Alec Schneider

September 20th, 2020

Database Project

All SQL, Python and HTML code can located at my [github repository](https://github.com/Alec-Schneider/FinancialHealthDatabase).

## Narrative:

People are struggling to understand their financial health now more than ever due to COVID-19. To better understand how we can improve our future financial well beings we need to understand how our debts, bank, and retirement funds change over time. Spending behaviors can be changed by seeing the effects on balances they have, leading to improved financial standing, better decision making, and long-term wealth accumulation to assist future generations. It is time people log their account balances, incomes, and expenses to receive quick and simple to understand feedback about their financial health in order to create a better world for themselves and others.

## Business Rules

* A person is considered a User the moment they sign up with their information.
* A User can have zero or more than one Account. An Account can only belong to one User.
* A User can have zero or many Cash Flow patterns and a Cash Flow pattern can only belong to one and only one user.
* An Account has one and only one Account Type and an Account type belongs to one and only one Category.
* A Category can have many Account Types and an Account Type can belong to zero or many Accounts.

## Glossary

* **User** – A person who has entered their personal information, and will b
* **Discretionary monthly Income** – Income after taxes
* **Average monthly spending** – average estimated monthly spending
* **Account category** – categorization of the account that describes the general characteristics of the account
  + **Debt** – an obligation to pay money to the creditor who loaned the money
  + **Investment** – an account in which you hold stocks, bonds, mutual funds, or other investment vehicles
  + **Bank** – accounts in which can be easily accessed and withdrawn from.

## What Would I Want To Know?

* What is my debt to income ratio?
* What percent of my monthly income am I spending?
* How has spending changed over time?
* How can I create a new user?
* How can I add a new account?
* Can I update my account balance?
* Can I update my cash flow information?
* Can I change my password?

## Conceptual Model

A screenshot of a cell phone

Description automatically generated

## Logical Model

A close up of text on a white background

Description automatically generated

## Physical Database Design

### Database creation

use IST659\_M401\_afschnei;

if object\_id('dbo.account\_updates') is not null

drop table dbo.account\_updates;

if object\_id('dbo.accounts') is not null

drop table dbo.accounts;

if object\_id('dbo.account\_type') is not null

drop table dbo.account\_type;

if object\_id('dbo.account\_category') is not null

drop table dbo.account\_category;

if object\_id('dbo.cash\_flow\_updates') is not null

drop table dbo.cash\_flow\_updates;

if object\_id('dbo.cash\_flow') is not null

drop table dbo.cash\_flow;

if object\_id('dbo.user\_login') is not null

drop table dbo.user\_login

if object\_id('dbo.users') is not null

drop table dbo.users;

-- Create table to stored all user information

create table users(

users\_id int identity,

first\_name varchar(50) not null,

last\_name varchar(50) not null,

middle\_name varchar(50),

date\_of\_birth date not null,

social\_security varchar(11) not null,

email varchar(150) not null,

gender char(1) not null,

creation\_date datetime not null default GetDate(),

-- Constraints

constraint users\_pk primary key (users\_id),

constraint u1\_users unique (social\_security),

constraint u2\_users unique (email)

);

-- Create login table

create table user\_login(

login\_id int identity,

users\_id int not null,

pass varchar(25) not null

constraint user\_login\_pk primary key (login\_id)

constraint user\_login\_fk1 foreign key (users\_id)

references users(users\_id)

)

-- Create table storing the high level account categories

create table account\_category(

account\_category\_id int identity,

account\_category\_name varchar(20) not null,

constraint account\_category\_pk primary key (account\_category\_id),

constraint account\_category\_u1 unique (account\_category\_name)

);

-- Create table containing all of the the account types

create table account\_type(

account\_type\_id int identity,

account\_type\_name varchar(30) not null,

account\_category\_id int not null,

constraint account\_type\_pk primary key (account\_type\_id),

constraint account\_type\_fk1 foreign key (account\_category\_id)

references account\_category(account\_category\_id),

constraint account\_type\_u1 unique (account\_type\_name)

