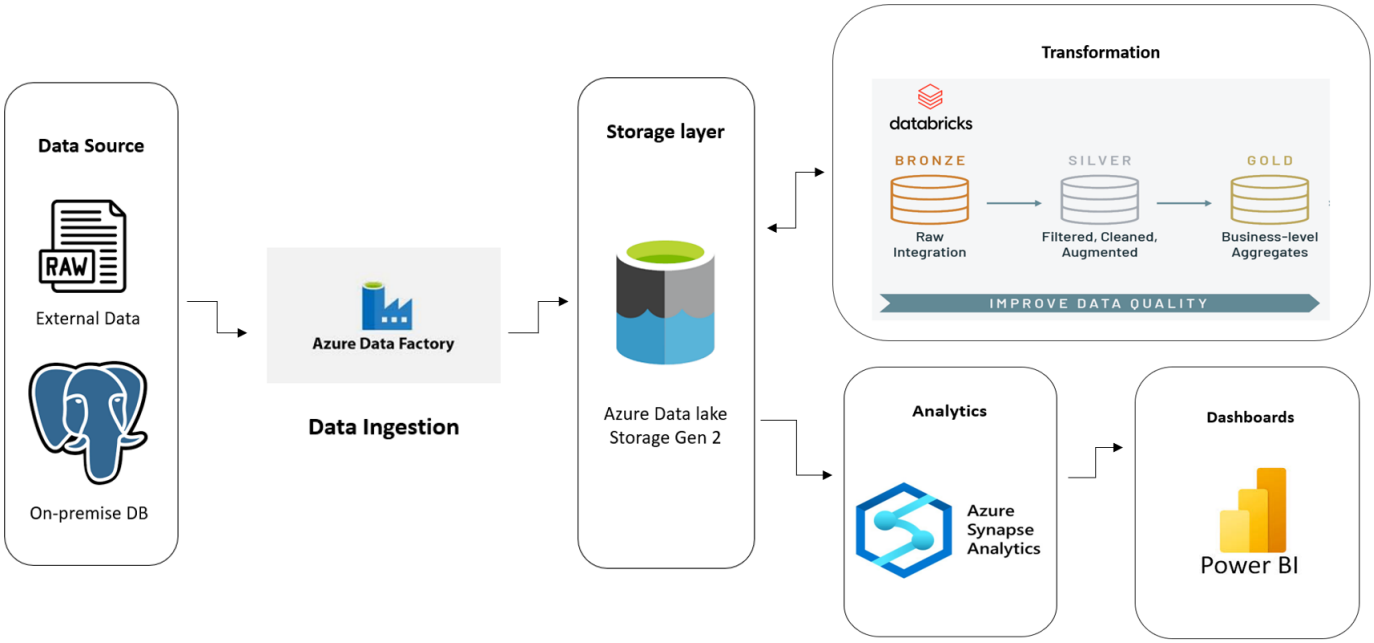
Data Engineering Project

Title: Amazon Books Reviews

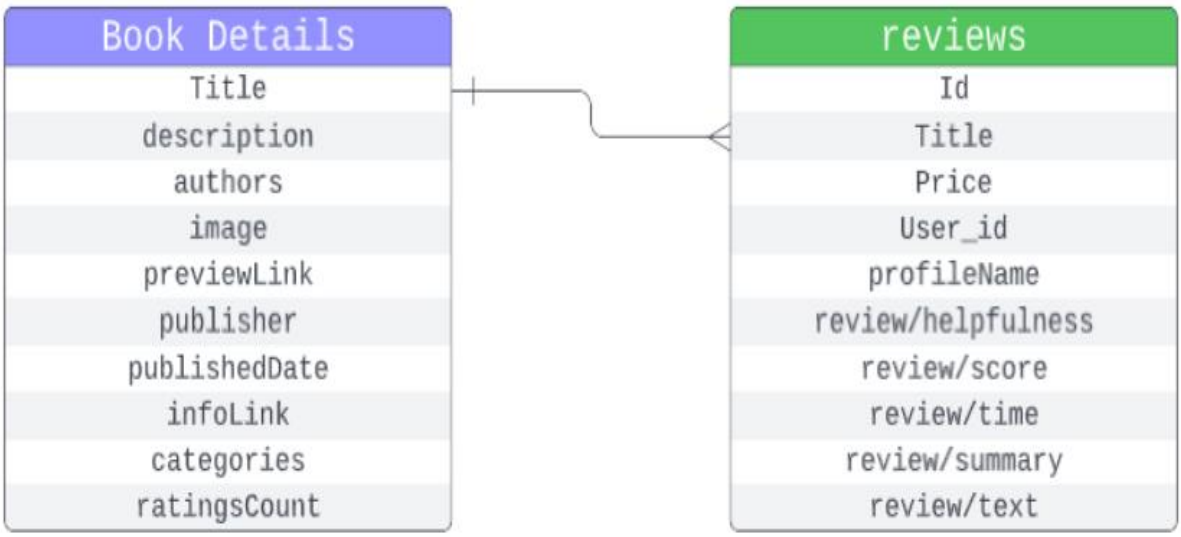
Prepared By: Rooban M

INDEX:

DESIGN AND ALGORITHM:



SCHEMA OF TABLES:



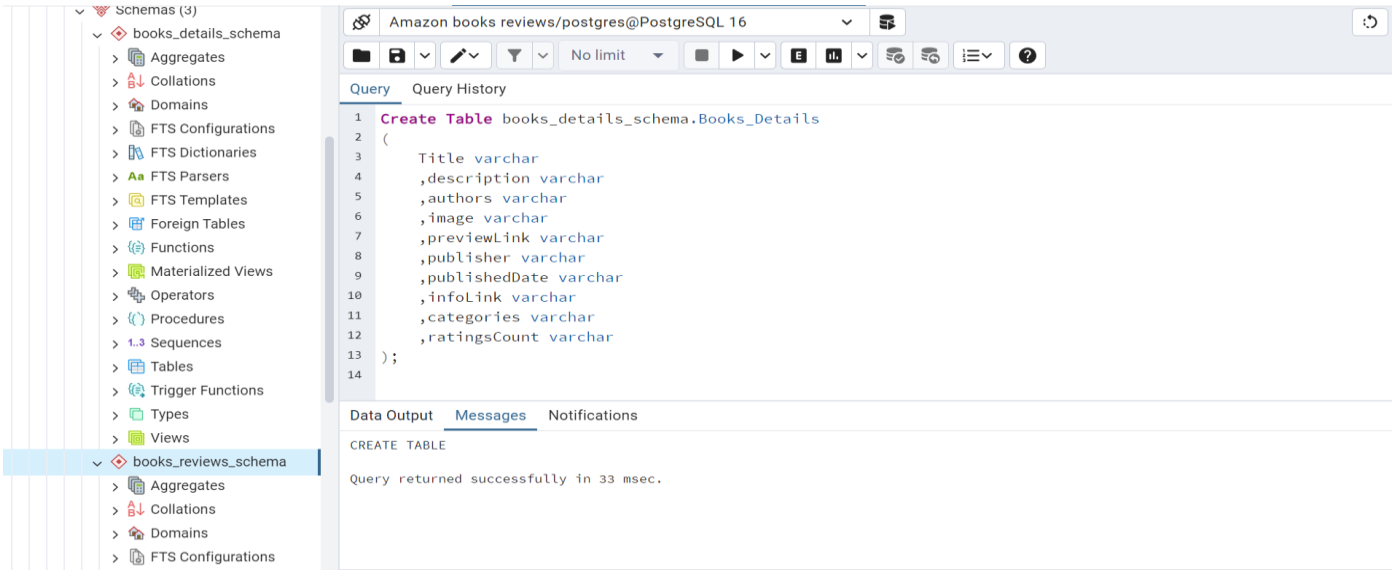
RAW DATA DETAILS:

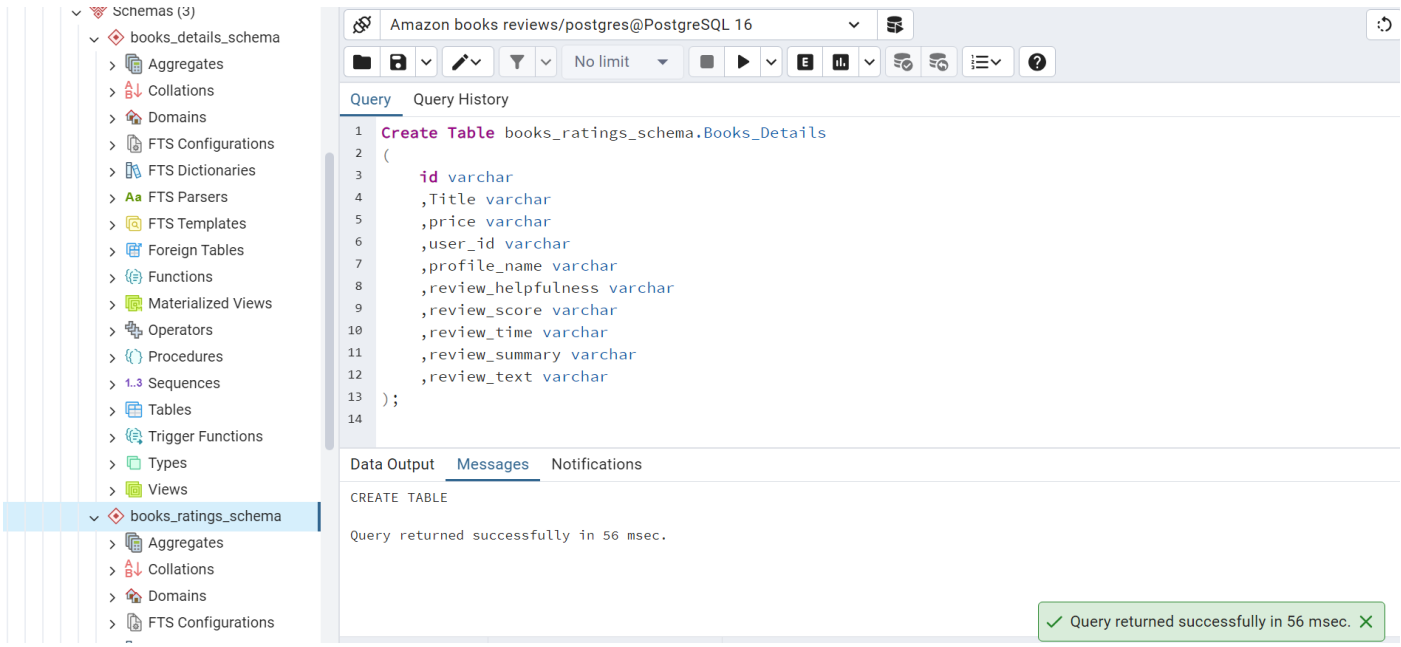
Data source: Kaggle (Link)

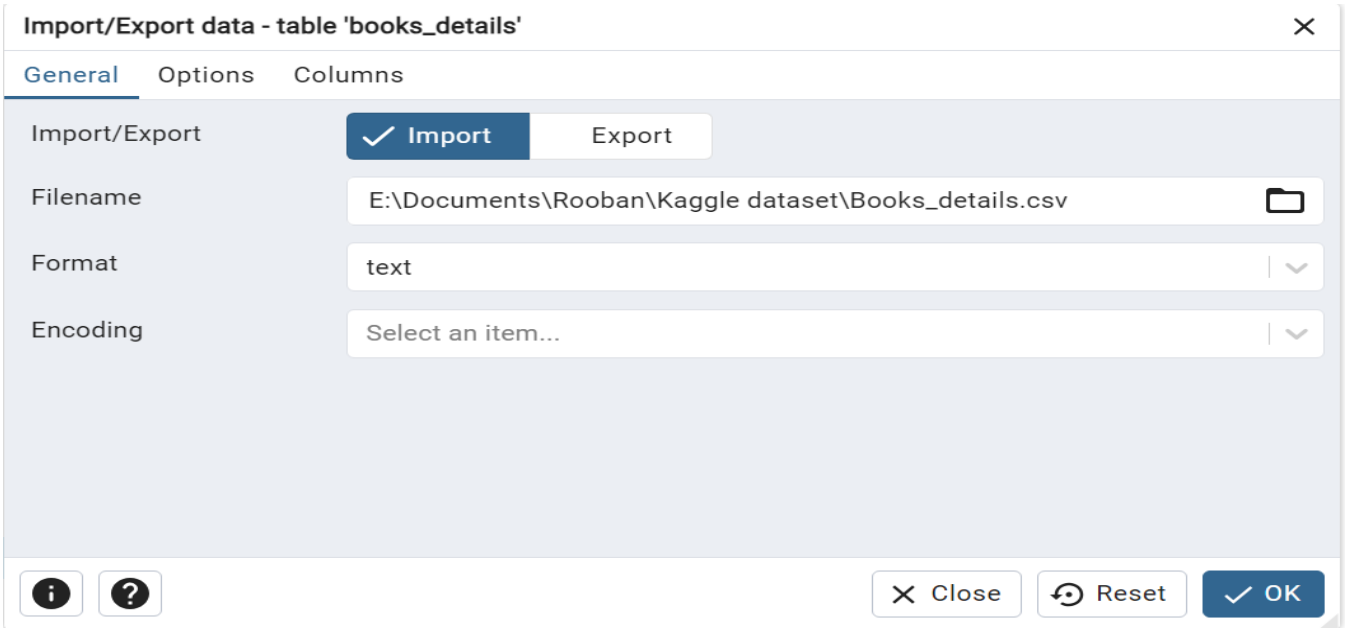
Table Name	File Size (Mb)	File Format	No.of.Columns	No.of.Rows
Books_Details	181	CSV	10	212404
Books_reviews	2860	CSV	10	3000000

