PROJECT REPORT

CLASS PROJECT PART – 2

INTRODUCTION AND PROBLEM DESCRIPTION

This project focuses on using the analytical skills by using big data technologies like AWS S3, AWS EMR – Hive and HDFS. We will be using the Amazon reviews dataset available in S3. Our dataset will be in parquet format to improve our processing and is partitioned by product category. We begin our analysis from 2005. We choose a few categories from the list available and exclude multiple reviews by customers and only choose the most recent reviews. We will use Spark Dataframe API to perform amazon reviews analysis by product category and year. We perform aggregations using pivot functionality and perform joins on dataframes for different product categories.

We aim to perform analysis on the amazon reviews dataset and use different window functions and analytical aggregate functions to demonstrate the concept of percentiles. We will also visualize some of our findings to provide clarity on the results obtained.

We begin by provisioning the EMR cluster and then copying the amazon reviews to EMR's HDFS. We then run the JupyterHub on the EMR cluster and use PySpark to run spark commands using Dataframe API.

TECHNICAL SCRIPTS EXPLANATION WITH VISUALIZATIONS

The steps we will perform to run spark commands using Dataframe API are as follows:

1. Create directories in HDFS for our product categories

   Query:

   ```
   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/
   ```

```
[hadoop@ip-172-31-68-196 ~]$ 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=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/
```

2. Copy data from S3 to these directories

   Query:

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

3. Create spark session

   Query:

   - from pyspark.sql import functions as F
   - spark

   Output:

```
In [1]: from pyspark.sql import functions as F
Starting Spark application
```

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
0	application_1588724034763_0008	pyspark	idle	Link	Link	✓

```
SparkSession available as 'spark'.

In [2]: spark
<pyspark.sql.session.SparkSession object at 0x7f7328fb7890>
```

4. Load data and perform exploratory data analysis

   Query:

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

df.printSchema()
print(df.columns)
```

Output:

```
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/*")

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)

In [5]: print(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']
```

5. Keep limited number of columns

Query:

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']
df_limited = df.select(columns_to_keep).filter(F.col("year")>2004)

df_limited.printSchema()

df_limited.rdd.getNumPartitions()

df_limited.count()

Output:

Keep only limited number of columns

```
In [6]: #Year filter
        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']
        df_limited = df.select(columns_to_keep).filter(F.col("year")>2004)

In [7]: 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)

In [8]: df_limited.rdd.getNumPartitions()

        216

In [9]: df_limited.count()

        71519024
```

6. Exclude multiple reviews

Query:

from pyspark.sql.window import Window

df_final = df_limited.select("*", F.row_number().over(Window.partitionBy("customer_id","product_category","product_id").orderBy("customer_id")).alias("row_num")).where("row_num = 1")

df_final.show(10)

type(df_final)

df_final.count()

df_final.drop("row_num")

df_final.persist()
df_final.show(2)

Output:

```
In [10]: from pyspark.sql.window import Window

         df_final = df_limited.select("*", F.row_number().over(Window.partitionBy("customer_id","product_category","product_id").orderBy(

         df_final.show(10)

