PRODUCTION

Movie Master

DIRECTOR

Team 20

CAMERA

Yihong, Qi, Yue, Yuting, Yinxing

DATE	SCENE	TAKE
04/23/2022	20	#1

Table of Contents.

ERD Part 01

People related Entities

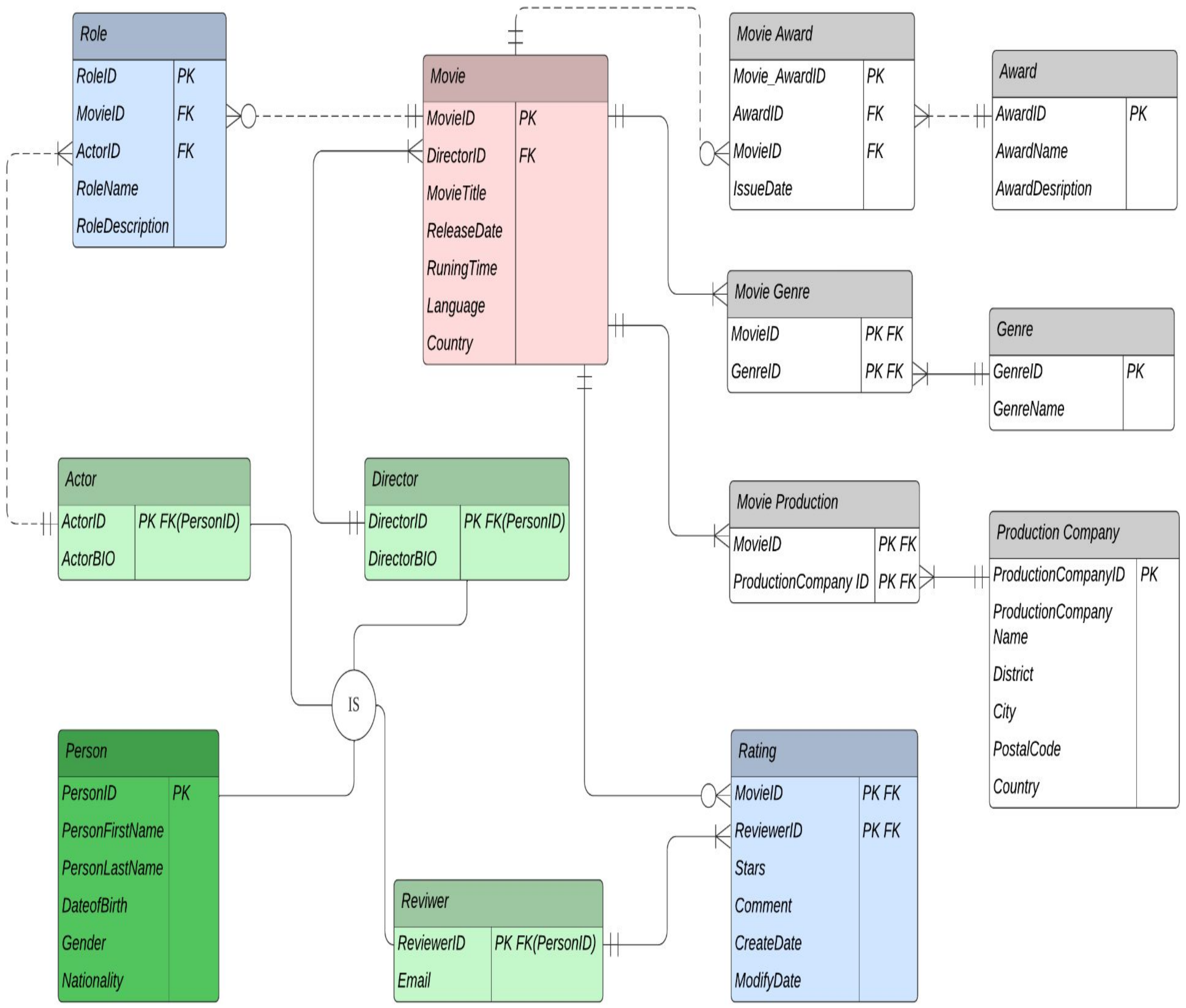
Each movie can have zero or many actors, but each actor should in one movie at least

Each actor can cast one or many roles in any movie

Each movie can only have one director, but one director can direct one to many movies

Each movie can have one or many rating records, but each each reviewer should review at least one movie

A person can be a actor, director, reviewer, or both (Supertype/Subtype)



