

# Data Engineering Project

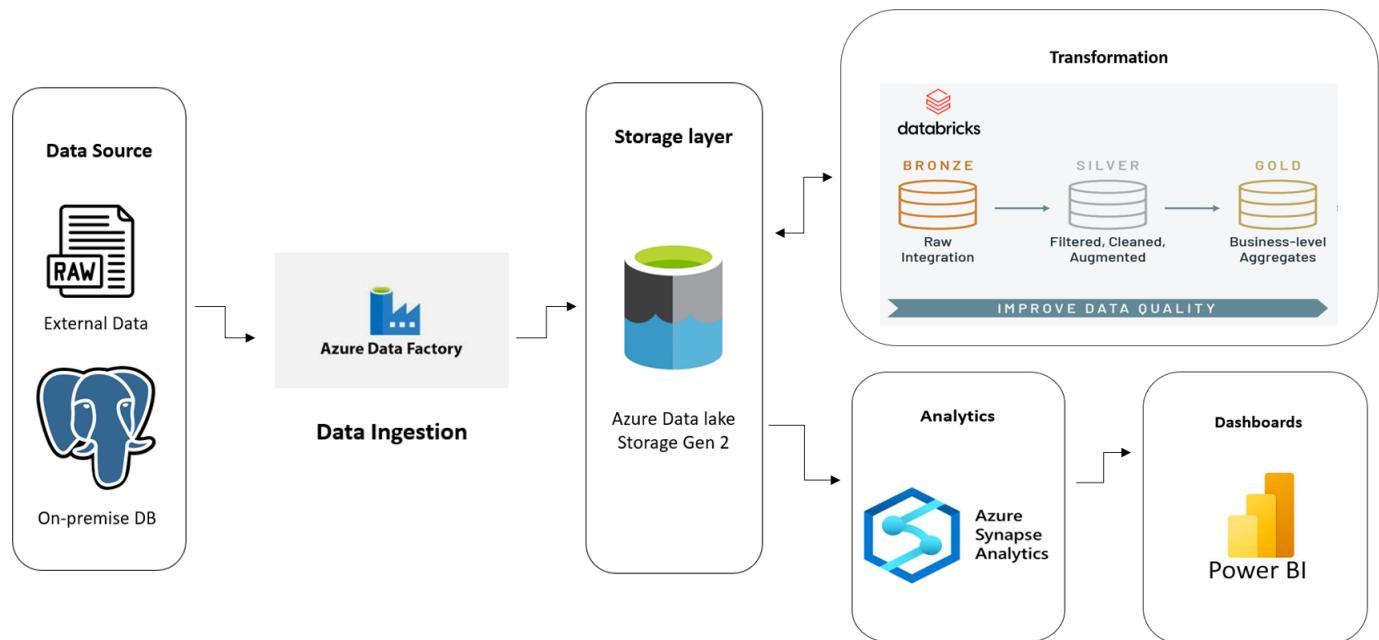
## **Title: Amazon Books Reviews**

Prepared By: Rooban M

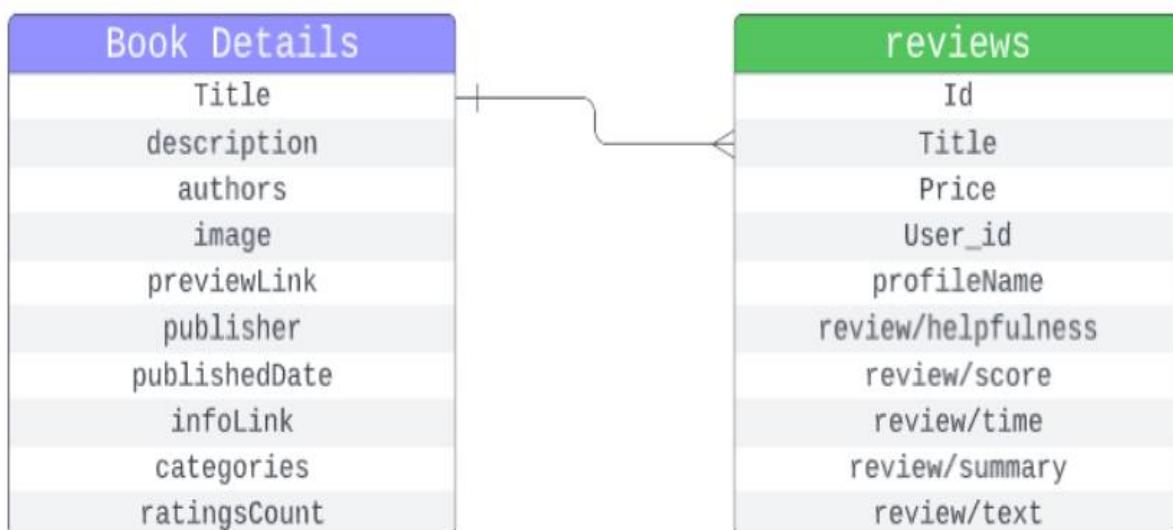
### INDEX:

S.NO	CONTENT
1.	Design and Algorithm
2.	Table schema
3.	Raw data details
4.	Data loading and preparation
5.	Creation of Azure storage account
6.	Data Ingestion
7.	Transformations
8.	Medallion Architecture
9.	Analytics and Visualization

## DESIGN AND ALGORITHM:



## SCHEMA OF TABLES:



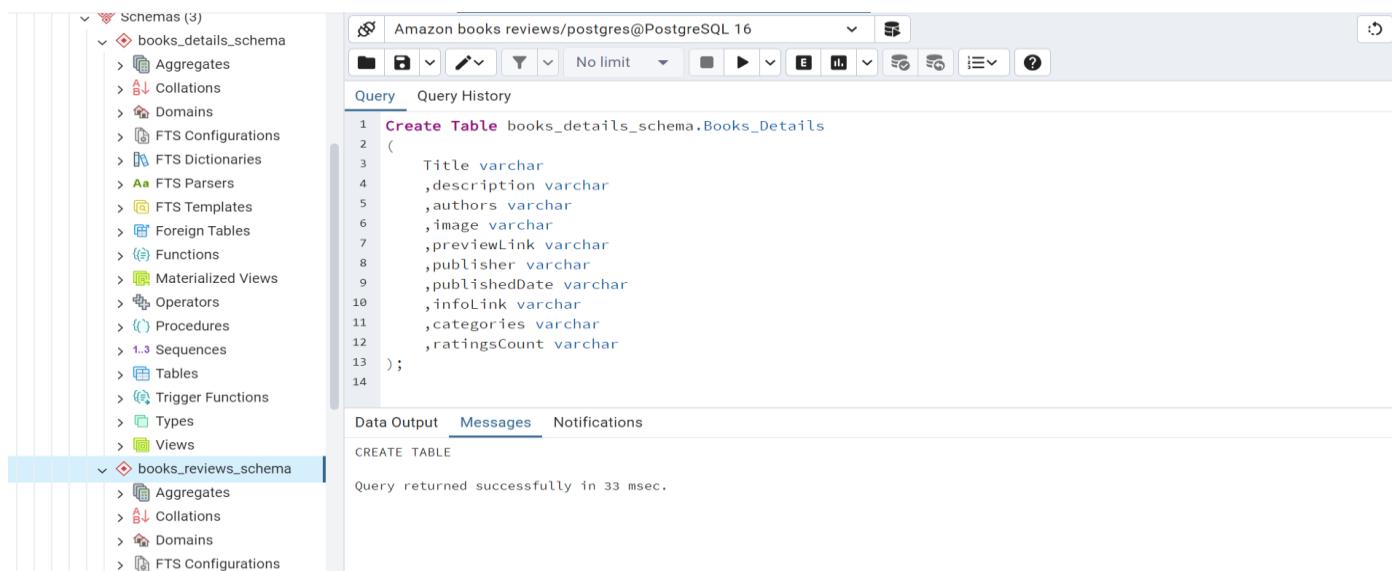
## RAW DATA DETAILS:

Data source: Kaggle ([Link](#))

Table Name	File Size (Mb)	File Format	No.of.Columns	No.of.Rows
Books_Details	181	CSV	10	212404
Books_reviews	2860	CSV	10	3000000

## STEP 1:DATA LOADING AND PREPARATION

### 1. Creating Tables in PostgreSQL and loading all the .csv files



The screenshot shows the pgAdmin 4 interface for managing a PostgreSQL database. On the left, the 'Schemas' tree view displays two schemas: 'books\_details\_schema' and 'books\_reviews\_schema'. The 'books\_details\_schema' node is expanded, showing various objects like Aggregates, Collations, Domains, FTS Configurations, etc. The 'books\_reviews\_schema' node is also expanded. In the center, a query editor window titled 'Amazon books reviews/postgres@PostgreSQL 16' contains the following SQL code:

```
1 Create Table books_details_schema.Books_Details
2 (
3     Title varchar
4     ,description varchar
5     ,authors varchar
6     ,image varchar
7     ,previewLink varchar
8     ,publisher varchar
9     ,publishedDate varchar
10    ,infoLink varchar
11    ,categories varchar
12    ,ratingsCount varchar
13 );
14
```

Below the query editor, the 'Messages' tab is selected, showing the message: 'CREATE TABLE'. At the bottom, it says 'Query returned successfully in 33 msec.'

The screenshot shows the pgAdmin interface with the following details:

- Schemas:** The left sidebar shows two schemas: `books_details_schema` and `books_ratings_schema`. The `books_ratings_schema` is currently selected.
- Query Editor:** The main window displays a SQL query for creating a table:
 

```

1 Create Table books_ratings_schema.Books_Details
2 (
3     id varchar
4     ,Title varchar
5     ,price varchar
6     ,user_id varchar
7     ,profile_name varchar
8     ,review_helpfulness varchar
9     ,review_score varchar
10    ,review_time varchar
11    ,review_summary varchar
12    ,review_text varchar
13 );
14
      
```
- Messages:** Below the query editor, the "Messages" tab shows the message: "Query returned successfully in 56 msec." followed by a green success icon.

### Import/Export data - table 'books\_details'

General Options Columns

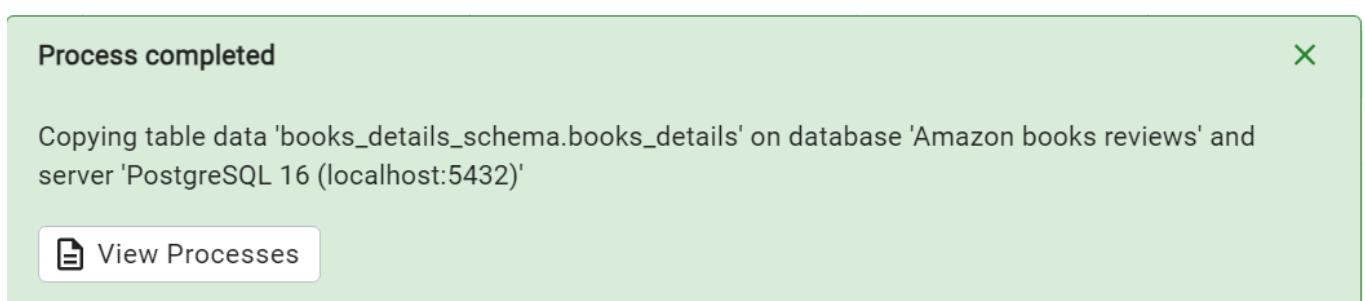
Import/Export ✓ Import Export

Filename: E:\Documents\Rooban\Kaggle dataset\Books\_details.csv ...

