NAME: KARTIK SHARMA

STUDENT ID: 8962512

REPORT

INDIVIDUAL PROJECT

SQL

TOPIC: NETFLIX MOVIES AND TV SHOWS

**Goal for Analyzing Netflix Data**

**The main goal of analyzing this Netflix dataset is to gain insights into the types of content available on the platform, understand viewing trends, and explore patterns related to movies and TV shows. Specifically, I want to:**

1. **Identify Content Trends: Analyze which genres, ratings, and content types (movies vs. TV shows) are most popular based on release years and availability.**
2. **Explore Director and Cast Influence: Investigate whether certain directors or cast members are associated with higher-rated content or specific genres.**
3. **Understand Global Reach: Examine the distribution of content by country to identify how Netflix caters to different international markets.**
4. **Duration Patterns: Analyze the duration of content (in terms of seasons for TV shows and length for movies) to see trends in binge-worthy or short-form content.**

**By achieving these goals, I aim to develop better data analysis skills, improve my understanding of content distribution on streaming platforms, and use SQL to extract valuable insights from real-world datasets.**

**DATA SOURCE**

<https://www.kaggle.com/datasets/shivamb/netflix-shows>

The dataset being analyzed comes from **Kaggle**, which provides a rich source of publicly available data for various projects. This particular dataset contains detailed information about Netflix titles, including movies and TV shows, and serves as a valuable resource for understanding content trends, viewer preferences, and the platform's global content distribution.

**SCHEMA**

To explain the schema with primary keys, foreign keys, and secondary keys within each table, I'll describe the structure and relationships between your tables in detail. Here's a textual representation of the schema:

**Tables:**

1. **Shows Table**:
   * **Primary Key**: show\_id (Unique identifier for each show)
   * **Attributes**:
     + title
     + type
     + release\_year
     + country
     + date\_added
   * **Foreign Keys**: None in this table.
2. **Directors Table**:
   * **Primary Key**: director\_id (Auto-generated unique identifier for each director)
   * **Attributes**:
     + director\_name (Unique constraint to avoid duplicate names)
   * **Foreign Keys**: None.
3. **Show\_Cast Table**:
   * **Primary Key**: cast\_id (Unique identifier for each cast entry)
   * **Attributes**:
     + show\_id (Foreign Key referencing Shows.show\_id)
     + actor\_name (Actor name involved in the show)
   * **Foreign Keys**: show\_id references Shows(show\_id)
4. **Ratings Table**:
   * **Primary Key**: show\_id (Same as in the Shows table, making it a one-to-one relationship)
   * **Attributes**:
     + rating
     + duration
     + listed\_in (Genres or categories)
     + description
   * **Foreign Keys**: show\_id references Shows(show\_id).

**Explanation of Keys:**

* **Primary Keys**:
  + Uniquely identify each row in the table (e.g., show\_id in Shows, director\_id in Directors, and cast\_id in Show\_Cast).
* **Foreign Keys**:
  + Establish relationships between tables. For example, in the Show\_Cast and Ratings tables, show\_id is a foreign key that links back to the Shows table, creating relationships between shows and their associated ratings and cast members.

