

Task 1.2 Databricks YouFlix DB Silver

Run Scenario:

1. Before starting the run scenario, clear bronze/youflix and silver/youflix directories.

2. Delete and re-create YouFlixDB database from scratch using DeploymentScript.sql script. It is necessary to get rid of the results of your previous execution and testing activities.

3. Drop YouFlix database from your Databricks workspace.

4. Go to data lake stdimentoringdatalakexx and proceed to Storage browser, then click on Tables and edit each of entity by setting watermark value to 2000-01-01T00:00:00.00Z.

5. Run pipeline from task 1.1.
6. Run the notebook uc1_load_bronze_to_silver.ipynb.

7. Create new notebook test in Azure Databricks.

8. Run display(dbutils.fs.ls("/mnt/data/silver/youflix")) in â€œtestâ€ notebook and take screenshot(s) of the results.

test

Python

☆

File

Edit

View

Run

Help

Last edit was now

► Run all

● sn-cluster-ed

Schedule

Table

+

🔍

🔼

📄

	path	name	size	modificationTime
1	dbfs:/mnt/data/silver/youflix/youflix_device/	youflix_device/	0	
2	dbfs:/mnt/data/silver/youflix/youflix_subscription/	youflix_subscription/	0	
3	dbfs:/mnt/data/silver/youflix/youflix_user/	youflix_user/	0	
4	dbfs:/mnt/data/silver/youflix/youflix_user_subscription_devic...	youflix_user_subscription_devic...	0	

↓

4 rows | 3.26s runtime

9. Run `display(dbutils.fs.ls("/mnt/data/silver/youflix/youflix_device"))` in `â€œtestâ€` notebook and take screenshot(s) of the results.

```
display(dbutils.fs.ls("/mnt/data/silver/youflix/youflix_device"))
```

(2) Spark Jobs

	path	name
2	dbfs:/mnt/data/silver/youflix/youflix_device/deletion_vector_12c9c9c3-8d4b-400f-8fc1-939169951353.bin	deletion_vector_12c9c9c3-8d4b-400f-8fc1-939169951353.bin
3	dbfs:/mnt/data/silver/youflix/youflix_device/deletion_vector_417949fe-618d-4135-bd38-0150faeea079.bin	deletion_vector_417949fe-618d-4135-bd38-0150faeea079.bin
4	dbfs:/mnt/data/silver/youflix/youflix_device/deletion_vector_4ffb1b0e-6f4d-488e-9963-6912db531e81.bin	deletion_vector_4ffb1b0e-6f4d-488e-9963-6912db531e81.bin
5	dbfs:/mnt/data/silver/youflix/youflix_device/deletion_vector_83b59193-b4d9-4e7e-b050-62ee9a82baa2.bin	deletion_vector_83b59193-b4d9-4e7e-b050-62ee9a82baa2.bin
6	dbfs:/mnt/data/silver/youflix/youflix_device/deletion_vector_fe61f419-efc5-41a2-a4cb-7e1898868f43.bin	deletion_vector_fe61f419-efc5-41a2-a4cb-7e1898868f43.bin
7	dbfs:/mnt/data/silver/youflix/youflix_device/part-00000-71b64ae7-6594-4594-b896-18bd88045fe3.c000.snappy.parquet	part-00000-71b64ae7-6594-4594-b896-18bd88045fe3.c000.snappy.parquet
8	dbfs:/mnt/data/silver/youflix/youflix_device/part-00000-75d51f80-9c99-4d9e-b092-e6b4d5370232.c000.snappy.parquet	part-00000-75d51f80-9c99-4d9e-b092-e6b4d5370232.c000.snappy.parquet
9	dbfs:/mnt/data/silver/youflix/youflix_device/part-00000-93b3df7b-416e-488e-a842-387447f7e713.c000.snappy.parquet	part-00000-93b3df7b-416e-488e-a842-387447f7e713.c000.snappy.parquet
10	dbfs:/mnt/data/silver/youflix/youflix_device/part-00000-9454098a-fabc-4208-9b68-8e5ed6e3e411.c000.snappy.parquet	part-00000-9454098a-fabc-4208-9b68-8e5ed6e3e411.c000.snappy.parquet
11	dbfs:/mnt/data/silver/youflix/youflix_device/part-00000-bee0bc12-c0d4-4e6d-846f-8827c28e4d27.c000.snappy.parquet	part-00000-bee0bc12-c0d4-4e6d-846f-8827c28e4d27.c000.snappy.parquet
12	dbfs:/mnt/data/silver/youflix/youflix_device/part-00000-ddc5c569-d6dc-486a-9883-caae3e121c30.c000.snappy.parquet	part-00000-ddc5c569-d6dc-486a-9883-caae3e121c30.c000.snappy.parquet