+-----------+--------------+----------+--------------+--------------------+-----------+-------------+-----------+-------------
---+-----------+----+--------------------+-------+
|customer_id|     review_id|product_id|product_parent|       product_title|star_rating|helpful_votes|total_votes|verified_purch
ase|review_date|year|    product_category|row_num|
+-----------+--------------+----------+--------------+--------------------+-----------+-------------+-----------+-------------
---+-----------+----+--------------------+-------+
|   10000034|R18KORVQBM4KTQ|B0096YOL56|     837621444|Trident Case AEGI...|          5|            0|          1|
Y| 2013-10-27|2013|            Wireless|      1|
|   10000128|R10ZQ3KGV1I2XB|1421663260|     593136478|Golden Retrievers...|          4|            0|          0|
Y| 2011-01-04|2011|               Books|      1|
|   10000166| RKNBFD2B5J2KL|B002DQL34G|     827931632|He-Man and the Ma...|          5|            1|          1|
Y| 2014-03-03|2014|           Video_DVD|      1|
|   10000337| R561SQN54II24|B0000648ZP|     830751694|       Big Wednesday|          5|            0|          0|
Y| 2015-03-19|2015|           Video_DVD|      1|
|   10000391|R1IFDIZYHYJB79|B00CQ35HBQ|     432392654|Kingston Technolo...|          5|           18|         20|
Y| 2014-10-29|2014|                  PC|      1|
|   10000392| RPGI2AI6S0E2X|0789211343|      16303633|Wristwatch Annual...|          5|            0|          0|
Y| 2013-12-12|2013|               Books|      1|
|   10000810|R3A54C9LEDSS7P|B00ESW9SWW|     212800988|Memory Card Carry...|          5|            0|          0|
Y| 2014-08-14|2014|                  PC|      1|
|   10000971|R2TTNXGH75RD6H|B00G9MJLIU|       7176679|Jaynell's Wolf (A...|          4|            2|          2|
N| 2013-10-29|2013|Digital_Ebook_Pur...|      1|
|   10001012|R1APK015FILZHN|B00FOILW5A|      76548106|      Grimm Season 3|          5|            0|          0|
Y| 2015-03-10|2015|Digital_Video_Dow...|      1|
|   10001052|R274ZLVO49RYA3|0757902189|     792270473|James Bond 007 Co...|          1|            9|         24|
N| 2008-04-05|2008|               Books|      1|
+-----------+--------------+----------+--------------+--------------------+-----------+-------------+-----------+-------------
---+-----------+----+--------------------+-------+
only showing top 10 rows

In [11]: #Type
         type(df_final)

<class 'pyspark.sql.dataframe.DataFrame'>

In [12]: #Count records in the dataframe
         df_final.count()

65911069

In [13]: #Drop column row_num
         df_final.drop("row_num")

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]

In [14]: #Persist Dataframe - will keep the data in the memory as much as possible after first action
         df_final.persist()
         df_final.show(2)

+-----------+--------------+----------+--------------+--------------------+-----------+-------------+-----------+-------------
---+-----------+----+----------------+-------+
|customer_id|     review_id|product_id|product_parent|       product_title|star_rating|helpful_votes|total_votes|verified_purch
ase|review_date|year|product_category|row_num|
+-----------+--------------+----------+--------------+--------------------+-----------+-------------+-----------+-------------
---+-----------+----+----------------+-------+
|   10000034|R18KORVQBM4KTQ|B0096YOL56|     837621444|Trident Case AEGI...|          5|            0|          1|
Y| 2013-10-27|2013|        Wireless|      1|
|   10000128|R10ZQ3KGV1I2XB|1421663260|     593136478|Golden Retrievers...|          4|            0|          0|
Y| 2011-01-04|2011|           Books|      1|
+-----------+--------------+----------+--------------+--------------------+-----------+-------------+-----------+-------------
---+-----------+----+----------------+-------+
only showing top 2 rows
```

Using Spark Dataframe API, answer the following questions.

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

1. Number of reviews

Query:

#1. No of reviews in a given year for a particular product category sorted in descending order

```
df_final.groupby(F.col("product_category"), F.col("year"))\
.agg(F.count(F.col("review_id")).alias("no-of-reviews"))\
.sort("no-of-reviews", ascending =  False)\
.show(10)
```

Output:

```
+--------------------+----+-------------+
|    product_category|year|no-of-reviews|
+--------------------+----+-------------+
|Digital_Ebook_Pur...|2014|      6723862|
|Digital_Ebook_Pur...|2015|      4609418|
|Digital_Ebook_Pur...|2013|      4569661|
|               Books|2014|      3540847|
|            Wireless|2015|      3000784|
|               Books|2013|      2965945|
|               Books|2015|      2860735|
|            Wireless|2014|      2834087|
|                  PC|2014|      2008495|
|                  PC|2015|      1886148|
+--------------------+----+-------------+
only showing top 10 rows
```

2. Number of users

Query:

#2. No of users in a given year for a particular product category sorted in descending order

