Data Cleaning with PySpark

April 10, 2020

1 Data Cleaning with PySpark

This document contains very useful codes to start spark session to work on data. The functions used here clean text data, filter data and sort data. These functions can be very useful to handle large datasets. Furthermore, addition of new columns based on some conditions is also demonstrated alongwith adding row IDs.

[ ]:
```
# Importing findspark
import findspark
# Initialize and provide path
findspark.init("/usr/local/spark")
# Or use this alternative
#findspark.init()

# Importing SparkSession
from pyspark.sql import SparkSession

# Building the SparkSession
spark = SparkSession.builder \
   .master("local") \
   .appName("Linear Regression Model") \
   .config("spark.executor.memory", "1gb") \
   .getOrCreate()

sc = spark.sparkContext
```

[160]:
```
from pyspark.sql import SparkSession
spark = SparkSession \
.builder \
.appName("Python Spark SQL basic example") \
.config("spark.some.config.option", "some-value") \
.getOrCreate()
```

[161]:
```
file_path = '/Users/MuhammadBilal/Desktop/Data Camp/PySpark/Data Cleaning with␣
 ↪PySpark/Data/DallasCouncilVoters.csv'
```

[162]:
```
import pandas as pd
```

```
# Importing pyspark.sql.functions as F
import pyspark.sql.functions as F
```

[163]:
```
# Loading the CSV file
voter_df = spark.read.format('csv').options(Header=True).
 ↪load('DallasCouncilVoters.csv')
```

[164]:
```
print(voter_df)
```

```
DataFrame[DATE: string, TITLE: string, VOTER_NAME: string, _c3: string]
```

[165]:
```
# Showing the distinct VOTER_NAME entries
voter_df.select(voter_df['VOTER_NAME']).distinct().show(40, truncate=False)

# Filtering voter_df where the VOTER_NAME is 1-20 characters in length
voter_df = voter_df.filter('length(VOTER_NAME) > 0 and length(VOTER_NAME) < 20')

# Filtering out voter_df where the VOTER_NAME contains an underscore
voter_df = voter_df.filter(~ F.col('VOTER_NAME').contains('_'))

# Showing the distinct VOTER_NAME entries again
voter_df.select('VOTER_NAME').distinct().show(40, truncate=False)
```

```
+------------------+
|VOTER_NAME        |
+------------------+
|Jennifer S. Gates |
|Philip T. Kingston|
|Michael S. Rawlings|
|Adam Medrano      |
|Casey Thomas      |
|Carolyn King Arnold|
|Scott Griggs      |
|B. Adam  McGough  |
|Lee Kleinman      |
|Sandy Greyson     |
|Casey Thomas"     |
|Rickey D. Callahan|
|Sandy  Greyson    |
|Jennifer S.  Gates|
|Philip T.  Kingston|
|Dwaine R. Caraway |
|Rickey D.  Callahan|
|Omar Narvaez      |
|Kevin Felder      |
|Tennell Atkins    |
|Mark  Clayton     |
```

```
|Scott  Griggs      |
|Lee M. Kleinman    |
|null               |
|Casey  Thomas"     |
|Casey  Thomas      |
|Monica R. Alonzo   |
|Tiffinni A. Young  |
|Erik Wilson        |
|Mark Clayton       |
|011018__42         |
+-------------------+
```

```
+-------------------+
|VOTER_NAME         |
+-------------------+
|Jennifer S. Gates  |
|Philip T. Kingston |
|Michael S. Rawlings|
|Adam Medrano       |
|Casey Thomas       |
|Carolyn King Arnold|
|Scott Griggs       |
|B. Adam  McGough   |
|Lee Kleinman       |
|Sandy Greyson      |
|Casey Thomas"      |
|Rickey D. Callahan |
|Sandy  Greyson     |
|Jennifer S.  Gates |
|Philip T.  Kingston|
|Dwaine R. Caraway  |
|Rickey D.  Callahan|
|Omar Narvaez       |
|Kevin Felder       |
|Tennell Atkins     |
|Mark  Clayton      |
|Scott  Griggs      |
|Lee M. Kleinman    |
|Casey  Thomas"     |
|Casey  Thomas      |
|Monica R. Alonzo   |
|Tiffinni A. Young  |
|Erik Wilson        |
|Mark Clayton       |
+-------------------+
```