Format: text ...

Encoding: Select an item... ...

X Close
↻ Reset
✓ OK



Import/Export data - table 'books\_ratings'

General Options Columns

Import/Export ✓ Import Export

Filename: E:\Documents\Rooban\Kaggle dataset\Books\_ratings.csv Folder

Format: text

Encoding: Select an item... Down arrow

i ? X Close ↻ Reset ✓ OK

		PID	Type	Server	Object	Start Time	Status	T
<input type="checkbox"/>	<span style="color: red;">X</span> <span style="color: green;">D</span>	12512	Import Data	PostgreSQL 16 (localhost:54...	Amazon books reviews/boo...	5/13/2024, 6:59:26 ...	Finished	C
<input type="checkbox"/>	<span style="color: red;">X</span> <span style="color: green;">D</span>	12296	Import Data	PostgreSQL 16 (localhost:54...	Amazon books reviews/boo...	5/13/2024, 6:58:07 ...	Failed	C
<input type="checkbox"/>	<span style="color: red;">X</span> <span style="color: green;">D</span>	14296	Import Data	PostgreSQL 16 (localhost:54...	Amazon books reviews/boo...	5/13/2024, 6:54:17 ...	Finished	C
<input type="checkbox"/>	<span style="color: red;">X</span> <span style="color: green;">D</span>	13776	Import Data	PostgreSQL 16 (localhost:54...	Amazon books reviews/boo...	5/13/2024, 6:49:21 ...	Failed	C

Process completed X

Copying table data 'books\_ratings\_schema.books\_ratings' on database 'Amazon books reviews' and server 'PostgreSQL 16 (localhost:5432)'

D View Processes

Process started X

Copying table data 'books\_ratings\_schema.books\_ratings' on database 'Amazon books reviews' and server 'PostgreSQL 16 (localhost:5432)'

D View Processes

Screenshot of pgAdmin 4 showing the results of a query against the `booksdetails` database.

**Query:**

```
1  SELECT * FROM booksdetails.books_details
```

**Data Output:**

	title	description
1	Its Only Art If Its Well Hung!	[null]
2	Dr. Seuss: American Icon	Philip Nel takes a fascinating look into the key aspects o
3	Wonderful Worship in Smaller Churches	This resource includes twelve principles in understandin
4	Whispers of the Wicked Saints	Julia Thomas finds her life spinning out of control after t
5	Nation Dance: Religion Identity and Cultural Difference in the Caribbean	[null]
6	The Church of Christ: A Biblical Ecclesiology for Today	In The Church of Christ: A Biblical Ecclesiology for Today
7	The Overbury affair (Avon)	[null]
8	A Walk in the Woods: a Play in Two Acts	[null]
9	Saint Hyacinth of Poland	The story for children 10 and up of St. Hyacinth the Dom
10	Rising Sons and Daughters: Life Among Japan's New Young	Wardell recalls his experience as a foreign student in Jap

Screenshot of pgAdmin 4 showing the results of a query against the `booksreviews` database.

**Query:**

```
1  SELECT * FROM booksreviews.books_reviews
```

**Data Output:**

	id	title	price	user_id	filename	profile_help
1	1882931173	Its Only Art If Its Well Hung!	[null]	AVCGYZL8FQQTD	Jim of Oz jim-of-oz	07-Jul
2	826414346	Dr. Seuss: American Icon	[null]	A30TK6U7DNS82R	Kevin Killian	10-Oct
3	826414346	Dr. Seuss: American Icon	[null]	A3UH4UZ4RSV082	John Granger	10-Nov
4	826414346	Dr. Seuss: American Icon	[null]	A2MVUWT453QH61	Roy E. Perry amateur philosopher	07-Jul
5	826414346	Dr. Seuss: American Icon	[null]	A22X4XUPKF66MR	D. H. Richards ninthwavestore	03-Mar
6	826414346	Dr. Seuss: American Icon	[null]	A2F6NONFUDB6UK	Malvin	02-Feb
7	826414346	Dr. Seuss: American Icon	[null]	A14OJS0VWMOS...	Midwest Book Review	03-Apr
8	826414346	Dr. Seuss: American Icon	[null]	A2RSXTDZDUSH4	J. Squire	0/0
9	826414346	Dr. Seuss: American Icon	[null]	A25MD512GUIW6W	J. P. HIGBED big fellow	0/0
10	826414346	Dr. Seuss: American Icon	[null]	A3VA4XFS5WNJ03	Donald Burnside	03-May

## STEP 2: CREATION OF AZURE STORAGE ACCOUNT

### 1. Creating azure Resource Group

The screenshot shows the Azure Resource Groups blade. The left sidebar lists various navigation options: Overview, Activity log, Access control (IAM), Tags, Resource visualizer, Events, Settings, Cost Management, Monitoring, Automation, and Help. The main area is titled 'Amazon\_books\_review' and shows the 'Essentials' view. It includes a search bar, filter buttons for 'Type equals all' and 'Location equals all', and a table listing one record: 'amazonbooksreview' (Storage account) located in South India.

### 2. Created new storage account in Azure cloud to store raw and transformed data.

The screenshot shows the Microsoft Azure deployment blade for a deployment named 'amazonbooksreview\_1714457108374'. The main message says 'Your deployment is complete'. Deployment details include: Deployment name: amazonbooksreview\_1714..., Subscription: Free Trial, Resource group: Amazon\_books\_review. The deployment started at 4/30/2024, 11:35:29 AM with Correlation ID: 54dea5fa-7c7a-4e32-a169-152dbcaa1. On the right side, there are promotional cards for Cost Management, Microsoft Defender for Cloud, and Free Microsoft tutorials.

### 3. creating a container inside Azure storage account and added directories for storing raw and transformed data.

Home > Resource groups > Amazon\_books\_review > amazonbooksreview | Containers

amazonbooksreview | Containers

Storage account

Search Container Change access level Restore containers Refresh Delete Give feedback

Diagnose and solve problems Access Control (IAM) Data migration Events Storage browser Data storage

Containers File shares Queues Tables Security + networking Data management Settings

Name Last modified Anonymous access level Lease state

\$logs	4/30/2024, 11:36:02 AM	Private	Available
amzon-books-review	4/30/2024, 11:45:03 AM	Private	Available

Show deleted containers

Search containers by prefix

Container

File shares

Queues

Tables

Security + networking

Data management

Settings

Home > Resource groups > Amazon\_books\_review > amazonbooksreview | Containers >

amzon-books-review | Container

Search Upload Add Directory Refresh Rename Delete Change tier Acquire lease Break lease Give feedback

Overview

Diagnose and solve problems Access Control (IAM)

Settings

Authentication method: Access key (Switch to Microsoft Entra user account)  
Location: amzon-books-review

Search blobs by prefix (case-sensitive)

Successfully added directory  
Successfully added directory 'Transformed Data'.

Name Modified Access tier Archive status Blob type Size

Raw Data				
Transformed Data				

## STEP 3: DATA INGESTION

### a. Data Ingestion Using Azure Data Factory

#### 1. Deploying Azure Data Factory

Home >

Microsoft.DataFactory-20240430114859 | Overview

Deployment

Search Delete Cancel Redeploy Download Refresh

Deployment succeeded  
Deployment 'Microsoft.DataFactory-20240430114859' to resource group 'Amazon\_books\_review' was successful.

Pin to dashboard Go to resource group

Overview

Your deployment is complete

Deployment name : Microsoft.DataFactory-2024043... Start time : 4/30/2024, 12:00:29 PM  
Subscription : Free Trial Correlation ID : 21e77914-5aba-4b31-a908-010...  
Resource group : Amazon\_books\_review

Deployment details

Next steps

Go to resource

Give feedback

Tell us about your experience with deployment

Cost management  
Get notified to stay within your budget and prevent unexpected charges on your bill.  
Set up cost alerts >

Microsoft Defender for Cloud  
Secure your apps and infrastructure  
Go to Microsoft Defender for Cloud >

Free Microsoft tutorials  
Start learning today >

## 2. Connecting On-premise PostgreSQL Database to Data factory using Microsoft Integration runtime.

The screenshot shows the Azure Data Factory pipeline configuration interface. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (1), 'Change Data Capture (preview)', 'Datasets', 'Data flows', and 'Power Query'. The main area shows a pipeline named 'pipeline1' with an activity 'Copy data' selected under 'Move and transform'. To the right, a detailed configuration pane for a 'PostgreSQL (Legacy)' connection is displayed:

- General** tab selected.
- Server name \***: localhost
- Port**: 5432
- Database name \***: amazonbooksreview
- User name \***: postgres
- Password**: (redacted)
- Azure Key Vault**: (disabled)
- Encryption method**: No encryption

At the bottom right of the configuration pane, there are 'Create' and 'Cancel' buttons, and a status message: 'Connection successful' with a checkmark icon and a 'Test connection' link.

The screenshot shows the Azure Data Factory pipeline configuration interface. The left sidebar shows 'Factory Resources' with 'Pipelines' (1) selected. The main area shows a pipeline named 'pipeline1' with an activity 'Copy data' selected under 'Move and transform'. A 'Set properties' dialog is open on the right:

**Set properties**

- Name**: PostgreSqlTable1
- Linked service \***: PostgreSQL1
- Connect via integration runtime \***: integrationRuntime1 (selected)
- Table name**: booksdetails.books\_details
- Enter manually**: (unchecked)

At the bottom right of the dialog, there are 'OK', 'Back', and 'Cancel' buttons.

## 3. Creating Data Ingestion Pipeline and loading data into azure data lake.

The screenshot shows the Azure Data Factory pipeline editor. A pipeline named "pipeline1" is selected. The pipeline contains two "Copy data" activities. The first activity copies data from "Books\_details" to "Books\_Reviews". The second activity copies data from "Books\_Reviews" back to "Books\_details". Both activities are marked as "Succeeded". The "Output" tab is selected, showing the run history with two entries:

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime	User properties	Activity run ID
Books_Reviews	Succeeded	Copy data	5/19/2024, 9:34:23 PM	14s	integrationRuntime1		5d01ee24
Books_details	Succeeded	Copy data	5/19/2024, 9:34:12 PM	11s	integrationRuntime1		30408e1a

The screenshot shows the Azure Storage Explorer interface. A container named "amazon-books-review" is selected. The "Overview" tab is active. The container has two blobs:

Name	Modified	Access tier	Archive status	Blob type	Size
books_reviews.csv	5/19/2024, 9:34:36 PM	Hot (Inferred)		Block blob	7.07
booksdetails.csv	5/19/2024, 9:34:22 PM	Hot (Inferred)		Block blob	13.6

## b. uploading Data files directly to Azure Data Lake from local.

The screenshot shows the Azure Storage Explorer interface. A container named "amazon-books-review" is selected. The "Overview" tab is active. The container has two folders:

- Raw Data
- Transformed Data

A modal window titled "Upload blob" is open on the right side. It contains a cloud icon and a text area with the placeholder "Drag and drop files here or Browse for files". Below the text area are two checkboxes: "Overwrite if files already exist" and "Advanced". At the bottom right of the modal is a large "Upload" button.

