

## MỤC LỤC

<b>TỔNG QUAN DỰ ÁN</b>	2
1. Mục tiêu dự án	2
2. Mô hình dữ liệu (Star Schema)	2
3. Xử lý thay đổi dữ liệu	2
4. Kiến trúc Medallion	2
5. Công nghệ & dịch vụ Azure	3
6. Nguồn dữ liệu	3
<b>CHUẨN BỊ HẠ TẦNG DEVELOPMENT</b>	4
1. Resource Group	4
2. Storage Account	4
3. Azure Databricks	5
4. Service Principal	6
<b>PIPELINE XỬ LÝ DỮ LIỆU</b>	7
1. Ingest (Bronze Layer)	7
2. Transform (Silver Layer)	11
3. Load (Gold Layer)	17
<b>CI/CD VÀ TỰ ĐỘNG HÓA</b>	25
1. Chuẩn bị hạ tầng Production	25
2. Kết nối Git với Workspace Dev	27
3. Thiết lập CI/CD Production	28
4. Tự động hóa Lịch Chạy Job	28
5. Kết nối Power BI với Azure Databricks	29






# TỔNG QUAN DỰ ÁN

## 1. Mục tiêu dự án


- Tự động thu thập thông tin phim mới phát hành và phim đang thịnh hành mỗi tuần.
- Hỗ trợ người xem đưa ra gợi ý nhanh chóng, giúp lựa chọn phim phù hợp với thể loại yêu thích.

## 2. Mô hình dữ liệu (Star Schema)

### ❖ Bảng Dimension

-  dim\_movie – Thông tin phim (ID, tiêu đề, thời lượng...)
-  dim\_genre – Thể loại phim
-  dim\_company – Công ty sản xuất
-  dim\_director – Đạo diễn
-  dim\_date – Tuần phân tích

### ❖ Bảng Fact

-  fact\_movies – Ghi nhận lượt xem, đánh giá, tương tác, liên kết các bảng dimension

## 3. Xử lý thay đổi dữ liệu

- CDC (Change Data Capture): Chỉ ingest dữ liệu mới hoặc đã thay đổi dựa trên timestamp.
- SCD (Slowly Changing Dimensions): Sử dụng Type 1 (ghi đè) để quản lý lịch sử dữ liệu dimension.

## 4. Kiến trúc Medallion

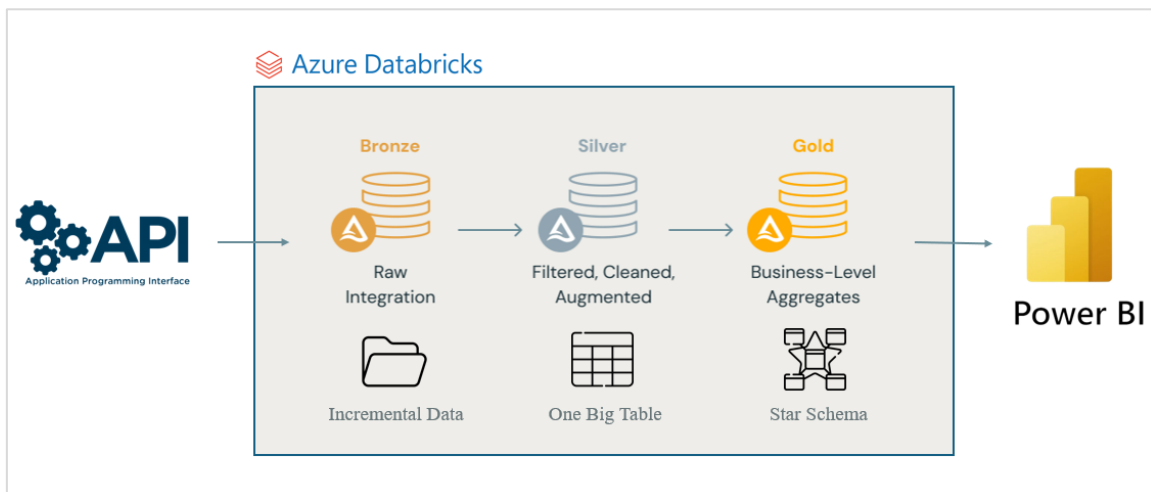
- Bronze – Dữ liệu thô ingest trực tiếp từ API (phim mới, phim xu hướng, thể loại)
- Silver – Dữ liệu đã làm sạch, chuẩn hóa schema
- Gold: Dữ liệu tổng hợp, sẵn sàng cho báo cáo và trực quan hóa.

## 5. Công nghệ & dịch vụ Azure

- Azure Data Lake Storage Gen2 – Tạo 3 container (bronze, silver, gold).
- Azure Databricks – Sử dụng Delta Lake để lưu trữ, Databricks SQL để biến đổi.
- Quản lý bảo mật – Service Principal để cấp quyền truy cập an toàn cho Databricks.
- CI/CD – Sử dụng GitHub Actions để tự động deploy notebooks giữa môi trường Dev/Prod.

## 6. Nguồn dữ liệu

- ❖ The Movie Database (TMDB) API: <https://www.themoviedb.org/>
  - Cung cấp metadata phim: ngày phát hành, poster, lượt bình luận, điểm đánh giá...
  - Hỗ trợ truy vấn incremental theo ngày/tuần để tối ưu băng thông và thời gian xử lý.

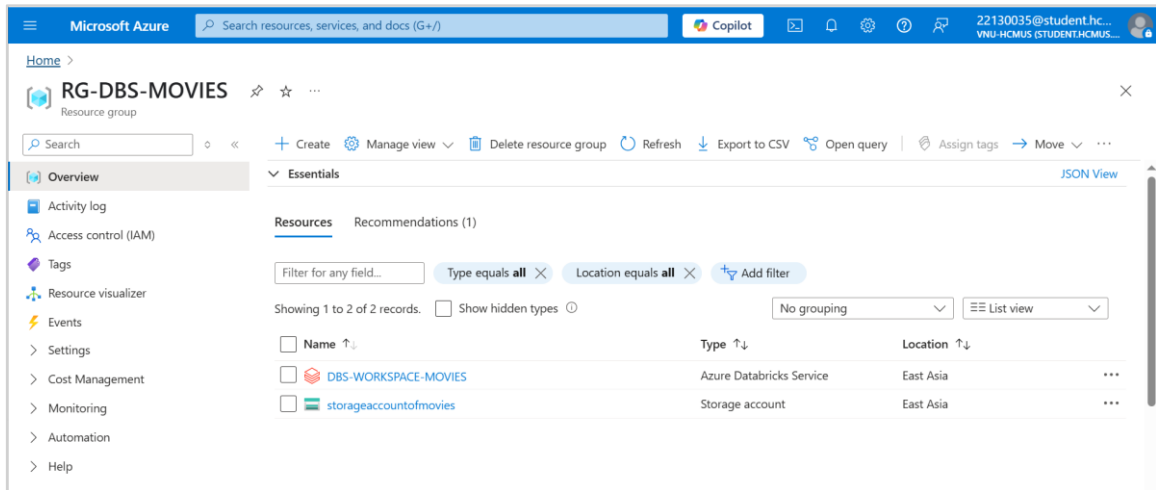


# CHUẨN BỊ HẠ TẦNG DEVELOPMENT

Trước khi triển khai pipeline, chúng ta cần thiết lập đầy đủ các tài nguyên Azure để đảm bảo môi trường sẵn sàng cho việc phát triển và vận hành.

## 1. Resource Group

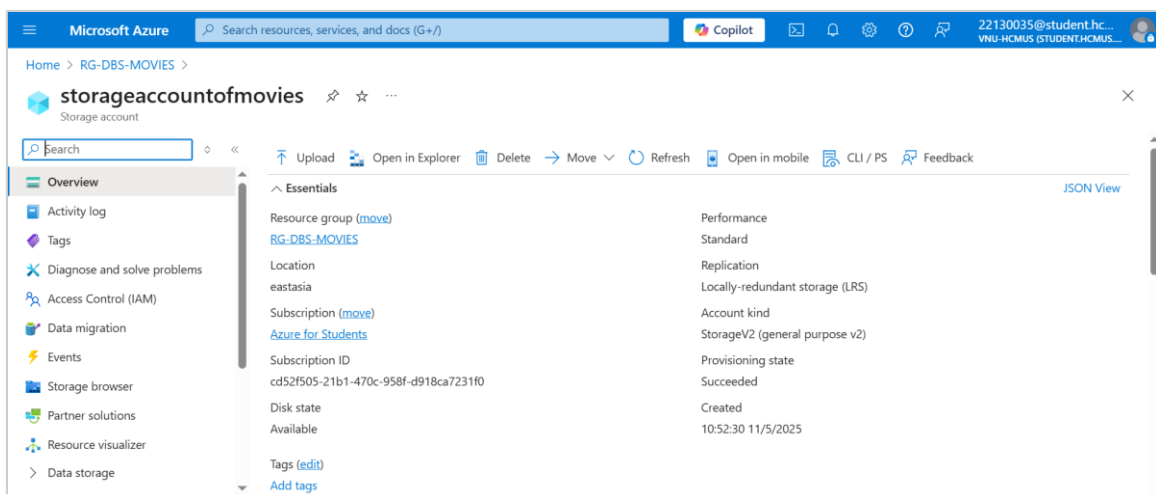
Tạo một Resource Group



Resource Group “RG-DBS-MOVIES” là vùng chứa tập trung các tài nguyên liên quan đến dự án, giúp chúng ta dễ dàng quản lý, theo dõi chi phí và quyền truy cập.

