Boyce Codd
Health Club

BOYCE CODD HEALTH CLUB

Database Design Proposal

1 May 2015

Megan Poyntz

Table of Contents

Executive Summary

Overview

On average, more than 4.8 million people use a gym monthly.[1] With their popularity and the high demand of memberships, health clubs are increasingly in need of a structured, well-designed database that will allow all the processes of the business of to run smoothly. From providing customers with the best service to efficiently managing employees, a database design that will provide consistent and accurate data is the key to a health club's success. Not only can a well-designed database help maintain a health club's processes, but it will also help in giving the health club a competitive advantage as it will provide timely and useful information to help the business grow.

Objectives

In this design proposal, a database for Boyce Codd Health Club is explored in detail. The purpose of the design proposal is to show a database that will help streamline processes, such as membership registration, scheduling fitness classes, booking the correct equipment, and much more. Although there is a set amount of data in this proposal, it is designed to allow for not only additional data, but growth in the structure, as well. Through diagrams, queries and examples, this design will show how useful a database can be to health clubs and similar business.

[1] "Gym Membership Statistics." *Statistic Brain RSS*. Web. 28 Apr. 2015.

Entity-Relationship Diagram

Boyce Codd Health Club

Entity-Relationship Diagram

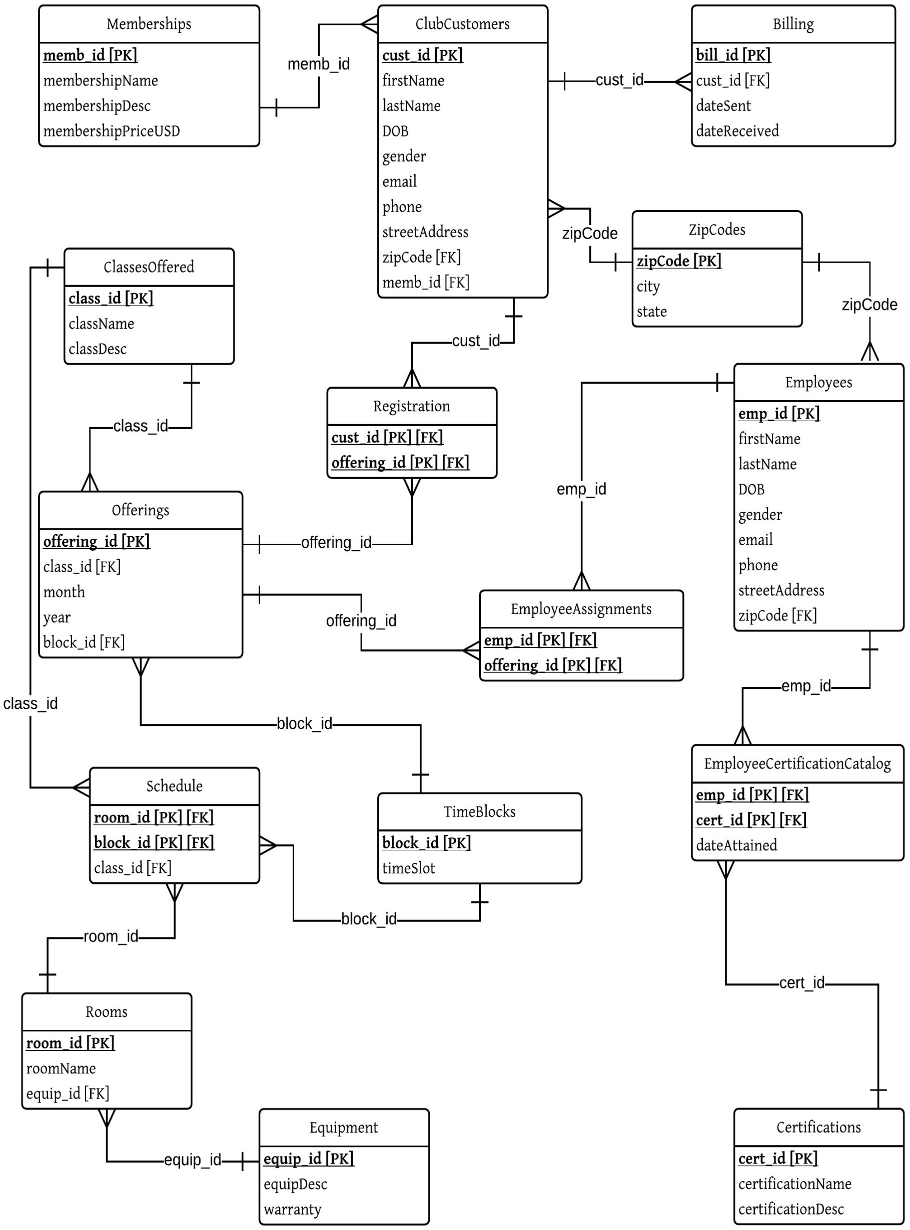


Table: Zip Codes

Explanation:

This table holds the Zip Codes and corresponding Cities and States for each Zip Code. The Zip Codes apply to both Employee and Club Customer profiles.

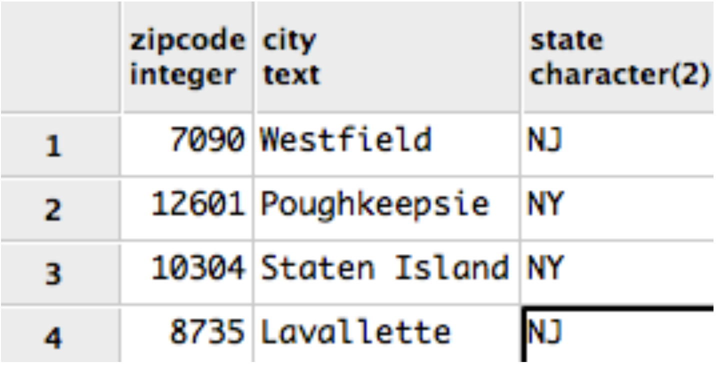
Create Statement:

```
CREATE TABLE ZipCodes(
   zipCode INT NOT NULL,
   city TEXT NOT NULL,
   state CHAR(2) NOT NULL,
PRIMARY KEY (zipCode)
);
```

Functional Dependencies:

zipCode → city, state

Sample Data:



	zipcode	city	state
	integer	text	character(2)
1	7090	Westfield	NJ
2	12601	Poughkeepsie	NY
3	10304	Staten Island	NY
4	8735	Lavallette	NJ

Table: Classes Offered

Explanation:

This table lists the fitness classes offered at the Health Club. This table is not to be confused with a schedule or the classes offered at a certain time. This is just a general “course catalog” that shows the types of classes the Health Club can offer. This table lists the class_id, class name and class description.

