Homework 2

Group 14: Sara Rydell & Filip Northman

Task (P)

Task 1

a. **We got the following feedback:**
   - To check if some of our database relationships were really needed to be presented as schemas, or if that data maybe could be present in other relations/tables, specifically concerning the relations "borrows", "lends", "makes", "pays" and "creates".
   - To evaluate the relation "makes" since it might cause troubles.
   - To change so that Users, Students and Admins should have foreign keys and remember that ISA subclasses in the ER do not exclude weak entities.
   - To explain the differences between a relational database and its ER-diagram in HW1.

   We made the following modifications:
   - We removed the schemas for the relations "borrows", "lends", "pays" and "creates" since those relationships didn't add anything but redundancy to the database if we were to add them as tables. Mainly since they don't have any unique attributes connected to them.
   - We removed the relationship "makes" since having that branch connected in somewhat of a loop in the database will only complicate the relationships and how to later find information in our database.
   - We specify now that UserID from table Users becomes a foreign key in the tables Students and Admins. We name them StudentID and AdminID as foreign keys respectively and reference them to the primary key User_ID.
   - In Q1 & Q2 we should have talked about the schemas/tables and not the entities, since the schemas are used to only identify which tables containing which primary keys, foreign keys, and attributes, should be used in a database. From the schemas we can then visualize this solution in an ER-diagram where each schema/table is visualized as one entity with the attributes and keys connected to them. Here the entities will also be connected to one another by relationships that can have different cardinalities as well as having attributes connected to them, if this relationship has cardinality many-to-many it is a junction table and should also be represented with a table in the database.

b. We want to make the following adjustments to our own accord:
   - We have added the attribute Transaction_Number to the relation Transactions, since we learnt from the detailed requirements that someone could in theory pay a fine in one or multiple transactions since we have the payment option Klarna.

- We also decided to change the domain type of the attribute “damage” in the relation Book from boolean to integer, since documenting the state of the book as a scale of 1-5 for instance, is a more logical solution in the end.
- We now clarify that the relation Fines will consist of the two composite keys Borrowing_ID and Amount since the fine will be connected to each loan and have a specified price.
- We also choose to have changed some of the names for the attributes so that they would work better with the format that will be in the database, for instance adding underscores between words in the names since it won’t be case sensitive and changing attributes like Date of payment into shorter names like Payment_Date.
- We chose that every loan can only be for one book since this solution will be more logical for calculating fines and updating Return_Date for each book respectively. This also means that we will not need a schema for this.

Task 2

We will examine:
Book(Physical ID:integer, Title:string, ISBN:integer, Author:string, Edition:integer, Genre:string, Language:string, Publisher:string, Publication_Date:date, Prequel:string, Damage:integer)

True Relation:
1. It is a true relation since it has a valid primary key, no attributes have the same name and each attribute has only one domain.

1NF:
2.1 We split the original schema into four smaller ones Physical_Book, Book, Author and Genre to get rid of the multivalued attributes. All of the new schemas beneath are now in 1NF.

- **Physical_Book**(Physical ID:integer, Damage:integer)
- **Book**(ISBN:integer, Title:string, Edition: integer, Language:string, Publisher:string, Publication_Date:date, Prequel_ISBN: integer)
- **Author**(Author_ID, Author_Name)
- **Genre**(ISBN:integer, Genre:string)

This is a solution since the same ISBN for multiple physical books means the same everything except Physical_ID, which must be different, and Damage, which may be different, given by the detailed requirements.
We now continue the normalization with all these schemas respectively.

2NF: We will now test **Physical_Book**(Physical ID:integer, Damage:integer)
3.1 Candidate keys: {Physical_ID}
3.2 Prime attributes: Physical_ID
3.3 Non prime attributes: Damage
3.4 Since there are non prime attributes, we continue with the steps for 2NF.

3.5 Since none of the non prime attributes will have the same value for each tuple, we continue with the 2NF steps.
3.6 Possible combinations of prime attributes: {Physical_ID}
3.7 Possible combinations after removing super keys: None
3.8 Possible combinations after removing subsets: None
3.9 Since there are no combinations left, the relation is in 2NF.

