Project 2

Chandra Sekhar Padhy **CSP180002**

Introduction and Problem Description :

We were given a dataset which was hosted on AWS. After exploring the dataset, many columns were explored like customer_id, review_id, product_id, helpful_votes, total_votes, star_rating, review_body, etc.

I tried to develop relationships between some of these coulmns mentioned above and thus draw effective conclusions on how various products performed, wheter the one which performed better were the digital or the printed articles to name a few.

I decided to filter the data before analysing to remove multiple reviews given by the same user and thus draw effective and accurate assumptions.

The most effective weeks for the performance of digital and printed books with the best ratings were analysed along the way. LDA was performed on the data set, topic modeling was done to analyse average star ratings below 3 as well as above 3 star ratings, so as covering most of the topics. Stop words which were a hinderance were dealt with for accurate results.

Project Solution :

Initial start to Spark Session :

```
spark
```

```
Starting Spark application
```

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
0	application_1588701453956_0008	pyspark3	idle	Link	Link	✓

```
SparkSession available as 'spark'.
<pyspark.sql.session.SparkSession object at 0x7fcb288a1b00>
```

```
from pyspark.sql import functions as F
```

Load the initial Dataset :

```
# 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/*")
```

Print Schema and Number of Columns :

```
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)
```

```
df.columns
```

```
['marketplace', 'customer_id', 'review_id', 'product_id', 'product_parent', 'product_title', 'star_rating', 'helpful_votes', 't
otal_votes', 'vine', 'verified_purchase', 'review_headline', 'review_body', 'review_date', 'year', 'product_category']
```

Columns to Keep:

```
columns_to_keep = ['customer_id', 'review_id', 'product_id', 'product_parent',
                   'product_title', 'star_rating', 'helpful_votes', 'total_votes',
                   'verified_purchase', 'review_date', 'year', 'product_category','review_body','review_headline']
df_limited = df.select(columns_to_keep).filter(F.col("year")>2004)
```

```
#print schema
df_limited.printSchema()
```

```
root
 |-- 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)
 |-- verified_purchase: string (nullable = true)
 |-- review_date: date (nullable = true)
 |-- year: integer (nullable = true)
 |-- product_category: string (nullable = true)
 |-- review_body: string (nullable = true)
 |-- review_headline: string (nullable = true)
```

Filter Data before further Analysis :

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

Filter_data = temp_data.rownum.isin(1)
Filtered_data = temp_data.where(Filter_data)
Filtered_data.persist()
```

```
DataFrame[customer_id: string, review_id: string, product_id: string, product_parent: string, product_title: string, star_ratin
g: int, helpful_votes: int, total_votes: int, verified_purchase: string, review_date: date, year: int, product_category: strin
g, review_body: string, review_headline: string, rownum: int]
```

Question 1) Explore the dataset and provide analysis by product-category and year

1) Number of reviews

```
Query : Filtered_data.groupby("year","product_category").agg(F.countDistinct("review_id").alias('Number of Reviews')).show(5)
```

Output :

```
Filtered_data.groupby("year","product_category").agg(F.countDistinct("review_id").alias('Number of Reviews')).show(5)
```

```
+----+--------------------+-----------------+
|year|    product_category|Number of Reviews|
+----+--------------------+-----------------+
|2014|               Books|          3540845|
|2010|Digital_Ebook_Pur...|           102515|
|2015|               Books|          2860727|
|2013|            Wireless|          1767125|
|2014|         Mobile_Apps|          1728278|
+----+--------------------+-----------------+
only showing top 5 rows
```

2) Number of users

Query :

```
Filtered_data.groupby("year","product_category").agg(F.countDistinct("customer_id").alias('Number of Users')).show(5)
```

Output:

```
Filtered_data.groupby("year","product_category").agg(F.countDistinct("customer_id").alias('Number of Users')).show(5)
```

```
+----+--------------------+---------------+
|year|    product_category|Number of Users|
+----+--------------------+---------------+
|2014|               Books|        1859221|
|2010|Digital_Ebook_Pur...|          61197|
|2015|               Books|        1548551|
|2013|            Wireless|        1193454|
|2014|         Mobile_Apps|         988660|
+----+--------------------+---------------+
only showing top 5 rows
```

3) Average and Median review stars

Query :

```
Filtered_data.groupby("year","product_category").agg(F.avg("star_rating").alias('Avg_Rating'),
                                                     F.expr('percentile_approx(star_rating,0.5)').alias('Median_Rating')).show()
```

Output :

