List of tables and columns:

|  |
| --- |
| Movies   * movie\_id |
| * movie\_name |
| * movie\_rating |
| * movie\_duration |
| * metascore\_value |
| * movie\_description |
| * release\_date |
| * movie\_image |
| * imdb\_rating |
| * movie\_votes |
| * is\_oscar\_winner |
| * dvd\_link |
| * bluray\_link |
| News\_List   * news\_id |
| * type\_id   Poll\_PollNominations |
| * poll\_id |
| * nominations\_id   Award\_AwardCategory |
| * award\_id |
| * category\_id   Movies\_List |
| * list\_id |
| * movie\_id   Timeslots |
| * channel\_id |
| * episode\_id |
| * start\_time |
| * date |
| * end\_time   TV\_Shows\_Star |
| * star\_id |
| * tvshow\_id   Showtimes |
| * theatre\_id |
| * movie\_id |
| * show\_time * show\_date   Movies\_Star |
| * movie\_id |
| * star\_id   Movies\_Director |
| * movie\_id |
| * director\_id   Movies\_Genre |
| * movie\_id |
| * genre\_id   TVShows\_Watchlist |
| * tvshow\_id |
| * user\_id   TVShows\_Creator |
| * tvshow\_id |
| * creator\_id   Genre |
| * genre\_id |
| * genre\_name   Director |
| * director\_id |
| * director\_bio |
| * director\_image |
| * person\_id   Star |
| * star\_id |
| * star\_bio |
| * star\_image |
| * person\_id   Metacritic\_Review |
| * review\_id |
| * review\_score |
| * review\_description |
| * movie\_id   Magazine |
| * magazine\_id |
| * author\_id |
| * magazine\_id |
| * magazine\_name   Author |
| * author\_id |
| * author\_name |
| * magazine\_id   Person |
| * person\_id |
| * first\_name |
| * last\_name |
| * birth\_date |
| * birth\_month |
| * birth\_year   City |
| * city\_id |
| * city\_id |
| * city\_name   State |
| * state\_id |
| * state\_id |
| * state\_name   Country |
| * country\_id |
| * country\_id |
| * country\_name   TV\_Shows |
| * tvshow\_id |
| * tvshow\_name |
| * tvshow\_rating |
| * tvshow\_duration |
| * tvshow\_release\_year |
| * tvshow\_end\_year |
| * tvshow\_imdb\_rating |
| * tvshow\_description |
| * watchnow\_link |
| * tvshow\_image |
| * tvshow\_votes   Trailer |
| * trailer\_id |
| * trailer\_duration |
| * trailer\_link |
| * movie\_id   Theatre |
| * theatre\_id |
| * theatre\_name |
| * phone\_no |
| * Address\_id   Rating |
| * rating\_id |
| * rating\_score |
| * movie\_id |
| * user\_id |
| * tvshow\_id   Writer |
| * writer\_id |
| * writer\_bio |
| * writer\_image |
| * person\_id   User |
| * user\_id |
| * email\_id |
| * password |
| * person\_id   Address |
| * address\_id |
| * address\_line1 |
| * zipcode |
| * city\_id   Creator |
| * creator\_id |
| * creator\_bio |
| * creator\_image * person\_id   Episodes |
| * episode\_id |
| * episode\_no |
| * episode\_name |
| * episode\_description   Seasons |
| * season\_id |
| * season\_id |
| * season\_name |
| * tvshow\_id   Collection |
| * collection\_id |
| * collection\_amount |
| * collection\_date |
| * collection\_day |
| * movie\_id   Award |
| * award\_id |
| * award\_name |
| * award\_date |
| * award\_description   Poll |
| * poll\_id |
| * poll\_name |
| * created\_by |
| * poll\_description |
| * poll\_image |
| * is\_featured\_poll |
| * is\_hotly\_poll |
| * poll\_link   Release\_List |
| * list\_id |
| * list\_name |
| * list\_author |
| * list\_published\_on |
| * list\_image   Award\_Category |
| * category\_id |
| * category\_name   Poll\_Nominations |
| * nominations\_id |
| * is\_winner |
| * movie\_id |
| * tvshow\_id   Channel |
| * channel\_id |
| * channel\_name |
| * channel\_logo   Award\_Nominations |
| * nominations\_id |
| * is\_winner |
| * movie\_id |
| * director\_id |
| * star\_id |
| * tvshow\_id   News |
| * news\_id |
| * news\_title |
| * published\_date |
| * news\_image |
| * news\_description |
| * news\_source\_name |
| * recent\_source\_name\_news |
| * permalink |
| * full\_article\_link |
| * movie\_id |
| * director\_id |
| * star\_id |
| * tvshow\_id   News\_Type |
| * type\_id |
| * type\_name   Category\_Nominations |
| * category\_id |
| * nominations\_id   Movies\_Writer |
| * movie\_id |
| * writer\_id   Movies\_Watchlist |
| * movie\_id |
| * user\_id |

