

Processing and Analyzing Goodreads Data Using Google Cloud and Tableau

By Alayne Cross

Data Engineering Final Project CIS 399

August 30, 2024

Introduction

The goal of this project is to process a dataset containing a list of books from Goodreads, including information on each book, such as review totals and averages. The data is processed through a Google Cloud data pipeline, which stores the dataset in Bigtable (NoSQL). This pipeline also handles the input of new batches of ratings by identifying the affected titles and updating the total number and average of ratings in the dataset. This document details every step of the process, and key insights are visualized using Tableau.

Data Source

<https://www.kaggle.com/datasets/bahramjannesarr/goodreads-book-datasets-10m>

This dataset was selected because of its high ratings on Kaggle, which indicate that it will be less likely to incur errors.

The features included in the .csv file are:

- **bookID** - A unique Identification number for each book.
- **Title** - The name under which the book was published.
- **Authors** - Names of the authors of the book. Multiple authors are delimited with -.
- **Average_rating** - The average rating of the book received in total.
- **Isbn** - Another unique number to identify the book, the International Standard Book Number.
- **Isbn13** - A 13-digit ISBN to identify the book, instead of the standard 11-digit ISBN.
- **Language_code** - Helps understand what is the primary language of the book. For instance, eng is standard for English.
- **Num_pages** - Number of pages the book contains.
- **Ratings_count** - Total number of ratings the book received.
- **Text_reviews_count** - Total number of written text reviews the book received.

Google Cloud Pipeline Step 1: Uploading .CSV to BigTable

Creating a new project

I created a new google cloud account to ensure that I would not run into any budgeting issues with the account that I've been using for this coursework, which had already incurred \$202 in charges out of the complementary \$300 provided to the new account. This new Google Cloud account created a new project for me, called "My First Project".


Creating an Instance in Bigtable




Instances

[+ CREATE INSTANCE](#)

Bigtable is a fully managed, wide-column NoSQL database that offers low latency and replication for high availability. You can provision Bigtable instances for your workload, then use one of the Bigtable client libraries to develop applications. [Learn more](#)

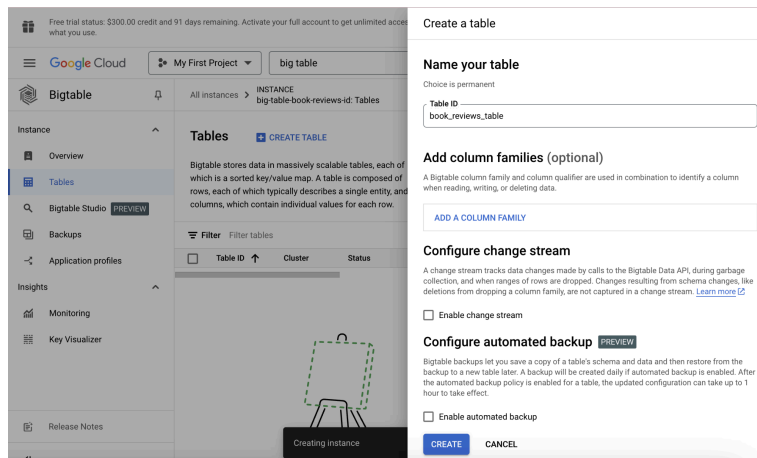
i You've got a new instance! Connect to it with the cbt command-line tool and create a table. [Learn more](#)

 **Filter** Filter instances

<input type="checkbox"/>	 Instance ID	Instance name 	Applicat
<input type="checkbox"/>	 big-table-book-reviews-id	big_table_book_reviews	default

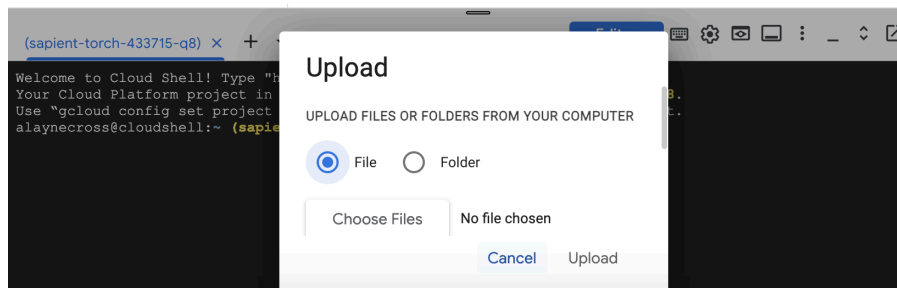
Creating a table

Then, I created a table to import the .CSV file to.



Uploading the .CSV

I uploaded the .CSV file to the Cloud Shell by using their upload tool.



