

AMAZON PRIME VIDEOS ANALYSIS AND SCD TYPE-1 IMPLEMENTATION DATA ENGINEERING C-29



Introduction

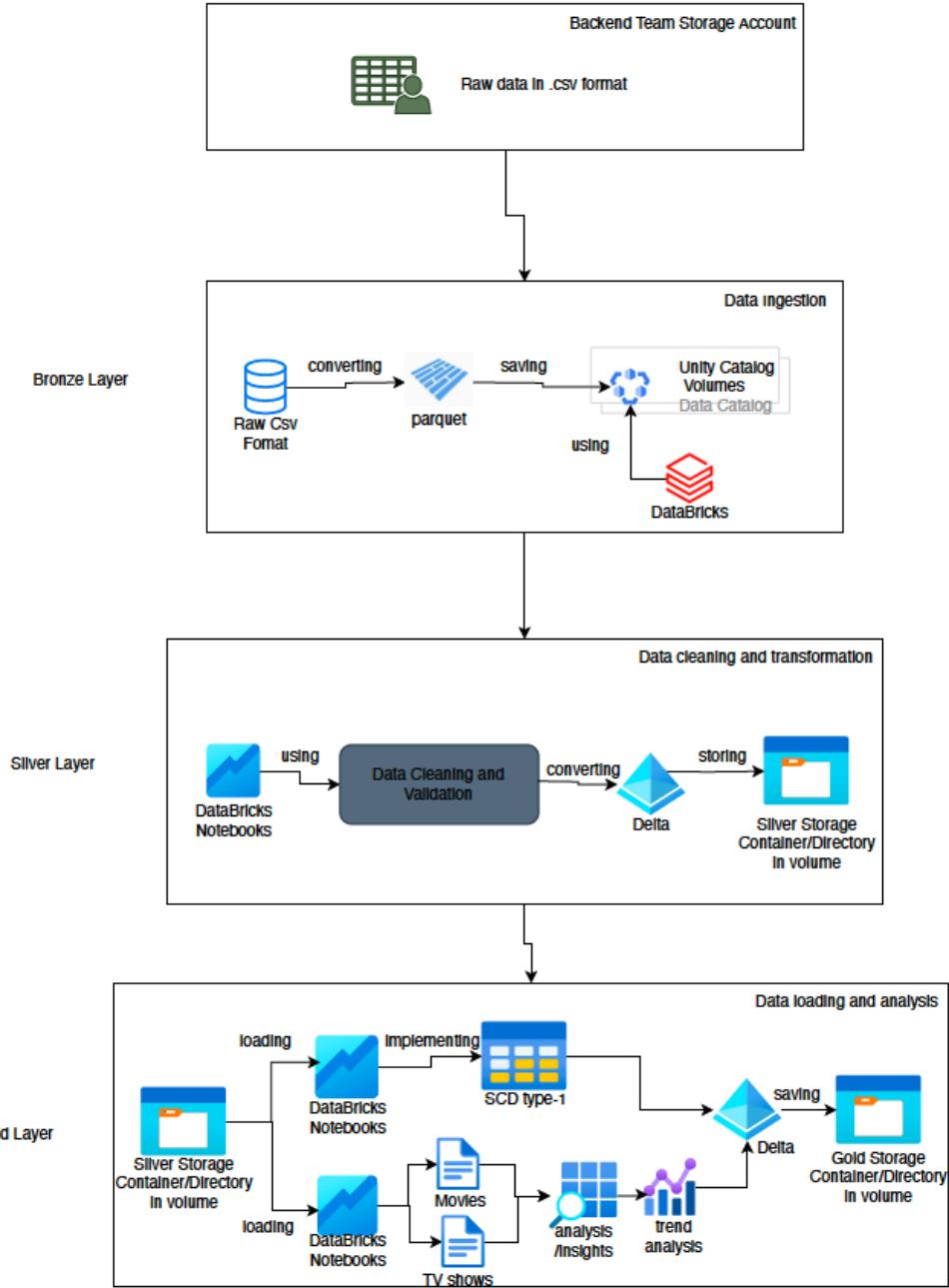
The growth of online streaming platforms such as Amazon Prime has led to an explosion of content in the form of movies and TV shows. To extract meaningful insights from this vast amount of data, it is crucial to design an efficient, scalable, and secure data pipeline that can handle ingestion, transformation,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

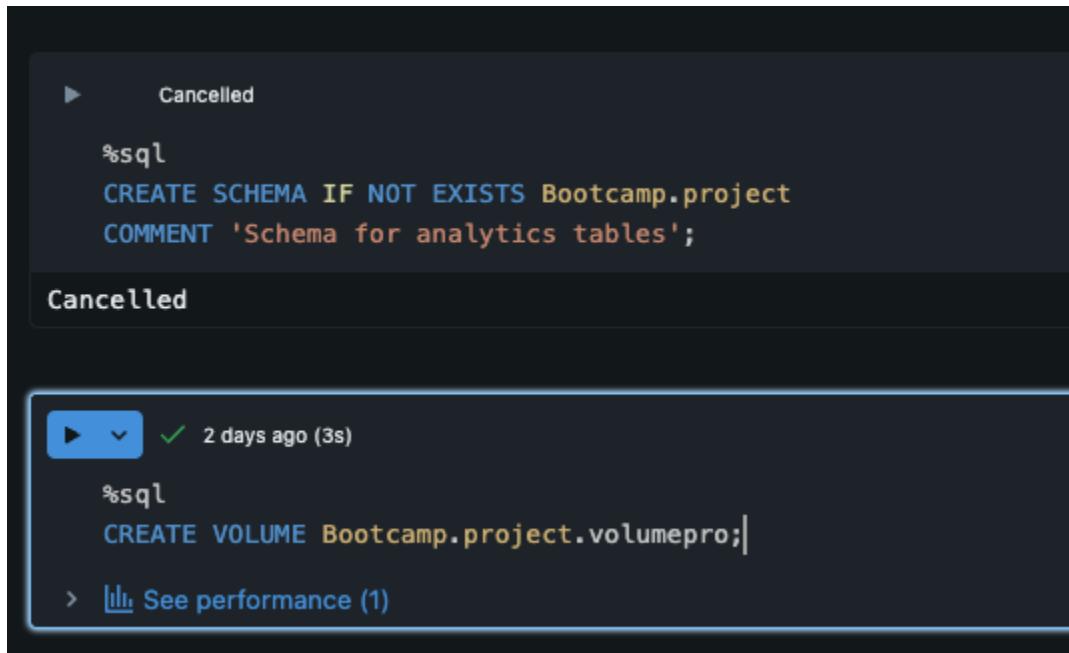
Architecture Diagram



Creating Volumes in Databricks

First create Catalog.

