Jake Dineen

**Data Admin Concepts & Database Management  
2017-0707 IST 659  
FINAL PROJECT  
CHAD HARPER**

**Table of Contents**

Scope

Conceptual Model

Glossary

Normalized Logical Model

Database Architecture

Data Questions

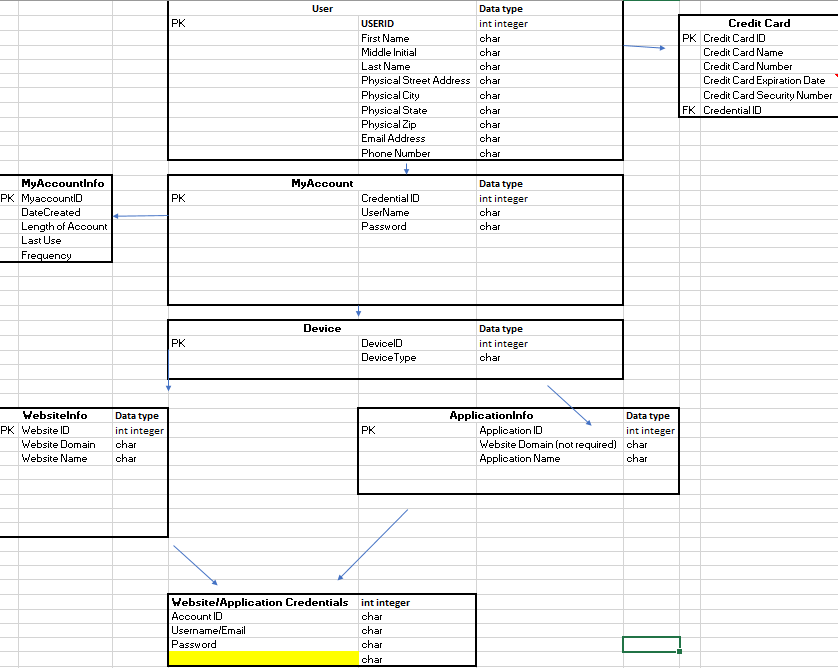
Implementation

Reflection

**Scope:**

One of the most common problems I have, and one that many people seem to have, judging by the number of memes floating around about the subject, is my inability to remember which username and password is associated with which application. It certainly would be nice if there were standardized/uniformed ways for us to log into all our accounts, whether they be to access our finances, to shop online, or to watch our favorite streaming service. Unfortunately, these different applications often require a different form of username/password credentials, and often the username that we have used on past accounts is somehow taken by someone else, no matter how unique it may be. Either that or we are prompted to enter an email address and a password, so we must try to remember which of our many email accounts, each with various passwords over the past few years, we are to use. Often, we are locked out of our account and forced to create a new password or fetch our existing credentials, which is not very cumbersome in and of itself, but there's is quite a bit of ambiguity involved, and quite a bit of accompanying frustration. It is my hope to create the framework for a user-friendly database that contains each/each/all our login credentials for our macro-ing pleasure. Is this a lifechanging application? By all means, no. Would it be useful and alleviate some of the stresses of daily routine, whether it be personal or professional? I think, yes. I've used a system like this on my desktop before, but never for my mobile device, so I thought that it would be interesting to see if I could blend the framework for a multifunctional/multidevice system. Because there is highly sensitive data, potentially, there needs to be layers of security that protect personal identification information from getting out, and the relationships need to be mapped in a way that will eliminate any potential mismatch of consumer data. For this project, my application/storage system will be recognized as 'my program'.

