NANDINI RATHORE

BOOTCAMP PROJECT -02

Project Steps

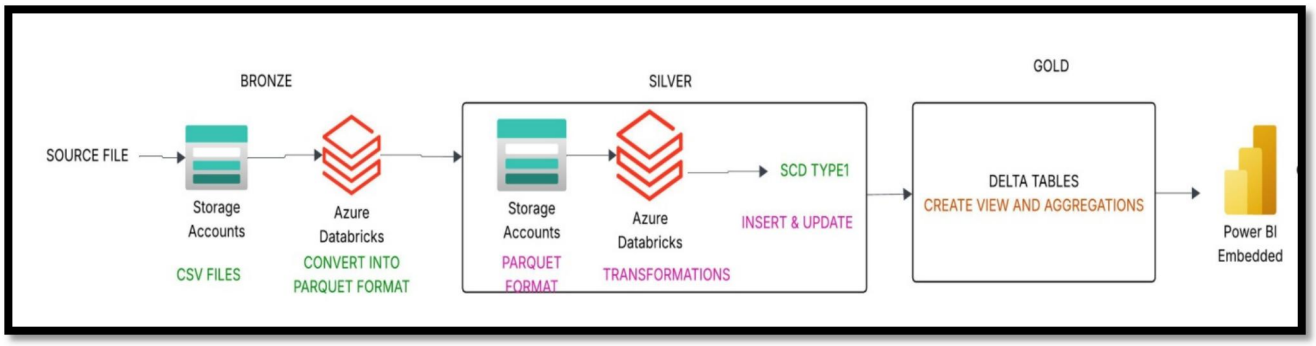
Same procedure until we store the clean data into gold layer as we did in project 1

Convert csv to parquet file

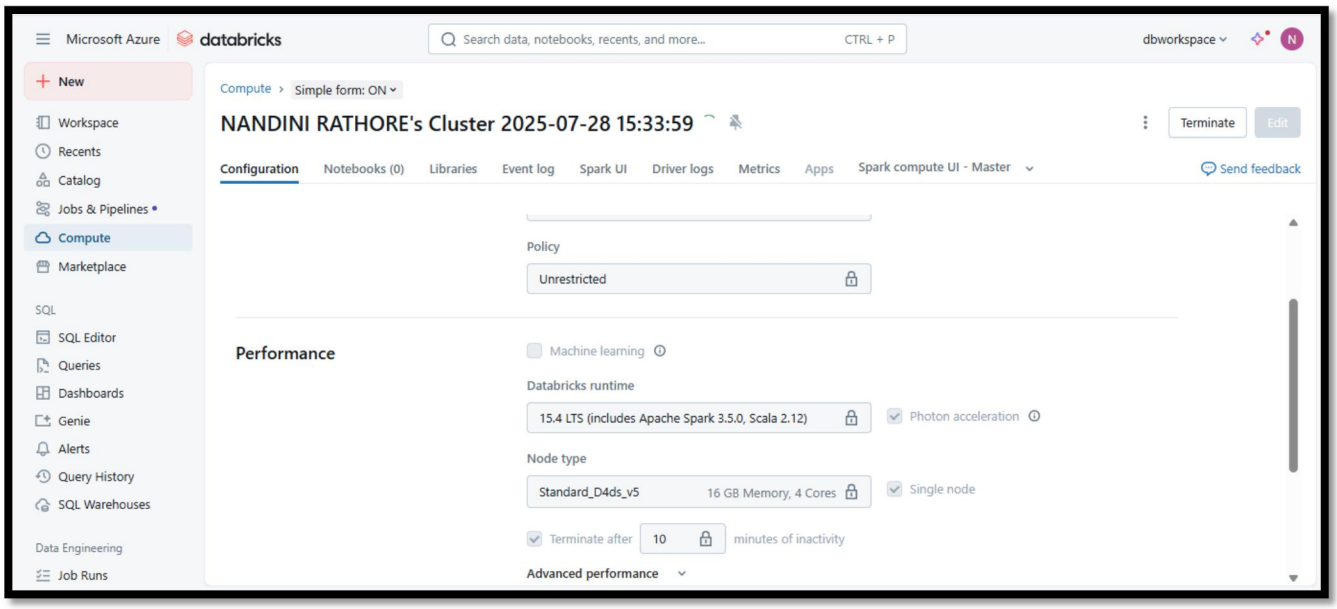
Second cleaning and doing scdtype1 and storing the data into gold layer and then followed by below some of the steps for analysis.

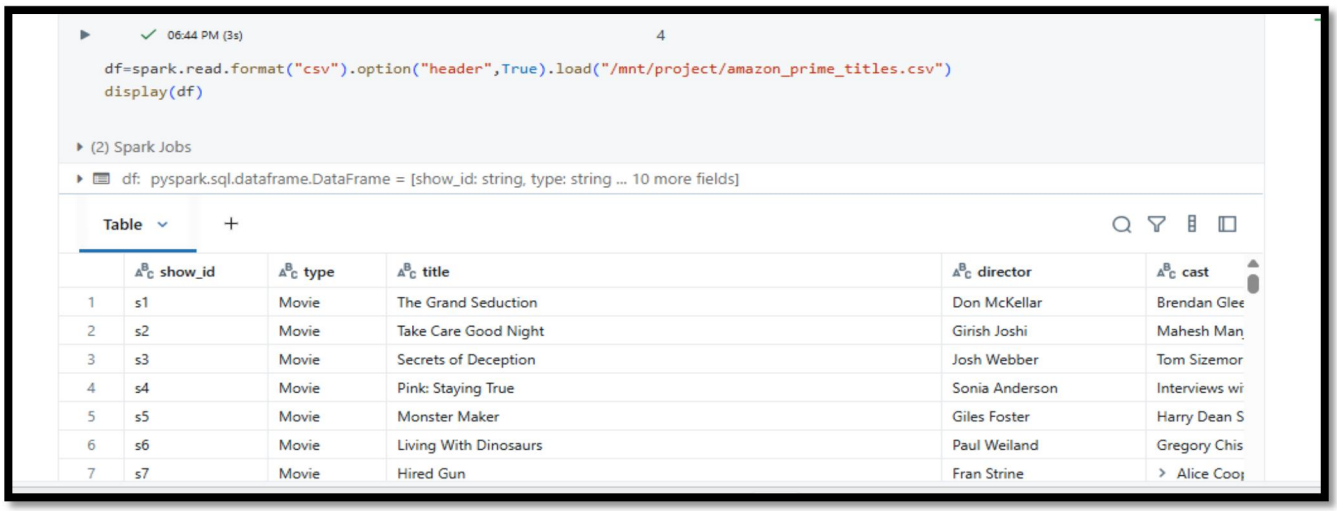
NANDINI RATHORE

ARCHITECTURE OF PROJECT:-

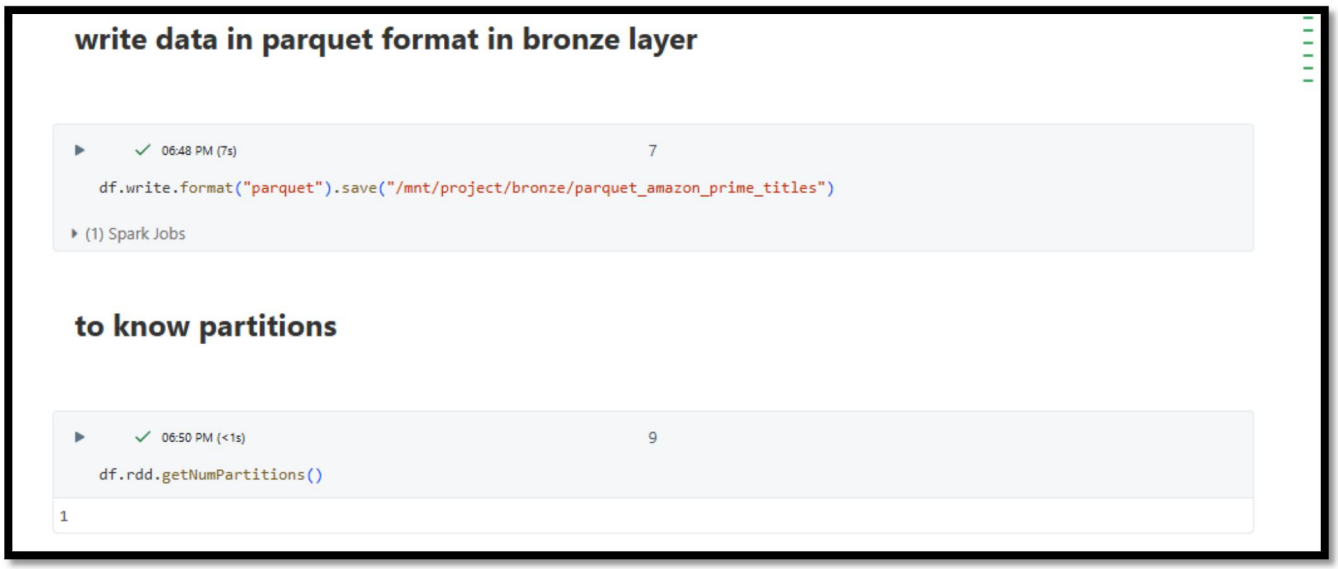


Data Ingestion (Backend Storage to Raw(Bronze) Container)