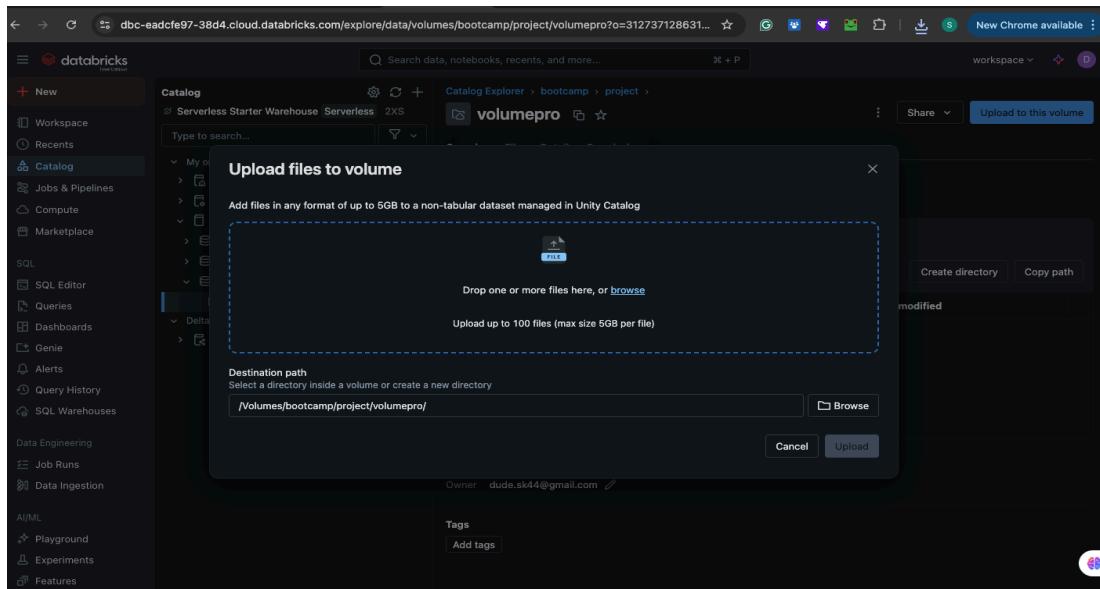
Create Schema by running this command in the Databricks notebook.



```
%sql
CREATE SCHEMA IF NOT EXISTS Bootcamp.project
COMMENT 'Schema for analytics tables';

%sql
CREATE VOLUME Bootcamp.project.volumepro;
```

Upload your raw data to the volume.



Upload files to volume

Add files in any format of up to 5GB to a non-tabular dataset managed in Unity Catalog

Drop one or more files here, or browse

Upload up to 100 files (max size 5GB per file)