List of indexes:

|  |  |
| --- | --- |
| INDEX\_NAME | TABLE\_NAME |
| email\_id | User |
| IX\_Relationship | Creator |
| IX\_Relationship1 | Rating |
| IX\_Relationship10 | Metacritic\_Review |
| IX\_Relationship14 | Director |
| IX\_Relationship15 | Star |
| IX\_Relationship16 | Writer |
| IX\_Relationship17 | User |
| IX\_Relationship18 | Person |
| IX\_Relationship19 | City |
| IX\_Relationship20 | State |
| IX\_Relationship22 | Trailer |
| IX\_Relationship23 | Address |
| IX\_Relationship24 | Theatre |
| IX\_Relationship27 | Rating |
| IX\_Relationship28 | Rating |
| IX\_Relationship29 | Collection |
| IX\_Relationship37 | Seasons |
| IX\_Relationship38 | Episodes |
| IX\_Relationship46 | Award\_Nominations |
| IX\_Relationship47 | Award\_Nominations |
| IX\_Relationship48 | Award\_Nominations |
| IX\_Relationship49 | Award\_Nominations |
| IX\_Relationship52 | Poll\_Nominations |
| IX\_Relationship54 | Poll\_Nominations |
| IX\_Relationship59 | News |
| IX\_Relationship60 | News |
| IX\_Relationship61 | News |
| IX\_Relationship62 | News |
| IX\_Relationship7 | Metacritic\_Review |
| IX\_Relationship8 | Metacritic\_Review |
| IX\_Relationship9 | Author |
| Key1 | Movies |
| Key10 | Director |
| Key11 | Star |
| Key12 | Trailer |
| Key13 | User |
| Key15 | Address |
| Key16 | Theatre |
| Key17 | Rating |
| Key18 | Writer |
| Key19 | Collection |
| Key2 | Genre |
| Key20 | Channel |
| Key21 | TV\_Shows |
| Key22 | Creator |
| Key23 | Episodes |
| Key24 | Release\_List |
| Key25 | News |
| Key26 | News\_Type |
| Key27 | Award |
| Key28 | Award\_Category |
| Key29 | Award\_Nominations |
| Key3 | Metacritic\_Review |
| Key30 | Poll |
| Key31 | Poll\_Nominations |
| Key32 | Movies\_Genre |
| Key33 | Movies\_Director |
| Key34 | Movies\_Star |
| Key35 | Movies\_Writer |
| Key36 | Showtimes |
| Key37 | TV\_Shows\_Star |
| Key38 | TVShows\_Creator |
| Key39 | Timeslots |
| Key4 | Magazine |
| Key40 | Seasons |
| Key41 | Movies\_List |
| Key42 | Award\_AwardCategory |
| Key43 | Category\_Nominations |
| Key44 | Poll\_PollNominations |
| Key45 | News\_List |
| Key46 | Movies\_Watchlist |
| Key47 | TVShows\_Watchlist |
| Key5 | Author |
| Key6 | Person |
| Key7 | City |
| Key8 | State |
| Key9 | Country |
| phone\_no | Theatre |

List of Procedures

Procedure: PrintChannelTVShowData (SQL Server)

Used to display the TV show name along with their episodes and season number telecasted on a particular channel and date.