**Conceptual Model:**

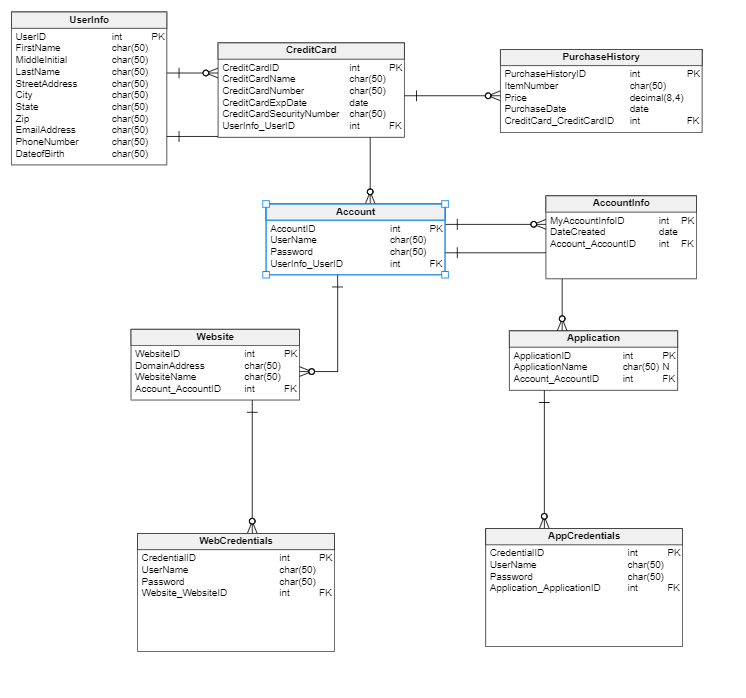


**Preliminary Glossary:**

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | | | |
| Entity | Attribute | Definition | Example |
| User | USERID | Unique Identifier (PK) | 1 |
| User | First Name | User's forename | John |
| User | Middle Initial | User's middle initial(s) | D |
| User | Last Name | User's surname | Doe |
| User | Physical Street Address | User's Physical Street Address | 1111 E Park Avenue |
| User | Physical City | User's Physical City | NY |
| User | Physical State | User's Physical State | NY |
| User | Physical Zip | User's Physical Zip | 51654 |
| User | Email Address | User's Main Email Address | [John.Doe@Gmail.com](mailto:John.Doe@Gmail.com) |
| User | Phone Number | User's Phone Number | 4808888888 |
| User | Date of Birth | User's Date of Birth | 8/4/1957 |
|  |  |  |  |
| Website | Website ID | Unique Identifier (PK) | 1 |
| Website | Website Domain | Website Domain | amazon.com |
| Website | Website Name | Website Name | amazon |
| Website | Platform Type (app, wap) | Platform Type | Website |
|  |  |  |  |
| Application | Application ID | Application ID | 1 |
| Application | Website Domain (not required) | Website Domain (not required) | chase.com |
| Application | Application Name | Application Name | chase |
| Application | Platform Type (app, wap) | Platform Type (app, wap) | application |
|  |  |  |  |
| Account | Account ID | Account ID | 1 |
| Account | Username | Username | Jdoe |
| Account | Password | Password | 123456 |
| Account | User Device Type | User Device Type | Mobile |
|  |  |  |  |
|  |  |  |  |
| Mycredentials | Credential ID | Credential ID | 1 |
| Mycredentials | UserName | UserName | Jdoe |
| Mycredentials | Password | Password | Jdoe123 |
| Mycredentials | Email Address | Email Address | [John.Doe@Gmail.com](mailto:John.Doe@Gmail.com) |
| Mycredentials | Date Created | Date Created | 7/29/2017 |
| Mycredentials | Length of Account | Length of Account ((Current Date - Date Created)/30) | 1 |
|  |  |  |  |
| Credit Card | Credit Card ID | Credit Card ID | 1 |
| Credit Card | Credit Card Name | Credit Card Name | John Doe |
| Credit Card | Credit Card Number | Credit Card Number | 8.88888880000000E+07 |
| Credit Card | Credit Card Expiration Data | Credit Card Expiration Data | 6.19 |
| Credit Card | Credit Card Security Number | Credit Card Security Number | 123 |
|  |  |  |  |
| Purchase History | Purchase History ID | Purchase History ID | 1 |
| Purchase History | Item Number | Item Number | 5 |
| Purchase History | Price | Price | 9.99 |
| Purchase History | Purchase Date | Purchase Date | 8/4/2017 |

\

**Normalized Logical Model Vizio:**



---------------------------------- DROP TABLE CALLS-------------------------------------------------

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'AppCredentials' )

BEGIN

DROP TABLE AppCredentials

END

GO

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'WebCredentials' )

BEGIN

DROP TABLE WebCredentials

END

GO

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'AccountInfo' )

BEGIN

DROP TABLE AccountInfo

END

GO

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'PurchaseHistory' )

BEGIN

DROP TABLE PurchaseHistory

END

GO

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'CreditCard' )

BEGIN

DROP TABLE CreditCard

END

GO

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'Application' )

BEGIN

DROP TABLE Application

END

GO

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'Website' )

BEGIN

DROP TABLE Website

END

GO

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'Account' )

BEGIN

DROP TABLE Account

END

GO

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'UserInfo' )

BEGIN

DROP TABLE "UserInfo"

END

---------------------------------- tables-------------------------------------------------

-- tables

-- Table: Account

CREATE TABLE Account (

AccountID int identity NOT NULL,

UserName char(50) NOT NULL,

Password char(50) NOT NULL,

UserInfo\_UserID int NOT NULL,

CONSTRAINT Account\_pk PRIMARY KEY (AccountID)

);

-- Table: AccountInfo

CREATE TABLE AccountInfo (

MyAccountInfoID int identity NOT NULL,

DateCreated date NOT NULL,

Account\_AccountID int NOT NULL,

CONSTRAINT AccountInfo\_pk PRIMARY KEY (MyAccountInfoID)

);

-- Table: AppCredentials

CREATE TABLE AppCredentials (

CredentialID int identity NOT NULL,

UserName char(50) NOT NULL,

Password char(50) NOT NULL,

Application\_ApplicationID int NOT NULL,

CONSTRAINT AppCredentials\_pk PRIMARY KEY (CredentialID)

);

-- Table: Application

CREATE TABLE Application (

ApplicationID int identity NOT NULL,

ApplicationName char(50) NULL,

Account\_AccountID int NOT NULL,

CONSTRAINT Application\_pk PRIMARY KEY (ApplicationID)

);

-- Table: CreditCard

CREATE TABLE CreditCard (

CreditCardID int identity NOT NULL,

CreditCardName char(50) NOT NULL,

CreditCardNumber char(50) NOT NULL,

CreditCardExpDate date NOT NULL,

CreditCardSecurityNumber char(50) NOT NULL,

UserInfo\_UserID int NOT NULL,

CONSTRAINT CreditCard\_pk PRIMARY KEY (CreditCardID)

);

-- Table: PurchaseHistory

CREATE TABLE PurchaseHistory (

PurchaseHistoryID int identity NOT NULL,

ItemNumber char(50) NOT NULL,

Price decimal(8,4) NOT NULL,

PurchaseDate date NOT NULL,

CreditCard\_CreditCardID int NOT NULL,

CONSTRAINT PurchaseHistory\_pk PRIMARY KEY (PurchaseHistoryID)

);

-- Table: UserInfo

CREATE TABLE UserInfo (

UserID int identity NOT NULL,

FirstName char(50) NOT NULL,

MiddleInitial char(50) NOT NULL,

LastName char(50) NOT NULL,

StreetAddress char(50) NOT NULL,

City char(50) NOT NULL,

State char(50) NOT NULL,

Zip char(50) NOT NULL,

EmailAddress char(50) NOT NULL,

PhoneNumber char(50) NOT NULL,

DateofBirth char(50) NOT NULL,

CONSTRAINT UserInfo\_pk PRIMARY KEY (UserID)

);

-- Table: WebCredentials

CREATE TABLE WebCredentials (

CredentialID int identity NOT NULL,

UserName char(100) NOT NULL,

Password char(50) NOT NULL,

Website\_WebsiteID int NOT NULL,

CONSTRAINT WebCredentials\_pk PRIMARY KEY (CredentialID)

);

-- Table: Website

CREATE TABLE Website (

WebsiteID int identity NOT NULL,

DomainAddress char(50) NOT NULL,

WebsiteName char(50) NOT NULL,

Account\_AccountID int NOT NULL,

CONSTRAINT Website\_pk PRIMARY KEY (WebsiteID)

);

----------------------------- foreign keys------------------------------------------

-- foreign keys

-- Reference: AccountInfo\_Account (table: AccountInfo)

ALTER TABLE AccountInfo ADD CONSTRAINT AccountInfo\_Account

FOREIGN KEY (Account\_AccountID)

REFERENCES Account (AccountID);

-- Reference: Account\_UserInfo (table: Account)

ALTER TABLE Account ADD CONSTRAINT Account\_UserInfo

FOREIGN KEY (UserInfo\_UserID)

REFERENCES UserInfo (UserID);

-- Reference: AppCredentials\_Application (table: AppCredentials)

ALTER TABLE AppCredentials ADD CONSTRAINT AppCredentials\_Application

FOREIGN KEY (Application\_ApplicationID)

