PROJECT REPORT (Part 2) - *Rohit Suhas Gurjar*

Table of Contents

Problem Description:

The later part of this project emphasizes on using Spark. The analysis is performed within Jupyter Hub environment through AWS Big Data technologies - AWS EMR. The Amazon reviews dataset is explored using Spark Dataframe API's. The aim of the project is to utilize spark concepts like LDA to get potential insights from user reviews by using topic modeling. Other functionalities such as aggregations, pivot columns are also used.

Creating Dataframe by reading data from a parquet file:

```
# Load Data Set
df = spark.read\
          .option("header", "true")\
          .option("inferSchema", "true")\
          .option("basePath", "hdfs:///hive/amazon-reviews-pds/parquet/")\
          .parquet("hdfs:///hive/amazon-reviews-pds/parquet/*")

df.printSchema()
```

```
root
 |-- marketplace: string (nullable = true)
 |-- customer_id: string (nullable = true)
 |-- review_id: string (nullable = true)
 |-- product_id: string (nullable = true)
 |-- product_parent: string (nullable = true)
 |-- product_title: string (nullable = true)
 |-- star_rating: integer (nullable = true)
 |-- helpful_votes: integer (nullable = true)
 |-- total_votes: integer (nullable = true)
 |-- vine: string (nullable = true)
 |-- verified_purchase: string (nullable = true)
 |-- review_headline: string (nullable = true)
 |-- review_body: string (nullable = true)
 |-- review_date: date (nullable = true)
 |-- year: integer (nullable = true)
 |-- product_category: string (nullable = true)
```

Excluding data before 2005

```
df_limited = df.filter(F.col("year")>2004)
```

Filtering data by excluding multiple reviews for same product by same users

```
from pyspark.sql.window import *
from pyspark.sql.functions import row_number
df_limited_1=df_limited.withColumn("rownum",row_number().over(Window.partitionBy
("customer_id","product_id").orderBy("customer_id","product_id")))

reviews_fil = df_limited_1.rownum.isin(1)
reviews_filter=df_limited_1.where(reviews_fil)
reviews_filter.persist()
```

3. Exploratory Data Analysis - Books and Digital E-books

Explore the dataset and provide analysis by product-category and year

Number of reviews

```
reviews_filter.groupby("year","product_category").agg(F.countDistinct("review_id
").alias('Number of Reviews')).show()
```

```
+----+--------------------+-----------------+
|year|    product_category|Number of Reviews|
+----+--------------------+-----------------+
|2014|               Books|          3540846|
|2010|Digital_Ebook_Pur...|           102513|
|2015|               Books|          2860745|
|2013|            Wireless|          1767132|
|2014|         Mobile_Apps|          1728281|
|2013|Digital_Video_Dow...|           722509|
|2011|Digital_Ebook_Pur...|           350133|
|2008|               Books|           827677|
|2012|         Mobile_Apps|           711483|
|2008|            Wireless|            63670|
|2011|               Books|          1303122|
|2015|           Video_DVD|           953033|
|2007|Digital_Video_Dow...|             2597|
|2012|           Video_DVD|           461957|
|2011|Digital_Video_Dow...|            20894|
|2009|Digital_Video_Dow...|             3262|
|2015|            Wireless|          3000787|
|2010|Digital_Video_Dow...|             6090|
|2013|               Books|          2965953|
|2009|                  PC|           130074|
+----+--------------------+-----------------+
only showing top 20 rows
```

Number of Users

```
reviews_filter.groupby("year","product_category").agg(F.countDistinct("customer_
id").alias('Number of Users')).show()
```

```
+----+--------------------+---------------+
|year|    product_category|Number of Users|
+----+--------------------+---------------+
|2014|               Books|        1859226|
|2010|Digital_Ebook_Pur...|          61196|
|2013|            Wireless|        1193462|
|2015|               Books|        1548552|
|2014|         Mobile_Apps|         988658|
|2013|Digital_Video_Dow...|         445955|
|2011|Digital_Ebook_Pur...|         183969|
|2008|               Books|         459240|
|2012|         Mobile_Apps|         419052|
|2008|            Wireless|          55686|
|2011|               Books|         752549|
|2015|           Video_DVD|         461602|
|2007|Digital_Video_Dow...|           2027|
|2012|           Video_DVD|         273944|
|2011|Digital_Video_Dow...|          14722|
|2009|Digital_Video_Dow...|           2407|
|2015|            Wireless|        1982264|
|2010|Digital_Video_Dow...|           4483|
|2013|               Books|        1620946|
|2009|                  PC|         107704|
+----+--------------------+---------------+
only showing top 20 rows
```

