Our project is a database of a streaming service, it holds movies and tv shows. The database also contains the data of Users along with their user information.

Card Type: Reference table describing credit card types. Supports payment validation.

```
CREATE TABLE CardType
(
  CardTypeID BIGINT IDENTITY(1,1) PRIMARY KEY,
  CardType   NVARCHAR(100)
);
```

User: Stores user account information such as name, email, and password. Required for authentication and linking to payment methods, watch, history, and purchase.

```
CREATE TABLE [User]
(
  UserID    BIGINT IDENTITY(1,1) PRIMARY KEY,
  FirstName  NVARCHAR(100) NOT NULL,
  LastName   NVARCHAR(100) NOT NULL,
  Email     NVARCHAR(100) NOT NULL UNIQUE,
  Password   NVARCHAR(100) NOT NULL
);
```

Payment Methods: Stores each user's payment card. Linked to user and Card Type to ensure integrity.

```
CREATE TABLE PaymentMethods
(
  CardNumber     BIGINT PRIMARY KEY,
  UserID        BIGINT UNIQUE,
  CardTypeID     BIGINT,
  ExpirationMonth BIGINT,
  ExpirationYear  BIGINT,
  FOREIGN KEY (UserID)    REFERENCES [User](UserID),
  FOREIGN KEY (CardTypeID) REFERENCES CardType(CardTypeID)
);
```

Company: Represent production studios needed for organizing an filtering content.

```
CREATE TABLE Company.
(
  CompanyID BIGINT IDENTITY(1,1) PRIMARY KEY,
  Name     NVARCHAR(100) NOT NULL UNIQUE
);
```

Director: Stores film directors allowing content to be linked to the person who directed it.

```
CREATE TABLE Director
(
  DirectorID BIGINT IDENTITY(1,1) PRIMARY KEY,
  FirstName  NVARCHAR(100) NOT NULL,
  LastName   NVARCHAR(100) NOT NULL
);
```

Content Type: Reference/category payment card. Linked to user and card type. Supports payment validation.

```
CREATE TABLE ContentType
(
  ContentTypeID BIGINT IDENTITY(1,1) PRIMARY KEY,
  Type        NVARCHAR(100)
);
```

Content: Main data table storing movies series documentaries etc. Reference company director and content type.

```
CREATE TABLE Content
(
  ContentID    BIGINT IDENTITY(1,1) PRIMARY KEY,
  Title        NVARCHAR(100) NOT NULL,
  ContentTypeID BIGINT NOT NULL,
  CompanyID    BIGINT NOT NULL,
  DirectorID   BIGINT NOT NULL,
  FOREIGN KEY (ContentTypeID) REFERENCES ContentType(ContentTypeID),
  FOREIGN KEY (CompanyID)    REFERENCES Company(CompanyID),
  FOREIGN KEY (DirectorID)   REFERENCES Director(DirectorID)
```

```
);
```

Watch log: Tracks when a user watches a piece of content. Used for analytic and recommend.

```
CREATE TABLE Watchlog
(
   WatchlogID BIGINT IDENTITY(1,1) PRIMARY KEY,
   WatchDate  DATE NOT NULL,
   UserID     BIGINT,
   ContentID  BIGINT,
   FOREIGN KEY (UserID)    REFERENCES [User](UserID),
   FOREIGN KEY (ContentID) REFERENCES Content(ContentID)
);
```

Purchase Content: Tracks content users have paid for. Useful for account history billing.

```
CREATE TABLE PurchasedContent
(
   PurchasedContentID BIGINT IDENTITY(1,1) PRIMARY KEY,
   ContentID        BIGINT,
   UserID           BIGINT,
   FOREIGN KEY (ContentID) REFERENCES Content(ContentID),
   FOREIGN KEY (UserID)    REFERENCES [User](UserID)
);
```

Actor: Stores actors who appear in content. Used for many to many via featuring.

```
CREATE TABLE Actor
(
   ActorID   BIGINT IDENTITY(1,1) PRIMARY KEY,
   FirstName NVARCHAR(100) NOT NULL,
   LastName  NVARCHAR(100) NOT NULL
);
```

Featuring: Many to many join table linking actors to content.

```
CREATE TABLE Featuring.
(
   FeaturingID BIGINT IDENTITY(1,1) PRIMARY KEY,
```

```
  ContentID   BIGINT NOT NULL,
  ActorID     BIGINT NOT NULL,
  FOREIGN KEY (ContentID) REFERENCES Content(ContentID),
  FOREIGN KEY (ActorID)   REFERENCES Actor(ActorID)
```

Use Cases:

Explanation: This use case retrieves all content on the platform and all information directly connected to that content, like Director, Content Type, Title and Company.

```
SELECT
  c.Title,
  ct.Type AS ContentType,
  comp.Name AS Company,
  d.FirstName + ' ' + d.LastName AS Director
FROM Content c
INNER JOIN ContentType ct ON c.ContentTypeID = ct.ContentTypeID
INNER JOIN Company comp ON c.CompanyID = comp.CompanyID
INNER JOIN Director d ON c.DirectorID = d.DirectorID;
```

Explanation: This use case retrieves data from the watch-log to show a given user what they have watched and when.

```
SELECT
  c.Title,
  w.WatchDate
FROM Watchlog w
INNER JOIN Content c ON w.ContentID = c.ContentID
WHERE w.UserID = '18'
ORDER BY w.WatchDate DESC;
```

Explanation: This use case lets a user see all content they have permanently purchased.

```
SELECT
   c.Title,
   ct.Type,
   comp.Name AS Company
FROM PurchasedContent pc
INNER JOIN Content c ON pc.ContentID = c.ContentID
INNER JOIN ContentType ct ON c.ContentTypeID = ct.ContentTypeID
INNER JOIN Company comp ON c.CompanyID = comp.CompanyID
WHERE pc.UserID = '4';
```

Explanation: This query uses info from the watch-log to determine what content is the most watched by all users, and display that content to the user.

```
SELECT
   c.Title,
   COUNT(w.WatchLogID) AS Watches
FROM Watchlog w
INNER JOIN Content c ON w.ContentID = c.ContentID
WHERE w.WatchDate >= DATEADD(DAY, 365, GETDATE())
GROUP BY c.Title
ORDER BY Watches DESC;
```

Explanation: This use case looks over all content on the platform for a particular actor and returns all content that actor is listed as present in.

```
SELECT
```

```
    c.Title,
    a.FirstName + ' ' + a.LastName AS Actor
FROM Content c
INNER JOIN Featuring f ON c.ContentID = f.ContentID
INNER JOIN Actor a ON f.ActorID = a.ActorID
WHERE a.LastName = 'Carpenter';
```