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

-- Template generated from Template Explorer using:

-- Create Procedure (New Menu).SQL

--

-- Use the Specify Values for Template Parameters

-- command (Ctrl-Shift-M) to fill in the parameter

-- values below.

--

-- This block of comments will not be included in

-- the definition of the procedure.

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

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

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

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

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

CREATE PROCEDURE PrintChannelTVShowData

-- Add the parameters for the stored procedure here

@channel\_id int,

@date date

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

if exists(select dbo.Timeslots.channel\_id, dbo.Timeslots.date, dbo.Timeslots.start\_time, dbo.Timeslots.end\_time from dbo.Timeslots where channel\_id=@channel\_id AND date=@date)

begin

Select dbo.Timeslots.channel\_id, dbo.Timeslots.date, dbo.Timeslots.start\_time, dbo.Timeslots.end\_time, dbo.Episodes.episode\_name, dbo.Seasons.season\_name, dbo.TV\_Shows.tvshow\_name

from Timeslots inner join dbo.Episodes on dbo.Episodes.episode\_id = dbo.Timeslots.episode\_id inner join dbo.Seasons on dbo.Seasons.season\_id = dbo.Episodes.season\_id inner join dbo.TV\_Shows on dbo.TV\_Shows.tvshow\_id = dbo.Seasons.tvshow\_id where channel\_id=@channel\_id AND date=@date;

end;

else

begin

print 'No data found';

end;

END

GO

Output: PrintChannelTVShowData 1,'2016-12-04';

Procedure: PrintChannelTVShowData – Oracle

create or replace PROCEDURE PrintChannelTVShowData (

p\_channel\_id IN number,

p\_date IN date)

AS

readChannelID NUMBER(38,0);

readDate DATE;

readStartTime VARCHAR2(5);

readEndTime VARCHAR2(5);

readEpisodeName VARCHAR2(50);

readSeasonName VARCHAR2(20);

readTVShowName VARCHAR2(30);

is\_found\_rec boolean := false;

Cursor timeslotsDataCursor IS SELECT "Timeslots"."channel\_id", "Timeslots"."DATE\_VALUE", "Timeslots"."start\_time", "Timeslots"."end\_time", "Episodes"."episode\_name", "Seasons"."season\_name", "TV\_Shows"."tvshow\_name" from "Timeslots" inner join "Episodes" on "Episodes"."episode\_id" = "Timeslots"."episode\_id" inner join "Seasons" on "Seasons"."season\_id" = "Episodes"."season\_id" inner join "TV\_Shows" on "TV\_Shows"."tvshow\_id" = "Seasons"."tvshow\_id" where "Timeslots"."channel\_id"=p\_channel\_id AND "Timeslots"."DATE\_VALUE"=p\_date;

BEGIN

for timeslotData in timeslotsDataCursor

loop

is\_found\_rec := true;

readChannelID := timeslotData."channel\_id";

readDate := timeslotData."DATE\_VALUE";

readStartTime := timeslotData."start\_time";

readEndTime := timeslotData."end\_time";

readEpisodeName := timeslotData."episode\_name";

readSeasonName := timeslotData."season\_name";

readTVShowName := timeslotData."tvshow\_name";

DBMS\_OUTPUT.PUT\_LINE(' Channel ID = '||readChannelID|| ' Date = ' || readDate || ' Start time = '||readStartTime|| ' End time = '||readEndTime||' Episode Name = '||readEpisodeName||' Season Name = '||readSeasonName||' TV Show name = '||readTVShowName);

end loop;

if not is\_found\_rec then

DBMS\_OUTPUT.PUT\_LINE('No records found');

end if;

END;

Output:

SET SERVEROUTPUT ON

EXEC PRINTCHANNELTVSHOWDATA(1,'06-DEC-16');

CREATE PROCEDURE DisplayCollectionsData

-- Add the parameters for the stored procedure here

AS

BEGIN

Declare @movieID int, @weekendCollections Float, @grossCollections Float, @movie\_id CHAR(2), @totalCollections varchar(30);

DECLARE moviesCursor CURSOR FOR

select Collection.movie\_id, SUM(collection\_amount) AS GrossCollections from Collection group by movie\_id order by GrossCollections desc

open moviesCursor

print 'MovieID GrossCollections';

FETCH NEXT FROM moviesCursor INTO @movieID, @grossCollections

WHILE @@FETCH\_STATUS = 0

-- Set @weekendCollections = (SELECT SUM(Collection.collection\_amount) AS WeekendCollections from Collection WHERE collection\_day IN ('Saturday', 'Sunday'));

-- Set @grossCollections = (SELECT SUM(Collection.collection\_amount) AS GrossCollections from Collection);

-- Insert statements for procedure here

BEGIN

-- SET @weekendCollections = (SELECT SUM(Collection.collection\_amount) AS WeekendCollections from Collection WHERE collection\_day IN ('Saturday', 'Sunday') AND movie\_id = @movieID);

--Set @grossCollections = (SELECT SUM(Collection.collection\_amount) AS GrossCollections from Collection Where movie\_id = @movieID);

Select @movie\_id = CAST((@movieID) AS CHAR(2));

Select @totalCollections = CAST((@grossCollections) AS varchar(30));

print @movie\_id+' '+@totalCollections;

-- Select movie\_id, (SELECT SUM(Collection.collection\_amount) from Collection WHERE collection\_day IN ('Saturday', 'Sunday') AND movie\_id = @movieID) AS WeekendCollections, SUM(Collection.collection\_amount) AS GrossCollections from Collection where movie\_id = @movieID Group By movie\_id Order By GrossCollections DESC;

FETCH NEXT FROM moviesCursor INTO @movieID, @grossCollections

END

CLOSE moviesCursor

DEALLOCATE moviesCursor

END

GO

DisplayCollectionsData

TRIGGERS – SQL SERVER

Trigger: ConvertHashPassword

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

-- Template generated from Template Explorer using:

-- Create Trigger (New Menu).SQL

--

-- Use the Specify Values for Template Parameters

-- command (Ctrl-Shift-M) to fill in the parameter

-- values below.

--

-- See additional Create Trigger templates for more

-- examples of different Trigger statements.

--

-- This block of comments will not be included in

-- the definition of the function.

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

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

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

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

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

CREATE TRIGGER ConvertHashPassword

ON dbo.[User]

AFTER INSERT

AS

BEGIN

Declare @fetchedPassword Varchar(32), @binaryPassword varbinary(32), @hashedPassword Varchar(max), @userID int;

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for trigger here

SET @userID = (Select USER\_ID from inserted);

SET @fetchedPassword = (Select password from inserted);

print @fetchedPassword;

SET @binaryPassword = HASHBYTES('MD5', @fetchedPassword);

print @binaryPassword;

SET @hashedPassword = CAST('' AS XML).value('xs:hexBinary(sql:variable("@binaryPassword"))', 'VARCHAR(32)');

print @hashedPassword;

Update dbo.[User]

SET password = @hashedPassword where user\_id=@userID;

END

GO

Output: INSERT INTO dbo.[User] (user\_id, email\_id, password, person\_id) Values (74, 'aadesh@gmail.com', 'dock443', 3);

**Trigger: UpdateMetascore (SQL Server)**

create trigger updateMetascore

on dbo.Metacritic\_review

after insert

AS

BEGIN

DECLARE @movieID as int;

declare @temp as int;

select @movieID = (select movie\_id from inserted);

select @temp=avg(review\_score) from dbo.Metacritic\_Review where movie\_id = @movieId

update dbo.Movies set metascore\_value = @temp where movie\_id = @movieId

END;

