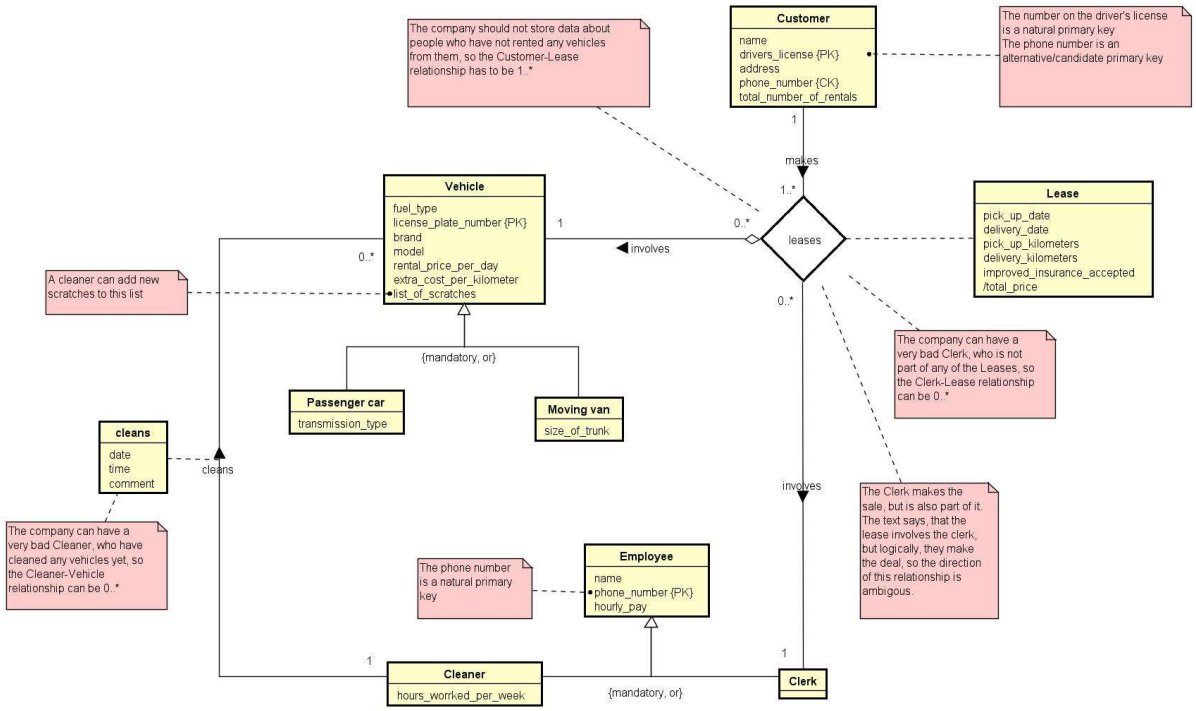
Exercise 1 – EER (20%)



To find the entities, I looked for the common names (marked red)

To find the attributes, I looked for nouns related to a specific entity (marked green)

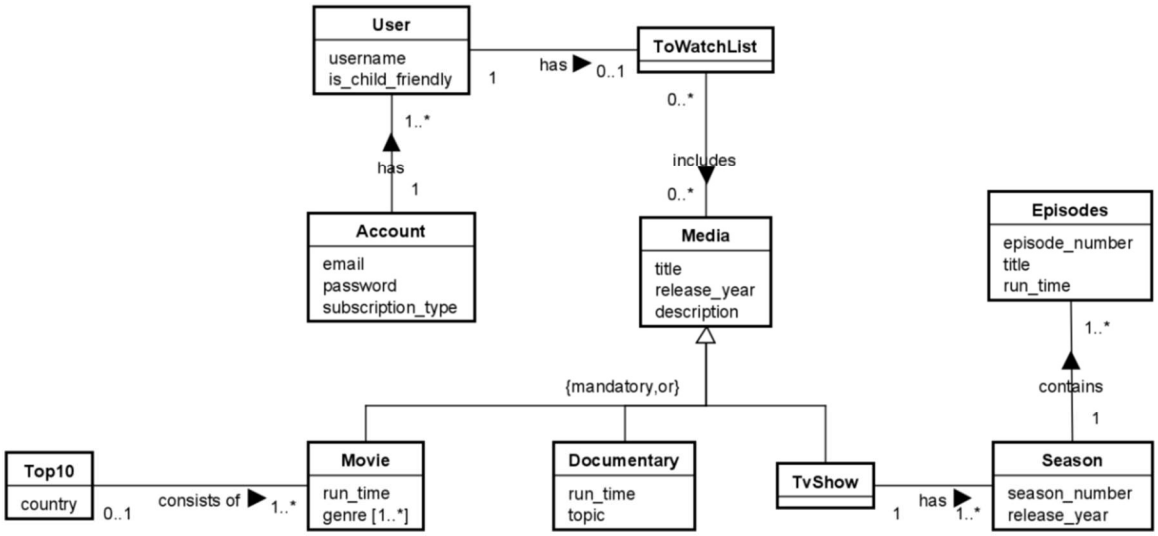
For relationships, I looked for verbs between entities (marked blue), but most of them were not included in the EER, as these interactions do not need to be stored. (e.g. Clerk servicing Customer)

