Relational Model:

List of Tables

1. STUDENT

Description: A table to store the personal details like Name, Address, Phonenumber, Department, Status(on hold or not) etc. about the student. The primary key is StudentNumber. The foreign key is StudentNumber to link with the LoginInfoStudent(a table which has the records if the student is authorized or not) and ProgramID, to link with the StudentProgramme (a table which has the information about the students enrollment).

SQL:

```
Create table Student(
StudentNumber varchar(10),
FName varchar(20),
LName varchar(20),
PhoneNumber numeric(10),
AlternatePhone numeric(10),
Street varchar(20),
City varchar(20),
State varchar(20),
PostCode numeric(8),
DOB date,
Sex char(1),
Nationality varchar(20),
Department varchar(30),
fine numeric(10),
 ProgramId numeric(5) NOT NULL,
constraint PK_Student Primary Key (StudentNumber),
constraint FK_StudentProgramme
Foreign Key (ProgramId)
References Programme (ProgramId)ON DELETE NO ACTION,
constraint FK_StudentLoginInfoStudent foreign key (StudentNumber) references
logininfostudent (studentnumber));
```

Functional Dependency:

StudentNumber → FName, Lname, PhoneNumber, AlternatePhone, City, Department, DOB, Fine, Nationality, PostCode, ProgramID, Sex, State, Street

Normal Form:

BCNF

2. AUTHOR

Description: A table to store the name of the all the authors, be it a books author, a journals author or a conferences author. The primary key is AuthorId which is assigned to uniquely define the author.

SQL:

```
Create Table Author(
AuthorId numeric(10),
AuthorName varchar(20),
Constraint PK_Author Primary Key(AuthorId)
);
```

Functional Dependency:

AuthorId → AuthorName

Normal Form:

BCNF

3. BOOK

Description: This table is to store every details about the books present in the library and also has attributes which suggests in which library the book is, then what kind of publication is it ie a book, a journal, a conference(here since the table is about the book it will never refer to a journal or a conference), the resource id to uniquely define that book among the other resources which include other books, journals, conferences, then the number of copies present and available for that instance, and finally whether an E-Copy of that book is available or not. Here the ISBN is the primary key and the LibraryId and

CategoryId together are the foreign key to link the table LibraryCategory(A table with the category information associated with the categoryID and the library to which it belongs to) hence LibraryID and CategoryID cannot be NULL. And ResourceID is a foreign key to link with the table Resources(A table where all the resourceID are stored and fetched from).

SQL:

```
Create Table Book (
ISBN varchar(20),
Title varchar(30),
Edition Varchar(20),
Year varchar(10),
Pulisher varchar(30),
LibraryId numeric(10) NOT NULL,
CategoryId numeric(10) NOT NULL,
ResourceId numeric(10) NOT NULL,
NumberOfCopies numeric(10),
AvailableCopies numeric(10),
Ecopy char check (Ecopy in ('0','1')),
Constraint PK_Book Primary Key (ISBN),
Constraint FK_BOOKRESOURCES Foreign Key(ResourceID)
References Resources(ResourceID),
Constraint FK_BookLibraryCategory Foreign Key(LibraryId, CategoryId)
References LibraryCategory(LibraryId, CategoryId) ON DELETE NO ACTION
);
```

Functional Dependency:

ISBN → Title, Edition, Year, Publisher, LibraryID, CategoryID, ResourceID, NumberOfCopies, AvailableCopies, Ecopy

Normal Form:

BCNF

4. BOOKAUTHOR

Description: A table that links the book and the author. Here the primary key is the fusion of ISBN and AUTHORID. The foreign Key is ISBN to link with BOOK(A table with all the details about the books) and the other foreign key is AuthorID to link with Author(A table which lists all the authors).