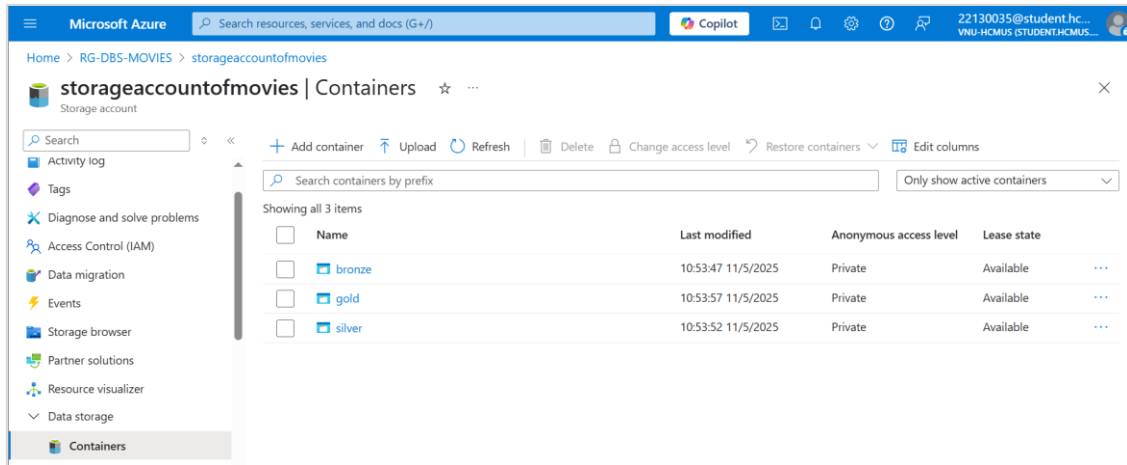
## 2. Storage Account

Tạo một Storage Account



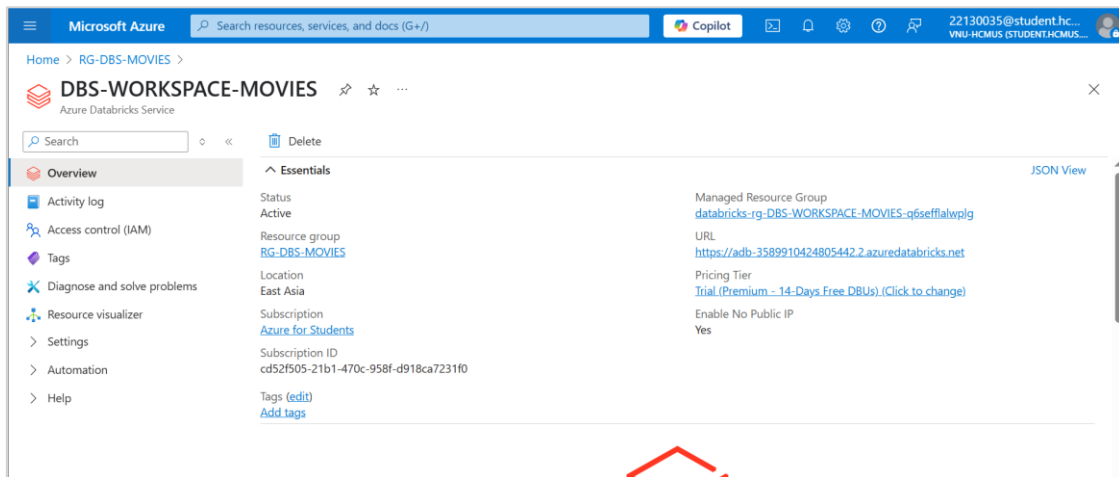
Storage Account “storageaccountofmovies” chịu trách nhiệm lưu trữ dữ liệu thô, trung gian và kết quả cuối cùng.

## Tạo các Containers trong Data Lake



## 3. Azure Databricks

### Tạo một Azure Databricks Workspace



Workspace “DBS-WORKSPACE-MOVIES” là nơi chạy các notebook và job xử lý dữ liệu. Chúng ta sẽ cấu hình cluster và kết nối tới Data Lake Storage Gen 2.

## 4. Service Principal

Tạo Service Principal và cấp quyền Reader/Contributor cho Databricks tới Storage Account, đảm bảo authentication an toàn khi Spark job truy xuất dữ liệu.

### Bronze Layer

```
container_name = "bronze"
storage_account_name = "storageaccountofmovies"
client_id = "01a8c98d-6c5d-45d5-8e61-077de7c4060d"
tenant_id = "40127cd4-45f3-49a3-b05d-315a43a9f033"
client_secret = "KzC8Q~gK6i.IRJRETPmzey_s3-EoW-bSn.cXKaw_"
```

### Silver Layer

```
container_name = "silver"
storage_account_name = "storageaccountofmovies"
client_id = "01a8c98d-6c5d-45d5-8e61-077de7c4060d"
tenant_id = "40127cd4-45f3-49a3-b05d-315a43a9f033"
client_secret = "KzC8Q~gK6i.IRJRETPmzey_s3-EoW-bSn.cXKaw_"
```

### Gold Layer

```
container_name = "gold"
storage_account_name = "storageaccountofmovies"
client_id = "01a8c98d-6c5d-45d5-8e61-077de7c4060d"
tenant_id = "40127cd4-45f3-49a3-b05d-315a43a9f033"
client_secret = "KzC8Q~gK6i.IRJRETPmzey_s3-EoW-bSn.cXKaw_"
```

## PIPELINE XỬ LÝ DỮ LIỆU

## 1. Ingest (Bronze Layer)

❖ Xác định khoảng thời gian tuần

Chạy Spark SQL trên bảng weekCollectData để tính ngày đầu tuần (GTE) và ngày cuối tuần (LTE).

```
sparksql = """SELECT
    MAX(DATE(DAY_LAST_WEEK)) + INTERVAL 1 day AS GTE,
    MAX(DATE(DAY_LAST_WEEK)) + INTERVAL 7 days AS LTE
FROM weekCollectData"""

sparkdf = spark.sql(sparksql)
row = sparkdf.select('GTE', 'LTE').collect()[0]
gte = row['GTE']
lte = row['LTE']
```

▶ (2) Spark Jobs

▶ sparkdf: pyspark.sql.dataframe.DataFrame = [GTE: date, LTE: date]

## ❖ Thu thập dữ liệu

➤ Lấy danh sách thể loại (genres)

Gọi endpoint `/genre/movie/list` để tải về bảng genres và lưu vào `/bronze/genres` (mode overwrite).

```

▶ ✓ May 15, 2025 (<1s) 2

url = "https://api.themoviedb.org/3/genre/movie/list?language=en"

headers = {
    "accept": "application/json",
    "Authorization": "Bearer eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI2NjQ1ZTU5MDQ5MjYwM2RjZTg1N2QzMjE0NjYzZGYyMyIsIm5iZiI6MS43NDY3MDQ1MzU0ODYwMDAyZSs5LCJzdWIiOiI2ODFjOTg5NzA3MGMxZWZjNjUzZWY3ZWM3N2Q1LCJzY29wZXMiOiIsYXBpX3JlYWQiXSwidmVyc2lvdCI6MX0.rjblOj3waWpva4HHpVwcm0Ij2-kLCBF10t3d8FwqhqnQ"
}

▶ ✓ May 15, 2025 (1s) 3

response = requests.get(url, headers=headers)

data = response.json().get('genres', [])

```

➤ Lấy phim xu hướng & phim mới

Gọi `/trending/movie/week/`

discover/movie?primary\_release\_date.gte=GTE&amp;primary\_release\_date.lte=LTE

```

▶ ✓ May 17, 2025 (<1s) 6

# URLs và headers

url_trend = "https://api.themoviedb.org/3/trending/movie/week?language=en-US"
url_new = f"https://api.themoviedb.org/3/discover/movie?include_adult=true&include_video=false&language=en-US&page=1&primary_release_date.gte={str(gte)}&primary_release_date.lte={str(lte)}&sort_by=popularity.desc"
url_base = "https://api.themoviedb.org/3/movie/"
url_language = "?language=en-US"

headers = {
    "accept": "application/json",
    "Authorization": "Bearer eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI2NjQ1ZTU5MDQ5MjYwM2RjZTg1N2QzMjE0NjYzZGYyMyIsIm5iZiI6MS43NDY3MDQ1MzU0ODYwMDAyZS5LCjZdWlI0IiI2ODFjOTg5NzA3MGMrZWEzNjUzZWZmN2Q1LCJzY29wZXMiOiIsiYXBpX3JlYWQiXSwidmVyc2lvbiI6MX0.rjbLOj3waKpva4HhpVWcm0Ij2-kLCBF10t3d8FwqhnQ"
}

