BOOTCAMP PROJECT-2

AMAZON PRIME VIDEOS ANALYSIS AND SCD TYPE-1 IMPLEMENTATION

DATA ENGINEERING C-29

Introduction

The growth of online streaming platforms such as Amazon Prime has led to an explosion of content in the form of movies and TV shows. To extract meaningful insights from this vast amount of data, it is crucial to design an efficient, scalable, and secure data pipeline that can handle ingestion, transformation, and analytical workloads.

This project focuses on building an **end-to-end data pipeline** using **Volumes** and **Databricks** to process and analyze the **Amazon Prime Movies and TV Shows dataset** (Kaggle Source). The pipeline is implemented following the **medallion architecture (Bronze–Silver–Gold layers)** to ensure a structured and incremental approach to data management.

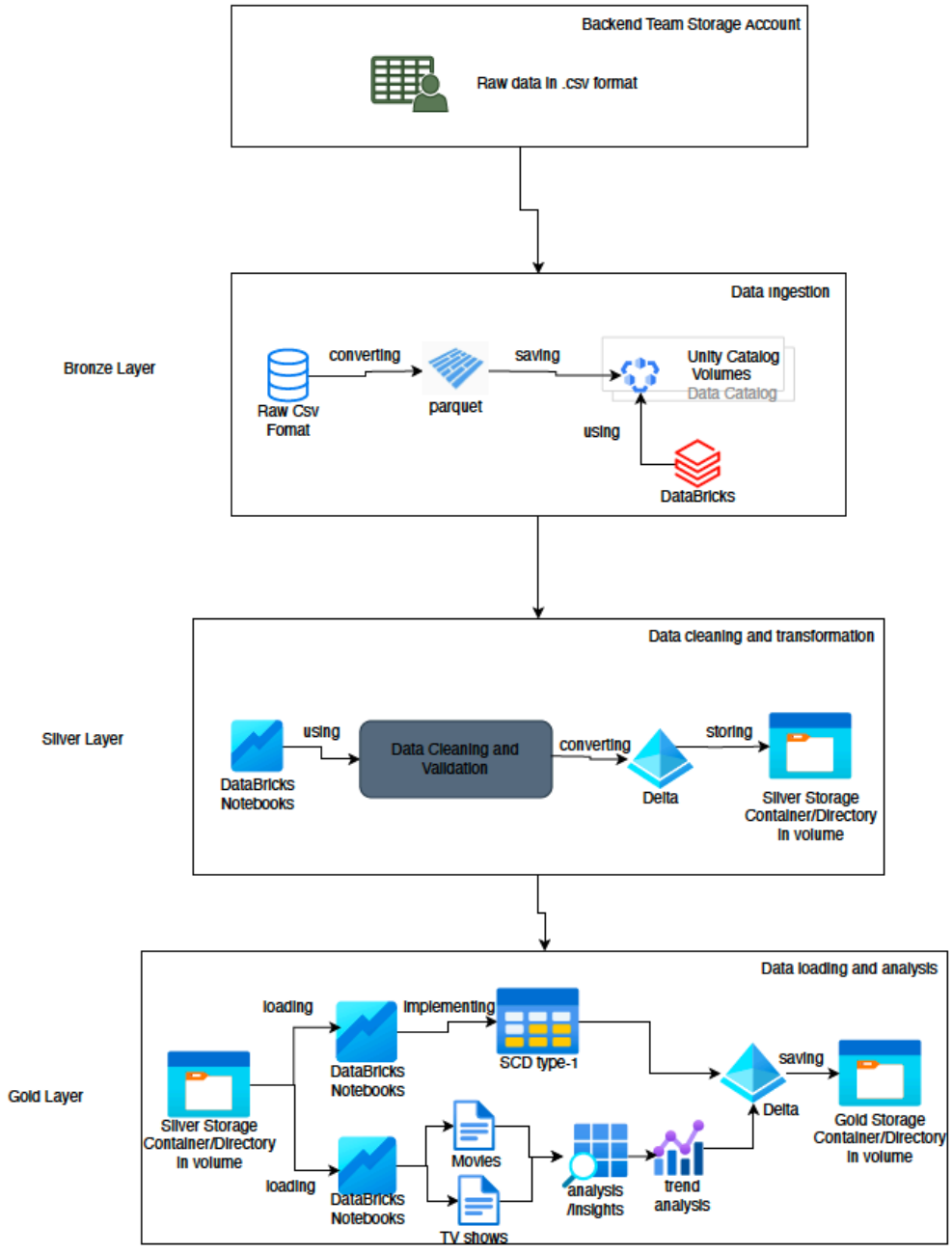
- In the **Bronze layer**, raw data is ingested directly from backend storage into Volumes, preserving the source format for audit and traceability.

- The **Silver layer** applies data cleaning and standardization processes, including schema enforcement, deduplication, handling null values, type casting, and feature engineering (e.g., extracting duration in minutes, creating adult-content flags). The dataset is further split into **movies_silver** and **tv_shows_silver** tables for specialized analysis.

- The **Gold layer** aggregates curated insights through Delta views and tables, supporting advanced analytical queries such as counts by type, top durations, rating-based statistics, country-level breakdowns, and temporal trends.

By leveraging **Delta format** and **Databricks notebooks**, the pipeline ensures both reliability and efficiency, while enabling downstream reporting and business intelligence integrations.

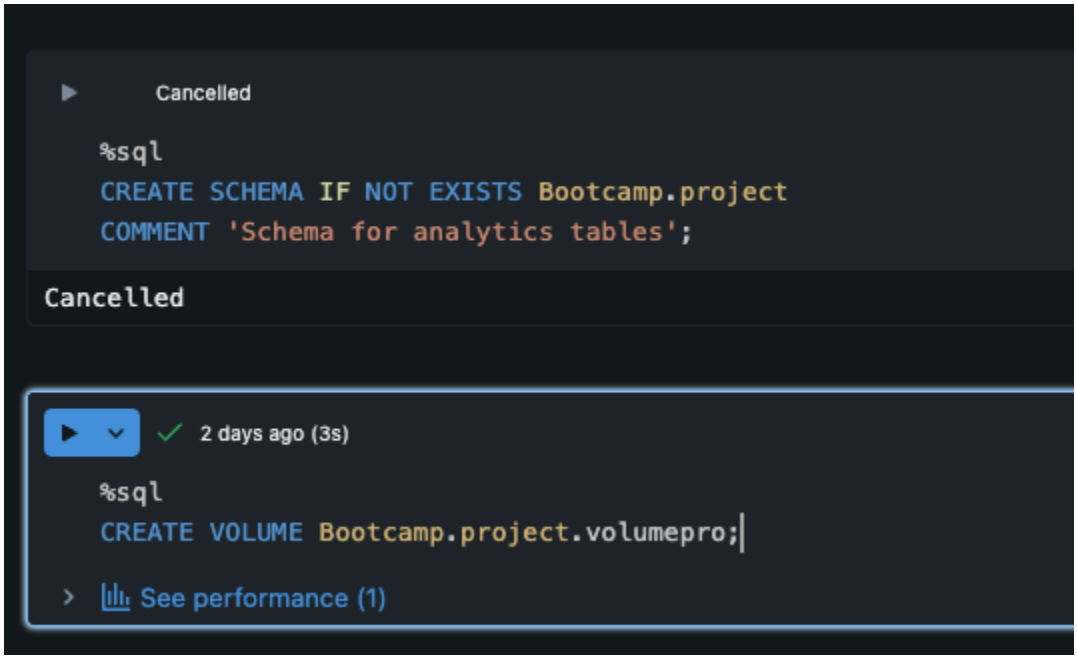
Architecture Diagram



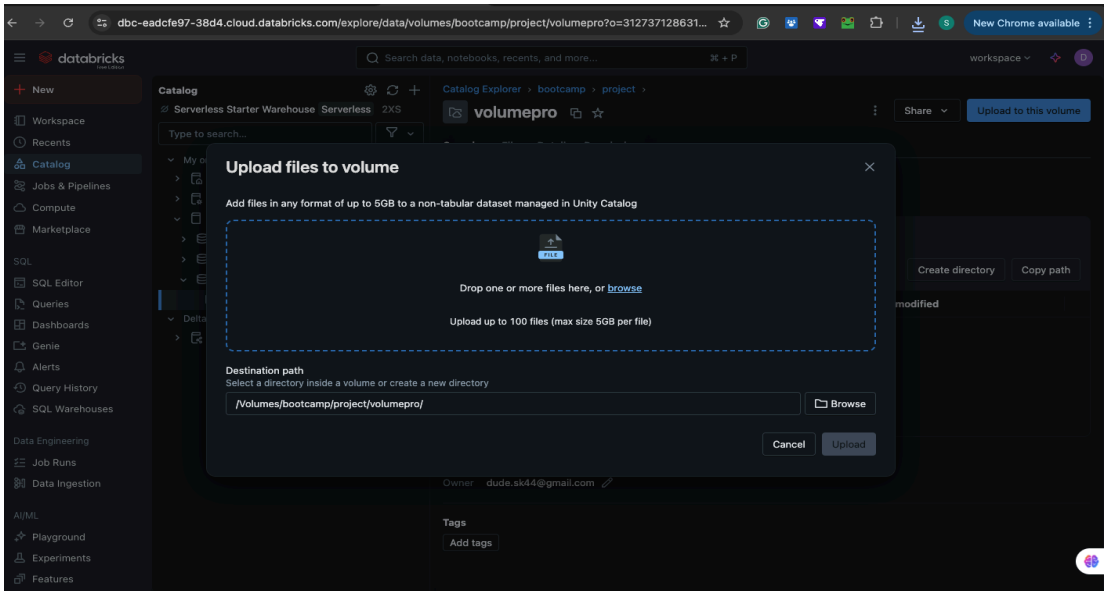
Creating Volumes in Databricks

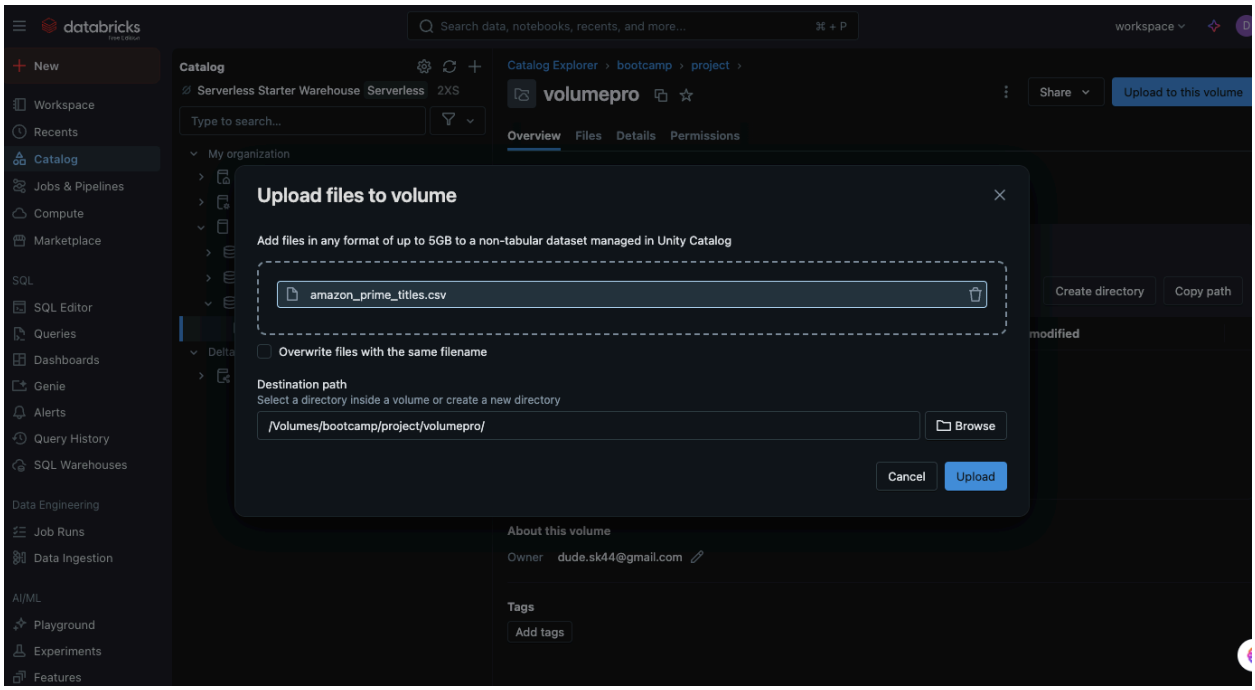
First create Catalog.

Create Schema by running this command in the Databricks notebook.

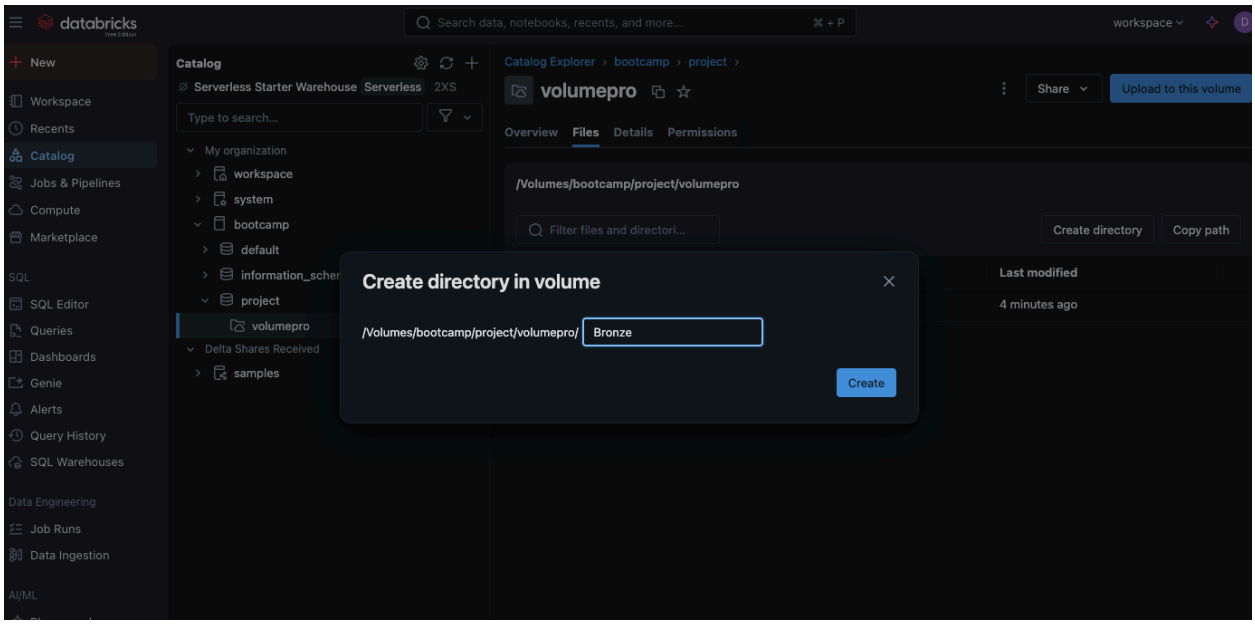


Upload your raw data to the volume.