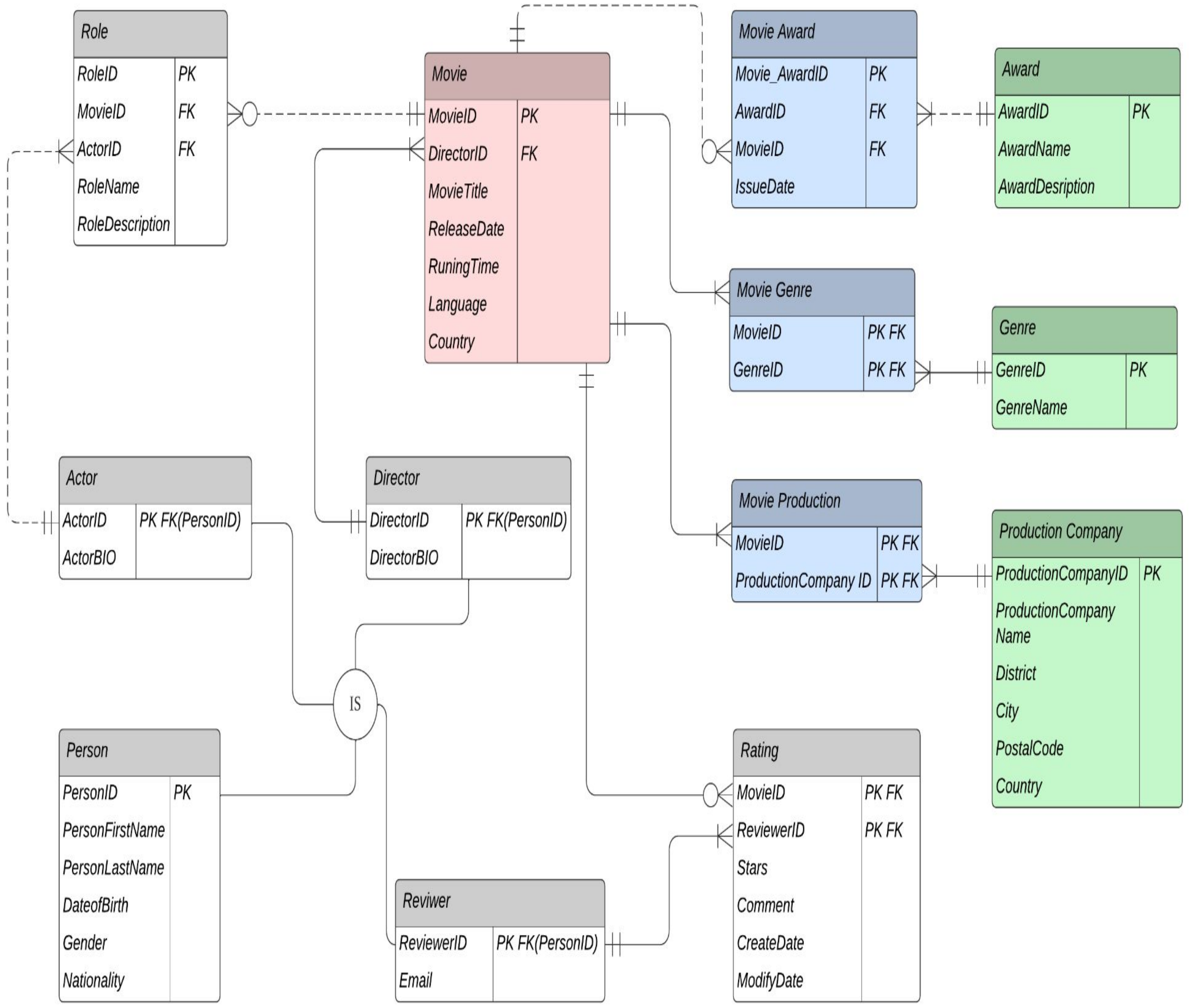
ERD Part 02

Movie related Entities

Each movie can have zero or many awards, but each award should be assigned to at least one movie before.

Each movie have at least one genre, and each genre can be assigned to one or many movies.

Each movie can have at least one production company or many companies, and each production company should have produced at least one movie or many movies.



Role	
RoleID	PK
MovieID	FK
ActorID	FK
RoleName	
RoleDescription	

Movie	
MovieID	PK
DirectorID	FK
MovieTitle	
ReleaseDate	
RuningTime	
Language	
Country	

Movie Award	
Movie_AwardID	PK
AwardID	FK
MovieID	FK
IssueDate	

Award	
AwardID	PK
AwardName	
AwardDesription	

Movie Genre	
MovieID	PK FK
GenreID	PK FK

Genre	
GenreID	PK
GenreName	

Movie Production	
MovieID	PK FK
ProductionCompany ID	PK FK

Production Company	
ProductionCompanyID	PK
ProductionCompany Name	
District	
City	
PostalCode	
Country	

Actor	
ActorID	PK FK(PersonID)
ActorBIO	

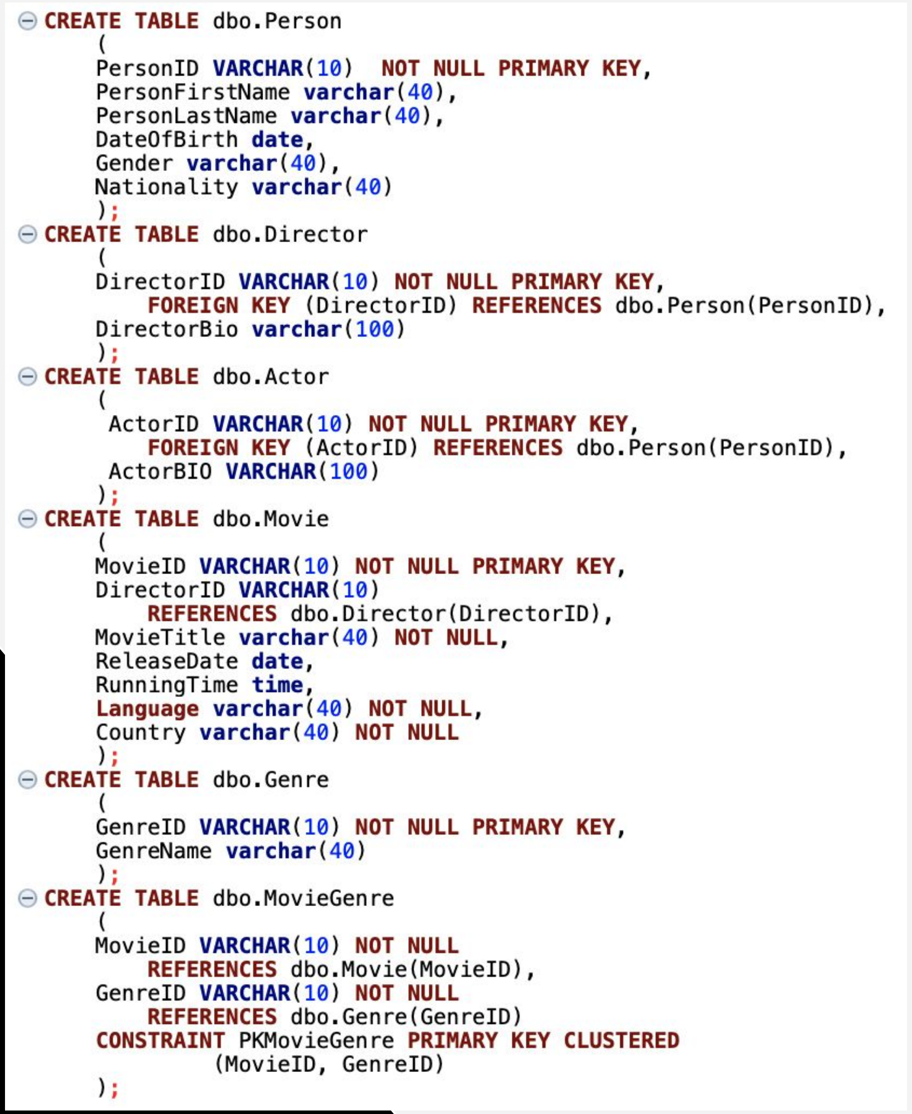
Director	
DirectorID	PK FK(PersonID)
DirectorBIO	

Person	
PersonID	PK
PersonFirstName	
PersonLastName	
DateofBirth	
Gender	
Nationality	

