COMP40725 Introduction to Relational Databases & SQL Programming: Assignment 1

Database Design & Implementation for DVD Store

By Noreen A. Lenihan

Semester 2
Dr. Mark Scanlon
Deadline date: 26.4.2014

Table of Contents

Noreen Lenihan

COMP40725 Assignment
Dr. Mark Scanlon

COMP40725 Introduction to Relational Databases & SQL Programming: Assignment 1

Database Design & Implementation for DVD Store
By Noreen A. Lenihan

Q1. Developing the Entity-Relationship Model

Prior to developing the graphical representation of the database in our design phase (ER modeling), it is always imperative to document any assumptions we are making in the process. In the present case, these assumptions are predicated on our understanding of the DVD Store's business rules. These are clearly stated here:

Assumptions

- Only registered members of the DVD store are eligible to rent DVDs (that are indeed, rentable, and not for sale)
- The duration of a rental lasts for only one night. However, human nature precludes this rule, and thus rentals may not be returned the next day, in which case, a surcharge for being overdue is applied to the member in question. These surcharges should be accounted for together with normal price of renting the movie.
- Customers should be accommodated when making inquiries about a DVD which stars a preferred actor, was directed by a particular director, or showcases a certain character. This should be reflected in the DB design. Further, customer queries that aim to identify the character played by a specified actor will also be catered to, to help pinpoint the character and DVD desired.
- One character (e.g. Jane Eyre) can be played by my actors in a film, that is, has various actors playing them at various life stages (e.g. Jane Eyre as a young girl, Jane Eyre as a governess). For the purposes of this database design, and completeness, it is

assumed that a query into the character played in a certain film, will return the name of all actors playing the character, attached to the life stage/change such that the actor names of, for example, 'Jane Eyre (young girl)', and Jane Eyre ('Governess') will be returned for such a query.

- One actor can also play multiple characters in a film. An easy example is the movie, 'Big Momma', in which Eddie Murphy plays multiple roles, e.g. Big Momma herself, and the young man in courtship.
- A rental's price is variable, according to the popularity/frequency of renting of that DVD. Thus, the database should enable the user to keep track of and modify the current rental price of the rental, as well as being able to store all of the prices it was every rented for (i.e. a history of independent rental transactions). This is necessary for tracking income from all rentals, also.
- The 'commercial status' of a DVD, that is, whether it is available for renting or sale, should be allowed to be modified in the DB. This owes to the fact that a rental may be outdated or no longer popular for renting. Accordingly, a copy of a DVD should have a document date of being added to inventory so that staff can deduce how long a DVD/copy has been rented for. For example, management might decree that after 6 years from being added to the system, all DVDs that did not achieve sales of more than X euro in the past year, should be now categorized as an exrental, and available for sale. **It is important to note that a DVD will not be for sale and for rent at the same time**. Intuitively, an exrental can be sold to registered members and/or regular members of the public.
- Once an exrental has been sold, its 'availability' status should be changed to 'not available', or an equivalent, to reflect that it cannot undergo any more transactions, though its sale will still be recorded in the DB.
- A DVD can only ever be classified as for 'rent' or for 'sale' at any one time.
- Information about a rental's current availability and price status should be updated whenever a member takes out a rental. Following this logic, a rental's availability status should be reset to 'Available' or the equivalent when it is returned by the member.
- Each DVD has many copies. For example, the film, 'Hunger Games' will have many copies available for rent so that multiple customers can take out the DVD at the same time. Each copy is a physical entity, (a type of DVD) and thus requires information to be stored about each instance of the DVD.
- It is noteworthy to mention that a copy refers to a single instance of a DVD disk, e.g. 'American Pie 3', OR a collection of DVDs that are required to be sold together as

dictated by law. To illustrate, a boxset of ‘Friends: Season 1 – 3’, will be sold, as an exrental, together or as a unit, and documented as one single copy.

- An important note regarding actors for animated movies is that animated movies that contain the voices of popular actors will be declared as having actors that ‘star’ in that movie, even if only they are the voiceovers. An example might be Cameron Diaz and Eddie Murphy as starring in Shrek, where they play the voice of Princess Fiona and Donkey, respectively.
- Finally, a DVD movie’s title is not unique. IMBD list up to 20 movies with the same title, e.g. ‘Hunted’. Though this mostly occurs with lesser known movies, nevertheless it is important to distinguish that a movie cannot be identified uniquely by its title.
- For the purposes of this database, a film will be considered to have only one genre. Where there is divided opinion between two genres for a film, the film will be associated with the globally more dominant one.

It is now timely to begin the recursive process of developing our ER model.

Step 1: Identifying Entities

In this section, we will list all potential and preliminary entity types, based on the client’s specification.

- DVD rental shop
- Customer
- Member
- Name
- Address
- Telephone Number
- Membership Number
- DVD
- Rental
- Film details
- Film name
- Film year
- Director
- Actor
- Price
- Copy

- Character
- Ex-rental
- Non-member
- Revenue

Step 2: Remove Duplicate Entities

From the list of possible entities as denoted in Step 1, we must now eliminate duplicate entities, entities that are not relevant to the database goals, or nouns that depict the system we are working on. Explanations are provided for retaining/removing each attribute.

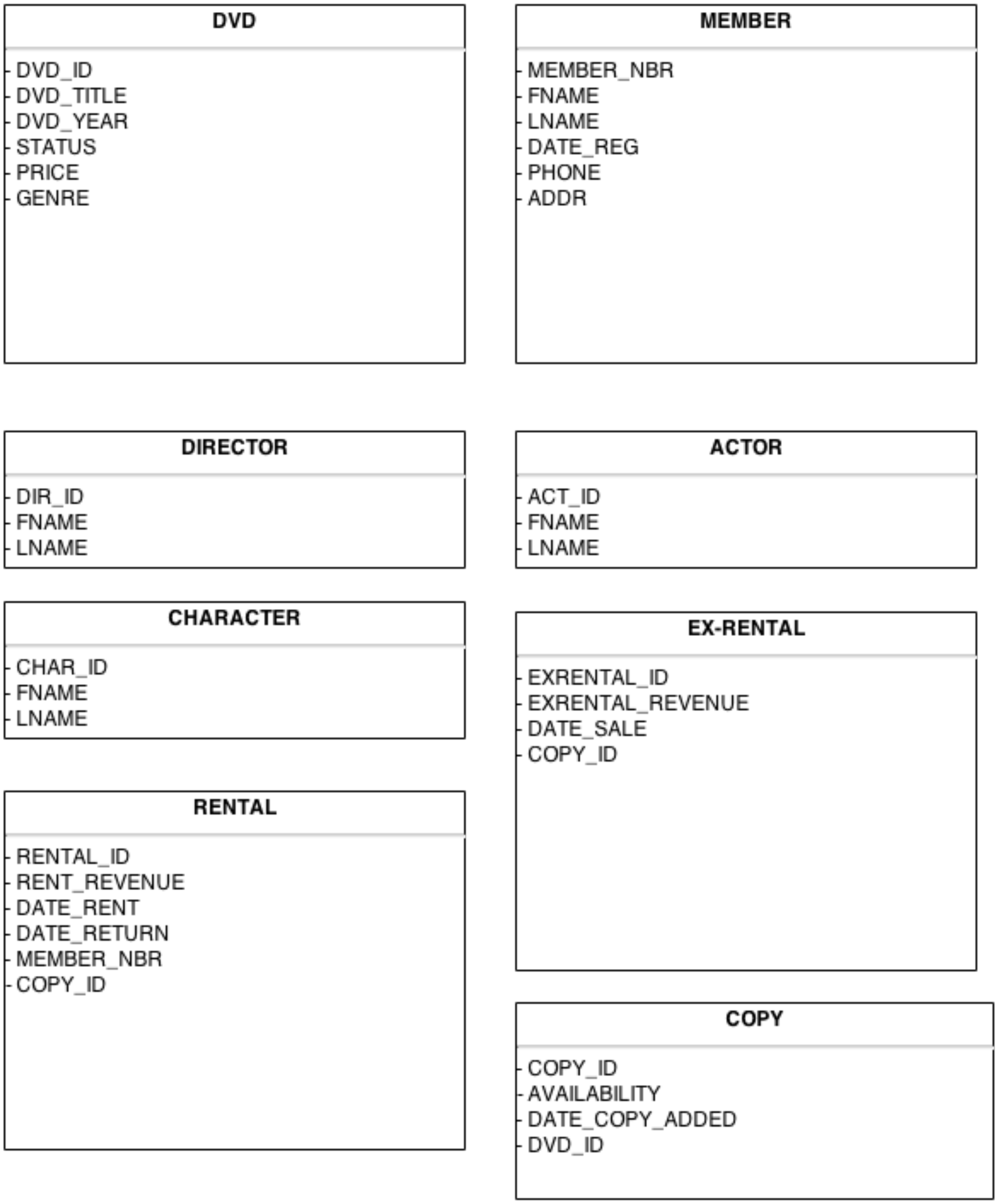
- ~~DVD rental shop~~ – This is the system we are aiming to map, and this will not be included in our ER model
- ~~Customer~~ – Unnecessary entity. Since we will note purchases of exrentals from members of the public, all we care about is the history of our registered members.
- Member – This candidate entity will be retained as it maps a real-world entity that has several of its own attributes
- ~~Name~~ – Discarded. Reason: Attribute of 'Member'
- ~~Address~~ – Discarded. Reason: Attribute of 'Member'
- ~~Telephone Number~~ – Discarded. Attribute of 'Member'
- ~~Membership Number~~ – Discarded. Reason: Not an entity in its own right. Rather, it may serve as the attribute/unique identifier/primary key for 'Member' entity
- DVD – Retained as an entity as it is necessary to store various details about the DVD whereby DVD is meant to 'encapsulate' everything about a film
- Rental – A rental will be kept as an entity as it will used to store information about the rental fee of a movie at a particular period, the member who took out a rental, the copy number of the DVD, the date it was taken out/returned, etc.
- ~~Film details~~ – Discarded. Consists of a host of details about the DVD that iscontained within the DVD entity
- ~~Film name~~ – Discarded. Reason: Attribute of DVD which stores movie details
- ~~Film year~~ – Discarded. Reason: Attribute of DVD
- Director – Retained. This is a separate entity that contains basic information about a director, and also a link to DVD entity.
- Actor – Retained. This should be afforded its own entity to store information about the actor and also to facilitate linking with DVD.

- ~~Price~~ – Discarded. This can be thought of as attribute that is dependent on each instance (tuple) of a rental and/or exrental.
- Copy – This should be an entity itself that is distinct from DVD, as each DVD (or movie) will have multiple copies available for rent at any one time. Thus, we will need to record information about its availability, the date the copy was added to inventory, and what DVD movie it is a copy of.
- Character – Film character warrants an entity as name will have to be stored for each character, as well as a link to each relevant DVD
- Ex-rentals – Since information on the sale price and DVD title will need to be recorded per entry, 'Exrental' elicits its own entity
- ~~Non-members~~ – Discarded. No information needs to be stored about members of the public who make a purchase of an exrental.
- ~~Revenue~~ – Discarded. Since a DVD rental will have varying prices over time, it is entirely necessary to keep a 'current rental price' associated with every DVD. Each instance of a rental, however, will have a column that associates a price/fee with that particular rental. Total revenue received from rentals can be attained by using a sum function on the rental table, with conditions (such as over a certain period), as necessary.

Step 3: Listing Attributes of Entities

Below is a listing of all the attributes necessary for each of our entities. On the next page, these attributes are listed diagrammatically (and using condensed naming conventions for our relational model) using draw.io*.

1. Member (member number, first name, surname, date of registration, phone, address)
2. DVD (DVD ID, title of movie, year of release, commercial status, price, genre)
3. Rental (Rental ID, rental revenue, date rented, date returned, member number, copy ID)
4. Director (Director ID, first name, surname)
5. Actor (Actor ID, first name, surname)
6. Copy (Copy ID, availability, date copy was added, DVD ID)
7. Character (Character ID, first name, surname)
8. Exrental (Exrental ID, exrental revenue, date of sale, copy ID)


DVD
- DVD_ID
- DVD_TITLE
- DVD_YEAR
- STATUS
- PRICE
- GENRE
MEMBER
- MEMBER_NBR
- FNAME
- LNAME
- DATE_REG
- PHONE
- ADDR
DIRECTOR
- DIR_ID
- FNAME
- LNAME
ACTOR
- ACT_ID
- FNAME
- LNAME
CHARACTER
- CHAR_ID
- FNAME
- LNAME
EX-RENTAL
- EXRENTAL_ID
- EXRENTAL_REVENUE
- DATE_SALE
- COPY_ID
RENTAL
- RENTAL_ID
- RENT_REVENUE
- DATE_RENT
- DATE_RETURN
- MEMBER_NBR
- COPY_ID
COPY
- COPY_ID
- AVAILABILITY
- DATE_COPY_ADDED
- DVD_ID

More descriptively, each of the entities requires the following attributes. The information in parentheses represents the hypothesized attribute name when mapped to a relational model.

1. Member

- ➢ Member number (member_nbr) – To assign each registered member of the DVD store a unique number
- ➢ Firstname (fname) and Surname (lname) – To record basic information about member's full name
- ➢ Date registered (date_reg) – All organizations require information of when a member joined with them, to verify registration took place.
- ➢ Phone (phone) and Address (addr) – Every member must provide contact information to facilitate correspondence between the store and the customer (e.g. for notification of overdue rentals, discounts/offers, etc.)

2. DVD

- ➢ DVD ID (dvd_id) – This is a unique identifier for each DVD, where DVD refers to the actual film and so it can be thought of as a 'film identifier number'
- ➢ Title (dvd_title) – Records the actual name of the film
- ➢ Year (dvd_year) – Records the year of release of film
- ➢ Status (status) – Status, in this context, refers to the type of transaction that the DVD is available for, that is, it is either 'for rent' or 'for sale'. It cannot be have both statuses at the same time (Business rule. See 'Assumptions').
- ➢ Price (price) - Price, in this sense, refers to either the current sales price of an exrental OR the current rental price of a DVD rental. The latter is an important consideration as we previously noted that a DVD rental's price could change over time. Thus, the price property, as defined in the DVD entity, will always refer to the 'current rental price' since the last update.
- ➢ Genre (genre) - Genre, as the name implies, simply concludes the genre of the film in question. Invariably, each film belongs to a genre, and thus it can be an attribute of film. Genre examples include comedy, romance, sci-fi, documentary, epic drama, action, psychological thriller.

3. Rental

A rental is operationally defined as a record of a DVD copy that is deemed rentable and is rented by a registered member. With that in mind, we can take note of the distinct and requisite attributes of rentals.

- Rental ID (rental_id) – Each rental will need a unique identifier associated with that rental.
- Rental Revenue/Fee (rental_revenue) – Each record of a rental will need its associated rental fee that is specific to that rental (the price of that DVD for renting **at that point in time**)
- Date rented (date_rent) – A record of the date that this rental transaction took place by a registered member
- Date returned (date_return) – A record of the date that the member returned the DVD rental
- Member Number (member_nbr) – We must document the member that rented a particular rental
- Copy ID (copy_id) – It is important to document, not the film (DVD) that the member rented, but the exact copy of the DVD as each DVD has multiple copies associated with it

4. Director

- Director ID (dir_id) – This property of Director entity will assign a unique ID for each director
- Firstname (fname) and Lastname (lname) – These attributes are necessary to provide information about the director when searching for films directed by a particular director.

5. Actor

Needless to say, an actor will only be included in this database whereby (s)he fulfills a role in a listed DVD. In other words, the plethora of all actors everywhere will not be included without an accompanying DVD title in stock by this store.

- Actor ID (act_id) – This property fulfills the unique identifier for each actor entered in this entity.
- Firstname (fname) and Surname (lname) – Every record of an actor will require the actor's firstname and surname.

6. Copy

A copy is defined as a hard-disk duplicate of some DVD movie. The store will stock multiple copies of a DVD title, e.g. there are 20 copies of Gladiator, 50 copies of 'The

Hunger Games'. This is to accommodate multiple, simultaneous requests of the movie. Each copy is a physical disk, and thus should be recorded as an entry in our Copy table.

- Copy ID (copy_id) – This is the surrogate key to uniquely identify every instance of a copy of a movie.
- Availability (availability) – This property of copy defines whether the copy is currently being rented out and/or sold, in which case, its value is 'N', or alternatively, it in stock and ready to be rented/sold (corresponding to a value of 'Y')
- Date Copy Added (date_copy_added) – The store may decide to increase its copies of certain popular movies, and so it is necessary to keep track of when a copy was added to inventory. This property of Copy may also help determine, in the future, when the store should change a DVD's status to 'for sale', if it has been in stock for several years and is decreasing in popularity.
- DVD ID (dvd_id) – This property will delimit what movie (encapsulated by 'DVD') that a copy is a copy of. For example, copy with copy_id 345 is a copy of dvd_id 5 which corresponds to the title, 'Home Alone 3'. Thus, this attribute is the link to the DVD entity.

7. <u>Character</u>

A character is defined as a protagonist in a movie that available for rent/sale in the store. Importantly, not all characters in a movie will be noted in the DB. Only protagonists, or characters in popular and scholarly imagination, will be listed in the Character entity/table. A character will not 'exist' if a copy of the movie in which they are part of is not available in the store.

- Character ID (char_id) – A unique identifier that is a surrogate key for each record of a character
- Firstname (fname) and Lastname (lname) – Basic information, such as these, will be stored for each character

8. <u>Exrental</u>

An exrental is a copy of a DVD which is no longer popular among members for renting, and/or is outdated in terms of years or sociocultural taste. Whether a DVD rental is demoted to 'exrental' status is subjective and dependent on the store's decision-makers. Obviously, when a copy is for sale, it can no longer be for rent, and thus its status in the DVD table should reflect this change (i.e. change the status to 'sale'). When an exrental is sold, its 'availability' field will also be updated to delineate the change (i.e. availability value is 'N').

- Exrental ID (exrental_id) – Each copy for sale (exrental) will have to be uniquely identified since it is a single physical copy
- Exrental Revenue (exrent_rev) – The sale price of each exrental will be documented in this column, allowing the users to aggregate the total sales of all exrentals sold. This attribute is necessary as the price can vary over time (see specification), and in some cases, the store may desperately want to sell the exrental by incrementally lowering the price.
- Date Exrental Sold (date_sale) – The date that an exrental was sold should be noted for verification of sale
- Copy ID (copy_id) – The copy number of the exrental being sold should be recorded. This attribute also provides a linking opportunity to the Copy entity.

Step 4: Marking the Primary Keys of Entities

It is informative, at this stage, to demarcate the primary key for each entity, that is, the attribute that will uniquely and comprehensively identify instances of an entity type. The primary key for each entity will be underlined below, followed by a brief justification of why we chose that attribute as a primary key.

