Amazon Customer Review Data Analysis

Introduction:

Big data is a term applied to data sets whose size or type is beyond the ability of traditional relational databases to capture, manage and process the data with low latency. Big data has one or more of the following characteristics: high volume, high velocity or high variety. Big data analytics is the often complex process of examining large and varied data sets, or big data, to uncover information -- such as hidden patterns, unknown correlations, market trends and customer preferences -- that can help organizations make informed business decisions.

Problem description:

In this project we are practicing and demonstrating out Big Data (BD) and Analytics skills. I am using Amazon customer review data which has more than 160 Million (160,796,570) observations. Using EMR, HDFS, Spark and Tableau, I am handling the data and systematically extracting information from. I am performing exploratory analysis on the review dataset and I am focusing on features like reviews and rating for detailed analysis. I am using Latent Dirichlet Allocation model for processing reviews and finding hidden patterns.

Data Set:
Amazon reviews dataset: *https://registry.opendata.aws/amazon-reviews/*

Project Environment:
AWS EMR - AWS Educate Class account.
Project tools:
1. AWS EMR
2. AWS HDFS
3. AWS Spark
4. Tableau

Dataset Requirements:

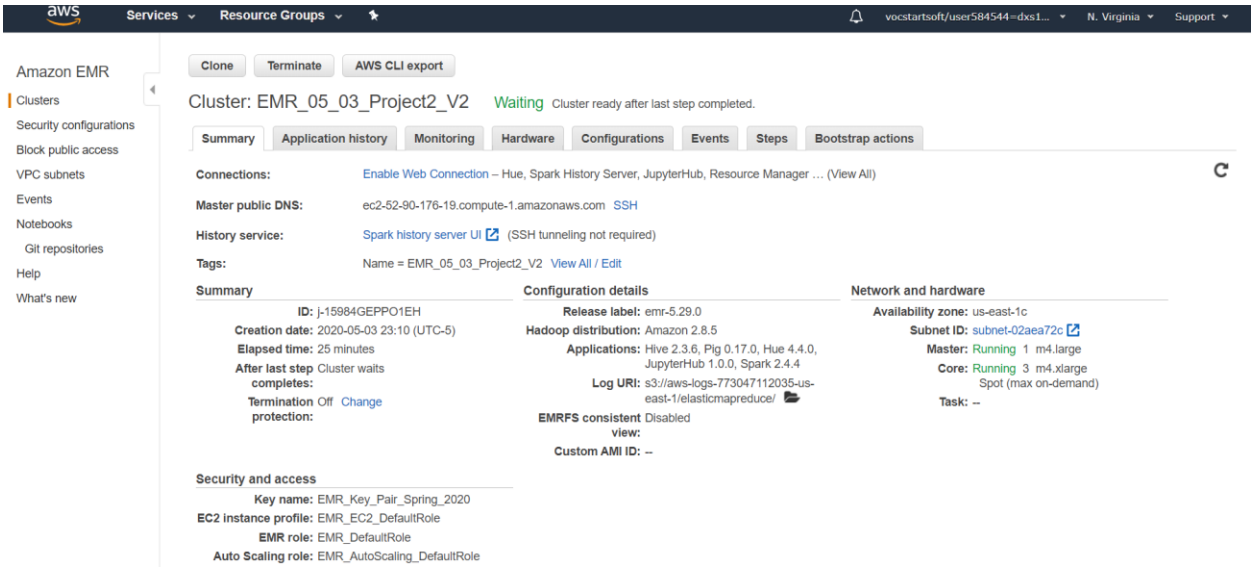
A) Use the following product categories:
- Digital_Ebook_Purchase
- Books
- Wireless
- PC
- Mobile_Apps
- Video_DVD
- Digital_Video_Download

B) Start your analysis from year 2005.

C) Exclude multiple reviews by the same users for the same product. In the case the same user has reviewed particular product more than once, exclude all reviews following the first review. First review should remain as part of the analysis

Step 1: AWS EMR:

Provisioned EMR with below details:



Step 2:-

A) Created directory for below each product category in HDFS:

- Digital_Ebook_Purchase
- Books
- Wireless
- PC
- Mobile_Apps
- Video_DVD
- Digital_Video_Download

hdfs dfs -mkdir -p /hive/amazon-reviews-pds/parquet/product_category=Digital_Ebook_Purchase/

hdfs dfs -mkdir -p /hive/amazon-reviews-pds/parquet/product_category=Books/

hdfs dfs -mkdir -p /hive/amazon-reviews-pds/parquet/product_category=Wireless/

hdfs dfs -mkdir -p /hive/amazon-reviews-pds/parquet/product_category=PC/

hdfs dfs -mkdir -p /hive/amazon-reviews-pds/parquet/product_category=Mobile_Apps/

hdfs dfs -mkdir -p /hive/amazon-reviews-pds/parquet/product_category=Video_DVD/

hdfs dfs -mkdir -p /hive/amazon-reviews-pds/parquet/product_category=Digital_Video_Download/

B) Copying dataset from S3 in HDFS for each of the product category mentioned:

s3-dist-cp --src=s3://amazon-reviews-pds/parquet/product_category=Wireless/ --dest=hdfs:///hive/amazon-reviews-pds/parquet/product_category=Wireless/

s3-dist-cp --src=s3://amazon-reviews-pds/parquet/product_category=Digital_Ebook_Purchase/ --dest=hdfs:///hive/amazon-reviews-pds/parquet/product_category=Digital_Ebook_Purchase/

s3-dist-cp --src=s3://amazon-reviews-pds/parquet/product_category=Books/ --dest=hdfs:///hive/amazon-reviews-pds/parquet/product_category=Books/

s3-dist-cp --src=s3://amazon-reviews-pds/parquet/product_category=PC/ --dest=hdfs:///hive/amazon-reviews-pds/parquet/product_category=PC/

s3-dist-cp --src=s3://amazon-reviews-pds/parquet/product_category=Mobile_Apps/ --dest=hdfs:///hive/amazon-reviews-pds/parquet/product_category=Mobile_Apps/

s3-dist-cp --src=s3://amazon-reviews-pds/parquet/product_category=Video_DVD/ --dest=hdfs:///hive/amazon-reviews-pds/parquet/product_category=Video_DVD/

s3-dist-cp --src=s3://amazon-reviews-pds/parquet/product_category=Digital_Video_Download/ --dest=hdfs:///hive/amazon-reviews-pds/parquet/product_category=Digital_Video_Download/

C) Importing dataset from HDFS into spark dataframe:

Code:-

```
# 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/*")
```

```
In [3]:
# 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/*")
```

D) Checking the schema of dataframe:

Code:

```
df.printSchema()
```

```
In [4]: 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)
```

E) Considering reviews after 2004 only:

Code:

```
df1 = df.filter(F.col("year")>2004)
```

```
In [4]: df1 = df.filter(F.col("year")>2004)
```

F) Finding the unique rows based on the product category, product id and customer id:

Code:

```
from pyspark.sql.window import Window
import pyspark.sql.functions as F
 df12 = df1.select("*",F.row_number().over(Window.partitionBy("product_category",'customer_id', 'product_id').orderBy(df['review_date'])))
```

```
In [5]: from pyspark.sql.window import Window
        import pyspark.sql.functions as F

        df12 = df1.select("*",F.row_number().over(Window.partitionBy("product_category",'customer_id', 'product_id')
                                                  .orderBy(df['review_date'])))
```

```
df12=df12.withColumnRenamed("row_number() OVER (PARTITION BY product_category, customer_id, product_id ORDER BY review_date ASC NULLS FIRST unspecifiedframe$())", "row_num")
```

```
In [6]: df12=df12.withColumnRenamed("row_number() OVER (PARTITION BY product_category, customer_id, product_id ORDER\
                                   BY review_date ASC NULLS FIRST unspecifiedframe$())", "row_num")
```

```
# The count of rows which we will be removing
df12.where(F.col("row_num")>1).count()
```

```
In [7]: # The count of rows which we will be removing
        df12.where(F.col("row_num")>1).count()

        5607955
```

```
# Unique rows
df2=df12.where(F.col("row_num")==1)
```

```
In [7]:  # Unique rows
         df2=df12.where(F.col("row_num")==1)
```

Ans: Df2 is the main filtered dataset on which I am performing my analysis.