REFERENCES Application (ApplicationID);

-- Reference: Application\_Account (table: Application)

ALTER TABLE Application ADD CONSTRAINT Application\_Account

FOREIGN KEY (Account\_AccountID)

REFERENCES Account (AccountID);

-- Reference: Credentials\_Website (table: WebCredentials)

ALTER TABLE WebCredentials ADD CONSTRAINT Credentials\_Website

FOREIGN KEY (Website\_WebsiteID)

REFERENCES Website (WebsiteID);

-- Reference: CreditCard\_UserInfo (table: CreditCard)

ALTER TABLE CreditCard ADD CONSTRAINT CreditCard\_UserInfo

FOREIGN KEY (UserInfo\_UserID)

REFERENCES UserInfo (UserID);

-- Reference: PurchaseHistory\_CreditCard (table: PurchaseHistory)

ALTER TABLE PurchaseHistory ADD CONSTRAINT PurchaseHistory\_CreditCard

FOREIGN KEY (CreditCard\_CreditCardID)

REFERENCES CreditCard (CreditCardID);

-- Reference: Website\_Account (table: Website)

ALTER TABLE Website ADD CONSTRAINT Website\_Account

FOREIGN KEY (Account\_AccountID)

REFERENCES Account (AccountID);

---------------------------------------------------------------DUMMY DATA----------------------------------------------------------------------------------

-- Dummy 'User' Table Data

Insert INTO "UserInfo" (FirstName, MiddleInitial, LastName, StreetAddress, City, State, Zip, EmailAddress, PhoneNumber, DateofBirth)

Values ('Jake', 'A', 'Dineen', '1811 E Park Ave', 'Gilbert', 'Arizona', '85234', 'Jdineen@Syr.edu', '4808888888', '08/12/1994')

, ('Nyk', 'A', 'Dineen', '1811 E Park Ave', 'Gilbert', 'Arizona', '85234', 'Ndineen@Gmail.com', '4808888889', '08/01/1991')

, ('Lee', 'F', 'Howard', '1444 E San Remo', 'Gilbert', 'Arizona', '85234', 'LeeFHoward@gmail.com', '4801234567', '01/07/1954')

, ('Anthony', 'G', 'Taylor', '926 E Gale', 'New York', 'New York', '10210', 'Ataylor@gmail.com', '9261453654', '08/16/1994')

, ('Francis', 'F', 'Nobody', '1234 E Shore', 'Los Angeles', 'California', '90210', 'Francis.Nobody@hotmail.com', '5451324564', '12/25/1945')

Select \* From "UserInfo"

-- Dummy 'CreditCard' Table Data

INSERT INTO CreditCard (CreditCardName, CreditCardNumber, CreditCardExpDate, CreditCardSecurityNumber, UserInfo\_UserID)

VALUES ('Visa', '1234 5678 9101 1121', 'August 2019', '123', '1')

, ('American Express', '3898 6990 8542 1623', 'February 2018', '464', '2')

, ('Mastercard', '2415 0877 4954 3486', 'March 2020', '943', '3')

, ('Visa', '9015 4133 7954 3126', 'December 2021', '514', '4')

, ('American Express', '9636 1137 5246 1464', 'January 2018', '853', '5')

, ('American Express', '4200 7400 7781 0412', 'August 2020', '761', '1')

Select \* from CreditCard

-- Dummy 'PurchaseHistory' Table Data

Insert Into PurchaseHistory (ItemNumber, Price, PurchaseDate, CreditCard\_CreditCardID)

Values ('0081', '9.99', 'June 22, 2017', '1')

,('0091', '9.99', 'July 22, 2017', '1')

,('0121', '9.99', 'August 22, 2017', '1')

, ('0157', '9.99', 'July 7 2017', '2')

, ('0181', '9.99', 'July 22 2017', '3')

, ('0206', '9.99', 'January 15, 2017', '4')

, ('0213', '9.99', 'January 31, 2017', '5')

, ('0251', '9.99', 'June 26, 2017', '6')

select \* from PurchaseHistory

-- Dummy 'Account' Table Data

INSERT INTO Account (Account.UserName, Password, UserInfo\_UserID)

Values ('Jdineen', 'Jake1234', '1')

, ('Ndineen', 'Dolphin1!', '2')

, ('LFH179', 'Pass123', '3')

, ('Atay561', 'Baseball1!', '4')

, ('FFNobody1', 'FootballFF', '5')

select \* from Account

-- Dummy 'Account Info' Table Data

INSERT INTO AccountInfo (DateCreated, Account\_AccountID)

VALUES ('June 22, 2017', '1')

, ('July 7 2017', '2')

, ('July 22 2017', '3')

, ('January 15, 2017', '4')

, ('January 31, 2017', '5')

Select \* From AccountInfo

-- Dummy 'Website' Table Data

INSERT INTO Website (DomainAddress, WebsiteName, Account\_AccountID)

Values ('Amazon.com', 'Amazon', '1')

, ('Gmail.com', 'Gmail', '1')

, ('Netflix.com', 'Netflix', '2')

, ('Hulu.com', 'Hulu', '2')

, ('Target.com', 'Target', '3')

, ('BestBuy.com', 'BestBuy', '3')

, ('Chase.com', 'Chase Banking', '4')

, ('2SU.com', '2SU', '4')

, ('2su.ischoolonline.syr.edu', '2SU Ischool', '5')

Select \* From Website

--Dummy 'Application' Table Data

Insert into Application (ApplicationName, Account\_AccountID)

Values ('Desert Schools Mobile Banking', '1')

,('American Express', '1')

,('Facebook', '1')

,('Instagram', '1')

,('Facebook', '2')

,('Instagram', '2')

,('Snapchat', '3')

,('Apple', '4')

,('Youtube', '5')

,('Spotify','5')

Select \* From Application

--Dummy 'WebCredentials' Table Data

Insert INTO WebCredentials (UserName, Password, Website\_WebsiteID)

Values ('Jdineen123@hotmail.com', 'Jake1234!', '1')

, ('Jdineen81294@gmail.com' , 'Password1!', '2')

,('Ndineen@yahoo.com' , 'Nyk1234', '3')

,('Ndineen@yahoo.com' , 'Nyk123', '4')

,('LFHoward', 'Meagain1!', '5')

,('Lhoward', ' Meagain2!', '6')

,('Ataylor', 'Football1!', '7')

,('AJTay123', 'Latitudes', '8')

