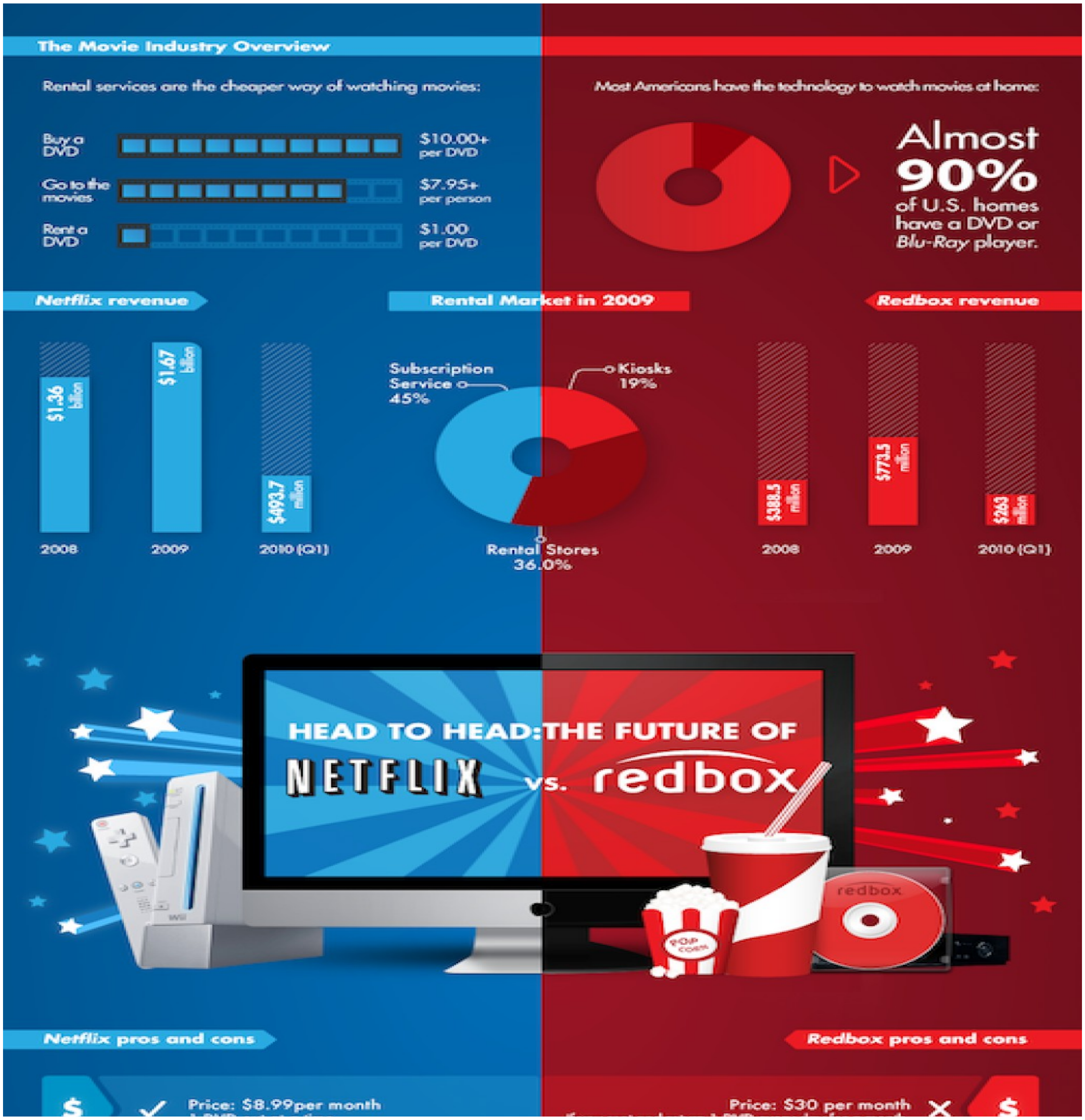
Team 4
IS380 Database Management Project

Video Rental Solution

[Authors remove for privacy purposes]



Introduction

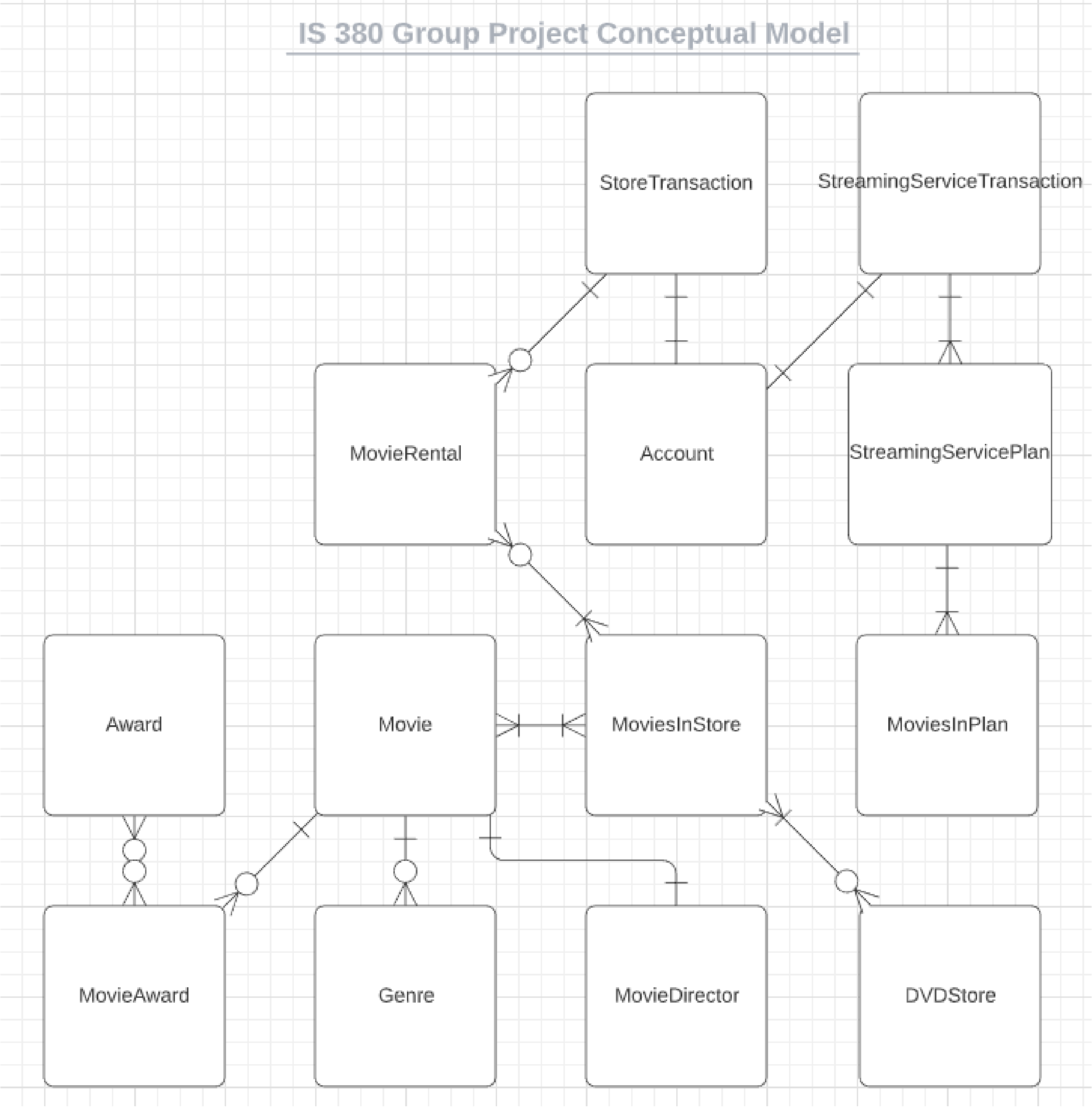
The movie rental industry has changed drastically over the years. In the past, movie fans could only rent movies at their local movie renting store, but the emergence of new technologies and movie distribution methods has changed the way the movie rental industry operates. Today, there are two major ways for movie fans to obtain their movies: the traditional movie rental store, and streaming online services.

While the movie rental industry is rapidly changing, movie rental stores are still very financially successful. Movie rental stores make their money through a number of different methods including membership fees, rental fees, late fees, and movie purchases. Because online movie rental services are both cheap and convenient, they are quickly becoming a strong competitor within the movie rental industry. Both options are ideal for movie rental enthusiasts who enjoy renting movies regularly but don't want to be bothered with returning the movies.

