Amackassin Swim and Tennis Club

By: Victoria Confeiteiro

Table of Contents

Executive Summary

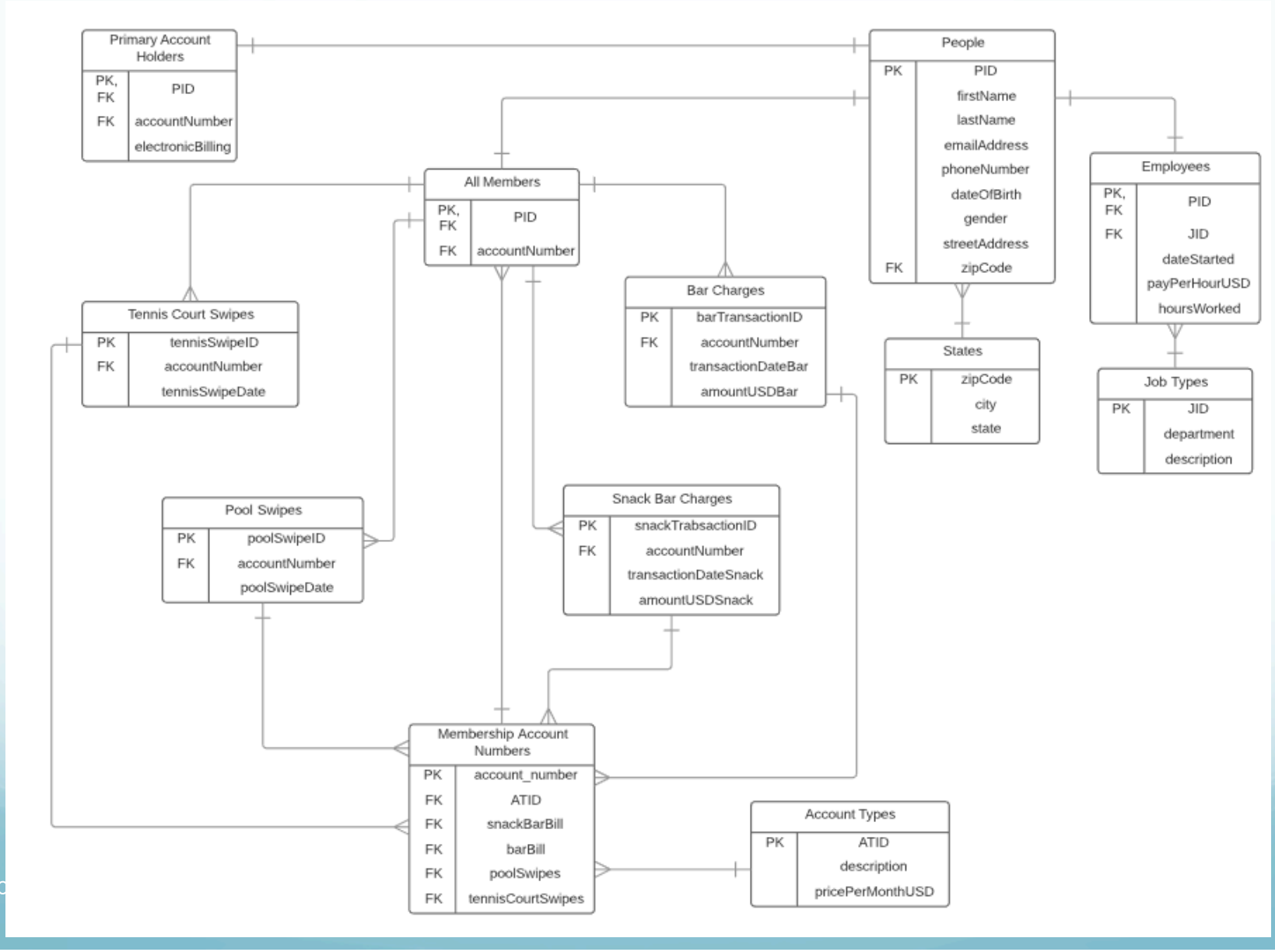
The Amackassin Club is a swim and tennis club located in Yonkers, New York. The club is members-only and every member is given a unique account number that they must give in order to purchase items at either the snack bar or bar.

In addition to keeping track of members, the Amackassin also has a large staff which must be included in the database. Many members are also employees which needs to be accounted for.

