books-analysis

```
%spark.pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, udf, when, explode, desc
from pyspark.sql.types import ArrayType, StringType
```

FINISHED

Took 34 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:14:11 PM.

```
%spark.pyspark
# Initialize Spark Session
spark = SparkSession.builder.appName("ReviewAnalysis").getOrCreate()

# Read data (30000 rows of ratings with emotions)
path = "/user/tl4151_nyu_edu/emotion_analysis_result_1209"
rating_df = spark.read.parquet(path)
print("Number of total rows:", rating_df.count())
print("Number of unique books:", rating_df.select("Title").distinct().count())
rating_df.show(20)
```

SPARK JOB FINISHED

```
Number of total rows: 93444
Number of unique books: 14815
+-----------+------------+-----------+------------+------------+------------+------------+----
---------------+------------+--------------------+
|     row_id|     sadness|        joy|        love|       anger|        fear|    surprise|
Title|review/score|         review/text|
+-----------+------------+-----------+------------+------------+------------+------------+----
---------------+------------+--------------------+
|25769803776|  0.03131323|0.060917772| 0.017827341|   0.8364197| 0.047393985| 0.006127944|Murd
er By The Boo...|         5.0|"(If you're inter...|
|25769803777|0.0027958236| 0.96540827| 0.016390339|  0.00815508|0.0031244447|  0.00412603|Murd
er By The Boo...|         5.0|This doesn't seem...|
|25769803778|0.0028010875| 0.99158317|0.0026439226|0.0013408175| 5.480072E-4|0.0010830157|Murd
er By The Boo...|         5.0|Rex Stout's Nero ...|
|25769803779|  0.37285906|0.020032752|   0.0710962|   0.5061638| 0.027043404| 0.002804824|Murd
er By The Boo...|         5.0|I loved just abou...|
|25769803780| 0.017133001| 0.42699972| 0.032291926|   0.4866251| 0.030644594|0.0063057253|Murd
```

Took 20 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:14:31 PM.

```
%spark.pyspark
# Only keep the top/ higheest emotion of each review
from pyspark.sql.functions import greatest, col, lit, when

# List of emotion columns
emotion_columns = ["sadness", "joy", "love", "anger", "fear", "surprise"]

# Add the 'emotion' column by finding the column with the highest score
rating_df = rating_df.withColumn(
    "emotion",
    when(col("sadness") == greatest(*[col(c) for c in emotion_columns]), "sadness")
    .when(col("joy") == greatest(*[col(c) for c in emotion_columns]), "joy")
    .when(col("love") == greatest(*[col(c) for c in emotion_columns]), "love")
    .when(col("anger") == greatest(*[col(c) for c in emotion_columns]), "anger")
    .when(col("fear") == greatest(*[col(c) for c in emotion_columns]), "fear")
    .when(col("surprise") == greatest(*[col(c) for c in emotion_columns]), "surprise")
```

SPARK JOB (http://nyu-dataproc-w-1.c.hpc-dataproc-19b8.internal:44737/jobs/job?id=7) FINISHED

```
)

# Show a few rows to verify the new
rating_df = rating_df.select("row_id", "Title", "review/score", "review/text", "emotion")
rating_df.show(5)
```

books-analysis

```
+----------+--------------------+------------+--------------------+-------+
|     row_id|               Title|review/score|         review/text|emotion|
+----------+--------------------+------------+--------------------+-------+
|25769803776|Murder By The Boo...|         5.0|"(If you're inter...|  anger|
|25769803777|Murder By The Boo...|         5.0|This doesn't seem...|    joy|
|25769803778|Murder By The Boo...|         5.0|Rex Stout's Nero ...|    joy|
|25769803779|Murder By The Boo...|         5.0|I loved just abou...|  anger|
|25769803780|Murder By The Boo...|         5.0|If you are a Nero...|  anger|
+----------+--------------------+------------+--------------------+-------+
only showing top 5 rows
```

Took 1 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:14:32 PM.

Top k keywords

FINISHED

Took 0 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:13:46 PM.

