Physical Model

➢ TABLE DEPARTMENT :-

Name	Type	Size	Constraint
deptID	INT	DEFAULT	PRIMARY KEY
deptName	VARCHAR2	45	NOT NULL
officeNo	INT	DEFAULT	NOT NULL

```
CREATE TABLE Department(
   deptID INT PRIMARY KEY,
   deptName VARCHAR2(45) NOT NULL,
   officeNo INT NOT NULL
);
```

➢ TABLE EMPLOYEE :-

Name	Type	Size	Constraint
employeeID	INT	DEFAULT	PRIMARY KEY
fullName	VARCHAR2	45	NOT NULL
contactNo	INT	DEFAULT	UNIQUE,NOT NULL
email	VARCHAR2	45	UNIQUE,NOT NULL
address	VARCHAR2	60	NOT NULL
hiringDate	DATE	DEFAULT	
deptID	INT	DEFAULT	NOT NULL,FOREIGN KEY

```
CREATE TABLE EMPLOYEE(
   employeeID INT PRIMARY KEY,
   fullName VARCHAR2(45) NOT NULL,
   contactNo INT UNIQUE NOT NULL,
   email VARCHAR2(45) UNIQUE NOT NULL,
   address VARCHAR2(60) NOT NULL,
   hiringDate DATE,
   deptID INT NOT NULL,
   FOREIGN KEY (deptID) references DEPARTMENT(DEPTID) ON DELETE CASCADE
);
```

➢ TABLE EMPLOYEE_SAL_DETAILS :-

Name	Type	Size	Constraint
accountNo	INT	DEFAULT	PRIMARY KEY
employeeID	INT	DEFAULT	UNIQUE,NOT NULL,FOREIGN KEY
accountName	VARCHAR2	45	NOT NULL
salary	FLOAT	DEFAULT	NOT NULL
deductions	FLOAT	DEFAULT	

```
CREATE TABLE EMPLOYEE_SAL_DETAILS(
   accountNo INT PRIMARY KEY,
   employeeID INT UNIQUE NOT NULL,
   accountName VARCHAR2(45) NOT NULL,
   salary FLOAT NOT NULL,
   deductions FLOAT,
   FOREIGN KEY (employeeID) references EMPLOYEE(employeeID) ON DELETE CASCADE
);
```

➢ TABLE ADMINISTRATION :-

Name	Type	Size	Constraint
administratorID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY
designation	VARCHAR2	45	

```
CREATE TABLE ADMINISTRATION(
   administratorID INT PRIMARY KEY,
   designation VARCHAR2(45),
   FOREIGN KEY (administratorID) REFERENCES EMPLOYEE(employeeID) ON DELETE CASCADE
);
```

➢ TABLE MENTOR :-

Name	Type	Size	Constraint
mentorID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY

```
CREATE TABLE MENTOR(
   mentorID INT PRIMARY KEY,
   FOREIGN KEY (mentorID) REFERENCES EMPLOYEE(employeeID) ON DELETE CASCADE
);
```

➢ TABLE INSTRUCTOR :-

Name	Type	Size	Constraint
instructorID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY

```
CREATE TABLE INSTRUCTOR(
   instructorID INT PRIMARY KEY,
   FOREIGN KEY (instructorID) REFERENCES EMPLOYEE(employeeID) ON DELETE CASCADE
);
```

➢ TABLE GUARDIAN :-

Name	Type	Size	Constraint
guardianID	INT	DEFAULT	PRIMARY KEY
fatherName	VARCHAR2	45	NOT NULL
motherName	VARCHAR2	45	NOT NULL
address	VARCHAR2	60	NOT NULL
phoneNo	INT	DEFAULT	NOT NULL

```
CREATE TABLE GUARDIAN(
   guardianID INT PRIMARY KEY,
   fatherName VARCHAR2(45) NOT NULL,
   motherName VARCHAR2(45) NOT NULL,
   address VARCHAR2(60) NOT NULL,
   phoneNo INT NOT NULL
);
```