The most important aspects of this database is keeping track of member's monthly bills and employee payroll. Employees are paid by the hour, which is logged using a time card which automatically updates the database. Members are billed monthly, the bill includes the monthly membership fee and then snack bar or bar charges.

The Amackassin also needs to keep track of the amount of times members use either the pool or tennis courts. Each member is issued a membership card that they must swipe in order to use either facility.

ER Diagram


Primary Account Holders
PK, FK PID
FK accountNumber
electronicBilling
People
PK PID
firstName
lastName
emailAddress
phoneNumber
dateOfBirth
gender
streetAddress
FK zipCode
Employees
PK, FK PID
FK JID
dateStarted
payPerHourUSD
hoursWorked
All Members
PK, FK PID
FK accountNumber
Tennis Court Swipes
PK tennisSwipeID
FK accountNumber
tennisSwipeDate
Bar Charges
PK barTransactionID
FK accountNumber
transactionDateBar
amountUSDBar
States
PK zipCode
city
state
Job Types
PK JID
department
description
Pool Swipes
PK poolSwipeID
FK accountNumber
poolSwipeDate
Snack Bar Charges
PK snackTrabsactionID
FK accountNumber
transactionDateSnack
amountUSDSnack
Membership Account Numbers
PK account_number
FK ATID
FK snackBarBill
FK barBill
FK poolSwipes
FK tennisCourtSwipes
Account Types
PK ATID
description
pricePerMonthUSD

Tables

People

This table is used to keep track of everyone that either works for, or is a member of the club along with their basic information.

```
CREATE TABLE People (
        PID serial,
        firstName text not null,
        lastName text not null,
        emailAddress text,
        phoneNumber bigint,
        dateOfBirth date,
        gender text not null,
        streetAddress text,
        zipCode int not null,
        primary key(PID)
);
```

Functional Dependencies

pid -> firstName, lastName, emailAddress, phoneNumber, dateOfBirth, gender, streetAddress, zipCode

People Table

	pid integer	firstname text	lastname text	emailaddress text	phonenumber bigint	dateofbirth date	gender text	streetaddress text	zipcode integer
1	1	Victoria	Confeiteiro	vconfeiteiro@optonline.net	9149241907	1996-11-29	Female	12 Tree Street	10701
2	2	Thomas	Duff	trogs@gmail.coom	5163456789	1960-05-14	Male	57 Chester Drive	11554
3	3	Alan	Labouseur	alan.labouseur.marist.edu	8455753000	1988-06-13	Male	3399 North Road	12704
4	4	Miceala	Martini	makemeamartini@gmail.com	5186451234	1995-10-02	Female	5 South Beach Road	33018
5	5	Julie	Martinelli	jmart3@gmail.com	9148901234	1996-10-02	Female	8 North Avenue	13742
6	6	James	Duff	duffman11@gmail.com	6469176453	1970-05-02	Male	52 Chester Drive	11554
7	7	Kerri	Waters	kerri.waters@amackassin.com	9145879012	1970-05-10	Female	34 Palisade Road	10705
8	8	Beyonce	Knowles	queenbey@hotmail.com	3478913452	1989-03-02	Female	5 Park Avenue	10001
9	9	Jay-Z	Carter	jayz@hotmail.com	3478913453	1987-03-02	Male	5 Park Avenue	10001
10	10	Joseph	Confeiteiro	jconfeiteiro@optonline.net	9149241907	2000-04-11	Male	5 Corely Street	17404
11	11	Hillary	Clinton	hillary4prez@yahoo.com	9144334396	1947-10-26	Female	3 President Court	13742
12	12	Bernie	Sanders	feelthebern@optonline.net	9149241907	1941-09-08	Male	5 So Close Street	13742
13	13	Tom	Haverford	entertainment720@gmail.com	9144334396	1985-05-25	Male	89 Tom Road	10703
14	14	Leslie	Knope	pawneeluvr@gmail.com	8459608790	1982-04-11	Female	4 Pawnee Street	10701
15	15	Andy	Dwyer	mouseratrules@gmail.com	6715438890	1985-11-07	Male	127 Champion Lane	10701