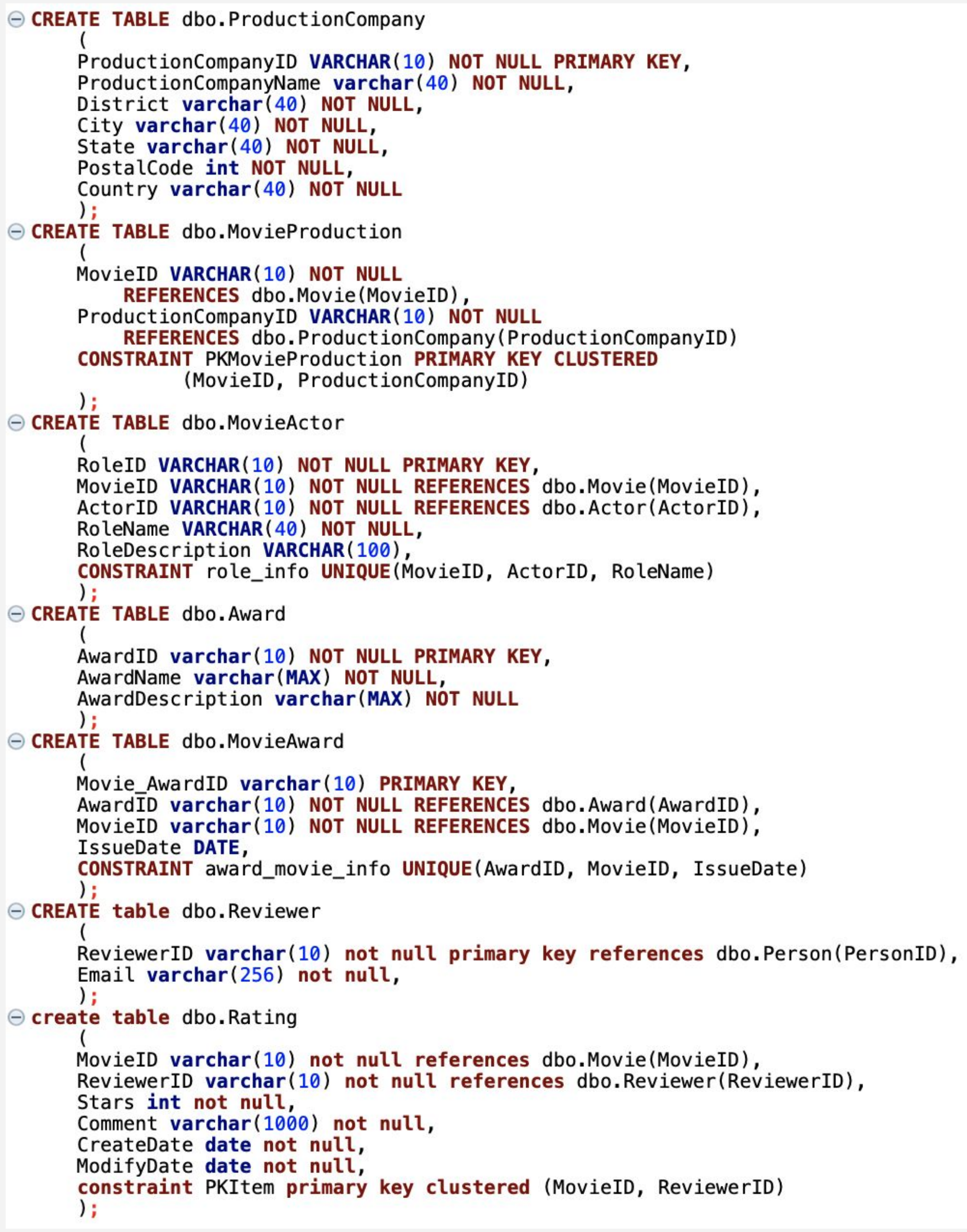
Reviwer	
ReviewerID	PK FK(PersonID)
Email	