Importing using Python Script

I needed to create a Python script to import the CSV to the BigTable, called import csv.py:

```
import csv
from google.cloud import bigtable
from google.api_core.exceptions import GoogleAPIError

# Initialize Bigtable client
project_id = 'sapient-torch-433715-q8'
instance_id = 'big-table-book-reviews-id'
table_id = 'books'
```

```

# Create a Bigtable client and instance reference
client = bigtable.Client(project=project_id, admin=True)
instance = client.instance(instance_id)
table = instance.table(table_id)

# Open the CSV file and read the data
print("Opening CSV file...")
try:
    with open('books_fixed.csv', mode='r', encoding='utf-8') as csvfile:
        reader = csv.DictReader(csvfile)
        for row in reader:
            print(f"Processing row: {row}") # Log the row being processed
            row_key = row['bookID']
            data = {
                'details': {
                    'title': row['title'],
                    'authors': row['authors'],
                    'average_rating': row['average_rating'],
                    'isbn': row['isbn'],
                    'isbn13': row['isbn13'],
                    'language_code': row['language_code'],
                    'num_pages': row.get('num_pages', 'N/A'), # Use 'N/A'
                }
            }

            if not present:
                'ratings_count': row['ratings_count'],
                'text_reviews_count': row['text_reviews_count'],
                'publication_date': row['publication_date'],
                'publisher': row['publisher'],
            }

            # Create a row object
            row_to_save = table.row(row_key)
            # Set the cells in the row
            for column, value in data['details'].items():
                # Encode the value to bytes
                row_to_save.set_cell('details', column,
str(value).encode('utf-8'))

            # Commit the changes to Bigtable
            row_to_save.commit()

```

```

        print(f"Successfully saved row with key: {row_key}") # Log
        successful save

    except FileNotFoundError:
        print("Error: The specified CSV file was not found.")
    except GoogleAPIError as api_error:
        print(f"Google API Error: {api_error}")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

# Read and print the data after importing
print("Reading imported data...")
try:
    # Correct the table reference
    rows = table.read_rows()
    for row in rows:
        print(row)
except GoogleAPIError as api_error:
    print(f"Google API Error while reading data: {api_error}")
except Exception as e:
    print(f"An unexpected error occurred while reading data: {e}")

print("Finished reading data.")

```

Debugging

There were some errors with one of the column names. After reviewing the original text file, I found that there was a line return just before that column. I removed it and reuploaded the .csv, and the table was successfully uploaded.

```

Successfully saved row with key: 45639
Processing row: {'bookID': '45641', 'title': 'Las aventuras de Tom Sawyer', 'authors': 'Mark Twain', 'average_rating': '3.91', 'isbn': '8497646983', 'isbn13': '9788497646987', 'language_code': 'spa', 'num_pages': '272', 'ratings_count': '113', 'text_reviews_count': '12', 'publication_date': '5/28/2006', 'publisher': 'Edimat Libros'}
Successfully saved row with key: 45641
Reading imported data...
Finished reading data.
alayneccross@cloudshell:~ (sapiant-torch-433715-q8) $

```

Success

The books table is uploaded to Bigtable.

Bigtable

TABLE

Overview

Overview

Table ID

books

Status

Ready

Storage utilization

3.3 MB

Automated backup

Not enabled

Enable

Change stream

Disabled

Deletion protection

Disabled

Authorized views

LIST AUTHORIZED VIEWS IN THE CLI

Using the Bigtable builder, the data can be viewed in a table.

Builder

Editor

Builder

Editor

RUN

default

CLEAR

DOCUMENTATION

Table

books

Clause

Limit

Limit *

100

ADD TO QUERY

RESULTS

Show timestamps

Rows per page:

20

1 – 20 of 100

<

>

Google Cloud Pipeline Step 2: Processing a batch of new reviews and adding them to the table

Generating new reviews

I created a new csv file with a set of ratings.

```
new_reviews.csv
1  bookID,userID,rating,timestamp
2  10,user_1,2,2024-08-07T15:30:34.039897
3  139,user_2,2,2024-08-11T15:30:34.039918
4  141,user_3,4,2024-08-13T15:30:34.039924
5  250,user_4,3,2024-08-19T15:30:34.039930
6  285,user_5,1,2024-08-03T15:30:34.039936
7  397,user_6,3,2024-08-15T15:30:34.039942
8  420,user_7,3,2024-08-01T15:30:34.039947
9  463,user_8,1,2024-08-18T15:30:34.039954
10 497,user_9,5,2024-07-29T15:30:34.039961
11 511,user_10,1,2024-08-21T15:30:34.039967
12
```

Only the bookID and rating values will need to be used to change the original table, but the other fields are included because those would be included in a real-life scenario.

Python Script to Process Reviews

I created a python script to do this, called process_new_reviews.py:

```
import csv

from google.cloud import bigtable
from datetime import datetime

# Initialize Bigtable client
project_id = 'sapient-torch-433715-q8'
instance_id = 'big-table-book-reviews-id'
table_id = 'new_reviews_table'

client = bigtable.Client(project=project_id, admin=True)
instance = client.instance(instance_id)
table = instance.table(table_id)
```



```

# Function to fetch current book data from the original table
def fetch_current_book_data(book_id):
    original_table = instance.table('books')
    row = original_table.read_row(book_id.encode('utf-8'))
    if row:
        # Log the fetched row data for debugging
        print(f"Fetch row data for bookID {book_id}: {row.cells}")

        average_rating = float(row.cells['details'].get('average_rating',
[b'0.0'])[0].decode('utf-8'))
        ratings_count = int(row.cells['details'].get('ratings_count',
[b'0'])[0].decode('utf-8'))
        return {'average_rating': average_rating, 'ratings_count':
ratings_count}
    else:
        print(f"No data found for bookID {book_id}")
        return {'average_rating': 0.0, 'ratings_count': 0}

# Create the new table with the necessary column families
print(f"Creating new table {table_id}...")
table.create(column_families={'details': None})

# Copy all rows from the 'books' table to the 'new_reviews_table'
original_table = instance.table('books')
rows = original_table.read_rows()
for row in rows:
    # Copy each row to the new table
    new_row = table.row(row.row_key)
    for family, columns in row.cells.items():
        for column, cells in columns.items():
            for cell in cells:
                new_row.set_cell(family, column, cell.value)
    new_row.commit()
print(f"All rows from 'books' copied to '{table_id}'.")

# Open the new reviews CSV file and apply updates
with open('new_reviews.csv', mode='r', encoding='utf-8') as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader:
        # The format is as follows: bookID, userID, rating, timestamp
        book_id = row['bookID']

