



Chapter 5: Data Pre-processing

Ex1: DallasCouncilVoters

Cho dữ liệu DallasCouncilVoters.csv

Yêu cầu:

1. Đọc dữ liệu => df
2. Cho biết dữ liệu có bao nhiêu dòng, in scheme. Hiển thị 5 dòng dữ liệu đầu tiên.
3. Kiểm tra dữ liệu NaN, null. Nếu dòng nào 'VOTER_NAME' có dữ liệu null thì xóa hết các dòng đó.
4. Kiểm tra dữ liệu trùng. Xóa dữ liệu trùng.
5. Tìm các VOTER_NAME duy nhất và hiển thị 10 thông tin đầu tiên.
6. Lọc dữ liệu theo điều kiện 'VOTER_NAME' có chiều dài từ 1-20 ký tự.
7. Loại bỏ các dữ liệu mà trong 'VOTER_NAME' có chứa dấu '_' (underscore)
8. Tạo cột 'splits' chứa thông tin được cắt theo khoảng trắng từ 'VOTER_NAME'
9. Tạo cột 'first_name' lấy dữ liệu từ phần tử đầu tiên trong cột 'splits'
10. Tạo cột 'last_name' lấy dữ liệu từ phần tử cuối cùng trong cột 'splits'
11. Tạo cột 'random_val' theo điều kiện: nếu cột 'TITLE' có nội dung là 'Councilmember' thì 'random_val' sẽ có giá trị rand(), nếu có nội dung là 'Mayor' thì 'random_val' sẽ có giá trị là 2, ngược lại sẽ có giá trị là 0.
12. Lọc các dòng dữ liệu có 'random_val' = 0. Hiển thị.
13. Xây dựng function: getFirstAndMiddle(names) trả về kết quả gồm First và Middle (names). Khai báo function vừa viết dưới dạng udf đặt tên là udfFirstAndMiddle.
14. Tạo cột first_and_middle_name bằng cách gọi udf trên với tham số truyền vào là cột 'splits'. In kết quả.
15. Xóa bỏ các cột 'first_name', 'splits'. In kết quả.
16. Thêm cột 'ROW_ID' bằng phương thức: monotonically_increasing_id() (trong pyspark.sql.functions).
17. Hiển thị 10 dòng đầu của dữ liệu với ROW_ID tăng dần.

```
In [1]: import findspark
findspark.init()
```

```
In [2]: from pyspark import SparkContext
from pyspark.conf import SparkConf
from pyspark.sql import SparkSession
```

```
In [3]: sc = SparkContext()
```

```
In [4]: spark = SparkSession(sc)
```



```
In [5]: #1.
df = spark.read.csv('voters_data/DallasCouncilVoters.csv', header=True,
                    inferSchema=True)
```

```
In [6]: #2.
df.count()
```

Out[6]: 44625

```
In [7]: df.printSchema()

root
 |-- DATE: string (nullable = true)
 |-- TITLE: string (nullable = true)
 |-- VOTER_NAME: string (nullable = true)
```

```
In [8]: df.show(5)
```

```
+-----+-----+-----+
|   DATE|   TITLE| VOTER_NAME|
+-----+-----+-----+
|02/08/2017|Councilmember| Jennifer S. Gates|
|02/08/2017|Councilmember| Philip T. Kingston|
|02/08/2017|      Mayor|Michael S. Rawlings|
|02/08/2017|Councilmember|      Adam Medrano|
|02/08/2017|Councilmember|      Casey Thomas|
+-----+-----+-----+
only showing top 5 rows
```

```
In [9]: from pyspark.sql.functions import col, udf
from pyspark.sql.functions import isnan, when, count, col
```

```
In [10]: #3. Kiểm tra dữ liệu NaN, null
df.select([count(when(isnan(c), c)).alias(c) for c in df.columns]).toPandas().T
```

```
Out[10]:
```

	0
DATE	0
TITLE	0
VOTER_NAME	0

```
In [11]: # => Không có dữ liệu NaN
```



```
In [12]: df.select([count(when(col(c).isNull(), c)).alias(c) for c in
                df.columns]).toPandas().T
```

