Building a Scalable ETL Pipeline for Netflix Content Analysis Using Azure Services.

Introduction:

The Netflix Titles dataset typically contains information about movies and TV shows available on Netflix. Here's a brief explanation of each column in the dataset:

1. **show_id**: A unique identifier for each show or movie in the dataset.
2. **type**: Indicates whether the entry is a movie or a TV show.
3. **title**: The title of the movie or TV show.
4. **director**: The name(s) of the director(s) of the movie or TV show.
5. **cast**: A list of actors and actresses who appear in the movie or TV show.
6. **country**: The country where the movie or TV show was produced or where it is available.
7. **date_added**: The date when the movie or TV show was added to Netflix.
8. **release_year**: The year in which the movie or TV show was originally released.
9. **rating**: The content rating of the movie or TV show (e.g., PG, R, TV-MA).
10. **duration**: The length of the movie (in minutes) or the duration of each episode (in minutes) for TV shows.
11. **listed_in**: The genre or category under which the movie or TV show is listed on Netflix (e.g., Comedy, Drama).
12. **description**: A summary or synopsis of the movie or TV show.

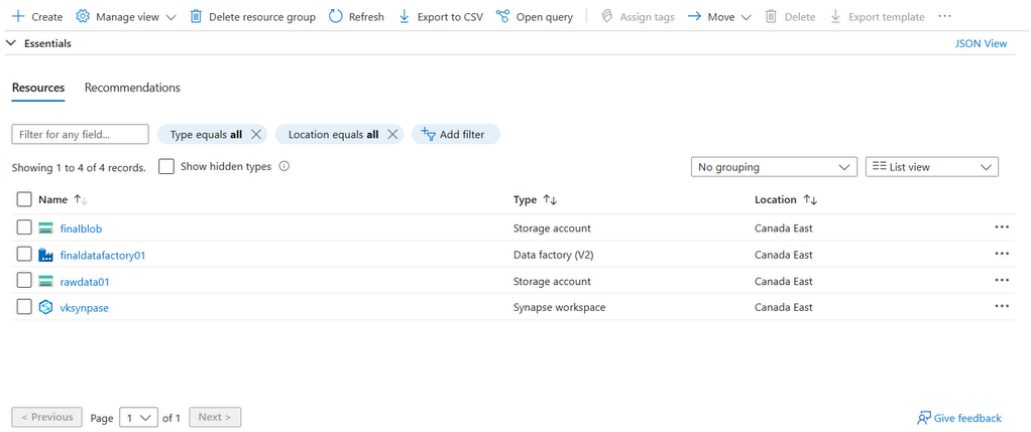
This dataset provides a comprehensive view of Netflix's content catalog, including details about the type, origin, and characteristics of each entry, which can be used for various types of analysis, such as content trends, viewing patterns, and regional availability.

Project Overview:

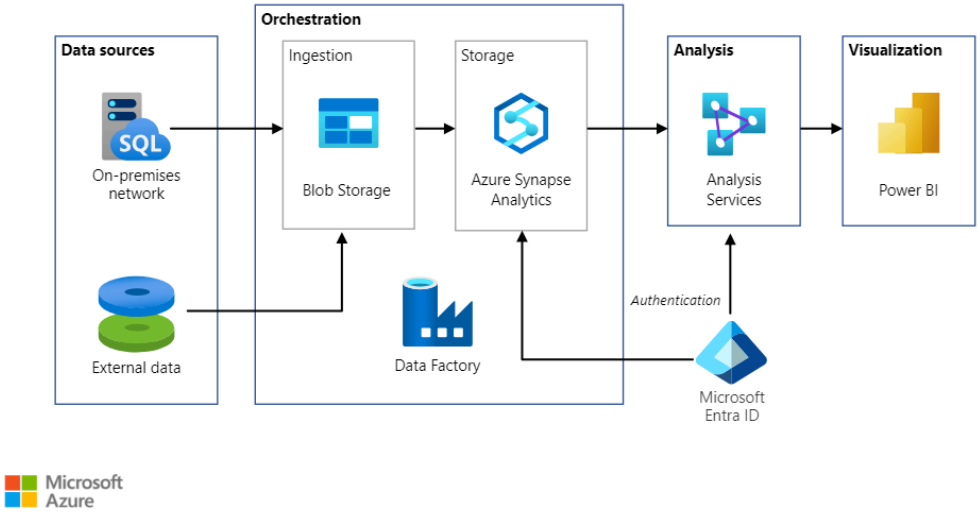
This project involves building an end-to-end data engineering pipeline using the Netflix Titles dataset. The pipeline will utilize SSMS for hosting the data, Azure Blob for storage, Azure Data Factory for orchestrating data movement, and Synapse Analytics for data transformation and cleaning. The processed data will be loaded into Power BI and will be used to create visual reports and dashboards. The solution ensures automation, scalability, and seamless data visualization.

- **Dataset**: Netflix Titles
- **Services Involved**:
  - Azure Blob Storage
  - SQL server management studio
  - Azure Data Factory
  - Azure Synapse Analytics
  - Power BI for visualization

Step 1: Creating the resource group.



