Fantastic Tigers

Campalot Resorts
Final Submission

Members:
Rita Mihalek, John McQueen, Taylor Campbell, and Molungoa Ramataboe

MILESTONE 1: Conceptual Design

Client Meeting

Meeting Details

We first were able to meet with our client, Campalot Resort last Monday March 20th. All members of our team were present and our meeting lasted approximately 13 minutes. Our question and answers are provided below.

Q&A

1. Do we want to show the amount of reservations each customer has made or just one?
    A. 1.Know the various reservations of customers over time
2. Would you like to extend your services to an online reservation service or keep it just in person or phone calls?
    A. Not online, just in person and phone calls
3. What is the maximum quantity of the service bundles and is it always the same for each service?
    A. We don't want every site for one bundle to be given a water site
4. Do you want there to be a separate table between individual customers and company customers/ walk in customers and pre-book customers?
    A. Useful to know which customers are different, It is up to resorts to keep track of if a customer is walk in, and keep email address or regular address
5. What are the special payment terms to those companies that qualify?
    A. Standard payment terms for customers
6. Do you want us to keep track of how much each amenity cost if so how much?
    A. Depend on the resort- use test data
7. Should there be a reference for which vendors have served which resorts in the past?
    A. Yes, pare down number of vendors we work with
8. Should vendors offering supplies (e.g. firewood, office supplies) be differentiated from the ones who provide services (e.g. pool cleaning, laundry)?
    A. Doesn't matter
9. Should there be an allowance for vendors that have to be paid in advance, or should all vendors be paid on delivery?

A. All vendors should be paid whenever they request

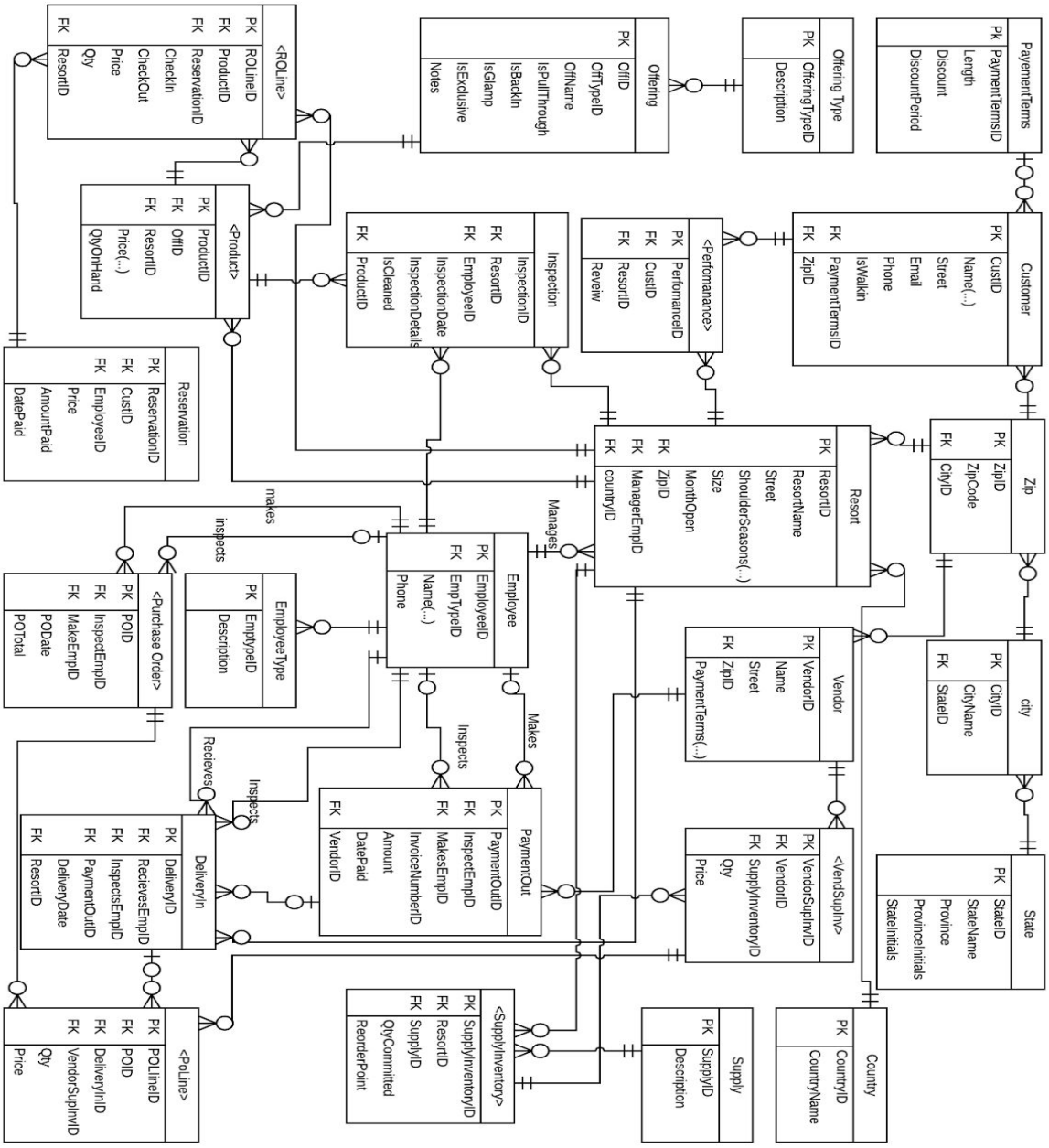
10. Would you like notes on every kind of lodging explaining significant attractions?
    A. Yes, it would be nice
11. Would you like to know when each resort and each single kind of lodging was last cleaned?
    A. Yes, and who it was serviced by
12. What did you like and not like about your original system?
    A. Each resort was on its own but it's time- make it more centrally
13. Should you be able to see the dates open of each resort and what dates are the shoulders?
    A. Show dates for which dates are shoulder dates and when discount rates apply
14. Should customers be able to leave feedback about each resort?
    A. No
15. For the lodging categories, every lodge belongs to one category and can only belong to one category? Ex. Belongs to 50-amp site and glamp unit
    A. No
16. Would you like peak and off-season prices for every kind of category?
    A. Yes, off-season prices are shoulder prices

Assumptions

1. We added an associative entity between resort and supplies. We did this so that an active list of the inventory can be kept for each resort.
2. We assumed we would only want to offer payment terms to our business customers, so we added a zero to one relationship so that payment terms can be added but are not necessary.
3. We created an associative entity between customer and resort called performance. We assumed that they would want to keep track of all the performance reviews from an individual.
4. An associative entity was added between product and vendor. We assumed that each vendor could offer more than one product for some of the resorts.
5. There is a zero to one relationship between purchase order and employee. We assumed that they would want to have record of the Purchase order before it is approved.

ERD

ERD Diagram



ERD Description

An Entity-Relationship Diagram is, at its base level, a blueprint for the developer to help them create a workable database. An entity is simply a table within the database that contains certain attributes. For instance, the topic "Resort" would be an entity, or table, that contained all the information about the different resorts in your organization. That information could include the name of the resort, its location, pertinent dates, etc. for each resort. Each table in the database has some unique attributes, and some attributes that connect to other tables. These connections are the relationships. In the "Resort" table, there might be a need to show which of your employees manages which resort. In that case there would have to be some way to connect "Resort" to the table "Employee". So, we build relationships so that you can access all of the information you need with a single query, instead of having to look it up one table at a time.

If, for example, you were to need the information pertaining to all guests who stayed in Glamp units in July of 2016 and the information was stored in several tables separately across your organization, it could possibly take the better part of a workday to track it all down and organize it into a useable format. This is where the database comes in. If all of that information were instead stored in a central database, where all of the tables were connected to one another in some way, the information you need is right at your fingertips any time you need it.