## STEP 4. TRANSFORMATIONS

### 1. Creating Azure Databricks workplace

The screenshot shows the Azure Databricks service page. At the top, there's a Microsoft Azure header with a search bar and user information. Below it, the 'Azure-Databricks' service page is displayed. On the left, a sidebar menu includes 'Overview' (which is selected), 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Settings', 'Automation', and 'Help'. In the center, there's a large red 'Databricks' logo and two prominent buttons: 'Launch Workspace' and 'Upgrade to Premium'. Below these are several quick links: 'Documentation', 'Getting Started', 'Import Data from File', 'Import Data from Azure Storage', 'Notebook', 'Admin Guide', and 'Link Azure ML workspace'. A search bar is also present at the top of the main content area.

### 2. Connecting Azure Blob storage with Azure Databricks.

The screenshot shows an Azure Databricks workspace. The left sidebar lists various notebooks and workspace components like Recents, Catalog, Workflows, Compute, SQL, and Data Engineering. The main area displays a Python notebook titled 'Amazon-book-review'. The notebook interface includes a code editor with a run history, a results viewer, and a sidebar with various icons. The code cell contains the following Python code:

```
%fs  
ls "/mnt/amazonbooksreviews"
```

The results section shows a table with the following data:

A <sub>b</sub> c path	A <sub>b</sub> c name	A <sub>b</sub> <sub>3</sub> size	A <sub>b</sub> <sub>3</sub> modificationTime
1 dbfs:/mnt/amazonbooksreviews/Raw Data/	Raw Data/	0	1714457774000
2 dbfs:/mnt/amazonbooksreviews/Transformed Data/	Transformed Data/	0	1714457788000

At the bottom, it says '2 rows | 8.63 seconds runtime' and 'Refreshed 6 minutes ago'.

### 3. Reading Data from Books\_Details Table.

```
▶ ▾ ✓ 07:41 PM (15s) 5 Python ⚡ ⌂ ⌃ ⌄ ⌅
#reading books details file from Azure Storage Account
Book_details_raw = spark.read.format("csv").option("header" ,True).option("inferSchema",True).load("/mnt/
amazonbooksreviews/Raw Data/books_data.csv")
Book_details_raw.count()

▶ (4) Spark Jobs
▶ Book_details_raw: pyspark.sql.dataframe.DataFrame = [Title: string, description: string ... 8 more fields]
212404
```

```
▶ ✓ 07:41 PM (<1s) 6
Book_details_raw.printSchema()

root
 |-- Title: string (nullable = true)
 |-- description: string (nullable = true)
 |-- authors: string (nullable = true)
 |-- image: string (nullable = true)
 |-- previewLink: string (nullable = true)
 |-- publisher: string (nullable = true)
 |-- publishedDate: string (nullable = true)
 |-- infoLink: string (nullable = true)
 |-- categories: string (nullable = true)
 |-- ratingsCount: string (nullable = true)
```

### 4. Removing unwanted characters from Authors and Categories columns and converting these columns into Arrays since they have nested values.

#### Original data: (Authors)

	A <sup>B</sup> C authors
30	... ['John Adrian']
31	... to the contemporary
32	... ['Rupert Fike']
33	... ['E. B. Harvey']
34	... ['Stefan Draminski']
35	... ['Raymond Charles Barker']
36	... null
37	... ['Lorenz Books', 'Margaret Malone', 'Paul Jackson']
38	... ['Kristin Ramsdell']
39	... capitals
40	... however unpleasant they may be
41	... ['Karen Cummings']
42	... ['Marie Weil', 'Michael S. Reisch', 'Mary L. Ohmer']
43	... ['May McGoldrick']

▶ ✓ 07:41 PM (<1s) 8

```
# we will remove unwanted characters like ('[',''],"'") in Authors Column
Book_details_cleaned = Book_details_raw.withColumn('authors', regexp_replace(col('authors'), "[\\[\\]]", ''))
```

▶ Book\_details\_cleaned: pyspark.sql.dataframe.DataFrame = [Title: string, description: string ... 8 more fields]

+ Code + Text

▶ ✓ 07:41 PM (<1s) 9

```
# Since the "Authors" column is having multiple authors for a single book title we will convert it into a array and explode it as separate rows.
Book_details_cleaned= Book_details_cleaned.withColumn("authors", split(Book_details_cleaned["authors"], ","))
```

▶ Book\_details\_cleaned: pyspark.sql.dataframe.DataFrame = [Title: string, description: string ... 8 more fields]

## Original data: (categories)

Table + New result table: ON ▾ Search

	categories
24	superstition
25	us scrutiny of the commingling of ... the stories are an exposé of the comforts and discomforts of that cohabitation."
26	[Religion]
27	[Psychology]
28	null
29	['Sports & Recreation']
30	null
31	round out the presidential collection. Lively narratives accompany each piece
32	[Biography & Autobiography]
33	['Bible stories, English']
34	[History]
35	[New Thought]
36	null
37	['Crafts & Hobbies']

▶ ✓ 07:41 PM (<1s) 10

```
Book_details_cleaned = Book_details_cleaned.withColumn('categories', regexp_replace(col('categories'), "[\\[\\]]", ''))
```

▶ Book\_details\_cleaned: pyspark.sql.dataframe.DataFrame = [Title: string, description: string ... 8 more fields]

+ Code + Text

▶ ✓ 07:41 PM (<1s) 11

```
# Since the "categories" column is having multiple category for a single book title we will convert it into a array and explode it as separate rows.
Book_details_cleaned = Book_details_cleaned.withColumn("categories", split(Book_details_cleaned["categories"], ","))
```

▶ Book\_details\_cleaned: pyspark.sql.dataframe.DataFrame = [Title: string, description: string ... 8 more fields]

## Transformed Data:

```
▶ ▾ ✓ 07:41 PM (<1s) 12
Book_details_cleaned.printSchema()

root
 |-- Title: string (nullable = true)
 |-- description: string (nullable = true)
 |-- authors: array (nullable = true)
 |   |-- element: string (containsNull = false)
 |-- image: string (nullable = true)
 |-- previewLink: string (nullable = true)
 |-- publisher: string (nullable = true)
 |-- publishedDate: string (nullable = true)
 |-- infoLink: string (nullable = true)
 |-- categories: array (nullable = true)
 |   |-- element: string (containsNull = false)
 |-- ratingsCount: string (nullable = true)
```

```
authors
└── array
    0: "Julie Strain"
    > [" like that of Lewis Carroll and Edward Lear"]
    > ["David R. Ray"]
    > ["Veronica Haddon"]
    > ["Edward Long"]
    > ["Everett Ferguson"]
    > ["Miriam Allen De Ford"]
    > ["Lee Blessing"]
    > ["Mary Fabyan Windeatt"]
    > ["Steven Wardell"]
    └── array
        0: "Camillia Fawzi El-Solh"
        1: " Judy Mabro"
    > ["Armando Salda A-Mora"]
```

```
categories
└── null
└── null
    └── array
        0: "Biography & Autobiography"
        > ["Social Science"]
        > ["Religion"]
        > ["Reference"]
        > ["Juvenile Nonfiction"]
        └── array
            0: "Allen Gersho"
            1: " Robert M. Gray"
        > ["History"]
        > ["New Zealand fiction"]
```

## 5. Removing Bad Data from RatingsCount column.

### Original data:

A	B	C	ratingsCount
			null
			> and the supernatural. It provides you with a balanced and unbiased account of everything from Japanese folklore and Indian witchcraf...
			['Sonia Rivera-Valdes']
			null
			exploring the president's background and biography as well as the work's artistic and historical significance. Taken together
1.0			1.0
			null
1.0			1.0
			null
1.0			1.0

### Transformation:

```
▶ ▾ ✓ 07:41 PM (2s) 14 Python 🗑 ⚙ ☰ ⋮
# Remove bad data from "ratingsCount" Column
Book_details_cleaned_1 = Book_details_cleaned.withColumn("Ratings_Count", Book_details_cleaned.ratingsCount.cast
(FloatingType()))
Book_details_cleaned_1.count()

▶ (2) Spark Jobs
▶ Book_details_cleaned_1: pyspark.sql.dataframe.DataFrame = [Title: string, description: string ... 9 more fields]
212404
```

```
▶ ▾ ✓ 07:41 PM (1s) 15 Python 🗑 ⚙ ☰ ⋮
Book_details_cleaned_1.select("ratingsCount", "Ratings_Count").show()

▶ (1) Spark Jobs
```

### Transformed Data:

```
+-----+-----+
| ratingsCount | Ratings_Count |
+-----+-----+
| NULL | NULL |
| A&C Black | NULL |
| NULL | NULL |
| NULL | NULL |
| NULL | NULL |
| 5.0 | 5.0 |
| NULL | NULL |
| 3.0 | 3.0 |
| NULL | NULL |
| http://books.goog... | NULL |
| NULL | NULL |
| NULL | NULL |
| NULL | NULL |
| 2016-01-05 | NULL |
```

## 6. Cleaning PublishedDate column.

Just now (1s) 16 Python

```
# Getting the publication year from "publishedDate" Column
# Spark 3.0 does not allow Date parsing of 'yyyy-mm-dd' and 'yyyy' in same column so we change the configuration here

spark.conf.set("spark.sql.legacy.timeParserPolicy", "LEGACY")

Book_details_cleaned_2 = Book_details_cleaned_1.withColumn("Year_of_publish", year(to_date("publishedDate", "yyyy")))
Book_details_cleaned_2.select("publishedDate", "Year_of_publish").show()
```

▶ (1) Spark Jobs

▶ Book\_details\_cleaned\_2: pyspark.sql.dataframe.DataFrame = [Title: string, description: string ... 10 more fields]

▶ (1) Spark Jobs

▶ Book\_details\_cleaned\_2: pyspark.sql.dataframe.DataFrame = [Title: string, description: string ... 10 more fields]

publishedDate	Year_of_publish
1996	1996
['Philip Nel']}	NULL
2000	2000
2005-02	2005
2003-03-01	2003
1996	1996
1960	1960
1988	1988
2009-01-01	2009
1995	1995
1994-02-17	1994
2005-07	2005
2018-11-06	2018
images	NULL
2018-02-27	2018
2009	2009
2010-01-28	2010
http://books.goog...	NULL

## 7. Cleaning all the Link based columns so that it contains only valid HTTP links.

10:37 AM (3s) 19 Python

```
# Dropping rows from Dataframe "Book_details_cleaned_2" which has Bad Data in Column "image"

Book_details_cleaned_3 = Book_details_cleaned_2.filter((col("image").startswith("http")) | (col("image").isNull()))
Book_details_cleaned_3.count()
```