➢ TABLE STUDENT :-

Name	Type	Size	Constraint
studentID	INT	DEFAULT	PRIMARY KEY
fullName	VARCHAR2	45	NOT NULL
DOB	DATE	DEFAULT	NOT NULL
currAddress	VARCHAR2	DEFAULT	NOT NULL
gender	CHAR	1	NOT NULL
phoneNo	INT	DEFAULT	UNIQUE,NOT NULL
email	VARCHAR2	40	UNIQUE,NOT NULL
deptID	INT	DEFAULT	NOT NULL,FOREIGN KEY
guardianID	INT	DEFAULT	NOT NULL,FOREIGN KEY

```
CREATE TABLE Student(
   studentID INT PRIMARY KEY,
   fullName VARCHAR2(45) NOT NULL,
   DOB DATE NOT NULL,
```

```
   currAddress VARCHAR2(60) NOT NULL,
   gender CHAR(1) NOT NULL,
   phoneNo INT UNIQUE NOT NULL,
   email VARCHAR2(40) UNIQUE NOT NULL,
   deptID INT NOT NULL,
   guardianID INT NOT NULL,
   FOREIGN KEY (deptID) references DEPARTMENT(deptID) ON DELETE CASCADE,
   FOREIGN KEY (guardianID) references GUARDIAN(guardianID) ON DELETE CASCADE
);
```

➢ TABLE COURSE :-

Name	Type	Size	Constraint
courseID	INT	DEFAULT	PRIMARY KEY
courseName	VARCHAR2	45	NOT NULL
credits	INT	DEFAULT	NOT NULL
hours	INT	DEFAULT	NOT NULL
numLectures	INT	DEFAULT	NOT NULL
deptID	INT	DEFAULT	NOT NULL,FOREIGN KEY

```
CREATE TABLE Course(
   courseID INT PRIMARY KEY,
   courseName VARCHAR2(45) NOT NULL,
   credits INT NOT NULL,
   hours INT NOT NULL,
   numLectures INT NOT NULL,
   deptID INT NOT NULL,
   FOREIGN KEY (deptID) references DEPARTMENT(DEPTID) ON DELETE CASCADE
);
```

➢ TABLE STUD_VISADETAILS :-

Name	Type	Size	Constraint
studentID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY
visaNo	INT	DEFAULT	UNIQUE,NOT NULL
visaStatus	VARCHAR2	40	NOT NULL

```
CREATE TABLE stud_visaDetails(
   studentID INT PRIMARY KEY,
   visaNo INT UNIQUE NOT NULL,
   visaStatus VARCHAR2(40) NOT NULL,
   FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE CASCADE
);
```

➢ TABLE COOP :-

Name	Type	Size	Constraint
coopID	INT	DEFAULT	PRIMARY KEY
company_Name	VARCHAR2	45	NOT NULL
position	VARCHAR2	45	NOT NULL

```
CREATE TABLE coop(
   coopID INT PRIMARY KEY,
   company_Name VARCHAR2(45) NOT NULL,
   position VARCHAR2(45) NOT NULL
);
```

➢ TABLE STUDENT_HAS_COOP :-

Name	Type	Size	Constraint
studentID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY
duration	INT	DEFAULT	NOT NULL
salary	INT	DEFAULT	NOT NULL
coopID	INT	DEFAULT	NOT NULL,FOREIGN KEY

```
CREATE TABLE student_has_coop(
   studentID INT PRIMARY KEY,
   duration INT NOT NULL,
   salary INT NOT NULL,
   coopID INT NOT NULL,
   FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE CASCADE,
   FOREIGN KEY (coopID) REFERENCES coop(coopID) ON DELETE CASCADE
);
```

➢ TABLE studAcademicRecord :-

Name	Type	Size	Constraint
studentID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY
lastCGPA	FLOAT	DEFAULT	NOT NULL
currCGPA	FLOAT	DEFAULT	NOT NULL

```
CREATE TABLE studAcademicRecord(
   studentID INT PRIMARY KEY,
   lastCGPA FLOAT NOT NULL,
   currCGPA FLOAT NOT NULL,
   FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE CASCADE
);
```

➢ TABLE ATTENDANCE :-

Name	Type	Size	Constraint
studentID	INT	DEFAULT	NOT NULL
curAttendance	INT	DEFAULT	NOT NULL
totaleAttendance	INT	DEFAULT	NOT NULL
courseID	INT	DEFAULT	NOT NULL
studentID , courseID			PRIMARY KEY,FOREIGN KEY

```
CREATE TABLE attendance(
   studentID INT NOT NULL,
   curAttendance INT NOT NULL,
   totalAttendance INT NOT NULL,
   courseID INT NOT NULL,
   PRIMARY KEY(studentID,courseID),
   FOREIGN KEY (studentID,courseID) REFERENCES student_registers_courses(studentID,courseID)
   ON DELETE CASCADE
);
```

