Daniel Piston

IST 659

IST 659 Database project report

Onward to Opportunity Database

Contents

[Summary 2](#_Toc87851472)

[Stakeholders 2](#_Toc87851473)

[Business Rules 2](#_Toc87851474)

[Glossary 3](#_Toc87851475)

[Data Questions 3](#_Toc87851476)

[Conceptual Model 5](#_Toc87851477)

[Logical Model 6](#_Toc87851478)

[Physical Database Design 7](#_Toc87851479)

[User Interface 20](#_Toc87851480)

[Database Updates 21](#_Toc87851481)

[Sample Reporting 21](#_Toc87851482)

[Reflection 24](#_Toc87851483)

[Final Summary 25](#_Toc87851484)

## Summary

The focus for this project will be to design and create a database for a department of Syracuse University called the Institute for Veterans and Military Families (IVMF). The IVMF is a research institute, but also provides the military connection community with programs designed to aid in providing education and training courses on entrepreneurship and career transition. The main program for career transition is called the Onward to Opportunity program (O2O). This program provides military connected individuals the opportunity to prepare for an industry recognized certification and the coursework required to prepare the individual. The data source for this project will be a SQL server database that stores the tables from Salesforce.

## Stakeholders

The stakeholders for this project are primarily focused on the business intelligence team (reporting) and the program leadership that consume on a regular basis in a detailed manner to support business decisions and reporting to external stakeholders. Additional internal stakeholders include organization leadership and the development (fundraising) team, which typically focus on high level data and interesting insights to gain attention. Implementing the database will be used to increase efficiency while also aiding in data cleaning to ease of the reporting burden. A goal of the BI team is also to increase the building and use of visualizations and dashboards, which the database will enhance.

## Business Rules

1. A contact is only created if they completed an application to be considered for the program.
2. Once a contact has completed an application, an assessment is automatically triggered to complete.
3. Each program enrollment has a corresponding application.
4. A participant is considered enrolled if they utilize a license to start their coursework.
5. Each participant is assigned a cohort if enrolled.
6. Each participant may enroll in multiple courses within a single program instance or cohort.
7. A participant can re-enroll if they never used a license to begin their coursework.
8. Each participant is assigned a single advisor throughout their course.

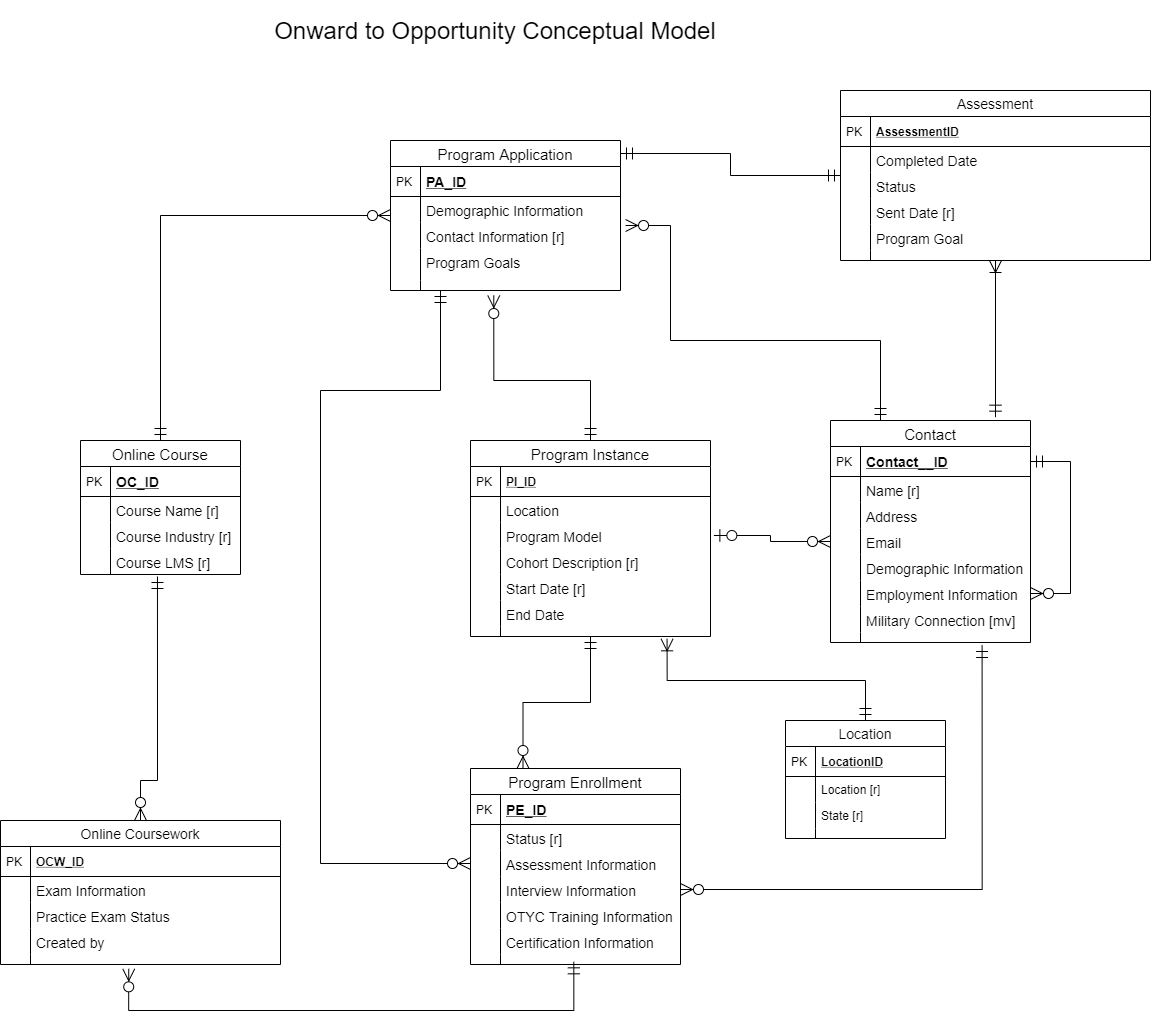
## Glossary

1. Participant: Anyone that applies to become enrolled in the program or enrolls in the program.
2. Application: The initial form that is completed to be considered for the program.
3. Enrollment: A record for each course a participant starts in the program.
4. Cohort: The class that a participant is part of. Typically annotated by location and time period that they started the program.
5. Program Instance: A unique number that denotes which cohort a participant enrolled in. Also denotes a table that contains all information on the specific cohort.
6. License: Refers to a Skillsoft or other LMS license that provides an online platform for certification coursework.