SQL:

```
Create Table BookAuthor(
ISBN varchar(20) NOT NULL,
AuthorId numeric(10) NOT NULL,
Constraint PK_BookAuthor Primary Key (ISBN, AuthorId),
Constraint FK_BookAuthorBook Foreign Key(ISBN)
References Book(ISBN) ON DELETE CASCADE,
Constraint FK_BookAuthorAuthor Foreign Key(AuthorId)
References Author(AuthorId)ON DELETE CASCADE
);
```

Functional Dependency:

No functional dependency

Normal Form:

Indecisive

5. BOOKRESERVED

Description: A table which stores the information about the books reserved by the faculty. The attributes in the table suggest the name of the faculty, the book which he reserved and the Coursename for which that book is reserved so that only those students who have enrolled for that course can checkout that book.
The table also stores when was the book reserved and till when. The primary key is the fusion of ISBN, FACULTYNUMBER, COURSENAME. The foreign key is ISBN to link it with Book(A table with all the details about the books available in the library). The other foreign key is FacultyNumber to link it with Faculty (A table with all the details about faculty members). The third foreign key is CourseName to link it with Course(The table with the description of all the courses).

SQL:

```
Create Table BookReserved(
ISBN varchar(20) NOT NULL,
FacultyNumber varchar(20) NOT NULL,
CourseName varchar(20) NOT NULL,
ResStartTime Date,
ResEndTime Date,
Constraint PK_BookReserved Primary Key(ISBN, FacultyNumber, CourseName),
Constraint FK_BookReservedBook Foreign Key(ISBN) References Book(ISBN) ON
DELETE CASCADE,
Constraint FK_BookReservedFaculty Foreign Key (FacultyNumber) References
Faculty(FacultyNumber) ON DELETE CASCADE,
Constraint FK_BookReservedCourse Foreign Key(CourseName) References
Courses(CourseName) ON DELETE CASCADE
);
```

Functional Dependency:

ISBN, FACULTYNUMBER, COURSENAME → RESSTARTITME, RESENDTIME

Normal Form:

BCNF

6. CAMERA

Description: A table with all the details about the camera which includes its type, lens type, available memory, the library in which it is present and the category it belongs to (this is basically added to know the type of item which is being handled). The primary key is CameraID. The foreign key is the fusion of LibraryID and CategoryID to link with LibraryCategory(A table with the category information associated with the categoryID and the library to which it belongs to).

SQL:

```
Create Table Camera(
CameraId Numeric(10),
LibraryId numeric(10) NOT NULL,
CategoryId numeric(10) NOT NULL,
Manufacturer varchar(20),
Model varchar(20),
Configuration varchar(20),
MemoryAvailable varchar(20),
Constraint PK_Camera Primary Key (CameraId),
Constraint FK_CameraCategory Foreign Key(LibraryId, CategoryId) References
LibraryCategory(LibraryId, CategoryId) ON DELETE NO ACTION
);
```

Functional Dependency:

CAMERAID → LIBRARYID, CATEGORYID, MANUFACTURER, MODEL, CONFIGURATION, MEMORYAVAILABLE.

Normal Form:

BCNF

7. CAMERACHECKOUT

Description: A table to keep a record of when the cameras were checked out and when were those checked in. A trigger will run and update the CameraCheckoutLog by calculating the fine if any and updating the availability of the camera. The foreign key is ReservationID to link with CameraReservation(A table that documents the reservation of the camera). So the importance of this table is run the above mentioned trigger so that we don’t need to load the attributes which are out of question.

SQL:

```
Create Table CameraCheckout
(
ReservationId Numeric(10),
CheckInTime Date,
CheckOutTime Date,
Constraint FK_CheckoutCameraReservation Foreign Key (ReservationId)
References CameraReservation(ReservationId)ON DELETE NO ACTION
);
```

Functional Dependency:

No functional dependency

Normal Form:

Indecisive

8. CAMERACHECKOUTLOG

Description: A table to keep a log of cameras that were checked out. It also suggests the end time of the reservation, the reservationID, the CheckInTime and CheckoutTime, the student and Faculty number to map which patron is associated with that Reservation and finally if any, the fine to be paid after the CheckIn. The foreign key is StudentNumber and FacultyNumber to link with the Student and Faculty table respectively. Since that needs to be updated into their respective profiles. No primary key.