2NF: We will now test **Book**(ISBN:integer, Title:string, Edition: integer, Language:string, Publisher:string, Publication_Date:date, Prequel_ISBN: integer)
3.1 Candidate keys: {ISBN}
3.2 Prime attributes: ISBN
3.3 Non prime attributes: Title, Edition, Language, Publisher, Publication_Date, Prequel_ISBN
3.4 Since there are non prime attributes, we continue with the steps for 2NF.
3.5 Since none of the non prime attributes will have the same value for each tuple, we continue with the 2NF steps.
3.6 Possible combinations of prime attributes: {ISBN}
3.7 Possible combinations after removing super keys: None
3.8 Possible combinations after removing subsets: None
3.9 Since there are no combinations left, the relation is in 2NF.

2NF: We will now test **Author**(Author_ID, Author_Name)
3.1 Candidate keys: {Author_ID}
3.2 Prime attributes: Authorl_ID
3.3 Non prime attributes: Author_Name
3.4 Since there are non prime attributes, we continue with the steps for 2NF.
3.5 Since none of the non prime attributes will have the same value for each tuple, we continue with the 2NF steps.
3.6 Possible combinations of prime attributes: {Author_ID}
3.7 Possible combinations after removing super keys: None
3.8 Possible combinations after removing subsets: None
3.9 Since there are no combinations left, the relation is in 2NF.

2NF: We will now test **Book_Genre**(ISBN:integer, Genre:string)
3.1 Candidate keys: {ISBN, Genre}
3.2 Prime attributes: ISBN, Genre
3.3 Non prime attributes: None
3.4 Since there are no non prime attributes the schema is in 2NF.

3NF: We will now examine **Physical_Book**(Physical ID:integer, Damage:integer)
4.1 Candidate keys: {Physical_ID}
4.2 Prime attributes: Physical_ID
4.3 Non prime attributes: Damage
4.4 There are not more than 1 non prime attribute so the relation is in 3NF

3NF: We will now examine **Book**(ISBN:integer, Title:string, Edition: integer, Language:string, Publisher:string, Publication_Date:date, Prequel_ISBN: integer)
4.1 Candidate keys: {ISBN}
4.2 Prime attributes: ISBN
4.3 Non prime attributes: Title, Edition, Language, Publisher, Publication_Date, Prequel
4.4 There are more than 1 non prime attributes so we continue with the steps for 3NF.
4.5 N = {Title, Edition, Language, Publisher, Publication_Date, Prequel}
4.6 Possible combinations containing N: {Title, Edition, Language, Publisher, Publication_Date, Prequel}, {ISBN, Title, Edition, Language, Publisher, Publication_Date, Prequel}
4.7 Combinations after removing super keys: {Title, Edition, Language, Publisher, Publication_Date, Prequel}
4.8 Possible combinations after removing subsets: {Title, Edition, Language, Publisher, Publication_Date, Prequel}
4.9 Since there are combinations left, we continue with the 3NF steps.
4.10

- Title can't be derived from {Edition, Language, Publisher, Publication_Date, Prequel}
- Edition can't be derived from {Title, Language, Publisher, Publication_Date, Prequel}
- Language can't be derived from {Title, Edition, Publisher, Publication_Date, Prequel}
- Publisher can't be derived from {Title, Edition, Language, Publication_Date, Prequel}
- Publication_Date can't be derived from {Title, Edition, Language, Publisher, Prequel}
- Prequel can't be derived from {Title, Edition, Language, Publisher, Publication_Date}

4.11 Since there is no violation the schema is in 3NF.

3NF: We will now examine **Author**(Author_ID, Author_Name)
4.1 Candidate keys: {Author_ID}
4.2 Prime attributes: Author_ID
4.3 Non prime attributes: Author_Name
4.4 There are not more than 1 non prime attribute so the relation is in 3NF.

3NF: We will now examine **Book_Genre**(ISBN:integer, Genre:string)
4.1 Candidate keys: {ISBN, Genre}
4.2 Prime attributes: ISBN, Genre
4.3 Non prime attributes: None
4.4 There are not more than 1 non prime attribute so the schema is in 3NF.

