Data Operations & Management

Pooja Sharma, Atlantic Technological University, L00171181

Task 1: Define scenario of your choice and Extract relevant unstructured data from any source (database, warehouse etc) and provide Evidence?

Github Link: https://github.com/poojavats/Spark_Analyzing_-Popular_-New-_York-_Times-Articles

Scenario: Analyzing Popular New York Times Articles.

I have Analysis on Analyzing Popular New York Times Articles, I have extract the data from below mentioned website by using Free API method and add the unstructured data(JSON Format) by using spark and Anaconda

Website-> https://developer.nytimes.com/

Tools and technology-> Spark, Anaconda(Jyupter Notebook), Python, Java

Task 2: Design complete ELT data pipeline using Apache Spark and analyse the data and get a meaningful insight from the data

Extract Data-> Extract data using Python and Spark:

Step 1: Launch a Python environment, such as Jupyter Notebook, using Anaconda.

Step 2: Import the required libraries and set up a SparkSession to work with Spark.

Step 3: Make API requests to the New York Times API using the requests library.

Step 4: Parse the JSON responses received from the API and extract the relevant data using Spark Library in Jupyter Notebook.

Step 5: Store the extracted data in an appropriate format (CSV) for further analysis**.**

Snippet of Unstructured data:

```
{"_id":{"$oid":"5b4aa4ead3089013507db18b"},"bestsellers_date":{"$date":{"
$numberLong":"1211587200000"}},"published_date":{"$date":{"$numberLong":"
1212883200000"}},"amazon_product_url":"http://www.amazon.com/Odd-Hours-
Dean-Koontz/dp/0553807056?tag=NYTBS-20","author":"Dean R
Koontz","description":"Odd Thomas, who can communicate with the dead,
confronts evil forces in a California coastal
town.","price":{"$numberInt":"27"},"publisher":"Bantam","title":"ODD
HOURS","rank":{"$numberInt":"1"},"rank_last_week":{"$numberInt":"0"},"wee
ks_on_list":{"$numberInt":"1"}}
{"_id":{"$oid":"5b4aa4ead3089013507db18c"},"bestsellers_date":{"$date":{"
$numberLong":"1211587200000"}},"published_date":{"$date":{"$numberLong":"
1212883200000"}},"amazon_product_url":"http://www.amazon.com/The-Host-
Novel-Stephenie-Meyer/dp/0316218502?tag=NYTBS-20","author":"Stephenie
Meyer","description":"Aliens have taken control of the minds and bodies
of most humans, but one woman won't
surrender.","price":{"$numberDouble":"25.99"},"publisher":"Little,
Brown","title":"THE
```


Fig 1: Snippet of Unstructured data

```
## Initialize the Spark environment in Python using the findspark library.

In [1]: import findspark
        findspark.init('C:\spark\spark-3.4.0-bin-hadoop3-scala2.13')

# Import the Python and pyspark Library

In [2]: import pandas as pd
        import numpy as np
        from datetime import date, timedelta, datetime
        import time

        import pyspark # only run this after findspark.init()
        from pyspark.sql import SparkSession, SQLContext
        from pyspark.context import SparkContext
        from pyspark.sql.functions import *
        from pyspark.sql.types import *

# Established a Connection to Spark Cluster

In [3]: from pyspark.sql import SparkSession

        spark = SparkSession.builder \
                .appName('Newyork_time Analysis') \
                .config('spark.executor.instances', '1') \
                .getOrCreate()
```

Fig 2. Spark and Library Setup

Step 6: Import the time Module

```
: import time

  start_time = time.time()
```

Fig 3: Time Module

This time module help me to calculate the total time for Analysis of my data.

