IST-659 M402 FINAL PROJECT

Database Management at Community Tutors, LLC.
Darrell Nelson II

DECEMBER 8, 2018

COMMUNITY TUTORS, LLC.
0987 Make Believe St; Welch, CA 00982 Suite #002

Database Management at Community Tutors, LLC.

Background

Community Tutors, LLC. is a small business founded by Darrell Nelson II to give children in the local community access to high level tutoring at reasonable prices. Community Tutors (CT) offers discounted pricing by leveraging undergraduate and graduate students to work part-time instead of hiring professional tutors. The company is experiencing a wave of prosperity that has led to a succession of hires and an even larger tidal wave of customers. To meet the demands of an up and coming business we have partnered with Gregory Block, Inc. where Prof. Block is the residing CEO. He has graciously allowed us to use his facilities as our temporary HQ for a percent of our profits. This memo acts as an update to Prof. Block on how we intend to utilize his accommodations more efficiently and increase revenue.

Summary

Community Tutors is experiencing an organization problem. With all the new tutors and students, it is becoming difficult to track which students met with which tutor and for how long. Originally this data was organized using an Excel spreadsheet to track the important customer/client interactions. However, as the number of clients grew the manager allowed employees to input data into the system. A move meant to improve efficiency and correctness quickly backfired as the spreadsheet became saturated with partial data entries, slang terms, short-hand, and duplicate data (Figure 1). This has led to a decrease in efficiency and poor customer satisfaction. Tutors are missing sessions and failing to input new students into the database properly. Leading to many clients leaving CT for other tutoring institutions. To prevent further damage to CT's reputation and bottom line, a new organizational and data hierarchy protocol is being installed. A new business rule, effective immediately, will only allow a designated database manager to edit the database. This will ensure that the system is cohesive, readable, and abides by company standards. The implementation of this new system should significantly lower the dropout rate of our clients who leave due to tutors missing sessions or mis-scheduling.

The following business rules will strictly be adhered to and will be the basis of the organizational tool:

- All tutors must register and qualify to teach at least one subject
- A client = a potential customer who has registered to be taught in at least one subject
- ALL tutoring sessions will be on a 1:1 basis
    - There will be NO tutors "sharing" the same room
- Each room will correspond to a different subject (e.g. Math Room = Room 1A; Science Room = Room 2A; etc.)
- A session may not be scheduled unless there is a tutor in that subject available
- All tutors will be paid based on their hourly rate with their client
- A tutor can have more than one hourly rate due to subject matter or mastery

There will be two new roles created in the company: 1) Database Manager and 2) Front-End Developer. It will be the database manager's job to build a database using SQL server. The front-end developer is responsible for designing the user interface as well as imputting new data into the system. This will reduce network exposure and create an "information" chain of command. Access has been decided as the front-end application to be used to update and build out the SQL database. Access' ability to create reports will be extremely beneficial in allowing invested parties to view CT's data and make strategic business decisions.

In order to implement the aforementioned changes: our regular database needs to broken down into its basic components and relationships. Then, built up again utilizing these relationships as the blueprint for information flow. Once, the relationships have been accurately implemented and tested the front-end service can be installed. This new organizational method will allow this company to grow to new heights and focus on the things that really matter: Tutoring!

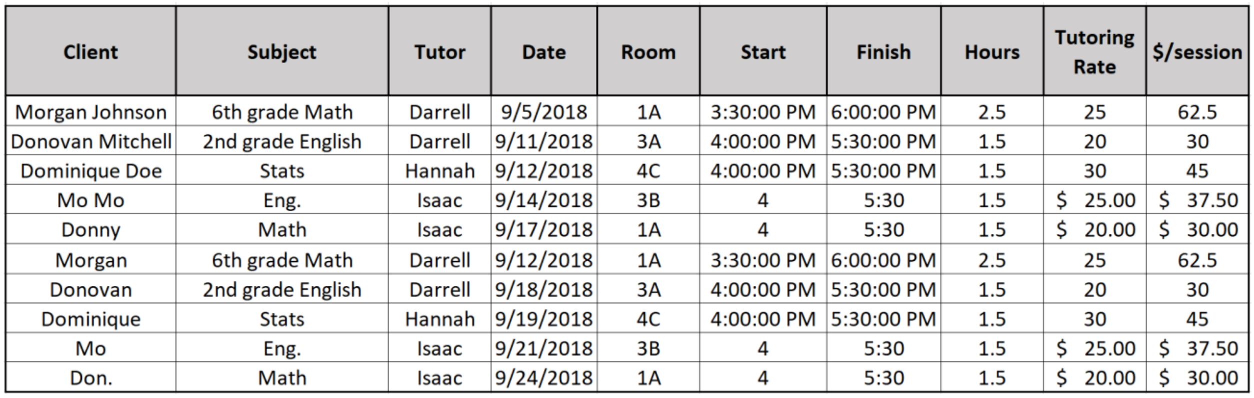
Data Questions

The database should be able to provide key insights into our business process that allow us to improve efficiency in multiple areas (e.g. training, marketing, and acceptable hourly rates). Below are a few high-value data questions that will be answered with the new database:

1. Which tutoring subject is the most popular?
    - **Identifies highest demand**
2. Which tutor is the most/least popular?
    - **Identifies teaching styles that are the most/least effective with our clients**
3. What day of the week has the highest demand?
    - **Identifies potential bottlenecks in scheduling**
4. What are the average hours of tutoring per client?
    - **Allows us to forecast resource requirements based on client growth**
5. What is the average tutoring rate ($/hour charged to clients)?
    - **Identifies what new & unexperienced tutors should charge**

Conceptual Model

The designated database manager will be Darrell Nelson II. Using the old database (Figure 1) as a reference he will create and implement the new database.



Client	Subject	Tutor	Date	Room	Start	Finish	Hours	Tutoring Rate	$/session
Morgan Johnson	6th grade Math	Darrell	9/5/2018	1A	3:30:00 PM	6:00:00 PM	2.5	25	62.5
Donovan Mitchell	2nd grade English	Darrell	9/11/2018	3A	4:00:00 PM	5:30:00 PM	1.5	20	30
Dominique Doe	Stats	Hannah	9/12/2018	4C	4:00:00 PM	5:30:00 PM	1.5	30	45
Mo Mo	Eng.	Isaac	9/14/2018	3B	4	5:30	1.5	$ 25.00	$ 37.50
Donny	Math	Isaac	9/17/2018	1A	4	5:30	1.5	$ 20.00	$ 30.00
Morgan	6th grade Math	Darrell	9/12/2018	1A	3:30:00 PM	6:00:00 PM	2.5	25	62.5
Donovan	2nd grade English	Darrell	9/18/2018	3A	4:00:00 PM	5:30:00 PM	1.5	20	30
Dominique	Stats	Hannah	9/19/2018	4C	4:00:00 PM	5:30:00 PM	1.5	30	45
Mo	Eng.	Isaac	9/21/2018	3B	4	5:30	1.5	$ 25.00	$ 37.50
Don.	Math	Isaac	9/24/2018	1A	4	5:30	1.5	$ 20.00	$ 30.00

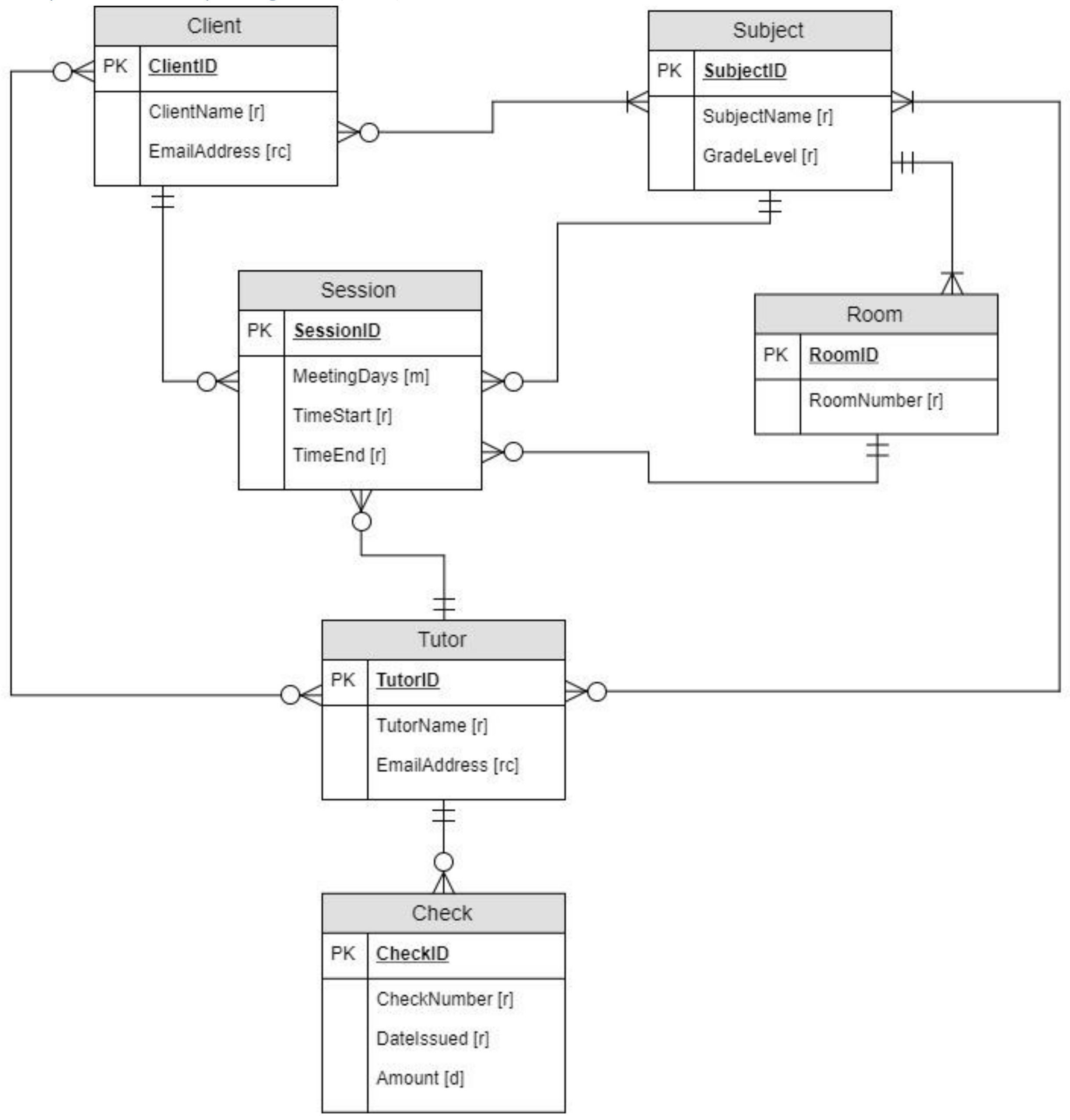
Figure 1. Old Database

After studying the business rules outlined in the Summary section coupled with the current data; the **entities**, **attributes**, and **cardinality** have been assigned as follows:

Relationships

1. Each client can be tutored in one or many subjects
2. Each client can have multiple sessions
3. Each session will only have one client. A session can't exist without a client
4. Each client may have multiple tutors
5. Each subject can be taught in many sessions
6. Each subject can be taught by many tutors
    a. A tutor must teach at least one or more subjects
7. Each subject can be taught in one or many rooms
    a. Each room will only have 1 subject taught in it
8. Each session must have a tutor
9. Each session must have a room
10. Each tutor gets one check from every client he/she has

Entity-Relationship Diagram (ERD)



***All entities and attributes are broken out in further detail in Appendix Table 1-A**

Normalized Logical Model

In order to eliminate relational dependencies, improve consistency, and reduce data redundancy/maintenance the data needs to be **normalized**. Starting with the table 'Client' we need to break out the attribute 'ClientName' into more fundamental parts (e.g. First Name, Last Name, etc.) this will allow greater information storage and make it easier to identity clients. It also eliminates the chance of adding nicknames. We also assign a "**unique key**" to the email address attribute. The unique key forces the database to only accept a unique combination of characters into that field when accepting a new entry. Therefore, a new database rule is born: “Every client must have their own unique email address”. When building a logical model, items in bold are required fields that cannot be left blank when inputting a new entry. So, if we make it required that every client must have a first name, last name, and email address those attributes will be placed in bold font in the logical model table for ‘Client’; they will also be assigned the datatype of varchar due to the varying lengths of people’s names and email addresses. The middle initial is not required because not everyone has a middle name and it is assigned the datatype char because we want the database to only accept the first letter of the first middle name even if they have multiple middle names.

All **primary keys** (PKs) in this database will use the surrogate keys provided by the server int identity property.

All arrows will be drawn from the PK of one table to the **foreign key** (FK) of another table.

Add a FK to the many side of each one to many relationship.

Since the ‘Session’ table contains a multivariable attribute and we don’t have a way to store multiple values for one attribute in a table, we need to create a new table for those. In most cases, we can combine the names of the original entity and the attribute to build a good name for this new table. Once the table is given its surrogate key from the database’s int identity, the attributes within this table are the PKs from both tables. Those PKs are now FKs in this new bridge table. The attributes ‘TimeStart’ and ‘TimeEnd’ are given the datatype time.

Because there is no functional way to implement a many-to-many relationship in a real database management system, we must use a bridge table, or technically speaking, a weak-associative table, between the participating tables in the many to many. The bridge tables contain the PK of the tables they are relating to (along with any other important information) which act as FKs. This database contains three bridge tables: ‘ClientSubject’, ‘ClientTutor’, and ‘TutorSubject’. All contain just the PKs of their associated tables as their FKs, except ‘TutorSubject’ additionally has a required attribute called ‘TutorRate’. This value requires tutors to tell clients how much they charge upfront based upon the subject and grade matter which diminishes the possibility of tutors unjustly charging one client more than the next for the same type of work.

The ‘Subject’ table has two required attributes with the varchar data type: ‘SubjectName’ and ‘GradeLevel’. This requires that all subjects input into the system will provide the tutors with the subject name and grade level so there is no confusion to the tutoring topic.

The ‘Room’ table has one unique attribute ‘RoomNumber’ that is given the data type varchar sine room numbers could possibly change in character length (e.g. 3rd floor room 1 could be 3A or 3-A or 301) in the future.

The ‘Tutor’ table is treated the same way as the ‘Client’ table.

The ‘Check’ table requires that the check number, date issued, and dollar amount be captured before adding a check to the database. This ensures that we have enough of the check data in case we ever need to go back due to a dispute.

Relational Diagram

Client (<u>ClientID</u>, FirstName, MiddleInitial, LastName, EmailAddress)

Session (<u>SessionID</u>, Date, TimeStart, TimeEnd, *RoomID, ClientID, SubjectID, TutorID, SessionMeetingDayID*)

SessionMeetingDay (<u>SessionMeetingDayID,</u> DayOfWeek)

Subject (<u>SubjectID</u>, SubjectName, GradeLevel)

ClientSubject (<u>ClientSubjectID</u>, *ClientID, SubjectID*)

Room (<u>RoomID</u>, RoomNumber, *SubjectID*)

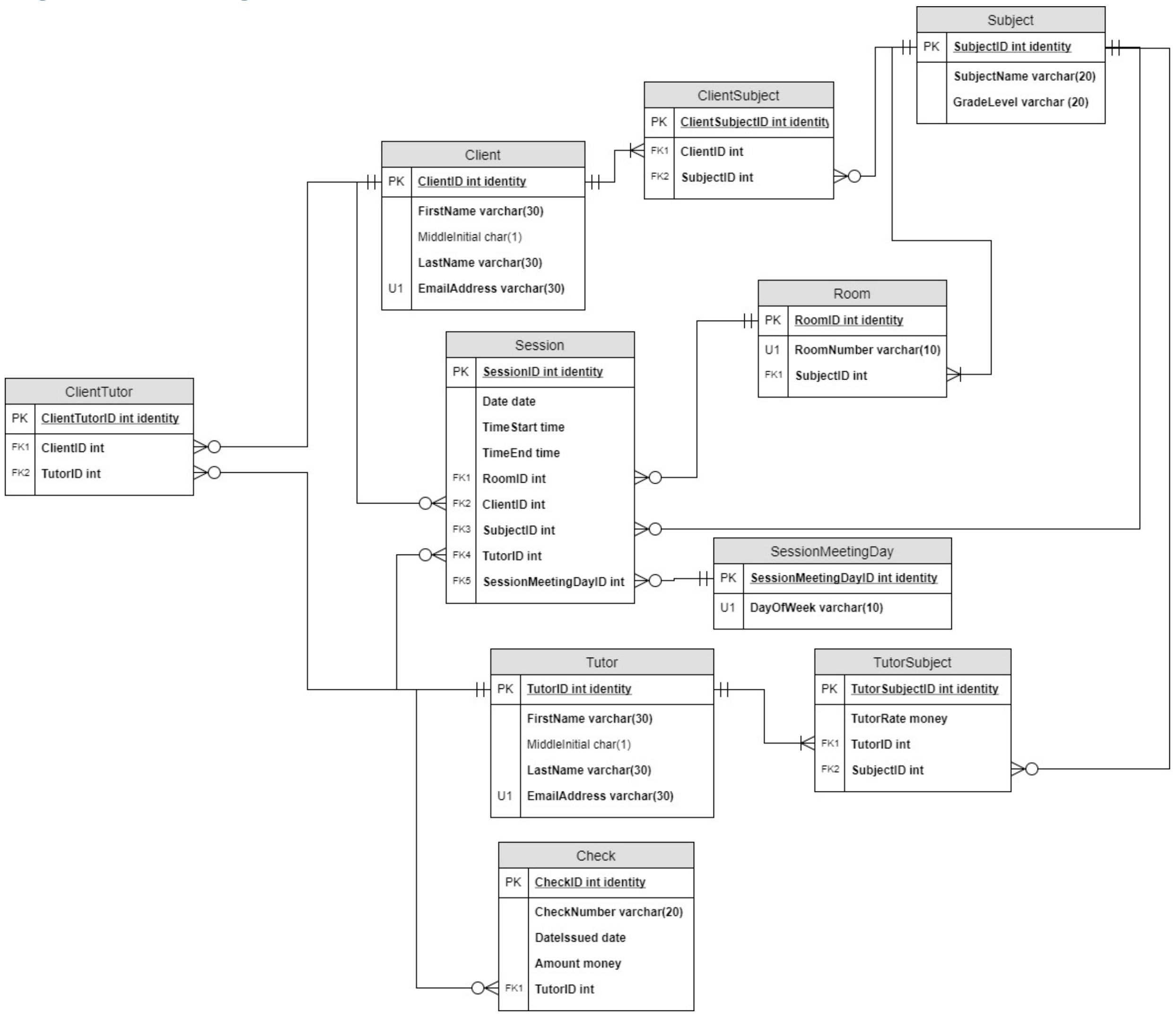
Tutor (<u>TutorID</u>, FirstName, MiddleInitial, LastName, EmailAddress)

ClientTutor (<u>ClientTutorID</u>, *ClientID, TutorID*)

TutorSubject (<u>TutorSubjectID</u>, TutorRate, *TutorID, SubjectID*)

Check (<u>CheckID</u>, CheckNumber, DateIssued, Amount, *TutorID*)