List of Primary Keys for Entities:

1. Member (<u>member number</u>, first name, surname, date of registration, phone, address)
2. DVD (<u>DVD ID</u>, title of movie, year of release, commercial status, price, genre)
3. Rental (<u>Rental ID</u>, rental revenue, date rented, date returned, member number, copy ID)
4. Director (<u>Director ID</u>, first name, surname)
5. Actor (<u>Actor ID</u>, first name, surname)
6. Copy (<u>Copy ID</u>, availability, date copy was added, DVD ID)
7. Character (<u>Character ID</u>, first name, surname)
8. Exrental (<u>Exrental ID</u>, exrental revenue, date of sale, copy ID)

Primary keys in relational model format:

1. Member (member_nbr, fname, lname, date_reg, phone, addr)
2. DVD (dvd_id, dvd_title, dvd_year, status, price, genre)
3. Rental (rental_id, rent_revenue, date_rent, date_return, member_nbr, copy_id)
4. Director (dir_id, fname, lname)
5. Actor (act_id, fname, lname)
6. Copy (copy_id, availability, date_copy_added, dvd_id)
7. Character (char_id, fname, lname)
8. Exrental (exrental_id, exrent_rev, date_sale, copy_id)

1. Member

Member_nbr will be a surrogate key that uniquely identifies each instance of the Member entity type.

2. DVD

DVD_id will be the auto-generated unique key that identifies each instance of a film.

3. Rental

Rental_id will be an auto-incremented unique key that identifies each tuple in the Rental entity type.

4. Director

Dir_id will the primary key for Director where each instance of this entity type will be assigned a unique identifier automatically.

5. Actor

Each instance of the actor entity will be uniquely identified by their auto-generated surrogate numeric key.

6. Copy

Each record of the Copy Entity will have a unique numeric identifier.

7. Character

Every record of the character entity is assigned a unique auto-generated character ID.

8. Exrental

Every record of the Exrental entity will have a unique autogenerated key.

Step 5: Define the Relationships

It is now incisive to define the relationships between entities, based on our final entity list described previously.

1. Member
2. DVD
3. Rental
4. Director
5. Actor
6. Copy
7. Character
8. Exrental

A. Member and Rental: A member will 'rent' a rental.
B. Member and Exrental: A member will potentially purchase an 'exrental' DVD.
C. DVD and Director: A DVD movie is directed by a director.
D. DVD and Actor: A DVD movie can contain actors, for example, in most non-animated productions, several actors will star.
E. DVD and Copy: A DVD movie will have many copies in stock by the store.
F. DVD and Character: A DVD movie will showcase many characters. For example, the movie, 'Bridget Jones' Diary' displays the infamous characters, 'Bridget Jones' and 'Mr. Darcy'.
G. Rental and Copy: A rental DVD is a rental of a copy of a particular movie.
H. Actor and Character: An actor can play one character or several characters both in the same movie, and in different movies. For example, the actor Robin Williams play many characters in 'Ms. Doubtfire' (the nanny and the man in pursuit of a woman) but Robin Williams also plays characters in many different movies, for

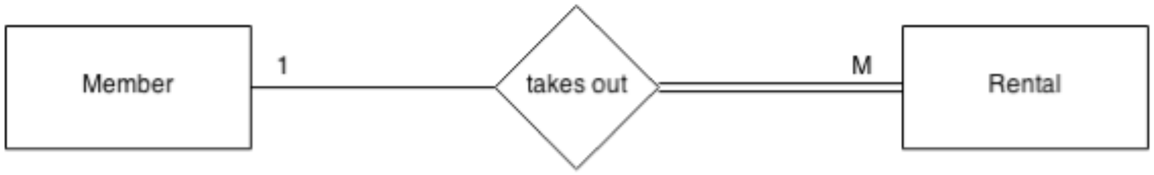
example, Robin Williams has one character in ‘Rude Awakenings’ and ‘Good Will Hunting’.

I. Copy and Exrental: A copy of a movie can be an exrental.

Step 6: Cardinality and Optionality of Relationships

The cardinality and optionality of the defined relationships are illuminated in the section. Cardinality is discussed in part (A) of each relationship, and optionality in part (B).

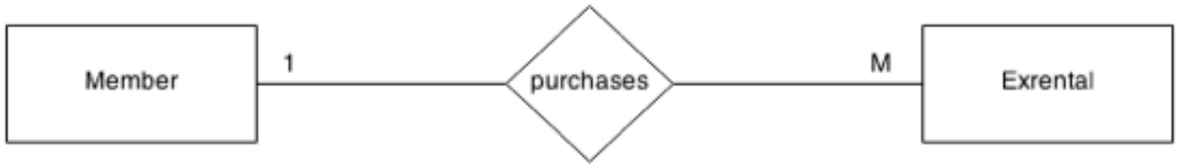
1.



A. The relationship between ‘Member’ and ‘Rental’ is a one-to-many (1:M). This is so because a member of the store can rent out many DVD rentals during his or her time as a registered member. However, one rental, which is defined as one transaction where a copy of a DVD is rented out to a member at a certain price (that is subject to variability), involves only one member. Every rental is assigned a unique rental ID and thus this can only be associated with one member.

B. The optionality of the relationship between Member and Rental is optional:mandatory. A member can take out a rental, but a rental must be taken out by a member. Briefly, a member can exist independently of any rentals under its account history. This might happen when a customer first registers to be a member, and does not have any rental history yet. However, a rental record cannot be recorded without a corresponding member. As described in the specification, DVD rentals can only be rented by a registered member.

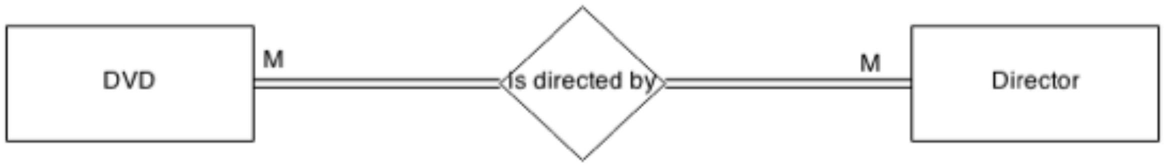
2.



A. The relationship between Member and Exrental is necessarily a 1:M. Tellingly, a member may purchase many exrental DVDs over the course of their registration with the DVD Store. However, each instance of an exrental will be recorded with one member number. In other words, the purchase of one exrental is associated with just one customer.

B. The optionality of Member and Exrental is optional:optional. Each member may choose to purchase an exrental, but is not required to. Likewise, each exrental may be purchased by a member, or any member of the public that is not registered with the DVD store. Therefore, the relationship is best described as optional:optiontal.

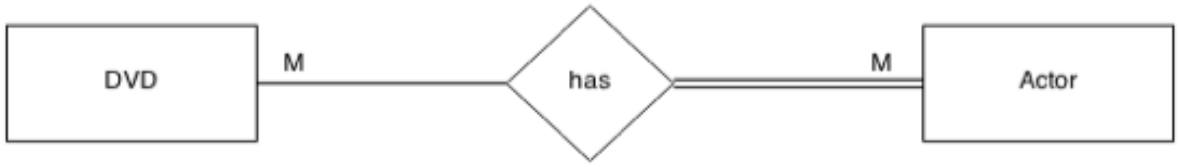
3.



A. The cardinality of the DVD and Director relationship is many-to-many. A DVD can be directed by several directors and a director listed in this DB may have directed any number of DVD movies.

B. The optionality of the DVD and Director relationship is a mandatory:mandatory one. A DVD movie must be directed by a director, as no film is created without a director. Equally, a director, that is listed in our database, must be pertinent to at least one DVD. For the purposes of the current DB then, a director cannot 'exist' without a corresponding movie.

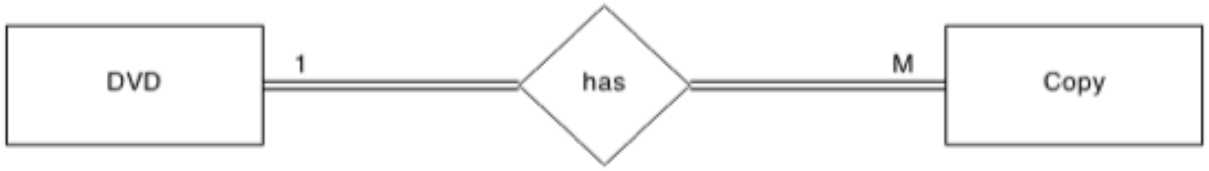
4.



A. The relationship between DVD movie and Actor is many-to-many. A DVD movie can contain many film actors. Likewise, a listed actor can feature in several DVD films.

B. The optionality of the DVD-Actor relationship is optionality:mandatory. A DVD can contain actors but it does not have to. Films of genre such as documentary (especially, about nature) rarely contain actors, for example, 'March of the Penguins'. Another genre, animated films, similarly sometimes contains no actors to provide voices of characters, e.g. 'Tom and Jerry: The Movie'. On the other hand, actors must feature in a DVD movie for inclusion in our database.

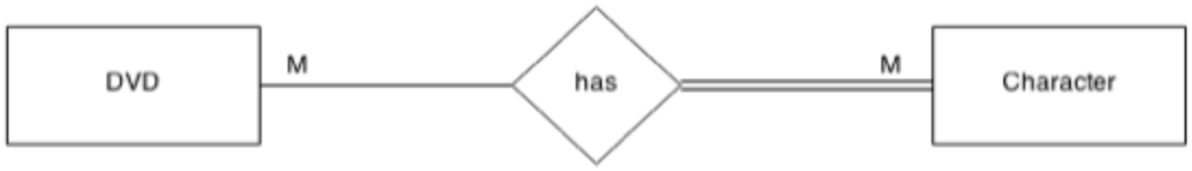
5.



A. The DVD-Copy relationship is 1:M. A DVD movie can have several copies of it. A copy is of one movie, however.

B. The DVD-Copy relationship is also mandatory:mandatory. A DVD must have a copy to enable renting/selling. A copy must be of a DVD movie.

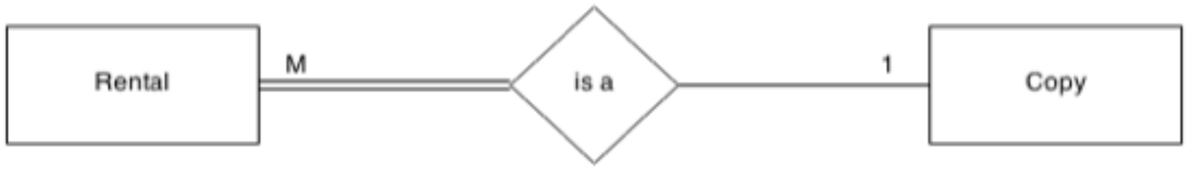
6.



A. The DVD-Character relationship is best perceived as a many-to-many (M:N). A DVD can contain several characters (e.g. Twilight has a cast of several actors) and a character can star in several DVD movies (e.g. the character 'Batman' stars in several movies – all the Batman movies).

B. The DVD-Character relationship is characterized as optional:mandatory relationship. A DVD may contain characters, but does not have to (e.g. documentaries). A character, if it is to be listed in our DB, must be part of at least one DVD movie.

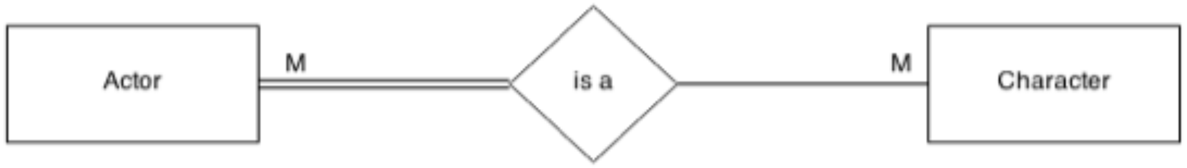
7.



A. The Rental-Copy relationship can be depicted as a many-to-one. A DVD rental is a transaction including only one copy. A copy of a DVD movie, however, can be rented many times.

B. The relationship between rental and copies is invariably mandatory-to-optional. A rental transaction must be of one DVD copy. Further, a copy can be a rental, but not exclusively, as a copy of a movie can also be available in the form of an exrental.

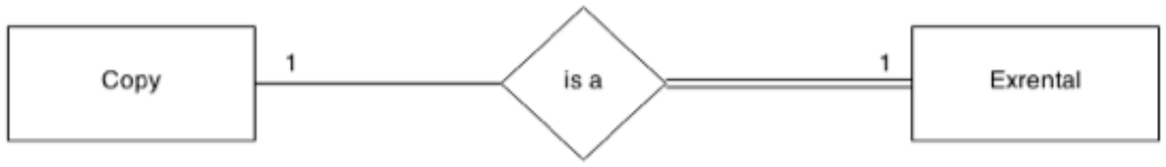
8.



A. The relationship between Actor and Character is many-to-many. One actor can have multiple character aliases in their career, e.g. Angeline Jolie plays Lara Croft in Tomb Raider, and Mrs. Smith in Mr. & Mrs. Smith. One character can also be played by many actors. An illustrative example of the latter case is in the movie Jane Eyre where Jane is played by a young actress in scenes of her as a young girl, and then she is played again by a different actress when showing Jane as a governess with many suitors.

B. The Actor-Character relationship is further ascribed as mandatory-to-optional. An actor must play at least one character to be included in our database. A character, in contrast, can be listed in our DB without an accompanying actor. This occurs when a character is animated, and is not voiced by an actor.

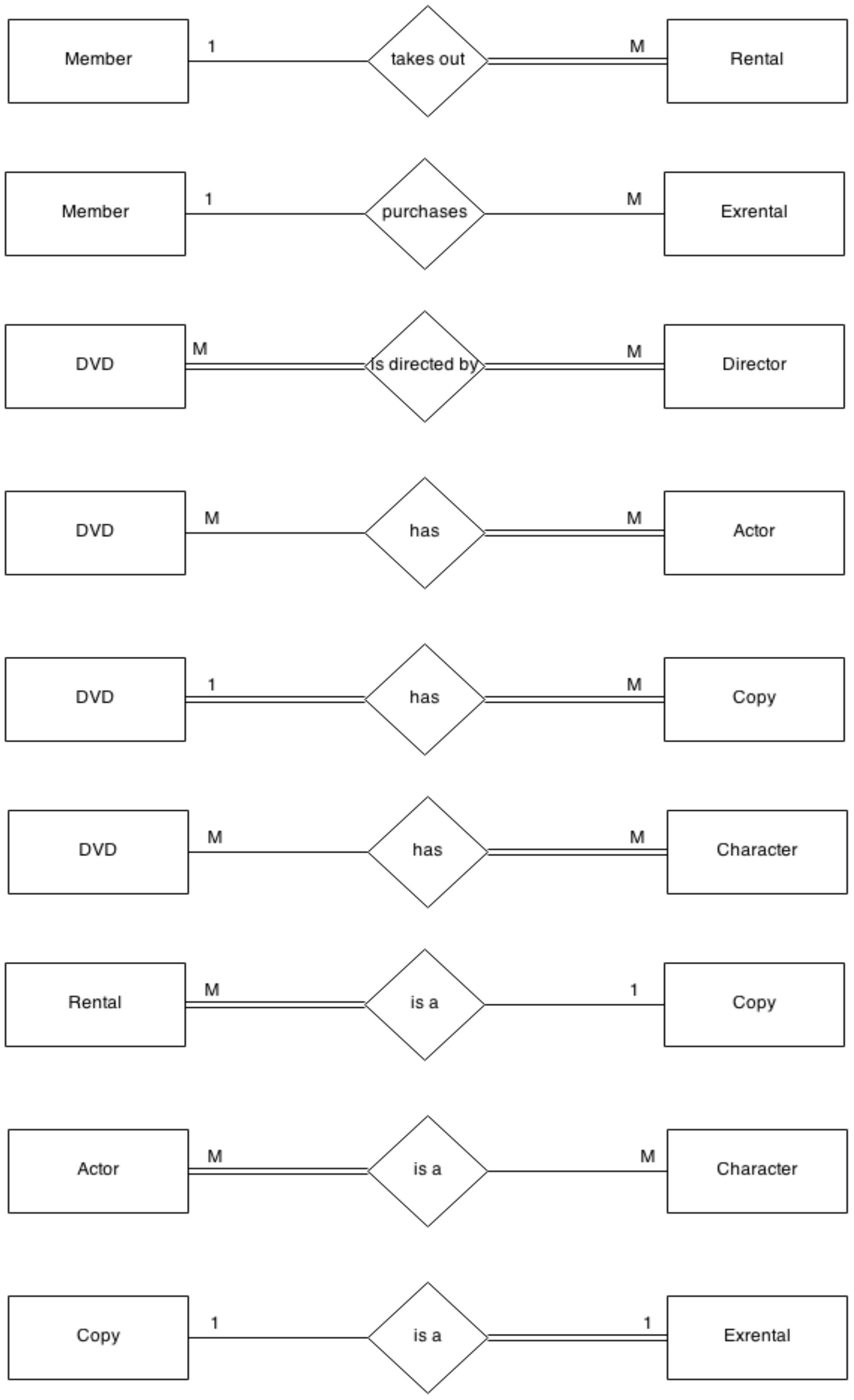
9.



A. The relationship between Copy and Exrental is a one-to-one. This is due to the fact that a copy can be used for selling once (an exrental). An exrental DVD is of one copy of that movie.

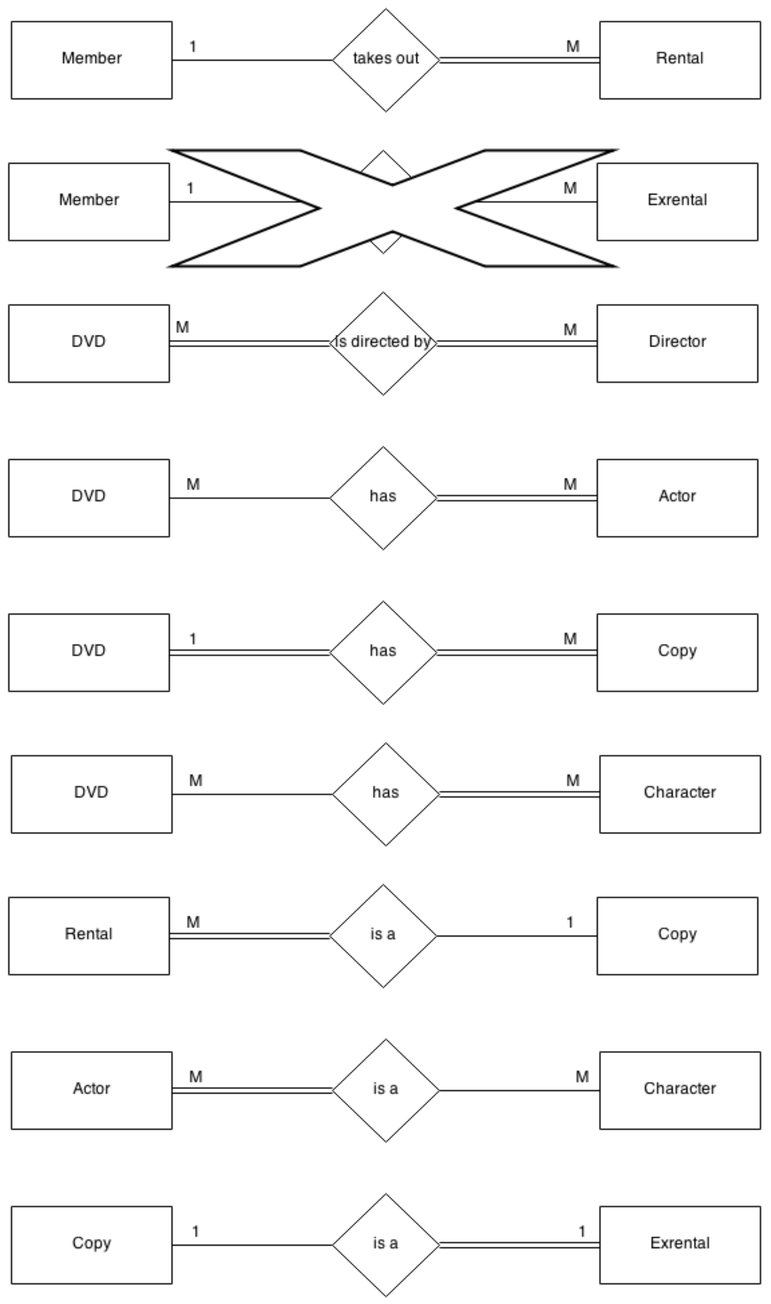
B. The relationship between Copy and Exrental is also an optional:mandatory one. A copy of a movie can be sold as an exrental or rented as a rental. Thus, this side is optional. An entry of an exrental must be of a copy of a DVD, however.