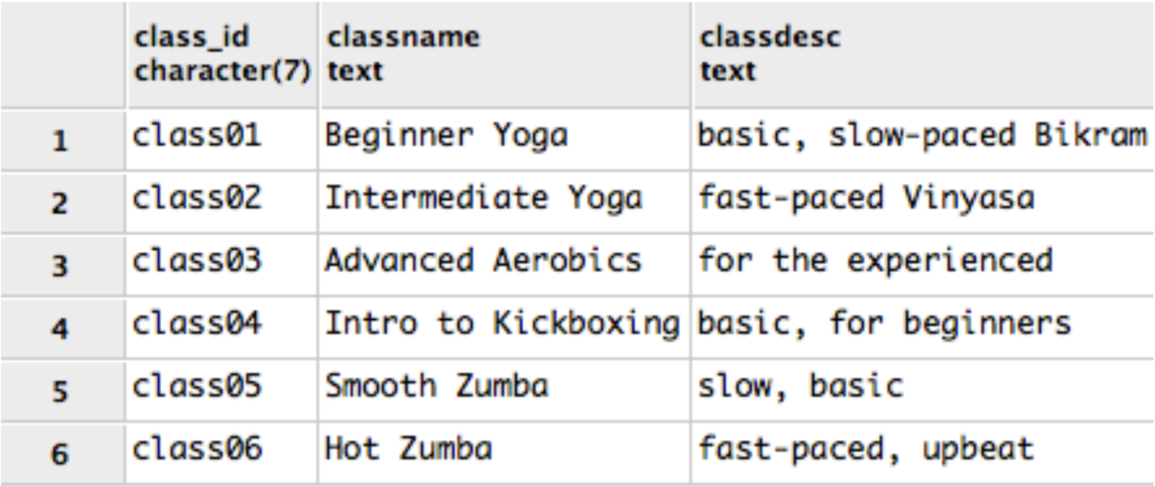
Create Statement:

```
CREATE TABLE ClassesOffered(
    class_id CHAR(7) NOT NULL,
    className TEXT NOT NULL,
    classDesc TEXT NOT NULL,
PRIMARY KEY (class_id)
);
```

Functional Dependencies:

class_id → className, classDesc

Sample Data:



	class_id character(7)	classname text	classdesc text
1	class01	Beginner Yoga	basic, slow-paced Bikram
2	class02	Intermediate Yoga	fast-paced Vinyasa
3	class03	Advanced Aerobics	for the experienced
4	class04	Intro to Kickboxing	basic, for beginners
5	class05	Smooth Zumba	slow, basic
6	class06	Hot Zumba	fast-paced, upbeat

Table: Memberships

Explanation:

This table describes the levels of memberships the Health Club offers. This includes the name of the membership, the description of the membership (what sets it apart from the rest of the memberships), and the price per month in USD of the membership. Each type of membership has a member _ID.

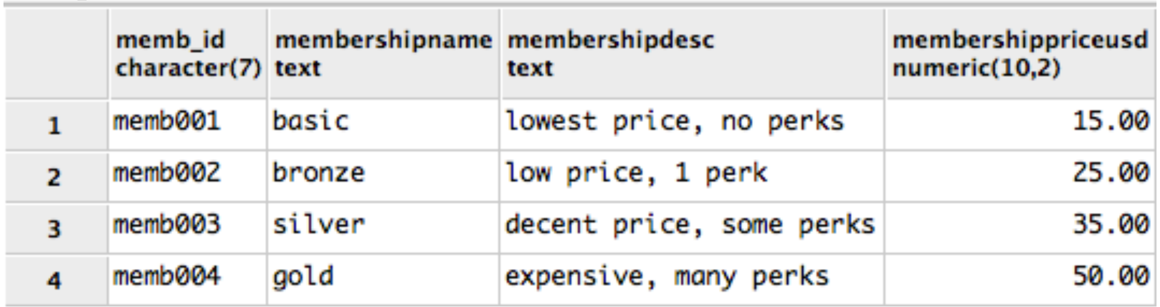
Create Statements:

```
CREATE TABLE Memberships(
    memb_id CHAR(7) NOT NULL,
    membershipName TEXT NOT NULL,
    membershipDesc TEXT NOT NULL,
    membershipPriceUSD NUMERIC(10,2),
 PRIMARY KEY (memb_id)
);
```

Functional Dependencies:

memb_id → membershipName, membershipDesc, membershipPriceUSD

Sample Data:



	memb_id character(7)	membershipname text	membershipdesc text	membershippriceusd numeric(10,2)
1	memb001	basic	lowest price, no perks	15.00
2	memb002	bronze	low price, 1 perk	25.00
3	memb003	silver	decent price, some perks	35.00
4	memb004	gold	expensive, many perks	50.00

Table: Time Blocks

Explanation:

The Time Blocks table is the table that shows the time slots that classes are offered. It is set up as time blocks because of future matrix scheduling. There is a block_ID which corresponds with a time slot, all of which are an hour and a half. These are the times fitness classes will be offered at the Health Club, as well as how long they will last.

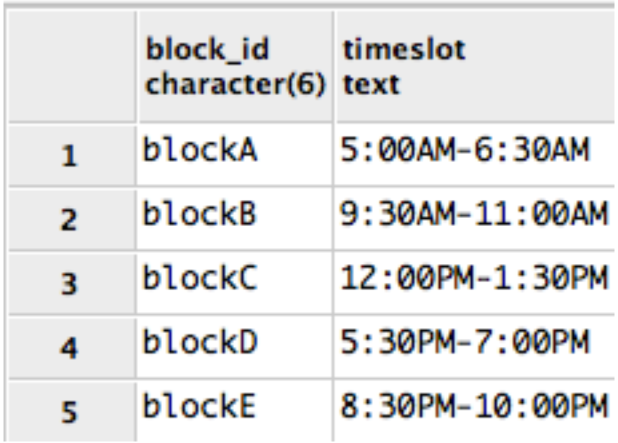
Create Statements:

```
CREATE TABLE TimeBlocks(
    block_id CHAR(6) NOT NULL,
    timeSlot text NOT NULL,
PRIMARY KEY (block_id)
);
```

Functional Dependencies:

block_id → timeSlot

Sample Data:



	block_id character(6)	timeslot text
1	blockA	5:00AM-6:30AM
2	blockB	9:30AM-11:00AM
3	blockC	12:00PM-1:30PM
4	blockD	5:30PM-7:00PM
5	blockE	8:30PM-10:00PM

Table: Equipment

Explanation:

The equipment table lists the different type of equipment that will be in the various rooms of the Health Club ready to use for fitness classes. Each type of equipment has an equipment ID, description, and warranty. The warranty is an important piece of information included in the database to help owners of the Health Club know when and if the warranty of the piece of equipment is up, in the event that something happens to the equipment.