Average and Median review stars

```
from pyspark.sql.functions import *
reviews_filter.groupby("year","product_category")
.agg(round(F.avg("star_rating"),3).alias('Avg_Rating'),

F.expr('percentile_approx(star_rating,0.5)').alias('Median_Rating')).show()
```

```
+----+--------------------+----------+-------------+
|year|    product_category|Avg_Rating|Median_Rating|
+----+--------------------+----------+-------------+
|2014|               Books|     4.473|            5|
|2010|Digital_Ebook_Pur...|     3.822|            4|
|2015|               Books|     4.497|            5|
|2013|            Wireless|      3.82|            4|
|2014|         Mobile_Apps|     3.969|            5|
|2013|Digital_Video_Dow...|     4.208|            5|
|2011|Digital_Ebook_Pur...|     4.056|            5|
|2008|               Books|     4.233|            5|
|2012|         Mobile_Apps|     3.995|            5|
|2008|            Wireless|      3.77|            4|
|2011|               Books|     4.251|            5|
|2015|           Video_DVD|      4.53|            5|
|2007|Digital_Video_Dow...|       3.6|            4|
|2012|           Video_DVD|     4.218|            5|
|2011|Digital_Video_Dow...|     3.778|            5|
|2009|Digital_Video_Dow...|       3.7|            4|
|2015|            Wireless|     3.985|            5|
|2010|Digital_Video_Dow...|     3.757|            4|
|2013|               Books|     4.412|            5|
|2009|                  PC|     3.967|            5|
+----+--------------------+----------+-------------+
only showing top 20 rows
```

Percentiles of length of the review. Use the following percentiles: [0.1, 0.25, 0.5, 0.75, 0.9, 0.95]

```
from pyspark.sql.functions import length,count, mean, stddev_pop, min, max
temp=reviews_filter.withColumn('length', length(df.review_body))
temp_1=temp.groupby("year","product_category").agg(F.avg("length").alias('Avg of
Reviews'))
colName = "Avg of Reviews"
quantileProbability = [0.1, 0.25, 0.5, 0.75, 0.9, 0.95]
relError = 0.05
temp_1.stat.approxQuantile("Avg of Reviews", quantileProbability, relError)
```

```
[202.3649030770691, 338.2845999711891, 586.5676289328576, 817.3758372362391,
961.9744705407476, 2207.5789473684213]
```

Percentiles for number of reviews per product. For example, 10% of books got 5 or less reviews. Use the following percentiles: [0.1, 0.25, 0.5, 0.75, 0.9, 0.95]

```
df_Q5=reviews_filter.groupby("year","product_id","product_category").agg(F.count
Distinct("review_id").alias('Number of Reviews'))
colName = "Number of Reviews"
quantileProbability = [0.1, 0.25, 0.5, 0.75, 0.9, 0.95]
relError = 0.05
df_Q5.stat.approxQuantile("Number of Reviews", quantileProbability, relError)
```

```
[1.0, 1.0, 2.0, 4.0, 4273.0, 31128.0]
```

Identify week number (each year has 52 weeks) for each year and product category with most positive reviews (4 and 5 star)

```
from pyspark.sql.functions import weekofyear
S = reviews_filter.star_rating.isin(4)
Q = reviews_filter.star_rating.isin(5)
df_Q6=reviews_filter.select("product_category","year","review_date").withColumn(
"week_number",weekofyear("review_date")).where(S | Q)
df_q6 = df_Q6.groupby("product_category","year","week_number")
        .agg(F.countDistinct("week_number").alias("count"))
df_q6.drop('count').show()
```

```
+--------------------+----+-----------+
|    product_category|year|week_number|
+--------------------+----+-----------+
|           Video_DVD|2015|         12|
|               Books|2011|         36|
|Digital_Ebook_Pur...|2015|         16|
|           Video_DVD|2011|         37|
|Digital_Ebook_Pur...|2014|         11|
|               Books|2008|         48|
|               Books|2007|         37|
|               Books|2004|         18|
|           Video_DVD|2000|         46|
|           Video_DVD|2000|          9|
|                  PC|2003|         14|
|               Books|1996|          6|
|                  PC|2003|         15|
|            Wireless|1999|         47|
|           Video_DVD|1998|         29|
|Digital_Ebook_Pur...|2015|         11|
|                  PC|2012|         24|
|               Books|2014|          6|
|               Books|2010|         12|
|Digital_Ebook_Pur...|2013|         49|
+--------------------+----+-----------+
only showing top 20 rows
```

4. Pivot Functionality

Provide detailed analysis of "Digital eBook Purchase" versus "Books"

