Project Part 2
SHRESHTHA JHA

Commands to load relevant data and relevant product_categories in Hdfs.

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=Wireless/

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

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/

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=Wireless/ \
--dest=hdfs:///hive/amazon-reviews-pds/parquet/product_category=Wireless/

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/

To load data in dataframe and filter reviews that are after year 2004

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

Excluding data before 2005

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

To filter multiple reviews by same user on same product

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

```
Filter = temp.rownum.isin(1)
filtered=temp.where(Filter)
filtered.persist()
```

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

1.1 Number of reviews

```
2. filtered.groupby("year","product_category").agg(F.countDistinct("rev
   iew_id").alias('Number_of_Review')).show(5)
```

output-

```
+------+--------------------------------------------------+----------------------------------------+
|year|     product_category|Number_of_Review|
```

```
+------+--------------------+------------------+
|2014|               Books|           3540828|
|2010|Digital_Ebook_Pur...|            102515|
|2015|               Books|           2860743|
|2013|            Wireless|           1767132|
|2014|         Mobile_Apps|           1728280|
+------+--------------------+------------------+

only showing top 5 rows
```

1.2 Number of distinct users

```
filtered.groupby("year","product_category").agg(F.countDistinct("customer_
id").alias('Number_of_distinct_Users')) \
.sort("year",ascending=True).show(10)
```

Output-

```
+------+--------------------+------------------------+
|year|    product_category|Number_of_distinct_Users|
+------+--------------------+------------------------+
|2005|            Wireless|                   10585|
|2005|Digital_Ebook_Pur...|                      17|
|2005|           Video_DVD|                   95195|
|2005|Digital_Video_Dow...|                       6|
|2005|               Books|                  290584|
|2005|                  PC|                   15780|
|2006|            Wireless|                   17984|
|2006|Digital_Ebook_Pur...|                      33|
|2006|           Video_DVD|                  105660|
|2006|                  PC|                   23174|
+------+--------------------+------------------------+
only showing top 10 rows
```

Inference- we have calculated number of distinct users per year starting from 20 05 per product category. We can also calculate highest and lowest number of dis tinct users that reviewed the products each year.

1.3. Average and Median review stars

```
filtered.groupby("year","product_category").agg(F.avg("star_rating").alias
('Average-Ratings'),
                                                F.expr('percentile_approx(star_ra
ting,0.5)').alias('Median-Ratings')) \
.sort("year","product category",ascending=True).show()
```

Output-

```
+----+--------------------+------------------+--------------+
|year|    product_category|   Average-Ratings|Median-Ratings|
+----+--------------------+------------------+--------------+
|2005|               Books| 4.148046708559013|             5|
|2005|Digital_Ebook_Pur...|3.5789473684210527|             4|
|2005|Digital_Video_Dow...|              3.75|             4|
|2005|                  PC| 3.616240022020369|             4|
|2005|           Video_DVD| 4.004813687570013|             5|
|2005|            Wireless| 3.414026193493874|             4|
|2006|               Books| 4.196543242891375|             5|
|2006|Digital_Ebook_Pur...| 4.027777777777778|             5|
|2006|Digital_Video_Dow...|3.6324324324324326|             4|
|2006|                  PC| 3.717627814972611|             4|
|2006|           Video_DVD|4.0803634706894805|             5|
|2006|            Wireless|3.5095432341239867|             4|
|2007|               Books| 4.258164000084097|             5|
|2007|Digital_Ebook_Pur...| 3.938976377952756|             5|
|2007|Digital_Video_Dow...|  3.59992298806315|             4|
|2007|                  PC|3.9428256549835523|             5|
|2007|           Video_DVD| 4.160460744563214|             5|
|2007|            Wireless| 3.761102731690967|             4|
|2008|               Books| 4.233264063406147|             5|
|2008|Digital_Ebook_Pur...| 3.945872801082544|             5|
+----+--------------------+------------------+--------------+

only showing top 20 rows
```

Inference- For some of records we can see that there is large difference betwee n Average ratings and median ratings, so we can say that for such product categ ory ratings are skewed on both sides.