Output: insert into Metacritic\_Review values( 32,80, 'beautiful movie',1,1,1);

Trigger (Oracle)

create or replace TRIGGER UPDATEMETASCORE

BEFORE INSERT ON "Metacritic\_Review"

REFERENCING NEW AS NEW OLD AS OLD

for EACH ROW

DECLARE

v\_movieID number;

v\_temp number;

BEGIN

v\_movieID := :new."movie\_id";

Select avg("review\_score") INTO v\_temp from "Metacritic\_Review" where "movie\_id"=v\_movieID;

update "Movies"

Set "metascore\_value" = v\_temp where "movie\_id"=v\_movieID;

END;

Function: CalculateIMDBRating – SQL Server

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

-- Template generated from Template Explorer using:

-- Create Scalar Function (New Menu).SQL

--

-- Use the Specify Values for Template Parameters

-- command (Ctrl-Shift-M) to fill in the parameter

-- values below.

--

-- This block of comments will not be included in

-- the definition of the function.

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

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

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

-- Author: <Author,,Name>

-- Create date: <Create Date, ,>

-- Description: <Description, ,>

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

CREATE FUNCTION CalculateIMDBRating

(

@movie\_id int

)

RETURNS float

AS

BEGIN

-- Declare the return variable here

DECLARE @sum float, @count int, @imdb\_rating float;

-- Add the T-SQL statements to compute the return value here

SET @sum = (Select SUM(Rating.rating\_score) from Rating where movie\_id=@movie\_id);

SET @count = (Select Count(Rating.rating\_score) from Rating where movie\_id=@movie\_id);

SET @imdb\_rating = @sum/@count;

-- Return the result of the function

RETURN @imdb\_rating;

END

GO

Output:

SELECT dbo.CalculateIMDBRating(1) AS IMDB\_Rating;

update Movies

set imdb\_rating = dbo.CalculateIMDBRating(14) where movie\_id=14;

select Movies.imdb\_rating from Movies where movie\_id=14;

Function: Get Date Difference - published News

CREATE FUNCTION dbo.GetDateDifference

(

@publishedDate datetime, @todaysDate datetime

)

RETURNS NVARCHAR(150)

AS

BEGIN

DECLARE @Years int, @Months int, @Days int, @tmpFromDate datetime

SET @Years = DATEDIFF(YEAR, @publishedDate, @todaysDate)

- (CASE WHEN DATEADD(YEAR, DATEDIFF(YEAR, @publishedDate, @todaysDate),

@publishedDate) > @todaysDate THEN 1 ELSE 0 END)

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

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

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

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

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

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

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

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

RETURN 'Years: ' + CAST(@Years AS VARCHAR(4)) +

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

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

END

GO

Output:

insert into News (news\_id, news\_title, published\_date, news\_description, news\_source\_name, tvshow\_id) values (14,'News Title','2010-06-22','News Description','News Source Name',1);

Select dbo.GetDateDifference(news.published\_date ,CURRENT\_TIMESTAMP) AS DateDiffrence from News;

Query 1: Location hierarchy collection analysis based on entertainment, revenue generated

select sum(c.collection\_amount) as grossIncome,

cy.city\_name, st.state\_name, ct.country\_name from Movies m

inner join Collection c

on c.movie\_id = m.movie\_id

inner join Showtimes s

on s.movie\_id = c.movie\_id

inner join Theatre t

on t.theatre\_id = s.theatre\_id

inner join Address a

on a.address\_id = t.address\_id

inner join city cy

on cy.city\_id = a.city\_id

inner join State st

on st.state\_id = cy.state\_id

inner join Country ct

on ct.country\_id = st.country\_id

GROUP BY rollup ( ct.country\_name, st.state\_name, cy.city\_name)

order by grossIncome desc

Query 2: Analysing which genre movies are most popular

select movie\_id, movie\_name, Family, Drama, Crime, Horror, Mystery, Adventure FROM