## Data Questions

1. How many participants have enrolled in the program during a specific period or all time?
2. How many participants have enrolled based on location?
3. How many participants have achieved a positive outcome (completed coursework, was hired etc.) during a specific period or all time?
4. What is the rate of completion for program participants?
5. Where are the program graduates currently located?
6. What are the demographic data for the participants that have enrolled?
7. Where companies do program graduates work for?
8. What predicts a positive outcome for a currently enrolled participant?
9. How many courses has one a participant taken in the program?

## Conceptual Model



## Logical Model

## Physical Database Design

The database design below was create using Microsoft SQL Server Management Studio. The data that was inserted into the created tables below came from another server. We used our current user SAS interface with the first database to act as bridge in which to rename and clean some of the data, and create new tables. The data was then inserted from SAS using SAS SQL syntax, into the tables created in Microsoft SQL Server Studio.

Steps to create database:

1. Create fresh table in new database with correct names and lengths
2. Extract specific columns from current back up database into SAS
3. Rename, clean and create new tables as needed in “Transfer” tables
4. Insert cleaned and renamed data into new database from SAS

--O2O Database table creation

--Created all tables in SQL

--Created transfer tables in SAS in which variables were renamed from data pulled from Salesforce database

--Inserted from SAS transfer tables into created SQL tables

--Each table has corresponding insert SAS syntax

--Table 1: Locations

--Create Locations table template for Location data

create table Locations

(

Location\_ID varchar(18) not null

,Location\_Name varchar(60)

,Location\_State varchar(20)

,Location\_Last\_Modified\_Date datetime not null

,constraint Locations\_PK primary key (Location\_ID)

)

select \* from Locations

/\*

--Transfer table to rename variables and clean

proc sql;

create table Locations\_Transfers as

select

ID as Location\_ID

,Namex as Location\_Name

,statex as Location\_State

,lastmodifieddate as Location\_Last\_Modified\_Date

from DBSFPRD.sf\_location

; quit;

--Location insert syntax in SAS

proc sql;

insert into DBO2OTST.Locations

select \*

from Locations\_Transfers

; quit;

\*/

--------------------------------------------------------------------------

--Table 2: Online\_Courses

--Create Online\_Courses table template for OC data

create table Online\_Courses

(

Online\_Course\_ID char(18) not null

,OC\_Condensed\_Industry varchar(50)

,OC\_Industry varchar(50)

,OC\_LMS varchar(30) not null

,OC\_Online\_Course varchar(80) not null

,OC\_Current\_Status char(5) not null

,constraint Online\_Courses\_PK primary key (Online\_Course\_ID)

)

select \* from Online\_Courses

/\* Online course transfer table creation in SAS to clean and change variable names

proc sql;

create table Online\_Courses\_Transfer as

select

ID as Online\_Course\_ID

,Condensed\_industry as OC\_Condensed\_Industry

,industry as OC\_Industry

,LMS as OC\_LMS

,namex as OC\_Online\_Course

,currentx as OC\_Current\_Status

from dbsfprd.sf\_online\_course

;quit;\*/

--Online\_Courses insert SAS syntax

/\*

proc sql;

insert into DBO2OTST.Online\_Courses

select \*

from Online\_Courses\_Transfer

; quit;\*/

---------------------------------------------------------------------------

--Contacts Look-Up Tables

drop table Branches

--Create Branches look up table

create table Branches

(

Branch\_ID int primary key

, Branch varchar(20) not null

)

insert into Branches

values (1, 'Air Force')

,(2, 'Army')

,(3, 'Coast Guard')

,(4, 'Marine Corps')

,(5, 'Navy')

,(10, 'Air Force Reserve')

,(11, 'Army Reserve')

,(12, 'Coast Guard Reserve')

,(13, 'Marine Corps Reserve')

,(14, 'Navy Reserve')

,(20, 'Air National Guard')

,(21, 'Army National Guard')

,(22, 'National Guard')

select \* from Branches

--Create Military\_Connection ID table

create table Military\_Connections

(

Military\_Connection\_ID int identity primary key

, Military\_Connection varchar(40)

)

insert into Military\_Connections

values ('Service Member')

,('Activated National Guard/Reserves')

,('Non-Activated National Guard/Reserves')

,('Spouse')

,('Veteran')

,('Undisclosed/Missing')

select \* from Military\_Connections

drop table Military\_Connections

---------------------------------------------------------------------------

--Table 3: Contacts

create table Contacts

(

Contact\_ID char(18)

,Contact\_Created\_Date datetime not null

,Contact\_Source varchar(150)

,Contact\_First\_Name varchar(50) not null

,Contact\_Last\_Name varchar(100)not null

,Contact\_Email varchar(100)

,Contact\_Phone varchar(50)

,Contact\_Home\_Phone varchar(50)

,Contact\_Cell\_Phone varchar(50)

,Contact\_Ethnicity varchar(50)

,Contact\_Gender varchar(30)

,Contact\_Level\_of\_Ed varchar(75)

,Contact\_Active\_Duty\_Status varchar(75)

,Contact\_Active\_Duty varchar(5)