Start_time=time.time()-> this code Assign the current time for the time measurement**.**

Step 7:

```
# import data
df = spark.read.json('NewYork.json')
```

Fig 4: *Import data*

Spark.read.json-> This method reads the json file name and inferes the schema also that is based on data, where my file name "NewYork.json"

Step 8: First 5 rows of the data frame.

```
df.show(5)
```

```
+--------------------+--------------------+----------------+----------------+--------------------+-------------+-------------
----+----+-------------+--------------------+-------------+--------------------+
|                 _id|                 URL|          author| bestsellers_date|         description|        price|   published_
date|    publisher|rank|rank_last_week|               title|weeks_on_list|      add_new_column|
+--------------------+--------------------+----------------+----------------+--------------------+-------------+-------------
----+----+-------------+--------------------+-------------+--------------------+
|{5b4aa4ead3089013...|http://www.amazon...|    Dean R Koontz|{{1211587200000}}|Odd Thomas, who c...|   {null, 27}|{{12128832000
00}}|       Bantam| {1}|           {0}|           ODD HOURS|          {1}|This is a new column|
|{5b4aa4ead3089013...|http://www.amazon...|  Stephenie Meyer|{{1211587200000}}|Aliens have taken...|{25.99, null}|{{12128832000
00}}|Little, Brown| {2}|           {1}|            THE HOST|          {3}|This is a new column|
|{5b4aa4ead3089013...|http://www.amazon...|     Emily Giffin|{{1211587200000}}|A woman's happy m...|{24.95, null}|{{12128832000
00}}| St. Martin's| {3}|           {2}|LOVE THE ONE YOU'...|          {2}|This is a new column|
|{5b4aa4ead3089013...|http://www.amazon...|Patricia Cornwell|{{1211587200000}}|A Massachusetts s...|{22.95, null}|{{12128832000
00}}|       Putnam| {4}|           {0}|           THE FRONT|          {1}|This is a new column|
|{5b4aa4ead3089013...|http://www.amazon...|  Chuck Palahniuk|{{1211587200000}}|An aging porn que...|{24.95, null}|{{12128832000
00}}|    Doubleday| {5}|           {0}|               SNUFF|          {1}|This is a new column|
+--------------------+--------------------+----------------+----------------+--------------------+-------------+-------------
----+----+-------------+--------------------+-------------+--------------------+
only showing top 5 rows
```

Fig 5: *display the 5 rows of data*

Step 9: Data Types of the Columns of DataFrame

```
: df.dtypes
```

```
: [('_id', 'struct<$oid:string>'),
   ('amazon_product_url', 'string'),
   ('author', 'string'),
   ('bestsellers_date', 'struct<$date:struct<$numberLong:string>>'),
   ('description', 'string'),
   ('price', 'struct<$numberDouble:string,$numberInt:string>'),
   ('published_date', 'struct<$date:struct<$numberLong:string>>'),
   ('publisher', 'string'),
   ('rank', 'struct<$numberInt:string>'),
   ('rank_last_week', 'struct<$numberInt:string>'),
   ('title', 'string'),
   ('weeks_on_list', 'struct<$numberInt:string>')]
```

Fig 6: display the datatypes of dataframe

Step 10: Statistical Summary of the Dataset

```
: df.describe().show()
```

```
+-------+--------------------+---------------+--------------------+---------+------------------+
|summary|  amazon_product_url|         author|         description|publisher|             title|
+-------+--------------------+---------------+--------------------+---------+------------------+
|  count|               10195|          10195|               10195|    10195|             10195|
|   mean|                null|           null|                null|     null|1877.7142857142858|
| stddev|                null|           null|                null|     null| 370.9760613506458|
|    min|http://www.amazon...|        AJ Finn|                    |      ACE|  10TH ANNIVERSARY|
|    max|https://www.amazo...|various authors|'Tis for the Rebe...|allantine|               ZOO|
+-------+--------------------+---------------+--------------------+---------+------------------+
```

Fig 7: Statistical Summary of the Dataset

Step 11: Calculate the count of distinct rows

```
df.distinct().count()
```

```
10195
```

