|  |
| --- |
| **ITS INCOM D1 2023-25 - ID. 48607 1° anno 108206**  **TECNICO SUPERIORE PER I METODI E LE TECNOLOGIE PER LO SVILUPPO DI SISTEMI SOFTWARE**  **ITS DATA ANALYST**  **Anno formativo 2024-25** |
| **LABORATORIO 1: LABORATORIO DI GESTIONE DATABASE**  **PROVA DI VALUTAZIONE FINALE** |
| **RELAZIONE TECNICA**  **GRUPPO XXXXX** |

-- ========================================================

-- SQL Documentation

-- ========================================================

-- ========================================================

-- Step 1: Create the Database

-- ========================================================

CREATE DATABASE Star\_Trek;

-- Purpose: Creates a new database called 'Star\_Trek'.

-- ========================================================

-- Step 2: Use the Created Database

-- ========================================================

USE Star\_Trek;

-- Purpose: Switches the context to the 'Star\_Trek' database,

-- so subsequent commands are executed within it.

-- ========================================================

-- Step 3: Create the Episodes Table

-- ========================================================

CREATE TABLE dbo.episodes (

    id\_episodes INT IDENTITY(1,1) PRIMARY KEY,  -- Unique identifier for each episode, auto-incremented

    episode\_number VARCHAR(MAX) NOT NULL,       -- Episode number, flexible data type to allow alphanumeric identifiers

    season INT                                  -- The season number to which the episode belongs

);

-- Purpose: Creates a table called 'episodes' to store information about each episode.

-- Columns:

-- - id\_episodes: Unique identifier for each episode, set to auto-increment.

-- - episode\_number: The episode number (e.g., 1, 2, 3, ...), cannot be null, and stored as VARCHAR(MAX).

-- - season: The season number to which the episode belongs.

-- ========================================================

-- Step 4: Create the Characters Table

-- ========================================================

CREATE TABLE dbo.characters (

    id\_characters INT IDENTITY PRIMARY KEY,    -- Auto-incrementing ID for each character's line

    character VARCHAR(255) NOT NULL,            -- Name of the character

    id\_episodes INT,                           -- ID of the episode to which the character belongs

    lines\_content VARCHAR(MAX),                -- The content of the character's lines in the episode

    FOREIGN KEY (id\_episodes) REFERENCES dbo.episodes(id\_episodes) -- Foreign key referencing the 'episodes' table

);

-- Purpose: Creates a table called 'characters' to store character lines for each episode.

-- Columns:

-- - id\_characters: Unique identifier for each character's line, auto-incremented.

-- - character: The name of the character (e.g., Captain Kirk, Spock).

-- - id\_episodes: Foreign key linking the character's lines to an episode.

-- - lines\_content: The content of the character's lines in the episode.

-- ========================================================

-- Step 5: Create Stored Procedure for Inserting Character Lines

-- ========================================================

CREATE PROCEDURE [dbo].[InsertCharacterLines]

@character NVARCHAR(255),

@id\_episodes INT,

@lines\_content NVARCHAR(MAX)

AS

BEGIN

    -- Insert the character's lines into the 'characters' table

    INSERT INTO dbo.characters (character, id\_episodes, lines\_content)

    VALUES (@character, @id\_episodes, @lines\_content);

END;

-- Purpose: Defines a stored procedure to insert character lines into the 'characters' table.

-- Parameters:

-- - @character: The character's name.

-- - @id\_episodes: The ID of the episode.

-- - @lines\_content: The content of the character's lines.

-- ========================================================

-- Step 6: Create Stored Procedure for Inserting New Episode

-- ========================================================

CREATE PROCEDURE [dbo].[InsertEpisode]

@episode\_number VARCHAR(255),

@season INT = NULL

AS

BEGIN

    -- Insert a new episode and return the ID of the newly inserted episode

    INSERT INTO dbo.episodes (episode\_number, season)

    OUTPUT INSERTED.id\_episodes  -- Returns the ID of the newly inserted episode

    VALUES (@episode\_number, @season);

    -- Return the ID of the newly inserted episode

    SELECT SCOPE\_IDENTITY() AS episode\_id;

END;

-- Purpose: Defines a stored procedure to insert a new episode into the 'episodes' table.

-- Parameters:

-- - @episode\_number: The episode number (e.g., 'S01E01').

-- - @season: The season number to which the episode belongs.

-- ========================================================

-- Step 7: Create a View for Counting Total Number of Lines per Episode by Character and Season

-- ========================================================

CREATE VIEW NBattute AS

SELECT COUNT(\*) AS NumeroBattute,        -- Count of lines spoken by each character in each episode

       E.episode\_number,                  -- Episode number

       E.season,                           -- Season number

       C.character                        -- Character speaking the lines

FROM dbo.episodes E

INNER JOIN dbo.characters C

    ON E.id\_episodes = C.id\_episodes

GROUP BY E.episode\_number, E.season, C.character;

-- Purpose: Creates a view to calculate the total number of lines spoken by each character in each episode.

-- It groups by episode\_number, season, and character.