```
df_final.groupby(F.col("product_category"), F.col("year"))\
.agg(F.count(F.col("customer_id")).alias("no-of-users"))\
.sort("no-of-users", ascending =  False)\
.show(10)
```

Output:

```
+--------------------+----+-----------+
|    product_category|year|no-of-users|
+--------------------+----+-----------+
|Digital_Ebook_Pur...|2014|    6723862|
|Digital_Ebook_Pur...|2015|    4609418|
|Digital_Ebook_Pur...|2013|    4569661|
|               Books|2014|    3540847|
|            Wireless|2015|    3000784|
|               Books|2013|    2965945|
|               Books|2015|    2860735|
|            Wireless|2014|    2834087|
|                  PC|2014|    2008495|
|                  PC|2015|    1886148|
+--------------------+----+-----------+
only showing top 10 rows
```

3. Average and Median review stars

Query:

#median review stars

colName = "star_rating"

quantileProbs = [0.5]

relError = 0.05

df_final.stat.approxQuantile("star_rating", quantileProbs, relError)

#average review stars

df_final.groupby(F.col("product_category"), F.col("year"))\

.agg(F.avg(F.col("star_rating")).alias("avg_star_rating"))\

.sort("avg_star_rating", ascending = False)\

.show(10)

Output:

```
In [17]: #3. Average and median review stars

         #median review stars
         colName = "star_rating"
         quantileProbs = [0.5]
         relError = 0.05

         df_final.stat.approxQuantile("star_rating", quantileProbs, relError)

         [5.0]
```

```
In [18]: #average review stars
         df_final.groupby(F.col("product_category"), F.col("year"))\
         .agg(F.avg(F.col("star_rating")).alias("avg_star_rating"))\
         .sort("avg_star_rating", ascending =  False)\
         .show(10)

         +--------------------+----+------------------+
         |    product_category|year|   avg_star_rating|
         +--------------------+----+------------------+
         |           Video_DVD|2015|4.5298993735769075|
         |               Books|2015| 4.497376373554348|
         |           Video_DVD|2014| 4.485470244629136|
         |               Books|2014| 4.473279698332066|
         |               Books|2013|  4.41250326624398|
         |           Video_DVD|2013| 4.409183119689357|
         |Digital_Ebook_Pur...|2015| 4.349698812301249|
         |Digital_Ebook_Pur...|2014| 4.332813493197808|
         |               Books|2012|  4.31468389465115|
         |Digital_Ebook_Pur...|2013| 4.301698966290935|
         +--------------------+----+------------------+
         only showing top 10 rows
```

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

Query:

#4. Percentiles of length of reviews

sample = df.withColumn("length_of_reviews", F.length(F.col("review_body"))) sample.show(5)

colname = "length_of_reviews"

quantileProbs = [0.1, 0.25, 0.5, 0.75, 0.9, 0.95]

relError = 0.05

sample.stat.approxQuantile(colname, quantileProbs, relError)

Output:

```
+-----------+-----------+--------------+----------+--------------+--------------------+-----------+-------------+-----------+--
--+-----------------+--------------------+--------------------+-----------+----+----------------+-----------------+
|marketplace|customer_id|     review_id|product_id|product_parent|       product_title|star_rating|helpful_votes|total_votes|vi
ne|verified_purchase|     review_headline|         review_body|review_date|year|product_category|length_of_reviews|
+-----------+-----------+--------------+----------+--------------+--------------------+-----------+-------------+-----------+--
--+-----------------+--------------------+--------------------+-----------+----+----------------+-----------------+
|         FR|    5625782|R3BI9MOWLB149P|B000BOEZTW|     764225016|Titanic [Édition ...|          4|            0|          0|
N|                Y|Le Titanic - Edit...|Retrouvez Jack Da...| 2014-04-09|2014|       Video_DVD|              611|
|         US|   53068969|R3UIB7DM8VZSR8|6301562925|     191905241|   Halloween 5 [VHS]|          3|            0|          0|
N|                N|Deals with many n...|This one picks up...| 1996-12-09|1996|       Video_DVD|              478|
|         FR|    9378802|R3246AQIMTPJLJ|B00F4T4DOG|     922259594|      Rush [Blu-ray]|          5|            0|          0|
N|                Y|un belle hommage ...|Rush rend hommage...| 2014-04-09|2014|       Video_DVD|              316|
|         US|   53068969|R3UIB7DM8VZSR8|6301562925|     191905241|   Halloween 5 [VHS]|          3|            0|          0|
N|                N|Deals with many n...|This one picks up...| 1996-12-09|1996|       Video_DVD|              478|
|         FR|   17723065|R16FNL5WUJ2WRK|B00G25HTMY|     407523545|Thor : Le Monde d...|          4|            0|          0|
N|                N|          très bien|rien à dire sur l...| 2014-04-09|2014|       Video_DVD|              133|
+-----------+-----------+--------------+----------+--------------+--------------------+-----------+-------------+-----------+--
--+-----------------+--------------------+--------------------+-----------+----+----------------+-----------------+
only showing top 5 rows

[74.0, 127.0, 233.0, 959.0, 44001.0, 51019.0]
```

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

Query:

#5. Percentiles for number of reviews per product

sample1 = df_final.groupby(F.col("product_category"))\

.agg(F.count(F.col("review_id")).alias("review_count"))

sample1.show(5)

colname = "review_count"

quantileProbs = [0.1, 0.25, 0.5, 0.75, 0.9, 0.95]

relError = 0.05

sample1.stat.approxQuantile(colname, quantileProbs, relError)

Output:

```
+--------------------+------------+
|    product_category|review_count|
+--------------------+------------+
|                  PC|     6897944|
|            Wireless|     9002606|
|Digital_Video_Dow...|     4115479|
|Digital_Ebook_Pur...|    17923458|
|               Books|    17134822|
+--------------------+------------+
only showing top 5 rows

[4115479.0, 5331078.0, 6897944.0, 17134822.0, 17923458.0, 17923458.0]
```

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

Query:
#6. Identify week number of year and product category with most positive reviews

df_weekno = df_final.withColumn("week_number", F.date_format(F.to_date("review_date", "yyyy-MM-dd"), "w"))
df_weekno.show(5)

df_weekno.groupby(F.col("product_category"), F.col("year"), F.col("week_number"))\
.agg(F.count(F.expr("star_rating >= 4")).alias("number_of_positive_reviews"))\
.sort("number_of_positive_reviews", ascending = False)\
.show(10)

Output:

```
+-----------+--------------+----------+--------------+--------------------+-----------+-------------+-----------+-------------
---+-----------+----+----------------+-------+-----------+
|customer_id|     review_id|product_id|product_parent|       product_title|star_rating|helpful_votes|total_votes|verified_purch
ase|review_date|year|product_category|row_num|week_number|
+-----------+--------------+----------+--------------+--------------------+-----------+-------------+-----------+-------------
---+-----------+----+----------------+-------+-----------+
|   10000034|R18KORVQBM4KTQ|B0096YOL56|     837621444|Trident Case AEGI...|          5|            0|          1|
Y| 2013-10-27|2013|        Wireless|      1|         44|
|   10000128|R10ZQ3KGV1I2XB|1421663260|     593136478|Golden Retrievers...|          4|            0|          0|
Y| 2011-01-04|2011|           Books|      1|          2|
|   10000166| RKNBFD2B5J2KL|B002DQL34G|     827931632|He-Man and the Ma...|          5|            1|          1|
Y| 2014-03-03|2014|       Video_DVD|      1|         10|
|   10000337| R561SQN54II24|B0000648ZP|     830751694|       Big Wednesday|          5|            0|          0|
Y| 2015-03-19|2015|       Video_DVD|      1|         12|
|   10000391|R1IFDIZYHYJB79|B00CQ35HBQ|     432392654|Kingston Technolo...|          5|           18|         20|
Y| 2014-10-29|2014|              PC|      1|         44|
+-----------+--------------+----------+--------------+--------------------+-----------+-------------+-----------+-------------
---+-----------+----+----------------+-------+-----------+
only showing top 5 rows