G) Number of reviews:

```
In [9]:  #Number of reviews
         df2.count()

         65911069
```

Ans: The count of reviews on which I am performing my analysis is 65911069

Step 3:
1. Explore the dataset and provide analysis by product-category and year:

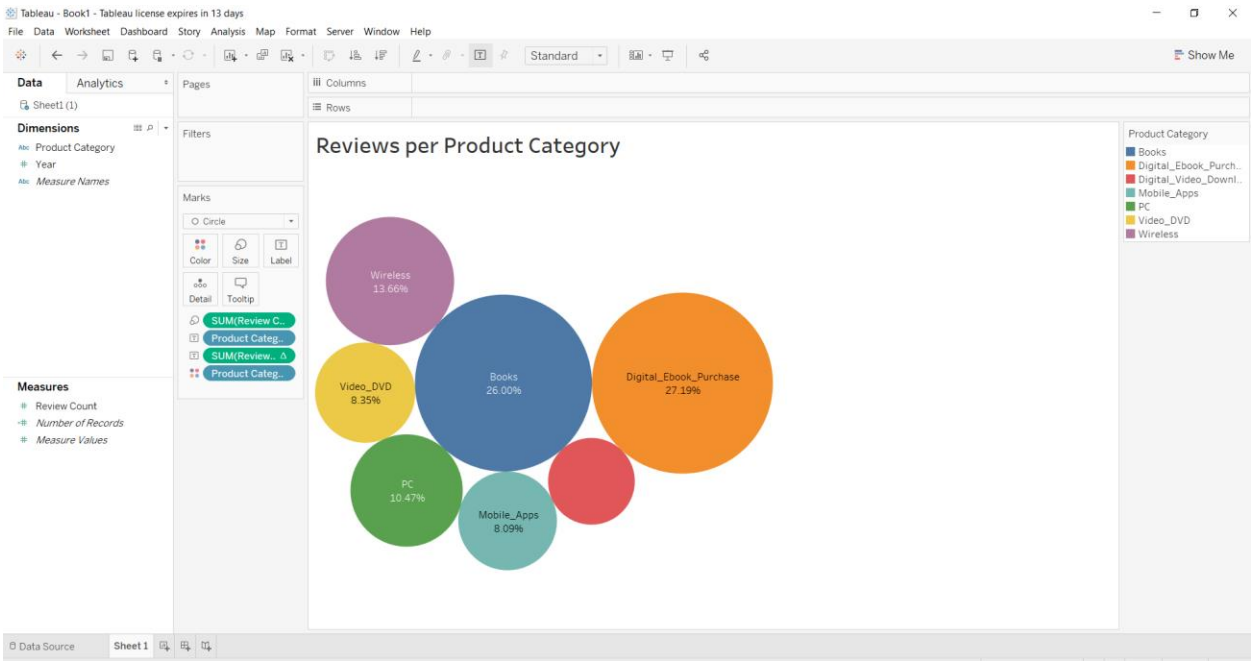
1. Number of reviews:

Code:

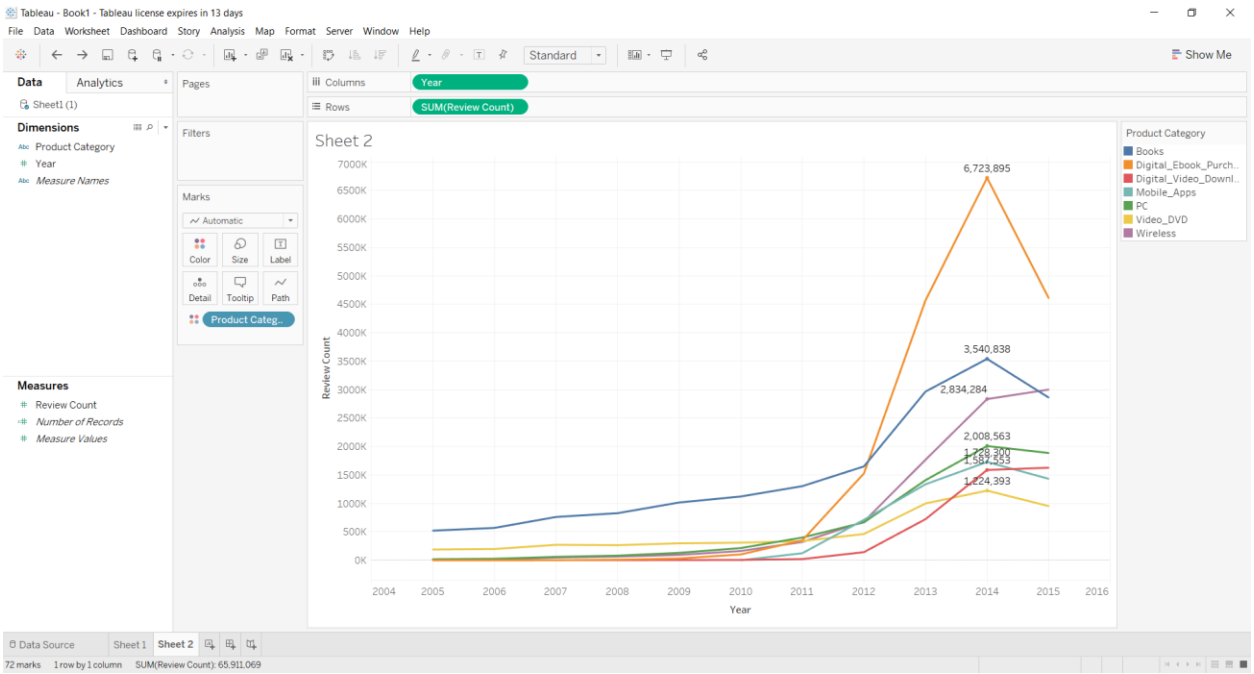
```
#1.Number of reviews product_category and year wise
df2.groupby("product_category","year")\
   .agg(F.count("review_id"))\
   .sort("year")\
   .show(80,truncate = False)
```

```
In [10]: #1.Number of reviews product_category and year wise
         df2.groupby("product_category","year")\
             .agg(F.count("review_id"))\
             .sort("year")\
             .show(80,truncate = False)
```

```
+----------------------+----+-----------------+
|product_category      |year|count(review_id)|
+----------------------+----+-----------------+
|PC                    |2005|18166            |
|Digital_Ebook_Purchase|2005|19               |
|Books                 |2005|521047           |
|Digital_Video_Download|2005|12               |
|Video_DVD             |2005|189265           |
|Wireless              |2005|11835            |
|Video_DVD             |2006|197551           |
|PC                    |2006|26291            |
|Wireless              |2006|19857            |
|Digital_Ebook_Purchase|2006|36               |
|Books                 |2006|568401           |
|Digital_Video_Download|2006|185              |
|PC                    |2007|59890            |
|Digital_Ebook_Purchase|2007|508              |
|Digital_Video_Download|2007|2597             |
|Books                 |2007|761037           |
|Video_DVD             |2007|271324           |
|Wireless              |2007|47738            |
|Wireless              |2008|63672            |
|Digital_Ebook_Purchase|2008|9607             |
|Video_DVD             |2008|265477           |
|PC                    |2008|81436            |
|Digital_Video_Download|2008|3083             |
|Books                 |2008|827721           |
|Digital_Ebook_Purchase|2009|31106            |
|Wireless              |2009|94052            |
|PC                    |2009|130075           |
|Digital_Video_Download|2009|3262             |
|Video_DVD             |2009|297763           |
|Books                 |2009|1015572          |
|Digital_Video_Download|2010|6090             |
|Books                 |2010|1120772          |
|Wireless              |2010|162371           |
|Digital_Ebook_Purchase|2010|102515           |
|Mobile_Apps           |2010|8                |
|Video_DVD             |2010|309009           |
|PC                    |2010|213890           |
```



Interpretation: We can see that Books and Digital_Ebooks_Purchase category constitutes more than 53% of the dataset.



Interpretation: We can see that Digital_Ebooks_Purchase observed maximum growth rate as well as the highest peak which indicate it as a booming category.

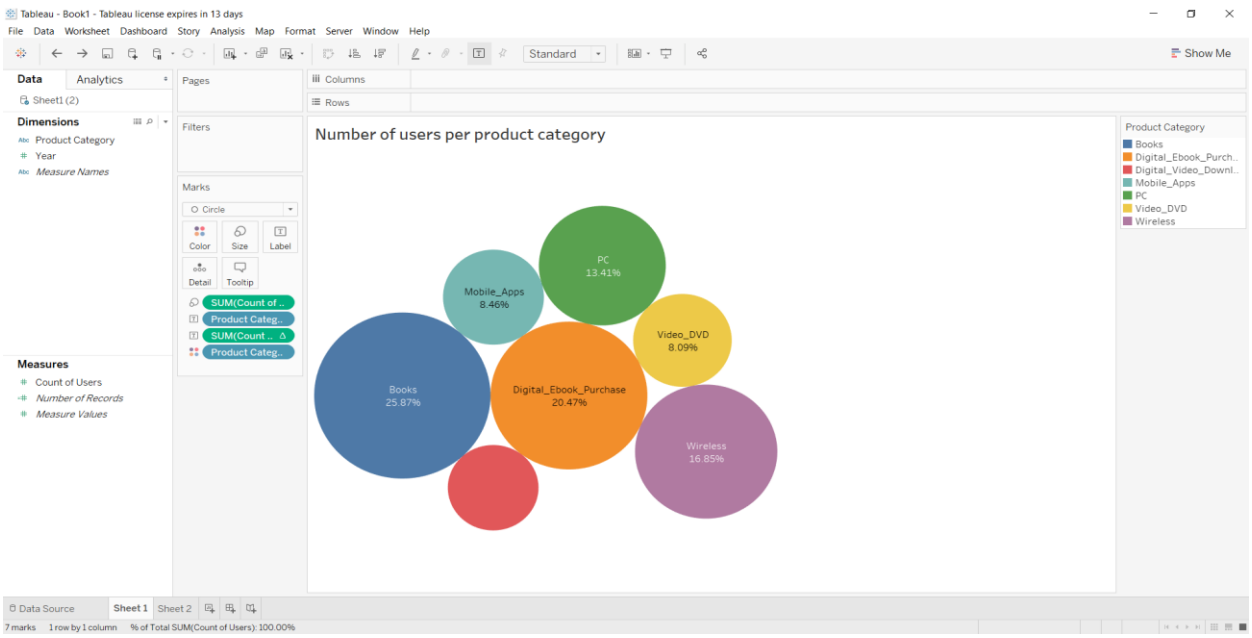
2. Number of users:

Code:

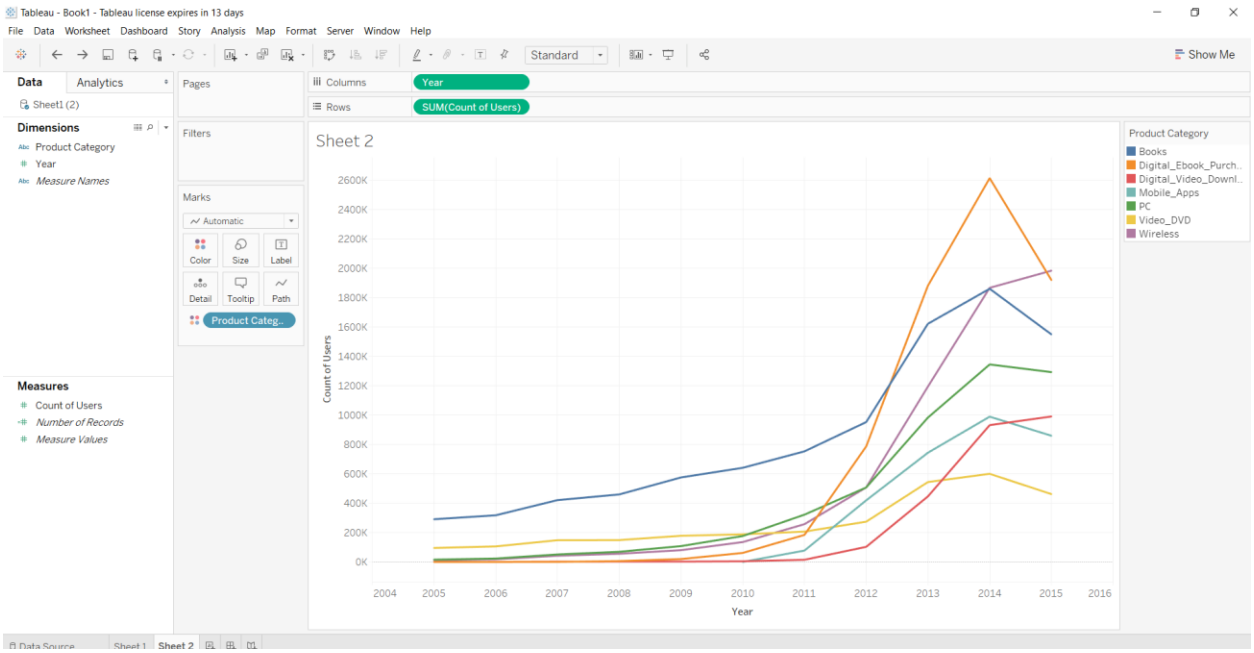
```
#2.Number of users product_category and year wise
df2.groupby("product_category","year")\
    .agg(F.countDistinct("customer_id").alias("Count of Users"))\
    .sort("year")\
    .show(80,truncate = False)
```

```
In [11]: #2.Number of users product_category and year wise
         df2.groupby("product_category","year")\
             .agg(F.countDistinct("customer_id").alias("Count of Users"))\
             .sort("year")\
             .show(80,truncate = False)
```

```
+----------------------+----+--------------+
|product_category      |year|Count of Users|
+----------------------+----+--------------+
|Wireless              |2005|10585         |
|PC                    |2005|15781         |
|Digital_Ebook_Purchase|2005|17            |
|Books                 |2005|290588        |
|Digital_Video_Download|2005|6             |
|Video_DVD             |2005|95199         |
|PC                    |2006|23177         |
|Wireless              |2006|17984         |
|Video_DVD             |2006|105659        |
|Digital_Ebook_Purchase|2006|33            |
|Digital_Video_Download|2006|154           |
|Books                 |2006|317358        |
|Digital_Video_Download|2007|2027          |
|Digital_Ebook_Purchase|2007|407           |
|PC                    |2007|51028         |
|Books                 |2007|420721        |
|Video_DVD             |2007|147934        |
|Wireless              |2007|42100         |
|Books                 |2008|459247        |
|Video_DVD             |2008|148566        |
|Wireless              |2008|55688         |
|PC                    |2008|69068         |
|Digital_Ebook_Purchase|2008|5829          |
|Digital_Video_Download|2008|2456          |
|Digital_Video_Download|2009|2407          |
|Books                 |2009|574958        |
|Video_DVD             |2009|177997        |
|Wireless              |2009|80038         |
|PC                    |2009|107705        |
|Digital_Ebook_Purchase|2009|19741         |
|Digital_Video_Download|2010|4483          |
|Video_DVD             |2010|188039        |
|Books                 |2010|640507        |
|PC                    |2010|176169        |
```



Interpretation: We can see from the above chart that customers buying books are 26% whereas for Ebooks its 20%. This shows there are more number of people who prefer buying books than Ebooks.



Interpretation: We can see that though all the categories are showing a growth in customers but top 3 are Books, Ebooks and Wireless categories. Digital_Ebook_Purchase has observed the maximum growth rate in customers.

3. Average and Median review stars:

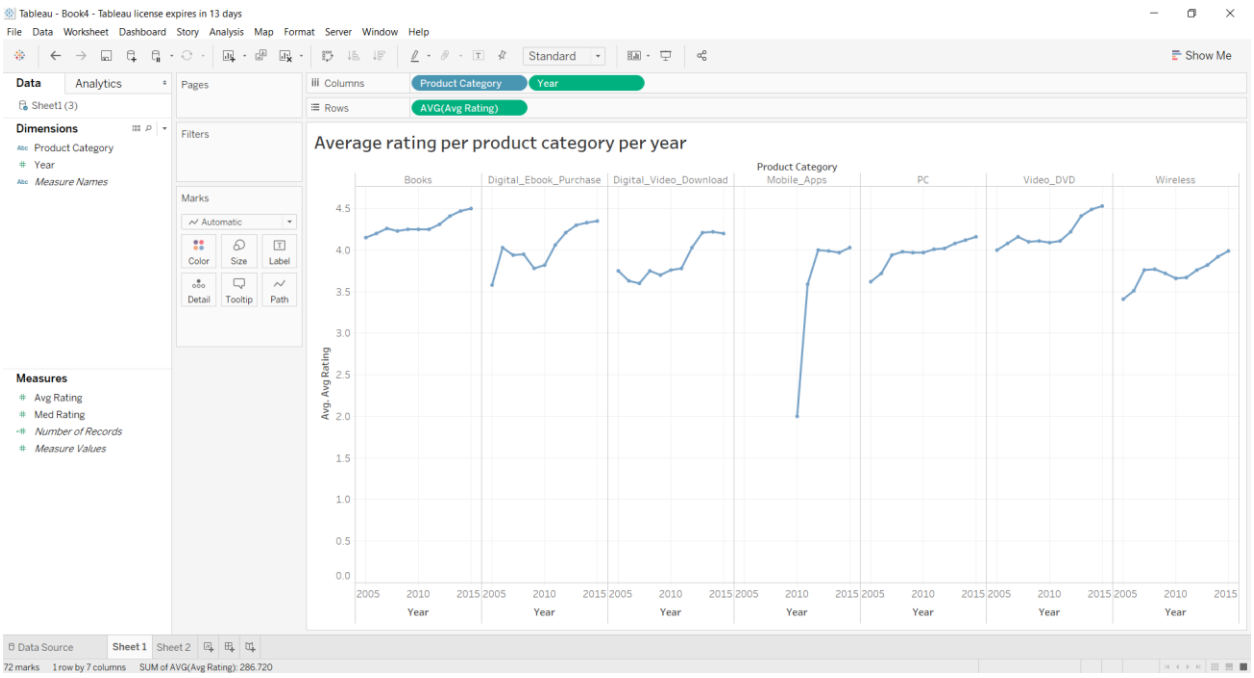
Code:

```
#3.Average and Median review stars product_category wise
df2.groupBy('product_category',"year")\
    .agg(F.round(F.avg("star_rating"),2).alias("avg_rating"),F.expr('percentile_approx(star_rating, 0.5)').alias('med_rating'))\
    .sort("year")\
    .show(80,truncate = False)
```

