*Project 1*

***Submitted by***

*SANSKRITI CHOUDHARY*

*Company:* ***Netflix***

## NETFLIX – Project Overview

Netflix was first founded in August of 1997 by two serial entrepreneurs, Marc Randolph and Reed Hastings. The company began out in Scotts Valley, California, and has grown to become one of the world's leading internet entertainment platforms.

In the beginning,  Netflix was purely a movie rental service. Users ordered movies on the Netflix website, and received DVDs in the post. When they were finished with them, they would simply post them back to Netflix in the envelopes provided.

At the time, this was seen as a boon to those who did not have a video rental store nearby. (remember those blissful late 90s and early 2000?).

Today, Netflix streams movies and has more than **208 million** paid subscribers in over **190 countries** around the world. It offers a wide range of TV series, documentaries, and feature films across a wide variety of genres and languages, including original productions.

In this Project, we are going to develop a database from scratch. That means, we would be taking the business rules as per "NETFLIX" and creating an entire schema explaining the connections of different entities, modules in the backend of NETFLIX webpage, and how they interact with each other.

## User Requirements

Phill Dunphy got a new OLED TV and a DOLBY ATMOS home theater for his luxurious $4mn house. Now, he wants to watch "Formula 1: Drive to Survive", which is available exclusively on NETFLIX.

Phill decides to watch the show on Netflix, for which he needs to buy a subscription, which would be a transaction between Phill and NETFLIX.

This is where we start noting down the user requirements and build a database which captures transactional and analytical data. Where, our end user Not only focuses on CRUD- Creating, Reading, Updating & Deleting the records, but is also interested in analysis, reporting, forecasting, etc.

Such databases have a smaller number of inserts and updates, as the end goal is to analyze the data as fast as possible.

After discussing, and having good brainstorming sessions with the team, we came up with the following Business Rules for our Database.

- We can classify content into Movies & TV Shows.
- There can be other types of content like trailers, documentaries etc.
- Content can be either Movies or TV shows but not both.
- Every content has ContentID which uniquely identifies the content, content title, Release date, ratings. TV Shows have the attribute number of seasons and movies have attribute duration.

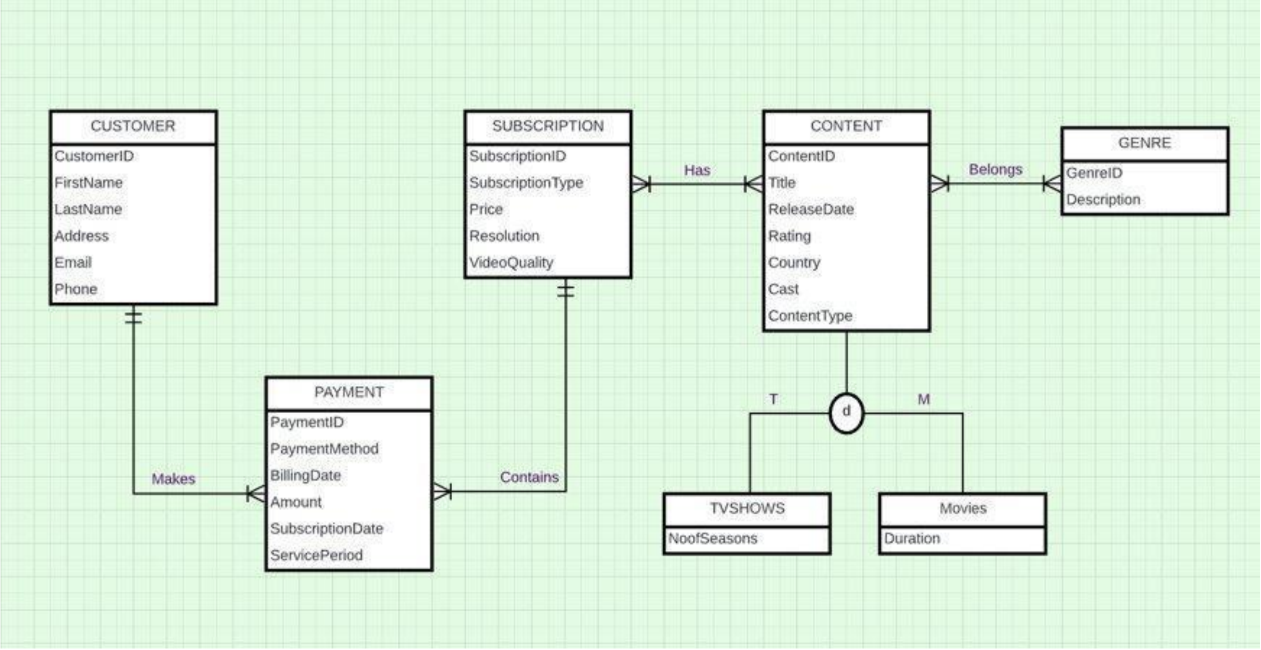
- To view the content, a customer must buy a subscription by making a payment. Customers can be identified by a unique identifier CustomerID.
- Other attributes include Name, which is a combination of first and last name, Address, Email, Phone Number.
- Subscription has a unique identifier SubscriptionID. Other attributes are Subscription type, Price and Resolution.
- Genre has attribute GenreID and Description.
- Payment is made by the customer to buy a subscription.
- It is uniquely identified by PaymentID, and other attributes are Payment Method, Billing Date and Amount Paid.
- A Customer must make at least one payment and a payment must have one and only one customer.
- A subscription can have many customers and a customer should have at least one subscription.
- A content must have at least one or many genres, and a genre should belong to one or many contents.
- A subscription may have at least one or many contents and a content must be part of at least one subscription.
- A content involves one or many casts, and a cast can be a part of one or many contents.

# Conceptual and Logical Database Designing

## E-R Model:

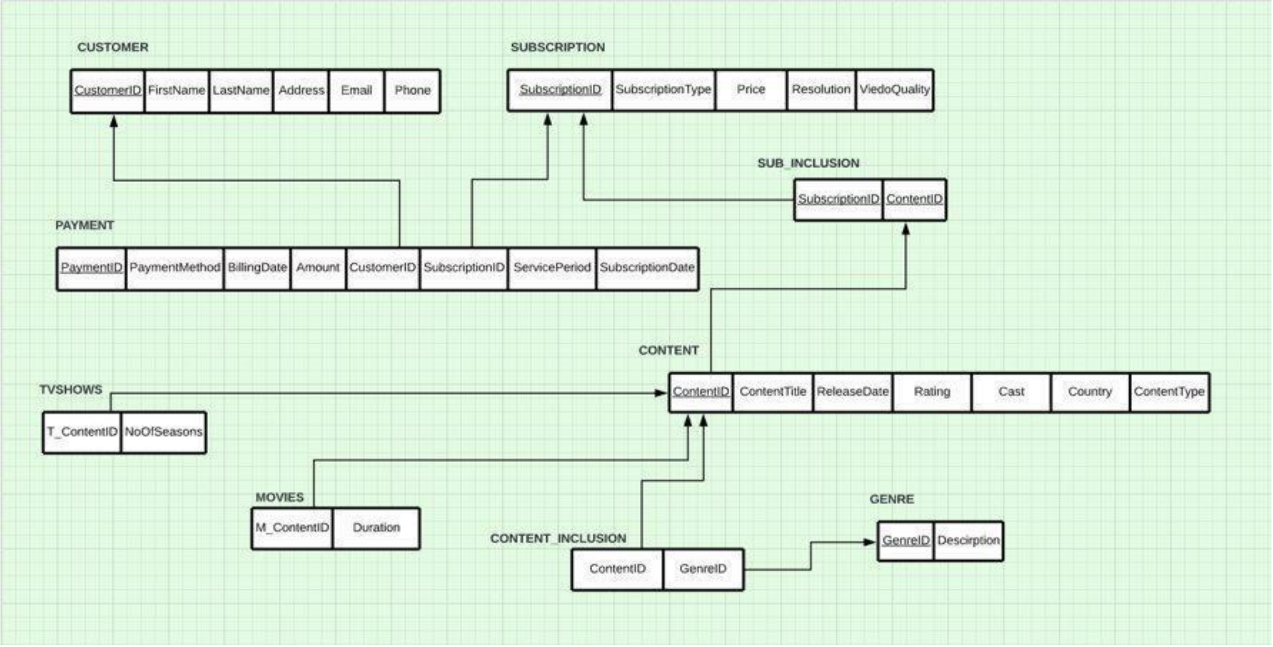
This is the Entity-Relationship Model. It is developed using the business rules provided by the customer for its development.

- ➢ Content has a ‘subtype discriminator’ <u>Content Type</u> which identifies if content belongs to movies or TV shows for each record in content table.
- ➢ Content table has partial specialization because a content can belong movies, TV shows and others such as trailers.
- ➢ Content table has disjoint constraint as a content can only belong to either movie or TV show at a time but not both.



## Relational Model:

The ER model is normalized to its 3rd Normal Form and thus the Relational Model is created. This form helps in database table creation.



## Module Description

MODULEWISE DESCRIPTION

This project divided into four modules. As stated earlier, the database maintains information about the customers making a transaction with Netflix, transaction made for a subscription type and there are different contents available in subscription type.

MODULE-1:

Module-1 deals with the Customer information. A customer has to enter his basic information to buy a Netflix subscription. This information entered is stored in the customer table.

MODULE-2:

Module-2 deals with payments made by the customer to buy a Netflix subscription.

MODULE-3:

Module-3 deals with the type of subscription that Netflix is offering for a particular cost. Each subscription has different content ids, screen resolution and number of users.

MODULE-4:

Module 4 deals content. Content is divided into TV shows; Movies and these TV shows and movies are further classified into different genres.

## Implementation

Implementation accounts for creation of database. The database is created with reference to the entity-relationship model and relational model. Post normalization, the database consists of 9 tables. The creation of tables is explained as below.

