IMDB Database:

# VIEWS

1) View to display the the details of celebs born today:

**Code:**

=================================================================================

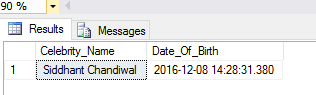
--View to get the details of the Celebs born today

create view celebs\_born\_today as select \* from dbo.get\_Celebs\_Born\_Today(GETDATE());

--Calling of view

select \* from celebs\_born\_today;

Output:



=================================================================================

2) =========================================================================================

Views for Featured Polls

=========================================================================================

**Code:**

select \* from Poll;

go

--View to display the Movie Featured Poll

Create view Movie\_Featured\_Poll as

select PollId,PollName,PollDescription,PostedBy,URL,PollType,IIF(MovieId = NULL,'',MovieId) as MovieId

from Poll where Poll.FeaturedPoll = 1 and PollType='movie';

go

--View to display the Directors' Featured Poll

Create view Director\_Featured\_Poll as

select PollId,PollName,PollDescription,PostedBy,URL,PollType,IIF(DirectorId = NULL,'',DirectorId) as DirectorId

from Poll where Poll.FeaturedPoll = 1 and PollType='director';

go

--View to display the Celebrity Featured Poll

Create view Celebrity\_Featured\_Poll as

select PollId,PollName,PollDescription,PostedBy,URL,PollType,IIF(CelebrityId = NULL,'',CelebrityId) as CelebrityId

from Poll where Poll.FeaturedPoll = 1 and PollType='celebrity';

go

--View to display the TV shows Featured Poll

Create view TVShow\_Featured\_Poll as

select PollId,PollName,PollDescription,PostedBy,URL,PollType,IIF(TVShowId = NULL,'',TVShowId) as TVShowId

from Poll where Poll.FeaturedPoll = 1 and PollType='tv show';

go

-- Diplaying values from view

select \* from TVShow\_Featured\_Poll;

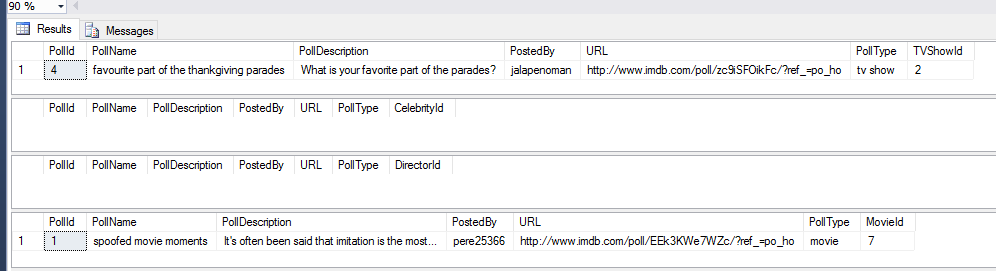
select \* from Celebrity\_Featured\_Poll;

select \* from Director\_Featured\_Poll;

select \* from Movie\_Featured\_Poll;

go

Outut:



3) **View for On\_Tonight**

create view On\_Tonight as

select tv.TVShowName,s.SeasonNumber,e.EpisodeNumber,e.EpsiodeName,c.ChannelName,t.StartTime,t.EndTime from TimeSlots t

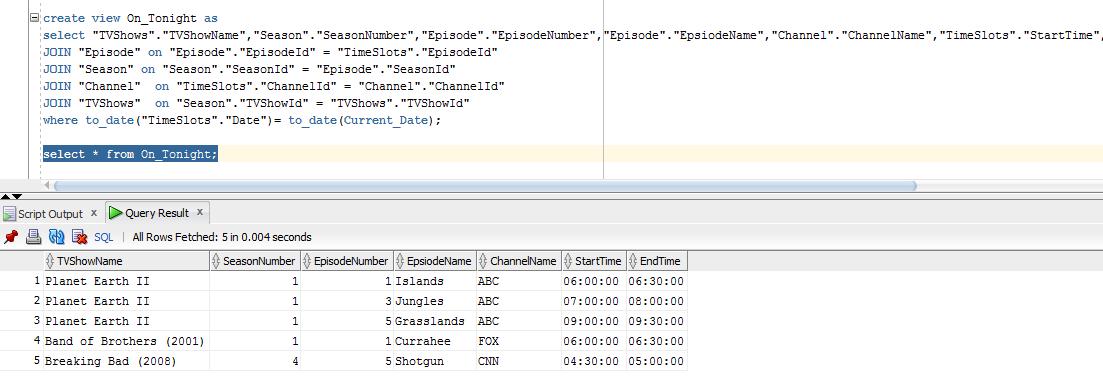
JOIN Episode e on e.EpisodeId = t.EpisodeId