Logical Model Diagram



Part 2

Physical Database Design

SQL Code:

```
*
	Section: Physical Database Design

	Action: Building tables listed in Logical Model Diagram. Must drop, then build
tables in a sequential order so no errors arise when we rerun this code in order to
repopulate the tables.
	Order is as follows:
	Drop --> Check, Session, TutorSubject, SessionMeetingDay, Room, ClientTutor,
ClientSubject, Tutor, Subject, & Client
	CREATE --> Client, Subject, Tutor, ClientSubject, ClientTutor, Room,
SessionMeetingDay, TutorSubject, Session, & Check
*/

--Dropping Tables in Order Specified in the Physical Database Design comment header
IF OBJECT_ID('[Check]') IS NOT NULL
DROP TABLE [Check];
GO

IF OBJECT_ID('[Session]') IS NOT NULL
DROP TABLE [Session];
GO

IF OBJECT_ID('TutorSubject') IS NOT NULL
DROP TABLE TutorSubject;
GO

IF OBJECT_ID('SessionMeetingDay') IS NOT NULL
DROP TABLE SessionMeetingDay;
GO

IF OBJECT_ID('Room') IS NOT NULL
DROP TABLE Room;
GO

IF OBJECT_ID('ClientTutor') IS NOT NULL
DROP TABLE ClientTutor;
GO

IF OBJECT_ID('ClientSubject') IS NOT NULL
DROP TABLE ClientSubject;
GO

IF OBJECT_ID('Tutor') IS NOT NULL
DROP TABLE Tutor;
GO

IF OBJECT_ID('[Subject]') IS NOT NULL
DROP TABLE [Subject];
GO
```

```
IF OBJECT_ID('Client') IS NOT NULL
DROP TABLE Client;
GO

-- Creating tables in the specified Order given in the Physical Database Design comment
header
	-- Create Client Table
	CREATE TABLE Client (
		-- Columns for the Client table
		ClientID int identity,
		FirstName varchar(30) not null,
		MiddleInitial char(1),
		LastName varchar(30) not null,
		EmailAddress varchar(30) not null,
		-- Constraints on the Client Table
		CONSTRAINT PK_Client PRIMARY KEY (ClientID),
		CONSTRAINT U1_Client UNIQUE(EmailAddress)
)
--End Creating the Client table

	-- Create Subject Table
	CREATE TABLE [Subject] (
		-- Columns for the Subject table
		SubjectID int identity,
		SubjectName varchar(20) not null,
		GradeLevel varchar(20) not null,
		-- Constraints on the Subject Table
		CONSTRAINT PK_Subject PRIMARY KEY (SubjectID)
)
--End Creating the Subject table

	-- Create Tutor Table
	CREATE TABLE Tutor (
		-- Columns for the Tutor table
		TutorID int identity,
		FirstName varchar(30) not null,
		MiddleInitial char(1),
		LastName varchar(30) not null,
		EmailAddress varchar(30) not null,
		-- Constraints on the Tutor Table
		CONSTRAINT PK_Tutor PRIMARY KEY (TutorID),
		CONSTRAINT U1_Tutor UNIQUE(EmailAddress)
)
--End Creating the Tutor table

	-- Creating the ClientSubject table
	CREATE TABLE ClientSubject (
		-- Columns for the ClientSubject table
		ClientSubjectID int identity,
		ClientID int not null,
		SubjectID int not null,
		-- Constraints on the ClientSubject table
		CONSTRAINT PK_ClientSubject PRIMARY KEY (ClientSubjectID),
		CONSTRAINT FK1_ClientSubject FOREIGN KEY (ClientID) REFERENCES
Client(ClientID),
		CONSTRAINT FK2_ClientSubject FOREIGN KEY (SubjectID) REFERENCES
[Subject](SubjectID)
)
```

```
-- End creating the ClientSubject table

	-- Creating the ClientTutor table
	CREATE TABLE ClientTutor (
		-- Columns for the ClientTutor table
		ClientTutorID int identity,
		ClientID int not null,
		TutorID int not null,
		-- Constraints on the ClientTutor table
		CONSTRAINT PK_ClientTutor PRIMARY KEY (ClientTutorID),
		CONSTRAINT FK1_ClientTutor FOREIGN KEY (ClientID) REFERENCES
Client(ClientID),
		CONSTRAINT FK2_ClientTutor FOREIGN KEY (TutorID) REFERENCES Tutor(TutorID)
)
-- End creating the ClientTutor table

	-- Creating the Room table
	CREATE TABLE Room (
		-- Columns for the Room table
		RoomID int identity,
		RoomNumber varchar(10) not null,
		SubjectID int not null,
		-- Constraints on the Room table
		CONSTRAINT PK_Room PRIMARY KEY (RoomID),
		CONSTRAINT U1_Room UNIQUE (RoomNumber),
		CONSTRAINT FK1_Room FOREIGN KEY (SubjectID) REFERENCES
[Subject](SubjectID),
)
-- End creating the Room table

	-- Creating the SessionMeetingDay table
	CREATE TABLE SessionMeetingDay (
		-- Columns for the SessionMeetingDay table
		SessionMeetingDayID int identity,
		[DayOfWeek] varchar(10) not null,
		-- Constraints on the SessionMeetingDay table
		CONSTRAINT PK_SessionMeetingDay PRIMARY KEY (SessionMeetingDayID),
		CONSTRAINT U1_SessionMeetingDay UNIQUE ([DayOfWeek]),
)
-- End creating the SessionMeetingDay table

	-- Creating the TutorSubject table
	CREATE TABLE TutorSubject (
		-- Columns for the TutorSubject table
		TutorSubjectID int identity,
		TutorRate money not null,
		TutorID int not null,
		SubjectID int not null,
		-- Constraints on the TutorSubject table
		CONSTRAINT PK_TutorSubject PRIMARY KEY (TutorSubjectID),
		CONSTRAINT FK1_TutorSubject FOREIGN KEY (TutorID) REFERENCES
Tutor(TutorID),
		CONSTRAINT FK2_TutorSubject FOREIGN KEY (SubjectID) REFERENCES
[Subject](SubjectID)
)
-- End creating the TutorSubject table

	-- Creating the Session table
	CREATE TABLE [Session] (
		-- Columns for the Session table
		SessionID int identity,
		[Date] date not null,
		TimeStart time not null,
		TimeEnd time not null,
		RoomID int not null,
		ClientID int not null,
		SubjectID int not null,
		TutorID int not null,
		SessionMeetingDayID int not null,
		-- Constraints on the Session table
		CONSTRAINT PK_Session PRIMARY KEY (SessionID),
		CONSTRAINT FK1_Session FOREIGN KEY (RoomID) REFERENCES Room(RoomID),
		CONSTRAINT FK2_Session FOREIGN KEY (ClientID) REFERENCES Client(ClientID),
		CONSTRAINT FK3_Session FOREIGN KEY (SubjectID) REFERENCES
[Subject](SubjectID),
		CONSTRAINT FK4_Session FOREIGN KEY (TutorID) REFERENCES Tutor(TutorID),
		CONSTRAINT FK5_Session FOREIGN KEY (SessionMeetingDayID) REFERENCES
SessionMeetingDay(SessionMeetingDayID)
)
-- End creating the Session table

	-- Creating the Check table
	CREATE TABLE [Check] (
		-- Columns for the Check table
		CheckID int identity,
		CheckNumber varchar(20) not null,
		DateIssued date not null,
		Amount money not null,
		TutorID int not null,
		-- Constraints on the Check table
		CONSTRAINT PK_Check PRIMARY KEY (CheckID),
		CONSTRAINT FK1_Check FOREIGN KEY (TutorID) REFERENCES Tutor(TutorID)
)
-- End creating the Check table
```