,Contact\_Birthdate datetime

,Contact\_Rank char(25)

,Contact\_Terminal\_Leave\_Date datetime

,Contact\_First\_Interview datetime

,Contact\_Date\_of\_Seperation datetime

,Contact\_Do\_Not\_Mail char(5)

,Contact\_Duty\_Station varchar(250)

,Contact\_Mailing\_State varchar(60)

,Contact\_Zip char(20)

,Contact\_Marital\_Status varchar(25)

,Contact\_Military\_Specialty varchar(300)

,Contact\_Employment\_status varchar(200)

,Contact\_Employer varchar(150)

,Contact\_Position varchar(150)

,Contact\_Employer\_Instituion varchar(200)

,Contact\_Employer\_Partner varchar(50)

,Contact\_Emp\_Start\_Date datetime

,Contact\_Employment\_Location varchar(200)

,Contact\_IVMF\_Start\_Date datetime

,Contact\_Employment\_Zip char(12)

,Contact\_Employment\_Notes varchar(1500)

,Contact\_Outcome\_Type varchar(10)

,Contact\_Placed\_By varchar(5)

,Contact\_Employment\_Position varchar(150)

,Contact\_IVMF\_Salary\_Range varchar(20)

,Contact\_Type\_of\_Degree varchar(75)

,Contact\_Confirmed\_Hire\_Date datetime

,Contact\_Hired\_Location varchar(150)

,Contact\_Hired\_Emp\_Part char(5)

,Contact\_Discharge\_Status varchar(60)

,Contact\_Salary\_Range varchar(20)

,Contact\_Matched\_Emp\_Part varchar(90)

,HHUSA\_Employ\_Start\_Date varchar(30)

,HHUSA\_Color\_Status char(10)

,HHUSA\_Status\_Notes varchar(200)

,Contact\_Hired\_Zip char(10)

,Contact\_Emp\_Outcome\_Type char(20)

,Contact\_HHUSA\_Opt\_Out varchar(60)

,Contact\_Sent\_to\_HHUSA datetime

,constraint Contacts\_PK primary key (Contact\_ID)

)

select \* from contacts

/\* Transfer table syntax to rename columns and clean if needed

proc sql ;

create table Contacts\_Transfer as

select

ID as Contact\_ID

,createddate as Contact\_Created\_Date

,contactsource as Contact\_Source

,firstname as Contact\_First\_Name

,lastname as Contact\_Last\_Name

,email as Contact\_Email

,phone as Contact\_Phone

,homephone as Contact\_Home\_Phone

,mobilephone as Contact\_Cell\_Phone

,ethnicity as Contact\_Ethnicity

,gender as Contact\_Gender

,level\_of\_education as Contact\_Level\_of\_Ed

,active\_duty\_status as Contact\_Active\_Duty\_Status

,active\_duty as Contact\_Active\_Duty

,birthdate as Contact\_Birthdate

,rank\_at\_separation as Contact\_Rank

,terminal\_leave\_date as Contact\_Terminal\_Leave\_Date

,date\_of\_first\_interview as Contact\_First\_Interview

,date\_of\_separation as Contact\_Date\_of\_Seperation

,DO\_NOT\_MAIL as Contact\_Do\_Not\_Mail

,duty\_station as Contact\_Duty\_Station

,mailingstate as Contact\_Mailing\_State

,mailingpostalcode as Contact\_Zip

,marital\_status as Contact\_Marital\_Status

,military\_specialty as Contact\_Military\_Specialty

/\*Unsure where these come from\*/

,Employment\_status as Contact\_Employment\_Status

,Employer as Contact\_Employer

,positionx as Contact\_Position

/\*IVMF fields\*/

,EMPLOYER\_INSTITUTION as Contact\_Employer\_Instituion

,EMPLOYER\_PARTNER as Contact\_Employer\_Partner

,EMPLOYMENT\_START\_DATE as Contact\_Emp\_Start\_Date

,EMPLOYMENT\_LOCATION as Contact\_Employment\_Location

,IVMF\_HIRED\_START\_DATE as Contact\_IVMF\_Start\_Date

,EMPLOYMENT\_ZIP\_CODE as Contact\_Employment\_Zip

,EMPLOYMENT\_NOTES as Contact\_Employment\_Notes

,OUTCOME\_TYPE as Contact\_Outcome\_Type

,PLACED\_BY as Contact\_Placed\_By

,EMPLOYMENT\_POSITION as Contact\_Employment\_Position

,EMPLOYMENT\_SALARY\_RANGE as Contact\_IVMF\_Salary\_Range

,TYPE\_OF\_DEGREE as Contact\_Type\_of\_Degree

/\*HHUSA Fields\*/

,CONFIRMED\_HIRE\_DATE as Contact\_Confirmed\_Hire\_Date

,HIRED\_LOCATION as Contact\_Hired\_Location

,HIRED\_BY\_PARTNER\_EMPLOYER as Contact\_Hired\_Emp\_Part

,DISCHARGE\_STATUS as Contact\_Discharge\_Status

,SALARY\_RANGE as Contact\_Salary\_Range

,MATCHED\_PARTNER\_EMPLOYER as Contact\_Matched\_Emp\_Part

,HHUSA\_EMPLOYMENT\_START\_DATE as HHUSA\_Emp\_Start\_Date

,HHUSA\_COLOR\_STATUS as HHUSA\_Color\_Status

,HIRE\_MATCH\_STATUS\_NOTES as Contact\_Status\_Notes

,HIRED\_ZIP\_CODE as Contact\_Hired\_Zip

,employment\_outcome\_type as Contact\_Emp\_Outcome\_Type

,hhusa\_opt\_out as Contact\_HHUSA\_Opt\_Out

,sent\_to\_hhusa as Contact\_Sent\_to\_HHUSA

from dbsfprd.sf\_contact

;

quit; \*/

/\* Insert syntax

proc sql;

insert into DBO2OTST.Contacts c

select \*

from Contacts\_Transfer

where not exists (select Contact\_ID from DBO2OTST.Contacts d where c.Contact\_ID = d.Contact\_ID)