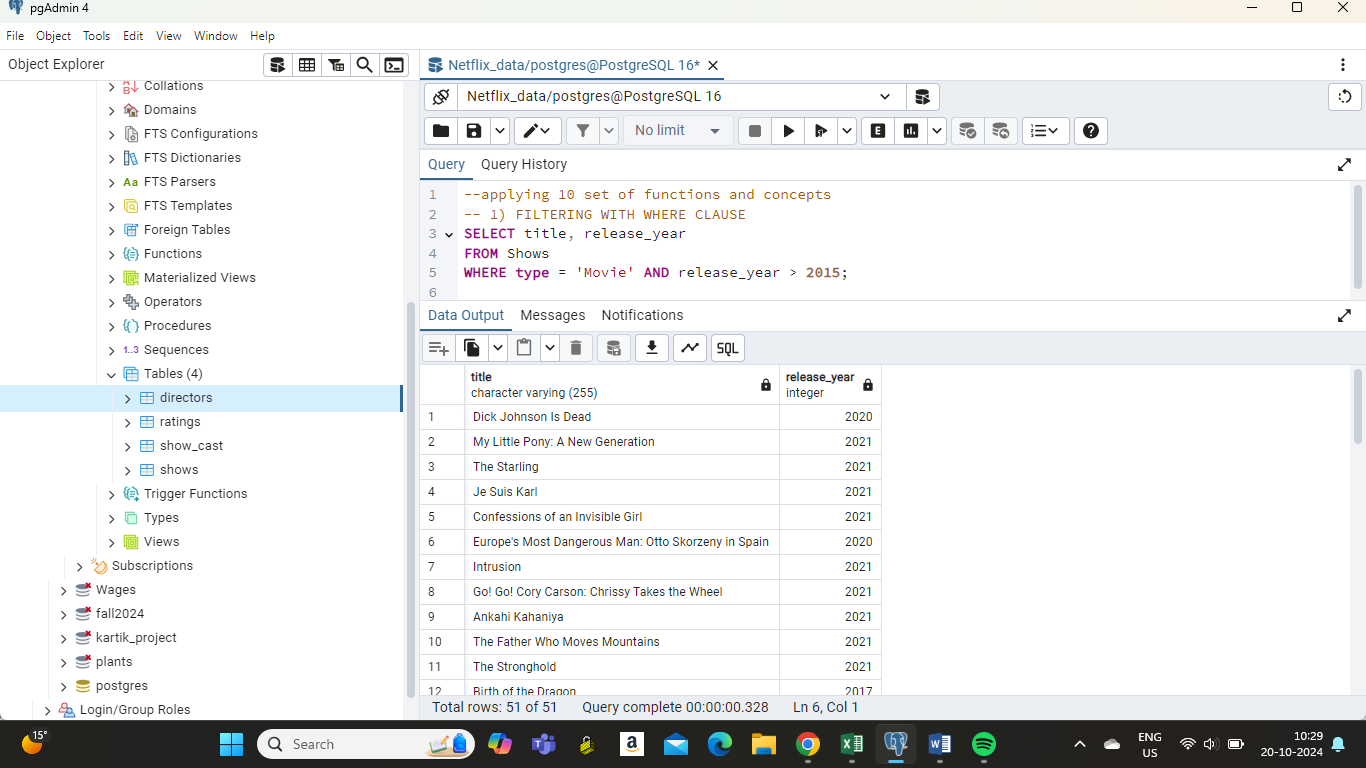
**ERD Diagram Overview:**

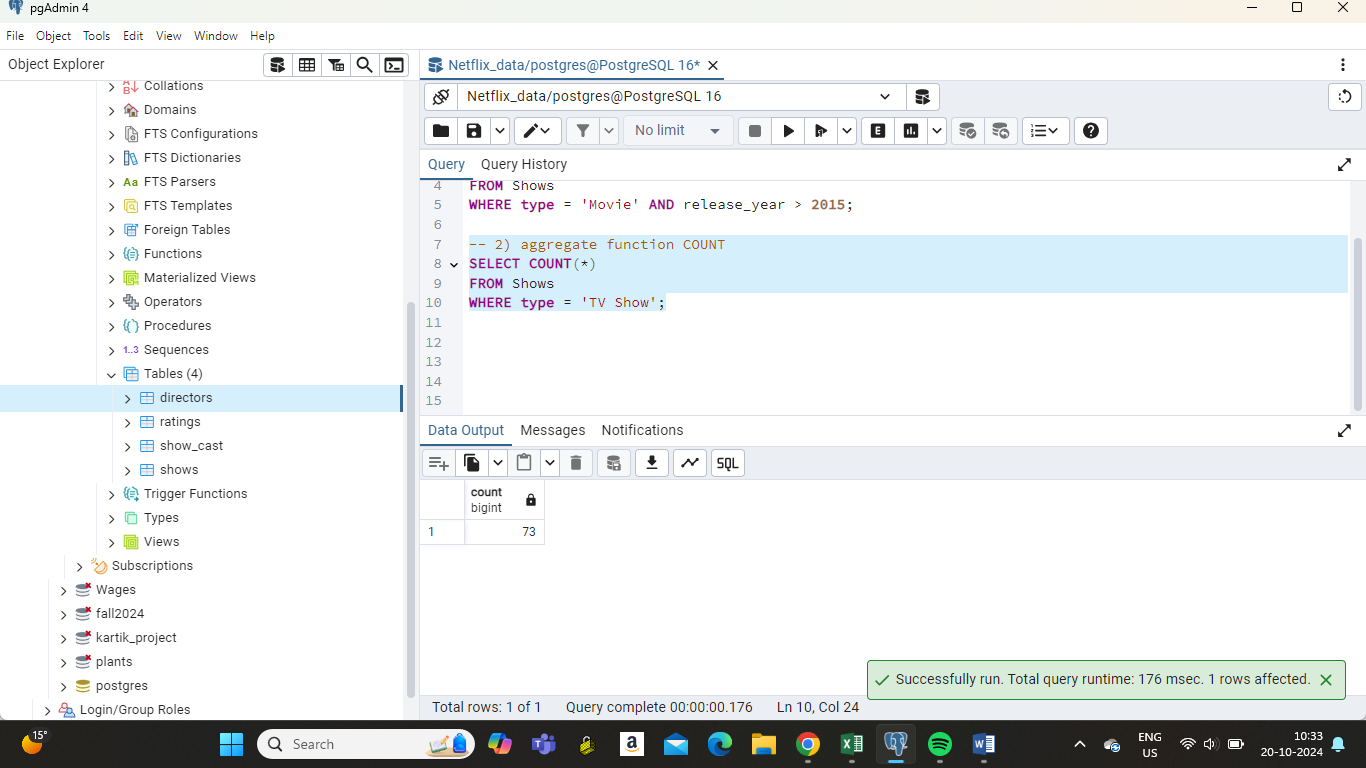
* **Shows** (Primary key: show\_id) is the central table.
* **Directors** is independent but could be linked to the Shows table through the director\_name.
* **Show\_Cast** has a show\_id foreign key, linking it to the Shows table (many cast members can be associated with one show).
* **Ratings** has a show\_id foreign key, which establishes a one-to-one relationship with Shows.