[166]:
```
# Importing pyspark.sql.functions as F
import pyspark.sql.functions as F

# Adding a column to voter_df for any voter with the title **Councilmember**
voter_df = voter_df.withColumn('random_val',
                               F.when(voter_df.TITLE == 'Councilmember', F.
 ↪rand()))

# Showing some of the DataFrame rows, noting whether the when clause worked
voter_df.show()
```

```
+----------+-------------+-------------------+----+-------------------+
|      DATE|        TITLE|         VOTER_NAME| _c3|         random_val|
+----------+-------------+-------------------+----+-------------------+
|02/08/2017|Councilmember|  Jennifer S. Gates|null| 0.2297909515137122|
|02/08/2017|Councilmember| Philip T. Kingston|null| 0.4340989569813255|
|02/08/2017|        Mayor|Michael S. Rawlings|null|               null|
|02/08/2017|Councilmember|       Adam Medrano|null| 0.6615422817570464|
|02/08/2017|Councilmember|       Casey Thomas|null| 0.9417275397661361|
|02/08/2017|Councilmember|Carolyn King Arnold|null|  0.998207049080341|
|02/08/2017|Councilmember|       Scott Griggs|null| 0.7724787582907507|
|02/08/2017|Councilmember|   B. Adam  McGough|null|0.33050314282366566|
|02/08/2017|Councilmember|       Lee Kleinman|null| 0.8471924017388808|
|02/08/2017|Councilmember|      Sandy Greyson|null|0.48206810286747703|
|02/08/2017|Councilmember|  Jennifer S. Gates|null|0.37766679559603666|
|02/08/2017|Councilmember| Philip T. Kingston|null| 0.9780404074336463|
|02/08/2017|        Mayor|Michael S. Rawlings|null|               null|
|02/08/2017|Councilmember|       Adam Medrano|null| 0.8167481498552961|
|02/08/2017|Councilmember|      Casey Thomas"|null|0.30656746357126485|
|02/08/2017|Councilmember|Carolyn King Arnold|null| 0.4621936573742206|
|02/08/2017|Councilmember| Rickey D. Callahan|null| 0.9831457897810741|
|01/11/2017|Councilmember|  Jennifer S. Gates|null|0.24918024159382401|
|04/25/2018|Councilmember|     Sandy  Greyson|null| 0.6268863717545166|
|04/25/2018|Councilmember| Jennifer S.  Gates|null| 0.9088197548661604|
+----------+-------------+-------------------+----+-------------------+
only showing top 20 rows
```

