Rebecca Karunakaran

IST 659-42024

Karate Studio Management System

Table of Contents

[Summary 3](#_Toc516505140)

[Stakeholder 3](#_Toc516505141)

[Business Rules 3](#_Toc516505142)

[Expectations and Outcomes 4](#_Toc516505143)

[Conceptual ERD Model 5](#_Toc516505144)

[Normalized Logical Model 6](#_Toc516505145)

[Physical Database design 7](#_Toc516505146)

[Data creation 10](#_Toc516505147)

[INSERT, UPDATE, DELETE AND alTER SCRIPT 10](#_Toc516505148)

[Views 15](#_Toc516505149)

[Studio Info View 15](#_Toc516505150)

[Class Schedule Studio Location View 17](#_Toc516505151)

[Event Schedule Studio Location View 18](#_Toc516505152)

[Student Information View 19](#_Toc516505153)

[Employee Information View 21](#_Toc516505154)

[STORED PROCEDURES AND FUNCTIONS 22](#_Toc516505155)

[sp\_AddModify\_Employee\_Info 22](#_Toc516505156)

[sp\_AddModify\_Student\_Info 26](#_Toc516505157)

[IMPLEMENTATION 35](#_Toc516505158)

[STUDIO LOCATION INFORMATION ACCESS REPORT 35](#_Toc516505159)

[CLASS SCHEDULE FOR STUDIO ACCESS REPORT 36](#_Toc516505160)

[EVENT SCHEDULE BY STUDIO ACCESS REPORT 37](#_Toc516505161)

[STUDIO INFORMATION BY STUDIO ACCESS REPORT 37](#_Toc516505162)

[Employee schedule ACCESS REPORT 38](#_Toc516505163)

[Employee MAINTENANCE ACCESS form 39](#_Toc516505164)

[Student MAINTENANCE ACCESS form 39](#_Toc516505165)

[REFLECTION 40](#_Toc516505166)

# Summary

I have decided to build a database for a Karate Studio where my daughter takes Karate lesson.

The Karate Studio has multiple locations. Each location has a number of Karate classes like ’freshman’, ’sophomore’,’ junior’,’ senior’,’ graduate’ and ’adult class’ and each class can have a number of sessions scheduled. Employees at the studio can be of the following type – full-time, part-time and apprentice. Apart from classes, the studio also hosts a number of events like belt award ceremony, movie night, ice cream social etc. that need to be scheduled and tracked.

# Stakeholder

The owners of the studio are the stakeholders for this project and would like a database to access and maintain student and employee information and track class schedules for all their locations.

# Business Rules

Each studio location has one or more of classes and a class can be held in one of more studio location. A class can have multiple sessions and each session is for one class only. Each session can have one or more students and a student can be register for only one session at any given time. A student may have a Parent (An Adult Student will not have Parent Information) and a Parent must have one or many Students.

Studio employees can be full-time, part-time and apprentices. An employee can work in one or more studio location and a studio location can have one of more employees. Salaries for the employee also various depending on the employee type. A studio can have multiple events scheduled and an event can be scheduled for one or more studio locations.

# Expectations and Outcomes

The stakeholders want to be able to use this database for the following –

1. Track general information about their studio locations like the address, studio manager etc.

There are 2 ways to view the studio location information

* Using a Database View StudioInfoView (page 15)
* MS Access Report ‘Studio Location Information’ (page 35)

1. Track and maintain information about all their employees like address, salary and employee schedule.

The stakeholder can use one of the following ways to maintain Employee information

* A Stored Procedure sp\_AddModify\_Employee\_Info (page 22)
* MS Access Form ‘Employee Maintenance Form’ (page 39)

1. Class Schedule for their studio locations.

Class Schedule can be tracked using

* Database View ClassScheduleStudioLocationView (page 17)
* MS Access Report ‘Class Schedule For Studio Location Report’ (page 36)

1. Track and maintain information about the students like name, address and class they are currently registered for.

The stakeholder can use one of the following ways to maintain Student information

* A Stored Procedure sp\_AddModify\_Student\_Info (page 26)
* MS Access Form ‘Student Maintenance Form’ (page 39)

1. Event Schedule for their studio locations.

Event Schedule can be viewed using

* Database View EventScheduleStudioLocationView (page 18)
* MS Access Report ‘Event Schedule by Studio Report’ (page 37)

# Conceptual ERD Model

****

**Attribute Property Used -**

**r** is a Required attribute.

**c** is a composite attribute

**u** is a unique attribute.

**d** is a derived attribute.

**m** is a multivalued attribute.

# Normalized Logical Model

****

**Rules –**

* I have added an associative entity to any many-to-many relationships
* I have chosen to use surrogate keys for all the entities including any associative entities.
* Any composite attribute has been decomposed to their constituent parts
* Multi-value attributes have been moved to a separate table like ‘EventDate’ which I decided to move to the associative entity ‘Event\_Location\_Schedule’.
* All data types are varchar unless they are used for mathematical computations. Hence, only the attribute ‘Salary’ has been defined as numeric
* The ‘Event\_Date’ attribute is defined as ‘DateTime’ since I want to capture not only the date of the event but also the start-time
* The ‘DOB’ attribute is defined as ‘Date’

# Physical Database design

|  |  |
| --- | --- |
| **Table Name** | **Brief Description** |
| Student | Main table to maintain information for a given student |
| Class | Table that contains information on a given class like name, size etc. |
| Class\_Schedule | Contains Information on the schedule for a class like class start/end time, instructor assigned to the class etc. |
| Class\_Location | An associative table between tables Class and Studio\_Location |
| Studio\_Location | Contains information about all the studio locations like name, address etc. |
| Parent | For students that are minors, information about the parent is stored in this table |
| Adult\_Student\_Info | For adult students, information about the student like address is stored in this table |
| Employee | Main table to maintain employee information |
| Employee\_Type | Employee type information like full-time/part-time/apprentice is store in this table |
| Employee\_Location | An associative table between tables Employee and Studio\_Location |
| Event | Main table to store events |
| Event\_Location\_Schedule | An associative table between tables Event and Studio\_Location |
|  |  |

/\*-----Create Table Scripts---------\*/

Drop table if exists Student

Drop table if exists Employee\_Location

Drop table if exists Class\_Schedule

Drop table if exists Class\_Location

Drop table if exists ClassidenF

Drop table if exists Event\_Location\_Schedule

Drop table if exists Studio\_Location

Drop table if exists Parent

Drop table if exists Employee

Drop table if exists Employee\_Type

Drop table if exists [Event]

Drop table if exists [Adult\_Student\_Info]

--Table : Employee\_Type

Create table Employee\_Type(

Employee\_Type\_ID int identity not null,

Employee\_Type varchar(20) not null,

Constraint Employee\_Type\_PK Primary Key(Employee\_Type\_ID),

Constraint Employee\_Type\_U1 Unique (Employee\_Type)

)

GO

--Table : Employee

Create table Employee(

Employee\_ID int identity not null,

Employee\_First\_Name varchar(20) not null,

Employee\_Last\_Name varchar(20) not null,

SSN varchar(9) not null,

DOB Date not null,

Street\_Address varchar(15) not null,

City varchar(15) not null,

State varchar(15) not null default 'California',

ZipCode varchar(5) not null,

Salary numeric(9,2) not null,

Employee\_Type\_ID int not null,

Constraint Employee\_PK Primary Key(Employee\_ID),

Constraint Employee\_FK1 Foreign Key(Employee\_Type\_ID) references Employee\_Type(Employee\_Type\_ID)

)

GO

--Table Sudio\_Location

Create table Studio\_Location(

Studio\_Location\_ID int identity not null,

Location\_Name varchar(30) not null,

Location\_Addresss varchar(15) not null,

City varchar(15) not null,

State varchar(15) not null,

ZipCode varchar(5) not null,

Start\_Year char(4) not null,

Manager\_ID int not null,

Constraint Studio\_Location\_PK Primary Key (Studio\_Location\_ID),

Constraint Studio\_Location\_FK Foreign Key (Manager\_ID) references Employee(Employee\_ID)

)

GO

--Table : Employee\_Location

Create table Employee\_Location(

Employee\_Location\_ID int identity not null,

Employee\_ID int not null,

Studio\_Location\_ID int not null,

Constraint Employee\_Location\_PK Primary Key (Employee\_Location\_ID),

Constraint Employee\_Location\_FK1 Foreign Key (Employee\_ID) references Employee(Employee\_ID),

Constraint Employee\_Location\_FK2 Foreign Key (Studio\_Location\_ID) references Studio\_Location(Studio\_Location\_ID)

)

GO

--Table : Class

Create table Class(

Class\_ID int identity not null,

Class\_Name varchar(20) not null,

Class\_Description varchar(30) not null,

Max\_Class\_Size int not null,

Constraint Class\_PK Primary Key (Class\_ID),

Constraint Class\_U1 Unique(Class\_Name)

)

GO

--Table Class\_Location

Create table Class\_Location(

Class\_Location\_ID int identity not null,

Class\_ID int not null,

Studio\_Location\_ID int not null,

Constraint Class\_Location\_PK Primary Key (Class\_Location\_ID),

Constraint Class\_Location\_FK1 Foreign Key (Class\_ID) references Class(Class\_ID),

Constraint Class\_Location\_FK2 Foreign Key (Studio\_Location\_ID) references Studio\_Location(Studio\_Location\_ID)

)

GO

--Table Parent

Create table Parent(

Parent\_ID int identity not null,

Parent\_First\_Name varchar(30) not null,

Parent\_Last\_Name varchar(30) not null,

Street\_Address varchar(15) not null,

City varchar(15) not null,

State varchar(15) not null,

ZipCode varchar(5) not null,

Email varchar(50) not null,

Phone varchar(15) not null,

Constraint Parent\_PK Primary Key (Parent\_ID)

)

GO

--Table Class\_Schedule

Create table Class\_Schedule(

Class\_Schedule\_ID int identity not null,

Class\_Location\_ID int not null,

Instructor\_ID int not null,

Class\_Start\_Time varchar(6) not null,

Class\_End\_Time varchar(6) not null,

Day\_Of\_The\_Week varchar(10) not null,

Constraint Class\_Schedule\_PK Primary Key (Class\_Schedule\_ID),

Constraint Class\_Schedule\_FK1 Foreign Key (Class\_Location\_ID) references Class\_Location(Class\_Location\_ID),

Constraint Class\_Schedule\_FK2 Foreign Key (Instructor\_ID) references Employee(Employee\_ID)

)

GO

--Table Student

Create table Student(

Student\_ID int identity not null,

Student\_First\_Name varchar(30) not null,

Student\_Last\_Name varchar(30) not null,

DOB Date not null,

Parent\_ID int, -- I made this field optional since adult students may not have a parent

Class\_Schedule\_ID int not null,

Constraint Student\_PK Primary Key (Student\_ID),

Constraint Student\_FK1 Foreign Key (Parent\_ID) references Parent(Parent\_ID),

Constraint Student\_FK2 Foreign Key (Class\_Schedule\_ID) references Class\_Schedule(Class\_Schedule\_ID)

)

GO

--Table Event

Create table [Event](

Event\_ID int identity not null,

Event\_Name varchar(20) not null,

Event\_Description varchar(30) ,

Constraint Event\_PK primary key (Event\_ID),

Constraint Event\_U1 unique (Event\_Name)

)

GO

--Create table Event\_Location\_Schedule

Create table Event\_Location\_Schedule(

Event\_Location\_Schedule\_ID int identity not null,

Studio\_Location\_ID int not null,

Event\_ID int not null,

Event\_Date DateTime not null,

Constraint Event\_Location\_Schedule\_PK Primary Key (Event\_Location\_Schedule\_ID),