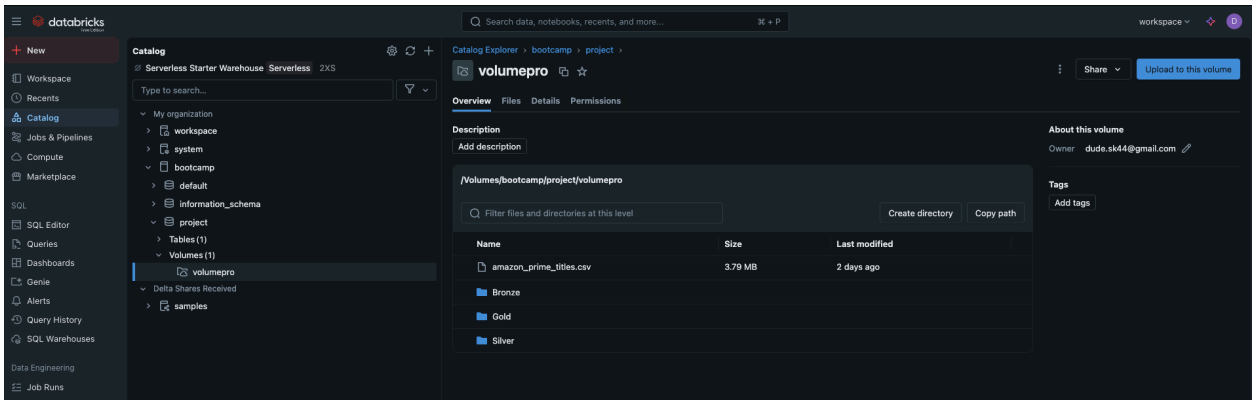




Create 3 directories inside the volume since we are implementing the Medallion architecture.

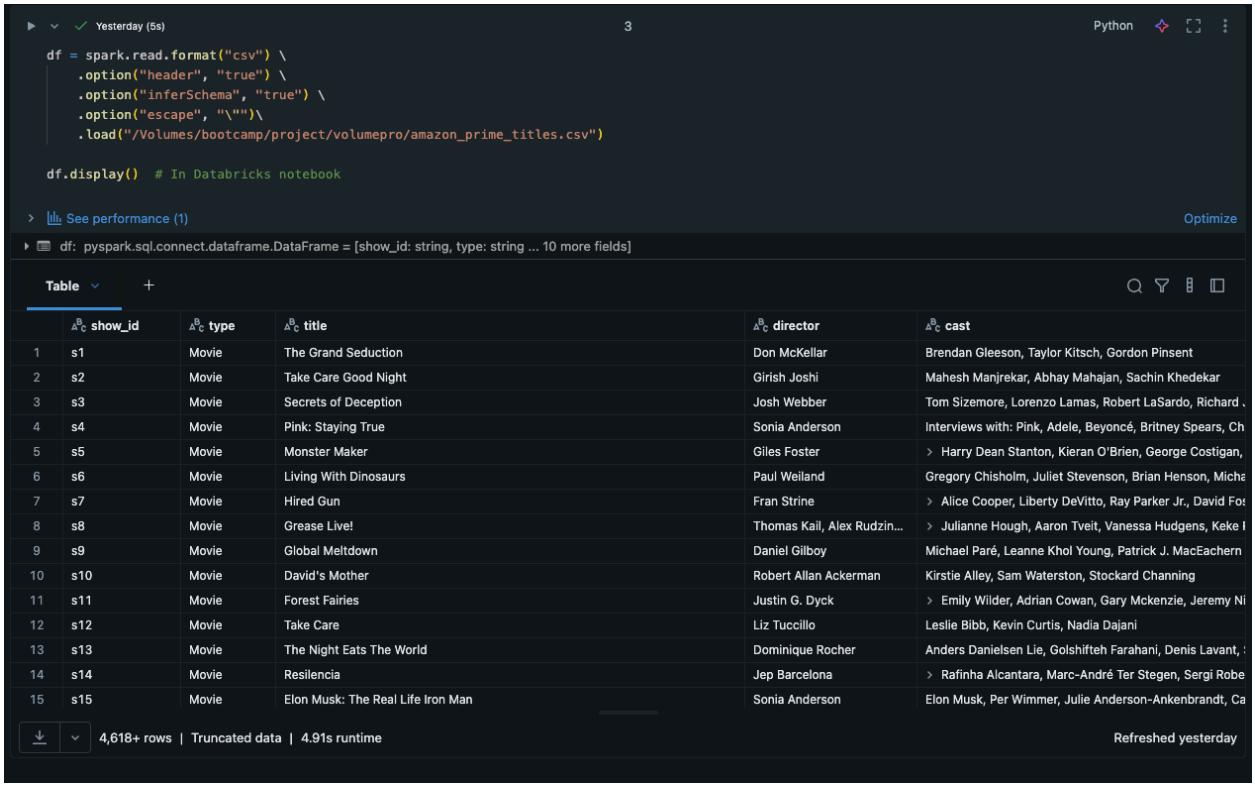


3 Directories created.



Data Ingestion

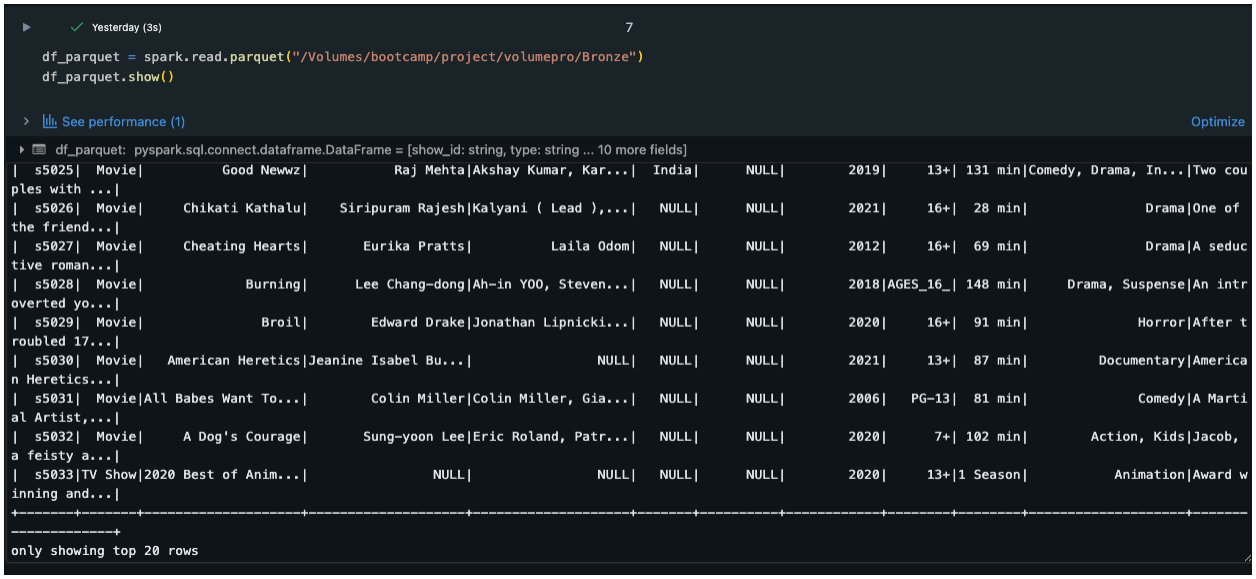
Loading data in the dataframe from the file that is saved in CSV file in the volume.



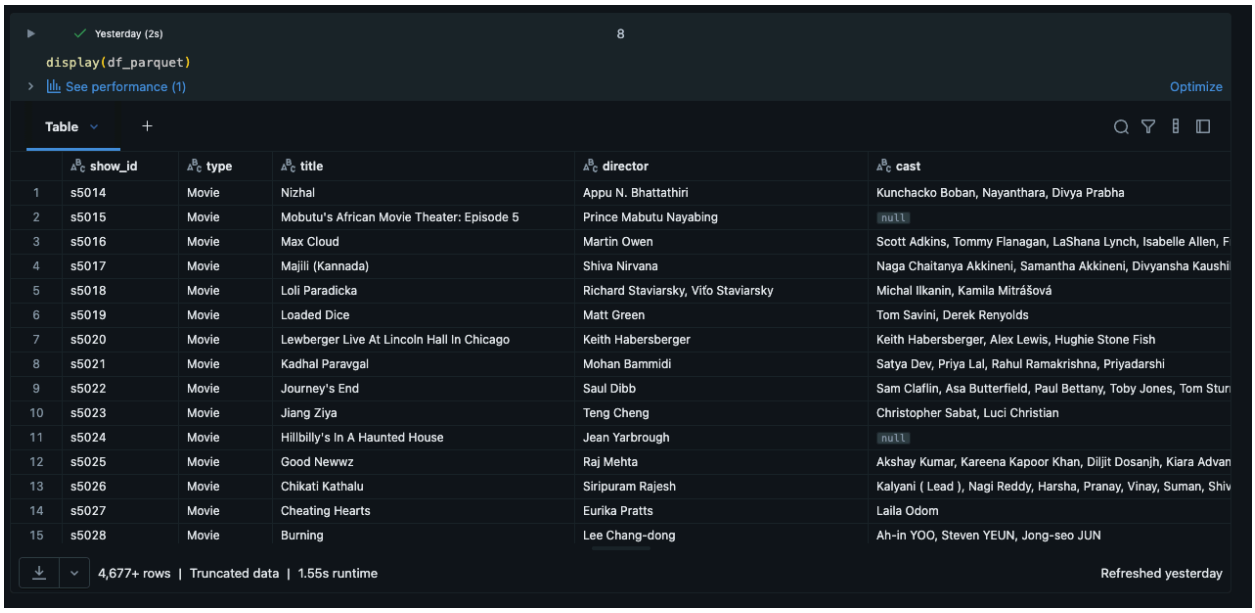
Saving the dataframe by converting it into the parquet format in the bronze layer.

Data Cleaning and Transformation

Loading the parquet file from the bronze layer.



Displaying the parquet file in table format.



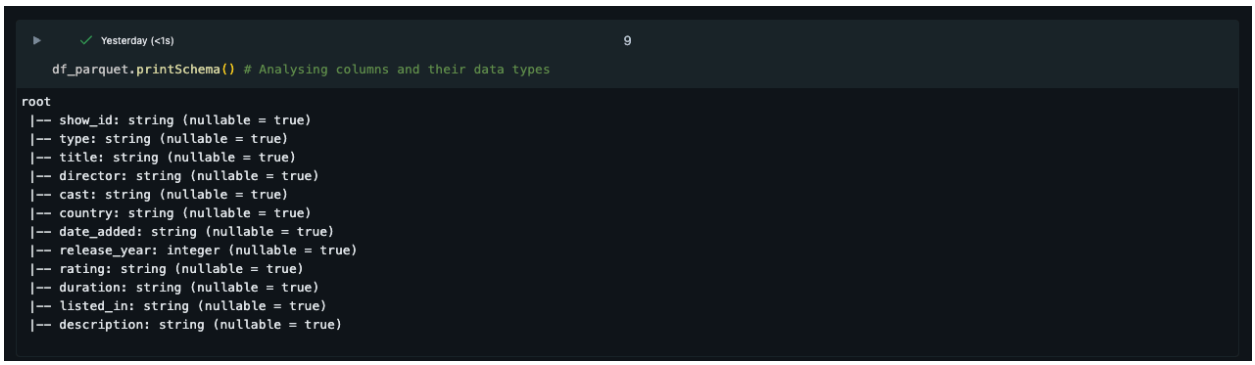
```
Yesterday (2s)                                                    8
display(df_parquet)
See performance (1)                                               Optimize
```

Table +

	show_id	type	title	director	cast
1	s5014	Movie	Nizhal	Appu N. Bhattathiri	Kunchacko Boban, Nayanthara, Divya Prabha
2	s5015	Movie	Mobutu's African Movie Theater: Episode 5	Prince Mabutu Nayabing	null
3	s5016	Movie	Max Cloud	Martin Owen	Scott Adkins, Tommy Flanagan, LaShana Lynch, Isabelle Allen, F
4	s5017	Movie	Majili (Kannada)	Shiva Nirvana	Naga Chaitanya Akkineni, Samantha Akkineni, Divyansha Kaushi
5	s5018	Movie	Loli Paradicka	Richard Staviarsky, Vito Staviarsky	Michal Ilkanin, Kamila Mitrášová
6	s5019	Movie	Loaded Dice	Matt Green	Tom Savini, Derek Renyolds
7	s5020	Movie	Lewberger Live At Lincoln Hall In Chicago	Keith Habersberger	Keith Habersberger, Alex Lewis, Hughie Stone Fish
8	s5021	Movie	Kadhal Paravgal	Mohan Bammidi	Satya Dev, Priya Lal, Rahul Ramakrishna, Priyadarshi
9	s5022	Movie	Journey's End	Saul Dibb	Sam Claflin, Asa Butterfield, Paul Bettany, Toby Jones, Tom Stur
10	s5023	Movie	Jiang Ziya	Teng Cheng	Christopher Sabat, Luci Christian
11	s5024	Movie	Hillbilly's In A Haunted House	Jean Yarbrough	null
12	s5025	Movie	Good Newwz	Raj Mehta	Akshay Kumar, Kareena Kapoor Khan, Diljit Dosanjh, Kiara Advan
13	s5026	Movie	Chikati Kathalu	Siripuram Rajesh	Kalyani (Lead), Nagi Reddy, Harsha, Pranay, Vinay, Suman, Shiv
14	s5027	Movie	Cheating Hearts	Eurika Pratts	Laila Odom
15	s5028	Movie	Burning	Lee Chang-dong	Ah-in YOO, Steven YEUN, Jong-seo JUN

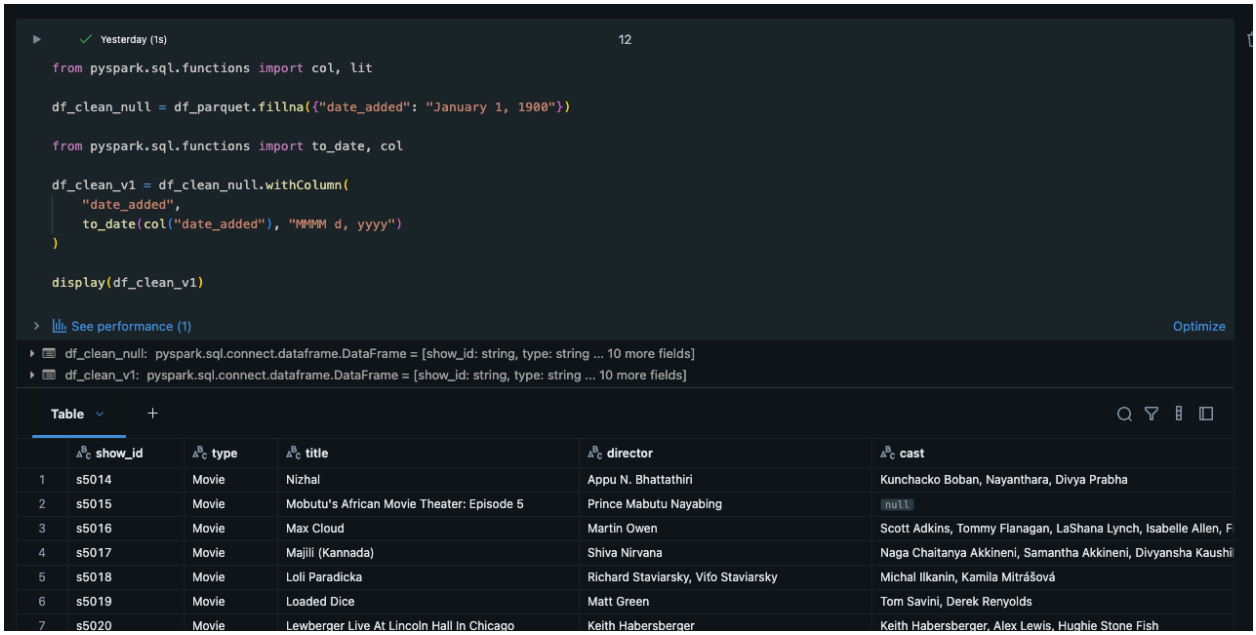
4,677+ rows | Truncated data | 1.55s runtime Refreshed yesterday

Analysing columns and its data types.



```
Yesterday (<1s)                                                   9
df_parquet.printSchema() # Analysing columns and their data types

root
 |-- show_id: string (nullable = true)
 |-- type: string (nullable = true)
 |-- title: string (nullable = true)
 |-- director: string (nullable = true)
 |-- cast: string (nullable = true)
 |-- country: string (nullable = true)
 |-- date_added: string (nullable = true)
 |-- release_year: integer (nullable = true)
 |-- rating: string (nullable = true)
 |-- duration: string (nullable = true)
 |-- listed_in: string (nullable = true)
 |-- description: string (nullable = true)
```

Replacing all the null values in the date added column with the default date. Converting datatype of date added column to date.