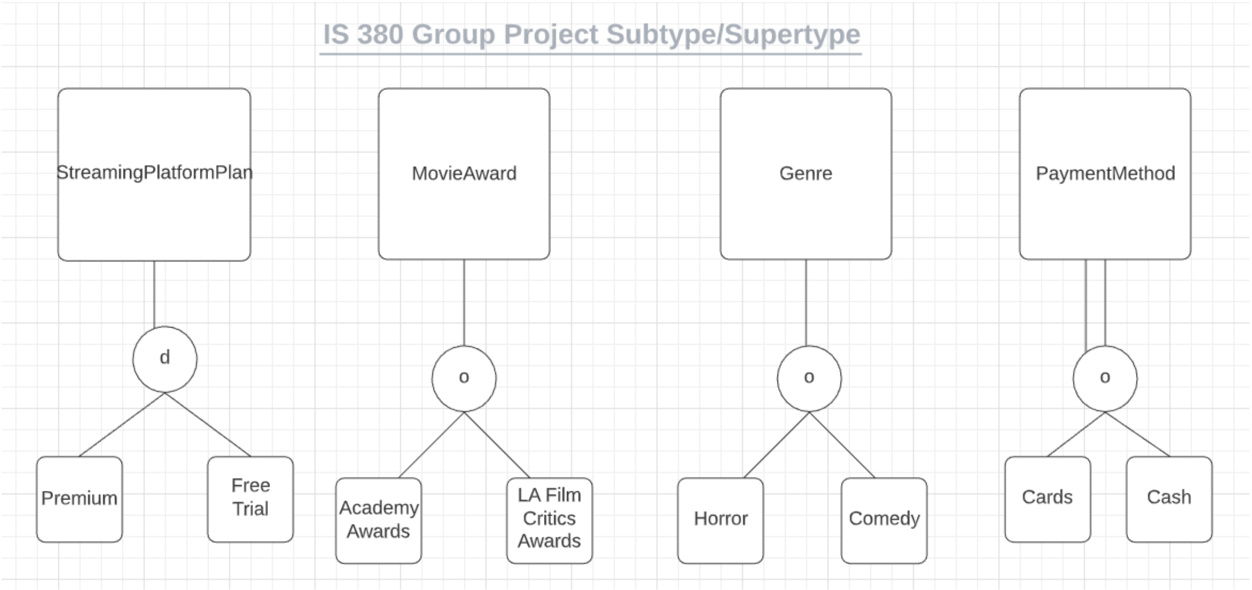
The companies that dominate the video rental industry are Netflix, Inc and Redbox Automated Retail LLC. Netflix, Inc. is an American over-the-top content platform and production company headquartered in Los Gatos, California, which was founded in 1997. Redbox Automated Retail LLC is an American video rental company specializing in DVD, Blu-ray, 4K UHD rentals, which was founded in 2002. The platforms' success is due to the good service both provide to their members and the wide and varied stock of videos/movies available to watch. The stores are having difficulties in managing the increasing amount of data used and generated because they can't cope with their own success. They now offer a larger selection of videos to a growing number of members, which means that the level of service they provide is falling. A system that will speed up the way they work, such as something to automate a lot of the day-to-day tasks that seem to take forever to complete is needed. The aim of this project is to develop a database application that will help solve the increasing problems of data management by facilitating efficient data storage and retrieval.

In the design of the database, we have included a conceptual model, a logical model, a data dictionary, physical design of the database, and the description of some of the tasks expected from the database and related SQL queries.

A.Conceptual Model

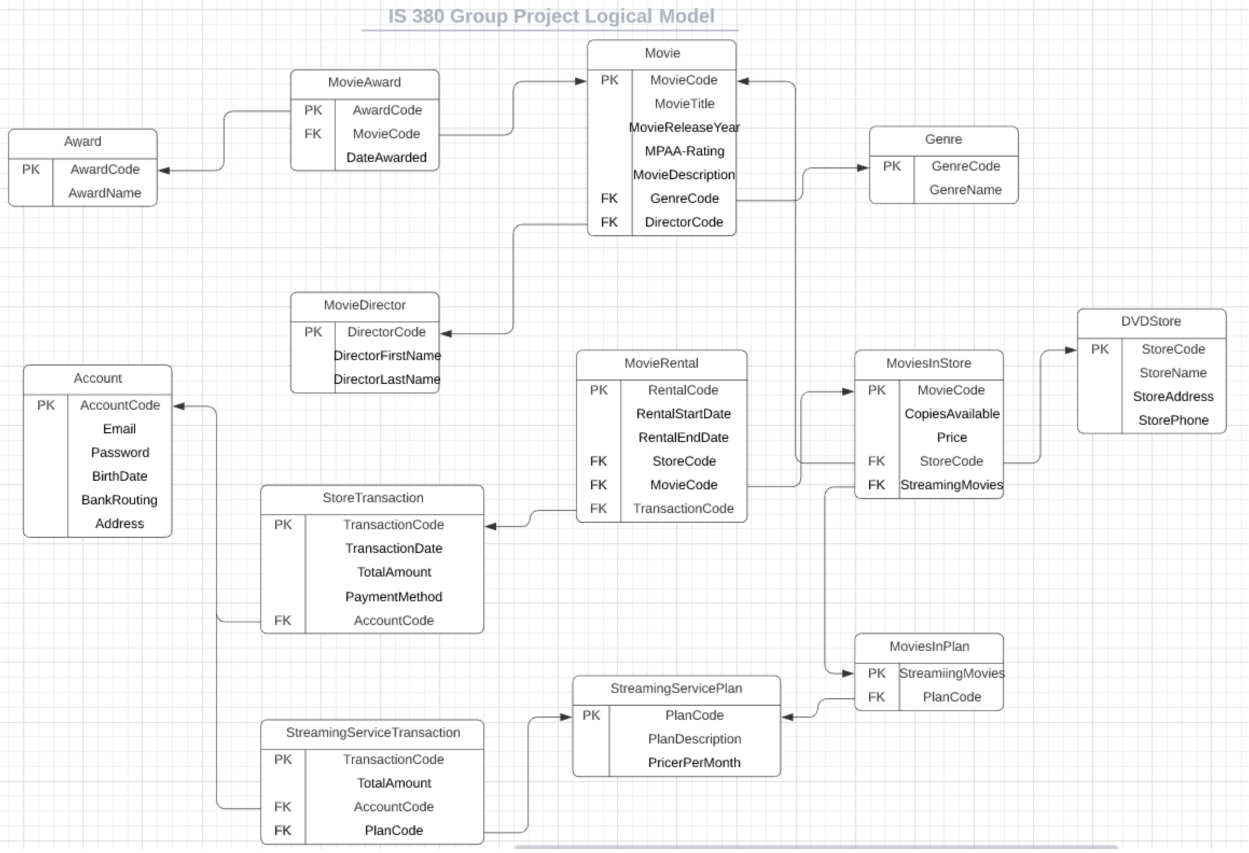


For our conceptual model, we have included 13 entities with appropriate relationships. We used multiple relationship types; mandatory one, optional one, mandatory many, and optional many, to clearly show the link between entities. Majority of the entities were linked with Movie Entity. For example, a one-to-many relationship is the type of cardinality that refers to the relationship between StreamingServicePlan and MoviesInPlan entities, in which a Streaming Plan may be linked to many elements of Movies available in the selected plan, but all of the movies are linked to only one element of streaming plan.

