University of Oklahoma

The Lightning Bugz

“Data Solutions So Bright, You’ll Bug Out”

Haley Begala
Wenjie Du
Matthew Harrison
Harris Jones
Amy Stall

MIS 3353 – Database Management

Dr. Durcikova

March 12, 2019

Executive Summary

The Lightning Bugz team is a group of students working to solve database problems and help track employees for Sonner Tire. This unique partnership allows us, as students, to learn and expand our database knowledge, by creating and implementing a functional database for Sonner Tire. As our only client, Sonner Tire has our full attention and we are eager to share our work on the database.

After reviewing the company's requirements for their database, we have created an ERD that allowed us to create the physical design of the database. The ERD consists of the revenue and expenditure cycles. Once the ERD was completed, we were able to normalize the data. The normalization helps in reducing the redundancy of data in the database. Lastly, we have created the physical design of the database and added data for Sonner Tire to use in their growing business. The total for this project came to $2,041.76, for a more intricate breakdown see the project management section of this document.

Contents

Executive Summary 2

Get to Know the Team: The Lightning Bugz 4

Conceptual Design 5

The Client Meeting 5

Q&A During the Meeting & Information We Learned 5

Significant Assumptions 6

What is an ERD? Why is it necessary? 6

Business Cycles Used 6

ERD Created 7

Changes made to generic ERDs 8

Logical Design 12

Normalization 12

Normalized Relations 12

Differences between ERD and Normalized Relations 13

Referential Integrity 13

Physical Design and Implementation 14

Data Dictionary 14

Denormalization 14

Implemented Physical Design 15

Challenges Faced/Addressed During Implementation 16

Strengths and Weaknesses Encountered During Implementation 16

Specific SQL Statements Requested 17

Three Additional Queries 21

User Documentation 24

What We Learned Throughout This Process 28

Appendix 30

Team Contract 30

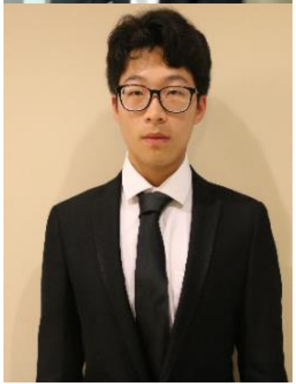
Data Dictionary Model 31

Project Management 37

Get to Know the Team: The Lightning Bugz

Haley Begala

Haley is currently a senior and is graduating in May 2019 with a degree in Business Analytics. She has accepted a full-time job with Protiviti, a business consulting firm. Haley has started to specialize in Healthcare Data Management and Information.

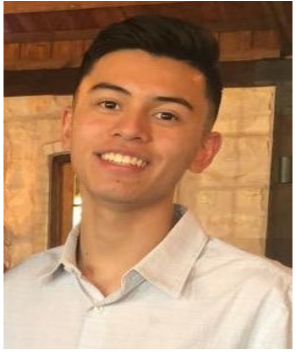


Wenjie Du

Wenjie is a junior and is planning to graduate in December 2020 with a bachelor's degree. This summer, he is going to enroll in Top 20 University to pursue a master's degree for MIS and look for an internship in Shanghai, China.

Matthew Harrison

Matthew is a junior and is planning to graduate with a Masters in Management Information Technology in December 2020. He works as an investment research intern at Plangroup Financial, an investment firm in Oklahoma City.



Harris Jones

Harris is a junior, pursing a major in Management information Systems and set to graduate in May 2020. He currently works as a server at Charleston's. During the summer, Harris plans to shadow for Metro Tech to further enhance his cranium.

Amy Stall

Amy is currently a senior, graduating in December of 2019. She is working towards a degree in Management Information Systems and a minor in Architectural Studies. This summer she is working as an Information Technology Intern at Bank of Oklahoma Financial in her hometown, Tulsa, Oklahoma.

Conceptual Design

In this section, we will cover the client meeting details, how the ERD was created, and which business cycles are including in the ERD. We will also describe the changes made to the generic cycles for Sonner Tire. The first client meeting occurred on 2/27/19 at 2:45pm in AH 7i, everyone on the team interviewed Helen Anderson, Co-owner of Sonner Tire. During this interview the team gathered information from questions developed from the Sonner Tire Requirements Document. These questions and answers are located in the Q&A section of this document.

The Client Meeting

This section contains information received during the client meeting. Helen Anderson, Co-owner of Sonner Tire, met with the team to answer questions developed form the Sonner Tire Requirements Document. This meeting lasted from 2:45-3pm on Tuesday 27th, 2019. All team members were present and assisted in the interview of Helen Anderson.

- Meeting Time: 2/27/19 at 2:45pm
- Location: AH 7i
- Interviewers: Haley Begala, Wenjie Du, Matthew Harrison, Harris Jones, Amy Stall
- Interviewee: Helen Anderson

Q&A During the Meeting & Information We Learned

Question 1: Are logging sales and purchases manual or automatic?
Answer: It is manual entry. We will enter billing address and record the services they are ordering (new tires, insurance for certain wheels, which tires purchased). We will also check what time they came and left (start date and time), know where the customer is waiting, and what time it is done, the manager makes decisions on what time a car is done.

Question 2: Do you notice any user-error?
Answer: Yes, we just type entries into excel and it is getting difficult.

Question 3: Is there segregation of duties during this process? (i.e. an employee dedicated to taking invoices and entering into the system)
Answer: The manager assigns which technician gets each job.

Question 4: Is the collective agency effective?
Answer: It depends.

Question 5: Is there certain mechanics that only do certain jobs?
Answer: Sometimes, it matters which mechanic takes which job. Bonuses are given depending on how fast people work. Young boys and girls can come in, learn a skill, and make some money. An apprentice can be assigned to somebody and it can be a little slower. Some mechanics are good, and some are not good with kids. However, customers cannot make requests upon mechanics.

Question 6: What sort of system or user interface do you use?
Answer: Excel. I am unsure of where the errors are.

Question 7: What is your budget for the project?
Answer: Free

Significant Assumptions

When creating the ERD for Sonner Tire, we have made 5 important assumptions that alter the ERD. These assumptions have been made based off the information in the Sonner Tire Requirements Document that we used to create the ERD. The assumptions are made because some of the information given by the company isn't specific, so we made the best assumption that we could from our knowledge of creating ERD's.

1. A customer can have several cars on file.
2. Installing, repairing, rotating, and servicing are all separate things that can be done, and any combination of them can be part of a sales order.
3. Manager says when the car is done.
4. Apprentice mechanics can only work on one car at a time.
5. Each employee has their own commission rate.
6. Insure all of the tires purchased, if insurance is purchased
7. Only one employee works on each car, they can perform multiple ProductTypes

What is an ERD? Why is it necessary?

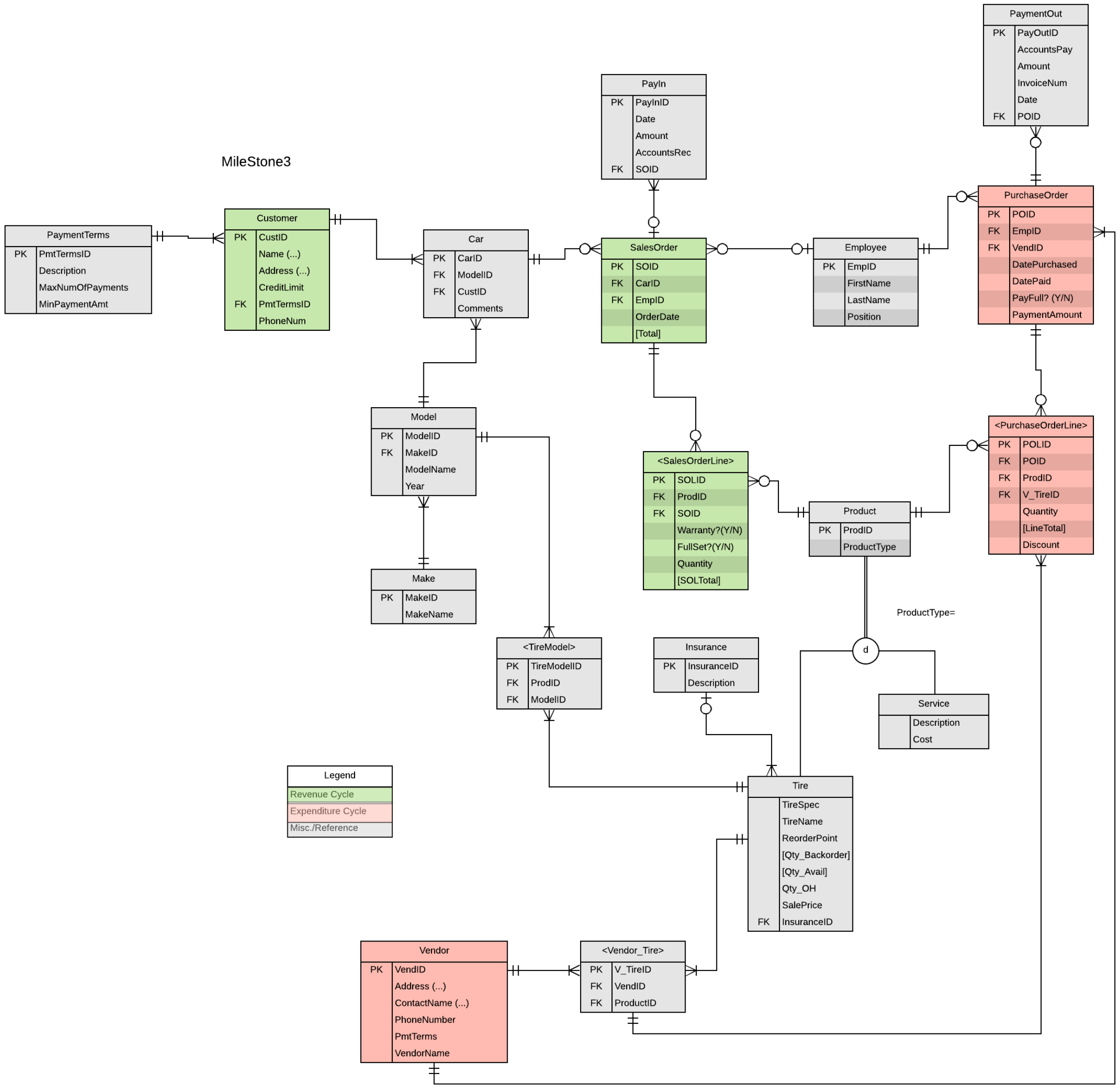
An Entity Relationship Diagram (ERD) is a representation of an information system that shows the relationships among entities such as, people, objects, and events. A database is a structured set of data stored on a computer and allows for easy electronic access of the data within. ERDs are used to help define business processes and often the foundation for database design. Sonner Tire will benefit from an ERD because they have several processes within their company. The ERD will act as a guide when exploring how the processes relate to one another.

Business Cycles Used

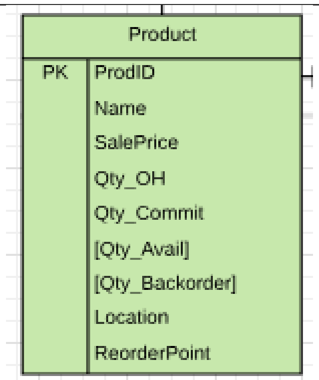
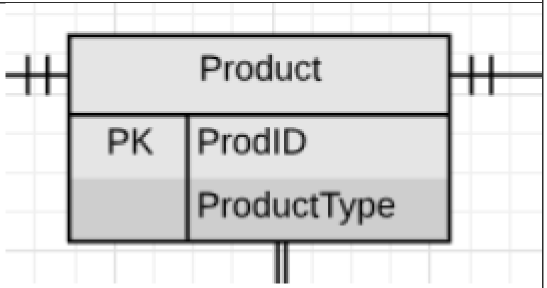
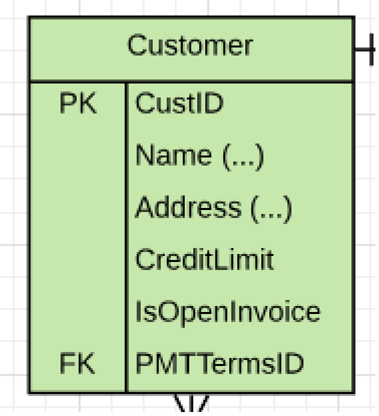
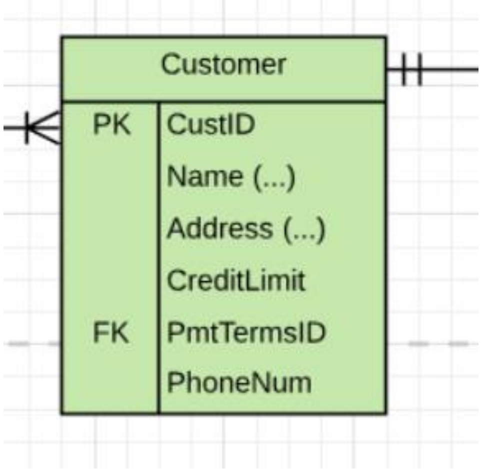
We have used the revenue and expenditure cycles in the ERD for Sonner Tire. The revenue cycle is used because Sonner Tire is selling new tires, repairs, and rotating services. When a company is producing revenue, the revenue cycle is important to include the customer information and the sales order information. The expenditure cycle is used because Sonner Tire purchases orders to receive new tires. The expenditure cycle is important to produce invoices for the company.