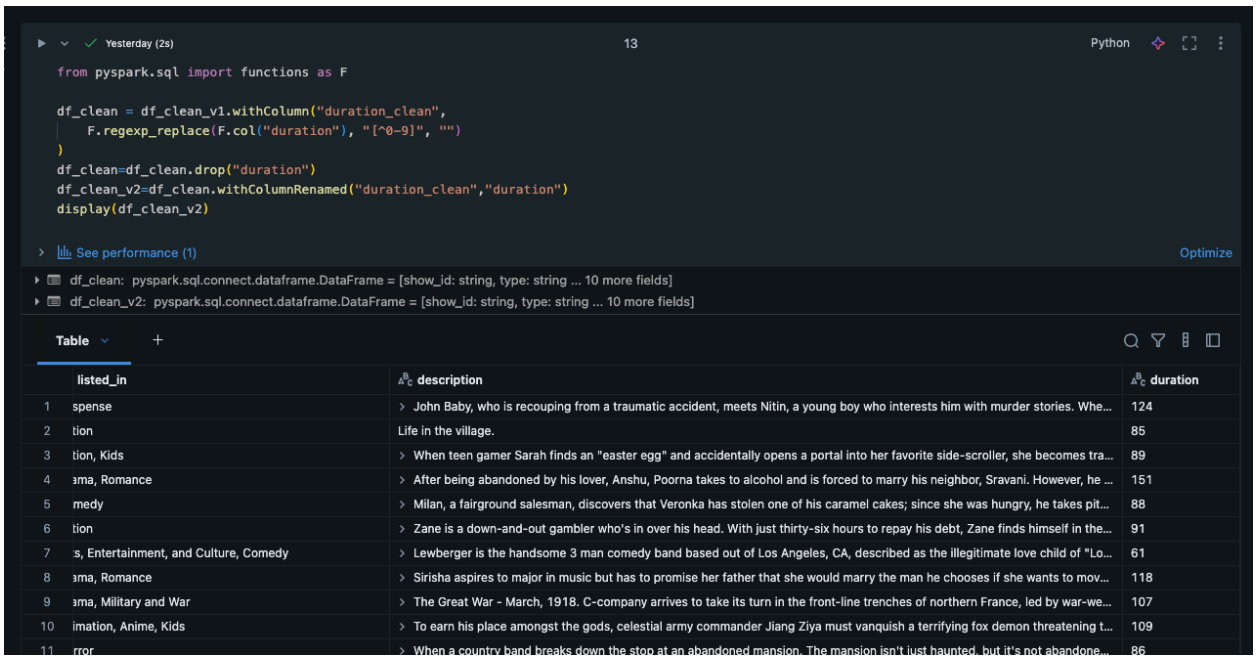


```
from pyspark.sql.functions import col, lit

df_clean_null = df_parquet.fillna({"date_added": "January 1, 1900"})

from pyspark.sql.functions import to_date, col

df_clean_v1 = df_clean_null.withColumn(
    "date_added",
    to_date(col("date_added"), "MMMM d, yyyy")
)

display(df_clean_v1)
```

df_clean_null: pyspark.sql.connect.dataframe.DataFrame = [show_id: string, type: string ... 10 more fields]
df_clean_v1: pyspark.sql.connect.dataframe.DataFrame = [show_id: string, type: string ... 10 more fields]

	show_id	type	title	director	cast
1	s5014	Movie	Nizhal	Appu N. Bhattathiri	Kunchacko Boban, Nayanthara, Divya Prabha
2	s5015	Movie	Mobutu's African Movie Theater: Episode 5	Prince Mabutu Nayabing	null
3	s5016	Movie	Max Cloud	Martin Owen	Scott Adkins, Tommy Flanagan, LaShana Lynch, Isabelle Allen, F
4	s5017	Movie	Majili (Kannada)	Shiva Nirvana	Naga Chaitanya Akkineni, Samantha Akkineni, Divyansha Kaushi
5	s5018	Movie	Loli Paradicka	Richard Staviarsky, Vito Staviarsky	Michal Ilkanin, Kamila Mitrášová
6	s5019	Movie	Loaded Dice	Matt Green	Tom Savini, Derek Renyolds
7	s5020	Movie	Lewberger Live At Lincoln Hall In Chicago	Keith Habersberger	Keith Habersberger, Alex Lewis, Hughie Stone Fish

Cleaning the duration column, removing characters from the duration column.



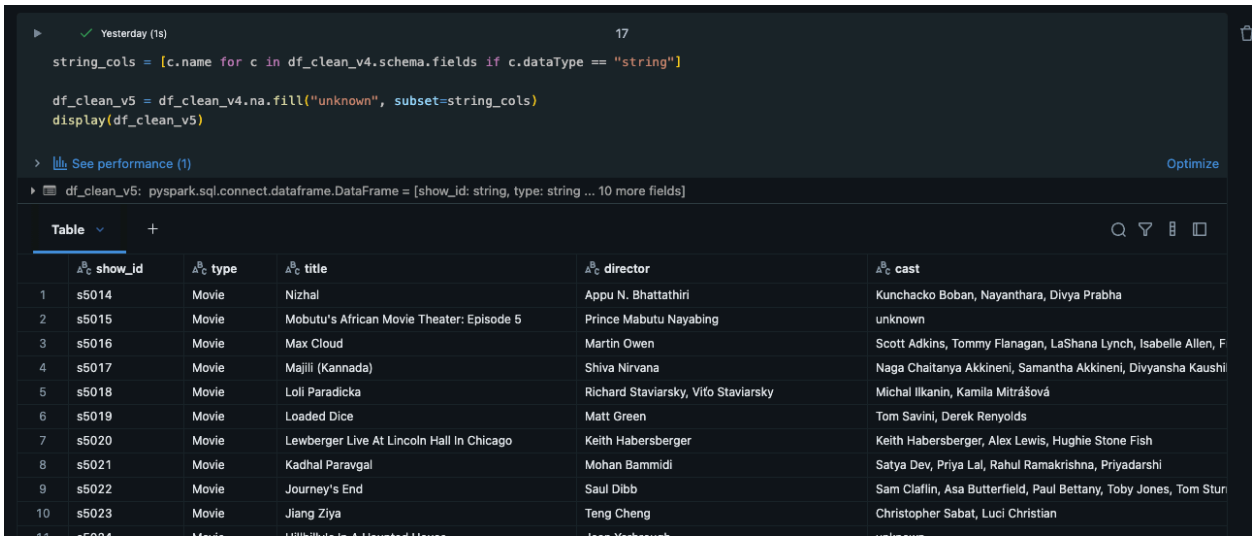
```
from pyspark.sql import functions as F

df_clean = df_clean_v1.withColumn("duration_clean",
    F.regexp_replace(F.col("duration"), "[^0-9]", "")
)
df_clean=df_clean.drop("duration")
df_clean_v2=df_clean.withColumnRenamed("duration_clean","duration")
display(df_clean_v2)
```

df_clean: pyspark.sql.connect.dataframe.DataFrame = [show_id: string, type: string ... 10 more fields]
df_clean_v2: pyspark.sql.connect.dataframe.DataFrame = [show_id: string, type: string ... 10 more fields]

	listed_in	description	duration
1	spense	John Baby, who is recouping from a traumatic accident, meets Nitin, a young boy who interests him with murder stories. Whe...	124
2	tion	Life in the village.	85
3	tion, Kids	When teen gamer Sarah finds an "easter egg" and accidentally opens a portal into her favorite side-scroller, she becomes tra...	89
4	ama, Romance	After being abandoned by his lover, Anshu, Poorna takes to alcohol and is forced to marry his neighbor, Sravani. However, he ...	151
5	medy	Milan, a fairground salesman, discovers that Veronka has stolen one of his caramel cakes; since she was hungry, he takes pit...	88
6	tion	Zane is a down-and-out gambler who's in over his head. With just thirty-six hours to repay his debt, Zane finds himself in the...	91
7	s, Entertainment, and Culture, Comedy	Lewberger is the handsome 3 man comedy band based out of Los Angeles, CA, described as the illegitimate love child of "Lo...	61
8	ama, Romance	Sirisha aspires to major in music but has to promise her father that she would marry the man he chooses if she wants to mov...	118
9	ama, Military and War	The Great War - March, 1918. C-company arrives to take its turn in the front-line trenches of northern France, led by war-we...	107
10	imation, Anime, Kids	To earn his place amongst the gods, celestial army commander Jiang Ziya must vanquish a terrifying fox demon threatening t...	109
11	rror	When a country band breaks down the stop at an abandoned mansion. The mansion isn't just haunted, but it's not abandone...	86

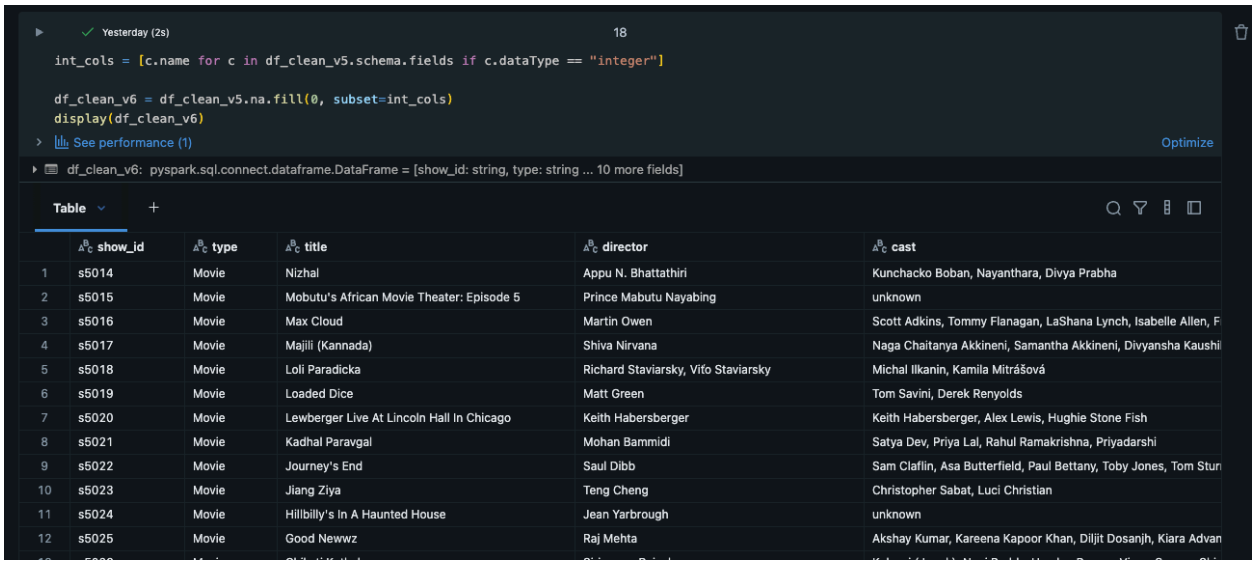
Casting the duration and release year column

Filling all the null values with ‘unknown’ for the columns whose datatype is string.



	show_id	type	title	director	cast
1	s5014	Movie	Nizhal	Appu N. Bhattathiri	Kunchacko Boban, Nayanthara, Divya Prabha
2	s5015	Movie	Mobutu's African Movie Theater: Episode 5	Prince Mabutu Nayabing	unknown
3	s5016	Movie	Max Cloud	Martin Owen	Scott Adkins, Tommy Flanagan, LaShana Lynch, Isabelle Allen, F
4	s5017	Movie	Majili (Kannada)	Shiva Nirvana	Naga Chaitanya Akkineni, Samantha Akkineni, Divyansha Kaushil
5	s5018	Movie	Loli Paradicka	Richard Staviarsky, Vito Staviarsky	Michal Ilkanin, Kamila Mitrášová
6	s5019	Movie	Loaded Dice	Matt Green	Tom Savini, Derek Renyolds
7	s5020	Movie	Lewberger Live At Lincoln Hall In Chicago	Keith Habersberger	Keith Habersberger, Alex Lewis, Hughie Stone Fish
8	s5021	Movie	Kadhal Paravgal	Mohan Bammidi	Satya Dev, Priya Lal, Rahul Ramakrishna, Priyadarshi
9	s5022	Movie	Journey's End	Saul Dibb	Sam Claflin, Asa Butterfield, Paul Bettany, Toby Jones, Tom Stur
10	s5023	Movie	Jiang Ziya	Teng Cheng	Christopher Sabat, Luci Christian

Filling all the null values with ‘0’ whose column datatype in integer.



```
Yesterday (2s)                                                    18
int_cols = [c.name for c in df_clean_v5.schema.fields if c.dataType == "integer"]

df_clean_v6 = df_clean_v5.na.fill(0, subset=int_cols)
display(df_clean_v6)
```

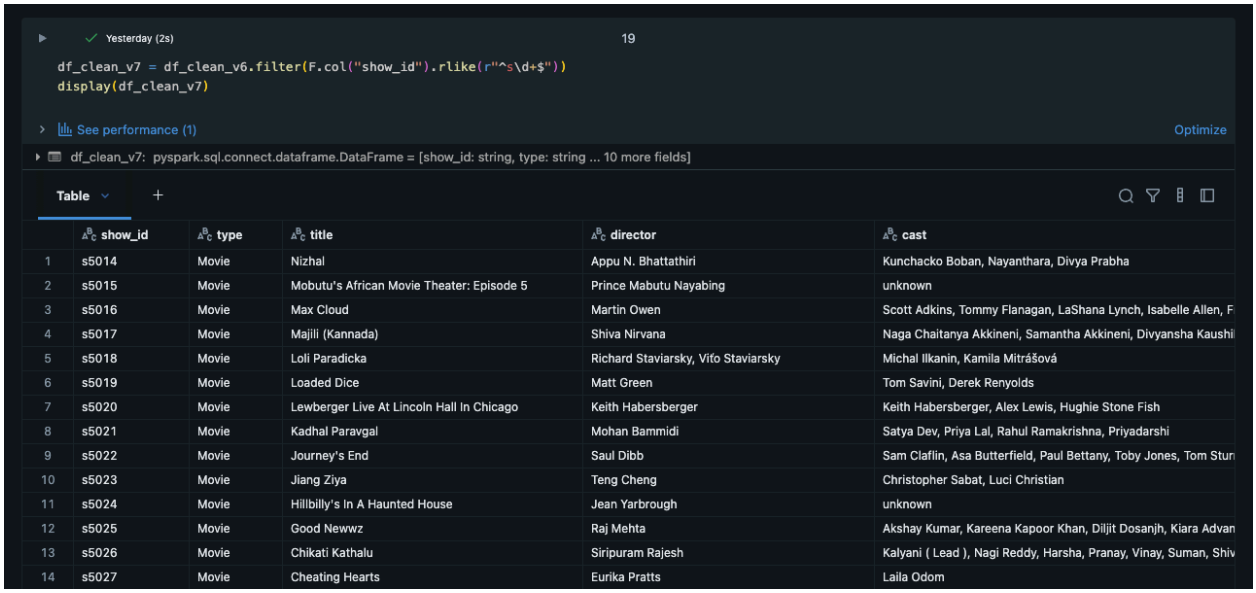
See performance (1) Optimize

df_clean_v6: pyspark.sql.connect.dataframe.DataFrame = [show_id: string, type: string ... 10 more fields]

