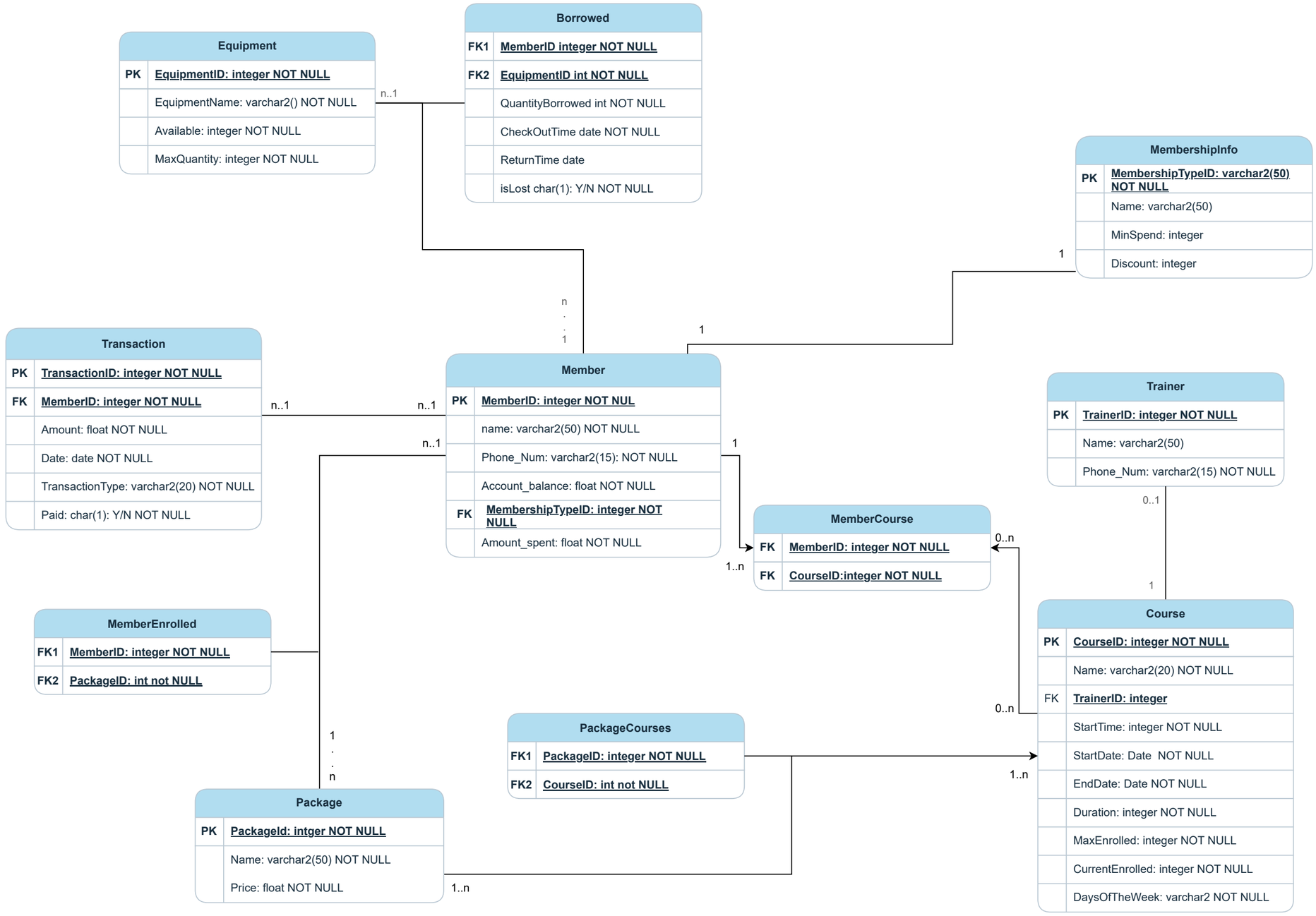
Equipment
PK EquipmentID: integer NOT NULL
EquipmentName: varchar2() NOT NULL
Available: integer NOT NULL
MaxQuantity: integer NOT NULL
n..1
Borrowed
FK1 MemberID integer NOT NULL
FK2 EquipmentID int NOT NULL
QuantityBorrowed int NOT NULL
CheckOutTime date NOT NULL
ReturnTime date
isLost char(1): Y/N NOT NULL
MembershipInfo
PK MembershipTypeID: varchar2(50) NOT NULL
Name: varchar2(50)
MinSpend: integer
Discount: integer
1
n
.
.
1
1
Transaction
PK TransactionID: integer NOT NULL
FK MemberID: integer NOT NULL
Amount: float NOT NULL
Date: date NOT NULL
TransactionType: varchar2(20) NOT NULL
Paid: char(1): Y/N NOT NULL
n..1
n..1
Member
PK MemberID: integer NOT NUL
name: varchar2(50) NOT NULL
Phone_Num: varchar2(15): NOT NULL
Account_balance: float NOT NULL
FK MembershipTypeID: integer NOT NULL
Amount_spent: float NOT NULL
n..1
1
1..n
Trainer
PK TrainerID: integer NOT NULL
Name: varchar2(50)
Phone_Num: varchar2(15) NOT NULL
0..1
1
MemberCourse
FK MemberID: integer NOT NULL
FK CourseID:integer NOT NULL
0..n
0..n
MemberEnrolled
FK1 MemberID: integer NOT NULL
FK2 PackageID: int not NULL
1
.
.
n
PackageCourses
FK1 PackageID: integer NOT NULL
FK2 CourseID: int not NULL
1..n
Course
PK CourseID: integer NOT NULL
Name: varchar2(20) NOT NULL
FK TrainerID: integer
StartTime: integer NOT NULL
StartDate: Date NOT NULL
EndDate: Date NOT NULL
Duration: integer NOT NULL
MaxEnrolled: integer NOT NULL
CurrentEnrolled: integer NOT NULL
DaysOfTheWeek: varchar2 NOT NULL
Package
PK PackageId: intger NOT NULL
Name: varchar2(50) NOT NULL
Price: float NOT NULL
1..n

