

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.

[+ Create](#)
[Manage view](#)
[Delete resource group](#)
[Refresh](#)
[Export to CSV](#)
[Open query](#)
[Assign tags](#)
[Move](#)
[Delete](#)
[Export template](#)
[JSON View](#)

[Essentials](#)

[Resources](#)
[Recommendations](#)

Filter for any field...
 Type equals **all**
Location equals **all**
Add filter

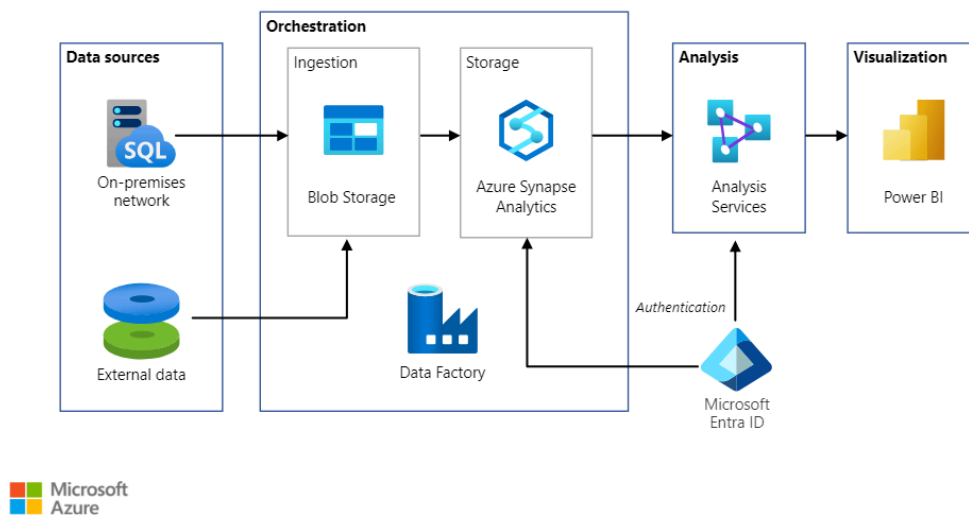
Showing 1 to 4 of 4 records.
 Show hidden types
 No grouping
 List view

Name ↑↓	Type ↑↓	Location ↑↓
<input type="checkbox"/> finalblob	Storage account	Canada East
<input type="checkbox"/> finaldatafactory01	Data factory (V2)	Canada East
<input type="checkbox"/> rawdata01	Storage account	Canada East
<input type="checkbox"/> vksynapse	Synapse workspace	Canada East

< Previous
 Page 1 of 1
 Next >

[Give feedback](#)

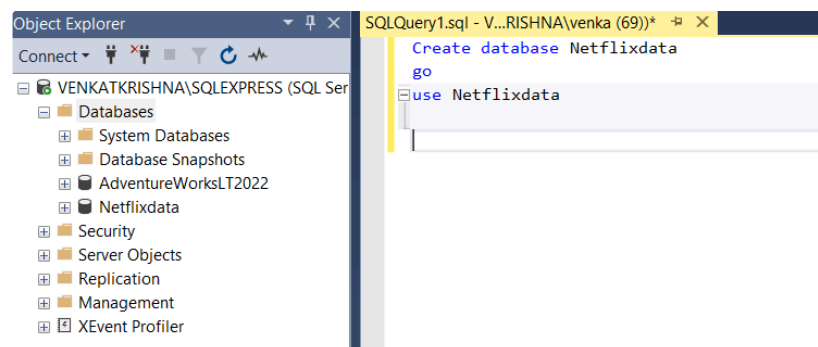
Creating BlobStorage and ADLS gen2 Storage account



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.

Microsoft Integration Runtime Configuration Manager

Home

Settings

Diagnostics

Update

Help

Issue with Integration Runtime Upgrade! [Click here](#) for more information

Self-hosted node is connected to the cloud service

Data Factory: Finalproject01

Integration Runtime: integrationRuntime1

Node: VENKATKRISHNA

Stop Service

Data Source Credential

Credential store: On-premises

Credential status: In sync

Last backup time: N/A

Generate Backup

Import Backup

Connected to the cloud service (Data Factory V2)

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

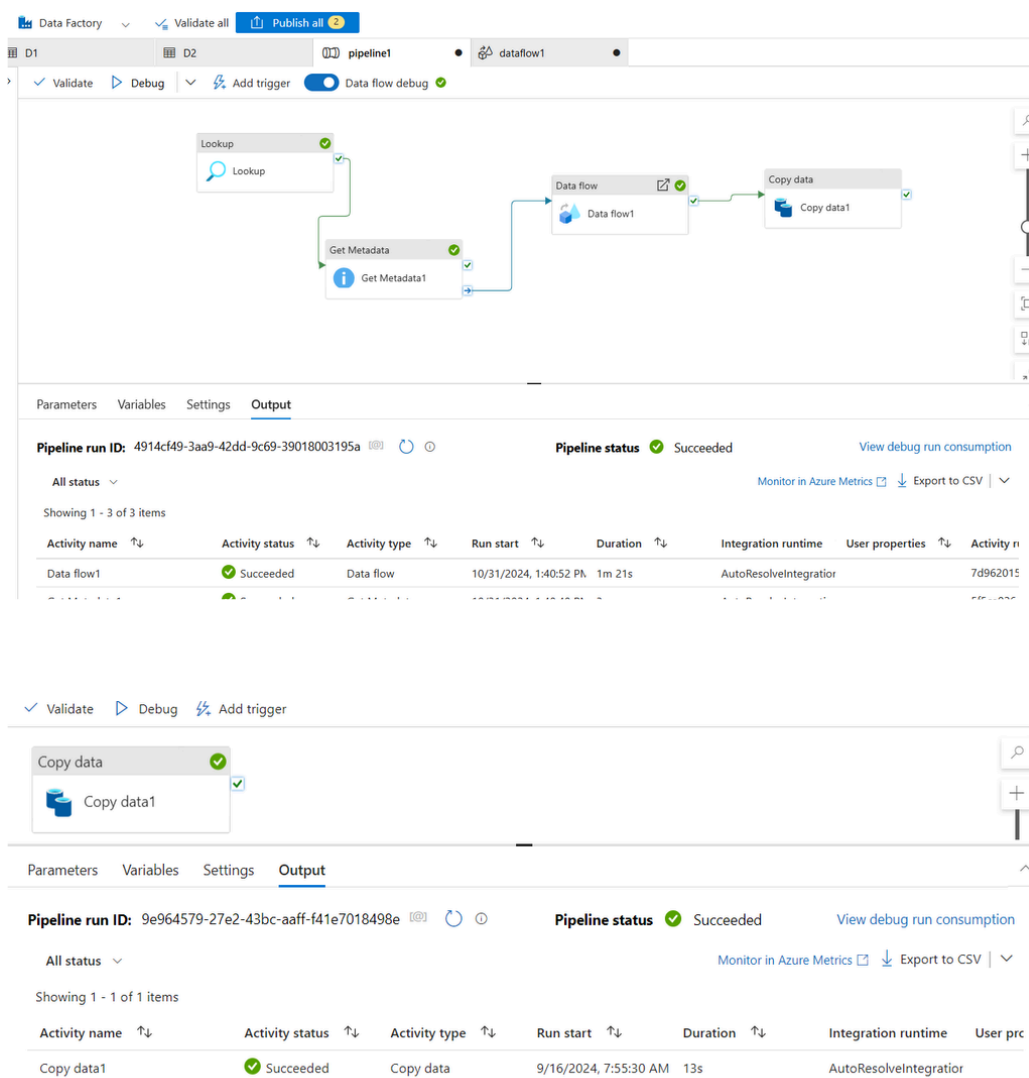
- **Purpose:** Retrieve metadata information (e.g., file size, last modified date) before processing.

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.



Upload + Add Directory Refresh Rename Delete Change tier Acquire lease Break lease Give feedback						
Authentication method: Access key (Switch to Microsoft Entra user account) Location: rawdata						
<input type="text" value="Search blobs by prefix (case-sensitive)"/> <input type="checkbox"/> Show deleted objects						
Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
<input type="checkbox"/> netflix_titles.csv	9/16/2024, 7:55:41 AM	Hot (Inferred)		Block blob	3.24 MiB	Available ***

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.

```
1 /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:

```
1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import col, trim
3
4 # Initialize Spark session in Synapse
5 spark = SparkSession.builder.getOrCreate()
6
7 # Load raw data from Bronze layer
8 bronze_df = spark.read.format("csv") \
9     .option("header", "true") \
10    .load("abfss://Bronze@rawad;sgen2.dfs.core.windows.net/bronze/netflix/raw/")
11
12 # Clean and transform the data
13 silver_df = bronze_df.dropna(subset=["title"]) \
14    .dropDuplicates() \
15    .withColumn("title", trim(col("title"))) \
16    .withColumn("date_added", col("date_added").cast("date")) \
```

```

17         .withColumn("release_year", col("release_year").cast("int"))
18
19 # Save to Silver layer
20 silver_df.write.format("parquet") \
21     .mode("overwrite") \
22     .save("abfss://Silver@rawadlsgen2.dfs.core.windows.net/silver/netflix/cleaned/")
23

```

New SQL script

New notebook

New data flow

New integration dataset

Upload

Download

New folder

Select all

More

← → ↕ bronze > Silver

Name	Last Modified	Content Type	Size
netflix_silver_layer_cleaned.csv	10/30/2024, 9:04:30 AM		3.3 MB

Showing 1 to 1 of 1 cached items

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:

```

1 # Load cleaned data from Silver layer
2 silver_df = spark.read.format("parquet") \
3     .load("abfss://<container>@<storage_account>.dfs.core.windows.net/silver/netflix/cleaned/")
4
5 # Transformation examples
6 # 1. Count by Genre

```

```

7 genre_count_df = silver_df.groupBy("listed_in").count()
8
9 # 2. Count of Titles by Release Year
10 year_count_df = silver_df.groupBy("release_year").count()
11
12 # 3. Monthly count of Titles added
13 monthly_count_df = silver_df.groupBy("date_format(date_added, 'yyyy-MM')").count()
14
15 # Save to Gold layer
16 genre_count_df.write.format("parquet") \
17     .mode("overwrite") \
18     .save("abfss://Gold@rawadlsgen2.dfs.core.windows.net/gold/netflix/genre_count/")
19
20 year_count_df.write.format("parquet") \
21     .mode("overwrite") \
22     .save("abfss://Gold@rawadlsgen2.dfs.core.windows.net/gold/netflix/year_count/")
23
24 monthly_count_df.write.format("parquet") \
25     .mode("overwrite") \
26     .save("abfss://Gold@rawadlsgen2.dfs.core.windows.net/gold/netflix/monthly_count/")
27

```

The screenshot shows the Azure Data Lake Storage Explorer interface. The top bar indicates the workspace is 'bronze'. Below the top bar, there are tabs for 'SQL script 1' and 'Notebook 1'. A notification banner states: 'Other users in your workspace may have access to modify this item. Do not use this item unless you trust all users who may have access to the workspace.' Below the notification, there are buttons for 'New SQL script', 'New notebook', 'New data flow', 'New integration dataset', 'Upload', 'Download', 'New folder', and 'More'. The breadcrumb navigation shows 'bronze > Gold'. Below the navigation, there is a table listing the files in the Gold layer:

Name	Last Modified	Content Type	Size
country_genre_count.csv	10/30/2024, 9:02:54 AM		157.6 KB
release_year_trend.csv	10/30/2024, 9:02:55 AM		1.7 KB
top_genres.csv	10/30/2024, 9:02:54 AM		372 B

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:

```

1 CREATE DATABASE NetflixAnalytics;
2 CREATE EXTERNAL DATA SOURCE NetflixDataLake
3 WITH (LOCATION = 'abfss://Gold@rawadlsgen2.dfs.core.windows.net');
4
5 CREATE EXTERNAL TABLE Netflix_Genre_Count (
6     genre STRING,

```

```

7      count INT
8  )
9  WITH (
10     LOCATION = '/gold/netflix/genre_count/',
11     DATA_SOURCE = NetflixDataLake,
12     FILE_FORMAT = SynapseParquetFormat
13 );
14
15 CREATE EXTERNAL TABLE Netflix_Year_Count (
16     release_year INT,
17     count INT
18 )
19 WITH (
20     LOCATION = '/gold/netflix/year_count/',
21     DATA_SOURCE = NetflixDataLake,
22     FILE_FORMAT = SynapseParquetFormat
23 );
24
25 CREATE EXTERNAL TABLE Netflix_Monthly_Count (
26     month STRING,
27     count INT
28 )
29 WITH (
30     LOCATION = '/gold/netflix/monthly_count/',
31     DATA_SOURCE = NetflixDataLake,
32     FILE_FORMAT = SynapseParquetFormat
33 );
34

```

```

12 CREATE DATABASE NetflixAnalytics;
13 CREATE EXTERNAL DATA SOURCE NetflixDataLake
14 WITH (LOCATION = 'abfss://Gold@rawadlsgen2.dfs.core.windows.net');
15
16 CREATE EXTERNAL TABLE Netflix_Genre_Count (
17     genre STRING,
18     count INT
19 )
20 WITH (
21     LOCATION = '/gold/netflix/genre_count/'

```

Results Messages



No results to show

Your query yielded no displayable results

00:00:02 Query executed successfully.

bronze

SQL script 1

Other users in your workspace may have access to modify this item. Do not use this item unless you trust all users who may have access to the workspace.

New SQL script

New data flow

New integration dataset

Upload

Download

New folder

Select all

More

← → ↕ bronze

Name	Last Modified	Content Type	Size
bronze	10/30/2024, 8:50:50 AM	Folder	
Gold	10/30/2024, 8:50:08 AM	Folder	
Silver	10/30/2024, 8:50:00 AM	Folder	
synapse	10/30/2024, 1:25:19 AM	Folder	

Scenario 2: Data Modelling in Azure Synapse.

Step 1: Load Data to Synapse (Bronze Layer)

bronze

SQL script 1

Other users in your workspace may have access to modify this item. Do not use this item unless you trust all users who may have access to the workspace.

Run

Undo

Publish

Query plan

Connect to Built-in

Use database NetflixAnalytics

```

34 -- Create an external table pointing to the Netflix dataset in the bronze layer
35 CREATE EXTERNAL TABLE bronze_netflix (
36     show_id NVARCHAR(50),
37     title NVARCHAR(255),
38     director NVARCHAR(255),
39     cast NVARCHAR(MAX),
40     country NVARCHAR(100),
41     date_added NVARCHAR(50),
42     release_year INT,
43     rating NVARCHAR(50),
44     duration NVARCHAR(50),
45     listed_in NVARCHAR(255),
46     description NVARCHAR(MAX),
47     type NVARCHAR(50)
48 )
49 WITH (
50     LOCATION = 'abfss://bronze@rawdata01.dfs.core.windows.net/netflix_cleaned.csv', -- Path in ADLS
51     DATA_SOURCE = NetflixDataSource,
52     FILE_FORMAT = CSVFormat
53 );
54 GO

```

Messages

Step 2: Transform Data for Silver Layer

```

bronze SQL script 1
Other users in your workspace may have access to modify this item. Do not use this item unless you trust all workspace.

Run Undo Publish Query plan Connect to Built-in Use database NetflixAnalytics

57 -- Create a silver table for cleaned and transformed data
58 -- Create a view for the silver layer with transformations
59 CREATE VIEW silver_netflix AS
60 SELECT
61     show_id,
62     title,
63     director,
64     cast,
65     country,
66     TRY_CAST(SUBSTRING(date_added, 1, CHARINDEX(',', date_added) - 1) AS DATE) AS date_added,
67     release_year,
68     rating,
69     CASE
70         WHEN duration LIKE '%Season%' THEN TRY_CAST(SUBSTRING(duration, 1, CHARINDEX(' ', duration) - 1) AS INT)
71         ELSE NULL
72     END AS num_seasons,
73     CASE
74         WHEN duration LIKE '%min%' THEN TRY_CAST(SUBSTRING(duration, 1, CHARINDEX(' ', duration) - 1) AS INT)
75         ELSE NULL
76     END AS duration_minutes,
77     listed_in,
78     description,
79     type
80 FROM bronze_netflix
81 WHERE release_year IS NOT NULL;
82 GO

```

Messages

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.

```

bronze SQL script 1
Other users in your workspace may have access to modify this item. Do not use this item unless you trust all workspace.

Run Undo Publish Query plan Connect to Built-in Use database NetflixAnalytics

81 GO
82
83
84 -- Create a view for the country dimension table
85 CREATE VIEW dim_country AS
86 SELECT DISTINCT
87     ROW_NUMBER() OVER (ORDER BY country) AS country_id,
88     country
89 FROM silver_netflix
90 WHERE country IS NOT NULL;
91 GO
92
93 CREATE VIEW dim_director AS
94 SELECT DISTINCT
95     ROW_NUMBER() OVER (ORDER BY director) AS director_id,
96     director
97 FROM silver_netflix
98 WHERE director IS NOT NULL;
99
100
101 CREATE VIEW dim_category AS
102 SELECT DISTINCT
103     ROW_NUMBER() OVER (ORDER BY listed_in) AS category_id,
104     listed_in AS category
105 FROM silver_netflix;
106
107

```

Step 4: . Create fact_netflix_shows Table

```
bronze SQL script 1
Run Undo Publish Query plan Connect to Built-in Use database NetflixAnalytics
104 listed_in AS category
105 FROM silver_netflix;
106
107
108 CREATE VIEW fact_netflix_shows AS
109 SELECT
110     s.show_id,
111     s.title,
112     d.director_id,
113     c.country_id,
114     cat.category_id,
115     s.date_added,
116     s.release_year,
117     s.rating,
118     s.num_seasons,
119     s.duration_minutes,
120     s.type
121 FROM silver_netflix s
122 LEFT JOIN dim_director d ON s.director = d.director
123 LEFT JOIN dim_country c ON s.country = c.country
124 LEFT JOIN dim_category cat ON s.listed_in = cat.category;
125
126
127 SELECT *
```

Step 4: Optimize the Fact Table

```
1 SELECT *
2 FROM fact_netflix_shows
3 WHERE release_year = 2021;
```

Step 5: Create Analytical Views

```
bronze SQL script 1
Run Undo Publish Query plan Connect to Built-in Use database NetflixAnalytics
131 -- View for popular content in a specific category
132 CREATE VIEW vw_popular_content AS
133 SELECT
134     f.title,
135     d.director,
136     c.country,
137     f.date_added,
138     f.release_year,
139     f.rating,
140     f.num_seasons,
141     f.duration_minutes
142 FROM fact_netflix_shows f
143 JOIN dim_director d ON f.director_id = d.director_id
144 JOIN dim_country c ON f.country_id = c.country_id
145 WHERE f.type = 'TV Show' AND f.rating IN ('TV-MA', 'R');
146
147
148 -- Query to get the number of shows per country and year
149 SELECT
150     c.country,
151     f.release_year,
152     COUNT(*) AS num_shows
153 FROM fact_netflix_shows f
154 JOIN dim_country c ON f.country_id = c.country_id
155 GROUP BY c.country, f.release_year
156 ORDER BY num_shows DESC;
157
Messages
```

Step 6: Run an Example Query on the Fact Table

```
1 SELECT
2     c.country,
3     f.release_year,
4     COUNT(*) AS num_shows
5 FROM fact_netflix_shows f
```

```

6 JOIN dim_country c ON f.country_id = c.country_id
7 GROUP BY c.country, f.release_year
8 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

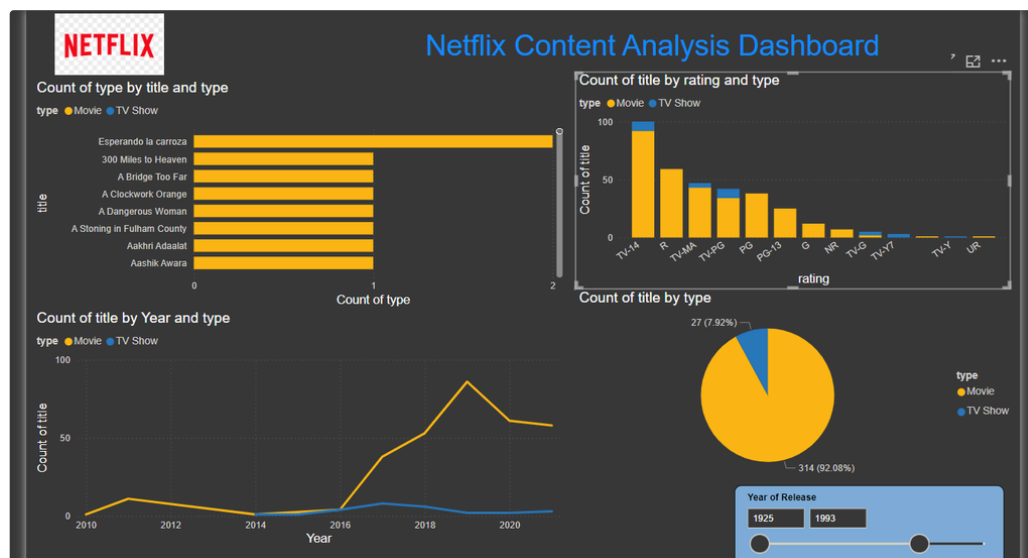
With data in the Gold layer and accessible through Synapse SQL Pool, you can connect Power BI to Synapse and create visualizations.

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.