Table +

	show_id	type	title	director	cast
1	s5014	Movie	Nizhal	Appu N. Bhattathiri	Kunchacko Boban, Nayanthara, Divya Prabha
2	s5015	Movie	Mobutu's African Movie Theater: Episode 5	Prince Mabutu Nayabing	unknown
3	s5016	Movie	Max Cloud	Martin Owen	Scott Adkins, Tommy Flanagan, LaShana Lynch, Isabelle Allen, F
4	s5017	Movie	Majili (Kannada)	Shiva Nirvana	Naga Chaitanya Akkineni, Samantha Akkineni, Divyansha Kaushi
5	s5018	Movie	Loli Paradicka	Richard Staviarsky, Vifo Staviarsky	Michal Ilkanin, Kamila Mitrášová
6	s5019	Movie	Loaded Dice	Matt Green	Tom Savini, Derek Renyolds
7	s5020	Movie	Lewberger Live At Lincoln Hall In Chicago	Keith Habersberger	Keith Habersberger, Alex Lewis, Hughie Stone Fish
8	s5021	Movie	Kadhal Paravgal	Mohan Bammidi	Satya Dev, Priya Lal, Rahul Ramakrishna, Priyadarshi
9	s5022	Movie	Journey's End	Saul Dibb	Sam Claflin, Asa Butterfield, Paul Bettany, Toby Jones, Tom Stur
10	s5023	Movie	Jiang Ziya	Teng Cheng	Christopher Sabat, Luci Christian
11	s5024	Movie	Hillbilly's In A Haunted House	Jean Yarbrough	unknown
12	s5025	Movie	Good Newwz	Raj Mehta	Akshay Kumar, Kareena Kapoor Khan, Diljit Dosanjh, Kiara Advan

Cleaning the showid column.



```
Yesterday (2s)                                                    19
df_clean_v7 = df_clean_v6.filter(F.col("show_id").rlike(r"^s\d+$"))
display(df_clean_v7)
```

See performance (1) Optimize

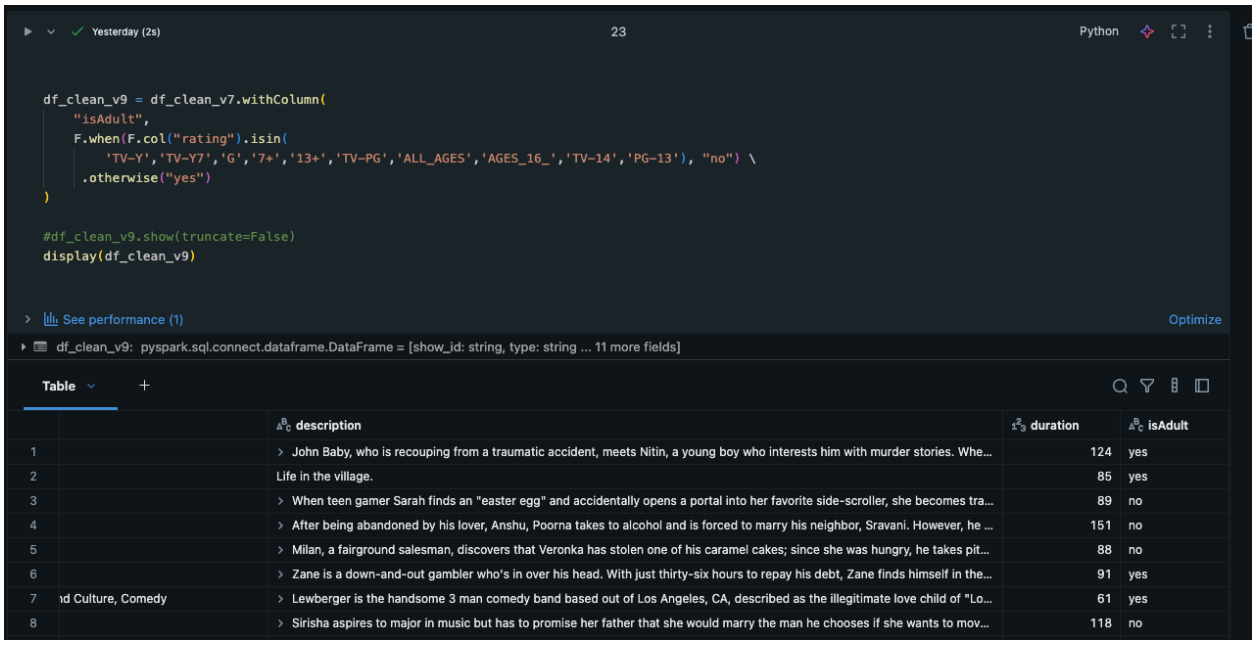
df_clean_v7: pyspark.sql.connect.dataframe.DataFrame = [show_id: string, type: string ... 10 more fields]

Table +

	show_id	type	title	director	cast
1	s5014	Movie	Nizhal	Appu N. Bhattathiri	Kunchacko Boban, Nayanthara, Divya Prabha
2	s5015	Movie	Mobutu's African Movie Theater: Episode 5	Prince Mabutu Nayabing	unknown
3	s5016	Movie	Max Cloud	Martin Owen	Scott Adkins, Tommy Flanagan, LaShana Lynch, Isabelle Allen, F
4	s5017	Movie	Majili (Kannada)	Shiva Nirvana	Naga Chaitanya Akkineni, Samantha Akkineni, Divyansha Kaushi
5	s5018	Movie	Loli Paradicka	Richard Staviarsky, Vifo Staviarsky	Michal Ilkanin, Kamila Mitrášová
6	s5019	Movie	Loaded Dice	Matt Green	Tom Savini, Derek Renyolds
7	s5020	Movie	Lewberger Live At Lincoln Hall In Chicago	Keith Habersberger	Keith Habersberger, Alex Lewis, Hughie Stone Fish
8	s5021	Movie	Kadhal Paravgal	Mohan Bammidi	Satya Dev, Priya Lal, Rahul Ramakrishna, Priyadarshi
9	s5022	Movie	Journey's End	Saul Dibb	Sam Claflin, Asa Butterfield, Paul Bettany, Toby Jones, Tom Stur
10	s5023	Movie	Jiang Ziya	Teng Cheng	Christopher Sabat, Luci Christian
11	s5024	Movie	Hillbilly's In A Haunted House	Jean Yarbrough	unknown
12	s5025	Movie	Good Newwz	Raj Mehta	Akshay Kumar, Kareena Kapoor Khan, Diljit Dosanjh, Kiara Advan
13	s5026	Movie	Chikati Kathalu	Siripuram Rajesh	Kalyani (Lead), Nagi Reddy, Harsha, Pranay, Vinay, Suman, Shiv
14	s5027	Movie	Cheating Hearts	Eurika Pratts	Laila Odom

Checking different types of rating present in the column.

Creating an new is adult and provide the values into it according to the condition.

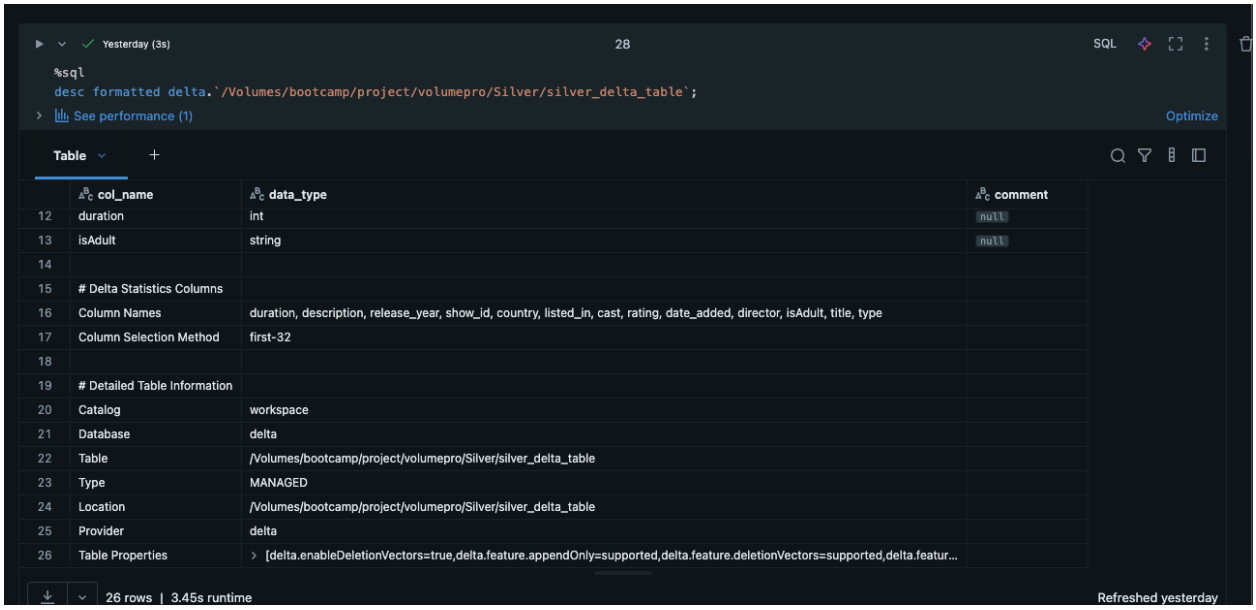


Save cleaned data in the silver layer in the delta format.

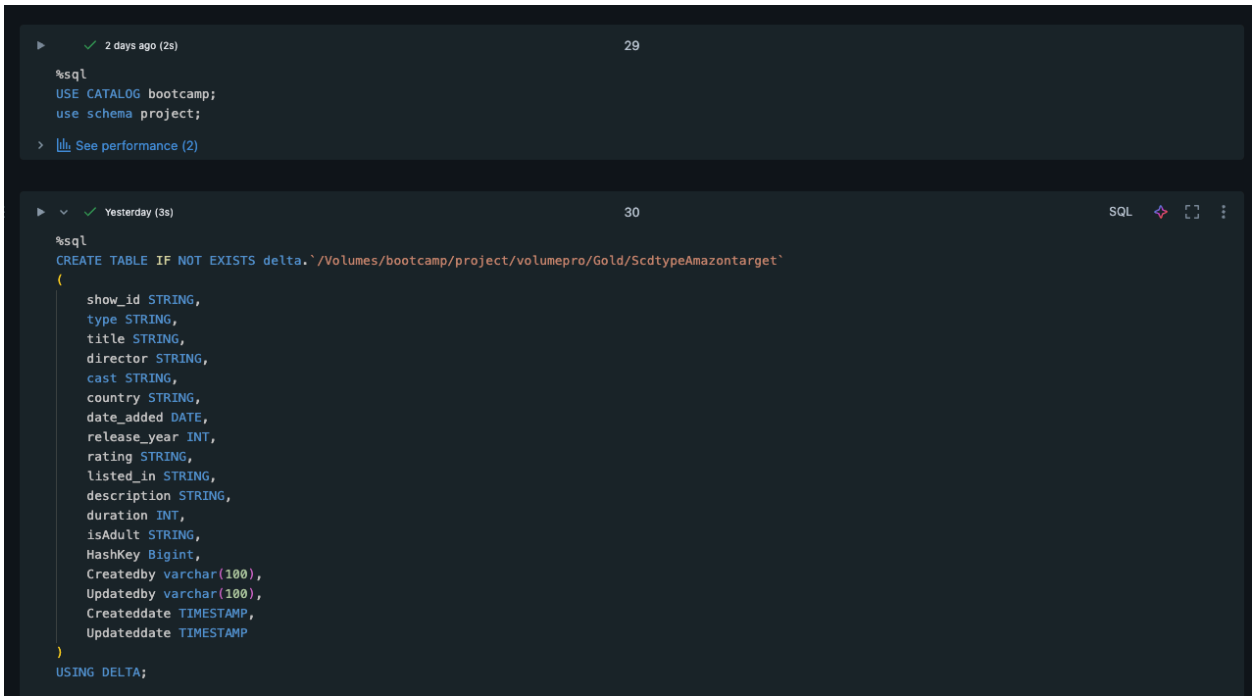
Data Loading and Analysis

Loading the delta table from the silver layer

viewing the data inside the delta table



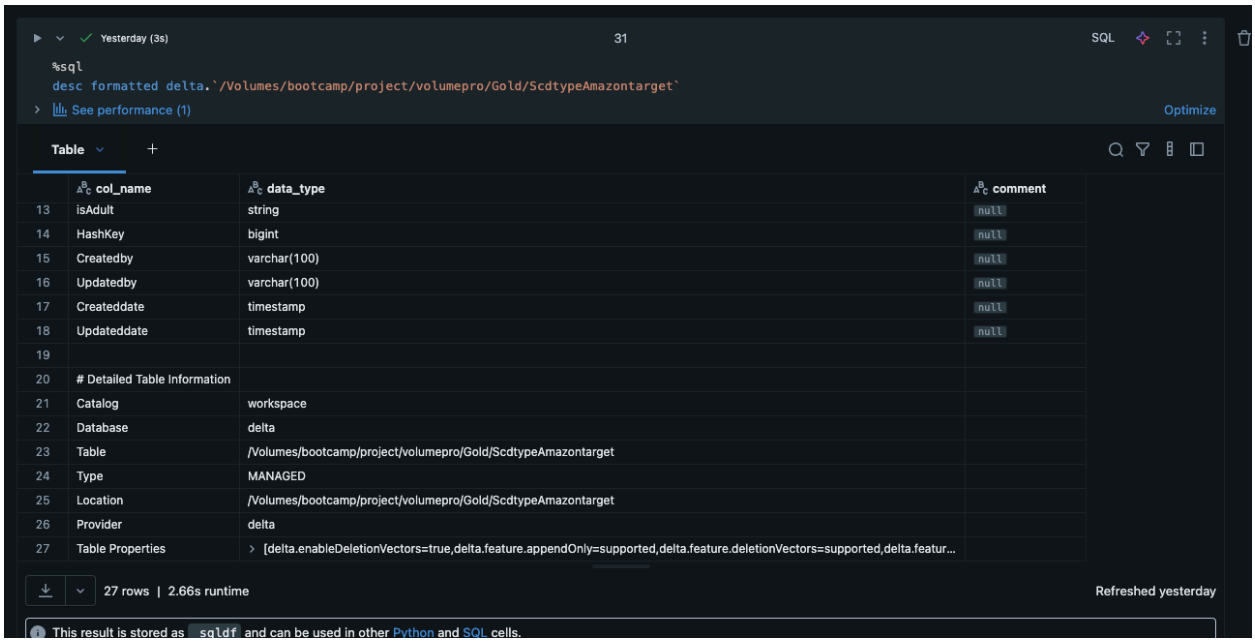
Creating scd type target delta table in the gold layer



```
%sql
USE CATALOG bootcamp;
use schema project;
```

See performance (2)

```
%sql
CREATE TABLE IF NOT EXISTS delta.`/Volumes/bootcamp/project/volumepro/Gold/ScdtypeAmazontarget`
(
    show_id STRING,
    type STRING,
    title STRING,
    director STRING,
    cast STRING,
    country STRING,
    date_added DATE,
    release_year INT,
    rating STRING,
    listed_in STRING,
    description STRING,
    duration INT,
    isAdult STRING,
    HashKey Bigint,
    Createdby varchar(100),
    Updatedby varchar(100),
    Createddate TIMESTAMP,
    Updateddate TIMESTAMP
)
USING DELTA;
```



```
%sql
desc formatted delta.`/Volumes/bootcamp/project/volumepro/Gold/ScdtypeAmazontarget`
```