Your time is valuable and the last thing your business needs is to waste resources trying to find the information you need to make important decisions. That is why the ERD process is so important. This process helps to connect all the parts of your organization and allows you to more efficiently allocate your time. It gives the database designer(s) a plan and specific guidelines to build a database that will be reliable, accurate, and most importantly, usable.

Business Cycles

In our ERD we began by using the default Revenue and Expenditure cycles and then examined what other entities would need to be applied to our diagram. Below are 6 significant differences we made to the default ERDs.

Changes to Default ERDs

1. Instead of keeping the address stored in customer table like the default ERD we kept the street in the Customer table and added a Zip reference table. We did this so we could eliminate the transitive dependencies.
2. Off of our product table we added an Offering table and off of that we added an OfferingType reference table to keep track of all the type of units that each resort has.

3. We added a Reservation entity and attached it to Employee because the employees will be making the reservations for the customers.
4. We added a Supply table and SupplyInventory table and connected it to Resort to be able to keep track of the inventory of each supply at each resort and know when we would need to reorder more.
5. We added an EmployeeType reference table to Employee to show which types that an employee could be.
6. We added an associative entity table called Performance between Resort and Customer. This will allow the company to be able to see that their customers had to say about their time at each resort.

MILESTONE 2: Logical Design

Normalization and Referential Integrity Constraints

Normalization Explanation

Normalization allows the developers to ensure that a database is trustworthy and efficient. It ensures that there is atomicity- that the columns of the database do not need to be broken down any further- and it eliminates data duplication issues(to make sure there is one unique instance of data stored). It is important for the database is trustworthy and efficient(faster) because your time is valuable. You do not want to spend any of that time pulling up information you cannot trust is accurate. If you implement a database system that you find you cannot trust, you and your employees will not use it, and you will have put time and resources into something that does not further your business. Efficiency is important because you have an extremely limited amount of time in which to get the information you need in order to drive your business forward. An efficient database gives you the information you need (and only that which you need) in a timely manner. Through normalization, you get to the information faster and you know that it is both accurate and useful.

Normalized Relations

PaymentTerms (PaymentTermsID, Length, Discount, DiscountPeriod)

Customer (CustID, FName, MInitial, LName, Street, ZipID, Email, Phone, IsWalkIn, PaymentTermsID)

Foreign Key PaymentTermsID references PaymentTerms
Null Allowed
On Delete Set Null

Foreign Key ZipID references Zip
Not Null
On Delete Restrict

Zip (ZipID, ZipCode, CityID)
Foreign Key CityID references City
Not Null
On Delete Restrict

City (CityID, CityName, StateID)
Foreign Key StateID references State
Not Null
On Delete Restrict

State (StateID, StateInitials, StateName, ProvinceInitials, ProvinceName)

Performance (PerformanceID, CustID, ResortID, Review)
Foreign Key CustID references Customer
Not Null
On Delete Restrict

Foreign Key ResortID references Resort
Not Null
On Delete Restrict

Resort (ResortID, ResortName, Street, ZipID, Nearest City, FirstShoulderStart, FirstShoulderEnd, SecondShoulderStart, SecondShoulderEnd, OpenDate, ClosedDate, ManagerEmpID, CountryID, Acres)
Foreign Key ZipID references Zip
Not Null
On Delete Restrict

Foreign Key ManagerEmpID references Employee
Not Null
On Delete Restrict

Foreign Key CountryID references Country
Not Null
On Delete Restrict

Country (CountryID, CountryName)

SupplyInventory (SupplyInventoryID, ResortID, SupplyID, QtyOnHand, QtyCommitted, ReorderPoint)

Foreign Key ResortID references Resort
Not Null
On Delete Restrict

Foreign Key SupplyID references Supply
Not Null
On Delete Restrict

Supply (SupplyID, Description, Qty, Price)

Employee (EmployeeID, EmpTypeID, FName, MInitial, LName, Phone)

Foreign Key EmpTypeID references EmployeeType
Not Null
On Delete Restrict

DeliveryIn (DeliveryID, InspectsEmpID, RecievesEmpID, PaymentOutID, DeliveryDate, ResortID)

Foreign Key InspectEmpID references Employee
Null Allowed
On Delete Set Null

Foreign Key RecievesEmpID references Employee
Not Null
On Delete Restrict

Foreign Key PaymentOutID references PaymentOut
Not Null
On Delete Restrict

Foreign Key ResortID references Resort

Not Null
On Delete Restrict

PurchaseOrder (POID, InspectsEmpID, MakesEmpID, PODate, POTotal)
Foreign Key InspectEmpID references Employee
Null Allowed
On Delete Set Null

Foreign Key MakeEmpID references Employee
Null Allowed
On Delete Restrict

POLine (POLID, POID, DeliveryInID, VendSupInvID, Qty, Price)
Foreign Key POID references PurchaseOrder
Not Null
On Delete Restrict

Foreign Key DeliveryInID references DeliveryIn
Not Null
On Delete Restrict

Foreign Key VendSupInvID references VendSupInv
Not Null
On Delete Restrict

Vendor (VendorID, Name, State, ZipID, Length, Discount, DiscountPeriod)
Foreign Key ZipID references Zip
Not Null
On Delete Restrict
VendSupInv (VendSupInvID, VendorID, SupplyInventoryID, OtyOnHand)
Foreign Key VendorID references Vendor
Not Null
On Delete Restrict

Foreign Key SupplyInventoryID references SupplyInventory
Not Null
On Delete Restrict

Product (ProductID, OffID, ResortID, WeekPeakPrice, NonWeekPeakPrice, WeekPrice, NonWeekPrice, DayUsePrice, OvernightPrice, QtyOnHand)

Foreign Key OffID references Offering
Not Null
On Delete Restrict

Foreign Key ResortID references Resort
Not Null
On Delete Restrict

Foreign Key CleanEmpID references Employee
Not Null
On Delete Restrict

Offering (OffID, OffTypeID, OffName, IsPullThrough, IsBackIn, IsGlamp, IsExclusive, Notes

ROLine (ROLID, ProductID, ReservationID, CheckIn, CheckOut, Price, Qty, ResortID)

Foreign Key ProductID references Product
Not Null
On Delete Restrict

Foreign Key ReservationID references Reservation
Not Null
On Delete Restrict

Foreign Key ResortID references Resort
Not Null
On Delete Restrict

Reservation (ReservationID, CustID, EmployeeID, Price, AmountPaid, DatePaid)

Foreign Key CustID references Customer
Not Null
On Delete Restrict

Foreign Key EmployeeID references Employee
Not Null
On Delete Restrict

EmployeeType (EmpTypeID, Description)

PaymentOut (PaymentOutID, InspectEmpID, MakeEmpID, InvoiceNumber, Amount, DatePaid, VendorID)

Foreign Key InspectEmpID references Employee
Null Allowed
On Delete Set Null

Foreign Key MakeEmpID references Employee
Not Null
On Delete Restrict

Foreign Key VendorID references Vendor
Not Null
On Delete Restrict

Inspection (InspectionID, ResortID, ProductID, InspectionDate, InspectionDetails, IsCleaned, EmployeeID)

Foreign Key ResortID references Resort
Not Null
On Delete Restrict

Foreign Key ProductID references Product
Not Null
On Delete Restrict

Foreign Key EmployeeID references Employee
Not Null
On Delete Restrict

OffType (OffTypeID, Description)

Differences Between ERD and Normalized Relations

In our ERD there are instances of derived attributes in Reservation, Resort, SupplyInventory, and Product. There are also several multi-variable attributes that had to be broken down to their base components. We also had Boolean attributes in Customer and Offering. Breaking these attributes apart helps us to avoid the risk of inaccurate data, data

redundancy, loss of data, and loss of valuable time. The addition of constraints to the normalized relations gives us the ability to update or delete data as needed without the risk of losing data or wasting time.