Out[12]:

	0
DATE	0
TITLE	195
VOTER_NAME	503

```
In [13]: # => Có dữ liệu null. Xóa dữ liệu có VOTER_NAME null
```

```
In [14]: df = df.dropna(subset='VOTER_NAME')
```

```
In [15]: df.select([count(when(col(c).isNull(), c)).alias(c) for c in
                df.columns]).toPandas().T
```

Out[15]:

	0
DATE	0
TITLE	0
VOTER_NAME	0

```
In [16]: # => hết dữ liệu null
```

```
In [17]: #4.
num_rows = df.count()
num_dist_rows = df.distinct().count()
dup_rows = num_rows - num_dist_rows
```

```
In [18]: display(num_rows, num_dist_rows, dup_rows)
```

44122

1273

42849



In [19]: *# Check duplicate*
`df.filter(df['VOTER_NAME'] == 'Philip T. Kingston').show(5)`

```
+-----+-----+-----+
|      DATE|      TITLE|    VOTER_NAME|
+-----+-----+-----+
|02/08/2017|Councilmember|Philip T. Kingston|
|02/08/2017|Councilmember|Philip T. Kingston|
|01/11/2017|Councilmember|Philip T. Kingston|
|09/14/2016|Councilmember|Philip T. Kingston|
|01/04/2017|Councilmember|Philip T. Kingston|
+-----+-----+-----+
only showing top 5 rows
```

In [20]: `df = df.drop_duplicates()`

In [21]: `df.count()`

Out[21]: 1273

In [22]: *#5. Show the distinct VOTER_NAME entries*
`df.select(df['VOTER_NAME']).distinct().show(10)`

```
+-----+
|      VOTER_NAME|
+-----+
|Tennell Atkins|
|the final 20...|
|Scott Griggs|
|Scott Griggs|
|Sandy Greyson|
|Michael S. Rawlings|
|the final 2018 A...|
|Kevin Felder|
|Adam Medrano|
|Casey Thomas|
+-----+
only showing top 10 rows
```

In [23]: `from pyspark.sql.functions import *`

In [24]: *#6. Filter df where the VOTER_NAME is 1-20 characters in length*
`df = df.filter('length(VOTER_NAME) > 0 and length(VOTER_NAME) < 20')`



In [25]: `df.show(5)`

```
+-----+-----+-----+
|    DATE|    TITLE|    VOTER_NAME|
+-----+-----+-----+
|04/11/2018|Deputy Mayor Pro Tem|    Adam Medrano|
|02/14/2018|    Councilmember|    Lee M. Kleinman|
|04/25/2018|    Councilmember|    Tennell Atkins|
|08/29/2018|    Councilmember|    Kevin Felder|
|10/18/2017|    Councilmember|Jennifer S.  Gates|
+-----+-----+-----+
```

only showing top 5 rows

In [26]: *#7. Filter out df where the VOTER_NAME contains an underscore*
`df = df.filter(~ col('VOTER_NAME').contains('_'))`

In [27]: *# Show the distinct VOTER_NAME entries again*
`df.select('VOTER_NAME').distinct().show(10, truncate=False)`

```
+-----+
|VOTER_NAME|
+-----+
|Tennell Atkins|
|Scott Griggs|
|Scott Griggs|
|Sandy Greyson|
|Michael S. Rawlings|
|Kevin Felder|
|Adam Medrano|
|Casey Thomas|
|Mark Clayton|
|Casey Thomas|
+-----+
```

only showing top 10 rows

Modifying DataFrame

In [28]: *#8. Add a new column called splits separated on whitespace*
`df = df.withColumn('splits', split(df.VOTER_NAME, '\s+'))`

In [29]: *#9. Create a new column called first_name based on the first item in splits*
`df = df.withColumn('first_name', df.splits.getItem(0))`

In [30]: *#10. Get the last entry of the splits list and create a column called last_name*
`df = df.withColumn('last_name', df.splits.getItem(size('splits') - 1))`