SQL:

```
Create Table CameraCheckoutLog
(
ReservationId Numeric(10),
CameraId varchar(10),
CheckInTime Timestamp,
CheckOutTime Timestamp,
EndTime Timestamp,
StudentNumber varchar(10),
FacultyNumber varchar(10),
fine numeric(10),
Constraint FK_CameraLogStudent Foreign Key (StudentNumber) References
Student(StudentNumber),
Constraint FK_CameraLogFaculty Foreign Key (FacultyNumber) References
Faculty(FacultyNumber)
);
```

Functional Dependency:

No functional dependency

Normal Form:

Indecisive

9. CAMERAQUEUE

Description: A table to generate the QueueNo on request by a patron. Here since there are limited number of cameras, it may happen that the demand is greater than the supply and hence we can manage this using a queue which follows the first in first out basis. The foreign is CameraID which links to Camera(A table with all the details about the camera). The other two foreign keys are StudentNumber and FacultyNumber which helps to link to Student and Faculty respectively.

SQL:

```
Create Table CameraQueue(
CameraId Varchar2(10),
QueueNo Numeric(10),
StartTime Timestamp,
EndTime Timestamp,
StudentNumber varchar(10),
FacultyNumber varchar(10),
Constraint FK_CameraQueueCamera Foreign Key(CameraId)
References Camera(CameraId),
Constraint FK_CameraQueueStudent Foreign Key (StudentNumber)
References Student(StudentNumber),
Constraint FK_CameraQueueFaculty Foreign Key (FacultyNumber)
References Faculty(FacultyNumber));
```

Functional Dependency:

No functional dependency

Normal Form:

Indecisive

10. CAMERARESERVATION

Description: A table which documents the camera reservation. It has the CameraID, the StartTime and EndTime of the reservation, and the ID of the patron who reserved it. The primary key is ReservationID. The foreign key is StudentNumber and FacultyNumber to link with the Student and Faculty table respectively.

SQL:

```
Create Table Checkout(
ResourceId numeric(10),
CheckoutTime Date,
ReservedUpto Date,
CheckInTime Date,
StudentNumber varchar(10),
FacultyNumber varchar(10),
Constraint FK_CheckoutStudent Foreign Key (StudentNumber)
 References Student(StudentNumber) ON DELETE NO ACTION,
Constraint FK_CheckoutFaculty Foreign Key (FacultyNumber)
 References Faculty(FacultyNumber) ON DELETE NO ACTION,
Constraint  FK_CheckoutResource Foreign Key(ResourceId)
References Resource(ResourceId) ON DELETE NO ACTION
);
```

Functional Dependency:

No functional dependency

Normal Form:

Indecisive

11. CHECKOUT

Description: A table to keep a record of when the books were checked out and when were those checked in. The CHECKOUTLOGTRIGGER will run on this table and calculate the fine and then update the CheckoutLog table with the increment in the availability of the books. The foreign key is ResourceID to link with Resources. The other foreign key is StudentNumber and FacultyNumber which links to the Student and Faculty table respectively. To keep track the patron that checked the book out.

SQL:

```
Create Table Checkout(
ResourceId numeric(10),
CheckoutTime Date,
ReservedUpto Date,
CheckInTime Date,
StudentNumber varchar(10),
FacultyNumber varchar(10),
Constraint FK_CheckoutStudent Foreign Key (StudentNumber) References
Student(StudentNumber) ON DELETE NO ACTION,
Constraint FK_CheckoutFaculty Foreign Key (FacultyNumber) References
Faculty(FacultyNumber) ON DELETE NO ACTION,
Constraint  FK_CheckoutResource Foreign Key(ResourceId) References
Resource(ResourceId) ON DELETE NO ACTION);
```

Functional Dependency:

No functional dependency

Normal Form:

Indecisive

12. CHECKOUTLOG

Description: A table to keep a log of books that were checked out. It also suggests the end time of the reservation, the ResourceID, the CheckInTime and CheckoutTime, the student and Faculty number to map which patron is associated with that Reservation and finally if any, the fine to be paid during the CheckIn. The foreign key is StudentNumber and FacultyNumber to link with the Student and Faculty table respectively. Since that needs to be updated into their respective profiles. The other foreign key is ResourceID to link with the Resources table.