Data Creation

SQL Code:

```
/*
	Section: Data Creation

	Action: Building INSERT statements to populate our newly created tables with
business data. Tables will be populated in the same order they were created. The first 5
INSERTs will be created using the defaults that SQL provides for the primary keys; the
last 5 INSERTs will be created NOT using the default settings and hardcoding that value
into the column.

*/

-- Adding Data to the Client table
INSERT INTO Client(FirstName, MiddleInitial, LastName, EmailAddress)
VALUES
('Morgan',null, 'Johnson', 'mjo@dodomain.xyz'),
```

```sql
('Donovan','J', 'Mitchell', 'dmit@dodomain.xyz'),
('Dominique','D', 'Doe', 'ddoe@dodomain.xyz'),
('Clark',null, 'Kent', 'cke@dodomain.xyz'),
('Peter', 'P', 'Parker', 'ppar@dodomain.xyz'),
('Joe','N', 'Shmo', 'jshmo@dodomain.xyz')

-- Adding Data to the Subject table
INSERT INTO [Subject](SubjectName, GradeLevel)
VALUES
('Math', '6th'),
('Math', '3rd'),
('English', '3rd'),
('English', '6th'),
('Math', '8th')

-- Adding Data to the Tutor table
INSERT INTO Tutor(FirstName, MiddleInitial, LastName, EmailAddress)
VALUES
('Darrell','L', 'Nelson II', 'dnel@dodomain.xyz'),
('Hannah','T', 'Turner', 'hturner@dodomain.xyz'),
('Isaac','J', 'Curry', 'icur@dodomain.xyz'),
('Nikolaus', null, 'Claus', 'nclaus@dodomain.xyz')

--Adding Data to the ClientSubject table
/* A select statement needs to be embedded into the INSERT statement in order to properly
look up the PKs from other tables. Email Address is the proper way to look up clients in
the client table since that is the only unique attribute in that table. Both SubjectName
and GradeLevel are required to properly identify the SubjectID */

INSERT INTO ClientSubject (ClientID, SubjectID)
VALUES ((SELECT ClientID FROM Client WHERE EmailAddress='mjo@dodomain.xyz'),
(SELECT SubjectID FROM [Subject] WHERE SubjectName='Math' AND GradeLevel='6th')),
((SELECT ClientID FROM Client WHERE EmailAddress='mjo@dodomain.xyz'),
(SELECT SubjectID FROM [Subject] WHERE SubjectName='English' AND GradeLevel='6th')),
((SELECT ClientID FROM Client WHERE EmailAddress='dmit@dodomain.xyz'),
(SELECT SubjectID FROM [Subject] WHERE SubjectName='English' AND GradeLevel='3rd')),
((SELECT ClientID FROM Client WHERE EmailAddress='dmit@dodomain.xyz'),
(SELECT SubjectID FROM [Subject] WHERE SubjectName='Math' AND GradeLevel='3rd')),
((SELECT ClientID FROM Client WHERE EmailAddress='jshmo@dodomain.xyz'),
(SELECT SubjectID FROM [Subject] WHERE SubjectName='Math' AND GradeLevel='8th'))

--Adding Data to the ClientTutor table
/* A select statement needs to be embedded into the INSERT statement in order to properly
look up the PKs from other tables. Email Address is the proper way to look up
clients/tutors in their respective tables since it is a unique attribute. */

INSERT INTO ClientTutor (ClientID, TutorID)
VALUES ((SELECT ClientID FROM Client WHERE EmailAddress='mjo@dodomain.xyz'),
(SELECT TutorID FROM Tutor WHERE EmailAddress='dnel@dodomain.xyz')),
((SELECT ClientID FROM Client WHERE EmailAddress='mjo@dodomain.xyz'),
(SELECT TutorID FROM Tutor WHERE EmailAddress='icur@dodomain.xyz')),
((SELECT ClientID FROM Client WHERE EmailAddress='dmit@dodomain.xyz'),
(SELECT TutorID FROM Tutor WHERE EmailAddress='dnel@dodomain.xyz')),
((SELECT ClientID FROM Client WHERE EmailAddress='dmit@dodomain.xyz'),
(SELECT TutorID FROM Tutor WHERE EmailAddress='icur@dodomain.xyz')),
((SELECT ClientID FROM Client WHERE EmailAddress='ddoe@dodomain.xyz'),
(SELECT TutorID FROM Tutor WHERE EmailAddress='hturner@dodomain.xyz')),
((SELECT ClientID FROM Client WHERE EmailAddress='jshmo@dodomain.xyz'),
(SELECT TutorID FROM Tutor WHERE EmailAddress='dnel@dodomain.xyz'))

/* All PKs for the rest of the tables will now be manually inserted instead
of relying on SQL to provide defaults. */

-- Adding Data to the Room table using

-- Turn on the identity insert for the Room section
SET IDENTITY_INSERT Room ON;
GO

INSERT INTO Room(RoomID, RoomNumber, SubjectID)
VALUES (1, '1A', (SELECT SubjectID FROM [Subject] WHERE SubjectName='Math' AND
GradeLevel='6th' )),
(2,'1B', (SELECT SubjectID FROM [Subject] WHERE SubjectName='English' AND
GradeLevel='3rd')),
(3,'1C', (SELECT SubjectID FROM [Subject] WHERE SubjectName='Math' AND GradeLevel='3rd'))

-- Turn off the identity insert for the Room section
SET IDENTITY_INSERT Room OFF;
GO

-- Adding Data to the SessionMeetingDay table using

-- Turn on the identity insert for the SessionMeetingDay section
SET IDENTITY_INSERT SessionMeetingDay ON;
GO

INSERT INTO SessionMeetingDay(SessionMeetingDayID, [DayOfWeek])
VALUES (1, 'Monday'),
(2, 'Tuesday'),
(3, 'Wednesday'),
(4, 'Thursday'),
(5, 'Friday'),
(6, 'Saturday'),
(7, 'Sunday')


-- Turn off the identity insert for the SessionMeetingDay section
SET IDENTITY_INSERT SessionMeetingDay OFF;
GO

-- Adding Data to the TutorSubject table using

-- Turn on the identity insert for the TutorSubject section
SET IDENTITY_INSERT TutorSubject ON;
GO

INSERT INTO TutorSubject(TutorSubjectID, TutorRate, TutorID, SubjectID)
VALUES (1, 25, (SELECT TutorID FROM Tutor WHERE EmailAddress='dnel@dodomain.xyz'),
 (SELECT SubjectID FROM [Subject] WHERE SubjectName='Math' AND GradeLevel='6th' )),
(2, 20, (SELECT TutorID FROM Tutor WHERE EmailAddress='dnel@dodomain.xyz'),
 (SELECT SubjectID FROM [Subject] WHERE SubjectName='English' AND GradeLevel='3rd' )),
(3, 30, (SELECT TutorID FROM Tutor WHERE EmailAddress='hturner@dodomain.xyz'),
 (SELECT SubjectID FROM [Subject] WHERE SubjectName='Math' AND GradeLevel='8th' )),
(4, 15, (SELECT TutorID FROM Tutor WHERE EmailAddress='icur@dodomain.xyz'),
 (SELECT SubjectID FROM [Subject] WHERE SubjectName='English' AND GradeLevel='3rd' ))
```

```sql
-- Turn off the identity insert for the TutorSubject section
SET IDENTITY_INSERT TutorSubject OFF;
GO

-- Adding Data to the Session table using

-- Turn on the identity insert for the Session section
SET IDENTITY_INSERT [Session] ON;
GO

INSERT INTO [Session](SessionID, [Date], TimeStart, TimeEnd, RoomID, ClientID, SubjectID,
TutorID, SessionMeetingDayID)
VALUES (1, '12/3/2016', '16:00', '17:00', (SELECT RoomID FROM Room WHERE
RoomNumber='1A'),
 (SELECT ClientID FROM Client WHERE EmailAddress='mjo@dodomain.xyz'),
 (SELECT SubjectID FROM [Subject] WHERE SubjectName='Math' AND GradeLevel='6th'),
 (SELECT TutorID FROM Tutor WHERE EmailAddress='dnel@dodomain.xyz'),
 (SELECT SessionMeetingDayID FROM SessionMeetingDay WHERE [DayOfWeek]='Monday')),


 (2, '12/4/2016', '13:00', '14:30', (SELECT RoomID FROM Room WHERE RoomNumber='1B'),
 (SELECT ClientID FROM Client WHERE EmailAddress='dmit@dodomain.xyz'),
 (SELECT SubjectID FROM [Subject] WHERE SubjectName='English' AND GradeLevel='3rd'),
 (SELECT TutorID FROM Tutor WHERE EmailAddress='dnel@dodomain.xyz'),
 (SELECT SessionMeetingDayID FROM SessionMeetingDay WHERE [DayOfWeek]='Tuesday')),

 (3, '12/4/2016', '16:00', '17:00', (SELECT RoomID FROM Room WHERE RoomNumber='1C'),
 (SELECT ClientID FROM Client WHERE EmailAddress='dmit@dodomain.xyz'),
 (SELECT SubjectID FROM [Subject] WHERE SubjectName='Math' AND GradeLevel='3rd'),
 (SELECT TutorID FROM Tutor WHERE EmailAddress='dnel@dodomain.xyz'),
 (SELECT SessionMeetingDayID FROM SessionMeetingDay WHERE [DayOfWeek]='Tuesday')),

  (4, '12/5/2016', '16:00', '17:00', (SELECT RoomID FROM Room WHERE RoomNumber='1A'),
 (SELECT ClientID FROM Client WHERE EmailAddress='jshmo@dodomain.xyz'),
 (SELECT SubjectID FROM [Subject] WHERE SubjectName='Math' AND GradeLevel='3rd'),
 (SELECT TutorID FROM Tutor WHERE EmailAddress='dnel@dodomain.xyz'),
 (SELECT SessionMeetingDayID FROM SessionMeetingDay WHERE [DayOfWeek]='Thursday'))

-- Turn off the identity insert for the Session section
SET IDENTITY_INSERT Session OFF;
GO

-- Adding Data to the Check table using

-- Turn on the identity insert for the Check section
SET IDENTITY_INSERT [Check] ON;
GO

INSERT INTO [Check](CheckID, CheckNumber, DateIssued, Amount, TutorID)
VALUES (1, '123456789', '11/16/2018', '125.00', (SELECT TutorID FROM Tutor WHERE
EmailAddress='dnel@dodomain.xyz')),
          (2, '112233441', '11/9/2018', '135.00', (SELECT TutorID FROM Tutor WHERE
EmailAddress='hturner@dodomain.xyz')),
          (3, '142536482', '12/2/2018', '200.00', (SELECT TutorID FROM Tutor WHERE
EmailAddress='icur@dodomain.xyz'))


-- Turn off the identity insert for the Check section
```

```
SET IDENTITY_INSERT [Check] OFF;
GO
```

Data Manipulation

Cases for the business purpose of each update or insert is in the comment sections of the SQL Code.

SQL Code:

```
/*
       Section: Data Manipulation

       Action: Building 3 UPDATE statements and 3 DELETE statements that tie into company
business rules.
*/

-- UPDATE #1: Email address for client, Peter Parker, has changed. One of our business
-- rules is that each client MUST have a unique email address in our system.

UPDATE Client SET EmailAddress = 'notspiderman@dodomain.xyz'
WHERE EmailAddress='ppar@dodomain.xyz'

-- UPDATE #2: Tutor, Isaac, raised his rate to teach 3rd grade engligh from $15 to $25 an
hour

UPDATE TutorSubject SET TutorRate = 25
WHERE TutorRate=15 AND TutorID = (SELECT TutorID FROM Tutor WHERE
EmailAddress='icur@dodomain.xyz')
AND SubjectID = (SELECT SubjectID FROM [Subject] WHERE SubjectName='English' AND
GradeLevel='3rd' )

-- UPDATE #3: Tutor, Hannah, got married and wants to change her last name in the
database.

UPDATE Tutor SET LastName = 'Myers'
WHERE EmailAddress = 'hturner@dodomain.xyz'

-- DELETE #1: The session with the client Joe is being canceled due to illness

DELETE FROM [Session] WHERE [Date] = '12/5/2016' AND TimeStart = '16:00'
AND ClientID =  (SELECT ClientID FROM Client WHERE EmailAddress='jshmo@dodomain.xyz');

-- DELETE #2: Tutor, Mr. Claus has decided to terminate his arrangement with Community
Tutors, LLC.

DELETE FROM Tutor WHERE EmailAddress = 'nclaus@dodomain.xyz';

-- DELETE #3: Community Tutors has decided to close on Sundays

DELETE FROM SessionMeetingDay WHERE [DayOfWeek] = 'Sunday';
```

Answering Data Questions

Screenshots of the outputs have been pasted at the bottom of the SELECT statement that created them.

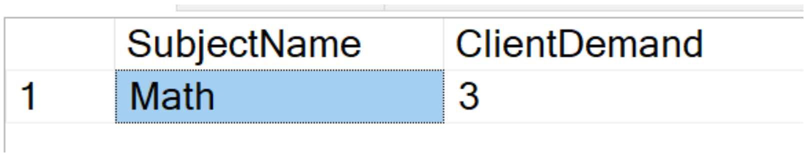
SQL Code:

```
*
       Section: Answering Data Questions

       Action: Create SELECT statements that answer the 5 data questions presented in the
'IST-659 M402 Final Project' document.The answers to these questions will provide key
insights into our business process and highlight potential areas for improvement.
*/

-- Question #1:      Which tutoring subject is the most popular?
-- Identifies highest demand on our resources (classroom, supplies, tutors, etc.)

SELECT TOP 1
Subject.SubjectName,
COUNT(ClientSubject.SubjectID) AS ClientDemand FROM ClientSubject
JOIN [Subject] ON Subject.SubjectID = ClientSubject.SubjectID
GROUP BY Subject.SubjectName
ORDER BY ClientDemand DESC
GO
```