Unigram

FINISHED

Took 0 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:13:46 PM.

```
%spark.pyspark                                                    SPARK JOB  FINISHED
from pyspark.sql.functions import explode, split, col, lower, regexp_replace


# Remove punctuation and convert to lowercase
rating_df = rating_df.withColumn(
    "cleaned_text",
    lower(regexp_replace(col("review/text"), "[^a-zA-Z0-9\\s]", ""))  # Keep only letters, nur
)

# Tokenize the review/text column
tokenized_df = rating_df.select(
    col("row_id"),
    col("cleaned_text"),
    explode(split(col("cleaned_text"), "\\s+")).alias("word")
)

print(tokenized_df.count())
tokenized_df.show(20)
```

```
|25769803776|if youre interest...|    pretty|
|25769803776|if youre interest...|    goodwe|
|25769803776|if youre interest...|     begin|
|25769803776|if youre interest...|      with|
|25769803776|if youre interest...|       the|
|25769803776|if youre interest...|    murder|
|25769803776|if youre interest...|        of|
|25769803776|if youre interest...|         a|
+-----------+--------------------+----------+
only showing top 20 rows
```

Took 7 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:14:39 PM.

books-analysis

```
%spark.pyspark
```

SPARK JOB FINISHED

```
# Filter out stopwords
uni_stop_words = ['a', 'an', 'the', 'is', 'in', 'at', 'of', 'on', 'and', 'to', 'for', 'with',
    'could', 'should', 'you', 'your', 'we', 'they', 'their', 'but', 'not', 'or', 'if', 'which
    'about', 'so', 'who', 'like', 'her', 'more', 'very', 'just', 'some', 'out', 'there', 'me'
    'do', 'also', 'it', 'dont', 'then', 'its', 'book', 'its', 'story', 'great', 'first', 'goo
    'through', 'way', 'these', 'well', 'its', 'know', 'two', 'am', 'them', 'make', 'little',
    'written', 'too', 'while', 'being', 'each', 'author', 'him', 'such', 'us', 'im', 'few', 'm
    'put', 'before', 'makes', 'last', 'read', 'lot', 'go', 'going', 'another', 'something', 'm
    'characters', 'new', 'series', 'those', 'our', 'over', 'character', 'anyone', 'old', 'int
    'different', 'stories', 'need', 'however', 'though', 'down', 'without', 'part', 'come', 'c
    'yet', 'used', 'highly', 'might', 'off', 'worth', 'once', 'keep', 'thing', 'three', 'enou
    'loved', 'buy', 'actually', 'understand', 'everything', 'everyone', 'although', 'having',
    , 'whole', 'able', 'short', 'school', 'mr', 'liked', 'left', 'place', 'seem', 'books', 'r
    'during', 'gives', 'here', 'simply', 'thats', 'getting', 'set', 'write', 'second', 'sure'
    'series', 'live', 'often', 'done', 'kind', 'read', 'try', 'big', 'believe', 'lives', 'alr
    'several', 'readers', 'page', 'i', 'truly', 'especially', 'least', 'anything', 'comes', 'c
    , 'wait', 'youll', 'together', 'lost', 'job', 'novels', 'care', 'review', 'small', 'gets'
    'went', 'movie', 'hes', 'shows', 'mind', 'using', 'interested', 'definitely', 'id', 'time
    'class', 'works', 'reason', 'writer', 'past', 'age', 'kids', 'home', 'kind', 'himself', '
        'side', 'mother', 'pretty', 'today', 'parents', 'events',
    'instead', 'ending', 'detail', 'later', 'son', 'seen', 'within', 'version', 'absolutely',
        'throughout', 'wrote', 'view', 'half', 'cover', 'ones', 'days',
    'called', 'next', 'thinking', 'text', 'yes', 'says'
    ]

# Convert the stop words list to a broadcast variable for efficiency
uni_broadcast_stop_words = spark.sparkContext.broadcast(uni_stop_words)

# Filter out stop words
filtered_df = tokenized_df.filter(~col("word").isin(uni_broadcast_stop_words.value))

print(filtered_df.count())
filtered_df.show(20)
```

```
3310953
+-----------+--------------------+------------+
|     row_id|        cleaned_text|        word|
+-----------+--------------------+------------+
|25769803776|if youre interest...|       audio|
|25769803776|if youre interest...|     edition|
|25769803776|if youre interest...|     michael|
|25769803776|if youre interest...|  pritchards|
|25769803776|if youre interest...|  unabridged|
```

```
|25769803776|if youre interest...|   narration|
|25769803776|if youre interest...|      goodwe|
|25769803776|if youre interest...|       begin|
|25769803776|if youre interest...|      murder|
|25769803776|if youre interest...|     leonard|
|25769803776|if youre interest...|       dykes|
|25769803776|if youre interest...|      fished|
```