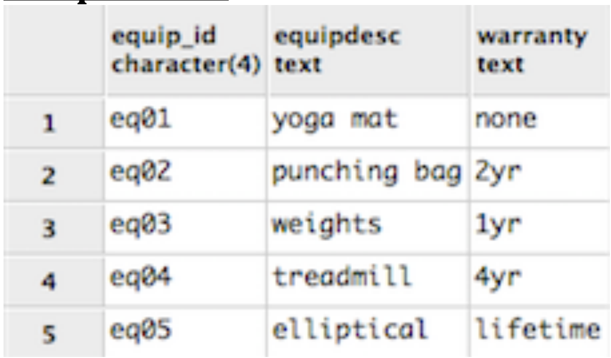
Create Statements:

```
CREATE TABLE Equipment(
    equip_id CHAR(4) NOT NULL,
    equipDesc TEXT NOT NULL,
    warranty TEXT NOT NULL,
PRIMARY KEY (equip_id)
);
```

Functional Dependencies:

Equip_id → equipDesc, warranty

Sample Data:



	equip_id character(4)	equipdesc text	warranty text
1	eq01	yoga mat	none
2	eq02	punching bag	2yr
3	eq03	weights	1yr
4	eq04	treadmill	4yr
5	eq05	elliptical	lifetime

Table: Certifications

Explanation:

The certifications table lists the types of certifications that many of the instructors and employees at the Health Club have. There is a certification ID, certification name, and a description of the certification, usually telling the hours or years of certification. As a business, the Health Club wants to offer the best services to its customers, which includes employing the best of the best instructors. The certifications for each instructor is one way the Health Club can ensure that they are providing the best services to their customers.

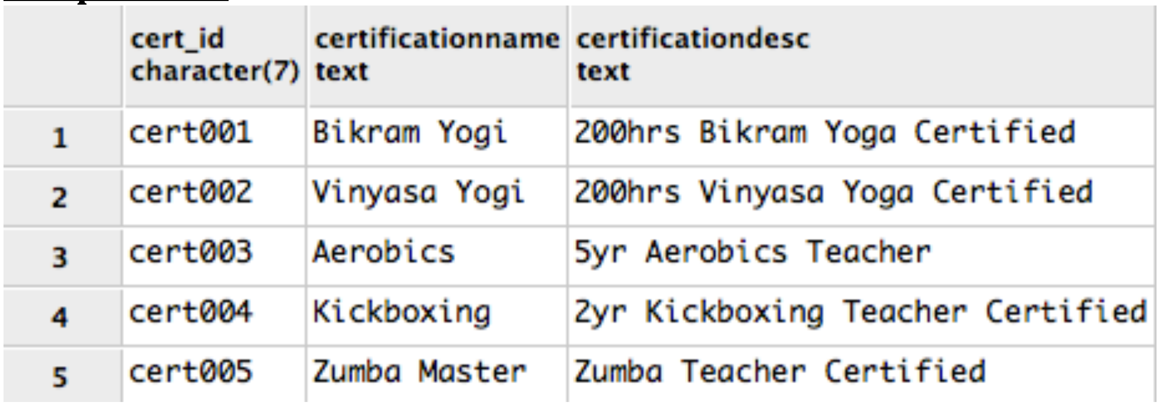
Create Statements:

```
CREATE TABLE Certifications(
    cert_id CHAR(7) NOT NULL,
    certificationName TEXT NOT NULL,
    certificationDesc TEXT NOT NULL,
 PRIMARY KEY (cert_id)
);
```

Functional Dependencies:

Cert_id → certificationName, certificationDesc

Sample Data:



	cert_id character(7)	certificationname text	certificationdesc text
1	cert001	Bikram Yogi	200hrs Bikram Yoga Certified
2	cert002	Vinyasa Yogi	200hrs Vinyasa Yoga Certified
3	cert003	Aerobics	5yr Aerobics Teacher
4	cert004	Kickboxing	2yr Kickboxing Teacher Certified
5	cert005	Zumba Master	Zumba Teacher Certified

Table: Rooms

Explanation:

This table lists the rooms available for classes in the Health Club, as well as information on the type of equipment that will be in each room. The room ID is unique for every room. This table also lists the “official” names of the rooms, too. By having the equipment ID in this table, it will be easier for the employee that books the fitness classes in the rooms to know what fitness classes should take place in what room.

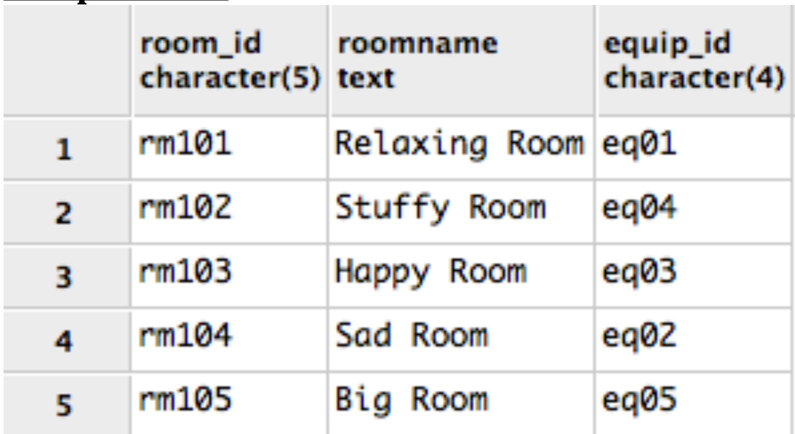
Create Statements:

```
CREATE TABLE Rooms(
    room_id CHAR(5) NOT NULL,
    roomName TEXT NOT NULL,
    equip_id CHAR(4) NOT NULL REFERENCES
Equipment(equip_id),
 PRIMARY KEY (room_id),
 FOREIGN KEY (equip_id) REFERENCES Equipment(equip_id)
);
```

Functional Dependencies:

Room_id → roomName, equip_id

Sample Data:



	room_id character(5)	roomname text	equip_id character(4)
1	rm101	Relaxing Room	eq01
2	rm102	Stuffy Room	eq04
3	rm103	Happy Room	eq03
4	rm104	Sad Room	eq02
5	rm105	Big Room	eq05

Table: Schedule

Explanation:

The schedule table lists what classes will take place in what rooms at what times (during what time block). The primary key for this table is room ID and block ID. We need both of these as primary keys because different rooms could be occupied at the same time, and one room could be occupied many times throughout the day. Both the room ID and the block ID determine the class ID. This serves as an associative table to connect the Rooms, Time Block and Classes Offered tables.

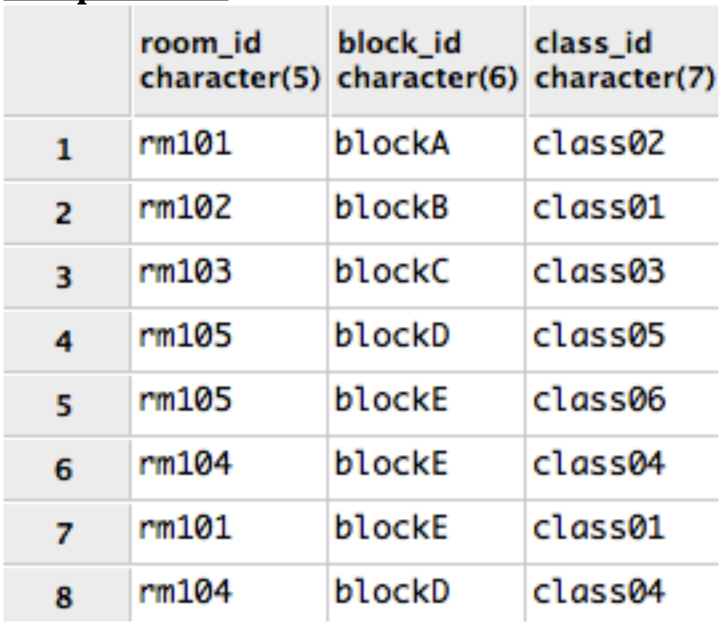
Create Statements:

```
CREATE TABLE Schedule(
   room_id CHAR(5) NOT NULL REFERENCES Rooms(room_id),
    block_id CHAR(6) NOT NULL REFERENCES TimeBlocks(block_id),
    class_id CHAR(7) NOT NULL REFERENCES ClassesOffered(class_id),
 PRIMARY KEY (room_id, block_id),
 FOREIGN KEY (room_id) REFERENCES Rooms(room_id),
 FOREIGN KEY (block_id) REFERENCES TimeBlocks(block_id),
 FOREIGN KEY (class_id) REFERENCES ClassesOffered(class_id)
```