,('FranNob', 'Frankie812' ,'9')

Select \* From WebCredentials

-- Dummy 'AppCredentials' Table Data

INSERT INTO AppCredentials(username, Password, Application\_ApplicationID)

Values ('Jdineen', 'globe123', '1')

,('Jdineen81294@gmail.com' , 'Password1!', '2')

,('NykDineen8193', 'Dineen123!', '3')

, ('Ndineen@yahoo.com' , 'Nyk123', '4')

,('LeeFHoward', 'Meagain123!', '5')

,('LFhoward', ' Meagain2186!', '6')

,('Ajftaylor', 'Football123!', '7')

,('AJTay', 'Latitudes11!', '8')

,('FFNob', 'Frankie812' ,'9')

Select \* From AppCredentials

----------------------------- Drops------------------------------------------

--Views

--DROP Views

Drop View UserCreditCardInformation

Drop View MyProgramUserCredentials

Drop View AllWebCredentials

Drop View AllAppCredentials

--Functions

Drop Function dbo.creditcardfetch

Drop Function DBO.RevenuePerUser

Drop Function DBO.ApplicationCounts

Drop Function DBO.AppNameFetch

Drop Function dbo.MyProgUNFetch

Drop Function dbo.MyProgPWFetch

Drop Function dbo.PoDateFetch

--Drop Sprocs

Drop Procedure AddUserInfo

Drop Procedure AddWebsiteInfo

Drop Procedure AddWebsiteandCreds

Drop Procedure AddAppandCreds

---------------------------------- Views-------------------------------------------------

--1. View that shows individual user linked to credit card, and purchase history.

GO

Create View UserCreditCardInformation As

Select

FirstName

,LastName

,CreditCardName

,CreditCardNumber

From UserInfo

Join CreditCard on CreditCard.UserInfo\_UserID=UserInfo.UserID

--2. View that shows individual user linked to ‘Myprogram’ Credentials.

GO

Create View MyProgramUserCredentials As

Select

FirstName

,LastName

,UserName

,Password

From Account

Left Join UserInfo on UserInfo.UserID=Account.UserInfo\_UserID

--3. View that users individual user linked to all Web Credentials.

GO

Create View AllWebCredentials As

Select

FirstName

,LastName

, Website.WebsiteName

, WebCredentials.UserName as WebUserName

, WebCredentials.Password as WebPassword

From Account

Left Join UserInfo on UserInfo.UserID=Account.UserInfo\_UserID

Join Website on Website.Account\_AccountID=Account.AccountID

Join WebCredentials on WebCredentials.Website\_WebsiteID=Website.WebsiteID

--4 View that users individual user linked to all App Credentials.

GO

CREATE View AllAppCredentials As

Select

FirstName

,LastName

, Application.ApplicationName

, AppCredentials.UserName as AppUserName

,AppCredentials.Password as AppPassword

From Account

Left Join UserInfo on UserInfo.UserID=Account.UserInfo\_UserID

Join Application on Application.Account\_AccountID= Account.AccountID

Join AppCredentials on AppCredentials.Application\_ApplicationID=Application.ApplicationID

-- Test Views

GO

Select \* from UserCreditCardInformation where FirstName = 'Jake' -- This view shows Credit Card Information

Select \* From MyProgramUserCredentials where FirstName = 'Jake' -- This View shows MyProgram credentials

Select \* From AllWebCredentials where FirstName = 'Jake' -- This View shows all stored Web Credentials

Select \* From AllAppCredentials where FirstName = 'Jake'-- This View shows all stored App Credentials

-- Create function that takes UserInfo.UserID and gives credit card id as an output

Go

Create Function dbo.creditcardfetch (@Userid int)

returns int

AS

BEGIN

Return

(

Select CreditCard.CreditCardID

from CreditCard

where creditcard.UserInfo\_UserID = @Userid)

END

Go

Select dbo.creditcardfetch (5)

--test

-- Create function that takes CreditCardID as an input and returns total revenue Generated

Go

Create Function DBO.RevenuePerUser (@CCID int)

returns decimal

AS

BEGIN

Return

(

Select sum(PurchaseHistory.Price) As totalrevenue

from PurchaseHistory

where PurchaseHistory.CreditCard\_CreditCardID = @CCID

)

END

GO

--test

Select DBO.RevenuePerUser (3)

-- Show the number of accounts a particular application name is related to

Go

Create Function DBO.ApplicationCounts (@AccID int)

returns int

AS

BEGIN

Return

(

select

count(application.applicationid)

from

Application

where Application.Account\_AccountID = @AccID

)

END

GO

--test

Select DBO.ApplicationCounts (3)

-- Return Application name from application id input

Go

Create Function DBO.AppNameFetch (@AppID int)

returns char(50)

AS

BEGIN

RETURN

(

Select

Application.ApplicationName

From Application

where Application.ApplicationID = @AppID

)

END

GO

--test

Select DBO.AppNameFetch (1)

-- Take UserID as an input and fetch username for My Program

Go

Create Function dbo.MyProgUNFetch (@UserID int)

returns char(50)

AS

Begin

RETURN

(

Select

Account.UserName

From Account

where Account.UserInfo\_UserID = @UserID

)

END

GO

--Test

select dbo.MyProgUNFetch (1)

-- Take UserID as an input and fetch password for My Program

Go

Create Function dbo.MyProgPWFetch (@UserID int)

returns char(50)

AS

Begin

RETURN

(

Select

Account.Password

From Account

where Account.UserInfo\_UserID = @UserID

)

END

GO

--Test

select dbo.MyProgPWFetch (1)

--Create function that shows the last purchase date for a credit card id

Go

CREATE FUNCTION dbo.PODateFetch (@CCID int)

returns date

AS

BEGIN

RETURN

(

select

max(PurchaseHistory.PurchaseDate)

from PurchaseHistory

where PurchaseHistory.CreditCard\_CreditCardID = @CCID

)

END

GO

--Test

select dbo.PODateFetch (1)

--Stored Procedures

--Create Stored Procedure that adds a User into the UserINFO table (Master Table)

Go

Create Procedure AddUserInfo(@firstname char(50), @middleinitial char(50), @lastname char(50), @streetaddress char(50), @city char(50), @state char(50), @zip char(50), @emailaddress char(50), @phonenumber char(50), @dateofbirth char(50))