12 rows | 0.52s runtime

Refreshed 1 minute ago

10. Run `display(dbutils.fs.ls("/mnt/data/silver/youflix/youflix_subscription"))` in `â€œtestâ€` notebook and take screenshot(s) of the results.

```
display(dbutils.fs.ls("/mnt/data/silver/youflix/youflix_device"))
```

(2) Spark Jobs

	^A _B path	^A _B name
2	dbfs:/mnt/data/silver/youflix/youflix_device/deletion_vector_12c9c9c3-8d4b-400f-8fc1-939169951353.bin	deletion_vector_12c9c9c3-8d4b-400f-8fc1-939169951353.bin
3	dbfs:/mnt/data/silver/youflix/youflix_device/deletion_vector_417949fe-618d-4135-bd38-0150faeea079.bin	deletion_vector_417949fe-618d-4135-bd38-0150faeea079.bin
4	dbfs:/mnt/data/silver/youflix/youflix_device/deletion_vector_4ffb1b0e-6f4d-488e-9963-6912db531e81.bin	deletion_vector_4ffb1b0e-6f4d-488e-9963-6912db531e81.bin
5	dbfs:/mnt/data/silver/youflix/youflix_device/deletion_vector_83b59193-b4d9-4e7e-b050-62ee9a82baa2.bin	deletion_vector_83b59193-b4d9-4e7e-b050-62ee9a82baa2.bin
6	dbfs:/mnt/data/silver/youflix/youflix_device/deletion_vector_fe61f419-efc5-41a2-a4cb-7e1898868f43.bin	deletion_vector_fe61f419-efc5-41a2-a4cb-7e1898868f43.bin
7	dbfs:/mnt/data/silver/youflix/youflix_device/part-00000-71b64ae7-6594-4594-b896-18bd88045fe3.c000.snappy.parquet	part-00000-71b64ae7-6594-4594-b896-18bd88045fe3.c000.snappy.parquet
8	dbfs:/mnt/data/silver/youflix/youflix_device/part-00000-75d51f80-9c99-4d9e-b092-e6b4d5370232.c000.snappy.parquet	part-00000-75d51f80-9c99-4d9e-b092-e6b4d5370232.c000.snappy.parquet
9	dbfs:/mnt/data/silver/youflix/youflix_device/part-00000-93b3df7b-416e-488e-a842-387447f7e713.c000.snappy.parquet	part-00000-93b3df7b-416e-488e-a842-387447f7e713.c000.snappy.parquet
10	dbfs:/mnt/data/silver/youflix/youflix_device/part-00000-9454098a-fabc-4208-9b68-8e5ed6e3e411.c000.snappy.parquet	part-00000-9454098a-fabc-4208-9b68-8e5ed6e3e411.c000.snappy.parquet
11	dbfs:/mnt/data/silver/youflix/youflix_device/part-00000-bee0bc12-c0d4-4e6d-846f-8827c28e4d27.c000.snappy.parquet	part-00000-bee0bc12-c0d4-4e6d-846f-8827c28e4d27.c000.snappy.parquet
12	dbfs:/mnt/data/silver/youflix/youflix_device/part-00000-ddc5c569-d6dc-486a-9883-caae3e121c30.c000.snappy.parquet	part-00000-ddc5c569-d6dc-486a-9883-caae3e121c30.c000.snappy.parquet

12 rows | 0.52s runtime

Refreshed 1 minute ago

11. Run `display(dbutils.fs.ls("/mnt/data/silver/youflix/youflix_user"))` in `â€œtestâ€` notebook and take screenshot(s) of the results.