books-analysis

Took 9 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:14:48 PM.

```
%spark.pyspark

# Count word frequencies
keyword_counts = filtered_df.groupBy("word").count()
sorted_keywords = keyword_counts.orderBy(col("count").desc())
sorted_keywords.show(100)
```

SPARK JOB FINISHED

```
+-------------+-----+
|         word|count|
+-------------+-----+
|         love|15089|
|      history| 6151|
|       family| 6108|
|        young| 5636|
|          war| 5157|
|     children| 4485|
|     american| 4232|
|      friends| 4115|
|      classic| 4074|
|      fiction| 3581|
|     language| 3316|
|          god| 3273|
|      started| 3217|
|    beautiful| 3180|
|      edition| 3121|
```

Took 8 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:14:56 PM.

Bigram

FINISHED

Took 0 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:13:55 PM.

```
%spark.pyspark
# First remove stop words from the sentence before finding bigrams
broadcast_uni_stop_words = spark.sparkContext.broadcast(uni_stop_words)

def remove_single_stopwords(sentence):
    if not sentence or sentence.strip() == "":
        return sentence
    words = sentence.split()
    # Remove single-word stop words
    filtered_words = [word for word in words if word not in broadcast_uni_stop_words.value]
    return " ".join(filtered_words)

remove_single_stopwords_udf = udf(remove_single_stopwords, StringType())
```

SPARK JOB FINISHED

books-analysis

```
# Define UDF to generate bigrams
def generate_ngrams(text, n):
    if not text or text.strip() == "":  # Handle None or empty strings
        return []
    words = text.split()
    ngrams = [' '.join(words[i:i+n]) for i in range(len(words) - n + 1)]
    return ngrams

bigrams_udf = udf(lambda text: generate_ngrams(text, 2), ArrayType(StringType()))


# Remove single-word stop words from the cleaned_text column
rating_df = rating_df.withColumn("cleaned_text_new", remove_single_stopwords_udf(col("cleaned
# Generate biagrams from
bigrams = rating_df.withColumn("bigrams", bigrams_udf(col("cleaned_text_new")))

# Explode 'bigrams' column to create one row per bigram
bigrams_df = bigrams.select(
    col("row_id"),
    col("cleaned_text"),
    explode(col("bigrams")).alias("bigram")  # Explode the bigrams column
)

print(bigrams_df.count())
bigrams_df.show(5)
```

```
3216411
+-----------+--------------------+--------------------+
|     row_id|        cleaned_text|              bigram|
+-----------+--------------------+--------------------+
|25769803776|if youre interest...|       audio edition|
|25769803776|if youre interest...|     edition michael|
|25769803776|if youre interest...|  michael pritchards|
|25769803776|if youre interest...|pritchards unabri...|
|25769803776|if youre interest...|unabridged narration|
+-----------+--------------------+--------------------+
only showing top 5 rows
```

Took 23 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:15:19 PM.

SPARK JOB FINISHED

```
%spark.pyspark
bigram_counts = bigrams_df.groupBy("bigram").count()
sorted_bigrams = bigram_counts.orderBy(col("count").desc())
sorted_bigrams.show(100, truncate=False)
```

```
+--------------------+-----+
|bigram              |count|
+--------------------+-----+
|science fiction     |863  |
|lord rings          |790  |
|civil war           |651  |
|robert jordan       |623  |
|20th century        |550  |
|united states       |532  |
|stephen king        |426  |
|se hinton           |394  |
|jane eyre           |392  |
```

```
|twists turns        |382  |
|subject matter      |375  |
|war ii              |371  |
|cs lewis            |368  |
|author carnegie     |363  |
```

books-analysis

Took 24 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:15:43 PM.

