University of Oklahoma

Sonner Tire Company

Anonymous

SQL Server: UOKA0329

Braum Russell, Mason Matray, Coleson Cheek, Galavonni Wilson, Shohruz Junaidov

Professor Swetha Siripurapu

Executive Summary

Anonymous is an IT solutions-based firm specializing in database design and development solutions for businesses to business transactions. With a decade of experience, we have provided secure and efficient data management solutions to our clients. Our loyalty to commitment enables companies to successfully run operations and securely hold all sensitive and public data needs. Sonner Tires is a retailer of tires and surrounding services. Anonymous provided a database to Sonner Tires to better track, control, and simplify the overall operations of their company.

After meeting with the client, our team, Anonymous, was provided with sample data to be included in the database. We used methods of cleaning data to make sure it was accurate within the database. After using these methods, the data was made non-redundant with better access for pulling queries as specified by Sonner Tire Company. We modeled our database based off the data provided to help aid specific queries. For example, we divided the attributes from the car table, and added reference tables to refer to the manufacturer, model, and car tire. This allows Sonner Tire company to keep better track of the cars being serviced.

After the ERD was created and the sample data was given, we began the implementation process of the project. All the data Sonner Tire needs to run an efficient process has been implemented. To ensure efficiency, we eliminated all unnecessary tables from the ERD to create the quickest return of data. For example, we eliminated the product cycle from our ERD to ensure this was met. This process was crucial in making sure all the keys matched up properly, as well as the data, to ensure that the queries ran correctly.

After the successful design and implementation of our database, we were able to use Structured Query Language (SQL) to efficiently retrieve and analyze data. With SQL, we were able to query our database and extract specific information that was asked from us to be able to report so Sonner Tire would be able to make informed business decisions. By leveraging the benefits of our well-designed database and SQL, we have been able to gain valuable insights to allow Sonner Tire to make data-driven decisions for the reports the company wanted to be able to query, as well as provide three additional queries we would recommend useful for making business decisions.

The finished project by Anonymous for Sonner Tire Company will cost an estimated amount between $5500-$6500 and will take two months to complete.

Contents

Get to Know the Team: Anonymous

Name	Major	Year	Experience	Background
Mason Matray 	MIS	Sophomore	N/A	I am from Blanchard, OK. I currently plan on going to Law School.
Coleson Cheek 	MIS	Junior	N/A	I am from Cleveland, Ok. I currently work for myself, but I plan to use this degree to find something new.
Shohruz Junaidov 	MIS	Sophomore	N/A	I am from Tajikistan. I am planning to enroll in MIT accelerated program and focus on data analytics and consulting areas.
Braum Russell 	MIS	Junior	Incoming Business System Analyst Intern at Koch Industries	I am from Indianapolis, IN but live in Fort Worth, TX. I plan on moving into Cybersecurity or Data Analytics.
Galavonni Wilson	MIS	Junior	N/A	I was born in Edmond, Oklahoma. My focus is on Cyber Security and Software

				Development.

Conceptual Design

Conceptual design is the initial phase of designing a database. It involves creating a representation of entities and their relationships with each other. In this first part of designing a database, it sets up a foundation for later design processes such as logical and physical design. In conceptual design, our input is our ERD and we process this with SQL to output invoices. Anonymous met with Professor Swetha to discuss our uncertainties and problems related to the ERD. The questions help us resolve our team's issues and begin building the ERD. The meeting was held over zoom with the entire team in attendance. After the meeting, the answers help us make significant changes to the integrated ERD. This meeting helped clarify the requirements needed for a successful entity relationship diagram.

The Client Meeting

Our team, Anonymous, met with Professor Swetha to discuss our questions for the assignment. This meeting lasted close to ten minutes.

- Meeting Time: 5:00 pm March 20, 2023
- Location: Zoom
- Interviewers: Mason Matray, Braum Russell, Coleson Cheek, Shohruz Junaidov, Galavonni Wilson
- Interviewee: Professor Swetha Siripurapu

Q&A During the Meeting & Information We Learned

1. Should phone numbers also be included for the customer?
    a. Yes, phone numbers should be included for customers.
2. What other customer information should be included?
    a. Think of it as a car dealership and include all information they would need.
3. Since we need to track what customers are the best and which are problematic, should we track this by including the total number of times a customer has missed their payments?
    a. Yes, this is a great way to track this.
4. It has been stated that we need to track which employees did what jobs. Are the only employees Mr. and Mrs. Anderson, their 5 sons, and 10 technicians.
    a. Yes, this is who to track.

Significant Assumptions

1. We assume that customers do not need to make an appointment and can come in for walk-ins for services if needed.
2. We assumed either owner of the car can pick up the car when it is ready for pick up.
3. We assumed that the company wanted to keep track of tire lifespan/safety, so we added TVDate to the TTireVendor entity.
4. We assumed an employee could either receive the delivery, inspect the delivery, or do both.
5. We assume that discount is referenced off the TSalesOrder entity because Sonner Tire Company needs to know if four tires were bought to be able to determine if the customer is eligible for the discount.

What is an ERD? Why is it necessary?

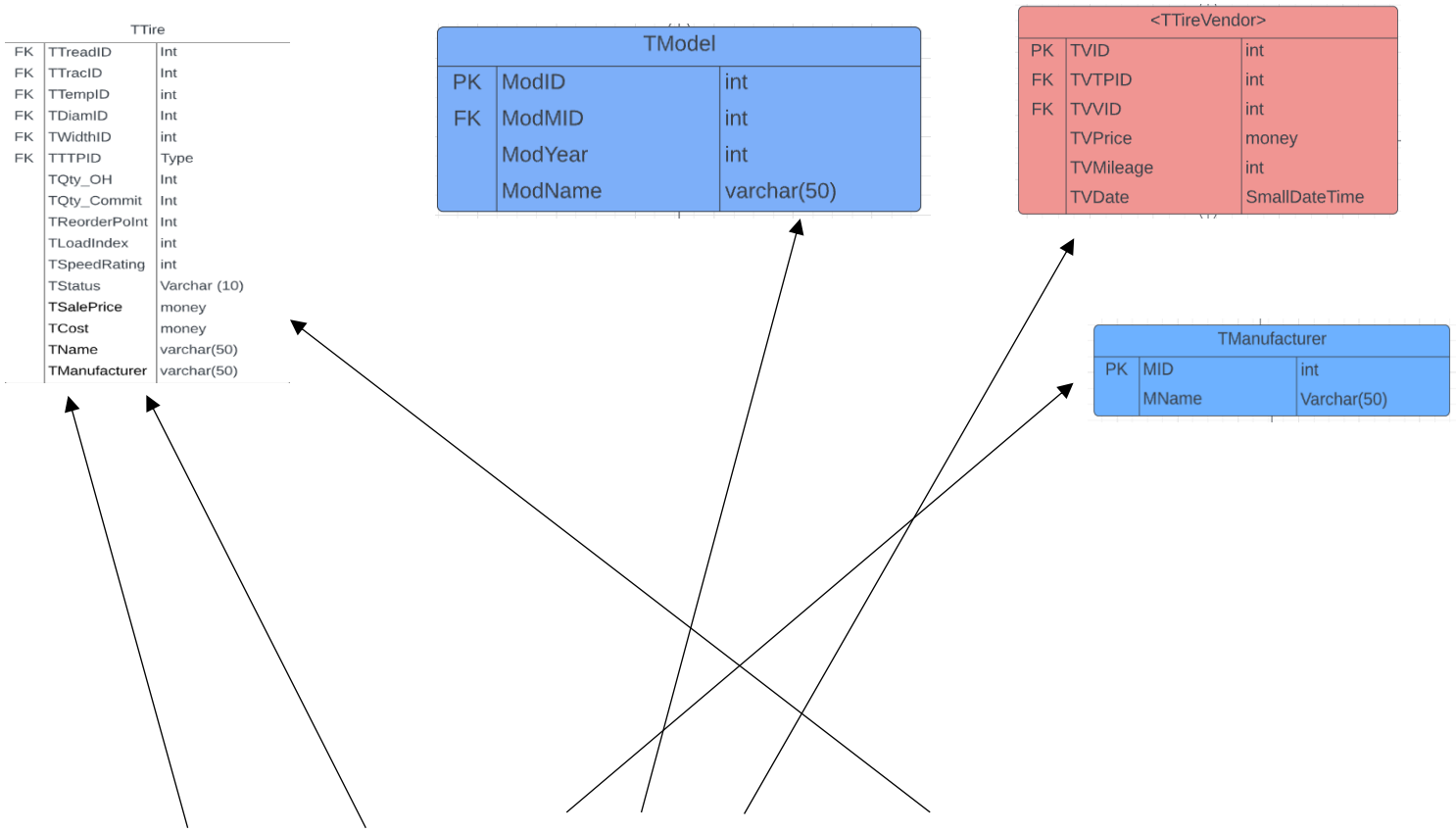
An ERD is a graphical representation of relationships within a company. It includes tables of data known as entities. These entities are related to each other with cardinalities showing their relationships. This is important to show how the company operates, keeping a database of all important information. In this project, the entity relationship diagram for Sonner Tire Company, needs to include information for tires, customers, employees, and their relationships in between.

Business Cycles Used

We are incorporating two cycles out of the three business cycles. In our ERD we incorporated the revenue and expenditure cycle. We picked these two business cycles to track the revenue generated within Sonner Tire Company and to track the expenditures of where money is being spent. The production cycle was not included in the ERD since the company does not produce tires. The expenditure cycle is the process of spending money to be able to reinvest in the company. The purpose of the expenditure cycle is to minimize the cost of acquiring and maintaining inventory, supplies, and other necessary services. The revenue cycle is used to provide the right product, in the right place, at the right time, for the right price. The tires are purchased from various vendors needing the expenditure cycle to be included. The revenue cycle is included in our ERD from the revenue generated from customers.

Data Provided by the Client

This section provides the data provided by Sonner Tires. This data will be stored in the database after being normalized to better serve the needs of Sonner Tire Company. Below you will find the first five rows of data and the specific locations of where it is located within our database. The data is linked to the proper attributes within the tables.