BCNF: We will now examine **Physical_Book**(Physical ID:integer, Damage:integer)
5.1 Candidate keys: {Physical_ID}
5.2 Prime attributes: Physical_ID
5.3 Non prime attributes: Damage
5.4 N = {Damage}
5.5 Possible combinations containing N: {Damage}, {Physical_ID, Damage}
5.6 Combinations after removing super keys: {Damage}
5.7 Since there are combinations left we continue.
5.8 Physical_ID can't be derived from {Damage}.

5.9 Since no violation is found the schema is BCNF and we are left with the relation:
Physical_Book(Physical ID:integer, Damage:integer)

BCNF: We will now examine **Book**(ISBN:integer, Title:string, Edition: integer, Language:string, Publisher:string, Publication_Date:date, Prequel_ISBN: integer)
5.1 Candidate keys: {ISBN}
5.2 Prime attributes: ISBN
5.3 Non prime attributes: Title, Edition, Language, Publisher, Publication_Date, Prequel
5.4 N = {Title, Edition, Language, Publisher, Publication_Date, Prequel}
5.5 Possible combinations containing N: {Title, Edition, Language, Publisher, Publication_Date, Prequel} and {ISBN, Title, Edition, Language, Publisher, Publication_Date, Prequel}
5.6 Combinations after removing super keys: {Title, Edition, Language, Publisher, Publication_Date, Prequel}
5.7 Since there are combinations left we continue.
5.8 ISBN can't be derived from {Title, Edition, Language, Publisher, Publication_Date, Prequel}
5.9 Since no violation is found the schema is in BCNF and we are left with the relation:
Book(ISBN:integer, Title:string, Edition: integer, Language:string, Publisher:string, Publication_Date:date, Prequel_ISBN: integer)

BCNF: We will now examine **Author**(Author_ID, Author_Name)
5.1 Candidate keys: {Author_ID}
5.2 Prime attributes: Author_ID
5.3 Non prime attributes: Author_Name
5.4 N = {Author_Name}
5.5 Possible combinations containing N: {Author_Name} and {Author_ID, Author_Name}
5.6 Combinations after removing super keys: {Author_Name}
5.7 Since there are combinations left we continue.
5.8 Author_ID can't be derived from {Author_Name}.
5.9 Since no violation is found the schema is BCNF and we are left with the relation:
Author(Author_ID:integer, Author_Name:string)

BCNF: We will now examine **Book_Genre**(ISBN:integer, Genre:string)
5.1 Candidate keys: {ISBN, Genre}
5.2 Prime attributes: ISBN, Genre
5.3 Non prime attributes: None
5.4 N = {}
5.5 Possible combinations containing N: {ISBN, Genre}
5.6 Combinations after removing super keys: None
5.7 Since there are no combinations left the schema is in BCNF and we are left with the relation:
Book_Genre(ISBN:integer, Genre:string)

We will examine:
User(User_ID:integer, Full_Name:string, Email:string, Address:string)

True relation:
1. The schema is a true relation since 1.1) it has a valid primary key, 1.2) no two attributes have the same name, and 1.3) each attribute has exactly one domain.

1NF:
2.1 There are no multivalued attributes so the schema is 1NF and we move on to 2NF.

2NF:
3.1 Candidate keys: {User_ID}, {Email}. Email being an option since no user can have the same email in this system based on the detailed requirements.
3.2 Prime attributes: User_ID, Email
3.3 Non prime attributes: Full_Name, Address
3.4 Since there are non prime attributes we continue.
3.5 No attribute can be derived from nothing so we continue.
3.6 Combinations of prime attributes: {User_ID}, {Email}, {User_ID, Email}
3.7 After removing all super keys we have: {}
3.8 We have none left.
3.9 There are no combinations left so the relation is 2NF and we move on to 3NF.