12. Run `display(dbutils.fs.ls("/mnt/data/silver/youflix/youflix_user_subscription_device"))` in `âœœtestâœœ` notebook and take screenshot(s) of the results.

```
display(dbutils.fs.ls("/mnt/data/silver/youflix/youflix_user_subscription_device"))
```

▶ (2) Spark Jobs

	^A _C path	^A _C name
1	dbfs:/mnt/data/silver/youflix/youflix_user_subscription_device/_delta_log/	_delta_log/
2	dbfs:/mnt/data/silver/youflix/youflix_user_subscription_device/part-00000-60e45aa0-f821-474f-a9a0-ce68a8f17ae7.c000.snappy.parquet	part-00000-60e45aa0-f821-474f-a9a0-ce68a8f17

2 rows | 14.85s runtime

Refreshed 11 minutes ago

13. In Synapse Workspace, navigate to Data section, find in Linked tab your container, open `silver/youflix` and check number of rows for each delta table using SQL query.

CopyPipelineAlldataSQL script 4

RunUndoPublishQuery planConnect toBuilt-inUse databasemaster

```
1  -- This is auto-generated code
2  SELECT
3    count(*)
4  FROM
5    OPENROWSET(
6      BULK 'https://stdimentoringdatalakeed.dfs.core.windows.net/data/silver/youflix/youflix_device/*.snappy.parquet',
7      FORMAT = 'PARQUET'
8    ) AS [result]
9
```

ResultsMessages

ViewTableChartExport results

Search

(No column name)

180

14. Take screenshot(s) of SQL queries in with count values.

CopyPipelineAlldataSQL script 4SQL script 5

RunUndoPublishQuery planConnect toBuilt-inUse databasemaster

```
1  -- This is auto-generated code
2  SELECT
3    count(*)
4  FROM
5    OPENROWSET(
6      BULK 'https://stdimentoringdatalakeed.dfs.core.windows.net/data/silver/youflix/youflix_subscription/*.snappy.parquet',
7      FORMAT = 'PARQUET'
8    ) AS [result]
9
```

ResultsMessages

ViewTableChartExport results

Search

(No column name)

15

Publish all

CopyPipelineAlldataSQL script 4SQL script 5SQL script 6

RunUndoPublishQuery planConnect toBuilt-inUse databasemaster

```
1  -- This is auto-generated code
2  SELECT
3    count(*)
4  FROM
5    OPENROWSET(
6      BULK 'https://stdimentoringdatalakeed.dfs.core.windows.net/data/silver/youflix/youflix_user_subscription_device/*.snappy.parquet',
7      FORMAT = 'PARQUET'
8    ) AS [result]
9
```

ResultsMessages

ViewTableChartExport results

Search

(No column name)

79938

15. Connect to MS SQL Server YouFlixDB database and run the following command: sql EXEC youflix_internal.sp_youflix_tables_insert_update 10000, 15;

YouflixDBExecute

SQLQuery1.sql - EP...Ekaterina_Dul (51))*EXEC youflix_internal.sp_youflix_tables_insert_update 10000, 15;

Object Explorer

connect

EPGETBIW05E7\MSSQLSERVER01 (SQL S

Databases

System Databases

Database Snapshots

YouflixDB

Database Diagrams

Tables

Views

External Resources

Synonyms

Programmability

Query Store

Service Broker

Storage

Security

Security

Server Objects

Replication

Always On High Availability

Management

Integration Services Catalogs

SQL Server Agent (Agent XPs disabled)