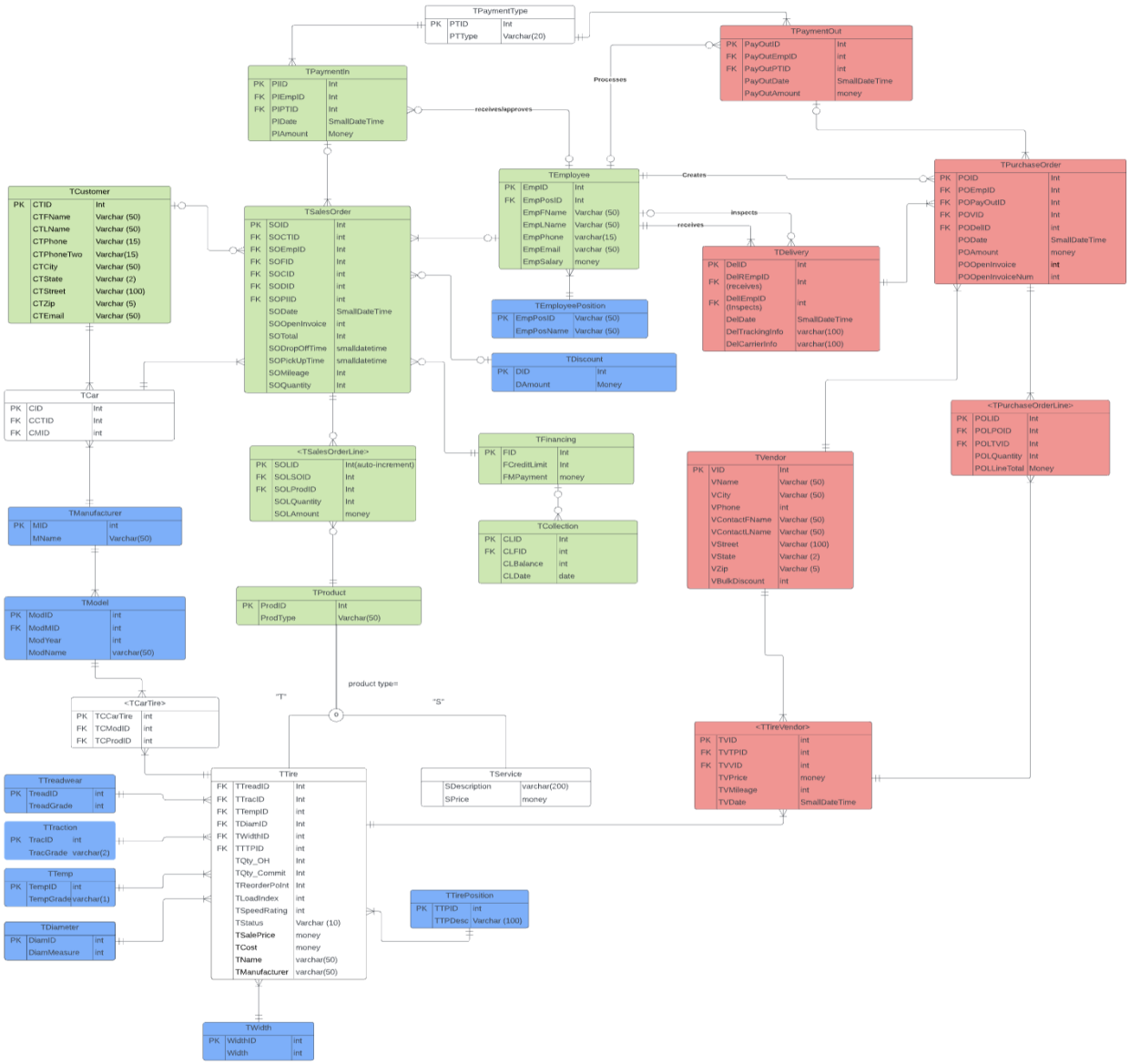


Tire Name	Manufacturer	Good for	Milage	Cost	Sale Price (without taxes)
235/55R17	BFGoodrich	Ford Escape 2014	60000	$ 88.50	$ 146.99
235/55ZR17	BFGoodrich	Ford Escape 2014	45000	$ 80.25	$ 131.99
235/55R17	G-Force R1	Ford Escape 2015	60000	$88.50	$146.99
235/55R17	G-Force Rival	Ford Escape 2016	45000	$80.25	$131.99
235/55R17	SureContact LX	Ford F-150 2017	60000	$130.50	$218.00

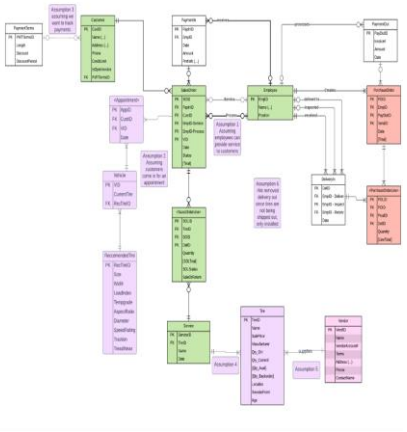
ERD Created

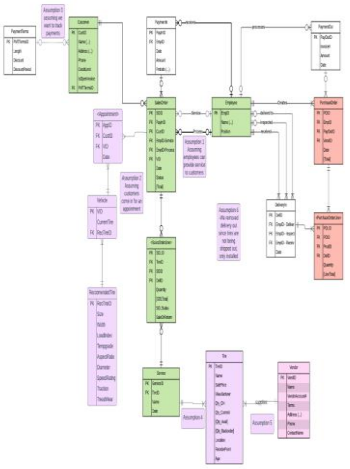
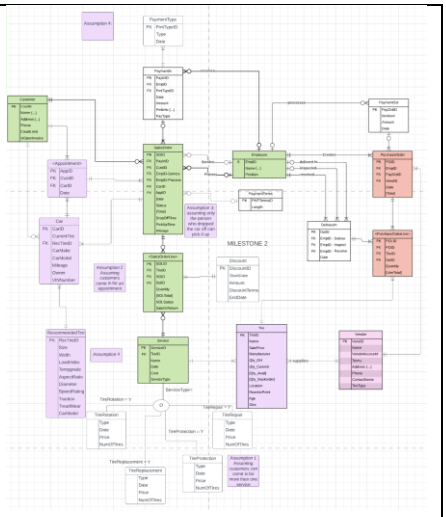
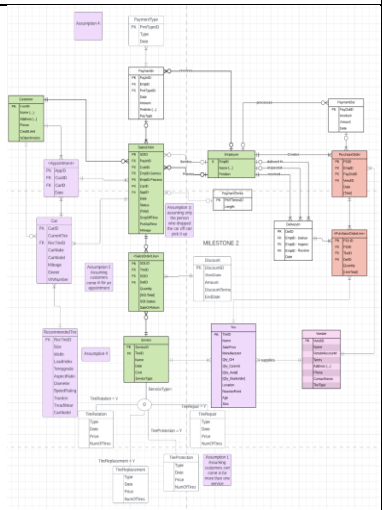
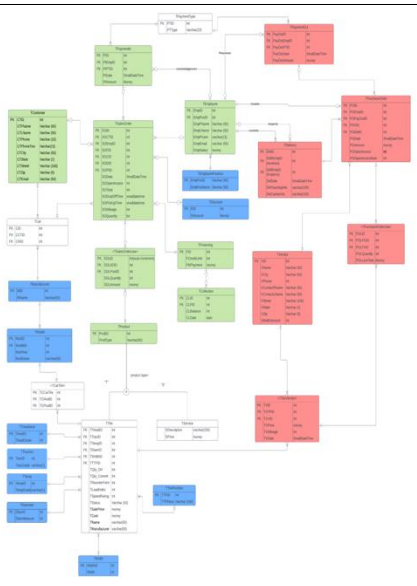
Here is our created ERD. We made some changes compared to the original integrated ERD such as removing the production cycle. We also added some entities and assumptions throughout.

https://lucid.app/lucidchart/46802608-b88f-49ae-ab3a-75e5150ebf20/edit?viewport_loc=-557%2C312%2C4241%2C2113%2CHF0RqB.qO5ix&invitationId=inv_20e5527f-9364-4d72-9e02-4bded376d013



Changes made to generic ERDs

Change #	Original ERD	Updated ERD
Milestone 1: The first change we made was taking out the production cycle. We assumed our company will not be producing any tires, but instead will be buying them from vendors. The second change we made was taking out the Delivery Out reference table. We took this out because we are assuming that we aren't packing or shipping any tires. Our customers will come in to the for the installation of their tires. The third change we made was adding an appointment reference table to our ERD. We are assuming that our customers can only come in for a tire installation through an appointment. This table allows us to keep track of all appointments. The fourth change that was made to our ERD was the addition of the recommended tire reference table. This was added to give our customers an understanding of what brands work best. The fifth change we made was adding a reference table to track our customers' vehicles and the tires used. This will allow us to have data that shows which tires work best and for what types of vehicles. It was also asked to be included in our client meeting. The sixth change we made was adding two relationships from the Employee table to the SalesOrder table. We labeled the two relationships as "Service" and "Process". Service including employees who actually work on the vehicles, and Process being the employees who attend to the customers.		

Milestone 2: The first change we made was adding the subtype/supertype for our service table. We added this because there are different types of services that can be done for the vehicles. Next we created a relationship between the Discount table and the SalesOrderLine table. Pick up time and drop off time were attributes added to the SalesOrder table, which was done to show when cars have been dropped off and when they were available for pickup. Next, we changed the relationship for POLine and connected it to the Tire table. We created a relationship between the Car table and the SalesOrder table, which was not on the original ERD. We added an attribute to our RecommendedTire table named "CarModel" and we also added the attribute "Age" to our Tire table. This was done to keep track of what tire works best on what model, and how old each tire is. A relationship between Vendor to Tire was created to keep track of what vendor makes what tire. A relationship was created between PaymentTerms and SalesOrder. This was originally connected to the Customer table but was moved so we can keep track of which term was used for each sales order.		
Milestone 3: For milestone 3 we rearranged our entire ERD. We made extensive changes throughout. We first started by removing our old super/sub type. We added a new one instead that splits the product table into either service or tire. We added six tire reference tables to help aid tire selection sizes. We added a reference table to sales order to account for financing and collection for unpaid amounts. We separated the car entity allowing for reference tables to account for tire model and tire manufacturer. Our ERD also had extensive relationship changes and cardinality changes. We also added an associative entity named CarTire to track which tires belong to certain cars.		

Logical Design

Logical design helps make the understanding of our entity relationship diagram easier to read. It allows for our diagram to be implemented for various uses. It also helps visualize where data is being included and pulled from. Logical design separates the content from the structure, resulting in an accurate and efficient database. Logical design has various forms or 0NF, 1NF, 2NF, and 3NF. We started by converting all of our entities to 3NF form. This allows for an easier way to understand our entity relationship diagram.

