# **Project Documentation: Netflix Data Analysis Using ELT**

## **Overview**

## This project demonstrates a comprehensive **ELT (Extract, Load, Transform)** workflow using **Python** and **SQL Server**. The focus was on transforming and analyzing Netflix dataset containing information about movies and TV shows. Key challenges included data cleaning, normalization, and answering analytical questions using SQL.

## **Data Cleaning**

### **Handling Foreign Characters**

Korean movie titles appeared as **unreadable characters** in SQL Server due to its inability to handle foreign characters in the Varchar data type. While changing the data type to NVarchar resolved this issue for many, it did not work in my case, likely due to an older SQL Server version. To overcome this:

I explicitly defined the NVARCHAR data type in the Python file before loading the data into SQL Server:

[python file code

dtype = {'title': NVARCHAR(length=1000)}]

Additionally, the dataset’s maximum character length for some columns exceeded optimal performance limits. Using Python’s **str.len()**, I calculated the maximum character length of each column and adjusted data types accordingly for better query performance.

### **Handling Null/Missing Values**

Using Python, I identified null values with the **isna().sum()** function. For instance, the **cast** column contained nulls, which were dropped using the **dropna()** method to ensure accurate results in subsequent analysis.

### **Removing Duplicates**

* Titles with the same name but different letter cases were standardized using the **LOWER()** or **UPPER()** SQL functions to detect duplicates.
* For identifying multi-column duplicates (title and type), I used the **CONCAT()** function to merge these columns into a single value because the **IN** operator cannot directly handle multiple columns. This approach allowed for accurate duplicate identification.
* Duplicate removal was implemented using a **Common Table Expression (CTE)** with the **ROW\_NUMBER()** function to retain only the first occurrence of each duplicate record.

### **Director Data Normalization**

The director data, stored as comma-separated values, was normalized to improve usability:

* Using SQL’s **string\_split()** function, we split the directors’ names into individual values using a comma as the delimiter. The **CROSS APPLY** operator was then used to group these split rows with their respective show\_id, creating a clean and normalized dataset.
* **TRIM()** was applied to clean leading/trailing spaces, and the results were stored in a new netflix\_director table for easier analysis.

### **Filling Missing Country Data**

A SQL query was used to populate missing country information by referencing data from the netflix\_country and netflix\_director tables. This ensured accurate data enrichment based on directors’ prior work.

## **Data Analysis**

### **Key Questions Addressed**

1. **Director Content Contribution**
   1. **Objective**: Calculate the number of movies and TV shows directed by each director.
   2. **Method**: Using conditional aggregation with CASE statements, the query counted distinct show\_ids for movies and TV shows separately, grouped by director.
2. **Diversified Directors**
   1. **Objective**: Identify directors who worked on both movies and TV shows.
   2. **Method**: Conditional aggregation was used to count distinct content types (Movie and TV Show). Directors with contributions to more than one type were filtered using a HAVING clause.

## **Key Learnings**

* **Optimized Data Types**: Adjusting character lengths for optimal SQL performance.
* **Effective Deduplication**: Using case standardization and CTEs for robust duplicate handling.
* **Data Normalization**: Enhanced query usability by splitting and cleaning multi-value fields.
* **SQL Analysis**: Leveraged SQL’s aggregation and conditional logic for meaningful insights.

## **Conclusion**

This project highlights the power of combining Python for data extraction and SQL Server for data transformation and analysis. By addressing real-world challenges like handling foreign characters, duplicates, and null values, I gained valuable insights into data cleaning and analysis.