);

-- Create table that stores all user accounts

create table accounts(

account\_id int identity,

account\_type\_id int not null,

users\_id int not null,

constraint account\_pk primary key (account\_id),

constraint account\_fk1 foreign key (account\_type\_id)

references account\_type(account\_type\_id),

constraint account\_fk2 foreign key (users\_id)

references users(users\_id)

)

-- Create the table to track the account updates over time, and the acount information

create table account\_updates(

account\_update\_id int identity,

balance decimal(9,2) not null,

annual\_interest\_rate decimal(4,2),

interest\_rate\_type varchar(8),

repayment\_term int,

monthly\_payment as Round((balance / repayment\_term) \* (1 + (annual\_interest\_rate / 100 / 12)),2),

account\_update\_date datetime not null default GetDate(),

account\_id int not null

constraint account\_update\_pk primary key (account\_update\_id)

constraint account\_update\_fk1 foreign key (account\_id)

references accounts(account\_id)

);

-- Create table to store cash flows of users

create table cash\_flow(

cash\_flow\_id int identity,

users\_id int not null,

constraint cash\_flow\_pk primary key (cash\_flow\_id),

constraint cash\_flow\_fk1 foreign key (users\_id)

references users(users\_id)

);

-- Create table to track a user's cash flow updates over time

create table cash\_flow\_updates(

cash\_flow\_update\_id int identity,

discretionary\_monthly\_income decimal(8,2) not null,

average\_monthly\_spending decimal(8, 2) not null,

cash\_flow\_update\_date datetime not null default GetDate(),

cash\_flow\_id int not null,

constraint cash\_flow\_update\_pk primary key (cash\_flow\_update\_id),

constraint cash\_flow\_update\_fk1 foreign key (cash\_flow\_id)

references cash\_flow(cash\_flow\_id)

);

### Views

-- Views

/\*

View that returns a table of all accounts based on their latest update date.

Returns all of the updated information.

\*/

if object\_id ('dbo.current\_accounts')is not null

drop view dbo.current\_accounts

go

create view current\_accounts as

select

\*

from

(

/\* return a table of the accounts ordered by their creation\_date

and a row\_number field of each cash flow in order to pick the latest one

\*/

SELECT

\*,

ROW\_NUMBER() over (partition by account\_id order by account\_update\_date desc) as RowNum

from

account\_updates) as x

WHERE RowNum = 1

go

-- select the view

select \* from current\_accounts

go

/\*

View that returns a table of all user cash flows based on their latest update date.

Returns all of the updated information.

\*/

if object\_id ('dbo.current\_cash\_flows')is not null

drop view dbo.current\_cash\_flows

go

create view current\_cash\_flows as

select

\*

from

(

/\* return a table of the cash\_flows ordered by their creation\_date

and a row\_number field of each cash flow in order to pick the latest one

\*/

select

\*,

row\_number() over (partition by cash\_flow\_id order by cash\_flow\_update\_date desc) as RowNum

from cash\_flow\_updates) as x

where RowNum = 1

go

select \* from current\_cash\_flows

go

/\*

Return a table of total debt balances by user, using the

current debt accounts

\*/

if object\_id('dbo.user\_debt') is not null

drop view dbo.user\_debt

go

create view user\_debt as

select

users.users\_id,

sum(latest\_accounts.balance) as current\_total\_debt

from

accounts acct

join

users on acct.users\_id = users.users\_id

join

(

select

account\_id,

balance,

account\_update\_date

from

dbo.current\_accounts

) as latest\_accounts on latest\_accounts.account\_id = acct.account\_id

join

account\_type actype on acct.account\_type\_id = actype.account\_type\_id

join

account\_category act\_cat on actype.account\_category\_id = act\_cat.account\_category\_id

where

act\_cat.account\_category\_name = 'Debt'