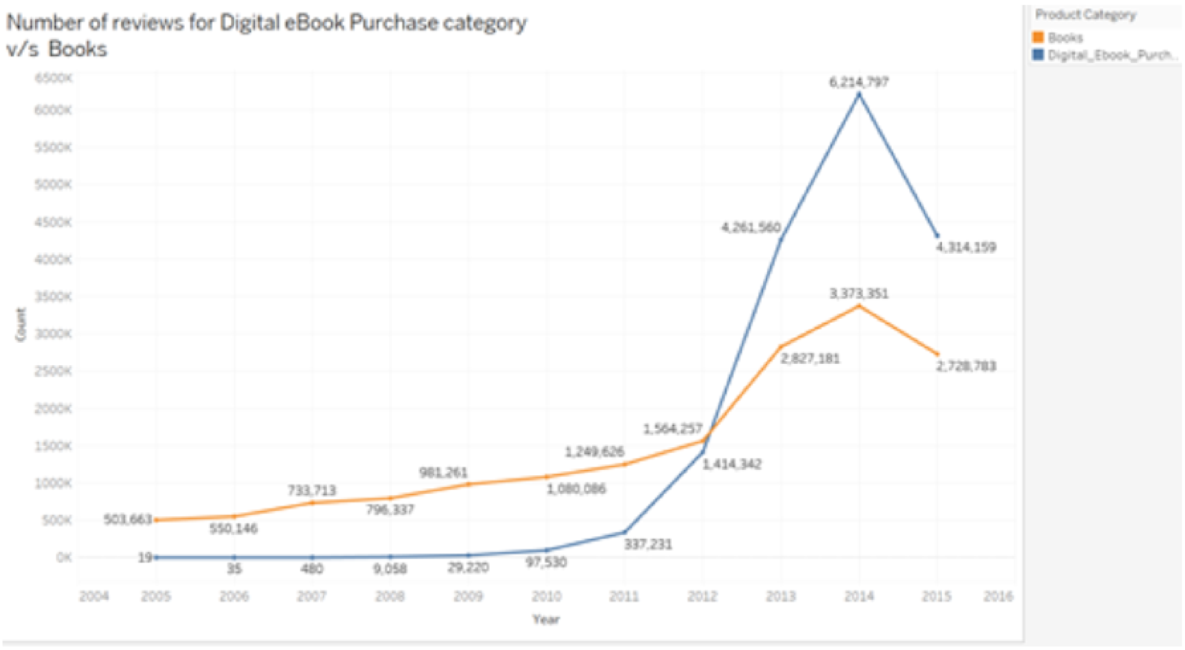
Using Spark Pivot functionality, produce DataFrame with following columns

```
pivot_Q2=['Digital_Ebook_Purchase','Books']
pivot_q2=reviews_filter.groupBy("year",F.month(F.col("review_date"))).pivot("pro
duct_category",pivot_Q2)\
 .agg((F.count("review_id")).alias("Number of reviews"),
 F.round(F.mean("star_rating"),3).alias("Average star
rating")).sort("year","month(review_date)").show()
```

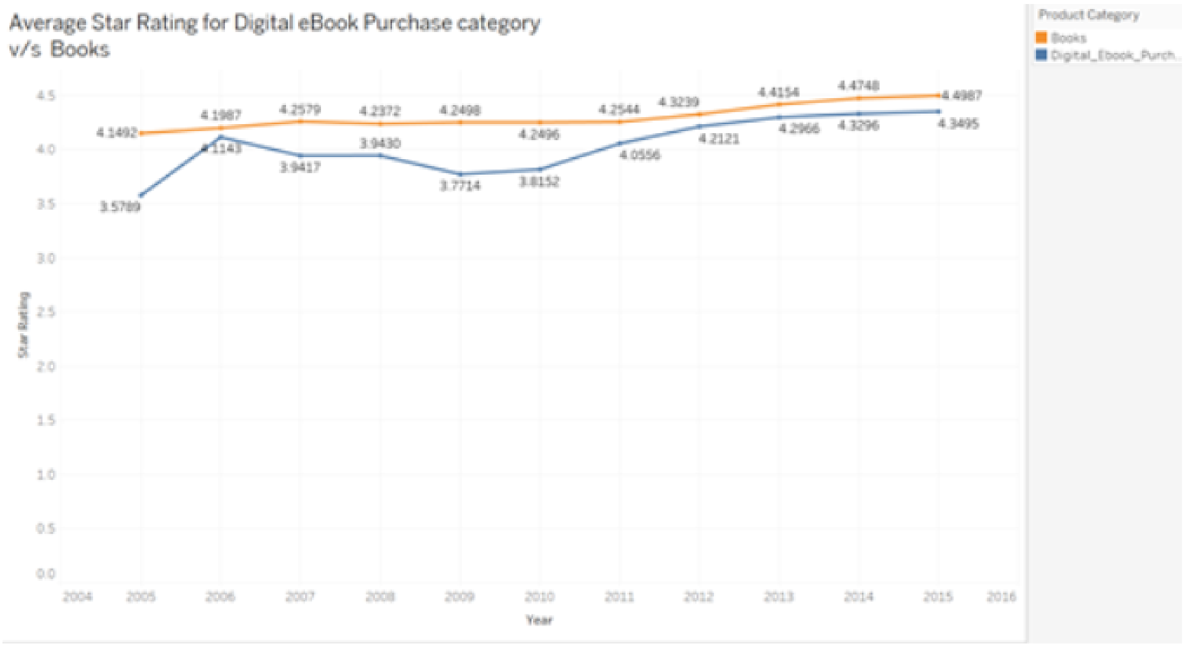
```
+----+------------------+----------------------------------------+--------------
----------------------------+-----------------------+-------------------------+
|year|month(review_date)|Digital_Ebook_Purchase_Number of
reviews|Digital_Ebook_Purchase_Average star rating|Books_Number of
reviews|Books_Average star rating|
+----+------------------+----------------------------------------+--------------
----------------------------+-----------------------+-------------------------+
|2005|                 1|                                       1|
                         5.0|                  40428|                    4.121|
|2005|                 2|                                    null|
                        null|                  33723|                    4.125|
|2005|                 3|                                       2|
                         4.5|                  38877|                    4.122|
|2005|                 4|                                       1|
                         5.0|                  36885|                    4.132|
|2005|                 5|                                       1|
                         1.0|                  36878|                    4.132|
|2005|                 6|                                    null|
                        null|                  36610|                    4.115|
|2005|                 7|                                       3|
                         2.0|                  45946|                    4.128|
|2005|                 8|                                       3|
                       2.667|                  58925|                    4.186|
|2005|                 9|                                       2|
                         4.0|                  58128|                    4.203|
|2005|                10|                                       4|
                         4.0|                  51211|                     4.18|
|2005|                11|                                       1|
                         5.0|                  40885|                    4.151|
|2005|                12|                                       1|
                         5.0|                  42526|                    4.126|
|2006|                 1|                                       8|
                       3.375|                  51998|                    4.135|
|2006|                 2|                                       5|
                         4.6|                  54416|                    4.203|
|2006|                 3|                                    null|
                        null|                  66895|                    4.233|
|2006|                 4|                                    null|
                        null|                  27677|                    4.132|
|2006|                 5|                                       1|
                         5.0|                  45005|                     4.18|
|2006|                 6|                                       5|
                         4.2|                  48049|                    4.184|
|2006|                 7|                                        1|
                        4.0|                    55793|                     4.2|
|2006|                 8|                                        9|
                      4.444|                    54419|                   4.213|
+----+-----------------+----------------------------------------+-------------
----------------------------+----------------------+-------------------------+
only showing top 20 rows
```