; quit;\*/

----------------------------------------------------------------------------

--Contacts Bridge tables to solve many to many relationship between contact table and branches table

create table Branches\_Bridge

(

Branch\_Bridge\_ID int identity not null

,Contact\_ID char(18) not null

,Branch\_ID int not null

,constraint Branches\_PK primary key (Branch\_Bridge\_ID)

,constraint Branches\_Bridge\_FK1 foreign key (Contact\_ID) references Contacts

,constraint Branches\_Bridge\_FK2 foreign key (Branch\_ID) references Branches

)

--Create Bridge Transfer table without any constraints to bring data in from SAS

create table Branches\_Bridge\_Transfer

(Contact\_ID char(18) not null

,Branch\_ID int not null)

select \* from Branches\_Bridge\_Transfer

--Insert from Transfer table to allow for identity function to work properly

SET IDENTITY\_INSERT dbo.Branches\_Bridge ON;

insert into Branches\_Bridge

(Contact\_ID, Branch\_ID)

select Contact\_ID, Branch\_ID

from Branches\_Bridge\_Transfer

SET IDENTITY\_INSERT dbo.Branches\_Bridge OFF;

select \* from Branches\_Bridge

/\* Branches SAS syntax to clean and inset into Branches\_Bridge\_Transfer table

proc sql;

create table Contact\_Branch\_Transfer as

select

ID as Contact\_ID

,BRANCH\_OF\_SERVICE as Contact\_Branch

,strip(scan(BRANCH\_OF\_SERVICE,1,',;')) as Branch\_1

,strip(scan(BRANCH\_OF\_SERVICE,2,',;')) as Branch\_2

,strip(scan(BRANCH\_OF\_SERVICE,3,',;')) as Branch\_3

,strip(scan(BRANCH\_OF\_SERVICE,4,',;')) as Branch\_4

,strip(scan(BRANCH\_OF\_SERVICE,5,',;')) as Branch\_5

from dbsfprd.sf\_contact

where find(contactsource,'O2O','i') > 0 or find(contactsource,'vctp','i') > 0

order by Contact\_ID

;

quit;

proc transpose data=Contact\_Branch\_Transfer out=Contact\_Branch\_Table (rename=(col1=Branch) drop=\_Name\_);

by Contact\_ID;

var Branch\_1 Branch\_2 Branch\_3 Branch\_4 Branch\_5;

run;

proc sql;

create table Branches\_Bridge\_Transfer as

select Contact\_ID,

case Branch

when 'Air Force' then 1

when 'Army' then 2

when 'Coast Guard' then 3

when 'Marine Corps' then 4

when 'Navy' then 5

when 'Air Force Reserve' then 10

when 'Army Reserve' then 11

when 'Coast Guard Reserve' then 12

when 'Marine Corps Reserve' then 13

when 'Navy Reserve' then 14

when 'Air National Guard' then 20

when 'Army National Guard' then 21

when 'National Guard' then 22

else 99 end as Branch\_ID

from (

select Contact\_ID,

case Branch

when 'Marines' then 'Marine Corps'

else Branch end as Branch

from Contact\_Branch\_Table

where Branch ne '')

; quit;

\*/

---------------------------------------------------------------------------------

--Create bridge table to solve many to many relationship between contact table and military connection table

create table Military\_Connections\_Bridge

(

MilConnect\_Bridge\_ID int identity not null

,Contact\_ID char(18) not null

,Military\_Connection\_ID int not null

,constraint MilConnect\_Bridge\_PK primary key (MilConnect\_Bridge\_ID)

,constraint MilConnect\_Bridge\_FK1 foreign key (Contact\_ID) references Contacts

,constraint MilConnect\_Bridge\_FK2 foreign key (Military\_Connection\_ID) references Military\_Connections

)

--select \* from Military\_Connections\_Bridge

--Create transfer table without constraints to transfer clean data from SAS

create table MilConnect\_Bridge\_Transfer

(Contact\_ID char(18) not null

,Military\_Connection\_ID int not null)

--select \* from MilConnect\_Bridge\_Transfer

--Insert data from transfer table into bridge table with contraints

SET IDENTITY\_INSERT dbo.Military\_Connections\_Bridge ON;

insert into Military\_Connections\_Bridge

(Contact\_ID, Military\_Connection\_ID)

select Contact\_ID, Military\_Connection\_ID

from MilConnect\_Bridge\_Transfer

SET IDENTITY\_INSERT dbo.Military\_Connections\_Bridge OFF;

/\* SAS syntaxt to clean military connection data and insert into military connection bridge table

proc sql;

create table Contact\_MilConnect\_Transfer as

select

ID as Contact\_ID

,military\_connection as Contact\_Military\_Connection

,strip(scan(military\_connection,1,',;/')) as Military\_Connection\_1

,strip(scan(military\_connection,2,',;/')) as Military\_Connection\_2

,strip(scan(military\_connection,3,',;/')) as Military\_Connection\_3

,strip(scan(military\_connection,4,',;/')) as Military\_Connection\_4

,strip(scan(military\_connection,5,',;/')) as Military\_Connection\_5

,strip(scan(military\_connection,6,',;/')) as Military\_Connection\_6

,strip(scan(military\_connection,7,',;/')) as Military\_Connection\_7

,strip(scan(military\_connection,8,',;/')) as Military\_Connection\_8

,strip(scan(military\_connection,9,',;/')) as Military\_Connection\_9

from dbsfprd.sf\_contact

where find(contactsource,'O2O','i') > 0 or find(contactsource,'vctp','i') > 0

order by Contact\_ID

;

quit;