See performance (1) Optimize

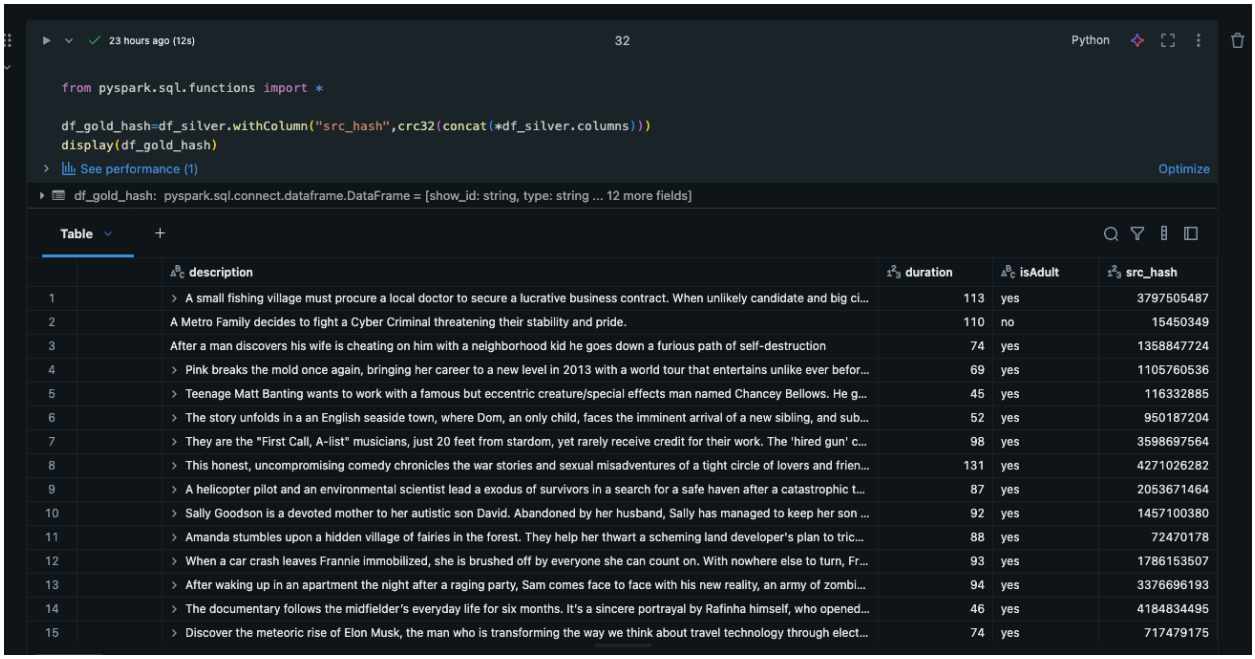
Table

	col_name	data_type	comment
13	isAdult	string	null
14	HashKey	bigint	null
15	Createdby	varchar(100)	null
16	Updatedby	varchar(100)	null
17	Createddate	timestamp	null
18	Updateddate	timestamp	null
19			
20	# Detailed Table Information		
21	Catalog	workspace	
22	Database	delta	
23	Table	/Volumes/bootcamp/project/volumepro/Gold/ScdtypeAmazontarget	
24	Type	MANAGED	
25	Location	/Volumes/bootcamp/project/volumepro/Gold/ScdtypeAmazontarget	
26	Provider	delta	
27	Table Properties	[delta.enableDeletionVectors=true,delta.feature.appendOnly=supported,delta.feature.deletionVectors=supported,delta.featur...	

27 rows | 2.66s runtime Refreshed yesterday

This result is stored as `sqldf` and can be used in other Python and SQL cells.

Creating hash column in the table.



```
from pyspark.sql.functions import *

df_gold_hash=df_silver.withColumn("src_hash",crc32(concat(*df_silver.columns)))
display(df_gold_hash)
```

23 hours ago (12s) 32 Python

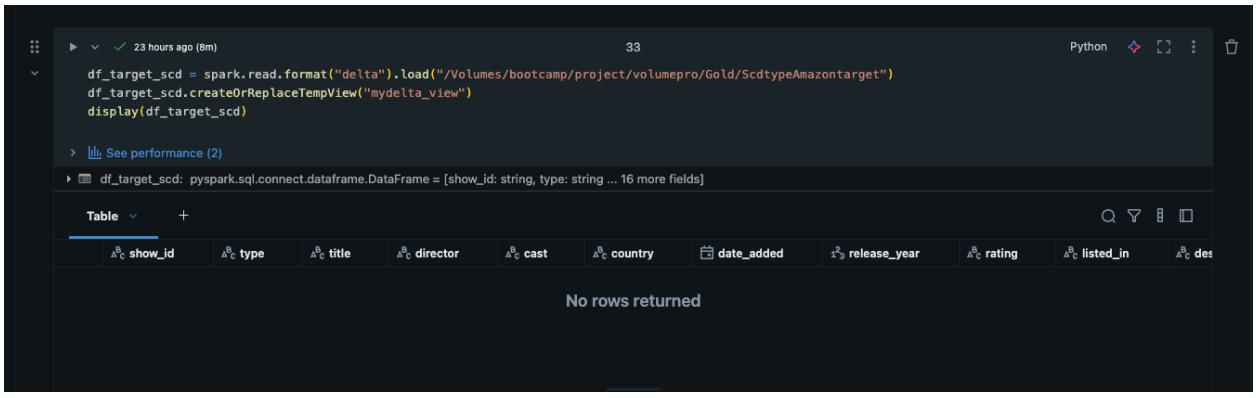
See performance (1) Optimize

df_gold_hash: pyspark.sql.connect.dataframe.DataFrame = [show_id: string, type: string ... 12 more fields]

Table

	description	duration	isAdult	src_hash
1	A small fishing village must procure a local doctor to secure a lucrative business contract. When unlikely candidate and big ci...	113	yes	3797505487
2	A Metro Family decides to fight a Cyber Criminal threatening their stability and pride.	110	no	15450349
3	After a man discovers his wife is cheating on him with a neighborhood kid he goes down a furious path of self-destruction	74	yes	1358847724
4	Pink breaks the mold once again, bringing her career to a new level in 2013 with a world tour that entertains unlike ever befor...	69	yes	1105760536
5	Teenage Matt Banting wants to work with a famous but eccentric creature/special effects man named Chancey Bellows. He g...	45	yes	116332885
6	The story unfolds in a an English seaside town, where Dom, an only child, faces the imminent arrival of a new sibling, and sub...	52	yes	950187204
7	They are the "First Call, A-list" musicians, just 20 feet from stardom, yet rarely receive credit for their work. The 'hired gun' c...	98	yes	3598697564
8	This honest, uncompromising comedy chronicles the war stories and sexual misadventures of a tight circle of lovers and frien...	131	yes	4271026282
9	A helicopter pilot and an environmental scientist lead a exodus of survivors in a search for a safe haven after a catastrophic t...	87	yes	2053671464
10	Sally Goodson is a devoted mother to her autistic son David. Abandoned by her husband, Sally has managed to keep her son ...	92	yes	1457100380
11	Amanda stumbles upon a hidden village of fairies in the forest. They help her thwart a scheming land developer's plan to tric...	88	yes	72470178
12	When a car crash leaves Frannie immobilized, she is brushed off by everyone she can count on. With nowhere else to turn, Fr...	93	yes	1786153507
13	After waking up in an apartment the night after a raging party, Sam comes face to face with his new reality, an army of zombi...	94	yes	3376696193
14	The documentary follows the midfielder's everyday life for six months. It's a sincere portrayal by Rafinha himself, who opened...	46	yes	4184834495
15	Discover the meteoric rise of Elon Musk, the man who is transforming the way we think about travel technology through elect...	74	yes	717479175

Reading the scd type table from the gold layet



```
df_target_scd = spark.read.format("delta").load("/Volumes/bootcamp/project/volumepro/Gold/ScdtypeAmazontarget")
df_target_scd.createOrReplaceTempView("mydelta_view")
display(df_target_scd)
```

23 hours ago (8m) 33 Python

See performance (2)

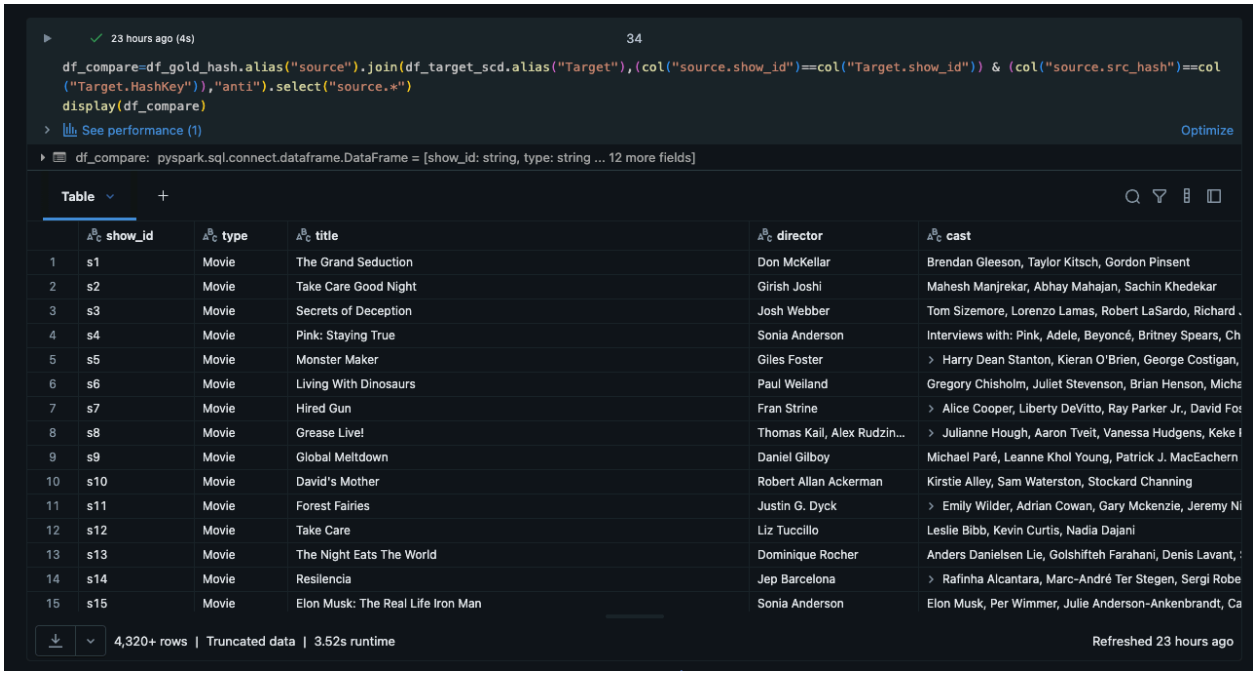
df_target_scd: pyspark.sql.connect.dataframe.DataFrame = [show_id: string, type: string ... 16 more fields]

Table

show_id	type	title	director	cast	country	date_added	release_year	rating	listed_in	des

No rows returned

Comparing the data in scd type table with new data in the source. Only the new or updated data will be shown here.



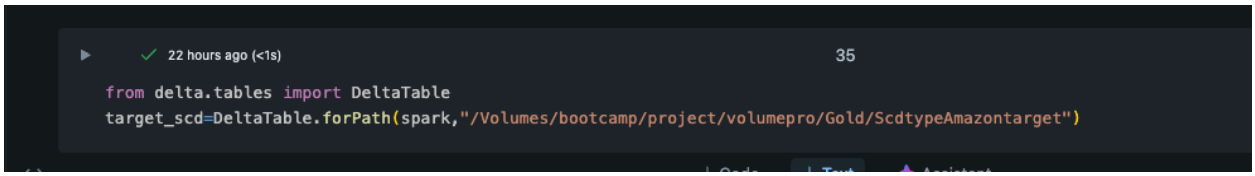
```
df_compare=df_gold_hash.alias("source").join(df_target_scd.alias("Target"),(col("source.show_id")==col("Target.show_id")) & (col("source.src_hash")==col("Target.HashKey")),"anti").select("source.*")
display(df_compare)
```

df_compare: pyspark.sql.connect.dataframe.DataFrame = [show_id: string, type: string ... 12 more fields]

	show_id	type	title	director	cast
1	s1	Movie	The Grand Seduction	Don McKellar	Brendan Gleeson, Taylor Kitsch, Gordon Pinsent
2	s2	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan, Sachin Khedekar
3	s3	Movie	Secrets of Deception	Josh Webber	Tom Sizemore, Lorenzo Lamas, Robert LaSardo, Richard
4	s4	Movie	Pink: Staying True	Sonia Anderson	Interviews with: Pink, Adele, Beyoncé, Britney Spears, Ch
5	s5	Movie	Monster Maker	Giles Foster	Harry Dean Stanton, Kieran O'Brien, George Costigan,
6	s6	Movie	Living With Dinosaurs	Paul Weiland	Gregory Chisholm, Juliet Stevenson, Brian Henson, Micha
7	s7	Movie	Hired Gun	Fran Strine	Alice Cooper, Liberty DeVitto, Ray Parker Jr., David Fos
8	s8	Movie	Grease Live!	Thomas Kail, Alex Rudzin...	Julianne Hough, Aaron Tveit, Vanessa Hudgens, Keke I
9	s9	Movie	Global Meltdown	Daniel Gilboy	Michael Paré, Leanne Khol Young, Patrick J. MacEachern
10	s10	Movie	David's Mother	Robert Allan Ackerman	Kirstie Alley, Sam Waterston, Stockard Channing
11	s11	Movie	Forest Fairies	Justin G. Dyck	Emily Wilder, Adrian Cowan, Gary Mckenzie, Jeremy Ni
12	s12	Movie	Take Care	Liz Tuccillo	Leslie Bibb, Kevin Curtis, Nadia Dajani
13	s13	Movie	The Night Eats The World	Dominique Rocher	Anders Danielsen Lie, Golshifteh Farahani, Denis Lavant,
14	s14	Movie	Resilencia	Jep Barcelona	Rafinha Alcantara, Marc-André Ter Stegen, Sergi Robe
15	s15	Movie	Elon Musk: The Real Life Iron Man	Sonia Anderson	Elon Musk, Per Wimmer, Julie Anderson-Ankenbrandt, Ca

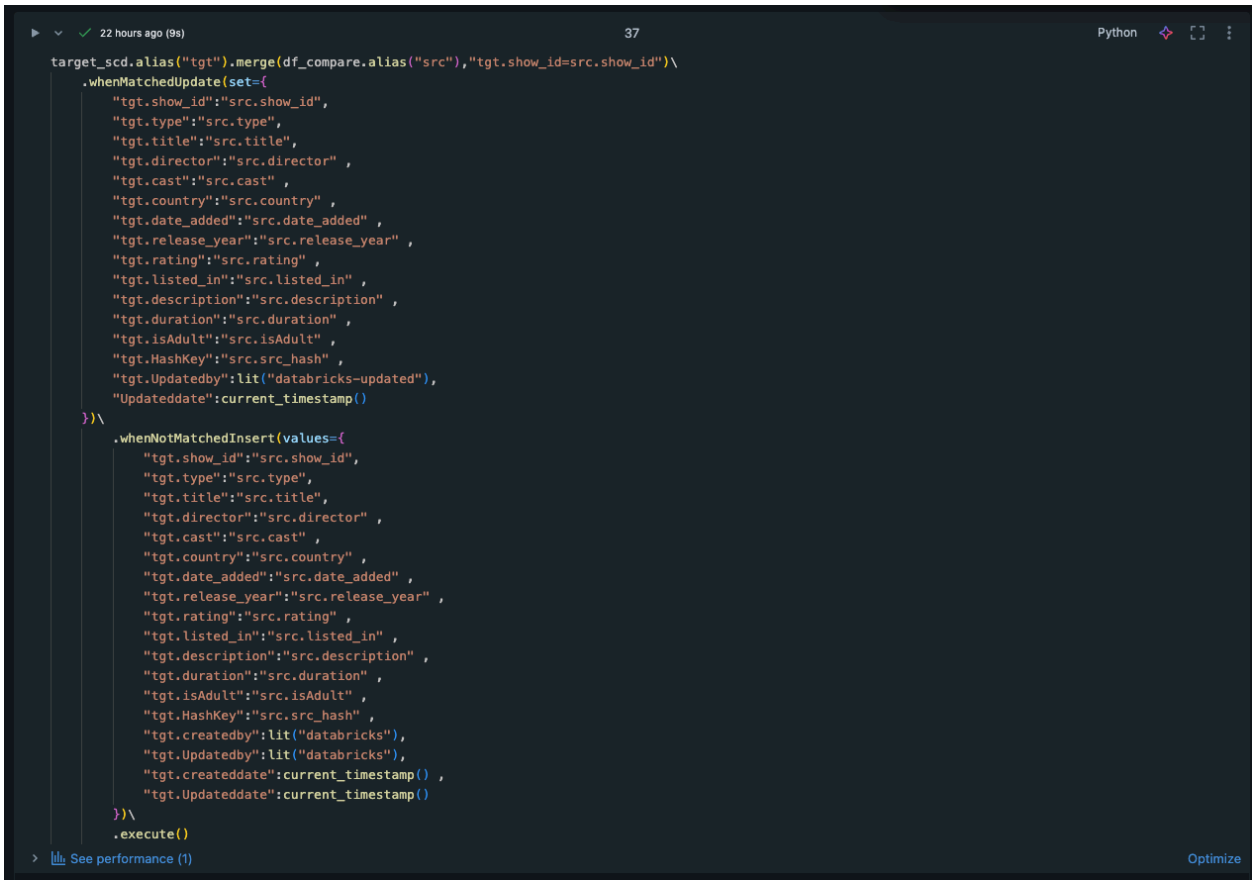
4,320+ rows | Truncated data | 3.52s runtime

Loading scd type in delta format from the gold layer.