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

```
from pyspark.sql.functions import stddev_pop,min,max,length,count,mean
dataframeL1=filtered.withColumn('length',length(df.review_body))
dataframeL2=dataframeL1.groupby("year","product_category").agg(F.avg("leng
th").alias('average_of_Reviews'))
columnName = "average_of_Reviews"
quantileProbs = [0.1, 0.25, 0.5, 0.75, 0.9, 0.95]
Error = 0.01
dataframeL2.stat.approxQuantile("average_of_Reviews",quantileProbs,Error)
```

Output-

```
[188.95367161124744, 349.1557032255313, 586.5676289328576, 845.37431319125
8, 961.9873203920449, 1170.0692938515842]
```

Inference- We can see that median length of review is 586.56 and lowest 10 th percentile length of review is 188.95 while highest 90 th percentile cutoff is 961.98

1.5. 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]

```
from pyspark.sql.functions import stddev_pop, min, max,length,count, mean
df1=filtered.groupby("year","product_id","product_category").agg(F.countDi
stinct("review_id").alias('Number_of_Review'))

quantile = [0.1, 0.25, 0.5, 0.75, 0.9, 0.95]
Err = 0.01
df1.stat.approxQuantile("Number of Review", quantile,Err)
```

Output-

```
1.0, 1.0, 1.0, 3.0, 9.0, 21.0]
```

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

```
from pyspark.sql.functions import *
rating4 = filtered.star_rating.isin(4)
rating5 = filtered.star_rating.isin(5)
dataframef_Q6=filtered.select("product_category","year","review_date") \
.withColumn("week_number",weekofyear("review_date")).where(rating4 | ratin
g5)
dataf_2 = dataframef_Q6.groupby("product_category","year","week_number").a
gg(F.countDistinct("week_number").alias("count"))
dataf_2.drop('count').show()
```

```
+--------------------+----+-----------+
|    product_category|year|week_number|
+--------------------+----+-----------+
|           Video_DVD|2015|         12|
|               Books|2011|         36|
|Digital_Ebook_Pur...|2015|         16|
```

```
|           Video_DVD|2011|         37|
|Digital_Ebook_Pur...|2014|         11|
|               Books|2008|         48|
|               Books|2007|         37|
|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
```

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

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

```
pivoted=filtered.groupBy("year",F.month(F.col("review_date"))).pivot
("product_category",to_pivot)\
 .agg((F.count("review_id")).alias("count_of_reviews"),
 F.round(F.mean("star_rating"),3).alias("Avg_star_rating")).sort("ye
ar","month(review_date)",ascending=True).show()
```