```

```

user_id = row['userID']
rating = float(row['rating'])
timestamp = datetime.fromisoformat(row['timestamp'])

# Fetch current data from the original table
current_data = fetch_current_book_data(book_id)

# Update the average rating and count
new_count = current_data['ratings_count'] + 1
new_average = ((current_data['average_rating'] *
current_data['ratings_count']) + rating) / new_count

# Save updated data back to Bigtable in the 'details' column
family
row_key = f"{book_id}".encode('utf-8')
row = table.row(row_key)
row.set_cell('details', 'average_rating',
str(new_average).encode('utf-8'))
row.set_cell('details', 'ratings_count',
str(new_count).encode('utf-8'))

# Optionally, you can log review data in a separate structure or
keep it in an external system
row.commit()

print(f"Updated bookID {book_id} with new review by {user_id}")

print("All data imported successfully.")

```

Success

```

n13', [<Cell value=b'9780822549208' timestamp=2024-08-27 14:44:51.739000+00:00>]], (b'language_code', [<Cell value=b'eng' timestamp=2024-08-27 14:44:51.739000+00:00>]), (b'num
m_pages', [<Cell value=b'144' timestamp=2024-08-27 14:44:51.739000+00:00>]), (b'publication_date', [<Cell value=b'10/1/1994' timestamp=2024-08-27 14:44:51.739000+00:00>]), (b
'publisher', [<Cell value=b'Lerner Publications' timestamp=2024-08-27 14:44:51.739000+00:00>]), (b'ratings_count', [<Cell value=b'105' timestamp=2024-08-27 14:44:51.739000+00
:00>]), (b'text_reviews_count', [<Cell value=b'5' timestamp=2024-08-27 14:44:51.739000+00:00>]), (b'title', [<Cell value=b'Nikola Tesla: A Spark of Genius' timestamp=2024-08-
27 14:44:51.739000+00:00>]]))]]
Updated bookID 497 with new review by user_9
Fetched row data for bookID 511: OrderedDict([(b'details', OrderedDict([(b'authors', [<Cell value=b'Julie Elizabeth Leto/Leslie Kelly/Kimberly Raye' timestamp=2024-08-27 14:44
:51.754000+00:00>]), (b'average_rating', [<Cell value=b'3.77' timestamp=2024-08-27 14:44:51.754000+00:00>]), (b'isbn', [<Cell value=b'0373792689' timestamp=2024-08-27 14:44:5
1.754000+00:00>]), (b'isbn13', [<Cell value=b'9780373792689' timestamp=2024-08-27 14:44:51.754000+00:00>]), (b'language_code', [<Cell value=b'eng' timestamp=2024-08-27 14:44:
51.754000+00:00>]), (b'num_pages', [<Cell value=b'249' timestamp=2024-08-27 14:44:51.754000+00:00>]), (b'publication_date', [<Cell value=b'6/27/2006' timestamp=2024-08-27 14:
44:51.754000+00:00>]), (b'publisher', [<Cell value=b'Harlequin Blaze' timestamp=2024-08-27 14:44:51.754000+00:00>]), (b'ratings_count', [<Cell value=b'478' timestamp=2024-08-
27 14:44:51.754000+00:00>]), (b'text_reviews_count', [<Cell value=b'12' timestamp=2024-08-27 14:44:51.754000+00:00>]), (b'title', [<Cell value=b'Boys of Summer' timestamp=202
4-08-27 14:44:51.754000+00:00>]]))]]
Updated bookID 511 with new review by user_10
All data imported successfully.
alainecross@cloudshell:~ (sapient-torch-433715-q8) $

```

Running the code resulted in a new table with updated values for the 10 titles. The values of the new table can be previewed in BigTable Studio.

