Project Deliverable

“Law Office”

Randall Scott Taylor

Syracuse University

Spring 2019

Dr. Gregory Block

CONTENTS

Project Summary:

The Law Office is newly formed within the two years and has established offices in the Financial District of Manhattan. The practice focuses on Criminal, Family, Torts, and Immigration law subject matters. There are systems that are being utilized in the firm, such as SaaS systems for legal referencing (Lexis), but the firm does not yet have its own central database.

The purpose of this Project is to convince stakeholders, in non-technical and technical demonstrations, that there is a need to create a centralized Database that can address some of the following problems:

PROBLEM(S) STATEMENT: Clients (how many, active clients or inactive clients), assignments (what clients are assigned to) Paralegals, Attorney(s). What Courts are these clients assigned to (how many different court cases, what type of case) and collection and assignment to each client the applicable Case Number and description. A firm calendar will also need to be established (How long did the Client/Case last, what is the average length of a Case). The calendar will be directly related to billing. The entry of discovery will also need to be attached to the Client records (How many items of Discovery were added; what is the average total of Discoverable items per client) as these correlates to hours worked and will need to be entered and tracked on the firm calendar. Finally, the tracking of Judicial Records will also need to be collected and stored.

Stakeholder Description:

The benefit of centralizing this data and being able to aggregately track its contents is a business value added action, that will benefit the following stakeholders:

Client: The preservation of the integrity of any case or cause of action is paramount within the judicial system. From a client stakeholder perspective, the centralization of such data is in their best interest from a legality standpoint, as well as from a business decision. Proper tracking as to the schedule, assignment, and elements of their respective cases all helps to expedite a very tenuous and stressful situation.

Legal Personnel *Paralegal*: The first person that the client stakeholder usually engages with, the Paralegal has a vital role that relates to the interactions of the Client stakeholder, all the way from case intake to scheduling. The establishment of a centralized database would be a massive value added to their business processes and would assist to expedite their support role within the litigation process.

Legal Personnel *Attorney*(s): Clients make appointments with their counsel and representation Paralegals work to sort out scheduling, judicial form generation, court room assistance, and billing. All these actions are such that the attorney can represent the interest of the client in the most productive way possible.

Glossary:

Contained below is a glossary of *Agents* and *Resources*: required attributes and their respective relationships, entity to entity.

Client: The C*lient* entity will require: client name, client address, and client e-mail address. One or many client(s) are assigned to Personnel. One to many Clients can owe hours or be billed (Billable) hours. Zero or one client can require Judicial Forms. One or many client have or, are assigned zero or many Discovery. Zero or many client can be scheduled to one or many Courts. Finally, One or many Client can be scheduled to one or many Firm Calendar entries, assignments, case numbers and descriptions, calendar entries, total hours billed, discovery, and judicial forms.

Personnel: The *Personnel* entity will require: Name, State Bar License Number, and hours owed. One or many Personnel is assigned or has one or many Clients. One or many Personnel can be scheduled to Firm Calendar. One or many hours owed Billing to one or many Personnel.

Court: The *Court* entity will require: Court Name, Court Number, Judicial Officer assigned, the hearing date, Hearing Description, and Case Number. One or Many Courts can be assigned to zero or many clients.

Firm Calendar: The *Firm Calendar* entity will require: Date and time entries. It will also require for an hour spent entry to be made on the corresponding date and time. One or many Firm Calendars can be scheduled to one or many Clients. One or many Firm Calendars can be scheduled to one or many Personnel.

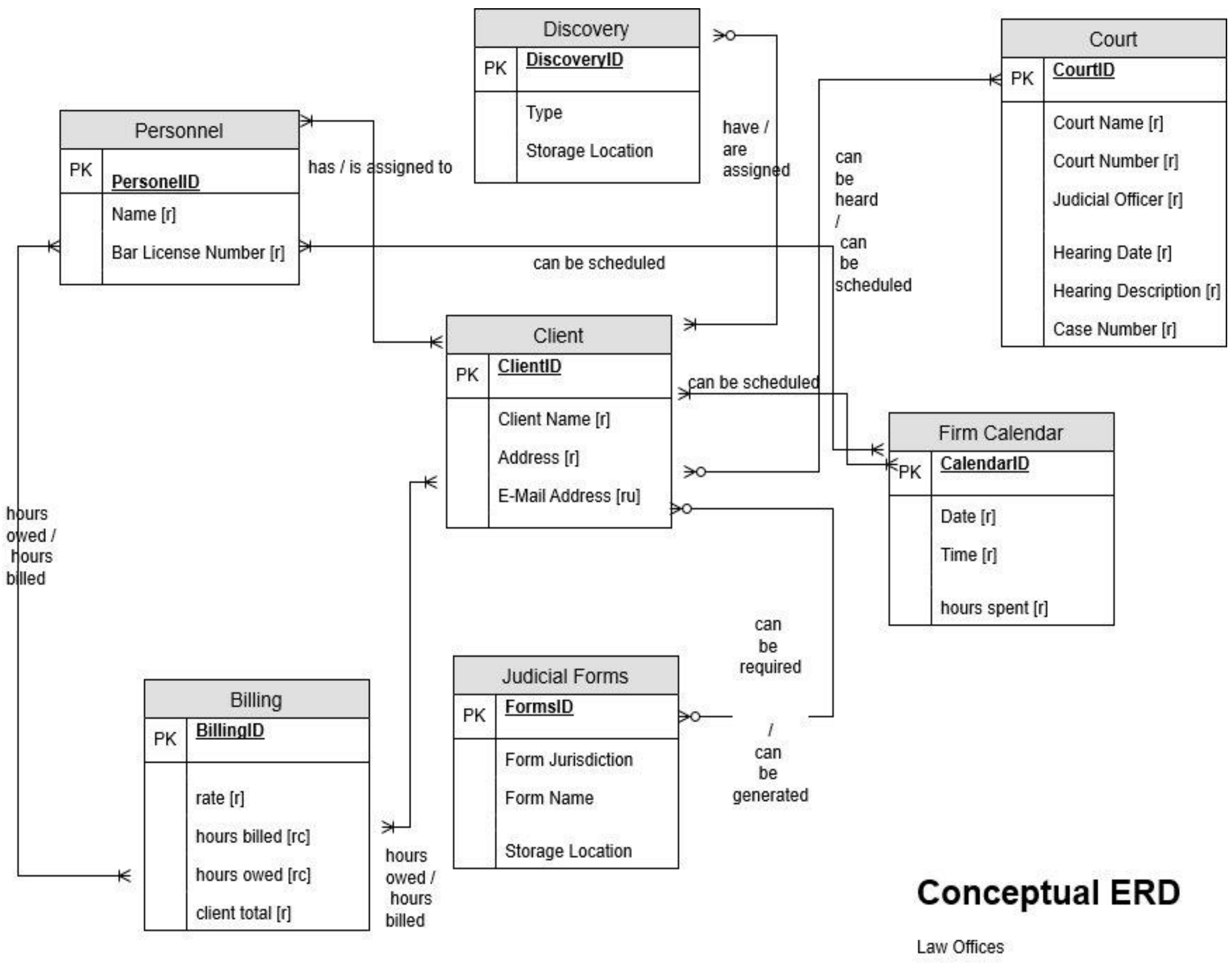
Billing: The *Billing* entity will require: rate (of personnel) along with hours billed (client) and hours owed(personnel). There will also be a client total hours entry required. One or many Billing hours can be billed to one or many client. One or many hours, at required specific rate, can be owed to one or many personnel.

Discovery: The *Discovery* entity will require: Type of discovery and discovery storage location entry. Zero or many Discovery are assigned to One or many clients.

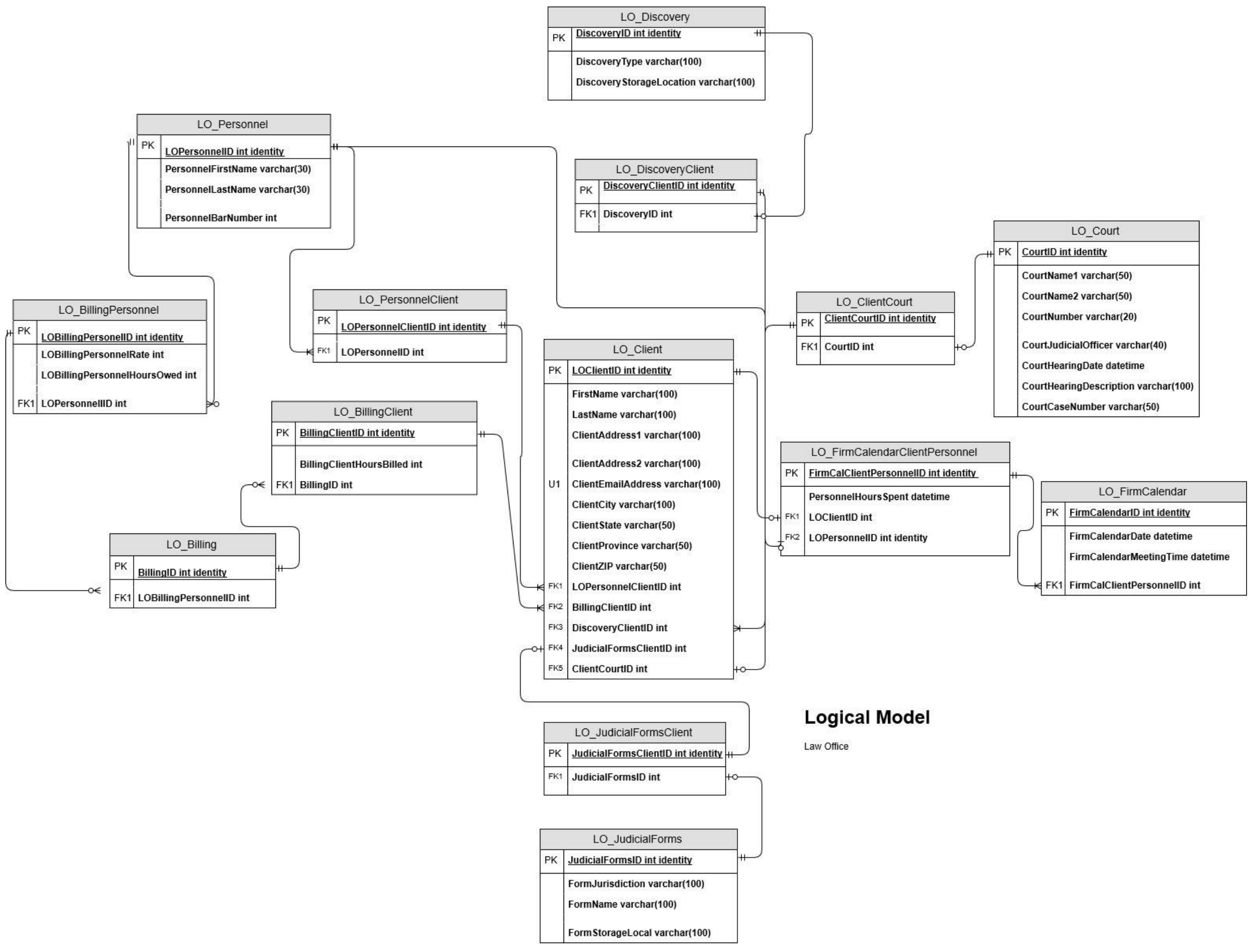
Judicial Forms: The *Judicial Forms* entity will require: Form Jurisdiction (what Court is the form from Civil or Criminal) the Form Name and the form storage location. Zero or many Judicial Forms can be required for zero or many Client.

Models: Conceptual and Logical.