**Creating the Database:**

```
CREATE DATABASE NETFLIX ;
```

- Using the Netflix Database for further database development:

```
USE NETFLIX ;
```

- Customer Table Creation:

```
CREATE TABLE Customer_T (
CustomerId int(10) NOT NULL AUTO_INCREMENT,
FirstName VARCHAR(50),
LastName VARCHAR(50),
Address VARCHAR(256),
Email VARCHAR(50),
Phone VARCHAR(20),
CONSTRAINT customer_pk PRIMARY KEY (CustomerId))
ENGINE = INNODB;
```

- Content Table Creation:

```
CREATE TABLE Content_T (
ContentId VARCHAR(10) NOT NULL,
ContentTitle VARCHAR(50),
ReleaseDate DATE,
Rating VARCHAR(10),
Country VARCHAR(256),
Cast VARCHAR(256),
ContentType VARCHAR(10),
CONSTRAINT content_pk PRIMARY KEY (ContentID))
ENGINE = INNODB;
```

- Subscription Table Creation:

```
CREATE TABLE Subscription_T (
SubscriptionId INT(2) NOT NULL,
SubscriptionType VARCHAR(10),
Price DOUBLE(10,2),
Resolution VARCHAR(10),
VideoQuality VARCHAR(10),
CONSTRAINT subscription_pk PRIMARY KEY (SubscriptionId))
ENGINE = INNODB;
```

- Genre Table Creation:

```
CREATE TABLE Genre_T (
GenreId INT(10) NOT NULL AUTO_INCREMENT,
GenreDescription VARCHAR(256),
CONSTRAINT genre_pk PRIMARY KEY (GenreId))
ENGINE = INNODB;
```

- Content-Genre Table Creation:

```
CREATE TABLE ContentGenre_T (
ContentId VARCHAR(10),
GenreId INT(10),
CONSTRAINT contentgenre_pk PRIMARY KEY (ContentId,GenreId),
CONSTRAINT contentgenre_fk FOREIGN KEY (ContentId) REFERENCES Content_T(ContentId) ON
UPDATE CASCADE ON DELETE CASCADE,
CONSTRAINT contentgenre_fk1 FOREIGN KEY (GenreId) REFERENCES Genre_T(GenreId) ON
UPDATE CASCADE ON DELETE CASCADE)
ENGINE = INNODB;
```

- Content-Subscription Table Creation:

```
CREATE TABLE ContentSubscription_T (
ContentId VARCHAR(10),
SubscriptionId INT(2),
CONSTRAINT contentsubscription_pk PRIMARY KEY (ContentId,SubscriptionId),
CONSTRAINT contentsubscription_fk FOREIGN KEY (ContentId) REFERENCES
Content_T(ContentId) ON UPDATE CASCADE ON DELETE CASCADE,
CONSTRAINT contentsubscription_fk1 FOREIGN KEY (SubscriptionId) REFERENCES
Subscription_T(SubscriptionId) ON UPDATE CASCADE ON DELETE CASCADE)
ENGINE = INNODB;
```

- TV Shows Table Creation:

```
CREATE TABLE TvShows_T (
ContentId VARCHAR(10),
Seasons VARCHAR(20),
CONSTRAINT tvshows_pk PRIMARY KEY (ContentId),
CONSTRAINT tvshows_fk FOREIGN KEY (ContentId) REFERENCES Content_T (ContentId) ON
UPDATE CASCADE ON DELETE CASCADE)
ENGINE = INNODB;
```

- Movies Table Creation:

```
CREATE TABLE Movies_T (
ContentId VARCHAR(10),
Duration VARCHAR(20),
CONSTRAINT movies_pk PRIMARY KEY (ContentId),
CONSTRAINT movies_fk FOREIGN KEY (ContentId) REFERENCES Content_T (ContentId) ON
UPDATE CASCADE ON DELETE CASCADE)
ENGINE = INNODB;
```

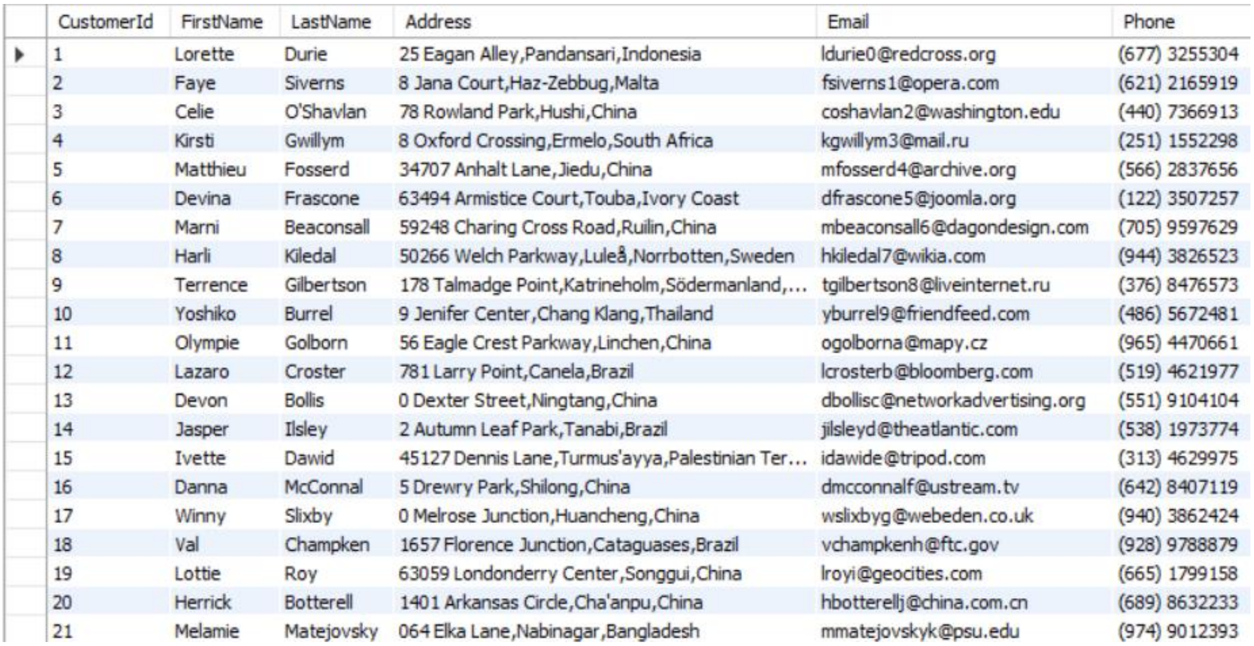
- Payment Table Creation:

```
CREATE TABLE Payment_T (
PaymentId VARCHAR(10) NOT NULL,
PaymentMethod VARCHAR(20),
SubscriptionDate Date,
BillingDate Date,
ServicePeriod VARCHAR(30),
Amount Double(10,2),
CustomerId INT(10),
SubscriptionId INT(2),
CONSTRAINT payment_pk PRIMARY KEY (PaymentId),
CONSTRAINT payment_fk FOREIGN KEY (CustomerId) REFERENCES customer_t(CustomerId) ON
UPDATE CASCADE ON DELETE CASCADE,
CONSTRAINT payment_fk1 FOREIGN KEY (SubscriptionId) REFERENCES
subscription_t(SubscriptionId) ON UPDATE CASCADE ON DELETE CASCADE)
ENGINE = INNODB;
```

## Testing

Testing for Netflix database has been carried out in the form of viewing tables and their creation along with the queries and their outputs. Few of them are listed below.

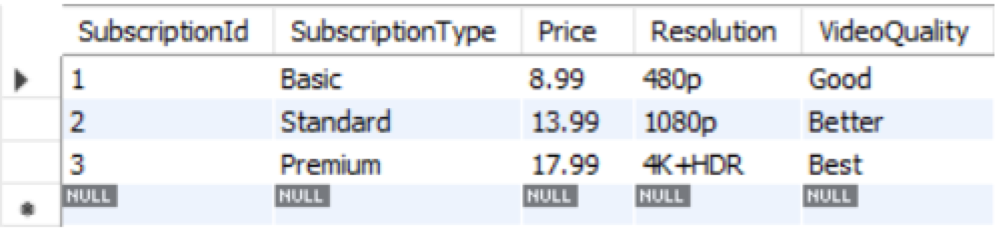
1. Customer Table:



| CustomerId | FirstName | LastName | Address | Email | Phone |
|---|---|---|---|---|---|
| 1 | Lorette | Durie | 25 Eagan Alley,Pandansari,Indonesia | ldurie0@redcross.org | (677) 3255304 |
| 2 | Faye | Siverns | 8 Jana Court,Haz-Zebbug,Malta | fsiverns1@opera.com | (621) 2165919 |
| 3 | Celie | O'Shavlan | 78 Rowland Park,Hushi,China | coshavlan2@washington.edu | (440) 7366913 |
| 4 | Kirsti | Gwillym | 8 Oxford Crossing,Ermelo,South Africa | kgwillym3@mail.ru | (251) 1552298 |
| 5 | Matthieu | Fosserd | 34707 Anhalt Lane,Jiedu,China | mfosserd4@archive.org | (566) 2837656 |
| 6 | Devina | Frascone | 63494 Armistice Court,Touba,Ivory Coast | dfrascone5@joomla.org | (122) 3507257 |
| 7 | Marni | Beaconsall | 59248 Charing Cross Road,Ruilin,China | mbeaconsall6@dagondesign.com | (705) 9597629 |
| 8 | Harli | Kiledal | 50266 Welch Parkway,Luleå,Norrbotten,Sweden | hkiledal7@wikia.com | (944) 3826523 |
| 9 | Terrence | Gilbertson | 178 Talmadge Point,Katrineholm,Södermanland,... | tgilbertson8@liveinternet.ru | (376) 8476573 |
| 10 | Yoshiko | Burrel | 9 Jenifer Center,Chang Klang,Thailand | yburrel9@friendfeed.com | (486) 5672481 |
| 11 | Olympie | Golborn | 56 Eagle Crest Parkway,Linchen,China | ogolborna@mapy.cz | (965) 4470661 |
| 12 | Lazaro | Croster | 781 Larry Point,Canela,Brazil | lcrosterb@bloomberg.com | (519) 4621977 |
| 13 | Devon | Bollis | 0 Dexter Street,Ningtang,China | dbollisc@networkadvertising.org | (551) 9104104 |
| 14 | Jasper | Ilsley | 2 Autumn Leaf Park,Tanabi,Brazil | jilsleyd@theatlantic.com | (538) 1973774 |
| 15 | Ivette | Dawid | 45127 Dennis Lane,Turmus'ayya,Palestinian Ter... | idawide@tripod.com | (313) 4629975 |
| 16 | Danna | McConnal | 5 Drewry Park,Shilong,China | dmcconnalf@ustream.tv | (642) 8407119 |
| 17 | Winny | Slixby | 0 Melrose Junction,Huancheng,China | wslixbyg@webeden.co.uk | (940) 3862424 |
| 18 | Val | Champken | 1657 Florence Junction,Cataguases,Brazil | vchampkenh@ftc.gov | (928) 9788879 |
| 19 | Lottie | Roy | 63059 Londonderry Center,Songgui,China | lroyi@geocities.com | (665) 1799158 |
| 20 | Herrick | Botterell | 1401 Arkansas Circle,Cha'anpu,China | hbotterellj@china.com.cn | (689) 8632233 |
| 21 | Melamie | Matejovsky | 064 Elka Lane,Nabinagar,Bangladesh | mmatejovskyk@psu.edu | (974) 9012393 |