Explorer

Tables 3

- book_reviews_table
- books
- new_reviews_table**

Builder

Table

new_reviews_table

Clause

Limit

Limit *

100

ADD TO QUERY

RESULTS

Show timestamps

Row key	details: authors	details: average_rating	details: isbn	details: isbn13	details: language_code	
1	J.K. Rowling/M...	4.57	0439785960	9780439785969	eng	▼
10	J.K. Rowling	2.0	0439827604	9780439827607	eng	▼
100	Heidi Boyd	3.78	1581805632	9781581805635	en-US	▼
10000	Kôbô Abe/E. Da...	3.78	0375726535	9780375726538	eng	▼
10002	Kôbô Abe/Don...	3.91	0231082819	9780231082815	eng	▼
10004	Kôbô Abe/Julie...	3.61	0375726543	9780375726545	eng	▼
10006	Paul Auster	3.78	0965913228	9780965913225	eng	
10008	Douglas Coupla...	3.65	1582346437	9781582346434	eng	▼

Rows per page: 20

1 - 20 of 100

Using Dataflow to Move Table from BigTable to BigQuery

I determined that it was necessary to move the dataset from BigTable to BigQuery to fulfill the project requirement of creating a data pipeline. Unfortunately, as shown in the following screenshots, this process encountered some errors and ultimately did not work as expected. However, I hope that the work I completed above (i.e. the processing of new reviews) and the steps I took to create this pipeline, will demonstrate my efforts and satisfy the requirement.

There is a guide on Google Cloud for creating this exact job.

Bigtable > Documentation > Guides

Was this helpful?  

Use the Bigtable change stream to BigQuery template

[Send feedback](#)

In this quickstart, you learn how to set up a Bigtable table with a change stream enabled, run a change stream pipeline, make changes to your table, and then see the changes streamed.

Before you begin

1. In the Google Cloud console, on the project selector page, select or create a Google Cloud project.

[Go to project selector](#)

2. [Make sure that billing is enabled for your Google Cloud project.](#)

3. Enable the Dataflow, Cloud Bigtable API, Cloud Bigtable Admin API, and BigQuery APIs.

[Enable the APIs](#)

4. In one of the following development environments, set up the gcloud CLI:


- **Cloud Shell:** to use an online terminal with the gcloud CLI already set up, activate Cloud Shell.


[Activate Cloud Shell on this page](#)

- **Local shell:** to use a local development environment, [install](#) and [initialize](#) the gcloud CLI.

I enabled the required APIs:

Enable access to APIs

 Confirm project

 Enable APIs

You are about to enable:

Dataflow API
Cloud Bigtable API
Cloud Bigtable Admin API
BigQuery API

[ENABLE](#)

I selected the “Bigtable change streams to BigQuery” template:

Streaming templates

Templates for processing data continuously:

- [Apache Kafka to Apache Kafka](#)
- [Apache Kafka to BigQuery](#)
- [Apache Kafka to Cloud Storage](#)
- [Change Data Capture from MySQL to BigQuery \(Stream\)](#)
- [Bigtable change streams to BigQuery](#)
- [Bigtable change streams to Pub/Sub](#)
- [Spanner change streams to BigQuery](#)
- [Spanner change streams to Cloud Storage](#)
- [Spanner change streams to Pub/Sub](#)
- [Datastream to BigQuery \(Stream\)](#)

I needed to create an “app profile” with the ability to connect to the Bigtable instance:

```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to sapient-torch-433715-q8.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
alayneccross@cloudshell:~ (sapient-torch-433715-q8)$ gcloud bigtable app-profiles create bigquery-export-profile \
--instance=big-table-book-reviews-id \
--description="Profile for exporting data to BigQuery" \
--route-to=big-table-book-reviews-id-cl \
--transactional-writes
Created app profile [bigquery-export-profile].
alayneccross@cloudshell:~ (sapient-torch-433715-q8)$
```

I created a dataset to be the destination for in BigQuery:

The screenshot shows the Google Cloud BigQuery console interface. On the left, the 'Explorer' sidebar displays a tree view of resources under the project 'sapient-torch-433715-q8'. The 'bigquerybookreviewstable' dataset is highlighted. The main panel shows the 'Dataset info' for 'bigquerybookreviewstable'. The dataset ID is 'sapient-torch-433715-q8.bigquerybookreviewstable'. It was created on August 27, 2024, at 1:46:55 PM UTC-4. The default table expiration is 'Never'. The last modified date is also August 27, 2024, at 1:46:55 PM UTC-4. The data location is 'US'. The description is empty. The default collation is 'Default collation'. The default rounding mode is 'ROUNDING_MODE_UNSPECIFIED'. The case insensitive flag is 'false'. There are no labels or tags. Below the dataset info, the 'Dataset replica info' section shows the primary location as 'US'.

Dataset info	
Dataset ID	sapient-torch-433715-q8.bigquerybookreviewstable
Created	Aug 27, 2024, 1:46:55 PM UTC-4
Default table expiration	Never
Last modified	Aug 27, 2024, 1:46:55 PM UTC-4
Data location	US
Description	
Default collation	Default collation
Default rounding mode	ROUNDING_MODE_UNSPECIFIED
Case insensitive	false
Labels	
Tags	

Dataset replica info	
Primary location	US

I set that dataset as the target:

Job name *
bigtable-2-biqq

Must be unique among running jobs

Regional endpoint *
us-central1 (Iowa)

Choose a Dataflow regional endpoint to deploy worker instances and store job metadata. You can optionally deploy worker instances to any available Google Cloud region or zone by using the worker region or worker zone parameters. Job metadata is always stored in the Dataflow regional endpoint. [Learn more](#)

Dataflow template *
Cloud Bigtable Change Streams to BigQuery

Streaming pipeline. Streams Bigtable data change records and writes them into BigQuery using Dataflow Runner V2.