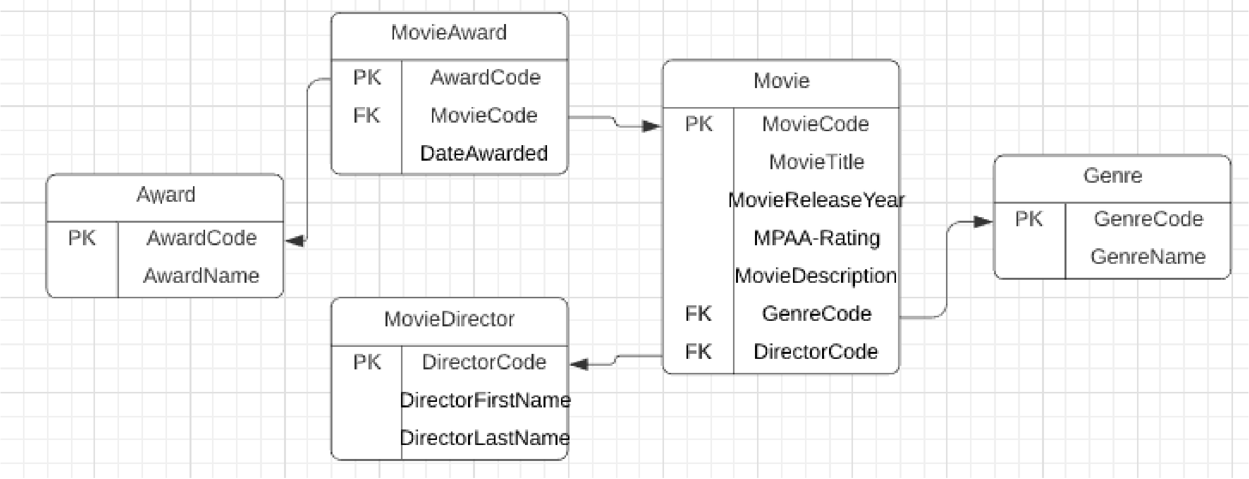


I have also included Supertype-Subtype Identity transform to create an identifying relationship between a supertype entity and its subtype entities to simplify the model and improve the query performance. In the model, there are two types of constraints in supertype/subtype relationships. One is called a completeness constraint, which indicates whether there must be an explicit subtype for each possible instance of the supertype. The other is called a disjointness constraint, which indicates whether a particular instance of a supertype could also be more than one of the subtypes. For a PaymentMethod entity, the method must be either Cards or Cash. There is no other possibility. This is represented by double lines coming down from the supertype entity type with overlap (o). Under the overlap rule, the method can be either Cards or Cash, but not both at the same time.

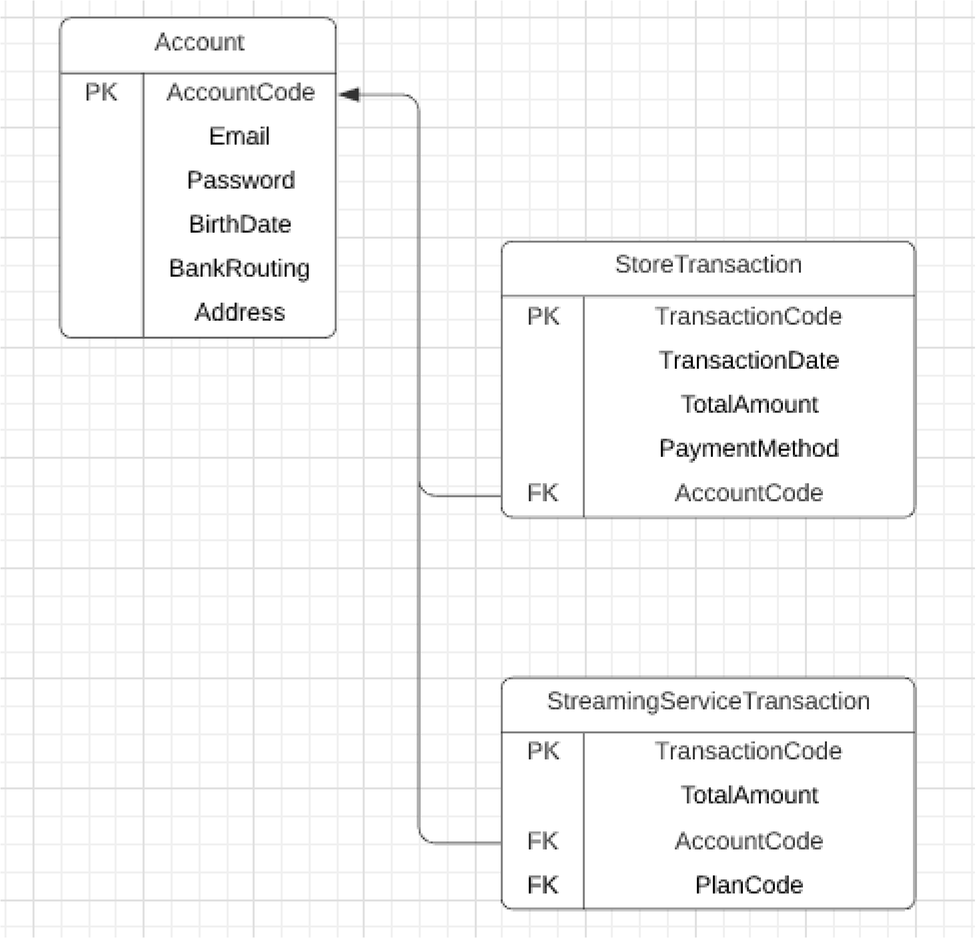
B.Logical Model



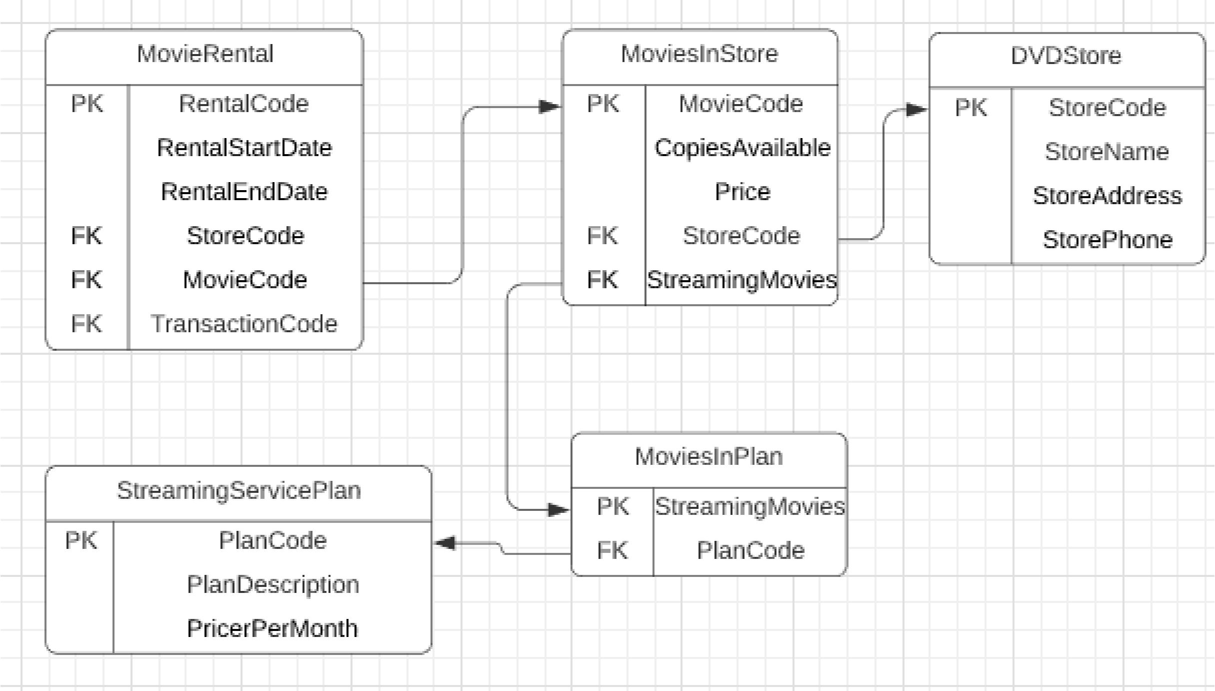
This is the complete logical model that we created. However, as you can see, EER diagrams are difficult to read when there are too many entities and relationships. To fix this solution, we decided to group the entities and relationships into entity clusters. Entity cluster is a set of one or more entity types and associated relationships grouped into a single abstract entity type. At the end, we clustered 13 entities into three clusters.



In the logical model, we have specified the primary keys and foreign keys, which were not present in the conceptual model. Along with the keys, we have also added the attributes that were not specified in the conceptual data model. For the Movie Entity Clustering, we could see that all of the entities are linked to the “Movie” entity. Directors who directed the movie will have to be linked to the movie. In order for the movie to receive the award, it needs to know which movie to grant. Every movie will have genre under description.



The Account Entity Clustering will cover the area of transaction and payment method. The customer will create the account to enter the credit/debit card information. This will determine how the customer will pay when he or she rents or purchase the video. For record purposes, store transactions and streaming service transactions will be created to show the amount and other essential information when they purchased a streaming service plan or at the physical location.



The Rental Entity Clustering can give information about the customer who utilizes the video rental service. With the MovieRental entity, the store can find out the customer's current status regarding the movie rented. MoviesInStore can be used to determine what movies are available to customers, whether it’s going to be for a physical store or streaming online. MoviesInPlan can show what movies are available when customers purchase the streaming service plan to watch. Therefore, the StreamingServicePlan entity can determine what streaming service plan customers purchased.

C.Data Dictionary

Entity: Movie

This table contains information about the movies available on the platform.