Functional Dependencies:

Room_id, block_id → class_id

Sample Data:



	room_id character(5)	block_id character(6)	class_id character(7)
1	rm101	blockA	class02
2	rm102	blockB	class01
3	rm103	blockC	class03
4	rm105	blockD	class05
5	rm105	blockE	class06
6	rm104	blockE	class04
7	rm101	blockE	class01
8	rm104	blockD	class04

Table: Club Customers

Explanation:

The Club Customers table is the table that shows all of the customers of the Boyce Codd Health Club. This table mimics a customers profile as it holds all of the customers personal information, such as their name, date of birth, contact information, and much more. Each customer of the club gets their own customer ID, so they are easily identifiable throughout the Health Club's database without giving away personal information.

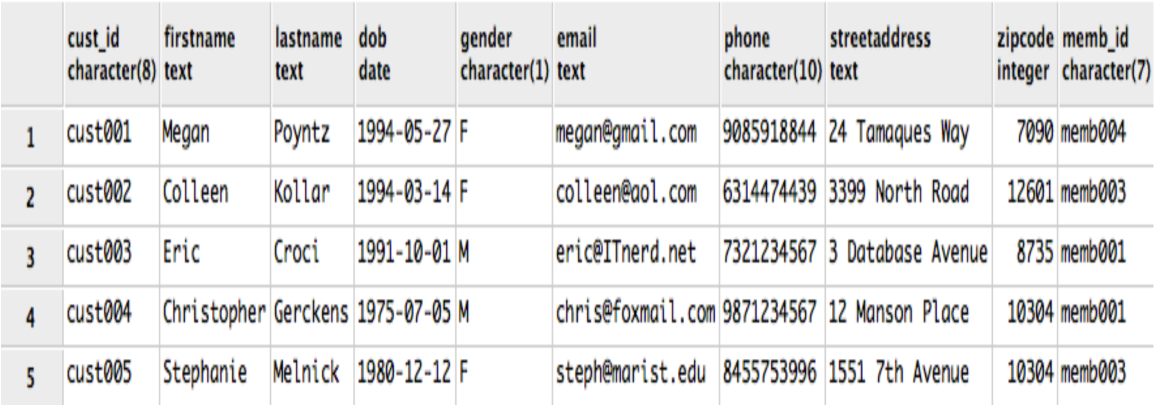
Create Statements:

```
CREATE TABLE ClubCustomers(
    cust_id CHAR(8) NOT NULL,
    firstName TEXT NOT NULL,
    lastName TEXT NOT NULL,
    DOB DATE NOT NULL,
    gender CHAR(1) NOT NULL,
    email TEXT NOT NULL,
    phone CHAR(10) NOT NULL,
    streetAddress TEXT NOT NULL,
    zipCode INT NOT NULL REFERENCES ZipCodes(zipCode),
    memb_id CHAR(7) NOT NULL REFERENCES Memberships(memb_id),
 PRIMARY KEY (cust_id),
 FOREIGN KEY (zipCode) REFERENCES ZipCodes(zipCode),
 FOREIGN KEY (memb_id) REFERENCES Memberships(memb_id)
);
```

Functional Dependencies:

Cust_id → firstName, lastName, DOB, gender, email, phone, streetAddress, zipCode, memb_id

Sample Data:



	cust_id character(8)	firstname text	lastname text	dob date	gender character(1)	email text	phone character(10)	streetaddress text	zipcode integer	memb_id character(7)
1	cust001	Megan	Poyntz	1994-05-27	F	megan@gmail.com	9085918844	24 Tamaques Way	7090	memb004
2	cust002	Colleen	Kollar	1994-03-14	F	colleen@aol.com	6314474439	3399 North Road	12601	memb003
3	cust003	Eric	Croci	1991-10-01	M	eric@ITnerd.net	7321234567	3 Database Avenue	8735	memb001
4	cust004	Christopher	Gerckens	1975-07-05	M	chris@foxmail.com	9871234567	12 Manson Place	10304	memb001
5	cust005	Stephanie	Melnick	1980-12-12	F	steph@marist.edu	8455753996	1551 7th Avenue	10304	memb003

Table: Offerings

Explanation:

This table goes into more specifics about what classes will be offered at what time of year. It lists the classes (through class ID), the month it will be offered, the year it will be offered, and the block ID, which lists the time of day the fitness class will take place. Each "offering" gets their own offering ID so that it is used once as the Health Club differentiates the class offerings at times of the year for different years.

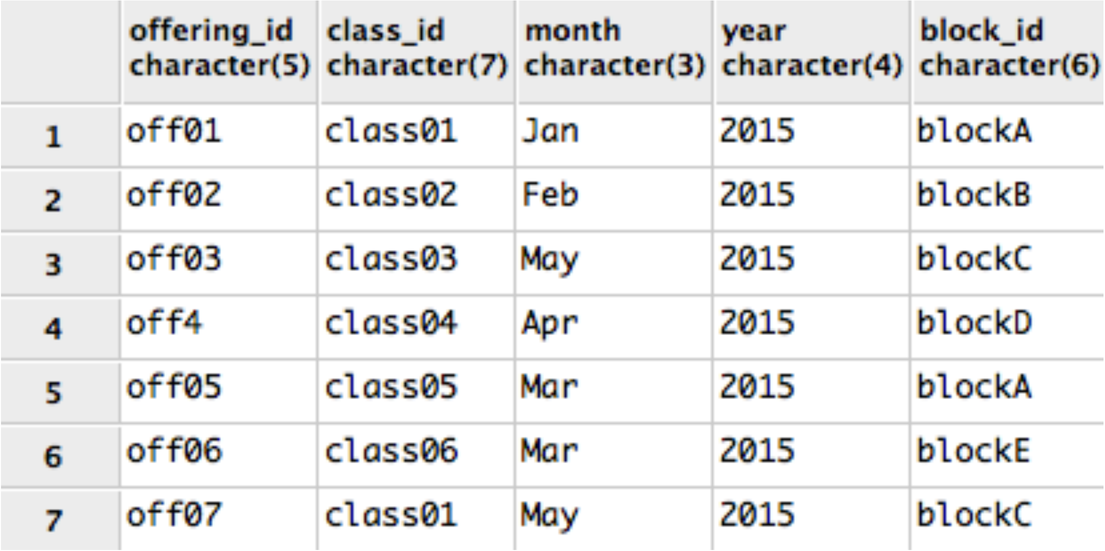
Create Statements:

```
CREATE TABLE Offerings(
    offering_id CHAR(5) NOT NULL,
    class_id CHAR(7) NOT NULL REFERENCES ClassesOffered(class_id),
    month CHAR(3) NOT NULL,
    year CHAR(4) NOT NULL,
    block_id CHAR(6) NOT NULL REFERENCES TimeBlocks(block_id),
 PRIMARY KEY (offering_id),
 FOREIGN KEY (class_id) REFERENCES ClassesOffered(class_id),
 FOREIGN KEY (block_id) REFERENCES TimeBlocks(block_id)
);
```