JOIN Season s on s.SeasonId=e.SeasonId

JOIN Channel c on t.ChannelId=c.ChannelId

JOIN TVShows tv on s.TVShowId=tv.TVShowId

where Date=convert(date,GETDATE());



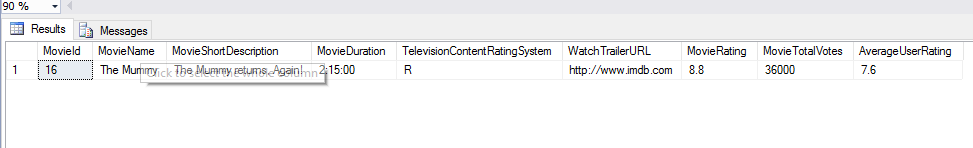
**4) View for Upcoming\_Movies**

create view Upcoming\_Movies as

select MovieId,MovieName,MovieShortDescription,MovieDuration,TelevisionContentRatingSystem,

WatchTrailerURL,MovieRating, MovieTotalVotes,AverageUserRating from Movie where ReleaseDate > GETDATE();

Output:



# PROCEDURES

1) **Procedure to display the TV Show details for a particular channel**

----------------------------------------------------------------------

create or replace PROCEDURE TVCHANNELTIMESLOTS

(

CHANNELID IN NUMBER

)

AS

TVSHOWID NUMBER ;

TVSHOWNAME varchar(30);

SEASONID NUMBER;

EPISODEID NUMBER;

STARTTIME varCHAR(50);

CURSOR cursor1 IS

Select "TVShows"."TVShowId","TVShows"."TVShowName","Season"."SeasonId",

"Episode"."EpisodeId" ,

"TimeSlots"."StartTime"

FROM "TVShows" JOIN

"Season" ON "TVShows"."TVShowId" = "Season"."TVShowId" JOIN

"Episode" ON "Season"."SeasonId" = "Episode"."SeasonId" JOIN

"TimeSlots" ON "Episode"."EpisodeId" = "TimeSlots"."EpisodeId" where "TimeSlots"."ChannelId" =CHANNELID;

BEGIN

DBMS\_OUTPUT.put\_line('TVSHOWID | TVSHOWNAME | SEASONID | EPISODEID | STARTTIME ');

FOR TV IN cursor1

LOOP

TVSHOWID:= TV."TVShowId";

TVSHOWNAME:= TV."TVShowName";

SEASONID:= TV."SeasonId";

EPISODEID:= TV."EpisodeId";

STARTTIME:= TV."StartTime";

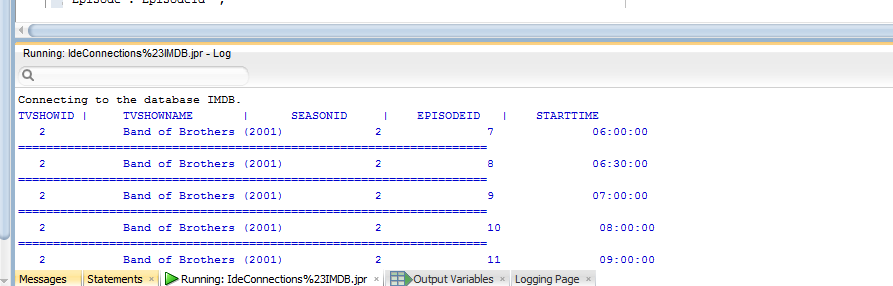
DBMS\_OUTPUT.put\_line(' '||TVSHOWID||' '||TVSHOWNAME||' '||SEASONID||' '||EPISODEID||' '||STARTTIME);