(select m.movie\_id, m.movie\_name, g.genre\_name

from Movies m inner join Movies\_Genre mg

on m.movie\_id = mg.movie\_id

inner join Genre g

on g.genre\_id = mg.genre\_id) As NumberOfMovies

pivot (count(genre\_name) for genre\_name in ([Family], [Drama], [Crime], [Horror], [Mystery], [Adventure])) as pvt

same analysis - oracle

select "movie\_id", Family, Drama, Crime, Horror, Mystery, Adventure FROM

(select m."movie\_id", g."genre\_name"

from "Movies" m inner join "Movies\_Genre" mg

on m."movie\_id" = mg."movie\_id"

inner join "Genre" g

on g."genre\_id" = mg."genre\_id")

pivot

(count("genre\_name") for "genre\_name" in ('Family' as Family, 'Drama' as Drama, 'Crime' as Crime, 'Horror' as Horror, 'Mystery' as Mystery, 'Adventure' as Adventure));

Query 3 – To determine most preferred/ watched movie.

Sql server

select movie\_name, COUNT(Movies\_Watchlist.movie\_id) AS MoviesCount,

RANK () OVER (ORDER BY COUNT(Movies\_Watchlist.movie\_id) DESC) AS Rank

from Movies\_Watchlist

INNER JOIN Movies

ON Movies.movie\_id=Movies\_Watchlist.movie\_id

group by Movies\_Watchlist.movie\_id, movie\_name;

same query - oracle

select "movie\_name", COUNT("Movies\_Watchlist"."movie\_id") AS MoviesCount,

RANK () OVER (ORDER BY COUNT("Movies\_Watchlist"."movie\_id") DESC) AS Rank from "Movies\_Watchlist" INNER JOIN "Movies" ON "Movies"."movie\_id"="Movies\_Watchlist"."movie\_id" group by "Movies\_Watchlist"."movie\_id", "movie\_name";

Query 4 – What genre movies are chosen by a particular age group users

select m.movie\_id, m.movie\_name, m.movie\_duration , g.genre\_name,

CONCAT(p.first\_name, ' ' , p.last\_name) as User\_Name

from Movies m join Movies\_Genre mg

on m.movie\_id = mg.movie\_id

inner join Genre g

on mg.genre\_id = g.genre\_id

inner join Movies\_Watchlist w

on m.movie\_id = w.movie\_id

join [IMDB\_Backup].[dbo].[User] u

on w.user\_id = u.user\_id

join Person p

on u.person\_id = p.person\_id

where p.birth\_year>1995

Query 5: Calculate diff between weekday and weekend collections

select m.movie\_name , sum(c.collection\_amount) as Collection\_amount ,'Weekend Sales' as Sales

from Collection c inner join Movies m

on c.movie\_id = m.movie\_id

where c.collection\_day in ('Saturday', 'Sunday')

group by movie\_name

union

select m.movie\_name, sum(c.collection\_amount)as Collection\_amount, 'Weekday Sales' as Sales from Collection c inner join Movies m

on c.movie\_id = m.movie\_id

where c.collection\_day not in ('Saturday', 'Sunday')

group by movie\_name

Query 6: Top Rated Top 10 Movies

select Movies.movie\_id, Movies.movie\_name, (SUM(Rating.rating\_score)/COUNT(Rating.rating\_score)) AS IMDB\_Rating

FROM Rating INNER JOIN Movies

ON Movies.movie\_id = Rating.movie\_id

Group by Movies.movie\_id, Movies.movie\_name

Order By IMDB\_Rating Desc;

- Oracle

select "Movies"."movie\_id", "Movies"."movie\_name", (SUM("Rating"."rating\_score")/COUNT("Rating"."rating\_score")) AS IMDB\_Rating FROM "Rating" INNER JOIN "Movies" ON "Movies"."movie\_id" = "Rating"."movie\_id" Group by "Movies"."movie\_id", "Movies"."movie\_name" Order By IMDB\_Rating Desc;