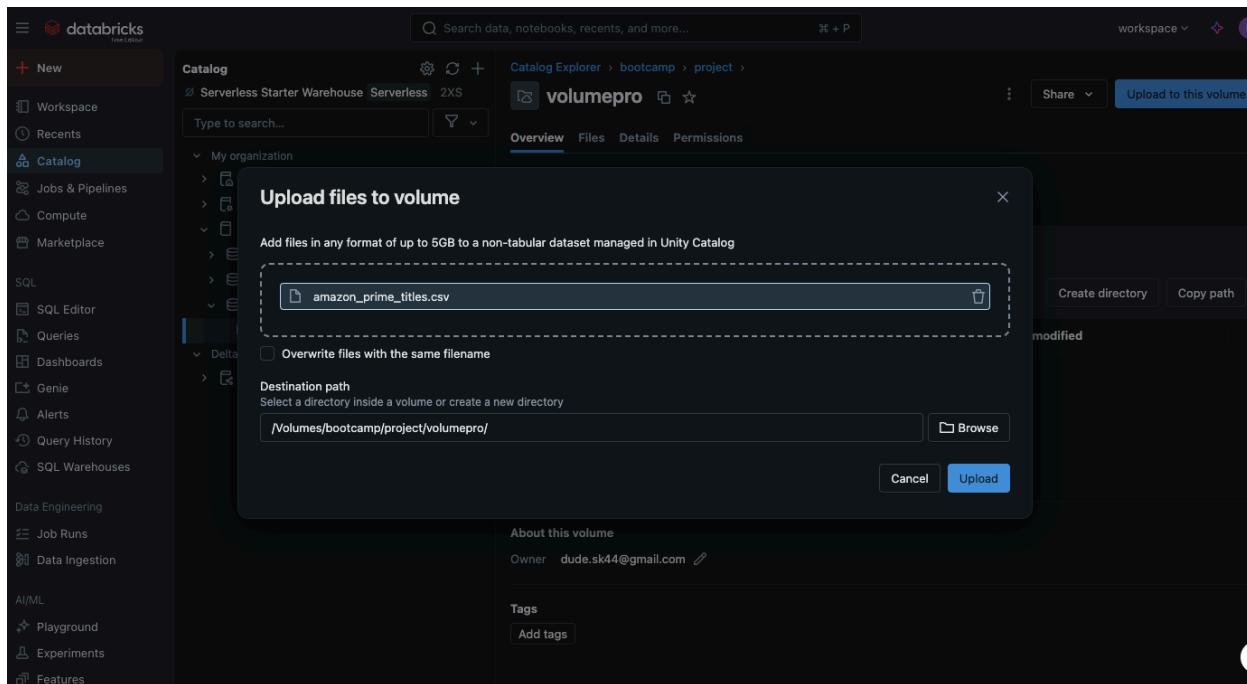
Destination path

Select a directory inside a volume or create a new directory

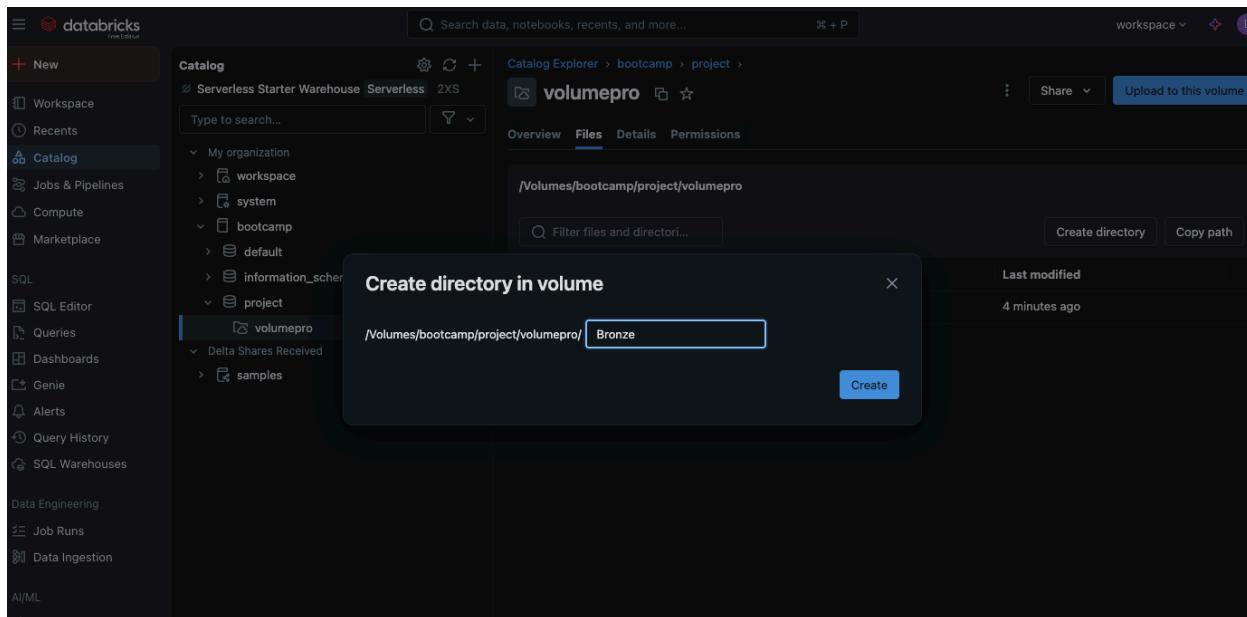
/Volumes/bootcamp/project/volumepro/

Owner: dude.sk44@gmail.com

Cancel Upload



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



3 Directories created.

The screenshot shows the Databricks Catalog Explorer interface. On the left sidebar, under the 'Catalog' section, there are three new directories: 'workspace', 'system', and 'bootcamp'. The 'bootcamp' directory is expanded, showing its sub-directories: 'default', 'information_schema', 'project', 'Tables (1)', 'Volumes (1)', and 'volumeopro'. The 'volumeopro' directory is also expanded, showing its sub-directories: 'Bronze', 'Gold', and 'Silver'. The main pane displays the contents of the 'volumeopro' directory, which contains a single file named 'amazon_prime_titles.csv'.

Data Ingestion

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

The screenshot shows a Databricks notebook cell with Python code. The code reads a CSV file named 'amazon_prime_titles.csv' located in the 'volumeopro' directory. It then displays the resulting DataFrame using the 'df.display()' command. The resulting DataFrame is shown below, containing 15 rows of movie information. The columns include show_id, type, title, director, and cast.

	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	Mahe什 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, Beyonc�, 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 Par�, 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-Andr� Ter Stegen, Sergi Robe
15	s15	Movie	Elon Musk: The Real Life Iron Man	Sonia Anderson	Elon Musk, Per Wimmer, Julie Anderson-Ankenbrandt, Ca

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

```
Writing the data into the bronze layer

▶ ✓ Yesterday (3s) 5
df.write.mode("overwrite").parquet("/Volumes/bootcamp/project/volumepro/Bronze")
> See performance (1) Optimize
```

Data Cleaning and Transformation

Loading the parquet file from the bronze layer.

```
✓ Yesterday (3s) 7
df_parquet = spark.read.parquet("/Volumes/bootcamp/project/volumepro/Bronze")
df_parquet.show()

> See performance (1) Optimize
+---+---+---+---+---+---+---+---+
| s5025| Movie| Good Newwz| Raj Mehta|Akshay Kumar, Kar...| India| NULL| 2019| 13+| 131 min| Comedy, Drama, In...|Two cou
ples with ...
| s5026| Movie| Chikati Kathalu| Siripuram Rajesh|Kalyani ( Lead ),...| NULL| NULL| 2021| 16+| 28 min| Drama|One of
the friend...
| s5027| Movie| Cheating Hearts| Eurika Pratts| Laila Odom| NULL| NULL| 2012| 16+| 69 min| Drama|A seduc
tive roman...
| s5028| Movie| Burning| Lee Chang-dong|Ah-in YOO, Steven...| NULL| NULL| 2018| AGES_16_| 148 min| Drama, Suspense|An intr
overted yo...
| s5029| Movie| Broil| Edward Drake|Jonathan Lipnicki...| NULL| NULL| 2020| 16+| 91 min| Horror|After t
roubled 17...
| s5030| Movie| American Heretics|Jeanine Isabel Bu...| NULL| NULL| NULL| 2021| 13+| 87 min| Documentary|America
n Heretics...
| s5031| Movie| All Babes Want To...| Colin Miller|Colin Miller, Gia...| NULL| NULL| 2006| PG-13| 81 min| Comedy|A Marti
al Artist...
| s5032| Movie| A Dog's Courage| Sung-yoon Lee|Eric Roland, Patr...| NULL| NULL| 2020| 7+| 102 min| Action, Kids|Jacob,
a feisty a...
| s5033| TV Show|2020 Best of Anim...| NULL| NULL| NULL| 2020| 13+| 1 Season| Animation|Award w
inning and...
+---+---+---+---+---+---+---+---+
only showing top 20 rows
```

Displaying the parquet file in table format.