```
Filtered_data.groupby("year","product_category").agg(F.avg("star_rating").alias('Avg_Rating'),
                                     F.expr('percentile_approx(star_rating,0.5)').alias('Median_Rating')).show()
```

```
+----+--------------------+------------------+-------------+
|year|    product_category|        Avg_Rating|Median_Rating|
+----+--------------------+------------------+-------------+
|2014|               Books|4.4732762187450135|            5|
|2010|Digital_Ebook_Pur...| 3.821964043584716|            4|
|2015|               Books| 4.497377416735384|            5|
|2013|            Wireless| 3.820195877706306|            4|
|2014|         Mobile_Apps|3.9685729279917052|            5|
|2013|Digital_Video_Dow...|  4.20839463591457|            5|
|2011|Digital_Ebook_Pur...|4.0555525408625765|            5|
|2008|               Books| 4.233252143319705|            5|
|2012|         Mobile_Apps|3.9950581955104916|            5|
|2008|            Wireless|  3.76971886288676|            4|
|2011|               Books| 4.251151283351917|            5|
|2015|           Video_DVD|  4.52990406398113|            5|
|2007|Digital_Video_Dow...|  3.59992298806315|            4|
|2012|           Video_DVD| 4.217597322758286|            5|
|2011|Digital_Video_Dow...|3.7777352349956925|            5|
|2009|Digital_Video_Dow...| 3.699570815450644|            4|
|2015|            Wireless| 3.985510516039572|            5|
|2010|Digital_Video_Dow...|3.7574712643678163|            4|
|2013|               Books|4.4125012812075175|            5|
|2009|                  PC|3.9672266000384395|            5|
+----+--------------------+------------------+-------------+
only showing top 20 rows
```

4) Percentiles of length of the review. Use the following percentiles:

Query :

```
from pyspark.sql.functions import length,count, mean, stddev_pop, min, max
df1 = Filtered_data.withColumn('length', length(df.review_body))
df2 = df1.groupby("year","product_category").agg(F.avg("length").alias('average of star reviews'))
colName = "average of star reviews"
quantileProbs = [0.1, 0.25, 0.5, 0.75, 0.9, 0.95]
relError = 0.05
df2.stat.approxQuantile("average of star reviews", quantileProbs, relError)
```

Output :

```
[205.48028507600662, 349.16350523270023, 586.2441338149835, 853.2765772362093
, 945.7590082915988, 2207.5789473684213]
```

5) Percentiles for number of reviews per product. For example, 10% of books got 5 or less reviews

Query :

```
from pyspark.sql.functions import length,count, mean, stddev_pop, min, max
df1 = Filtered_data.groupby("year","product_id","product_category").agg(F.countDistinct("review_id").alias('Total_Number_of_Reviews'))
colName = "Total_Number_of_Reviews"
quantileProbs = [0.1, 0.25, 0.5, 0.75, 0.9, 0.95]
relError = 0.05
df1.stat.approxQuantile("Total_Number_of_Reviews", quantileProbs, relError)
```

Output :

```
[1.0, 1.0, 2.0, 4.0, 5108.0, 31128.0]
```

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

Query :

```
from pyspark.sql.functions import *
A = Filtered_data.star_rating.isin(4)
X = Filtered_data.star_rating.isin(5)
df_week_number=Filtered_data.select("product_category","year","review_date").withColumn("week_number",weekofyear("review_date")).where(A | X)
df_2 = df_week_number.groupby("product_category","year","week_number").agg(F.countDistinct("week_number").alias("count"))
df_2.drop('count').show()
```

Output :

product_category	year	week_number
Digital_Ebook_Pur...	2014	11
Digital_Ebook_Pur...	2015	16
Video_DVD	2015	12
Video_DVD	2011	37
Books	2011	36
Books	2007	37
Books	2008	48
Digital_Ebook_Pur...	2015	11
Mobile_Apps	2013	2
Digital_Ebook_Pur...	2013	49
Digital_Ebook_Pur...	2013	19
Books	2014	6
Books	2009	36
PC	2010	20
Books	2010	3
Video_DVD	2009	9
Books	2010	12
Books	2006	12
PC	2012	24
Mobile_Apps	2011	32

only showing top 20 rows

Q2) Provide detailed analysis of "Digital eBook Purchase" versus Books.

1. Using Spark Pivot functionality, produce DataFrame with following columns:
    A. Year
    B. Month
    C. Total number of reviews for "Digital eBook Purchase" category
    D. Total number of reviews for "Books" category
    E. Average stars for reviews for "Digital eBook Purchase" category
    F. Average stars for reviews for "Books" category

Query :