Visualizations

Produce two graphs to demonstrate aggregations performed above:



Number of Reviews for Digital E-book Purchase is less than that for Books till 2012 but then increases substantially than Books till 2014.



The Average star rating for Books is consistently high over the years as compared to Digital E-books.

5. Comparison of Products - Books

Identify similar products (books) in both categories. Use "product_title" to match products. To account for potential differences in naming of products, compare titles after stripping spaces and converting to lower case.

```
digital_filter=['Digital_Ebook_Purchase']
digital=reviews_filter.groupBy("product_title","product_category")\
        .agg((F.count("review_id")).alias("Digital Ebook number of reviews"),

 F.round(F.mean("star_rating"),3).alias("Digital_book_Average_star_rating"))
        .filter(F.col("product_category").isin(digital_filter))

trim_digital=digital.select(F.lower(F.trim(F.col("product_title"))).alias("produ
ct_title"              ),F.col("Digital Ebook number of reviews") \
             ,F.col("Digital_book_Average_star_rating"))
```

```
book_filter=['Books']
book=reviews_filter.groupBy("product_title","product_category")\
     .agg((F.count("review_id")).alias("book_number_of_reviews"),
     F.round(F.mean("star_rating"),3).alias("book_Average_star_rating"))
     .filter(F.col("product_category").isin(book_filter))

trim_book=book.select(F.lower(F.trim(F.col("product_title"))).alias("product_tit
le")
          ,F.col("book_number_of_reviews"), F.col("book_Average_star_rating"))
```

```
joinExpression = trim_book["product_title"] == trim_digital["product_title"]
joinType = "inner"
combined=trim_book.join(trim_digital, joinExpression, joinType)
combined.show()
```

```
----------------------------+
|       product_title|book_number_of_reviews|book_Average_star_rating|
product_title|Digital Ebook number of reviews|Digital_book_Average_star_rating|
+--------------------+----------------------+------------------------+----------
----------+-------------------------------+--------------------------------+
|"rays of light": ...|                     2|                     5.0|"rays of
light": ...|                             1|                             5.0|
|"the siege of khe...|                    19|                   4.316|"the siege
of khe...|                           156|                           3.327|
|          'dem bon'z|                     4|                     5.0|
'dem bon'z|                              2|                             5.0|
|   0400 roswell time|                     1|                     5.0|   0400
roswell time|                            6|                           3.667|
|10 smart things g...|                    19|                   4.789|10 smart
things g...|                             6|                           4.833|
|10 smart things g...|                     1|                     5.0|10 smart
things g...|                             6|                           4.833|
|100 prayers for y...|                    11|                     5.0|100
prayers for y...|                         7|
5.0|
|13 cent killers: ...|                    37|                   2.811|13 cent
killers: ...|                            15|                           3.933|
|25 essentials: te...|                    41|                   4.439|25
essentials: te...|                          1|
5.0|
|30 before 30: tra...|                     2|                     3.5|30 before
30: tra...|                            33|                            4.97|
|300 hard word sea...|                     2|                     4.5|300 hard
word sea...|                            7|                             1.0|
|42 rules to incre...|                     1|                     5.0|42 rules
to incre...|                            2|                             5.0|
|50 american heroe...|                     2|                     5.0|50
american heroe...|                          3|
4.0|
|50 successful har...|                    49|                   4.347|50
successful har...|                          3|
4.667|
|52 prepper projec...|                    30|                     3.9|52 prepper
projec...|                           2|                             4.5|
|73 north: the bat...|                     6|                     5.0|73 north:
the bat...|                            1|                            5.0|
|<i>change</i> the...|                     8|                    4.75|
<i>change</i> the...|                             2|
  4.5|
|      a changed life|                     5|                     4.2|      a
changed life|                            36|                           4.222|
|a chip off the ol...|                     1|                     5.0|a chip off
the ol...|                             1|                            5.0|
|a closer look at ...|                     1|                     5.0|a closer
look at ...|                            1|                             1.0|
+--------------------+----------------------+------------------------+----------
----------+--------------------------------+-------------------------------+
only showing top 20 rows
```

```
Rating=F.col("book_Average_star_rating")>4
combined.where(Rating).count()
```

```
276592
```

```
rating=F.col("Digital_book_Average_star_rating")>4
combined.where(rating).count()
```

```
245532
```

Printed books has more number of ratings greater than 4 as compared to that of digital books.