Normalization

Normalization is the relationship of attributes within entities throughout an entity relationship diagram. Normalization helps identify relationships to make understanding of the ERD easier. It is important to the Sonner Tire Company project because employees can clearly see where data is coming from and what data is related to each other. Normal forms are a set of guidelines that help ensure that a database is well-structured and free of data anomalies.

Normalization of the Data Provided by the Client

For the normalization of the data provided by the client we had to make changes for some tables. For example, the car table needed to be separated into different tables to accommodate for the manufacturer, model, and car tire. This step would help us reduce data redundancy and make the data more cohesive. For the manufacturer table, we created MID as the primary key to keep specific vehicle manufacturers separate. For the model table we assigned ModID as the primary key and included the foreign key ModMID as a reference from the manufacturer table. We also added the attributes ModYear and ModName to avoid redundancy. The next table CarTire was included with the assigned primary key TCCarTire, the foreign key TCModID to reference the Model table, and the foreign key TCProdID to reference the Product table and the Tire sub type table. Atomicity was very important, and the data had to be normalized to result in proper normal form of 3NF. The creation of these extra table allowed us to have the data more accessible and reduce redundancies. This allowed us to have 3NF form. The tables added will make it easier for the implementation of the data in Microsoft SQL Server. Below you will find these tables in normalized form with the foreign key constraints.

TManufacturer(MID, MName)
TModel(ModID, ModMID*, ModYear, ModName)
<TCarTire>(TCCarTire, TCModID*, TCProdID*)

Normalized Relations

Underline: Primary Key
Asterisk * : Foreign Key

TCustomer(CTID, CTFName, CTLName, CTPhone, CTPhoneTwo, CTCity, CTState, CTStreet, CTZip, CTEmail)
TCar(CID, CCTID*, CMID*)
    Foreign Key CCTID references TCarTire
    Not Null, On Delete Restrict
    Foreign key CMID references TModel
TManufacturer(MID, MName)
TModel(ModID, ModMID*, ModYear, ModName)
Foreign Key ModMID references TManufacturer
<TCarTire>(TCCarTire, TCModID*, TCProdID*)
Foreign Key TCModID references TModel
Not Null On Delete Restrict
Foreign key TCProdID references TTire
Not Null On Delete Restrict
TProduct(ProdID, ProdType)
TService(ProdID, SDescription, SPrice)
Primary key ProdID references TProduct
TTire(TireID, TTracID*, TTempID,* TDiamID*, TWidthID,* TTTPID*, TQTY_OH, TQTY_Commit, TReorderPoint, TRatio, TLoadIndex, TSpeedRacing, TStatus, TPrice)
    Foreign key TTracID references TTraction
    Not null on delete restrict
    Foreign key TTempID references TTemp
    Not null on delete restrict
    Foreign Key TDiamID references TDiameter
    Not null on delete restrict
    Foreign Key TWidthID references TWidth
    Not null on delete restrict
    Forein key TTTPID references TTirePosition
    Not null on delete restrict
TTraction(TracID, TracGrade)
TWidth(WidthID, Width)
TTreadwear(TreadID, TreadGrade)
TDiameter(DiamID, DiamMeasure)

TTemp(TempID, TempGrade)
TVendor(VID, VName, VCity, VPhone, VContactFName, VContactLName, VStreet, VState, VZip, VBulkDiscount)
TTirePosition(TTPID, TTPDesc)
<TTireVendor>(TVID, TVPTID*,TVVID*, TVPrice, TVMileage, TVDescription, TVDate)
Foreign key TVPTID references TTire
Not null on delete restrict
Foreign key TVVID references TVendor
Not null on delete restrict
TPaymentType (PTID, PTType)
TEmployeePosition(EmpPosID, EmpPosName)
TEmployee(EmpID, EmpPosID*, EmpFName, EmpLName, EmpPhone, EmpEmail, EmpSalary)
Foreign key EmpPosID references TEmployee
Not Null On Delete Restrict
TDeliveryIn(DelID, DelREmpID*, DelIEmpID*, DelDate, DelTrackingInfo, DelCarrierInfo)
Foreign key DelREmpID references TEmployee
Not null on delete restric
Foreign Key DelIEmpID references TEmployee
Not null on delete restrict
TPaymentOut(PayOutID, PayOutEmpID*, PayOutPTID*, PayOutDate, PayOutAmount)
Foreign Key PayOutEmpID references TEmployee
Not Null On Delete Restrict
Foreign Key PayOutPTID references TPaymentType
Not Null On Delete Restrict
TPurchaseOrder(POID, POEmpID*, POPayOutID*, POVID*, PODelID*, PODate, POAmount, POOpenInvoice, POOpenInvoiceNum)
Foreign key POEmpID references TEmployee
Not Null On Delete Restrict
Foreign key POPayOutID references TPaymentOut
Not Null On Delete Restrict
Foreign Key POVID references TVendor
Not Null On Delete Restrict
Foreign key PODelID references TDelivery
Not Null On Delete Restrict
TPurchaseOrderLine(POLID, POLPOID*, POLTVID*, POLQuantity, POLLineTotal)
Foreign key POLPOID references TPurchaseOrder
Not Null On Delete Restrict
Foreign key POLTVID references <TTireVendor>

Not Null On Delete Restrict
TPaymentIN(PIID, PIEmpID*, PIPTID*, PIDate, PIAmount)
Foreign Key PIEmpID references TEmployee
Not null on delete restrict
Foreign Key PIPTID references TPaymentType
Not null on delete restrict
TDiscount (DID, DAmount)
TFinancing (FID, FCreditLimit, FMPayment)
TCollection(CLID, CLFID*, CLBalance, CLDate)
Foreign Key CLFID references TFinancing
Not Null On Delete Restrict
TSalesOrder(SOID, SOCTID*, SOEmpID*, SOFID*, SOCID*, SODID*, SOPIID*, SODate, SOOpenInvoice, SOTotal, SODropOffTime, SOPickUpTim, SOMileage, SOTireQuantity)
Foreign Key SOCTID References TCustomer
Not Null on Delete Restrict
Foreign Key SOEmpID References TEmployee
Not Null on Delete Restrict
Foreign Key SOFID References TEmployee
Not Null on Delete Restrict
Foreign Key SOCID References TCar
Not Null on Delete Restrict
Foreign Key SODID References TDiscount
Not Null on Delete Restrict
Foreign Key SOPIID References TPaymentIn
Not Null on Delete Restrict
TSalesOrderLine (SOLID, SOLSOID*, SOLProdID*, SOLQuantity, SOLAmount)
Foreign Key SOLSOID references TSalesOrder
Not Null On Delete Restrict
Foreign Key SOLProdID references TProduct
Not Null On Delete Restrict

Differences between ERD and Normalized Relations

ERDs and normalized relations are both used to design databases. ERDs are used to visualize the relationships between the entities, where normalized relations refer to the process of reducing redundancy and ensuring well-structured relations. Normalization involves breaking down tables into smaller, more specialized tables, linking them through relationships. This process eliminates data redundancy and helps ensure that the data in the tables are consistent and accurate. Normalized relations are beneficial over ERDs due to improving data integrity, performance, and simplified maintenance.

Referential Integrity

The referential integrity constraint requires that the values used as foreign keys in one table must correspond to the values used as primary keys in another related table. Referential integrity also helps ensure accurate data and consistency throughout all of our data within an entity relationship diagram. Referential integrity helps provide integrity and reliability of a database and the relationships within the entity relationship diagram. It also helps with the accuracy of data retrieval when pulling certain data. It is important because it helps retrieve data from proper entities and the associated foreign key entities.

Physical Design and Implementation

Physical design involves transforming a data model into a database schema, allowing for implementation into database management program. For this project, Microsoft SQL Server Management Studio will be used as the database management program. Implementation is the process of turning a database into a physical database schema with populated fields of data. Physical design and implementation ensure adequate database performance as well efficiency. It also provides database integrity, security, and recoverability.

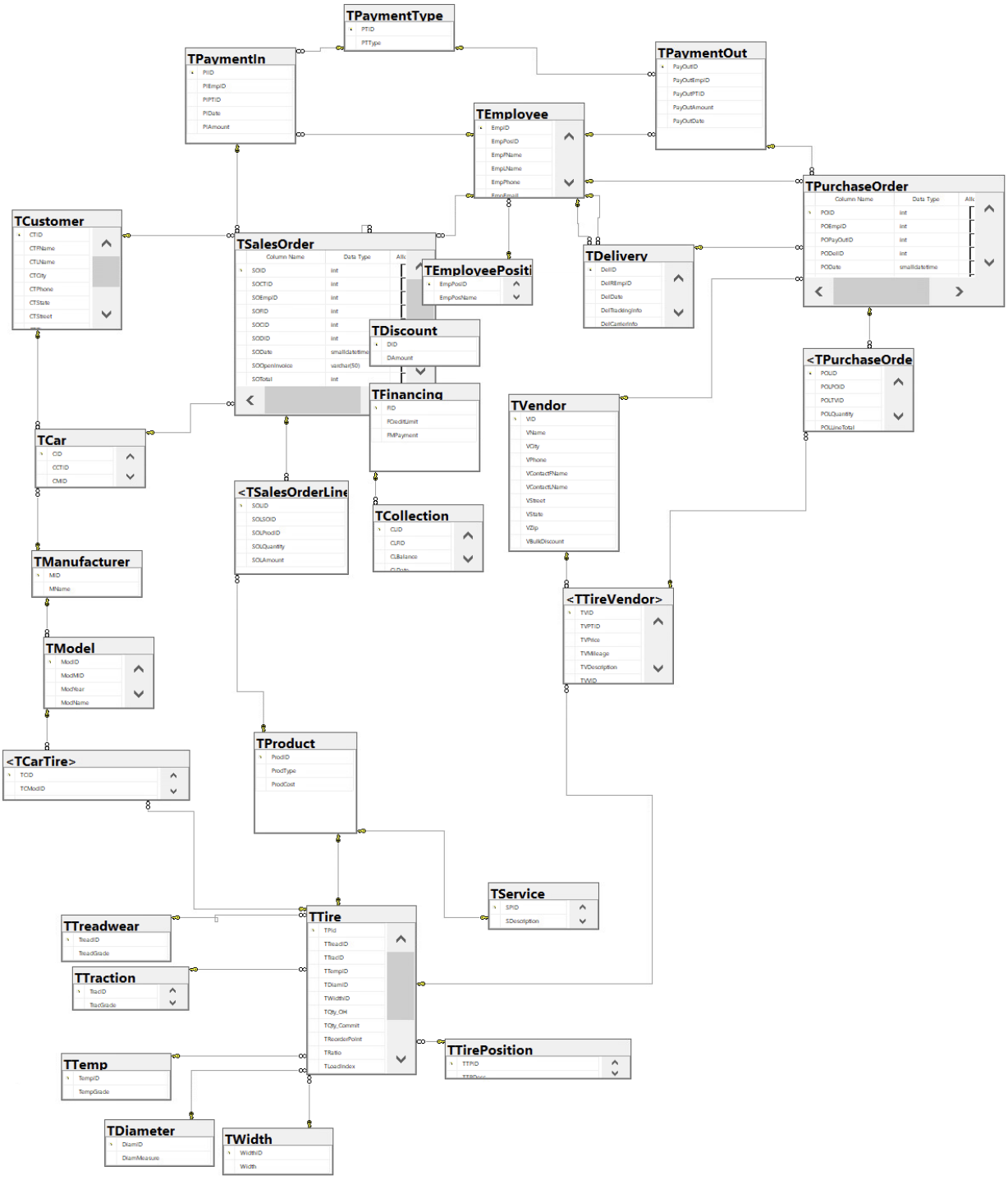
Data Dictionary

A data dictionary works as a reference guide to an ERD. It helps provide accuracy and consistency within the database. Data dictionaries help databases become more effective and function more efficiently. Our group converted our ERD into a data dictionary by taking the mandatory side entity and converting it into spreadsheet format. Within the data dictionary, the entities need to have a table name, field name, data type, data size if it is Varchar, if it is null, if it is a primary key, which table the foreign keys reference, and the sample size. Having this data within the data dictionary allows for implementation to Microsoft SQL Server Management Studio into an ERD allowing for SQL commands to be ran using the database.