The normalization of data helps us move forward to understand the things we need to change to make our database more efficient. We are finally able to see how the information will fit together in a meaningful way as far as the SQL database is concerned. This will assist us in the implementation of the database when we are building the relationships between the entities that make the database work. Through normalization we are able to remove attributes that are unneeded or can be derived from other attributes in the table. Those unnecessary attributes tend to clog the database and cause it to work less efficiently. This process helps us move into the implementation phase by allowing us to see which attributes needed to be pruned or adjusted in order to build reliable relationships in the database.

Referential Integrity Constraint Explanation

A referential integrity constraint is a specific set of rules for specific tables in the database. These rules tell the tables in the database how to react to different circumstances. For instance, if you were to delete one piece of information that is connected to other tables within the database, the rules tell the tables that are connected what to do with the information they are borrowing from the table you deleted. The referential integrity constraint also defines where the borrowed information came from in the first place.

These constraints are very important to the database process because they protect the integrity of the information by keeping vital pieces of information safe from being deleted or changed arbitrarily. In other words, referential integrity constraints keep the database accurate and improve the ease of use. You will not have to worry that if you delete one piece of information, the whole database falls apart. They also protect against mistakes by employees entering inaccurate information, or deleting important tables by accident. Ease of use is improved because the constraints tell the database which tables to update, and how they are supposed to update. This saves you the time you would normally spend going through and changing every table. The referential integrity constraints are a safeguard against mistakes and wasted time.

MILESTONE 3: Physical Design and Implementation

Physical Design and Denormalization

Data Dictionary

PaymentTerms					
FieldName	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
PaymentTermsID	PK	int	Yes	1	Payment terms to company customers
Length		varchar(50)	No		Length of time before full payment is due in days
Discout		decimal(1,2)	No		Percentage discount for early payment
DiscoutPeriod		varchar(50)	No		Length of discount period in days
Customer					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
CustID	PK	int	Yes	1	
FName		varchar(50)	Yes		
MInitial		char(1)	No		One letter for middle initial(eg W)
LName		varchar(50)	Yes		
Street		varchar(50)	Yes		Street address (eg 2500 Asp Avenue)
Email		varchar(100)	Yes		Valid Email Address (eg Tummy@ou.edu)
Phone		varchar(16)	Yes		Valid Phone Number(eg 405-233-7575)
IsWalkIn		bit	Yes		Walk-in reservation by customer
PaymentTermsID		int	No	1	Payment terms to select customers
ZipID	FK references Zip	int	Yes	1	
Country					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
CountryID	PK	int	Yes	1	
CountryName		varchar(50)	Yes		USA or Canada

Zip					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
ZipID	PK	int	Yes	1	
ZipCode		varchar(6)	Yes		5 digits Zip code(eg 05727)
CityID	FK references City	int	Yes		
City					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
CityID	PK	int	Yes	1	
CityName		varchar(100)	Yes		City Name (eg Norman)
StateID	FK references State	int	Yes	1	
State					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
StateID	PK	int	Yes	1	
StateInitials		char(2)	No		2 letter state abreviation (eg OK)
StateName		varchar(50)	No		Full state Name (eg Oklahoma)
ProvinceInitials		char(2)	No		2 letter province abreviation (eg AL)
ProvinceName		varchar(50)	No		Full Province name (eg Alberta)
Performance					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
PerformanceID	PK	int	Yes	1	
CustID	FK references Customer	int	Yes	1	
ResortID	FK references Resort	int	Yes	1	
Review		varchar(100)	No		Customer short review (eg. Good Customer Service)

Supply					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
SupplyID	PK	int	Yes	1	
Description		varchar(100)	Yes		Name of supply or service (eg Towels)
Reservation					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
ReservationID	PK	int	Yes	1	
CustID	FK references Customer	int	Yes	1	
EmployeeID	FK references Employee	int	Yes	1	
Price		smallmoney	Yes		Full price in dollars
AmountPaid		smallmoney	No		Advance payment amount in dollars
DatePaid		date	No		Date advance payment was made
ROLine					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
ROLID	PK	int	Yes	1	
ProductID	FK references Product	int	Yes	1	
ReservationID	FK references Reservation	int	Yes	1	
CheckIn		datetime	No		Date guest checked in (eg 2017-12-26)
CheckOut		datetime	No		Date guest checked out (eg 2017-12-30)
Price		smallmoney	Yes		Price of stay in dollars
Qty		smallint	Yes		Number of units reserved per reservation
ResortID	FK references Resort	int	Yes	1	

Resort					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
ResortID	PK	int	Yes	1	
ResortName		varchar(100)	Yes		The full name of the resort (eg Minor's Hollow)
Street		varchar(50)	Yes		Street address for resort (eg 2500 Asp Avenue)
FirstShoulderStart		date	No		Date first shoulder season begins(eg 2017-12-23)
FirstShoulderEnd		date	No		Date first shoulder season ends (eg 2018-01-03)
SecondShoulderStart		date	No		Date second shoulder season begins (eg 2018-02-01)
SecondShoulderEnd		date	No		Date second shoulder season ends (eg 2018-03-19)
OpenDate		date	No		Date resort opens for the season (eg 2017-01-01)
ClosedDate		date	No		Date resort closes for the season (eg 2017-12-31)
ZipID	FK references Zip	int	Yes	1	
ManagerEmpID	FK references Employee	int	Yes	1	
CountryID	FK references Country	int	Yes	1	
Acre		int	no		The amount of acreage for each resort (eg 25)
SupplyInventory					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
SupplyInventoryID	PK	int	Yes	1	
ResortID	FK references Resort	int	Yes	1	
SupplyID	FK references Supply	int	Yes	1	
QtyOnHand		smallint	No		Current quantity of specific supply on Hand (eg 45)
QtyCommitted		smallint	No		Quantity of specific supply on hand committed to use (eg 23)
ReorderPoint		smallint	No		Amount of supply that signals a reorder when reached (eg20)

Employee					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
EmployeeID	PK	int	Yes	1	
EmpTypeID	FK references EmployeeType	int	Yes	1	
FName		varchar(50)	Yes		First name of employee (eg Frank)
MInitial		char(1)	No		First letter of employee middle name (eg W)
LName		varchar(50)	Yes		Last name of employee (eg Smith)
Phone		varchar(16)	No		Phone number for employee (eg 405-333-7575)
EmployeeType					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
EmpTypeID	PK	int	Yes	1	
Description		varchar(100)	Yes		Employee title (eg Manager)
Product					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
ProductID	PK	int	Yes	1	
OffID	FK references Offering	int	Yes	1	
ResortID	FK references Resort	int	Yes	1	
WeekPeakPrice		smallmoney	No		Price in dollars for a week during peak season (eg $465.08)
NonWeekPeakPrice		smallmoney	No		Price in dollars for weekend stays during peak season (eg $204.50)
WeekPrice		smallmoney	No		Price in dollars for a week during non peak times (eg $437.02)
NonWeekPrice		smallmoney	No		Price in dollars for a weekend stay during non peak times (eg $627.08)
DayUsePrice		smallmoney	No		Price in dollars for use of a service for 1 day (eg $88.45)
OvernightPrice		smallmoney	No		Price in dollars for 1 night stay (eg $274.55)
QtyOnHand		smallint	No		Quantity of total units on hand (eg 74)

VendSupInv					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
VendSupInvID	PK	int	Yes	1	
VendorID	FK references Vendor	int	Yes	1	
SupplyInventoryID	FK references SupplyInventory	int	Yes	1	
Qty		smallint	Yes		Quantity ordered (eg 18)
Price		smallmoney	Yes		Total price in dollars of items ordered (eg $300.00)
Offering					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
OffID	PK	int	Yes	1	
OffTypeID	FK references OfferingType	int	Yes	1	
OffName		varchar(50)	Yes		Name of specific offerings and services
IsPullThrough		bit	No		The site is pull through
IsBackIn		bit	No		The site is back in
IsGlamp		bit	No		The site is a glamp site
IsExclusive		bit	No		The offering is excusive to a specific resort
Notes		varchar(100)	No		Short note regarding the service or offering