The Conceptual Model: Law Office


Discovery
PK DiscoveryID
Type
Storage Location
Court
PK CourtID
Court Name [r]
Court Number [r]
Judicial Officer [r]
Hearing Date [r]
Hearing Description [r]
Case Number [r]
Personnel
PK PersonelID
Name [r]
Bar License Number [r]
has / is assigned to
have / are assigned
can be heard / can be scheduled
can be scheduled
Client
PK ClientID
Client Name [r]
Address [r]
E-Mail Address [ru]
can be scheduled
Firm Calendar
PK CalendarID
Date [r]
Time [r]
hours spent [r]
hours owed / hours billed
can be required / can be generated
Billing
PK BillingID
rate [r]
hours billed [rc]
hours owed [rc]
client total [r]
Judicial Forms
PK FormsID
Form Jurisdiction
Form Name
Storage Location
hours owed / hours billed
Conceptual ERD
Law Offices

The Logical Model: Law Office (UPDATED)


LO_Discovery
PK DiscoveryID int identity
DiscoveryType varchar(100)
DiscoveryStorageLocation varchar(100)
LO_Personnel
PK LOPersonnelID int identity
PersonnelFirstName varchar(30)
PersonnelLastName varchar(30)
PersonnelBarNumber int
LO_DiscoveryClient
PK DiscoveryClientID int identity
FK1 DiscoveryID int
LO_Court
PK CourtID int identity
CourtName1 varchar(50)
CourtName2 varchar(50)
CourtNumber varchar(20)
CourtJudicialOfficer varchar(40)
CourtHearingDate datetime
CourtHearingDescription varchar(100)
CourtCaseNumber varchar(50)
LO_BillingPersonnel
PK LOBillingPersonnelID int identity
LOBillingPersonnelRate int
LOBillingPersonnelHoursOwed int
FK1 LOPersonnelID int
LO_PersonnelClient
PK LOPersonnelClientID int identity
FK1 LOPersonnelID int
LO_ClientCourt
PK ClientCourtID int identity
FK1 CourtID int
LO_Client
PK LOClientID int identity
FirstName varchar(100)
LastName varchar(100)
ClientAddress1 varchar(100)
ClientAddress2 varchar(100)
U1 ClientEmailAddress varchar(100)
ClientCity varchar(100)
ClientState varchar(50)
ClientProvince varchar(50)
ClientZIP varchar(50)
FK1 LOPersonnelClientID int
FK2 BillingClientID int
FK3 DiscoveryClientID int
FK4 JudicialFormsClientID int
FK5 ClientCourtID int
LO_BillingClient
PK BillingClientID int identity
BillingClientHoursBilled int
FK1 BillingID int
LO_FirmCalendarClientPersonnel
PK FirmCalClientPersonnelID int identity
PersonnelHoursSpent datetime
FK1 LOClientID int
FK2 LOPersonnelID int identity
LO_FirmCalendar
PK FirmCalendarID int identity
FirmCalendarDate datetime
FirmCalendarMeetingTime datetime
FK1 FirmCalClientPersonnelID int
LO_Billing
PK BillingID int identity
FK1 LOBillingPersonnelID int
Logical Model
Law Office
LO_JudicialFormsClient
PK JudicialFormsClientID int identity
FK1 JudicialFormsID int
LO_JudicialForms
PK JudicialFormsID int identity
FormJurisdiction varchar(100)
FormName varchar(100)
FormStorageLocal varchar(100)

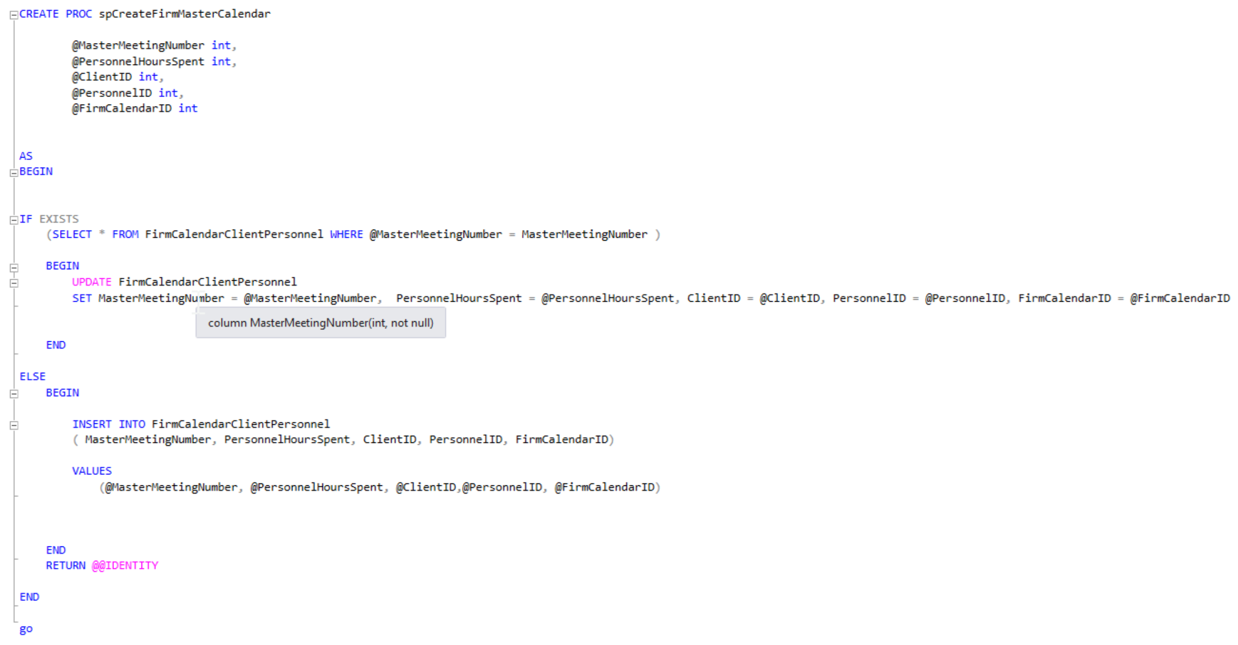
PROJECT MILESTONE II BEGINS:

Data Questions:

Having established the business need for the creation of a centralized database as established in the problem statement, we should now be able to ask the following questions about the student's data:

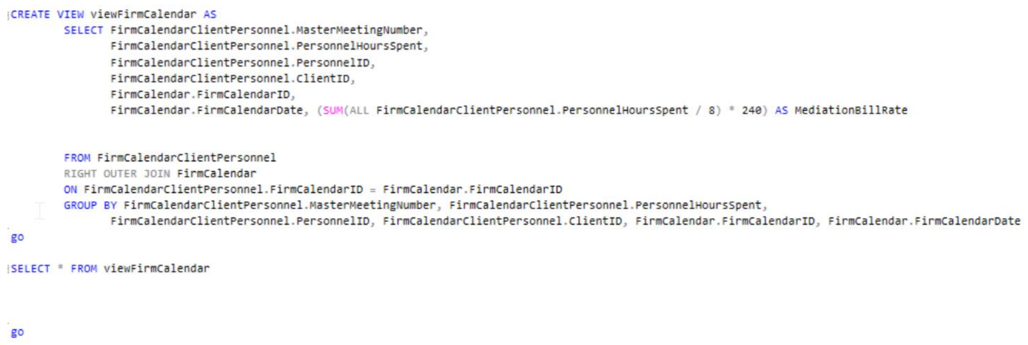
I. <u>Firm Calendar</u>: As a user, how does one demonstrate a Law Firm Master Calendar schedule, whereby a user can create/view TOTAL calendar personnel assignments?

a). Stored procedure `spCreateFirmMasterCalendar`has been created to create Calendar entries to the appropriate Master Calendar (named `FirmCalendarClientPersonnel`, not Master Calendar, as to not utilize the office term within the database) as seen below:



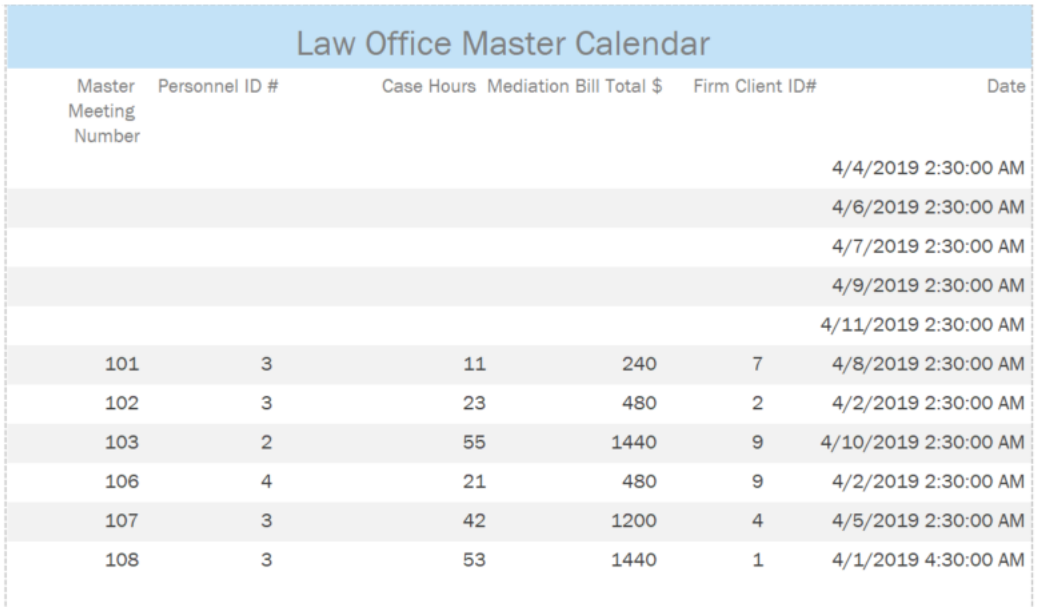
```
CREATE PROC spCreateFirmMasterCalendar

        @MasterMeetingNumber int,
        @PersonnelHoursSpent int,
        @ClientID int,
        @PersonnelID int,
        @FirmCalendarID int


AS
BEGIN


IF EXISTS
    (SELECT * FROM FirmCalendarClientPersonnel WHERE @MasterMeetingNumber = MasterMeetingNumber )

    BEGIN
        UPDATE FirmCalendarClientPersonnel
        SET MasterMeetingNumber = @MasterMeetingNumber,  PersonnelHoursSpent = @PersonnelHoursSpent, ClientID = @ClientID, PersonnelID = @PersonnelID, FirmCalendarID = @FirmCalendarID


    END

ELSE
    BEGIN

        INSERT INTO FirmCalendarClientPersonnel
        ( MasterMeetingNumber, PersonnelHoursSpent, ClientID, PersonnelID, FirmCalendarID)

        VALUES
            (@MasterMeetingNumber, @PersonnelHoursSpent, @ClientID,@PersonnelID, @FirmCalendarID)


    END
    RETURN @@IDENTITY

END

go
```

b). further a view `viewFirmCalendar` has been created to view those results, via SQL.



```
CREATE VIEW viewFirmCalendar AS
        SELECT FirmCalendarClientPersonnel.MasterMeetingNumber,
               FirmCalendarClientPersonnel.PersonnelHoursSpent,
               FirmCalendarClientPersonnel.PersonnelID,
               FirmCalendarClientPersonnel.ClientID,
               FirmCalendar.FirmCalendarID,
               FirmCalendar.FirmCalendarDate, (SUM(ALL FirmCalendarClientPersonnel.PersonnelHoursSpent / 8) * 240) AS MediationBillRate


        FROM FirmCalendarClientPersonnel
        RIGHT OUTER JOIN FirmCalendar
        ON FirmCalendarClientPersonnel.FirmCalendarID = FirmCalendar.FirmCalendarID
        GROUP BY FirmCalendarClientPersonnel.MasterMeetingNumber, FirmCalendarClientPersonnel.PersonnelHoursSpent,
               FirmCalendarClientPersonnel.PersonnelID, FirmCalendarClientPersonnel.ClientID, FirmCalendar.FirmCalendarID, FirmCalendar.FirmCalendarDate
go

SELECT * FROM viewFirmCalendar


go
```

c). finally, a report has been created with Access "Firm Master Calendar Report," to address this question, with aggregate of average hours displayed with, as seen below:

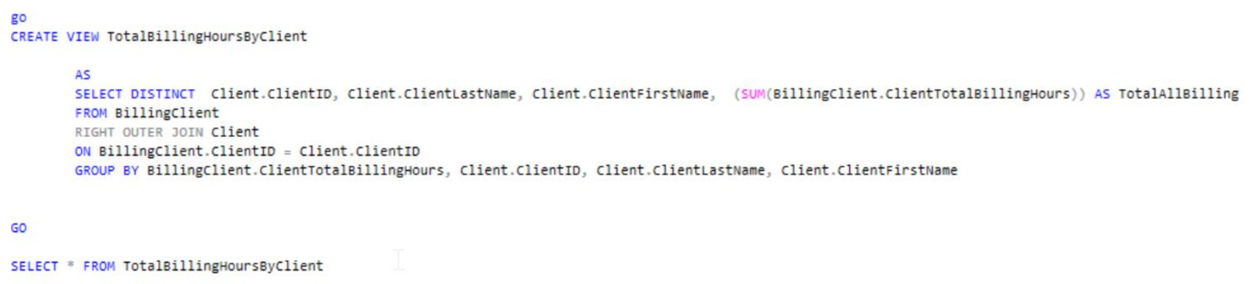


Law Office Master Calendar

Master Meeting Number	Personnel ID #	Case Hours	Mediation Bill Total $	Firm Client ID#	Date
					4/4/2019 2:30:00 AM
					4/6/2019 2:30:00 AM
					4/7/2019 2:30:00 AM
					4/9/2019 2:30:00 AM
					4/11/2019 2:30:00 AM
101	3	11	240	7	4/8/2019 2:30:00 AM
102	3	23	480	2	4/2/2019 2:30:00 AM
103	2	55	1440	9	4/10/2019 2:30:00 AM
106	4	21	480	9	4/2/2019 2:30:00 AM
107	3	42	1200	4	4/5/2019 2:30:00 AM
108	3	53	1440	1	4/1/2019 4:30:00 AM