Constraint Event\_Location\_Schedule\_FK1 Foreign Key(Studio\_Location\_ID) references Studio\_Location(Studio\_Location\_ID),

Constraint Event\_Location\_Schedule\_FK2 Foreign Key(Event\_ID) references [Event](Event\_ID)

)

GO

--Create table Adult\_Student\_Info

Create table Adult\_Student\_Info(

Adult\_Student\_Info\_ID int identity not null,

Street\_Address varchar(15) not null,

City varchar(15) not null,

State varchar(15) not null,

ZipCode varchar(5) not null,

Email varchar(50) not null,

Phone varchar(15) not null,

Constraint Adult\_Student\_Info\_PK Primary Key (Adult\_Student\_Info\_ID)

)

GO

/\*-----End Create Table Scripts---------\*/

# Data creation

## INSERT, UPDATE, DELETE AND alTER SCRIPT

/\*---- Needed to make some corrections to the tables---------------\*/

--Typo in Column Name Location\_Address

SP\_RENAME 'Studio\_Location.Location\_Addresss', 'Location\_Address', 'COLUMN'

GO

-- Increasing the of some column since they could hold longer string values

Alter table Employee

Alter Column Street\_Address varchar(30) not null

GO

Alter table Studio\_Location

Alter Column Location\_Address varchar(30) not null

GO

Alter table Parent

Alter Column Street\_Address varchar(30) not null

GO

--Need to change some columns

Alter table Class

Add Age\_Group varchar(15) not null

GO

Alter table Class

Alter Column Class\_Description varchar(50) not null

GO

Alter table [Event]

Alter Column Event\_Name varchar(30) not null

GO

Alter table [Event]

Alter Column Event\_Description varchar(50)

GO

--Added FK to table Student for table Adult\_Student\_Info

Alter table Student

Add Adult\_Student\_Info\_ID int -- I made this an optional since only Adult Sudents will have this field populate

GO

Alter table Student

Add Constraint Student\_FK3 Foreign Key (Adult\_Student\_Info\_ID) references Adult\_Student\_Info(Adult\_Student\_Info\_ID)

GO

/\*----End Needed to make some corrections to the tables---------------\*/

/\*----Insert Record into tables ------\*/

--Table Employee\_Type

Insert into Employee\_Type(Employee\_Type)

values('FullTime'),('PartTime'),('Apprentice')

GO

--Review to see if data is correct

Select \* from Employee\_Type

GO

--Table Employee

--I'm not including the State Column in the insert because I want it to default to 'California'

Insert into Employee(Employee\_First\_Name, Employee\_Last\_Name, SSN, DOB, Street\_Address, City, ZipCode, Salary, Employee\_Type\_ID)

values('Daniel','Martinez','600345999','03/04/2000','34 Cross Street','San Jose','97654',40000,1)

--The rest of the inserts have State Column specified

Insert into Employee(Employee\_First\_Name, Employee\_Last\_Name, SSN, DOB, Street\_Address, City, State, ZipCode, Salary, Employee\_Type\_ID)

values('Gina','Daniel','603853856','09/21/2003','3435 Mulberry Ct','San Jose','California','93411',10000,3)

Insert into Employee(Employee\_First\_Name, Employee\_Last\_Name, SSN, DOB, Street\_Address, City, ZipCode, Salary, Employee\_Type\_ID)

values('Kim','Lee','938647846','01/20/1995','456 HorseShoe Lane','San Jose','94633',60000,1)

Insert into Employee(Employee\_First\_Name, Employee\_Last\_Name, SSN, DOB, Street\_Address, City, ZipCode, Salary, Employee\_Type\_ID)

values('Sam','Victor','653647888','07/22/1999','BlueCross Road','Santa Clara','97364',60000,2)

Insert into Employee(Employee\_First\_Name, Employee\_Last\_Name, SSN, DOB, Street\_Address, City, ZipCode, Salary, Employee\_Type\_ID)

values('Shirley','Lou','65367833','03/04/2001','First Street','Sunnyvale','95633',60000,2)

--Review to see if data is correct

Select \* from Employee

GO

-- Table Studio Location

Insert into Studio\_Location(Location\_Name,Location\_Address,City,State, ZipCode,Start\_Year,Manager\_ID)

values('America''s Best Karate','12 street','San Jose','California','93532','2015',3)

Insert into Studio\_Location(Location\_Name,Location\_Address,City,State, ZipCode,Start\_Year,Manager\_ID)

values('Victory Marital Arts','3444 Mckee Avenue','Santa Clara','California','93266','2010',3)

Insert into Studio\_Location(Location\_Name,Location\_Address,City,State, ZipCode,Start\_Year,Manager\_ID)

values('Xtremen Martial Arts','78 Walnut Grove','Sunnyvale','California','95623','2017',4)

--Review to see if data is correct

Select \* from Studio\_Location

GO

--Delete from Studio\_Location since I accidently inserted duplicate rows

Delete from Studio\_Location where Studio\_Location\_ID between 1 and 7

GO

--Table Class

Insert into Class(Class\_Name,Class\_Description,Max\_Class\_Size,Age\_Group)

values('Freshman','Beginner''s Class Belts - white to yellow1 belt',30, '5 and above')

Insert into Class(Class\_Name,Class\_Description,Max\_Class\_Size,Age\_Group)

values('Sophomore','Intermediate 1 - orange to green1',30, '5 and above')

Insert into Class(Class\_Name,Class\_Description,Max\_Class\_Size,Age\_Group)

values('Junior','Intermediate 2 - purple to blue1',30, '5 and above')

Insert into Class(Class\_Name,Class\_Description,Max\_Class\_Size,Age\_Group)

values('Senior','Advanced - brown to red-black',30, '5 and above')

Insert into Class(Class\_Name,Class\_Description,Max\_Class\_Size,Age\_Group)

values('Graduates','Proficient - black belt',20, '10 and above')

Insert into Class(Class\_Name,Class\_Description,Max\_Class\_Size,Age\_Group)

values('Adult','Cardio Class',20, '14 and above')

Insert into Class(Class\_Name,Class\_Description,Max\_Class\_Size,Age\_Group)

values('Pee-Wee','Toddlers',20, '3 and above')

--Review to see if data is correct

Select \* from Class

GO

--Table Class\_Location

Insert into Class\_Location(Class\_ID,Studio\_Location\_ID)

values(1,8),(1,10),(2,8),(2,9),(2,10),(3,9),(3,10) ,(4,8),(5,9),(6,10),(6,8),(7,8) ,(5,8),(2,8),(3,10)

--Review to see if data is correct

select \* from Class\_Location

--Remove Duplicates

delete from Class\_Location

where Class\_Location\_ID=22

delete from Class\_Location

where Class\_Location\_ID=23

--Review to see if data is correct

select \* from Class\_Location

GO

--Table Class\_Schedule

Insert into Class\_Schedule(Class\_Location\_ID,Instructor\_ID,Class\_Start\_Time,Class\_End\_Time,Day\_Of\_The\_Week)

values(9,1,'5:30pm','6:30pm','Monday')

Insert into Class\_Schedule(Class\_Location\_ID,Instructor\_ID,Class\_Start\_Time,Class\_End\_Time,Day\_Of\_The\_Week)

values(9,1,'4:30pm','6:00pm','Friday')

Insert into Class\_Schedule(Class\_Location\_ID,Instructor\_ID,Class\_Start\_Time,Class\_End\_Time,Day\_Of\_The\_Week)

values(10,2,'5:00pm','6:00pm','Wednesday')

Insert into Class\_Schedule(Class\_Location\_ID,Instructor\_ID,Class\_Start\_Time,Class\_End\_Time,Day\_Of\_The\_Week)

values(11,2,'2:30pm','3:30pm','Saturday')

Insert into Class\_Schedule(Class\_Location\_ID,Instructor\_ID,Class\_Start\_Time,Class\_End\_Time,Day\_Of\_The\_Week)

values(11,3,'5:30pm','6:30pm','Monday')

Insert into Class\_Schedule(Class\_Location\_ID,Instructor\_ID,Class\_Start\_Time,Class\_End\_Time,Day\_Of\_The\_Week)

values(12,4,'4:30pm','5:30pm','Tuesday')

Insert into Class\_Schedule(Class\_Location\_ID,Instructor\_ID,Class\_Start\_Time,Class\_End\_Time,Day\_Of\_The\_Week)

values(13,5,'5:30pm','6:30pm','Friday')

Insert into Class\_Schedule(Class\_Location\_ID,Instructor\_ID,Class\_Start\_Time,Class\_End\_Time,Day\_Of\_The\_Week)

values(12,5,'1:30pm','3:30pm','Saturday')

Insert into Class\_Schedule(Class\_Location\_ID,Instructor\_ID,Class\_Start\_Time,Class\_End\_Time,Day\_Of\_The\_Week)

values(14,2,'5:30pm','6:30pm','Monday')

Insert into Class\_Schedule(Class\_Location\_ID,Instructor\_ID,Class\_Start\_Time,Class\_End\_Time,Day\_Of\_The\_Week)

values(15,1,'7:00pm','8:30pm','Tuesday')

Insert into Class\_Schedule(Class\_Location\_ID,Instructor\_ID,Class\_Start\_Time,Class\_End\_Time,Day\_Of\_The\_Week)

values(16,3,'7:00pm','8:30pm','Thursday')

Insert into Class\_Schedule(Class\_Location\_ID,Instructor\_ID,Class\_Start\_Time,Class\_End\_Time,Day\_Of\_The\_Week)

values(19,3,'5:00pm','6:30pm','Saturday')

--Review to see if data is correct

Select \* from Class\_Schedule

GO

Update Class\_Schedule

set Class\_Start\_Time = '5:30pm'

where Class\_Schedule\_ID = 24

Update Class\_Schedule

set Class\_Start\_Time = '5:00pm'

where Class\_Schedule\_ID = 23

Select \* from Class\_Schedule

GO

--Table Parent

Insert into Parent(Parent\_First\_Name,Parent\_Last\_Name,Street\_Address, City, State, ZipCode,Email,Phone)

values('Rebecca','Karunakaran','264 Joshua Court','San Jose','California','97656','Rebecca\_Karunakaran@gmail.com','408-865-9776')

Insert into Parent(Parent\_First\_Name,Parent\_Last\_Name,Street\_Address, City, State, ZipCode,Email,Phone)

values('James','Smith','6 Washington Way','Santa Clara','California','97656','James\_Smith@yahoo.com','408-463-5424')

Insert into Parent(Parent\_First\_Name,Parent\_Last\_Name,Street\_Address, City, State, ZipCode,Email,Phone)

values('Gina','Parker','78 North Street','San Jose','California','95432','Gina\_Parker@gmail.com','765-443-5235')

Insert into Parent(Parent\_First\_Name,Parent\_Last\_Name,Street\_Address, City, State, ZipCode,Email,Phone)

values('John','Franklin','336 Main Street','Sunnyvale','California','94322','John\_Franklin@hotmail.com','408-459-1112')

--Review to see if data is correct

Select \* from Parent

GO

--Table Student

Insert into Student(Student\_First\_Name,Student\_Last\_Name,DOB,Parent\_ID,Class\_Schedule\_ID)

values('Liam','Smith','01/20/2010',2,12)

Insert into Student(Student\_First\_Name,Student\_Last\_Name,DOB,Parent\_ID,Class\_Schedule\_ID)

values('Ava','Smith','01/20/2012',2,13)

Insert into Student(Student\_First\_Name,Student\_Last\_Name,DOB,Parent\_ID,Class\_Schedule\_ID)

values('Nia','Karunakaran','06/06/2011',1,19)