Fig1. Customer Table

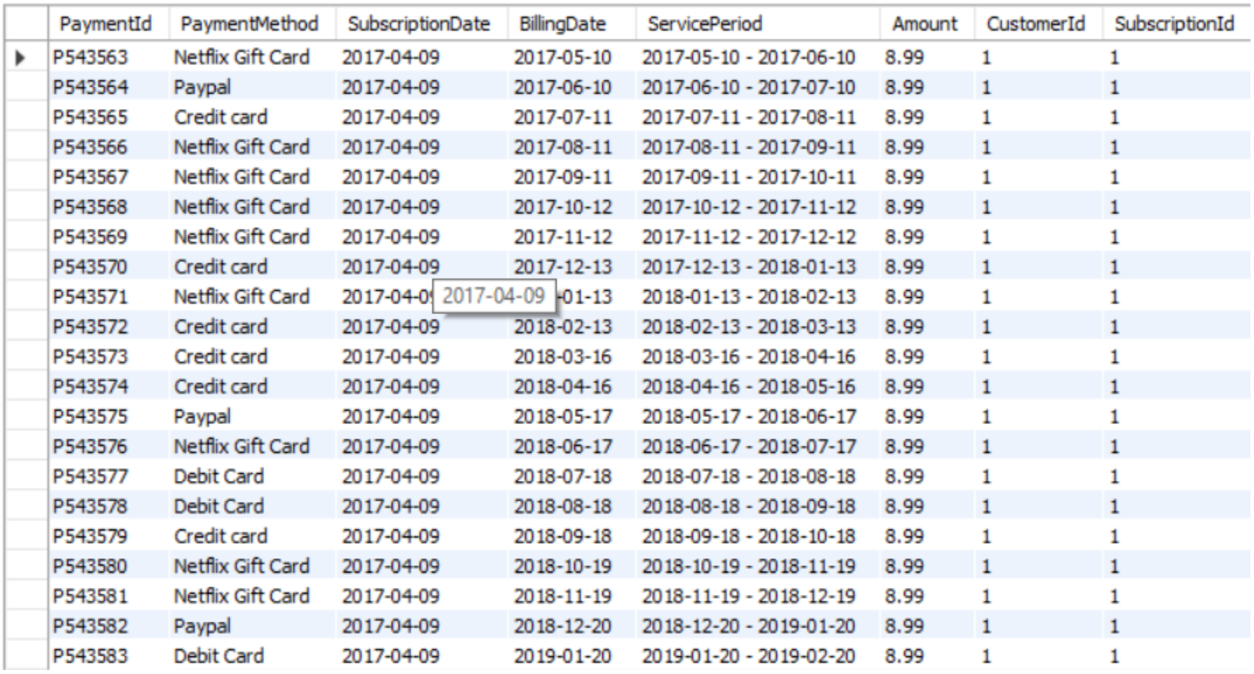
2. Subscription Table:



| SubscriptionId | SubscriptionType | Price | Resolution | VideoQuality |
|---|---|---|---|---|
| 1 | Basic | 8.99 | 480p | Good |
| 2 | Standard | 13.99 | 1080p | Better |
| 3 | Premium | 17.99 | 4K+HDR | Best |
| NULL | NULL | NULL | NULL | NULL |

Fig2. Subscription Table

### 3. Payment Table:

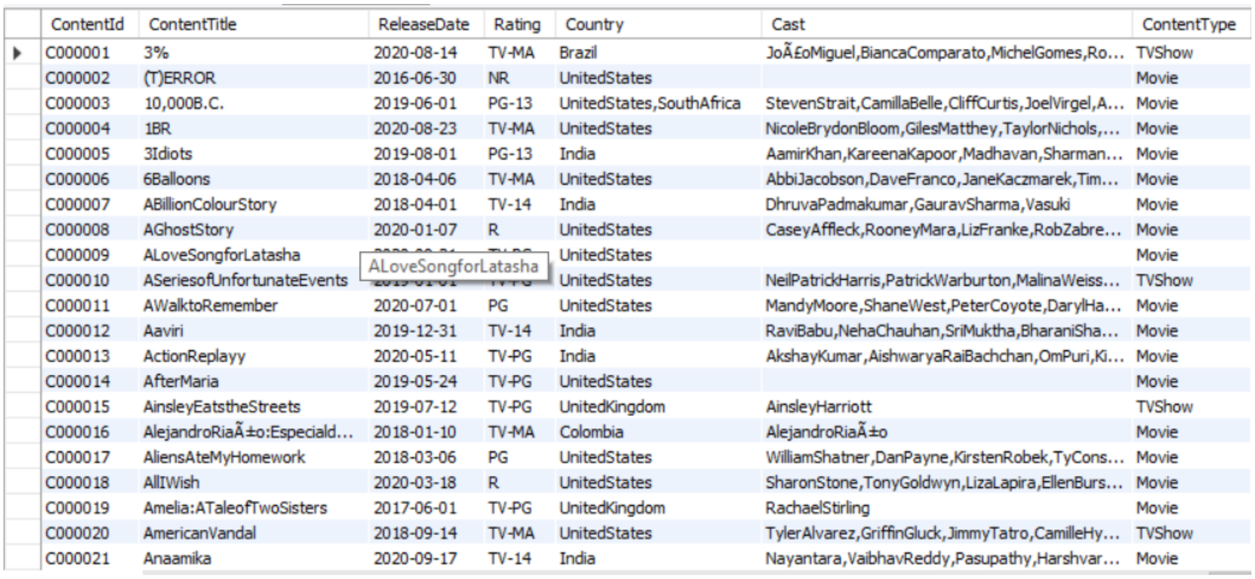


| PaymentId | PaymentMethod | SubscriptionDate | BillingDate | ServicePeriod | Amount | CustomerId | SubscriptionId |
|---|---|---|---|---|---|---|---|
| P543563 | Netflix Gift Card | 2017-04-09 | 2017-05-10 | 2017-05-10 - 2017-06-10 | 8.99 | 1 | 1 |
| P543564 | Paypal | 2017-04-09 | 2017-06-10 | 2017-06-10 - 2017-07-10 | 8.99 | 1 | 1 |
| P543565 | Credit card | 2017-04-09 | 2017-07-11 | 2017-07-11 - 2017-08-11 | 8.99 | 1 | 1 |
| P543566 | Netflix Gift Card | 2017-04-09 | 2017-08-11 | 2017-08-11 - 2017-09-11 | 8.99 | 1 | 1 |
| P543567 | Netflix Gift Card | 2017-04-09 | 2017-09-11 | 2017-09-11 - 2017-10-11 | 8.99 | 1 | 1 |
| P543568 | Netflix Gift Card | 2017-04-09 | 2017-10-12 | 2017-10-12 - 2017-11-12 | 8.99 | 1 | 1 |
| P543569 | Netflix Gift Card | 2017-04-09 | 2017-11-12 | 2017-11-12 - 2017-12-12 | 8.99 | 1 | 1 |
| P543570 | Credit card | 2017-04-09 | 2017-12-13 | 2017-12-13 - 2018-01-13 | 8.99 | 1 | 1 |
| P543571 | Netflix Gift Card | 2017-04-09 | 2018-01-13 | 2018-01-13 - 2018-02-13 | 8.99 | 1 | 1 |
| P543572 | Credit card | 2017-04-09 | 2018-02-13 | 2018-02-13 - 2018-03-13 | 8.99 | 1 | 1 |
| P543573 | Credit card | 2017-04-09 | 2018-03-16 | 2018-03-16 - 2018-04-16 | 8.99 | 1 | 1 |
| P543574 | Credit card | 2017-04-09 | 2018-04-16 | 2018-04-16 - 2018-05-16 | 8.99 | 1 | 1 |
| P543575 | Paypal | 2017-04-09 | 2018-05-17 | 2018-05-17 - 2018-06-17 | 8.99 | 1 | 1 |
| P543576 | Netflix Gift Card | 2017-04-09 | 2018-06-17 | 2018-06-17 - 2018-07-17 | 8.99 | 1 | 1 |
| P543577 | Debit Card | 2017-04-09 | 2018-07-18 | 2018-07-18 - 2018-08-18 | 8.99 | 1 | 1 |
| P543578 | Debit Card | 2017-04-09 | 2018-08-18 | 2018-08-18 - 2018-09-18 | 8.99 | 1 | 1 |
| P543579 | Credit card | 2017-04-09 | 2018-09-18 | 2018-09-18 - 2018-10-18 | 8.99 | 1 | 1 |
| P543580 | Netflix Gift Card | 2017-04-09 | 2018-10-19 | 2018-10-19 - 2018-11-19 | 8.99 | 1 | 1 |
| P543581 | Netflix Gift Card | 2017-04-09 | 2018-11-19 | 2018-11-19 - 2018-12-19 | 8.99 | 1 | 1 |
| P543582 | Paypal | 2017-04-09 | 2018-12-20 | 2018-12-20 - 2019-01-20 | 8.99 | 1 | 1 |
| P543583 | Debit Card | 2017-04-09 | 2019-01-20 | 2019-01-20 - 2019-02-20 | 8.99 | 1 | 1 |