A diagrammatic version of the relationships are shown on the next page.

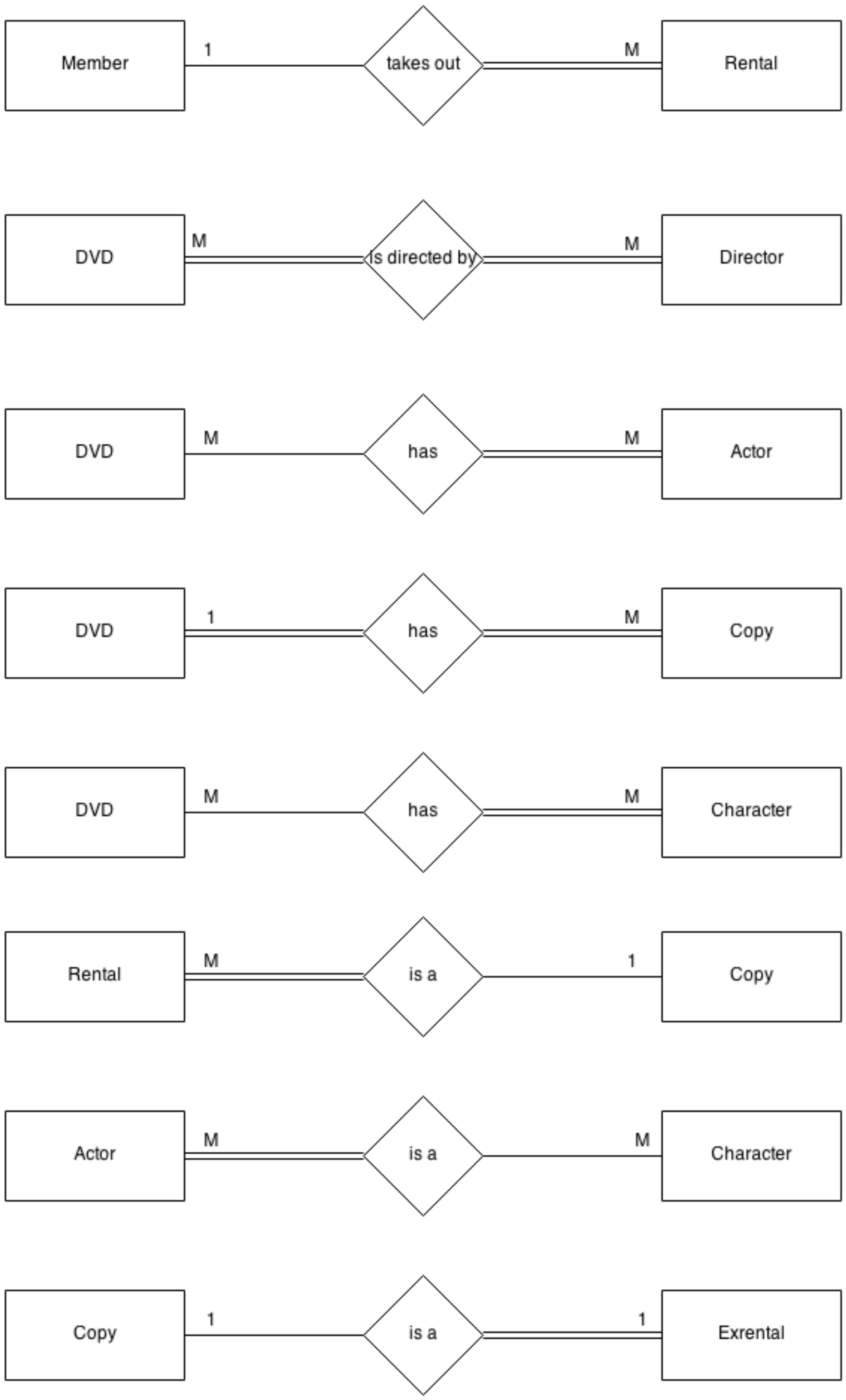

Member
1
takes out
M
Rental
Member
1
purchases
M
Exrental
DVD
M
is directed by
M
Director
DVD
M
has
M
Actor
DVD
1
has
M
Copy
DVD
M
has
M
Character
Rental
M
is a
1
Copy
Actor
M
is a
M
Character
Copy
1
is a
1
Exrental

Step 7: Removing Redundant Relationships

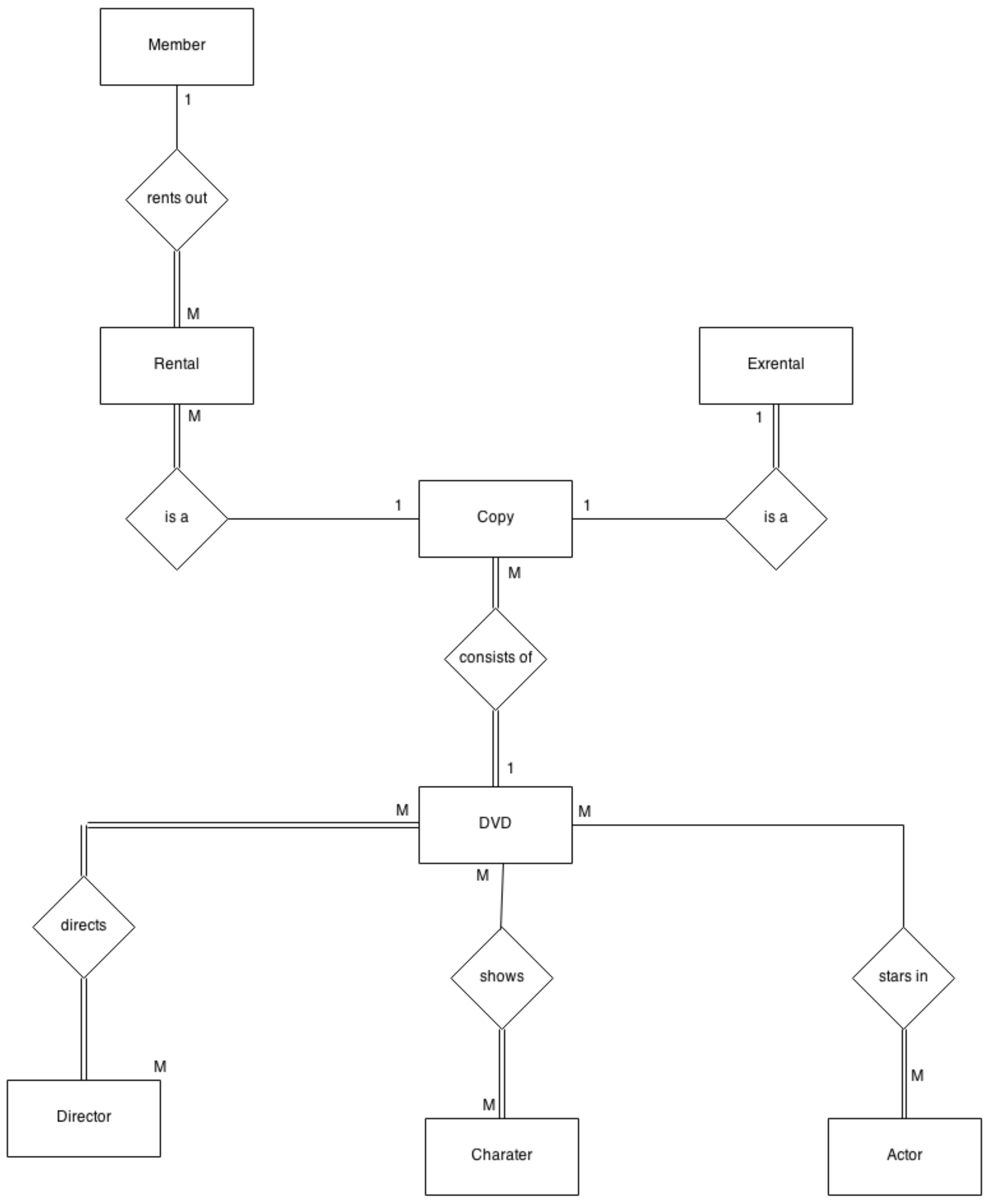
The relationship between Member and Exrental is deemed redundant. Our database does not need to store information about what exrental DVDs were purchased by which members, as they could be purchased by members of the public too, for which no record is necessary. Therefore, this relationship can be removed (see diagram below) to reveal our final set of relationships.



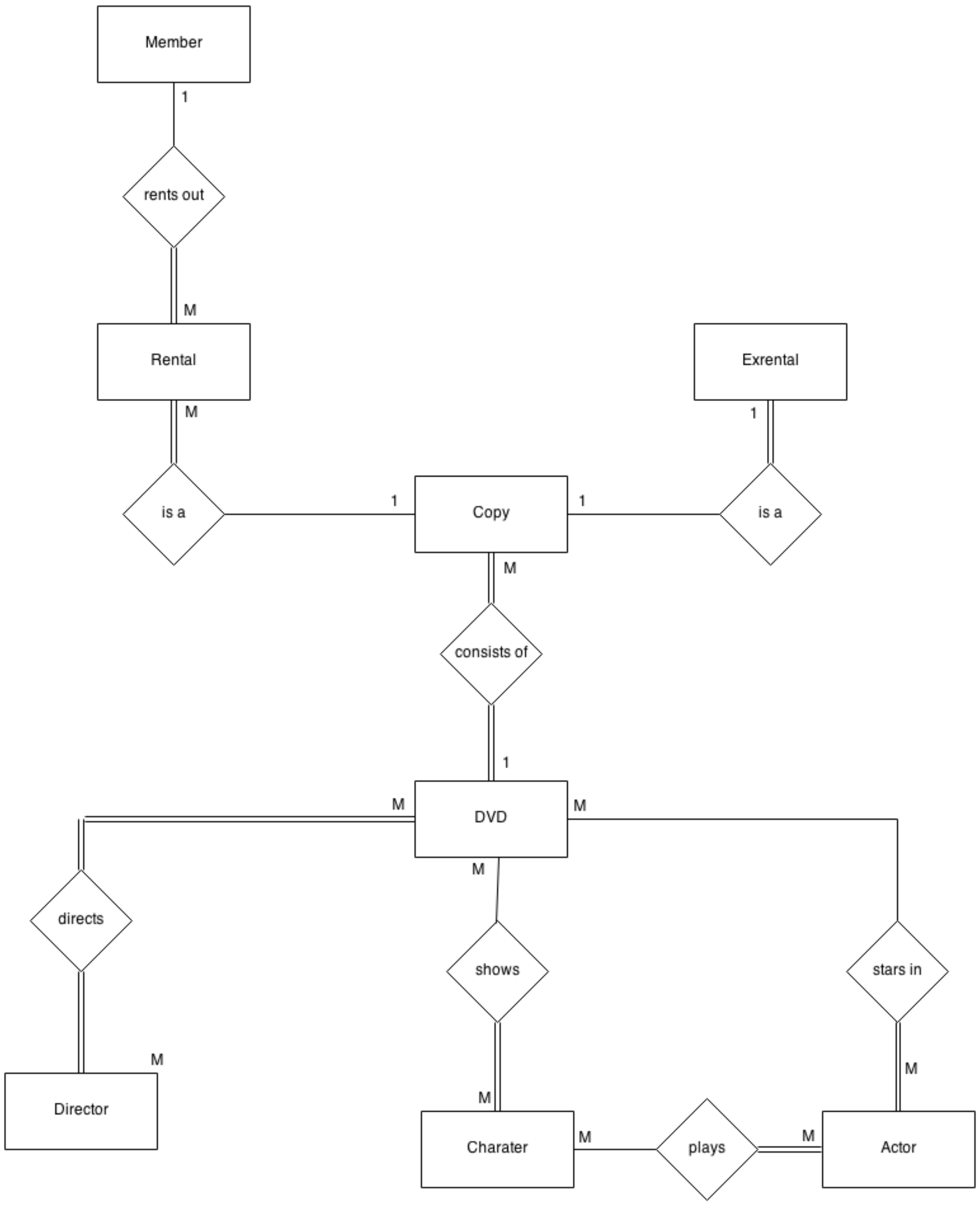
Now, our final relationships look like the following:


Member
1
takes out
M
Rental
DVD
M
is directed by
M
Director
DVD
M
has
M
Actor
DVD
1
has
M
Copy
DVD
M
has
M
Character
Rental
M
is a
1
Copy
Actor
M
is a
M
Character
Copy
1
is a
1
Exrental

ERD Draft 1: At this point in the iterative process, my thinking precluded the association between Character and Actor. Specifically, I did not ascribe a relationship here, as I thought customers would simply search for a film based on a favourite actor. This was a myopic viewpoint, as it did not facilitate occasions where customers wanted to retrieve the character a particular actor played in a DVD, that is connecting the characters and actors of a movie.


Member
1
rents out
M
Rental
M
is a
1
Copy
1
is a
1
Exrental
M
consists of
1
M
DVD
M
M
directs
shows
stars in
M
M
M
Director
Charater
Actor

ERD Draft 2 - Final: My final and revised ERD reflects the new relationship between Character and Actor, which represents a M:M relationship (which necessitates a junction table). This is a much more complete, and complex design that accommodates a wide search for films based on the customer's knowledge of characters and actors in DVDs.


Member
1
rents out
M
Rental
M
is a
1
Copy
1
is a
1
Exrental
M
consists of
1
DVD
M
M
M
directs
shows
stars in
M
Director
M
Charater
M
plays
M
M
Actor

Q2. Mapping the ER model to a Relational Model

Mapping our conceptual (ER) model to one suitable for use in a relational database is represented textually below. Primary keys are underlined, and foreign keys are italicized. Also, please note that table names are prefixed with 'ASS1_' as a requirement of this assignment (Alex Cronin, Personal Communication (Moodle), 2014). Further, the rationale for the choice of primary and foreign keys for each entity type are clearly documented.

Normalization to Third Normal Form: All relational tables have been normalized to 3NF, and detail is provided below as to why each with is comfortably in 3NF. A database is in third normal form when there are no repeating attributes or groups of attributes (1NF), every non-key attribute is dependent on the whole key (2NF), and when there are no non-key attributes that depend on another non-key attribute (3NF). We can observe that each of the tables below demonstrates normalization to 3NF.

1. ASS1_DVD(<u>DVD_ID</u>, DVD_TITLE, DVD_YEAR, STATUS, PRICE, GENRE)

   This textual mapping represents the DVD entity being mapped to a relation, and its attributes to column headings in the relation. DVD_ID is the primary key as it can uniquely identify each instance of a movie. DVD_TITLE was also thought to be a candidate for the primary key initially, however, it was discarded due to the fact that multiple, mostly lesser-known, movies have the same title (that cannot be copyright – See Assumptions). The DVD relation contains no foreign keys.

   Normalization: Every column contains atomic values, and there are no repeating groups (1NF), all of the attributes depend fully on DVD_ID (e.g. DVD_ID determines title, year of release, commercial status, price, and genre). Finally, there are no functional dependencies between non-key attributes (e.g. 'genre' does not depend on DVD title, as we stated in our assumptions that, while surprisingly, many DVD movies can have the same title, therefore, genre must depend on the unique DVD_ID). Price, status, year of release and title are all further dependent on DVD_ID.

2. ASS1_COPY(<u>COPY_ID</u>, AVAILABILITY, DATE_COPY_ADDED, *DVD_ID*)

   This mapping shows that COPY_ID is the unique identifier of each individual occurrence of a copy of a DVD. There is no other contender for the primary key. DVD_ID is the foreign key in this relation to link it with the DVD relation to retrieve what DVD/movie the copy is a copy of.

Normalization: Every column contains atomic values, and there are no repeating groups (1NF), all of the attributes depend fully on COPY_ID (e.g. COPY_ID determines availability, date copy was added and what DVD it is a copy of). There exist no partial dependencies, as each attribute depends fully on the key. Finally, there are no functional dependencies between non-key attributes. Availability of the copy, date it was added to inventory, and DVD_ID are all further directly dependent on COPY_ID.

3. ASS1_MEMBER(<u>MEMBER_NBR</u>, FNAME, LNAME, DATE_REG, PHONE, ADDR)

MEMBER_NBR is the primary key for the Member Relation. This key can uniquely identify an occurrence of the Member entity type, while the other keys, individually or separately (e.g. FNAME and LNAME) cannot claim to achieve this. There is no foreign key associated with this table.

Normalization: Every column contains atomic values, and there are no repeating groups (1NF), all of the attributes depend fully on MEMBER_NBR (e.g. MEMBER_NBR determines first name, surname, date of registration, phone number, and address). Finally, there are no functional dependencies between non-key attributes; all of the non-key attributes depend solely and directly on MEMBER_NBR.

4. ASS1_DIRECTOR(<u>DIR_ID</u>, FNAME, LNAME)

DIR_ID is the primary key for the Director relation, which maps the Director entity. This is due to the fact that first names and surnames of directors are not sufficiently unique to be able to represent each occurrence of Director. There is no foreign key associated with the Director relation (Relevant junction table is described below).

Normalization: Every column contains atomic values, and there are no repeating groups (1NF), all of the attributes depend fully on the primary key, DIR_ID (e.g. DIR_ID determines firstname and surname of director). Finally, there are no functional dependencies between non-key attributes, that is, surname does not depend on firstname, rather it has to depend on DIR_ID as many directors have the same last name but different first names.

5. ASS1_FILM_CHAR(<u>CHAR_ID</u>, FNAME, LNAME)

CHAR_ID is the primary key for the ASS1_FILM_CHAR relation which maps from the Character entity. Char_ID is necessarily a surrogate key in this case, also. This relation contains no foreign key (Junction table below).

Normalization: Every column contains atomic values, and there are no repeating groups (1NF), all of the attributes depend fully on CHAR_ID (e.g. CHAR_ID determines first name and surname of character). Finally, there are no functional dependencies between non-key attributes; last name does not depend of first name of character, as some character may have the same last names but different first names, and thus, both first name and surname are both dependent solely on CHAR_ID.