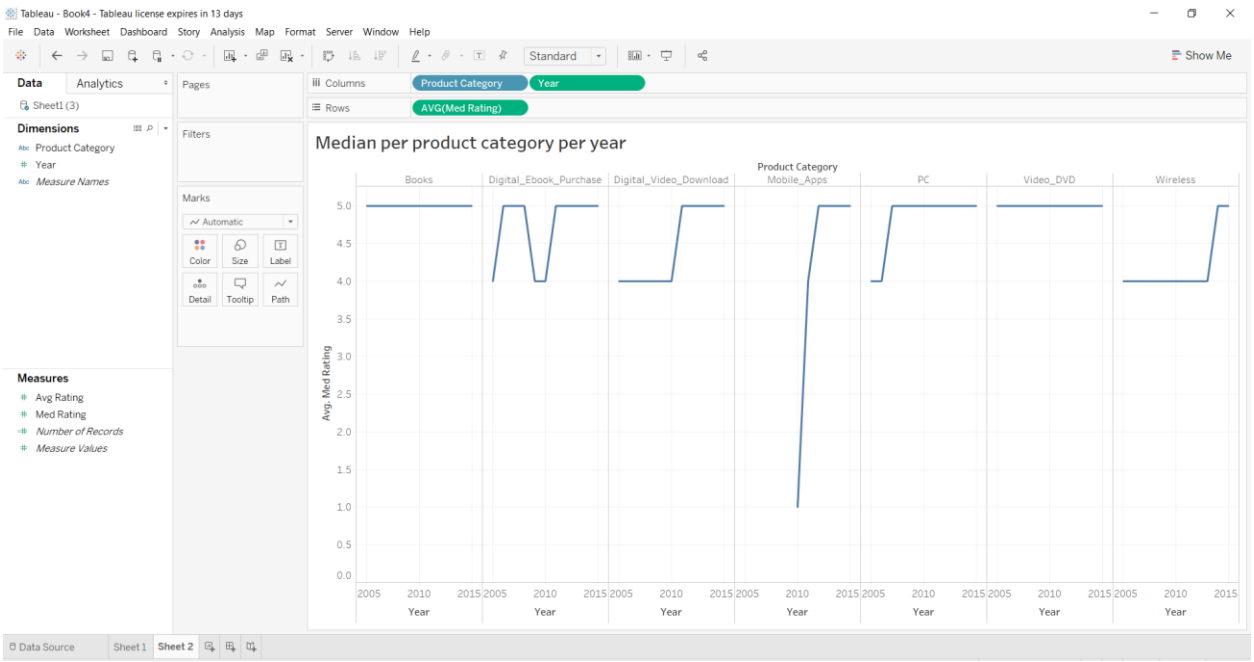
In [16]:

```
#3.Average and Median review stars product_category wise
df2.groupBy('product_category',"year")\
    .agg(F.round(F.avg("star_rating"),2).alias("avg_rating"),F.expr('percentile_approx(star_rating, 0.5)')\
         .alias('med_rating'))\
    .sort("year")\
    .show(80,truncate = False)
```

```
+----------------------+----+----------+----------+
|product_category      |year|avg_rating|med_rating|
+----------------------+----+----------+----------+
|PC                    |2005|3.62      |4         |
|Books                 |2005|4.15      |5         |
|Digital_Video_Download|2005|3.75      |4         |
|Digital_Ebook_Purchase|2005|3.58      |4         |
|Wireless              |2005|3.41      |4         |
|Video_DVD             |2005|4.0       |5         |
|PC                    |2006|3.72      |4         |
|Digital_Ebook_Purchase|2006|4.03      |5         |
|Video_DVD             |2006|4.08      |5         |
|Wireless              |2006|3.51      |4         |
|Digital_Video_Download|2006|3.63      |4         |
|Books                 |2006|4.2       |5         |
|Wireless              |2007|3.76      |4         |
|Video_DVD             |2007|4.16      |5         |
|PC                    |2007|3.94      |5         |
|Books                 |2007|4.26      |5         |
|Digital_Ebook_Purchase|2007|3.94      |5         |
|Digital_Video_Download|2007|3.6       |4         |
|Books                 |2008|4.23      |5         |
|Digital_Ebook_Purchase|2008|3.95      |5         |
|Video_DVD             |2008|4.1       |5         |
|PC                    |2008|3.98      |5         |
|Digital_Video_Download|2008|3.75      |4         |
|Wireless              |2008|3.77      |4         |
|Wireless              |2009|3.72      |4         |
|Digital_Video_Download|2009|3.7       |4         |
|Books                 |2009|4.25      |5         |
|Video_DVD             |2009|4.11      |5         |
|Digital_Ebook_Purchase|2009|3.78      |4         |
|PC                    |2009|3.97      |5         |
|Video_DVD             |2010|4.09      |5         |
|Wireless              |2010|3.66      |4         |
|Books                 |2010|4.25      |5         |
|PC                    |2010|3.97      |5         |
|Digital_Video_Download|2010|3.76      |4         |
```



Interpretation: We can see from the above chart that average rating of all the product categories have increased over the years. The Mobile apps observed the maximum growth rate in average rating and proves that as the technologies improved the customers became more and more satisfied.



Interpretation: We can see from the above chart that median rating of all the product categories have been between 4 and 5 except Mobile Apps. The Mobile apps observed the maximum growth rate in median of rating and proves that as the technologies improved the customers became more and more satisfied and maximum customers started giving higher star rating.

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

Code:-

```
a.#Below code performs percentile of length of the review over the entire dataset:
from pyspark.sql.functions import length
df3=df2.select("*",length("review_body"))
df4=df3.withColumnRenamed("length(review_body)", "review_len")
```

```
In [9]: from pyspark.sql.functions import length
        df3=df2.select("*",length("review_body"))
        df4=df3.withColumnRenamed("length(review_body)", "review_len")

        [64.0, 130.0, 220.0, 812.0, 41200.0, 51019.0]
```

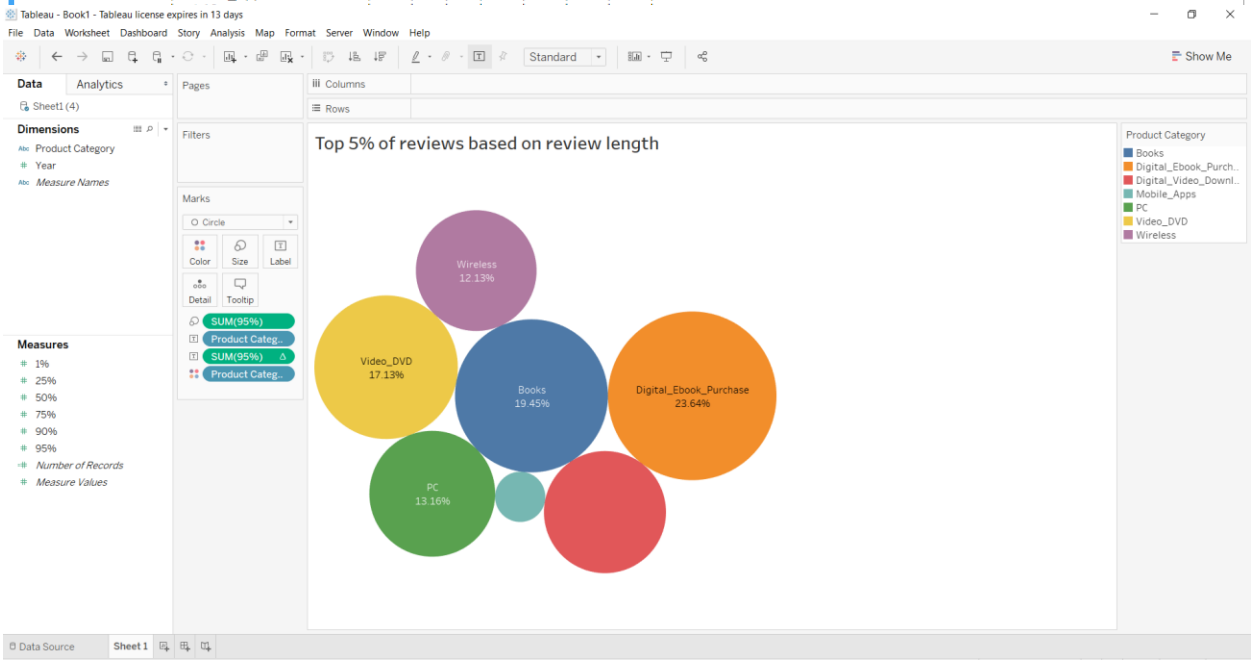
Ans: Seeing the percentile of length of reviews, we can say that only 5% of the reviews have length greater than 51,019 words.

```
b#Below code performs percentile of length of the review over the entire dataset per product category and year wise:
df4.groupby("product_category","year").agg(F.round(F.expr('percentile_approx(Review_Len, 0.1)')).alias('%1'),\
                                           F.round(F.expr('percentile_approx(Review_Len, 0.25)')).alias('%25'),\
                                           F.round(F.expr('percentile_approx(Review_Len, 0.50)')).alias('%50'),\
                                           F.round(F.expr('percentile_approx(Review_Len, 0.75)')).alias('%75'),\
                                           F.round(F.expr('percentile_approx(Review_Len, 0.75)')).alias('%90'),\
                                           F.round(F.expr('percentile_approx(Review_Len, 0.75)')).alias('%95'))\
```

.sort("product_category","year").show(80,truncate=False)

```
In [10]: df4.groupby("product_category","year").agg(F.round(F.expr('percentile_approx(Review_Len, 0.1)')).alias('%1'),\
                                                    F.round(F.expr('percentile_approx(Review_Len, 0.25)')).alias('%25'),\
                                                    F.round(F.expr('percentile_approx(Review_Len, 0.50)')).alias('%50'),\
                                                    F.round(F.expr('percentile_approx(Review_Len, 0.75)')).alias('%75'),\
                                                    F.round(F.expr('percentile_approx(Review_Len, 0.75)')).alias('%90'),\
                                                    F.round(F.expr('percentile_approx(Review_Len, 0.75)')).alias('%95'))\
.sort("product_category","year").show(80,truncate=False)
```

product_category	year	%1	%25	%50	%75	%90	%95
Books	2005	208	367	674	1286	1286	1286
Books	2006	189	339	628	1222	1222	1222
Books	2007	144	258	499	1005	1005	1005
Books	2008	150	267	521	1054	1054	1054
Books	2009	138	249	505	1058	1058	1058
Books	2010	140	251	519	1115	1115	1115
Books	2011	157	259	523	1133	1133	1133
Books	2012	137	208	433	985	985	985
Books	2013	117	145	244	558	558	558
Books	2014	21	71	161	386	386	386
Books	2015	12	35	100	307	307	307
Digital_Ebook_Purchase	2005	139	305	2468	3489	3489	3489
Digital_Ebook_Purchase	2006	179	249	900	2330	2330	2330
Digital_Ebook_Purchase	2007	127	244	502	1181	1181	1181
Digital_Ebook_Purchase	2008	119	225	458	996	996	996
Digital_Ebook_Purchase	2009	136	228	423	826	826	826
Digital_Ebook_Purchase	2010	144	240	450	884	884	884
Digital_Ebook_Purchase	2011	165	260	469	893	893	893
Digital_Ebook_Purchase	2012	123	163	293	596	596	596
Digital_Ebook_Purchase	2013	113	133	198	389	389	389
Digital_Ebook_Purchase	2014	47	113	174	360	360	360
Digital_Ebook_Purchase	2015	23	73	154	340	340	340
Digital_Video_Download	2005	591	656	1134	1483	1483	1483
Digital_Video_Download	2006	107	171	313	602	602	602
Digital_Video_Download	2007	125	197	356	644	644	644
Digital_Video_Download	2008	121	194	346	651	651	651
Digital_Video_Download	2009	126	210	384	703	703	703
Digital_Video_Download	2010	128	209	378	708	708	708
Digital_Video_Download	2011	139	201	360	695	695	695
Digital_Video_Download	2012	115	137	201	386	386	386
Digital_Video_Download	2013	110	124	159	257	257	257
Digital_Video_Download	2014	15	45	118	192	192	192
Digital_Video_Download	2015	10	19	50	119	119	119
Mobile_Apps	2010	71	72	103	148	148	148



Interpretation: The above chart shows top 5% of the reviews based on reviews length. We can say that there is not significant difference in the size of the categories.

5. 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]

Code:-

```
#Below code performs 10,25,50,90 and 95 percentiles over product category and year for review count:
import pyspark.sql.functions as F
df_Q5 = df2.groupBy("product_category","year","product_title").agg(F.count("review_id").alias('review_count'))
df_Q5.groupby("product_category","year").agg(F.round(F.expr('percentile_approx(review_count, 0.1)')).alias('%1'),\
                                             F.round(F.expr('percentile_approx(review_count, 0.25)')).alias('%25'),\
                                             F.round(F.expr('percentile_approx(review_count, 0.50)')).alias('%50'),\
                                             F.round(F.expr('percentile_approx(review_count, 0.75)')).alias('%75'),\
                                             F.round(F.expr('percentile_approx(review_count, 0.75)')).alias('%90'),\
                                             F.round(F.expr('percentile_approx(review_count, 0.75)')).alias('%95'))\
.sort("product_category","year").show(80,truncate=False)
```

In [9]:
```
import pyspark.sql.functions as F
df_Q5 = df2.groupBy("product_category","year","product_title").agg(F.count("review_id").alias('review_count'))
df_Q5.groupby("product_category","year").agg(F.round(F.expr('percentile_approx(review_count, 0.1)')).alias('%1'),\
                                             F.round(F.expr('percentile_approx(review_count, 0.25)')).alias('%25'),\
                                             F.round(F.expr('percentile_approx(review_count, 0.50)')).alias('%50'),\
                                             F.round(F.expr('percentile_approx(review_count, 0.75)')).alias('%75'),\
                                             F.round(F.expr('percentile_approx(review_count, 0.75)')).alias('%90'),\
                                             F.round(F.expr('percentile_approx(review_count, 0.75)')).alias('%95'))\
.sort("product_category","year").show(80,truncate=False)
```

product_category	year	%1	%25	%50	%75	%90	%95
Books	2005	1	1	1	2	2	2
Books	2006	1	1	1	2	2	2
Books	2007	1	1	1	2	2	2
Books	2008	1	1	1	2	2	2
Books	2009	1	1	1	2	2	2
Books	2010	1	1	1	2	2	2
Books	2011	1	1	1	2	2	2
Books	2012	1	1	1	2	2	2
Books	2013	1	1	1	3	3	3
Books	2014	1	1	1	3	3	3
Books	2015	1	1	1	2	2	2
Digital_Ebook_Purchase	2005	1	1	1	1	1	1
Digital_Ebook_Purchase	2006	1	1	1	1	1	1
Digital_Ebook_Purchase	2007	1	1	1	1	1	1
Digital_Ebook_Purchase	2008	1	1	1	1	1	1
Digital_Ebook_Purchase	2009	1	1	1	2	2	2
Digital_Ebook_Purchase	2010	1	1	1	2	2	2
Digital_Ebook_Purchase	2011	1	1	1	3	3	3
Digital_Ebook_Purchase	2012	1	1	2	4	4	4
Digital_Ebook_Purchase	2013	1	1	2	5	5	5
Digital_Ebook_Purchase	2014	1	1	2	5	5	5
Digital_Ebook_Purchase	2015	1	1	2	4	4	4
Digital_Video_Download	2005	1	1	1	1	1	1
Digital_Video_Download	2006	1	1	1	1	1	1
Digital_Video_Download	2007	1	1	1	2	2	2
Digital_Video_Download	2008	1	1	1	2	2	2
Digital_Video_Download	2009	1	1	1	2	2	2
Digital_Video_Download	2010	1	1	1	2	2	2
Digital_Video_Download	2011	1	1	1	2	2	2
Digital_Video_Download	2012	1	1	2	3	3	3
Digital_Video_Download	2013	1	1	2	7	7	7
Digital_Video_Download	2014	1	1	3	10	10	10

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

Code:-

```
df4=
df2.select("product_category","year","review_date",'star_rating').withColumn('week',F.weekofyear(df2.review_date))\
      .where(F.column("star_rating").isin([4,5]))
df6 = df4.groupby("product_category","year","week")\
  .agg(F.count("star_rating").alias("rating_count"))\
  .sort("product_category","year")
from pyspark.sql.window import Window
import pyspark.sql.functions as F
df7 =
df6.select("*",F.rank().over(Window.partitionBy("product_category",'year').orderBy(F.col('rating_count').desc())))
df8=df7.withColumnRenamed("RANK() OVER (PARTITION BY product_category, year ORDER BY rating_count DESC NULLS LAST unspecifiedframe$())",'rank')
df8.where(F.col("rank")==1).show(80,truncate = False)
```

In [8]:
```
#df4 is a dataframe with coloumns "product_category","year","review_date",'star_rating' and 'week'
df4= df2.select("product_category","year","review_date",'star_rating').withColumn('week',F.weekofyear(df2.review_date))\
        .where(F.column("star_rating").isin([4,5]))
```

In [9]:
```
df4.count()
```

```
52537357
```

In [10]:
```
df4.show(5,truncate=False)
```

```
+----------------+----+-----------+-----------+----+
|product_category|year|review_date|star_rating|week|
+----------------+----+-----------+-----------+----+
|Books           |2011|2011-10-30 |5          |43  |
|Books           |2014|2014-05-02 |5          |18  |
|Books           |2008|2008-03-24 |5          |13  |
|Books           |2010|2010-05-08 |5          |18  |
|Books           |2013|2013-08-12 |5          |33  |
+----------------+----+-----------+-----------+----+
only showing top 5 rows
```

In [11]:
```
df6 = df4.groupby("product_category","year","week")\
   .agg(F.count("star_rating").alias("rating_count"))\
   .sort("product_category","year")
```

In [12]:
```
df6.show(5,truncate=False)
```

```
+----------------+----+----+------------+
|product_category|year|week|rating_count|
+----------------+----+----+------------+
|Books           |2005|17  |6663        |
|Books           |2005|5   |6632        |
|Books           |2005|21  |6249        |
|Books           |2005|34  |9774        |
|Books           |2005|12  |6298        |
+----------------+----+----+------------+
only showing top 5 rows
```

In [13]:
```
from pyspark.sql.window import Window
import pyspark.sql.functions as F

df7 = df6.select("*",F.rank().over(Window.partitionBy("product_category",'year').orderBy(F.col('rating_count').desc())))
```

In [14]:
```
df8=df7.withColumnRenamed("RANK() OVER (PARTITION BY product_category, year ORDER BY rating_count\
                           DESC NULLS LAST unspecifiedframe$())",'rank')
df8.where(F.col("rank")==1).show(20,truncate = False)
```

```
+----------------------+----+----+------------+----+
|product_category      |year|week|rating_count|rank|
+----------------------+----+----+------------+----+
|Books                 |2005|31  |11399       |1   |
|Books                 |2006|45  |15769       |1   |
|Books                 |2007|2   |29867       |1   |
|Books                 |2008|1   |16441       |1   |
|Books                 |2009|37  |21677       |1   |
|Books                 |2010|5   |24043       |1   |
|Books                 |2011|52  |28267       |1   |
|Books                 |2012|52  |62085       |1   |
|Books                 |2013|1   |73910       |1   |
|Books                 |2014|1   |86638       |1   |
|Books                 |2015|2   |95507       |1   |
|Digital_Ebook_Purchase|2005|43  |3           |1   |
|Digital_Ebook_Purchase|2006|31  |3           |1   |
|Digital_Ebook_Purchase|2006|34  |3           |1   |
|Digital_Ebook_Purchase|2006|8   |3           |1   |
|Digital_Ebook_Purchase|2007|48  |78          |1   |
|Digital_Ebook_Purchase|2008|14  |250         |1   |
|Digital_Ebook_Purchase|2009|52  |712         |1   |
|Digital_Ebook_Purchase|2010|52  |2873        |1   |
|Digital_Ebook_Purchase|2011|52  |9745        |1   |
+----------------------+----+----+------------+----+
only showing top 20 rows
```

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

1. Using Spark Pivot functionality, produce DataFrame with following columns:

1. Year

2. Month

3. Total number of reviews for "Digital eBook Purchase" category

4. Total number of reviews for "Books" category

5. Average stars for reviews for "Digital eBook Purchase" category
6. Average stars for reviews for "Books" category

Code:-

```
df10 = df2.withColumn('month',F.month(df2.review_date))\
        .select("product_category","review_id","star_rating","year","month")
df10.show(10, False)
```

```
In [22]: df10 = df2.withColumn('month',F.month(df2.review_date))\
                .select("product_category","review_id","star_rating","year","month")
         df10.show(10, False)
```

```
+----------------+--------------+-----------+----+-----+
|product_category|review_id     |star_rating|year|month|
+----------------+--------------+-----------+----+-----+
|Books           |R1MKP6SU9MAHWT|5          |2011|10   |
|Books           |R1YB0J8DC0TKS2|5          |2014|5    |
|Books           |R2IEM1VVAHVQNG|5          |2008|3    |
|Books           |R20L65O2S58MAO|5          |2010|5    |
|Books           |R38P8K3BI379KU|5          |2013|8    |
|Books           |RS7TO93OWV5VA |5          |2013|12   |
|Books           |RJE8D9YSE0TEX |5          |2008|6    |
|Books           |R17HQIAO8MSGNZ|5          |2012|11   |
|Books           |R2M6061U2N92VD|5          |2014|5    |
|Books           |R3RWT30V27AWRZ|2          |2013|7    |
+----------------+--------------+-----------+----+-----+
only showing top 10 rows
```

```
category_to_filter=["Digital_Ebook_Purchase","Books"]
Q21=df10.groupBy("Year","Month").pivot("product_category",category_to_filter)\

.agg(F.count("review_id").alias("review_count"),F.round(F.mean("star_rating"),3).alias("Avg_rating"))\
   .sort("Year","Month", ascending=False)
Q21.show(132,truncate = False)
Q21=Q21.na.fill(0)
Q21.show(132,truncate = False)
```

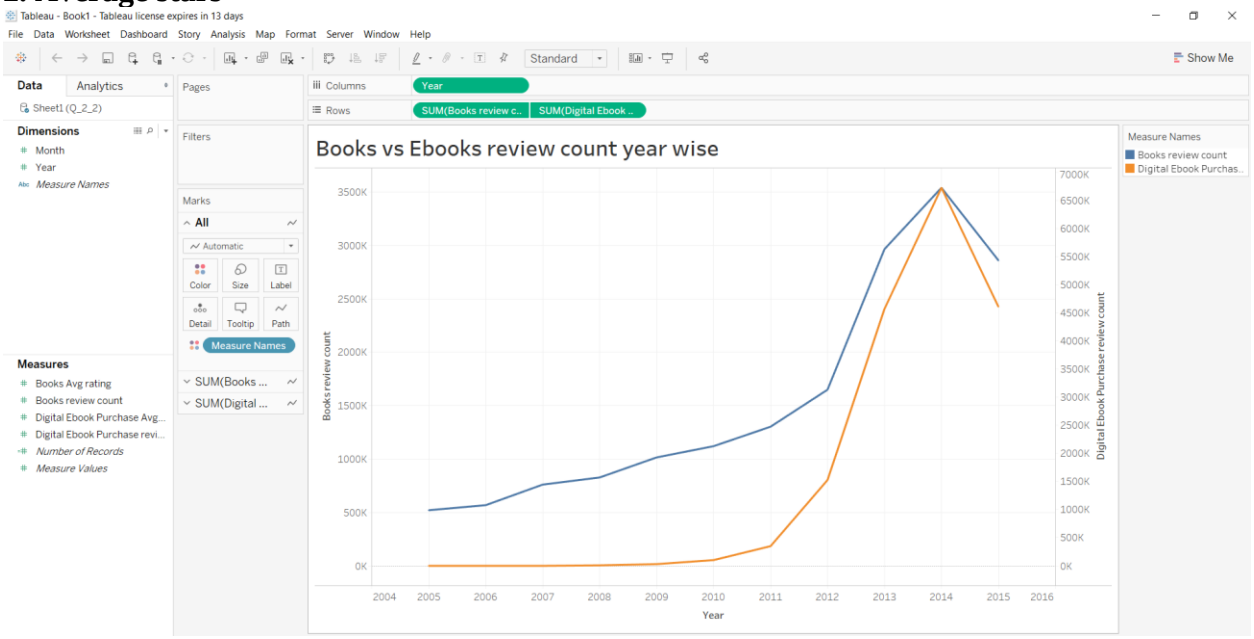
```
In [24]: category_to_filter=["Digital_Ebook_Purchase","Books"]
         Q21=df10.groupBy("Year","Month").pivot("product_category",category_to_filter)\
             .agg(F.count("review_id").alias("review_count"),F.round(F.mean("star_rating"),3).alias("Avg_rating"))\
             .sort("Year","Month", ascending=False)
         Q21.show(132,truncate = False)
         Q21=Q21.na.fill(0)
         Q21.show(132,truncate = False)
```

```
+----+-----+-----------------------------------+-------------------------------+-----------------+---------------+
|Year|Month|Digital_Ebook_Purchase_review_count|Digital_Ebook_Purchase_Avg_rating|Books_review_count|Books_Avg_rating|
+----+-----+-----------------------------------+-------------------------------+-----------------+---------------+
|2015|8    |578604                             |4.338                          |347646           |4.48           |
|2015|7    |564277                             |4.341                          |339311           |4.477          |
|2015|6    |534045                             |4.35                           |326946           |4.486          |
|2015|5    |581539                             |4.343                          |327538           |4.49           |
|2015|4    |597905                             |4.354                          |336509           |4.498          |
|2015|3    |695166                             |4.354                          |387025           |4.507          |
|2015|2    |538418                             |4.358                          |380738           |4.51           |
|2015|1    |519404                             |4.36                           |414950           |4.522          |
|2014|12   |621821                             |4.335                          |383955           |4.514          |
|2014|11   |574097                             |4.343                          |311702           |4.498          |
|2014|10   |616969                             |4.342                          |343254           |4.499          |
|2014|9    |547985                             |4.338                          |338125           |4.49           |
|2014|8    |635120                             |4.32                           |341036           |4.491          |
|2014|7    |633503                             |4.323                          |337097           |4.493          |
|2014|6    |515230                             |4.333                          |222120           |4.438          |
|2014|5    |520153                             |4.337                          |220739           |4.445          |
```

Interpretation: I have used sparks pivot functionality on Digital_Ebook_Purchase and Books product categories to show average star rating and average review count per month per year.

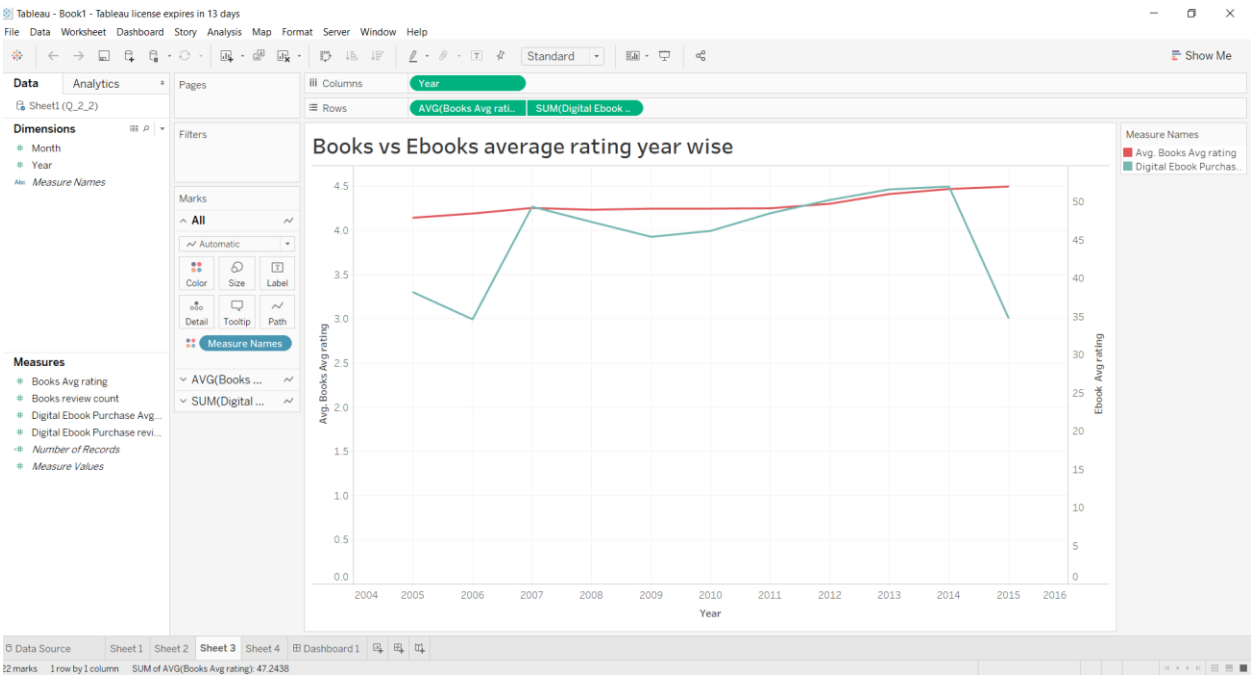
2. Produce two graphs to demonstrate aggregations from #1:
1. Number of reviews

2. Average stars



Interpretation: The above graph between Books and Digital Books review count per year shows that there has been rapid growth in purchase of products in both categories. For Digital books we can see that between 2005 to 2010 there was almost no to very less purchase. In 2014 we can see that both the categories are sharing the same peak. Thus, we can say that there has been a higher growth rate in Digital books than Books category.

Interpretation: The above graph between Books and Digital Books review count per month shows that there has been a higher purchase from months between March and August.



Interpretation: The above graph between Books and Digital Books average rating per year shows that there for Books category the average rating is between 4 and 4.5, whereas for Digital books the average rating started from 3-3.5 during 2004 to 2006 and increased to 4 afterwards.

3. 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.

Code:-

```
#Q2.3
#df12 will have df2 dataset with condition product_category is "Digital_Ebook_Purchase" or "Books" and star_rating is 4 or 5
category_to_filter=["Digital_Ebook_Purchase","Books"]
Rating=[4,5]
df12=df2.select("product_category","product_title","star_rating")\
   .filter((F.col("product_category").isin(category_to_filter)) & (F.col("star_rating").isin(Rating)))
#Q231a is a dataframe with product_category Digital_Ebook_Purchase
Q231a= df12.groupby("product_category", F.lower(F.trim(F.col("product_title"))).alias("product_title"))\
        .agg(F.round(F.avg("star_rating"),2).alias("Ebook_Avg_rating"))\
        .filter(F.col("product_category")=="Digital_Ebook_Purchase")
#Q231b is a dataframe with product_category Books
Q231b= df12.groupby("product_category", F.lower(F.trim(F.col("product_title"))).alias("product_title"))\
        .agg(F.round(F.avg("star_rating"),2).alias("book_Avg_rating"))\
        .filter(F.col("product_category")=="Books")
#Q231c is a dataframe created by join Q231a and Q231b where product title is same
```

Q231c=Q231a.join(Q231b, (Q231a["product_title"] == Q231b['product_title'])).drop(Q231a["product_title"])
Q231c.show(10)

```
In [8]:
#Q2.3
#df12 will have df2 dataset with condition product_category is "Digital_Ebook_Purchase" or "Books" and star_rating is 4 or 5

category_to_filter=["Digital_Ebook_Purchase","Books"]
Rating=[4,5]

df12=df2.select("product_category","product_title","star_rating")\
    .filter((F.col("product_category").isin(category_to_filter)) & (F.col("star_rating").isin(Rating)))

#Q231a is a dataframe with product_category Digital_Ebook_Purchase
Q231a= df12.groupby("product_category",  F.lower(F.trim(F.col("product_title"))).alias("product_title"))\
          .agg(F.round(F.avg("star_rating"),2).alias("Ebook_Avg_rating"))\
          .filter(F.col("product_category")=="Digital_Ebook_Purchase")

#Q231b is a dataframe with product_category Books
Q231b= df12.groupby("product_category",  F.lower(F.trim(F.col("product_title"))).alias("product_title"))\
          .agg(F.round(F.avg("star_rating"),2).alias("book_Avg_rating"))\
          .filter(F.col("product_category")=="Books")
#Q231c is a dataframe created by join Q231a and Q231b where product title is same
Q231c=Q231a.join(Q231b, (Q231a["product_title"] == Q231b['product_title'])).drop(Q231a["product_title"])

Q231c.show(10)
```

```
+--------------------+---------------+---------------+--------------------+--------------+
|    product_category|Ebook_Avg_rating|product_category|       product_title|book_Avg_rating|
+--------------------+---------------+---------------+--------------------+--------------+
|Digital_Ebook_Pur...|            5.0|          Books|"rays of light": ...|           5.0|
|Digital_Ebook_Pur...|           4.99|          Books|"the siege of khe...|          4.94|
|Digital_Ebook_Pur...|            5.0|          Books|          'dem bon'z|           5.0|
|Digital_Ebook_Pur...|           4.67|          Books|   0400 roswell time|           5.0|
|Digital_Ebook_Pur...|           4.83|          Books|10 smart things g...|           5.0|
|Digital_Ebook_Pur...|            5.0|          Books|100 prayers for y...|           5.0|
|Digital_Ebook_Pur...|            5.0|          Books|13 cent killers: ...|          4.81|
|Digital_Ebook_Pur...|            5.0|          Books|25 essentials: te...|          4.72|
|Digital_Ebook_Pur...|           4.97|          Books|30 before 30: tra...|           5.0|
|Digital_Ebook_Pur...|            5.0|          Books|42 rules to incre...|           5.0|
+--------------------+---------------+---------------+--------------------+--------------+
only showing top 10 rows
```

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

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.

Code:-

Q231c.stat.corr("Ebook_Avg_rating", "book_Avg_rating")

```
In [9]:
Q231c.stat.corr("Ebook_Avg_rating", "book_Avg_rating")
```

```
0.16606580243319788
```

Interpretation: We have performed correlation between Ebooks and Books average rating for each product title to find any significant value. But the correlation of 0.166 indicates there is no significant correlation between them.

4. Using provided LDA starter notebook, perform LDA topic modeling for the reviews in Digital_Ebook_Purchase and Books categories.

1. Perform LDA separately for reviews with 1/2 stars and reviews with 4/5 stars.

2. Add stop words to the standard list as needed. In the example notebook, you can see some words like 34, br, p appear in the topics.

3. Identify 5 top topics for each case (1/2 versus 4/5)

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

Imported 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
```

Part1:-

Loading Dataframes with product category Digital_Ebook_Purchase and Books and Rating 4 or 5:

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

Creating Dataframe with only narrative and unique ID:

```
#from pyspark.sql.functions import monotonically_increasing_id, concat

df1 = df_ml.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()
```

Persisting and finding the size of the dataset:

```
corpus_df.persist()
```

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

```
Corpus size: 466817
+--------------------+----------+
|         review_text|        id|
+--------------------+----------+
|Great story! Ray ...|8589934592|
|Four Stars It's a...|8589934593|
|Four Stars Good r...|8589934594|
|Really enjoyed th...|8589934595|
|Five Stars Amazin...|8589934596|
+--------------------+----------+
only showing top 5 rows
```

Tokenizing narrative text:

```
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|
+--------------------+--------------------+------+
|Great story! Ray ...|[great, story, ra...|    29|
|Four Stars It's a...|[four, stars, it,...|    17|
|Four Stars Good r...|[four, stars, goo...|     6|
|Really enjoyed th...|[really, enjoyed,...|    29|
|Five Stars Amazin...|[five, stars, ama...|     7|
|Five Stars excell...|[five, stars, exc...|     4|
|Good read The las...|[good, read, the,...|    24|
|Five Stars Great ...|[five, stars, gre...|     8|
|Five Stars Best h...|[five, stars, bes...|     8|
|Loved it I read e...|[loved, it, i, re...|    17|
|Four Stars Easy r...|[four, stars, eas...|    11|
|wow I Should have...|[wow, i, should, ...|    23|
|Five Stars If you...|[five, stars, if,...|    14|
|Really Great Read...|[really, great, r...|    20|
|Inspirational. Fo...|[inspirational, f...|    53|
|Five Stars This i...|[five, stars, thi...|    13|
|Good read! Very g...|[good, read, very...|     5|
|Five Stars Awesom...|[five, stars, awe...|     4|
|stunning ending I...|[stunning, ending...|    35|
|Five Stars Great ...|[five, stars, gre...|     4|
+--------------------+--------------------+------+
only showing top 20 rows
```

Making 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']
```

Removing stop words from the tokens:

```
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|
+--------------------+----------+--------------------+--------------------+
|Great story! Ray ...|8589934592|[great, story, ra...|[great, story, ra...|
|Four Stars It's a...|8589934593|[four, stars, it,...|[four, stars, gre...|
|Four Stars Good r...|8589934594|[four, stars, goo...|[four, stars, goo...|
|Really enjoyed th...|8589934595|[really, enjoyed,...|[really, enjoyed,...|
|Five Stars Amazin...|8589934596|[five, stars, ama...|[five, stars, ama...|
+--------------------+----------+--------------------+--------------------+
only showing top 5 rows
```

```
+--------------------+----------+--------------------+--------------------+--------------------+
|         review_text|        id|               words|            filtered|       filtered_more|
+--------------------+----------+--------------------+--------------------+--------------------+
|Great story! Ray ...|8589934592|[great, story, ra...|[great, story, ra...|[great, story, ra...|
|Four Stars It's a...|8589934593|[four, stars, it,...|[four, stars, gre...|[stars, great, ge...|
|Four Stars Good r...|8589934594|[four, stars, goo...|[four, stars, goo...|[stars, good, rea...|
|Really enjoyed th...|8589934595|[really, enjoyed,...|[really, enjoyed,...|[really, enjoyed,...|
|Five Stars Amazin...|8589934596|[five, stars, ama...|[five, stars, ama...|[stars, amazing, ...|
+--------------------+----------+--------------------+--------------------+--------------------+
only showing top 5 rows
```

Vectorizing(converting the words into numbers)

```
# Term Frequency Vectorization  - Option 2 (CountVectorizer)    :
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)
```

```
+--------------------+--------------------+----------+
|       filtered_more|            features|        id|
+--------------------+--------------------+----------+
|[great, story, ra...|(10000,[1,2,6,7,2...|8589934592|
|[stars, great, ge...|(10000,[0,2,9,32,...|8589934593|
|[stars, good, rea...|(10000,[0,4,9,137...|8589934594|
|[really, enjoyed,...|(10000,[7,10,11,1...|8589934595|
|[stars, amazing, ...|(10000,[0,9,29,97...|8589934596|
+--------------------+--------------------+----------+
only showing top 5 rows
```

Making the dataframe to train LDA model

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

Taining LDA Model:

```
#k=10 means 10 words per topic
lda = LDA(k=5, maxIter=10)
model = lda.fit(countVectors)
```

Displaying words for top 5 topics

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

```
like
----------
topic:  3
----------
story
love
read
life
like
way
time
really
family
know
----------
topic:  4
----------
read
great
good
like
author
reading
story
time
books
people
----------
```

Part2:-

Loading Dataframes with product category Digital_Ebook_Purchase and Books and Rating 1 or 2 :

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

Creating Dataframe with only narrative and unique ID:

```
#from pyspark.sql.functions import monotonically_increasing_id, concat

df1 = df_ml.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()
```

Persisting and finding the size of the dataset:

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

```
Corpus size: 466817
+--------------------+----------+
|         review_text|        id|
+--------------------+----------+
|Great story! Ray ...|8589934592|
|Four Stars It's a...|8589934593|
|Four Stars Good r...|8589934594|
|Really enjoyed th...|8589934595|
|Five Stars Amazin...|8589934596|
+--------------------+----------+
only showing top 5 rows
```

Tokenizing narrative text:

```
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|
+--------------------+--------------------+------+
|Great story! Ray ...|[great, story, ra...|    29|
|Four Stars It's a...|[four, stars, it,...|    17|
|Four Stars Good r...|[four, stars, goo...|     6|
|Really enjoyed th...|[really, enjoyed,...|    29|
|Five Stars Amazin...|[five, stars, ama...|     7|
|Five Stars excell...|[five, stars, exc...|     4|
|Good read The las...|[good, read, the,...|    24|
|Five Stars Great ...|[five, stars, gre...|     8|
|Five Stars Best h...|[five, stars, bes...|     8|
|Loved it I read e...|[loved, it, i, re...|    17|
|Four Stars Easy r...|[four, stars, eas...|    11|
|wow I Should have...|[wow, i, should, ...|    23|
|Five Stars If you...|[five, stars, if,...|    14|
|Really Great Read...|[really, great, r...|    20|
|Inspirational. Fo...|[inspirational, f...|    53|
|Five Stars This i...|[five, stars, thi...|    13|
|Good read! Very g...|[good, read, very...|     5|
|Five Stars Awesom...|[five, stars, awe...|     4|
|stunning ending I...|[stunning, ending...|    35|
|Five Stars Great ...|[five, stars, gre...|     4|
+--------------------+--------------------+------+
only showing top 20 rows
```

Making 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']
```

Removing stop words from the tokens:

```
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=stop
wordList)
tokenized_df2 = remover.transform(tokenized_df1)
tokenized_df2.show(5)
```

```
+--------------------+----------+--------------------+--------------------+
|         review_text|        id|               words|            filtered|
+--------------------+----------+--------------------+--------------------+
|Great story! Ray ...|8589934592|[great, story, ra...|[great, story, ra...|
|Four Stars It's a...|8589934593|[four, stars, it,...|[four, stars, gre...|
|Four Stars Good r...|8589934594|[four, stars, goo...|[four, stars, goo...|
|Really enjoyed th...|8589934595|[really, enjoyed,...|[really, enjoyed,...|
|Five Stars Amazin...|8589934596|[five, stars, ama...|[five, stars, ama...|
+--------------------+----------+--------------------+--------------------+
only showing top 5 rows

+--------------------+----------+--------------------+--------------------+--------------------+
|         review_text|        id|               words|            filtered|       filtered_more|
+--------------------+----------+--------------------+--------------------+--------------------+
|Great story! Ray ...|8589934592|[great, story, ra...|[great, story, ra...|[great, story, ra...|
|Four Stars It's a...|8589934593|[four, stars, it,...|[four, stars, gre...|[stars, great, ge...|
|Four Stars Good r...|8589934594|[four, stars, goo...|[four, stars, goo...|[stars, good, rea...|
|Really enjoyed th...|8589934595|[really, enjoyed,...|[really, enjoyed,...|[really, enjoyed,...|
|Five Stars Amazin...|8589934596|[five, stars, ama...|[five, stars, ama...|[stars, amazing, ...|
+--------------------+----------+--------------------+--------------------+--------------------+
only showing top 5 rows
```

Vectorizing(converting the words into numbers)

```
# Term Frequency Vectorization  - Option 2 (CountVectorizer)    :
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)
```

```
+--------------------+--------------------+----------+
|       filtered_more|            features|        id|
+--------------------+--------------------+----------+
|[great, story, ra...|(10000,[1,2,6,7,2...|8589934592|
|[stars, great, ge...|(10000,[0,2,9,32,...|8589934593|
|[stars, good, rea...|(10000,[0,4,9,137...|8589934594|
|[really, enjoyed,...|(10000,[7,10,11,1...|8589934595|
|[stars, amazing, ...|(10000,[0,9,29,97...|8589934596|
+--------------------+--------------------+----------+
only showing top 5 rows
```

Making the dataframe to train LDA model

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

Taining LDA Model:

```
#k=10 means 10 words per topic
lda = LDA(k=5, maxIter=10)
```

```
model = lda.fit(countVectors)
```

Displaying words for top 5 topics

```
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
love
read
characters
series
good
great
author
loved
reading
----------
topic:  1
----------
read
good
story
great
really
like
love
stars
time
didn
----------
```

topic: 2
---
read
great
stars
books
series
story
reading
like
characters
loved
---
topic: 3
---
story
love
read
life
like
way
family
time
characters
written
---
topic: 4
---
read
good
great
author
like
reading
story
time
books
people

Interpretation: We have performed LDA for both low and high ratings but haven't observed any significant difference between the reviews. Some of the common topics are good, great, read, love and like, though we expected for bad rating the top words will be bad, verybad. Thus, we can say that LDA is not suitable for this review dataset.

Conclusion:

After handling the big data and performing exploratory analysis on the Amazon review dataset, I came out with various hidden patterns. The features I focused on are review count, product id, star rating and reviews. Observing them I found that only 2 categories i.e. Books and Digital Ebooks Purchase category constitutes more than 53% of the dataset. All the product categories are showing a growth in number of customers, but Digital Ebook Purchase has observed the maximum growth rate followed by Books and Wireless categories. There is a growth in average rating for all product categories but the Mobile apps experienced the maximum growth rate in average rating. A similar pattern is shown while observing the medians of the product categories. The main reason behind rapid increase in average and median rating of Mobile Apps might be improve in technologies, making apps more user friendly with no bugs. I used various sparks functionalities like percentiles and pivots and found interesting information like there are only 5% of the reviews have length greater than 51,019 words. One might think that as the as technologies improving, people are shifting from paper to online, but the data shows the other way there are still more number of people who prefer buying books than Ebooks, though there is close competition between them. From 2005 to 2010 there was almost no to very less purchase for Digital ebooks but in 2014 both the categories are sharing the same peak. Thus, we can say that there has been a higher growth rate in Digital books than Books category. On observing review count per month between Books and Digital Books I found that there is a higher purchase between March and August. I also performed correlation test between Digital Ebooks Purchase and Books average rating for each product title to find any significant value, but the value of 0.166 indicates there is no a significant correlation between them. I performed a LDA model for both low and high ratings for top 2 categories to find any significant pattern, but haven't observed any significant difference between the reviews. Some of the common topic words are good, great, read, love and like, though I expected for bad rating the top words will be bad, very bad. Thus, I can say that LDA is not suitable for this review dataset.

References:-

https://searchbusinessanalytics.techtarget.com/definition/big-data-analytics

https://www.ibm.com/analytics/hadoop/big-data-analytics

https://aws.amazon.com/emr/faqs/