2017-04-09

Fig3. Payment Table

### 4. Content Table:

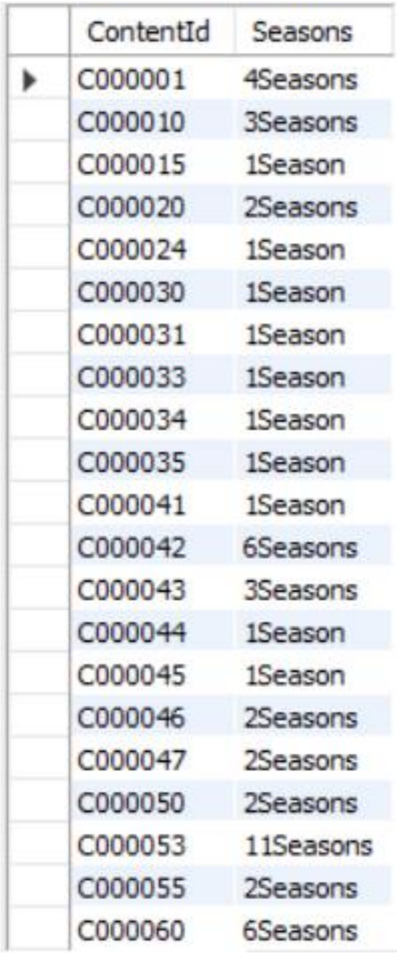


| ContentId | ContentTitle | ReleaseDate | Rating | Country | Cast | ContentType |
|---|---|---|---|---|---|---|
| C000001 | 3% | 2020-08-14 | TV-MA | Brazil | JoÃ£oMiguel,BiancaComparato,MichelGomes,Ro... | TVShow |
| C000002 | (T)ERROR | 2016-06-30 | NR | UnitedStates |  | Movie |
| C000003 | 10,000B.C. | 2019-06-01 | PG-13 | UnitedStates,SouthAfrica | StevenStrait,CamillaBelle,CliffCurtis,JoelVirgel,A... | Movie |
| C000004 | 1BR | 2020-08-23 | TV-MA | UnitedStates | NicoleBrydonBloom,GilesMatthey,TaylorNichols,... | Movie |
| C000005 | 3Idiots | 2019-08-01 | PG-13 | India | AamirKhan,KareenaKapoor,Madhavan,Sharman... | Movie |
| C000006 | 6Balloons | 2018-04-06 | TV-MA | UnitedStates | AbbiJacobson,DaveFranco,JaneKaczmarek,Tim... | Movie |
| C000007 | ABillionColourStory | 2018-04-01 | TV-14 | India | DhruvaPadmakumar,GauravSharma,Vasuki | Movie |
| C000008 | AGhostStory | 2020-01-07 | R | UnitedStates | CaseyAffleck,RooneyMara,LizFranke,RobZabre... | Movie |
| C000009 | ALoveSongforLatasha | [illegible] | [illegible] | UnitedStates |  | Movie |
| C000010 | ASeriesofUnfortunateEvents | 2019-01-01 | TV-PG | UnitedStates | NeilPatrickHarris,PatrickWarburton,MalinaWeiss... | TVShow |
| C000011 | AWalktoRemember | 2020-07-01 | PG | UnitedStates | MandyMoore,ShaneWest,PeterCoyote,DarylHa... | Movie |
| C000012 | Aaviri | 2019-12-31 | TV-14 | India | RaviBabu,NehaChauhan,SriMuktha,BharaniSha... | Movie |
| C000013 | ActionReplayy | 2020-05-11 | TV-PG | India | AkshayKumar,AishwaryaRaiBachchan,OmPuri,Ki... | Movie |
| C000014 | AfterMaria | 2019-05-24 | TV-PG | UnitedStates |  | Movie |
| C000015 | AinsleyEatstheStreets | 2019-07-12 | TV-PG | UnitedKingdom | AinsleyHarriott | TVShow |
| C000016 | AlejandroRiaÃ±o:Especiald... | 2018-01-10 | TV-MA | Colombia | AlejandroRiaÃ±o | Movie |
| C000017 | AliensAteMyHomework | 2018-03-06 | PG | UnitedStates | WilliamShatner,DanPayne,KirstenRobek,TyCons... | Movie |
| C000018 | AllIWish | 2020-03-18 | R | UnitedStates | SharonStone,TonyGoldwyn,LizaLapira,EllenBurs... | Movie |
| C000019 | Amelia:ATaleofTwoSisters | 2017-06-01 | TV-PG | UnitedKingdom | RachaelStirling | Movie |
| C000020 | AmericanVandal | 2018-09-14 | TV-MA | UnitedStates | TylerAlvarez,GriffinGluck,JimmyTatro,CamilleHy... | TVShow |
| C000021 | Anaamika | 2020-09-17 | TV-14 | India | Nayantara,VaibhavReddy,Pasupathy,Harshvar... | Movie |

ALoveSongforLatasha

Fig4. Content Table

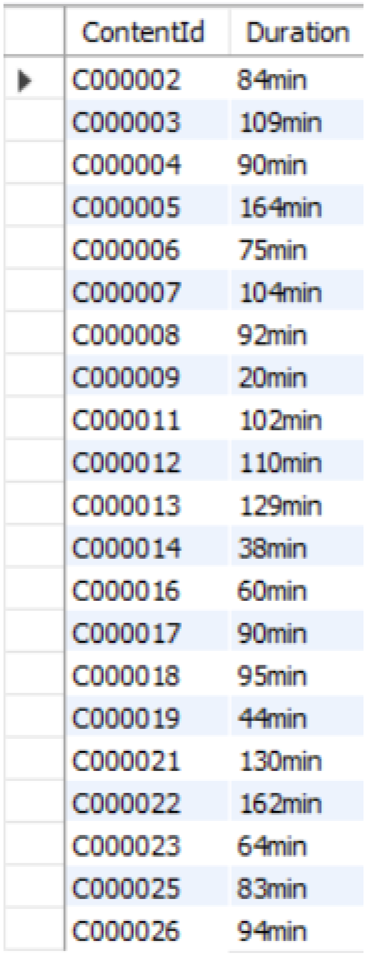
**5.** TV Shows Table:



| ContentId | Seasons |
|---|---|
| C000001 | 4Seasons |
| C000010 | 3Seasons |
| C000015 | 1Season |
| C000020 | 2Seasons |
| C000024 | 1Season |
| C000030 | 1Season |
| C000031 | 1Season |
| C000033 | 1Season |
| C000034 | 1Season |
| C000035 | 1Season |
| C000041 | 1Season |
| C000042 | 6Seasons |
| C000043 | 3Seasons |
| C000044 | 1Season |
| C000045 | 1Season |
| C000046 | 2Seasons |
| C000047 | 2Seasons |
| C000050 | 2Seasons |
| C000053 | 11Seasons |
| C000055 | 2Seasons |
| C000060 | 6Seasons |

Fig5. TV Shows Table

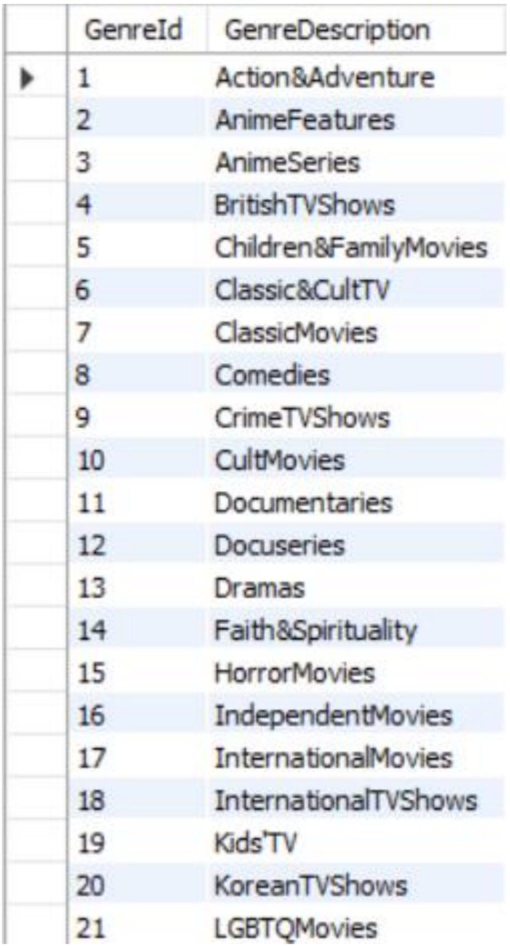
6. Movies Table:



| ContentId | Duration |
|---|---|
| C000002 | 84min |
| C000003 | 109min |
| C000004 | 90min |
| C000005 | 164min |
| C000006 | 75min |
| C000007 | 104min |
| C000008 | 92min |
| C000009 | 20min |
| C000011 | 102min |
| C000012 | 110min |
| C000013 | 129min |
| C000014 | 38min |
| C000016 | 60min |
| C000017 | 90min |
| C000018 | 95min |
| C000019 | 44min |
| C000021 | 130min |
| C000022 | 162min |
| C000023 | 64min |
| C000025 | 83min |
| C000026 | 94min |

Fig6. Movies Table

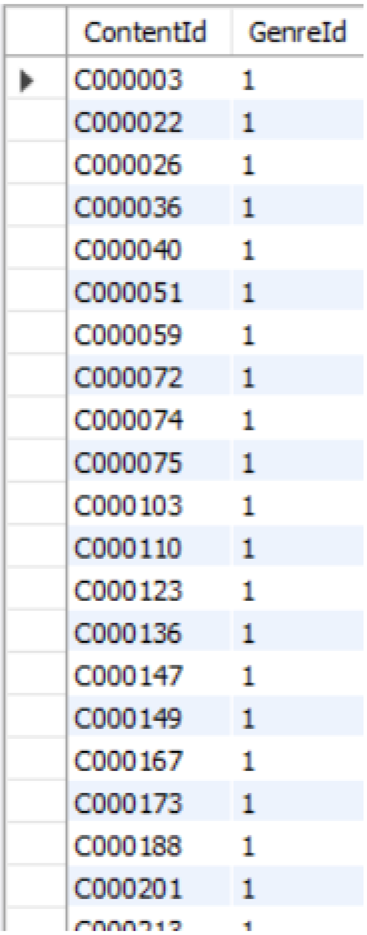
7. Genre Table:



| GenreId | GenreDescription |
|---|---|
| 1 | Action&Adventure |
| 2 | AnimeFeatures |
| 3 | AnimeSeries |
| 4 | BritishTVShows |
| 5 | Children&FamilyMovies |
| 6 | Classic&CultTV |
| 7 | ClassicMovies |
| 8 | Comedies |
| 9 | CrimeTVShows |
| 10 | CultMovies |
| 11 | Documentaries |
| 12 | Docuseries |
| 13 | Dramas |
| 14 | Faith&Spirituality |
| 15 | HorrorMovies |
| 16 | IndependentMovies |
| 17 | InternationalMovies |
| 18 | InternationalTVShows |
| 19 | Kids'TV |
| 20 | KoreanTVShows |
| 21 | LGBTQMovies |

Fig7. Genre Table

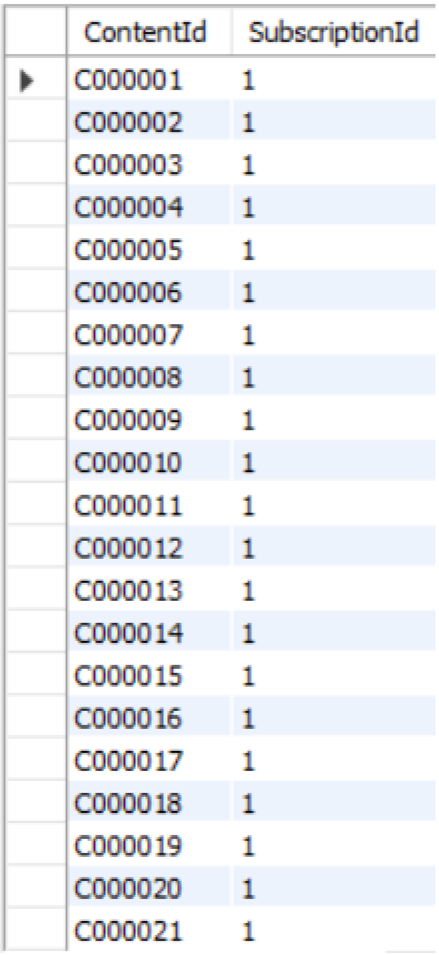
8.Content-Genre Table:



| ContentId | GenreId |
|---|---|
| C000003 | 1 |
| C000022 | 1 |
| C000026 | 1 |
| C000036 | 1 |
| C000040 | 1 |
| C000051 | 1 |
| C000059 | 1 |
| C000072 | 1 |
| C000074 | 1 |
| C000075 | 1 |
| C000103 | 1 |
| C000110 | 1 |
| C000123 | 1 |
| C000136 | 1 |
| C000147 | 1 |
| C000149 | 1 |
| C000167 | 1 |
| C000173 | 1 |
| C000188 | 1 |
| C000201 | 1 |
| C000213 | 1 |

Fig8. Content-Genre Table

9. Content-Subscription Table:

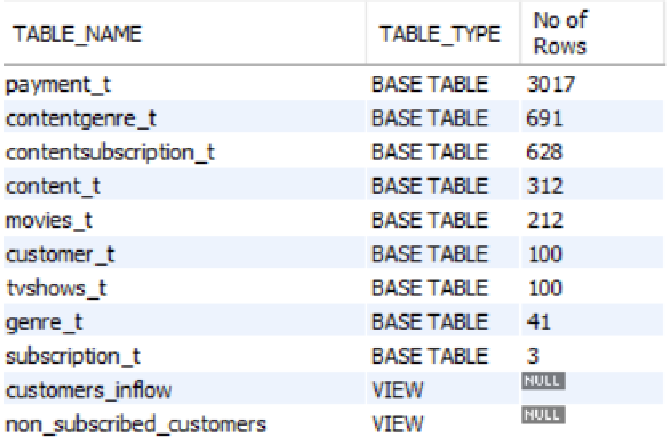


| ContentId | SubscriptionId |
|---|---|
| C000001 | 1 |
| C000002 | 1 |
| C000003 | 1 |
| C000004 | 1 |
| C000005 | 1 |
| C000006 | 1 |
| C000007 | 1 |
| C000008 | 1 |
| C000009 | 1 |
| C000010 | 1 |
| C000011 | 1 |
| C000012 | 1 |
| C000013 | 1 |
| C000014 | 1 |
| C000015 | 1 |
| C000016 | 1 |
| C000017 | 1 |
| C000018 | 1 |
| C000019 | 1 |
| C000020 | 1 |
| C000021 | 1 |