6. ASS1_ACTOR(<u>ACT_ID</u>, FNAME, LNAME)

ACT_ID is the surrogate primary key for the Actor entity type, mapped textually here. There is no foreign key for this relation (its junction table (DVD-Actor relationship) will follow).

Normalization: Every column contains atomic values, and there are no repeating groups (1NF), all of the attributes depend fully on ACT_ID (e.g. ACT_ID determines first name and surname of actors). Finally, there are no functional dependencies between non-key attributes; surname of actor does not have a functional dependency on first name, as many actors can have the same surname and different first names.

7. ASS1_CHAR_ACTOR(*<u>CHAR_ID</u>*, *<u>ACT_ID</u>*)

The CHAR_ACTOR relation is a junction table to suffice for the many-to-many relationship between the Character and Actor entities. Invariably then, both CHAR_ID and ACT_ID together form the composite key and also act as the foreign keys to the FILM_CHAR and ACTOR relations, respectively.

Normalization: This junction table is normalized to 2NF and 3NF already by virtue of the fact that it has zero non-key attributes (that it contains no repeating groups is a given, also).

8. ASS1_RENTAL(RENTAL_ID, RENT_REVENUE, DATE_RENT, DATE_RETURN, *MEMBER_NBR*, *COPY_ID*)

RENTAL_ID is the primary key for the RENTAL relation, which maps the DVD Rental entity. This was chosen because a numeric surrogate key is preferable over any of the other attributes/natural keys already in the entity. Its foreign keys are MEMBER_NBR and COPY_ID to link up with the Member relation when recording what member took out what DVD rental, and also to record what copy the member rented.

Normalization: Every column contains atomic values, and there are no repeating groups (1NF), all of the attributes depend fully on RENTAL_ID (e.g. RENTAL_ID determines the income derived from the rental (which may have surcharges from late fees), the date rented, the date the copy was returned, the member that rented it, and the exact being rented). Finally, there are no functional dependencies between non-key attributes; none of the non-key attributes depend on any other non-key attributes.

9. ASS1_EXRENTAL(EXRENTAL_ID, EXRENTAL_REVENUE, DATE_SALE, *COPY_ID*)

EXRENTAL_ID is the primary key of choice as there is no other efficient way of recording a unique occurrence of this entity type. COPY_ID is the foreign key so as to link with the Copy relation to record information about what copy of a DVD was sold.

Normalization: Every column contains atomic values, and there are no repeating groups (1NF), all of the attributes depend fully on EXRENTAL_ID (e.g. EXRENTAL_ID determines exrental revenue/income, date of sale, and the copy sold). Finally, there are no functional dependencies between non-key attributes; exrental revenue does not depend on Copy ID (as the price is variable) and it doesn't depend on the date of the sale. Likewise, date_sale and copy_id can exist independently of one another.

10. ASS1_DVD_DIR(*DIR_ID*, *DVD_ID*)

DIR_ID and DVD_ID both present as the primary composite key as the only way to uniquely identify the DVDs a Director has directed (M:N) is to utilize both IDs, in this junction table. They further serve as foreign keys to the respective relations to get this information.

Normalization: This junction table is in 1NF due to absence of repeating groups. It is automatically in 2NF and 3NF because it contains zero non-key attributes.

11. ASS1_DVD_ACT(*ACT_ID*, *DVD_ID*)

ACT_ID and DVD_ID are both primary keys in this table, as it is a junction table between the Actor and DVD relations. Both keys also are foreign keys to assist with linking back to these tables to populate them.

Normalization: This junction table is in 1NF due to lack of repeating groups. Further, it contains no non-key attributes, and thus is in 2NF and 3NF.

12. ASS1_DVD_CHAR(*CHAR_ID*, *DVD_ID*)

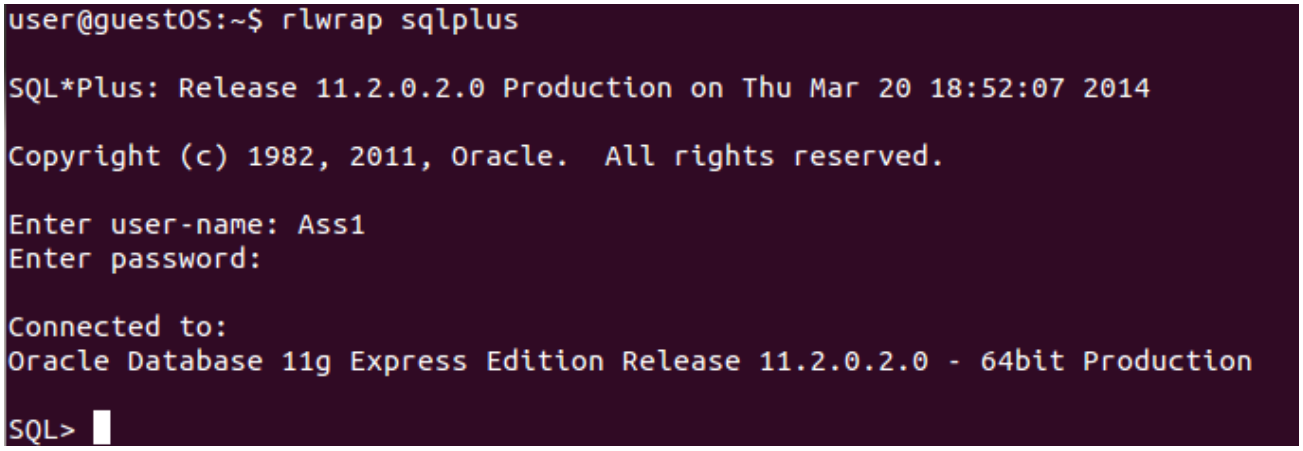
Finally, CHAR_ID and DVD_ID make up the composite primary key in this relation which maps the junction table to assist the decomposition of the M:N relationship between Character and DVD entities. Further, both keys serve as foreign keys to the necessary tables to enable a way to retrieve what characters feature in what movies, or what movies feature a particular character.

Normalization: This junction table is in 1NF due to lack of repeating groups. Further, it contains no non-key attributes, and thus is in 2NF and 3NF.

Screenshots

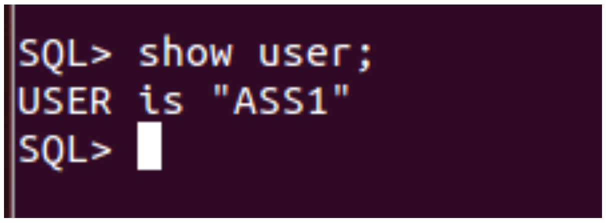
Q3. Create the database in Oracle, showing screenshots of queries.

The schema/workspace, 'Ass1', was first created using the Oracle Express 11g Edition web interface, as per assignment guidelines. The following screenshot depicts connecting to this schema from the terminal, where we will use SQL queries to create our tables.



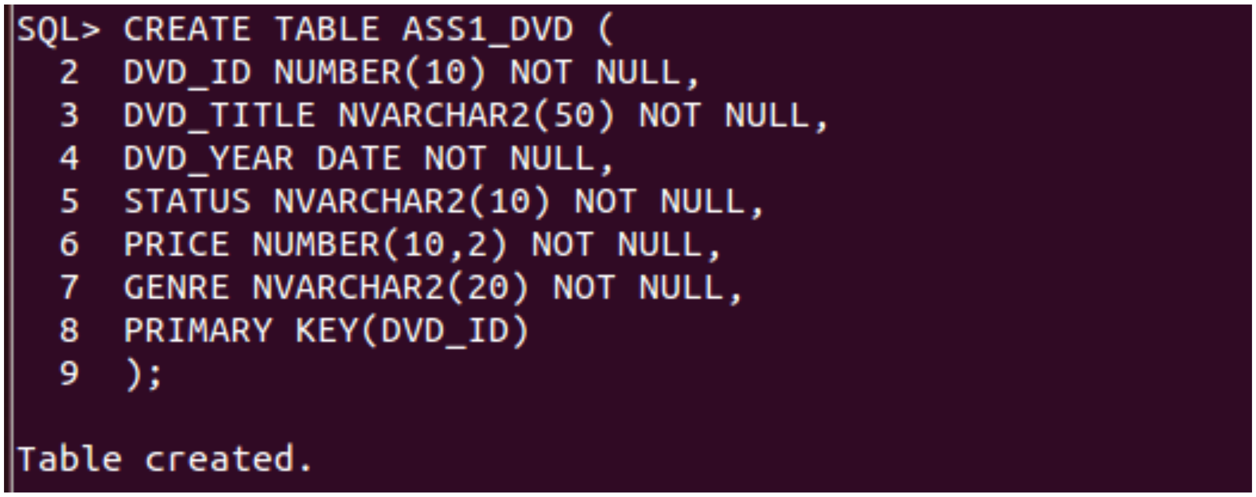
```
user@guestOS:~$ rlwrap sqlplus

SQL*Plus: Release 11.2.0.2.0 Production on Thu Mar 20 18:52:07 2014

Copyright (c) 1982, 2011, Oracle.  All rights reserved.

Enter user-name: Ass1
Enter password:

Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL> 
```

The following screenshots confirms the user (or schema):



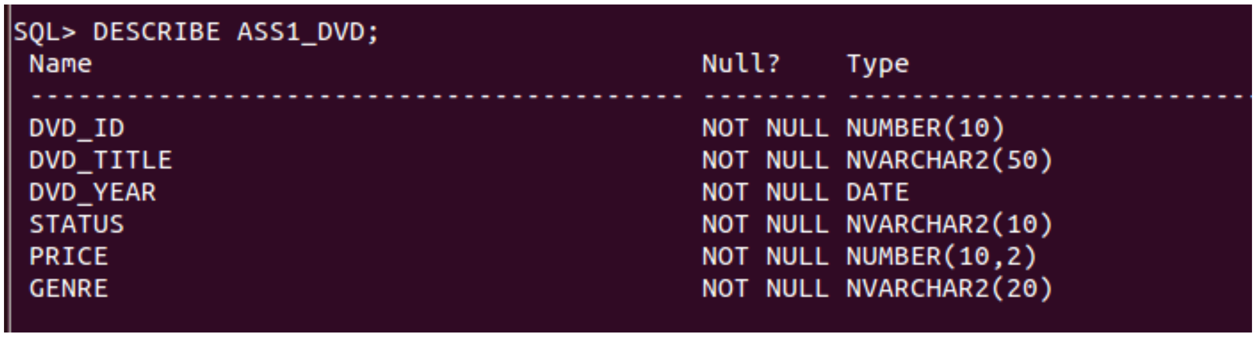
```
SQL> show user;
USER is "ASS1"
SQL> 
```

1. Creation of ASS1_DVD table:



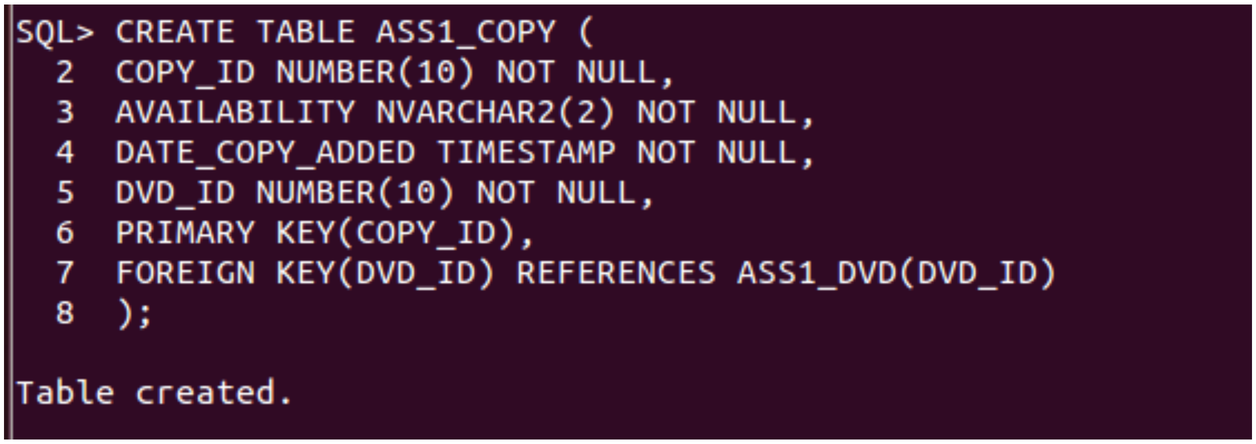
```
SQL> CREATE TABLE ASS1_DVD (
  2  DVD_ID NUMBER(10) NOT NULL,
  3  DVD_TITLE NVARCHAR2(50) NOT NULL,
  4  DVD_YEAR DATE NOT NULL,
  5  STATUS NVARCHAR2(10) NOT NULL,
  6  PRICE NUMBER(10,2) NOT NULL,
  7  GENRE NVARCHAR2(20) NOT NULL,
  8  PRIMARY KEY(DVD_ID)
  9  );

Table created.
```

2. View table description:



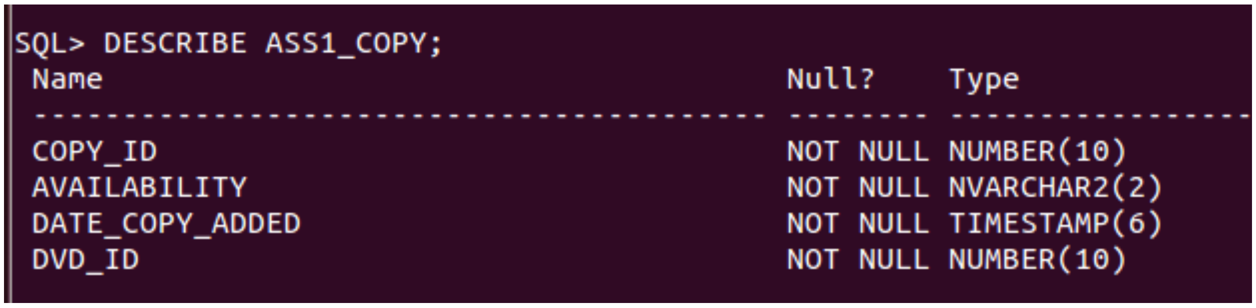
```
SQL> DESCRIBE ASS1_DVD;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DVD_ID                                    NOT NULL NUMBER(10)
 DVD_TITLE                                 NOT NULL NVARCHAR2(50)
 DVD_YEAR                                  NOT NULL DATE
 STATUS                                    NOT NULL NVARCHAR2(10)
 PRICE                                     NOT NULL NUMBER(10,2)
 GENRE                                     NOT NULL NVARCHAR2(20)
```

3. Creation of ASS1_COPY table:



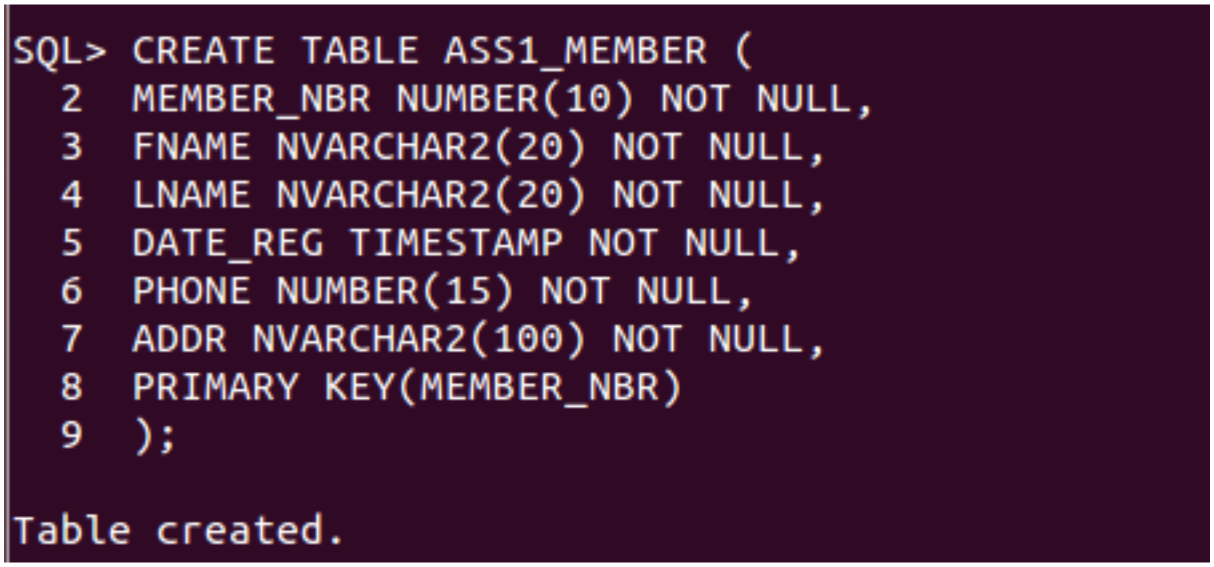
```
SQL> CREATE TABLE ASS1_COPY (
COPY_ID NUMBER(10) NOT NULL,
AVAILABILITY NVARCHAR2(2) NOT NULL,
DATE_COPY_ADDED TIMESTAMP NOT NULL,
DVD_ID NUMBER(10) NOT NULL,
PRIMARY KEY(COPY_ID),
FOREIGN KEY(DVD_ID) REFERENCES ASS1_DVD(DVD_ID)
);

Table created.
```

4. View table description:



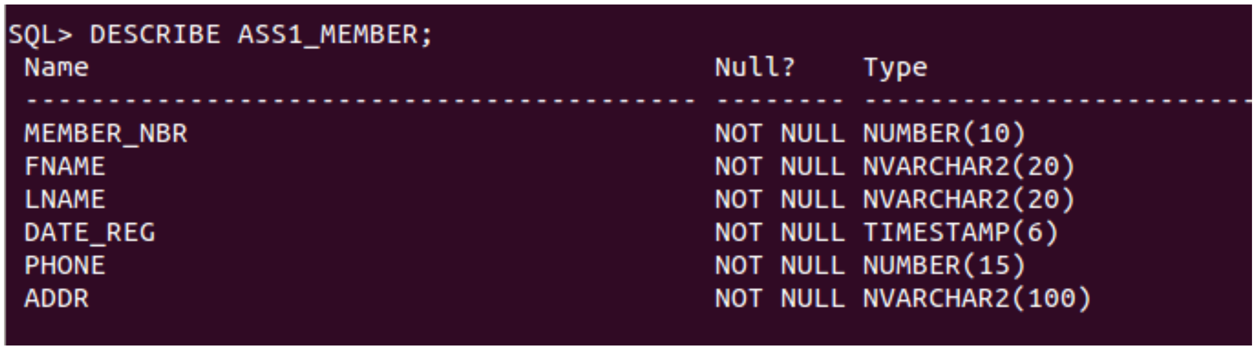
```
SQL> DESCRIBE ASS1_COPY;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 COPY_ID                                   NOT NULL NUMBER(10)
 AVAILABILITY                              NOT NULL NVARCHAR2(2)
 DATE_COPY_ADDED                           NOT NULL TIMESTAMP(6)
 DVD_ID                                    NOT NULL NUMBER(10)
```