Insert into Student(Student\_First\_Name,Student\_Last\_Name,DOB,Parent\_ID,Class\_Schedule\_ID)

values('Kevin','Parker','09/11/2008',3,32)

Insert into Student(Student\_First\_Name,Student\_Last\_Name,DOB,Parent\_ID,Class\_Schedule\_ID)

values('Joseph','Frankin','01/20/2007',4,28)

--Inserted an Adult Records who will not have a Parent Record

Insert into Student(Student\_First\_Name,Student\_Last\_Name,DOB,Parent\_ID,Class\_Schedule\_ID)

values('Zion','Charles','01/20/1998',NULL,32)

Insert into Student(Student\_First\_Name,Student\_Last\_Name,DOB,Parent\_ID,Class\_Schedule\_ID)

values('Gina','Davis','01/20/1985',NULL,32)

--Review to see if data is correct

select \* from Student

GO

--Updating Student Zion Charles to an Adult class since he is over 18years

Update Student

set Class\_Schedule\_ID = 34

where Student\_ID = 11

GO

--Updating Student Gina Davis to an Adult class since she is over 18years

Update Student

set Class\_Schedule\_ID = 34

where Student\_ID = 12

GO

--Added new table Adult\_Student\_Info and need to update values in the Student table for the Adult record

Update Student

Set Adult\_Student\_Info\_ID = 1

where Student\_ID = 11

GO

Update Student

Set Adult\_Student\_Info\_ID = 2,

where Student\_ID = 12

GO

--Review to see if data is correct

select \* from Student

GO

--Table Event

Insert into [Event](Event\_Name,Event\_Description)

values('Black Belt Ceremony','Black Belt will be awraded to qualifying students'),

('Ice-Cream Social',null), ('Movie Night',null)

--Review to see if data is correct

Select \* from [Event]

GO

--Table Event\_Location\_Schedule

Insert into Event\_Location\_Schedule(Studio\_Location\_ID,Event\_ID,Event\_Date)

values(8,1,'06/09/2018 5:00PM'), (8,2,'09/09/2018 6:30PM'), (9,1,'07/05/2018 5:30PM'), (10,3,'08/23/2018 7:00 PM'), (10,1,'06/20/2018 4:00PM')

GO

--Review to see if data is correct

Select \* from Event\_Location\_Schedule

GO

--Table Employee\_Location

Insert into Employee\_Location(Employee\_ID,Studio\_Location\_ID)

values(1,8),(1,10),(2,9),(3,8),(4,8),(5,10),(3,9)

GO

--Review to see if data is correct

Select \* from Employee\_Location

GO

--Table Adult\_Student\_Info

Insert into Adult\_Student\_Info(Street\_Address,City,State,ZipCode,Email,Phone)

values('56 Blue Hawk Dr','Morgan Hill','California','95436','Zion\_Charles@yahoo.com','408-938-8473')

GO

Insert into Adult\_Student\_Info(Street\_Address,City,State,ZipCode,Email,Phone)

values('600 Shorelane','Sunnyvale','California','92323','Gine\_Davis@hotmail.com','408-364-2222')

GO

--Review to see if data is correct

Select \* from Adult\_Student\_Info

GO

/\*-----------End Insert Statements------------------\*/

## Views

### Studio Info View

**PURPOSE**

One of the request from the stakeholder was to be able to track general information about their studio locations like the address, studio manager etc. The StudioInfoView was built to display the requested information.

------View to track Studio Information -----------

drop View StudioInfoView

GO

Create View StudioInfoView As

Select Studio\_Location.Location\_Name,

Studio\_Location.Location\_Address as StreetAddress ,

Studio\_Location.City,

Studio\_Location.State ,

Studio\_Location.ZipCode,

(Employee.Employee\_First\_Name + ' ' + Employee.Employee\_Last\_Name) as Manager,

Count(Employee.Employee\_ID) as TotalEmployees -- Used Aggregate Function Count to find total employee for a studio

From Studio\_Location

Join Employee on Studio\_Location.Manager\_ID = Employee.Employee\_ID

join Employee\_Location on Employee\_Location.Studio\_Location\_ID = Studio\_Location.Studio\_Location\_ID

Group by Studio\_Location.Location\_Name,

Studio\_Location.Location\_Address ,

Studio\_Location.City,

Studio\_Location.State ,

Studio\_Location.ZipCode,

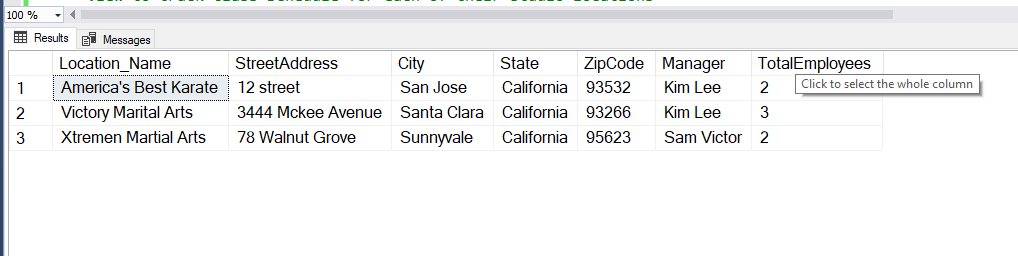
(Employee.Employee\_First\_Name + ' ' + Employee.Employee\_Last\_Name)

GO

Select \* from StudioInfoView

GO

**RESULT**



### Class Schedule Studio Location View

**PURPOSE**

The Stakeholders also wanted a way to view the Class Schedule for all their Studios. This view was built to enable to get that information.

-----View to track Class Schedule for each of their studio locations --------

Drop View ClassScheduleStudioLocationView

GO

Create View ClassScheduleStudioLocationView as

Select Studio\_Location.Location\_Name as Studio,

Class.Class\_Name as Class,

(Class\_Schedule.Class\_Start\_Time + ' to ' + Class\_Schedule.Class\_End\_Time) as ClassTiming,

Class\_Schedule.Day\_Of\_the\_Week as Day,

(Employee.Employee\_First\_Name + ' ' + Employee.Employee\_Last\_Name) as Instructor

from Class\_Schedule

join Class\_Location on Class\_Schedule.Class\_Location\_ID = Class\_Location.Class\_Location\_ID

join Class on Class\_Location.Class\_ID = Class.Class\_ID

join Studio\_Location on Class\_Location.Studio\_Location\_ID = Studio\_Location.Studio\_Location\_ID

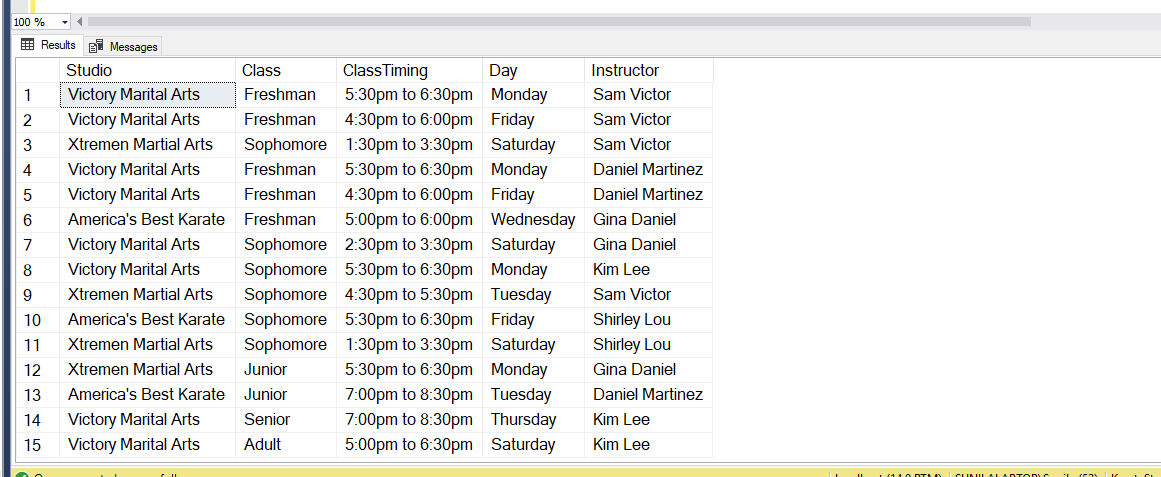
join Employee on Class\_Schedule.Instructor\_ID = Employee.Employee\_ID

GO

Select \* from ClassScheduleStudioLocationView

GO

**RESULT**



### Event Schedule Studio Location View

**PURPOSE**

The Stakeholders also wanted a way to view the Event Schedule for all their Studios. This view was built to enable to get that information.

-----View to track Event Schedule for each of their studio locations --------

Drop View EventScheduleStudioLocationView

GO

Create View EventScheduleStudioLocationView as

Select Studio\_Location.Location\_Name as Studio,

Event.Event\_Name as Event,

(FORMAT(Event\_Location\_Schedule.Event\_Date,'dd/MM/yy') + ' ' + FORMAT(Event\_Location\_Schedule.Event\_Date,'hh:mm tt')) as Date -- want to convert into readable format with AM/PM

from Event\_Location\_Schedule

join Event on Event\_Location\_Schedule.Event\_ID = Event.Event\_ID

join Studio\_Location on Event\_Location\_Schedule.Studio\_Location\_ID = Studio\_Location.Studio\_Location\_ID

GO

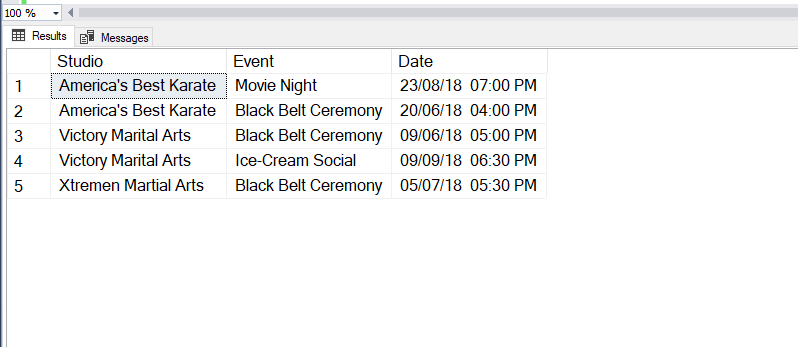
**RESULT**

--View Event for all Studios

Select \* from EventScheduleStudioLocationView

order by Studio

GO

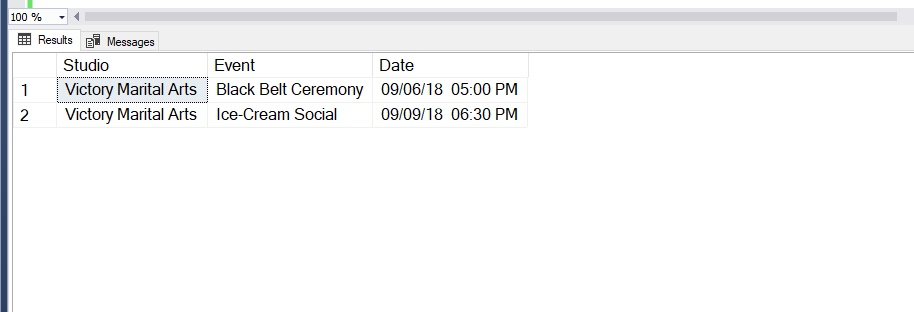


--View Event per Studio

Select \* from EventScheduleStudioLocationView

where Studio = 'Victory Marital Arts'

GO



### Student Information View

**PURPOSE**

The Stakeholders want to track information about the students like name, address and class they are currently registered for. The StudentInformationView does just that.