proc transpose data=Contact\_MilConnect\_Transfer out=Military\_Connection\_Table (rename=(col1=Military\_Connection) drop=\_Name\_);

by Contact\_ID;

var Military\_Connection\_1 - Military\_Connection\_9;

where Contact\_Military\_Connection ne '';

run;

proc sql;

create table MilConnect\_Bridge\_Transfer as

select \*

from (

select Contact\_ID,

case Military\_Connection

when 'Active Duty' then 1

when 'Active duty' then 1

when 'Servicemember' then 1

when 'Activated National Guard' then 2

when 'Activated NationalGuard' then 2

when 'Activated Reserve' then 2

when 'National Guard' then 3

when 'Reserves' then 3

when 'Military Spouse' then 3

when 'Partner' then 3

when 'Spouse' then 3

when 'Spouse of a service member' then 3

when 'on Spouse of a service member' then 3

when 'I am a retired' then 4

when 'Veteran' then 4

when 'veteran' then 4

when 'Retiree' then 4

when 'separated service member' then 4

when 'Unclear - BSF Referral' then 5

when 'None of the above' then 5

end as Military\_Connection

from Military\_Connection\_Table

where Military\_Connection ne '')

; quit;

proc sql;

insert into DBO2OTST.MilConnect\_Bridge\_Transfer

select \*

from MilConnect\_Bridge\_Transfer

; quit;

\*/

----------------------------------------------------------------------------

--Table 3: Program\_Instances

create table Program\_Instances

(

Program\_Instance\_ID char(18) not null

,PI\_O2O\_Model varchar(12)

,PI\_Program char(12)

,PI\_Program\_Instance\_desc varchar(60)

,PI\_Number char(10)

,PI\_Cohort\_Name varchar(50)

,PI\_Start\_date datetime

,PI\_End\_Date datetime

,Location\_ID char (18)

,constraint Program\_Instance\_PK primary key (Program\_Instance\_ID)

,constraint Program\_Instance\_FK1 foreign key (Location\_ID) references Locations

)

select \* from Program\_Instances

/\*

--Look-up Table 1: PI\_Bridge

create table Program\_Instances\_Bridge

(

PI\_Bridge\_ID int identity

,Contact\_ID char(18)

,Program\_Instance\_ID char(18)

,constraint PI\_Bridge\_PK primary key (PI\_Bridge\_ID)

,constraint PI\_Bridge\_FK1 foreign key (Contact\_ID) references Contacts

,constraint PI\_Bridge\_FK2 foreign key (Program\_Instance\_ID) references Program\_Instances

)

insert into Program\_Instances\_Bridge

values (

--select \* from Program\_Instances\_Bridge

create table PI\_Bridge\_Transfer

( Contact\_ID char(18)

,Program\_Instance\_ID char(18))

select \* from PI\_Bridge\_Transfer

where Contact\_ID is null

SET IDENTITY\_INSERT dbo.Program\_Instances\_Bridge ON;

insert into Program\_Instances\_Bridge

(Contact\_ID, Program\_Instance\_ID)

select Contact\_ID, Program\_Instance\_ID

from PI\_Bridge\_Transfer

select \* from Program\_Instances\_Bridge\*/

---------------------------------------------------------------------------

--Table 4: Program\_Applications

--Program Applications

create table Program\_Applications

( Program\_Application\_ID char(18) primary key not null

,PA\_Application\_Date datetime

,PA\_Created\_Date datetime

,PA\_Application\_Number char (8)

,PA\_Program\_Name varchar(25)

,PA\_Heard\_About\_Through varchar(100)

,PA\_Online\_Course\_Preference varchar(100)

,PA\_Program\_Instance\_Desc varchar(50)

,PA\_Status varchar(50)

,PA\_Program\_Application\_Origin varchar(50)

,PA\_Program\_Completion\_Goal varchar(100)

,PA\_Live\_Within\_50m varchar(60)

,PA\_Cybervet\_Geographic\_Status varchar(40)

,PA\_O2O\_Career\_Track varchar(30)

,PA\_OTYC\_Start\_Date datetime

,PA\_Orientation\_Status char(8)

,PA\_Discharge\_Status char(21)

,PA\_Assessment\_Status char(23)

,PA\_Assessment\_Submitted\_Date datetime

,PA\_Withdraw\_Reason varchar(33)

,Contact\_ID char(18)

,Program\_Instance\_ID char(18)

,Online\_Course\_ID char(18)

,constraint Program\_Applications\_FK1 foreign key (Contact\_ID) references Contacts

,constraint Program\_Applications\_FK2 foreign key (Program\_Instance\_ID) references Program\_Instances

,constraint Program\_Applications\_FK3 foreign key (Online\_Course\_ID) references Online\_Courses

)

--Program\_Applications insert SAS syntax

/\*

Insert syntaxx only includes contacts that are in the contact table

Current data has rows for contacts not in the contact table, currently being cleaned at the source

proc sql;

insert into DBO2OTST.Program\_Applications

select \*

from Program\_Applications\_Transfer pa

where exists (select Contact\_ID from DBO2OTST.Contacts c where c.Contact\_ID = pa.Contact\_ID)

; quit;

\*/

------------------------------------------------------------------------------------------------------------

--Program Enrollments table

create table Program\_Enrollments

(

Program\_Enrollment\_ID char(18)

,PE\_Assessment\_Submitted\_Date datetime

,PE\_Created\_Date datetime

,PE\_Heard\_About\_Description char(300)

,PE\_Heard\_About\_From char(100)

,PE\_Heard\_About\_Program char(300)

,PE\_Orientation\_Status char(20)

,PE\_Orientation\_Date datetime

,PE\_Advisor char(20)

,PE\_Status char(20)

,Contact\_ID char(18)

,Program\_Instance\_ID char(18)

,Program\_Application\_ID char(18)

,constraint Program\_Enrollments\_PK primary key (Program\_Enrollment\_ID)