Fig9. Content-Subscription Table

Below table shows all the tables and views available in the database along with the count of rows for respective table.

```
SELECT
TABLE_NAME,
TABLE_TYPE,
SUM(TABLE_ROWS) AS "No of Rows"
FROM
INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA = 'netflix'
GROUP BY TABLE_NAME
ORDER BY SUM(TABLE_ROWS)
DESC;
```



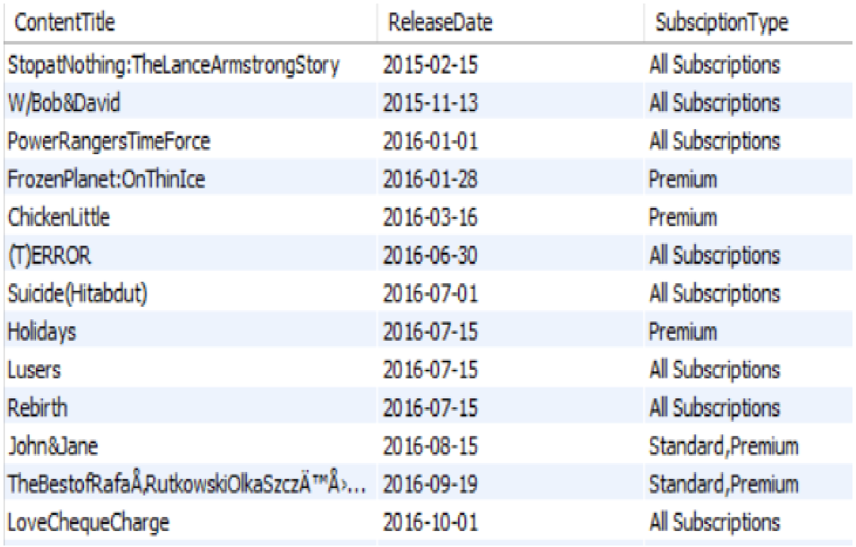
| TABLE_NAME | TABLE_TYPE | No of Rows |
|---|---|---|
| payment_t | BASE TABLE | 3017 |
| contentgenre_t | BASE TABLE | 691 |
| contentsubscription_t | BASE TABLE | 628 |
| content_t | BASE TABLE | 312 |
| movies_t | BASE TABLE | 212 |
| customer_t | BASE TABLE | 100 |
| tvshows_t | BASE TABLE | 100 |
| genre_t | BASE TABLE | 41 |
| subscription_t | BASE TABLE | 3 |
| customers_inflow | VIEW | NULL |
| non_subscribed_customers | VIEW | NULL |

## SQL Queries

### Query 1:

In this query we find the availability of a particular content amongst the other content types. To achieve the result, we have used Natural Join, Group by and Order By.

```
Select ContentTitle, ReleaseDate,
CASE
WHEN count(contentid) = 3 then 'All Subscriptions'
WHEN count(contentid) = 2 then 'Standard, Premium'
WHEN count(contentid) =1 then 'Premium'
END as SubscriptionType
from subscription_t
NATURAL JOIN contentsubscription_t
NATURAL JOIN content_t
GROUP BY ContentId
ORDER BY ReleaseDate,ContentTitle;
```



| ContentTitle | ReleaseDate | SubscriptionType |
|---|---|---|
| StopatNothing:TheLanceArmstrongStory | 2015-02-15 | All Subscriptions |
| W/Bob&David | 2015-11-13 | All Subscriptions |
| PowerRangersTimeForce | 2016-01-01 | All Subscriptions |
| FrozenPlanet:OnThinIce | 2016-01-28 | Premium |
| ChickenLittle | 2016-03-16 | Premium |
| (T)ERROR | 2016-06-30 | All Subscriptions |
| Suicide(Hitabdut) | 2016-07-01 | All Subscriptions |
| Holidays | 2016-07-15 | Premium |
| Lusers | 2016-07-15 | All Subscriptions |
| Rebirth | 2016-07-15 | All Subscriptions |
| John&Jane | 2016-08-15 | Standard,Premium |
| TheBestofRafaÅ‚RutkowskiOlkaSzczÄ™Å›... | 2016-09-19 | Standard,Premium |
| LoveChequeCharge | 2016-10-01 | All Subscriptions |

Fig10. Query1 Output

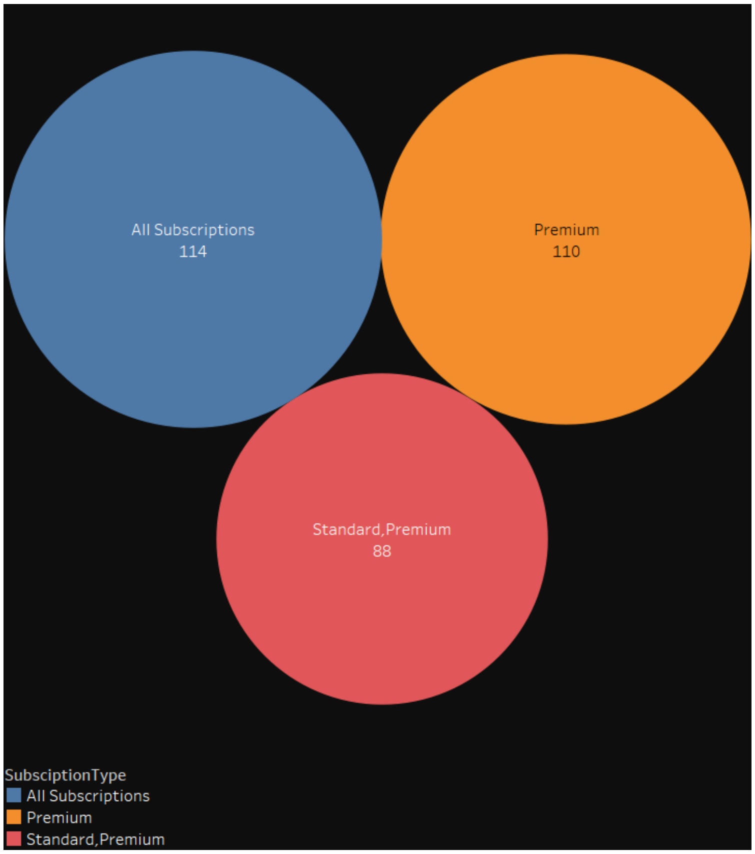
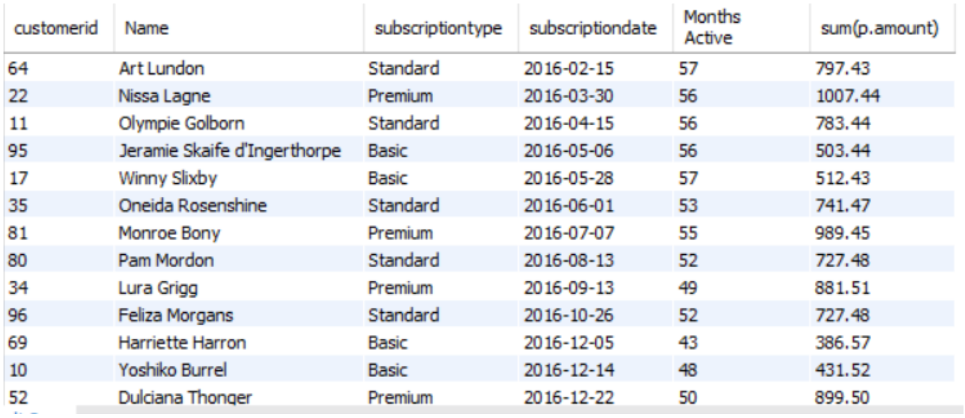


Fig11. Tableau Data of Query1

**Query 2:**

This Query is an example of Inner Join. We used Inner Join to extract data of customers and the tenure of their remaining subscription period along with the amount they have paid so far for their subscription with NETFLIX.

```
SELECT
c.customerid,concat(c.firstname," ",c.lastname) AS "Name",
s.subscriptiontype,
p.subscriptiondate,count(1) As "Months Active" ,sum(p.amount)
FROM payment_t AS p INNER JOIN subscription_t AS s
ON p.subscriptionid = s.SubscriptionId
INNER JOIN customer_t AS c
ON p.customerid = c.customerid
Group By c.customerid,s.subscriptionid
ORDER BY subscriptiondate ASC ,sum(p.amount) DESC;
```



| customerid | Name | subscriptiontype | subscriptiondate | Months Active | sum(p.amount) |
|---|---|---|---|---|---|
| 64 | Art Lundon | Standard | 2016-02-15 | 57 | 797.43 |
| 22 | Nissa Lagne | Premium | 2016-03-30 | 56 | 1007.44 |
| 11 | Olympie Golborn | Standard | 2016-04-15 | 56 | 783.44 |
| 95 | Jeramie Skaife d'Ingerthorpe | Basic | 2016-05-06 | 56 | 503.44 |
| 17 | Winny Slixby | Basic | 2016-05-28 | 57 | 512.43 |
| 35 | Oneida Rosenshine | Standard | 2016-06-01 | 53 | 741.47 |
| 81 | Monroe Bony | Premium | 2016-07-07 | 55 | 989.45 |
| 80 | Pam Mordon | Standard | 2016-08-13 | 52 | 727.48 |
| 34 | Lura Grigg | Premium | 2016-09-13 | 49 | 881.51 |
| 96 | Feliza Morgans | Standard | 2016-10-26 | 52 | 727.48 |
| 69 | Harriette Harron | Basic | 2016-12-05 | 43 | 386.57 |
| 10 | Yoshiko Burrel | Basic | 2016-12-14 | 48 | 431.52 |
| 52 | Dulciana Thonger | Premium | 2016-12-22 | 50 | 899.50 |