Column	Data Type	Description	Range	Required	PK or FK
MovieCode	INT	Primary key of the “Movie” entity	00001-99999	Y	PK
MovieTitle	VARCHAR	Title of the movie		Y	
MovieReleaseYear	DATE	Year the movie was released		Y	
MovieDescription	VARCHAR	Brief description about the movie		Y	
MPAA-Rating	VARCHAR	The rating the movie received as established by MPAA. e.g. PG-13		Y	
GenreCode	INT	Genre the movie belongs to. e.g. Horror	00001-99999	Y	FK

DirectorCode	**INT**	**Unique code for each director**	**00001-99999**	**Y**	**FK**

Entity: MovieRental

This table contains information about the movies the customer has rented.

Column	Data Type	Description	Range	Required	PK or FK
MovieCode	**INT**	**Primary key of the “Movie” entity**	**00001-99999**	**Y**	**PK**
MovieTitle	**VARCHAR**	**Title of the movie**		**Y**	
MovieReleaseYear	**DATE**	**Year the movie was released**		**Y**	
MovieDescription	**VARCHAR**	**Brief description about the movie**		**Y**	
MPAA-Rating	**VARCHAR**	**The rating the movie received as established by MPAA. e.g. PG-13**		**Y**	
GenreCode	**INT**	**Genre the**	**00001-**	**Y**	**FK**

		movie belongs to. e.g. Horror	99999		
DirectorCode	INT	Unique code for each director	00001-99999	Y	FK

Entity: Genre

This table contains information about the genre the movie belongs along with their descriptions.

Column	Date Type	Description	Range	Required	PK or FK
GenreCode	INT	Unique identifier for “Genre”	00001-99999	Y	PK
GenreName	CHAR	Name of the genre		Y	

Entity: MoviesInStore

This table contains information about the movies that are available physically in stores.

Column	Date Type	Description	Range	Required	PK or FK
MovieCode	INT	Unique identifier for “MoviesInStore”	00001-99999	Y	PK
CopiesAvailable	INT	# of copies available at a particular	00001-99999	Y	

		location			
Price	VARCHAR	The price of each movie		Y	
StoreCode	INT	Physical location store codes	00001-99999	Y	FK
StreamingMovies	VARCHAR	Movies that are available to be streamed on the platform		Y	FK

Entity: MoviesInPlan

This table contains information about the movies that are available to be streamed on the platform.

Column	Date Type	Description	Range	Required	PK or FK
StreamingMovies	VARCHAR	Unique identifier for the “MoviesInPlan” entity		Y	PK
PlanCode	INT	Type of plan the customer is subscribed to	00001-99999	Y	FK

Entity: MovieDirector

This table contains information about the movie’s director.

Column	Date Type	Description	Range	Required	PK or FK

DirectorCode	INT	Unique identifier for the “MovieDirector” entity	00001-99999	Y	PK
DirectorFirstName	VARCHAR	Movie director’s first name		Y	
DirectorLastName	VARCHAR	Movie director’s last name		Y	

Entity: MovieAward

This table contains information about any awards the movie has won.

Column	Date Type	Description	Range	Required	PK or FK
AwardCode	INT	Unique identifier for the “MovieAward”	00001-99999	Y	PK
DateAwarded	DATE	Date of the award		Y	
MovieCode	INT	Unique code of each movie	00001-99999	Y	FK

Entity: Award

This table contains the names of the awards each movie has won.

Column	Date Type	Description	Range	Required	PK or FK
AwardCode	INT	Unique identifier for	00001-99999	Y	PK

		the “Award”			
AwardName	**VARCHAR**	**Name of the award**		**Y**	

Entity: StreamingServicePlan

This table contains information on the streaming service plan the customer chose.

Column	Date Type	Description	Range	Required	PK or FK
PlanCode	**INT**	**Primary key of “StreamingServicePlan” entity**	**00001-99999**	**Y**	**PK**
PlanDescription	**VARCHAR**	**A brief description of each service plan available on the platform**		**Y**	
PricePerMonth	**INT**	**Monthly subscription price for each plan**	**00001-99999**	**Y**	

Entity: StreamingServiceTransaction

This table contains information on the streaming service plan the customer chose.

Column	Date Type	Description	Range	Required	PK or FK
TransactionCode	**INT**	**Code generated every time the customer makes a transaction**	**00001-99999**	**Y**	**PK**

TotalAmount	**INT**	**Total amount of the transaction**	**00001-99999**	**Y**	
PlanCode	**INT**	**Type of plan the customer is subscribed to**	**00001-99999**	**Y**	**FK**
AccountCode	**INT**	**Member's unique account code**	**00001-99999**	**Y**	**FK**

Entity: DVDStore

This table contains information about the individual physical locations of DVD stores.

Column	Date Type	Description	Range	Required	PK or FK
StoreCode	**INT**	**Primary key of "DVDStore" entity**	**00001-99999**	**Y**	**PK**
StoreName	**VARCHAR**	**Name of the store**		**Y**	
StoreAddress	**VARCHAR**	**Address of the store**		**Y**	
StorePhone	**INT**	**Store's phone number**	**00001-99999**	**Y**	

Entity: Account

This table contains personal information about the primary account holder.

Column	Date Type	Description	Range	Required	PK or FK

AccountCode	INT	Primary key of the “Account” entity	00001-99999	Y	PK
Email	VARCHAR	Customer’s email address		Y	
Password	VARCHAR	Customer’s account password		Y	
BirthDate	DATE	Customer’s date of birth		N	
BankRouting	INT	Bank account routing number	00001-99999	N	
Address	VARCHAR	Customer’s address		Y	

Entity: StoreTransaction

This table contains information about the transactions the customer has made in store.

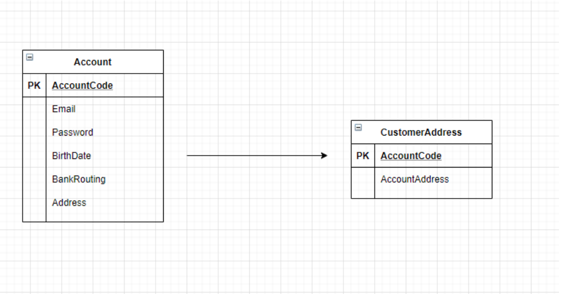
Column	Date Type	Description	Range	Required	PK or FK
TransactionCode	INT	Unique identifier of “StoreTransaction” entity	00001-99999	Y	PK
TransactionDate	DATE	Date of the transaction		Y	
TotalAmount	INT	Total amount of	00001-	Y	

		the transaction	99999		
PaymentMethod	VARCHAR	Type of method of payment. e.g. Visa, Mastercard, American Express		Y	
AccountCode	INT	Member’s unique account code	00001-99999	Y	FK

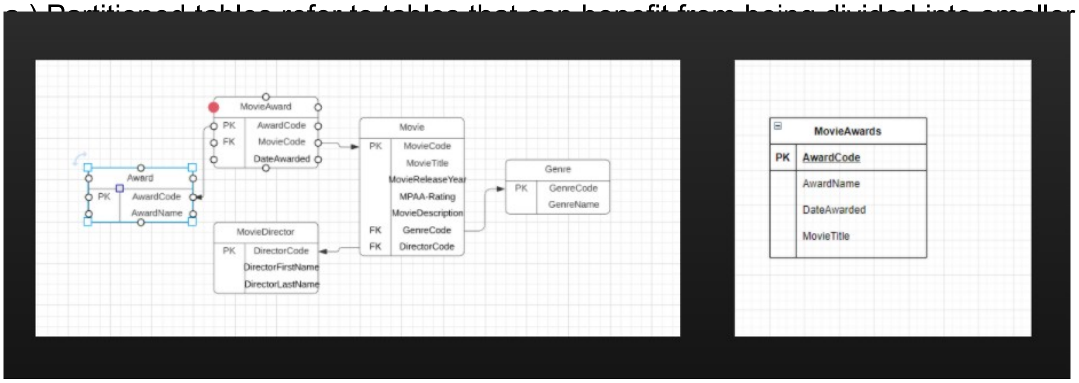
The purpose of the database:

As mentioned in the introduction, the growing selection of movies and the number of members subscribing to streaming services pose a big challenge for many streaming and video rental companies, which means that the level of service they provide is falling. Therefore, we would need to create a database solution that will meet our business needs to automate and speed up a lot of the day-to-day tasks that will otherwise take forever to complete if done manually. The aim of creating a database solution is to develop a database application that will help solve the increasing problems of data management by facilitating efficient data storage and retrieval. For that to be successful, we need to make sure that we create a database that will correctly store and retrieve data efficiently.