➢ TABLE GradeReport :-

Name	Type	Size	Constraint
reportID	INT	DEFAULT	PRIMARY KEY
courseID	INT	DEFAULT	NOT NULL,FOREIGN KEY
studentID	INT	DEFAULT	NOT NULL,FOREIGN KEY
numericGrade	FLOAT	DEFAULT	NOT NULL
semester	INT	DEFAULT	NOT NULL

```
CREATE TABLE GradeReport(
   reportID INT PRIMARY KEY,
   courseID INT NOT NULL,
   studentID INT NOT NULL,
   numericGrade FLOAT NOT NULL,
   semester INT NOT NULL,
   FOREIGN KEY (courseID) REFERENCES Course(courseID) ON DELETE CASCADE,
   FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE CASCADE
);
```

➢ TABLE FEE_PAYMENT :-

Name	Type	Size	Constraint
billID	INT	DEFAULT	PRIMARY KEY
studentID	INT	DEFAULT	NOT NULL,FOREIGN KEY
date_of_payment	DATE	DEFAULT	NOT NULL
totalAmnt	INT	DEFAULT	NOT NULL
Amount_Paid	INT	DEFAULT	NOT NULL
semester	INT	DEFAULT	NOT NULL

```
CREATE TABLE Fee_Payment(
   billID INT PRIMARY KEY,
   studentID INT NOT NULL,
   date_of_payment DATE NOT NULL,
   totalAmnt INT NOT NULL,
   Amount_Paid INT NOT NULL,
   semester INT NOT NULL,
   FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE CASCADE
);
```

➢ TABLE FeeDues :-

Name	Type	Size	Constraint
billID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY
dueAmnt	INT	DEFAULT	NOT NULL

```
CREATE TABLE feeDues(
   billID INT PRIMARY KEY,
   dueAmnt INT NOT NULL,
   FOREIGN KEY (billID) REFERENCES Fee_Payment(billID) ON DELETE CASCADE
);
```

➢ TABLE STUDENT_REGISTERS_COURSES :-

Name	Type	Size	Constraint
studentID	INT	DEFAULT	NOT NULL,FOREIGN KEY
courseID	INT	DEFAULT	NOT NULL,FOREIGN KEY
registrationDate	DATE	DEFAULT	
(studentID,courseID)			PRIMARY KEY

```
CREATE TABLE student_registers_courses(
   studentID INT NOT NULL,
   courseID INT NOT NULL,
   registrationDate DATE,
   PRIMARY KEY(studentID,courseID),
   FOREIGN KEY (courseID) REFERENCES Course(courseID) ON DELETE CASCADE,
   FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE CASCADE
);
```

➢ TABLE BUILDING :-

Name	Type	Size	Constraint
buildingID	INT	DEFAULT	PRIMARY KEY
buildingName	CHAR	1	NOT NULL

```
CREATE TABLE BUILDING(
   buildingID INT PRIMARY KEY,
   buildingName VARCHAR(45) NOT NULL
);
```

➢ TABLE ROOM :-

Name	Type	Size	Constraint
roomNO	INT	DEFAULT	PRIMARY KEY
buildingID	INT	DEFAULT	NOT NULL,FOREIGN KEY
capacity	INT	DEFAULT	NOT NULL

```
CREATE TABLE ROOM(
   roomNO INT PRIMARY KEY,
   buildingID INT NOT NULL,
   capacity INT NOT NULL,
   FOREIGN KEY (buildingID) references building(buildingID) ON DELETE CASCADE
);
```

➢ TABLE GROUPS :-

Name	Type	Size	Constraint
groupNo	INT	DEFAULT	PRIMARY KEY
courseID	INT	DEFAULT	NOT NULL,FOREIGN KEY
instructorID	INT	DEFAULT	NOT NULL,FOREIGN KEY
mentorID	INT	DEFAULT	NOT NULL,FOREIGN KEY
buildingID	INT	DEFAULT	NOT NULL,FOREIGN KEY

```
CREATE TABLE GROUPS(
   groupNo INT PRIMARY KEY,
   courseID INT NOT NULL,
   instructorID INT NOT NULL,
   mentorID INT NOT NULL,
   buildingID INT NOT NULL,
   FOREIGN KEY (courseID) REFERENCES Course(courseID) ON DELETE CASCADE,
   FOREIGN KEY (instructorID) REFERENCES INSTRUCTOR(instructorID) ON DELETE CASCADE,
   FOREIGN KEY (mentorID) REFERENCES MENTOR(mentorID) ON DELETE CASCADE,
   FOREIGN KEY (buildingID) REFERENCES BUILDING(buildingID) ON DELETE CASCADE
);
```