group by users.users\_id

go

select \* from user\_debt order by current\_total\_debt desc

go

/\*

Return a table of total bank balances by user, using the

current bank accounts

\*/

if object\_id('dbo.user\_bank') is not null

drop view dbo.user\_bank

go

create view user\_bank as

select

users.users\_id,

sum(latest\_accounts.balance) as current\_total\_bank\_balances

from

accounts acct

join

users on acct.users\_id = users.users\_id

join

(

select

account\_id,

balance,

account\_update\_date

from

dbo.current\_accounts

) as latest\_accounts on latest\_accounts.account\_id = acct.account\_id

join

account\_type actype on acct.account\_type\_id = actype.account\_type\_id

join

account\_category act\_cat on actype.account\_category\_id = act\_cat.account\_category\_id

where

act\_cat.account\_category\_name = 'Bank'

group by users.users\_id

go

select \* from dbo.user\_bank

/\*

Return a table of total investment balances by user, using the

current investment accounts

\*/

if object\_id('dbo.user\_investment') is not null

drop view dbo.user\_investment

go

create view user\_investment as

select

users.users\_id,

sum(latest\_accounts.balance) as current\_total\_investment\_balances

from

accounts acct

join

users on acct.users\_id = users.users\_id

join

(

select

account\_id,

balance,

account\_update\_date

from

dbo.current\_accounts

) as latest\_accounts on latest\_accounts.account\_id = acct.account\_id

join

account\_type actype on acct.account\_type\_id = actype.account\_type\_id

join

account\_category act\_cat on actype.account\_category\_id = act\_cat.account\_category\_id

where

act\_cat.account\_category\_name = 'Investment'

group by users.users\_id

go

select \* from dbo.user\_investment

select

users.users\_id

from

accounts acct

right join

users on acct.users\_id = users.users\_id

select

bank.users\_id,

bank.current\_total\_bank\_balances,

debt.current\_total\_debt,

inv.current\_total\_investment\_balances

from

dbo.user\_bank bank

join

dbo.user\_debt debt on bank.users\_id = debt.users\_id

join

dbo.user\_investment inv on bank.users\_id = inv.users\_id

select

bank.users\_id,

bank.current\_total\_bank\_balances,

debt.current\_total\_debt,

inv.current\_total\_investment\_balances

from

dbo.user\_bank bank

full outer join

dbo.user\_debt debt on bank.users\_id = debt.users\_id

full outer join

dbo.user\_investment inv on bank.users\_id = inv.users\_id

-- user incomes

if object\_id('dbo.user\_incomes') is not null

drop view dbo.user\_incomes

go

create view user\_incomes as

select

u.users\_id,

cf.cash\_flow\_id,

latest\_income.discretionary\_monthly\_income

from

cash\_flow cf

join

users u on cf.users\_id = u.users\_id

join

(

select

cash\_flow\_id,

discretionary\_monthly\_income

from

dbo.current\_cash\_flows

) as latest\_income on cf.cash\_flow\_id = latest\_income.cash\_flow\_id

go

select \* from user\_incomes

go

if object\_id('dbo.debt\_to\_income\_ratios') is not null

drop view dbo.debt\_to\_income\_ratios

go

create view debt\_to\_income\_ratios as

select

debt.users\_id,

round(debt.current\_total\_debt/(inc.discretionary\_monthly\_income \*12),2) as debt\_to\_income\_ratio

from

dbo.user\_debt debt

join

dbo.user\_incomes inc on debt.users\_id = inc.users\_id

go

select \* from debt\_to\_income\_ratios order by debt\_to\_income\_ratio desc

### Functions

-- Function to return the Debt to Income ratio for users

-- get the latest income for each user with a cash\_flow

select

u.users\_id,

cf.cash\_flow\_id,

latest\_income.discretionary\_monthly\_income

from

cash\_flow cf

join

users u on cf.users\_id = u.users\_id

join

(

select

cash\_flow\_id,

discretionary\_monthly\_income

from

dbo.current\_cash\_flows

) as latest\_income on cf.cash\_flow\_id = latest\_income.cash\_flow\_id

go

if object\_id('dbo.user\_debt\_to\_income') is not null

drop function dbo.user\_debt\_to\_income

go

/\*

Find the debt to income ratio of every user

with debt accounts and a cash flow in the

database.

\*/

create function dbo.user\_debt\_to\_income(@userID int)

returns decimal(6,2) as

begin

declare @returnVal decimal(6,2)

/\*

return a view that calculates the debt to income

ratio for each user. Find only ratio given for the

passed userID.

\*/

select

@returnVal = debt\_to\_income\_ratio

from

dbo.debt\_to\_income\_ratios

where users\_id = @userID

return @returnVal

end

go

-- Run the function for all users

select

users\_id,

first\_name,

last\_name,

dbo.user\_debt\_to\_income(users\_id) as debt\_to\_income\_ratio

from

users

order by debt\_to\_income\_ratio desc

go

if object\_id('dbo.spend\_ratio') is not null

drop function dbo.spend\_ratio

go

create function dbo.spend\_ratio(@userID int)

returns decimal(6,2) as

begin

declare @returnVal decimal(6,2)

select

@returnVal = round(average\_monthly\_spending / discretionary\_monthly\_income, 2)\*100

from

dbo.current\_cash\_flows curr\_cf

join

cash\_flow on curr\_cf.cash\_flow\_id = cash\_flow.cash\_flow\_id

where cash\_flow.users\_id = @userID

return @returnVal

end

go

select

\*,

dbo.spend\_ratio(users\_id) as spending\_to\_income\_percent

from

users

order by spending\_to\_income\_percent desc

### What is my debt to income ratio?

Two ways to achieve

create view debt\_to\_income\_ratios as

select

debt.users\_id,

round(debt.current\_total\_debt/(inc.discretionary\_monthly\_income \*12),2) as debt\_to\_income\_ratio

from

dbo.user\_debt debt

join

dbo.user\_incomes inc on debt.users\_id = inc.users\_id

go

if object\_id('dbo.user\_debt\_to\_income') is not null

drop function dbo.user\_debt\_to\_income

go

/\*

Find the debt to income ratio of every user

with debt accounts and a cash flow in the

database.

\*/

create function dbo.user\_debt\_to\_income(@userID int)

returns decimal(6,2) as

begin

declare @returnVal decimal(6,2)

/\*

return a view that calculates the debt to income

ratio for each user. Find only ratio given for the

passed userID.

\*/

select

@returnVal = debt\_to\_income\_ratio

from

dbo.debt\_to\_income\_ratios

where users\_id = @userID

return @returnVal

end

go

-- Run the function for all users

select

users\_id,

first\_name,

last\_name,

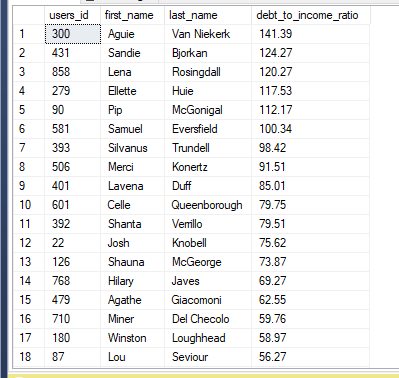
dbo.user\_debt\_to\_income(users\_id) as debt\_to\_income\_ratio

from

users

order by debt\_to\_income\_ratio desc

go



### What percent of my monthly income am I spending?

create function dbo.spend\_ratio(@userID int)

returns decimal(6,2) as

begin

declare @returnVal decimal(6,2)

select

@returnVal = round(average\_monthly\_spending / discretionary\_monthly\_income, 2)\*100