```

➤ Bổ sung thông tin chi tiết

Với mỗi movie\_id, dùng hai endpoint /movie/{id} và /movie/{id}/credits để gán thêm companies, revenue, budget, runtime, directors.

```

▶ ✓ May 17, 2025 (46s) 9

# Duyệt qua từng movie_id và lấy thông tin
movie_companies_trend = []
movie_revenue_trend = []
movie_budget_trend = []
movie_runtime_trend = []
movie_directors_trend = []
movie_companies_new = []
movie_revenue_new = []
movie_budget_new = []
movie_runtime_new = []
movie_directors_new = []

for movie_id in tqdm(lst_id_trend, desc="Trending movie info"):
    companies, revenue, budget, runtime, directors = get_movie_info(movie_id)
    movie_companies_trend.append(companies)
    movie_revenue_trend.append(revenue)
    movie_budget_trend.append(budget)
    movie_runtime_trend.append(runtime)
    movie_directors_trend.append(directors)

```



```

for movie_id in tqdm(lst_id_new, desc="New movie info"):
    companies, revenue, budget, runtime, directors = get_movie_info(movie_id)
    movie_companies_new.append(companies)
    movie_revenue_new.append(revenue)
    movie_budget_new.append(budget)
    movie_runtime_new.append(runtime)
    movie_directors_new.append(directors)

# Thêm vào DataFrame
df_trend['production_companies'] = movie_companies_trend
df_trend['revenue'] = movie_revenue_trend
df_trend['budget'] = movie_budget_trend
df_trend['runtime'] = movie_runtime_trend
df_trend['directors'] = movie_directors_trend
df_new['production_companies'] = movie_companies_new
df_new['revenue'] = movie_revenue_new
df_new['budget'] = movie_budget_new
df_new['runtime'] = movie_runtime_new
df_new['directors'] = movie_directors_new

```

### ➤ Chuyển sang Spark DataFrame

Biến đổi Pandas DataFrame → Spark DataFrame, thêm cột collect\_date = GTE

(áp dụng kỹ thuật CDC Timestamp)

```

▶ May 17, 2025 (<1s) 10 Python
# chuyển dataframe pandas thành dataframe spark
sparkdf_trend = spark.createDataFrame(df_trend).withColumn('collect_date', lit(gte))
sparkdf_new = spark.createDataFrame(df_new).withColumn('collect_date', lit(gte))

▶ sparkdf_new: pyspark.sql.dataframe.DataFrame = [adult: boolean, backdrop_path: string ... 18 more fields]
▶ sparkdf_trend: pyspark.sql.dataframe.DataFrame = [backdrop_path: string, id: long ... 19 more fields]

```

### ➤ Ghi vào Bronze Layer & cập nhật lịch

Genres: lưu Delta table tại /bronze/genres (mode overwrite).

```

▶ May 15, 2025 (33s) 5 Python
pathGenres = '/mnt/storageaccountofmovies/bronze/genres'

(
    genres_spark
    .write
    .format("delta")
    .mode("overwrite")
    .save(pathGenres)
)

▶ (4) Spark Jobs

```

Movies: ghi hai Delta paths /bronze/DataTrend và /bronze/DataNew (mode append, mergeSchema=true).

```
▶ ✓ May 17, 2025 (5s) 11

pathDataTrend = '/mnt/storageaccountofmovies/bronze/DataTrend'
pathDataNew = '/mnt/storageaccountofmovies/bronze/DataNew'

(
    sparkdf_trend
    .write
    .format("delta")
    .option("mergeSchema", "true")
    .mode("append")
    .save(pathDataTrend)
)

(
    sparkdf_new
    .write
    .format("delta")
    .option("mergeSchema", "true")
    .mode("append")
    .save(pathDataNew)
)
```

Cập nhật tuần: chèn DAY\_FIRST\_WEEK=GTE, DAY\_LAST\_WEEK=LTE vào bảng weekCollectData.

```
▶ ✓ May 17, 2025 (2s) 12 Python

time_getdata = f"""
INSERT INTO weekCollectData (DAY_FIRST_WEEK, DAY_LAST_WEEK)
VALUES('{str(gte)}', '{str(lte)}')
"""

spark.sql(time_getdata)

▶ (4) Spark Jobs

DataFrame[num_affected_rows: bigint, num_inserted_rows: bigint]
```

## 2. Transform (Silver Layer)

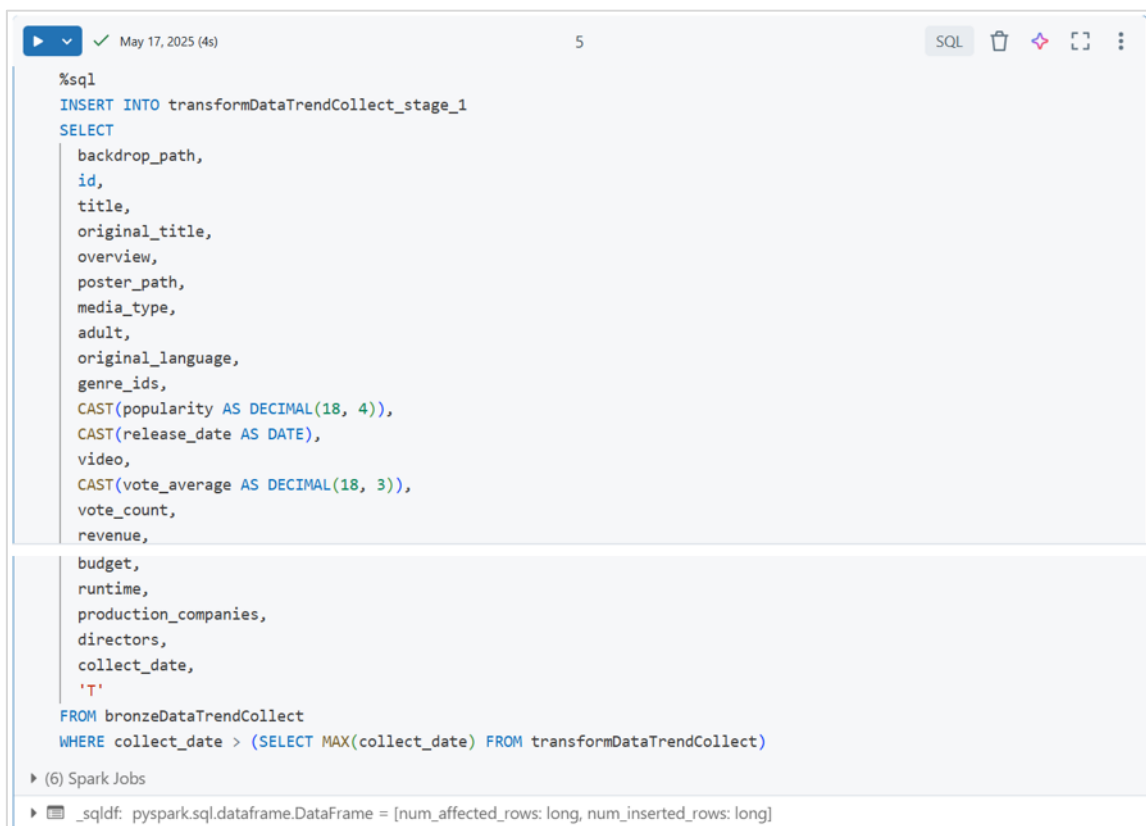
### ❖ Stage 1 – Chuẩn hóa & đánh dấu loại dữ liệu

Lấy dữ liệu thô từ bronzeDataTrendCollect và bronzeDataNewCollect.

CAST lại kiểu cột:

- popularity → DECIMAL(18,4)
- vote\_average → DECIMAL(18,3)
- release\_date → DATE

Thêm cột flag\_category bằng literal ('T' cho phim xu hướng, 'N' cho phim mới).



```
%sql
INSERT INTO transformDataTrendCollect_stage_1
SELECT
  backdrop_path,
  id,
  title,
  original_title,
  overview,
  poster_path,
  media_type,
  adult,
  original_language,
  genre_ids,
  CAST(popularity AS DECIMAL(18, 4)),
  CAST(release_date AS DATE),
  video,
  CAST(vote_average AS DECIMAL(18, 3)),
  vote_count,
  revenue,
  budget,
  runtime,
  production_companies,
  directors,
  collect_date,
  'T'
FROM bronzeDataTrendCollect
WHERE collect_date > (SELECT MAX(collect_date) FROM transformDataTrendCollect)
```


▶ (6) Spark Jobs

▶ \_sqldf: pyspark.sql.dataframe.DataFrame = [num\_affected\_rows: long, num\_inserted\_rows: long]

```
%sql
INSERT INTO transformDataNewCollect_stage_1
SELECT
  backdrop_path,
  id,
  title,
  original_title,
  overview,
  poster_path,
  'Movie',
  adult,
  original_language,
  genre_ids,
  CAST(popularity AS DECIMAL(18, 4)),
  CAST(release_date AS DATE),
  video,
  CAST(vote_average AS DECIMAL(18, 3)),
  vote_count,
  revenue,

  budget,
  runtime,
  production_companies,
  directors,
  collect_date,
  'N'
FROM bronzeDataNewCollect
WHERE collect_date > (SELECT MAX(collect_date) FROM transformDataNewCollect)
```

▶ (6) Spark Jobs

▶  \_sqldf: pyspark.sql.dataframe.DataFrame = [num\_affected\_rows: long, num\_inserted\_rows: long]

## ❖ Stage 2 – Phân rã mảng (Explode) thành dòng

Dùng `explode(genre_ids)` tách mỗi phần tử trong mảng `genre_ids` thành một dòng riêng.

