
|   537676|    22915|ASSORTED BOTTLE T...|     120| 12/8/10 9:53|     0.36|     12386|Australia|
```

## Sort By Country In Descending Order

```
ndf = df.sort('Country',ascending=False)
ndf.show()
```

```
+---------+---------+--------------------+--------+-------------+---------+----------+--------------+
|InvoiceNo|StockCode|         Description|Quantity|  InvoiceDate|UnitPrice|CustomerID|       Country|
+---------+---------+--------------------+--------+-------------+---------+----------+--------------+
|   540644|    20728| LUNCH BAG CARS BLUE|      10|1/10/11 14:16|     1.65|     16303|United Kingdom|
|   536365|   85123A|WHITE HANGING HEA...|       6| 12/1/10 8:26|     2.55|     17850|United Kingdom|
|   540644|    20971|PINK BLUE FELT CR...|      12|1/10/11 14:16|     1.25|     16303|United Kingdom|
|   536365|    71053| WHITE METAL LANTERN|       6| 12/1/10 8:26|     3.39|     17850|United Kingdom|
|   540644|    20972|PINK CREAM FELT C...|      12|1/10/11 14:16|     1.25|     16303|United Kingdom|
|   536365|   84406B|CREAM CUPID HEART...|       8| 12/1/10 8:26|     2.75|     17850|United Kingdom|
|   540644|    22570|FELTCRAFT CUSHION...|       4|1/10/11 14:16|     3.75|     16303|United Kingdom|
|   536365|   84029G|KNITTED UNION FLA...|       6| 12/1/10 8:26|     3.39|     17850|United Kingdom|
|   540644|    22569|FELTCRAFT CUSHION...|       4|1/10/11 14:16|     3.75|     16303|United Kingdom|
|   536365|   84029E|RED WOOLLY HOTTIE...|       6| 12/1/10 8:26|     3.39|     17850|United Kingdom|
|   540644|    22568|FELTCRAFT CUSHION...|       4|1/10/11 14:16|     3.75|     16303|United Kingdom|
|   536365|    22752|SET 7 BABUSHKA NE...|       2| 12/1/10 8:26|     7.65|     17850|United Kingdom|
|   540644|    22749|FELTCRAFT PRINCES...|       8|1/10/11 14:16|     3.75|     16303|United Kingdom|
|   536365|    21730|GLASS STAR FROSTE...|       6| 12/1/10 8:26|     4.25|     17850|United Kingdom|
|   540644|    22751|FELTCRAFT PRINCES...|       8|1/10/11 14:16|     3.75|     16303|United Kingdom|
|   536366|    22633|HAND WARMER UNION...|       6| 12/1/10 8:28|     1.85|     17850|United Kingdom|
|   540644|    22147|FELTCRAFT BUTTERF...|      12|1/10/11 14:16|     1.45|     16303|United Kingdom|
|   536366|    22632|HAND WARMER RED P...|       6| 12/1/10 8:28|     1.85|     17850|United Kingdom|
```

# Sort Multiple Columns

```
ndf = df.sort('Country','Quantity',ascending=[True,False])

ndf.display()
```

Table

| | InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | Cu |
|---|---|---|---|---|---|---|---|
| 1 | 540267 | 22492 | MINI PAINT SET VINTAGE | 576 | 1/6/11 11:12 | 0.55 | 12 |
| 2 | 540267 | 21915 | RED HARMONICA IN BOX | 240 | 1/6/11 11:12 | 1.06 | 12 |
| 3 | 540267 | 21914 | BLUE HARMONICA IN BOX | 240 | 1/6/11 11:12 | 1.06 | 12 |
| 4 | 540267 | 22720 | SET OF 3 CAKE TINS PANTRY DESIGN | 240 | 1/6/11 11:12 | 4.25 | 12 |
| 5 | 540267 | 22522 | CHILDS GARDEN FORK BLUE | 192 | 1/6/11 11:12 | 0.72 | 12 |
| 6 | 540267 | 22722 | SET OF 6 SPICE TINS PANTRY DESIGN | 168 | 1/6/11 11:12 | 3.45 | 12 |
| 7 | 540267 | 22620 | 4 TRADITIONAL SPINNING TOPS | 160 | 1/6/11 11:12 | 1.06 | 12 |

10,000 rows | Truncated data

# Alternative Method To Sort : orderBy