XEvent Profiler

150 %

Messages

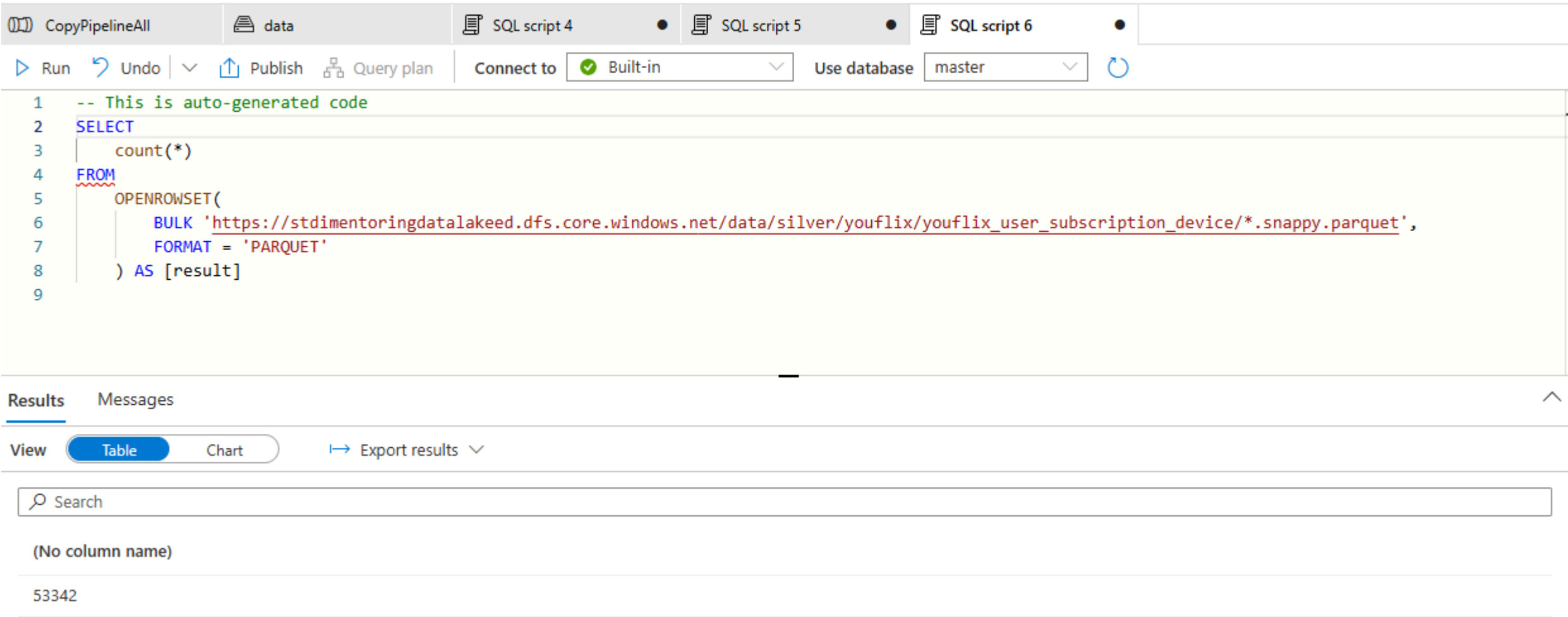
15: updating users in YOUFLIX.user table.
Moving users to the main YOUFLIX schema.
10000 users moved to the main YOUFLIX schema.
52984 devices moved to the main YOUFLIX schema.

Completion time: 2025-01-14T22:13:36.5832918+04:00

16. Run pipeline from task 1.1.

17. Run the notebook uc1_load_bronze_to_silver.ipynb.

18. In Synapse Workspace, navigate to Data section, find in Linked tab your container, open bronze/youflix and check number of rows for each newly loaded file using SQL query.