Denormalization

Denormalization is the process of adding relationships with redundancy to help with using queries. Tables should be broken up along the "thematic" lines to allow for data to flow more efficiently. Denormalization's main goal is to reduce the number of joins needed to query and respond with the proper information. This can be achieved by combining smaller tables together or having separate tables with redundant information. We implemented this within our database for our super/sub type. We did this by combining the information in the sub-types with the main entity, "Service." We used the code example in class to add our super/sub type properly within Walton. Completing this within the Microsoft SQL management studio allows us to have faster SQL query response times, resulting in the proper information.

Implemented Physical Design



Challenges Faced/Addressed During Implementation

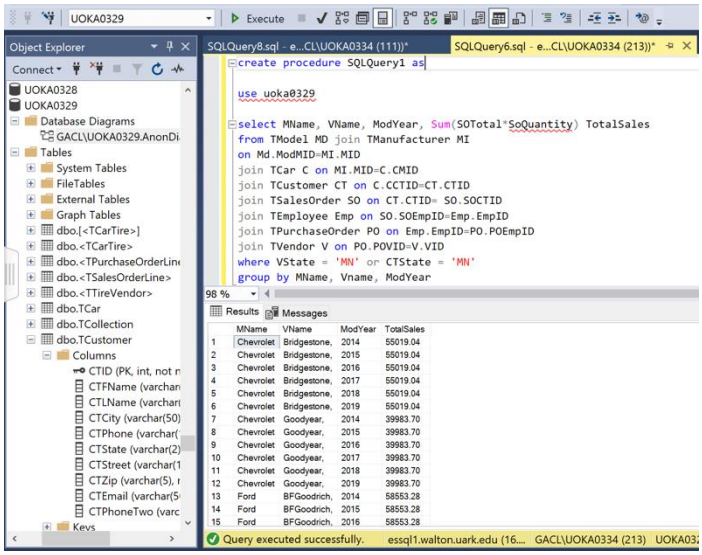
Our biggest challenge we faced was the implementation of our database into Microsoft SQL Server Management Studio. We ran into various errors such that prevented us from implementing the database. We overcame this problem by dividing up the tables between our group and looking at each table one by one. By doing this we were successfully able to implement our database from LucidChart. Our second biggest problem was having the correct relationships appear in Microsoft SQL Management Studio. We addressed this problem by comparing our ERD in LucidChart to the implemented version. By doing this we found our missing relationships. We fixed our issues by using the ALTER commands to change our entity codes to have the right relationships appear. Our last problem we faced as a group was the implementation of dummy data. We were able to overcome this issue by working collectively to find the correct data that needed to be added to the database. With our collective teamwork, we were able to tackle the issues we faced.

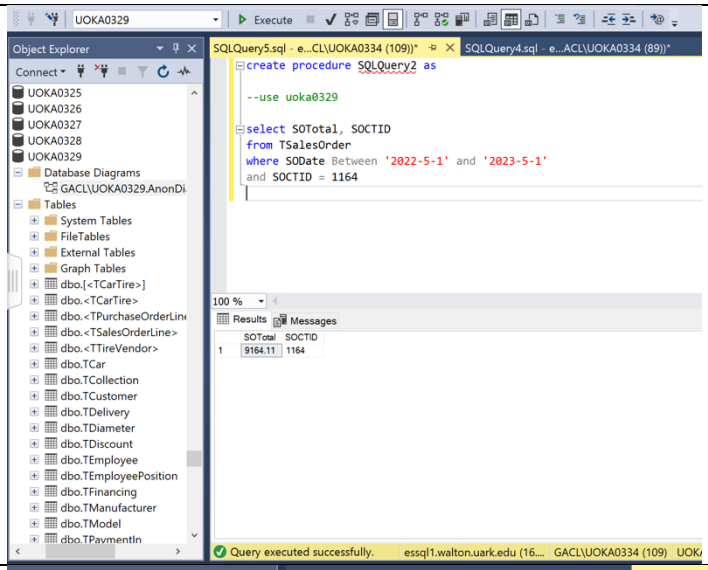
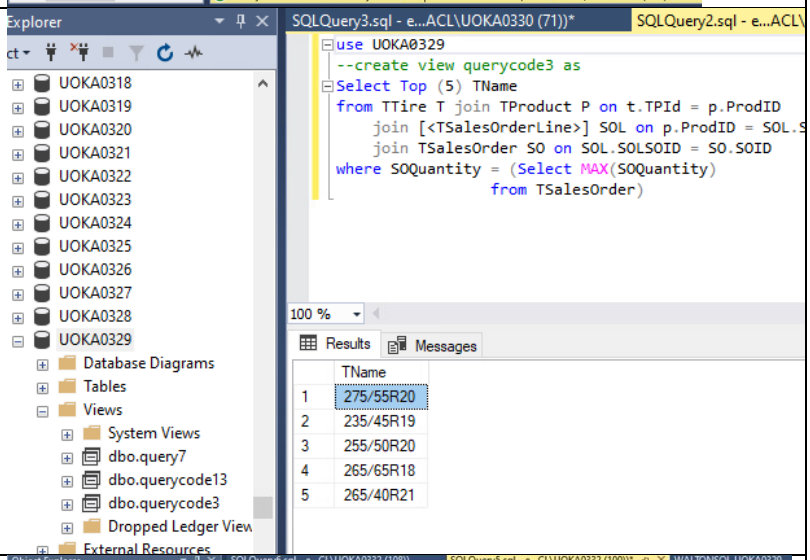
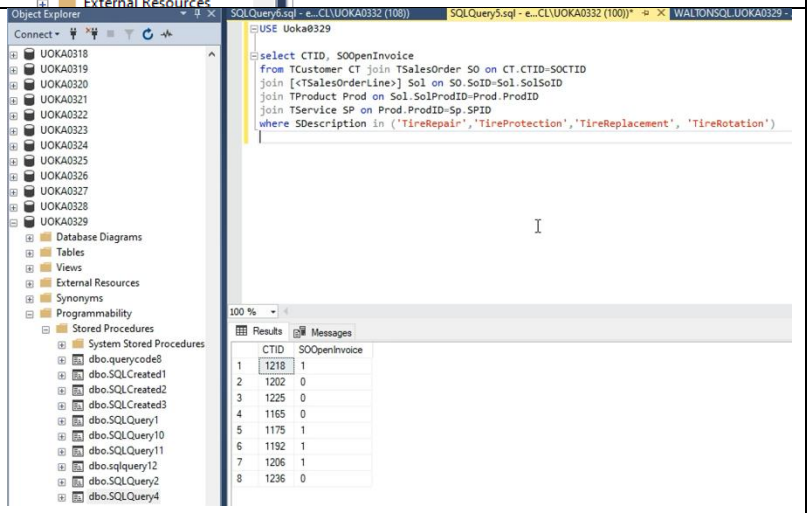
Strengths and Weaknesses Encountered During Implementation