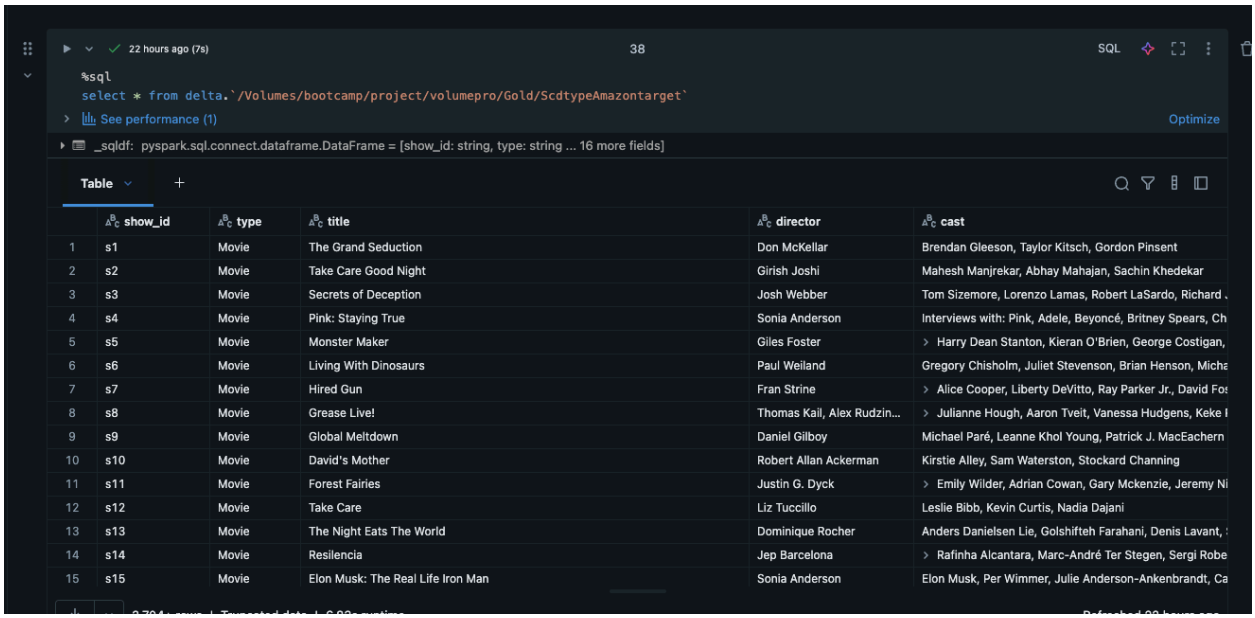


```
from delta.tables import DeltaTable
target_scd=DeltaTable.forPath(spark,"/Volumes/bootcamp/project/volumepro/Gold/ScdtypeAmazontarget")
```

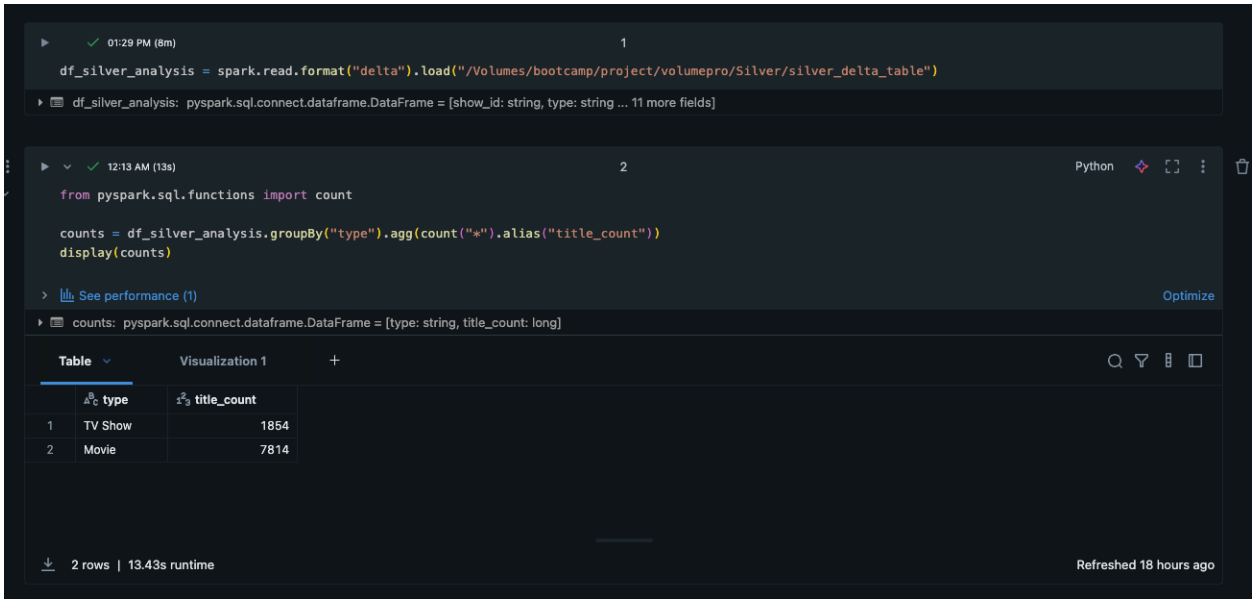
Implementing the scd type-1 logic and mapping the columns.



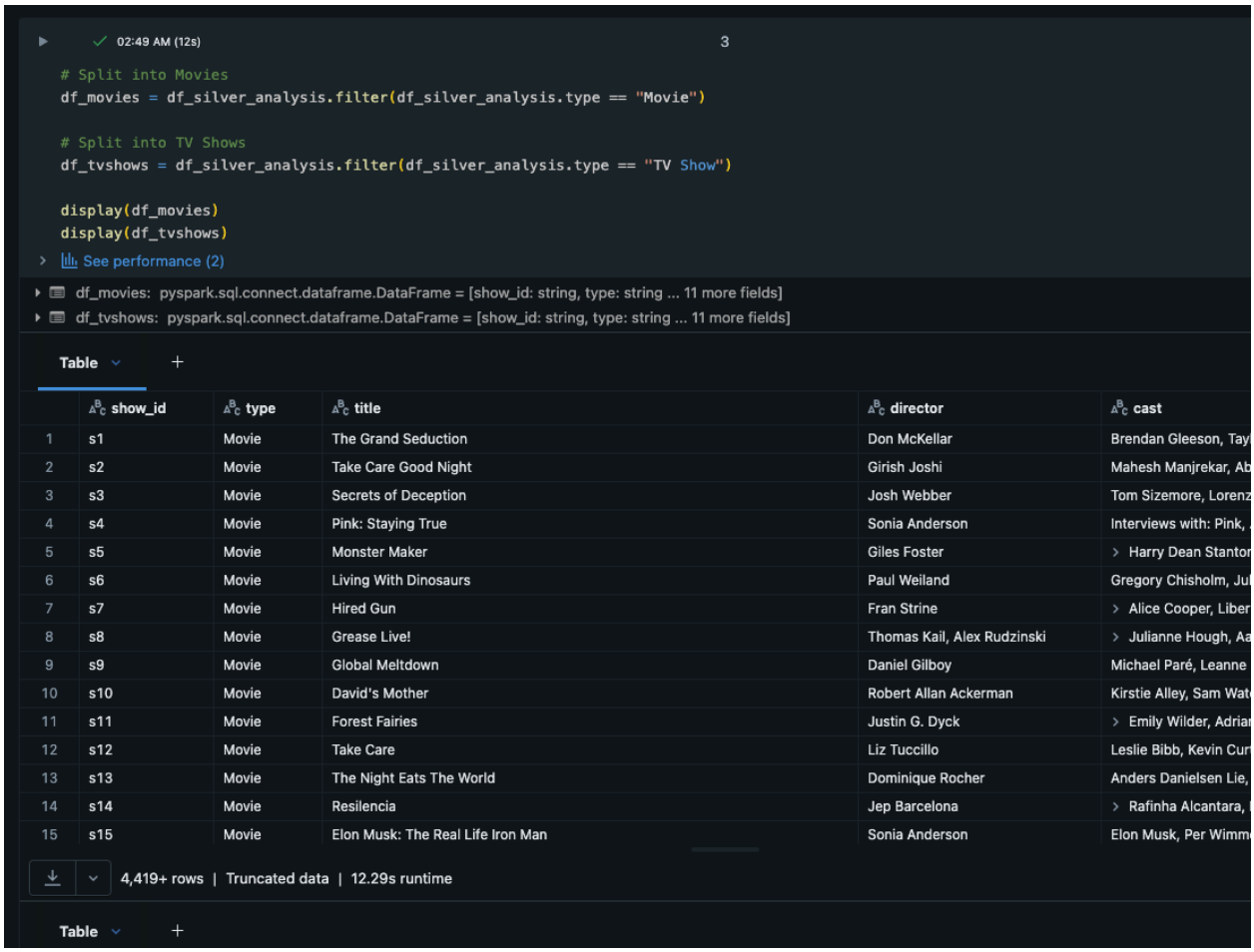
Checking the inserted values in the scd type table.



Loading the data from the silver layer for the analysis. Showing the number of titles for TV shows and movies.



Separating the data into two dataframes movies and tv shows.



```
02:49 AM (12s)                                                    3

# Split into Movies
df_movies = df_silver_analysis.filter(df_silver_analysis.type == "Movie")

# Split into TV Shows
df_tvshows = df_silver_analysis.filter(df_silver_analysis.type == "TV Show")

display(df_movies)
display(df_tvshows)
```

See performance (2)

df_movies: pyspark.sql.connect.dataframe.DataFrame = [show_id: string, type: string ... 11 more fields]

df_tvshows: pyspark.sql.connect.dataframe.DataFrame = [show_id: string, type: string ... 11 more fields]

Table +

	show_id	type	title	director	cast
1	s1	Movie	The Grand Seduction	Don McKellar	Brendan Gleeson, Tay
2	s2	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Ab
3	s3	Movie	Secrets of Deception	Josh Webber	Tom Sizemore, Lorenz
4	s4	Movie	Pink: Staying True	Sonia Anderson	Interviews with: Pink,
5	s5	Movie	Monster Maker	Giles Foster	Harry Dean Stantor
6	s6	Movie	Living With Dinosaurs	Paul Weiland	Gregory Chisholm, Ju
7	s7	Movie	Hired Gun	Fran Strine	Alice Cooper, Liber
8	s8	Movie	Grease Live!	Thomas Kail, Alex Rudzinski	Julianne Hough, Aa
9	s9	Movie	Global Meltdown	Daniel Gilboy	Michael Paré, Leanne
10	s10	Movie	David's Mother	Robert Allan Ackerman	Kirstie Alley, Sam Wat
11	s11	Movie	Forest Fairies	Justin G. Dyck	Emily Wilder, Adriar
12	s12	Movie	Take Care	Liz Tuccillo	Leslie Bibb, Kevin Cur
13	s13	Movie	The Night Eats The World	Dominique Rocher	Anders Danielsen Lie,
14	s14	Movie	Resilencia	Jep Barcelona	Rafinha Alcantara,
15	s15	Movie	Elon Musk: The Real Life Iron Man	Sonia Anderson	Elon Musk, Per Wimm

4,419+ rows | Truncated data | 12.29s runtime

Table +

TV shows's data frame.

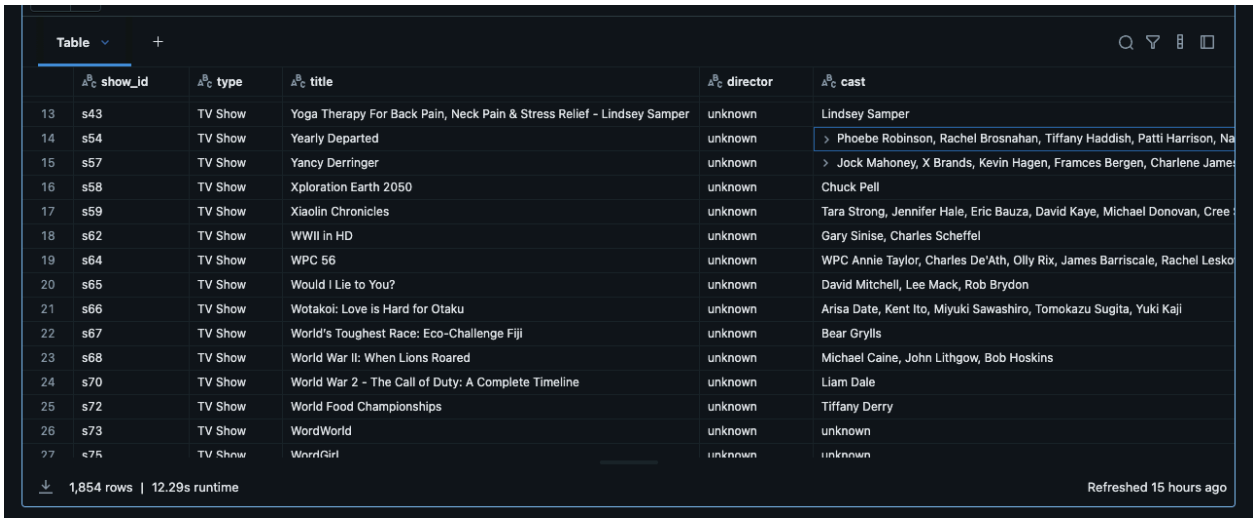


Table +

	show_id	type	title	director	cast
13	s43	TV Show	Yoga Therapy For Back Pain, Neck Pain & Stress Relief - Lindsey Samper	unknown	Lindsey Samper
14	s54	TV Show	Yearly Departed	unknown	Phoebe Robinson, Rachel Brosnahan, Tiffany Haddish, Patti Harrison, Na
15	s57	TV Show	Yancy Derringer	unknown	Jock Mahoney, X Brands, Kevin Hagen, Framces Bergen, Charlene James
16	s58	TV Show	Xploration Earth 2050	unknown	Chuck Pell
17	s59	TV Show	Xiaolin Chronicles	unknown	Tara Strong, Jennifer Hale, Eric Bauza, David Kaye, Michael Donovan, Cree
18	s62	TV Show	WWII in HD	unknown	Gary Sinise, Charles Scheffel
19	s64	TV Show	WPC 56	unknown	WPC Annie Taylor, Charles De'Ath, Olly Rix, James Barriscale, Rachel Lesko
20	s65	TV Show	Would I Lie to You?	unknown	David Mitchell, Lee Mack, Rob Brydon
21	s66	TV Show	Wotakoi: Love is Hard for Otaku	unknown	Arisa Date, Kent Ito, Miyuki Sawashiro, Tomokazu Sugita, Yuki Kaji
22	s67	TV Show	World's Toughest Race: Eco-Challenge Fiji	unknown	Bear Grylls
23	s68	TV Show	World War II: When Lions Roared	unknown	Michael Caine, John Lithgow, Bob Hoskins
24	s70	TV Show	World War 2 - The Call of Duty: A Complete Timeline	unknown	Liam Dale
25	s72	TV Show	World Food Championships	unknown	Tiffany Derry
26	s73	TV Show	WordWorld	unknown	unknown
27	s75	TV Show	WordGirl	unknown	unknown