Fig 8: distinct count

Step 12: Remove the Duplicates values

```
df_drop = df.dropDuplicates()
df_drop.show(5)
```

```
+--------------------+--------------------+--------------------+----------------+--------------------+------------+--------------------+
|                 _id|                 URL|              author| bestsellers_date|         description|       price|   published_date|       publisher|rank|rank_last_week|               title|weeks_on_list|      add_new_column|
+--------------------+--------------------+--------------------+----------------+--------------------+------------+--------------------+
|{5b4aa4ead3089013...|http://www.amazon...|James Patterson a...|{{1217030400000}}|A sailing vacatio...|{27.99, null}|{{1218326400000}}|   Little, Brown| {6}|           {6}|                SAIL|          {7}|This is a new column|
|{5b4aa4ead3089013...|http://www.amazon...|    David Wroblewski|{{1228521600000}}|A mute takes refu...|{25.95, null}|{{1229817600000}}|            Ecco| {4}|           {7}|THE STORY OF EDGA...|         {26}|This is a new column|
|{5b4aa4ead3089013...|http://www.amazon...|     Gregory Maguire|{{1230940800000}}|A looming civil w...|    {null, 0}|{{1232236800000}}|          Morrow|{18}|           {0}|    A LION AMONG MEN|          {0}|This is a new column|
|{5b4aa4ead3089013...|http://www.amazon...|       Sandra Brown|{{1257552000000}}|A woman runs a De...|    {null, 0}|{{1258848000000}}|Simon & Schuster|{17}|           {0}|           RAINWATER|          {0}|This is a new column|
|{5b4aa4ead3089013...|http://www.amazon...|        John Grisham|{{1259366400000}}|Stories set in ru...|   {null, 24}|{{1260662400000}}|       Doubleday| {6}|           {4}|         FORD COUNTY|          {4}|This is a new column|
+--------------------+--------------------+--------------------+----------------+--------------------+------------+--------------------+
only showing top 5 rows
```

Fig 9: remove duplicate value

df_drop = df.dropDuplicates() creates a new DataFrame df_drop by removing duplicate rows from the original DataFrame df**.**

Step 13: Shows the ten entries of "Author","Title" and "rank"

Where the df represent the dataframe and select is used to reprent the specific columns from the dataset.

```
df.select("author", "title", "rank").show(10)
```

```
+--------------------+--------------------+----+
|              author|               title|rank|
+--------------------+--------------------+----+
|       Dean R Koontz|           ODD HOURS| {1}|
|     Stephenie Meyer|            THE HOST| {2}|
|        Emily Giffin|LOVE THE ONE YOU'...| {3}|
|   Patricia Cornwell|           THE FRONT| {4}|
|     Chuck Palahniuk|               SNUFF| {5}|
|James Patterson a...|SUNDAYS AT TIFFANY'S| {6}|
|       John Sandford|        PHANTOM PREY| {7}|
|       Jimmy Buffett|          SWINE NOT?| {8}|
|    Elizabeth George|     CARELESS IN RED| {9}|
|      David Baldacci|     THE WHOLE TRUTH|{10}|
+--------------------+--------------------+----+
only showing top 10 rows
```

Fig 10: Shows entry of columns("author,title,rank")

Step 14: df.select("title", when(df.title != 'ODD HOURS', 1).otherwise(0)).show(10)

selects the "title" column from the DataFrame df and creates a new column that indicates whether each title is equal to 'ODD HOURS' or not. The result is displayed for the first 10 rows.

```
df.select("title", when(df.title != 'ODD HOURS', 1).otherwise(0)).show(10)
```

```
+--------------------+-------------------------------------------------------+
|               title|CASE WHEN (NOT (title = ODD HOURS)) THEN 1 ELSE 0 END|
+--------------------+-------------------------------------------------------+
|           ODD HOURS|                                                      0|
|            THE HOST|                                                      1|
|LOVE THE ONE YOU'...|                                                      1|
|           THE FRONT|                                                      1|
|               SNUFF|                                                      1|
|SUNDAYS AT TIFFANY'S|                                                      1|
|        PHANTOM PREY|                                                      1|
|          SWINE NOT?|                                                      1|
|     CARELESS IN RED|                                                      1|
|     THE WHOLE TRUTH|                                                      1|
+--------------------+-------------------------------------------------------+
only showing top 10 rows
```