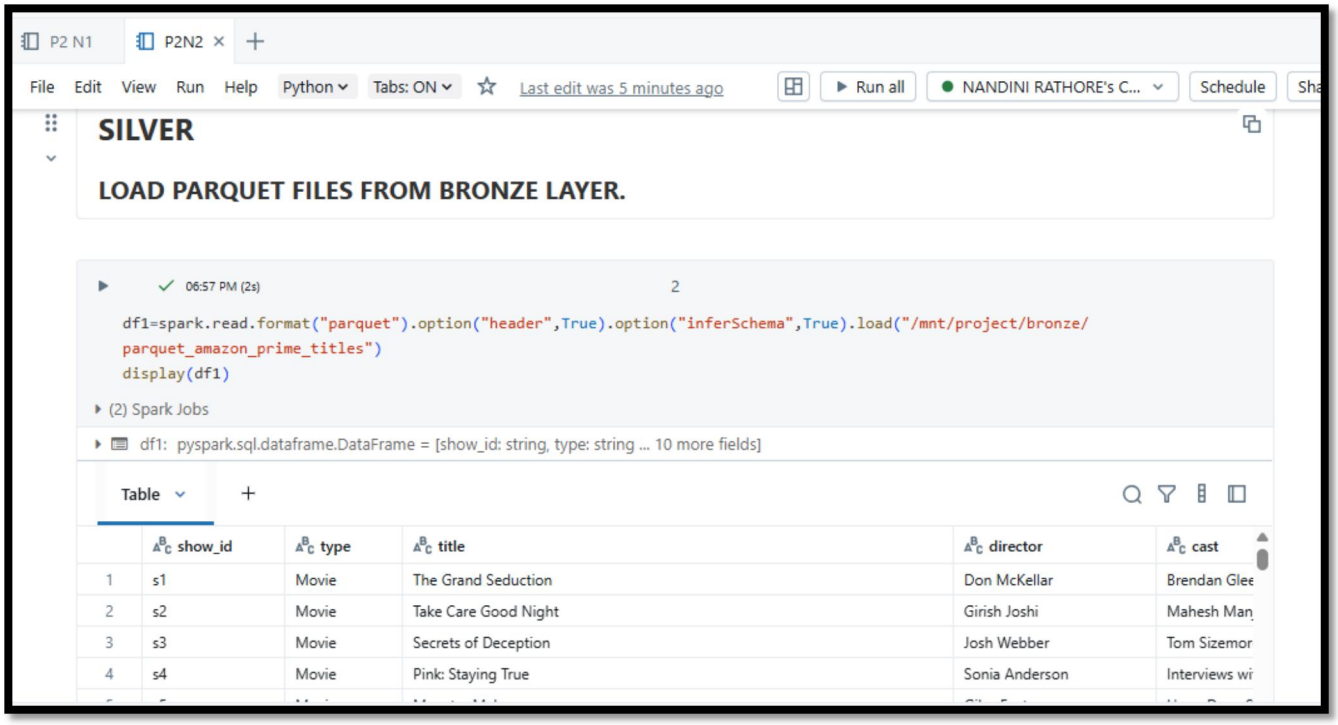


1.CONVERT RAW DATA IN PARQUET FILE FORMAT IN DATABRICKS NOTEBOOK.



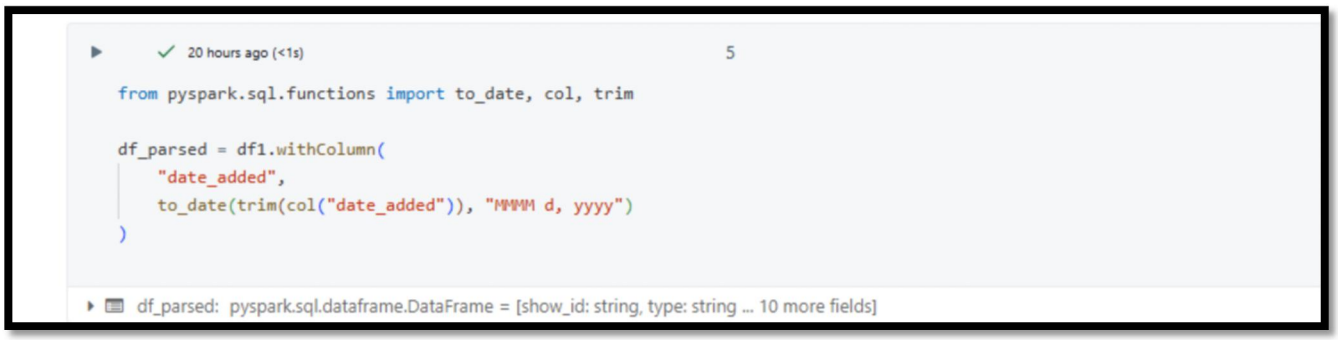
Df.rdd.getNumPartitions() shows no. of parquet file in storage accounts.

Parquet is columnar, compressed, and optimized for analytics. It reduces storage and speeds up queries compared to CSV.

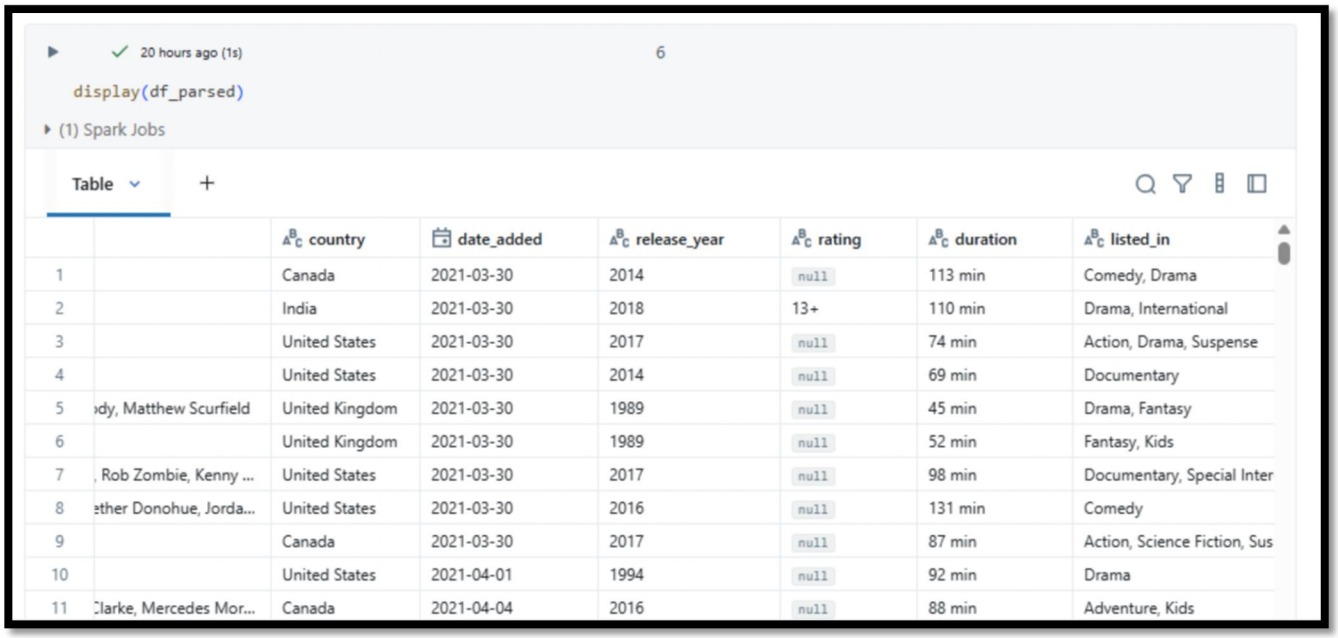


Now read that parquet file in notebook2 and done with transformations.

NANDINI RATHORE



To keep date in yyyy-mm-dd format.



NANDINI RATHORE

for duration.

```
▶   ✓ 19 hours ago (1s)                                    8

from pyspark.sql.functions import regexp_extract, col, when, trim, lower
from pyspark.sql.types import IntegerType

df1 = df_parsed.withColumn("duration", trim(col("duration")))  # remove extra spaces

df_durations = (
    df_parsed
    .withColumn(
        "duration_value",
        regexp_extract(col("duration"), r'(\d+)', 1).cast(IntegerType())
    )
    .withColumn(
        "duration_type",
        when(lower(col("duration")).like("%min%"), "minutes")
        .when(lower(col("duration")).like("%season%"), "seasons")
        .otherwise("unknown")
    )
```

To extract the numeric value only.

Regexp_extract is a pyspark function used to extract the substring from text column like 90min→90.

R'(/d+) 1→r raw string ,1 means it reminds regexp_extract to return 1 capture group.

```
# Show changes
df_durations.select("duration", "duration_value", "duration_type").show(10, False)

display(df_durations)
```

▸ (1) Spark Jobs

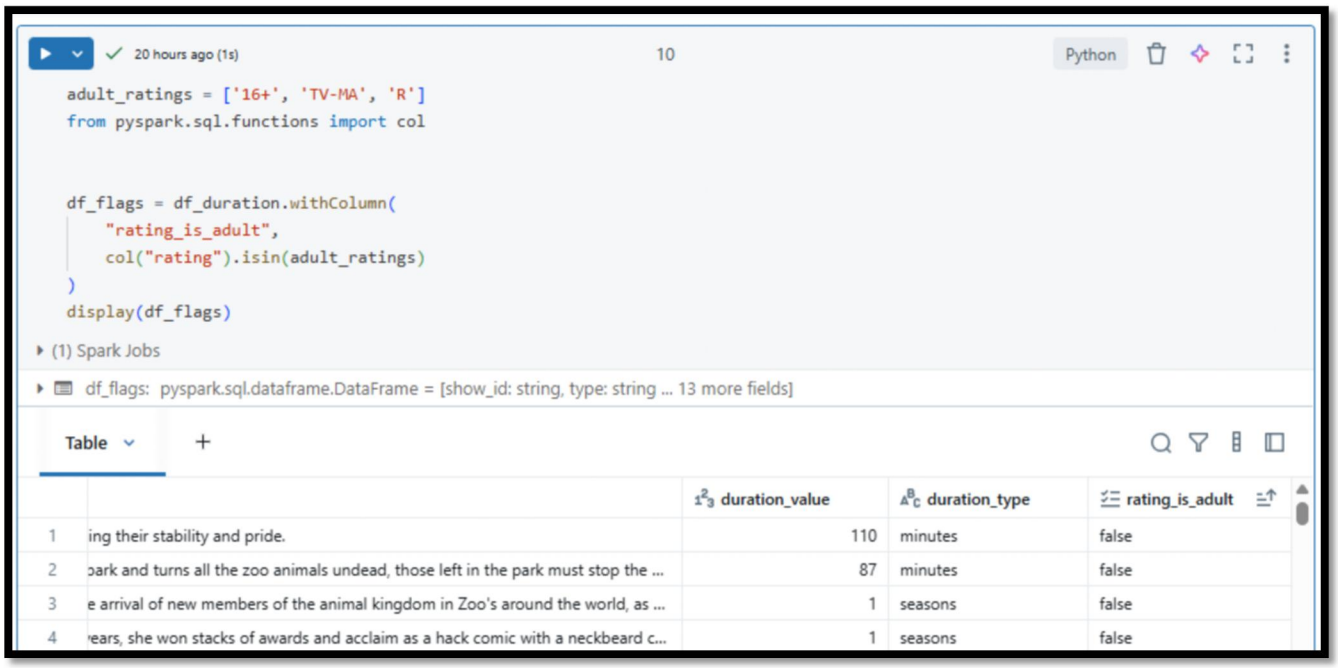
▸ df_durations: pyspark.sql.dataframe.DataFrame = [show_id: string, type: string ... 12 more fields]