Job Types

This table lists the different jobs at the club and displays their department and specific job position.

```
INSERT INTO Job_Types (Department, jobPosition)
VALUES ('Bar', 'Bartender'),
                ('Pool', 'Lifeguard'),
                ('Pool', 'Swim Team Coach'),
                ('Bar', 'Bar Back'),
                ('Snack Bar', 'Chef'),
                ('Snack Bar', 'Counter Worker'),
                ('Snack Bar', 'Snack Bar Manager'),
                ('House', 'Janitor'),
                ('Management', 'Manager'),
                ('Management', 'Assistant Manager'),
                ('Tennis', 'Court Caretaker'),
                ('Tennis', 'Coach'),
                ('Tennis', 'Assistant Coach');
```

Functional Dependencies

jid -> department, jobPosition

Job Types Results

	jid integer	department text	jobposition text
1	1	Bar	Bartender
2	2	Pool	Lifeguard
3	3	Pool	Swim Team Coach
4	4	Bar	Bar Back
5	5	Snack Bar	Chef
6	6	Snack Bar	Counter Worker
7	7	Snack Bar	Snack Bar Manager
8	8	House	Janitor
9	9	Management	Manager
10	10	Management	Assistant Manager
11	11	Tennis	Court Caretaker
12	12	Tennis	Coach
13	13	Tennis	Assistant Coach

Employees

The Employees tables stores information about each employee including their specific job, the date they started working at the club, their pay per hour, and the hours they've worked this pay period.

```
CREATE TABLE Employees (
        PID INT NOT NULL,
        JID int not null,
        dateStarted date,
        payPerHourUSD int,
        hoursWorked int,
        primary key(PID, JID)
);
```

Functional Dependencies

pid, jid -> dateStarted, payPerHourUSD, hoursWorked

Employees Table

	pid integer	jid integer	datestarted date	payperhourusd integer	hoursworked integer
1	1	9	2011-05-30	10	75
2	3	1	2010-05-25	12	100
3	4	6	2012-05-25	11	75
4	5	7	2010-07-04	22	100
5	10	2	2014-05-30	8	125
6	11	1	1996-04-14	30	300
7	18	9	1999-11-29	40	300
8	24	5	2016-06-16	20	400
9	28	8	2014-02-04	7	120
10	29	4	2005-06-11	15	150
11	30	15	2009-03-25	20	200

States

The states table stores the city, state, and zip code of everyone in the People table.

```
CREATE TABLE States (
        zipCode int not null,
        city text not null,
        state text not null,
        primary key(zipCode)
);
```

Functional Dependencies

zipCode -> city, state

States Table

	zipcode integer	city text	state text
1	10701	Yonkers	New York
2	10703	Yonkers	New York
3	10705	Yonkers	New York
4	11554	East Meadow	New York
5	12704	Poughkeepsie	New York
6	13742	New Rochelle	New York
7	10001	New York	New York
8	17404	Scranton	Pennsylvania
9	21210	Baltimore	Maryland
10	33018	Miami	Florida

Primary Account Holders

The Primary Account Holders table stores all the primary account holders for each membership account. The address and email address associated with the primary account holder is used for billing or just general communication.

```
CREATE TABLE Primary_Account_Holders (
        PID INT NOT NULL,
        accountNumber INT not null,
        electronicBilling boolean not null,
        primary key(PID)
);
```

Functional Dependencies

pid -> accountNumber, electronicBilling

Primary Account Holders Table

	pid integer	accountnumber integer	electronicbilling boolean
1	16	1123	t
2	11	1506	t
3	14	1010	t
4	25	2300	f
5	22	1190	t
6	29	1183	t
7	8	1164	t
8	2	1789	t
9	13	1122	t
10	12	1643	t

Account Types

The Account Types table stores the 4 different types of membership accounts, their description, and their monthly membership fee.

```
CREATE TABLE Account_Types (
        ATID serial,
        description text not null,
        pricePerMonthUSD int not null,
        primary key(ATID)
);
```

Functional Dependencies

ATID -> description, pricePerMonthUSD

Account Types Table

	atid integer	description text	pricepermonthusd integer
1	1	single, no kids	300
2	2	married, no kids	350
3	3	single, with kids	500
4	4	married, with kids	600

Bar Charges

This table stores all of the transaction totals, per day, for each member account. At the bar, members are allowed to run up an ongoing tab associated with their account, and at the end of the day the bartender will input the tab's total. This total is then added to the member's ongoing monthly bill.

```
CREATE TABLE Bar_Charges (
        barTransactionID serial,
        accountNumber int not null,
        transactionDateBar date not null,
        amountUSDBar int,
        primary key(barTransactionID)
);
```