SQL:

```
Create Table CheckOutLog( ResourceId numeric(10),
CheckoutTime Timestamp,
ReservedUpto Timestamp,
CheckInTime Timestamp,
StudentNumber varchar(10),
FacultyNumber varchar(10),
 typepub varchar(20),
namepub varchar(20),
fine numeric(10),
Constraint FK_CheckoutStudent1 Foreign Key (StudentNumber) References
Student(StudentNumber),
Constraint FK_CheckoutFaculty1 Foreign Key (FacultyNumber) References
Faculty(FacultyNumber),
Constraint  FK_CheckoutResource1 Foreign Key(ResourceId) References
Resources(ResourceId)
);
```

Functional Dependency:

No functional dependency

Normal Form:

Indecisive

13. CHECKOUTROOM

Description: A table to keep a record of when the room were checked out and when were those checked in. The ROOMCHECKOUTLOGTRIGGER will run on this table and calculate the fine and then also update the CheckoutRoomLog table by incrementing the availability of the room. So the importance of this table is run the above mentioned trigger so that we don't need to load the attributes which are out of question.

SQL:

```
Create Table CheckoutRoom(
ReservationId Numeric(10),
CheckInTime Timestamp,
CheckOutTime Timestamp,
Constraint FK_CheckoutRoomRoomReservation Foreign Key (ReservationId)
References RoomReservation(ReservationId)
);
```

Functional Dependency:

No functional dependency

Normal Form:

Indecisive

14. CHECKOUTROOMLOG

Description: A table to keep a log of rooms that were checked out. It also suggests the end time of the reservation, the ReservationID, the CheckInTime and CheckoutTime, the student and Faculty number to map which patron is associated with that Reservation and finally if any, the fine to be paid during the CheckIn. The foreign key is StudentNumber and FacultyNumber to link with the Student and Faculty table respectively. Since that needs to be updated into their respective profiles. The other foreign key is ReservationID to link with the RoomReservation table.

SQL:

```
Create Table CheckoutRoomLog(
ReservationId Numeric(10),
RoomId numeric(10),
StartTime Timestamp,
EndTime Timestamp,
CheckInTime Timestamp,
CheckOutTime Timestamp,
StudentNumber varchar(10),
FacultyNumber varchar(10),
Constraint FK_RoomReservationRoom1 foreign Key(RoomId) References
Room(RoomId),
Constraint FK_RoomReservationStudent1 Foreign Key (StudentNumber)
References Student(StudentNumber),
Constraint FK_RoomReservationFaculty1 Foreign Key (FacultyNumber)
References Faculty(FacultyNumber)
);
```

Functional Dependency:

No functional dependency

Normal Form:

Indecisive

15. CHECKOUTWAIT

Description: A table to store the details of the requested items to be checked out. It has the FacultyNumber, StudentNumber, TimeStamp of request, the ResourceID, and few other details of the book. Here the significance of the table is a stored procedure is run on this table to decide who gets the book when it is available once again. Now the priority is highest for the Faculty and then for the students the timestamp of request is checked and assigned the priority. The foreign key is ResourceID which links to Resources. The other foreign keys are StudentNumber and FacultyNumber to link with Students and Faculty respectively.

SQL:
```
Create Table CheckoutWait(
ResourceId numeric(10),
Timestamp Date,
StudentNumber varchar(10),
FacultyNumber varchar(10),
reservedupto timestamp,
typepub varchar(20),
add namepub varchar(20),
Constraint FK_CheckoutWaitStudent Foreign Key (StudentNumber) References
Student(StudentNumber),
Constraint FK_CheckoutWaitFaculty Foreign Key (FacultyNumber) References
Faculty(FacultyNumber),
Constraint  FK_CheckoutWaitResource Foreign Key(ResourceId) References
Resources(ResourceId)
);
```

Functional Dependency:

No functional dependency

Normal Form:

Indecisive

16. CONFERENCE

Description: A table with all the records of the conferences available and present in which library. It has the attributes that suggests the total number of copies of that particular conference, its name, if there is an e-copy available or not, the total number of available copies that can be checked out, and other details about the conference paper. The primary key is ConferenceNumber. The foreign key is LibraryID which links to the Library Category table. The other foreign key is ResourceID which links to the Resources Table.

SQL:

```
Create Table Conference (
 ConferenceNumber varchar(20),
Title varchar(30),
Year varchar(10),
LibraryId numeric(10) NOT NULL,
CategoryId numeric(10) NOT NULL,
ResourceId numeric(10) NOT NULL,
Ecopy char check (Ecopy in ('0','1')),
ConferenceName varchar(20),
NumberOfCopies  numeric(10),
 AvailableCopies  numeric(10),
Constraint PK_Conference Primary Key (ConferenceNumber),
Constraint FK_Conference LibraryCategory Foreign Key(LibraryId, CategoryId)
References LibraryCategory(LibraryId, CategoryId) ON DELETE NO ACTION,
constraint FK_ConferenceResource Foreign Key (ResourceId)
References Resource1 (ResourceId) ON DELETE NO ACTION
);
```

Functional Dependency:

ConferenceNumber → ConferenceName, CategoryID, E-copy, Title, Year, ResourceId, LibraryID, NumberOfCopies, AvailableCopies.

Normal Form:

BCNF

17. CONFERENCEAUTHOR

Description: A table that links the conference and the author. Here the primary key is the fusion of ConferenceNumber and AUTHORID. The foreign Key is ConferenceNumber to link with Conference Table and the other foreign key is AuthorID to link with Author(A table which lists all the authors).

SQL:

```
Create Table ConferenceAuthor
(
ConferenceNumber varchar(20) NOT NULL,
AuthorId numeric(10) NOT NULL,
Constraint PK_ConferenceAuthor Primary Key (ConferenceNumber, AuthorId),
Constraint FK_ConferenceAuthorConference Foreign Key(ConferenceNumber)
References Conference (ConferenceNumber) ON DELETE CASCADE,
Constraint FK_ConferenceAuthorAuthor Foreign Key(AuthorId) References
Author(AuthorId) ON DELETE CASCADE
);
```

Functional Dependency:

No functional dependency

Normal Form:

Indecisive

18. COURSES

Description: A table to give the description for the given course. The courseName is the primary key.

SQL:

```
create table Courses(
CourseName varchar(20),
CourseDescription varchar(30),
Constraint PK_Courses Primary Key (CourseName)
);
```

Functional Dependency:

CourseName→ CourseDescription

Normal Form:

BCNF

19. COURSESTAKEN

Description: A table to store the courses taken by the student. The primary key is the fusion of StudentNumber and CourseName. The foreign key is CourseName to link with Courses table and the other foreign key is StudentNumber to link with Students Table.

SQL:

```
create Table CoursesTaken(
StudentNumber varchar(10),
CourseName varchar(20),
Constraint PK_CoursesTaken Primary Key (StudentNumber, CourseName),
Constraint FK_CoursesTakenStudent Foreign Key (StudentNumber)
References Student(StudentNumber),
Constraint FK_CoursesTakenCourses Foreign Key (CourseName)
 References Courses(CourseName)
);
```

Functional Dependency:

No Functional Dependency

Normal Form:

Indecisive

20. COURSESTEACHES

Description: A table to store which course is taught by which faculty. The primary key is the combination of CourseName and FacultyNumber. The foreign key is CourseName to link with Courses and the other foreign key is FacultyNumber to link with the FacultyTable.

SQL:

```
create Table CoursesTeaches(
FacultyNumber varchar(10),
CourseName varchar(20),
Constraint PK_CoursesTeaches Primary Key (FacultyNumber, CourseName),
Constraint FK_CoursesTeachesStudent Foreign Key (FacultyNumber)
 References Faculty(FacultyNumber),
Constraint FK_CoursesTeachesCourses Foreign Key (CourseName)
References Courses(CourseName)
);
```

Functional Dependency:

No functional dependency

Normal Form:

Indecisive

21. ECOPYCHECKOUT

Description: A table to record the Ecopy checkout by a patron.

SQL:

```
create Table E-CopyCheckout(
FacultyNumber varchar(10),
NameOfPub varchar(10),
StudentNumber varchar(10),
TypeOfPublication varchar(10),
ResourceID number,
CheckOutTime Timestamp,
);
```

Functional Dependency:

No functional dependency

Normal Form: Indecisive

22. EPUBLICATIONCHECKOUT

Description: A table to record the checking out of the an E-publication.

SQL:

```
create Table E-PublishcationCheckout(
FacultyNumber varchar(10),
StudentNumber varchar(10),
ResourceID number,
CheckOutTime Timestamp,
);
```

Functional Dependency:

No functional dependency

Normal Form:

Indecisive

23. FACULTY

Description: A table to store the personal details like Name, Address, Phonenumber, Department, Status(on hold or not) etc. about the faculty. The primary key is FacultyNumber. The foreign key is FacultyNumber to link with the LoginInfoStudent(a table which has the records if the person is authorized or not) and CategoryID, to link with the FacultyCategory table.

SQL:

```
Create Table Faculty(FacultyNumber varchar(10),
FName varchar(20),
LName varchar(20),
CategoryId numeric(10) Not NULL,
Nationality varchar(20),
Department varchar(30),
Hold char(5),
fine numeric(10),
Constraint PK_Faculty Primary Key (FacultyNumber),
constraint FK_FACULTYLoginInfoStudent foreign key (FACULTYNUMBER)
references logininfoFACULTY(FACULTYnumber));
```

Functional Dependency:

FacultyNumber → Fname, LName, CategoryId, Nationality, Department, Hold, Fine

Normal Form:

BCNF

24. FACULTYCATEGORY

Description: A table which stores the designation of the faculty and maps it to the corresponding CategoryID. The primary key is the CategoryID.

SQL:

```
Create Table FacultyCategory(
CategoryId numeric(10),
CategoryDesc varchar(30),
Constraint PK_FacultyCategory Primary Key (CategoryId),
);
```

Functional Dependency:

CategoryID → CategoryDesc

Normal Form:

BCNF

25. JOURNAL

Description: This table is to store every details about the journals present in the library and also has attributes which suggests in which library the journal is, then what kind of publication is it ie a book, a journal, a conference(here since the table is about the journal it will never refer to a book or a conference), the resource id to uniquely define that journal among the other resources which include other journals, books, conferences, then the number of copies present and available for that instance, and finally whether an E-Copy of that journal is available or not. Here the ISSN is the primary key and the LibraryId and CategoryId together are the foreign key to link the table LibraryCategory(A table with the category information associated with the categoryID and the library to which it belongs to) hence LibraryID and CategoryID cannot be NULL. And ResourceID is a foreign key to link with the table Resources(A table where all the resourceID are stored and fetched from).

SQL:

```
Create Table Journal( ISSN varchar(20),
Title varchar(30),
Year varchar(10),
LibraryId numeric(10) NOT NULL,
CategoryId numeric(10) NOT NULL,
ResourceId numeric(10) NOT NULL,
NumberOfCopies numeric(10),
AvailableCopies numeric(10),
Ecopy char check (Ecopy in ('0','1')),
Constraint PK_Journal Primary Key (ISSN),
Constraint FK_JournalLibraryCategory Foreign Key(LibraryId, CategoryId)
References LibraryCategory(LibraryId, CategoryId) ON DELETE NO ACTION

constraint FK_JournalResource Foreign Key (ResourceId)
References Resource1 (ResourceId) ON DELETE NO ACTION;
);
```

Functional Dependency:

ISSN → AvailableCpoies, CategoryID, ECopy, LibraryID, NumberOfCopies, ResourceID, Title, Year

Normal Form:

BCNF

26. JOURNALAUTHOR

Description: A table that links the journal and the author. Here the primary key is the fusion of ISSN and AUTHORID. The foreign Key is ISSN to link with Journal Table and the other foreign key is AuthorID to link with Author(A table which lists all the authors).