DBMS\_OUTPUT.PUT\_LINE

('===================================================================');

END LOOP;

END;



2) Procedure to get the total cast members in a movie

Code:

--------------------------------------------------------------

create procedure totalcastMembers (@movieName varchar(50))

as

declare @totalDirector int,

@totalWriter int,

@totalCelebs int,

@totalCast int

begin

if Not Exists(Select moviename from movie where MovieName=@movieName )

begin

select 'Movie does not exist'

end

else

begin

-- Total number of Writers

Set @totalWriter=( Select Count(mw.Writerid) from movie m inner join Movie\_Bridge\_Writer mw on mw.MovieId=m.MovieId where @movieName=m.MovieName

Group by m.moviename );

-- Total number of Directors

Set @totalDirector=(Select Count(md.directorid) from movie m inner join Director\_Movie\_Bridge md on m.MovieId=md.MovieId where @movieName=m.MovieName

Group by m.moviename );

-- Total number of Celebrities

Set @totalCelebs=(Select count(mc.celebrityID) from Movie\_Bridge\_Celebrity mc inner join movie m on mc.MovieId=m.MovieId where @movieName=m.MovieName

Group by Moviename);

-- Total Cast

Set @totalCast=@totalWriter+@totalDirector+@totalCelebs;

print N'Total Writers: ' + Cast(@totalWriter as varchar(40));

print N'Total Directors: ' + Cast(@totalDirector as varchar(40));

print N'Total Celebrities: ' + Cast(@totalCelebs as varchar(40));

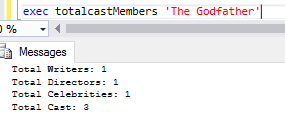
print N'Total Cast: ' +Cast(@totalCast as varchar(40));

end

end

----------------------------------------------------------------

exec totalcastMembers @movieName='The Godfather'



# FUNCTIONS

1)**Function to get the date difference from the current date in the format of Year, Months and Days**

**Code:**

CREATE FUNCTION dbo.GetDateDifference

(

@FromDate DATETIME, @ToDate DATETIME

)

RETURNS NVARCHAR(100)

AS

Begin

Declare @Output varchar(40)

IF(ISDATE(@FromDate)=1)

BEGIN

DECLARE @Years INT, @Months INT, @Days INT, @tmpFromDate DATETIME

SET @Years = DATEDIFF(YEAR, @FromDate, @ToDate)

- (CASE WHEN DATEADD(YEAR, DATEDIFF(YEAR, @FromDate, @ToDate),

@FromDate) > @ToDate THEN 1 ELSE 0 END)

SET @tmpFromDate = DATEADD(YEAR, @Years , @FromDate)

SET @Months = DATEDIFF(MONTH, @tmpFromDate, @ToDate)

- (CASE WHEN DATEADD(MONTH,DATEDIFF(MONTH, @tmpFromDate, @ToDate),

@tmpFromDate) > @ToDate THEN 1 ELSE 0 END)

SET @tmpFromDate = DATEADD(MONTH, @Months , @tmpFromDate)

SET @Days = DATEDIFF(DAY, @tmpFromDate, @ToDate)

- (CASE WHEN DATEADD(DAY, DATEDIFF(DAY, @tmpFromDate, @ToDate),

@tmpFromDate) > @ToDate THEN 1 ELSE 0 END)

Set @Output= 'Years: ' + CAST(@Years AS VARCHAR(4)) +

' Months: ' + CAST(@Months AS VARCHAR(2)) +

' Days: ' + CAST(@Days AS VARCHAR(2))

Return @Output;

End

Else

Begin

set @Output='Not a valid date';

Return @Output;