,constraint Program\_Enrollments\_FK1 foreign key (Contact\_ID) references Contacts

,constraint Program\_Enrollments\_FK2 foreign key (Program\_Instance\_ID) references Program\_Instances

,constraint Program\_Enrollments\_FK3 foreign key (Program\_Application\_ID) references Program\_Applications

)

/\*Problem with enrollments and applications constraints\*/

drop table Program\_Enrollments

select \* from Program\_Enrollments

/\*SAS program enrollment transfer table to clean and insert statement to only include

contacts in the contact table and applications in the application table

/\*Insert contents of transfer table into database table\*/

proc sql;

insert into DBO2OTST.Program\_Enrollments

select \*

from

select

ID as Program\_Enrollment\_ID

,assessment\_submitted\_date as PE\_Assessment\_Submitted\_Date

,createddate as PE\_Created\_Date

,HEARD\_ABOUT\_THE\_PROGRAM\_DESCRIPT as PE\_Heard\_About\_Description

,HEARD\_ABOUT\_THE\_PROGRAM\_FROM as PE\_Heard\_About\_From

,HEARD\_ABOUT\_THE\_PROGRAM as PE\_Heard\_About\_Program

,orientation\_status as PE\_Orientation\_Status

,orientation\_date as PE\_Orientation\_Date

,Advisor as PE\_Advisor

,Statusx as PE\_Status

,Contact as Contact\_ID

,Program\_Instance as Program\_Instance\_ID

,Program\_Application as Program\_Application\_ID

from dbsfprd.sf\_program\_instance\_attended ia

where exists (select Contact\_ID from DBO2OTST.Contacts c where c.Contact\_ID = ia.contact) and

exists (select Program\_Application\_ID from DBO2OTST.Program\_Applications d where d.Program\_Application\_ID = ia.program\_application)

;

quit;\*/

----------------------------------------------------------------------------------------

--Create table for online coursework data

create table Online\_Courseworks

(

Online\_Coursework\_ID char(18)

,OCW\_Advisor char(50)

,OCW\_Date\_Exam\_Fee\_Paid datetime

,OCW\_Dollar\_Amount\_Paid decimal

,OCW\_Exam\_Date datetime

,OCW\_Exam\_Fee\_Paid char(5)

,OCW\_Exam\_Passed char(5)

,OCW\_Extension\_Granted char(5)

,OCW\_Practice\_Exam\_Status char(20)

,OCW\_Primary\_Course char(5)

,OCW\_Online\_Course\_Number char(10)

,Online\_Course\_ID char(18)

,Program\_Enrollment\_ID char(18)

/\*Constraints for Online\_Courseworks table\*/

,constraint Online\_Courseworks\_PK1 primary key (Online\_Coursework\_ID)

,constraint Online\_Courseworks\_FK1 foreign key (Program\_Enrollment\_ID) references Program\_Enrollments

,constraint Online\_Courseworks\_FK2 foreign key (Online\_Course\_ID) references Online\_Courses

)

select \* from Online\_Courseworks

/\*Insert transfer table contents to database table

proc sql;

insert into DBO2OTST.Online\_Courseworks

select \*

from

( select

ID as Online\_Coursework\_ID

,advisor as OCW\_Advisor

,date\_exam\_fee\_paid as OCW\_Date\_Exam\_Fee\_Paid

,dollar\_amount\_paid as OCW\_Dollar\_Amount\_Paid

,exam\_date as OCW\_Exam\_Date

,exam\_fee\_paid as OCW\_Exam\_Fee\_Paid

,exam\_passed as OCW\_Exam\_Passed

,extension\_granted as OCW\_Extension\_Granted

,practice\_exam\_status as OCW\_Practice\_Exam\_Status

,primaryx as OCW\_Primary\_Course

,namex as OCW\_Online\_Course\_Number

,ONLINE\_COURSE as Online\_Course\_ID

,PROGRAM\_ENROLLMENT as Program\_Enrollment\_ID

from dbsfprd.sf\_online\_coursework ocw

where exists (select Program\_Enrollment\_ID from DBO2OTST.Program\_Enrollments c where c.Program\_Enrollment\_ID = ocw.Program\_Enrollment))

; quit;\*/

-------------------------------------------------------------------

--Assessments table creation

create table Assessments

(

Assessment\_ID char(18)

,Assessment\_Source char(40)

,Assessment\_Completion\_Date datetime

,Assessment\_Completion\_Status char(40)

,Assessment\_Created\_Date datetime

,Assessment\_Program\_Goal CHAR(150)

,Contact\_ID char(18)

,Program\_Application\_ID char(18)

,constraint Assessments\_PK primary key (Assessment\_ID)

,constraint Assessments\_FK1 foreign key (Contact\_ID) references Contacts

,constraint Assessments\_FK2 foreign key (Program\_Application\_ID) references Program\_Applications

)

--Insert statement from SAS

/\*

proc sql;

insert into dbo2otst.Assessments

select \*

from(

select

ID as Assessment\_ID

,Namex as Assessment\_Source

,survey\_completion\_date as Assessment\_Completion\_Date

,ASSESSMENT\_COMPLETION\_STATUS as Assessment\_Completion\_Status

,createddate as Assessment\_Created\_Date

,Program\_completion\_goal as Assessment\_Program\_Goal

,Contact as Contact\_ID

,Program\_Application as Program\_Application\_ID

from dbsfprd.sf\_assessment a

where exists (select Contact\_ID from DBO2OTST.Contacts c where c.Contact\_ID = a.contact) and