[167]:
```
# Adding a column to voter_df for a voter based on their position
voter_df = voter_df.withColumn('random_val',
                               F.when(voter_df.TITLE == 'Councilmember', F.
 ↪rand())
                               .when(voter_df.TITLE == 'Mayor', 2)
                               .otherwise(0))

# Showing some of the DataFrame rows
voter_df.show()
```

```
# Using the .filter() clause with random_val
voter_df.filter(voter_df.random_val == 0).show()
```

```
+----------+-------------+-------------------+----+-------------------+
|      DATE|        TITLE|         VOTER_NAME| _c3|         random_val|
+----------+-------------+-------------------+----+-------------------+
|02/08/2017|Councilmember|  Jennifer S. Gates|null| 0.8644351120082215|
|02/08/2017|Councilmember| Philip T. Kingston|null| 0.8155637287245125|
|02/08/2017|        Mayor|Michael S. Rawlings|null|                2.0|
|02/08/2017|Councilmember|       Adam Medrano|null|0.27037893697166593|
|02/08/2017|Councilmember|       Casey Thomas|null| 0.2687923714348158|
|02/08/2017|Councilmember|Carolyn King Arnold|null| 0.7358533109308102|
|02/08/2017|Councilmember|       Scott Griggs|null| 0.2605767071497559|
|02/08/2017|Councilmember|   B. Adam  McGough|null|0.13065050235887377|
|02/08/2017|Councilmember|       Lee Kleinman|null| 0.8135707436592344|
|02/08/2017|Councilmember|      Sandy Greyson|null| 0.4201808324735504|
|02/08/2017|Councilmember|  Jennifer S. Gates|null|0.15101933309579108|
|02/08/2017|Councilmember| Philip T. Kingston|null|0.04645795958372578|
|02/08/2017|        Mayor|Michael S. Rawlings|null|                2.0|
|02/08/2017|Councilmember|       Adam Medrano|null|0.07096191874031543|
|02/08/2017|Councilmember|      Casey Thomas"|null|0.47978955155569747|
|02/08/2017|Councilmember|Carolyn King Arnold|null| 0.3607247885303757|
|02/08/2017|Councilmember| Rickey D. Callahan|null| 0.7015530106679958|
|01/11/2017|Councilmember|  Jennifer S. Gates|null|0.36920792760599186|
|04/25/2018|Councilmember|     Sandy  Greyson|null| 0.6176342406331573|
|04/25/2018|Councilmember| Jennifer S.  Gates|null| 0.5640947585167294|
+----------+-------------+-------------------+----+-------------------+
only showing top 20 rows
```

```
+----------+--------------------+-----------------+----+----------+
|      DATE|               TITLE|       VOTER_NAME| _c3|random_val|
+----------+--------------------+-----------------+----+----------+
|04/25/2018|Deputy Mayor Pro Tem|     Adam Medrano|null|       0.0|
|04/25/2018|       Mayor Pro Tem|Dwaine R. Caraway|null|       0.0|
|06/20/2018|Deputy Mayor Pro Tem|     Adam Medrano|null|       0.0|
|06/20/2018|       Mayor Pro Tem|Dwaine R. Caraway|null|       0.0|
|06/20/2018|Deputy Mayor Pro Tem|     Adam Medrano|null|       0.0|
|06/20/2018|       Mayor Pro Tem|Dwaine R. Caraway|null|       0.0|
|08/15/2018|Deputy Mayor Pro Tem|     Adam Medrano|null|       0.0|
|08/15/2018|Deputy Mayor Pro Tem|     Adam Medrano|null|       0.0|
|09/18/2018|Deputy Mayor Pro Tem|     Adam Medrano|null|       0.0|
|09/18/2018|       Mayor Pro Tem|   Casey  Thomas"|null|       0.0|
|04/25/2018|Deputy Mayor Pro Tem|     Adam Medrano|null|       0.0|
|04/25/2018|       Mayor Pro Tem|Dwaine R. Caraway|null|       0.0|
|04/11/2018|       Mayor Pro Tem|Dwaine R. Caraway|null|       0.0|
|04/11/2018|Deputy Mayor Pro Tem|     Adam Medrano|null|       0.0|
```

```
|04/11/2018|       Mayor Pro Tem|Dwaine R. Caraway|null|       0.0|
|04/11/2018|Deputy Mayor Pro Tem|     Adam Medrano|null|       0.0|
|04/11/2018|       Mayor Pro Tem|Dwaine R. Caraway|null|       0.0|
|06/13/2018|Deputy Mayor Pro Tem|     Adam Medrano|null|       0.0|
|06/13/2018|       Mayor Pro Tem|Dwaine R. Caraway|null|       0.0|
|04/11/2018|Deputy Mayor Pro Tem|     Adam Medrano|null|       0.0|
+----------+--------------------+-----------------+----+----------+
only showing top 20 rows
```

[168]:
```
# Adding a new column called splits separated on whitespace
voter_df = voter_df.withColumn('splits', F.split(voter_df.VOTER_NAME, '\s+'))

# Createing a new column called first_name based on the first item in splits
voter_df = voter_df.withColumn('first_name', voter_df.splits.getItem(0))

# Createing a new column called first_name based on the first item in splits
voter_df = voter_df.withColumn('last_name', voter_df.splits.getItem(1))

voter_df.show()
```

```
+----------+------------+------------------+----+------------------+---------
----------+----------+---------+
|      DATE|       TITLE|        VOTER_NAME| _c3|        random_val|
splits|first_name|last_name|
+----------+------------+------------------+----+------------------+---------
----------+----------+---------+
|02/08/2017|Councilmember|  Jennifer S. Gates|null|
0.8644351120082215|[Jennifer, S., Ga…|  Jennifer|       S.|
|02/08/2017|Councilmember| Philip T. Kingston|null| 0.8155637287245125|[Philip,
T., King…|    Philip|       T.|
|02/08/2017|       Mayor|Michael S. Rawlings|null|               2.0|[Michael,
S., Raw…|   Michael|       S.|
|02/08/2017|Councilmember|       Adam Medrano|null|0.27037893697166593|
[Adam, Medrano]|      Adam|  Medrano|
|02/08/2017|Councilmember|       Casey Thomas|null| 0.2687923714348158|
[Casey, Thomas]|     Casey|   Thomas|
|02/08/2017|Councilmember|Carolyn King Arnold|null| 0.7358533109308102|[Carolyn,
King, A…|   Carolyn|     King|
|02/08/2017|Councilmember|       Scott Griggs|null| 0.2605767071497559|
[Scott, Griggs]|     Scott|   Griggs|
|02/08/2017|Councilmember|   B. Adam  McGough|null|0.13065050235887377| [B.,
Adam, McGough]|        B.|     Adam|
|02/08/2017|Councilmember|       Lee Kleinman|null| 0.8135707436592344|
[Lee, Kleinman]|       Lee| Kleinman|
|02/08/2017|Councilmember|      Sandy Greyson|null| 0.4201808324735504|
[Sandy, Greyson]|     Sandy|  Greyson|
|02/08/2017|Councilmember|  Jennifer S.
```

```
Gates|null|0.15101933309579108|[Jennifer, S., Ga…|  Jennifer|       S.|
|02/08/2017|Councilmember| Philip T. Kingston|null|0.04645795958372578|[Philip,
T., King…|    Philip|       T.|
|02/08/2017|        Mayor|Michael S. Rawlings|null|                2.0|[Michael,
S., Raw…|   Michael|       S.|
|02/08/2017|Councilmember|       Adam Medrano|null|0.07096191874031543|
[Adam, Medrano]|      Adam|  Medrano|
|02/08/2017|Councilmember|      Casey Thomas"|null|0.47978955155569747|
[Casey, Thomas"]|     Casey|  Thomas"|
|02/08/2017|Councilmember|Carolyn King Arnold|null| 0.3607247885303757|[Carolyn,
King, A…|   Carolyn|     King|
|02/08/2017|Councilmember| Rickey D. Callahan|null| 0.7015530106679958|[Rickey,
D., Call…|    Rickey|       D.|
|01/11/2017|Councilmember|  Jennifer S.
Gates|null|0.36920792760599186|[Jennifer, S., Ga…|  Jennifer|       S.|
|04/25/2018|Councilmember|     Sandy  Greyson|null| 0.6176342406331573|
[Sandy, Greyson]|     Sandy|  Greyson|
|04/25/2018|Councilmember| Jennifer S.  Gates|null|
0.5640947585167294|[Jennifer, S., Ga…|  Jennifer|       S.|
+----------+-------------+-------------------+----+------------------+---------
-----------+----------+---------+
only showing top 20 rows
```

