University of Oklahoma

Team Led Zeppelin
Kali Curtis, Drake Detar, Dominick Demas, Jackson Dedrick, Jonathan Domingo
MIS 3353 – Database Management
Dr. Clarke Daugherty
March 24, 2021

Led Zeppelin
Just Do IT

Executive Summary

This project was our interpretation of Sonner Tire's request to adopt a database system for use in their company. They requested that we create a database that keeps track of their sales and expenditure processes better than the one they have been using previously and allows for them to effortlessly search the database for any information they need. We accomplished this by creating an entity relationship diagram (ERD) of the business processes, converting that diagram into a normalized relation that shows all of the relationships between entities, and finally turning that into code that would be input into the database server during the physical implementation.

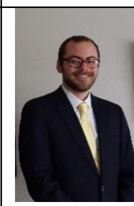
For the creation of the ERD, we took the descriptions of the business, given to us by Mr. and Mrs. Anderson, along with the answers that we received from them during our question-and-answer session and used that information to create the starting diagram that we would begin this project with. Because Sonner Tires does not have any production or manufacturing in their business, we decided to use only the revenue and expenditure cycles in our ERD.

During the logical design phase of our project, we had to convert our final diagram into normalized relations, which is when the entities are in a form that is more efficient for a database. The first major change that takes place during this conversion are changing attributes into a single-value form, like the attribute, Name, being changed into two separate attributes, FirstName and LastName. Another change is removing redundant data. An example of this is creating separate tables for the vehicle information like Make and Model rather than keeping that information within the Vehicle table. Table names are given a T prefix to designate it as a table and attribute names are given a prefix based on which table it is in to provide greater clarification of where the attribute is located.

The physical implementation phase consists of inserting our database into a server and providing it with sample data to test the efficiency and reliability of our database to prepare it for use in Sonner Tire's day-to-day activities. Firstly, we needed to figure out what data types each attribute had to be and we did this by laying out all of our entities and attributes in a data dictionary, which shows whether each attribute is allowed to be empty and if it is a piece of data that references another table. We inserted testing data that we believed was relevant to Sonner Tire and would be similar to the real-life data that they would accumulate. We tested the specific queries that were requested of us and provided the results below. We have also provided specific steps to take in order to access the database.

Contents

Get to Know the Team: Led Zeppelin

Name	Major	Year in School	Internship Experience	Brief Background	Photo
Kali Curtis	MIS	Junior	N/A	I'm from Oklahoma City. I'm an MIS major at OU.	
Drake Detar	MIS	Senior	N/A	I'm from Dallas and currently am a senior here at OU and majoring in MIS.	
Jackson Dedrick	Accounting	Senior	N/A	I am an accounting major studying for an MIS minor.	
Dom Demas	MIS	Junior	N/A	I am from Columbus, Ohio and am majoring in MIS.	
Jonathan Domingo	MIS	Senior	N/A	I am an MIS major at OU and currently a senior.	

Conceptual Design

In the conceptual design phase, an Entity Diagram Relationship (ERD) is created. Before we can begin diagramming an ERD, we need to think about what kind of needs the client has for their database. Accordingly, we met with the client and asked questions about the information that was given to us. We provided details about our meeting with the client, along with the information that we learned in the meeting. This step was essential to creating our ERD for Sonner Tire, so that we could ensure that we understood the clients' needs. We will later explain the purpose of an ERD, the significant assumptions we had to make, and what business cycles we used to create the ERD. Upon reading this section, you will gain a better understanding of the purpose of conceptual design when creating a database.

The Client Meeting

Below, we will list the details of our team meeting with Clarke Daugherty, who met with us on behalf of our client Sonner Tire. The purpose of this meeting was to give our team the opportunity to ask the client any questions we may have regarding this assignment. The meeting occurred over zoom and lasted 20 minutes. Our team members, Kali Curtis, Drake Detar, Dominick Demas, Jackson Dedrick, and Jonathon Domingo, each asked questions about the case provided to us in order to determine what we should include in the ERD. During the interview, Clarke Daugherty specified what kind of data we should include in the database.

- Meeting Time: Wednesday, March 17, at 12:15pm.
- Location: Zoom.
- Interviewers: Kali Curtis, Drake Detar, Dominick Demas, Jackson Dedrick, and Jonathon Domingo.
- Interviewee: Clarke Daugherty

Q&A During the Meeting & Information We Learned

Listed below are the questions that our team asked the client. We specifically had questions about what data Sonner Tire would like to include in the database, what information they may already have, and general questions about their business operations. All of the client's answers are recorded for each question. We used the client's responses to assist us in designing the database.

Q: Do you have a history of each customer that shows what they had purchased, if they purchased it upfront, and if they missed any payment deadlines?
A: We don't have any of the data currently. Essentially when we talk about good or bad customers. We don't need you to put that into a database. We just need their information to be on the data. We just need it when they paid and so on. Was it on time? Not who's good and who's bad.

Q: Do you want to record how long your services take to complete?
A: It's not critical. You can add that. We're not worried about that right now. We're just worried about getting a database and the core of our business.

Q: Do we need to track office supplies?

A: No, we don't need to track office supplies or anything like that.

Q: Do you need to track the purchasing/inventory of raw materials?
A: We actually don't purchase raw materials. The only thing we sell are tires. We don't make tires, we just procure them.

Q: How many tires do you go through in a week?
A: So that would be anecdotal. We don't have that information on hand. We operate small towns and nothing really at all. It varies at product. But we want you to do that to accord.

Q: If you are low on stock or high on stock do you ever order more than once a month or once every other month? In the article it says they order once a month and have a reorder point of like 15 from what I read.
A: If you guys are able to reach our reorder point. So, whatever the reorder point of that tire is , will vary. On the number of times we purchased it on how it gets to point a and to the reorder point…. Yes, how fast that tire will get to the reorder point. We have 1000 tires and if we sold 750 tires then we would need to buy again. If we sold 10 then we would rebuy later.

Q: Are ford, Honda, Chevrolet and Toyota the only manufacturers you buy from?
A: It's just sample data. You bring your car into wherever. Everyone's got a different car. My point is there's many other manufacturers, we're not just holding to those.

Q: Are there only certain makes of tires that you have in stock? And if so, would you custom order tires by a customer's request?
A: So, a customer brings their car right. That car has a make. Then that car has a model. So, those are all related to each other in a way. To answer that question, no we will not order custom, but we will have the tires in stock and will order on what the make and model.
So basically, a customer can have a ton of different cars, there's no cap on what number of cars we sell tires for.

Q: Do we need to track each customer's credit limit?
A: We do need to track our customer's credit limit. Just one person and I bought some tires one day but I just got paid, and something happened, and I don't have a couple 100 dollars and I have to buy my next set and we have to keep a credit score for all our customers.

Q: Is there a max number of tires we can sell to each customer?
A: So there's not a max. No one buys more than 4 tires at a time, but there's not a cap. But we do only sell to retail customers.

Significant Assumptions

1. The customer picking up the car is the same customer that is dropping it off.
2. Discounts will be listed in a separate entity called PaymentTerms.
3. The discounts will be a percentage.

What is an ERD? Why is it necessary?

An ERD maps out the relationships between various entities. An entity is a thing or process used in a business. For example, “Employee” and “SalesOrder” are entities that would need to be mapped in the ERD. The ERD would then describe the relationships between these entities using verbs and cardinalities. So, the ERD may specify that an Employee can *process* zero to many SalesOrders. It is important that we use ERDs when creating databases so that we can plan out all of the necessary entities and relationship that Sonner Tire uses in their business operations. After completing the ERD, it should be an accurate representation of all the different business operations that need to be tracked in the database.

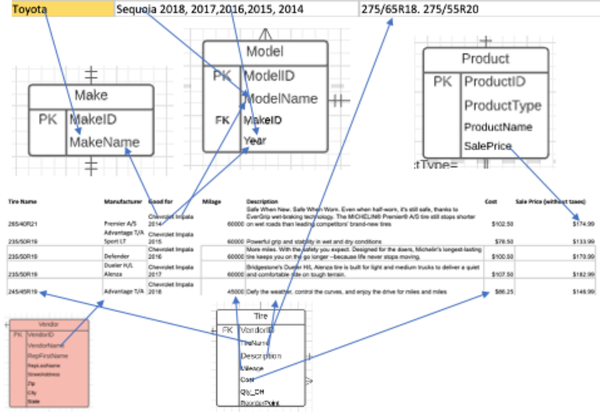
Business Cycles Used

We used the revenue and expenditures cycles because Sonner Tire only needs to track the revenue they receive from selling tires and minimize the cost of inventory and supplies and other services. The business revenue cycle occurs anytime a company sells products or services. Within the revenue cycle, the company’s revenue activities will be recorded. The expenditure cycle occurs with the purchase of and payment for goods and services. Through the expenditure cycle, Sonner Tires will be able to better track and purchase inventory. Sooner Tire only sells products and services, and buys products such as the tires they work with, thus the company does not need to track productions. Sonner Tire needs to record: sales made to customers, information about the customer, employee, product, the vendor, payment, inventory, and goods purchased, which is why the revenue and expenditure cycle are appropriate to use for the company’s needs.

Data Provided by Client

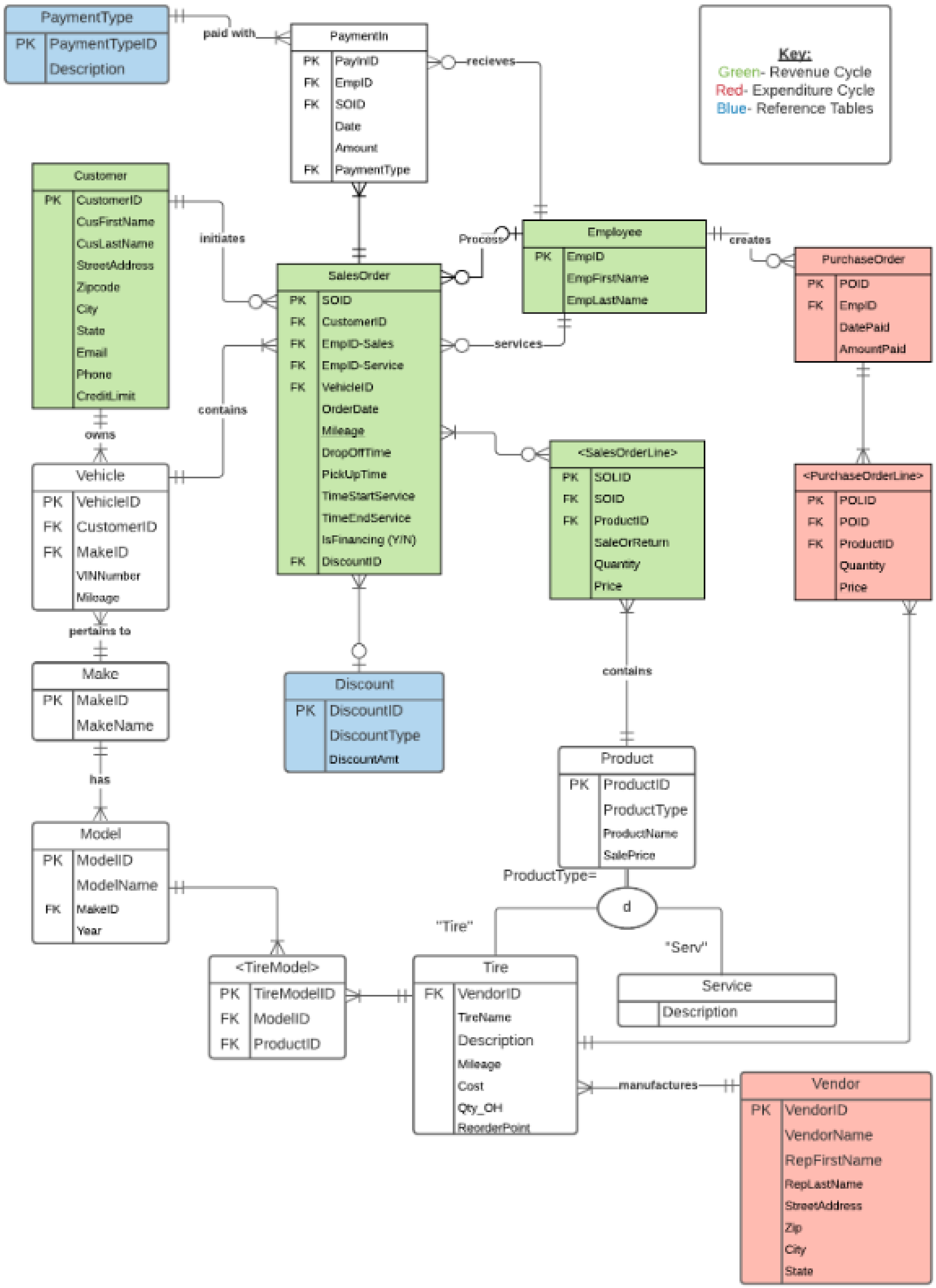
In the cars table provided by the client, there are three tables necessary: make, model, and tire. The make name will be associated with the make table. For the model table, name and year will be associated with it as attributes. Lastly, in the tire table, the attributes associated with it are name, mileage, description, cost, and sale price. This data tells us that we will need to have relationships between make, model, and tire in our ERD. Below is the first five records with data for each of the tables and their associated attributes that were given to us in the cars table.

Car Manufacturer	Model	Tire
Ford	Escape 2019, 2018, 2017, 2016, 2015, 2014	235/55R17, 235/50R18, 235/50R19, 235/45R19
Ford	Mustang 2019, 2018, 2017, 2016, 2015, 2014	235/50R18, 255/40R19
Ford	Edge 2019, 2018, 2017, 2016, 2015, 2014	245/50R20, 265/40R21
Ford	Explorer 2019, 2018, 2017, 2016, 2015, 2014	255/50R20, 245/60R18, 235/65R18
Honda	Accord 2018, 2017, 2016, 2015, 2014	235/40R19, 225/50R17, 215/55R17
Honda	Civic 2019, 2018, 2017, 2016, 2015,	215/50R17, 215/55R16,205/55R16
Honda	Odyssey 2019, 2018, 2017, 2016, 2015, 2014	235/65R17, 235/65R18, 235/60R18
Honda	Pilot 2018, 2017, 2016, 2015, 2014	245/50R20, 245/65R17
Chevrolet	Malibu 2019, 2018, 2017, 2016, 2015	205/65R16, 215/60R16
Chevrolet	Impala 2018, 2017, 2016, 2015, 2014	235/50R18, 245/45R19, 225/60R16
Chevrolet	Tahoe 2017, 2016, 2015, 2014	265/70R17,265/65R18, 275/55R20
Chevrolet	Silverado 2019,2018, 2017, 2016, 2015	255/70R17,265/70R17
Toyota	Camry 2018, 2017, 2016, 2015	205/65R26, 215/55R17
Toyota	Corolla 2017, 2016, 2015, 2014, 2013	195/65R15,205/55R16
Toyota	4Runner 2018, 2017,2016, 2015	215/45R17, 245/60R20, 205/55R16
Toyota	Sequoia 2018, 2017,2016,2015, 2014	275/65R18. 275/55R20



ERD Created

On the following page, we have provided a picture of the ERD that we created for Sonner Tire based off of the case information given to us and the feedback from the client. Our ERD is color coded according to the revenue cycle, expenditure cycle, and reference tables. The ERD was created using LucidChart. Many of the entities in our diagram were taken from the generic revenue and business cycles, but we had to make some changes based on what Sonner Tires needed. We will go into further detail about the changes we had to make in the query feasibility and current ERD section.


PaymentType
PK PaymentTypeID
Description
paid with
PaymentIn
PK PayInID
FK EmpID
FK SOID
Date
Amount
FK PaymentType
recieves
Key:
Green- Revenue Cycle
Red- Expenditure Cycle
Blue- Reference Tables
Customer
PK CustomerID
CusFirstName
CusLastName
StreetAddress
Zipcode
City
State
Email
Phone
CreditLimit
initiates
Process
Employee
PK EmpID
EmpFirstName
EmpLastName
creates
PurchaseOrder
PK POID
FK EmpID
DatePaid
AmountPaid
SalesOrder
PK SOID
FK CustomerID
FK EmpID-Sales
FK EmpID-Service
FK VehicleID
OrderDate
Mileage
DropOffTime
PickUpTime
TimeStartService
TimeEndService
IsFinancing (Y/N)
FK DiscountID
services
contains
owns
Vehicle
PK VehicleID
FK CustomerID
FK MakeID
VINNumber
Mileage
<SalesOrderLine>
PK SOLID
FK SOID
FK ProductID
SaleOrReturn
Quantity
Price
<PurchaseOrderLine>
PK POLID
FK POID
FK ProductID
Quantity
Price
pertains to
Make
PK MakeID
MakeName
Discount
PK DiscountID
DiscountType
DiscountAmt
contains
has
Product
PK ProductID
ProductType
ProductName
SalePrice
ProductType=
d
"Tire"
"Serv"
Model
PK ModelID
ModelName
FK MakeID
Year
<TireModel>
PK TireModelID
FK ModelID
FK ProductID
Tire
FK VendorID
TireName
Description
Mileage
Cost
Qty_OH
ReorderPoint
Service
Description
manufactures
Vendor
PK VendorID
VendorName
RepFirstName
RepLastName
StreetAddress
Zip
City
State

Query Feasibility and Current ERD

This chart contains each of the query questions requrested by Sonner Tirea. We determined which tables would be needed to run the query, and provided a projected SQL statement. Additionally, we added comments about what changes we needed to make to the generic ERD in order to make these queries work. This process is a vital precursor to the logical design process so that we can ensure that our database will be able to run the queries that Sonner Tire needs.

#	Query Question	Tables needed to run the query	Projected SQL Statement
1	Total sales (in dollars) by region for a given tire manufacturer and car manufacturer. It would be great if we can specify the car model and year too (note that we would like to be able to input the month to be calculated).	Vendor, Make, Model, SalesOrder, <SalesOrderLine>, Customer We added the entities Make, Model, and Year. These entities have a relationship with vehicle.	Select OrderDate, Mod.ModelName, M.MakeName, Year, VendorName, SUM(Quantity*Price) as TotalSales, State From Model Mod Join Make M On Mod.MakeID = M.MakeID Join Vehicle V On V.MakeID = M.MakeID Join Customer C On V.CustomerID = C.CustomerID Join SalesOrder SO On C.CustomerID = SO.CustomerID Join SalesOrderLine SOL On SO.SOID = SOL.SOLID Join Product P On SOL.SOLID = P.ProductID Join Tire T On P.ProductID = T.ProductID Join Vendor Ven On T.VendorID = Ven.VendorID Where Month(OrderDate) = ‘__’, ModelName = ‘__’ , MakeName = ‘__’, Year = ‘__’, VendorName = ‘__’, State = ‘__’ Group by OrderDate, ModelName, MakeName, Year, VendorName, State

2	Total sales (in dollars) by a customer in a given year.	PaymentIn, SalesOrder, Customer	Select CustomerID, LastName, FirstName, OrderDate, SUM(Amount) as TotalSales From Customer C Join SalesOrder SO On C.CustomerID = SO.SOID Join PaymentIn PI On SO.SOID = PI.PayInID Where Year(OrderDate) = ‘__’ Group by CustomerID, LastName, FirstName, OrderDate
3	The five highest selling tires.	<SalesOrderLine>, Product, Tire We added Tire and Service as sub-types of Product.	Select TOP 5 TireName, MAX(Quantity) From Tire T Join Product P On P.ProductID = T.ProductID Join SaleOrderLine SOL On P.ProductID = SOL.SOLID Group by TireName
4	Itemized invoices for jobs for each customer that need to include tires purchased/tire rotation/tire repair/tire protection.	SalesOrder, <SalesOrderLine>, Customer, Product	Select CustomerID, LastName, FirstName, ProductType, P.Price, Quantity From Customer C Join SalesOrder SO On C.CustomerID = SO.SOID Join SalesOrderLine SOL On SO.SOID = SOL.SOLID Join Product P On P.ProductID = SO.ProductID Join Tire T On T.ProductID = P.ProductID Group By CustomerID
5	The number and type of job performed by	Employee	Select EmpID, FirstName, LastName, ServiceTypeID

	each of our employees.		
6	Number of times a tire protection has been purchased for a particular tire and number of times free service has been applied (free tire damage repair, free replacement).	SalesOrder, <SalesOrderLine>, Product Still not sure how to use sub type super type. Also do not know how to group by tire repair and replacement	Select SalesOrder From SaleOrderLine On SOL on P.Product ID= SOL.SOLIDID From Product P Join Service Serv On P.ProductP = Serv.ProductID
7	The following items for Purchase Orders: manufacturer name, number of POs, total cost.	PurchaseOrder, <PurchaseOrderLine>, Vendor, Tire	Select COUNT(POID) as NumPOS, SUM(Cost) as TotalCost From (Select Distinct VendorID, POID, VendorName From PurchaseOrder PO Join PurchaseOrderline POL On PO.POID = POL.POID Join Tire T On T.ProductID = POL.ProductID Join Vendor V On T.VendorID = V.VendorID) SQ
8	Number of orders and total sales per customer in the past 2 years. This report is particularly important as it shows the number of	Customer, SalesOrder, <SalesOrderLine>, Product	Select CustomerID, LastName, FirstName, SUM(Quantity*SalePrice) as TotalSales, COUNT(SOID) as NumOrders, OrderDate From Customer C Join SalesOrder SO On C.CustomerID = SO.SalesOrderID Join SalesOrderLine SOL On SO.SalesOrderID = SOL.SalesOrderID Join Product P On SOL.ProductID = P.ProductID

	returning customers.		Where Year(OrderDate) = GETDATE() -2 Group by CustomerID, LastName, FirstName, OrderDate
9	List of tires that have not been purchased within the last 6 months (in order to better manage inventory).	Product, SalesOrderLine, SalesOrder	Select TireName, OrderDate From Product P Join Tire T On P.ProductID = T.ProductID Left Join SalesOrderLine SOL On P.ProductID = SOL.SOLID Join SalesOrder SO On SOL.SOLID = SO.SOID Where ProductType = “Tire” and Month(OrderDate) Between Month(GetDate())-6 And Month(GetDate())
10	Names of customers who took advantage of the financing option, date purchased, total amount purchased, credit limit, number of payments made, the total amount paid, outstanding amount, is time to pay-off less than 6 months, all displayed from the latest date and	Finance, PaymentTerms, Customer, SalesOrder, PaymentIn	Select FirstName, LastName, OrderDate, SUM(OrderTotal), CreditLimit, COUNT(PayInID), SUM(Amount), OrderTotal – SUM(Amount) As Outstanding_Amount From Finance F Join Customer C On F.FinanceID = C.FinanceID Join PaymentTerms PT On C.TermsID = PT.TermsID Join SalesOrder SO On C.CustomerID = SO.CustomerID Join PaymentIn P On SO.PayInID = P.PayInID Where C.FinanceID Not Null Group By FirstName, LastName, OrderDate, CreditLimit Having Length < 6 Order By OrderDate Desc, Outstanding_Amount Desc

	then the largest amount owed.		
11	Total profit per tire type and manufacturer type in the past 6 months.	Product, Tire, <SalesOrderLine>, SalesOrderLine, Make Not sure if TotalProfit is correct.	Select SUM((P.Price-Cost) * Quantity) as TotalProfit, TireName, ModelName, OrderDate From Make Mk Join Model Md On Mk.MakeID = Md.ModelID Join ModelTire MT On Md.ModelID = MT.ModelTireID Join Product P On MT.ModelTireID = P.ProductID Join SalesOrderLine SOL On P.ProductID = SOL.SOLID Join SalesOrder SO ON SOL.SOLID = SO.SOID Where Month(OrderDate) = GETDATE() – 6 Group by TireName, ModelName, OrderDate
12	List of all customers that have not made a purchase within the last 12 months from the current date.	Customer, SalesOrder	Select CustomerID, LastName, FirstName,OrderDate From Customer C Left Join SalesOrder SO On C.CustomerID = SO.SOID Where Month(OrderDate) between Month(GetDate())-12 And Month(GetDate())
13	List of customers whose average sales is less than the average of all sales. This will help us to find customers whom	Customer, SalesOrder	Select Customer, AVERAGE(COST) as Average Cost, Average(OrderTotal) as Total Average Sales Join Customer C On SalesOrder SOI C.Customer = C.SalesOrder Where Average Cost < Total Average Sales Order By (MAX) Average Cost ascen

	we should target to get a higher volume of sales.		

Logical Design

The logical design phase occurs after the conceptual design phase. In this phase, the ERD has already been created, so now it is important to ensure that the database is designed correctly so that it can run without issues. In the logical design phase, entities are converted into relations. Before we can begin writing out the relations, we undergo the process of normalization. Normalization is a crucial part of the logical design process. Then, there are several rules and constraints that need to be followed when converting the entities into relations, which we will discuss further in detail.

Normalization

The process of normalization is intended to make the database is reliable and efficient. To normalize the data structure, we must ensure that each column is “atomic” meaning it cannot be broken down any further. For example, we don’t want to have any multi-valued attributes like “name” in the database because it is not specific enough. We must break it down into its base components of first name and last name. Second, the columns must not contain redundant data, which increases the amount of time it takes to run queries. By doing these two things we can ensure our database will have data integrity and run efficiently. Lastly, we must remove dependencies. To do this, we make sure there are not any dependency constraints. Constraints will be described further in the data integrity section.

Normalization of the Data Provided by the Client

To normalize the data that was provided by the client, we first had to ensure that each column was broken down into its most basic form and did not have multiple values. From here we then ensured that there were no partial or functional dependencies in the tables. What I mean by this is that every column is predicted by the key element within the table and that key element only.

TMake(MakeID, MMakeName)

TCarModel(ModelID, CMMakeID, CMModelName, CMYear)

Foreign Key CMMakeID references TMake

Not Null

On delete Restrict

TModelTire(MTID, ModModelID, ModProductID-Tire)

Foreign Key ModModelID references TCarModel

Not Null

On delete restrict

TCar(CarID, CCarManufacturer, CModelID, CModel, CYear, CTire)

Foreign Key CModelID references TCarModel

Not Null

On delete Restrict

TTires(TireID, TireName, TireManufacturer, TireGoodFor, TireMileage, TireDescription, TireCost, TireSalesPrice, TireQTY_OH, TireQTY_Committed, TireReorderPoint)

Normalized Relations

Revenue Cycle

TCustomer(CustomerID, CustFirstName, CustLastName, CustStreetAddress, CustZipcode, CustCity, CustState, CustEmail)

TEmployee(EmpID, EmpFirstName, EmpLastName)

TDiscount(DiscountID, DDiscountType)

TSalesOrder(SOID, SOPayInID, SOCustomerID, SOEmpID-Sales, SOEmpID-Service, SODiscountID, SOVehicleID, SOTechFirstName, SOTechLastName, SOOrderDate, SOMileage, SODropOffTime, SOPickUpTime, SOTimeStartService, SOTimeEndService, SOIsFinancing)

Foreign Key SOPayInID references TPaymentIn
Null Allowed
On delete set null
Foreign Key SOCustomerID references TCustomer
Not Null
On delete restrict
Foreign Key SOEmpID-Sales references TEmployee
Not Null
On delete restrict
Foreign Key SOEmpID-Service references TEmployee
Null allowed
On delete set null
Foreign Key SODiscountID references TDiscount
Null Allowed
On delete set null
Foreign Key SOVehicleID references TVehicle
Not Null
On delete restrict

TSalesOrderLine(SOLID, SOLSOID, SOProductID, SOLStatus, SOLSaleOrReturn, SOLQuantity, SOLPrice)

Foreign Key SOLSOID references TSalesOrder
Not Null
On delete restrict
Foreign Key SOProductID references TProduct
Not Null
On delete restrict

TPaymentType(PaymentTypeID, PTDescription)

TPaymentIn(PayInID, PayEmpID, PayPaymentType, PayDate, PayAmount, PayCardNumber, PayExpirationDate, PaySecurityCode)

Foreign Key PayPaymentType references TPaymentType
Null allowed
On delete set null
Foreign Key PayEmpID references TEmployee
Not Null
On delete Restrict

Expenditure Cycle

TVendor(VendorID, VVendorName, VSalesRepFirstName, VSalesRepLastName)

TPurchaseOrder(POID, POEmpID, PODatePaid, POAmountPaid)
Foreign Key POEmpID references TEmployee
Not Null
On Delete Restrict

TPurchaseOrderLine(POLID, POLPOID, POLProductID, POLQuantity, POLPrice)
Foreign Key POLPOID references TPurchaseOrder
Not Null
On Delete Restrict
Foreign Key POLProductID references TTire
Not Null
On Delete Restrict

TMake(MakeID, MMakeName)

TModel(ModelID, MOMakeID, MOModelName, MOYear)
Foreign Key MOMakeID references TMake
Not null
On delete Restrict

TVehicle(VehicleID, VEHCustomerID, VEHMakeID, VINNumber)
Foreign Key VEHCustomerID references TCustomer
Not null
On Delete Restrict
Foreign Key VEHMakeID references TMake
Not null
On delete restrict

TProduct(ProductID, PProductType, PSalePrice)

TService(ServProductID, ServLifeTimeProtection)

TTire(TireProductID, TireVendorID, TireName, TireDescription, TireMileage, TireCost, TireQty_OH, TireQTY_Committed, TireReorderPoint)

Foreign Key TIVendorID references TVendor
Not Null
On Delete Restrict

TTireModel(TireModelID, TMModelID, TMProductID)
Foreign Key TMModelID references TModel
Not Null
On Delete Restrict
Foreign Key TMProductID references Ttire
Not Null
On Delete Restrict

Differences between ERD and Normalized Relations

One difference between ERDs and normalized relations are that ERDs can have multi-valued attributes, while normalized relations should be broken down into smaller attributes in order to make the entity atomic. Atomicity is important so that reports made within the database are efficient and accurate. Similarly, normalized relations do not include derived attributes. This is important because derived attributes are calculated from other attributes, so they do not need to be included in normalized relations. Furthermore, the names of the entities in normalized relations are different from the names of the entities in the ERD. In normalized relations, we add a T to the beginning of the name of the entity to indicate that it is a table. Additionally, the attributes in normalized relations have unique names. This is beneficial because it will prevent us from getting unambiguous column names in our queries.

Database Integrity

Data integrity means that the reports generated from the database are trustworthy. Normalization is one way to ensure that data integrity is accomplished. There are three integrity constraints: entity, referential, and domain. The entity integrity constraint is that every entity must have a primary key that isn't null and doesn't change over time. The referential integrity applies to the relationships between entities. It states that for each relationship, the foreign key in one entity must match the primary key in the other entity, or null if applicable. The last is the domain integrity constraint. This constraint says that every value in a column must be of the same data type, like integer or string. We ensured that these constraints were enforced by having a related primary key, none of which that are null, and foreign key for each of the relationships in our diagram. We enforced the referential constraint by not allowing any multi-value attributes.

Physical Design and Implementation

The Physical Design phase is the part of the database building process where we choose which relational database management system (RDBMS) we will we be using. This is important because different RDBMS have different data types that they use to store information. We will be using Microsoft SQL Server as our RDBMS. The next step is the actual implementation of data into the database which we did using dummy data. The purpose of this was to make sure everything in the database was working without any errors. Without this phase of the database design process, we wouldn’t be able to create a database. Rather, we would just have the conceptual design all planned out on paper.

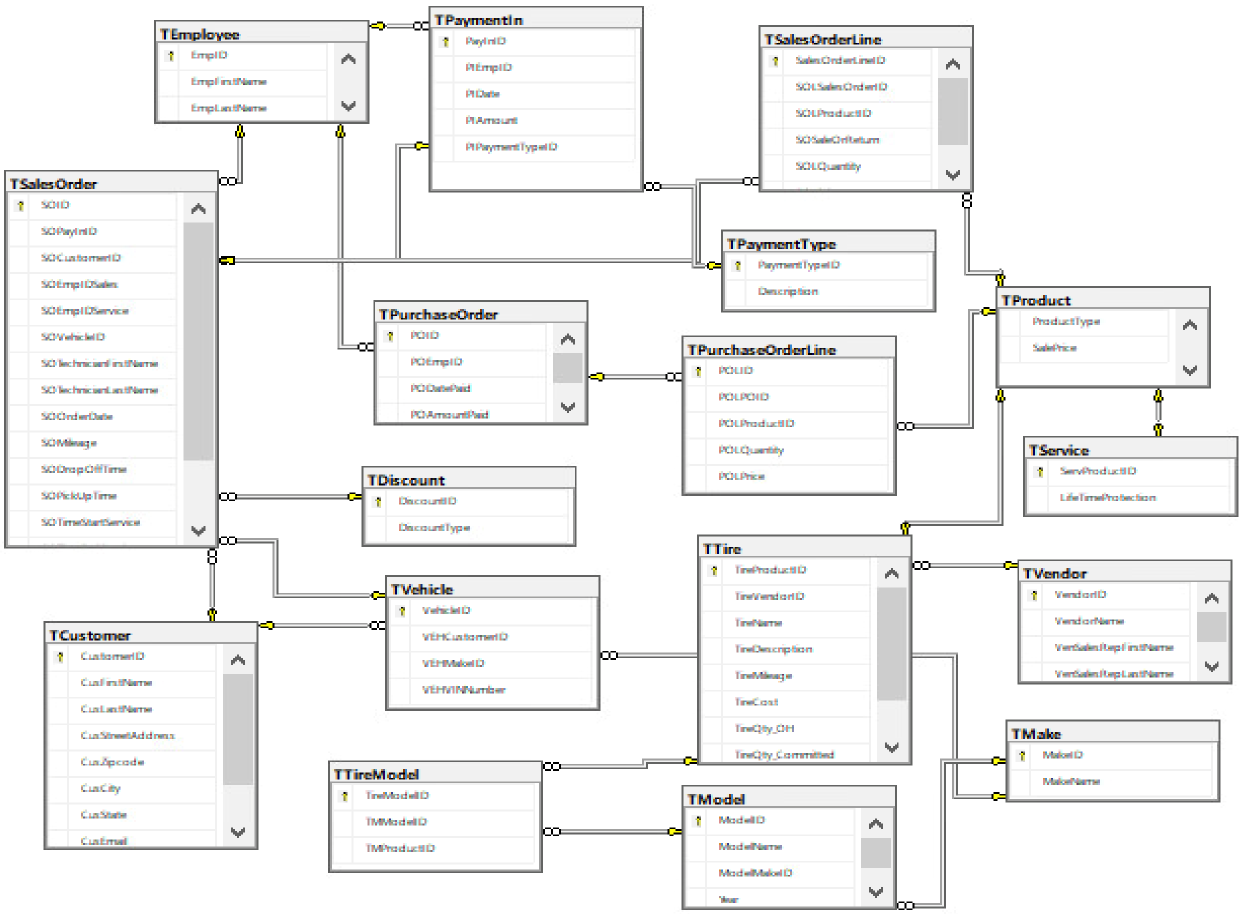
Data Dictionary

A data dictionary is a collection of information describing the data included in a database and the relationships between the information. It is used as a way to better understand the structure and information within a database. It includes things like entity names, attributes, and their data types. As an example, the data dictionary for our project includes things like the Customer table and its attributes being things like their first name, last name, and the customer type. The data types, whether or not it is allowed to be null, what table it references if it is a foreign key, and a sample of the key will be included on the same row as the attribute.

Denormalization

Denormalization is the process of removing of some of the normalized relations of the data in order to improve performance. When denormalizing the data, it will make it more efficient to run SQL queries that include a long list of join statements. While we recognize that denormalization results in data duplication and redundancy, we made the choice to denormalize some of our data. First, we decided to remove the year table and list the year in the model table instead. We also decided to list the customer’s state and zip code as attributes in the customer table, rather than having separate tables for them. We decided to do this with the vendor’s address as well. These changes make it easier for us and the client to write queries and will allow the queries to run quicker.

Implemented Physical Design



Strengths and Weaknesses Encountered During Implementation

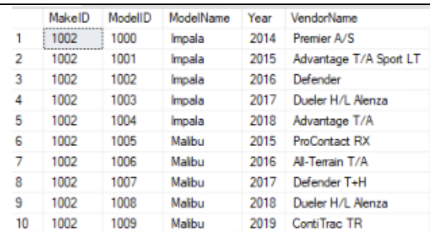
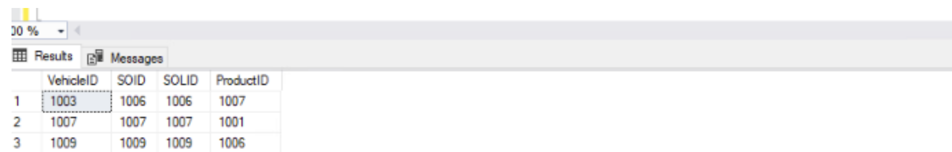
One of our strengths was that our ERD was close to finalized before going into the implementation phase, which made it easier for us to create tables in the database. Creating the tables in the database was also one of our strengths because we simply had to create the tables in the order that we wrote our normalized relations. However, one of our weaknesses when creating the tables in the database was that the attributes of the tables were named inconsistently since we were working on them separately. The inconsistency of the attribute names made it more difficult for us to write the SQL queries and implement them into our database. To fix this issue, we would need to delete the tables and create them again with the appropriately named attributes. Given the time constraints, we decided to leave the tables the way they are.

Specific SQL Statements Requested

Here we will list the specific programs we were asked to execute by the client in the database, as well as additional queries we believe would be useful for the operation. We have included the request asked of us, the SQL code needed to implement the program, and an image of the result of the program. Some of the ouputs are empty because none of the sample data applied to the requested query. When more data is added to the database, it will show more results.

<table>
<tr><th>Query #</th><th>Question</th><th>SQL</th><th>Partial Output</th></tr>
<tr>
<td>1</td>
<td>Total sales (in dollars) for a given tire manufacturer and car manufacturer. It would be great if we can specify the car model and year too (note that we would like to be able to input the month to be calculated).</td>
<td>SELECT MakeID, ModelID, MOModelName, MOYear, VendorName, Sum(Quantity*SalePrice) TotalSales
FROM View1 JOIN View2
ON
View1.ProductID=View2.ProductID
WHERE month(date)=[Parameter]

View1
SELECT MakeID, ModelID, ModelName, Year, VendorName
FROM TMake
JOIN TModel
ON MakeID = ModelMakeID
JOIN TTireModel
ON ModelID = TMModelID
JOIN TTire
ON TMProductID = TireProductID
Join TVendor
ON VendorID = TireVendorID

View2</td>
<td>
<table>
<tr><th></th><th>MakeID</th><th>ModelID</th><th>ModelName</th><th>Year</th><th>VendorName</th></tr>
<tr><td>1</td><td>1002</td><td>1000</td><td>Impala</td><td>2014</td><td>Premier A/S</td></tr>
<tr><td>2</td><td>1002</td><td>1001</td><td>Impala</td><td>2015</td><td>Advantage T/A Sport LT</td></tr>
<tr><td>3</td><td>1002</td><td>1002</td><td>Impala</td><td>2016</td><td>Defender</td></tr>
<tr><td>4</td><td>1002</td><td>1003</td><td>Impala</td><td>2017</td><td>Dueler H/L Alenza</td></tr>
<tr><td>5</td><td>1002</td><td>1004</td><td>Impala</td><td>2018</td><td>Advantage T/A</td></tr>
<tr><td>6</td><td>1002</td><td>1005</td><td>Malibu</td><td>2015</td><td>ProContact RX</td></tr>
<tr><td>7</td><td>1002</td><td>1006</td><td>Malibu</td><td>2016</td><td>All-Terrain T/A</td></tr>
<tr><td>8</td><td>1002</td><td>1007</td><td>Malibu</td><td>2017</td><td>Defender T+H</td></tr>
<tr><td>9</td><td>1002</td><td>1008</td><td>Malibu</td><td>2018</td><td>Dueler H/L Alenza</td></tr>
<tr><td>10</td><td>1002</td><td>1009</td><td>Malibu</td><td>2019</td><td>ContiTrac TR</td></tr>
</table>

00 %
Results Messages
<table>
<tr><th></th><th>VehicleID</th><th>SOID</th><th>SOLID</th><th>ProductID</th></tr>
<tr><td>1</td><td>1003</td><td>1006</td><td>1006</td><td>1007</td></tr>
<tr><td>2</td><td>1007</td><td>1007</td><td>1007</td><td>1001</td></tr>
<tr><td>3</td><td>1009</td><td>1009</td><td>1009</td><td>1006</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td>SELECT VehicleID, SOID, SOLID, ProductID
FROM TVehicle
JOIN TSalesOrder
ON VehicleID = SOVehicleID
JOIN TSalesOrderLine SOL
ON SOID = SOLID
JOIN TProduct
ON SOLProductID = ProductID
WHERE ProductType='Tire'</td><td></td></tr>
<tr><td>2</td><td>Total sales (in dollars) by a customer in a given year.</td><td>SELECT CusFirstName, CusLastName, Sum(PIAmount) TotalSales
FROM TCustomer
JOIN TSalesOrder
ON CustomerID = SOCustomerID
JOIN TPaymentIn
ON SOID = PISOID
GROUP BY CusFirstName, CusLastName</td><td><table><tr><th></th><th>CusFirstName</th><th>CusLastName</th><th>TotalSales</th></tr><tr><td>1</td><td>Jakeem</td><td>Bryan</td><td>863.94</td></tr><tr><td>2</td><td>Medge</td><td>Kirk</td><td>831.94</td></tr><tr><td>3</td><td>Reese</td><td>Levy</td><td>60</td></tr><tr><td>4</td><td>Quincy</td><td>Williams</td><td>353.98</td></tr></table></td></tr>
<tr><td>3</td><td>The five highest selling tires.</td><td>SELECT TOP 5 TireProductID, ProductName, Count(SOLID) TimesSold
FROM TSalesOrderLine
JOIN TProduct
ON SOLProductID = ProductID
JOIN TTire
ON TireProductID = ProductID
GROUP BY TireProductID, ProductName
ORDER BY TimesSold DESC</td><td><table><tr><th></th><th>TireProductID</th><th>TireName</th><th>TimesSold</th></tr><tr><td>1</td><td>1001</td><td>235/50R19</td><td>2</td></tr><tr><td>2</td><td>1002</td><td>235/50R19</td><td>1</td></tr><tr><td>3</td><td>1004</td><td>235/50R19</td><td>1</td></tr><tr><td>4</td><td>1006</td><td>265/40R21</td><td>1</td></tr><tr><td>5</td><td>1007</td><td>265/40R21</td><td>2</td></tr></table></td></tr>
</table>

<table>
<tr><td>4</td><td>itemized invoices for jobs for each customer that need to include tires purchased/tire rotation/tire repair/tire protection</td><td>SELECT CustomerID, SOID, VehicleID, SOOrderDate, SOLQuantity, ProductID, CusFirstName, CusLastName, SalePrice, (SOLQuantity*SalePrice) LineTotal
FROM TCustomer JOIN TVehicle on CustomerID=VEHCustomerID join TSalesOrder on VehicleID=SOVehicleID join TSalesOrderLine on SOID = SOLSOID join TProduct on SOLProductID = ProductID
WHERE CustID=[] AND CarID=[] AND Date=[]</td><td>(with the WHERE clause commented out)
<table>
<tr><th>CustomerID</th><th>SOID</th><th>VehicleID</th><th>SOOrderDate</th><th>SOLQuantity</th><th>ProductID</th><th>CusFirstName</th><th>CusLastName</th><th>SalePrice</th><th>LineTotal</th></tr>
<tr><td>1007</td><td>1000</td><td>1007</td><td>2021-04-17</td><td>1</td><td>1007</td><td>Quincy</td><td>Williams</td><td>148.99</td><td>148.99</td></tr>
<tr><td>1003</td><td>1010</td><td>1003</td><td>2020-10-20</td><td>4</td><td>1001</td><td>Jakeem</td><td>Bryan</td><td>133.99</td><td>535.96</td></tr>
<tr><td>1001</td><td>1011</td><td>1001</td><td>2020-11-22</td><td>2</td><td>1010</td><td>Reese</td><td>Levy</td><td>15</td><td>30</td></tr>
<tr><td>1003</td><td>1003</td><td>1003</td><td>2020-11-18</td><td>2</td><td>1006</td><td>Jakeem</td><td>Bryan</td><td>163.99</td><td>327.98</td></tr>
<tr><td>1007</td><td>1004</td><td>1007</td><td>2020-07-25</td><td>1</td><td>1002</td><td>Quincy</td><td>Williams</td><td>170.99</td><td>170.99</td></tr>
<tr><td>1009</td><td>1005</td><td>1009</td><td>2021-03-22</td><td>1</td><td>1007</td><td>Medge</td><td>Kirk</td><td>148.99</td><td>148.99</td></tr>
<tr><td>1009</td><td>1006</td><td>1009</td><td>2020-11-07</td><td>4</td><td>1001</td><td>Medge</td><td>Kirk</td><td>133.99</td><td>535.96</td></tr>
<tr><td>1001</td><td>1007</td><td>1001</td><td>2020-06-19</td><td>2</td><td>1010</td><td>Reese</td><td>Levy</td><td>15</td><td>30</td></tr>
<tr><td>1007</td><td>1008</td><td>1007</td><td>2020-04-28</td><td>2</td><td>1011</td><td>Quincy</td><td>Williams</td><td>17</td><td>34</td></tr>
<tr><td>1009</td><td>1009</td><td>1009</td><td>2020-05-27</td><td>1</td><td>1004</td><td>Medge</td><td>Kirk</td><td>146.99</td><td>146.99</td></tr>
</table></td></tr>
<tr><td>5</td><td>The number and type of job performed by each of our employees.</td><td>SELECT EmpID, EmpFirstName, EmpLastName, Count(SOEmpIDSales) Sales, Count(SOEmpIDService) Service

FROM TEmployee Join TSalesOrder On EmpID = SOEmpIDService

GROUP BY EmpID, EmpFirstName, EmpLastName</td><td>
<table>
<tr><th></th><th>EmpID</th><th>EmpFirstName</th><th>EmpLastName</th><th>Sales</th><th>Service</th></tr>
<tr><td>1</td><td>1001</td><td>Troy</td><td>Pruitt</td><td>3</td><td>3</td></tr>
<tr><td>2</td><td>1002</td><td>Paul</td><td>Trujillo</td><td>1</td><td>1</td></tr>
<tr><td>3</td><td>1005</td><td>Barry</td><td>Wolf</td><td>2</td><td>2</td></tr>
<tr><td>4</td><td>1006</td><td>Jessamine</td><td>Haynes</td><td>2</td><td>2</td></tr>
<tr><td>5</td><td>1008</td><td>Mona</td><td>Home</td><td>2</td><td>2</td></tr>
</table></td></tr>
<tr><td>6</td><td>Number of times tire protection has been purchased for</td><td>SELECT TireProductID, TireName, Count(TireProductID) NumPurchases
FROM TSalesOrderLine JOIN TProduct On SOLProductID = ProductID Join TTire On ProductID</td><td>
<table>
<tr><th>TireProductID</th><th>TireName</th><th>NumPurchases</th></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td>a particular tire</td><td>= TireProductID Join TService On ProductID = ServProductID WHERE ProductType='Serv' And ServDescription = 'tire protection' GROUP BY TireProductID, TireName</td><td></td></tr>
<tr><td>7</td><td>The following items for Purchase Orders: manufacturer name, number of POs, total cost.</td><td>SELECT VendorID, VendorName, Count(POID) NumPurchases, Sum(POLQuantity*POLPrice) Total FROM TPurchaseOrder Join TPurchaseOrderLine On POID = POLPOID Join TTire On POLProductID = TireProductID Join TVendor On TireVendorID = VendorID GROUP BY VendorID, VendorName</td><td>
<table>
<tr><th></th><th>VendorID</th><th>VendorName</th><th>NumPurchases</th><th>Total</th></tr>
<tr><td>1</td><td>1001</td><td>Advantage T/A Sport LT</td><td>2</td><td>1267.92</td></tr>
<tr><td>2</td><td>1002</td><td>Defender</td><td>1</td><td>170.99</td></tr>
<tr><td>3</td><td>1004</td><td>Advantage T/A</td><td>1</td><td>182.99</td></tr>
<tr><td>4</td><td>1006</td><td>All-Terrain T/A</td><td>1</td><td>327.98</td></tr>
<tr><td>5</td><td>1007</td><td>Defender T+H</td><td>2</td><td>297.98</td></tr>
</table>
</td></tr>
<tr><td>8</td><td>Number of orders and total sales per customer in the past 2 years. This report is particularly important as it shows the number of returning customers.</td><td>SELECT CustomerID, CusFirstName, CusLastName, Count(SOID) FROM TCustomer Join TSalesOrder On CustomerID = SOCustomerID WHERE SOOrderDate >= 2 years GROUP BY CustomerID, CusFirstName, CusLastName</td><td>
<table>
<tr><th></th><th>CustomerID</th><th>CusFirstName</th><th>CusLastName</th><th>NumSOS</th></tr>
<tr><td>1</td><td>1000</td><td>Dalton</td><td>Paul</td><td>1</td></tr>
<tr><td>2</td><td>1004</td><td>Lars</td><td>Richmond</td><td>1</td></tr>
<tr><td>3</td><td>1005</td><td>Rylee</td><td>Merrill</td><td>2</td></tr>
<tr><td>4</td><td>1006</td><td>Hedwig</td><td>Dodson</td><td>2</td></tr>
<tr><td>5</td><td>1007</td><td>Quincy</td><td>Williams</td><td>2</td></tr>
<tr><td>6</td><td>1008</td><td>Salvador</td><td>Shepherd</td><td>2</td></tr>
</table>
</td></tr>
</table>

<table>
<tr>
<td>9</td>
<td>List of tires that have not been purchased within the last 6 months (in order to better manage inventory).</td>
<td>SELECT ProductID, ProductName
FROM TSalesOrderLine
LEFT JOIN TProduct WHERE
SOLProductID IS Null AND
ProductType = ‘Tire’</td>
<td>Results Messages
<table><tr><th></th><th>ProductID</th><th>TireName</th></tr></table></td>
</tr>
<tr>
<td>10</td>
<td>Names of customers who took advantage of the financing option, date purchased, total amount purchased, credit limit, the number of payments made, the total amount paid, outstanding amount, is time to pay-off less than 6 months, all displayed from the latest date and then the</td>
<td>Payment type – cash, credit, check
SELECT CusFirstName,
CusLastName, Total, Paid, (Total - Paid) Remaining
FROM TCustomer Join
TSalesOrder On CustomerID =
SOCustomerID Join SQ1 On
SOCustomerID = SQ1.CustomerID
JOIN SQ2 ON
SQ1.SOID=SQ2.SOID
WHERE IsFinancing = ‘Y’ AND
Month(SOOrderDate) =
Month(GETDATE()) – 6

SQ1
SELECT CustomerID, SOID,
Sum(SOLQuantity*SOLPrice) Total
FROM TVehicle Join TSalesOrder
On VehicleID = SOVehicleID Join
TSalesOrderLine On SOID =
SOLSOID
GROUP BY CustomerID, SOID

SQ2</td>
<td><table><tr><th></th><th>CustomerID</th><th>SOID</th><th>Total</th></tr>
<tr><td>1</td><td>1001</td><td>1004</td><td>170.99</td></tr>
<tr><td>2</td><td>1003</td><td>1000</td><td>148.99</td></tr>
<tr><td>3</td><td>1003</td><td>1006</td><td>731.96</td></tr>
<tr><td>4</td><td>1007</td><td>1001</td><td>30</td></tr>
<tr><td>5</td><td>1007</td><td>1005</td><td>148.99</td></tr>
<tr><td>6</td><td>1007</td><td>1007</td><td>30</td></tr>
<tr><td>7</td><td>1009</td><td>1003</td><td>327.98</td></tr>
<tr><td>8</td><td>1009</td><td>1008</td><td>34</td></tr>
<tr><td>9</td><td>1009</td><td>1009</td><td>718.95</td></tr></table>
<table><tr><th></th><th>CustomerID</th><th>CusFirstName</th><th>CusLastName</th><th>Paid</th></tr>
<tr><td>1</td><td>1001</td><td>Reese</td><td>Levy</td><td>16.89</td></tr>
<tr><td>2</td><td>1003</td><td>Jakeem</td><td>Bryan</td><td>57.44</td></tr>
<tr><td>3</td><td>1007</td><td>Quincy</td><td>Williams</td><td>42.39</td></tr>
<tr><td>4</td><td>1009</td><td>Medge</td><td>Kirk</td><td>60.33</td></tr></table></td>
</tr>
</table>

<table>
<tr><td></td><td>largest amount owed.</td><td>SELECT CustomerID, CusFirstName, CusLastName, Sum(PIAmount) Paid
FROM TCustomer Join TVehicle On CustomerID = VEHCustomerID Join TSalesOrder On SOVehicleID = VehicleID Join TPaymentIn On SOID = PISOID
GROUP BY CustomerID, CusFirstName, CusLastName</td><td></td></tr>
<tr><td>11</td><td>Total profit per tire type and manufacturer type in the past 6 months.</td><td>SELECT ProductID, ProductName, SUM((SOLPrice-POLCost)*Quantity) Profit
FROM TSalesOrder join TSalesOrderLine on SOID=SOLSOID join TProduct P on SOLProductID=ProductID join TTire on ProductID=TireProductID
WHERE Month(SOOrderDate) = Month(GETDATE()) – 6
GROUP BY ProductID, ProductName</td><td>
Results | Messages
<table><tr><th></th><th>ProductID</th><th>TireName</th><th>Profit</th></tr><tr><td>1</td><td>1001</td><td>235/50R19</td><td>296.92</td></tr><tr><td>2</td><td>1002</td><td>235/50R19</td><td>63.49</td></tr><tr><td>3</td><td>1004</td><td>235/50R19</td><td>105.49</td></tr><tr><td>4</td><td>1006</td><td>265/40R21</td><td>152.98</td></tr><tr><td>5</td><td>1007</td><td>265/40R21</td><td>155.48</td></tr></table><table><tr><th>ProductID</th><th>TireName</th><th>Profit</th></tr></table></td></tr>
<tr><td>12</td><td>List of all customers that have not made a purchase within the last 12 months from the current date.</td><td>SELECT CustomerID, CusFirstName, CusLastName
FROM TCustomer C JOIN TVehicle V on C.CustomerID=V.VEHCustomerID
Left Join TSalesOrder on VehicleID= SOVehicleID
WHERE SOID IS Null AND Month(SOOrderDate) = Month(GETDATE()) – 12</td><td>
<table><tr><th>CustomerID</th><th>CusFirstName</th><th>CusLastName</th></tr></table></td></tr>
</table>

<table>
<tr>
<td>13</td>
<td>List of customers whose average sales is less than the average of all sales. This will help us to find customers whom we should target to get a higher volume of sales.</td>
<td>SELECT CusFirstName, CusLastName, AVG(SOLQuantity*SOLPrice) AVGPurchase
FROM TCustomer Join TSalesOrder On CustomerID = SOCustomerID Join TSalesOrderLine On SOID = SOLSOID
WHERE AVG(SOLQuantity*SOLPrice) < (SELECT AVG(SOLQuantity*SOLPrice) From TSalesOrderLine)</td>
<td>
<table>
<tr><th></th><th>CusFirstName</th><th>CusLastName</th><th>AVGPurchase</th></tr>
<tr><td>1</td><td>Rylee</td><td>Merrill</td><td>30</td></tr>
<tr><td>2</td><td>Dalton</td><td>Paul</td><td>170.99</td></tr>
<tr><td>3</td><td>Lars</td><td>Richmond</td><td>148.99</td></tr>
<tr><td>4</td><td>Quincy</td><td>Williams</td><td>34</td></tr>
</table>
</td>
</tr>
</table>

Three Additional Queries

Listed below are additional queries that can be used by the owners. These queries help provide the owners retrieve information about specific customers and manufacturers they deal with the most. With this additional information the company can make better decisions about what manufacturers to buy from and what customers need the most.

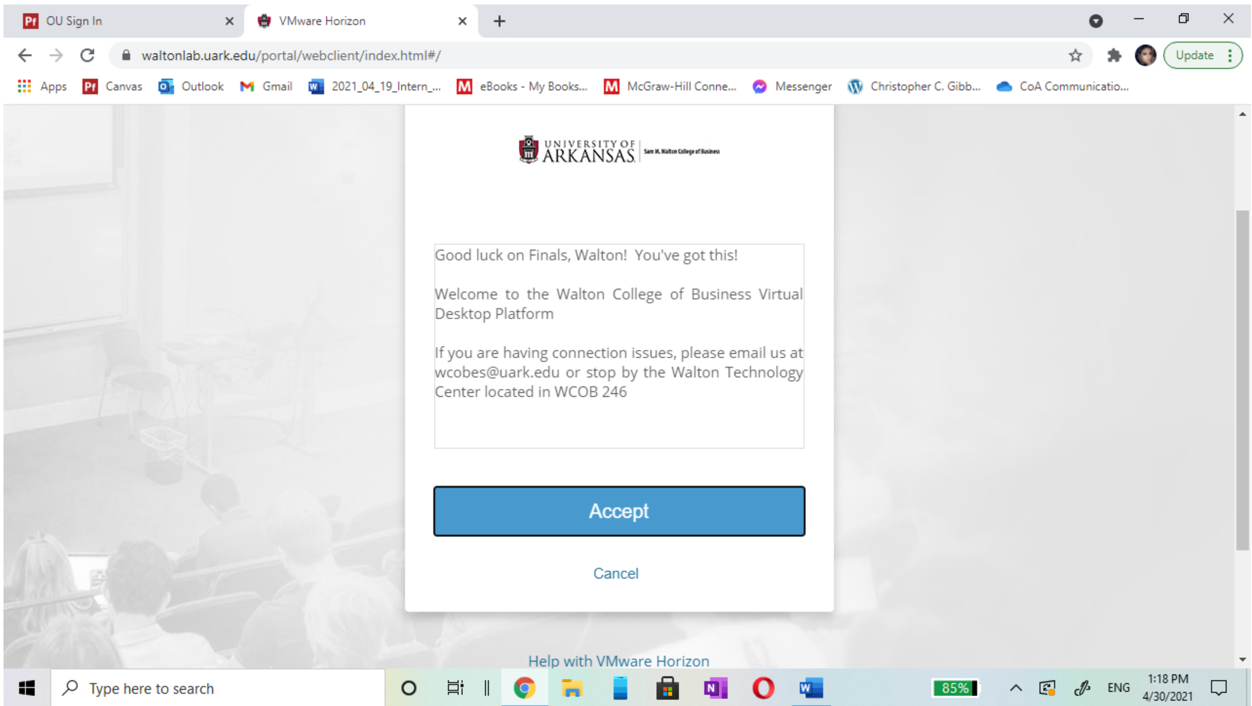
Query #	Question	Why is this important	SQL	Partial Output	Recap of Findings
1	Three lowest sold tires in the past six months	It will be important to know what type of tires are not selling as much so we can consider not	SELECT Lower 3 TireProductID, ProductName, Count(SOLID) TimesSold FROM TSalesOrderLine SOL JOIN TProduct P ON SOL.SOLProductID = P.ProductID		This query will help the owners determine which tire's are selling the least and allow them to be determine if they want

		buying them to make better use of our inventory	JOIN TTire ON TireProductID = P.ProductID GROUP BY TireProductID, ProductName		to keep purchasing from that manufacturer.
2	Names of the customers who spent the most money	It would be good to know what customers are spending the most money so we can make sure we do not lose them.	Select CustomerID, CusLastName, CusFirstName, SUM(PIAmount) as TotalSales From TCustomer C Join TSalesOrder SO On C.CustomerID = SO.SOCustomerID Join PaymentIn PI On SO.SOID = PI.PISOID Group by CustomerID, CusLastName, CusFirstName Order By TotalSales		This query will let the owners know who their top customer is so they can offer special discounts accordingly.
3	Make, model, and year of top three cars that come in the most	Knowing the top makes and models of cars that have been coming in is good so we will be able to use it in advertisements.	Select Mod.MOModelName, M.MMakeName, MOYear, SUM(SOLQuantity*SOLPrice) as TotalSales From Model Mod Join Make M On Mod.MOMakeID = M.MakeID Join Vehicle V On V.VEHMakeID = M.MakeID Join Customer C On V.VEHCustomerID = C.CustomerID Join SalesOrder SO On C.CustomerID = SO.SOCustomerID		This query will help determine which services will be performed most frequently.

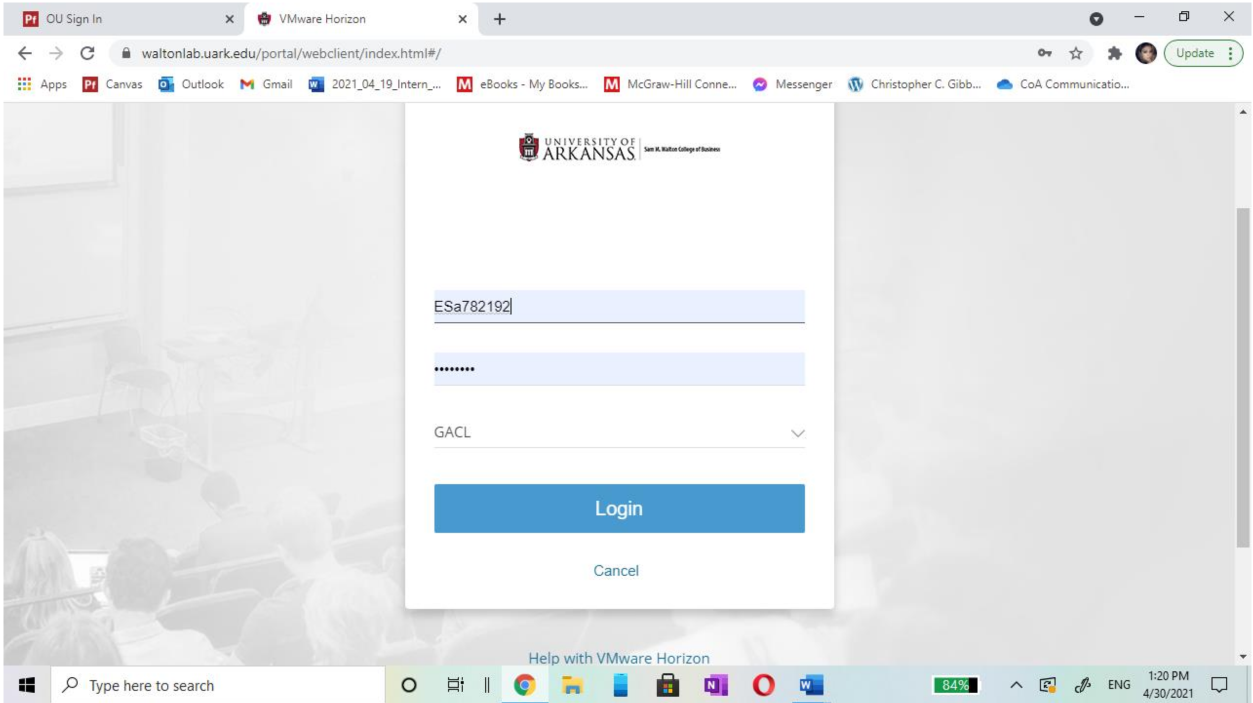
			Join SalesOrderLine SOL On SO.SOID = SOL.SOLSOID Where MOModelName = ‘__’ , MMakeName = ‘__’, MOYear = ‘__’ Group by MOModelName, MMakeName, MOYear Order By TotalSales		

User Documentation

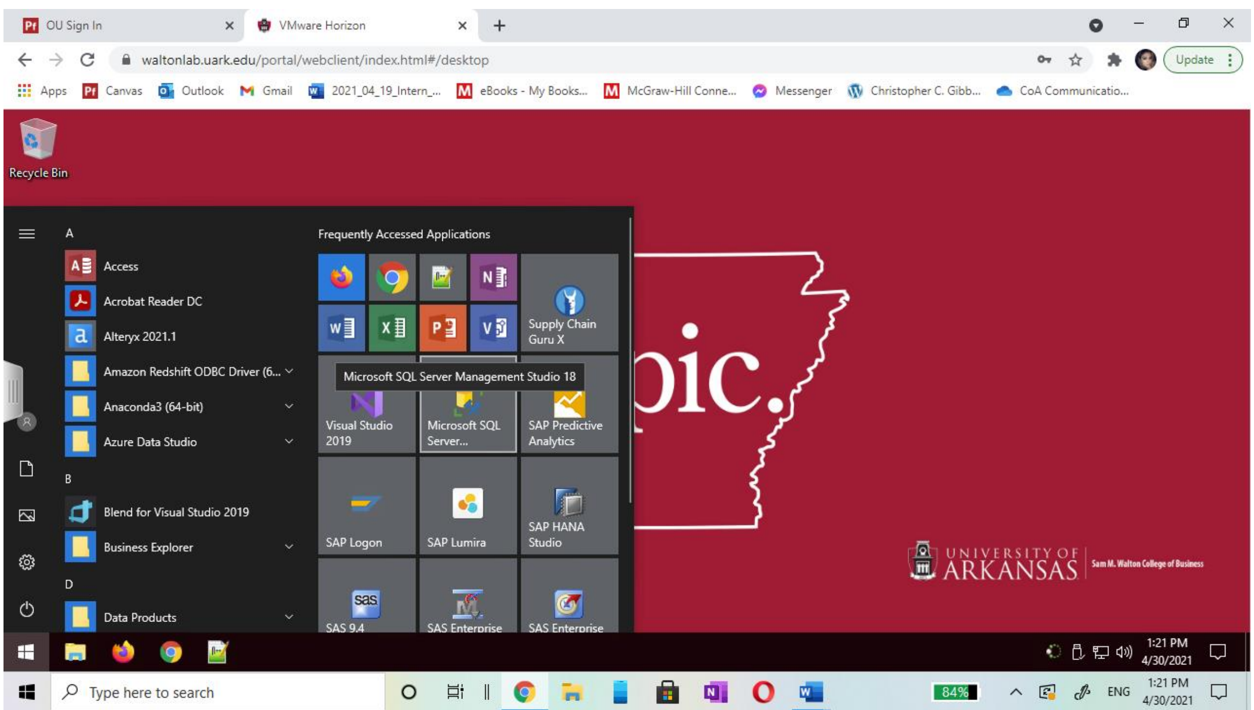
We have listed the steps along with screenshots to show how to access the database. The database is accessed through a virtual environment called Walton Lab. You will be provided with a personal username and password, where you can access our team database. The database is called Esa782192. Within the database, you can input data and execute queries to show results. All of the requested queries are saved as views, which can be accessed in a folder within the database.



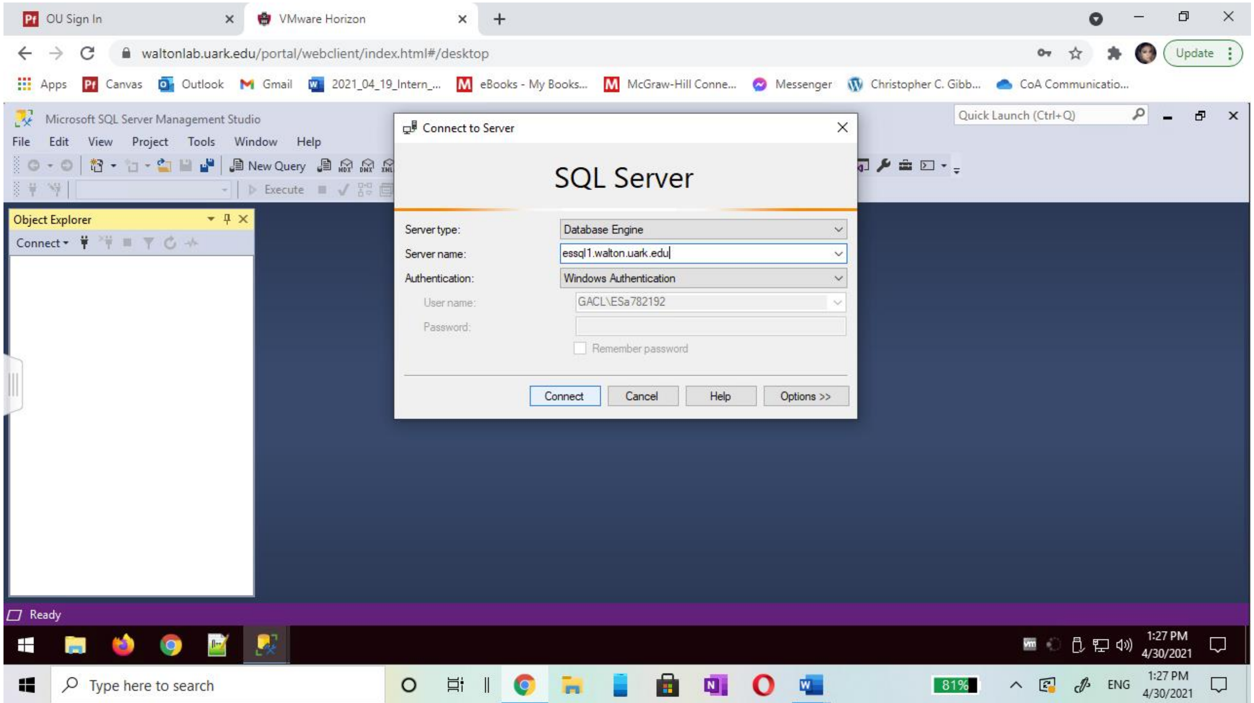
Step 1: Go to waltonlab.uark.edu



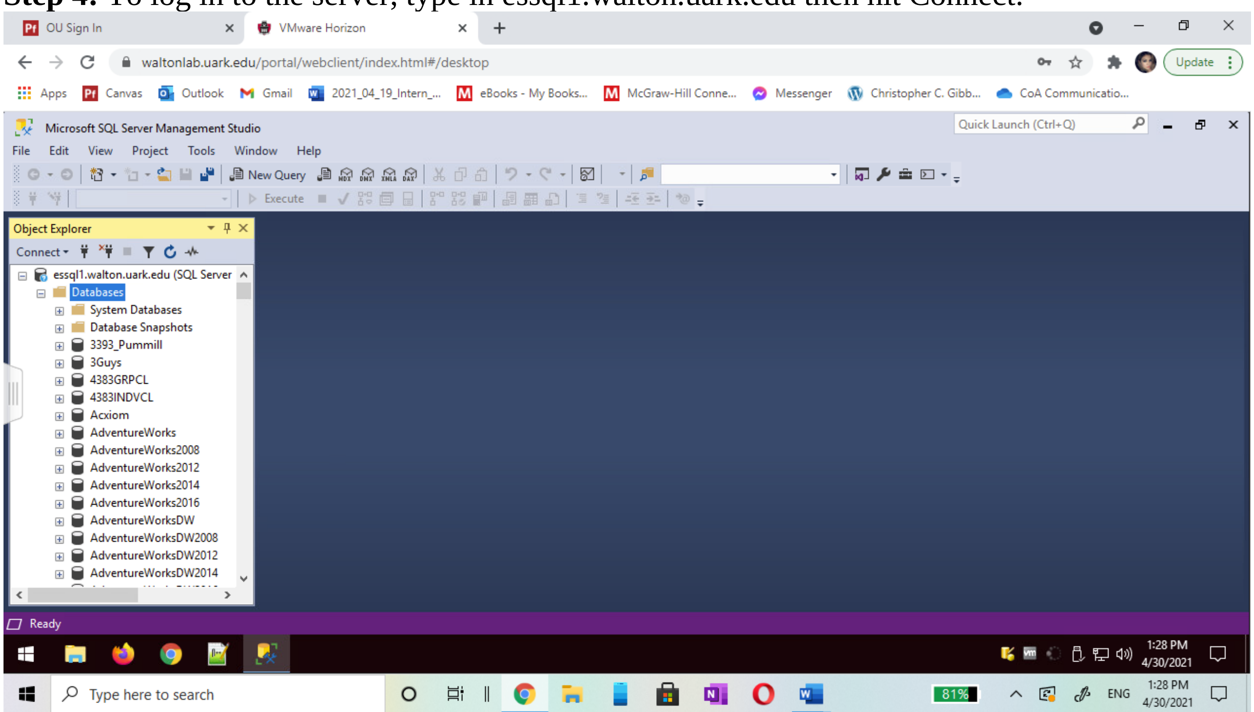
Step 2: Enter in your assigned username and password for the database. Once you are logged in, click Enterprise Systems.



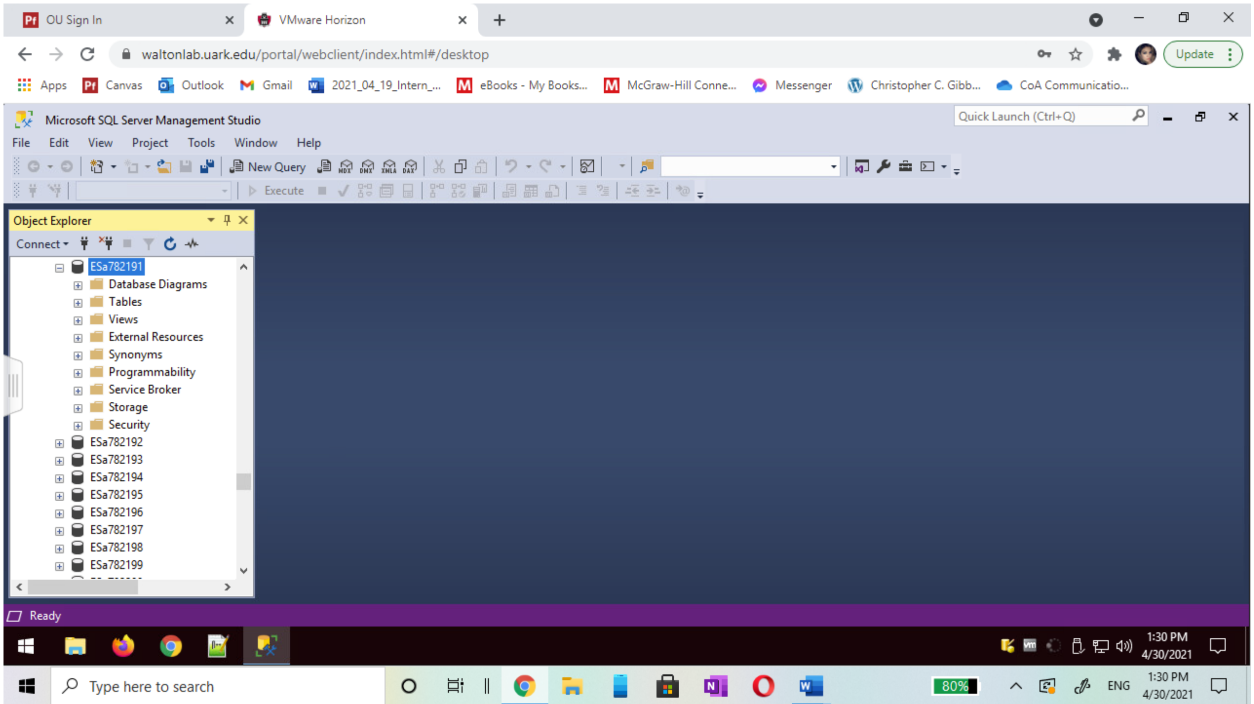
Step 3: Click on the Windows start icon in the bottom left corner *within the virtual environment*. Then, click on Microsoft SQL Server.



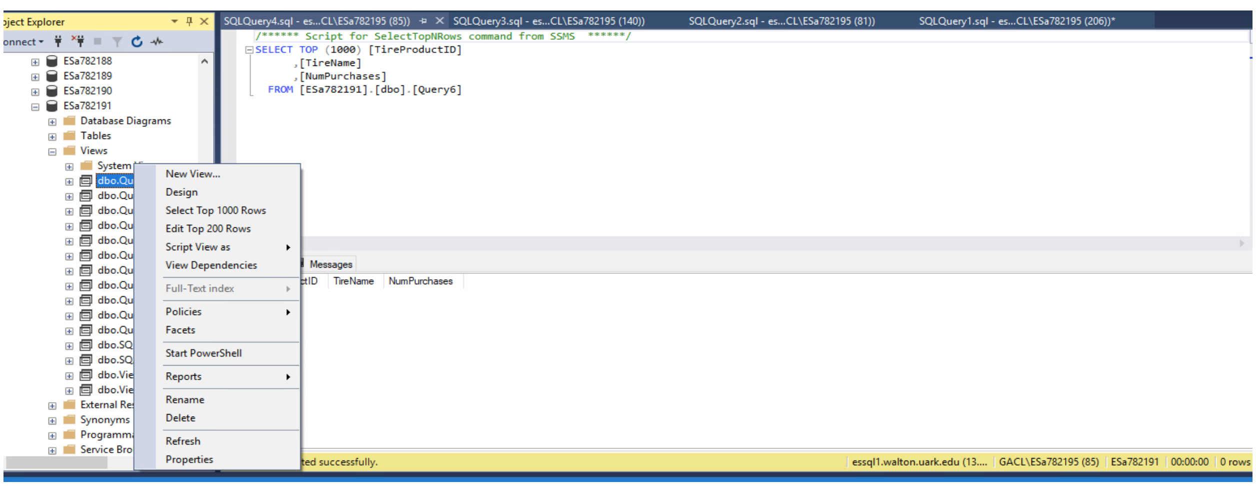
Step 4: To log in to the server, type in essql1.walton.uark.edu then hit Connect.



Step 5: Click the plus sign next to the Databases folder. Immediately after you click this, type ESA on your keyboard.



Step 6: Scroll down until you find our database ESa782191. Click on the plus sign next to ESa782191 to expand it. The folder called "Views" is where the saved queries are located. In the tool bar towards the top, you can write a new query or execute one.



Step 7: To select a stored view to run, right click on the view you would like to see. Choose the "Select Top 1000 Rows" item and check the output box for all the data.

What We Learned Throughout This Process

Member Name:	**What you learned:**
Drake Detar	I learned that it takes a significant amount of time to design and create a database. From speaking with the client, to then implementing data within the database, it’s easy to see why developing a database should be a team effort. With that being said, I learned how to find the needs of the customer to create a database that they can use in their day-to-day operations to run more efficiently.
Dom Demas	I learned that there are so many little things that you may not see at first sight, and they can mess up all the work you have done so far. Therefore, it is good to work as a team so other people might be able to see the small errors that you are not able to find.
Jonny Domingo	I learned that it’s quite impressive how much thought and structure goes behind just a single company’s database. Something as simple as a company called Sonner Tires (though unrealistic) has such a complex system navigating and helping them find sales and such. I can’t imagine how complex databases are for say Google, or one of the top dogs.
Jackson Dedrick	I learned that when it comes to group projects, communication is greatly important. It was very important in this project because there are so many different elements that go into he final product and whenever one person would change something, it affected other pieces, and communication allows all the members to stay on the same page.
Kali Curtis	I learned that creating a database is a continuous cycle. You consistently have to go back and fix your ERD, your normalizations (say, if you want to denormalize something), your physical implementation of the database, and even your queries. Overall, I’ve learned about how all of the processes work together and lead to the final product of an actual database.

As a team we learned how to address the needs of a client and design and implement a database that fit those needs. We also learned how important each step of the process is of building the database. This is because each phase builds on each other and one small mistake can cost you a lot of time down the road.

Appendix

Team Contract

We have provided a screenshot of our team contract below. It includes our names, emails, phone numbers, strengths, and our availability to meet. Then, as a group we decided what our expectations for each other would be for the assignment. These expectations will be used as a guideline for our confidential peer evaluations. Lastly, we will deduct points from the peer evaluations for not having our work done and for not attending team meetings.

Team Members

Name	Email	Phone	Strengths	Availability to Meet
Jonathon Domingo	jonnyd@ou.edu	405-549-0383	Teamwork, hard worker, motivator	Available other than mon/wed nights and tues/thurs 5:30-7
Drake Detar	Drakedetar33@ou.edu	2148865753	Problem Solver, efficient, reliable	Anytime except between 3-6 on Mon/Wed and 4-8:30 Tues/Thurs
Jackson Dedrick	oudedrick@ou.edu	405-837-2502	Logical, reliable, problem solving	Everyday of the week after 6:45. Mon-Wed btw. 11-5
Kali Curtis	Kali.d.curtis-1@ou.edu	405-401-0882	Teamwork, editing and management	Everyday before 9am. More times TBD
Dominic Demas	domonick.p.demas-1@ou.edu	6148004608	Problem Solving, hard working, teamwork	every weekday after 4 and I usually have a match on weekends but if I don't I'll let you guys know.

Team Expectations for the confidential peer evaluation :

- Completing Assignments by the due date
- Being present for team meetings
- Handling your workload so other team members don't have to
- Communicate effectively when issues arise to help solve the problem as quickly as possible.

The behavior for which points will be deducted on the confidential peer evaluation:

- Not having work done by set dates
- Not showing up to team meetings

Data Dictionary Model

A Data dictionary is used to help the developer understand and organize the data within the database. It lists every table in the database as well as the attributes and their data types. The different types of data you will see in this database are varchar(characters), integers, dates, float (decimal number) and time. The different data types tell the developer and the database what kind of data should be implemented into the database for each specific attribute.

Table	Field Name	PK?	Data Type	Size	Null	References (Foreign Key)	Sample
TCustomer	CustomerID	Y	int (auto Increment)		Not Null		1-1000
	CusFirstName	N	varchar	50	Not null		
	CusLastName	N	varchar	50	Not Null		
	CusStreetAddress	N	varchar	100	Not Null		
	CusZipCode	N	varchar	5	Not Null		
	CusCity	N	varchar	50	not Null		
	CusState	N	varchar	2	Not null		
	CusEmail	N	varchar	50	Not Null		
	CusPhone	N	varchar	14	Not Null		
TEmployee	EmpID	Y	int (auto Increment)		Not Null		1-1000
	EmpFirstName	N	varchar	50	Not Null		
	EmpLastName	N	varchar	50	Not Null		
TDiscount	DiscountID	Y	Int (auto Increment)		not null		1-1000
	DDiscountType	N	varchar	2	Not Null		
TPaymentType	PaymentTypeID	Y	int (auto Increment)		Not Null		1-1000
	PTDescription	N	varchar	100	Not Null		
TPaymentIn	PayInID	Y	int (auto Increment)		Not Null		1-1000
	PayEmpID	N	int		Not Null	TEmployee	1-1000
	PayDate	N	date		Not Null		
	PayAmount	N	float		Not Null		
	PayPaymentTypeID	N	int		Not Null	TPaymentType	1-1000
TMake	MakeID	Y	int		Not Null		1-1000
	MMakeName	N	varchar	60	Not Null		
TModel	ModelID	Y	int (auto Increment)		Not Null		1-1000
	MOModelName	N	varchar	60	Not Null		
	MOMakeID	N	int		Not Null	TMake	1-1000
	MOYear	N	int		Not Null		

TVehicle	VehicleID	Y	int (auto Increment)		Not Null		
	VEHCustomerID	N	int		Not Null	TCustomer	1-1000
	VEHMakeID	N	int		Not Null	TMake	1-1000
	VEHVINNumber	N	varchar	17	Not Null		
TSalesOrder	SOID	Y	int (auto Increment)		Not Null		1-1000
	SOPayInID	N	int		Not Null	TPaymentIn	1-1000
	SOCustomerID	N	int		Not Null	TCustomer	1-1000
	SOEmpIDSales	N	int		Not Null	TEmployee	1-1000
	SOEmpIDService	N	int		Not Null	TEmployee	1-1000
	SOVehicleID	N	int		Not Null	TVehicle	1-1000
	SOTechnicianFirstNar	N	varchar		Not Null		
	SOTechnicianLastNan	N	varchar		Not Null		
	SOOrderDate	N	date		Not Null		
	SOMileage	N	int		Not Null		
	SODropOffTime	N	time		Not Null		
	SOPickUpTime	N	time		Not Null		
	SOTimeStartService	N	time		Not Null		
	SOTimeEndService	N	time		Not Null		
	SOisFinancing	N	varchar	1	Not Null		
	SODiscountID	N	int		Null	TDiscount	1-1000
TProduct	ProductID	Y	int (auto Increment)		Not Null		1-1000
	ProductType	N	varchar	6	Not Null		
	SalePrice	N	float		Not Null		
TSalesOrderLine	SalesOrderLineID	Y	int (auto Increment)		Not Null		1-1000
	SOLSalesOrderID	N	int		Not Null	TSalesOrder	1-1000
	SOLProductID	N	int		Not Null	TProduct	1-1000
	SOLSaleOrReturn	N	varchar	1	Not Null		
	SOLQuantity	N	int		Not Null		
	SOLPrice	N	float		Not Null		

TVendor	VendorID	Y	int (auto Increment)		Not Null		1-1000
	VendorName	N	varchar	60	Not Null		
	VenSalesRepFirstNar	N	varchar	60	Not Null		
	VenSalesRepLastNam	N	varchar	60	Not Null		
TTire	TireProductID	Y	int		Not Null	TProduct	1-1000
	TireVendorID	N	int		Not Null	TVendor	1-1000
	TireName	N	varchar	60	Not Null		
	TireDescription	N	varchar	500	Not Null		
	TireMileage	N	int		Not Null		
	TireCost	N	float		Not Null		
	TireQty_OH	N	int		Not Null		
	TireQty_Committed	N	int		Not Null		
	TireReorderPoint	N	int		Not Null		
TTireModel	TireModelID	Y	int (auto Increment)		Not Null		1-1000
	TMModelID	N	int		Not Null	TModel	1-1000
	TMProductID	N	int		Not Null	TTire	1-1000
TService	ServProductID	Y	int		Not Null	TProduct	1-1000
	LifeTimeProtection	N	varchar	1	Not Null		
TPurchaseOrder	POID	Y	int (auto Increment)		Not Null		1-1000
	POEmpID	N	int		Not Null	TEmployee	1-1000
	PODatePaid	N	date		Not Null		
	POAmountPaid	N	date		Not Null		
TPurchaseOrderLine	POLID	Y	int (auto Increment)		Not Null		1-1000
	POLPOID	N	int		Not Null	TPurchaseOrder	1-1000
	POLProductID	N	int		Not Null	TProduct	1-1000
	POLQuantity	N	int		Not Null		
	POLPrice	N	float		Not Null		

Project Management

The total time spent on the project added up to a total of 785 minutes (~13 hours), making the cost of the project $19,625. Milestone 1 had the most amount of time put into it, while the last submission had the least. Below are screenshots of the different tasks for milestone 1 and the time it took to complete as well as one for the final submission. Most of the estimations for planned minutes were accurate within about 5 minutes. The longest task was inputting data into the database during milestone 3. This is because we kept running into errors and had to figure out what the issue was. It was also very tedious implementing all the different tables which added more time.

Project Start Date	3/15/2021		Project End Date	5/2/2021		Cost (per 60 min)	$25
Final Submission							
	Drake		100	60	60	0	
	Domonick		100	30	35	0	
	Kali		100	60	60	0	
	Jackson		100	30	45		
	Jonathon		100	30	40	0	
Sub Total						0	0 $0
Read Case + Prepare Questions for client	Drake		100	20	25	-5	
	Domonick		100	25	25	0	
	Jackson		100	30	20	10	
	Kali		100	20	15	5	
	Jonathon		100	30	30	0	
Client Meeting	Drake		100	15	15	0	
	Domonick		100	15	15	0	
	Jackson		100	15	15	0	
	Kali		100	15	15	0	
	Jonathon		100	15	15	0	
ERD Design	Drake		100	60	40	20	
	Domonick		100	15	15	0	
	Jackson		100	120	90	30	
	Kali		100	60	50	10	
	Jonathon		100	50	55	-5	
Assumptions	Drake		100	5	5	0	
	Domonick		100	10	10	0	
	Jackson		100	10	10	0	
	Kali		100	10	10	0	
	Jonathon		100	5	5	0	
Write-up preparation	Drake		100	10	10	0	
	Domonick		100	50	50	0	
	Jackson		100	30	30	0	
	Kali		100	120	120	0	