

# Loading natural language text

INTRODUCTION TO SPARK SQL IN PYTHON



**Mark Plutowski**  
Data Scientist

# The dataset

**The Project Gutenberg eBook of The Adventures of Sherlock Holmes,**

by Sir Arthur Conan Doyle.

Available from [gutenberg.org](https://www.gutenberg.org)

# Loading text

```
df = spark.read.text('sherlock.txt')
```

```
print(df.first())
```

```
Row(value='The Project Gutenberg EBook of The Adventures of Sherlock Holmes')
```

```
print(df.count())
```

```
5500
```

# Loading parquet

```
df1 = spark.read.load('sherlock.parquet')
```

# Loaded text

```
df1.show(15, truncate=False)
```

```
+-----+
|value                                     |
+-----+
|The Project Gutenberg EBook of The Adventures of Sherlock Holmes |
|by Sir Arthur Conan Doyle                                         |
|(#15 in our series by Sir Arthur Conan Doyle)                   |
|                                                                    |
|Copyright laws are changing all over the world. Be sure to check the|
|copyright laws for your country before downloading or redistributing|
|this or any other Project Gutenberg eBook.                       |
|                                                                    |
|This header should be the first thing seen when viewing this Project|
|Gutenberg file. Please do not remove it. Do not change or edit the|
|header without written permission.                                |
|                                                                    |
|Please read the "legal small print," and other information about the|
|eBook and Project Gutenberg at the bottom of this file. Included is|
|important information about your specific rights and restrictions in|
+-----+
```

# Lower case operation

```
df = df1.select(lower(col('value')))  
  
print(df.first())
```

```
Row(lower(value)=  
      'the project gutenber ebook of the adventures of sherlock holmes')
```

```
df.columns
```

```
['lower(value)']
```

# Alias operation

```
df = df1.select(lower(col('value')).alias('v'))
```

```
df.columns
```

```
['v']
```

# Replacing text

```
df = df1.select(regex_replace('value', 'Mr\.', 'Mr').alias('v'))
```

"Mr. Holmes." ==> "Mr Holmes."

```
df = df1.select(regex_replace('value', 'don\'t', 'do not').alias('v'))
```

"don't know." ==> "do not know."



# Tokenizing text

```
df = df2.select(split('v', '[ ]').alias('words'))  
df.show(truncate=False)
```

# Tokenizing text – output

```
+-----+
|words|
+-----+
|[the, project, gutenber, ebook, of, the, adventures, of, sherlock, holmes]|
|[by, sir, arthur, conan, doyle]|
|[ (#15, in, our, series, by, sir, arthur, conan, doyle)]|
|[]|
.
.
.
|[please, read, the, "legal, small, print,", and, other, information, about, the]|
.
.
.
|[**welcome, to, the, world, of, free, plain, vanilla, electronic, texts**]|
+-----+
```

# Split characters are discarded

```
punctuation = "_|.\?\\!\\",\\'\\[\\]\\*()"  
df3 = df2.select(split('v', '[ %s]' % punctuation).alias('words'))
```

```
df3.show(truncate=False)
```

# Split characters are discarded – output

```
+-----+
|words|
+-----+
|[the, project, gutenber, ebook, of, the, adventures, of, sherlock, holmes]|
|[by, sir, arthur, conan, doyle]|
|[, #15, in, our, series, by, sir, arthur, conan, doyle, ]|
|[]|
.|
.|

|[please, read, the, , legal, small, print, , , and, other, information, about, the]|
.|
.|
.|

|[, , welcome, to, the, world, of, free, plain, vanilla, electronic, texts, , ]|
++-----+
```

# Exploding an array

```
df4 = df3.select(explode('words').alias('word'))  
df4.show()
```

# Exploding an array – output

```
+-----+  
|      word|  
+-----+  
|       the|  
|  project|  
| gutenbergl|  
|      ebook|  
|         of|  
|       the|  
|adventures|  
|         of|  
|  sherlock|  
|   holmes|  
|        by|  
|       sir|  
|   arthur|  
|    conan|  
|    doyle|  
+-----+
```

# Explode increases row count

```
print(df3.count())
```

```
5500
```

```
print(df4.count())
```

```
131404
```

# Removing empty rows

```
print(df.count())
```

```
131404
```

```
nonblank_df = df.where(length('word') > 0)
```

```
print(nonblank_df.count())
```

```
107320
```