Trigrams

FINISHED

Took 0 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:14:03 PM.

```
%spark.pyspark

# Functions to remove stopwords and generate ngram are already defined in the bigrams code, s
trigram_udf = udf(lambda text: generate_ngrams(text, 3), ArrayType(StringType()))
trigrams = rating_df.withColumn("trigrams", trigram_udf(col("cleaned_text_new")))
trigrams_df = trigrams.select(
    col("row_id"),
    col("cleaned_text_new"),
    explode(col("trigrams")).alias("trigram")  # Explode the trigrams column
)

trigram_counts = trigrams_df.groupBy("trigram").count().orderBy(desc("count"))
trigram_counts.show(100, truncate=False)
```

SPARK JOB FINISHED

```
+---------------------------+-----+
|trigram                    |count|
+---------------------------+-----+
|camp green lake            |281  |
|f scott fitzgerald         |185  |
|count monte cristo         |163  |
|win friends influence      |158  |
|outsiders se hinton        |151  |
|c s lewis                  |89   |
|lord rings trilogy         |77   |
|michael j fox              |75   |
|lion witch wardrobe        |74   |
|christ clone trilogy       |73   |
|adventures huckleberry finn|69   |
|hated hated hated          |68   |
|boy named stanley          |67   |
|stranger strange land      |61   |
|boy named nonyboy          |60   |
```

Took 24 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:16:07 PM.

Top keywords

FINISHED

1. Themes/ Genres: family, history, war, spiritual, entertaining, mystery, death, vampire, gang, religion, art, fairy tale, love
2. Authors/Characters: robert jordan, se hinton, stephen king, F. Scott Fitzgerald, anne rice, dale carnegie, helen fielding, kurt vonnegut, lemony snicket, lord rings

3. Literary Periods/Movements: american dream, 20th century, american literature, civil war, 19th century
4. Adjective: boring, disapointed, 4 stars, blah blah blah

books-analysis

Took 0 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:14:07 PM.

Partition by top keywords

FINISHED

Took 0 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:14:09 PM.

SPARK JOB FINISHED

```
%spark.pyspark
from pyspark.sql.functions import col, array, when, lit, explode, expr

# Step 1: Define the list of keywords
keywords = [
    "family", "history", "war", "spiritual", "entertaining", "mystery", "death", "vampire", "
    "art", "fairy tale", "love", "robert jordan", "se hinton", "stephen king", "F. Scott Fitz
    "dale carnegie", "helen fielding", "kurt vonnegut", "lemony snicket", "american dream", "
    "american literature", "civil war", "19th century", "lord rings", "boring", "disapointed"
]

# Step 2: Create a column containing an array of matched keywords
keywords_df = rating_df.withColumn(
    "matched_keywords",
    array(*[when(col("cleaned_text_new").contains(keyword), lit(keyword)) for keyword in keyw
)

# Step 3: Remove nulls from the `matched_keywords` array
keywords_df = keywords_df.withColumn(
    "matched_keywords",
    expr("filter(matched_keywords, x -> x IS NOT NULL)")
)

print("Count of rows before exploding by keywords and drop rows w/o keywords", keywords_df.co

# Step 4: Explode the `matched_keywords` column to create one row per keyword
keywords_df = keywords_df.withColumn("keyword", explode(col("matched_keywords")))

print("Count of rows after exploding by keywords", keywords_df.count())
print("Number of unique books with review containing top keywords", keywords_df.select("Title
keywords_df.show(10)
```

```
Count of rows before exploding by keywords and drop rows w/o keywords 93444
Count of rows after exploding by keywords 78292
Number of unique books with review containing top keywords 10190
+-----------+--------------------+------------+--------------------+-------+-----------------
--+--------------------+-------------------+-------+
|     row_id|               Title|review/score|         review/text|emotion|        cleaned_te
xt|    cleaned_text_new|   matched_keywords|keyword|
+-----------+--------------------+------------+--------------------+-------+-----------------
--+--------------------+-------------------+-------+
|25769803776|Murder By The Boo...|         5.0|"(If you're inter...|  anger|if youre interes
t...|audio edition mic...|[family, art, love]| family|
|25769803776|Murder By The Boo...|         5.0|"(If you're inter...|  anger|if youre interes
t...|audio edition mic...|[family, art, love]|    art|
|25769803776|Murder By The Boo...|         5.0|"(If you're inter...|  anger|if youre interes
```

```
t...|audio edition mic...|[family, art, love]|   love|
|25769803778|Murder By The Boo...|         5.0|Rex Stout's Nero ...|    joy|rex stouts nero
w...|rex stouts nero w...|          [mystery]|mystery|
```