SQL:

```
Create Table JournalAuthor
(
ISSN varchar(20) NOT NULL,
AuthorId numeric(10) NOT NULL,
Constraint PK_JournalAuthor Primary Key (ISSN, AuthorId),
Constraint FK_JounalAuthorJournal Foreign Key(ISSN) References Journal(ISSN)
ON DELETE CASCADE,
Constraint FK_JournalAuthorAuthor Foreign Key(AuthorId) References
Author(AuthorId) ON DELETE CASCADE
);
```

Functional Dependency:

No functional dependency

Normal Form:

Indecisive

27. LIBRARY

Description: A table which stores the name of the library and maps it to a unique LibraryId for future references. The primary key is LibraryID.

SQL:

```
Create table Library(
LibraryId numeric(10),
LibraryName varchar(20),
Constraint PK_Library Primary Key (LibraryId)
);
```

Functional Dependency:

LibraryID → LibraryName

Normal Form:

BCNF

28. LIBRARYCATEGORY

Description: A table to map the category of the resource with the library it is present in. The CategoryID and LibraryID is the primary key. The foreign key is LibraryID to link it with the Library table.

SQL:

```
Create table LibraryCategory(
CategoryId numeric(10) NOT NULL,
Category varchar(20),
LibraryId numeric(10) NOT NULL,
Constraint PK_LibraryCategory Primary Key (LibraryId, CategoryId),
Constraint FK_LibraryCategoryLibrary Foreign Key (LibraryId)
 References Library(LibraryId) ON DELETE NO ACTION
);
```

Functional Dependency:

CategoryId → Category

Normal Form:

1 NF

29. LOGININFOFACULTY

Description: A table which will store the Username and Password with the FacultyNumber. This is used for authorized Faculty Login. The primary key is FacultyNumber.

SQL:

```
create table logininfofaculty
(
USERNAME VARCHAR2(20),
PASSWORD VARCHAR2(20),
FACULTYNUMBER VARCHAR2(10),
constraint PK_LoginInfoFACULTY primary key (FACULTYNUMBER)
);
```

Functional Dependency:

FacultyNumber→ Username, Password
Username → Password

Normal Form:

3 NF

30. LOGININFOSTUDENT

Description: A table which will store the Username and Password with the StudentNumber. This is used for authorized Faculty Login. The primary key is StudentNumber.

SQL:

```
create table logininfoStudnet
(
USERNAME VARCHAR2(20),
PASSWORD VARCHAR2(20),
STUDENTNUMBER VARCHAR2(10),
constraint PK_LoginInfoFACULTY primary key (STUDENTNUMBER)
);
```

Functional Dependency:

StudentNumber→ Username, Password
Username → Password

Normal Form:

3 NF

31. PROGRAMME

Description: The table stores the enrollment details of the student. This includes the DegreeProgram, Year, Classifcation of Studying Level etc. The primary key is ProgramID.

SQL:

```
Create Table Programme (
ProgramId numeric(5),
Classification varchar(20),
 DegreeProgram varchar(10),
Year varchar(10),
Constraint PK_Programme Primary Key (ProgramId)
);
```

Functional Dependency:

ProgramID → Classification, DegreeProgram, Year

Normal Form:

BCNF

32. REMINDER

Description: A table to store the reminder associate it with the respective patron. Every time the patron logs in a stored procedure will run and if there is anything to remind the patron about, it will send the respective patron.

SQL:

```
create Table Reminder(
FacultyNumber varchar(10),
ReminderID Number(10),
ReminderMessage varchar(100),
ReminderResourceType varchar(50),
StudentNumber varchar(10),
TimeStamp Timestamp,
Constraint FK_REMINDERFACULTY Foreign Key(FacultyNumber)
 References Faculty(FacultyNumber),
Constraint FK_REMINDERSTUDENT Foreign Key(StudentNumber)
 References Faculty(StudentNumber),
);
```

Functional Dependency:

No functional dependency

Normal Form:

Indecisive

33. RESOURCES

Description: A table to record total number of available resources.