Fig12. Query2 Output

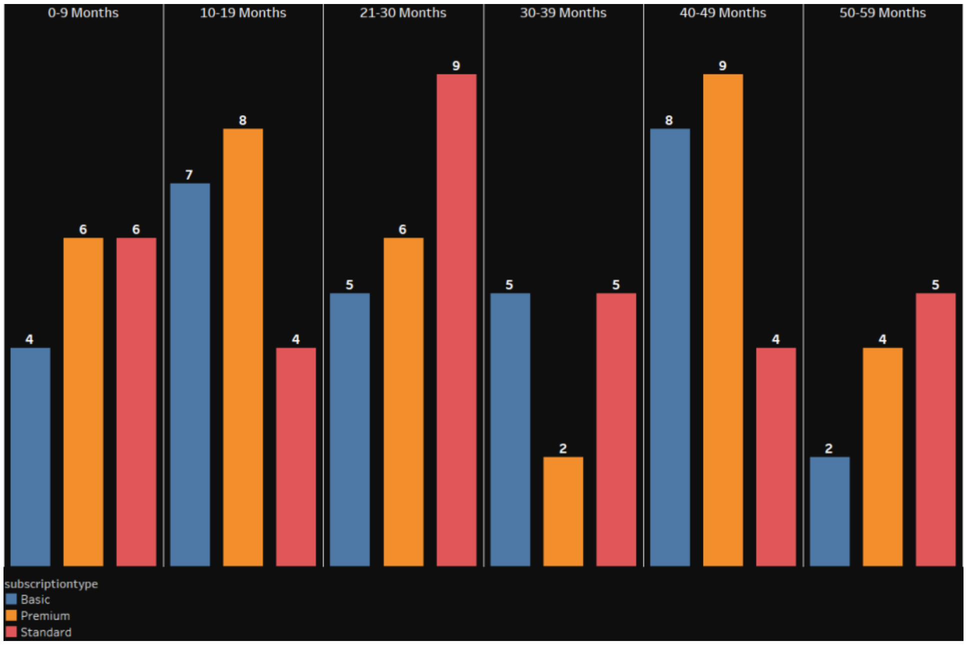


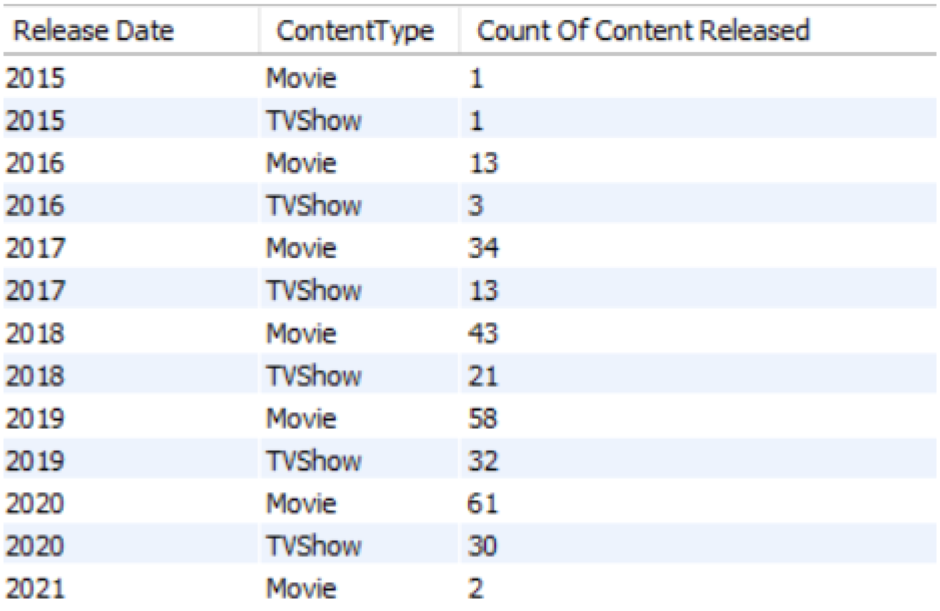
Fig13. Tableau Data of Query2

### Query 3:

This query is a representation of year wise content count. This is achieved by using Group By and Order By.

The tableau representation gives a pictorial view with respect to years and content type and count.

```
SELECT Year(ReleaseDate) "Release Date",
ContentType,count(distinct contentid) "Count Of Content Released"
from content_t
GROUP BY YEAR(ReleaseDate),ContentType
ORDER BY YEAR(ReleaseDate);
```



| Release Date | ContentType | Count Of Content Released |
|---|---|---|
| 2015 | Movie | 1 |
| 2015 | TVShow | 1 |
| 2016 | Movie | 13 |
| 2016 | TVShow | 3 |
| 2017 | Movie | 34 |
| 2017 | TVShow | 13 |
| 2018 | Movie | 43 |
| 2018 | TVShow | 21 |
| 2019 | Movie | 58 |
| 2019 | TVShow | 32 |
| 2020 | Movie | 61 |
| 2020 | TVShow | 30 |
| 2021 | Movie | 2 |

Fig14. Query 3 Output

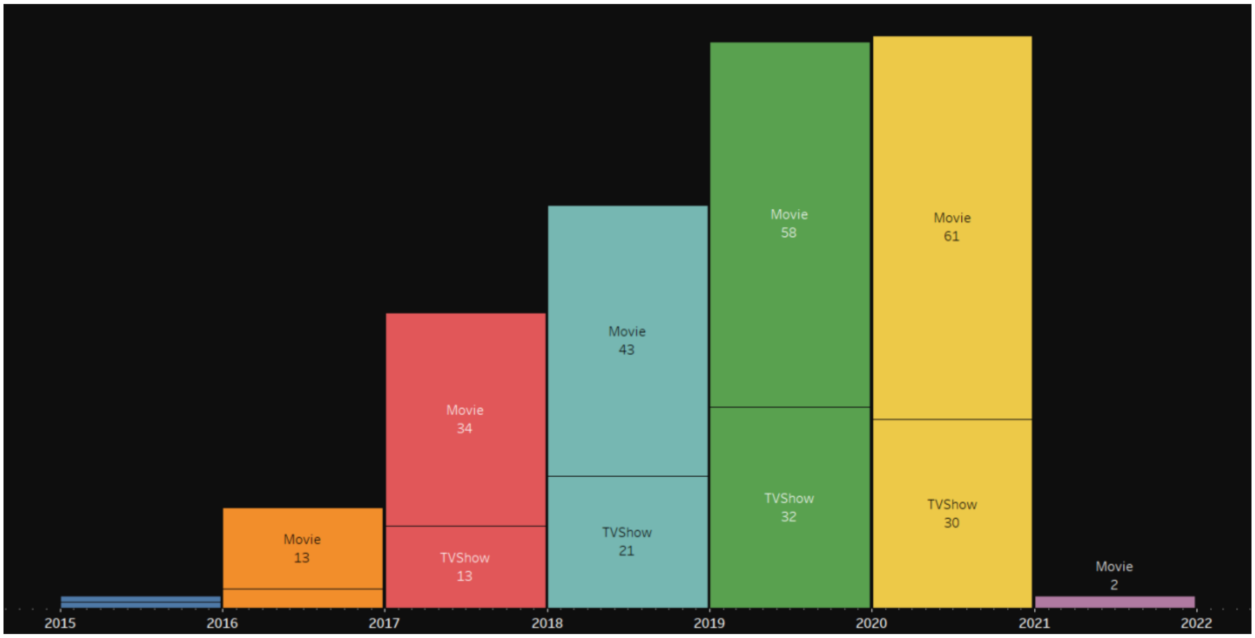
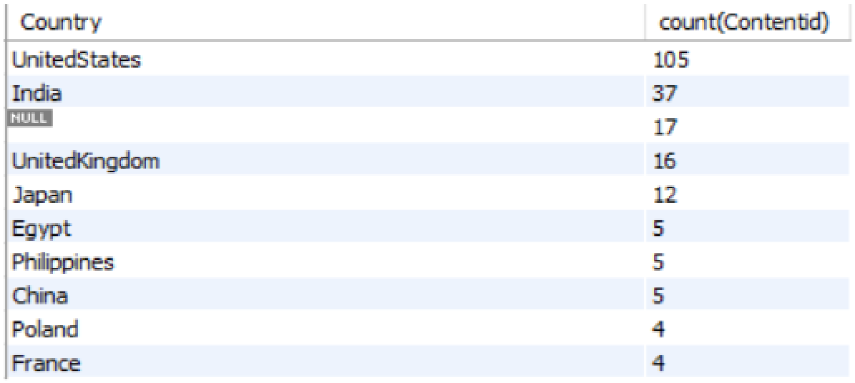


Fig15. Tableau Data of Query3

**Query 4:**

This query provides the count of content being produced by different countries. Here we have limited our output to 10. This is achieved by using Group By and Order By.

```
Select Country,count (Contentid) from content_t
GROUP BY Country
ORDER BY count (Contentid) DESC
LIMIT 10;
```



| Country | count(Contentid) |
|---|---|
| UnitedStates | 105 |
| India | 37 |
| NULL | 17 |
| UnitedKingdom | 16 |
| Japan | 12 |
| Egypt | 5 |
| Philippines | 5 |
| China | 5 |
| Poland | 4 |
| France | 4 |

Fig12. Query4 Output

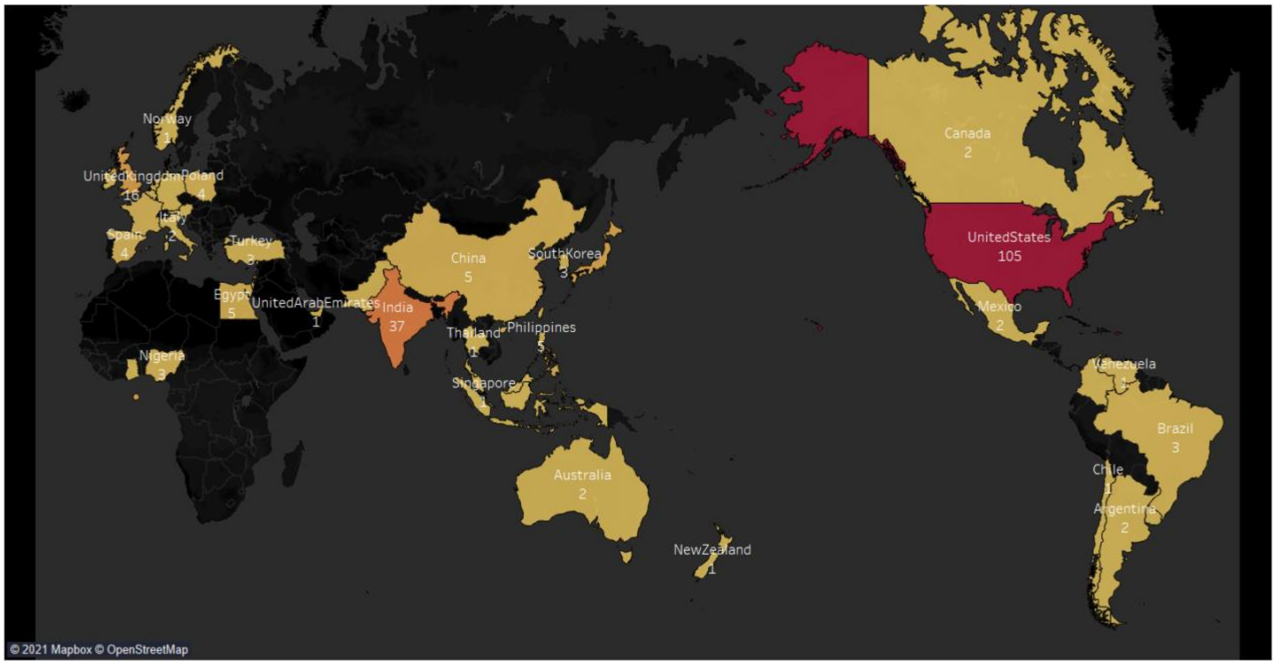
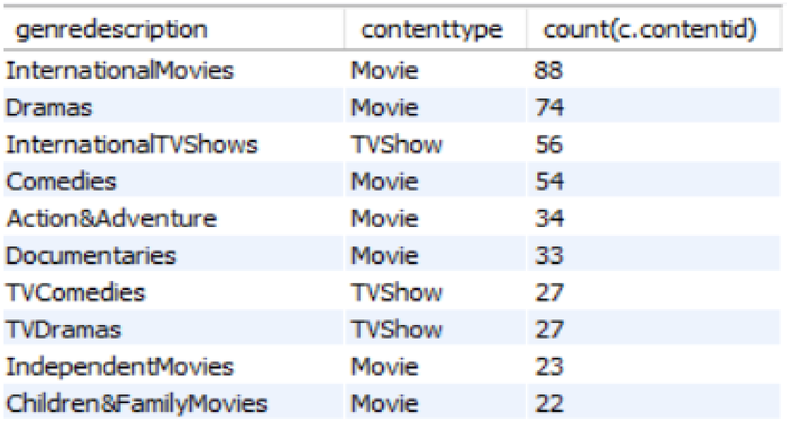


Fig15. Tableau Data of Query4

**Query 5:**

This query gives us a result of genre description, its relative content type along with the count of it that is present in the Netflix database. The limit of display count set is 10.

```
SELECT g.genredescription,count(c.contentid),contenttype FROM content_t as c,contentgenre_t
as cg,genre_t as g
WHERE c.contentid = cg.contentid AND g.genreid = cg.genreid
GROUP BY g.genreid,contenttype
ORDER BY count(c.contentid) DESC LIMIT 10;
```



| genredescription | contenttype | count(c.contentid) |
|---|---|---|
| InternationalMovies | Movie | 88 |
| Dramas | Movie | 74 |
| InternationalTVShows | TVShow | 56 |
| Comedies | Movie | 54 |
| Action&Adventure | Movie | 34 |
| Documentaries | Movie | 33 |
| TVComedies | TVShow | 27 |
| TVDramas | TVShow | 27 |
| IndependentMovies | Movie | 23 |
| Children&FamilyMovies | Movie | 22 |

Fig12. Query5 Output

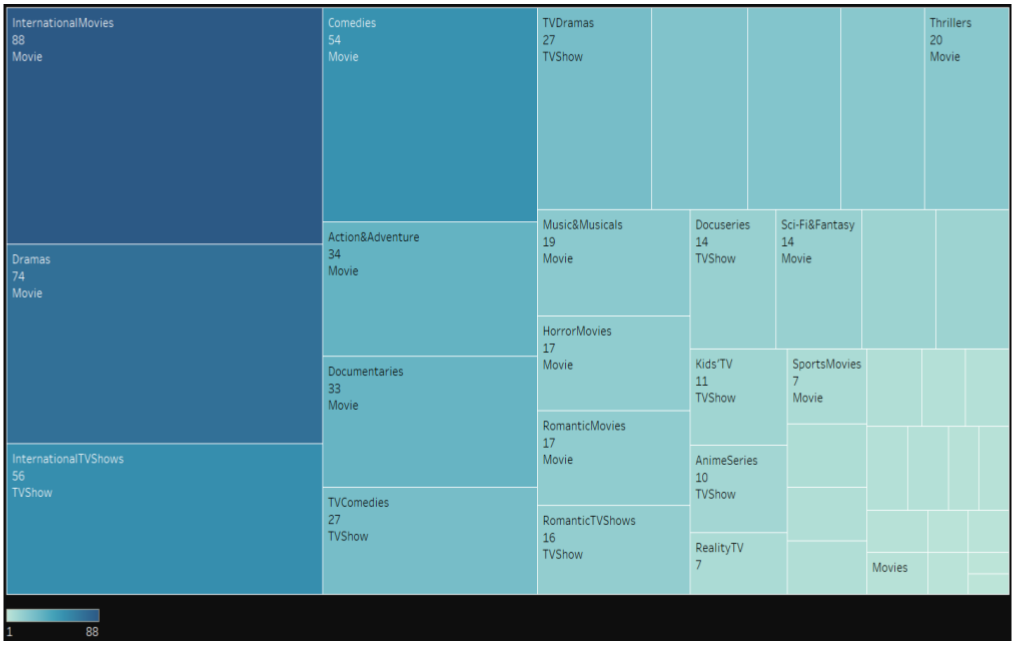


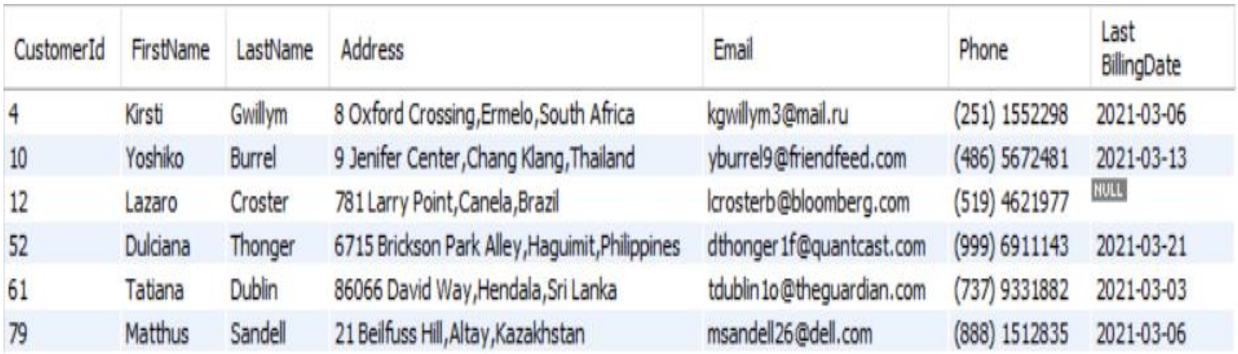
Fig15. Tableau Data of Query

**Query 6:**

This is a view created to display all the non-subscribed customers who have not paid for their subscription past their billing date.

```
CREATE VIEW NON_SUBSCRIBED_CUSTOMERS AS
SELECT * FROM customer_t NATURAL JOIN (SELECT T1.customerid,MAX(BillingDate)
AS 'Last BillingDate'
FROM payment_t as p RIGHT OUTER JOIN  (SELECT c.customerid from Customer_t as c
LEFT OUTER JOIN payment_t as p
on p.customerid = c.customerid
and DATEDIFF(CURTIME(),p.BillingDate) <= 31
WHERE p.BillingDate IS NULL) AS T1 ON
p.customerid = t1.customerid
GROUP BY CUSTOMERID) AS T;

SELECT * from NON_SUBSCRIBED_CUSTOMERS;
```



| CustomerId | FirstName | LastName | Address | Email | Phone | Last BillingDate |
|---|---|---|---|---|---|---|
| 4 | Kirsti | Gwillym | 8 Oxford Crossing,Ermelo,South Africa | kgwillym3@mail.ru | (251) 1552298 | 2021-03-06 |
| 10 | Yoshiko | Burrel | 9 Jenifer Center,Chang Klang,Thailand | yburrel9@friendfeed.com | (486) 5672481 | 2021-03-13 |
| 12 | Lazaro | Croster | 781 Larry Point,Canela,Brazil | lcrosterb@bloomberg.com | (519) 4621977 | NULL |
| 52 | Dulciana | Thonger | 6715 Brickson Park Alley,Haguimit,Philippines | dthonger1f@quantcast.com | (999) 6911143 | 2021-03-21 |
| 61 | Tatiana | Dublin | 86066 David Way,Hendala,Sri Lanka | tdublin1o@theguardian.com | (737) 9331882 | 2021-03-03 |
| 79 | Matthus | Sandell | 21 Belfuss Hill,Altay,Kazakhstan | msandell26@dell.com | (888) 1512835 | 2021-03-06 |