Vendor					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
VendorID	PK	int	Yes	1	
Name		varchar(50)	Yes		Full vendor name (eg Staples)
Street		varchar(100)	Yes		Street address of vendor (eg 2525 Concord Drive)
ZipID	FK references Zip	int	Yes	1	
Length		varchar(10)	No		Length of payment terms for full payment (eg 30 days)
Discount		decimal (1,2)	No		Discount percentage for early payment (eg 0.05))
Discount Period		varchar(10)	No		Length of discount period (eg 10 days)
PurchaseOrder					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
POID	PK	int	Yes	1	
InspectsEmpID	FK references Employee	int	No	1	Employee ID of employee that approves order
MakesEmpID	FK references Employee	int	No	1	Employee ID of employee that makes initial order
PODate		date	Yes		Date order is made (eg 2017-18-01)
POTotal		smallmoney	Yes		Total cost of order (eg $2530)
DeliveryIn					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
DeliveryInID	PK	int	Yes	1	
RecievesEmpID	FK references Employee	int	No	1	Employee ID of employee that recieves delivery
InspectEmpID	FK references Employee	int	No	1	Employee ID of employee that inspects the delivery
PaymentOutID	FK referencces PaymentOut	int	Yes	1	
DeliveryDate		date	No		Date delivery was accepted (eg 2017-01-01)
ResortID	FK references Resort	int	Yes		Resort that accepted delivery

PaymentOut					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
PaymentOutID	PK	int	Yes	1	
InspectsEmpID	FK references Employee	int	No	1	Employee that approves payment
MakesEmpID	FK references Employee	int	Yes	1	Employee that makes payment
InvoiceNumber		int	Yes		Invoice number for order history (eg 200)
Amount		smallmoney	Yes		Total amount owed in dollars (eg $1319)
DatePaid		date	No		Date payment was made (eg 2016-11-06)
VendorID	FK references Vendor	int	Yes	1	Vendor that made delivery
POLine					
Field	**Key**	**Data Type**	**Reqd**	**Default Value**	**Description**
POLID	PK	int	Yes	1	
POID	FK references PurchaseOrder	int	Yes	1	
DeliveryInID	FK references DeliveryIn	int	Yes	1	
VendSupInvID	FK references VendSupInv	int	Yes	1	
Qty		smallint	Yes	1	Quantity of supplies ordered (eg 65)
Price		smallmoney	Yes		Total price in dollars (eg $500)

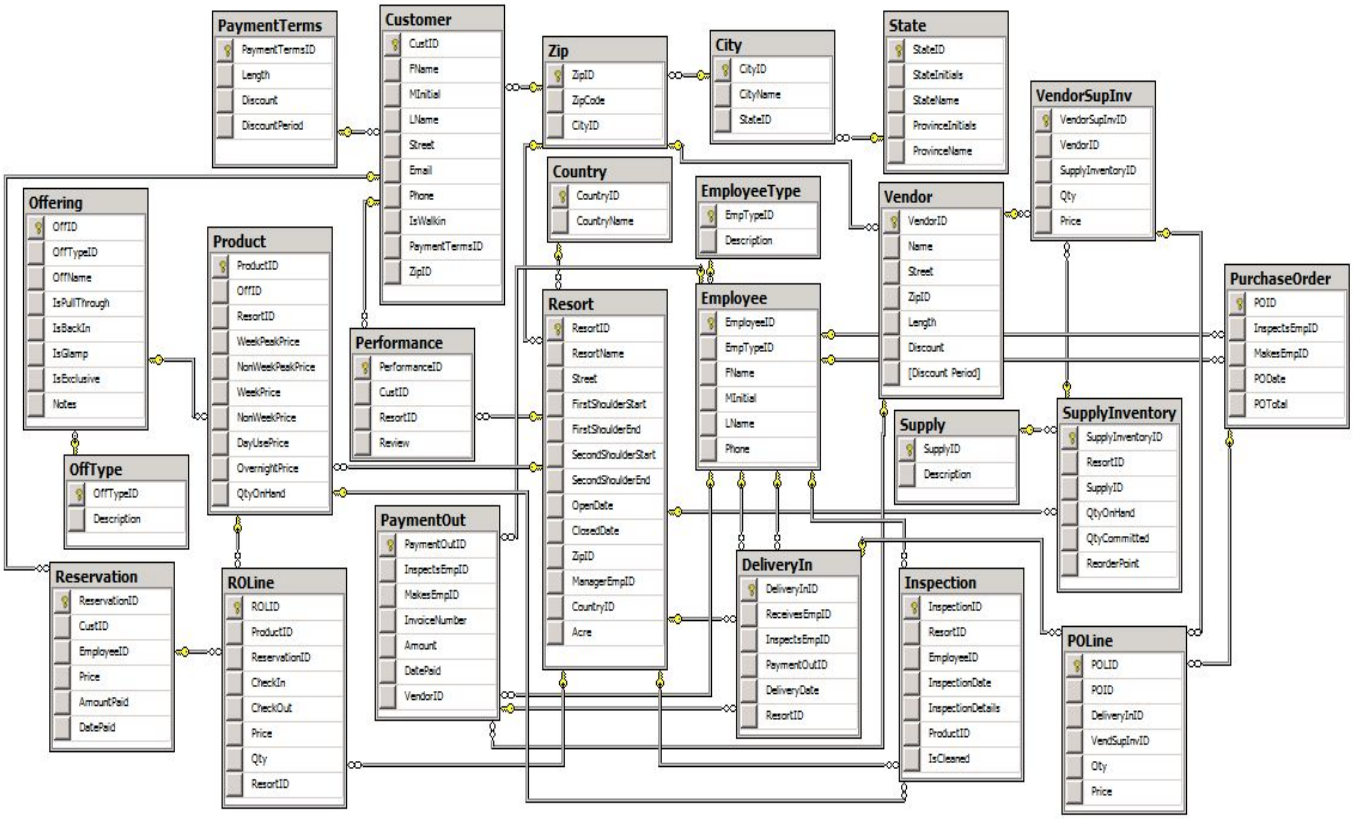
Inspection					
Field	**Key**	**DataType**	**Reqd**	**Default Value**	**Description**
InspectionID	PK	int	Yes	1	
ResortID	FK	int	Yes	1	
EmployeeID	FK	int	Yes	1	
InspectDate		date	No		Date product was last inspected and cleaned
InspectionDetails		Varchar(50)	No		Product meets expectations or does not meet expectations
ProductID	FK	int	Yes	1	
IsCleaned		bit	Yes		The site has been cleaned if it is 1 and if not it is 0
OfferingType					
Field	**Key**	**DataType**	**Reqd**	**Default Value**	**Description**
OffTypeID	PK	int	Yes	1	
Description		varchar	Yes		Ex: amenity, 30-amp unit

Denormalization

None of our relations needed to be denormalized. We did not find it necessary to denormalize the relations because it would introduce data integrity issues by accepting duplicate data. The database itself is pretty small and with the fast technology available efficiency of the database will not be an issue.

Implementation

Diagram of Implemented Database



Implementation Challenges

When importing that data and working with SQL we found a lot of weaknesses and strengths with it. A weakness of SQL that challenged us was how it does not allow you to implement a composite primary key. This was hard for us to go about because a lot of our associative entities needed a composite primary key, but because of this weakness we had to use a surrogate primary key. Another weakness we found with SQL was that there is a time lag that affects us when somebody edits something on the mac it won't reflect on the windows computer for a while later. It also runs slower on the Mac computers, two of our group members were

using macs and it was frustrating when it would freeze every time we would try to change something. A strength we found when working with SQL was its ability to tell us in our queries where there was an error. On some databases it will just say error or not work and not explain where the problem is, but with SQL it would tell you what line the error is in and what it is around. Some challenges we had when trying to import data we thought we were allowed to create the tables in SQL with all of the field names and then import data from excel tables but we were not able to do that. This forced us to have to delete all of the tables and redo it in excel then import them from that.