▸ df1: pyspark.sql.dataframe.DataFrame = [show_id: string, type: string ... 10 more fields]

Table +

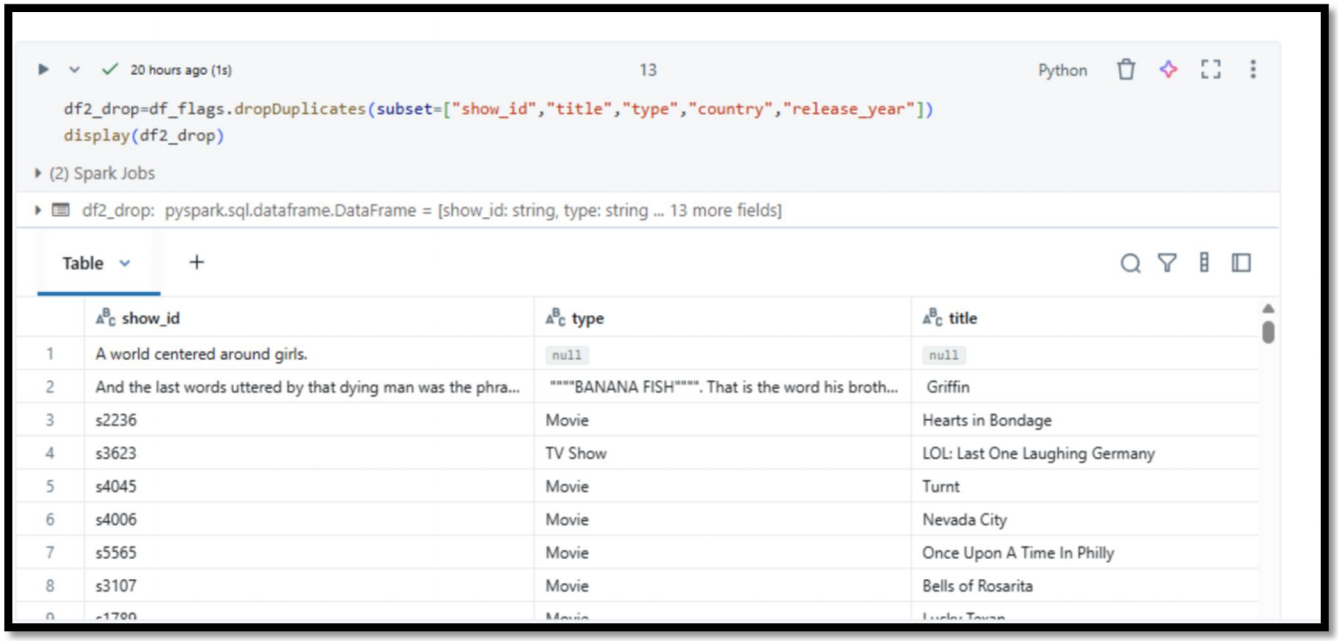
		duration_value	duration_type
1	t procure a local doctor to secure a lucrative business contract. When unlikely candidate and big city doctor...	113	minutes
2	ht a Cyber Criminal threatening their stability and pride.	110	minutes
3	is cheating on him with a neighborhood kid he goes down a furious path of self-destruction	74	minutes
4	e again, bringing her career to a new level in 2013 with a world tour that entertains unlike ever before! Get i...	69	minutes
5	nts to work with a famous but eccentric creature/special effects man named Chancey Bellows. He gets more ...	45	minutes
6	nglish seaside town, where Dom, an only child, faces the imminent arrival of a new sibling, and subsequentl...	52	minutes

NANDINI RATHORE

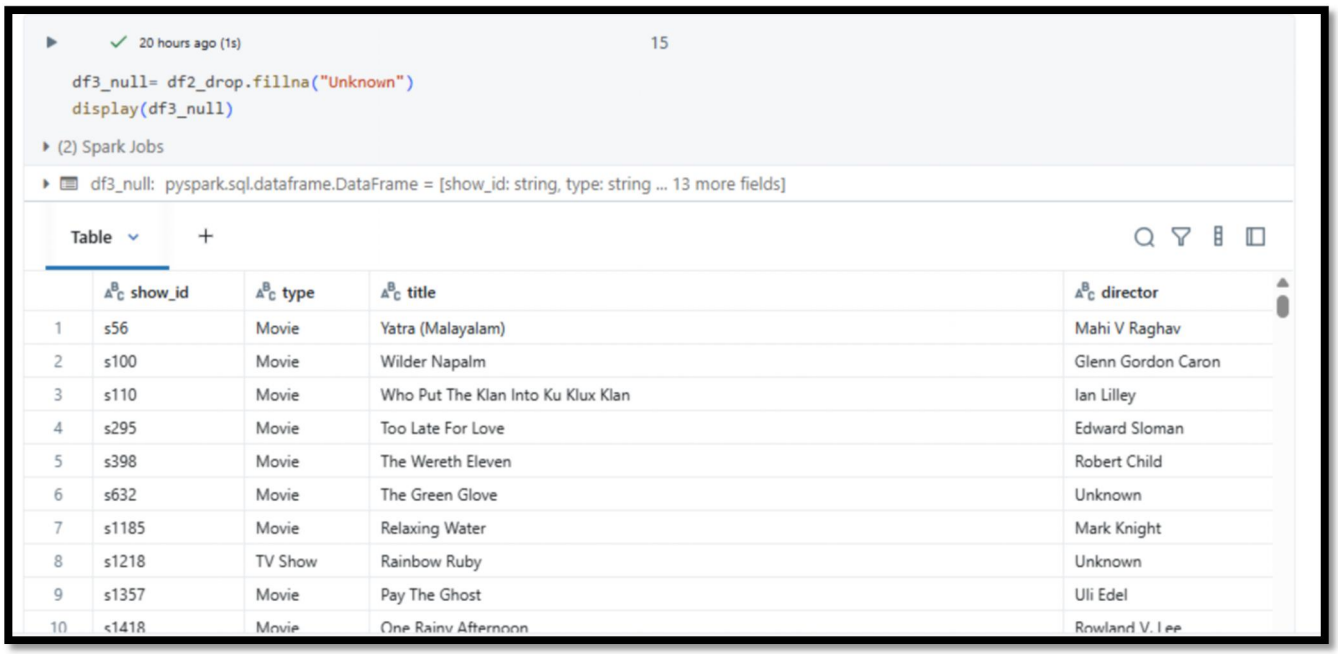


Adult=ratings=[‘16+’,’TV-MA’,’R’]

Creating new column by using with column,isin () is a function ,if the condition matches than it is true ,otherwise false.



NANDINI RATHORE



```
df3_null= df2_drop.fillna("Unknown")
display(df3_null)
```

20 hours ago (1s) 15

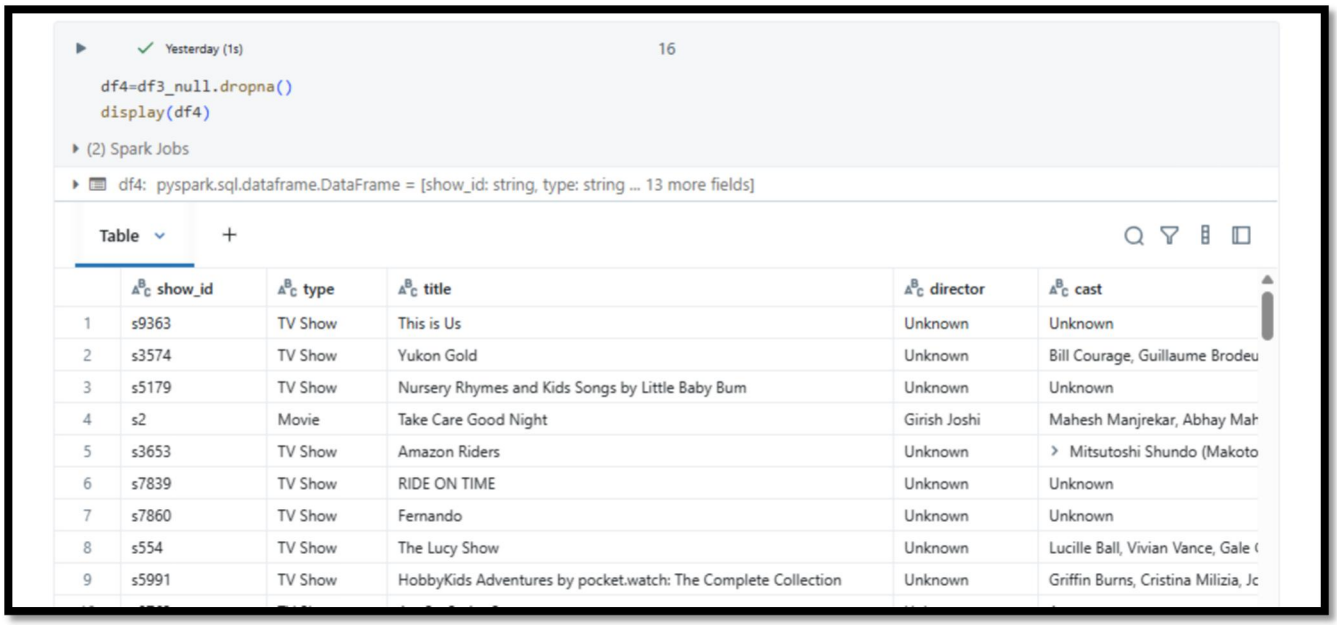
(2) Spark Jobs

df3_null: pyspark.sql.dataframe.DataFrame = [show_id: string, type: string ... 13 more fields]