```
ndf = df.orderBy('Country')
ndf.display()
```

Table

| | InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | Cu |
|---|---|---|---|---|---|---|---|
| 1 | 540700 | 21581 | SKULLS DESIGN COTTON TOTE BAG | 6 | 1/11/11 9:47 | 2.25 | 12 |
| 2 | 536389 | 22941 | CHRISTMAS LIGHTS 10 REINDEER | 6 | 12/1/10 10:03 | 8.5 | 12 |
| 3 | 540700 | 22619 | SET OF 6 SOLDIER SKITTLES | 8 | 1/11/11 9:47 | 3.75 | 12 |
| 4 | 536389 | 21622 | VINTAGE UNION JACK CUSHION COVER | 8 | 12/1/10 10:03 | 4.95 | 12 |
| 5 | 540700 | 84997B | RED 3 PIECE RETROSPOT CUTLERY SET | 6 | 1/11/11 9:47 | 3.75 | 12 |
| 6 | 536389 | 21791 | VINTAGE HEADS AND TAILS CARD GAME | 12 | 12/1/10 10:03 | 1.25 | 12 |
| 7 | 540700 | 20727 | LUNCH BAG BLACK SKULL. | 20 | 1/11/11 9:47 | 1.65 | 12 |

10,000 rows | Truncated data

# How To Create New Column

You can create a calculated column with the help of **withColumn**. For Example We will create a column name **Payment** that is multiplication of UnitPrice and Quantity.

```
df = df.withColumn('Payment',col('Quantity')*col('UnitPrice'))
df.show()
```

```
+---------+---------+--------------------+--------+------------+---------+----------+--------------+-----------------
-+
|InvoiceNo|StockCode|         Description|Quantity| InvoiceDate|UnitPrice|CustomerID|       Country|           Paymen
t|
+---------+---------+--------------------+--------+------------+---------+----------+--------------+-----------------
-+
|   536365|   85123A|WHITE HANGING HEA...|       6|12/1/10 8:26|     2.55|     17850|United Kingdom|15.29999999999999
9|
|   536365|    71053| WHITE METAL LANTERN|       6|12/1/10 8:26|     3.39|     17850|United Kingdom|             20.3
```

```
4|
|   536365|   84406B|CREAM CUPID HEART...|       8|12/1/10 8:26|     2.75|     17850|United Kingdom|              22.
0|
|   536365|   84029G|KNITTED UNION FLA...|       6|12/1/10 8:26|     3.39|     17850|United Kingdom|             20.3
4|
|   536365|   84029E|RED WOOLLY HOTTIE...|       6|12/1/10 8:26|     3.39|     17850|United Kingdom|             20.3
4|
|   536365|    22752|SET 7 BABUSHKA NE...|       2|12/1/10 8:26|     7.65|     17850|United Kingdom|              15.
3|
|   536365|    21730|GLASS STAR FROSTE...|       6|12/1/10 8:26|     4.25|     17850|United Kingdom|              25.
5|
|   536366|    22633|HAND WARMER UNION...|       6|12/1/10 8:28|     1.85|     17850|United Kingdom|11.10000000000000
```

# How To Create A New Column With A Constant Value In It.

Let's Say we want to Create a Column with a Constant Value i.e. 1 in it . We will have to use a function **lit**. The lit function is used to create a literal column with a specific value. You can replace lit(1) with any other constant value you want, such as lit("Hello") to create a new column with a constant string value.

We will have to import lit from pyspark.sql.functions.