Rating	
MovieID	PK FK
ReviewerID	PK FK
Stars	
Comment	
CreateDate	
ModifyDate	

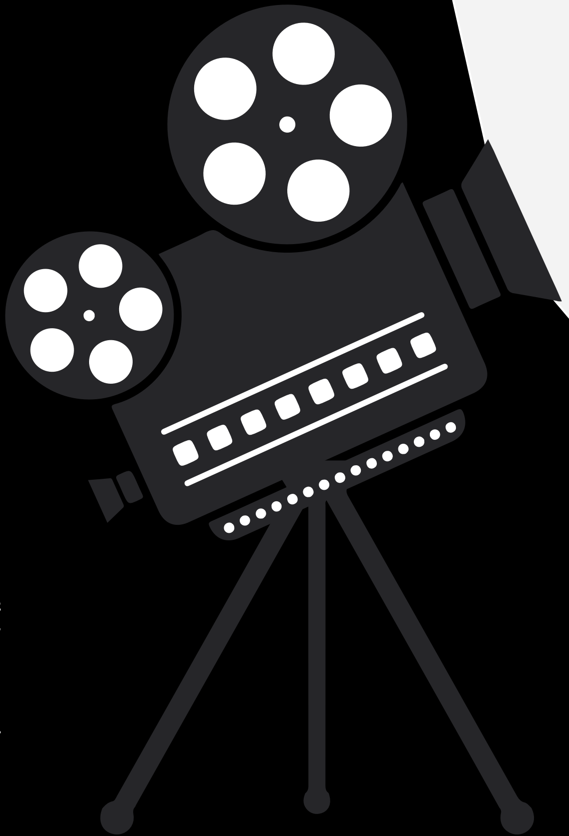
SQL DDL



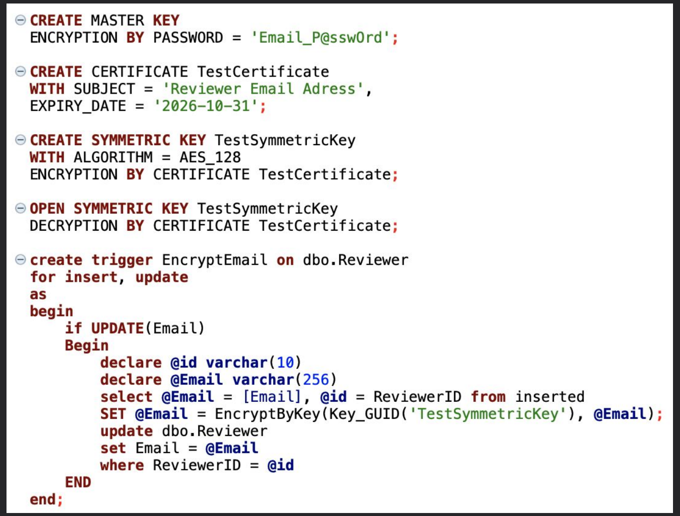
```
CREATE TABLE dbo.Person
        (
        PersonID VARCHAR(10)  NOT NULL PRIMARY KEY,
        PersonFirstName varchar(40),
        PersonLastName varchar(40),
        DateOfBirth date,
        Gender varchar(40),
        Nationality varchar(40)
        );
CREATE TABLE dbo.Director
        (
        DirectorID VARCHAR(10) NOT NULL PRIMARY KEY,
            FOREIGN KEY (DirectorID) REFERENCES dbo.Person(PersonID),
        DirectorBio varchar(100)
        );
CREATE TABLE dbo.Actor
        (
         ActorID VARCHAR(10) NOT NULL PRIMARY KEY,
            FOREIGN KEY (ActorID) REFERENCES dbo.Person(PersonID),
         ActorBIO VARCHAR(100)
        );
CREATE TABLE dbo.Movie
        (
        MovieID VARCHAR(10) NOT NULL PRIMARY KEY,
        DirectorID VARCHAR(10)
            REFERENCES dbo.Director(DirectorID),
        MovieTitle varchar(40) NOT NULL,
        ReleaseDate date,
        RunningTime time,
        Language varchar(40) NOT NULL,
        Country varchar(40) NOT NULL
        );
CREATE TABLE dbo.Genre
        (
        GenreID VARCHAR(10) NOT NULL PRIMARY KEY,
        GenreName varchar(40)
        );
CREATE TABLE dbo.MovieGenre
        (
        MovieID VARCHAR(10) NOT NULL
            REFERENCES dbo.Movie(MovieID),
        GenreID VARCHAR(10) NOT NULL
            REFERENCES dbo.Genre(GenreID)
        CONSTRAINT PKMovieGenre PRIMARY KEY CLUSTERED
                (MovieID, GenreID)
        );
```



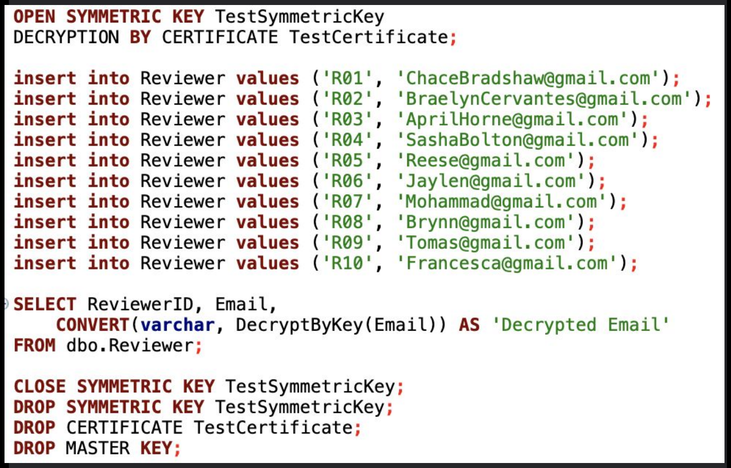
```
CREATE TABLE dbo.ProductionCompany
        (
        ProductionCompanyID VARCHAR(10) NOT NULL PRIMARY KEY,
        ProductionCompanyName varchar(40) NOT NULL,
        District varchar(40) NOT NULL,
        City varchar(40) NOT NULL,
        State varchar(40) NOT NULL,
        PostalCode int NOT NULL,
        Country varchar(40) NOT NULL
        );
CREATE TABLE dbo.MovieProduction
        (
        MovieID VARCHAR(10) NOT NULL
            REFERENCES dbo.Movie(MovieID),
        ProductionCompanyID VARCHAR(10) NOT NULL
            REFERENCES dbo.ProductionCompany(ProductionCompanyID)
        CONSTRAINT PKMovieProduction PRIMARY KEY CLUSTERED
                (MovieID, ProductionCompanyID)
        );
CREATE TABLE dbo.MovieActor
        (
        RoleID VARCHAR(10) NOT NULL PRIMARY KEY,
        MovieID VARCHAR(10) NOT NULL REFERENCES dbo.Movie(MovieID),
        ActorID VARCHAR(10) NOT NULL REFERENCES dbo.Actor(ActorID),
        RoleName VARCHAR(40) NOT NULL,
        RoleDescription VARCHAR(100),
        CONSTRAINT role_info UNIQUE(MovieID, ActorID, RoleName)
        );
CREATE TABLE dbo.Award
        (
        AwardID varchar(10) NOT NULL PRIMARY KEY,
        AwardName varchar(MAX) NOT NULL,
        AwardDescription varchar(MAX) NOT NULL
        );
CREATE TABLE dbo.MovieAward
        (
        Movie_AwardID varchar(10) PRIMARY KEY,
        AwardID varchar(10) NOT NULL REFERENCES dbo.Award(AwardID),
        MovieID varchar(10) NOT NULL REFERENCES dbo.Movie(MovieID),
        IssueDate DATE,
        CONSTRAINT award_movie_info UNIQUE(AwardID, MovieID, IssueDate)
        );
CREATE table dbo.Reviewer
        (
        ReviewerID varchar(10) not null primary key references dbo.Person(PersonID),
        Email varchar(256) not null,
        );
create table dbo.Rating
        (
        MovieID varchar(10) not null references dbo.Movie(MovieID),
        ReviewerID varchar(10) not null references dbo.Reviewer(ReviewerID),
        Stars int not null,
        Comment varchar(1000) not null,
        CreateDate date not null,
        ModifyDate date not null,
        constraint PKItem primary key clustered (MovieID, ReviewerID)
        );
```