The screenshot shows a Jupyter Notebook cell with the following code:

```
▶ ✓ Yesterday (2s)
display(df_parquet)
> See performance (1)
```

The output displays a table with 15 rows of movie data:

	show_id	type	title	director	cast
1	s5014	Movie	Nizhal	Appu N. Bhattachari	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 Kaushik
5	s5018	Movie	Loli Paradicka	Richard Stavarsky, Vito Stavarsky	Michal Ilkanin, Kamila Mitrášová
6	s5019	Movie	Loaded Dice	Matt Green	Tom Savini, Derek Reynolds
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 Sturridge
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 Advani
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

At the bottom of the cell, it says "4,677+ rows | Truncated data | 1.55s runtime". On the right, it says "Refreshed yesterday".

Analysing columns and its data types.

The screenshot shows a Jupyter Notebook cell with the following code:

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

The output shows the schema of the DataFrame:

```
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.

```
▶ ✓ Yesterday (1s) 12
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)

> [See performance ()] Optimize
▶ df_clean_null: pyspark.sql.connect.DataFrame = [show_id: string, type: string ... 10 more fields]
▶ df_clean_v1: pyspark.sql.connect.DataFrame = [show_id: string, type: string ... 10 more fields]

Table + 


|   | show_id | type  | title                                     | director                          | cast                                                           |
|---|---------|-------|-------------------------------------------|-----------------------------------|----------------------------------------------------------------|
| 1 | s5014   | Movie | Nizhal                                    | Appu N. Bhattachari               | 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 Kaushil  |
| 5 | s5018   | Movie | Loli Paradicka                            | Richard Stavarsky, Vito Stavarsky | 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.

```
▶ ✓ Yesterday (2s) 13
Python ⚡ 🔍 ⚙️ ⚡
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)

> [See performance ()] Optimize
▶ df_clean: pyspark.sql.connect.DataFrame = [show_id: string, type: string ... 10 more fields]
▶ df_clean_v2: pyspark.sql.connect.DataFrame = [show_id: string, type: string ... 10 more fields]

Table + 


|    | listed_in                             | description                                                                                                                         | duration |
|----|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|----------|
| 1  | spense                                | > John Baby, who is recuperating 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 stoop at an abandoned mansion, The mansion isn't just haunted, but it's not abandone...       | 86       |


```

Casting the duration and release year column

```
▶  ✓ Yesterday (2s) 16 Python ⚡ ⏹ ⏷
cols_to_cast = {
    "duration": "int",
    "release_year": "int"
}

df_clean_v4 = df_clean_v2 # start with original DataFrame

for col, dtype in cols_to_cast.items():
    df_clean_v4 = df_clean_v4.withColumn(col, F.col(col).try_cast(dtype)) # update each time

df_clean_v4.printSchema()
display(df_clean_v4)

> [See performance (1)] Optimize
> df_clean_v4: pyspark.sql.connect.DataFrame = [show_id: string, type: string ... 10 more fields]
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: date (nullable = false)
|-- release_year: integer (nullable = true)
|-- rating: string (nullable = true)
```

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

```
▶  ✓ Yesterday (1s) 17 Optimize
string_cols = [c.name for c in df_clean_v4.schema.fields if c.dataType == "string"]

df_clean_v5 = df_clean_v4.na.fill("unknown", subset=string_cols)
display(df_clean_v5)

> [See performance (1)]
> df_clean_v5: pyspark.sql.connect.DataFrame = [show_id: string, type: string ... 10 more fields]

Table +
```

	show_id	type	title	director	cast
1	s5014	Movie	Nizhal	Appu N. Bhattachari	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 Kaushik
5	s5018	Movie	Loli Paradicca	Richard Stavarsky, Vito Stavarsky	Michal Ilkanin, Kamila Mitrášová
6	s5019	Movie	Loaded Dice	Matt Green	Tom Savini, Derek Reynolds
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 Sturridge
10	s5023	Movie	Jiang Ziya	Teng Cheng	Christopher Sabat, Luci Christian
11	s5024	Movie	Uttama Villain	Ivan Yeshchenko	unknown

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)

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

Table +
```

	show_id	type	title	director	cast
1	s5014	Movie	Nizhal	Appu N. Bhattachari	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 Kaush
5	s5018	Movie	Loli Paradicka	Richard Stavarsky, Vito Stavarsky	Michal Ilkanin, Kamila Mitrášová
6	s5019	Movie	Loaded Dice	Matt Green	Tom Savini, Derek Reynolds
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 Clafin, 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

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)

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

Table +
```

	show_id	type	title	director	cast
1	s5014	Movie	Nizhal	Appu N. Bhattachari	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 Kaush
5	s5018	Movie	Loli Paradicka	Richard Stavarsky, Vito Stavarsky	Michal Ilkanin, Kamila Mitrášová
6	s5019	Movie	Loaded Dice	Matt Green	Tom Savini, Derek Reynolds
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 Clafin, 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.

```
▶ ✓ Yesterday (1s) 20
values_list = [row["rating"] for row in df_clean_v7.select("rating").distinct().collect()]
print(values_list)

> See performance \(1\) Optimize
['TV-14', 'UNRATED', 'AGES_16_', '18+', 'NOT RATE', '13+', 'TV-Y', 'TV-PG', 'TV-Y7', 'unknown', 'G', 'TV-MA', '7+', 'NR', '16+', 'R', 'ALL_AGES', 'NC-17', 'PG', 'ALL', 'TV-NR', 'AGES_18_', 'TV-G', 'PG-13', '16']
```

Creating a new `is_adult` and provide the values into it according to the condition.

Yesterday (2s) 23 Python

```
df_clean_v9 = df_clean_v7.withColumn("isAdult", F.when(F.col("rating").isin('TV-Y', 'TV-Y7', 'G', '7+', '13+', 'TV-PG', 'ALL_AGES', 'AGES_16_+', 'TV-14', 'PG-13'), "no") .otherwise("yes"))  
  
#df_clean_v9.show(truncate=False)  
display(df_clean_v9)  
  
> !ll See performance (1) Optimize  
> df_clean_v9: pyspark.sql.connect.DataFrame = [show_id: string, type: string ... 11 more fields]  
  
Table +  


|   | A <sub>c</sub> description                                                                                                          | A <sub>c</sub> duration                                                                                                      | A <sub>c</sub> isAdult |     |
|---|-------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|------------------------|-----|
| 1 | > John Baby, who is recuperating from a traumatic accident, meets Nitin, a young boy who interests him with murder stories. Whe...  | 124                                                                                                                          | yes                    |     |
| 2 | Life in the village.                                                                                                                | 85                                                                                                                           | yes                    |     |
| 3 | > When teen gamer Sarah finds an "easter egg" and accidentally opens a portal into her favorite side-scroller, she becomes tra...   | 89                                                                                                                           | no                     |     |
| 4 | > After being abandoned by his lover, Anshu, Poorna takes to alcohol and is forced to marry his neighbor, Sravani. However, he ...  | 151                                                                                                                          | no                     |     |
| 5 | > Milan, a fairground salesman, discovers that Veronika has stolen one of his caramel cakes; since she was hungry, he takes pit...  | 88                                                                                                                           | no                     |     |
| 6 | > 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                                                                                                                           | yes                    |     |
| 7 | nd 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                     | yes |
| 8 | > 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                                                                                                                          | no                     |     |


```

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

```
▶ ✓ Yesterday (2s) 24
# Path inside the volume
volume_path = "/Volumes/bootcamp/project/volumepro/Silver/silver_delta_table"

# Save dataframe as Delta
df_clean_v9.write.format("delta").mode("overwrite").save(volume_path)

> See performance \(?\) Optimizer
```

Data Loading and Analysis

Loading the delta table from the silver layer

```
▶ ✓ 23 hours ago (3s) 26
df_silver = spark.read.format("delta").load("/Volumes/bootcamp/project/volumepro/Silver/silver_delta_table")
df_silver.createOrReplaceTempView("mydelta_view")
> [See performance (1)] Optimize
▶ df_silver: pyspark.sql.connect.DataFrame = [show_id: string, type: string ... 11 more fields]
```

viewing the data inside the delta table

```
▶ ✓ Yesterday (6s) 27
%sql
SELECT *
FROM delta.`/Volumes/bootcamp/project/volumepro/Silver/silver_delta_table`;

> [See performance (1)] Optimize
```

Table	+ A _c show_id	A _c type	A _c title	A _c director	A _c cast
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 Fot
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
16	s16	Movie	Summer '03	Becca Gleason	> Joey King, Jack Klimek, Andrea Savage, Paul Scheer, Ji

```
▶ ✓ Yesterday (3s) 28
%sql
desc formatted delta.`/Volumes/bootcamp/project/volumepro/Silver/silver_delta_table`;
> [See performance (1)] Optimize
```

Table	+ A _c col_name	A _c data_type	A _c comment
12	duration	int	null
13	isAdult	string	null
14			
15	# Delta Statistics Columns		
16	Column Names	duration, description, release_year, show_id, country, listed_in, cast, rating, date_added, director, isAdult, title, type	
17	Column Selection Method	first-32	
18			
19	# Detailed Table Information		
20	Catalog	workspace	
21	Database	delta	
22	Table	/Volumes/bootcamp/project/volumepro/Silver/silver_delta_table	
23	Type	MANAGED	
24	Location	/Volumes/bootcamp/project/volumepro/Silver/silver_delta_table	
25	Provider	delta	
26	Table Properties	> [delta.enableDeletionVectors=true,delta.feature.appendOnly=true,delta.feature.deletionVectors=true,delta.feature...	

↓ 26 rows | 3.45s runtime Refreshed yesterday

Creating scd type target delta table in the gold layer

```
▶ ✓ 2 days ago (2s) 29
%sql
USE CATALOG bootcamp;
use schema project;
> [!]: See performance (2)

▶ ✓ Yesterday (3s) 30
%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;
```

```
▶ ✓ Yesterday (3s) 31
%sql
desc formatted delta.`/Volumes/bootcamp/project/volumepro/Gold/ScdtypeAmazontarget`
> [!]: See performance ()
```

Optimize

Table	+		
13	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...	

Refreshed yesterday

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

Creating hash column in the table.

```
32
from pyspark.sql.functions import *
df_gold_hash=df_silver.withColumn("src_hash",crc32(concat(*df_silver.columns)))
display(df_gold_hash)
> [See performance (1)]
df_gold_hash: pyspark.sql.connect.DataFrame = [show_id: string, type: string ... 12 more fields]
```

The screenshot shows a Jupyter Notebook cell with the following code:

```
from pyspark.sql.functions import *
df_gold_hash=df_silver.withColumn("src_hash",crc32(concat(*df_silver.columns)))
display(df_gold_hash)
```

The output of the code is displayed below the cell, showing a table with 15 rows. The table has columns: description, duration, isAdult, and src_hash. The description column contains movie plots, and the src_hash column contains numerical values.

	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	1786153607
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	717479155

Reading the scd type table from the gold layet

```
33
df_target_scd = spark.read.format("delta").load("/Volumes/bootcamp/project/volumepro/Gold/ScdtypeAmazontarget")
df_target_scd.createOrReplaceTempView("mydelta_view")
display(df_target_scd)
> [See performance (2)]
df_target_scd: pyspark.sql.connect.DataFrame = [show_id: string, type: string ... 16 more fields]
```

The screenshot shows a Jupyter Notebook cell with the following code:

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

The output of the code is displayed below the cell, showing a table with 16 columns. The message "No rows returned" is displayed.

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.

The screenshot shows a Jupyter Notebook cell with the following code:

```
▶ ✓ 23 hours ago (4s) 34
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)
> See performance (1)
df_compare: pyspark.sql.connect.DataFrame = [show_id: string, type: string ... 12 more fields]
```

The DataFrame has the following schema:

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

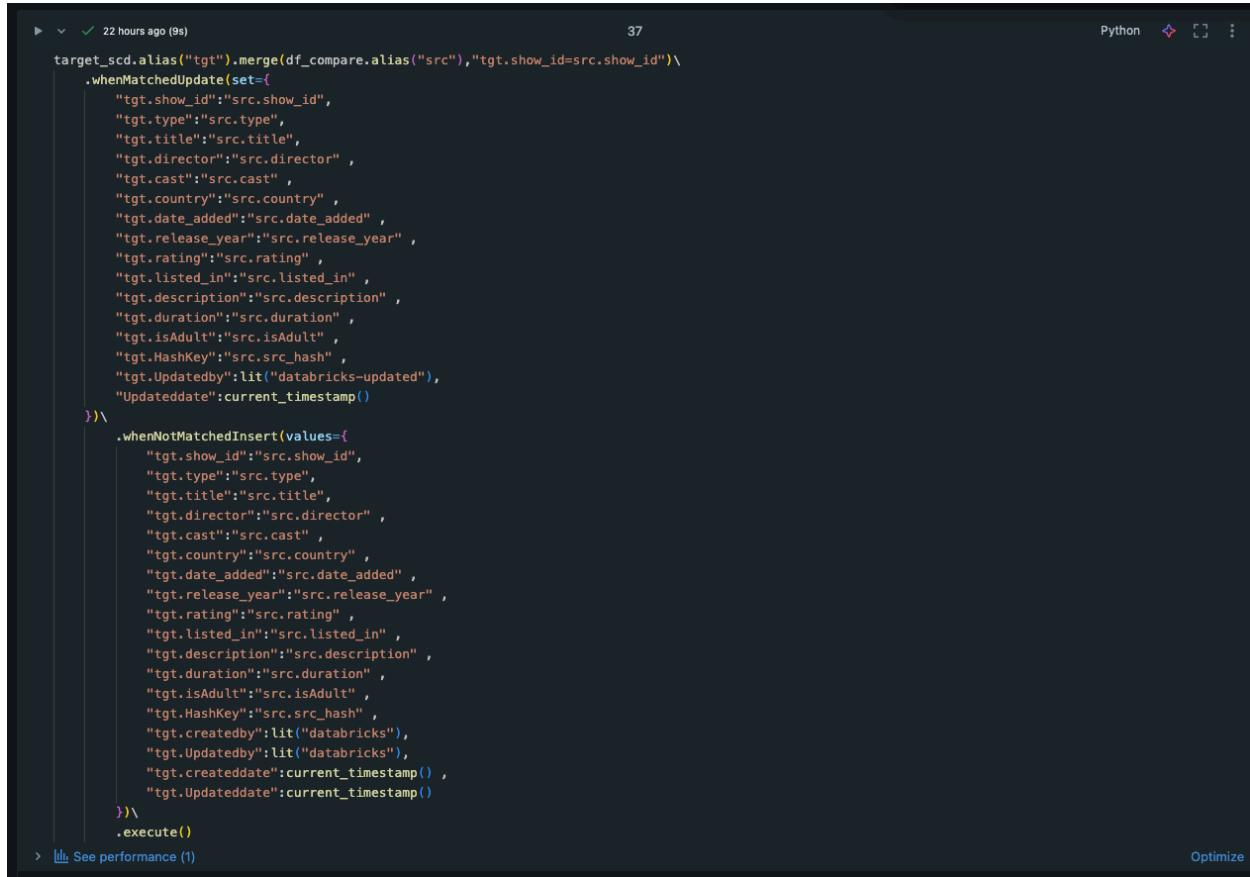
At the bottom of the cell, it says "4,320+ rows | Truncated data | 3.52s runtime".

Loading scd type in delta format from the gold layer.

The screenshot shows a Jupyter Notebook cell with the following code:

```
▶ ✓ 22 hours ago (<1s) 35
from delta.tables import DeltaTable
target_scd=DeltaTable.forName(spark,"/Volumes/bootcamp/project/volumepro/Gold/ScdtypeAmazonTarget")
```

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



The screenshot shows a Databricks notebook cell with the following Python code:

```
target_scd.alias("tgt").merge(df_compare.alias("src"), "tgt.show_id=src.show_id")\
    .whenMatchedUpdate(set={
        "tgt.show_id": "src.show_id",
        "tgt.type": "src.type",
        "tgt.title": "src.title",
        "tgt.director": "src.director",
        "tgt.cast": "src.cast",
        "tgt.country": "src.country",
        "tgt.date_added": "src.date_added",
        "tgt.release_year": "src.release_year",
        "tgt.rating": "src.rating",
        "tgt.listed_in": "src.listed_in",
        "tgt.description": "src.description",
        "tgt.duration": "src.duration",
        "tgt.isAdult": "src.isAdult",
        "tgt.HashKey": "src.src_hash",
        "tgt.Updatedby": lit("databricks-updated"),
        "tgt.Updateddate": current_timestamp()
    })\
    .whenNotMatchedInsert(values={
        "tgt.show_id": "src.show_id",
        "tgt.type": "src.type",
        "tgt.title": "src.title",
        "tgt.director": "src.director",
        "tgt.cast": "src.cast",
        "tgt.country": "src.country",
        "tgt.date_added": "src.date_added",
        "tgt.release_year": "src.release_year",
        "tgt.rating": "src.rating",
        "tgt.listed_in": "src.listed_in",
        "tgt.description": "src.description",
        "tgt.duration": "src.duration",
        "tgt.isAdult": "src.isAdult",
        "tgt.HashKey": "src.src_hash",
        "tgt.Createdby": lit("databricks"),
        "tgt.Updatedby": lit("databricks"),
        "tgt.Createddate": current_timestamp(),
        "tgt.Updateddate": current_timestamp()
    })\
    .execute()
```

The code implements SCD Type-1 logic by merging the source DataFrame (src) into the target DataFrame (tgt). It handles matched updates and new inserts. The target columns are mapped from the source columns using the provided logic. The code uses Databricks' built-in functions like `lit` and `current_timestamp`.

Checking the inserted values in the scd type table.

22 hours ago (7s) 38

```
%sql
select * from delta.`/Volumes/bootcamp/project/volumepro/Gold/ScdtypeAmazontarget`
```

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

Table +

	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	Maheesh 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, Miche
7	s7	Movie	Hired Gun	Fran Strine	> Alice Cooper, Liberty DeVitto, Ray Parker Jr., David Fot
8	s8	Movie	Grease Live!	Thomas Kail, Alex Rudzinski	> 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

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

01:29 PM (8m) 1

```
df_silver_analysis = spark.read.format("delta").load("/Volumes/bootcamp/project/volumepro/Silver/silver_delta_table")
```

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

12:13 AM (13s) 2

```
from pyspark.sql.functions import count

counts = df_silver_analysis.groupBy("type").agg(count("*").alias("title_count"))
display(counts)
```

See performance (1)

counts: pyspark.sql.connect.DataFrame = [type: string, title_count: long]

Table + Visualization 1

	type	title_count
1	TV Show	1854
2	Movie	7814

2 rows | 13.43s runtime Refreshed 18 hours ago

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)
> [lib] See performance (2)

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

Table +
```

	^{A_C} show_id	^{A_C} type	^{A_C} title	^{A_C} director	^{A_C} 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 Stanton
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, Adri...
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

TV shows's data frame.

```
Table +
```

	^{A_C} show_id	^{A_C} type	^{A_C} title	^{A_C} director	^{A_C} 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, Frances 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	Wotaku: 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	WwwrGirl	unknown	unknown

↓ 1,854 rows | 12.29s runtime Refreshed 15 hours ago

Top 5 movies with their durations.

```
▶ ✓ 12:26 AM (1s) 4
from pyspark.sql.functions import col
top5_movies = df_movies.orderBy(col("duration").desc()).limit(5)
# display(top5_movies)

display(top5_movies.select("title", "type", "duration"))

> [See performance (1)] Optimize
▶ top5_movies: pyspark.sql.connect.DataFrame = [show_id: string, type: string ... 11 more fields]

Table + 
-----+
# title      type duration
1 Soothing Surf at Del Norte for Sleep black scre... Movie 601
2 Himalayan Singing Bowls 9 hours Movie 550
3 Gentle Midnight Rain black screen 9 hours Movie 541
4 Midnight Thunderstorm for Sleep 9 Hours Movie 541
5 Gentle evening rain Movie 541
-----+
↓ 5 rows | 1.50s runtime Refreshed 17 hours ago
```

Top 5 tv shows with there duration.

```
▶ ✓ 12:37 AM (2s) 5
from pyspark.sql.functions import col
top5_tvshows = df_tvshows.orderBy(col("duration").desc()).limit(5)
# display(top5_tvshows)

display(top5_tvshows.select("title", "type", "duration"))

> [See performance (1)] Optimize
▶ top5_tvshows: pyspark.sql.connect.DataFrame = [show_id: string, type: string ... 11 more fields]

Table + 
-----+
# title      type duration
1 The Amazing Race TV Show 29
2 Silent Witness TV Show 21
3 Survivor TV Show 19
4 Top Gear (UK) TV Show 15
5 America's Next Top Model TV Show 14
-----+
↓ 5 rows | 2.14s runtime Refreshed 17 hours ago
```

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)

> [Info] See performance (2)
▶ stats_shows: pyspark.sql.connect.DataFrame = [rating: string, avg_duration: double ... 3 more fields]
▶ df_filtered_shows: pyspark.sql.connect.DataFrame = [rating: string, avg_duration: double ... 3 more fields]

Table + 
↓ 14 rows | 2.49s runtime Refreshed 17 hours ago
rating avg_duration median_duration max_duration num_titles
TV-14 1.9615384615384615 1 15 208
18+ 1.5068493150684932 1 6 146
13+ 1.3843283582089552 1 11 268
TV-Y 2.1486486486486487 1 10 74
TV-PG 2.301775147928994 1 29 169
TV-Y7 1.8461538461538463 1 6 39
unknown 1 1 1 6
TV-MA 2.155844155844156 1 21 77
7+ 1.3608247422680413 1 6 97
NR 2.2758620689655173 1 12 29
16+ 1.3781818181818182 1 8 275
ALL 1.8678571428571429 1 14 280
TV-NR 1.3904761904761904 1 11 105
TV-G 1.9506172839506173 1 14 81

↓ 14 rows | 2.49s runtime Refreshed 17 hours ago
rating avg_duration median_duration max_duration num_titles
13+ 1.3843283582089552 1 11 268
16+ 1.3781818181818182 1 8 275
```

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

```
▶  ✓ 01:05 AM (3s) 7
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)
> See performance (2)

▶ stats_movies: pyspark.sql.connect.DataFrame = [rating: string, avg_duration: double ... 3 more fields]
▶ df_filtered_movies: pyspark.sql.connect.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  | 105.666666666666667 | 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.

```
▶ ✓ 03:04 AM (2s) 8
from pyspark.sql.functions import col, count, row_number
from pyspark.sql.window import Window

# Filter for desired ratings and exclude unknown countries
df_filtered_movies = df_movies.filter(
    (col("rating").isin("13+", "16+")) & (col("country") != "unknown")
)

# Count titles per country and rating
df_country_rating_counts = df_filtered_movies.groupBy("country", "rating") \
    .agg(count("*").alias("country_titles"))

# Define window partitioned by rating and ordered by count descending
window_spec = Window.partitionBy("rating").orderBy(col("country_titles").desc())

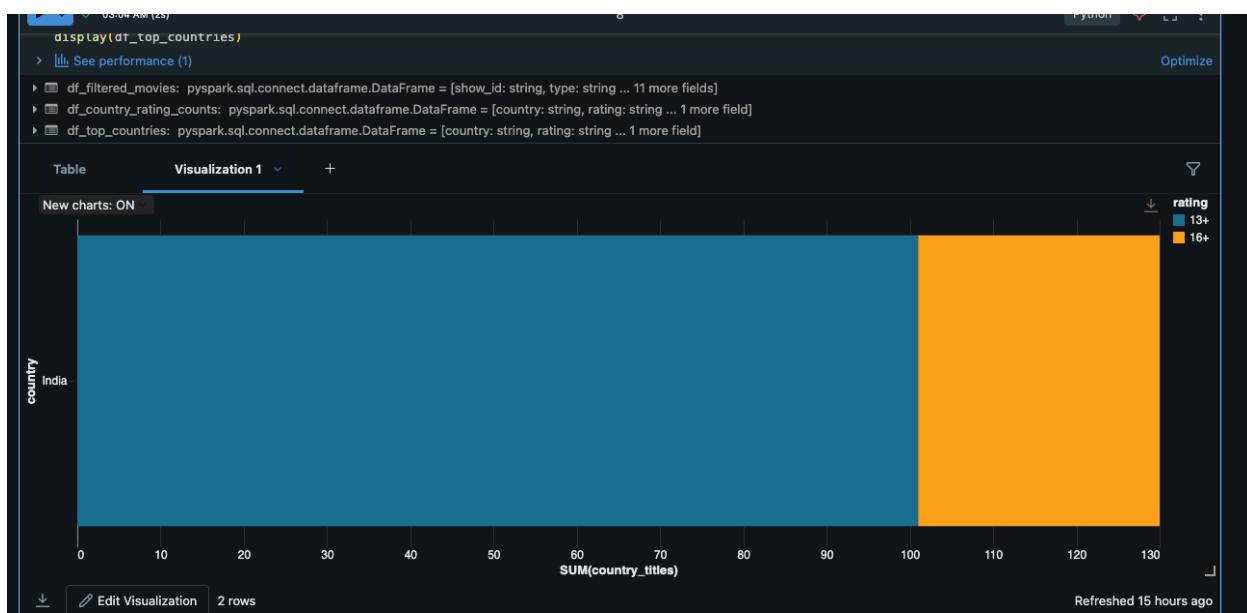
# Add row number to identify top country per rating
df_top_countries = df_country_rating_counts.withColumn("rank", row_number().over(window_spec)) \
    .filter(col("rank") == 1) \
    .drop("rank")

display(df_top_countries)
> [See performance ()] Optimize
```

df_filtered_movies: pyspark.sql.connect.DataFrame = [show_id: string, type: string ... 11 more fields]
df_country_rating_counts: pyspark.sql.connect.DataFrame = [country: string, rating: string ... 1 more field]
df_top_countries: pyspark.sql.connect.DataFrame = [country: string, rating: string ... 1 more field]

Table Visualization 1 +

	country	rating	country_titles
1	India	13+	101
2	India	16+	29



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 = [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 = [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 operation! Moving all data to a single partition, this can cause serious performance degradation.
warnings.warn(
Table Visualization 1 +
New charts: ON
400
300
200
100
0
AVG(annual_tv_shows)
release_year
1940 1950 1960 1970 1980 1990 2000 2010 2020
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.

01:45 PM (2s) 11 Python Optimize

```
from pyspark.sql.functions import *
avg_duration_df = df_silver_analysis.filter(col("rating") != "unknown").groupBy("rating").agg(avg("duration").alias("avg_duration"))

# Show result
display(avg_duration_df)
> See performance (1)
> avg_duration_df: pyspark.sql.connect.DataFrame[rating: string, avg_duration: double]
```

Table Visualization 1 +

	rating	avg_duration
1	TV-14	1.9615384615384615
2	UNRATED	92.06060606060606
3	AGES_16_	117.5
4	18+	76.75623491552695
5	NOT RATE	105.66666666666667
6	13+	86.82522437411431
7	16	89
8	TV-Y	2.1496486486486487
9	TV-PG	2.301775147928994
10	TV-Y7	1.8461538461538463
11	G	70.81720430107526
12	TV-MA	2.155844155844156
13	7+	61.79480519480519
14	NR	61.99551569506726
15	16+	75.73626373626374

24 rows | 1.65s runtime Refreshed 4 hours ago

Creating visualisation for average duration for each ratings.



Total number of titles released by each country.

12

```
filtered_df = df_silver_analysis.filter((df_silver_analysis.rating == "13+") & (df_silver_analysis.country != "unknown"))
from pyspark.sql.functions import count

count_by_country = filtered_df.groupBy("country").agg(count("title").alias("title_count"))
display(count_by_country.orderBy("title_count", ascending=False))

trend_df = filtered_df.groupBy("country", "release_year").agg(count("title").alias("yearly_count"))
# trend_df.orderBy("country", "release_year").show()
display(trend_df.orderBy("country", "release_year"))
```

See performance (2)

filtered_df: pyspark.sql.connect.DataFrame = [show_id: string, type: string ... 11 more fields]
count_by_country: pyspark.sql.connect.DataFrame = [country: string, title_count: long]
trend_df: pyspark.sql.connect.DataFrame = [country: string, release_year: integer ... 1 more field]

country	title_count
India	105
United States	38
Canada	7
United Kingdom	5
India, United States	4
Italy	4
Australia	3
Germany	2
United States, China	2
United States, France	2
United States, India	2
Spain	2
Japan	1
India, Denmark	1
Austria	1

27 rows | 2.59s runtime

Refreshed 4 hours ago

Creating visualization for the same.

