**1.توضیحات جداول (Table Descriptions)**

**جدول: Movie**

* + **توضیح:** این جدول اطلاعات اصلی و جامع مربوط به هر فیلم را ذخیره می‌کند. MovieID به عنوان کلید اصلی، شناسه منحصر به فرد هر فیلم است.
  + **ستون‌ها:**

MovieID (PK) : شناسه یکتای هر فیلم.

Title : عنوان فیلم.

ReleaseYear : سال انتشار فیلم.

Summary : خلاصه داستان فیلم.

DurationInMinutes : مدت زمان فیلم به دقیقه.

Country : کشور اصلی سازنده فیلم.

TMDbScore : میانگین نمره فیلم از سایت TMDb .

DirectorID (FK) : شناسه کارگردان فیلم که به جدول Person اشاره دارد.

* **جدول: Person**
  + **توضیح:** این جدول برای ذخیره اطلاعات اشخاص (بازیگران و کارگردانان) به صورت منحصر به فرد طراحی شده است.
  + **ستون‌ها:**

PersonID (PK) : شناسه یکتای هر شخص.

FullName : نام کامل شخص.

BirthDate : تاریخ تولد.

Gender : جنسیت.

Nationality : ملیت (بر اساس محل تولد).

* **جدول: Movie\_Actor**
  + **توضیح:** این یک جدول واسط (Junction Table) برای پیاده‌سازی رابطه چندبه‌چند بین فیلم‌ها و بازیگران است. هر ردیف در این جدول، نشان‌دهنده حضور یک بازیگر در یک فیلم است.

**۲. کوئری‌های کاربردی (Practical Queries)**

* **کوئری ۱ :** این کوئری فیلم‌های منتشر شده بعد از سال ۲۰۱۰ را بر اساس نمره به صورت نزولی لیست می‌کند.

SELECT

Title,

ReleaseYear,

TMDbScore

FROM

Movie

WHERE

ReleaseYear > 2010

ORDER BY

TMDbScore DESC;

* **کوئری ۲ :**  این کوئری تمام فیلم‌های متعلق به ژانر 'Action' را پیدا می‌کند. برای این کار، از دستور JOIN برای اتصال سه جدول Movie، Movie\_Genre و Genre به یکدیگر استفاده شده است.

SELECT

m.Title,

m.ReleaseYear,

g.GenreName

FROM

Movie AS m

JOIN

Movie\_Genre AS mg ON m.MovieID = mg.MovieID

JOIN

Genre AS g ON mg.GenreID = g.GenreID

WHERE

g.GenreName = 'Action';

**کوئری ۳:** این کوئری بازیگرانی که در بیشترین فیلم حضور داشته‌اند را لیست می‌کند. در این کوئری از **GROUP BY** برای گروه‌بندی بر اساس هر بازیگر و از تابع تجمعی **COUNT** برای شمارش فیلم‌های هر گروه استفاده شده است.

SELECT

p.FullName,

COUNT(ma.MovieID) AS MovieCount

FROM

Person AS p

JOIN

Movie\_Actor AS ma ON p.PersonID = ma.PersonID

GROUP BY

p.PersonID, p.FullName

ORDER BY

MovieCount DESC

LIMIT 10;

**کوئری ۴:** این کوئری میانگین نمره فیلم‌ها را برای هر ژانر محاسبه می‌کند. برای این کار از **GROUP BY** برای گروه‌بندی بر اساس ژانر، از تابع تجمعی **AVG**  برای محاسبه میانگین، و از دستور **HAVING**  برای فیلتر کردن ژانرهایی که کمتر از حد نصاب فیلم دارند، استفاده شده است.

SELECT

g.GenreName,

COUNT(m.MovieID) AS NumberOfMovies,

AVG(m.TMDbScore) AS AverageScore

FROM

Genre AS g

JOIN

Movie\_Genre AS mg ON g.GenreID = mg.GenreID

JOIN

Movie AS m ON mg.MovieID = m.MovieID

WHERE

m.TMDbScore IS NOT NULL

GROUP BY

g.GenreName

HAVING

NumberOfMovies > 5

ORDER BY

AverageScore DESC;

**کوئری ۶:** این کوئری سال شروع به کار (اولین فیلم) را برای بازیگرانی که بیش از یک فیلم در دیتابیس دارند، پیدا می‌کند. در این کوئری از اتصال سه جدول، گروه‌بندی بر اساس بازیگر و تابع تجمعی MIN برای یافتن قدیمی‌ترین سال انتشار استفاده شده است.