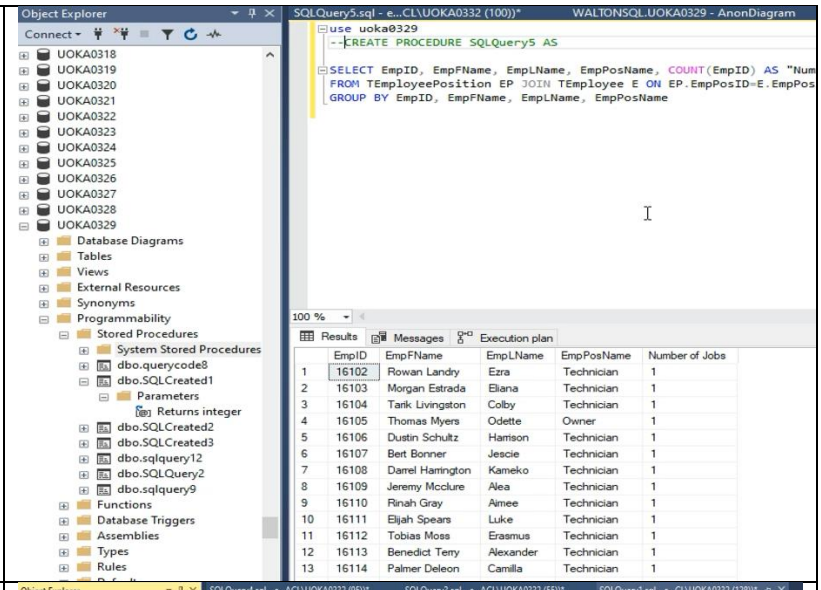
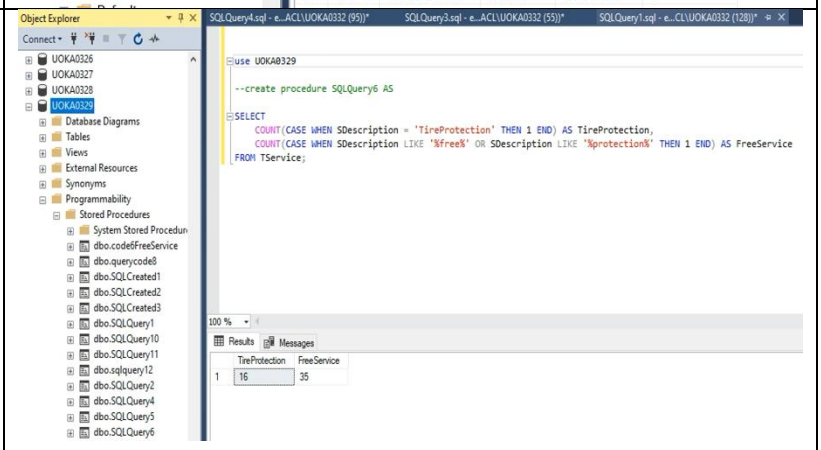
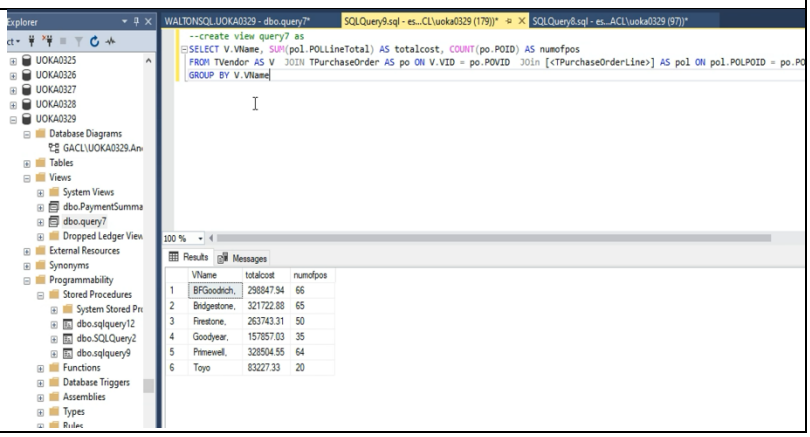
For physical design and implementation, we had strong strengths with the actual implementation and troubleshooting. Once we ran into errors, our team worked together to collectively find the issue and correct it. We also had strengths in building and writing our ERDs. However, our weaknesses resided in the functionality of our entity relationships. We ran into several problems when we added our tables into the diagram within Microsoft SQL Server Management Studio. We successfully had every table added in the diagram, but a few tables were lacking relationships to the corresponding tables. These weaknesses resulted in all of us retyping code, searching for our underlying problem. We also ran into severe trouble when implementing dummy data. We had many tables that would not accept data to be inputted. This took us the longest time out of the physical design portion of the Sonner Tire Company project.

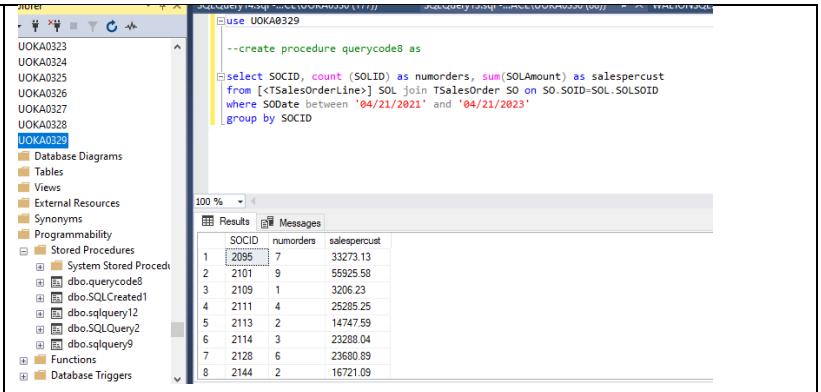
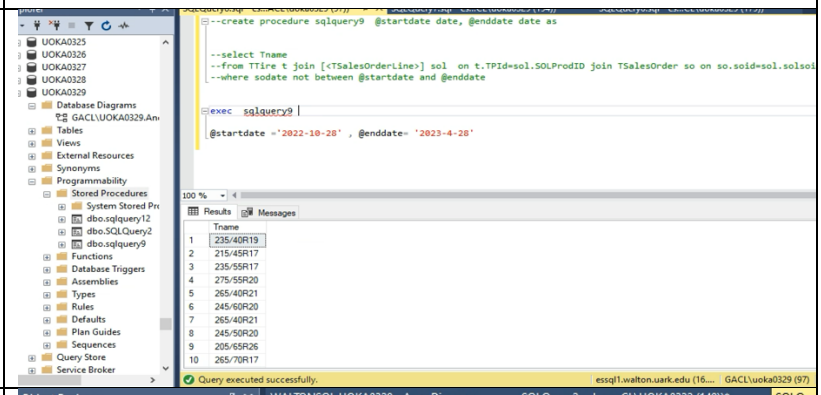
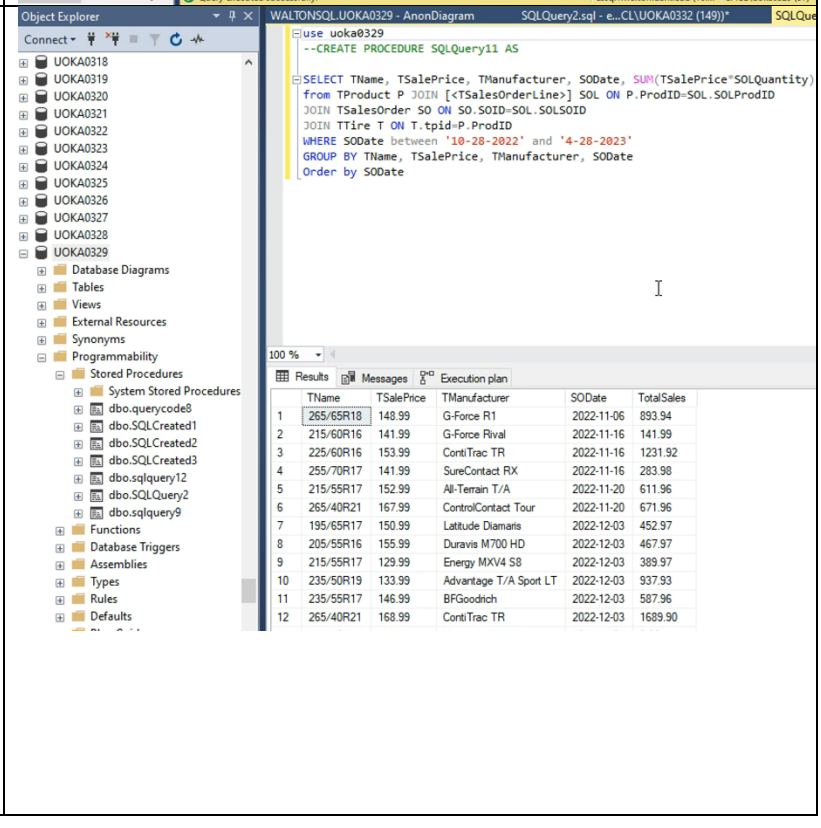
Specific SQL Statements Requested

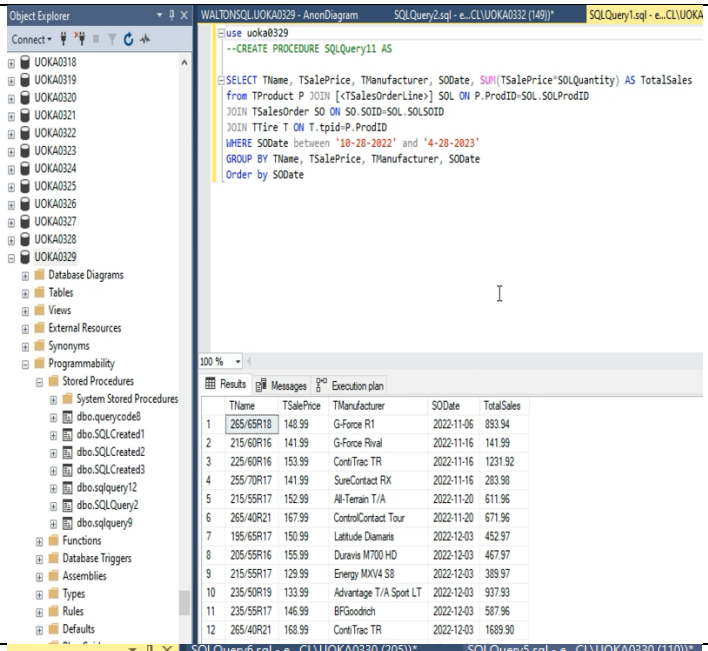
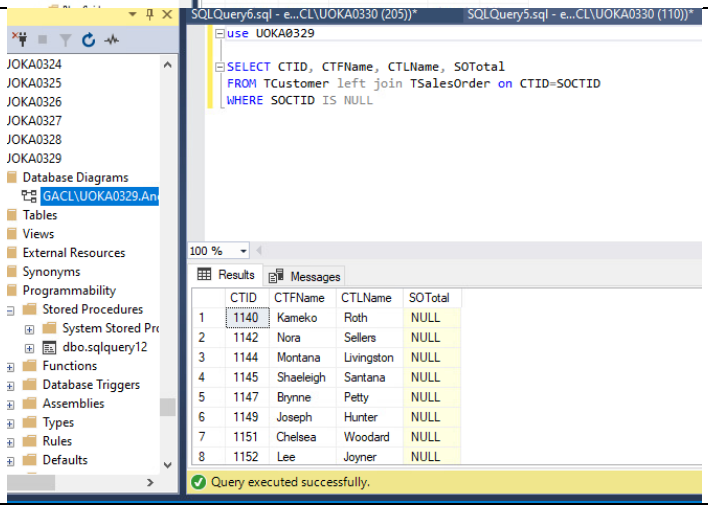
In this section we will explain the specific programs we were asked to execute by Sonner Tire Company in the database. We will also show our additional queries we believe would be useful for the operation. We have included the request asked of us, the SQL code needed to implement the program, and an image of the result of the program.