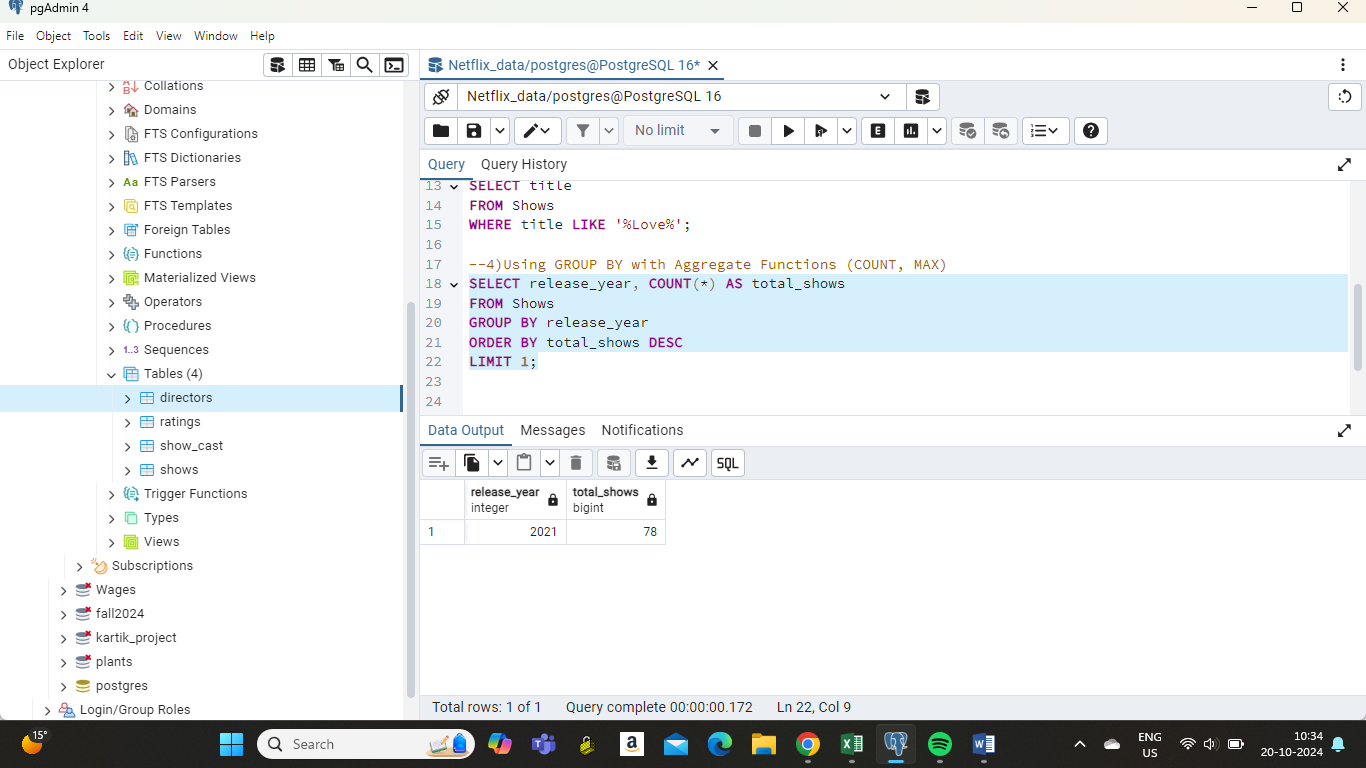
**SQL QUERIES**

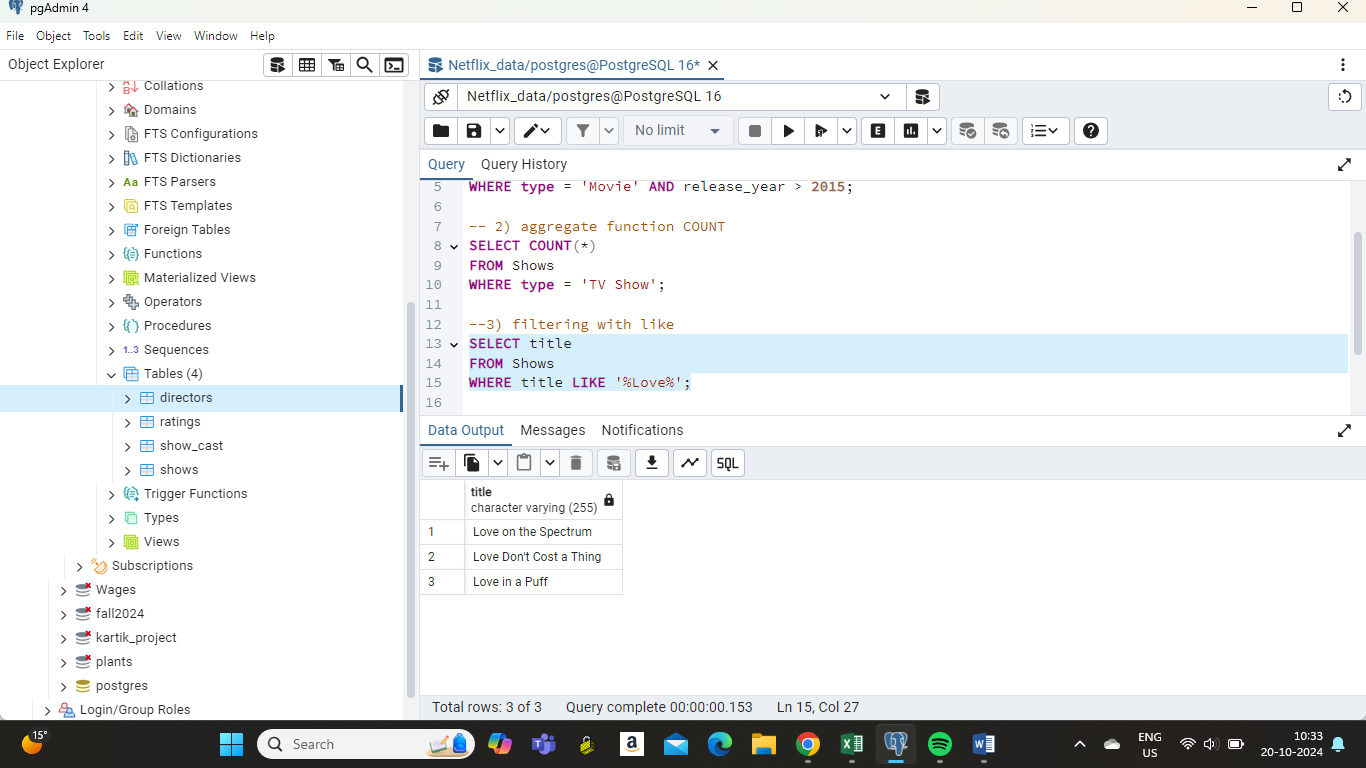
* I have populated the 4 tables in the schema with data – All fact tables have more than 50 lines of data each.
* Applied at least 10 SQL functions through queries (Filters, Joins, Sub-queries, CTEs, etc.) -
* Created 2 views for end users similar to what we saw during our sessions