Functional Dependencies:

Offering_id → class_id, month, year, block_id

Sample Data:



	offering_id character(5)	class_id character(7)	month character(3)	year character(4)	block_id character(6)
1	off01	class01	Jan	2015	blockA
2	off02	class02	Feb	2015	blockB
3	off03	class03	May	2015	blockC
4	off4	class04	Apr	2015	blockD
5	off05	class05	Mar	2015	blockA
6	off06	class06	Mar	2015	blockE
7	off07	class01	May	2015	blockC

Table: Registration

Explanation:

The registration table shows what customers showed up for what classes. This table serves as an associative table between club customers and offerings. Because the offerings table lists specifically what classes will be held at what times during a specific time of year in a specific year, the customers can sign up for as many classes as they want. The information will not repeat.

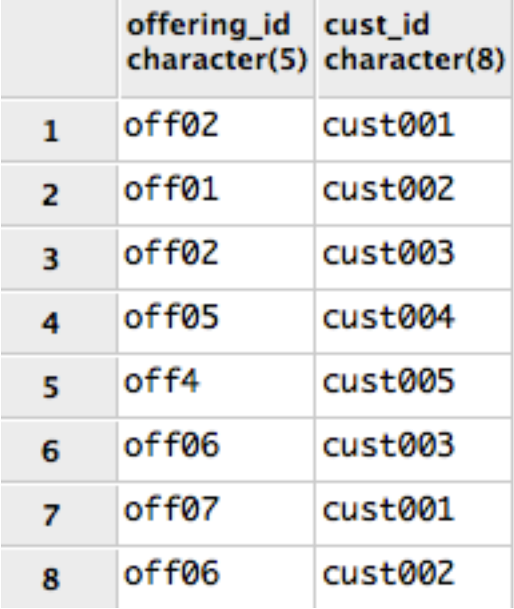
Create Statements:

```
CREATE TABLE Registration(
    offering_id CHAR(5) NOT NULL REFERENCES Offerings(offering_id),
    cust_id CHAR(8) NOT NULL REFERENCES ClubCustomers(cust_id),
 PRIMARY KEY (offering_id, cust_id),
 FOREIGN KEY (offering_id) REFERENCES Offerings(offering_id),
 FOREIGN KEY (cust_id) REFERENCES ClubCustomers(cust_id)
);
```

Functional Dependencies:

None

Sample Data:



	offering_id character(5)	cust_id character(8)
1	off02	cust001
2	off01	cust002
3	off02	cust003
4	off05	cust004
5	off4	cust005
6	off06	cust003
7	off07	cust001
8	off06	cust002

Table: Employees

Explanation:

The employees table is much like the club customers table. This table lists all the personal information of the employees of Boyce Codd Health Club, from names and birth dates to contact information and addresses. Just as we gave club customers a customer ID so their information would be protected, the employees get an employee ID. The employee ID serves as the employees' version of security as they will most likely be displayed across the database many times. For example, a fitness class instructor will teach many classes at many times throughout the year, so to preserve their personal information, their employee ID will show up throughout the database.

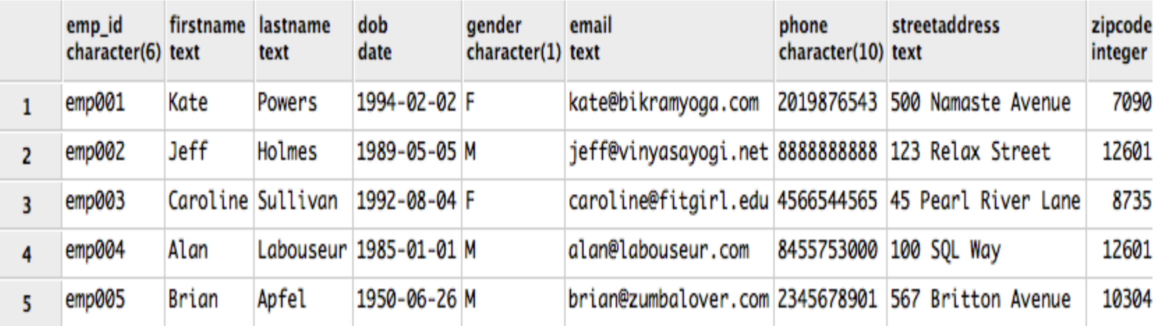
Create Statements:

```
CREATE TABLE Employees(
    emp_id CHAR(6) NOT NULL,
    firstName TEXT NOT NULL,
    lastName TEXT NOT NULL,
    DOB DATE NOT NULL,
    gender CHAR(1) NOT NULL,
    email TEXT NOT NULL,
    phone CHAR(10) NOT NULL,
    streetAddress TEXT NOT NULL,
    zipCode INT NOT NULL REFERENCES ZipCodes(zipCode),
 PRIMARY KEY (emp_id),
 FOREIGN KEY (zipCode) REFERENCES ZipCodes(zipCode)
);
```

Functional Dependencies:

Emp_id → firstName, lastName, DOB, gender, email, phone, streetAddress, zipCode

Sample Data:



	emp_id character(6)	firstname text	lastname text	dob date	gender character(1)	email text	phone character(10)	streetaddress text	zipcode integer
1	emp001	Kate	Powers	1994-02-02	F	kate@bikramyoga.com	2019876543	500 Namaste Avenue	7090
2	emp002	Jeff	Holmes	1989-05-05	M	jeff@vinyasayogi.net	8888888888	123 Relax Street	12601
3	emp003	Caroline	Sullivan	1992-08-04	F	caroline@fitgirl.edu	4566544565	45 Pearl River Lane	8735
4	emp004	Alan	Labouseur	1985-01-01	M	alan@labouseur.com	8455753000	100 SQL Way	12601
5	emp005	Brian	Apfel	1950-06-26	M	brian@zumbalover.com	2345678901	567 Britton Avenue	10304

Table: Employee Certification Catalog

Explanation:

This table links together the employees and the certifications table. It shows what employees have what certifications. Because many employees can have many different types of certifications, it needed to be set up as an associative table. This table also contains the date that the certification was attained by the employee. This will help the Health Club's managers and customers know how up-to-date the instructors are, and whether or not they should be going for a re-certificaiton.