19. Take screenshot(s) of SQL queries with count values.

Full Notebook:

```
```python
```

Authenticate Databricks to access Data Lake

value must be updated with Azure Key Vault-backed secret scope name

must be updated with Azure Key Vault value for application id

must be updated with Azure Key Vault value for tenant id

must be updated with Azure Key Vault value for client secret

```
AppID = dbutils.secrets.get(scope="secret-scope-ed", key="applicationID>")
TenantID = dbutils.secrets.get(scope="secret-scope-ed", key="tenantID")
ClientSecret = dbutils.secrets.get(scope="secret-scope-ed", key="clientSecret")
```

```
configs = {"fs.azure.account.auth.type": "OAuth", "fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider", "fs.azure.account.oauth2.client.id": AppID, "fs.azure.account.oauth2.client.secret": ClientSecret, "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/{tenant}/oauth2/token".format(tenant=TenantID)}
```

Mounting data in an Azure storage account using an Azure Active Directory (Azure AD) application service principal for authentication

dbutils.fs.unmount("/mnt/data") #use to unmount data if needed

must be replaced with your Azure Data Lake Storage Gen2 name

```
try: dbutils.fs.mount(source="abfss://data@stdimentoringdatalakeed.dfs.core.windows.net/", mount_point="/mnt/data", extra_configs=configs) except Exception as e: if "Directory already mounted" in str(e): pass # Ignore error if already mounted. else: raise e ```
```

```
python dbutils.fs.mounts()
```

```
[MountInfo(mountPoint='/databricks-datasets', source='databricks-datasets', encryptionType=''),
MountInfo(mountPoint='/Volumes', source='UnityCatalogVolumes', encryptionType=''),
MountInfo(mountPoint='/databricks/mlflow-tracking', source='databricks/mlflow-tracking', encryptionType=''),
MountInfo(mountPoint='/databricks-results', source='databricks-results', encryptionType=''),
MountInfo(mountPoint='/databricks/mlflow-registry', source='databricks/mlflow-registry', encryptionType=''),
MountInfo(mountPoint='/Volume', source='DbfsReserved', encryptionType=''),
MountInfo(mountPoint='/volumes', source='DbfsReserved', encryptionType=''),
MountInfo(mountPoint='/mnt/data', source='wasbs://data@stdimentoringdatalakeed.blob.core.windows.net/', encryptionType=''),
MountInfo(mountPoint='/', source='DatabricksRoot', encryptionType=''),
MountInfo(mountPoint='/volume', source='DbfsReserved', encryptionType='')]
```

```
```python %sql -- Creating database and delta tables if not exist
```

```
CREATE DATABASE IF NOT EXISTS YouFlix;
CREATE TABLE IF NOT EXISTS YouFlix.youflix_user_delta( user_id BIGINT, user_name STRING, user_email STRING, first_name STRING, last_name STRING, user_date_of_birth DATE, user_address STRING, user_phone STRING, created_timestamp TIMESTAMP, expiration_timestamp TIMESTAMP, modified_timestamp TIMESTAMP ) USING DELTA LOCATION '/mnt/data/silver/youflix/youflix_user';
```

```
CREATE TABLE IF NOT EXISTS YouFlix.youflix_device_delta( device_id BIGINT, device_name STRING, device_type STRING, device_os STRING, created_timestamp TIMESTAMP ) USING DELTA LOCATION '/mnt/data/silver/youflix/youflix_device';
```

```
CREATE TABLE IF NOT EXISTS YouFlix.youflix_subscription_delta( subscription_id BIGINT, subscription_name STRING, subscription_type STRING, subscription_video_quality STRING, subscription_max_devices INT, created_timestamp TIMESTAMP, expiration_timestamp TIMESTAMP ) USING DELTA LOCATION '/mnt/data/silver/youflix/youflix_subscription';
```

```
CREATE TABLE IF NOT EXISTS YouFlix.youflix_user_subscription_device_delta( user_subscription_device_id BIGINT, user_id BIGINT, device_id BIGINT, created_timestamp TIMESTAMP ) USING DELTA LOCATION '/mnt/data/silver/youflix/youflix_user_subscription_device';
```