▶ (2) Spark Jobs

▶ Book\_details\_cleaned\_3: pyspark.sql.dataframe.DataFrame = [Title: string, description: string ... 10 more fields]

191174

10:37 AM (2s) 20 Python

```
# Dropping rows from Dataframe "Book_details_cleaned_2" which has Bad Data in Column "previewLink"

Book_details_cleaned_3 = Book_details_cleaned_3.filter((col("previewLink").startswith("http")) | (col("previewLink").isNull()))
Book_details_cleaned_3.count()
```

▶ (2) Spark Jobs

▶ Book\_details\_cleaned\_3: pyspark.sql.dataframe.DataFrame = [Title: string, description: string ... 10 more fields]

191159

▶ ✓ 10:37 AM (2s) 21

```
# Dropping rows from Dataframe "Book_details_cleaned_2" which has Bad Data in Column "infoLink"

Book_details_cleaned_3 = Book_details_cleaned_2.filter((col("infoLink").startswith("http")) | (col("infoLink").isNull()))
Book_details_cleaned_3.count()

▶ (2) Spark Jobs
▶ Book_details_cleaned_3: pyspark.sql.dataframe.DataFrame = [Title: string, description: string ... 10 more fields]
190900
```

## Original Data: has invalid links

▶ **B C** infoLink

- ▶ <http://books.google.nl/books?id=399SPgAACAAJ&dq=Nation+Dance:+Religion,+Identity+and+Cultural+Difference+in+the+Caribbean>
- ▶ [http://books.google.nl/books?id=kVqRaiPlx88C&dq=The+Church+of+Christ:+A+Biblical+Ecclesiology+for+Today&hl=&source=gbs\\_api](http://books.google.nl/books?id=kVqRaiPlx88C&dq=The+Church+of+Christ:+A+Biblical+Ecclesiology+for+Today&hl=&source=gbs_api)
- ▶ [http://books.google.nl/books?id=mHLTngEACAAJ&dq=The+Overbury+affair+\(Avon\)&hl=&source=gbs\\_api](http://books.google.nl/books?id=mHLTngEACAAJ&dq=The+Overbury+affair+(Avon)&hl=&source=gbs_api)
- ▶ [http://books.google.nl/books?id=6HDOWAEACAAJ&dq=A+Walk+in+the+Woods:+a+Play+in+Two+Acts&hl=&source=gbs\\_api](http://books.google.nl/books?id=6HDOWAEACAAJ&dq=A+Walk+in+the+Woods:+a+Play+in+Two+Acts&hl=&source=gbs_api)
- ▶ [http://books.google.nl/books?id=lmLqAAAACAAJ&dq=Saint+Hyacinth+of+Poland&hl=&source=gbs\\_api](http://books.google.nl/books?id=lmLqAAAACAAJ&dq=Saint+Hyacinth+of+Poland&hl=&source=gbs_api)
- ▶ **▼ http://books.google.nl/books?**
  - id=rbLZugEACAAJ&dq=Rising+Sons+and+Daughters:+Life+Among+Japan%27s+New+Young&hl=&source=gbs\_api
- ▶ [http://books.google.nl/books?id=o7izAAAAIAAJ&dq=Muslim+Women%27s+Choices:+Religious+Belief+and+Social+Reality+\(Cross+...](http://books.google.nl/books?id=o7izAAAAIAAJ&dq=Muslim+Women%27s+Choices:+Religious+Belief+and+Social+Reality+(Cross+...)
- ▶ [http://books.google.nl/books?id=iTueuAAACAAJ&dq=Dramatica+for+Screenwriters&hl=&source=gbs\\_api](http://books.google.nl/books?id=iTueuAAACAAJ&dq=Dramatica+for+Screenwriters&hl=&source=gbs_api)
- ▶ [http://books.google.nl/books?id=tX1IswEACAAJ&dq=Mensa+Number+Puzzles+\(Mensa+Word+Games+for+Kids\)&hl=&source=gbs\\_api](http://books.google.nl/books?id=tX1IswEACAAJ&dq=Mensa+Number+Puzzles+(Mensa+Word+Games+for+Kids)&hl=&source=gbs_api)
- ▶ **▼ and video signals (excluding other data types such as financial data or general purpose computer data). The emphasis is on the conversion of analog waveforms into efficient digital representations and on the compression of digital information into the fewest possible bits. Both operations should yield the highest possible reconstruction fidelity subject to constraints on the bit rate and implementation complexity."**
- ▶ [https://play.google.com/store/books/details?id=EzxODwAAQBAJ&source=gbs\\_api](https://play.google.com/store/books/details?id=EzxODwAAQBAJ&source=gbs_api)
- ▶ [http://books.google.nl/books?id=15NOAQAAACAAJ&dq=Gold+and+greenstone&hl=&source=gbs\\_api](http://books.google.nl/books?id=15NOAQAAACAAJ&dq=Gold+and+greenstone&hl=&source=gbs_api)

## Cleaned data:

▶ **B C** image

- ▶ [http://books.google.com/books/content?id=DykPAAAACAAJ&printsec=frontcover&img=1&zoom=1&source=gbs\\_api](http://books.google.com/books/content?id=DykPAAAACAAJ&printsec=frontcover&img=1&zoom=1&source=gbs_api)
- ▶ [http://books.google.com/books/content?id=2tsDAAAACAAJ&printsec=frontcover&img=1&zoom=1&source=gbs\\_api](http://books.google.com/books/content?id=2tsDAAAACAAJ&printsec=frontcover&img=1&zoom=1&source=gbs_api)
- ▶ [http://books.google.com/books/content?id=aRSIgJlq6JwC&printsec=frontcover&img=1&zoom=1&source=gbs\\_api](http://books.google.com/books/content?id=aRSIgJlq6JwC&printsec=frontcover&img=1&zoom=1&source=gbs_api)
- ▶ **null**
- ▶ [http://books.google.com/books/content?id=kVqRaiPlx88C&printsec=frontcover&img=1&zoom=1&edge=curl&source=gbs\\_api](http://books.google.com/books/content?id=kVqRaiPlx88C&printsec=frontcover&img=1&zoom=1&edge=curl&source=gbs_api)
- ▶ **null**
- ▶ **null**
- ▶ [http://books.google.com/books/content?id=lmLqAAAACAAJ&printsec=frontcover&img=1&zoom=1&source=gbs\\_api](http://books.google.com/books/content?id=lmLqAAAACAAJ&printsec=frontcover&img=1&zoom=1&source=gbs_api)
- ▶ **null**
- ▶ [http://books.google.com/books/content?id=o7izAAAAIAAJ&printsec=frontcover&img=1&zoom=1&source=gbs\\_api](http://books.google.com/books/content?id=o7izAAAAIAAJ&printsec=frontcover&img=1&zoom=1&source=gbs_api)
- ▶ [http://books.google.com/books/content?id=iTueuAAACAAJ&printsec=frontcover&img=1&zoom=1&source=gbs\\_api](http://books.google.com/books/content?id=iTueuAAACAAJ&printsec=frontcover&img=1&zoom=1&source=gbs_api)
- ▶ [http://books.google.com/books/content?id=tX1IswEACAAJ&printsec=frontcover&img=1&zoom=1&source=gbs\\_api](http://books.google.com/books/content?id=tX1IswEACAAJ&printsec=frontcover&img=1&zoom=1&source=gbs_api)
- ▶ [http://books.google.com/books/content?id=EzxODwAAQBAJ&printsec=frontcover&img=1&zoom=1&edge=curl&source=gbs\\_api](http://books.google.com/books/content?id=EzxODwAAQBAJ&printsec=frontcover&img=1&zoom=1&edge=curl&source=gbs_api)
- ▶ **null**

## 8. Exploding the Array Columns.

```
▶ ⏴ ✓ 10:37 AM (<1s) 23 Python 🗑 ⚡ ⌂ ⋮
# Exploding "authors" and "categories" Columns

Book_details_cleaned_final = Book_details_cleaned_3.select("Title","description",explode("authors").alias("authors"),
"image","previewLink","publisher","publishedDate","infoLink",explode("categories").alias("categories"),"Ratings_Count",
"Year_of_publish")

▶ 📄 Book_details_cleaned_final: pyspark.sql.dataframe.DataFrame = [Title: string, description: string ... 9 more fields]

+-----+-----+-----+
|       Title|     authors|    categories|
+-----+-----+-----+
|Wonderful Worship...|   David R. Ray|      Religion|
|A husband for Kutani|   Frank Owen|      History|
|  Paolo Di Canio Hb| Donald Laming| Psychology|
|Muslim Women's Ch...|   Judy Mabro|      Religion|
|Its Only Art If I...|   Julie Strain| Comics & Graphic ...|
|Homer or Moses?: ...| Arthur J. Droke|      Religion|
|The Oxford Handbo...|   Robert Kane| Philosophy|
|Rising Sons and D...| Steven Wardell| Social Science|
| Gold and greenstone|   Barry Crump| New Zealand fiction|
|Eyewitness Travel...| Matthew Willis|      Europe|
|Eyewitness Travel...| Jonathan Bousfield|      Europe|
|Saint Hyacinth of...| Mary Fabyan Windeatt| Biography & Autobiography|
|"The Ultimate Gui...| Fiona Cownie|      Law|
|The Church of Chr...| Everett Ferguson|      Religion|
|Overcoming Hypert...| Kenneth H. Cooper| Health & Fitness|
|Dramatica for Scr...| Armando Salda A-Mora| Reference|
|Mensa Number Puzz...| Evelyn B. Christe...| Juvenile Nonfiction|
|Eyewitness Travel...| Dorling Kindersle...|      Europe|
```

```
▶ ⏴ ✓ 10:37 AM (<1s) 25 Python 🗑 ⚡ ⌂ ⋮
Book_details_cleaned_final.printSchema()

root
 |-- Title: string (nullable = true)
 |-- description: string (nullable = true)
 |-- authors: string (nullable = false)
 |-- image: string (nullable = true)
 |-- previewLink: string (nullable = true)
 |-- publisher: string (nullable = true)
 |-- publishedDate: string (nullable = true)
 |-- infoLink: string (nullable = true)
 |-- categories: string (nullable = false)
 |-- Ratings_Count: float (nullable = true)
 |-- Year_of_publish: integer (nullable = true)
```

## 9. Null Handling and cleaning all the Columns data.

▶ ✓ 10:37 AM (<1s) 26

```
# Null Handling for String Columns

replacement_values = {"Title": "N.A", "description": "N.A", "authors": "N.A", "image": "N.A", "previewLink": "N.A", "publisher": "N.A", "infoLink": "N.A", "categories": "N.A", "publishedDate": "N.A", "Ratings_Count": 0 }

Book_details_cleaned_final = Book_details_cleaned_final.fillna(replacement_values)
```