Functional Dependencies

barTransactionID -> accountNumber, transactionDateBar, amountUSDBar

Bar Charges Table

	bartransactionid integer	accountnumber integer	transactiondatebar date	amountusdbar integer
1	1	1010	2017-06-01	150
2	2	1506	2017-06-01	5
3	3	1506	2017-06-02	15
4	4	2300	2017-06-03	120
5	5	1643	2017-06-03	5
6	6	1122	2017-06-05	5
7	7	1164	2017-06-05	300
8	8	1010	2017-06-05	350
9	9	2300	2017-06-06	95
10	10	1123	2017-06-09	400
11	11	1183	2017-06-12	13
12	12	1164	2017-06-12	300
13	13	1789	2017-06-15	10

Snack Bar Charges

The snack bar operates the same way as the bar regarding members charging items to their account. This table is updated at the end of every work day by the snack bar manager. This total is then added to the member's ongoing monthly bill.

```
CREATE TABLE Snack_Bar_Charges (
        snackBarTransactionID serial,
        accountNumber int not null,
        transactionDateSnack date not null,
        amountUSDSnack int,
        primary key(snackBarTransactionID)
);
```

Functional Dependencies

snackBarTransactionID -> accountNumber, transactionDateSnack, amountUSDSnack

Snack Bar Charges Table

	snackbartransactionid integer	accountnumber integer	transactiondatesnack date	amountusdsnack integer
1	1	1506	2017-06-01	5
2	2	2300	2017-06-01	200
3	3	1010	2017-06-01	300
4	4	1122	2017-06-02	200
5	5	1506	2017-06-03	5
6	6	1010	2017-06-03	300
7	7	2300	2017-06-03	200
8	8	1123	2017-06-06	150
9	9	1164	2017-06-07	30
10	10	1164	2017-06-08	5
11	11	1122	2017-06-08	300
12	12	1122	2017-06-09	500
13	13	2300	2017-06-13	500
14	14	1123	2017-06-14	195
15	15	1190	2017-06-14	230
16	16	1183	2017-06-23	50
17	17	1789	2017-06-23	400
18	18	1643	2017-06-24	5

Membership Account Numbers

This table stores each membership account number and associated information such as the account type, their bar and snack bar running totals, and the total amount of times they've swiped their membership card at the pool or tennis court upon entry.

```
CREATE TABLE Membership_Account_Numbers (
        accountNumber int not null,
        ATID int not null,
        snackBarBill int,
        barBill int,
        poolSwipes int,
        tennisCourtSwipes int,
        primary key(accountNumber)
);
```

Functional Dependencies

accountNumber -> ATID, snackBarBill, barBill, poolSwipes, tennisCourtSwipes

Membership Account Numbers Table

	accountnumber integer	atid integer	snackbarbill integer	barbill integer	poolswipes integer	tenniscourtswipes integer
1	1010	4	600	500	4	1
2	1506	1	10	20	10	4
3	2300	4	1000	215	1	2
4	1123	2	345	1245	5	7
5	1190	2	230	10	6	4
6	1183	1	200	13	3	7
7	1164	2	35	700	1	11
8	1789	3	453	10	2	6
9	1122	1	1000	5	4	3
10	1643	1	5	5	2	7

All Members

This table stores all the members and their membership account number. This is especially useful for families where there are multiple people associated with one account.

```
CREATE TABLE All_Members (
        PID INT NOT NULL,
        accountNumber INT not null,
        primary key(PID)
);
```

Functional Dependencies

pid -> accountNumber

All Members Table

	pid integer	accountnumber integer
1	2	1789
2	6	1789
3	8	1164
4	9	1164
5	11	1506
6	12	1643
7	13	1122
8	14	1010
9	15	1123
10	16	1123
11	17	1010
12	19	1010

Tennis Court Swipes

This table stores data that is created whenever a member swipes their membership card to use the tennis courts. This table includes the member's account number and the date they used the tennis court.

```
CREATE TABLE Tennis_Court_Swipes (
        tennisSwipeID serial,
        accountNumber INT not null,
        tennisSwipeDate date,
        primary key(tennisSwipeID)
);
```

Functional Dependencies

tennisSwipeID -> accountNumber, tennisSwipeDate

Tennis Court Swipes Table

	tennisswipeid integer	accountnumber integer	tennisswipedate date
1	1	1123	2017-06-01
2	2	1183	2017-06-01
3	3	1164	2017-06-02
4	4	1506	2017-06-02
5	5	1789	2017-06-02
6	6	1643	2017-06-03
7	7	1190	2017-06-03
8	8	1164	2017-06-03
9	9	1183	2017-06-03
10	10	1122	2017-06-04
11	11	1789	2017-06-04
12	12	1643	2017-06-04

Pool Swipes

Similar to the tennis courts, use of the pool also requires members to swipe their membership card. This table includes the member's account number and the date they used pool.

```
CREATE TABLE Pool_Swipes (
        poolSwipeID serial,
        accountNumber int not null,
        poolSwipeDate date,
        primary key(poolSwipeID)
);
```

Functional Dependencies

poolSwipeID -> accountNumber, poolSwipeDate

Pool Swipes Table

	poolswipeid integer	accountnumber integer	poolswipedate date
1	1	1010	2017-06-01
2	2	1506	2017-06-01
3	3	1643	2017-06-01
4	4	1123	2017-06-02
5	5	1506	2017-06-02
6	6	1789	2017-06-02
7	7	1506	2017-06-03
8	8	1122	2017-06-04
9	9	1164	2017-06-05
10	10	1010	2017-06-06
11	11	1190	2017-06-06
12	12	1122	2017-06-06

Views

Employee Info

This displays every employee's first name, last name, phone number, and their job position.

```
DROP VIEW IF EXISTS EmployeeInformation;

CREATE VIEW EmployeeInformation AS
SELECT employees.pid, people.firstName, people.lastName, people.phoneNumber, Job_Types.jobPosition, payPerHourUSD AS jobInfo
FROM people
INNER JOIN employees on people.pid = employees.pid
INNER JOIN Job_Types on employees.jid = Job_Types.jid
ORDER BY PID ASC;

SELECT * FROM EmployeeInformation;
```

	pid integer	firstname text	lastname text	phonenumber bigint	jobposition text	jobinfo integer
1	1	Victoria	Confeiteiro	9149241907	Manager	10
2	3	Alan	Labouseur	8455753000	Bartender	12
3	4	Miceala	Martini	5186451234	Counter Worker	11
4	5	Julie	Martinelli	9148901234	Snack Bar Manager	22
5	10	Joseph	Confeiteiro	9149241907	Lifeguard	8
6	11	Hillary	Clinton	9144334396	Bartender	30
7	18	Ron	Swanson	8451234567	Manager	40
8	24	Richie	Ortiz	9143708919	Chef	20
9	28	Jean-Ralphio	Saperstein	914336767	Janitor	7
10	29	John John	Florence	8702335678	Bar Back	15

Members and Employees

This view displays all information associated with all members that are also employees.

```
DROP VIEW IF EXISTS MembersAndEmployees;

CREATE VIEW MembersAndEmployees AS
SELECT All_Members.pid, people.firstName, people.lastName, people.phoneNumber, All_Members.accountNumber, Job_Types.jobPosition AS MemberInfo
FROM People
INNER JOIN All_Members on people.pid = All_Members.pid
INNER JOIN Employees on people.pid = employees.pid
INNER JOIN Job_Types on employees.jid = Job_Types.jid
ORDER BY PID ASC;;

SELECT * FROM MembersAndEmployees;
```

	pid integer	firstname text	lastname text	phonenumber bigint	accountnumber integer	memberinfo text
1	11	Hillary	Clinton	9144334396	1506	Bartender
2	29	John John	Florence	8702335678	1183	Bar Back

Reports

Single Members, No Kids Pool Usage

This returns the first and last names, and the amount of times members that have the single, no kids membership used the pool.

```
SELECT people.firstName, people.lastName, Membership_Account_Numbers.poolSwipes
FROM People
INNER JOIN All_Members on People.pid = All_Members.pid
INNER JOIN Membership_Account_Numbers on All_Members.accountNumber = Membership_Account_Numbers.accountNumber
INNER JOIN Account_Types on Membership_Account_Numbers.ATID = Account_Types.ATID
WHERE Account_Types.ATID = '1'
ORDER BY Membership_Account_Numbers.poolSwipes ASC;
```

	firstname text	lastname text	poolswipes integer
1	Bernie	Sanders	2
2	John John	Florence	3
3	Tom	Haverford	4
4	Hillary	Clinton	10