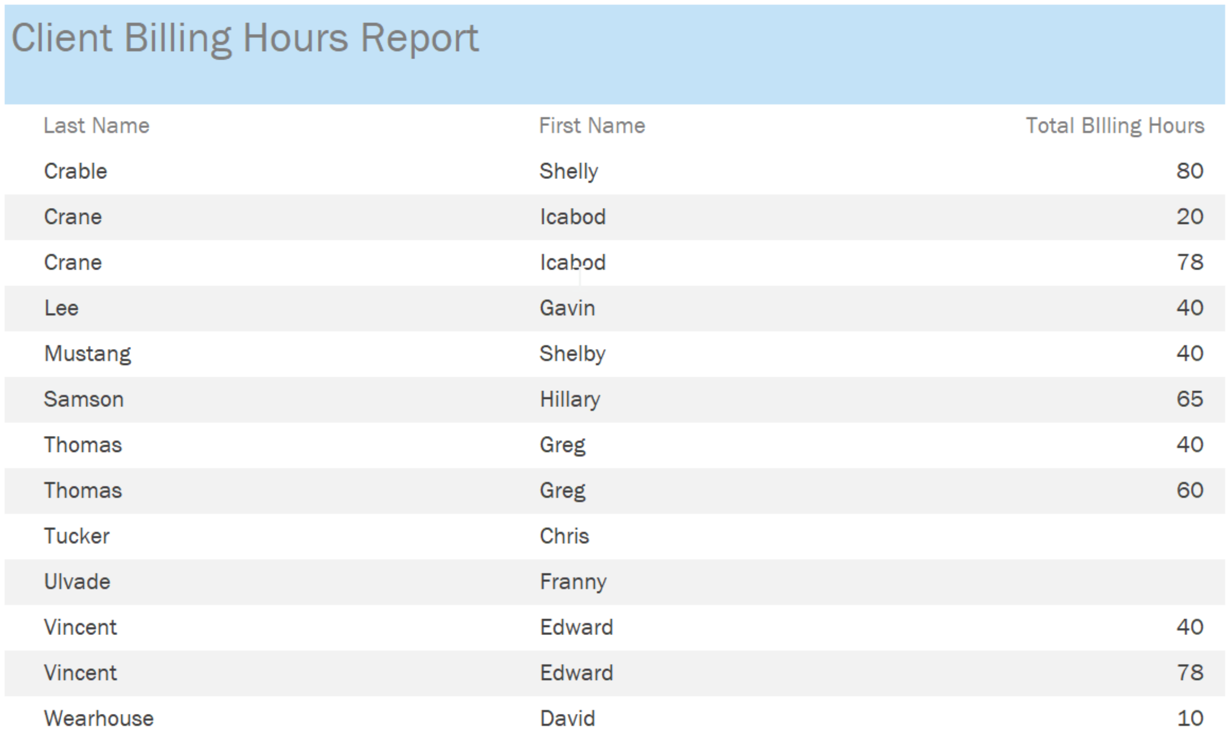
II. <u>Billing</u>: is essential and knowing how much is owed is important for revenue and payroll computation. As a user, how can one view the total bill for clients, the total monies owed and for how many hours?

a). a view was created within the database, `TotalBillingHoursByClient` to answer this data question, as seen below:



```
go
CREATE VIEW TotalBillingHoursByClient

        AS
        SELECT DISTINCT  Client.ClientID, Client.ClientLastName, Client.ClientFirstName,  (SUM(BillingClient.ClientTotalBillingHours)) AS TotalAllBilling
        FROM BillingClient
        RIGHT OUTER JOIN Client
        ON BillingClient.ClientID = Client.ClientID
        GROUP BY BillingClient.ClientTotalBillingHours, Client.ClientID, Client.ClientLastName, Client.ClientFirstName


GO

SELECT * FROM TotalBillingHoursByClient
```

b). a report has been created within Access to show the details per client name, how much total billing is owed, and for how many hours, as seen below:

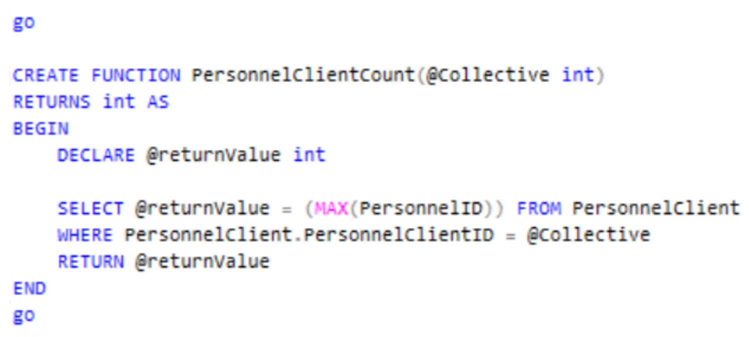


Client Billing Hours Report

Last Name	First Name	Total BIlling Hours
Crable	Shelly	80
Crane	Icabod	20
Crane	Icabod	78
Lee	Gavin	40
Mustang	Shelby	40
Samson	Hillary	65
Thomas	Greg	40
Thomas	Greg	60
Tucker	Chris	
Ulvade	Franny	
Vincent	Edward	40
Vincent	Edward	78
Wearhouse	David	10

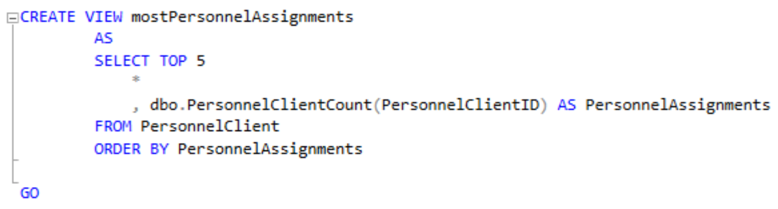
III. <u>Legal Personnel Assignment</u>: it is important to not only view the Firm Calendar for scheduling purposes, but, a report will be needed for Personnel Assignments; to evenly disperse clients. As a user, how can one determine the amount of personnel assignments each personnel member has, and who has the least, the most?

a). Function `PersonnelClientCount` was created to answer this question via the SQL, as seen below:



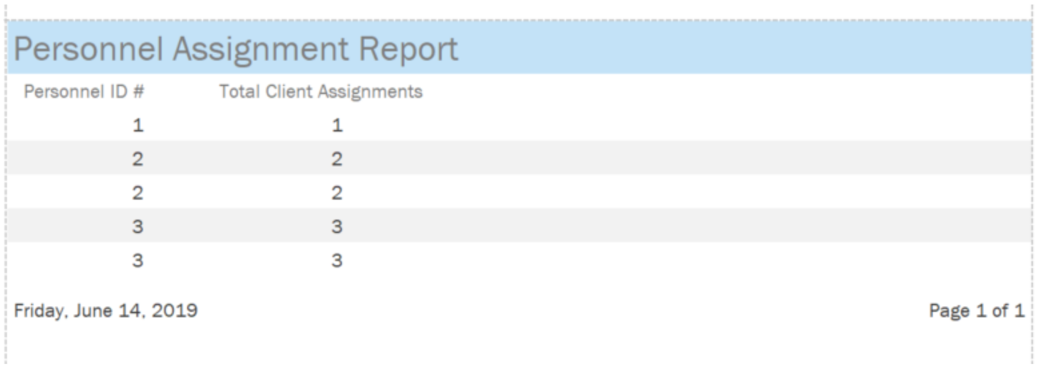
```
go

CREATE FUNCTION PersonnelClientCount(@Collective int)
RETURNS int AS
BEGIN
    DECLARE @returnValue int

    SELECT @returnValue = (MAX(PersonnelID)) FROM PersonnelClient
    WHERE PersonnelClient.PersonnelClientID = @Collective
    RETURN @returnValue
END
go
```

b). To further assist in answering this data question, View `mostPersonnelAssignments` was created that invokes the `PersonnelClientCount` Function, as seen below:



```
CREATE VIEW mostPersonnelAssignments
        AS
        SELECT TOP 5
            *
            , dbo.PersonnelClientCount(PersonnelClientID) AS PersonnelAssignments
        FROM PersonnelClient
        ORDER BY PersonnelAssignments

GO
```

c). To assist with staff, a report has been created within the UI, Personnel Assignment List, as seen below:

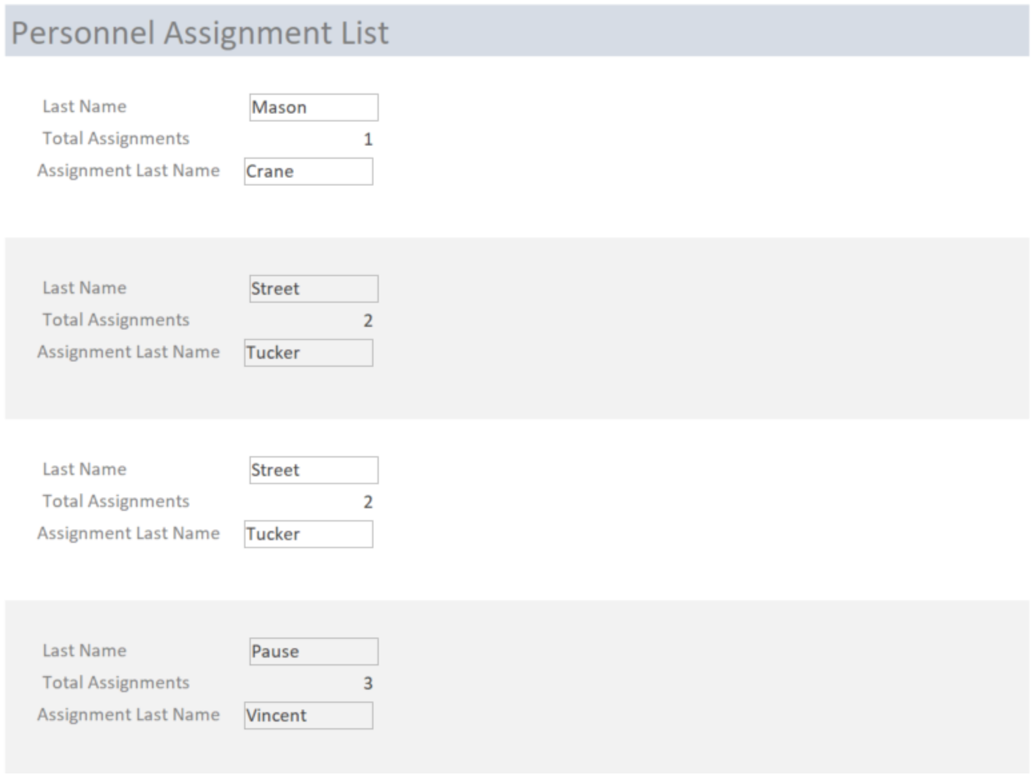


Personnel Assignment Report

Personnel ID #	Total Client Assignments
1	1
2	2
2	2
3	3
3	3

Friday, June 14, 2019 Page 1 of 1

d). Finally, a Query was formed to total the amount of client assignments per Personnel/Client assignments, to Personnel, as seen below

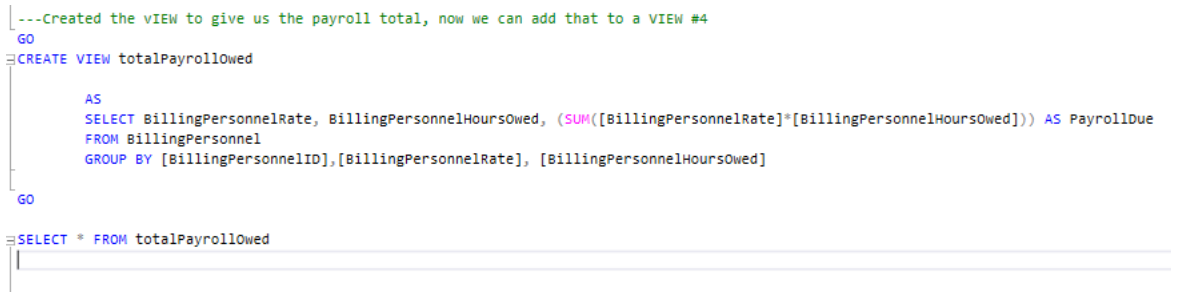


Personnel Assignment List

Last Name	Mason
Total Assignments	1
Assignment Last Name	Crane
Last Name	Street
Total Assignments	2
Assignment Last Name	Tucker
Last Name	Street
Total Assignments	2
Assignment Last Name	Tucker
Last Name	Pause
Total Assignments	3
Assignment Last Name	Vincent

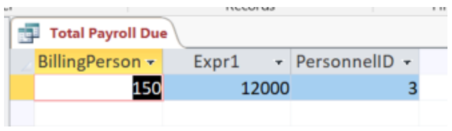
IV. Payroll Reconciliation: is the name of the game, extract the total billing hours tied to a client, and then charging them the rate of $150.00 per hour (or whatever Payrate is applicable). As a user, how can one collect: total billing hours, for which client and, how much that client owes for payment?

a). Having created the view to assist, a View `totalPayrollOwed` was created to display the information, in a non-editable format, as seen below:

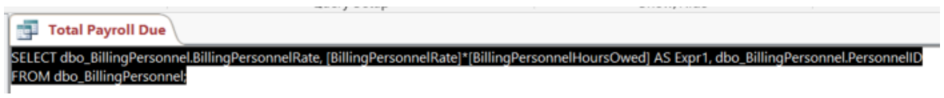


```
---Created the vIEW to give us the payroll total, now we can add that to a VIEW #4
GO
CREATE VIEW totalPayrollOwed

        AS
        SELECT BillingPersonnelRate, BillingPersonnelHoursOwed, (SUM([BillingPersonnelRate]*[BillingPersonnelHoursOwed])) AS PayrollDue
        FROM BillingPersonnel
        GROUP BY [BillingPersonnelID],[BillingPersonnelRate], [BillingPersonnelHoursOwed]

GO

SELECT * FROM totalPayrollOwed
```

b). Having created the view to assist, a query was created in Access to display the information and aggregate totals, as seen below:

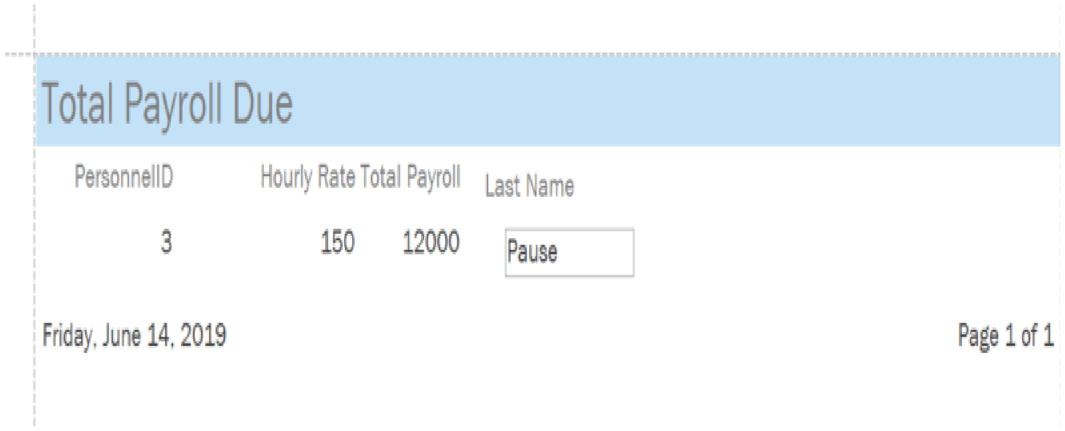


BillingPerson	Expr1	PersonnelID
150	12000	3



```
SELECT dbo_BillingPersonnel.BillingPersonnelRate, [BillingPersonnelRate]*[BillingPersonnelHoursOwed] AS Expr1, dbo_BillingPersonnel.PersonnelID
FROM dbo_BillingPersonnel;
```

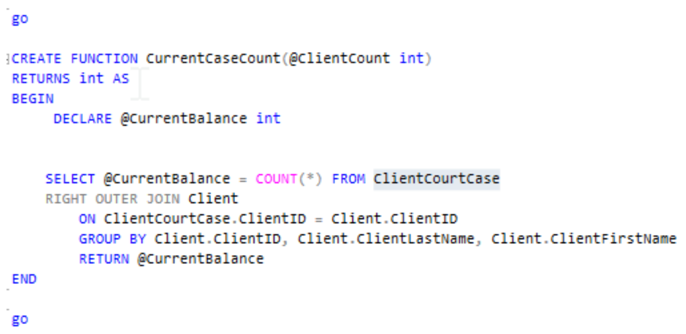
c). Finally, a report has been created within the UI, for easy user interface / display:



PersonnelID	Hourly Rate	Total Payroll	Last Name
3	150	12000	Pause

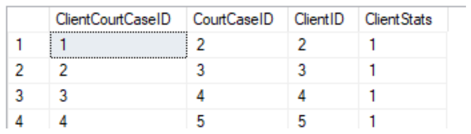
V. <u>Case Management</u>: To manage all resources efficiently, case management reporting will need to be created to list current case load, but also current case count per client. As a user, how can one: create a report that will collect all current cases per client, and how many active cases, per client?

a). To answer this data question, a Function `CurrentCaseCount` has been created to count all within the `ClientCourtCase,` as seen below



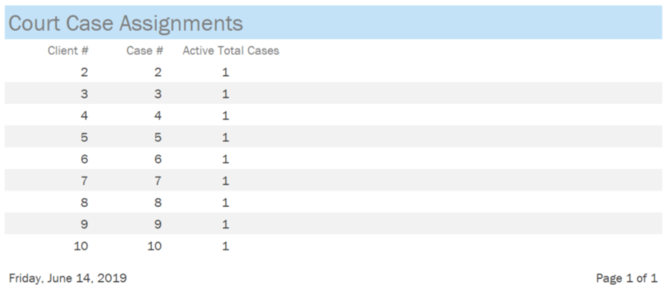
```
go

CREATE FUNCTION CurrentCaseCount(@ClientCount int)
RETURNS int AS
BEGIN
     DECLARE @CurrentBalance int


    SELECT @CurrentBalance = COUNT(*) FROM ClientCourtCase
    RIGHT OUTER JOIN Client
        ON ClientCourtCase.ClientID = Client.ClientID
        GROUP BY Client.ClientID, Client.ClientLastName, Client.ClientFirstName
        RETURN @CurrentBalance
END

go
```

b). Further, a View has been created `ClientCourtCaseAssignments` that references the above referenced function and displays that case assignments to personnel, as ClientStats, as seen below:



	ClientCourtCaseID	CourtCaseID	ClientID	ClientStats
1	1	2	2	1
2	2	3	3	1
3	3	4	4	1
4	4	5	5	1

c). So that staff can allocate resources accordingly, a report has been created in Access to display this information in a user-friendly format, as seen below:

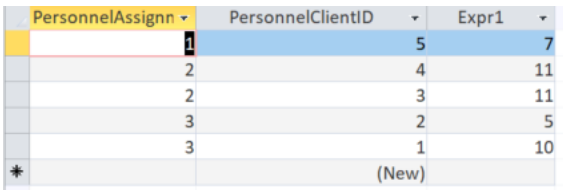


Court Case Assignments

Client #	Case #	Active Total Cases
2	2	1
3	3	1
4	4	1
5	5	1
6	6	1
7	7	1
8	8	1
9	9	1
10	10	1

Friday, June 14, 2019 Page 1 of 1

d). Finally, a query has been established such that Personnel Assignments through the Personnel Client Table can be calculated, in total active cases, total, as seen below:



PersonnelAssignn	PersonnelClientID	Expr1
1	5	7
2	4	11
2	3	11
3	2	5
3	1	10
	(New)	

<u>Reflection:</u>

I had no idea what I was doing, nor, what I was getting into … third day into this class.

The above is the best way one can summarize, in a simple statement as to how at the beginning of this project, the researcher was/is a complete and blundering novice. The researcher documents this in the reflection, not as a statement with negative connotations, but rather, with hope for future course works in the subject manner, as the progress thusly made, from April to now, is – amazing.

The researcher still feels like a blundering novice in the subject, however, it is with happy reflection that it can be stated that it is becoming more of a comfortable subject matter, indeed each small victory, correct syntax used, remembering the correct forms for views, functions – all of the elements required to do basic DDL coding, tables creation, validation, bug testing – all of this is a new concept, all of it was completely foreign to the researcher at the beginning of this course, and thankfully now. is not such a foreign concept.

The manner in which ERD Cardinality functions, the concept of bridge tables, the logical model – these are ways of perceiving information that I had never once considered. I never considered that there were levels of normal form, three standards (that we've learned) about, let alone one standard form, had never once entered my vernacular, or mind mappings.

My assumptions from the start of the project, having now painted a picture of my lack of understanding regarding SQL, were very incorrect. I had 'application' 'app' in my mind the whole of the beginning of the class, without an understanding as to: how the tables related to one another, nor, how the transacted data. This is something that I am still learning and evolving within, but the hands-on approach of the coursework, helped to cement a grasp on the beginning of these concepts. For instance, the data questions; mine evolved, as I am sure others found as well. What to ask and how to ask it; aggregate versus just lists. These are items that have changed and evolved along the instruction route.

Thee are aspects of this project that I would have liked to change. In an attempt to ensure complete and total adherence to the table requirements, and further, knowing/understanding the business process within the ‘Law Office’ caused a bit of over zealousness in the creation of tables. For instance, the Discovery table, Judicial Forms table, their respective bridge tables to the Client table, all work, all functioning; all completely not used in the project and thus not necessary – other than to prove that the student could successfully create tables and think outside the box of a small system and envision an entire database office solution. In retrospect, a simple case management / calendar database would have been most likely sufficient model, or even a judicial forms database would have sufficed. These items stated, I think it was rather bold to take a big bite out of this project; nothing is worth doing unless you do it full, and correct.

While this project, at the end of the day, may not be one hundred percent correct in its findings, the researcher is extremely please with the learning experience, the individual finds themselves thinking in SQL, and that is an extremely interesting result of the time spent learning the material.

...here ends the reflection…

Tables

Column Name	Type	Properties	Descriptions

Personnel

Columns

PersonnelID	int IDENTIT	NOT NULL	PK	
PersonnelFirstName	varchar(30)	NOT NULL		
PersonnelLastName	varchar(30)	NOT NULL		
PersonnelBarNumber	varchar(30)			

Billing

Columns

BillingID	int IDENTITY	NOT NULL	PK	
BillingClientLastName	varchar(100)			
BillingClientFirstName	varchar(100)			
BillingDescription1	varchar(100)			
BillingDescription2	varchar(100)			
BillingDate	DATETIME			

JudicialForms

Columns

JudicialFormsID	int IDENTITY	NOT NULL	PK	
FormJurisdiction	varchar(100)	NOT NULL		
FormName	varchar(100)	NOT NULL		
FormStorageLocal	varchar(100)	NOT NULL		

Discovery

Columns

DiscoveryID	int IDENTITY	NOT NULL	PK	
DiscoveryType	varchar(100)	NOT NULL		
DiscoveryStorageLocation	varchar(100)	NOT NULL		

Client

Columns

ClientID	int IDENTITY	NOT NULL	PK	
ClientFirstName	varchar(100)	NOT NULL		
ClientLastName	varchar(100)	NOT NULL		
ClientAddress1	varchar(50)	NOT NULL		
ClientAddress2	varchar(50)			
ClientEmailAddress	varchar(100)	NOT NULL	U1	
ClientCity	varchar(100)			
ClientState	varchar(100)			
ClientProvince	varchar(50)			
ClientZIP	varchar(20)			

CourtCase

Columns

CourtCaseID	int IDENTITY	NOT NULL	PK	
CourtName1	varchar(50)	NOT NULL		
CourtName2	varchar(50)	NOT NULL		
CourtNumber	varchar(20)	NOT NULL		
CourtJudicialOfficer1	varchar(40)	NOT NULL		
CourtJudicialOfficer2	varchar(40)	NOT NULL		
CourtCaseNumber	varchar(50)	NOT NULL		

FirmCalendar

Columns

FirmCalendarID	int IDENTITY	NOT NULL	PK	
FirmCalendarDate	DATETIME	NOT NULL		
FirmCalendarMeetingTime	varchar(50)	NOT NULL		
FirmCalendarMeetingInfo	varchar(100)	NOT NULL		

CourtHearingDate

Columns

CourtHearingDateID	int IDENTITY	NOT NULL		
CourtHearingDateDescription	varchar(100)	NOT NULL		
CourtHearingHeard	DATETIME	NOT NULL		

This concludes the main tables of Law Office, the following are all reference tables

Reference Tables

Column Name	Type	Properties	Descriptions

BillingPersonnel

Columns

BillingPersonnelID	int IDENTITY	NOT NULL	PK	
BillingPersonnelRate	int	NOT NULL		
BillingPersonnelHoursOwed	int	NOT NULL		
PersonnelID	int	NOT NULL	U1, FK	
BillingID	int	NOT NULL	U2, FK	

PersonnelClient

Columns

BillingClientID	int IDENTITY	NOT NULL	PK	
ClientTotalBillingHours	int	NOT NULL		
BillingID	int	NOT NULL	FK	
ClientID	int	NOT NULL	FK	

BillingClient

Columns

BillingClientID	int IDENTITY	NOT NULL	PK	
ClientTotalBillingHours	int	NOT NULL		
BillingID	int	NOT NULL	FK	
ClientID	int	NOT NULL	FK	

JudicialFormsClient

Columns

JudicialFormsClientID	int IDENTITY	NOT NULL	PK	
JudicialFormsID	int	NOT NULL		
ClientID	int	NOT NULL	FK	

DiscoveryClient

Columns

DiscoveryClientID	int IDENTITY	NOT NULL	PK	
DiscoveryID	int	NOT NULL	FK	
ClientID	int	NOT NULL	FK	

ClientCourtCase

Columns

ClientCourtCaseID	int IDENTITY	NOT NULL	PK, U1	
CourtCaseID	int	NOT NULL	FK	
ClientID	int (50)	NOT NULL	FK	

FirmCalendarClientPersonnel

Columns

FirmCalendarClientPersonnelID	int IDENTITY	NOT NULL	PK	*Master Calendar
MasterMeetingNumber	int	NOT NULL		
PersonnelHoursSpent	int			
ClientID	int		FK	
PersonnelID	int		FK	
FirmCalendarID	int		FK	

CourtCaseHearingDate

Columns

CourtCaseHearingDateID	int IDENTITY	NOT NULL	PK	
CourtCaseID	DATETIME	NOT NULL	FK	
CourtHearingDateID	varchar(50)	NOT NULL	FK	

FirmCalendarCourtHearingDate

Columns

FirmCalendarCourtHearingDateID	int IDENTITY	NOT NULL	PK	
FirmCalendarID	int	NOT NULL	FK	
CourtHearingDateID	int	NOT NULL	FK	

This concludes all reference tables of Law Office

PHYSICAL DATABASE DESIGN:

DDL CODE: Tables

```
/*
       This concludes the drop database objects;

       Now; create TABLES in DEPENDENCY ORDER: Create INDEPENDENT TABLES FIRST, then
TABLES that
       are DEPENDENT.

*/

--Law Office

CREATE TABLE Personnel
(
       PersonnelID int IDENTITY NOT NULL,
       PersonnelFirstName varchar(30) NOT NULL,
       PersonnelLastName varchar(30) NOT NULL,
       PersonnelBarNumber varchar(30),
       CONSTRAINT PersonnelPK PRIMARY KEY (PersonnelID)

);
go

CREATE TABLE Billing
(
       BillingID int IDENTITY NOT NULL,
       BillingClientLastName varchar(100),
       BillingClientFirstName varchar(100),
       BillingDescription1 varchar(100),
       BillingDescription2 varchar(100),
       BillingDate DATETIME,
       CONSTRAINT BillingPK PRIMARY KEY (BillingID)

);
go

CREATE TABLE JudicialForms
(
       JudicialFormsID int IDENTITY NOT NULL,
       FormJurisdiction varchar(100) NOT NULL,
       FormName varchar(100) NOT NULL,
       FormStorageLocal varchar(100) NOT NULL,
       CONSTRAINT JudicialFormsPK PRIMARY KEY (JudicialFormsID)

);
go

-- All items tested correctly

CREATE TABLE Discovery
(
       DiscoveryID int IDENTITY NOT NULL,
       DiscoveryType varchar(100) NOT NULL,
       DiscoveryStorageLocation varchar(100) NOT NULL,
	CONSTRAINT DiscoveryPK PRIMARY KEY (DiscoveryID)

);
go

CREATE TABLE Client
(
	ClientID int IDENTITY NOT NULL,
	ClientFirstName varchar(100) NOT NULL,
	ClientLastName varchar(100) NOT NULL,
	ClientAddress1 varchar(50) NOT NULL,
	ClientAddress2 varchar(50),
	ClientEmailAddress varchar(100) NOT NULL,
	ClientCity varchar(100) ,
	ClientState varchar(100) ,
	ClientProvince varchar(50),
	ClientZIP varchar(20) ,
	CONSTRAINT ClientPK PRIMARY KEY (ClientID),
	CONSTRAINT ClientEmailAddressU1 UNIQUE (ClientEmailAddress),

);
go

CREATE TABLE CourtCase
(
	CourtCaseID int IDENTITY NOT NULL,
	CourtName1 varchar(50) NOT NULL,
	CourtName2 varchar (50) NOT NULL,
	CourtNumber varchar(20) NOT NULL,
	CourtJudicialOfficer1 varchar(40) NOT NULL,
	CourtJudicialOfficer2 varchar(40),
	CourtCaseNumber varchar(50) NOT NULL,
	CONSTRAINT CourtCasePK PRIMARY KEY (CourtCaseID)

);
go

CREATE TABLE FirmCalendar
(
	FirmCalendarID int IDENTITY NOT NULL,
	FirmCalendarDate DATETIME NOT NULL,
	FirmCalendarMeetingTime varchar(50) NOT NULL,
	FirmCalendarMeetingInfo varchar(100) NOT NULL,
	CONSTRAINT FirmCalendarPK PRIMARY KEY (FirmCalendarID)

);
go

CREATE TABLE CourtHearingDate
(
	CourtHearingDateID int IDENTITY NOT NULL,
	CourtHearingDateDescription varchar(100) NOT NULL,
	CourtHearingHeard DATETIME NOT NULL,
	CONSTRAINT CourtHearingDatePK PRIMARY KEY (CourtHearingDateID)

);
Go  -- All items tested correctly
```

DDL CODE: Reference table creation

```
-- All items tested correctly



-- It is at this point that all Dependent Tables will be created

CREATE TABLE BillingPersonnel
(
	BillingPersonnelID int IDENTITY NOT NULL,
	BillingPersonnelRate int NOT NULL,
	BillingPersonnelHoursOwed int NOT NULL,
	PersonnelID int NOT NULL,
	BillingID int NOT NULL,
	CONSTRAINT PersonnelIDU1 UNIQUE(PersonnelID),
	CONSTRAINT BilingIDU2 UNIQUE(BillingID),
	CONSTRAINT BillingPersonnelPK PRIMARY KEY (BillingPersonnelID),
	CONSTRAINT BillingPersonnelFK1 FOREIGN KEY (PersonnelID) REFERENCES Personnel
(PersonnelID),
	CONSTRAINT BillingPersonnelFK2 FOREIGN KEY (BillingID) REFERENCES
Billing(BillingID)

);
go

CREATE TABLE PersonnelClient
(
	PersonnelClientID int IDENTITY NOT NULL,
	PersonnelID int NOT NULL,
	ClientID int NOT NULL,
	CONSTRAINT PersonnelClientPK PRIMARY KEY(PersonnelClientID),
	CONSTRAINT PersonnelClient_FK1 FOREIGN KEY (PersonnelID) REFERENCES
Personnel(PersonnelID),
	CONSTRAINT PersonnelClient_FK2 FOREIGN KEY (ClientID) REFERENCES Client(ClientID)
);
go

CREATE TABLE BillingClient
(
	BillingClientID int IDENTITY NOT NULL,
	ClientTotalBillingHours int,
	BillingID int NOT NULL,
	ClientID int NOT NULL,
	CONSTRAINT BillingClientPK PRIMARY KEY (BillingClientID),
	CONSTRAINT BillingClient_FK1 FOREIGN KEY (BillingID) REFERENCES
Billing(BillingID),
	CONSTRAINT BillingClient_FK2 FOREIGN KEY(ClientID) REFERENCES Client(ClientID),


);
go

CREATE TABLE JudicialFormsClient
(
	JudicialFormsClientID int IDENTITY NOT NULL,
	JudicialFormsID int NOT NULL,
	ClientID int NOT NULL,
	CONSTRAINT JudicialFormsClientPK PRIMARY KEY (JudicialFormsClientID),
	CONSTRAINT JudicialFormsClient_FK1 FOREIGN KEY (JudicialFormsID) REFERENCES
JudicialForms(JudicialFormsID),
	CONSTRAINT JudicialFormsClient_FK2 FOREIGN KEY (ClientID) REFERENCES
Client(ClientID)

);
go

CREATE TABLE DiscoveryClient
(
	DiscoveryClientID int IDENTITY NOT NULL,
	DiscoveryID int NOT NULL,
	ClientID int NOT NULL,
	CONSTRAINT DiscoveryIDU1 UNIQUE(DiscoveryID),
	CONSTRAINT ClientIDU2 UNIQUE(ClientID),
	CONSTRAINT DiscoveryClientPK PRIMARY KEY (DiscoveryClientID),
	CONSTRAINT DiscoveryClientFK1 FOREIGN KEY (DiscoveryClientID) REFERENCES
Discovery(DiscoveryID),
	CONSTRAINT DiscoveryClientFK2 FOREIGN KEY (DiscoveryClientID) REFERENCES
Client(ClientID)

);
go

CREATE TABLE ClientCourtCase
(
	ClientCourtCaseID int IDENTITY NOT NULL,
	CONSTRAINT ClientCourtCasePK PRIMARY KEY (ClientCourtCaseID),
	CourtCaseID int NOT NULL,
	ClientID int NOT NULL,
	CONSTRAINT CourtCaseIDU1 UNIQUE(CourtCaseID),
	CONSTRAINT ClientIDU3 UNIQUE(ClientID),
	CONSTRAINT ClientCourtCaseFK1 FOREIGN KEY (CourtCaseID) REFERENCES
CourtCase(CourtCaseID),
	CONSTRAINT ClientCourtCaseFK2 FOREIGN KEY (ClientID) REFERENCES Client(ClientID)

);
Go

-- All items tested correctly


CREATE TABLE FirmCalendarClientPersonnel
(

	FirmCalendarClientPersonnelID int IDENTITY NOT NULL,
	MasterMeetingNumber int NOT NULL,
	PersonnelHoursSpent int  ,
	ClientID int ,
	PersonnelID int ,
	FirmCalendarID int ,
	CONSTRAINT FirmCalendarClientPersonnelPK PRIMARY KEY
(FirmCalendarClientPersonnelID),
	CONSTRAINT FirmCalendarClientPersonnelFK1 FOREIGN KEY (ClientID) REFERENCES
Client(ClientID),
	CONSTRAINT FirmCalendarClientPersonnelFK2 FOREIGN KEY (PersonnelID) REFERENCES 
Personnel(PersonnelID),
	CONSTRAINT FirmCalendarClientPersonnelFK3 FOREIGN KEY( FirmCalendarID) REFERENCES 
FirmCalendar(FirmCalendarID)

);
go

CREATE TABLE CourtCaseHearingDate
(
	CourtCaseHearingDateID int IDENTITY NOT NULL,
	CourtCaseID int NOT NULL,
	CourtHearingDateID int NOT NULL,
	CONSTRAINT CourtCaseIDU2 UNIQUE(CourtCaseID),
	CONSTRAINT CourtHearingDateIDU2 UNIQUE(CourtHearingDateID),
	CONSTRAINT CourtCaseHearingDatePK PRIMARY KEY (CourtCaseHearingDateID),
	CONSTRAINT CourtCaseHearingDateFK1 FOREIGN KEY (CourtCaseID) REFERENCES 
CourtCase(CourtCaseID),
	CONSTRAINT CourtCaseHearingDateFK2 FOREIGN KEY (CourtHearingDateID) REFERENCES 
CourtHearingDate(CourtHearingDateID)

);
go

CREATE TABLE FirmCalendarCourtHearingDate
(
	FirmCalendarCourtHearingDateID int IDENTITY NOT NULL,
	FirmCalendarID int NOT NULL,
	CourtHearingDateID int NOT NULL,
	CONSTRAINT FirmCalendarID UNIQUE(FirmCalendarID),
	CONSTRAINT CourtHearingDateID UNIQUE(CourtHearingDateID),
	CONSTRAINT FirmCalendarCourtHearingDatePK PRIMARY KEY 
(FirmCalendarCourtHearingDateID),
	CONSTRAINT FirmCalendarCourtHearingDateFK1 FOREIGN KEY 
(FirmCalendarCourtHearingDateID) REFERENCES FirmCalendar(FirmCalendarID),
	CONSTRAINT FirmCalendarCourtHearingDateFK2 FOREIGN KEY 
(FirmCalendarCourtHearingDateID) REFERENCES CourtHearingDate(CourtHearingDateID)


);
go


/*
	This ends the body of the Law Office Database tables.
```

DATA CREATION:

Inserts, Alters, Update, Deletion of Data

```
/*
        This ends the body of the Law Office Database tables.

        Moving Forward we will see our
        INSERT statements
        PROCEDURES
        VIEWS
        AGGREGATES

*/

/*
       DATA CREATION

       In order to answer data questions; there has to be data present that can be
aggregatly ran against.
*/

--creating client data, adding 10 clients to clients table

INSERT INTO Client
(ClientFirstName, ClientLastName, ClientAddress1, ClientAddress2, ClientEmailAddress,
ClientCity, ClientState, ClientZIP)

VALUES

('David', 'Wearhouse', '84 Home Lane', 'Apartment 2', 'dwearhouse@mail.mail', 'New York',
'New State', '071000'),
('Edward', 'Vincent', '83 House Street', 'none', 'evin@email.mail', 'OldeTowne', 'Texas',
'071002'),
('Franny', 'Ulvade', '82 State Street', '3 floor', 'fuvalde@mail.org', 'Kingston',
'Mississippi', '071001'),
('Greg', 'Thomas', '24 Basic Bia Ave', '2 floor', 'gthom@funktown.org', 'Kings Failing',
'New Hampshire', '071003'),
('Hillary', 'Samson', '12 Trail Place Ln', 'none', 'Hsam@att.com', 'Wylie', 'Texas',
'071020'),
('Icabod', 'Crane', '1123 Sleepy Street Road', 'none', 'icrane@home.org', 'Sleepy
Hollow', 'New York', '071001'),
('Shelly', 'Crable', '41 Patricia Ln', 'Lot 1', 'scrable@att.org', 'Westminister',
'Colorado', '90212'),
('Gavin', 'Lee', '45 West Street', 'Lot 12', 'glee@wiznet.com', 'McKinney', 'Nevada',
'12010'),
('Chris', 'Tucker', '1121 Lefty Lane', 'Apt 123', 'tuckerc@me.org', 'Detriot',
'Michigan', '65520'),
('Shelby', 'Mustang', '66 Big Street', 'none', 'smustang@ford.com', 'Chicago',
'Illinois', '102801'),
('Randall', 'Taylor', '66 Big Street', 'none', 'rtaylor@ford.com', 'Chicago', 'Illinois',
'102801')

go


-- Adding Court(s), Location(s), Judge(s), Case Numbers to the Court Case Table
```

```
INSERT INTO CourtCase
(CourtName1, CourtName2, CourtNumber, CourtJudicialOfficer1, CourtJudicialOfficer2,
CourtCaseNumber)

VALUES

('District', 'Court Room A', '222nd', 'Judge Alex Jones', 'Associate Judge', '071000'),
('Municipal', 'Court Room B', '83rd', 'Judge Chris Tucker', 'Associate Judge','071002'),
('Federal', 'Court Room c', '4th Appeals Court', 'Judge Harris', 'Associate Judge',
'071001'),
('District', 'Court Room D', '24th', 'Judge Johnson', 'Associate Judge', '071003'),
('Tax', 'Court Room B', '12th', 'Judge Blinkon', 'Associate Judge', '071020'),
('Appelate Court', 'Court Room A', '112th', 'Judge Blankenship', 'Associate Judge',
'071001'),
('Probate Court', 'Court Room A', '41st', 'Judge Qye', 'Associate Judge', '90212'),
('District Court', 'Court Room B', '45th ', 'Judge Jacobs', 'Associate Judge', '12010'),
('Appeals Court', 'Court Room B', '11th', 'Judge Johnson', 'Associate Judge', '65520'),
('District', 'Court Room B', '222ns', 'Judge Hisname', 'Associate Judge', '102801')

go

--SELECT *
--FROM CourtCase

INSERT INTO Personnel
(PersonnelFirstName, PersonnelLastName, PersonnelBarNumber)
VALUES
('Perry','Mason','323'),
('Della','Street','232'),
('Sarah','Pause','544'),
('William','Fontaine','656')
go

--adding Personnel to Client relationships

INSERT INTO PersonnelClient (PersonnelID, ClientID)
VALUES (
       (SELECT PersonnelID FROM Personnel WHERE PersonnelLastName = 'Pause'),
       (SELECT ClientID FROM Client WHERE ClientLastName = 'Crable')
          )
go

INSERT INTO PersonnelClient (PersonnelID, ClientID)
VALUES (
              (SELECT PersonnelID FROM Personnel WHERE PersonnelLastName = 'Pause'),
              (SELECT ClientID FROM Client WHERE ClientLastName = 'Vincent')
              )
go

INSERT INTO PersonnelClient (PersonnelID, ClientID)
VALUES (
              (SELECT PersonnelID FROM Personnel WHERE PersonnelLastName = 'Street'),
              (SELECT ClientID FROM Client WHERE ClientLastName = 'Tucker')
              )
go

INSERT INTO PersonnelClient (PersonnelID, ClientID)
VALUES (
              (SELECT PersonnelID FROM Personnel WHERE PersonnelLastName = 'Street'),
              (SELECT ClientID FROM Client WHERE ClientLastName = 'Tucker')
              )
go

INSERT INTO PersonnelClient (PersonnelID, ClientID)
VALUES (
              (SELECT PersonnelID FROM Personnel WHERE PersonnelLastName = 'Mason'),
              (SELECT ClientID FROM Client WHERE ClientLastName = 'Crane')
              )
go

INSERT INTO PersonnelClient (PersonnelID, ClientID)
VALUES (
              (SELECT PersonnelID FROM Personnel WHERE PersonnelLastName = 'Fontaine'),
              (SELECT ClientID FROM Client WHERE ClientLastName = 'Tucker')
              )
go

INSERT INTO PersonnelClient (PersonnelID, ClientID)
VALUES (
              (SELECT PersonnelID FROM Personnel WHERE PersonnelLastName = 'Pause'),
              (SELECT ClientID FROM Client WHERE ClientLastName = 'Thomas')
              )
go

INSERT INTO PersonnelClient (PersonnelID, ClientID)
VALUES (
              (SELECT PersonnelID FROM Personnel WHERE PersonnelLastName = 'Pause'),
              (SELECT ClientID FROM Client WHERE ClientLastName = 'Wearhouse')
              )
go

--Adding Data to the Billing Table, Billing Descriptions and Billing Dates

-- I forgot to add the BillingDate attribute to the Billing Table



go
INSERT INTO Billing
(BillingClientFirstName,  BillingClientLastName, BillingDescription1,
BillingDescription2, BillingDate )

VALUES

('David','Wearhouse','District Court Room A','222nd Judge Alex Jones Associate Judge',
'2019-04-02'),
('Edward','Vincent','Municipal Court Room B', '83rd Judge Chris Tucker Associate
Judge','2019-04-01'),
('Franny','Ulvade','Federal  Court Room C', '4th Appeals Court Judge Harris Associate
Judge', '2019-04-08'),
('Icabod','Crane','District Court Room D', '24th Judge Johnson Associate Judge', '2019-
04-07'),
('Shelly','Crable','Tax Court Room B', '12th Judge Blinkon Associate Judge', '2019-04-
04'),
('Gavin','Lee','Appelate Court Court Room A', '112th Judge Blankenship Associate Judge',
'2019-04-20'),
```

```sql
('Hillary','Samson','Probate Court Court Room A', '41st Judge Qye Associate Judge',
'2019-04-21'),
('Greg','Thomas','District Court Court Room B', '45th  Judge Jacobs Associate Judge',
'2019-04-14'),
('Chris','Tucker','Appeals Court Court Room B', '11th Judge Johnson Associate Judge',
'2019-04-26'),
('Shelby','Mustang','District Court Room B', '222nd Judge Hisname Associate Judge',
'2019-04-30'),
('Randall', 'Taylor', 'Just a Demonstration', 'Of the Delete','2019-04-04')

go

-- The following insert statements are inserting Billing data, Client data, and hours
billed, to Billing Client

INSERT INTO BillingClient (BillingID, ClientID, ClientTotalBillingHours)
VALUES (
       (SELECT BillingID FROM Billing WHERE BillingClientLastName = 'Wearhouse'),
       (SELECT ClientID FROM Client WHERE ClientLastName = 'Wearhouse'),
          ('10')
          )
go

INSERT INTO BillingClient (BillingID, ClientID, ClientTotalBillingHours)
VALUES (
       (SELECT BillingID FROM Billing WHERE BillingClientLastName = 'Thomas'),
       (SELECT ClientID FROM Client WHERE ClientLastName = 'Thomas'),
          ('40')
          )
go

INSERT INTO BillingClient (BillingID, ClientID, ClientTotalBillingHours)
VALUES (
       (SELECT BillingID FROM Billing WHERE BillingClientLastName = 'Crable'),
       (SELECT ClientID FROM Client WHERE ClientLastName = 'Crable'),
          ('80')
          )
go

INSERT INTO BillingClient (BillingID, ClientID, ClientTotalBillingHours)
VALUES (
       (SELECT BillingID FROM Billing WHERE BillingClientLastName = 'Lee'),
       (SELECT ClientID FROM Client WHERE ClientLastName = 'Lee'),
          ('40')
          )
go

INSERT INTO BillingClient (BillingID, ClientID, ClientTotalBillingHours)
VALUES (
       (SELECT BillingID FROM Billing WHERE BillingClientLastName = 'Mustang'),
       (SELECT ClientID FROM Client WHERE ClientLastName = 'Mustang'),
          ('40')
          )
go

INSERT INTO BillingClient (BillingID, ClientID, ClientTotalBillingHours)
VALUES (
       (SELECT BillingID FROM Billing WHERE BillingClientLastName = 'Crane'),
       (SELECT ClientID FROM Client WHERE ClientLastName = 'Crane'),
          ('78')
          )
go

INSERT INTO BillingClient (BillingID, ClientID, ClientTotalBillingHours)
VALUES (
       (SELECT BillingID FROM Billing WHERE BillingClientLastName = 'Samson'),
       (SELECT ClientID FROM Client WHERE ClientLastName = 'Samson'),
          ('65')
          )
go

INSERT INTO BillingClient (BillingID, ClientID, ClientTotalBillingHours)
VALUES (
       (SELECT BillingID FROM Billing WHERE BillingClientLastName = 'Vincent'),
       (SELECT ClientID FROM Client WHERE ClientLastName = 'Vincent'),
          ('78')
          )
go

INSERT INTO BillingClient (BillingID, ClientID, ClientTotalBillingHours)
VALUES (
       (SELECT BillingID FROM Billing WHERE BillingClientLastName = 'Thomas'),
       (SELECT ClientID FROM Client WHERE ClientLastName = 'Thomas'),
          ('60')
          )
go


go

INSERT INTO BillingClient (BillingID, ClientID, ClientTotalBillingHours)
VALUES (
       (SELECT BillingID FROM Billing WHERE BillingClientLastName = 'Vincent'),
       (SELECT ClientID FROM Client WHERE ClientLastName = 'Vincent'),
          ('40')
          )
go

INSERT INTO BillingClient (BillingID, ClientID, ClientTotalBillingHours)
VALUES (
       (SELECT BillingID FROM Billing WHERE BillingClientLastName = 'Crane'),
       (SELECT ClientID FROM Client WHERE ClientLastName = 'Crane'),
          ('20')
          )
go

-- The following insert statements are to enter values within the BillingPersonnel table


INSERT INTO BillingPersonnel (BillingPersonnelRate, BillingPersonnelHoursOwed,
PersonnelID, BillingID)
VALUES (
              ('150'), ('80'),
              (SELECT PersonnelID FROM Personnel WHERE PersonnelLastName = 'Pause'),
              (SELECT ClientID FROM Client WHERE ClientLastName = 'Crable')
              )
```

```
go





/*

	DELETE STATEMENTS :
	While going throught the Client and Billing Tables; entries I noticed that I
accidentally added myself to the list

	UPDATE STATEMENTS :
	Client table, David Wearhouse has moved and the address needs to be updated.
*/

DELETE FROM Client
WHERE ClientID = 11

-- DELETE STATEMENTS: Delete record Billing ID via the Last Name

DELETE FROM Billing

WHERE BillingClientLastName = 'Taylor'


--Confirming the data changes within these tables
SELECT * FROM Client
SELECT * FROM Billing

--UPDATE STATEMENTS : Client table, David Wearhouse has moved and the address needs to be
updated.

UPDATE Client
SET ClientAddress1 = '40 Westwood lane'
WHERE ClientID = 1
go

UPDATE Client
SET ClientAddress2 = 'TownHouse 3'
WHERE CLientID = 1
go
```