-----View to track Student Information --------

Drop View StudentInformationView

GO

Create View StudentInformationView as

Select(Student.Student\_First\_Name + ' ' + Student.Student\_Last\_Name) as StudentName,

isnull((Parent.Parent\_First\_Name + ' ' + Parent.Parent\_Last\_Name),'N/A') as ParentName, --Adult Students will not have a Parent Record

convert(varchar, Student.DOB, 101) as DOB,

(DATEDIFF (yy, Student.DOB, GETDATE())) as Age,

(Case When Student.Parent\_ID is not null then Parent.Phone else Adult\_Student\_Info.Phone end) as ContactNumber,

Studio\_Location.Location\_Name as Studio,

Class.Class\_Name as Class ,

Class\_Schedule.Day\_Of\_The\_Week as Day,

(Class\_Schedule.Class\_Start\_Time + ' to ' + Class\_Schedule.Class\_End\_Time) as ClassTiming

from Student

left join Parent on Student.Parent\_ID = Parent.Parent\_ID -- left join to get Student records that have not Parent Records because they are Adults

join Class\_Schedule on Student.Class\_Schedule\_ID = Class\_Schedule.Class\_Schedule\_ID

join Class\_Location on Class\_Schedule.Class\_Location\_ID = Class\_Location.Class\_Location\_ID

join Studio\_Location on Class\_Location.Studio\_Location\_ID = Studio\_Location.Studio\_Location\_ID

left join Adult\_Student\_Info on Student.Adult\_Student\_Info\_ID =Adult\_Student\_Info.Adult\_Student\_Info\_ID

join Class on Class\_Location.Class\_ID = Class.Class\_ID

GO

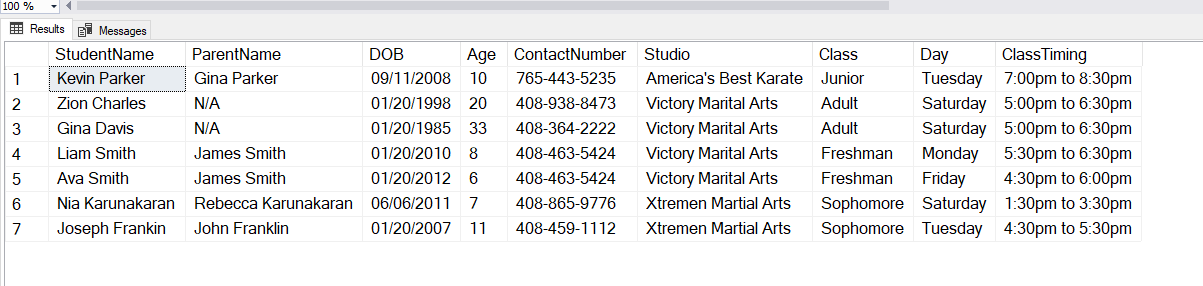
**RESULT**

--View Students for all Studios

Select \* from StudentInformationView

order by Studio,Class

GO

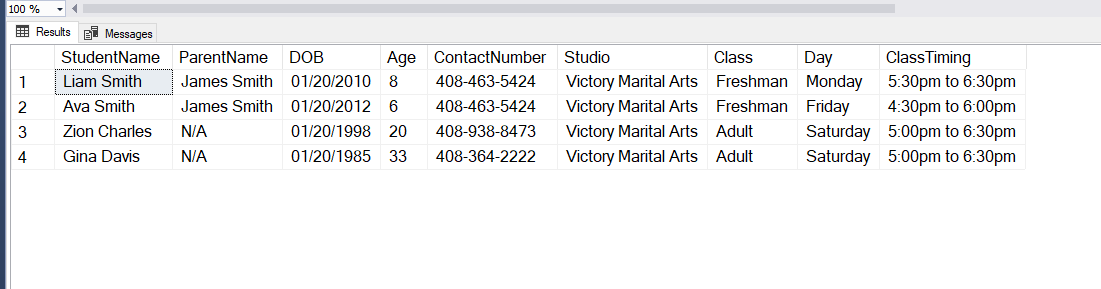


--View Students per Studio

Select \* from StudentInformationView

where Studio = 'Victory Marital Arts'

GO



### Employee Information View

**PURPOSE**

This view allows the stakeholder view all their employee’s schedule and as well allows individual employee to view their individual schedule.

-----View to track Employee Information --------

Drop View EmployeeInformationView

GO

Create View EmployeeInformationView as

Select (Employee.Employee\_First\_Name + ' ' + Employee.Employee\_Last\_Name) as EmployeeName,

Studio\_Location.Location\_Name as Studio,

Class.Class\_Name,

(Class\_Schedule.Class\_Start\_Time + ' to ' + Class\_Schedule.Class\_End\_Time) as ClassTiming,

Class\_Schedule.Day\_Of\_The\_Week Day

from Employee

join Class\_Schedule on Class\_Schedule.Instructor\_ID = Employee.Employee\_ID

join Class\_Location on Class\_Schedule.Class\_Location\_ID = Class\_Location.Class\_Location\_ID

join Class on Class\_Location.Class\_ID = Class.Class\_ID

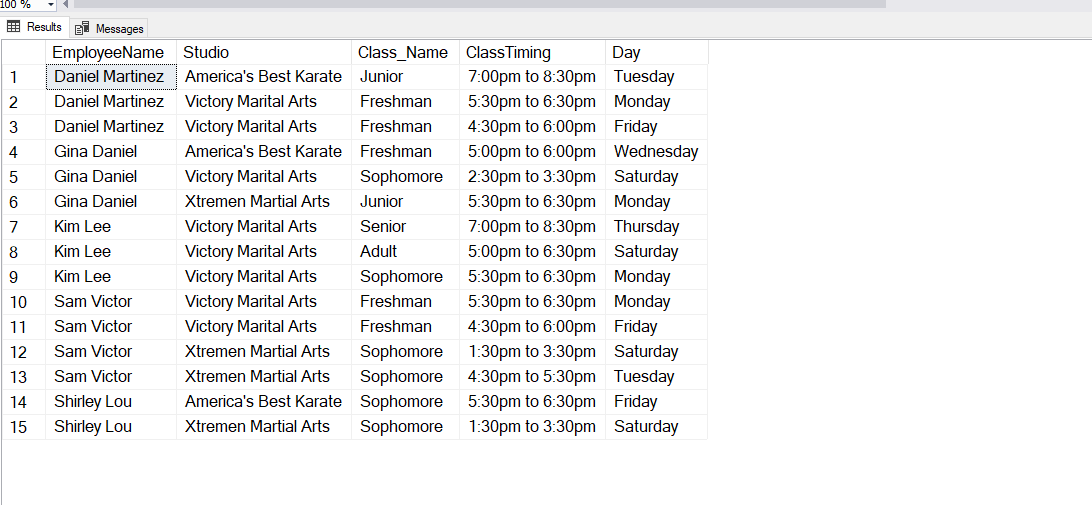
join Studio\_Location on Class\_Location.Studio\_Location\_ID = Studio\_Location.Studio\_Location\_ID

**RESULT**

select \* from EmployeeInformationView

order by EmployeeName,Studio

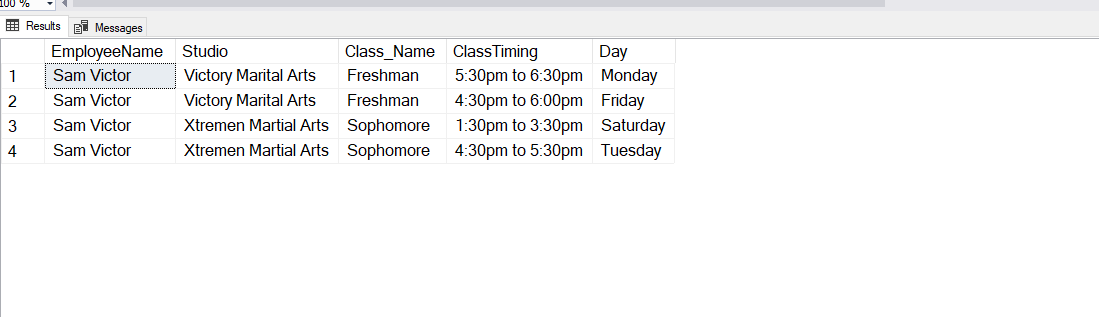
GO



select \* from EmployeeInformationView

where EmployeeName = 'Sam Victor'

GO



## STORED PROCEDURES AND FUNCTIONS

### sp\_AddModify\_Employee\_Info

**PURPOSE**

The Stakeholders wanted to be able to maintain information about all their employees like address, salary and employee schedule. This stored procedure allows them to both add or update employee information.

--Function to retrieve the Employee\_Type\_ID for a given Employee\_Type to be used in

--Stored Procedure sp\_AddModify\_EMployee\_Info

Create Function dbo.Lookup\_EmployeeType\_ID(@Employee\_Type varchar(20))

Returns int as

Begin

Declare @returnValue int

Select @returnValue = Employee\_Type\_ID

from Employee\_Type

Where Employee\_Type = @Employee\_Type

--Send the Employee\_Type\_ID back

return @returnValue

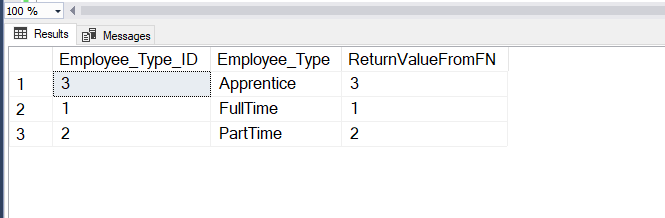
End

Go

--Check to see if function works correctly

Select \*, dbo.Lookup\_EmployeeType\_ID(Employee\_Type) as ReturnValueFromFN

from Employee\_Type



---Stored Procedure to Add/Modify Employee Information

Drop Procedure dbo.sp\_AddModify\_EMployee\_Info

Create Procedure sp\_AddModify\_Employee\_Info(

@Employee\_First\_Name varchar(20),

@Employee\_Last\_Name varchar(20),

@Employee\_SSN varchar(9),

@Employee\_DOB Date,

@Street\_Address varchar(30),

@City varchar(15),

@State varchar(15),

@ZipCode varchar(5),

@Salary numeric(9,2),

@Employee\_Type varchar(20)

)

AS

Begin

Declare @Employee\_Type\_ID int

--Get the Employee\_Type\_ID fisrt for the Employee\_Type provided as the input parameter

Select @Employee\_Type\_ID = dbo.Lookup\_EmployeeType\_ID(@Employee\_Type)

From Employee\_Type

--Check to see if the record exist. If it does, update the record otherwise add the record

If Exists (Select \* from Employee

where Employee\_First\_Name = @Employee\_First\_Name

and Employee\_Last\_Name = @Employee\_Last\_Name)

Begin

Update Employee

Set SSN = @Employee\_SSN,

DOB = @Employee\_DOB,

Street\_Address = @Street\_Address,

City = @City,

State = @State,

ZipCode = @ZipCode,

Salary = @Salary,

Employee\_Type\_id = dbo.Lookup\_EmployeeType\_ID(@Employee\_Type) – Lookup function

Where Employee\_First\_Name = @Employee\_First\_Name

And Employee\_Last\_Name = @Employee\_Last\_Name

End

Else

Begin

Insert into Employee(Employee\_First\_Name, Employee\_Last\_Name, SSN, DOB, Street\_Address, City, State, ZipCode, Salary, Employee\_Type\_ID)

values(@Employee\_First\_Name, @Employee\_Last\_Name, @Employee\_SSN, @Employee\_DOB, @Street\_Address, @City, @State, @ZipCode, @Salary, @Employee\_Type\_ID)