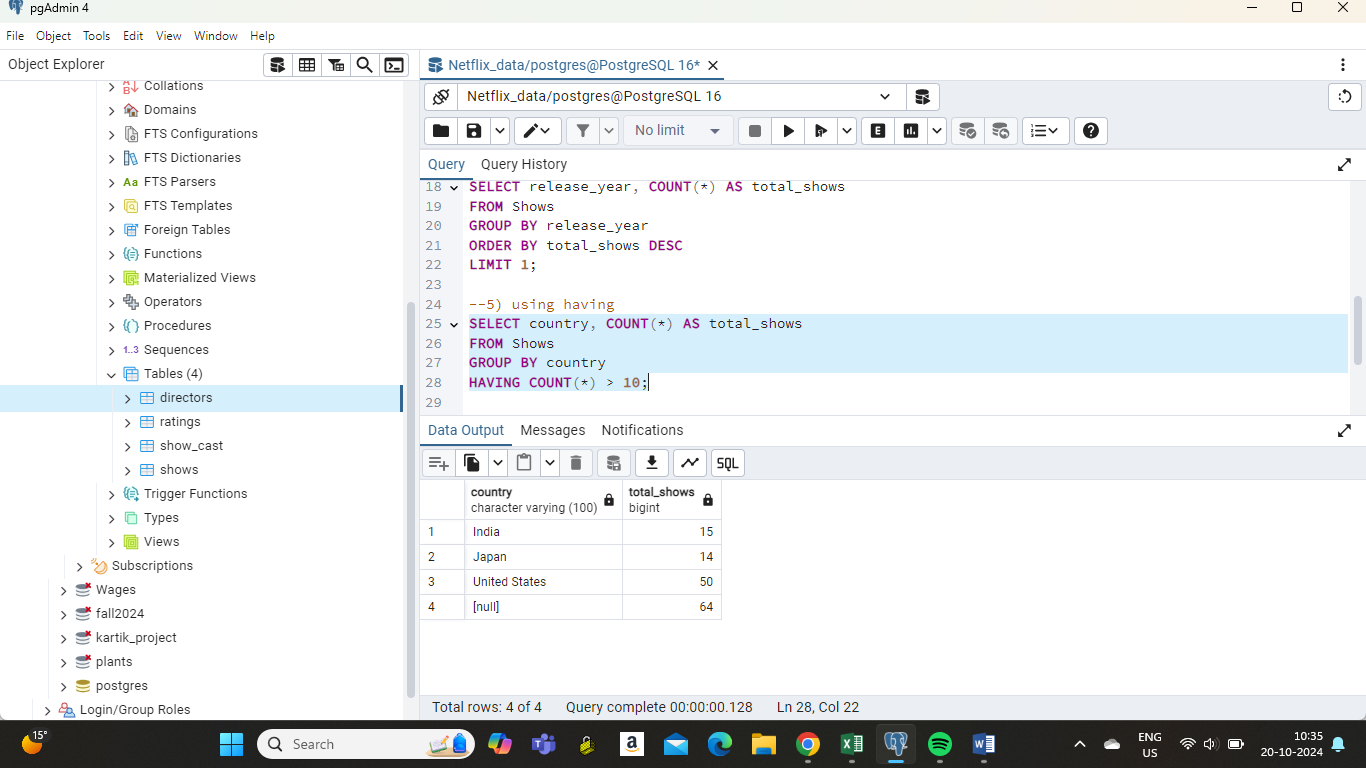
1. **10 DIFFERENT FUNCTIONS AND QUERY RESULT