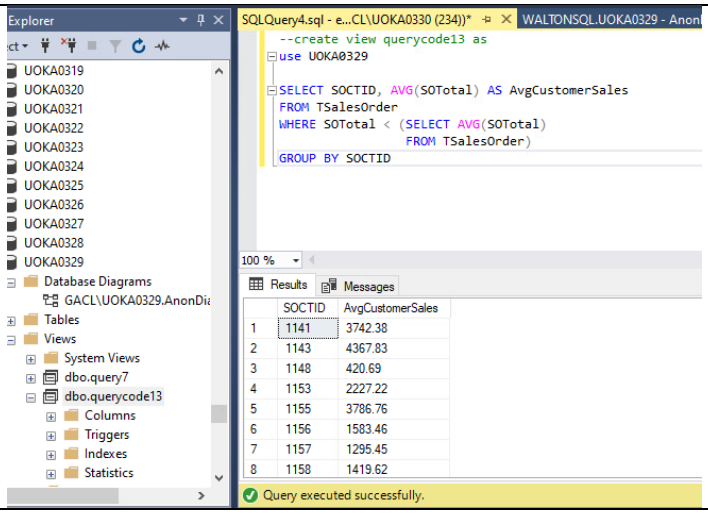
Query#	Question	SQL	Partial Output
1	Total sales (in dollars) by region for a given tire manufacturer and car manufacturer. It would be great if we can specify the car model and year too.	select MName, VName, ModYear, Sum(SOTotal*SoQuantity) TotalSales from TModel MD join TManufacturer MI on Md.ModMID=MI.MID join TCar C on MI.MID=C.CMID join TCustomer CT on C.CCTID=CT.CTID join TSalesOrder SO on CT.CTID= SO.SOCTID join TEmployee Emp on SO.SOEmpID=Emp.EmpID join TPurchaseOrder PO on Emp.EmpID=PO.POEmpID join TVendor V on PO.POVID=V.VID where VState = 'MN' or CTState = 'MN' group by MName, Vname, ModYear	

2	Total sales (in dollars) by a customer in a given year.	select SOTotal, SOCTID from TSalesOrder where SODate Between '2022-5-1' and '2023-5-1' and SOCTID = 1164	
3	The five highest selling tires.	Select Top (5) TName from TTire T join TProduct P on t.TPId = p.ProdID join [<TSalesOrderLine>] SOL on p.ProdID = SOL.SOLPRODID join TSalesOrder SO on SOL.SOLSOID = SO.SOID where SOQuantity = (Select MAX(SOQuantity) from TSalesOrder)	
4	Itemized invoices for jobs for each customer that need to include tires purchased/tire rotation/tire repair/tire protection.	select CTID, SOOpenInvoice from TCustomer CT join TSalesOrder SO on CT.CTID=SOCTID join [<TSalesOrderLine>] Sol on SO.SoID=Sol.SolSoID join TProduct Prod on Sol.SolProdID=Prod.ProdID join TService SP on Prod.ProdID=Sp.SPID where SDescription in ('TireRepair','TireProtection','TireReplacement', 'TireRotation')	

5	The number and type of job performed by each of our employees.	SELECT EmpID, EmpFName, EmpLName, EmpPosName, COUNT(EmpID) AS "Number of Jobs" FROM TEmployeePosition EP JOIN TEmployee E ON EP.EmpPosID=E.EmpPosID GROUP BY EmpID, EmpFName, EmpLName, EmpPosName	
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).	SELECT COUNT(CASE WHEN SDescription = 'TireProtection' THEN 1 END) AS TireProtection, COUNT(CASE WHEN SDescription LIKE '%free%' OR SDescription LIKE '%protection%' THEN 1 END) AS FreeService FROM TService;	
7	The following items for Purchase Orders: manufacturer name, number of POs, total cost.	SELECT V.VName, SUM(pol.POLLineTotal) AS totalcost, COUNT(po.POID) AS numofpos FROM TVendor AS V JOIN TPurchaseOrder AS po ON V.VID = po.POVID JOin [<TPurchaseOrderLine>] AS pol ON pol.POLPOID = po.POID GROUP BY V.VName	

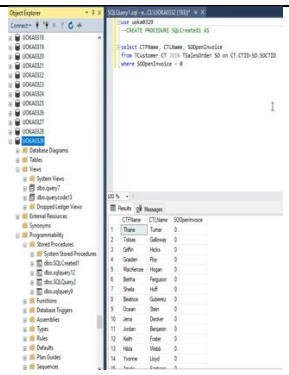
8	Number of orders and total sales per customer in the past 2 years. This report is particularly important as it shows the number of returning customers.	select SOLCustID, count (SOLID) as numorders, sum(SOLTotal) as salespercust from TSalesOrderLine SOL join TSalesOrder SO on SO.SOID=SOL.SOLSOID where SODate between '04/21/2023' and '04/21/2021' group by SOLCustID	
9	List of tires that have not been purchased within the last 6 months.	select Tname from TTire t join [<TSalesOrderLine>] sol on t.TPId=sol.SOLProdID join TSalesOrder so on so.soid=sol.solsoid where sodate not between '2023-4-28' and '2022-10-28'	
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 then	select CTFName, CTLName, FCreditLimit, SODate, SOTotal, FMPayment, (SOTotal-FMPayment) AS Outstanding, COUNT(FMPayment) AS NumOfPmts from TCustomer CT JOIN TSalesOrder SO on CT.CTID=SO.SOCTID JOIN TFinancing F on SO.SOFID=F.FID WHERE SODate between '10-28-2022' and '4-28-2023' GROUP BY CTFName, CTLName, FCreditLimit, SODate, SOTotal, FMPayment ORDER BY Outstanding DESC, SODate DESC	

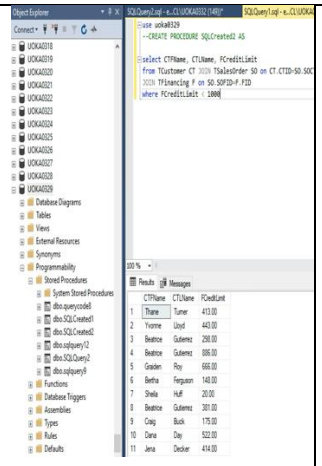
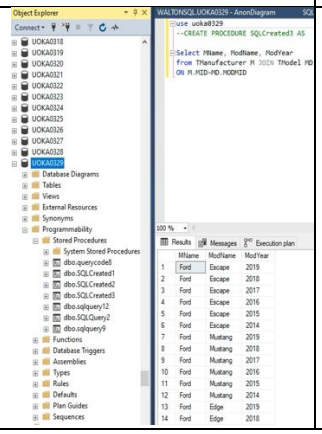
	the largest amount owed.		
11	Total profit per tire type and manufacturer type in the past 6 months.	SELECT TName, TSalePrice, TManufacturer, SODate, SUM(TSalePrice*SOLQuantity) AS TotalSales from TProduct P JOIN [<TSalesOrderLine>] SOL ON P.ProdID=SOL.SOLProdID JOIN TSalesOrder SO ON SO.SOID=SOL.SOLSOID JOIN TTire T ON T.tpid=P.ProdID WHERE SODate between '10-28-2022' and '4-28-2023' GROUP BY TName, TSalePrice, TManufacturer, SODate Order by SODate	
12	List of all customers that have not made a purchase within the last 12 months from the current date.	SELECT CTID, CTFName, CTLName, SOTotal FROM TCustomer left join TSalesOrder on CTID=SOCTID WHERE SOCTID IS NULL	

13	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.	SELECT SOCTID, AVG(SOTotal) AS AvgCustomerSales FROM TSalesOrder WHERE SOTotal < (SELECT AVG(SOTotal) FROM TSalesOrder) GROUP BY SOCTID	

Three Additional Queries

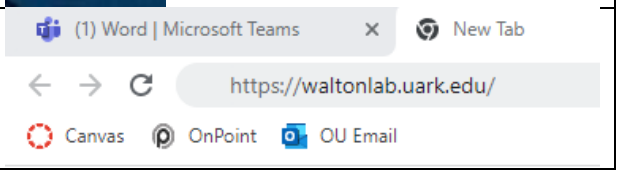
Here are three additional queries we made that we think would be useful to Sonner Tire Company. We think Sonner Tire Company will find these SQL commands useful for making business decisions. We included three queries, the importance of the code, SQL code, the partial output, and our recap of our findings.

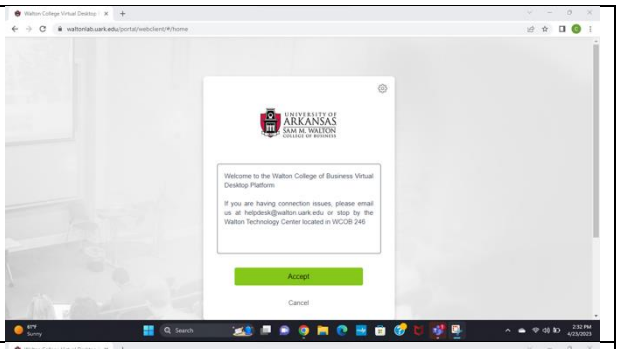
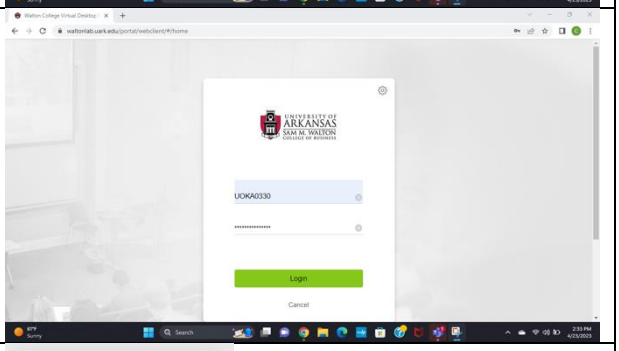
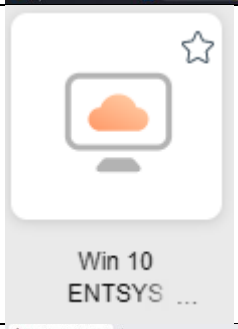
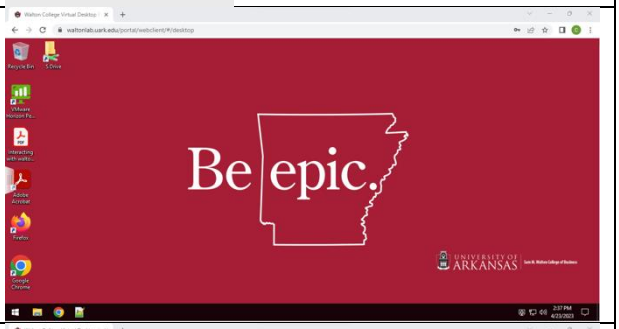
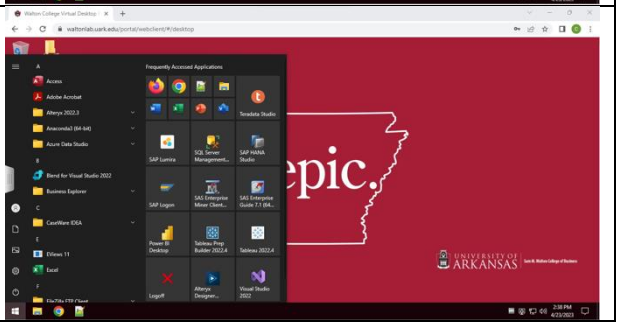
Q#	Question	Why is this important	SQL	Partial Output	Recap of Findings
1	List the customers that currently have an open invoice.	Customer that have not paid and have an open invoice can be sent to collection by Sonner Tire Company	select CTFName, CTLName, SOOpenInvoice from TCustomer CT JOIN TSalesOrder SO on CT.CTID=SO.SOCTID where SOOpenInvoice = 0		The query displays the current customers that have not paid their invoice yet and Sonner can send them to collection as specified in the document.