SELECT

p.FullName,

COUNT(m.MovieID) AS MovieCount,

MIN(m.ReleaseYear) AS DebutYear

FROM

Person AS p

JOIN

Movie\_Actor AS ma ON p.PersonID = ma.PersonID

JOIN

Movie AS m ON ma.MovieID = m.MovieID

WHERE

m.ReleaseYear IS NOT NULL

GROUP BY

p.PersonID, p.FullName

HAVING

MovieCount > 1

ORDER BY

DebutYear ASC;

**کوئری ۷:** این کوئری فیلم‌های باکیفیت و جدید را پیدا می‌کند که نمره‌شان از میانگین کل نمرات بالاتر است. تکنیک اصلی در اینجا، استفاده از یک Subquery در شرط WHERE برای محاسبه دینامیک میانگین کل است.

SELECT

Title,

TMDbScore,

ReleaseYear

FROM

Movie

WHERE

TMDbScore > (SELECT AVG(TMDbScore) FROM Movie WHERE TMDbScore IS NOT NULL)

AND ReleaseYear > 2000

ORDER BY

TMDbScore DESC;

**کوئری ۸:** این کوئری فیلم‌های مشترک بین دو بازیگر خاص را پیدا می‌کند. این کار با یک تکنیک هوشمندانه انجام می‌شود: ابتدا با WHERE ... IN تمام فیلم‌های هر دو بازیگر انتخاب شده، سپس با GROUP BY بر اساس فیلم گروه‌بندی شده و در نهایت با HAVING COUNT(...) = 2 فقط فیلم‌هایی که هر دو بازیگر را شامل می‌شوند، انتخاب می‌گردند.

SELECT

m.Title,

m.ReleaseYear

FROM

Movie AS m

JOIN

Movie\_Actor AS ma ON m.MovieID = ma.MovieID

JOIN

Person AS p ON ma.PersonID = p.PersonID

WHERE

p.FullName IN ('Leonardo DiCaprio', 'Tom Hardy')

GROUP BY

m.MovieID, m.Title, m.ReleaseYear

HAVING

COUNT(p.PersonID) = 2;

کوئری ۹: این کوئری با ترکیب چند شرط ساده در بخش WHERE با استفاده از عملگر AND، داده‌ها را به صورت دقیق فیلتر می‌کند تا فیلم‌های آمریکایی جدید با مدت زمان کوتاه را بیابد.

SELECT

Title,

ReleaseYear,

DurationInMinutes,

Country

FROM

Movie

WHERE

Country = 'United States of America'

AND ReleaseYear > 2010

AND DurationInMinutes < 110

AND DurationInMinutes IS NOT NULL

ORDER BY

TMDbScore DESC;

کوئری ۱۰: این کوئری فیلم‌های یک کارگردان خاص که در یک کشور مشخص ساخته شده‌اند را پیدا می‌کند. نکته کلیدی در اینجا، اتصال (JOIN) جدول Movie به Person از طریق ستون DirectorID است.

SELECT

m.Title,

p.FullName AS Director,

m.TMDbScore,

m.Country AS MovieCountry

FROM

Movie AS m

JOIN

Person AS p ON m.DirectorID = p.PersonID

WHERE

p.FullName = 'Christopher Nolan'

AND m.Country LIKE '%United States of America%'

ORDER BY

m.TMDbScore DESC;

**۳. نماها (Views)**

* **View : v\_MovieWithDirector**
  + **توضیح :**  این View برای ساده‌سازی دسترسی به اطلاعات فیلم به همراه نام کارگردان آن ساخته شده است. با استفاده از این View، دیگر نیازی به نوشتن مکرر JOIN بین جداول فیلم و شخص نیست..
  + این View یک کوئری SELECT را که شامل اتصال (JOIN) بین جداول Movie و Person بر اساس DirectorID است، در خود ذخیره می‌کند و نتیجه را به صورت یک جدول مجازی نمایش می‌دهد.

CREATE OR REPLACE VIEW v\_MovieWithDirector AS

SELECT

m.MovieID,

m.Title,

m.ReleaseYear,

m.TMDbScore,

m.Country,

p.FullName AS DirectorName

FROM

Movie AS m

JOIN

Person AS p ON m.DirectorID = p.PersonID;

**مثال استفاده :**

**SELECT \* FROM v\_MovieWithDirector WHERE DirectorName = 'Christopher Nolan';**

**View : v\_GenreStats**

این View یک گزارش آماری آماده از ژانرها ارائه می‌دهد که در آن، تعداد کل فیلم‌ها و میانگین نمره آن‌ها برای هر ژانر محاسبه شده است.

این View یک کوئری پیچیده را که از اتصال (JOIN) سه جدول Genre, Movie\_Genre, و Movie و همچنین گروه‌بندی (GROUP BY) بر اساس نام ژانر استفاده می‌کند، کپسوله می‌کند. توابع تجمعی COUNT و AVG برای انجام محاسبات آماری به کار رفته‌اند.