… Here ends the Data Creation section …

<u>Stored Procedures, Functions, Views:</u>

```
/*
              Stored procedures for:

*/

-- Create a stored procedure for entering Case information into the CourtCase table
(sp_#1)

CREATE PROC spCreateCase
              @CourtName1 varchar(50),
              @CourtName2 varchar(50),
              @CourtNumber varchar(20),
              @CourtJudicialOfficer1 varchar(40),
              @CourtJudicialOfficer2 varchar(40),
              @CourtCaseNumber varchar(50)

AS
BEGIN
IF EXISTS
       (SELECT * FROM CourtCase WHERE CourtCaseNumber = @CourtCaseNumber)

       BEGIN
              UPDATE CourtCase
              SET CourtName1 = @CourtName1, CourtName2 = @CourtName2, CourtNumber =
@CourtNumber,
              CourtJudicialOfficer1 = @CourtJudicialOfficer1, CourtJudicialOfficer2 =
@CourtJudicialOfficer2,
              CourtCaseNumber = @CourtCaseNumber
       END

ELSE
       BEGIN
              INSERT INTO CourtCase
              (CourtName1, CourtName2, CourtNumber, CourtJudicialOfficer1,
CourtJudicialOfficer2, CourtCaseNumber)

              VALUES
              (@CourtName1, @CourtName2, @CourtNumber, @CourtJudicialOfficer1,
@CourtJudicialOfficer2, @CourtCaseNumber)
       END
       RETURN @@IDENTITY

END
go


-- Let's add some Cases into the Court Cases table via the store procedure just created
-- Executing the Stored Procedure: spCreateCase (executing stored procedure #1)


EXEC spCreateCase
              @CourtName1 = 'District',
              @CourtName2 = 'Court Room A',
              @CourtNumber = '222d ',
              @CourtJudicialOfficer1 = 'Judge Johnson ',
              @CourtJudicialOfficer2 = 'Associate Judge ',
              @CourtCaseNumber = '902132'

go

-- Create a stored procedure for entering Case information into the CourtCase table sp_#2



CREATE PROC spCreateClientCourtCase
              @CourtCaseID int,
              @ClientID int
AS
BEGIN
IF EXISTS
       (SELECT * FROM ClientCourtCase WHERE CourtCaseID = @CourtCaseID)

       BEGIN
              UPDATE ClientCourtCase
              SET CourtCaseID = @CourtCaseID, ClientID = @ClientID

       END

ELSE
       BEGIN
              INSERT INTO ClientCourtCase
              (CourtCaseID, ClientID)

              VALUES
              (@CourtCaseID, @ClientID)
       END
       RETURN @@IDENTITY

END
go

-- Let's add some ase information into the CourtCase table via the store procedure just
created
-- Executing the Stored Procedure: spCreateCase (executing stored procedure #2) creation
of 9 entries


EXEC spCreateClientCourtCase
              @CourtCaseID = '2',
              @ClientID = '2'

go

EXEC spCreateClientCourtCase
              @CourtCaseID = '3',
              @ClientID = '3'

go

EXEC spCreateClientCourtCase
              @CourtCaseID = '4',
              @ClientID = '4'
```

```
go

EXEC spCreateClientCourtCase
              @CourtCaseID = '5',
              @ClientID = '5'

go

EXEC spCreateClientCourtCase
              @CourtCaseID = '6',
              @ClientID = '6'

go

EXEC spCreateClientCourtCase
              @CourtCaseID = '7',
              @ClientID = '7'

go

EXEC spCreateClientCourtCase
              @CourtCaseID = '8',
              @ClientID = '8'

go

EXEC spCreateClientCourtCase
              @CourtCaseID = '9',
              @ClientID = '9'

go

EXEC spCreateClientCourtCase
              @CourtCaseID = '10',
              @ClientID = '10'

go

-- Create a stored procedure for creating a Firm Calendar Entry into the Firm Calendar
table sp_#3 ( FirmCalendarMeetingTime,
--FirmCalendarMeetingInfo,FirmCalendarDate)

CREATE PROC spCreateFirmCalendarEntry

              @FirmCalendarID int,
              @FirmCalendarMeetingTime varchar(100),
              @FirmCalendarMeetingInfo varchar(50),
              @FirmCalendarDate DATETIME
AS
BEGIN
IF EXISTS
       (SELECT * FROM FirmCalendar WHERE @FirmCalendarID = FirmCalendarID)

       BEGIN
              UPDATE FirmCalendar
              SET FirmCalendarMeetingTime = @FirmCalendarMeetingTime,
FirmCalendarMeetingInfo = @FirmCalendarMeetingInfo,
                     FirmCalendarDate = @FirmCalendarDate
```

```
	END

ELSE
	BEGIN
		INSERT INTO FirmCalendar
		( FirmCalendarID,FirmCalendarMeetingTime, FirmCalendarMeetingInfo,
FirmCalendarDate)

		VALUES
		(@FirmCalendarID, @FirmCalendarMeetingTime, @FirmCalendarMeetingInfo,
@FirmCalendarDate)
	END
	RETURN @@IDENTITY

END
go
-- Let's add some case information into the FirmCalendar table via the store procedure
just created
-- Executing the Stored Procedure: spCreateFirmCalendarEntry (executing stored procedure
#3) creation of 10 Calendar entries
--I know that Setting the IDentity Insert to ON is not recommended, however, I will note
in reflection that I just can not get this table to work, otherwise and I need it to work

SET IDENTITY_INSERT [dbo].FirmCalendar ON

go

EXEC spCreateFirmCalendarEntry

		@FirmCalendarDate = '2019-04-01 04:30',
		@FirmCalendarMeetingTime = 'Court Room A District Court',
		@FirmCalendarMeetingInfo = 'David Warehouse Case, Discovery Hearing',
		@FirmCalendarID = '1'

go

EXEC spCreateFirmCalendarEntry

		@FirmCalendarDate = '2019-04-02 02:30',
		@FirmCalendarMeetingTime = 'Municipal Court Court Room B Judge Tucker',
		@FirmCalendarMeetingInfo = 'Edward Vincent Case, Discovery Hearing',
			@FirmCalendarID = '2'


go

EXEC spCreateFirmCalendarEntry

		@FirmCalendarDate = '2019-04-04 02:30',
		@FirmCalendarMeetingTime = 'Federal Court Room c Judge Harris ',
		@FirmCalendarMeetingInfo = 'Franny Ulvade Case, Discovery Hearing',
		@FirmCalendarID ='3'


go

EXEC spCreateFirmCalendarEntry
```

```
              @FirmCalendarDate = '2019-04-05 02:30',
              @FirmCalendarMeetingTime = 'District Court Room D ',
              @FirmCalendarMeetingInfo = 'Greg Thomas Case, Discovery Hearing',
              @FirmCalendarID ='4'


go

EXEC spCreateFirmCalendarEntry

              @FirmCalendarDate = '2019-04-06 02:30',
              @FirmCalendarMeetingTime = 'Tax Court, Court Room B  Judge Blinkon',
              @FirmCalendarMeetingInfo = 'Hillary Samson Case, Discovery Hearing',
              @FirmCalendarID ='5'


go

EXEC spCreateFirmCalendarEntry

              @FirmCalendarDate = '2019-04-07 02:30',
              @FirmCalendarMeetingTime = 'Appeals Court Court Room B Judge Hisname',
              @FirmCalendarMeetingInfo = 'Icabod Crane Case, Discovery Hearing',
              @FirmCalendarID ='6'


go

EXEC spCreateFirmCalendarEntry

              @FirmCalendarDate = '2019-04-08 02:30',
              @FirmCalendarMeetingTime = 'Probate Court -- Miss you Mama <3',
              @FirmCalendarMeetingInfo = 'Shelly Taylor Crable Case, Probate Hearing',
              @FirmCalendarID ='7'


go

EXEC spCreateFirmCalendarEntry

              @FirmCalendarDate = '2019-04-09 02:30',
              @FirmCalendarMeetingTime = 'District Court Court Room B Judge Qye',
              @FirmCalendarMeetingInfo = 'Gavin Lee Case, Discovery Hearing',
              @FirmCalendarID ='8'


go

EXEC spCreateFirmCalendarEntry

              @FirmCalendarDate = '2019-04-10 02:30',
              @FirmCalendarMeetingTime = 'Appeals Court Court Room B Judge Johnson',
              @FirmCalendarMeetingInfo = 'Chris Tucker Case, Discovery Hearing',
              @FirmCalendarID ='9'


go
```

```
EXEC spCreateFirmCalendarEntry

              @FirmCalendarDate = '2019-04-11 02:30',
              @FirmCalendarMeetingTime = 'District Court Room B Judge Hisname',
              @FirmCalendarMeetingInfo = 'Shelby Mustang Case, Discovery Hearing',
              @FirmCalendarID ='10'


go

SET IDENTITY_INSERT [dbo].FirmCalendar OFF

go
SELECT * FROM FirmCalendar


--Right this is getting interesting, now, let us tie the Firm Calendar entries, to the
Personnel and Clients, via the
-- FirmCalendarClientPersonnel table (thats a mouthful, remind me to reflect upon this)
sp_#4
go

CREATE PROC spCreateFirmMasterCalendar

              @MasterMeetingNumber int,
              @PersonnelHoursSpent int,
              @ClientID int,
              @PersonnelID int,
              @FirmCalendarID int


AS
BEGIN


IF EXISTS
       (SELECT * FROM FirmCalendarClientPersonnel WHERE @MasterMeetingNumber =
MasterMeetingNumber )

       BEGIN
              UPDATE FirmCalendarClientPersonnel
              SET MasterMeetingNumber = @MasterMeetingNumber,  PersonnelHoursSpent =
@PersonnelHoursSpent, ClientID = @ClientID, PersonnelID = @PersonnelID, FirmCalendarID =
@FirmCalendarID


       END

ELSE
       BEGIN

              INSERT INTO FirmCalendarClientPersonnel
              ( MasterMeetingNumber, PersonnelHoursSpent, ClientID, PersonnelID,
FirmCalendarID)

              VALUES
```

```
                     (@MasterMeetingNumber, @PersonnelHoursSpent, @ClientID,@PersonnelID,
@FirmCalendarID)


       END
       RETURN @@IDENTITY

END

go


---- Let's add some case information into the FirmCalendarClientPersonnel table via the
stored procedure just created
-- Executing the Stored Procedure: spCreateFirmMasterCalendar (executing stored procedure
#4) creation of 10 Calendar entries

go
EXEC spCreateFirmMasterCalendar

              @FirmCalendarID = '7',
              @MasterMeetingNumber = '101',
              @PersonnelHoursSpent = '11',
              @ClientID = '7',
              @PersonnelID = '3'

go

EXEC spCreateFirmMasterCalendar
              @MasterMeetingNumber = '102',
              @PersonnelHoursSpent = '23',
              @ClientID = '2',
              @PersonnelID = '3',
              @FirmCalendarID = '2'

go


EXEC spCreateFirmMasterCalendar
              @MasterMeetingNumber = '103',
              @PersonnelHoursSpent = '55',
              @ClientID = '9',
              @PersonnelID = '2',
              @FirmCalendarID = '9'

go

EXEC spCreateFirmMasterCalendar
              @MasterMeetingNumber = '106',
              @PersonnelHoursSpent = '21',
              @ClientID = '9',
              @PersonnelID = '4',
              @FirmCalendarID = '2'

go

EXEC spCreateFirmMasterCalendar
```

```
              @MasterMeetingNumber = '107',
              @PersonnelHoursSpent = '42',
              @ClientID = '4',
              @PersonnelID = '3',
              @FirmCalendarID = '4'

go

EXEC spCreateFirmMasterCalendar
              @MasterMeetingNumber = '108',
              @PersonnelHoursSpent = '53',
              @ClientID = '1',
              @PersonnelID = '3',
              @FirmCalendarID = '1'

go

-- Views (5 pertaint to Data Questions)
-- Create a View of the Firm Calendar and Master Calendar Assignments, such, that
Personnel know their exact Scheduling  #1

CREATE VIEW viewFirmCalendar AS
              SELECT FirmCalendarClientPersonnel.MasterMeetingNumber,
                        FirmCalendarClientPersonnel.PersonnelHoursSpent,
                        FirmCalendarClientPersonnel.PersonnelID,
                        FirmCalendarClientPersonnel.ClientID,
                        FirmCalendar.FirmCalendarID,
                        FirmCalendar.FirmCalendarDate, (SUM(ALL
FirmCalendarClientPersonnel.PersonnelHoursSpent / 8) * 240) AS MediationBillRate


              FROM FirmCalendarClientPersonnel
              RIGHT OUTER JOIN FirmCalendar
              ON FirmCalendarClientPersonnel.FirmCalendarID = FirmCalendar.FirmCalendarID
              GROUP BY FirmCalendarClientPersonnel.MasterMeetingNumber,
FirmCalendarClientPersonnel.PersonnelHoursSpent,
                        FirmCalendarClientPersonnel.PersonnelID,
FirmCalendarClientPersonnel.ClientID, FirmCalendar.FirmCalendarID,
FirmCalendar.FirmCalendarDate
go

SELECT * FROM viewFirmCalendar



go


--We need to create a view that will tell us our Total Billing Hours, per each Client  #2





go
CREATE VIEW TotalBillingHoursByClient

              AS
              SELECT DISTINCT  Client.ClientID, Client.ClientLastName,
Client.ClientFirstName,  (SUM(BillingClient.ClientTotalBillingHours)) AS TotalAllBilling
              FROM BillingClient
              RIGHT OUTER JOIN Client
              ON BillingClient.ClientID = Client.ClientID
              GROUP BY BillingClient.ClientTotalBillingHours, Client.ClientID,
Client.ClientLastName, Client.ClientFirstName


GO

SELECT * FROM TotalBillingHoursByClient







--Create a Function to select the Top 5 Personnel Client Assignments that is then tied
into a view, for the purpose of answering data question,
-- top 5 personnel assignments #3
go

CREATE FUNCTION PersonnelClientCount(@Collective int)
RETURNS int AS
BEGIN
       DECLARE @returnValue int

       SELECT @returnValue = (MAX(PersonnelID)) FROM PersonnelClient
       WHERE PersonnelClient.PersonnelClientID = @Collective
       RETURN @returnValue
END
go


CREATE VIEW mostPersonnelAssignments
              AS
              SELECT TOP 5
                     *
                     , dbo.PersonnelClientCount(PersonnelClientID) AS
PersonnelAssignments
              FROM PersonnelClient
              ORDER BY PersonnelAssignments

GO

SELECT * FROM mostPersonnelAssignments






---Created the vIEW to give us the payroll total, now we can add that to a VIEW #4
GO
CREATE VIEW totalPayrollOwed

              AS
              SELECT BillingPersonnelRate, BillingPersonnelHoursOwed,
(SUM([BillingPersonnelRate]*[BillingPersonnelHoursOwed])) AS PayrollDue
              FROM BillingPersonnel
              GROUP BY [BillingPersonnelID],[BillingPersonnelRate],
[BillingPersonnelHoursOwed]

GO

SELECT * FROM totalPayrollOwed




-- Create a view and a function that will create a view of current court cases, and the
amount of cases per client

go

CREATE FUNCTION CurrentCaseCount(@ClientCount int)
RETURNS int AS
BEGIN
        DECLARE @CurrentBalance int


       SELECT @CurrentBalance = COUNT(*) FROM ClientCourtCase
       RIGHT OUTER JOIN Client
              ON ClientCourtCase.ClientID = Client.ClientID
              GROUP BY Client.ClientID, Client.ClientLastName, Client.ClientFirstName
              RETURN @CurrentBalance
END

go


CREATE VIEW ClientCourtCaseAssignments

              AS
              SELECT *
                        , dbo.CurrentCaseCount(ClientCourtCaseID) AS ClientStats

       FROM ClientCourtCase

go

SELECT * FROM ClientCourtCaseAssignments

go
```

… here ends the DDL code of the work

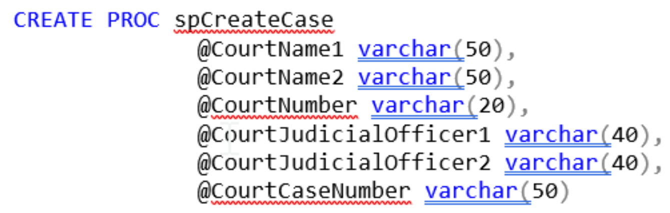
IMPLEMENTATION:

As required, per the milestone, the following section reflects examples of the implementation of Law Office Database.

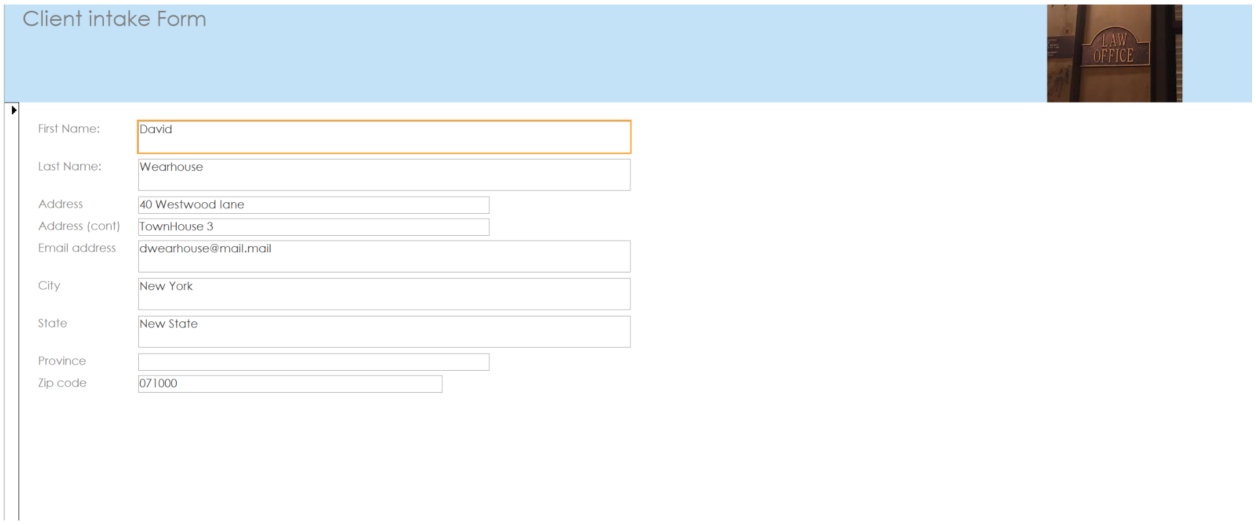
Per the call of the question, there will follow two examples of data entry forms; the reports have been previously demonstrated in the Data Question(s) section answer, but the researcher will include examples within this section, for completeness.