# Adding a row id column

```
df2 = df.select('word', monotonically_increasing_id().alias('id'))  
  
df2.show()
```

# Adding a row id column – output

```
+-----+-----+
|      word| id|
+-----+-----+
|      the|  0|
|  project|  1|
| gutenbergl  2|
|      ebook|  3|
|         of|  4|
|      the|  5|
|adventures|  6|
|         of|  7|
|  sherlock|  8|
|    holmes|  9|
|         by| 10|
|      sir| 11|
|   arthur| 12|
|    conan| 13|
|    doyle| 14|
|     #15| 15|
+-----+-----+
```

# Partitioning the data

```
df2 = df.withColumn('title', when(df.id < 25000, 'Preface')  
                           .when(df.id < 50000, 'Chapter 1')  
                           .when(df.id < 75000, 'Chapter 2')  
                           .otherwise('Chapter 3'))
```

```
df2 = df2.withColumn('part', when(df2.id < 25000, 0)  
                           .when(df2.id < 50000, 1)  
                           .when(df2.id < 75000, 2)  
                           .otherwise(3))  
df2.show()
```

# Partitioning the data – output

```
+-----+----+-----+-----+
|word      |id |title          |part|
+-----+----+-----+-----+
|the       |0  |Preface        |0    |
|project   |1  |Preface        |0    |
|guttenberg|2  |Preface        |0    |
|ebook     |3  |Preface        |0    |
|of        |4  |Preface        |0    |
|the       |5  |Preface        |0    |
|adventures|6  |Preface        |0    |
|of        |7  |Preface        |0    |
|sherlock  |8  |Preface        |0    |
|holmes    |9  |Preface        |0    |
```

# Repartitioning on a column

```
df2 = df.repartition(4, 'part')
```

```
print(df2.rdd.getNumPartitions())
```

```
4
```

# Reading pre-partitioned text

```
$ ls sherlock_parts
```

```
sherlock_part0.txt  
sherlock_part1.txt  
sherlock_part2.txt  
sherlock_part3.txt  
sherlock_part4.txt  
sherlock_part5.txt  
sherlock_part6.txt  
sherlock_part7.txt  
sherlock_part8.txt  
sherlock_part9.txt  
sherlock_part10.txt  
sherlock_part11.txt  
sherlock_part12.txt  
sherlock_part13.txt
```

# Reading pre-partitioned text

```
df_parts = spark.read.text('sherlock_parts')
```

# Let's practice!

INTRODUCTION TO SPARK SQL IN PYTHON



# Moving window analysis

INTRODUCTION TO SPARK SQL IN PYTHON



**Mark Plutowski**  
Data Scientist

# The raw text

ADVENTURE I. A SCANDAL IN BOHEMIA

I.

To Sherlock Holmes she is always the woman. I have seldom heard him mention her under any other name. In his eyes she eclipses and predominates the whole of her sex. It was not that he felt any emotion akin to love for Irene Adler. All emotions, and that one particularly, were abhorrent to his cold, precise but admirably balanced mind. He was, I take it, the most perfect reasoning and observing machine that the world has seen, but as a lover he would have placed himself in a false position. He never spoke of the softer passions, save with a gibe and a sneer. They were admirable things for the observer--excellent for drawing the veil from men's motives and actions. But for the trained reasoner to admit such intrusions into his own delicate and finely adjusted temperament was to introduce a distracting factor which might throw a doubt upon all his mental results. Grit in a sensitive instrument, or a crack in one of his own high-power lenses, would not be more disturbing than a strong emotion in a nature such as his. And yet there was but one woman to him, and that woman was the late Irene Adler, of dubious and questionable memory.

# The processed text

```
+-----+-----+
|  word| id|part|
+-----+-----+
| scandal|305|  1|
|      in|306|  1|
| bohemia|307|  1|
|       i|308|  1|
|      to|309|  1|
|sherlock|310|  1|
|  holmes|311|  1|
|      she|312|  1|
|      is|313|  1|
| always|314|  1|
|      the|315|  1|
|  woman|316|  1|
|       i|317|  1|
|   have|318|  1|
| seldom|319|  1|
|  heard|320|  1|
|    him|321|  1|
| mention|322|  1|
|     her|323|  1|
|   under|324|  1|
+-----+-----+
```