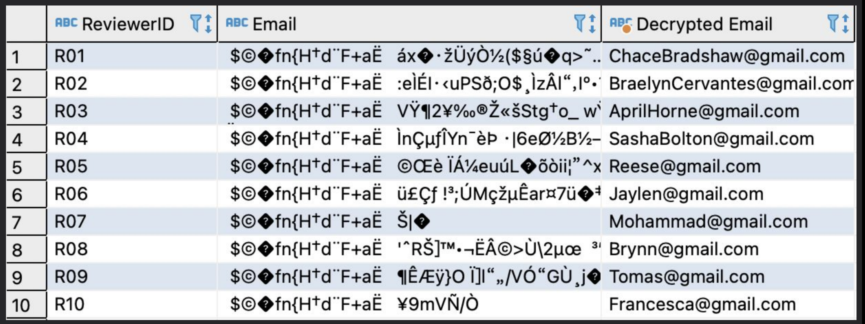
Encrypt Reviewer Email Using Trigger



```sql
CREATE MASTER KEY
ENCRYPTION BY PASSWORD = 'Email_P@ssw0rd';

CREATE CERTIFICATE TestCertificate
WITH SUBJECT = 'Reviewer Email Adress',
EXPIRY_DATE = '2026-10-31';

CREATE SYMMETRIC KEY TestSymmetricKey
WITH ALGORITHM = AES_128
ENCRYPTION BY CERTIFICATE TestCertificate;

OPEN SYMMETRIC KEY TestSymmetricKey
DECRYPTION BY CERTIFICATE TestCertificate;

create trigger EncryptEmail on dbo.Reviewer
for insert, update
as
begin
    if UPDATE(Email)
    Begin
        declare @id varchar(10)
        declare @Email varchar(256)
        select @Email = [Email], @id = ReviewerID from inserted
        SET @Email = EncryptByKey(Key_GUID('TestSymmetricKey'), @Email);
        update dbo.Reviewer
        set Email = @Email
        where ReviewerID = @id
    END
end;
```

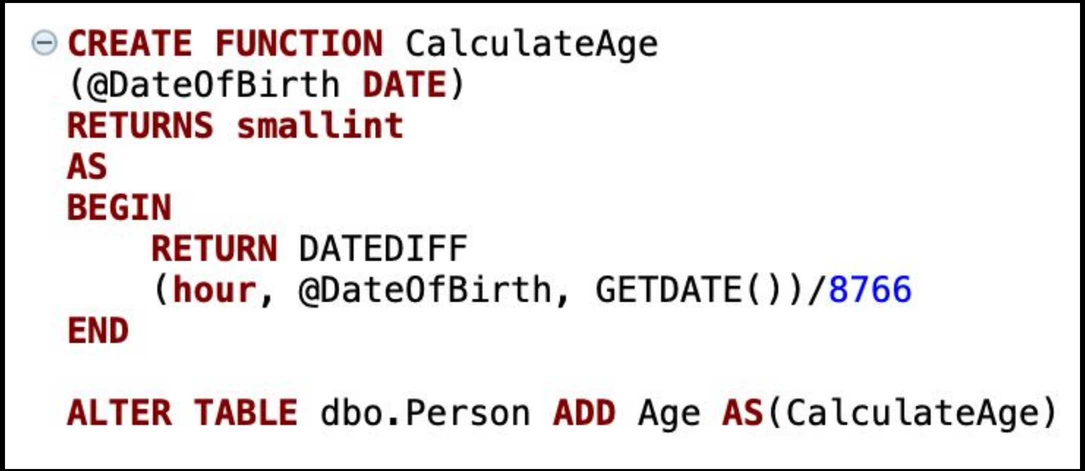


```sql
OPEN SYMMETRIC KEY TestSymmetricKey
DECRYPTION BY CERTIFICATE TestCertificate;

insert into Reviewer values ('R01', 'ChaceBradshaw@gmail.com');
insert into Reviewer values ('R02', 'BraelynCervantes@gmail.com');
insert into Reviewer values ('R03', 'AprilHorne@gmail.com');
insert into Reviewer values ('R04', 'SashaBolton@gmail.com');
insert into Reviewer values ('R05', 'Reese@gmail.com');
insert into Reviewer values ('R06', 'Jaylen@gmail.com');
insert into Reviewer values ('R07', 'Mohammad@gmail.com');
insert into Reviewer values ('R08', 'Brynn@gmail.com');
insert into Reviewer values ('R09', 'Tomas@gmail.com');
insert into Reviewer values ('R10', 'Francesca@gmail.com');

SELECT ReviewerID, Email,
    CONVERT(varchar, DecryptByKey(Email)) AS 'Decrypted Email'
FROM dbo.Reviewer;

CLOSE SYMMETRIC KEY TestSymmetricKey;
DROP SYMMETRIC KEY TestSymmetricKey;
DROP CERTIFICATE TestCertificate;
DROP MASTER KEY;
```



	ReviewerID	Email	Decrypted Email
1	R01	$©�fn{H†d¨F+aË áx�·žÜýÒ½($§ú�q>˜..	ChaceBradshaw@gmail.com
2	R02	$©�fn{H†d¨F+aË :eÌÉl·‹uPSð;O$¸ÌzÂl“,l°·	BraelynCervantes@gmail.com
3	R03	$©�fn{H†d¨F+aË VŸ¶2¥‰®Ž«šStg†o_ w	AprilHorne@gmail.com
4	R04	$©�fn{H†d¨F+aË ÌnÇµƒÎYn¯èÞ ·|6eØ½B½–	SashaBolton@gmail.com
5	R05	$©�fn{H†d¨F+aË ©Œè ÏÁ¼euúL�õòii¦”^x	Reese@gmail.com
6	R06	$©�fn{H†d¨F+aË ü£Çƒ !³;ÚMçžµÊar¤7ü�	Jaylen@gmail.com
7	R07	$©�fn{H†d¨F+aË Š|�	Mohammad@gmail.com
8	R08	$©�fn{H†d¨F+aË '^RŠ]™•¬ËÂ©>Ù\2µœ ³	Brynn@gmail.com
9	R09	$©�fn{H†d¨F+aË ¶ÊÆÿ}O Ï]l“„/VÓ“GÙ¸j�	Tomas@gmail.com
10	R10	$©�fn{H†d¨F+aË ¥9mVÑ/Ò	Francesca@gmail.com