6. Topic Modeling using LDA

Using provided LDA starter notebook, perform LDA topic modeling for the reviews in Digital_Ebook_Purchase and Books categories. Consider reviews for the January of 2015 only.

```
from pyspark.mllib.clustering import LDA, LDAModel
from pyspark.mllib.linalg import Vectors
from pyspark.ml.feature import CountVectorizer, IDF,RegexTokenizer, Tokenizer
from pyspark.sql.types import ArrayType
from pyspark.sql.types import StringType
from pyspark.sql.types import *
from pyspark.sql.functions import udf
from pyspark.sql.functions import struct
import re
from pyspark.ml.feature import StopWordsRemover
from pyspark.ml.clustering import LDA
from pyspark.ml.feature import CountVectorizer
```

Topic Modeling for star rating greater than 3

Filtering dataset for the 2015 data.

```
df_ml =
reviews_filter.filter((F.col("product_category")=="Digital_Ebook_Purchase") |
(F.col("product_category")=="Books") \
                   & (F.col("year")==2015) \
                   & (F.col("review_date")<'2015-02-01')
                   & (F.col("star_rating")>3))
```

Concatenating text and headline of reviews and creating an id column.

```
from pyspark.sql.functions import monotonically_increasing_id, concat
df_lda = df_ml.withColumn('review_text',
                       F.concat(F.col('review_headline'),F.lit(' '),
F.col('review_body')))
corpus =df_lda.select('review_text')

# This will return a new DF with all the columns + id
corpus_df = corpus.withColumn("id", F.monotonically_increasing_id())
# Remove records with no review text
corpus_df = corpus_df.dropna()
```

```
corpus_df.persist()
print('Corpus size:', corpus_df.count())
corpus_df.show(5)
```

```
Corpus size: 18287527
+--------------------+---+
|         review_text| id|
+--------------------+---+
|Nice Story but ve...|  0|
|Beautiful and hea...|  1|
|Worth The Wait. T...|  2|
|written before. I...|  3|
|Entertaining Rev....|  4|
+--------------------+---+
only showing top 5 rows
```

Producing a list of tokens for each review text and displaying its count:

```
tokenizer = Tokenizer(inputCol="review_text", outputCol="words")
countTokens = udf(lambda words: len(words), IntegerType())
'''
tokenized_df = tokenizer.transform(corpus_df)
tokenized_df.select("review_text", "words").withColumn("tokens",
countTokens(col("words"))).show()
'''
regexTokenizer = RegexTokenizer(inputCol="review_text",
                                outputCol="words",pattern="\\w+", gaps=False)
# alternatively, pattern="\\w+", gaps(False) pattern="\\W"

tokenized_df = regexTokenizer.transform(corpus_df)
tokenized_df.select("review_text", "words") \
    .withColumn("tokens", countTokens(F.col("words"))).show()
```

```
+--------------------+--------------------+------+
|         review_text|               words|tokens|
+--------------------+--------------------+------+
|Nice Story but ve...|[nice, story, but...|    40|
|Beautiful and hea...|[beautiful, and, ...|    76|
|Worth The Wait. T...|[worth, the, wait...|    77|
|written before. I...|[written, before,...|   327|
|Entertaining Rev....|[entertaining, re...|    51|
|Fastest 600 page ...|[fastest, 600, pa...|    45|
|Amazing It is a c...|[amazing, it, is,...|    27|
|Huge impact Profo...|[huge, impact, pr...|    27|
|LOVED LOVED LOVED...|[loved, loved, lo...|    25|
|Five Stars very h...|[five, stars, ver...|     4|
|This is an awesom...|[this, is, an, aw...|    26|
|Kept me intereste...|[kept, me, intere...|    29|
|So many of these ...|[so, many, of, th...|    50|
|she is an incredi...|[she, is, an, inc...|    43|
|Thoroughly enjoye...|[thoroughly, enjo...|    42|
|This book has mad...|[this, book, has,...|    39|
|Not as good as th...|[not, as, good, a...|    33|
|Writer's Block Wo...|[writer, s, block...|    74|
|One of my favorit...|[one, of, my, fav...|    43|
|Wow This book was...|[wow, this, book,...|    74|
+--------------------+--------------------+------+
only showing top 20 rows
```