SQL:

```
Create table Resources (
ResourceID Number(10)
);
```

Functional Dependency:

No functional dependency

Normal Form:

Indecisive

34. ROOM

Description: A table to store the details about the room which suggests where the room is, the room number, the capacity of the room, the type of the room etc. The primary key is RoomID. The foreign key is LibraryID to link it with the LibraryCategory table.

SQL:

```
Create Table Room(
RoomId Numeric(10),
RoomNumber numeric(10),
LibraryId numeric(10) NOT NULL,
CategoryId numeric(10) NOT NULL,
FloorNo numeric(10),
Capacity numeric(10),
Type varchar(20),
Constraint PK_Room primary key(RoomId),
Constraint FK_RoomCategory Foreign Key(LibraryId, CategoryId) References
LibraryCategory(LibraryId, CategoryId) ON DELETE NO ACTION
);
```

Functional Dependency:

RoomID → Capacity, CategoryID, FloorNo, LibraryID, RoomNumber, Type

Normal Form:

BCNF

35. ROOMRESERVATION

Description: A table which documents the room reservation. It has the ReservationID, the StartTime and EndTime of the reservation, and the ID of the patron who reserved it. The primary key is ReservationID. The foreign key is StudentNumber and FacultyNumber to link with the Student and Faculty table respectively.

SQL:

```
Create Table RoomReservation(
ReservationId Numeric(10),
RoomId varchar(10),
StartTime timestamp,
EndTime timestamp,
StudentNumber varchar(10),
FacultyNumber varchar(10),
Constraint PK_RoomReservation Primary Key(ReservationId),
Constraint FK_RoomReservationRoom foreign Key(RoomId) References
Room(RoomId)ON DELETE NO ACTION,
Constraint FK_RoomReservationStudent Foreign Key (StudentNumber)
References Student(StudentNumber)ON DELETE NO ACTION,
Constraint FK_RoomReservationFaculty Foreign Key (FacultyNumber) References
Faculty(FacultyNumber)ON DELETE NO ACTION
);
```

Functional Dependency:

ReservationId → FacultyNumber, StudentNumber, RoomID, StartTime, EndTime

Normal Form:

BCNF

36. ROOMRESERVATIONLOG

Description: A table which just stores the details about a reservation made when the checkout has been made. The details it has includes, RoomId, Patron ID, StartTime and EndTime.

SQL:

```
create table roomreservationlog
(
roomid numeric(10),
starttime timestamp,
endtime timestamp,
studentnumber varchar(10),
facultynumber varchar(10)
);
```

Functional Dependency:

No Functional Dependency

Normal Form:

Indecisive

37. ROOMRESERVATIONNULLLOG

Description: A table which just stores the details about a reservation released when the patron is unsuccessful to checkin post 1 hour of the scheduled reservation. The details it has includes, RoomId, Patron ID, StartTime and EndTime.

SQL:

```
create table roomreservationnulllog
(
roomid numeric(10),
starttime timestamp,
endtime timestamp,
studentnumber varchar(10),
facultynumber varchar(10)
);
```

Functional Dependency:
No Functional Dependency

Normal Form:

Indecisive

38. StudentHold:

Description: A table which just stores the details about Student who is on hold and hence has no library access until the fine has been paid.

SQL:

```
create table roomreservationnulllog
(
studentnumber varchar(10),
constraint PK_StudentHold primary key (STUDENTNUMBER)
Constraint FK_StuentHoldStudent Foreign Key (StudentNumber) References
Student(StudentNumber)ON DELETE NO ACTION);
```

Functional Dependency:
No Functional Dependency

Normal Form:

Indecisive

39. FineCollected:

Description: A table which just stores the details about a transaction that incurred a fine.

SQL:

```
create table roomreservationnulllog
(
CheckoutTime Timestamp,
DurationReturn timestamp,
ReturnTime timestamp,
studentnumber varchar(10),
facultynumber varchar(10),
TypeOfResource varchar(20),
Fine Number
);
```

Functional Dependency:
No Functional Dependency

Normal Form:

Indecisive