3NF:
4.1 Candidate keys: {User_ID}, {Email}
4.2 Prime attributes: User_ID, Email
4.3 Non prime attributes: Full_Name, Address
4.4 There are 2 non prime attributes so we continue.
4.5 The set of all non prime attributes: N={Full_Name, Address}
4.6 Combinations of N adding prime attributes: {Full_Name, Address}, {Full_Name, Address, User_ID}, {Full_Name, Address, Email}, {Full_Name, Address, User_ID, Email}
4.7 After removing the super keys we have the following: {Full_Name, Address}
4.8 We are left with: {Full_Name, Address}
4.9 There is one combination left so we continue.
4.10 {Full_Name, Address}
4.10.1 We remove Address from the set.
4.10.2 Address can't with certainty be derived from Full_Name since there might be people with the same names living at different addresses.
4.10.3 We repeat although with Full_Name.
4.10.1 We remove Full_name from the set.
4.10.2 Full_Name can't with certainty be derived from Address since there might be multiple people living at the same addresses although having different names.
4.11 We didn't find a violation so the relation is in 3NF and we move on to BCNF.

BCNF:
5.1 Candidate keys: {User_ID}, {Email}
5.2 Prime attributes: User_ID, Email
5.3 Non prime attributes: Full_Name, Address
5.4 The set of all non prime attributes: N={Full_Name, Address}
5.5 Combinations of N adding prime attributes: {Full_Name, Address}, {Full_Name, Address, User_ID}, {Full_Name, Address, Email}, {Full_Name, Address, User_ID, Email}
5.6 After removing the super keys we have the following: {Full_Name, Address}
5.7 Since there are combinations left we continue.
5.8 The remaining combination can't with certainty derive any prime attribute not included in the combination.
5.9 We didn't find any violations so the relation is in BCNF and we are left with the relation:
User(User ID:integer, Full_Name:string, Email:string, Address:string)

We will examine:
Student(Student ID:integer, Programme:string)

True relation:
1. The schema is a true relation since 1.1) it has a valid primary key, 1.2) no two attributes have the same name, and 1.3) each attribute has exactly one domain.

1NF:
2.1 There are no multivalued attributes so the schema is in 1NF and we move on to 2NF.

2NF:
3.1 Candidate keys: {Student_ID}
3.2 Prime attributes: Student_ID
3.3 Non prime attributes: Programme
3.4 There are non prime attributes so we continue.
3.5 No non prime attribute will have the same value for each tuple so we continue.
3.6 Combinations of prime attributes: {Student_ID}
3.7 We are left with: {}
3.8 We are left with: {}
3.9 Since there are no combinations left after 3.8, the relation is in 2NF, we move on to 3NF.

3NF:
4.1 Candidate keys: {Student_ID}
4.2 Prime attributes: Student_ID
4.3 Non prime attributes: Programme
4.4 There are less than 2 non prime attributes so the relation is in 3NF, we move on to BCNF.

BCNF:
5.1 Candidate keys: {Student_ID}
5.2 Prime attributes: Student_ID

5.3 Non prime attributes: Programme
5.4 The set of all non prime attributes: N={Programme}
5.5 Combinations of N adding prime attributes: {Programme}, {Programme, Student_ID}
5.6 We are left with: {Programme}
5.7 There are combinations left so we continue.
5.8 The combination can't derive any prime attribute not included in the combination so the relation is in BCNF and we are left with the relation:
Student(Student_ID:integer, Programme:string)

We will examine:
Admin(Admin ID:integer, Department:string, Phone_Number:integer)

True relation:
1. The schema is a true relation since 1.1) it has a valid primary key, 1.2) no two attributes have the same name, and 1.3) each attribute has exactly one domain.

1NF:
2.1 There are no multivalued attributes so the schema is in 1NF and we move on to 2NF.

2NF
3.1 Candidate keys: {Admin_ID}, {Phone_Number} since we know from the detailed requirements that "Multiple admins can't have the same phone number".
3.2 Prime attributes: Admin_ID, Phone_Number
3.3 Non prime attributes: Department
3.4 There are non prime attributes so we continue.
3.5 No non prime attribute will have the same value for each tuple so we continue.
3.6 Combinations of prime attributes: {Admin_ID}, {Phone_Number}, {Admin_ID, Phone_Number}
3.7 We are left with: {}
3.8 We are left with: {}
3.9 There are no combinations left after 3.8 so the relation is in 2NF and we move on to 3NF.