Tương tự với `production_companies` và `directors`.

```
May 17, 2025 (2s) 7

INSERT INTO transformDataTrendCollect_stage_2
SELECT
  backdrop_path,
  id,
  title,
  original_title,
  overview,
  poster_path,
  media_type,
  adult,
  original_language,
  explode(genre_ids),
  popularity,
  release_date,
  video,
  vote_average,
  vote_count,
  revenue,
  budget,
  runtime,
  explode(production_companies),
  explode(directors),
  collect_date,
  flag_category
FROM transformDataTrendCollect_stage_1

(4) Spark Jobs
_sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]
```

```
May 17, 2025 (3s) 12

INSERT INTO transformDataNewCollect_stage_2
SELECT
  backdrop_path,
  id,
  title,
  original_title,
  overview,
  poster_path,
  media_type,
  adult,
  original_language,
  explode(genre_ids),
  popularity,
  release_date,
  video,
  vote_average,
  vote_count,
  revenue,
  budget,
  runtime,
  explode(production_companies),
  explode(directors),
  collect_date,
  flag_category
FROM transformDataNewCollect_stage_1

(4) Spark Jobs
_sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]
```

### ❖ Stage 3 – Thêm dữ liệu vào bảng chính

Biến đổi cột `production_companies` thành hai phần `company` và `country`.

Chèn dữ liệu vào `transformDataTrendCollect` và `transformDataNewCollect`.



```
INSERT INTO transformDataTrendCollect
SELECT
  backdrop_path,
  id,
  title,
  original_title,
  overview,
  poster_path,
  media_type,
  adult,
  original_language,
  genre_ids,
  popularity,
  release_date,
  video,
  vote_average,
  vote_count,
  revenue,
  budget,
  runtime,
  trim(element_at(split(trim(both '{}' from production_companies), ','), 3)),
  trim(element_at(split(trim(both '{}' from production_companies), ','), -1)),
  directors,
  collect_date,
  flag_category
FROM transformDataTrendCollect_stage_2
```

▶ (4) Spark Jobs

▶ \_sqldf: pyspark.sql.dataframe.DataFrame = [num\_affected\_rows: long, num\_inserted\_rows: long]

```
INSERT INTO transformDataNewCollect
```

```
SELECT
```

```
  backdrop_path,
```

```
  id,
```

```
  title,
```

```
  original_title,
```

```
  overview,
```

```
  poster_path,
```

```
  media_type,
```

```
  adult,
```

```
  original_language,
```

```
  genre_ids,
```

```
  popularity,
```

```
  release_date,
```

```
  video,
```

```
  vote_average,
```

```
  vote_count,
```

```
  revenue,
```

```
  budget,
```

```
  runtime,
```

```
  trim(element_at(split(trim(both '{}' from production_companies), ','), 3)),
```

```
  trim(element_at(split(trim(both '{}' from production_companies), ','), -1)),
```


```
  directors,
```

```
  collect_date,
```

```
  flag_category
```

```
FROM transformDataNewCollect_stage_2
```

▶ (4) Spark Jobs

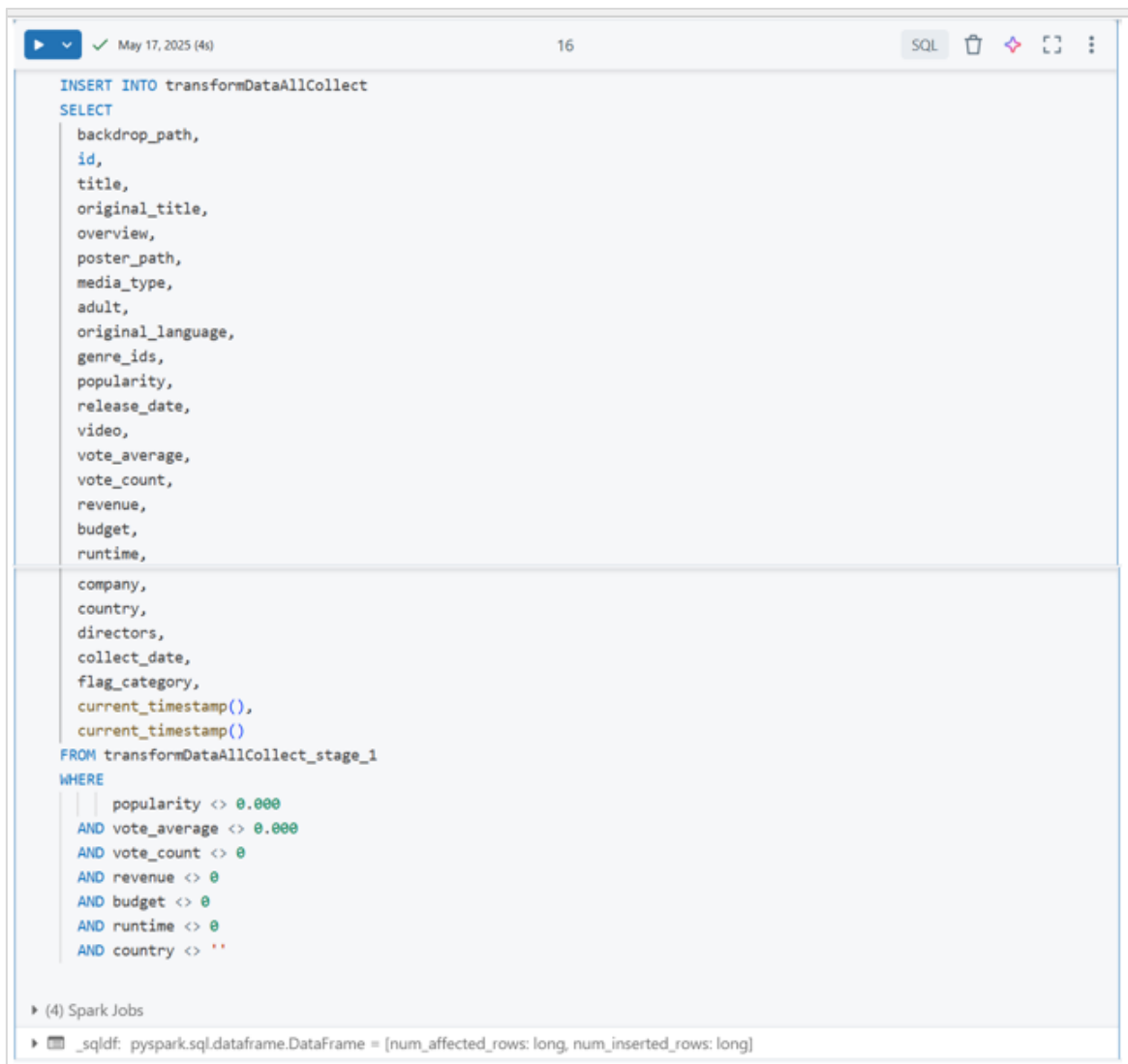
▶  \_sqldf: pyspark.sql.dataframe.DataFrame = [num\_affected\_rows: long, num\_inserted\_rows: long]

❖ Hợp nhất, lọc & tạo Silver Table chính

UNION hai bảng transformDataTrendCollect và transformDataNewCollect vào bảng tạm transformDataAllCollect\_stage\_1

INSERT INTO transformDataAllCollect

- ✓ Lọc các bản ghi không hợp lệ (popularity, vote\_count, revenue,... ≠ 0; country ≠ "")
- ✓ Thêm time\_inserted và time\_updated = current\_timestamp()



The screenshot shows a Databricks SQL interface. At the top, there's a status bar with a play button, a checkmark, the date 'May 17, 2025 (4s)', and the page number '16'. To the right are icons for 'SQL', a trash can, a plus sign, a refresh icon, and a menu icon. The main area contains an SQL query:


```
INSERT INTO transformDataAllCollect
SELECT
  backdrop_path,
  id,
  title,
  original_title,
  overview,
  poster_path,
  media_type,
  adult,
  original_language,
  genre_ids,
  popularity,
  release_date,
  video,
  vote_average,
  vote_count,
  revenue,
  budget,
  runtime,
  company,
  country,
  directors,
  collect_date,
  flag_category,
  current_timestamp(),
  current_timestamp()
FROM transformDataAllCollect_stage_1
WHERE
  popularity <> 0.000
  AND vote_average <> 0.000
  AND vote_count <> 0
  AND revenue <> 0
  AND budget <> 0
  AND runtime <> 0
  AND country <> ''
```

Below the query, there's a section for 'Spark Jobs' with a link to '(4) Spark Jobs'. At the bottom, there's a status bar showing the execution details: 'pyspark.sql.dataframe.DataFrame = [num\_affected\_rows: long, num\_inserted\_rows: long]'.