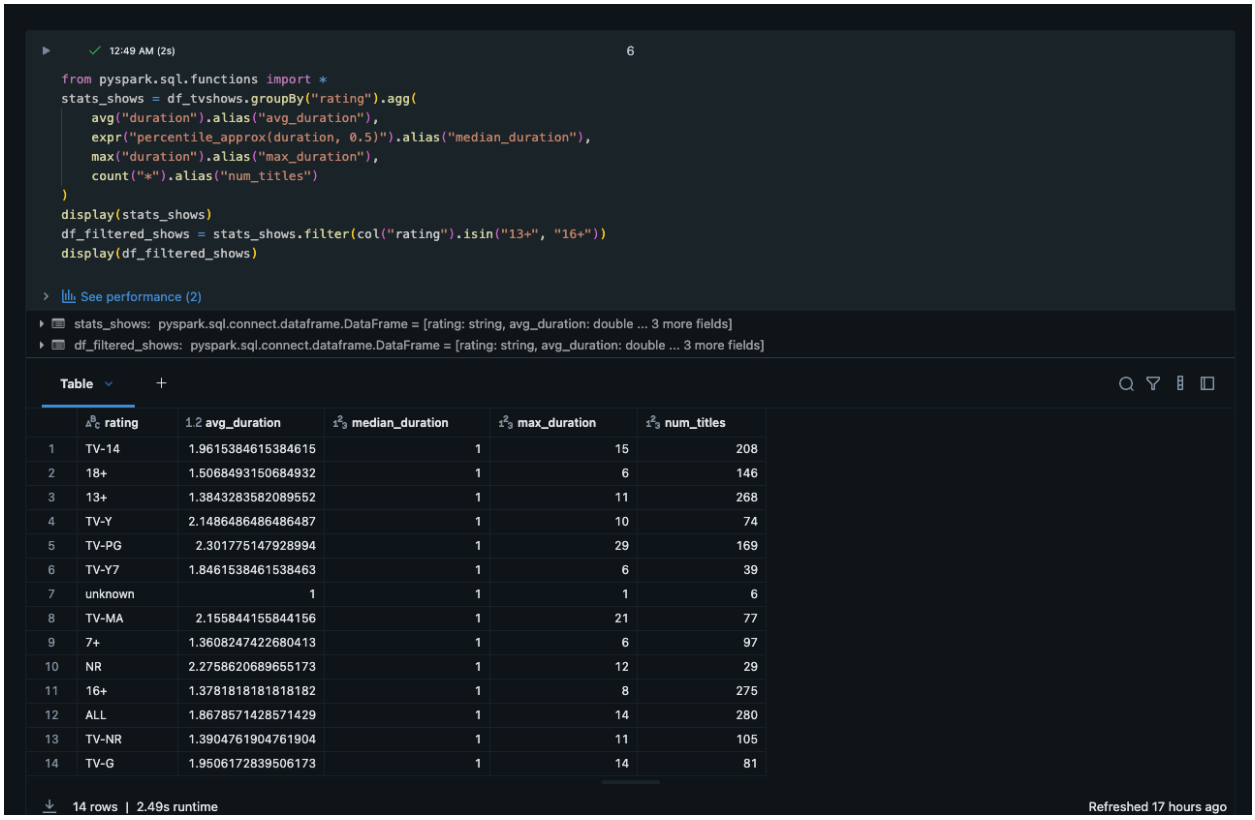
1,854 rows | 12.29s runtime

Refreshed 15 hours ago

Top 5 movies with their durations.

Top 5 tv shows with there duration.

Computing TV show's average/median/max duration and number of titles for each rating (especially **13+** and **16+**)



12:49 AM (2s) 6

```
from pyspark.sql.functions import *
stats_shows = df_tvshows.groupBy("rating").agg(
    avg("duration").alias("avg_duration"),
    expr("percentile_approx(duration, 0.5)").alias("median_duration"),
    max("duration").alias("max_duration"),
    count("*").alias("num_titles")
)
display(stats_shows)
df_filtered_shows = stats_shows.filter(col("rating").isin("13+", "16+"))
display(df_filtered_shows)
```

See performance (2)

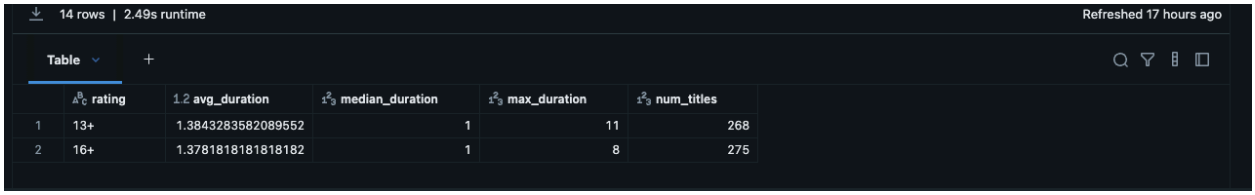
stats_shows: pyspark.sql.connect.dataframe.DataFrame = [rating: string, avg_duration: double ... 3 more fields]

df_filtered_shows: pyspark.sql.connect.dataframe.DataFrame = [rating: string, avg_duration: double ... 3 more fields]

Table +

	rating	avg_duration	median_duration	max_duration	num_titles
1	TV-14	1.9615384615384615	1	15	208
2	18+	1.5068493150684932	1	6	146
3	13+	1.3843283582089552	1	11	268
4	TV-Y	2.1486486486486487	1	10	74
5	TV-PG	2.301775147928994	1	29	169
6	TV-Y7	1.8461538461538463	1	6	39
7	unknown	1	1	1	6
8	TV-MA	2.155844155844156	1	21	77
9	7+	1.3608247422680413	1	6	97
10	NR	2.2758620689655173	1	12	29
11	16+	1.3781818181818182	1	8	275
12	ALL	1.8678571428571429	1	14	280
13	TV-NR	1.3904761904761904	1	11	105
14	TV-G	1.9506172839506173	1	14	81

14 rows | 2.49s runtime Refreshed 17 hours ago

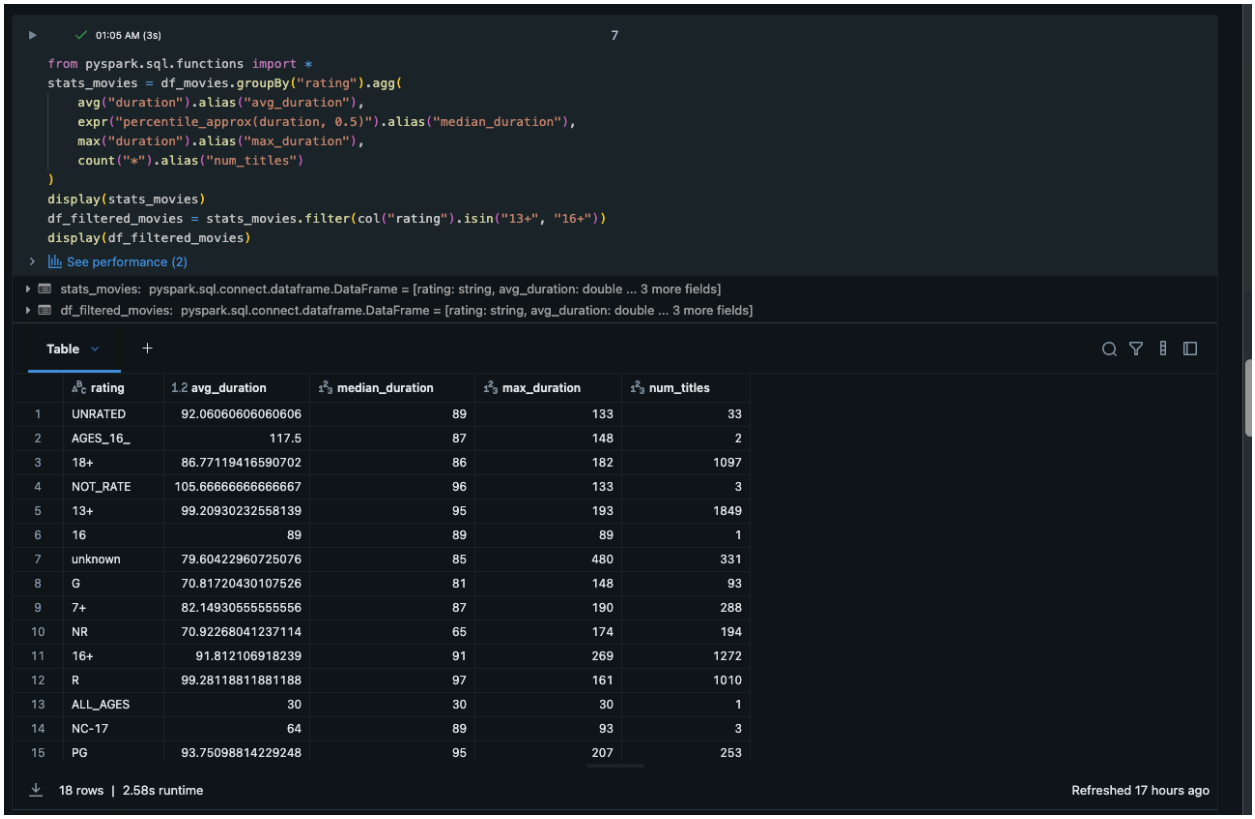


14 rows | 2.49s runtime Refreshed 17 hours ago

Table +

	rating	avg_duration	median_duration	max_duration	num_titles
1	13+	1.3843283582089552	1	11	268
2	16+	1.3781818181818182	1	8	275

Computing movie's average/median/max duration and number of titles for each rating (especially **13+** and **16+**)



```
from pyspark.sql.functions import *
stats_movies = df_movies.groupBy("rating").agg(
    avg("duration").alias("avg_duration"),
    expr("percentile_approx(duration, 0.5)").alias("median_duration"),
    max("duration").alias("max_duration"),
    count("*").alias("num_titles")
)
display(stats_movies)
df_filtered_movies = stats_movies.filter(col("rating").isin("13+", "16+"))
display(df_filtered_movies)
```

01:05 AM (3s)

See performance (2)

stats_movies: pyspark.sql.connect.dataframe.DataFrame = [rating: string, avg_duration: double ... 3 more fields]

df_filtered_movies: pyspark.sql.connect.dataframe.DataFrame = [rating: string, avg_duration: double ... 3 more fields]

Table

	rating	avg_duration	median_duration	max_duration	num_titles
1	UNRATED	92.06060606060606	89	133	33
2	AGES_16_	117.5	87	148	2
3	18+	86.77119416590702	86	182	1097
4	NOT_RATE	105.66666666666667	96	133	3
5	13+	99.20930232558139	95	193	1849
6	16	89	89	89	1
7	unknown	79.60422960725076	85	480	331
8	G	70.81720430107526	81	148	93
9	7+	82.14930555555556	87	190	288
10	NR	70.92268041237114	65	174	194
11	16+	91.812106918239	91	269	1272
12	R	99.28118811881188	97	161	1010
13	ALL_AGES	30	30	30	1
14	NC-17	64	89	93	3
15	PG	93.75098814229248	95	207	253