➢ TABLE GROUP_DAYSLOT :-

Name	Type	Size	Constraint
day	CHAR	1	NOT NULL
groupNo	INT	DEFAULT	NOT NULL,FOREIGN KEY
startTime	INT	DEFAULT	NOT NULL
endTime	INT	DEFAULT	NOT NULL

```
CREATE TABLE Group_dayslot(
   day VARCHAR(10) NOT NULL,
   groupNo INT NOT NULL,
   startTime INTERVAL DAY TO SECOND(2) NOT NULL,
   endTime INTERVAL DAY TO SECOND(2) NOT NULL,
   FOREIGN KEY (groupNo) REFERENCES GROUPS(groupNo) ON DELETE CASCADE
);
```

➢ TABLE ASSIGNMENT :-

Name	Type	Size	Constraint
assignmentID	INT	DEFAULT	PRIMARY KEY
groupNo	INT	DEFAULT	NOT NULL,FOREIGN KEY
courseID	INT	DEFAULT	NOT NULL,FOREIGN KEY
deadlineDate	DATE	DEFAULT	NOT NULL
maxScore	INT	DEFAULT	NOT NULL

```
CREATE TABLE assignment(
   assignmentID INT PRIMARY KEY,
   groupNo INT NOT NULL,
   courseID INT NOT NULL,
   deadlineDate DATE NOT NULL,
   maxScore INT NOT NULL,
   FOREIGN KEY (groupNo) REFERENCES GROUPS(groupNo) ON DELETE CASCADE,
   FOREIGN KEY (courseID) REFERENCES COURSE(COURSEID) ON DELETE CASCADE
);
```

➢ TABLE ASSIGNMENT_SUBMISSIONS :-

Name	Type	Size	Constraint
assignmentID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY
studentID	INT	DEFAULT	NOT NULL,FOREIGN KEY
submissionDate	DATE	DEFAULT	NOT NULL
evaluationDate	DATE	DEFAULT	NOT NULL
marksObtained	INT	DEFAULT	NOT NULL

```
CREATE TABLE assignment_submissions(
   assignmentID INT PRIMARY KEY,
   studentID INT NOT NULL,
   submissionDate DATE NOT NULL,
   evaluationDate DATE NOT NULL,
   marksObtained INT NOT NULL,
   FOREIGN KEY (assignmentID) REFERENCES assignment(assignmentID) ON DELETE CASCADE,
   FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE CASCADE
);
```

➢ TABLE LIBRARIAN :

Name	Type	Size	Constraint
librarianID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY

```
CREATE TABLE Librarian(
   librarianID INT PRIMARY KEY,
   FOREIGN KEY (librarianID) REFERENCES EMPLOYEE(employeeID) ON DELETE CASCADE
);
```

➢ TABLE BOOK :-

Name	Type	Size	Constraint
bookID	INT	DEFAULT	PRIMARY KEY
bookName	VARCHAR2	45	

```
CREATE TABLE BOOK(
   bookID INT PRIMARY KEY,
   bookName VARCHAR2(45)
);
```

➢ TABLE LIBRARIAN_MAINTAINS :-

Name	Type	Size	Constraint
bookID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY
librarianID	INT	DEFAULT	NOT NULL,FOREIGN KEY
amount	FLOAT	DEFAULT	NOT NULL
quantity	INT	DEFAULT	NOT NULL

```
CREATE TABLE Librarian_maintains(
   bookID INT PRIMARY KEY,
   librarianID INT NOT NULL,
   amount FLOAT NOT NULL,
   quantity INT NOT NULL,
   FOREIGN KEY (bookID) references BOOK(bookID) ON DELETE CASCADE,
   FOREIGN KEY (librarianID) references Librarian(librarianID) ON DELETE CASCADE
);
```

➢ TABLE BOOK_HAS_COPIES :-

Name	Type	Size	Constraint
copyNo	INT	DEFAULT	NOT NULL
bookID	INT	DEFAULT	NOT NULL , FOREIGN KEY
status	CHAR	1	NOT NULL
copyNo,bookID			PRIMARY KEY

```
CREATE TABLE Book_has_copies(
   copyNo INT NOT NULL,
   bookID INT NOT NULL,
   status CHAR NOT NULL,
   PRIMARY KEY(copyNo,bookID),
   FOREIGN KEY (bookID) references BOOK(bookID) ON DELETE CASCADE
);
```

➢ TABLE INSTRUCTOR_HAS_BOOKS :-

Name	Type	Size	Constraint
copyNo	INT	DEFAULT	NOT NULL
bookID	INT	DEFAULT	NOT NULL
instructorID	INT	DEFAULT	NOT NULL,FOREIGN KEY
issueDate	DATE	DEFAULT	NOT NULL
returnDate	DATE	DEFAULT	-
copyNo,bookID			PRIMARY KEY , FOREIGN KEY

```
CREATE TABLE instructor_has_books(
   copyNo INT PRIMARY KEY,
   instructorID INT NOT NULL,
   issueDate DATE NOT NULL,
   returnDate DATE,
   FOREIGN KEY (copyNo) references Book_has_copies(copyNo) ON DELETE CASCADE,
   FOREIGN KEY (instructorID) references INSTRUCTOR(instructorID) ON DELETE CASCADE
);
```

➢ TABLE STUDENT_ISSUES_BOOKS :-

Name	Type	Size	Constraint
copyNo	INT	DEFAULT	NOT NULL
bookID	INT	DEFAULT	NOT NULL
studentID	INT	DEFAULT	NOT NULL,FOREIGN KEY
issueDate	DATE	DEFAULT	NOT NULL
returnDate	DATE	DEFAULT	-
copyNo,bookID			PRIMARY KEY , FOREIGN KEY

```
CREATE TABLE student_issues_books(
   copyNo INT PRIMARY KEY,
   studentID INT NOT NULL,
   issueDate DATE NOT NULL,
   returnDate DATE,
   FOREIGN KEY (copyNo) references Book_has_copies(copyNo) ON DELETE CASCADE,
   FOREIGN KEY (studentID) references Student(studentID) ON DELETE CASCADE
);
```

➢ Sequences:-

These 7 sequences are used to make it easy for insertion of Primary Key values , IDs and to get rid of checking the previous value .

```
create sequence ForStudents start with 1;
create sequence ForGuardians start with 1;
create sequence ForDept start with 10 increment by 10;
create sequence ForEmp start with 100;
create sequence ForGroups start with 1;
create sequence ForBookCopies start with 1;
create sequence ForCourses start with 10 increment by 10;
```

➢ Triggers:-

➢ Trigger to update the status of the book when any student issues/returns the book.

```
CREATE OR REPLACE TRIGGER update_Stud_issue_Status
AFTER INSERT OR UPDATE OR DELETE ON STUDENT_ISSUES_BOOKS
FOR EACH ROW
DECLARE
   currStatus CHAR;
BEGIN
   IF inserting THEN
      select status into currStatus from BOOK_HAS_COPIES where (copyNo =
      :New.copyNo
      AND bookID = :New.bookID);
      IF currStatus='N' THEN raise_application_error( -20001, 'BOOK
      ALREADY ISSUED!!!');
      ELSE update book_has_copies set status = 'N' where (copyNo =
      :New.copyNo AND bookID = :New.bookID);
       END IF;
    ELSIF (updating AND (:NEW.returnDate IS NOT NULL)) OR deleting THEN update
    book_has_copies set status = 'Y' where (copyNo = :OLD.copyNo AND bookID =
    :OLD.bookID);
   END IF;
END;
```

- Trigger to update the status of the book when any instructor issues/returns the book.

```
CREATE OR REPLACE TRIGGER update_instructor_issue_Status
AFTER INSERT OR UPDATE OR DELETE ON INSTRUCTOR_HAS_BOOKS
FOR EACH ROW
DECLARE
  currStatus CHAR;
BEGIN
  IF inserting THEN
    select status into currStatus from BOOK_HAS_COPIES where (copyNo
    = :New.copyNo AND bookID = :New.bookID);
    IF currStatus='N' THEN raise_application_error( -20001, 'BOOK
    ALREADY ISSUED!!!');
    ELSE update book_has_copies set status = 'N' where (copyNo =
    :New.copyNo AND bookID = :New.bookID);
    END IF;
  ELSIF (updating AND (:NEW.returnDate IS NOT NULL)) OR deleting
  THEN update book_has_copies set status = 'Y' where (copyNo =
  :OLD.copyNo AND bookID = :OLD.bookID);
  END IF;
END;
```

- Trigger to insert the billID of fee_payment table into dues table if there is some due.

```
CREATE TRIGGER FEE_PAYMENT_TRIGGER
AFTER INSERT ON FEE_PAYMENT
FOR EACH ROW
DECLARE
due NUMBER:=:NEW.totalAmnt-:NEW.AMOUNT_PAID;
BEGIN
IF due!=0 THEN
insert into FEEDUES values(:NEW.billID,due);
END IF;
END;
```

- Trigger to insert default attendance values in attendance table whenever student registers a new course.

```
CREATE OR REPLACE TRIGGER on_stud_courses_registration
AFTER INSERT ON student_registers_courses
FOR EACH ROW
BEGIN
INSERT INTO ATTENDANCE VALUES(:NEW.STUDENTID,0,0,:NEW.COURSEID);
END;
```