...

```
```python import re from delta.tables import * from pyspark.sql.functions import * from pyspark.sql import Window
```

## dictionary "entities" store the name of the entities and it's BK

```
entities = {"device": "device_id", "subscription": "subscription_id", "user": "user_id", "user_subscription_device": "user_subscription_device_id"}
```

```
try: # looping through every entity for entity in entities.items():
```

```
 bronzePath = "/mnt/data/bronze/youflix/youflix_{entity_name}".format(entity_name=entity[0])
 silverPath = "/mnt/data/silver/youflix/youflix_{entity_name}".format(entity_name=entity[0])
 processedPath = "/mnt/data/bronze/youflix/processed/youflix_{entity_name}".format(entity_name=entity[0])
```

```
 # files to load from bronze to silver
 filePaths = dbutils.fs.ls(bronzePath)
```

```
 if filePaths:
```

```
 # MERGE BRONZE TO SILVER
```

```
 # TODO
 # use pyspark spark.read.load method to create dataframe based on csv files in the bronzePath
 # do not forget that file has header
```

```
 bronzeDF = spark.read.option("header", "true").option("inferSchema", "true").csv(bronzePath)
```

```
 partition = Window.partitionBy(entity[1]).orderBy(col("filedate").desc())
```

```
 bronzeDF_cln = (bronzeDF.withColumn("filedate",
 to_timestamp(regexp_extract(input_file_name(), '([\d]{14})', 0),
 'yyyyMMddhhmmss'))
 .withColumn("rn", row_number().over(partition))
 .filter("rn == 1")
)
```

```
 # get delta table at the silver path
 silver_table = DeltaTable.forPath(spark, silverPath)
```

```
 # TODO
 # add your code into brackets below
 # use pyspark merge method to merge bronzeDF_cln dataframe into silver_table by BK. BK for table can be accessible by entity[1]
 # refer to the https://learn.microsoft.com/en-us/azure/databricks/delta/merge to learn about upsert into a Delta Lake table using merge.
 # this article https://docs.delta.io/latest/delta-update.html#table-deletes-updates-and-merges&language-python will help to understand how to delete, update and merge Delta tables
```

```
 (
 silver_table.alias('silver')
 .merge(
 bronzeDF_cln.alias('updates'),
 f'silver.{entity[1]} = updates.{entity[1]}'
)
 .whenMatchedUpdate(
 set=
 {
 col: f"updates.{col}" for col in bronzeDF.columns
 }
)
 .whenNotMatchedInsert(values=
 {
 col: f"updates.{col}" for col in bronzeDF.columns
 }
)
 .execute()
)
```

```
 # MOVE TO PROCESSED DIRECTORY
```

```
 # looping through every file in directory
 for file_info in filePaths:
 # creating tuple to store (year, month, day) of the file
 file_date = (re.split("_", file_info.name)[-1][0:4], re.split("_", file_info.name)[-1][4:6],
 re.split("_", file_info.name)[-1][6:8])

 # TODO
 # complete mv command to move files from bronzePath to processedPath according to the structure in 1.2.7.
 # use file_date tuple to get year, month and day of the file, use file_info.name to get name of file
 dbutils.fs.mv(file_info.path, processedPath + "/" + file_date[0] + "/" + file_date[1] + "/" + file_date[
 2] + "/" + file_info.name)
```

```
 # REMOVE Success.csv
```

```
 # TODO
 dbutils.fs.rm(f"/mnt/data/bronze/youflix/Success_{entity[0]}.txt")
```

```
 else:
 print("Entity \"{entity}\" - No files for load".format(entity=entity[0]))
```

```
except Exception as e: print(e)
```

...