5. Creation of ASS1_MEMBER table:



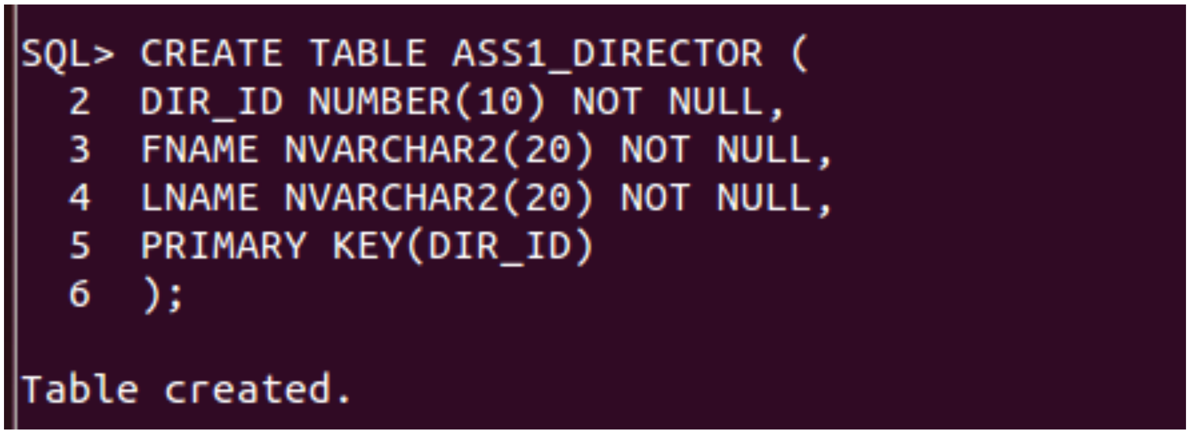
```
SQL> CREATE TABLE ASS1_MEMBER (
MEMBER_NBR NUMBER(10) NOT NULL,
FNAME NVARCHAR2(20) NOT NULL,
LNAME NVARCHAR2(20) NOT NULL,
DATE_REG TIMESTAMP NOT NULL,
PHONE NUMBER(15) NOT NULL,
ADDR NVARCHAR2(100) NOT NULL,
PRIMARY KEY(MEMBER_NBR)
);

Table created.
```

6. View table description:



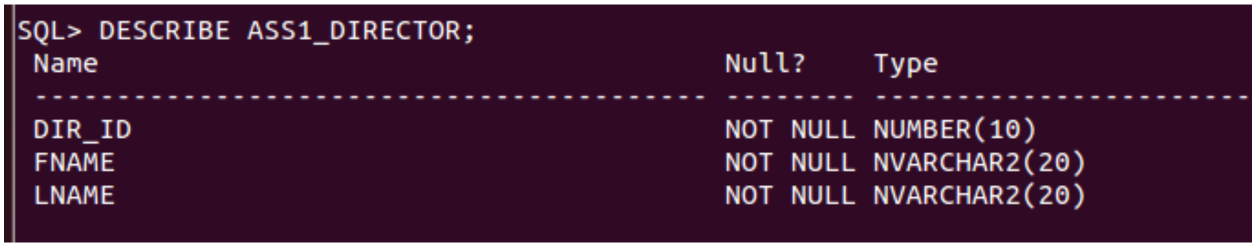
```
SQL> DESCRIBE ASS1_MEMBER;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 MEMBER_NBR                                NOT NULL NUMBER(10)
 FNAME                                     NOT NULL NVARCHAR2(20)
 LNAME                                     NOT NULL NVARCHAR2(20)
 DATE_REG                                  NOT NULL TIMESTAMP(6)
 PHONE                                     NOT NULL NUMBER(15)
 ADDR                                      NOT NULL NVARCHAR2(100)
```

7. Creation of ASS1_DIRECTOR table:



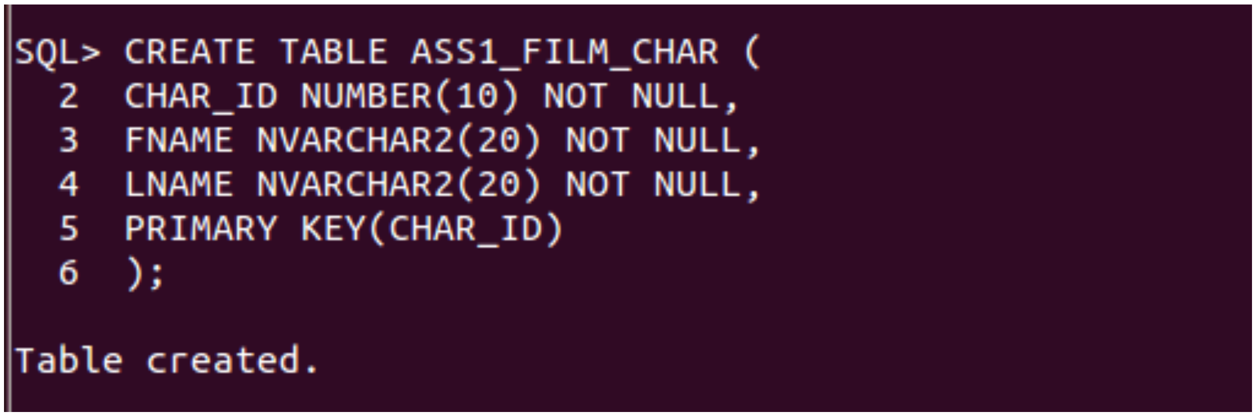
```
SQL> CREATE TABLE ASS1_DIRECTOR (
  2  DIR_ID NUMBER(10) NOT NULL,
  3  FNAME NVARCHAR2(20) NOT NULL,
  4  LNAME NVARCHAR2(20) NOT NULL,
  5  PRIMARY KEY(DIR_ID)
  6  );

Table created.
```

8. Viewing table description:



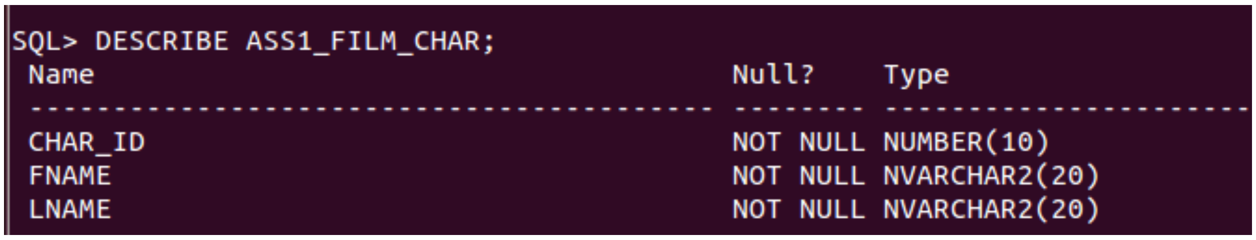
```
SQL> DESCRIBE ASS1_DIRECTOR;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DIR_ID                                    NOT NULL NUMBER(10)
 FNAME                                     NOT NULL NVARCHAR2(20)
 LNAME                                     NOT NULL NVARCHAR2(20)
```

9. Creation of ASS1_FILM_CHAR table:



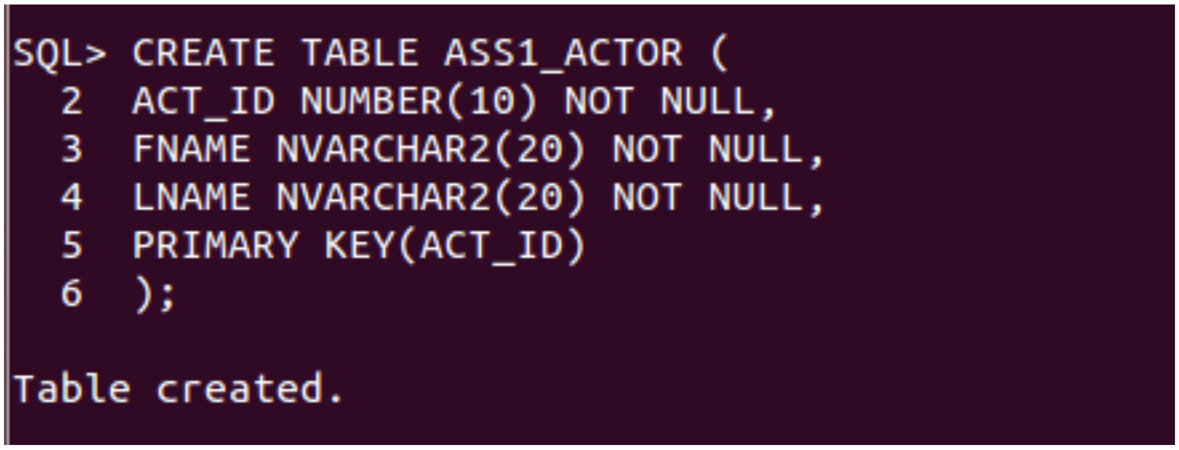
```
SQL> CREATE TABLE ASS1_FILM_CHAR (
  2  CHAR_ID NUMBER(10) NOT NULL,
  3  FNAME NVARCHAR2(20) NOT NULL,
  4  LNAME NVARCHAR2(20) NOT NULL,
  5  PRIMARY KEY(CHAR_ID)
  6  );

Table created.
```

10. View table description:



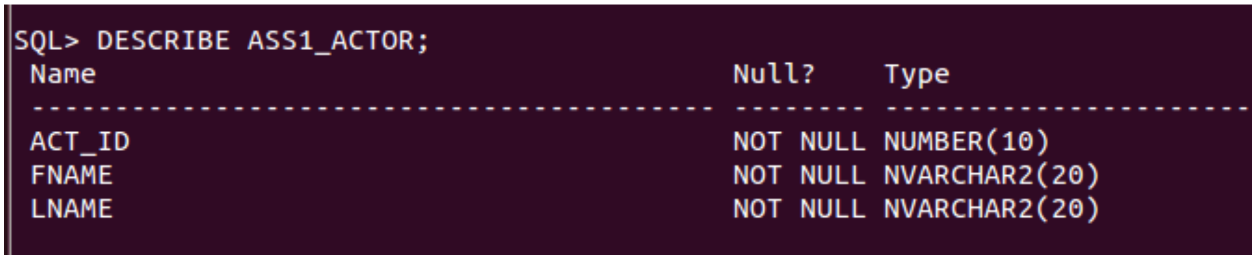
```
SQL> DESCRIBE ASS1_FILM_CHAR;
 Name                                      Null?    Type
 ----------------------------------------- -------- ---------------------
 CHAR_ID                                   NOT NULL NUMBER(10)
 FNAME                                     NOT NULL NVARCHAR2(20)
 LNAME                                     NOT NULL NVARCHAR2(20)
```

11. Create table ASS1_ACTOR:



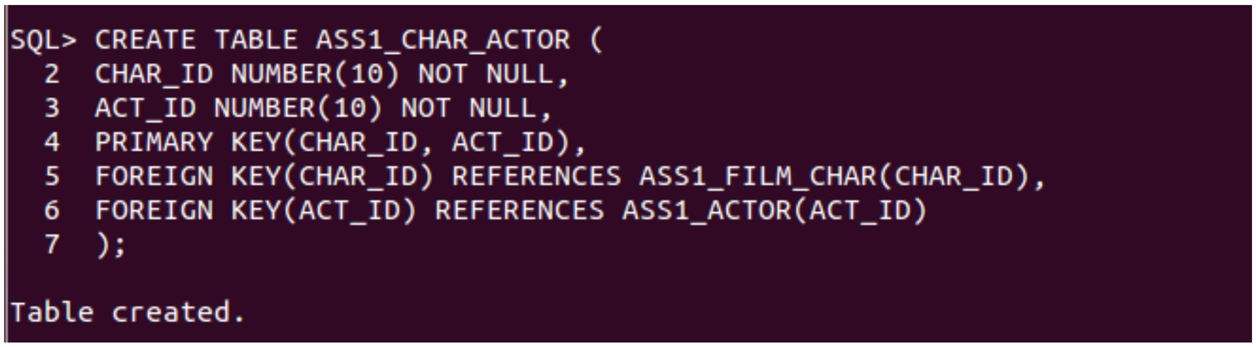
```
SQL> CREATE TABLE ASS1_ACTOR (
  2  ACT_ID NUMBER(10) NOT NULL,
  3  FNAME NVARCHAR2(20) NOT NULL,
  4  LNAME NVARCHAR2(20) NOT NULL,
  5  PRIMARY KEY(ACT_ID)
  6  );

Table created.
```

12. View table description:



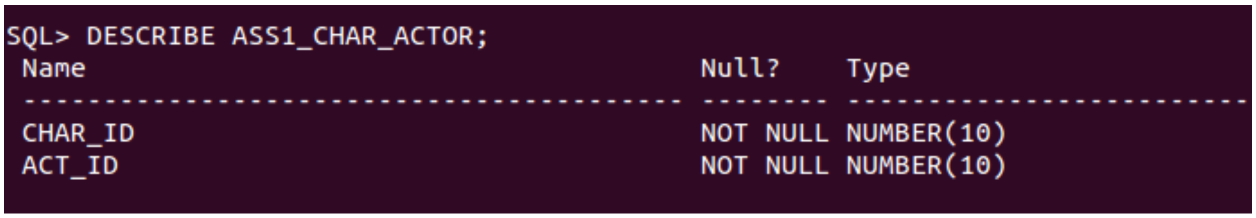
```
SQL> DESCRIBE ASS1_ACTOR;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ACT_ID                                    NOT NULL NUMBER(10)
 FNAME                                     NOT NULL NVARCHAR2(20)
 LNAME                                     NOT NULL NVARCHAR2(20)
```

13. Creation of ASS1_CHAR_ACTOR:



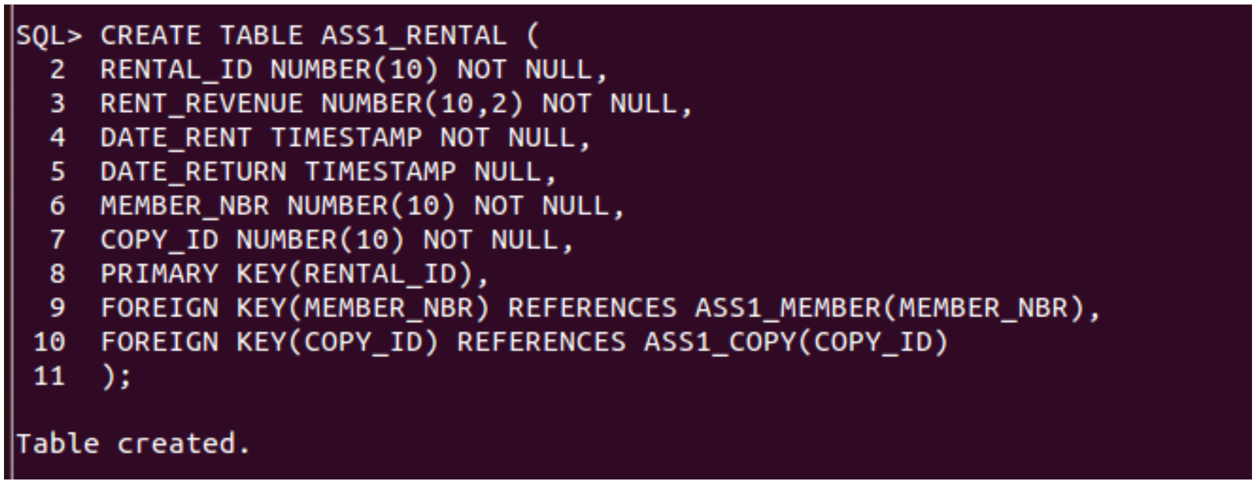
```
SQL> CREATE TABLE ASS1_CHAR_ACTOR (
  2  CHAR_ID NUMBER(10) NOT NULL,
  3  ACT_ID NUMBER(10) NOT NULL,
  4  PRIMARY KEY(CHAR_ID, ACT_ID),
  5  FOREIGN KEY(CHAR_ID) REFERENCES ASS1_FILM_CHAR(CHAR_ID),
  6  FOREIGN KEY(ACT_ID) REFERENCES ASS1_ACTOR(ACT_ID)
  7  );

Table created.
```

14. View table description of CHAR_ACTOR:



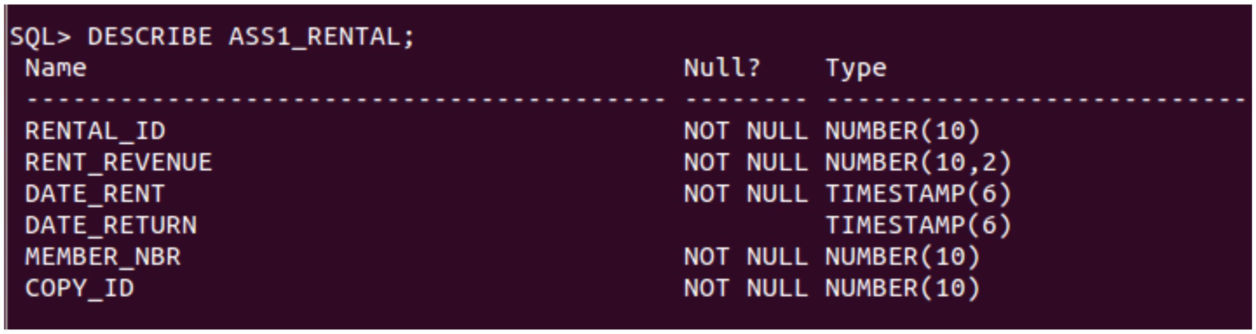
```
SQL> DESCRIBE ASS1_CHAR_ACTOR;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CHAR_ID                                   NOT NULL NUMBER(10)
 ACT_ID                                    NOT NULL NUMBER(10)
```

15. Creation of ASS1_RENTAL table:



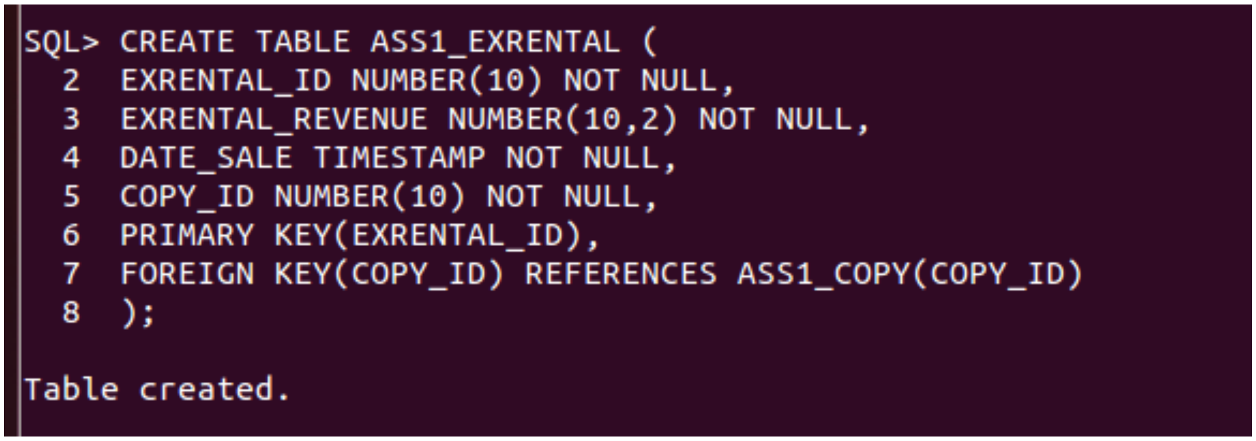
```
SQL> CREATE TABLE ASS1_RENTAL (
RENTAL_ID NUMBER(10) NOT NULL,
RENT_REVENUE NUMBER(10,2) NOT NULL,
DATE_RENT TIMESTAMP NOT NULL,
DATE_RETURN TIMESTAMP NULL,
MEMBER_NBR NUMBER(10) NOT NULL,
COPY_ID NUMBER(10) NOT NULL,
PRIMARY KEY(RENTAL_ID),
FOREIGN KEY(MEMBER_NBR) REFERENCES ASS1_MEMBER(MEMBER_NBR),
FOREIGN KEY(COPY_ID) REFERENCES ASS1_COPY(COPY_ID)
);

Table created.
```