# Partitions

```
df.select('part', 'title').distinct().sort('part').show(truncate=False)
```

```
+----+-----+
|part|title          |
+----+-----+
|1    |Sherlock Chapter I  |
|2    |Sherlock Chapter II |
|3    |Sherlock Chapter III|
|4    |Sherlock Chapter IV |
|5    |Sherlock Chapter V  |
|6    |Sherlock Chapter VI |
|7    |Sherlock Chapter VII|
|8    |Sherlock Chapter VIII|
|9    |Sherlock Chapter IX |
|10   |Sherlock Chapter X  |
|11   |Sherlock Chapter XI |
|12   |Sherlock Chapter XII|
+----+-----+
```

id	word
0	the
1	project
2	guttenberg
3	ebook
4	of
5	the
6	adventures
7	of
8	sherlock
9	holmes
10	by
11	sir
12	arthur
13	conan

id	word
0	the
1	project
2	guttenberg
3	of
4	of
5	the
6	adventures
7	of
8	sherlock
9	holmes
10	by
11	sir
12	arthur
13	conan

id	word
0	the
1	project
2	guttenberg
3	ebook
4	of
5	the
6	adventures
7	of
8	sherlock
9	holmes
10	by
11	sir
12	arthur
13	conan

id	word
0	the
1	project
2	gutenberg
3	ebook
4	of
5	the
6	adventures
7	of
8	sherlock
9	holmes
10	by
11	sir
12	arthur
13	conan



id	word
0	the
1	project
2	autenberga
3	ebook
4	of
5	the
6	
7	of
8	sherlock
9	holmes
10	by
11	sir
12	arthur
13	conan

id	word
0	the
1	project
2	guttenberg
3	check
4	of
5	the
6	adventures
7	of
8	sherlock
9	holmes
10	by
11	sir
12	arthur
13	conan

id	word
0	the
1	project
2	guttenberg
3	ebook
4	of
5	the
6	adventures
7	of
8	sherlock
9	Holmes
10	by
11	sir
12	arthur
13	conan

id	word
0	the
1	project
2	gutenberg
3	ebook
4	of
5	the
6	adventures
7	of
8	sherlock
9	holmes
10	by
11	arthur
12	conan
13	doyle

id	word
0	the
1	project
2	guttenberg
3	ebook
4	of
5	the
6	adventures
7	of
8	sherlock
9	holmes
10	by

11	sir
12	arthur
13	conan

# The words are indexed

```
+---+-----+
| id|      word|
+---+-----+
| 0|      the|
| 1|   project|
| 2|  gutenberg|
| 3|      ebook|
| 4|        of|
| 5|      the|
| 6|adventures|
| 7|        of|
| 8|  sherlock|
| 9|   holmes|
|10|        by|
|11|       sir|
|12|   arthur|
|13|    conan|
|14|   doyle|
|15|    #15|
|16|       in|
|17|      our|
|18|   series|
|19|        by|
+---+-----+
```

# A moving window query

```
query = """
    SELECT id, word AS w1,
    LEAD(word,1) OVER(PARTITION BY part ORDER BY id ) AS w2,
    LEAD(word,2) OVER(PARTITION BY part ORDER BY id ) AS w3
    FROM df
    """

spark.sql(query).sort('id').show()
```

```
+---+-----+-----+-----+
| id|      w1|      w2|      w3|
+---+-----+-----+-----+
|  0|    the|  project| gutenberg|
|  1| project| gutenberg|    ebook|
|  2| gutenberg|    ebook|    of|
|  3|    ebook|    of|    the|
|  4|    of|    the| adventures|
|...|.....|.....|.....|
```

# Moving window output

```
+---+-----+-----+-----+
| id|      w1|      w2|      w3|
+---+-----+-----+-----+
|  0|    the| project| gutenbergl
|  1| project| gutenbergl      ebook| |
|  2| gutenbergl      ebook|      of|
|  3|      ebook|      of|      the|
|  4|      of|      the|adventures|
|  5|      the|adventures|      of|
|  6|adventures|      of| sherlock|
|  7|      of| sherlock|      holmes|
|  8| sherlock|      holmes|      by|
|  9|      holmes|      by|      sir|
| 10|      by|      sir|      arthur|
| 11|      sir|      arthur|      conan|
| 12|      arthur|      conan|      doyle|
+---+-----+-----+-----+
```



# LAG window function

```
lag_query = """
    SELECT
    id,
    LAG(word,2) OVER(PARTITION BY part ORDER BY id ) AS w1,
    LAG(word,1) OVER(PARTITION BY part ORDER BY id ) AS w2,
    word AS w3
    FROM df
    ORDER BY id
    """

spark.sql(lag_query).show()
```

# LAG window function – output

```
+---+-----+-----+-----+
| id|      w1|      w2|      w3|
+---+-----+-----+-----+
|  0|    null|    null|    the|
|  1|    null|    the| project|
|  2|    the| project| gutenberg|
|  3| project| gutenberg|    ebook|
|  4| gutenberg|    ebook|    of|
|  5|    ebook|    of|    the|
|  6|    of|    the|adventures|
|  7|    the|adventures|    of|
|  8|adventures|    of| sherlock|
|  9|    of| sherlock|    holmes|
| 10| sherlock|    holmes|    by|
| 11|    holmes|    by|    sir|
| 12|    ...|    ...|    ...|
+---+-----+-----+-----+
```

# Windows stay within partition

```
lag_query = """
    SELECT
    id,
    LAG(word,2) OVER(PARTITION BY part ORDER BY id ) AS w1,
    LAG(word,1) OVER(PARTITION BY part ORDER BY id ) AS w2,
    word AS w3
    FROM df
    WHERE part=2
    """

spark.sql(lag_query).show()
```

# Windows stay within partition – output

```
+-----+-----+-----+-----+
|  id|          w1|          w2|          w3|
+-----+-----+-----+-----+
|8859|      null|      null|    part2|
|8860|      null|    part2| adventure|
|8861|    part2| adventure|        ii|
|8862| adventure|        ii|      the|
|8863|        ii|      the|red-headed|
|8864|      the|red-headed|    league|
|....|.....|.....|.....|
```

# Repartitioning

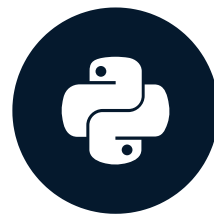
- PARTITION BY
- `repartition()`

# Let's practice!

INTRODUCTION TO SPARK SQL IN PYTHON

# Common word sequences

INTRODUCTION TO SPARK SQL IN PYTHON



**Mark Plutowski**  
Data Scientist



# Training



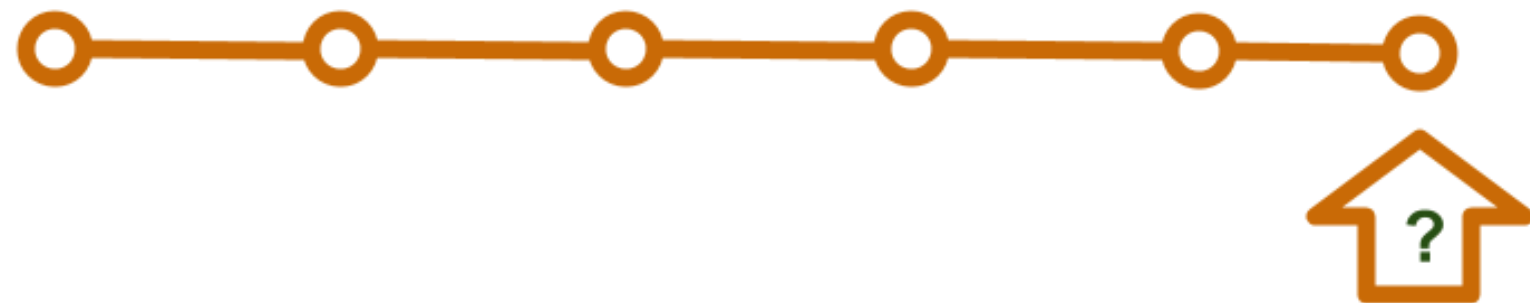


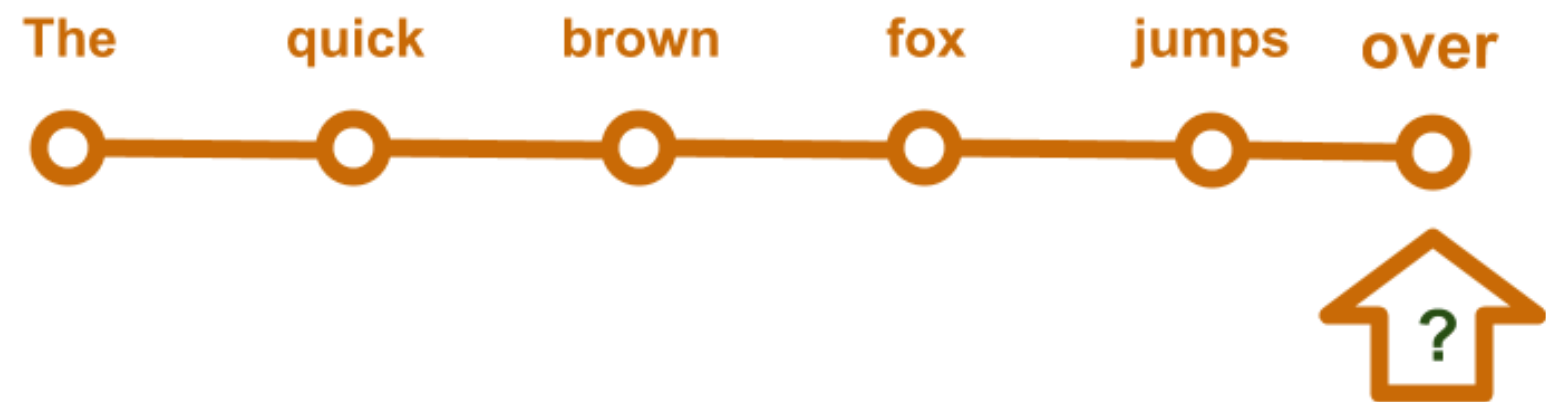
# Predicting

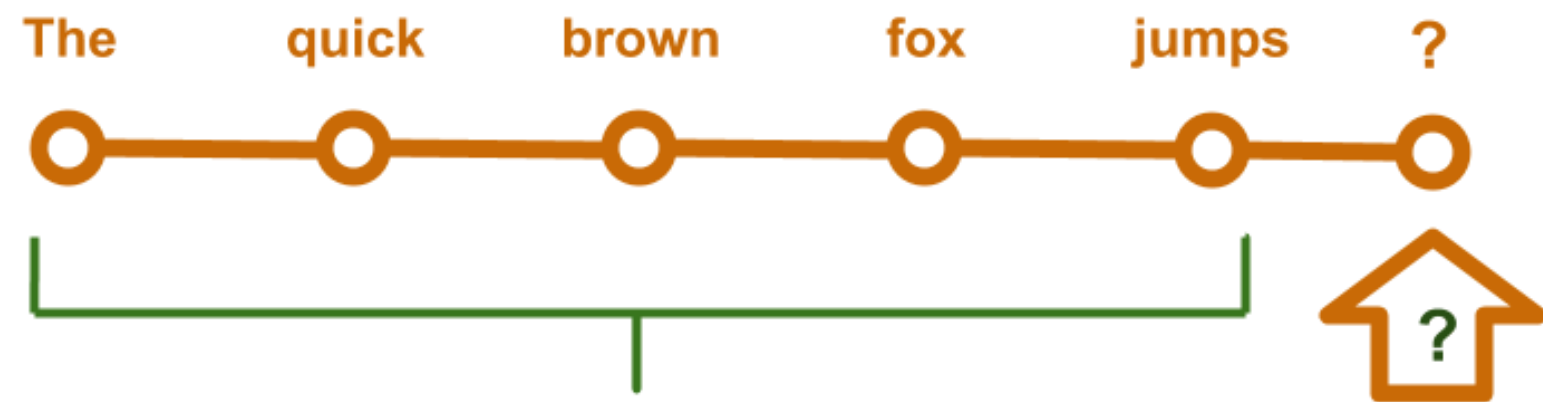


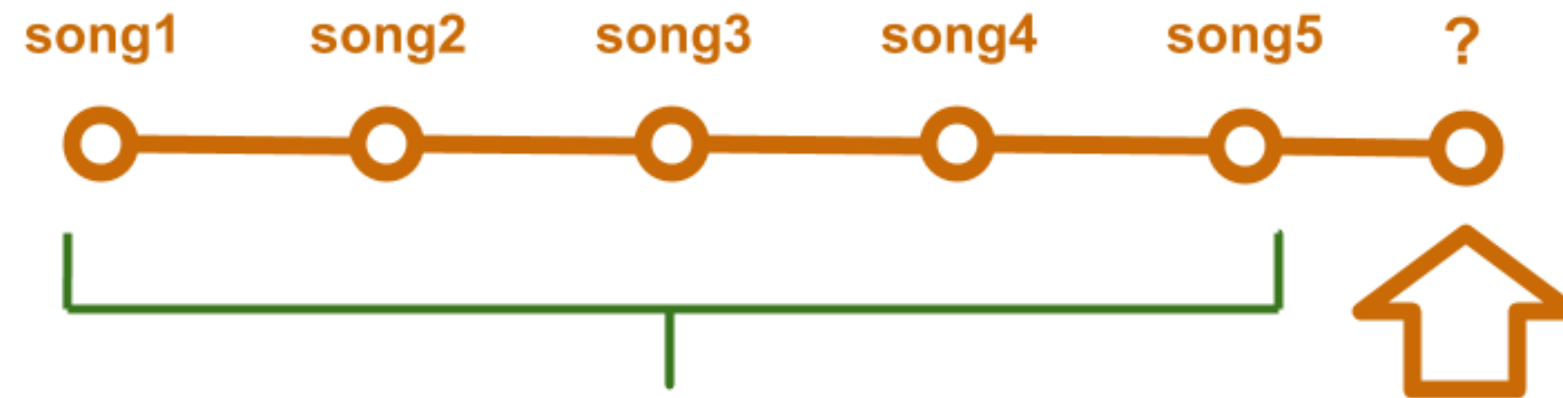
# Endword Prediction

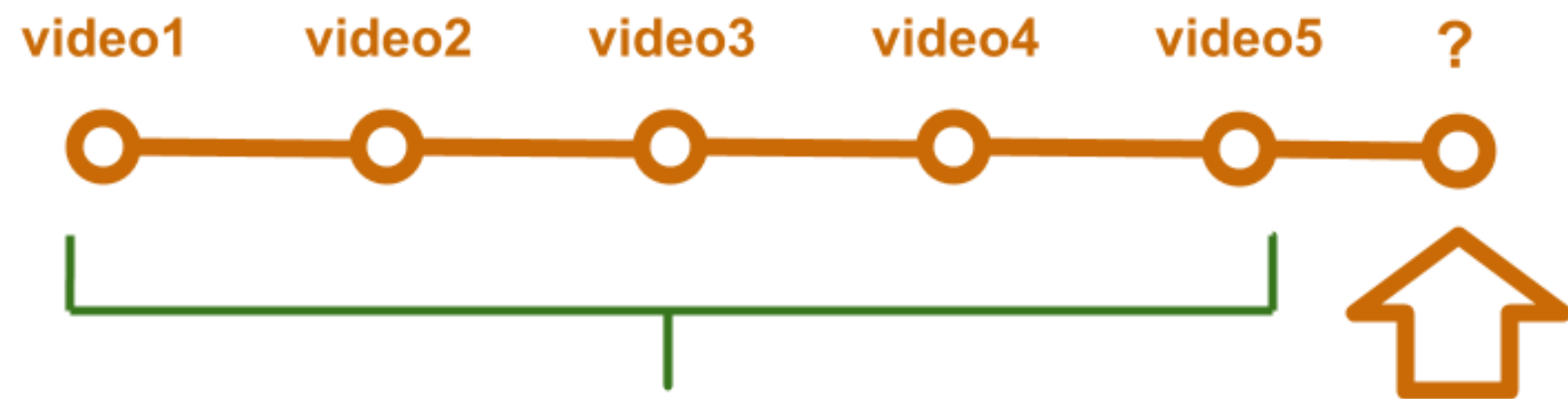
















# Categorical Data

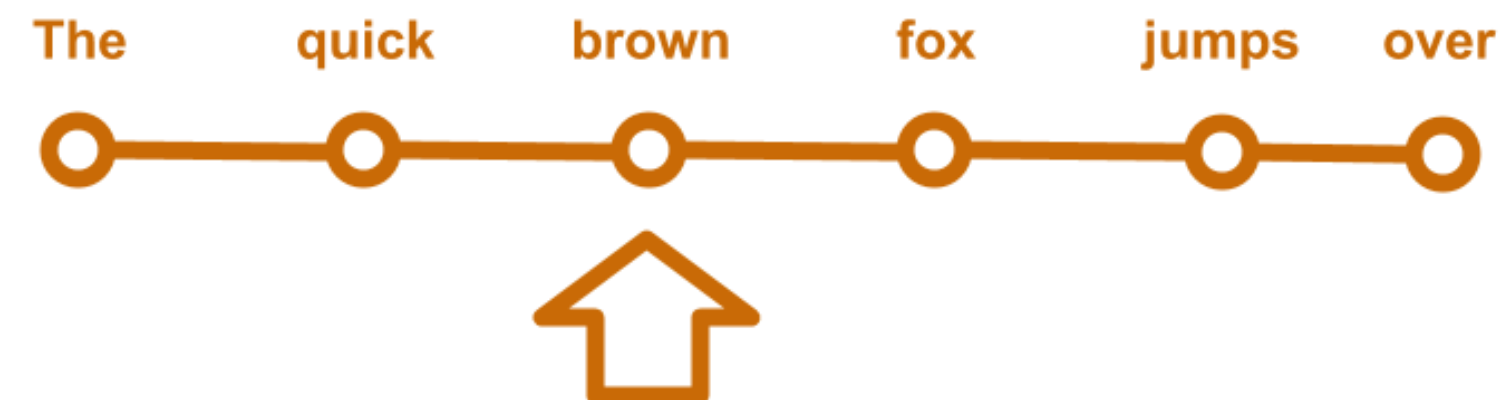
# Categorical vs Ordinal

- **Categorical:** he, hi, she, that, they
- **Ordinal:** 1, 2, 3, 4, 5

# Sequence Analysis



<u>Word</u>	
quick	⇐ preceding row
<b>brown</b>	⇐ current row
fox	⇐ following row



# 3-tuples

```
query3 = """
    SELECT
    id,
    word AS w1,
    LEAD(word,1) OVER(PARTITION BY part ORDER BY id ) AS w2,
    LEAD(word,2) OVER(PARTITION BY part ORDER BY id ) AS w3
    FROM df
    """
```

# A window function SQL as subquery

```
query3agg = """
SELECT w1, w2, w3, COUNT(*) as count FROM (
    SELECT
    word AS w1,
    LEAD(word,1) OVER(PARTITION BY part ORDER BY id ) AS w2,
    LEAD(word,2) OVER(PARTITION BY part ORDER BY id ) AS w3
    FROM df
)
GROUP BY w1, w2, w3
ORDER BY count DESC
"""

spark.sql(query3agg).show()
```

# A window function SQL as subquery – output

```
+-----+-----+-----+-----+
|  w1 |  w2 |  w3 | count |
+-----+-----+-----+-----+
| one | of | the | 49 |
|  i |think| that| 46 |
|  it | is |  a | 46 |
|  it | was |  a | 45 |
| that|  it | was | 38 |
| out | of | the | 35 |
|.....|.....|.....|.....|
```

# Most frequent 3-tuples

```
+-----+-----+-----+-----+
|  w1|   w2|   w3|count|
+-----+-----+-----+-----+
| one|  of| the|   49|
|  i|think|that|   46|
| it|  is|  a|   46|
| it| was|  a|   45|
| that| it| was|   38|
| out|  of| the|   35|
| that|  i| have|   35|
|there| was|  a|   34|
|  i|  do| not|   34|
| that| it|  is|   33|
| that| he| was|   30|
| that| he| had|   30|
| that|  i| was|   28|
+-----+-----+-----+-----+
```



# Another type of aggregation

```
query3agg = """
SELECT w1, w2, w3, length(w1)+length(w2)+length(w3) as length FROM (
    SELECT
    word AS w1,
    LEAD(word,1) OVER(PARTITION BY part ORDER BY id ) AS w2,
    LEAD(word,2) OVER(PARTITION BY part ORDER BY id ) AS w3
    FROM df
    WHERE part <> 0 and part <> 13
)
GROUP BY w1, w2, w3
ORDER BY length DESC
"""

spark.sql(query3agg).show(truncate=False)
```

# Another type of aggregation

```
+-----+-----+-----+-----+
|          w1|          w2|          w3|length|
+-----+-----+-----+-----+
|comfortable-looking|      building|    two-storied|    38|
|      widespread|comfortable-looking|      building|    37|
|   extraordinary|   circumstances|    connected|    35|
|   simple-minded|   nonconformist|    clergyman|    35|
|   particularly|      malignant|   boot-slitting|    34|
|   unsystematic|   sensational|    literature|    33|
|   oppressively|   respectable|   frock-coat|    33|
|     relentless|   keen-witted|   ready-handed|    33|
| travelling-cloak|              and|   close-fitting|    32|
|     ruddy-faced|   white-aproned|     landlord|    32|
| fellow-countryman|      colonel|     lysander|    32|
+-----+-----+-----+-----+
```

# Let's practice

INTRODUCTION TO SPARK SQL IN PYTHON