Fig12. Query6 Output

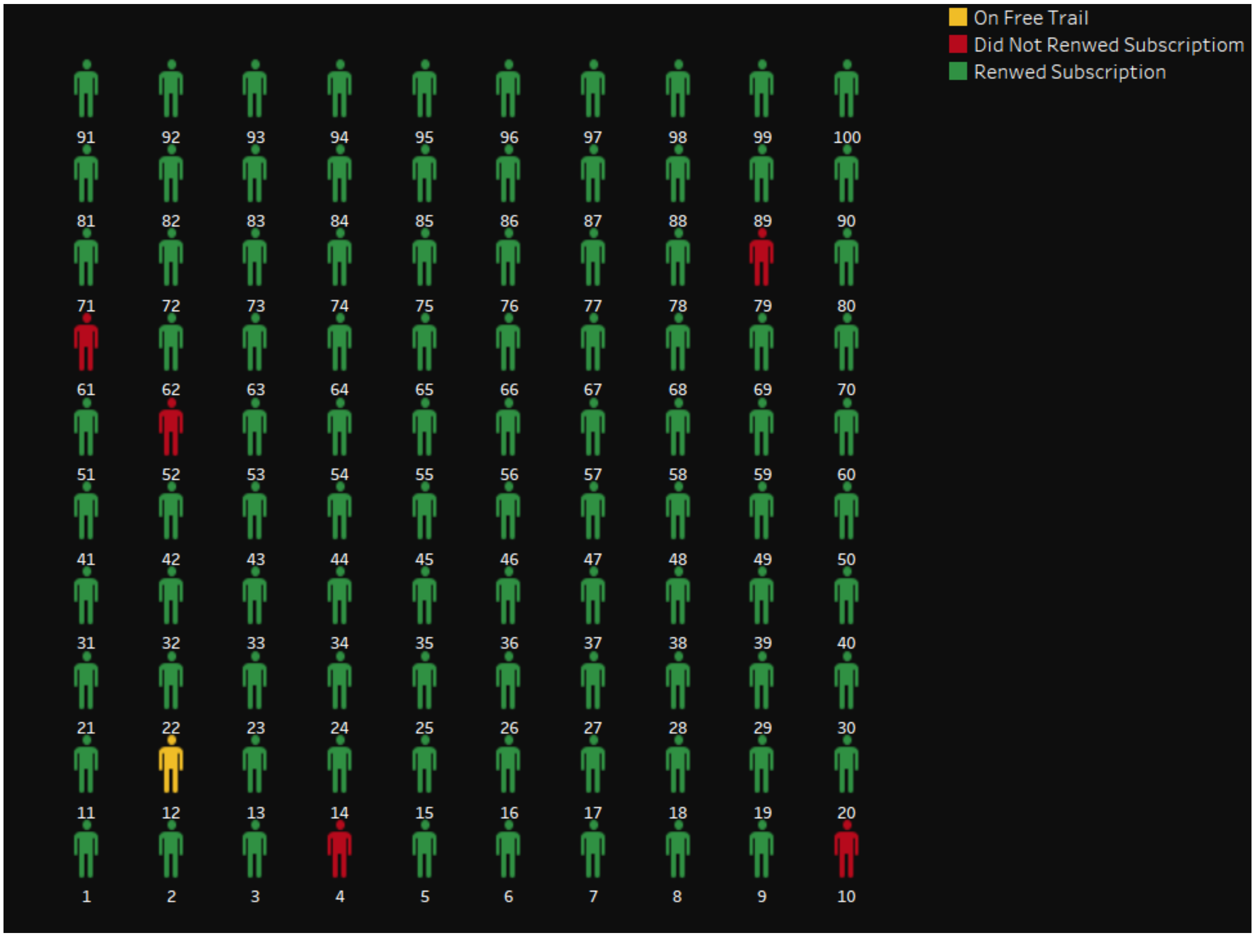


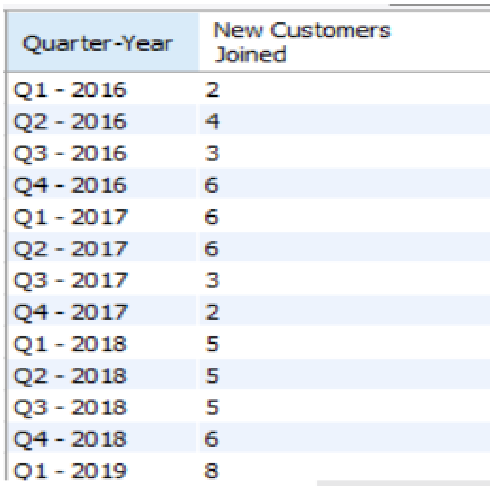
Fig15. Tableau Data of Query6

**Query 7:**

This view gives us detailed information of Customers inflow for each quarter. It clearly displays the total number of customers joined per quarter.

```
CREATE VIEW CUSTOMERS_INFLOW AS
SELECT CONCAT("Q",quarter(SubscriptionDate)," - ",year(subscriptionDate)) "Quarter-Year",
count(DISTINCT customerid) "New Customers Joined"
FROM payment_t
GROUP by quarter(SubscriptionDate),year(subscriptionDate)
ORDER BY year(subscriptionDate),quarter(SubscriptionDate);

SELECT * from CUSTOMERS_INFLOW;
```



| Quarter-Year | New Customers Joined |
|---|---|
| Q1 - 2016 | 2 |
| Q2 - 2016 | 4 |
| Q3 - 2016 | 3 |
| Q4 - 2016 | 6 |
| Q1 - 2017 | 6 |
| Q2 - 2017 | 6 |
| Q3 - 2017 | 3 |
| Q4 - 2017 | 2 |
| Q1 - 2018 | 5 |
| Q2 - 2018 | 5 |
| Q3 - 2018 | 5 |
| Q4 - 2018 | 6 |
| Q1 - 2019 | 8 |

Fig12. Query7 Output

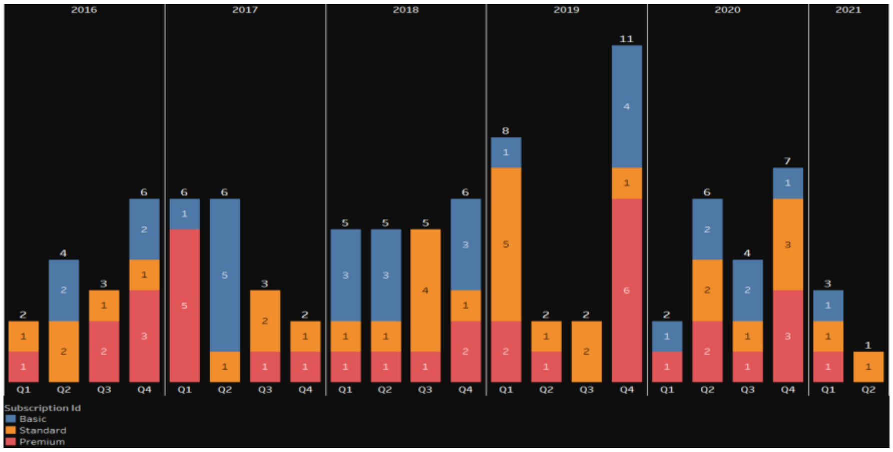


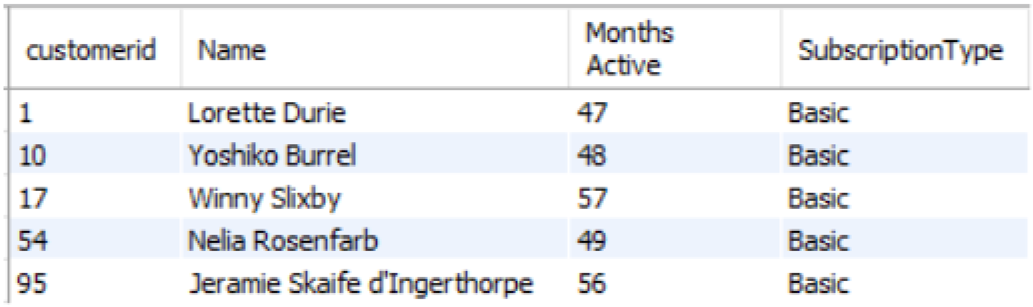
Fig15. Tableau Data of Query7

**Query 8:**

This is a Stored Procedure which provides count of active months of the customer and type of subscription.

```
DELIMITER //
CREATE PROCEDURE Active_months (IN m int(10),subtype varchar(20))
BEGIN
Select
c.customerid,concat(c.firstname," ",c.lastname) AS "Name",count(p.BillingDate) As "Months
Active" ,s.SubscriptionType
from payment_t as p
INNER JOIN customer_t as c
on p.customerid = c.customerid INNER JOIN subscription_t as s
on s.subscriptionid = p.subscriptionid
Group By c.customerid,s.subscriptionid
HAVING count(p.BillingDate) > m and lower(s.SubscriptionType) = lower(subtype);
END //
DELIMITER ;
```

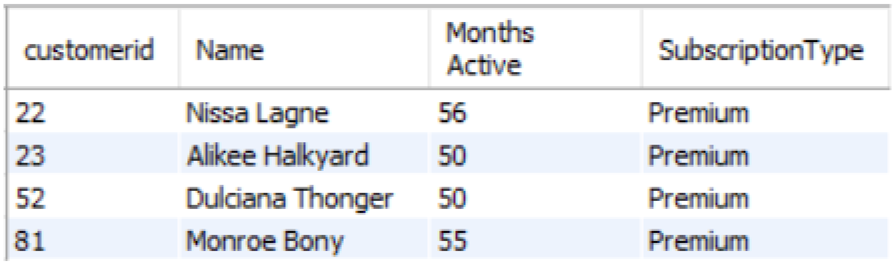
```
CALL ACTIVE_MONTHS (45,'basic');
```



| customerid | Name | Months Active | SubscriptionType |
|---|---|---|---|
| 1 | Lorette Durie | 47 | Basic |
| 10 | Yoshiko Burrel | 48 | Basic |
| 17 | Winny Slixby | 57 | Basic |
| 54 | Nelia Rosenfarb | 49 | Basic |
| 95 | Jeramie Skaife d'Ingerthorpe | 56 | Basic |

Fig16. Query8 Output

```
CALL ACTIVE_MONTHS (50,'premium');
```



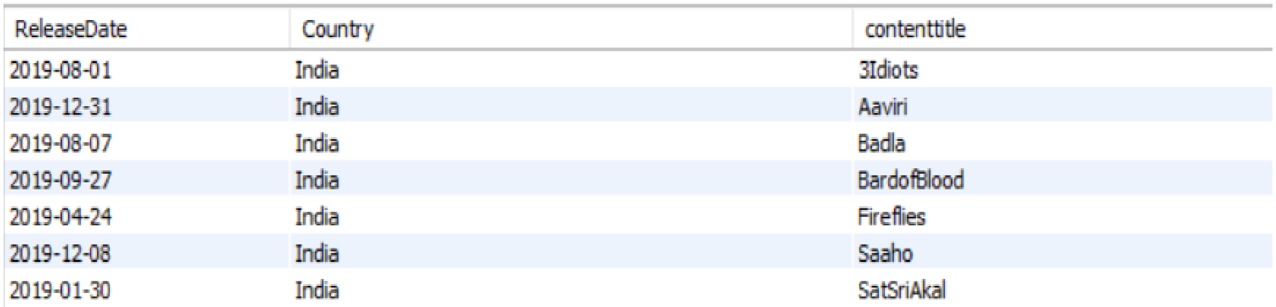
| customerid | Name | Months Active | SubscriptionType |
|---|---|---|---|
| 22 | Nissa Lagne | 56 | Premium |
| 23 | Alikee Halkyard | 50 | Premium |
| 52 | Dulciana Thonger | 50 | Premium |
| 81 | Monroe Bony | 55 | Premium |

Fig17. Query8 Output

**Query 9:**

This Stored Procedure provides data of content released in a particular country in a particular given period of time.

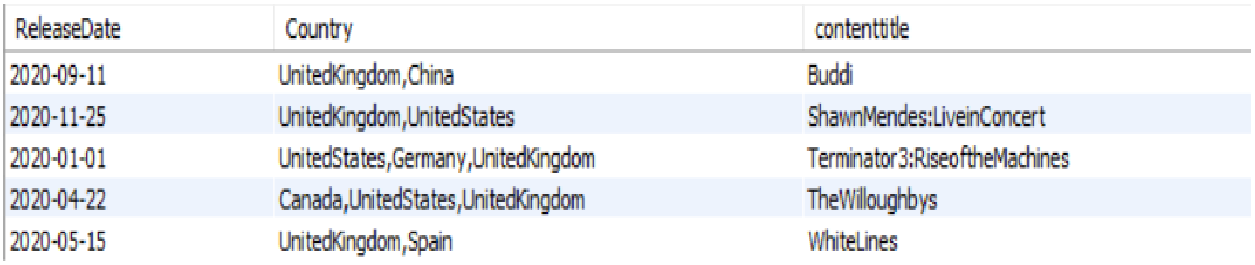
```
DELIMITER //
CREATE PROCEDURE released_data (IN y_date int(10), c varchar(20))
BEGIN
set @country = lower(c);
SELECT ReleaseDate, Country, contenttitle from content_T
WHERE year(ReleaseDate) =y_date
and lower(country) like CONCAT('%', @country, '%');
END //
DELIMITER ;
CALL released_data(2019, "India");
```



| ReleaseDate | Country | contenttitle |
|---|---|---|
| 2019-08-01 | India | 3Idiots |
| 2019-12-31 | India | Aaviri |
| 2019-08-07 | India | Badla |
| 2019-09-27 | India | BardofBlood |
| 2019-04-24 | India | Fireflies |
| 2019-12-08 | India | Saaho |
| 2019-01-30 | India | SatSriAkal |

Fig18. Query9 Output

```
CALL released_data(2019, "United Kingdom");
```



| ReleaseDate | Country | contenttitle |
|---|---|---|
| 2020-09-11 | UnitedKingdom,China | Buddi |
| 2020-11-25 | UnitedKingdom,UnitedStates | ShawnMendes:LiveinConcert |
| 2020-01-01 | UnitedStates,Germany,UnitedKingdom | Terminator3:RiseoftheMachines |
| 2020-04-22 | Canada,UnitedStates,UnitedKingdom | TheWilloughbys |
| 2020-05-15 | UnitedKingdom,Spain | WhiteLines |

Fig19. Query9 Output