Adding Stop words to the list

```
stop_words = ['a', 'about', 'above', 'across', 'after', 'afterwards', 'again',
'against', 'all', 'almost', 'alone', 'along', 'already', 'also', 'although',
'always', 'am', 'among', 'amongst', 'amoungst', 'amount', 'an', 'and',
'another', 'any', 'anyhow', 'anyone', 'anything', 'anyway', 'anywhere', 'are',
'around', 'as', 'at', 'back', 'be', 'became', 'because', 'become', 'becomes',
'becoming', 'been', 'before', 'beforehand', 'behind', 'being', 'below',
'beside', 'besides', 'between', 'beyond', 'bill', 'both', 'bottom', 'but', 'by',
'call', 'can', 'cannot', 'cant', 'co', 'computer', 'con', 'could', 'couldnt',
'cry', 'de', 'describe', 'detail', 'do', 'done', 'down', 'due', 'during',
'each', 'eg', 'eight', 'either', 'eleven', 'else', 'elsewhere', 'empty',
'enough', 'etc', 'even', 'ever', 'every', 'everyone', 'everything',
'everywhere', 'except', 'few', 'fifteen', 'fify', 'fill', 'find', 'fire',
'first', 'five', 'for', 'former', 'formerly', 'forty', 'found', 'four', 'from',
'front', 'full', 'further', 'get', 'give', 'go', 'had', 'has', 'hasnt', 'have',
'he', 'hence', 'her', 'here', 'hereafter', 'hereby', 'herein', 'hereupon',
'hers', 'herself', 'him', 'himself', 'his', 'how', 'however', 'hundred', 'i',
'ie', 'if', 'in', 'inc', 'indeed', 'interest', 'into', 'is', 'it', 'its',
'itself', 'keep', 'last', 'latter', 'latterly', 'least', 'less', 'ltd', 'made',
'many', 'may', 'me', 'meanwhile', 'might', 'mill', 'mine', 'more', 'moreover',
'most', 'mostly', 'move', 'much', 'must', 'my', 'myself', 'name', 'namely',
'neither', 'never', 'nevertheless', 'next', 'nine', 'no', 'nobody', 'none',
'noone', 'nor', 'not', 'nothing', 'now', 'nowhere', 'of', 'off', 'often', 'on',
'once', 'one', 'only', 'onto', 'or', 'other', 'others', 'otherwise', 'our',
'ours', 'ourselves', 'out', 'over', 'own', 'part', 'per', 'perhaps', 'please',
'put', 'rather', 're', 'same', 'see', 'seem', 'seemed', 'seeming', 'seems',
'serious', 'several', 'she', 'should', 'show', 'side', 'since', 'sincere',
'six', 'sixty', 'so', 'some', 'somehow', 'someone', 'something', 'sometime',
'sometimes', 'somewhere', 'still', 'such', 'system', 'take', 'ten', 'than',
'that', 'the', 'their', 'them', 'themselves', 'then', 'thence', 'there',
'thereafter', 'thereby', 'therefore', 'therein', 'thereupon', 'these', 'they',
'thick', 'thin', 'third', 'this', 'those', 'though', 'three', 'through',
'throughout', 'thru', 'thus', 'to', 'together', 'too', 'top', 'toward',
'towards', 'twelve', 'twenty', 'two', 'un', 'under', 'until', 'up', 'upon',
'us', 'very', 'via', 'was', 'we', 'well', 'were', 'what', 'whatever', 'when',
'whence', 'whenever', 'where', 'whereafter', 'whereas', 'whereby', 'wherein',
'whereupon', 'wherever', 'whether', 'which', 'while', 'whither', 'who',
'whoever', 'whole', 'whom', 'whose', 'why', 'will', 'with', 'within', 'without',
'would', 'yet', 'you', 'your', 'yours', 'yourself', 'yourselves', '']

stop_words = stop_words + ['br','book','34','y','m','zu','ich']
```

Removing stop words and extract filtered data:

```
remover = StopWordsRemover(inputCol="words", outputCol="filtered")
tokenized_df1 = remover.transform(tokenized_df)
tokenized_df1.show(5)

stopwordList = stop_words

remover=StopWordsRemover(inputCol="filtered", outputCol="filtered_more"
,stopWords=stopwordList)
tokenized_df2 = remover.transform(tokenized_df1)
tokenized_df2.show(5)
```

```
+--------------------+---+--------------------+--------------------+
|         review_text| id|               words|            filtered|
+--------------------+---+--------------------+--------------------+
|Nice Story but ve...|  0|[nice, story, but...|[nice, story, rus...|
|Beautiful and hea...|  1|[beautiful, and, ...|[beautiful, heart...|
|Worth The Wait. T...|  2|[worth, the, wait...|[worth, wait, sto...|
|written before. I...|  3|[written, before,...|[written, really,...|
|Entertaining Rev....|  4|[entertaining, re...|[entertaining, re...|
+--------------------+---+--------------------+--------------------+
only showing top 5 rows

+--------------------+---+--------------------+--------------------+------------
--------+
|         review_text| id|               words|            filtered|
filtered_more|
+--------------------+---+--------------------+--------------------+------------
--------+
|Nice Story but ve...|  0|[nice, story, but...|[nice, story, rus...|[nice,
story, rus...|
|Beautiful and hea...|  1|[beautiful, and, ...|[beautiful, heart...|[beautiful,
heart...|
|Worth The Wait. T...|  2|[worth, the, wait...|[worth, wait, sto...|[worth,
wait, sto...|
|written before. I...|  3|[written, before,...|[written, really,...|[written,
really,...|
|Entertaining Rev....|  4|[entertaining, re...|[entertaining, re...|
[entertaining, re...|
+--------------------+---+--------------------+--------------------+------------
--------+
only showing top 5 rows
```

Producing features for filtered data:

```
cv = CountVectorizer(inputCol="filtered_more", outputCol="features", vocabSize =
10000)
cvmodel = cv.fit(tokenized_df2)
featurized_df = cvmodel.transform(tokenized_df2)
vocab = cvmodel.vocabulary
featurized_df.select('filtered_more','features','id').show(5)

countVectors = featurized_df.select('features','id')
countVectors.persist()
print('Records in the DF:', countVectors.count())
```

```
+--------------------+--------------------+---+
|       filtered_more|            features| id|
+--------------------+--------------------+---+
|[nice, story, rus...|(10000,[0,1,4,5,7...|  0|
|[beautiful, heart...|(10000,[1,5,9,12,...|  1|
|[worth, wait, sto...|(10000,[1,28,57,8...|  2|
|[written, really,...|(10000,[0,4,5,9,1...|  3|
|[entertaining, re...|(10000,[0,22,37,4...|  4|
+--------------------+--------------------+---+
only showing top 5 rows

Records in the DF: 18287527
```