16. View table description:



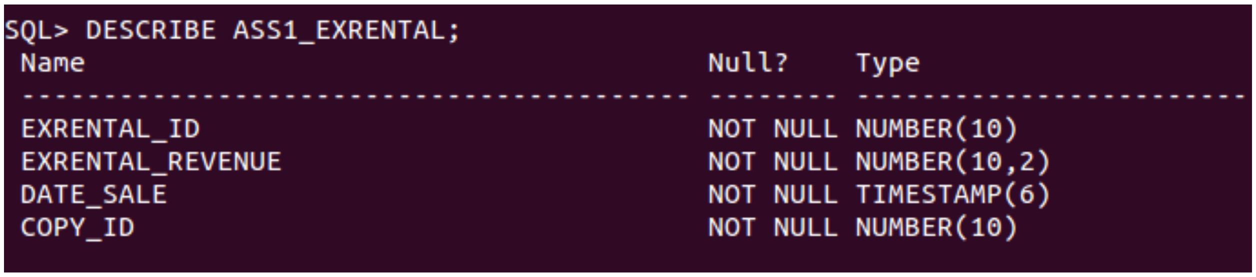
```
SQL> DESCRIBE ASS1_RENTAL;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 RENTAL_ID                                 NOT NULL NUMBER(10)
 RENT_REVENUE                              NOT NULL NUMBER(10,2)
 DATE_RENT                                 NOT NULL TIMESTAMP(6)
 DATE_RETURN                                        TIMESTAMP(6)
 MEMBER_NBR                                NOT NULL NUMBER(10)
 COPY_ID                                   NOT NULL NUMBER(10)
```

17. Create ASS1_EXRENTAL table:

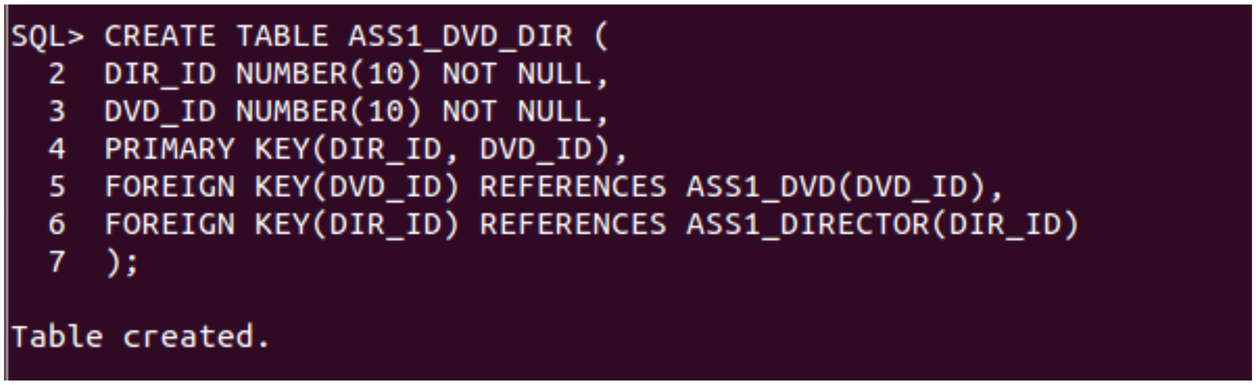


```
SQL> CREATE TABLE ASS1_EXRENTAL (
EXRENTAL_ID NUMBER(10) NOT NULL,
EXRENTAL_REVENUE NUMBER(10,2) NOT NULL,
DATE_SALE TIMESTAMP NOT NULL,
COPY_ID NUMBER(10) NOT NULL,
PRIMARY KEY(EXRENTAL_ID),
FOREIGN KEY(COPY_ID) REFERENCES ASS1_COPY(COPY_ID)
);

Table created.
```

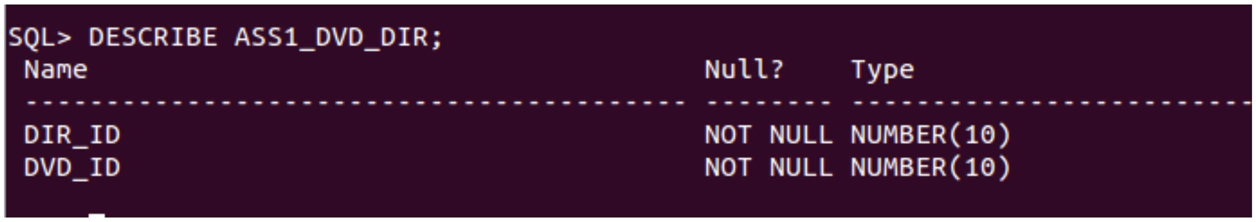
18. View table description:


```
SQL> DESCRIBE ASS1_EXRENTAL;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EXRENTAL_ID                               NOT NULL NUMBER(10)
 EXRENTAL_REVENUE                          NOT NULL NUMBER(10,2)
 DATE_SALE                                 NOT NULL TIMESTAMP(6)
 COPY_ID                                   NOT NULL NUMBER(10)
```

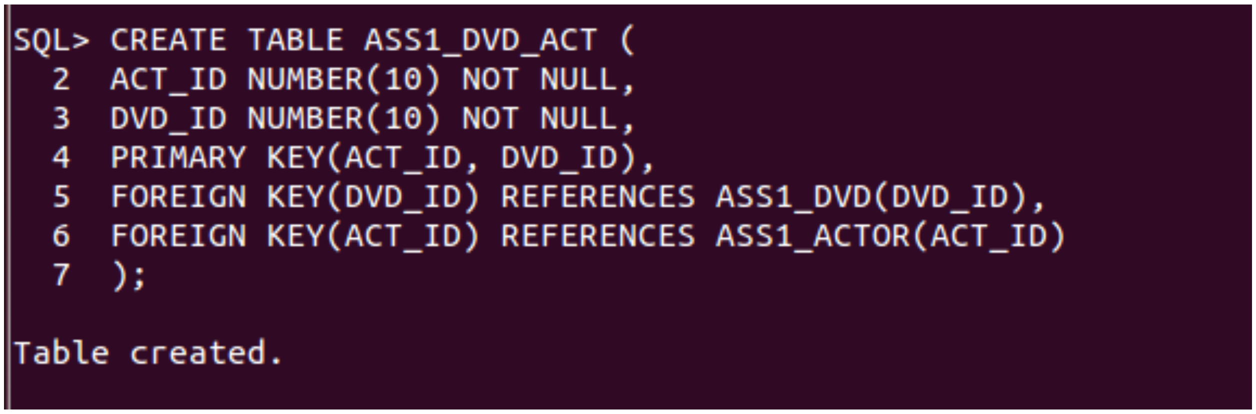
19. Creation of ASS1_DVD_DIR junction table:


```
SQL> CREATE TABLE ASS1_DVD_DIR (
  2  DIR_ID NUMBER(10) NOT NULL,
  3  DVD_ID NUMBER(10) NOT NULL,
  4  PRIMARY KEY(DIR_ID, DVD_ID),
  5  FOREIGN KEY(DVD_ID) REFERENCES ASS1_DVD(DVD_ID),
  6  FOREIGN KEY(DIR_ID) REFERENCES ASS1_DIRECTOR(DIR_ID)
  7  );

Table created.
```

20. View table description:

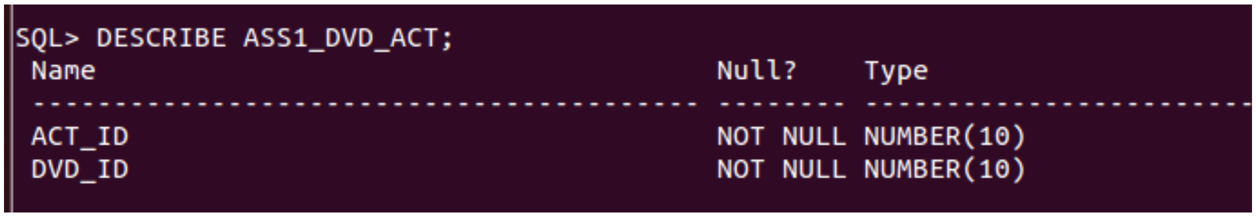

```
SQL> DESCRIBE ASS1_DVD_DIR;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DIR_ID                                    NOT NULL NUMBER(10)
 DVD_ID                                    NOT NULL NUMBER(10)
```

21. Creation of ASS1_DVD_ACT junction table:



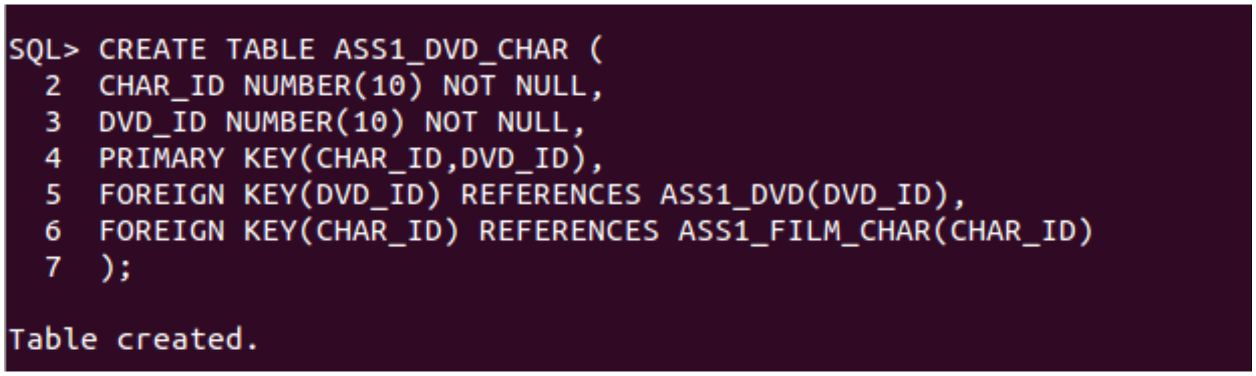
```
SQL> CREATE TABLE ASS1_DVD_ACT (
  2  ACT_ID NUMBER(10) NOT NULL,
  3  DVD_ID NUMBER(10) NOT NULL,
  4  PRIMARY KEY(ACT_ID, DVD_ID),
  5  FOREIGN KEY(DVD_ID) REFERENCES ASS1_DVD(DVD_ID),
  6  FOREIGN KEY(ACT_ID) REFERENCES ASS1_ACTOR(ACT_ID)
  7  );

Table created.
```

22. View table description:



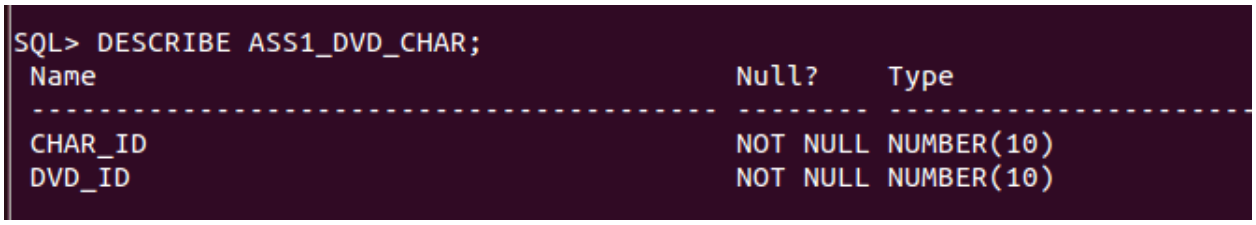
```
SQL> DESCRIBE ASS1_DVD_ACT;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 ACT_ID                                    NOT NULL NUMBER(10)
 DVD_ID                                    NOT NULL NUMBER(10)
```

23. Creation of ASS1_DVD_CHAR junction table:



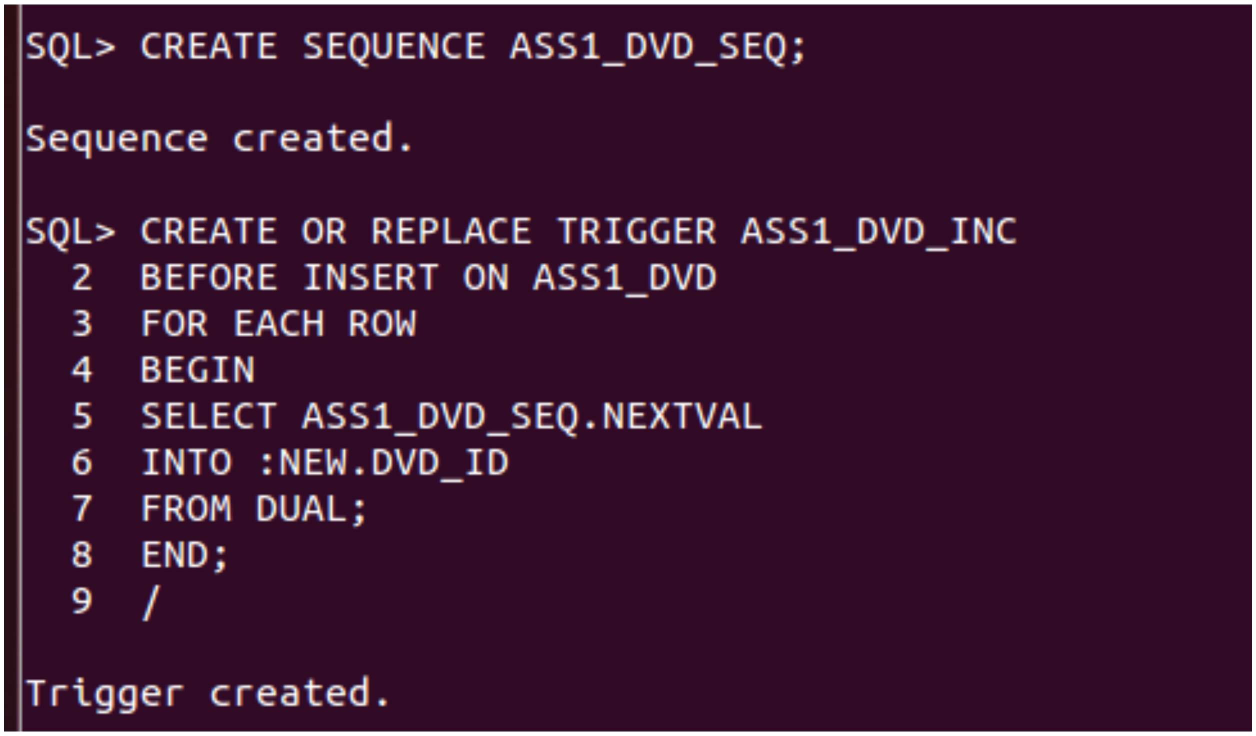
```
SQL> CREATE TABLE ASS1_DVD_CHAR (
  2  CHAR_ID NUMBER(10) NOT NULL,
  3  DVD_ID NUMBER(10) NOT NULL,
  4  PRIMARY KEY(CHAR_ID,DVD_ID),
  5  FOREIGN KEY(DVD_ID) REFERENCES ASS1_DVD(DVD_ID),
  6  FOREIGN KEY(CHAR_ID) REFERENCES ASS1_FILM_CHAR(CHAR_ID)
  7  );

Table created.
```

24. View table description:



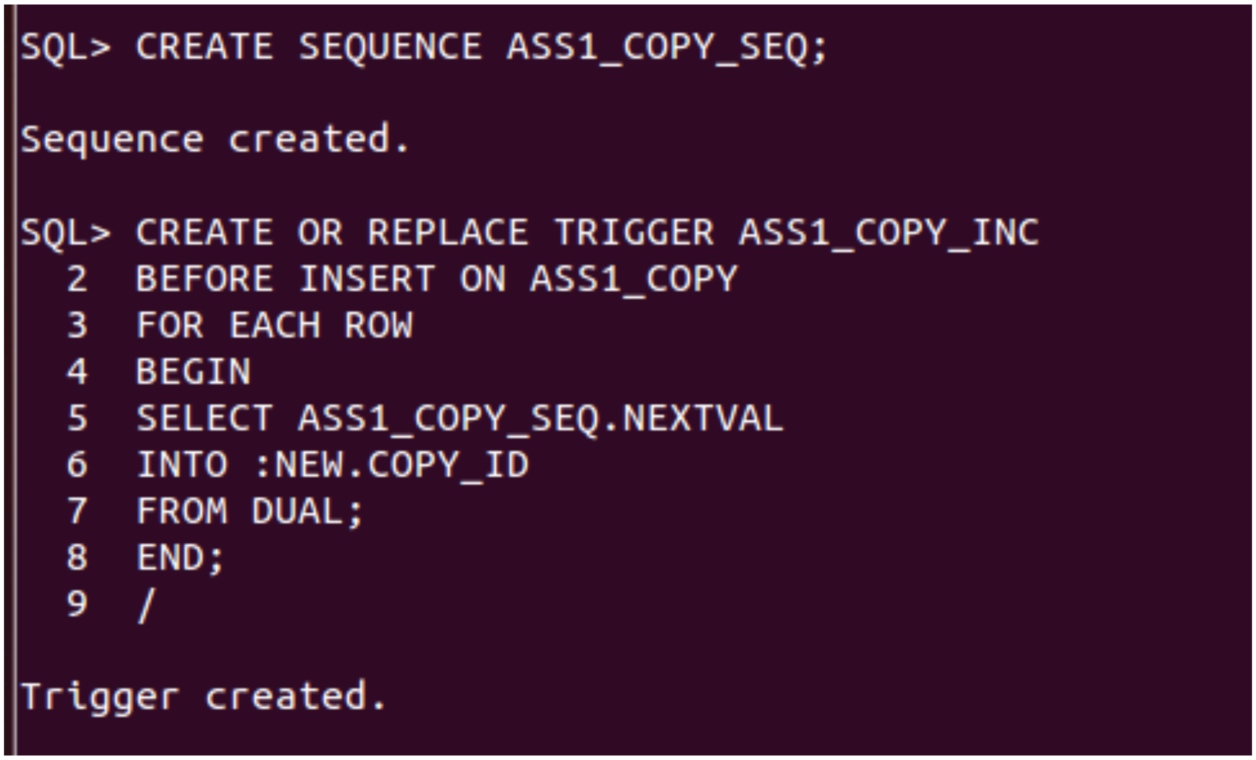
```
SQL> DESCRIBE ASS1_DVD_CHAR;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 CHAR_ID                                   NOT NULL NUMBER(10)
 DVD_ID                                    NOT NULL NUMBER(10)
```