+--------------------+----+-----------+--------------------------+
|    product_category|year|week_number|number_of_positive_reviews|
+--------------------+----+-----------+--------------------------+
|Digital_Ebook_Pur...|2015|          8|                    182020|
|Digital_Ebook_Pur...|2014|          1|                    173040|
|Digital_Ebook_Pur...|2015|         11|                    166546|
|Digital_Ebook_Pur...|2015|          9|                    165342|
|Digital_Ebook_Pur...|2015|         10|                    160533|
|Digital_Ebook_Pur...|2015|         12|                    158034|
|Digital_Ebook_Pur...|2015|         13|                    150088|
|Digital_Ebook_Pur...|2014|         30|                    147867|
|Digital_Ebook_Pur...|2014|         31|                    146915|
|Digital_Ebook_Pur...|2014|         32|                    145729|
+--------------------+----+-----------+--------------------------+
only showing top 10 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

Query:

#1. Using Spark Pivot Functionality, produce a dataframe with relevant columns

dfwithMonth = df_final.withColumn("month", F.month(F.col("review_date")))

categories_to_pivot = ['Digital_Ebook_Purchase','Books']

df_pivoted = dfwithMonth.groupby(F.col("year")).pivot("product_category", categories_to_pivot)\

.agg(F.count(F.col("review_id")).alias("number_of_reviews"),

F.avg(F.col("star_rating")).alias("avg_review_stars"))

df_pivoted.show(10)

df_pivoted.cache()

Output:

```
+----+------------------------------------+-----------------------------------+---------------------+---------------
------+
|year|Digital_Ebook_Purchase_number_of_reviews|Digital_Ebook_Purchase_avg_review_stars|Books_number_of_reviews|Books_avg_review
_stars|
+----+------------------------------------+-----------------------------------+---------------------+---------------
------+
|2007|                                 508|                 3.938976377952756|               761029|    4.258168873
985091|
|2015|                             4609418|                 4.349698812301249|              2860735|    4.497376373
554348|
|2006|                                  36|                 4.027777777777778|               568389|   4.1965502499
168705|
|2013|                             4569661|                 4.301698966290935|              2965945|     4.41250326
624398|
|2014|                             6723862|                 4.332813493197808|              3540847|    4.473279698
332066|
|2012|                             1526595|                 4.214259184656048|              1649719|     4.31468389
465115|
|2009|                               31105|                3.7770776402507638|              1015574|    4.246829871
580013|
|2005|                                  19|                3.5789473684210527|               521022|    4.148055168
495764|
|2010|                              102514|                3.8219560255184657|              1120761|    4.246933110
627511|
|2011|                              350133|                 4.055544607334927|              1303119|    4.251156648
011425|
+----+------------------------------------+-----------------------------------+---------------------+---------------
------+
only showing top 10 rows

DataFrame[year: int, Digital_Ebook_Purchase_number_of_reviews: bigint, Digital_Ebook_Purchase_avg_review_stars: double, Books_n
umber_of_reviews: bigint, Books_avg_review_stars: double]
```