End

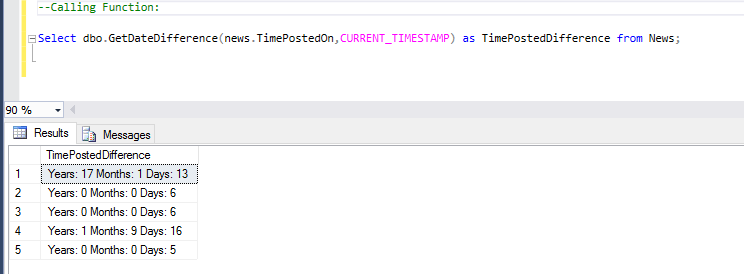
Return @Output;

End

GO

**Calling Function:**

Select dbo.GetDateDifference(news.TimePostedOn,CURRENT\_TIMESTAMP) from News



**2)** **Function to get the list of Celebrities born on any given day**

**Code:**

---------------------------------------------------------

Create function get\_Celebs\_Born\_Today(@currentdate smalldatetime)

returns Table

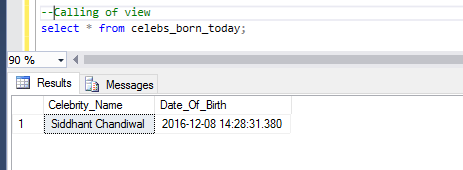
as

return select Celebrity.CelebrityName as Celebrity\_Name,Celebrity.DateofBirth as Date\_Of\_Birth from Celebrity where DAY(Celebrity.DateofBirth)=DAY(@currentdate) AND MONTH(Celebrity.DateofBirth)=MONTH(@currentdate);

-------------------------------------------------------------

**Calling the function:**

select \* from get\_Celebs\_Born\_Today(GETDATE());



3) Function to Calculate Award Winning percentage for Movies, TV Shows, Celebrities, Writers and Directors

**Code:**

create FUNCTION dbo.GetPercentage

(@Value1 VARCHAR(50), @ID INT)

RETURNS VARCHAR(50)

AS

BEGIN

DECLARE @Count1 Int,@Count2 Int,@Percentage Float, @output varchar(30)

--Movie

IF @Value1 = 'Movie'

BEGIN

IF NOT EXISTS(Select movieid from Nomination where movieid=@ID)

Begin

set @output='Not a valid ID'

SET @Percentage = 0

Return @output

End

Else

Begin

SET @Count1 = (Select COUNT(Nomination.movieid) from Nomination where Nomination.movieid =@ID And Nomination.Winner=1);

SET @Count2 = (Select COUNT(Nomination.movieid) from Nomination where Nomination.movieid =@ID);

END

IF @Count2 =0

BEGIN

SET @Percentage = 0

END

ELSE

BEGIN

SET @Percentage = (@Count1\*100)/@Count2;

END

END

--TV Show

IF @Value1 = 'TVShow'

BEGIN

IF NOT EXISTS(Select tvshowid from Nomination where tvshowid=@ID)

Begin

set @output='Not a valid ID'

SET @Percentage = 0

Return @output

End

Else

BEGIN

SET @Count1 = (Select COUNT(Nomination.tvshowid) from Nomination where Nomination.tvshowid =@ID And Nomination.Winner=1);

SET @Count2 = (Select COUNT(Nomination.tvshowid) from Nomination where Nomination.tvshowid =@ID);

END

IF @Count2 =0

BEGIN

SET @Percentage = 0

END

ELSE

BEGIN

SET @Percentage = (@Count1\*100)/@Count2;

END

END

--Celebrity

IF @Value1 = 'Celebrity'

BEGIN

IF NOT EXISTS(Select celebrityid from Nomination where celebrityid=@ID)

Begin

set @output='Not a valid ID'

SET @Percentage = 0

Return @output

End

Else

Begin

SET @Count1 = (Select COUNT(Nomination.celebrityid) from Nomination where Nomination.celebrityid =@ID And Nomination.Winner=1);

SET @Count2 = (Select COUNT(Nomination.celebrityid) from Nomination where Nomination.celebrityid =@ID);