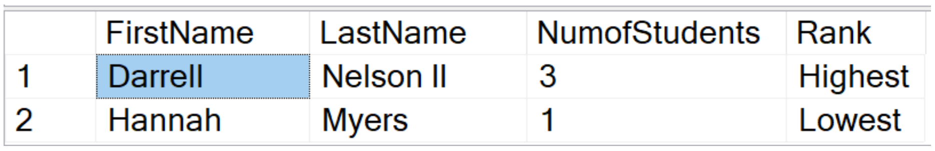
	SubjectName	ClientDemand
1	Math	3

Screenshot1. Most Popular Subject

```
/* EXPLAINATION:  Counting all the SubjectIDs with same values in the ClientSubject table
is the most appropriate way to get an accurate count of all the subjects our clients are
getting tutored in. I also pulled SubjectName from the Subject table to view which
Subject had the highest count. */


-- Question #2:      Which tutor is the most/least popular?
-- Identifies teaching styles that are the most/least effective with our clients

WITH StudentCounts
AS
(
      SELECT TutorID, COUNT(ClientTutor.ClientID) AS NumOfStudents
      FROM  ClientTutor
      GROUP BY TutorID
),
StudentCountsMinMax
AS
(
      SELECT MIN(NumOfStudents) AS MinNumOfStudents,
             MAX(NumOfStudents) AS MaxNumOfStudents
      FROM  StudentCounts
)
SELECT TU.FirstName, TU.LastName, SC.NumofStudents,
      CASE SC.NumOfStudents WHEN MM.MinNumOfStudents THEN 'Lowest' ELSE 'Highest' END AS
[Rank]
FROM  StudentCounts AS SC
INNER JOIN StudentCountsMinMax AS MM ON SC.NumOfStudents IN (MM.MinNumOfStudents,
MM.MaxNumOfStudents)
INNER JOIN Tutor AS TU ON TU.TutorID = SC.TutorID
```



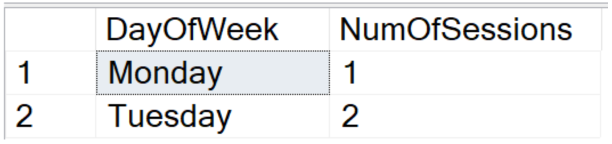
	FirstName	LastName	NumofStudents	Rank
1	Darrell	Nelson II	3	Highest
2	Hannah	Myers	1	Lowest

Screenshot 2: Most/Least Popular Tutor

```
/* EXPLAINATION:  Counting all the ClientIDs in table ClientTutor and grouping them by
TutorID returns the total amount of clients that have been scheduled to meet or have
meeten with a particular tutor in the past. Creating an alias column that accepts integer
values allows us to look at the min/max of students in the list of tutors. A "WHEN"
clause allows us to display the min calculation as 'Lowest' in a column called
StudentCounts and the max calculation as 'Highest in a column.' */


 -- Question #3:     What day of the week has the highest demand?
-- Identifies potential bottlenecks in scheduling

SELECT
SessionMeetingDay.DayOfWeek,
 Count(SessionID) as NumOfSessions FROM [Session]
 JOIN SessionMeetingDay ON SessionMeetingDay.SessionMeetingDayID =
Session.SessionMeetingDayID
 GROUP BY SessionMeetingDay.DayOfWeek
GO
```



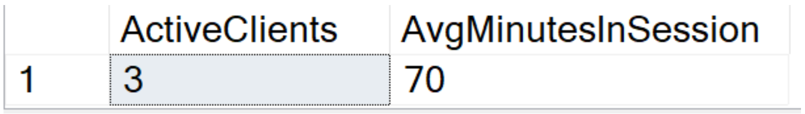
	DayOfWeek	NumOfSessions
1	Monday	1
2	Tuesday	2

Screenshot3: Number of Sessions on Each Day

```
/* EXPLAINATION:  Counting all the SessionIDs in the same DayOfWeek column is the most
appropriate way to get an accurate count of all the sessions taking place each day in the
week. I didn't use the "Top 1" function to only select the best value because I believe
it makes more business sense to see how ALL of the days of the week fair against each
other. There will always be 7 days in a week, so this table output should be appropriate
at all times in the near future.
*/


-- Question #4:      What are the average hours of tutoring per client?
-- Allows us to forecast resource requirements based on client growth

SELECT
COUNT(Session.ClientID) as ActiveClients,
AVG(DATEDIFF(n, TimeStart, TimeEnd)) as AvgMinutesInSession
FROM Client
 JOIN [Session] ON Session.ClientID = Client.ClientID
GO
```



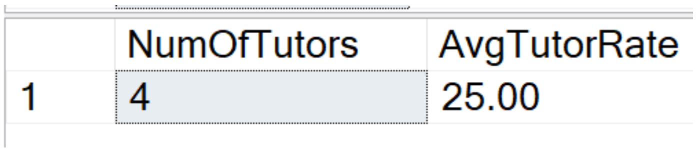
	ActiveClients	AvgMinutesInSession
1	3	70

Screenshot 4: Average Hours of Tutoring per Client

```
/* EXPLAINATION:  Counting all the ClientIDs that are enrolled in sessions will return
the total number of active clients. Taking the difference of the start and stop times of
each session using DATEDIFF and averaging them returns the average time spent in minutes
of each client in a session.
*/


-- Question #5: What is the average tutor rate ($/hour charged to clients)?
-- Identifies what new & unexperienced tutors should charge

SELECT
COUNT(TutorSubject.TutorID) as NumOfTutors,
AVG(TutorSubject.TutorRate) as AvgTutorRate
FROM TutorSubject
GO
```



	NumOfTutors	AvgTutorRate
1	4	25.00

Screenshot 5: Average $/hr charged per client

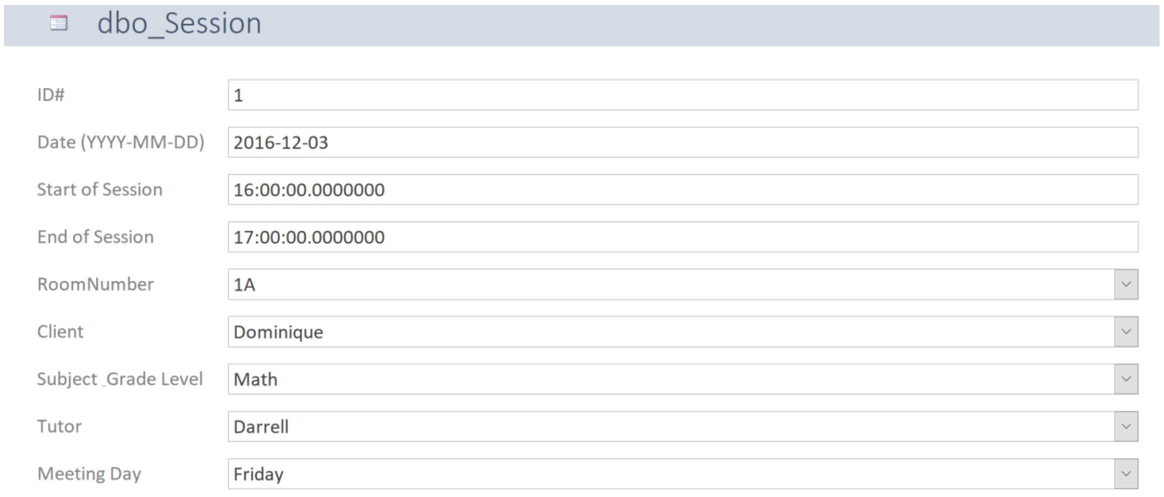
```
/* EXPLAINATION:  Counting all the TutorIDs and averaging their TutorRate in the
TutorSubject table is the most efficient way to calculate what our tutors charge on
average per hour.
*/
```

Implementation

Forms

Session Form

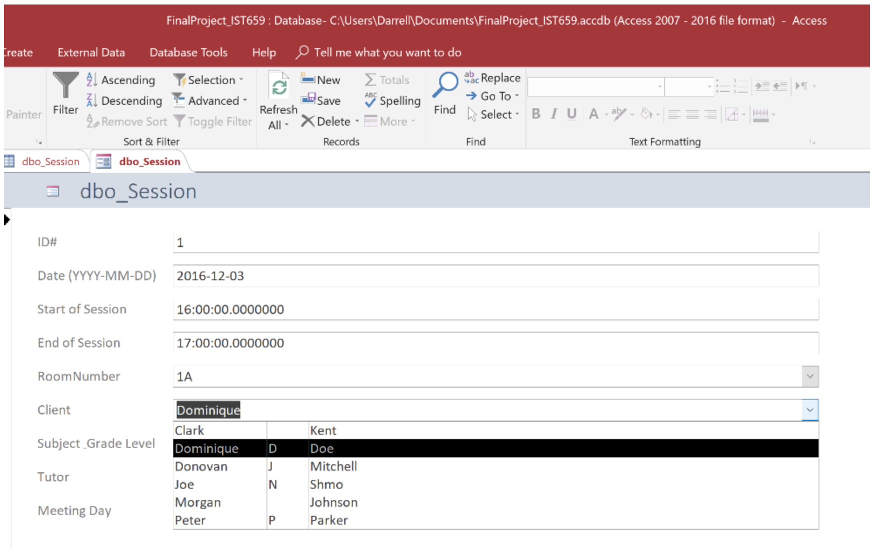
Front End created for the Session table for clients to have a visible view of their scheduled tutoring session:



dbo_Session

ID#	1
Date (YYYY-MM-DD)	2016-12-03
Start of Session	16:00:00.0000000
End of Session	17:00:00.0000000
RoomNumber	1A
Client	Dominique
Subject_Grade Level	Math
Tutor	Darrell
Meeting Day	Friday

The drop-down list for Client reveals the First, Middle, and Last name of each student in the database:

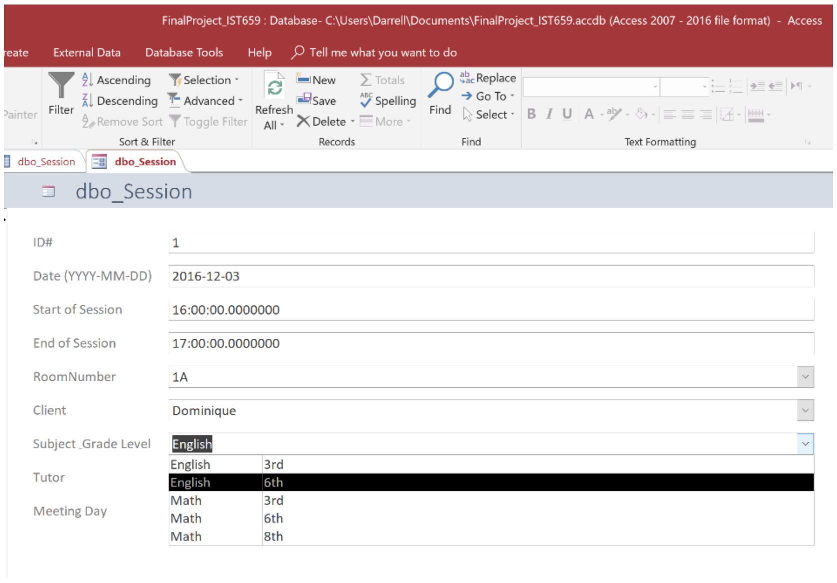

FinalProject_IST659 : Database- C:\Users\Darrell\Documents\FinalProject_IST659.accdb (Access 2007 - 2016 file format) - Access
reate External Data Database Tools Help Tell me what you want to do
Painter Filter Ascending Descending Remove Sort Selection Advanced Toggle Filter Sort & Filter
Refresh All New Save Delete Totals Spelling More Records
Find Replace Go To Select Find
Text Formatting
dbo_Session dbo_Session

dbo_Session

ID#	1
Date (YYYY-MM-DD)	2016-12-03
Start of Session	16:00:00.0000000
End of Session	17:00:00.0000000
RoomNumber	1A
Client	Dominique
Subject_Grade Level	
Tutor	
Meeting Day	