### 3. Load (Gold Layer)

❖ Cấu hình các bảng dimensions và bảng fact:

 Dim\_movie

```
▶ ✓ May 17, 2025 (2s) 4 SQL ✕ ⌵ ⋮  
  
%sql  
-- tạo bảng dim_movie  
DROP TABLE IF EXISTS dim_movie;  
CREATE TABLE IF NOT EXISTS dim_movie (  
    movie_id BIGINT,  
    title STRING,  
    release_date DATE,  
    overview STRING,  
    time_inserted TIMESTAMP,  
    time_updated TIMESTAMP  
)  
USING DELTA  
LOCATION '/mnt/storageaccountofmovies/gold/dim_movie'  
  
OK
```

 Dim\_genre

```
▶ ✓ May 13, 2025 (2s) 5 SQL ✕ ⌵ ⋮  
  
%sql  
-- tạo bảng dim_genre  
DROP TABLE IF EXISTS dim_genre;  
CREATE TABLE IF NOT EXISTS dim_genre (  
    genre_id BIGINT,  
    genre_name STRING,  
    time_inserted TIMESTAMP,  
    time_updated TIMESTAMP  
)  
USING DELTA  
LOCATION '/mnt/storageaccountofmovies/gold/dim_genre'  
  
OK
```

## Dim\_company

```
▶ ✓ May 17, 2025 (2s) 6

%sql
-- tạo bảng dim_company
DROP TABLE IF EXISTS dim_company;
CREATE TABLE IF NOT EXISTS dim_company (
  company_id BIGINT,
  company_name STRING,
  country STRING,
  time_inserted TIMESTAMP,
  time_updated TIMESTAMP
)
USING DELTA
LOCATION '/mnt/storageaccountofmovies/gold/dim_company'
```

OK

## Dim\_director

```
▶ ✓ May 17, 2025 (2s) 7 SQL 🗑️ ✨ 📄 ⋮

%sql
-- tạo bảng dim_director
DROP TABLE IF EXISTS dim_director;
CREATE TABLE IF NOT EXISTS dim_director (
  director_id BIGINT,
  director_name STRING,
  time_inserted TIMESTAMP,
  time_updated TIMESTAMP
)
USING DELTA
LOCATION '/mnt/storageaccountofmovies/gold/dim_director'
```

OK

## Dim\_date

```
▶ ✓ May 17, 2025 (2s) 8 SQL 🗑️ ✨ 📄 ⋮

%sql
-- tạo bảng dim_date
DROP TABLE IF EXISTS dim_date;
CREATE TABLE IF NOT EXISTS dim_date (
  date_id BIGINT,
  weekStartDate DATE,
  weekEndDate DATE,
  time_inserted TIMESTAMP,
  time_updated TIMESTAMP
)
USING DELTA
LOCATION '/mnt/storageaccountofmovies/gold/dim_date'
```

OK

## Fact\_movies

```
May 17, 2025 (2s) 9 SQL

-- tạo bảng fact_movies
DROP TABLE IF EXISTS fact_movies;
CREATE TABLE IF NOT EXISTS fact_movies (
  movie_id BIGINT,
  genre_id BIGINT,
  company_id BIGINT,
  director_id BIGINT,
  date_id BIGINT,
  popularity DECIMAL(18, 4),
  vote_average DECIMAL(18, 3),
  vote_count BIGINT,
  revenue BIGINT,
  budget BIGINT,
  runtime BIGINT,
  flag_category STRING,
  time_inserted TIMESTAMP,
  time_updated TIMESTAMP
)
USING DELTA
LOCATION '/mnt/storageaccountofmovies/gold/fact_movies'
```

❖ Load dữ liệu vào các bảng Dimensions cố định (chỉ chạy một lần)

- Dim\_genre

```
May 15, 2025 (6s) 1 SQL

%sql
INSERT INTO dim_genre
SELECT
  id,
  name,
  current_timestamp(),
  current_timestamp()
FROM bronzeDataGenres

(4) Spark Jobs
_sqlldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]
```

- Dim\_date (2024/01/01 – 2024/12/31)

Start: ngày bắt đầu tuần, End: ngày kết thúc tuần

```
May 17, 2025 (3s) 5 SQL

%sql
INSERT INTO dim_date
SELECT
  id,
  start,
  end,
  current_timestamp(),
  current_timestamp()
FROM weeks_time_table

(4) Spark Jobs
_sqlldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]
```

❖ Load dữ liệu vào các bảng Dimensions còn lại (lập lịch chạy cùng bảng Fact)

#### 🚦 Stage 1 – Chọn bản ghi mới / đã cập nhật

- Lấy các giá trị DISTINCT từ bảng transformDataAllCollect (title/company/directors).
- Chỉ chọn những bản ghi có time\_updated lớn hơn MAX(time\_updated) hiện tại trong bảng dimension (hoặc ngày mặc định 1900-01-01).
- Kết quả lưu vào bảng stage\_1.

```
▶ May 17, 2025 (4s) 2

%sql
CREATE OR REPLACE TABLE dim_movie_stage_1 AS
SELECT
  DISTINCT title,
  release_date,
  overview
FROM transformdataallcollect
WHERE time_updated > (
  SELECT IFNULL(MAX(time_updated), TO_DATE('1900-01-01'))
  FROM dim_movie
)

▶ (7) Spark Jobs
▶ _sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]
```

```
▶ May 17, 2025 (4s) 7

%sql
CREATE OR REPLACE TABLE dim_company_stage_1 AS
SELECT
  DISTINCT company,
  country
FROM transformdataallcollect
WHERE time_updated > (
  SELECT IFNULL(MAX(time_updated), TO_DATE('1900-01-01'))
  FROM dim_company
)

▶ (7) Spark Jobs
▶ _sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]
```

```
▶ May 17, 2025 (5s) 12

%sql
CREATE OR REPLACE TABLE dim_director_stage_1 AS
SELECT
  DISTINCT directors
FROM transformdataallcollect
WHERE time_updated > (
  SELECT IFNULL(MAX(time_updated), TO_DATE('1900-01-01'))
  FROM dim_director
)

▶ (7) Spark Jobs
▶ _sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]
```

## 🚦 Stage 2 – So sánh & gán ID tạm

- LEFT JOIN bảng stage\_1 với bảng dimension chính theo key (title = title, company = company\_name, directors = director\_name).
- Với mỗi record:
  - Nếu chưa có trong bảng chính → gán ID\_tạm = row\_number() OVER (...)
  - Nếu đã có → giữ lại ID\_cũ từ bảng chính.
  - Lưu kết quả vào stage\_2.

```
May 17, 2025 (4s) 3

CREATE OR REPLACE TABLE dim_movie_stage_2 AS
SELECT
  CASE WHEN goldDim.title IS NULL
    THEN row_number() OVER (ORDER BY silverDim.title)
    ELSE goldDim.movie_id
  END AS movie_id,
  silverDim.title,
  silverDim.release_date,
  silverDim.overview,
  goldDim.title AS gold_title
FROM dim_movie_stage_1 silverDim
LEFT OUTER JOIN dim_movie goldDim
ON silverDim.title = goldDim.title
WHERE goldDim.title IS NULL
   OR silverDim.release_date <> goldDim.release_date
   OR silverDim.overview <> goldDim.overview

(6) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]
```

```
May 17, 2025 (5s) 8

%sql
CREATE OR REPLACE TABLE dim_company_stage_2 AS
SELECT
  CASE WHEN goldDim.company_name IS NULL
    THEN row_number() OVER (ORDER BY silverDim.company)
    ELSE goldDim.company_id
  END AS company_id,
  silverDim.company,
  silverDim.country,
  goldDim.company_name AS gold_company
FROM dim_company_stage_1 silverDim
LEFT OUTER JOIN dim_company goldDim
ON silverDim.company = goldDim.company_name
WHERE goldDim.company_name IS NULL
   OR silverDim.country <> goldDim.country

(6) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]
```

```
May 17, 2025 (4s) 13 SQL 🗑️ ✨ 📄 ⋮

%sql
CREATE OR REPLACE TABLE dim_director_stage_2 AS
SELECT
  row_number() OVER (ORDER BY silverDim.directors) AS director_id,
  silverDim.directors
FROM dim_director_stage_1 silverDim
LEFT OUTER JOIN dim_director goldDim
ON silverDim.directors = goldDim.director_name
WHERE goldDim.director_name IS NULL

(6) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]
```

### 🚦 Stage 3 – Tính ID cuối (surrogate key)

- CROSS JOIN với truy vấn lấy MAX(id) hiện tại từ bảng dimension chính.
  - Với record mới (không khớp), tăng ID\_tạm + MAX(id) để tránh trùng.
  - Với record cũ, giữ nguyên ID\_cũ.
- Kết quả lưu vào stage\_3.

```
▶ ✓ May 17, 2025 (5s) 4

%sql
CREATE OR REPLACE TABLE dim_movie_stage_3 AS
SELECT
  CASE WHEN gold_title IS NULL
    THEN movie_id + MAX_MOVIE
    ELSE movie_id
  END AS movie_id,
  title,
  release_date,
  overview
FROM dim_movie_stage_2
CROSS JOIN (
  SELECT IFNULL(MAX(movie_id), 0) AS MAX_MOVIE
  FROM dim_movie
)

▶ (7) Spark Jobs
▶ _sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]
```

```
▶ ✓ May 17, 2025 (4s) 9

%sql
CREATE OR REPLACE TABLE dim_company_stage_3 AS
SELECT
  CASE WHEN gold_company IS NULL
    THEN company_id + MAX_COMPANY
    ELSE company_id
  END AS company_id,
  company,
  country
FROM dim_company_stage_2
CROSS JOIN (
  SELECT IFNULL(MAX(company_id), 0) AS MAX_COMPANY
  FROM dim_company
)

▶ (7) Spark Jobs
▶ _sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]
```

```
▶ ✓ May 17, 2025 (4s) 14

%sql
CREATE OR REPLACE TABLE dim_director_stage_3 AS
SELECT
  director_id + MAX_DIRECTOR AS director_id,
  directors
FROM dim_director_stage_2
CROSS JOIN (
  SELECT IFNULL(MAX(director_id), 0) AS MAX_DIRECTOR
  FROM dim_director
)

▶ (7) Spark Jobs
▶ _sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]
```