Fig 11: Create New Columns to check the titles

Step 15: Based on step 13 and 14 pick the names of authors display the result

df[df.author.isin("Stephenie Meyer", "Emily Giffin")].show(5)-> where expression checks that if the values in the "author" column of the DataFrame match any of the specified values ("Stephenie Meyer" or "Emily Giffin").

```
df[df.author.isin("Stephenie Meyer", "Emily Giffin")].show(5)
```

```
+--------------------+--------------------+--------------+----------------+--------------------+-------------+--------------
--+-------------+----+--------------+--------------------+-------------+
|                 _id|  amazon_product_url|        author| bestsellers_date|         description|        price|   published_da
te|    publisher|rank|rank_last_week|               title|weeks_on_list|
+--------------------+--------------------+--------------+----------------+--------------------+-------------+--------------
--+-------------+----+--------------+--------------------+-------------+
|{5b4aa4ead3089013...|http://www.amazon...|Stephenie Meyer|{{1211587200000}}|Aliens have taken...|{25.99, null}|{{121288320000
0}}|Little, Brown| {2}|           {1}|          THE HOST|          {3}|
|{5b4aa4ead3089013...|http://www.amazon...|   Emily Giffin|{{1211587200000}}|A woman's happy m...|{24.95, null}|{{121288320000
0}}| St. Martin's| {3}|           {2}|LOVE THE ONE YOU'...|          {2}|
|{5b4aa4ead3089013...|http://www.amazon...|Stephenie Meyer|{{1212192000000}}|Aliens have taken...|{25.99, null}|{{121348800000
0}}|Little, Brown| {2}|           {2}|          THE HOST|          {4}|
|{5b4aa4ead3089013...|http://www.amazon...|   Emily Giffin|{{1212192000000}}|A woman's happy m...|{24.95, null}|{{121348800000
0}}| St. Martin's| {4}|           {3}|LOVE THE ONE YOU'...|          {3}|
|{5b4aa4ead3089013...|http://www.amazon...|Stephenie Meyer|{{1212796800000}}|Aliens have taken...|{25.99, null}|{{121409280000
0}}|Little, Brown| {2}|           {2}|          THE HOST|          {5}|
+--------------------+--------------------+--------------+----------------+--------------------+-------------+--------------
--+-------------+----+--------------+--------------------+-------------+
only showing top 5 rows
```

Fig 12: Display the author name with the data

Step 16: This code represents that show the author and title is true if the title has "the words" in titles.

```
df.select("author", "title", df.title.like("% THE %")).show(15)
```

```
+--------------------+--------------------+------------------+
|              author|               title|title LIKE % THE %|
+--------------------+--------------------+------------------+
|       Dean R Koontz|           ODD HOURS|             false|
|     Stephenie Meyer|            THE HOST|             false|
|        Emily Giffin|LOVE THE ONE YOU'...|              true|
|   Patricia Cornwell|           THE FRONT|             false|
|     Chuck Palahniuk|               SNUFF|             false|
|James Patterson a...|SUNDAYS AT TIFFANY'S|             false|
|       John Sandford|        PHANTOM PREY|             false|
|       Jimmy Buffett|          SWINE NOT?|             false|
|    Elizabeth George|     CARELESS IN RED|             false|
|      David Baldacci|     THE WHOLE TRUTH|             false|
|        Troy Denning|           INVINCIBLE|             false|
|          James Frey|BRIGHT SHINY MORNING|             false|
|         Garth Stein|THE ART OF RACING...|              true|
|     Debbie Macomber|       TWENTY WISHES|             false|
|         Jeff Shaara|      THE STEEL WAVE|             false|
+--------------------+--------------------+------------------+
only showing top 15 rows
```

Fig 13: Display the code of true and false condition based on "the" title.