Clark		Kent
Dominique	D	Doe
Donovan	J	Mitchell
Joe	N	Shmo
Morgan		Johnson
Peter	P	Parker

The drop-down list for Subject_Grade Level reveals the Subject and Grade Level of all students that have requested to be tutored in a subject (not all clients have decided which subjects they want to be tutored in):

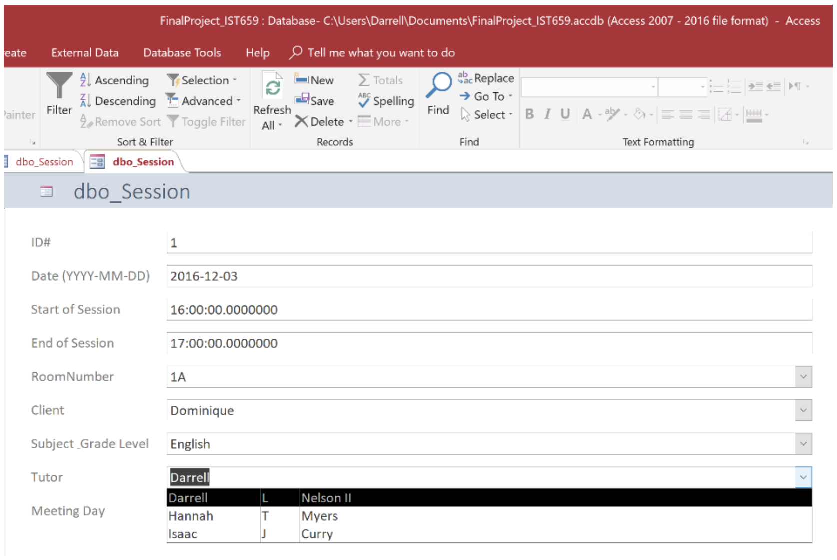

FinalProject_IST659 : Database- C:\Users\Darrell\Documents\FinalProject_IST659.accdb (Access 2007 - 2016 file format) - Access
reate External Data Database Tools Help Tell me what you want to do
Painter Filter Ascending Descending Remove Sort Selection Advanced Toggle Filter Sort & Filter
Refresh All New Save Delete Totals Spelling More Records
Find Replace Go To Select Find
Text Formatting
dbo_Session dbo_Session

dbo_Session

ID#	1
Date (YYYY-MM-DD)	2016-12-03
Start of Session	16:00:00.0000000
End of Session	17:00:00.0000000
RoomNumber	1A
Client	Dominique
Subject_Grade Level	English
Tutor	
Meeting Day	

English	3rd
English	6th
Math	3rd
Math	6th
Math	8th

The drop-down list for Tutor reveals the First, Middle, and Last name of each tutor in our database:



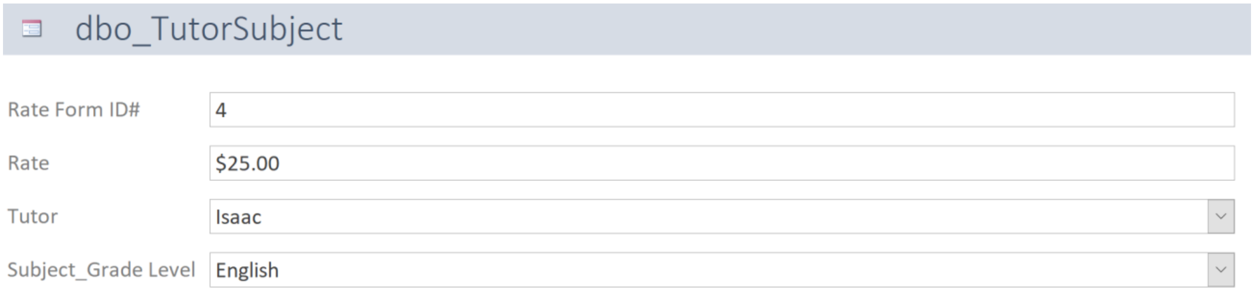
dbo_Session

ID#	1
Date (YYYY-MM-DD)	2016-12-03
Start of Session	16:00:00.0000000
End of Session	17:00:00.0000000
RoomNumber	1A
Client	Dominique
Subject_Grade Level	English
Tutor	Darrell
Meeting Day	

Darrell	L	Nelson II
Hannah	T	Myers
Isaac	J	Curry

Tutor Rate Form

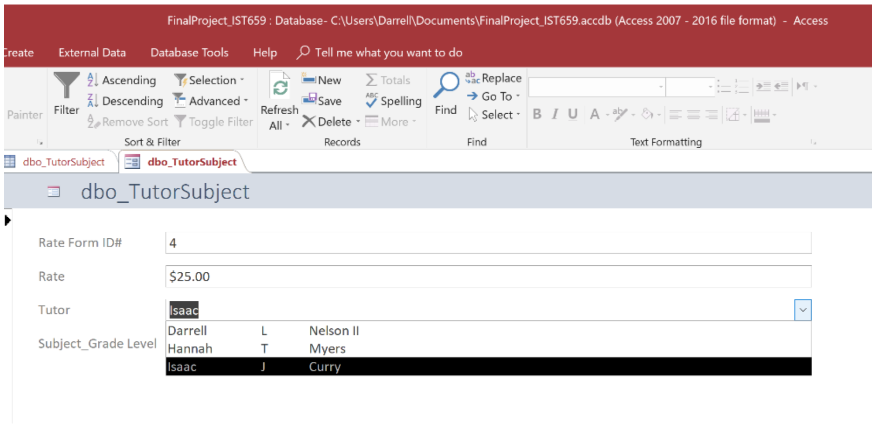
A form created for interested parties to view how much a tutor chargers per hour for a particular subject:



dbo_TutorSubject

Rate Form ID#	4
Rate	$25.00
Tutor	Isaac
Subject_Grade Level	English

The drop-down list for Tutor reveals the First, Middle, and Last name of each tutor in our database:

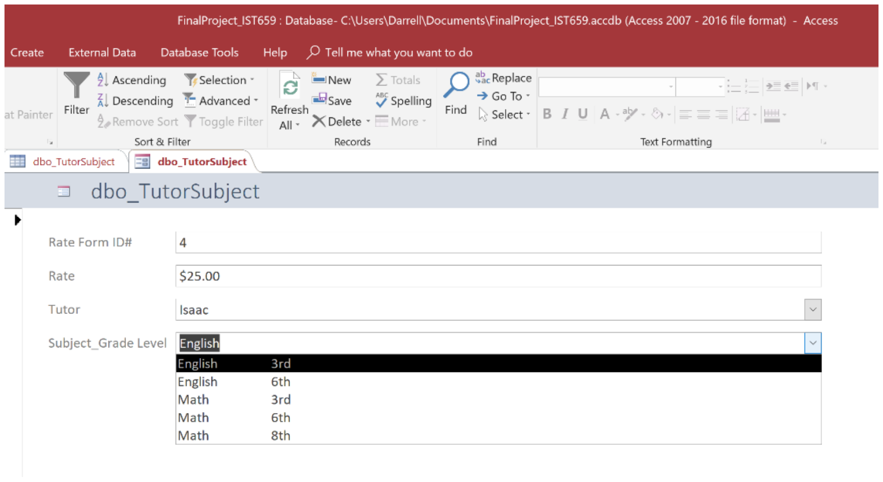


dbo_TutorSubject

Rate Form ID#	4
Rate	$25.00
Tutor	Isaac
Subject_Grade Level	

Darrell	L	Nelson II
Hannah	T	Myers
Isaac	J	Curry

The drop-down list for Subject_Grade Level reveals the Subjects and Grade Levels that the Tutor is qualified to teach:

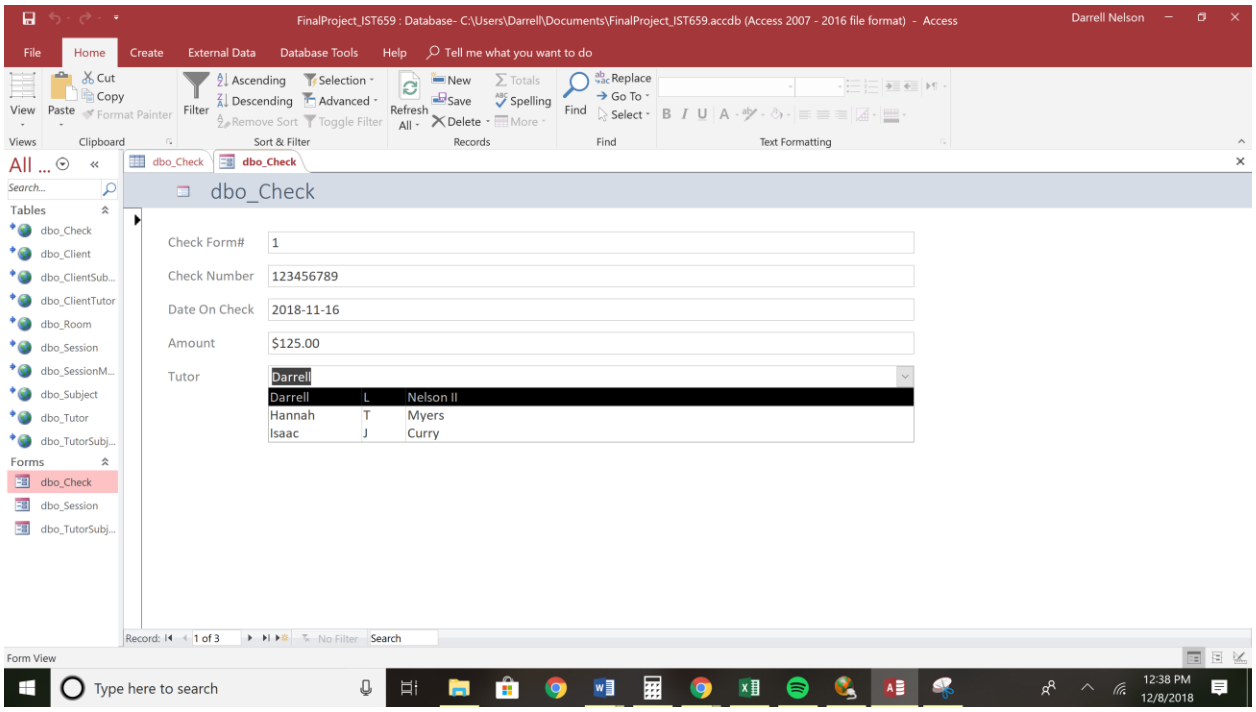

FinalProject_IST659 : Database- C:\Users\Darrell\Documents\FinalProject_IST659.accdb (Access 2007 - 2016 file format) - Access
Create
External Data
Database Tools
Help
Tell me what you want to do
dbo_TutorSubject