Target

BigQuery dataset *
sapient-torch-433715-q8:bigtable_exports.new_reviews

The dataset name of the destination BigQuery table.

OPTIONAL TARGET PARAMETERS

When running this job, I ran into a few errors like "Template launch failed" and "Exception in thread 'main'" with no specific error details provided.

The job appeared to queue but never successfully executed, so there was a failure in transferring the data to BigQuery, which I can't overcome.

books-2-bigq

STOP

+ IMPORT AS PIPELINE

SEND FEEDBACK

JOB GRAPH

EXECUTION DETAILS

JOB METRICS

COST

RECOMMENDATIONS

Graph view

Stage workflow

Critical path

Logs

HIDE

1

JOB LOGS

WORKER LOGS

DIAGNOSTICS

DATA SAMPLING

Severity

Error

Filter

Search all fields and values

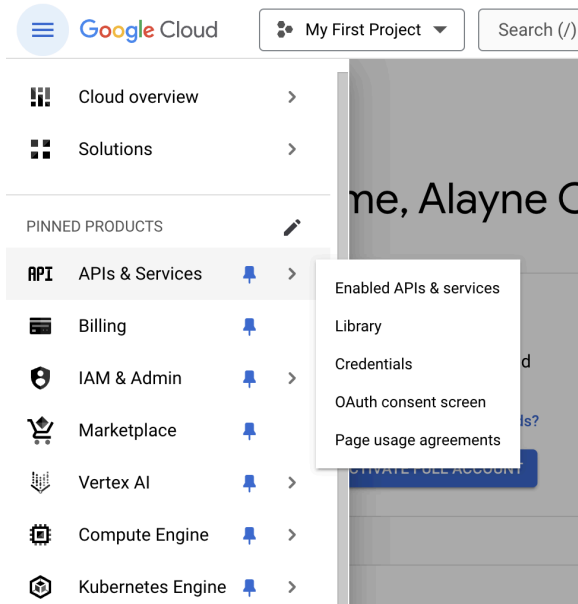
2:16 PM - 2:20 PM

SEVERITY	TIMESTAMP	SUMMARY
		No older entries found matching current filter.
	2024-08-27 14:17:55.289 EDT	Exception in thread "main" ; see consecutive INFO logs for details.
		<div>Explain this log entry</div> <div>Open in Logs Explorer</div>
		<pre>{ insertId: "4569592807989421798:652905:0:8955" jsonPayload: {2} labels: {6} logName: "projects/sapient-torch-433715-q8/logs/dataflow.googleapis.com%2Flauncher" receiveTimestamp: "2024-08-27T18:18:00.242398819Z" resource: {2} severity: "ERROR" timestamp: "2024-08-27T18:17:55.289952Z" }</pre>
	2024-08-27 14:17:55.315 EDT	Error: Template launch failed: exit status 1
		<div>Explain this log entry</div> <div>Open in Logs Explorer</div>
		<pre>{ insertId: "4569592807989421798:652905:0:10280" jsonPayload: {2} labels: {6} logName: "projects/sapient-torch-433715-q8/logs/dataflow.googleapis.com%2Flauncher" receiveTimestamp: "2024-08-27T18:18:00.242398819Z" resource: {2} }</pre>

Preparing to Use Looker Studio: Uploading the CSV to BigQuery

The errors prevented me from transferring the data from BigTable to BigQuery automatically, but it's still necessary to have the data in BigQuery in order to create the visualizations with Google's Looker Studio.

First, I enabled the BigQueryAPI:



Then, I create a bucket for .CSV file storage:

← Create a bucket

✓ **Name your bucket**
Name: book-review-final-project-bucket

✓ **Choose where to store your data**
Location: us (multiple regions in United States)
Location type: Multi-region

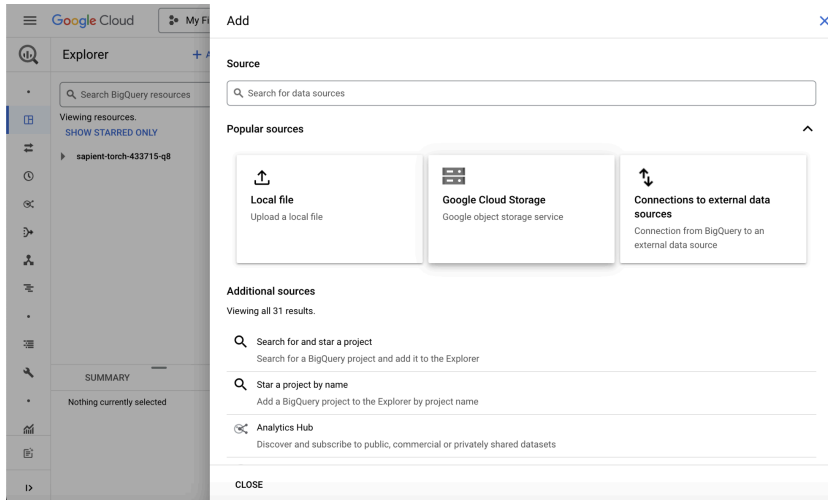
✓ **Choose a storage class for your data**
Default storage class: Standard

✓ **Choose how to control access to objects**
Public access prevention: Off
Access control: Uniform