Table +

	show_id	type	title	director
1	s56	Movie	Yatra (Malayalam)	Mahi V Raghav
2	s100	Movie	Wilder Napalm	Glenn Gordon Caron
3	s110	Movie	Who Put The Klan Into Ku Klux Klan	Ian Lilley
4	s295	Movie	Too Late For Love	Edward Sloman
5	s398	Movie	The Wereth Eleven	Robert Child
6	s632	Movie	The Green Glove	Unknown
7	s1185	Movie	Relaxing Water	Mark Knight
8	s1218	TV Show	Rainbow Ruby	Unknown
9	s1357	Movie	Pay The Ghost	Uli Edel
10	s1418	Movie	One Rainy Afternoon	Rowland V. Lee

Used to fill null values unknown.



```
df4=df3_null.dropna()
display(df4)
```

Yesterday (1s) 16

(2) Spark Jobs

df4: pyspark.sql.dataframe.DataFrame = [show_id: string, type: string ... 13 more fields]

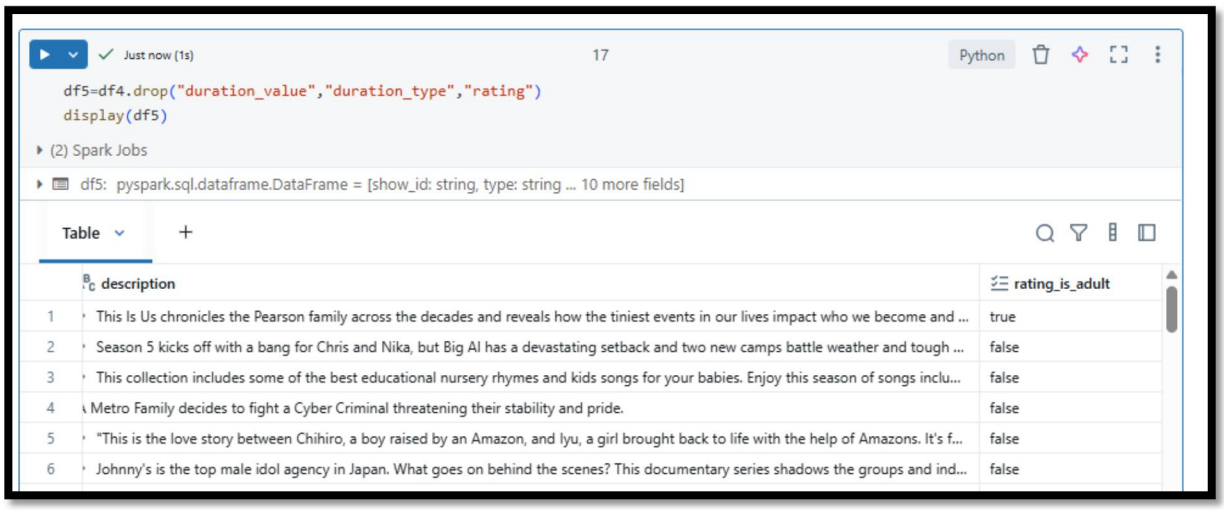
Table +

	show_id	type	title	director	cast
1	s9363	TV Show	This is Us	Unknown	Unknown
2	s3574	TV Show	Yukon Gold	Unknown	Bill Courage, Guillaume Brodeu
3	s5179	TV Show	Nursery Rhymes and Kids Songs by Little Baby Bum	Unknown	Unknown
4	s2	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mah
5	s3653	TV Show	Amazon Riders	Unknown	Mitsutoshi Shundo (Makoto
6	s7839	TV Show	RIDE ON TIME	Unknown	Unknown
7	s7860	TV Show	Fernando	Unknown	Unknown
8	s554	TV Show	The Lucy Show	Unknown	Lucille Ball, Vivian Vance, Gale
9	s5991	TV Show	HobbyKids Adventures by pocket.watch: The Complete Collection	Unknown	Griffin Burns, Cristina Milizia, Jc

During Silver transformation, you remove rows/columns that have null values (empty fields).

- Ensures data quality and avoids issues in Gold analysis.

NANDINI RATHORE



Just now (1s) 17 Python

```
df5=df4.drop("duration_value","duration_type","rating")
display(df5)
```

(2) Spark Jobs

df5: pyspark.sql.dataframe.DataFrame = [show_id: string, type: string ... 10 more fields]

Table +

	description	rating_is_adult
1	This Is Us chronicles the Pearson family across the decades and reveals how the tiniest events in our lives impact who we become and ...	true
2	Season 5 kicks off with a bang for Chris and Nika, but Big Al has a devastating setback and two new camps battle weather and tough ...	false
3	This collection includes some of the best educational nursery rhymes and kids songs for your babies. Enjoy this season of songs inclu...	false
4	Metro Family decides to fight a Cyber Criminal threatening their stability and pride.	false
5	"This is the love story between Chihiro, a boy raised by an Amazon, and Iyu, a girl brought back to life with the help of Amazons. It's f...	false
6	Johnny's is the top male idol agency in Japan. What goes on behind the scenes? This documentary series shadows the groups and ind...	false

filter out null values.

3 minutes ago (<1s) 19

```
from pyspark.sql.functions import*
```

3 minutes ago (1s) 20

```
df3_null.filter((col('show_id').isNotNull()) | (col('show_id') == ''))
display(df3_null)
```

(2) Spark Jobs

Table +

	show_id	type	title	director
1	s56	Movie	Yatra (Malayalam)	Mahi V Raghav
2	s100	Movie	Wilder Napalm	Glenn Gordon Caron

Scd typ1

NANDINI RATHORE

scd typ1

07:33 PM (1s) 32

```
%sql
create table if not exists amazonscd1(show_id varchar(10),type varchar(500), title varchar(1000), director varchar(1000), cast
varchar(1000),country varchar(100), date_added date, release_year timestamp, rating_is_adult boolean, duration int, listed_in
varchar(1000), description varchar(1000),createdby varchar(100),createddate timestamp,updatedby varchar(100),updateddate
timestamp, hashkey bigint)
using delta
location"dbfs:/user/hive/warehouse/mnt/project/amazonscd1"
```

OK

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

Creating delta target table.

07:33 PM (<1s) 33

```
%sql
select * from amazonscd1
```

_sqldf: pyspark.sql.dataframe.DataFrame = [show_id: string, type: string ... 15 more fields]

Table +

v_id	type	title	director	cast	country	date_added	release_year	rating_

No rows returned

07:33 PM (<1s) 34 SQL

```
%sql desc formatted amazonscd1
```

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

Table +

	col_name	data_type
17	hashkey	bigint
18		
19	# Detailed Table Information	
20	Catalog	spark_catalog
21	Database	default
22	Table	amazonscd1
23	Created Time	Sun Aug 17 23:33:09 UTC 2025
24	Last Access	UNKNOWN
25	Created By	Spark 3.5.0
26	Type	EXTERNAL

NANDINI RATHORE

To know the type of table , I used this command.

Creating widgets.

```
07:33 PM (<1s)                                                    35
dbutils.widgets.text("file_name"," ")   # create widgets
file_name=dbutils.widgets.get("file_name")   # return empty.
print(file_name)
```

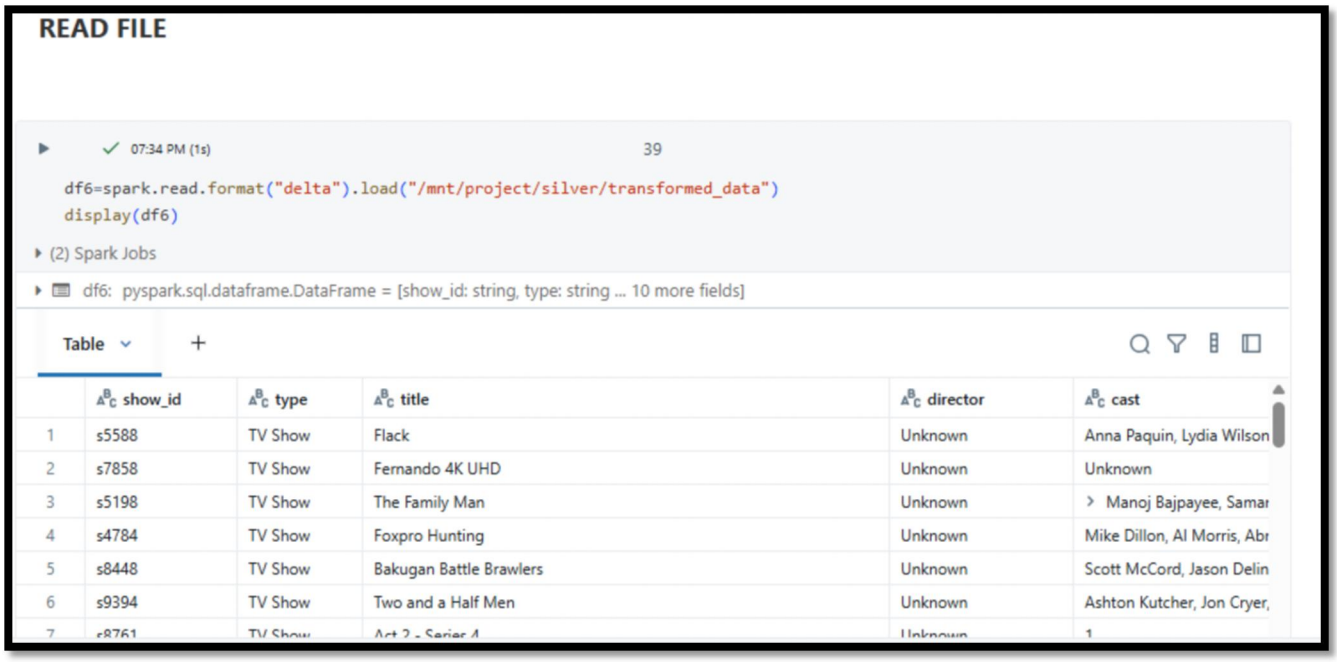
```
transformed_data
```

+ Code + Text Assistant

```
07:34 PM (<1s)                                                    36
srcpath="/mnt/project/silver/"+file_name
print(srcpath)
tgtpath="/mnt/project/amazonscd1"
```

```
/mnt/project/silver/ transformed_data
```

```
07:34 PM (<1s)                                                    37
tgtpath="/mnt/project/amazonscd1"
```



READ FILE

```
07:34 PM (1s)                                                     39
df6=spark.read.format("delta").load("/mnt/project/silver/transformed_data")
display(df6)
```