D.Physical Design



a.) Partitioned tables refer to tables that can benefit from being divided into smaller segments;



this can make managing complex data more simple and improve query performance. One useful partitioned table that could be made for our case is from the Account table. The Account table has personal customer information that includes the home address. We could do a vertical partition and make a table with just the AccountCode and Address and use this table for queries related to finding information on where your customers are located. Denormalization refers to the practice of taking a normalized table and undoing the normalization to enhance query performance. If you are trying to find a query relating to the movie awards, then you could denormalize the award table and into a table that has AwardCode, AwardName, DateAwarded, MovieTitle. A normalized table can make the query process more complex since you would have to grab the attributes from 3 different tables as opposed to just 1.

b.) Since for our case we are designing a database for a video streaming/renting platform, we need for our objects to have access to functions so that our tables have relevant information. In our case we need to ensure our sales department has up to date information, our subject would be the Sales Department and our object would be StoreTransaction. The action our object is allowed to do is insert so that our StoreTransaction table is current. Since this table includes our regular customers, we will have a total amount limit set to whatever the max amount a member can be charged with. For our Accounting Department we would use the Account object and allow for the action of read and have a password constraint as well so that only authorized users have access to the table.

c.) Making sure each attribute has its proper data type is essential for creating an efficient database. Our case requires that we use 3 different data types. For any attribute that requires a number (such as AccountCode, TotalAmount, and BrankRouting) we use the INT data type since they are made up of numbers. For attributes relating to date we used the DATE data type so that it maintains the organization and readability of our information. Lastly we used the VARCHAR data type for any attributes relating to names, addresses or payment types. We had to use the VARCHAR so that we could allow for the variability of responses.

d.) Our database, depending on how many customers we are able to acquire, could have millions of records. Due to the size of our database, we have chosen to go with the Indexed File method. Indexed file organization refers to a file in which each record has a record key that uniquely identifies. Each record will contain a field for a record key; for example, if I wanted to know the director's name, I just need the DirectorCode. The indexed method enables our queries to extract the records they need more efficiently..

e.) A database recovery policy refers to a set of guidelines an organization follows to ensure that there is a mechanism/procedure in place to restore a database loss or repair database damage. To ensure that we always have access to our data even if we are not connected to a cloud, we will utilize Back-Facilities. Since our data is always being updated, we will need to have periodic backups (once a night). We will also have to utilize Hot Backups since we have different tables that have no relation to each other. Our backups will be stored in a secure, off-site location. We plan to practice journalizing all our transactions and keeping essential data related to a transaction in a Transaction log. We will also have a database change log to monitor all our tables. In case of Database failure, we will utilize Recovery Manager for cases that are not very serious. We will use the Roll forward method if we need to redo the changes made by a transaction and use the Roll back method if we are trying to revert a transaction to its original state. If we have a serious malfunction/failure then we can utilize Disk Mirroring and if it is very serious then we have to use Restore/Rerun. We will also make sure our employees are aware of tools available to them, and are aware of our Database Recovery policy, through training.

E.SQL Queries

Now that we have established the actual design of the database, in order to make it function, we will need to implement the database into SQL, for this particular application, the flavour of SQL that we chose is MySQL. The first step in implementing the database into MySQL is to create the tables, which come from the entities shown in the design, and to establish the data type that those columns will hold. Our whole database design consists of 13 entities or classes, that means we will need to create 13 tables in MySQL with the correct data types occupying each column. When we are creating our tables, it is important to also establish the relationship that an entity might have with another entity, which is portrayed by primary keys and foreign keys.

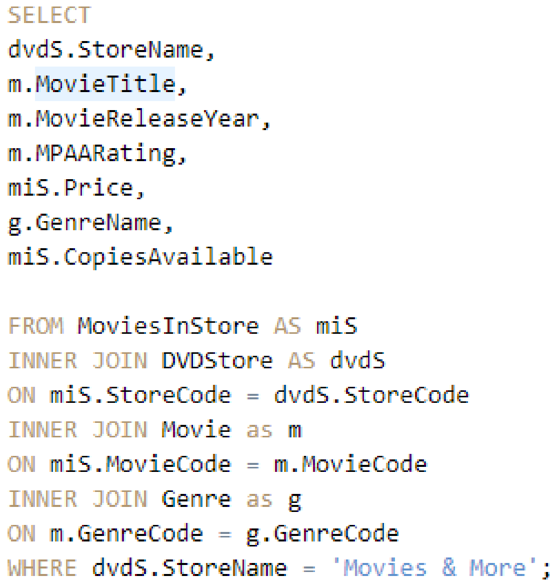
The next step in implementing the actual database into MySQL would have to be inserting dummy data, data that does not contain any real information. For this project, we just want to be able to test the functionality of our database in MySQL, so it is not very important to us if the data is inaccurate or not timely. However, if this application were to launch in a real business, having data and information that is timely, accurate, and relevant to its time is extremely important. Once we have finally inserted our dummy data, we can now come up with a variety of different queries to fetch certain things that are expected from our database and perform common tasks that one customer or user might do.

Some of the expected tasks that our database will be required to perform consist of a handful of queries. A couple of these queries are the following: one of them allows the user to browse all movies by a certain director and another one is being able to view all movies that a certain DVD store has. Having these types of queries will be the pinnacle of our database and they will be most commonly used by customers and users of our application. A wide variety of SQL queries will be showcased below; we will display the actual SQL code and then the outcome of running the query.

The queries that will be showcased for this project are the following:

1) Browse all movies available in a selected DVD store
2) Browse all movies that were directed by a specific person
3) Browse all movies that have been given an award
4) Browse movies by a specific genre
5) Sort movies by MPAA rating
6) Retrieve all movies based on the year of release

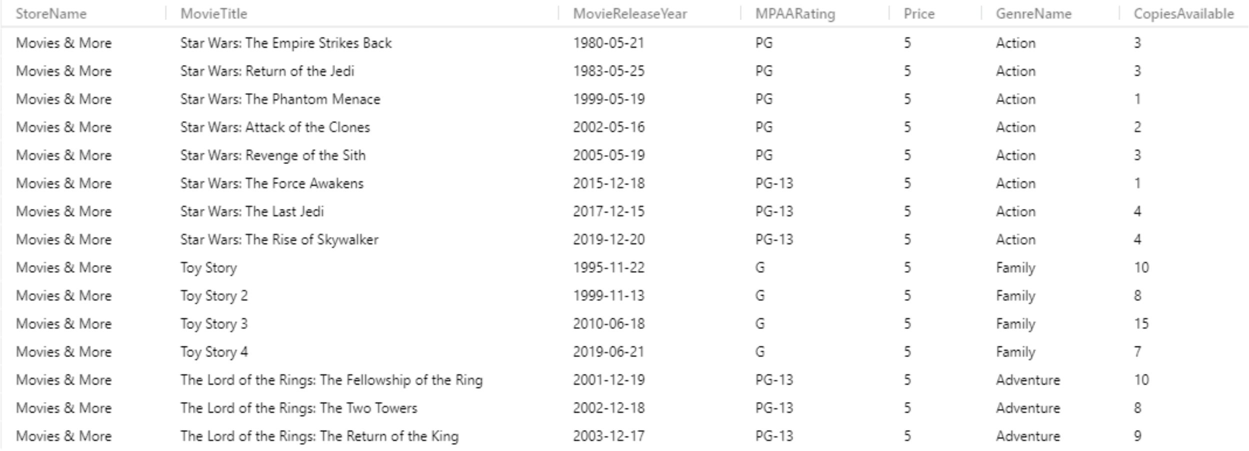
Query #1:
Browse all movies available in a selected DVD store



```
SELECT
dvdS.StoreName,
m.MovieTitle,
m.MovieReleaseYear,
m.MPAARating,
miS.Price,
g.GenreName,
miS.CopiesAvailable

FROM MoviesInStore AS miS
INNER JOIN DVDStore AS dvdS
ON miS.StoreCode = dvdS.StoreCode
INNER JOIN Movie as m
ON miS.MovieCode = m.MovieCode
INNER JOIN Genre as g
ON m.GenreCode = g.GenreCode
WHERE dvdS.StoreName = 'Movies & More';
```

Outcome of running Query #1:

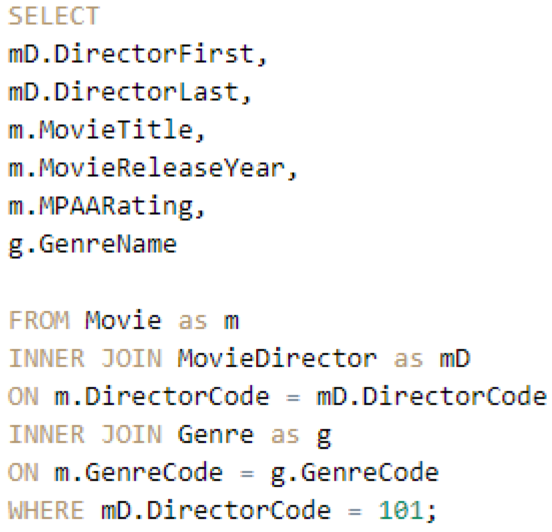


StoreName	MovieTitle	MovieReleaseYear	MPAARating	Price	GenreName	CopiesAvailable
Movies & More	Star Wars: The Empire Strikes Back	1980-05-21	PG	5	Action	3
Movies & More	Star Wars: Return of the Jedi	1983-05-25	PG	5	Action	3
Movies & More	Star Wars: The Phantom Menace	1999-05-19	PG	5	Action	1
Movies & More	Star Wars: Attack of the Clones	2002-05-16	PG	5	Action	2
Movies & More	Star Wars: Revenge of the Sith	2005-05-19	PG	5	Action	3
Movies & More	Star Wars: The Force Awakens	2015-12-18	PG-13	5	Action	1
Movies & More	Star Wars: The Last Jedi	2017-12-15	PG-13	5	Action	4
Movies & More	Star Wars: The Rise of Skywalker	2019-12-20	PG-13	5	Action	4
Movies & More	Toy Story	1995-11-22	G	5	Family	10
Movies & More	Toy Story 2	1999-11-13	G	5	Family	8
Movies & More	Toy Story 3	2010-06-18	G	5	Family	15
Movies & More	Toy Story 4	2019-06-21	G	5	Family	7
Movies & More	The Lord of the Rings: The Fellowship of the Ring	2001-12-19	PG-13	5	Adventure	10
Movies & More	The Lord of the Rings: The Two Towers	2002-12-18	PG-13	5	Adventure	8
Movies & More	The Lord of the Rings: The Return of the King	2003-12-17	PG-13	5	Adventure	9

As you can see, this particular query fetches us all movies that are available in a selected DVD store, it tells us what the price of each movie is, what genre the movie belongs to, and how many copies are available in the store. If we want to check other DVD stores, we can simply do that by changing the string in the WHERE clause. For example, we can replace “Movies & More” by another store called “DVDs4Less”, and once we run the query, it should fetch us all types of movies that the store, DVDs4Less, has.

Query #2:

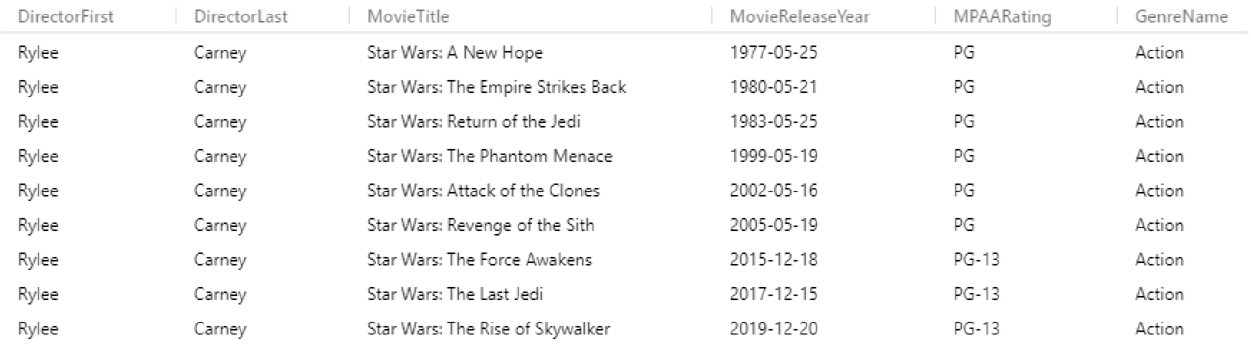
Browse all movies that were directed by a specific person



```
SELECT
mD.DirectorFirst,
mD.DirectorLast,
m.MovieTitle,
m.MovieReleaseYear,
m.MPAARating,
g.GenreName

FROM Movie as m
INNER JOIN MovieDirector as mD
ON m.DirectorCode = mD.DirectorCode
INNER JOIN Genre as g
ON m.GenreCode = g.GenreCode
WHERE mD.DirectorCode = 101;
```

Outcome of running Query #2:

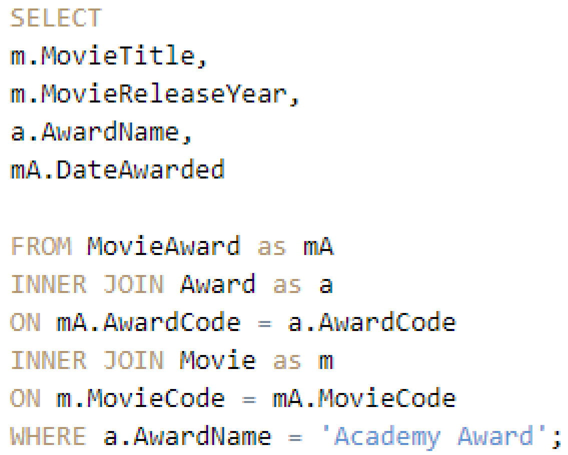


DirectorFirst	DirectorLast	MovieTitle	MovieReleaseYear	MPAARating	GenreName
Rylee	Carney	Star Wars: A New Hope	1977-05-25	PG	Action
Rylee	Carney	Star Wars: The Empire Strikes Back	1980-05-21	PG	Action
Rylee	Carney	Star Wars: Return of the Jedi	1983-05-25	PG	Action
Rylee	Carney	Star Wars: The Phantom Menace	1999-05-19	PG	Action
Rylee	Carney	Star Wars: Attack of the Clones	2002-05-16	PG	Action
Rylee	Carney	Star Wars: Revenge of the Sith	2005-05-19	PG	Action
Rylee	Carney	Star Wars: The Force Awakens	2015-12-18	PG-13	Action
Rylee	Carney	Star Wars: The Last Jedi	2017-12-15	PG-13	Action
Rylee	Carney	Star Wars: The Rise of Skywalker	2019-12-20	PG-13	Action

This query is designed for customers so that they can limit their movie searches that were directed by a specific person. As a result of running this query, we get back the director's name, the title of the movie they directed, the release year of the movie, and etc. Again, if we wanted to view other movies created by other directors, we can simply change the integer value in the WHERE clause.

Query #3:

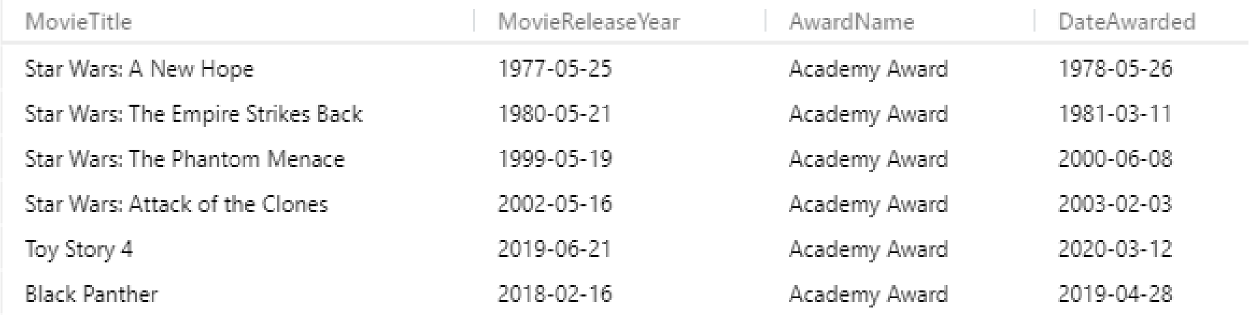
Browse all movies that have been given a specific award



```
SELECT
m.MovieTitle,
m.MovieReleaseYear,
a.AwardName,
mA.DateAwarded

FROM MovieAward as mA
INNER JOIN Award as a
ON mA.AwardCode = a.AwardCode
INNER JOIN Movie as m
ON m.MovieCode = mA.MovieCode
WHERE a.AwardName = 'Academy Award';
```

Outcome of running Query #3:

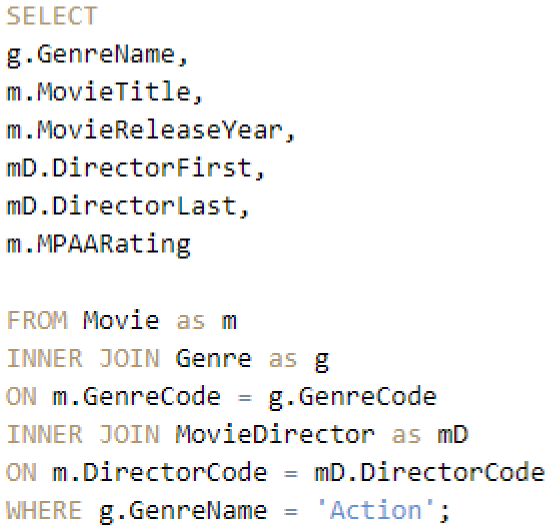


MovieTitle	MovieReleaseYear	AwardName	DateAwarded
Star Wars: A New Hope	1977-05-25	Academy Award	1978-05-26
Star Wars: The Empire Strikes Back	1980-05-21	Academy Award	1981-03-11
Star Wars: The Phantom Menace	1999-05-19	Academy Award	2000-06-08
Star Wars: Attack of the Clones	2002-05-16	Academy Award	2003-02-03
Toy Story 4	2019-06-21	Academy Award	2020-03-12
Black Panther	2018-02-16	Academy Award	2019-04-28

This query allows users and customers to filter movies based on the awards that they've won. It fetches the title of the movie, the release year of the movie, the name of the award, and the date it has won that particular award. So in this example, we want to retrieve all movies that have won an Academy award, also known as an Oscar award. We can also filter by different types of awards, like before, all we need to do is change the string to a name of a different award in the WHERE clause.

Query #4:

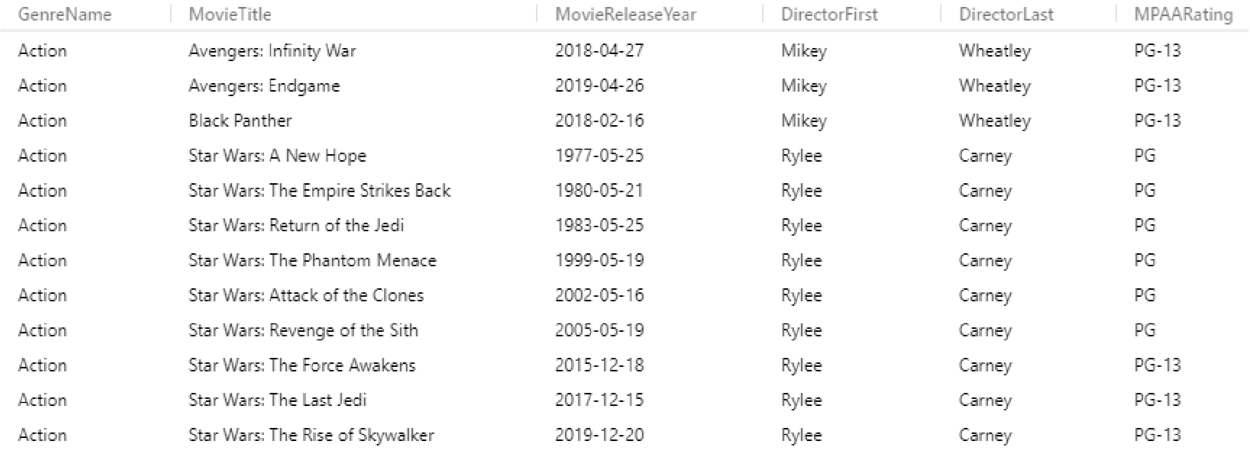
Browse movies by a specific genre



```
SELECT
g.GenreName,
m.MovieTitle,
m.MovieReleaseYear,
mD.DirectorFirst,
mD.DirectorLast,
m.MPAARating

FROM Movie as m
INNER JOIN Genre as g
ON m.GenreCode = g.GenreCode
INNER JOIN MovieDirector as mD
ON m.DirectorCode = mD.DirectorCode
WHERE g.GenreName = 'Action';
```

Outcome of running Query #4:

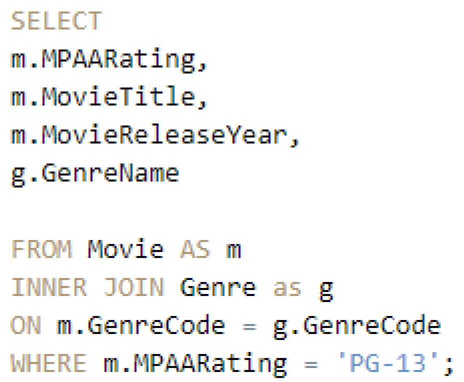


GenreName	MovieTitle	MovieReleaseYear	DirectorFirst	DirectorLast	MPAARating
Action	Avengers: Infinity War	2018-04-27	Mikey	Wheatley	PG-13
Action	Avengers: Endgame	2019-04-26	Mikey	Wheatley	PG-13
Action	Black Panther	2018-02-16	Mikey	Wheatley	PG-13
Action	Star Wars: A New Hope	1977-05-25	Rylee	Carney	PG
Action	Star Wars: The Empire Strikes Back	1980-05-21	Rylee	Carney	PG
Action	Star Wars: Return of the Jedi	1983-05-25	Rylee	Carney	PG
Action	Star Wars: The Phantom Menace	1999-05-19	Rylee	Carney	PG
Action	Star Wars: Attack of the Clones	2002-05-16	Rylee	Carney	PG
Action	Star Wars: Revenge of the Sith	2005-05-19	Rylee	Carney	PG
Action	Star Wars: The Force Awakens	2015-12-18	Rylee	Carney	PG-13
Action	Star Wars: The Last Jedi	2017-12-15	Rylee	Carney	PG-13
Action	Star Wars: The Rise of Skywalker	2019-12-20	Rylee	Carney	PG-13

This query allows customers to browse all movies based on a particular genre. If a user wants to sort by action or adventure movies they can; if we were to change the string in the WHERE clause to “Adventure” or “Fantasy” and run the query, it would fetch us all movies that are within a certain category/genre.

Query #5:

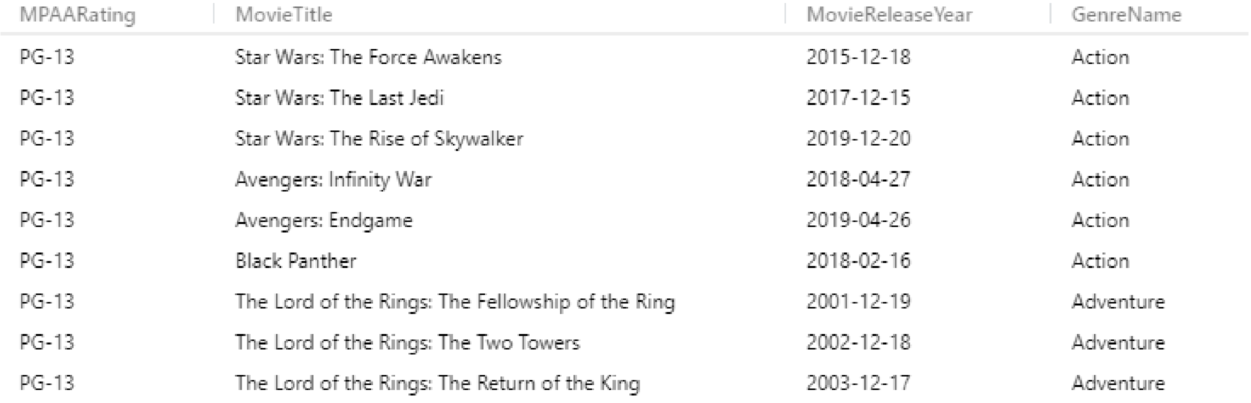
Sort movies by MPAA rating



```
SELECT
m.MPAARating,
m.MovieTitle,
m.MovieReleaseYear,
g.GenreName

FROM Movie AS m
INNER JOIN Genre as g
ON m.GenreCode = g.GenreCode
WHERE m.MPAARating = 'PG-13';
```

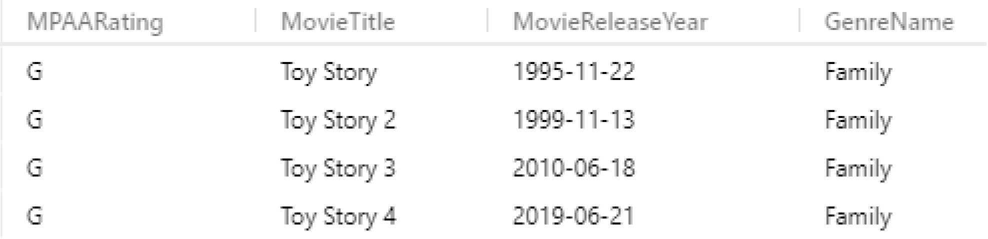
Outcome of running Query #5:



MPAARating	MovieTitle	MovieReleaseYear	GenreName
PG-13	Star Wars: The Force Awakens	2015-12-18	Action
PG-13	Star Wars: The Last Jedi	2017-12-15	Action
PG-13	Star Wars: The Rise of Skywalker	2019-12-20	Action
PG-13	Avengers: Infinity War	2018-04-27	Action
PG-13	Avengers: Endgame	2019-04-26	Action
PG-13	Black Panther	2018-02-16	Action
PG-13	The Lord of the Rings: The Fellowship of the Ring	2001-12-19	Adventure
PG-13	The Lord of the Rings: The Two Towers	2002-12-18	Adventure
PG-13	The Lord of the Rings: The Return of the King	2003-12-17	Adventure

This is a short and simple query and it allows customers to browse all movies depending on the specific MPAA (Motion Picture Association of America) rating. In the images listed above, the query fetches movies that are rated PG-13, if for example we want movies that are G rated, we can change that if we alter the string in the WHERE clause; simply change “PG-13” to “G” and the query will retrieve all movies that are G rated, shown below.

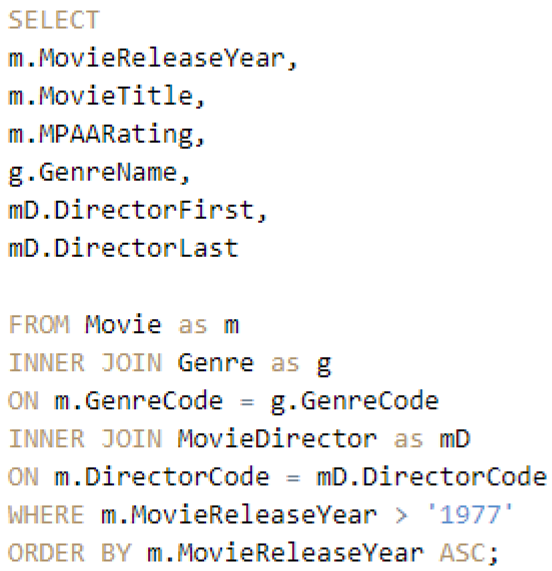
Outcome of running Query #5 (WHERE m.MPAARating = ‘G’;)



MPAARating	MovieTitle	MovieReleaseYear	GenreName
G	Toy Story	1995-11-22	Family
G	Toy Story 2	1999-11-13	Family
G	Toy Story 3	2010-06-18	Family
G	Toy Story 4	2019-06-21	Family

Query #6:

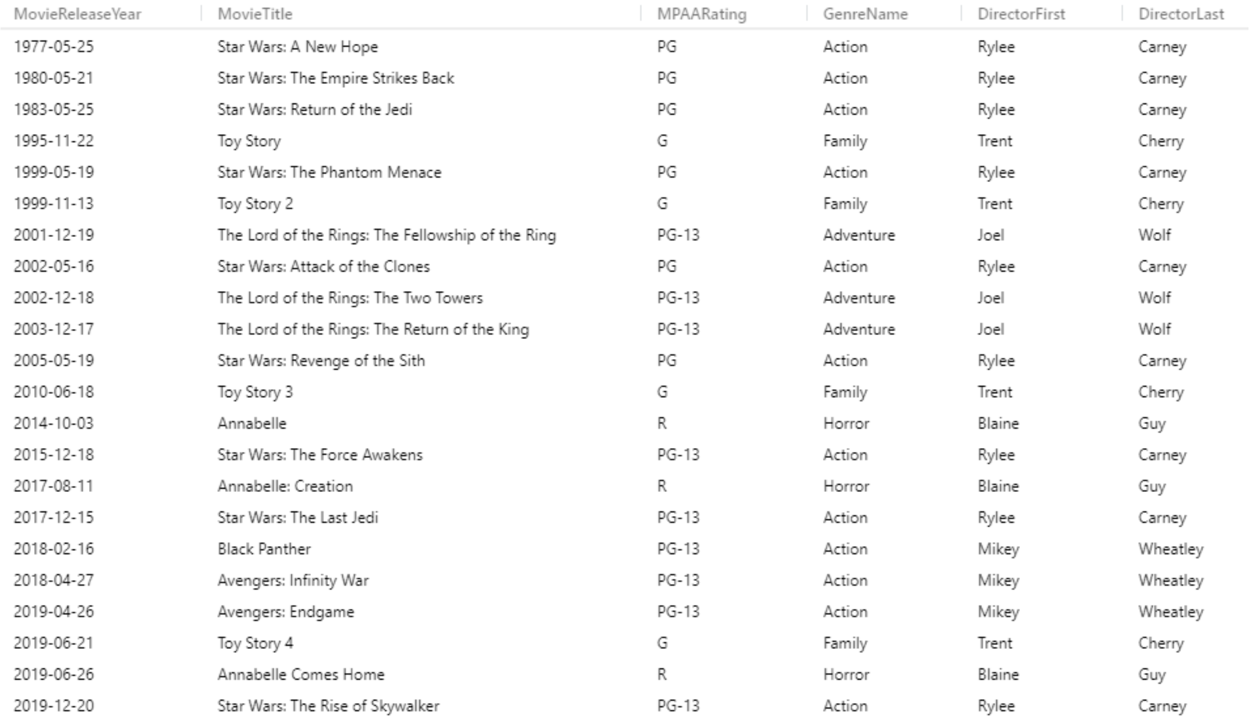
Retrieve all movies based on their year of release



```
SELECT
m.MovieReleaseYear,
m.MovieTitle,
m.MPAARating,
g.GenreName,
mD.DirectorFirst,
mD.DirectorLast

FROM Movie as m
INNER JOIN Genre as g
ON m.GenreCode = g.GenreCode
INNER JOIN MovieDirector as mD
ON m.DirectorCode = mD.DirectorCode
WHERE m.MovieReleaseYear > '1977'
ORDER BY m.MovieReleaseYear ASC;
```

Outcome of running Query #6:



MovieReleaseYear	MovieTitle	MPAARating	GenreName	DirectorFirst	DirectorLast
1977-05-25	Star Wars: A New Hope	PG	Action	Rylee	Carney
1980-05-21	Star Wars: The Empire Strikes Back	PG	Action	Rylee	Carney
1983-05-25	Star Wars: Return of the Jedi	PG	Action	Rylee	Carney
1995-11-22	Toy Story	G	Family	Trent	Cherry
1999-05-19	Star Wars: The Phantom Menace	PG	Action	Rylee	Carney
1999-11-13	Toy Story 2	G	Family	Trent	Cherry
2001-12-19	The Lord of the Rings: The Fellowship of the Ring	PG-13	Adventure	Joel	Wolf
2002-05-16	Star Wars: Attack of the Clones	PG	Action	Rylee	Carney
2002-12-18	The Lord of the Rings: The Two Towers	PG-13	Adventure	Joel	Wolf
2003-12-17	The Lord of the Rings: The Return of the King	PG-13	Adventure	Joel	Wolf
2005-05-19	Star Wars: Revenge of the Sith	PG	Action	Rylee	Carney
2010-06-18	Toy Story 3	G	Family	Trent	Cherry
2014-10-03	Annabelle	R	Horror	Blaine	Guy
2015-12-18	Star Wars: The Force Awakens	PG-13	Action	Rylee	Carney
2017-08-11	Annabelle: Creation	R	Horror	Blaine	Guy
2017-12-15	Star Wars: The Last Jedi	PG-13	Action	Rylee	Carney
2018-02-16	Black Panther	PG-13	Action	Mikey	Wheatley
2018-04-27	Avengers: Infinity War	PG-13	Action	Mikey	Wheatley
2019-04-26	Avengers: Endgame	PG-13	Action	Mikey	Wheatley
2019-06-21	Toy Story 4	G	Family	Trent	Cherry
2019-06-26	Annabelle Comes Home	R	Horror	Blaine	Guy
2019-12-20	Star Wars: The Rise of Skywalker	PG-13	Action	Rylee	Carney

This query returns all movies that were released in a specific year. We can control what the query returns by altering the string in the WHERE clause; we can have the query retrieve movies that were released in a year or within a range of years. This query will allow users to sort movies by oldest to newest or by newest to oldest.