▶ Book\_details\_cleaned\_final: pyspark.sql.dataframe.DataFrame = [Title: string, description: string ... 9 more fields]

▶ ✓ 10:37 AM (8s) 27 Python

```
# Dropping all further rows which has nulls and dropping Duplicate rows

Book_details_cleaned_final = Book_details_cleaned_final.dropna()
Book_details_cleaned_final = Book_details_cleaned_final.dropDuplicates()
Book_details_cleaned_final.count()

▶ (3) Spark Jobs
```

▶ Book\_details\_cleaned\_final: pyspark.sql.dataframe.DataFrame = [Title: string, description: string ... 9 more fields]

194273

▶ ✓ 10:37 AM (1s) 29

```
# cleaning the column values by removing unwanted characters for one last time.

# "authors" column
Book_details_cleaned_final = Book_details_cleaned_final.withColumn("authors", regexp_replace(col("authors"), '\\\\', '')).withColumn("authors", regexp_replace(col("authors"), "'", ''))

# "publisher" column
Book_details_cleaned_final = Book_details_cleaned_final.withColumn("publisher", regexp_replace(col("publisher"), '\\\\', '')).withColumn("publisher", regexp_replace(col("publisher"), "'", ''))

# "categories" column
Book_details_cleaned_final = Book_details_cleaned_final.withColumn("categories", regexp_replace(col("categories"), '\\\\', '')).withColumn("categories", regexp_replace(col("categories"), "'", ''))
```

▶ Book\_details\_cleaned\_final: pyspark.sql.dataframe.DataFrame = [Title: string, description: string ... 9 more fields]

## 10. Reading Books\_Reviews Table.

▶ ✓ 07:45 PM (31s) 4 Python

```
#reading Books_rating file from Azure Storage Account

Books_ratings_raw = spark.read.format("csv").option("header", True).option("inferSchema", True).load("/mnt/amazonbooksreviews/Raw Data/Books_rating.csv")
Books_ratings_raw.count()

▶ (4) Spark Jobs
```

▶ Books\_ratings\_raw: pyspark.sql.dataframe.DataFrame = [Id: string, Title: string ... 8 more fields]

3000000

Sample Data:

▶ ▾ ✓ 07:40 PM (<1s)

Books\_ratings\_raw.display()

▶ (1) Spark Jobs

Table + New result table: ON ▾

	A <sub>c</sub> Price	A <sub>c</sub> User_id	A <sub>c</sub> profileName	A <sub>c</sub> review/helpfulness	A <sub>c</sub> review/score	A <sub>c</sub> review/time	A <sub>c</sub> review/summary
8	null	A2RSSXTDZDUSH4	J. Squire	0/0	5.0	1231200000	Academia At It's Best
9	null	A25MD512GUIW6W	"J. P. HIGBED ""big fellow"""	0/0	5.0	1209859200	And to think that I read it on tl
10	null	A3VA4XFS5WNJO3	Donald Burnside	3/5	4.0	1076371200	Fascinating account of a geniu
11	19.40	AZ01OBU20TBOP	Rev. Pamela Tinnin	8/10	5.0	991440000	Outstanding Resource for Sma
12	19.40	A373VVEU6Z9M0N	Dr. Terry W. Dorsett	1/1	5.0	1291766400	Small Churches CAN Have Wo
13	19.40	AGKGDOH65VTRR4	"Cynthia L. Lajoy ""Cindy La Joy"""	1/1	5.0	1248307200	Not Just for Pastors!
14	19.40	A3OQWLU31B1Y1	Maxwell Grant	1/1	5.0	1222560000	Small church pastor? This is th
15	10.95	A3Q12RK71N74LB	Book Reader	7/11	1.0	1117065600	not good
16	10.95	A1E9M6APK30ZAU	V. Powell	1/2	4.0	1119571200	Here is my opinion
17	10.95	AUROVA5HOC66C	"LoveToRead ""Actually Read Books""	1/2	1.0	1119225600	Buyer beware
18	10.95	A1YLDZ3VHR6QPZ	Clara	2/4	5.0	1115942400	Fall on your knee's
19	10.95	ACO23CG8K8T77	Tonya	5/9	5.0	1117065600	Bravo Veronica
20	10.95	A1VK81CRRC7MLM	"missyLou ""apple"""	1/3	5.0	1130025600	Wonderful
21	10.95	A2GKUH6OBW7POH	julee gle	0/2	5.0	1118102400	Awesome !

↓ ↓ 2,761 rows | Truncated data | 0.67 seconds runtime Refreshed 39 minutes ago

## 11. Transformations

▶ ✓ 07:46 PM (<1s)

# the column "reviews/helpfulness" is not consistent so will convert into percentage for better understanding.

# splitting the column based on '/' to separate the numerator and denominator values.

```
Books_ratings_cleaned_1 = Books_ratings_raw.withColumn("Reviews", split(Books_ratings_raw["review/helpfulness"], "/")[0]) \
    .withColumn("Helpfulness", split(Books_ratings_raw["review/helpfulness"], "/")[1])
```

# casting the separate columns to FloatType to remove the string values.

```
Books_ratings_cleaned_1 = Books_ratings_cleaned_1.withColumn("Reviews", Books_ratings_cleaned_1.Reviews.cast(FloatType()))
Books_ratings_cleaned_1 = Books_ratings_cleaned_1.withColumn("Helpfulness", Books_ratings_cleaned_1.Helpfulness.cast(FloatType()))
```

# Taking the Percentage

```
Books_ratings_cleaned_1 = Books_ratings_cleaned_1.withColumn("Review_helpfulness", round(col("Reviews")/col("Helpfulness") *100,2))
```

▶ Books\_ratings\_cleaned\_1: pyspark.sql.dataframe.DataFrame = [Id: string, Title: string ... 11 more fields]

▶ ✓ 07:46 PM (<1s)

# Casting "review/score" as FloatType to remove bad data

```
Books_ratings_cleaned_1 = Books_ratings_cleaned_1.withColumn("review/score", Books_ratings_cleaned_1["review/score"].cast(FloatType()))

Books_ratings_cleaned_2 = Books_ratings_cleaned_1.dropna(subset = "review/score" )
```

▶ Books\_ratings\_cleaned\_1: pyspark.sql.dataframe.DataFrame = [Id: string, Title: string ... 11 more fields]

▶ Books\_ratings\_cleaned\_2: pyspark.sql.dataframe.DataFrame = [Id: string, Title: string ... 11 more fields]

▶ ✓ 2 minutes ago (<1s)

10

Python

# converting "review/time" column from unix timestamp to datetime

```
Books_ratings_cleaned_3 = Books_ratings_cleaned_2.withColumn("review_Time", from_unixtime(Books_ratings_cleaned_2["review/time"].cast("long")))
```

▶ Books\_ratings\_cleaned\_3: pyspark.sql.dataframe.DataFrame = [Id: string, Title: string ... 12 more fields]

## Transformed Data:

▶ ✓ 3 minutes ago (1s) 7 Python

```
Books_ratings_cleaned_1.select("review/helpfulness", "Review_helpfulness").show()
```

▶ (1) Spark Jobs

review/helpfulness	Review_helpfulness
7/7	100.0
10/10	100.0
10/11	90.91
7/7	100.0
3/3	100.0
2/2	100.0
3/4	75.0
0/0	NULL
0/0	NULL
3/5	60.0
8/10	80.0
1/1	100.0
1/1	100.0
1/1	100.0
7/11	63.64
1/2	50.0
1/2	50.0
2/4	50.0

▶ ✓ Just now (<1s) 11 Python

```
Books_ratings_cleaned_3.select("review/time", "review_Time").show()
```

▶ (1) Spark Jobs

review/time	review_Time
940636800	1999-10-23 00:00:00
1095724800	2004-09-21 00:00:00
1078790400	2004-03-09 00:00:00
1090713600	2004-07-25 00:00:00
1107993600	2005-02-10 00:00:00
1127174400	2005-09-20 00:00:00
1100131200	2004-11-11 00:00:00
1231200000	2009-01-06 00:00:00
1209859200	2008-05-04 00:00:00
1076371200	2004-02-10 00:00:00
991440000	2001-06-02 00:00:00
1291766400	2010-12-08 00:00:00
1248307200	2009-07-23 00:00:00
1222560000	2008-09-28 00:00:00
1117065600	2005-05-26 00:00:00
1119571200	2005-06-24 00:00:00
1119225600	2005-06-20 00:00:00
1115942400	2005-05-13 00:00:00

## 12. Null Handling

▶ ✓ Just now (14s) 12 Python

```
# checking how many rows of data is having null values in "Price" column
```

```
Books_ratings_cleaned_3.where(col("Price").isNull()).count()
```

▶ (2) Spark Jobs

2504309

▶ ✓ 08:00 PM (<1s) 13 Python

```
# Since more than 84% of the data in "price" Column is 'Null' it will not give any meaningful insites.
```

```
# So Dropping the column
```

```
Books_ratings_cleaned_4 = Books_ratings_cleaned_3.drop("Price")
```

▶ Books\_ratings\_cleaned\_4: pyspark.sql.DataFrame = [Id: string, Title: string ... 11 more fields]

08:00 PM (<1s) 14 Python ⚡ ⚡ ⚡ ⚡ ⚡

```
Books_ratings_cleaned_4.printSchema()

root
|-- Id: string (nullable = true)
|-- Title: string (nullable = true)
|-- User_id: string (nullable = true)
|-- profileName: string (nullable = true)
|-- review/helpfulness: string (nullable = true)
|-- review/score: float (nullable = true)
|-- review/time: string (nullable = true)
|-- review/summary: string (nullable = true)
|-- review/text: string (nullable = true)
|-- Reviews: float (nullable = true)
|-- Helpfulness: float (nullable = true)
|-- Review_helpfulness: double (nullable = true)
|-- review_Time: string (nullable = true)
```

08:11 PM (14s) 15 Python ⚡ ⚡ ⚡ ⚡ ⚡

```
# Null Handling for All Columns
mean_value = Books_ratings_cleaned_4.agg(mean("review/score")).collect()[0][0]

replacement_values = {"Title": "N.A", "User_id": "N.A", "profileName": "N.A", "review/summary": "N.A", "review/text": "N.A",
                     "Review_helpfulness": 0, "review/score": mean_value }

Books_ratings_cleaned_4 = Books_ratings_cleaned_4.fillna(replacement_values)

▶ (2) Spark Jobs
▶ Books_ratings_cleaned_4: pyspark.sql.dataframe.DataFrame = [Id: string, Title: string ... 11 more fields]
```

## 13. Writing the Transformed Data back to Azure blob storage in Parquet format.

30

```
# Write the Transformed/Cleaned data into Azure Data lake
Book_details_cleaned_final.write.option("header",True).csv("/mnt/amazonbooksreviews/Transformed Data/Books_Details_cleaned.csv")
```