Return @@IDENTITY

End

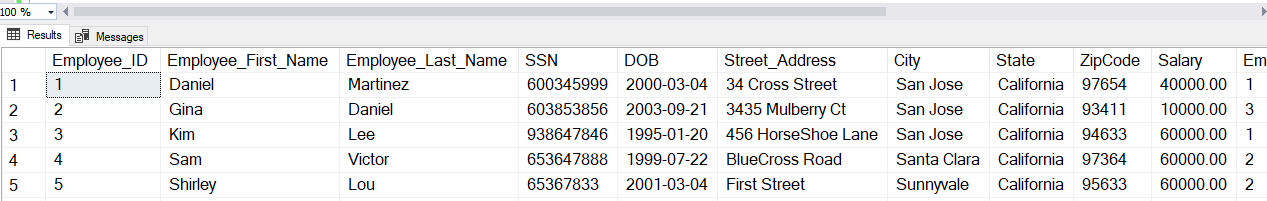
End

Go

--Check to see if the stored procedure works correctly

--Let’s view the existing Employee table

select \* from Employee



--Insert a new record via stored procedure

Declare @ReturnValue int

EXEC @ReturnValue = dbo.sp\_AddModify\_Employee\_Info

@Employee\_First\_Name = 'Liam',

@Employee\_Last\_Name = 'Mathew',

@Employee\_SSN = '6993673333',

@Employee\_DOB = '08/20/1995',

@Street\_Address = '56 Main Stree',

@City = 'San Jose',

@State = 'California',

@ZipCode = '93456',

@Salary = 40000,

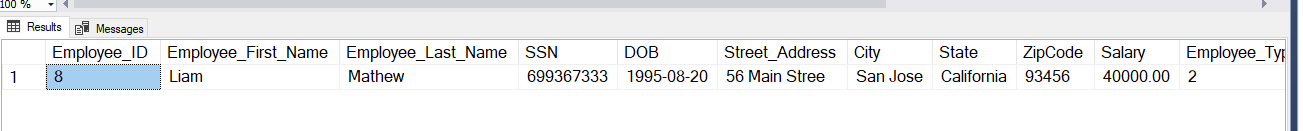
@Employee\_Type = 'PartTime'

GO

SELECT \* FROM Employee where Employee\_ID = @ReturnValue

GO

--We see our new record being inserted



--Update the same record, changing city from 'San Jose' to 'Sunnyvale' via the stored procedure

EXEC dbo.sp\_AddModify\_Employee\_Info @Employee\_First\_Name = 'Liam',

@Employee\_Last\_Name = 'Mathew',

@Employee\_SSN = '6993673333',

@Employee\_DOB = '08/20/1995',

@Street\_Address = '56 Main Stree',

@City = 'Sunnyvale',--'San Jose' to 'Sunnyvale

@State = 'California',

@ZipCode = '93456',

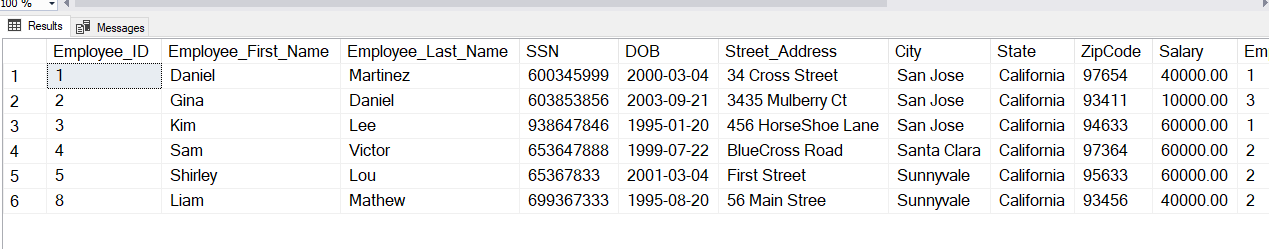
@Salary = 40000,

@Employee\_Type = 'PartTime'

GO

--Let’s confirm the record was successfully updated

select \* from Employee



### sp\_AddModify\_Student\_Info

**PURPOSE**

The Stakeholders wanted to be able to maintain information about all their student like address, dob, class schedule etc. This stored procedure allows them to both add or update student information.

--Before we implement the stored procedure, we need to implement 2 functions that will be called by our stored procedure

--Function to retrieve the Class\_Schedule\_ID based on the input parameters

Drop Function dbo.Lookup\_Class\_Schedule\_ID

Create Function dbo.Lookup\_Class\_Schedule\_ID(@Class varchar(20),@Studio varchar(30),@Day\_Of\_Week varchar(10),@Class\_Start\_Time varchar(6))

Returns int as

Begin

Declare @returnValue int , @Class\_Location\_ID int, @Class\_ID int, @Studio\_Location\_ID int

--Get the Class\_ID and Studio\_Location\_ID

Select @Class\_ID = Class\_ID from Class where Class\_Name = @Class

Select @Studio\_Location\_ID = Studio\_Location\_ID from Studio\_Location where Location\_Name = @Studio

--Get the Class\_Location\_ID first

Select @Class\_Location\_ID = Class\_Location\_ID

from Class\_Location

Where Class\_ID = @Class\_ID

And Studio\_Location\_ID = @Studio\_Location\_ID

Select @returnValue = Class\_Schedule\_ID

from Class\_Schedule

Where Class\_Location\_ID = @Class\_Location\_ID

and Day\_of\_the\_Week = @Day\_Of\_Week

and Class\_Start\_Time = @Class\_Start\_Time

--Send the Class\_Schedule\_ID back

return @Class\_Location\_ID

End

Go

--Check to see if function works correctly

Select Class.Class\_Name, Studio\_Location.Location\_Name,Class\_Schedule.Class\_Schedule\_ID,Class\_Schedule.Class\_Location\_ID,

dbo.Lookup\_Class\_Schedule\_ID(Class.Class\_Name, Studio\_Location.Location\_Name,Class\_Schedule.Day\_Of\_The\_Week,Class\_Schedule.Class\_Start\_Time) as ReturnValueFromFunction

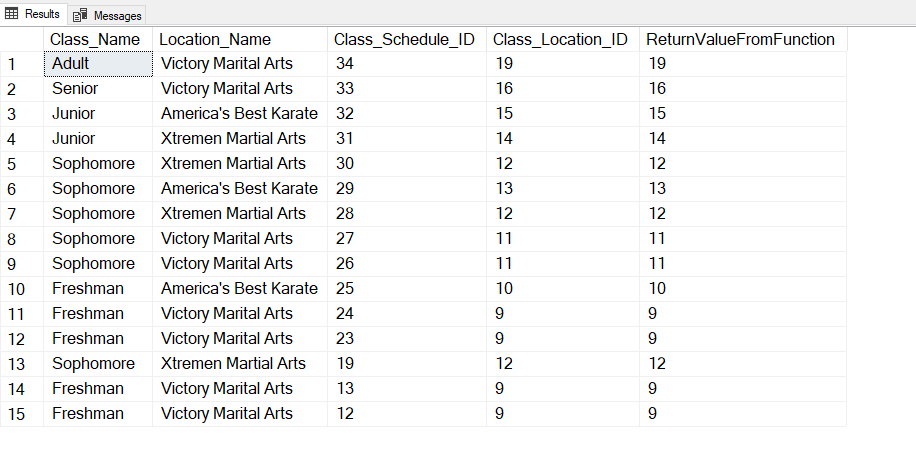
from Class\_Schedule

join Class\_Location on Class\_Schedule.Class\_Location\_ID = Class\_Location.Class\_Location\_ID

join Class on Class.Class\_ID = Class\_Location.Class\_ID

join Studio\_Location on Class\_Location.Studio\_Location\_ID = Studio\_Location.Studio\_Location\_ID

order by Class\_Schedule\_ID desc



--Function to check if the class is full

Drop Function dbo.Is\_Class\_Full

Create Function dbo.Is\_Class\_Full(@Class varchar(20),@Studio varchar(30),@Day\_Of\_Week varchar(10),@Class\_Start\_Time varchar(6))

Returns int as

Begin

Declare @returnValue int , @Class\_Size int , @Class\_Count int

--Get the Class Size

Select @Class\_Size = Max\_Class\_Size from Class where Class\_Name = @Class

-- Get the Count of Student for the given class/studio/dayofweek/starttime combination

Select @Class\_Count = count(Student\_ID)

from Student

inner join Class\_Schedule on Student.Class\_Schedule\_ID = Class\_Schedule.Class\_Schedule\_ID

join Class\_Location on Class\_Location.Class\_Location\_ID = dbo.Lookup\_Class\_Schedule\_ID(@Class, @Studio,@Day\_Of\_Week,@Class\_Start\_Time)

--Send the return value which indicates whether class is full or no

If (@Class\_Count < @Class\_Size )

Select @returnValue = 0 --Class is not full

else

Select @returnValue = 1 --Class is full

return @returnValue

End

GO

--Check to see if function works correctly

Select Class.Class\_Name, Studio\_Location.Location\_Name,Class\_Schedule.Class\_Start\_Time, Class\_Schedule.Day\_Of\_The\_Week,count(Student\_ID) as Count,

dbo.Is\_Class\_Full(Class.Class\_Name, Studio\_Location.Location\_Name,Class\_Schedule.Day\_Of\_The\_Week,Class\_Schedule.Class\_Start\_Time) as ReturnValueFromFunction

from Student

join Class\_Schedule on Student.Class\_Schedule\_ID = Class\_Schedule.Class\_Schedule\_ID

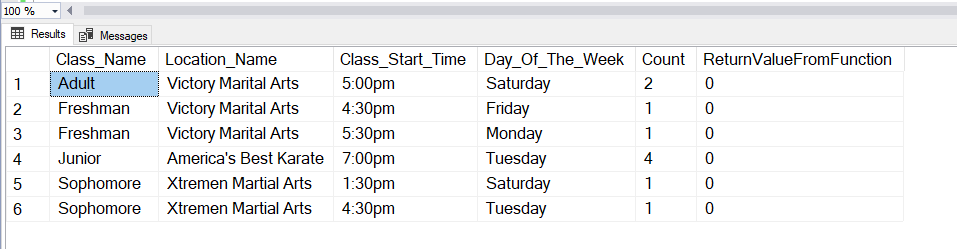
join Class\_Location on Class\_Schedule.Class\_Location\_ID = Class\_Location.Class\_Location\_ID

join Class on Class.Class\_ID = Class\_Location.Class\_ID

join Studio\_Location on Class\_Location.Studio\_Location\_ID = Studio\_Location.Studio\_Location\_ID

group by Class.Class\_Name, Studio\_Location.Location\_Name,Class\_Schedule.Class\_Start\_Time, Class\_Schedule.Day\_Of\_The\_Week

order by Class\_Name, Studio\_Location.Location\_Name



---Stored Procedure to Add/Modify Student Information

Drop Procedure dbo.sp\_AddModify\_Student\_Info

Create Procedure sp\_AddModify\_Student\_Info(

@Student\_First\_Name varchar(20),

@Student\_Last\_Name varchar(20),

@Student\_DOB Date,

@Parent\_First\_Name varchar(20),

@Parent\_Last\_Name varchar(20),

@Street\_Address varchar(30),

@City varchar(15),

@State varchar(15),

@ZipCode varchar(5),

@Email varchar(50),

@Phone varchar(15),

@Class varchar(20),

@Studio varchar(30),

@Day\_Of\_Week varchar(10),

@Class\_Start\_Time varchar(6)