#### 🚦 Stage 4 – MERGE (update/insert) vào bảng dimension

- Cú pháp MERGE INTO dim\_... USING stage\_3 ON key:
  - WHEN MATCHED → UPDATE các cột  
(country, release\_date, overview, time\_updated = current\_timestamp())
  - WHEN NOT MATCHED → INSERT toàn bộ cột  
bao gồm surrogate key, fields, time\_inserted = current\_timestamp(),  
time\_updated = current\_timestamp().

```
May 17, 2025 (3s) 5

%sql
MERGE INTO dim_movie goldDim
USING dim_movie_stage_3 silverDim
ON goldDim.title = silverDim.title
WHEN MATCHED THEN
  UPDATE SET
    goldDim.release_date = silverDim.release_date,
    goldDim.overview = silverDim.overview,
    goldDim.time_updated = current_timestamp()
WHEN NOT MATCHED THEN
  INSERT (
    movie_id,
    title,
    release_date,
    overview,
    time_inserted,
    time_updated
  )
  VALUES (
    silverDim.movie_id,
    silverDim.title,
    silverDim.release_date,
    silverDim.overview,
    current_timestamp(),
    current_timestamp()
  )
```

▶ (6) Spark Jobs

▼ \_sqldf: pyspark.sql.dataframe.DataFrame

```
▶ ✓ May 17, 2025 (3s) 10 SQL 🗑️ ⚡ 📄 ⋮

%sql
MERGE INTO dim_company goldDim
USING dim_company_stage_3 silverDim
ON goldDim.company_name = silverDim.company
WHEN MATCHED THEN
  UPDATE SET
    goldDim.country = silverDim.country,
    goldDim.time_updated = current_timestamp()
WHEN NOT MATCHED THEN
  INSERT (
    company_id,
    company_name,
    country,
    time_inserted,
    time_updated
  )
VALUES (
  silverDim.company_id,
  silverDim.company,
  silverDim.country,
  current_timestamp(),
  current_timestamp()
)

▶ (6) Spark Jobs
▶ 📄 _sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_updated_rows: long ... 2 more fields]
```

Bảng Dim\_director không có cột bổ sung thông tin cho cột directors  
nên không cần dùng MERGE INTO

```
▶ ✓ May 17, 2025 (3s) 15

%sql
INSERT INTO dim_director
SELECT
  director_id,
  directors,
  current_timestamp() AS time_inserted,
  current_timestamp() AS time_updated
FROM dim_director_stage_3

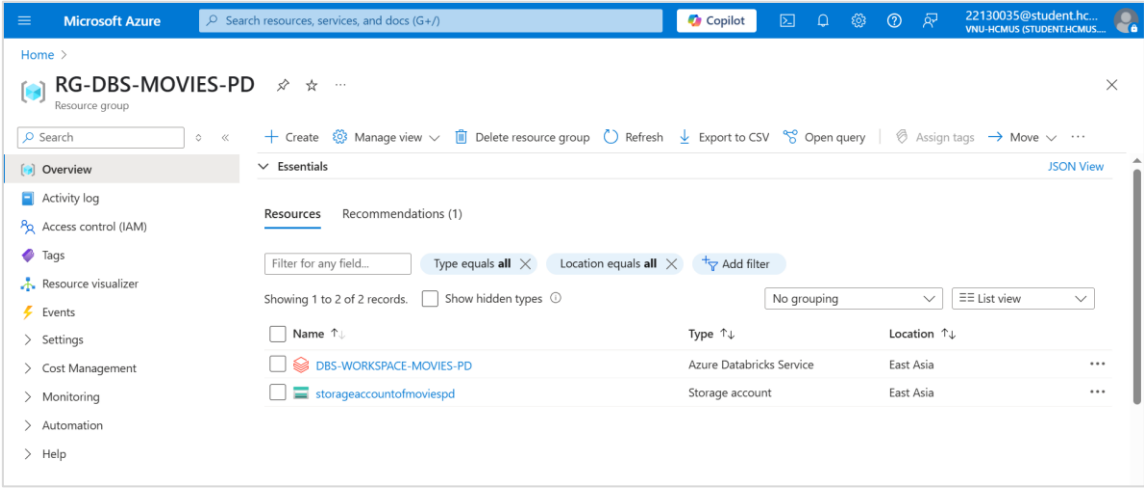
▶ (4) Spark Jobs
▶ 📄 _sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]
```



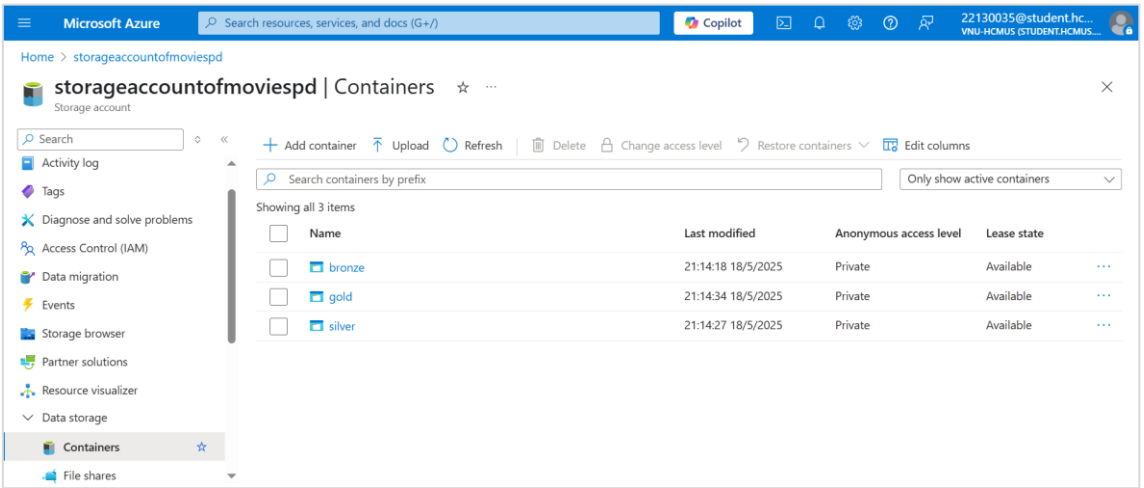
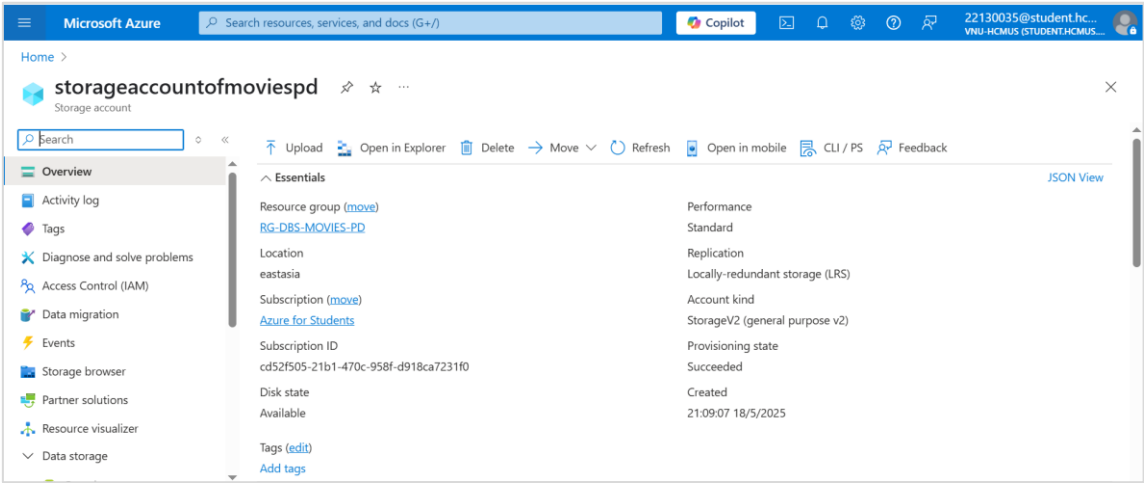
# CI/CD VÀ TỰ ĐỘNG HÓA

## 1. Chuẩn bị hạ tầng Production

### ❖ Resource Group



### ❖ Storage Account



## ❖ Azure Databricks

The screenshot shows the Microsoft Azure portal interface. At the top, there's a search bar and a user profile. The main content area displays the 'DBS-WORKSPACE-MOVIES-PD' workspace overview. On the left, there's a sidebar with navigation options like 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Resource visualizer', 'Settings', 'Automation', and 'Help'. The 'Overview' section is active, showing details such as 'Status: Active', 'Resource group: RG-DBS-MOVIES-PD', 'Location: East Asia', 'Subscription: Azure for Students', and 'Subscription ID: cd52f505-21b1-470c-958f-d918ca7231f0'. On the right, there's a 'JSON View' tab and a list of managed resource groups, including 'databricks-rg-DBS-WORKSPACE-MOVIES-PD-ibapgnlyxu4p4'. The URL 'https://adb-3335360397670417.17.azuredatabricks.net' is also visible.