Function for Calculating Age

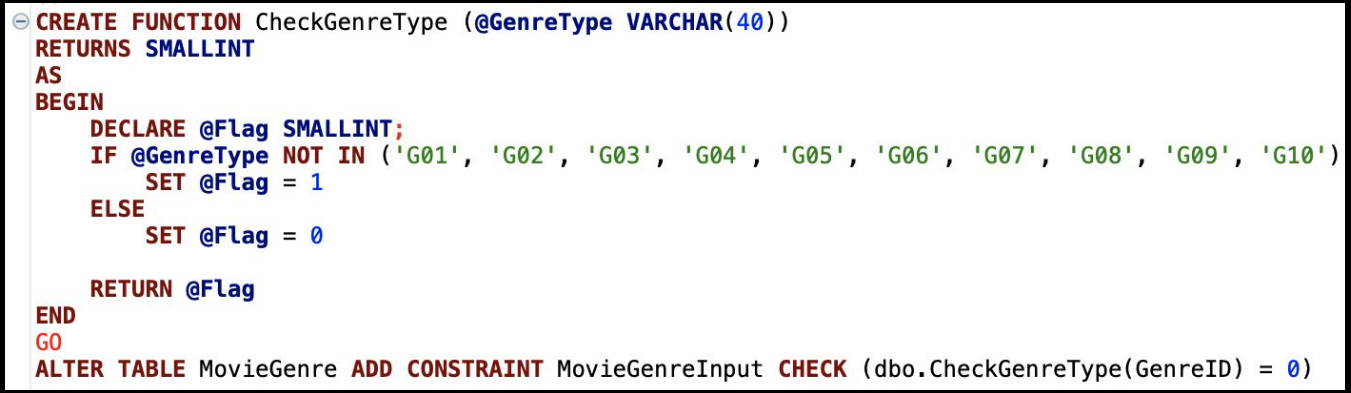


```
CREATE FUNCTION CalculateAge
(@DateOfBirth DATE)
RETURNS smallint
AS
BEGIN
    RETURN DATEDIFF
    (hour, @DateOfBirth, GETDATE())/8766
END

ALTER TABLE dbo.Person ADD Age AS(CalculateAge)
```

Calculate Person’s Age by Date of Birth

Alter table add column

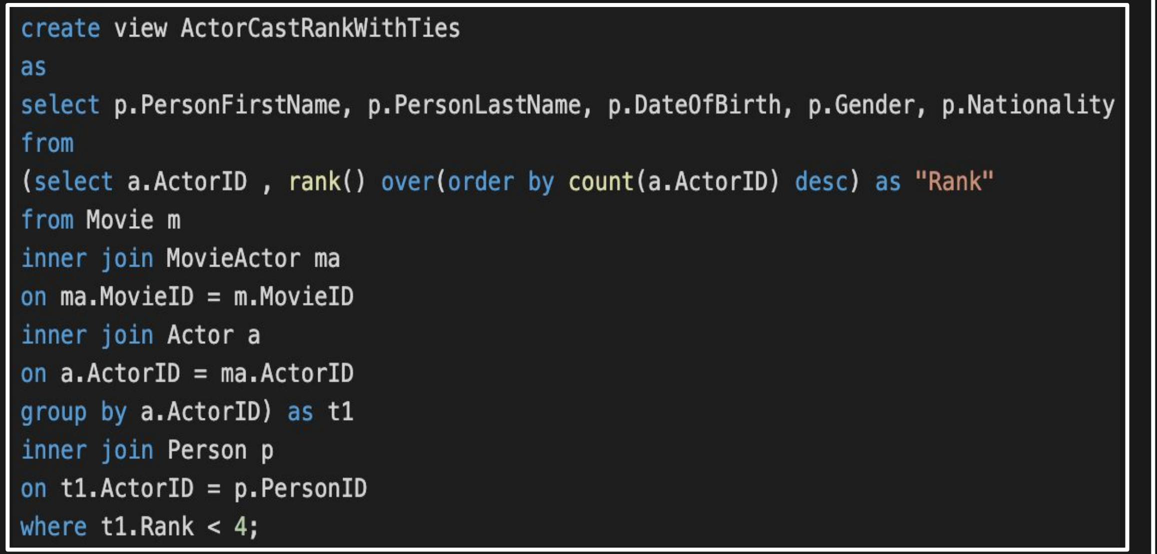
Function For Check Inserted Genre Type



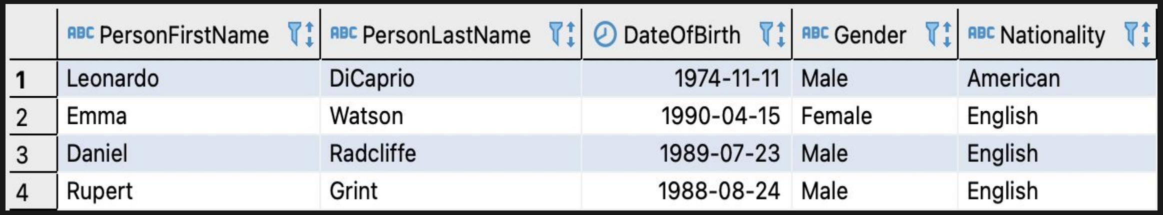
```
CREATE FUNCTION CheckGenreType (@GenreType VARCHAR(40))
RETURNS SMALLINT
AS
BEGIN
    DECLARE @Flag SMALLINT;
    IF @GenreType NOT IN ('G01', 'G02', 'G03', 'G04', 'G05', 'G06', 'G07', 'G08', 'G09', 'G10')
        SET @Flag = 1
    ELSE
        SET @Flag = 0

    RETURN @Flag
END
GO
ALTER TABLE MovieGenre ADD CONSTRAINT MovieGenreInput CHECK (dbo.CheckGenreType(GenreID) = 0)
```