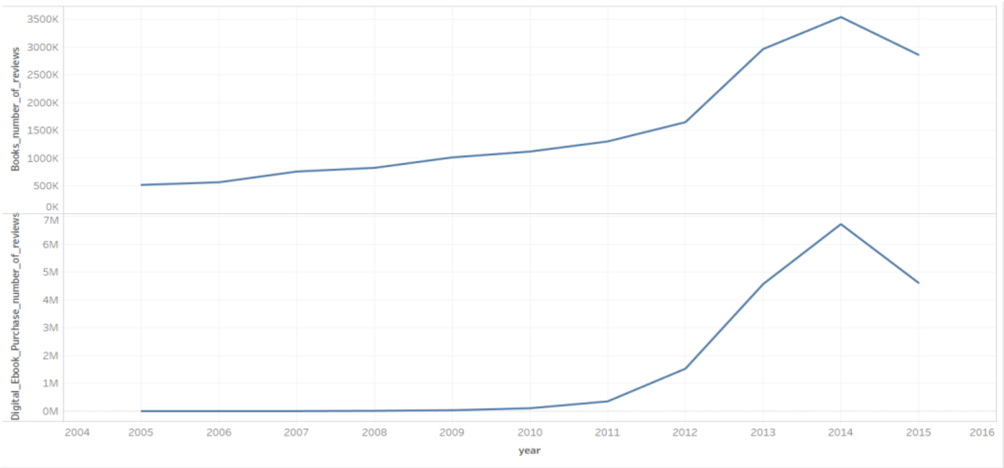
2. Produce two graphs to demonstrate aggregations from #1:

Query:

df_pivoted.coalesce(1).write.csv(path = 'hdfs:////user/livy/df_pivoted.csv', header = 'true')

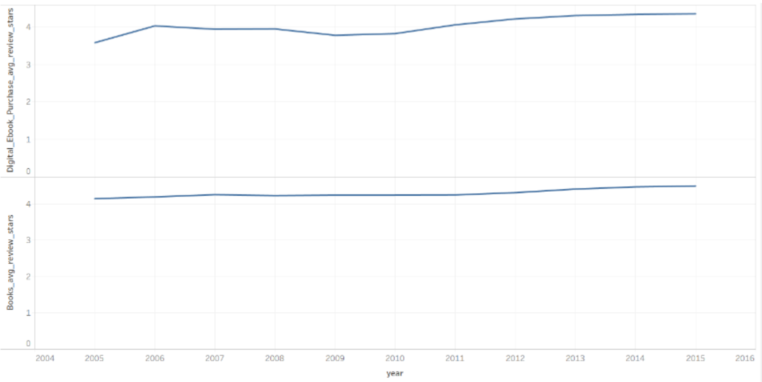
hdfs dfs -copyToLocal /user/livy/df_pivoted.csv ~/

1. Number of reviews



The number of reviews for Digital eBook Purchases are way more than the Books.

2. Average stars



The average star rating for books have been consistent over the years whereas the average star rating for digital eBooks do go below an average of 4 stars.

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.

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 visa versa. Alternatively, you can make the conclusion by using appropriate pairwise statistic.

Query:

```
for i in df_final.columns:
   df_final = df_final.withColumn(i, F.ltrim(F.rtrim(df_final[i])))
```

df_final = df_final.withColumn('product_title', F.lower(F.col('product_title')))

df_final.show(5)

df_books = df_final.select("*").where(F.col('product_category').like('%Books%'))

df_books = df_books.groupBy('product_title').agg(F.avg('star_rating'))

df_ebooks = df_final.select("*").where(F.col('product_category').like('%Digital_Ebook_Purchase%'))

df_ebooks = df_ebooks.groupBy('product_title').agg(F.avg('star_rating'))

df_books.show(5)

df_ebooks.show(5)

innerjoin = df_books.join(df_ebooks, df_books.product_title == df_ebooks.product_title)

innerjoin.show(5)

Output:

```
+--------------------+----------------+--------------------+-----------------+
|       product_title| avg(star_rating)|       product_title|  avg(star_rating)|
+--------------------+----------------+--------------------+-----------------+
|"rays of light": ...|             5.0|"rays of light": ...|              5.0|
|"the siege of khe...|4.315789473684211|"the siege of khe...| 3.326923076923077|
|           'dem bon'z|             5.0|           'dem bon'z|              5.0|
|   0400 roswell time|             5.0|   0400 roswell time|3.6666666666666665|
|10 smart things g...|             4.8|10 smart things g...| 4.833333333333333|
+--------------------+----------------+--------------------+-----------------+
only showing top 5 rows
```

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.