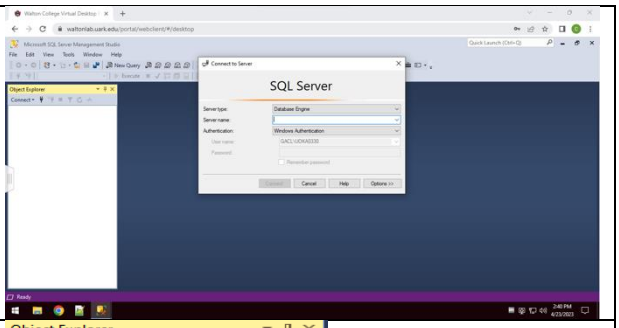
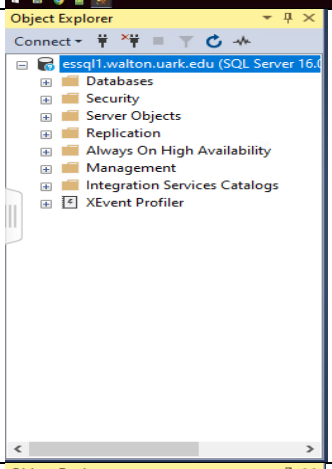
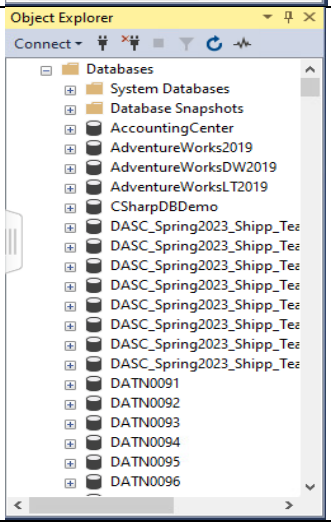
2	List the customers that have been accepted for financing their payments with a credit limit lower than $1000.	Sonner Tire can track the customers that are financing payments with a low credit limit to estimate if the customer will fulfill all payments.	select CTFName, CTLName, FCreditLimit from TCustomer CT JOIN TSalesOrder SO on CT.CTID=SO.SOCTID JOIN TFinancing F on SO.SOFID=F.FID where FCreditLimit < 1000		The query displays the current customers that have applied for financing and have a low credit limit that is under $1000.
3	List all of the various models, years, and various manufacturers of all cars that Sonner Tire services.	Sonner Tire Company can track the model and manufacturers of cars they service to help know which car specific tire needs to be installed or purchased.	Select MName, ModName, ModYear from TManufacturer M JOIN TModel MD ON M.MID=MD.MODMID		The query displays the year, manufacturer, and model of all cars that are serviced at Sonner Tire Company.

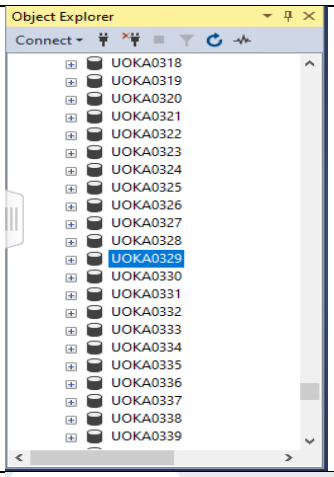
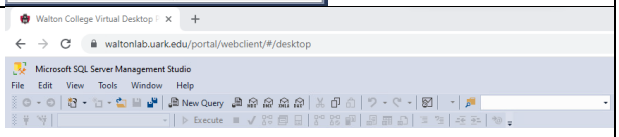
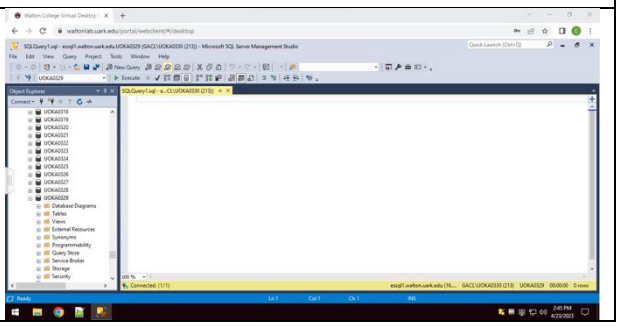
User Documentation

The following is a step-by-step guide to access your database. Please follow the instructions and use the pictures provided to help you understand the process. After the following steps have been completed, you will officially open the database and can start running your queries to read the results.

Step 1: After turning on your computer, open up Google Chrome by double clicking the icon.	
Step 2: While Google Chrome is open, type https://waltonlab.uark.edu/ into your search bar.	

Step 3: Once you type in the previous link, click on the green “Accept” button.	
Step 4: Type in your login information, including the username and password given to you and click the green “Login” button to begin logging in.	
Step 5: After you are logged in you will be brought to a new screen. Once you reach this screen click the “Win 10 ENTSYS” icon.	
Step 6: After step 5 is completed, you will be brought to this screen. Once you are here click on the Windows icon in the bottom left-hand corner of your screen.	
Step 7: After clicking on the Windows icon, this screen will open. When you make it to this screen you will want to click on the “SQL Server Management” icon found under the “Frequently Assessed Applications” tab.	

Step 8: Once you click on “SQL Server Management” the following screen will appear. In the “server name:” box, you will want to enter the following: **essql1.walton.uark.edu** After you type this in, click “Connect”	
Step 9: After connecting, on the left side of your screen click on “Databases” folder to open the databases used on this server.	
Step 10: After you click on databases, the folder will expand. You will want to scroll down until you find your username that you previously typed to login on step 4. For example, you will want to scroll down until you find “UOKA0329”.	

Step 11: Once you find the database that you want to open, double click on it and it will officially open.	
Step 12: After opening the database you will want to open a new query. To open this query, look near the top of your screen and click on the “New Query” icon to open the new query.	
Step 13: After step 12 is complete, your new query will be open, as seen in the screenshot provided. You can now begin running queries and reading the results.	

What We Learned Throughout This Process

Our team, Anonymous, learned a lot throughout the project with Sonner Tire Company. We had to face many challenges within building the ERD, completing logical design, and completing physical design. We faced various challenges but worked together as a team to find solutions for our issues. We learned many great tips to ensure success when implementing a database into database management program. For our project, we used Microsoft SQL Server Management Studio and learned the proper way to include and recall data.

Member Name:	**What you learned:**
Braum Russell	The biggest thing I learned is when implementing databases, keep your datatypes and data dictionary clean and correct. There were multiple times where I tried to insert data in, and over and over it would not enter until I realized the data type that was declared to use was incorrect.
Galavonni Wilson	I learned throughout this process that there was more thought put into this project than anticipated. I realized that data implementation, integrity and accessibility learned from the introductory MIS course holds true to today. Every step of this process is just as important as the one that follows and the one that precedes it. This process was very meticulous and was a thought provoking challenge. I learned that I thoroughly enjoyed this process, and it has solidified my desire to stay within this field.
Mason Matray	After completing this project, I learned many valuable ideas. I first learned how to design an ERD database using the three business cycles. After that, I learned how to implement our ERD into Microsoft SQL Server Management Studio. Along with learning implementation, I also learned troubleshooting tips on how to resolve issues resulting from errors. I also learned how to create dummy data and implement it into our physical design, allowing us to run queries. I also learned great team working skills. Our team came together and worked amazingly well together to completely redesign the entire ERD before final submission.
Coleson Cheek	The first thing I learned is that creating an ERD from scratch is a lot of trial and error but isn't very difficult. The next big thing I learned is that implementing data can be hard if you don't the data dictionary perfect. We found that out and ran into many errors when inserting data into our tables. Lastly, I learned how to use Waltonlabs, which kind of brought the project full circle.
Shohruz Junaidov	The group project to create a database for a tire company was an important learning process that involved making mistakes and improving gradually. Working on this project, I learned about the process of creating a database from scratch. Through this experience, I gained valuable skills and knowledge in using tools and software for designing databases, as well as collaborating with others in a group setting.

Appendix

This section contains extra information that was not needed to be included in the main document. It includes the Anonymous team contract, data dictionary model, and the project management pricing sheet for all three milestones.

Team Contract

Team Members

Name	Email	Phone	Strengths	Availability to Meet
Mason Matray	mason.j.matray@ou.edu	405-464-8033	Organization and SQL	MW after class and TH at 5:00
Shohruz Junaidov	Shohruz.Junaidov-1@ou.edu	646-474-8281	SQL and Communication.	Mon - 4:30 - 6 pm; Wed -9 am - 12 pm; Tues/Thur after 6; Fri - after 3 pm;
Coleson Cheek	coleson.cheek@ou.edu	918-706-4062	Communication, organization, time management	Mon/Tues/Wed after 2:45, Thurs after 4:15, anytime Friday.
Braum Russell	braum.p.russell-1@ou.edu	317-670-2338	SQL, Time management, Team Building	Tues/Thurs after 3 Mon/Wed after 4:30
Galavonni Wilson	micahglwilson@ou.edu	(405) 468-2200	SQL, Time Management, Thorough, Problem Solver	Tues/Thur 9am-1pm, after 4:30pm Fri after 11:30am, All day Sat/Sun

Data Dictionary Model

Here is our data dictionary model for our Sonner Tire Company project. Each table is color coded to differentiate each one. Our data dictionary model includes the table name, field name, data type, size, if it is null, if it is a primary key, references, and the sample size.