25. Creation of sequence and trigger to auto-increment DVD ID when new insertions are made into ASS1_DVD table:

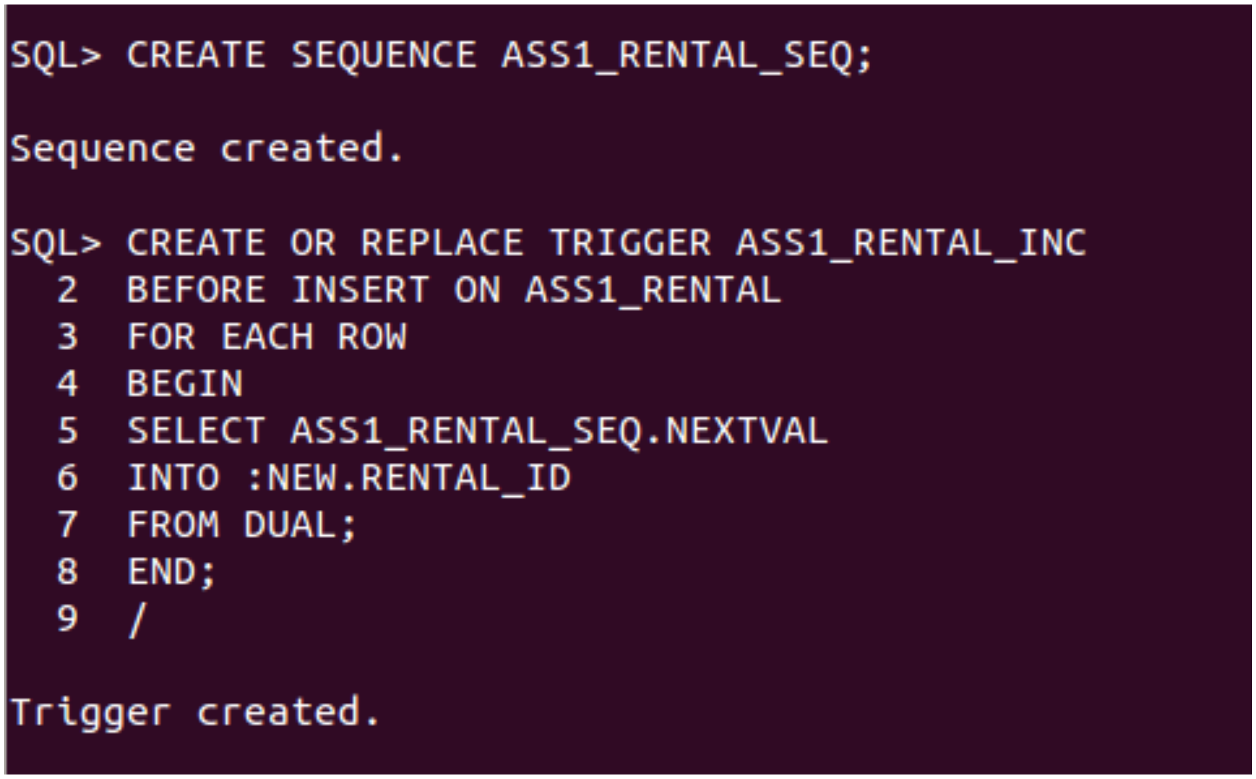


```
SQL> CREATE SEQUENCE ASS1_DVD_SEQ;

Sequence created.

SQL> CREATE OR REPLACE TRIGGER ASS1_DVD_INC
  2  BEFORE INSERT ON ASS1_DVD
  3  FOR EACH ROW
  4  BEGIN
  5  SELECT ASS1_DVD_SEQ.NEXTVAL
  6  INTO :NEW.DVD_ID
  7  FROM DUAL;
  8  END;
  9  /

Trigger created.
```

26. Creation of sequence and trigger to auto-increment Copy_ID when new insertions are made into ASS1_COPY table:

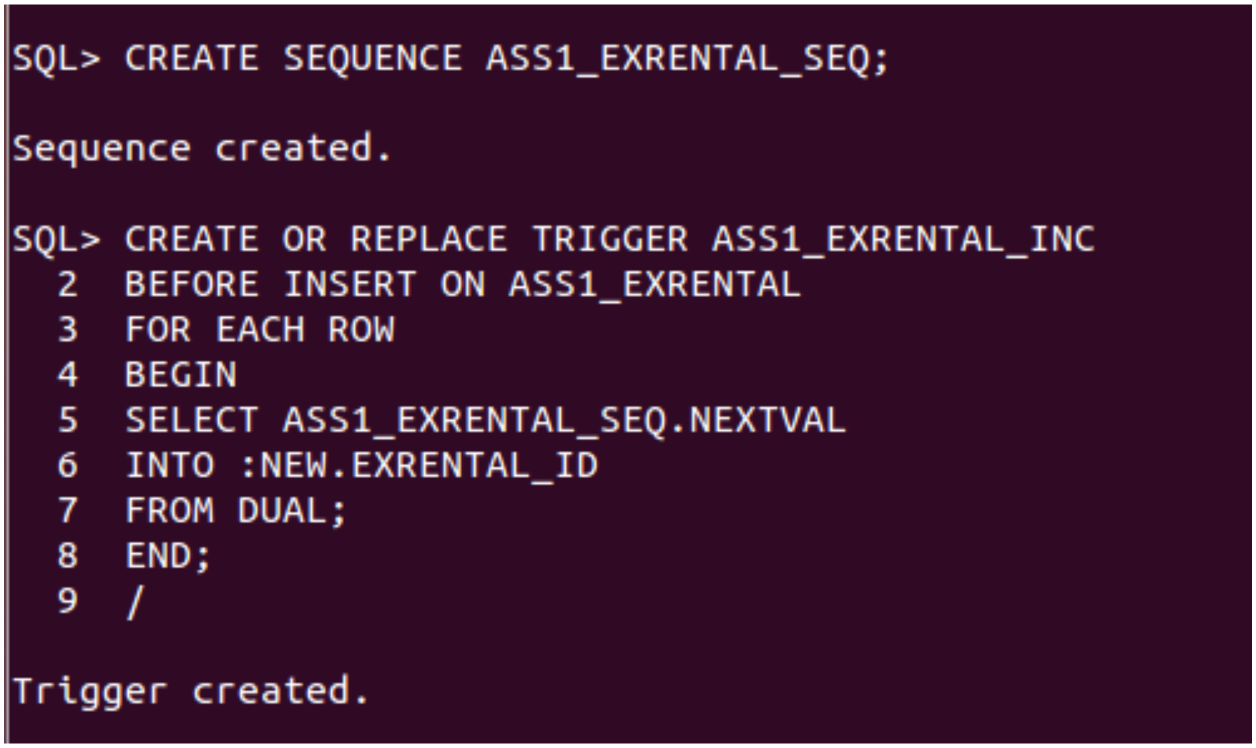

```
SQL> CREATE SEQUENCE ASS1_COPY_SEQ;

Sequence created.

SQL> CREATE OR REPLACE TRIGGER ASS1_COPY_INC
  2  BEFORE INSERT ON ASS1_COPY
  3  FOR EACH ROW
  4  BEGIN
  5  SELECT ASS1_COPY_SEQ.NEXTVAL
  6  INTO :NEW.COPY_ID
  7  FROM DUAL;
  8  END;
  9  /

Trigger created.
```

27. Creation of sequence and trigger to auto-increment Rental ID when new insertions are made into ASS1_Rental table:



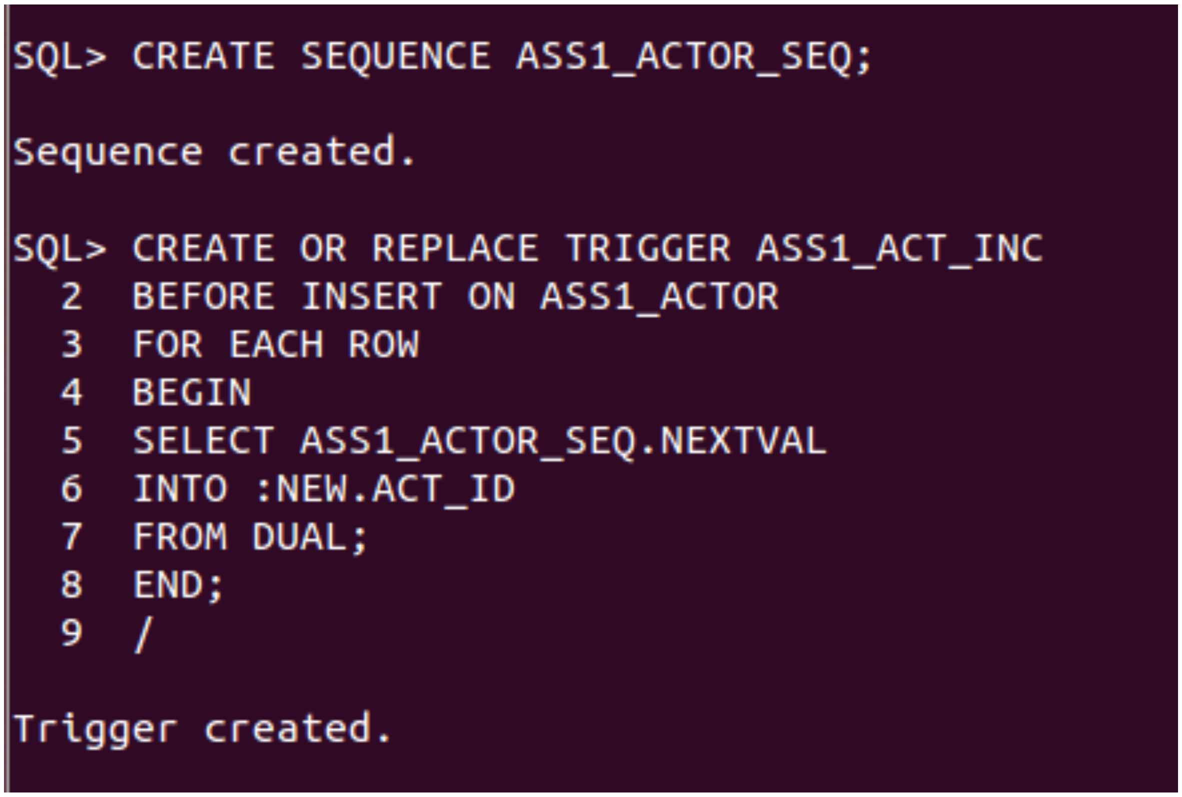
```
SQL> CREATE SEQUENCE ASS1_RENTAL_SEQ;

Sequence created.

SQL> CREATE OR REPLACE TRIGGER ASS1_RENTAL_INC
  2  BEFORE INSERT ON ASS1_RENTAL
  3  FOR EACH ROW
  4  BEGIN
  5  SELECT ASS1_RENTAL_SEQ.NEXTVAL
  6  INTO :NEW.RENTAL_ID
  7  FROM DUAL;
  8  END;
  9  /

Trigger created.
```

28. Creation of sequence and trigger to auto-increment Exrental ID when new insertions are made into ASS1_Exrental table:



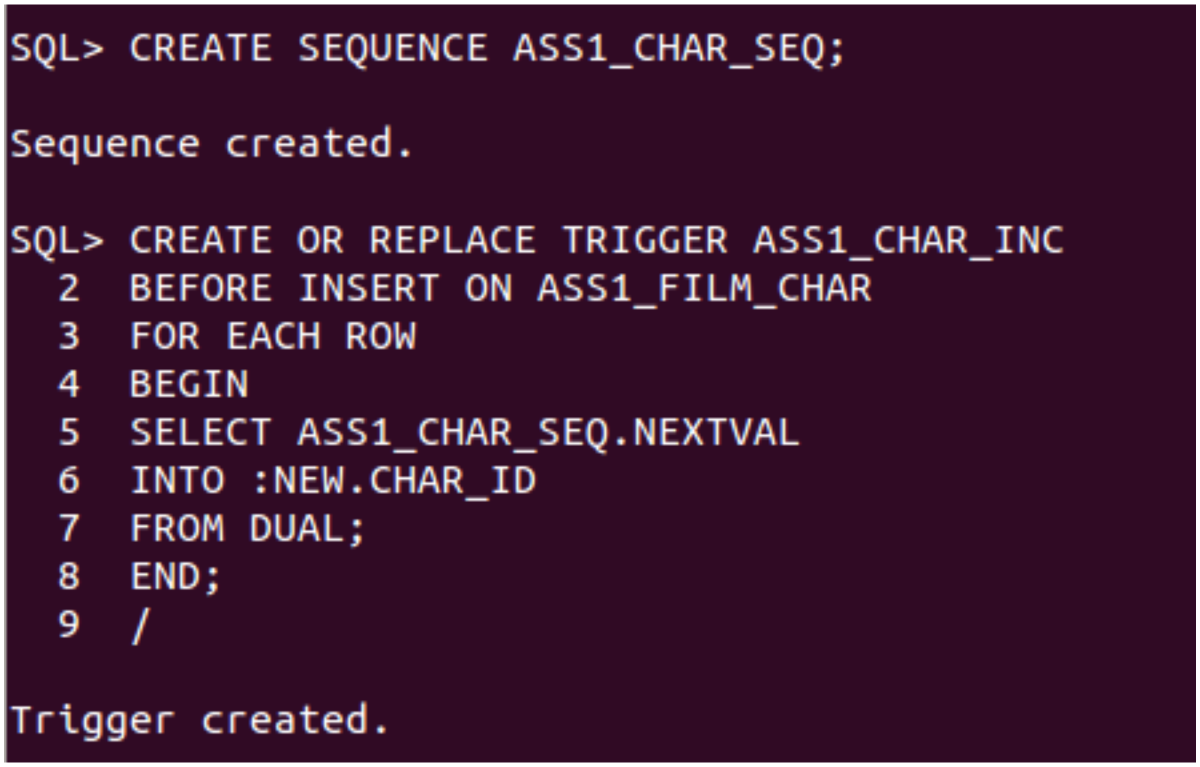
```
SQL> CREATE SEQUENCE ASS1_EXRENTAL_SEQ;

Sequence created.

SQL> CREATE OR REPLACE TRIGGER ASS1_EXRENTAL_INC
  2  BEFORE INSERT ON ASS1_EXRENTAL
  3  FOR EACH ROW
  4  BEGIN
  5  SELECT ASS1_EXRENTAL_SEQ.NEXTVAL
  6  INTO :NEW.EXRENTAL_ID
  7  FROM DUAL;
  8  END;
  9  /

Trigger created.
```

29. Creation of sequence and trigger to auto-increment Actor ID when new insertions are made into ASS1_ACTOR table:



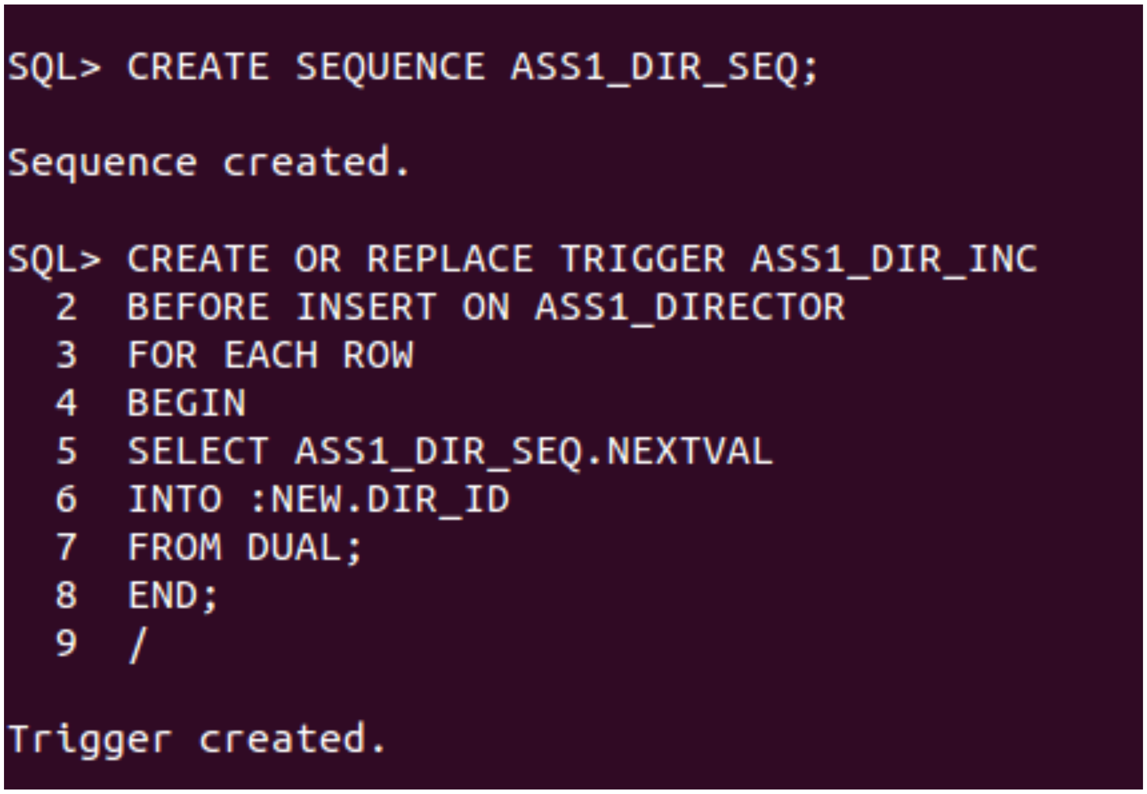
```
SQL> CREATE SEQUENCE ASS1_ACTOR_SEQ;

Sequence created.

SQL> CREATE OR REPLACE TRIGGER ASS1_ACT_INC
     BEFORE INSERT ON ASS1_ACTOR
     FOR EACH ROW
     BEGIN
     SELECT ASS1_ACTOR_SEQ.NEXTVAL
     INTO :NEW.ACT_ID
     FROM DUAL;
     END;
     /

Trigger created.
```