SCREENSHOTS**

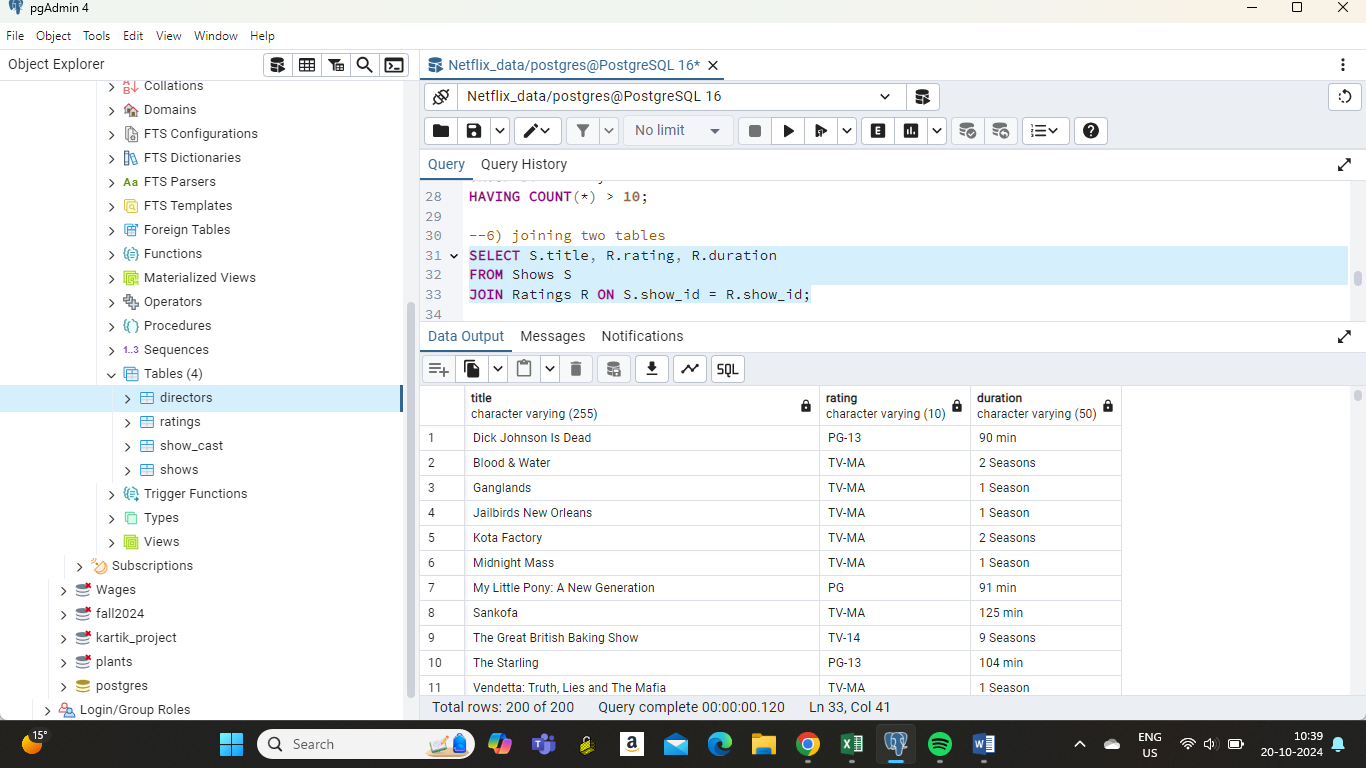


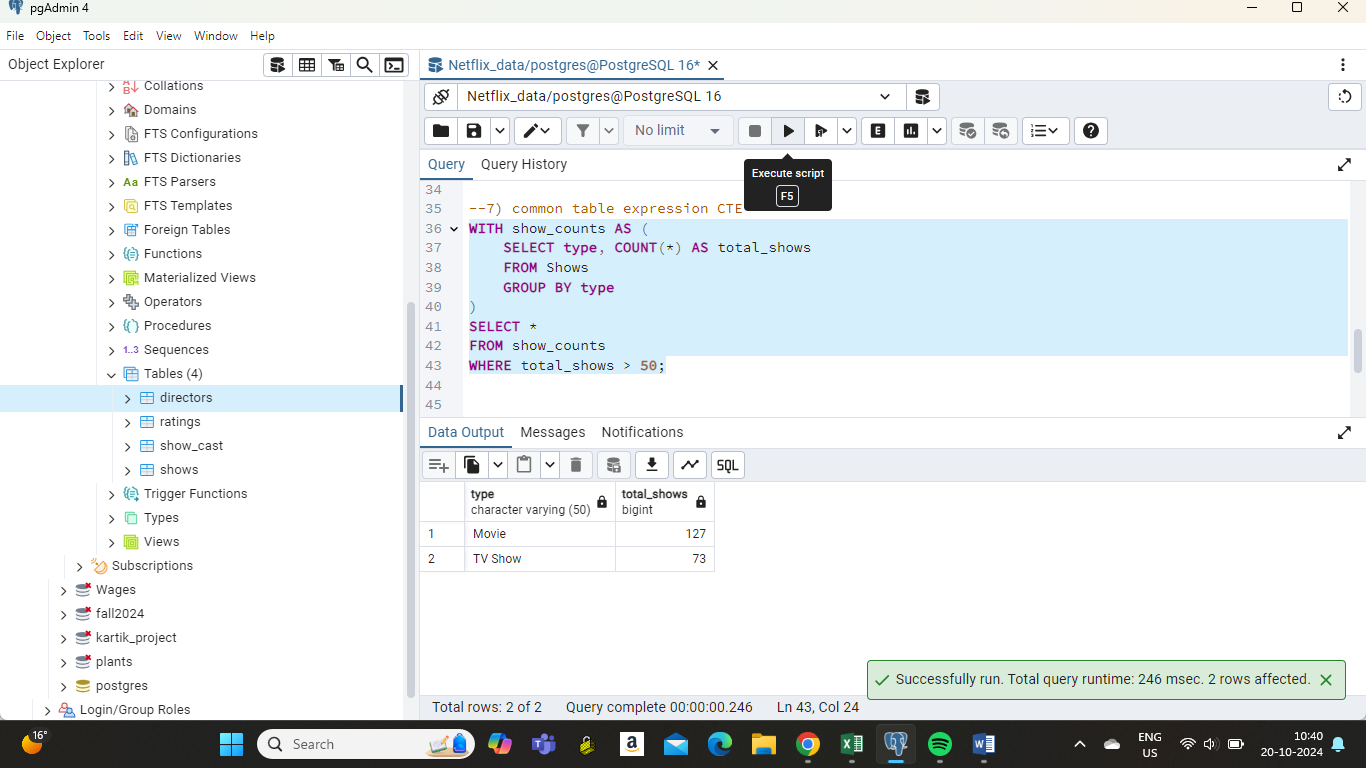