SQL

Required Queries

1. Which employee serviced (inspected, cleaned) each lodging unit at each resort and when did it most recently happen?

```
SELECT O.OffName, I.InspectionDate, IsCleaned, E.FName, E.LName
FROM Product P,Employee E, Inspection I, Offering O
WHERE I.ProductID = P.ProductID AND I.EmployeeID= E.EmployeeID AND
P.OffID=O.OffID
ORDER BY 2 DESC;
```

	OffName	InspectionDate	IsCleaned	FName	LName
1		2018-04-09	0	Sara	Pacheco
2		2017-10-06	0	Yoshio	Keller
3		2017-05-11	0	Dara	Ayala
4		2017-05-01	1	Chadwick	Maddox
5		2017-05-01	0	Elliott	Gregory
6		2016-12-16	0	Elizabeth	Parsons
7		2016-11-09	1	Berk	Atkins
8	Water park	2016-11-02	1	Burke	Campos
9		2016-07-11	1	Astra	Robbins
10		2016-06-10	1	Hilda	Mckinney

2. For the Marmotvale property, which lodging sites are available for the night of July 24, 2017. For each of these, is the site pull-through or back-in, what power connection is available, and does it have a river view?

SELECT R.ResortName, O.IsPullThrough, O.IsBackIn, T.Description, O.Notes

FROM Resort R, ROLine L, Product P, Offering O, OfferingType T

WHERE R.ResortID = L.ResortID AND L.ProductID = P.ProductID AND P.OffID = O.OffID AND O.OffTypeID = T.OffTypeID AND R.ResortID = 15;

	ResortName	IsPullThrough	IsBackIn	Description	Notes
1	Marmotvale	0	1	glamp-unit	River View

3. For each purchase order that was placed in June 2017, show each line, including the PO number, the name of the resort for which the order was intended, the name of the vendor, the quantity ordered, the item price, and the line total.

SELECT PODate, O.POID,ResortName, V.Qty, L.Price, POTotal

FROM PurchaseOrder O, POLine L, DeliveryIn D, Resort R, VendorSupInv V

WHERE O.POID = L.POID AND L.DeliveryInID = D.DeliveryInID AND R.ResortID = D.ResortID AND V.VendorSupInvID = L.VendSupInvID AND PODate LIKE '%2017-06%';

	PODate	POID	ResortName	Qty	Price	POTotal
1	2017-06-22	3	Skyline Mine Resort	70	320.00	899.34
2	2017-06-22	6	Grifter's Ridge	42	360.00	2674.33
3	2017-06-22	9	Beaverside Station	52	200.00	789.00

4. For the three most recently paid invoices, list the (vendor) invoice number, the name of the vendor who sent the invoice, the total amount paid, and the name of the employee who approved payment of the invoice.

SELECT TOP 3 P.InvoiceNumber, V.Name as [VendorName], E.FName, E.LName, P.InspectsEmpID, DatePaid

FROM PaymentOut P, Vendor V, Employee E

WHERE P.VendorID = V.VendorID AND E.EmployeeID=P.InspectsEmpID

Order By DatePaid DESC;

	InvoiceNumber	VendorName	FName	LName	InspectsEmpID	DatePaid
1	183	Amet Corporation	Ingrid	Ipswitch	9	2018-04-09
2	193	Quisque Inc.	Gale	Gable	7	2017-10-08
3	153	Posuere Cubilia Associates	Debra	Dodson	4	2017-05-13

5. Suppose a customer just called and wants to book a tent-camping site at a resort in Colorado for a week, starting October 13, 2017. List all the relevant options, including the name of the resort, the identity of the site(s) available at that resort, and the price for each of those sites

SELECT R.ResortName, T.Description AS "Camp Site", P.WeekPrice, P.NonWeekPrice
FROM Reservation N, ROLine L, Resort R, Zip Z, City C, State S, Product P, Offering O, OfferingType T

WHERE N.ReservationID = L.ReservationID AND L.ProductID = P.ProductID AND P.OffID = O.OffID AND O.OffTypeID = T.OffTypeID AND L.ResortID = R.ResortID AND R.ZipID = Z.ZipID AND Z.CityID = C.CityID AND C.StateID = S.StateID AND S.StateName LIKE 'Colorado'AND T.OffTypeID = 2

	ResortName	Camp-Site type	WeekPrice	NonWeekPrice
1	Copperhammer Hills	non-electric	296.08	289.21

6. Provide a report of all those vendors that provide pool-cleaning services for resorts in Utah. Also provide a query that allows the company to input (via the interface, i.e., not part of your "query") the specific service as well as the state.

SELECT V.Name, P.Description, E.StateName

FROM Vendor V, SupplyInventory S,Supply P, VendorSupInv I, Resort R, Zip Z, City C, State E

WHERE V.VendorID = I.VendorID AND S.SupplyID = P.SupplyID AND I.SupplyInventoryID = S.SupplyInventoryID AND S.ResortID = R.ResortID AND V.ZipID = Z.ZipID AND Z.CityID = C.CityID AND C.StateID = E.StateID AND E.StateID = 4 AND S.SupplyID = 11;

/*Enter the value you would like to use in the spots marked /**/. */

SELECT V.Name, P.Description, E.StateName

FROM Vendor V, SupplyInventory S,Supply P, VendorSupInv I, Resort R, Zip Z, City C, State E

WHERE V.VendorID = I.VendorID AND S.SupplyID = P.SupplyID AND I.SupplyInventoryID = S.SupplyInventoryID AND S.ResortID = R.ResortID AND V.ZipID = Z.ZipID AND Z.CityID = C.CityID AND C.StateID = E.StateID AND E.StateID = /**/ AND S.SupplyID = /**/;

	Name	Description	StateName
1	Imperdiet Erat PC	Pool Clean	Utah

7. What supplies have been delivered to the Camp Carbon resort in 2017?

SELECT R.ResortName,S.SupplyID, S.Description,I.QtyOnHand

FROM DeliveryIn D, POLine P, SupplyInventory I, Resort R, Supply S

WHERE P.DeliveryInID = D.DeliveryInID AND D.ResortID = R.ResortID AND R.ResortID = I.ResortID AND I.SupplyID = S.SupplyID AND I.ResortID = 6 AND D.DeliveryDate LIKE '%2017%';

Results Messages

	ResortName	SupplyID	Description	QtyOnHand
1	Camp Carbon	1	Stationary	83

8. Which customers have placed multiple bookings within the last 12 months? At which resorts have these bookings been placed?

SELECT C.FName, C.LName, R.ResortName

FROM Reservation N, ROLine O, Customer C, Resort R

WHERE N.ReservationID = O.ReservationID AND N.CustID = C.CustID AND O.ResortID = R.ResortID AND N.CustID IN (SELECT COUNT(CustID) FROM Reservation GROUP BY CustID HAVING COUNT(CustID) > 1)

ORDER BY C.FName;

Results Messages

	FName	LName	ResortName
1	David	Pond	Ramshorn Ranch
2	David	Pond	Coldwater RV Park
3	David	Pond	Miner's Hollow
4	Joe	Oswald	Copperhammer Hills
5	Joe	Oswald	Emperor's Roost
6	Joe	Oswald	Great Western Resort