30. Creation of sequence and trigger to auto-increment CHAR ID when new insertions are made into ASS1_FILM_CHAR table:



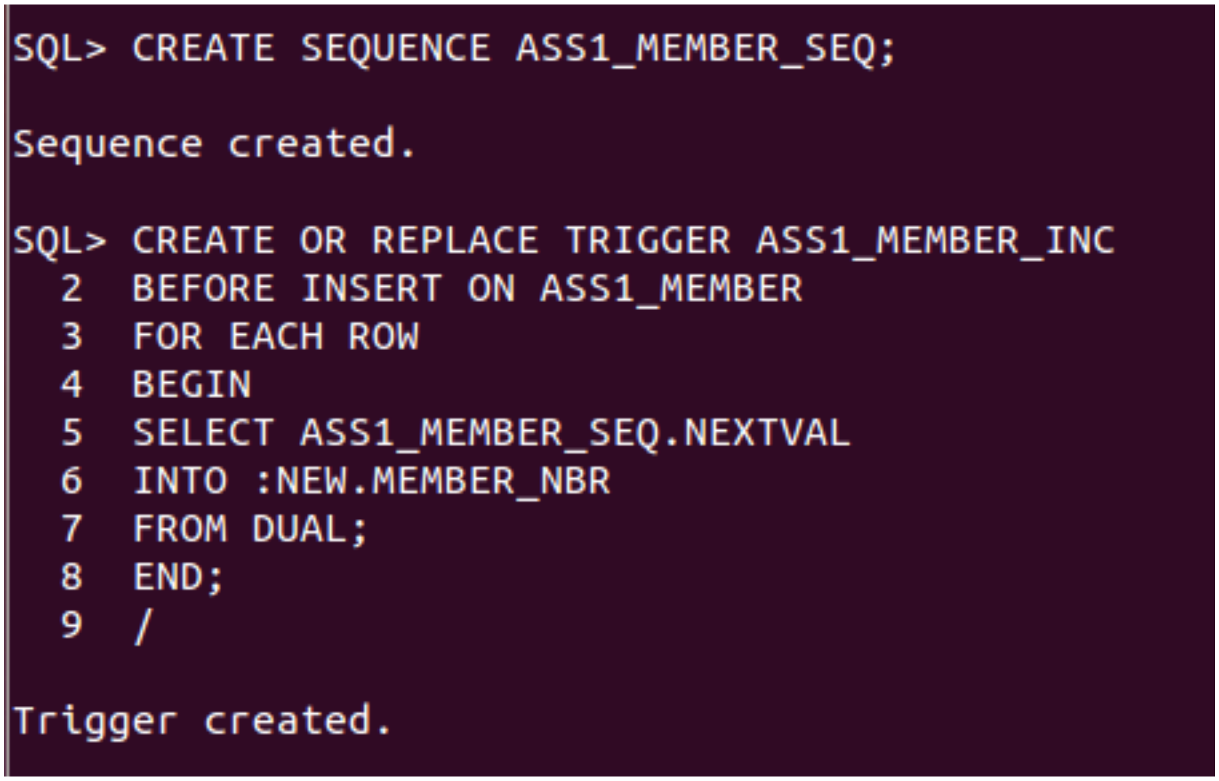
```
SQL> CREATE SEQUENCE ASS1_CHAR_SEQ;

Sequence created.

SQL> CREATE OR REPLACE TRIGGER ASS1_CHAR_INC
BEFORE INSERT ON ASS1_FILM_CHAR
FOR EACH ROW
BEGIN
SELECT ASS1_CHAR_SEQ.NEXTVAL
INTO :NEW.CHAR_ID
FROM DUAL;
END;
/

Trigger created.
```

31. Creation of sequence and trigger to auto-increment DIRECTOR ID when new insertions are made into ASS1_DIRECTOR table:



```
SQL> CREATE SEQUENCE ASS1_DIR_SEQ;

Sequence created.

SQL> CREATE OR REPLACE TRIGGER ASS1_DIR_INC
BEFORE INSERT ON ASS1_DIRECTOR
FOR EACH ROW
BEGIN
SELECT ASS1_DIR_SEQ.NEXTVAL
INTO :NEW.DIR_ID
FROM DUAL;
END;
/

Trigger created.
```

32. Creation of sequence and trigger to auto-increment MEMBER ID when new insertions are made into ASS1_MEMBER table:

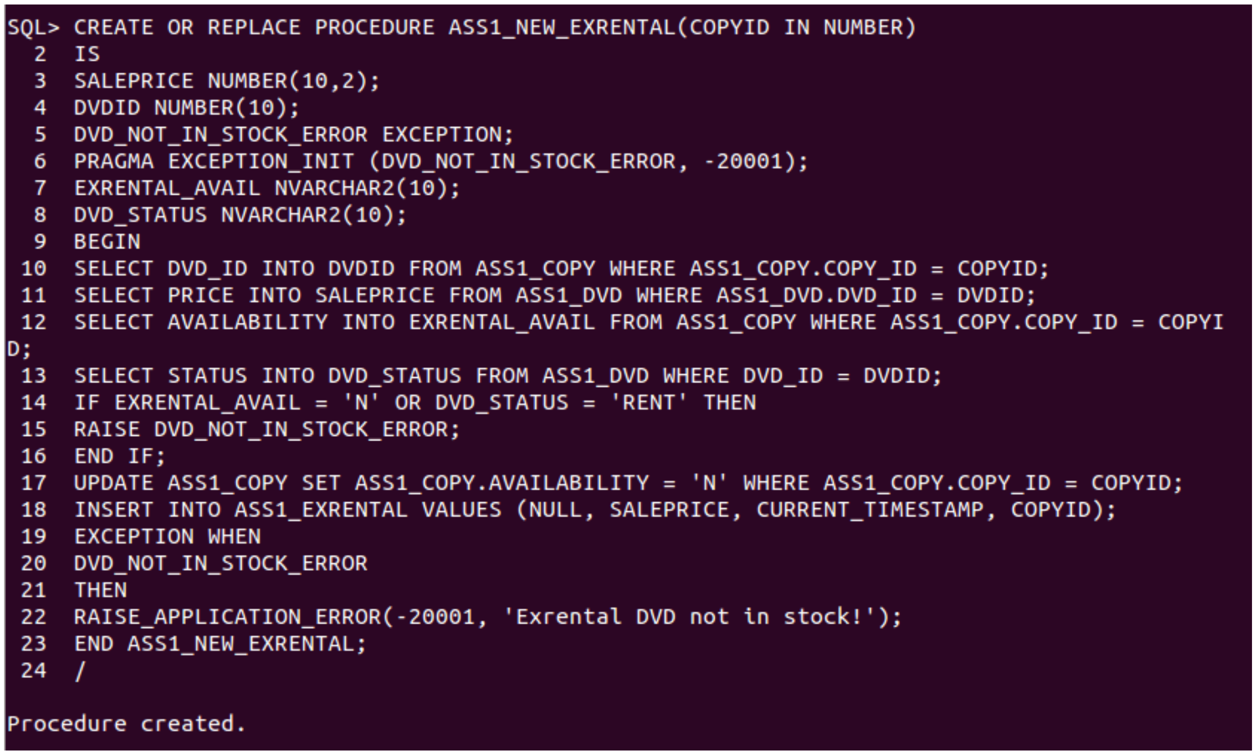


```
SQL> CREATE SEQUENCE ASS1_MEMBER_SEQ;

Sequence created.

SQL> CREATE OR REPLACE TRIGGER ASS1_MEMBER_INC
BEFORE INSERT ON ASS1_MEMBER
FOR EACH ROW
BEGIN
SELECT ASS1_MEMBER_SEQ.NEXTVAL
INTO :NEW.MEMBER_NBR
FROM DUAL;
END;
/

Trigger created.
```

Below is a procedure created with the purpose of automating some of updates that need to happen every time an exrental is sold, which are:

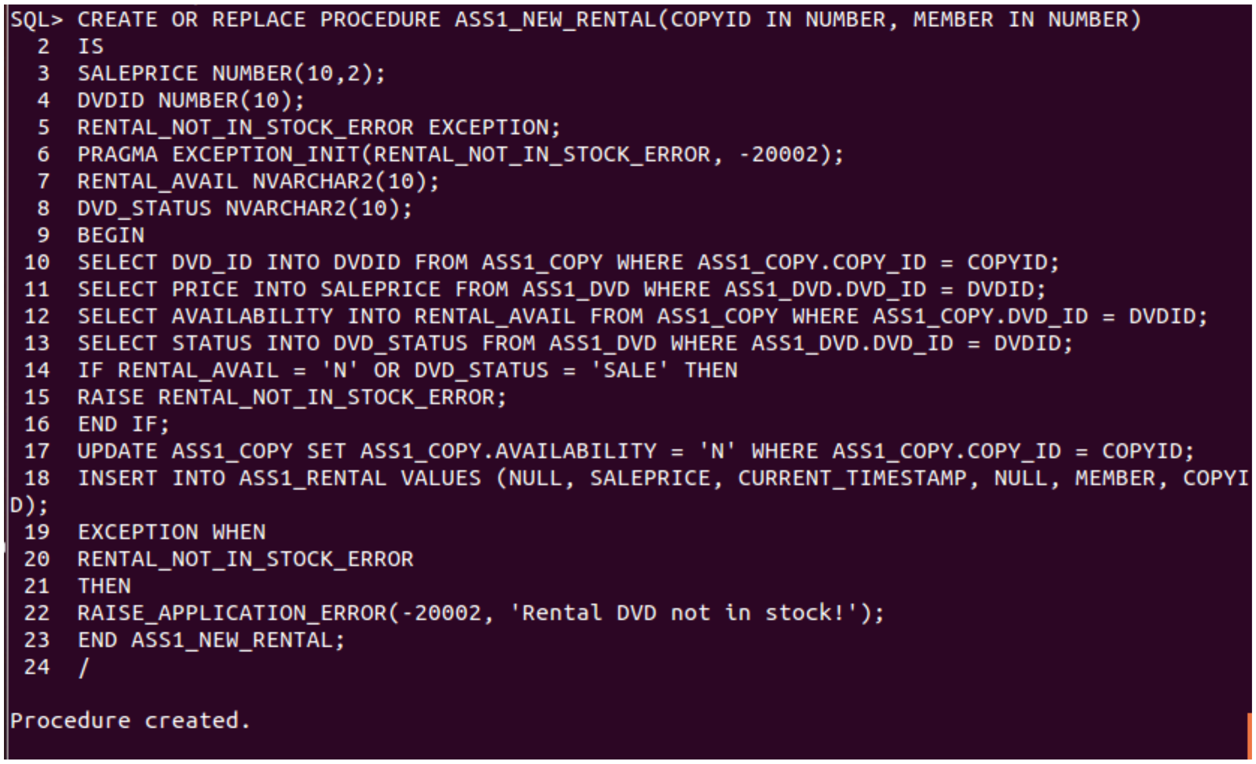
1. Check that the copy of the exrental is available (i.e. ‘AVAILABILITY’ in ASS1_COPY table is set to ‘Y’. Otherwise, generate ‘DVD_NOT_IN_STOCK’ error.
2. Check that the commercial status of the copy is indeed for ‘sale’ and not for ‘rent’
3. If both checks are carried out, and the copy is for sale and available, update its availability to ‘N’ to reflect that it is sold, and also perform an insertion in our ‘Exrental’ table.

This procedure takes in the copy_id of the exrental as the parameter, and generates a new exrental record, using this information to generate it. It also updates the availability status in the Copy table.



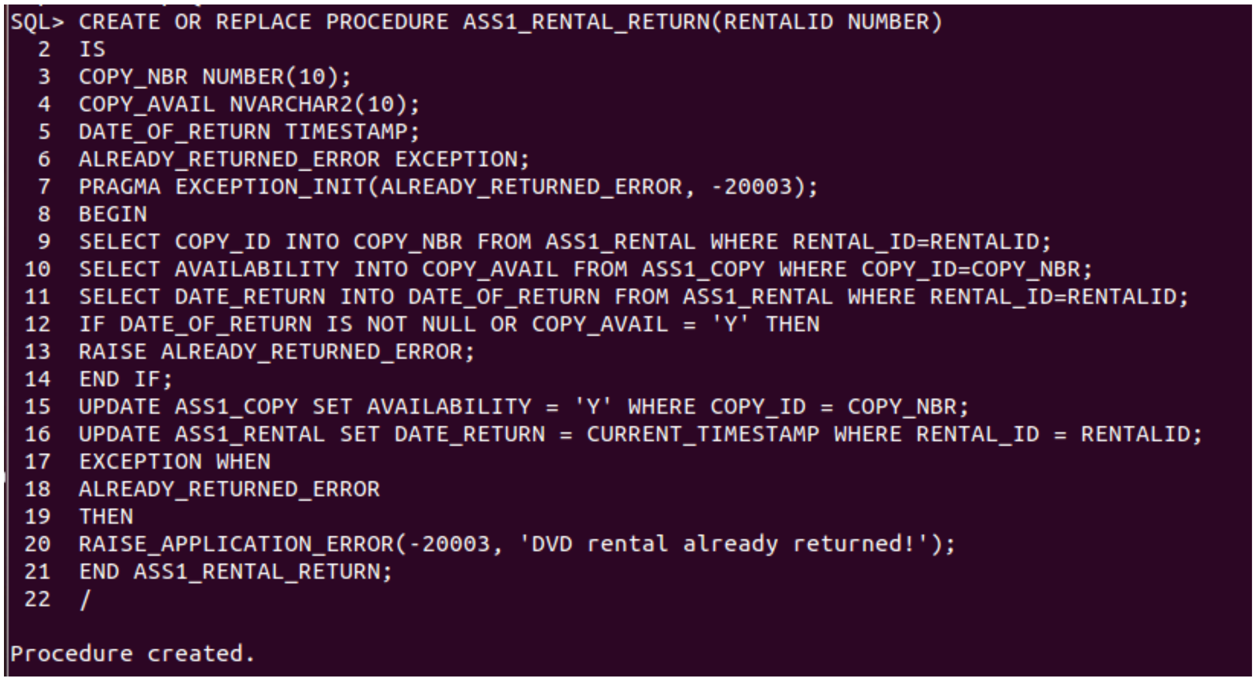
```
SQL> CREATE OR REPLACE PROCEDURE ASS1_NEW_EXRENTAL(COPYID IN NUMBER)
IS
SALEPRICE NUMBER(10,2);
DVDID NUMBER(10);
DVD_NOT_IN_STOCK_ERROR EXCEPTION;
PRAGMA EXCEPTION_INIT (DVD_NOT_IN_STOCK_ERROR, -20001);
EXRENTAL_AVAIL NVARCHAR2(10);
DVD_STATUS NVARCHAR2(10);
BEGIN
SELECT DVD_ID INTO DVDID FROM ASS1_COPY WHERE ASS1_COPY.COPY_ID = COPYID;
SELECT PRICE INTO SALEPRICE FROM ASS1_DVD WHERE ASS1_DVD.DVD_ID = DVDID;
SELECT AVAILABILITY INTO EXRENTAL_AVAIL FROM ASS1_COPY WHERE ASS1_COPY.COPY_ID = COPYI
D;
SELECT STATUS INTO DVD_STATUS FROM ASS1_DVD WHERE DVD_ID = DVDID;
IF EXRENTAL_AVAIL = 'N' OR DVD_STATUS = 'RENT' THEN
RAISE DVD_NOT_IN_STOCK_ERROR;
END IF;
UPDATE ASS1_COPY SET ASS1_COPY.AVAILABILITY = 'N' WHERE ASS1_COPY.COPY_ID = COPYID;
INSERT INTO ASS1_EXRENTAL VALUES (NULL, SALEPRICE, CURRENT_TIMESTAMP, COPYID);
EXCEPTION WHEN
DVD_NOT_IN_STOCK_ERROR
THEN
RAISE_APPLICATION_ERROR(-20001, 'Exrental DVD not in stock!');
END ASS1_NEW_EXRENTAL;
/

Procedure created.
```

This procedure facilitates a new DVD rental transaction. It first ensures that the desired copy is ‘for rent’ and that it is currently available (‘Y’). If so, it completes an insertion into the ASS1_RENTAL table and also sets the availability of the copy to ‘N’ until the member returns the copy. It takes copy_id of the rental, and member_nbr of the member renting the DVD, as arguments.



```
SQL> CREATE OR REPLACE PROCEDURE ASS1_NEW_RENTAL(COPYID IN NUMBER, MEMBER IN NUMBER)
     IS
     SALEPRICE NUMBER(10,2);
     DVDID NUMBER(10);
     RENTAL_NOT_IN_STOCK_ERROR EXCEPTION;
     PRAGMA EXCEPTION_INIT(RENTAL_NOT_IN_STOCK_ERROR, -20002);
     RENTAL_AVAIL NVARCHAR2(10);
     DVD_STATUS NVARCHAR2(10);
     BEGIN
     SELECT DVD_ID INTO DVDID FROM ASS1_COPY WHERE ASS1_COPY.COPY_ID = COPYID;
     SELECT PRICE INTO SALEPRICE FROM ASS1_DVD WHERE ASS1_DVD.DVD_ID = DVDID;
     SELECT AVAILABILITY INTO RENTAL_AVAIL FROM ASS1_COPY WHERE ASS1_COPY.DVD_ID = DVDID;
     SELECT STATUS INTO DVD_STATUS FROM ASS1_DVD WHERE ASS1_DVD.DVD_ID = DVDID;
     IF RENTAL_AVAIL = 'N' OR DVD_STATUS = 'SALE' THEN
     RAISE RENTAL_NOT_IN_STOCK_ERROR;
     END IF;
     UPDATE ASS1_COPY SET ASS1_COPY.AVAILABILITY = 'N' WHERE ASS1_COPY.COPY_ID = COPYID;
     INSERT INTO ASS1_RENTAL VALUES (NULL, SALEPRICE, CURRENT_TIMESTAMP, NULL, MEMBER, COPYI
D);
     EXCEPTION WHEN
     RENTAL_NOT_IN_STOCK_ERROR
     THEN
     RAISE_APPLICATION_ERROR(-20002, 'Rental DVD not in stock!');
     END ASS1_NEW_RENTAL;
     /

Procedure created.
```

Finally, this procedure updates the availability status of a copy when a member returns a rental. It also updates the date of return in the Rental table using the current date and time (‘current timestamp’). It takes the rental_id of the DVD rental as an argument.



```
SQL> CREATE OR REPLACE PROCEDURE ASS1_RENTAL_RETURN(RENTALID NUMBER)
IS
COPY_NBR NUMBER(10);
COPY_AVAIL NVARCHAR2(10);
DATE_OF_RETURN TIMESTAMP;
ALREADY_RETURNED_ERROR EXCEPTION;
PRAGMA EXCEPTION_INIT(ALREADY_RETURNED_ERROR, -20003);
BEGIN
SELECT COPY_ID INTO COPY_NBR FROM ASS1_RENTAL WHERE RENTAL_ID=RENTALID;
SELECT AVAILABILITY INTO COPY_AVAIL FROM ASS1_COPY WHERE COPY_ID=COPY_NBR;
SELECT DATE_RETURN INTO DATE_OF_RETURN FROM ASS1_RENTAL WHERE RENTAL_ID=RENTALID;
IF DATE_OF_RETURN IS NOT NULL OR COPY_AVAIL = 'Y' THEN
RAISE ALREADY_RETURNED_ERROR;
END IF;
UPDATE ASS1_COPY SET AVAILABILITY = 'Y' WHERE COPY_ID = COPY_NBR;
UPDATE ASS1_RENTAL SET DATE_RETURN = CURRENT_TIMESTAMP WHERE RENTAL_ID = RENTALID;
EXCEPTION WHEN
ALREADY_RETURNED_ERROR
THEN
RAISE_APPLICATION_ERROR(-20003, 'DVD rental already returned!');
END ASS1_RENTAL_RETURN;
/

Procedure created.
```