-- ========================================================

-- Step 8: Create a View for Counting Total Number of Words per Episode by Character and Season

-- ========================================================

CREATE VIEW NParole AS

SELECT (LEN(lines\_content) - LEN(REPLACE(lines\_content, ' ', '')) + 1) AS NumeroParole, -- Word count for each line

       E.episode\_number,                  -- Episode number

       E.season,                           -- Season number

       C.character                        -- Character speaking the lines

FROM dbo.episodes E

INNER JOIN dbo.characters C

    ON E.id\_episodes = C.id\_episodes

GROUP BY E.episode\_number, E.season, C.character, lines\_content;

-- Purpose: Creates a view to calculate the total number of words spoken by each character in each episode.

-- Uses the LEN function to calculate the word count by counting spaces.

-- ========================================================

-- Step 9: Create a View for Counting Total Number of Characters (Excluding Spaces) per Episode by Character and Season

-- ========================================================

CREATE VIEW LenTotBattute AS

SELECT SUM(LEN(lines\_content)) AS TotaleCaratteri,  -- Total characters spoken, excluding spaces

       E.episode\_number,                             -- Episode number

       E.season,                                    -- Season number

       C.character                                  -- Character speaking the lines

FROM dbo.episodes E

INNER JOIN dbo.characters C

    ON E.id\_episodes = C.id\_episodes

GROUP BY E.episode\_number, E.season, C.character;

-- Purpose: Creates a view to calculate the total number of characters (excluding spaces) spoken by each character in each episode.

-- ========================================================

-- Step 10: Create a View for Counting Total Number of Characters (Excluding Spaces) per Season by Character

-- ========================================================

CREATE VIEW TopBattute AS

SELECT SUM(LEN(C.lines\_content)) AS TotaleCaratteri,  -- Total characters spoken by the character in a season

       E.season,                                     -- Season number

       C.character                                   -- Character speaking the lines

FROM dbo.episodes E

INNER JOIN dbo.characters C

    ON E.id\_episodes = C.id\_episodes

GROUP BY E.season, C.character;

-- Purpose: The 'TopBattute' view calculates the total number of characters spoken by each character in each season.

-- It aggregates the total number of characters spoken across all episodes in a given season.

-- ========================================================

-- Step 11: Create a View for Counting Distinct Characters Across All Episodes

-- ========================================================

CREATE VIEW TotalePersonaggi AS

SELECT

    (SELECT COUNT(DISTINCT character) FROM dbo.characters) AS TotalePersonaggi,  -- Total distinct characters across all episodes

    E.season,                            -- Season number

    COUNT(DISTINCT C.character) AS PersonaggiPerStagione  -- Distinct characters per season

FROM dbo.characters C

INNER JOIN dbo.episodes E

    ON C.id\_episodes = E.id\_episodes

GROUP BY E.season;

-- Purpose: Creates a view to calculate the total number of distinct characters across all episodes and the number of distinct characters per season.

-- ========================================================

-- Step 12: Create a View for Calculating Character Participation Percentage in the Entire Series

-- ========================================================

CREATE VIEW PercentualePersonaggio AS

SELECT

    c.character,

    ROUND(COUNT(DISTINCT c.id\_episodes) \* 1.0 /

          (SELECT COUNT(DISTINCT e.id\_episodes) FROM dbo.episodes e) \* 100, 2) AS PercentualePartecipazione

FROM dbo.characters c

JOIN dbo.episodes e

    ON c.id\_episodes = e.id\_episodes

GROUP BY c.character;

-- Purpose: Creates a view to calculate the percentage of lines spoken by each character in the entire series.

-- It divides the number of lines spoken by each character by the total number of episodes in the series, multiplied by 100.

-- ========================================================

-- Step 13: Create a Table for Storing Star Trek Seasons Information

-- ========================================================

CREATE TABLE dbo.Stagioni (

    id\_season INT NOT NULL IDENTITY(1,1) PRIMARY KEY,    -- Unique identifier for each season

    season\_code CHAR(3) NOT NULL,                         -- Season code (e.g., TOS, TNG)

    season\_name VARCHAR(100) NOT NULL                     -- Full name of the season (e.g., 'Star Trek - The Original Series')

);

-- Purpose: Creates a table to store information about different Star Trek seasons.

-- Columns:

-- - id\_season: Unique identifier for each season, set to auto-increment.

-- - season\_code: A fixed-length character field that stores the code for each season.

-- - season\_name: A variable-length string field that stores the full name of the season.

-- ========================================================

-- Step 14: Insert Initial Data into the Stagioni Table for Various Star Trek Series

-- ========================================================

INSERT INTO dbo.Stagioni(season\_code, season\_name)

VALUES

    ('TOS', 'Star Trek - The Original Series'),     -- Original Star Trek series

    ('TAS', 'Star Trek - The Animated Series'),     -- Animated Star Trek series

    ('TNG', 'Star Trek - The Next Generation'),     -- Star Trek: The Next Generation

    ('DS9', 'Star Trek - Deep Space Nine'),         -- Star Trek: Deep Space Nine

    ('VOY', 'Star Trek - Voyager'),                 -- Star Trek: Voyager

    ('ENT', 'Star Trek - Enterprise');              -- Star Trek: Enterprise

-- Purpose: Inserts initial data for the various seasons of the Star Trek series into the 'Stagioni' table.