[169]:
```
from pyspark.sql.types import StringType
def getFirstAndMiddle(names):
  # Return a space separated string of names
  return ' '.join(names[:-1])

# Defining the method as a UDF
udfFirstAndMiddle = F.udf(getFirstAndMiddle, StringType())

# Creating a new column using your UDF
voter_df = voter_df.withColumn('first_and_middle_name',␣
 ↪udfFirstAndMiddle(voter_df.splits))

# Dropping the unnecessary columns then showing the DataFrame
voter_df = voter_df.drop('first_name')
voter_df = voter_df.drop('splits')
voter_df.show()
```

```
+----------+-------------+-------------------+----+------------------+---------
+---------------------+
|      DATE|        TITLE|         VOTER_NAME| _c3|
random_val|last_name|first_and_middle_name|
+----------+-------------+-------------------+----+------------------+---------
+---------------------+
|02/08/2017|Councilmember|  Jennifer S. Gates|null| 0.8644351120082215|
```

```
S.|          Jennifer S.|
|02/08/2017|Councilmember| Philip T. Kingston|null| 0.8155637287245125|
T.|            Philip T.|
|02/08/2017|        Mayor|Michael S. Rawlings|null|                2.0|
S.|           Michael S.|
|02/08/2017|Councilmember|       Adam Medrano|null|0.27037893697166593|
Medrano|                 Adam|
|02/08/2017|Councilmember|       Casey Thomas|null| 0.2687923714348158|
Thomas|                Casey|
|02/08/2017|Councilmember|Carolyn King Arnold|null| 0.7358533109308102|
King|         Carolyn King|
|02/08/2017|Councilmember|       Scott Griggs|null| 0.2605767071497559|
Griggs|                Scott|
|02/08/2017|Councilmember|   B. Adam  McGough|null|0.13065050235887377|
Adam|              B. Adam|
|02/08/2017|Councilmember|       Lee Kleinman|null| 0.8135707436592344|
Kleinman|                  Lee|
|02/08/2017|Councilmember|      Sandy Greyson|null| 0.4201808324735504|
Greyson|               Sandy|
|02/08/2017|Councilmember|  Jennifer S. Gates|null|0.15101933309579108|
S.|          Jennifer S.|
|02/08/2017|Councilmember| Philip T. Kingston|null|0.04645795958372578|
T.|            Philip T.|
|02/08/2017|        Mayor|Michael S. Rawlings|null|                2.0|
S.|           Michael S.|
|02/08/2017|Councilmember|       Adam Medrano|null|0.07096191874031543|
Medrano|                 Adam|
|02/08/2017|Councilmember|      Casey Thomas"|null|0.47978955155569747|
Thomas"|               Casey|
|02/08/2017|Councilmember|Carolyn King Arnold|null| 0.3607247885303757|
King|         Carolyn King|
|02/08/2017|Councilmember| Rickey D. Callahan|null| 0.7015530106679958|
D.|            Rickey D.|
|01/11/2017|Councilmember|  Jennifer S. Gates|null|0.36920792760599186|
S.|          Jennifer S.|
|04/25/2018|Councilmember|     Sandy  Greyson|null| 0.6176342406331573|
Greyson|               Sandy|
|04/25/2018|Councilmember| Jennifer S.  Gates|null| 0.5640947585167294|
S.|          Jennifer S.|
+----------+-------------+-------------------+----+-------------------+---------
+---------------------+
only showing top 20 rows
```

[180]:
```
# Counting the rows in voter_df
print("\nThere are %d rows in the voter_df DataFrame.\n" % voter_df.count())
```

```
# Adding a ROW_ID
voter_df = voter_df.withColumn('ROW_ID', F.monotonically_increasing_id())

# Showing the rows with 10 highest IDs in the set
voter_df.orderBy(voter_df.ROW_ID.desc()).show(10)
```

```
There are 43515 rows in the voter_df DataFrame.

+----------+--------------------+------------------+----+--------------------+-
--------+---------------------+------+
|      DATE|               TITLE|         VOTER_NAME| _c3|
random_val|last_name|first_and_middle_name|ROW_ID|
+----------+--------------------+------------------+----+--------------------+-
--------+---------------------+------+
|11/20/2018|       Councilmember|      Mark  Clayton|null| 0.05723639309034956|
Clayton|                Mark| 43514|
|11/20/2018|       Councilmember|     Tennell Atkins|null| 0.29540794069367904|
Atkins|              Tennell| 43513|
|11/20/2018|       Councilmember|       Kevin Felder|null|  0.5438480318254185|
Felder|               Kevin| 43512|
|11/20/2018|       Councilmember|       Omar Narvaez|null| 0.27402155387798866|
Narvaez|                Omar| 43511|
|11/20/2018|       Councilmember|Rickey D.  Callahan|null| 0.42388973473989966|
D.|            Rickey D.| 43510|
|11/20/2018|       Mayor Pro Tem|     Casey  Thomas"|null|                 0.0|
Thomas"|               Casey| 43509|
|11/20/2018|Deputy Mayor Pro Tem|       Adam Medrano|null|                 0.0|
Medrano|                Adam| 43508|
|11/20/2018|               Mayor|Michael S. Rawlings|null|                 2.0|
S.|          Michael S.| 43507|
|11/20/2018|       Councilmember|Philip T.  Kingston|null|0.014168346407937848|
T.|           Philip T.| 43506|
|11/20/2018|       Councilmember| Jennifer S.  Gates|null|  0.5143874695312031|
S.|         Jennifer S.| 43505|
+----------+--------------------+------------------+----+--------------------+-
--------+---------------------+------+
only showing top 10 rows
```