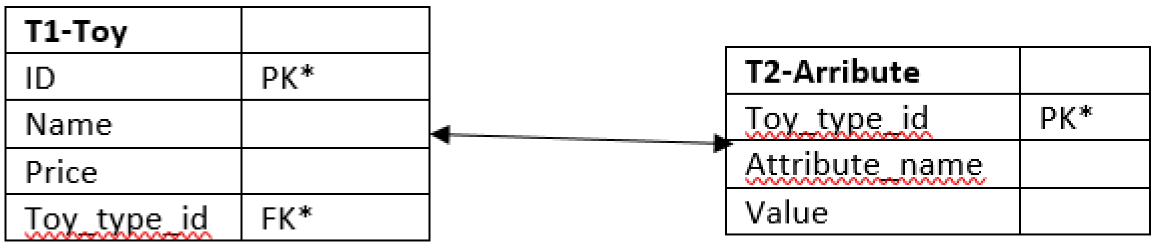
Databases Lab1

Exercise 1:

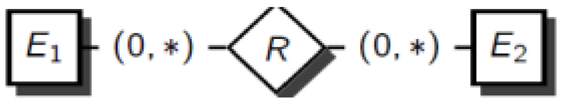


T1-Toy	
ID	PK*
Name	
Price	
Toy_type_id	FK*

T2-Arribute	
Toy_type_id	PK*
Attribute_name	
Value	

I would store the information in this database like the tables shown above. The main table T1-Toy uses ID as its primary key. This allows it to see the identify instances of each toy uniquely. T1 uses ID, Name, Price and Toy_type_id to describe the toy. The Toy_type_id attribute is a foreign key. This relates to the attribute table were the attribute can be assigned and given a value. The primary key on this table is also the Toy_type_id to connect both tables and identify that specific toy. T1 holds the child entity while the other table is a parent entitie. The Toy_type_id is used to connect the specific attributes of the toys (material, age) to the common attributes like name and price. If a car was Toy_type 1 we could assign the Attributes to Engine_size or Petrol_or_Diesel and then give them the value. Separating the common and specific attributes allows for normalisation by getting rid of redundant data.

Exercise 2:

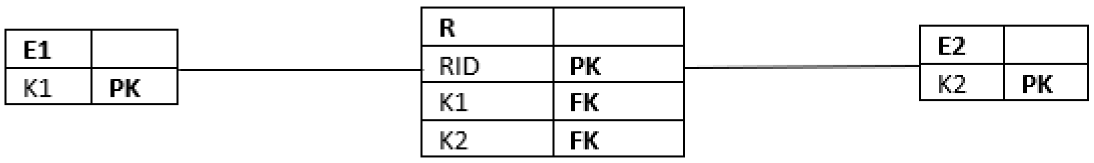


```
CREATE TABLE E1 (
   K1 INT PRIMARY KEY
);

CREATE TABLE E2 (
   K2 INT PRIMARY KEY
);

CREATE TABLE R (
   RID INT PRIMARY KEY,
   K1 INT,
   K2 INT,
   FOREIGN KEY (K1) REFERENCES E1 (K1),
   FOREIGN KEY (K2) REFERENCES E1 (K2)
);
```

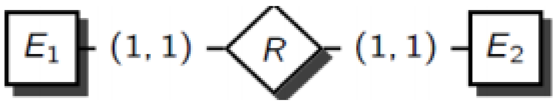
Many to many relationships with optional data entry.



E1	
K1	PK

R	
RID	PK
K1	FK
K2	FK

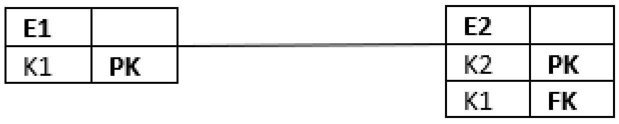
E2	
K2	PK

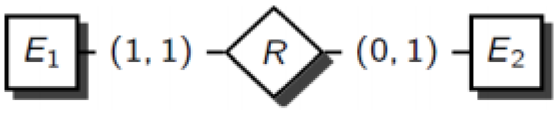


```
CREATE TABLE E1 (
   K1 PRIMARY KEY
);
```

```
CREATE TABLE E2 (
   K2 INT PRIMARY KEY,
   K1 INT,
   FOREIGN KEY (k1) REFERENCES E1 (k1)
);
```

One to one relationship with a mandatory entry for data.