## ❖ Service Principle

The image displays three separate code snippets, each representing a Python script used to create a service principal in Azure Databricks. Each snippet is shown in a light blue box with a play button icon and a status indicator (a green checkmark) indicating successful execution. The status also shows the date 'May 19, 2025 (<1s)' and a count of '1'.

The first snippet is for a 'bronze' container:

```
container_name = "bronze"
storage_account_name = "storageaccountofmoviespd"
client_id = "262edb55-23bf-44bd-ab83-d1992e9caf22"
tenant_id = "40127cd4-45f3-49a3-b05d-315a43a9f033"
client_secret = "z.c8Q~qbvxRA_tKU0nFgrfuqv5DIU_ah5K4LJasY"
```

The second snippet is for a 'silver' container:

```
container_name = "silver"
storage_account_name = "storageaccountofmoviespd"
client_id = "262edb55-23bf-44bd-ab83-d1992e9caf22"
tenant_id = "40127cd4-45f3-49a3-b05d-315a43a9f033"
client_secret = "z.c8Q~qbvxRA_tKU0nFgrfuqv5DIU_ah5K4LJasY"
```

The third snippet is for a 'gold' container:

```
container_name = "gold"
storage_account_name = "storageaccountofmoviespd"
client_id = "262edb55-23bf-44bd-ab83-d1992e9caf22"
tenant_id = "40127cd4-45f3-49a3-b05d-315a43a9f033"
client_secret = "z.c8Q~qbvxRA_tKU0nFgrfuqv5DIU_ah5K4LJasY"
```

## 2. Kết nối Git với Workspace Dev

The screenshot shows the Databricks Settings page. The left sidebar contains navigation options like Workspace, Recents, Catalog, Workflows, Compute, Marketplace, SQL, SQL Editor, Queries, Dashboards, Genie, Alerts, Query History, and SQL Warehouses. The main content area is titled 'Settings' and has a sub-section 'Linked accounts' under 'Git integration'. It shows a 'GitHub (Linked)' account for 'Tran-Quang-Huy-Huy' with buttons for 'Configure in GitHub' and 'Unlink'.

Microsoft Azure databricks Search data, notebooks, recents, and more... CTRL + P DBS-WORKSPACE-MOVIES

**Settings**

- Workspace admin
  - Appearance
  - Identity and access
  - Security
  - Compute
  - Development
  - Notifications
  - Advanced
- User
  - Profile
  - Preferences
  - Developer
  - Linked accounts

**Linked accounts**

Connect your Databricks account to other services

**Git integration**

**With co-versioned repo**

Databricks Git folders and Repos allow you to clone a remote Git repo, which you can specify when you add a Git folder. [Learn more](#)

**Set your Git provider and credentials**

You can also set your Git provider credentials via API. [Learn more](#)

GitHub (Linked) Tran-Quang-Huy-Huy [Configure in GitHub](#) [Unlink](#)

The screenshot shows the Databricks Workspace page. The left sidebar is the same as the previous screenshot. The main content area is titled 'Workspace' and shows a path 'Workspace > Users > 22130035@student.hcmus.edu.vn > movie-insights'. Below this is a table listing the contents of the repository.

Microsoft Azure databricks Search data, notebooks, recents, and more... CTRL + P DBS-WORKSPACE-MOVIES

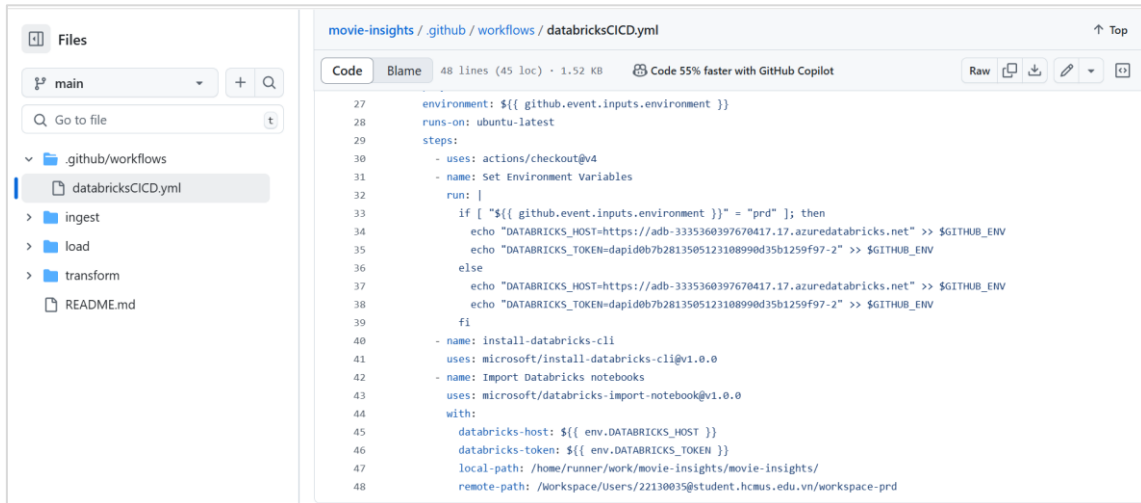
**Workspace**

- Home
- Workspace
- Favorites
- Trash

Workspace > Users > 22130035@student.hcmus.edu.vn > **movie-insights** [Send feedback](#) [Share](#) [Create](#)

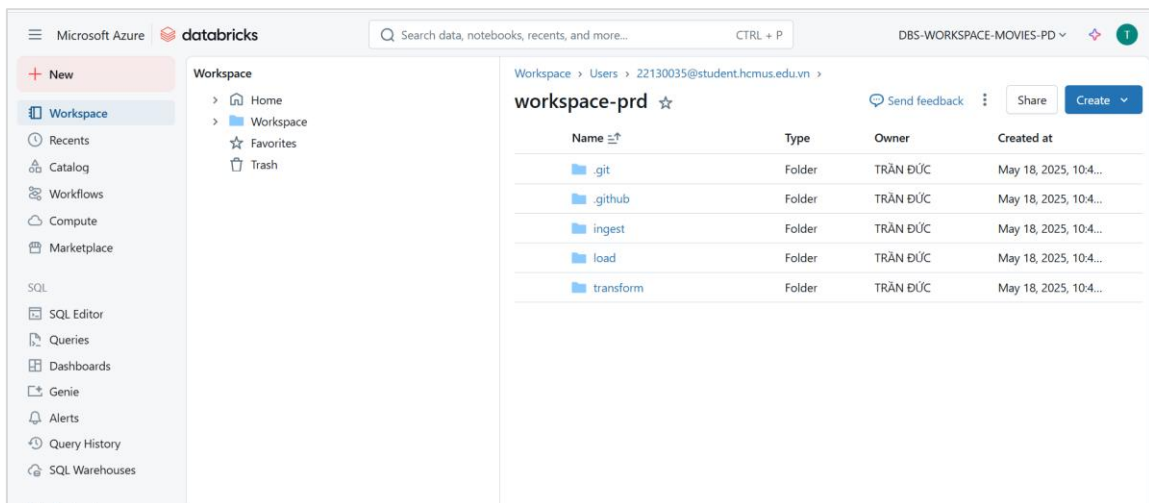
Name	Type	Owner	Created at
.github	Folder	TRẦN ĐỨC	May 25, 2025, 08:56 PM
ingest	Folder	TRẦN ĐỨC	May 18, 2025, 10:27 AM
load	Folder	TRẦN ĐỨC	May 25, 2025, 08:56 PM
transform	Folder	TRẦN ĐỨC	May 19, 2025, 04:34 PM
README.md	File	TRẦN ĐỨC	May 17, 2025, 06:54 PM

### 3. Thiết lập CI/CD Production



The screenshot shows a GitHub repository for 'movie-insights' with a workflow file named 'databricksCICD.yml'. The file is 48 lines long and 1.52 KB. It is a YAML file defining a CI/CD pipeline. The pipeline starts with an 'environment' variable set to 'prd'. It then runs on 'ubuntu-latest' and has two main steps: 'Set Environment Variables' and 'Install Databricks CLI'. The 'Set Environment Variables' step uses 'actions/checkout@v4' and sets 'DATABRICKS\_HOST' and 'DATABRICKS\_TOKEN' based on the environment. The 'Install Databricks CLI' step uses 'microsoft/install-databricks-cli@v1.0.0' and 'microsoft/databricks-import-notebook@v1.0.0' to install the CLI and import notebooks. The pipeline also sets 'local-path' and 'remote-path' for the notebooks.