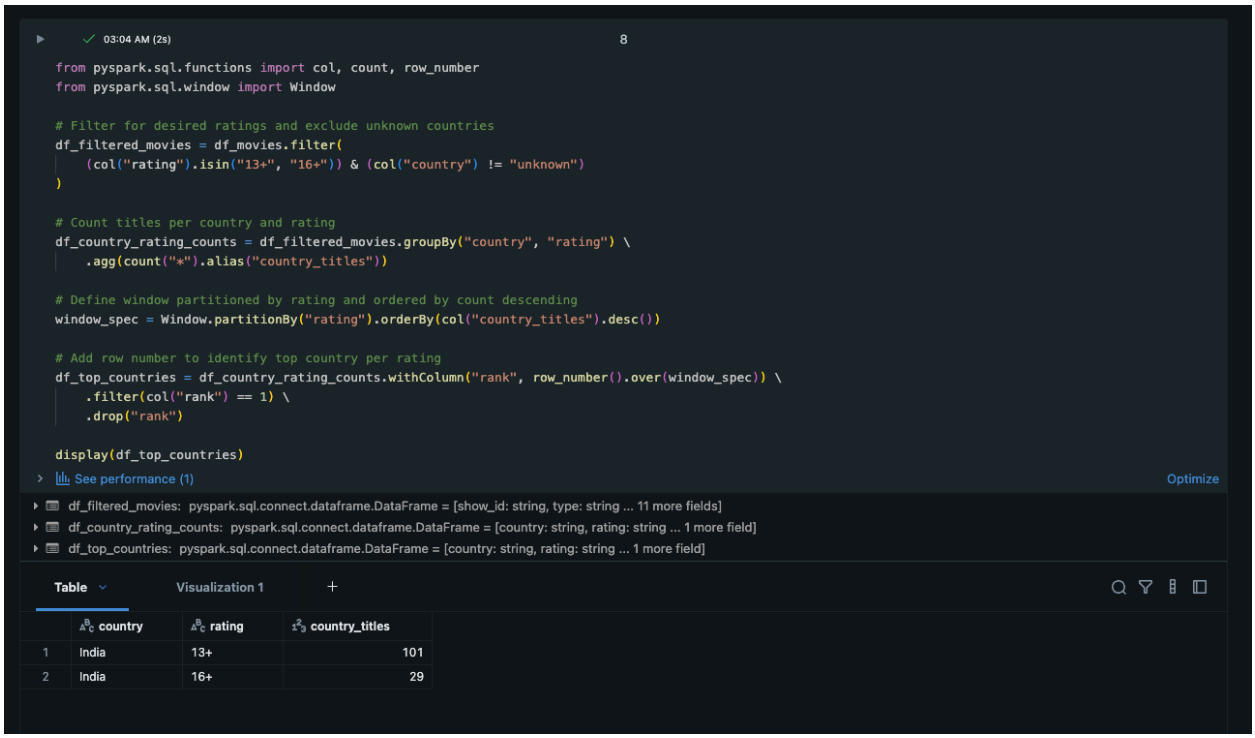
18 rows | 2.58s runtime

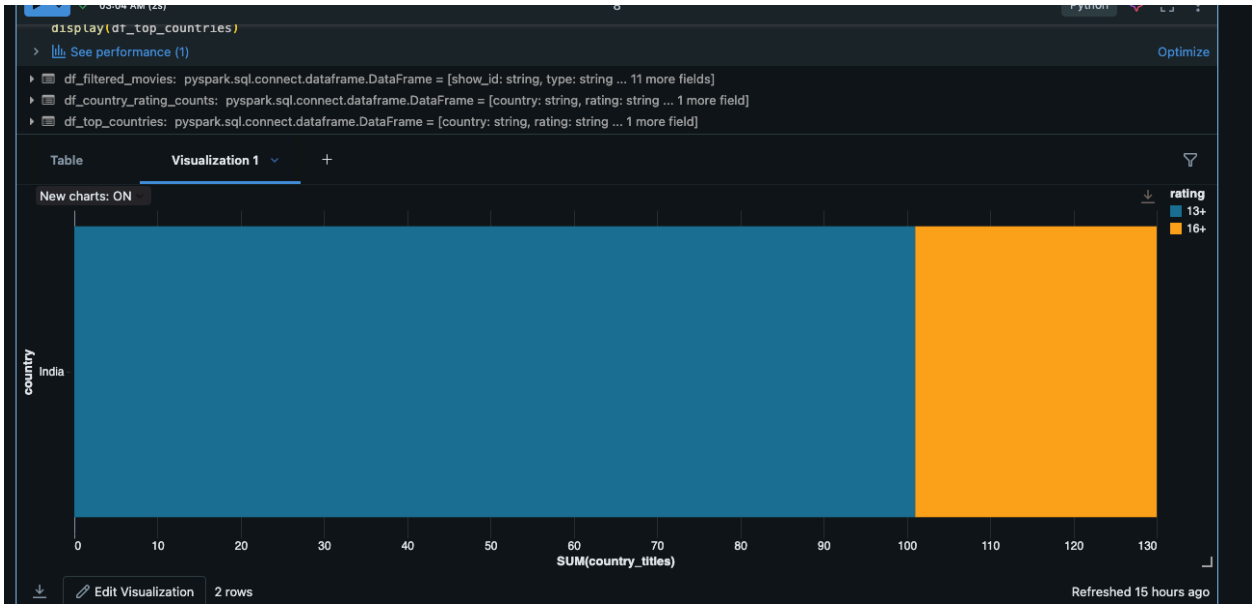
Refreshed 17 hours ago

Table

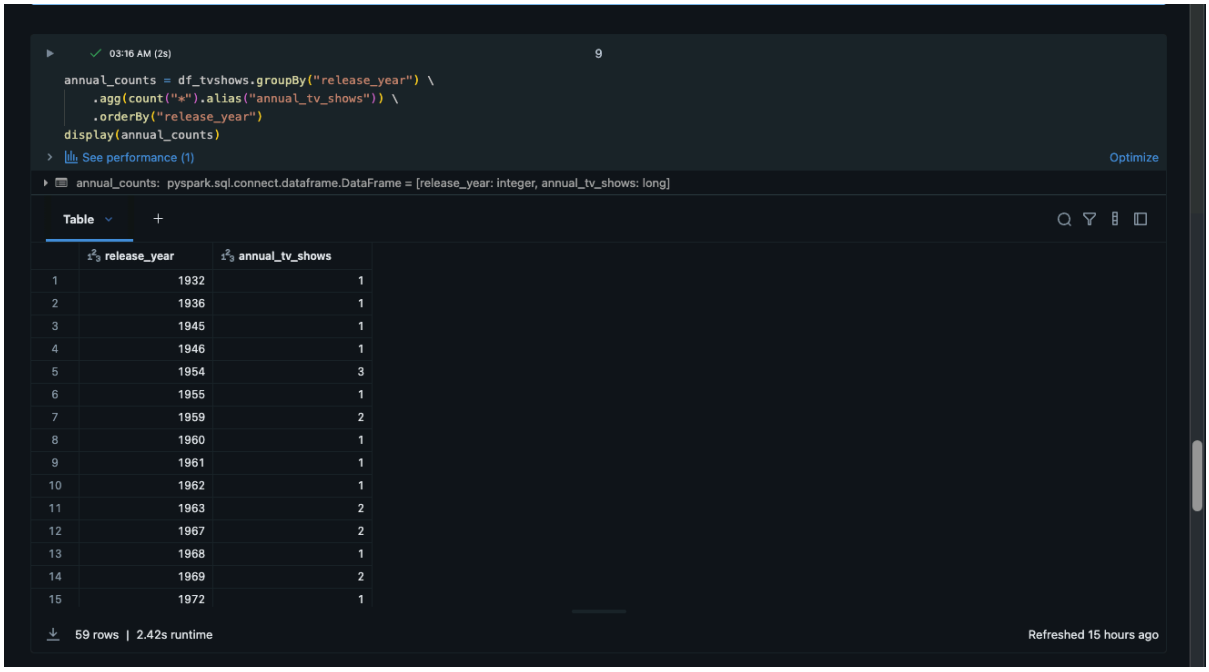
	rating	avg_duration	median_duration	max_duration	num_titles
1	13+	99.20930232558139	95	193	1849
2	16+	91.812106918239	91	269	1272

Most movies released by the particular rating by each country.





The annual number of TV shows released each year.



```
03:16 AM (2s)                                                    9

annual_counts = df_tvshows.groupBy("release_year") \
    .agg(count("*").alias("annual_tv_shows")) \
    .orderBy("release_year")
display(annual_counts)
```

See performance (1) Optimize

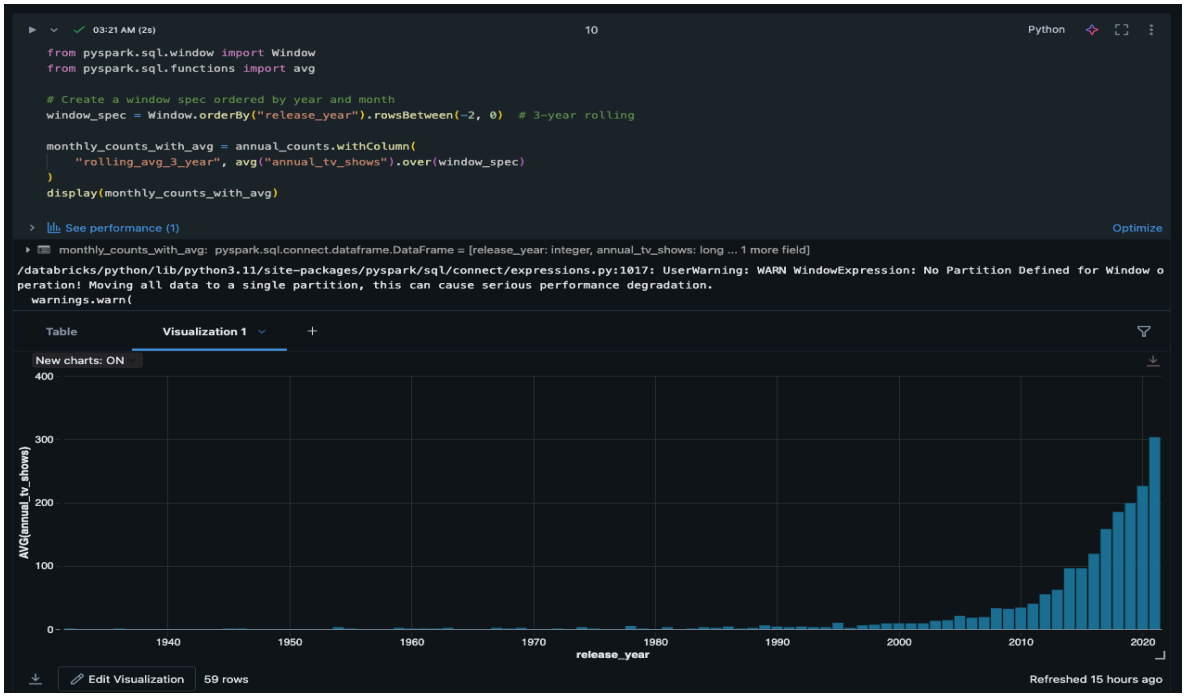
annual_counts: pyspark.sql.connect.dataframe.DataFrame = [release_year: integer, annual_tv_shows: long]

Table +

	release_year	annual_tv_shows
1	1932	1
2	1936	1
3	1945	1
4	1946	1
5	1954	3
6	1955	1
7	1959	2
8	1960	1
9	1961	1
10	1962	1
11	1963	2
12	1967	2
13	1968	1
14	1969	2
15	1972	1

59 rows | 2.42s runtime Refreshed 15 hours ago

Total number of tv shows released by the rolling average of 3 years



```
03:21 AM (2s)                                                    10                    Python

from pyspark.sql.window import Window
from pyspark.sql.functions import avg

# Create a window spec ordered by year and month
window_spec = Window.orderBy("release_year").rowsBetween(-2, 0)  # 3-year rolling

monthly_counts_with_avg = annual_counts.withColumn(
    "rolling_avg_3_year", avg("annual_tv_shows").over(window_spec)
)
display(monthly_counts_with_avg)
```

See performance (1) Optimize

monthly_counts_with_avg: pyspark.sql.connect.dataframe.DataFrame = [release_year: integer, annual_tv_shows: long ... 1 more field]

```
/databricks/python/lib/python3.11/site-packages/pyspark/sql/connect/expressions.py:1017: UserWarning: WARN WindowExpression: No Partition Defined for Window o
peration! Moving all data to a single partition, this can cause serious performance degradation.
  warnings.warn(
```

Table Visualization 1 +

New charts: ON

Edit Visualization 59 rows Refreshed 15 hours ago

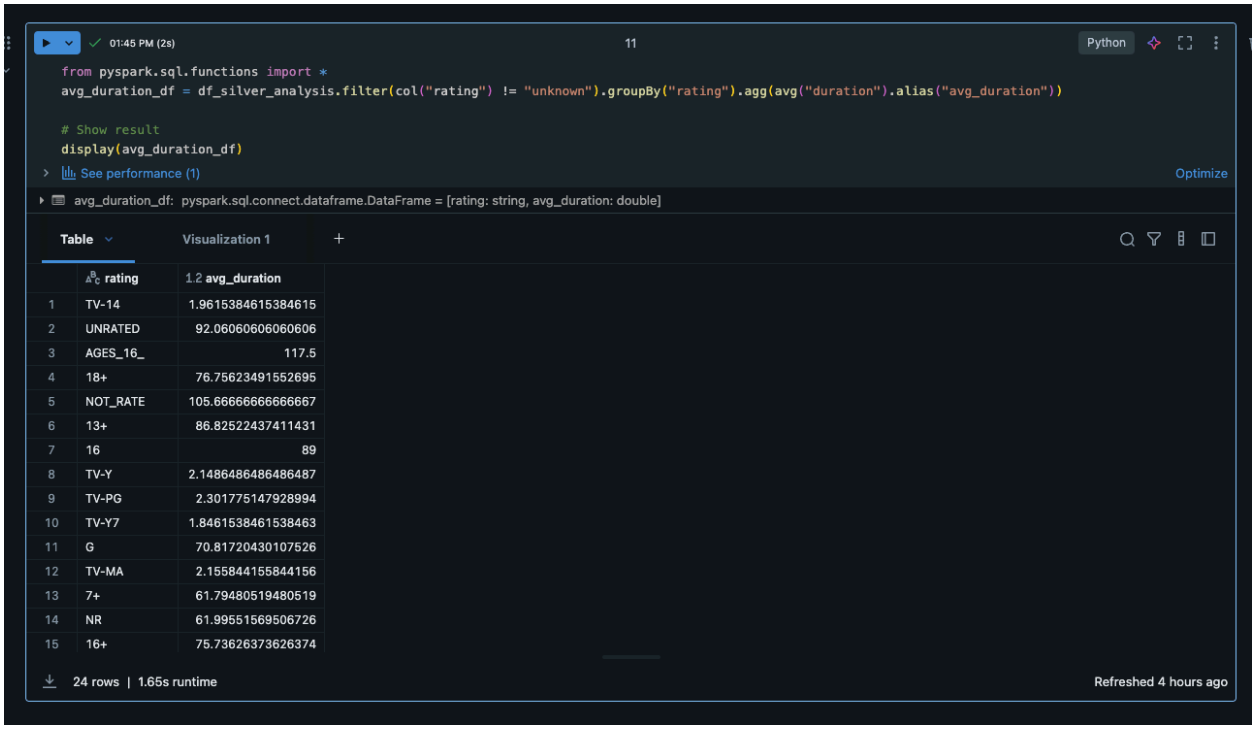
warnings.warn(

Table | Visualization 1 | +

	release_year	annual_tv_shows	rolling_avg_3_year
1	1932	1	1
2	1936	1	1
3	1945	1	1
4	1946	1	1
5	1954	3	1.6666666666666667
6	1955	1	1.6666666666666667
7	1959	2	2
8	1960	1	1.3333333333333333
9	1961	1	1.3333333333333333
10	1962	1	1
11	1963	2	1.3333333333333333
12	1967	2	1.6666666666666667
13	1968	1	1.6666666666666667
14	1969	2	1.6666666666666667
15	1972	1	1.3333333333333333

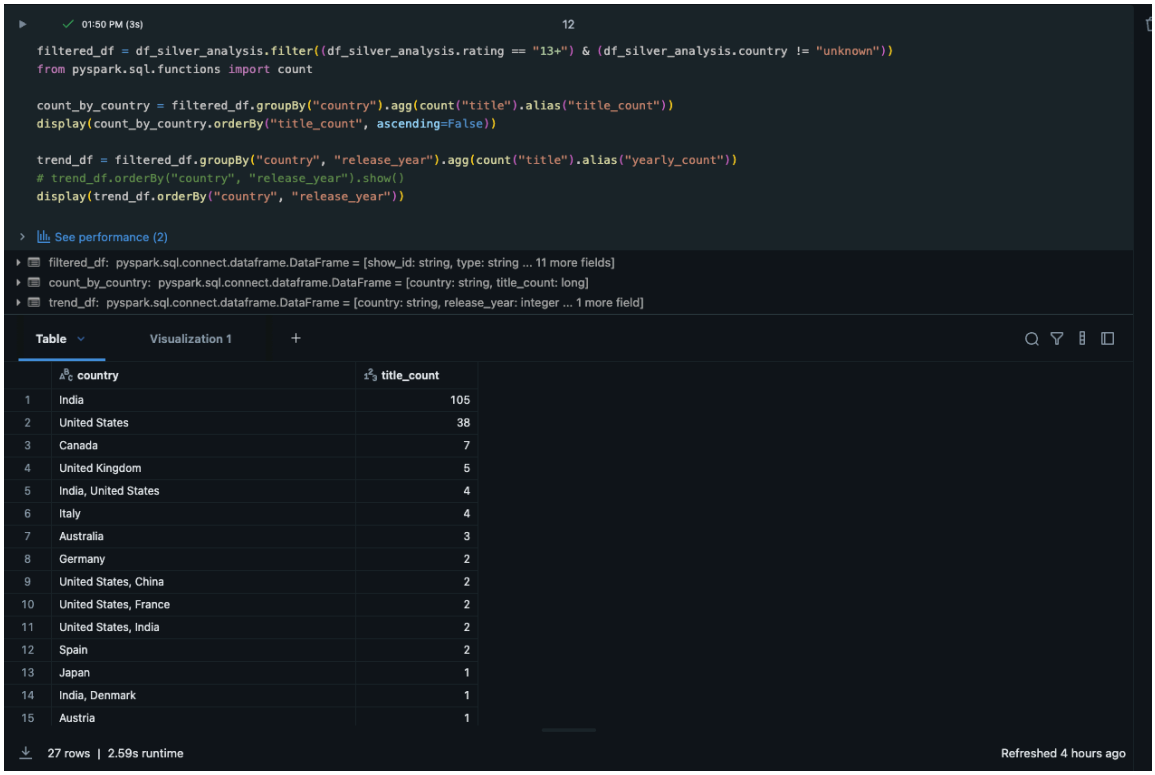
59 rows | 2.04s runtime

Average duration of movie and tv shows for each rating.



Creating visualisation for average duration for each ratings.

Total number of titles released by each country.



Creating visualization for the same.