Step 17: Remove Columns

```
df_remove = df.drop("publisher", "published_date").show(5)
```

```
+--------------------+--------------------+----------------+----------------+--------------------+-------------+----+--------
------+--------------------+-------------+--------------------+
|                 _id|                 URL|          author| bestsellers_date|         description|        price|rank|rank_las
t_week|               title|weeks_on_list|      add_new_column|
+--------------------+--------------------+----------------+----------------+--------------------+-------------+----+--------
------+--------------------+-------------+--------------------+
|{5b4aa4ead3089013...|http://www.amazon...|    Dean R Koontz|{{1211587200000}}|Odd Thomas, who c...|   {null, 27}| {1}|
{0}|           ODD HOURS|          {1}|This is a new column|
|{5b4aa4ead3089013...|http://www.amazon...|  Stephenie Meyer|{{1211587200000}}|Aliens have taken...|{25.99, null}| {2}|
{1}|            THE HOST|          {3}|This is a new column|
|{5b4aa4ead3089013...|http://www.amazon...|     Emily Giffin|{{1211587200000}}|A woman's happy m...|{24.95, null}| {3}|
{2}|LOVE THE ONE YOU'...|          {2}|This is a new column|
|{5b4aa4ead3089013...|http://www.amazon...|Patricia Cornwell|{{1211587200000}}|A Massachusetts s...|{22.95, null}| {4}|
{0}|           THE FRONT|          {1}|This is a new column|
|{5b4aa4ead3089013...|http://www.amazon...|  Chuck Palahniuk|{{1211587200000}}|An aging porn que...|{24.95, null}| {5}|
{0}|               SNUFF|          {1}|This is a new column|
+--------------------+--------------------+----------------+----------------+--------------------+-------------+----+--------
------+--------------------+-------------+--------------------+
only showing top 5 rows
```

Fig 14: Drop the publisher and published date columns

Step 18: Handling the missing values

```
# Replace null values using df.na.fill()
df = df.na.fill(0)  # Replace all null values in the DataFrame with 0

# Replace null values using df.fillna()
df = df.fillna(0)  # Replace all null values in the DataFrame with 0
```

Fig 15: handling the missing values

Step 19: Partitions the dataframe with 10 and 1 partitions.

Two different ways to control the number of partitions in a DataFrame in Apache Spark.

```
: # Dataframe with 10 partitions
  df.repartition(10).rdd.getNumPartitions()

  # Dataframe with 1 partition
  df.coalesce(1).rdd.getNumPartitions()
```

```
: 1
```

Fig 16: Partitions the dataframe

- df.repartition(10).rdd.getNumPartitions(): This code repartitions the DataFrame df into 10 partitions using the repartition() method. It redistributes the data across the specified number of partitions.
- The getNumPartitions() method called on the RDD (Resilient Distributed Dataset) representation of the DataFrame returns the number of partitions in the resulting RDD.
- df.coalesce(1).rdd.getNumPartitions(): This code reduces the number of partitions in the DataFrame df to 1 using the coalesce() method.
- repartition() is used to increase or decrease the number of partitions in a DataFrame by shuffling the data across partitions.
- coalesce() is used to decrease the number of partitions in a DataFrame by minimizing data movement and merging partitions whenever possible.