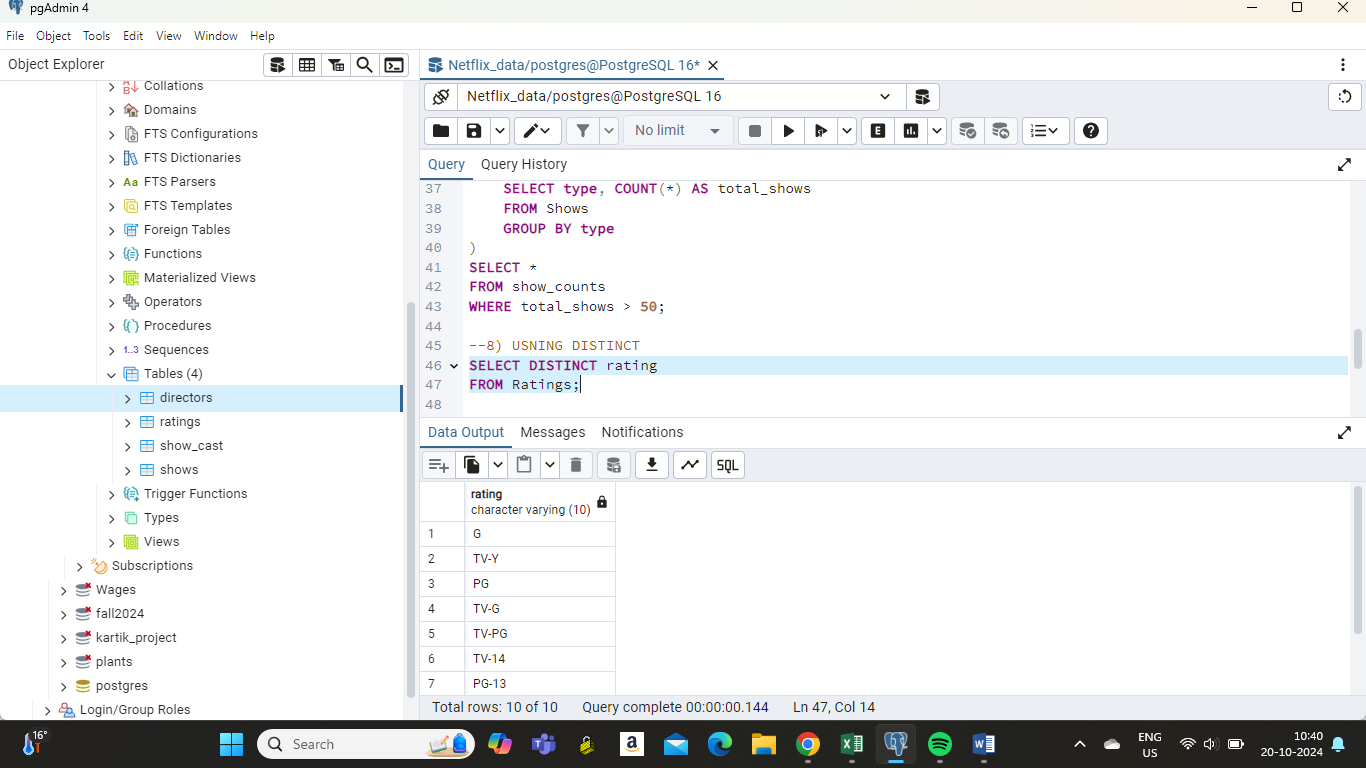


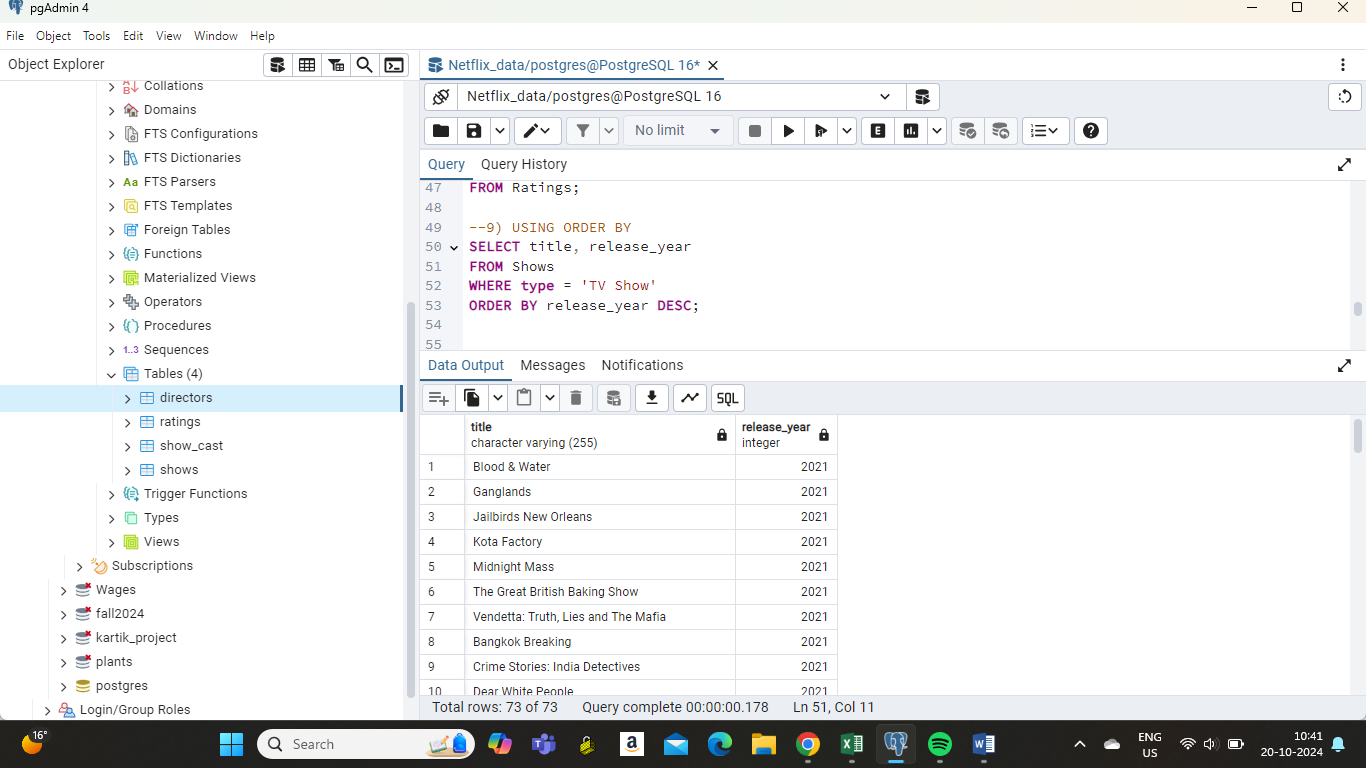