)

As

Begin

Declare @IsAdult char(1), @Class\_Schedule\_ID int, @Parent\_ID int, @Adult\_Student\_Info int, @Is\_Class\_Full int

--Check to see if the Student is an Adult or not, this will determine which to add the contact information

If (datediff(yy,@Student\_DOB,getdate()) >= 18)

select @IsAdult=1

Else

select @IsAdult=0

--Get the Class\_Schedule\_ID based on the input parameters

Select @Class\_Schedule\_ID = dbo.Lookup\_Class\_Schedule\_ID(@Class,@Studio,@Day\_Of\_Week,@Class\_Start\_Time)

From Class\_Schedule

--Check to see if the record exist. If it does, update the record otherwise add the record

If Exists (Select \* from Student

where Student\_First\_Name = @Student\_First\_Name

and Student\_Last\_Name = @Student\_Last\_Name)

Begin

--Record Exists so Modify

Update Student

Set DOB = @Student\_DOB,

Class\_Schedule\_ID = @Class\_Schedule\_ID

Where Student\_First\_Name = @Student\_First\_Name

And Student\_Last\_Name = @Student\_Last\_Name

If (@IsAdult=0)

Begin

--If Student is a Minor, the address is stored in the Parent table

Update Parent

Set Street\_Address = @Street\_Address,

City = @City,

State = @State,

ZipCode = @ZipCode,

Email = @Email,

Phone = @Phone

Where Parent\_ID = (Select Parent\_ID from Student where

Student\_First\_Name = @Student\_First\_Name

And Student\_Last\_Name = @Student\_Last\_Name)

End

Else

Begin

-- Student is an Adult, the address is stored in Adult\_Student\_Info table

Update Adult\_Student\_Info

Set Street\_Address = @Street\_Address,

City = @City,

State = @State,

ZipCode = @ZipCode,

Email = @Email,

Phone = @Phone

Where Adult\_Student\_Info\_ID = (Select Adult\_Student\_Info\_ID from Student where Student\_First\_Name = @Student\_First\_Name

And Student\_Last\_Name = @Student\_Last\_Name)

End

End

Else

Begin

--Record does not exists so Add

-- But first see if the class is full or not

Select @Is\_Class\_Full = dbo.Is\_Class\_Full(@Class, @Studio,@Day\_Of\_Week,@Class\_Start\_Time)

If (@Is\_Class\_Full = 0) -- which means it not full

Begin

If (@IsAdult=0)

Begin

--Student is a minor, so add the Parent Record first

Insert into Parent(Parent\_First\_Name,Parent\_Last\_Name,Street\_Address, City, State, ZipCode,Email,Phone)

values(@Parent\_First\_Name,@Parent\_Last\_Name,@Street\_Address, @City, @State, @ZipCode,@Email,@Phone)

-- we should get the Parent\_ID from the newly added record

Select @Parent\_ID = @@IDENTITY

--Finally insert the student record

Insert into Student(Student\_First\_Name,Student\_Last\_Name,DOB,Parent\_ID, Class\_Schedule\_ID)

values (@Student\_First\_Name, @Student\_Last\_Name, @Student\_DOB, @Parent\_ID,@Class\_Schedule\_ID)

End

Else

Begin

--Student is an adult, so add a Adult\_Student\_Info Record first

Insert into Adult\_Student\_Info(Street\_Address,City,State,ZipCode,Email,Phone)

values(@Street\_Address, @City, @State, @ZipCode, @Email, @Phone)

-- we should get the Adult\_Student\_Info\_ID from the newly added record

Select @Adult\_Student\_Info = @@IDENTITY

--Finally insert the student record

Insert into Student(Student\_First\_Name,Student\_Last\_Name,DOB,Class\_Schedule\_ID, Adult\_Student\_Info\_ID)

values (@Student\_First\_Name, @Student\_Last\_Name, @Student\_DOB,@Class\_Schedule\_ID , @Adult\_Student\_Info)

End

End

End

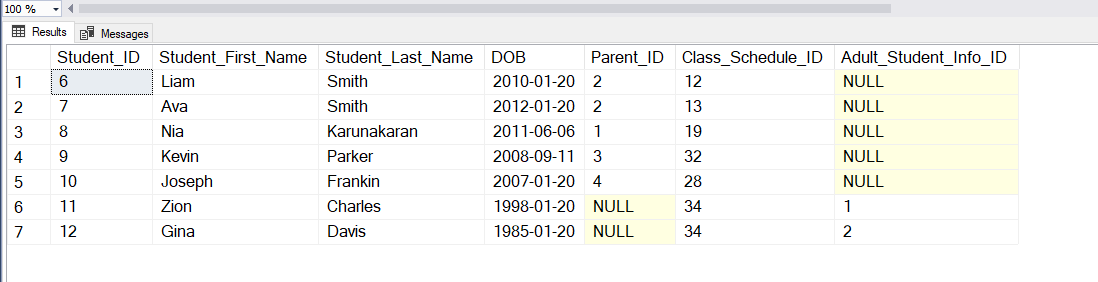
End

GO

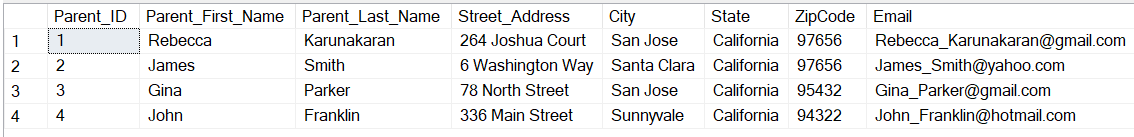
--Update existing student who is a minor to change the DOB from '01/02/2012' to '03/05/2012

-- and Street Address from '6 Washington Way' to '600 Washington Way'

select \* from Student



select \* from Parent



--Get the Class Schedule Info for 'Ava Smith'

select Class.Class\_Name,Studio\_Location.Location\_Name , Class\_Schedule.Day\_Of\_The\_Week, Class\_Schedule.Class\_Start\_Time

from Student

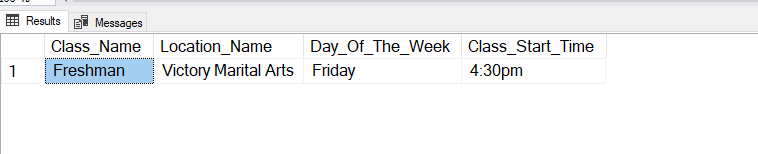
join Class\_Schedule on Student.Class\_Schedule\_ID = Class\_Schedule.Class\_Schedule\_ID

join Class\_Location on Class\_Schedule.Class\_Location\_ID = Class\_Location.Class\_Location\_ID

join Class on Class\_Location.Class\_ID = Class.Class\_ID

join Studio\_Location on Studio\_Location.Studio\_Location\_ID = Class\_Location.Studio\_Location\_ID

where Student.Student\_First\_Name = 'Ava' and Student.Student\_Last\_Name = 'Smith'



--Execute the Stored Procedure to Update a student who is a minor to change the DOB from '01/02/2012' to '03/05/2012

-- and Street Address from '6 Washington Way' to '600 Washington Way

EXEC dbo.sp\_AddModify\_Student\_Info 'Ava','Smith','03/05/2012','James','Smith','600 Washington Way',

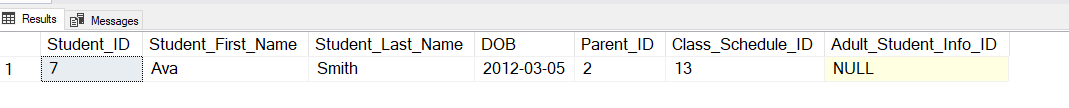
'Santa Clara','California','97656','James\_Smith@yahoo.com','408-463-5424',

'Freshman','Victory Marital Arts','Friday','4:30pm'

--Let’s validate if the record was updated successfully

Select \* from Student where Student\_First\_Name = 'Ava' And Student\_Last\_Name = 'Smith'

--Yes, DOB was changed successfully '01/02/2012' to '03/05/2012



Select \* from Parent where Parent\_ID = (select Parent\_ID from Student

where Student\_First\_Name = 'Ava' And Student\_Last\_Name = 'Smith' )

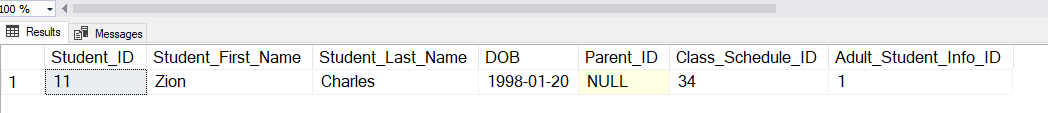
--Yes, Street Address was changed successfully '6 Washington Way' to '600 Washington Way'



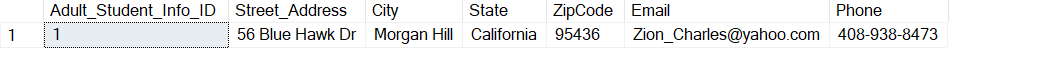
--Update existing student 'Zion Charles' who is an Adult to change the DOB from '01/20/1998' to '01/25/1998'

-- and Email from 'Zion\_Charles@yahoo.com' to 'Zion\_Charles@hotmail.com'

select \* from Student where Student\_First\_Name ='Zion' and Student\_Last\_Name = 'Charles'



select \* from Adult\_Student\_Info where Adult\_Student\_Info\_ID = 1



--Get the Class Schedule Info for 'Zion Charles'

select Class\_Schedule.class\_schedule\_ID,Class.Class\_Name,Studio\_Location.Location\_Name , Class\_Schedule.Day\_Of\_The\_Week, Class\_Schedule.Class\_Start\_Time

from Student

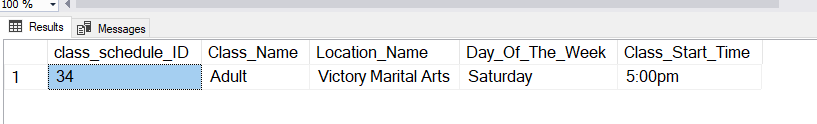
join Class\_Schedule on Student.Class\_Schedule\_ID = Class\_Schedule.Class\_Schedule\_ID

join Class\_Location on Class\_Schedule.Class\_Location\_ID = Class\_Location.Class\_Location\_ID

join Class on Class\_Location.Class\_ID = Class.Class\_ID

join Studio\_Location on Studio\_Location.Studio\_Location\_ID = Class\_Location.Studio\_Location\_ID

where Student.Student\_First\_Name = 'Zion' and Student.Student\_Last\_Name = 'Charles'



--Execute the Stored Procedure to Update the record

EXEC dbo.sp\_AddModify\_Student\_Info 'Zion','Charles','01/25/1998',NULL,NULL,'56 Blue Hawk Dr',

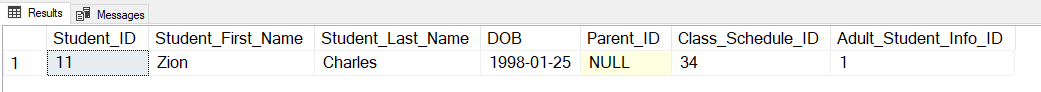
'Morgan Hill','California','95436','Zion\_Charles@hotmail.com','408-938-8473',

'Adult','Victory Marital Arts','Saturday','5:00pm'

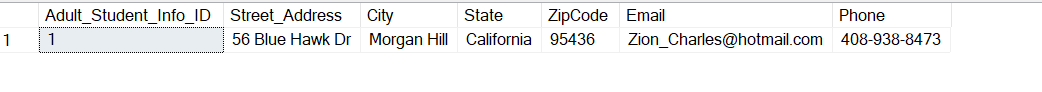
--Yes, DOB changed from '01/20/1998' to '01/25/1998'