Output-

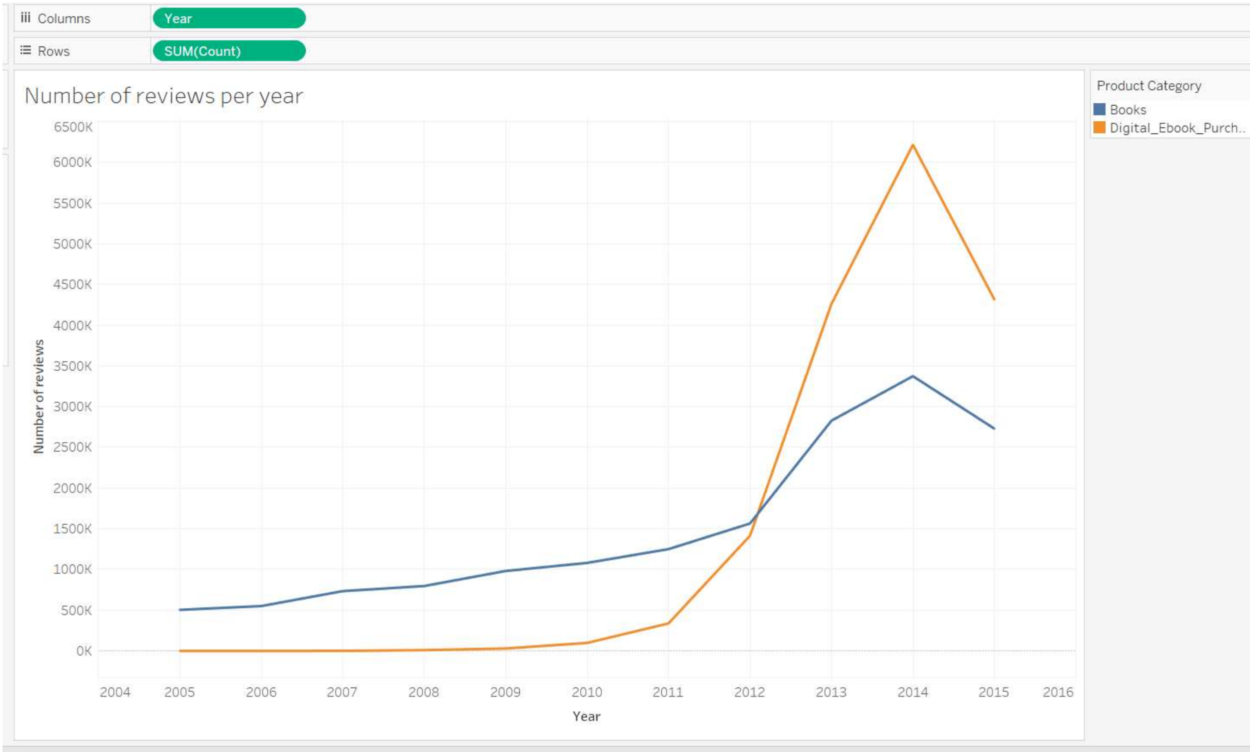
```
 |year|month(review_date)|Digital_Ebook_Purchase_count_of_reviews|Digital_
Ebook_Purchase_Avg_star_rating|Books_count_of_reviews|Books_Avg_star_ratin
g|
+----+------------------+---------------------------------------+----------
---------------------------------+----------------------+-------------------
+
|2005|                 1|                                      1|
5.0|             40428|                4.121|
|2005|                 2|                                   null|
null|             33722|                4.125|
|2005|                 3|                                      2|
4.5|             38878|                4.122|
|2005|                 4|                                      1|
5.0|             36890|                4.132|
```

```
|2005|                  5|                                     1|
1.0|                 36874|                 4.132|
|2005|                  6|                                  null|
null|                 36604|                 4.115|
|2005|                  7|                                     3|
2.0|                 45945|                 4.128|
|2005|                  8|                                     3|
2.667|                 58929|                 4.186|
|2005|                  9|                                     2|
4.0|                 58128|                 4.203|
|2005|                 10|                                     4|
4.0|                 51209|                  4.18|
|2005|                 11|                                     1|
5.0|                 40887|                 4.151|
|2005|                 12|                                     1|
5.0|                 42524|
```

Inference- For each year and each month for each product category we have calculated number of reviews and average star ratings.

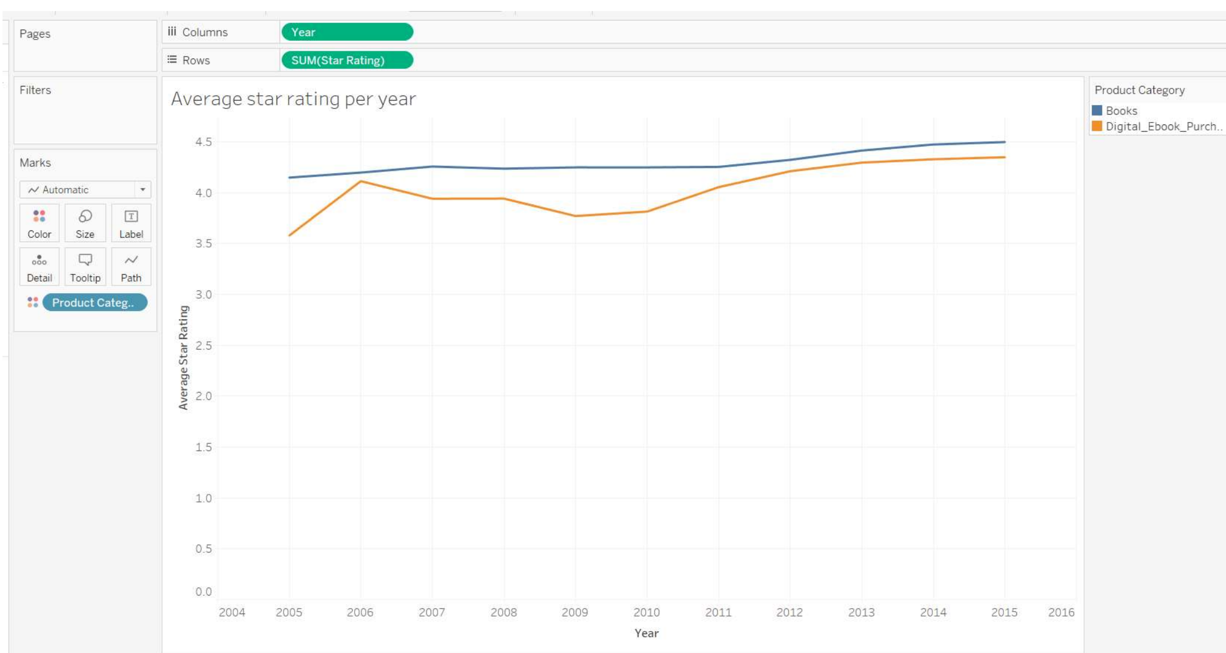
2.2 Produce two graphs to demonstrate aggregations from #1:

1. Number of reviews



Inference- We can see that number of reviews for Digital ebook purchase has upward trend from year 2012.

2. Average stars



Inference- We can see that average star rating for both the categories is almost constant over the years.

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

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

```
prodfilter=['Digital_Ebook_Purchase']
digital_e_book=filtered.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")).filte
r(F.col("product_category").isin(prodfilter))
```

```
trimmeed_dig_book=digital_e_book.select(F.lower(F.trim(F.col("product_titl
e"))).alias("product_title"),F.col("dig_book_count_of_reviews") \
                      ,F.col("dig_book_Avg_star_rating"))
```

```
var=['Books']
book=filtered.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("pr
oduct_title"),F.col("book_count_of_reviews") \
                      ,F.col("book_Avg_star_rating"))
```

```
joinExpression = trimmed_book["product_title"] == trimmeed_dig_book["produ
ct_title"]
joinType = "inner"
final=trimmed_book.join(trimmeed_dig_book, joinExpression, joinType)
```

```
final.show()
```

Output-

```
|       product_title|book_count_of_reviews|book_Avg_star_rating|       pr
oduct_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 sieg
e of khe...|                      156|                   3.327|
|           'dem bon'z|                    4|                 5.0|
'dem bon'z|                        2|                     5.0|
|   0400 roswell time|                    1|                 5.0|   0400 r
oswell 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 praye
rs for y...|                        7|                     5.0|
|13 cent killers: ...|                   37|               2.811|13 cent k
illers: ...|                       15|
```

We can see that there is difference in average ratings for printed books and digital books for some records.

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

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

Output- 276595

```
star1=F.col("dig_book_Avg_star_rating")>4
final.where(star1).count()
```

Output-

245529

Inference-

We can see that printed book has got more number of higher rating i.e count of more than 4 star ratings is higher for printed books as compared to digital book star ratings.

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.

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

LDA for reviews with 4 /5 star ratings.

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

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

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

LDA for reviews with 1 /2 stars.

```
df_ml1 = filtered.filter((F.col("product_category")=="Digital_Ebook_Purcha
se") | (F.col("product_category")=="Books") \
                     & (F.col("year")==2015) \
                     & (F.col("review_date")<'2015-02-01')
                     & (F.col("star rating")<3))
```

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

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

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.**

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

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

Topics for reviews with 1/2 star rating

Output-

```
topic: 0
---------------
story
```

```
good
characters
read
series
love
author
time
like
great
---------------
topic:  1
---------------
good
read
story
great
stars
really
like
love
characters
series
---------------
topic:  2
---------------
read
series
books
great
like
love
reading
loved
story
wait
---------------
topic:  3
---------------
story
read
love
characters
written
like
great
life
novel
way
---------------
topic:  4
---------------
read
good
like
```

```
great
books
reading
new
story
easy
author
---------------
topic:  5
---------------
read
great
time
like
reading
good
history
life
know
stars
```

Topics for reviews with 4/5 star rating

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

```
love
world
---------------
topic:  4
---------------
read
story
good
great
characters
---------------
topic:  5
---------------
read
like
great
time
interesting
```

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

Inference-

We can see that for ratings greater than 3 there are more positive words which justifies higher star ratings.

Similarly for reviews with star ratings less than 3 there are still some positive words. In this case topic modelling might not be so effective. In this case we might need to increase number of iterations and add more stop words to get ideal output.

References- Consulted with Hemant Taneja.