FORMS

Client intake form: based upon the Client Table, simple intake forms similar to the similar to the spCreateCase (referenced below) but within the Forms creation feature of Access:

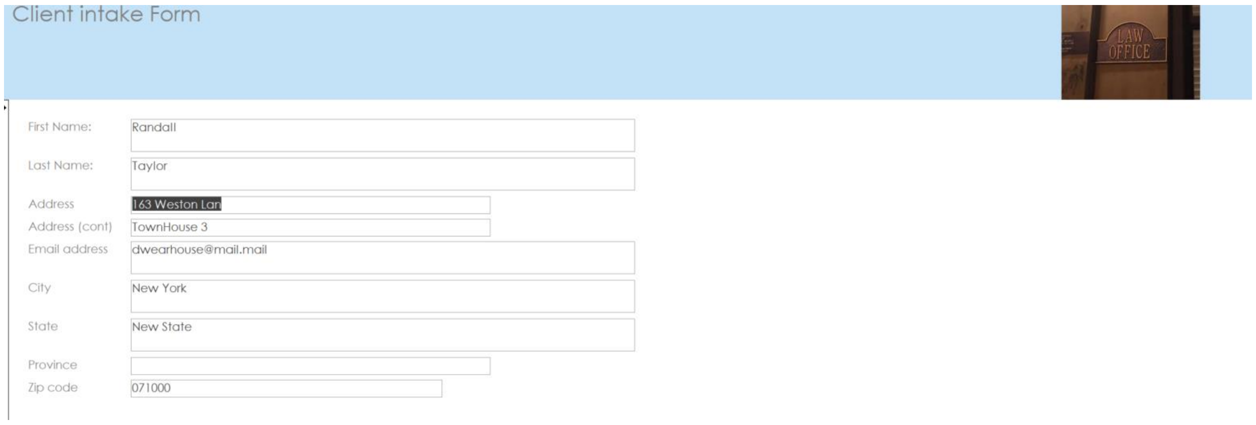


```
CREATE PROC spCreateCase
             @CourtName1 varchar(50),
             @CourtName2 varchar(50),
             @CourtNumber varchar(20),
             @CourtJudicialOfficer1 varchar(40),
             @CourtJudicialOfficer2 varchar(40),
             @CourtCaseNumber varchar(50)
```



Client intake Form

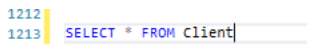
First Name:	David
Last Name:	Wearhouse
Address	40 Westwood lane
Address (cont)	TownHouse 3
Email address	dwearhouse@mail.mail
City	New York
State	New State
Province	
Zip code	071000



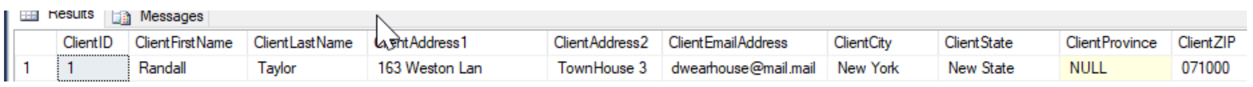
Client intake Form

First Name:	Randall
Last Name:	Taylor
Address	163 Weston Lan
Address (cont)	TownHouse 3
Email address	dwearhouse@mail.mail
City	New York
State	New State
Province	
Zip code	071000

Form functions correctly, Randall Taylor has been added to the client's tables, in SQL:



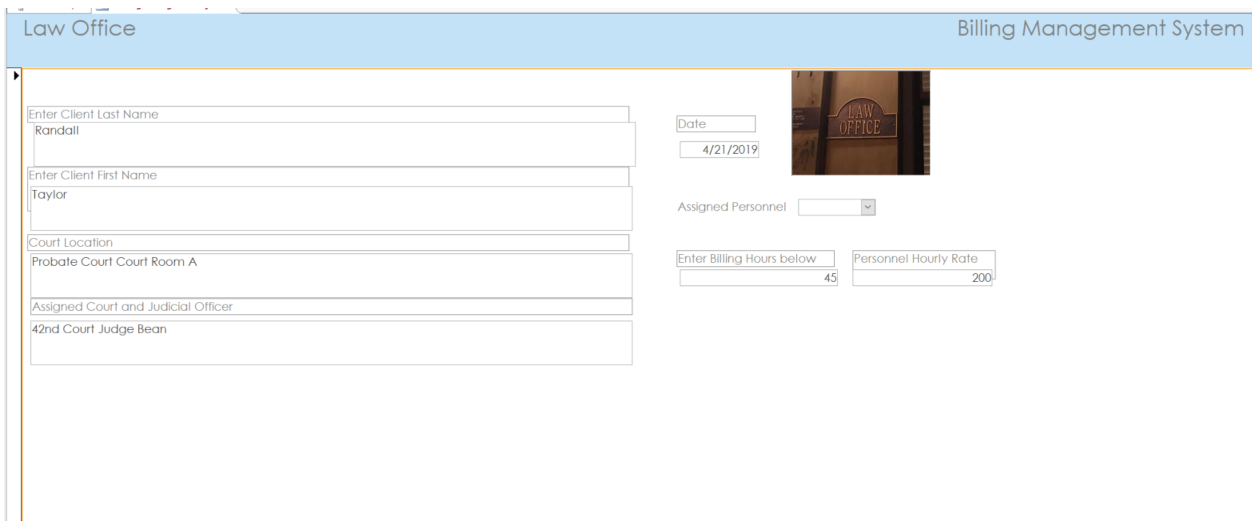
```
SELECT * FROM Client
```



Results Messages

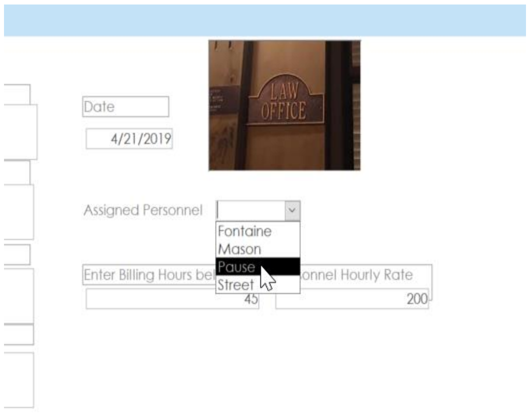
	ClientID	ClientFirstName	ClientLastName	ClientAddress1	ClientAddress2	ClientEmailAddress	ClientCity	ClientState	ClientProvince	ClientZIP
1	1	Randall	Taylor	163 Weston Lan	TownHouse 3	dwearhouse@mail.mail	New York	New State	NULL	071000

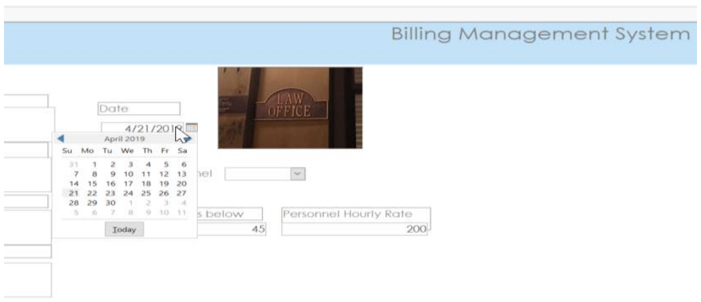
Billing Management System: based up the BillingPersonnel Table, BillingClientTable;
<u>Form: Billing Management System</u>



Law Office Billing Management System

Enter Client Last Name	Randall
Enter Client First Name	Taylor
Court Location	Probate Court Court Room A
Assigned Court and Judicial Officer	42nd Court Judge Bean
Date	4/21/2019
Assigned Personnel	
Enter Billing Hours below	45
Personnel Hourly Rate	200



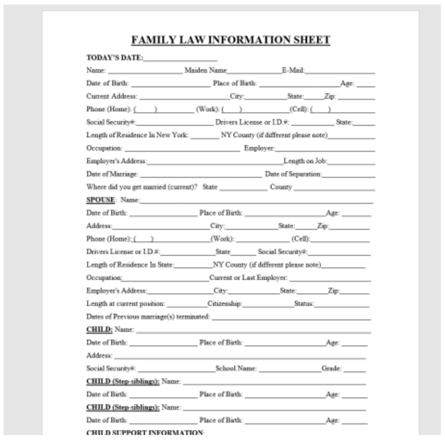


Conclusion

The purpose of this Project is to convince stakeholders, in non-technical and technical demonstrations, that there is a need to create a centralized Database that will resolve and centralize the following: Clients, Personnel (Paralegal and Attorney), Court records. Tracking of billing matters, establishment of a firm calendar has been established. The entry of discovery and the tracking of Judicial Forms can now be attached to the Client records.

APPENDEX

Raw Data Example 1:



FAMILY LAW INFORMATION SHEET

TODAY'S DATE:____________

Name: __________ Maiden Name__________ E-Mail:__________

Date of Birth: __________ Place of Birth: __________ Age: ____

Current Address: __________ City:______ State:____ Zip: ______

Phone (Home): (____) ______ (Work): (____) ______ (Cell): (____) ______

Social Security#:__________ Drivers License or I.D.#: ______ State:____

Length of Residence In New York: ______ NY County (if different please note)______

Occupation: __________ Employer:__________

Employer's Address:__________ Length on Job:______

Date of Marriage: __________ Date of Separation:______

Where did you get married (current)? State ______ County ______

SPOUSE: Name:__________

Date of Birth: __________ Place of Birth: __________ Age: ____

Address:__________ City:______ State:____ Zip:______

Phone (Home): (____) ______ (Work): ______ (Cell):______

Drivers License or I.D.#:______ State____ Social Security#:______

Length of Residence In State:______ NY County (if different please note)______

Occupation:__________ Current or Last Employer: __________

Employer's Address:__________ City:______ State:____ Zip:______

Length at current position: ______ Citizenship:______ Status:______

Dates of Previous marriage(s) terminated: __________

CHILD: Name: __________

Date of Birth: __________ Place of Birth: __________ Age: ____

Address: __________

Social Security#: __________ School Name: __________ Grade: ____

CHILD (Step-siblings): Name: __________

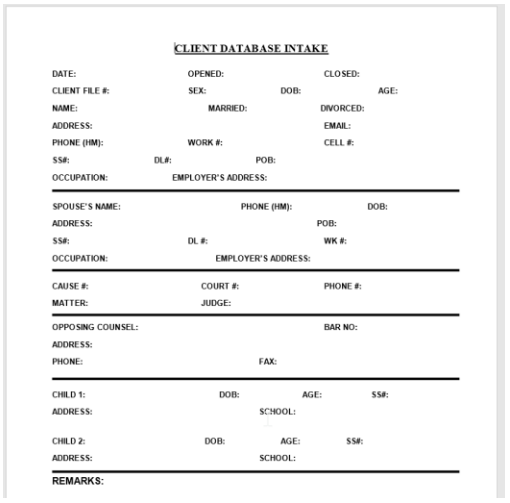
Date of Birth: __________ Place of Birth: __________ Age: ____

CHILD (Step-siblings): Name: __________

Date of Birth: __________ Place of Birth: __________ Age: ____

CHILD SUPPORT INFORMATION:

Raw Data Example 2:



CLIENT DATABASE INTAKE

DATE: OPENED: CLOSED:
CLIENT FILE #: SEX: DOB: AGE:
NAME: MARRIED: DIVORCED:
ADDRESS: EMAIL:
PHONE (HM): WORK #: CELL #:
SS#: DL#: POB:
OCCUPATION: EMPLOYER'S ADDRESS:

SPOUSE'S NAME: PHONE (HM): DOB:
ADDRESS: POB:
SS#: DL #: WK #:
OCCUPATION: EMPLOYER'S ADDRESS:

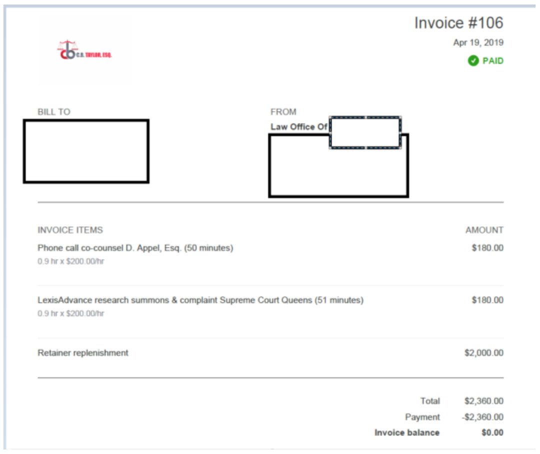
CAUSE #: COURT #: PHONE #:
MATTER: JUDGE:

OPPOSING COUNSEL: BAR NO:
ADDRESS:
PHONE: FAX:

CHILD 1: DOB: AGE: SS#:
ADDRESS: SCHOOL:

CHILD 2: DOB: AGE: SS#:
ADDRESS: SCHOOL:

REMARKS:



C.D. TAYLOR, ESQ.

Invoice #106
Apr 19, 2019
✓ PAID

BILL TO
[redacted]

FROM
Law Office Of [redacted]
[redacted]

INVOICE ITEMS	AMOUNT
Phone call co-counsel D. Appel, Esq. (50 minutes) 0.9 hr x $200.00/hr	$180.00
LexisAdvance research summons & complaint Supreme Court Queens (51 minutes) 0.9 hr x $200.00/hr	$180.00
Retainer replenishment	$2,000.00

Total	$2,360.00
Payment	-$2,360.00
Invoice balance	**$0.00**

Raw Data Example 3: