Ngram Analysis and Filtering Notebook

In this notebook, the `Ngram` data set will be imported, analyzed, filtered, and exported as a new data set.

In [1]:
```
# Import the engram data and convert it to a pandas data frame
ngram_df = spark.read.csv("/user/hadoop/eng_1M_1gram/eng_1M_1gram.csv", header=True)
```

```
Starting Spark application
```

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
1	application_1631753792564_0003	pyspark	idle	Link	Link	✔

```
SparkSession available as 'spark'.
```

In [2]:
```
# Confirm import success
ngram_df.show(5)
```

```
+---------+----+---------+-----+-----+
|    token|year|frequency|pages|books|
+---------+----+---------+-----+-----+
|inGermany|1927|        2|    2|    2|
|inGermany|1929|        1|    1|    1|
|inGermany|1930|        1|    1|    1|
|inGermany|1933|        1|    1|    1|
|inGermany|1934|        1|    1|    1|
+---------+----+---------+-----+-----+
only showing top 5 rows
```

The data set appears to have been imported successfully. I will now need to see the schema of this data set.

In [3]:
```
# Get ngram data frame schema
ngram_df.printSchema()
```

```
root
 |-- token: string (nullable = true)
 |-- year: string (nullable = true)
 |-- frequency: string (nullable = true)
 |-- pages: string (nullable = true)
 |-- books: string (nullable = true)
```

The ngram data frame has 5 columns, all of which are strings. These columns are:

- `token` : the word of interest
- `year` : the year of interest
- `frequency` : total number of times the word was used in the given year
- `pages` : the number of pages containing the word of interest in the year of interest
- `books` : the number of books containing the word of interest in the year of interest

I will now check to see how many rows are in the data frame.

In [4]:
```
# Count the number of rows in the data set.
ngram_df.count()
```

```
261823225
```

There are 261,823,225 rows of data, or a little more than a quarter billion rows. We want to filter this so that we are only looking at rows corresponding to the word `data`. This will be done using a SQL function built into spark.

In [6]:
```
# Create a SQL view of the ngram data frame
ngram_df.createOrReplaceTempView("ngram_view")
# Run the SQL query and confirm that it ran successfully.
spark.sql(" \
          SELECT * \
          FROM ngram_view \
          WHERE token = 'data'\
          ").show(10)
```

```
+-----+----+---------+-----+-----+
|token|year|frequency|pages|books|
+-----+----+---------+-----+-----+
| data|1584|       16|   14|    1|
| data|1614|        3|    2|    1|
| data|1627|        1|    1|    1|
| data|1631|       22|   18|    1|
| data|1637|        1|    1|    1|
| data|1638|        2|    2|    1|
| data|1640|        1|    1|    1|
| data|1642|        1|    1|    1|
| data|1644|        4|    4|    1|
| data|1647|        1|    1|    1|
+-----+----+---------+-----+-----+
only showing top 10 rows
```

The SQL querry appears to have worked correctly. The data has all been filtered. I will now put this filtered data into a new data frame.

In [7]:
```
# Set a new data frame, "data_df," equal to the SQL search
data_df = spark.sql(" \
          SELECT * \
          FROM ngram_view \
          WHERE token = 'data'\
          ")
# Confirm that the data frame was created successfully.
data_df.show(5)
```

```
+-----+----+---------+-----+-----+
|token|year|frequency|pages|books|
+-----+----+---------+-----+-----+
| data|1584|       16|   14|    1|
| data|1614|        3|    2|    1|
| data|1627|        1|    1|    1|
| data|1631|       22|   18|    1|
| data|1637|        1|    1|    1|
+-----+----+---------+-----+-----+
only showing top 5 rows
```

The data frame appears to have been created successfully. I will now check how many rows are in this new data frame.

In [8]:
```
# Count rows in the data frame
data_df.count()
```

```
316
```

There are 316 rows in this new data frame, meaning that the word `data` has been found in books published in 316 different years.

We now need to export this data frame for further analysis.

In [9]:

```
# export the data frame
data_df.write.csv('/user/hadoop/eng_data_1gram/eng_1M_1gram_data_token.csv', header=Tru
```