(2) Spark Jobs

df6: pyspark.sql.dataframe.DataFrame = [show_id: string, type: string ... 10 more fields]

Table +

	show_id	type	title	director	cast
1	s5588	TV Show	Flack	Unknown	Anna Paquin, Lydia Wilson
2	s7858	TV Show	Fernando 4K UHD	Unknown	Unknown
3	s5198	TV Show	The Family Man	Unknown	Manoj Bajpayee, Samar
4	s4784	TV Show	Foxpro Hunting	Unknown	Mike Dillon, Al Morris, Abr
5	s8448	TV Show	Bakugan Battle Brawlers	Unknown	Scott McCord, Jason Delin
6	s9394	TV Show	Two and a Half Men	Unknown	Ashton Kutcher, Jon Cryer,
7	s8761	TV Show	Act 2 - Series 4	Unknown	1

NANDINI RATHORE

07:34 PM (<1s) 40

```
from delta import DeltaTable
dbtable=DeltaTable.forPath(spark,"dbfs:/user/hive/warehouse/mnt/project/amazonscd1")
dbtable.toDF().show()
```

```
+-------+----+-----+--------+----+-------+----------+------------+---------------+--------+---------+-----------+---------+---------
--+---------+-----------+-------+
|show_id|type|title|director|cast|country|date_added|release_year|rating_is_adult|duration|listed_in|description|createdby|createdda
te|updatedby|updateddate|hashkey|
+-------+----+-----+--------+----+-------+----------+------------+---------------+--------+---------+-----------+---------+---------
--+---------+-----------+-------+
+-------+----+-----+--------+----+-------+----------+------------+---------------+--------+---------+-----------+---------+---------
--+---------+-----------+-------+
```

Generate Hashkey

07:35 PM (<1s) 42

```
df_hashkey=df_drp_col.withColumn("src_hashkey",crc32(concat(*df_drp_col.columns)))
display(df_hashkey)
```

(2) Spark Jobs

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

Table +

	show_id	type	title	director	cast
1	s9363	TV Show	This is Us	Unknown	Unknown
2	s3574	TV Show	Yukon Gold	Unknown	Bill Courage, Guillaume Brodeur, Ken
3	s5179	TV Show	Nursery Rhymes and Kids Songs by Little Baby Bum	Unknown	Unknown
4	s2	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan,
5	s3653	TV Show	Amazon Riders	Unknown	Mitsutoshi Shundo (Makoto Shidc
6	s7839	TV Show	RIDE ON TIME	Unknown	Unknown

Just now (1s) 43 Python

```
df_join=df_hashkey.alias("src").join(dbtable.toDF().alias("tgt"),col("src.show_id")==col("tgt.show_id"),"anti").select("src.*")
display(df_join)
```

(3) Spark Jobs

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

NANDINI RATHORE

merge

07:39 PM (4s) 45

```
dbtable.alias("tgt").merge(df_join.alias("src"),"src.show_id==tgt.show_id")\
  .whenMatchedUpdate(set={"show_id":"src.show_id",
    "type":"src.type","title":"src.title","director":"src.director","cast":"src.cast","country":"src.country",
    "date_added":"src.date_added","release_year":"src.release_year","duration":"src.duration","listed_in":"src.listed_in",
    "description":"src.description","rating_is_adult":"src.rating_is_adult","updateddate":"current_timestamp()","updatedby":lit
    ("databricks-updated"),"hashkey":"src.src_hashkey"
  })\
    .whenNotMatchedInsert(values={"show_id":"src.show_id",
    "type":"src.type","title":"src.title","director":"src.director","cast":"src.cast","country":"src.country",
    "date_added":"src.date_added","release_year":"src.release_year","duration":"src.duration","listed_in":"src.listed_in",
    "description":"src.description","rating_is_adult":"src.rating_is_adult",
    "updateddate":"current_timestamp()","updatedby":lit("databricks"),"createdby":lit("databricks"),
    "createddate":"current_timestamp()","hashkey":"src.src_hashkey"
    })\
        .execute()
```

(9) Spark Jobs

07:40 PM (<1s) 46

```
%sql
select * from amazonscd1
```

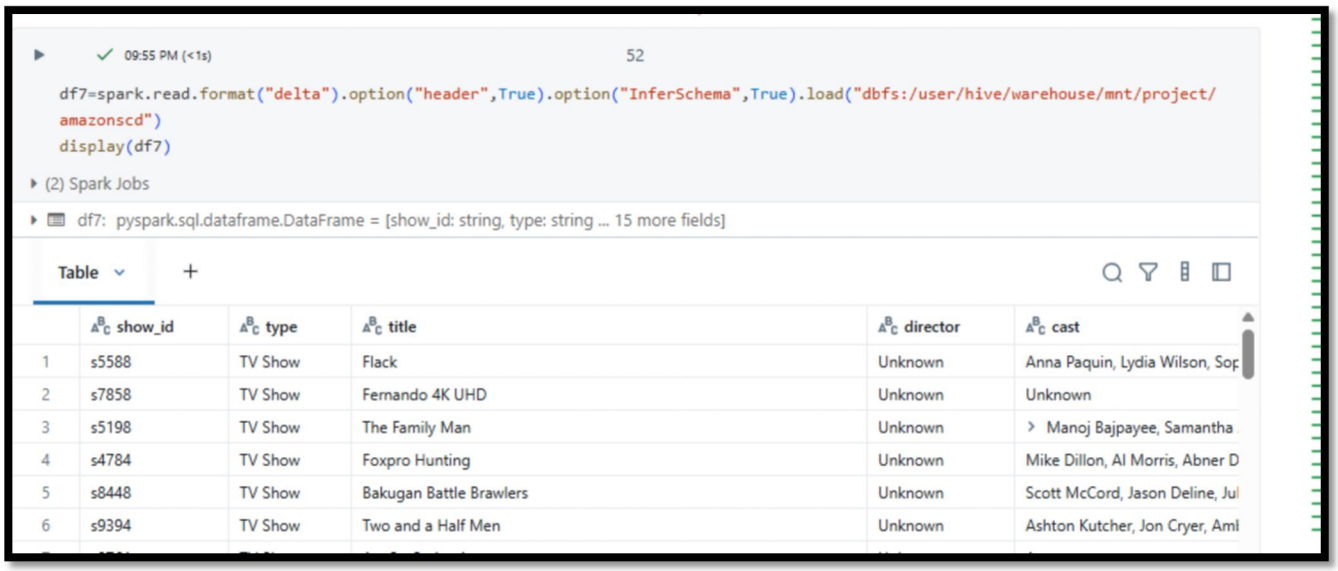
(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [show_id: string, type: string ... 15 more fields]

Table +

	show_id	type	title	director	cast
1	s5588	TV Show	Flack	Unknown	Anna Paquin, Lydia Wilson
2	s7858	TV Show	Fernando 4K UHD	Unknown	Unknown
3	s5198	TV Show	The Family Man	Unknown	Manoj Bajpayee, Samar
4	s4784	TV Show	Foxpro Hunting	Unknown	Mike Dillon, Al Morris, Abr
5	s8448	TV Show	Bakugan Battle Brawlers	Unknown	Scott McCord, Jason Delin
6	s9394	TV Show	Two and a Half Men	Unknown	Ashton Kutcher, Jon Cryer,
7	s8761	TV Show	Act 2 - Series 4	Unknown	1

NANDINI RATHORE



09:55 PM (<1s) 52

```
df7=spark.read.format("delta").option("header",True).option("InferSchema",True).load("dbfs:/user/hive/warehouse/mnt/project/amazonscd")
display(df7)
```

(2) Spark Jobs

df7: pyspark.sql.dataframe.DataFrame = [show_id: string, type: string ... 15 more fields]

Table +

	show_id	type	title	director	cast
1	s5588	TV Show	Flack	Unknown	Anna Paquin, Lydia Wilson, Sop
2	s7858	TV Show	Fernando 4K UHD	Unknown	Unknown
3	s5198	TV Show	The Family Man	Unknown	Manoj Bajpayee, Samantha
4	s4784	TV Show	Foxpro Hunting	Unknown	Mike Dillon, Al Morris, Abner D
5	s8448	TV Show	Bakugan Battle Brawlers	Unknown	Scott McCord, Jason Deline, Jul
6	s9394	TV Show	Two and a Half Men	Unknown	Ashton Kutcher, Jon Cryer, Aml

save into gold layer.

3 minutes ago (4s) 54

```
df7.write.format("delta").mode("overwrite").save("/mnt/project/amazonscd")
```