3NF:
4.1 Candidate keys: {Admin_ID}, {Phone_Number}
4.2 Prime attributes: Admin_ID, Phone_Number
4.3 Non prime attributes: Department
4.4 There are less than 2 non prime attributes so the schema is in 3NF, we move on to BCNF.

BCNF:
5.1 Candidate keys: {Admin_ID}, {Phone_Number}
5.2 Prime attributes: Admin_ID, Phone_Number
5.3 Non prime attributes: Department
5.4 The set of all non prime attributes: N={Department}

5.5 Combinations of N adding prime attributes: {Department}, {Department, Admin_ID}, {Department, Phone_Number}, {Department, Admin_ID, Phone_Number}
5.6 We are left with: {Department}
5.7 There are combinations left after 5.7 so we continue.
5.8 The combination can't derive any prime attribute not included in the combination so the relation is in BCNF and we are left with the relation:
Admin(Admin_ID:integer, Department:string, Phone_Number:integer)

We will examine:
Loan(Borrowing ID:integer, Physical ID:integer, User ID:integer, Borrowing_date:date, Due_Date:date, Return_Date:date)

True relation:
1.1 It is a true relation since it has a valid primary key, no attributes have the same name and each attribute has only one domain.

1NF:
2.1 There are no multivalued attributes since each book lended will be registered as one loan, so the schema is 1NF and we move on to 2NF.

2NF:
3.1 Candidate keys: {Borrowing_ID}. A person could for instance Borrow and return the same book during the same day and then only Borrowing_ID would work as a Candidate key.
3.2 Prime attributes: Borrowing_ID
3.3 Non prime attributes: Physical_ID, User_ID, Borrowing_Date, Due_Date, Return_Date
3.4 Since there are non prime attributes we continue.
3.5 We remove no non prime attributes since none of them have the same value for each tuple.
3.6 This is fulfilled.
3.7 After removing the super keyes we are left with: {}
3.8 We are left with {}
3.9 Since there are no combinations left the relation is 2NF we move on to 3NF.

3NF:
4.1 Candidate keys: {Borrowing_ID}
4.2 Prime attributes: Borrowing_ID
4.3 Non prime attributes: Physical_ID, User_ID, Borrowing_Date, Due_Date, Return_Date
4.4 There are more than 2 non prime attributes so we continue.
4.5 The set of all non prime attributes: N={Physical_ID, User_ID, Borrowing_Date, Due_Date, Return_Date}
4.6 Combinations of N adding prime attributes: {Physical_ID, User_ID, Borrowing_Date, Due_Date, Return_Date}, {Physical_ID, User_ID, Borrowing_Date, Due_Date, Return_Date, Borrowing_ID}
4.7 We are now left with: {Physical_ID, User_ID, Borrowing_Date, Due_Date, Return_Date}

4.8 None are removed.
4.9 There are combinations left so we continue.
4.10.1 Due_Date has been removed: {Physical_ID, User_ID, Borrowing_Date, Return_Date}
4.10.2 Due_date can be derived from the combination since we can calculate the Due_Date from knowing which day the loan was made and for how long someone can borrow a book, 7 days or less according to the detailed requirements and it can be changed at any time. To fix this violation we changed this attribute and get the following combination by changing Due_Date into Max_Time which will say for how long the book can be borrowed instead of the date it should be returned:
{Physical_ID:integer, User_ID:integer, Borrowing_Date:date, Max_Time:integer, Return_Date:date} and we start over from 4.1.
4.1 Candidate keys: {Borrowing_ID}
4.2 Prime attributes: Borrowing_ID
4.3 Non prime attributes: Physical_ID, User_ID, Borrowing_Date, Max_Time, Return_Date
4.4 There are more than 2 non prime attributes so we continue.
4.5 The set of all non prime attributes: N={Physical_ID, User_ID, Borrowing_Date, Max_Time, Return_Date}
4.6 Combinations of N adding prime attributes: {Physical_ID, User_ID, Borrowing_Date, Max_Time, Return_Date}, {Physical_ID, User_ID, Borrowing_Date, Max_Time, Return_Date, Borrowing_ID}
4.7 We are now left with: {Physical_ID, User_ID, Borrowing_Date, Max_Time, Return_Date}
4.8 None are removed.
4.9 There are combinations left so we continue.
4.10.1 Physical_ID has been removed: {User_ID, Borrowing_Date, Max_Time, Return_Date}
4.10.2 Physical_ID can’t be derived from the combination since the user can make multiple loans a day so we continue.
4.10.3 We repeat.
4.10.1 User_ID has been removed: {Physical_ID, Borrowing_Date, Max_Time, Return_Date}
4.10.2 User_ID can’t be derived from the combination since multiple users can borrow the same book in one day so we continue.
4.10.3 We repeat.
4.10.1 Borrowing_Date has been removed: {Physical_ID, User_ID, Max_Time, Return_Date}
4.10.2 Borrowing_Date can’t be derived from the combination so we continue.
4.10.3 We repeat.
4.10.1 Max_Time has been removed: {Physical_ID, User_ID, Borrowing_Date, Return_Date}
4.10.2 Max_Time can’t be derived from the combination so we continue.
4.10.3 We repeat.
4.10.1 Return_Date has been removed: {Physical_ID, User_ID, Borrowing_Date, Max_Time}
4.10.2 Return_Date can’t be derived from the combination so we continue.
4.10.3 We do not have to repeat.
4.11 The relation is in 3NF so we move on to BCNF.

BCNF:
5.1 Candidate keys: {Borrowing_ID}
5.2 Prime attributes: Borrowing_ID
5.3 Non prime attributes: Physical_ID, User_ID, Borrowing_Date, Max_Time, Return_Date
5.4 The set of all non prime attributes: N={Physical_ID, User_ID, Borrowing_Date, Max_Time, Return_Date}
5.5 Combinations of N adding prime attributes: {Physical_ID, User_ID, Borrowing_Date, Max_Time, Return_Date}, {Physical_ID, User_ID, Borrowing_Date, Max_Time, Return_Date, Borrowing_ID}
5.6 We are now left with: {Physical_ID, User_ID, Borrowing_Date, Max_Time, Return_Date}
5.7 There are combinations left so we continue.
5.8 The prime attribute Borrowing_ID can not be derived from this combination.
5.9 We didn't find any violations so the relation is in BCNF and we are left with the relation:
Loan(Borrowing_ID:integer, Physical_ID:integer, User_ID:integer, Borrowing_Date:date, Max_Time:integer, Return_Date:date)

We will examine:
Fine(Borrowing ID:integer, Amount:integer)

True relation:
1. It is a true relation since it has only unique rows, no attributes have the same name and each attribute has only one domain.

1NF:
2.1 There are no multivalued attributes so the schema is in 1NF and we move on to 2NF.

2NF:
3.1 Candidate keys: {Borrowing_ID, Amount}
3.2 Prime attributes: Borrowing_ID, Amount
3.3 Non prime attributes: {}
3.4 There are no non prime attributes so the schema is in 2NF and we move on to 3NF.

3NF:
4.1 Candidate keys: {Borrowing_ID, Amount}
4.2 Prime attributes: Borrowing_ID, Amount
4.3 Non prime attributes: {}
4.4 There are less than 2 non prime attributes so the schema is in 3NF and we stop here.
The fact that the relation is in BCNF follows so we are left with the relation:
Fine(Borrowing_ID:integer, Amount:integer)

We will examine:
Transaction(Transaction_ID:integer, Transaction_Number:integer, Borrowing_ID:integer, Amount:integer, Payment_Date:date, Payment_Method:string)

True Relation:
1. The schema is a true relation since it 1.1) has a valid primary key, 1.2) no two attributes have the same name, and 1.3) each attribute has exactly one domain.

1NF:
2.1 Since there are no multivalued attributes the schema is in 1NF.

2NF:
3.1 Candidate keys: {Transaction_ID}
3.2 Prime attributes: Transaction_ID
3.3 Non prime attributes: Transaction_Number, Borrowing_ID, Amount, Payment_Date, Payment_Method
3.4 Since there are non prime attributes we continue with the steps for 2NF.
3.5 Since none of the non prime attributes will have the same value for each tuple, we continue with the 2NF steps.
3.6 Possible combination of prime attributes: {Transaction_ID}
3.7 Possible combinations after removing super keys: None
3.8 Possible combination after removing subsets: None
3.9 Since there are no combinations left the relation is in 2NF.

3NF:
4.1 Candidate keys: {Transaction_ID}
4.2 Prime attributes: Transaction_ID
4.3 Non prime attributes: Transaction_Number, Borrowing_ID, Amount, Payment_Date, Payment_Method
4.4 There are more than 1 non prime attributes so we continue with the steps for 3NF.
4.5 N = {Transaction_Number, Borrowing_ID, Amount, Payment_Date, Payment_Method}
4.6 Possible combinations containing N: {Transaction_Number, Borrowing_ID, Amount, Payment_Date, Payment_Method}, {Transaction_ID, Transaction_Number, Borrowing_ID, Amount, Payment_Date, Payment_Method}
4.7 Combinations after removing super keys: {Transaction_Number, Borrowing_ID, Amount, Payment_Date, Payment_Method}
4.8 Possible combinations after removing subsets: {Transaction_Number, Borrowing_ID, Amount, Payment_Date, Payment_Method}
4.9 Since there are combinations left we continue with the 3NF steps.
4.10

- Transaction_Number can't be derived from {Borrowing_ID, Amount, Payment_Date, Payment_Method}
- Borrowing_ID can't be derived from {Transaction_Number, Amount, Payment_Date, Payment_Method}
- Amount can't be derived from {Transaction_Number, Borrowing_ID, Payment_Date, Payment_Method}

- Payment_Date can't be derived from {Transaction_Number, Borrowing_ID, Amount, Payment_Method}
- Payment_Method can't be derived from {Transaction_Number, Borrowing_ID, Amount, Payment_Date}

4.11 Since there is no violation the schema is in 3NF.

BNCF:

5.1 Candidate keys: {Transaction_ID}
5.2 Prime attributes: Transaction_ID
5.3 Non prime attributes: Transaction_Number, Borrowing_ID, Amount, Payment_Date, Payment_Method
5.4 N = {Transaction_Number, Borrowing_ID, Amount, Payment_Date, Payment_Method}
5.5 Possible combinations containing N:
{Transaction_Number, Borrowing_ID, Amount, Payment_Date, Payment_Method},
{Transaction_ID, Transaction_Number, Borrowing_ID, Amount, Payment_Date, Payment_Method}
5.6 Combinations after removing super keys:
{Transaction_Number, Borrowing_ID, Amount, Payment_Date, Payment_Method}
5.7 Since there are combinations left we continue with the steps.
5.8 Transaction_ID can't be derived from {Transaction_Number, Borrowing_ID, Amount, Payment_Date, Payment_Method}.
5.9 Since no violation is found the schema is in BCNF and we are left with the relation:
Transaction(Transaction_ID:integer, Transaction_Number:integer, Borrowing_ID:integer, Amount:integer, Payment_Date:date, Payment_Method:string)

List normalized schemas:

- Book(ISBN:integer, Title:string, Edition:integer, Language:string, Publisher:string, Publication_Date:date, Prequel_ISBN:integer)
- Book_Genre(ISBN:integer, Genre:string)
- Author(Author_ID:integer, Author_Name:string)
- Book_Writer(ISBN:integer, Author_ID:integer)
- Physical_Book(Physical ID:integer, ISBN:integer, Damage:integer)
- User(User ID:integer, Full_Name:string, Email:string, Address:string)
- Student(Student_ID:integer, Programme:string)
- Admin(Admin_ID:integer, Department:string, Phone_Number:integer)
- Loan(Borrowing_ID:integer, Physical_ID:integer, User_ID:integer, Borrowing_Date:date, Max_Time:integer, Return_Date:date)
- Fine(Borrowing_ID:integer, Amount:integer)
- Transaction(Transaction_ID:integer, Transaction_Number:integer, Borrowing_ID:integer, Amount:integer, Payment_Date:date, Payment_Method:string)

Additional notes:

All schemas in the final list should be in BCNF.

We have checked so that our solution should be in line with the case study's Detailed Requirements.

We have checked so that our solution should be in line with the Important notes:
- A Physical ID, Borrowing ID, User ID or Transaction ID may only refer to one single physical book, borrowing, user or transaction respectively.
- A physical book may have multiple authors, genres and prequels.
- Only the attributes Author, Edition, Genre, Language, Publisher, Publication_Date, Prequel, Damage, Borrowing_Date, Due_Date, Return_Date can have null data instances in the database.

Task (P+)

1. One disadvantage of allowing null values is that NULL values can cause problems when selecting data. This is because when comparing an unknown value to any other value, the result is always unknown and not included in the results. This can lead to miss leading results when analyzing the content of the database. Then you could use the IS NULL or IS NOT NULL operators to check for a NULL value in order to handle this problem although it could be easily overlooked if a value unexpectedly would be NULL.
   Another disadvantage is that the database leaves spare room for adding values to the NULL values in the tuples later on. It even reserves extra bytes should there be a value larger than what is usually stored in the field. The result is that the database might take up more hard drive storage space than if you used regular values.

2. **We have been given the following relation:**
   Teacher(Teacher_ID:integer, School_ID:integer,Classes:string, School_Address:string, School_Zip_Code:integer, School_Country:string)

a) Prime attributes: Teacher_ID, School_ID
   Non prime attributes: Classes, School_Adress, School_Zip_Code, School_Country
   Candidate keys: {Teacher_ID, School_ID}

b) **True relation:**
   The schema is a true relation although it is designed in a way that it will be violating the 4 normal forms which we will explain below.

   1NF:
   Requirements:
   - Be a true relation
   - Have no multivalued attribute

   Teacher is a true relation.
   It violates 1NF by containing the following multivalued attributes:
   School_ID ,Classes, School_Address, School_Zip_Code, School_Country
   Classes reasonably is a multivalued attribute since one teacher can have several classes. The information could theoretically also be saved as one string of all classes a teacher teaches, although that is an inefficient way of storing this information.
   A teacher could also work at multiple schools so School_ID, School_Adress, School_Zip_Code and School_Country can thus all be multivalued. For instance someone could live at the border of two countries like Sweden and Norway and teach classes at two different schools and then all these attributes would likely be in two versions for all attributes at each school respectively.

2NF:
Requirements:
- Bee in 1NF
- In all dependencies X→Y, where Y is a non prime attribute, X cannot be a proper subset of any candidate key.

Teacher isn't in 1NF which is a violation of 2NF. It violates the second requirement since Classes is only dependent on the Teacher_ID, a proper subset of the candidate key, as classes are reasonably connected to the Teacher_ID only and not the School_ID. Another example is that School_ID which is a proper subset to the candidate key can derive the attributes School_Zip_Code, School_Country and School_Adress since School_ID should be able to uniquely identify each school. Thus this shows that the relation violates 2NF.

3NF:
Requirements:
- Be in 2NF
- For all dependencies X→Y, at least one of the following must be true:
    X is a super key
    X contains Y
    Y \ X are all prime

Teacher isn't in 2NF which is a violation of 3NF. It violates the second requirement and isn't in 3NF since School_Country can be derived from the combination of School_Adress and School_Zip_Code as that's the way the system of addresses and zip code works. There is no reasonable world where two schools exist in different countries with the same address and zip code, that would be crazy. In the important notes we also see that the address is unique for each school but zip code is not. Therefore our argument holds true, there can be no 2 schools with the same address and zip code in 2 different countries.

BCNF:
Requirements:
- Be in 3NF
- For all dependencies X→Y, at least one of the following must be true:
    X is a super key {school_id teacherid}
    X contains Y

Teacher isn't in 3NF which is a violation of BCNF. It also violates the second requirement since the combination of attributes {Shcool_Address, School_Zip_Code, School_Country} which isn't a super key can derive the school and thus the prime attribute School_ID which isn't part of the combination, thus the relation violates BCNF. Also the previous argument used in 3NF makes so that BCNF isn't achieved.