from

dbo.current\_cash\_flows curr\_cf

join

cash\_flow on curr\_cf.cash\_flow\_id = cash\_flow.cash\_flow\_id

where cash\_flow.users\_id = @userID

return @returnVal

end

go

select

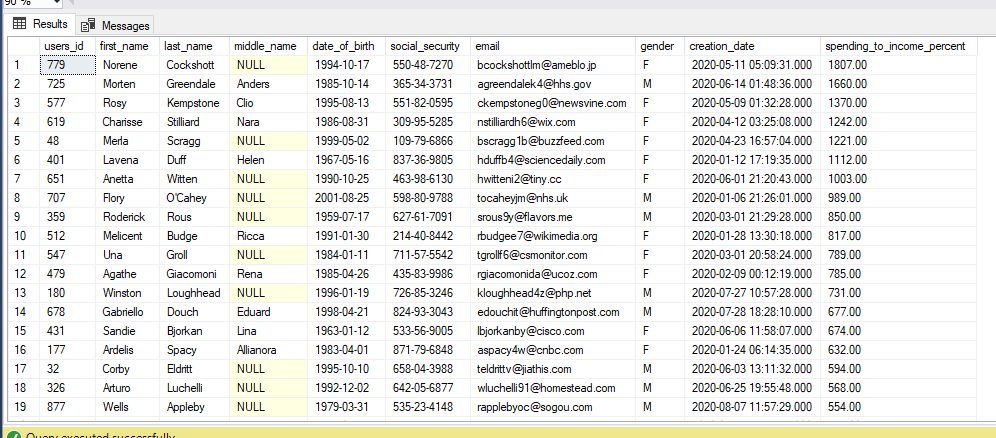
\*,

dbo.spend\_ratio(users\_id) as spending\_to\_income\_percent

from

users

order by spending\_to\_income\_percent desc



### How has spending changed over time?

Used the below query in python and looked at user 650’s spending behavior over time.

select

\*,

row\_number() over (partition by cash\_flow\_updates.cash\_flow\_id order by

cash\_flow\_updates.cash\_flow\_update\_date desc) as RowNum

from

cash\_flow\_updates

join

cash\_flow on cash\_flow.cash\_flow\_id = cash\_flow\_updates.cash\_flow\_id

A close up of a map

Description automatically generated

### How can I create new users?

-- Create a new user and enter their login info into the user\_login table

create procedure new\_user(@fname varchar(50), @lname varchar(50), @mname varchar(50) = Null,

@dob date, @ss varchar(11), @email varchar(150), @gender char(1),

@pass varchar(25))

as

begin

-- insert new user into the users table

insert into users

(first\_name, last\_name, middle\_name, date\_of\_birth, social\_security, email, gender)

values

(@fname, @lname, @mname, @dob, @ss, @email, @gender)

-- insert login information into user\_login

insert into user\_login

(users\_id, pass)

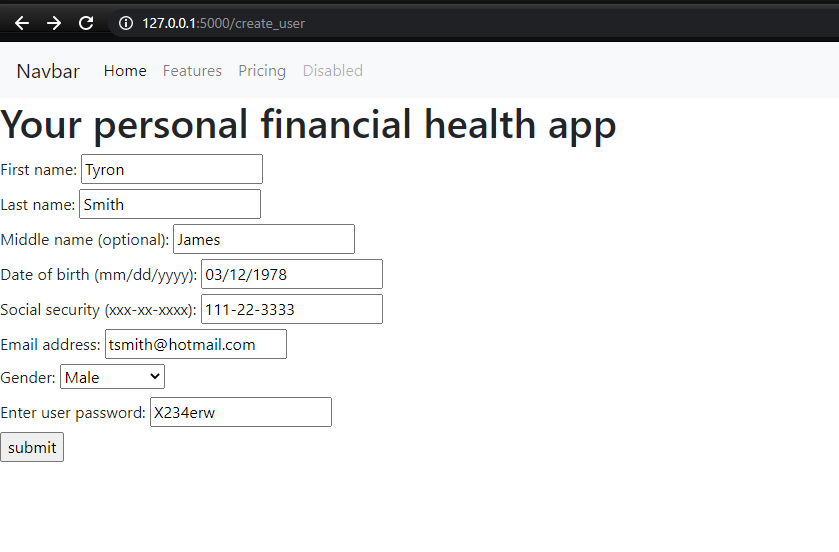
values

(@@identity, @pass)