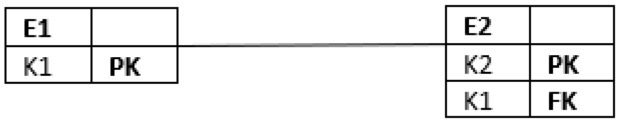


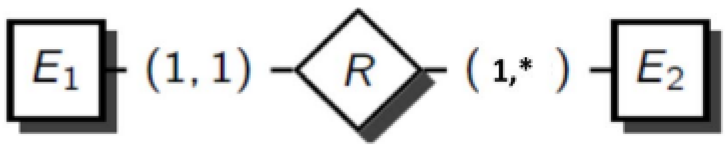


```
CREATE TABLE E1 (
   K1 INT PRIMARY KEY
);
```

```
CREATE TABLE E2 (
   K2 INT PRIMARY KEY,
   K1 INT,
   FOREIGN KEY (K1) REFERENCES E1 (K1)
);
```

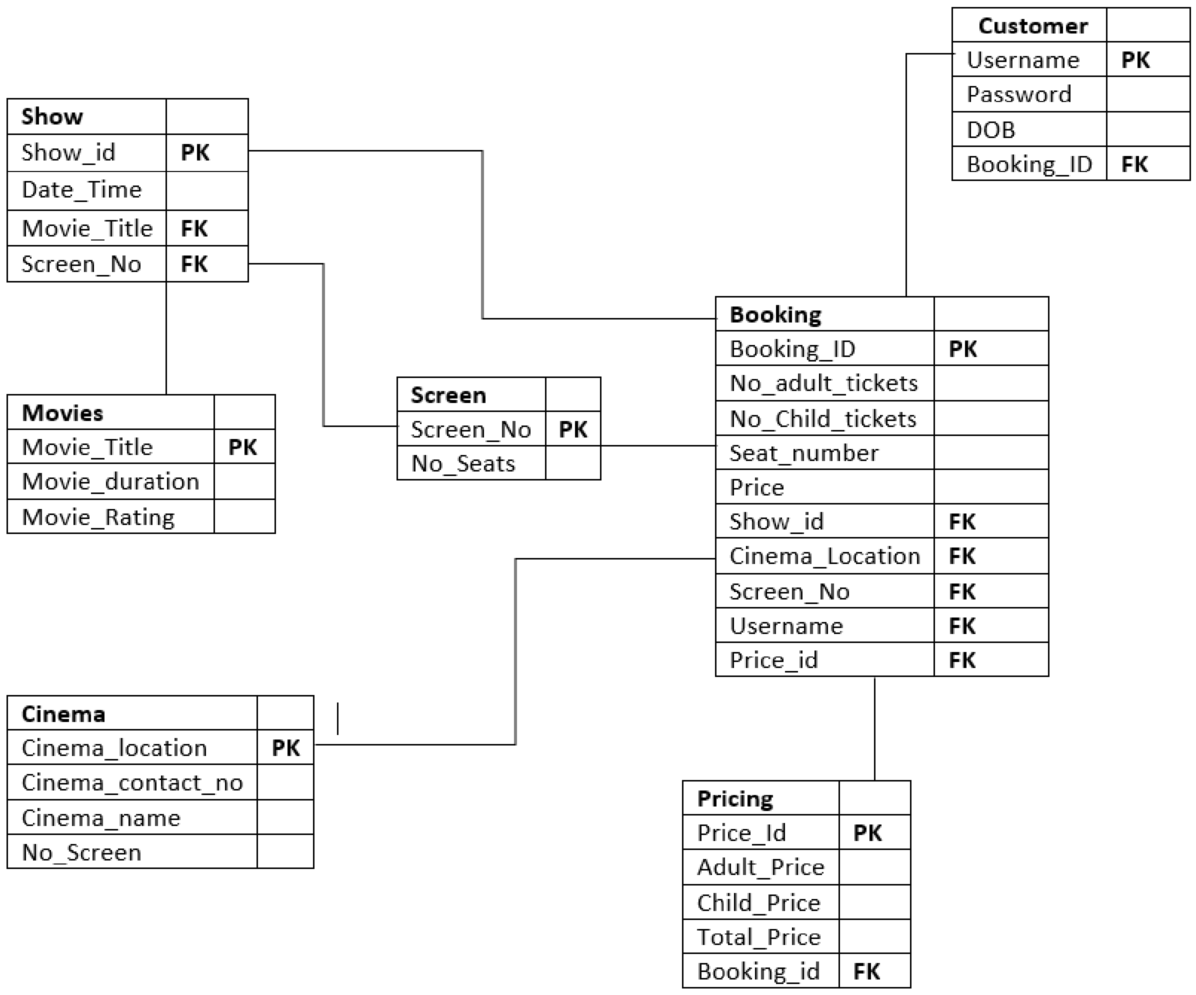
One to one relationship with optional data entry





This is not possible as to insert something into E1, E2 must also have data with in it. This means it would be impossible to create either table. The foreign keys cannot enforce the minimum cardinality in this specific instance.

Exercise 3:



Cinema and Booking have a one to many relationships.
Screen and Booking have a zero to many relationship and booking has one instance of the Screen.
Screen and show one to many relationships.
Show and Booking have a one to zero or many relationship and booking has one instance of Show.
Pricing and Booking have a zero to many relationship and booking has one instance of the Pricing.
Customer and Booking have a zero to many relationship and booking has one instance of the Customer.
Show and Movie have a many to many relationship.