Performing LDA:

```
#k=10 means 10 words per topic
lda = LDA(k=10, maxIter=10)
model = lda.fit(countVectors)
```

```
topics = model.describeTopics(5)
topics_rdd = topics.rdd

topics_words = topics_rdd\
       .map(lambda row: row['termIndices'])\
       .map(lambda idx_list: [vocab[idx] for idx in idx_list])\
       .collect()

for idx, topic in enumerate(topics_words):
    print ("topic: ", idx)
    print ("----------")
    for word in topic:
       print (word)
    print ("----------")
```

```
topic:  0
----------
story
characters
love
read
series
----------
```

```
topic:  1
----------
good
read
story
great
really
----------
topic:  2
----------
read
series
books
love
great
----------
topic:  3
----------
story
life
love
read
world
----------
topic:  4
----------
read
story
good
mystery
like
```

```
topic:  5
----------
read
like
great
time
interesting
----------
topic:  6
----------
read
reading
characters
story
great
----------
topic:  7
----------
great
read
life
good
information
----------
topic:  8
```

```
----------
love
story
like
really
read
----------
topic:  9
----------
read
author
good
books
like
----------
```

Topic Modeling for star rating less than 3

Filtering dataset for the 2015 data.

```
df_ml1 =
reviews_filter.filter((F.col("product_category")=="Digital_Ebook_Purchase") |
(F.col("product_category")=="Books") \
                   & (F.col("year")==2015) \
                   & (F.col("review_date")<'2015-02-01')
                   & (F.col("star_rating")<3))
```

Concatenating text and headline of reviews and creating an id column.

```
df_lda1 = df_ml1.withColumn('review_text',
                       F.concat(F.col('review_headline'),F.lit(' '),
F.col('review_body')))
corpus1 =df_lda1.select('review_text')
```

```
# This will return a new DF with all the columns + id
corpus_df1 = corpus1.withColumn("id", F.monotonically_increasing_id())
# Remove records with no review text
corpus_df1 = corpus_df1.dropna()

corpus_df1.persist()
print('Corpus size:', corpus_df.count())
corpus_df1.show(5)

corpus_df1.printSchema()
```

```
Corpus size: 18287527
+--------------------+---+
|         review_text| id|
+--------------------+---+
|Nice Story but ve...|  0|
|Beautiful and hea...|  1|
|Worth The Wait. T...|  2|
|written before. I...|  3|
|Entertaining Rev....|  4|
+--------------------+---+
only showing top 5 rows

root
 |-- review_text: string (nullable = true)
 |-- id: long (nullable = false)
```

Producing a list of tokens for each review text and displaying its count:

```
tokenizer1 = Tokenizer(inputCol="review_text", outputCol="words")
countTokens1 = udf(lambda words: len(words), IntegerType())
'''
tokenized_df = tokenizer.transform(corpus_df)
tokenized_df.select("review_text", "words").withColumn("tokens",
countTokens(col("words"))).show()
'''
regexTokenizer1 = RegexTokenizer(inputCol="review_text",
                                outputCol="words",pattern="\\w+", gaps=False)
# alternatively, pattern="\\w+", gaps(False) pattern="\\W"

tokenized_df1 = regexTokenizer1.transform(corpus_df1)
tokenized_df1.select("review_text", "words") \
    .withColumn("tokens", countTokens1(F.col("words"))).show()
```

```
+--------------------+--------------------+------+
|         review_text|               words|tokens|
+--------------------+--------------------+------+
|Nice Story but ve...|[nice, story, but...|    40|
|Beautiful and hea...|[beautiful, and, ...|    76|
|Worth The Wait. T...|[worth, the, wait...|    77|
|written before. I...|[written, before,...|   327|
|Entertaining Rev....|[entertaining, re...|    51|
|Fastest 600 page ...|[fastest, 600, pa...|    45|
|Amazing It is a c...|[amazing, it, is,...|    27|
|Huge impact Profo...|[huge, impact, pr...|    27|
|LOVED LOVED LOVED...|[loved, loved, lo...|    25|
|Five Stars very h...|[five, stars, ver...|     4|
|This is an awesom...|[this, is, an, aw...|    26|
|Kept me intereste...|[kept, me, intere...|    29|
|So many of these ...|[so, many, of, th...|    50|
|she is an incredi...|[she, is, an, inc...|    43|
|Thoroughly enjoye...|[thoroughly, enjo...|    42|
|Not as good as th...|[not, as, good, a...|    33|
|Writer's Block Wo...|[writer, s, block...|    74|
|One of my favorit...|[one, of, my, fav...|    43|
|Wow This book was...|[wow, this, book,...|    74|
|THE BEST OF THE B...|[the, best, of, t...|    86|
+--------------------+--------------------+------+
only showing top 20 rows
```