end

go

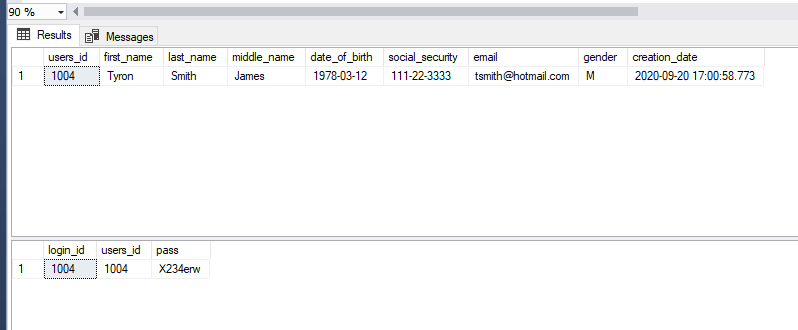
Created a website gui using python and html to enter information into the procedure:



-- Check results after creting a user through the flask based app via python

select top 1 \* from users order by users\_id desc

select \* from user\_login where users\_id = (select top 1 users\_id from users order by users\_id desc)



### How can I add a new account?

-- create an account for an existing user

create procedure create\_account(@users\_id int, @acc\_type\_name varchar(30), @bal decimal(9,2), @rate decimal(4,2) = Null,

@rate\_type varchar(8) = Null, @term int = Null)

as

begin

-- begin transaction

begin try

begin transaction

-- declare account id to be defined later

declare @acct\_id int

-- return the account type id to enter into the accounts table

declare @acc\_type int = (select account\_type\_id

from account\_type

where account\_type\_name = @acc\_type\_name)

-- create the account in the accounts table

insert into accounts

(account\_type\_id, users\_id)

values

(@acc\_type, @users\_id)

-- set the row identity to account is

set @acct\_id = @@identity

-- execute the account update procedure to enter the account info

exec update\_account @acct\_id=@acct\_id, @bal=@bal, @rate=@rate,

@rate\_type=@rate\_type, @term=@term

if @@TRANCOUNT > 0

commit

end try

begin catch

if @@TRANCOUNT > 0

rollback

SELECT

ERROR\_NUMBER() AS ErrorNumber,

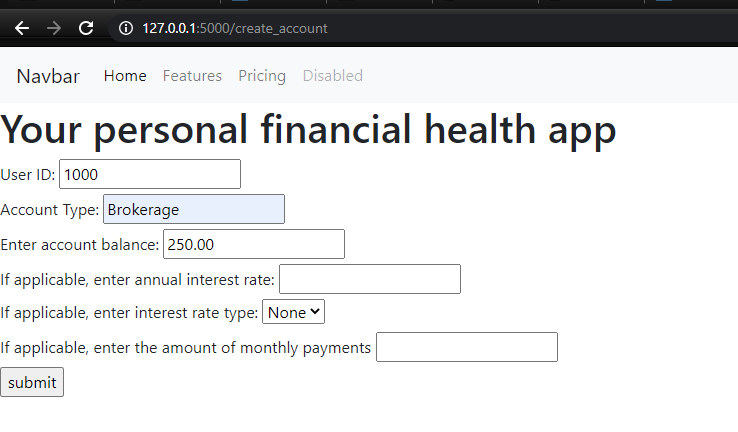
ERROR\_MESSAGE() AS ErrorMessage

end catch

end

go

Created a website gui using python and html to enter new accounts into the procedure:



-- Check results after entering an account through the flask based app via python

select top 1 \* from accounts order by account\_id desc

select

\*

from

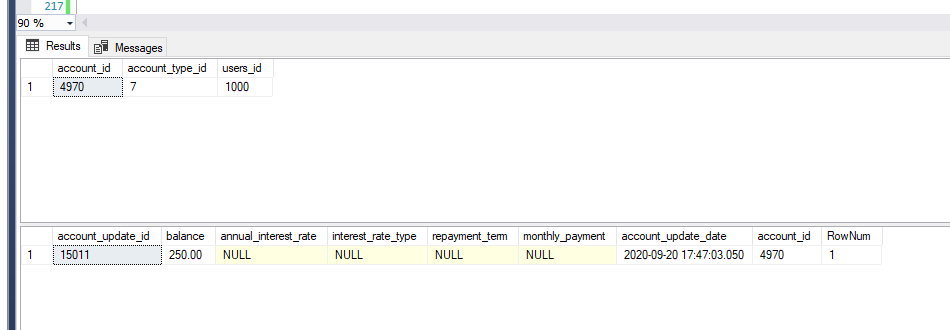
current\_accounts

where account\_id in (select top 1

account\_id

from accounts

order by account\_id desc)



### Can I update my account balance?

-- Create procedure to update accounts

create procedure update\_account(@acct\_id int, @bal decimal(9,2), @rate decimal(4,2) = Null, @rate\_type varchar(8) = Null,

@term int = Null)

as

begin

-- update the information provided

insert into account\_updates

(account\_id ,balance, annual\_interest\_rate, interest\_rate\_type,

repayment\_term)

values

(@acct\_id, @bal, @rate, @rate\_type, @term)

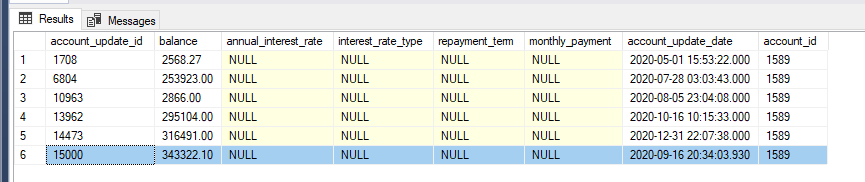
end

go

exec update\_account @acct\_id=1589, @bal=343322.1

select \* from account\_updates where account\_id = 1589

go



### Can I update my cash flow information?

-- Create procedure to update cash flow of a user

create procedure update\_cash\_flow(@cf\_id int, @income decimal(8,2), @spend decimal(8,2))

as

begin

-- insert new cash flow update

insert into cash\_flow\_updates

(cash\_flow\_id, discretionary\_monthly\_income, average\_monthly\_spending)

values

(@cf\_id, @income, @spend)

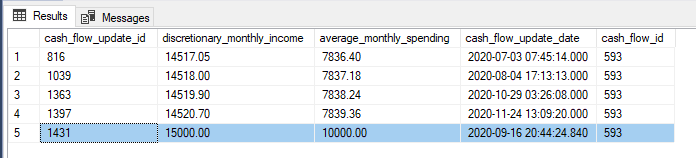
end

go

exec update\_cash\_flow 593, 15000, 10000

select \* from cash\_flow\_updates where cash\_flow\_id = 593

go



### Can I change my password?

-- update a user's password

create procedure update\_password(@user\_id int, @pass varchar(25))

as

begin

-- update the user's login password

update

user\_login

set

pass = @pass

where

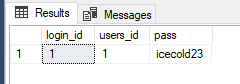
users\_id = @user\_id

end

go

exec update\_password 1, icecold23

select \* from user\_login where users\_id = 1



## Summary

A database is only as good as its design, which is entirely based on the problems the database is supposed to solve. The previous statement is what I takeaway the most from this project, and I believe it will help me in my future data endeavors. It was a great experience to be able to think of how an end to end solution could be developed to help people better understand their financial health. Being able to write the SQL code, insert the data, and interact with the data through python scripts and a python-built website was powerful and exciting. This project was great for my technical development, and something I will leverage in the future.