Table-level constraint for checking movie genre type

Views For the TOP 3 Actors in the Number of Movies He/She Has Cast In

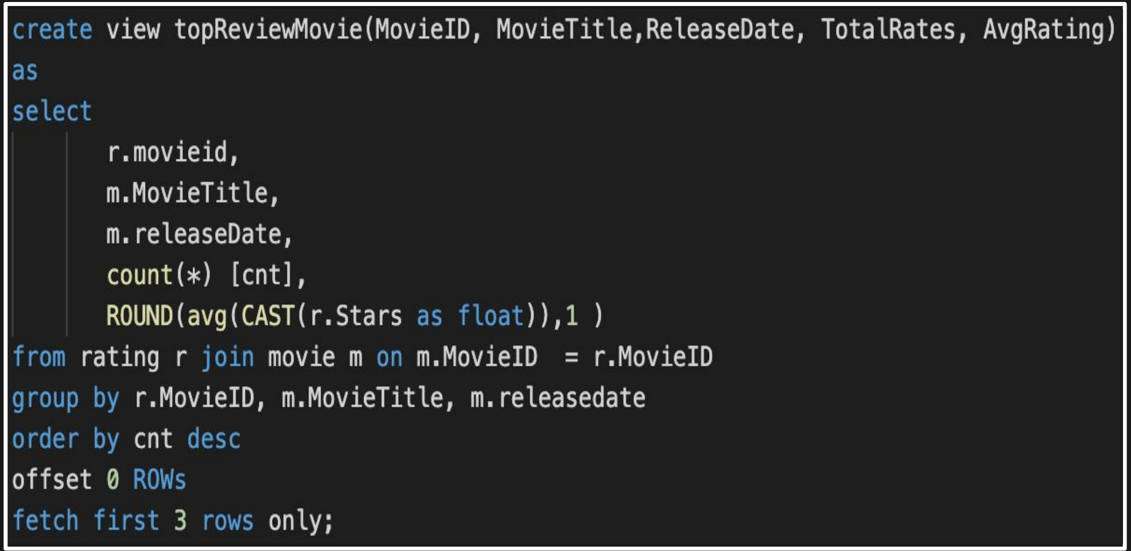


```sql
create view ActorCastRankWithTies
as
select p.PersonFirstName, p.PersonLastName, p.DateOfBirth, p.Gender, p.Nationality
from
(select a.ActorID , rank() over(order by count(a.ActorID) desc) as "Rank"
from Movie m
inner join MovieActor ma
on ma.MovieID = m.MovieID
inner join Actor a
on a.ActorID = ma.ActorID
group by a.ActorID) as t1
inner join Person p
on t1.ActorID = p.PersonID
where t1.Rank < 4;
```

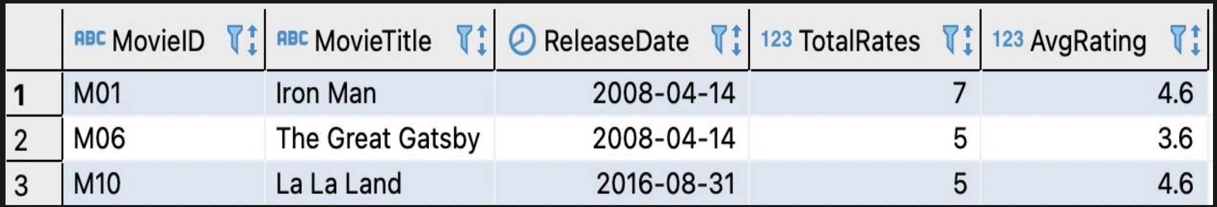


	PersonFirstName	PersonLastName	DateOfBirth	Gender	Nationality
1	Leonardo	DiCaprio	1974-11-11	Male	American
2	Emma	Watson	1990-04-15	Female	English
3	Daniel	Radcliffe	1989-07-23	Male	English
4	Rupert	Grint	1988-08-24	Male	English

Views For the TOP 3 Review Movies

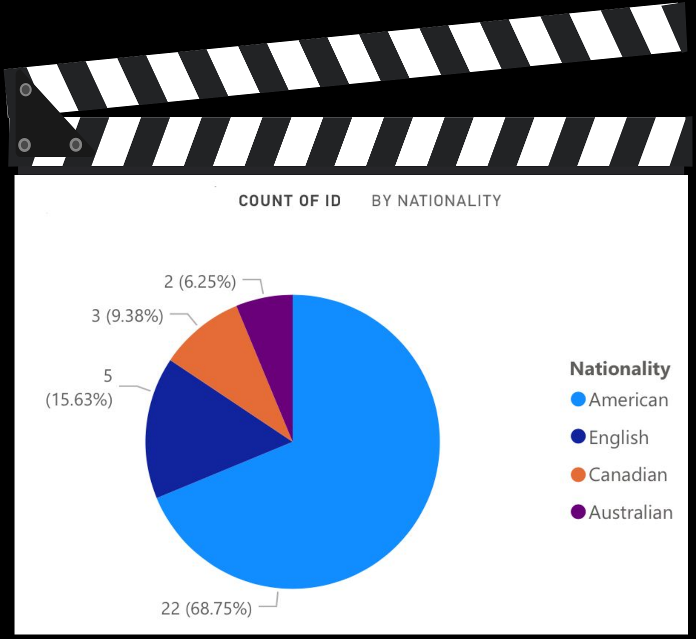


```
create view topReviewMovie(MovieID, MovieTitle,ReleaseDate, TotalRates, AvgRating)
as
select
        r.movieid,
        m.MovieTitle,
        m.releaseDate,
        count(*) [cnt],
        ROUND(avg(CAST(r.Stars as float)),1 )
from rating r join movie m on m.MovieID  = r.MovieID
group by r.MovieID, m.MovieTitle, m.releasedate
order by cnt desc
offset 0 ROWs
fetch first 3 rows only;
```

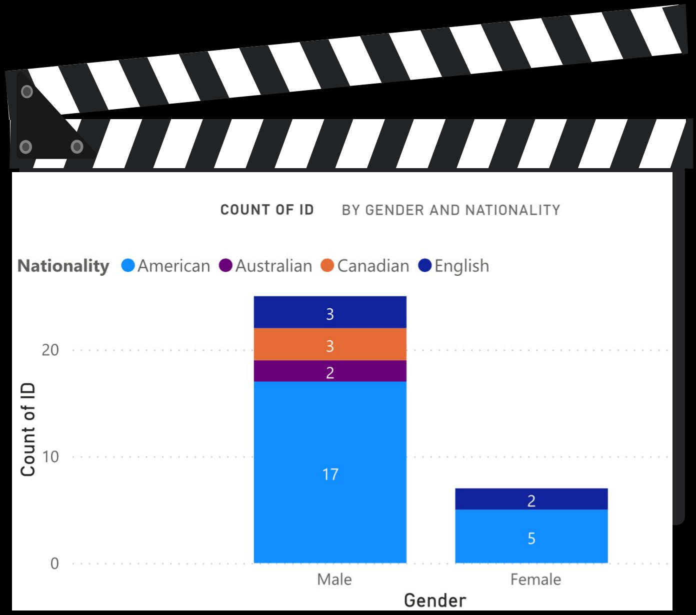


	MovieID	MovieTitle	ReleaseDate	TotalRates	AvgRating
1	M01	Iron Man	2008-04-14	7	4.6
2	M06	The Great Gatsby	2008-04-14	5	3.6
3	M10	La La Land	2016-08-31	5	4.6