END

IF @Count2 =0

BEGIN

SET @Percentage = 0

END

ELSE

BEGIN

SET @Percentage = (@Count1\*100)/@Count2;

END

END

--Director

IF @Value1 = 'Director'

BEGIN

IF NOT EXISTS(Select directorid from Nomination where directorid=@ID)

Begin

set @output='Not a valid ID'

SET @Percentage = 0

Return @output

End

Else

Begin

SET @Count1 = (Select COUNT(Nomination.directorid) from Nomination where Nomination.directorid =@ID And Nomination.Winner=1);

SET @Count2 = (Select COUNT(Nomination.directorid) from Nomination where Nomination.directorid =@ID);

END

IF @Count2 =0

BEGIN

SET @Percentage = 0

END

ELSE

BEGIN

SET @Percentage = (@Count1\*100)/@Count2;

END

END

IF @Value1 = 'Writer'

BEGIN

IF NOT EXISTS(Select writerid from Nomination where writerid=@ID)

Begin

set @output='Not a valid ID'

SET @Percentage = 0

Return @output

End

Else

Begin

SET @Count1 = (Select COUNT(Nomination.writerid) from Nomination where Nomination.writerid =@ID And Nomination.Winner=1);

SET @Count2 = (Select COUNT(Nomination.writerid) from Nomination where Nomination.writerid =@ID);

END

IF @Count2 =0

BEGIN

SET @Percentage = 0

END

ELSE

BEGIN

SET @Percentage = (@Count1\*100)/@Count2;

END

END

IF @Value1 Not IN ('Writer','Director','Movie','Celebrity','TVShow')

Begin

set @output='Not a valid input value'

Return @output

End

--RETURN CAST(CONCAT(@Percentage,'%') As varchar(50));

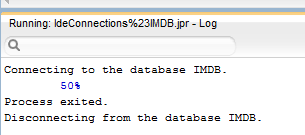
RETURN CAST(CONCAT(@Percentage,'%') As varchar(50));

END

Go

**Calling the Function**

Select dbo.GetPercentage('Director',1) AS WinningPercentage;



4) **Find the Total Number of TV-Co-actors an Actor worked with**

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE FUNCTION get\_Co\_Celebrity\_count

(

-- Add the parameters for the function here

@celebrityId int

)

RETURNS INT

AS

BEGIN

DECLARE @count int

DECLARE @finalCount int

DECLARE @outputmessage varchar(30)

IF NOT EXISTS(Select celebrityid from Celebrity\_Bridge\_TvShow where CelebrityId=@celebrityId)

BEGIN

set @outputmessage='Not a valid input'

set @finalCount=0

Return @finalCount;

END

ELSE

BEGIN

select @count= count(TVShowId) from Celebrity\_Bridge\_TvShow cbt where CelebrityId=@celebrityId;

with c1 as(

Select TVShowId,COUNT(Celebrity\_Bridge\_TvShow.CelebrityId)

As TotalCelebrity from Celebrity\_Bridge\_TvShow Group by TVShowId)

,c2 as (select TvShowID from Celebrity\_Bridge\_TvShow where CelebrityId=@celebrityId)

select @finalCount = sum(c1.TotalCelebrity)-@count from c1 JOIN c2 on c2.TVShowId=c1.TVShowId;

-- Return the result of the function

RETURN @finalCount;

END

RETURN @finalCount;

END

GO

**Calling the function**

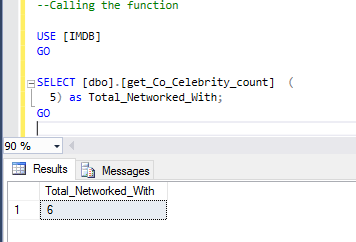
USE [IMDB]

GO

SELECT [dbo].[get\_Co\_Celebrity\_count] (

1) as Total\_Networked\_With;

GO



5) **Function to get the TopRated Movies based on IMdbRating, AverageUserRating, MovieTotalVotes, GrossIncome parameters**

CREATE FUNCTION get\_topRatedMovies

(

-- Add the parameters for the function here

@count int

)