✓ **Choose how to protect object data**
Custom soft delete policy: Disabled
Object versioning: Disabled
Bucket retention policy: Disabled
Object retention: Disabled
Encryption type: Google-managed

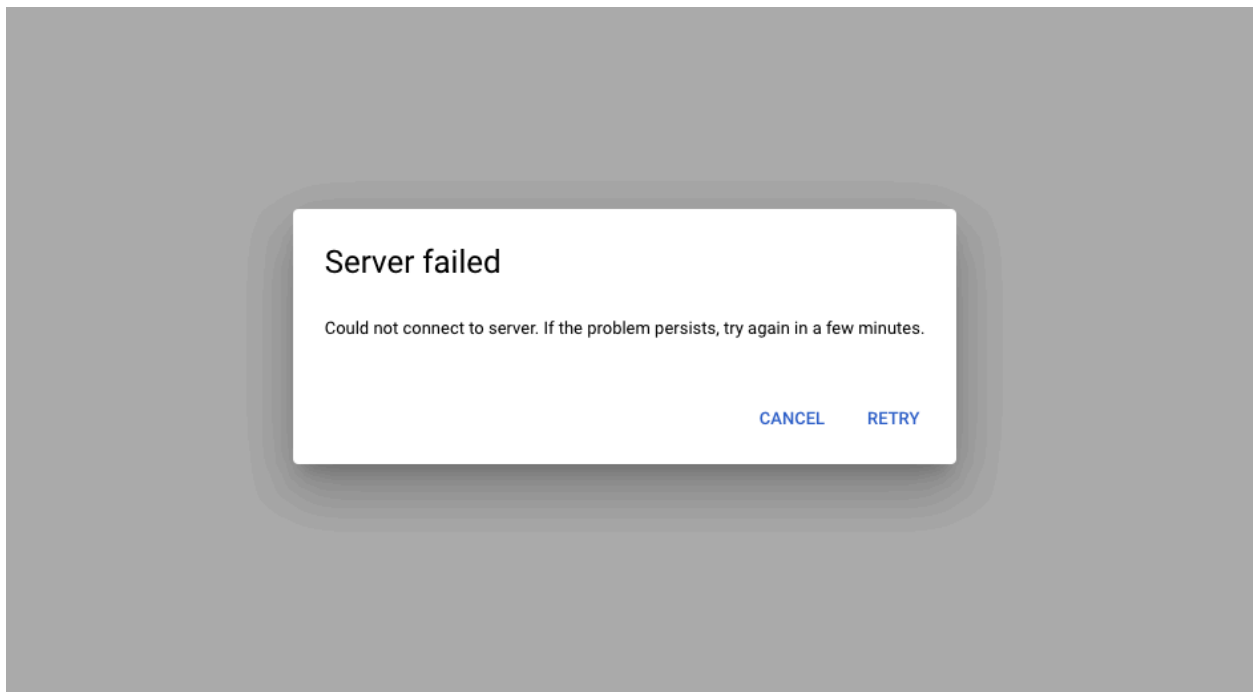
PROCESSING... CANCEL

Then I upload the CSV into BigQuery.



Visualization in LookerStudio

LookerStudio would not work due to a server failure.



Pivoting to Tableau

I downloaded Tableau and uploaded the dataset.

The screenshot shows the Tableau Desktop interface with a data source named 'books_fixed' loaded. The interface is divided into several sections:

- Connections:** A sidebar on the left showing the 'books_fixed' connection as a 'Text file'.
- Files:** A sidebar on the left showing the 'books_fixed.csv' file.
- Main View:** The central area displays the data source 'books_fixed.csv' with 12 fields and 11127 rows. A 'Go to Worksheet' button is visible.
- Fields List:** A table on the left side of the main view lists the fields in the data source:

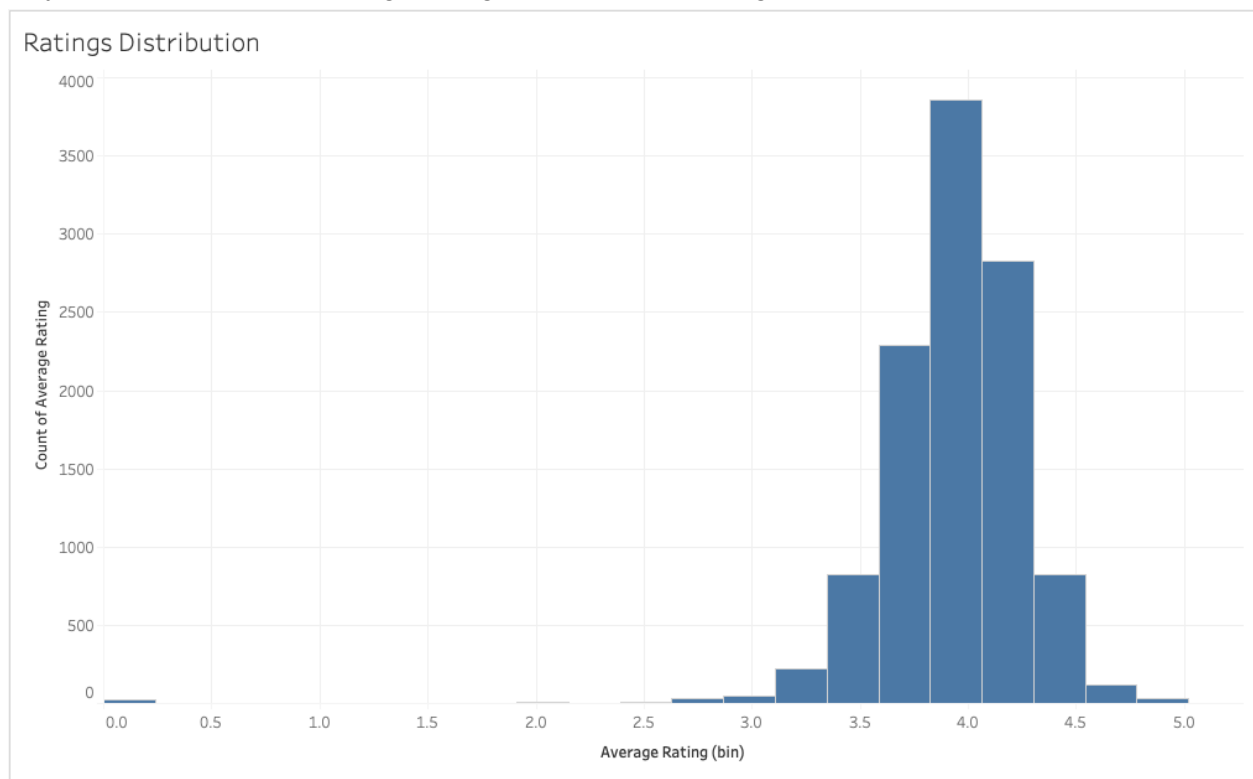
Type	Field Name	Physical Table	Remote ...
#	Book ID	books_fixed.csv	bookID
Abc	Title	books_fixed.csv	title
Abc	Authors	books_fixed.csv	authors

- Data Preview:** A table on the right side of the main view shows the first few rows of the data source:

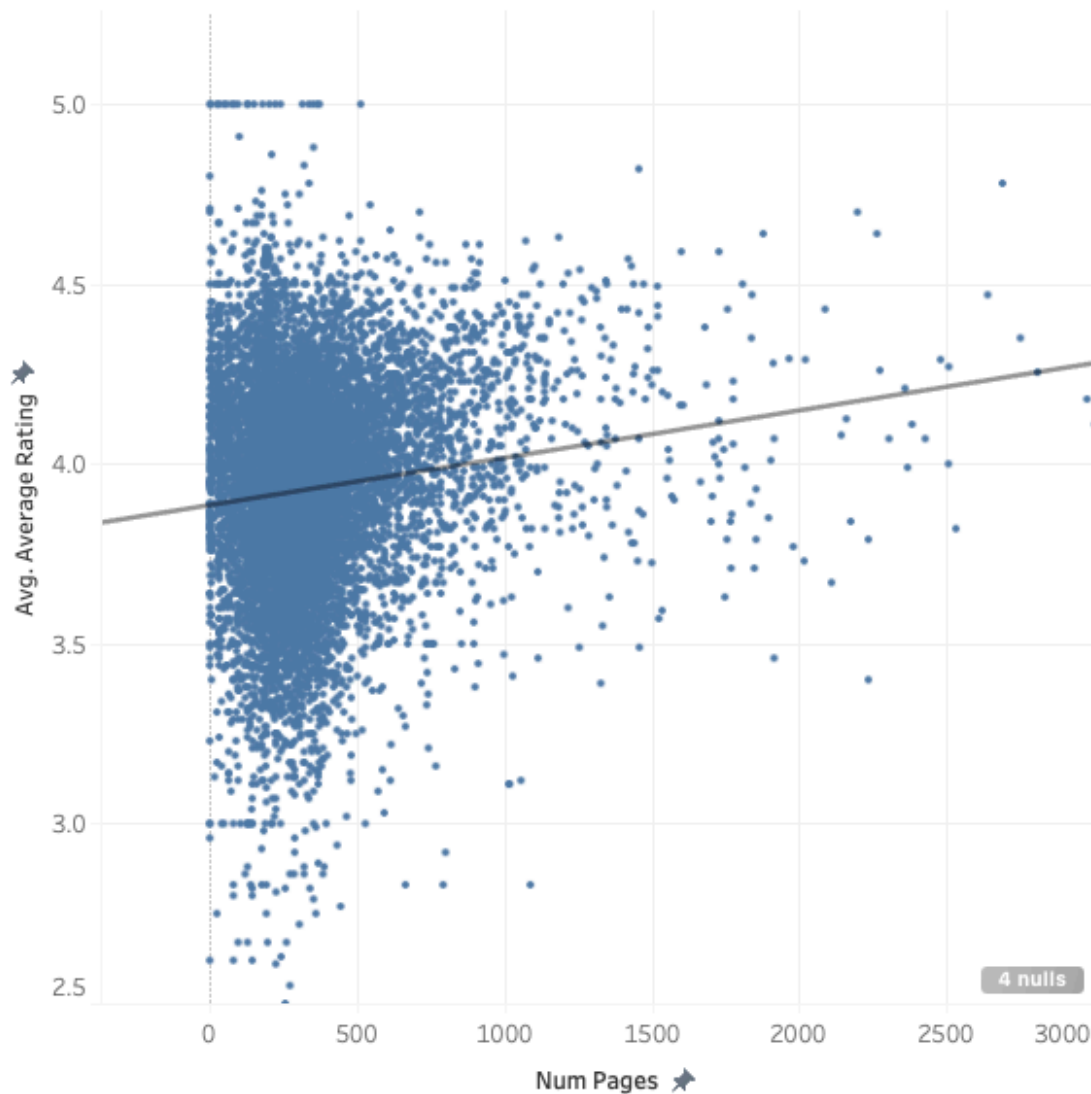
#	Abc books_fixed.csv Book ID	Abc books_fixed.csv Title	Abc books_fixed.csv Authors	# books_fixed.csv Average Rating	Abc books_fixed.csv Isbn
1	1	Harry Potter and the Half-Blo...	J.K. Rowling/Mary GrandPré	4.57000	0439785960
2	2	Harry Potter and the Order of...	J.K. Rowling/Mary GrandPré	4.49000	0439358078
4	4	Harry Potter and the Chamb...	J.K. Rowling	4.42000	0439554896
5	5	Harry Potter and the Prisone...	J.K. Rowling/Mary GrandPré	4.56000	043965548X
8	8	Harry Potter Boxed Set Book...	J.K. Rowling/Mary GrandPré	4.78000	0439682584
9	9	Unauthorized Harry Potter B...	W. Frederick Zimmerman	3.74000	0976540606