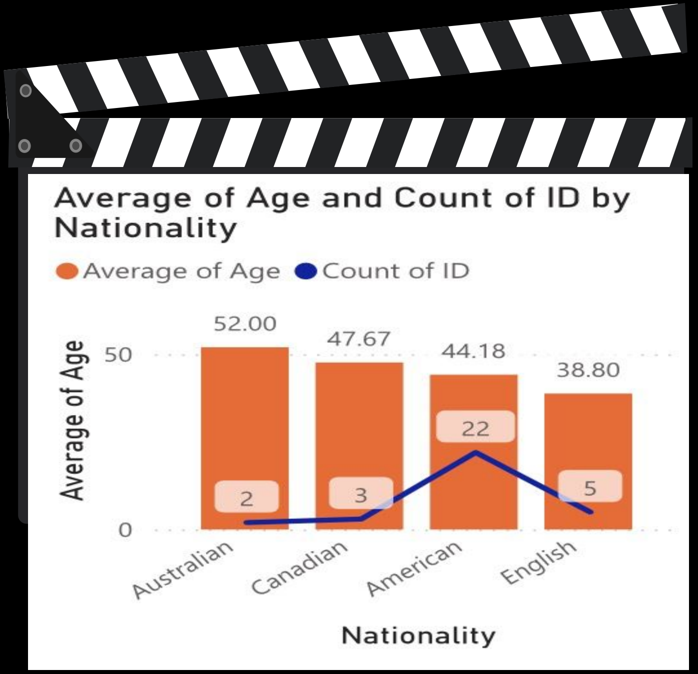
Data Analysis with Power BI



Nationality



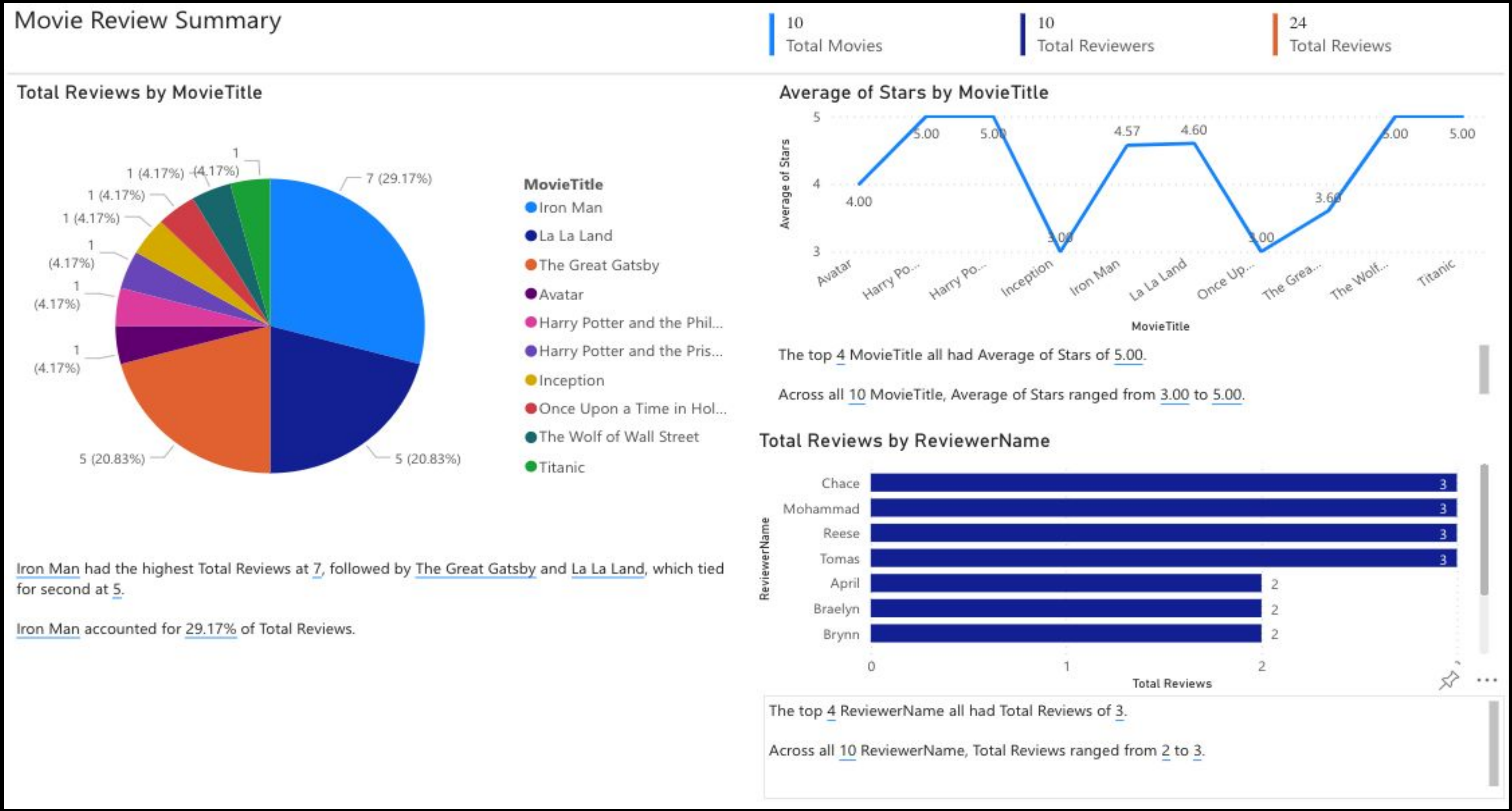
Gender & Nationality



Age & Nationality

Analyze movie rating and reviewers

https://app.powerbi.com/links/imeYarzzLq?ctid=a8eec281-aaa3-4dae-ac9b-9a398b9215e7&pbi_source=linkShare



Movie Review Summary

10 Total Movies

10 Total Reviewers

24 Total Reviews

Total Reviews by MovieTitle

Iron Man had the highest Total Reviews at 7, followed by The Great Gatsby and La La Land, which tied for second at 5.

Iron Man accounted for 29.17% of Total Reviews.

Average of Stars by MovieTitle

The top 4 MovieTitle all had Average of Stars of 5.00.

Across all 10 MovieTitle, Average of Stars ranged from 3.00 to 5.00.

Total Reviews by ReviewerName

The top 4 ReviewerName all had Total Reviews of 3.

Across all 10 ReviewerName, Total Reviews ranged from 2 to 3.

Thank you!

Q & A