Note: The number of partitions affects how the data is distributed across the cluster and can impact parallelism and performance in Spark operations

Step 20: Converting dataframe into an RDD

The RDD represents the data in a distributed manner, allowing for parallel processing in Apache Spark. Converting a DataFrame into an RDD can be useful in scenarios where you need to apply RDD-specific operations or transformations that are not available directly on DataFrames.

rdd_convert = df.rdd

```
rdd_convert = df.rdd
```

Fig 17: Converting dataframe in to RDD

Step 21: Create Content of Dataframe in pandas

CSV_data=df.toPandas() and display the data in structed format using pandas.

```
: CSV_data.head()
```

:

	_id	URL	author	bestsellers_date	description	price	published_date	publisher	rank	rank_las
0	(5b4aa4ead3089013507db18b,)	http://www.amazon.com/Odd-Hours-Dean-Koontz/dp...	Dean R Koontz	((1211587200000,),)	Odd Thomas, who can communicate with the dead,...	(None, 27)	((1212883200000,),)	Bantam	(1,)	
1	(5b4aa4ead3089013507db18c,)	http://www.amazon.com/The-Host-Novel-Stephenie...	Stephenie Meyer	((1211587200000,),)	Aliens have taken control of the minds and bod...	(25.99, None)	((1212883200000,),)	Little, Brown	(2,)	
2	(5b4aa4ead3089013507db18d,)	http://www.amazon.com/Love-Youre-With-Emily-Gi...	Emily Giffin	((1211587200000,),)	A woman's happy marriage is shaken when she en...	(24.95, None)	((1212883200000,),)	St. Martin's	(3,)	

Fig 18: Display the dataframe in pandas

Step 22: statistical summary of pandas dataframe

```
CSV_data.describe()
```

	_id	URL	author	bestsellers_date	description	price	published_date	publisher	rank	rank_last
count	10195	10195	10195	10195	10195	10195	10195	10195	10195	
unique	10195	2329	738	529	2972	38	529	176	20	
top	(5b4aa4ead3089013507db18b,)	http://www.amazon.com/All-Light-We-Cannot-See/...	John Grisham	((1211587200000,),)		(None, 0)	((1212883200000,),)	Putnam	(1,)	
freq	1	141	226	20	246	6184	20	1061	529	