AS

BEGIN

Declare @UserCount int

Select @UserCount = Count(UserInfo.UserID) From UserInfo

where UserInfo.FirstName = @firstname

If @UserCount = 0

Begin

Insert Into UserInfo (FirstName, MiddleInitial, LastName, StreetAddress, City, State, Zip, EmailAddress, PhoneNumber, DateofBirth)

values (@firstname, @middleinitial, @lastname, @streetaddress, @city, @state, @zip, @emailaddress, @phonenumber, @dateofbirth)

End

RETURN @@Identity

END

--Test

Exec AddUserInfo 'Bailee', 'A', 'Dineen', '1811 E Park Avenue', 'Gilbert', 'Arizona', '85234', 'Bdineen@gmail.com', '4801231231', '12/15/1998'

-- Check to see if it worked

Select \* from UserInfo

Where FirstName = 'Bailee'

--Create Stored Procedure that Accepts a Website Domain, Name, and Account ID

Select \* From Website

Go

Create Procedure AddWebsiteInfo (@domainaddress char(50), @websitename char(50), @accountid int)

AS

BEGIN

Declare @websitecount int

select @websitecount = count(Website.WebsiteID) From Website

where DomainAddress = @domainaddress

if @websitecount = 0

Begin

Insert into Website (DomainAddress, WebsiteName, Account\_AccountID)

Values (@domainaddress, @websitename, @accountid)

END

RETURN @@Identity

END

--Test

Exec AddWebsiteInfo 'Twitter.com', 'Twitter', '1'

-- Check to see if it worked

Select \* from Website where WebsiteName = 'Twitter'

--Create Stored Procedure to add Website Information and Website Credentials

Go

Create Procedure AddWebsiteandCreds (@domainaddress char(50), @websitename char(50), @accountid int, @Username char(50), @password char(50))

AS

BEGIN

Declare @websitecount int

select @websitecount = count(Website.WebsiteID) From Website

where DomainAddress = @domainaddress

if @websitecount = 0

Begin

Insert into Website (DomainAddress, WebsiteName, Account\_AccountID)

Values (@domainaddress, @websitename, @accountid)

Insert into WebCredentials (WebCredentials.UserName, Password, Website\_WebsiteID)

Values (@Username, @password, @@IDENTITY)

END

RETURN @@Identity

END

--Test

Exec AddWebsiteandCreds 'Capitalone.com', 'Capital One', '1', 'Jadineen81294', 'Longitudes123!@#'

-- Check to see if it worked

Select \* From AllWebCredentials where FirstName = 'Jake'

--Create Stored Procedure to add Application Information and Application Credentials

Select \* From Application

Go

Create Procedure AddAppandCreds (@applicationname char(50), @accountid int, @Username char(50), @password char(50))

AS

BEGIN

Declare @appcount int

select @appcount = count(Application.ApplicationID) From Application

where ApplicationName = @applicationname

if @appcount = 0

Begin

Insert into Application (ApplicationName, Application.Account\_AccountID)

Values (@applicationname, @accountid)

Insert into AppCredentials (UserName, Password, Application\_ApplicationID)

Values (@Username, @password, @@IDENTITY)

END

RETURN @@Identity

END

--Test

Exec AddAppandCreds 'twitter', '1', 'Jadineen81294', 'Longitudes123!@#'

-- Check to see if it worked

Select \* From AllAppCredentials where FirstName = 'Jake'

--Create Stored Procedure to add CreditCard Information

Go

Create Procedure AddCreditCard (@creditcardname char(50), @creditcardnumber char(50), @creditcardexpdate char(50), @creditcardsecnumber char(50), @userid int)

AS

BEGIN

Insert into CreditCard (CreditCardName, CreditCardNumber, CreditCardExpDate, CreditCardSecurityNumber, Creditcard.UserInfo\_UserID)

Values (@creditcardname, @creditcardnumber, @creditcardexpdate, @creditcardsecnumber, @userid)

END

--Test

Exec AddCreditCard 'Visa' , ' 12341234123412341234', 'August 2020', '789', '1'

-- Check to see if it worked

Select \* From UserCreditCardInformation

--Updates

-- Test Update to change credit card expiration date

select \* From CreditCard

Go

Update CreditCard

Set CreditCardExpDate = '10-01-2018'

where Creditcard.UserInfo\_UserID = 1

-- Test Update to change credit card number typo

select \* from creditcard

Update CreditCard

Set CreditCardNumber = '1234 1234 1234 1234'

where CreditCardID = 9

select \* from creditcard

**Answering Data Questions**

Now that the initial design is done, we have much easier navigation of our database through a series of views, functions and stored procedures, as written above the database architecture section.

While this is essentially acting as a credential storage system, there are more complex questions that can be asked, both from the business side of things, and from the user side of things. Below are some examples of those questions.

Views are created as hotkeys, of a sort, on which tables and columns that we frequently find ourselves searching for. What our manually created views within this database show us are:

**UserCreditCardInformation-** Shows First Name, Last Name, Credit Card Name, and Credit Card Number on file. This is used more from a business perspective, but can also be featured on a UI to show the particular user which credit card they are using to pay for the monthly subscription.

**MyProgramUserCredentials:** This view shows User Web Credentials for My Program, the name that we are arbitrarily assigning to the overarching database. This can be used to fetch account information if someone were to forget their ‘master’ password that contains the sub accounts beneath it.

**AllWebCredentials-** Essentially, the point of this project was to be able to store and fetch usernames and passwords for all accounts that I may have, whether they be mobile app based, or web based. This view will fetch a user’s first name, last name, site name, site username and site password.

**AllAppCredentials:** Essentially, the point of this project was to be able to store and fetch usernames and passwords for all accounts that I may have, whether they be mobile app based, or web based. This view will fetch a user’s first name, last name, application name, application username and application password.

These views act as a backend to deciphering user information. The physical design of a UI would require much more security if this was a program cast out to the general public, as my scope presents it as. For personal consumption, if I were to only have my own usernames and passwords in here, I would not necessarily have to worry about increased security, or limiting access on views. As it stands, this database, at this level, and with my knowledge from the course, was built for internal consumption only. What I could do at this point, going back to week 10 learnings, is give front end access to certain users based on their userid’s within my database. For example, if we had the ‘Myprogram’ username of Jdineen, I would only want to grant access to the data that is relative to their userid. This would likely have to be done through some sort of authentication, as I can’t find information on conditional grants or revokes, ie. Grant Select on UserInfo to Jdineen where UserID = 1.

If the information was to be public, I could only show the below columns through filtered Grant commands, as I must mask the credit card information and all the passwords for security reasons.