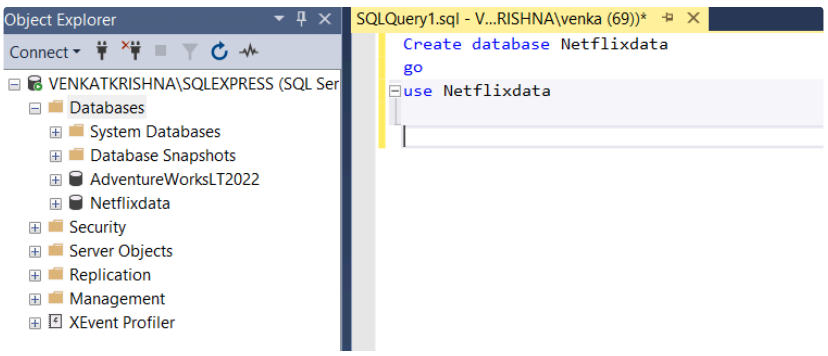
Creating BlobStorage and ADLS gen2 Storage account



Architecture Diagram

Data Ingestion:

Step 1: Install the SQL server Management Studio, connect to a database, and create a new one.

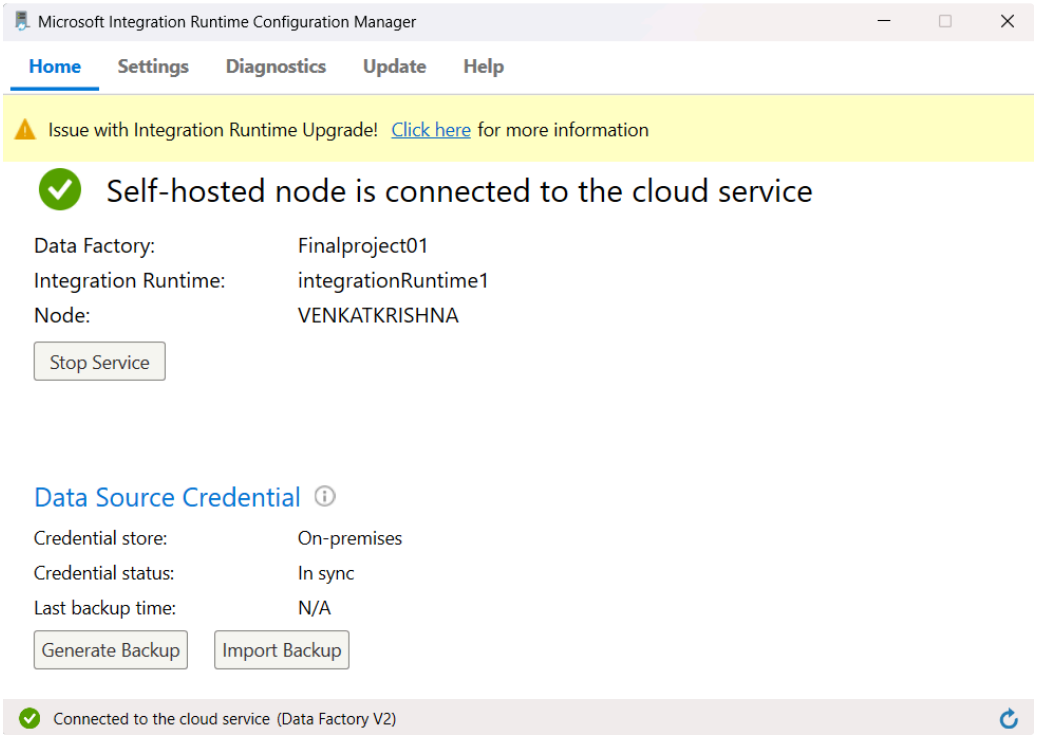


Step 2: Click on that database, import a flat file, upload the CSV file, and run the SQL command.

Step 3: Launch the data factory and perform copy activity from the SQL server to blob storage

To do that we have to create a self-integration run time because this is an on-premises SQL server and we have to self-host it

To do that we have to install the integration run time software enter the key and register for it. once you complete it.



Use Azure Data Factory to create a pipeline and perform activities that transfer the CSV file from the ADLS gen2 for processing in Synapse.

Lookup Activity

- **Purpose**: Check metadata or file existence in the source location.
- **Configuration**:
  - Drag the Lookup activity onto the pipeline canvas.
  - Set up the dataset pointing to the source location of `netflix_titles.csv`.
  - Optionally, configure to read specific properties (e.g., first row) for validation.