Fig 19: Statistical Summary of Pandas dataframe

Step 23: Display the Top Authors by Number of Books

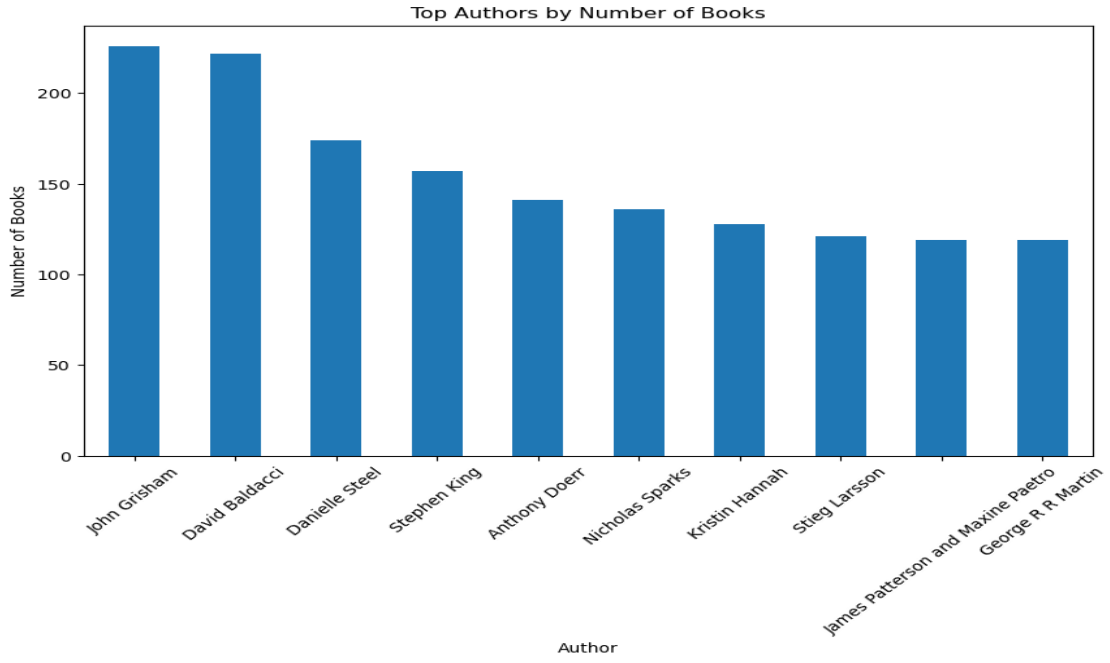

Fig 20: Top Authors by Numbers of books

Code Snippet:

```
import matplotlib.pyplot as plt

# Get the top authors with the highest numbe
top_authors = author_counts.head(10)  # Chan

# Plotting the top authors
plt.figure(figsize=(10, 6))
top_authors.plot(kind='bar')
plt.xlabel('Author')
plt.ylabel('Number of Books')
plt.title('Top Authors by Number of Books')
plt.xticks(rotation=45)
plt.show()
```

Fig 21: code for top authors by numbers of books

Step 24: As per Analysis we find that the Topmost Title-> "ALL THE LIGHT WE CANNOT SEE"
And top most Author-> "Anthony Doerr"
Code Snippet:

```
from collections import Counter


# Step 1: Count the occurrences of each title
title_counts = Counter(CSV_data['title'])

# Step 2: Find the topmost title(s)
top_titles = title_counts.most_common(1)

# Step 3: Filter books with the topmost title(s)
top_books = CSV_data[CSV_data['title'] == top_titles[0][0]]

# Step 4: Count the occurrences of each author in the top books
author_counts = Counter(top_books['author'])

# Step 5: Find the topmost author(s)
top_authors = author_counts.most_common(1)

# Print the results
print("Topmost Title:", top_titles[0][0])
print("Topmost Author:", top_authors[0][0])
```

```
Topmost Title: ALL THE LIGHT WE CANNOT SEE
Topmost Author: Anthony Doerr
```

Fig 22: Display the topmost title and author

Step 25: Represent the Author and Title relationship

Code Snippet:

```
# Step 6: Visualize the author-title relationship
authors = [author for author, _ in top_authors]
title_occurrences = [count for _, count in top_authors]

plt.bar(authors, title_occurrences)
plt.xlabel('Authors')
plt.ylabel('Number of Books')
plt.title('Author-Title Relationship (Top 10 Books)')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

Fig23: Represent the code for author and title relationship

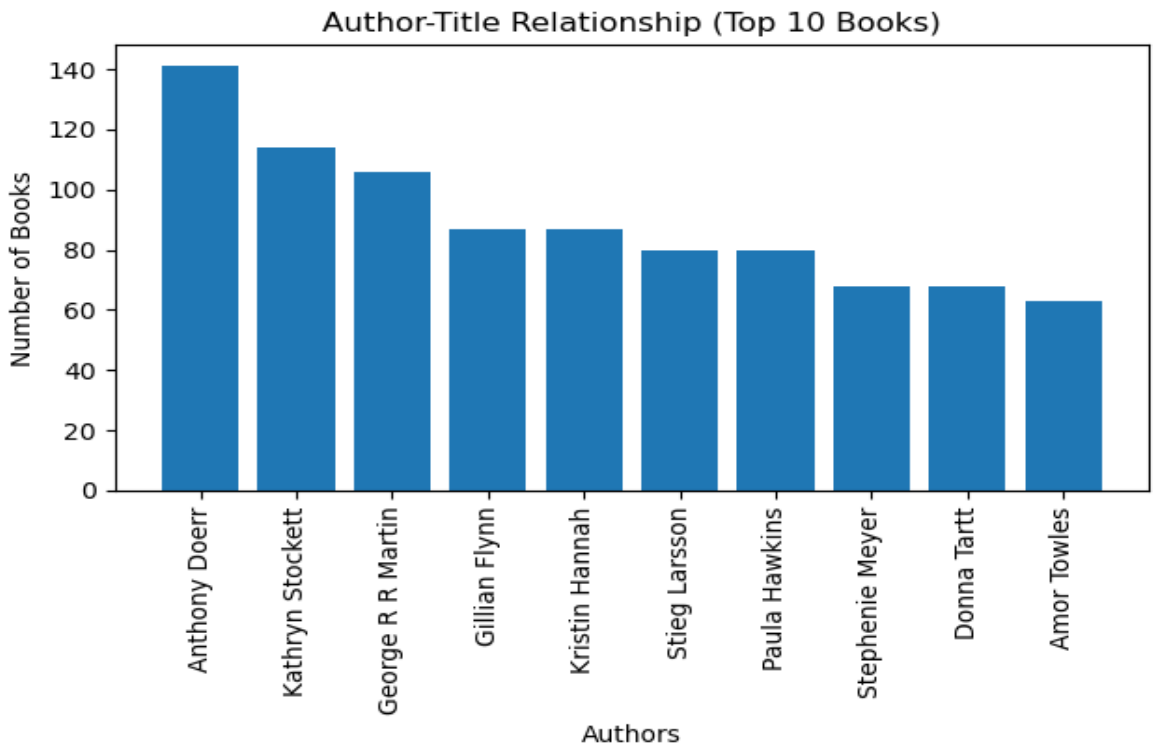

Fig24: Relationship of author and number of books

Task 4: scalability analysis:

Performance Analysis of Computer:

Fig25: Performance Analysis of computer

Instance 1

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
        .appName('Newyork_time Analysis') \
        .config('spark.executor.instances', '1') \
        .getOrCreate()
```