Table	Field Name	Data Type	Size	Null	PK?	References	Sample
TCustomer	CTID	Int		Not Null	Y		(1000,1)
	CTFName	Varchar	50	Not Null	N		James
	CTLName	Varchar	50	Not Null	N		Clark
	CTCity	Varchar	50	Not Null	N		Norman
	CTPhone	Varchar	15	Not Null	N		111-111-1111
	CTPhoneTw	Varchar	15	Not Null	N		222-222-2222
	CTState	Varchar	2	Not Null	N		Oklahoma
	CTStreet	Varchar	100	Not Null	N		646 Main Street
	CTZip	Varchar	5	Not Null	N		87463
	CTEmail	Varchar	50	Null Allowe	N		jackdaily@ou.edu
TCar	CID	Int		Not Null	Y		(2000,1)
	CCTID	int		Not Null	N	TCustomer	
	CMID	Int		Not Null	N	TManufacturer	
	COwnerFNa	Varchar	50	Not Null	N		Kate
	COwnerLNa	Varchar	50	Not Null	N		Thompson
TManufactu	MID	int		Not Null	Y		(3000,1)
	MName	Int		Not Null	N		Ford
TModel	ModID	Int		Not Null	Y		(4000,1)
	ModMID	Int		Not Null	N	TModel	
	ModYear	int		Not Null	N		2011
	ModName	Varchar	50	Not Null	N		Escape
TCarTire	TCID	Int		Not Null	Y		(5000,1)
	TCModID	int		Not Null	N	TModel	
	TCProdID	Int		Not Null	N	TProduct	
TProduct	ProdID	Int		Not Null	Y		(6000,1)
	ProdType	Varchar	10	Not Null	N		Tire, Service
TService	SDescriptio	Varchar	200	Not Null	N		TireRotation
	SPrice	Money		Not Null	N		17
TTire	TTreadID	Int		Not Null	N	TTreadwear	
	TTracID	Int		Not Null	N	TTraction	
	TTempID	Int		Not Null	N	TTemp	
	TDiamID	Int		Not Null	N	TDiameter	
	TWidthID	Int		Not Null	N	TWidth	
	TQty_OH	Int		Not Null	N		6
	Tqty_Comm	Int		Not Null	N		5
	TReorderPoi	Int		Not Null	N		16
	TLoadIndex	Int		Not Null	N		120
	TSpeedRati	Int		Null Allowe	N		100
	TStatus	Varchar	10	Not Null	N		Repair
	TSalePrice	money		Not Null	N		500
	TName	Varchar	50	Null Allowe	N		235./45/r17
	Tmanufactu	Varchar	50	Null Allowe	n		Cooper
	Tcost	money		Not Null	n		150
TTreadwear	TreadID	Int		Not Null	Y		(7000,1)
	TreadGrade	Int		Not Null	N		100
TTraction	TracID	Int		Not Null	Y		(8000,1)
	TracGrade	Varchar	2	Not Null	N		AA
TTemp	TempID	Int		Not Null	Y		(9000,1)
	TempGrade	Varchar	1	Not Null	N		B
TDiameter	DiamID	Int		Not Null	Y		(10000,1)
	DiamMeasu	Int		Not Null	N		17
TWidth	WidthID	Int		Not Null	Y		(11000,1)
	Width	Int		Not Null	N		180
TVendor	VID	Int		Not Null	Y		(12000,1)
	VName	Varchar	50	Not Null	N		Goodyear
	VCity	Varchar	50	Not Null	N		Boston
	VPhone	int		Not Null	N		2.382E+09
	VContactFN	Varchar	50	Not Null	N		Bob
	VContactLN	Varchar	50	Not Null	N		Jones
	VStreet	Varchar	100	Not Null	N		3428 Oak Drive
	VState	Varchar	2	Not Null	N		MA
	VZip	Varchar	5	Not Null	N		78943
	VBulkDiscou	Int		Not Null	N		25

TTirePosition	TTPID	int		Not Null	N		(27000,1)	
	TTPDesc	Varchar	100	Not Null	n			
TTireVendor	TVID	int		Not Null	Y		(13000,1)	
	TVPTID	int		Not Null	N	TProduct		
	TVVID	Int		Not Null	N	TVendor		
	TVPrice	Money		Not Null	N			200
	TVMileage	Int		Not Null	N			10000
	TVDate	SmallDateTime		Null Allowed	N			2/3/20
TPaymentType	PTID	Int		Not Null	N		(14000,1)	
	PTType	Varchar	20	Not Null	N			Cash
TEmployeePosition	EmpPosID	Int		Not Null	Y		(15000,1)	
	EmpPosName	Varchar	50	Not Null	N			Recieves
TEmployee	EmpID	int		Not Null	Y		(16000,1)	
	EmpPosID	Int		Not Null	N	TEmployeePosition		
	EmpFName	Varchar	50	Not Null	N			Jerry
	EmpLName	Varchar	50	Not Null	N			Polk
	EmpPhone	varchar	15	Null Allowed	N			8734619024
	EmpSalary	money		Not Null	N			1000
	EmpEmail	Varchar	50	Not Null	N			jerrypolk@sonner.com
TDelivery	DelID	Int		Not Null	Y		(17000,1)	
	DelREmpID	Int		Not Null	N	TEmployee		
	DelIEmpID	Int		Null Allowed	N	TEmployee		
	DelDate	SmallDateTime		Not Null	N			2/7/20
	DelTrackingInfo	Varchar	100	Not Null	N			78930779028
	DelCarrierInfo	Varchar	100	Not Null	N			FedEx
TPaymentOut	PayOutID	int		Not Null	Y		(18000,1)	
	PayOutEmpID	Int		Not Null	N	TEmployee		
	PayOutPTID	int		Not Null	N	TPaymentType		
	PayOutAmount	Money		Not Null	N			897
	PayOutDate	SmallDateTime		Not Null	N			7/3/19
TPurchaseOrder	POID	int		Not Null	Y		(19000,1)	
	POEmpID	int		Not Null	N	TEmployee		
	POPayOutID	int		Not Null	N	TPaymentOut		
	POVID	int		Null Allowed	N	TVendor		
	PODelID	int		Not Null	N	TDelivery		
	PODate	SmallDateTime		Not Null	N			5/2/18
	POAmount	Money		Not Null	N			328
	POOpenInvoice	Int		Not Null	N			0
	POOpenInvoiceNum	int		Not Null	N			327219
TPurchaseOrderLine	POLID	int		Not Null	Y		(20000,1)	
	POLPOID	int		Not Null	N	TPurchaseOrder		
	POLTVID	Int		Not Null	N	TTireVendor		
	POLQuantity	Int		Not Null	N			4
	POLLineTotal	Money		Not Null	N			4
TPaymentIn	PIID	Int		Not Null	Y		(21000,1)	
	PIEmpID	Int		Not Null	N	Temployee		
	PIPTID	Int		Not Null	N	TPaymentType		
	PIDate	SmallDateTime		Not Null	N			6/23/20
	PIAmount	Money		Not Null	N			736
TDiscount	DID	int		Not Null	Y		(22000,1)	
	DAmount	Money		Not Null	N			827
TFinancing	FID	Int		Not Null	Y		(23000,1)	
	FCreditLimit	Int		Not Null	N			800
	FMPayment	money		Not Null	N			200
TCollection	CLID	Int		Not Null	Y		(26000,1)	
	CLFID	int		Not Null	N	TFinancing		
	CLBalance	int		Not Null	N			250
	CLDate	SmallDateTime		Not Null	N			2/9/21
TSalesOrder	SOID	int		Not Null	Y		(24000,1)	
	SOCTID	Int		Not Null	N	TCustomer		
	SOEmpID	int		Not Null	N	Temployee		
	SOFID	int		Not Null	N	Tfinancing		
	SODID	int		Not Null	N	Tdiscount		
	SOCID	int		Not Null	N	Tcar		
	SOPIID	int		Not Null	N	TPaymentIn		
	SODate	SmallDateTime		Null Allowed	N			3/20/21
	SOOpenInvoice	int		Not Null	N			1
	SOTotal	int		Null Allowed	N			100
	SODropOffTime	SmallDateTime		Not Null	N			2:00
	SOPickUpTime	SmallDateTime		Not Null	N			4:00
	SOMileage	int		Not Null	N			67000
	SOTireQuantity	int		Not Null	N			4
TSalesOrderLine	SOLID	Int		Not Null	Y		(25000,1)	
	SOLSOID	Int		Not Null	N	TSalesOrder		
	SOLProdID	int		Not Null	N	TProduct		
	SOLQuantity	int		Not Null	N			4
	SOLAmount	Money		Not Null	N			$1,000

Project Management

Project Start Date	3/20/23	Project End Date	30-Apr-23	Cost (per 60 min)	$25
	Student Name	Duration (Min)	% Complete	Subtotal Minutes	Subtotal Cost
Milestone 1					
Read Case + Prepare Questions for client	Braum Russell	60	30	60	$25
	Shohruz Junaidov	60	30	60	$25
	Mason Matray	60	30	60	$25
	Coleson Cheek	60	30	60	$25
	Galavonni Wilson	60	30	60	$25
Client Meeting	Braum Russell	7	30	7	$3
	Shohruz Junaidov	7	30	7	$3
	Mason Matray	7	30	7	$3
	Coleson Cheek	7	30	7	$3
	Galavonni Wilson	7	30	7	$3
ERD Design	Braum Russell	45	30	45	$19
	Shohruz Junaidov	45	30	45	$19
	Mason Matray	45	30	45	$19
	Coleson Cheek	45	30	45	$19
	Galavonni Wilson	45	30	45	$19
Assumptions	Braum Russell	60	30	60	$25
	Shorhuz Junaidov	60	30	60	$25
	Mason Matray	60	30	60	$25
	Coleson Cheek	60	30	60	$25
	Galavonni Wilson	60	30	60	$25
Write-up preparation	Braum Russell	120	30	120	$50
	Shohruz Junaidov	120	30	120	$50
	Mason Matray	120	30	120	$50
	Coleson Cheek	120	30	120	$50
	Galavonni Wilson	120	30	120	$50
Sub Total				980	$408
Milestone 2					
	Braum Russell	240	60	240	$100
	Shohruz Junaidov	240	60	240	$100
	Mason Matray	240	60	240	$100
	Coleson Cheek	240	60	240	$100
	Galavonni Wilson	240	60	240	$100
Sub Total				1200	$500
Milestone 3					
	Braum Russell	600	90	600	$250
	Shohruz Junaidov	600	90	600	$250
	Mason Matray	600	90	600	$250
	Coleson Cheek	600	90	600	$250
	Galavonni Wilson	600	90	600	$250
Sub Total				3000	$1,250
Final Submission					
	Braum Russell	1800	100	1800	$750
	Shohruz Junaidov	1800	100	1800	$750
	Mason Matray	1800	100	1800	$750
	Coleson Cheek	1800	100	1800	$750
	Galavonni Wilson	1800	100	1800	$750
Sub Total				9000	$3,750
			Total	14180	$5,908