```
from pyspark.sql.functions import lit

ndf = df.withColumn('new',lit(1))
ndf.show()
```

```
+---------+---------+--------------------+--------+------------+---------+----------+--------------+-----------------
-+---+
|InvoiceNo|StockCode|         Description|Quantity| InvoiceDate|UnitPrice|CustomerID|       Country|           Paymen
t|new|
+---------+---------+--------------------+--------+------------+---------+----------+--------------+-----------------
-+---+
|   536365|   85123A|WHITE HANGING HEA...|       6|12/1/10 8:26|     2.55|     17850|United Kingdom|15.29999999999999
9|  1|
|   536365|    71053| WHITE METAL LANTERN|       6|12/1/10 8:26|     3.39|     17850|United Kingdom|             20.3
4|  1|
|   536365|   84406B|CREAM CUPID HEART...|       8|12/1/10 8:26|     2.75|     17850|United Kingdom|              22.
0|  1|
|   536365|   84029G|KNITTED UNION FLA...|       6|12/1/10 8:26|     3.39|     17850|United Kingdom|             20.3
4|  1|
|   536365|   84029E|RED WOOLLY HOTTIE...|       6|12/1/10 8:26|     3.39|     17850|United Kingdom|             20.3
4|  1|
|   536365|    22752|SET 7 BABUSHKA NE...|       2|12/1/10 8:26|     7.65|     17850|United Kingdom|              15.
3|  1|
|   536365|    21730|GLASS STAR FROSTE...|       6|12/1/10 8:26|     4.25|     17850|United Kingdom|              25.
5|  1|
|   536366|    22633|HAND WARMER UNION...|       6|12/1/10 8:28|     1.85|     17850|United Kingdom|11.10000000000000
```

# How To Delete A Column In Spark

To delete a column in Spark, you can use the drop method on a DataFrame. Here's an example:

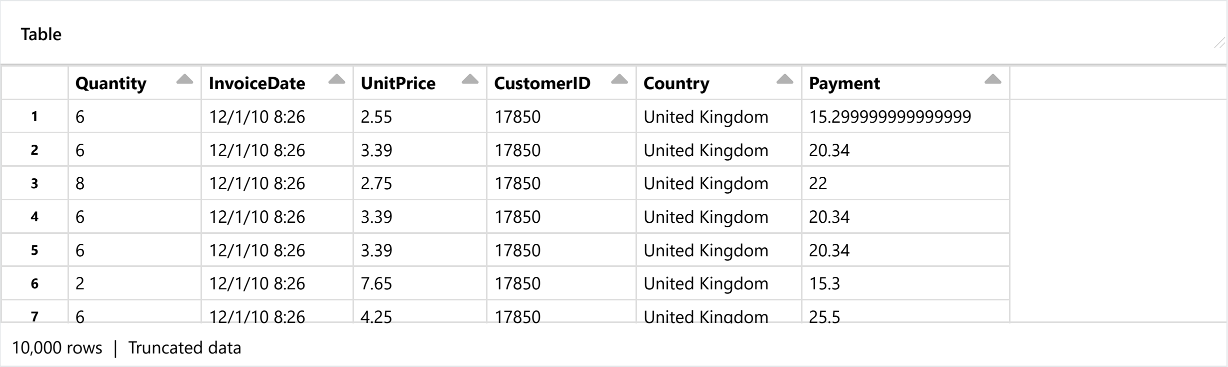
```
# Delete the "Age" column
df = df.drop("Description")

# Display the DataFrame without the deleted column
df.show()
```

```
+---------+---------+--------+------------+---------+----------+--------------+------------------+
|InvoiceNo|StockCode|Quantity| InvoiceDate|UnitPrice|CustomerID|       Country|           Payment|
+---------+---------+--------+------------+---------+----------+--------------+------------------+
|   536365|   85123A|       6|12/1/10 8:26|     2.55|     17850|United Kingdom|15.299999999999999|
|   536365|    71053|       6|12/1/10 8:26|     3.39|     17850|United Kingdom|             20.34|
|   536365|   84406B|       8|12/1/10 8:26|     2.75|     17850|United Kingdom|              22.0|
|   536365|   84029G|       6|12/1/10 8:26|     3.39|     17850|United Kingdom|             20.34|
|   536365|   84029E|       6|12/1/10 8:26|     3.39|     17850|United Kingdom|             20.34|
|   536365|    22752|       2|12/1/10 8:26|     7.65|     17850|United Kingdom|              15.3|
|    536365|    21730|       6|12/1/10 8:26|     4.25|     17850|United Kingdom|              25.5|
|    536366|    22633|       6|12/1/10 8:28|     1.85|     17850|United Kingdom|11.100000000000001|
|    536366|    22632|       6|12/1/10 8:28|     1.85|     17850|United Kingdom|11.100000000000001|
|    536367|    84879|      32|12/1/10 8:34|     1.69|     13047|United Kingdom|             54.08|
|    536367|    22745|       6|12/1/10 8:34|      2.1|     13047|United Kingdom|12.600000000000001|
|    536367|    22748|       6|12/1/10 8:34|      2.1|     13047|United Kingdom|12.600000000000001|
|    536367|    22749|       8|12/1/10 8:34|     3.75|     13047|United Kingdom|              30.0|
|    536367|    22310|       6|12/1/10 8:34|     1.65|     13047|United Kingdom| 9.899999999999999|
|    536367|    84969|       6|12/1/10 8:34|     4.25|     13047|United Kingdom|              25.5|
|    536367|    22623|       3|12/1/10 8:34|     4.95|     13047|United Kingdom|14.850000000000001|
|    536367|    22622|       2|12/1/10 8:34|     9.95|     13047|United Kingdom|              19.9|
```

You can also delete multiple columns by passing a list of column names to the drop method. For example, to delete both the "InvoiceNo" and "StockCode" columns:

```
ndf = df.drop('InvoiceNo','StockCode')
ndf.display()
```



Table

|   | Quantity | InvoiceDate | UnitPrice | CustomerID | Country | Payment |
|---|---|---|---|---|---|---|
| 1 | 6 | 12/1/10 8:26 | 2.55 | 17850 | United Kingdom | 15.299999999999999 |
| 2 | 6 | 12/1/10 8:26 | 3.39 | 17850 | United Kingdom | 20.34 |
| 3 | 8 | 12/1/10 8:26 | 2.75 | 17850 | United Kingdom | 22 |
| 4 | 6 | 12/1/10 8:26 | 3.39 | 17850 | United Kingdom | 20.34 |
| 5 | 6 | 12/1/10 8:26 | 3.39 | 17850 | United Kingdom | 20.34 |
| 6 | 2 | 12/1/10 8:26 | 7.65 | 17850 | United Kingdom | 15.3 |
| 7 | 6 | 12/1/10 8:26 | 4.25 | 17850 | United Kingdom | 25.5 |

10,000 rows | Truncated data

Note that the drop method returns a new DataFrame with the specified columns removed. It does not modify the original DataFrame in place.

# GroupBy In Spark

In Spark, you can use the groupBy method to group a DataFrame by one or more columns. This allows you to perform aggregate operations on the grouped data.

# What Top 5 Countries Has Highest Number Of Sales

```
pdf = df.groupBy('Country').sum('Payment')
pdf.sort('sum(Payment)',ascending=False).show(5)
```

```
+--------------+------------------+
|       Country|      sum(Payment)|
+--------------+------------------+
|United Kingdom| 965042.7499999746|
|          EIRE|29037.420000000002|
|   Netherlands|27200.859999999993|
|       Germany|22237.810000000005|
|        France|21773.329999999998|
+--------------+------------------+
only showing top 5 rows
```

# Aggregate Functions And GroupBy

We can also use groupby with aggregate Functions. Like in this example , we wanted to know the total sales and average sales in each country.

```
tdf = df.groupby('Country').agg(

sum('Payment').alias('TotalSales'),    # alias is to give the column name
avg('Payment').alias('AverageSales')

)

tdf.display()
```

Table

| | Country | TotalSales | AverageSales |
|---|---|---|---|
| 1 | Sweden | 3153.859999999999 | 76.92341463414633 |
| 2 | Germany | 22237.810000000005 | 22.645427698574345 |
| 3 | France | 21773.329999999998 | 22.516370217166493 |
| 4 | Belgium | 2640.5199999999995 | 18.59521126760563 |
| 5 | Finland | 892.8000000000001 | 52.517647058823535 |
| 6 | Italy | 2395.5099999999993 | 21.388482142857136 |
| 7 | EIRE | 29037.420000000002 | 57.61392857142857 |

24 rows

Table

| | path | name | size | modificationTime |
|---|---|---|---|---|
| 1 | dbfs:/databricks-datasets/ | databricks-datasets/ | 0 | 0 |
| 2 | dbfs:/databricks-results/ | databricks-results/ | 0 | 0 |

2 rows