Here at VIA Car Rental we rent out cars. We are currently using spreadsheets for our data, but figured it was time to upgrade, as our business grows. Here's the information we need to keep track of. We rent out vehicles, two types, that is. We have passenger cars, and moving vans, depending on people's needs. For any vehicle, we track the following information: the fuel type, license plate number, brand and model. Each vehicle has a fixed rental-price per day, and furthermore there's an extra cost for each kilometer driven. For the passenger cars we have both manual and automatic transmissions, so that information must be accessible. For the vans, the size of the trunk in cubic meters is relevant. Customers rent cars by obtaining leases created by our clerks. For customers we need their names, driver's license, address, and phone number. We also track their total number of rentals for bonus purposes. A lease involves the leased vehicle, a customer, and the clerk creating the lease. We need the date of when the vehicle is picked up, and when it will be delivered. We mark the kilometers driven of the vehicle at pick up, and again at return, to calculate the extra expenses for kilometers driven. We offer improved insurance, so we must include, whether the customer accepted. Upon return the total price of the lease is calculated, based on days rented, the extra insurance, and how many kilometers the customer drove. We need to know which clerk created the lease in case of complaints or other problems. As already hinted, we have employees. The clerks service the customers, managing leases and such. They are full-time employees. We also have part-time employees, our cleaners, they clean the cars after their return. For cleaners we track how many hours per week they work. Cleaners clean cars, whenever they are returned. We need the time and date for each cleaning, and sometimes cleaners leave comments too, e.g. the vehicle was extra muddy. If the cleaner finds new scratches on the vehicle, they add a description of the scratch to the vehicle's list of prior scratches. We have other

employees too, e.g. accountants or customer service, but they do not have unique responsibilities regarding cars or leases, so we just consider them employees. For all employees we care about their name, phone number and hourly pay.

Exercise 2 – EER to Relational Schema/GRD (15%)

In this exercise your task is to map the below EER diagram to a Relational Schema/Logical Model (Figure 17.8 in the book).



Give a detailed explanation about how the current state of the Relational Schema/Logical Model looks after each step of the mapping process.

To derived the EER diagram these 9 steps were followed

1. Determine strong entities

   First, strong entities, that can exist on their own were identified. These are: Accounts and TOP10. The relational schema includes these entities with the same attributes that they had in the EER. If they did not have a natural primary key (somehow none of them, not even Account, for which email should be a PK, so I decided that this is going to be the PK), surrogate keys (SK) were created for them. Media has inheritance, so it is handled later.

2. Determine weak entities
   Weak entities cannot exist without a strong entity. These are: User, ToWatchList, Season, Episodes. They are handled similarly to strong entities, including all the attributes form the EER. Foreign keys will be added to them in a later step.
3. 1 to many relationships
   The 1 side becomes the parent, and its PK is referenced in the child as a foreign key (FK). These are the following relationships, with the parent first: Account-User, Season-Episodes, TvShow-Season, Top10-Movie. Since inheritance is not handled yet, TvShow-Season and Top10-Movie must be done later. If the parent side is 1, and the child's primary key is only partial, they would make up a composite primary key together with the parent's PK.
4. 1 to 1 relationships
   The User-ToWatchList is 1 to 0..1, so it is mandatory on the parent (User) side, but optional on the child (ToWatchList) side. The PK of the parent is added to the child.
5. Recursive relationships
   There are no recursive relationships in this database. They would be handled by adding a reference to the same kind of entity within it.
6. Inheritance

It is handled differently based on its type, in this case it is {mandatory, or}. In this case, 3 entities will be created for the subtypes, and all the attributes from the superclass will be moved to each subclass. So the Media entity will not exist, and the Movie, Documentary and TvShow will have its attributes. For each of these entities, surrogate keys were made. Furthermore, the TvShow-Season relationship now can be created by adding a reference to the TvShow's PK in the Season. The Top10-Movie relationship is handled in the same way. Movie's multivalued attribute is handled later.

7. Many to many relationships
   There is one many to many relationship between ToWatchList and Media. Since the Media entity does not exist, it is not possible to reference it. Usually, a join-table is created for these kind of relationships, including PKs from both entities, and any additional relationship attributes. But since one of the PKs would have to reference a 3 different entities, it cannot be done this way. I have created three attributes for the three foreign keys, and in each entry two of them is going to be null. This makes the DDL more complicated because it has to be checked, that one out of three employees is not null, and the other two is null, but the GR diagram is simpler.
8. Complex relationships
   There are no complex relationships in this database, but they would be handled similarly to many to many relationships.
9. Multivalued attributes
   A new entity has to be created for all the attributes, referencing the owner. The Movie entity has many genres, so a Genre entity is created, and it has a FK to Movie. It has a composite key, both attributes are combined.

Now the relational schema can be converted to a GR Diagram.

Account (email, password, subscription_type) PK: email	Top10 (country) PK: country
User (username, is_child_friendly, account_email) PK: username, account_email FK: account_email **ref** Account(email)	ToWatchList (to_watch_list_id, username, account_email) PK: to_watch_list_id (SK) FK: username, account_email **ref** User(username, account_email)
Season (season_id, season_number, release_year, tvshow_id) PK: season_id (SK) FK: tvshow_id **ref** TvShow(tvshow_id)	Episodes (episode_id, episode_number, title, run_time, season_id) PK: episode_id (SK), season_id FK: season_id **ref** Season(season_id)
Movie (movie_id, title, release_year, description, run_time, Top10_country) PK: movie_id (SK) FK: Top10_country **ref** Top10(country)	Documentary(documentary_id, title, release_year, description, run_time, topic) PK: documentary_id (SK)
TvShow (tvshow_id, title, release_year, description) PK: tvshow_id (SK)	MediaInToWatchList (to_watch_list_id, media_id, movie_id, documentarty_id, tvshow_id) PK: to_watch_list_id, media_id (SK) FK: movie_id **ref** Movie(movie_id) FK: documentary_id **ref** Documentary(documentary_id) FK: tvshow_id **ref** TvShow(tvshow_id)
Genre (name, movie_id) PK: name, movie_id FK: movie_id **ref** Movie(movie_id)	

Exercise 3 – Diagram to DDL (15%)

Based on the GR the DDL was written in SQL. Two rows of data were inserted into each table.