-- and Email from 'Zion\_Charles@yahoo.com' to 'Zion\_Charles@hotmail.com'

select \* from Student where Student\_First\_Name ='Zion' and Student\_Last\_Name = 'Charles'



select \* from Adult\_Student\_Info where Adult\_Student\_Info\_ID = 1



---Let's use the Stored Proc to Add a new student who is a minor

--Execute the Stored Procedure to Update the record

EXEC dbo.sp\_AddModify\_Student\_Info 'Taylor','Swift','01/25/2003','Ronald','Smith','34 Hollywood Dr',

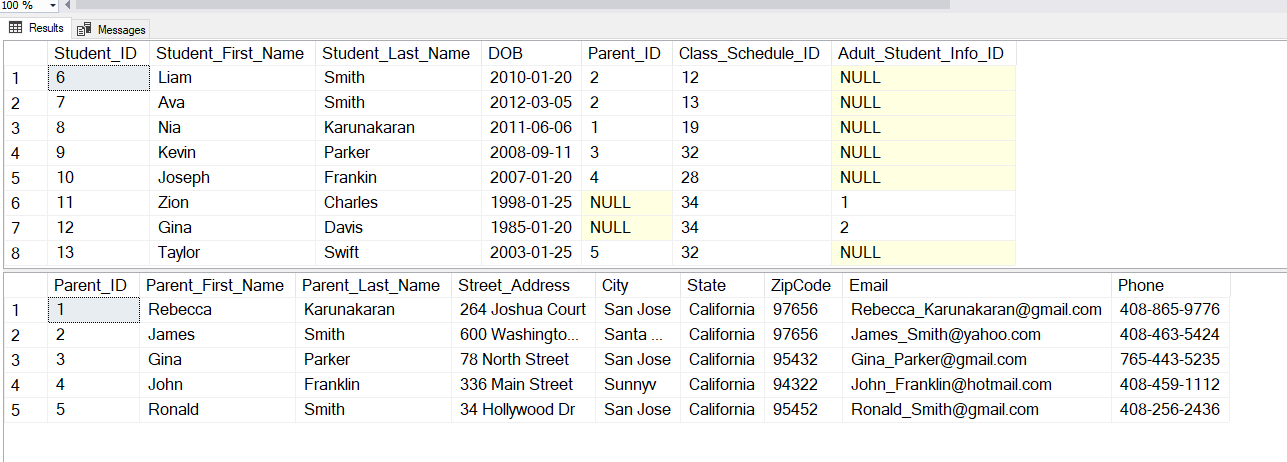
'San Jose','California','95452','Ronald\_Smith@gmail.com','408-256-2436',

'Junior','America''s Best Karate','Tuesday','7:00pm'

--Let’s check if the record was inserted correctly. Since the student is a minor, we should a parent record inserted as well

select \* from Student

select \* from Parent



---Let's use the Stored Proc to Add a new student who is an Adult

--Execute the Stored Procedure to Update the record

EXEC dbo.sp\_AddModify\_Student\_Info 'Tom','James','01/25/1990',NULL,NULL,'34 Eagle Lane',

'San Jose','California','93432','Tom\_James@yahoo.com','125-343-6564',

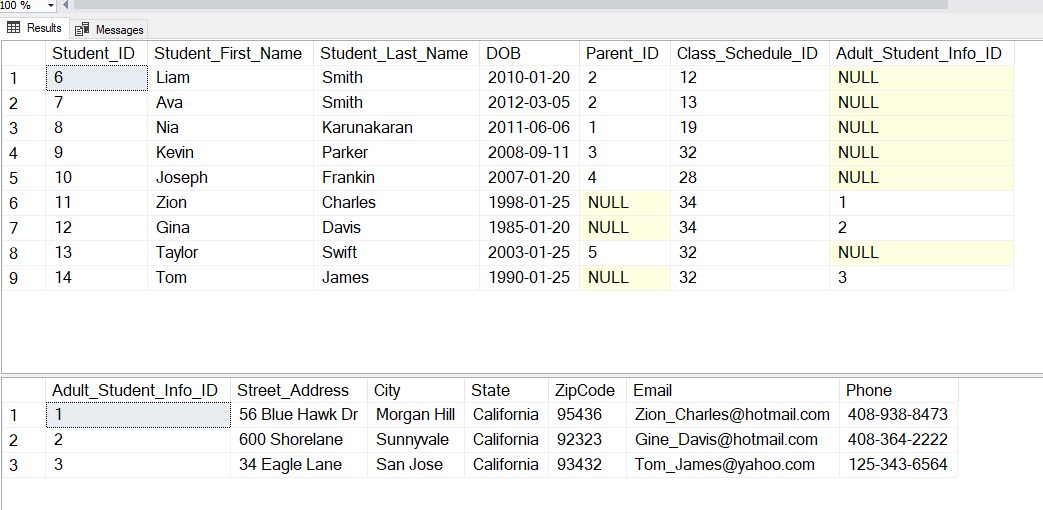
'Junior','America''s Best Karate','Tuesday','7:00pm'

--Let’s check if the record was inserted correctly. Since the student is an Adult, the ParentID in the Student table should be null

--and a record should be inserted in table Adult\_Student\_Info

select \* from Student

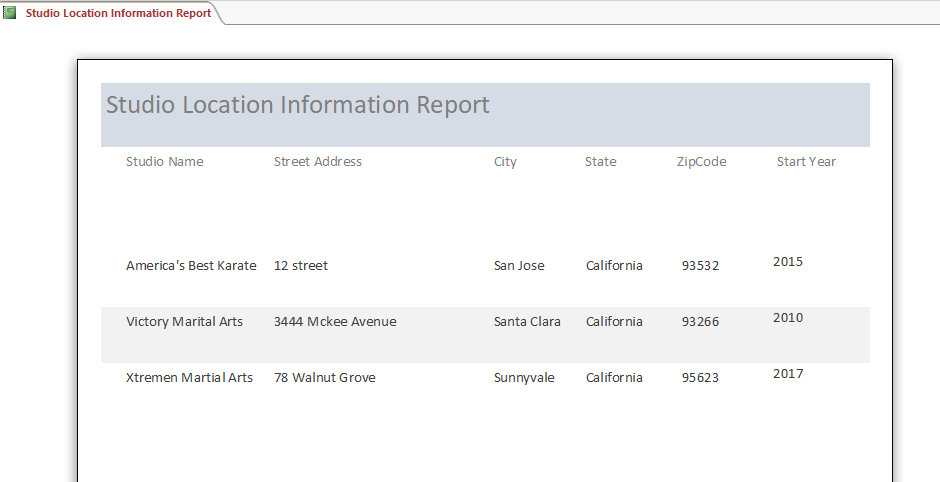
select \* from Adult\_Student\_Info



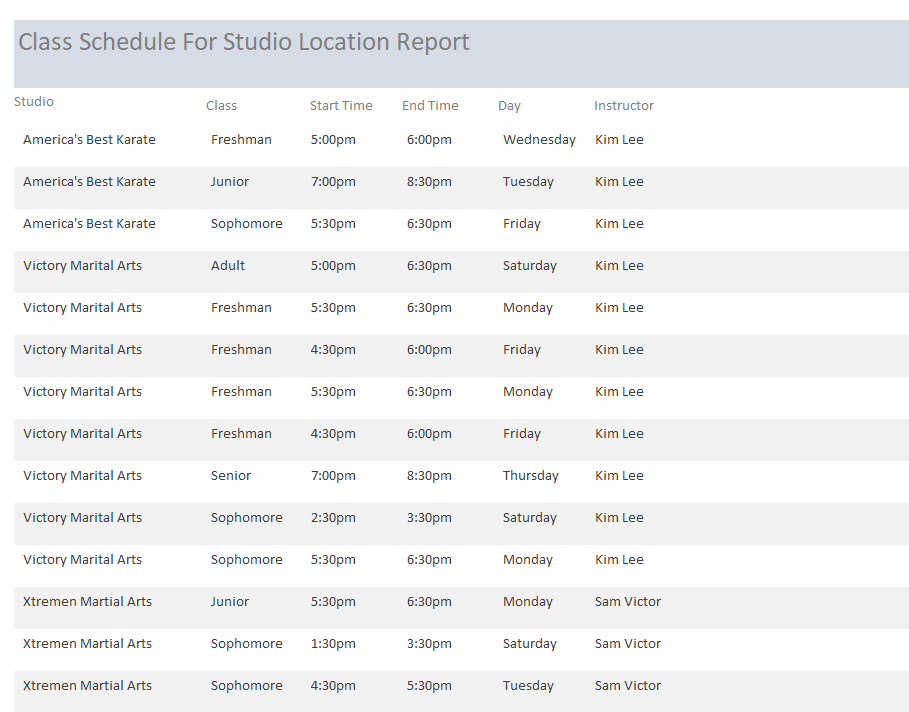
# IMPLEMENTATION

I have decided to MS Access to implement the Data Entry/Maintenance screens and Reports for my stakeholders. This was my first time using this tool so there was some learning curve involved but the tool itself proved to be user friendly and easy to use.

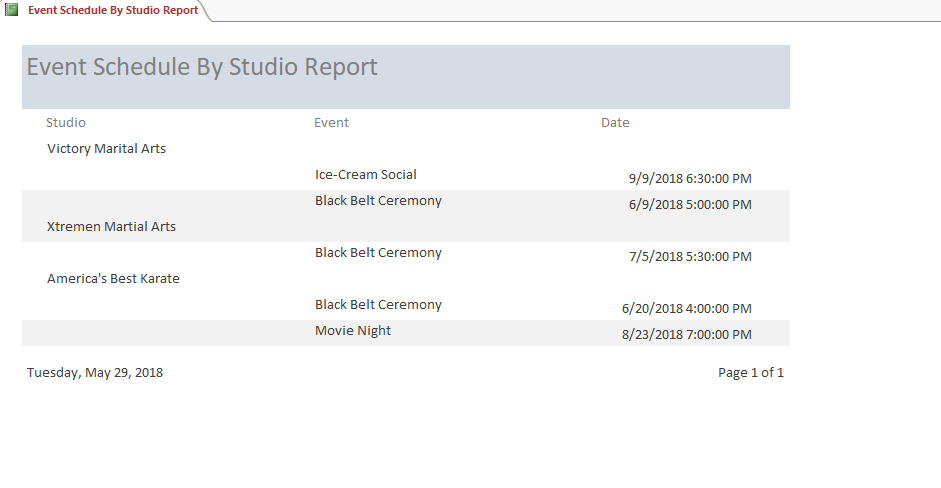
## STUDIO LOCATION INFORMATION ACCESS REPORT