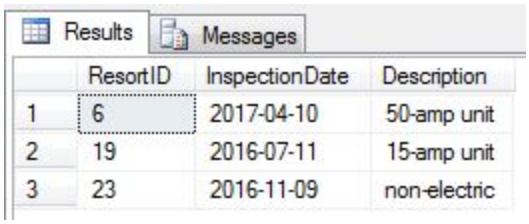
9. Which five resorts have generated the greatest revenue over the last 12 months? Which resorts have generated the least revenue over the last 12 months? For all resorts, show how much revenue the resort has generated per acre; report these in order from the highest revenue per acre to the lowest.

```
SELECT T.ResortName, SUM (R.AmountPaid)/T.Acre AS [Total earned per acre]
FROM Reservation R, ROLine L, Resort T
WHERE R.ReservationID = L.ReservationID AND T.ResortID = L.ResortID AND
R.Datepaid > '2016-05-01'
GROUP BY T.ResortName, T.Acre
ORDER BY 2 desc;
```

	ResortName	Total earned per acre
1	Marmotvale	35.00
2	Miner's Hollow	10.00
3	Copperhammer Hills	9.5652
4	Coldwater RV Park	9.0909
5	Great Western Resort	8.9285
6	Sanford Springs Ranch	8.1395
7	Six Falls Resort	6.4516
8	Ramshorn Ranch	5.2631
9	Emperor's Roost	5.00
10	Lago Urso Camping Resort	NULL

10. Which resort has the most lodging sites that have gone more than a week since they were last cleaned/inspected?

```
SELECT R.ResortID, I.InspectionDate, T.Description
FROM Resort R, Inspection I, Product P, Offering O, OfferingType T
WHERE R.ResortID = I.ResortID AND I.ProductID = P.ProductID AND P.OffID =
O.OffID AND O.OffTypeID = T.OffTypeID AND I.InspectionDate < '2017-04-24';
```



Results | Messages

	ResortID	InspectionDate	Description
1	6	2017-04-10	50-amp unit
2	19	2016-07-11	15-amp unit
3	23	2016-11-09	non-electric

11. For each resort, show the overall vacancy rate (1 - (total lodging-days booked divided by total lodging-days in the month)) for the month of September 2017. Order results from the highest vacancy rate to the lowest.

```
SELECT DISTINCT R.ResortName ,(P.QtyOnHand - L.Qty) AS [Vacancy]
FROM Resort R, ROLine L, Product P
WHERE R.ResortID = P.ResortID AND P.ProductID = L.ProductID
ORDER BY 2 DESC;
```

	ResortName	Vacancy
1	Heart-of-the-Stone	191
2	Exit Canyon Ranch Resort	184
3	Sanford Springs Ranch	183
4	Ed's RV and Camper Park	162
5	Lincoln Creek Luxury Resort	119
6	Ramshorn Ranch	112
7	Ramshorn Ranch	109
8	Coldwater RV Park	83
9	Exit Canyon Ranch Resort	73

12. For each resort, how many customers to which the company extends payment terms have made purchases in the last 12 months? In your report, also include any resorts for which no customers that receive payment terms have made purchases.

```
SELECT COUNT(N.CustID) AS [Extended Payment Terms], R.ResortName, C.FName
AS [Customer First], C.LName AS [Customer Last]
FROM Resort R, ROLine L, Reservation N LEFT JOIN Customer C ON N.CustID =
C.CustID
WHERE R.ResortID = L.ResortID AND L.ReservationID = N.ReservationID AND
N.CustID = C.CustID AND N.DatePaid > '2016-05-01' AND C.PaymentTermsID
BETWEEN 1 AND 3
GROUP BY R.ResortName, N.CustID, C.FName, C.LName;
```

Additional Queries

13. Which employee inspected the delivery in on the date of 2017-09-09?
This would be helpful for the customer in case there was a problem with the delivery on that specific date. With this information they would be able to figure out who inspected a delivery on a specific date.

```
SELECT E.FName, E.MInitial, E.LName, D.DeliveryInID, DeliveryDate
FROM Employee E, DeliveryIn D
WHERE E.EmployeeID = D.InspectsEmpID  AND DeliveryDate = '2017-09-09'
```

	FName	MInitial	LName	DeliveryInID	DeliveryDate
1	Ingrid	I	Ipswitch	8	2017-09-09

14. Which employee made a payment out which cost more than $3500, and which resort do they work for? What invoice number was it?
This query would be helpful if the company budgets a maximum of $3500 for the payment out per invoice for each resort and would like to investigate which employee exceeded the budget, for what invoice, and at what resort so they can find reasons why.

```
SELECT E.EmployeeID, E.LName, E.FName, R.ResortName, P.InvoiceNumber
FROM Employee E, PaymentOut P, Resort R
WHERE R.ResortID = E.EmployeeID AND E.EmployeeID = P.MakesEmpID AND
Amount > 3500;
```

Results | Messages

	EmployeeID	LName	FName	ResortName	InvoiceNumber
1	4	Dodson	Debra	Lakehollow Park	123

15. Which resort has a day use price below $50.00?
This query will be useful if the management wants to know which resorts have a day use price below $50.00, in case they want to increase prices.

```
SELECT R.ResortName, P.DayUsePrice
FROM Resort R, Product P
WHERE R.ResortID = P.ResortID AND P.DayUsePrice < 50;
```

Results | Messages

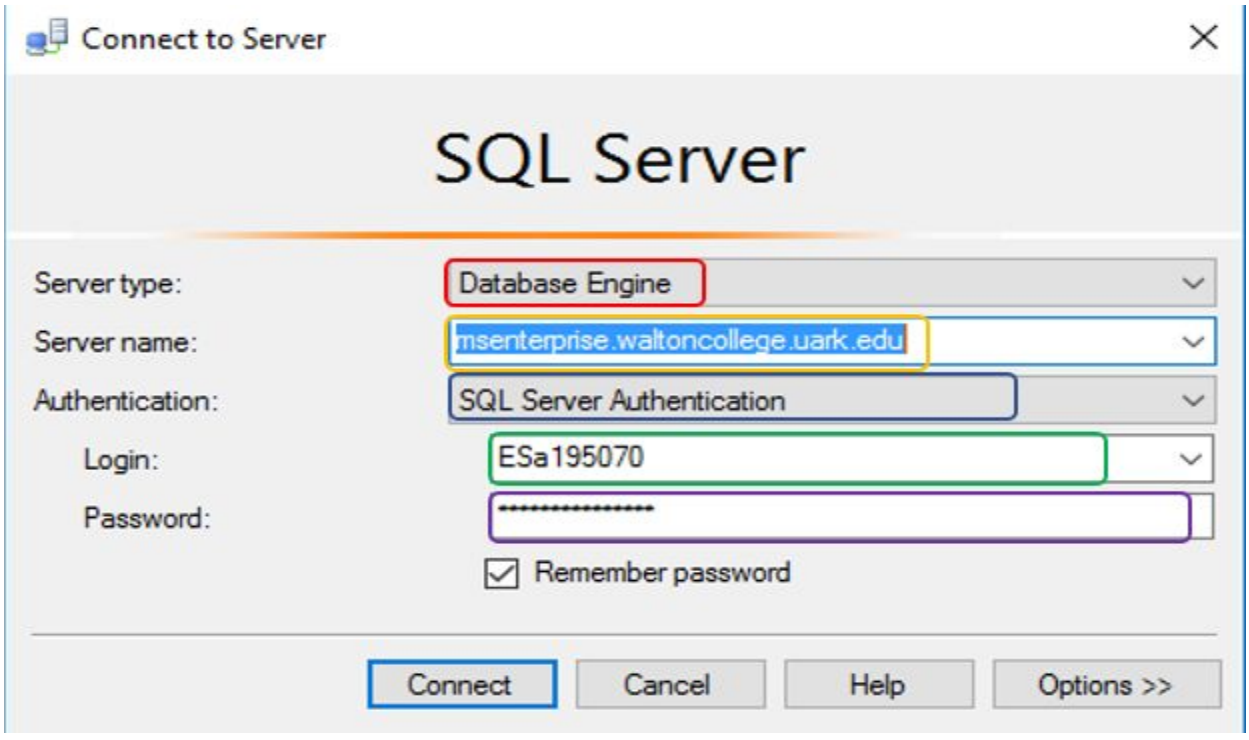
	ResortName	DayUsePrice
1	Heart-of-the-Stone	47.57
2	Sanford Springs Ranch	41.81

USER MANUAL

How to Open the Database

Step 1: Open the Microsoft SQL Server Management Studio on your Windows computer. (If you are using a Mac, open SQL Server Management Studio on the remote desktop).