Removing Stop words and extracting filtered data:

```
remover1 = StopWordsRemover(inputCol="words", outputCol="filtered")
tokenized_df2 = remover1.transform(tokenized_df1)
tokenized_df2.show(5)

stopwordList = stop_words

remover1=StopWordsRemover(inputCol="filtered", outputCol="filtered_more"
,stopWords=stopwordList)
tokenized_df3 = remover1.transform(tokenized_df2)
tokenized_df3.show(5)
```

```
+--------------------+---+--------------------+--------------------+
|         review_text| id|               words|            filtered|
+--------------------+---+--------------------+--------------------+
|Nice Story but ve...|  0|[nice, story, but...|[nice, story, rus...|
|Beautiful and hea...|  1|[beautiful, and, ...|[beautiful, heart...|
|Worth The Wait. T...|  2|[worth, the, wait...|[worth, wait, sto...|
|written before. I...|  3|[written, before,...|[written, really,...|
|Entertaining Rev....|  4|[entertaining, re...|[entertaining, re...|
+--------------------+---+--------------------+--------------------+
only showing top 5 rows

+--------------------+---+--------------------+--------------------+------------
--------+
|         review_text| id|               words|            filtered|
filtered_more|
+--------------------+---+--------------------+--------------------+------------
--------+
|Nice Story but ve...|  0|[nice, story, but...|[nice, story, rus...|[nice,
story, rus...|
|Beautiful and hea...|  1|[beautiful, and, ...|[beautiful, heart...|[beautiful,
heart...|
|Worth The Wait. T...|  2|[worth, the, wait...|[worth, wait, sto...|[worth,
wait, sto...|
|written before. I...|  3|[written, before,...|[written, really,...|[written,
really,...|
|Entertaining Rev....|  4|[entertaining, re...|[entertaining, re...|
[entertaining, re...|
+--------------------+---+--------------------+--------------------+------------
--------+
only showing top 5 rows
```

Producing Features for filtered data:

```
cv1 = CountVectorizer(inputCol="filtered_more", outputCol="features", vocabSize
= 10000)
cvmodel1 = cv1.fit(tokenized_df3)
featurized_df1 = cvmodel1.transform(tokenized_df3)
vocab1 = cvmodel1.vocabulary
featurized_df1.select('filtered_more','features','id').show(5)

countVectors2 = featurized_df1.select('features','id')
countVectors2.persist()
print('Records in the DF:', countVectors2.count())
```

```
+--------------------+--------------------+---+
|       filtered_more|            features| id|
+--------------------+--------------------+---+
|[nice, story, rus...|(10000,[0,1,4,5,7...|  0|
|[beautiful, heart...|(10000,[1,5,9,12,...|  1|
|[worth, wait, sto...|(10000,[1,27,56,8...|  2|
|[written, really,...|(10000,[0,4,5,9,1...|  3|
|[entertaining, re...|(10000,[0,22,37,4...|  4|
+--------------------+--------------------+---+
only showing top 5 rows

Records in the DF: 17950045
```

Performing LDA:

```
lda1 = LDA(k=10, maxIter=5)
model1 = lda1.fit(countVectors2)
```

```
topics1 = model1.describeTopics()
topics_rdd1 = topics1.rdd

topics_words1 = topics_rdd1\
       .map(lambda row: row['termIndices'])\
       .map(lambda idx_list: [vocab1[idx] for idx in idx_list])\
       .collect()

for idx, topic in enumerate(topics_words1):
    print ("topic: ", idx)
    print ("----------")
    for word in topic:
       print (word)
    print ("----------")
```

```
topic:  0
----------
story
good
characters
read
love
series
author
time
like
great
----------
topic:  1
----------
good
read
story
great
stars
```

```
really
like
love
series
characters
----------
topic:  2
----------
read
series
books
great
like
love
reading
story
loved
wait
----------
topic:  3
----------
story
read
love
characters
like
written
great
novel
way
life
----------
topic:  4
----------
read
good
like
books
great
reading
new
story
easy
author
----------
```

```
topic:  5
----------
read
great
time
like
history
reading
good
life
know
```

```
people
----------
topic:  6
----------
read
love
loved
series
characters
story
reading
great
wait
books
----------
topic:  7
----------
great
read
reading
life
recommend
story
good
god
series
stars
----------
topic:  8
----------
story
love
really
life
like
read
characters
loved
stars
know
----------
topic:  9
----------
read
good
author
characters
story
enjoyed
books
like
great
reading
----------
```

Does topic modeling provides good approximation to number of stars given in the review?

Both analysis performed above for star rating greater than 3 and less than 3 illustrate top topics with similar words. Hence topic modeling does not provide good approximation to the number of stars given in the review in this case.

7. Conclusion:

The analysis performed explores the fundamental attributes of the data. A focus on the "books" product category enables us to extract some findings from the data. Number of Reviews for Digital E-book Purchase is less than that for Books till 2012 but then increases substantially than Books till 2014. The Average star rating for Books is consistently high over the years as compared to Digital E-books. Further, topic modeling using LDA was performed which helped to fetch top topics from reviews of Books and E-books. However for different star ratings the model was not effective in providing good approximation to number of stars given in the review by users.

8. References:

Amazon Reviews Dataset: https://registry.opendata.aws/amazon-reviews/

Documentation: https://s3.amazonaws.com/amazon-reviews-pds/readme.html

Code: https://spark.apache.org/docs/latest/api/python/index.html