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?

Query:

#Import 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

#For star rating 4 and 5

df_ml = df.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)

#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()

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

corpus_df.printSchema()
```

```
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()


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

stop_words = stop_words + ['br','book','34']


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)

# 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)

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

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

"""
ll = model.logLikelihood(countVectors)
lp = model.logPerplexity(countVectors)
print("The lower bound on the log likelihood of the entire corpus: " + str(ll))
print("The upper bound on perplexity: " + str(lp))

# Describe topics.
topics = model.describeTopics(3)
print("The topics described by their top-weighted terms:")
topics.show(truncate=False)
```

```
# Shows the result
transformed = model.transform(countVectors)
transformed.show(truncate=False)
"""
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 ("----------")
```

Note:

For 1 and 2 star rating, the only difference in code is:

```
df_ml = df.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)
```

Output:

For 4 and 5 star rating:

```
('topic: ', 0)
----------
life
story
read
```

```
love
people
time
god
author
great
world
----------
('topic: ', 1)
----------
read
characters
series
love
reading
books
story
world
good
like
----------
('topic: ', 2)
----------
story
good
characters
read
like
author
really
little
plot
time
----------
('topic: ', 3)
----------
story
great
read
like
really
character
reading
characters
author
forward
----------
('topic: ', 4)
----------
great
read
story
life
reading
way
good
like
really
```

```
recipes
----------
('topic: ', 5)
----------
like
read
really
love
story
know
way
didn
good
books
----------
('topic: ', 6)
----------
love
story
series
read
great
books
loved
characters
wait
like
----------
('topic: ', 7)
----------
read
good
story
great
stars
books
reading
loved
characters
enjoyed
----------
('topic: ', 8)
----------
good
read
kindle
great
author
easy
information
like
books
interesting
----------
('topic: ', 9)
----------
read
great
```

```
love
story
good
loved
know
like
books
time
----------
```

For 1 and 2 star rating:

```
('topic: ', 0)
----------
read
great
life
kindle
god
author
time
easy
love
recommend
----------
('topic: ', 1)
----------
read
good
like
time
reading
books
world
great
author
years
----------
('topic: ', 2)
----------
story
life
love
good
read
characters
time
like
author
world
----------
('topic: ', 3)
----------
story
read
characters
great
like
```

```
really
character
author
good
novel
----------
('topic: ', 4)
----------
read
story
great
life
reading
family
way
people
women
good
----------
('topic: ', 5)
----------
love
story
like
read
really
series
loved
know
characters
way
----------
('topic: ', 6)
----------
love
series
read
story
great
books
loved
characters
wait
best
----------
('topic: ', 7)
----------
read
good
story
great
stars
books
reading
characters
loved
really
----------
```

```
('topic: ', 8)
----------
good
read
story
author
la
que
interesting
thought
short
stories
----------
('topic: ', 9)
----------
read
like
good
story
really
time
know
love
great
loved
----------
```

We can conclude that average star rating is not a good approximation of to differentiate between positive and negative reviews as there are similar topics for both 1-2 stars and 4-5 stars and we cannot really differentiate between the star ratings.

CONCLUSION

We were able to make use of big data technologies to analyse the amazon reviews dataset, answer data exploratory questions, compare product categories and observe trends in metrics over time. We were successfully able to perform exploratory data analysis and perform queries using Spark Dataframe API. When we were trying to compare the average star ratings for product categories 'Books' and 'Digital eBooks', the number of reviews were way more for eBooks than Books and the average star rating for Books were consistently greater than 4 stars as compared to eBooks. This just demonstrates one of our findings. To conclude, we were able to perform detailed analysis on the amazon review dataset and we made use of Spark Dataframe API to obtain results.

REFERENCES

https://piazza.com/class

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

https://hopsworks.readthedocs.io/en/0.9/user_guide/hopsworks/jupyter.html

https://towardsdatascience.com/the-best-pandas-plotting-features-c9789e04a5a0

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