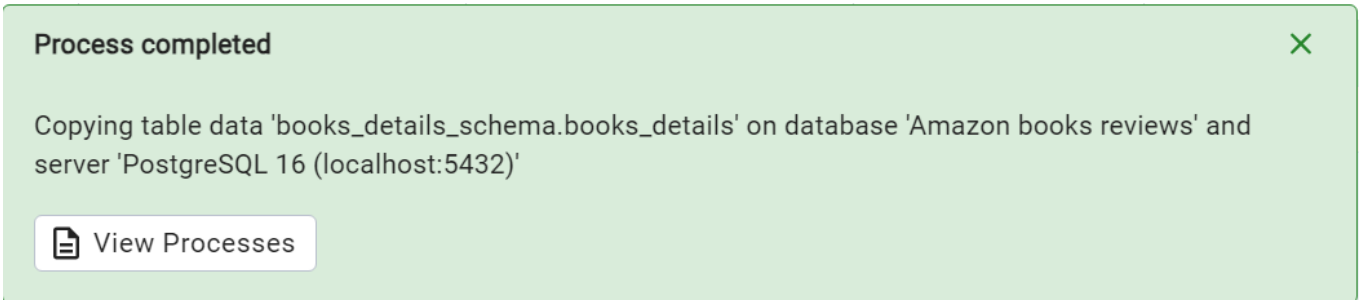
STEP 1:DATA LOADING AND PREPARATION

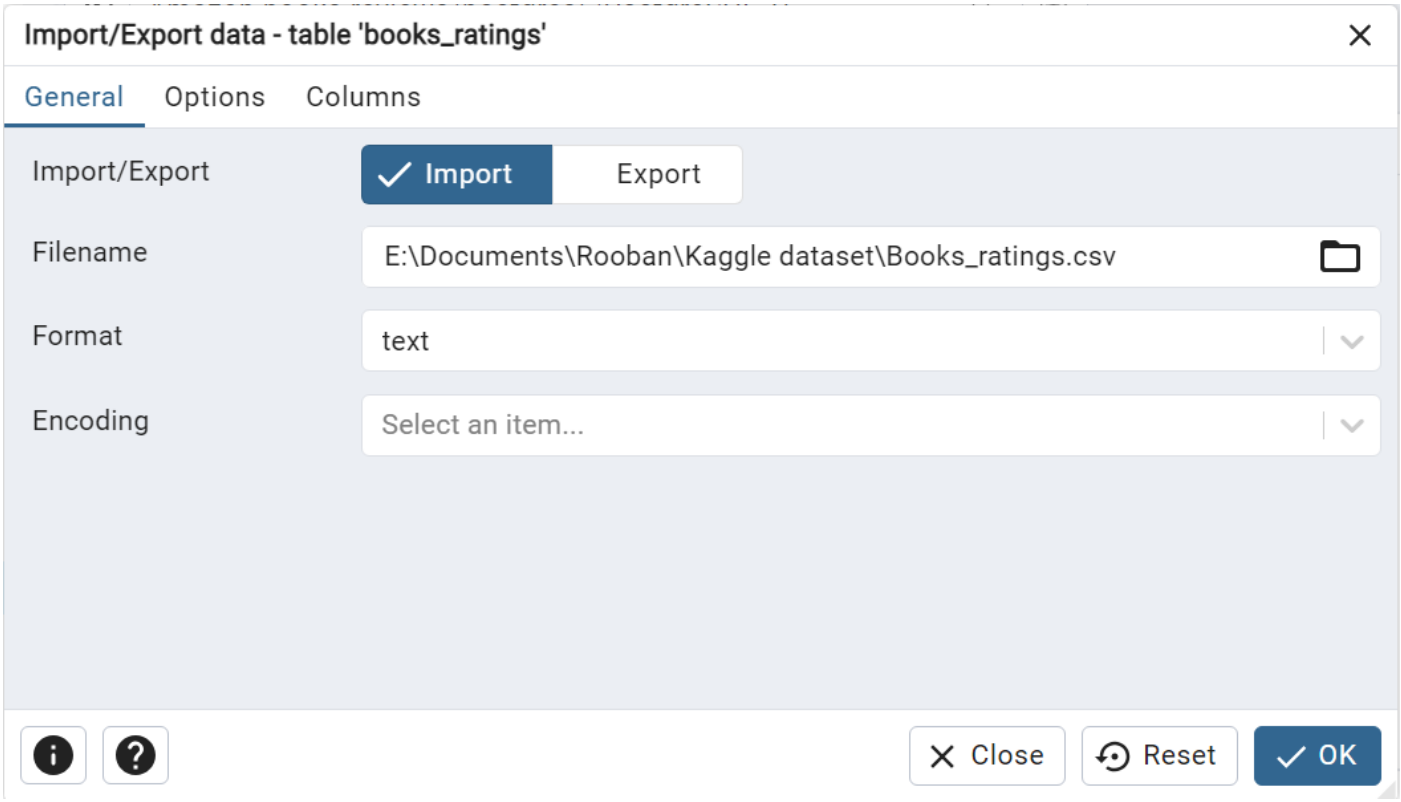
1. Creating Tables in PostgreSQL and loading all the .csv files

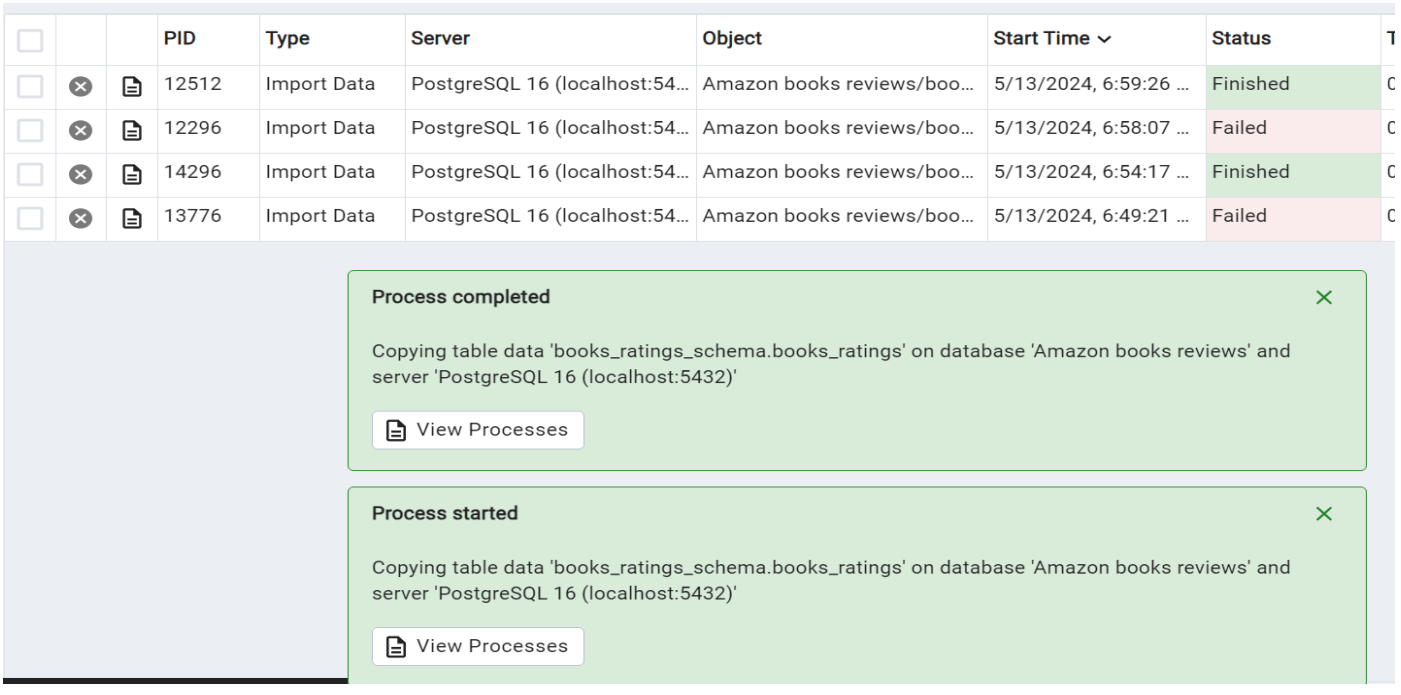


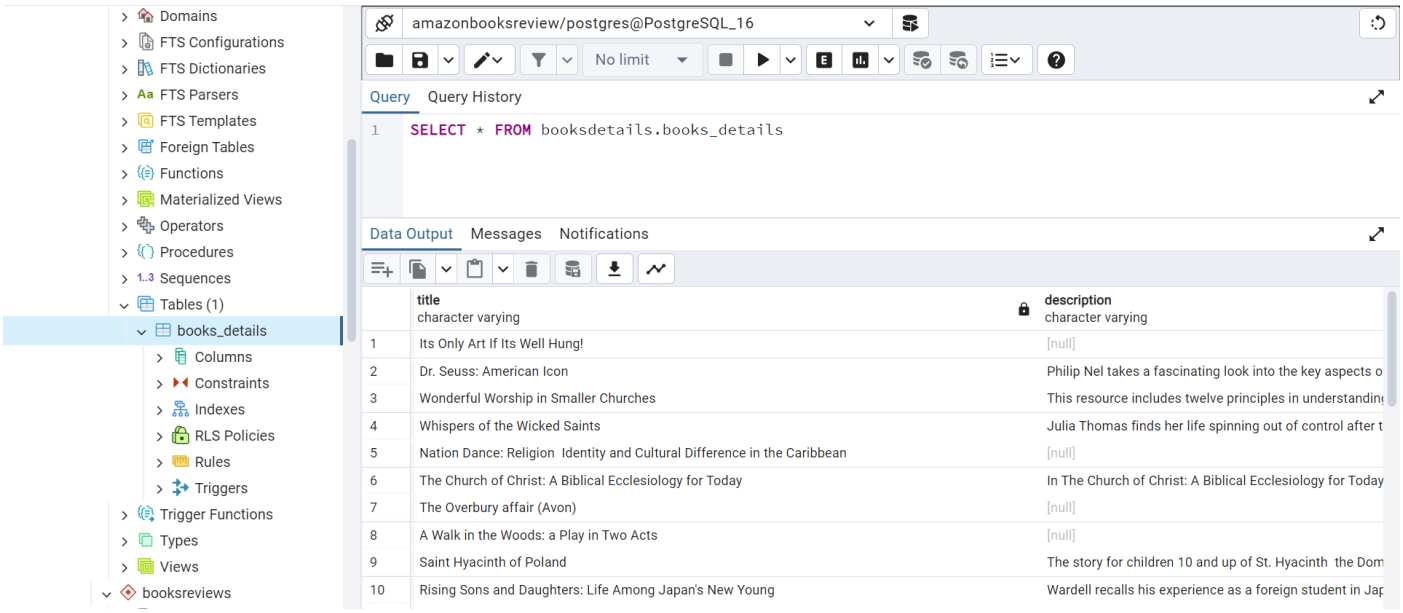

Schemas (3)
books_details_schema
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables
Trigger Functions
Types
Views
books_ratings_schema
Aggregates
Collations
Domains
FTS Configurations
Amazon books reviews/postgres@PostgreSQL 16
No limit
Query
Query History
Create Table books_ratings_schema.Books_Details
(
id varchar
,Title varchar
,price varchar
,user_id varchar
,profile_name varchar
,review_helpfulness varchar
,review_score varchar
,review_time varchar
,review_summary varchar
,review_text varchar
);
Data Output
Messages
Notifications
CREATE TABLE
Query returned successfully in 56 msec.
Query returned successfully in 56 msec.