Rate Form ID#	4
Rate	$25.00
Tutor	Isaac
Subject_Grade Level	English

English	3rd
English	6th
Math	3rd
Math	6th
Math	8th

Check Form

A form that allows interested parties to see how much a tutor got paid on a given payday. The drop-down list for Tutor reveals the First, Middle, and Last name of each tutor in our database:



Check Form#	1
Check Number	123456789
Date On Check	2018-11-16
Amount	$125.00
Tutor	Darrell

Darrell	L	Nelson II
Hannah	T	Myers
Isaac	J	Curry

Reports

5 reports have been created to assist in answering the 5 key data questions. These reports can be generated and given to any party that believes an answer to this particular data question is of value. This

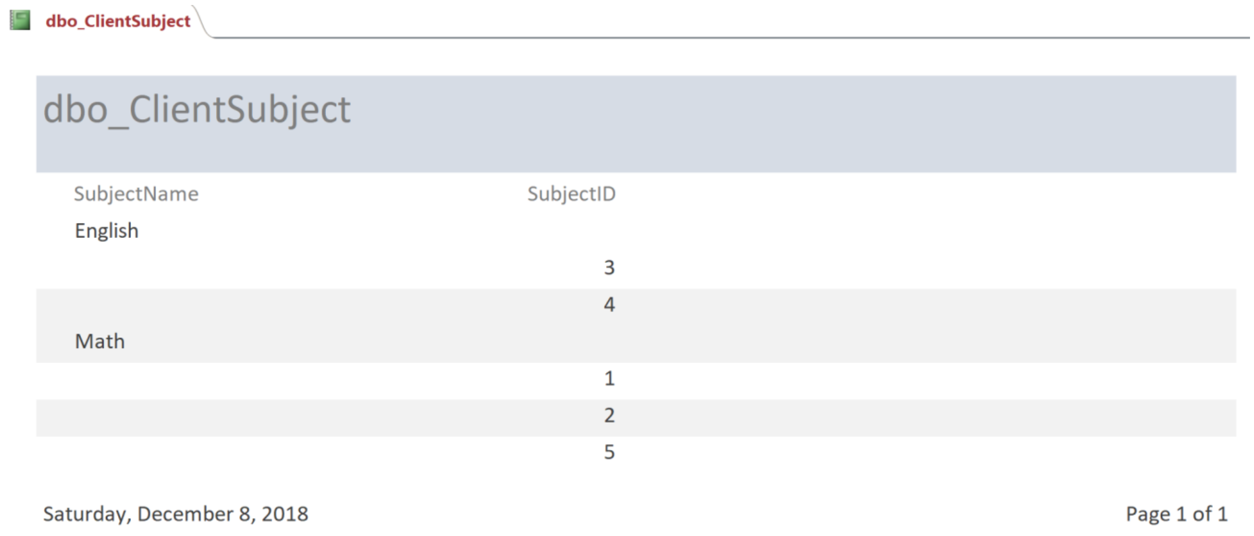
reduces the amount of users on actual database and prevents users from slowing down the server with convoluted queries.

Data Question 1:

1. Which tutoring subject is the most popular?
    - **Identifies highest demand**

The below report lists out all of the Subjects that the company has ever taught in a session. The SubjectIDs column is a representation of the number of clients that have requested each subject. The report shows that Math has the most instances of requests.

I'm unaware of how to only return the MAXIMUM value in column SubjectID in Access so the report gives us all the Subjects.

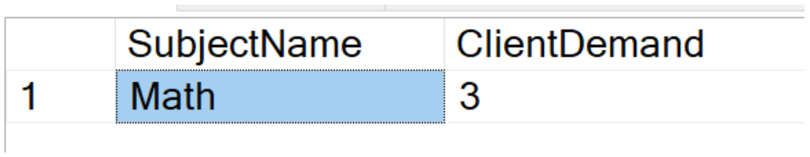

dbo_ClientSubject

dbo_ClientSubject

SubjectName	SubjectID
English	
	3
	4
Math	
	1
	2
	5

Saturday, December 8, 2018 Page 1 of 1

The actual SQL output for Data Question 1:



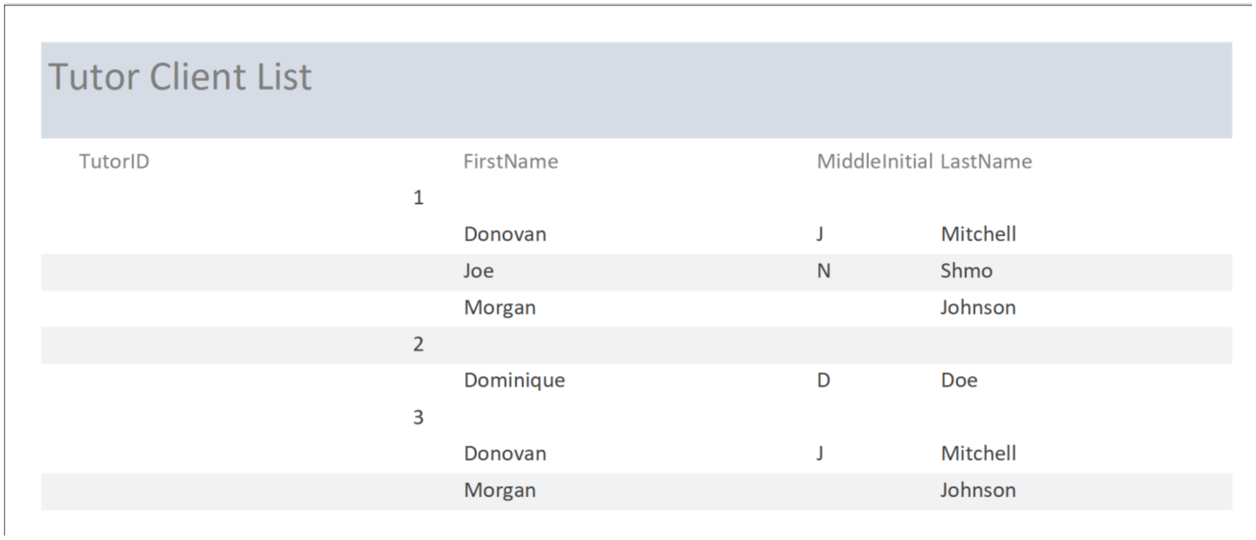
	SubjectName	ClientDemand
1	Math	3

Data Question 2:

1. Which tutor is the most/least popular?
    - **Identifies teaching styles that are the most/least effective with our clients**

The report below lists all of the clients that a tutor has had in a session. By this view you can tell which tutor has the most clients and which has the least.

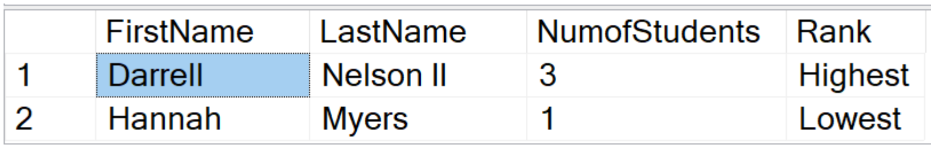
I’m unaware of how to only return the MAXIMUM and MINIMUM values in a column in an Access report so this report lists all the clients from all the tutors.



Tutor Client List

TutorID	FirstName	MiddleInitial	LastName
1			
	Donovan	J	Mitchell
	Joe	N	Shmo
	Morgan		Johnson
2			
	Dominique	D	Doe
3			
	Donovan	J	Mitchell
	Morgan		Johnson

The actual SQL output for Data Question 2:

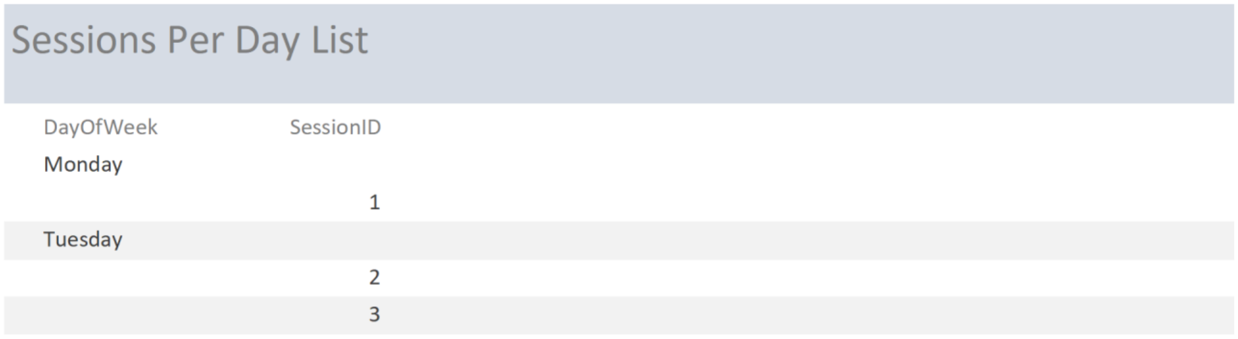


	FirstName	LastName	NumofStudents	Rank
1	Darrell	Nelson II	3	Highest
2	Hannah	Myers	1	Lowest

Data Question 3:

1. What day of the week has the highest demand?
   - **Identifies potential bottlenecks in scheduling**

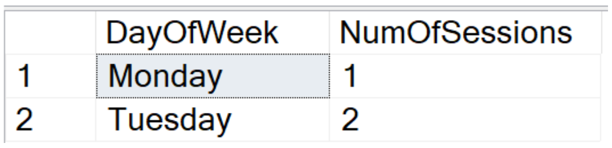
This report lists all of the days that a session was ever held and lists all of these sessions by their ID number.