--Grants/Revoke

Grant select on UserInfo (FirstName, LastName) to guestuser

Grant select on PurchaseHistory (Itemnumber, Price, PurchaseDate) to guestuser

Grant select on Account (AccountID, UserName) to guestuser

Grant Select on Website (DomainAddress, WebsiteName) to guestuser

Grant Select on WebCredentials (UserName) to guestuser

Grant Select on Application (ApplicationName) to guestuser

Grant Select on AppCredentials (Username) to guestuser

Revoke select on CreditCard to guestuser

Revoke all on Userinfo to guestuser

Revoke all on PurchaseHistory to guestuser

Revoke all on Account to guestuser

Revoke all on Website to guestuser

Revoke all on Webcredentials to guestuser

Revoke all on Application to guestuser

Revoke all on AppCredentials to guestuser

This would be a way for a user to backend into the database to query for their existing information, but the object of the final vision of this project would to have a third-party application working in the background that stores user credentials, if given permission, and then can autofill login information with what has been stored. To put it into other words, the outcome can’t be shown at this point, but the groundwork is laid for the next steps.

In addition to views, we can create functions and stored procedures to make our lives easier. This would, however, mainly be applicable to the business users of the application.

**Function dbo.creditcardfetch-** This function takes the Master UserId as an input, and returns the credit card id as an output. This could be useful for shortening query lengths when dealing with customer service calls.,

**Function DBO.RevenuePerUser-** This function takes CreditCardID as an input and returns the total revenue generated from that specific credit card id. This would be useful in a business setting to quantify customer value.

**Function DBO.ApplicationCounts-** This function acts as a way of showing highly populated applications, and their credentials.

**Function DBO.AppNameFetch-** This function takes an applicationid as an input and returns the character representation of the applicationname.

**Function dbo.MyProgUNFetch-** This function takes the Master UserId as an input and returns the related ‘master’ username.

**Function dbo.MyProgPWFetch-** This function takes the Master UserId as an input and returns the related ‘master’ password.

**Function dbo.PoDateFetch-** This function takes the creditcardid as an input, and returns the last transaction date associated with that account.

**Procedure AddUserInfo-** This sproc acts as a shortcut to add a user into the UserInfo table, if they don’t currently exist.

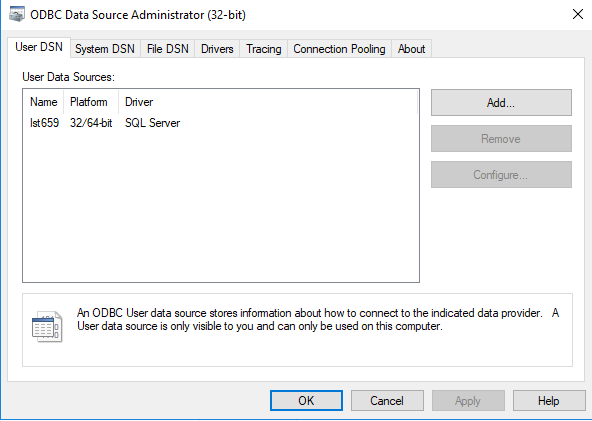
**Procedure AddWebsiteInfo-** This sproc acts as a shortcut to add a website, as it relates to a specifc user, into the Website table, if the website does not currently exist.

**Procedure AddWebsiteandCreds-** This sproc acts as a shortcut that replicates the above procedure, but takes it a step further, and also gives the ability to add in the associated username and password for a specific websiteid.

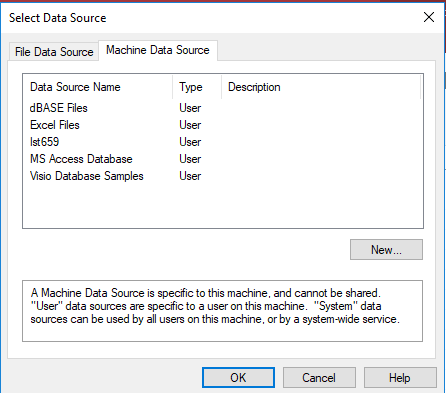
**Procedure AddAppandCreds-** This sproc acts as a shortcut to add an application, as it relates to a specific user, into the application table, as well as allowing for the input of username and password into the application credentials table.

**Implementation**

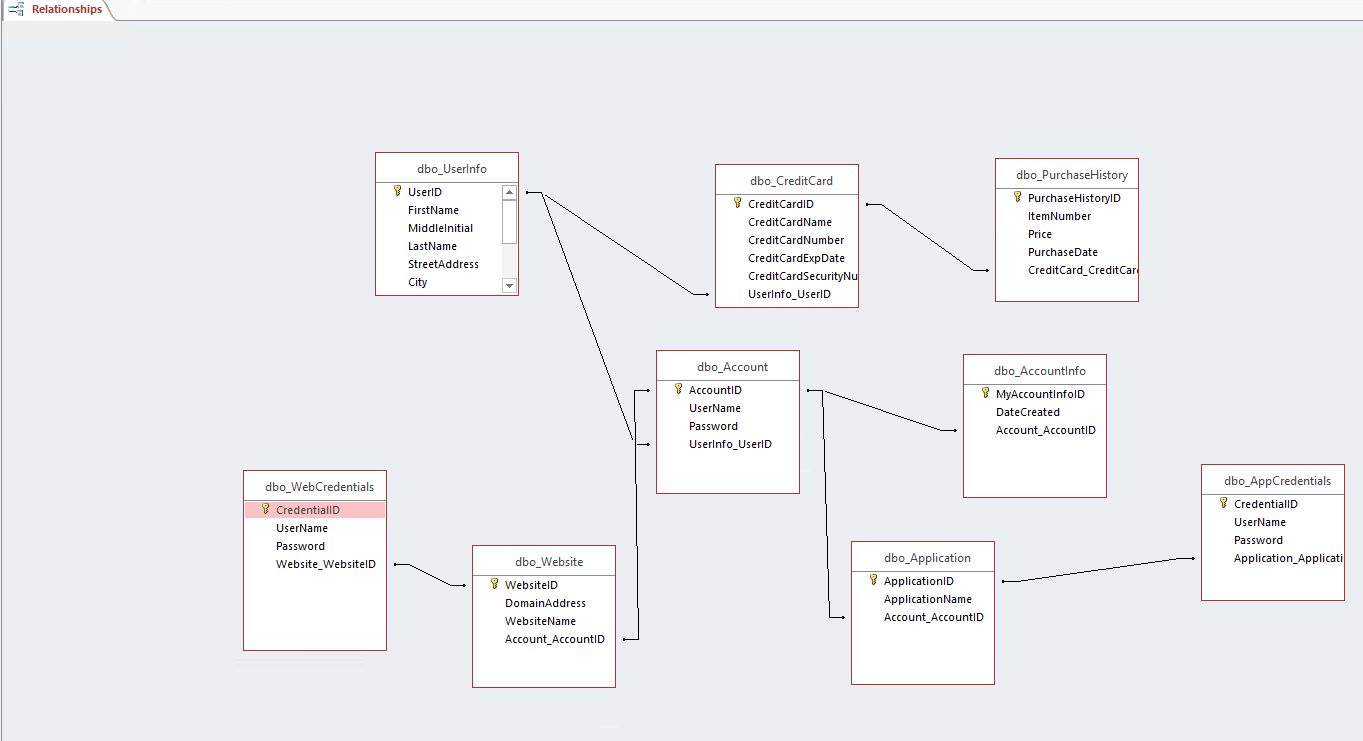
**ODBC:**



**Access, Connecting to Database:**

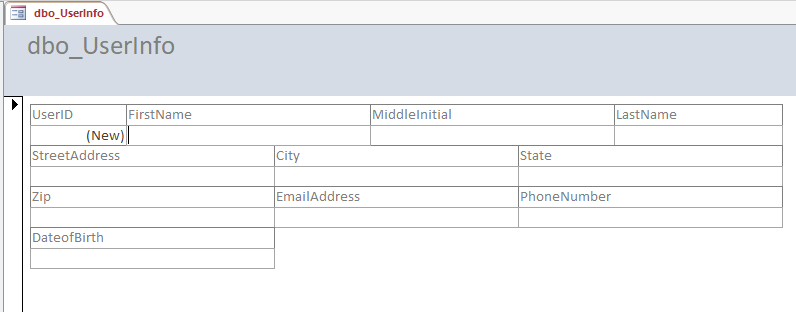


**Access Relationship Mapping:**



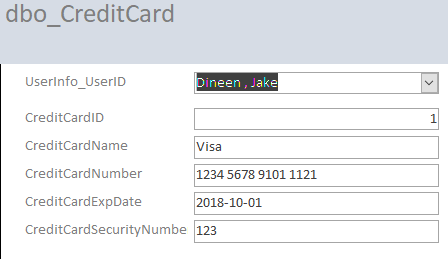
**Form: UserInfo**

The below form can be used to add a user into the UserInfo table. This is essentially the master table within the database. In context of the project, this will be each individual that has registered to use my application.



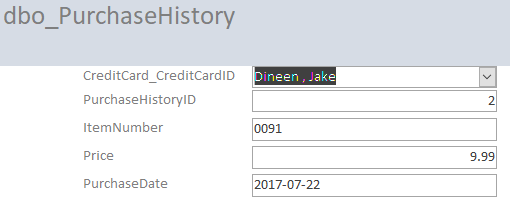
**Form: Credit Card**

The below form can be used to add credit card information into the CreditCard table, using the UserID related First and Last name associated with the account.



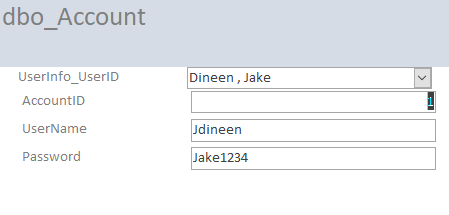
**Form: Purchase History**

The below form adds purchase history records into the Purchase History table. The Credit Card ID query is a reference to both the Credit Card table, and the UserInfo table, populating a name for easier data entry.



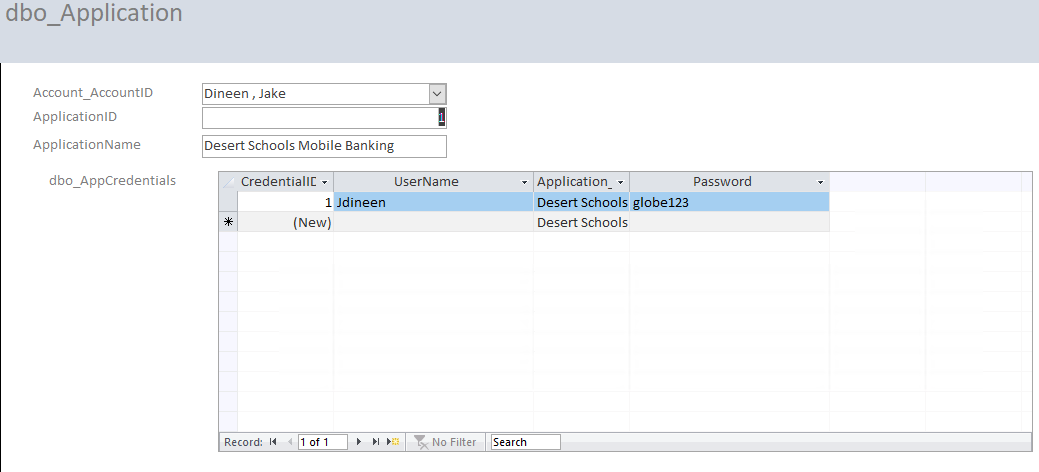
**Form: Account**

The below form allows for us to store master user credentials for my program into the database, showing a person’s FirstName and Lastname from a dropdown for easier entry.

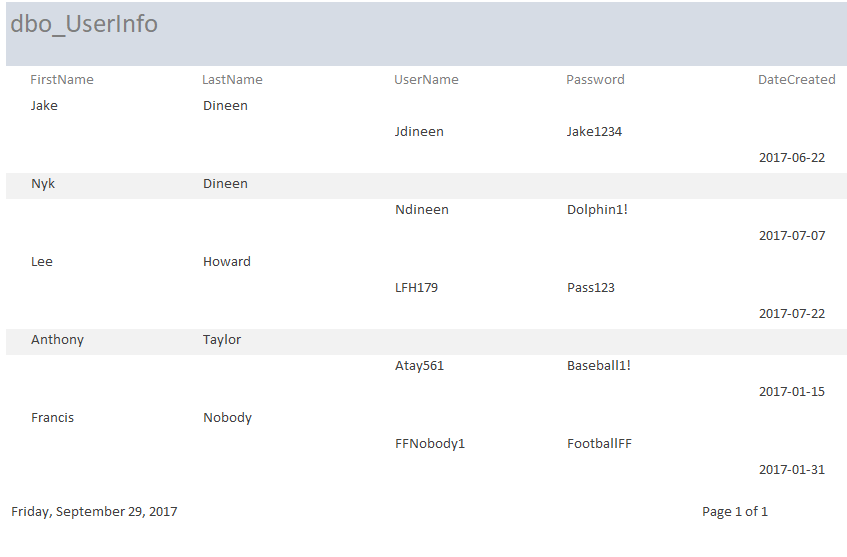


**Form: Application + Application Credentials**

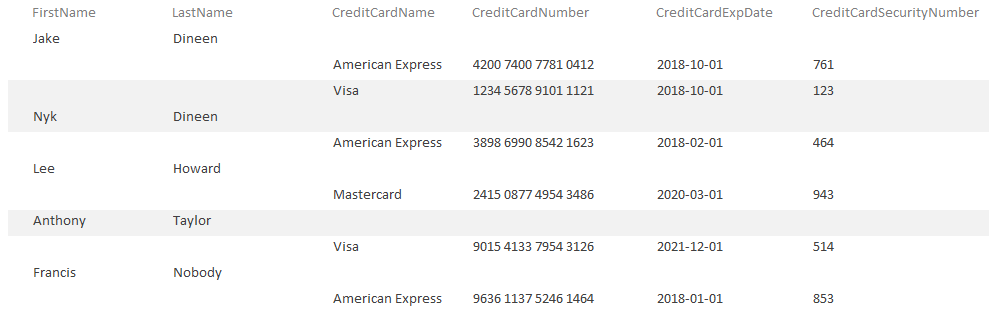
The below form would allow for data entry into both the Application and Application Credentials tables. The accountID would act as the filtered input, linked all the way back to the Userinfo table via FKs. In the Subform, The applicationame column acts as a drop down character derivation of the application id, making it easier to choose which application to insert credentials into.



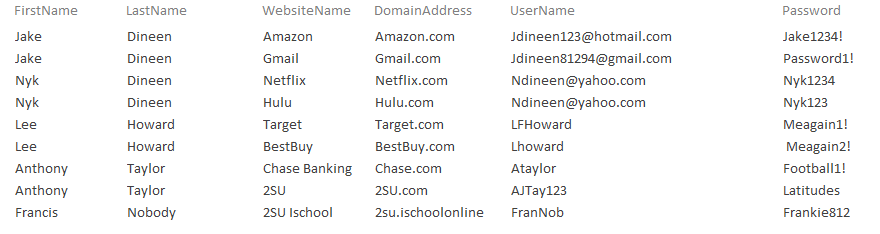
**Report: Show All Master User Names and Passwords, along with Names, and date the account was created.**



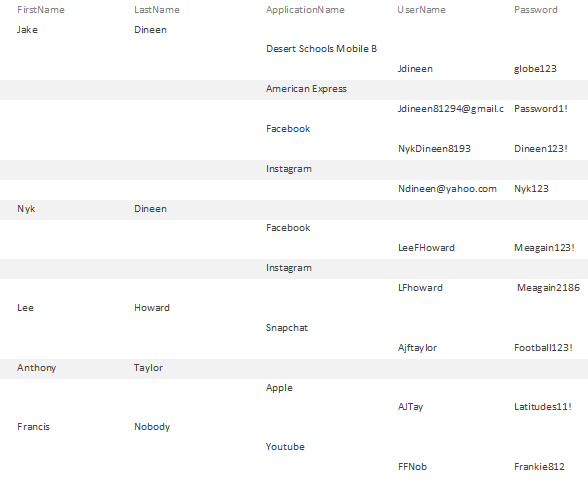
**Report: Show All Credit Card Information on File:**



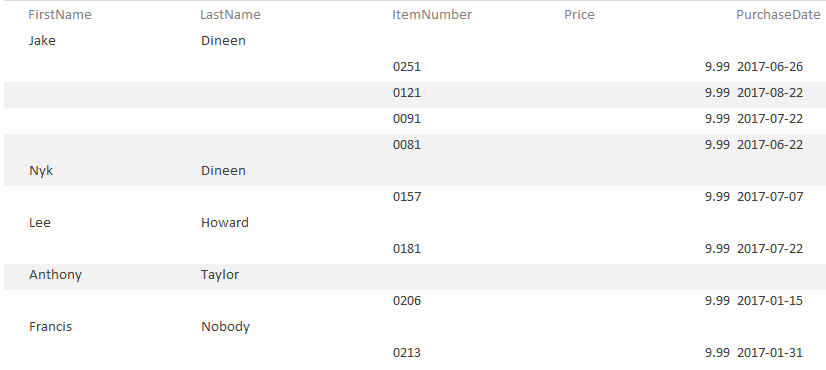
**Report: Show All Website Account Information**



**Report: Show All Application Account Information**



Report: Show Purchase History by User



Reflection/Summary

At the beginning of this project, I think I overstepped in terms of my problem domain. It may have been much easier to create a personalized database than try to and envision one for mass consumption. While the architecture is the same, essentially, the security is far beyond my current knowledge. If this was for internal consumption, and I was simply storing my own user credentials and my own credit card information, this would have all went swimmingly, but when I add in data for other accounts, it went slightly awry.

If I were to continue to work on this database, I would want to hone in on both the security, and the automation. The purpose of an application like this is to minimize any kind of data entry. It’ll be a third-party application working in the background, and will either capture and store data from the form fills when they register for the application, or will ask for permission to store credentials when trying to login to either a site or an application.

I think the basic knowledge I’ve learned throughout this course has been extremely handy. I’ve worked more in R in the past, simply importing files in, but I believe that a lot of work is done in industry by connecting into sql and performing modeling in R.

Until I took this class, my knowledge stopped at Join statements, so adding functions, views and sprocs into my arsenal will be extremely beneficial moving forward.

I mainly chose Access as my front end due to familiarity with the environment, both from this class, and from working with it in the past. It acts as a simple mechanism to run moderately advanced queries, and acts as a way to insert data into tables. I personally think that it is faster to grant access to Stored Procedures and let entry happen from there, but I understand that’s not always desired or possible in a work setting.

I think that I may do some research on how third-party credential storage systems actually work, and try to better understand the layers of security and authentication needed when there is personal identifiers specified within the data. The reason I say this is because no matter what kind of data I may end up working with in the future, there is a high likelihood that at least some of it will be confidential in nature.