In [31]: *# Drop the splits column*
`# df = df.drop('splits')`



In [32]: *# Show the voter_df DataFrame*
`df.show(3)`

```
+-----+-----+-----+-----+-----+
+-----+
|      DATE|          TITLE|    VOTER_NAME|          splits|first_name
|last_name|
+-----+-----+-----+-----+-----+
+-----+
|04/11/2018|Deputy Mayor Pro Tem|    Adam Medrano|    [Adam, Medrano]|    Adam
|  Medrano|
|02/14/2018|    Councilmember|Lee M. Kleinman|[Lee, M., Kleinman]|    Lee
|  Kleinman|
|04/25/2018|    Councilmember|Tennell Atkins|    [Tennell, Atkins]|    Tennell
|  Atkins|
+-----+-----+-----+-----+-----+
+-----+
only showing top 3 rows
```

In [33]: *#11. Add a column to df for any voter with the title 'Councilmember'*
`df = df.withColumn('random_val', when(df.TITLE == 'Councilmember', rand()))`

In [34]: *# Show some of the DataFrame rows, noting whether the when clause worked*
`#df.show(5)`

In [35]: *#. Add a column to df for a voter based on their position*
`df = df.withColumn('random_val',
 when(df.TITLE == 'Councilmember', rand())
 .when(df.TITLE == 'Mayor', 2)
 .otherwise(0))`



In [36]: *# Show some of the DataFrame rows*
`df.show(5)`

```
+-----+-----+-----+-----+-----+
---+-----+-----+
|      DATE|      TITLE|      VOTER_NAME|      splits|first_
name|last_name|      random_val|
+-----+-----+-----+-----+-----+
---+-----+-----+
|04/11/2018|Deputy Mayor Pro Tem|      Adam Medrano|      [Adam, Medrano]|
Adam|  Medrano|      0.0|
|02/14/2018|      Councilmember|      Lee M. Kleinman|[Lee, M., Kleinman]|
Lee| Kleinman|      0.7266891908590055|
|04/25/2018|      Councilmember|      Tennell Atkins|      [Tennell, Atkins]|      Ten
nell|      Atkins|1.716281340619074...|
|08/29/2018|      Councilmember|      Kevin Felder|      [Kevin, Felder]|      K
evin|      Felder|0.047122114981064556|
|10/18/2017|      Councilmember|Jennifer S. Gates|[Jennifer, S., Ga...| Jenn
ifer|      Gates|      0.47584042942379867|
+-----+-----+-----+-----+-----+
---+-----+-----+
only showing top 5 rows
```

In [37]: *#12. Use the .filter() clause with random_val*
`df.filter(df.random_val == 0).show(5)`

```
+-----+-----+-----+-----+-----+
---+-----+-----+
|      DATE|      TITLE|      VOTER_NAME|      splits|first_n
ame|last_name|random_val|
+-----+-----+-----+-----+-----+
---+-----+-----+
|04/11/2018|Deputy Mayor Pro Tem|      Adam Medrano|      [Adam, Medrano]|      A
dam|  Medrano|      0.0|
|04/12/2017|      Mayor Pro Tem|      Monica R. Alonzo|[Monica, R., Alonzo]|      Mon
ica|      Alonzo|      0.0|
|06/28/2017|Deputy Mayor Pro Tem|      Adam Medrano|      [Adam, Medrano]|      A
dam|  Medrano|      0.0|
|01/03/2018|Deputy Mayor Pro Tem|      Adam Medrano|      [Adam, Medrano]|      A
dam|  Medrano|      0.0|
|01/17/2018|      Mayor Pro Tem|Dwaine R. Caraway|[Dwaine, R., Cara...|      Dwa
ine|      Caraway|      0.0|
+-----+-----+-----+-----+-----+
---+-----+-----+
only showing top 5 rows
```