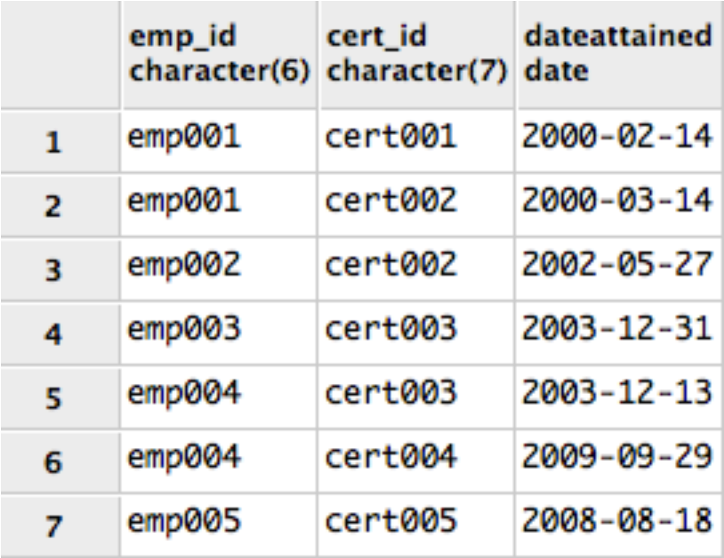
Create Statements:

```
CREATE TABLE EmployeeCertificationCatalog(
    emp_id CHAR(6) NOT NULL REFERENCES Employees(emp_id),
    cert_id CHAR(7) NOT NULL REFERENCES Certifications(cert_id),
    dateAttained DATE NOT NULL, PRIMARY KEY (emp_id, cert_id),
 FOREIGN KEY (emp_id) REFERENCES Employees(emp_id),
 FOREIGN KEY (cert_id) REFERENCES Certifications(cert_id)
);
```

Functional Dependencies:

Emp_id, cert_id → dateAttained

Sample Data:



	emp_id character(6)	cert_id character(7)	dateattained date
1	emp001	cert001	2000-02-14
2	emp001	cert002	2000-03-14
3	emp002	cert002	2002-05-27
4	emp003	cert003	2003-12-31
5	emp004	cert003	2003-12-13
6	emp004	cert004	2009-09-29
7	emp005	cert005	2008-08-18

Table: Employee Assignments

Explanation:

The employee assignments table shows what employees will be teaching what classes for the determine month and year. Rather than having a large table that lists all the classes and all the employees, with all of the times and all of the customers registered, this associative table separates the data to help it stay consistent.

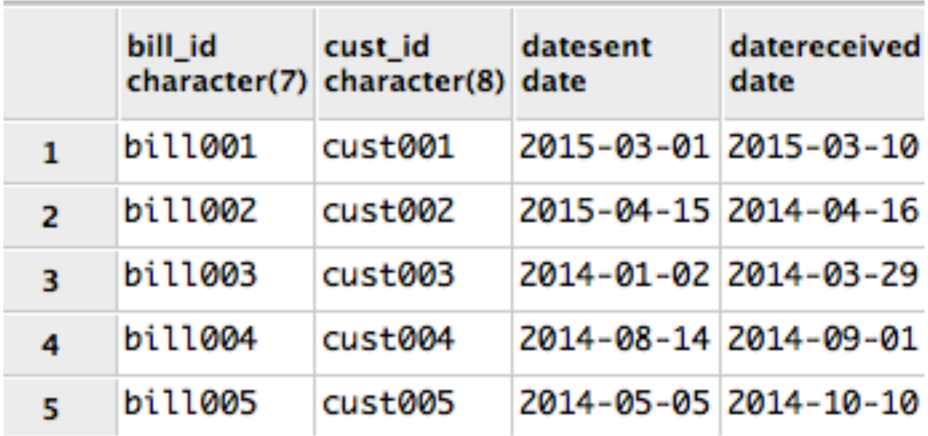
Create Statements:

```
CREATE TABLE EmployeeAssignments(
    emp_id CHAR(6) NOT NULL REFERENCES Employees(emp_id),
    offering_id CHAR(5) NOT NULL REFERENCES
Offerings(offering_id),
 PRIMARY KEY (emp_id, offering_id),
 FOREIGN KEY (emp_id) REFERENCES Employees(emp_id),
 FOREIGN KEY (offering_id) REFERENCES Offerings(offering_id)
);
```

Functional Dependencies:

None

Sample Data:



	bill_id character(7)	cust_id character(8)	datesent date	datereceived date
1	bill001	cust001	2015-03-01	2015-03-10
2	bill002	cust002	2015-04-15	2014-04-16
3	bill003	cust003	2014-01-02	2014-03-29
4	bill004	cust004	2014-08-14	2014-09-01
5	bill005	cust005	2014-05-05	2014-10-10

Table: Billing

Explanation:

As with every business, customers need to get billed for the services that the company provides. The billing table is the way that Boyce Codd Health Club will keep track of what customers have been billed. This table includes a billing ID, so that employees can get a bill more than once (since they will most likely be using the Health Club's services for more than one billing period). It also includes the customer it was sent to (through customer ID), and the date the bill was sent and then received.

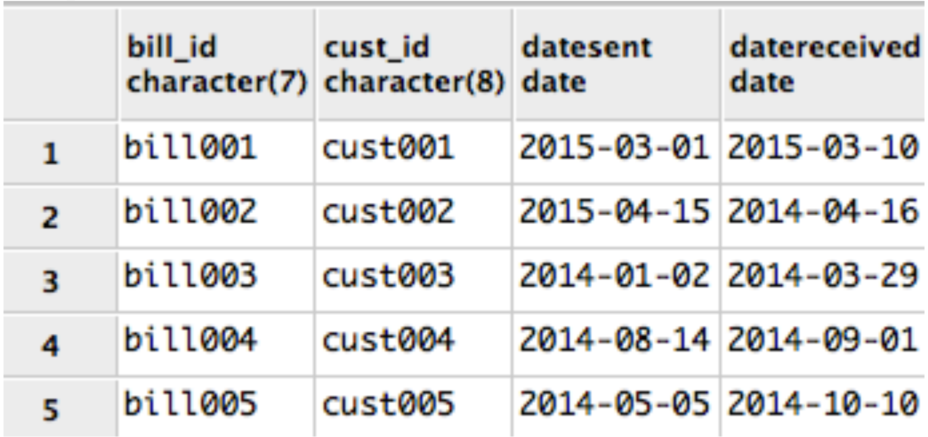
Create Statements:

```
CREATE TABLE Billing(
    bill_id CHAR(7) NOT NULL,
    cust_id CHAR(8) NOT NULL REFERENCES ClubCustomers(cust_id),
    dateSent DATE NOT NULL,
    dateReceived DATE NOT NULL,
 PRIMARY KEY (bill_id),
 FOREIGN KEY (cust_id) REFERENCES ClubCustomers(cust_id)
);
```

Functional Dependencies:

Bill_id → cust_id, dateSent, dateReceived

Sample Data:



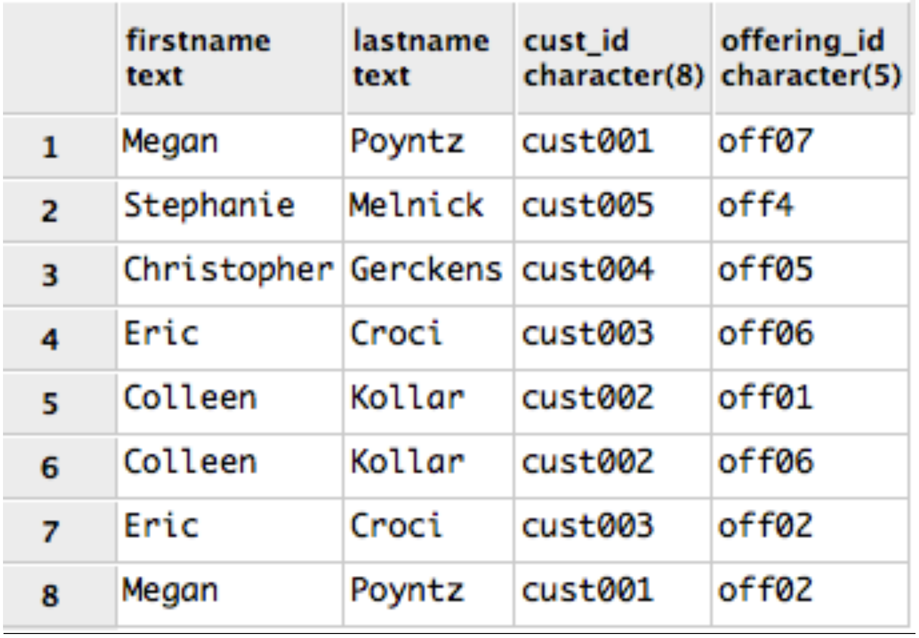
	bill_id character(7)	cust_id character(8)	datesent date	datereceived date
1	bill001	cust001	2015-03-01	2015-03-10
2	bill002	cust002	2015-04-15	2014-04-16
3	bill003	cust003	2014-01-02	2014-03-29
4	bill004	cust004	2014-08-14	2014-09-01
5	bill005	cust005	2014-05-05	2014-10-10

Views

What Club Customers are registered for what offering(class):

```
CREATE VIEW CustomerLookUp AS
SELECT DISTINCT
    c.firstName,
    c.lastName,
    r.cust_id,
    r.offering_id
FROM
    ClubCustomers c,
    Registration r
WHERE c.cust_id = r.cust_id
```

Sample Data:



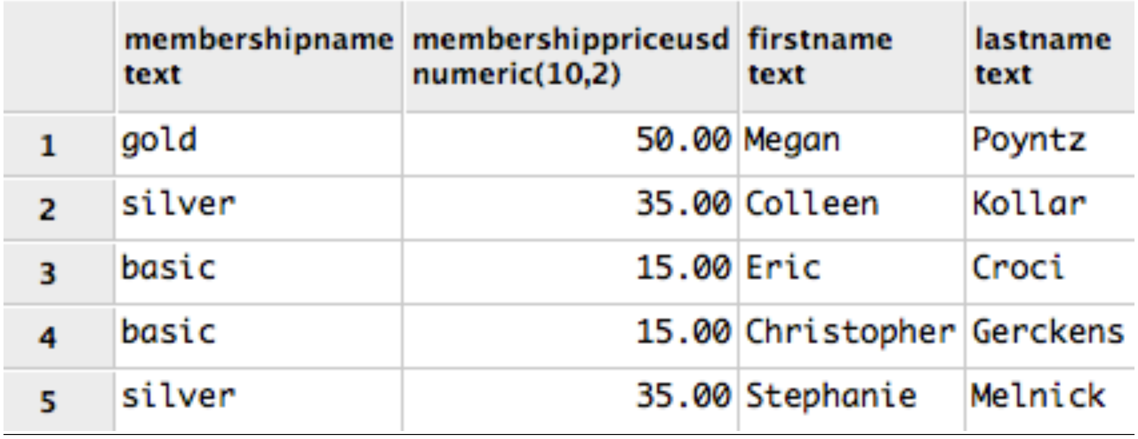
	firstname text	lastname text	cust_id character(8)	offering_id character(5)
1	Megan	Poyntz	cust001	off07
2	Stephanie	Melnick	cust005	off4
3	Christopher	Gerckens	cust004	off05
4	Eric	Croci	cust003	off06
5	Colleen	Kollar	cust002	off01
6	Colleen	Kollar	cust002	off06
7	Eric	Croci	cust003	off02
8	Megan	Poyntz	cust001	off02

Views

What Club Customers have what type of memberships and what they pay for that membership:

```
CREATE VIEW CustMemberships AS
SELECT
    m.membershipName,
    m.membershipPriceUSD,
    c.firstName,
    c.lastName
FROM
    Memberships m,
        ClubCustomers c
WHERE m.memb_id = c.memb_id
```

Sample Data:



	membershipname text	membershippriceusd numeric(10,2)	firstname text	lastname text
1	gold	50.00	Megan	Poyntz
2	silver	35.00	Colleen	Kollar
3	basic	15.00	Eric	Croci
4	basic	15.00	Christopher	Gerckens
5	silver	35.00	Stephanie	Melnick

Query

```
SELECT distinct c.cust_id, c.firstName, c.lastName,
reg.cust_id, reg.offering_id
FROM ClubCustomers c, Registration reg
WHERE c.cust_id = reg.cust_id
```

This query displays each customer and what offerings they are taking (what class they are taking in a certain month, year, time slot).

Stored Procedures

```
CREATE OR REPLACE FUNCTION membMthlyPriceUSD() returns
trigger as $$
BEGIN
IF (Memberships.membershipPriceUSD is NULL)
FROM Memberships
WHERE Memberships.membershipPriceUSD is NULL
THEN
UPDATE Memberships SET amount = 0.00 WHERE amount is NULL;
END if;
Return new;
END
$$LANGUAGE plpgsql;
```

The purpose of this stored procedure is to update the membership price in the Memberships table is NULL exists for any monthly price amount. It will be NULL in the off chance that a Club Customer's membership information is inputted incorrectly or not inputted at all. This stored procedure will update the membership price to $0.00 so the management of Boyce Codd Health Club will be able to clearly see that a mistake was made in the process of putting in the membership information.

Triggers

```
CREATE TRIGGER NULLfix
AFTER INSERT OR UPDATE
ON Memberships
FOR EACH ROW EXECUTE
PROCEDURE
membMthlyPriceUSD();
```

The purpose of this trigger is to fix the NULLs that exists in the membershipPriceUSD in the Memberships table. When this trigger is executed, is runs the membMthlyPriceUSD() procedure.