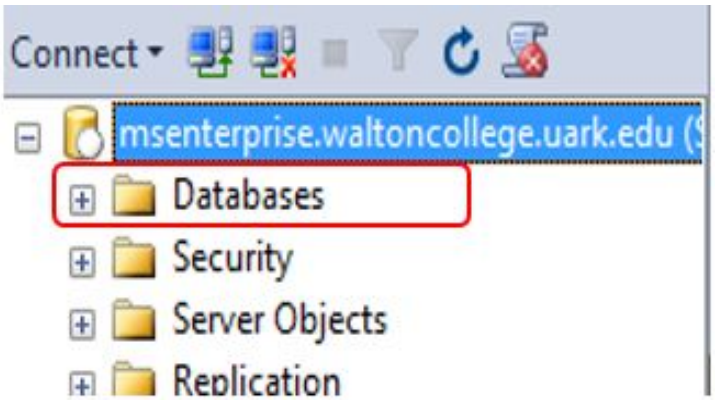
Step 2: Enter your login credentials.



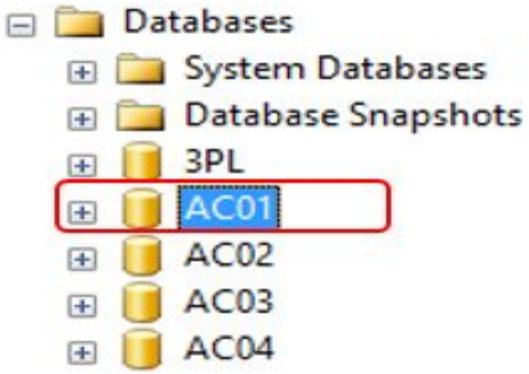
(a) Server Type: Choose Database Engine

(b) Server Name: msenterprise.waltoncollege.uark.edu
(c) Authentication: Choose SQL Server Authentication
(d) Login: Enter the login name assigned to you by the resort manager
(e) Password: Enter the password assigned to you by the resort manager

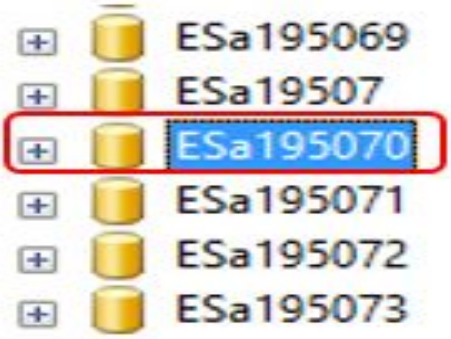
Step 3: Expand the Databases folder



Step 4: Click on any one of the databases, type Esa and look for the Esa login you were assigned by the resort manager.

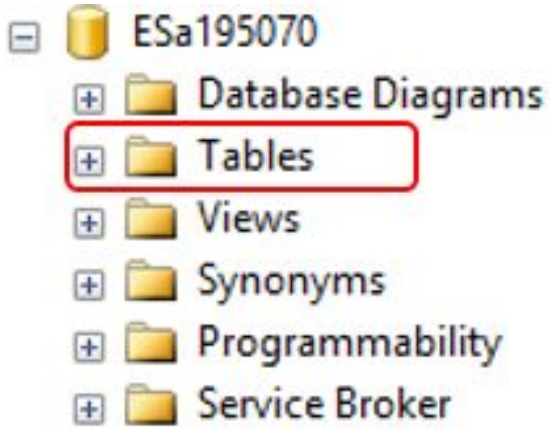


Step 5: Expand the database matching your Esa login, example Esa195070



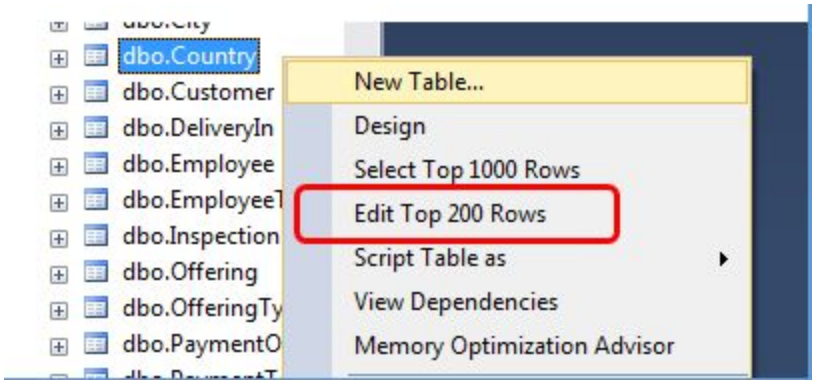
How to Enter Data

Step 1: Once you have opened the database matching your Esa login, expand the Tables subfolder