2. Get Metadata Activity

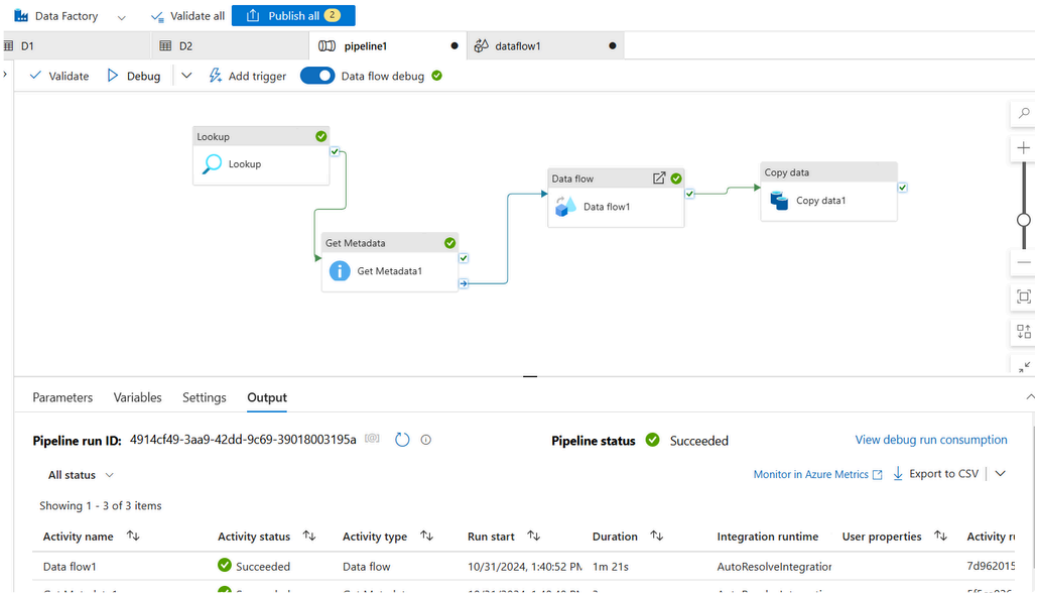
- **Purpose**: Retrieve metadata information (e.g., file size, last modified date) before processing.

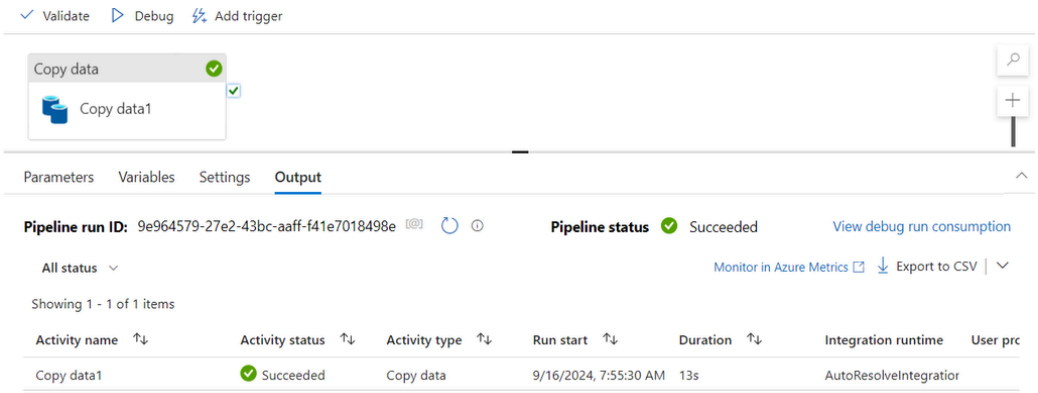
3. Data Flow Activity

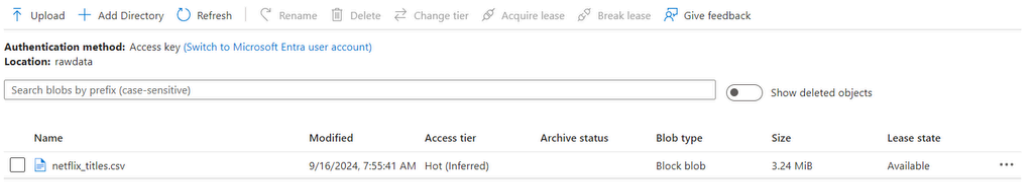
- **Purpose**: Transform and prepare data for the bronze layer.
- **Configuration**:
  - **Source**: Set `netflix_titles.csv` as the source.
  - **Transformations**: Applied filter data, Joins, and renaming columns transformations
  - **Sink**: Set ADLS Gen2 as the sink (bronze layer).

Copy Data Activity (Stage Data to ADLS Gen2)

- **Purpose**: Copy data from the source to a staging area in ADLS Gen2.







[Data Transformation:]

Step 1: Bronze Layer (Raw Data Ingestion)

Objective: Collect raw data and store it in a data lake (Azure Data Lake Storage - ADLS) without any modifications.

In this project:

- We assume you have already ingested the Netflix data into Azure Data Lake Storage (ADLS) Gen2.
- The data should be stored in a folder structure that makes it easy to identify as the Bronze layer.

```
/bronze/netflix/raw/
```

The data in this layer should be treated as immutable and will serve as the source for all downstream transformations.

Step 2: Silver Layer (Data Cleaning and Enrichment)

Objective: Perform data cleaning and minimal transformations to make the data reliable and consistent. This involves removing duplicates, handling null values, and casting data types.