Security

```
CREATE ROLE Admin
REVOKE ALL ON Memberships  FROM Admin;
REVOKE ALL ON ClassesOffered     FROM Admin;
REVOKE ALL ON Offerings    FROM Admin;
REVOKE ALL ON Schedule     FROM Admin;
REVOKE ALL ON Rooms   FROM Admin;
REVOKE ALL ON Customers    FROM Admin;
REVOKE ALL ON EmployeeAssignments     FROM Admin;
REVOKE ALL ON Registration FROM Admin;
REVOKE ALL ON Equipment    FROM Admin;
REVOKE ALL ON TimeBlocks   FROM Admin;
REVOKE ALL ON Billing FROM Admin; REVOKE ALL ON ZipCodes    FROM
Admin; REVOKE ALL ON Employees   FROM Admin;
REVOKE ALL ON EmployeeCertificationCatalog FROM Admin;
REVOKE ALL ON Certifications     FROM Admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON Memberships   TO ADMIN;
GRANT SELECT, INSERT, UPDATE, DELETE ON ClassesOffered TO ADMIN;
GRANT SELECT, INSERT, UPDATE, DELETE ON OfferingsTO ADMIN;
GRANT SELECT, INSERT, UPDATE, DELETE ON Schedule TO ADMIN;
GRANT SELECT, INSERT, UPDATE, DELETE ON Rooms    TO ADMIN;
GRANT SELECT, INSERT, UPDATE, DELETE ON CustomersTO ADMIN;
GRANT SELECT, INSERT, UPDATE, DELETE ON EmployeeAssignments TO
ADMIN;
GRANT SELECT, INSERT, UPDATE, DELETE ON Registration   TO ADMIN;
GRANT SELECT, INSERT, UPDATE, DELETE ON EquipmentTO ADMIN;
GRANT SELECT, INSERT, UPDATE, DELETE ON TimeBlocks    TO ADMIN;
GRANT SELECT, INSERT, UPDATE, DELETE ON Billing  TO ADMIN;
GRANT SELECT, INSERT, UPDATE, DELETE ON ZipCodes TO ADMIN;
GRANT SELECT, INSERT, UPDATE, DELETE ON EmployeesTO ADMIN;
GRANT SELECT, INSERT, UPDATE, DELETE ON
EmployeeCertificationCatalog     TO ADMIN;
GRANT SELECT, INSERT, UPDATE, DELETE ON Certifications TO ADMIN;
```

This security role grants full access to the Admin on Boyce Codd Health Club's database. The admin can select, insert data, update data, or delete data on any of the tables within the database. This is the highest level of security access. Due to preserving the Health Club's data, the Admin-level access on the Boyce Codd Health Club Database should be limited to one or two people. The more peope that get the Admin-level access, the more likely it is for there to be inconsistency in the data.

Security

```
CREATE ROLE emp001
REVOKE ALL ON Memberships  FROM emp001;
REVOKE ALL ON ClassesOffered     FROM emp001;
REVOKE ALL ON Offerings    FROM emp001;
REVOKE ALL ON Schedule     FROM emp001;
REVOKE ALL ON Rooms   FROM emp001;
REVOKE ALL ON Customers    FROM emp001;
REVOKE ALL ON EmployeeAssignments     FROM emp001;
REVOKE ALL ON Registration FROM emp001;
REVOKE ALL ON Equipment    FROM emp001;
REVOKE ALL ON TimeBlocks   FROM emp001;
REVOKE ALL ON Billing FROM emp001;
REVOKE ALL ON ZipCodes     FROM emp001;
REVOKE ALL ON Employees    FROM emp001;
REVOKE ALL ON EmployeeCertificationCatalog  FROM emp001;
REVOKE ALL ON Certifications     FROM emp001;

GRANT SELECT ON ClassesOffered   TO emp001;
GRANT SELECT ON Offerings  TO emp001;
GRANT SELECT ON Schedule   TO emp001;
GRANT SELECT ON Rooms TO emp001;
GRANT SELECT, INSERT, UPDATE, DELETE ON EmployeeAssignments TO
emp001;
GRANT SELECT ON Registration     TO emp001;
GRANT SELECT, INSERT, UPDATE ON Equipment   TO emp001;
GRANT SELECT ON TimeBlocks TO emp001;
GRANT SELECT, UPDATE ON Employees     TO emp001;
GRANT SELECT, INSERT, UPDATE ON EmployeeCertificationCatalog TO
emp001;
GRANT SELECT ON Certifications   TO emp001;
```

This level security access is for employees, with the given example of emp001. In an employee-level security access, employees can select and view Classes Offered, Offerings, Schedule, Rooms, Registration, Time Blocks, and Certifications. They can also insert, update, and delete on the Employee Assignments table, giving them access to change around their schedule if, for example, two employees wanted to switch classes, or one employee needs to cover for another employee. The employees can also insert new equipment they may have brought or require for the class, as well as update whether or not the equipment is running low or needs to be checked out. Employees can view their personal information, as well as update their personal information. Lastly, employees can add the certifications they received in the Employee Certification Catalog table.

Implementation Notes

This database design proposal for Boyce Codd Health Club scratches the surface of what the database could be like in reality. There are separate tables for employees and customers to make linking each table and the type of person it represents to their prospective duty (e.g. customers take classes offered, employees have employee assignments, etc.). This could also be done as a "persons" table with entity subtypes, such as employees and customers.

Additionally, there are many tables for the overall fitness class schedule process. While these could be combined into a larger table, there are too many relationships that need associative tables, as there are a large amount of many-to-many relationships.

As for the Time Block table, the blocks will help with future matrix scheduling. These classes are going to be offered monthly, and keep repeating. Rather than putting the time and day of the classes, block scheduling helps with rearranging classes on different days throughout the week, month and year.

The database makes the assumption that customers are members of the Health Club for the classes. There is no table to figure out what customers are members of the health club for other reasons, such as general equipment use.

Known Problems

There are many, many more specific types of employees. Not all employees teach classes, as implied by the database for the sake of consistency. There are receptionists, management, custodians, etc. that should and could be added to the database. This database is also very present and future focused as far as time goes. There is no table listing a Club customer's history (how long they have been part of the Health Club, what classes they have taken, etc.). In addition, not all customers join a health club for the classes. They usually join the health club to solely use the machines, which would not require any real registration at all.

Future Enhancements

All future enhancements include the addition of different tables and services. In the future, Boyce Codd Health Club can grow by adding tables and services for:

- Annual member discounts (the longer you stay with Boyce Codd Health Club, the bigger discount you get on your monthly bill)
- A Club Customer reward system (allows the Club's customers to earn points for every class they take, and redeem them in the form of a free class)
- Employee salary payment (this could be in the form of a base salary or hourly salary, depending on the type of employee and the tasks they perform)
- Allowing employees to also be members. For example, an individual can teach a class and attend classes at the same time
- As stated in the implementation notes, there could be a "people" table with entity subtypes (employees, customers, staff, management, etc.)
- General gym use. More often than not, customers join a gym to use the equipment at their leisure, and could go their whole gym membership without ever taking a class
- Past customer history. This could aid in implementing a rewards system. There could be a section of the database that shows how long a customer has been a member of the gym, what classes they attended, etc. Not only does this help with tracking customer history, but this could also help with the business' data analytics. By tracking past customer involvement, the Health Club can see what classes were post popular, what fitness instructors were most popular, and then alter processes and strategies to past findings and analyses