RETURNS TABLE

AS

RETURN

Select TOP (@count) Movie.MovieName,Movie.MovieRating As ImdbRating,Movie.AverageUserRating,Movie.MovieTotalVotes,

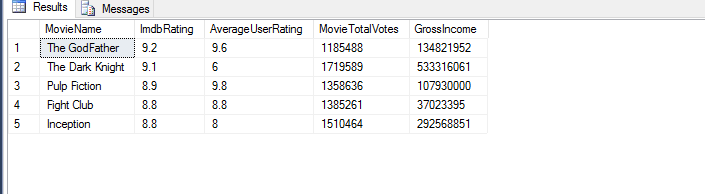
BoxOffice.GrossIncome From Movie

JOIN BoxOffice ON Movie.MovieId = BoxOffice.MovieId

Order By ImdbRating DESC,AverageUserRating DESC,MovieTotalVotes DESC,GrossIncome DESC;

**Calling Function**

select \* from get\_topRatedMovies(5);



# TRIGGERS

1) Creating a database trigger for updating the box office details for which the screening details has been inserted or updated.

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

CREATE TRIGGER addMovieRevenue ON Screening

AFTER INSERT,UPDATE

AS

BEGIN

DECLARE @revenueGenerated int,

@movieId int,

@moviename varchar(50)

--get the values from the inserted table into @revenueGeneratedTable

Select @revenueGenerated=DayCollection,@movieId=MovieId from inserted;

--get the name of the movie using @movieid

select @moviename=MovieName from Movie;

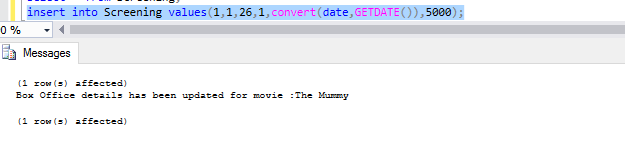
--Update the Box office table with the day's revenue generated from the screening table

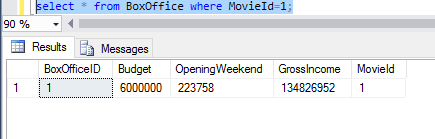
Update BoxOffice set GrossIncome = GrossIncome+@revenueGenerated where MovieId=@movieId;

PRINT 'Box Office details has been updated for movie :'+@moviename;

end;

Output:





**2)** Trigger for Episode table which will update Timeslot and Season table once an entry is inserted in Episode table

**Code:**

=================================================================

CREATE TRIGGER add\_time\_slot ON Episode

AFTER INSERT

AS

BEGIN

DECLARE @episodeId int,

@seasonId int,

@channel varchar(50),

@temp\_episodeId int

--get the values from the inserted table into @episodeId and @seasonId

Select @episodeId=EpisodeId,@seasonId=SeasonId from inserted;

--updating the total number of episodes for a particular season in Season table

update Season set Totalepisodes = Totalepisodes+1 where SeasonId=@seasonId;

PRINT N'-------------Total number of TV shows has been updated in the Season Table.------------------------';

--Retriving other Episode's Id of the same season

Select TOP 1 @temp\_episodeId=EpisodeId from Episode where SeasonId=@seasonId and EpisodeId!=@episodeId;

--Retriving Channel id of that episode to find the channel on which the telecast would happen

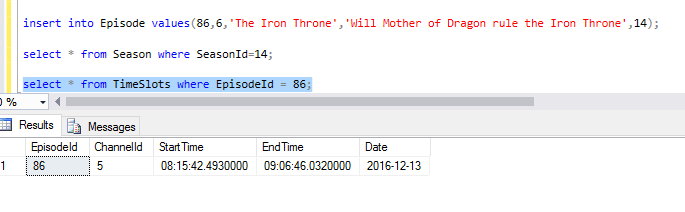
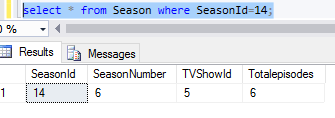
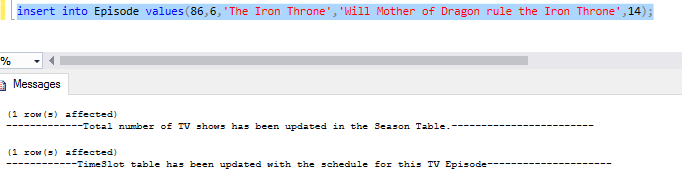
select @channel=ChannelId from TimeSlots where EpisodeId=@temp\_episodeId;