Took 47 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:16:54 PM.

books-analysis

```
%spark.pyspark
# Drop duplicates and only select the needed columns
result_df = keywords_df.dropDuplicates(["Title", "review/score", "emotion", "keyword"])
result_df = result_df.select("Title", "review/score", "emotion", "keyword")
print("Count of rows after droping duplicates:", result_df.count())
print("Number of unique books with review containing top keywords", result_df.select("Title")
result_df.show(20)
```

SPARK JOB FINISHED

```
Count of rows after droping duplicates: 47499
Number of unique books with review containing top keywords 10189
+--------------------+------------+-------+---------+
|               Title|review/score|emotion|  keyword|
+--------------------+------------+-------+---------+
|Moderatto En Dire...|         1.0|  anger|      art|
|Flood Summer: A N...|         4.0|   love|      war|
|Naval warfare und...|         4.0|    joy|  history|
|   Shopaholic & Baby|         2.0|   love|     love|
|    Cain His Brother|         4.0|    joy|  history|
|    Cain His Brother|         4.0|sadness|      art|
|Silent Cry (Willi...|         5.0|    joy|  mystery|
|     Force Of Reason|         5.0|  anger|  history|
|POEM OF THE MAN-G...|         5.0|    joy|      art|
|Loving Mr. Spock:...|         3.0|    joy|      war|
|Quo Vadis (The Be...|         4.0|    joy|  history|
|Great Gatsby (Eve...|         3.0|   love|      war|
|Among Warriors: A...|         5.0|    joy|      art|
```

Took 1 min 1 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:17:55 PM.

```
%spark.pyspark
# Convert review/score column to float and remove invalid ones with -1
result_df = result_df.withColumn(
    "review/score",
    when(col("review/score").cast("float").isNotNull(), col("review/score").cast("float"))
    .otherwise(-1.0)
)

# Save the result
output_path = "/user/tl4151_nyu_edu/books_result_1209"
result_df.write.mode("overwrite").partitionBy("emotion", "keyword", "review/score").parquet(ou
```

SPARK JOB FINISHED

Took 1 min 34 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:19:29 PM.

FINISHED

Took 0 sec. Last updated by anonymous at December 09 2024, 4:23:17 PM.

```
%spark.pyspark
emotion_distribution = rating_df.groupBy("emotion").count().orderBy("count", ascending=False)
emotion_distribution.show()
```

SPARK JOB FINISHED

```
+--------+-----+
| emotion|count|
+--------+-----+
|     joy|56056|
|   anger|24844|
|    love| 7526|
| sadness| 4422|
|surprise| 3733|
|    fear| 2863|
+--------+-----+
```

books-analysis

Took 4 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:19:33 PM.