(4) Spark Jobs

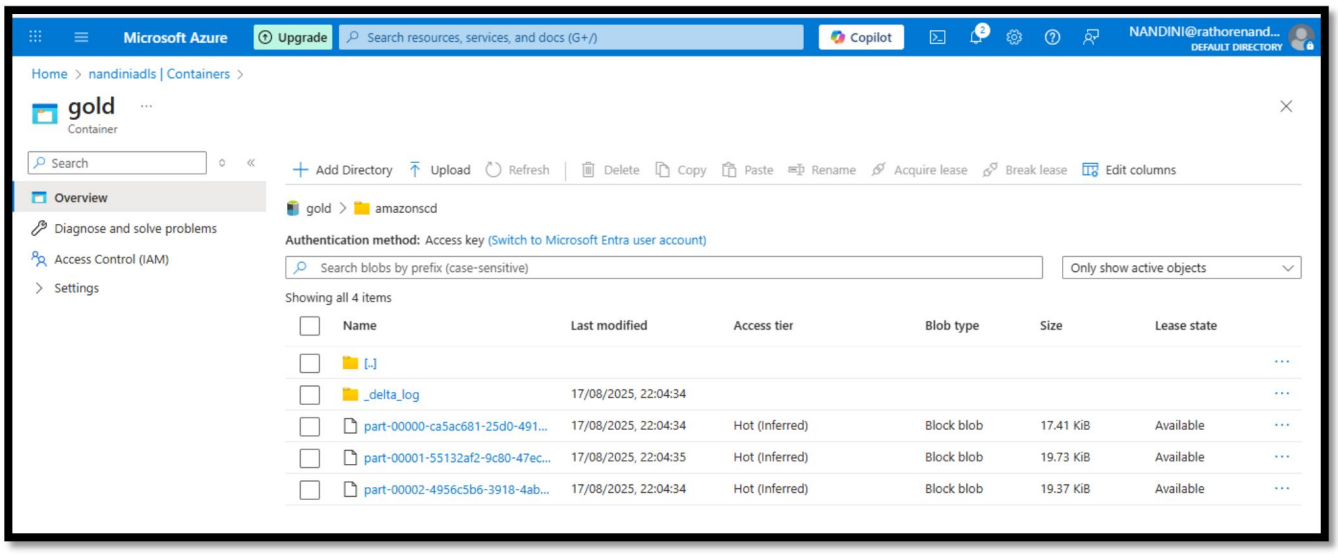
2 minutes ago (<1s) 55 Python

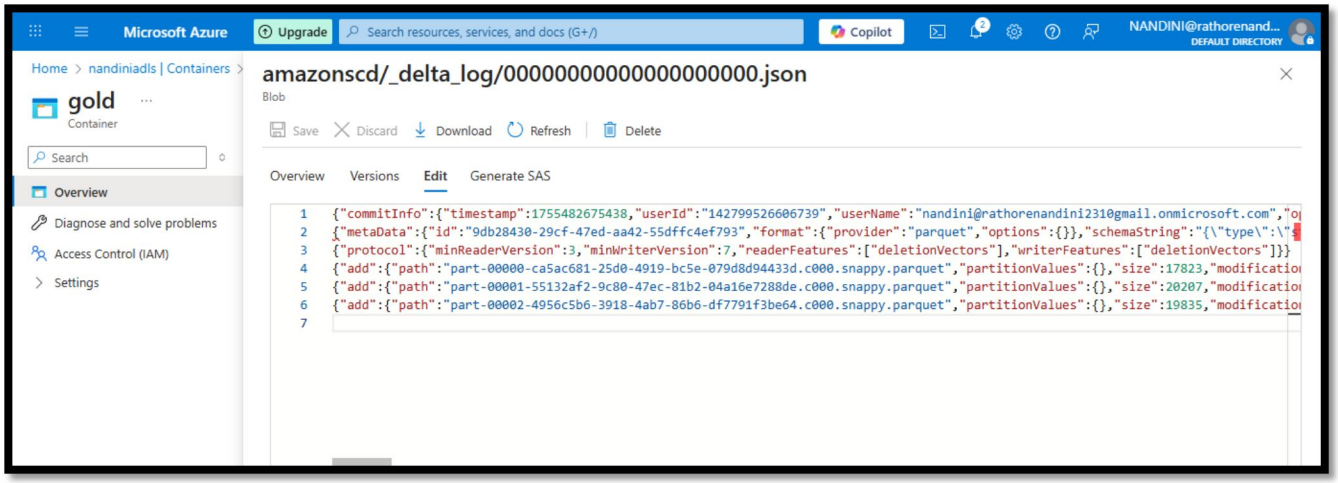
```
df7.rdd.getNumPartitions()
```

3

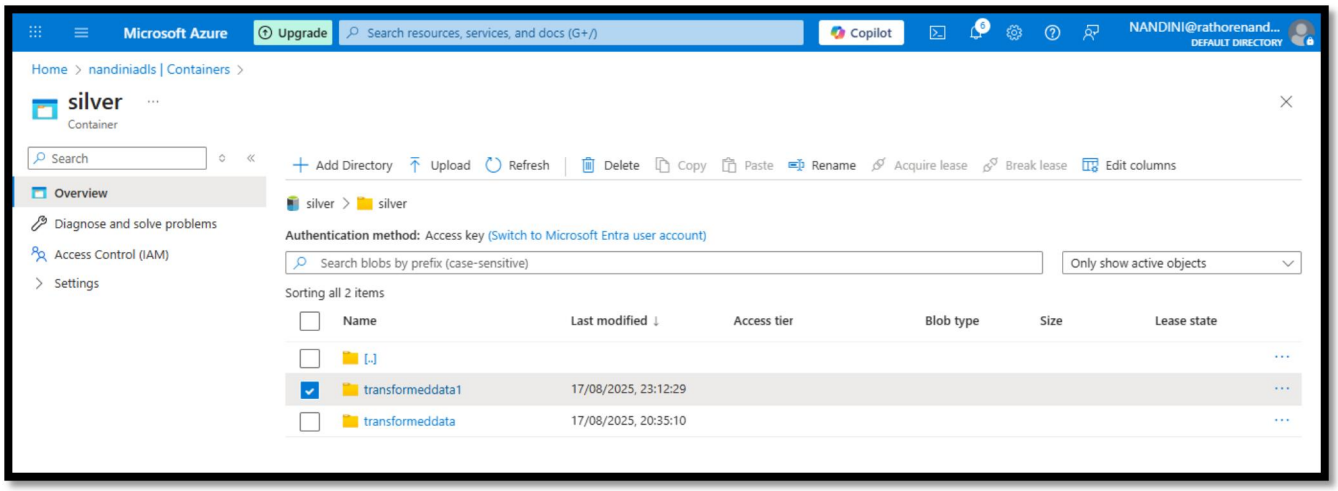
[Shift+Enter] to run and move to next cell
[Ctrl+Shift+P] to open the command palette
[Esc H] to see all keyboard shortcuts

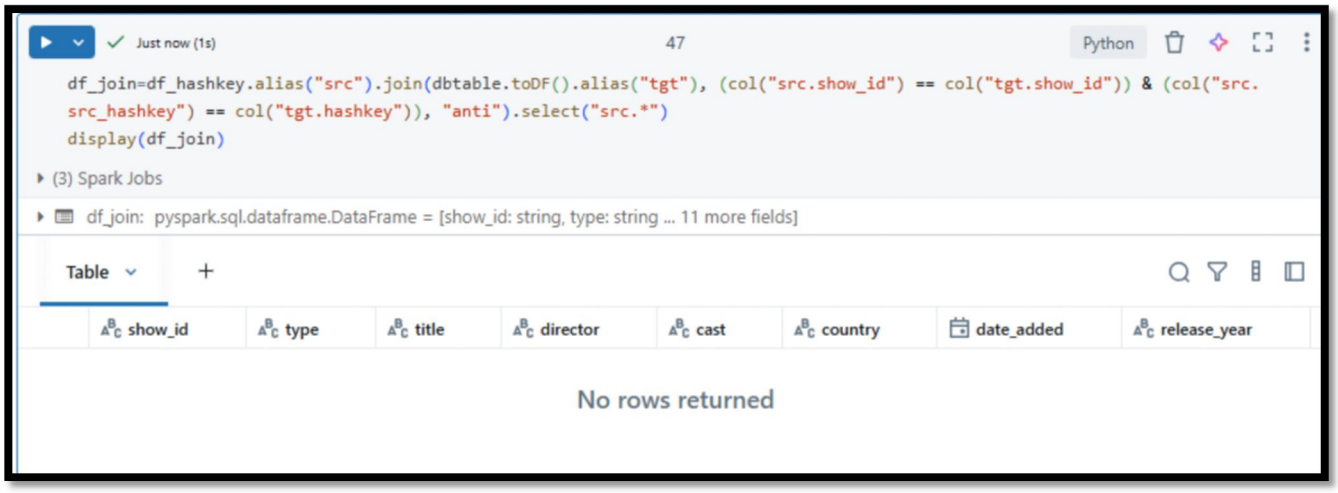
NANDINI RATHORE





Update data:→





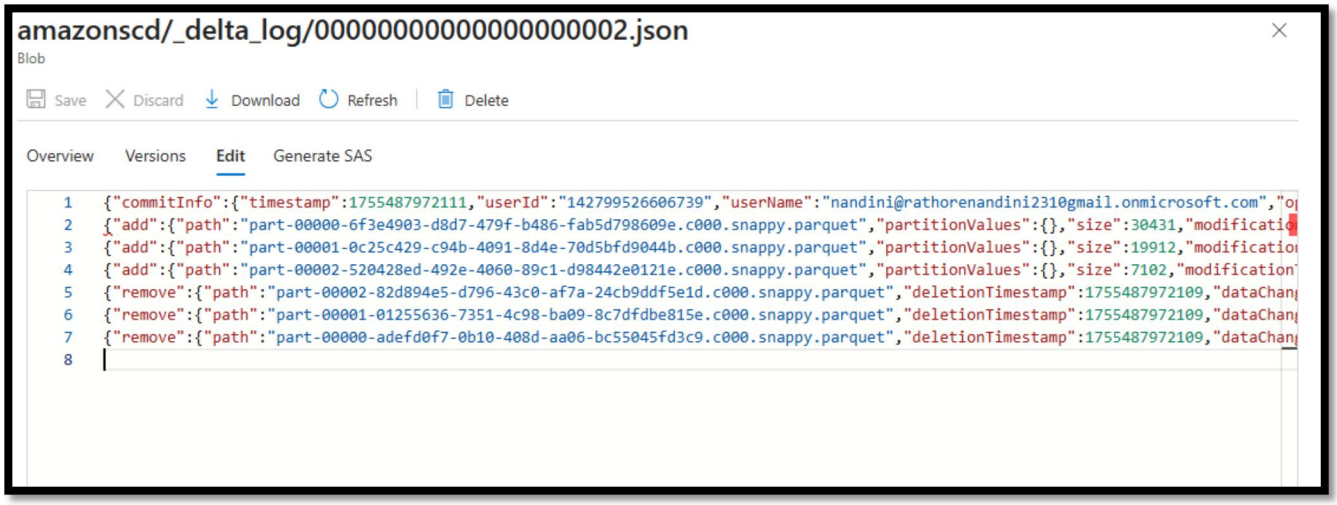
		createdby	createddate	updatedby	updateddate	hashkey
127	n more crazy celebrities...	databricks	2025-08-18T00:50:46.342+00:...	databricks	2025-08-18T00:50:46.342+00:...	932506278
128	ccessibility twist.	databricks	2025-08-18T00:50:46.342+00:...	databricks	2025-08-18T00:50:46.342+00:...	1592503650
129	. Marco is one of the las...	databricks	2025-08-18T00:50:46.342+00:...	databricks	2025-08-18T00:50:46.342+00:...	2427629804
130	kids are able to draw th...	databricks	2025-08-18T00:50:46.342+00:...	databricks	2025-08-18T00:50:46.342+00:...	2895324716
131	ween heavenly prophec...	databricks	2025-08-18T00:50:46.342+00:...	databricks	2025-08-18T00:50:46.342+00:...	548107059
132	al musical casts will deli...	databricks	2025-08-18T00:50:46.342+00:...	databricks	2025-08-18T00:50:46.342+00:...	2055276638
133	the local dragon, Jane ...	databricks	2025-08-18T00:50:46.342+00:...	databricks	2025-08-18T00:50:46.342+00:...	3888875124
134	ary voyages of scientific...	databricks	2025-08-18T00:50:46.342+00:...	databricks	2025-08-18T00:50:46.342+00:...	406954415
135		databricks	2025-08-18T00:50:46.342+00:...	databricks	2025-08-18T00:50:46.342+00:...	3372397798
136	d vehicles. Starting with ...	databricks	2025-08-18T00:50:46.342+00:...	databricks	2025-08-18T00:50:46.342+00:...	1274564948
137	m the real places menti...	databricks	2025-08-18T00:50:46.342+00:...	databricks	2025-08-18T00:50:46.342+00:...	1146893072
138	at the Blue Shamrock. I...	databricks	2025-08-18T00:50:46.342+00:...	databricks	2025-08-18T00:50:46.342+00:...	1913837364
139	strophically when she c...	databricks	2025-08-18T00:50:46.342+00:...	databricks-updated	2025-08-18T03:05:53.525+00:...	3246562303
140	n the Car Team! Every d...	databricks	2025-08-18T00:50:46.342+00:...	databricks-updated	2025-08-18T03:05:53.525+00:...	1438034224

After update in scd type1 these are the updated rows.