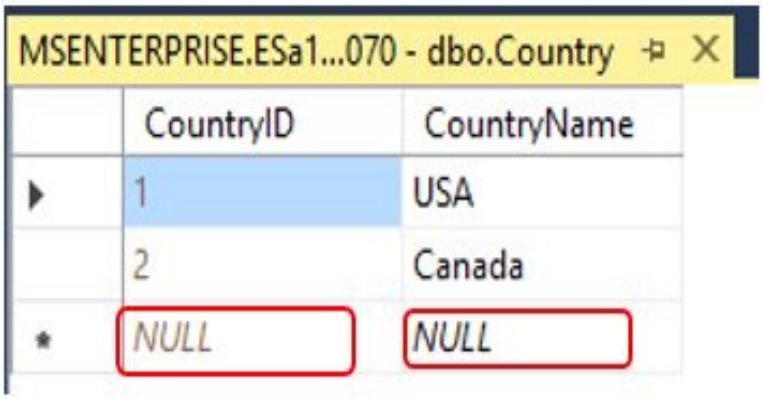


Step 2: Select the desired table, example dbo.Country

Step 3: Right click and select the Edit Top 200 Rows option

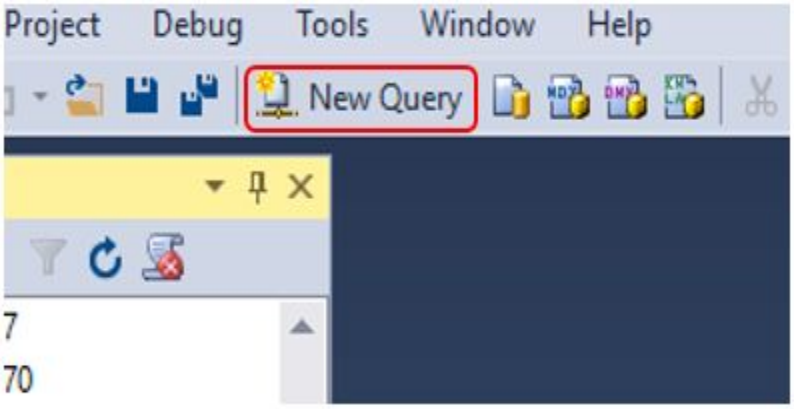


Step 4: You can manually enter new data, if needed, in the Null spaces

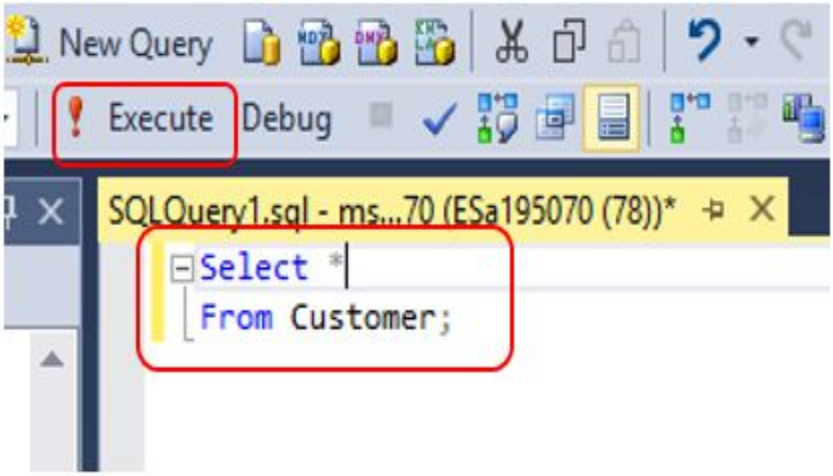


How to Run Queries

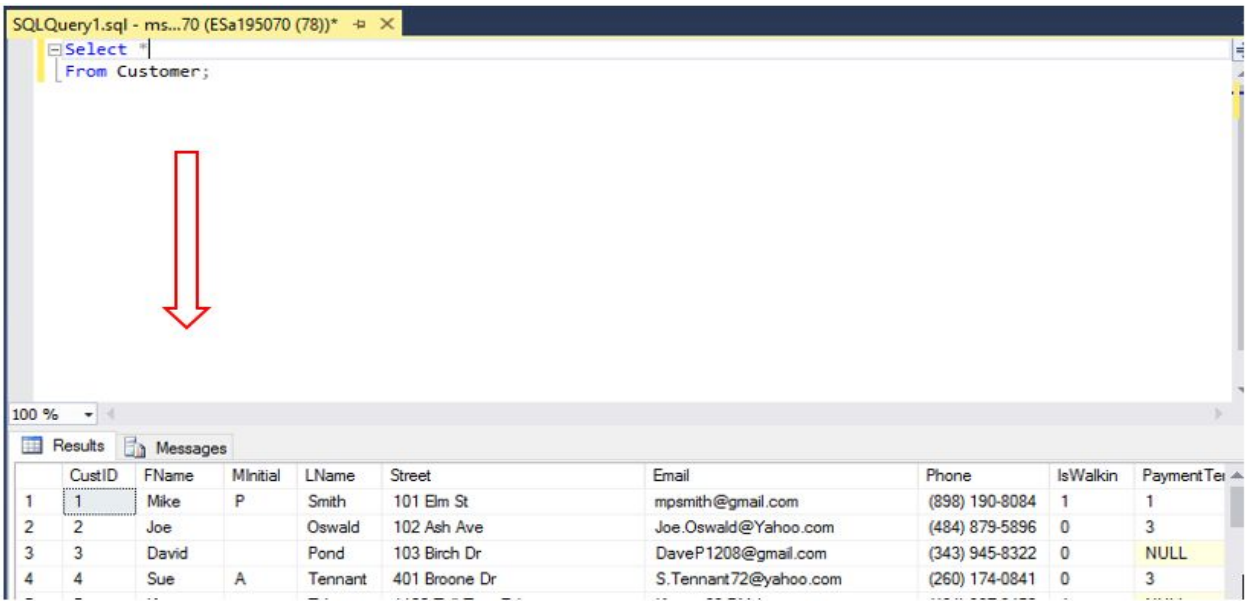
Step 1: Click the New Query button on the toolbar



Step 2: Write a Select statement that returns the information required for your report and click the execute button on the toolbar



Step 3: Look for your results in the window below the select statement



	CustID	FName	MInitial	LName	Street	Email	Phone	IsWalkin	PaymentTer
1	1	Mike	P	Smith	101 Elm St	mpsmith@gmail.com	(898) 190-8084	1	1
2	2	Joe		Oswald	102 Ash Ave	Joe.Oswald@Yahoo.com	(484) 879-5896	0	3
3	3	David		Pond	103 Birch Dr	DaveP1208@gmail.com	(343) 945-8322	0	NULL
4	4	Sue	A	Tennant	401 Broone Dr	S.Tennant72@yahoo.com	(260) 174-0841	0	3

LESSONS LEARNED

Individual Responses

Molungoa Ramataboe

From conceptual design, I learned that it is important to understand the client requests and ask as many questions as possible for their requests before going any further. I also learned that as a database builder, it is important to realize that the client may not know everything they definitely need but it's my works as database builder to help him or her. While building the ERDs, I learned that it is important to understand the potential queries that user may need to process while building the ERD. I also learned that the database that's not build properly can hurt the business, as the business may not be able to run desired queries and also data may be inaccurate which may be hurt the decision making of the company managers.

Taylor Campbell

The first day of class was very overwhelming, as I had no idea what to expect from this class. Throughout this semester I have learned a lot of things, not just about this class but also about working with a group. Before taking this class I did not know how to make a database or even how to draw an ERD. I have learned that building a database can be very tedious and time consuming but it is beneficial for the people using it. I learned from the project that it is important to understand all what exactly the customer wants, it there was a next time I would've read the portfolio several more times before starting the ERD, rather than having to redo the whole thing multiple times. After doing the project I learned working in groups can be difficult at times. It’s hard to meet with everybody when we all had different schedules and other class responsibilities.

Rita Mihalek

Through the completion of this project, I have learned a great deal about both database development and how to work effectively within a group structure. Before this project my only experience with SQL was a brief tutorial in Microsoft Access. After the process of building a database from the ground up, I now have a much better appreciation of the work that goes in behind the scenes of an effective, reliable database. I also now have a good idea of how much there is yet to learn about database development and management. Working in a group structure is always a challenge, because it is a different experience every time. This group worked well together, but our schedules were so different and finding times to work together in

person presented a challenge. However, we were fortunate to have technology at our disposal such as Google Docs and GroupMe, which made it easier to communicate effectively.

John McQueen

This project has allowed me to learn many things related to a business database and has provided me with practice on how to take a project from design step by step to the final implementation. Before this class I had very little experience in SQL and had never worked with an ERD. Through the design part of the class I was able to learn how to create ERDs and understand how they work as a map for the development of the database. I learned how to take information from a client about a potential database, and how to turn that into assumptions we would need to make about our own database. By having to query our own database, it has helped me to understand the steps needed to writing better queries. Working in groups has also shown me some of the difficulties around trying to plan meetings and separate work.

Group Response

We learned many important lessons over the course of this project. The most important lesson learned was the importance of communication. When we met with the customer, for instance, we had to learn how to translate database jargon into everyday terms that someone who had no experience in database structure would understand. Not only did we have to understand the terms well enough to use them effectively, we also had to communicate them effectively. How we communicated with each other during the development process was extremely important. There was a learning curve at the beginning as we learned each other's strengths and weaknesses. The more we worked together, the easier it was.

We also had to have a working knowledge of how the business functioned. In order to build a working database that would function effectively for the customer we had to not only know that information provided in the case in and out, we had to understand it well enough to make assumptions about how the business would run in the future. This gave us an education in how database management and business management interact with one another.

Group Contract

Team Name: Fantastic Tigers Logo:

Team Motto: _____

Team Members

Name	Email	Phone	Strengths	Availability to Meet
Rita Mihalek	rkmihalek@ou.edu	316-708-2035	organized Quick Learner	Friday all Day Evenings NO Thurs Saturday All Day
John McQueen	John.C.McQueen-1@OU.edu	918-530-8747	Quick learner Programming Design	Almost Always except Sunday night
MOLUNGOA RAMATABOE "WILLIAM"	Ramataboe@ou.edu	4052488492	Research Reading Connections	Friday all day Evenings Mornings Weekends
Taylor campbell	taylor.rcampbell@yahoo.com			

Unique Capabilities: * Good a working the problem, figuring it out

* Diversity

Team Expectations (for Peer Evaluation):

* Involvement, showing up, participation
* Preparation
* follow through, Honesty
* Respect

Presentation Date Preferences (Rank Order Available Dates):

1) 3-27

2) 4-19