-- inserting into timeslot table with the channel id retrived above and randomly generated time slots and Date =date\_of\_insertion + 5 days

insert into TimeSlots values(@episodeId,@channel,dateadd(millisecond, cast(3600000 \* RAND() as int), convert(time, '08:00')),DATEADD(MINUTE,30,dateadd(millisecond, cast(3600000 \* RAND() as int), convert(time, '08:00'))),GETDATE()+5);

PRINT N'------------TimeSlot table has been updated with the schedule for this TV Episode---------------------';

end;



3) Trigger for User\_Movie\_Bridge table which will update Average User rating in Movie table once an entry is inserted in User\_Movie\_Bridge table

CREATE TRIGGER update\_avg\_rating ON User\_Movie\_Bridge

AFTER INSERT,UPDATE

AS

BEGIN

DECLARE @movieId int,

@userRating float,

@tempRating float,

@finalRating float,

@countUsers int

--get the values from the inserted table into @movieId and @userRating

Select @movieId=movieId,@userRating=UserRating from inserted;

Select @tempRating = AverageUserRating from Movie where MovieId=@movieId;

Select @countUsers=count(MovieId) from User\_Movie\_Bridge where MovieId=@movieId;

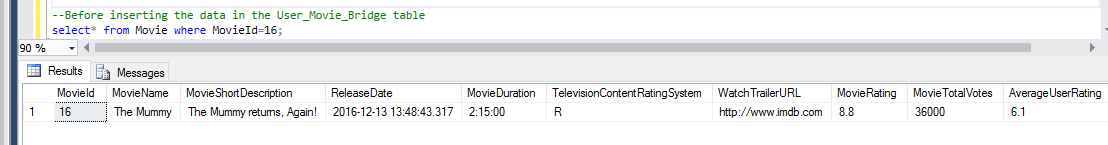
set @finalRating=((@tempRating\*(@countUsers-1))+@userRating)/@countUsers;

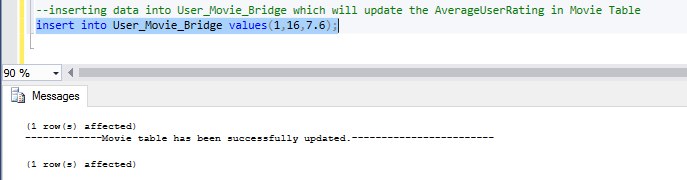
--updating the total number of episodes for a particular season in Season table

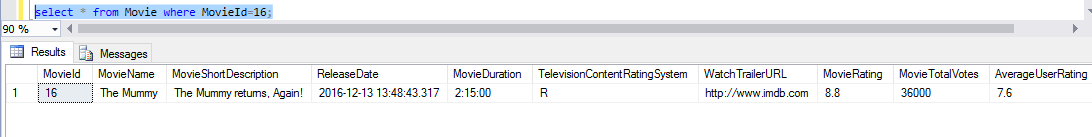
update Movie set AverageUserRating = @finalRating where MovieId=@movieId;

PRINT N'-------------Movie table has been successfully updated.------------------------';

end;







# QUERIES

1) **Displaying list of movies with their Genres and Count**

**Code:**

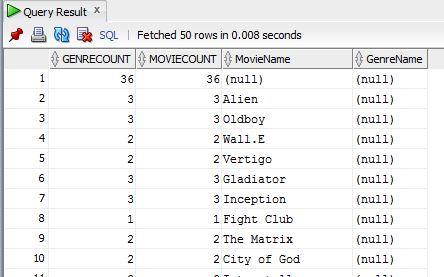
select Count(g.GenreId) as GenreCount,COUNT(m.MovieID) as MovieCount,MovieName, GenreName

from Genre g inner join Movie\_Genre\_BridgeTable mg

on mg.GenreId=g.GenreId inner join Movie m

on m.MovieId=mg.MovieId

GROUP BY Cube(GenreName,m.MovieName);