Sessions Per Day List

DayOfWeek	SessionID
Monday	
	1
Tuesday	
	2
	3

The actual SQL output for Data Question 3:

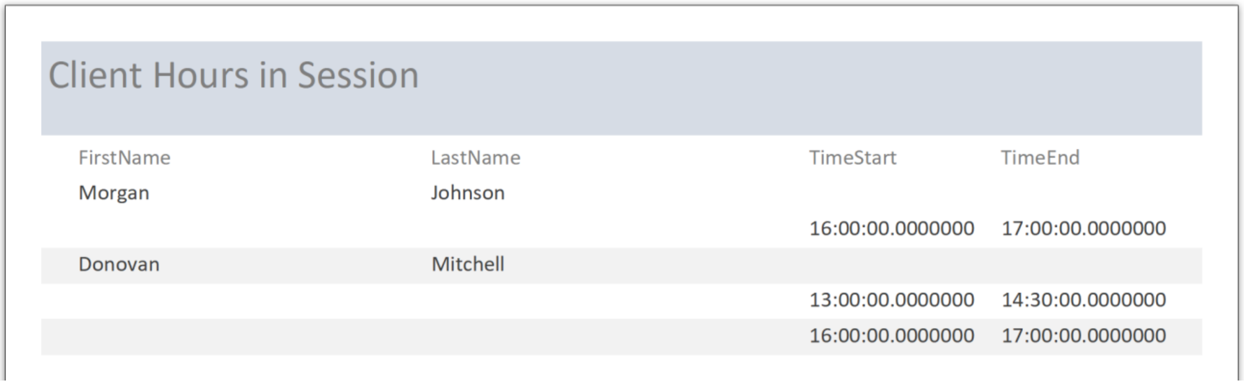


	DayOfWeek	NumOfSessions
1	Monday	1
2	Tuesday	2

Data Question 4:

1. What are the average hours of tutoring per client?
   - **Allows us to forecast resource requirements based on client growth**

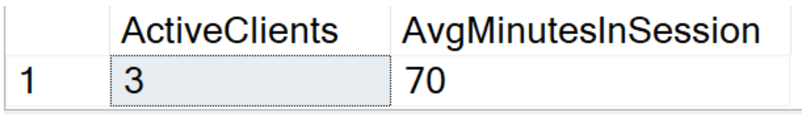
The below report lists the start and end time of each session a client has throughout the week. You can then tally these values and divide by the total number of clients in the report.



Client Hours in Session

FirstName	LastName	TimeStart	TimeEnd
Morgan	Johnson		
		16:00:00.0000000	17:00:00.0000000
Donovan	Mitchell		
		13:00:00.0000000	14:30:00.0000000
		16:00:00.0000000	17:00:00.0000000

The actual SQL output for Data Question 4:

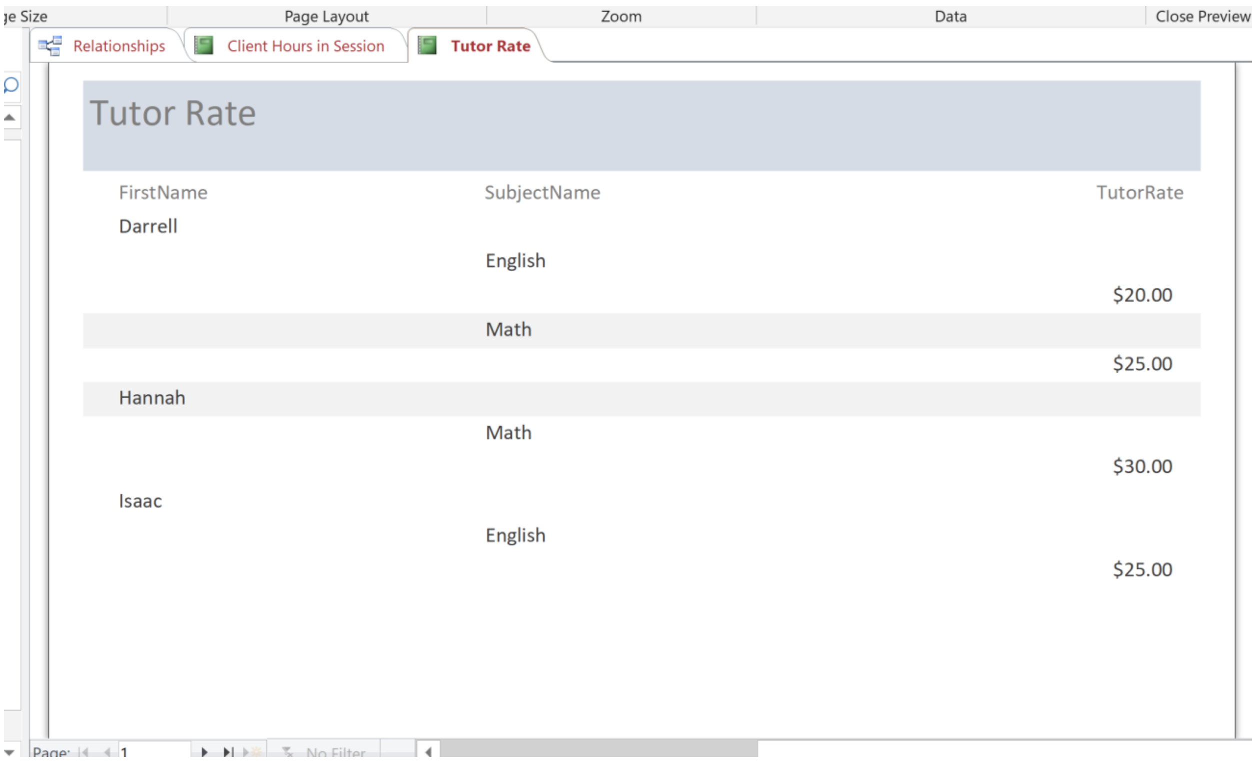


	ActiveClients	AvgMinutesInSession
1	3	70

Data Question 5:

1. What is the average tutoring rate ($/hour charged to clients)?
    - **Identifies what new & unexperienced tutors should charge**

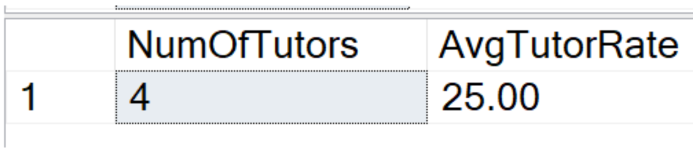
The below report lists the tutoring rate for each tutor based on their subject. An average can be calculated by summing up the values in the TutorRate column and dividing them by the number of instances.



Tutor Rate

FirstName	SubjectName	TutorRate
Darrell		
	English	
		$20.00
	Math	
		$25.00
Hannah		
	Math	
		$30.00
Isaac		
	English	
		$25.00

The actual SQL output for Data Question 5:



	NumOfTutors	AvgTutorRate
1	4	25.00

Reflection

Building a SQL database was a lot easier than I thought, however manipulating the database and creating queries that answer pertinent questions was truly difficult. I had to omit one of my data questions from this final document due to my inability to properly query it in the database. I have gained a new appreciation for the expertise it requires to create queries and provide useful information to stake holders.

When inputting data into tables in the 'Data Creation' section, the table 'Subject' caused some grief. I did not create a unique identifier that allowed me to select a unique row in the table without hardcoding the PK or using a subSELECT statements that looked for two arguments. Making sure each row in a entity table can be referenced with one proper variable (unique constraint) is a nice luxury that I wish I had known in the beginning.

This database also has a lot of relationships that tie Tutors, Clients, and/or Subjects together. Before I created the plethora of FKs that I did in my Logical model, it would have been better to create references to bridge tables (i.e. ClientSubjet, ClientTutor, and TutorSubject) to reduce the amount of FKs and keep things more organized. Following the flow of data was very hard initially when building this database due to all of the interconnections.

It would have been beneficial to create a VIEW that combines Tutors, Clients, and Sessions early on. This would of allowed me to easily look at different relationships and decipher what types of UPDATE and DELETE statements are appropriate. I am truly thankful that I created *Figure 1. Old Database* before I began coding. It allowed me to figure out relationships when I got stuck looking at the Logical model.

As a budding information professional, this experience will truly help me when I'm building my own databases. They might not be in SQL or any other DBMS but knowing that I should build/organize tables in 1NF will give me a leg up when I need to lean on other teams to build front-end applications on my database.

Glossary

1. Entity = the thing you need to store data about (e.g. tutors, clients, etc.)
2. Attribute = characteristics of things you need to store
3. Cardinality = the number of elements in a set or other grouping, as a property of that grouping
4. Normalize = organizing data in such a way that guarantees all of our specified query data will be accessed when we need to analyze/view/update a specific table
5. Unique key = indicates a unique constraint; no other entry in this table can have the same value
6. Primary Key = unique identifier for that table; allows you to uniquely identify every row in that table
7. Foreign Key = a primary key that is copied into another table to create relationships between the FK table and the PK table
8. Surrogate Key = any column or set of columns that can be declared as the primary key instead of a "real" or natural key

Appendix

Table 1-A

Entity	**Attribute**
Client	ClientID ClientName [r] EmailAddress [rc]
Subject	SubjectID SubjectName [r] GradeLevel [r]
Tutor	TutorID TutorName [r] EmailAddress [rc]
Session	SessionID MeetingDays [m] TimeStart [r] TimeEnd [r]
Room	RoomID RoomNumber [r]
Check	CheckID CheckNumber [r] DateIssued [r] Amount [d]

Relationships

1. Each client can be tutored in one or many subjects
2. Each client can have multiple sessions
3. Each session will only have one client. A session can’t exist without a client
4. Each client may have multiple tutors

5. Each subject can be taught in many sessions
6. Each subject can be taught by many tutors
   a. A tutor must teach at least one or more subjects
7. Each subject can be taught in one or many rooms
   a. Each room will only have 1 subject taught in it
8. Each session must have a tutor
9. Each session must have a room
10. Each tutor gets one check from every client he/she has