CREATE OR REPLACE VIEW v\_GenreStats AS

SELECT

g.GenreName,

COUNT(m.MovieID) AS NumberOfMovies,

ROUND(AVG(m.TMDbScore), 2) AS AverageScore

FROM

Genre AS g

JOIN

Movie\_Genre AS mg ON g.GenreID = mg.GenreID

JOIN

Movie AS m ON mg.MovieID = m.MovieID

WHERE

m.TMDbScore IS NOT NULL

GROUP BY

g.GenreName;

**مثال استفاده :**

**SELECT \* FROM v\_GenreStats WHERE NumberOfMovies > 10 ORDER BY AverageScore DESC;**

**4. رویه ذخیره شده (Stored Procedure)**

**Stored Procedure : sp\_GetMoviesByGenre**

این رویه ذخیره شده مانند یک تابع قابل استفاده مجدد در پایگاه داده عمل می‌کند. این رویه، نام یک ژانر را به عنوان ورودی دریافت کرده و لیستی از تمام فیلم‌های متعلق به آن ژانر خاص را که بر اساس نمره مرتب شده‌اند، برمی‌گرداند.

این رویه یک کوئری SELECT را که شامل اتصال (JOIN) بین سه جدول است، کپسوله می‌کند. با استفاده از یک پارامتر ورودی (IN genre\_name\_param)، نتایج به صورت پویا بر اساس ورودی کاربر فیلتر می‌شوند. این کار باعث استفاده مجدد از کد شده و منطق را در سمت پایگاه داده متمرکز می‌کند.

DELIMITER $$

CREATE PROCEDURE sp\_GetMoviesByGenre(IN genre\_name\_param VARCHAR(100))

BEGIN

SELECT

m.Title,

m.ReleaseYear,

m.TMDbScore

FROM

Movie AS m

JOIN

Movie\_Genre AS mg ON m.MovieID = mg.MovieID

JOIN

Genre AS g ON mg.GenreID = g.GenreID

WHERE

g.GenreName = genre\_name\_param

ORDER BY

m.TMDbScore DESC;

END$$

DELIMITER ;

**مثال استفاده :**

CALL sp\_GetMoviesByGenre('Crime');

**۵. ماشه (Trigger)**

قبل از تعریف Trigger، ابتدا جدولی که برای ثبت گزارش‌ها استفاده می‌شود را مستند می‌کنیم.

* **جدول پشتیبان: Movie\_Log** 
  + **توضیح:** این جدول به عنوان یک دفتر گزارش برای ردیابی تغییرات در جدول Movie عمل می‌کند. هر ردیف در این جدول، نشان‌دهنده یک عملیات به‌روزرسانی است.

**Trigger : trg\_AfterMovieUpdate**

این Trigger مانند یک سیستم حسابرسی خودکار عمل می‌کند. وظیفه آن این است که هر زمان اطلاعات یک فیلم در جدول Movie ویرایش (UPDATE) می‌شود، به صورت اتوماتیک یک گزارش از این تغییر را در جدول Movie\_Log ثبت کند.

این Trigger به جدول Movie متصل شده و طوری تنظیم شده است که **AFTER** (بعد از) هر رویداد **UPDATE**  فعال شود. به ازای هر ردیف آپدیت شده، یک دستور INSERT در جدول Movie\_Log اجرا می‌کند. این Trigger از شبه‌رکورد **NEW**  برای دسترسی به مقادیر جدید ردیف )مانند MovieID و (TMDbScore پس از به‌روزرسانی، جهت ایجاد یک پیام گزارش معنادار، استفاده می‌کند.

-- First, the log table is created

CREATE TABLE IF NOT EXISTS Movie\_Log (

LogID INT PRIMARY KEY AUTO\_INCREMENT,

MovieID INT,

ActionType VARCHAR(255),

LogTimestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Then, the trigger is created

DROP TRIGGER IF EXISTS trg\_AfterMovieUpdate;

DELIMITER $$

CREATE TRIGGER trg\_AfterMovieUpdate

AFTER UPDATE ON Movie

FOR EACH ROW

BEGIN

INSERT INTO Movie\_Log (MovieID, ActionType)

VALUES (NEW.MovieID, CONCAT('Movie details updated. New score: ', NEW.TMDbScore));

END$$

DELIMITER ;

**تست :**

-- 1. Perform an update on a movie to fire the trigger

UPDATE Movie SET TMDbScore = 9.9 WHERE MovieID = 278;

-- 2. Check the log table to see the result

SELECT \* FROM Movie\_Log;