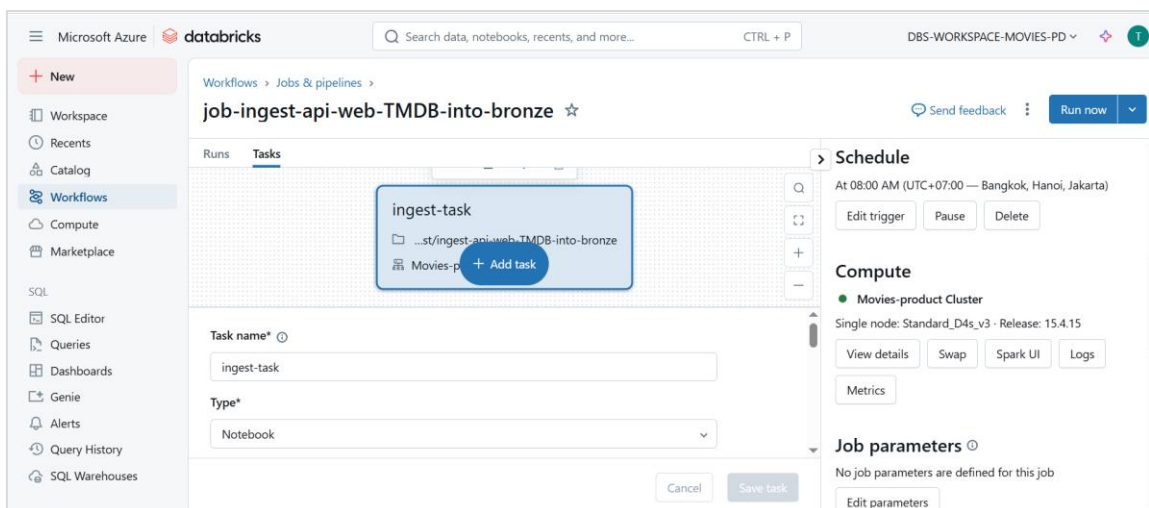
```
27 environment: ${ github.event.inputs.environment }}
28 runs-on: ubuntu-latest
29 steps:
30   - uses: actions/checkout@v4
31   - name: Set Environment Variables
32     run: |
33       if [ "${ github.event.inputs.environment }}" = "prd" ]; then
34         echo "DATABRICKS_HOST=https://adb-3335360397670417.17.azuredatabricks.net" >> $GITHUB_ENV
35         echo "DATABRICKS_TOKEN=dapido7b2813505123108990d35b1259f97-2" >> $GITHUB_ENV
36       else
37         echo "DATABRICKS_HOST=https://adb-3335360397670417.17.azuredatabricks.net" >> $GITHUB_ENV
38         echo "DATABRICKS_TOKEN=dapido7b2813505123108990d35b1259f97-2" >> $GITHUB_ENV
39       fi
40   - name: install-databricks-cli
41     uses: microsoft/install-databricks-cli@v1.0.0
42   - name: Import Databricks notebooks
43     uses: microsoft/databricks-import-notebook@v1.0.0
44     with:
45       databricks-host: ${ env.DATABRICKS_HOST }}
46       databricks-token: ${ env.DATABRICKS_TOKEN }}
47       local-path: /home/runner/work/movie-insights/movie-insights/
48       remote-path: /Workspace/Users/22130035@student.hcmus.edu.vn/workspace-prd
```



The screenshot shows the Databricks workspace interface. The left sidebar contains navigation options like 'Workspace', 'Catalog', 'Workflows', 'Compute', 'Marketplace', 'SQL', 'SQL Editor', 'Queries', 'Dashboards', 'Genie', 'Alerts', 'Query History', and 'SQL Warehouses'. The main area shows the 'workspace-prd' workspace, which is owned by 'TRẦN ĐỨC' and created on May 18, 2025. The workspace contains several folders: '.git', '.github', 'ingest', 'load', and 'transform'.

Name	Type	Owner	Created at
.git	Folder	TRẦN ĐỨC	May 18, 2025, 10:4...
.github	Folder	TRẦN ĐỨC	May 18, 2025, 10:4...
ingest	Folder	TRẦN ĐỨC	May 18, 2025, 10:4...
load	Folder	TRẦN ĐỨC	May 18, 2025, 10:4...
transform	Folder	TRẦN ĐỨC	May 18, 2025, 10:4...

### 4. Tự động hóa Lịch Chạy Job



The screenshot shows the Databricks interface for configuring a job named 'job-ingest-api-web-TMDB-into-bronze'. The job is currently in the 'Tasks' tab. The 'Task name' is 'ingest-task' and the 'Type' is 'Notebook'. The 'Schedule' section shows the job is scheduled to run at 08:00 AM (UTC+07:00) in Bangkok, Hanoi, Jakarta. The 'Compute' section shows the job is running on the 'Movies-product Cluster' with a single node of type 'Standard\_D4s\_v3' and release '15.4.15'. The 'Job parameters' section shows that no job parameters are defined for this job.

**Task name\*** ingest-task

**Type\*** Notebook

**Schedule**

At 08:00 AM (UTC+07:00 — Bangkok, Hanoi, Jakarta)

Edit trigger Pause Delete

**Compute**

● Movies-product Cluster

Single node: Standard\_D4s\_v3 · Release: 15.4.15

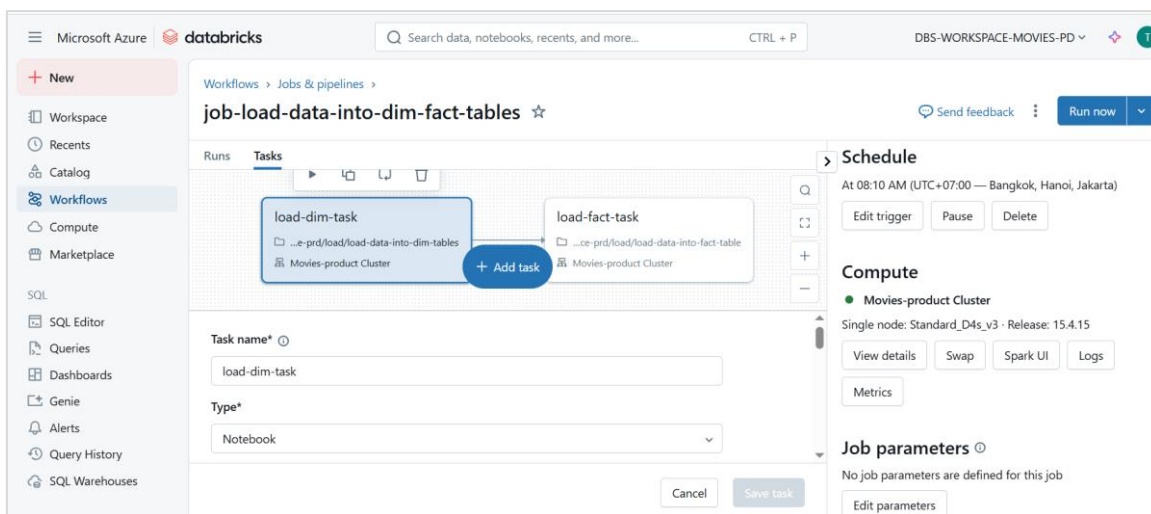
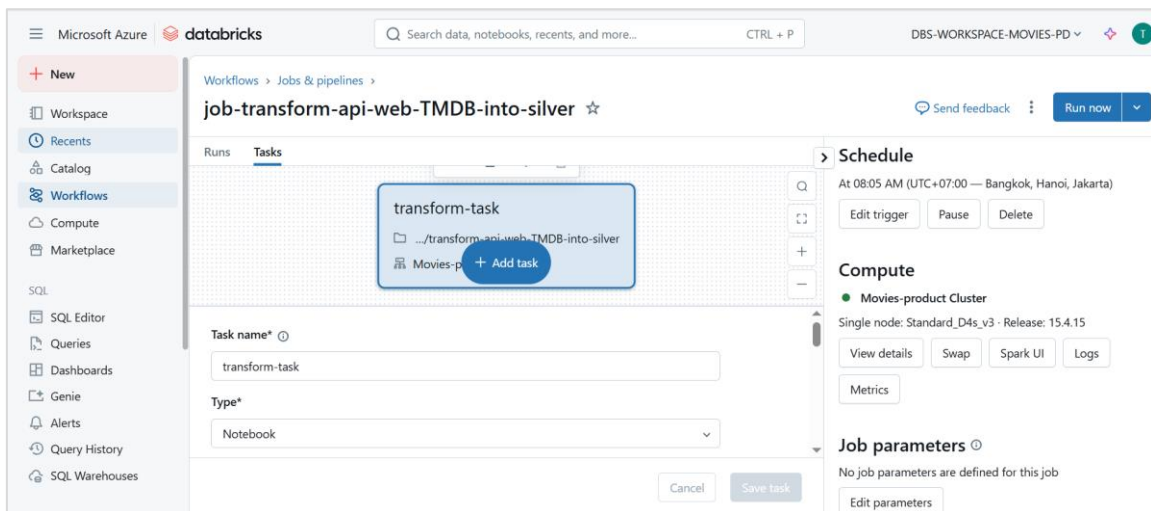
View details Swap Spark UI Logs

Metrics

**Job parameters**

No job parameters are defined for this job

Edit parameters



## 5. Kết nối Power BI với Azure Databricks