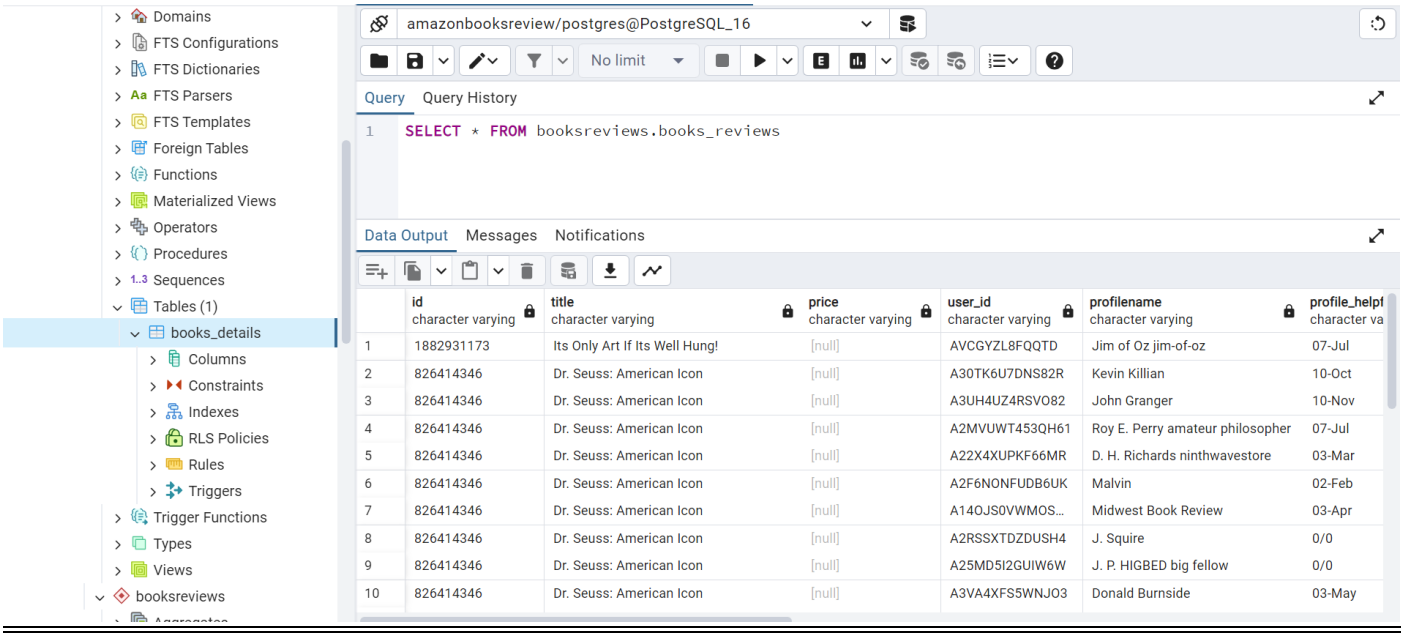

Import/Export data - table 'books_details'
General
Options
Columns
Import/Export
Import
Export
Filename
E:\Documents\Rooban\Kaggle dataset\Books_details.csv
Format
text
Encoding
Select an item...
Close
Reset
OK


Process completed
Copying table data 'books_details_schema.books_details' on database 'Amazon books reviews' and server 'PostgreSQL 16 (localhost:5432)'
View Processes


Import/Export data - table 'books_ratings'
General
Options
Columns
Import/Export
Import
Export
Filename
E:\Documents\Rooban\Kaggle dataset\Books_ratings.csv
Format
text
Encoding
Select an item...
Close
Reset
OK

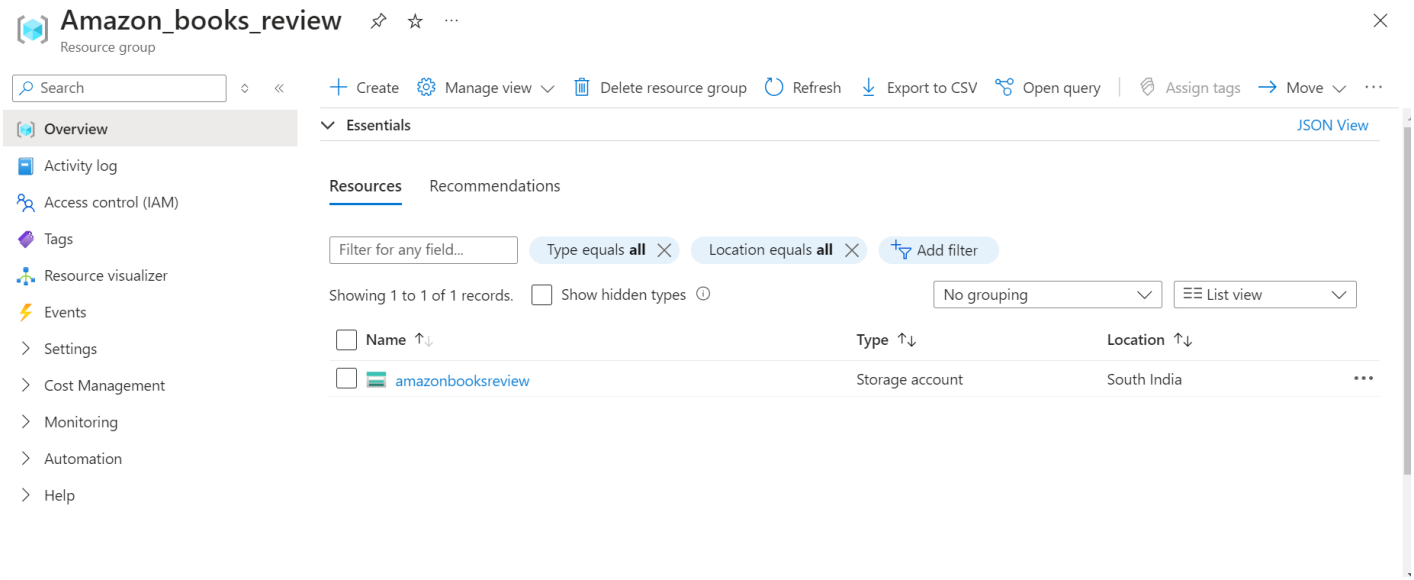

PID
Type
Server
Object
Start Time
Status
12512 Import Data PostgreSQL 16 (localhost:54... Amazon books reviews/boo... 5/13/2024, 6:59:26 ... Finished
12296 Import Data PostgreSQL 16 (localhost:54... Amazon books reviews/boo... 5/13/2024, 6:58:07 ... Failed
14296 Import Data PostgreSQL 16 (localhost:54... Amazon books reviews/boo... 5/13/2024, 6:54:17 ... Finished
13776 Import Data PostgreSQL 16 (localhost:54... Amazon books reviews/boo... 5/13/2024, 6:49:21 ... Failed
Process completed
Copying table data 'books_ratings_schema.books_ratings' on database 'Amazon books reviews' and server 'PostgreSQL 16 (localhost:5432)'
View Processes
Process started
Copying table data 'books_ratings_schema.books_ratings' on database 'Amazon books reviews' and server 'PostgreSQL 16 (localhost:5432)'
View Processes


Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
books_details
Columns
Constraints
Indexes
RLS Policies
Rules
Triggers
Trigger Functions
Types
Views
booksreviews
amazonbooksreview/postgres@PostgreSQL_16
No limit
Query
Query History
1 SELECT * FROM booksdetails.books_details
Data Output
Messages
Notifications
title character varying
description character varying
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

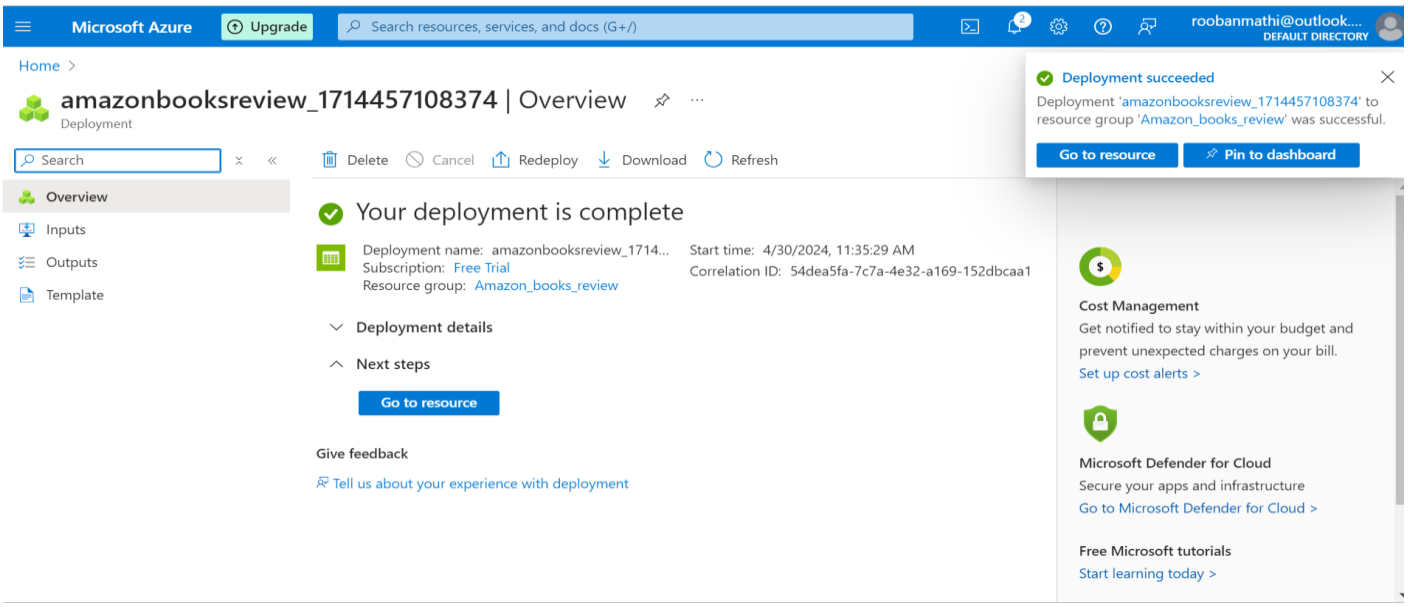

Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
books_details
Columns
Constraints
Indexes
RLS Policies
Rules
Triggers
Trigger Functions
Types
Views
booksreviews
amazonbooksreview/postgres@PostgreSQL_16
No limit
Query
Query History
1 SELECT * FROM booksreviews.books_reviews
Data Output
Messages
Notifications
id character varying
title character varying
price character varying
user_id character varying
profilename character varying
profile_helpf character va
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] A3UH4UZ4RSVO82 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] A2RSSXTDZDUSH4 J. Squire 0/0
9 826414346 Dr. Seuss: American Icon [null] A25MD5I2GUIW6W J. P. HIGBED big fellow 0/0
10 826414346 Dr. Seuss: American Icon [null] A3VA4XFS5WNJO3 Donald Burnside 03-May

STEP 2: CREATION OF AZURE STORAGE ACCOUNT

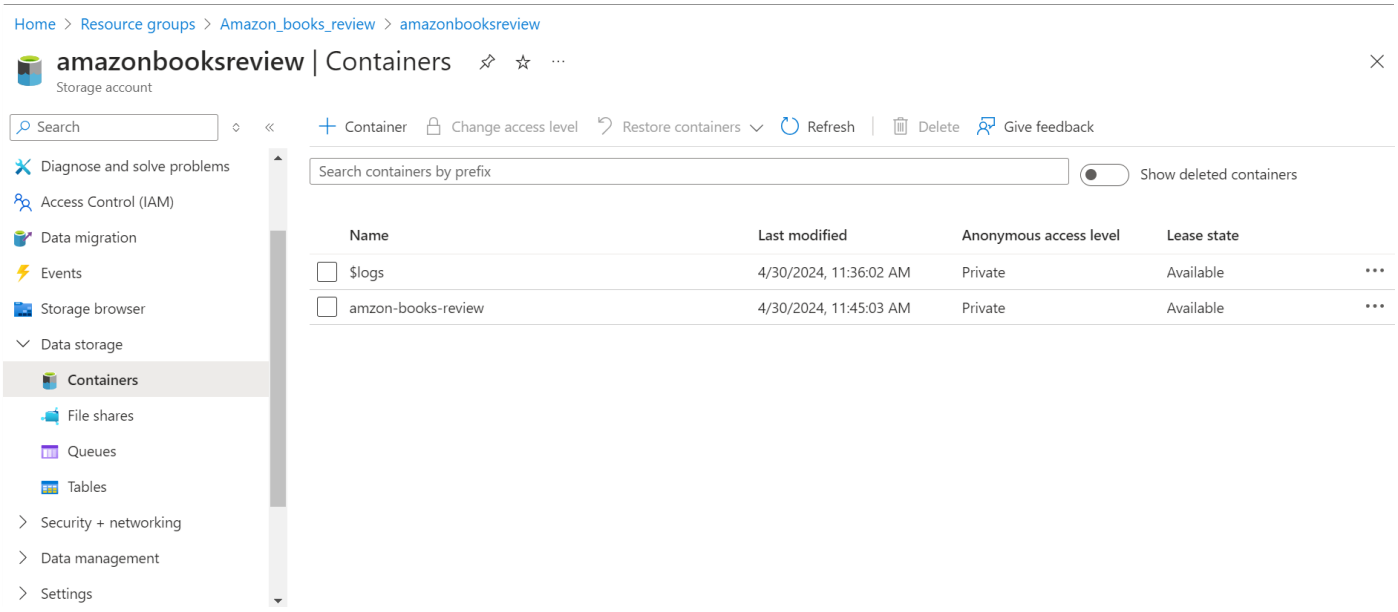
1. Creating azure Resource Group

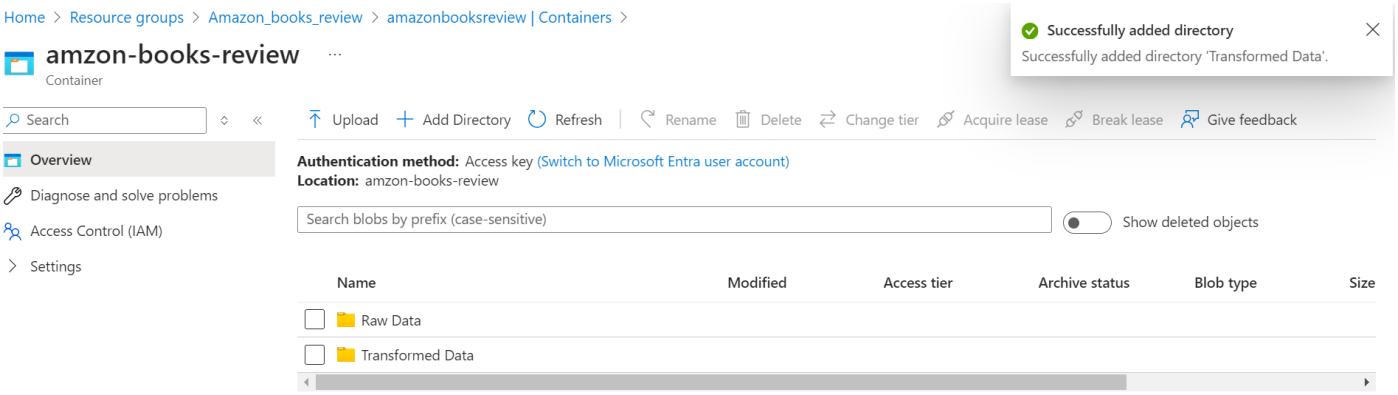


2. Created new storage account in Azure cloud to store raw and transformed data.



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

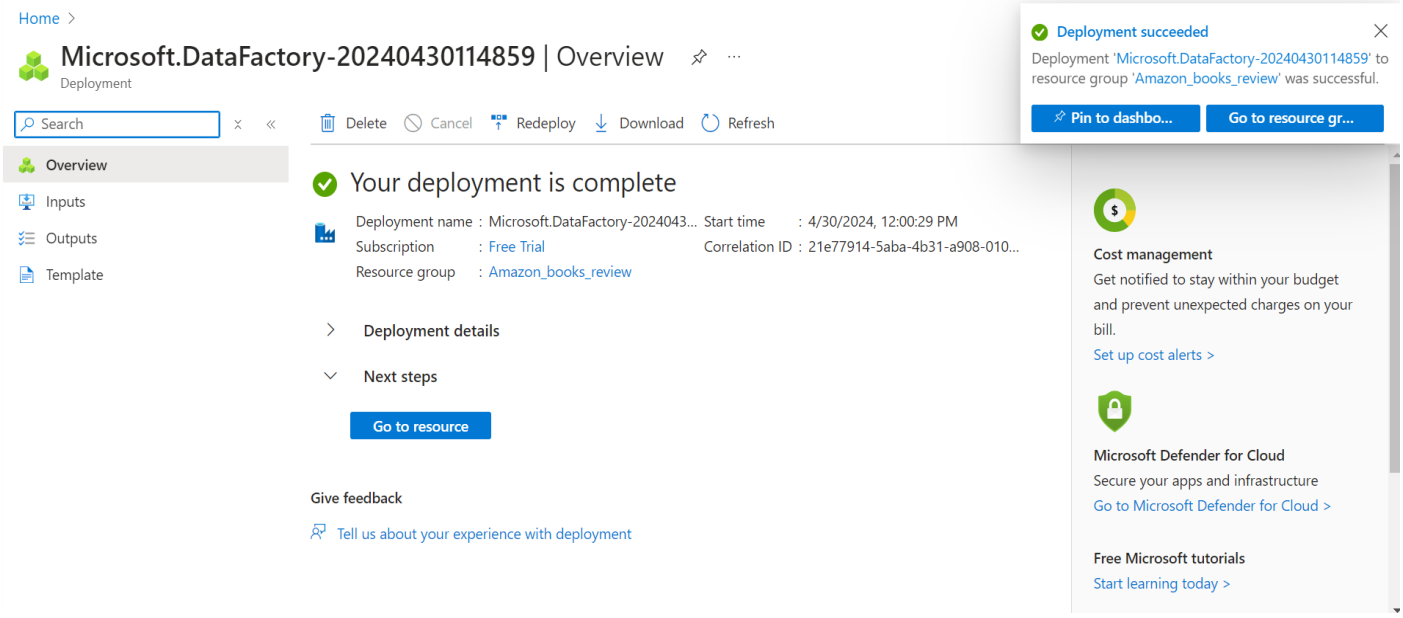




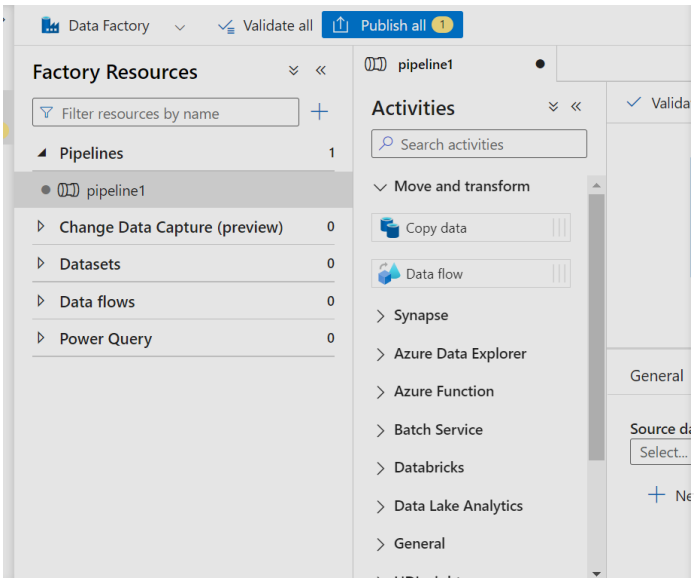
STEP 3: DATA INGESTION

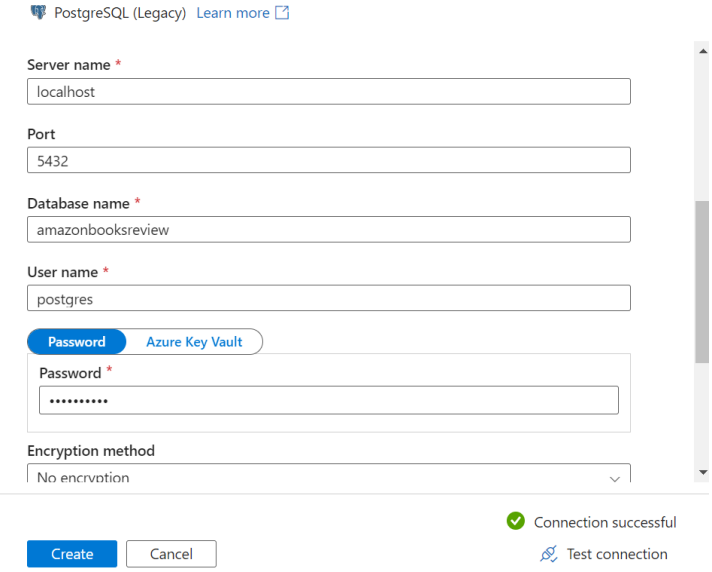
a. Data Ingestion Using Azure Data Factory

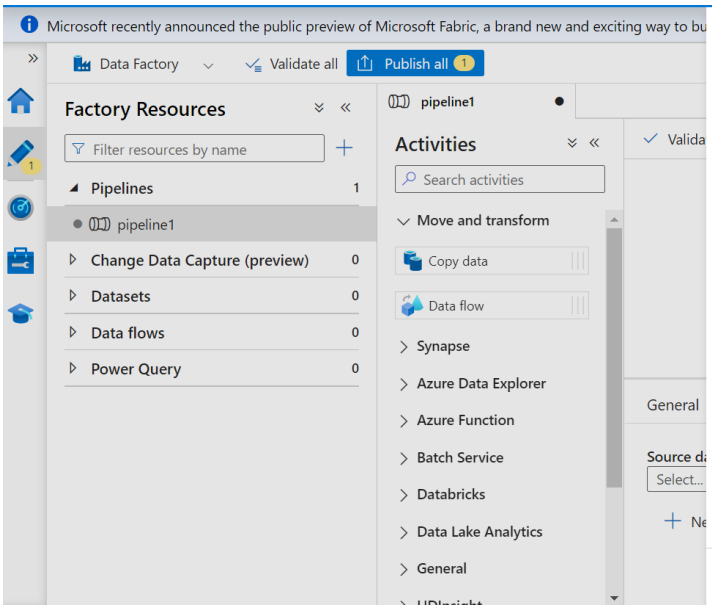
1. Deploying Azure Data Factory

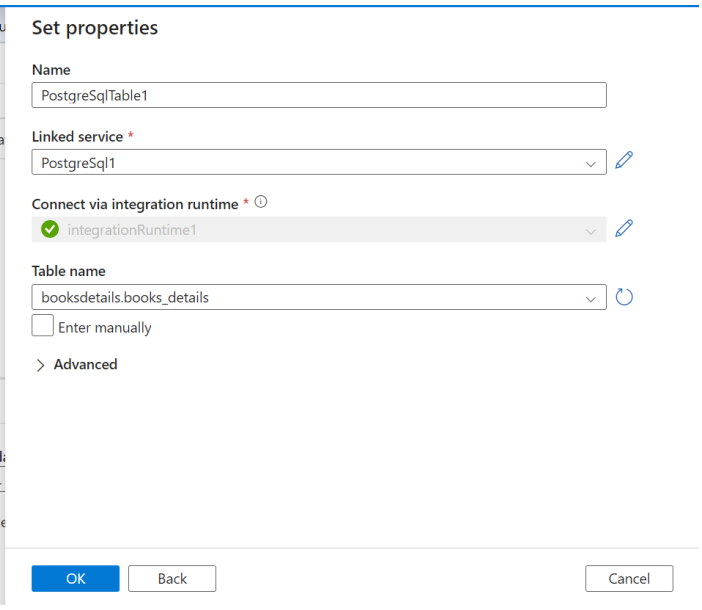


2. Connecting On-premise PostgreSQL Database to Data factory using Microsoft Integration runtime.

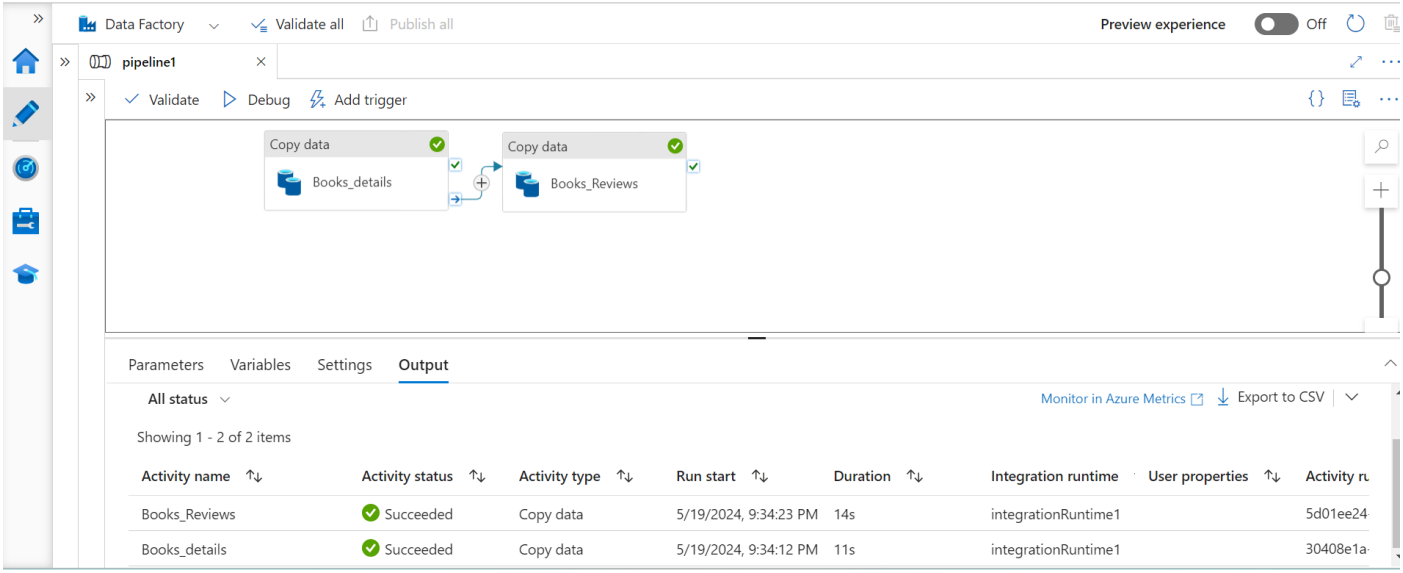


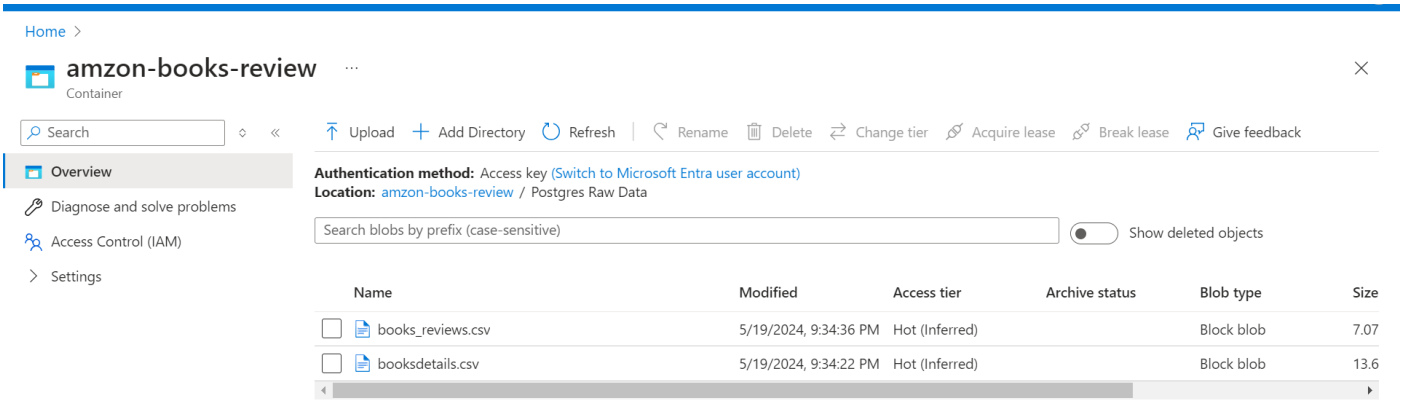




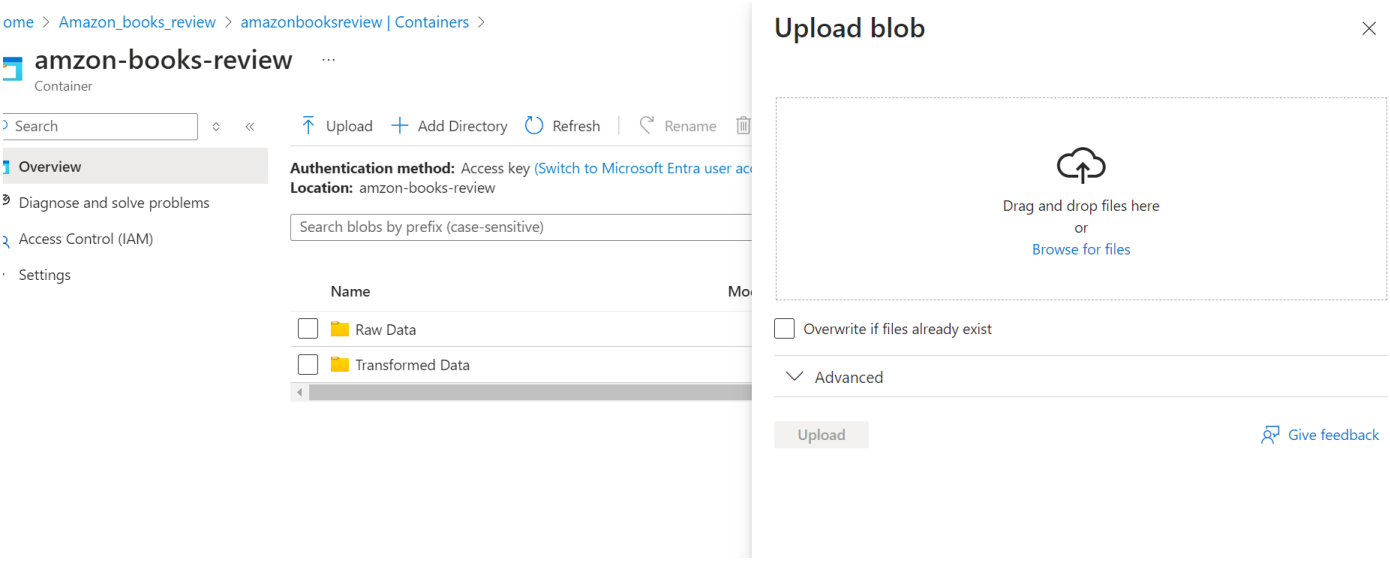


3. Creating Data Ingestion Pipeline and loading data into azure data lake.



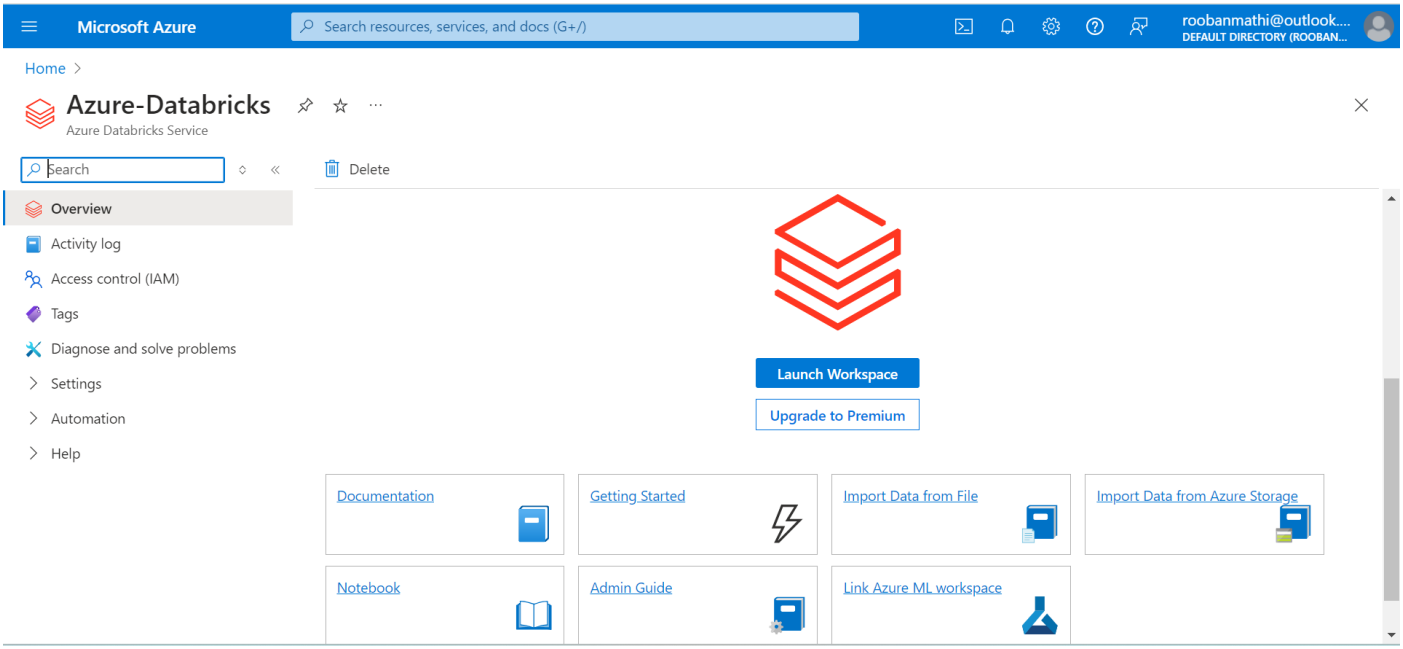


b. uploading Data files directly to Azure Data Lake from local.

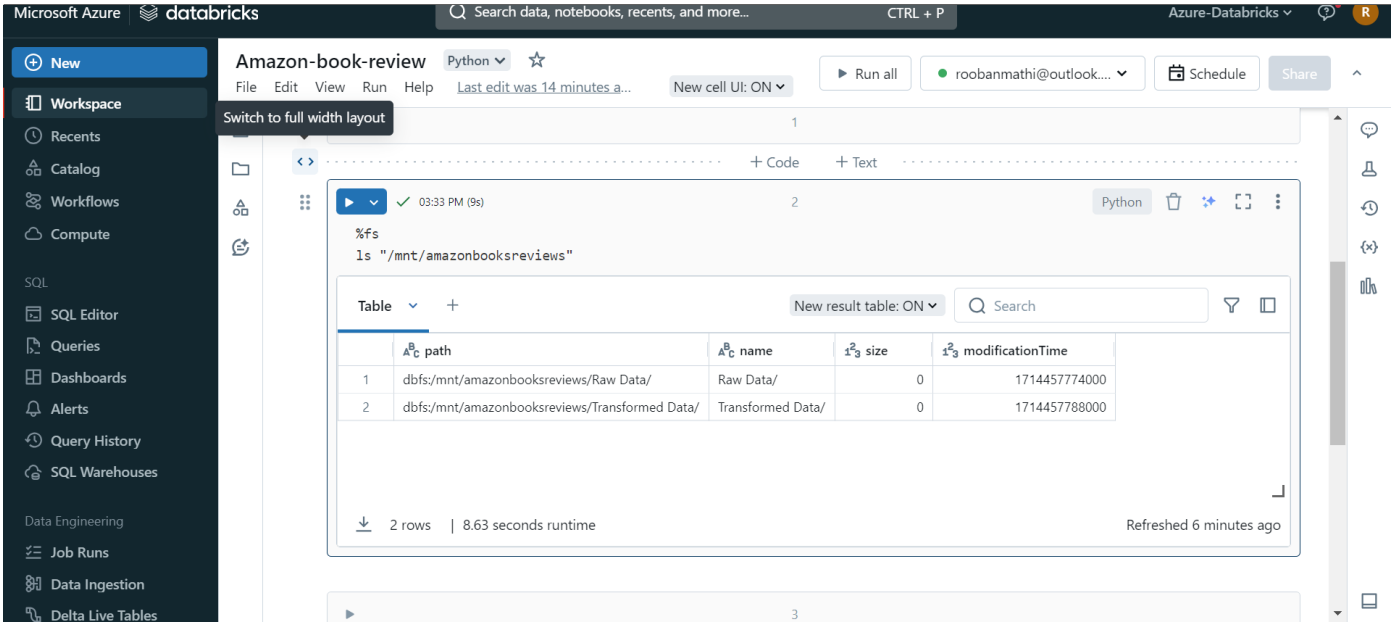


STEP 4. TRANSFORMATIONS

1. Creating Azure Databricks workplace



2. Connecting Azure Blob storage with Azure Databricks.



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.

<u>Original data: (Authors)</u>

		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 seperate 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	nterpretations+of+the+History+of...	['Religion']
27		['Psychology']
28	l=&source=gbs_api	null
29	&source=gbs_api	['Sports & Recreation']
30	ds+Cope+When+Things+Don%27...	null
31		round out the presidential collection. Lively narratives accompany each piece
32		['Biography & Autobiography']
33	_api	['Bible stories, English']
34		['History']
35		['New Thought']
36		null
37	Truly+ComprehensiveCollection+o...	['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 seperate 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:

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
null
null
null
null
exploring the president's background and biography as well as the work's artistic and historical significance. Taken together
1.0
null
1.0
null
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
(FloatType()))
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
NULL	NULL
NULL	NULL
NULL	NULL
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_3.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

infoLink
http://books.google.nl/books?id=399SPgAACAAJ&dq=Nation+Dance:+Religion,+Identity+and+Cultural+Difference+in+the+Caribbe...
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=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=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=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
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
http://books.google.nl/books?id=I5NOAQAACAAJ&dq=Gold+and+greenstone&hl=&source=gbs_api

Cleaned data:

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=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
null
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
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=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=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. Droge|            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 & Autob...|
|"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                                                    Python
# 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:

```
Books_ratings_raw.display()
```

(1) Spark Jobs

Table

New result table: ON

	Price	User_id	profileName	review/helpfulness	review/score	review/time	review/summary
8	null	A2RSSXTDZDUSH4	J. Squire	0/0	5.0	1231200000	Academia At It's Best
9	null	A25MD5I2GUIW6W	"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	AZ0IOBU20TBOP	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	AGKGOH65VTRR4	"Cynthia L. Lajoy ""Cindy La Joy"""	1/1	5.0	1248307200	Not Just for Pastors!
14	19.40	A3OQWLU31BU1Y	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	AUR0VA5H0C66C	"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 qlee	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) 6 Python

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

# splitting the column based on '/' to seperate 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 seperate 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) 9

```
# 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

```
# 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.DataFrame = [Id: string, Title: string ... 11 more fields]

```
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)
```

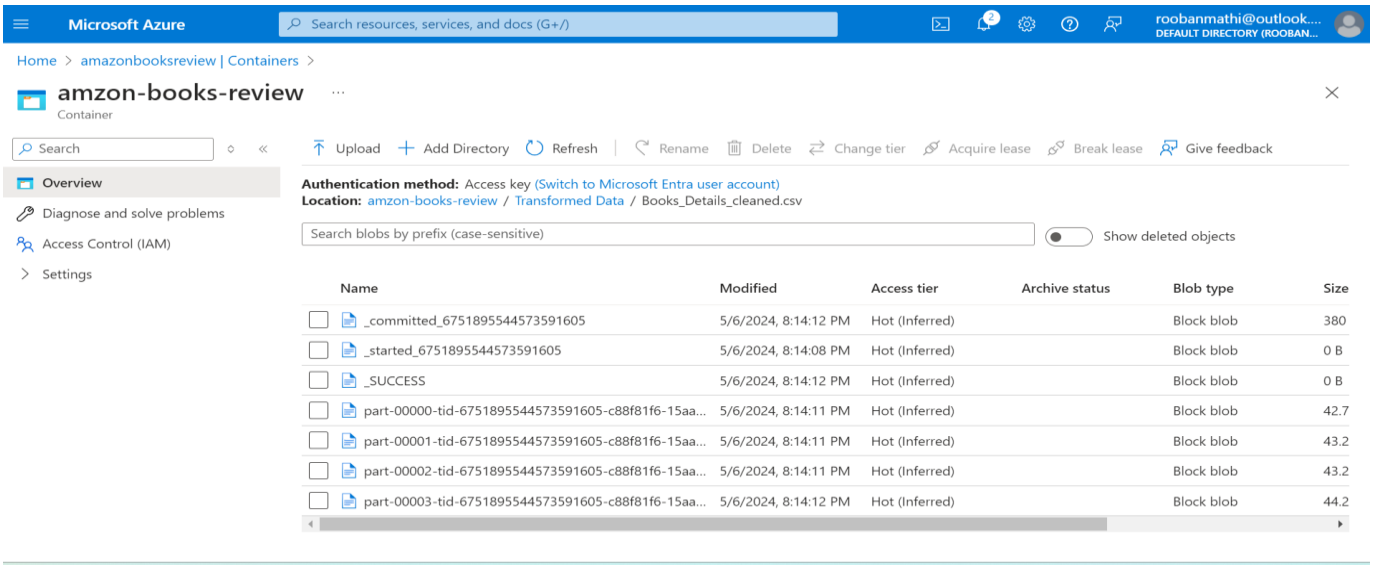
```
# Null Handling for All Columns
mean_value = Books_ratings_cleaned_4.agg(mean("review/score")).collect()[0][0]

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.

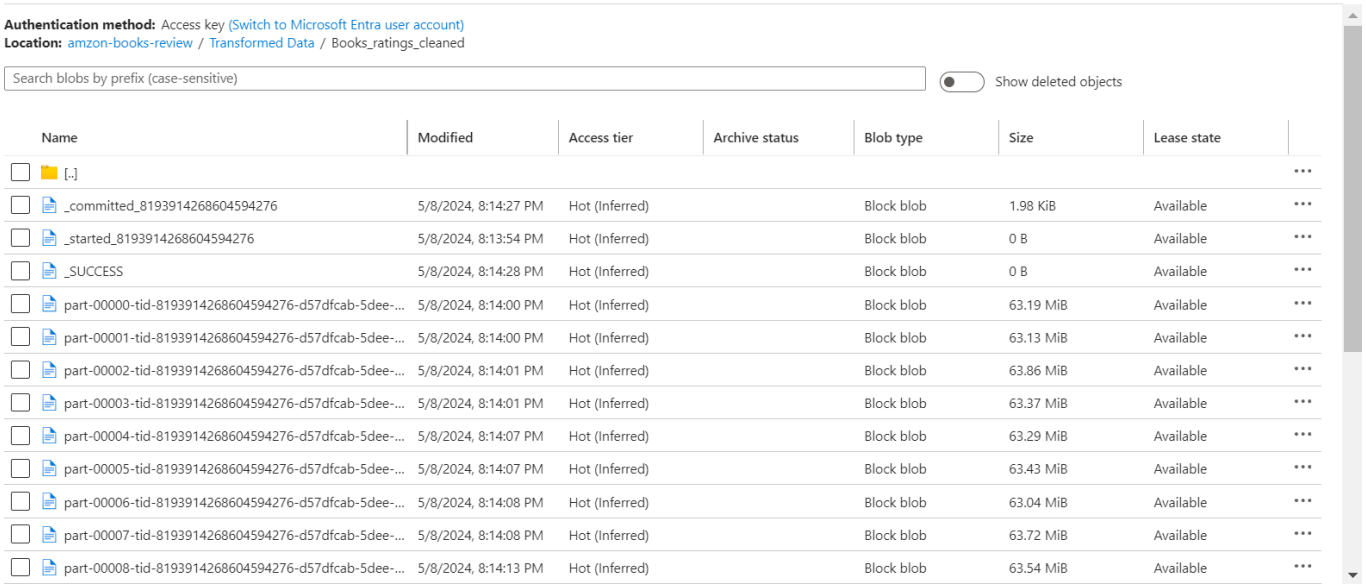
```
# Write the Transformed/Cleaned data into Azure Data lake

Book_details_cleaned_final.write.option("header",True).csv("/mnt/amazonbooksreviews/Transformed Data/Books_Details_cleaned.csv")
```



```
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]
```

+ Code + Text

```
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: amzon-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

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

OK

```
%sql
SELECT * FROM Books_Details_temp
```

(1) Spark Jobs

display_query_3 (id: 786a1a94-c45c-4995-ad10-bbc70ae85c3a) *Last updated: 2 days ago*

Dashboard | Raw Data

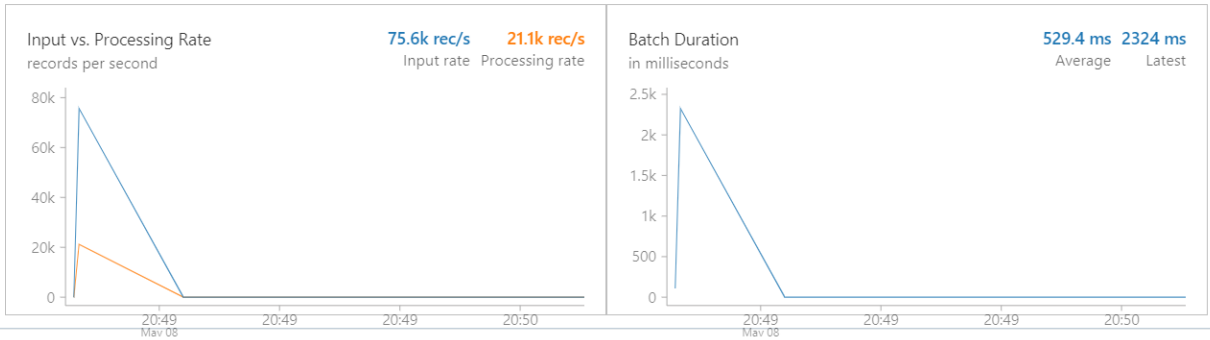


Table | New result table: ON

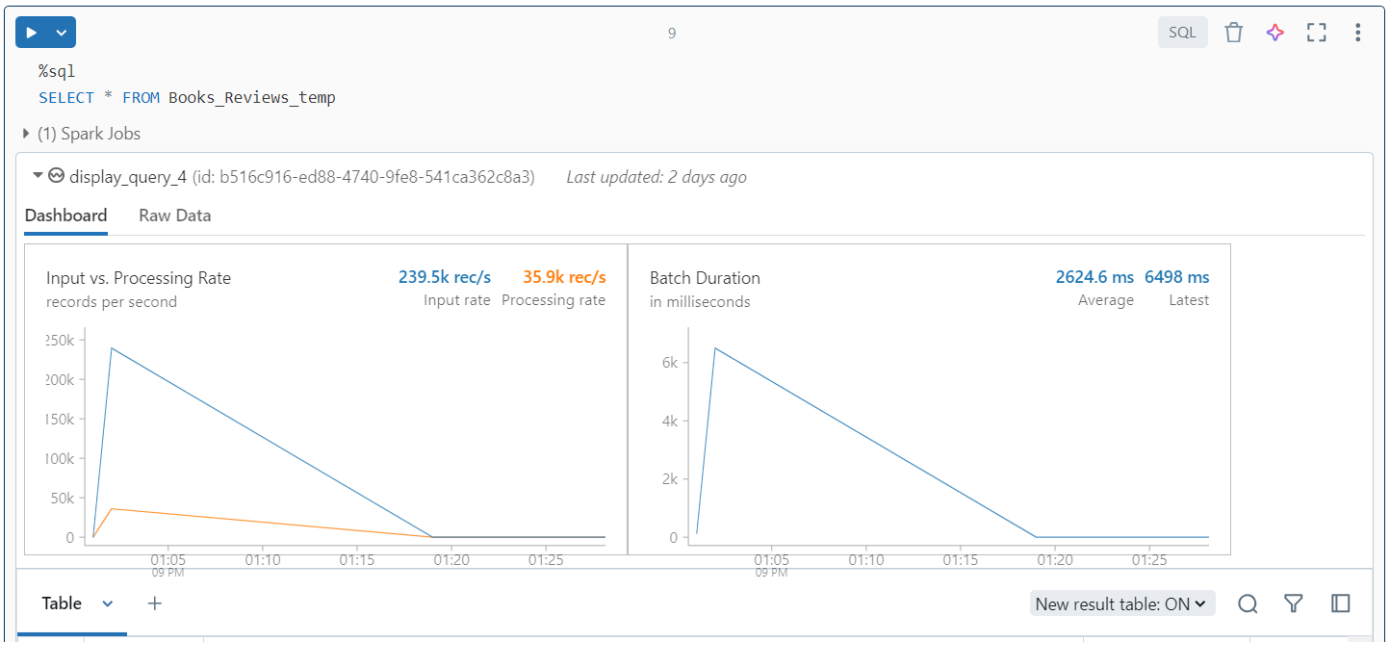
3. Reading **Books_Reviews** table as Stream

```
# 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"))
```

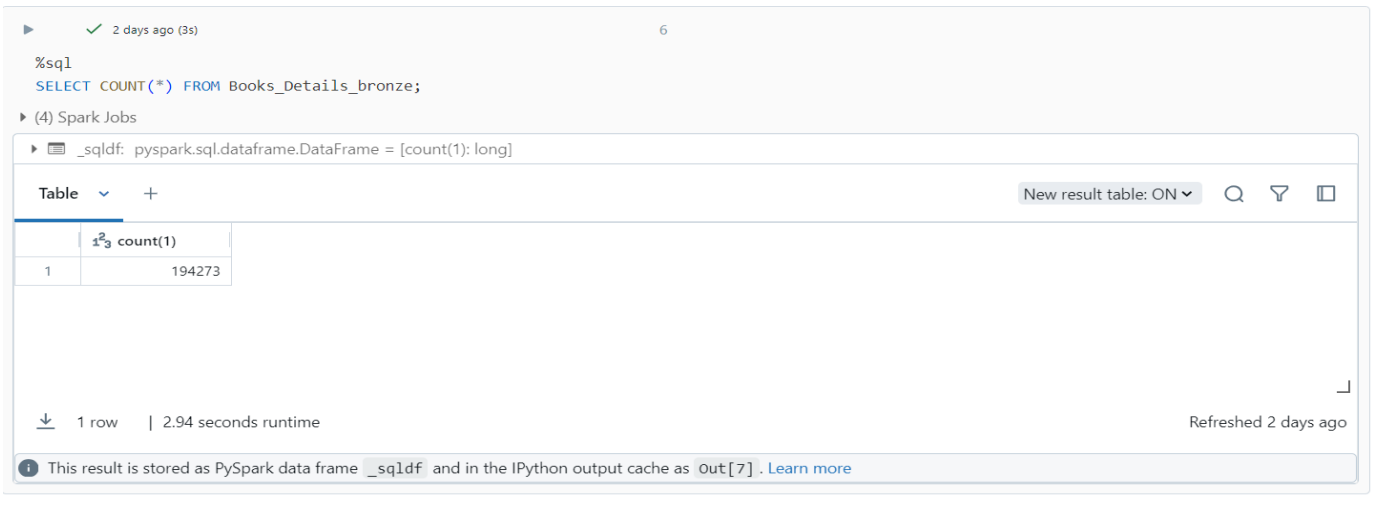
4. Creating Temp view

```
%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.



```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>
```

2 days ago (1s)

```sql
%sql
SELECT count(*) FROM Books_reviews_bronze;
```

(3) Spark Jobs

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

Table

	count(1)
1	2981936

1 row | 0.86 seconds runtime

Refreshed 2 days ago

This result is stored as PySpark data frame `_sqldf` and in the IPython output cache as `Out[24]`. Learn more

2 days ago (1s)

```sql
%sql
DESCRIBE EXTENDED Books_reviews_bronze;
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [col_name: string, data_type: string ... 1 more field]

Table

	col_name	data_type	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

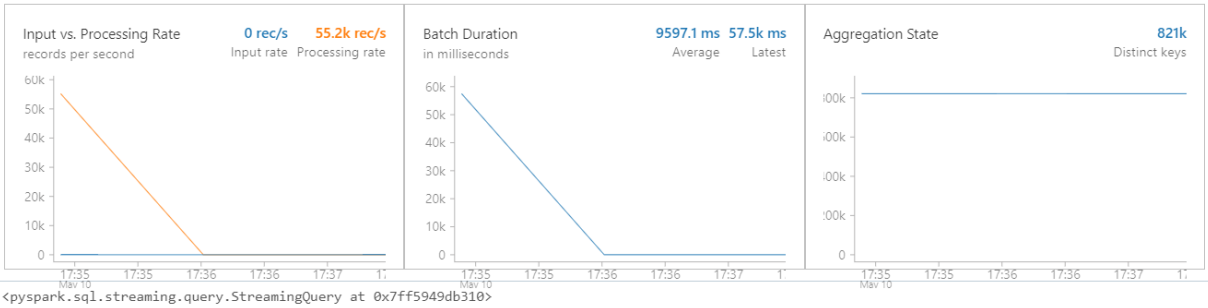
17 Python

```
# 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



<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]

Table

	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

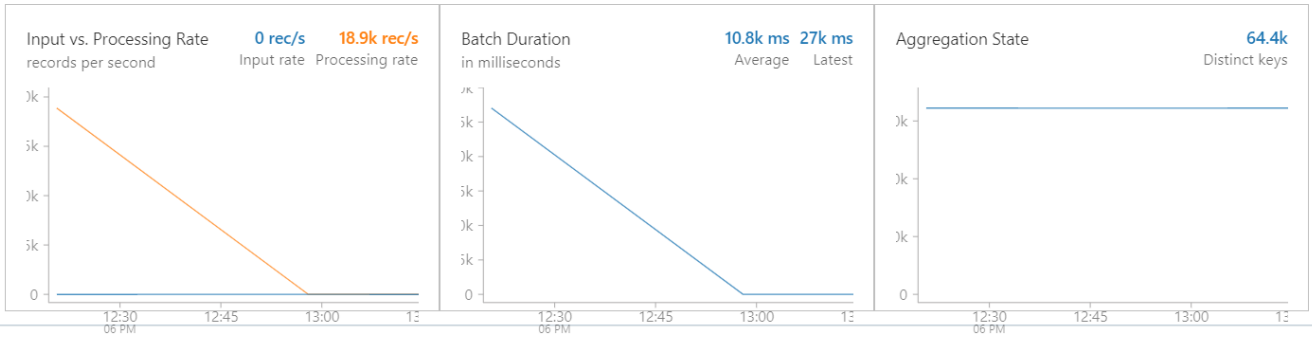
23

```
%sql
SELECT * FROM Amazon_Books_Reviews_Gold_Temp
```

(1) Spark Jobs

display_query_6 (id: 314bbf75-ab4b-4677-807c-3e3cfcbad8c1) *Last updated: 18 minutes ago*

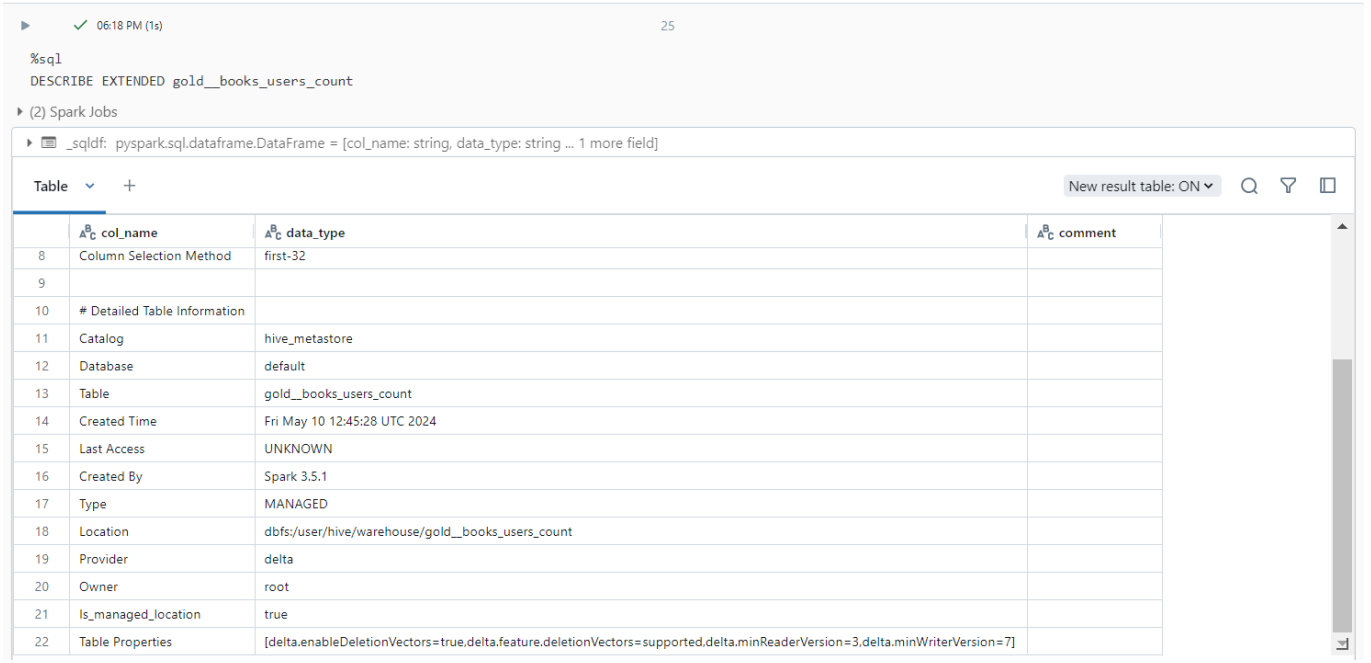
Dashboard Raw Data



Table

New result table: ON

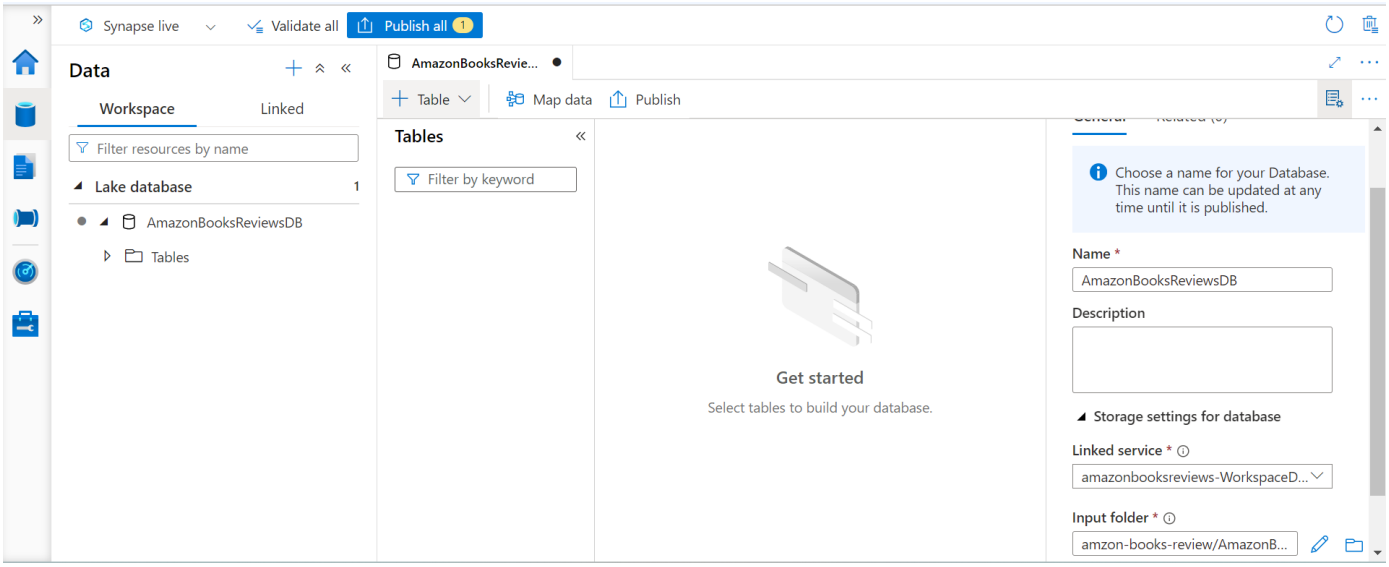
9. Writing Data into **Gold** Table

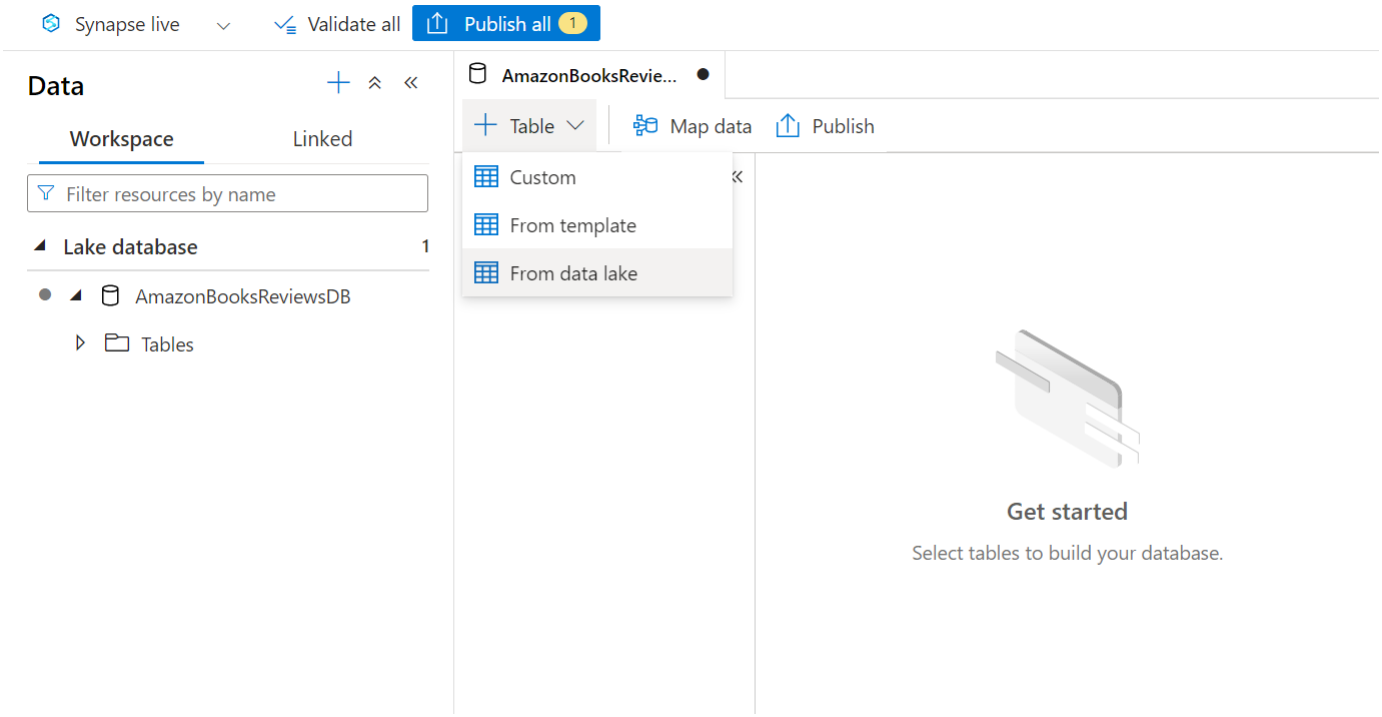


	col_name	data_type	comment
8	Column Selection Method	first-32	
9			
10	# Detailed Table Information		
11	Catalog	hive_metastore	
12	Database	default	
13	Table	gold__books_users_count	
14	Created Time	Fri May 10 12:45:28 UTC 2024	
15	Last Access	UNKNOWN	
16	Created By	Spark 3.5.1	
17	Type	MANAGED	
18	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]	

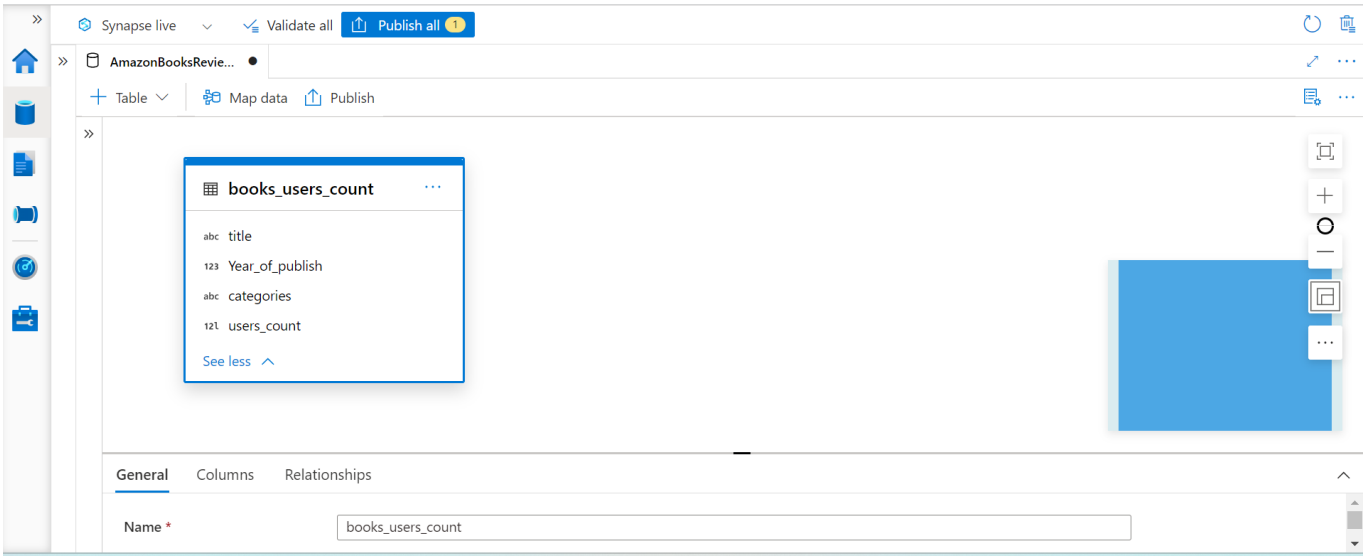
STEP 4: ANALYTICS AND VISUALIZATION

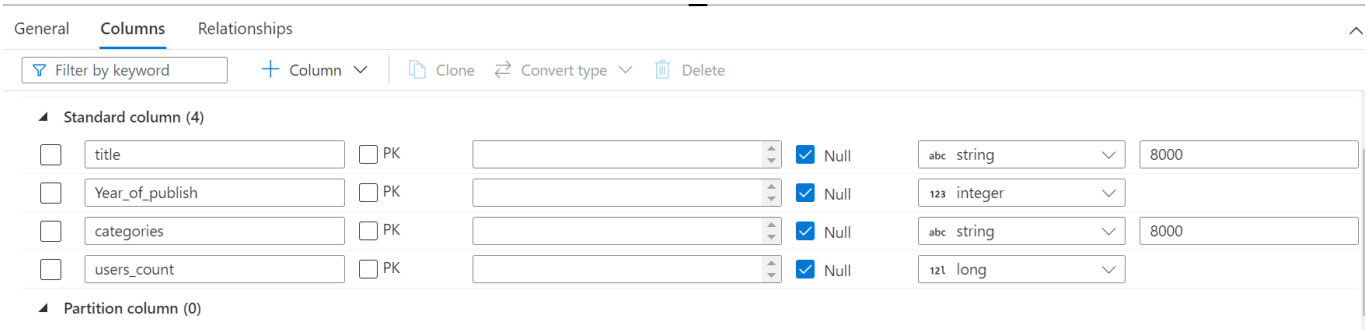
1. Set up Azure Synapse Analytics

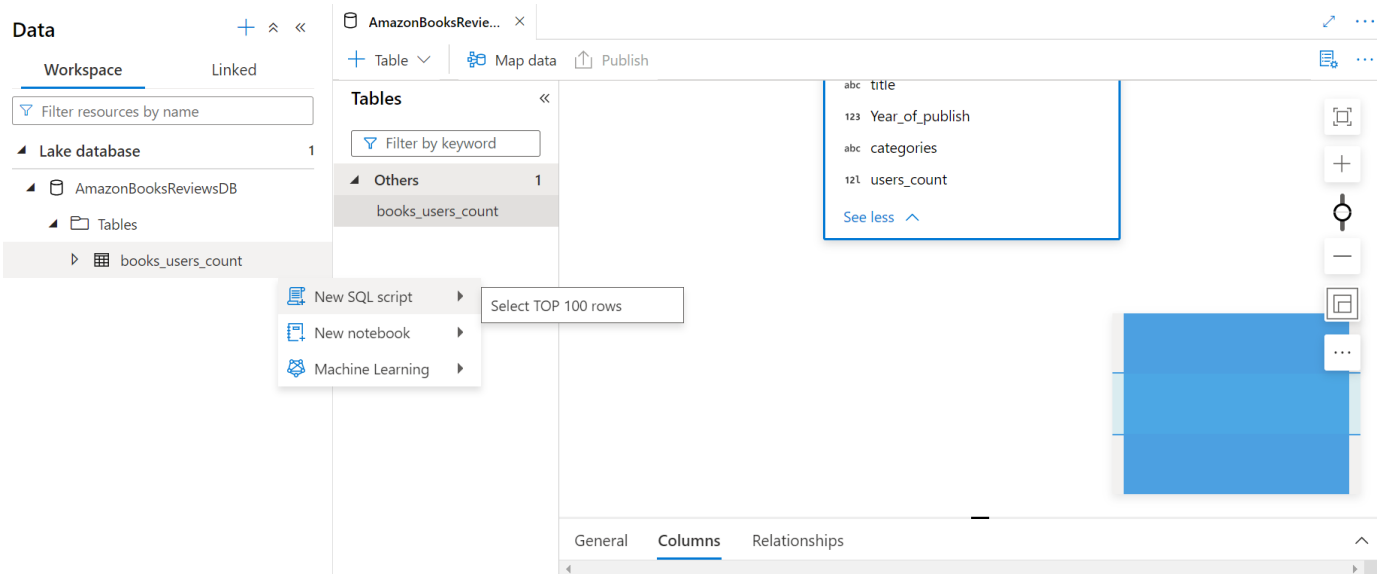




2. Tables Creation and Data loading



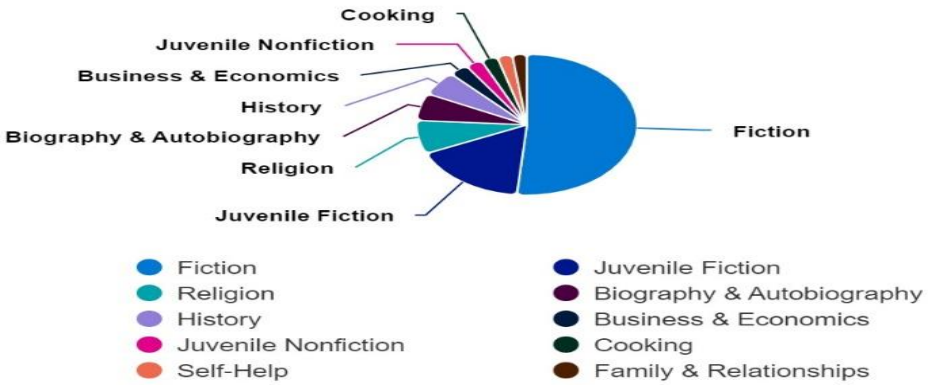


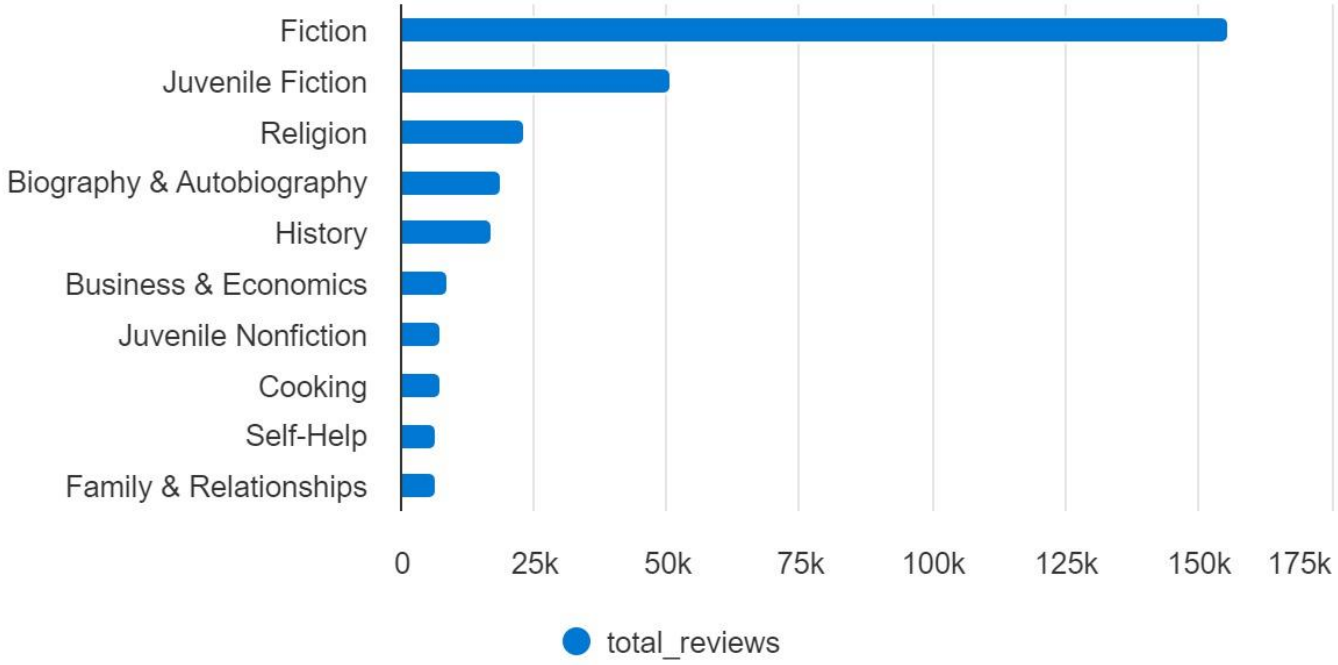


Visualization:

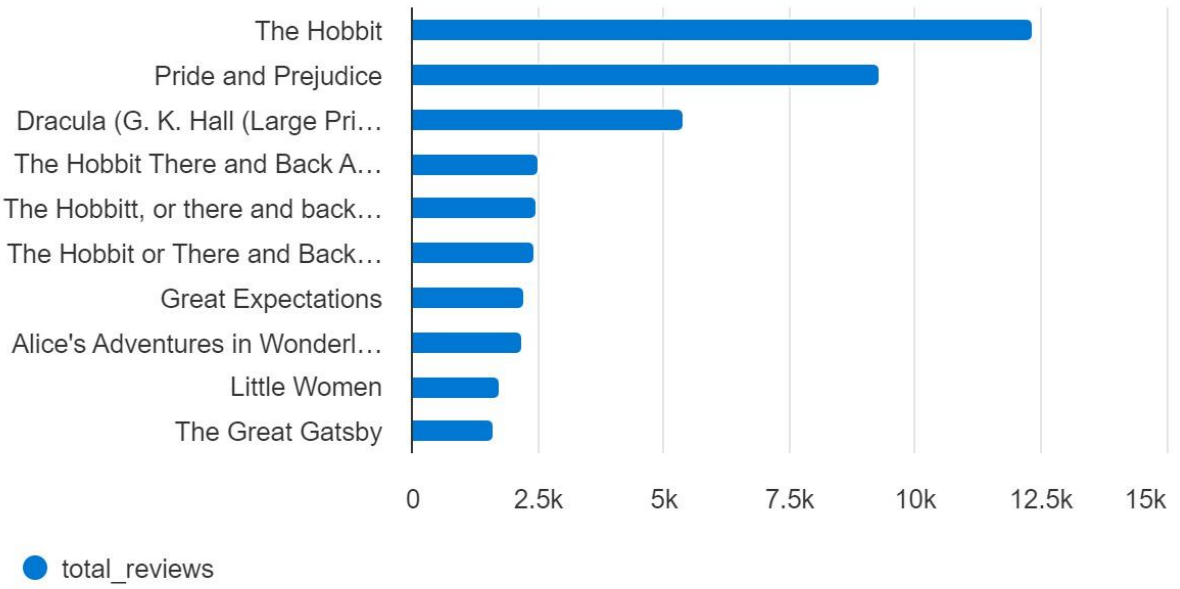
➔ Books categories with the greatest number of Reviews.

```sql
-- TOP 10 categories of books that got most number of reviews

SELECT
    categories,
    sum(users_count) as total_reviews
FROM books_users_count
GROUP BY categories
ORDER BY total_reviews DESC
OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY;

```

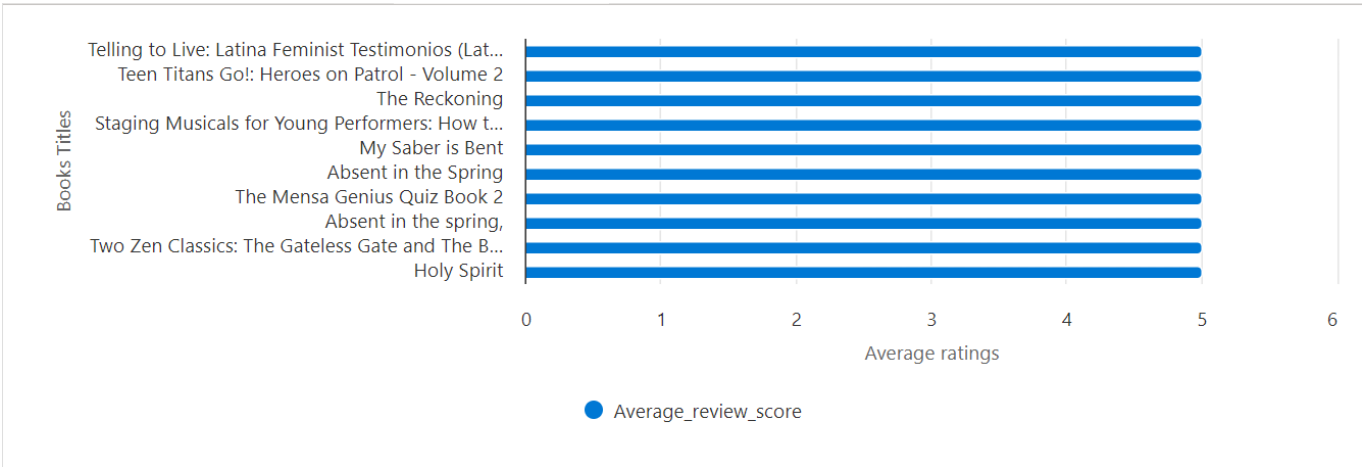




➔ Books with Most Reviews



➔ Top 10 Highest Rated Books



➔ Users with the greatest number of reviews.