The bottom bar shows the 'Data Source' tab and the 'Sheet 1' tab.

Creating visualizations was a simple process that provided insights into the data. For the ratings distribution, this visualization shows that the average rating is quite high, and there are relatively very few books with an average rating lower than 3.5 or higher than 4.5 stars.

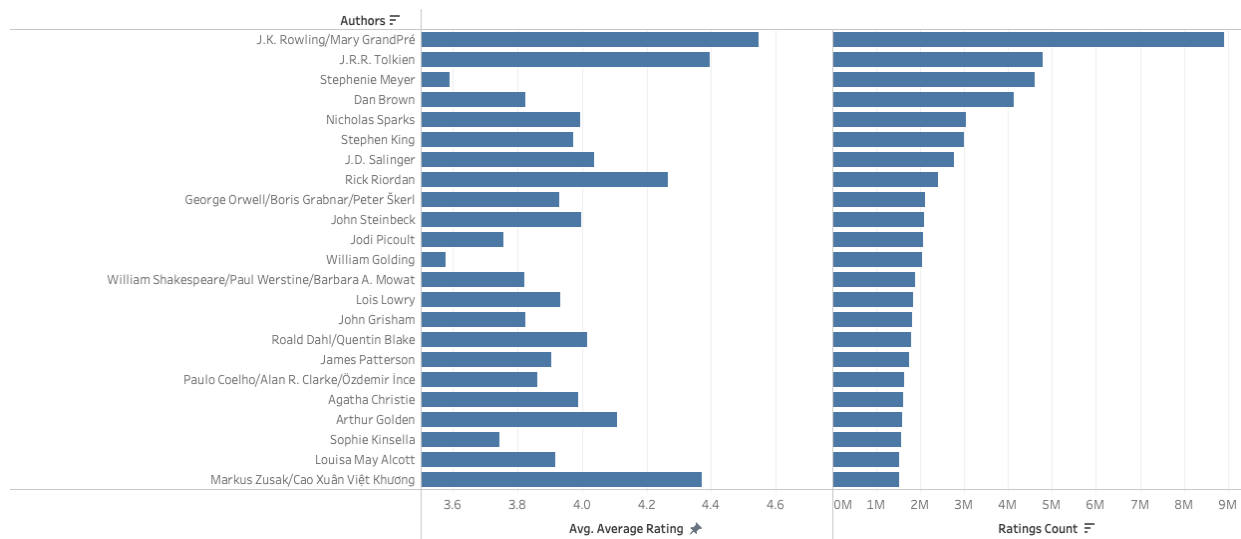


Ratings vs. Book Length



There is correlation between the number of pages and the average rating. This might indicate that readers have more appreciation for books when they've invested more time reading them, or only those that enjoy a long book will finish it and therefore will rate it more highly, or that longer books tend to be of higher quality, but we can't know the cause from this data alone.

Author Average Ratings by Ratings Count



This visualization shows the authors with the most ratings and the average ratings they have across all of their books. The X axis for average rating is cropped to show the variance in ratings because most reviews are within the 3.5 - 4.5 stars range.

Conclusion

This project successfully accomplished all the core requirements, despite encountering and overcoming several challenges along the way. The key obstacles included issues with the CSV file, difficulties with the Dataflow process for moving data from BigTable to BigQuery, and challenges with Google Cloud Looker Studio. The CSV file was corrected, ensuring accurate data processing, and Tableau was used as an alternative to Looker Studio for data visualization.

Although the Dataflow process did not fully succeed in transferring data from BigTable to BigQuery, this step was ultimately not critical to the overall success of the project. The pipeline successfully processed new reviews, and the visualizations in Tableau provided valuable insights from the dataset.