```
pivot_table = ['Digital_Ebook_Purchase','Books']
pivot_output = Filtered_data.groupBy("year",F.month(F.col("review_date"))).pivot("product_category",pivot_table)\
 .agg((F.count("review_id")).alias("total_count_of_reviews"),
 F.round(F.mean("star_rating"),3).alias("Average_star_rating")).sort("year","month(review_date)",ascending=True).show()
```

Output :

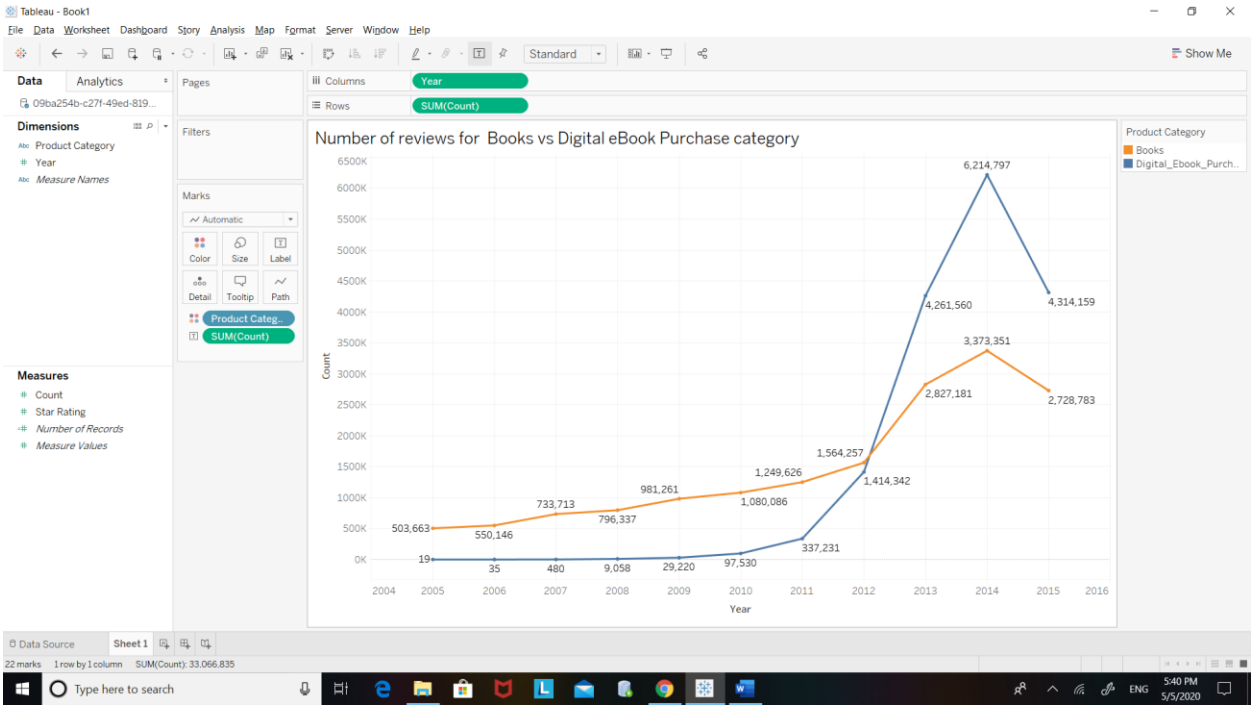
```
pivot_table = ['Digital_Ebook_Purchase','Books']
pivot_output = Filtered_data.groupBy("year",F.month(F.col("review_date"))).pivot("product_category",pivot_table)\
 .agg((F.count("review_id")).alias("total_count_of_reviews"),
 F.round(F.mean("star_rating"),3).alias("Average_star_rating")).sort("year","month(review_date)",ascending=True).show()
```

```
+----+------------------+----------------------------------------------+---------------------------------------+-------------
--------------+------------------------+
|year|month(review_date)|Digital_Ebook_Purchase_total_count_of_reviews|Digital_Ebook_Purchase_Average_star_rating|Books_total_c
ount_of_reviews|Books_Average_star_rating|
+----+------------------+----------------------------------------------+---------------------------------------+-------------
--------------+------------------------+
|2005|                 1|                                             1|                                    5.0|
40423|                   4.121|
|2005|                 2|                                          null|                                   null|
33728|                   4.125|
|2005|                 3|                                             2|                                    4.5|
38877|                   4.122|
|2005|                 4|                                             1|                                    5.0|
36886|                   4.132|
|2005|                 5|                                             1|                                    1.0|
36877|                   4.132|
|2005|                 6|                                          null|                                   null|
36607|                   4.115|
|2005|                 7|                                             3|                                    2.0|
45946|                   4.128|
|2005|                 8|                                             3|                                  2.667|
58922|                   4.186|
|2005|                 9|                                             2|                                    4.0|
58133|                   4.203|
|2005|                10|                                             4|                                    4.0|
51214|                    4.18|
|2005|                11|                                             1|                                    5.0|
40886|                   4.151|
|2005|                12|                                             1|                                    5.0|
```

Q2) 2)

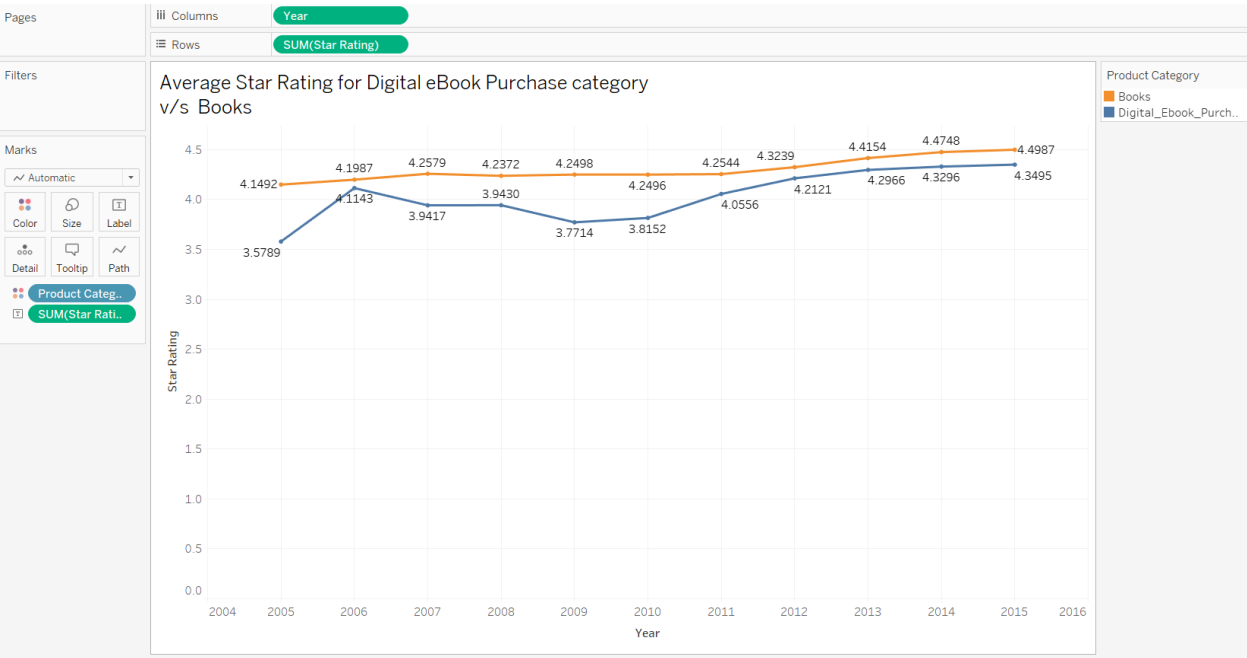
2. Produce two graphs to demonstrate aggregations from #1: 1. Number of reviews 2. Average stars

1 – Number of reviews for Books Vs Digital eBook Purchase category



From the above graph we can conclude that the total count for the number of reviews for digital books is greater than that for the printed books.

2 – Avearage Star rating for Digital eBook Purchase category vs Books



From the above graph we can conclude that the average rating for printed books is more than the average star rating for Digital eBook category.

Q3) Identify similar products (books) in both categories. Use "product_title" to matchproducts. To account for potential differences in naming of products, compare titlesafter stripping spaces and converting to lower case.

Is there a difference in average rating for the similar books in digital and printed form?

Query :

```
productfilter=['Digital_Ebook_Purchase']
digital_e_book_purchase = Filtered_data.groupBy("product_title","product_category")\
 .agg((F.count("review_id")).alias("dig_book_count_of_reviews"),
 F.round(F.mean("star_rating"),3).alias("dig_book_Avg_star_rating")).filter(F.col("product_category").isin(productfilter))

trimmed_digital_book = digital_e_book_purchase.select(F.lower(F.trim(F.col("product_title"))).alias("product_title"),F.col("dig_
book_count_of_reviews") \
                      ,F.col("dig_book_Avg_star_rating"))

var = ['Books']
book = Filtered_data.groupBy("product_title","product_category")\
 .agg((F.count("review_id")).alias("book_count_of_reviews"),
 F.round(F.mean("star_rating"),3).alias("book_Avg_star_rating")).filter(F.col("product_category").isin(var))

trimmed_book = book.select(F.lower(F.trim(F.col("product_title"))).alias("product_title"),F.col("book_count_of_reviews") \
                      ,F.col("book_Avg_star_rating"))

join_1 = trimmed_book["product_title"] == trimmed_digital_book["product_title"]
joinType = "inner"
final=trimmed_book.join(trimmed_digital_book, join_1, joinType)

final.show()
```

Output :

product_title	book_count_of_reviews	book_Avg_star_rating	product_title	dig_book_count_of_reviews	dig_book_Avg_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

The above analysis shows that Printed Books has more higher rated stars as compared to the digital form.

Q2) 3) 2) To answer #1, you may calculate number of items with high stars in digital form versus printed form, and vise versa. Alternatively, you can make the conclusion by using appropriate pairwise statistic

Query :

```
high_stars = F.col("book_Avg_star_rating")>4
final.where(high_stars).count()


high_stars_1 = F.col("dig_book_Avg_star_rating")>4
final.where(high_stars_1).count()
```

Output :

```
245526
```

Importing ML Libraries :

```
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
```

Q2) 4) 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.

Topic Modelling for reviews more than three stars

Query :

In [11]:

```
df_ml = Filtered_data.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))
```

```
df1 = df_ml.withColumn('review_text',
                       F.concat(F.col('review_headline'),F.lit(' '), F.col('review_body')))
df_2 =df1.select('review_text')
```

```
corpus_df = df_2.withColumn("id", F.monotonically_increasing_id())
```

```
corpus_df = corpus_df.dropna()
```

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

```
Corpus size: 18287533
+--------------------+---+
|         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
```

In [20]:

```
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

```
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','m','y','ich','zu']
```

```
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
```

```
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: 18287533
```

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 ("----------")
```

Output :

```
topic:  0
----------
story
love
characters
read
series
----------
topic:  1
----------
good
read
story
great
really
----------
topic:  2
----------
read
series
books
great
love
----------
topic:  3
----------
story
life
read
love
world
```

Topic Modeling for Review Stars less than 3

```
df_ml_2 = Filtered_data.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))
```

```
df1 = df_ml_2.withColumn('review_text',
                       F.concat(F.col('review_headline'),F.lit(' '), F.col('review_body')))
corpus =df1.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_df.printSchema()
```

```
Corpus size: 17950046
+--------------------+---+
|         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)
```

```
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|
|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
```

```
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
```

```
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,[1,27,56,8...|  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: 17950046
```

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

```
Records in the DF: 17950046
```

Performing LDA :

```
lda = LDA(k=10, maxIter=5)
model = lda.fit(countVectors)
```

```
topics = model.describeTopics()
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 ("----------")
```

Output:

```
topic:  0
----------
story
good
characters
read
love
series
author
time
like
great
----------
topic:  1
----------
good
read
story
great
stars
really
like
love
characters
series
----------
topic:  2
----------
read
series
books
like
great
love
reading
story
loved
wait
----------
```

Q2) 4) Does topic modeling provides good approximation to number of stars given in the review

Ans : After running the LDAs for both the review stars greater than and less than 3 we can conclude that the topic modeling is does not give a good approximation to number of stars given in the review.

Conclusion :

Interpreting from the graph we could conclude that the count of reviews was more for Digital eBooks as compared to the Printed ones whereas the average star rating was more for the Printed Books as compared to the Digital eBooks.

Added to that we could also come to a conclusion that the topic modeling run for the LDA was not effective.

References :

Class Notes : Lectures 8,9,10,11

Class Mate : Prathamesh Namjoshi

Tableau

https://spark.apache.org/docs/latest/api.html