Fig 26: Create app name and showing the configuration of core

```
# End the timer
end_time = time.time()

# Print the total time
print("Processing time:", end_time - start_time, "seconds")
```

```
Processing time: 1376.6520583629608 seconds
```

Instance 1->

Fig 27: Total processing time of Instance 1

```
# End the timer
end_time = time.time()

# Print the total time
print("Processing time:", end_time - start_time, "seconds")
```

```
Processing time: 26.267320156097412 seconds
```

Instance 2:

Fig 28: Total Processing time of Instance 2

```
# End the timer
end_time = time.time()

# Print the total time
print("Processing time:", end_time - start_time, "seconds")
```

```
Processing time: 26.175273418426514 seconds
```

Instance 3:

Fig 29: Total processing time of instance 3

Conclusions:

The topmost title in the dataset is "**ALL THE LIGHT WE CANNOT SEE**" with the topmost author name being "**Anthony Doerr**". This suggests that "ALL THE LIGHT WE CANNOT SEE" is a popular

book in the dataset, and “**Anthony Doerr**“ is the author associated with it. Among all the authors in the dataset, “**John Grisham**” stands out with the highest number of books. He has a total of **250** books associated with his name. This indicates that Anthony Doerr is a prolific author within the dataset, having contributed significantly to the collection of books.

The total processing time of the dataset and analysis on Instance one was **1376.65 seconds**, on Instance 2 it was **26.26 seconds**, and on Instance 3 it was **26.17 seconds**. This suggests that Instance 1 had the longest processing time, while Instances 2 and 3 were significantly faster.

The difference in processing time across instances indicates variations in computational resources or workload distribution. Instance 1 might have lower processing capabilities or be overloaded with other tasks, resulting in slower performance. Instances 2 and 3, on the other hand, showed better performance and efficiency. It is important to consider the computational resources and workload distribution when performing data analysis. In summary, the conclusions highlight the popularity of the book "ALL THE LIGHT WE CANNOT SEE" by Anthony Doerr, the prolific nature of Anthony Doerr's contributions to the dataset, and the significance of selecting appropriate computational resources and optimizing workload distribution for efficient data processing and analysis.