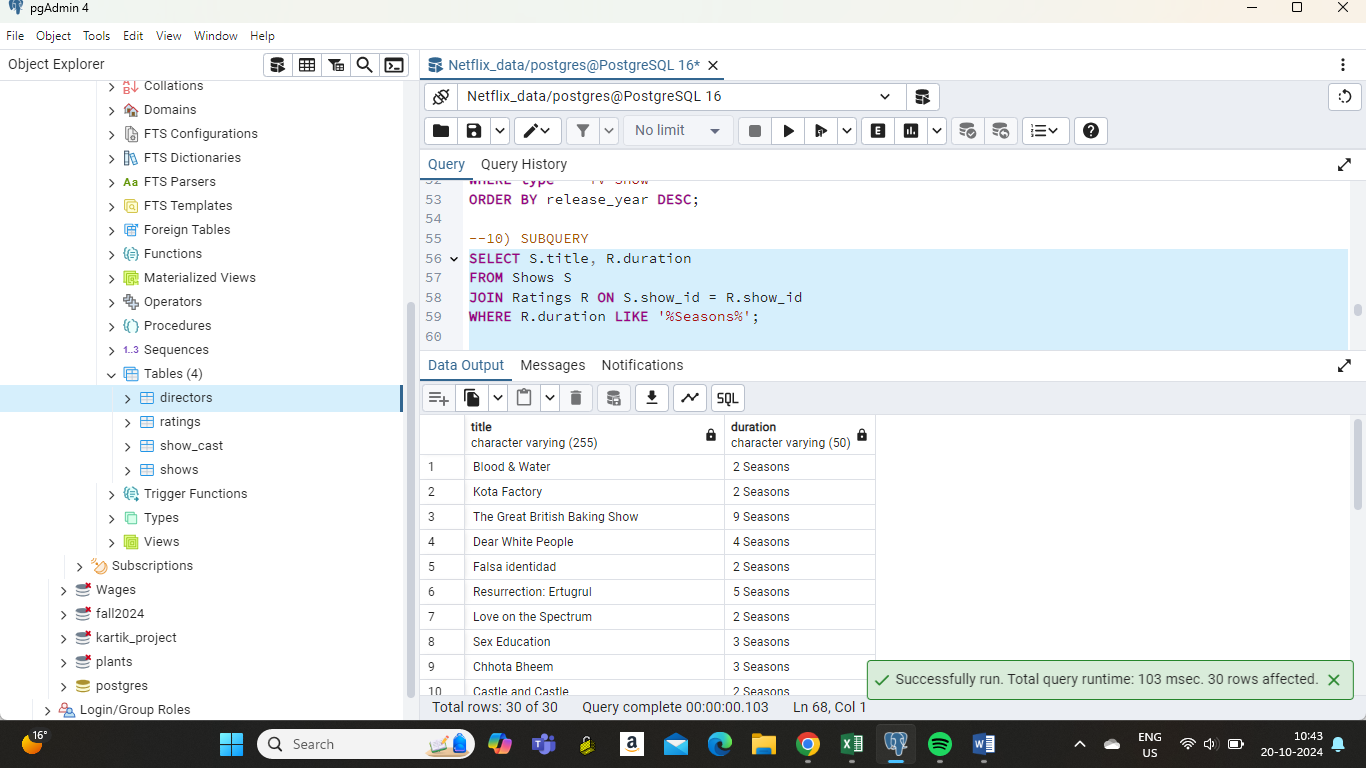












**VIEWS**