**View on AWARDS sql server**

create view v\_AwardDetails

as

select award.award\_name,c.category\_name ,

coalesce(

(select concat(p.first\_name, p.last\_name) from star s, person p

where s.star\_id = an.star\_id

and s.person\_id = p.person\_id),

(select concat(p.first\_name, p.last\_name) from director d, person p

where d.director\_id = an.director\_id

and d.director\_id = p.person\_id),

(select m.movie\_name from

Movies m where m.movie\_id = an.movie\_id),

(select t.tvshow\_name from

TV\_Shows t where t.tvshow\_id = an.tvshow\_id)) as nominee\_id,

case when is\_winner = 1

then 'Winner'

else

' '

end as Winner

from Award\_AwardCategory a inner join Award\_Category c

on a.category\_id = c.category\_id

inner join Award award

on award.award\_id = a.award\_id

inner join Category\_Nominations cn

on c.category\_id = cn.category\_id

inner join Award\_Nominations an

on an.nominations\_id = cn.nominations\_id

select \* from v\_AwardDetails

create view v\_MovieTheatreDetails

as

select m.movie\_name as Movie, g.genre\_name as Genre , concat(pd.first\_name, pd.last\_name) as Cast , 'Director' AS Role, t.theatre\_name as Theatre, sh.show\_time as ShowTime

from Movies m inner join Movies\_Director md

on m.movie\_id = md.movie\_id

inner join Director d

on md.director\_id = d.director\_id

inner join Person pd

on d.person\_id = pd.person\_id

inner join Movies\_Genre mg

on mg.movie\_id = m.movie\_id

inner join genre g

on mg.genre\_id = g.genre\_id

inner join Showtimes sh

on sh.movie\_id = m.movie\_id

inner join Theatre t

on t.theatre\_id = sh.theatre\_id

UNION

select m.movie\_name, g.genre\_name, concat(pw.first\_name,pw.last\_name) , 'Writer' as Role ,t.theatre\_name, sh.show\_time

from Movies m

inner join Movies\_Writer mw

on m.movie\_id = mw.movie\_id

inner join Writer w

on mw.writer\_id = w.writer\_id

inner join Person pw

on w.person\_id = pw.person\_id

inner join Movies\_Genre mg

on mg.movie\_id = m.movie\_id

inner join genre g

on mg.genre\_id = g.genre\_id

inner join Showtimes sh

on sh.movie\_id = m.movie\_id

inner join Theatre t

on t.theatre\_id = sh.theatre\_id

union

select m.movie\_name, g.genre\_name, concat(ps.first\_name,ps.last\_name) , 'Star' as Role , t.theatre\_name, sh.show\_time

from Movies m

inner join Movies\_Star ms

on m.movie\_id = ms.movie\_id

inner join Star s

on ms.star\_id = s.star\_id

inner join Person ps

on s.person\_id = ps.person\_id

inner join Movies\_Genre mg

on mg.movie\_id = m.movie\_id

inner join genre g

on mg.genre\_id = g.genre\_id

inner join Showtimes sh

on sh.movie\_id = m.movie\_id

inner join Theatre t

on t.theatre\_id = sh.theatre\_id

select \* from v\_MovieTheatreDetails;

view v\_MovieDetails – Displays movie details with corresponding actors, directors, writer, genre and so on.

create view v\_MovieDetails

as

select m.movie\_name as Movie, g.genre\_name as Genre , concat(pd.first\_name, pd.last\_name) as Cast , 'Director' AS Role

from Movies m inner join Movies\_Director md

on m.movie\_id = md.movie\_id

inner join Director d

on md.director\_id = d.director\_id

inner join Person pd

on d.person\_id = pd.person\_id

inner join Movies\_Genre mg

on mg.movie\_id = m.movie\_id

inner join genre g

on mg.genre\_id = g.genre\_id

UNION

select m.movie\_name, g.genre\_name, concat(pw.first\_name,pw.last\_name) , 'Writer' as Role