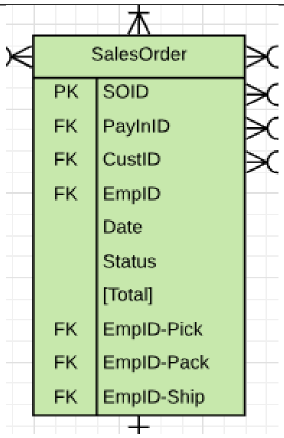
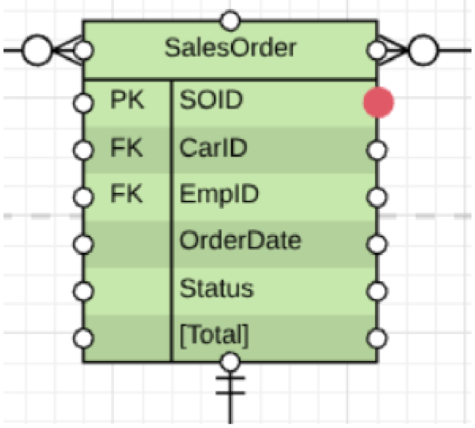
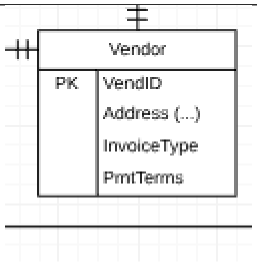
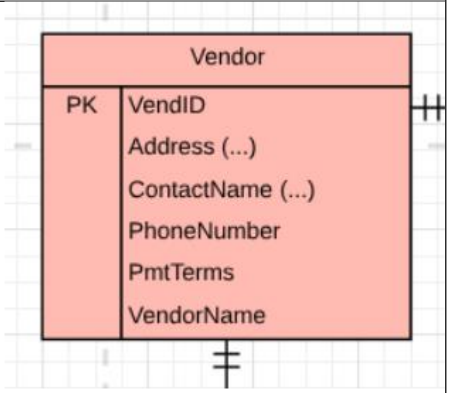
ERD Created

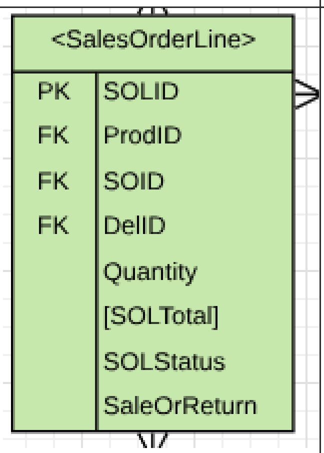
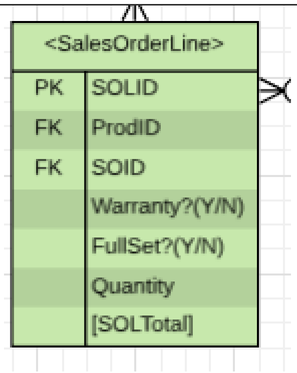
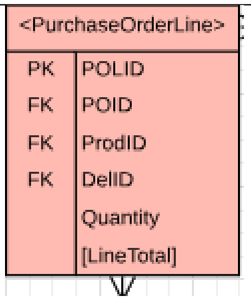
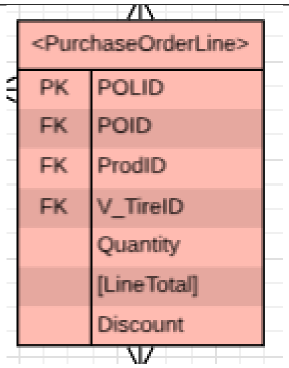
Below is the ERD for Sonner Tire. This ERD should consider all the requirements from Sonner Tire. The ERD is created to produce a database that the employees and managers of the company can use to record purchases and sales by Sonner Tire. The ERD includes the revenue and expenditure cycles.

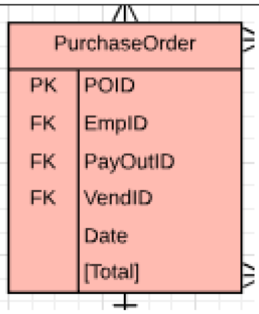
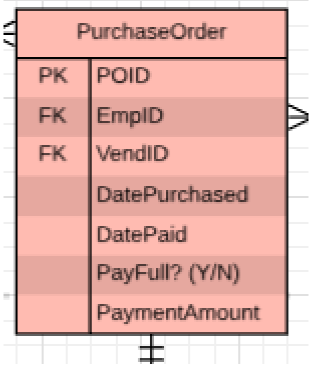
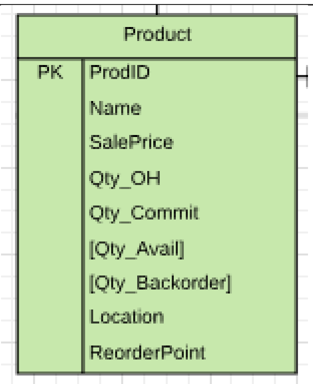
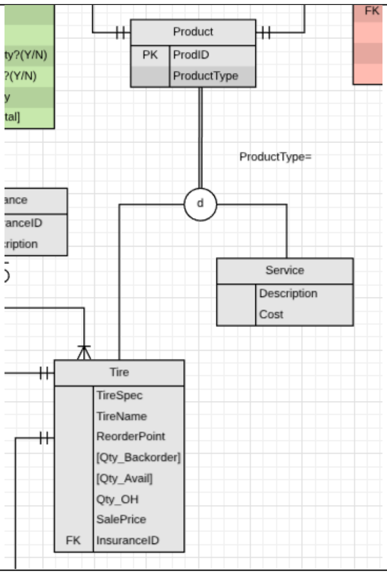


Changes made to generic ERDs

Change Description	Original ERD	Updated ERD
We changed the Product entity to be a super-type for Tire and Service.	 Product PK ProdID Name SalePrice Qty_OH Qty_Commit [Qty_Avail] [Qty_Backorder] Location ReorderPoint	
We also made several changes to the Customer entity. On top of normal information needed, Sonner Tire would like to keep track of customer payments with greater detail. For this, we have added fields to track whether a payment is overdue, how long they have to make the payment, and a running balance on a customer's account.	 Customer PK CustID Name (...) Address (...) CreditLimit IsOpenInvoice FK PMTTermsID	 Customer PK CustID Name (...) Address (...) CreditLimit FK PmtTermsID PhoneNum

The client sells tires in quantities of 1-4, so it is important that the ‘SalesOrder’ table include whether a full set has been sold or not. There are likely to be multiple mechanics working on a car, so we added multiple employees to each Sales Order, so that the client can keep track of which employee performs each task. This helps with quality assurance and aids in managing the employees time.	 SalesOrder PK SOID FK PayInID FK CustID FK EmpID Date Status [Total] FK EmpID-Pick FK EmpID-Pack FK EmpID-Ship	 SalesOrder PK SOID FK CarID FK EmpID OrderDate Status [Total]
The client mentioned having trouble managing invoices and not paying them multiple times. We decided to build on the ‘Vendor’ table to help the client organize their payments to the vendors. For easier access and organized contact information, we added a ‘ContactName’ and ‘PhoneNumber’ attribute to the	 Vendor PK VendID Address (...) InvoiceType PmtTerms	 Vendor PK VendID Address (...) ContactName (...) PhoneNumber PmtTerms VendorName

vendor table so the client can be consistent in the Sales Representative they talk to and build those relationships.		
The associative entity, SalesOrderLine was changed to make sure everything SonnerTire was asking for was there. They don't have a delivery so that foreign key was taken out. They are worried about the warranty and whether or not the customer bought a full set so those attributes were added. Also a customer cannot return a tire or service so SaleOrReturn was taken out.	 <SalesOrderLine> PK SOLID FK ProdID FK SOID FK DelID Quantity [SOLTotal] SOLStatus SaleOrReturn	 <SalesOrderLine> PK SOLID FK ProdID FK SOID Warranty?(Y/N) FullSet?(Y/N) Quantity [SOLTotal]
The associative entity, PurchaseOrderLine, was changed by taking out the delivery foreign key because SonnerTire doesn't deliver. A discount was added, because SonnerTire gets	 <PurchaseOrderLine> PK POLID FK POID FK ProdID FK DelID Quantity [LineTotal]	 <PurchaseOrderLine> PK POLID FK POID FK ProdID FK V_TireID Quantity [LineTotal] Discount

discounts for large purchases.		
The product entity was changed by taking out some of the attributes that were not needed by SonnerTire. PayOutID foreign key was removed. PayFull was added because this SonnerTire needed to know if they had paid the invoice or not. Date was split into DatePurchased and DatePaid because these can be different. And PaymentAmount was added to keep track of the amount SonnerTire paid at a certain time.	 PurchaseOrder PK POID FK EmpID FK PayOutID FK VendID Date [Total]	 PurchaseOrder PK POID FK EmpID FK VendID DatePurchased DatePaid PayFull? (Y/N) PaymentAmount
A sub-type/super-type was created for Product. SonnerTire sells two different products, Service and Tire. Each of these uses the attribute ProdID, but they have many different attributes that are located in their sub-types.	 Product PK ProdID Name SalePrice Qty_OH Qty_Commit [Qty_Avail] [Qty_Backorder] Location ReorderPoint	 Product PK ProdID ProductType ProductType= d Service Description Cost Tire TireSpec TireName ReorderPoint [Qty_Backorder] [Qty_Avail] Qty_OH SalePrice FK InsuranceID ty?(Y/N) ?(Y/N) y tal] ance anceID ription FK

Logical Design

In this section, we will discuss and define the logical design used in transforming the conceptual design into stable database structures that can be used into a data base management system. The conceptual entities, attributes, and instances will be turned into relations, columns, and rows. Most importantly, in this step we have normalized the data. Normalizing the data takes away repetitive data and will help in the physical design of the database.

Normalization

Normalization is a data analysis technique used when creating a database system. To normalize a database, data is split into schema, or related components. We normalize data to create a cleaner, less repetitive, data set. This ensures that we prevent errors when the databse is updated. The normalization is in 3NF because the table is in 2NF and all of the columns are non-transitively dependent on the primary key. This means that each column's value relies on another column through a second column.

Normalized Relations

TState(StateCode, StateName)

TZip(ZipCode, City, ZStateCode)

Foreign Key ZStateCode references TState, not null on delete restrict

TMake(MakeID, MakeName)

TPaymentTerms(PmtTermsID, PTDescription, PTMaxNumOfPayments, PTMinPaymentAmt)

TInsurance(InsuranceID, IDescription)

TProduct(ProductID, PProductType)

TEmployee(EmpID, EmpFName, EmpLName, EmpPosition)

TService(ServiceProdID, SDescription, SCost)

TCar(CarID, CarModelID, CarCustID, CarComments)

Foreign Key CarCustID references TCustomer not null on delete restrict

Foreign Key CarModelID references Tmodel not null on delete restrict

TModel(ModelID, ModMakeID, ModName, ModYear)

Foreign Key ModMakeID references TMake, not null on delete restrict

TSalesOrder(SOID, SOCarID, SOEmpID, SOOrderDate)

Foreign Key SOCarID references TCar, not null on delete restrict

Foreign Key SOEmpID references TEmployee, null allowed on delete set null

TPayIn(PayInID, PISOID, PIAccountsRec, PIDate, PIAmount)

Foreign Key PISOID references TSalesOrder, null allowed on delete set null

TSalesOrderLine(SOLID, SOLProdID, SOLSOID, SOLWarranty, SOLFullSet, SOLQuantity)

Foreign Key SOLProdID references TProduct, not null on delete restrict

Foreign Key SOLSOID references TSalesOrder, not null on delete restrict

TPurchaseOrder(POID, POEmpID, POVendID, PODatePurchased, PODatePaid, POPayFull, POPaymentAmount)

Foreign Key POEmpID references TEmployee, not null on delete restrict

Foreign Key POVendID references TVendor, not null on delete restrict

TPaymentOut(PayOutID, POutPOID, POutAccountsPay, POutAmount, POutInvoiceNum, POutDate)

Foreign Key POutPOID references TPurchaseOrder, null allowed on delete restrict

TPurchaseOrderLine(POLID, POLPOID, POLProdID, POLV_TireID, POLQuantity, POLDiscount)

Foreign Key POLPOID references TPurchaseOrder, null allowed on delete set null

Foreign Key POLProdID references TProduct, null allowed on delete set null

Foreign Key POLV_TireID references TV_Tire, not null on delete restrict

TTireModel(TireModelID, TMProdID, TMModelID)

Foreign Key TMProdID references TProduct not null on delete restrict

Foreign Key TMModelID references TModel not null on delete restrict

TCustomer (CustID, CFirstName, CLastName, CStreet, CZipCode, CState, CCreditLimit, CPmtTermsID, CPhoneNum)

Foreign Key CZipCode references TZip, not null on delete restrict

Foreign Key CPmtTermsID references TPaymentTerms, not null on delete restrict

Foreign Key CState references TState, not null on delete restrict

TTire(TireProdID, TTireName, TReorderPoint, TQtyOH, TSalePrice, TInsuranceID)

Foreign key TInsuranceID references TInsurance, null allowed on delete set null

TVendor_Tire(V_TireID, VTVendID, VTProductID)

Foreign key VTVendID references TVendor, not null on delete restrict

Foreign key VTProductID references TProduct, not null on delete restrict

TVendor(VendID, VStreet, VZipCode, VState, VContactFirstName, VContactLastName, VPhoneNumber, VPmtTerms, VVendorName)

Foreign Key VZipCode references TZip, not null on delete restrict

Foreign key VState references TState, not null on delete restrict

Differences between ERD and Normalized Relations

In the normalization we add TState and TZip. These are both required to normalize the composite attribute Address(..). Because of this, TCustomer and TVendor both had the foreign key ZipCode added. Also, for each of the attributes in the ERD, a nickname for the table was added before them in the normalization. Adding the nickname ensures that there won't be any repetitions for the names in the normalized data. Also, all of the derived attributes were taken out for the normalization. This is because those attributes are to help the client understand where that data is coming from, but they don't serve a purpose in the database.

Referential Integrity

A referential integrity refers to the accuracy and consistency of data within a relationship. A relationship exists between two entities in an ERD. The referential integrity requires that, if a foreign key is used in the entity, then it must reference a valid primary key in another entity. For example, Sonner Tire has customers who must have at least one car, but each car only belongs to one customer. Therefore, the customer donates its primary key to the entity car. The foreign key in car, CustomerID, is a valid foreign key because it is representing a valid primary key in the customer entity. Some fields in this database are essential. Fields could not be deleted without ruining data. For instance, a car would never not have a "car make". Constraints are implemented to make sure data that is essential cannot be deleted by a user.

Physical Design and Implementation

Data Dictionary

A Data dictionary is a set of information describing the contents, format, and structure of a database and the relationship between its elements. It's used as a glossary reference to define all different data objects in the database. Typical fields included in a data dictionary are column names, data types, formats, and any restraints that go along with a field. By having all this information readily available, anyone who has technical skills can reference, change, and maintain the database.

Data Dictionary

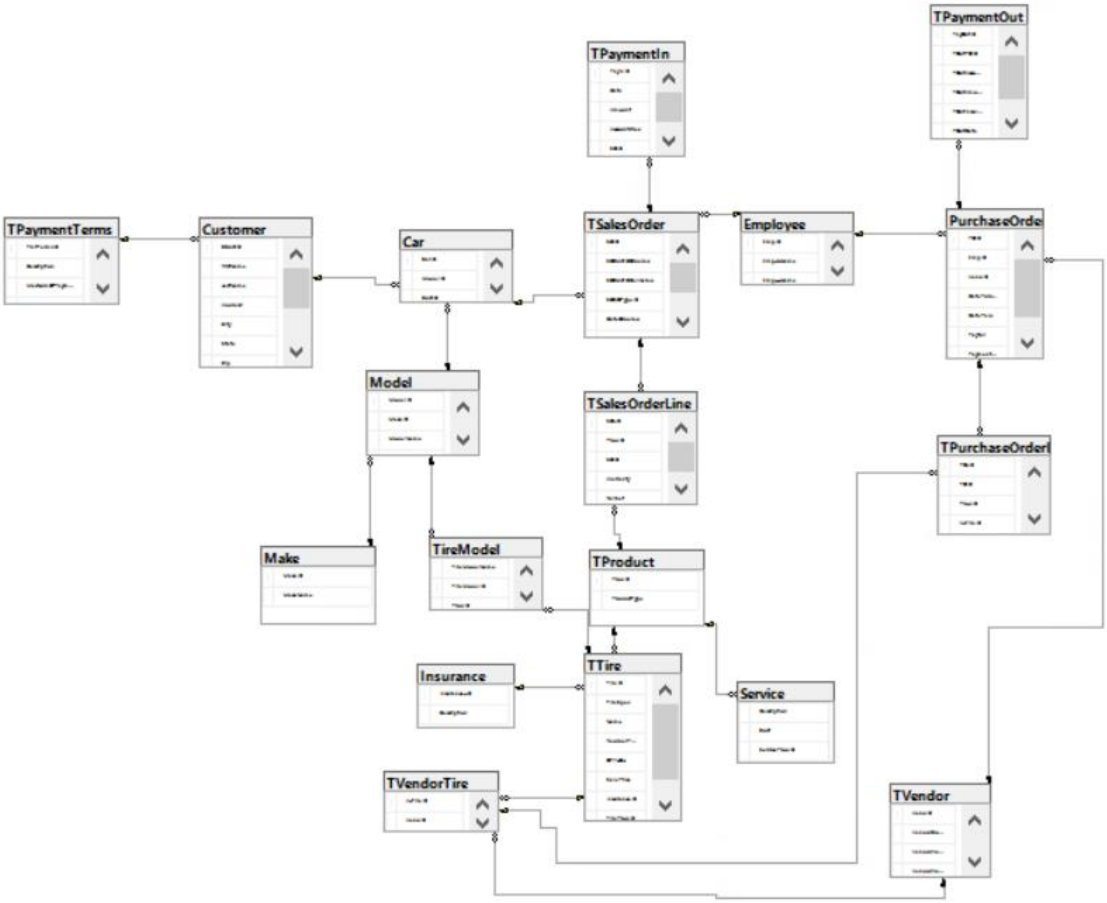
Column Name	Data Type	Data Format	Field Size	Constraints	Example
CustID	Integer	NNNN	4	Not null	1234
Firstname	nVarchar		50	Not null	Lindsey
Lastname	nVarchar		50	Not null	Lohan
Cstreet	nVarchar		50	Not null	LLRocks, Inc. 1749 Old Mill Rd
CZipCode	nVarchar		50	Not null	11566
CCreditLimit	Money		8		$500.50
CPmtTerms	nVarchar		50		Ex. 1/10,n/30

Denormalization

Denormalization is the process of simplifying the tables within a database, usually by reducing the number of tables in the database. There are problems that might occur with this if not done carefully, such as duplicated data and reduced atomicity, making the database more complicated and harder to work with efficiently. But when done correctly, it allows the database to process data faster.

One point where we engaged in denormalization within our database is when we deliberately left out reference tables for the zip codes, states, and other address fields for our customers' data. By leaving out these reference tables, we opened up the possibility of having small instances of data duplication. This not only simplified the process of creating the database, but also increased processing times by leaving out these extra tables. In the real functioning database, we would insert these reference tables for zip code, state, country, and any other geographic data with a set number of possibilities.

Implemented Physical Design



Challenges Faced/Addressed During Implementation

1. **Coordinating our work in the database**

   With five people all working on the same database simultaneously, everyone's affected each other. We had to systematically assign each person specific tasks and make sure no one was repeating work or hindering someone else.

2. **Keeping track of small details and typos**

   When one small detail was changed, such as a data type or an entity name, it would subsequently affect all the queries and other tables associated with it. Once again, we had to agree on a system to make sure everyone was on the same page.

Strengths and Weaknesses Encountered During Implementation

Strengths:

1. We worked together very well as a team. We never encountered any major disagreement or had any problems getting along with each other personally.
2. We had the technical skills. With all our different backgrounds and areas of expertise, we were able to tackle every problem we encountered.

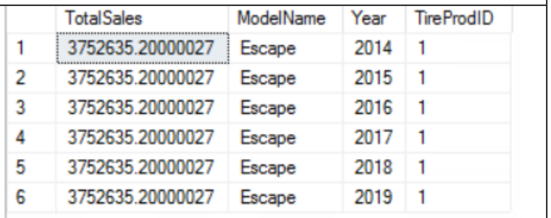
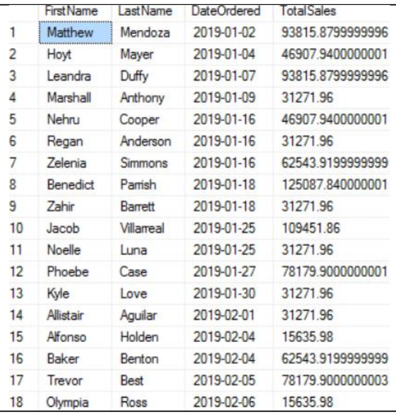
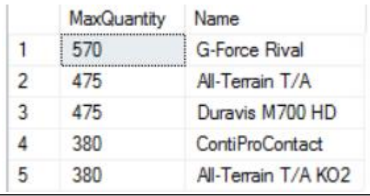
Weaknesses:

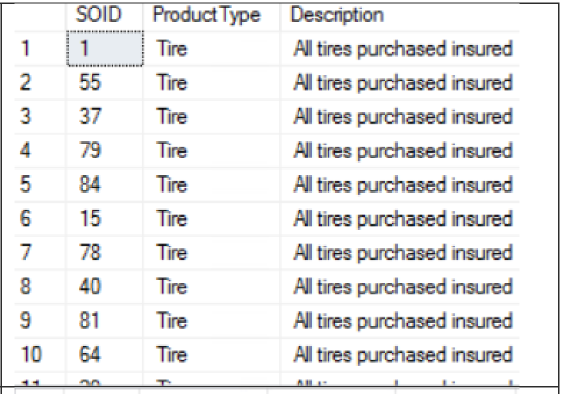
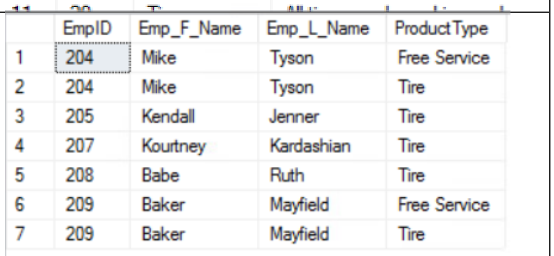
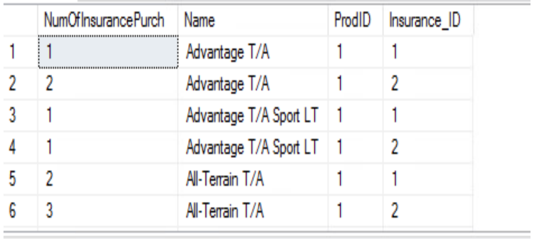
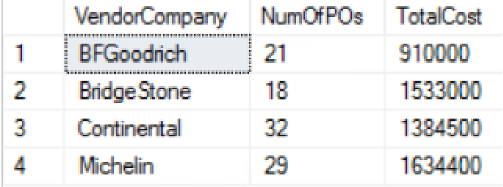
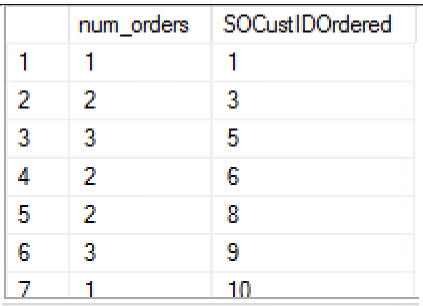
1. Coordinating our time was a major weakness. With all of us on vastly different schedules, it was difficult to get everyone to meet at the same time.
2. We were not prepared for technical difficulties. While we were prepared to deal with the actual work, we were not prepared for internet difficulties, crashing computers, and inconsistent file storage, all of which hindered our progress at some point.

Specific SQL Statements Requested

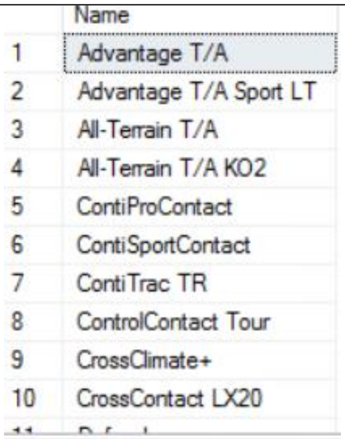
In this section, we have included the SQL code for the queries SonnerTire has requested. The question is provided in the table, then the SQL, and what the output should be when a user enters the code. We have also provided three additional queries we are suggesting SonnerTire implement in their day-to-day use of the database.

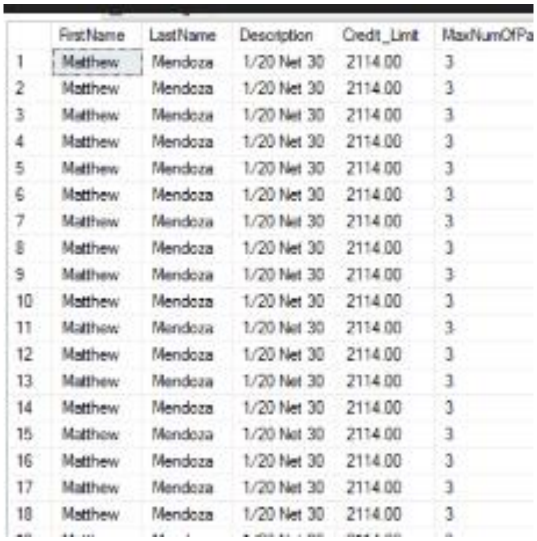
<table>
<tr><th>Query #</th><th>Question</th><th>SQL</th><th>Partial Output</th></tr>
<tr><td>1</td><td>Total sales (in dollars) by region for a given tire manufacturer and car manufacturer. It would be great if we can specify the car model and year too (note that we would like to be able to input the month to be calculated).</td><td>SELECT distinct
SUM(SOL.Quantity*T.SalePrice) AS
TotalSales($), ModelName, M.Year
FROM TSalesOrder SO JOIN
TSalesOrderLine SOL ON
SO.SOID=SOL.SOID
JOIN TProduct P ON
SOL.ProdID=P.ProdID
JOIN Tire T ON P.ProdID=T.TireProdID
JOIN TireModel TM ON
T.TireProdID=TM.TireModelID
JOIN Model M ON
TM.TireModelID=M.TireModelID

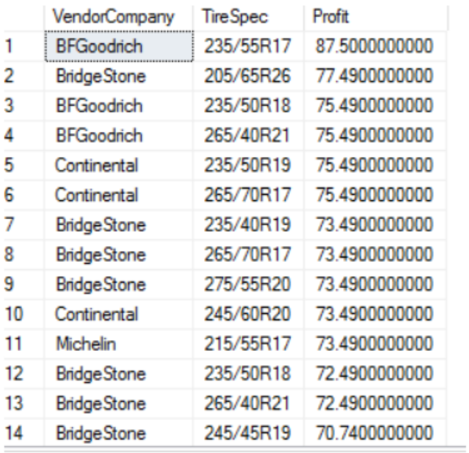
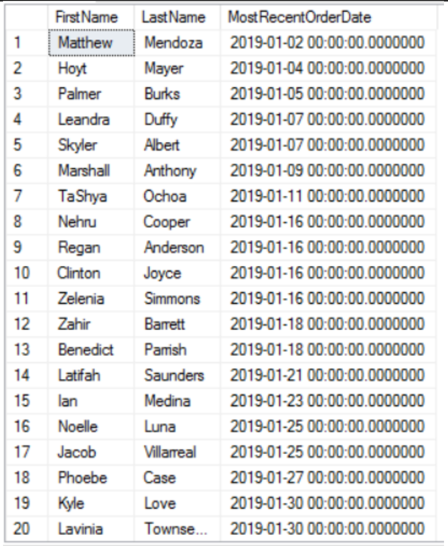
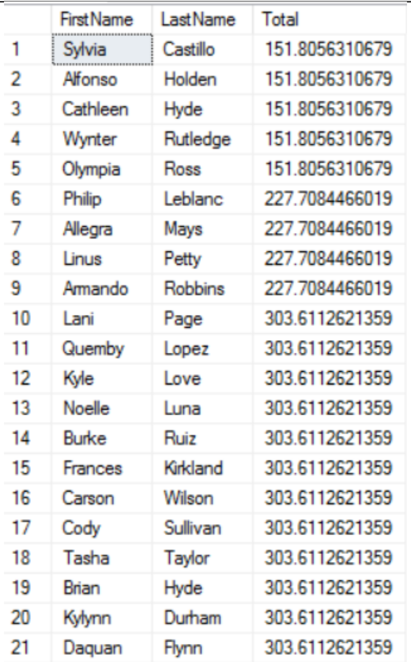
GROUP BY ModelName, M.Year,
T.TireProdID</td><td>
<table>
<tr><th></th><th>TotalSales</th><th>ModelName</th><th>Year</th><th>TireProdID</th></tr>
<tr><td>1</td><td>3752635.20000027</td><td>Escape</td><td>2014</td><td>1</td></tr>
<tr><td>2</td><td>3752635.20000027</td><td>Escape</td><td>2015</td><td>1</td></tr>
<tr><td>3</td><td>3752635.20000027</td><td>Escape</td><td>2016</td><td>1</td></tr>
<tr><td>4</td><td>3752635.20000027</td><td>Escape</td><td>2017</td><td>1</td></tr>
<tr><td>5</td><td>3752635.20000027</td><td>Escape</td><td>2018</td><td>1</td></tr>
<tr><td>6</td><td>3752635.20000027</td><td>Escape</td><td>2019</td><td>1</td></tr>
</table></td></tr>
<tr><td>2</td><td>Total sales (in dollars) by customer in a given year</td><td>SELECT C.FirstName, C.LastName,
SO.DateOrdered,
SUM(Quantity*SalePrice) AS TotalSales
FROM Customer C JOIN Car ON
C.CCustID=Car.CustID
JOIN TSalesOrder SO ON
Car.CarID=SO.CarID
JOIN TSalesOrderLine SOL ON
SO.SOID=SOL.SOID
JOIN TProduct P ON
SOL.ProdID=P.ProdID
JOIN TTire T ON
P.ProdID=T.TireProdID
GROUP BY C.FirstName, C.LastName,
SO.DateOrdered
ORDER BY SO.DateOrdered</td><td>
<table>
<tr><th></th><th>FirstName</th><th>LastName</th><th>DateOrdered</th><th>TotalSales</th></tr>
<tr><td>1</td><td>Matthew</td><td>Mendoza</td><td>2019-01-02</td><td>93815.8799999996</td></tr>
<tr><td>2</td><td>Hoyt</td><td>Mayer</td><td>2019-01-04</td><td>46907.9400000001</td></tr>
<tr><td>3</td><td>Leandra</td><td>Duffy</td><td>2019-01-07</td><td>93815.8799999996</td></tr>
<tr><td>4</td><td>Marshall</td><td>Anthony</td><td>2019-01-09</td><td>31271.96</td></tr>
<tr><td>5</td><td>Nehru</td><td>Cooper</td><td>2019-01-16</td><td>46907.9400000001</td></tr>
<tr><td>6</td><td>Regan</td><td>Anderson</td><td>2019-01-16</td><td>31271.96</td></tr>
<tr><td>7</td><td>Zelenia</td><td>Simmons</td><td>2019-01-16</td><td>62543.9199999999</td></tr>
<tr><td>8</td><td>Benedict</td><td>Parrish</td><td>2019-01-18</td><td>125087.840000001</td></tr>
<tr><td>9</td><td>Zahir</td><td>Barrett</td><td>2019-01-18</td><td>31271.96</td></tr>
<tr><td>10</td><td>Jacob</td><td>Villarreal</td><td>2019-01-25</td><td>109451.86</td></tr>
<tr><td>11</td><td>Noelle</td><td>Luna</td><td>2019-01-25</td><td>31271.96</td></tr>
<tr><td>12</td><td>Phoebe</td><td>Case</td><td>2019-01-27</td><td>78179.9000000001</td></tr>
<tr><td>13</td><td>Kyle</td><td>Love</td><td>2019-01-30</td><td>31271.96</td></tr>
<tr><td>14</td><td>Allistair</td><td>Aguilar</td><td>2019-02-01</td><td>31271.96</td></tr>
<tr><td>15</td><td>Alfonso</td><td>Holden</td><td>2019-02-04</td><td>15635.98</td></tr>
<tr><td>16</td><td>Baker</td><td>Benton</td><td>2019-02-04</td><td>62543.9199999999</td></tr>
<tr><td>17</td><td>Trevor</td><td>Best</td><td>2019-02-05</td><td>78179.9000000003</td></tr>
<tr><td>18</td><td>Olympia</td><td>Ross</td><td>2019-02-06</td><td>15635.98</td></tr>
</table></td></tr>
<tr><td>3</td><td>The five highest selling tires</td><td>SELECT TOP(5) COUNT(SOL.Quantity)
Quantity, T.Name
FROM TSalesOrderLine SOL JOIN
TProduct ON SOL.ProdID=P.ProdID
JOIN TTire T ON
P.ProdID=T.TireProdID
GROUP BY T.Name
ORDER BY MaxQuantity Desc</td><td>
<table>
<tr><th></th><th>MaxQuantity</th><th>Name</th></tr>
<tr><td>1</td><td>570</td><td>G-Force Rival</td></tr>
<tr><td>2</td><td>475</td><td>All-Terrain T/A</td></tr>
<tr><td>3</td><td>475</td><td>Duravis M700 HD</td></tr>
<tr><td>4</td><td>380</td><td>ContiProContact</td></tr>
<tr><td>5</td><td>380</td><td>All-Terrain T/A KO2</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 SO.SOID, ProductType, I.Description
FROM TSalesOrder SO JOIN TSalesOrderLine SOL ON SO.SOID=SOL.SOID
JOIN TProduct P ON SOL.ProdID=P.ProdID
JOIN TTire T ON P.ProdID=T.TireProdID
JOIN Insurance I ON T.InsuranceID=I.Insurance_ID</td>
<td>
<table>
<tr><th></th><th>SOID</th><th>ProductType</th><th>Description</th></tr>
<tr><td>1</td><td>1</td><td>Tire</td><td>All tires purchased insured</td></tr>
<tr><td>2</td><td>55</td><td>Tire</td><td>All tires purchased insured</td></tr>
<tr><td>3</td><td>37</td><td>Tire</td><td>All tires purchased insured</td></tr>
<tr><td>4</td><td>79</td><td>Tire</td><td>All tires purchased insured</td></tr>
<tr><td>5</td><td>84</td><td>Tire</td><td>All tires purchased insured</td></tr>
<tr><td>6</td><td>15</td><td>Tire</td><td>All tires purchased insured</td></tr>
<tr><td>7</td><td>78</td><td>Tire</td><td>All tires purchased insured</td></tr>
<tr><td>8</td><td>40</td><td>Tire</td><td>All tires purchased insured</td></tr>
<tr><td>9</td><td>81</td><td>Tire</td><td>All tires purchased insured</td></tr>
<tr><td>10</td><td>64</td><td>Tire</td><td>All tires purchased insured</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 distinct E.EmpID, Emp_F_Name, Emp_L_Name, ProductType
FROM Employee E JOIN TSalesOrder SO ON E.EmpID=SO.EmpID
JOIN TSalesOrderLine SOL ON SO.SOID=SOL.SOID JOIN TProduct P ON SOL.ProdID=P.ProdID
Missing COUNT(E.SOLID)</td>
<td>
<table>
<tr><th></th><th>EmpID</th><th>Emp_F_Name</th><th>Emp_L_Name</th><th>ProductType</th></tr>
<tr><td>1</td><td>204</td><td>Mike</td><td>Tyson</td><td>Free Service</td></tr>
<tr><td>2</td><td>204</td><td>Mike</td><td>Tyson</td><td>Tire</td></tr>
<tr><td>3</td><td>205</td><td>Kendall</td><td>Jenner</td><td>Tire</td></tr>
<tr><td>4</td><td>207</td><td>Kourtney</td><td>Kardashian</td><td>Tire</td></tr>
<tr><td>5</td><td>208</td><td>Babe</td><td>Ruth</td><td>Tire</td></tr>
<tr><td>6</td><td>209</td><td>Baker</td><td>Mayfield</td><td>Free Service</td></tr>
<tr><td>7</td><td>209</td><td>Baker</td><td>Mayfield</td><td>Tire</td></tr>
</table></td>
</tr>
<tr>
<td>6</td>
<td>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).</td>
<td>Select Count(InsuranceID) as ‘NumOFInsurancePurch’ , Name, ProdID, Insurance_ID
From Insurance I Join TTire T
On I.Insurance_ID = T.InsuranceID
Join TProduct P on P.prodID = T.TireProdID
Group by name, prodid, I.insurance_ID</td>
<td>
<table>
<tr><th></th><th>NumOfInsurancePurch</th><th>Name</th><th>ProdID</th><th>Insurance_ID</th></tr>
<tr><td>1</td><td>1</td><td>Advantage T/A</td><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td><td>Advantage T/A</td><td>1</td><td>2</td></tr>
<tr><td>3</td><td>1</td><td>Advantage T/A Sport LT</td><td>1</td><td>1</td></tr>
<tr><td>4</td><td>1</td><td>Advantage T/A Sport LT</td><td>1</td><td>2</td></tr>
<tr><td>5</td><td>2</td><td>All-Terrain T/A</td><td>1</td><td>1</td></tr>
<tr><td>6</td><td>3</td><td>All-Terrain T/A</td><td>1</td><td>2</td></tr>
</table></td>
</tr>
<tr>
<td>7</td>
<td>The following items for PurchaseOrders: manufacturer name, number of POs, total cost.</td>
<td>SELECT V.VendorCompany, COUNT(PO.POID) NumOfPOs, SUM(POL.Quantity*PO.PaymentAmount) TotalCost
FROM TVendor V
JOIN PurchaseOrder PO ON V.VendID=PO.VendID
JOIN TPurchaseOrderLine POL ON PO.POID=POL.POID
GROUP BY V.VendorCompany</td>
<td>
<table>
<tr><th></th><th>VendorCompany</th><th>NumOfPOs</th><th>TotalCost</th></tr>
<tr><td>1</td><td>BFGoodrich</td><td>21</td><td>910000</td></tr>
<tr><td>2</td><td>BridgeStone</td><td>18</td><td>1533000</td></tr>
<tr><td>3</td><td>Continental</td><td>32</td><td>1384500</td></tr>
<tr><td>4</td><td>Michelin</td><td>29</td><td>1634400</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</td>
<td>Select Count ([TSalesOrder].[SOID]) as num_Orders, [TSalesOrder].[SOCustIDOrdered]
From [ESa195416].[dbo].[TSalesOrder]
Inner join TSalesOrderLine
On TSalesOrder.SOID = TSalesOrderLine.SOID
Inner join TProduct
On TProduct.ProdID=
Where [TSalesOrder].DateOrdered > dateadd(year.- - 1, getdate())
Group by [TsalesOrder].SOCustIDOrdered</td>
<td>
<table>
<tr><th></th><th>num_orders</th><th>SOCustIDOrdered</th></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td><td>3</td></tr>
<tr><td>3</td><td>3</td><td>5</td></tr>
<tr><td>4</td><td>2</td><td>6</td></tr>
<tr><td>5</td><td>2</td><td>8</td></tr>
<tr><td>6</td><td>3</td><td>9</td></tr>
<tr><td>7</td><td>1</td><td>10</td></tr>
</table></td>
</tr>
</table>

	returning customers.		

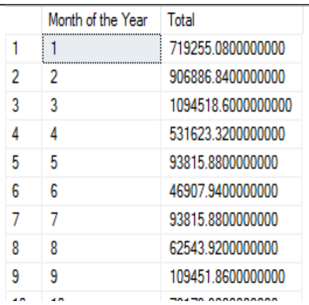
9	List of tires that have not been purchased within the last 6 months (in order to better manage inventory).	SELECT distinct T.Name
		FROM TSalesOrder SO JOIN TSalesOrderLine SOL ON SO.SOID=SOL.SOID
		LEFT JOIN TProduct P ON SOL.ProdID=P.ProdID
		LEFT JOIN TTire T ON P.ProdID=T.TireProdID
		WHERE DateOrdered <= DateAdd(month, -6, getdate())
			
			Name
			1 Advantage T/A
			2 Advantage T/A Sport LT
			3 All-Terrain T/A
			4 All-Terrain T/A KO2
			5 ContiProContact
			6 ContiSportContact
			7 ContiTrac TR
			8 ControlContact Tour
			9 CrossClimate+
			10 CrossContact LX20

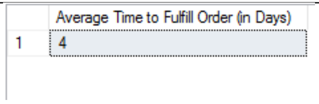
10	Names of customer rs who took advantage of the financing option, date purchased, total amount purchased, credit limit, number of payments made, total amount paid, outstanding amount, is time less than 6 months, all displayed from latest date and then largest amount owed.	SELECT DISTINCT C.FirstName, C.LastName, P.[Description], C.Credit_Limit, P.MaxNumOfPayments , SO.DateOrdered, SUM(PII.Amount) AS PaidTotal,COUNT(PII.PayInID) AS NumberPaymentsMade, SUM(SL.Quantity*T.SalePrice) AS TotalPurchased, (SUM(SL.Quantity*T.SalePrice)-SUM(PII.Amount)) AS AmountOwed,DATEDIFF(Day, DAY(GETDATE()), DAY(SO.DateOrdered)+180) AS DaysLeftToPay
		FROM TPaymentTerms P LEFT JOIN Customer C ON P.PmtTermsID = C.PmtTermsID LEFT JOIN Car Ca ON C.CCustID = Ca.CustID LEFT JOIN TSalesOrder SO ON Ca.CarID =SO.CarID LEFT JOIN TPaymentIn PII ON PII.SOID = So.SOID LEFT JOIN TSalesOrderLine SL ON PII.SOID = SL.SOID LEFT JOIN TProduct PR ON SL.ProdID = PR.ProdID LEFT JOIN TTire T ON PR.ProdID = T.TireProdID
		GROUP BY C.FirstName, C.LastName, P.[Description], C.Credit_Limit, P.MaxNumOfPayments , SO.DateOrdered, SL.Quantity, T.SalePrice, SL.SOLID
		ORDER BY 6, 10 DESC
			
			FirstName LastName Description Credit_Limit MaxNumOfPa
			1 Matthew Mendoza 1/20 Net 30 2114.00 3
			2 Matthew Mendoza 1/20 Net 30 2114.00 3
			3 Matthew Mendoza 1/20 Net 30 2114.00 3
			4 Matthew Mendoza 1/20 Net 30 2114.00 3
			5 Matthew Mendoza 1/20 Net 30 2114.00 3
			6 Matthew Mendoza 1/20 Net 30 2114.00 3
			7 Matthew Mendoza 1/20 Net 30 2114.00 3
			8 Matthew Mendoza 1/20 Net 30 2114.00 3
			9 Matthew Mendoza 1/20 Net 30 2114.00 3
			10 Matthew Mendoza 1/20 Net 30 2114.00 3
			11 Matthew Mendoza 1/20 Net 30 2114.00 3
			12 Matthew Mendoza 1/20 Net 30 2114.00 3
			13 Matthew Mendoza 1/20 Net 30 2114.00 3
			14 Matthew Mendoza 1/20 Net 30 2114.00 3
			15 Matthew Mendoza 1/20 Net 30 2114.00 3
			16 Matthew Mendoza 1/20 Net 30 2114.00 3
			17 Matthew Mendoza 1/20 Net 30 2114.00 3
			18 Matthew Mendoza 1/20 Net 30 2114.00 3

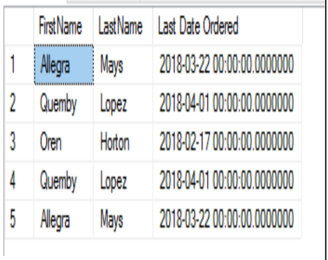
<table>
<tr>
<td>11</td>
<td>Total profit per tire type and manufacturer type
in the past 6 months</td>
<td>SELECT DISTINCT V.VendorCompany, T.TireSpec, (T.SalePrice - TM.Cost) AS Profit
FROM TSalesOrder SO JOIN TSalesOrderLine SL ON SO.SOID = SL.SOID JOIN TProduct P ON SL.ProdID = P.ProdID JOIN TTire T ON P.ProdID = T.TireProdID JOINTireModel TM ON T.TireID = TM.TireModelID JOIN TVendorTire VT ON TM.TireModelID = VT.V_TireID JOIN TVendor V ON VT.VendID = V.VendID
WHERE MONTH(SO.DateOrdered) > MONTH(GETDATE())-6
GROUP BY V.VendorCompany, T.TireSpec, T.SalePrice, TM.Cost
ORDER BY 3 DESC</td>
<td>
<table>
<tr><th></th><th>VendorCompany</th><th>TireSpec</th><th>Profit</th></tr>
<tr><td>1</td><td>BFGoodrich</td><td>235/55R17</td><td>87.5000000000</td></tr>
<tr><td>2</td><td>BridgeStone</td><td>205/65R26</td><td>77.4900000000</td></tr>
<tr><td>3</td><td>BFGoodrich</td><td>235/50R18</td><td>75.4900000000</td></tr>
<tr><td>4</td><td>BFGoodrich</td><td>265/40R21</td><td>75.4900000000</td></tr>
<tr><td>5</td><td>Continental</td><td>235/50R19</td><td>75.4900000000</td></tr>
<tr><td>6</td><td>Continental</td><td>265/70R17</td><td>75.4900000000</td></tr>
<tr><td>7</td><td>BridgeStone</td><td>235/40R19</td><td>73.4900000000</td></tr>
<tr><td>8</td><td>BridgeStone</td><td>265/70R17</td><td>73.4900000000</td></tr>
<tr><td>9</td><td>BridgeStone</td><td>275/55R20</td><td>73.4900000000</td></tr>
<tr><td>10</td><td>Continental</td><td>245/60R20</td><td>73.4900000000</td></tr>
<tr><td>11</td><td>Michelin</td><td>215/55R17</td><td>73.4900000000</td></tr>
<tr><td>12</td><td>BridgeStone</td><td>235/50R18</td><td>72.4900000000</td></tr>
<tr><td>13</td><td>BridgeStone</td><td>265/40R21</td><td>72.4900000000</td></tr>
<tr><td>14</td><td>BridgeStone</td><td>245/45R19</td><td>70.7400000000</td></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 C.FirstName, C.LastName, MAX(S.DateOrdered)
FROM Customer C JOIN Car Ca ON C.CCustID = Ca.CustID LEFT JOIN TSalesOrder S ON Ca.CarID = S.CarID
WHERE YEAR(S.DateOrdered) < YEAR(GETDATE())-1
We did not start the customer data until 2019, so see a rendition of this query below.
SELECT C.FirstName, C.LastName, MAX(S.DateOrdered) AS MostRecentOrderDate
FROM Customer C JOIN Car Ca ON C.CCustID = Ca.CustID LEFT JOIN TSalesOrder S ON Ca.CarID = S.CarID
WHERE Month(S.DateOrdered) < Month(GETDATE())-2
GROUP BY C.FirstName, C.LastName
ORDER BY 3 ASC</td>
<td>
<table>
<tr><th></th><th>FirstName</th><th>LastName</th><th>MostRecentOrderDate</th></tr>
<tr><td>1</td><td>Matthew</td><td>Mendoza</td><td>2019-01-02 00:00:00.0000000</td></tr>
<tr><td>2</td><td>Hoyt</td><td>Mayer</td><td>2019-01-04 00:00:00.0000000</td></tr>
<tr><td>3</td><td>Palmer</td><td>Burks</td><td>2019-01-05 00:00:00.0000000</td></tr>
<tr><td>4</td><td>Leandra</td><td>Duffy</td><td>2019-01-07 00:00:00.0000000</td></tr>
<tr><td>5</td><td>Skyler</td><td>Albert</td><td>2019-01-07 00:00:00.0000000</td></tr>
<tr><td>6</td><td>Marshall</td><td>Anthony</td><td>2019-01-09 00:00:00.0000000</td></tr>
<tr><td>7</td><td>TaShya</td><td>Ochoa</td><td>2019-01-11 00:00:00.0000000</td></tr>
<tr><td>8</td><td>Nehru</td><td>Cooper</td><td>2019-01-16 00:00:00.0000000</td></tr>
<tr><td>9</td><td>Regan</td><td>Anderson</td><td>2019-01-16 00:00:00.0000000</td></tr>
<tr><td>10</td><td>Clinton</td><td>Joyce</td><td>2019-01-16 00:00:00.0000000</td></tr>
<tr><td>11</td><td>Zelenia</td><td>Simmons</td><td>2019-01-16 00:00:00.0000000</td></tr>
<tr><td>12</td><td>Zahir</td><td>Barrett</td><td>2019-01-18 00:00:00.0000000</td></tr>
<tr><td>13</td><td>Benedict</td><td>Parrish</td><td>2019-01-18 00:00:00.0000000</td></tr>
<tr><td>14</td><td>Latifah</td><td>Saunders</td><td>2019-01-21 00:00:00.0000000</td></tr>
<tr><td>15</td><td>Ian</td><td>Medina</td><td>2019-01-23 00:00:00.0000000</td></tr>
<tr><td>16</td><td>Noelle</td><td>Luna</td><td>2019-01-25 00:00:00.0000000</td></tr>
<tr><td>17</td><td>Jacob</td><td>Villarreal</td><td>2019-01-25 00:00:00.0000000</td></tr>
<tr><td>18</td><td>Phoebe</td><td>Case</td><td>2019-01-27 00:00:00.0000000</td></tr>
<tr><td>19</td><td>Kyle</td><td>Love</td><td>2019-01-30 00:00:00.0000000</td></tr>
<tr><td>20</td><td>Lavinia</td><td>Townse...</td><td>2019-01-30 00:00:00.0000000</td></tr>
</table>
</td>
</tr>
<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 C.FirstName, C.LastName, AVG(SL.Quantity*T.SalePrice) AS Total
FROM Customer C JOIN Car Ca ON C.CCustID = Ca.CustID
JOIN TSalesOrder SO ON Ca.CarID = SO.CarID
JOIN TSalesOrderLine SL ON SO.SOID = SL.SOID
JOIN TProduct P ON SL.ProdID = P.ProdID
JOIN TTire T ON T.TireProdID = P.ProdID
GROUP BY C.FirstName, C.LastName
HAVING AVG(SL.Quantity*T.SalePrice) < (SELECT AVG(SL.Quantity*T.SalePrice) FROM TSalesOrderLine SL JOIN TProduct P ON SL.ProdID = P.ProdID
JOIN TTire T ON T.TireProdID = P.ProdID)
ORDER BY 3 ASC</td>
<td>
<table>
<tr><th></th><th>FirstName</th><th>LastName</th><th>Total</th></tr>
<tr><td>1</td><td>Sylvia</td><td>Castillo</td><td>151.8056310679</td></tr>
<tr><td>2</td><td>Alfonso</td><td>Holden</td><td>151.8056310679</td></tr>
<tr><td>3</td><td>Cathleen</td><td>Hyde</td><td>151.8056310679</td></tr>
<tr><td>4</td><td>Wynter</td><td>Rutledge</td><td>151.8056310679</td></tr>
<tr><td>5</td><td>Olympia</td><td>Ross</td><td>151.8056310679</td></tr>
<tr><td>6</td><td>Philip</td><td>Leblanc</td><td>227.7084466019</td></tr>
<tr><td>7</td><td>Allegra</td><td>Mays</td><td>227.7084466019</td></tr>
<tr><td>8</td><td>Linus</td><td>Petty</td><td>227.7084466019</td></tr>
<tr><td>9</td><td>Armando</td><td>Robbins</td><td>227.7084466019</td></tr>
<tr><td>10</td><td>Lani</td><td>Page</td><td>303.6112621359</td></tr>
<tr><td>11</td><td>Quemby</td><td>Lopez</td><td>303.6112621359</td></tr>
<tr><td>12</td><td>Kyle</td><td>Love</td><td>303.6112621359</td></tr>
<tr><td>13</td><td>Noelle</td><td>Luna</td><td>303.6112621359</td></tr>
<tr><td>14</td><td>Burke</td><td>Ruiz</td><td>303.6112621359</td></tr>
<tr><td>15</td><td>Frances</td><td>Kirkland</td><td>303.6112621359</td></tr>
<tr><td>16</td><td>Carson</td><td>Wilson</td><td>303.6112621359</td></tr>
<tr><td>17</td><td>Cody</td><td>Sullivan</td><td>303.6112621359</td></tr>
<tr><td>18</td><td>Tasha</td><td>Taylor</td><td>303.6112621359</td></tr>
<tr><td>19</td><td>Brian</td><td>Hyde</td><td>303.6112621359</td></tr>
<tr><td>20</td><td>Kylynn</td><td>Durham</td><td>303.6112621359</td></tr>
<tr><td>21</td><td>Daquan</td><td>Flynn</td><td>303.6112621359</td></tr>
</table>
</td>
</tr>
</table>

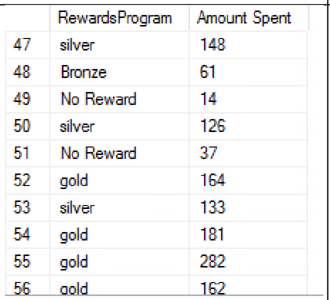
Three Additional Queries

In this section, we have created three additional queries for SonnerTire. We think these queries will help SonnerTire improve their business transactions and relationships with customers and vendors.

Query #	Question	Why is this important	SQL	Partial Output	Recap of Findings
1	How can Sonner Tire know what times of the year that they are busiest (by amount of revenue brought in)?	By knowing which months of the year that more orders are placed, the company will know when they can expect increases in revenue, as well as planning for an increased budget for wages, products, and general supplies.	SELECT MONTH(DateOrdered) as 'Month of the Year' , SUM(SL.Quantity*T.SalePrice) AS Total FROM Customer C JOIN Car Ca ON C.CCustID = Ca.CustID JOIN TSalesOrder SO ON Ca.CarID = SO.CarID JOIN TSalesOrderLine SL ON SO.SOID = SL.SOID JOIN TProduct P ON SL.ProdID = P.ProdID JOIN TTire T ON T.TireProdID = P.ProdID GROUP BY MONTH(DateOrdered) HAVING AVG(SL.Quantity*T.SalePrice) > 0	 Month of the Year Total 1 1 719255.0800000000 2 2 906886.8400000000 3 3 1094518.6000000000 4 4 531623.3200000000 5 5 93815.8800000000 6 6 46907.9400000000 7 7 93815.8800000000 8 8 62543.9200000000 9 9 109451.8600000000	The most revenue brought in was, by far, in February and March. This tells Sonner Tire that they should budget more for utilities, supplies, and employee wages during this time.

2	How can Sonner Tire know how long it takes them to complete an order on average?	By knowing how long the average order takes to complete, the company can see if they are reaching their goals for timely service.	SELECT AVG(DATEDIFF(DAY,DateOrdered,DateCompleted)) as 'Average Time to Fulfill Order (in Days)'
			
			FROM TSalesOrder;
				
				Average Time to Fulfill Order (in Days)
				1 4
					The average time to complete a sales order at Sonner Tire is currently 4 days. This is a long time for most people to go without a car, and they might want to identify inefficiencies in their workflow processes.

3	How can Sonner Tire identify customers who have not purchased a service from them in the last year?	By identifying customers who have not purchased from Sonner Tire in a long time, they can reach out to these customers and possibly win back their business by sending out coupons or contacting them by phone/email.	SELECT C.FirstName, C.LastName, DateOrdered AS 'Last Date Ordered'
			FROM Customer C JOIN Car Ca ON C.CCustID = Ca.CustID
			JOIN TSalesOrder SO ON Ca.CarID = SO.CarID
			JOIN TSalesOrderLine SL ON SO.SOID = SL.SOID
			WHERE DateOrdered< DATEADD(year,-1,GETDATE())
				
				FirstName LastName Last Date Ordered
				1 Allegra Mays 2018-03-22 00:00:00.0000000
				2 Quemby Lopez 2018-04-01 00:00:00.0000000
				3 Oren Horton 2018-02-17 00:00:00.0000000
				4 Quemby Lopez 2018-04-01 00:00:00.0000000
				5 Allegra Mays 2018-03-22 00:00:00.0000000
					Sonner Tire only has 5 customers who have not placed an order within a year. Still, it would not hurt for them to reach out and see of they could win these customers back.

4	How can Sonner tire help increase repeating customers?	By implementing a rewards program, we can offer customers discounts for completing orders based off the amount they spend.	Select Case when sum(Amount) between 0 and 49 then 'No Reward' when sum(Amount) between 50 and 99 then 'Bronze' when sum(Amount) between 100 and 149 then 'silver' when sum(Amount) > 150 then 'gold' end as RewardsProgram, sum(amount) as 'Amount Spent' from Customer C join Car on C.CCustID = Car.CustID Join TSalesOrder SO on So.CarID = car.CarID	 RewardsProgram Amount Spent 47 silver 148 48 Bronze 61 49 No Reward 14 50 silver 126 51 No Reward 37 52 gold 164 53 silver 133 54 gold 181 55 gold 282 56 gold 162	Sonner Tire can have a rewards program in order to gain repeating customers or customer loyalty.

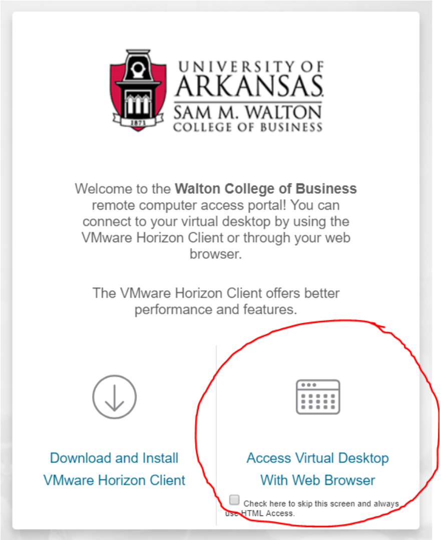
			join TPaymentIn PI on PI.SOID = SO.SOID group by CustID		

User Documentation

The following instructions will walk the user through the entire process of accessing the database. It begins with accessing, opening, and signing into the software and continues to explain how to run SQL queries to obtain information for various business purposes.

1. Go to https://waltonlab.uark.edu/portal/ in web browser

2. Select **“Access Virtual Desktop With Web Browser”**



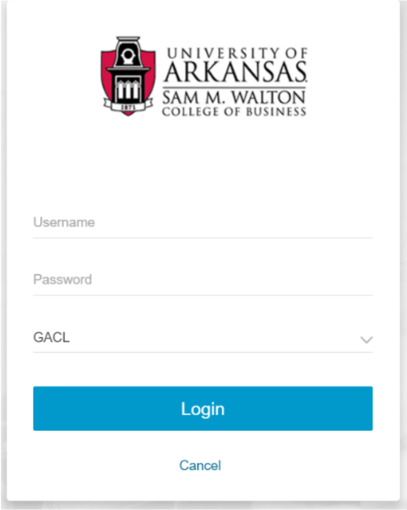
UNIVERSITY OF ARKANSAS SAM M. WALTON COLLEGE OF BUSINESS

Welcome to the **Walton College of Business** remote computer access portal! You can connect to your virtual desktop by using the VMware Horizon Client or through your web browser.

The VMware Horizon Client offers better performance and features.

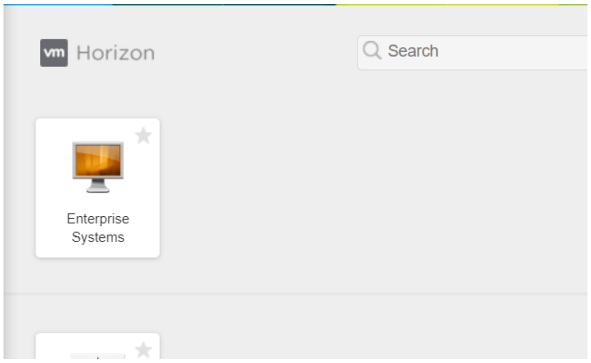
Download and Install VMware Horizon Client
Access Virtual Desktop With Web Browser
□ Check here to skip this screen and always use HTML Access.

3. Sign in using given credentials

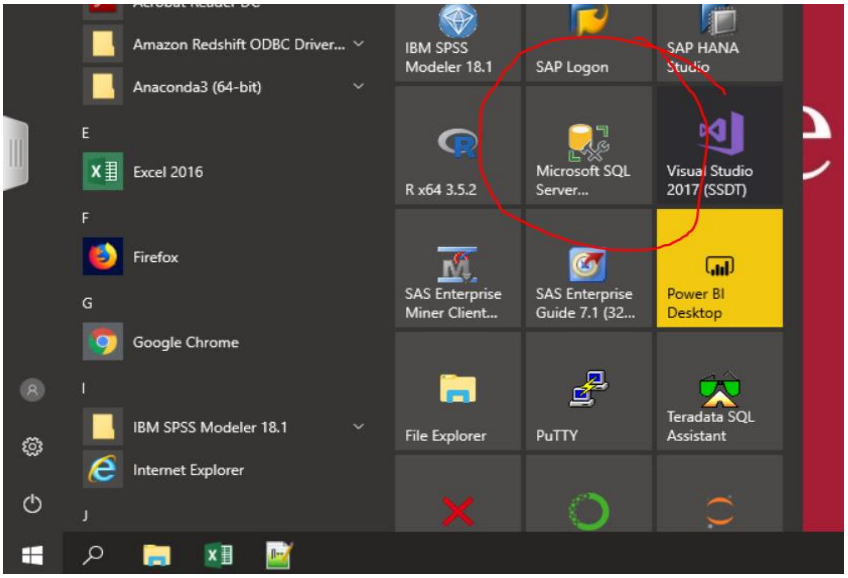


UNIVERSITY OF ARKANSAS SAM M. WALTON COLLEGE OF BUSINESS

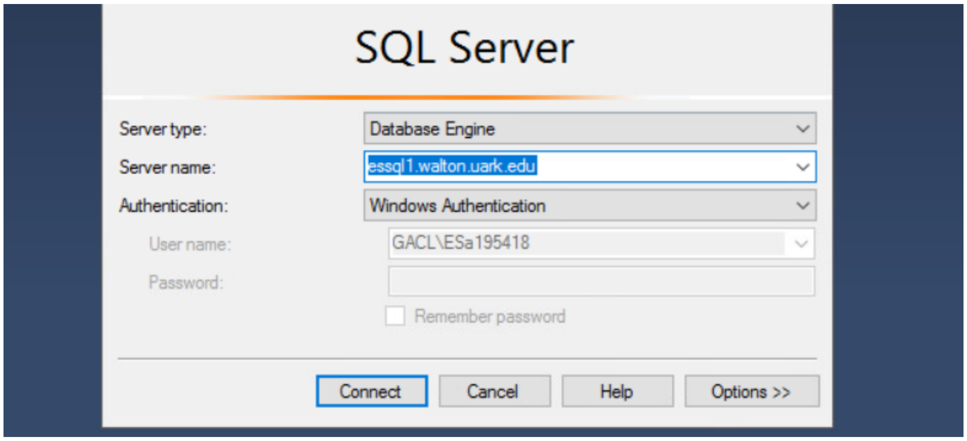
4. Select **“Enterprise Systems”**



5. Under the start menu, search for and select **“Microsoft SQL Server Management Studio”**



6. In the pop-up window, enter **“essql1.walton.uark.edu”** in the **“Server Name”** field and click **“Connect”**


SQL Server

Server type: Database Engine

Server name: essql1.walton.uark.edu

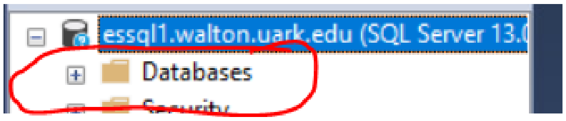
Authentication: Windows Authentication

User name: GACL\ESa195418

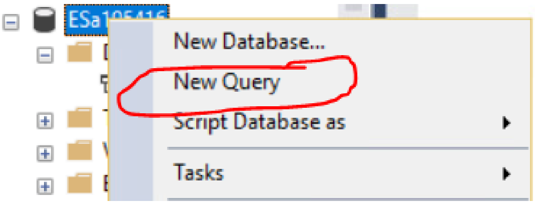
Password:

□ Remember password

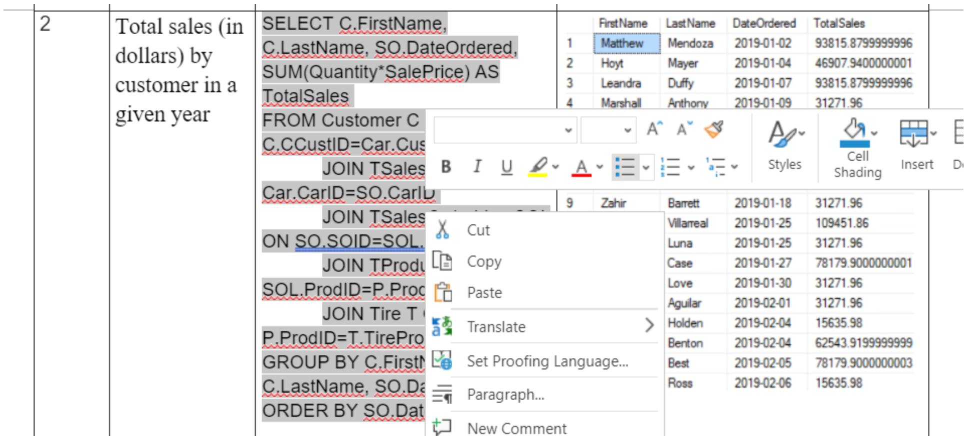
7. On the left side of the screen, in the object explorer toolbar, expand the **“Databases”** tab and navigate to the database titled **“ESa195416”**



8. Right click on **“ESa195416”** and select **“New Query”**



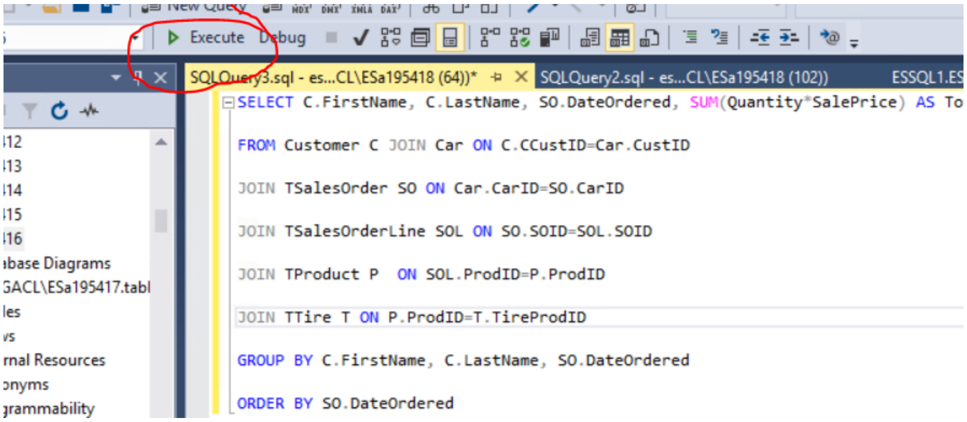
9. A text box now opens. Determine which query needs to be run by referencing the **“Specific SQL Statements Requested Section”** earlier in this document. Once determined which needs to be run, copy the code and paste into the text box in the database.



2	Total sales (in dollars) by customer in a given year	SELECT C.FirstName, C.LastName, SO.DateOrdered, SUM(Quantity*SalePrice) AS TotalSales
		FROM Customer C
		C.CCustID=Car.Cus
		JOIN TSales
		Car.CarID=SO.CarID
		JOIN TSales
		ON SO.SOID=SOL.
		JOIN TProdu
		SOL.ProdID=P.Proc
		JOIN Tire T
		P.ProdID=T.TirePro
		GROUP BY C.FirstN
		C.LastName, SO.Da
		ORDER BY SO.Dat	

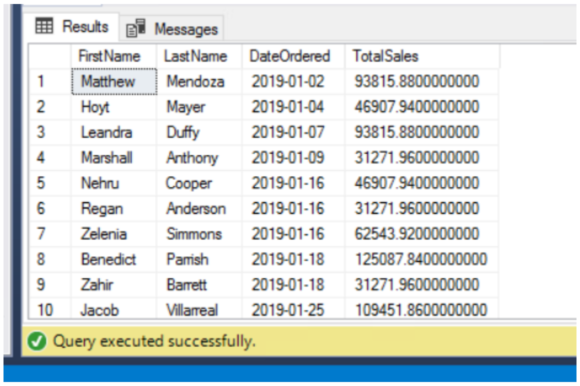
	FirstName	LastName	DateOrdered	TotalSales
1	Matthew	Mendoza	2019-01-02	93815.8799999996
2	Hoyt	Mayer	2019-01-04	46907.9400000001
3	Leandra	Duffy	2019-01-07	93815.8799999996
4	Marshall	Anthony	2019-01-09	31271.96
9	Zahir	Barrett	2019-01-18	31271.96
		Villarreal	2019-01-25	109451.86
		Luna	2019-01-25	31271.96
		Case	2019-01-27	78179.9000000001
		Love	2019-01-30	31271.96
		Aguilar	2019-02-01	31271.96
		Holden	2019-02-04	15635.98
		Benton	2019-02-04	62543.9199999999
		Best	2019-02-05	78179.9000000003
		Ross	2019-02-06	15635.98

10. Then click **“Execute”** above the text box



```
SELECT C.FirstName, C.LastName, SO.DateOrdered, SUM(Quantity*SalePrice) AS To

FROM Customer C JOIN Car ON C.CCustID=Car.CustID

JOIN TSalesOrder SO ON Car.CarID=SO.CarID

JOIN TSalesOrderLine SOL ON SO.SOID=SOL.SOID

JOIN TProduct P  ON SOL.ProdID=P.ProdID

JOIN TTire T ON P.ProdID=T.TireProdID

GROUP BY C.FirstName, C.LastName, SO.DateOrdered

ORDER BY SO.DateOrdered
```

11. Results are displayed below the text entry box

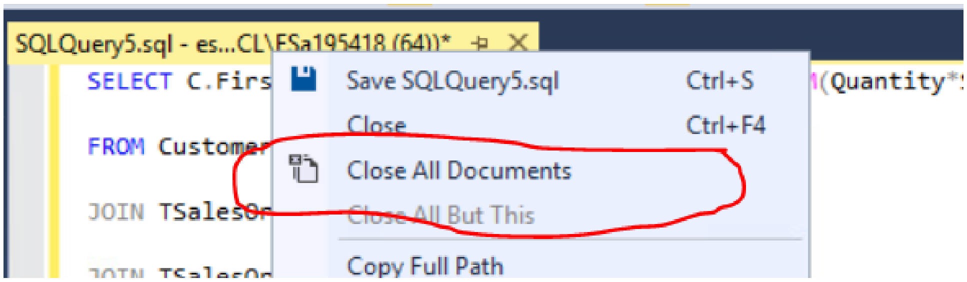


	FirstName	LastName	DateOrdered	TotalSales
1	Matthew	Mendoza	2019-01-02	93815.8800000000
2	Hoyt	Mayer	2019-01-04	46907.9400000000
3	Leandra	Duffy	2019-01-07	93815.8800000000
4	Marshall	Anthony	2019-01-09	31271.9600000000
5	Nehru	Cooper	2019-01-16	46907.9400000000
6	Regan	Anderson	2019-01-16	31271.9600000000
7	Zelenia	Simmons	2019-01-16	62543.9200000000
8	Benedict	Parrish	2019-01-18	125087.8400000000
9	Zahir	Barrett	2019-01-18	31271.9600000000
10	Jacob	Villarreal	2019-01-25	109451.8600000000

Query executed successfully.

12. To run a new query, simply repeat from steps 8 though 10

13. When done using database, right click the tab at the top and select **“Close all documents”** **Do not save any changes made**.


SQLQuery5.sql - es...CL\FSa195418 (64))*
Save SQLQuery5.sql Ctrl+S
Close Ctrl+F4
Close All Documents
Close All But This
Copy Full Path

```
SELECT C.Firs                                   (Quantity*

FROM Custome

JOIN TSalesOr

JOIN TSalesOr
```

What We Learned Throughout This Process

As a team, we have learned how to work together to solve problems and complete major tasks. This semester we had the opportunity to get to know each others strengths and weaknesses and how each of us work in a team. We had some frustrating moments getting the database to work in the server that was provided, and luckily, someone always came up with a solution when the rest of us couldn't figure it out.

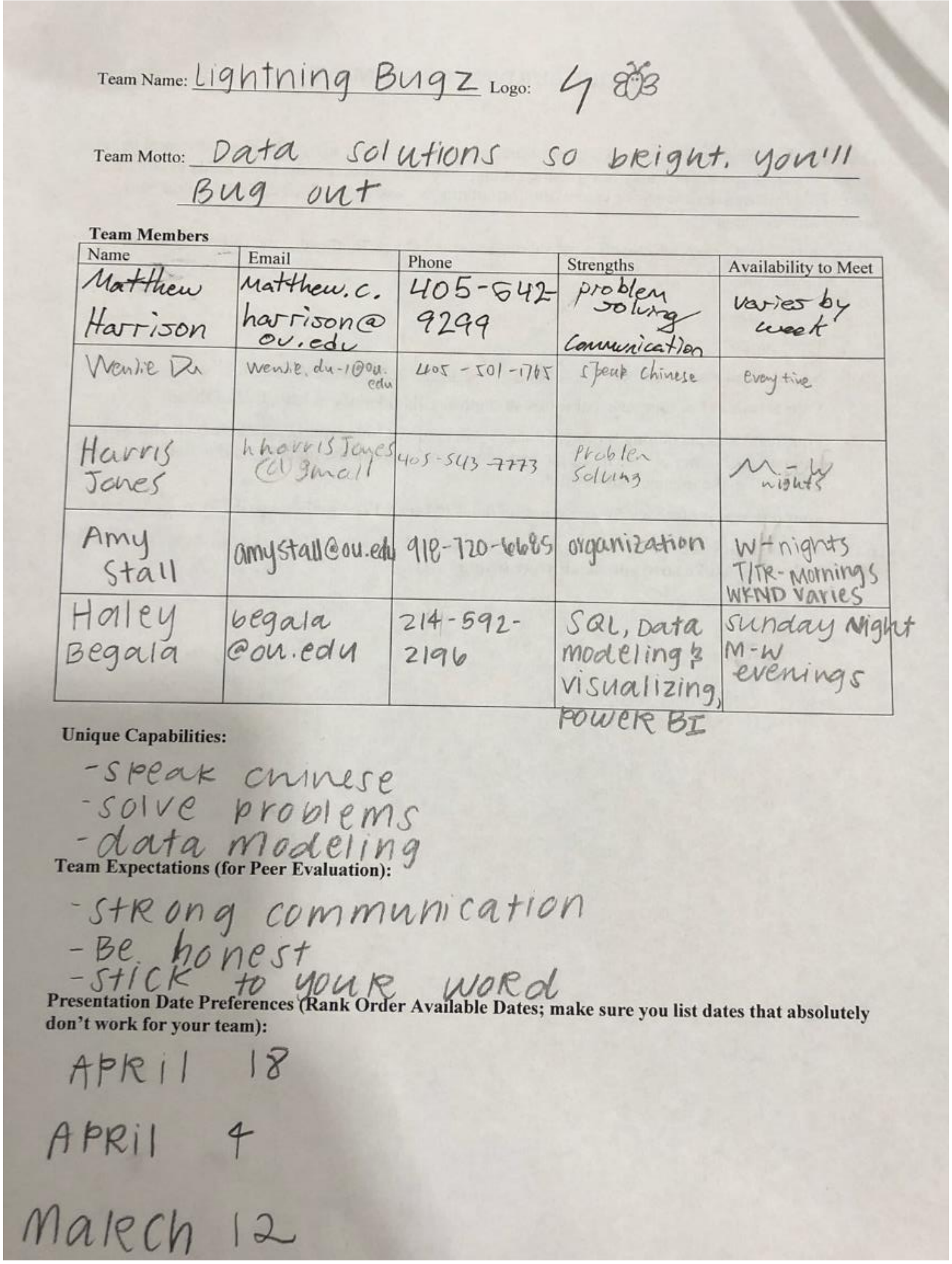
Member Name:	What you learned:
Haley Begala	This project taught me a lot, but most importantly, I learned how to take advantage of the strengths in the group. Our group is diverse, and my team members brought many skills to the table. We had many discussions about whose skills we could capitalize on. Learning how to capitalize on people's skills made the process more efficient. We tried to task things out based on strengths, but also wanted people to practice some of the things they are weaker in. There were moments where I really disliked this project, but now that we've completed it, I feel happy with the work my team put forward.
Wenjie Du	This project gives me a view of whole process for making ERD and generating data as well as importing data, which combine all database skills we learned in classes. It is a tough project but it is worthy. Most importantly, we found and corrected technical mistakes one by one through discussing with each other in a group and learning from other groups representations as well. Although the whole process is frustrated, we worked together and finished it on time. Moreoverour teammates are best since they help me a lot in this project. My English is kind of rough, so they helped me to check paragraph I wrote. I am happy to work with them!
Harris Jones	The Sonner Tire group project had not only taught me valuable technical skills, but also let me grow in my own skills. Learning how to fluently read ERDs and normalized relations has given me confidence in creating databases. Not only have I learned valuable technical skills, but I've also learned a great deal of communicating efficiently in a group of my peers. This group workflow has been unique as we have all been the busiest, we've ever been in our lives. The amount I've been able to progress myself in database work would have been significantly less without collaboration. As we were all able to help each other in our weak points, we have grown stronger through learning from each other. Overall, this project has allowed me to develop and reinforce skills in creating a functioning database.
Matthew Harrison	This project gave me valuable experience with working in a group. I learned to delegate my work and ask for help with tasks when I knew something exceeded my comfort zone. The project was definitely challenging, and all of us in the group learned a lot going through the required milestones for the project, but it was great to see everyone working together on particularly tasks that none of us could accomplish

	efficiently on our own. Moreover, as one of the first large technical projects I have completed in my education, it was refreshing to see everything I have learned over the years put into practice. Everything from software development and coding skills to my expository writing skills were used at some point in this project. Overall, the entire endeavor was exciting and challenging as we put together all the working parts to come up with a finished product, and I am excited to work on more projects like this in the future, whether it be in college or in my career.
Amy Stall	This is the first semester long project I have worked on with a group, so one of the biggest things I learned was how to work with a group of people for multiple months. We eventually figured out what skills people could bring forward to help the team, and in the future, I know to have everyone figure out their strengths and what they can do to help before furthering into the project. As far as databases go, I learned a lot about ERD’s and of course building a database. It was really nice to see my SQL skills put to work after working on building the actual database the queries would run on. A project management tool I was able to work on was task delegation. I typically have a hard time letting other people work on things because I am a perfectionist, but this project was too big to do alone so working with people was a big lesson.

Appendix

Team Contract

Below is our team contract. This was made at the beginning of the semester so that we could learn more about eachothers strengths, availability, capabilities, and our team expectations. We have been able to follow these rules and hold eachother to our expectations.



Team Name: Lightning Bugz Logo: 4

Team Motto: Data solutions so bright, you'll Bug out

Team Members

Name	Email	Phone	Strengths	Availability to Meet
Matthew Harrison	matthew.c.harrison@ou.edu	405-542-9299	problem solving communication	varies by week
Wenlie Du	wenlie.du-1@ou.edu	405-501-1765	speak chinese	Every time
Harris Jones	hharrisJones@gmail	405-543-7773	Problem Solving	M-W nights
Amy Stall	amystall@ou.edu	918-720-6685	organization	W-nights T/TR-Mornings WKND varies
Haley Begala	begala@ou.edu	214-592-2196	SQL, Data modeling & visualizing, POWER BI	sunday Night M-W evenings

Unique Capabilities:

- speak chinese
- solve problems
- data modeling

Team Expectations (for Peer Evaluation):

- strong communication
- Be honest
- stick to your word

Presentation Date Preferences (Rank Order Available Dates; make sure you list dates that absolutely don't work for your team):

April 18

April 4

March 12

Data Dictionary Model

The data dictionary allows us to identity the data type with each attribute in the entities. This will help when entering data and when running queries.

Data Dictionary						
Table	Field Name	Data Type	Field Size	Constraints	References	Example
TState	Statecode*	char	2	Primary key		OK
	StateName	char	30	Not Null		Oklahoma
TZip	Zipcode*	int	10	Primary Key		73072
	City	char	32	Not null		norman
	ZStateCode	char	2	Foreign Key	TState	OK
Tyear	YearID	int	3	Primary Key Auto increment by 1		2
	Description	nvarchar	50	Not null		1990
Tmodel	ModelID*	int	4	Primary key		1
	Description	nvarchar		Not null		F150
TInsurance	InsuranceID	int	4	Primary Key		1
	Description	ncarchar	50	Not null		All tires insured/ not tires
TProduct	ProdID*	int	4	Primary Key		1
	ProductType	nvarchar	50	Not null		bumper
Employee	EmployeeID*	int	4	Primary Key		1
	Emp_F_Name	nvarchar	50	Not null		John
	Emp_L_Name	nvarchar	50	Not null		Smith
	Position	ncarchar	50	Not null		rotator
TPayment Terms	PmtTermsID*	int	4	Primary key		1
	MaxNumOfPayments	int	4	Not null		5 payment max

Table	Field Name	Data Type	Field Size	Constraints	References	Example
	MinPaymentAmt	int	4	Not null		1 payment minimum
TPayIn	PayInID*	int	4	Primary Key		1
	Date	date		Not null		3/3/19
	Amount	Money	8	not null		$330
	Description	nvarchar	50	Not null		Visa card
TService	TireModelID*	int	4	Primary Key		1
	SCost	money	8	Not null		$50
	SDescription	nvarchar	50	Not null		?
TRepairService	RepairServiceID*	int	4	Not null		1
	RSCarID	itn	4	Foreign Key, on delete restirct	Tcar	1
	RSpayInID	int	4	Foreign Key, on delete restrict	TpayIn	2
	RSCarID	int	4	Foreign Key, on delete restrict		3
TCar	CarID*	int	4	Primary Key		1
	CarModel	int	4	Foreign Key, on delete restrict	TModel	2
	CarCustID	int	4	Foreign Key, null allowed on delete restrict	TCustomer	3
	Comments	nvarchar	50			She’s a classic be careful.

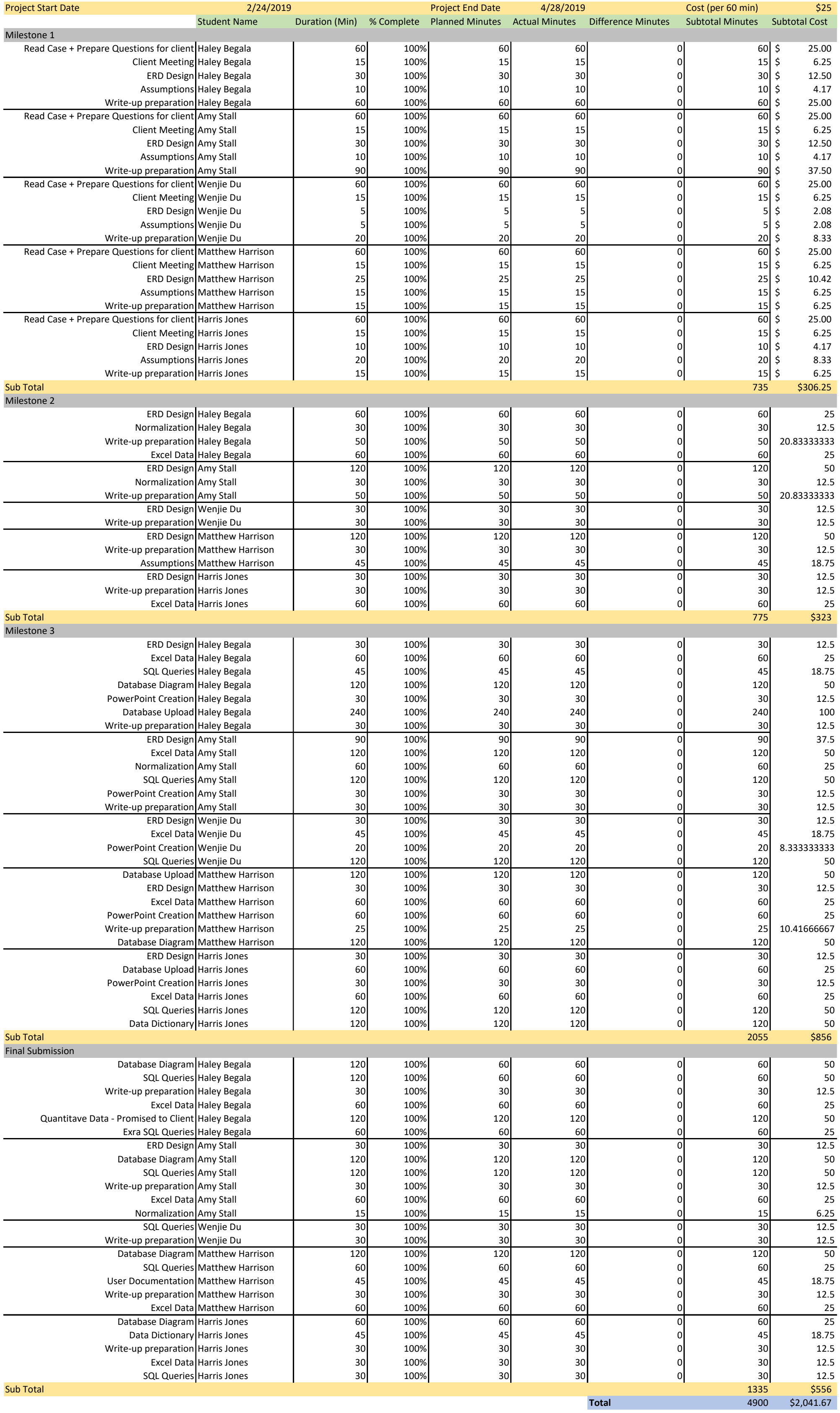
Table	Field Name	Data Type	Field Size	Constraints	References	Example
TMake	MakeID*	int	4	Primary Key		1
	MakeName	nvarchar	50	Not null	TYear	Toyota
TSalesOrder	SOID*	int	4	Primary Key		1
	SICarid	int	4	Foreign Key, on delete restrict	TCar	2
	SOOrderDate	Date		Not null		3/3/19
	SOStatus	Nvarchar	50	Not null		complete
	SOEmpID	int	4	Foreign Key, on delete restrict	TEmployee	4
TSalesOrderLine	SOLID*	int	4	Primary Key		1
	SOLProdID	int	4	Foreign Key, on delete restrict	SOLProdId	2
	SOLSOID	int	4	Foreign Key, on delete restrict	TSalesOrder	3
	SOLWarranty	nvarchar	50			2 years
	SOLFullSet	nvarchar	50	Not null		Yes
	SOLQuantity	int	4	Not null		50
TPurchase Order	POID*	int	4	Primary Key		1
	POEmptID	int	4	Foreign Key, on delete restrict	TEmployee	2

Table	Field Name	Data Type	Field Size	Constraints	References	Example
	PovPmtID	int	4	Foreign Key, null allowed on delete restrict	TVendorPmtBook	3
	PODatePurchased	nvarchar	50	Not null		3/3/19
	PODatePaid	nvarchar	50	Not null		3/4/19
	POPayFull	nvarchar	50	Not null		No
	POPaymentAmount	Money	8	Not null		$400
TPurchaseOrderLine	POLID*	int	4	Primary Key		1
	POLPOID	int	4	Foreign Key, on delete restrict	TPurchase Order	2
	POLPRODID	int	4	Foreign Key, on delete restrict	Tproduct	3
	POLV_TireID	int	4	Foreign Key, on delete restrict	TV_Tire	4
	quantity	int	4	Not null		300
	discount	nvarchar	50	Not null		50%
TTireModel	TireModelID*	int	4	Primary Key		1
	TMProdID	int	4	Foreign Key, on delete restrict	Tproduct	2
	TMModelID	int	4	Foreign Key, on delete restrict	TModel	3
TCustomer	CustID*	int	4	Not Null		1

Table	Field Name	Data Type	Field Size	Constraints	References	Example
	CFirstName	nvarchar	50	Not null		John
	CLastName	nvarchar	50	Not null		Smith
	CStreet	nvarchar	50	Not null		2020 kanye dr
	CZipcode	nvarchar	50	Foreign Key, on delete restrict	Tzip	73071
	CCreditLimit	nvarchar	50			500
	CPmtTerms	nvarchar	50	Not Null		1/10, 1/30n
	CPhoneNum	nvarchar	50	Notn ull		405-555-5555
TTire	TireProdID*	int	4	Primary Key		1
	TireAspectRatio	nvarchar	50	Not Nul		66R
	TireDiameter	nvarchar	50	Not null		205(mm)
	TireLoadIndex	nvarchar	50	Not null		98T
	TireSpeedRate	nvarchar	50	Not null		4 star
	TireName	Nvarchar`	50	Not null		GoodYear
	TireSpecName	nvcarchar	50	Not null		
	Mialage	int	6	Not null		30,000
	Description	nvarchar	250	Not null		All terraine
	cost	int	3	Not null		140
	Sale_Price	int	3	Not null		200
	TireInsuranceID	int	4	Foreign Key, null allowed on delete set null	TInsurance	1
TVendor	VendID*	int	4	Primary key		1

Table	Field Name	Data Type	Field Size	Constraints	References	Example
	VStreet	Nvarchar	50	Not null		300 corporate ave
	VZipCode	nvarchar	50	Foreign Key, on delete restrict	TZipCode	73071
	Vstate	nvarchar	50	Foreign key, on delete restrict	TState	oklahoma
	VContactFirstname	nvarchar	50	Not null		John
	VContactLastName	nvarchar	50	Not null		Smith
	VPhoneNumber	nvarchar	50	Not null		405-444-444
	VPmtTerms	nvarchar	50	Not null		1/10, 30/n
	VVendorName	nvarchar	50	Not null		Goodyear inc.
TVendor_Tire	V_TireID	int	4	Primary Key		1
	VTVendID	int	4	Foreign Key, on delete restrict		2
	VTProductID	int	4	Foreign Key, on delete restrict		3

Project Management



Project Start Date	2/24/2019			Project End Date	4/28/2019		Cost (per 60 min)	$25
	Student Name	Duration (Min)	% Complete	Planned Minutes	Actual Minutes	Difference Minutes	Subtotal Minutes	Subtotal Cost
Milestone 1								
Read Case + Prepare Questions for client	Haley Begala	60	100%	60	60	0	60	$ 25.00
Client Meeting	Haley Begala	15	100%	15	15	0	15	$ 6.25
ERD Design	Haley Begala	30	100%	30	30	0	30	$ 12.50
Assumptions	Haley Begala	10	100%	10	10	0	10	$ 4.17
Write-up preparation	Haley Begala	60	100%	60	60	0	60	$ 25.00
Read Case + Prepare Questions for client	Amy Stall	60	100%	60	60	0	60	$ 25.00
Client Meeting	Amy Stall	15	100%	15	15	0	15	$ 6.25
ERD Design	Amy Stall	30	100%	30	30	0	30	$ 12.50
Assumptions	Amy Stall	10	100%	10	10	0	10	$ 4.17
Write-up preparation	Amy Stall	90	100%	90	90	0	90	$ 37.50
Read Case + Prepare Questions for client	Wenjie Du	60	100%	60	60	0	60	$ 25.00
Client Meeting	Wenjie Du	15	100%	15	15	0	15	$ 6.25
ERD Design	Wenjie Du	5	100%	5	5	0	5	$ 2.08
Assumptions	Wenjie Du	5	100%	5	5	0	5	$ 2.08
Write-up preparation	Wenjie Du	20	100%	20	20	0	20	$ 8.33
Read Case + Prepare Questions for client	Matthew Harrison	60	100%	60	60	0	60	$ 25.00
Client Meeting	Matthew Harrison	15	100%	15	15	0	15	$ 6.25
ERD Design	Matthew Harrison	25	100%	25	25	0	25	$ 10.42
Assumptions	Matthew Harrison	15	100%	15	15	0	15	$ 6.25
Write-up preparation	Matthew Harrison	15	100%	15	15	0	15	$ 6.25
Read Case + Prepare Questions for client	Harris Jones	60	100%	60	60	0	60	$ 25.00
Client Meeting	Harris Jones	15	100%	15	15	0	15	$ 6.25
ERD Design	Harris Jones	10	100%	10	10	0	10	$ 4.17
Assumptions	Harris Jones	20	100%	20	20	0	20	$ 8.33
Write-up preparation	Harris Jones	15	100%	15	15	0	15	$ 6.25
Sub Total							735	$306.25
Milestone 2								
ERD Design	Haley Begala	60	100%	60	60	0	60	25
Normalization	Haley Begala	30	100%	30	30	0	30	12.5
Write-up preparation	Haley Begala	50	100%	50	50	0	50	20.83333333
Excel Data	Haley Begala	60	100%	60	60	0	60	25
ERD Design	Amy Stall	120	100%	120	120	0	120	50
Normalization	Amy Stall	30	100%	30	30	0	30	12.5
Write-up preparation	Amy Stall	50	100%	50	50	0	50	20.83333333
ERD Design	Wenjie Du	30	100%	30	30	0	30	12.5
Write-up preparation	Wenjie Du	30	100%	30	30	0	30	12.5
ERD Design	Matthew Harrison	120	100%	120	120	0	120	50
Write-up preparation	Matthew Harrison	30	100%	30	30	0	30	12.5
Assumptions	Matthew Harrison	45	100%	45	45	0	45	18.75
ERD Design	Harris Jones	30	100%	30	30	0	30	12.5
Write-up preparation	Harris Jones	30	100%	30	30	0	30	12.5
Excel Data	Harris Jones	60	100%	60	60	0	60	25
Sub Total							775	$323
Milestone 3								
ERD Design	Haley Begala	30	100%	30	30	0	30	12.5
Excel Data	Haley Begala	60	100%	60	60	0	60	25
SQL Queries	Haley Begala	45	100%	45	45	0	45	18.75
Database Diagram	Haley Begala	120	100%	120	120	0	120	50
PowerPoint Creation	Haley Begala	30	100%	30	30	0	30	12.5
Database Upload	Haley Begala	240	100%	240	240	0	240	100
Write-up preparation	Haley Begala	30	100%	30	30	0	30	12.5
ERD Design	Amy Stall	90	100%	90	90	0	90	37.5
Excel Data	Amy Stall	120	100%	120	120	0	120	50
Normalization	Amy Stall	60	100%	60	60	0	60	25
SQL Queries	Amy Stall	120	100%	120	120	0	120	50
PowerPoint Creation	Amy Stall	30	100%	30	30	0	30	12.5
Write-up preparation	Amy Stall	30	100%	30	30	0	30	12.5
ERD Design	Wenjie Du	30	100%	30	30	0	30	12.5
Excel Data	Wenjie Du	45	100%	45	45	0	45	18.75
PowerPoint Creation	Wenjie Du	20	100%	20	20	0	20	8.333333333
SQL Queries	Wenjie Du	120	100%	120	120	0	120	50
Database Upload	Matthew Harrison	120	100%	120	120	0	120	50
ERD Design	Matthew Harrison	30	100%	30	30	0	30	12.5
Excel Data	Matthew Harrison	60	100%	60	60	0	60	25
PowerPoint Creation	Matthew Harrison	60	100%	60	60	0	60	25
Write-up preparation	Matthew Harrison	25	100%	25	25	0	25	10.41666667
Database Diagram	Matthew Harrison	120	100%	120	120	0	120	50
ERD Design	Harris Jones	30	100%	30	30	0	30	12.5
Database Upload	Harris Jones	60	100%	60	60	0	60	25
PowerPoint Creation	Harris Jones	30	100%	30	30	0	30	12.5
Excel Data	Harris Jones	60	100%	60	60	0	60	25
SQL Queries	Harris Jones	120	100%	120	120	0	120	50
Data Dictionary	Harris Jones	120	100%	120	120	0	120	50
Sub Total							2055	$856
Final Submission								
Database Diagram	Haley Begala	120	100%	60	60	0	60	50
SQL Queries	Haley Begala	120	100%	60	60	0	60	50
Write-up preparation	Haley Begala	30	100%	30	30	0	30	12.5
Excel Data	Haley Begala	60	100%	60	60	0	60	25
Quantitave Data - Promised to Client	Haley Begala	120	100%	120	120	0	120	50
Exra SQL Queries	Haley Begala	60	100%	60	60	0	60	25
ERD Design	Amy Stall	30	100%	30	30	0	30	12.5
Database Diagram	Amy Stall	120	100%	120	120	0	120	50
SQL Queries	Amy Stall	120	100%	120	120	0	120	50
Write-up preparation	Amy Stall	30	100%	30	30	0	30	12.5
Excel Data	Amy Stall	60	100%	60	60	0	60	25
Normalization	Amy Stall	15	100%	15	15	0	15	6.25
SQL Queries	Wenjie Du	30	100%	30	30	0	30	12.5
Write-up preparation	Wenjie Du	30	100%	30	30	0	30	12.5
Database Diagram	Matthew Harrison	120	100%	120	120	0	120	50
SQL Queries	Matthew Harrison	60	100%	60	60	0	60	25
User Documentation	Matthew Harrison	45	100%	45	45	0	45	18.75
Write-up preparation	Matthew Harrison	30	100%	30	30	0	30	12.5
Excel Data	Matthew Harrison	60	100%	60	60	0	60	25
Database Diagram	Harris Jones	60	100%	60	60	0	60	25
Data Dictionary	Harris Jones	45	100%	45	45	0	45	18.75
Write-up preparation	Harris Jones	30	100%	30	30	0	30	12.5
Excel Data	Harris Jones	30	100%	30	30	0	30	12.5
SQL Queries	Harris Jones	30	100%	30	30	0	30	12.5
Sub Total							1335	$556
						Total	4900	$2,041.67