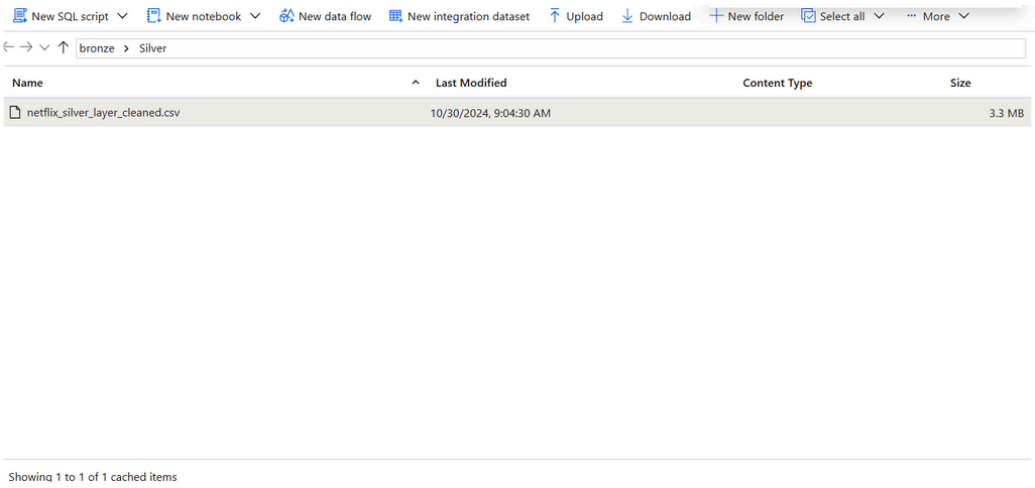
Steps:

1. **Load Data from Bronze Layer**:
   - Use PySpark to read data from the Bronze layer. PySpark, a Spark API for Python, is available in Synapse Analytics and enables distributed data processing.
2. **Data Cleaning**:
   - Drop duplicates, remove rows with null values in critical columns (like `title`), and trim whitespace from string columns.
   - Convert columns to appropriate data types (e.g., cast `date_added` to date format and `release_year` to integer).
3. **Save Data to Silver Layer**:
   - Save the cleaned data as Parquet files (optimized for analytical querying) in the Silver layer.

PySpark Code for Bronze to Silver Transformation:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, trim

# Initialize Spark session in Synapse
spark = SparkSession.builder.getOrCreate()

# Load raw data from Bronze layer
bronze_df = spark.read.format("csv") \
               .option("header", "true") \
               .load("abfss://Bronze@rawad;sgen2.dfs.core.windows.net/bronze/netflix/raw/")

# Clean and transform the data
silver_df = bronze_df.dropna(subset=["title"]) \
                     .dropDuplicates() \
                     .withColumn("title", trim(col("title"))) \
                     .withColumn("date_added", col("date_added").cast("date")) \
```

```
                    .withColumn("release_year", col("release_year").cast("int"))

# Save to Silver layer
silver_df.write.format("parquet") \
               .mode("overwrite") \
               .save("abfss://Silver@rawadlsgen2.dfs.core.windows.net/silver/netflix/cleaned/")

```



Key Points:

- **Drop Duplicates**: Ensures each record is unique.
- **Handle Nulls**: Avoids issues during analysis by excluding rows missing essential data.
- **Data Type Casting**: Necessary for accurate reporting and querying.

Step 3: Gold Layer (Aggregation and Analytics-Ready Data)

Objective: Further transform and aggregate data to make it ready for analytics and reporting. This layer should contain pre-aggregated data (like counts and summaries) to reduce processing time for reporting tools.

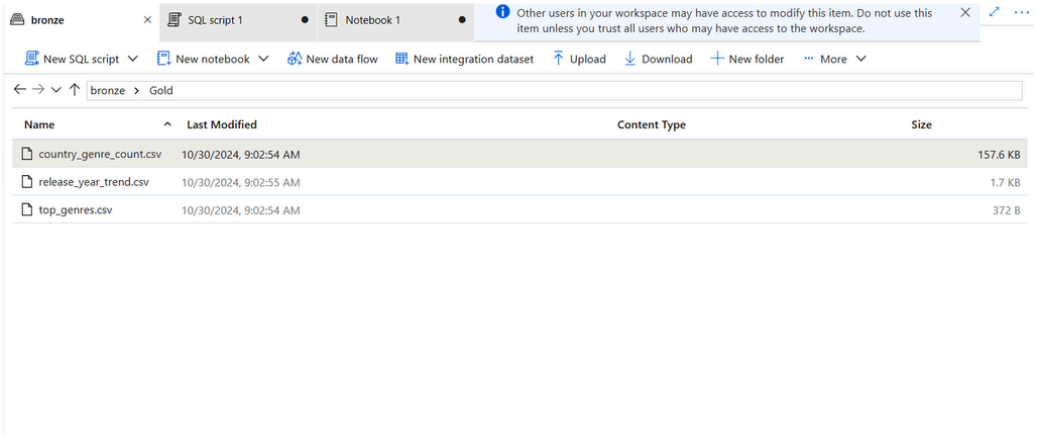
Steps:

1. **Load Data from Silver Layer**:
   - Read the cleaned and transformed data from the Silver layer.
2. **Data Aggregations**:
   - **Genre Count**: Count the number of titles for each genre.
   - **Yearly Release Count**: Count the number of titles released each year.
   - **Monthly Addition Count**: Count the number of titles added per month.
3. **Save Aggregated Data to Gold Layer**:
   - Save these aggregated datasets to the Gold layer, ready for Power BI reporting.

PySpark Code for Silver to Gold Transformation:

```
# Load cleaned data from Silver layer
silver_df = spark.read.format("parquet") \
               .load("abfss://<container>@<storage_account>.dfs.core.windows.net/silver/netflix/cleaned/")

# Transformation examples
# 1. Count by Genre
```

```
genre_count_df = silver_df.groupBy("listed_in").count()