Review/Score Distribution

FINISHED

Took 0 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:14:18 PM.

```
%spark.pyspark
from pyspark.sql.functions import col, when

# Convert review/score datatype from string to float
rating_df = rating_df.withColumn(
    "review/score",
    when(col("review/score").cast("float").isNotNull(), col("review/score").cast("float"))
    .otherwise(-1.0)
)

# Show the result
rating_df.show()
```

SPARK JOB (http://nyu-dataproc-w-1.c.hpc-dataproc-19b8.internal:44737/jobs/job?id=43) FINISHED

```
+----------+--------------------+------------+--------------------+-------+--------------------
--+--------------------+
|     row_id|               Title|review/score|         review/text|emotion|        cleaned_te
xt|    cleaned_text_new|
+----------+--------------------+------------+--------------------+-------+--------------------
--+--------------------+
|2576980377 6|Murder By The Boo...|         5.0|"(If you're inter...|  anger|if youre interes
t...|audio edition mic...|
|2576980377 7|Murder By The Boo...|         5.0|This doesn't seem...|    joy|this doesnt seem
...|known wolfe outst...|
|2576980377 8|Murder By The Boo...|         5.0|Rex Stout's Nero ...|    joy|rex stouts nero
w...|rex stouts nero w...|
|2576980377 9|Murder By The Boo...|         5.0|I loved just abou...|  anger|i loved just abo
u...|wistful infatuate...|
|2576980378 0|Murder By The Boo...|         5.0|If you are a Nero...|  anger|if you are a ner
o...|nero wolfe fanand...|
|2576980378 1|Murder By The Boo...|         4.0|In this story, Wo...|    joy|in this story wo
```

Took 0 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:19:33 PM.

```
%spark.pyspark
review_score_distribution = rating_df.groupBy("review/score").count().orderBy("count", ascend
review_score_distribution.show()
```

SPARK JOB FINISHED

```
+------------+-----+
|review/score|count|
+------------+-----+
|         5.0|55043|
|         4.0|21164|
|         3.0| 7796|
|         1.0| 7461|
|         2.0| 4930|
|        -1.0|  565|
+------------+-----+
```

books-analysis

Took 4 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:19:37 PM.

Emotion vs Rating Distribution

FINISHED

Took 0 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:14:23 PM.

1. Distribution of Rating for each Emotion

FINISHED

Took 0 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:14:25 PM.

```
%spark.pyspark
rating_distribution_by_emotion = result_df.groupBy("emotion", "review/score").count().orderBy(

# Show the results for each emotion
emotions = result_df.select("emotion").distinct().collect()
for emotion_row in emotions:
    emotion = emotion_row["emotion"]
    print(f"Distribution of Ratings for Emotion: {emotion}")
    rating_distribution_by_emotion.filter(col("emotion") == emotion).show()
```

SPARK JOB FINISHED

```
Distribution of Ratings for Emotion: joy
+-------+------------+-----+
|emotion|review/score|count|
+-------+------------+-----+
|    joy|        -1.0|   17|
|    joy|         1.0| 1239|
|    joy|         2.0| 1323|
|    joy|         3.0| 2572|
|    joy|         4.0| 5728|
|    joy|         5.0|13189|
+-------+------------+-----+

Distribution of Ratings for Emotion: love
+-------+------------+-----+
|emotion|review/score|count|
+-------+------------+-----+
|   love|        -1.0|    6|
```

Took 56 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:20:33 PM.

2. Distribution of Emotion for each Rating

FINISHED

Took 0 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:14:28 PM.

books-analysis

```
%spark.pyspark                                                                    SPARK JOB  FINISHED
emotion_distribution_by_rating = result_df.groupBy("review/score", "emotion").count().orderBy(

# Show the results for each rating
ratings = result_df.select("review/score").distinct().collect()
for rating_row in ratings:
    rating = rating_row["review/score"]
    print(f"Distribution of Emotions for Rating: {rating}")
    emotion_distribution_by_rating.filter(col("review/score") == rating).show()
```

```
Distribution of Emotions for Rating: 1.0
+------------+--------+-----+
|review/score| emotion|count|
+------------+--------+-----+
|         1.0|   anger| 1232|
|         1.0|    fear|  127|
|         1.0|     joy| 1239|
|         1.0|    love|  194|
|         1.0| sadness|  362|
|         1.0|surprise|   84|
+------------+--------+-----+

Distribution of Emotions for Rating: 5.0
+------------+--------+-----+
|review/score| emotion|count|
+------------+--------+-----+
|         5.0|   anger| 4657|
```

Took 57 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:21:30 PM.