2) **Rank the movies according to the input by the user based on MovieRating or MovieTotalVotes or ReleaseDate**

ALTER procedure [dbo].[ranking](@input char(25))

as

begin

if(@input='MovieRating')

begin

select \* from dbo.Movie order by MovieRating desc

end

else if(@input='MovieTotalVotes')

begin

select \* from dbo.Movie order by MovieTotalVotes desc

end

else if (@input='ReleaseDate')

begin

select \* from dbo.Movie order by ReleaseDate desc

end

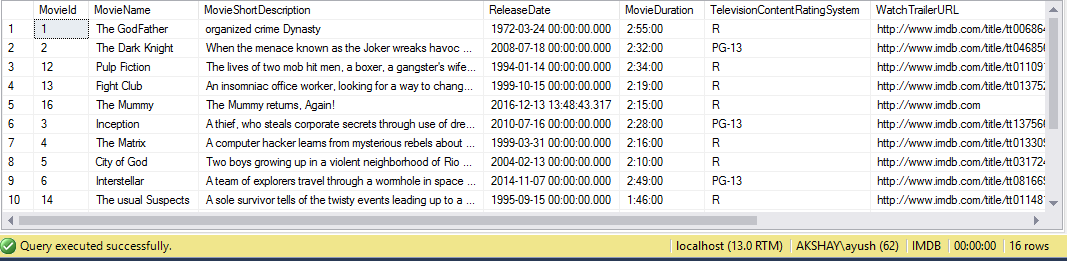
else

begin

select 'input not valid'

end

end



3**) Display the Movie Gross Collection in each Country and the Total Gross Collection**

--Creating View to store the Movie details based on its collection from each Theater

GO

Create VIEW TOTALCOLLECTIONVIEW

AS

Select Movie.MovieName,Country.CountryName,Theatre.TheatreName,SUM(Screening.DayCollection) AS TotalCollection FROM

City JOIN State ON

City.StateId = State.StateId JOIN Country ON

Country.CountryId = State.CountryId JOIN Address ON

Address.CityId = City.CityId JOIN Theatre ON

Theatre.AddressId =Address.AddressId JOIN Screening ON

Screening.TheatreId = Theatre.TheatreId JOIN Movie ON

Screening.MovieId = Movie.MovieId

GROUP BY Movie.MovieName,Country.CountryName,

Country.CountryName,Theatre.TheatreName,Screening.DayCollection;

GO

--Pivoting Based On the CountryName

Select MovieName,USA,INDIA,UK

FROM

(SELECT MovieName,CountryName,TotalCollection FROM TOTALCOLLECTIONVIEW) AS Totalollections

PIVOT(SUM(TotalCollection) FOR CountryName IN([USA],[INDIA],[UK])) AS pvt;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SELECT

(CASE WHEN GROUPING(MovieName) = 1 THEN 'Total' ELSE MovieName END) AS MOVIENAME,

SUM(USA) AS USA,

SUM(INDIA) AS INDIA,

SUM(UK) AS UK,

SUM(USA + INDIA + UK) AS Total

FROM

(

SELECT

MovieName,

-- Selecting Cases based on the CountryName

(CASE WHEN CountryName = 'USA' THEN TotalCollection ELSE 0 END) AS USA,

(CASE WHEN CountryName = 'INDIA' THEN TotalCollection ELSE 0 END) AS INDIA,

(CASE WHEN CountryName = 'UK' THEN TotalCollection ELSE 0 END) AS UK

FROM

(

-- Original input

SELECT

MovieName,

CountryName,SUM(TotalCollection) AS TotalCollection

FROM TOTALCOLLECTIONVIEW GROUP BY MovieName,CountryName

) i

) j

GROUP BY MovieName WITH ROLLUP