# 2. Count of Titles by Release Year
year_count_df = silver_df.groupBy("release_year").count()

# 3. Monthly count of Titles added
monthly_count_df = silver_df.groupBy("date_format(date_added, 'yyyy-MM')").count()

# Save to Gold layer
genre_count_df.write.format("parquet") \
                  .mode("overwrite") \
                  .save("abfss://Gold@rawadlsgen2.dfs.core.windows.net/gold/netflix/genre_count/")

year_count_df.write.format("parquet") \
                 .mode("overwrite") \
                 .save("abfss://Gold@rawadlsgen2.dfs.core.windows.net/gold/netflix/year_count/")

monthly_count_df.write.format("parquet") \
                      .mode("overwrite") \
                      .save("abfss://Gold@rawadlsgen2.dfs.core.windows.net/gold/netflix/monthly_count/")

```



Key Points:

- Aggregated datasets make reporting faster, as Power BI can directly use these precomputed metrics.
- **File Format**: Saving as Parquet files ensures efficient storage and faster query performance.

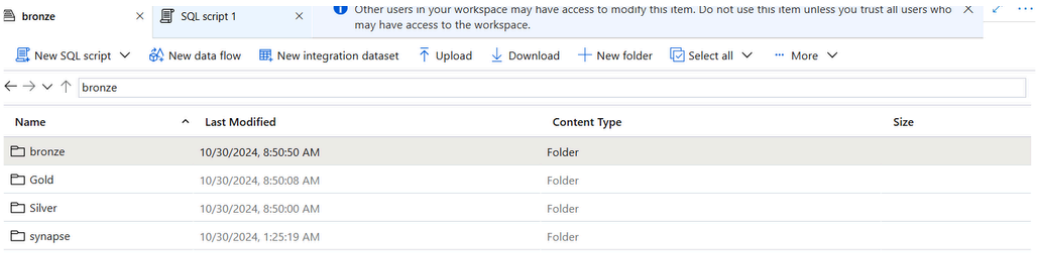
Step 4: Loading into Synapse SQL Pool (for Power BI Reporting)

To allow Power BI to connect to and query the data in the Gold layer, create external tables in Synapse SQL Pool that point to the data stored in ADLS.

SQL Script to Create External Tables:

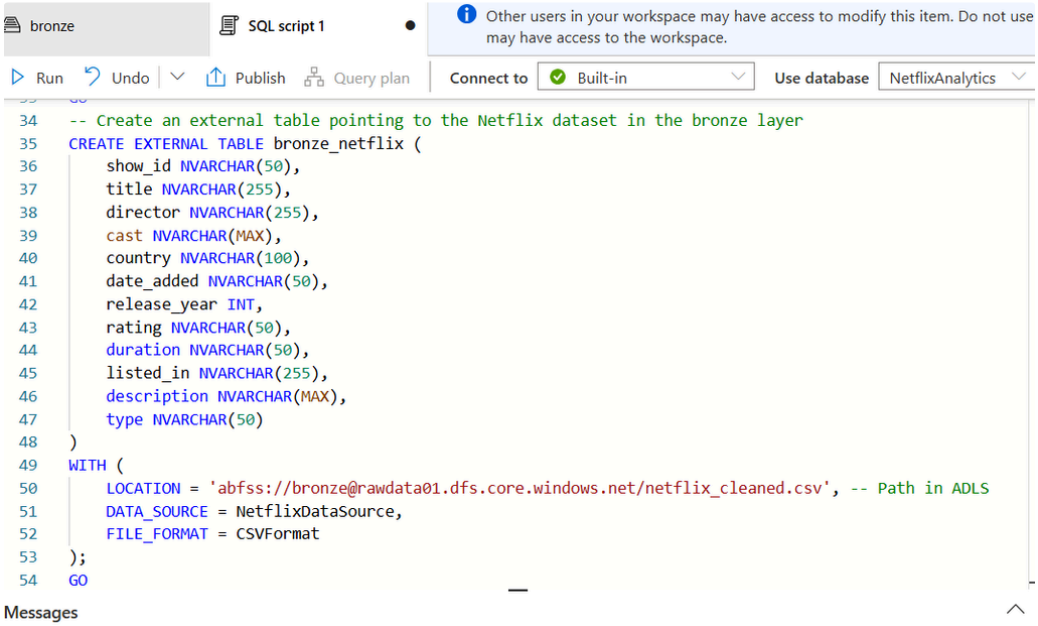
```
CREATE DATABASE NetflixAnalytics;
CREATE EXTERNAL DATA SOURCE NetflixDataLake
WITH (LOCATION = 'abfss://Gold@rawadlsgen2.dfs.core.windows.net');

CREATE EXTERNAL TABLE Netflix_Genre_Count (
    genre STRING,
```

```
    count INT
)
WITH (
    LOCATION = '/gold/netflix/genre_count/',
    DATA_SOURCE = NetflixDataLake,
    FILE_FORMAT = SynapseParquetFormat
);

CREATE EXTERNAL TABLE Netflix_Year_Count (
    release_year INT,
    count INT
)
WITH (
    LOCATION = '/gold/netflix/year_count/',
    DATA_SOURCE = NetflixDataLake,
    FILE_FORMAT = SynapseParquetFormat
);

CREATE EXTERNAL TABLE Netflix_Monthly_Count (
    month STRING,
    count INT
)
WITH (
    LOCATION = '/gold/netflix/monthly_count/',
    DATA_SOURCE = NetflixDataLake,
    FILE_FORMAT = SynapseParquetFormat
);

```

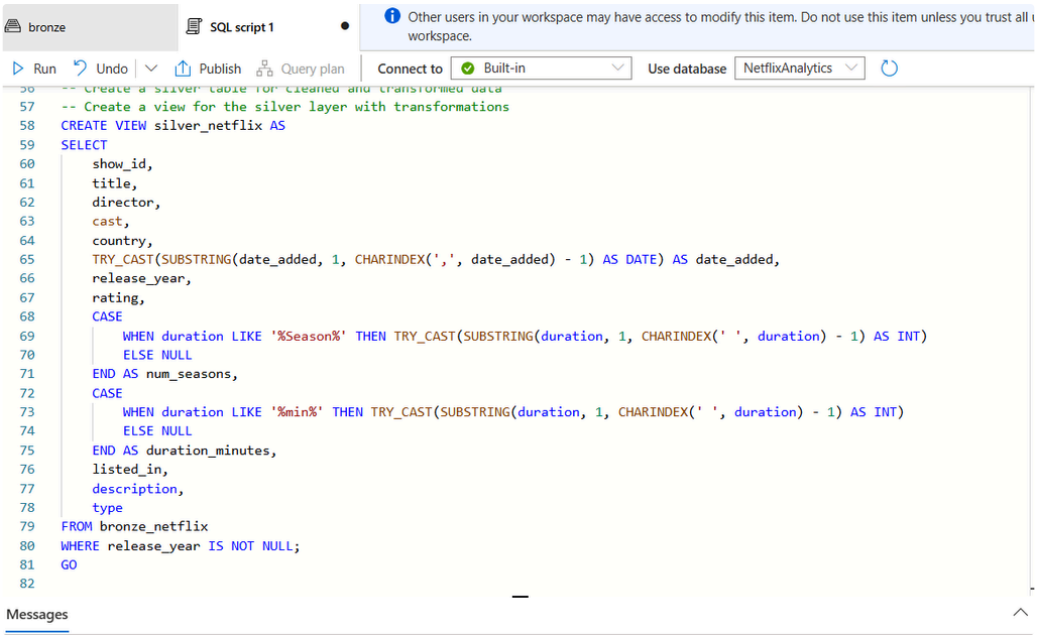


Scenario 2: Data Modelling in Azure Synapse.

Step 1: Load Data to Synapse (Bronze Layer)



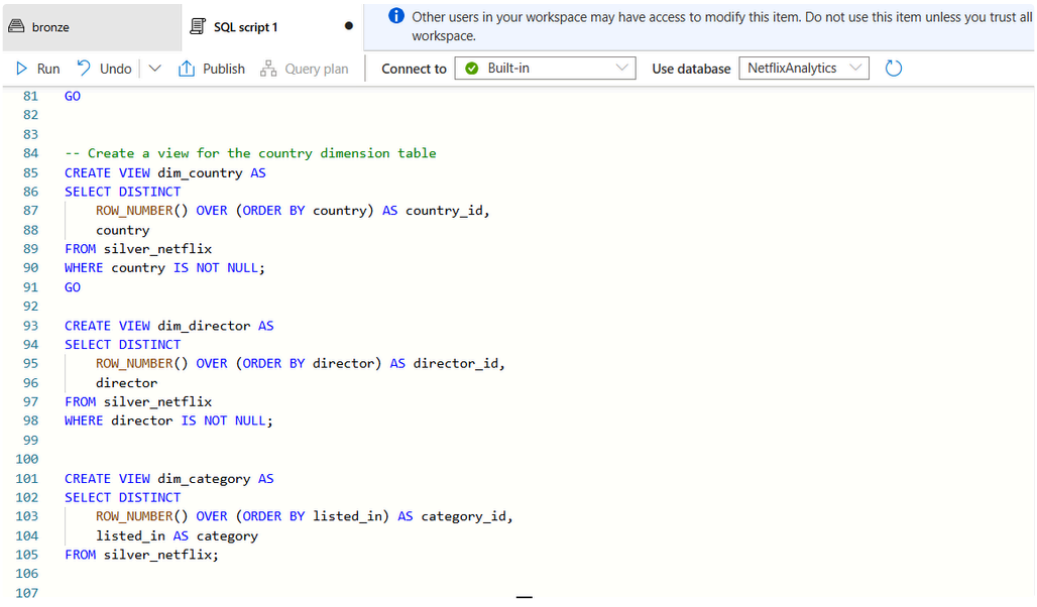
Step 2: Transform Data for Silver Layer



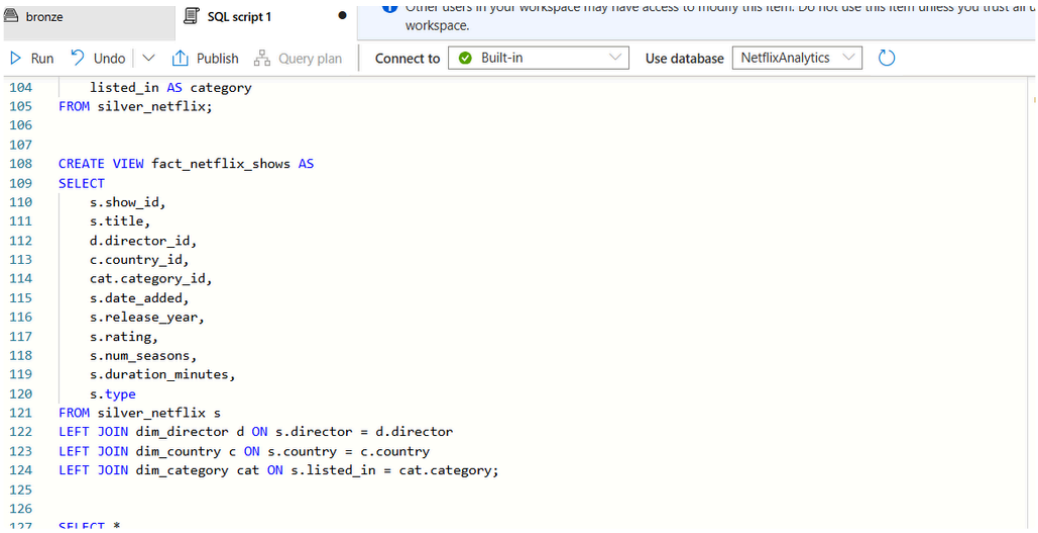
Step 3: Define Dimension and Fact Tables for Gold Layer

Create `dim_country` Table

1. **Objective**: Define a dimension table for the `Country`, director and category.



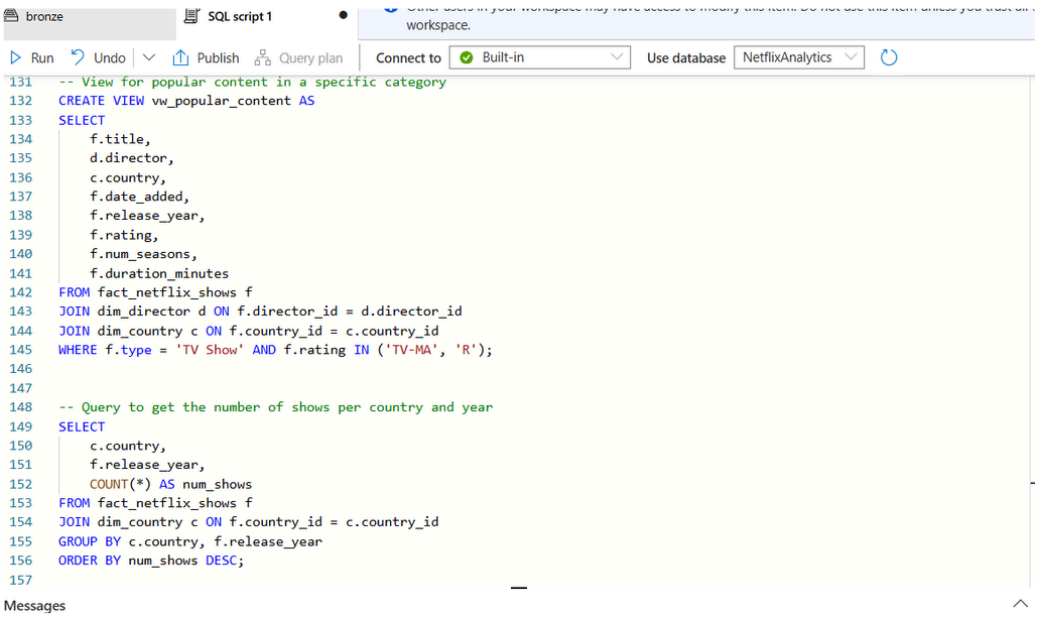
Step 4: . Create `fact_netflix_shows` Table



Step 4: Optimize the Fact Table

```
SELECT *
FROM fact_netflix_shows
WHERE release_year = 2021;
```

Step 5: Create Analytical Views



Step 6: Run an Example Query on the Fact Table

```
SELECT
    c.country,
    f.release_year,
    COUNT(*) AS num_shows
FROM fact_netflix_shows f
```

```
JOIN dim_country c ON f.country_id = c.country_id
GROUP BY c.country, f.release_year
ORDER BY num_shows DESC;
```

Explanation:

- **External Tables**: Allow Synapse SQL Pool to access data directly from ADLS without copying it into Synapse.
- **Data Source**: Points to the data lake storage account.
- **File Format**: Set up as Parquet for optimized analytics performance.

Step 5: Reporting with Power BI

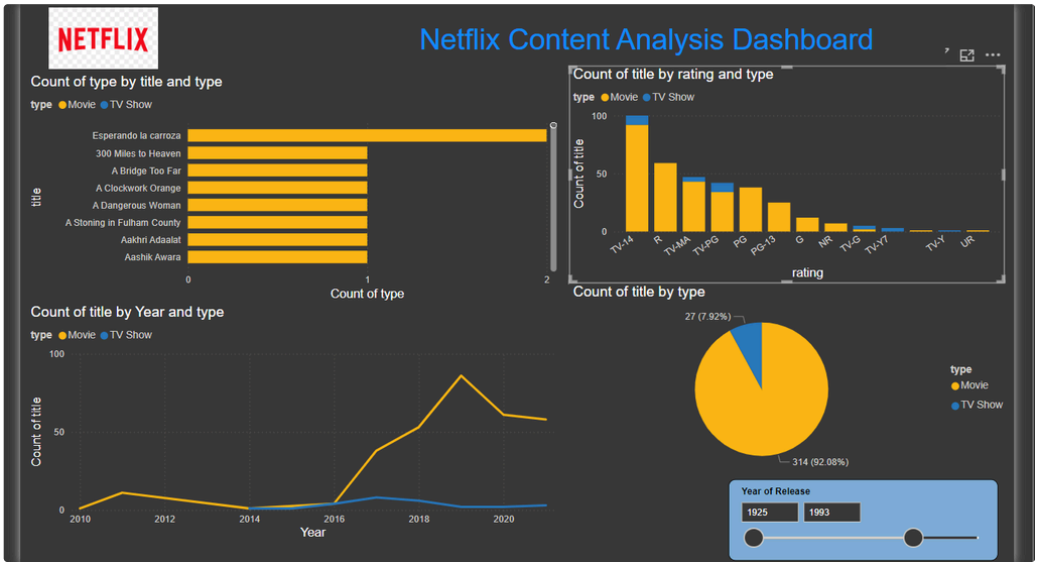
With data in the Gold layer and accessible through Synapse SQL Pool, you can connect Power BI to Synapse and create visualizations.

Power BI Report Examples:

1. **Genre Distribution**:
   - Use a bar or pie chart to show the distribution of content across genres.
2. **Titles by Release Year**:
   - A line chart showing the number of titles released each year, highlighting growth trends.
3. **Monthly Additions**:
   - A line or bar chart showing titles added monthly, useful for observing seasonal patterns.

Power BI Connection Steps:

1. Open Power BI Desktop and choose "Get Data."
2. Select "Azure Synapse Analytics" as the data source.
3. Connect to the Synapse SQL Pool and select the external tables created.



Why Choose the Netflix Dataset for Business Analysis?

- **Consumer Insights**: The dataset provides detailed information on content type, genres, and release years, helping to understand viewer preferences and popular genres.
- **Global Content Distribution**: With data on content availability across countries, this dataset allows analysis of geographical content distribution, supporting market expansion strategies.

- **Trend Analysis**: The release year and date added fields enable insights into content release patterns and trends, aiding in decisions about content production and licensing.

Key Findings from the Netflix Dataset

- **Top Genres and Content-Type**: Identifies the most popular genres and distinguishes between movies and TV shows, supporting strategic content curation.
- **Country-Specific Preferences**: Shows the diversity of content by country, which helps in targeting and localizing content offerings.
- **Content Growth Over Time**: Reveals trends in content additions by year, reflecting Netflix's expansion and investment in original content and licensing.

Common errors while working on this Project:

- **Integration Runtime Issues**: Failure to correctly configure the self-hosted integration runtime could disrupt data movement.
- **Data Format Mismatches**: Inconsistent or incorrect CSV file formats might cause data parsing errors.
- **Data Path Errors**: Incorrect file paths in code might result in failures during data reading/writing operations.
- **SQL Server Connectivity**: Connectivity issues between SQL Server Management Studio and Azure Data Factory could halt the pipeline.
- **Transformation Errors**: Errors in SQL transformations or PySpark code could lead to inaccurate or incomplete data.
- **Performance Issues**: Large datasets may cause performance bottlenecks in data processing or transformation steps.
- **Security Concerns**: Inadequate security configurations may expose data to unauthorized access or breaches.
- **Visualization Inconsistencies**: Incorrect data loading into Power BI could lead to misleading visualizations or reports.

Conclusion

This project successfully demonstrates the construction of a robust end-to-end data engineering pipeline using the Netflix Titles dataset. By leveraging a combination of powerful tools and services—Azure Blob Storage, SQL Server Management Studio, Azure Data Factory, Azure Synapse Analytics, and Power BI—the pipeline achieves efficient data ingestion, transformation, and visualization.

Key Achievements:

1. **Data Ingestion and Storage**: The pipeline effectively integrates SQL Server for hosting and Azure Blob Storage for scalable data storage, ensuring smooth data transfer and storage.
2. **Data Transformation**: Utilizing Azure Synapse Analytics and Spark, the project converts raw data into clean, structured formats like Delta and Parquet, enhancing data processing efficiency and performance.
3. **Automation and Scalability**: By incorporating Azure Data Factory, the project automates data movement and transformation processes, supporting scalability and flexibility in handling large datasets.
4. **Visualization**: The final integration with Power BI enables meaningful data visualization, facilitating insightful analysis and reporting.

Future Enhancements:

To further enhance the project, consider adding:

- Detailed data quality checks and validation processes to ensure accuracy and consistency.
- Specifics on the types of visualizations and dashboards in Power BI to better illustrate the insights derived from the data.
- Performance optimization strategies such as query optimization resource scaling and error handling mechanisms to address potential issues and improve pipeline efficiency.

Overall, this project showcases a well-rounded approach to building a data engineering pipeline, with clear steps and effective use of Azure services, positioning it as a solid foundation for advanced data analytics and reporting Using Azure tools.