from Movies m

inner join Movies\_Writer mw

on m.movie\_id = mw.movie\_id

inner join Writer w

on mw.writer\_id = w.writer\_id

inner join Person pw

on w.person\_id = pw.person\_id

inner join Movies\_Genre mg

on mg.movie\_id = m.movie\_id

inner join genre g

on mg.genre\_id = g.genre\_id

union

select m.movie\_name, g.genre\_name, concat(ps.first\_name,ps.last\_name) , 'Star' as Role

from Movies m

inner join Movies\_Star ms

on m.movie\_id = ms.movie\_id

inner join Star s

on ms.star\_id = s.star\_id

inner join Person ps

on s.person\_id = ps.person\_id

inner join Movies\_Genre mg

on mg.movie\_id = m.movie\_id

inner join genre g

on mg.genre\_id = g.genre\_id

select \* from v\_MovieDetails;

view v\_MovieTheaterDetails – Displays movies running at different theatres, corresponding address and their show times.

create view v\_MovieTheaterDetails

as

(select m.movie\_name as Movie, t.theatre\_id , concat(address\_line1,',',zipcode,',',city\_name,',',state\_name,',',country\_name)as Address, Show\_Date, show\_time

from Movies m

inner join Movies\_Star ms

on m.movie\_id = ms.movie\_id

inner join Showtimes sh

on sh.movie\_id = m.movie\_id

inner join Theatre t

on t.theatre\_id = sh.theatre\_id

inner join Address a

on t.address\_id = a.address\_id

inner join City c

on c.city\_id =a.city\_id

inner join State s

on s.state\_id = c.state\_id

inner join country co

on s.country\_id = co.country\_id

)

select \* from dbo.MovieTheaterDetails

View – Top rated movies to display top movies based on their imdb rating, collection amount and user votes

CREATE VIEW V\_TopRatedMovies AS

select m.movie\_name as 'TOP RATED MOVIES', movie\_description, m.imdb\_rating, c.collection\_amount, m.movie\_votes

from Movies m inner join Collection c

on c.movie\_id = m.movie\_id

where m.imdb\_rating > 8 AND m.metascore\_value > 80

AND m.movie\_votes > 1000000 and c.collection\_amount>500000

order by m.imdb\_rating desc

SELECT \* FROM V\_ TopRatedMovies

View V\_UpcomingReleases to display movies yet to be released

create view V\_UpcomingReleases AS

select \* from Movies m where m.release\_date > GETDATE()

SELECT \* FROM V\_UpcomingReleases

view V\_MovieReviewsAuthorwise to display the movie’s metacritic review is given by which magazine and its corresponding author

create view V\_MovieReviewsAuthorwise AS

select m.movie\_id, m.movie\_name, m.metascore\_value, mr.review\_score,

mr.review\_description, z.magazine\_name, a.author\_name

from Movies m inner join Metacritic\_Review mr

on m.movie\_id = mr.movie\_id

inner join Magazine z

on z.magazine\_id = mr.magazine\_id

inner join Author a

on a.author\_id = mr.author\_id

SELECT \* FROM V\_ MovieReviewsAuthorwise

create view v\_TVShowDetails as

select t.tvshow\_name , concat(pd.first\_name, pd.last\_name) as Cast , 'Creator' AS Role

from TV\_Shows t inner join TVShows\_Creator md

on md.tvshow\_id = t.tvshow\_id

inner join Creator d

on d.creator\_id = md.creator\_id

inner join Person pd

on pd.person\_id = d.person\_id

union

select tv.tvshow\_name as tvshow, concat(ps.first\_name,ps.last\_name) as Cast , 'Star' as Role

from TV\_Shows tv

inner join TV\_Shows\_Star ms

on ms.tvshow\_id = tv.tvshow\_id

inner join Star s

on s.star\_id = ms.star\_id

inner join Person ps

on ps.person\_id = s.person\_id

select \* from dbo.v\_TVShowDetails