Name	Modified	Access tier	Archive status	Blob type	Size
_committed_6751895544573591605	5/6/2024, 8:14:12 PM	Hot (Inferred)		Block blob	380
_started_6751895544573591605	5/6/2024, 8:14:08 PM	Hot (Inferred)		Block blob	0 B
_SUCCESS	5/6/2024, 8:14:12 PM	Hot (Inferred)		Block blob	0 B
part-00000-tid-6751895544573591605-c88f81f6-15aa...	5/6/2024, 8:14:11 PM	Hot (Inferred)		Block blob	42.7
part-00001-tid-6751895544573591605-c88f81f6-15aa...	5/6/2024, 8:14:11 PM	Hot (Inferred)		Block blob	43.2
part-00002-tid-6751895544573591605-c88f81f6-15aa...	5/6/2024, 8:14:11 PM	Hot (Inferred)		Block blob	43.2
part-00003-tid-6751895544573591605-c88f81f6-15aa...	5/6/2024, 8:14:12 PM	Hot (Inferred)		Block blob	44.2

▶ ✓ 08:13 PM (<1s) 16

```
# final Transformed Data

Books_ratings_cleaned_final = Books_ratings_cleaned_4.select("Id","Title","User_id","profileName","Review_helpfulness",col("review/score").alias("review_score"),col("review/time").alias("review_time_unix"),"review_Time",col("review/summary").alias("review_summary"),col("review/text").alias("review_text"))
```

▶ Books\_ratings\_cleaned\_final: pyspark.sql.dataframe.DataFrame = [Id: string, Title: string ... 8 more fields]

▶ ✓ 08:13 PM (34s) 17

```
# Write the Transformed/Cleaned data into Azure Data lake

Books_ratings_cleaned_final.write.option("header",True).parquet("/mnt/amazonbooksreviews/Transformed Data/Books_ratings_cleaned")
```

▶ (1) Spark Jobs

**Authentication method:** Access key ([Switch to Microsoft Entra user account](#))  
**Location:** [amazon-books-review](#) / [Transformed Data](#) / Books\_ratings\_cleaned

Search blobs by prefix (case-sensitive)

Show deleted objects

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state	...
📁 [...]							...
📄 _committed_8193914268604594276	5/8/2024, 8:14:27 PM	Hot (Inferred)		Block blob	1.98 KiB	Available	...
📄 _started_8193914268604594276	5/8/2024, 8:13:54 PM	Hot (Inferred)		Block blob	0 B	Available	...
📄 _SUCCESS	5/8/2024, 8:14:28 PM	Hot (Inferred)		Block blob	0 B	Available	...
📄 part-00000-tid-8193914268604594276-d57dfcab-5dee...	5/8/2024, 8:14:00 PM	Hot (Inferred)		Block blob	63.19 MiB	Available	...
📄 part-00001-tid-8193914268604594276-d57dfcab-5dee...	5/8/2024, 8:14:00 PM	Hot (Inferred)		Block blob	63.13 MiB	Available	...
📄 part-00002-tid-8193914268604594276-d57dfcab-5dee...	5/8/2024, 8:14:01 PM	Hot (Inferred)		Block blob	63.86 MiB	Available	...
📄 part-00003-tid-8193914268604594276-d57dfcab-5dee...	5/8/2024, 8:14:01 PM	Hot (Inferred)		Block blob	63.37 MiB	Available	...
📄 part-00004-tid-8193914268604594276-d57dfcab-5dee...	5/8/2024, 8:14:07 PM	Hot (Inferred)		Block blob	63.29 MiB	Available	...
📄 part-00005-tid-8193914268604594276-d57dfcab-5dee...	5/8/2024, 8:14:07 PM	Hot (Inferred)		Block blob	63.43 MiB	Available	...
📄 part-00006-tid-8193914268604594276-d57dfcab-5dee...	5/8/2024, 8:14:08 PM	Hot (Inferred)		Block blob	63.04 MiB	Available	...
📄 part-00007-tid-8193914268604594276-d57dfcab-5dee...	5/8/2024, 8:14:08 PM	Hot (Inferred)		Block blob	63.72 MiB	Available	...
📄 part-00008-tid-8193914268604594276-d57dfcab-5dee...	5/8/2024, 8:14:13 PM	Hot (Inferred)		Block blob	63.54 MiB	Available	...

## Implementing Streaming pipeline using Medallion Architecture:

### 1. Reading Books\_details as a streaming source.

Medallion Architecture For Books Details

▶ ✓ 08:19 PM (1s) 2 Python

```
(spark.readStream
  .format("cloudFiles")
  .option("cloudFiles.format", "parquet")
  .option("cloudFiles.schemaLocation", "/mnt/amazonbooksreviews/Transformed Data/Books_Details_cleaned")
  .load("/mnt/amazonbooksreviews/Transformed Data/Books_Details_cleaned")
  .createOrReplaceTempView("Books_Details_Raw"))
```

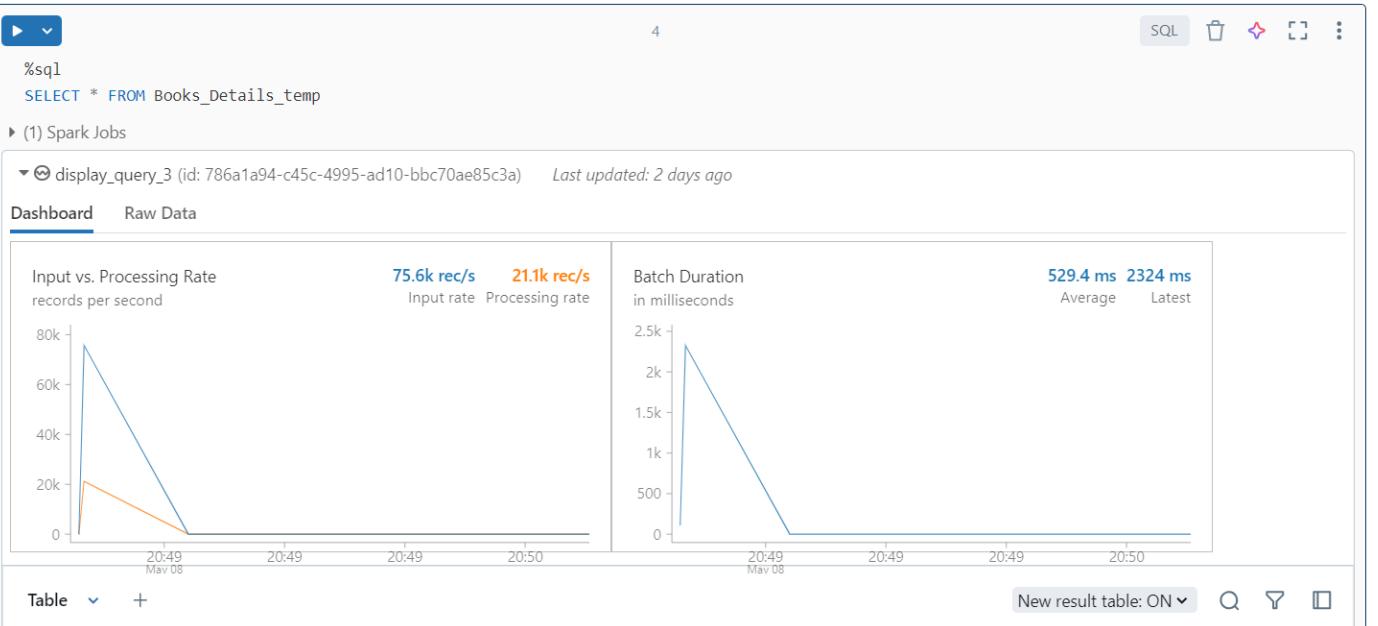
▶ (1) Spark Jobs

### 2. Creating Temp view

▶ ✓ 08:22 PM (<1s) 3

```
%sql
CREATE OR REPLACE TEMPORARY VIEW Books_Details_temp AS (
  SELECT *
  FROM Books_Details_Raw
)
```

OK



### 3. Reading Books\_Reviews table as Stream

▶ ✓ 2 days ago (2s) 7

```
# Reading data from Books_Reviews as a Stream.

(spark.readStream
  .format("cloudFiles")
  .option("cloudFiles.format", "parquet")
  .option("cloudFiles.schemaLocation", "/mnt/amazonbooksreviews/Transformed Data/Books_ratings_cleaned")
  .load("/mnt/amazonbooksreviews/Transformed Data/Books_ratings_cleaned")
  .createOrReplaceTempView("Books_ratings_Raw"))
```