Names:
Michael Beccarelli
Quanwei Lei
JhihYang Wu

DESIGN

PERSONALIZED DETAILS (subject to change):
($200) Junior package - [Strength 001, Yoga 001]
($300) Senior package - [Strength 002, Yoga 002]
($400) Strength package - [Strength 001, Strength 002]

CONNECTION DESIGN:

- Members need to be connected to:
    - Transactions
    - Courses (note: members enroll in packages not individual courses)
    - Package
    - Borrowed transactions
    - Membership information

- Trainers need to be connected to:
    - Courses (the ones they teach)

- Informative tables:
    - Membership_info
    - Member
    - Trainer
    - Course
    - Package
    - Equipment
    - Transaction

- Relational tables:
    - MemberEnrolled
    - PackageCourses
    - MemberCourse
    - Borrowed

PK
FK

Member:
- MemberId (integer)
- Name (varchar2)
- Tele-num (varchar2)
- MembershipID (varchar2)
- Balance (float)
- AmountSpent (float)

MemberShip_info:
- MembershipID (integer)
- Min_Spend (float)
- Discount Amount (Integer)

MemberEnrolled: (Relational)
- MemberId (Integer)
- PackageID (Integer)

MemberCourse: (Relational):
- MemberId (Integer)
- CourseID (Integer)

Trainer:
- Trainer_Id (integer)
- Name (varchar2)
- Tel-num (varchar2)

Package:
- PackageId (integer)
- PackageName (varchar2)
- Price (float)

PackageCourses: (Relational)
- PackageID (Integer)
- CourseID (Integer)

Course:
- Course_Id (integer)
- Course _Name (varchar2)
- TrainerID (integer)
- Course_Time (integer)
- Start_Date (Date)

- End_Date (Date)
- Duration (integer)
- Max_Enroll (integer)
- Currently_Enrolled (integer) (update on member deletion)
- DaysOfTheWeek (varchar2)

Transaction
- TransactionID (Integer)
- MemberId (integer)
- Amount (float)
- Date (date)
- TransactionType (Recharge record, package purchase) (varchar2)
- Paid (char: Y/N)

Equipment
- EquipmentId (Integer)
- Equipment name (varchar2)
- Available (based on borrowed) (Integer)
- Max_Quantity (inventory and losses) (Integer)

Borrowed (Relational)
- MemberId (Integer)
- EquipmentId (Integer)
- Quantity Borrowed (Integer)
- Check out time (Date)
- Return Time (Date)
- isLost (char: Y/N) (char)

INTERACTIONS:

Deleting a member:

- Verify unreturned equipment, if any exist, mark as lost.
- If a member is actively participating in any course, remove their slot and delete their participation records.
- If a member is deleted do we keep the transactions they participated in?

Adding a member:

- Get basic information
- Prompt for open package enrollment (price and package)
- Add member and update MemberEnroll, MemberCourse

Adding a course:

- Get basic information about the course
- Update Course table

Deleting a course:

- Print names and numbers of the member current in the course (use MemberCourses)
- Update Courses, Package, PackageCourses

Adding a package:

- Get basic information about the package
- Show all available courses and prompt for course addition
- Add courses to package
- Update package table

Deleting a package:

- Get package information
- Update tables Package, PackageCourses

Updating a package:

- Get basic package information
- Prompt to change price/courses
- Update tables package, PackageCourses

CONSTRAINTS:

Do not affect member's courses on a package change.

Check member transaction history for unpaid amounts, if found print them out, and prevent deletion of said account.

Only 1 Trainer per course.

Trainer's individual teaching times cannot overlap.

TABLES:

```
create table Transaction (
        TransactionID integer,
        MemberID integer,
        Amount float,
        TransactionDate Date,
        TransactionType varchar2(100),
        Paid char(1),
        primary key (TransactionID)
    )

create table Equipment (
        EquipmentID integer,
        EquipmentName varchar2(100),
        Available integer,
        MaxQuantity integer,
        primary key (EquipmentID)
    )

create table Borrowed (
        MemberID integer,
        EquipmentID integer,
        QuantityBorrowed integer,
        CheckOutTime Date,
        ReturnTime Date,
        isLost char(1),
        primary key (MemberID, EquipmentID, CheckOutTime)
    )

create table Member (
        MemberID integer,
        Name varchar2(50),
        PhoneNum char(12),
        AccountBalance float,
        MembershipTypeID integer,
        AmountSpent float,
        primary key (MemberID)
    )

create table MemberEnrolled (
        MemberID integer,
        PackageID Integer,
        primary key (MemberID, PackageID)
```

```
    )

create table Package (
        PackageID integer,
        Name varchar2(50),
        Price float,
        primary key (PackageID)
    )

create table PackageCourses (
        PackageID integer,
        CourseID Integer,
        primary key (PackageID, CourseID)
    )

create table MembershipInfo (
        MembershipTypeID integer,
        Name varchar2(50),
        MinSpend float,
        Discount integer,
        primary key (MembershipTypeID, Name)
    )

create table Trainer (
        TrainerID integer,
        Name varchar2(50),
        PhoneNum char(12),
        primary key (TrainerID, PhoneNum)
    )

create table Course (
        CourseID integer,
        Name varchar2(100),
        TrainerID integer,
        StartTime integer,
        StartDate Date,
        EndDate Date,
        Duration integer,
        MaxEnrolled integer,
        CurrentEnrolled integer,
        DaysOfTheWeek varchar2(14),
        primary key (CourseID, Name)
    )
```

```
create table MemberCourse (
        MemberID integer,
        CourseID integer,
        primary key (MemberID, CourseID)
    )
```

Normalization Analysis:

Table: <table_name>

- PK - <PK attribute>
- FK - <FK attribute(s)>
- FDs:
    - …
    - …
- Adhere? Y/N
    - Reason: …

Table: Equipment

- PK - EquipmentID
- FK - None
- FDs:
    - EquipmentID -> EquipmentName
    - EquipmentID -> Available
    - EquipmentID -> MaxQuantity
- Adhere? Y
    - This table does not contain any partial or transitive dependencies. All of the columns are uniquely named and data types are consistent among the columns.

Table: Borrowed

- PK - MemberID+EquipmentID
- FK - MemberID, EquipmentID
- FDs:
    - MemberID+EquipmentID -> QuantityBorrowed
    - MemberID+EquipmentID -> CheckOutTime
    - MemberID+EquipmentID -> ReturnTime
    - MemberID+EquipmentID -> isLost
- Adhere? Y
    - This table does not contain any transitive dependencies and because we have two attributes making up the PK those attributes do not have a partial dependency. All of the columns are uniquely named and data type is consistent among them.

Table: Transaction

- PK - TransactionID
- FK - None
- FDs:
    - TransactionID -> Amount
    - TransactionID -> Date
    - TransactionID -> TransactionType
    - TransactionID -> Paid

- Adhere? Y
    - This table does not contain any partial or transitive dependencies. All of the columns are uniquely named and data type is consistent among them.

Table: Member
- PK - MemberID
- FK - MembershipType
- FDs:
    - MemberID -> name
    - MemberID -> Phone_Num
    - MemberID -> Account_Balance
    - MemberID -> Amount_Spent
    - MemberID -> MembershipType
- Adhere? Y
    - This table does not contain any partial or transitive dependencies. All of the columns are uniquely named and data type is consistent among them.

Table: MembershipInfo
- PK - MemberTypeID
- FK - None
- FDs:
    - MemberTypeID -> Name
    - MemberTypeID -> MinSpend
    - MemberTypeID -> Discount
- Adhere? Y
    - This table does not contain any partial or transitive dependencies. All of the columns are uniquely named and data type is consistent among them.

Table: MemberEnrolled
- PK - MemberID+PackageID
- FK - MemberID, PackageID
- FDs: None
- Adhere? Y
    - This table is primarily used as a relational table. It does not contain any transitive dependencies and because the PK is created by the two attributes in this table, there is no partial dependency here either.

Table: Package
- PK - PackageID
- FK - None
- FDs:
    - PackageID -> Name
    - PackageID -> Price
- Adhere? Y

    - This table does not contain any partial or transitive dependencies. All of the columns are uniquely named and data type is consistent among them.

Table: PackageCourses
- PK - PackageID+CourseID
- FK - PackageID, CourseID
- FDs: None
- Adhere? Y
    - This table is primarily used as a relational table. It does not contain any transitive dependencies and because the PK is created by the two attributes in this table, there is no partial dependency here either.

Table: MemberCourse
- PK - MemberID+CourseID
- FK - MemberID, CourseID
- FDs: None
- Adhere? Y
    - This table is primarily used as a relational table. It does not contain any transitive dependencies and because the PK is created by the two attributes in this table, there is no partial dependency here either.

Table: Course
- PK - CourseID
- FK - TrainerID
- FDs:
    - CourseID -> Name
    - CourseID -> trainerID
    - CourseID -> StartTime
    - CourseID -> StartDate
    - CourseID -> EndDate
    - CourseID -> duration
    - CourseID -> MaxEnrolled
    - CourseID -> CurrenEnrolled
    - CourseID -> DaysOftheWeek
- Adhere? Y
    - This table does not contain any partial or transitive dependencies. All of the columns are uniquely named and data type is consistent among them.

Table: Trainer
- PK - TrainerID
- FK - None
- FDs:
    - TrainerID -> Name
    - TrainerID -> Phone_num

    - MemberID+EquipmentID -> ReturnTime
    - MemberID+EquipmentID -> isLost
- Adhere? Y
    - This table does not contain any partial or transitive dependencies. All of the columns are uniquely named and data type is consistent among them.

Query 4 Description:

Prompt: Get the names and phone numbers of all of the trainers for a specific member.

Query: SELECT DISTINCT T.Name, T.PhoneNum FROM Trainer T JOIN Course C ON T.TrainerID = C.TrainerID JOIN MemberCourse MC ON C.CourseID = MC.CourseID WHERE MC.MemberID = <memberID>

Explanation: This is a common tool inside and outside of gym applications, such as in university environments, students are able to find the email addresses and the names of their advisors and professors in one common place most of the time. This is essentially the same service and it allows the admin and members to see the member to trainer relationship.