****

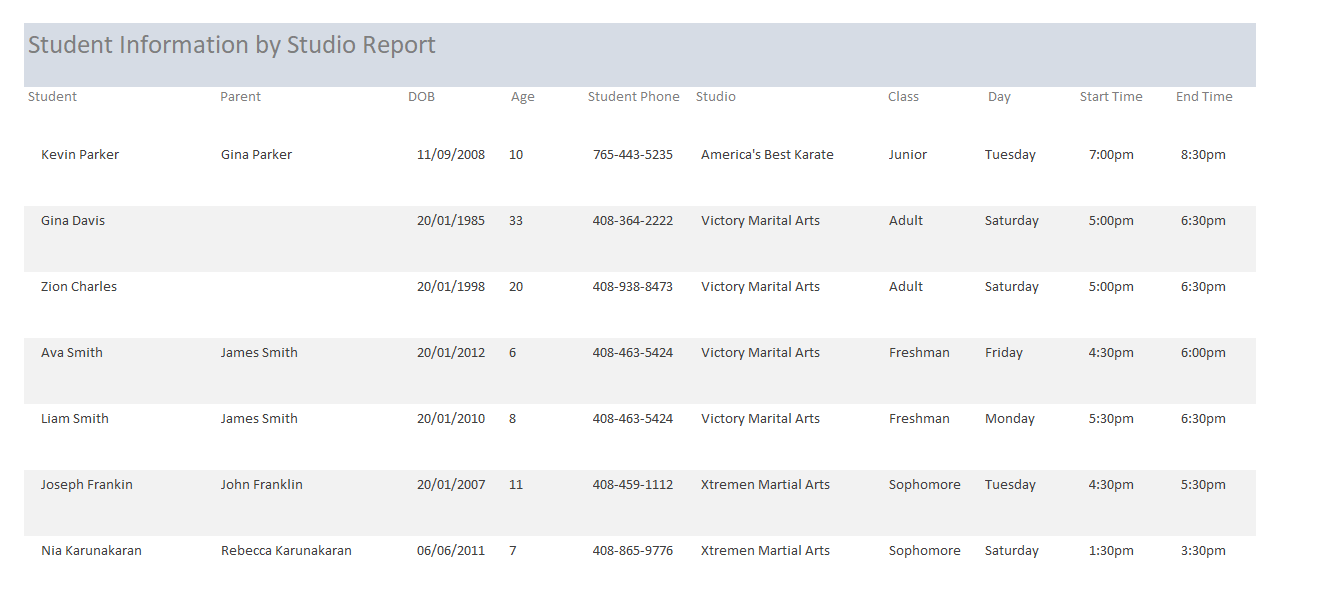
## CLASS SCHEDULE FOR STUDIO ACCESS REPORT

****

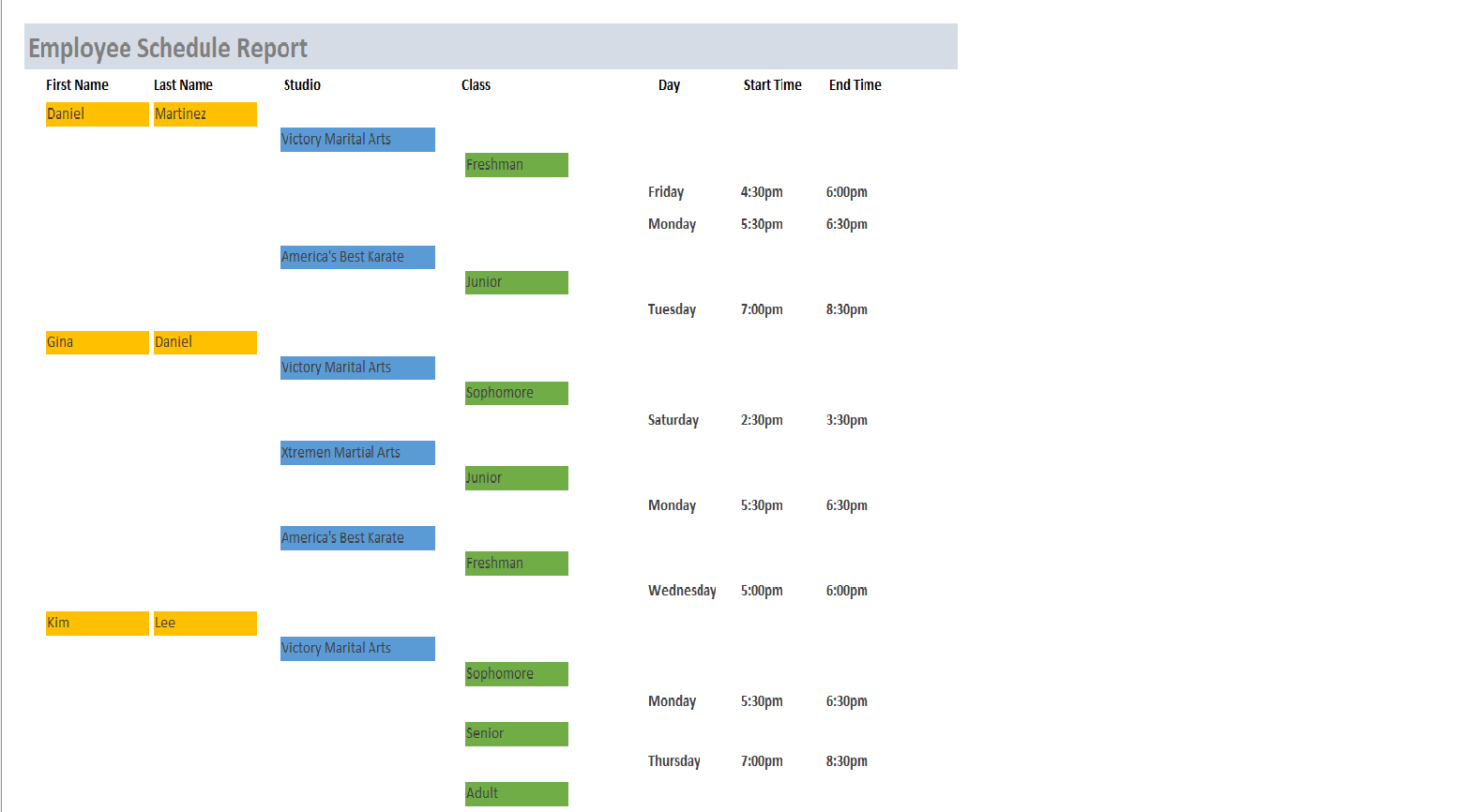
## EVENT SCHEDULE BY STUDIO ACCESS REPORT

****

## STUDIO INFORMATION BY STUDIO ACCESS REPORT

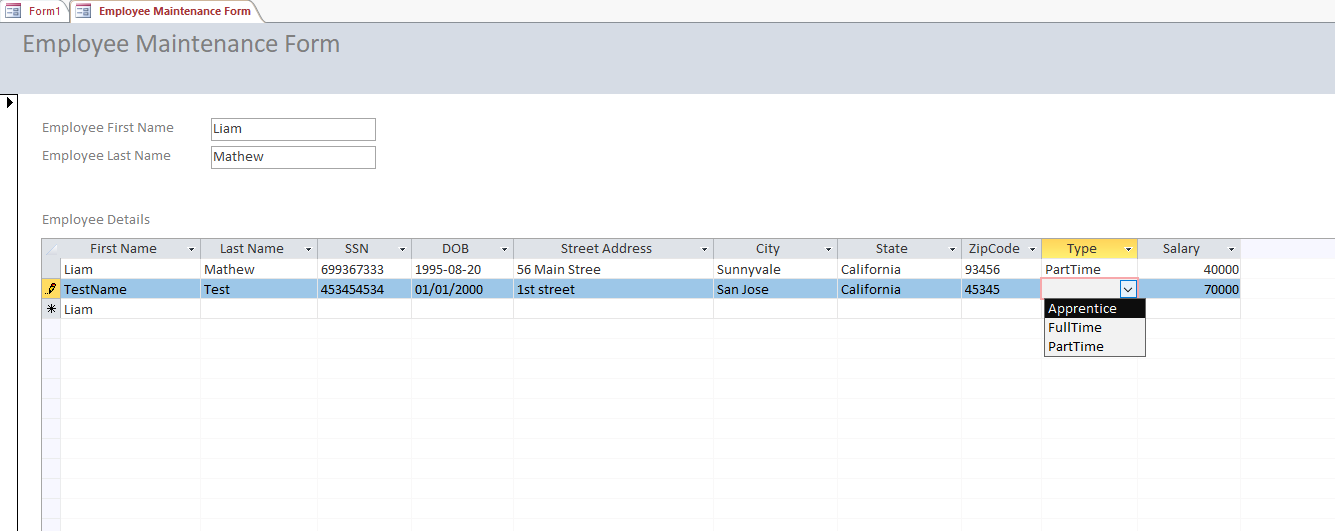
****

## Employee schedule ACCESS REPORT

****

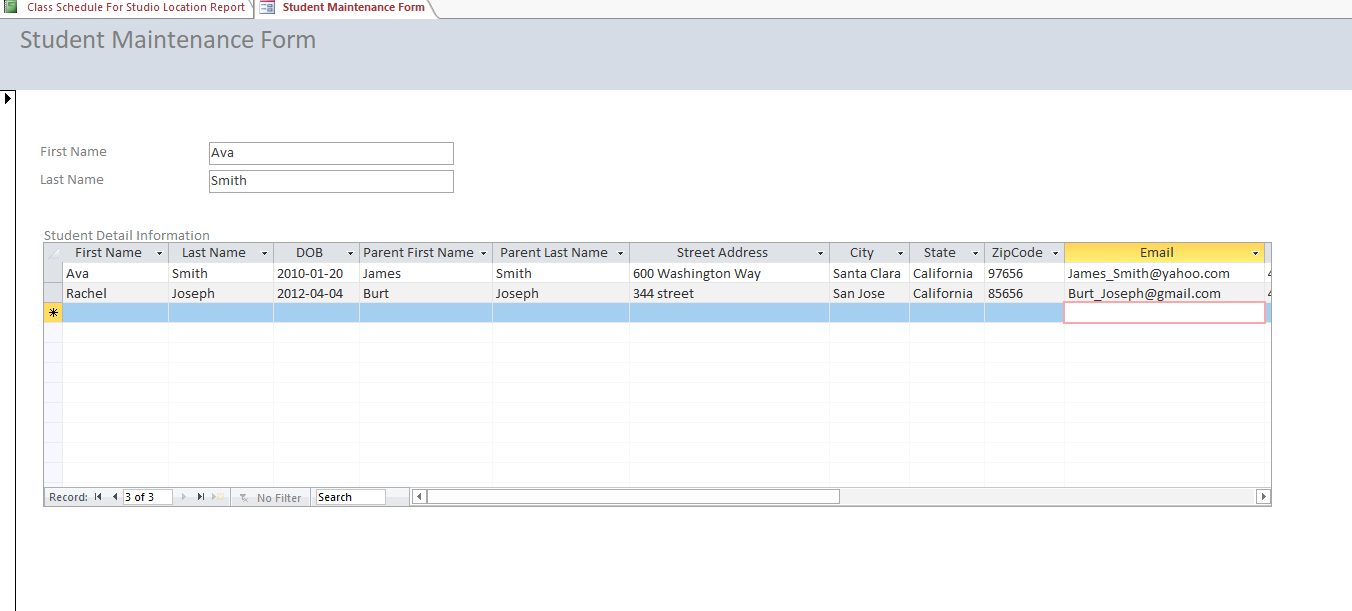
## Employee MAINTENANCE ACCESS form

MS Form to view existing employee as well as add new employees

****

## Student MAINTENANCE ACCESS form

MS Form to view existing students as well as add new students

****

# REFLECTION

Having prior knowledge and experience with RDBMS like MS SQL and Oracle it was nice to brush on the data modeling process. Here are some thoughts that I’d like to share

* As an IT professional I have noticed that most people don’t invest enough time to create the conceptual and normalized data model. This project has made me realize the importance of investing the time to create the data model which is the foundation to any database.
* Once the data model has been created, most people in the real world don’t update it when the business requirement changes. During the project, I had to make several changes to the data model but once it was updated, it helped me design and implement the database objects without having to go the database to see the table structure or relationships.
* Last but not least, this project has reiterated that the database is the foundation to any project and investing the time to get it right definitely pays off when it come to implementing the User Interface.