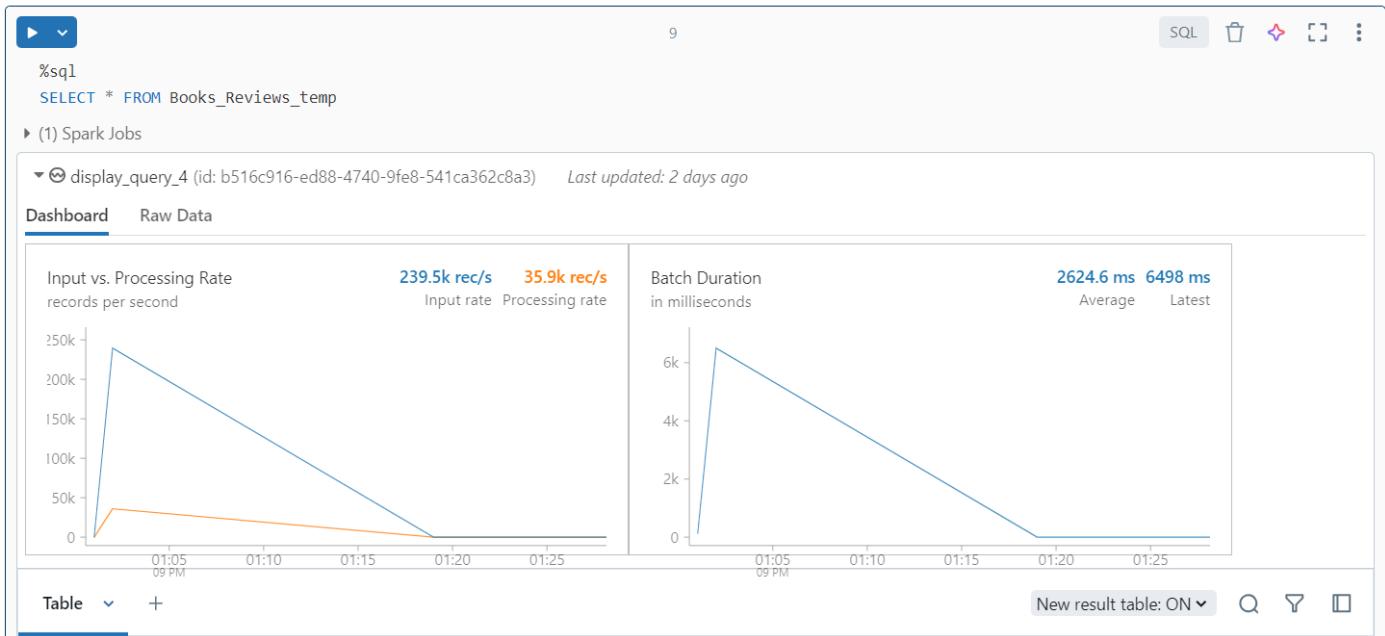
### 4. Creating Temp view

▶ ✓ 2 days ago (<1s) 8

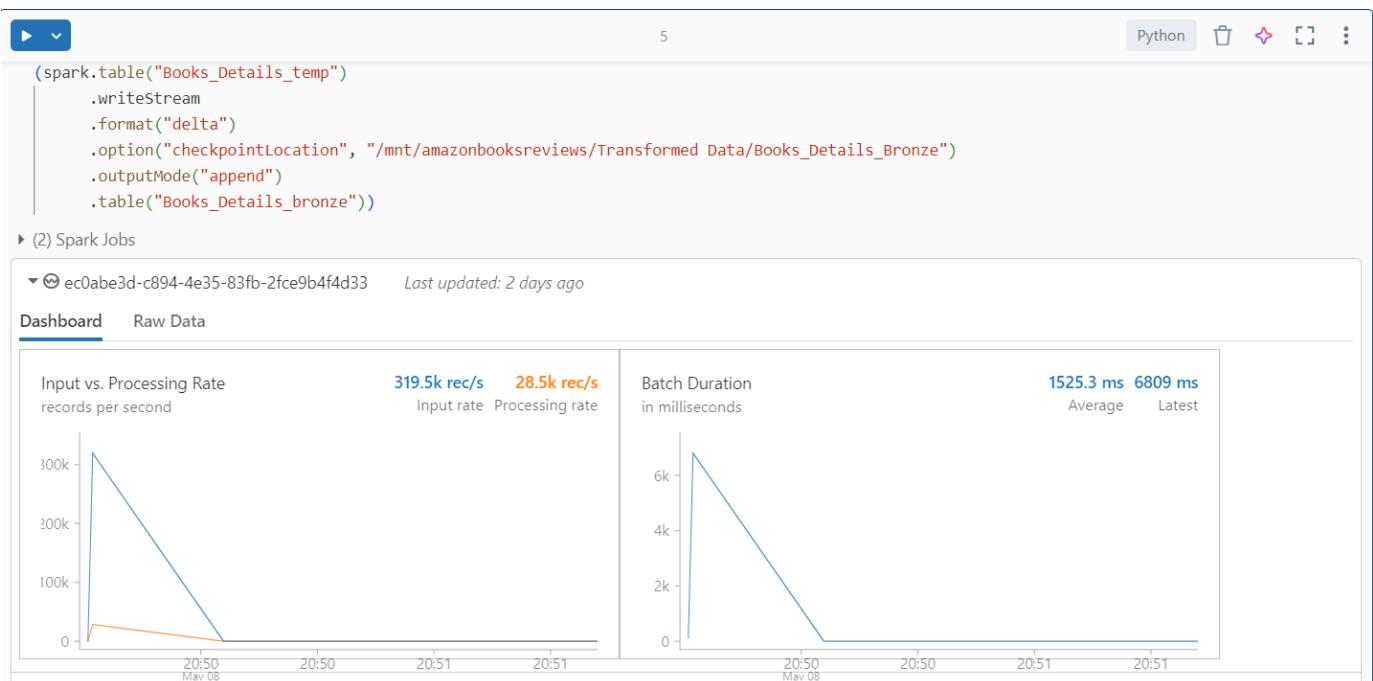
%sql  
--Creating a Temporary view

```
CREATE OR REPLACE TEMPORARY VIEW Books_Reviews_temp AS (
  SELECT *
  FROM Books_ratings_Raw )
```

OK



## 5. Loading raw data to Bronze Tables.



10

Python    

```
(spark.table("Books_Reviews_temp")
    .writeStream
    .format("delta")
    .option("checkpointLocation", "/mnt/amazonbooksreviews/Transformed Data/Books_reviews_Bronze")
    .outputMode("append")
    .table("Books_reviews_bronze"))

▶ (1) Spark Jobs
▶ c095cc1c-e4e1-4328-ab91-f3d1a76b6028 Last updated: 2 days ago
<pyspark.sql.streaming.query.StreamingQuery at 0x7f066430c990>
```

11

SQL    

%sql  
SELECT count(\*) FROM Books\_reviews\_bronze;

▶ (3) Spark Jobs

▶  \_sqlfd: pyspark.sql.dataframe.DataFrame = [count(1): long]

	1	2	3
1	2981936		

New result table: ON   

↓ 1 row | 0.86 seconds runtime Refreshed 2 days ago

This result is stored as PySpark data frame \_sqlfd and in the IPython output cache as Out[24]. [Learn more](#)

12

%sql  
DESCRIBE EXTENDED Books\_reviews\_bronze;

▶ (2) Spark Jobs

▶  \_sqlfd: pyspark.sql.dataframe.DataFrame = [col\_name: string, data\_type: string ... 1 more field]

	A <sub>c</sub> col_name	A <sub>c</sub> data_type	A <sub>c</sub> comment
13	# Delta Statistics Columns		
14	Column Names	> Id, review_score, Title, profileName, review_text, review_Time, Review_helpfulness, review_time_unix, User_id, _rescued_data, review_su...	
15	Column Selection Method	first-32	
16			
17	# Detailed Table Information		
18	Catalog	hive_metastore	
19	Database	default	
20	Table	Books_reviews_bronze	
21	Created Time	Wed May 08 17:00:15 UTC 2024	
22	Last Access	UNKNOWN	
23	Created By	Spark 3.5.1	
24	Type	MANAGED	
25	Location	dbfs:/user/hive/warehouse/books_reviews_bronze	
26	Provider	delta	
27	Owner	root	

## 6. Reading Bronze tables as Streams.

05:30 PM (<1s) 14 Python

```
# reading data from bronze table as a stream and creating a temp view

(spark.readStream
 | .table("Books_Details_bronze")
 | .createOrReplaceTempView("Books_Details_bronze_temp"))
```

+ Code + Text

05:30 PM (<1s) 15

```
(spark.readStream
 | .table("Books_reviews_bronze")
 | .createOrReplaceTempView("Books_reviews_bronze_temp"))
```

## 7. Loading Data into Silver Table

05:31 PM (<1s) 16 SQL

```
%sql

--Creating Temp View for silver table

CREATE OR REPLACE Temporary View Amazon_books_review_temp AS (
  SELECT b.Title, authors, publisher, YEAR(to_date(Year_of_publish)) as Year_of_publish , categories, Ratings_Count,
  User_id, profileName, Review_helpfulness, review_score, review_Time
  FROM Books_Details_bronze_temp a
  INNER JOIN Books_reviews_bronze_temp b
  ON a.Title = b.Title
  WHERE YEAR(review_Time) > 2010)
```

OK

17 Python

```
# Loading Data into Silver table

(spark.table("Amazon_books_review_temp")
 | .writeStream
 | .format("delta")
 | .option("checkpointLocation", "/mnt/amazonbooksreviews/Transformed Data/Amazon_books_reviews_silver")
 | .outputMode("append")
 | .table("Amazon_Books_Reviews_Silver"))

(2) Spark Jobs
```

8127f851-c093-42a6-b306-5527bdd73c48 Last updated: 7 minutes ago

Dashboard Raw Data

The dashboard displays three charts:

- Input vs. Processing Rate:** A line chart showing records per second. The input rate is 0 rec/s and the processing rate is 55.2k rec/s. The x-axis shows time from 17:35 to 17:37 on May 10.
- Batch Duration:** A line chart showing batch duration in milliseconds. The average batch duration is 9597.1 ms and the latest is 57.5 ms. The x-axis shows time from 17:35 to 17:37 on May 10.
- Aggregation State:** A line chart showing distinct keys over time. The count remains constant at 821k from 17:35 to 17:37 on May 10.

<pyspark.sql.streaming.query.StreamingQuery at 0x7ff5949db310>

▶ ✓ 5 minutes ago (1s)

```
%sql
SELECT count(*) FROM Amazon_Books_Reviews_Silver
```

► (3) Spark Jobs

▶ \_sqldf: pyspark.sql.dataframe.DataFrame = [count(1): long]

	1 <sup>2</sup> <sub>3</sub> count(1)
1	509681

## 8. Creating Gold temp views

▶ ✓ 06:05 PM (<1s) 21

```
# Reading Silver table as a Stream
```

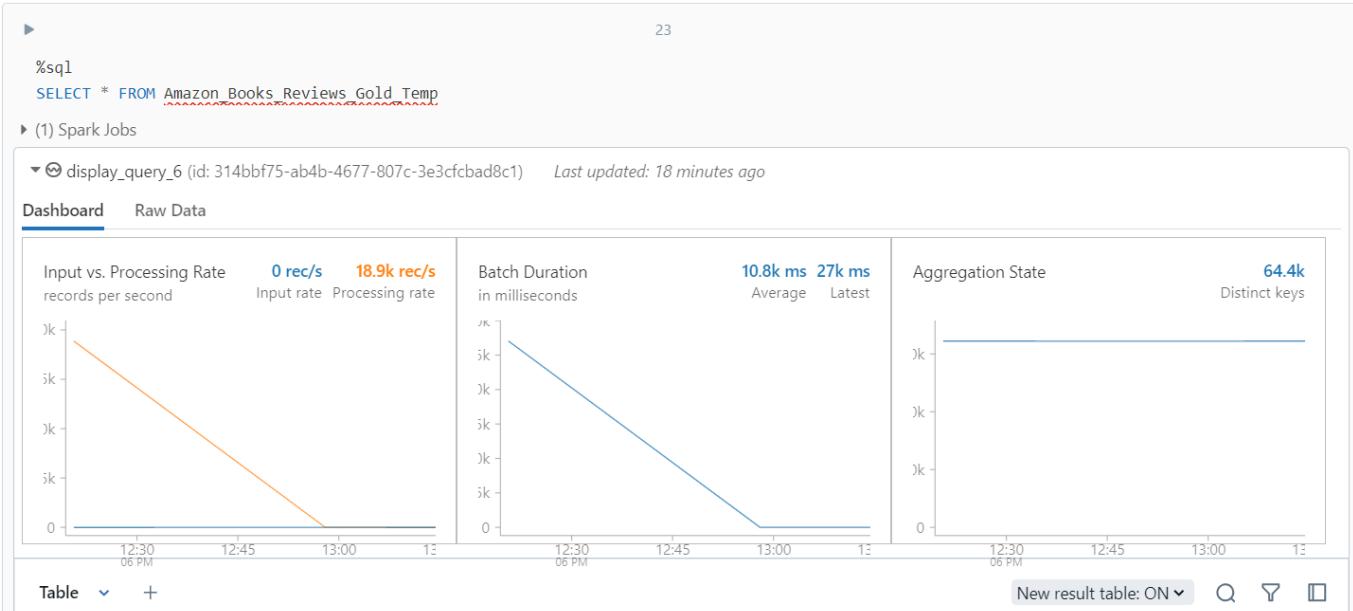
```
(spark.readStream
  .table("Amazon_Books_Reviews_Silver")
  .createOrReplaceTempView("Amazon_Books_Reviews_Silver_Temp"))
```