Accounts that don't use electronic billing

This return a list of accounts and their primary account holders that choose not to use electronic billing. This also returns their address to be used for billing.

```
SELECT people.firstName, people.lastName, people.streetAddress, people.zipCode, states.city, states.state
FROM People
INNER JOIN states on people.zipCode = states.zipCode
INNER JOIN Primary_Account_Holders on people.pid = Primary_Account_Holders.pid
WHERE Primary_Account_Holders.electronicBilling = 'NO'
ORDER BY firstName ASC;
```

	firstname text	lastname text	streetaddress text	zipcode integer	city text	state text
1	Jim	Halpert	98 Green Lane	17404	Scranton	Pennsylvania

Stored Procedures

Return Bar Transactions

This procedure will return all the bar transactions with their account number and the transaction dollar amount for any day inputted.

```
CREATE OR REPLACE FUNCTION barTransactionDates(date, refcursor) RETURNS refcursor AS
$$
DECLARE
        barTransactionDates date :=$1;
        resultset refcursor :=$2;
BEGIN
        open resultset FOR
                SELECT Bar_Charges.barTransactionID,
                       Bar_Charges.accountNumber,
                       Bar_Charges.amountUSDBar
                FROM Bar_Charges
                WHERE barTransactionDates = Bar_Charges.transactionDateBar;
        RETURN resultset;
END
$$
language plpgsql;
--Example
SELECT barTransactionDates('2017-06-05', 'results');
FETCH ALL FROM results;
```

Return Bar Transactions

```
--Example
SELECT barTransactionDates('2017-06-05', 'results');
FETCH ALL FROM results;
```

	bartransactionid integer	accountnumber integer	amountusdbar integer
1	6	1122	5
2	7	1164	300
3	8	1010	350

Snack Bar Day Total

This procedure will return the snack bar total for any day inputted.

```
CREATE OR REPLACE FUNCTION snackBarTotals(date, refcursor) RETURNS refcursor AS
$$
DECLARE
        snackBarTotals date :=$1;
        resultset refcursor :=$2;
BEGIN
        open resultset FOR
                SELECT sum(amountUSDSnack)
                FROM Snack_Bar_Charges
                WHERE snackBarTotals = transactionDateSnack;
        RETURN resultset;
END
$$
language plpgsql;
```

Snack Bar Total

```
--Example
SELECT snackBarTotals('2017-06-01', 'results');
FETCH ALL FROM results;
```

	sum bigint
1	505

Security

Manager

The manager(s) of the Amackassin can view, update, or delete any information on the database.

```
create role Manager;
grant all on all tables in schema public to Manager;
```

Bartender

At the end of the day, the bartender enters in all of the bar transactions into the Bar Charges table, so the bartender must be able to enter in a transaction, or update the record if they made a mistake while entering.

```
create role Bartender;
revoke all on all tables in schema public from Bartender;
grant select on all tables in schema public to Bartender;
grant insert on barCharges to Bartender;
grant update on barCharges to Bartender;
```

Assistant Manager

The assistant manager is allowed to update records already stored in the Employees, People, All_Members, and Primary_Account_Holders table. Managers are the only ones allowed to add new members or delete old ones, but the assistant manager is allowed to make changes to these records.

```
create role Assistant_Manager;
revoke all on all tables in schema public from Assistant_Manager;
grant select on all tables in schema public to Assistant_Manager;
grant update on Employees, People, All_Members, Primary_Account_Holders to Assistant_Manager;
```

Implementation Notes, Known Problems, Future Enhancements

This database is made up of purely sample data, therefore this example is much smaller than it would realistically be. This database however could accommodate all of the data the club would need if it were implemented.

In the future, I would like to add a Kids table as a subtype of the People table. In reflecting on the design, while it is perfectly fine now, kids that are on their parent's membership account don't need an address, email address, or phone number associated with them. Right now they just have their parent's information in these columns, but in the future this could be eliminated.

In the future it would also be beneficial to create a table that is automatically updated when an employee clocks in or out. This would be helpful to the manager while they are calculating employee payroll.