3. Most Frequent Top Keywords in each Emotion

FINISHED

Took 0 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:14:33 PM.

```
%spark.pyspark                                                                    SPARK JOB  FINISHED
from pyspark.sql.window import Window
from pyspark.sql.functions import row_number

# Group by emotion and keyword to count occurrences
keyword_distribution = result_df.groupBy("emotion", "keyword").count()

# Use a window to rank the keywords for each emotion by count
window_spec = Window.partitionBy("emotion").orderBy(col("count").desc())

# Add a rank column and filter for the top 10 keywords per emotion
most_frequent_keywords = keyword_distribution.withColumn(
    "rank", row_number().over(window_spec)
).filter(col("rank") <= 10)

# Show the top 10 keywords for each emotion
print("Top 10 Keywords for Each Emotion")
most_frequent_keywords.orderBy("emotion", "rank").show(60, truncate=False)  # Show all 6*10=6(
```

```
Top 10 Keywords for Each Emotion
+--------+------------+-----+----+
|emotion |keyword     |count|rank|
+--------+------------+-----+----+
|anger   |art         |2080 |1   |
|anger   |war         |2037 |2   |
|anger   |love        |1461 |3   |
|anger   |history     |829  |4   |
|anger   |family      |760  |5   |
|anger   |death       |596  |6   |
|anger   |boring      |448  |7   |
|anger   |mystery     |351  |8   |
|anger   |religion    |259  |9   |
|anger   |entertaining|217  |10  |
|fear    |art         |579  |1   |
|fear    |war         |394  |2   |
|fear    |love        |286  |3   |
|fear    |family      |138  |4   |
```

books-analysis

Took 21 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:21:51 PM.

```
%spark.pyspark                                                                    SPARK JOB  FINISHED
# Calculate table to plot a heapmap that shows the relationship between top keywords and ratin
from pyspark.sql.functions import col, count, sum as _sum

# Generate table to plot the heapmap
top_keywords = [
    "family", "history", "war", "spiritual", "entertaining", "mystery", "death", "vampire", "g
    "fairy tale", "love", "robert jordan", "se hinton", "stephen king", "F. Scott Fitzgerald"
    "helen fielding", "kurt vonnegut", "lemony snicket", "lord rings", "american dream", "20th
    "civil war", "19th century", "boring", "disapointed", "4 stars", "blah blah blah"
]

# Filter the DataFrame for top keywords
filtered_df = result_df.filter(col("keyword").isin(top_keywords))

# Group by keyword and review score, and calculate counts
grouped_df = filtered_df.groupBy("keyword", "review/score").agg(count("*").alias("count"))

# Calculate total counts per keyword
total_counts_df = grouped_df.groupBy("keyword").agg(_sum("count").alias("total_count"))

# Join to calculate percentages
percentage_df = grouped_df.join(total_counts_df, "keyword")
percentage_df = percentage_df.withColumn("percentage", (col("count") / col("total_count")) * 1
percentage_df.write.mode("overwrite").csv("/user/tl4151_nyu_edu/heatmap_data_new.csv", header=
```

Took 22 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:22:13 PM.

```
%spark.pyspark                                                                    SPARK JOB  FINISHED
percentage_df.show(5)

+--------+------------+-----+-----------+------------------+
| keyword|review/score|count|total_count|        percentage|
+--------+------------+-----+-----------+------------------+
|religion|         4.0|  187|        867|21.568627450980394|
|religion|         2.0|   52|        867| 5.997693194925029|
|religion|         1.0|   74|        867| 8.535178777393309|
```

```
|religion|         3.0|   87|        867|10.034602076124568|
|religion|         5.0|  467|        867|  53.8638985005767|
+--------+------------+-----+-----------+------------------+
only showing top 5 rows
```

books-analysis

Took 20 sec. Last updated by tl4151_nyu_edu at December 10 2024, 5:22:33 PM.

```
%spark.pyspark
```

READY