▶ ✓ 06:12 PM (<1s) 22

```
%sql
-- creating gold temp view with some aggregations
```

```
CREATE OR REPLACE TEMPORARY VIEW Amazon_Books_Reviews_Gold_Temp AS (
  SELECT title, Year_of_publish, categories, Count(user_id) as users_count
  FROM Amazon_Books_Reviews_Silver_Temp
  GROUP BY title, Year_of_publish, categories
)
```

OK



## 9. Writing Data into Gold Table

24

```
# Creating Gold delta table from temp view.
# using 'complete' as outputMode since streaming pipeline does not allow aggregations on 'append' mode.

(spark.table("Amazon_Books_Reviews_Gold_Temp")
 .writeStream
 .format("delta")
 .option("checkpointLocation", "/mnt/amazonbooksreviews/Transformed Data/books_users_count")
 .outputMode("complete")
 .table("gold_books_users_count"))
```

(1) Spark Jobs

d98d0054-4b1b-44f2-9810-c0bbc4bc29be Last updated: 15 minutes ago

[Dashboard](#) [Raw Data](#)

<pyspark.sql.streaming.query.StreamingQuery at 0x7ff5644af150>

25

```
%sql
DESCRIBE EXTENDED gold_books_users_count
```

(2) Spark Jobs

\_sqldf: pyspark.sql.dataframe.DataFrame = [col\_name: string, data\_type: string ... 1 more field]

Table	col_name	data_type	comment
8	Column Selection Method	first-32	
9			
10	# Detailed Table Information		
11	Catalog	hive_metastore	
12	Database	default	
13	Table	gold_books_users_count	
14	Created Time	Fri May 10 12:45:28 UTC 2024	
15	Last Access	UNKNOWN	
16	Created By	Spark 3.5.1	
17	Type	MANAGED	
18	Location	dbfs:/user/hive/warehouse/gold_books_users_count	
19	Provider	delta	
20	Owner	root	
21	Is_managed_location	true	
22	Table Properties	[delta.enableDeletionVectors=true,delta.feature.deletionVectors=supported,delta.minReaderVersion=3,delta.minWriterVersion=7]	

## STEP 4: ANALYTICS AND VISUALIZATION

### 1. Set up Azure Synapse Analytics

The screenshot shows the Microsoft Power BI Data workspace interface. On the left, there's a sidebar with icons for Home, Databases, Workbooks, and Reports. The main area has tabs for 'Data' (selected), 'Workspace' (under 'Lake database'), and 'Linked'. A search bar at the top says 'Filter resources by name'. In the center, there's a 'Tables' section with a 'Get started' button and a note to 'Select tables to build your database.' To the right, a large panel for creating a new database is open. It asks to 'Choose a name for your Database. This name can be updated at any time until it is published.' The 'Name' field is set to 'AmazonBooksReviewsDB'. Below it are fields for 'Description' (empty) and 'Storage settings for database'. Under 'Linked service', it shows 'amazonbooksreviews-WorkspaceD...' and 'Input folder' 'amzon-books-review/AmazonB...'. There are also 'Edit' and 'Delete' buttons.

This screenshot is similar to the previous one but focuses on creating a new table. The 'Table' tab in the top navigation bar is selected. A dropdown menu shows options: 'Custom' (selected), 'From template', and 'From data lake'. The rest of the interface is identical to the first screenshot, including the 'Get started' button and the database creation panel on the right.

## 2. Tables Creation and Data loading

The screenshot shows the 'AmazonBooksReviewsDB' workspace with the 'Tables' tab selected. A table named 'books\_users\_count' is highlighted with a blue border. Its schema is visible: 'title', 'Year\_of\_publish', 'categories', and 'users\_count'. Below the table, there are tabs for 'General' (selected), 'Columns', and 'Relationships'. The 'Name' field is set to 'books\_users\_count'. On the right side, there are several small icons for modifying the table structure.

The screenshot shows the 'Columns' tab of a table definition in Azure Data Studio. The table is named 'books\_users\_count' and contains four columns:

- title**: abc string, 8000 bytes, nullable
- Year\_of\_publish**: 123 integer, 4 bytes, nullable
- categories**: abc string, 8000 bytes, nullable
- users\_count**: 12l long, 8 bytes, nullable

Below this, the 'Data' workspace shows the table structure again, along with options to run a SQL script or notebook.

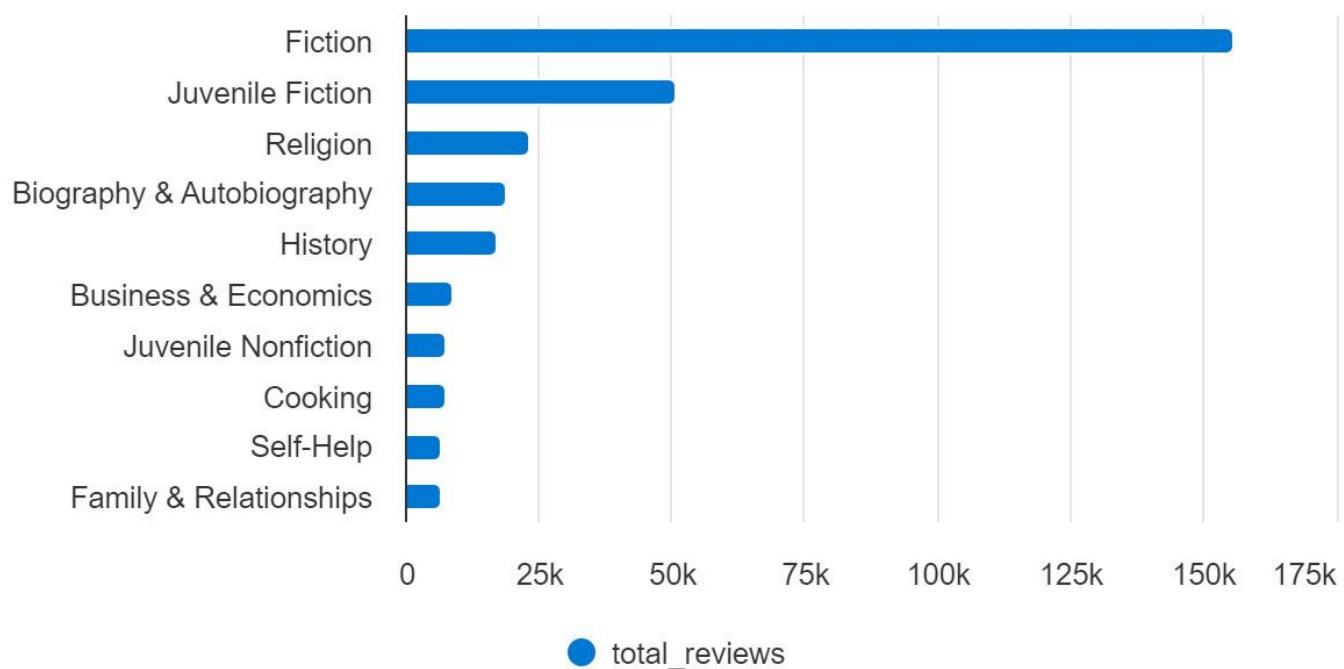
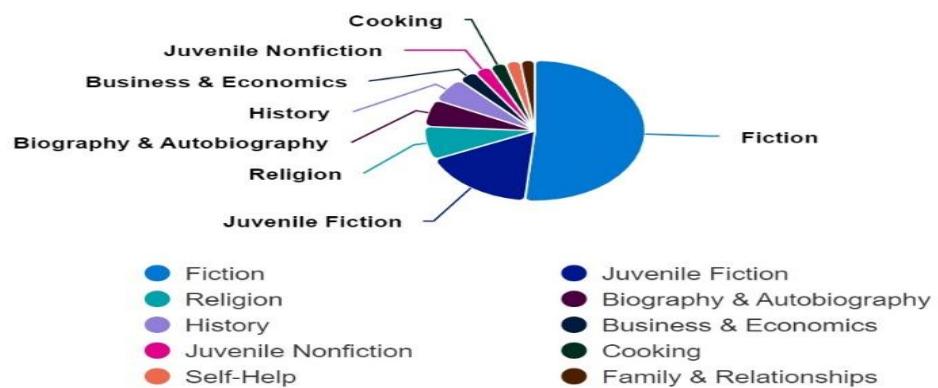
## Visualization:

→ Books categories with the greatest number of Reviews.

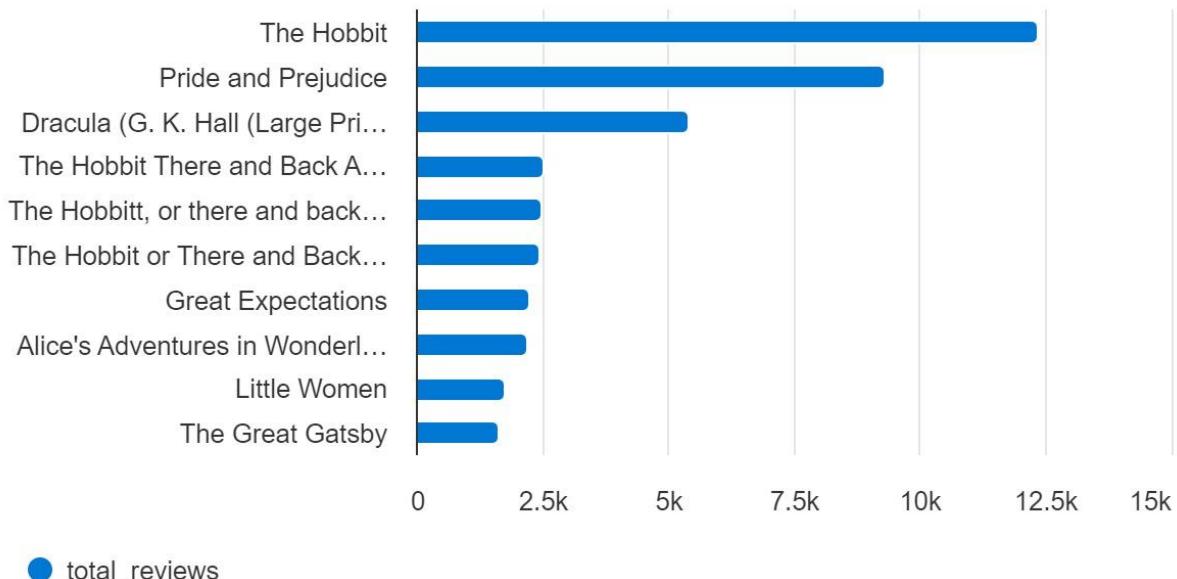
```

1 -- TOP 10 categories of books that got most number of reviews
2
3 SELECT
4     categories,
5     sum(users_count) as total_reviews
6 FROM books_users_count
7 GROUP BY categories
8 ORDER BY total_reviews DESC
9 OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY;
10

```



➔ Books with Most Reviews



## → Top 10 Highest Rated Books



## → Users with the greatest number of reviews.