NANDINI RATHORE

	show_id	type
127	s7104	TV Show
128	s6060	TV Show
129	s6076	TV Show
130	s5784	TV Show
131	s8804	TV Show
132	s5040	TV Show
133	s6489	TV Show
134	s8799	TV Show
135	s8891	TV Show
136	s6593	TV Show
137	s4652	TV Show
138	s7167	TV Show
139	s5588	TV Show
140	s5005	TV Show

updated rows.



This is a **Delta Lake transaction log entry**. Each JSON file inside the _delta_log folder represents one commit to the Delta table.

- The commitInfo block tells me **what operation happened** (here it's a WRITE / OVERWRITE) and how many rows were written (140 rows).
- The add block shows the details of new Parquet files that were added to the table, including **number of records, file path, min/max statistics**.

- The remove block shows which old files were removed or replaced as part of this commit.

In short, this JSON is Delta's way of tracking **all changes** to the table so that features like **ACID transactions, time travel, and versioning** are possible.

This JSON file is part of the `_delta_log` in Delta Lake. It represents one atomic commit to the Delta table — recording the operation type (`WRITE`/`MERGE`/`DELETE`), row counts, and metadata about files added or removed. Delta uses these logs to ensure ACID transactions, schema enforcement, and time travel.

VIEWS & AGGREGATES.

Group rows by columns and aggregations.

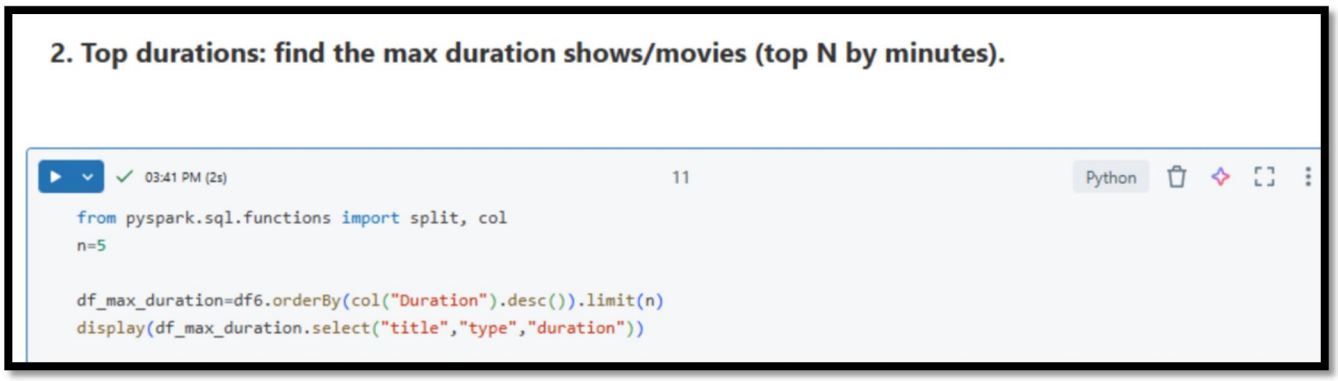
Generally , df.groupBy(“column”).count().

Creating the view and then save as delta in delta table.

I didn’t create view first , because I have to do transformations in file.if first created the view than it is not possible.

NANDINI RATHORE

2. Top durations: find the max duration shows/movies (top N by minutes).



Orderby sort data in descending and ascending orders.

Table +

	title	type	duration
1	Take Care Good Night	Movie	110
2	Top Gear (UK)	TV Show	15
3	My Magic Pet Morphle - Adventures of Mila & Morph...	TV Show	14
4	Pokémon the Series: Diamond and Pearl	TV Show	12
5	CoComelon - Kids Songs and Nursery Rhymes	TV Show	11

NANDINI RATHORE

03:47 PM (1s) 12

```
df_max_duration.createOrReplaceTempView("max_durations")
data=spark.sql("select title,type,duration from max_durations")
display(data)
```

(1) Spark Jobs

data: pyspark.sql.dataframe.DataFrame = [title: string, type: string ... 1 more field]

Table +

	title	type	duration
1	Take Care Good Night	Movie	110
2	Top Gear (UK)	TV Show	15
3	My Magic Pet Morphle - Adventures of Mila & Morph...	TV Show	14
4	Pokémon the Series: Diamond and Pearl	TV Show	12
5	CoComelon - Kids Songs and Nursery Rhymes	TV Show	11

01:47 PM (1s) 13 Python

```
df_duration=df6.groupBy("duration").count()
display(df_duration)
```

df_duration: pyspark.sql.dataframe.DataFrame = [duration: integer, count: long]

Table +

	duration	count
1	2	58
2	3	26
3	1	20
4	4	19
5	5	5
6	7	3
7	10	2
8	11	2
9	15	1
10	12	1
11	8	1

NANDINI RATHORE

```
03:58 PM (1s)                                                    17                                                    Python
df6.createOrReplaceTempView("vw_country_13_plus")
data1=spark.sql("select duration, title, type,rating_is_adult from vw_country_13_plus where rating_is_adult=False")
n=10
display(data1.limit(n))

(1) Spark Jobs
data1:  pyspark.sql.dataframe.DataFrame = [duration: integer, title: string ... 2 more fields]
```

Table +

	duration	title	type	rating_is_adult
1	2	Fernando 4K UHD	TV Show	false
2	2	The Family Man	TV Show	false
3	1	Foxpro Hunting	TV Show	false
4	1	Bakugan Battle Brawlers	TV Show	false
5	11	Two and a Half Men	TV Show	false
6	2	Act 2 - Series 4	TV Show	false
7	2	Fernando (Dubbed)	TV Show	false

Avg,median,max

```
05:03 PM (5s)                                                    23
from pyspark.sql.functions import avg, expr, max, count
df_rating_stats = df6.groupBy("rating_is_adult").agg(

    avg("duration").alias("avg_duration"),
    expr("percentile_approx(duration, 0.5)").alias("median_duration"),
    max("duration").alias("max_duration"),
    count("*").alias("num_titles")
)

# Step 4: Display results
display(df_rating_stats)

# Step 5: Create a Gold view for BI or SQL queries
df_rating_stats.createOrReplaceTempView("vw_rating_duration_stats")
df_rating_stats.write.format("delta").mode("overwrite").save("/mnt/project/vw_rating_duration_stats")

(7) Spark Jobs
```

Percentile_approx →gives the approx % of the column.

Table +

	rating_is_adult	avg_duration	median_duration	max_duration	num_titles
1	true	2.9	2	7	20
2	false	3.9916666666666667	2	110	120

NANDINI RATHORE

4.Country breakdown: for 13+ and 16+, count titles by country; identify which country has the most in each rating band.

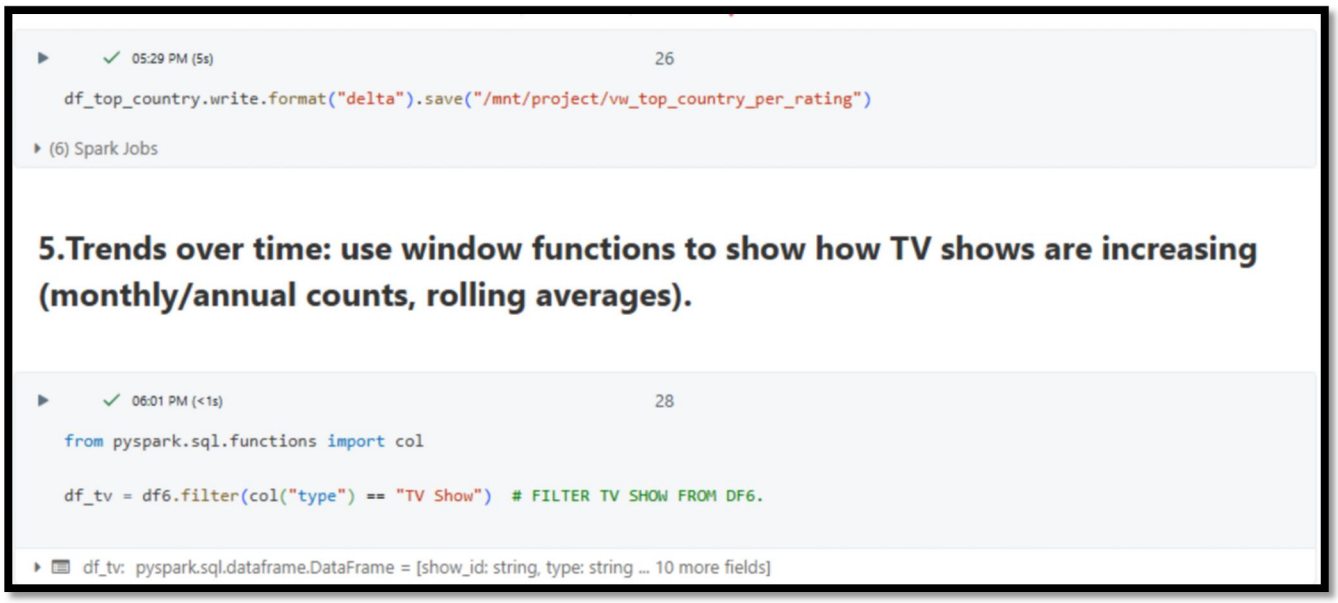
```
▶  ✓ 05:25 PM (1s)                                                    25

from pyspark.sql.window import Window
from pyspark.sql.functions import row_number, desc, count, col

# Step 1: Count titles by rating & country
df_country_count = df6.groupBy("rating_is_adult", "country").agg(count("*").alias("titles"))

# Step 2: Define a window: partition by rating, order by num_titles descending
windowSpec = Window.partitionBy("rating_is_adult").orderBy(desc("titles"))

# Step 3: Add row number in each partition
df_ranked = df_country_count.withColumn("rank", row_number().over(windowSpec))

# Step 4: Keep only top country per rating
df_top_country = df_ranked.filter(col("rank") == 1).drop("rank")

# Step 5: Create view
df_top_country.createOrReplaceTempView("vw_top_country_per_rating")
display(df_top_country)
```

Window function perform calculation on entire rows without collapsing them in one row. Windows provides the detail of each row.

Table +

	rating_is_adult	country	titles
1	false	Unknown	114
2	true	Unknown	15

NANDINI RATHORE

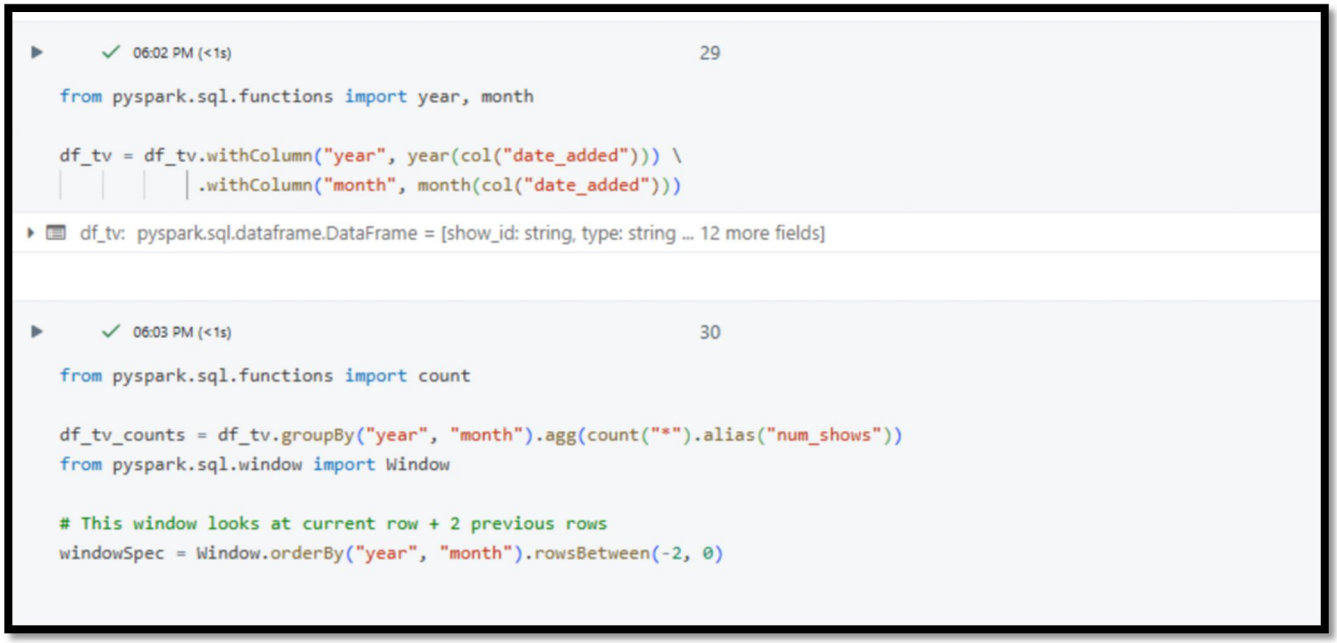


```
05:29 PM (5s)                                                    26
df_top_country.write.format("delta").save("/mnt/project/vw_top_country_per_rating")
(6) Spark Jobs
```

5.Trends over time: use window functions to show how TV shows are increasing (monthly/annual counts, rolling averages).

```
06:01 PM (<1s)                                                   28
from pyspark.sql.functions import col

df_tv = df6.filter(col("type") == "TV Show")  # FILTER TV SHOW FROM DF6.
df_tv:  pyspark.sql.dataframe.DataFrame = [show_id: string, type: string ... 10 more fields]
```

Flter the tv show from dataframe.



```
06:02 PM (<1s)                                                   29
from pyspark.sql.functions import year, month

df_tv = df_tv.withColumn("year", year(col("date_added"))) \
            .withColumn("month", month(col("date_added")))
df_tv:  pyspark.sql.dataframe.DataFrame = [show_id: string, type: string ... 12 more fields]

06:03 PM (<1s)                                                   30
from pyspark.sql.functions import count

df_tv_counts = df_tv.groupBy("year", "month").agg(count("*").alias("num_shows"))
from pyspark.sql.window import Window

# This window looks at current row + 2 previous rows
windowSpec = Window.orderBy("year", "month").rowsBetween(-2, 0)
```

Adding two new columns by using withcolumn→year,month

NANDINI RATHORE

06:03 PM (4s) 31

```
from pyspark.sql.functions import avg as spark_avg

df_tv_trends = df_tv_counts.withColumn(
    "rolling_avg_3mo", spark_avg("num_shows").over(windowSpec)
)
df_tv_trends = df_tv_trends.orderBy("year", "month")
display(df_tv_trends)
```

(3) Spark Jobs

df_tv_trends: pyspark.sql.dataframe.DataFrame = [year: integer, month: integer ... 2 more fields]

Table +

	year	month	num_shows	rolling_avg_3mo
1	2021	3	5	5
2	2021	4	10	7.5
3	2021	5	11	8.666666666666666
4	2021	6	25	15.333333333333334
5	2021	7	16	17.333333333333332
6	2021	8	24	21.666666666666668

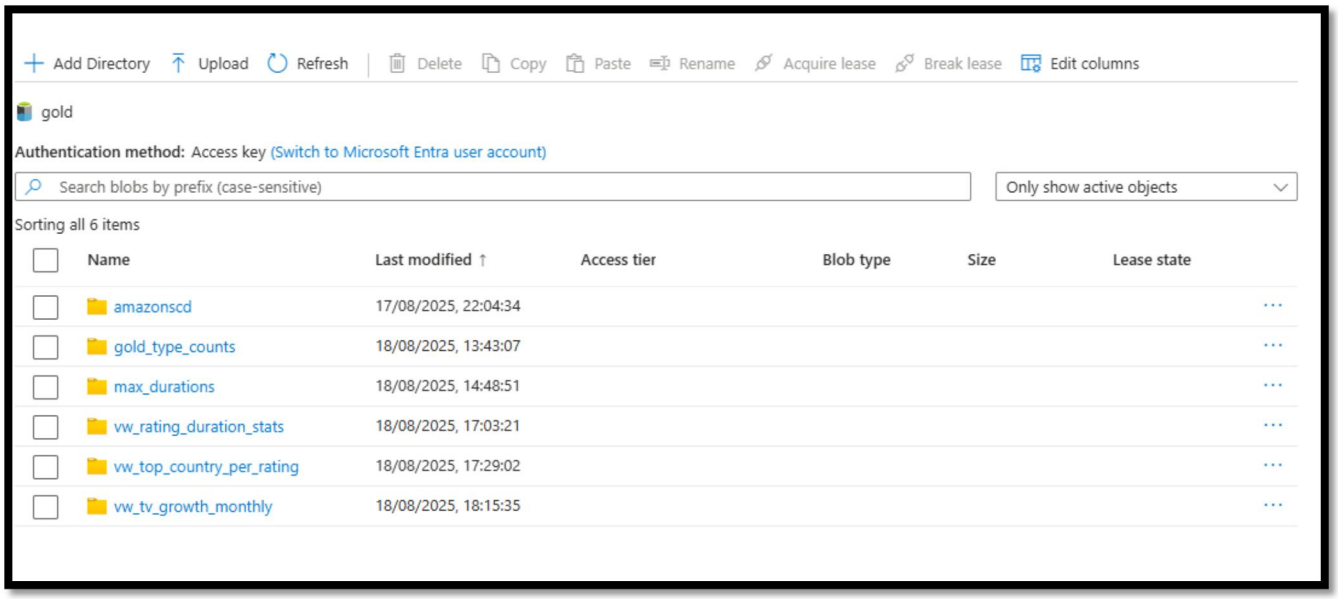
```
df_tv_trends.createOrReplaceTempView("vw_tv_growth_monthly")
data4=spark.sql("select year, month, num_shows from vw_tv_growth_monthly")
display(data4)
data4.write.format("delta").save("/mnt/project/vw_tv_growth_monthly")
```

▸ (9) Spark Jobs

▸ data4: pyspark.sql.dataframe.DataFrame = [year: integer, month: integer ... 1 more field]

Table +

	year	month	num_shows
1	2021	3	5
2	2021	4	10
3	2021	5	11
4	2021	6	25
5	2021	7	16
6	2021	8	24
7	2021	9	29
8	2021	10	19



CONCLUSION:→

In this project, I successfully built a data pipeline using **Databricks** to process raw CSV data into structured and optimized Delta tables across the **Bronze, Silver, and Gold layers**.

- In the **Bronze layer**, I ingested raw CSV files and converted them into Parquet for efficient storage.
- In the **Silver layer**, I applied **data cleaning, schema validation, duplicate removal, and SCD Type 1 logic** to maintain up-to-date records with hash keys.
- In the **Gold layer**, I performed **aggregations, window functions, and analytical transformations** (like median duration using percentile_approx, rating-based grouping, and conditional columns).

The final curated Gold data was stored in Delta format, on which I created views .

Overall, this project demonstrates my ability to design a **scalable, optimized, and production-ready data pipeline** in Databricks using **PySpark**, ensuring clean, consistent, and analysis-ready datasets.