UDF

In [38]: `from pyspark.sql.types import *`



```
In [39]: def getFirstAndMiddle(names):
# Return a space separated string of names
return ' '.join(names[:-1])
```

```
In [40]: #13. Define the method as a UDF
udfFirstAndMiddle = udf(getFirstAndMiddle, StringType())
```

```
In [41]: #14. Create a new column using your UDF
df = df.withColumn('first_and_middle_name', udfFirstAndMiddle(df.splits))
```

```
In [42]: #15. Drop the unnecessary columns then show the DataFrame
df = df.drop('first_name')
df = df.drop('splits')
```

```
In [43]: df.show(5)
```

```
+-----+-----+-----+-----+-----+
---+-----+
|      DATE|      TITLE|      VOTER_NAME|last_name|      random_
val|first_and_middle_name|
+-----+-----+-----+-----+-----+
---+-----+
|04/11/2018|Deputy Mayor Pro Tem|      Adam Medrano|  Medrano|
0.0|      Adam|
|02/14/2018|      Councilmember|  Lee M. Kleinman| Kleinman|  0.7266891908590
055|      Lee M.|
|04/25/2018|      Councilmember|  Tennell Atkins|  Atkins|1.71628134061907
4...|      Tennell|
|08/29/2018|      Councilmember|      Kevin Felder|  Felder|0.047122114981064
556|      Kevin|
|10/18/2017|      Councilmember|Jennifer S.  Gates|  Gates| 0.47584042942379
867|      Jennifer S.|
+-----+-----+-----+-----+-----+
---+-----+
only showing top 5 rows
```

Adding an ID Field

```
In [44]: # Select all the unique council voters
df = df.select(df["VOTER_NAME"]).distinct()

# Count the rows in voter_df
print("\nThere are %d rows in the df DataFrame.\n" % df.count())
```

There are 27 rows in the df DataFrame.

```
In [45]: #16. Add a ROW_ID
df = df.withColumn('ROW_ID', monotonically_increasing_id())
```




In [46]: *#17. Show the rows with 10 highest IDs in the set*
`df.orderBy(df.ROW_ID.desc()).show(10)`

```
+-----+-----+
| VOTER_NAME | ROW_ID |
+-----+-----+
| Lee Kleinman | 1709396983808 |
| Erik Wilson | 1700807049216 |
| Carolyn King Arnold | 1632087572480 |
| Rickey D. Callahan | 1597727834112 |
| Monica R. Alonzo | 1382979469312 |
| Lee M. Kleinman | 1228360646656 |
| Jennifer S. Gates | 1194000908288 |
| Philip T. Kingston | 1185410973696 |
| Dwaine R. Caraway | 1142461300736 |
| Rickey D. Callahan | 1125281431553 |
+-----+-----+
only showing top 10 rows
```

IDs with different partitions

In [47]: *# Mở rộng*

In [48]: *# Print the number of partitions in each DataFrame*
`print("\nThere are %d partitions in the df DataFrame.\n" % df.rdd.getNumPartitions)`

There are 200 partitions in the df DataFrame.

- Make sure to store the result of `.rdd.max()[0]` in the variable.
- `monotonically_increasing_id()` returns an integer. You can modify that value in-line.
- Make sure to show both Data Frames.

In [49]: *# Determine the highest ROW_ID and save it in previous_max_ID*
`previous_max_ID = df.select('ROW_ID').rdd.max()[0]`

Add a ROW_ID column to df_april starting at the desired value
`voter_df_april = df.withColumn('ROW_ID',
monotonically_increasing_id() + previous_max_ID)`



```
In [50]: # Show the ROW_ID from both DataFrames and compare
df.select('ROW_ID').show(5)
voter_df_april.select('ROW_ID').show(5)
```

```
+-----+
|      ROW_ID|
+-----+
|  8589934592|
|  34359738368|
|  42949672960|
|  51539607552|
| 103079215104|
+-----+
only showing top 5 rows
```

```
+-----+
|      ROW_ID|
+-----+
|1717986918400|
|1743756722176|
|1752346656768|
|1760936591360|
|1812476198912|
+-----+
only showing top 5 rows
```