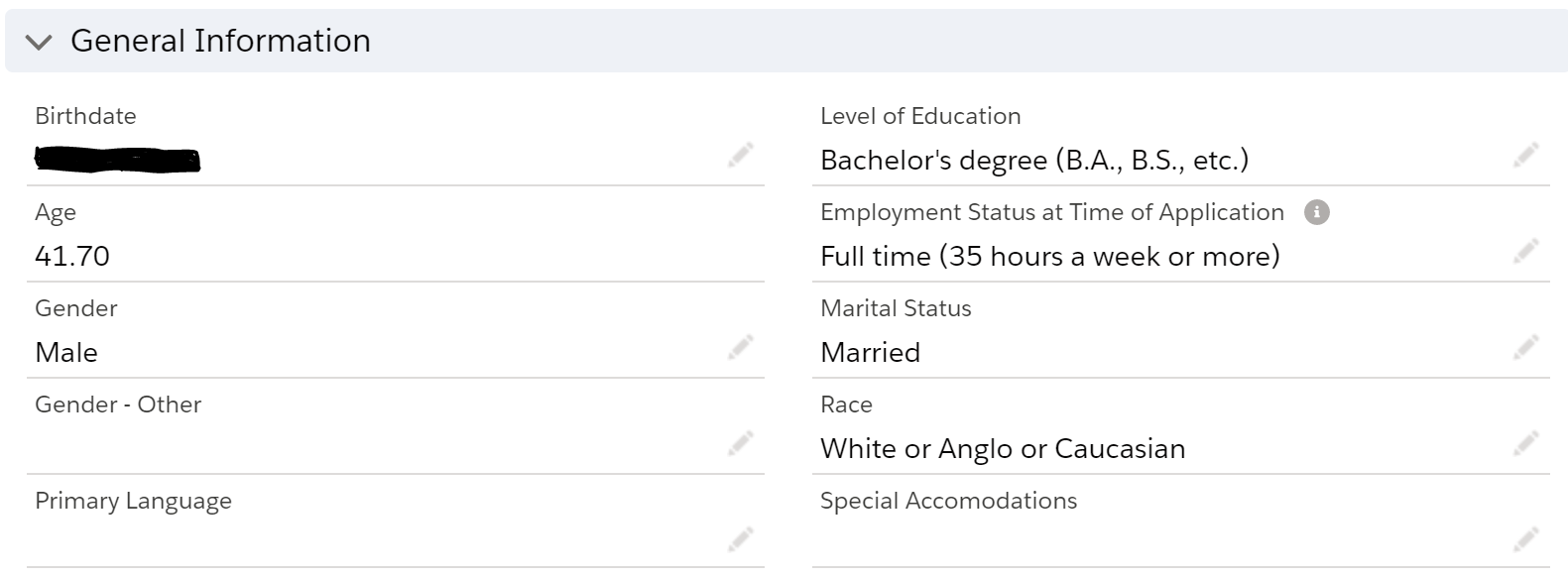
exists (select Program\_Application\_ID from DBO2OTST.Program\_Applications d where d.Program\_Application\_ID = a.program\_application))

;quit;

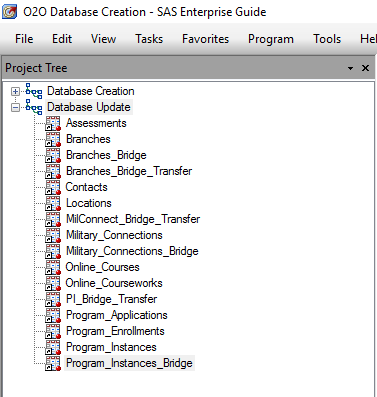
\*/

## User Interface

The User Interface’s for this database current exist in multiple forms. The first user interface for data input is Salesforce. This user interface is used for all data input which is then saved to a back up database from which the new database is built to intake data from.



The second form of the user interface for this data is SAS Analytic tools which include SAS Enterprise Guide, and SAS Visual Analytics. The SAS tools are set up to connect to SQL server and recognize the database tables via SAS Libraries. Currently, our administrator has set up the connect for this database to SAS.



## Database Updates

All database updates will be completed the same way that the data was inserted into the new database from the current database, via SAS as an intermediary. Updates will be automatically schedule in SAS Enterprise Guide and all tables will include a query similar to the query below:

**proc** **sql**;

insert into DBO2OTST.Locations

select \*

from (

select

ID as Location\_ID

,Namex as Location\_Name

,statex as Location\_State

,lastmodifieddate as Location\_Last\_Modified\_Date

from DBSFPRD.sf\_location l

where not exists (select Location\_ID from DBO2OTST.Locations d where l.ID = d.Location\_ID))

; **quit**;

The query above is an update for the Locations table. This query inserts data into the Locations database table from a sub-query that only includes data that does not currently exist in the database. It will be done via a sub-query to avoid having to create an additional table as a transfer table as was done when the data was first inserted into the tables.

## Sample Reporting

Example View

go

Create view v\_O2O\_Outcome\_Table as

select

c.Contact\_ID

,Contact\_Created\_Date

,Contact\_Employment\_status

,Contact\_Employer

,Contact\_Position

,Contact\_Employer\_Instituion

,Contact\_Employer\_Partner

,Contact\_Emp\_Start\_Date

,Contact\_Employment\_Location

,Contact\_IVMF\_Start\_Date

,Contact\_Employment\_Zip

,Contact\_Employment\_Notes

,Contact\_Outcome\_Type

,Contact\_Employment\_Position

,Contact\_IVMF\_Salary\_Range

,Contact\_Type\_of\_Degree

,Contact\_Confirmed\_Hire\_Date

,Contact\_Hired\_Location

,Contact\_Salary\_Range

,Contact\_Hired\_Zip

,Contact\_Emp\_Outcome\_Type

,Contact\_HHUSA\_Opt\_Out

,Contact\_Sent\_to\_HHUSA

,pe.Program\_Enrollment\_ID

,pe.PE\_Created\_Date

,pe.PE\_Advisor

,pe.PE\_Status

,pe.Program\_Instance\_ID

,pa.Program\_Application\_ID

,pa.PA\_Status

,pa.PA\_Program\_Application\_Origin

,pa.PA\_Program\_Completion\_Goal

,pa.PA\_Live\_Within\_50m

,pa.PA\_Orientation\_Status

,i.PI\_O2O\_Model

,i.pi\_program

,i.PI\_Program\_Instance\_desc

,i.PI\_Start\_date

from Contacts as c

right join Program\_Enrollments pe on c.Contact\_ID = pe.Contact\_ID

left join Program\_Applications pa on pa.Program\_Application\_ID = pe.Program\_Application\_ID

left join Program\_Instances i on pe.Program\_Instance\_ID = i.Program\_Instance\_ID

where SUBSTRING(c.Contact\_Source, 1, 3) = 'O2O' or SUBSTRING(c.Contact\_Source, 1, 4) = 'VCTP'

go

Example Function:

go

create function dbo.f\_Age\_at\_Enroll (@Contact\_ID char(18))

returns int as

begin

declare @returnValue int

select @returnValue = Datediff(year, C.Contact\_Birthdate, pe.PE\_Created\_Date)

from Contacts c

right join Program\_Enrollments pe on c.Contact\_ID = pe.Contact\_ID

where @Contact\_ID = c.Contact\_ID

return @returnValue

end

go

select dbo.f\_Age\_at\_Enroll (c.Contact\_ID) as AgeatEnroll

from Contacts c

Example Gender Query:

**proc** **sql**;

select

sum(case when Contact\_\_Gender = 'Male' then **1** else **0** end)/count(Contact\_\_Contact\_ID) as Male format=percent8.2

,sum(case when Contact\_\_Gender = 'Female' then **1** else **0** end)/count(Contact\_\_Contact\_ID) as Female format=percent8.2

,sum(case when Contact\_\_Gender = 'Other' then **1** else **0** end)/count(Contact\_\_Contact\_ID) as Other format=percent8.2

,sum(case when Contact\_\_Gender = '' then **1** else **0** end)/count(Contact\_\_Contact\_ID) as Missing format=percent8.2

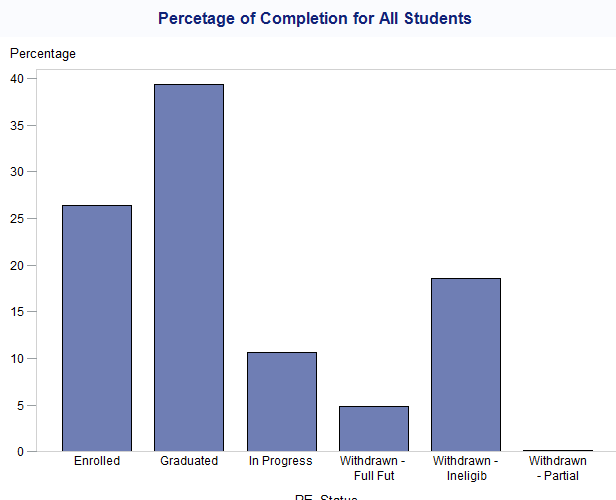
from o2odsprd.cohorts

;**quit**;

Results:



Example Program Enrollment Status Report



## Reflection

The largest assumptions I made during this project was that I would be able to get much more than done I was able to get done. The overall scope of the project was too large for this class. Part of that was on purpose in that I wanted to get done a large portion of the database needed for my organization, but part of that was accidental in that I did not realize I would need additional tables until I started coding the database. Additionally, there are still data integrity issues in the operational database (Salesforce) that causes issues with the foreign keys constraints that I did not anticipate (which is why the insert statements for the tables align with the data that already exists in the other tables). For example, there were contact ID in the Program Enrollments table that did not exist in the Contacts table, which should not be possible if the operational database was set up correctly. In this case, it was not set up correctly and is in the process of being fixed. This has many downstream effects on my database.

This project was a draft of the database we are going to create for my organization. Moving forward, I will need to go back to the consumers of the data and double check we have the correct fields and some of the fields need to be edited. For example, the Program Enrollments Status field was too short. In addition to these, other fields need to be added that I did not know that I needed in the beginning to multiple tables. The new column need for multiple tables is an ID field, for which I would need to create an additional table to connect to. Additionally, I used SAS as an intermediary for this project, which for the data that does not need to be cleaned was unnecessary. When created the final product, I am going to pull directly from the other database in SQL Management Studio, reducing the number of steps needed for the final product and updates.

Overall, the experience of creating my own database was incredibly eye opening to how difficult it is to pull off. The overall coding is not that difficult but ensuring everything will work seamlessly together is quite difficult to get right. How data will interact with the constraints is something I learned very deeply after failing to insert data many, many times and not being able to figure out why. The coding of the database was much easier with a good planning phase, and I now know more than ever how important a detailed planning phase is in this type of project.

## Final Summary

For my project, certain details such as the user interface and source for the data were already determined by the business need. The final user interface for data input is Salesforce, and the user interface for data analysis is SAS Enterprise Guide and SAS Visual Analytics. All these UI’s are currently connected and did not need additional support on my end to utilize them. The overall project stayed relatively the same, however I did make edits to part I of this project by editing the field lengths, adding fields that I did not originally include and delete a table that was not needed (Program Instance Bridge table). To answer the data questions, I created a reporting view which is basically a copy of the current Salesforce report we currently utilize to report. This view contains many more joins but allows much more control as to how the data is joined. While Salesforce is good for many things, it is difficult to determine how Salesforce joins many tables behind the scenes. Working from a database allows for much more control of the data. The data can now be used in our static reports from code, or dynamic reports in Visual Analytics to allow consumers of the data to utilize it. This project is now ready to be revised, edited, and then will be ready to be implemented as the source of our data needs. This eliminates the manual process of downloading excel reports from Salesforce and uploading them into SAS. Additional work will be needed to optimize the process; however, this is a large step in the right direction.